MADHA
Expertise | Empathy | Excellence
ENGINEERING COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COMMON FOR: DEPARTMENT OF INFORMATION TECHNOLOGY

CS8691 – ARTIFICIAL INTELLIGENCE

R – 2017

LECTURE NOTES

UNIT I

INTRODUCTION TO Al AND PRODUCTION SYSTEMS

CHAPTER - 1

What is Artificial Intelligence?

1. INTELLIGENCE

❖ The capacity to learn and solve problems.

❖ In particular,

- the ability to solve novel problems (i.e solve new problems)
- the ability to act rationally (i.e act based on reason)
- the ability to act like humans

1.1 What is involved in intelligence?

• Ability to interact with the real world
– to perceive, understand, and act
– e.g., speech recognition and understanding and synthesis
– e.g., image understanding
– e.g., ability to take actions, have an effect

• Reasoning and Planning
– modeling the external world, given input
– solving new problems, planning, and making decisions
– ability to deal with unexpected problems, uncertainties

• Learning and Adaptation
– we are continuously learning and adapting
– our internal models are always being ‒updated‖
• e.g., a baby learning to categorize and recognize animals

2. ARTIFICIAL INTELLIGENCE

It is the study of how to make computers do things at which, at the moment, people are better.

The term AI is defined by each author in own ways which falls into 4 categories

1. The system that think like humans.
2. System that act like humans.
3. Systems that think rationally.
4. Systems that act rationally.

2.1 SOME DEFINITIONS OF AI

- **Building systems that think like humans**

  ‒The exciting new effort to make computers think … machines with minds, in the full and literal sense‖ -- Haugeland, 1985

  ‒The automation of activities that we associate with human thinking, … such as decision-making, problem solving, learning, …‖ -- Bellman, 1978

- **Building systems that act like humans**

  ‒The art of creating machines that perform functions that require intelligence when performed by people‖ -- Kurzweil, 1990

  ‒The study of how to make computers do things at which, at the moment, people are better‖ -- Rich and Knight, 1991

- **Building systems that think rationally**

  ‒The study of mental faculties through the use of computational models‖ -- Charniak and McDermott, 1985

  ‒The study of the computations that make it possible to perceive, reason, and act‖ -Winston, 1992

- **Building systems that act rationally**

  ‒A field of study that seeks to explain and emulate intelligent behavior in terms of computational processes‖ -- Schalkoff, 1990

  ‒The branch of computer science that is concerned with the automation of intelligent behavior‖ -- Luger and Stubblefield, 1993

2.1.1. Acting Humanly: The Turing Test Approach

- ❖ Test proposed by Alan Turing in 1950
- ❖ The computer is asked questions by a human interrogator.

The computer passes the test if a human interrogator, after posing some written questions, cannot tell whether the written responses come from a person or not. Programming a computer to pass, the computer need to possess the following capabilities:

- ❖ **Natural language processing** to enable it to communicate successfully in English.
- ❖ **Knowledge representation** to store what it knows or hears
- ❖ **Automated reasoning** to use the stored information to answer questions and to draw new conclusions.
- ❖ **Machine learning** to adapt to new circumstances and to detect and extrapolate patterns.

To pass the complete Turing Test, the computer will need

- ❖ Computer vision to perceive the objects, and
- ❖ Robotics to manipulate objects and move about.

2.1.2 Thinking humanly: The cognitive modeling approach

We need to get inside actual working of the human mind:

(a) Through introspection – trying to capture our own thoughts as they go by;

(b) Through psychological experiments

Allen Newell and Herbert Simon, who developed GPS, the ―General Problem Solver‖ tried to trace the reasoning steps to traces of human subjects solving the same problems. The interdisciplinary field of cognitive science brings together computer models from AI and experimental techniques from psychology to try to construct precise and testable theories of the workings of the human mind

2.1.3 Thinking rationally : The "laws of thought approach"

The Greek philosopher Aristotle was one of the first to attempt to codify ―right thinking that is irrefutable (ie. Impossible to deny) reasoning processes. His syllogism provided patterns for argument structures that always yielded correct conclusions when given correct premises—for example, Socrates is a man; all men are mortal; therefore Socrates is mortal.‖. These laws of thought were supposed to govern the operation of the mind; their study initiated a field called logic.

2.1.4 Acting rationally : The rational agent approach

An agent is something that acts. Computer agents are not mere programs, but they are expected to have the following attributes also: (a) operating under autonomous control, (b) perceiving their environment, (c) persisting over a prolonged time period, (e) adapting to change. A rational agent is one that acts so as to achieve the best outcome.

3. HISTORY OF AI

• 1943: early beginnings

– McCulloch & Pitts: Boolean circuit model of brain

• 1950: Turing

– Turing's "Computing Machinery and Intelligence–

• 1956: birth of AI

– Dartmouth meeting: "Artificial Intelligence–name adopted

• 1950s: initial promise

– Early AI programs, including

– Samuel's checkers program

– Newell & Simon's Logic Theorist

• 1955-65: –great enthusiasm‖

– Newell and Simon: GPS, general problem solver

– Gelertner: Geometry Theorem Prover

– McCarthy: invention of LISP

• 1966—73: Reality dawns

– Realization that many AI problems are intractable

– Limitations of existing neural network methods identified

• Neural network research almost disappears

• 1969—85: Adding domain knowledge

– Development of knowledge-based systems

– Success of rule-based expert systems,

• E.g., DENDRAL, MYCIN

• But were brittle and did not scale well in practice

• 1986-- Rise of machine learning

– Neural networks return to popularity

– Major advances in machine learning algorithms and applications

• 1990-- Role of uncertainty

– Bayesian networks as a knowledge representation framework

• 1995--AI as Science

– Integration of learning, reasoning, knowledge representation

– AI methods used in vision, language, data mining, etc

3.1 AI Technique

AI technique is a method that exploits knowledge that should be represented in such a way that:

• **The knowledge captures generalizations**. In other words, it is not necessary to represent separately each individual situation. Instead, situations that share important properties are grouped together. If knowledge does not have this property, inordinate amounts of memory and updating will be required. So we usually call something without this property "data" rather than knowledge.

• **It can be understood by people who must provide it.** Although for many programs, the bulk of the data can be acquired automatically (for example, by taking readings from a variety of instruments), in many AI domains, most of the knowledge a program has must ultimately be provided by people in terms they understand.

• **It can easily be modified to correct errors** and to reflect changes in the world and in our world view.

• **It can be used in a great many situations even if it is not totally accurate or complete.**

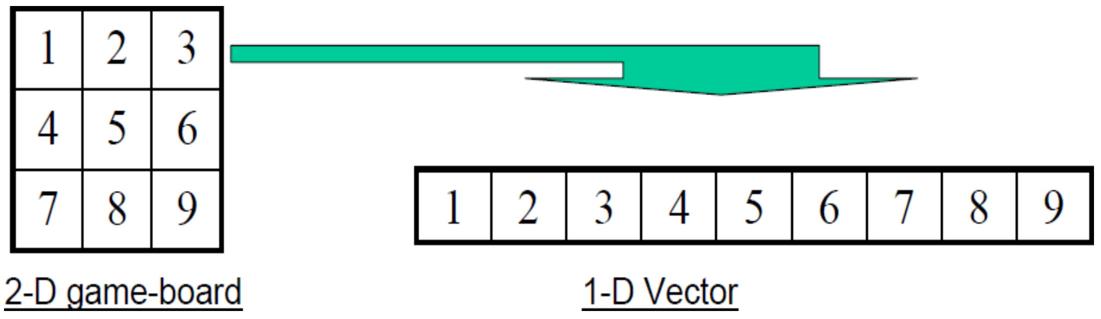
• **It can be used to help overcome its own sheer bulk by helping to narrow the range of possibilities that must usually be considered.**

Although AI techniques must be designed in keeping with these constraints imposed by AI problems, there is some degree of independence between problems and problem-solving techniques. It is possible to solve AI problems without using AI techniques (although, as we suggested above, those solutions are not likely to be very good).

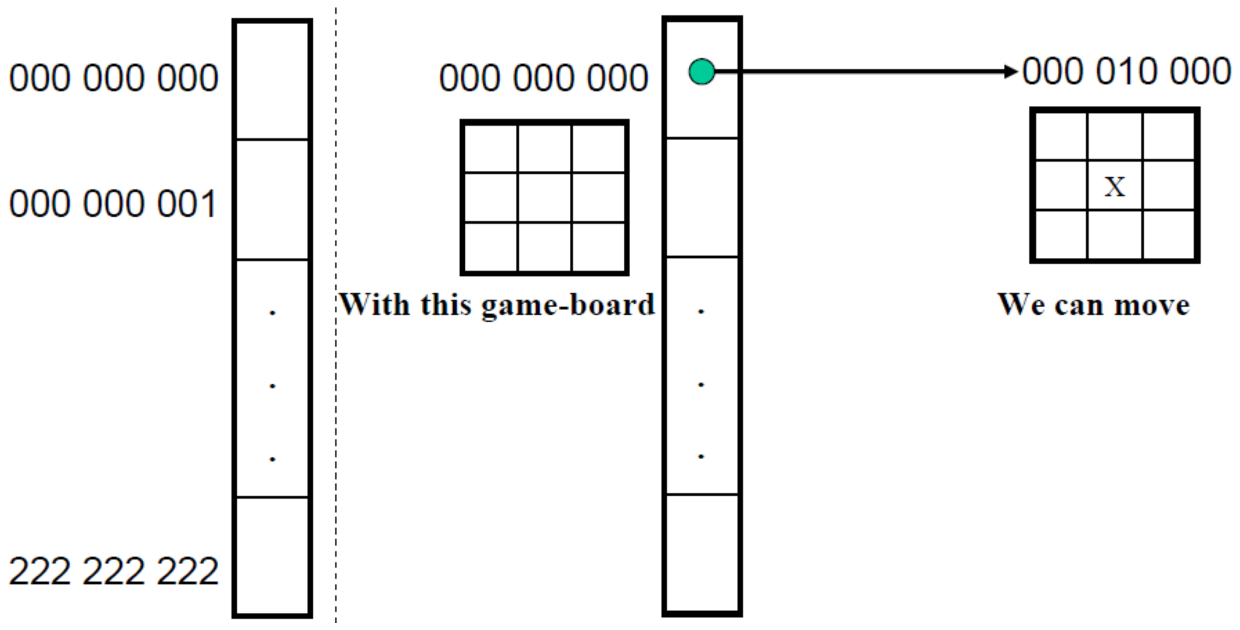
Tic-Tac-Toe

Solution 1

- **Data Structure:**



- **Elements of vector:**
  - ✓ 0 : Empty
  - ✓ 1 : X
  - ✓ 2: O
- The vector is a ternary number
- Store inside the program a move-table (lookup table):
- #Elements in the table: 19683 (=3^9)
- Element = A vector which describes the most suitable move from the current game-board



Algorithm

1. View the vector as a ternary number. Convert it to a decimal number.

2. Use the computed number as an index into Move-Table and access the vector stored there.

3. Set the new board to that vector.

Comments

1. A lot of space to store the Move-Table.

2. A lot of work to specify all the entries in the Move-Table.

3. Difficult to extend.

Solution 2

Data Structure

- Use vector, called board, as Solution 1
- However, elements of the vector:
  - ✓ : Empty
  - ✓ : X
  - ✓ : O
- Turn of move: indexed by integer
  - ✓ 1,2,3, etc.

Function Library:

1. Make2:

- ➢ Return a location on a game-board.

```
IF (board[5] = 2)
RETURN 5; //the center cell.
ELSE
RETURN any cell that is not at the board‘s corner;
// (cell: 2,4,6,8)
```

1	2	3
4	5	6
7	8	9

- ➢ Let P represent for X or O
- ➢ can_win(P) :
  - ✓ P has filled already at least two cells on a straight line (horizontal, vertical, or diagonal)
- ➢ cannot_win(P) = NOT(can_win(P))

2. Posswin(P):

```
    IF (cannot_win(P))
    RETURN 0;
    ELSE
    RETURN index to the empty cell on the line of
    can_win(P)
```

- ➢ Let odd numbers are turns of X
- ➢ Let even numbers are turns of O

```
3. Go(n): make a move.
    IF odd(turn) THEN            // for X
        Board[n] = 3
    ELSE                         // for O
        Board[n] = 5
    turn = turn + 1
```

Algorithm:

```
1. Turn = 1: (X moves)
    Go(1) //make a move at the left-top cell
2. Turn = 2: (O moves)
    IF board[5] is empty THEN
        Go(5)
    ELSE
        Go(1)
3. Turn = 3: (X moves)
    IF board[9] is empty THEN
        Go(9)
    ELSE
        Go(3).
4. Turn = 4: (O moves)
    IF Posswin(X) <> 0 THEN
        Go(Posswin(X))
        //Prevent the opponent to win
    ELSE Go(Make2)
5. Turn = 5: (X moves)
    IF Posswin(X) <> 0 THEN
        Go(Posswin(X))
        //Win for X.
    ELSE IF Posswin(O) <> THEN
```

```
        Go(Posswin(O))
        //Prevent the opponent to win
    ELSE IF board[7] is empty THEN
        Go(7)
    ELSE Go(3).
```

Comments:

1. Not efficient in time, as it has to check several conditions before making each move.

2. Easier to understand the program‘s strategy.

3. Hard to generalize.

4. Checking for a possible win is quicker.

5. Human finds the row-scan approach easier, while computer finds the number-counting approach more efficient.

Solution 3

Data Structure

1. Game-board: Use vector as described for the above program

2. List:

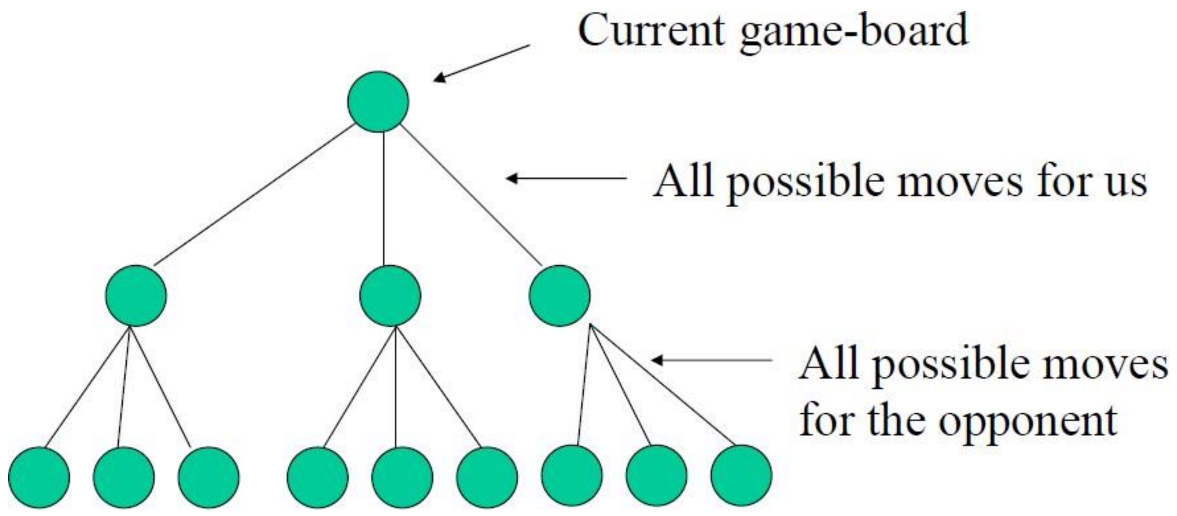
- ✓ Contains possible game-boards generated from the current game-board
- ✓ Each the game-board is augmented a score indicating the possibility of victory of the current turn

Algorithm:

1. If it is a win, give it the highest rating.

2. Otherwise, consider all the moves the opponent could make next. Assume the opponent will make the move that is worst for us. Assign the rating of that move to the current node.

3. The best node is then the one with the highest rating.


Current game-board
All possible moves for us
All possible moves for the opponent

Comments:

1. Require much more time to consider all possible moves.

2. Could be extended to handle more complicated games.

CHAPTER - 2

PROBLEMS, PROBLEM SPACES AND SEARCH

2. FORMULATING PROBLEMS

Problem formulation is the process of deciding what actions and states to consider, given a goal

$$\begin{tikzcd}
\text{Formulate Goal, Formulate problem} \arrow[d] \\
\text{Search} \arrow[d] \\
\text{Execute}
\end{tikzcd}$$

2.1 WELL-DEFINED PROBLEMS AND SOLUTIONS

A problem can be defined formally by four components:

1. Initial state
2. Successor function
3. Goal test
4. Path cost

1. **Initial State**

The starting state which agent knows itself.

1. **Successor Function**

- A description of the possible actions available to the agent.
- State x, successor – FN (x) returns a set of < action, successor> ordered pairs, where each action is a legal action in a state x and each successor is a state that can be reached from x by applying that action.

2.1 State Space

The set of all possible states reachable from the initial state by any sequence of actions. The initial state and successor function defines the state space. The state space forms a graph in which nodes are state and axis between the nodes are action.

2.2 Path

A path in the state space is a sequence of state connected by a sequence of actions.

2. **Goal Test**

Test to determine whether the given state is the goal state. If there is an explicit set of possible goal states, then we can whether any one of the goal state is reached or not.

Example : In chess, the goal is to reach a state called ―checkmate‖ where the opponent‘s king is under attack and can‘t escape.

3. **Path cost**

A function that assigns a numeric cost to each path. The cost of a path can be described as the sum of the costs of the individual actions along that path.

Step cost of taking an action ‗a‘ to go from one state ‗x‘ to state ‗y‘ is denoted by C(x,a,y)

C-Cost , x,y- states , Action , Step costs are non-negative

These 4 elements are gathered into a data structure that is given as input to problem solving algorithm. A solution quality is measured by path cost function. An optimal solution has lowest path cost among all solutions.

Total cost = Path cost + Search cost

Example: Route finding problem

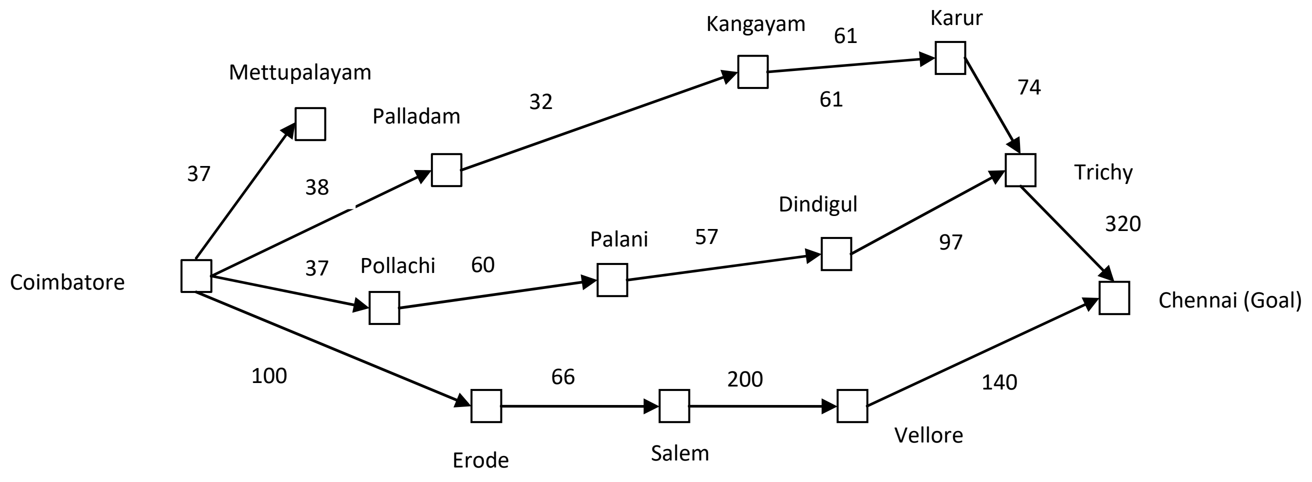


Fig: 1 Route Finding Problem

Initial State: In (Coimbatore)

Successor Function: {< Go (Pollachi), In (Pollachi)>

< Go (Erode), In (Erode)>

< Go (Palladam), In (Palladam)>

< Go (Mettupalayam), In (Mettupalayam)>}

Goal Test: In (Chennai)

Path Cost: {(In (Coimbatore),}

{Go (Erode),} = 100 [kilometers]

{In (Erode)}

Path cost = 100 + 66 + 200 + 140 = 506

2.2 TOY PROBLEM

Example-1 : Vacuum World

Problem Formulation

- States
  - – 2 x 22 = 8 states

– Formula n2n states

• Initial State

– Any one of 8 states

• Successor Function

– Legal states that result from three actions (Left, Right, Absorb)

• Goal Test

– All squares are clean

• Path Cost

– Number of steps (each step costs a value of 1)

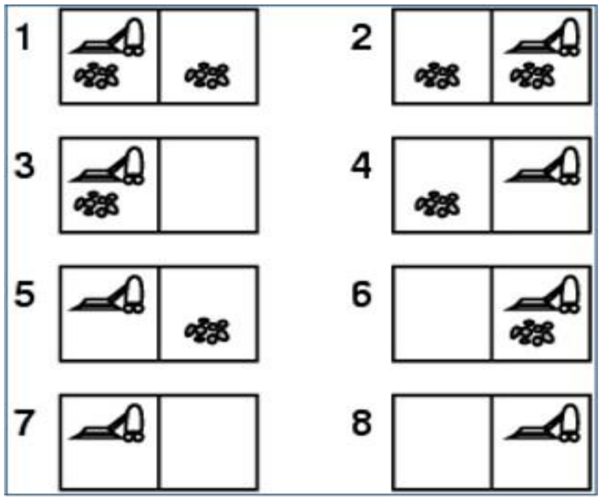


Fig 1.2 Vacuum World

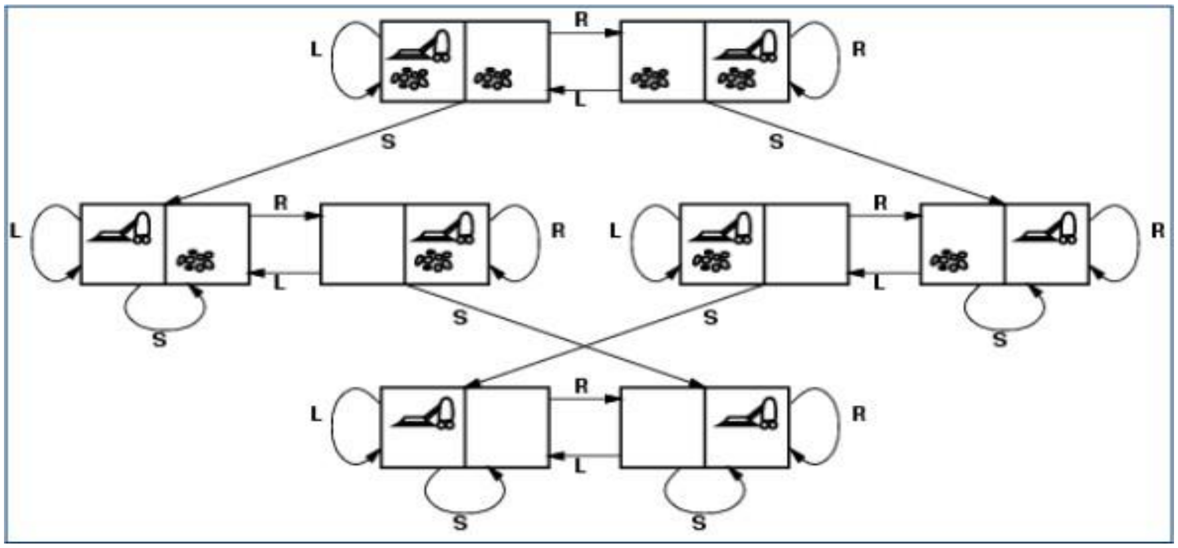


Fig: 1.3 State Space for the Vacuum World

State Space for the Vacuum World

Labels on Arcs denote L: Left, R: Right, S: Suck

Example 2: Playing chess

Initial State: Described as an 8 X 8 array where each positions contains a symbol standing for the appropriate piece in the official chess position.

Successor function: The legal states that results from set of rules.

They can be described easily by as a set of rules consisting of two parts: a left side that serves as a pattern to be matched against the current board position and a right side that describes the changes to be made to the board position to reflect the move. An example is shown in the following figure.

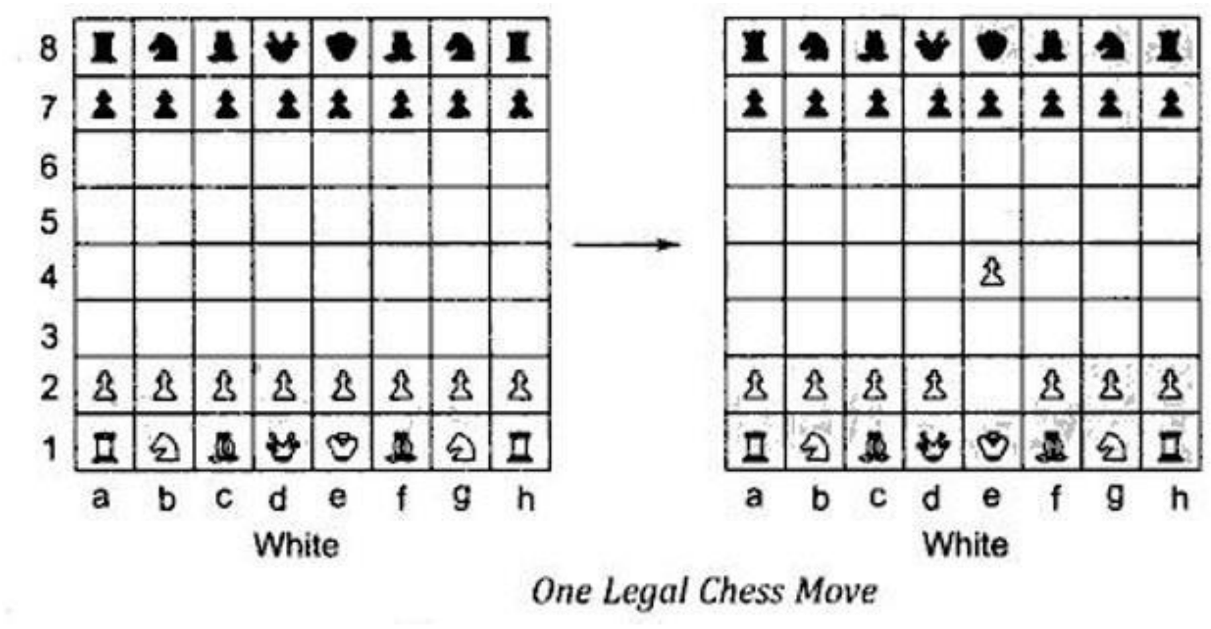


Fig 1.4:The legal states that results from set of rules

However if we write rules like the one above, we have to write a very large number of them since there has to be a separate set of rule for each of them roughly 10^{120} possible board positions.

Practical difficulties to implement large number of rules,

1. It will take too long to implement large number of rules and could not be done without mistakes.
2. No program could easily handle all those rules and storing it possess serious difficulties.

In order to minimize such problems, we have to write rules describing the legal moves in as a general way as possible. The following is the way to describe the chess moves.

Current Position

While pawn at square (e, 2), AND Square (e, 3) is empty, AND Square (e , 4) is empty.

Changing Board Position

Move pawn from Square (e, 2) to Square (e , 4) .

Some of the problems that fall within the scope of AI and the kinds of techniques will be useful to solve these problems.

GoalTest
Any position in which the opponent does not have a legal move and his or her king is under attack.

Example: 3 Water Jug Problem

A Water Jug Problem: You are given two jugs, a 4-gallon one and a 3-gallon one, a pump which has unlimited water which you can use to fill the jug, and the ground on which water may be poured. Neither jug has any measuring markings on it. How can you get exactly 2 gallons of water in the 4-gallon jug?

State: (x, y) x= 0, 1, 2, 3, or 4 y= 0, 1, 2, 3

x represents quantity of water in 4-gallon jug and y represents quantity of water in 3-gallon jug.

•**Start state**: (0, 0).

•**Goal state:** (2, n) for any n. Attempting to end up in a goal state.(since the problem doesn‘t specify the quantity of water in 3-gallon jug)

1. (x, y) If x <4	→(4, y)	Fill the 4-gallon jug
2. (x, y) If y <3	→(x, 3)	Fill the 3-gallon jug
3. (x, y) If x >0	→(x −d, y)	Pour some water out of the 4-gallon jug
4. (x, y) If y >0	→(x, y −d)	Pour some water out of the 3-gallon jug

5. (x, y) If x >0	→(0, y)	Empty the 4-gallon jug on the ground
6. (x, y) If y >0	→(x, 0)	Empty the 3-gallon jug on the ground
7. (x, y) If x +y ≥4,y >0	→(4, y −(4 −x))	Pour water from the 3-gallon jug into the 4-gallon jug until the 4-gallon jug is full
8. (x, y) If x +y ≥3,x >0	→(x −(3 −y), 3)	Pour water from the 4-gallon jug into the 3-gallon jug until the 3-gallon jug is full
9. (x, y) If x +y ≤4,y >0	→(x +y, 0)	Pour all the water from the 3-gallon jug into the 4-gallon jug
10. (x, y) If x +y ≤3,x >0	→(0, x +y)	Pour all the water from the 4-gallon jug into the 3-gallon jug
11. (0, 2)	→(2, 0)	Pour the 2 gallons from the 3-gallon Jug into the 4-gallon jug
12. (2, y)	→(0, y)	Empty the 2 gallons in the 4-gallon Jug on the ground

Production rules for the water jug problem

Trace of steps involved in solving the water jug problem First solution

Number of Steps	**Rules applied**	**4-g jug**	**3-g jug**
1	Initial state	0	0
2	R2 {Fill 3-g jug}	0	3
3	R7 {Pour all water from 3 to 4-g jug}	3	0
4	R2 {Fill 3-g jug}	3	3
5	R5 {Pour from 3 to 4-g jug until it is full}	4	2
6	R3 {Empty 4-gallon jug}	0	2
7	R7 {Pour all water from 3 to 4-g jug}	2	0

Goal State

Second Solution

Number of Steps	Rules applied	4-g jug	3-g jug
1	Initial state	0	0
2	R1 {Fill 4-gallon jug}	4	0
3	R6 {Pour from 4 to 3-g jug until it is full}	1	3
4	R4 {Empty 3-gallon jug}	1	0
5	R8 {Pour all water from 4 to 3-gallon jug}	0	1
6	R1 {Fill 4-gallon jug}	4	1
7	R6 {Pour from 4 to 3-g jug until it is full}	2	3
8	R4 {Empty 3-gallon jug}	2	0

Goal State

Example - 5 8-puzzle Problem

The 8-puzzle problem consists of a 3 x 3 board with eight numbered tiles and a blank space. A tile adjacent to the blank space can slide into the space. The object is to reach a specified goal state.

States: A state description specifies the location of each of the eight tiles and the blank in one of the nine squares.

Initial state: Any state can be designated as the initial state.

Successor function: This generates the legal states that result from trying the four actions (blank moves Left, Right, Up, or Down).

Goal test: This checks whether the state matches the goal configuration (Other goal configurations are possible.)

Path cost: Each step costs 1, so the path cost is the number of steps in the path.

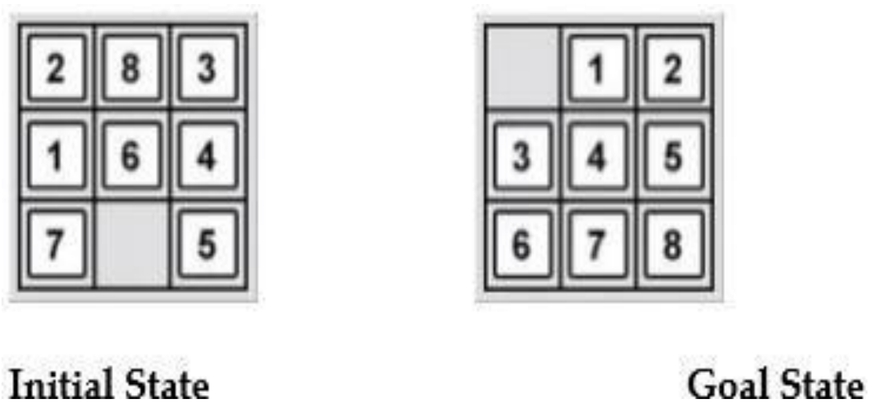


Fig 1.5 8 Puzzle Problem

Exampe-6 8-queens problem

The goal of the 8-queens problem is to place eight queens on a chessboard such that no queen attacks any other. (A queen attacks any piece in the same row, column or diagonal.

States: Any arrangement of 0 to 8 queens on the board is a state.

Initial state: No queens on the board.

Successor function: Add a queen to any empty square.

Goal test: 8 queens are on the board, none attacked.

Path cost: Zero (search cost only exists)

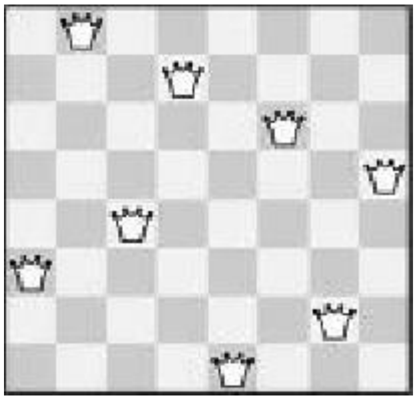


Fig 1.6 Solution to the 8 queens problem

2.3 PRODUCTION SYSTEMS

Production system is a mechanism that describes and performs the search process.

A production system consists of four basic components:

1. A set of rules of the form Ci → Ai where Ci is the condition part and Ai is the action part. The condition determines when a given rule is applied, and the action determines what happens when it is applied.

   (i.e A set of rules, each consist of left side (a pattern) that determines the applicability of the rule and a right side that describes the operations to be performed if the rule is applied)

2. One or more knowledge databases that contain whatever information is relevant for the given problem. Some parts of the database may be permanent, while others may temporary and only exist during the solution of the current problem. The information in the databases may be structured in any appropriate manner.

3. A control strategy that determines the order in which the rules are applied to the database, and provides a way of resolving any conflicts that can arise when several rules match at once.

4. A rule applier which is the computational system that implements the control strategy and applies the rules.

In order to solve a problem

- ❖ We must first reduce it to one for which a precise statement can be given. This can be done by defining the problem‘s state space (start and goal states) and a set of operators for moving that space.
- ❖ The problem can be solved by searching for a path through the space from the initial state to a goal state.
- ❖ The process of solving the problem can be usefully modeled as a production system.

2.3.1 Control strategies

By considering control strategies we can decide which rule to apply next during the process of searching for a solution to problem.

The two requirements of good control strategy are that

- ❖ **It should cause motion**: consider water jug problem, if we implement control strategy of starting each time at the top of the list of rules, it will never leads to solution. So we need to consider control strategy that leads to solution.
- ❖ **It should be systematic:** choose at random from among the applicable rules. This strategy is better than the first. It causes the motion. It will lead to the solution eventually. Doing like this is not a systematic approach and it leads to useless sequence of operators several times before finding final solution.

2.3.1.1 Systematic control strategy for the water jug problem

2.3.1.1.1 Breadth First Search (Blind Search)

Let us discuss these strategies using water jug problem. These may be applied to any search problem.

- ❖ Construct a tree with the initial state as its root.
- ❖ Generate all the offspring of the root by applying each of the applicable rules to the initial state.

$$\begin{tikzcd} & (0, 0) \arrow[dl] \arrow[dr] & \\ (4, 0) & & (0, 3) \end{tikzcd}$$

- ❖ Now for each leaf node, generate all its successors by applying all the rules that are appropriate.

$$\begin{tikzcd}
 & & (0,0) \arrow[dl] \arrow[drr] & & & \\
 & (4,0) \arrow[dl] \arrow[d] \arrow[dr] & & & (0,3) \arrow[dl] \arrow[d] \arrow[dr] & \\
(4,3) & (0,0) & (1,3) & (4,3) & (0,0) & (3,0)
\end{tikzcd}$$

- ❖ Continue this process until some rule produces a goal state.

Algorithm

1. Create a variable called NODE-LIST and set it to initial state.
2. Unit a goal state is found or NODE-LIST is empty do
    a. Remove the first element from NODE-LIST and call it E. if NODE-LIST is empty, quit.
    b. For each way that each rule can match the state described in E do:
        i. Apply the rule to generate a new state.
        ii. If the new state is a goal state, quit and return this state.
        iii. Otherwise, add the new state to the end of NODE-LIST.

2.3.1.1.2 Depth First Search

Algorithm

1. If the initial state is the goal state, quit and return success.
2. Otherwise do the following until success or failure is signaled:
    a. Generate a successor, E, of initial state. If there are no more successor, signal failure.
    b. Call depth first search, with E as the initial state.
    c. If the success is returned, signal success. Otherwise continue in this loop.

Backtracking

- ❖ In this search, we pursue a single branch of the tree until it yields a solution or until a decision to terminate the path is made.
- ❖ It makes sense to terminate a path if it reaches dead-end, produces a previous state. In such a state backtracking occurs.

- ❖ Chronological Backtracking: order in which steps are undone depends only on the temporal sequence in which steps were initially made.
- ❖ Specifically most recent step is always the first to be undone. This is also simple backtracking.

$$\begin{tikzcd}
 & & (0,0) \arrow[dl] \\
 & (4,0) \arrow[dl] & \\
(4,3) & &
\end{tikzcd}$$

Advantages of Depth First search

- ❖ DFS requires less memory since only the nodes on the current path are stored.
- ❖ By chance DFS may find a solution without examining much of the search space at all.

Advantages of Breath First search

- ❖ BFS cannot be trapped exploring a blind alley.
- ❖ If there is a solution, BFS is guaranteed to find it.
- ❖ If there are multiple solutions, then a minimal solution will be found.

Traveling Salesman Problem (with 5 cities):

A salesman is supposed to visit each of 5 cities shown below. There is a road between each pair of cities and the distance is given next to the roads. Start city is A. The problem is to find the shortest route so that the salesman visits each of the cities only once and returns to back to A.

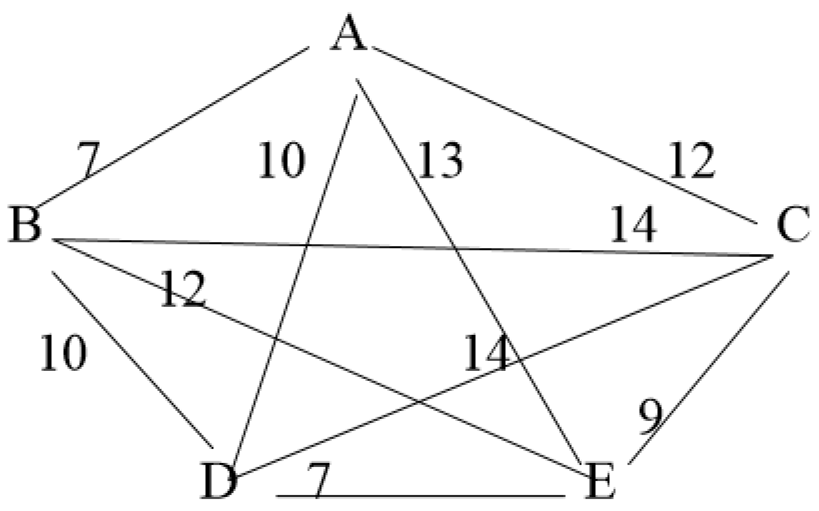


Fig Travelling Salesman Problem

•A simple, motion causing and systematic control structure could, in principle solve this problem.

•Explore the search tree of all possible paths and return the shortest path.

•This will require 4! paths to be examined.

•If number of cities grow, say 25 cities, then the time required to wait a salesman to get the information about the shortest path is of 0(24!) which is not a practical situation.

•This phenomenon is called combinatorial explosion.

•We can improve the above strategy as follows.

Branch and Bound

- ❖ Begin generating complete paths, keeping track of the shortest path found so far.
- ❖ Give up exploring any path as soon as its partial length become greater than the shortest path found so far.
- ❖ This algorithm is efficient than the first one, still requires exponential time a some number raised to N.

2.3.2 Heuristic Search

•Heuristics are criteria for deciding which among several alternatives be the most effective in order to achieve some goal.

•Heuristic is a technique that improves the efficiency of a search process possibly by sacrificing claims of systematic and completeness. It no longer guarantees to find the best answer but almost always finds a very good answer.

•Using good heuristics, we can hope to get good solution to hard problems (such as travelling salesman) in less than exponential time.

•There are **general-purpose** heuristics that are useful in a wide variety of problem domains.

•We can also construct **special purpose** heuristics, which are domain specific.

2.3.2.1 General Purpose Heuristics

• A general-purpose heuristics for combinatorial problem is nearest neighbor algorithms which works by selecting the locally superior alternative.

• For such algorithms, it is often possible to prove an upper bound on the error which provide reassurance that one is not paying too high a price in accuracy for speed.

• In many AI problems, it is often hard to measure precisely the goodness of a particular solution.

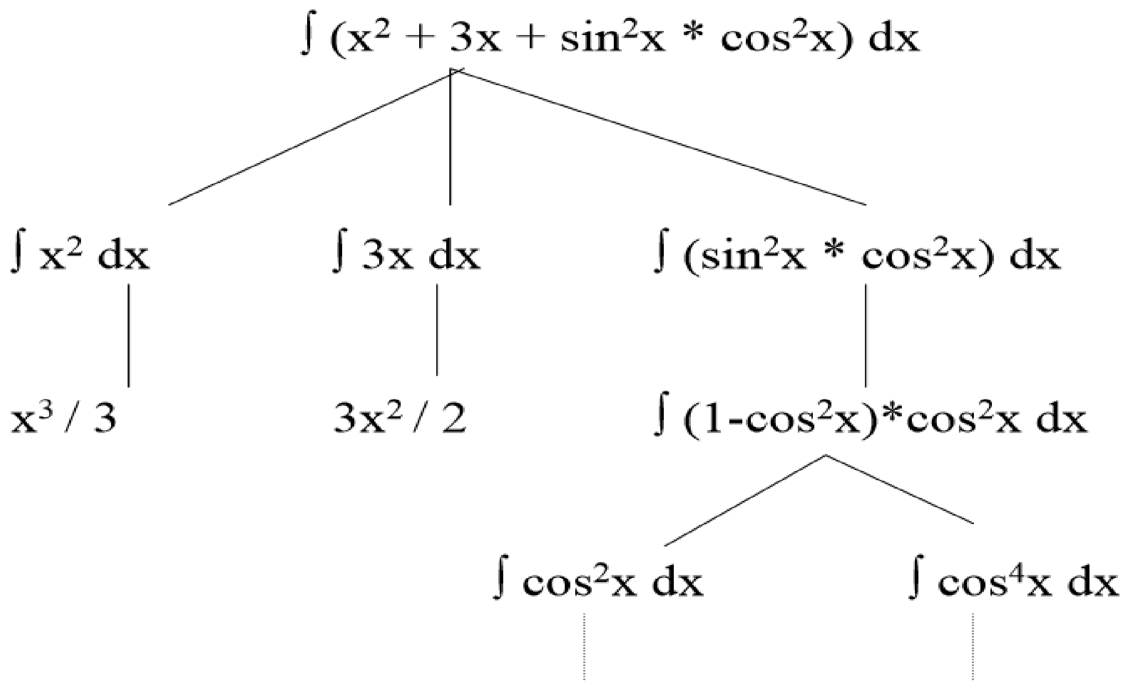
- For real world problems, it is often useful to introduce heuristics based on relatively unstructured knowledge. It is impossible to define this knowledge in such a way that mathematical analysis can be performed.
- In AI approaches, behavior of algorithms are analyzed by running them on computer as contrast to analyzing algorithm mathematically.
- There are at least many reasons for the adhoc approaches in AI.
    - ❖ It is a lot more fun to see a program do something intelligent than to prove it.
    - ❖ AI problem domains are usually complex, so generally not possible to produce analytical proof that a procedure will work.
    - ❖ It is even not possible to describe the range of problems well enough to make statistical analysis of program behavior meaningful.
- But still it is important to keep performance question in mind while designing algorithm.
- One of the most important analysis of the search process is straightforward i.e., ―Number of nodes in a complete search tree of depth D and branching factor F is F*D‖.
- This simple analysis motivates to
    - ❖ Look for improvements on the exhaustive search.
    - ❖ Find an upper bound on the search time which can be compared with exhaustive search procedures.

2.4 PROBLEM CHARACTERISTICS

Heuristic search is a very general method applicable to a large class of problem. In order to choose the most appropriate method (or combination of methods) for a particular problem it is necessary to analyze the problem along several key dimensions:

2.4.1 Is the problem decomposable into a set of independent smaller sub problems?

Example: Suppose we want to solve the problem of computing the integral of the following expression $\int(x^2+ 3x + \sin^2 x * \cos^2 x)\, dx$



$\int (x^2 + 3x + \sin^2 x * \cos^2 x)\,dx$
$\int x^2\,dx$
$\int 3x\,dx$
$\int (\sin^2 x * \cos^2 x)\,dx$
$x^3 / 3$
$3x^2 / 2$
$\int (1-\cos^2 x)*\cos^2 x\,dx$
$\int \cos^2 x\,dx$
$\int \cos^4 x\,dx$

Fig 1.7 Decomposition problem

•We can solve this problem by breaking it down into three smaller sub problems, each of which we can then be solved using a small collection of specific rules.

•Decomposable problems can be solved by the divide and-conquer technique.

•Use of decomposing problems:

-Each sub-problem is simpler to solve

-Each sub-problem can be handed over to a different processor. Thus can be solved in parallel processing environment.

•There are non-decomposable problems. For example, Block world problem is non decomposable.

2.4.2. Can solution steps be ignored or at least undone if they prove to be unwise?

•In real life, there are three types of problems: Ignorable, Recoverable and Irrecoverable.
•Let us explain each of these through examples.

Example1 :(Ignorable): In theorem proving-(solution steps can be ignored)

•Suppose we have proved some lemma in order to prove a theorem and eventually realized that lemma is no help at all, then ignore it and prove another lemma.

•Can be solved by using simple control strategy.

Example2: (Recoverable):8 puzzle-(solution steps can be undone)

•8 puzzle: Objective is to rearrange a given initial configuration of eight numbered tiles on 3 X 3 board (one place is empty) into a given final configuration (goal state).

•Rearrangement is done by sliding one of the tiles into Empty Square.

•Solved by backtracking so control strategy must be implemented using a push down stack.

Example3: (Irrecoverable): Chess (solution steps cannot be undone)

•A stupid move cannot be undone

•Can be solved by planning process

2.4.3. Is the knowledge Base consistent?

Example: Inconsistent knowledge:

Target problem: A man is standing 150 ft from a target. He plans to hit the target by shooting a gun that fires bullets with velocity of 1500 ft/sec. How high above the target should he aim?

Solution:

•Velocity of bullet is 1500 ft./sec i.e., bullet takes 0.1 secto reach the target.

•Assume bullet travels in a straight line.

•Due to gravity, the bullet falls at a distance (1/2) gt^2= (1/2)(32)(0.1)2 = 0.16ft.

•So if man aims up 0.16 feet high from the target, then bullet will hit the target.

•Now there is a contradiction to the fact that bullet travel in a straight line because the bullet in actual will travel in an arc. Therefore there is inconsistency in the knowledge used.

2.4.4. What is the Role of knowledge?

•In Chess game, knowledge is important to constrain the search for a solution otherwise just the rule for determining legal moves and some simple control mechanism that implements an appropriate search procedure is required.

•Newspapers scanning to decide some facts, a lot of knowledge is required even to be able to recognize a solution.

2.4.5. Is a good solution Absolute or Relative?

•In water jug problem there are two ways to solve a problem. If we follow one path successfully to the solution, there is no reason to go back and see if some other path might also lead to a solution. Here a solution is absolute.

•In travelling salesman problem, our goal is to find the shortest route. Unless all routes are known, the shortest is difficult to know. This is a best-path problem whereas water jug is any-path problem.

•Any path problem can often be solved in reasonable amount of time using heuristics that suggest good paths to explore.

•Best path problems are in general computationally harder than any-path.

2.4.6. Does the task Require Interaction with a Person?

- **Solitary problem**, in which there is no intermediate communication and no demand for an explanation of the reasoning process.
- **Conversational problem**, in which intermediate communication is to provide either additional assistance to the computer or additional information to the user.

2.4.7. Problem classification

- There is a variety of problem-solving methods, but there is no one single way of solving all problems.
- Not all new problems should be considered as totally new. Solutions of similar problems can be exploited.

2.5 PRODUCTION SYSTEM CHARACTERISTICS

Production systems are important in building intelligent matches which can provide us a good set of production rules, for solving problems.

There are four types of production system characteristics, namely

1. Monotonic production system

2. Non-monotonic production system

3. Commutative law based production system, and lastly

4. Partially commutative law based production system

1. **Monotonic Production System (MPS):** The Monotonic production system (MPS) is a system in which the application of a rule never prevents later application of the another rule that could also have been applied at the time that the first rule was selected

2. **Non-monotonic Production (NMPS):** The non-monotonic production system is a system in which the application of a rule prevents the later application of the another rule which may not have been applied at the time that the first rule was selected, i.e. it is a system in which the above rule is not true, i.e. the monotonic production system rule not true.

3. **Commutative Production System (CPS):** Commutative law based production systems is a system in which it satisfies both monotonic & partially commutative.

4. **Partially Commutative Production System (PCPS):** The partially commutative production system is a system with the property that if the application of those rules that is allowable & also transforms from state x to state _y‘.

	Monotonic (Characteristics)	Non-monotonic
Partially commutative	Theorem proving	Robot navigation
Non-partial commutative	Chemical synthesis	Bridge game

Table 1.1 The Four categories of Production System

Well the question may arise here such as:

- can the production systems be described by a set of characteristics?

- Also, can we draw the relationship between problem types & the types of production systems, suited to solve the problems, yes, we can by using above rules.

CHAPTER - 3

PROBLEM SOLVING METHODS, HEURISTIC SEARCH TECHNIQUES

Search techniques are the general problem solving methods. When there is a formulated search problem, a set of search states, a set of operators, an initial state and a goal criterion we can use search techniques to solve a problem.

3.1 Matching:

Problem solving can be done through search. Search involves choosing among the rules that can be applied at a particular point, the ones that are most likely to lead to a solution. This can be done by extracting rules from large number of collections.

How to extract from the entire collection of rules that can be applied at a given point?

❖ This can be done by Matching between current state and the precondition of the rules.

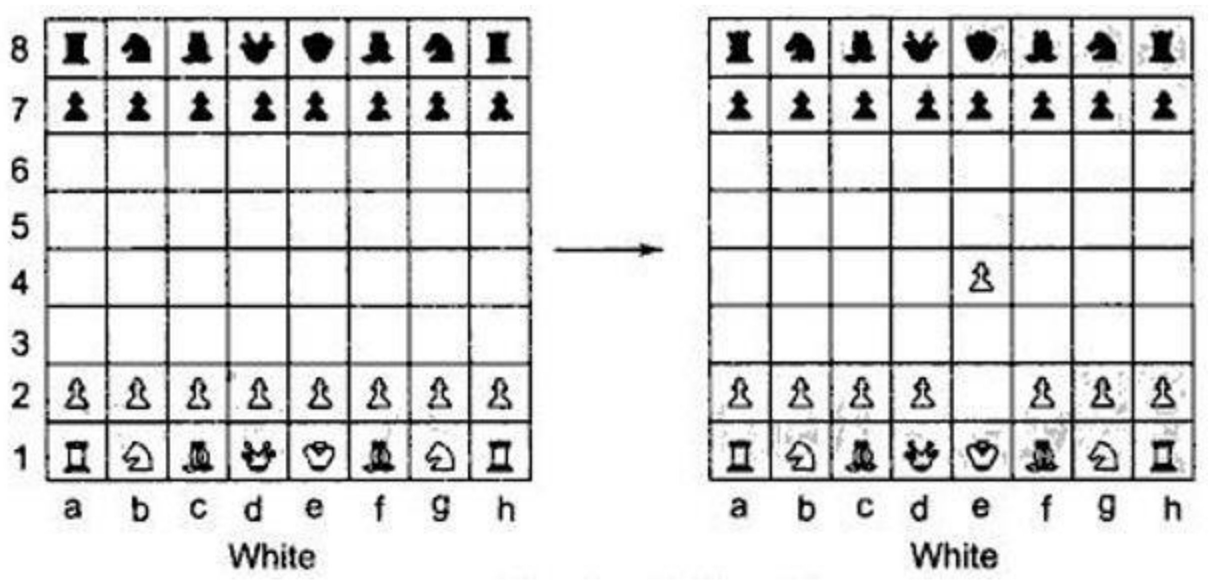
One way to select applicable rules is to do simple search through all the rules comparing each one‘s preconditions to the current state and extracting the one that match. But there are two problems with this simple solution.

1. In big problems large number of rules are used. Scanning through all of this rules at every step of the search would be hopelessly inefficient.
2. It is not clearly visible to find out which condition will be satisfied.

Some of the matching techniques are described below:

3.1.1 Indexing: To overcome above problems indexing is used. In this instead of searching all the rules the current state is used as index into the rules, and select the matching rules immediately e.g consider the chess game playing. Here the set of valid moves is very large. To reduce the size of this set only useful moves are identified. At the time of playing the game, the next move will very much depend upon the current move. As the game is going on there will be only _few‘ moves which are applicable in next move. Hence it will be wasteful effort to check applicability of all moves. Rather, the important and valid legal moves are directly stored as rules and through indexing the applicable rules are found. Here, the indexing will store the current board position. The indexing make the matching process easy, at the cost of lack of generality in the statement rules. Practically there is a tradeoff between the ease of writing rules and simplicity of matching process. The indexing technique is not very well suited for the rule base where rules are written in high level predicates. In PROLOG and many theorem proving systems, rules are indexed by predicates they contain. Hence all the applicable rules can be indexed quickly.

Example:



1.8 One Legal Chess Move

3.1.1.1 Matching with variable: In the rule base if the preconditions are not stated as exact description of particular situation, the indexing technique does not work well. In certain situations they describe properties that the situation must have. In situation, where single conditions is matched against a single element in state description, the unification procedure can be used. However in practical situation it is required to match complete set of rules that match the current state. In forward and backward chaining system, the depth first searching technique is used to select the individual rule. In the situation where multiple rules are applicable, conflict resolution technique is used to choose appropriate applicable rule. In the case of the situations requiring multiple match, the unification can be applied recursively, but more efficient method is used to use RETE matching algorithm.

3.1.1.2 Complex matching variable: A more complex matching process is required when preconditions of a rule specify required properties that are not stated explicitly in the description of current state. However the real world is full of uncertainties and sometimes practically it is not possible to define the rule in exact fashion. The matching process become more complicated in the situation where preconditions approximately match the current situation e.g a speech understanding program must contain the rules that map from a description of a physical wave form to phones. Because of the presence of noise the signal becomes more variable that there will be only approximate match between the rules that describe an ideal sound and the input that describes that unideal world. Approximate matching is particularly difficult to deal with, because as we increase the tolerance allowed in the match the new rules need to be written and it will increase number of rules. It will increase the size of main search process. But approximate matching is nevertheless superior to exact matching in situation such as speech understanding, where exact matching may result in no rule being matched and the search process coming to a grinding halt.

3.2 HEURISTIC SEARCH TECHNIQUES

3.2.1 Hill Climbing

- ❖ Hill climbing is the optimization technique which belongs to a family of local search. It is relatively simple to implement, making it a popular first choice. Although more advanced algorithms may give better results in some situations hill climbing works well.
- ❖ Hill climbing can be used to solve problems that have many solutions, some of which are better than others. It starts with a random (potentially poor) solution, and iteratively makes small changes to the solution, each time improving it a little. When the algorithm cannot see any improvement anymore, it terminates. Ideally, at that point the current solution is close to optimal, but it is not guaranteed that hill climbing will ever come close to the optimal solution.
- ❖ For example, hill climbing can be applied to the traveling salesman problem. It is easy to find a solution that visits all the cities but is be very poor compared to the optimal solution. The algorithm starts with such a solution and makes small improvements to it, such as switching the order in which two cities are visited. Eventually, a much better route is obtained.
- ❖ Hill climbing is used widely in artificial intelligence, for reaching a goal state from a starting node. Choice of next node and starting node can be varied to give a list of related algorithms.
- ❖ Hill climbing attempts to maximize (or minimize) a function f (x), where x are discrete states. These states are typically represented by vertices in a graph, where edges in the graph encode nearness or similarity of a graph. Hill climbing will follow the graph from vertex to vertex, always locally increasing (or decreasing) the value of f, until a local maximum (or local minimum) x_m is reached. Hill climbing can also operate on a continuous space: in that case, the algorithm is called gradient ascent (or gradient descent if the function is minimized).*.
- ❖ Problems with hill climbing: local maxima (we've climbed to the top of the hill, and missed the mountain), plateau (everything around is about as good as where we are),ridges (we're on a ridge leading up, but we can't directly apply an operator to improve our situation, so we have to apply more than one operator to get there).Solutions include: backtracking, making big jumps (to handle plateaus or poor local maxima), applying multiple rules before testing (helps with ridges).Hill climbing is best suited to problems where the heuristic gradually improves the closer it gets to the solution; it works poorly where there are sharp drop-offs. It assumes that local improvement will lead to global improvement.
- ❖ Search methods based on hill climbing get their names from the way the nodes are selected for expansion. At each point in the search path a successor node that appears to lead most quickly to the top of the hill (goal) selected for exploration. This method requires that some information be available with which to evaluate and order the most

promising choices. Hill climbing is like depth first searching where the most promising child is selected for expansion.

- ❖ Hill climbing is a variant of generate and test in which feedback from the test procedure is used to help the generator decide which direction to move in the search space. Hill climbing is often used when a good heuristic function is available for evaluating states but when no other useful knowledge is available. For example, suppose you are in an unfamiliar city without a map and you want to get downtown. You simply aim for the tall buildings. The heuristic function is just distance between the current location and the location of the tall buildings and the desirable states are those in which this distance is minimized.

3.2.1 Simple Hill Climbing

The simplest way to implement hill climbing is the simple hill climbing whose algorithm is as given below:

Algorithm: Simple Hill Climbing:

```
Step 1: Evaluate the initial state. It it is also a goal state, then return it and quit.
Otherwise continue with the initial state as the current state.

Step 2: Loop until a solution is found or until there are no new operators left to be
applied in the current state:

        (a) Select an operator that has not yet been applied to the current state and apply it
        to produce a new state.

        (b) Evaluate the new state.

            (i) If it is a goal state, then return it and quit.

            (ii) If it is not a goal state, but it is better than the current state, then make it the
            current state.

            (iii) If it is not better than the current state, then continue in the loop.
```

The key difference between this algorithm and the one we gave for generate and test is the use of an evaluation function as a way to inject task specific knowledge into the control process. It is the use of such knowledge that makes this heuristic search method. It is the same knowledge that gives these methods their power to solve some otherwise intractable problems

To see how hill climbing works, let‘s take the puzzle of the four colored blocks. To solve the problem we first need to define a heuristic function that describes how close a particular configuration is to being a solution. One such function is simply the sum of the number of different colors on each of the four sides. A solution to the puzzle will have a value of 16. Next we need to define a set of rules that describe ways of transforming one configuration to another. Actually one rule will suffice. It says simply pick a block and rotate it 90 degrees in any direction. Having provided these definitions the next step is to generate a starting configuration. This can either be done at random or with the aid of the heuristic function. Now by using hill climbing, first we generate a new state by selecting a block and rotating it. If the resulting state is better than we keep it. If not we return to the previous state and try a different perturbation.

3.2.2 Problems in Hill Climbing

3.2.2.1 Steepest – Ascent Hill Climbing:

An useful variation on simple hill climbing considers all the moves form the current state and selects the best one as the next state. This method is called steepest – ascent hill climbing or gradient search. Steepest Ascent hill climbing contrasts with the basic method in which the first state that is better than the current state is selected. The algorithm works as follows.

Algorithm: Steepest – Ascent Hill Climbing

Step 1: Evaluate the initial state. If it is also a goal state, then return it and quit .Otherwise, continue with the initial state as the current state.

Step 2: Loop until a solution is found or until a complete iteration produces no change to current state:

(a) Let SUCC be a state such that any possible successor of the current state will be better than SUCC.

(b) For each operator that applies to the current state do:

i. Apply the operator and generate a new state.

ii. Evaluate the new state. If it is a goal state, then return it and quit. If not, compare it to SUCC. If it is better then set SUCC to this state. If it is not better, leave SUCC alone.

(c) If the SUCC is better than current state, then set current state to SUCC.

To apply steepest- ascent hill climbing to the colored blocks problem, we must consider all perturbations of the initial state and choose the best. For this problem this is difficult since there are so many possible moves. There is a trade-off between the time required to select a move and the number of moves required to get a solution that must be considered when deciding which method will work better for a particular problem. Usually the time required to select a move is longer for steepest – ascent hill climbing and the number of moves required to get to a solution is longer for basic hill climbing.

Both basic and steepest ascent hill climbing may fail to find a solution. Either algorithm may terminate not by finding a goal state but by getting to a state from which no better states can be generated. This will happen if the program has reached either a local maximum, a plateau, or a ridge.

Hill climbing Disadvantages

Local Maximum: A local maximum is a state that is better than all its neighbors but is not better than some other states farther away.

Plateau: A Plateau is a flat area of the search space in which a whole set of neighboring states have the same value.

Ridge: A ridge is a special kind of local maximum. It is an area of the search space that is higher than surrounding areas and that itself has a slope.

Ways out

- ❖ Backtrack to some earlier node and try going in a different direction.
- ❖ Make a big jump to try to get in a new section.
- ❖ Moving in several directions at once.

Hill climbing is a **local method:** Decides what to do next by looking only at the immediate consequence of its choice.

Global information might be encoded in heuristic functions.

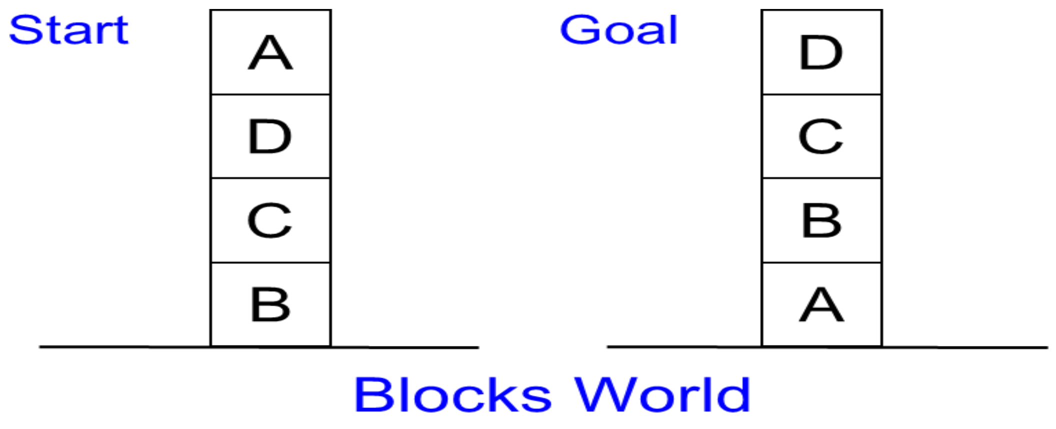
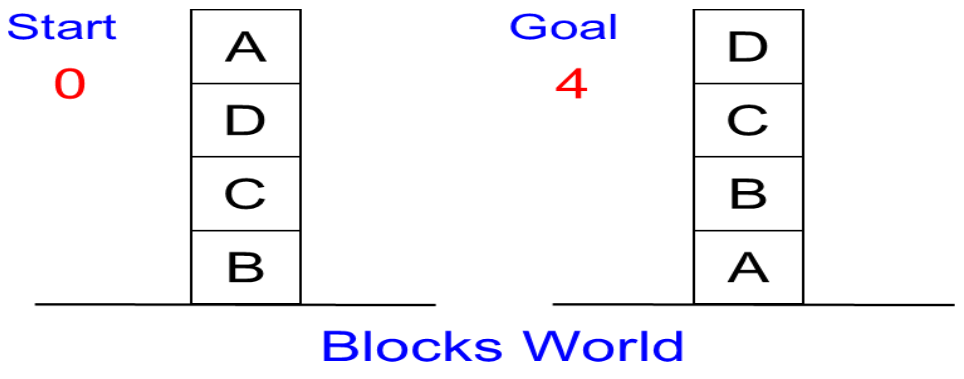
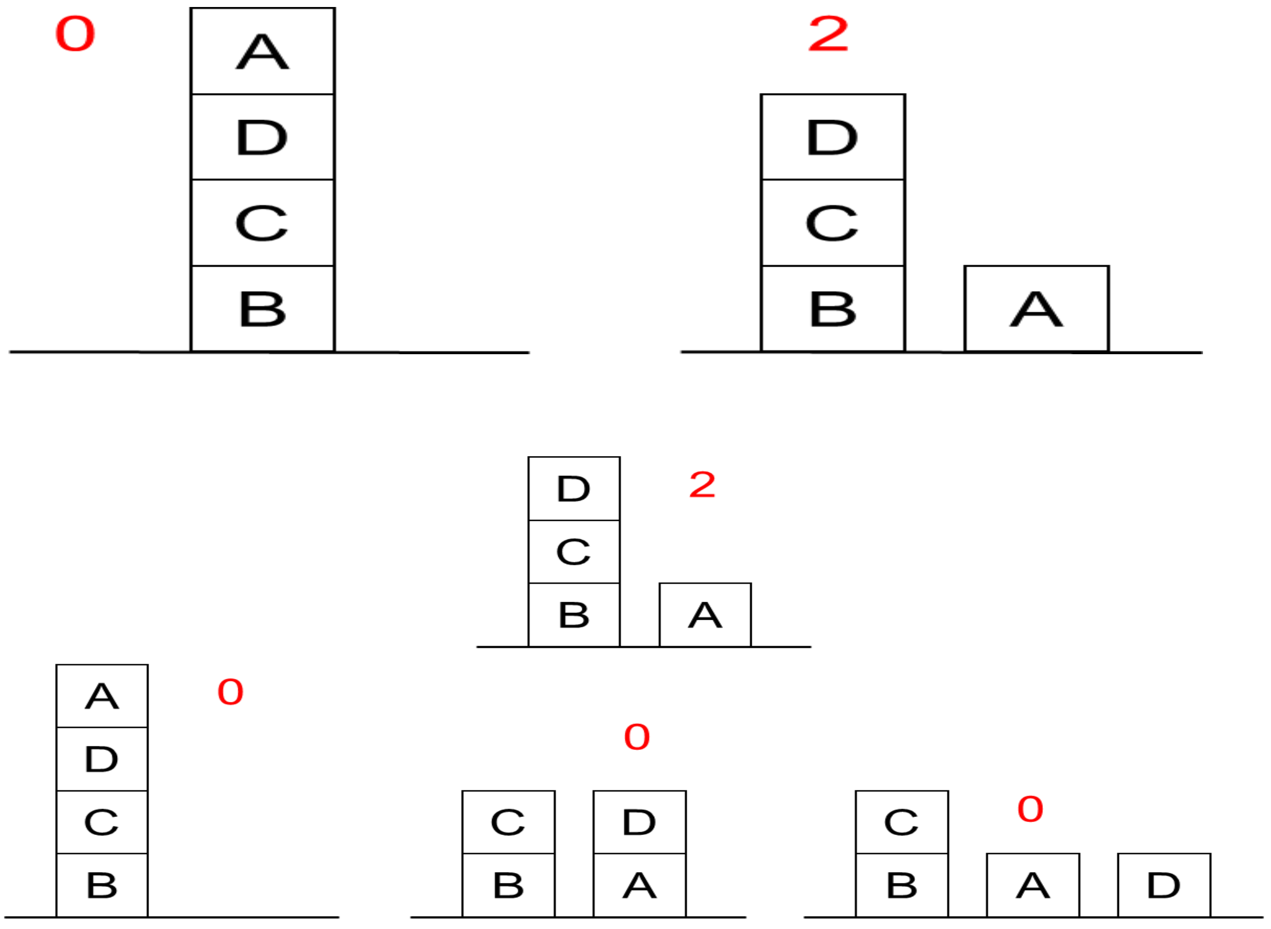


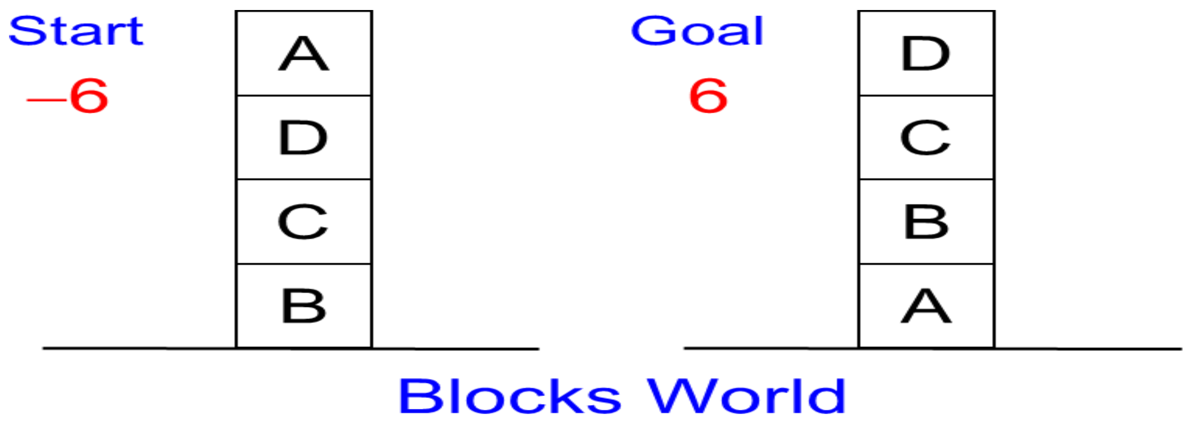
Fig 1.9 Three Possible Moves



Local heuristic:

- ❖ +1 for each block that is resting on the thing it is supposed to be resting on.
- ❖ -1 for each block that is resting on a wrong thing.

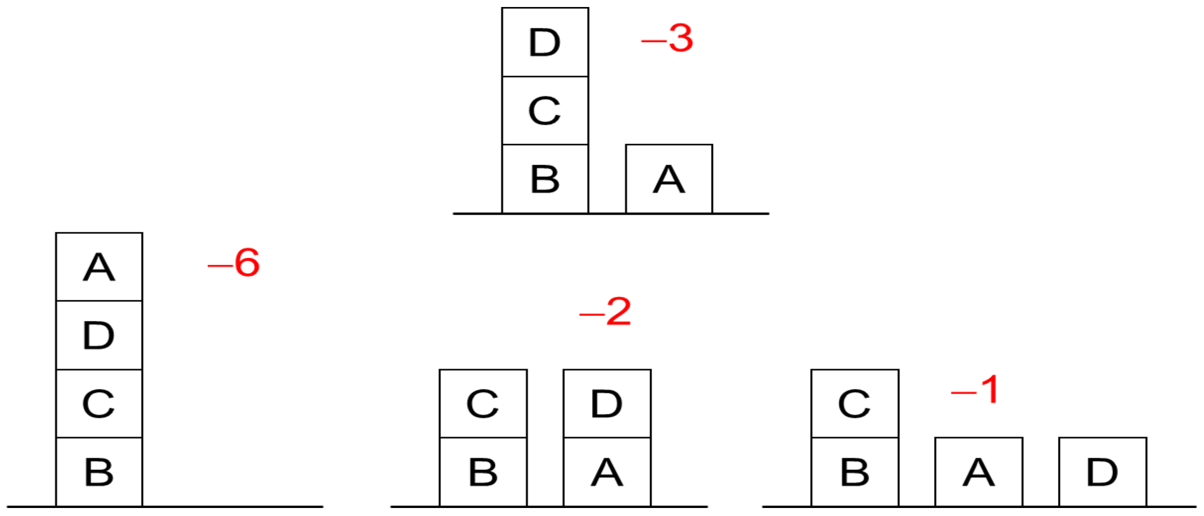




Global heuristic:

For each block that has the correct support structure: +1 to every block in the support structure.

For each block that has a wrong support structure: -1 to every block in the support structure.



Hill climbing conclusion

- ❖ Can be very inefficient in a large, rough problem space.
- ❖ Global heuristic may have to pay for computational complexity.
- ❖ Often useful when combined with other methods, getting it started right in the right general neighbourhood.

3.2.3 Simulated Annealing

The problem of local maxima has been overcome in simulated annealing search. In normal hill climbing search, the movements towards downhill are never made. In such algorithms the search may stuck up to local maximum. Thus this search cannot guarantee complete solutions. In contrast, a random search(or movement) towards successor chosen randomly from the set of successor will be complete, but it will be extremely inefficient. The combination of hill climbing and random search, which yields both efficiency and completeness is called simulated annealing.

The simulated annealing method was originally developed for the physical process of annealing. That is how the name simulated annealing was found and restored. In simulated annealing searching algorithm, instead of picking the best move, a random move is picked. The standard simulated annealing uses term objective function instead of heuristic function. If the move improves the situation it is accepted otherwise the algorithm accepts the move with some probability less than

This probability is

$P= e^{-\Delta E/kT}$

Where $-\Delta E$ is positive charge in energy level, t is temperature and k is Boltzman constant. As indicated by the equation the probability decreases with badness of the move (evaluation gets worsened by amount $-\Delta E$). The rate at which $-\Delta E$ is cooled is called annealing schedule. The proper annealing schedule is maintained to monitor T.

This process has following differences from hill climbing search:

- ❖ The annealing schedule is maintained.
- ❖ Moves to worse states are also accepted.
- ❖ In addition to current state, the best state record is maintained.

The algorithm of simulated annealing is presented as follows:

Algorithm: "simulated annealing‖

1. Evaluate the initial state. Mark it as current state. Till the current state is not a goal state, initialize best state to current state. If the initial state is the best state, return it and quit.
2. Initialize T according to annealing schedule.
3. Repeat the following until a solution is obtained or operators are not left:
   a. Apply yet unapplied operator to produce a new state
   b. For new state compute $-\Delta E$= value of current state – value of new state. If the new state is the goal state then stop, or if it is better than current state, make it as current state and record as best state.
   c. If it is not better than the current state, then make it current state with probability P.
   d. Revise T according to annealing schedule
4. Return best state as answer.

3.3 Best-First Search:

It is a general heuristic based search technique. In best first search, in the graph of problem representation, one evaluation function (which corresponds to heuristic function) is attached with every node. The value of evaluation function may depend upon cost or distance of current node from goal node. The decision of which node to be expanded depends on the value of this evaluation function. The best first can understood from following tree. In the tree, the attached value with nodes indicates utility value. The expansion of nodes according to best first search is illustrated in the following figure.

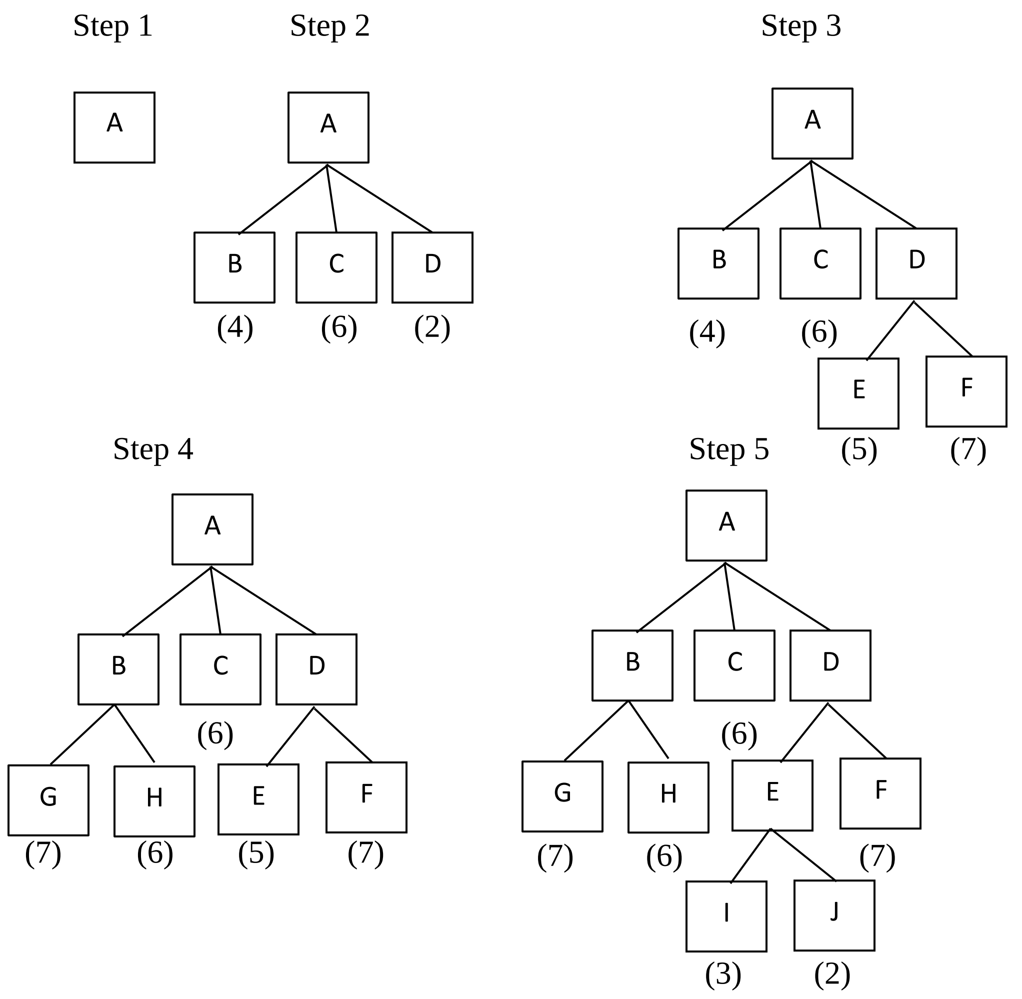


fig:1.10 Tree getting expansion according to best first search

Here, at any step, the most promising node having least value of utility function is chosen for expansion.

In the tree shown above, best first search technique is applied, however it is beneficial sometimes to search a graph instead of tree to avoid the searching of duplicate paths. In the process to do so, searching is done in a directed graph in which each node represents a point in the problem space. This graph is known as OR-graph. Each of the branches of an OR graph represents an alternative problem solving path.

Two lists of nodes are used to implement a graph search procedure discussed above. These are

1.OPEN: these are the nodes that have been generated and have had the heuristic function applied to them but not have been examined yet.

2.CLOSED: these are the nodes that have already been examined. These nodes are kept in the memory if we want to search a graph rather than a tree because whenever a node will be generated, we will have to check whether it has been generated earlier.

The best first search is a way of combining the advantage of both depth first and breath first search. The depth first search is good because it allows a solution to be found without all competing branches have to be expanded. Breadth first search is good because it does not get trapped on dead ends of path. The way of combining this is to follow a single path at a time but switches between paths whenever some competing paths looks more promising than current one does. Hence at each step of best first search process, we select most promising node out of successor nodes that have been generated so far.

The functioning of best first search is summarized in the following steps:

1. It maintains a list open containing just the initial state.
2. Until a goal is found or there are no nodes left in open list do:
    a. Pick the best node from open,
    b. Generate its successor, and for each successor:
        i. Check, and if it has not been generated before evaluate it and add it to open and record its parent.
        ii. If it has been generated before, and new path is better than the previous parent then change the parent.

The algorithm for best first search is given as follows:

Algorithm: Best first search

1. Put the initial node on the list say _OPEN‘.
2. If (OPEN = empty or OPEN= goal) terminate search, else
3. Remove the first node from open(say node is a)
4. If (a=goal) terminate search with success else
5. Generate all the successor node of _a‘. Send node _a‘ to a list called _CLOSED‘. Find out the value of heuristic function of all nodes. Sort all children generated so far on the basis of their utility value. Select the node of minimum heuristic value for further expansion.
6. Go back to step 2.

The best first search can be implemented using priority queue. There are variations of best first search. Example of these are greedy best first search, A* and recursive best first search.

3.3.2 The A* Algorithm:

The A* algorithm is a specialization of best first search. It most widely known form of best first search. It provides genera guidelines about how to estimate goal distance for general search graph. at each node along a path to the goal node, the A* algorithm generate all successor nodes and computes an estimate of distance (cost) from the start node to a goal node through each of the successors. if then chooses the successor with shortest estimated distance from expansion. It calculates the heuristic function based on distance of current node from the start node and distance of current node to goal node.

The form of heuristic estimation function for A* is defined as follows:

f(n)=g(n)+h(n)

where f(n)= evaluation function

g(n)= cost (or distance) of current node from start node.

h(n)= cost of current node from goal node.

In A* algorithm the most promising node is chosen from expansion. The promising node is decided based on the value of heuristic function. Normally the node having lowest value of f (n) is chosen for expansion. We must note that the goodness of a move depends upon the nature of problem, in some problems the node having least value of heuristic function would be most promising node, where in some situation, the node having maximum value of heuristic function is chosen for expansion. A* algorithm maintains two lists. One store the list of open nodes and other maintain the list of already expanded nodes. A* algorithm is an example of optimal search algorithm. A search algorithm is optimal if it has admissible heuristic. An algorithm has admissible heuristic if its heuristic function h(n) never overestimates the cost to reach the goal. Admissible heuristic are always optimistic because in them, the cost of solving the problem is less than what actually is. The A* algorithm works as follows:

A* algorithm:

1. Place the starting node _s‘ on _OPEN‘ list.
2. If OPEN is empty, stop and return failure.
3. Remove from OPEN the node _n‘ that has the smallest value of f*(n). if node _n is a goal node, return success and stop otherwise.
4. Expand _n‘ generating all of its successors _n‘ and place _n‘ on CLOSED. For every successor _n‘ if _n‘ is not already OPEN , attach a back pointer to _n‘. compute f*(n) and place it on CLOSED.
5. Each _n‘ that is already on OPEN or CLOSED should be attached to back pointers which reflect the lowest f*(n) path. If _n‘ was on CLOSED and its pointer was changed, remove it and place it on OPEN.
6. Return to step 2.

3.4 Constraint Satisfaction

The general problem is to find a solution that satisfies a set of constraints. The heuristics which are used to decide what node to expand next and not to estimate the distance to the goal. Examples of this technique are design problem, labeling graphs robot path planning and crypt arithmetic puzzles.

In constraint satisfaction problems a set of constraints are available. This is the search space. Initial State is the set of constraints given originally in the problem description. A goal state is any state that has been constrained enough. Constraint satisfaction is a two-step process.

1. First constraints are discovered and propagated throughout the system.

2. Then if there is not a solution search begins, a guess is made and added to this constraint. Propagation then occurs with this new constraint.

Algorithm

1. Propogate available constraints:

- ✓ Open all objects that must be assigned values in a complete solution.
- ✓ Repeat until inconsistency or all objects are assigned valid values:

Select an object and strengthen as much as possible the set of constraints that apply to object.

If set of constraints different from previous set then open all objects that share any of these constraints. Remove selected object.

2. If union of constraints discovered above defines a solution return solution.

3. If union of constraints discovered above defines a contradiction return failure.

4. Make a guess in order to proceed. Repeat until a solution is found or all possible solutions exhausted:

- ✓ Select an object with a no assigned value and try to strengthen its constraints.
- ✓ Recursively invoke constraint satisfaction with the current set of constraints plus the selected strengthening constraint.

Crypt arithmetic puzzles are examples of constraint satisfaction problems in which the goal to discover some problem state that satisfies a given set of constraints. some problems of crypt arithmetic are show below

```
    S E N D       D O N A L D        C R O S S
  + M O R E     + G E R A L D      + R O A D S
  ----------    -------------      -----------
  M O N E Y     R O B E R T        D A N G E R
```

Here each decimal digit is to be assigned to each of the letters in such a way that the answer to the problem is correct. If the same letter occurs more than once it must be assigned the same digit each time. No two different letters may be assigned the same digit.

The puzzle SEND + MORE = MONEY, after solving, will appear like this

```
    S  E  N  D
    9  5  6  7
 +  M  O  R  E
    1  0  8  5
-------------------
 M  O  N  E  Y
 1  0  6  5  2
```

State production and heuristics for crypt arithmetic problem.

Ans.

The heuristics and production rules are specific to the following example:

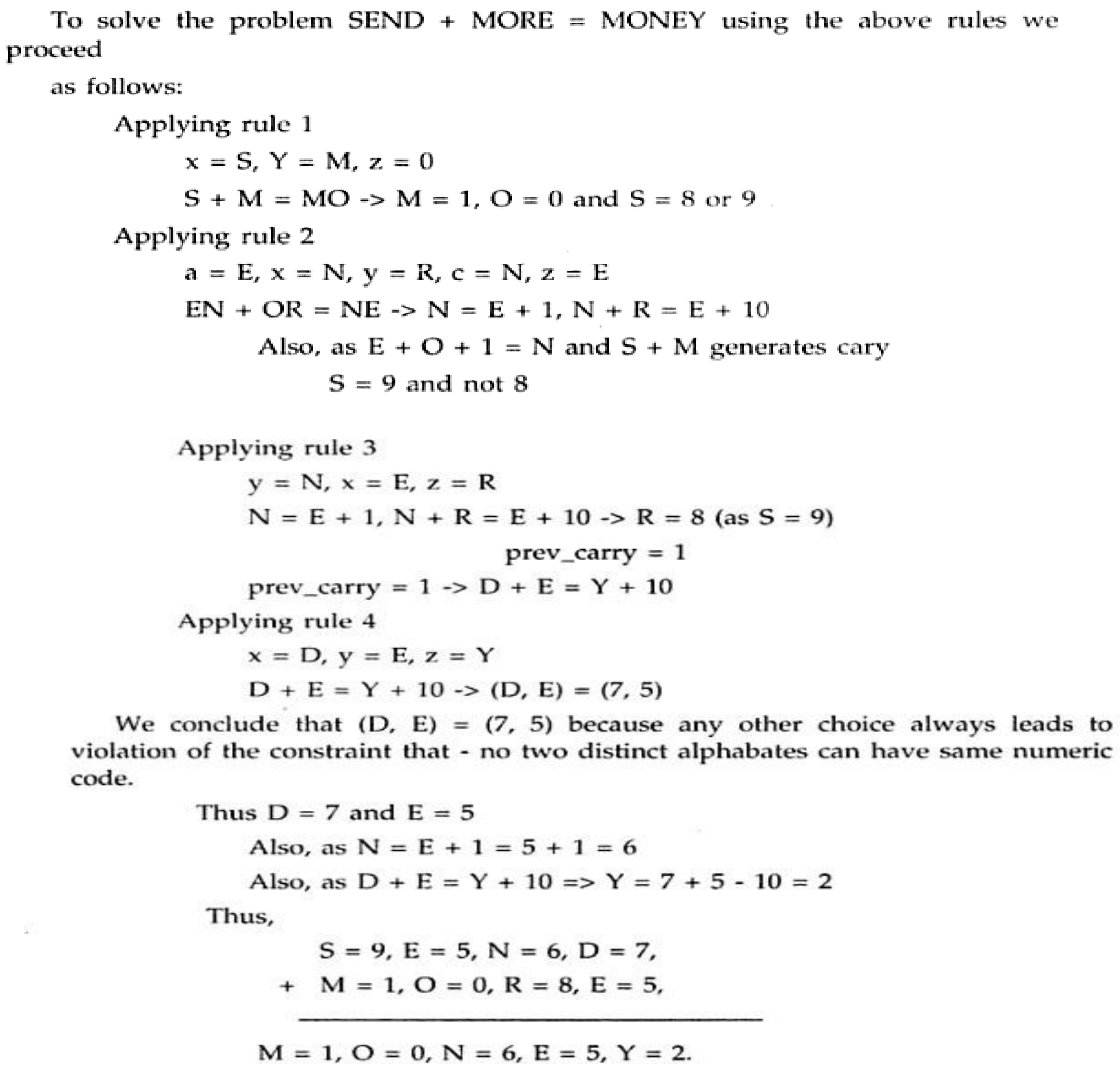
```
    S  E  N  D
    M  O  R  E
----------------------
 M  O  N  E  Y
```

Heuristics Rules

1. If sum of two _n�Vdigit operands yields _n+1‘ digit result then the _n+1‘th digit has to be one.
2. Sum of two digits may or may not generate carry.
3. Whatever might be the operands the carry can be either 0 or 1.
4. No two distinct alphabets can have same numeric code.
5. Whenever more than 1 solution appears to be existing, the choice is governed by the fact that no two alphabets can have same number code.

Production Rules:

```
1. x  +  y = yz                ->  y = 1, z = 0, x = 9
2. ax  +  0y = cz              ->  c = a +1, x + y = z + 10
3. y = x + 1, y + z = x + 10   ->  z = 8 or 9 and prev_carry = 1
4. x  +  y = z + 10            ->  (x, y) = (8, 9) | (7, 9) | (6, 9) |
                                   (5, 9) | (4, 9) | (3, 9) |
                                   (2, 9) | (1, 9) | (7, 8) |
                                   (6, 8) | (5, 8) | (4, 8) |
                                   (3, 8) | (2, 8) | (6, 7) |
                                   (5, 7) | (4 ,7) | (3, 7) |
                                   (5, 6) | (4, 6).
```



To solve the problem SEND + MORE = MONEY using the above rules we proceed

as follows:

Applying rule 1

x = S, Y = M, z = 0

S + M = MO -> M = 1, O = 0 and S = 8 or 9

Applying rule 2

a = E, x = N, y = R, c = N, z = E

EN + OR = NE -> N = E + 1, N + R = E + 10

Also, as E + O + 1 = N and S + M generates cary

S = 9 and not 8

Applying rule 3

y = N, x = E, z = R

N = E + 1, N + R = E + 10 -> R = 8 (as S = 9)

prev_carry = 1

prev_carry = 1 -> D + E = Y + 10

Applying rule 4

x = D, y = E, z = Y

D + E = Y + 10 -> (D, E) = (7, 5)

We conclude that (D, E) = (7, 5) because any other choice always leads to violation of the constraint that - no two distinct alphabates can have same numeric code.

Thus D = 7 and E = 5

Also, as N = E + 1 = 5 + 1 = 6

Also, as D + E = Y + 10 => Y = 7 + 5 - 10 = 2

Thus,

```
      S = 9, E = 5, N = 6, D = 7,
   +  M = 1, O = 0, R = 8, E = 5,
   ________________________________
    M = 1, O = 0, N = 6, E = 5, Y = 2.
```

3.5 Means – end Analysis

Means-ends analysis allows both backward and forward searching. This means we could solve major parts of a problem first and then return to smaller problems when assembling the final solution.

The means-ends analysis algorithm can be said as follows:

1. Until the goal is reached or no more procedures are available:
   - ✓ Describe the current state the goal state and the differences between the two.
   - ✓ Use the difference the describes a procedure that will hopefully get nearer to goal.
   - ✓ Use the procedure and update current state

If goal is reached then success otherwise fail.

For using means-ends analysis to solve a given problem, a mixture of the two directions, forward and backward, is appropriate. Such a mixed strategy solves the major parts of a problem first and then goes back and solves the small problems that arise by putting the big pieces together. The means-end analysis process detects the differences between the current state and goal state. Once such difference is isolated an operator that can reduce the difference has to be found. The operator may or may not be applied to the current state. So a sub problem is set up of getting to a state in which this operator can be applied.

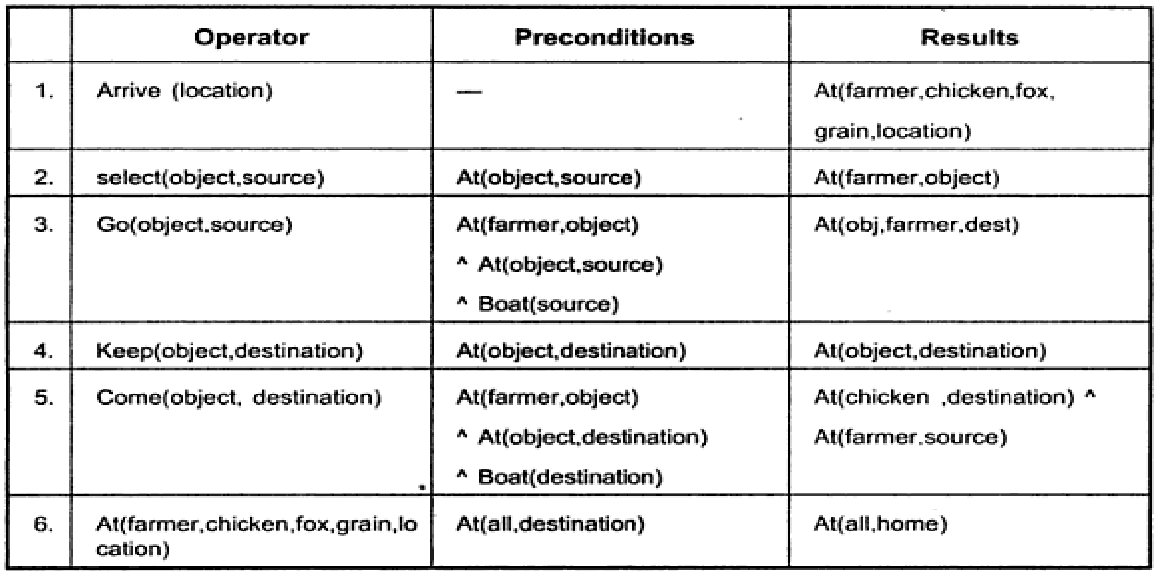
In operator sub goaling backward chaining and forward chaining is used in which first the operators are selected and then sub goals are set up to establish the preconditions of the operators. If the operator does not produce the goal state we want, then we have second sub problem of getting from the state it does produce to the goal. The two sub problems could be easier to solve than the original problem, if the difference was chosen correctly and if the operator applied is really effective at reducing the difference. The means-end analysis process can then be applied recursively.

This method depends on a set of rues that can transform one problem state into another. These rules are usually not represented with complete state descriptions on each side. Instead they are represented as a left side that describes the conditions that must be met for the rules to be applicable and a right side that describes those aspects of the problem state that will be changed by the application of the rule. A separate data structure called a difference table which uses the rules by the differences that they can be used to reduce.

Example Solve the following problem using means-ends analysis

A farmer wants to cross the river along with fox, chicken and grain. He can take only one along with him at a time. If fox and chicken are left alone the fox may eat the chicken. If chicken and grain are left alone the chicken may eat the grain. Give the necessary solution.

Solution:



	Operator	Preconditions	Results
1.	Arrive (location)	—	At(farmer,chicken,fox, grain,location)
2.	select(object,source)	At(object,source)	At(farmer,object)
3.	Go(object,source)	At(farmer,object) ^ At(object,source) ^ Boat(source)	At(obj,farmer,dest)
4.	Keep(object,destination)	At(object,destination)	At(object,destination)
5.	Come(object, destination)	At(farmer,object) ^ At(object,destination) ^ Boat(destination)	At(chicken ,destination) ^ At(farmer,source)
6.	At(farmer,chicken,fox,grain,location)	At(all,destination)	At(all,home)

Difference table.

	Actions	Arrive	Select	GO	Keep	Come
1.	Arrive at the location	*				
2.	Take Chicken		*			
3.	Travel by boat to destination			*		
4.	Leave chicken at destination				*	
5.	Come back at source					*
6.	Take fox	*				
7.	Travel by boat to destination		*			
8.	Keep fox at destination and take chicken back to source.			*	*	
9.	Keep chicken at source and take grain to destination.		*	*		
10.	Keep grain to destination			*		

PART-A

1. What is AI?
2. What are the task domains of artificial intelligence?
3. List the properties of knowledge?
4. What is an AI technique?
5. What are the steps to build a system that solves a problem?
6. What is a state space?
7. Explain the process operationalization?
8. How to define the problem as a state space search?
9. What does the production system consists of?
10. What are the requirements of a good control strategy?
11. What is chronological backtracking?
12. Give the advantages of depth-first search?
13. Give the advantages of breadth-first search?
14. What is combinatorial explosion?
15. Give an example of a heuristic that is useful for combinatorial problems?
16. What is heuristic?
17. Define heuristic function?
18. What is the purpose of heuristic function?
19. Write down the various problem characteristics?
20. What is certain outcome and uncertain outcome problem with examples?
21. What are the classes of problems with respect to solution steps with eg?
22. Illustrate the difference between any-path and best problem with examples?
23. What are the types of problems with respect to task interaction with a person?
24. What is propose and refine?
25. What is monotonic production system?
26. What is nonmonotonic production system?
27. What is commutative and partially commutative production system?

28. What are weak methods?

29. Write generate and test algorithm?
30. How is hill climbing different from generate and test?
31. When hill climbing fails to find a solution?
32. What is local maximum?
33. What is ridge?
34. What is plateau?
35. List the ways to overcome hill climbing problems?
36. Differentiate steepest accent from basic hill climbing?
37. Differentiate simple hill climbing from simulated annealing?
38. What are the components essential to select an annealing schedule?
39. What is best first search process?
40. State _Graceful decay of admissibility'
41. What is an agenda?
42. What is the limitation of problem reduction algorithm?
43. What is constraint satisfaction?
44. What is operator subgoaling?
45. Define playing chess.

PART-B

1. Explain briefly the various problem characteristics?
2. Illustrate how to define a problem as a state space search with an example?
3. Discuss the merits and demerits of depth-first and breadth-first search with the algorithm?
4. What are the problems encountered during hill climbing and what are the ways available to deal with these problems?
5. Explain the process of simulated annealing with example?
6. Write A* algorithm and discuss briefly the various observations about algorithm?
7. Discuss AO* algorithm in detail?
8. Write in detail about the constraint satisfaction procedure with example?

9. Explain how the steepest accent hill climbing works?

10. Write in detail about the mean end analysis procedure with example?

UNIT-2

REPRESENTATION OF KNOWLEDGE

CHAPTER-1

GAME PLAYING

2.1 Introduction

Game Playing is one of the oldest sub-fields in AI. Game playing involves abstract and pure form of competition that seems to require intelligence. It is easy to represent the states and actions. To implement the game playing very little world knowledge is required.

The most common used AI technique in game is search. Game playing research has contributed ideas on how to make the best use of time to reach good decisions.

Game playing is a search problem defined by:

- Initial state of the game
- Operators defining legal moves
- Successor function
- Terminal test defining end of game states
- Goal test
- Path cost/utility/payoff function

More popular games are too complex to solve, requiring the program to take its best guess. " for example in chess, the search tree has 1040 nodes (with branching factor of 35). It is the opponent because of whom uncertainty arises.

Characteristics of game playing

1. There are always an "unpredictable" opponent:
    - The opponent introduces uncertainty
    - The opponent also wants to win

The solution for this problem is a strategy, which specifies a move for every possible opponent reply.

2. Time limits:

Game are often played under strict time constraints (eg:chess) and therefore must be very effectively handled.

There are special games where two players have exactly opposite goals. There are also perfect information games(sch as chess and go) where both the players have access to the same information about the game in progress (e.g. tic-tac-toe). In imoerfect game information games (such as bridge or certain card games and games where dice is used). Given sufficient time and space, usually an optimum solution can be obtained for the former by exhaustive search, though not for the latter.

Types of games

There are basically two types of games

- Deterministic games
- Chance games

Game like chess and checker are perfect information deterministic games whereas games like scrabble and bridge are imperfect information. We will consider only two player discrete, perfect information games, such as tic-tac-toe, chess, checkers etc... . Two- player games are easier to imagine and think and more common to play.

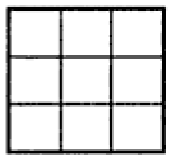
Minimize search procedure

Typical characteristic of the games is to look ahead at future position in order to succeed. There is a natural correspondence between such games and state space problems.

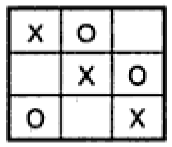
In a game like tic-tac-toe

- States-legal board positions
- Operators-legal moves
- Goal-winning position

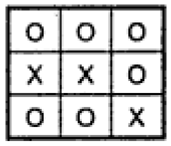
The game starts from a specified initial state and ends in position that can be declared win for one player and loss for other or possibly a draw. Game tree is an explicit representation of all possible plays of the game. We start with a 3 by 3 grid..



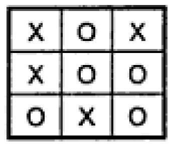
Then the two players take it in turns to place a there marker on the board(one player uses the „X‟ marker, the other uses the „O‟ marker). The winner is the player who gets 3 of these markers in a row, eg.. if X wins



Another possibility is that no1 wins eg..



Or the third possibility is a draw case

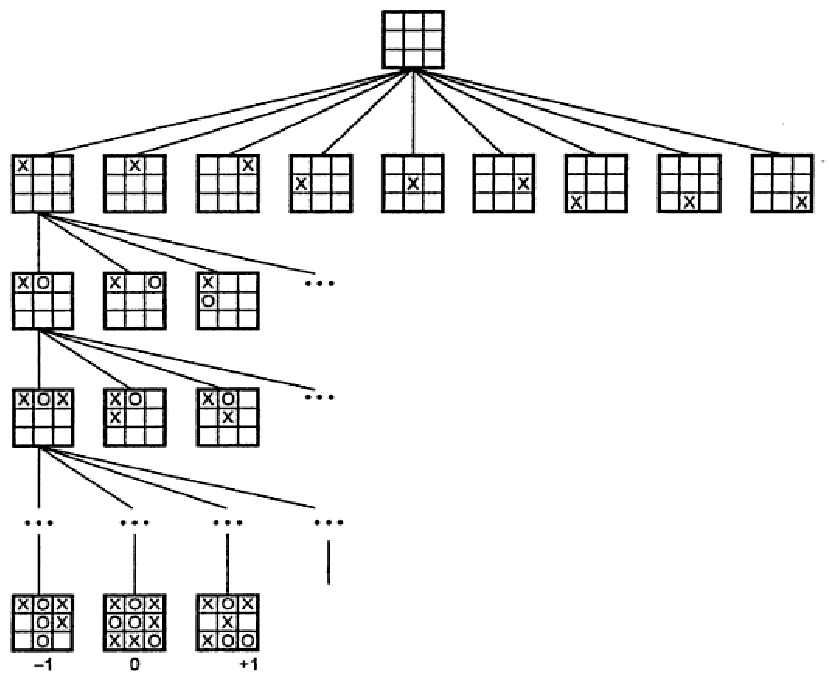


Search tree for tic-tac-toe

The root node is an initial position of the game. Its successors are the positions that the first player can reach in one move; their successors are the positions resulting from the second player's replies and so on. Terminal or leaf nodes are presented by WIN, LOSS or DRAW. Each path from the root ro a terminal node represents a different complete play of the game. The moves available to one player from a given position can be represented by OR links whereas the moves available to his opponent are AND links.

The trees representing games contain two types of nodes:

- MAX- nodes (assume at even level from root)
- MIN - nodes [assume at odd level from root)



Search tree for tic-tac-toe

the leaves nodes are labeled WIN, LOSS or DRAW depending on whether they represent a win, loss or draw position from Max‟s viewpoint. Once the leaf nodes are assigned their WIN-LOSS or DRAW status, each nodes in the game tree can be labeled WIN, LOSS or DRAW by a bottom up process.

Game playing is a special type of search, where the intention of all players must be taken into account.

Minimax procedure

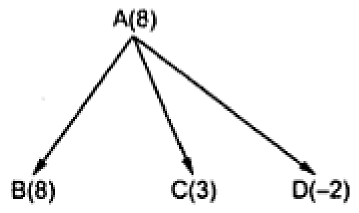
- Starting from the leaves of the tree (with final scores with respect to one player, MAX), and go backwards towards the root.
- At each step, one player (MAX) takes the action that leads to the highest score, while the other player(MIN) takes the action that leads to the lowest score.
- All the nodes in the tree will be scored and the path from root to the actual result is the one on which all node have the same score.

The minimax procedure operates on a game tree and is recursive procedure where a player tries to minimize its opponent‟s advantage while at the same time maximize its own. The player hoping for positive number is called the maximizing player. His opponent is the minimizing player. If the player to move is the maximizing player, he is looking for a path leading to a large positive number and his opponent will try to force the play toward situation with strongly negative static evaluations. In game playing first construct the tree up till the depth-bound and then compute the evaluation function for the leaves. The next step is to propagate the values up to the starting.

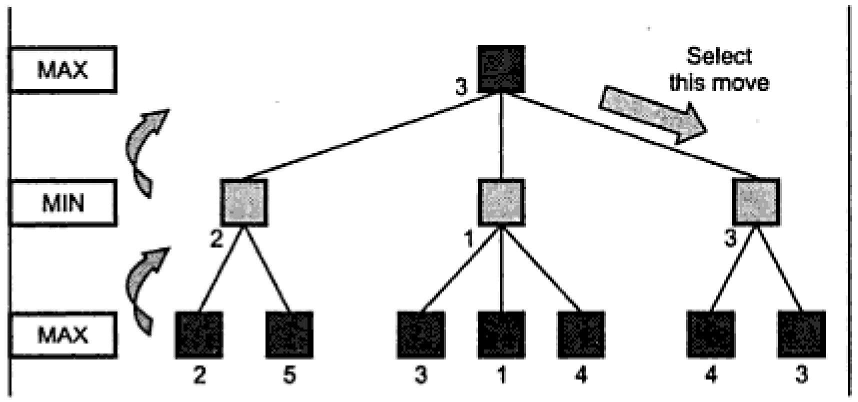
The procedure by which the scoring information passes up the game tree is called the MINIMAX procedures since the score at each node is either minimum or maximum of the scores at the nodes immediately below

One-ply search

In this fig since it is the maximizing search ply 8 is transferred upwards to A



Two-ply search



Static evaluation function

To play an entire game we need to combine search oriented and non-search oriented techniques. The idea way to use a search procedure to find a solution to the problem statement is to generate moves through the problem space until a goal state is reached. Unfortunately for games like chess even with a good plausible move generator, it is not possible to search until goal state is reached. In the amount of time available it is possible to generate the tree at the most 10 to 20 ply deep. Then in order to choose the best move, the resulting board positions must be compared to discover which is most advantageous. This is done using the static evaluation function. The static evaluation function evaluates individual board positions by estimating how much likely they are eventually to lead to a win.

The minimax procedure is a depth-first, depth limited search procedure.

- If the limit of search has reached, compute the static value of the current position relative to the appropriate layer as given below (maximizing or minimizing player). Report the result (value and path).
- If the level is minimizing level(minimizer‟s turn)
- Generate the successors of the current position. Apply MINIMAX to each of the successors. Return the minimum of the result.
- If the level is a maximizing level. Generate the successors of current position

Apply MINIMAX to each of these successors. Return the maximum of the result.

The maximum algorithm uses the following procedures

1. MOVEGEN(POS)

   It is plausible move generator. It returns a list of successors of „Pos‟.

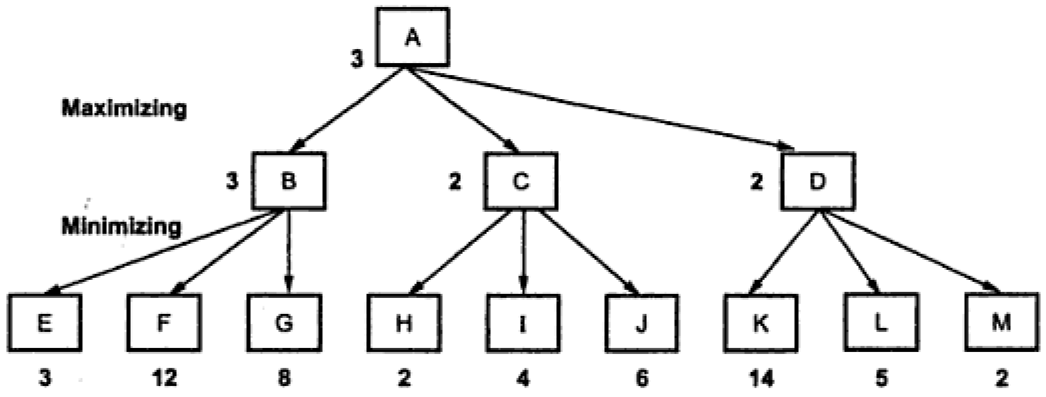
2. STSTIC (Pos, Depth)

   The static evaluation function that returns a number representing the goodness of „pos‟ from the current point of view.

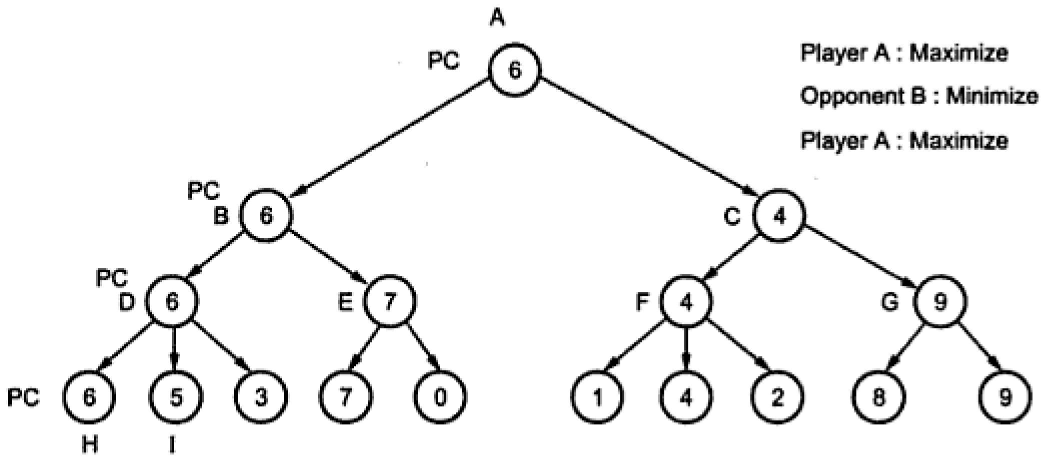
3. DEEP-ENOUGH

   It returns true if the search to be stopped at the current level else it returns false.

A MINIMAX example



Another example of minimax search procedure



In the above example, a Minimax search in a game tree is simulated. Every leaf has a corresponding value, which is approximated from player A‟s view point. When a path is chosen, the value of the child will be passed back to the parent. For example, the value for D is 6, which is the maximum value of its children, while the value for C is 4 which is the minimum value of F and G. In this example the best sequence of moves found by the maximizing/minimizing procedure is the path through nodes A, B, D and H, which is called the principal continuation. The nodes on the path are denoted as PC (principal continuation) nodes. For simplicity we can modify the game tree values slightly and use only maximization operations. The trick is to maximize the scores by negating the returned values from the children instead of searching for minimum scores and estimate the values at leaves from the player‟s own viewpoint

Alpha-beta cutoffs

The basic idea of alpha-beta cutoffs is “It is possible to compute the correct minimax decision without looking at every node in the search tree”. This is called pruning (allow us to ignore portions of the search tree that make no difference to the final choice).

The general principle of alpha-beta pruning is

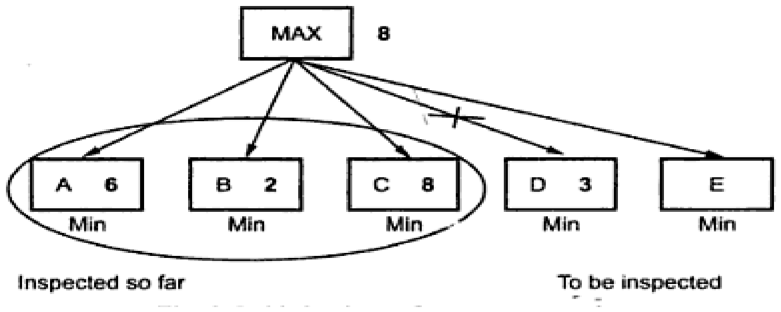
- Consider a node n somewhere in the tree, such that a player has a chance to move to this node.
- If player has a better chance m either at the parent node of n (or at any choice point further up) then n will never be reached in actual play.

When we are doing a search with alpha-beta cut-offs, if a node‟s value is too high, the minimizer will make sure it‟s never reached (by turning off the path to get a lower value). Conversely, if a node‟s value is too low, the maximizer will make sure it‟s never reached. This gives us the following definitions

- **Alpha:** the highest value that the maximize can guarantee himself by making some move at the current node OR at some node earlier on the path to this node.
- **Beta:** the lowest value that the minimizer can guarantee by making some move at the current node OR at some node earlier on the path to this node.

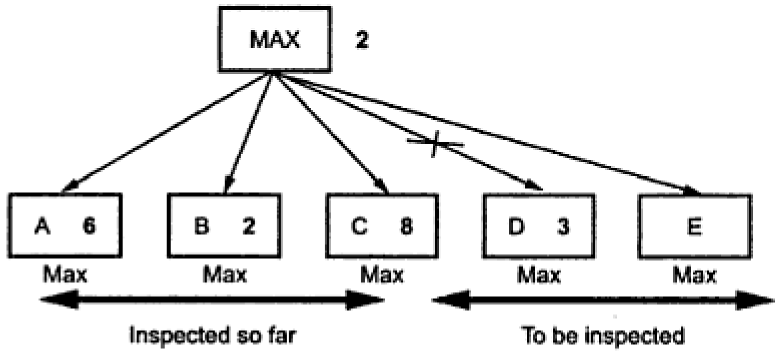
The maximize is constantly trying to push the alpha value up by finding better moves; the minimizer is trying to push the beta value down. If a node‟s value is between alpha and beta, then the players might reach it. At the beginning, at the root of the tree, we don‟t have any guarantees yet about what values the maximizer and minimizer can achieve. So we set beta to ∞ and alpha to -∞. Then as we move down the tree, each node starts with beta and alpha values passed down from its parent.

Consider a situation in which the MIN – children of a MAX-node have been partially inspected



Alpha-beta for a max node

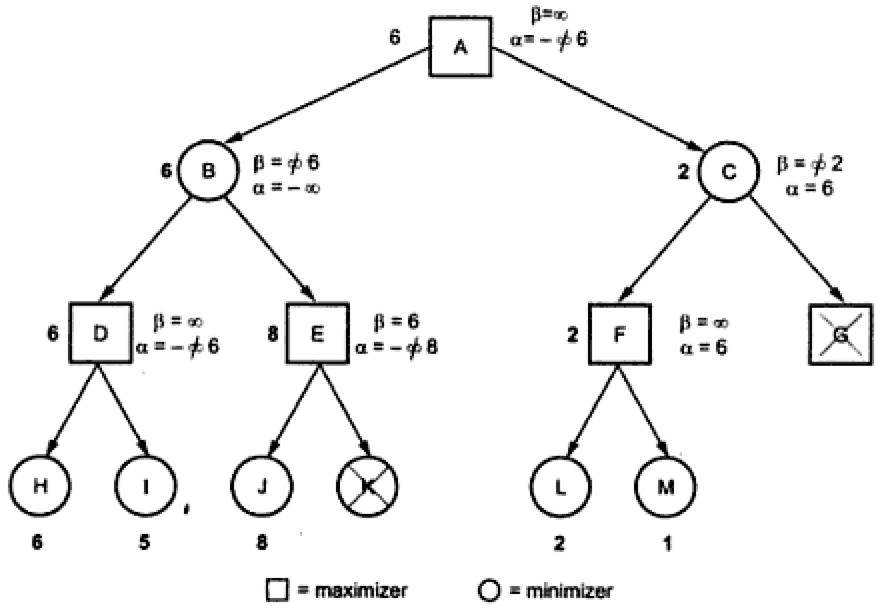
At this point the “tentative” value which is backed up so far of F is 8. MAX is not interested in any move which has a value of less than 8, since it is already known that 8 is the worst that MAX can do, so far. Thus the node D and all its descendent can be pruned or excluded from further exploration, since MIN will certainly go for a value of 3 rather than 8. Similarly for a MIN-node:



Alpha-beta for a min node

MIN is trying to minimize the game-value. So far, the value 2 is the best available form MIN‟s point of view. MIN will immediately reject node D, which can be stopped for further exploration.

In a game tree, each node represents a board position where one of the players gets to choose a move. For example, in the fig below look at the node C. As soon as we look at its left child, we realize that if the players reach node C, the minimizer can limit the utility to 2. But the maximize can get utility 6 by going to node B instead, so he would never let the game reach C. therefore we don‟t even have to look at C‟s other children



Tree with alpha-beta cut-offs

Initially at the root of the tree, there is no guarantee about what values the maximizer and minimizer can achieve. So beta is set to ∞ and alpha to -∞. Then as we move down the tree, each node starts with beta and alpha values passed down from its parent. It it‟s a maximize node, and then alpha is increased if a child value is greater than the current alpha value. Similarly, at a minimizer node, beta may be decreased. This is shown in the fig.

At each node, the alpha and beta values may be updated as we iterate over the node‟s children. At node E, when alpha is updated to a value of 8, it ends up exceeding beta. This is a point where alpha beta pruning is required we know the minimizer would never let the game reach this node so we don‟t have to look at its remaining children. In fact, pruning happens exactly when the alpha and beta lines hit each other in the node value.

Algorithm-Alpha-beta

```
int AlphaBeta(int depth, int alpha, int beta)
{
     if (depth == 0)
          return Evaluate();
     GenerateLegalMoves();
     while (MovesLeft()) {
          MakeNextMove();
          val = -AlphaBeta(depth - 1, -beta, -alpha);
          UnmakeMove();
          if (val >= beta)
               return beta;
          if (val > alpha)
               alpha = val;
     }
     return alpha;
}
```

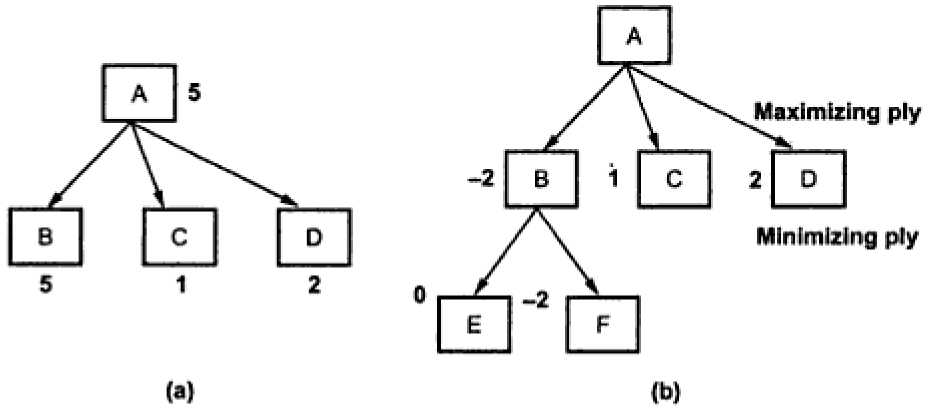
If the highlighted characters are removed, what is left is a min-max function. The function is passed the depth it should search, and –INIFINITY as alpha and +INIFINITY as beta.

```
val = AlphaBeta(5, -INFINITY, INFINITY);
```

This does a five-ply search

The Horizon effect

A potential problem in game tree search to a fixed depth is the horizon effect, which occurs when there is a drastic change in value immediately beyond the place where the algorithm stops searching. Consider the tree shown in the below fig.A. it has nodes A, B, C and D. at this level since it is a maximizing ply, the value which will passed up at A is 5.



Suppose node B is examined one more level as shown in fig B. then we see because of a minimizing ply value at B is -2 and hence the value passed to A is 2. This results in a drastic change in the situation. There are two proposed solutions to this problem, neither very satisfactory.

Secondary search

One proposed solution is to examine the search beyond the apparently best one to see if something is looming just over the horizon. In that case we can revert to the second-best move. Obviously then the second-best move has the same problem and there is not time to search beyond all possible acceptable moves.

Waiting for Quiescence

- If a position looks "dynamic", don‟t even bother to evaluate it.
- Instead, do a small secondary search until things calm down.
- E.g after capturing a piece, things look good, but this would be misleading if opponent was about to capture right back.
- In general such factors are called continuation heuristics

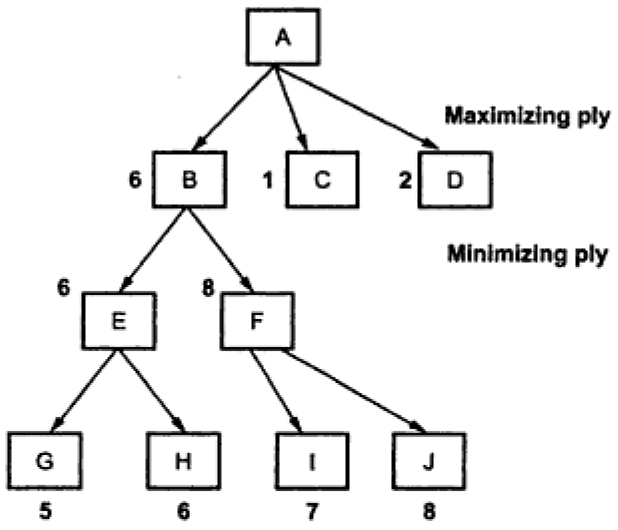


Fig above shows the further exploration of the tree in fig B. Here now since the tree is further explored, the value, which is passed to A is 6. Thus the situation calms down. This is called as waiting for quiescence. This helps in avoiding the horizon effect of a drastic change of values.

Iterative Deepening

Rather than searching to a fixed depth in the game tree, first search only single ply, then apply MINMAX to 2 ply, further 3 ply till the final goal state is searched. This is called as iterative deepening. Besides providing good control of time, iterative deepening is usually more efficient than an equivalent direct search. The reason is that the results of previous iterations can improve the move ordering of new iteration, which is critical for efficient searching.

2.2 Knowledge Representation

Representation and Mapping

- Problem solving requires large amount of knowledge and some mechanism for manipulating that knowledge.
- The Knowledge and the Representation are distinctentities, play a central but distinguishable roles in intelligent system.
    - **Knowledge** is a description of the world;

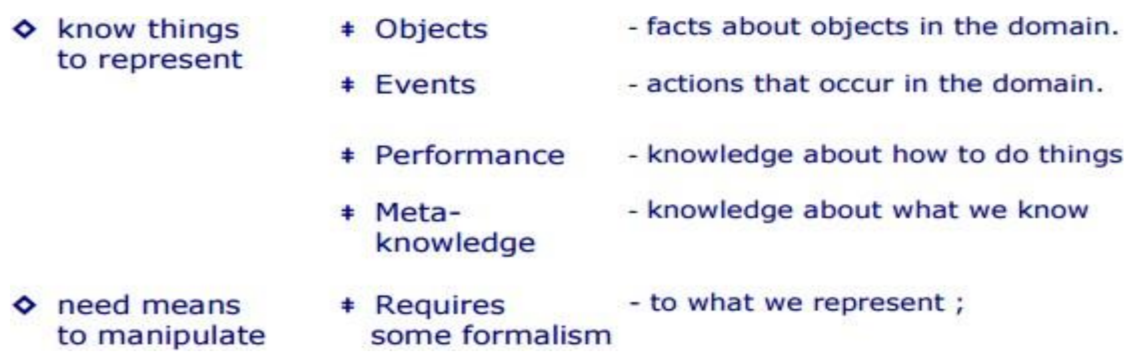
      it determines a *system's competence* by what it knows.
    - **Representation** is the way knowledge is encoded;

      it defines the *system's performance* in doing something.

- **Facts** Truths about the real world and what we represent. This can be regarded as the knowledge level

➢ In simple words, we *:*

- need to know about *things we want to represent* , and
- need some means by which *things we can manipulate*.



◇ know things to represent	⁑ Objects	- facts about objects in the domain.
	⁑ Events	- actions that occur in the domain.
	⁑ Performance	- knowledge about how to do things
	⁑ Meta-knowledge	- knowledge about what we know
◇ need means to manipulate	⁑ Requires some formalism	- to what we represent ;

Thus, knowledge representation can be considered at two levels :

➢ knowledge level at which facts are described, and

➢ symbol level at which the representations of the objects, defined in terms of symbols, can be manipulated in the programs.

Note : A good representation enables fast and accurate access to knowledge and understanding of the content.

Mapping between Facts and Representation

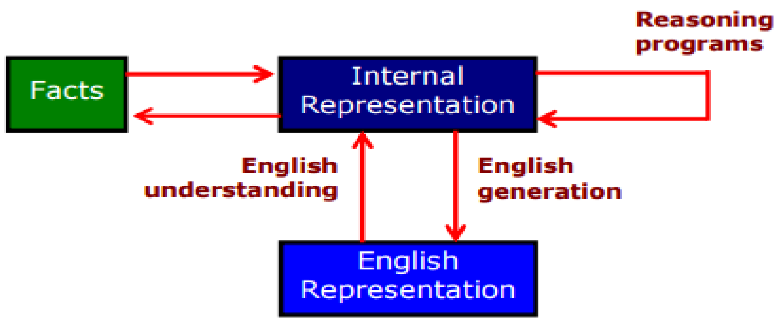
➢ Knowledge is a collection of *"facts"* from some domain.

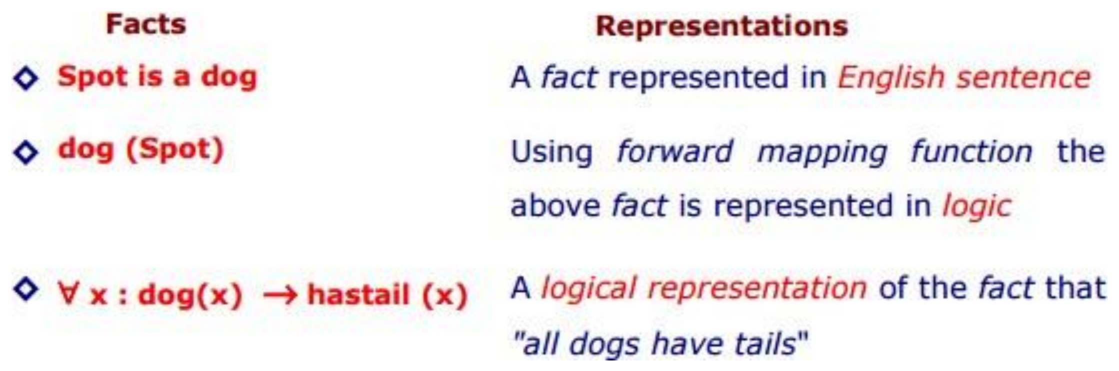
➢ We need a representation of *"facts"* that can be manipulated by a program. Normal English is insufficient, too hard currently for a computer program to draw inferences in natural languages.

➢ Thus some symbolic representation is necessary.

➢ Therefore, we must be able to map *"facts to symbols"* and *"symbols to facts"* using *forward and backward representation mapping.*

Example : Consider an English sentence



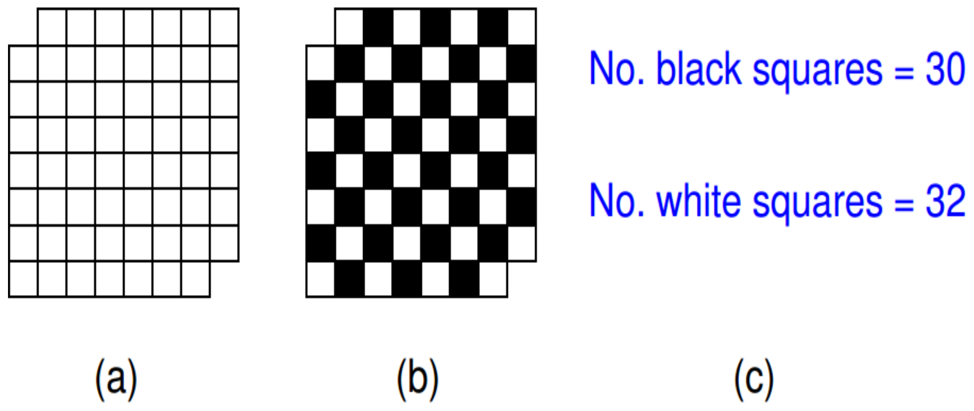


Facts	Representations
◇ Spot is a dog	A *fact* represented in *English sentence*
◇ dog (Spot)	Using *forward mapping function* the above *fact* is represented in *logic*
◇ ∀ x : dog(x) → hastail (x)	A *logical representation* of the *fact* that *"all dogs have tails"*

Now using deductive mechanism we can generate a new representation of object:

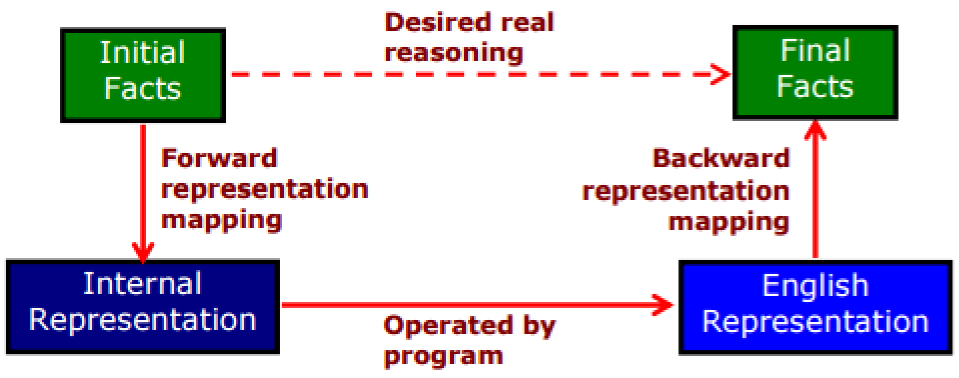
Hastail (Spot)	A new object representation
Spot has a tail [it is new knowledge]	Using backward mapping function to generate English sentence

- Good representation can make a reasoning program trivial
  - The Mutilated Checkerboard Problem: "Consider a normal checker board from which two squares, in opposite corners, have been removed. The task is to cover all the remaining squares exactly with dominoes, each of which covers two squares. No overlapping, either of dominoes on top of each other or of dominoes over the boundary of the mutilated board are allowed. Can this task be done?"



- ➢ Forward and Backward Representation

  The forward and backward representations are elaborated below



- The doted line on top indicates the abstract reasoning process that a program is intended to model.
- The solid lines on bottom indicate the concrete reasoning process that the program performs.

KR System Requirements

- ➢ A good knowledge representation enables fast and accurate access to knowledge and understanding of the content.
- ➢ A knowledge representation system should have following properties.

  Representational Adequacy the ability to represent all kinds of knowledge that are needed in that domain.

Inferential Adequacy the ability to manipulate the representational structures to derive new structure corresponding to new knowledge inferred from old.

Inferential Efficiency the ability to incorporate additional information into the knowledge structure that can be used to focus attention of the inference mechanisms in the most promising direction.

Acquisitional Efficiency the ability to acquire new knowledge using automatic methods whenever possible rather than reliance on human intervention

Note: To date no single system can optimizes all of the above properties.

2.3 Knowledge Representation Schemes

There are four types of Knowledge representation :

Relational, Inheritable, Inferential, and Declarative/Procedural.

- **Relational Knowledge :**
  - provides a framework to compare two objects based on equivalent attributes.
  - any instance in which two different objects are compared is a relational type of knowledge.
- **Inheritable Knowledge**
  - is obtained from associated objects.
  - it prescribes a structure in which new objects are created which may inherit all or a subset of attributes from existing objects.
- **Inferential Knowledge**
  - is inferred from objects through relations among objects.
  - e.g., a word alone is a simple syntax, but with the help of other words in phrase the reader may infer more from a word; this inference within linguistic is called semantics.
- **Declarative Knowledge**
  - a statement in which knowledge is specified, but the use to which that knowledge is to be put is not given.
  - e.g. laws, people's name; these are facts which can stand alone, not dependent on other knowledge;
- **Procedural Knowledge**

- a representation in which the control information, to use the knowledge, is embedded in the knowledge itself.
- e.g. computer programs, directions, and recipes; these indicate specific use or implementation;

Relational Knowledge

This knowledge associates elements of one domain with another domain.

- Relational knowledge is made up of objects consisting of attributes and their corresponding associated values.
- The results of this knowledge type is a mapping of elements among different domains.

The table below shows a simple way to store facts.

- The facts about a set of objects are put systematically in columns.
- This representation provides little opportunity for inference.

Table - Simple Relational Knowledge

Player	Height	Weight	Bats - Throws
Aaron	6-0	180	Right - Right
Mays	5-10	170	Right - Right
Ruth	6-2	215	Left - Left
Williams	6-3	205	Left - Right

- ✓ Given the facts it is not possible to answer simple question such as :

  " Who is the heaviest player ? ".

  but if a procedure for finding heaviest player is provided, then these facts will enable that procedure to compute an answer.
- ✓ We can ask things like who "bats – left" and "throws – right".

Inheritable Knowledge

- ➢ Here the knowledge elements inherit attributes from their parents.

- ➢ The knowledge is embodied in the design hierarchies found in the functional, physical and process domains. Within the hierarchy, elements inherit attributes from their parents, but in many cases not all attributes of the parent elements be prescribed to the child elements.
- ➢ The *inheritance* is a powerful form of inference, but not adequate. The basic KR needs to be augmented with inference mechanism.
- ➢ The KR in hierarchical structure, shown below, is called *"semantic network"* or a collection of *"frames" or "slot-and-filler structure"*. The structure shows property inheritance and way for insertion of additional knowledge.
- ➢ Property inheritance: The objects or elements of specific classes inherit attributes and values from more general classes. The classes are organized in a generalized hierarchy.

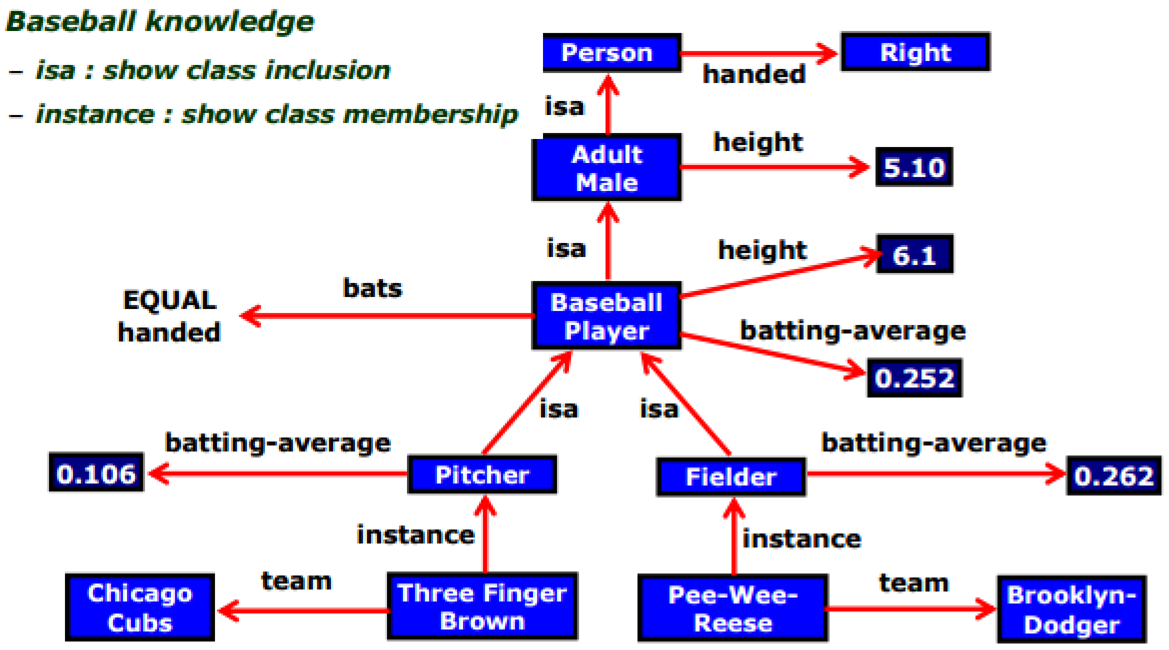


Fig. Inheritable knowledge representation (KR)

- ✓ The directed arrows represent *attributes* (*isa*, *instance*, *team*) originates at object being described and terminates at object or its value.
- ✓ The box nodes represents *objects* and *values* of the attributes.

- ➢ Viewing a node as a frame

Example :	Baseball-player
isa :	Adult-Male

Bates :	EQUAL handed
Height :	6.1
Batting-average :	0.252

- ➢ Algorithm : Property Inheritance

  Retrieve a value V for an attribute A of an instance object O

Steps to follow:

1. Find object **O** in the knowledge base.
2. If there is a value for the attribute **A** then report that value.
3. Else, if there is a value for the attribute instance; If not, then fail.
4. Else, move to the node corresponding to that value and look for a value for the attribute **A**; If one is found, report it.
5. Else, do until there is no value for the "**isa**" attribute or until an answer is found :
   (a) Get the value of the "**isa**" attribute and move to that node.
   (b) See if there is a value for the attribute **A**; If yes, report it.

- ➢ This algorithm is simple. It describes the basic mechanism of inheritance. It does not say what to do if there is more than one value of the instance or "isa" attribute.
- ➢ This can be applied to the example of knowledge base illustrated, in the previous slide, to derive answers to the following queries :
  - ✓ team (Pee-Wee-Reese) = Brooklyn–Dodger
  - ✓ batting–average(Three-Finger-Brown) = 0.106
  - ✓ height (Pee-Wee-Reese) = 6.1
  - ✓ bats (Three Finger Brown) = right

Inferential Knowledge

- ➢ This knowledge generates new information from the given information.
- ➢ This new information does not require further data gathering form source, but does require analysis of the given information to generate new knowledge.

Example :

– given a set of relations and values, one may infer other values or relations.

- a predicate logic (a mathematical deduction) is used to infer from a set of attributes.
- inference through predicate logic uses a set of logical operations to relate individual data.
- the symbols used for the logic operations are :

" → " (implication), " ¬ " (not), " V " (or), " Λ " (and),
" ∀ " (for all), " ∃ " (there exists).

Examples of predicate logic statements :

1. *"Wonder"* is a name of a dog :	dog (wonder)
2. All dogs belong to the class of animals :	∀ x : dog (x) → animal(x)
3. All animals either live on land or in water :	∀ x : animal(x) → live (x, land) V live (x, water)

From these three statements we can infer that :

" Wonder lives either on land or on water."

Note : If more information is made available about these objects and their relations, then more knowledge can be inferred.

Declarative/Procedural Knowledge

Differences between Declarative/Procedural knowledge is not very clear.

Declarative knowledge :

Here, the knowledge is based on declarative facts about *axioms* and *domains* .

- ✓ axioms are assumed to be true unless a counter example is found to invalidate them.
- ✓ domains represent the physical world and the perceived functionality.
- ✓ axiom and domains thus simply exists and serve as declarative statements that can stand alone.

Procedural knowledge:

Here, the knowledge is a mapping process between domains that specify “what to do when” and the representation is of “how to make it” rather than “what it is”. The procedural knowledge :

- ✓ may have inferential efficiency, but no inferential adequacy and acquisitional efficiency.
- ✓ are represented as small programs that know how to do specific things, how to proceed.

Example : A parser in a natural language has the knowledge that a noun phrase may contain articles, adjectives and nouns. It thus accordingly call routines that know how to process articles, adjectives and nouns.

Issues in Knowledge Representation

- ➢ The fundamental goal of Knowledge Representationis to facilitate inference (conclusions) from knowledge.
- ➢ The issues that arise while using KR techniques are many. Some of these are explained below.
  - ✓ Important Attributes :

    Any attribute of objects so basic that they occur in almost every problem domain ?
  - ✓ Relationship among attributes:

    Any important relationship that exists among object attributes ?
  - ✓ Choosing Granularity :

    At what level of detail should the knowledge be represented ?
  - ✓ Set of objects :

    How sets of objects be represented ?
  - ✓ Finding Right structure :

    Given a large amount of knowledge stored, how can relevant parts be accessed ?
- ➢ **Important Attributes**
  - ✓ There are attributes that are of general significance.
  - ✓ There are two attributes "instance" and "isa", that are of general importance. These attributes are important because they support property inheritance.
- ➢ **Relationship among Attributes**
  - ✓ The attributes to describe objects are themselves entities they represent.
  - ✓ The relationship between the attributes of an object, independent of specific knowledge they encode, may hold properties like:

✓ Inverses, existence in an isa hierarchy, techniques for reasoning about values and single valued attributes.

- **Inverses :**

This is about consistency check, while a value is added to one attribute. The entities are related to each other in many different ways. The figure shows attributes (isa, instance, and team), each with a directed arrow, originating at the object being described and terminating either at the object or its value.

There are two ways of realizing this:

- first, represent two relationships in a single representation; e.g., a logical representation, team(Pee-Wee-Reese, Brooklyn–Dodgers), that can be interpreted as a statement about Pee-Wee-Reese or Brooklyn–Dodger.
- second, use attributes that focus on a single entity but use them in pairs, one the inverse of the other; for e.g., one, team = Brooklyn– Dodgers , and the other, team = Pee-Wee-Reese, ...

This second approach is followed in semantic net and frame-based systems, accompanied by a knowledge acquisition tool that guarantees the consistency of inverse slot by checking, each time a value is added to one attribute then the corresponding value is added to the inverse.

- Existence in an "isa" hierarchy

  This is about generalization-specialization, like, classes of objects and specialized subsets of those classes. There are attributes and specialization of attributes.

  Example: the attribute "height" is a specialization of general attribute "physical-size" which is, in turn, a specialization of "physical-attribute". These generalization-specialization relationships for attributes are important because they support inheritance.

- Techniques for reasoning about values

  This is about reasoning values of attributes not given explicitly. Several kinds of information are used in reasoning, like,

height	: must be in a unit of length,
age	: of person can not be greater than the age of person's parents.

  The values are often specified when a knowledge base is created.

- Single valued attributes

This is about a specific attribute that is guaranteed to take a unique value.

Example : A baseball player can at time have only a single height and be a member of only one team. KR systems take different approaches to provide support for single valued attributes.

- ➢ Choosing Granularity

  What level should the knowledge be represented and what are the primitives ?

  - ✓ Should there be a small number or should there be a large number of low-level primitives or High-level facts.
  - ✓ High-level facts may not be adequate for inference while Low-level primitives may require a lot of storage.

    Example of Granularity :

    - Suppose we are interested in following facts

      John spotted Sue.
    - This could be represented as

      Spotted (agent(John), object (Sue))
    - Such a representation would make it easy to answer questions such are

      Who spotted Sue ?
    - Suppose we want to know

      Did John see Sue ?
    - Given only one fact, we cannot discover that answer.
    - We can add other facts, such as

      Spotted (x , y) -> saw (x , y)
    - We can now infer the answer to the question.

- ➢ Set of Objects

  - ✓ Certain properties of objects that are true as member of a set but not as individual;

    Example : Consider the assertion made in the sentences "there are more sheep than people in Australia", and "English speakers can be found all over the world."

- ✓ To describe these facts, the only way is to attach assertion to the sets representing people, sheep, and English.
- ✓ The reason to represent sets of objects is :

  If a property is true for all or most elements of a set, then it is more efficient to associate it once with the set rather than to associate it explicitly with every elements of the set.
- ✓ This is done in different ways :
  - in logical representation through the use of universal quantifier, and
  - in hierarchical structure where node represent sets, the inheritance propagate set level assertion down to individual.

Example: assert large (elephant); Remember to make clear distinction between,

- ✓ whether we are asserting some property of the set itself, means, the set of elephants is large, or
- ✓ asserting some property that holds for individual elements of the set , means, any thing that is an elephant is large.

There are three ways in which sets may be represented :

- ✓ Name, as in the example. Inheritable KR, the node - Baseball- Player and the predicates as Ball and Batter in logical representation.
- ✓ Extensional definition is to list the numbers, and
- ✓ In tensional definition is to provide a rule, that returns true or false depending on whether the object is in the set or not.

- ➢ **Finding Right Structure**
  - ✓ Access to right structure for describing a particular situation.
  - ✓ It requires, selecting an initial structure and then revising the choice. While doing so, it is necessary to solve following problems :
    - how to perform an initial selection of the most appropriate structure.
    - how to fill in appropriate details from the current situations.
    - how to find a better structure if the one chosen initially turns out not to be appropriate.
    - what to do if none of the available structures is appropriate.

- when to create and remember a new structure.

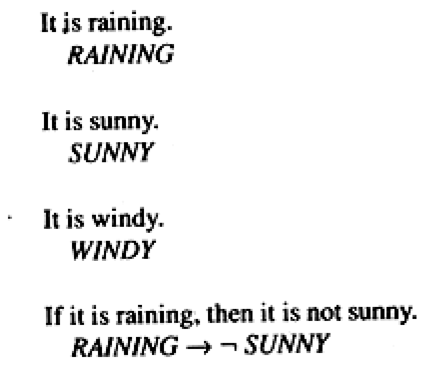
✓ There is no good, general purpose method for solving all these problems. Some knowledge representation techniques solve some of them.

CHAPTER 2

Knowledge Representation using predicate logic

2.4 Representing Simple Facts in Logic

AI system might need to represent knowledge. Propositional logic is one of the fairly good forms of representing the same because it is simple to deal with and a decision procedure for it exists. Real-world facts are represented as logical propositions and are written as well-formed formulas (wff's) in propositional logic, as shown in Figure below. Using these propositions, we may easily conclude it is not sunny from the fact that its raining. But contrary to the ease of using the propositional logic there are its limitations. This is well demonstrated using a few simple sentence like:



It is raining.
RAINING

It is sunny.
SUNNY

It is windy.
WINDY

If it is raining, then it is not sunny.
RAINING $\rightarrow \neg$ *SUNNY*

Some simple facts in Propositional logic

Socrates is a man.

We could write:

SOCRATESMAN

But if we also wanted to represent

Plato is a man.

we would have to write something such as:

PLATOMAN

which would be a totally separate assertion, and we would not be able to draw any conclusions about similarities between Socrates and Plato. It would be much better to represent these facts as:

MAN(SOCRATES)

MAN(PLATO)

since now the structure of the representation reflects the structure of the knowledge itself. But to do that, we need to be able to use predicates applied to arguments. We are in even more difficulty if we try to represent the equally classic sentence

All men are mortal.

We could represent this as:

MORTALMAN

But that fails to capture the relationship between any individual being a man and that individual being a mortal. To do that, we really need variables and quantification unless we are willing to write separate statements about the mortality of every known man.

Let's now explore the use of predicate logic as a way of representing knowledge by looking at a specific example. Consider the following set of sentences:

1 . Marcus was a man.

2. Marcus was a Pompeian.

3. All Pompeians were Romans.

4. Caesar was a ruler.

5. All Romans were either loyal to Caesar or hated him.

6. Everyone is loyal to someone.

7. People only try to assassinate rulers they are not loyal to.

8. Marcus tried to assassinate Caesar.

The facts described by these sentences can be represented as a set of wff's in predicate logic as follows:

1. Marcus was a man.

man(Marcus)

Although this representation fails to represent the notion of past tense (which is clear in the English sentence), it captures the critical fact of Marcus being a man. Whether this omission is acceptable or not depends on the use to which we intend to put the knowledge.

2. Marcus was a Pompeian.

Pompeian(Marcus)

3.All Pompeians were Romans.

$\forall x : \text{Pompeian}(x) \rightarrow \text{Roman}(x)$

4.Caesar was a ruler.

ruler(Caesar)

Since many people share the same name, the fact that proper names are often not references to unique individuals, overlooked here. Occasionally deciding which of several people of the same name is being referred to in a particular statement may require a somewhat more amount of knowledge and logic.

5. All Romans were either loyal to Caesar or hated him.

∀x: Roman(x)→ loyalto(x, Caesar) V hate(Caesar)

Here we have used the inclusive-or interpretation of the two types of Or supported by English language. Some people will argue, however, that this English sentence is really stating an exclusive-or. To express that. we would have to write:

∀x: Roman(x) → [(loyalto(x, Caesar) V hate(x, Caesar))∧

Not (loyalto(x, Caesar) ∧hate(x, Caesar))]

6. Everyone is loyal to someone.

∀x:∃y : loyalto(x,y)

The scope of quantifiers is a major problem that arises when trying to convert English sentences into logical statements. Does this sentence say, as we have assumed in writing the logical formula above, that for each person there exists someone to whom he or she is loyal, possibly a different someone for everyone? Or does it say that there is someone to whom everyone is loyal?

$$\begin{tikzcd}
\neg \mathit{loyalto}(\mathit{Marcus}, \mathit{Caesar}) \\
\begin{array}{l} \mathit{person}(\mathit{Marcus}) \land \\ \mathit{ruler}(\mathit{Caesar}) \land \\ \mathit{tryassassinate}(\mathit{Marcus}, \mathit{Caesar}) \end{array} \arrow[u, "\text{(7, substitution)}"'] \\
\begin{array}{l} \mathit{person}(\mathit{Marcus}) \\ \mathit{tryassassinate}(\mathit{Marcus}, \mathit{Caesar}) \end{array} \arrow[u, "(4)"'] \\
\mathit{person}(\mathit{Marcus}) \arrow[u, "(8)"']
\end{tikzcd}$$

An Attempt to Prove not loyal to(Marcus,Caesar)

7. People only try to assassinate rulers they are not loyal to.

∀x : ∀y : person(x) ∧ ruler(y) ∧ tryassasinate(x,y) → ¬loyalto(x,y)

8.Like the previous one this sentence too is ambiguous which may lead to more than one conclusion. The usage of “try to assassinate” as a single predicate gives us a fairly simple representation with which we can reason about trying to assassinate. But there might be connections as try to assassinate and not actually assassinate could not be made easily.

9. Marcus tried to assassinate Caesar.

tryassasinate (Marcus,Caesar)

now, say suppose we wish to answer the following question:

Was Marcus loyal to Caesar?

What we do is start reasoning backward from the desired goal which is represented in predicate logic as:

$\neg$loyalto(Marcus, Caesar)

Figure 4.2 shows an attempt to produce a proof of the goal by reducing the set of necessary but as yet unattained goals to the empty sets. The attempts fail as we do not have any statement to prove person(Marcus). But the problem is solved just by adding an additional statement i.e.

10. All men are people.

$\forall$x : man(x) $\rightarrow$ person(x)

Now we can satisfy the last goal and produce a proof that Marcus was not loyal to Caesar.

2.4.1 Representing Instance and isa relationships

- Knowledge can be represented as classes, objects, attributes and Super class and sub class relationships.
- Knowledge can be inference using property inheritance. In this elements of specific classes inherit the attributes and values.
- Attribute instance is used to represent the relationship "Class membership " (element of the class)
- Attribute isa is used to represent the relationship "Class inclusion" (super class, sub class relationship)

Three ways of representing class membership

1. $man(Marcus)$
2. $Pompiean(Marcus)$
3. $\forall\ x: Pompiean(x) \rightarrow Roman(x)$
4. $ruler(Caesar)$
5. $\forall x: Roman(x) \rightarrow loyalto(x,Caesar) \vee hate(x,Caesar)$

1. $instance(Marcus,man)$
2. $instance(Marcus, Pompiean)$
3. $\forall x: instance(x, Pompiean) \rightarrow instance(x,Roman)$
4. $instance(Caesar,ruler)$
5. $\forall x: instance(x, Roman) \rightarrow loyalto(x,Caesar) \vee hate(x,Caesar)$

1. $instance(Marcus,man)$
2. $instance(Marcus, Pompiean)$
3. $isa(Pompiean,Roman)$
4. $instance(Caesar,ruler)$
5. $\forall x: instance(x, Roman) \rightarrow loyalto(x,Caesar)\ hate(x,Caesar)$
6. $\forall\ x: \forall\ y: \forall\ z: instance(x,y) \wedge isa(y,z) \rightarrow instance(x,z)$

These examples illustrate two points. The first is fairly specific. It is that, although class and superclass memberships are important facts that need to be represented those memberships need not be represented with predicates labelled instance and isa. In fact, in a logical framework it is usually unwieldy to do that, and instead unary predicates corresponding to the classes are often used. The second point is more general. There are usually several different ways of representing a given fact within a particular representational framework, be it logic or anything else. The choice depends partly on which deductions need to be supported most efficiently and partly on taste. The only important thing is that within a particular knowledge base consistency of representation is critical. Since any particular inference rule is designed to work on one particular form of representation, it is necessary that all the knowledge to which that rule is intended to apply be in the form that the rule demands. Many errors in the reasoning performed by knowledge-based programs are the result of inconsistent representation decisions. The moral is simply to be careful.

2.4.2 Computable functions and predicates

- Some of the computational predicates like Less than, Greater than used in knowledge representation.
- It generally return true or false for the inputs.
  Examples: Computable predicates

  gt(1,0) or lt(0,1)

gt(5,4) or gt(4,5)

Computable functions: gt(2+4, 5)

Consider the following set of facts, again involving Marcus:

1. marcus was a man

man(Marcus)

2. Marcus was a pompeian

Pompeian(Marcus)

3. Marcus was born in 40 A.D

born(marcus, 40)

4. All men are mortal

∀x: men(x)→ mortal(x)

5. All Pompeians died when the volcano erupted in 79 A.D

erupted(volcano,79) & x :pompeian(x)→died(x, 79)

6. No mortal lives longer than150 years

∀x: ∀t1: ∀t2: mortal(x) & born(x,t1) & gt(t2-t1,150)→ dead(x,t1)

7. It is Now 1991

Now=1991

8. Alive means not dead

∀x: ∀t: [alive(x,t) →~dead(x,t)] & [~dead(x,t)→alive(x,t)]

9. If someone dies then he is dead at all later times

∀x: ∀t1: ∀t2: died(x,t1) & gt(t2,t1)→ dead(x1,t2)

This representation says that one is dead in all years after the one in which one died. It ignores the question of whether one is dead in the year in which one died.

1. man(Marcus)

2. Pompeian(Marcus)

3. born(marcus, 40)

4. ∀x: men(x)→ mortal(x)

5. ∀:pompeian(x)→died(x, 79)

6. erupted(volcano,79)

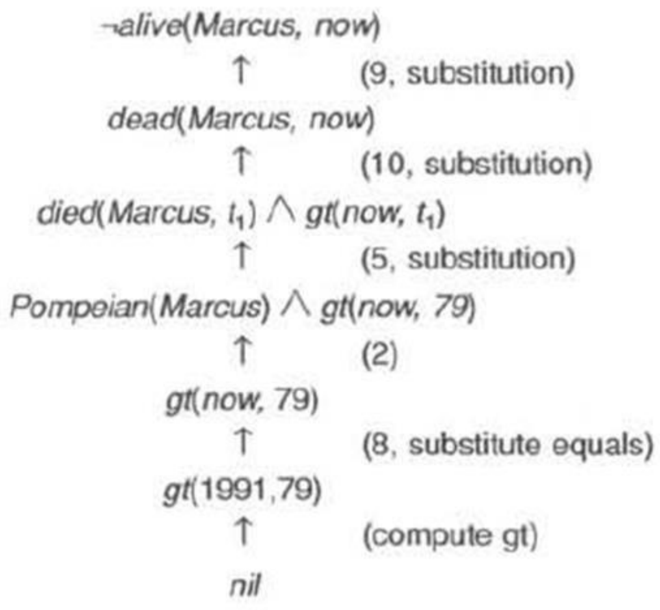
7. ∀ x: ∀t1: ∀t2: mortal(x) & born(x,t1) & gt(t2-t1,150)→

dead(x,t1)

8. Now=1991

9. ∀x: ∀ t: [alive(x,t) →~dead(x,t)] &[~dead(x,t)→alive(x,t)]

10. ∀ x: ∀t1: ∀t2: died(x,t1) & gt(t2,t1)→ dead(x1,t2)



One Way of Proving That Marcus Is Dead

$\neg alive(Marcus, now)$

$\uparrow$ (9, substitution)

$dead(Marcus, now)$

$\uparrow$ (7, substitution)

$mortal(Marcus) \wedge$
$born(Marcus, t_1) \wedge$
$gt(now - t_1, 150)$

$\uparrow$ (4, substitution)

$man(Marcus) \wedge$
$born(Marcus, t_1) \wedge$
$gt(now - t_1, 150)$

$\uparrow$ (1)

$born(Marcus, t_1) \wedge$
$gt(now - t_1, 150)$

$\uparrow$ (3)

$gt(now - 40, 150)$

$\uparrow$ (8)

$gt(1991 - 40, 150)$

$\uparrow$ (compute minus)

$gt(1951, 150)$

$\uparrow$ (compute gt)

nil

Another Way of Proving That Marcus is Dead

Two things should be clear from the proofs we have just shown:

- Even very simple conclusions can require many steps to prove.
- A variety of processes, such as matching, substitution, and application of modus ponens are involved in the production of a proof, This is true even for the simple statements we are using, It would be worse if we had implications with more than a single term on the right or with complicated expressions involving ands and ors on the left.

Disadvantage:

- ➢ Many steps required to prove simple conclusions
- ➢ Variety of processes such as matching and substitution used to prove simple conclusions

2.5 Resolution

- ➢ Resolution is a proof procedure by refutation.
- ➢ To prove a statement using resolution it attempt to show that the negation of that statement.

Algorithm: Convert to Clause Form

1. Eliminate →, using the fact that a → b is equivalent to ¬ a V b. Performing this transformation on the wff given above yields

$\forall x: \neg$ [Roman(x) Ʌ know(x, Marcus)] V

[hate(x, Caesar) V ($\forall y : \neg(\exists z$: hate(y, z)) V thinkcrazy(x, y))]

2. Reduce the scope of each ¬ to a single term, using the fact that ¬ (¬ p) = p, deMorgan's laws [which say that ¬ (a Ʌ b) = ¬ a V ¬ b and ¬ (a V b) = ¬ a Ʌ ¬ b], and the standard correspondences between quantifiers [$\neg \forall x$: P(x) = $\exists x$: ¬ P(x) and ¬ $\exists x$: P(x) = $\forall$ x: ¬P(x)]. Performing this transformation on the wff from step 1 yields

$\forall x$: [¬ Roman(x) V ¬ know(x, Marcus)] V

[hate(x, Caesar) V ($\forall y$: $\forall z$: ¬ hate(y, z) V thinkcrazy(x, y))]

3. Standardize variables so that each quantifier binds a unique variable. Since variables are just dummy names, this process cannot affect the truth value of the wff. For example, the formula

$\forall x$: P(x) V $\forall x$: Q(x)

would be converted to

$\forall x$: P(x) V $\forall y$: Q(y)

4. Move all quantifiers to the left of the formula without changing their relative order. This is possible since there is no conflict among variable names. Performing this operation on the formula of step 2, we get

$\forall x$: $\forall y$: Az: [¬ Roman(x) V ¬ know(x, Marcus)] V

[hate(x, Caesar) V (¬ hale(y, z) V thinkcrazy(x, y))]

At this point, the formula is in what is known as prenex normal form. It consists of a prefix of quantifiers followed by a matrix, which is quantifier-free.

5. Eliminate existential quantifiers. A formula that contains an existentially quantified variable asserts that there is a value that can be substituted for the variable that makes the formula true. We can eliminate the quantifier by substituting for the variable a reference to a function that produces the desired value. Since we do not necessarily know how to produce the value, we must create a new function name for every such replacement. We make no assertions about these functions except that they must exist. So, for example, the formula

$\exists y$: President(y)

can be transformed into the formula

President(S1)

where S1 is a function with no arguments that somehow produces a value that satisfies President. If existential quantifiers occur within the scope of universal quantifiers, then the value that satisfies the predicate may depend on the values of the universally quantified variables. For example, in the formula

∀x: ∃y: father-of(y, x)

the value of y that satisfies father-of depends on the particular value of x. Thus we must generate functions with the same number of arguments as the number of universal quantifiers in whose scope the expression occurs. So this example would be transformed into

∀x: father-of(S2(x), x)

These generated functions are called Skolem functions. Sometimes ones with no arguments are called Skolem constants.

6. Drop the prefix. At this point, all remaining variables are universally quantified, so the prefix can just be dropped and any proof procedure we use can simply assume that any variable it sees is universally quantified. Now the formula produced in step 4 appears as

[¬ Roman(x) V ¬ know(x, Marcus)] V

[hate(x, Caesar) V (¬ hate(y, z) V thinkcrazy(x, y))]

7. Convert the matrix into a conjunction of disjuncts. In the case or our example, since there are no and‟s, it is only necessary to exploit the associative property of or [i.e., (a Ʌ b) V c = (a V c) Ʌ (b Ʌ c)] and simply remove the parentheses, giving

¬ Roman(x) V ¬ know(x, Marcus) V

hate(x, Caesar) V ¬ hate(y, z) V thinkcrazy(x, y)

However, it is also frequently necessary to exploit the distributive property [i.e. , (a Ʌ b) V c = (a V c) Ʌ (b V c)]. For example, the formula

(winter Ʌ wearingboots) V (summer Ʌ wearingsandals)

Becomes, after one application of the rule

[winter V (summer Ʌ wearingsandals)]

Ʌ [wearingboots V (summer Ʌ wearingsandals)]

And then, after a second application, required since there are still conjuncts joined by OR's, (winter V summer) Λ

(winter V wearingsandals) Λ

(wearingboots V summer) Λ

(wearingboots V wearingsandals)

8. Create a separate clause corresponding to each conjunct. In order for a wff to be true, all the clauses that are generated from it must be true. If we are going to be working with several wff‟s, all the clauses generated by each of them can now be combined to represent the same set of facts as were represented by the original wff's.

9. Standardize apart the variables in the set of clauses generated in step 8. By this we mean rename the variables so that no two clauses make reference to the same variable. In making this transformation, we rely on the fact that

(∀x: *P(x)* Λ *Q(x))* = ∀x: *P(x)* Λ ∀x: *Q(x)*

Thus since each clause is a separate conjunct and since all the variables are universally quantified, there need be no relationship between the variables of two clauses, even if they were generated from the same wff.

Performing this final step of standardization is important because during the resolution procedure it is sometimes necessary to instantiate a universally quantified variable (i.e., substitute for it a particular value). But, in general, we want to keep clauses in their most general form as long as possible. So when a variable is instantiated, we want to know the minimum number of substitutions that must be made to preserve the truth value of the system.

After applying this entire procedure to a set of wff's, we will have a set of clauses, each of which is a disjunction of *literals.* These clauses can now be exploited by the resolution procedure to generate proofs.

2.5.1 Resolution in Propositional Logic

In propositional logic, the procedure for producing a proof by resolution of proposition P with respect to a set of axioms F is the following.

ALGORITHM: PROPOSITIONAL RESOLUTION

1. Convert all the propositions of F to clause form

2. Negate P and convert the result to clause form. Add it to the set of clauses obtained in step 1.

3. Repeat until either a contradiction is found or no progress can be made:

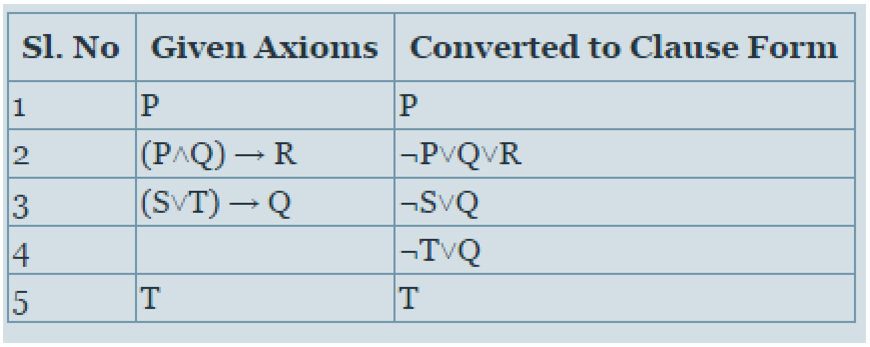
a) Select two clauses. Call these the parent clauses.

b) Resolve them together. The resulting clause, called the resolvent, will be the disjunction of all of the literals of both of the parent clauses with the following exception: If there are any pairs of literals L and ¬L such that one of the parent clauses contains L and the other contains ¬L, then select one such pair and eliminate both L and ¬L from the resolvent.

c) If the resolvent is the empty clause, then a contradiction has been found. If it is not then add it to the set of clauses available to the procedure.

Suppose we are given the axioms shown in the first column of Table 1 and we want to prove R.First we convert the axioms to clause which is already in clause form. Then we begin selecting pairs of clauses to resolve together. Although any pair of clauses can be resolved, only those pairs that contain complementary literals will produce a resolvent that is likely to lead to the goal of sequence of resolvents shown in figure 1. We begin by resolving with the clause ɷR since that is one of the clauses that must be involved in the contradiction we are trying to find.

One way of viewing the resolution process is that it takes a set of clauses that are all assumed to be true and based on information provided by the others, it generates new clauses that represent restrictions on the way each of those original clauses can be made true. A contradiction occurs when a clause becomes so restricted that there is no way it can be true. This is indicated by the generation of the empty clause.



Sl. No	Given Axioms	Converted to Clause Form
1	P	P
2	(P∧Q) → R	¬P∨Q∨R
3	(S∨T) → Q	¬S∨Q
4		¬T∨Q
5	T	T

Table 1 Some Facts in Propositional Logic

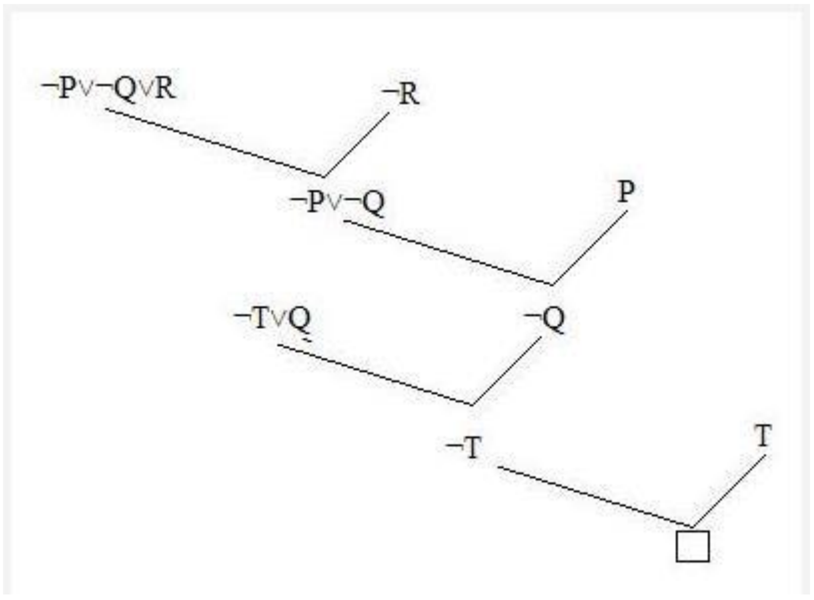


Figure : Resolution In Propositional Logic

2.5.3 UNIFICATION ALGORITHM

In propsoitional logic it is easy to determine that two literals can not both be true at the same time. Simply look for L and ~L . In predicate logic, this matching process is more complicated, since bindings of variables must be considered.

For example man (john) and man(john) is a contradiction while man (john) and man(Himalayas) is not. Thus in order to determine contradictions we need a matching procedure that compares two literals and discovers whether there exist a set of substitutions that makes them identical . There is a recursive procedure that does this matching . It is called Unification algorithm.

In Unification algorithm each literal is represented as a list, where first element is the name of a predicate and the remaining elements are arguments. The argument may be a single element (atom) or may be another list. For example we can have literals as

(tryassassinate Marcus Caesar)

(tryassassinate Marcus (ruler of Rome))

To unify two literals , first check if their first elements re same. If so proceed. Otherwise they can not be unified. For example the literals

(try assassinate Marcus Caesar)

(hate Marcus Caesar)

Can not be Unfied. The unification algorithm recursively matches pairs of elements, one pair at a time. The matching rules are :

i) Different constants , functions or predicates can not match, whereas identical ones can.

ii) A variable can match another variable , any constant or a function or predicate expression, subject to the condition that the function or [predicate expression must not contain any instance of the variable being matched (otherwise it will lead to infinite recursion).

iii) The substitution must be consistent. Substituting y for x now and then z for x later is inconsistent. (a substitution y for x written as y/x)

The Unification algorithm is listed below as a procedure UNIFY (L1, L2). It returns a list representing the composition of the substitutions that were performed during the match. An empty list NIL indicates that a match was found without any substitutions. If the list contains a single value F, it indicates that the unification procedure failed.

The empty list, NIL, indicates that a match was found without any substitutions. The list consisting of the single value FAIL indicates that the unification procedure failed.

```
Algorithm: Unify(L1, L2)
I. If L1 or L2 are both variables or constants, then:
    (a) If L1 and L2 are identical, then return NIL.
    (b) Else if L1 is a variable, then if L1 occurs in L2 then return {FAIL}, else return (L2/L1).
    (c) Else if L2 is a variable, then if L2 occurs in L1 then return {FAIL} , else return (L1/L2).
    (d) Else return {FAIL}.
2. If the initial predicate symbols in L1 and L2 are not identical, then return {FAIL}.
3. If LI and L2 have a different number of arguments, then return {FAIL}.
4. Set SUBST to NIL. (At the end of this procedure, SUBST will contain all the substitutions used to unify L1 and L2.)
5. For i ← 1 to number of arguments in L1 :
    (a) Call Unify with the ith argument of L1 and the ith argument of L2, putting result in S.
    (b) If S contains FAIL then return {FAIL}.
    (c) If S is not equal to NIL then:
        (i) Apply S to the remainder of both L1 and L2.
```

(ii) SUBST: = APPEND(S, SUBST).

6. Return SUBST.

2.5.4 Resolution in Predicate Logic

ALGORITHM: RESOLUTION IN PREDICATE LOGIC

1. Convert all the statements of F to clause form

2. Negate P and convert the result to clause form. Add it to the set of clauses obtained in step 1.

3. Repeat until either a contradiction is found or no progress can be made or a predetermined amount of effort has been expended:

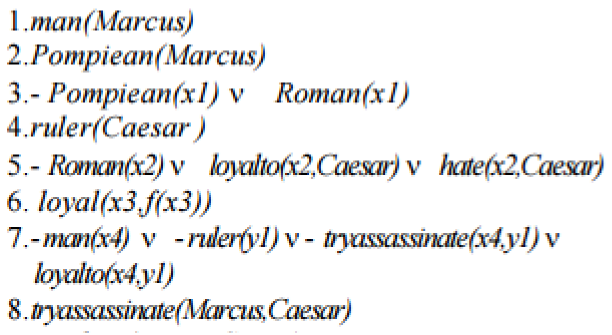
a) Select two clauses. Call these the parent clauses.

b) Resolve them together. The resulting clause, called the resolvent, will be the disjunction of all of the literals of both the parent clauses with appropriate substitutions performed and with the following exception: If there is one pair of literals T1 and T2 such that one of the parent clauses contains T1 and the other contains T2 and if T1 and T2 are unifiable, then neither T1 nor T2 should appear in the resolvent. We call T1 and T2 complementary literals. Use the substitution produced by the unification to create the resolvent. If there is one pair of complementary literals, only one such pair should be omitted from the resolvent.

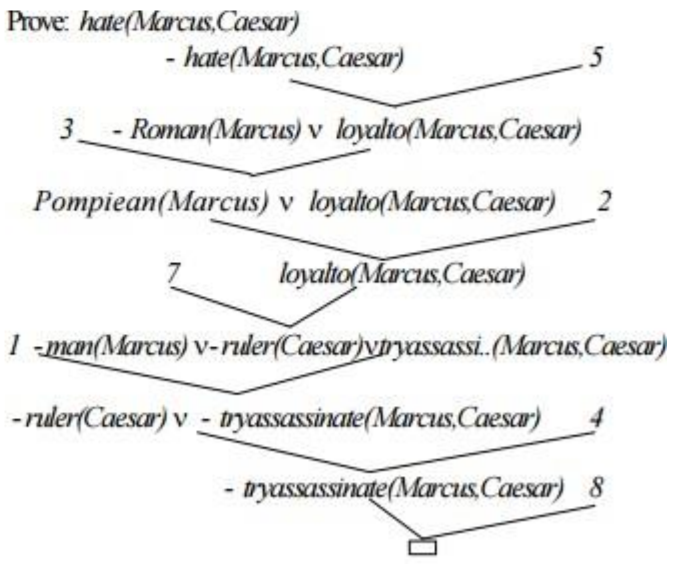
c) If the resolvent is the empty clause, then a contradiction has been found. If it is not then add it to the set of clauses available to the procedure.

If the choice of clauses to resolve together at each step is made in certain systematic ways, then the resolution procedure will find a contradiction if one exists. However, it may take a very long time. There exist strategies for making the choice that can speed up the process considerably as given below.

Axioms in clause form

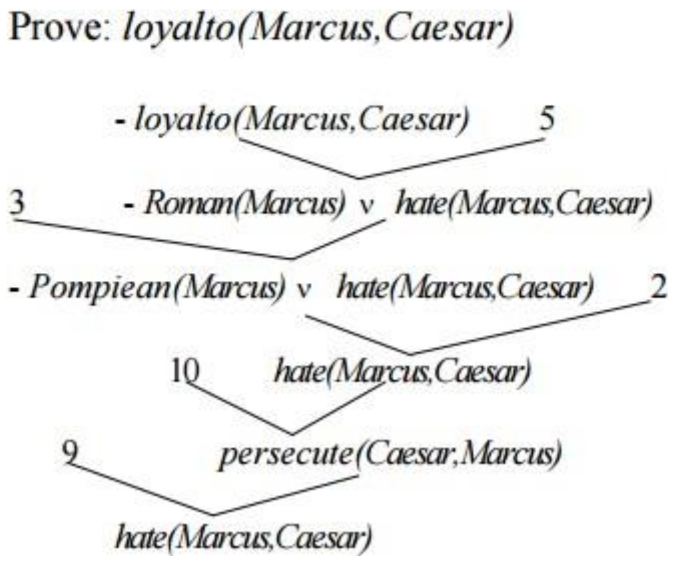


1.*man(Marcus)*
2.*Pompiean(Marcus)*
3.- *Pompiean(x1)* v *Roman(x1)*
4.*ruler(Caesar)*
5.- *Roman(x2)* v *loyalto(x2,Caesar)* v *hate(x2,Caesar)*
6. *loyal(x3,f(x3))*
7.-*man(x4)* v -*ruler(y1)* v - *tryassassinate(x4,y1)* v
loyalto(x4,y1)
8.*tryassassinate(Marcus,Caesar)*



Prove: hate(Marcus,Caesar)
- hate(Marcus,Caesar)
5
3
- Roman(Marcus) v loyalto(Marcus,Caesar)
Pompiean(Marcus) v loyalto(Marcus,Caesar)
2
7
loyalto(Marcus,Caesar)
1
-man(Marcus) v -ruler(Caesar) v tryassassi..(Marcus,Caesar)
-ruler(Caesar) v - tryassassinate(Marcus,Caesar)
4
- tryassassinate(Marcus,Caesar)
8

A Resolution Proof



Prove: loyalto(Marcus,Caesar)
- loyalto(Marcus,Caesar)
5
3
- Roman(Marcus) v hate(Marcus,Caesar)
- Pompiean(Marcus) v hate(Marcus,Caesar)
2
10
hate(Marcus,Caesar)
9
persecute(Caesar,Marcus)
hate(Marcus,Caesar)

An Unsuccessful Attempt at Resolution

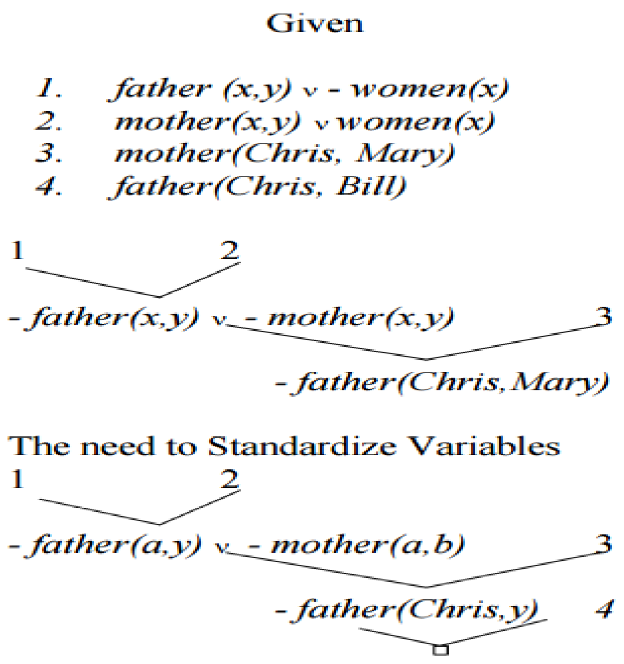
9. persecute(x, y) $\rightarrow$ hate(y, x)

10. hate(x,y) $\rightarrow$ persecute(y, x)

Converting to clause form, we get

9. $\neg$ persecute(x_5,y_2) $\vee$ hate(y_2,x_5)

10. $\neg$hate(x_6,y_3) $\vee$ persecute(y_3,x_6)


Given
1. father (x,y) v - women(x)
2. mother(x,y) v women(x)
3. mother(Chris, Mary)
4. father(Chris, Bill)
1
2
- father(x,y) v - mother(x,y)
3
- father(Chris,Mary)
The need to Standardize Variables
1
2
- father(a,y) v - mother(a,b)
3
- father(Chris,y)
4

2.5.5 Procedural v/s Declarative Knowledge

- A Declarative representation is one in which knowledge is specified but the use to which that knowledge is to be put in, is not given.
- A Procedural representation is one in which the control information that is necessary to use the knowledge is considered to be embedded in the knowledge itself.
- To use a procedural representation, we need to augment it with aninterpreter that follows the instructions given in the knowledge.
- The difference between the declarative and the procedural views of knowledge lies in where control information resides.

  man(Marcus)

  man(Caesar)

  person(Cleopatra)

  ∀x: man(x)→ person(x)

  person(x)?

  Now we want to extract from this knowledge base the ans to the question:

Ǝy : person (y)

Marcus, Ceaser and Cleopatra can be the answers

- As there is more than one value that satisfies the predicate, but only one value is needed, the answer depends on the order in which the assertions are examined during the search of a response.
- If we view the assertions as declarative, then we cannot depict how they will be examined. If we view them as procedural, then they do.
- Let us view these assertions as a non deterministic program whoseoutput is simply not defined, now this means that there is no difference between Procedural & Declarative Statements. But most of the machines don‟t do so, they hold on to what ever method they have, either sequential or in parallel.
- The focus is on working on the control model.

man(Marcus)

man (Ceaser)

Vx : man(x)

person(x)

Person(Cleopatra)

- If we view this as declarative then there is no difference with the previous statement. But viewed procedurally, and using the control model, we used to got Cleopatra as the answer, now the answer is marcus.
- The answer can vary by changing the way the interpreter works.
- The distinction between the two forms is often very fuzzy. Rather then trying to prove which technique is better, what we should do is to figure out what the ways in which rule formalisms and interpreters can be combined to solve problems.

2.5.6 Logic Programming

- Logic programming is a programming language paradigm in whichlogical assertions are viewed as programs, e.g : PROLOG
- A PROLOG program is described as a series of logical assertions, eachof which is a Horn Clause.
- A Horn Clause is a clause that has at most one positive literal.

- Eg p, ¬ p V q etc are also Horn Clauses.
- The fact that PROLOG programs are composed only of Horn Clauses and not of arbitrary logical expressions has two important consequences.
- Because of uniform representation a simple & effective interpreter can be written.
- The logic of Horn Clause systems is decidable.
- Even PROLOG works on backward reasoning.
- The program is read top to bottom, left to right and search is performed depth-first with backtracking.
- There are some syntactic difference between the logic and the PROLOG representations as mentioned
- The key difference between the logic & PROLOG representation is that PROLOG interpreter has a fixed control strategy, so assertions in the PROLOG program define a particular search path to answer any question. Whereas Logical assertions define set of answers that they justify, there can be more than one answers, it can be forward or backward tracking.
- Control Strategy for PROLOG states that we begin with a problem statement, which is viewed as a goal to be proved.

- Look for the assertions that can prove the goal.
- To decide whether a fact or a rule can be applied to the current problem, invoke a standard unification procedure.
- Reason backward from that goal until a path is found that terminates with assertions in the program.
- Consider paths using a depth-first search strategy and use backtracking.
- Propagate to the answer by satisfying the conditions.

2.5.7 Forward v/s Backward Reasoning

- The objective of any search is to find a path through a problem spacefrom the initial to the final one.
- There are 2 directions to go and find the answer

    - ✓ Forward
    - ✓ Backward
- ➢ 8-square problem
- ➢ Reason forward from the initial states: Begin building a tree of move sequences that might be solution by starting with the initialconfiguration(s) at the root of the tree. Generate the next level of tree by finding all the rules whose left sides match the root node and use the right sides to create the new configurations. Generate each node by taking each node generated at the previous level and applying to it all of the rules whose left sides match it. Continue.
- ➢ Reason backward from the goal states: Begin building a tree of move sequences that might be solution by starting with the goal configuration(s) at the root of the tree. Generate the next level of tree by finding all the rules whose right sides match the root node and use the left sides to create the new configurations. Generate each node by taking each node generated at the previous level and applying to it all of the rules whose right sides match it. Continue. This is also called Goal-Directed Reasoning.
- ➢ To summarize, to reason forward, the left sides (pre-conditions) are matched against the current state and the right sides (the results) are used to generate new nodes until the goal is reached.
- ➢ To reason backwards, the right sides are matched against the current node and the left sides are used to generate new nodes.
- ➢ Factors that influence whether to choose forward or backward reasoning:
    - ✓ Are there more possible start states or goal states? We would like to go from smaller set of states to larger set of states.
    - ✓ In which direction is the branching factor (the average number of nodes that can be reached directly from a single node) greater? We would like to proceed in the direction with the lower branching factor.
    - ✓ Will the program be asked to justify its reasoning process to the user? It so, it is important to proceed in the direction that corresponds more closely with the way user will think.
    - ✓ What kind of event is going to trigger a problem-solving episode? If it is the arrival of a new fact , forward reasoning should be used. If it a query to which response is desired, use backward reasoning.
- ➢ Home to unknown place example.
- ➢ MYCIN

- Bidirectional Search (The two searches must pass each other)
- Forward Rules: which encode knowledge about how to respond to certain input configurations.
- Backward Rules: which encode knowledge about how to achieve particular goals.
- Backward- Chaining Rule Systems
  - ✓ PROLOG is an example of this.
  - ✓ These are good for goal-directed problem solving.
  - ✓ Hence Prolog & MYCIN are examples of the same.
- Forward - Chaining Rule Systems
  - ✓ We work on the incoming data here.
  - ✓ The left sides of rules are matched with against the state description.
  - ✓ The rules that match the state dump their right side assertions into the state.
  - ✓ Matching is more complex for forward chaining systems.
  - ✓ OPS5, Brownston etc. are the examples of the same.
- Combining Forward v/s Backward Reasoning
  - ✓ Patients example of diagnosis.
  - ✓ In some systems, this is only possible in reversible rules.

Matching

- Till now we have used search to solve the problems as the application of appropriate rules.
- We applied them to individual problem states to generate new states to which the rules can then be applied, until a solution is found.
- We suggest that a clever search involves choosing from among the rules that can be applied at a particular point, but we do not talk about how to extract from the entire collection of rules those that can be applied at a given point.
- To do this we need matching.

Indexing

- Do a simple search through all the rules, comparing each one‟s precondition to the current state and extracting all the ones that match.
- But this has two problems
- In order to solve very interesting problems, it will be necessary to use a large number of rules, scanning through all of them at every step of the search would be hopelessly inefficient.
- It is not always immediately obvious whether a rule‟s preconditions are satisfied by a particular state.
- To solve the first problem, use simple indexing. E.g. in Chess, combine all moves at a particular board state together.

Matching with Variables

- The problem of selecting applicable rules is made more difficult when preconditions are not stated as exact descriptions of particular situations but rather describe properties that the situations must have.
- Then we need to match a particular situation and the preconditions of a given situation.
- In many rules based systems, we need to compute the whole set of rules that match the current state description. Backward Chaining Systems usually use depth-first backtracking to select individual rules, but forward chaining systems use Conflict Resolution Strategies.
- One efficient many to many match algorithm is RETE

Complex &Approximate Matching

- A more complex matching process is required when the preconditions of a rule specify required properties that are not stated explicitly in thedescription of the current state. In this case, a separate set of rules must be used to describe how some properties can be inferred from others.
- An even more complex matching process is required if rules should be applied if their preconditions approximately match the current situation. Example of listening to a recording of a telephonic conversation.
- For some problems, almost all the action is in the matching of the rulesto the problem state. Once that is done, so few rules apply that theremaining search is trivial. Example ELIZA

Conflict Resolution

- The result of the matching process is a list of rules whose antecedents have matched the current state description along with whatever variable binding were generated by the matching process.
- It is the job of the search method to decide on the order in which the rules will be applied. But sometimes it is useful to incorporate some of the decision making into the matching process. This phase is called conflict resolution.
- There are three basic approaches to the problem of conflict resolution in the production system
  - ✓ Assign a preference based on the rule that matched.
  - ✓ Assign a preference based on the objects that matched.
  - ✓ Assign a preference based on the action that the matched rule would perform.

STRUCTURED REPRESNTATION OF KNOWLEDGE

- Representing knowledge using logical formalism, like predicate logic, has several advantages. They can be combined with powerful inference mechanisms like resolution, which makes reasoning with facts easy. But using logical formalism complex structures of the world, objects and their relationships, events, sequences of events etc. cannot be described easily.
- A good system for the representation of structured knowledge in a particular domain should posses the following four properties:

  (i) Representational Adequacy:- The ability to represent all kinds of knowledge that are needed in that domain.

  (ii) Inferential Adequacy :- The ability to manipulate the represented structure and infer new structures.

  (iii) Inferential Efficiency:- The ability to incorporate additional information into the knowledge structure that will aid the inference mechanisms.

  (iv) Acquisitional Efficiency :- The ability to acquire new information easily, either by direct insertion or by program control.
- The techniques that have been developed in AI systems to accomplish these objectives fall under two categories:

1. Declarative Methods:- In these knowledge is represented as static collection of facts which are manipulated by general procedures. Here the facts need to be stored only one and they can be used in any number of ways. Facts can be easily added to declarative systems without changing the general procedures.

2. Procedural Method:- In these knowledge is represented as procedures. Default reasoning and probabilistic reasoning are examples of procedural methods. In these, heuristic knowledge of "How to do things efficiently "can be easily represented.

- In practice most of the knowledge representation employ a combination of both. Most of the knowledge representation structures have been developed to handle programs that handle natural language input. One of the reasons that knowledge structures are so important is that they provide a way to represent information about commonly occurring patterns of things . such descriptions are some times called schema. One definition of schema is
- "Schema refers to an active organization of the past reactions, or of past experience, which must always be supposed to be operating in any well adapted organic response".
- By using schemas, people as well as programs can exploit the fact that the real world is not random. There are several types of schemas that have proved useful in AI programs. They include

  (i) Frames:- Used to describe a collection of attributes that a given object possesses (eg: description of a chair).

  (ii) Scripts:- Used to describe common sequence of event (eg:- a restaurant scene).

  (iii) Stereotypes :- Used to described characteristics of people.

  (iv) Rule models:- Used to describe common features shared among a set of rules in a production system.
- Frames and scripts are used very extensively in a variety of AI programs. Before selecting any specific knowledge representation structure, the following issues have to be considered.

  (i) The basis properties of objects , if any, which are common to every problem domain must be identified and handled appropriately.

  (ii) The entire knowledge should be represented as a good set of primitives.

  (iii) Mechanisms must be devised to access relevant parts in a large knowledge base.

PART – A

1. How is predicate logic helpful in knowledge representation?

2. Define semantic networks.

3. What is the need of facts and its representation?

4. What is property inheritance?

5. Discuss in brief about ISA and Instance classes.

6. Give some use of conceptual dependency.

7. Define inference.

8. Define logic.

9. Write short notes on uniqueness quantifier.

10. Write short notes on uniqueness operator.

11. Define WWF with an example.

12. Define FOL with an example.

13. Difference between propositional and FOL logic.

14. Define forward chaining and backward chaining.

15. Define Horn clause.

16. Define Canonical horn clause.

17. Write notes on long term and short term memory.

18. Name any 3 frame languages.

19. Write in short about iterative deepening..

20. Is minimax depth fist search or Breadth first search.

PART – B

1. Issues in knowledge representation

2. State Representation of facts in predicate logic.

3. How will you represent facts in propositional logic with an example?

4. Explain Resolution in brief with an example.

5. Write algorithm for propositional resolution and Unification algorithm.

6. Explain in detail about forward and backward chaining with suitable example.

7. Explain steps involved in Matching.

8. Explain the different logics used for knowledge representation.

9. How will you represent facts in Proportional logic with an example.

10. Explain resolution in brief with an example.

11. Explain in detail about minimax procedure.

12. Explain the effect of Alpha beta cut off over minimax.

13. How would the minimax procedure have to be modified to be used by a program playing 3 or 4 persons instead of 2 persons.

UNIT-3

CHAPTER 1

3.1 KNOWLEDGE INFERENCE

The object of a knowledge representation is to express knowledge in a computer tractable form, so that it can be used to enable our AI agents to perform well.

A knowledge **representation language** is defined by two aspects:

1. Syntax The syntax of a language defines which configurations of the components of the language constitute valid sentences.
2. Semantics The semantics defines which facts in the world the sentences refer to, and hence the statement about the world that each sentence makes.

This is a very general idea, and not restricted to natural language.

Suppose the language is arithmetic, then „x‟, „³‟ and „y‟ are components (or symbols or words) of the language the syntax says that „x ³ y‟ is a valid sentence in the language, but „³ ³ x y‟ is not the semantics say that „x ³ y‟ is false if y is bigger than x, and true otherwise A good knowledge representation system for any particular domain should possess the following properties:

1. Representational Adequacy – the ability to represent all the different kinds of knowledge that might be needed in that domain.
2. Inferential Adequacy – the ability to manipulate the representational structures to derive new structures (corresponding to new knowledge) from existing structures.
3. Inferential Efficiency – the ability to incorporate additional information into the knowledge structure which can be used to focus the attention of the inference mechanisms in the most promising directions.
4. Acquisitional Efficiency – the ability to acquire new information easily. Ideally the agent should be able to control its own knowledge acquisition, but direct insertion of information by a „knowledge engineer‟ would be acceptable.

In practice, the theoretical requirements for good knowledge representations can usually be achieved by dealing appropriately with a number of practical requirements:

1. The representations need to be complete – so that everything that could possibly need to be represented, can easily be represented.

2. They must be computable – implementable with standard computing procedures.

3. They should make the important objects and relations explicit and accessible – so that it is easy to see what is going on, and how the various components interact.

4. They should suppress irrelevant detail – so that rarely used details don‟t introduce necessary complications, but are still available when needed.

5. They should expose any natural constraints – so that it is easy to express how one object or relation influences another.

6. They should be transparent – so you can easily understand what is being said.

7. The implementation needs to be concise and fast – so that information can be stored, retrieved and manipulated rapidly.

A Knowledge representation formalism consists of collections of condition-action rules (Production Rules or Operators), a database which is modified in accordance with the rules, and a Production System Interpreter which controls the operation of the rules i.e The 'control mechanism' of a Production System, determining the order in which Production Rules are fired. A system that uses this form of knowledge representation is called a production system.

3.2 Production Based System

A production system consists of four basic components:

1. A set of rules of the form Ci ® Ai where Ci is the condition part and Ai is the action part. The condition determines when a given rule is applied, and the action determines what happens when it is applied.

2. One or more knowledge databases that contain whatever information is relevant for the given problem. Some parts of the database may be permanent, while others may temporary and only exist during the solution of the current problem. The information in the databases may be structured in any appropriate manner.

3. A control strategy that determines the order in which the rules are applied to the database, and provides a way of resolving any conflicts that can arise when several rules match at once.

4. A rule applier which is the computational system that implements the control strategy and applies the rules.

Four classes of production systems:-

1. A monotonic production system

2. A non monotonic production system

3.A partially commutative production system

4. A commutative production system.

Advantages of production systems:-

1. Production systems provide an excellent tool for structuring AI programs.

2. Production Systems are highly modular because the individual rules can be added, removed or modified independently.

3. The production rules are expressed in a natural form, so the statements contained in the knowledge base should the a recording of an expert thinking out loud.

Disadvantages of Production Systems:-

One important disadvantage is the fact that it may be very difficult to analyse the flow of control within a production system because the individual rules don‟t call each other.

Production systems describe the operations that can be performed in a search for a solution to the problem. They can be classified as follows.
Monotonic production system :-

A system in which the application of a rule never prevents the later application of another rule, that could have also been applied at the time the first rule was selected.

Partially commutative production system:-

A production system in which the application of a particular sequence of rules transforms state X into state Y, then any permutation of those rules that is allowable also transforms state x into state Y.

Theorem proving falls under monotonic partially communicative system. Blocks world and 8 puzzle problems like chemical analysis and synthesis come under monotonic, not partially commutative systems. Playing the game of bridge comes under non monotonic , not partially commutative system.

For any problem, several production systems exist. Some will be efficient than others. Though it may seem that there is no relationship between kinds of problems and kinds of production systems, in practice there is a definite relationship.

Partially commutative , monotonic production systems are useful for solving ignorable problems. These systems are important for man implementation standpoint because they can be implemented without the ability to backtrack to previous states, when it is discovered that an incorrect path was followed. Such systems increase the efficiency since it is not necessary to keep track of the changes made in the search process.

Monotonic partially commutative systems are useful for problems in which changes occur but can be reversed and in which the order of operation is not critical (ex: 8 puzzle problem).

Production systems that are not partially commutative are useful for many problems in which irreversible changes occur, such as chemical analysis. When dealing with such systems, the order in which operations are performed is very important and hence correct decisions have to be made at the first time itself.

3.3 Frame Based System

A frame is a data structure with typical knowledge about a particular object or concept. Frames, first proposed by Marvin Minsky in the 1970s.

Example : Boarding pass frames

QANTAS BOARDING PASS		**AIR NEW ZEALAND BOARDING PASS**	
***Carrier*:**	*QANTAS AIRWAYS*	***Carrier*:**	*AIR NEW ZEALAND*
***Name*:**	*MR N BLACK*	***Name*:**	*MRS J WHITE*
***Flight*:**	*QF 612*	***Flight*:**	*NZ 0198*
***Date*:**	*29DEC*	***Date*:**	*23NOV*
***Seat*:**	*23A*	***Seat*:**	*27K*
***From*:**	*HOBART*	***From*:**	*MELBOURNE*

Each frame has its own name and a set of attributes associated with it. Name, weight, height and age are slots in the frame Person. Model, processor, memory and price are slots in the frame Computer. Each attribute or slot has a value attached to it.

Frames provide a natural way for the structured and concise representation of knowledge.

A frame provides a means of organising knowledge in slots to describe various attributes and characteristics of the object.

Frames are an application of object-oriented programming for expert systems.

Object-oriented programming is a programming method that uses objects as a basis for analysis, design and implementation.

In object-oriented programming, an object is defined as a concept, abstraction or thing with crisp boundaries and meaning for the problem at hand. All objects have identity and are clearly distinguishable. Michael Black, Audi 5000 Turbo, IBM Aptiva S35 are examples of objects.

An object combines both data structure and its behaviour in a single entity. This is in sharp contrast to conventional programming, in which data structure and the program behaviour have concealed or vague connections.

When an object is created in an object-oriented programming language, we first assign a name to the object, then determine a set of attributes to describe the object‟s characteristics, and at last write procedures to specify the object‟s behaviour.

A knowledge engineer refers to an object as a frame (the term, which has become the AI jargon).

Frames as a knowledge representation technique

The concept of a frame is defined by a collection of slots. Each slot describes a particular attribute or operation of the frame.

Slots are used to store values. A slot may contain a default value or a pointer to another frame, a set of rules or procedure by which the slot value is obtained.

Typical information included in a slot

Frame name.

Relationship of the frame to the other frames. The frame IBM Aptiva S35 might be a member of the class Computer, which in turn might belong to the class Hardware.

Slot value. A slot value can be symbolic, numeric or Boolean. For example, the slot Name has symbolic values, and the slot Age numeric values. Slot values can be assigned when the frame is created or during a session with the expert system.

Default slot value. The default value is taken to be true when no evidence to the contrary has been found. For example, a car frame might have four wheels and a chair frame four legs as default values in the corresponding slots.

Range of the slot value. The range of the slot value determines whether a particular object complies with the stereotype requirements defined by the frame. For example, the cost of a computer might be specified between $750 and $1500.

Procedural information. A slot can have a procedure attached to it, which is executed if the slot value is changed or needed.

Most frame based expert systems use two types of methods:

WHEN CHANGED and WHEN NEEDED

A WHEN CHANGED method is executed immediately when the value of its attribute changes.

A WHEN NEEDED method is used to obtain the attribute value only when it is needed.

A WHEN NEEDED method is executed when information associated with a particular attribute is needed for solving the problem, but the attribute value is undetermined.

Most frame based expert systems allow us to use a set of rules to evaluate information contained in frames.

How does an inference engine work in a frame based system?

In a rule based system, the inference engine links the rules contained in the knowledge base with data given in the database.

When the goal is set up, the inference engine searches the knowledge base to find a rule that has th goal in its consequent.

If such a rule is found and its IF part matches data in the database, the rule is fired and the specified object, the goal, obtains its value. If no rules are found that can derive a value for the goal, the system queries the user to supply that value.

In a frame based system, the inference engine also searches for the goal. But

In a frame based system, rules play an auxiliary role. Frames represent here a major source of knowledge and both methods and demons are used to add actions to the frames.

Thus the goal in a frame based system can be established either in a method or in a demon.

Difference between methods and demons:

A demon has an IF-THEN structure. It is executed whenever an attribute in the demon‟s IF statement changes its value. In this sense, demons and methods are very similar and the two terms are often used as synonyms.

However, methods are more appropriate if we need to write complex procedures. Demons on the other hand, are usually limited to IF-THEN statements.

3.3 Inference

Two control strategies: forward chaining and backward chaining

Forward chaining:

Working from the facts to a conclusion. Sometimes called the datadriven approach. To chain forward, match data in working memory against 'conditions' of rules in the rule-base. When one of them fires, this is liable to produce more data. So the cycle continues

Backward chaining:

Working from the conclusion to the facts. Sometimes called the goal-driven approach.

To chain backward, match a goal in working memory against 'conclusions' of rules in the rule-base.

When one of them fires, this is liable to produce more goals. So the cycle continues.

The choice of strategy depends on the nature of the problem. Assume the problem is to get from facts to a goal (e.g. symptoms to a diagnosis).

Backward chaining is the best choice if:

The goal is given in the problem statement, or can sensibly be guessed at the beginning of the consultation; or:

The system has been built so that it sometimes asks for pieces of data (e.g. "please now do the gram test on the patient's blood, and tell me the result"), rather than expecting all the facts to be presented to it.

This is because (especially in the medical domain) the test may be expensive, or unpleasant, or dangerous for the human participant so one would want to avoid doing such a test unless there was a good reason for it.

Forward chaining is the best choice if:

All the facts are provided with the problem statement; or:

There are many possible goals, and a smaller number of patterns of data; or:

There isn't any sensible way to guess what the goal is at the beginning of the consultation.

Note also that a backwards-chaining system tends to produce a sequence of questions which seems focussed and logical to the user, a forward-chaining system tends to produce a sequence which seems random & unconnected.

If it is important that the system should seem to behave like a human expert, backward chaining is probably the best choice.

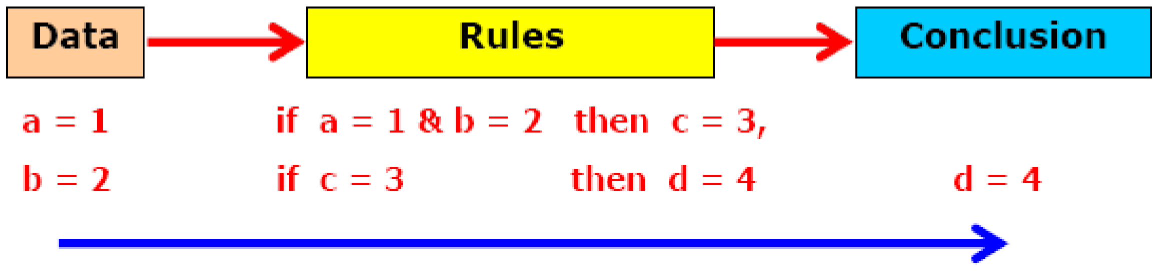
3.3.1 Forward Chaining Algorithm

Forward chaining is a techniques for drawing inferences from Rule base. Forward-chaining inference is often called data driven.

‡ The algorithm proceeds from a given situation to a desired goal,adding new assertions (facts) found.

‡ A forward-chaining, system compares data in the working memory against the conditions in the IF parts of the rules and determines which rule to fire.

‡ Data Driven



Example : Forward Channing

■ Given : A Rule base contains following Rule set

Rule 1: If A and C Then F

Rule 2: If A and E Then G

Rule 3: If B Then E

Rule 4: If G Then D

■ Problem : Prove

If A and B true Then D is true

Solution :

(i) Start with input given A, B is true and then

start at Rule 1 and go forward/down till a rule

“fires" is found.

First iteration :

(ii) Rule 3 fires : conclusion E is true

new knowledge found

(iii) No other rule fires;

end of first iteration.

(iv) Goal not found;

new knowledge found at (ii);

go for second iteration

Second iteration :

(v) Rule 2 fires : conclusion G is true

new knowledge found

(vi) Rule 4 fires : conclusion D is true

Goal found;

Proved

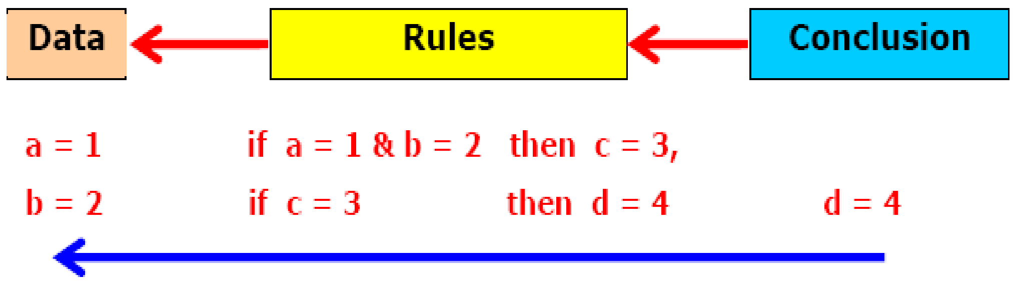
3.3.2 Backward Chaining Algorithm

Backward chaining is a techniques for drawing inferences from Rule base. Backward-chaining inference is often called goal driven.

‡ The algorithm proceeds from desired goal, adding new assertions found.

‡ A backward-chaining, system looks for the action in the THEN clause of the rules that matches the specified goal.

Goal Driven



Example : Backward Channing

■ Given : Rule base contains following Rule set

Rule 1: If A and C Then F

Rule 2: If A and E Then G

Rule 3: If B Then E

Rule 4: If G Then D

■ Problem : Prove

If A and B true Then D is true

Solution :

(i) Start with goal ie D is true

go backward/up till a rule "fires" is found.

First iteration :

(ii) Rule 4 fires :

new sub goal to prove G is true

go backward

(iii) Rule 2 "fires"; conclusion: A is true

new sub goal to prove E is true

go backward;

(iv) no other rule fires; end of first iteration.

new sub goal found at

(iii)go for second iteration

Second iteration :

(v) Rule 3 fires :

conclusion B is true (2nd input found)

both inputs A and B ascertained

Proved

CHAPTER-2

3.4 Fuzzy Logic

Fuzzy Logic (FL) is a method of reasoning that resembles human reasoning. The approach of FL imitates the way of decision making in humans that involves all intermediate possibilities between digital values YES and NO.

The conventional logic block that a computer can understand takes precise input and produces a definite output as TRUE or FALSE, which is equivalent to human‟s YES or NO.

The inventor of fuzzy logic, Lotfi Zadeh, observed that unlike computers, the human decision making includes a range of possibilities between YES and NO, such as −

CERTAINLY YES
POSSIBLY YES
CANNOT SAY
POSSIBLY NO
CERTAINLY NO

The fuzzy logic works on the levels of possibilities of input to achieve the definite output.

Implementation

- It can be implemented in systems with various sizes and capabilities ranging from small micro-controllers to large, networked, workstation-based control systems.
- It can be implemented in hardware, software, or a combination of both.

Why Fuzzy Logic?

Fuzzy logic is useful for commercial and practical purposes.

- It can control machines and consumer products.
- It may not give accurate reasoning, but acceptable reasoning.
- Fuzzy logic helps to deal with the uncertainty in engineering.

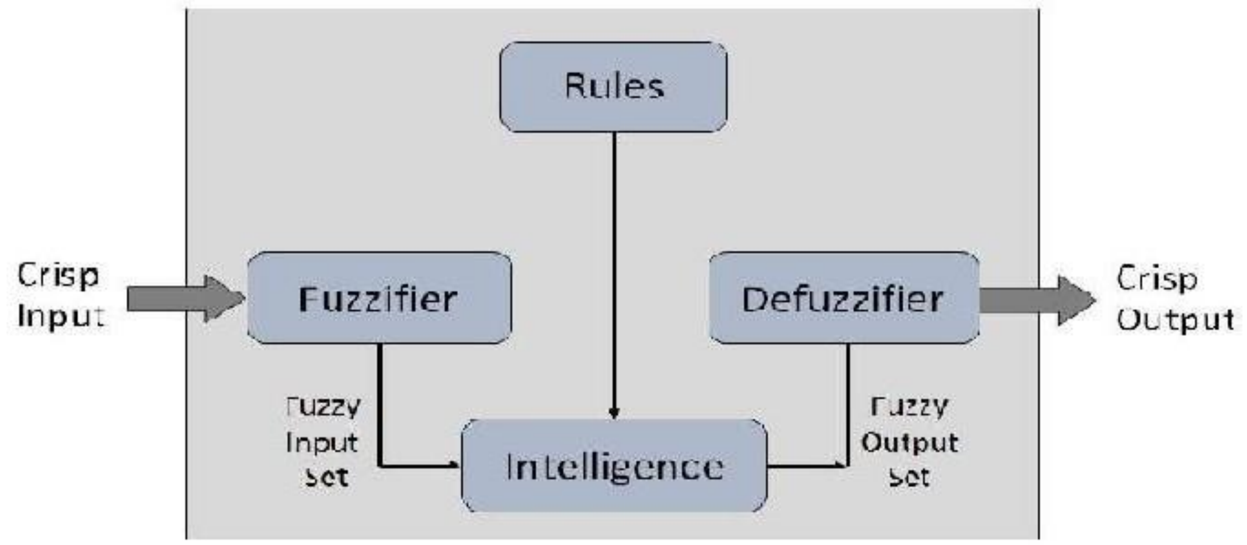
Fuzzy Logic Systems Architecture

It has four main parts as shown −

- Fuzzification Module − It transforms the system inputs, which are crisp numbers, into fuzzy sets. It splits the input signal into five steps such as −

LP	x is Large Positive
MP	x is Medium Positive
S	x is Small
MN	x is Medium Negative
LN	x is Large Negative

- Knowledge Base − It stores IF-THEN rules provided by experts.
- Inference Engine − It simulates the human reasoning process by making fuzzy inference on the inputs and IF-THEN rules.
- Defuzzification Module − It transforms the fuzzy set obtained by the inference engine into a crisp value.



The membership functions work on fuzzy sets of variables.

Membership Function

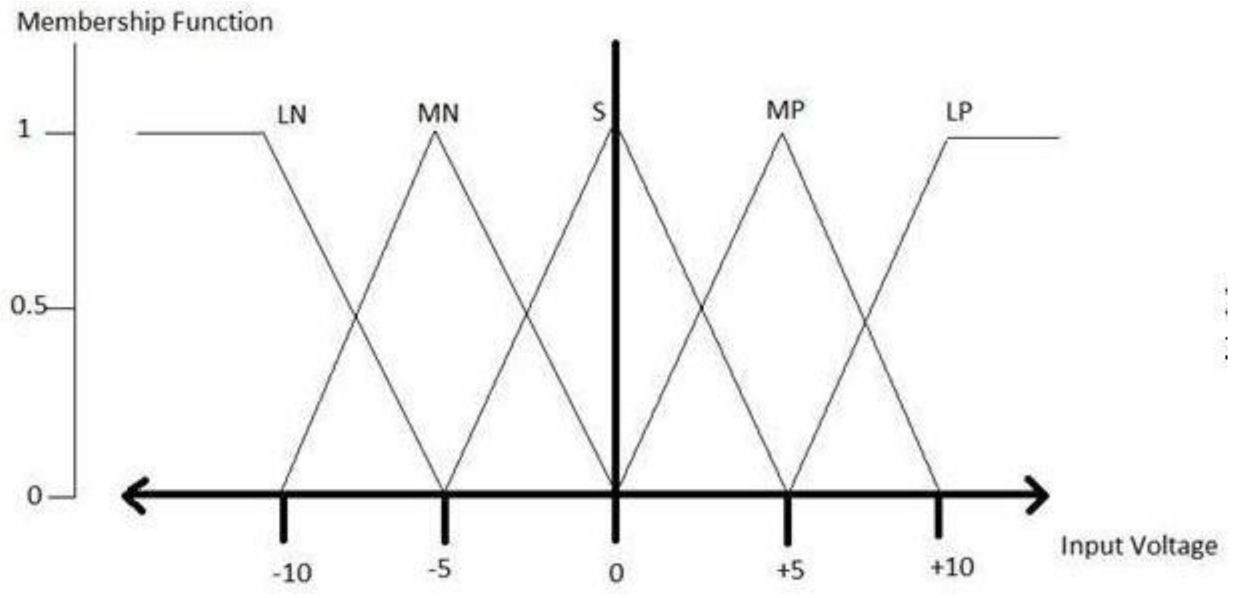
Membership functions allow you to quantify linguistic term and represent a fuzzy set graphically. A membership function for a fuzzy set A on the universe of discourse X is defined as µA:X → [0,1].

Here, each element of X is mapped to a value between 0 and 1. It is called membership value or degree of membership. It quantifies the degree of membership of the element in X to the fuzzy set A.

- x axis represents the universe of discourse.
- y axis represents the degrees of membership in the [0, 1] interval.

There can be multiple membership functions applicable to fuzzify a numerical value. Simple membership functions are used as use of complex functions does not add more precision in the output.

All membership functions for LP, MP, S, MN, and LN are shown as below −

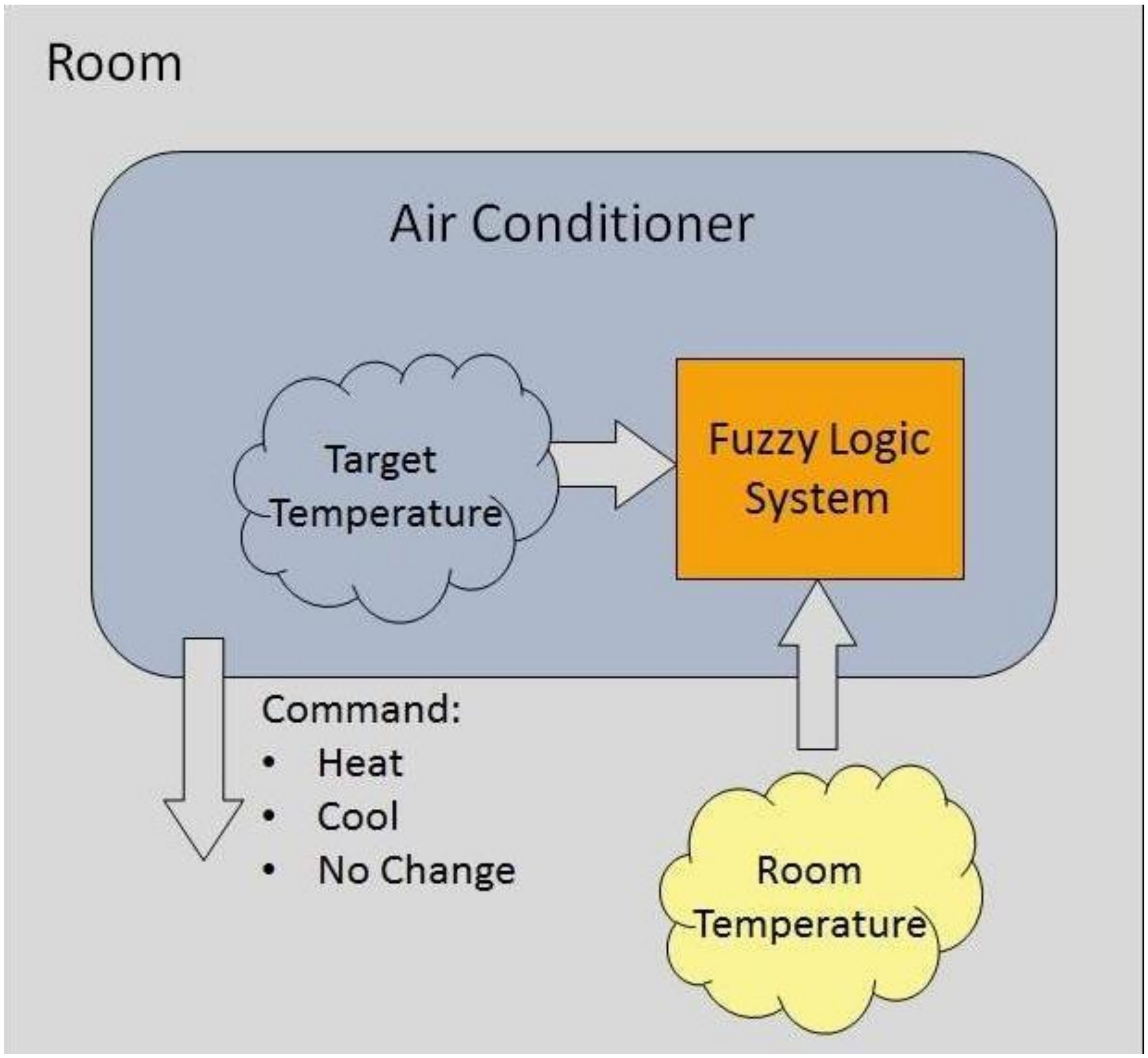


The triangular membership function shapes are most common among various other membership function shapes such as trapezoidal, singleton, and Gaussian.

Here, the input to 5-level fuzzifier varies from -10 volts to +10 volts. Hence the corresponding output also changes.

Example of a Fuzzy Logic System

Let us consider an air conditioning system with 5-lvel fuzzy logic system. This system adjusts the temperature of air conditioner by comparing the room temperature and the target temperature value.



Algorithm

- Define linguistic variables and terms.
- Construct membership functions for them.
- Construct knowledge base of rules.
- Convert crisp data into fuzzy data sets using membership functions. (fuzzification)
- Evaluate rules in the rule base. (interface engine)
- Combine results from each rule. (interface engine)
- Convert output data into non-fuzzy values. (defuzzification)

Logic Development

Step 1: Define linguistic variables and terms

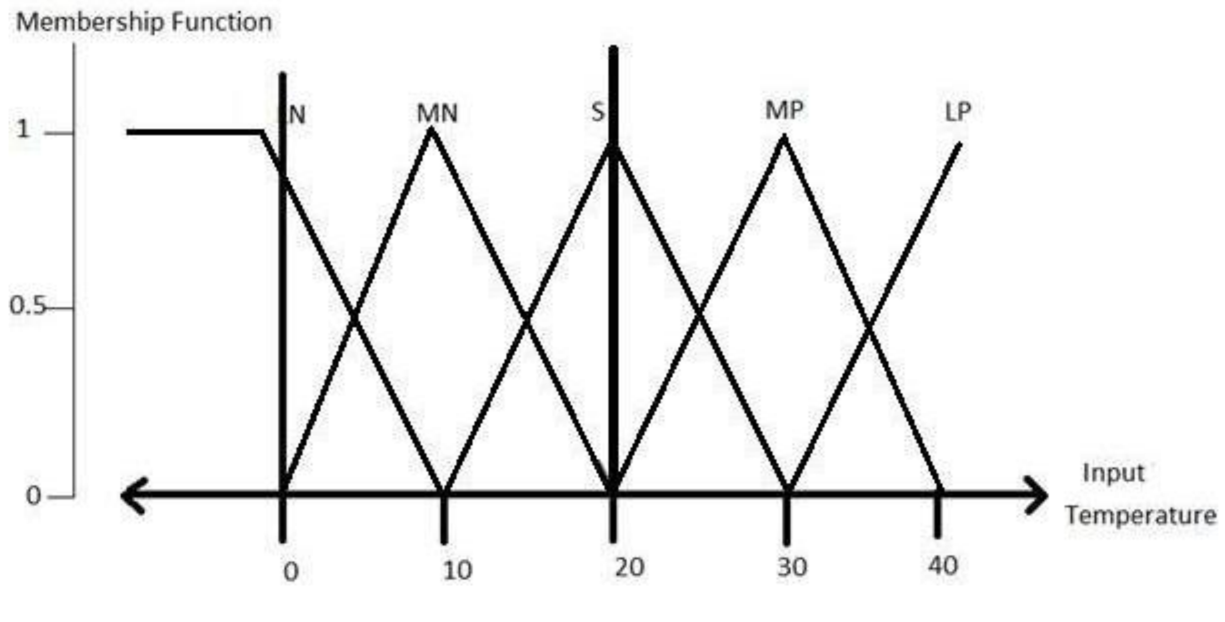
Linguistic variables are input and output variables in the form of simple words or sentences. For room temperature, cold, warm, hot, etc., are linguistic terms.

Temperature (t) = {very-cold, cold, warm, very-warm, hot}

Every member of this set is a linguistic term and it can cover some portion of overall temperature values.

Step 2: Construct membership functions for them

The membership functions of temperature variable are as shown −



Step3: Construct knowledge base rules

Create a matrix of room temperature values versus target temperature values that an air conditioning system is expected to provide.

RoomTemp. /Target	Very_Cold	Cold	Warm	Hot	Very_Hot
Very_Cold	No_Change	Heat	Heat	Heat	Heat
Cold	Cool	No_Change	Heat	Heat	Heat
Warm	Cool	Cool	No_Change	Heat	Heat
Hot	Cool	Cool	Cool	No_Change	Heat
Very_Hot	Cool	Cool	Cool	Cool	No_Change

Build a set of rules into the knowledge base in the form of IF-THEN-ELSE structures.

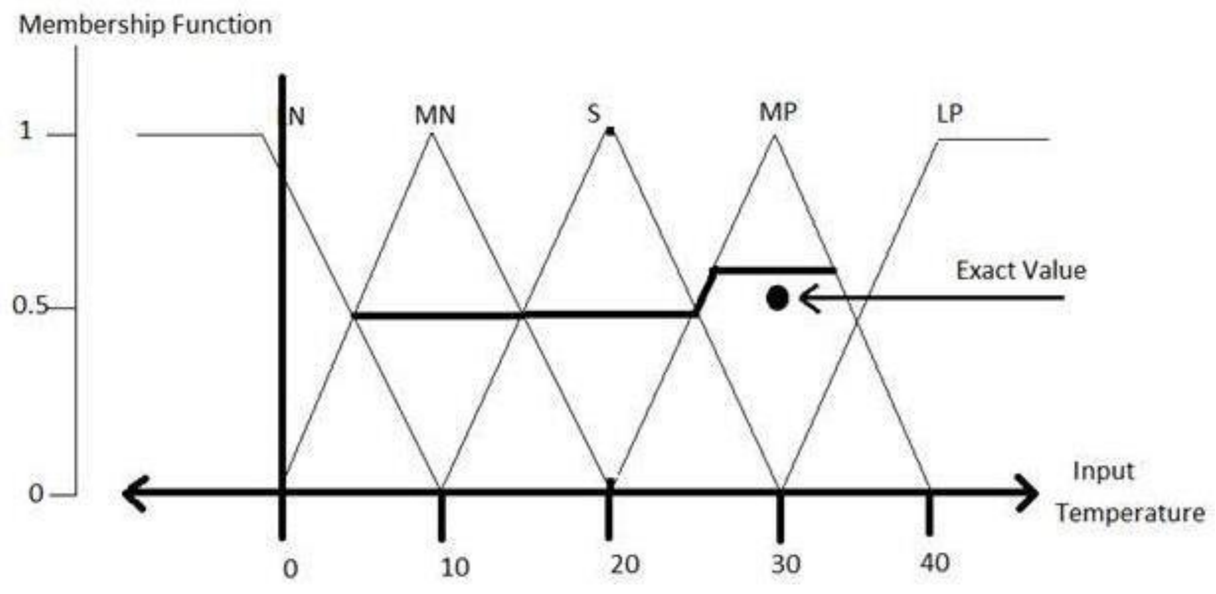
Sr. No.	Condition	Action
1	IF temperature=(Cold OR Very_Cold) AND target=Warm THEN	Heat
2	IF temperature=(Hot OR Very_Hot) AND target=Warm THEN	Cool
3	IF (temperature=Warm) AND (target=Warm) THEN	No_Change

Step 4: Obtain fuzzy value

Fuzzy set operations perform evaluation of rules. The operations used for OR and AND are Max and Min respectively. Combine all results of evaluation to form a final result. This result is a fuzzy value.

Step 5: Perform defuzzification

Defuzzification is then performed according to membership function for output variable.



Application Areas of Fuzzy Logic

The key application areas of fuzzy logic are as given −

Automotive Systems

- Automatic Gearboxes
- Four-Wheel Steering
- Vehicle environment control

Consumer Electronic Goods

- Hi-Fi Systems
- Photocopiers
- Still and Video Cameras
- Television

Domestic Goods

- Microwave Ovens
- Refrigerators
- Toasters
- Vacuum Cleaners
- Washing Machines

Environment Control

- Air Conditioners/Dryers/Heaters
- Humidifiers

Advantages of FLSs

- Mathematical concepts within fuzzy reasoning are very simple.
- You can modify a FLS by just adding or deleting rules due to flexibility of fuzzy logic.
- Fuzzy logic Systems can take imprecise, distorted, noisy input information.
- FLSs are easy to construct and understand.
- Fuzzy logic is a solution to complex problems in all fields of life, including medicine, as it resembles human reasoning and decision making.

Disadvantages of FLSs

- There is no systematic approach to fuzzy system designing.
- They are understandable only when simple.
- They are suitable for the problems which do not need high accuracy.

3.5 Certainty Factor

A certainty factor (CF) is a numerical value that expresses a degree of subjective belief that a particular item is true. The item may be a fact or a rule. When probabilities are used attention must be paid to the underlying assumptions and probability distributions in order to show validity. Bayes‟ rule can be used to combine probability measures.

Suppose that a certainty is defined to be a real number between -1.0 and +1.0, where 1.0 represents complete certainty that an item is true and -1.0 represents complete certainty that an item is false. Here a CF of 0.0 indicates that no information is available about either the truth or the falsity of an item. Hence positive values indicate a degree of belief or evidence that an item is true, and negative values indicate the opposite belief. Moreover it is common to select a positive number that represents a minimum threshold of belief in the truth of an item. For example, 0.2 is a commonly chosen threshold value.

Form of certainty factors in ES

IF <evidence>
THEN <hypothesis> {cf }

cf represents belief in hypothesis H given that evidence E has occurred

It is based on 2 functions
i) Measure of belief MB(H, E)
ii) Measure of disbelief MD(H, E)

Indicate the degree to which belief/disbelief of hypothesis H is increased if evidence E were observed

Total strength of belief and disbelief in a hypothesis:

$$cf = \frac{MB(H, E) - MD(H, E)}{1 - \min[MB(H, E), MD(H, E)]}$$

3.6 Bayesian networks

- Represent dependencies among random variables
- Give a short specification of conditional probability distribution
- Many random variables are conditionally independent
- Simplifies computations
- Graphical representation
- DAG – causal relationships among random variables
- Allows inferences based on the network structure

Definition of Bayesian networks

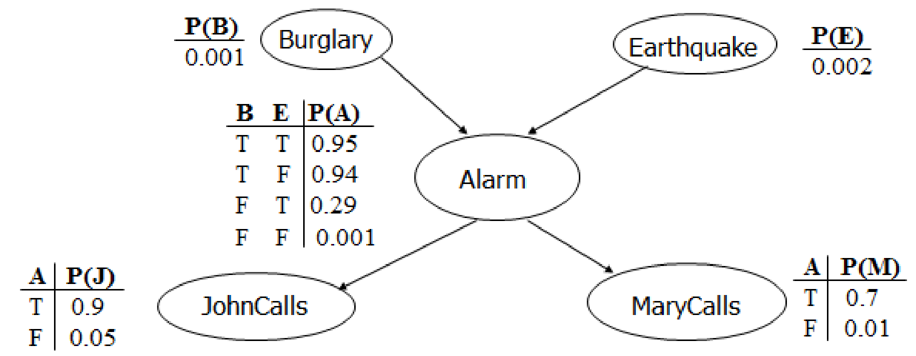
A BN is a DAG in which each node is annotated with quantitative probability information, namely:

- Nodes represent random variables (discrete or continuous)
- Directed links X→Y: X has a direct influence on Y, X is said to be a parent of Y
- each node X has an associated conditional probability table, **$P(X_i \mid Parents(X_i))$** that quantify the effects of the parents on the node

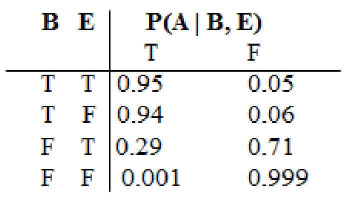
Example: Weather, Cavity, Toothache, Catch

- Weather, Cavity $\rightarrow$ Toothache, Cavity $\rightarrow$ Catch

Example



Conditional probability table



B	E	P(A | B, E) T	P(A | B, E) F
T	T	0.95	0.05
T	F	0.94	0.06
F	T	0.29	0.71
F	F	0.001	0.999

Bayesian network semantics

A) Represent a probability distribution

B) Specify conditional independence – build the network

A) each value of the probability distribution can be computed as:

$$\mathbf{P(X_1=x_1 \wedge \ldots X_n=x_n) = P(x_1,\ldots, x_n) = \Pi_{i=1,n} P(x_i \mid Parents(x_i))}$$

where Parents(xi) represent the specific values of Parents(Xi)

Building the network

$\mathbf{P(X_1=x_1 \wedge \ldots X_n=x_n) = P(x_1,\ldots, x_n) =}$

$\mathbf{P(x_n \mid x_{n-1},\ldots, x_1) * P(x_{n-1},\ldots, x_1) = \ldots =}$

$\mathbf{P(x_n \mid x_{n-1},\ldots, x_1) * P(x_{n-1} \mid x_{n-2},\ldots, x_1) * \ldots P(x_2 \mid x_1) * P(x_1) = \Pi_{i=1,n} P(x_i \mid x_{i-1},\ldots, x_1)}$

- We can see that **$P(X_i \mid X_{i-1},\ldots, X_1) = P(x_i \mid \text{Parents}(X_i))$** if **$\text{Parents}(X_i) \subseteq \{ X_{i-1},\ldots, X_1\}$**
- The condition may be satisfied by labeling the nodes in an order consistent with a DAG
- Intuitively, the parents of a node X_i must be all the nodes $X_{i-1},\ldots, X_1$ which have a direct influence on X_i.
- Pick a set of random variables that describe the problem
- Pick an ordering of those variables
- **while** there are still variables **repeat**

  (a) choose a variable X_i and add a node associated to X_i

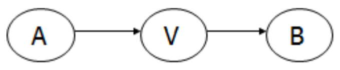
  (b) assign $\text{Parents}(X_i) \leftarrow$ a minimal set of nodes that already exists in the network such that the conditional independence property is satisfied

  (c) define the conditional probability table for X_i
- Because each node is linked only to previous nodes $\rightarrow$ DAG
- P(MaryCalls | JohnCals, Alarm, Burglary, Earthquake) = P(MaryCalls | Alarm)

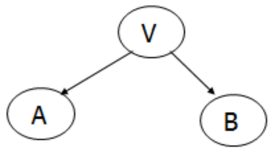
Compactness of node ordering

- Far more compact than a probability distribution
- Example of **locally structured system** (or ***sparse***): each component interacts directly only with a limited number of other components
- Associated usually with a linear growth in complexity rather than with an exponential one
- *The order of adding the nodes is important*
- The correct order in which to add nodes is to <u>add the “root causes” first</u>, then the variables they influence, and so on, until we reach the leaves

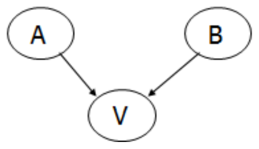
Probabilistic Interfaces



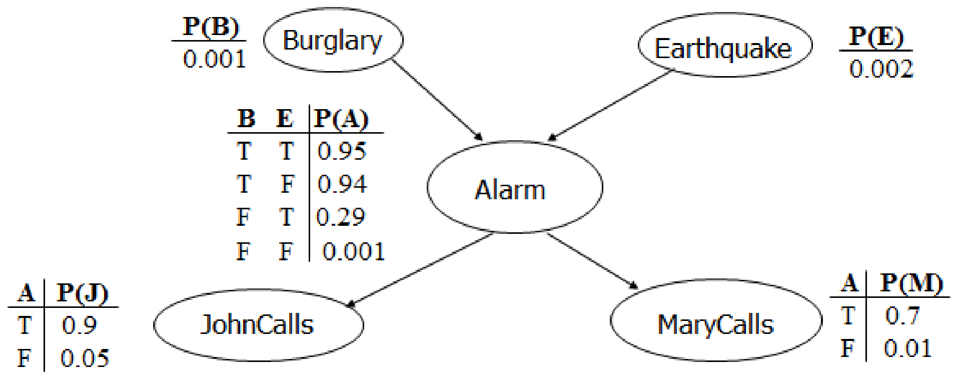
$P(A \wedge V \wedge B) = P(A) * P(V|A) * P(B|V)$



$P(A \wedge V \wedge B) = P(V) * P(A|V) * P(B|V)$



$P(A \wedge V \wedge B) = P(A) * P(B) * P(V|A,B)$



P(B)
0.001

P(E)
0.002

B	E	P(A)
T	T	0.95
T	F	0.94
F	T	0.29
F	F	0.001

A	P(J)
T	0.9
F	0.05

A	P(M)
T	0.7
F	0.01

$P(J \wedge M \wedge A \wedge \sim B \wedge \sim E) =$

$P(J|A) * P(M|A) * P(A|\sim B \wedge \sim E) * P(\sim B) \wedge P(\sim E) = 0.9 * 0.7 * 0.001 * 0.999 * 0.998 = 0.00062$

$P(A|B) = P(A|B,E) * P(E|B) + P(A|B,\sim E) * P(\sim E|B) = P(A|B,E) * P(E) + P(A|B,\sim E) * P(\sim E)$

$= 0.95 * 0.002 + 0.94 * 0.998 = 0.94002$

Different types of inferences

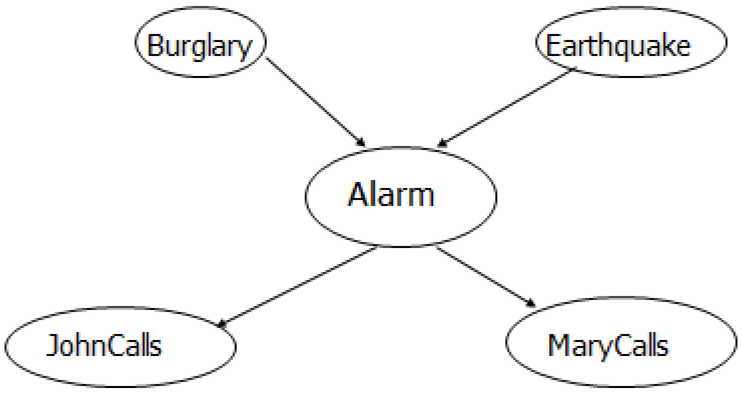
Diagnosis inferences (effect → cause)

P(Burglary | JohnCalls)

Causal inferences (cause → effect)

P(JohnCalls |Burglary),

P(MaryCalls | Burgalry)



Intercausal inferences (between cause and common effects)

P(Burglary | Alarm ∧Earthquake)

Mixed inferences

P(Alarm | JohnCalls ∧ ~Earthquake) → diag + causal

P(Burglary | JohnCalls ∧ ~ Earthquake) → diag + intercausal

3.7 Dempster-Shafer Theory

- Dempster-Shafer theory is an approach to combining evidence
- Dempster (1967) developed means for combining degrees of belief derived from independent items of evidence.
- His student, Glenn Shafer (1976), developed method for obtaining degrees of belief for one question from subjective probabilities for a related question

- People working in Expert Systems in the 1980s saw their approach as ideally suitable for such systems.
- Each fact has a degree of support, between 0 and 1:
  - ✓ 0 No support for the fact
  - ✓ 1 full support for the fact
- Differs from Bayesian approah in that:
  - ✓ Belief in a fact and its negation need not sum to 1.
  - ✓ Both values can be 0 (meaning no evidence for or against the fact)

Set of possible conclusions: Θ

Θ = { θ1 , θ2 , …, θn }

Where:

- ✓ Θ is the set of possible conclusions to be drawn
- ✓ Each θi is mutually exclusive: at most one has to be true.
- ✓ Θ is Exhaustive: At least one θi has to be true.

Frame of discernment

Θ = { θ1 , θ2 , …, θn }

- Bayes was concerned with evidence that supported single conclusions (e.g., evidence for each outcome θi in Θ):
- p(θi| E)
- D-S Theoryis concerned with evidences which support
- subsets of outcomes in Θ, e.g., θ1 v θ2 v θ3 or {θ1, θ2, θ3}
- The "frame of discernment" (or "Power set") of Θ is the set of all possible subsets of Θ:

  E.g., if Θ = {θ1, θ2, θ3}

- Then the frame of discernment of Θ is:

  (Ø, θ1, θ2, θ3, {θ1, θ2}, {θ1, θ3}, {θ2, θ3}, { θ1, θ2, θ3})

- Ø, the empty set, has a probability of 0, since one of the outcomes has to be true.
- Each of the other elements in the power set has a probability between 0 and 1.
- The probability of {θ1, θ2, θ3} is 1.0 since one has to be true.

Mass function m(A):

- (where A is a member of the power set) = proportion of all evidence that supports this element of the power set.
- "The mass m(A) of a given member of the power set, A, expresses the proportion of all relevant and available evidence that supports the claim that the actual state belongs to A but to no particular subset of A."
- "The value of m(A) pertains only to the set A and makes no additional claims about any subsets of A, each of which has, by definition, its own mass.
- Each m(A) is between 0 and 1.
- All m(A) sum to 1.
- m(Ø) is 0 - at least one must be true.

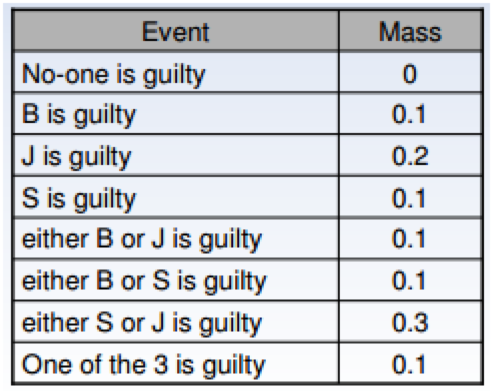
  Interpretation of m({AvB})=0.3

- Means there is evidence for {AvB} that cannot be divided among more specific beliefs for A or B.

  Example

  - 4 people (B, J, S and K) are locked in a room when the lights go out.
  - When the lights come on, K is dead, stabbed with a knife.
  - Not suicide (stabbed in the back)
  - No-one entered the room.
  - Assume only one killer.

- Θ = { B, J, S}
- P(Θ) = (Ø, {B}, {J}, {S}, {B,J}, {B,S}, {J,S}, {B,J,S})
- Detectives, after reviewing the crime-scene, assign mass probabilities to various elements of the power set:



Event	Mass
No-one is guilty	0
B is guilty	0.1
J is guilty	0.2
S is guilty	0.1
either B or J is guilty	0.1
either B or S is guilty	0.1
either S or J is guilty	0.3
One of the 3 is guilty	0.1

Belief in A:

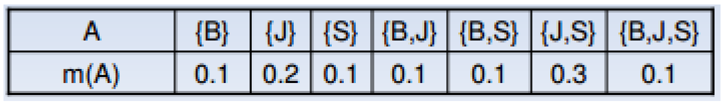
The belief in an element A of the Power set is the sum of the masses of elements which are subsets of A (including A itself).

E.g., given A={q1, q2, q3}

Bel(A) = m(q1)+m(q2)+m(q3)+ m({q1, q2})+m({q2, q3})+m({q1, q3})+m({q1, q2, q3})

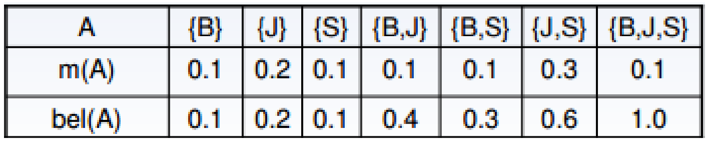
Example

- Given the mass assignments as assigned by the detectives:



A	{B}	{J}	{S}	{B,J}	{B,S}	{J,S}	{B,J,S}
m(A)	0.1	0.2	0.1	0.1	0.1	0.3	0.1

- bel({B}) = m({B}) = 0.1
- bel({B,J}) = m({B})+m({J})+m({B,J}) =0.1+0.2+0.1=0.4
- Result:



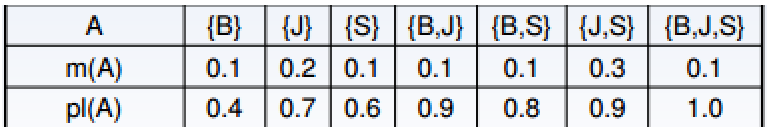
A	{B}	{J}	{S}	{B,J}	{B,S}	{J,S}	{B,J,S}
m(A)	0.1	0.2	0.1	0.1	0.1	0.3	0.1
bel(A)	0.1	0.2	0.1	0.4	0.3	0.6	1.0

Plausibility of A: pl(A)

The plausability of an element A, pl(A), is the sum of all the masses of the sets that intersect with the set A:

E.g. pl({B,J}) = m(B)+m(J)+m(B,J)+m(B,S) +m(J,S)+m(B,J,S) = 0.9

All Results:



A	{B}	{J}	{S}	{B,J}	{B,S}	{J,S}	{B,J,S}
m(A)	0.1	0.2	0.1	0.1	0.1	0.3	0.1
pl(A)	0.4	0.7	0.6	0.9	0.8	0.9	1.0

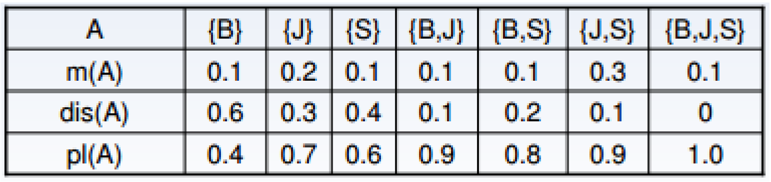
Disbelief (or Doubt) in A: dis(A)

The disbelief in A is simply bel(¬A).

It is calculated by summing all masses of elements which do not intersect with A.

The plausibility of A is thus 1-dis(A):

pl(A) = 1- dis(A)



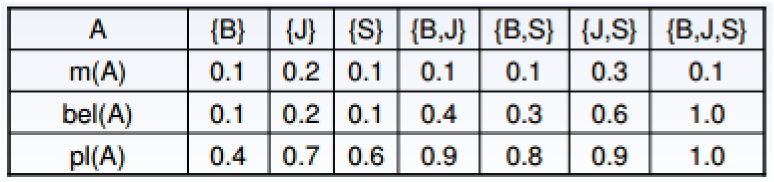
A	{B}	{J}	{S}	{B,J}	{B,S}	{J,S}	{B,J,S}
m(A)	0.1	0.2	0.1	0.1	0.1	0.3	0.1
dis(A)	0.6	0.3	0.4	0.1	0.2	0.1	0
pl(A)	0.4	0.7	0.6	0.9	0.8	0.9	1.0

Belief Interval of A:

The certainty associated with a given subset A is defined by the belief interval:

[bel(A) pl(A)]

E.g. the belief interval of {B,S} is: [0.1 0.8]

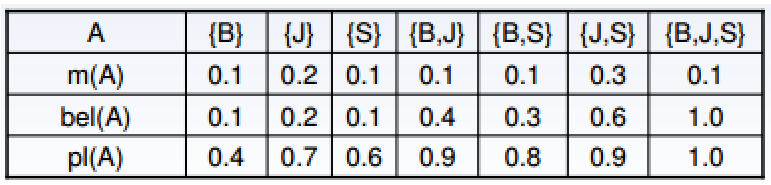


A	{B}	{J}	{S}	{B,J}	{B,S}	{J,S}	{B,J,S}
m(A)	0.1	0.2	0.1	0.1	0.1	0.3	0.1
bel(A)	0.1	0.2	0.1	0.4	0.3	0.6	1.0
pl(A)	0.4	0.7	0.6	0.9	0.8	0.9	1.0

Belief Intervals & Probability

The probability in A falls somewhere between bel(A) and pl(A).

- bel(A) represents the evidence we have for A directly So prob(A) cannot be less than this value.
- pl(A) represents the maximum share of the evidence we could possibly have, if, for all sets that intersect with A, the part that intersects is actually valid. So pl(A) is the maximum possible value of prob(A).



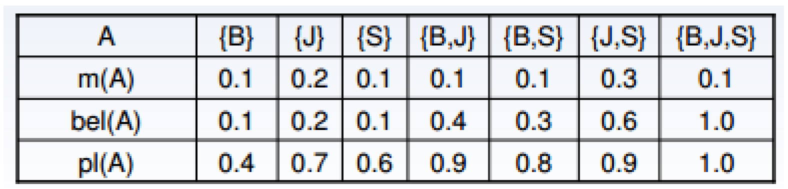
A	{B}	{J}	{S}	{B,J}	{B,S}	{J,S}	{B,J,S}
m(A)	0.1	0.2	0.1	0.1	0.1	0.3	0.1
bel(A)	0.1	0.2	0.1	0.4	0.3	0.6	1.0
pl(A)	0.4	0.7	0.6	0.9	0.8	0.9	1.0

Belief Intervals:

Belief intervals allow Demspter-Shafer theory to reason about the degree of certainty or certainty of our beliefs.

- A small difference between belief and plausibility shows that we are certain about our belief.
- A large difference shows that we are uncertain about our belief.

However, even with a 0 interval, this does not mean we know which conclusion is right. Just how probable it is!



A	{B}	{J}	{S}	{B,J}	{B,S}	{J,S}	{B,J,S}
m(A)	0.1	0.2	0.1	0.1	0.1	0.3	0.1
bel(A)	0.1	0.2	0.1	0.4	0.3	0.6	1.0
pl(A)	0.4	0.7	0.6	0.9	0.8	0.9	1.0

PART – A

1. Define NMR

2. Define Justifications

3. What is non monotonic inference?

4. Difference between JTMS and LTMS

5. Define Bayes theorem.

6. What do you mean by Rule based system?

7. Define fuzzy logic.

8. What is credit assignment problem?

9. Define Frame problems.

14. What do you understand by Default reasoning.

15. Define frames.

16. What are singular extensions?

17. What is a Bayesian network?

18. Define dumpster Shafer theory.

PART-B

1. Distinguish between

    a. Production Based System.

    b. Frame Based System

2. What is uncertainty? How to reason out in each situation? What are the various strategies under such case?

3. Write a note on a. Fuzzy reasoning b. Bayesian probability c. Certainty factors

4. What is certainty factor? Compute certainty factor based on hypothesis.

5. How does inference engine work in a frame based system.

6. Explain Bayesian Network.

UNIT 4

PLANNING AND MACHINE LEARNING

4.1 Planning With State Space Search

The agent first generates a goal to achieve and then constructs a plan to achieve it from the Current state.

Problem Solving To Planning

Representation Using Problem Solving Approach

- ✓ Forward search
- ✓ Backward search
- ✓ Heuristic search

Representation Using Planning Approach

- ✓ STRIPS-standard research institute problem solver.
- ✓ Representation for states and goals
- ✓ Representation for plans
- ✓ Situation space and plan space
- ✓ Solutions

Why Planning?

Intelligent agents must operate in the world. They are not simply passive reasons (Knowledge Representation, reasoning under uncertainty) or problem solvers (Search), they must also act on the world.

We want intelligent agents to act in "intelligent ways". Taking purposeful actions, predicting the expected effect of such actions, composing actions together to achieve complex goals. E.g. if we have a robot we want robot to decide what to do; how to act to achieve our goals.

Planning Problem

How to change the world to suit our needs

Critical issue: we need to reason about what the world will be like after doing a few actions, not just what it is like now

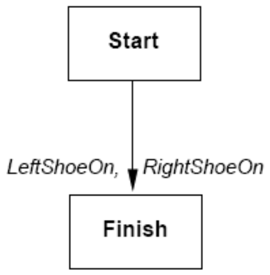
GOAL: Craig has coffee

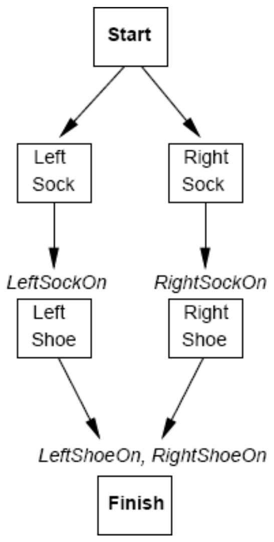
CURRENTLY: robot in mailroom, has no coffee, coffee not made, Craig in office etc.

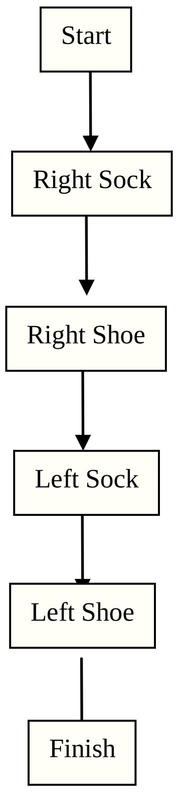
TO DO: goto lounge, make coffee

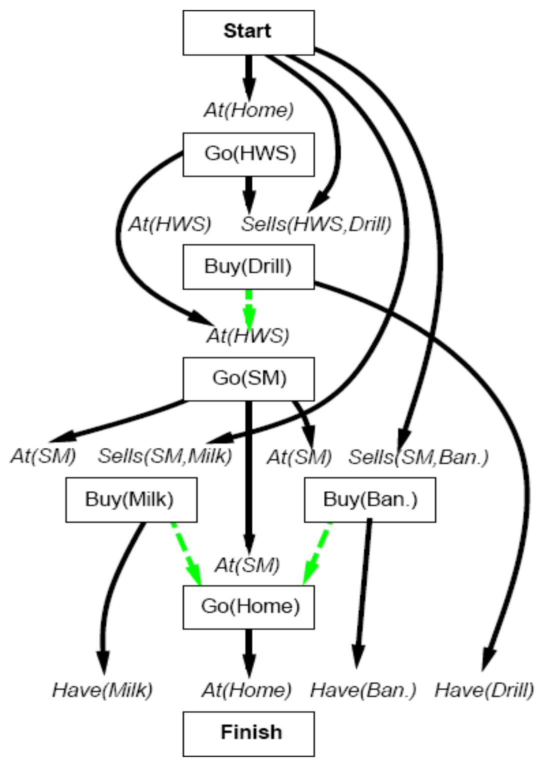
Partial Order Plan

- A partially ordered collection of steps
    - *Start step* has the initial state description and its effect
    - *Finish step* has the goal description as its precondition
    - *Causal links* from outcome of one step to precondition of another step
    - *Temporal ordering* between pairs of steps
- An open condition is a precondition of a step not yet causally linked
- A plan is ***complete*** if every precondition is achieved
- A precondition is ***achieved*** if it is the effect of an earlier step and no possibly intervening step undoes it









Partial Order Plan Algorithm

```
function POP(initial, goal, operators) returns plan
   plan ← MAKE-MINIMAL-PLAN(initial, goal)
   loop do
       if SOLUTION?(plan) then return plan
       $S_{need}$, $c$ ← SELECT-SUBGOAL(plan)
       CHOOSE-OPERATOR(plan, operators, $S_{need}$, $c$)
       RESOLVE-THREATS(plan)
   end

function SELECT-SUBGOAL(plan) returns $S_{need}$, $c$
   pick a plan step $S_{need}$ from STEPS(plan)
       with a precondition $c$ that has not been achieved
   return $S_{need}$, $c$
```

```
procedure CHOOSE-OPERATOR($plan, operators, S_{need}, c$)
    choose a step $S_{add}$ from $operators$ or STEPS($plan$) that has $c$ as an effect
    if there is no such step then fail
    add the causal link $S_{add} \xrightarrow{c} S_{need}$ to LINKS($plan$)
    add the ordering constraint $S_{add} \prec S_{need}$ to ORDERINGS($plan$)
    if $S_{add}$ is a newly added step from $operators$ then
        add $S_{add}$ to STEPS($plan$)
        add $Start \prec S_{add} \prec Finish$ to ORDERINGS($plan$)
```

```
procedure RESOLVE-THREATS($plan$)
    for each $S_{threat}$ that threatens a link $S_i \xrightarrow{c} S_j$ in LINKS($plan$) do
        choose either
            Demotion: Add $S_{threat} \prec S_i$ to ORDERINGS($plan$)
            Promotion: Add $S_j \prec S_{threat}$ to ORDERINGS($plan$)
        if not CONSISTENT($plan$) then fail
    end
```

4.2 Stanford Research Institute Problem Solver (STRIPS)

STRIPS is a classical planning language, representing plan components as states, goals, and actions, allowing algorithms to parse the logical structure of the planning problem to provide a solution.

In STRIPS, state is represented as a conjunction of positive literals. Positive literals may be a propositional literal (e.g., Big ^ Tall) or a first-order literal (e.g., At(Billy, Desk)). The positive literals must be grounded – may not contain a variable (e.g., At(x, Desk)) – and must be function-free – may not invoke a function to calculate a value (e.g., At(Father(Billy), Desk)). Any state conditions that are not mentioned are assumed false.

The goal is also represented as a conjunction of positive, ground literals. A state satisfies a goal if the state contains all of the conjuncted literals in the goal; e.g., Stacked ^ Ordered ^ Purchased satisfies Ordered ^ Stacked.

Actions (or operators) are defined by action schemas, each consisting of three parts:

- The action name and any parameters.
- Preconditions which must hold before the action can be executed. Preconditions are represented as a conjunction of function-free, positive literals. Any variables in a precondition must appear in the action"s parameter list.
- Effects which describe how the state of the environment changes when the action is executed. Effects are represented as a conjunction of function-free literals. Any

variables in a precondition must appear in the action‟s parameter list. Any world state not explicitly impacted by the action schema‟s effect is assumed to remain unchanged.

The following, simple action schema describes the action of moving a box from location x to location y:

Action: *MoveBox*(*x*, *y*)
Precond: *BoxAt*(*x*)
Effect: *BoxAt*(*y*), ¬ *BoxAt*(*x*)

If an action is applied, but the current state of the system does not meet the necessary preconditions, then the action has no effect. But if an action is successfully applied, then any positive literals, in the effect, are added to the current state of the world; correspondingly, any negative literals, in the effect, result in the removal of the corresponding positive literals from the state of the world.

For example, in the action schema above, the effect would result in the proposition BoxAt(*y*) being added to the known state of the world, while BoxAt(*x*) would be *removed* from the known state of the world. (Recall that state only includes positive literals, so a negation effect results in the *removal* of positive literals.) Note also that positive effects can not get duplicated in state; likewise, a negative of a proposition that is not currently in state is simply ignored. For example, if *Open*(*x*) was not previously part of the state, ¬ *Open*(*x*) would have no effect.

A STRIPS problem includes the complete (but relevant) initial state of the world, the goal state(s), and action schemas. A STRIPS algorithm should then be able to accept such a problem, returning a solution. The solution is simply an action sequence that, when applied to the initial state, results in a state which satisfies the goal.

4.2.1 STRIPS Planning Algorithm

As previously referenced, STRIPS began as an automated planning algorithm and has double-meaning to describe the language (described above) used to provide input to that algorithm. While the algorithm does not scale well to real-world problems, it, like the language, serves as a foundational starting point to developing and understanding more powerful automated planning algorithms. The STRIPS algorithm [3] is found below, followed by a brief commentary:

```
STRIPS($A$, $s$, $g$)
$p$ = empty plan
loop…
if $s$ satisfies $g$ then return $p$
$a$ = [an applicable action in $A$, relevant for $g$]
if $a$ = null, then return failure
$p'$ = STRIPS($A$, $s$, precond($a$))
if $p'$ = failure, then return failure
$s$ = apply $p'$ to $s$
```

s = apply *a* to *s*
p = *p* + *p'* + *a*

In the above STRIPS algorithm, *A* represents all of the possible, grounded actions (i.e., action schemas with variables replaced with values), while *s* is the current state, and *g* is the goal state.

Critically important to note is that this algorithm is a *backward-search* algorithm. In other words, the goal state of the planning problem is fed into the algorithm as *s*, while the initial state of the planning problem is provided as *g*. The returned solution, if any, simply needs to be reversed to provide the solution to the planning problem. Simply put, it works backwards from the goal to figure out a logical path to get there.

4.3 Explanation

Most expert systems have explanation facilities that allow the user to ask questions- why and how it reached some conclusion

The questions are answered by referring to the system goals, the rules being used, and existing problem solving. The rules typically reflect empirical or "compiled" knowledge. They are codes of an expert‟s rule of thumb, not the expert‟s deeper understanding.

Example:

Dialog with an expert system designed to give advice on car problems.

System	Does the car start?
User	No.
System	Does the engine turn over?
User	yes
System	Do you smell gas?
User	yes
System	Recommendation-wait 5 minutes and try again.
User	Why?
System	I used the rule:
	If not start, and engine_turn_over, and smell_gas
	Then recommendation is „wait 5 minutes and try again‟

User

Note: The rule gives the correct advice for a flooded car. And knows the questions to be ask to determine if the car is flooded, but it does not contain the knowledge of what a flooded car is and why waiting will help.

Types of Explanation

There are four types of explanations commonly used in expert systems.

- ✓ Rule trace reports on the progress of a consultation;
- ✓ Explanation of how the system reached to the give conclusion;
- ✓ Explanation of why the system did not give any conclusion.
- ✓ Explanation of why the system is asking a question;

4.4 Learning

Machine Learning

- ➢ Like human learning from past experiences,a computer does not have "experiences".
- ➢ A computer system learns from data, which represent some "past experiences" of an application domain.
- ➢ Objective of machine learning : learn a target function that can be used to predict the values of a discrete class attribute, e.g., approve or not-approved, and high-risk or low risk.
- ➢ The task is commonly called: **Supervised learning, classification, or inductive learning**

Supervised Learning

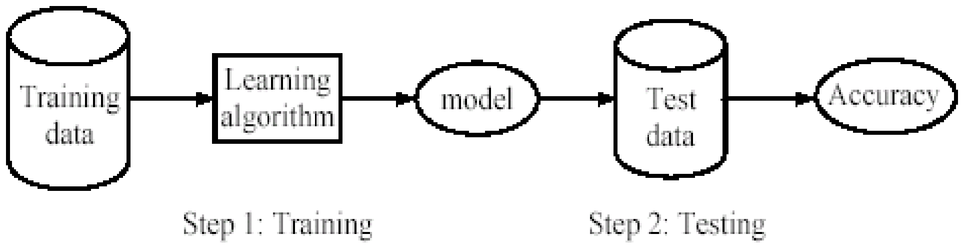
Supervised learning is a machine learning technique for learning a function from training data. The training data consist of pairs of input objects (typically vectors), and desired outputs. The output of the function can be a continuous value (called regression), or can predict a class label of the input object (called classification). The task of the supervised learner is to predict the value of the function for any valid input object after having seen a number of training examples (i.e. pairs of input and target output). To achieve this, the learner has to generalize from the presented data to unseen situations in a "reasonable" way.

Another term for supervised learning is classification. Classifier performance depend greatly on the characteristics of the data to be classified. There is no single classifier that works best on all given problems. Determining a suitable classifier for a given problem is however still more an art than a science. The most widely used classifiers are the Neural Network (Multi-layer Perceptron), Support Vector Machines, k-Nearest Neighbors, Gaussian Mixture Model, Gaussian, Naive Bayes, Decision Tree and RBF classifiers.

Supervised learning process: two steps

- **Learning** (training): Learn a model using the training data
- **Testing**: Test the model using unseen test data to assess the model accuracy

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}},$$



Supervised vs. unsupervised Learning

- **Supervised learning**:

  classification is seen as supervised learning from examples.

  - ✓ **Supervision**: The data (observations, measurements, etc.) are labeled with pre-defined classes. It is like that a "teacher" gives the classes (supervision).
  - ✓ **Test data** are classified into these classes too.

- **Unsupervised learning** (clustering)

  - ✓ Class labels of the data are unknown
  - ✓ Given a set of data, the task is to establish the existence of classes or clusters in the data

Decision Tree

- A decision tree takes as input an object or situation described by a set of attributes and returns a “decision” – the predicted output value for the input.
- A decision tree reaches its decision by performing a sequence of tests.

  Example : “HOW TO” manuals (for car repair)

A decision tree reaches its decision by performing a sequence of tests. Each internal node in the tree corresponds to a test of the value of one of the properties, and the branches from the node are labeled with the possible values of the test. Each leaf node in the tree specifies the value to be returned if that leaf is reached. The decision tree representation seems to be very natural for humans; indeed, many "How To" manuals (e.g., for car repair) are written entirely as a single decision tree stretching over hundreds of pages.

A somewhat simpler example is provided by the problem of whether to wait for a table at a restaurant. The aim here is to learn a definition for the goal predicate Will Wait. In setting this up as a learning problem, we first have to state what attributes are available to describe examples in the domain. we will see how to automate this task; for now, let's suppose we decide on the following list of attributes:

1. Alternate: whether there is a suitable alternative restaurant nearby.

2. Bar: whether the restaurant has a comfortable bar area to wait in.

3. Fri/Sat: true on Fridays and Saturdays.

4. Hungry: whether we are hungry.

5. Patrons: how many people are in the restaurant (values are None, Some, and Full).

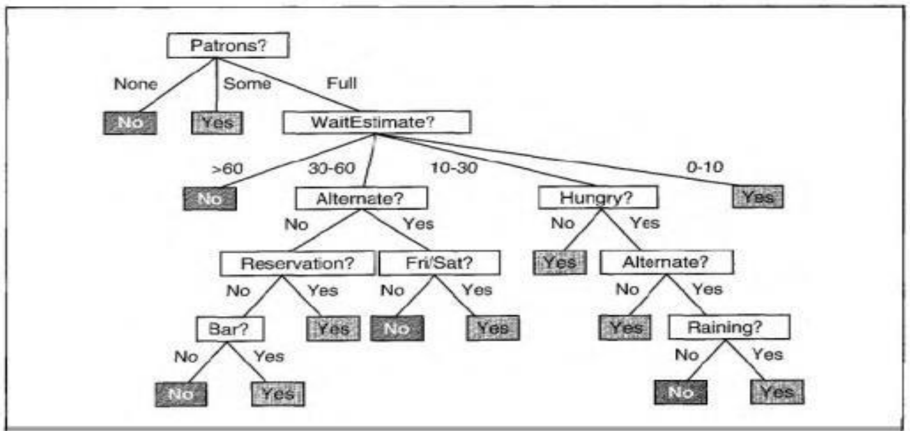
6. Price: the restaurant's price range ($, $$, $$$).

7. Raining: whether it is raining outside.

8. Reservation: whether we made a reservation.

9. Type: the kind of restaurant (French, Italian, Thai, or burger).

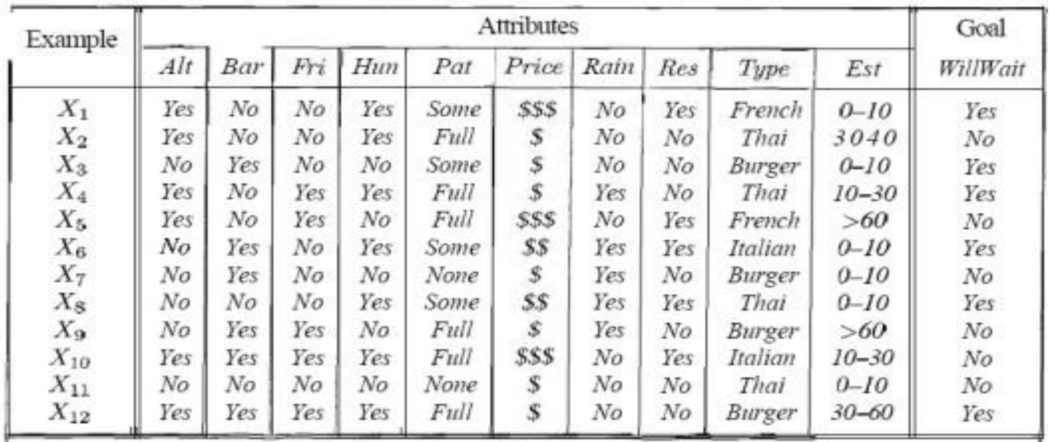
10. Wait Estimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).



A decision tree for deciding whether to wait for a table.

Decision tree induction from examples

An example for a Boolean decision tree consists of a vector of' input attributes, X, and a single Boolean output value y. A set of examples (X1,Y1) . . . , (X2, y2) is shown in Figure. The positive examples are the ones in which the goal *Will Wait* is true (XI, ***X3,*** . . .); the negative examples are the ones in which it is false (X2, *X5*, . . .). The complete set of examples is called the **training set.**

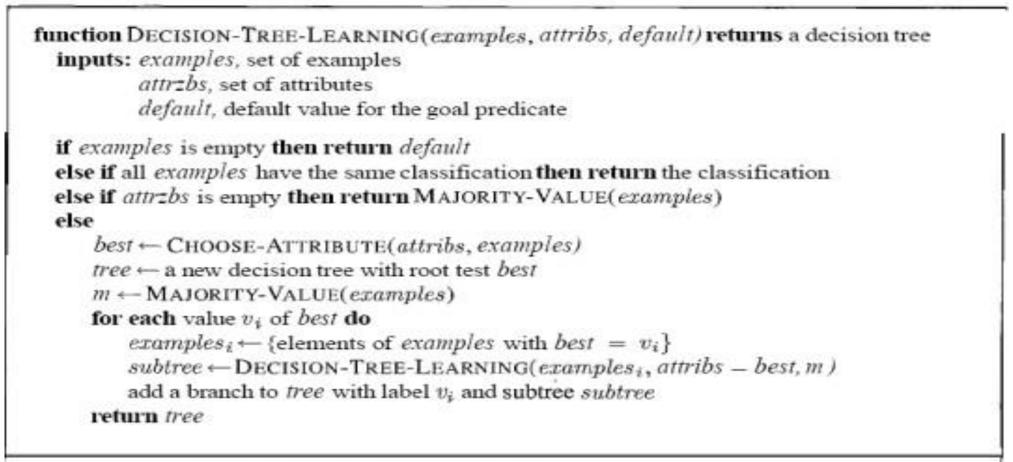


Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*3040*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	*Yes*

Examples for the restaurant domain.

Decision Tree Algorithm

The basic idea behind the Decision-Tree-Learning-Algorithm is to test the most important attribute first. By "most important," we mean the one that makes the most difference to the classification of an example. That way, we hope to get to the correct classification with a small number of tests, meaning that all paths in the tree will be short and the tree as a whole will be small.



```
function DECISION-TREE-LEARNING(examples, attribs, default) returns a decision tree
  inputs: examples, set of examples
          attrzbs, set of attributes
          default, default value for the goal predicate

  if examples is empty then return default
  else if all examples have the same classification then return the classification
  else if attrzbs is empty then return MAJORITY-VALUE(examples)
  else
      best ← CHOOSE-ATTRIBUTE(attribs, examples)
      tree ← a new decision tree with root test best
      m ← MAJORITY-VALUE(examples)
      for each value $v_i$ of best do
          $examples_i$ ← {elements of examples with $best = v_i$}
          subtree ← DECISION-TREE-LEARNING($examples_i$, attribs − best, m)
          add a branch to tree with label $v_i$ and subtree subtree
      return tree
```

The decision tree learning algorithm.

Reinforcement Learning

- ➢ Learning what to do to maximize reward
  - ✓ Learner is not given training
  - ✓ Only feedback is in terms of reward
  - ✓ Try things out and see what the reward is
- ➢ Different from Supervised Learning
  - ✓ Teacher gives training examples

Examples

- ➢ Robotics: Quadruped Gait Control, Ball Acquisition (Robocup)
- ➢ Control: Helicopters
- ➢ Operations Research: Pricing, Routing, Scheduling
- ➢ Game Playing: Backgammon, Solitaire, Chess, Checkers
- ➢ Human Computer Interaction: Spoken Dialogue Systems
- ➢ Economics/Finance: Trading

Markov decision process VS Reinforcement Learning

- ➢ Markov decision process

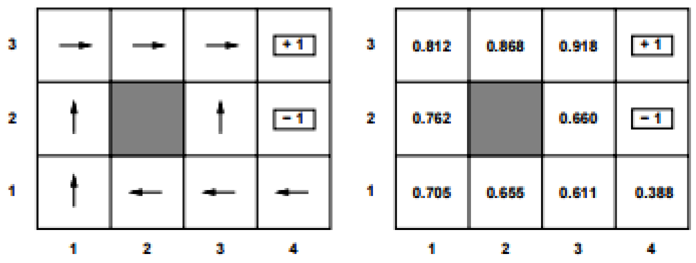
- ✓ Set of state S, set of actions A
- ✓ Transition probabilities to next states T(s, a, a‟)
- ✓ Reward functions R(s)

- ➢ RL is based on MDPs, but
  - ✓ Transition model is not known
  - ✓ Reward model is not known
- ➢ MDP computes an optimal policy
- ➢ RL learns an optimal policy

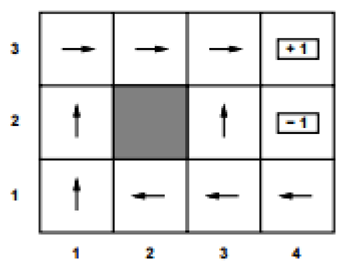
Types of Reinforcement Learning

- ➢ Passive Vs Active
  - ✓ Passive: Agent executes a fixed policy and evaluates it
  - ✓ Active: Agents updates policy as it learns
- ➢ Model based Vs Model free
- ➢ Model-based: Learn transition and reward model, use it to get optimal policy
- ➢ Model free: Derive optimal policy without learning the model

Passive Learning



- ➢ Evaluate how good a policy π is
- ➢ Learn the utility $U^{\pi}(s)$ of each state
- ➢ Same as policy evaluation for known transition & reward models



Agent executes a sequence of trials:

(1, 1) → (1, 2) → (1, 3) → (1, 2) → (1, 3) → (2, 3) → (3, 3) → (4, 3)$_{+1}$
(1, 1) → (1, 2) → (1, 3) → (2, 3) → (3, 3) → (3, 2) → (3, 3) → (4, 3)$_{+1}$
(1, 1) → (2, 1) → (3, 1) → (3, 2) → (4, 2)$_{-1}$

Goal is to learn the expected utility $U_\pi(s)$

$$U^\pi(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi, s_0 = s\right]$$

Direct Utility Estimation

- ➢ Reduction to inductive learning
  - ✓ Compute the empirical value of each state
  - ✓ Each trial gives a sample value
  - ✓ Estimate the utility based on the sample values
- ➢ Example: First trial gives
  - ✓ State (1,1): A sample of reward 0.72
  - ✓ State (1,2): Two samples of reward 0.76 and 0.84
  - ✓ State (1,3): Two samples of reward 0.80 and 0.88
- ➢ Estimate can be a running average of sample values
- ➢ Example: U(1, 1) = 0.72,U(1, 2) = 0.80,U(1, 3) = 0.84, . . .
- ➢ Ignores a very important source of information

- ➢ The utility of states satisfy the Bellman equations

$$U^{\pi}(s) = R(s) + \gamma \sum_{s'} T(s, \pi(s), s')U^{\pi}(s')$$

- ➢ Search is in a hypothesis space for U much larger than needed
- ➢ Convergence is very slow
- ➢ Make use of Bellman equations to get $U^{\pi}(s)$
- ➢ Need to estimate T(s, π(s), s‟) and R(s) from trials
- ➢ Plug-in learnt transition and reward in the Bellman equations
- ➢ Solving for U^{π}: System of n linear equations
- ➢ Estimates of T and R keep changing
- ➢ Make use of modified policy iteration idea
  - ✓ Run few rounds of value iteration
  - ✓ Initialize value iteration from previous utilities
  - ✓ Converges fast since T and R changes are small
- ➢ ADP is a standard baseline to test „smarter‟ ideas
- ➢ ADP is inefficient if state space is large
  - ✓ Has to solve a linear system in the size of the state space
  - ✓ Backgammon: 10^{50} linear equations in 10^{50} unknowns

Temporal Difference Learning

- ➢ Best of both worlds
  - ✓ Only update states that are directly affected
  - ✓ Approximately satisfy the Bellman equations
  - ✓ Example:

    $(1, 1) \rightarrow (1, 2) \rightarrow (1, 3) \rightarrow (1, 2) \rightarrow (1, 3) \rightarrow (2, 3) \rightarrow (3, 3) \rightarrow (4, 3)_{+1}$

    $(1, 1) \rightarrow (1, 2) \rightarrow (1, 3) \rightarrow (2, 3) \rightarrow (3, 3) \rightarrow (3, 2) \rightarrow (3, 3) \rightarrow (4, 3)_{+1}$

$(1, 1) \rightarrow (2, 1) \rightarrow (3, 1) \rightarrow (3, 2) \rightarrow (4, 2)_{-1}$

- After the first trial, U(1, 3) = 0.84,U(2, 3) = 0.92
- Consider the transition (1, 3) → (2, 3) in the second trial
- If deterministic, then U(1, 3) = −0.04 + U(2, 3)
- How to account for probabilistic transitions (without a model)

➢ TD chooses a middle ground

$$U^{\pi}(s) \leftarrow (1-\alpha)U^{\pi}(s) + \alpha(R(s) + \gamma U^{\pi}(s'))$$

➢ Temporal difference (TD) equation, α is the learning rate

➢ The TD equation

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(R(s) + \gamma U^{\pi}(s') - U^{\pi}(s))$$

➢ TD applies a correction to approach the Bellman equations

✓ The update for s‟ will occur T(s, π(s), s‟) fraction of the time

✓ The correction happens proportional to the probabilities

✓ Over trials, the correction is same as the expectation

➢ Learning rate α determines convergence to true utility

✓ Decrease α_s proportional to the number of state visits

✓ Convergence is guaranteed if

$$\sum_{m=1}^{\infty} \alpha_s(m) = \infty \qquad \sum_{m=1}^{\infty} \alpha_s^2(m) < \infty$$

✓ Decay α_s (m) = 1/m satisfies the condition

➢ TD is model free

TD Vs ADP

➢ TD is mode free as opposed to ADP which is model based

➢ TD updates observed successor rather than all successors

- The difference disappears with large number of trials
- TD is slower in convergence, but much simpler computation per observation

Active Learning

- Agent updates policy as it learns
- Goal is to learn the optimal policy
- Learning using the passive ADP agent
  - ✓ Estimate the model R(s),T(s, a, s‟) from observations
  - ✓ The optimal utility and action satisfies

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s')U(s')$$

  - ✓ Solve using value iteration or policy iteration

- Agent has “optimal” action
- Simply execute the “optimal” action

Exploitation vs Exploration

- The passive approach gives a greedy agent
- Exactly executes the recipe for solving MDPs
- Rarely converges to the optimal utility and policy
  - ✓ The learned model is different from the true environment
- Trade-off
  - ✓ Exploitation: Maximize rewards using current estimates
  - ✓ Agent stops learning and starts executing policy
  - ✓ Exploration: Maximize long term rewards
  - ✓ Agent keeps learning by trying out new things
- Pure Exploitation

- ✓ Mostly gets stuck in bad policies

- ➢ Pure Exploration
  - ✓ Gets better models by learning
  - ✓ Small rewards due to exploration
- ➢ The multi-armed bandit setting
  - ✓ A slot machine has one lever, a one-armed bandit
  - ✓ n-armed bandit has n levers
- ➢ Which arm to pull?
  - ✓ Exploit: The one with the best pay-off so far
  - ✓ Explore: The one that has not been tried

Exploration

- ➢ Greedy in the limit of infinite exploration (GLIE)
  - ✓ Reasonable schemes for trade off
- ➢ Revisiting the greedy ADP approach
  - ✓ Agent must try each action infinitely often
  - ✓ Rules out chance of missing a good action
  - ✓ Eventually must become greedy to get rewards
- ➢ Simple GLIE
  - ✓ Choose random action 1/t fraction of the time
  - ✓ Use greedy policy otherwise
- ➢ Converges to the optimal policy
- ➢ Convergence is very slow

Exploration Function

- A smarter GLIE
  - ✓ Give higher weights to actions not tried very often
  - ✓ Give lower weights to low utility actions
- Alter Bellman equations using optimistic utilities $U^{+}(s)$

$$U^{+}(s) = R(s) + \gamma \max_{a} f\left(\sum_{s'} T(s,a,s')U^{+}(s'), N(a,s)\right)$$

- The exploration function f (u, n)
  - ✓ Should increase with expected utility u
  - ✓ Should decrease with number of tries n
- A simple exploration function

$$f(u,n) = \begin{cases} R^{+}, & \text{if } n < N \\ u, & \text{otherwise} \end{cases}$$

- Actions towards unexplored regions are encouraged
- Fast convergence to almost optimal policy in practice

Q-Learning

- Exploration function gives a active ADP agent
- A corresponding TD agent can be constructed
  - ✓ Surprisingly, the TD update can remain the same
  - ✓ Converges to the optimal policy as active ADP
  - ✓ Slower than ADP in practice
- Q-learning learns an action-value function Q(a; s)
  - ✓ Utility values $U(s) = \max_a Q(a; s)$
- A model-free TD method
  - ✓ No model for learning or action selection

- Constraint equations for Q-values at equilibrium

$$Q(a,s) = R(s) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q(a',s')$$

- Can be updated using a model for T(s; a; s‟)
- The TD Q-learning does not require a model

$$Q(a,s) \leftarrow Q(a,s) + \alpha(R(s) + \gamma \max_{a'} Q(a',s') - Q(a,s))$$

- Calculated whenever a in s leads to s‟
- The next action a_{next} = $\text{argmax}_{a‟}$ f (Q(a‟; s‟);N(s‟; a‟))
- Q-learning is slower than ADP
- Trade-o: Model-free vs knowledge-based methods

PART- A

1. What are the components of planning system?
2. What is planning?
3. What is nonlinear plan?
4. List out the 3 types of machine learning?
5. What is Reinforcement Learning?
6. What do you mean by goal stack planning?
7. Define machine learning.
8. What are the types of Reinforcement Learning.

PART B

1. Briefly explain the advanced plan generation systems.

2. Explain Machine Learning.
3. Explain STRIPS.
4. Explain Reinforcement Learning.
5. Briefly explain Partial Order Plan.
6. Explain in detail about various Machine learning methods.

UNIT 5

CHAPTER 1

5.1 EXPERT SYSTEMS

An expert system is a computer program that represents and reasons with knowledge of some specialist subject with a view to solving problems or giving advice.

To solve expert-level problems, expert systems will need efficient access to a substantial domain knowledge base, and a reasoning mechanism to apply the knowledge to the problems they are given. Usually they will also need to be able to explain, to the users who rely on them, how they have reached their decisions.

They will generally build upon the ideas of knowledge representation, production rules, search, and so on, that we have already covered.

Often we use an expert system shell which is an existing knowledge independent framework into which domain knowledge can be inserted to produce a working expert system. We can thus avoid having to program each new system from scratch.

5.2 Typical Tasks for Expert Systems

There are no fundamental limits on what problem domains an expert system can be built to deal with. Some typical existing expert system tasks include:

1. The interpretation of data

    Such as sonar data or geophysical measurements

2. Diagnosis of malfunctions

    Such as equipment faults or human diseases

3. Structural analysis or configuration of complex objects

    Such as chemical compounds or computer systems

4. Planning sequences of actions

    Such as might be performed by robots

5. Predicting the future

Such as weather, share prices, exchange rates

However, these days, “conventional” computer systems can also do some of these things

5.3 Characteristics of Expert Systems

Expert systems can be distinguished from conventional computer systems in that:

1. They simulate human reasoning about the problem domain, rather than simulating the domain itself.

2. They perform reasoning over representations of human knowledge, in addition to doing numerical calculations or data retrieval. They have corresponding distinct modules referred to as the inference engine and the knowledge base.

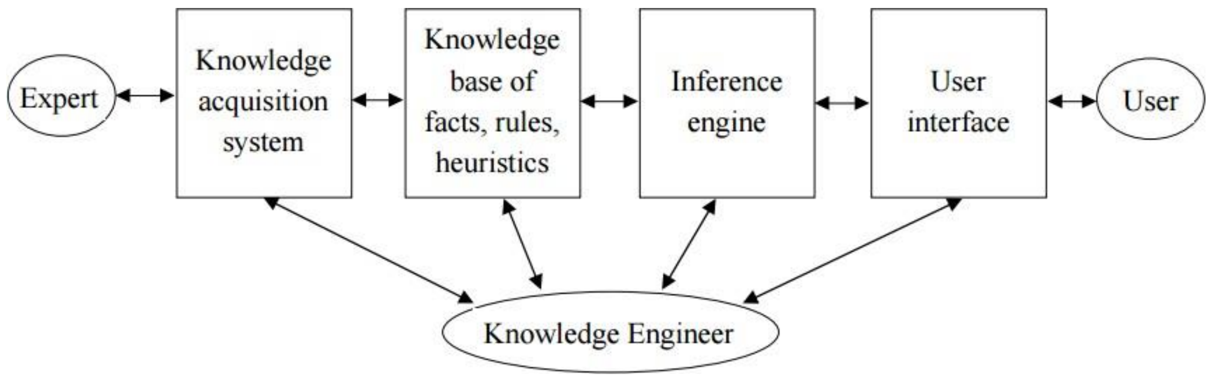
3. Problems tend to be solved using heuristics (rules of thumb) or approximate methods or probabilistic methods which, unlike algorithmic solutions, are not guaranteed to result in a correct or optimal solution.

4. They usually have to provide explanations and justifications of their solutions or recommendations in order to convince the user that their reasoning is correct.

Note that the term Intelligent Knowledge Based System (IKBS) is sometimes used as a synonym for Expert System.

5.3 The Architecture of Expert Systems

The process of building expert systems is often called knowledge engineering. The knowledge engineer is involved with all components of an expert system:


Expert
Knowledge acquisition system
Knowledge base of facts, rules, heuristics
Inference engine
User interface
User
Knowledge Engineer

Building expert systems is generally an iterative process. The components and their interaction will be refined over the course of numerous meetings of the knowledge engineer with the experts and users. We shall look in turn at the various components.

5.3.1 Knowledge Acquisition

The knowledge acquisition component allows the expert to enter their knowledge or expertise into the expert system, and to refine it later as and when required.

Historically, the knowledge engineer played a major role in this process, but automated systems that allow the expert to interact directly with the system are becoming increasingly common.

The knowledge acquisition process is usually comprised of three principal stages:

1. Knowledge elicitation is the interaction between the expert and the knowledge engineer/program to elicit the expert knowledge in some systematic way.

2. The knowledge thus obtained is usually stored in some form of human friendly intermediate representation.

3. The intermediate representation of the knowledge is then compiled into an executable form (e.g. production rules) that the inference engine can process.

In practice, much iteration through these three stages is usually required!

5.3.2 Knowledge Elicitation

The knowledge elicitation process itself usually consists of several stages:

1. Find as much as possible about the problem and domain from books, manuals, etc. In particular, become familiar with any specialist terminology and jargon.

2. Try to characterize the types of reasoning and problem solving tasks that the system will be required to perform.

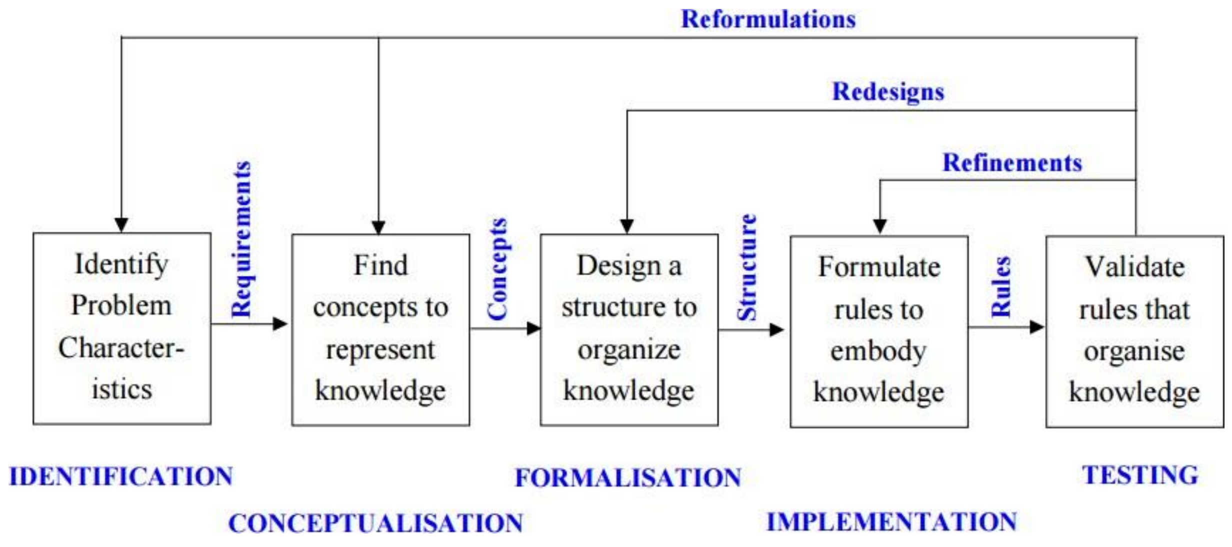
3. Find an expert (or set of experts) that is willing to collaborate on the project. Sometimes experts are frightened of being replaced by a computer system!

4. Interview the expert (usually many times during the course of building the system). Find out how they solve the problems your system will be expected to solve. Have them check and refine your intermediate knowledge representation.

This is a time intensive process, and automated knowledge elicitation and machine learning techniques are increasingly common modern alternatives.

5.3.3 Stages of Knowledge Acquisition

The iterative nature of the knowledge acquisition process can be represented in the following diagram.


Reformulations
Redesigns
Refinements
Identify Problem Character-istics
Requirements
Find concepts to represent knowledge
Concepts
Design a structure to organize knowledge
Structure
Formulate rules to embody knowledge
Rules
Validate rules that organise knowledge
IDENTIFICATION
CONCEPTUALISATION
FORMALISATION
IMPLEMENTATION
TESTING

5.3.4 Levels of Knowledge Analysis

Knowledge identification: Use in depth interviews in which the knowledge engineer encourages the expert to talk about how they do what they do. The knowledge engineer should understand the domain well enough to know which objects and facts need talking about.

Knowledge conceptualization: Find the primitive concepts and conceptual relations of the problem domain.

Epistemological analysis: Uncover the structural properties of the conceptual knowledge, such as taxonomic relations (classifications).

Logical analysis: Decide how to perform reasoning in the problem domain. This kind of knowledge can be particularly hard to acquire.

Implementation analysis: Work out systematic procedures for implementing and testing the system.

Capturing Tacit/Implicit Knowledge

One problem that knowledge engineers often encounter is that the human experts use tacit/implicit knowledge (e.g. procedural knowledge) that is difficult to capture.

There are several useful techniques for acquiring this knowledge:

1. **Protocol analysis**: Tape-record the expert thinking aloud while performing their role and later analyze this. Break down their protocol/account into the smallest atomic units of thought, and let these become operators.

2. **Participant observation**: The knowledge engineer acquires tacit knowledge through practical domain experience with the expert.

3. **Machine induction**: This is useful when the experts are able to supply examples of the results of their decision making, even if they are unable to articulate the underlying knowledge or reasoning process.

Which is/are best to use will generally depend on the problem domain and the expert.

5.3.5 Representing the Knowledge

We have already looked at various types of knowledge representation. In general, the knowledge acquired from our expert will be formulated in two ways:

1. **Intermediate representation** – a structured knowledge representation that the knowledge engineer and expert can both work with efficiently.

2. **Production system** – a formulation that the expert system's inference engine can process efficiently.

It is important to distinguish between:

1. **Domain knowledge** – the expert's knowledge which might be expressed in the form of rules, general/default values, and so on.

2. **Case knowledge** – specific facts/knowledge about particular cases, including any derived knowledge about the particular cases.

The system will have the domain knowledge built in, and will have to integrate this with the different case knowledge that will become available each time the system is used.

CHAPTER -2

5.4 Meta Knowledge

Knowledge about knowledge

- Meta knowledge can be simply defined as knowledge about knowledge.
- Meta knowledge is knowledge about the use and control of domain knowledge in an expert system.

5.5 Roles in Expert System Development

Three fundamental roles in building expert systems are:

1. Expert - Successful ES systems depend on the experience and application of knowledge that the people can bring to it during its development. Large systems generally require multiple experts.

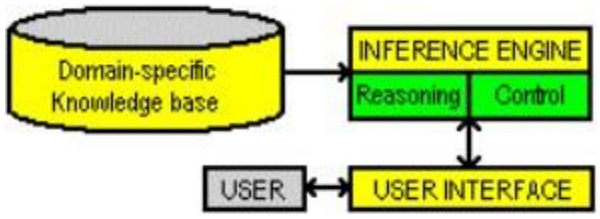
2. Knowledge engineer - The knowledge engineer has a dual task. This person should be able to elicit knowledge from the expert, gradually gaining an understanding of an area of expertise. Intelligence, tact, empathy, and proficiency in specific techniques of knowledge acquisition are all required of a knowledge engineer. Knowledge-acquisition techniques include conducting interviews with varying degrees of structure, protocol analysis, observation of experts at work, and analysis of cases.

On the other hand, the knowledge engineer must also select a tool appropriate for the project and use it to represent the knowledge with the application of the knowledge acquisition facility.

3. User - A system developed by an end user with a simple shell, is built rather quickly an inexpensively. Larger systems are built in an organized development effort. A prototype-oriented iterative development strategy is commonly used. ESs lends themselves particularly well to prototyping.

5.6 Typical Expert System

1. A problem-domain-specific knowledge base that stores the encoded knowledge to support one problem domain such as diagnosing why a car won't start. In a rule-based expert system, the knowledge base includes the if-then rules and additional specifications that control the course of the interview.



2. An inference engine a set of rules for making deductions from the data and that implements the reasoning mechanism and controls the interview process. The inference engine might be generalized so that the same software is able to process many different knowledge bases.

3. The user interface requests information from the user and outputs intermediate and final results. In some expert systems, input is acquired from additional sources such as data bases and sensors.

An expert system shell consists of a generalized inference engine and user interface designed to work with a knowledge base provided in a specified format. A shell often includes tools that help with the design, development and testing of the knowledge base. With the shell approach, expert systems representing many different problem domains may be developed and delivered with the same software environment. .

There are special high level languages used to program expert systems egg PROLOG

The user interacts with the system through a user interface which may use menus, natural language or any other style of interaction). Then an inference engine is used to reason with both the expert knowledge (extracted from our friendly expert) and data specific to the particular problem being solved. The expert knowledge will typically be in the form of a set of IF-THEN rules. The case specific data includes both data provided by the user and partial conclusions (along with certainty measures) based on this data. In a simple forward chaining rule-based system the case specific data will be the elements in working memory.

How an expert system works Car engine diagnosis

1. IF engine_getting_petrol

   AND engine_turns_over

   THEN problem_with_spark_plugs

2. IF NOT engine_turns_over

   AND NOT lights_come_on

   THEN problem_with_battery

3. IF NOT engine_turns_over

   AND lights_come_on

   THEN problem_with_starter

4. IF petrol_in_fuel_tank

   THEN engine_getting_petrol

There are three possible problems with the car:

- problem_with_spark_plugs,
- problem_with_battery,
- problem_with_starter.

The system will ask the user:

- Is it true that there's petrol in the fuel tank?

Let's say that the answer is yes. This answer would be recorded, so that the user doesn't get asked the same question again. Anyway, the system now has proved that the engine is getting petrol, so now wants to find out if the engine turns over. As the system doesn't yet know whether this is the case, and as there are no rules which conclude this, the user will be asked:

- Is it true that the engine turns over?

Lets say this time the answer is no. There are no other rules which can be used to prove ``problem_with_spark_plugs'' so the system will conclude that this is not the solution to the problem, and will consider the next hypothesis: problem_with_battery. It is true that the engine does not turn over (the user has just said that), so all it has to prove is that the lights don't come one. It will ask the user

- Is it true that the lights come on?

Suppose the answer is no. It has now proved that the problem is with the battery. Some systems might stop there, but usually there might be more than one solution, (e.g., more than one fault with the car), or it will be uncertain which of various solutions is the right one. So usually all hypotheses are considered. It will try to prove ``problem_with_starter'', but given the existing data (the lights come on) the proof will fail, so the system will conclude that the problem is with the battery. A complete interaction with our very simple system might be:

- System: Is it true that there's petrol in the fuel tank?

User: Yes.

System: Is it true that the engine turns over?

User: No.

System Is it true that the lights come on?

User: No.

System: I conclude that there is a problem with battery.

Note that in general, solving problems using backward chaining involves searching through all the possible ways of proving the hypothesis, systematically checking each of them.

Questions

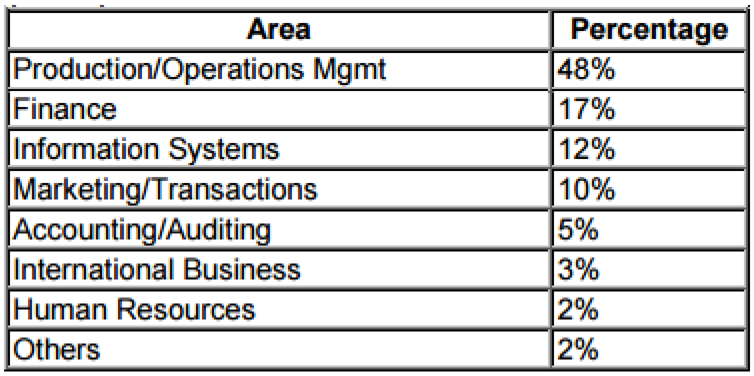
1. ``Briefly describe the basic architecture of a typical expert system, mentioning the function of each of the main components.''

2. ``A travel agent asks you to design an expert system to help people choose where to go on holiday. Design a set of decisions to help you give advice on which holiday to take.

Expert System Use

Expert systems are used in a variety of areas, and are still the most popular developmental approach in the artificial intelligence world.

The table below depicts the percentage of expert systems being developed in particular areas:



Area	Percentage
Production/Operations Mgmt	48%
Finance	17%
Information Systems	12%
Marketing/Transactions	10%
Accounting/Auditing	5%
International Business	3%
Human Resources	2%
Others	2%

- Medical screening for cancer and brain tumours
- Matching people to jobs
- Training on oil rigs
- Diagnosing faults in car engines
- Legal advisory systems
- Mineral prospecting

5.6.1 MYCIN

Tasks and Domain

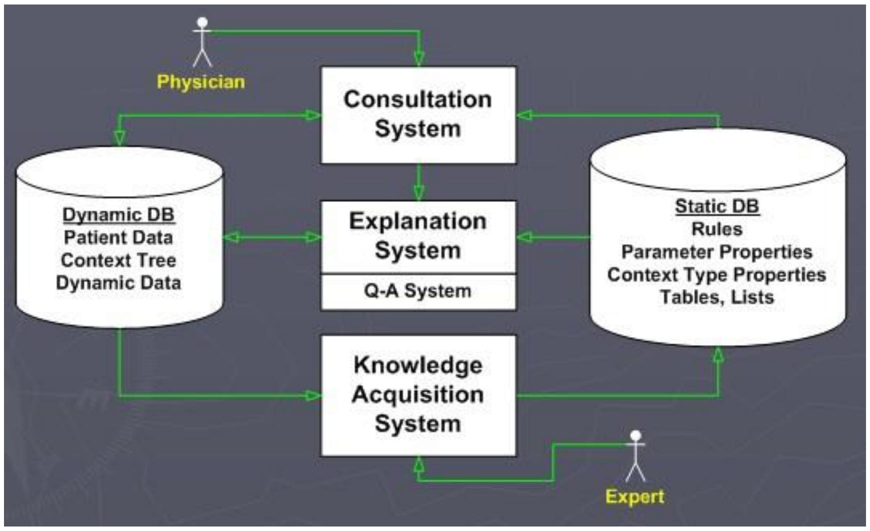
- ▶ Disease DIAGNOSIS and Therapy SELECTION
- ▶ Advice for non-expert physicians with time considerations and incomplete evidence on:
    - ▪ Bacterial infections of the blood
    - ▪ Expanded to meningitis and other ailments

System Goals

- ▶ Utility

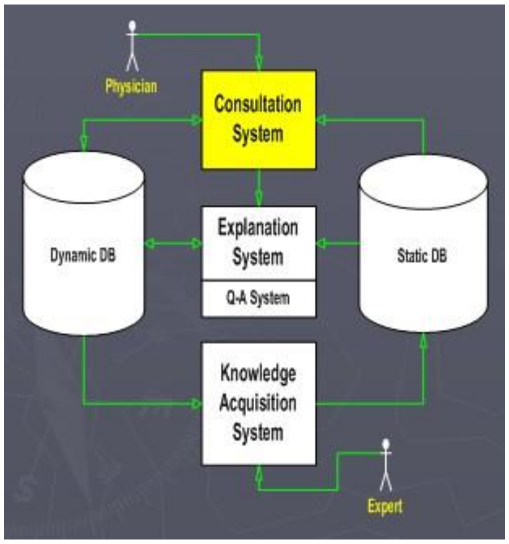
  - Be useful, to attract assistance of experts
  - Demonstrate competence
  - Fulfill domain need (i.e. penicillin)
- Flexibility
  - Domain is complex, variety of knowledge types
  - Medical knowledge rapidly evolves, must be easy to maintain K.B.
- Interactive Dialogue
  - Provide coherent explanations (symbolic reasoning paradigm)
  - Allow for real-time K.B. updates by experts
- Fast and Easy
  - Meet time constraints of the medical field

Architecture



Consultation System

- ► Performs Diagnosis and Therapy Selection
- ► Control Structure reads Static DB (rules) and read/writes to Dynamic DB (patient, context)
- ► Linked to Explanations
- ► Terminal interface to Physician



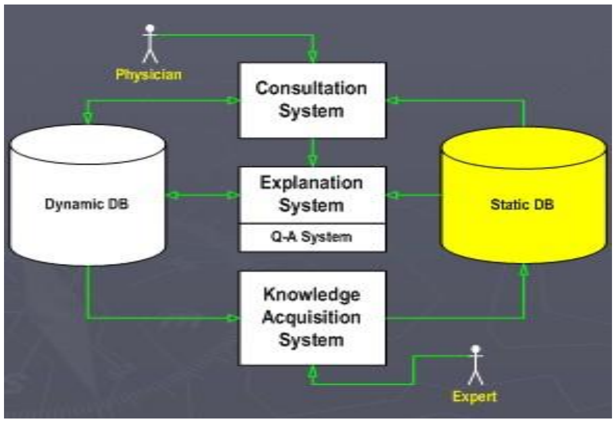
- ► User-Friendly Features:

- Users can request rephrasing of questions
- Synonym dictionary allows latitude of user responses
- User typos are automatically fixed

- Questions are asked when more data is needed
    - If data cannot be provided, system ignores relevant rules
- Goal-directed Backward-chaining Depth-first Tree Search
- High-level Algorithm:
    - Determine if Patient has significant infection
    - Determine likely identity of significant organisms
    - Decide which drugs are potentially useful
    - Select best drug or coverage of drugs

Static Database

- Rules
- Meta-Rules
- Templates
- Rule Properties
- Context Properties
- Fed from Knowledge Acquisition System



Production Rules

- ► Represent Domain-specific Knowledge
- ► Over 450 rules in MYCIN
- ► Premise-Action (If-Then) Form:

  <predicate function><object><attrib><value>

- ► Each rule is completely modular, all relevant context is contained in the rule with explicitly stated premises

MYCIN P.R. Assumptions

- ► Not every domain can be represented, requires formalization (EMYCIN)
- ► Only small number of simultaneous factors (more than 6 was thought to be unwieldy)
- ► IF-THEN formalism is suitable for Expert Knowledge Acquisition and Explanation sub-systems

Judgmental Knowledge

- ► Inexact Reasoning with Certainty Factors (CF)
- ► CF are not Probability!
- ► Truth of a Hypothesis is measured by a sum of the CFs

    - Premises and Rules added together
    - Positive sum is confirming evidence
    - Negative sum is disconfirming evidence

Sub-goals

- At any given time MYCIN is establishing the value of some parameter by sub-goaling
- Unity Paths: a method to bypass sub-goals by following a path whose certainty is known (CF==1) to make a definite conclusion
- Won't search a sub-goal if it can be obtained from a user first (i.e. lab data)

Preview Mechanism

- Interpreter reads rules before invoking them
- Avoids unnecessary deductive work if the sub-goal has already been tested/determined
- Ensures self-referencing sub-goals do not enter recursive infinite loops

Meta-Rules

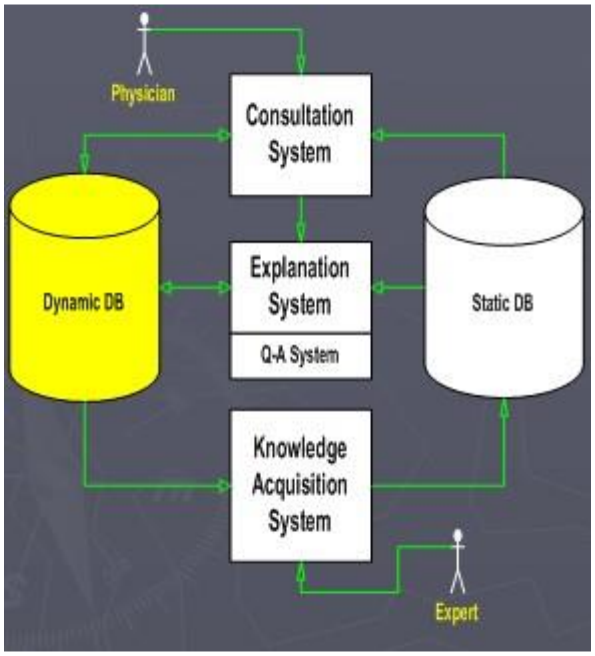
- Alternative to exhaustive invocation of all rules
- Strategy rules to suggest an approach for a given sub-goal
    - Ordering rules to try first, effectively pruning the search tree
- Creates a search-space with embedded information on which branch is best to take
- High-order Meta-Rules (i.e. Meta-Rules for Meta-Rules)
    - Powerful, but used limitedly in practice
- Impact to the Explanation System:
    - (+) Encode Knowledge formerly in the Control Structure
    - (-) Sometimes create "murky" explanations

Templates

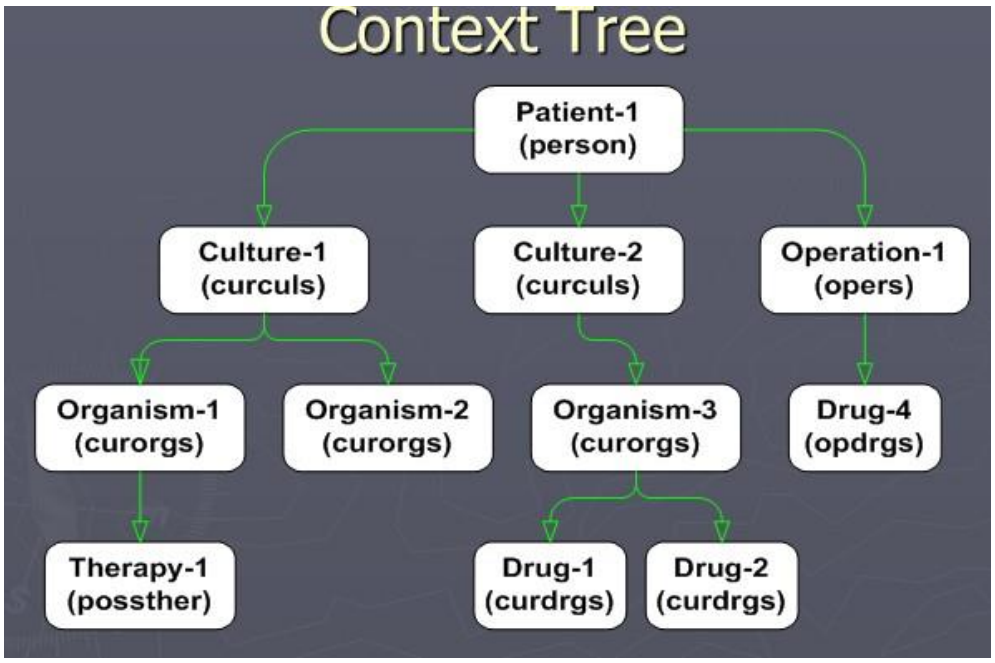
- ► The Production Rules are all based on Template structures
- ► This aids Knowledge-base expansion, because the system can “understand” its own representations
- ► Templates are updated by the system when a new rule is entered

Dynamic Database

- ► Patient Data
- ► Laboratory Data
- ► Context Tree
- ► Built by Consultation System
- ► Used by Explanation System



Context Tree



Therapy Selection

- Plan-Generate-and-Test Process
- Therapy List Creation
  - Set of specific rules recommend treatments based on the probability (not CF) of organism sensitivity
  - Probabilities based on laboratory data
  - One therapy rule for every organism
- Assigning Item Numbers
  - Only hypothesis with organisms deemed "significantly likely" (CF) are considered
  - Then the most likely (CF) identity of the organisms themselves are determined and assigned an Item Number
  - Each item is assigned a probability of likelihood and probability of sensitivity to drug
- Final Selection based on:

- Sensitivity
- Contraindication Screening
- Using the minimal number of drugs and maximizing the coverage of organisms

▶ Experts can ask for alternate treatments

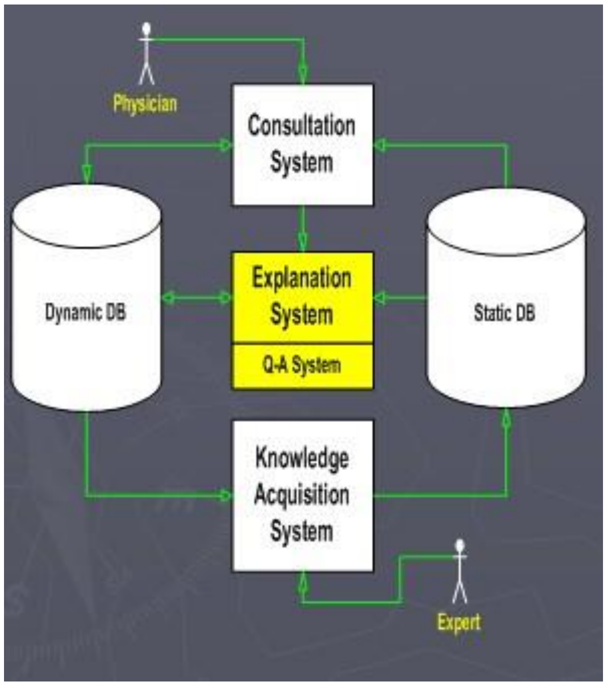
- Therapy selection is repeated with previously recommended drugs removed from the list

Explanation System

▶ Provides reasoning why a conclusion has been made, or why a question is being asked

▶ Q-A Module

▶ Reasoning Status Checker



▶ Uses a trace of the Production Rules for a basis, and the Context Tree, to provide context

- Ignores Definitional Rules (CF == 1)

- Two Modules
    - Q-A Module
    - Reasoning Status Checker

Q-A Module

- Symbolic Production Rules are readable
- Each <predicate function> has an associated translation pattern:

GRID (THE (2) ASSOCIATED WITH (1) IS KNOWN)

VAL (((2 1)))

PORTAL (THE PORTAL OF ENTRY OF *)

PATH-FLORA (LIST OF LIKELY PATHOGENS)

i.e. (GRID (VAL CNTXT PORTAL) PATH-FLORA) becomes:

"The list of likely pathogens associated with the portal of entry of the organism is known."

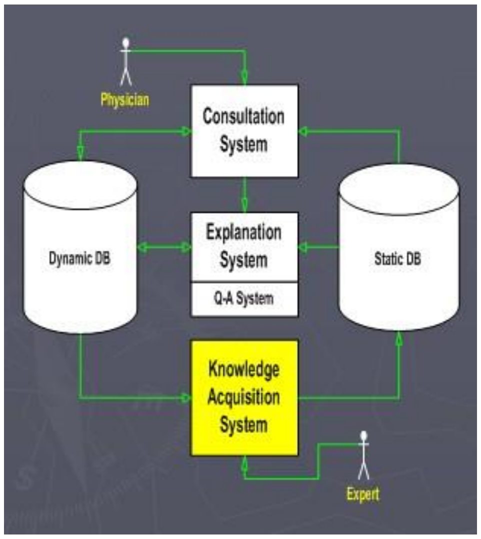
Reasoning Status Checker

- Explanation is a tree traversal of the traced rules:
    - WHY – moves up the tree
    - HOW – moves down (possibly to untried areas)
- Question is rephrased, and the rule being applied is explained with the translation patterns

Knowledge Acquisition System

- Extends Static DB via Dialogue with Experts
- Dialogue Driven by System
- Requires minimal training for Experts

► Allows for Incremental Competence, NOT an All-or-Nothing model



► IF-THEN Symbolic logic was found to be easy for experts to learn, and required little training by the MYCIN team

► When faced with a rule, the expert must either except it or be forced to update it using the education process

Education Process

1. Bug is uncovered, usually by Explanation process
2. Add/Modify rules using *subset of English* by experts
3. Integrating new knowledge into KB
    - Found to be difficult in practice, requires detection of contradictions, and complex concepts become difficult to express

Results

► Never implemented for routine clinical use

► Shown to be competent by panels of experts, even in cases where experts themselves disagreed on conclusions

► Key Contributions:

- Reuse of Production Rules (explanation, knowledge acquisition models)
- Meta-Level Knowledge Use

5.6.2 DART

The Dynamic Analysis and Replanning Tool, commonly abbreviated to DART, is an artificial intelligence program used by the U.S. military to optimize and schedule the transportation of supplies or personnel and solve other logistical problems.

DART uses intelligent agents to aid decision support systems located at the U.S. Transportation and European Commands. It integrates a set of intelligent data processing agents and database management systems to give planners the ability to rapidly evaluate plans for logistical feasibility. By automating evaluation of these processes DART decreases the cost and time required to implement decisions.

DART achieved logistical solutions that surprised many military planners. Introduced in 1991, DART had by 1995 offset the monetary equivalent of all funds DARPA had channeled into AI research for the previous 30 years combined

Development and introduction

DARPA funded the MITRE Corporation and Carnegie Mellon University to analyze the feasibility of several intelligent planning systems. In November 1989, a demonstration named The Proud Eagle Exercise indicated many inadequacies and bottlenecks within military support systems. In July, DART was previewed to the military by BBN Systems and Technologies and the ISX Corporation (now part of Lockheed Martin Advanced Technology Laboratories) in conjunction with the United States Air Force Rome Laboratory. It was proposed in November 1990, with the military immediately demanding that a prototype be developed for testing. Eight weeks later, a hasty but working prototype was introduced in 1991 to the USTRANSCOM at the beginning of Operation Desert Storm during the Gulf War.

Impact

Directly following its launch, DART solved several logistical nightmares, saving the military millions of dollars. Military planners were aware of the tremendous obstacles facing moving military assets from bases in Europe to prepared bases in Saudi Arabia, in preparation for Desert Storm. DART quickly proved its value by improving upon existing plans of the U.S. military. What surprised many observers were DART's ability to adapt plans rapidly in a crisis environment.

DART's success led to the development of other military planning agents such as:

- RDA - Resource Description and Access system
- DRPI - Knowledge-Based Planning and Scheduling Initiative, a successor of DART

5.7 Expert Systems shells

Initially each expert system is build from scratch (LISP). Systems are constructed as a set of declarative representations (mostly rules) combined with an interpreter for those representations. It helps to separate the interpreter from domain-specific knowledge and to create a system that could be used construct new expert system by adding new knowledge corresponding to the new problem domain. The resulting interpreters are called shells. Example of shells is EMYCIN (for Empty MYCIN derived from MYCIN).

Shells − A shell is nothing but an expert system without knowledge base. A shell provides the developers with knowledge acquisition, inference engine, user interface, and explanation facility. For example, few shells are given below −

- Java Expert System Shell (JESS) that provides fully developed Java API for creating an expert system.
- Vidwan, a shell developed at the National Centre for Software Technology, Mumbai in 1993. It enables knowledge encoding in the form of IF-THEN rules.

Shells provide greater flexibility in representing knowledge and in reasoning than MYCIN. They support rules, frames, truth maintenance systems and a variety of other reasoning mechanisms.

Early expert system shells provide mechanisms for knowledge representation, reasoning and explanation. Later these tools provide knowledge acquisition. Still expert system shells need to integrate with other programs easily. Expert systems cannot operate in a vacuum. The shells must provide an easy-to-use interface between an expert system written with the shell and programming environment.

PART-A

1. What are the phases in Expert system development?

2. Explain identification phase.

3. Explain conceptualization phase.

4. Explain formalization phase.

5. Explain implementation and testing phases.

6. Describe demonstration prototype.

7. Describe research prototype.

8. Describe field prototype.

9. What is meant by production prototype?

10. What is meant by commercial system?

11. When is Expert System Development Possible?

12. When is Expert System Development justified?

13. When is Expert System Development appropriate?

14. What are the questions to ask when selecting an Expert system tool?

15. What are the limitations of expert systems?

16. What are the problems the company faces when trying to apply expert system?

17. What are the common pitfalls in planning an expert system?

18. What are the pitfalls in dealing with the domain expert?

19. What are the pitfalls during the development process?

20. Where is expert system work being done?

21. What is meant by a commercial expert system?

22. Name any two expert system used in research?

23. Name any two expert system used in business?

24. Explain XCON?

25. Name any three universities and mention the expert system tools developed there?

26. Name any three research organization and mention the expert system tools developed there?

27. What are expert systems?

28. What are the most important aspects of expert system?

29. What are the characteristics of expert system?

30. Sketch the general features of expert system?

31. Who are all involved in the expert system building?

32. Explain the role of domain expert?

33. Explain the role of knowledge engineer?

34. What is the use of expert system building tool?

35. Give the structure of an expert system?

36. Explain the knowledge acquisition process?

37. Define MYCIN.

38. Define DART.

PART-B

1. Explain the tasks involved in building expert system?

2. Explain the various stages of expert system development?

3. Explain the difficulties involved in developing an expert system?

4. What are common pitfalls in planning an expert system?

5. Explain the pitfalls in dealing with the domain expert?

6. Explain the pitfalls during the development process?

7. Explain expert system work at universities and research organizations?

8. Explain expert system work at knowledge engineering companies?

9. What is meant by high performance expert system? How is it used in research and in business?

10. In detail Explain XCON?

11. Draw the schematic of expert system and explain.

12. Explain the following in detail.. a. MYCIN   b. EMYCIN

MADHA
Expertise | Empathy | Excellence
ENGINEERING COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COMMON FOR: DEPARTMENT OF INFORMATION TECHNOLOGY

CS8651 – INTERNET PROGRAMMING

R – 2017

LECTURE NOTES

CS8651 INTERNET PROGRAMMING
UNIT I WEBSITE BASICS, HTML 5, CSS 3, WEB 2.0 9
Web Essentials: Clients, Servers and Communication – The Internet – Basic Internet protocols – World wide web – HTTP Request Message – HTTP Response Message – Web Clients – Web Servers – HTML5 – Tables – Lists – Image – HTML5 control elements – Semantic elements – Drag and Drop – Audio – Video controls - CSS3 – Inline, embedded and external style sheets – Rule cascading – Inheritance – Backgrounds – Border Images – Colors – Shadows – Text – Transformations – Transitions – Animations.

WEB ESSENTIALS

1.1 Web Essentials:
Server:
The software that distributes the information and the machine where the information and software reside is called the server.
• provides requested service to client
• e.g., Web server sends requested Web page
Client:
The software that resides on the remote machine, communicates with the server, fetches the information, processes it, and then displays it on the remote machine is called the client.
• initiates contact with server (―speaks first‖)
• typically requests service from server
• Web: client implemented in browser
Web server:
Software that delivers Web pages and other documents to browsers using the HTTP protocol
Web Page:
A web page is a document or resource of information that is suitable for the World Wide Web and can be accessed through a web browser.
Website:
A collection of pages on the World Wide Web that are accessible from the same URL and typically residing on the same server.
URL:
Uniform Resource Locator, the unique address which identifies a resource on the Internet for routing purposes.
1.2 Client-server paradigm:
IThe Client-Server paradigm is the most prevalent model for distributed computing protocols. It is the basis of all distributed computing paradigms at a higher level of abstraction. It is service-oriented, and employs a request-response protocol.
A server process, running on a server host, provides access to a service. A client process, running on a client host, accesses the service via the server process.The interaction of the process proceeds according to a protocol.
The primary idea of a client/server system is that you have a central repository of information—some kind of data, often in a database—that you want to distribute on demand to some set of people or machines.
1.3 The Internet:
• Medium for communication and interaction in inter connected network.

• Makes information constantly and instantly available to anyone with a connection.

Web Browsers:

• User agent for Web is called a browser:

o Internet Explorer

o Firefox

Web Server:

• Server for Web is called Web server:

o Apache (public domain)

o MS Internet Information Server

Protocol:

Protocols are agreed formats for transmitting data between devices.

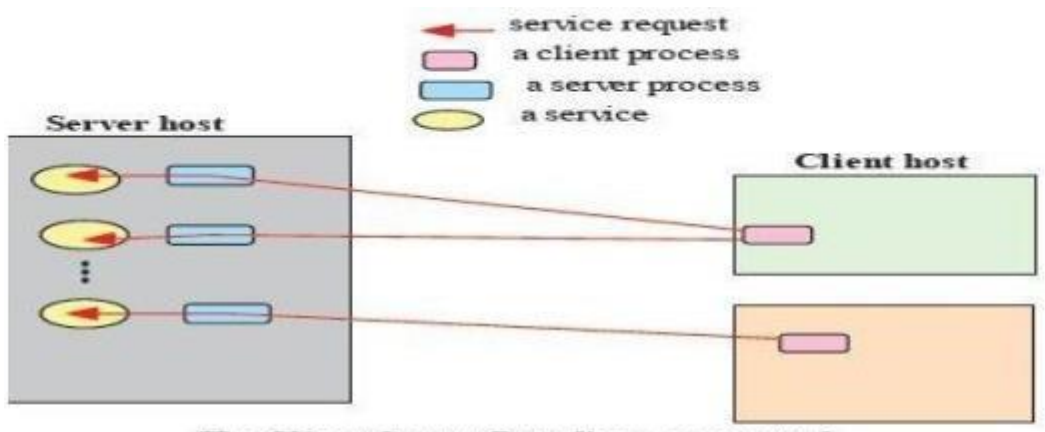
The protocol determines:

i. The error checking required

ii. Data compression method used

iii. The way the end of a message is signalled

iv. The way the device indicates that it has received the message



The Client-Server Paradigm, conceptual

1.4 Internet Protocol:

There are many protocols used by the Internet and the WWW:

o TCP/IP

o HTTP

o FTP

o Electronic mail protocols IMAP

o POP

TCP/IP

The Internet uses two main protocols (developed by Vincent Cerf and Robert Kahn)

Transmission control protocol (TCP):Controls disassembly of message into packets at the origin reassembles at the destination

Internet protocol (IP):Specifies the addressing details for each packet Each packet is labelled with its origin and destination.

1.5 Hypertext Transfer Protocol (HTTP)

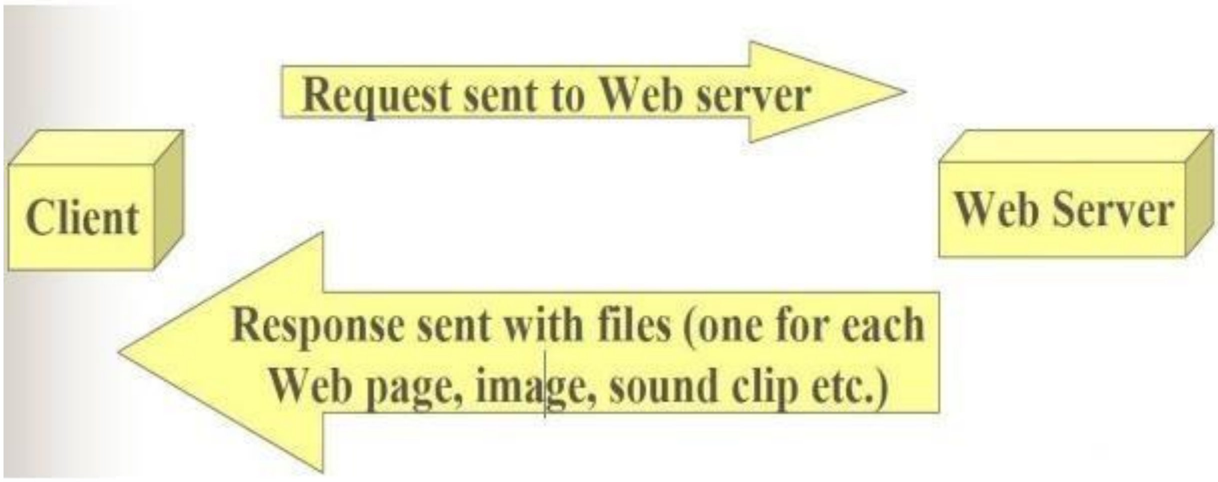
• The hypertext transfer protocol (HTTP) was developed by Tim Berners-Lee in 1991

• HTTP was designed to transfer pages between machines

• The client (or Web browser) makes a request for a given page and the Server is responsible for finding it and returning it to the client

• The browser connects and requests a page from the server

• The server reads the page from the file system, sends it to the client and terminates the connection.


Request sent to Web server
Client
Web Server
Response sent with files (one for each Web page, image, sound clip etc.)

Electronic Mail Protocols:
• Electronic mail uses the client/server model
• The organisation has an email server devoted to handling email
o Stores and forwards email messages
• Individuals use email client software to read and send email
o (e.g. Microsoft Outlook, or Netscape Messenger)
• Simple Mail Transfer Protocol (SMTP)

o Specifies format of mail messages
• Post Office Protocol (POP) tells the email server to:
o Send mail to the user’s computer and delete it from the server
o Send mail to the user’s computer and do not delete it from the server
o Ask whether new mail has arrived.

1.6 Interactive Mail Access Protocol (IMAP)
Newer than POP, provides similar functions with additional features.
o e.g. can send specific messages to the client rather than all the messages.
A user can view email message headers and the sender’s name before downloading the entire message.
Allows users to delete and search mailboxes held on the email server.
The disadvantage of POP: You can only access messages from one PC.
The disadvantage of IMAP : Since email is stored on the email server, there is a need for more and more expensive (high speed) storage space.
1.7 World Wide Web: comprises software (Web server and browser) and data (Web sites).
Internet Protocol (IP) Addresses:
- Every node has a unique numeric address
- Form: 32-bit binary number
- New standard, IPv6, has 128 bits (1998)
- Organizations are assigned groups of IPs for their computers
- **Domain names**
- Form: host-name. domain-names
- First domain is the smallest (Google)
- Last domain specifies the type of organization (.com)
- Fully qualified domain name - the host name and all of the domain names
- DNS servers - convert fully qualified domain names to IPs

1.8 HTTP:
●Hypertext Transfer Protocol (HTTP) is the communication protocol used by the Internet to transfer hypertext documents.

- A protocol to transfer hypertext requests and information between servers and browsers

•Hypertext is text, displayed on a computer, with references (hyperlinks) to other text that the reader can immediately follow, usually by a mouse HTTP is behind every request for a web document or graph, every click of a hypertext link, and every submission of a form.

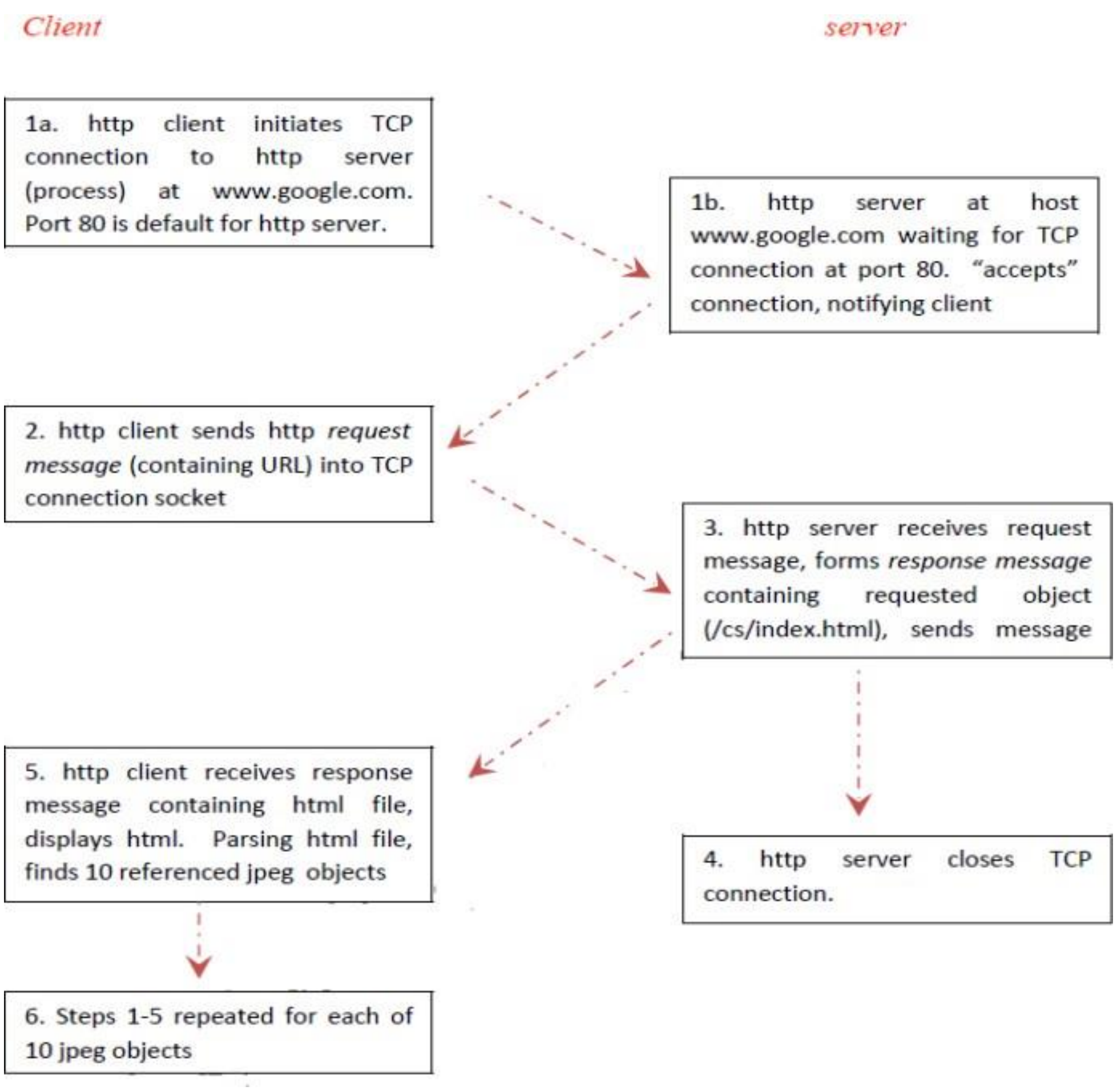
•HTTP specifies how clients **request** data, and how servers **respond** to these requests.

•The client makes a request for a given page and the server is responsible for finding it and returning it to the client.

•The browser connects and requests a page from the server.

•The server reads the page from the file system and sends it to the client and then terminates the connection

HTTP Transactions


Client
server
1a. http client initiates TCP connection to http server (process) at www.google.com. Port 80 is default for http server.
1b. http server at host www.google.com waiting for TCP connection at port 80. "accepts" connection, notifying client
2. http client sends http request message (containing URL) into TCP connection socket
3. http server receives request message, forms response message containing requested object (/cs/index.html), sends message
4. http server closes TCP connection.
5. http client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects
6. Steps 1-5 repeated for each of 10 jpeg objects

1.9 HTTP Message:

HTTP message is the information transaction between the client and server.

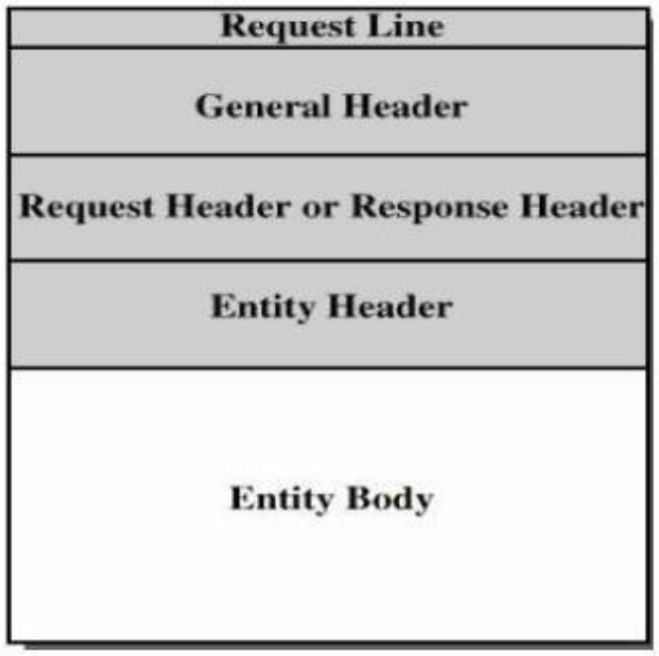
Two types of HTTP Message:

1. Requests

a. Client to server

2. Responses

a. Server to client



Fields

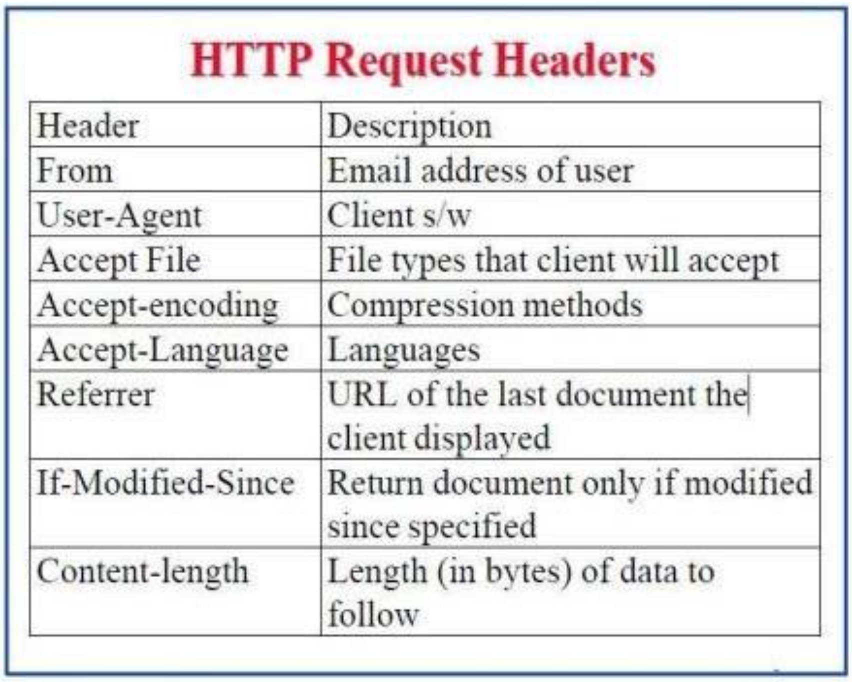
· Request line or Response line
· General header
· Request header or Response header
· Entity header
· Entity body

.10 Request Message:

Request Line:

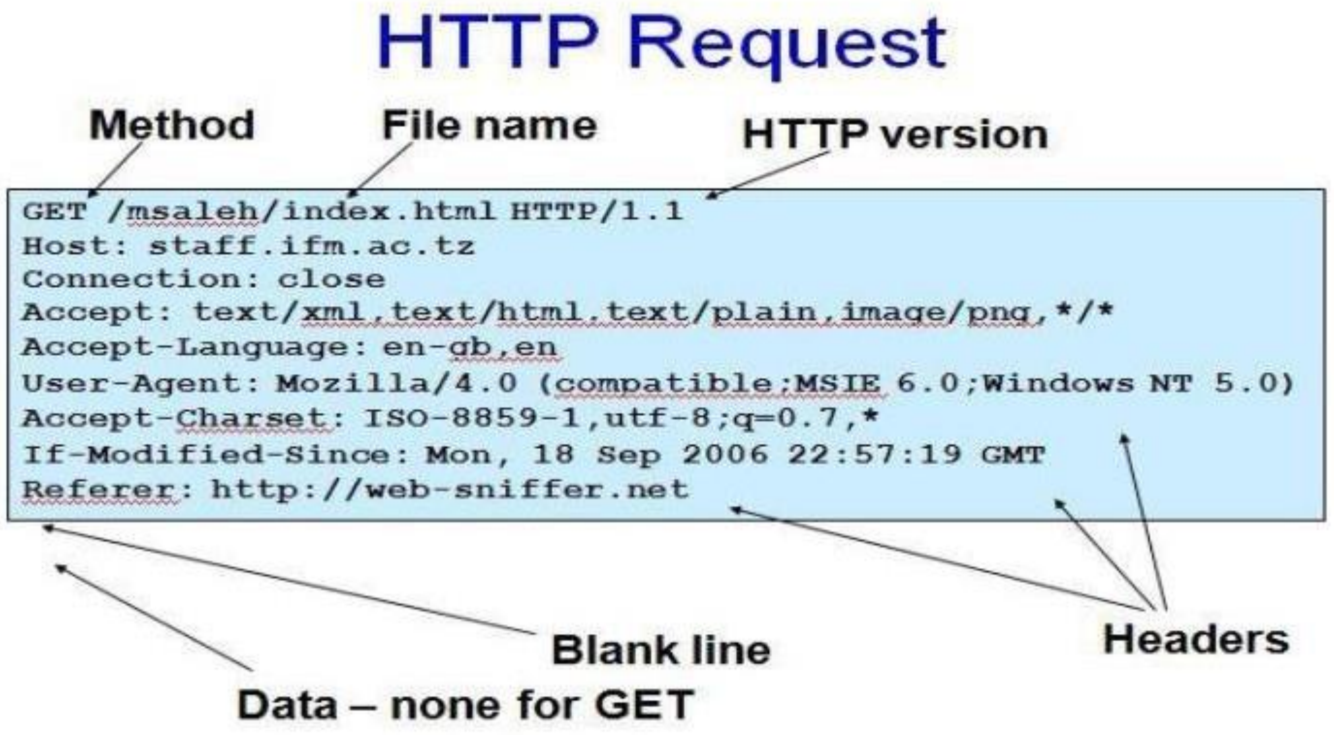
• A request line has three parts, separated by spaces
o a *method* name
o the local path of the requested resource
o the version of HTTP being used
• A typical request line is:
o GET /path/to/file/index.html HTTP/1.1
• Notes:
o **GET** is the most common HTTP method; it says "give me this resource". Other methods include **POST** and **HEAD.** Method names are always uppercase
o The path is the part of the URL after the host name, also called the *request URI*
o The HTTP version always takes the form "**HTTP/x.x**", uppercase.

Request Header:



HTTP Request Headers

Header	Description
From	Email address of user
User-Agent	Client s/w
Accept File	File types that client will accept
Accept-encoding	Compression methods
Accept-Language	Languages
Referrer	URL of the last document the client displayed
If-Modified-Since	Return document only if modified since specified
Content-length	Length (in bytes) of data to follow



```
GET /msaleh/index.html HTTP/1.1
Host: staff.ifm.ac.tz
Connection: close
Accept: text/xml,text/html,text/plain,image/png,*/*
Accept-Language: en-gb,en
User-Agent: Mozilla/4.0 (compatible;MSIE 6.0;Windows NT 5.0)
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*
If-Modified-Since: Mon, 18 Sep 2006 22:57:19 GMT
Referer: http://web-sniffer.net
```

.11 Response Message:

Response Line:

• A request line has three parts, separated by spaces

o the HTTP version,

o a *response status code* that gives the result of the request, and

o an English *reason phrase* describing the status code

• Typical status lines are:

o HTTP/1.0 200 OK or

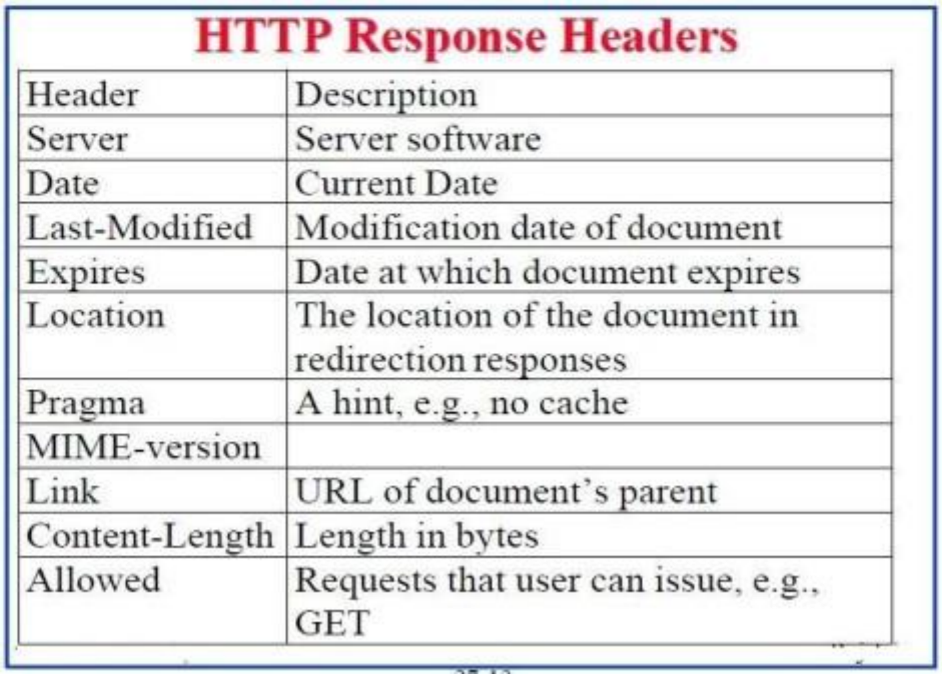
o HTTP/1.0 404 Not Found

• Notes:

o The HTTP version is in the same format as in the request line, "**HTTP/x.x**".

o The status code is meant to be computer-readable; the reason phrase is meant to be human-readable, and may vary.

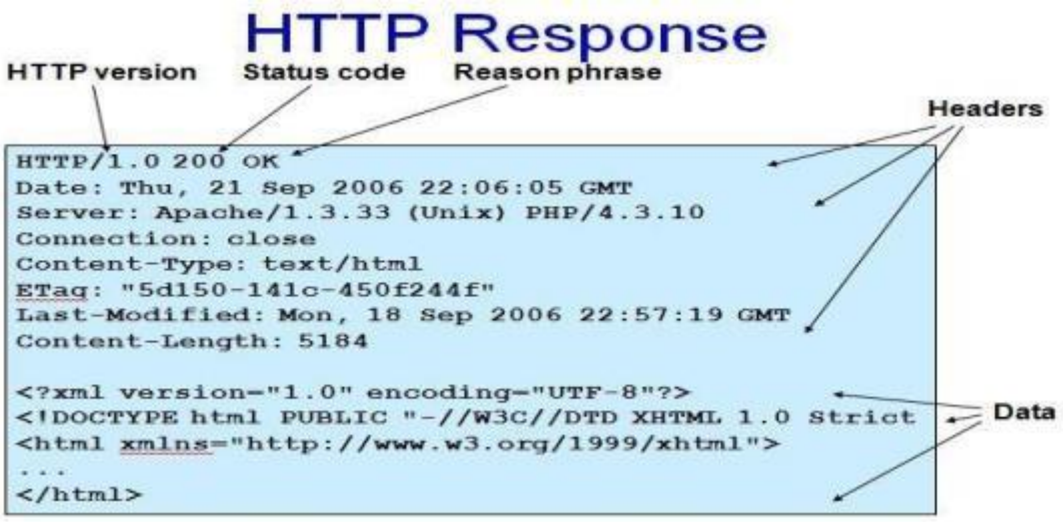
HTTP Request Header:



HTTP Response Headers

Header	Description
Server	Server software
Date	Current Date
Last-Modified	Modification date of document
Expires	Date at which document expires
Location	The location of the document in redirection responses
Pragma	A hint, e.g., no cache
MIME-version	
Link	URL of document's parent
Content-Length	Length in bytes
Allowed	Requests that user can issue, e.g., GET

EXAMPLE



```
HTTP/1.0 200 OK
Date: Thu, 21 Sep 2006 22:06:05 GMT
Server: Apache/1.3.33 (Unix) PHP/4.3.10
Connection: close
Content-Type: text/html
ETag: "5d150-141c-450f244f"
Last-Modified: Mon, 18 Sep 2006 22:57:19 GMT
Content-Length: 5184

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict
<html xmlns="http://www.w3.org/1999/xhtml">
...
</html>
```

HTTP Method:

• HTTP method is supplied in the request line and specifies the operation that the client has requested.

Some common methods:

• Options
• Get
• Head
• Post
• Put
• Move
• Delete

Two methods that are mostly used are the GET and POST:

o **GET** for queries that can be safely repeated o **POST** for operations that may have side effects (e.g. ordering a book from an on-line store).

The GET Method

• It is used to retrieve information from a specified URI and is assumed to be a safe, repeatable operation by browsers, caches and other HTTP aware components

• Operations have no side effects and GET requests can be re-issued.

• For example, displaying the balance of a bank account has no effect on the account and can be safely repeated.

• Most browsers will allow a user to refresh a page that resulted from a **GET**, without displaying any kind of warning

• Proxies may automatically retry **GET** requests if they encounter a temporary network connection problem.

• GET requests is that they can only supply data in the form of parameters encoded in the URI (known as a **Query String**) – [downside]

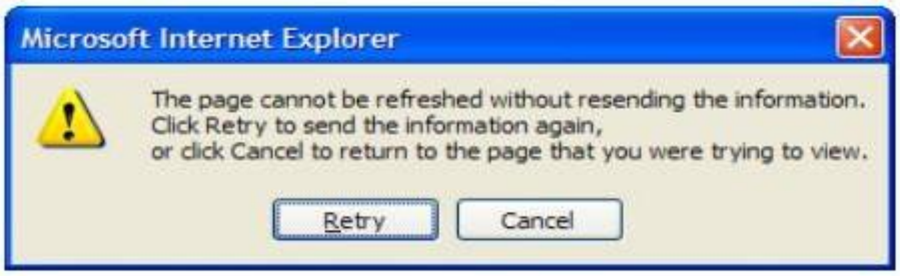
• Cannot be unused for uploading files or other operations that require large amounts of data to be sent to the server.

The POST Method

• Used for operations that have side effects and cannot be safely repeated.

• For example, transferring money from one bank account to another has side effects and should not be repeated without explicit approval by the user.

• If you try to refresh a page in Internet Explorer that resulted from a **POST**, it displays the following message to warn you that there may



The page cannot be refreshed without resending the information.
Click Retry to send the information again,
or click Cancel to return to the page that you were trying to view.

The **POST** request message has a content body that is normally used to send parameters and data

• The IIS server returns two status codes in its response for a **POST** request

o The first is **100 Continue** to indicate that it has successfully received the **POST** request

o The second is **200 OK** after the request has been processed.

HTTP response status codes

• Informational (1xx)

• Successful (2xx)

• Redirection (3xx)

o 301: moved permanently

• Client error (4xx)

o 403 : forbidden

o 404: Not found

• Server error (5xx)

o 503: Service unavailable

o 505: HTTP version not supported

1.12 HTTP

❖ **Features**

• Persistent TCP Connections: Remain open for multiple requests

• Partial Document Transfers: Clients can specify start and stop positions

• Conditional Fetch: Several additional conditions

• Better content negotiation
• More flexible authentication.
❖ **Web Browsers :**
• Mosaic - NCSA (Univ. of Illinois), in early 1993 First to use a GUI, led to Explosion of Web use Initially for X-Windows, under UNIX, but was ported to other platforms by late 1993
• Browsers are clients - always initiate, servers react (although sometimes servers require responses)
• Most requests are for existing documents, using Hypertext Transfer Protocol
• (HTTP) But some requests are for program execution, with the output being returned as a document.
Browser: A web browser is a software application for retrieving, presenting, and traversing information resources on the World Wide Web.
❖ **Web Servers:**
- Provide responses to browser requests, either existing documents or dynamicallyBuilt documents.
- Browser-server connection is now maintained through more than one request- Response cycle
- All communications between browsers and servers use Hypertext Transfer Protocol
- Web servers run as background processes in the operating system.
- Monitor a communications port on the host, accepting HTTP messages when they appear
All current Web servers came from either
1. The original from CERN
2. The second one, from NCSA
- Web servers have two main directories:
1. Document root (servable documents)
2. Server root (server system software)
- Document root is accessed indirectly by clients
- Its actual location is set by the server Configuration file
- Requests are mapped to the actual location
- Virtual document trees
- Virtual hosts
- Proxy servers
- Web servers now support other Internet protocols

- Apache (open source, fast, reliable)
- IIS
- Maintained through a program with a GUI interface.

HTML 5

HTML is the main markup language for describing the structure of web pages.

HTML stands for HyperText Markup Language. HTML is the basic building block of World Wide Web.

Hypertext is text displayed on a computer or other electronic device with references to other text that the user can immediately access, usually by a mouse click or key press.

Apart from text, hypertext may contain tables, lists, forms, images, and other presentational elements. It is an easy-to-use and flexible format to share information over the Internet.

Markup languages use sets of markup tags to characterize text elements within a document, which gives instructions to the web browsers on how the document should appear.

HTML was originally developed by Tim Berners-Lee in 1990. He is also known as the father of the web. In 1996, the World Wide Web Consortium (W3C) became the authority to maintain the HTML specifications. HTML also became an international standard (ISO) in 2000. HTML5 is the latest version of HTML. HTML5 provides a faster and more robust approach to web development.

HTML Tags and Elements

HTML is written in the form of HTML elements consisting of markup tags. These markup tags are the fundamental characteristic of HTML. Every markup tag is composed of a keyword, surrounded by angle brackets, such as `<html>`, `<head>`, `<body>`, `<title>`, `<p>`, and so on.

HTML tags normally come in pairs like `<html>` and `</html>`. The first tag in a pair is often called the opening tag (or start tag), and the second tag is called the closing tag (or end tag).

An opening tag and a closing tag are identical, except a slash (`/`) after the opening angle bracket of the closing tag, to tell the browser that the command has been completed.

Inserting Images into Web Pages

Images enhance visual appearance of the web pages by making them more interesting and colorful.

The `<img>` tag is used to insert images in the HTML documents. It is an empty element and contains attributes only. The syntax of the `<img>` tag can be given with:

```
<img src="url" alt="some_text">
```

The following example inserts three images on the web page:

Example
Try this code »

```
<img src="kites.jpg" alt="Flying Kites">
<img src="sky.jpg" alt="Cloudy Sky">
<img src="balloons.jpg" alt="Balloons">
```

Each image must carry at least two attributes: the `src` attribute, and an `alt` attribute.

The `src` attribute tells the browser where to find the image. Its value is the URL of the image file.

Whereas, the `alt` attribute provides an alternative text for the image, if it is unavailable or cannot be displayed for some reason. Its value should be a meaningful substitute for the image.

HTML Tables

Creating Tables in HTML

HTML table allows you to arrange data into rows and columns. They are commonly used to display tabular data like product listings, customer's details, financial reports, and so on.

You can create a table using the `<table>` element. Inside the `<table>` element, you can use the `<tr>` elements to create rows, and to create columns inside a row you can use the `<td>` elements. You can also define a cell as a header for a group of table cells using the `<th>` element.

The following example demonstrates the most basic structure of a table.

Example

```
<table>
    <tr>
        <th>No.</th>
        <th>Name</th>
        <th>Age</th>
    </tr>
    <tr>
        <td>1</td>
        <td>Peter Parker</td>
        <td>16</td>
    </tr>
    <tr>
        <td>2</td>
        <td>Clark Kent</td>
        <td>34</td>
    </tr>
</table>
```

Tables do not have any borders by default. You can use the CSS border property to add borders to the tables. Also, table cells are sized just large enough to fit the contents by default. To add more space around the content in the table cells you can use the CSS padding property.

Defining a Table Header, Body, and Footer

HTML provides a series of tags <thead>, <tbody>, and <tfoot> that helps you to create more structured table, by defining header, body and footer regions, respectively.

The following example demonstrates the use of these elements.

Example

```
<table>
    <thead>
        <tr>
            <th>Items</th>
            <th>Expenditure</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Stationary</td>
            <td>2,000</td>
        </tr>
        <tr>
            <td>Furniture</td>
            <td>10,000</td>
        </tr>
    </tbody>
    <tfoot>
        <tr>
```

```
<th>Total</th>
<td>12,000</td>
```

```
        </tr>
    </tfoot>
</table>
```

HTML Lists

HTML lists are used to present list of information in well formed and semantic way. There are three different types of list in HTML and each one has a specific purpose and meaning.

- **Unordered list** — Used to create a list of related items, in no particular order.
- **Ordered list** — Used to create a list of related items, in a specific order.
- **Description list** — Used to create a list of terms and their descriptions.

HTML Unordered Lists

An unordered list created using the `<ul>` element, and each list item starts with the `<li>` element.

The list items in unordered lists are marked with bullets. Here's an example:

Example

```
<ul>
    <li>Chocolate Cake</li>
    <li>Black Forest Cake</li>
    <li>Pineapple Cake</li>
</ul>
```

— The output of the above example will look something like this:

- Chocolate Cake
- Black Forest Cake
- Pineapple Cake

You can also change the bullet type in your unordered list using the CSS list-style-type property. The following style rule changes the type of bullet from the default *disc* to *square*:

Example

```
ul {
    list-style-type: square;
}
```

Please check out the tutorial on CSS lists to learn about styling HTML lists in details.

HTML Ordered Lists

An ordered list created using the `<ol>` element, and each list item starts with the `<li>` element. Ordered lists are used when the order of the list's items is important.

The list items in an ordered list are marked with numbers. Here's an example:

Example

```
<ol>
    <li>Fasten your seatbelt</li>
    <li>Starts the car's engine</li>
    <li>Look around and go</li>
</ol>
```
— The output of the above example will look something like this:

1. Fasten your seatbelt
2. Starts the car's engine
3. Look around and go

HTML5 Image

HTML Images Syntax

In HTML, images are defined with the `<img>` tag.

The `<img>` tag is empty, it contains attributes only, and does not have a closing tag.

The `src` attribute specifies the URL (web address) of the image:

```
<img src="url">
```

EXAMPLE

```
<!DOCTYPE html>

<html>

<body>



<h2>HTML Image</h2>

<img src="html5.gif" alt="HTML5 Icon" style="width:128px;height:128px;">



</body>

</body>

</html>
```

OUTPUT

HTML Image

HTML Form

HTML Forms are required to collect different kinds of user inputs, such as contact details like name, email address, phone numbers, or details like credit card information, etc.

Forms contain special elements called **controls** like ***inputbox, checkboxes, radio-buttons, submit buttons***, etc. Users generally complete a form by modifying its controls e.g. entering text, selecting items, etc. and submitting this form to a web server for further processing.

The <form> tag is used to create an HTML form. Here's a simple example of a login form:

Example

```
<form>
    <label>Username: <input type="text"></label>
    <label>Password: <input type="password"></label>
    <input type="submit" value="Submit">
</form>
```

The following section describes different types of controls that you can use in your form.

Input Element

This is the most commonly used element within HTML forms.

It allows you to specify various types of user input fields, depending on the type attribute. An input element can be of type *text field*, *password field*, *checkbox*, *radio button*, *submit button*, *reset button*, *file select box*, as well as several new input types introduced in HTML5.

The most frequently used input types are described below.

Text Fields

Text fields are one line areas that allow the user to input text.

Single-line text input controls are created using an <input> element, whose type attribute has a value of text. Here's an example of a single-line text input used to take username:

Example

```
<form>
    <label for="username">Username:</label>
    <input type="text" name="username" id="username">
</form>
```

— The output of the above example will look something like this:

Username:

Password Field

Password fields are similar to text fields. The only difference is; characters in a password field are masked, i.e. they are shown as asterisks or dots. This is to prevent someone else from reading the password on the screen. This is also a single-line text input controls created using an `<input>` element whose `type` attribute has a value of `password`.

Example

```
<form>
    <label for="user-pwd">Password:</label>
    <input type="password" name="user-password" id="user-pwd">
</form>
```

— The output of the above example will look something like this:

Password:

Radio Buttons

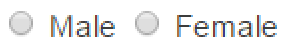
Radio buttons are used to let the user select exactly one option from a pre-defined set of options. It is created using an `<input>` element whose `type` attribute has a value of `radio`.

Example

Try this code »

```
<form>
    <input type="radio" name="gender" id="male">
    <label for="male">Male</label>
    <input type="radio" name="gender" id="female">
    <label for="female">Female</label>
</form>
```

— The output of the above example will look something like this:



Checkboxes

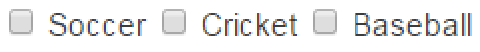
Checkboxes allows the user to select one or more option from a pre-defined set of options. It is created using an `<input>` element whose `type` attribute has a value of `checkbox`.

Example

```
<form>
    <input type="checkbox" name="sports" id="soccer">
    <label for="soccer">Soccer</label>
```

```
<input type="checkbox" name="sports" id="cricket">
```

```
    <label for="cricket">Cricket</label>
    <input type="checkbox" name="sports" id="baseball">
    <label for="baseball">Baseball</label>
</form>
```
— The output of the above example will look something like this:



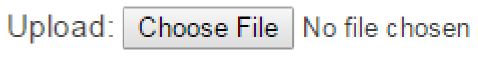
File Select box

The file fields allow a user to browse for a local file and send it as an attachment with the form data. Web browsers such as Google Chrome and Firefox render a file select input field with a Browse button that enables the user to navigate the local hard drive and select a file.

This is also created using an `<input>` element, whose `type` attribute value is set to `file`.

Example

```
<form>
    <label for="file-select">Upload:</label>
    <input type="file" name="upload" id="file-select">
</form>
```
— The output of the above example will look something like this:

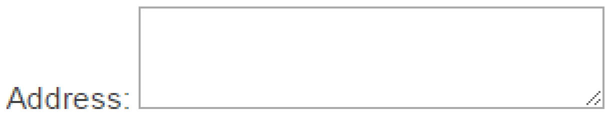


Textarea

Textarea is a multiple-line text input control that allows a user to enter more than one line of text. Multi-line text input controls are created using an `<textarea>` element.

Example

```
<form>
    <label for="address">Address:</label>
    <textarea rows="3" cols="30" name="address" id="address"></textarea>
</form>
```
— The output of the above example will look something like this:



Select Boxes

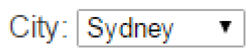
A select box is a dropdown list of options that allows user to select one or more option from a pull-down list of options. Select box is created using the <select> element and <option> element.

The <option> elements within the <select> element define each list item.

Example

```
<form>
    <label for="city">City:</label>
    <select name="city" id="city">
        <option value="sydney">Sydney</option>
        <option value="melbourne">Melbourne</option>
        <option value="cromwell">Cromwell</option>
    </select>
</form>
```

— The output of the above example will look something like this:



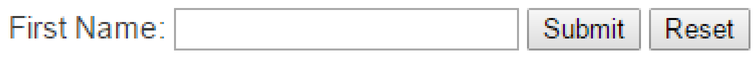
Submit and Reset Buttons

A submit button is used to send the form data to a web server. When submit button is clicked the form data is sent to the file specified in the form's action attribute to process the submitted data.

A reset button resets all the forms control to default values. Try out the following example by typing your name in the text field, and click on submit button to see it in action.

Example

```
<form action="action.php" method="post">
    <label for="first-name">First Name:</label>
    <input type="text" name="first-name" id="first-name">
    <input type="submit" value="Submit">
    <input type="reset" value="Reset">
</form>
```



<u>HTML5 Colors</u>

<!DOCTYPE html>

<html>

<body>

<h1 style="background-color:Tomato;">Tomato</h1>

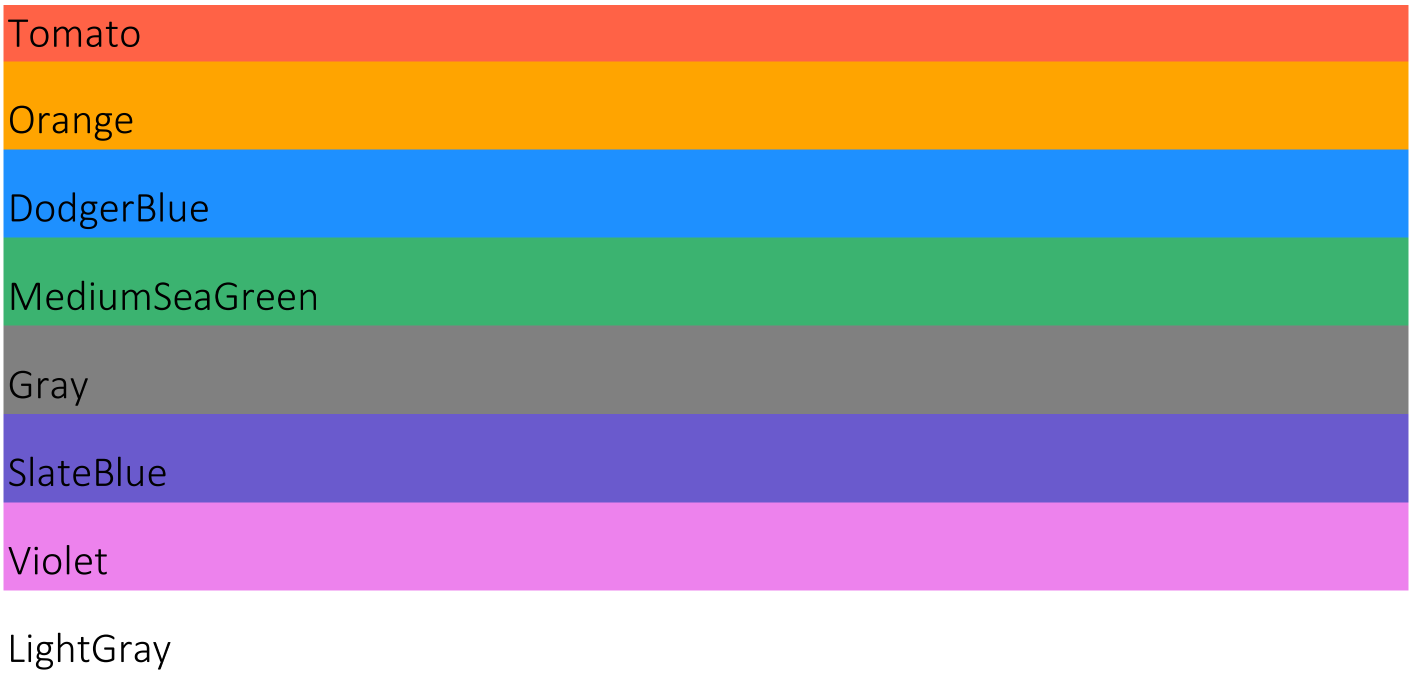
<h1 style="background-color:Orange;">Orange</h1>

<h1 style="background-color:DodgerBlue;">DodgerBlue</h1>

```
<h1 style="background-color:MediumSeaGreen;">MediumSeaGreen</h1>
```

```
<h1 style="background-color:Gray;">Gray</h1>

<h1 style="background-color:SlateBlue;">SlateBlue</h1>

<h1 style="background-color:Violet;">Violet</h1>

<h1 style="background-color:LightGray;">LightGray</h1>


</body>

</html>
```

OUTPUT



```
Tomato
Orange
DodgerBlue
MediumSeaGreen
Gray
SlateBlue
Violet
LightGray
```

HTML5 Audio

Embedding Audio in HTML Document

Inserting audio onto a web page was not easy before, because web browsers did not have a uniform standard for defining embedded media files like audio.

Using the HTML5 audio Element

The newly introduced HTML5 `<audio>` element provides a standard way to embed audio in web pages. However, the audio element is relatively new but it works in most of the modern web browsers.

The following example simply inserts an audio into the HTML5 document, using the browser default set of controls, with one source defined by the `src` attribute.

Example

```
<audio controls="controls" src="media/birds.mp3">
   Your browser does not support the HTML5 Audio element.
</audio>
```

An audio, using the browser default set of controls, with alternative sources.

Example

```
<audio controls="controls">
   <source src="media/birds.mp3" type="audio/mpeg">
   <source src="media/birds.ogg" type="audio/ogg">
   Your browser does not support the HTML5 Audio element.
</audio>
```

HTML5 Video

Embedding Video in HTML Document

Inserting video onto a web page was not relatively easy, because web browsers did not have a uniform standard for defining embedded media files like video.

Using the HTML5 video Element

The newly introduced HTML5 `<video>` element provides a standard way to embed video in web pages. However, the video element is relatively new, but it works in most of the modern web browsers.

The following example simply inserts a video into the HTML document, using the browser default set of controls, with one source defined by the `src` attribute.

Example

```
<video controls="controls" src="media/shuttle.mp4">
   Your browser does not support the HTML5 Video element.
</video>
```

A video, using the browser default set of controls, with alternative sources.

Example

```
<video controls="controls">
   <source src="media/shuttle.mp4" type="video/mp4">
   <source src="media/shuttle.ogv" type="video/ogg">
   Your browser does not support the HTML5 Video element.
</video>
```

New HTML5 Elements

The most interesting new HTML5 elements are:

New **semantic elements** like `<header>`, `<footer>`, `<article>`, and `<section>`.

New **attributes of form elements** like number, date, time, calendar, and range.

New **graphic elements**: `<svg>` and `<canvas>`.

New **multimedia elements**: `<audio>` and `<video>`.

What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of **non-semantic** elements: `<div>` and `<span>` - Tells nothing about its content.

Examples of **semantic** elements: `<form>`, `<table>`, and `<article>` - Clearly defines its content.

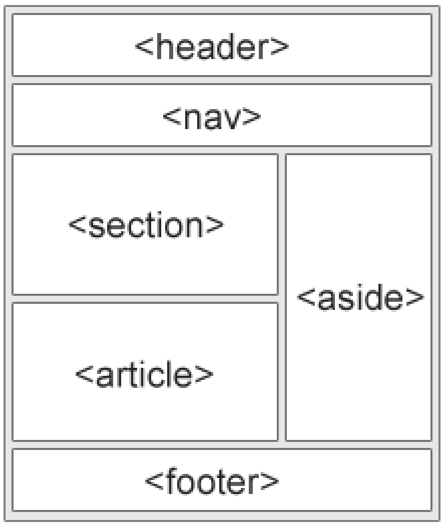
New Semantic Elements in HTML5

Many web sites contain HTML code like:

<div id="nav"> <div class="header"> <div id="footer">
to indicate navigation, header, and footer.

HTML5 offers new semantic elements to define different parts of a web page:

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>



HTML5 <section> Element

The <section> element defines a section in a document.

According to W3C's HTML5 documentation: "A section is a thematic grouping of content, typically with a heading."

A home page could normally be split into sections for introduction, content, and contact information.

Example

```
<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature (WWF) is ... </p>
</section>
```

HTML5 <article> Element

The <article> element specifies independent, self-contained content.

An article should make sense on its own, and it should be possible to read it independently from the rest of the web site.

Examples of where an <article> element can be used:

- Forum post
- Blog post
- Newspaper article

Example

```
<article>
  <h1>What Does WWF Do?</h1>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

HTML5 <header> Element

The <header> element specifies a header for a document or section.

The <header> element should be used as a container for introductory content.

You can have several <header> elements in one document.

The following example defines a header for an article:

Example

```
<article>
  <header>
    <h1>What Does WWF Do?</h1>
    <p>WWF's mission:</p>
  </header>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

HTML5 <footer> Element

The <footer> element specifies a footer for a document or section.

A <footer> element should contain information about its containing element.

A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc.

You may have several <footer> elements in one document.

Example

```
<footer>
  <p>Posted by: Hege Refsnes</p>
  <p>Contact information: <a href="mailto:someone@example.com">
  someone@example.com</a>.</p>
</footer>
```

HTML5 <figure> and <figcaption> Elements

The purpose of a figure caption is to add a visual explanation to an image.

In HTML5, an image and a caption can be grouped together in a <figure> element:

Example

```
<figure>
  <img src="pic_trulli.jpg" alt="Trulli">
  <figcaption>Fig1. - Trulli, Puglia, Italy.</figcaption>
</figure>
```

OUTPUT

Places to Visit

Puglia's most famous sight is the unique conical houses (Trulli) found in the area around Alberobello, a declared UNESCO World Heritage Site.

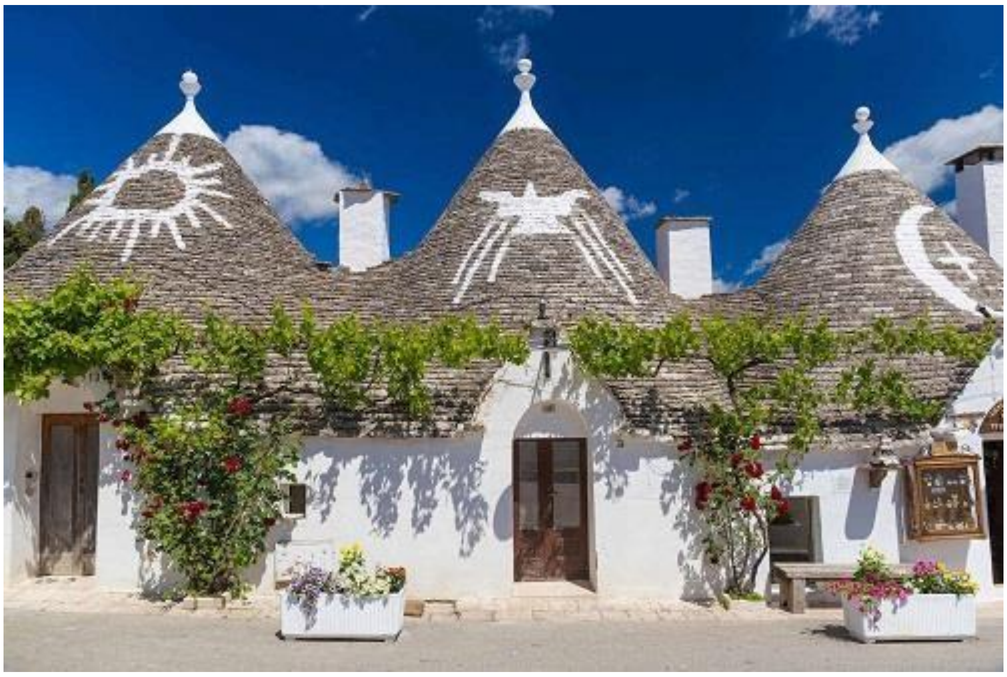


Fig.1 - Trulli, Puglia, Italy.

Semantic Elements in HTML5

Below is an alphabetical list of the new semantic elements in HTML5.

The links go to our complete HTML5 Reference.

Tag	Description
<article>	Defines an article
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide

<figcaption>	Defines a caption for a <figure> element

<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<main>	Specifies the main content of a document
<mark>	Defines marked/highlighted text
<nav>	Defines navigation links
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time

HTML5 Drag and Drop

Drag the W3Schools image into the rectangle.

Drag and Drop

Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location.

In HTML5, drag and drop is part of the standard: Any element can be draggable.

HTML Drag and Drop Example

The example below is a simple drag and drop example:

Example

```
<!DOCTYPE HTML>
<html>
<head>
<script>
function allowDrop(ev) {
  ev.preventDefault();
}

function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

<img id="drag1" src="img_logo.gif" draggable="true" ondragstart="drag(event)" width="336" height="69">

</body>
</html>
```

OUTPUT

Drag the W3Schools image into the rectangle:

HTML5 <nav> Element

The <nav> element defines a set of navigation links.

Notice that NOT all links of a document should be inside a <nav> element. The <nav> element is intended only for major block of navigation links.

Example

```
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
  <a href="/jquery/">jQuery</a>
</nav>
```

What Is CSS3 And Why Is It Used?

To help build highly interactive online pages, CSS3 is invariably used due to its importance in providing greater options in the design process. When marketing products and services, web design plays a vital part; a site should be created in a manner that will draw potential customers to explore and revisit a website more often. Many **web design firm**s are developing and enhancing websites through the use of CSS3 as this is a great form of web development. This article will help define CSS3 and will point out its advantages.

Definition

The acronym CSS stands for Cascading Style Sheets which is used to augment the functionality, versatility. and efficient performance of site content. It allows for the creation of content-rich websites that do not require much weight or codes; this translates into more interactive graphics and animation, superior user-interface, and significantly more organization and rapid download time.

It is used with HTML to create content structure, with CSS3 being used to format structured content. It is responsible for font properties, colors, text alignments, graphics, background images, tables and other components. This tool provides extra capabilities such as absolute, fixed and relative positioning of various elements. The increasing popularity of CSS3 when used by **web design firm**s stimulates major browsers such as Google Chrome, Firefox, Safari, and IE9 to adopt and embrace this programming language.

Advantages

Although CSS3 is not the only web development solution, it does allow provide greater advantages for several reasons.

- **Customization** – A web page can be customized and alterations created in the design by simply changing a modular file.
- **Bandwidth Requirements** – It decreases server bandwidth requirements, giving rapid download time when a site is accessed with desktop or hand-held devices, providing an improved user experience.
- **Consistency** – It delivers consistent and accurate positioning of navigational elements on the website.
- **Appealing** – It makes the site more appealing with adding videos and graphics easier.
- **Viewing** – It allows online videos to be viewed without the use of third-party plug-ins.
- **Visibility** – It delivers the opportunity to improve brand visibility by designing effective online pages.
- **Cost Effective** – It is cost-effective, time-saving, and supported by most browsers.

Since the introduction of CSS3, there is greater control of the presentation of content and various elements on a website; however it is not really responsible for overall design as it only specifies the structure and content presentation of certain web pages.

External, internal, and inline CSS styles

Cascading Style Sheets (CSS) are files with styling rules that govern how your website is presented on screen. CSS rules can be applied to your website's HTML files in various ways. You can use an **external stylesheet**, an **internal stylesheet**, or an **inline style**. Each method has advantages that suit particular uses.

An **external stylesheet** is a standalone .css file that is linked from a web page. The advantage of external stylesheets is that it can be created once and the rules applied to multiple web pages. Should you need to make widespread changes to your site design, you can make a single change in the stylesheet and it will be applied to all linked pages, saving time and effort.

An **internal stylesheet** holds CSS rules for the page in the **head** section of the HTML file. The rules only apply to that page, but you can configure CSS classes and IDs that can be used to style multiple elements in the page code. Again, a single change to the CSS rule will apply to all tagged elements on the page.

Inline styles relate to a specific HTML tag, using a **style** attribute with a CSS rule to style a specific page element. They're useful for quick, permanent changes, but are less flexible than external and internal stylesheets as each inline style you create must be separately edited should you decide to make a design change.

Using external CSS stylesheets

An HTML page styled by an external CSS stylesheet must reference the .css file in the document head. Once created, the CSS file must be uploaded to your server and linked in the HTML file with code such as:

```
<link href="style.css" rel="stylesheet" type="text/css">
```

You can name your stylesheet whatever you wish, but it should have a .css file extension.

Using internal CSS stylesheets

Rather than linking an external .css file, HTML files using an internal stylesheet include a set of rules in their **head** section. CSS rules are wrapped in <style> tags, like this:

```
<head>
<style type="text/css">

  h1 {
        color:#fff
        margin-left: 20px;
    }

  p {
      font-family: Arial, Helvetica, Sans Serif;
     }



</style>
</head>
```

Using inline styles

Inline styles are applied directly to an element in your HTML code. They use the **style** attribute, followed by regular CSS properties.

For example:

```
<h1 style="color:red;margin-left:20px;">Today’s Update</h1>
```

Rule Cascading

Cascade and inheritance

Conflicting rules

CSS stands for **Cascading Style Sheets**, and that first word *cascading* is incredibly important to understand — the way that the cascade behaves is key to understanding CSS.
At some point, we will find that the CSS have created two rules which could potentially apply to the same element. The **cascade**, and the closely-related concept of **specificity**, are mechanisms that control which rule applies when there is such a conflict. Which rule is styling your element may not be the one you expect, so you need to understand how these mechanisms work.
Also significant here is the concept of **inheritance**, which means that some CSS properties by default inherit values set on the current element's parent element, and some don't. This can also cause some behavior that you might not expect.

The cascade

Stylesheets **cascade** — at a very simple level this means that the order of CSS rules matter; when two rules apply that have equal specificity the one that comes last in the CSS is the one that will be used.

EXAMPLE
In the below example, we have two rules that could apply to the `h1`. The `h1` ends up being colored blue — these rules have an identical selector and therefore carry the same specificity, so the last one in the source order wins.

```
h1 {

    color: red;
}

h1 {

   color: blue;

}

   <h1>This is my heading.</h1>
```

OUTPUT

```
This is my heading.
```

Specificity

Specificity is how the browser decides which rule applies if multiple rules have different selectors, but could still apply to the same element. It is basically a measure of how specific a selector's selection will be:

- An element selector is less specific — it will select all elements of that type that appear on a page — so will get a lower score.
- A class selector is more specific — it will select only the elements on a page that have a specific `class` attribute value — so will get a higher score.

Example time! Below we again have two rules that could apply to the `h1`. The below `h1` ends up being colored red — the class selector gives its rule a higher specificity, and so it will be applied even though the rule with the element selector appears further down in the source order.

EXAMPLE

```
main-heading {

   color: red;

}

      h1 {

   color: blue;

}
```

```
.   <h1 class="main-heading">This is my heading.</h1>
```

OUTPUT

This is my heading.

Inheritance

Inheritance also needs to be understood in this context — some CSS property values set on parent elements are inherited by their child elements, and some aren't.

For example, if you set a color and font-family on an element, every element inside it will also be styled with that color and font, unless you've applied different color and font values directly to them.

Some properties do not inherit — for example if you set a width of 50% on an element, all of its descendants do not get a width of 50% of their parent's width. If this was the case, CSS would be very frustrating to use!

```
body {
   color: blue;
}

span {
   color: black;
}
```

```
<p>As the body has been set to have a color of blue this is inherited through the
descendants.</p>

<p>We can change the color by targetting the element with a selector,
 such as this

<span>span</span>.</p>
```

OUTPUT

As the body has been set to have a color of blue this is inherited through the descendants.

We can change the color by targetting the element with a selector, such as this **span.**

CSS3 Shadow Effects

With CSS3 you can create two types of shadows: text-shadow (adds shadow to text) and box-shadow (adds shadow to other elements).

CSS3 Text Shadow

The text-shadow property can take up to four values:

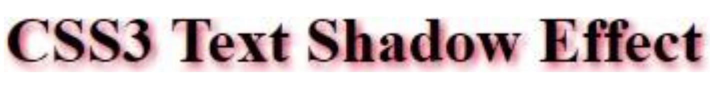
- the horizontal shadow
- the vertical shadow
- the blur effect
- the color

Examples:

- Normal text shadow

```
• h1 {
• text-shadow: 2px 2px 5px crimson;
```

```
}
```

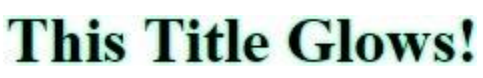


CSS3 Text Shadow Effect

- Glowing text effect

```
• h1 {
• text-shadow: 0 0 4px #00FF9C;
```

```
}
```



This Title Glows!

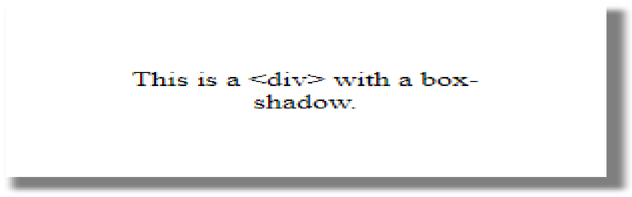
CSS3 Box Shadow

The box-shadow property can take up to six values:

- (optional) the inset keyword (changes the shadow to one inside the frame)
- the horizontal shadow
- the vertical shadow
- the blur effect
- the spreading
- the color

Examples:

```
.first-div {
            box-shadow: 1px 1px 5px 3px grey;
}
```



CSS Animations

CSS allows animation of HTML elements without using JavaScript or Flash!

What are CSS Animations?

An animation lets an element gradually change from one style to another.

You can change as many CSS properties you want, as many times you want.

To use CSS animation, you must first specify some keyframes for the animation.

Keyframes hold what styles the element will have at certain times.

The @keyframes Rule

When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, you must bind the animation to an element.

The following example binds the "example" animation to the <div> element. The animation will last for 4 seconds, and it will gradually change the background-color of the <div> element from "red" to "yellow":

Example

```
/* The animation code */
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

Note: The animation-duration property defines how long time an animation should take to complete. If the animation-duration property is not specified, no animation will occur, because the default value is 0s (0 seconds).

In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

It is also possible to use percent. By using percent, you can add as many style changes as you like.

The following example will change the background-color of the <div> element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

Example

```
/* The animation code */
@keyframes example {
  0%  {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

CSS Animation Properties

The following table lists the @keyframes rule and all the CSS animation properties:

Property	**Description**
@keyframes	Specifies the animation code

animation	A shorthand property for setting all the animation properties

animation-delay	Specifies a delay for the start of an animation
animation-direction	Specifies whether an animation should be played forwards, backwards or in alternate cycles
animation-duration	Specifies how long time an animation should take to complete one cycle
animation-fill-mode	Specifies a style for the element when the animation is not playing (before it starts, after it ends, or both)
animation-iteration-count	Specifies the number of times an animation should be played
animation-name	Specifies the name of the @keyframes animation
animation-play-state	Specifies whether the animation is running or paused
animation-timing-function	Specifies the speed curve of the animation

CSS Transitions

CSS Transitions is a module of CSS that lets you create gradual transitions between the values of specific CSS properties. The behavior of these transitions can be controlled by specifying their timing function, duration, and other attributes.

Properties

- transition
- transition-delay
- transition-duration
- transition-property
- transition-timing-function

The **transition** CSS property is a shorthand property for transition-property, transition-duration, transition-timing-function, and transition-delay.

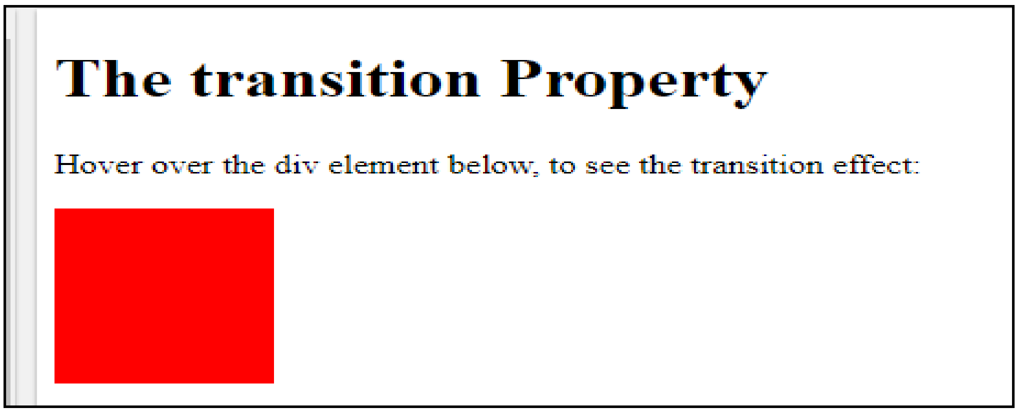
CSS transition Property

Example

Hover over a <div> element to gradually change the width from 100px to 300px:

```
div {
  width: 100px;
  transition: width 2s;
}

div:hover {
  width: 300px;
}
```

OUTPUT


The transition Property
Hover over the div element below, to see the transition effect:

.

Definition and Usage

The transition property is a shorthand property for:

- transition-property
- transition-duration
- transition-timing-function

Property Values

Value	Description
transition-property	Specifies the name of the CSS property the transition effect is for
transition-duration	Specifies how many seconds or milliseconds the transition effect takes to complete
transition-timing-function	Specifies the speed curve of the transition effect
transition-delay	Defines when the transition effect will start
initial	Sets this property to its default value. Read about *initial*
inherit	Inherits this property from its parent element. Read about *inherit*

Example

When an <input type="text"> gets focus, gradually change the width from

100px to 250px:

```
input[type=text] {
  width: 100px;
  transition: width .35s ease-in-out;
}
```

```
input[type=text]:focus {
  width: 250px;
}
```

OUTPUT

The width Property

Set the width of the input field to 100 pixels. However, when the input field gets focus, make it 250 pixels wide:

Search:

CSS background-color

The `background-color` property specifies the background color of an element.

Example

The background color of a page is set like this:

```
body {
  background-color: lightblue;
}
```

CSS background-image

The `background-image` property specifies an image to use as the background of an element.

By default, the image is repeated so it covers the entire element.

Example

The background image for a page can be set like this:

```
body {
  background-image: url("paper.gif");
}
```

CSS background - Shorthand property

To shorten the code, it is also possible to specify all the background properties in one single property. This is called a shorthand property.

The shorthand property for background is `background`.

Example

Use the shorthand property to set all the background properties in one declaration:

```
body {
  background: #ffffff url("img_tree.png") no-repeat right top;
}
```

CSS Border - Shorthand Property

As you can see from the examples above, there are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the individual border properties in one property.

The `border` property is a shorthand property for the following individual border properties:

- `border-width`
- `border-style` (required)
- `border-color`

Example

```
p {
  border: 5px solid red;
}
```

Result:

Some text

UNIT II - CLIENT SIDE PROGRAMMING	
Java Script: An introduction to JavaScript – JavaScript DOM Model – Date and Objects – Regular Expressions – Exception Handling – Validation – Built-in objects – Event Handling – DHTML with JavaScript – JSON introduction – Syntax – Function Files – Http Request – SQL.	
PART - A	
Q.No	**Questions**
1.	**Evaluate** various Java Script Object models.
2.	**Define** DOM.
3.	**Give** any four methods of Date objects.

JavaScript Get Date Methods

These methods can be used for getting information from a date object:

Method	**Description**
getMonth()	Get the **month** as a number (0-11)
getHours()	Get the **hour** (0-23)
getSeconds()	Get the **second** (0-59)
getTime()	Get the time (milliseconds since January 1, 1970)
Date.now()	Get the time. ECMAScript 5.

JavaScript Set Date Methods

Set Date methods let you set date values (years, months, days, hours, minutes, seconds, milliseconds) for a Date Object.

Set Date Methods

Set Date methods are used for setting a part of a date:

Method	Description
setFullYear()	Set the year (optionally month and day)
setMilliseconds()	Set the milliseconds (0-999)
setMonth()	Set the month (0-11)
setTime()	Set the time (milliseconds since January 1, 1970)

4. **Write** the JavaScript methods to retrieve the data and time based on the computer locale.

```
<script>

// Use of Date.now() function
var d = Date(Date.now());

// Converting the number of millisecond in date string
a = d.toString()

// Printing the current date
document.write("The current date is: " + a)

</script>
```

OUTPUT

```
The current date is: Thu Jan 09 2020 20:35:39 GMT+0530 (India Standard Time)
```

5. Can you **list** the different methods defined in document and window object of JavaScript.

Window Object

The window object represents an open window in a browser.

Window Object Methods

Method	Description
alert()	Displays an alert box with a message and an OK button
atob()	Decodes a base-64 encoded string
blur()	Removes focus from the current window
btoa()	Encodes a string in base-64
clearInterval()	Clears a timer set with setInterval()
clearTimeout()	Clears a timer set with setTimeout()
close()	Closes the current window
confirm()	Displays a dialog box with a message and an OK and a Cancel button
focus()	Sets focus to the current window
getComputedStyle()	Gets the current computed CSS styles applied to an element
getSelection()	Returns a Selection object representing the range of text selected by the user
matchMedia()	Returns a MediaQueryList object representing the specified CSS media query string
moveBy()	Moves a window relative to its current position
moveTo()	Moves a window to the specified position
open()	Opens a new browser window
print()	Prints the content of the current window
prompt()	Displays a dialog box that prompts the visitor for input

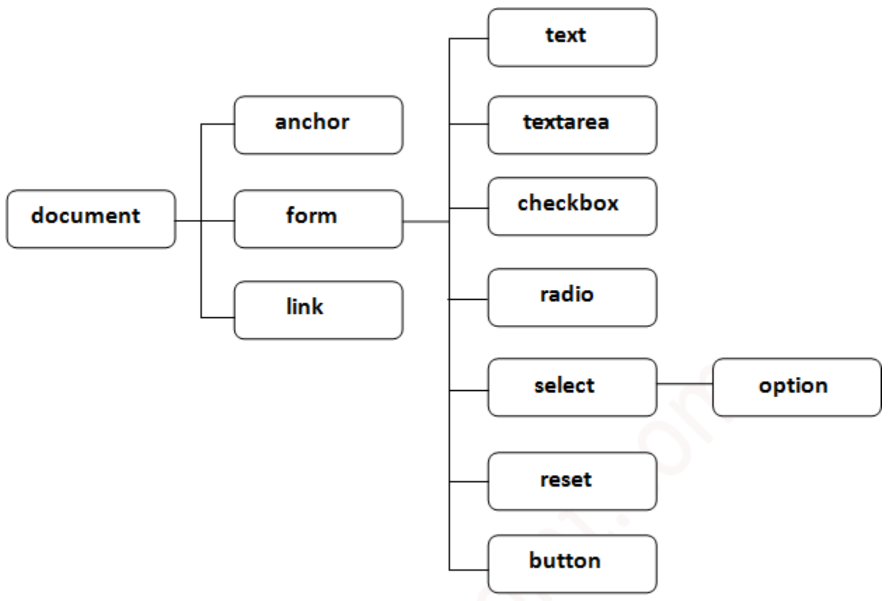
requestAnimationFrame()	Requests the browser to call a function to update an animation before the next repaint
resizeBy()	Resizes the window by the specified pixels
resizeTo()	Resizes the window to the specified width and height
scroll()	Deprecated. This method has been replaced by the scrollTo() method.
scrollBy()	Scrolls the document by the specified number of pixels
scrollTo()	Scrolls the document to the specified coordinates
setInterval()	Calls a function or evaluates an expression at specified intervals (in milliseconds)
setTimeout()	Calls a function or evaluates an expression after a specified number of milliseconds
stop()	Stops the window from loading

Document Object Model

1. Document Object
2. Properties of document object
3. Methods of document object
4. Example of document object

➢ The **document object** represents the whole html document.
➢ When html document is loaded in the browser, it becomes a document object.
➢ It is the **root element** that represents the html document. It has properties and methods.
➢ By the help of document object, we can add dynamic content to our web page

➢ According to W3C - *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*



Properties of document object

Let's see the properties of document object that can be accessed and modified by the document object.

Methods of document object

We can access and change the contents of document by its methods.

The important methods of document object are as follows:

Method	Description
write("string")	writes the given string on the doucment.
writeln("string")	writes the given string on the doucment with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
getElementsByClassName()	returns all the elements having the given class name.

6.	**Name** which parser is best in parsing in large size documents. Why?
7.	**Summarize** benefits of using JavaScript code in an HTML document. **Advantages of JavaScript** The merits of using JavaScript are − • **Less server interaction** − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server. • **Immediate feedback to the visitors** − They don't have to wait for a page reload to see if they have forgotten to enter something. • **Increased interactivity** − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard. • **Richer interfaces** − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.
8.	**Predict** the need for client and server side scripting. **Client-side scripting** (embedded scripts) is code that exists inside the client's HTML page. This code will be processed on the client machine and the HTML page will NOT perform a PostBack to the web-server. Traditionally, client-side scripting is used for page navigation, data validation and formatting. The language used in this scripting is JavaScript. JavaScript is compatible and is able to run on any internet browser. The two main benefits of client-side scripting are: 1. The user's actions will result in an immediate response because they don't require a trip to the server. 2. Fewer resources are used and needed on the web-server. **Server-side scripting** is a technique used in web development which involves employing **scripts** on a web **server** which produce a response customized for each user's (client's) request to the website. The alternative is for the web **server** itself to deliver a static web page. The **client-side script** executes the code to the **client side** which is visible to the users while a **server-side script** is executed **in the server** end which users cannot see.
9.	**Interpret** how exceptions are handled in Java script.

The try...catch...finally Statement

The latest versions of JavaScript added exception handling capabilities. JavaScript implements the **try...catch...finally** construct as well as the **throw** operator to handle exceptions.

You can **catch** programmer-generated and **runtime** exceptions, but you cannot **catch** JavaScript syntax errors.

Here is the **try...catch...finally** block syntax –

```
<script type = "text/javascript">
  <!--
    try {
      // Code to run
      [break;]
    }

    catch ( e ) {
      // Code to run if an exception occurs
      [break;]
    }

    [ finally {
      // Code that is always executed regardless of
      // an exception occurring
    }]
  //-->
</script>
```

The **try** block must be followed by either exactly one **catch** block or one **finally** block

(or one of both). When an exception occurs in the **try** block, the exception is placed in **e** and the

catch block is executed. The optional **finally** block executes unconditionally after try/catch.

Examples

Here is an example where we are trying to call a non-existing function which in turn is raising

an exception. Let us see how it behaves without **try...catch**–

```
<html>
  <head>
    <script type = "text/javascript">
      <!--
        function myFunc() {
          var a = 100;
          alert("Value of variable a is : " + a );
        }
      //-->
    </script>
  </head>

  <body>
    <p>Click the following to see the result:</p>

    <form>
      <input type = "button" value = "Click Me" onclick = "myFunc();" />
    </form>
  </body>
</html>
```

10.

Define JavaScript statement with an example.

JavaScript Statements

Example

```
var x, y, z;    // Statement 1
x = 5;          // Statement 2
y = 6;          // Statement 3
z = x + y;      // Statement 4
```

JavaScript statements are composed of:
Values, Operators, Expressions, Keywords, and Comments.
This statement tells the browser to write "Hello Dolly." inside an HTML element with id="demo":

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Statements</h2>

<p>In HTML, JavaScript statements are executed by the browser.</p>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML = "Hello Dolly.";

</script>

</body>

</html>
```

OUTPUT

JavaScript Statements

In HTML, JavaScript statements are executed by the browser.

Hello Dolly.

11. **Point out** any two techniques of event programming.

What is an Event ?

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.

Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.

Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.

Please go through this small tutorial for a better understanding HTML Event Reference. Here we will see a few examples to understand a relation between Event and JavaScript −

onclick Event Type

This is the most frequently used event type which occurs when a user clicks the left button of his mouse. You can put your validation, warning etc., against this event type.

Example

Try the following example.

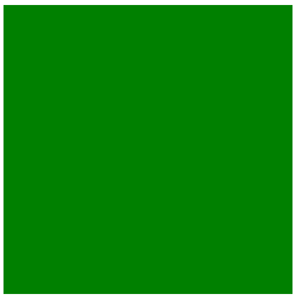
```
<html>

  <head>

    <script type = "text/javascript">

      <!--

        function sayHello() {

          alert("Hello World")

        }

      //-->

    </script>

  </head>
  <body>

    <p>Click the following button and see result</p>

    <form>

      <input type = "button" onclick = "sayHello()" value = "Say Hello" />

    </form>

  </body>

</html>
```

Example 2

```
<!doctype html>
<html>
<head>
          <script>
          function hov() {
                    var e = document.getElementById('hover');
                    e.style.display = 'none';
          }
          </script>
</head>
<body>
          <div id="hover" onmouseover="hov()"
          style="background-color:green;height:200px;width:200px;">
          </div>
</body>
</html>
```

OUTPUT

Before mouse is taken over green square-



Green square gets disappear after mouse is taken over it.

12. **What** is DHTML?

DHTML

DHTML stands for Dynamic HTML, it is totally different from HTML. The DHTML make use of Dynamic object model to make changes in settings and also in properties and methods.

	DHTML allows different scripting languages in a web page to change their variables, which enhance the effects, looks and many others functions after the whole page have been fully loaded or under a view process, or otherwise static HTML pages on the same. DHTML is used to create interactive and animated web pages that are generated in real-time, also known as dynamic web pages so that when such a page is accessed, the code within the page is analyzed on the web server and the resulting HTML is sent to the client's web browser.
13.	**Differentiate** HTML and DHTML **Some differences between HTML and DHTML:** • HTML is a mark-up language, while DHTML is a collection of technology. • DHTML creates dynamic web pages, whereas HTML creates static web pages. • DHTML allows including small animations and dynamic menus in Web pages. • DHML used events, methods, properties to insulate dynamism in HTML Pages. • DHML is basically using JavaScript and style sheets in an HTML page. • HTML sites will be slow upon client-side technologies, while DHTML sites will be fast enough upon client-side technologies. • HTML creates a plain page without any styles and Scripts called as HTML. Whereas, DHTML creates a page with HTML, CSS, DOM and Scripts called as DHTML. • HTML cannot have any server side code but DHTML may contain server side code. • In HTML, there is no need for database connectivity, but DHTML may require connecting to a database as it interacts with user. • HTML files are stored with .htm or .html extension, while DHTML files are stored with .dhtm extension. • HTML does not require any processing from browser, while DHTML requires processing from browser which changes its look and feel.
14.	**Point** out the key features of DHTML. **Key Features:** Following are the some major key features of DHTML: • Tags and their properties can be changed using DHTML. • It is used for real-time positioning. • Dynamic fonts can be generated using DHTML. • It is also used for data binding. • It makes a webpage dynamic and be used to create animations, games, applications along with providing new ways of navigating through websites. • The functionality of a webpage is enhanced due to the usage of low-bandwidth effect by DHTML. • DHTML also facilitates the use of methods, events, properties, and codes.
15.	**Classify** the data types in JSON **JSON** (JavaScript **Object** Notation) is a lightweight **data-**interchange **format**. It is easy for humans to read and write. ....**JSON** is built on two **structures**: A collection of name/value pairs. In various languages, this is realized as an **object**, record, struct, dictionary, hash table, keyed list, or associative array.

JSON Data Types. At the granular level, **JSON** consist of 6 **data types**. First four **data types** (string, number, boolean and null) can be referred as simple **data types**. Other two **data types** (**object** and array) can be referred as complex **data types**.

16.

How to convert text into a JavaScript object using JSON?

Converting JSON text to JavaScript Object

JSON (JavaScript Object Notation) is a lightweight data-interchange format. As its name suggests, JSON is derived from the JavaScript programming language, but it's available for use by many languages including Python, Ruby, PHP, and Java and hence, it can be said as language-independent. For humans, it is easy to read and write and for machines, it is easy to parse and generate. It is very useful for storing and exchanging data.

A JSON object is a key-value data format that is typically rendered in curly braces. JSON object consist of curly braces (**{ }**) at the either ends and have key-value pairs inside the braces. Each key-value pair inside braces are separated by comma (**,**). JSON object looks something like this :

```
{

    "key":"value",

    "key":"value",

    "key":"value",

}
```

Example for a JSON object :

```
{

    "rollno":101",

    "name":"Mayank",

    "age":20,

}
```

Conversion of JSON text to Javascript Object

JSON text/object can be converted into Javascript object using the function **JSON.parse()**.

```
var object1 = JSON.parse('{"rollno":101, "name":"Mayank", "age":20}');
```

For getting the value of any key from a Javascript object, we can use the values as: **object1.rollno**

17. **Evaluate** the syntax to create arrays in JSON.

JavaScript | JSON Arrays

The JSON Arrays is similar to JavaScript Arrays.

Syntax of Arrays in JSON Objects:

```
// JSON Arrays Syntax

{
    "name":"Peter parker",
    "heroName": "Spiderman",
    "friends" : ["Deadpool", "Hulk", "Wolverine"]
}
```

Accessing Array Values:
The Array values can be accessed using the index of each element in an Array.

EXAMPLE

```
<script>
var myObj, i, x = "";
myObj = {
"name":"John",
"age":30,
  "cars":[ "Ford", "BMW", "Fiat" ]
};

for (i in myObj.cars) {
  x += myObj.cars[i] + "<br>";
}
document.getElementById("demo").innerHTML = x;
</script>
```

OUTPUT

```
Looping through an array using a for in loop:

Ford
BMW
Fiat
```

18. **How** will you make a request with JSON?

What is a JSON request?
JavaScript Object Notation (**JSON**) is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications (e.g., sending some data from the server to the client, so it can be displayed on a web page, or vice versa).

jQuery getJSON() Method

The jQuery getJSON() method sends asynchronous http GET request to the server and retrieves the data in JSON format by setting accepts header to application/json, text/javascript. This is same as get() method, the only difference is that getJSON() method specifically retrieves JSON data whereas get() method retrieves any type of data. It is like shortcut method to retrieve JSON data.

```
$.getJSON(url,[data],[callback]);
```

Parameter Description:

- url: request url from which you want to retrieve the data
- data: JSON data to be sent to the server as a query string
- callback: function to be executed when request succeeds

The following example shows how to retrieve JSON data using getJSON() method.

Example: jQuery getJSON() Method

```
$.getJSON('/jquery/getjsondata', {name:'Steve'}, function (data, textStatus, jqXHR){
   $('p').append(data.firstName);
});

<p></p>
```

OUTPUT

jQuery get() method demo

Steve

19. **Define** DDL and DML

DDL(Data Definition Language) : DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.

Examples of DDL commands:

- CREATE - is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
- DROP - is used to delete objects from the database.
- ALTER-is used to alter the structure of the database.
- TRUNCATE-is used to remove all records from a table, including all spaces allocated for the records are removed.
- COMMENT -is used to add comments to the data dictionary.
- RENAME -is used to rename an object existing in the database

20. **Write** SQL query to find minimum and maximum marks in a table.

DML(Data Manipulation Language) : The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

Examples of DML:

- INSERT - is used to insert data into a table.
- UPDATE - is used to update existing data within a table.
- DELETE - is used to delete records from a database table.

PART - B

Q.No	Questions
1.	i) **Examine** variables and data types in JavaScript.

Variables in JavaScript:

Variables in JavaScript are containers which hold reusable data. It is the basic unit of storage in a program.

- The value stored in a variable can be changed during program execution.
- A variable is only a name given to a memory location, all the operations done on the variable effects that memory location.
- In JavaScript, all the variables must be declared before they can be used.

JavaScript variables are containers for storing data values.

In this example, x, y, and z, are variables:

Examples

var x = 5;
var y = 6;
var z = x + y;
	var price1 = 5;
	var price2 = 6;
	var total = price1 + price2;

Declaring (Creating) JavaScript Variables

Creating a variable in JavaScript is called "declaring" a variable.

You declare a JavaScript variable with the var keyword:

```
var carName;
```

After the declaration, the variable has no value (technically it has the value of undefined).

To **assign** a value to the variable, use the equal sign:

```
carName = "Volvo";
```

You can also assign a value to the variable when you declare it:

```
var carName = "Volvo";
```

EXAMPLE

`<script>`
`var carName = "Volvo";`
`document.getElementById("demo").innerHTML`
`= carName;`
`</script>`
	OUTPUT
	JavaScript Variables
	
	Create a variable, assign a value to it, and display it:
	
	Volvo

ii) **Give** various Operators in JavaScript.

JavaScript Operators

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic on numbers:

Operator	Description	Example

+	Addition	var x = 5;
		var y = 2;
		var z = x + y;

-	Subtraction	var x = 5;
		var y = 2;
		var z = x - y;

*	Multiplication	var x = 5;
		var y = 2;
		var z = x * y;

**	Exponentiation (ES2016)	
/	Division	
%	Modulus (Division Remainder)	
++	Increment	
--	Decrement	

JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

Operator	Example	Same As

=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

The **addition assignment** operator (+=) adds a value to a variable.

Assignment

```
var x = 10;
x += 5;
```

JavaScript String Operators

The + operator can also be used to add (concatenate) strings.

Example

```
var txt1 = "John";
var txt2 = "Doe";
var txt3 = txt1 + " " + txt2;
```

The result of txt3 will be:

John Doe

The += assignment operator can also be used to add (concatenate) strings:

Example

```
var txt1 = "What a very ";
txt1 += "nice day";
```

The result of txt1 will be:

What a very nice day

When used on strings, the + operator is called the concatenation operator.

JavaScript Comparison Operators

Operator	**Description**
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type

>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

JavaScript Logical Operators

Operator	**Description**
||	logical or

JavaScript Type Operators

Operator	**Description**
instanceof	Returns true if an object is an instance of an object type

Type operators are fully described in the **JS Type Conversion** chapter.

JavaScript Bitwise Operators

Bit operators work on 32 bits numbers.

Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

Operator	**Description**	**Example**	**Same as**	**Result**	**Decimal**
&	AND	5 & 1	0101 & 0001	0001	1
|	OR	5 | 1	0101 | 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	Zero fill left shift	5 << 1	0101 << 1	1010	10
>>	Signed right shift	5 >> 1	0101 >> 1	0010	2

	>>>	Zero fill right shift	5 >>> 1	0101 >>> 1	0010	2

The examples above uses 4 bits unsigned examples. But JavaScript uses 32-bit signed numbers. Because of this, in JavaScript, ~ 5 will not return 10. It will return -6.
~00000000000000000000000000000101 will return 11111111111111111111111111111010

2.

(i) **Summariz**e about DOM Model.

What is the DOM?

The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The W3C DOM standard is separated into 3 different parts:

- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

JavaScript HTML DOM

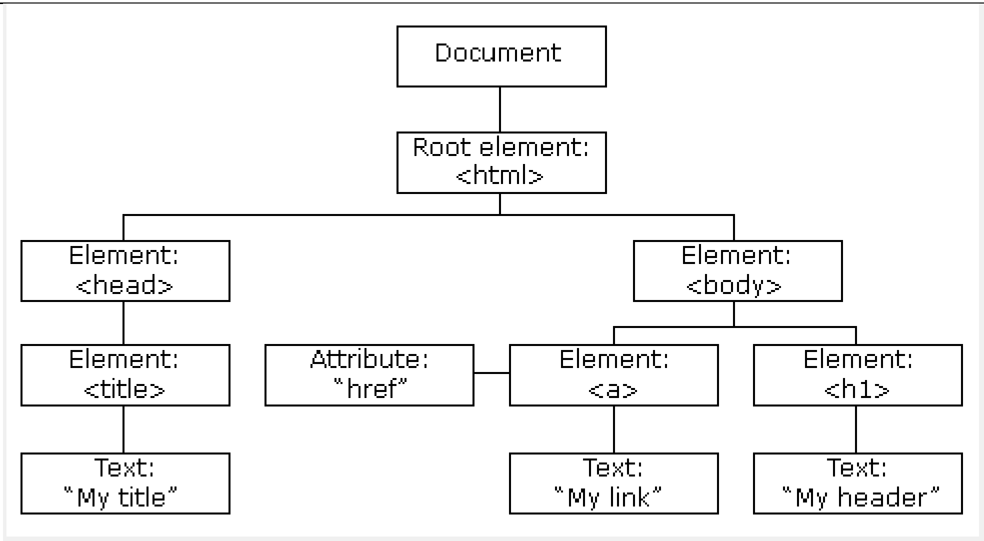
With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

The HTML DOM (Document Object Model)

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The **HTML DOM** model is constructed as a tree of **Objects**:

The HTML DOM Tree of Objects



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

(ii) **Describe** the concepts of Popup Boxes.

JavaScript Popup Boxes

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

JavaScript Popup Boxes

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

Alert Box	Syntax	Example
An alert box is often used if you want to make sure information comes through to the user. When an alert box pops up, the user will have to click "OK" to proceed	window.alert("*sometext*"); The window.alert() method can be written without the window prefix	alert("I am an alert box!");

	Confirm Box
	
	When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
	
	If the user clicks "OK", the box returns **true**. If the user clicks "Cancel", the box returns **false**.
		Syntax
		
		window.confirm("*sometext*");
		
		The window.confirm() method can be written without the window prefix.
			Example
			
			`if (confirm("Press a button!")) {`
			`  txt = "You pressed OK!";`
			`} else {`
			`  txt = "You pressed Cancel!";`
			`}`

	Prompt Box
	
	A prompt box is often used if you want the user to input a value before entering a page.
	
	If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.
		Syntax
		
		window.prompt("*sometext*","*defaultText*");
		
		The window.prompt() method can be written without the window prefix.
			Example
			
			`var person = prompt("Please enter your name", "Harry Potter");`
			
			`if (person == null || person == "") {`
			`  txt = "User cancelled the prompt.";`
			`} else {`
			`  txt = "Hello " + person + "! How are you today?";`
			`}`

3. (i) **Write** short notes on Date and Objects.

The **Date Object**. The **Date object** is a built-in **object** in **JavaScript** that stores the **date** and time. It provides a number of built-in methods for formatting and managing that data. By default, a new **Date** instance without arguments provided creates an **object** corresponding to the current **date** and time

The Date Object

The `Date` object is a built-in object in JavaScript that stores the date and time. It provides a number of built-in methods for formatting and managing that data.

By default, a new `Date` instance without arguments provided creates an object corresponding to the current date and time. To demonstrate JavaScript's `Date`, let's create a variable and assign the current date to it.

	EXAMPLE
	
	now.js
	
	`// Set variable to current date and time`
	`const now = new Date();`
	
	`// View the output`
	`now;`
	`Output`
	`Wed Oct 18 2017 12:41:34 GMT+0000 (UTC)`

`new Date()`	Current date and time
`new Date(timestamp)`	Creates date based on milliseconds since Epoch time
`new Date(date string)`	Creates date based on date string
`new Date(year, month, day, hours, minutes, seconds, milliseconds)`	Creates date based on specified date and time

(ii) **Explain** in detail about Regular expressions.

A JavaScript Regular Expression (or **Regex**) is a sequence of characters that we can utilize to work effectively with strings. Using this syntax, we can:

- **search** for text in a string
- **replace** substrings in a string
- **extract** information from a string

4. (i) **Describe** the control statements in Java.

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions.

You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- Use `if` to specify a block of code to be executed, if a specified condition is true
- Use `else` to specify a block of code to be executed, if the same condition is false
- Use `else if` to specify a new condition to test, if the first condition is false
- Use `switch` to specify many alternative blocks of code to be executed

JavaScript Loops

Loops are handy, if you want to run the same code over and over again, each time with a different value.

Often this is the case when working with arrays:

```
<script>

var cars = ["BMW", "Volvo", "Saab", "Ford", "Fiat", "Audi"];

var text = "";

var i;
for (i = 0; i < cars.length; i++) {
  text += cars[i] + "<br>";


}

document.getElementById("demo").innerHTML = text;

</script>
```

OUTPUT

```
JavaScript For Loop

BMW
Volvo
Saab
Ford
Fiat
Audi
```

	(ii) **Discuss** any two validation functions in java script.

5.	(i) **Write** a Java script to find the Prime number between 1 and 100.
	
	`<html>`
	`   <head>`
	`      <title>JavaScript Prime</title>`
	`   </head>`
	
	`   <body>`
	`      <script>`
	`         for (var limit = 1; limit <= 20; limit++) {`
	`            var a = false;`
	`            for (var i = 2; i <= limit; i++) {`
	`               if (limit%i===0 &amp;&amp; i!==limit) {`
	`                  a = true;`
	`               }`
	`            }`
	`            if (a === false) {`
	`               document.write("<br>"+limit);`
	`            }`
	`         }`
	`      </script>`
	`   </body>`
	`</html>`
	
	(ii) **Write** a Java Script to find factorial of a given number.
	
	`function factorial(n) {`
	`  return (n != 1) ? n * factorial(n - 1) : 1;`
	`}`
	
	`alert( factorial(5) ); // 120`

6.	(i) **Demonstrate** a java script for displaying the context menu.
	
	(ii) **Demonstrate** a java script to display the welcome message using the alert whenever button of a html form is pressed.
	`<html>`
	`        <head>`
	`                <title>Display Alert Message on Button Click Event.</title>`
	`                <script type="text/javascript">`
	`                        function showMessage(){`
	`                                alert("Welcome friends, You pressed the Button.");`
	`                        }`
	`                </script>`
	`        </head>`
	`<body>`

```
        <center>
                <h1>Display Alert Message on Button Click Event.</h1>
                <b>Click on button to display message: </b><br><br>
                <input type="button" id="btnShowMsg" value="Click Me!"
onClick='showMessage()'/>
        </center>
</body>

</html>
```

Result

Welcome friends, You pressed the Button.

7.

(i) Evaluate how DOM Nodes and Trees can be used.

What is the purpose of HTML DOM Node Tree?

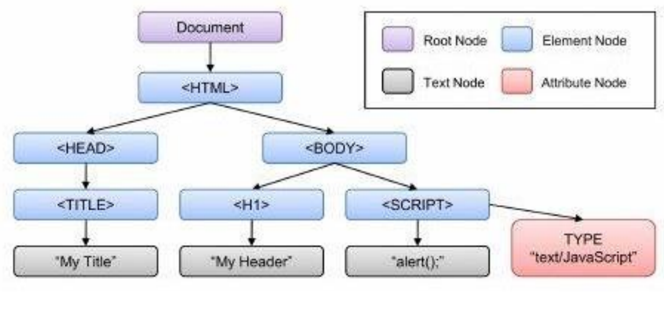
- HTML DOM view the HTML document with a tree structure format and it consists of root node and child nodes.

- The node-tree is being accessed using the tree formation and the structure in which the elements get created.

- The contents that are used being modified or removed using the new elements and it can be created within the limitations.

- The structure consists of a document that is the root and within it Root element <html> from where the tree starts.

- It consists of sub-elements like <head> and <body> and other text and attributes written in the HTML format.



(ii) Evaluate the way of Traversing and modifying a DOM Tree.

8.

(i) **Discuss** the concepts of Registering Event handlers.

As mentioned above, **events** are actions or occurrences that happen in the system you are programming — the system produces (or "fires") a signal of some kind when an event occurs, and also provides a mechanism by which some kind of action can be automatically taken (that is, some code running) when the event occurs. For example in an airport when the runway is clear for a plane to take off, a signal is communicated to the pilot, and as a result, they commence piloting the plane.

In the case of the Web, events are fired inside the browser window, and tend to be attached to a specific item that resides in it — this might be a single element, set of elements, the HTML document loaded in the current tab, or the entire browser window. There are a lot of different types of events that can occur, for example:

The user clicking the mouse over a certain element or hovering the cursor over a certain element.

The user pressing a key on the keyboard.

The user resizing or closing the browser window.

A web page finishing loading.

A form being submitted.

A video being played, or paused, or finishing play.

An error occurring.

Each available event has an **event handler**, which is a block of code (usually a JavaScript function that you as a programmer create) that will be run when the event fires. When such a block of code is defined to be run in response to an event firing, we say we are **registering an event handler**. Note that event handlers are sometimes called **event listeners** — they are pretty much interchangeable for our purposes, although strictly speaking, they work together. The listener listens out for the event happening, and the handler is the code that is run in response to it happening.
In the following example, we have a single `<button>`, which when pressed, makes the background change to a random color:

```
<button>Change color</button>
```

The JavaScript looks like so:

```
const btn = document.querySelector('button');

function random(number) {
  return Math.floor(Math.random() * (number+1));
}

btn.onclick = function() {
  const rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' +
random(255) + ')';
  document.body.style.backgroundColor = rndCol;
}
```

(ii) **Discuss** the concepts of Event Handling.

	## JavaScript Events
	
	HTML events are **"things"** that happen to HTML elements.
	
	When JavaScript is used in HTML pages, JavaScript can **"react"** on these events.
	
	## HTML Events
	
	An HTML event can be something the browser does, or something a user does.
	
	Here are some examples of HTML events:
	
	• An HTML web page has finished loading
	• An HTML input field was changed
	• An HTML button was clicked
	
	Often, when events happen, you may want to do something.
	
	JavaScript lets you execute code when events are detected.
	
	HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.
	
	With single quotes:
	
	`<element event='some JavaScript'>`
	
	With double quotes:
	
	`<element event="some JavaScript">`
	
	In the following example, an `onclick` attribute (with code), is added to a `<button>` element:
	
	## Example
	
	`<button onclick="document.getElementById('demo').innerHTML = Date()">The time is?</button>`
	
	OUTPUT
	Sun Jan 12 2020 06:00:41 GMT+0530 (India Standard Time)
	
	In the example above, the JavaScript code changes the content of the element with id="demo".

9.	**Analyze** a web page to create a clock with timing event.
	function showTime(){
	    var date = new Date();
	    var h = date.getHours(); // 0 - 23
	    var m = date.getMinutes(); // 0 - 59

```
    var s = date.getSeconds(); // 0 - 59
    var session = "AM";

    if(h == 0){
        h = 12;
    }

    if(h > 12){
        h = h - 12;
        session = "PM";
    }

    h = (h < 10) ? "0" + h : h;
    m = (m < 10) ? "0" + m : m;
    s = (s < 10) ? "0" + s : s;

    var time = h + ":" + m + ":" + s + " " + session;
    document.getElementById("MyClockDisplay").innerText = time;
    document.getElementById("MyClockDisplay").textContent = time;

    setTimeout(showTime, 1000);

}

showTime();
```

```
<html>
<head>
<title>Digital Clock</title>
<style>
#clock{
        color:#F0F;
}


</style>
</head>
<body>
<script>
function digclock()
{
        var d = new Date()
        var t = d.toLocaleTimeString()
        document.getElementById("clock").innerHTML = t

}


setInterval(function(){digclock()},1000)


</script>

Digital Clock

<div id="clock">


</div>

</body>

</html>
```

OUTPUT

Output :

Digital Clock
5:54:26 PM

10. (i) **Write** short notes on DHTML with JavaScript.

Dynamic HyerText Markup Language (DHTML) is a combination of Web development technologies used to create dynamically changing websites. Web pages may include animation, dynamic menus and text effects. The technologies used include a combination of HTML, JavaScript or VB Script,
CSS and the document object model (DOM).

Designed to enhance a Web user's experience, DHTML includes the following features:

- Dynamic content, which allows the user to dynamically change Web page content
- Dynamic positioning of Web page elements
- Dynamic style, which allows the user to change the Web page's color, font, size or content

EXAMPLE

```
<! DOCTYPE html PUBLIC "-//abc//DTD XHTML 1.1//EN"
"http://www.abc.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.abc.org/1999/xhtml">
<head>
<title>DHTML example</title>
<script type="text/JavaScript">
   function greet_user()
   {
      var name=document.getElementById("userName").value;
```

	if(name=="")
	        {
	                alert("Welcome"+name);
	        }
	        else
	        {
	                alert("Please provide User Name")
	        }
	    }
	</script>
	</head>
	    <body>
	        <table border="1" cellspacing="3">
	        <tr>
	                <td colspan="2"><h6> Please Enter Your Name </h6></td>
	        </tr>
	        <tr>
	                <td><h4>User Name </h4></td>
	                <td><input type="text" id="userName" ></td>
	        </tr>
	        <tr>
	                <td colspan="2"><input type="button" value="Submit"
	onclick="greet_user()"/>
	        </table>
	    </body>
	</html>
	(ii) **Classify** moving elements.

11.	**Explain** the concept of JSON with example. JSON is a format for storing and transporting data. JSON is often used when data is sent from a server to a web page. **What is JSON?** • JSON stands for **J**ava**S**cript **O**bject **N**otation • JSON is a lightweight data interchange format • JSON is language independent ***** • JSON is "self-describing" and easy to understand * The JSON syntax is derived from JavaScript object notation syntax, but the JSON format is text only. Code for reading and generating JSON data can be written in any programming language.

JSON Example

This JSON syntax defines an employees object: an array of 3 employee records (objects):

JSON Example

```
{
"employees":[
  {"firstName":"John", "lastName":"Doe"},
  {"firstName":"Anna", "lastName":"Smith"},
  {"firstName":"Peter", "lastName":"Jones"}
]
}
```

12. **Describe** in detail about JSON Objects and Arrays.

JSON Arrays

Arrays as JSON Objects

Example

```
[ "Ford", "BMW", "Fiat" ]
```

Arrays in JSON are almost the same as arrays in JavaScript.

In JSON, array values must be of type string, number, object, array, boolean or *null*.

In JavaScript, array values can be all of the above, plus any other valid JavaScript expression, including functions, dates, and *undefined.*

Arrays in JSON Objects

Arrays can be values of an object property:

Example

```
{
"name":"John",
"age":30,
"cars":[ "Ford", "BMW", "Fiat" ]
}
```

	Accessing Array Values
	
	You access the array values by using the index number:
	
	Example
	
	`x = myObj.cars[0];`
	
	Looping Through an Array
	
	You can access array values by using a **for-in** loop:
	
	Example
	
	`for (i in myObj.cars) {`
	`  x += myObj.cars[i];`
	`}`
	
	OUTPUT
	
	Looping through an array using a for in loop:
	
	Ford
	BMW
	Fiat

13.	**Analyze** about Function files and Http Request with sample program.
	
	`<!doctype html>`
	`<title>Example</title>`
	
	`<script>`
	`// Store XMLHttpRequest and the JSON file location in variables`
	`var xhr = new XMLHttpRequest();`
	`var url = "https://www.quackit.com/json/tutorial/artists.txt";`
	
	`// Called whenever the readyState attribute changes`
	`xhr.onreadystatechange = function() {`
	
	`  // Check if fetch request is done`
	`  if (xhr.readyState == 4 && xhr.status == 200) {`
	
	`    // Parse the JSON string`
	`    var jsonData = JSON.parse(xhr.responseText);`
	
	`    // Call the showArtists(), passing in the parsed JSON string`
	`    showArtists(jsonData);`
	`  }`
	`};`

```
// Do the HTTP call using the url variable we specified above
xhr.open("GET", url, true);
xhr.send();

// Function that formats the string with HTML tags, then outputs the result
function showArtists(data) {
    var output = "<ul>"; // Open list
    var i;

    // Loop through the artists, and add them as list items
    for (var i in data.artists) {
        output += "<li>" + data.artists[i].artistname + " (Born: " + data.artists[i].born + ")</li>";
    }

    output += "</ul>"; // Close list

    // Output the data to the "artistlist" element
    document.getElementById("artistList").innerHTML = output;
}
</script>

<!-- The output appears here -->
<div id="artistList"></div>
```

OUTPUT

- Leonard Cohen (Born: 1934)
- Joe Satriani (Born: 1956)
- Snoop Dogg (Born: 1971)

14.

Summarize about
(i) SQL Data Definition Commands
Examples of Sql DDL commands are

- CREATE – Create an object. I mean, create a database, table, triggers, index, functions, stored procedures, etc.
- DROP – This SQL DDL command helps to delete objects. For example, delete tables, delete a database, etc.
- ALTER – Used to alter the existing database or its object structures.
- TRUNCATE – This SQL DDL command removes records from tables
- RENAME – Renaming the database objects

(ii) Data Manipulation Commands.
Examples of DML commands are

- SELECT – This SQL DML command select records or data from a table
- INSERT – Insert data into a database table.
- UPDATE – This SQL DML command will update existing records within a table

	• DELETE – Delete unwanted records from a table

PART – C

Q.No	Questions

1.	**Analyze** the merits and demerits of DOM parser with neat example.
	
	`<!DOCTYPE html>`
	`<html>`
	`<head>`
	`   <script type="text/javascript">`
	`      var str = "<root><customer name='John' address='Chicago'></customer></root>";`
	`      function CreateXMLDocument () {`
	`         var xmlDoc = null;`
	`         if (window.DOMParser) {`
	`            var parser = new DOMParser ();`
	`            xmlDoc = parser.parseFromString (str, "text/xml");`
	`         } else if (window.ActiveXObject) {`
	`            xmlDoc = new ActiveXObject ("Microsoft.XMLDOM");`
	`            xmlDoc.async = false;`
	`            xmlDoc.loadXML (str);`
	`         }`
	
	`         var customerNode = xmlDoc.getElementsByTagName ("customer")[0];`
	`         var customerName = customerNode.getAttribute ("name");`
	`         alert ("The name of the first customer is " + customerName);`
	`      }`
	`   </script>`
	`</head>`
	`<body>`
	`   <button onclick="CreateXMLDocument ();">Create an XML document from a string</button>`
	`</body>`
	`</html>`

2.	**Consider** a java script and HTML to create a page with two panes. The first pane (on left) should have a text area where HTML code can bet typed by the user. The pane on the left should have a text area where HTML code can be typed by the user. The pane on the right side should display the preview of the HTML code typed by the user, as it would be seen on the browser.

3.	**Develop** a DHTML program to handle the user click event.
	
	`<!DOCTYPE html>`
	`<html>`

```
<body>

<p>Click the button to display the time.</p>

<button onclick="getElementById('demo').innerHTML=Date()">What is the
time?</button>

<p id="demo"></p>

</body>
</html>
```

OUTPUT

```
Click the button to display the time.

What is the time?

Sat Jan 11 2020 22:36:24 GMT+0530 (India Standard Time)
```

4. **Formulate** a JavaScript program that work with JSON.

JSON Object Syntax

Example

```
{ "name":"John", "age":30, "car":null }
```

- JSON objects are surrounded by curly braces {}.
- JSON objects are written in key/value pairs.
- Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null).
- Keys and values are separated by a colon.
- Each key/value pair is separated by a comma.

Accessing Object Values

You can access the object values by using dot (.) notation:

Example

```
myObj = { "name":"John", "age":30, "car":null };
x = myObj.name;
```

You can also access the object values by using bracket ([]) notation:

Example

```
myObj = { "name":"John", "age":30, "car":null };
x = myObj["name"];
```

Looping an Object

You can loop through object properties by using the for-in loop:

Example

In a for-in loop, use the bracket notation to access the property *values*:

Example

```
<!DOCTYPE html>
<html>
<body>

<p>How to loop through all properties in a JSON object.</p>

<p id="demo"></p>

<script>
var myObj, x;
myObj = {"name":"John", "age":30, "car":null};
for (x in myObj) {
  document.getElementById("demo").innerHTML += x + " " + myObj[x] + "<br>";
}
</script>

</body>
</html
```

OUTPUT

```
How to loop through all properties in a JSON object.

name John
age 30
car null
```

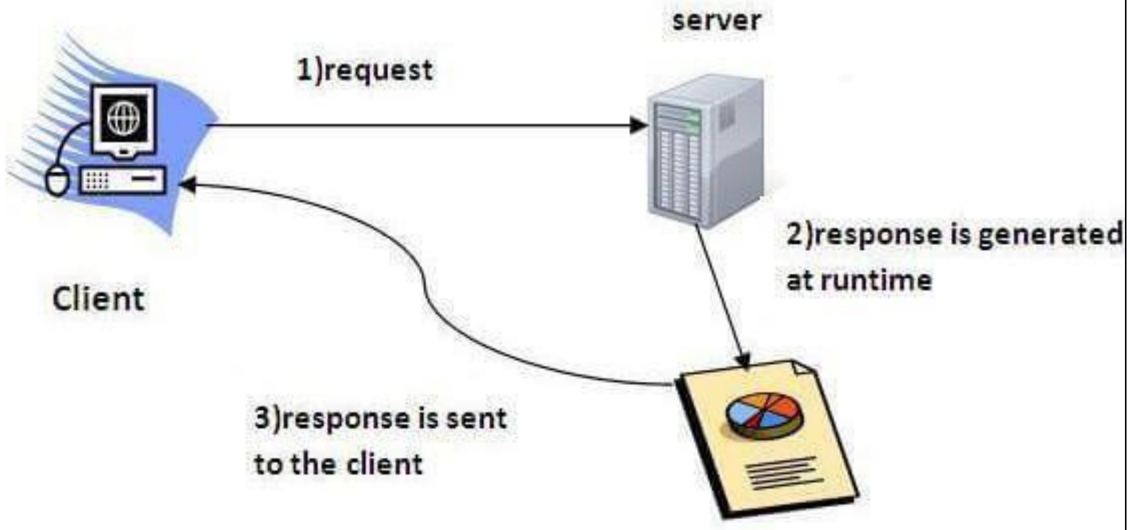
Nested JSON Objects

Values in a JSON object can be another JSON object.

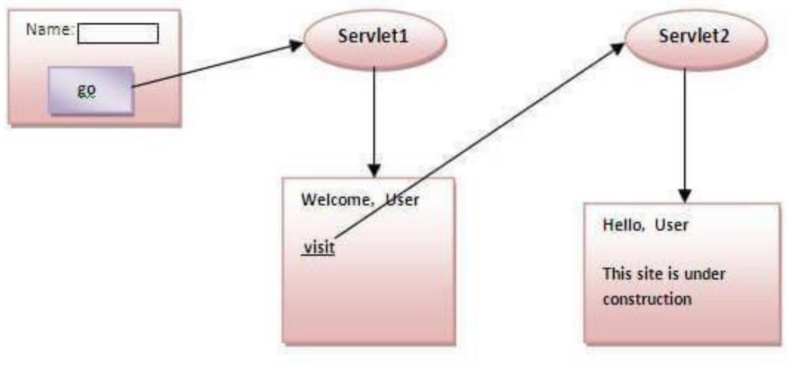
Example

```
myObj = {
  "name":"John",
```

	```
	 "age":30,
	 "cars": {
	  "car1":"Ford",
	  "car2":"BMW",
	  "car3":"Fiat"
	 }
	}
	```
	 You can access nested JSON objects by using the dot notation or bracket notation:
	
	Example
	```
	x = myObj.cars.car2;
	// or:
	x = myObj.cars["car2"];
	```
	 OUTPUT
	
	How to access nested JSON objects.
	
	BMW
	BMW

CS8651 Internet Programming – 2017Reg	
UNIT III - SERVER SIDE PROGRAMMING	
Servlets: Java Servlet Architecture – Servlet Life Cycle – Form GET and POST actions – Session Handling – Understanding Cookies – Installing and Configuring Apache Tomcat Web Server; DATABASE CONNECTIVITY: JDBC perspectives, JDBC program example; **JSP:** Understanding Java Server Pages – JSP Standard Tag Library (JSTL) – Creating HTML forms by embedding JSP code.	
PART-A	
Q.No	**Questions**
1.	**What** are servlets? A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes. ➢ Servlet is a technology which is used to create a web application. ➢ Servlet is an API that provides many interfaces and classes including documentation. ➢ Servlet is an interface that must be implemented for creating any Servlet. ➢ Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests. ➢ Servlet is a web component that is deployed on the server to create a dynamic web page.  server 1)request 2)response is generated at runtime Client 3)response is sent to the client
2.	**List** the application of servlets. Servlets may be used at different levels on a distributed framework. The following are some examples of servlet usage: ➢ To accept form input and generate HTML Web pages dynamically.

	➢ As part of middle tiers in enterprise networks by connecting to SQL databases via JDBC. ➢ In conjunction with applets to provide a high degree of interactivity and dynamic Web content generation. ➢ For collaborative applications such as online conferencing. ➢ A community of servlets could act as active agents which share data with each other. ➢ Servlets could be used for balancing load among servers which mirror the same content. ➢ Protocol support is one of the most viable uses for servlets. For example, a file service can start with NFS and move on to as many protocols as desired; the transfer between the protocols would be made transparent by servlets. Servlets could be used for tunneling over HTTP to provide chat, newsgroup or other file server functions.
3.	**Summarize** the advantages and disadvantages of servlets. **Servlet Advantage** ➢ Servlets provide a way to generate dynamic documents that is both easier to write and faster to run. ➢ Provide all the powerfull features of JAVA, such as Exception handling and garbage collection. ➢ Servlet enables easy portability across Web Servers. ➢ Servlet can communicate with different servlet and servers. ➢ Since all web applications are stateless protocol, servlet uses its own API to maintain session **Servlet Disadvantage** ➢ Designing in servlet is difficult and slows down the application. ➢ Writing complex business logic makes the application difficult to understand. ➢ You need a Java Runtime Environment on the server to run servlets. ➢ CGI is a completely language independent protocol, so you can write ➢ CGIs in whatever languages you have available (including Java if you want to).
4.	**Show** how is session tracking is achieved by the URL rewriting? **URL Rewriting** In URL rewriting, we append a token or identifier to the URL of the next Servlet or the next resource. We can send parameter name/value pairs using the following format: url?name1=value1&name2=value2&?? A name and a value is separated using an equal = sign, a parameter name/value pair is separated from another parameter using the ampersand(&). When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server. From a Servlet, we can use getParameter() method to obtain a parameter value.



Advantage of URL Rewriting

1. It will always work whether cookie is disabled or not (browser independent).
2. Extra form submission is not required on each pages.

Disadvantage of URL Rewriting

1. It will work only with links.
2. It can send Only textual information.

URL Rewriting

URL rewriting is another way to support anonymous session tracking. With URL rewriting, every local URL the user might click on is dynamically modified, or rewritten, to include extra information. The extra information can be in the form of extra path information, added parameters, or some custom, server-specific URL change.

Example 7-2 shows a revised version of our shopping cart viewer that uses URL rewriting in the form of extra path information to anonymously track a shopping cart.
Example 7-2. Session tracking using URL rewriting import java.io.*;

```
 import javax.servlet.*;
import javax.servlet.http.*;
public class ShoppingCartViewerRewrite extends HttpServlet { public void
doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException { res.setContentType("text/html");
PrintWriter out = res.getWriter(); out.println(""); out.println("");
// Get the current session ID, or generate one if necessary
 String sessionid = req.getPathInfo();
 if (sessionid == null)
{ sessionid = generateSessionId();
}
// Cart items are associated with the session ID
String[] items = getItemsFromCart(sessionid);
 // Print the current cart items.
out.println("You currently have the following items in your cart:
");
 if (items == null) { out.println("None"); } else { out.println("
          ");
          for (int i = 0; i < items.length; i++)
          { out.println("

     •    " + items[i]); } out.println("

"); }
// Ask if the user wants to add more items or check out. // Include the session
ID in the action URL.
 out.println("
");
out.println("Would you like to
");
out.println("\"); out.println("\"); out.println("
          ");

          // Offer a help page. Include the session ID in the URL. out.println("For
          help, click here");

          out.println(""); }

          private static String generateSessionId() { String uid = new
          java.rmi.server.UID().toString();

          // guaranteed unique return java.net.URLEncoder.encode(uid);

           // encode any special chars }

          private static String[] getItemsFromCart(String sessionid) {

          // Not implemented }

           }
```

This servlet first tries to retrieve the current session ID using getPathInfo() . If a session ID is not specified, it calls generateSessionId() to generate a new unique session ID using an RMI class designed specifically for this. The session ID is used to fetch and display the current items in the cart. The ID is then added to the form's ACTION attribute, so it can be retrieved by the ShoppingCart servlet. The session ID is also added to a new help URL that invokes the Help servlet. This wasn't possible with hidden form fields because the Help servlet isn't the target of a form submission. docstore.mik.ua/orelly/java-ent/servlet/ch07_03.htm

5.

Compare GET and POST request type.

1) In case of Get request, only **limited amount of data** can be sent because data is sent in header.	In case of post request, **large amount of data** can be sent because data is sent in body.
3) Get request **can be bookmarked.**	Post request **cannot be bookmarked.**
5) Get request is **more efficient** and used more than Post.	Post request is **less efficient** and used less than get.

6.

Summarize the servlet interface and its methods.

Servlet Interface

1. Servlet Interface
2. Methods of Servlet interface

- **Servlet interface provides** common behavior to all the servlets.
- Servlet interface defines methods that all servlets must implement.

- Servlet interface needs to be implemented for creating any servlet (either directly or indirectly).
- It provides 3 life cycle methods that are used to initialize the servlet, to service the requests, and to destroy the servlet and 2 non-life cycle methods.

Methods of Servlet interface

There are 5 methods in Servlet interface. The init, service and destroy are the life cycle methods of servlet. These are invoked by the web container.

Method	Description
public void init(ServletConfig config)	initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service(ServletRequest request,ServletResponse response)	provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	is invoked only once and indicates that servlet is being destroyed.
public ServletConfig getServletConfig()	returns the object of ServletConfig.
public String getServletInfo()	returns information about servlet such as writer, copyright, version etc.

Servlet Example by implementing Servlet interface

File: First.java

```
import java.io.*;
import javax.servlet.*;

public class First implements Servlet{
ServletConfig config=null;

public void init(ServletConfig config){
this.config=config;
System.out.println("servlet is initialized");
}
```

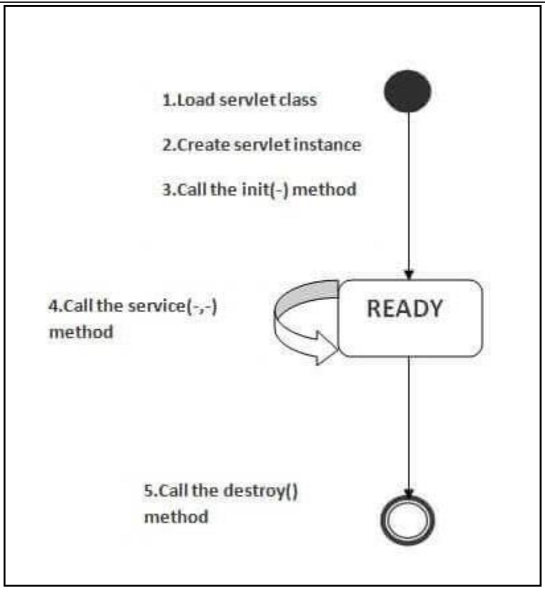
```

 public void service(ServletRequest req,ServletResponse res)
 throws IOException,ServletException{

res.setContentType("text/html");

 PrintWriter out=res.getWriter();
 out.print("<html><body>");
 out.print("<b>hello simple servlet</b>");
 out.print("</body></html>");

}
 public void destroy(){System.out.println("servlet is destroyed");}
 public ServletConfig getServletConfig(){return config;}
 public String getServletInfo(){return "copyright 2007-1010";}

}
```

7. **Sketch** the Servlet life cycle.

Life Cycle of a Servlet (Servlet Life Cycle)

The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

A. Life Cycle of a Servlet
   1. Servlet class is loaded
   2. Servlet instance is created
   3. init method is invoked
   4. service method is invoked
   5. destroy method is invoked


1.Load servlet class
2.Create servlet instance
3.Call the init(-) method
4.Call the service(-,-) method
READY
5.Call the destroy() method

As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.

8. **Show** the use of 'param' variable in JSP.

Jsp param

<jsp:param> tag is used to represent parameter value during jsp forward or include action this should be the sub tag of <jsp:forward> or <jsp:include>.

When an include or forward element is invoked, the original request object is provided to the target page. If you wish to provide additional data to that page, you can append parameters to the request object by using the jsp:param element.

Syntax

```
<jsp:param name=" " value=" "/>
```

Example

```
<jsp:include page="contact.jsp"/>
<jsp:param name="param1" value="value1"/>
</jsp:include>
```

Example

```
<jsp:forward page="home.jsp"/>
```

	`<jsp:param name="param1" value="value1"/>`
	`</jsp:forward>`

9.	**Quote** the uses of cookies. **<u>Cookies in Servlet</u>** A **cookie** is a small piece of information that is persisted between the multiple client requests. A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number. How Cookie works By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user. Types of Cookie There are 2 types of cookies in servlets. 1. Non-persistent cookie 2. Persistent cookie Non-persistent cookie It is **valid for single session** only. It is removed each time when user closes the browser. Persistent cookie It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout. **Advantage of Cookies** 1. Simplest technique of maintaining the state. 2. Cookies are maintained at client side. **Disadvantage of Cookies** 1. It will not work if cookie is disabled from the browser.

	2. Only textual information can be set in Cookie object.

10.	**Analyze** about java scriplet.
	In JavaServer Pages (JSP) technology, a **scriptlet** is a piece of **Java**-code embedded in the HTML-like JSP code. The **scriptlet** is everything inside the <% %> tags. Between these the user can add any valid **Scriptlet** i.e. any valid **Java** Code. In AppleScript, a **scriptlet** is a small script.
	
	JSP scriptlet tag
	
	A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:
	
	`<%  java source code %>`
	
	Example of JSP scriptlet tag
	
	In this example, we are displaying a welcome message.
	
	`<html>`
	`<body>`
	`<% out.print("welcome to jsp"); %>`
	`</body>`
	`</html>`

11.	**Express** appropriate java script code to remove an element (current element) from a DOM.
	
	A. Removing Existing HTML Elements
	
	To remove an HTML element, use the `remove()` method:
	
	1. Example
	
	`<div>`
	`  <p id="p1">This is a paragraph.</p>`
	`  <p id="p2">This is another paragraph.</p>`
	`</div>`
	
	`<script>`
	`var elmnt = document.getElementById("p1");`
	`elmnt.remove();`
	`</script>`

12.	**Define** JSP.

	JavaServer Pages (JSP) is a technology for developing Webpages that supports dynamic content. This helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>. A JavaServer Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands. Using JSP, you can collect input from users through Webpage forms, present records from a database or another source, and create Webpages dynamically. JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components, passing control between pages, and sharing information between requests, pages etc.
13.	**List** any three advantages of java servlet over JSP. 1. Being an extension to Java servlet, it can use every feature of Java Servlet. Also, custom tags can be used along with it. 2. There is no need to recompile JSP when changed. The changes automatically appear when run. 3. The tags which are used are easy to understand and write. 4. Supports Java API's which can now be easily used and integrated with the HTML code. 5. The results which are obtained are in HTML format, so can be opened on any browsers. 6. Customized JSP tags can be used. **Ex:** Tags with XML. 7. Changes can be added into the business logic page rather than changing in each and every page.
14.	**Rewrite** the code segment to store current server time in session using Java Servlet API. **Write a servlet application to print the current date and time.** **Answer:** The most important advantage of using Servlet is that we can use all the

15.

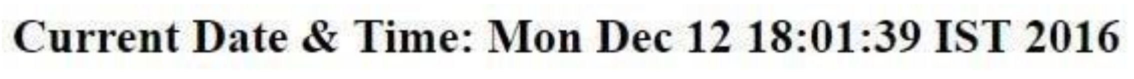
methods available in core java. The Date class is available in java.util package.

Below program shows how to print the current date and time. We can use simple Date object with toString() to print current date and time.

DateSrv.java

```
import java.io.*;
import javax.servlet.*;


public class DateSrv extends GenericServlet
{
    //implement service()
    public void service(ServletRequest req, ServletResponse res) throws
IOException, ServletException
    {
        //set response content type
        res.setContentType("text/html");
        //get stream obj
        PrintWriter pw = res.getWriter();
        //write req processing logic
        java.util.Date date = new java.util.Date();
        pw.println("<h2>"+"Current Date & Time: "
+date.toString()+"</h2>");
        //close stream object
        pw.close();
    }
}
```

Output:



```
Current Date & Time: Mon Dec 12 18:01:39 IST 2016
```

15.	**Compare** the difference between JSP and servlet.

Difference Between Servlet and JSP

In this article we will list some of the differences between Servlets and JSP.

SERVLET	**JSP**
A servlet is a server-side program and written purely on Java.	JSP is an interface on top of Servlets. In another way, we can say that JSPs are extension of servlets to minimize the effort of developers to write User Interfaces using Java programming.
Servlets run faster than JSP	JSP runs slower because it has the transition phase for converting from JSP page to a Servlet file. Once it is converted to a Servlet then it will start the compilation
Executes inside a Web server, such as Tomcat	A JSP program is compiled into a Java servlet before execution. Once it is compiled into a servlet, it's life cycle will be same as of servlet. But, JSP has it's own API for the lifecycle.
Receives HTTP requests from users and provides HTTP responses	Easier to write than servlets as it is similar to HTML.
We can not build any custom tags	One of the key advantage is we can build custom tags using JSP API (there is a separate

	package available for writing the custom t gs) which can be available as the re-usable components with lot of flexibility
Servlet advantages include: **1. Performance :** get loaded upon first request and remains in memory idenfinately. **2. Simplicity :** Run inside controlled server environment. No specific client software is needed:web broser is enough **3. Session Management** : overcomes HTTP's stateless nature **4. Java Technology :** network access,Database connectivity, j2ee integration	**JSP Provides an extensive infrastructu e for:** 1. Tracking sessions. 2. Managing cookies. 3. Reading and sending HTML headers. 4. Parsing and decoding HTML form data 5. **JSP is Efficient:** Every request for a JSP is handled by a simple Java thread 6. **JSP is Scalable:** Easy integration with other backend services 7. **Seperation of roles:** Developers, Cont nt Authors/Graphic Designers/Web Masters

II. Difference between Servlet and JSP

In brief, it can be defined as Servlet are the java programs that run on a Web server and act as a middle layer between a request coming from HTTP client and databases or applications on the HTTP server.While JSP is simply a text document that contains two types of text: static text which is predefined and dynamic text which is rendered after server response is received.

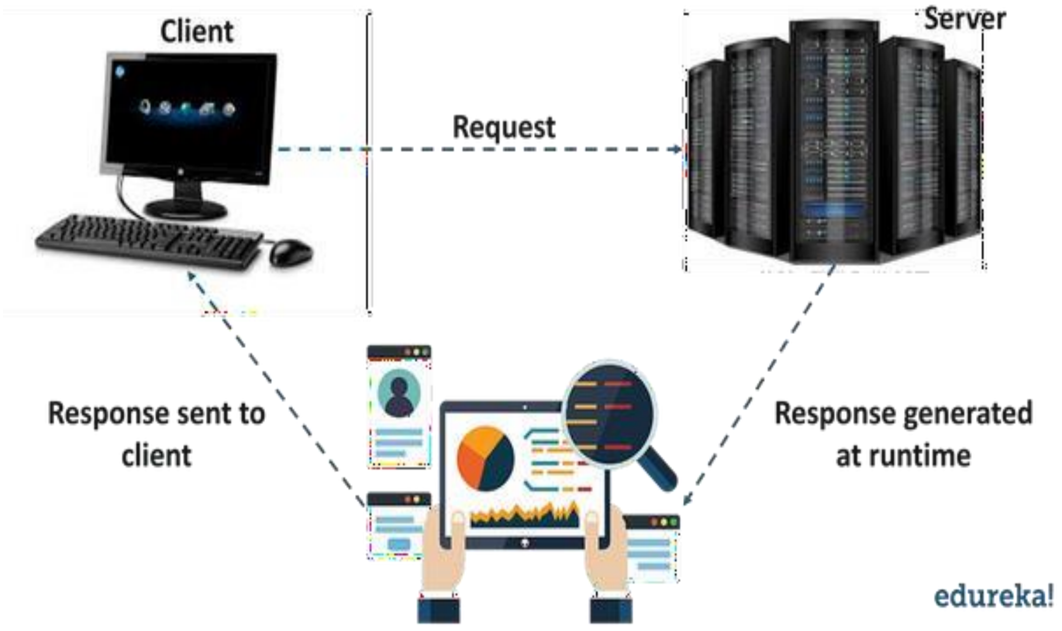
sr. No.	Key	Servlet	JSP
1	Implementation	Servlet is developed on Java language.	JSP is primarily written in HTML language although Java code could also be written o it but for it, JSTL or other language is required.
2	MVC	In contrast to MVC we can state servlet as a controller which receives the request process and send back the response.	On the other hand, JSP plays the role of view to render the response returned by th servlet.
3	Request type	Servlets can accept and process all type of protocol requests.	JSP on the other hand is compatible with HTTP request only.
4	Session Management	In Servlet by default session management is not enabled, the user has to enable it explicitly.	On the other hand in JSP session management is automatically enabled.
5	Performance	Servlet is faster than JSP.	JSP is slower than Servlet because first the translation of JSP to java code is taking place and then compiles.
6	Modification reflected	Modification in Servlet is a time-consuming task because it includes reloading, recompiling and restarting the server as we made any change in our code to get reflected.	On the other hands JSP modification is fas as just need to click the refresh button and code change would get reflected.

16.

Summarize briefly about the interaction between a webserver and a servlet.

How does a web server interact with a servlet?

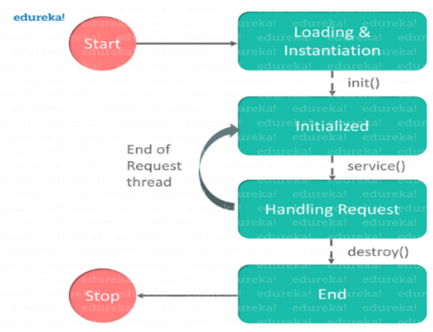
A servlet is a **Java Programming** language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. It is also a web component that is deployed on the server to create a dynamic web page.



In this figure you can see, a client sends a request to the server and the server generates the response, analyses it and sends the response to the client.

Stages of the Servlet Life Cycle: The Servlet life cycle mainly goes through four stages,

- Loading a Servlet.
- Initializing the Servlet.
- Request handling
- Destroying the Servlet.



Let’s look at each of these stages in details:

1. **Loading a Servlet**: The first stage of the Servlet life cycle involves loading and initializing the Servlet by the Servlet container. The Web container or Servlet Container can load the Servlet at either of the following two stages :Initializing the context, on configuring the Servlet with a zero or positive integer value.If the Servlet is not preceding the stage, it may delay the loading process until the Web container determines that this Servlet is needed to service a request.
2. **Initializing a Servlet**: After the Servlet is instantiated successfully, the Servlet container initializes the instantiated Servlet object. The container initializes the Servlet object by invoking the *init(ServletConfig)* method which accepts ServletConfig object reference as a parameter.
3. **Handling request**: After initialization, the Servlet instance is ready to serve the client requests. The Servlet container performs the following operations when the Servlet instance is located to service a request :It creates the **ServletRequest** and **ServletResponse.** In this case, if this is an HTTP request then the Web container creates **HttpServletRequest** and **HttpServletResponse** objects which are subtypes of the **ServletRequest** and **ServletResponse** objects respectively.
4. **Destroying a Servlet**: When a Servlet container decides to destroy the Servlet, it performs the following operations,It allows all the threads currently running in the service method of the Servlet instance to complete their jobs and get released.After currently running threads have completed their jobs, the Servlet container calls the **destroy()** method on the Servlet instance.

After the **destroy()** method is executed, the Servlet container releases all the references of this Servlet instance so that it becomes eligible for garbage collection.

17.

Define JDBC.

What is JDBC?

JDBC stands for **J**ava **D**ata**b**ase **C**onnectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

- Making a connection to a database.
- Creating SQL or MySQL statements.
- Executing SQL or MySQL queries in the database.
- Viewing & Modifying the resulting records.

Fundamentally, JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database. Java can be used to write different types of executables, such as −

- Java Applications
- Java Applets
- Java Servlets
- Java ServerPages (JSPs)
- Enterprise JavaBeans (EJBs).

All of these different executables are able to use a JDBC driver to access a database, and take advantage of the stored data.

JDBC provides the same capabilities as ODBC, allowing Java programs to contain database-independent code.

18. **Formulate** the three methods that are central to the life cycle of the servlet.

Servlets - Life Cycle

A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet.

- The servlet is initialized by calling the **init()** method.
- The servlet calls **service()** method to process a client's request.
- The servlet is terminated by calling the **destroy()** method.
- Finally, servlet is garbage collected by the garbage collector of the JVM.

Now let us discuss the life cycle methods in detail.

The init() Method

The init method is called only once. It is called only when the servlet is created, and not called for any user requests afterwards. So, it is used for one-time initializations, just as with the init method of applets.

The servlet is normally created when a user first invokes a URL corresponding to the servlet, but you can also specify that the servlet be loaded when the server is first started.

When a user invokes a servlet, a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate. The init() method simply creates or loads some data that will be used throughout the life of the servlet.

The init method definition looks like this −

```
public void init() throws ServletException {
   // Initialization code...
}
```

The service() Method

The service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.

Each time the server receives a request for a servlet, the server spawns a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

A. The destroy() Method

The destroy() method is called only once at the end of the life cycle of a servlet. This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.

After the destroy() method is called, the servlet object is marked for garbage collection. The destroy method definition looks like this −

```
public void destroy() {
   // Finalization code...
}
```

19. **Distinguish** between servlets and JSP.

Difference between Servlet and JSP

SERVLET	JSP
Servlet is a java code.	JSP is a html based code.
Writing code for servlet is harder than JSP as it is html in java.	JSP is easy to code as it is java in html.
Servlet plays a controller role in MVC approach.	JSP is the view in MVC approach for showing output.
Servlet is faster than JSP.	JSP is slower than Servlet because the first step i JSP lifecycle is the translation of JSP to java cod and then compile.
Servlet can accept all protocol requests.	JSP only accept http requests.
In Servlet, we can override the service() method.	In JSP, we cannot override its service() method.

In Servlet by default session management is not enabled, user have to enable it explicitly.	In JSP session management is automatically enabled.
In Servlet we have to implement everything like business logic and presentation logic in just one servlet file.	In JSP business logic is separated from presentati n logic by using javaBeans.
Modification in Servlet is a time consuming task because it includes reloading, recompiling and restarting the server.	JSP modification is fast, just need to click the refresh button.

20.

Discuss the need to use JSTL tags?

III. JSTL (JSP Standard Tag Library)

The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.

A. Advantage of JSTL

1. **Fast Development** JSTL provides many tags that simplify the JSP.
2. **Code Reusability** We can use the JSTL tags on various pages.
3. **No need to use scriptlet tag** It avoids the use of scriptlet tag.

B. JSTL Tags

There JSTL mainly provides five types of tags:

Tag Name	Description

Core tags	The JSTL core tag provide variable support, URL management, flow control, etc. The URL for the core tag is **http://java.sun.com/jsp/jstl/core**. The prefix of core tag is **c**.
Function tags	The functions tags provide support for string manipulation and string length. The URL for the functions tags is **http://java.sun.com/jsp/jstl/functions** and prefix is **fn**.
Formatting tags	The Formatting tags provide support for message formatting, numbe date formatting, etc. The URL for the Formatting tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt**.
XML tags	The XML tags provide flow control, transformation, etc. The URL for t XML tags is **http://java.sun.com/jsp/jstl/xml** and prefix is **x**.
SQL tags	The JSTL SQL tags provide SQL support. The URL for the SQL tags is **http://java.sun.com/jsp/jstl/sql** and prefix is **sql**.

PART-B

Q.No	Questions
1.	(i) **Integrate** how servlets work and its life cycle. (ii) Explain and **develop** the Servlet API.
2.	(i) **Analyze** a JavaScript to find factorial of a given number.

```
function factorial(x)
{

  if (x === 0)
 {
    return 1;
 }
  return x * factorial(x-1);

}
console.log(factorial(5));

OUTPUT : 120
```

(ii) **Differentiate** GET and POST method.

Get vs. Post

There are many differences between the Get and Post request. Let's see these differences:

GET	POST
1) In case of Get request, only **limited amount of data** can be sent because data is sent in header.	In case of post request, **large amount of data** can be sent because data is sent in body.
2) Get request is **not secured** because data is exposed in URL bar.	Post request is **secured** because data is not exposed in URL bar.
3) Get request **can be bookmarked.**	Post request **cannot be bookmarked.**
4) Get request is **idempotent** . It means second request will be ignored until response of first request is delivered	Post request is **non-idempotent.**
5) Get request is **more efficient** and used more than Post.	Post request is **less efficient** and used less than get.

GET and POST

Two common methods for the request-response between a server and client are:

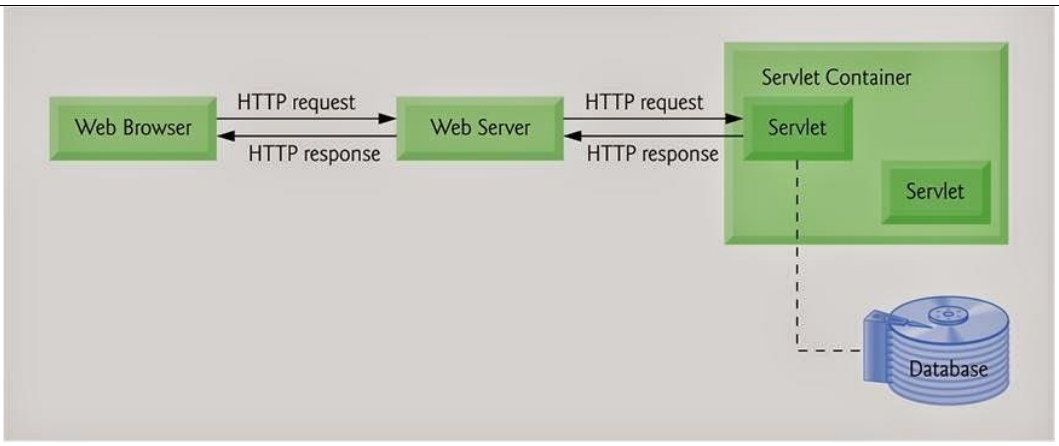
- **GET**- It requests the data from a specified resource
- **POST**- It submits the processed data to a specified resource

3.

Demonstrate the Servlet architecture and explain its working principle.

Servlet Architecture

4.



Servlets read the explicit data sent by the clients (browsers). This includes an HTML form on a Web page or it could also come from an applet or a custom HTTP client program.

Read the implicit HTTP request data sent by the clients (browsers). This includes cookies, media types and compression schemes the browser understands, and so forth.

Process the data and generate the results. This process may require talking to a database, executing an RMI or CORBA call, invoking a Web service, or computing the response directly.

Send the explicit data (i.e., the document) to the clients (browsers). This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), Excel, etc.

Send the implicit HTTP response to the clients (browsers). This includes telling the browsers or other clients what type of document is being returned (e.g., HTML), setting cookies and caching parameters, and other such tasks.

Servlet API:

Servelt API contains three packages

javax.servlet: Package contains a number of classes and interfaces that describe the contract
between a servlet class and the runtime environment provided for an instance of such a class a
conforming servelt container.
javax.servlet.aanotation: Package contains a number of annotations that allow users to use
annotations to declare servlets , filters, listeners and specify the metadata for the declared component
javax.servlet.http: Package contains a number of classes and interfaces that describe and define the contract between a servlet class rnning under the HTTP protocal and the runtime environment provided for an instance of such class by a confirming servlet container.

4.	**Consider** a database that has a table Employee with two columns Employee Id and Name. Assume that the administrator user id and password to access to access the database table are Scott and Tiger. Write a JDBC program that can query and print all entries in the

	table employee. Make the database using type 2 driver database.driver and connection string jdbc :db.oci.
5.	**Describe** in detail the session handling in server side programming.

IV. Managing Session in Servlets

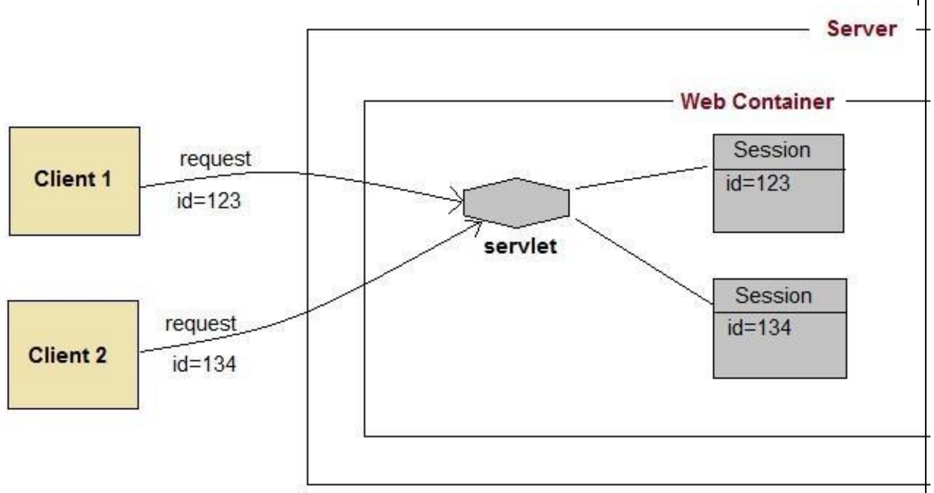
We all know that **HTTP** is a stateless protocol. All requests and responses are independent. But sometimes you need to keep track of client's activity across multiple requests. For eg. When a User logs into your website, not matter on which web page he visits after logging in, his credentials will be with the server, until he logs out. So this is managed by creating a session.

Session Management is a mechanism used by the **Web container** to store session information for a particular user. There are four different techniques used by Servlet application for session management. They are as follows:

1. **Cookies**
2. **Hidden form field**
3. **URL Rewriting**
4. **HttpSession**

Session is used to store everything that we can get from the client from all the requests the client makes.

A. How Session Works



B.

he basic concept behind session is, whenever a user starts using our application, we can save a unique identification information about him, in an object which is available throughout the application, until its destroyed. So wherever the user goes, we will always have his information and we can always manage which user is doing what. Whenever a user wants to exit from your application, destroy the object with his information.

6.

(i) **Discuss** about JSTL.

V. JSTL (JSP Standard Tag Library)

The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.

A. Advantage of JSTL

1. **Fast Development** JSTL provides many tags that simplify the JSP.
2. **Code Reusability** We can use the JSTL tags on various pages.
3. **No need to use scriptlet tag** It avoids the use of scriptlet tag.

B. JSTL Tags

There JSTL mainly provides five types of tags:

Tag Name	Description
Core tags	The JSTL core tag provide variable support, URL management, flow etc. The URL for the core tag is **http://java.sun.com/jsp/js l/c** prefix of core tag is **c**.
Function tags	The functions tags provide support for string manipulation and strin The URL for the functions tags is **http://java.sun.com/jsp/jstl/functions** and prefix is **fn**.
Formatting tags	The Formatting tags provide support for message formatting, numb date formatting, etc. The URL for the Formatting tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt**.
XML tags	The XML tags provide flow control, transformation, etc. The UI L for tags is **http://java.sun.com/jsp/jstl/xml** and prefix is **x**.
SQL tags	The JSTL SQL tags provide SQL support. The URL for the SQL tags is **http://java.sun.com/jsp/jstl/sql** and prefix is **sql**.

(ii) **Summarize** a client server JSP program to find simple interest and display the result in client.

7.

Explain the use of cookies for tracking for tracking requests with a program.

Session Tracking in JSP

Session Tracking :

HTTP is a "stateless" protocol which means each time a client retrieves a Web page, the client opens a new connection to the Web server and the server does not keep any record of previous client request.Session tracking is a mechanism that is used to maintain state about a series of requests from the same user(requests originating from the same browser) across some period of time. A session id is a unique token number assigned to a specific user for the duration of that user's session.

Need Of Session Tracking :

HTTP is a stateless protocol so When there is a series of continuous request and response from a same client to a server, the server cannot identify which client is sending request.If we want to maintain the conversational state,session tracking is needed. For example, in a shopping cart application a client keeps on adding items into his cart using multiple requests.When every request is made,the server should identify in which client's cart the item is to be added. So in this scenario, there is a certain need for session tracking.

Solution is, when a client makes a request it should introduce itself by providing unique identifier every time.There are four ways to maintain session between web client and web server.

Methods to track session :

Cookies
URL Rewriting
Hidden Fields
Session API

Cookies :

Cookies mostly used for session tracking. Cookie is a key value pair of information, sent by the server to the browser. This should be saved by the browser in its space in the client computer.

Whenever the browser sends a request to that server it sends the cookie along with it. Then the server can identify the client using the cookie.

This is not an effective way because many time browser does not support a cookie or users can opt to disable cookies using their browser preferences. In such case, the browser will not save the cookie at client computer and session tracking fails.

8.

(i) **Explain** about the standard actions in JSP.

Actions are used for controlling the behavior of servlet engine.

How many standard Action Tags are available in JSP?

There are 11 types of Standard Action Tags as following:

- jsp:useBean
- jsp:include
- jsp:setProperty
- jsp:getProperty
- jsp:forward
- jsp:plugin
- jsp:attribute
- jsp:body
- jsp:text
- jsp:param
- jsp:attribute
- jsp:output

(ii) **Analyze** MVC architecture of JSP.

VI. MVC in JSP

1. MVC in JSP
2. Example of following MVC in JSP

MVC stands for Model View and Controller. It is a **design pattern** that separates the business logic, presentation logic and data.

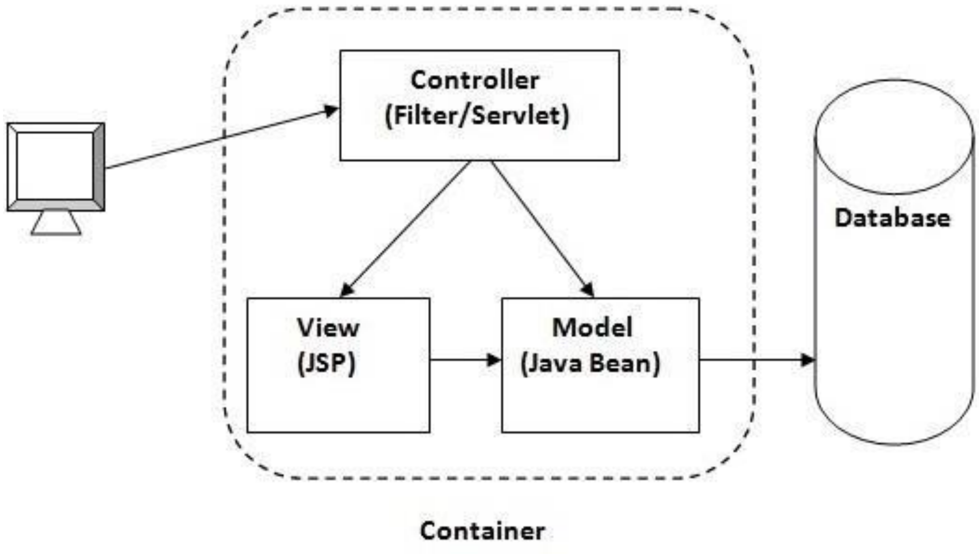
Controller acts as an interface between View and Model. Controller intercepts all the incoming requests.

Model represents the state of the application i.e. data. It can also have business logic.

View represents the presentaion i.e. UI(User Interface).

a) *Advantage of MVC (Model 2) Architecture*

1. Navigation Control is centralized
2. Easy to maintain the large application



9. **Explain** in detail about Servlet database connectivity with an example of student database.

Example of Registration form in servlet

Table creation :

```
CREATE TABLE "REGISTERUSER"
  (    "NAME" VARCHAR2(4000),
  "PASS" VARCHAR2(4000),
  "EMAIL" VARCHAR2(4000),
  "COUNTRY" VARCHAR2(4000)
  )
/
```

Example of Registration form in servlet

In this example, we have created the three pages.

- register.html
- Register.java

- web.xml

---

register.html

In this page, we have getting input from the user using text fields and combobox. The information entered by the user is forwarded to Register servlet, which is responsible to store the data into the database.

```
<html>
<body>
<form action="servlet/Register" method="post">

Name:<input type="text" name="userName"/><br/><br/>
Password:<input type="password" name="userPass"/><br/><br/>
Email Id:<input type="text" name="userEmail"/><br/><br/>
Country:
<select name="userCountry">
. <option>India</option>
. <option>Pakistan</option>
. <option>other</option>
. </select>
.
. <br/><br/>
. <input  type="submit"  value="register"/>
.
. </form>
. </body>
.  </html>
```

Register.java

This servlet class receives all the data entered by user and stores it into the database. Here, we are performing the database logic. But you may separate it, which will be better for the web application.

```
import java.io.*;
import java.sql.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;

public class Register extends HttpServlet {
public void doPost(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {

.   response.setContentType("text/html");
. PrintWriter out = response.getWriter();
.
. String n=request.getParameter("userName");
. String p=request.getParameter("userPass");
. String e=request.getParameter("userEmail");
. String c=request.getParameter("userCountry");
.
. try{
.   Class.forName("oracle.jdbc.driver.OracleDriver");
. Connection con=DriverManager.getConnection(
.   "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
.
. PreparedStatement ps=con.prepareStatement(
. "insert into registeruser values(?,?,?,?)");
.
.  ps.setString(1,n);
.  ps.setString(2,p);
.  ps.setString(3,e);
.  ps.setString(4,c);
.
. int i=ps.executeUpdate();
. if(i>0)
. out.print("You are successfully registered...");
.
.
. }catch (Exception e2) {System.out.println(e2);}
.
. out.close();
. }
.
. }
```

10.

Demonstrate the procedure of installing and configuring Apache Tomcat.

How To Install Apache Tomcat 8 on Ubuntu 16.04

Apache Tomcat is a web server and servlet container that is used to serve Java applications. Tomcat is an open source implementation of the Java Servlet and JavaServer Pages technologies, released by the Apache Software Foundation. This tutorial covers the basic installation and some configuration of the latest release of Tomcat 8 on your Ubuntu 16.04 server.

Step 1: Install Java

Tomcat requires Java to be installed on the server so that any Java web application code can be executed. We can satisfy that requirement by installing OpenJDK with apt-get.

First, update your apt-get package index:

```
sudo apt-get update
```

Then install the Java Development Kit package with apt-get:

```
sudo apt-get install default-jdk
```

Now that Java is installed, we can create a `tomcat` user, which will be used to run the Tomcat service.

Step 2: Create Tomcat User

For security purposes, Tomcat should be run as an unprivileged user (i.e. not root). We will create a new user and group that will run the Tomcat service.

First, create a new `tomcat` group:

```
sudo groupadd tomcat
```

Next, create a new `tomcat` user. We'll make this user a member of the `tomcat` group, with a home directory of `/opt/tomcat` (where we will install Tomcat), and with a shell of `/bin/false` (so nobody can log into the account):

```
sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat
```

Now that our `tomcat` user is set up, let's download and install Tomcat.

Step 3: Install Tomcat

The best way to install Tomcat 8 is to download the latest binary release then configure it manually.

We will install Tomcat to the `/opt/tomcat` directory. Create the directory, then extract the archive to it with these commands:

```
sudo mkdir /opt/tomcat

sudo tar xzvf apache-tomcat-8*tar.gz -C /opt/tomcat --strip-
components=1
```

Next, we can set up the proper user permissions for our installation.

Step 4: Update Permissions

The `tomcat` user that we set up needs to have access to the Tomcat installation. We'll set that up now.

Change to the directory where we unpacked the Tomcat installation:

```
cd /opt/tomcat
```

Give the `tomcat` group ownership over the entire installation directory:

```
sudo chgrp -R tomcat /opt/tomcat
```

Next, give the `tomcat` group read access to the `conf` directory and all of its contents, and **execute** access to the directory itself:

```
sudo chmod -R g+r conf
sudo chmod g+x conf
```

Make the `tomcat` user the owner of the `webapps`, `work`, `temp`, and `logs` directories:

```
sudo chown -R tomcat webapps/ work/ temp/ logs/
```

Now that the proper permissions are set up, we can create a systemd service file to manage the Tomcat process.

Step 5: Create a systemd Service File

We want to be able to run Tomcat as a service, so we will set up systemd service file.

Tomcat needs to know where Java is installed. This path is commonly referred to as "JAVA_HOME". The easiest way to look up that location is by running this command:

Step 6: Adjust the Firewall and Test the Tomcat Server

Now that the Tomcat service is started, we can test to make sure the default page is available.

Before we do that, we need to adjust the firewall to allow our requests to get to the service. If you followed the prerequisites, you will have a `ufw` firewall enabled currently.

Tomcat uses port `8080` to accept conventional requests. Allow traffic to that port by typing:

```
sudo ufw allow 8080
```

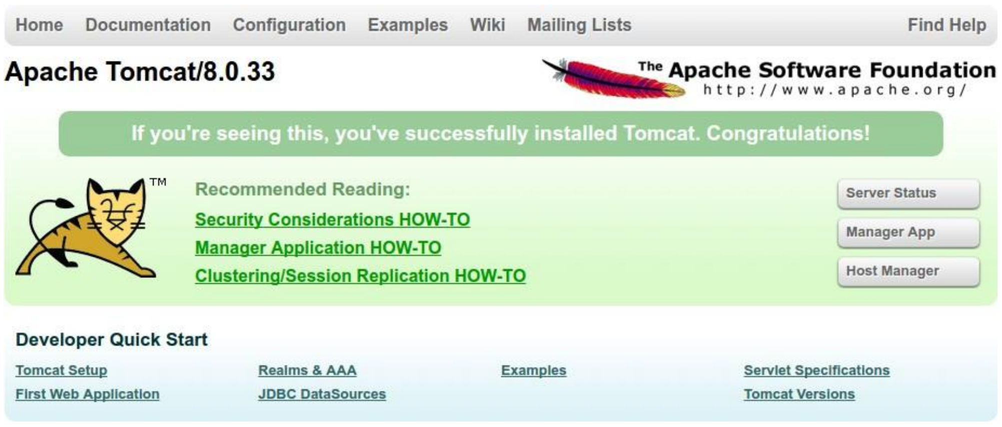
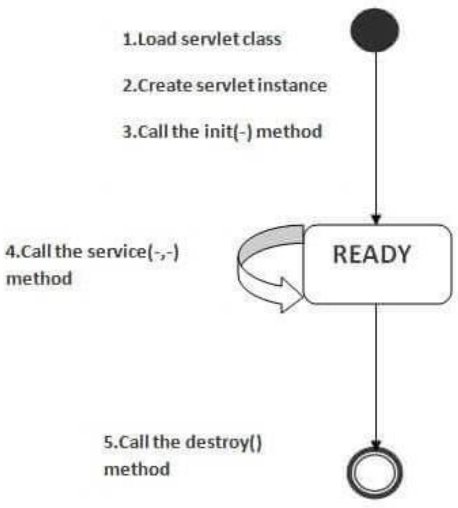
Step 7: Configure Tomcat Web Management Interface

Step 8: Access the Web Interface

Now that we have create a user, we can access the web management interface again in a web browser. Once again, you can get to the correct interface by entering your server's domain name or IP address followed on port 8080 in your browser:

```
Open in web browser
http://server_domain_or_IP:8080
```

The page you see should be the same one you were given when you tested earlier:

	 Home Documentation Configuration Examples Wiki Mailing Lists Find Help Apache Tomcat/8.0.33 The Apache Software Foundation http://www.apache.org/ If you're seeing this, you've successfully installed Tomcat. Congratulations! Recommended Reading: Security Considerations HOW-TO Manager Application HOW-TO Clustering/Session Replication HOW-TO Server Status Manager App Host Manager Developer Quick Start Tomcat Setup First Web Application Realms & AAA JDBC DataSources Examples Servlet Specifications Tomcat Versions Your installation of Tomcat is complete! Your are now free to deploy your own Java web applications!
11.	(i) **Discuss** about the Servlet life cycle. **VII. Life Cycle of a Servlet (Servlet Life Cycle)** The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet: 1. Servlet class is loaded. 2. Servlet instance is created. 3. init method is invoked. 4. service method is invoked. 5. destroy method is invoked.  1.Load servlet class 2.Create servlet instance 3.Call the init(-) method 4.Call the service(-,-) method READY 5.Call the destroy() method

As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.

(ii) **List** JSP advantages.

1. Advantages of JSP over Servlet

There are many advantages of JSP over the Servlet. They are as follows:

a) *1) Extension to Servlet*

JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

b) *2) Easy to maintain*

JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.

c) *3) Fast Development: No need to recompile and redeploy*

If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

d) *4) Less code than Servlet*

In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

12.

(i) **Explain** and write a simple JDBC program.

Java Database Connectivity with MySQL

To connect Java application with the MySQL database, we need to follow 5 following steps.

1. **Driver class:** The driver class for the mysql database is **com.mysql.jdbc.Driver**.
2. **Connection URL:** The connection URL for the mysql database is **jdbc:mysql://localhost:3306/sonoo** where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we

may also use IP address, 3306 is the port number and sonoo is the database name. We may use any database, in such case, we need to replace the sonoo with our database name.

3. **Username:** The default username for the mysql database is **root**.
4. **Password:** It is the password given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

Let's first create a table in the mysql database, but before creating table, we need to create database first.

```
create database sonoo;
use sonoo;
create table emp(id int(10),name varchar(40),age int(3));
```

Example to Connect Java Application with mysql database

In this example, sonoo is the database name, root is the username and password both.

```
import java.sql.*;
class MysqlCon{
public static void main(String args[]){
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/sonoo","root","root");
//here sonoo is database name, root is username and password
Statement stmt=con.createStatement();
. ResultSet  rs=stmt.executeQuery("select  *  from  emp");
. while(rs.next())
. System.out.println(rs.getInt(1)+"    "+rs.getString(2)+"    "+rs.getString(3));
. con.close();
. }catch(Exception  e){  System.out.println(e);}
. }
. }
```

(ii) **List** various JSP scripting components.

VIII. JSP Scripting Element

JSP Scripting element are written inside <% %> tags. These code inside <% %> tags are processed by the JSP engine during translation of the JSP page. Any other text in the JSP page is considered as HTML code or plain text.

Example:

```
 html>
  <head>
```

	```
	    <title>My First JSP Page</title>
	  </head>
	  <%
	    int count = 0;
	  %>
	  <body>
	    Page Count is <% out.println(++count); %>
	  </body>
	</html>
	```
	
	### 1. Types of scripting elements
	
	(table below)
	
	### B. JSP Comment
	
	JSP Comment is used when you are creating a JSP page and want to put in comments about what you are doing. JSP comments are only seen in the JSP page. These comments are not included in servlet source code during translation phase, nor they appear in the HTTP response. Syntax of JSP comment is as follows :
	
	`<%-- JSP  comment --%>`
	
	Simple Example of JSP Comment
	```
	<html>
	  <head>
	    <title>My First JSP Page</title>
	  </head>
	  <%
	    int count = 0;
	  %>
	  <body>
	    <%-- Code to show page count  --%>
	    Page Count is <% out.println(++count); %>
	  </body>
	</html>
	```

13.	(i) **Demonstrate** with suitable example for core and formatting tags in JSTL.

Comment	<%-- comment --%>
Declaration	<%! declarations %>
Expression	<%= expression %>

JSTL Formatting tags

The formatting tags provide support for message formatting, number and date formatting etc. The url for the formatting tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt.**

The JSTL formatting tags are used for internationalized web sites to display and format text, the time, the date and numbers. The syntax used for including JSTL formatting library in your JSP is:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"  %>
```

Formatting Tags	Descriptions
fmt:parseNumber	It is used to Parses the string representation of a currency, percentage or number.
fmt:timeZone	It specifies a parsing action nested in its body or the time zone for any time formatting.
fmt:formatNumber	It is used to format the numerical value with specific format or precision.
fmt:parseDate	It parses the string representation of a time and date.
fmt:bundle	It is used for creating the ResourceBundle objects which will be used by their tag body.
fmt:setTimeZone	It stores the time zone inside a time zone configuration variable.
fmt:setBundle	It loads the resource bundle and stores it in a bundle configuration variable or the named scoped variable.
fmt:message	It display an internationalized message.
fmt:formatDate	It formats the time and/or date using the supplied pattern and styles.

JSTL Core <c:choose>, <c:when>, <c:otherwise> Tag

The < c:choose > tag is a conditional tag that establish a context for mutually exclusive conditional operations. It works like a Java **switch** statement in which we choose between a numbers of alternatives.

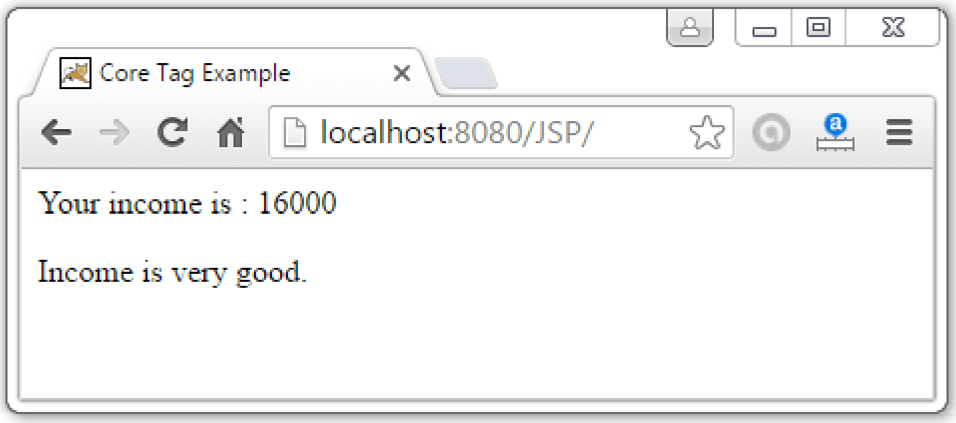
Example

Let's see the simple example of < c:choose >, < c:when > < c:otherwise > tag:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:set var="income" scope="session" value="${4000*4}"/>
<p>Your income is : <c:out value="${income}"/></p>
<c:choose>
.    <c:when test="${income <= 1000}">
.        Income is not good.
.    </c:when>
.    <c:when test="${income > 10000}">
.         Income is very good.
.    </c:when>
.    <c:otherwise>
.        Income is undetermined...
.    </c:otherwise>
. </c:choose>
. </body>
. </html>
```

This will produce the following result:

```
Your income is : 16000
Income is very good.
```



Your income is : 16000

Income is very good.

(ii) **Demonstrate** with suitable example for SQL and XML tags in JSTL.

JSTL SQL

The **<sql:setDataSource>** tag sets the data source configuration variable or saves the data-source information in a scoped variable that can be used as input to the other JSTL database actions.

Attribute

The **<sql:setDataSource>** tag has the following attributes −

Attribute	Description
driver	Name of the JDBC driver class to be registered
url	JDBC URL for the database connection
user	Database username
password	Database password
password	Database password
dataSource	Database prepared in advance
var	Name of the variable to represent the database
scope	Scope of the variable to represent the database

Example

Consider the following information about your MySQL database setup −

- We are using **JDBC MySQL** driver.
- We are going to connect to TEST database on local machine.
- We would use **user_id** and **mypassword** to access TEST database.

All the above parameters will vary based on your MySQL or any other database setup. Considering the above parameters, following example uses the **setDataSource** tag −

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>

<%@ taglib uri = "http://java.sun.com/jsp/jstl/sql" prefix = "sql"%>
```

```
<html>
  <head>
    <title>JSTL sql:setDataSource Tag</title>
  </head>

  <body>
    <sql:setDataSource var = "snapshot" driver = "com.mysql.jdbc.Driver"
      url = "jdbc:mysql://localhost/TEST"
      user = "user_id"  password = "mypassword"/>
    <sql:query dataSource = "${snapshot}" sql = "..." var = "result" />

  </body>
</html>
```

JSTL XML

<x:parse> Tag

The <x:parse> tag is used for parse the XML data specified either in the tag body or an attribute. It is used for parse the xml content and the result will stored inside specified variable.

The syntax used for including the <x:parse> tag is:

```
<x:parse attributes> body content </x:parse>
```

EXAMPLE

Let us put the following content in **novels.xml** file:

```
<books>
<book>
  <name>Three mistakes of my life</name>
  <author>Chetan Bhagat</author>
  <price>200</price>
</book>
<book>
  <name>Tomorrow land</name>
  <author>NUHA</author>
  <price>2000</price>
```

```
</book>
</books>
```

index.jsp

(IN the same directory)

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<html>
<head>
  <title>x:parse Tag</title>
</head>
<body>
<h2>Books Info:</h2>
<c:import var="bookInfo" url="novels.xml"/>

<x:parse xml="${bookInfo}" var="output"/>
<p>First Book title: <x:out select="$output/books/book[1]/name" /></p>
<p>First Book price: <x:out select="$output/books/book[1]/price" /></p>
<p>Second Book title: <x:out select="$output/books/book[2]/name" /></p>
<p>Second Book price: <x:out select="$output/books/book[2]/price" /></p>

</body>
</html>
```

Output:

```
Books Info:

First Book title: Three mistakes of my life

First Book price: 200

Second Book title: Tomorrow land

Second Book price: 2000
```

14. **Define** HTML and JSP. Use the same and design a scientific calculator.

Calculator.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
        <head>
                <title>Calculator</title>
```

```
		<style>
			h1 {
				font-family: Arial;
				font-size: 14pt;
				font-weight: normal;
			}
			td input {
				font-family: Arial;
				font-size: 10pt;
				width: 30px;
			}

			input.double {
				width: 60px;
			}

			input.doubleheight {
				height: 48px;
			}

			input.display {
				border: 1px solid black;
				readonly: true;
				width: 120px;
				padding: 2px;
			}
		</style>
	</head>

	<body>
		<h1>Calculator</h1>

		<form method="GET" action="calculate.html">
		<table border="0" cellpadding="0" cellspacing="0">
		<tr>
			<td colspan="4">
				<input class="display" type="text" id="display" value="<c:out
value="${displayAmount}"/>" readonly/>
			</td>
		</tr>
		<tr>
			<td><input type="submit" name="button" id="btn-7"
value="7"/></td>
			<td><input type="submit" name="button" id="btn-8"
value="8"/></td>
			<td><input type="submit" name="button" id="btn-9"
value="9"/></td>
			<td><input type="submit" name="button" id="btn-/"
value="/"/></td>
```

```
                              <td><input type="submit" name="button" id="btn-C"
value="C"/></td>
                    </tr>
                    <tr>
                              <td><input type="submit" name="button" id="btn-4"
value="4"/></td>
                              <td><input type="submit" name="button" id="btn-5"
value="5"/></td>
                              <td><input type="submit" name="button" id="btn-6"
value="6"/></td>
                              <td><input type="submit" name="button" id="btn-*"
value="*"/></td>
                              <td></td>
                    </tr>
                    <tr>
                              <td><input type="submit" name="button" id="btn-1"
value="1"/></td>
                              <td><input type="submit" name="button" id="btn-2"
value="2"/></td>
                              <td><input type="submit" name="button" id="btn-3"
value="3"/></td>
                              <td><input type="submit" name="button" id="btn--" value="-
"/></td>
                              <td rowspan="2"><input class="doubleheight" type="submit"
name="button" id="btn-=" value="="/></td>
                    </tr>
                    <tr>
                              <td colspan="2"><input class="double" type="submit"
name="button" id="btn-0" value="0"/></td>
                              <td><input type="button" name="button" id="btn-."
value="."/></td>
                              <td><input type="submit" name="button" id="btn-+"
value="+"/></td>
                    </tr>
                    </table>
                    </form>
          </body>
</html>
```

PART – C

Q.No	**Questions**
1.	**Design** a HTML forms by embedding JSP code for submission of a resume to a job portal website with appropriate database connectivity.
2.	**Evaluate** a complete query application for books database using JDBC.
3.	**Write** a program that allows the user to **select** a favourite programming language and post the choice to the server. The response is a web page in which the user can click a

	link to view a list of book recommendations. The cookies previously stored on the client are read by the servlet and form a web page containing the book recommendation. Use servlet cookies and HTML.
4.	**Develop** a JSP program to display the grade of a student by accepting the marks of five subjects.

CS8651 Internet Programming – 2017Reg	
PHP: An introduction to PHP – Using PHP – Variables – Program control – Built-in functions – Form Validation – Regular Expressions – File handling – Cookies – Connecting to Database; **XML:** Basic XML – Document Type Definition – XML Schema DOM and Presenting XML, XML Parsers and Validation, XSL and XSLT Transformation, News Feed (RSS and ATOM).	
Internet Programming – UNIT-IV	
Q.No	**Questions**
1.	**Define** PHP. List the features.

What is PHP?

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

PHP is an amazing and popular language!

It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)!
It is deep enough to run the largest social network (Facebook)!
It is also easy enough to be a beginner's first server side language!

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension "`.php`"

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

2. **List** the rules for creating variables in PHP.

PHP Variables

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

- A variable starts with the `$` sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (`$age` and `$AGE` are two different variables)
- PHP variable names are case-sensitive

3. **Illustrate** a PHP program to determine the type of browser that a web client is using.

Display the Browser – PHP Script

The following PHP function can be used to display the browser:

```
<?php

function get_the_browser()
{

if(strpos($_SERVER['HTTP_USER_AGENT'], 'MSIE') !== false)
return 'Internet explorer';
elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Trident') !== false)
return 'Internet explorer';
elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Firefox') !== false)
return 'Mozilla Firefox';
elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Chrome') !== false)
return 'Google Chrome';
elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Opera Mini') !== false)
return "Opera Mini";
elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Opera') !== false)
return "Opera";
elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Safari') !== false)
return "Safari";
else
return 'Other';
}

?>
```

In the above code, we are checking each possible browser that may be and return the browser name. Here we haven’t checked the Mozilla because of most of the browser using this as the user agent string.

Below is how to display the browser name on our web page:

```
Echo get_the_browser();
```

4. **Name** any four built-in functions in PHP.

PHP Reference

The PHP reference contains different categories of all PHP functions and constants, along with examples.

Array	Calendar	Date	Directory	Error	File
system	Filter	FTP	Libxml	Mail	Math
Misc	MySQLi	Network	SimpleXML	Stream	
String	XML Parser	Zip	Timezones		

5. **Infer** when should the super global arrays in PHP be used?

Superglobals were introduced in PHP 4.1.0, and are built-in variables that are always available in all scopes.

PHP Global Variables - Superglobals

Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

Which super global array in PHP would contain a HTML form's POST data?

PHP Superglobal - $_POST

Super global variables are built-in variables that are always available in all scopes.

PHP $_POST

PHP $_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to the file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_POST to collect the value of the input field:

Example

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>
```

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_POST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>

</body>
</html>
```

6.

Classify the difference between echo() & print() functions.

PHP echo and print Statements

With PHP, there are two basic ways to get output: echo and print.

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small:

The PHP echo Statement	**The PHP print Statement**
➢ echo has no return value ➢ echo can take multiple parameters ➢ echo is marginally faster than print.	➢ print has a return value of 1 so it can be used in expressions. ➢ print can take one argument
The echo statement can be used with or without parentheses: echo or echo().	The print statement can be used with or without parentheses: print or print().
Example-1 <?php echo "<h2>PHP is Fun!</h2>"; echo "Hello world! "; echo "I'm about to learn PHP! "; echo "This ", "string ", "was ", "made ", "with multiple parameters."; ?>	Example-1 <?php print "<h2>PHP is Fun!</h2>"; print "Hello world! "; print "I'm about to learn PHP!"; ?>
PHP is Fun! Hello world!	**PHP is Fun!** Hello world!

	Example-2 `<?php` `$txt1 = "Learn PHP";` `$txt2 = "W3Schools.com";` `$x = 5;` `$y = 4;` `echo "<h2>" . $txt1 . "</h2>";` `echo "Study PHP at " . $txt2 . " ";` `echo $x + $y;` `?>`	Example-2 `<?php` `$txt1 = "Learn PHP";` `$txt2 = "W3Schools.com";` `$x = 5;` `$y = 4;` `print "<h2>" . $txt1 . "</h2>";` `print "Study PHP at " . $txt2 . " ";` `print $x + $y;` `?>`
	Learn PHP Study PHP at W3Schools.com 9	**Learn PHP** Study PHP at W3Schools.com 9

7. **List** any two advantages of XML document.

Using XML to exchange information offers many benefits.
Advantages of XML include the following:

- XML uses human, not computer, language. XML is readable and understandable, even by novices, and no more difficult to code than HTML.
- XML is completely compatible with Java™ and 100% portable. Any application that can process XML can use your information, regardless of platform.
- XML is extendable. Create your own tags, or use tags created by others, that use the natural language of your domain, that have the attributes you need, and that makes sense to you and your users.

8. **Give** the difference between DTD and XML schema for defining XML document structure with appropriate examples.

DTD vs XSD

There are many differences between DTD (Document Type Definition) and XSD (XML Schema Definition). In short, DTD provides less control on XML structure whereas XSD (XML schema) provides more control.

The important differences are given below:

No.	**DTD**(Document Type Definition)	**XSD**(XML Schema Definition)
1)	DTD stands for **Document Type Definition**.	XSD stands for XML Schema Definition.
2)	DTDs are derived from **SGML** syntax.	XSDs are written in XML.

3)	DTD **doesn't support datatypes**.	XSD **supports datatypes** for elements and attributes.
4)	DTD **doesn't support namespace**.	XSD **supports namespace**.
5)	DTD **doesn't define order** for child elements.	XSD **defines order** for child elements.
6)	DTD is **not extensible**.	XSD is **extensible**.
7)	DTD is **not simple to learn**.	XSD is **simple to learn** because you don't need to learn new language.
8)	DTD provides **less control** on XML structure.	XSD provides **more control** on XML structure.

9. **Analyze** about Query String in PHP.

Query string

The information can be sent across the web pages. This information is called query string. This query string can be passed from one page to another by appending it to the address of the page. You can pass more than one query string by inserting the & sign between the query strings. A query string can contain two things: the query string ID and its value. The query string passed across the web pages is stored in $_REQUEST, $_GET, or $_POST variable.

Query string handling in PHP

Query strings

To access the data in a query string you can use the *$_GET* global array. Each element in this array has a key which is the name of the query string variable and a value which is the value of that variable.

```
<a href="mypage.php?variable1=value1&variable2=value2">my link</a>
```

This link loads the page mypage.php with two variables *variable1* and *variable2* with values value1 and value2 respectively.

```
echo $_GET['variable1'];
echo $_GET['variable2'];
// outputs:
//value1
//value2
```

Form data

The get method of forms sends the data to a page via a query string.

```
<form name="form1" id="form1" method="get" action="">
  <input name="textbox" id="textbox" type="text" value="value1" />
  <input name="textbox2" id="textbox2" type="text" value="value2" />
  <input type="submit" name="submitbutton" id="submitbutton" value="Submit" />
</form>
```

This form passes the value of the two text boxes to the page myform.php.

```
print_r($_GET);
// outputs:
// Array (
//  [textbox] => value1
//  [textbox2] => value2
//  [submitbutton] => Submit
// )

echo $_GET['textbox'];
//outputs: value1
```

10. **Show** an example for XML namespace.

A **Namespace** is a set of unique names. Namespace is a mechanisms by which element and attribute name can be assigned to a group. The Namespace is identified by URI(Uniform Resource Identifiers).

Namespace Declaration

A Namespace is declared using reserved attributes. Such an attribute name must either be **xmlns** or begin with **xmlns:** shown as below −

```
<element xmlns:name = "URL">
```

Syntax

- The Namespace starts with the keyword **xmlns**.
- The word **name** is the Namespace prefix.
- The **URL** is the Namespace identifier.

Example

Namespace affects only a limited area in the document. An element containing the declaration and all of its descendants are in the scope of the Namespace. Following is a simple example of XML Namespace −

```
<?xml version = "1.0" encoding = "UTF-8"?>
<cont:contact xmlns:cont = "www.tutorialspoint.com/profile">
   <cont:name>Tanmay Patil</cont:name>
   <cont:company>TutorialsPoint</cont:company>
   <cont:phone>(011) 123-4567</cont:phone>
</cont:contact>
```

Here, the Namespace prefix is **cont**, and the Namespace identifier (URI) as *www.tutorialspoint.com/profile*. This means, the element names and attribute names with the **cont** prefix (including the contact element), all belong to the *www.tutorialspoint.com/profile* namespace.

11.

Define XML parse tree.

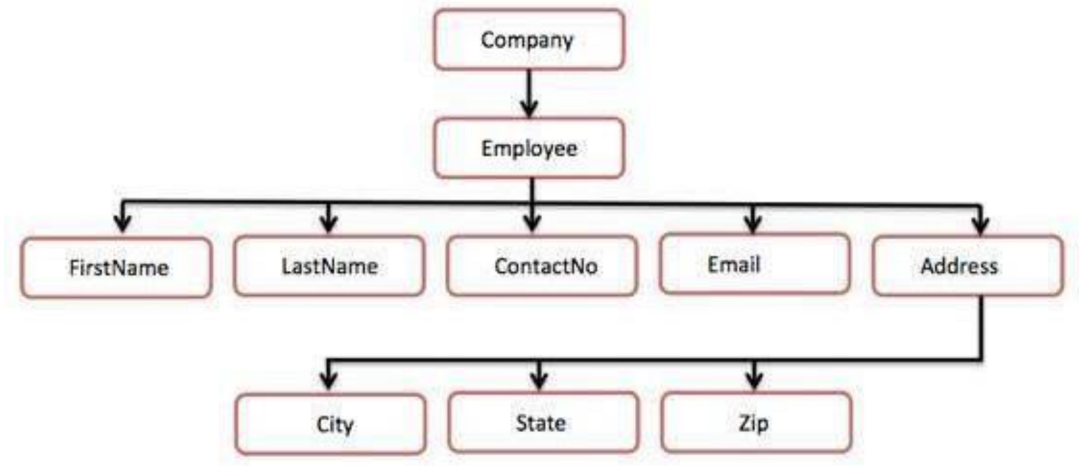
An XML document is always descriptive. The tree structure is often referred to as **XML Tree** and plays an important role to describe any XML document easily.

The tree structure contains root (parent) elements, child elements and so on. By using tree structure, you can get to know all succeeding branches and sub-branches starting from the root. The parsing starts at the root, then moves down the first branch to an element, take the first branch from there, and so on to the leaf nodes.

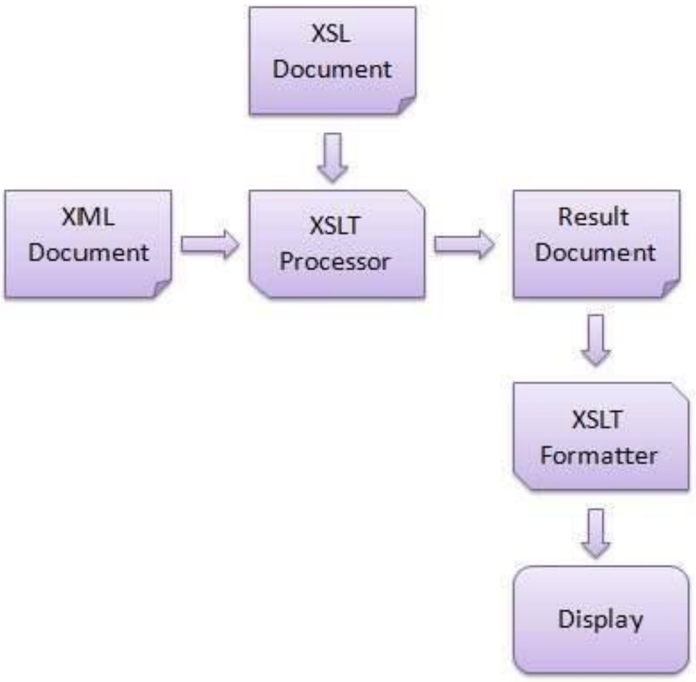
Example

Following example demonstrates simple XML tree structure −

```
<?xml version = "1.0"?>
<Company>
   <Employee>
      <FirstName>Tanmay</FirstName>
```

	```
	      <LastName>Patil</LastName>
	      <ContactNo>1234567890</ContactNo>
	      <Email>tanmaypatil@xyz.com</Email>
	      <Address>
	         <City>Bangalore</City>
	         <State>Karnataka</State>
	         <Zip>560212</Zip>
	      </Address>
	   </Employee>
	</Company>
	```
	Following tree structure represents the above XML document −
	 Company
	Employee
	FirstName
	LastName
	ContactNo
	Email
	Address
	City
	State
	Zip
	In the above diagram, there is a root element named as <company>. Inside that, there is one more element <Employee>. Inside the employee element, there are five branches named <FirstName>, <LastName>, <ContactNo>, <Email>, and <Address>. Inside the <Address> element, there are three sub-branches, named <City> <State> and <Zip>.

12.	**Identify** why XSLT is an important tool for development of web applications. ## What is XSLT XSLT, Extensible Stylesheet Language Transformations, provides the ability to transform XML data from one format to another automatically. ### How XSLT Works An XSLT stylesheet is used to define the transformation rules to be applied on the target XML document. XSLT stylesheet is written in XML format. XSLT Processor takes the XSLT stylesheet and applies the transformation rules on the target XML document and then it generates a formatted document in the form of XML, HTML, or text format. This formatted document is then utilized by XSLT formatter to generate the actual output which is to be displayed to the end-user.



Advantages

Here are the advantages of using XSLT −

- Independent of programming. Transformations are written in a separate xsl file which is again an XML document.
- Output can be altered by simply modifying the transformations in xsl file. No need to change any code. So Web designers can edit the stylesheet and can see the change in the output quickly.

13. **Assess** the data types in XML schema.

You can define XML schema elements in the following ways −

Simple Type

Simple type element is used only in the context of the text. Some of the predefined simple types are: xs:integer, xs:boolean, xs:string, xs:date. For example −

```
<xs:element name = "phone_number" type = "xs:int" />
```

Complex Type

A complex type is a container for other element definitions. This allows you to specify which child elements an element can contain and to provide some structure within your XML documents. For example −

```
<xs:element name = "Address">
   <xs:complexType>
      <xs:sequence>
         <xs:element name = "name" type = "xs:string" />
         <xs:element name = "company" type = "xs:string" />
         <xs:element name = "phone" type = "xs:int" />
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

14.

Explain DTD for XML Schemas.

A **document type definition** (**DTD**) is a set of *markup declarations* that define a *document type* for a SGML-family markup language (GML, SGML, XML, HTML).

A DTD defines the valid building blocks of an XML document. It defines the document structure with a list of validated elements and attributes. A DTD can be declared inline inside an XML document, or as an external reference.

XML DTD schema example

An example of a very simple external XML DTD to describe the schema of a list of persons might consist of:

```
<!ELEMENT people_list (person)*>
<!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT birthdate (#PCDATA)>
<!ELEMENT gender (#PCDATA)>
<!ELEMENT socialsecuritynumber (#PCDATA)>
```

Taking this line by line:

1. `people_list` is a valid element name, and an instance of such an element contains any number of `person` elements. The `*` denotes there can be 0 or more `person` elements within the `people_list` element.
2. `person` is a valid element name, and an instance of such an element contains one element named `name`, followed by one named `birthdate` (optional), then `gender` (also optional) and `socialsecuritynumber` (also optional). The `?` indicates that an element is optional. The reference to the `name` element name has no `?`, so a `person` element *must* contain a `name` element.
3. `name` is a valid element name, and an instance of such an element contains "parsed character data" (#PCDATA).
4. `birthdate` is a valid element name, and an instance of such an element contains parsed character data.
5. `gender` is a valid element name, and an instance of such an element contains parsed character data.
6. `socialsecuritynumber` is a valid element name, and an instance of such an element contains parsed character data.

An example of an XML file that uses and conforms to this DTD follows. The DTD is referenced here as an external subset, via the SYSTEM specifier and a URI. It assumes that we can identify the DTD with the relative URI reference "example.dtd"; the "people_list" after "!DOCTYPE" tells us that the root tags, or the first element defined in the DTD, is called "people_list":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE people_list SYSTEM "example.dtd">
<people_list>
  <person>
    <name>Fred Bloggs</name>
    <birthdate>2008-11-27</birthdate>
    <gender>Male</gender>
  </person>
</people_list>
```

15. **Evaluate** the process of displaying XML document in browser.

Display an XML Document in a Web Browser

Displaying XML Using CSS

XML stands for **E**xtensible **M**arkup **L**anguage. It is a dynamic markup language. It is used to transform data from one form to another form.
An XML file can be displayed using two ways. These are as follows :-

1. Cascading Style Sheet
2. Extensible Stylesheet Language Transformation

Displaying XML file using CSS :
CSS can be used to display the contents of the XML document in a clear and precise manner. It gives the design and style to whole XML document.

- **Basic steps in defining a CSS style sheet for XML :**
  For defining the style rules for the XML document, the following things shoulde be done :-

  1. Define the style rules for the text elements such as font-size, color, font-weight, etc.
  2. Define each element either as a block, inline or list element, using the display property of CSS.
  3. Identify the titles and bold them.
- **Linking XML with CSS :**
  In order to display the XML file using CSS, link XML file with CSS. Below is the syntax for linking the XML file with CSS:
  <?xml-stylesheet type="text/css" href="name_of_css_file.css"?>
- **Example 1.**
  In this example, the XML file is created that contains the information about five books and displaying the XML file using CSS.
  XML file :

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="Rule.css"?>
<books>
```

```
  <heading>Welcome To GeeksforGeeks </heading>
  <book>
     <title>Title -: Web Programming</title>
     <author>Author -: Chrisbates</author>
     <publisher>Publisher -: Wiley</publisher>
     <edition>Edition -: 3</edition>
     <price> Price -: 300</price>
  </book>
  <book>
     <title>Title -: Internet world-wide-web</title>
     <author>Author -: Ditel</author>
     <publisher>Publisher -: Pearson</publisher>
     <edition>Edition -: 3</edition>
     <price>Price -: 400</price>
  </book>

  </books>
```

In the above example, Books.xml is linked with Rule.css which contains the corresponding style sheet rules.

CSS FILE :

```
books {
    color: white;
    background-color : gray;
    width: 100%;
}
 heading {
    color: green;
    font-size : 40px;
    background-color : powderblue;
}
 heading, title, author, publisher, edition, price {
    display : block;
}
 title {
    font-size : 25px;
    font-weight : bold;
}
```

- **Output :**

16.	**Summarize** about the need for Namespace in XML.

XML namespaces are used for providing uniquely named elements and attributes in an XML document. They are defined in a W3C recommendation.[1][2] An XML instance may contain element or attribute names from more than one XML vocabulary. If each vocabulary is given a namespace, the ambiguity between identically named elements or attributes can be resolved.

How to get rid of name conflict?

1) By Using a Prefix

You can easily avoid the XML namespace by using a name prefix.

```
<h:table>
  <h:tr>
    <h:td>Aries</h:td>
    <h:td>Bingo</h:td>
  </h:tr>
</h:table>
<f:table>
  <f:name>Computer table</f:name>
  <f:width>80</f:width>
.  <f:length>120</f:length>
.</f:table>
```

Note: In this example, you will get no conflict because both the tables have specific names.

2) By Using xmlns Attribute

You can use xmlns attribute to define namespace with the following syntax:

```
<element xmlns:name = "URL">
```

Let's see the example:

```
<root>
<h:table xmlns:h="http://www.abc.com/TR/html4/">
  <h:tr>
    <h:td>Aries</h:td>
    <h:td>Bingo</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="http://www.xyz.com/furniture">
.  <f:name>Computer table</f:name>
.  <f:width>80</f:width>
.  <f:length>120</f:length>
.</f:table>
.</root>
```

In the above example, the <table> element defines a namespace and when a namespace is defined for an element, the child elements with the same prefixes are associated with the same namespace.

17.

Analyze on ATOM in RSS.

What is Atom 1.0 ?

Atom is the name of an XML-based Web content and metadata syndication format, and an application-level protocol for publishing and editing Web resources belonging to periodically updated websites.

Atom is a relatively recent spec and is much more robust and feature-rich than RSS. For instance, where RSS requires descriptive fields such as title and link only in item breakdowns, Atom requires these things for both items and the full Feed.

All Atom Feeds must be well-formed XML documents, and are identified with the *application/atom+xml* media type.

Structure of an Atom 1.0 Feed

A Feed consists of some metadata, followed by any number of entries. Here is a basic structure of an Atom 1.0 Feed.

```
<?xml version="1.0"?>
<feed xmlns="http://www.w3.org/2005/Atom">
   <title>...</title>
   <link>...</link>
   <updated>...</updated>

   <author>
      <name>...</name>
   </author>

   <id>...</id>

   <entry>
      <title>...</title>
      <link>...</link>
      <id>...</id>

      <updated>...</updated>
      <summary>...</summary>
   </entry>

</feed>
```

Atom 1.0 Feed Tags

An Atom 1.0 Feed Document will be constructed of the following two elements:

- <feed> Elements
- <entry> Elements

18. **Summarize** the advantage of RSS documents?

RSS - Advantages

RSS is taking off so quickly because people are liking it. RSS is easy to use and it has advantages for a publisher as well as for a subscriber. Here we have listed out a few advantages of RSS for subscribers as well as for publishers.

Advantages for Subscribers

RSS subscribers are the people who subscribe to read a published Feed. Here are some of the advantages of RSS Feeds for subscribers:

- **All news at one place:** You can subscribe to multiple news groups and then you can customize your reader to have all the news on a single page. It will save you a lot of time.
- **News when you want it:** Rather than waiting for an e-mail, you go to your RSS reader when you want to read a news. Furthermore, RSS Feeds display more quickly than information on web-sites, and you can read them offline if you prefer.
- **Get the news you want:** RSS Feed comes in the form of headlines and a brief description so that you can easily scan the headlines and click only those stories that interest you.
- **Freedom from e-mail overload:** You are not going to get any email for any news or blog update. You just go to your reader and you will find updated news or blog automatically whenever there is a change on the RSS server.
- **Easy republishing:** You may be both a subscriber and a publisher. For example, you may have a web-site that collects news from various other sites and then republishes it. RSS allows you to easily capture that news and display it on your site.

Advantages for Publishers

RSS publishers are the people who publish their content through RSS feed. We would suggest you to use RSS:

- if you want to get your message out and easily,
- if you want people to see what you publish, and
- if you want your news to bring people back to your site.

Here are some of the advantages of RSS if you publish on the Web:

- **Easier publishing:** RSS is really simple publishing. You don't have to maintain a database of subscribers to send your information to them, instead they will access your Feed using a reader and will get updated content automatically.
- **A simpler writing process:** If you have a new content on your web site, you only need to write an RSS Feed in the form of titles and short descriptions, and link back to your site.
- **An improved relationship with your subscribers:** Because people subscribe from their side, they don't feel as if you are pushing your content on them.
- **The assurance of reaching your subscribers:** RSS is not subject to spam filters, your subscribers get the Feeds, which they subscribe to and nothing more.
- **Links back to your site:** RSS Feeds always include links back to a website. It directs a lot of traffic towards your website.
- **Relevance and timeliness:** Your subscribers always have the latest information fromyour site.

19. **Rewrite** the declaration for elements in XML.

20. How would you **prepare** the steps to get the RSS file on web?

Uploading an RSS Feed

Here are the simple steps to put your RSS Feed on the web.

- First decide which version of RSS Feed you are going to use for your site. We would recommend you to use the latest version available.
- Create your RSS Feed in a text file with extension either .xml or .rdf. Upload this file on your web server.
- You should validate your RSS Feed before making it live. Check the next chapter on RSS Feed Validation.
- Create a link on your Web Pages for the RSS Feed file. You will use a small yellow button for the link that says either RSS or XML.

Now, your RSS Feed is online and people can start using it. But there are ways to promote your RSS Feed so that more number of people can use your RSS Feed.

Promote Your RSS Feed

- Submit your RSS Feed to the RSS Feed Directories. There are many directories available on the web, where you can register your Feed. Some of them are given here:
  - Syndic8: Over 300,000 Feeds listed.
  - Daypop: Over 50,000 feeds listed.
  - Newsisfree: Over 18,000 Feeds.
- Register your Feed with the major search engines. Similar to your web pages, you can add your Feed as well with the following major search engines.
  - Yahoo - http://publisher.yahoo.com/promote.php

	○ Google - http://www.google.com/webmasters/add.html ○ Bing - http://www.bing.com/toolbox/submit-site-url Keeping Up-To-Date Feed As we have explained earlier, RSS Feed makes sense for the site which are changing their content very frequently, for example, any news or blogging sites. So now, you have got RSS Feed buttons from Google, Yahoo, and MSN. You must make sure to update your content frequently and that your RSS Feed is constantly available.
PART-B	
1.	(i) **Describe** about the introduction and installation of PHP. **Introduction to PHP** PHP is one of the most widely used server side scripting language for web development. Popular websites like Facebook, Yahoo, Wikipedia etc are developed using PHP. PHP is so popular because it's very simple to learn, code and deploy on server, hence it has been the first choice for beginners since decades. Uses of PHP To further fortify your trust in PHP, here are a few applications of this amazing scripting language: 1. It can be used to **create Web applications** like Social Networks(Facebook, Digg), Blogs(Wordpress, Joomla), eCommerce websites(OpenCart, Magento etc.) etc. 2. **Common Line Scripting**. You can write PHP scripts to perform different operations on any machine, all you need is a PHP parser for this. 3. **Create Facebook applications** and easily integrate Facebook plugins in your website, using Facebook's PHP SDK. Check this link for more information. 4. **Sending Emails** or building email applications because PHP provides with a robust email sending function. 5. Wordpress is one of the most used blogging(CMS) platform in the World, and if you know PHP, you can try a hand in **Wordpress plugin development**.

Manual Installation

Step 1: Download the files. Download the latest **PHP** 5 ZIP package from www.**php**.net/downloads.**php**. ...

Step 2: Extract the files. ...

Step 3: Configure **php**. ...

Step 4: Add C:\php to the path environment variable. ...

Step 5: Configure **PHP** as an Apache module. ...

Step 6: Test a **PHP** file.

(ii) **Design** simple calculator using PHP.

Calculator.php

<!DOCTYPE html>

```
<html>
	<head>
		<title>Simple Calculator In PHP | Webdevtrick.com</title>
		<meta charset="utf-8">
		<meta http-equiv="X-UA-Compatible" content="IE=edge">
		<meta name="viewport" content="width=device-width, initial-scale=1">

		<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css"
rel="stylesheet">
	</head>
	<body>

		<div class="container" style="margin-top: 50px">

		<?php

		// If the submit button has been pressed
		if(isset($_POST['submit']))
		{
		// Check number values
		if(is_numeric($_POST['number1']) && is_numeric($_POST['number2']))
		{
		// Calculate total
		if($_POST['operation'] == 'plus')
		{
		$total = $_POST['number1'] + $_POST['number2'];
		}
		if($_POST['operation'] == 'minus')
		{
		$total = $_POST['number1'] - $_POST['number2'];
		}
		if($_POST['operation'] == 'multiply')
		{
		$total = $_POST['number1'] * $_POST['number2'];
```

	```
	}
	if($_POST['operation'] == 'divided by')
	{
	$total = $_POST['number1'] / $_POST['number2'];
	}

	// Print total to the browser
	echo "<h1>{$_POST['number1']} {$_POST['operation']}
	{$_POST['number2']} equals {$total}</h1>";

	} else {

	// Print error message to the browser
	echo 'Numeric values are required';

	}
	}
	// end PHP. Code by webdevtrick.com
	?>

	<!-- Calculator form by webdevtrick.com -->
	<form method="post" action="calculator.php">
	  <input name="number1" type="text" class="form-control" style="width:
	150px; display: inline" />
	  <select name="operation">
	    <option value="plus">Plus</option>
	    <option value="minus">Minus</option>
	    <option value="multiply">Multiply</option>
	    <option value="divided by">Devide</option>
	  </select>
	  <input name="number2" type="text" class="form-control" style="width:
	150px; display: inline" />
	  <input name="submit" type="submit" value="Calculate" class="btn btn-
	primary" />
	</form>

	</div>

	</body>
	</html>
	?>
	```
2.	**Explain** about control statements and data types in PHP with example. **<u>Control Statements in PHP with Examples</u>**

Like any other languages, PHP is built out of a series of control statements. The control statement can be an assignment, a function call, a loop, a conditional statement or even a statement that does nothing or an empty statement.

In PHP we have the following conditional statements:

if statement – We use this control statement to execute some code only if a specified condition is true.

if…else statement – We use this control statement to execute some code if a condition is true and another code if the condition is false.

if…elseif….else statement – We use this control statement to select one of several blocks of code to be executed

switch statement – We use this control statement to select one of many blocks of code to be executed

1. The if Statement

Use the if statement to execute some code only if a specified condition is true.
The expression is evaluated to its Boolean value. If expression evaluates to TRUE, PHP will execute statement, and if it evaluates to FALSE – it'll ignore it

Syntax

```
if (condition) {
code to be executed if condition is true;
}
```

The following example would display " A is bigger than B" if $a is bigger than $b:

```
<?php
if ($a > $b)
echo "A is bigger than B";
?>
```

2. The if…else Statement

elseif, as its name suggests, is a combination of if and else. Like else, it extends an if statement to execute a different statement in case the original if expression evaluates to FALSE. However, unlike else, it will execute that alternative expression only if the elseif conditional expression evaluates to TRUE.

```
if (condition)
code to be executed if condition is true;
else
code to be executed if condition is false;
```

For example, the following code would display a is bigger than b, a equal to b or a is smaller than b:

```
<?php
if ($a > $b) {
echo "a is bigger than b";
} elseif ($a == $b) {
echo "a is equal to b";
} else {
echo "a is smaller than b";
}
?>
```

3. The if…elseif….else Statement

4. The Switch Statement

The switch statement is similar to IF statements on the same expression. In many occasions,

	Use the if….elseif…else statement to select one of several blocks of code to be executed.
	
	if (condition)
	code to be executed if condition is true;
	elseif (condition)
	code to be executed if condition is true;
	else
	code to be executed if condition is false;
	
	Note: Note that elseif and else if will only be considered exactly the same when using curly brackets as in the above example. When using a colon to define your if/elseif conditions, you must not separate else if into two words, or PHP will fail with a parse error.
		you may want to compare the same variable (or expression) with many different values, and execute a different piece of code depending on which value it equals to. This is exactly what the switch statement is for.
		
		switch ( )
		{
		case condition1
		break;
		case condition2
		break;
		}
		
		For example, the following code would display $i matched value as 0 or 1 or 2:
		<?php
		switch ($i) {
		case 0:
		echo "i equals 0";
		case 1:
		echo "i equals 1";
		case 2:
		echo "i equals 2";
		}
		?>

3.	(i) **Create** an XML document that marks up various sports and their descriptions. Use XSLT to tabulate neatly the elements and attributes of the document. <?xml version="1.0"?> -<events league="WC Falun" tournament="2010/2011" template="World Cup" sport="Cross Country Skiing" ut="2012-09-05" id="821135"> -<event id="866683" status="Finished" round="8001 - 1/1 (Final)" date="2011-03-20 13:15:00" name="10 km Freestyle Handicap Pursuit"> -<results participantname="Marit Bjoergen" participantid="43427"> <result id="9498426" value="1" type="rank"/> <result id="9498424" value="27:58.0" type="duration"/>	

```
<result id="9505038" value="200" type="points"/>

<result id="9498425" value="" type="comment"/>

<result id="9497448" value="1" type="startnumber"/>

</results>


-<results participantname="Justyna Kowalczyk" participantid="43775">

<result id="9498429" value="2" type="rank"/>

<result id="9498427" value="+1:58.0" type="duration"/>

<result id="9505039" value="160" type="points"/>

<result id="9498428" value="" type="comment"/>

<result id="9497454" value="2" type="startnumber"/>

</results>
</event></events>
```

(ii) **Illustrate** a JSP page that enables the user to input the first name and in response outputs the last name.

POST Method Example Using Form

Below is the **main.jsp** JSP program to handle the input given by web browser using the GET or the POST methods.

```
<html>
  <head>
    <title>Using GET and POST Method to Read Form Data</title>
  </head>

  <body>
    <center>
    <h1>Using POST Method to Read Form Data</h1>

    <ul>
      <li><p><b>First Name:</b>
        <%= request.getParameter("first_name")%>
      </p></li>
      <li><p><b>Last  Name:</b>
        <%= request.getParameter("last_name")%>
      </p></li>
    </ul>
  </body>
</html>
```

Following is the content of the **Hello.htm** file −

```
<html>
  <body>

    <form action = "main.jsp" method = "POST">
      First Name: <input type = "text" name = "first_name">
      <br />
      Last Name: <input type = "text" name = "last_name" />
      <input type = "submit" value = "Submit" />
    </form>

  </body>
</html>
```

Let us now keep **main.jsp** and hello.htm in **<Tomcat-installationdirectory>/webapps/ROOT directory**. When you access ***http://localhost:8080/Hello.htm***, you will receive the following output.

```
First Name: 
Last Name: 
```

Try to enter the First and the Last Name and then click the submit button to see the result on your local machine where tomcat is running.

4. **Create** a webserver based chat application using PHP. The application should provide the following functions Login, Send message (to one or more contacts) and Receive messages (from one or more contacts)

5. (i) **Write** a PHP program that tests whether an email address is input correctly. Test your program with both valid and invalid email addresses.

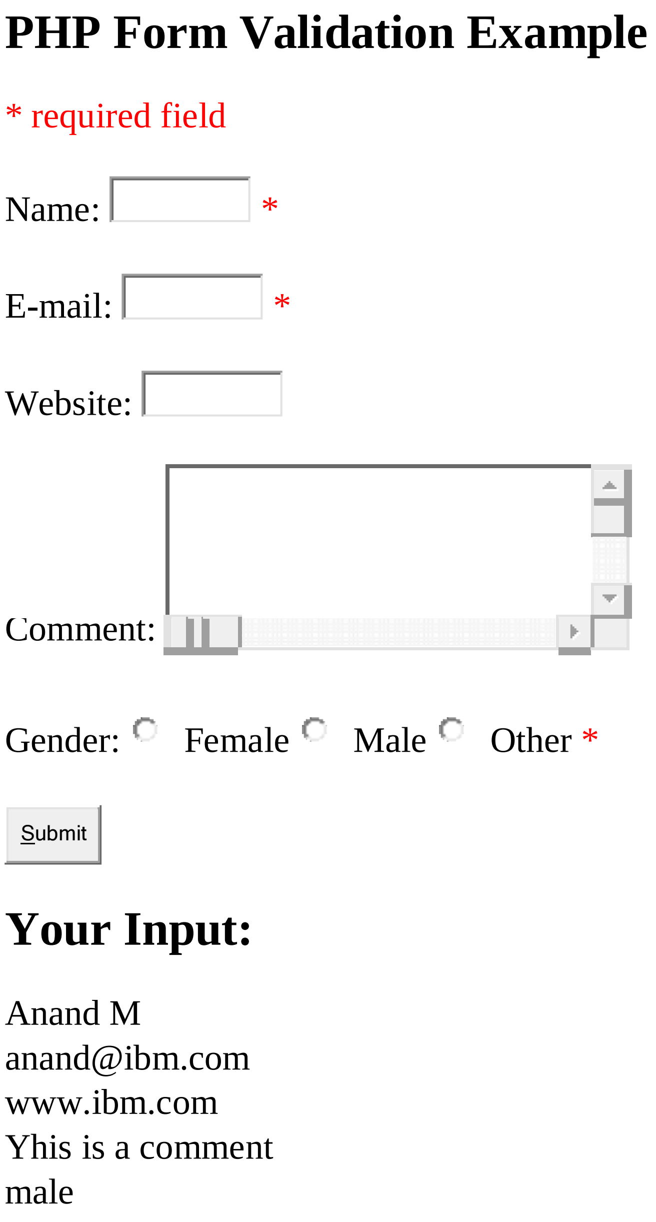
PHP - Validate Name, E-mail, and URL

Example

```
<?php
// define variables and set to empty values
```

```
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid (this regular expression
also allows dashes in the URL)
    if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}
?>
```

OUTPUT



PHP Form Validation Example

* required field

Name: *

E-mail: *

Website:

Comment:

Gender: Female Male Other *

Submit

Your Input:

Anand M
anand@ibm.com
www.ibm.com
Yhis is a comment
male

6. **Identify** and explain about database connectivity illustrate PHP connectivity with any of the databases.

METHOD FOR: CONNECTING TO MYSQL USING MYSQL

The MySQL Improved extension uses the *mysqli* class, which replaces the set of legacy MySQL functions.

To connect to MySQL using the MySQL Improved extension, follow these steps:

1. Use the following PHP code to connect to MySQL and select a database. Replace *username* with your username, *password* with your password, and *dbname* with the database name:

```
2.    <?php

3.        $mysqli = new mysqli("localhost", "username", "password
", "dbname");

?>
```

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

7.

(i) **Discuss** on methods for using cookies in PHP.

Cookies in PHP

Cookies are used to store the information of a web page in a remote browser, so that when the same user comes back to that page, that information can be retrieved from the browser itself.

Uses of cookie

Cookies are often used to perform following tasks:

- **Session management**: Cookies are widely used to manage user sessions. For example, when you use an online shopping cart, you keep adding items in the cart and finally when you checkout, all of those items are added to the list of items you have purchased. This can be achieved using cookies.
- **User identification**: Once a user visits a webpage, using cookies, that user can be remembered. And later on, depending upon the search/visit pattern of the user,

	content which the user likely to be visited are served. A good example of this is 'Retargetting'. A concept used in online marketing, where depending upon the user's choice of content, advertisements of the relevant product, which the user may buy, are served.
	
	• **Tracking / Analytics**: Cookies are used to track the user. Which, in turn, is used to analyze and serve various kind of data of great value, like location, technologies (e.g. browser, OS) form where the user visited, how long (s)he stayed on various pages etc.
	
	How to create a cookie in PHP
	
	PHP has a setcookie() function to send a cookie. We will discuss this function in detail now.
	
	`setcookie(name, value, expire, path, domain, secure, httponly)`
	
	setcookie() returns boolean.
	
	Example:
	
	Following example shows how to create a cookie in PHP.
	
	`<?php`
	`$cookie_value = "w3resource tutorials";`
	`setcookie("w3resource", $cookie_value, time()+3600, "/home/your_usename/", "example.com", 1, 1);`
	`if (isset($_COOKIE['cookie']))`
	`echo $_COOKIE["w3resource"];`
	`?>`
	
	(ii) **Give** a note on regular expressions.

8.	**Summarize** in detail the XML schema, built in and user defined data types. **What is an XML Schema?** An XML Schema describes the structure of an XML document.

The XML Schema language is also referred to as XML Schema Definition (XSD).

XSD Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

The purpose of an XML Schema is to define the legal building blocks of an XML document:

- the elements and attributes that can appear in a document
- the number of (and order of) child elements
- data types for elements and attributes
- default and fixed values for elements and attributes

XSD Simple Elements

XML Schemas define the elements of your XML files.

A simple element is an XML element that contains only text. It cannot contain any other elements or attributes.

The text can be of many different types like boolean, string, date, etc.), or it can be a custom type that you can define yourself.

You can also add restrictions (facets) to a data type in order to limit its content, or you can require the data to match a specific pattern.

The syntax for defining a simple element is:

```
<xs:element name="xxx" type="yyy"/>
```

where xxx is the name of the element and yyy is the data type of the element.

XML Schema has a lot of built-in data types. The most common types are:

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Example

Simple element definitions:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

Here are some XML elements:

```
<lastname>Ronald</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

Simple elements may have a default value which is automatically assigned to the element when no other value is specified as shown below:

```
<xs:element name="color" type="xs:string" default="red"/>
```

How to Define a Complex Element

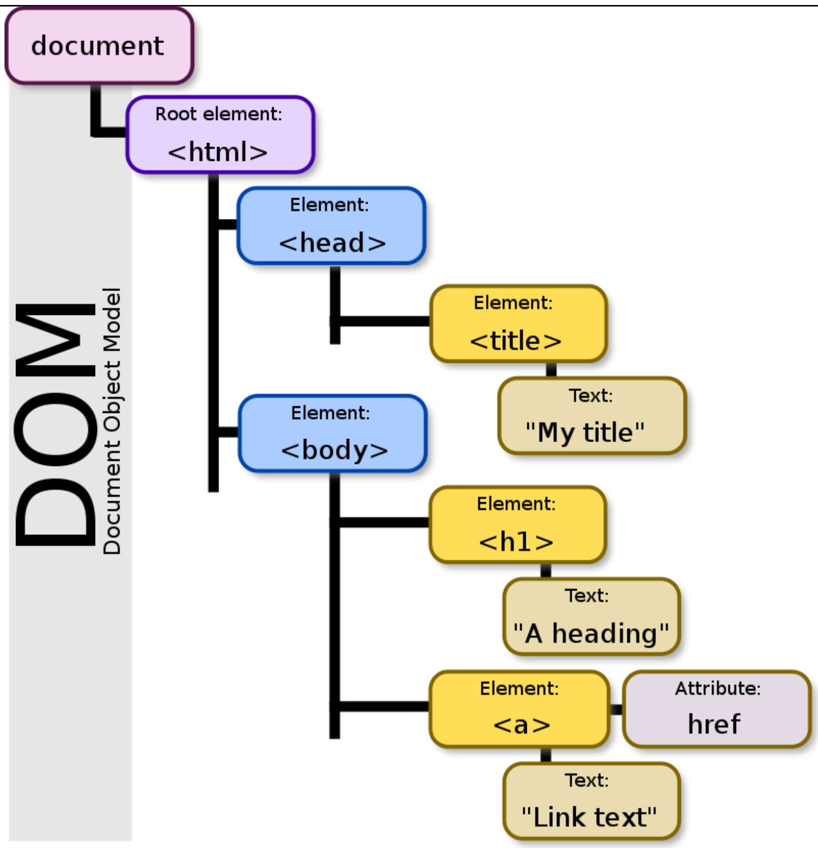
We can define a complex element in an XML Schema two different ways:

1. The "employee" element can be declared directly by naming the element, like this:

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

	2. The "employee" element can have a type attribute that refers to the name of the complex type to use:
	
	`<xs:element name="employee" type="personinfo"/>`
	
	`<xs:complexType name="personinfo">`
	`  <xs:sequence>`
	`    <xs:element name="firstname" type="xs:string"/>`
	`    <xs:element name="lastname" type="xs:string"/>`
	`  </xs:sequence>`
	`</xs:complexType>`
	
	If you use the method described above, several elements can refer to the same complex type, like this:
	
	`<xs:element name="employee" type="personinfo"/>`
	`<xs:element name="student" type="personinfo"/>`
	`<xs:element name="member" type="personinfo"/>`
	
	`<xs:complexType name="personinfo">`
	`  <xs:sequence>`
	`    <xs:element name="firstname" type="xs:string"/>`
	`    <xs:element name="lastname" type="xs:string"/>`
	`  </xs:sequence>`
	`</xs:complexType>`

9.	(i) **Demonstrate** the building blocks of DOM. The **Document Object Model** (**DOM**) is a cross-platform and language-independent interface that treats an XML or HTML document as a tree structure wherein each node is an object representing a part of the document. The DOM represents a document with a logical tree. Each branch of the tree ends in a node, and each node contains objects. DOM methods allow programmatic access to the tree; with them one can change the structure, style or content of a document. Nodes can have event handlers attached to them. Once an event is triggered, the event handlers get executed.



(ii) **Classify** the types of DTD.

XML DTD

An XML document with correct syntax is called "Well Formed".

An XML document validated against a DTD is both "Well Formed" and "Valid".

DTD stands for Document Type Definition.

A DTD defines the structure and the legal elements and attributes of an XML document.

A "Valid" XML document is "Well Formed", as well as it conforms to the rules of a DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
```

```
<body>Don't forget me this weekend!</body>
</note>
```

The DOCTYPE declaration above contains a reference to a DTD file. The content of the DTD file is shown and explained below.

The purpose of a DTD is to define the structure and the legal elements and attributes of an XML document:

Note.dtd:

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

The DTD above is interpreted like this:

- !DOCTYPE note - Defines that the root element of the document is note
- !ELEMENT note - Defines that the note element must contain the elements: "to, from, heading, body"
- !ELEMENT to - Defines the to element to be of type "#PCDATA"
- !ELEMENT from - Defines the from element to be of type "#PCDATA"
- !ELEMENT heading - Defines the heading element to be of type "#PCDATA"
- !ELEMENT body - Defines the body element to be of type "#PCDATA"

Tip: #PCDATA means parseable character data.

10. How do you **infer the significant** differences between DID and XML schema for defining XML document structures with appropriate examples.

Difference Between XML Schema and DTD

XML Schema vs. DTD

DTD, or Document Type Definition, and XML Schema, which is also known as XSD, are two ways of describing the structure and content of an XML document. DTD is the older of the two, and as such, it has limitations that XML Schema has tried to improve.

Summary:

1. XML Schema is namespace aware, while DTD is not.

2. XML Schemas are written in XML, while DTDs are not.

3. XML Schema is strongly typed, while DTD is not.

4. XML Schema has a wealth of derived and built-in data types that are not available in DTD.

5. XML Schema does not allow inline definitions, while DTD does.

11.

(i) **List** out data types data types of XML

XML Schema Data Types

XML Schema data types can be generally categorized a "simple type" (including embedded simple type) and "complex type." The "embedded simple type" is already defined, but can be used to create a new type through restriction or extension.

Table : XML Schema Data Types

	User can independently define. This type is used when a restriction is placed on an embedded simple type to create and use a new type.
	User can independently define. This type is used when the type has a child element or attribute.

A simple type is a type that only contains text data. This type can be used with element declarations and attribute declarations. On the other hand, a complex data type is a type that has a child element or attribute structure.

●**Simple Type Example**

```
<xs:element name="Department" type="xs:string" />
```

	Here, the section described together with "xs:string" is an embedded simple type according to XML Schema. In this example, we have established the definition that the data type for the element called "Department" is a text string.
	
	●Complex Type Example
	
	`<xs:complexType name="EmployeeType">`
	`  <xs:sequence maxOccurs="unbounded">`
	`    <xs:element ref="Name" />`
	`    <xs:element ref="Department" />`
	`  </xs:sequence>`
	`</xs:complexType>`
	`<xs:element name="Name" type="xs:string" />`
	`<xs:element name="Department" type="xs:string" />`
	
	(ii) **Explain** about the attributes of XML.
	
	XML Elements vs. Attributes
	
	Take a look at these examples:
	
	`<person gender="female">`
	`  <firstname>Anna</firstname>`
	`  <lastname>Smith</lastname>`
	`</person>`
	
	`<person>`
	`  <gender>female</gender>`
	`  <firstname>Anna</firstname>`
	`  <lastname>Smith</lastname>`
	`</person>`
	
	In the first example gender is an attribute. In the last, gender is an element. Both examples provide the same information.
	
	There are no rules about when to use attributes or when to use elements in XML.

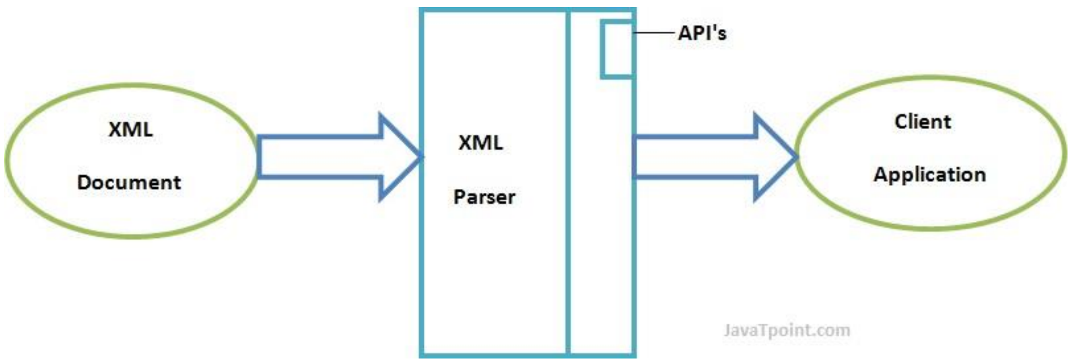
12.	**Summarize** on the following (i) DOM based Parsing. (ii) SAX based Parsing.

XML Parsers

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

XML parser validates the document and check that the document is well formatted.

Let's understand the working of XML parser by the figure given below:



Types of XML Parsers

These are the two main types of XML Parsers:

1. DOM
2. SAX

DOM (Document Object Model)

A DOM document is an object which contains all the information of an XML document. It is composed like a tree structure. The DOM Parser implements a DOM API. This API is very simple to use.

Features of DOM Parser

A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information of the original XML document by invoking methods on this document object.

DOM Parser has a tree based structure.

Advantages

1) It supports both read and write operations and the API is very simple to use.

2) It is preferred when random access to widely separated parts of a document is required.

Disadvantages

1) It is memory inefficient. (consumes more memory because the whole XML document needs to loaded into memory).

2) It is comparatively slower than other parsers.

SAX (Simple API for XML)

A SAX Parser implements SAX API. This API is an event based API and less intuitive.

Features of SAX Parser

It does not create any internal structure.

Clients does not know what methods to call, they just overrides the methods of the API and place his own code inside method.

It is an event based parser, it works like an event handler in Java.

Advantages

1) It is simple and memory efficient.

2) It is very fast and works for huge documents.

Disadvantages

1) It is event-based so its API is less intuitive.

2) Clients never know the full information because the data is broken into pieces.

13. (i) **Compare and contrast** RSS & ATOM.

Difference Between RSS and ATOM

RSS vs ATOM

- Really Simple Syndication or RSS has been the standard for web feeds for a considerable time.
- Web feeds contains either a summary or the full text content of a web page.
- The problem with RSS is the often confusing and non standard conventions used by RSS due in part to its scattered development.
- The advent of the ATOM syndication standard was a response to the design flaws of the RSS standard.
- The primary advantage of the ATOM is its adaptation as the IETF standard.
- Being an IETF standard, each atom feed contains an explicit declaration of the format of the content along with what language is used.
- RSS feeds do not declare its content, but since it only contains plain text or escaped HTML, it is rather easy for the browser to distinguish which is which.

A major flaw of RSS is in its code. RSS code isn’t really very usable in other XML vocabularies since it wasn’t really intended to do so at the very beginning. ATOM code has been built from the ground with modularity in mind. Therefore, a great majority of its code is reusable even with other XML vocabularies like RSS.

Summary:

1. ATOM is an IETF standard while RSS is not
2. ATOM feeds explicitly indicates the content while the browser is left to figure out whether the RSS feed contains plain text or escaped HTML
3. ATOM code is modular and reusable while RSS code is not
4. RSS still holds dominance in the syndication format due to its head start and popularity

(ii) **Explain** in detail about XSL elements.

XSLT <xsl:element>

Definition and Usage

The <xsl:element> element is used to create an element node in the output document.

Syntax

```
<xsl:element
name="name"
namespace="URI"
use-attribute-sets="namelist">

  <!-- Content:template -->

</xsl:element>
```

Attributes

Attribute	**Value**	**Description**
name	name	Required. Specifies the name of the element to be created (the value of the name attribute can be set to an expression that is computed at run-time, like this: <xsl:element name="{$country}" />
namespace	URI	Optional. Specifies the namespace URI of the eler nt (the value of the namespace attribute can be set to an expression that is computed at run-time, like this: <xsl:element name="{$country}" namespace="{$someuri}"/>
use-attribute-sets	namelist	Optional. A white space separated list of attribute-sets containing attributes to be added to the element

Example 1

Create a "singer" element that contains the value of each artist element:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <xsl:for-each select="catalog/cd">
    <xsl:element name="singer">
      <xsl:value-of select="artist" />
    </xsl:element>
    <br />
  </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

EXAMPLE OUTPUT FILES

XML File

```
<catalog>
<cd>
<title>Empire Burlesque</title>
<artist>Bob Dylan</artist>
<country>USA</country>
<company>Columbia</company>
<price>10.90</price>
<year>1985</year>
</cd>
<cd>
<title>Hide your heart</title>
<artist>Bonnie Tyler</artist>
<country>UK</country>
<company>CBS Records</company>
<price>9.90</price>
<year>1988</year>
</cd>
</catalog>
```

XSL File

```
xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version=
"1.0">
<xsl:template match="/">
<html>
<body>
<h2>My CD Collection</h2>
<xsl:for-each select="catalog/cd">
```

```
<xsl:element name="singer">
<xsl:value-of select="artist"/>
</xsl:element>
<br/>
</xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

RESULT

My CD Collection

Bob Dylan
Bonnie Tyler
Dolly Parton
Gary Moore

14.

Explain in detail about
(i) XSL and XSLT transformation

XSLT - Transformation

What is XSLT?

XSL Transformations (XSLT 2.0) is a language for transforming XML documents into other XML documents, text documents or HTML documents. You might want to format a chapter of a book using XSL-FO, or you might want to take a database query and format it as HTML.

Wildly Popular

XSLT has become the language of choice for a very wide range of XML applications. It is of course still used to produce XSL-FO documents for printing, but it is also used to integrate back-end software for Web sites. We can find XSLT inside most modern Web browsers, so that XML can be transformed on the fly without the user even noticing; you will find XSLT on the desktop, in servers, in network appliances.

What is XSLT Used For?

If you make a purchase on eBay, or buy a book at Amazon, chances are that pretty much everything you see on every Web page has been processed with XSLT. Use XSLT to process multiple XML documents and to produce any combination of text, HTML and XML output. XSLT support is shipped with all major computer operating systems today, as well as being built in to all major Web browsers.

XSLT – Transformation : STEPS

1) Start with a Raw XML Document
2) Create an XSL Style Sheet
3) Link the XSL Style Sheet to the XML Document

1) Start with a Raw XML Document

We want to **transform** the following XML document ("cdcatalog.xml") into XHTML:

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.</catalog>
```

2) Create an XSL Style Sheet

Then you create an XSL Style Sheet ("cdcatalog.xsl") with a transformation template:

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

3) Link the XSL Style Sheet to the XML Document

Add the XSL style sheet reference to your XML document ("cdcatalog.xml"):

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>


    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.</catalog>
```

If you have an XSLT compliant browser it will nicely **transform** your XML into XHTML.

Sample OUTPUT

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore

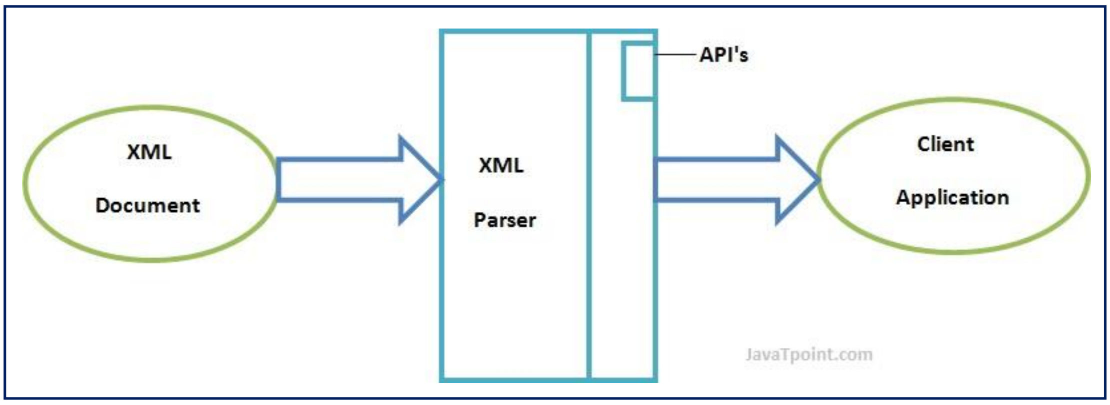
(ii) Comparison of DOM & SAX

XML Parsers

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

XML parser validates the document and check that the document is well formatted.

Let's understand the working of XML parser by the figure given below:



Types of XML Parsers

These are the two main types of XML Parsers:

3. DOM
4. SAX

DOM (Document Object Model)

A DOM document is an object which contains all the information of an XML document. It is composed like a tree structure. The DOM Parser implements a DOM API. This API is very simple to use.

Features of DOM Parser

A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information of the original XML document by invoking methods on this document object.

DOM Parser has a tree based structure.

Advantages

1) It supports both read and write operations and the API is very simple to use.

2) It is preferred when random access to widely separated parts of a document is required.

Disadvantages

1) It is memory inefficient. (consumes more memory because the whole XML document needs to loaded into memory).

2) It is comparatively slower than other parsers.

SAX (Simple API for XML)

A SAX Parser implements SAX API. This API is an event based API and less intuitive.

Features of SAX Parser

It does not create any internal structure.

Clients does not know what methods to call, they just overrides the methods of the API and place his own code inside method.

It is an event based parser, it works like an event handler in Java.

Advantages

1) It is simple and memory efficient.

2) It is very fast and works for huge documents.

Disadvantages

1) It is event-based so its API is less intuitive.

2) Clients never know the full information because the data is broken into pieces.

	PART-C
1.	**Explain** how you shall carry out String Manipulations using a PHP Program.

Manipulating PHP Strings

PHP provides many built-in functions for manipulating strings like calculating the length of a string, find substrings or characters, replacing part of a string with different characters, take a string apart, and many others.

Here are the examples of some of these functions.

Calculating the Length of a String

The `strlen()` function is used to calculate the number of characters inside a string. It also includes the blank spaces inside the string.

Example

Run this code »

```
<?php
$my_str = 'Welcome to Tutorial Republic';

// Outputs: 28
echo strlen($my_str);
?>
```

Counting Number of Words in a String

The `str_word_count()` function counts the number of words in a string.

Example

Run this code »

```
<?php
$my_str = 'The quick brown fox jumps over the lazy dog.';

// Outputs: 9
echo str_word_count($my_str);
?>
```

Replacing Text within Strings

The `str_replace()` replaces all occurrences of the search text within the target string.

Example

Run this code »

```
<?php
$my_str = 'If the facts do not fit the theory, change the facts.';

// Display replaced string
echo str_replace("facts", "truth", $my_str);
?>
```

The output of the above code will be:

```
If the truth do not fit the theory, change the truth.
```

You can optionally pass the fourth argument to the `str_replace()` function to know how many times the string replacements was performed, like this.

Example

Run this code »

```
<?php
$my_str = 'If the facts do not fit the theory, change the facts.';

// Perform string replacement
str_replace("facts", "truth", $my_str, $count);

// Display number of replacements performed
echo "The text was replaced $count times.";
?>
```

The output of the above code will be:

```
The text was replaced 2 times.
```

Reversing a String

The `strrev()` function reverses a string.

Example

Run this code »

```
<?php
$my_str = 'You can do anything, but not everything.';

// Display reversed string
echo strrev($my_str);
?>
```

	The output of the above code will be: .gnihtyreve ton tub ,gnihtyna od nac uoY
2.	**Design** a PHP application for College Management System with appropriate built-in functions and database.
3.	**Design** application to send an email using PHP.

PHP mail() Function

Example

Send a simple email:

```
<?php
// the message
$msg = "First line of text\nSecond line of text";

// use wordwrap() if lines are longer than 70 characters
$msg = wordwrap($msg,70);

// send email
mail("someone@example.com","My subject",$msg);
?>
```

Syntax

```
mail(to,subject,message,headers,parameters);
```

Parameter Values

Parameter	Description
to	Required. Specifies the receiver / receivers of the email
subject	Required. Specifies the subject of the email. **Note:** This parameter cannot contain any newline characters
message	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters. **Windows note:** If a full stop is found on the beginning of a line in the message, it might be removed. To solve this problem, replace the full stop with a double dot: <?php

	$txt = str_replace("\n.", "\n..", $txt); ?>
headers	Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n). **Note:** When sending an email, it must contain a From header. This can be set with this parameter or in the php.ini file.
parameters	Optional. Specifies an additional parameter to the sendmail program (the one defined in the sendmail_path configuration setting). (i.e. this can be used to set the envelope sender address when using sendmail with the -f sendmail option)

Technical Details

Return Value:	Returns the hash value of the *address* parameter, or FALSE on failure. **Note:** Keep in mind that even if the email was accepted for delivery, it does NOT mean the email is actually sent and received!
PHP Version:	4+
PHP Changelog:	PHP 7.2: The headers parameter also accepts an array PHP 5.4: Added header injection protection for the *headers* parameter. PHP 4.3.0: (Windows only) All custom headers (like From, Cc, Bcc and Date) are supported, and are not case-sensitive. PHP 4.2.3: The *parameter* parameter is disabled in safe mode PHP 4.0.5: The *parameter* parameter was added

More Examples

Send an email with extra headers:

```
<?php
$to = "somebody@example.com";
$subject = "My subject";
$txt = "Hello world!";
$headers = "From: webmaster@example.com" . "\r\n" .
"CC: somebodyelse@example.com";

mail($to,$subject,$txt,$headers);
?>
```

	Send an HTML email:
	
	`<?php`
	`$to = "somebody@example.com, somebodyelse@example.com";`
	`$subject = "HTML email";`
	
	`$message = "`
	`<html>`
	`<head>`
	`<title>HTML email</title>`
	`</head>`
	`<body>`
	`<p>This email contains HTML Tags!</p>`
	`<table>`
	`<tr>`
	`<th>Firstname</th>`
	`<th>Lastname</th>`
	`</tr>`
	`<tr>`
	`<td>John</td>`
	`<td>Doe</td>`
	`</tr>`
	`</table>`
	`</body>`
	`</html>`
	`";`
	
	`// Always set content-type when sending HTML email`
	`$headers = "MIME-Version: 1.0" . "\r\n";`
	`$headers .= "Content-type:text/html;charset=UTF-8" . "\r\n";`
	
	`// More headers`
	`$headers .= 'From: <webmaster@example.com>' . "\r\n";`
	`$headers .= 'Cc: myboss@example.com' . "\r\n";`
	
	`mail($to,$subject,$message,$headers);`
	`?>`

4.	**Summarize** about XML schema and XML Parsers and Validation. **XML Schema** **What is an XML Schema?** An XML Schema describes the structure of an XML document.

The XML Schema language is also referred to as XML Schema Definition (XSD).

XSD Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

The purpose of an XML Schema is to define the legal building blocks of an XML document:

- the elements and attributes that can appear in a document
- the number of (and order of) child elements
- data types for elements and attributes
- default and fixed values for elements and attributes

Why Learn XML Schema?

In the XML world, hundreds of standardized XML formats are in daily use.

Many of these XML standards are defined by XML Schemas.

XML Schema is an XML-based (and more powerful) alternative to DTD.

XML Parser

All major browsers have a built-in XML parser to access and manipulate XML.

The XML DOM (Document Object Model) defines the properties and methods for accessing and editing XML.

However, before an XML document can be accessed, it must be loaded into an XML DOM object.

All modern browsers have a built-in XML parser that can convert text into an XML DOM object.

Parsing a Text String

This example parses a text string into an XML DOM object, and extracts the info from it with JavaScript:

Example

```
<html>
<body>

<p id="demo"></p>

<script>
var text, parser, xmlDoc;

text = "<bookstore><book>" +
"<title>Everyday Italian</title>" +
"<author>Giada De Laurentiis</author>" +
"<year>2005</year>" +
"</book></bookstore>";

parser = new DOMParser();
xmlDoc = parser.parseFromString(text,"text/xml");

document.getElementById("demo").innerHTML =
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
</script>

</body>
</html>
```

OUTPUT

```
Everyday Italian
```

XML - Validation

Validation is a process by which an XML document is validated. An XML document is said to be valid if its contents match with the elements, attributes and associated document type declaration(DTD), and if the document complies with the constraints expressed in it. Validation is dealt in two ways by the XML parser. They are −

- Well-formed XML document
- Valid XML document

Well-formed XML Document

An XML document is said to be **well-formed** if it adheres to the following rules −

- Non DTD XML files must use the predefined character entities for **amp(&)**, **apos(single quote)**, **gt(>)**, **lt(<)**, **quot(double quote)**.
- It must follow the ordering of the tag. i.e., the inner tag must be closed before closing the outer tag.
- Each of its opening tags must have a closing tag or it must be a self ending tag.(<title>....</title> or <title/>).
- It must have only one attribute in a start tag, which needs to be quoted.
- **amp(&)**, **apos(single quote)**, **gt(>)**, **lt(<)**, **quot(double quote)** entities other than these must be declared.

Example

Following is an example of a well-formed XML document −

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>
<!DOCTYPE address
[
  <!ELEMENT address (name,company,phone)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT company (#PCDATA)>
  <!ELEMENT phone (#PCDATA)>
]>

<address>
  <name>Tanmay Patil</name>
  <company>TutorialsPoint</company>
  <phone>(011) 123-4567</phone>
</address>
```

The above example is said to be well-formed as −

- It defines the type of document. Here, the document type is **element** type.
- It includes a root element named as **address**.
- Each of the child elements among name, company and phone is enclosed in its self explanatory tag.
- Order of the tags is maintained.

Valid XML Document

If an XML document is well-formed and has an associated Document Type Declaration (DTD), then it is said to be a valid XML document.

CS8651- Internet Programming 2017Reg	
UNIT V - INTRODUCTION TO AJAX and WEB SERVICES	
JAX: Ajax Client Server Architecture – XML Http Request Object – Call Back Methods; **Web Services:** Introduction – Java web services Basics – Creating, Publishing, Testing and Describing a Web services WSDL) –Consuming a web service, Database Driven web service from an application – SOAP.	
	PART-A
No	**Questions**
1.	**Describe** AJAX Control Extender Toolkit. **What is the ASP.NET AJAX Control Toolkit?** The ASP.NET AJAX Control Toolkit is an open-source project built on top of the Microsoft ASP.NET AJAX framework. It is a joint effort between Microsoft and the ASP.NET AJAX community that provides a powerful infrastructure to write reusable, customizable and extensible ASP.NET AJAX extenders and controls, as well as a rich array of controls that can be used out of the box to create an interactive Web experience. They are designed using concepts that are familiar to ASP.NET Web Forms application developers. Using the Ajax Control Toolkit, you can build Ajax-enabled ASP.NET Web Forms applications and ASP.NET MVC Web applications by dragging the controls from the Visual Studio Toolbox onto a page. The Ajax Control Toolkit is an open-source project that is part of the CodePlex Foundation The AJAX Control Toolkit contains more than 30 controls that enable you to easily create rich, interactive web pages.
2.	**Discuss** the advantages of AJAX. <u>Advantages of AJAX</u> • Reduce the traffic travels between the client and the server. • Response time is faster so increases performance and speed. • You can use JSON (JavaScript Object Notation) which is alternative to XML. JSON is key value pair and works like an array. • You can use Firefox browser with an add-on called as Firebug to debug all Ajax calls. • Ready Open source JavaScript libraries available for use - JQuery, Prototype, Scriptaculous, etc.. • AJAX communicates over HTTP Protocol.
3.	**Identify** the role of a callback function in performing a partial page update in an AJAX application. **<u>Partial-page rendering with UpdatePanels</u>**

One of the most fascinating controls in the ASP.NET AJAX framework is the *UpdatePanel*. This new control replaces the need for a page to refresh during a postback. Only portions of a page, designated by the UpdatePanel, are updated. This technique is known as *partial-page rendering* and can be highly effective in improving the user experience.

6.1.1. Evolution of the UpdatePanel

For years, programming with the XMLHttpRequest object has been the most commonly used approach for communicating with the server from client-side script. The complexities involved in coding those types of applications scared away a lot of developers. To assist, the overall scripting model in ASP.NET 2.0 was significantly enhanced to introduce the idea of *script callbacks*—a way for server controls to communicate with client-side scripts between callbacks. This model was powerful because it offered access to the state of all the controls on the page during a callback. Unfortunately, many developers found the model difficult to work with, and numerous concerns were raised. The lack of support for passing complex types as parameters to the server (only strings were allowed) made the prototype too rigid and exposed its limitations. Developers began to look elsewhere for solutions.

In an effort to address these concerns, members of the ASP.NET team began work on a communication library built on top of the callbacks. The primary objective of the library was to simplify the use of callbacks and to provide a rich set of APIs for enabling the exchange of complex and simple types between the server and client. From this library came a control called the *RefreshPanel*. The purpose of the RefreshPanel was to offer a server control that refreshed the contents of a page without a page refresh. Out of this hard work, the UpdatePanel emerged, with deeper integration into the page lifecycle and a more transparent footprint on the page.

NOTE

A *callback* is a piece of code that is passed in as a parameter or argument to other code. The other piece of code can call the callback code (usually a function) at any time, even numerous times, in response to some processing.

4.	**Differentiate** AJAX forms with HTML5 forms. **AJAX** is the name of a communication architecture between web pages and server side. **JQuery** is a javascript library that is written for unifying JS method calls (regarding DOM manipulation, String and Array functions, DOM queries...etc.) in all browsers. **HTML5** is the rendering specification to be implemented by all browser providers. For this rendering to work JS Engine should also be updated, so HTML5 also means a JS engine with new features like drawing on a canvas.
5.	What is XML Http Request object? **List** its properties.

The XMLHttpRequest Object

With the XMLHttpRequest object you can update parts of a web page, without reloading the whole page.

The XMLHttpRequest Object

The XMLHttpRequest object is used to exchange data with a server behind the scenes.

The XMLHttpRequest object is **the developers dream**, because you can:

- Update a web page without reloading the page
- Request data from a server after the page has loaded
- Receive data from a server after the page has loaded
- Send data to a server in the background

XMLHttpRequest Object Methods

Method	Description
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(method,url,async,uname,pswd)	Specifies the type of request, the URL, if the request should be handled asynchronously or not, and other optional attributes of a request

	method: the type of request: GET or POST url: the location of the file on the server async: true (asynchronous) or false (synchronous)
send(string)	send(string) Sends the request off to the server. string: Only used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

XMLHttpRequest Object Properties

Property	**Description**
readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseXML	Returns the response data as XML data
statusText	Returns the status-text (e.g. "Not Found" or "OK")

6. **Summarize** the need of SOAP and show its structure.

XML Soap

- SOAP stands for **S**imple **O**bject **A**ccess **P**rotocol
- SOAP is an application communication protocol
- SOAP is a format for sending and receiving messages
- SOAP is platform independent

- SOAP is based on XML
- SOAP is a W3C recommendation

Why SOAP?

It is important for web applications to be able to communicate over the Internet.

The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.

A SOAP message is an ordinary XML document containing the following elements:

- An Envelope element that identifies the XML document as a SOAP message
- A Header element that contains header information
- A Body element that contains call and response information
- A Fault element containing errors and status information

7.

Can you **develop** the service end point interface in RPC?

Example

Sample Java Beans service endpoint implementation and interface

The following example illustrates a simple explicit Java Beans service endpoint implementation and the associated service endpoint interface.

```
/** This is an excerpt from the service implementation file,
EchoServicePortTypeImpl.java  package com.ibm.was.wssample.echo;
 import java.io.ByteArrayInputStream;
 import java.io.ByteArrayOutputStream;
 import javax.xml.bind.JAXBContext;
 import javax.xml.bind.Marshaller;
 import javax.xml.bind.Unmarshaller;
 import javax.xml.transform.stream.StreamSource;

@javax.jws.WebService(serviceName = "EchoService",
                      endpointInterface =
"com.ibm.was.wssample.echo.EchoServicePortType",

targetNamespace="http://com/ibm/was/wssample/echo/",
                      portName="EchoServicePort")
public class EchoServicePortTypeImpl implements EchoServicePortType {

                public EchoServicePortTypeImpl() {
                }

                public String invoke(String obj) {
                        String str;
                        ....
                        str = obj;
                        ....

                        return str;
                }

}
/** This is a sample EchoServicePortType.java service interface */
 import javax.jws.WebMethod;
 import javax.jws.WebParam;
 import javax.jws.WebResult;
 import javax.jws.WebService;
 import javax.xml.ws.*;


@WebService(name = "EchoServicePortType",
                   targetNamespace =
"http://com/ibm/was/wssample/echo/",
                   wsdlLocation="WEB-INF/wsdl/Echo.wsdl")
public interface EchoServicePortType {


    /** ...the method process ...*/
        @WebMethod
    @WebResult(name = "response", targetNamespace =
"http://com/ibm/was/wssample/echo/")
    @RequestWrapper(localName = "invoke", targetNamespace =
"http://com/ibm/was/wssample/echo/", className =
"com.ibm.was.wssample.echo.Invoke")
    @ResponseWrapper(localName = "echoStringResponse",
targetNamespace = "http://com/ibm/was/wssample/echo/", className =
"com.ibm.was.wssample.echo.EchoStringResponse")
    public String invoke(
        @WebParam(name = "arg0", targetNamespace =
"http://com/ibm/was/wssample/echo/")
        String arg0);

}
```

Sample Provider endpoint implementation

The following example illustrates a simple Provider service endpoint interface for a Java class.

```
package jaxws.provider.source;
 import javax.xml.ws.Provider;
 import javax.xml.ws.WebServiceProvider;
 import javax.xml.transform.Source;

@WebServiceProvider() public class SourceProvider implements
Provider<Source> {

    public Source invoke(Source data) {
        return data;
    }
}
```

8. **List** any four examples of web services.

<u>A Web Service Example</u>

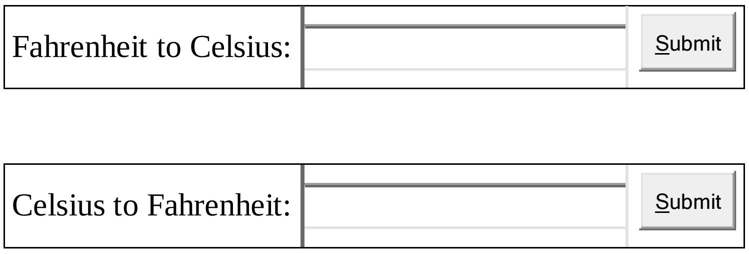
In the following example we will use ASP.NET to create a simple Web Service that converts the temperature from Fahrenheit to Celsius, and vice versa:

```
<%@ WebService Language="VBScript" Class="TempConvert" %>

Imports System
Imports System.Web.Services

Public Class TempConvert :Inherits WebService

<WebMethod()> Public Function FahrenheitToCelsius(ByVal
Fahrenheit As String) As String
  dim fahr
  fahr=trim(replace(Fahrenheit,",","."))
  if fahr="" or IsNumeric(fahr)=false then return "Error"
  return ((((fahr) - 32) / 9) * 5)
end function

<WebMethod()> Public Function CelsiusToFahrenheit(ByVal Celsius
As String) As String
  dim cel
  cel=trim(replace(Celsius,",","."))
  if cel="" or IsNumeric(cel)=false then return "Error"
  return ((((cel) * 9) / 5) + 32)
end function

end class
```

This document is saved as an .asmx file. This is the ASP.NET file extension for XML Web Services.

Put the Web Service on Your Web Site

Using a form and the HTTP POST method, you can put the web service on your site, like this:



<u>code to add the Web Service to a web page:</u>

```
<form action='tempconvert.asmx/FahrenheitToCelsius'
method="post" target="_blank">
<table>
  <tr>
    <td>Fahrenheit to Celsius:</td>
    <td>
    <input class="frmInput" type="text" size="30" name="Fahrenhei
t">
    </td>
  </tr>
  <tr>
    <td></td>
    <td align="right">
     <input type="submit" value="Submit" class="button">
     </td>
  </tr>
</table>
</form>

<form action='tempconvert.asmx/CelsiusToFahrenheit'
method="post" target="_blank">
<table>
  <tr>
    <td>Celsius to Fahrenheit:</td>
    <td>
    <input class="frmInput" type="text" size="30" name="Celsius">
    </td>
  </tr>
  <tr>
    <td></td>
    <td align="right">
    <input type="submit" value="Submit" class="button">
    </td>
  </tr>
</table>
</form>
```

Substitute the "tempconvert.asmx" with the address of your web service like:

http://www.example.com/xml/tempconvert.asmx

9. **Discover** an example for web service registry along with its functions.

Web Services Discovery provides access to software systems over the Internet using standard protocols. In the most basic scenario there is a *Web Service Provider* that publishes a service and a *Web Service Consumer* that uses this service. Web Service Discovery is the process of finding suitable web services for a given task.[1]

Publishing a web service involves creating a software artifact and making it accessible to potential consumers. Web service providers augment a service endpoint

	interface with an interface description using the Web Services Description Language (WSDL) so that a consumer can use the service. **Universal Description, Discovery, and Integration** (**UDDI**) is an XML-based registry for business internet services. A provider can explicitly register a service with a *Web Services Registry* such as UDDI or publish additional documents intended to facilitate discovery such as Web Services Inspection Language (WSIL) documents. The service users or consumers can search web services manually or automatically. The implementation of UDDI servers and WSIL engines should provide simple search APIs or web-based GUI to help find Web services. Web services may also be discovered using multicast mechanisms like WS-Discovery, thus reducing the need for centralized registries in smaller networks.
10.	**Analyze** the need for web service. **We should use web services as it comes with various advantages listed below** **Re-usability** Once we develop some **business logic**,we can make it **reuse** for other applications **Example:** If **10 different applications** requires to **use** our **logic** We can expose our **logic** over a **network** as a **web service** So all the **10 different application** can **access** it from the **network**. **Interoperability** It provides the freedom for a **developers** to choose whatever the **technology** they want to use for development. Web services uses a **set of standards** and **protocols** and enable us to achieve interoperability. Hence applications developed in **Java,Mainframe,Ruby** or any other technology can call the web service and use it. **Loosely coupled** Web service exist **independent** of the other parts of the application that uses it. So any **changes** to the **application** can be made **without affecting** the web service. **Deployability** It is very **easy** to **deploy** the **web application** as they are deployed over standard internet technologies.
11.	**Give** the uses of WSDL along with its definition.

WSDL stands for Web Services Description Language. It is the standard format for describing a web service. WSDL was developed jointly by Microsoft and IBM.

Features of WSDL

- WSDL is an XML-based protocol for information exchange in decentralized and distributed environments.
- WSDL definitions describe how to access a web service and what operations it will perform.
- WSDL is a language for describing how to interface with XML-based services.
- WSDL is an integral part of Universal Description, Discovery, and Integration (UDDI), an XML-based worldwide business registry.
- WSDL is the language that UDDI uses.
- WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'.

WSDL Usage

WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.

12. **Compare** SOAP and HTTP.

Difference between SOAP and HTTP

SOAP	HTTP
➢ SOAP was originally defined as S- Simple O- Object A-Access P-protocol. ➢ It is a protocol specification which is used for exchanging structured information. ➢ It is used in the implementation of web services in computer-based networks. ➢ SOAP for its message format relies on XML Information set and sometimes relies on other application layer protocols as well, such as Hypertext Transfer Protocol (HTTP) or Simple	➢ The HTTP or Hypertext Transfer Protocol (HTTP) is an application protocol which is used for distributed, collaborative and hypermedia information systems. ➢ HTTP is widely regarded as the foundation of data communication for the World Wide Web (WWW). ➢ Hypertext is a structured text that uses logical links or hyperlinks between those nodes that containing text. HTTP is the protocol

Mail Transfer Protocol (SMTP). ➢ It is used for message negotiation and transmission mainly. ➢ SOAP forms the foundation layer of a web services protocol stack.	for exchanging or transferring hypertext. ➢ The standards development of HTTP when it was innovated was coordinated by the Internet Engineering Task Force and the World Wide Web Consortium also called as W3C.

13.	**Summarize** the need for enhancing security in web services. **Definition - What does *Web Services Security (WS Security)* mean?** Web Services Security (WS Security) is a specification that defines how security measures are implemented in web services to protect them from external attacks. It is a set of protocols that ensure security for SOAP-based messages by implementing the principles of confidentiality, integrity and authentication. Because Web services are independent of any hardware and software implementations, WS-Security protocols need to be flexible enough to accommodate new security mechanisms and provide alternative mechanisms if an approach is not suitable. Because SOAP-based messages traverse multiple intermediaries, security protocols need to be able to identify fake nodes and prevent data interpretation at any nodes. WS-Security combines the best approaches to tackle different security problems by allowing the developer to customize a particular security solution for a part of the problem. For example, the developer can select digital signatures for non-repudiation and Kerberos for authentication.
14.	**Name** the types of indicators along with the definition. **Web Services Security (WSS)** Web Services Security (WSS or WS-Security) describes enhancements to SOAP messaging in order to provide quality of protection through message integrity, and single message authentication. These mechanisms can be used to accommodate a wide variety of security models and encryption technologies. The scope of the Web Services Security Technical Committee is the support of security mechanisms in the following areas:

	Using XML Signature to provide SOAP message integrity for Web Services Using XML Encryption to provide SOAP message confidentiality for Web Services Attaching and/or referencing security tokens in headers of SOAP messages. Options include: Username token SAML XrML Kerberos X.509 Carrying security information for potentially multiple, designated actors Associating signatures with security tokens Each of the security mechanisms will use implementation and language neutral XML formats defined in XML Schema.
15.	**Classify** the basic concepts behind JAX-RPC technology. JAX-RPC Java APIs for XML-based Remote Procedure Call (JAX-RPC) help with Web service interoperability and accessibility by defining Java APIs that Java applications use to develop and access Web services. JAX-RPC fully embraces the heterogeneous nature of Web services -- it allows a JAX-RPC client to talk to another Web service deployed on a different platform and coded in a different language. Similarly, it also allows clients on other platforms and coded in different languages to talk to a JAX-RPC service. JAX-RPC also defines the mapping between WSDL service descriptions and Java interfaces. Th the JAX-RPC technology and describes its client and server programming models. JAX-RPC hides the complexity of underlying protocols and message-level processing from application developers crafting Web services using the Java 2 platform. The API combines XML with Remote Procedure Call (RPC), which is a mechanism enabling clients to execute procedures on distributed or remote systems, so that developers can build Web services and clients. The JAX-RPC remote procedure calls are represented by an XML infoset and they are carried over a network transport. While the JAX-RPC APIs rely on a XML-based protocol and a network transport, the APIs themselves are independent of a specific protocol or transport. The current JAX-RPC implementation relies on the SOAP 1.1 protocol and HTTP 1.1 network transport.
16.	**What are the benefits of UDDI?** Problems the UDDI specification can help to solve: • Making it possible to discover the right business from the millions currently online • Defining how to enable commerce once the preferred business is discovered • Reaching new customers and increasing access to current customers • Expanding offerings and extending market reach

	• Solving customer-driven need to remove barriers to allow for rapid participation in the global Internet economy • Describing services and business processes programmatically in a single, open, and secure environment
17.	**What are the core elements of UDDI?** UDDI defines four core data elements within the data model: • businessEntity (modeling business information) • businessService (describing a service) • tModel (describing specifications, classifications, or identifications) • binding Template (mapping between a businessService and the set of tModels that describe its technical fingerprint)
18.	**Rewrite** the definition for UDDI. UDDI is an XML-based standard for describing, publishing, and finding web services. • UDDI stands for **Universal Description, Discovery, and Integration.** • UDDI is a specification for a distributed registry of web services. • UDDI is a platform-independent, open framework. • UDDI can communicate via SOAP, CORBA, Java RMI Protocol. • UDDI uses Web Service Definition Language(WSDL) to describe interfaces to web services. • UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services. • UDDI is an open industry initiative, enabling businesses to discover each other and define how they interact over the Internet. UDDI has two sections − • A registry of all web service's metadata, including a pointer to the WSDL description of a service. • A set of WSDL port type definitions for manipulating and searching that registry.
19.	**Give** the usage of UDDI in web service. UDDI is an XML-based standard for describing, publishing, and finding web services. • UDDI stands for **Universal Description, Discovery, and Integration.** • UDDI is a specification for a distributed registry of web services. • UDDI is a platform-independent, open framework.

	• UDDI can communicate via SOAP, CORBA, Java RMI Protocol. • UDDI uses Web Service Definition Language(WSDL) to describe interfaces to web services. • UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services. • UDDI is an open industry initiative, enabling businesses to discover each other and define how they interact over the Internet. UDDI has two sections − • A registry of all web service's metadata, including a pointer to the WSDL description of a service. • A set of WSDL port type definitions for manipulating and searching that registry.
20.	**Define** WSDL. WSDL stands for Web Services Description Language. It is the standard format for describing a web service. WSDL was developed jointly by Microsoft and IBM. **Features of WSDL** • WSDL is an XML-based protocol for information exchange in decentralized and distributed environments. • WSDL definitions describe how to access a web service and what operations it will perform. • WSDL is a language for describing how to interface with XML-based services. • WSDL is an integral part of Universal Description, Discovery, and Integration (UDDI), an XML-based worldwide business registry. • WSDL is the language that UDDI uses. • WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'. **WSDL Usage** WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.
	PART-B
1.	(i) **Describe** in detail about the AJAX architecture.

(ii) **List out** the call back methods.

2.

(i) **Analyze** various concepts of RPC.

Remote Procedure Call (RPC)

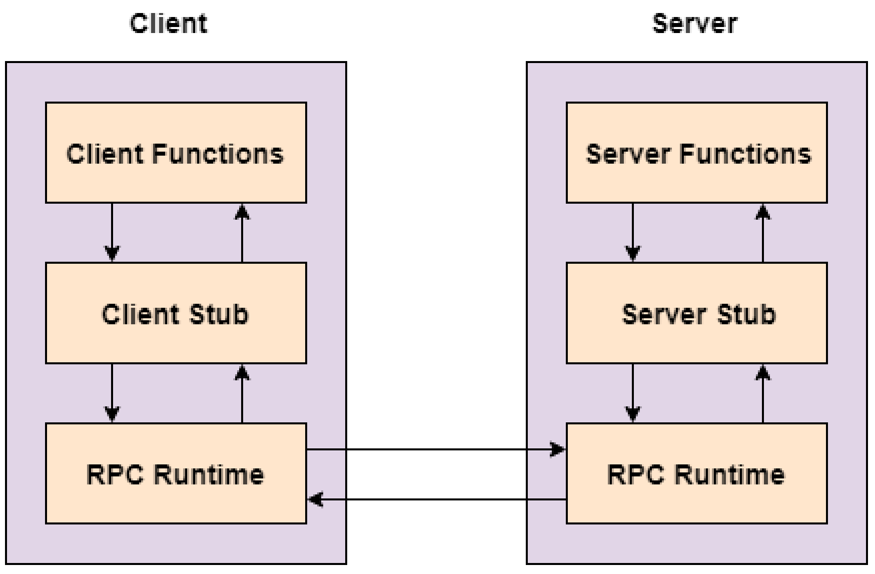
A remote procedure call is an interprocess communication technique that is used for client-server based applications. It is also known as a subroutine call or a function call.

A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request, it sends the required response back to the client. The client is blocked while the server is processing the call and only resumed execution after the server is finished.

The sequence of events in a remote procedure call are given as follows:

- The client stub is called by the client.
- The client stub makes a system call to send the message to the server and puts the parameters in the message.
- The message is sent from the client to the server by the client's operating system.
- The message is passed to the server stub by the server operating system.
- The parameters are removed from the message by the server stub.
- Then, the server procedure is called by the server stub.

A diagram that demonstrates this is as follows:



Advantages of Remote Procedure Call

Some of the advantages of RPC are as follows:

- Remote procedure calls support process oriented and thread oriented models.
- The internal message passing mechanism of RPC is hidden from the user.
- The effort to re-write and re-develop the code is minimum in remote procedure calls.
- Remote procedure calls can be used in distributed environment as well as the local environment.
- Many of the protocol layers are omitted by RPC to improve performance.

Disadvantages of Remote Procedure Call

Some of the disadvantages of RPC are as follows:

- The remote procedure call is a concept that can be implemented in different ways. It is not a standard.
- There is no flexibility in RPC for hardware architecture. It is only interaction based.
- There is an increase in costs because of remote procedure call.

(ii) **Explain** the basic concepts behind JAX-RPC.

JAX-RPC stands for Java API for XML-based RPC. It's an API for building Web services and clients that used remote procedure calls (RPC) and XML. Often used in a distributed client/server model, an RPC mechanism enables clients to execute procedures on other systems.

In JAX-RPC, a remote procedure call is represented by an XML-based protocol such as SOAP. The SOAP specification defines envelope structure, encoding rules, and a convention for representing remote procedure calls and responses. These calls and responses are transmitted as SOAP messages over HTTP. In this release, JAX-RPC relies on SOAP 1.1 and HTTP 1.1.

Although JAX-RPC relies on complex protocols, the API hides this complexity from the application developer. On the server side, the developer specifies the remote procedures by defining methods in an interface written in the Java programming language. The developer also codes one or more classes that implement those methods. Client programs are also easy to code. A client creates a proxy, a local object representing the service, and then simply invokes methods on the proxy.

With JAX-RPC, clients and Web services have a big advantage--the platform independence of the Java programming language. In addition, JAX-RPC is not restrictive: a JAX-RPC client can access a Web service that is not running on the Java platform and vice versa. This flexibility is possible because JAX-RPC uses technologies defined by the World Wide

Web Consortium (W3C): HTTP, SOAP, and the Web Service Description Language (WSDL). WSDL specifies an XML format for describing a service as a set of endpoints operating on messages.

3. **Explain** in detail with an example of Java Web Services.

With a simple example illustrate the steps to create a java web service. (NOV/DEC 2012)

Writing a java web service

Currency conversion Service

- ◆ Writing a server for a service using JWSDP 1.3 tools
- ◆ Application: currency converter
- ■ Three operations:
- ● fromDollars
- ● fromEuros
- ● fromYen
- ■ Input: value in specified currency
- ■ Output: object containing input value and equivalent values in other two currencies

Writing server software

1. Write service endpoint interface
• May need to write additional classes representing data structures
2. Write class implementing the interface
3. Compile classes
4. Create configuration files and run JWSDP tools to create web service
5. Deploy web service to Tomcat

service endpoint interface

- ◆ The Web service endpoint interface is used to define the 'Web services methods'.
- ◆ A Web service endpoint interface must conform to the rules of a JAX-RPC service definition
interface.
- ◆ a service endpoint interface (SEI) that defines the interface of the web service.
- ◆ Configuration files are XML files that can be changed as needed. Developers can use
configuration files to change settings without recompiling applications. Administrators can use
configuration files to set policies that affect how applications run on their computers.
- ◆ config.xml : Defines the URL for WSDL file location. Each Web services has a corresponding
WSDL (Web service Definition Language) document.

JWSDP: Server

Rules for Service endpoint interface

- ■ Must extend java.rmi.Remote
- ● declares a set of methods that may be invoked from a remote Java Virtual Machine(JVM)
- ■ Every method must throw java.rmi.RemoteException

■ Parameter/return value data types are restricted
■ No public static final declarations (global constants) It must not have constant declarations
◆ **Allowable parameter/return value data types**
■ Java primitives (int, boolean, *etc*.)
■ Primitive wrapper classes (Integer, *etc*.)
■ String, Date, Calendar, BigDecimal, BigInteger
■ java.xml.namespace.QName, java.net.URI
■ Struct: class consisting entirely of public instance variables
■ Array of any of the above
◆ **Struct for currency converter app (data type for return values)**

```
package myCurCon;

public class ExchangeValues {
    public double dollars;
    public double euros;
    public double yen;
}
```

◆ **Service endpoint interface**

```
package myCurCon;

public interface CurCon extends java.rmi.Remote {
    public ExchangeValues fromDollars(double dollars)
        throws java.rmi.RemoteException;
    public ExchangeValues fromEuros(double euros)
        throws java.rmi.RemoteException;
    public ExchangeValues fromYen(double yen)
        throws java.rmi.RemoteException;
}
```

◆ Three files ExchangeValues.java, CurCon.java and CurConImpl.java written for the web s
◆ Class CurConImpl contains methods implements sevice endpoint interface, for *example:*

```
public ExchangeValues fromDollars(double dollars)
    throws java.rmi.RemoteException
{
    ExchangeValues ev = new ExchangeValues();
    ev.dollars = dollars;
    ev.euros = dollars * dollar2euro;
    ev.yen = dollars * dollar2yen;
    return ev;
}
```

Packaging server software

- Configuration file input to wscompile to create server

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration
  xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
  <service
    name="HistoricCurrencyConverter"
    targetNamespace="http://tempuri.org/wsdl"
    typeNamespace="http://tempuri.org/types"
    packageName="myCurCon">
    <interface name="myCurCon.CurCon" />
  </service>
</configuration>
```

- Configuration file for web service

```
<?xml version="1.0" encoding="UTF-8"?>
<webServices
  xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/dd"
  version="1.0"
  targetNamespaceBase="http://tempuri.org/wsdl"
  typeNamespaceBase="http://tempuri.org/types"
```

- Configuration file for web service

```
  urlPatternBase="/converter">
```

Context path

```
  <endpoint
    name="CurrConverter"
    displayName="Currency Converter"
    description=
      "Converts between dollars, euros, and yen."
    interface="myCurCon.CurCon"
    model="/WEB-INF/model.xml.gz"
    implementation="myCurCon.CurConImpl"/>
```

Like servlet in web.xml

```
  <endpointMapping
    endpointName="CurrConverter"
    urlPattern="/currency" />
```

Like servlet-mapping in web.xml

```
</webServices>
```

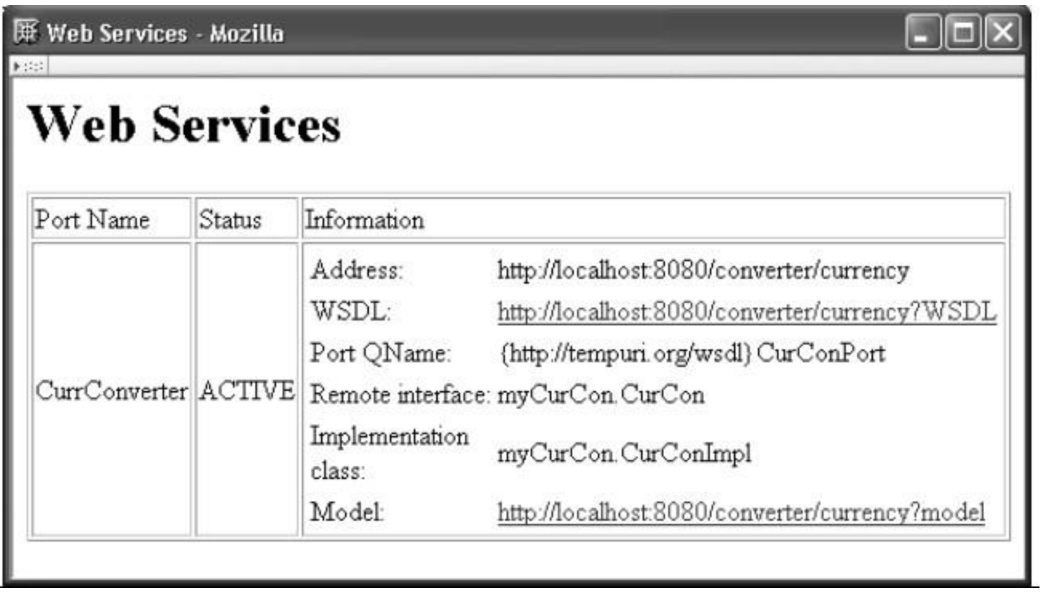
- Also need a minimal web.xml

```
<web-app
  xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
  version="2.4">

  <display-name>Historic Currency Converter</display-name>
  <description>
    This web service converts between three currencies using their
    exchange rates as of a fixed date.
  </description>
</web-app>
```

Run jar and wsdeploy to create a Web Archive (WAR) file converter.war

■ Name must match urlPatternBase value

- jaxrpc-ri.xml: Defines the various end points for referencing a Web service.
- *wscompile*: The wscompile tool generates stubs, and WSDL files used in JAX-RPC clients and
services. The tool reads as input a configuration file and either a WSDL file or an RMI interface
that defines the service.
- *wsdeploy*: Reads a WAR file (something like Jar file) and the jaxrpc-ri.xml file and then
generates another WAR file that is ready for deployment
- Write service endpoint interface
  - May need to write additional classes representing data structures
- Write class implementing the interface
- Compile classes
- Create configuration files and run JWSDP tools to create web service
- Deploy web service to Tomcat
- Just copy converter.war to Tomcat webapps directory
  - May need to use Manager app to deploy
  - Enter converter.war in "WAR or Directory URL" text box
- Testing success:
  - Visit http://localhost:8080/converter/currency



Web Services

Port Name	Status	Information	
CurrConverter	ACTIVE	Address:	http://localhost:8080/converter/currency
		WSDL:	http://localhost:8080/converter/currency?WSDL
		Port QName:	{http://tempuri.org/wsdl} CurConPort
		Remote interface:	myCurCon.CurCon
		Implementation class:	myCurCon.CurConImpl
		Model:	http://localhost:8080/converter/currency?model

4. **Discuss** in detail the architecture of web services.

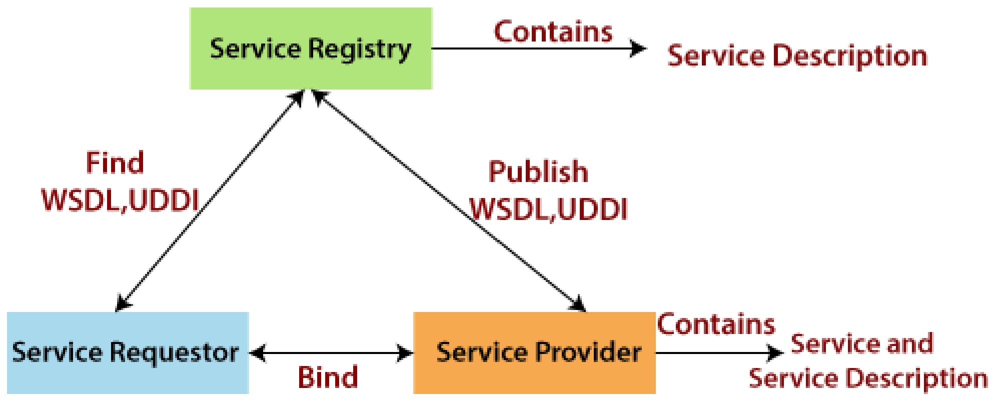
Architecture of Web Services

The Web Services architecture describes how to instantiate the elements and implement the operations in an interoperable manner.

The architecture of web service interacts among three roles: **service provider, service requester,** and **service registry**. The interaction involves the three operations: **publish, find,** and **bind**. These operations and roles act upon the **web services artifacts**. The web service artifacts are the web service software module and its description.

The service provider hosts a network-associable module (web service). It defines a service description for the web service and publishes it to a service requestor or service registry. These service requestor uses a find operation to retrieve the service description locally or from the service registry. It uses the service description to bind with the service provider and invoke with the web service implementation.

The following figure illustrates the operations, roles, and their interaction.



Web Service Roles, Operations and Artifacts

Roles in a Web Service Architecture

There are three roles in web service architecture:

- Service Provider
- Service Requestor
- Service Registry

Service Provider

From an architectural perspective, it is the platform that hosts the services.

Service Requestor

Service requestor is the application that is looking for and invoking or initiating an interaction with a service. The browser plays the requester role, driven by a consumer or a program without a user interface.

Service Registry

Service requestors find service and obtain binding information for services during development.

Operations in a Web Service Architecture

Three behaviors that take place in the microservices:

- Publication of service descriptions **(Publish)**
- Finding of services descriptions **(Find)**
- Invoking of service based on service descriptions **(Bind)**

Publish: In the publish operation, a service description must be published so that a service requester can find the service.

Find: In the find operation, the service requestor retrieves the service description directly. It can be involved in two different lifecycle phases for the service requestor:

- At design, time to retrieve the service's interface description for program development.
- And, at the runtime to retrieve the service's binding and location description for invocation.

Bind: In the bind operation, the service requestor invokes or initiates an interaction with the service at runtime using the binding details in the service description to locate, contact, and invoke the service.

Artifacts of the web service

There are two artifacts of web services:

- Service
- Service Registry

Service: A service is an **interface** described by a service description. The service description is the implementation of the service. A service is a software module deployed on network-accessible platforms provided by the service provider. It interacts with a service requestor. Sometimes it also functions as a requestor, using other Web Services in its implementation.

Service Description: The service description comprises the details of the **interface** and **implementation** of the service. It includes its **data types, operations, binding information,** and **network location**. It can also categorize other metadata to enable discovery and utilize by service requestors. It can be published to a service requestor or a service registry.

5.	(i) **Deduce** any two elements of WSDL.

WSDL (Web services description language)

A web service cannot be used if it cannot be found. The client invoking the web service should know where the web service actually resides.

Secondly, the client application needs to know what the web service actually does, so that it can invoke the right web service. This is done with the help of the WSDL, known as the Web services description language. The WSDL file is again an XML-based file which basically tells the client application what the web service does. By using the WSDL document, the client application would be able to understand where the web service is located and how it can be utilized.

Web Service Example

An example of a WSDL file is given below.

```
<definitions>
   <message name="TutorialRequest">
      <part name="TutorialID" type="xsd:string"/>
   </message>

   <message name="TutorialResponse">
      <part name="TutorialName" type="xsd:string"/>
   </message>

   <portType name="Tutorial_PortType">
      <operation name="Tutorial">
         <input message="tns:TutorialRequest"/>
         <output message="tns:TutorialResponse"/>
      </operation>
   </portType>

   <binding name="Tutorial_Binding" type="tns:Tutorial_PortType">
      <soap:binding style="rpc"
         transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="Tutorial">
         <soap:operation soapAction="Tutorial"/>
         <input>
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encodi
ng/"
               namespace="urn:examples:Tutorialservice"
               use="encoded"/>
         </input>

           <output>
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encodi
ng/"
               namespace="urn:examples:Tutorialservice"
               use="encoded"/>
         </output>
      </operation>
   </binding>
</definitions>
```

(ii) **Explain** the steps for writing web service.

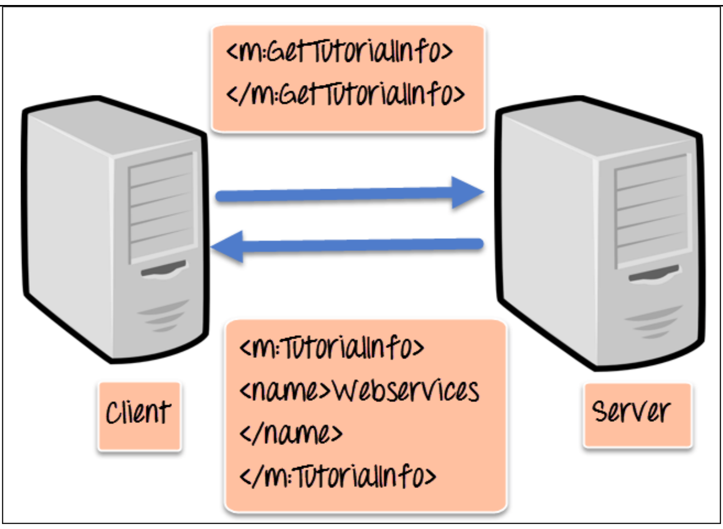
6.

Describe briefly about the elements of WSDL.

he important aspects to note about the above WSDL declaration are as follows;

1. **<message>** - The message parameter in the WSDL definition is used to define the different data elements for each operation performed by the web service. So in the example above, we have 2 messages which can be exchanged between the web service and the client application, one is the "TutorialRequest", and the other is the "TutorialResponse" operation. The TutorialRequest contains an element called "TutorialID" which is of the type string. Similarly, the TutorialResponse operation contains an element called "TutorialName" which is also a type string.
2. **<portType>** - This actually describes the operation which can be performed by the web service, which in our case is called Tutorial. This operation can take 2 messages; one is an input message, and the other is the output message.
3. **<binding>** - This element contains the protocol which is used. So in our case, we are defining it to use http (**http://schemas.xmlsoap.org/soap/http**). We also specify other details for the body of the operation, like the

	namespace and whether the message should be encoded.
7.	(i) **Summarize** on the structure of SOAP. **SOAP (Simple Object Access Protocol)** SOAP is known as a transport-independent messaging protocol. SOAP is based on transferring XML data as SOAP Messages. Each message has something which is known as an XML document. Only the structure of the XML document follows a specific pattern, but not the content. The best part of Web services and SOAP is that its all sent via HTTP, which is the standard web protocol. Here is what a SOAP message consists of ○ Each SOAP document needs to have a root element known as the <Envelope> element. The root element is the first element in an XML document. ○ The "envelope" is in turn divided into 2 parts. The first is the header, and the next is the body. ○ The header contains the routing data which is basically the information which tells the XML document to which client it needs to be sent to. ○ The body will contain the actual message. The diagram below shows a simple example of the communication via SOAP.



(ii) **Describe** briefly about SOAP & HTTP.

8.

(i) **Demonstrate** the building blocks of SOAP.

XML Soap

- SOAP stands for **S**imple **O**bject **A**ccess **P**rotocol
- SOAP is an application communication protocol
- SOAP is a format for sending and receiving messages
- SOAP is platform independent
- SOAP is based on XML
- SOAP is a W3C recommendation

Why SOAP?

It is important for web applications to be able to communicate over the Internet.

The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.

SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

SOAP Building Blocks

A SOAP message is an ordinary XML document containing the following elements:

- An Envelope element that identifies the XML document as a SOAP message
- A Header element that contains header information
- A Body element that contains call and response information
- A Fault element containing errors and status information

Syntax Rules

Here are some important syntax rules:

- A SOAP message MUST be encoded using XML
- A SOAP message MUST use the SOAP Envelope namespace
- A SOAP message must NOT contain a DTD reference
- A SOAP message must NOT contain XML Processing Instructions

(ii) **Classify** the encoding of struct data and array.

SOAP - Encoding

SOAP includes a built-in set of rules for encoding data types. It enables the SOAP message to indicate specific data types, such as integers, floats, doubles, or arrays.

- SOAP data types are divided into two broad categories − scalar types and compound types.
- Scalar types contain exactly one value such as a last name, price, or product description.
- Compound types contain multiple values such as a purchase order or a list of stock quotes.
- Compound types are further subdivided into arrays and structs.

Compound Types

SOAP arrays have a very specific set of rules, which require that you specify both the element type and array size. SOAP also supports multidimensional arrays, but not all SOAP implementations support multidimensional functionality.

To create an array, you must specify it as an *xsi:type* of array. The array must also include an *arrayType* attribute. This attribute is required to specify the data type for the contained elements and the dimension(s) of the array.

For example, the following attribute specifies an array of 10 double values –

```
arrayType = "xsd:double[10]"
```

In contrast, the following attribute specifies a two-dimensional array of strings –

```
arrayType = "xsd:string[5,5]"
```

Here is a sample SOAP response with an array of double values –

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-
envelope"
   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

   <SOAP-ENV:Body>
      <ns1:getPriceListResponse
         xmlns:ns1 = "urn:examples:pricelistservice"
         SOAP-ENV:encodingStyle =
"http://www.w3.org/2001/12/soap-encoding">

         <return xmlns:ns2 =
"http://www.w3.org/2001/09/soap-encoding"
            xsi:type = "ns2:Array" ns2:arrayType =
"xsd:double[2]">
            <item xsi:type = "xsd:double">54.99</item>
            <item xsi:type = "xsd:double">19.99</item>
         </return>
      </ns1:getPriceListResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Structs contain multiple values, but each element is specified with a unique accessor element. For example, consider an item within a product catalog. In this case, the struct might contain a product SKU, product name, description, and price. Here is how such a struct would be represented in a SOAP message –

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-
envelope"
   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

   <SOAP-ENV:Body>
      <ns1:getProductResponse
         xmlns:ns1 = "urn:examples:productservice"
         SOAP-ENV:encodingStyle =
"http://www.w3.org/2001/12/soap-encoding">

         <return xmlns:ns2 = "urn:examples" xsi:type =
"ns2:product">
            <name xsi:type = "xsd:string">Red Hat
Linux</name>
            <price xsi:type = "xsd:double">54.99</price>
            <description xsi:type = "xsd:string">
               Red Hat Linux Operating System
            </description>
            <SKU xsi:type = "xsd:string">A358185</SKU>
         </return>
      </ns1:getProductResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

9. **Analyze** the various steps in database driven web service with some example.

Overview of Database Web Services

Web services enable application-to-application interaction over the Web, regardless of platform, language, or data formats. The key ingredients, including Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI), have been adopted across the entire software industry. Web services usually refer to services implemented and deployed in middle-tier application servers. However, in heterogeneous and disconnected environments, there is an increasing need to access stored procedures, as well as data and metadata, through Web services interfaces.

The Database Web services technology is a database approach to Web services. It works in the following two directions:

- Accessing database resources as a Web service
- Consuming external Web services from the database

Oracle Database can access Web services through PL/SQL packages and Java classes deployed within the database. Turning Oracle Database into a Web service provider leverages investment in Java stored procedures, PL/SQL packages, predefined SQL queries, and data manipulation language (DML). Conversely, consuming external Web services from the database, together with integration with the SQL engine, enables Enterprise Information Integration.

Using Oracle Database as Web Services Provider

Web Services use industry-standard mechanisms to provide easy access to remote content and applications, regardless of the platform and location of the provider and implementation and data format. Client applications can query and retrieve data from Oracle Database and call stored procedures using standard Web service protocols. There is no dependency on Oracle-specific database connectivity protocols. This approach is highly beneficial in heterogeneous, distributed, and disconnected environments.

You can call into the database from a Web service, using the database as a service provider. This enables you to leverage existing or new SQL, PL/SQL, Java stored procedures, or Java classes within Oracle Database. You can access and manipulate database tables from a Web service client.

10.

Illustrate on web services for writing web service client along with the description of WSDL.

Creating a Web Service Client

Creating a web service client application always starts with an existing WSDL file.

Typically, you retrieve the WSDL directly from a web service provider using the `wsimport` tool. The `wsimport` tool then generates the corresponding Java source code for the interface described by the WSDL. The Java compiler, `javac`, is then called to compile the source into class files. The programming code uses the generated classes to access the web service.

Creating a Client from WSDL

To create a client from WSDL, you must create the following files:

- Client Java File (fromwsdl)
- Client Configuration File (fromwsdl)
- `build.xml`
- `build.properties`

Client Java File (fromwsdl)

The client Java file defines the functionality of the web service client. The following code shows the `AddNumbersClient.java` file that is provided in the sample.

```
package fromjava.client;

import com.sun.xml.ws.Closeable;
import java.rmi.RemoteException;

public class AddNumbersClient {
  public static void main (String[] args) {
    AddNumbersImpl port = null;
    try {
      port = new
AddNumbersImplService().getAddNumbersImplPort();
      int number1 = 10;
      int number2 = 20;
      System.out.printf ("Invoking addNumbers(%d, %d)\n",
          number1, number2);
      int result = port.addNumbers (number1, number2);
      System.out.printf (
          "The result of adding %d and %d is %d.\n\n",
          number1, number2, result);

      number1 = -10;
      System.out.printf ("Invoking addNumbers(%d, %d)\n",
          number1, number2);
      result = port.addNumbers (number1, number2);
      System.out.printf (
          "The result of adding %d and %d is %d.\n",
          number1, number2, result);
    } catch (AddNumbersException_Exception ex) {
      System.out.printf (
          "Caught AddNumbersException_Exception: %s\n",
          ex.getFaultInfo ().getDetail ());
    } finally {
      ((Closeable)port).close();
    }
  }
}
```

This file specifies two positive integers that are to be added by the web service, passes the integers to the web service and gets the results from the web service via the `port.addNumbers` method, and prints the results to the screen. It then specifies a negative number to be added, gets the results (which should be an exception), and prints the results (the exception) to the screen.

Client Configuration File (fromwsdl)

This is a sample `custom-client.xml` file. The `wsdlLocation`, `package name`, and `jaxb:package name` xml tags are unique to each client and are highlighted in bold text

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bindings
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  wsdlLocation="http://localhost:8080/wsit-enabled-fromwsdl/
      addnumbers?wsdl"
  xmlns="http://java.sun.com/xml/ns/jaxws">
  <bindings node="ns1:definitions"
      xmlns:ns1="http://schemas.xmlsoap.org/wsdl/">
    <package name="fromwsdl.client"/>
  </bindings>
  <bindings node="ns1:definitions/ns1:types/xsd:schema
      [@targetNamespace='http://duke.org']"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:ns1="http://schemas.xmlsoap.org/wsdl/">
    <jaxb:schemaBindings>
      <jaxb:package name="fromwsdl.client"/>
    </jaxb:schemaBindings>
  </bindings>
</bindings>
```

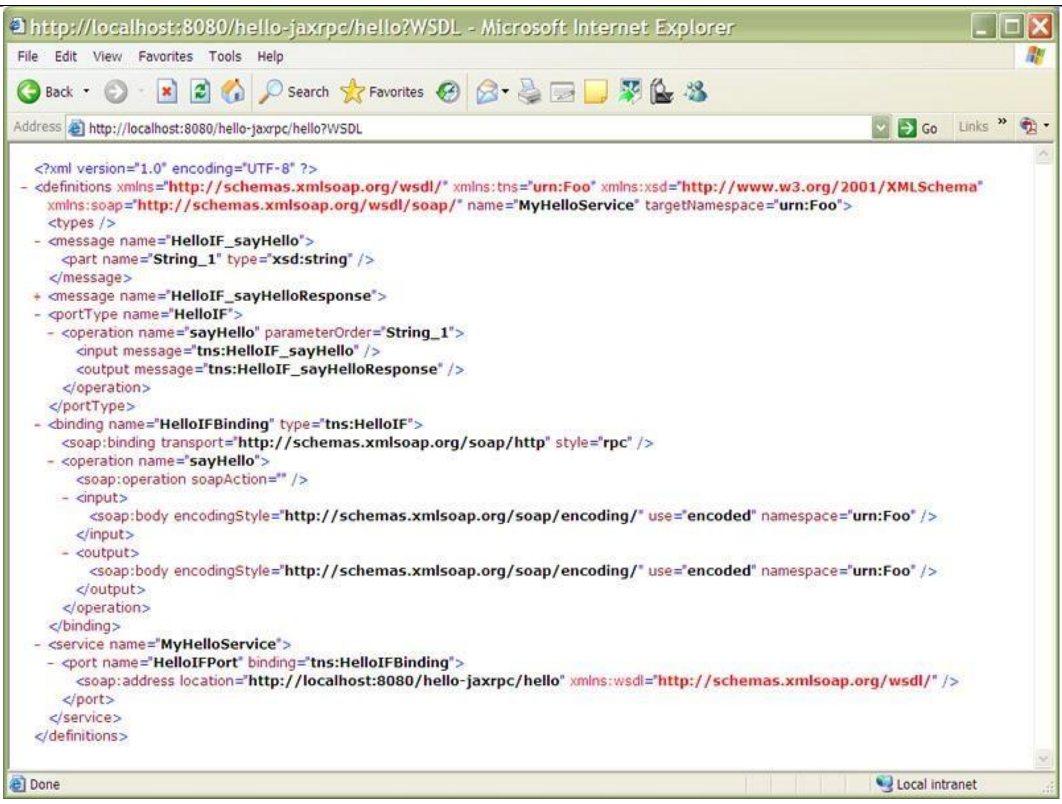
11.

(i) **List out** the installation steps of JWSDP.

JWSDP

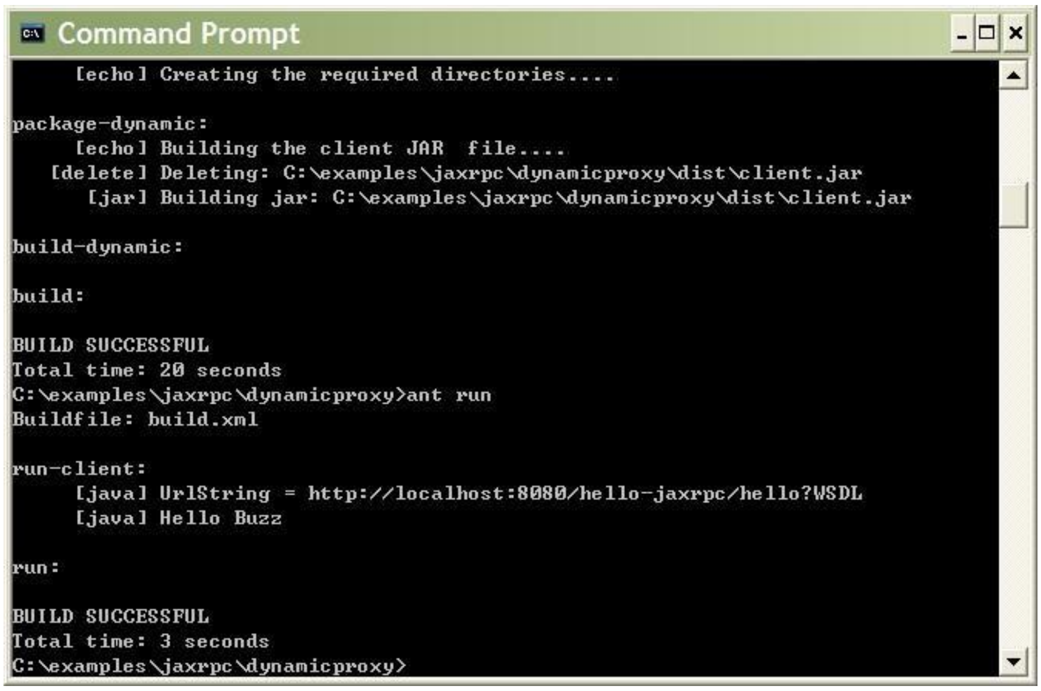
```
1. Install JDK 6.0 (i.e., JDK 1.6.0)

   Set up the following environment variables:
   JAVA_HOME    C:\Program Files\Java\jdk1.6.0_07

   Add the following path:
   C:\Program Files\Java\jdk1.6.0_07\bin
```

```
2. Install JWSDP 2.0 & Tomcat 5.0 for JWSDP (based upon Tomcat 5.0.19
that implements
   the Java Server Pages 2.0 and Java Servlet 2.4 specifications)

   Set up the following environment variables:
   JWSDP_HOME C:\Sun\jwsdp-2.0
   ANT_HOME    C:\Sun\jwsdp-2.0\apache-ant

   Add the following path:
   C:\Sun\jwsdp-2.0\jwsdp-shared\bin;C:\Sun\jwsdp-2.0\apache-ant\bin

3. Copy examples.zip into C:\ and extract here

4. Copy lib.zip into C:\Sun\jwsdp-2.0\server directory and extract
here
   Delete the file "lib.zip"

5. Replace saaj-impl.jar file at the following directories by saaj-
impl-1.3.jar.
   Rename it to saaj-impl.jar.

   C:\Sun\jwsdp-2.0\saaj\lib
   C:\Sun\tomcat50-jwsdp\saaj\lib

6. Modify C:\examples\common\build.properties for the first four
lines as follows:

   tutorial.home=C:
   tutorial.install=${tutorial.home}
   username=hxu
   password=12345

   where "hxu" and "12345" are the username and password for the
Tomcat server.

7. Build server:
   cd C:\examples\jaxrpc\helloservice
   ant build

   Start Tomcat from JWSDP 2.0

   Deploy server:
   ant deploy

   Note: If application already exists at path /hello-jaxrpc, you
should use the
   command "ant undeploy" to undeploy the web service first.

   Verify the deployment:
   To verify that the service has been successfully deployed, open a
browser window
   and specify the service endpoint's URL as follows:

   http://localhost:8080/hello-jaxrpc/hello?WSDL
   You should get the following display.
```


http://localhost:8080/hello-jaxrpc/hello?WSDL - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/hello-jaxrpc/hello?WSDL
Go Links
Done
Local intranet

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="urn:Foo" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="MyHelloService" targetNamespace="urn:Foo">
    <types />
  - <message name="HelloIF_sayHello">
      <part name="String_1" type="xsd:string" />
    </message>
  + <message name="HelloIF_sayHelloResponse">
  - <portType name="HelloIF">
    - <operation name="sayHello" parameterOrder="String_1">
        <input message="tns:HelloIF_sayHello" />
        <output message="tns:HelloIF_sayHelloResponse" />
      </operation>
    </portType>
  - <binding name="HelloIFBinding" type="tns:HelloIF">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc" />
    - <operation name="sayHello">
        <soap:operation soapAction="" />
      - <input>
          <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" namespace="urn:Foo" />
        </input>
      - <output>
          <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" namespace="urn:Foo" />
        </output>
      </operation>
    </binding>
  - <service name="MyHelloService">
    - <port name="HelloIFPort" binding="tns:HelloIFBinding">
        <soap:address location="http://localhost:8080/hello-jaxrpc/hello" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
      </port>
    </service>
  </definitions>
```

```
8. Build client:
   cd C:\examples\jaxrpc\dynamicproxy
   ant build

   Run client:
   ant run
```



```
     [echo] Creating the required directories....

package-dynamic:
     [echo] Building the client JAR  file....
   [delete] Deleting: C:\examples\jaxrpc\dynamicproxy\dist\client.jar
      [jar] Building jar: C:\examples\jaxrpc\dynamicproxy\dist\client.jar

build-dynamic:

build:

BUILD SUCCESSFUL
Total time: 20 seconds
C:\examples\jaxrpc\dynamicproxy>ant run
Buildfile: build.xml

run-client:
     [java] UrlString = http://localhost:8080/hello-jaxrpc/hello?WSDL
     [java] Hello Buzz

run:

BUILD SUCCESSFUL
Total time: 3 seconds
C:\examples\jaxrpc\dynamicproxy>
```

	(ii) **Describ**e on Simple Object Access Protocol. **XML Soap** • SOAP stands for **S**imple **O**bject **A**ccess **P**rotocol • SOAP is an application communication protocol • SOAP is a format for sending and receiving messages • SOAP is platform independent • SOAP is based on XML • SOAP is a W3C recommendation Why SOAP? It is important for web applications to be able to communicate over the Internet. The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this. SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages. SOAP Building Blocks A SOAP message is an ordinary XML document containing the following elements: • An Envelope element that identifies the XML document as a SOAP message • A Header element that contains header information • A Body element that contains call and response information • A Fault element containing errors and status information Syntax Rules Here are some important syntax rules: • A SOAP message MUST be encoded using XML • A SOAP message MUST use the SOAP Envelope namespace • A SOAP message must NOT contain a DTD reference • A SOAP message must NOT contain XML Processing Instructions
12.	(i) **Discuss** the XMLHttpRequest Object with example.

AJAX - The XMLHttpRequest Object

The keystone of AJAX is the XMLHttpRequest object.

The XMLHttpRequest Object

All modern browsers support the XMLHttpRequest object.

The XMLHttpRequest object can be used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Create an XMLHttpRequest Object

All modern browsers (Chrome, Firefox, IE7+, Edge, Safari Opera) have a built-in XMLHttpRequest object.

Syntax for creating an XMLHttpRequest object:

```
variable = new XMLHttpRequest();
```

Example

```
var xhttp = new XMLHttpRequest();
```

Access Across Domains

For security reasons, modern browsers do not allow access across domains.

This means that both the web page and the XML file it tries to load, must be located on the same server.

The examples on W3Schools all open XML files located on the W3Schools domain.

If you want to use the example above on one of your own web pages, the XML files you load must be located on your own server.

Example

```
if (window.XMLHttpRequest) {
    // code for modern browsers
    xmlhttp = new XMLHttpRequest();
 } else {
    // code for old IE browsers
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```

(ii) **Describe** about Java web service basics.

JAX-WS Example RPC Style

Creating JAX-WS example is a easy task because it requires no extra configuration settings.

JAX-WS API is inbuilt in JDK, so you don't need to load any extra jar file for it. Let's see a simple example of JAX-WS example in RPC style.

There are created 4 files for hello world JAX-WS example:

1. HelloWorld.java
2. HelloWorldImpl.java
3. Publisher.java
4. HelloWorldClient.java

The first 3 files are created for server side and 1 application for client side.

JAX-WS Server Code

File: HelloWorld.java

```
package com.javatpoint;
import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;
//Service Endpoint Interface
@WebService
@SOAPBinding(style = Style.RPC)
public interface HelloWorld{
.    @WebMethod String getHelloWorldAsString(String name);
.}
```

File: HelloWorldImpl.java

```
package com.javatpoint;
import javax.jws.WebService;
//Service Implementation
@WebService(endpointInterface = "com.javatpoint.HelloWorld")
```

```
public class HelloWorldImpl implements HelloWorld{
    @Override
    public String getHelloWorldAsString(String name) {
        return "Hello World JAX-WS " + name;
    }
. }
```

File: Publisher.java

```
package com.javatpoint;
import javax.xml.ws.Endpoint;
//Endpoint publisher
public class HelloWorldPublisher{
    public static void main(String[] args) {
       Endpoint.publish("http://localhost:7779/ws/hello", new HelloWorldIm
pl());
        }
}
```

How to view generated WSDL

After running the publisher code, you can see the generated WSDL file by visiting the URL:

```
http://localhost:7779/ws/hello?wsdl
```

JAX-WS Client Code

File: HelloWorldClient.java

```
package com.javatpoint;
import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;
public class HelloWorldClient{
    public static void main(String[] args) throws Exception {
    URL url = new URL("http://localhost:7779/ws/hello?wsdl");

        //1st argument service URI, refer to wsdl document above
.   //2nd argument is service name, refer to wsdl document above
.       QName qname = new QName("http://javatpoint.com/", "HelloWorld
ImplService");
.       Service service = Service.create(url, qname);
.       HelloWorld hello = service.getPort(HelloWorld.class);
```

	. System.out.println(hello.getHelloWorldAsString("javatpoint rpc"));
	 . }
	 . }
	
	 Output:
	
	 `Hello World JAX-WS javatpoint rpc`

13.	(i) **Explain** in detail about SOAP encoding. • For transfer between client and server in a SOAP message, we encode them in XML. SOAP Encoding is an extension of the SOAP framework specification that defines how a data value should be encoded in an XML format. SOAP Data Model is defined as an adjunct in SOAP 1.2 specification. SOAP encoding offers the following rules to convert any data value defined in SOAP data model into XML format. Converting a data value into XML format is called serialization or encoding. Rule 1. A simple value node with a labeled inbound edge will be serialized into a single XML element with the edge's label as the element's name and node value as the element's text content. Rule 2. When serializing a node into an XML element, an "xsi:type" attribute can be added to specify the value type of this note. For more information on "xsi:type", see the other sections in this book. Rule 3. A compound value node with labeled outbound edges, a data structure, will be serialized into a single XML element with child elements. One outbound edge will be serialized into one child element with element's name equal to the edge's label. The order of child elements is not significant. Rule 4. A compound value node with non-labeled outbound edges, a data array, will be serialized into a single XML element with child elements. One outbound edge will be serialized into one child element with element's name equal to any label as long as it's the same for all child elements. The order of child elements signifies the position values of outbound edges. Rule 5. When serializing an array, an "enc:itemType" attribute can be added to specify the value type of its sub nodes, and an "enc:arraySize" attribute can be added to specify the number of values in the array. (ii) **Point out** the RPC representation model. **What Is JAX-RPC?** JAX-RPC stands for Java API for XML-based RPC. It's an API for building Web services and clients that used remote procedure calls (RPC) and XML. Often used in a distributed client/server model, an RPC mechanism enables clients to execute procedures on other systems.

In JAX-RPC, a remote procedure call is represented by an XML-based protocol such as SOAP. The SOAP specification defines envelope structure, encoding rules, and a convention for representing remote procedure calls and responses. These calls and responses are transmitted as SOAP messages over HTTP. In this release, JAX-RPC relies on SOAP 1.1 and HTTP 1.1.

Although JAX-RPC relies on complex protocols, the API hides this complexity from the application developer. On the server side, the developer specifies the remote procedures by defining methods in an interface written in the Java programming language. The developer also codes one or more classes that implement those methods. Client programs are also easy to code. A client creates a proxy, a local object representing the service, and then simply invokes methods on the proxy.

With JAX-RPC, clients and Web services have a big advantage--the platform independence of the Java programming language. In addition, JAX-RPC is not restrictive: a JAX-RPC client can access a Web service that is not running on the Java platform and vice versa. This flexibility is possible because JAX-RPC uses technologies defined by the World Wide Web Consortium (W3C): HTTP, SOAP, and the Web Service Description Language (WSDL). WSDL specifies an XML format for describing a service as a set of endpoints operating on messages.

14.

Explain the structure of a WSDL document, its elements and their purposes with appropriate examples.

A WSDL document defines **services** as collections of network endpoints, or **ports**. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: **messages**, which are abstract descriptions of the data being exchanged, and **port types** which are abstract collections of **operations**. The concrete protocol and data format specifications for a particular port type constitutes a reusable **binding**. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service. Hence, a WSDL document uses the following elements in the definition of network services:

- **Types**– a container for data type definitions using some type system (such as XSD).
- **Message**– an abstract, typed definition of the data being communicated.

- **Operation**– an abstract description of an action supported by the service.
- **Port Type**–an abstract set of operations supported by one or more endpoints.
- **Binding**– a concrete protocol and data format specification for a particular port type.
- **Port**– a single endpoint defined as a combination of a binding and a network address.
- **Service**– a collection of related endpoints.

In addition, WSDL defines a common **binding** mechanism. This is used to attach a specific protocol or data format or structure to an abstract message, operation, or endpoint. It allows the reuse of abstract definitions.

WSDL Document Example

The following example shows the WSDL definition of a simple service providing stock quotes. The service supports a single operation called GetLastTradePrice, which is deployed using the SOAP 1.1 protocol over HTTP. The request takes a ticker symbol of type string, and returns the price as a float. A detailed description of the elements used in this definition can be found in Section 2 (core language) and Section 3 (SOAP binding).

This example uses a fixed XML format instead of the SOAP encoding (for an example using the SOAP encoding, see Example 4).

Example 1 SOAP 1.1 Request/Response via HTTP

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://example.com/stockquote.wsdl"
          xmlns:tns="http://example.com/stockquote.wsdl"
          xmlns:xsd1="http://example.com/stockquote.xsd"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns="http://schemas.xmlsoap.org/wsdl/">

    <types>
       <schema targetNamespace="http://example.com/stockquote.xsd"
              xmlns="http://www.w3.org/2000/10/XMLSchema">
          <element name="TradePriceRequest">
             <complexType>
                 <all>
                     <element name="tickerSymbol" type="string"/>
                 </all>
             </complexType>
          </element>
          <element name="TradePrice">
             <complexType>
                 <all>
                     <element name="price" type="float"/>
                 </all>
               </complexType>
          </element>
       </schema>
    </types>

    <message name="GetLastTradePriceInput">
       <part name="body" element="xsd1:TradePriceRequest"/>
    </message>

    <message name="GetLastTradePriceOutput">
       <part name="body" element="xsd1:TradePrice"/>
    </message>

    <portType name="StockQuotePortType">
       <operation name="GetLastTradePrice">
          <input message="tns:GetLastTradePriceInput"/>
          <output message="tns:GetLastTradePriceOutput"/>
       </operation>
    </portType>

    <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
       <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
       <operation name="GetLastTradePrice">
          <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
          <input>
             <soap:body use="literal"/>
          </input>
          <output>
             <soap:body use="literal"/>
          </output>
       </operation>
    </binding>

    <service name="StockQuoteService">
       <documentation>My first service</documentation>
       <port name="StockQuotePort" binding="tns:StockQuoteBinding">
          <soap:address location="http://example.com/stockquote"/>
       </port>
    </service>

</definitions>
```

PART – C	
Q.No	**Questions**
1.	**Create** an XML HttpRequest to retrieve data from an XML file and display the data in an HTML table. The data to be retrieved is a collection of stationary items stored in an XML file. <u>The XML Document Used</u> INPUT: XML file called "cd_catalog.xml".

Display XML Data in an HTML Table

This example loops through each <CD> element, and displays the values of the <ARTIST> and the <TITLE> elements in an HTML table:

Example

```
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
  border-collapse:collapse;
}
th, td {
  padding: 5px;
}
</style>
</head>
<body>

<table id="demo"></table>

<script>
function loadXMLDoc() {
  var xmlhttp = new XMLHttpRequest();
  xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      myFunction(this);
    }
  };
  xmlhttp.open("GET", "cd_catalog.xml", true);
  xmlhttp.send();
}
function myFunction(xml) {
  var i;
  var xmlDoc = xml.responseXML;
  var table="<tr><th>Artist</th><th>Title</th></tr>";
  var x = xmlDoc.getElementsByTagName("CD");
  for (i = 0; i <x.length; i++) {
    table += "<tr><td>" +
    x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValu
e +
    "</td><td>" +
    x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue
 +
    "</td></tr>";
  }
  document.getElementById("demo").innerHTML = table;
}
</script>

</body>
</html>
```

OUTPUT

Artist	Title
Bob Dylan	Empire Burlesque
Bonnie Tyler	Hide your heart
Dolly Parton	Greatest Hits
Gary Moore	Still got the blues
Eros Ramazzotti	Eros
Bee Gees	One night only
Dr.Hook	Sylvias Mother

2. **Summarize** Ajax Client server architecture in detail.

What is AJAX?

AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.

AJAX is not a programming language.

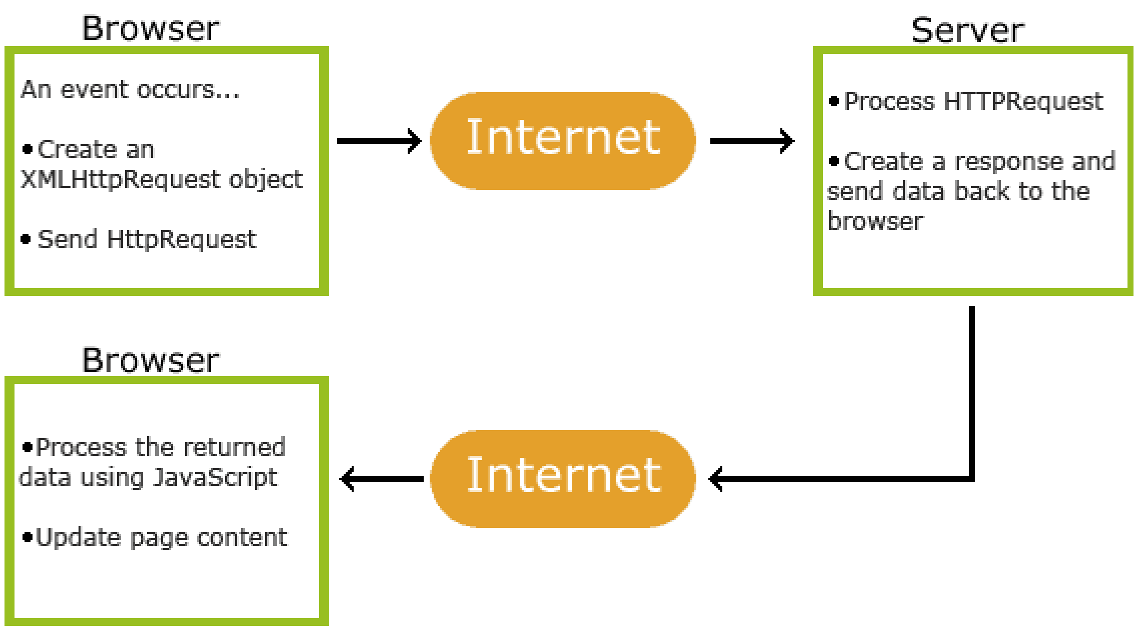
AJAX just uses a combination of:

- A browser built-in XMLHttpRequest object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)

AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

How AJAX Works

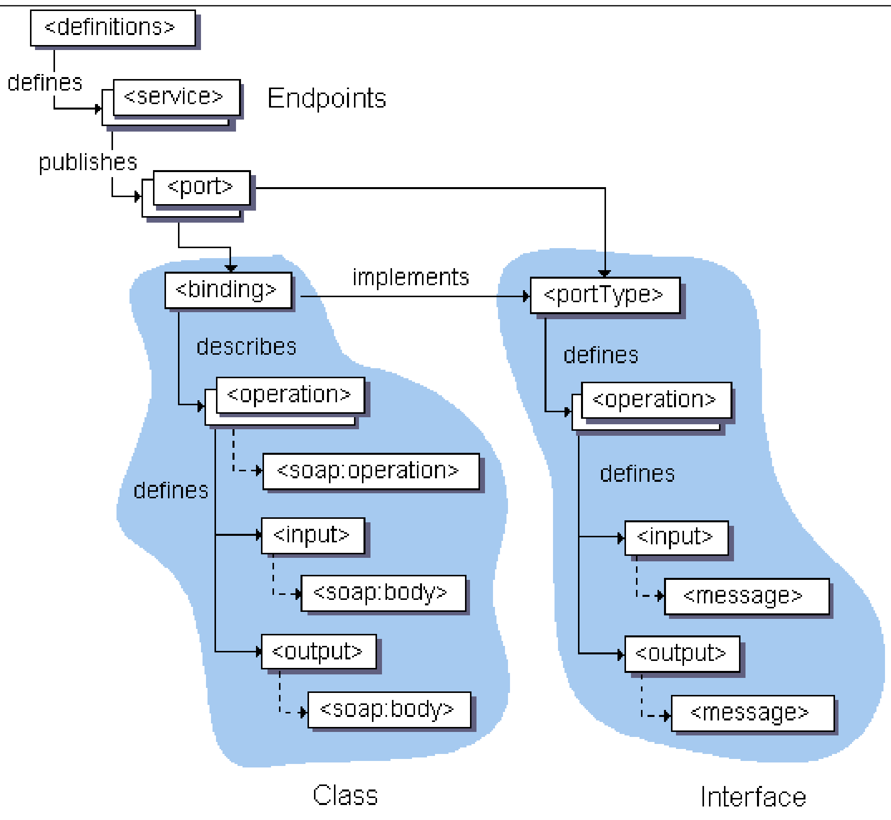

Browser
An event occurs...
•Create an XMLHttpRequest object
•Send HttpRequest
Internet
Server
•Process HTTPRequest
•Create a response and send data back to the browser
Browser
•Process the returned data using JavaScript
•Update page content
Internet

1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

3. **Give** the basic structure of a WSDL and show how they are used to create, publish, test and describe web services.

Structure of a WSDL Document

Web Services Description Language (WSDL) is an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages. The diagram below illustrates the elements that are present in a WSDL document, and indicates their relationships. To see an example of how this is implemented in a WSDL document, see Example of a WSDL Document .



WSDL Document Elements

A WSDL document has a definitions element that contains the other five elements, types, message, portType, binding and service. The following sections describe the features of the generated client code.

WSDL supports the XML Schemas specification (XSD) as its type system.

definitions

Contains the definition of one or more services. JDeveloper generates the following attribute declarations for this section:

- `name` is optional.
- `targetNamespace` is the logical namespace for information about this service. WSDL documents can import other WSDL documents, and setting targetNamespace to a unique value ensures that the namespaces do not clash.
- `xmlns` is the default namespace of the WSDL document, and it is set to `http://schemas.xmlsoap.org/wsdl/`.
- All the WSDL elements, such as `<definitions>`, `<types>` and `<message>` reside in this namespace.

	• `xmlns:xsd` and `xmlns:soap` are standard namespace definitions that are used for specifying SOAP-specific information as well as data types. • `xmlns:tns` stands for this namespace. • `xmlns:ns1` is set to the value of the `schema targetNamespace`, in the `<types>` section.
4.	**Compare** and **contrast** the additional web application architecture and AJAX Based web application architecture. **<u>Traditional Web Applications vs. Ajax Applications</u>** The following highlights the key differences between traditional web applications and Ajax-based web applications. ***Traditional Web Applications*** ➢ Figure 15.1 presents the typical interactions between the client and the server in a tradi-tional web application, such as one that uses a user registration form. ➢ First, the user fills in the form's fields, then submits the form (Fig. 15.1, *Step 1*). The browser generates a re-quest to the server, which receives the request and processes it (*Step 2*). ➢ The server generates and sends a response containing the exact page that the browser will render (*Step 3*), which causes the browser to load the new page (*Step 4*) and temporarily makes the browser win-dow blank. Note that the client *waits* for the server to respond and *reloads the entire page* with the data from the response (*Step 4*). ➢ While such a **synchronous request** is being processed on the server, the user cannot interact with the client web page.

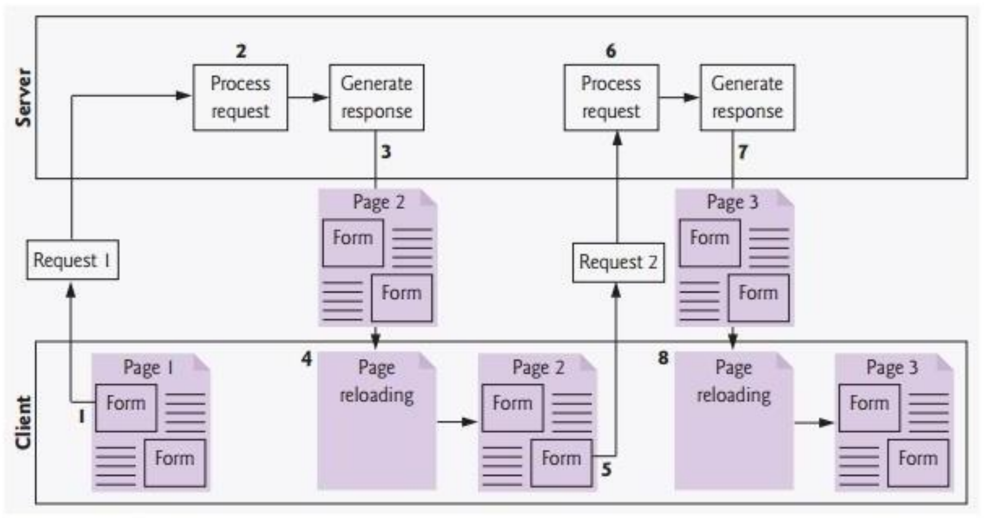


Fig. 15.1 | Classic web application reloading the page for every user interaction.

- ➢ Frequent long periods of waiting, due perhaps to Internet congestion, have led some users to refer to the World Wide Web as the "World Wide Wait."
- ➢ If the user interacts with and submits an-other form, the process begins again (*Steps 5–8*).

This model was originally designed for a web of hypertext documents—what some people call the "brochure web."

As the web evolved into a full-scale applications platform, the model shown in Fig. 15.1 yielded "choppy" application performance.

Every full-page refresh required users to re-establish their understanding of the full-page contents.

Users began to demand a model that would yield the responsive feel of desktop applications.

Ajax Web Applications

- ➢ Ajax applications add a layer between the client and the server to manage communication between the two (Fig. 15.2). When the user interacts with the page, the client creates an XMLHttpRequest object to manage a request (*Step 1*).
- ➢ The XMLHttpRequest object sends the request to the server (*Step 2*) and awaits the response.
- ➢ The requests are **asynchronous**, so the user can continue interacting with the application on the client-side while the server processes the

earlier request concurrently. Other user interactions could result in addition-al requests to the server (*Steps 3* and *4*).

- Once the server responds to the original request (*Step 5*), the XMLHttpRequest object that issued the request calls a client-side function to process the data returned by the server.
- This function—known as a **callback function**— uses **partial page updates** (*Step 6*) to display the data in the existing web page *without re-loading the entire page*. At the same time, the server may be responding to the second re-quest (*Step 7*) and the client-side may be starting to do another partial page update (*Step 8*).
- The callback function updates only a designated part of the page.
- Such partial page up-dates help make web applications more responsive, making them feel more like desktop applications.
- The web application does not load a new page while the user interacts with it.

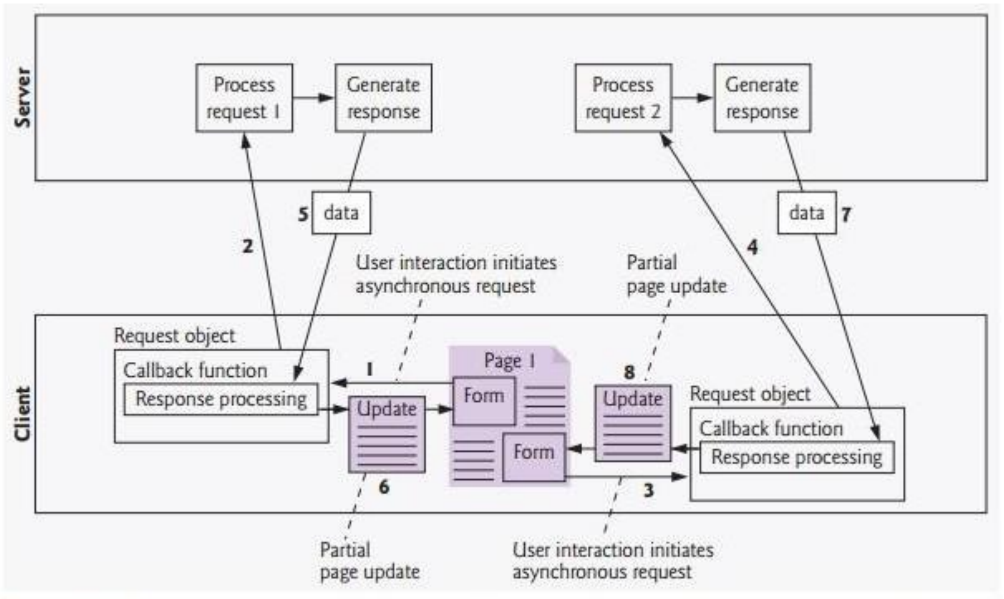


Fig. 15.2 | Ajax-enabled web application interacting with the server asynchronously.

MADHA
Expertise | Empathy | Excellence
ENGINEERING COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COMMON FOR: DEPARTMENT OF INFORMATION TECHNOLOGY

CS3351 – DIGITAL PRINCIPLES AND COMPUTER ORGANIZATION

YEAR / SEM : IV / VII

R – 2017
LECTURE NOTES

UNIT V -APPLICATION LAYER

WWW and HTTP – FTP – Email –Telnet –SSH – DNS – SNMP.

PART A

1. What is the function (Define) of SMTP? (May & Nov 2015)

The TCP/IP protocol supports electronic mail on the Internet is called Simple Mail Transfer (SMTP). It is a system for sending messages to other computer users based on e-mail addresses. SMTP provides mail exchange between users on the same or different computers.

2. What is the difference between a user agent (UA) and a mail transfer agent (MTA)?

The UA prepares the message, creates the envelope, and puts the message in the envelope. The MTA transfers the mail across the Internet.

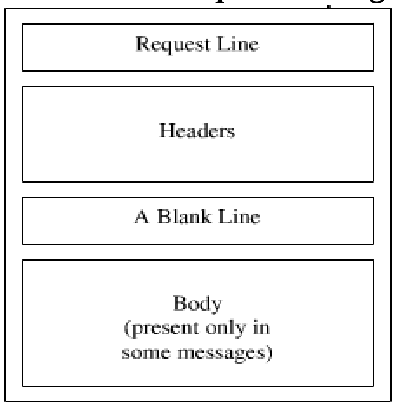
3. How doesMIME (Differ) enhance SMTP? (Nov/Dec 2007)(Or) State the difference between SMTPand MIME (NOV/DEC 2014)

MIME is a supplementary protocol that allows non-ASCII data to be sent through SMTP. MIME transforms non-ASCII data at the sender site to NVT ASCII data and deliverers it to the client SMTP to be sent through the Internet. The server SMTP at the receiving side receives the NVT ASCII data and delivers it to MIME to be transformed back to the original data.

4. Why is an application such as POP needed for electronic messaging?

Workstations interact with the SMTP host, which receives the mail on behalf of every host in the organization, to retrieve messages by using a client-server protocol such as Post Office Protocol, version 3(POP3). Although POP3 is used to download messages from the server, the SMTP client still needed on the desktop to forward messages from the workstation user to its SMTP mail server.

5. Give the format of HTTP request message?



6. What is the purpose of Domain Name System?

Domain Name System can map a name to an address and conversely an address to name.

7. Discuss the three main division of the domain name space.

Domain name space is divided into three different sections: generic domains, country domains & inverse domain.

Generic domain: Define registered hosts according to their generic behavior, uses generic suffixes.

Country domain: Uses two characters to identify a country as the last suffix.

Inverse domain:Finds the domain name given the IP address.

8. Define CGI?

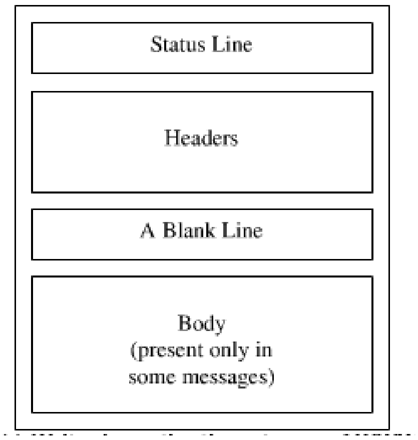
CGI is a standard for communication between HTTP servers and executable programs. It is used in crating dynamic documents.

9. What are the requests messages support SNMP and explain it?

- GET
- SET

The former is used to retrieve a piece of state from some node and the latter is used to store a new piece of state in some node.

10. Give the format of HTTP response message?



11. Why name services are sometimes called as middleware?

Name services are sometimes called middleware because they fill a gap between applications and the underlying network

12. What are the types of DNS Message

Two types of messages

- Query: header and question records
- Response: Header, question records, answer records, authoritative records, and additional records.

13. What is POP3? (Nov 2016)

POP3 (Post Office Protocol 3) is the most recent version of a standard protocol for receiving e-mail. POP3 is a client/serverprotocol in which e-mail is received and held for you by your Internet server. POP and IMAP deal with the receiving of e-mail and are not to be confused with the Simple Mail Transfer Protocol (SMTP), a protocol for transferring e-mail across the Internet.

14. What is IMAP4? (Nov 2016)

IMAP (Internet Message Access Protocol) is a standard protocol for accessing e-mail from your local server. IMAP (the latest version is IMAP Version 4) is a client/server protocol in which e-mail is received and held for you by your Internet server.IMAP can be thought of as a remote file server. POP3 can be thought of as a "store-and-forward" service.

15. What is DNS? (Apr /may 2010)

The **DNS** translates Internet domain and host names to IP addresses. DNS automatically converts the names we type in our Web browser address bar to the IP addresses of Web servers hosting those sites.

16. What is persistent HTTP?What are the advantages of allowing persistent TCP Connections in HTTP?(May 2013) (Nov 2016)

HTTP persistent connection, also called HTTP keep-alive, or HTTP connection reuse, is the idea of using a single TCP connection to send and receive multiple HTTP requests/responses, as opposed to opening a new connection for every single request/response pair.

Persistent HTTP connections have a number of advantages:

- By opening and closing fewer TCP connections, CPU time is saved in routers and hosts (clients, servers, proxies, gateways, tunnels, or caches), and memory used for TCP protocol control blocks can be saved in hosts.

17. Is a cryptographic hash function, an irreversible mapping? Justify your answer.

1. It is really, really hard to infer the input from the hash **because there are an infinite amount of input strings that will generate the same output** (irreversible property).
2. However, ***finding*** even a single instance of multiple input strings that generate the same output is also really, really hard (collision resistant property).

18. Define SNMP?

SNMP is a frame work for managing devices in an internet using TCP/IP suite. It provides fundamental operations for monitoring and maintaining an internet.

19. What DNS cache issues are involved in changing the IP address of a webServer host name? Nov/Dec 2013

The Domain Name System supports DNS cache servers which store DNS query results for a period of time determined in the configuration (time-to-live) of the domain name record in question. Typically, such caching DNS servers, also called DNS caches, also implement the recursive algorithm necessary to resolve a given name starting with the DNS root through to the authoritative name servers of the queried domain. With this function implemented in the name server, user applications gain efficiency in design and operation.

20. Differentiate application programs and application protocols. Nov/Dec 2013

An application program (sometimes shortened to application) is any program designed to perform a specific function directly for the user or, in some cases, for another application program. Examples of application programs include word processors; database programs; Web browsers; development tools; drawing, paint, and image editing programs; and communication programs. Application programs use the services of the computer'soperating system and other supporting programs.

Application protocols govern various processes, such as the process for downloading a web page, or for sending e-mail. The application protocol directs how these processes are done.

21. What are the two mainly used application protocols

- SimpleMail Transfer Protocol (SMTP) is used to exchangeelectronic mail.
- Hypertext Transport Protocol (HTTP) is used to communicatebetween web browsers and web servers.

22. Define HTTP protocol .

HTTP PROTOCOL

- Protocol for transfer of data between Web servers and Web clients (browsers).
- “The Hypertext Transfer Protocol (HTTP) is an **application-level protocol** for **distributed**, collaborative, hypermedia information systems.
- Popular Web servers:
    - Apache HTTPD, JBoss and Tomcat
- Popular Web clients:
    - Firefox and Opera

23. Define web services.

The term *Web services* describes a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDIopenstandards over an Internet protocolbackbone. SOAP is used to transfer the data, WSDL is used for describing the services available and UDDI is used for listing what services are available. Used for businesses to communicate with each other and with clients, Web services allow organizations to communicate data without intimate knowledge of each other's IT systems behind the firewall.

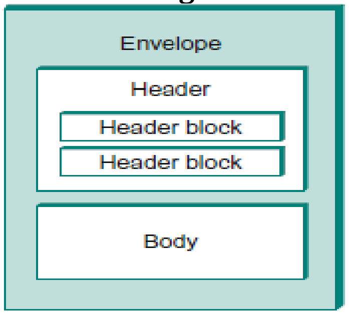
24. What is SOAP?

- SOAP stands for Simple Object Access Protocol
- SOAP is a communication protocol
- SOAP is for communication between applications
- SOAP is a format for sending messages
- SOAP communicates via Internet
- SOAP is platform independent
- SOAP is language independent
- SOAP is based on XML
- SOAP is simple and extensible
- SOAP allows you to get around firewalls

25. What is WSDL?Nov-Dec 2017

The Web Services Description Language (WSDL) is an XML-based language used to describe the services a business offers and to provide a way for individuals and other businesses to access those services electronically.

26. Draw SOAP message structure



27. Define MIME

MIME, an acronym for **Multipurpose Internet Mail Extensions**, specifies how messages must be formatted so that they can be exchanged between different email systems. MIME is a very flexible format, permitting one to include virtually any type of file or document in an email message. MIME messages can contain text, images, audio, video, or other application-specific data.

28. List down the key lengths supported by PGP (NOV/DEC 2014)

The "length" is a formal characterization of one of the mathematical values that constitute the *key pair.* Thus, the public and the private key don't have independent lengths per se; the private/public key pair has a length, which, by extension, is also said to be the length of the public key *and* of the private key.
The length is not the actual bit length of the encoding of either the public or private key, although there are correlations

29. What are the groups of HTTP header? (May 2015)

- Accept
- HTTP_User-Agent
- Content-Language
- Content-Length
- Content-Type
- Date
- Expires:
- Host
- Location
- Retry-After

30 . Define URL (May 2016)

A URL (**Uniform Resource Locator**), as the name suggests, provides a way to locate a resource on the web, the hypertext system that operates over the internet. The URL contains the name of the protocol to be used to access the resource and a resource name. The first part of a URL identifies what protocol to use. The second part identifies the IP address or domain name where the resource is located.

31. Mention the different levels in domain name space (May 2016)

- Top Level Domains
- Second Level Domails
- Third Level Domails

32. Mention the types of HTTP messages

- HTTP request message
- HTTP response message

33. State the usage of conditional get in HTTP (Apr/May 2017)

A conditional GET is an HTTP GET request that may return an HTTP 304 response (instead of HTTP 200). An HTTP 304 response indicates that the resource has not been modified since the previous GET, and so the resource is not returned to the client in such a response.

34. Present the information contained in a DNS resource record? (Apr/May 2017)

A resource record is a name-to-value binding, a 5-tuple that contains the following fields:

- **Domain name**: the domain to which this record applies.

- **Class**: set to IN for internet information. For other information other codes may be specified.
- **Type**: tells what kind of record it is.
- **Time to live**: Upper Limit on the time to reach the destination
- **Value**: can be an IP address, a string or a number depending on the record type.

35. Write the use of HTTP (Apr-May 2017)

HTTP (Hypertext Transfer Protocol) is the set of rules for transferring files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web. As soon as a Web user opens their Web browser, the user is indirectly making use of HTTP. HTTP is an application protocol that runs on top of the TCP/IP suite of protocols (the foundation protocols for the Internet).

36. Name four factors needed for a secure network?

Privacy: The sender and the receiver expect confidentiality.

Authentication: The receiver is sure of the sender's identity and that an imposter has not sent the message.

Integrity: The data must arrive at the receiver exactly as it was sent.

Non-Reputation: The receiver must able to prove that a received message came from a specific sender

37. Define SSH?

Secure Shell is used to provide a remote login, and used to remotely execute commands and transfer files and also provide strong client/server authentication / message integrity.

38. What is TELNET PROTOCOL?

A TELNET connection is a Transmission Control Protocol (TCP) connection used to transmit data with interspersed TELNET control information.

The TELNET Protocol is built upon three main ideas: first, the concept of a "Network Virtual Terminal"; second, the principle of negotiated options; and third, a symmetric view of terminals and processes.

39. What is PGP?

Pretty Good Privacy. A program using public key encryption popularly used with email
A high security RSA public-key encryption application for MS-DOS, Unix, VAX/VMS, and other computers. It was written by Philip R. Zimmermann of Phil's Pretty Good(tm) Software and later augmented by a cast of thousands, especially including Hal Finney, Branko Lankester, and Peter Gutmann.

40. What is SSH?

(**S**ecure **Sh**ell) A security protocol for logging into a remote server. SSH provides an encrypted session for transferring files and executing server programs. Also serving as a secure client/server connection for applications such as database access and e-mail SSH supports a variety of authentication methods.

41. What are the applications of TELNET?

TELNET enables the establishment of a connection to a remote system in such a way that the local terminal appears to be a terminal at the remote system. TELNET general purpose client server application program.
Using the Telnet protocol user on a local host can remote-login and execute commands on another distant host **(Time sharing environment, Network virtual terminal)**

42. Define WWW

World Wide Web: The Web today is a repository of information in which the documents, called ***web pages,*** are distributed all over the world and related documents are linked together

The purpose of the Web has gone beyond the simple retrieving of linked documents.

The WWW today is a distributed client-server service, in which a client using a browser can access a service using a server. However, the service provided is distributed over many locations called *sites.* Each site holds one or more web pages.

PART-B

1. Explain WWW in detail (World Wide Web)

The idea of the Web was first proposed by Tim Berners-Lee in 1989 at *CERN*. The Web today is a repository of information in which the documents, called ***web pages,*** are distributed all over the world and related documents are linked together. The popularity and growth of the Web can be related to two terms in the above statement: *distributed* and *linked*

Distribution allows the growth of the Web. Each web server in the world can add a new web page to the repository and announce it to all Internet users without overloading a few servers. Linking allows one web page to refer to another web page stored in another server somewhere else in the world. The linking of web pages was achieved using a concept called ***hypertext***

Architecture

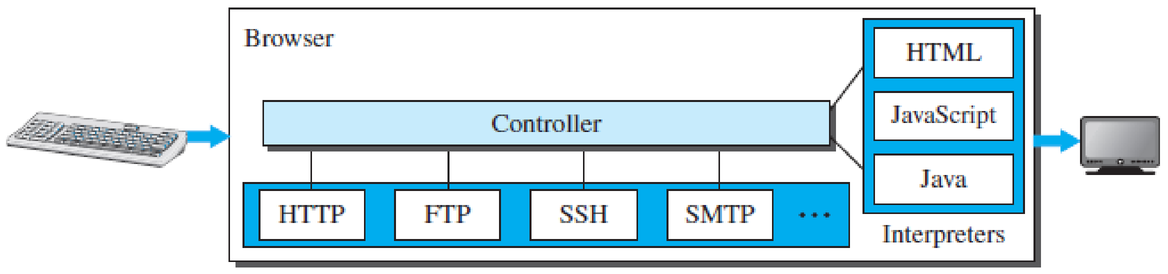
The WWW today is a distributed client-server service, in which a client using a browser can access a service using a server. However, the service provided is distributed over many locations called ***sites***.

Each site holds one or more web pages. Each web page, however, can contain some links to other web pages in the same or other sites. In other words, a web page can be simple or composite. A simple web page has no links to other web pages; a composite web page has one or more links to other web pages. Each web page is a file with a name and address.

Web Client (Browser)

A variety of vendors offer commercial **browsers** that interpret and display a web page, and all of them use nearly the same architecture. Each browser usually consists of three parts: a controller, client protocols, and interpreters.

Figure 26.2 *Browser*



The controller receives input from the keyboard or the mouse and uses the client programs to access the document. After the document has been accessed, the controller uses one of the interpreters to display the document on the screen. The client protocol can be one of the protocols described later, such as HTTP or

FTP. The interpreter can be HTML, Java, or JavaScript, depending on the type of document. Some commercial browsers include Internet Explorer, Netscape Navigator, and Firefox

Web Server

The web page is stored at the server. Each time a request arrives, the corresponding document is sent to the client. To improve efficiency, servers normally store requested files in a cache in memory; memory is faster to access than a disk. A server can also become more efficient through multithreading or multiprocessing. In this case, a server can answer more than one request at a time. Some popular web servers include Apache and Microsoft Internet Information Server

<u>Uniform Resource Locator (URL)</u>

A web page, as a file, needs to have a unique identifier to distinguish it from other web pages. To define a web page, we need three identifiers: *host*, *port*, and *path*. However, before defining the web page, we need to tell the browser what client server application we want to use, which is called the *protocol*. This means we need four identifiers to define the web page. The first is the type of vehicle to be used to fetch the web page; the last three make up the combination that defines the destination object (web page).

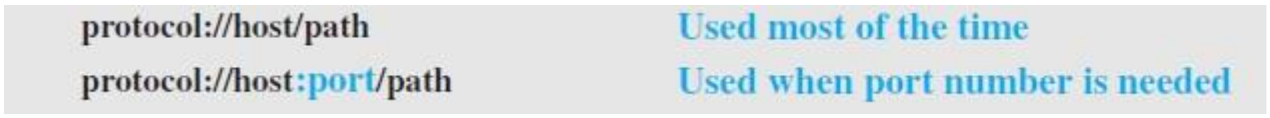
❑ ***Protocol.*** The first identifier is the abbreviation for the client-server program that we need in order to access the web page. Although most of the time the protocol is HTTP (HyperText Transfer Protocol), we can also use other protocols such as FTP (File Transfer Protocol).

❑ ***Host.*** The host identifier can be the IP address of the server or the unique name given to the server. IP addresses can be defined in dotted decimal notation, (such as 64.23.56.17); the name is normally the domain name that uniquely defines the host, such as *forouzan.com*

❑ ***Port.*** The port, a 16-bit integer, is normally predefined for the client-server application. For example, if the HTTP protocol is used for accessing the web page, the well-known port number is 80. However, if a different port is used, the number can be explicitly given.

❑ ***Path.*** The path identifies the location and the name of the file in the underlying operating system. The format of this identifier normally depends on the operating system. In UNIX, a path is a set of directory names followed by the file name, all separated by a slash. For example, */top/next/last/myfile* is a path that uniquely defines a file named *myfile,* stored in the directory *last*, which itself is part of the directory *next*, which itself is under the directory *top*.

To combine these four pieces together, the **uniform resource locator (URL)** has been designed; it uses three different separators between the four pieces as shown below:



protocol://host/path	Used most of the time
protocol://host:port/path	Used when port number is needed

<u>Web Documents</u>

The documents in the WWW can be grouped into three broad categories: static, dynamic, and active.

Static Documents

Static documents are fixed-content documents that are created and stored in a server. The client can get a copy of the document only. Languages used: HTML, XML, XSL, XHTML

Dynamic Documents

A **dynamic document** is created by a web server whenever a browser requests the document. When a request arrives, the web server runs an application program or a script that creates the dynamic document. The server returns the result of the program or script as a response to the browser that requested the document. Languages used: JSP, ASP

\

Active Documents

For many applications, we need a program or a script to be run at the client site. These are called ***active documents.*** Use java applets

2. Explain working of E-mail, describe how SMTP is used E-mail application in detail? (Apr/may 2011& 2010, Nov/Dec 2013) (Nov 2015)

E-mail (electronic mail) is the exchange of computer-stored messages by telecommunication.

- The model that works best for email is the Client-Server model.
- Clients carry out user interactions with the email server.
- Forms in which clients appear:
  - Application based - these are installed onto user's machines and include Microsoft Outlook and the freely available Outlook Express and Eudora.
  - Web based - these appear in a web browser's window and include Hotmail, Yahoo and Outlook web client.
- Basic functions include: (Services)
  - Ability to create new emails.
  - Display and store received emails.
  - Hold address lists of contacts, a calendar, journal and other extra functions that help organize the user's working day.
  - The client is also configured with the account information and names or IP addresses of the email servers with which it will be communicating.
- An email server is typically a combination of processes running on a server with a large storage capacity it includes a list of users and rules, and the capability to receive, send and store emails and attachments.
- Should process emails for months as sending, receiving and maintenance tasks are carried out at scheduled times. The client only has to connect to the email server when it sends and checks/receives new email.
- Sometimes it may be permanently connected to the server to allow access to shared address books or calendar information – this is typical of a LAN-based email server.
- Most email servers conduct email services by running two separate processes on the same machine.

- One process is the POP3 (Post Office protocol 3) server, which holds emails in a queue and delivers emails to the client when they are requested.
- The other is the SMTP (simple mail transfer protocol) server that receives outgoing emails from clients and sends and receives email from other SMTP servers.
- These two processes are linked by an internal mail delivery mechanism that moves mail between the POP3 and SMTP servers.
- When the client calls the email server to send or check for mail it connects to the server on certain TCP/IP ports:
    - SMTP on port 25
    - POP3 on port 110.

MAIL PROTOCOLS

- **SMTP -** Simple Mail Transport Protocol is used on the internet, it is not a transport layer protocol but is an application layer protocol.
- **POP3** - Post Office Protocol version 3 is used by clients to access an internet mail server to get mail. It is not a transport layer protocol.
- **IMAP4** - Internet Mail Access Protocol version 4 is the replacement for POP3.
- **MIME** - Multipurpose Internet Mail Extension is the protocol that defines the way files are attached to SMTP messages.

3. Explain the salient features of the SMTP protocol (Email protocols) (12)(MAY/JUNE 2009) SMTP protocol (May 2015& 2016, Apr/May 2017)

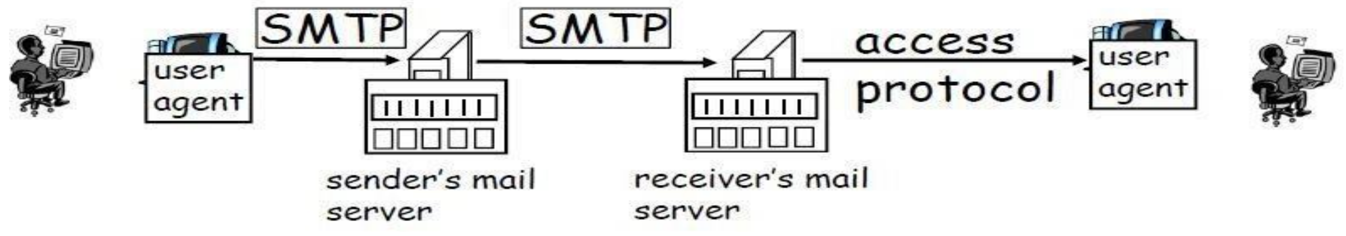
- Simple Mail Transfer Protocol (SMTP) is used to send mail across the internet. This protocol transfers electronic mail (e-mail) from the mail server of a source to the mail servers of destinations.
- The mail is enclosed in what is called an **envelope.** The envelope contains the TO and FROM fields and these are followed by the mail. The mail consists of two parts namely the Header and the Data. The Header has the TO and FROM fields.
- In SMTP data portion can contain only printable ASCII characters. The old method of sending a binary file was to send it in uuencoded form but there was no way to distinguish between the many types of binary files possible eg. .tar, .gz , .dvi etc.

There are four types of programs used in the process of sending and receiving mail. They are:

- **MUA** - Mail users agent. This is the program a user will use to type e-mail. It usually incorporates an editor for support. The user types the mail and it is passed to the sending MTA.

- **MTA** - Message transfer agent is used to pass mail from the sending machine to the receiving machine. There is a MTA program running on both the sending and receiving machine.

- **LDA** - Local delivery agent on the receiving machine receives the mail from its MTA.
- **Mail notifier** - This program notifies the recipient that they have mail.



Example

1) Ali uses UA to compose message and send it "to" ahmed@ksu.edu.sa

2) Ali's UA sends the message to his mail server; message placed in message queue

3) Client side of SMTP opens TCP connection with Ahmed's mail server

4) SMTP client sends Ali's message over the TCP connection

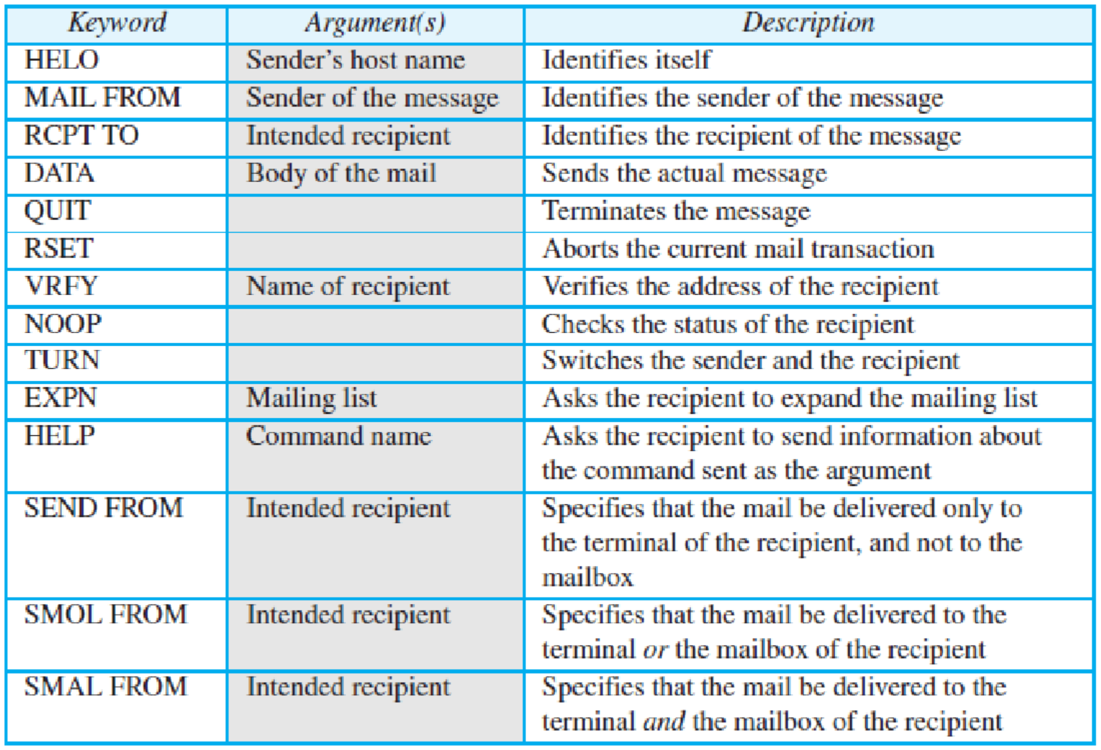
5) Ahmed's mail server receives and places the message in Ahmed's mailbox

6) Ahmed invokes his user agent to read message

SMTP Commands & Responses

SMTP uses commands and responses to transfer messages between an MTA client and an MTA server. The command is from an MTA client to an MTA server; the response is from an MTA server to the MTA client. Each command or reply is terminated by a two character (carriage return and line feed) end-of-line token.
It consists of a keyword followed by zero or more arguments. SMTP defines 14 commands

Table 26.6 *SMTP commands*

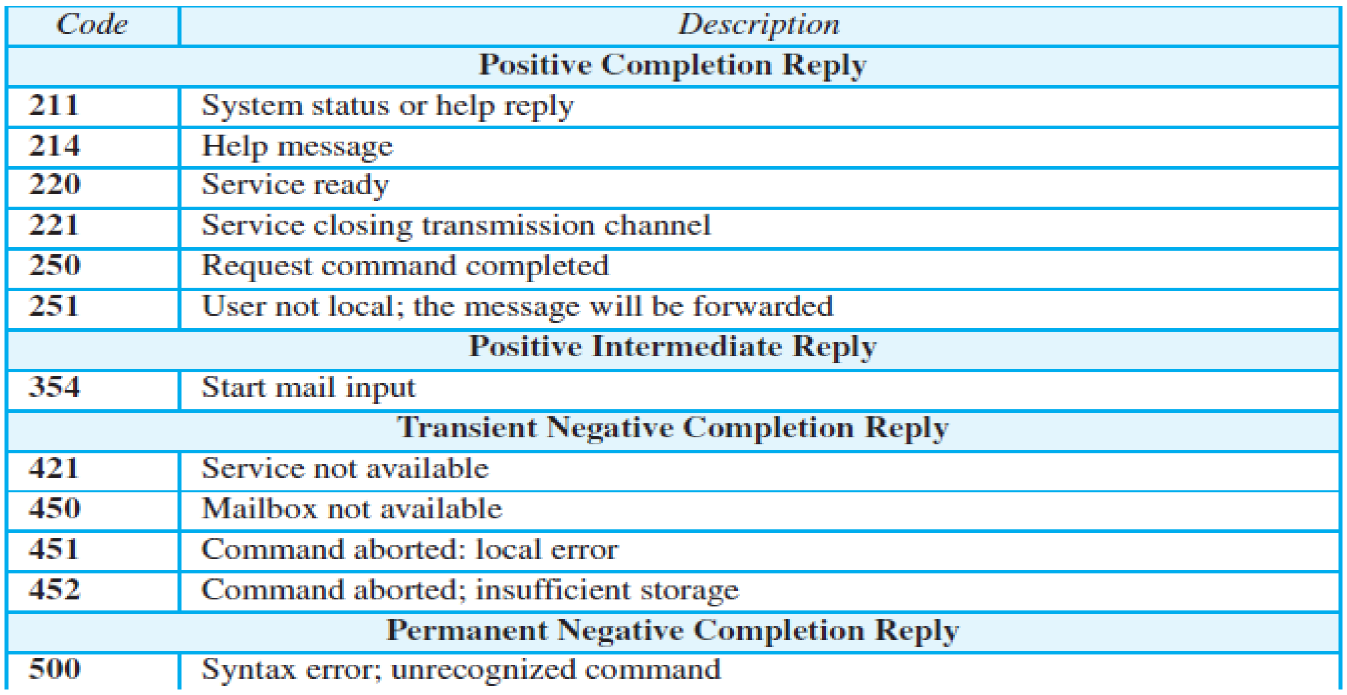


Keyword	*Argument(s)*	*Description*
HELO	Sender's host name	Identifies itself
MAIL FROM	Sender of the message	Identifies the sender of the message
RCPT TO	Intended recipient	Identifies the recipient of the message
DATA	Body of the mail	Sends the actual message
QUIT		Terminates the message
RSET		Aborts the current mail transaction
VRFY	Name of recipient	Verifies the address of the recipient
NOOP		Checks the status of the recipient
TURN		Switches the sender and the recipient
EXPN	Mailing list	Asks the recipient to expand the mailing list
HELP	Command name	Asks the recipient to send information about the command sent as the argument
SEND FROM	Intended recipient	Specifies that the mail be delivered only to the terminal of the recipient, and not to the mailbox
SMOL FROM	Intended recipient	Specifies that the mail be delivered to the terminal *or* the mailbox of the recipient
SMAL FROM	Intended recipient	Specifies that the mail be delivered to the terminal *and* the mailbox of the recipient

Responses

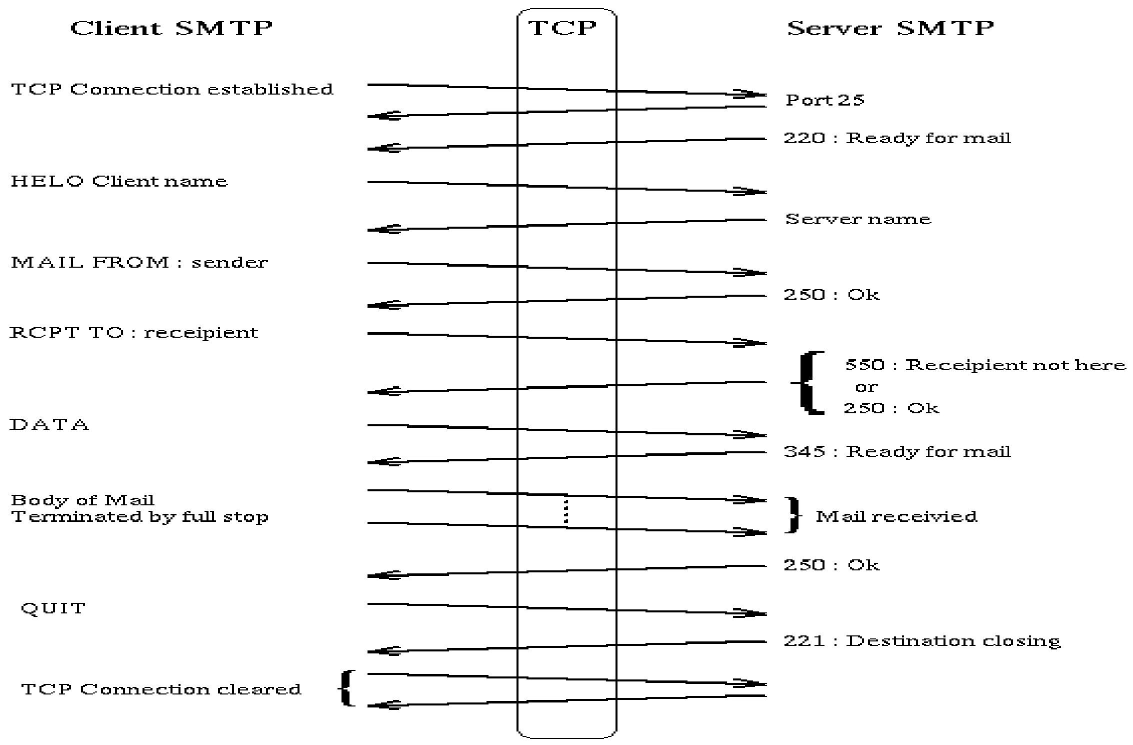
Responses are sent from the server to the client. A response is a three digit code that may be followed by additional textual information.

Table 26.7 *Responses*



Code	*Description*
	Positive Completion Reply
211	System status or help reply
214	Help message
220	Service ready
221	Service closing transmission channel
250	Request command completed
251	User not local; the message will be forwarded
	Positive Intermediate Reply
354	Start mail input
	Transient Negative Completion Reply
421	Service not available
450	Mailbox not available
451	Command aborted: local error
452	Command aborted; insufficient storage
	Permanent Negative Completion Reply
500	Syntax error; unrecognized command

501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command temporarily not implemented
550	Command is not executed; mailbox unavailable
551	User not local
552	Requested action aborted; exceeded storage location
553	Requested action not taken; mailbox name not allowed
554	Transaction failed



<u>Mail Transfer Phases</u>
Connection Establishment
Message Transfer
Connection Termination

<u>PROBLEMS WITH SMTP</u>

1. There is no convenient way to send nonprintable characters
2. There is no way to know if one has received mail or not or has read it or not.
3. Someone else can send a mail on my behalf.

4.Explain in detail about MIME: Multipurpose Internet Mail Extensions.

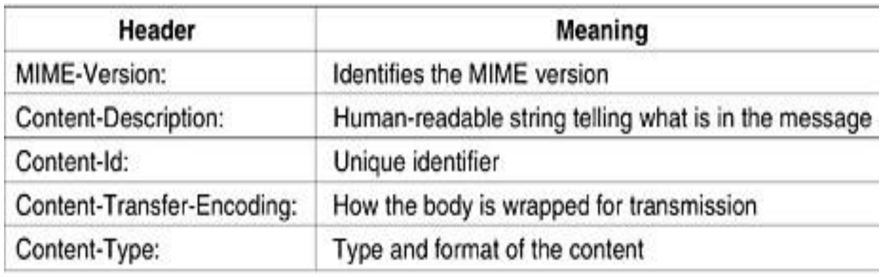
MIME, an acronym for Multipurpose Internet Mail Extensions, specifies how messages must be formatted so that they can be exchanged between different email systems. MIME is a very flexible format, permitting one to include virtually any type of file or document in an email message. MIME messages can contain text, images, audio, video, or other application-specific data. Specifically, MIME allows mail messages to contain:

- Multiple objects in a single message.
- Text having unlimited line length or overall length.
- Character sets other than ASCII, allowing non-English language messages.
- Multi-font messages.
- Binary or application specific files.
- Images, Audio, Video and multi-media messages.

A secure version of MIME, S/MIME (Secure/Multipurpose Internet Mail Extensions), is defined to support encryption of email messages. Based on the MIME standard, S/MIME provides the following cryptographic security services for electronic messaging applications: authentication, message integrity and non-repudiation of origin and privacy and data security.

MIME standard converts (encodes) non-text files into text that is normally unreadable and then, at the other end, reconverts (decodes) the files to their originalform.

MIME defines five new message headers



Header	Meaning
MIME-Version:	Identifies the MIME version
Content-Description:	Human-readable string telling what is in the message
Content-Id:	Unique identifier
Content-Transfer-Encoding:	How the body is wrapped for transmission
Content-Type:	Type and format of the content

MIME-Version: *Any message not containing a MIME-Version: header is assumed to be an English plaintext message and is processed as such.*

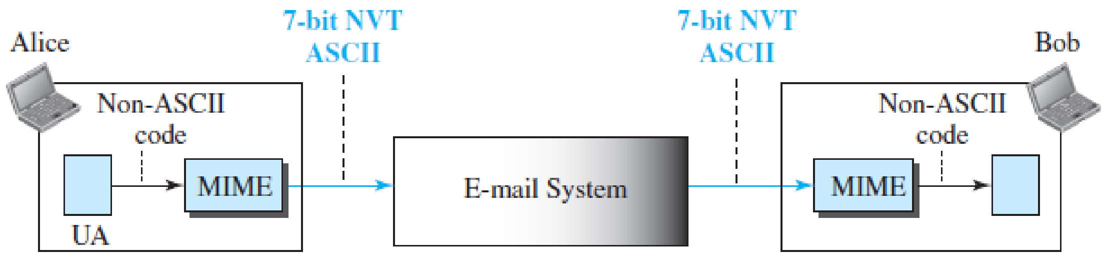
The Content-Description: header is an ASCII string telling what is in the message. This header is needed so the recipient will know whether it is worth decoding and reading the message.

The Content-Id: header identifies the content.

The Content-Transfer-Encoding: tells how the body is wrapped for transmission through a network that may object to most characters other than letters, numbers, and punctuation marks.

Content-Type: It specifies the nature of the message body. Seven types are defined in RFC 2045, each of which has one or more subtypes. The type and subtype are separated by a slash, as in Content-Type: video/mpeg

Figure 26.18 *MIME*



5. Write notes on IMAP, POP3 (8)Nov/Dec 2013 (May 2015,Apr/May 2017)

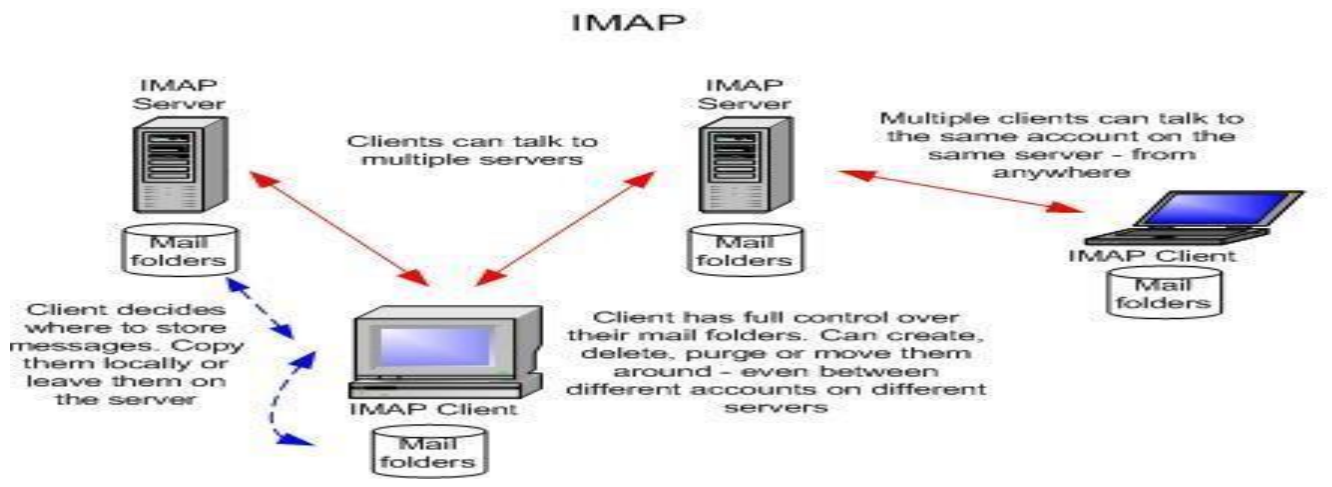
Internet Message Access Protocol (IMAP) is a method of accessing electronic mail that is kept on a mail server. IMAP permits a "client" email program to access remote message stores as if they were local. Clients may store local copies of the messages, but these are considered to be a temporary cache. Email stored on an IMAP server can be manipulated from a desktop computer remotely, without the need to transfer messages or files back and forth between these computers.

- IMAP can do all three modes: offline, online processing and disconnected operations. In the online mode, the mail client does not copy mails in a shared server all at once and then delete them. It is an interactive client-server model, where the client can ask the server for headers, or the bodies of specified messages, or to search for messages meeting certain criteria.

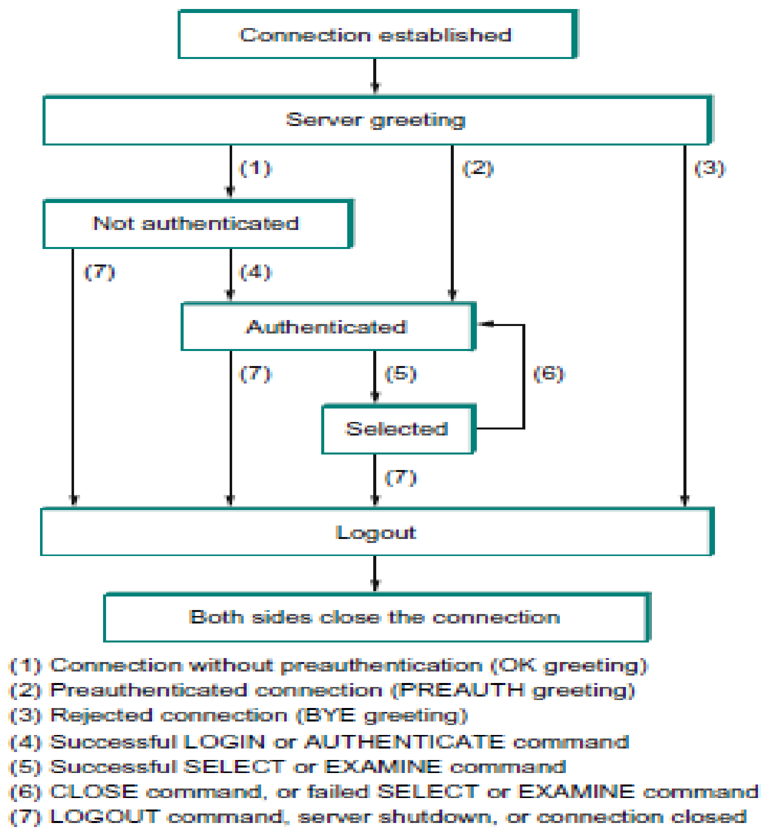
- IMAP includes operations for creating, deleting, and renaming mailboxes; checking for new messages; permanently removing messages; setting and clearing flags; server-based and MIME parsing, and searching; and selective fetching of message attributes, texts, and portions thereof for efficiency. IMAP allows clients to access messages (both new and saved) from more than one computer, that feature has become extremely important as reliance on electronic messaging and use of multiple computers increase.

The current version of IMAP is version 4 revision 1(IMAP4 rev1). Key features for IMAP4 include:

- Fully compatible with Internet messaging standards, e.g. MIME.
- Allow message access and management from more than one computer.
- Provide support for "online", "offline", and "disconnected" access modes
- Support for concurrent access to shared mailboxes
- Client software needs no knowledge about the server's file store format



IMAP
IMAP Server
Mail folders
Clients can talk to multiple servers
IMAP Server
Mail folders
Multiple clients can talk to the same account on the same server - from anywhere
IMAP Client
Mail folders
Client decides where to store messages. Copy them locally or leave them on the server
IMAP Client
Mail folders
Client has full control over their mail folders. Can create, delete, purge or move them around - even between different accounts on different servers



Connection established
Server greeting
(1)
(2)
(3)
Not authenticated
(7)
(4)
Authenticated
(7)
(5)
(6)
Selected
(7)
Logout
Both sides close the connection
(1) Connection without preauthentication (OK greeting)
(2) Preauthenticated connection (PREAUTH greeting)
(3) Rejected connection (BYE greeting)
(4) Successful LOGIN or AUTHENTICATE command
(5) Successful SELECT or EXAMINE command
(6) CLOSE command, or failed SELECT or EXAMINE command
(7) LOGOUT command, server shutdown, or connection closed

FIGURE 9.2 IMAP state transition diagram.

POP3 Protocol (8.M) (Nov 2016)

- The **POP (Post Office Protocol 3)** protocol provides a simple, standardized way for users to access mailboxes and download messages to their computers.
- There are two main versions of this protocol, POP2 and POP3, to which ports 109 and 110 are allocated respectively and which operate using radically different text commands.
- When using the POP protocol all your eMail messages will be downloaded from the mail server to your local computer.
- POP was designed to support "offline" mail processing, in which, mail is delivered to a server, and a personal computer user periodically invokes a mail "client" program that connects to the server and downloads all of the pending mail to the user's own machine. The offline access mode is a kind of

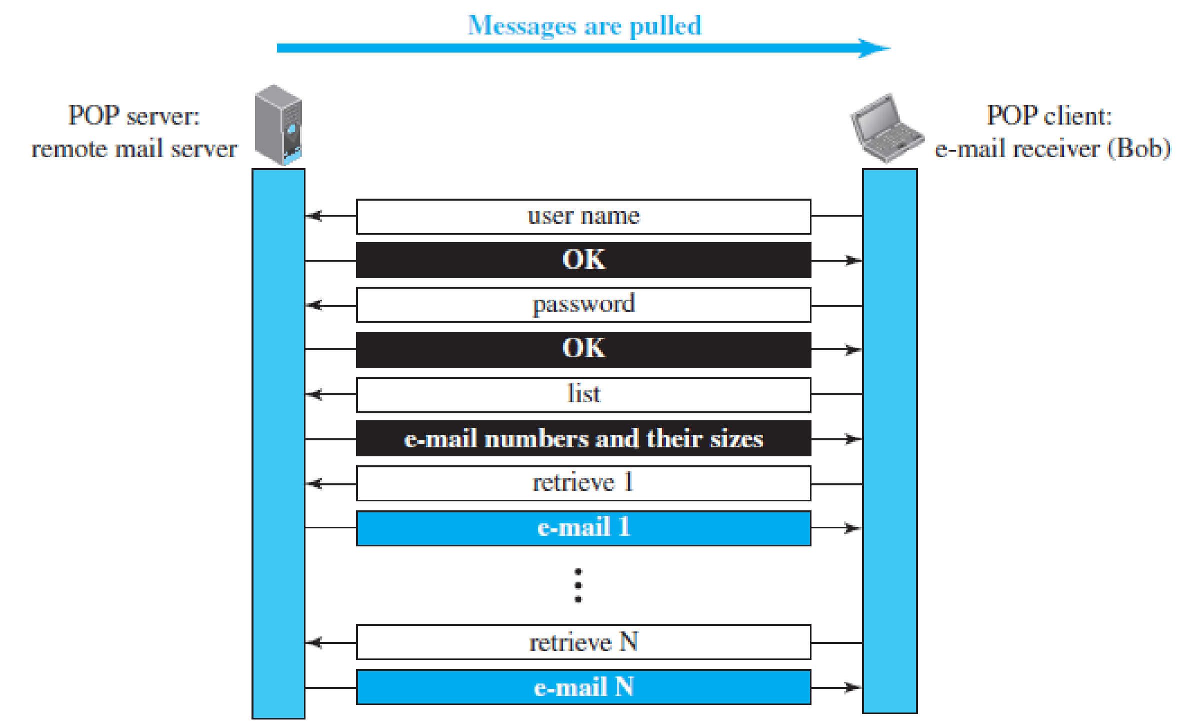
store-and-forward service, intended to move mail (on demand) from the mail server (drop point) to a single destination machine, usually a PC or Mac. Once delivered to the PC or Mac, the messages are then deleted from the mail server.

Once the connection is established, the POP3 protocol goes through three states in sequence:

- Authorization - The authorization state deals with having the user log in.
- Transactions - The transaction state deals with the user collecting the e-mails and marking them for deletion from the mailbox.
- Update – The update state actually causes the e-mails to be deleted

Command	**Description**
USER identification	This command makes it possible to be authenticated. It must be followed by the user name, i.e. a character string identifying the user on the server. The USER command must precede the *PASS* command.
PASS password	The *PASS* command makes it possible to specify the user's password where the name has been specified by a prior *USER* command.
STAT	Information on the messages contained on the server
RETR	Number of the message to be picked up
DELE	Number of the message to be deleted
LIST [msg]	Number of the message to be displayed
NOOP	Allows the connection to be kept open in the event of inactivity
TOP <messageID><n>	Command displaying *n* lines of the message, where the number is given in the argument. In the event of a positive response from the server, it will send back the message headers, then a blank line and finally the first *n* lines of the message.
UIDL [msg]	Request to the server to send back a line containing information about the message possibly given in the argument. This line contains a character string called a *unique identifier listing*, making it possible to uniquely identify the message on the server, independently of the session. The optional argument is a number relating to a message existing on the POP server, i.e. an undeleted message).
QUIT	The *QUIT* command requests exit from the POP3 server. It leads to the deletion of all messages marked as deleted and sends back the status of this action.

Figure 26.17 *POP3*


Messages are pulled
POP server: remote mail server
POP client: e-mail receiver (Bob)
user name
OK
password
OK
list
e-mail numbers and their sizes
retrieve 1
e-mail 1
retrieve N
e-mail N

6. i) What is HTTP protocol used for? (OR) Write notes on URLS(NOV/DEC 2014) (Nov 2015) (May 2016)

ii) What is the default port number of HTTP protocol?

iii) Discuss the features of HTTP and also discuss how HTTP works.

HTTP PROTOCOL

The **HyperText Transfer Protocol (HTTP)** is used to define how the client-server programs can be written to retrieve web pages from the Web. An HTTP client sends a request; an HTTP server returns a response. The server uses the port number 80; the client uses a temporary port number.

- ➢ Protocol for transfer of data between Web servers and Web clients (browsers).
- ➢ "The Hypertext Transfer Protocol (HTTP) is an **application-level protocol** for **distributed**, collaborative, hypermedia information systems.
- ➢ Popular Web servers:
  - Apache HTTPD, JBoss and Tomcat
- ➢ Popular Web clients:
  - Firefox and Opera

HTTP Properties

1) A comprehensive addressing scheme

The HTTP protocol uses the concept of reference provided by the Universal Resource Identifier (URI) as a location (URL) or name (URN), for indicating the resource on which a method is to be applied.

- Every resource accessible through HTTP is identified by a Uniform Resource Location (URL), which is a location-specific identifier.
    - For example,
        - http://www.cs.uct.ac.za:80/
        - ftp://ftp.cs.uct.ac.za/
- A Uniform Resource Identifier (URI) is a standard format (<scheme>:<identifier>) generic identifier.
    - For example,
        - mailto:hussein@cs.uct.ac.za
- A Uniform Resource Name (URN) is one example of a location-independent URI.
    - For example urn:isbn:123-456-789

2) Client-Server architecture

The HTTP protocol is based on a request/response paradigm. The communication generally takes place over a TCP/IP connection on the Internet. The default port is 80, but other ports can be used. A requesting program (a client) establishes a connection with a receiving program (a server) and sends a request to the server in the form of a request method, URI, and protocol version, followed by a message containing request modifiers, client information, and possible body content. The server responds with a status line, including its protocol version and a success or error code, followed by a message containing server information, entity meta_information, and possible body content.

3) HTTP protocol is connectionless

This HTTP protocol is called connectionless because once the single request has been satisfied, the connection is dropped.

4) HTTP protocol is stateless

After the server has responded to the client's request, the connection between client and server is dropped and forgotten. There is no "memory" between client connections. The pure HTTP server implementation treats every request as if it was brand-new (without context), i.e. not maintaining any connection information between transactions.

Other HTTP Features

- Persistent connections
- Cache control

Persistent Connections

Persistent connections provide a mechanism by which a client and a server can signal the close of a TCP connection. With them, it is possible to establish a TCP connection, send a request and get a response, and then send additional requests and get additional responses. By amortizing the TCP setup and release over multiple requests, the relative overhead due to TCP is much less per request. It is also possible to pipeline requests, that is, send request 2 before the response to request 1 has arrived.

Persistent HTTP connections have a number of advantages:

- By opening and closing fewer TCP connections, CPU time is saved in routers and hosts (clients, servers, proxies, gateways, tunnels, or caches), and memory used for TCP protocol control blocks can be saved in hosts.
- HTTP requests and responses can be pipelined in a connection. Pipelining allows a client to make multiple requests without waiting for each response, allowing a single TCP connection to be used much more efficiently, with much lower elapsed time.

Caching

The goal of caching in HTTP is to eliminate the need to send requests in many cases, and to eliminate the need to send full responses in many other cases. That is, there are two main reasons that web caching is used:

- To reduce latency because the request is satisfied from the cache (which is closer to the client) instead of the origin server, it takes less time for the client to get the object and display it. This makes Web sites seem more responsive.
- To reduce traffic because each object is only gotten from the server once, it reduces the amount of bandwidth used by a client. This saves money if the client is paying by traffic, and keeps their bandwidth requirements lower and more manageable.

Nonpersistent versus Persistent Connections

- ***Nonpersistent Connections***

In a **nonpersistent connection**, one TCP connection is made for each request/response.
The following lists the steps in this strategy:
1. The client opens a TCP connection and sends a request.
2. The server sends the response and closes the connection.
3. The client reads the data until it encounters an end-of-file marker; it then closes the connection.

- ***Persistent Connections***

HTTP version 1.1 specifies a **persistent connection** by default. In a persistent connection, the server leaves the connection open for more requests after sending a response. The server can close the connection at the request of a client or if a time-out has been reached. The sender usually sends the length of the data with each response.

Message Formats

The HTTP protocol defines the format of the request and response messages

Request Message

HTTP Methods

HTTP allows an open-ended set of methods to be used to indicate the purpose of a request. The three most often used methods are **GET, HEAD**, and **POST**.

Method	Description
OPTIONS	capabilities of resource/server
GET	retrieve resource
HEAD	retrieve headers for resource
POST	submit data to server
PUT	replace/insert resource on server
DELETE	remove resource from server
TRACE	trace request route through Web

The GET method

The GET method requests the server to send the page. The page is suitably encoded in MIME. The vast majority of requests to Web servers are GETs. The usual form of GET is GET filename HTTP/1.1 Where filename names the resource (file) to be fetched and 1.1 is the protocol version being used.

The Head Method

The HEAD method is used to ask only for information about a document, not for the document itself. HEAD is much faster than GET, as a much smaller amount of data is transferred. It's often used by clients who use caching, to see if the document has changed since it was last accessed. If it was not, then the local copy can be reused, otherwise the updated version must be retrieved with a GET.

The PUT method

The PUT method is the reverse of GET: instead of reading the page, it writes the page. This method makes it possible to build a collection of Web pages on a remote server.

The POST method

The POST method is used to transfer data from the client to the server.

DELETE

DELETE does what you might expect: it removes the page. There is no guarantee that DELETE succeeds, since even if the remote HTTP server is willing to delete the page

TRACE

The TRACE method is for debugging. It instructs the server to send back the request. This method is useful when requests are not being processed correctly and the client wants to know what request the server actually got.

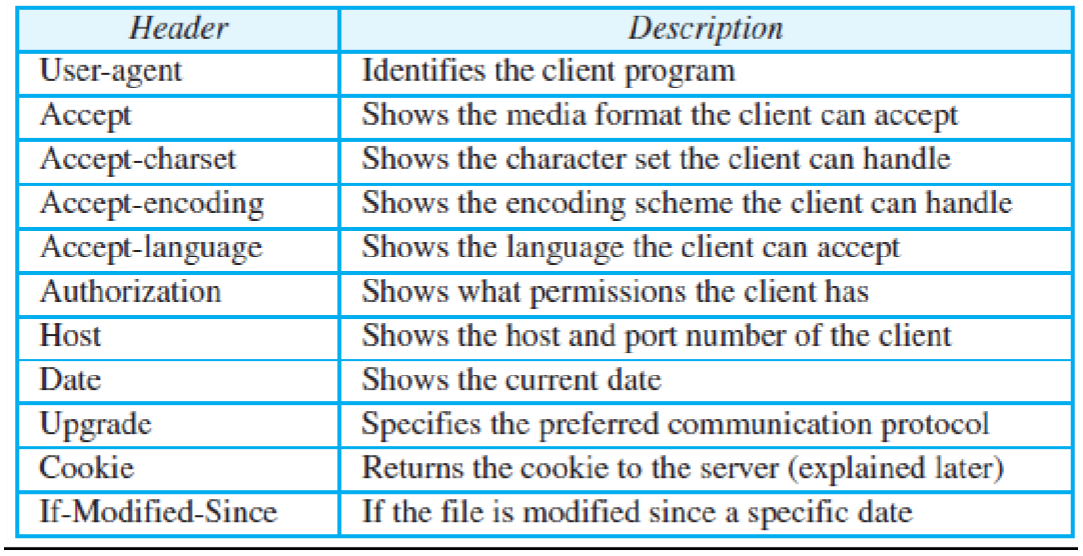
OPTION The OPTIONS method provides a way for the client to query the server about its properties or those of a specific file. Telling whether the request was satisfied, and if not, why not.

Status	Reason	Description
200	OK	Successful request
206	Partial Content	Successful request for partial content
301	Moved Permanently	Resource has been relocated
304	Not Modified	Conditional GET but resource has not changed
400	Bad Request	Request not understood

403	Forbidden	Access to resource not allowed
404	Not Found	URI/resource not found on server
500	Internal Server Error	Unexpected error

<u>HTTP Header Fields(Nov/Dec 2007)</u>

Table 26.2 *Request header names*



Header	*Description*
User-agent	Identifies the client program
Accept	Shows the media format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client has
Host	Shows the host and port number of the client
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Cookie	Returns the cookie to the server (explained later)
If-Modified-Since	If the file is modified since a specific date

An HTTP transaction consists of a header followed optionally by an empty line and some data. The header will specify such things as the action required of the server, or the type of data being returned, or a status code. The use of header fields sent in HTTP transactions gives the protocol great flexibility. These fields allow descriptive information to be sent in the transaction, enabling authentication, encryption, and/or user identification. The header is a block of data preceding the actual data, and is often referred to as meta information, because it is information about information.

- ***Accept***: Indicates which data formats are acceptable.
  - Accept: text/html, text/plain
- ***HTTP_User-Agent***.

  The browser the client is using to send the request.
  General format: software/version library/version.
- ***Content-Language***: Language of the content
  - Content-Language: english
- ***Content-Length***: Size of message body
  - Content-Length: 1234
- ***Content-Type***: MIME type of content body
  - Content-Type: text/html

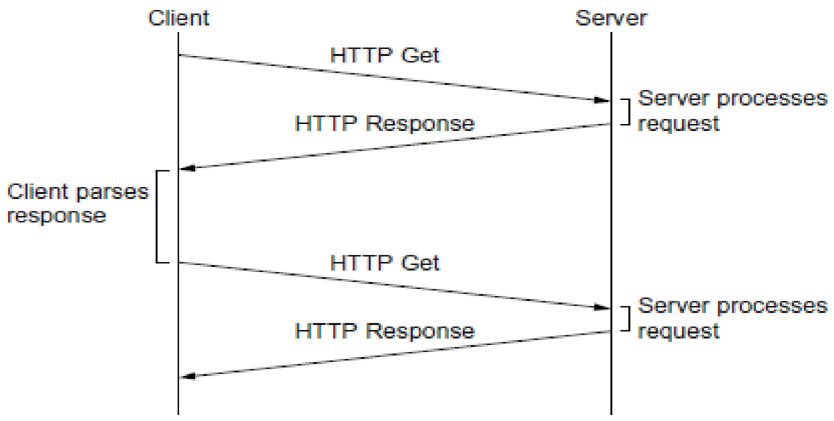


FIGURE 9.5 HTTP 1.1 behavior with persistent connections.

- ***Date***: The Date header represents the date and time at which the message was originated
  - Date: Tue, 15 Nov 1994 08:12:31 GMT
- ***Expires***: When content is no longer valid
  - Expires: Tue, 15 Nov 1994 08:12:31 GMT
- ***Host***: Machine that request is directed to
  - Host: www.cs.uct.ac.za

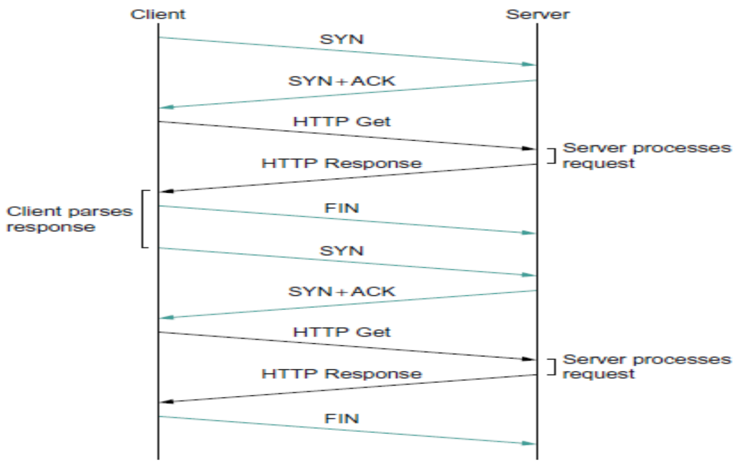


FIGURE 9.4 HTTP 1.0 behavior.

- ***Location***:
  The Location response header field defines the exact location of the resource that was identified by the request URI. If the value is a full URL, the server returns a "redirect" to the client to retrieve the specified object directly.
  - Location: http://myserver.org/
- ***Retry-After***: Indicates that client must try again in future
  - Retry-After: 120

Response Message

A response message consists of a status line, header lines, a blank line, and sometimes a body. The first line in a response message is called the *status line*. There are three fields in this line separated by spaces and terminated by a carriage return and line feed

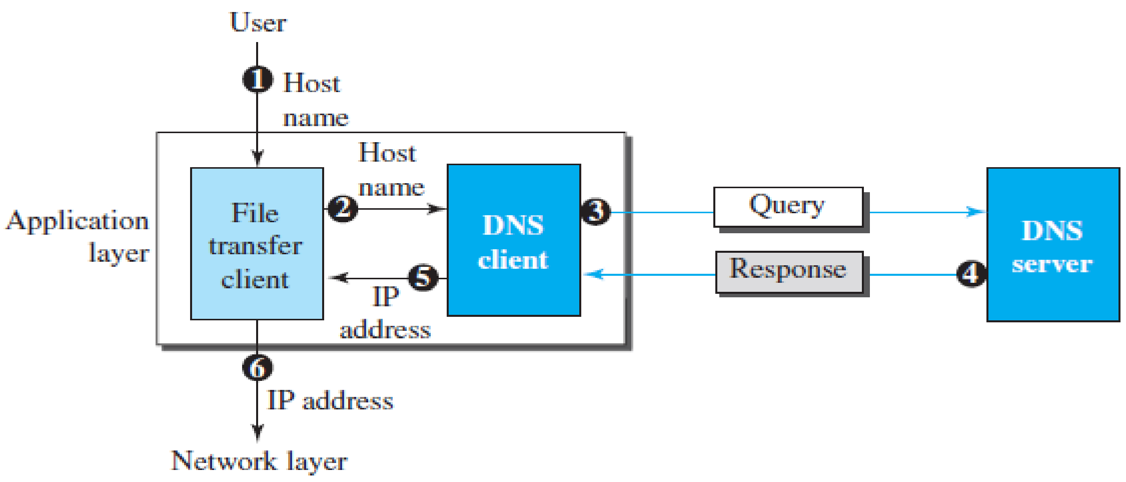
Table 26.3 *Response header names*

Header	*Description*
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Server	Gives information about the server
Set-Cookie	The server asks the client to save a cookie
Content-Encoding	Specifies the encoding scheme
Content-Language	Specifies the language
Content-Length	Shows the length of the document
Content-Type	Specifies the media type
Location	To ask the client to send the request to another site
Accept-Ranges	The server will accept the requested byte-ranges
Last-modified	Gives the date and time of the last change

7. Discuss briefly DNS (Domain Name System)& its advantages *(Apr /may2010) (Nov/Dec 2014)(Nov 2015 & 2016)(Apr/May 2017)*

The **DNS** translates Internet domain and host names to IP addresses. DNS automatically converts the names we type in our Web browser address bar to the IP addresses of Web servers hosting those sites.

Figure 26.28 *Purpose of DNS*



The following six steps map the host name to an IP address:
1. The user passes the host name to the file transfer client.
2. The file transfer client passes the host name to the DNS client.

3. Each computer, after being booted, knows the address of one DNS server. The DNS client sends a message to a DNS server with a query that gives the file transfer server name using the known IP address of the DNS server.
4. The DNS server responds with the IP address of the desired file transfer server.
5. The DNS server passes the IP address to the file transfer client.
6. The file transfer client now uses the received IP address to access the file transfer server.

Namespace:

The names assigned to computers must be selected from a name space. The name must be unique because the addresses are unique. A namespace that maps each address to a unique name can organize in two ways.

1. Flat Namespace
2. Hierarchical Namespace

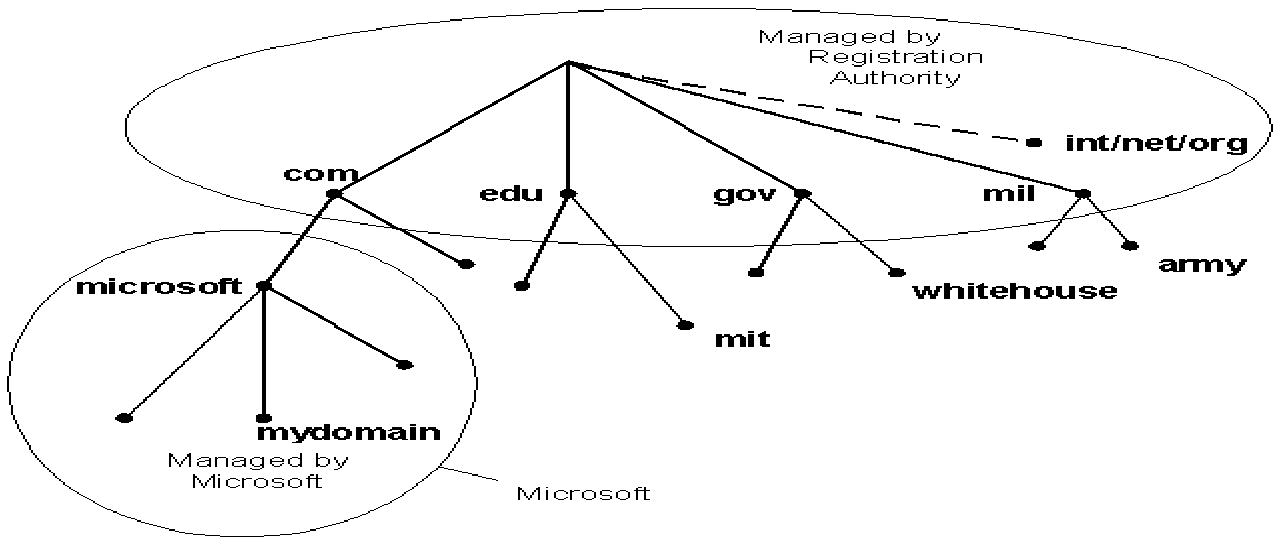
Flat Namespace A name is assigned to an address. A name in this space is a sequence of characters without structure. The main disadvantage of flat namespace is that, it cannot use in a large system such as the internet.

Hierarchical Namespace Each name is made of several parts. The first part can defined the nature of the organization, the second part can defined the name, and the third part can define department and so on. The authority to assign and control the namespaces can be decentralized.

Domain Hierarchy:
DNS is hierarchical in structure. A domain is a subtree of the domain name space. All the related information about a particular network (generally maintained by an organization, firm or university) should be available at one place. The organization should have complete control over what it includes in its network and how does it "organize" its network. Meanwhile, all this information should be available transparently to the outside world.

Conceptually, the internet is divide into several hundred top level domains where each domain covers many hosts. Each domain is partitioned in subdomains which may be further partitioned into subsubdomains and so on... So the domain space is partitioned in a tree like structure as shown below.

The internet uses a hierarchical tree structure of Domain Name Servers for IP address resolution of a host name.



The top level domains are either generic or names of countries. eg of generic top level domains are .edu .mil .gov .org .net .com .int etc. For countries we have one entry for each country as defined in ISO3166. eg. .in (India) .uk (United Kingdom).

The leaf nodes of this tree are target machines. Obviously we would have to ensure that the names in a row in a subdomain are unique. The max length of any name between two dots can be 63 characters. The absolute address should not be more than 255 characters. Domain names are case insensitive. Also in a name only letters, digits and hyphen are allowed. For eg. www.iitk.ac.in is a domain name corresponding to a machine named www under the sub domain iitk.ac.in.

Domain Name
A name that identifies one or more *IP addresses*. For example, the domain name *microsoft.com* represents about a dozen IP addresses. Domain names are used in URLs to identify particular Web pages. For example, in the URL *http://www.pcwebopedia.com/index.html*, the domain name is *pcwebopedia.com.*

Types of Domain Name

1. Fully Qualified Domain Name (FQDN)
2. Partially Qualified Domain Name (PQDN)

    1. FQDN: A fully qualified domain name (FQDN) consists of the host name plus domain name. e.g. **computername.domain.com**
    2. PQDN: A partially Qualified Domain Name (PQDN) stats from a node, but it does not reach the root. E.g. **computername**

Three main components of DNS

1. Resolver
2. Name server
3. Database of Resource Records(RRs)

Resolver: A host that needs to map an address to a name or a name to an address calls a DNS client called a resolver. The resolver accesses the closest DNS server with a mapping request. If the server has the information, its satisfies the resolver; after the resolver receives the mapping, it interprets the response to see if it's a real resolution or an error, and finally delivers the result to the process that requested it.

i) Mapping names to address
The resolver gives a domain name to the server and ask for the corresponding address.

ii) Mapping address to names
A client can send an IP address to a server to be mapped to a domain name.

iii) Recursive resolution
The resolver can ask for a recursive answer from a name server. This means that the resolver expects the server to supply the final answer. If the server is the authority for the domain name, it checks its database

and response. When the query is finally resolved, the response travel back until it finally reaches the requesting client. This is called recursive resolution.

iv) Iterative resolution

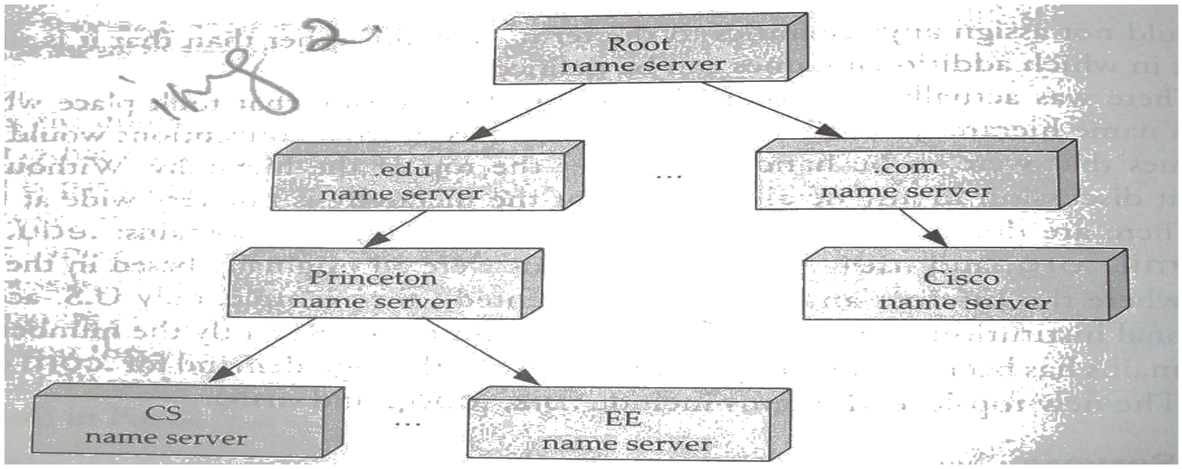
If the client doesn't ask for a recursive answer, the mapping can done iteratively. If the newly addressed the server can resolve the problem, it answers the query with the IP address. Otherwise it returns the IP address of a new server to a client. Now the client must repeat the query to the second server. This process is called iterative resolution because the client repeats the same query to multiple servers.

v) Caching

Each time a server receives a query for a name that is not in its domain, it needs to search its database for a server IP address. Reduction of this time put increase efficiency. When a server asks for a mapping from another server and receives the response, it stores this information in its cache memory before sending it to the client. If the same or another client asks for the same mapping, it can check its cache memory and solve the problem. This mechanism is called caching.

Name Servers

- The first step is to partition the hierarchy into sub trees called zones. Each zone can be thought of a corresponding to some administrative authority that is responsible for that portion of the hierarchy.
- DNS server is used to distribute the information among many computers. Specifically, the information contained in each zone is implemented in two or more name servers for the sake of redundancy, that is, the information is still available even if one name server fails. Each name server, in turn, is a program that can be accessed over the Internet.
- Client send queries to name servers, and name servers respond with the requested information. Sometimes the response contains the final answer that the client wants, and sometimes the response contains a pointer to another server that the client should query next.



Name Server Types

Name server types are:

1. **Root Server :**A root server is a server whose zone consist of the whole tree. A root server usually does not store any information about domains. But delegates it's authority to other servers, keeping references to those servers.
2. **Primary Server:** A Primary server is a server that stores a file about the zone for which it is an authority. It is responsible for creating, maintaining and updating the zone file. It stores the zone file on a local disk.
3. **Secondary Server:** A secondary server transfers the complete information about a zone from another server and stores the file on its local disk. The secondary server neither creates nor updates the zone files. If updating is required it must be done by the primary server, which sends the updated version to the secondary. When the secondary downloads information from the primary it is called zone transfer.

Types of Records

There are two types of records are used in DNS.

1. The question records
2. Resource Records

The question records: Is used by the client to get information from a server. This contains a domain name.
Resource Records: Each domain name is associated with a record called the resource record. The server database consists of resource records. The resource records are used in the answer, authoritative and additional information section of the response message.
Each name server implements the zone information as a collection of resource records. In essence, a resource record is a name-to-value binding, a 5-tuple that contains the following fields:

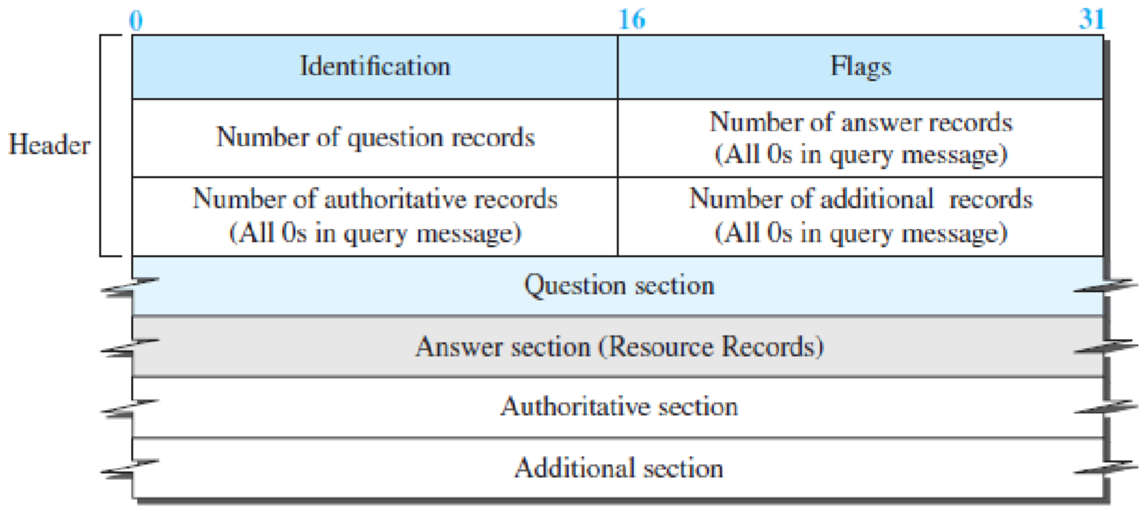
- **Domain name**: the domain to which this record applies.
- **Class**: set to IN for internet information. For other information other codes may be specified.
- **Type**: tells what kind of record it is.
- **Time to live**: Upper Limit on the time to reach the destination
- **Value**: can be an IP address, a string or a number depending on the record type.

Resource Record Type	Contents	Use
A	Host Address	Used to hold a specific host's IP address.
CNAME	Canonical Name (alias)	Used to make an alias name for a host.
MX	Mail Exchanger	Provides message routing to a mail server, plus backup server(s) in case the target server isn't active.
NS	Name Server	Provides a list of authoritative servers for a domain or indicates authoritative DNS servers for any delegated sub-domains.
PTR	Pointer	Used for reverse lookup—resolving an IP address into a domain name using the IN-ADDR.ARPA domain.

SOA	Start of Authority	Used to determine the DNS server that's the primary server for a DNS zone and to store other zone property information.

DNS Messages
To retrieve information about hosts, DNS uses two types of messages: *query* and *response*. Both types have the same format

Figure 26.38 *DNS message*


0
16
31
Header
Identification
Flags
Number of question records
Number of answer records (All 0s in query message)
Number of authoritative records (All 0s in query message)
Number of additional records (All 0s in query message)
Question section
Answer section (Resource Records)
Authoritative section
Additional section

Note:
The query message contains only the question section.
The response message includes the question section, the answer section, and possibly two other sections.

DDNS

When the DNS was designed, no one predicted that there would be so many address changes. In DNS, when there is a change, such as adding a new host, removing a host, or changing an IP address, the change must be made to the DNS master file. These types of changes involve a lot of manual updating. The size of today's Internet does not allow for this kind of manual operation The DNS master file must be updated dynamically.

The **Dynamic Domain Name System (DDNS)** therefore was devised to respond to this need. In DDNS, when a binding between a name and an address is determined, the information is sent, usually by DHCP to a primary DNS server To provide security and prevent unauthorized changes in the DNS records, DDNS can use an authentication mechanism.

Advantages:

More Reliable: Delivers messages to the users with zero downtime.

Faster: DNS are connected well at intersections of internet. Anycast technology enables requests are answered to the next closest node in the case of maintenance or downtime.

Smarter: Automatic corrections of typos.

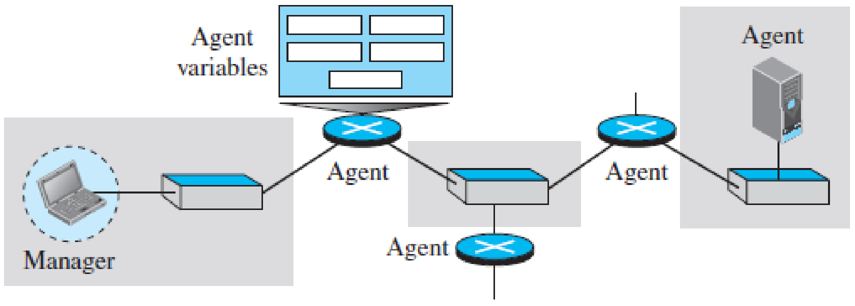
8. BrieflyExplain the concept of SNMP (Simple Network Management Protocol) (Apr /may2011) (May 2015) (Nov 2016)

SNMP is a frame work for managing devices in an internet using TCP/IP suite. It provides fundamental operations for monitoring and maintaining an internet.

Concept:

- SNMP uses the concept of manager and agent. Manager usually a host controls and monitors a set of agents, usually routers. A management station, called a manager, is a host that runs the SNMP client program. A managed station, called an agent, is a router or host that runs the SNMP server program.
- Management is achieved through simple interaction between a manager and an agent. The agent keeps performance information in a database. The manager has access to the values in the database.

Figure 27.2 *SNMP concept*



Managers and Agents

A management station, called a *manager*, is a host that runs the SNMP client program. A managed station, called an *agent*, is a router (or a host) that runs the SNMP server program. Management is achieved through simple interaction between a manager and an agent. The agent keeps performance information in a database. The manager has access to the values in the database.

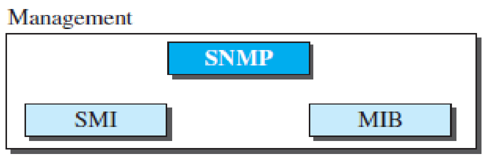
Management with SNMP is based on three basic ideas

1. A manager checks an agent by requesting information that reflects the behavior of the agent.
2. A manager forces an agent to perform a task by resetting values in the agent database.
3. An agent contributes the management process by warning the manager of an unusual situation.

Management Components

To do management tasks, SNMP uses two other protocols: **Structure of Management Information (SMI)** and **Management Information Base (MIB).** In other words, management on the Internet is done through the cooperation of three protocols: SNMP, SMI, and MIB

Figure 27.3 *Components of network management on the Internet*

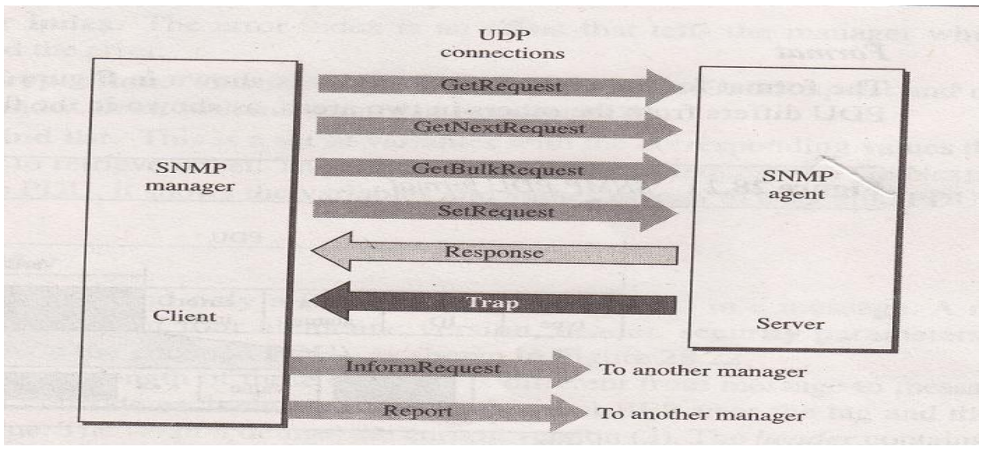


Role of SNMP
SNMP has some very specific roles in network management. It defines the format of the packet to be send from a manager to an agent and vice versa. It also interprets the result and creates statistics. The packet exchange contains the object names (variables) and their status (values). SNMP is response for reading and changing these values.

Role of SMI
SMI defines the general rules for naming objects, defining object types (including range and length), and showing how to encode objects and values.

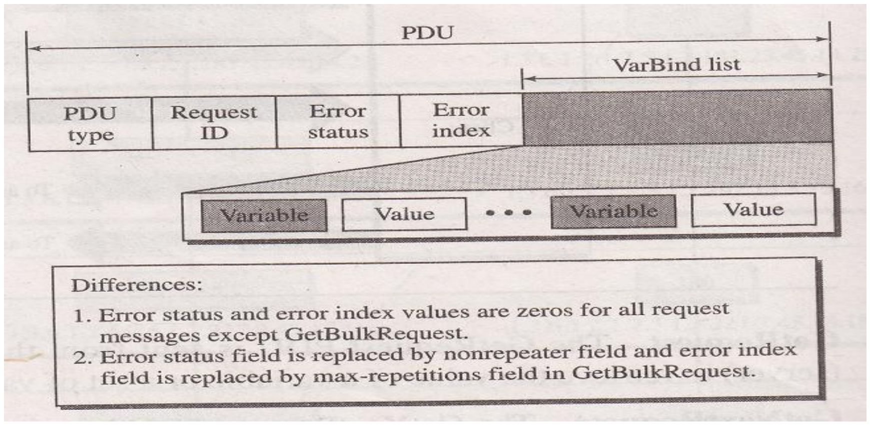
Role of MIB
MIB creates a collection of named objects, their types, and their relationships to each other in an entity to be managed

PDU's
SNMP V3 defines 8 types of packets



PDU Type	Name	Description
1	get-request	Get one or more variables .(manager to agent)
2	get-next-request	Get next variable after one or more specified variables. (manager to agent)
3	Get-bulk-request	To retrieve a large amount of data
4	set-request	Set one or more variables. (manager to agent)
5	get-response	Return value of one or More variables. (agent to manager)
6	trap	Notify manager of an event. (agent to manager)
7	Inform request	To get the value of some variable from agents under the control of the remote manager. The remote manager response with a response PDU. (one manager to another remote manager)
8	report	To report some types of errors between managers.

SNMP PDU Format



Differences:
1. Error status and error index values are zeros for all request messages except GetBulkRequest.
2. Error status field is replaced by nonrepeater field and error index field is replaced by max-repetitions field in GetBulkRequest.

- **PDU type -** This field defines the type of the PDU.
- **Request ID** – This field is a sequence number used by the manager in a request PDU and repeated by the agent in a response. It is used to match a request to a response.
- **Error status** – This is an integer that is used only in response PDU's to show the types of errors reported by the agent. Its value is zero in request PDU's.

Types of errors:

Error	Name	Description
0	no error	OK
1	too big	Reply does not fit into one message
2	no such name	The variable specified does not exist
3	bad value	Invalid value specified in a set request.
4	read only	The variable to be changed is read only.
5	general error	General error

Non Repeater: This field is used only in get-bulk-request and replaces the error status field which is empty in request PDU's.

- **Error Index**: Error index is an offset that tells the manager which variable caused the error.
  Max-repetition: This field is also used only in get-bulk-request and replaces the error index filed, which is empty in request PDU's
- **VarBind List**: This is a set of variables with corresponding values the manager wants to retrieve or set. The values or null in get-request and get-next-request.

SNMP messages:

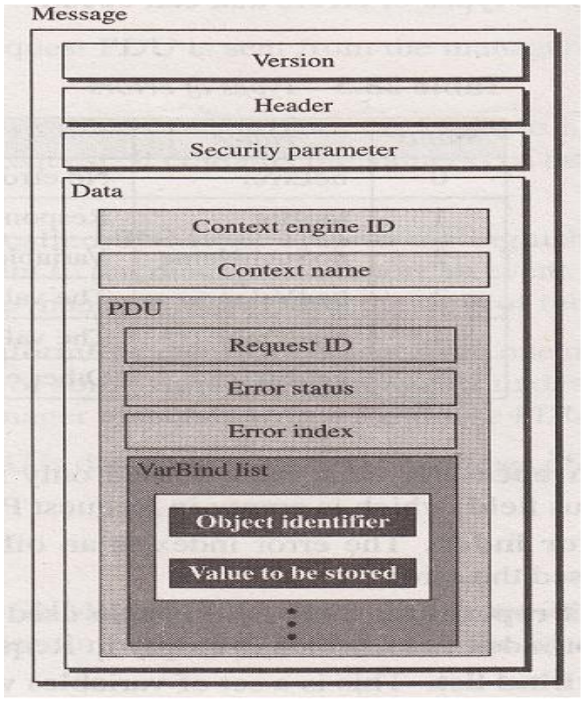
- SNMP does not send only a PDU, it embeds the PDU in a message. A message in SNMPv3 is made of four elements: version, header, security parameter and data.
- The **version**, defines the current version (3)
- The **header** contains values for message identification, maximum message size, message flag and a message security model.
- The message **security parameter** is used to create a message digest.
- The **data** contain the PDU. If the data are encrypted, there is information about the encrypting engine and the encrypting context followed by the encrypted PDU. If the data are not encrypted, the data consist of just the PDU.

UDP Ports:

SNMP uses the services of UDP on two well-known ports, 161 and 162. The well-known port 161 is used by the server (agent), and the well-known port 162 is used by the client (Manager).

Security:

- SNMPv3 provides two types of security: general and specific.
- SNMPv3 provides message authentication, privacy, and manager authorization.
- SNMPv3 allows a manager remotely change the security configuration, which means that the manager does not have to be physically present at the manager station.



9. Expalin the concept of TELNET (8.m)

TELNET is an abbreviation for TErminaL NETwork. TELNET enables the establishment of a connection to a remote system in such a way that the local terminal appears to be a terminal at the remote system. TELNET general purpose client server application program.

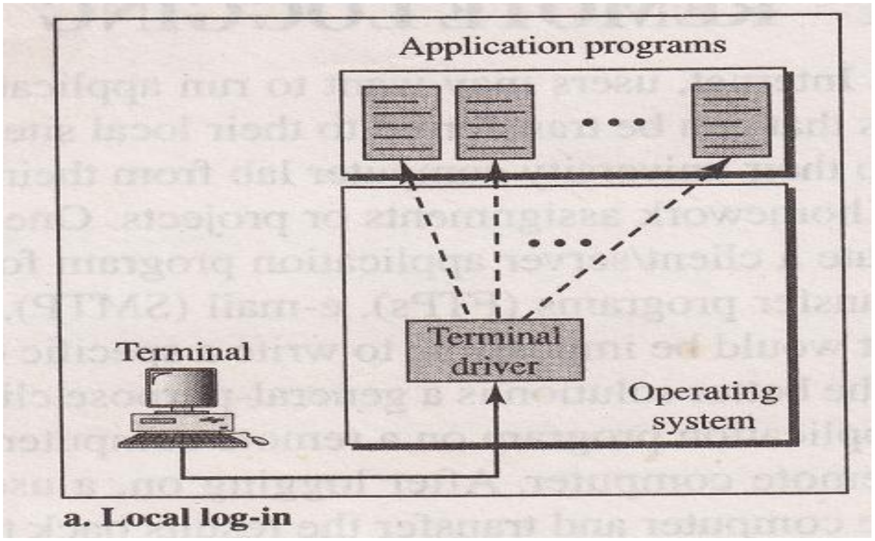
Using the Telnet protocol user on a local host can remote-login and execute commands on another distant host

Time sharing environment:

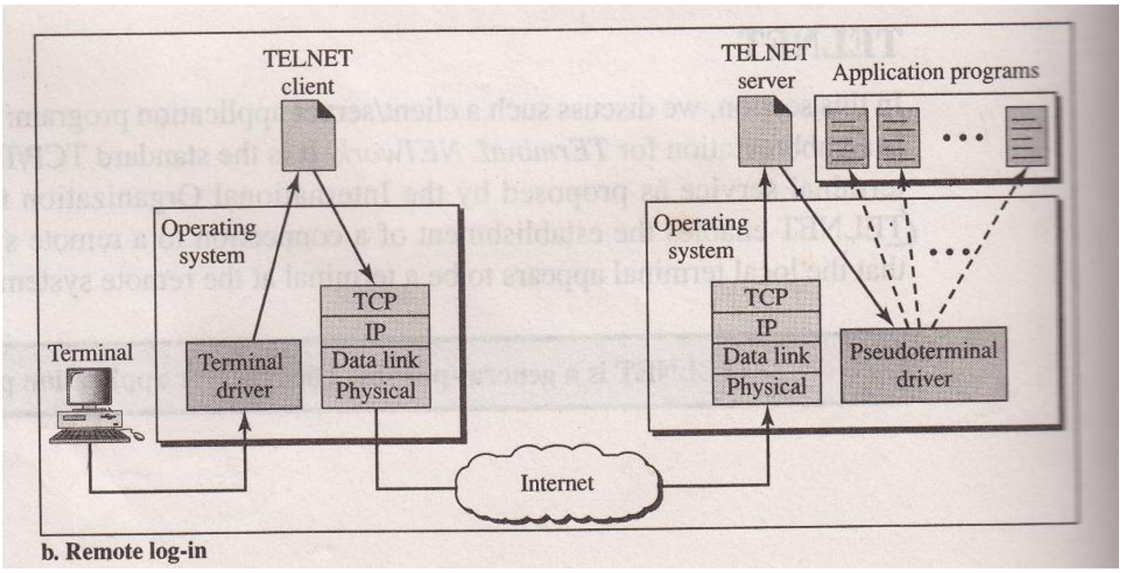
TELNET was designed at a time when most operating systems, such as UNIX, were operating in a time sharing environment. In this environment, a large computer supports multiple users. The interaction between a user and the computer occurs through a terminal, which is usually a combination of keyboard, monitor and mouse.

Logging:

In a time sharing environment, users are the part of the system with some rights to access the resources. To access the system, the authorized user logs in to the system with a user id or login name. This system also includes password checking to prevent an unauthorized users from accessing the resources.



a. Local log-in



b. Remote log-in

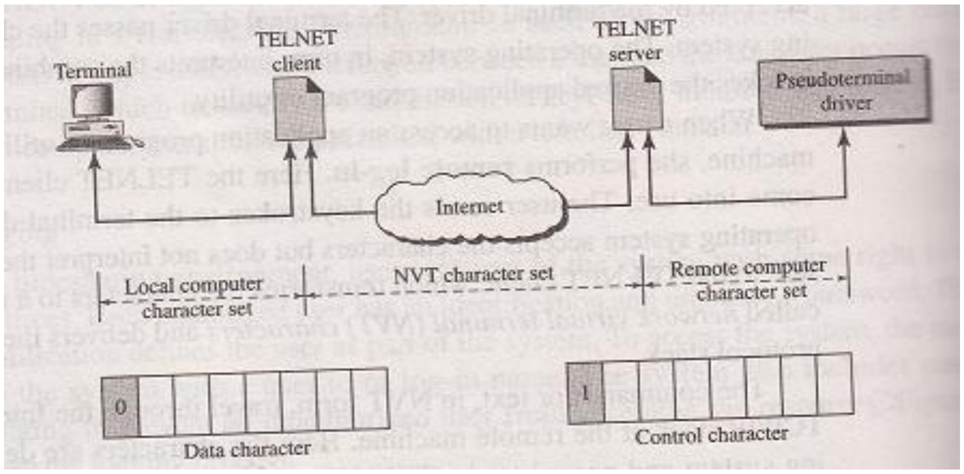
- When a user logs into a local timesharing system, it is called local log-in. As a user types at a terminal the keystrokes are accepted by the terminal deriver. The terminal driver passes the characters to the operating system. The operating system, in turn, interprets the combination of characters and invokes the desired application program or utility.

- When a user wants to access an application program or utility located on a remote machine, she performs remote log-in. Here the TELNET client and server programs come into use. The user sends the keystrokes to the terminal driver, where the local operating system accepts the characters but does not interpret them. The characters are sent to the TELNET client, which transforms the characters to a universal character set called network virtual terminal (NVT) characters and delivers them to the local TCP/IP protocol stack.

- The commands or text, in NVT form, travel through the Internet and arrive at the TCP/IP stack at the remote machine. Here the characters are delivered to the operating system and passed to the TELNET server, which changes the characters to the corresponding characters understandable by the remote computer. However, the characters cannot be passed directly to the operating system because the remote operating system is not designed to receive characters from a TELNET server: It is designed to receive characters from a terminal driver. The solution is to add a piece of software called a pseudo terminal driver which pretends that the characters are coming from a terminal. The operating system then passes the characters to the appropriate application program.

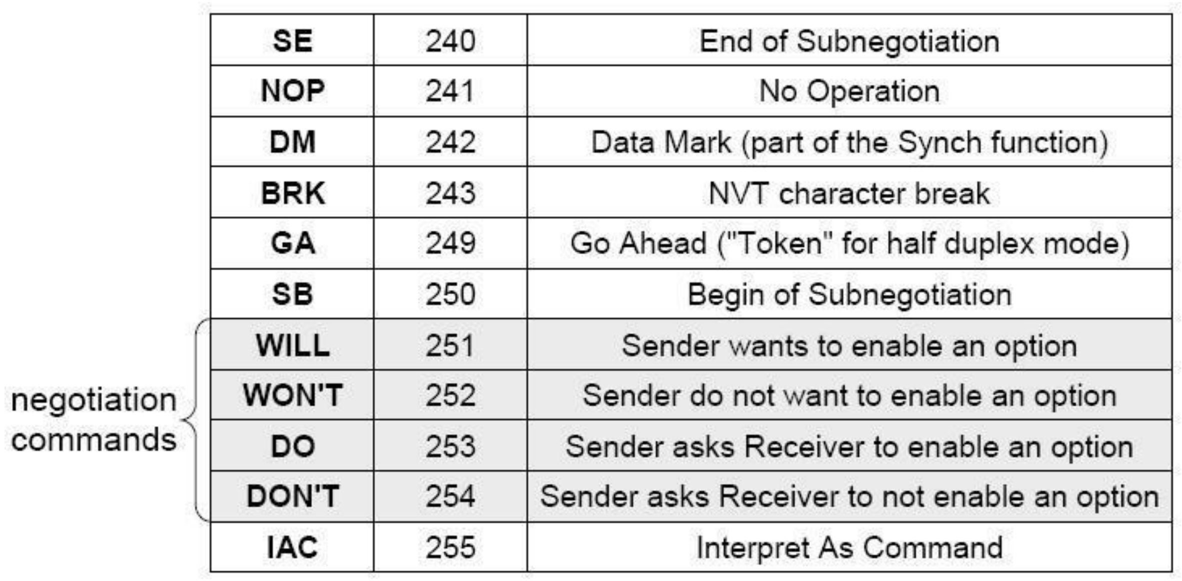
- **Network Virtual Terminal (NVT):**

Network Virtual Terminal, which transforms the characters to a universal character set and delivers them to the local TCP/IP stack.



NVT’s Character Set

- NVT uses two sets of characters one for data and other for control.
- NVT generally use the 8 bit character set for both.
- NVT’s data character set is the US ASCII 7-bit code.
- NVT can handle the printable characters with ASCII codes 32-126 plus a small set of control characters:


negotiation commands

SE	240	End of Subnegotiation
NOP	241	No Operation
DM	242	Data Mark (part of the Synch function)
BRK	243	NVT character break
GA	249	Go Ahead ("Token" for half duplex mode)
SB	250	Begin of Subnegotiation
WILL	251	Sender wants to enable an option
WON'T	252	Sender do not want to enable an option
DO	253	Sender asks Receiver to enable an option
DON'T	254	Sender asks Receiver to not enable an option
IAC	255	Interpret As Command

Embedding:
TELNET uses only one TCP connection. The same connection is used for sending both data and control characters. TELNET accomplishes this by embedding the control characters in the data stream. However, to

distinguish data from control characters, each sequence of control characters is preceded by a special control character called interpret as control (IAC).

Options:

TELNET lets the client and server negotiate options before or during the use of service. Options are extra features available to a user with a more sophisticated terminal. Some common options:

Code	**Option**	**Meaning**
0	Binary	Interpret as 8-bit transmission
1	Echo	Echo the data received on one side to the other
3	Suppress go ahead	Suppress go-ahead signals after data
5	Status	Request the status of TELNET
6	Timing mark	Define the timing marks
24	Terminal type	Set the terminal type
32	Terminal speed	Set the terminal speed.
34	Line mode	Change to line mode

Option Negotiation:

To use any of the options mentioned in the previous section first requires option negotiation between the client and the server. In this four control character are used

Character	**Decimal**	**Binary**	**Meaning**
WILL	251	11111011	1. Offering to enable 2. Accepting a request to enable
WONT	252	11111100	1. Rejecting a request to enable 2. Offering to disable 3. Accepting a request to disable
DO	253	11111101	1. Approving an offer to enable 2. Requesting to enable
DON’T	254	11111110	1. Disapproving an offer to enable 2. Approving an offer to disable 3. Requesting to disable.

A party can offer to enable or disable an option if it has the right to do so. The offering can be approved or disapproved by the other party. To offer enabling, the offering party sends the WILL command, which means “Will I enable the option?” The other party sends either the DO command, which means “please do,” or the DON’T command, which means “Please don’t.” To offer disabling, the offering party sends the WONT command, which means “I won’t use this option anymore.” The answer must be the don’t command, which means “Don’t use it anymore.”

Sub option Negotiation:

Some option requires addition information. To define the type or speed of a terminal, the negotiation includes a string or a number to define the type or speed.

Character	**Decimal**	**Binary**	**Meaning**
SE	240	11110000	Suboption end
SB	250	11111010	Suboption begin

<u>Mode of operation:</u>

TELNET implementations operate in one of three modes. **Default mode**, **Character mode**, or **Line mode**.

Default Mode:

The default mode is used if no other modes are invoked through option negotiation. In this mode the echoing is done by the client. The user types a character, and the client echoes the character on the screen, but does not send it until a whole line is completed.

Character Mode:
In the character mode, each character typed is send by the client to the server. The server normally echoes the character back to be displayed on the client screen. In this mode the echoing of the character can be delayed with the transmission time is long.
Line Mode:
A new mode has been proposed to compensate for the deficiencies of the default mode and character mode. In this mode line editing is done by the client. The client then sends the whole line to the server.

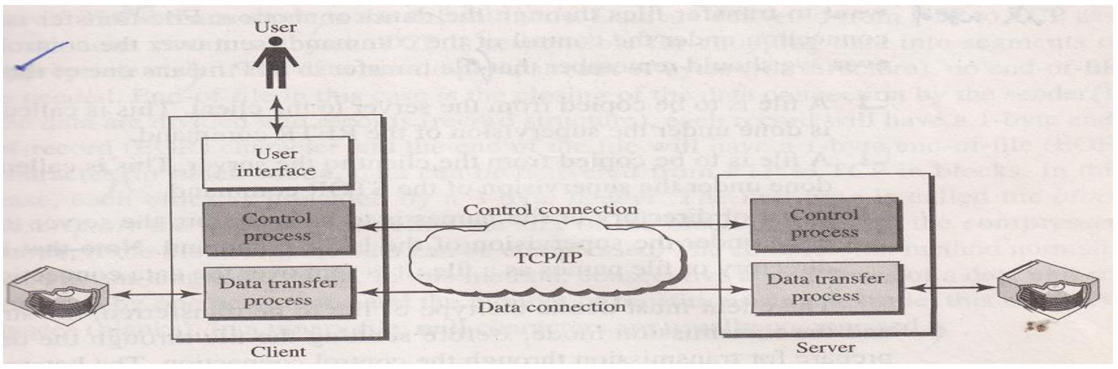
10. **Discuss the various commands used in FTP (12) (MAY/JUNE 2009) (or) Discuss FTP with suitable diagram (Apr /may2011) (8)**

FTP (File Transfer Protocol)

- FTP is the standard mechanism provided by TCP/IP for copying a file from one host to another. Although transferring files from one system to another seems simple and straight forward.
- FTP differs from other client server applications, in that it establishes to connections between the hosts. One connection is used for data transfer, the other for control information (commands and responses). The control connection uses very simple rules of communications. We need to transfer only a line of command or a line of response at a time. The data connection needs more complex rules, due to the variety of data types transferred.
- FTP uses the services of the TCP. It needs two TCP connections, the port 21 is used for control connection and the port 20 is used for data connection.

Basic Model of FTP:

➢ The client has three components: **user interface**, **client control process** and the **client data transfer process.**
➢ The server has two components: the **server control process** and the **server data transfer process**.
➢ The control connection is need between the control processes. The data connection is made between the data transfer processes.



The control connection remains, connected during the entire interactive FTP session.
The data connection is opened and then closed for each file transferred.

Communication over Control Connection:
FTP uses the 7 bit ASCII character set to communicate across the control connection. Communication is achieved through commands and responses. In this, send one command or response at a time. Each command are response is only one short line, so need not worry about file format or file structure. Each line is terminated with a two character end of line token.

Communication over Data Connection:
It is used to transfer files through the data connection. File transfer occurs over the data connection under the control of the commands send over the control connection. File transfer in FTP means one of three things:

1. A file is to be copied from the server to the client. This is called **retrieving a file**. It is done under the supervision of the RETR command.
2. A file is to be copied from the client to the server. This is called **storing a file**. It is done under the supervision of STOR command.
3. A list of directories or file names is to be sending from the server to the client. This is done under the supervision of LIST command. FTP treats a list of directory or file names as a file. It is send over the data connection.

The client must define the **type of file to be transferred**, the **structure of the data** and **the transmission mode**. Before sending the file through the data connection, prepare for transmission for the control connection.

File Type:
FTP can transfer one of the following file types across the data connection: an ASCII file, EBCDIC file or IMAGE file.

- ➢ The **ASCII file** is the default format for transferring text files. Each character is encoded using 7-bit ASCII.
- ➢ A file can be transferred using **EBCDIC file.** Abbreviation of **E**xtended **B**inary-**C**oded **D**ecimal **I**nterchange **C**ode. EBCDIC is an IBM code for representing characters as numbers.
- ➢ The **IMAGE file** is the default format for transferring binary files.

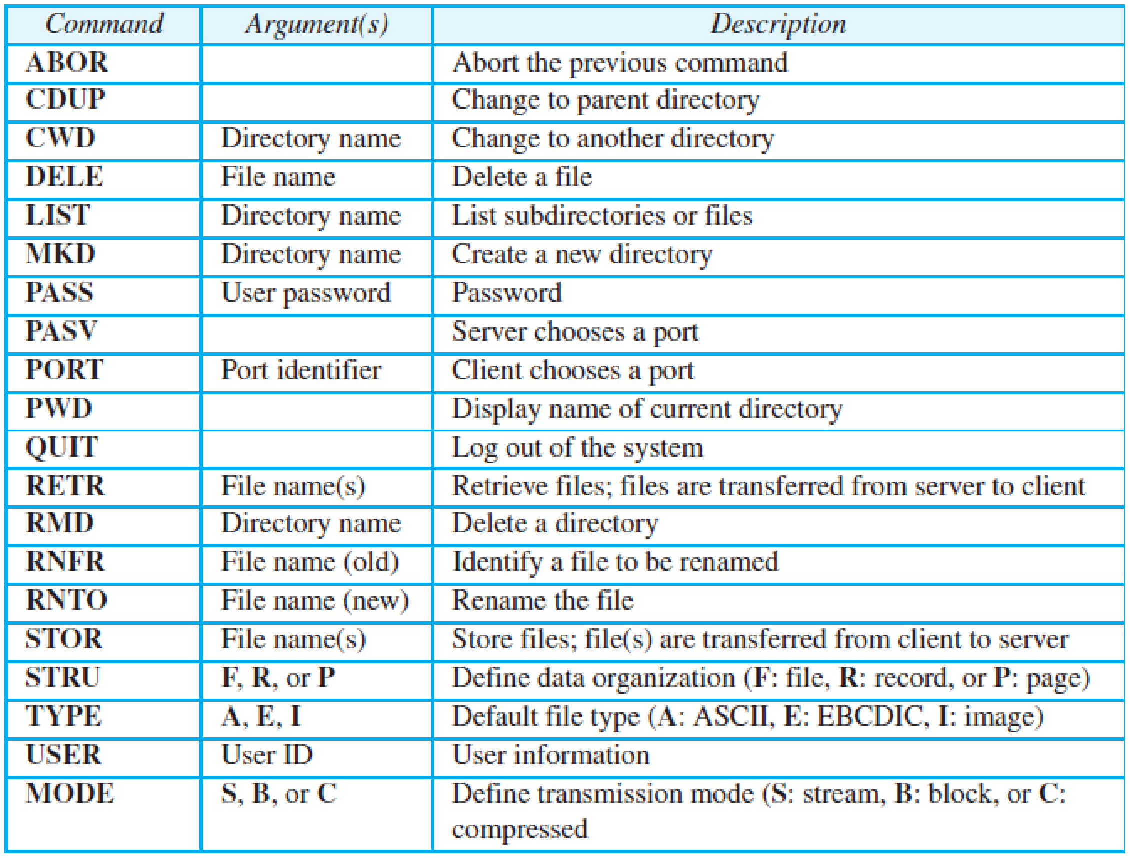
Data Structure:
FTP can transfer a file across the data connection by using one of the following interpretations about the structure of the data: **file structure**, **record structure** and **page structure**.

- ➢ In the **file structure** format, the file is a continuous stream of bytes.
- ➢ In the **record structure** format, the file is divided in to records. This can be used only with text files.
- ➢ In the **page structure** format, the file is divided into pages, with each page having a page number and a page header. The pages can be stored and accessed randomly and sequentially.

Transmission Mode:
FTP can transfer a file across the data connection by using one of the following three transmission modes: **stream mode**, **block mode** and **compressed mode**.
In **stream mode**, the data are delivered from FTP to TCP as a continuous stream of bytes. TCP is responsible for chopping data into segments of appropriate sizes. If the data are divided in to records, each record will have a one byte end of record (EOR) character and end of the file will have a one byte EOF character.
In **block mode**, the data can delivered from FTP to TCP blocks; in this case each block is preceded by a three byte header. The first byte is called the block descriptor; the next two bytes define the size of the block in bytes.
In the **compressed mode**, if the file is big the data can be compressed. The compressed method normally used is run length encoding. In this method, consecutive appearances of a data unit are replaced by one occurrence and the no of repetitions. In a text file this is usually spaces. In a binary null characters are compressed.

FTP Commands

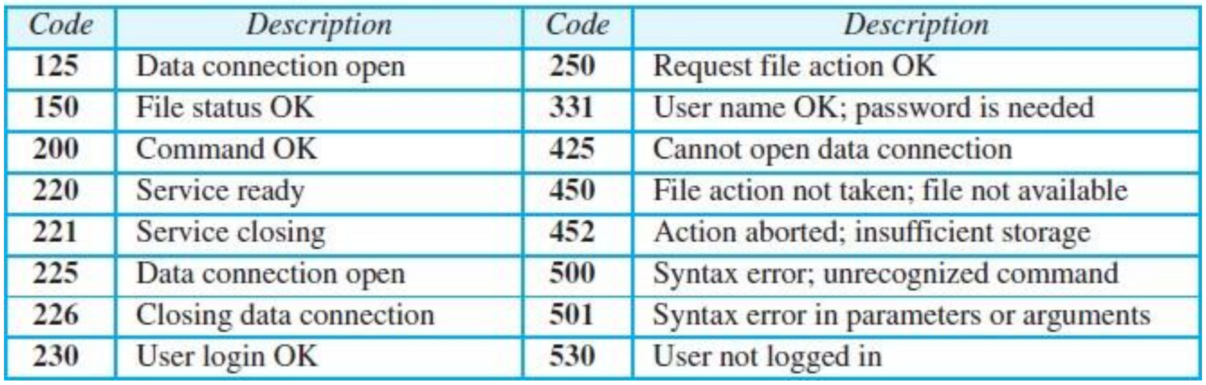
Table 26.4 *Some FTP commands*



Command	*Argument(s)*	*Description*
ABOR		Abort the previous command
CDUP		Change to parent directory
CWD	Directory name	Change to another directory
DELE	File name	Delete a file
LIST	Directory name	List subdirectories or files
MKD	Directory name	Create a new directory
PASS	User password	Password
PASV		Server chooses a port
PORT	Port identifier	Client chooses a port
PWD		Display name of current directory
QUIT		Log out of the system
RETR	File name(s)	Retrieve files; files are transferred from server to client
RMD	Directory name	Delete a directory
RNFR	File name (old)	Identify a file to be renamed
RNTO	File name (new)	Rename the file
STOR	File name(s)	Store files; file(s) are transferred from client to server
STRU	**F**, **R**, or **P**	Define data organization (**F**: file, **R**: record, or **P**: page)
TYPE	**A**, **E**, **I**	Default file type (**A**: ASCII, **E**: EBCDIC, **I**: image)
USER	User ID	User information
MODE	**S**, **B**, or **C**	Define transmission mode (**S**: stream, **B**: block, or **C**: compressed

Every FTP command generates at least one response. A response has two parts: a three-digit number followed by text. The numeric part defines the code; the text part defines needed parameters or further explanations. The first digit defines the status of the command. The second digit defines the area in which the status applies. The third digit provides additional information.

Table 26.5 *Some responses in FTP*

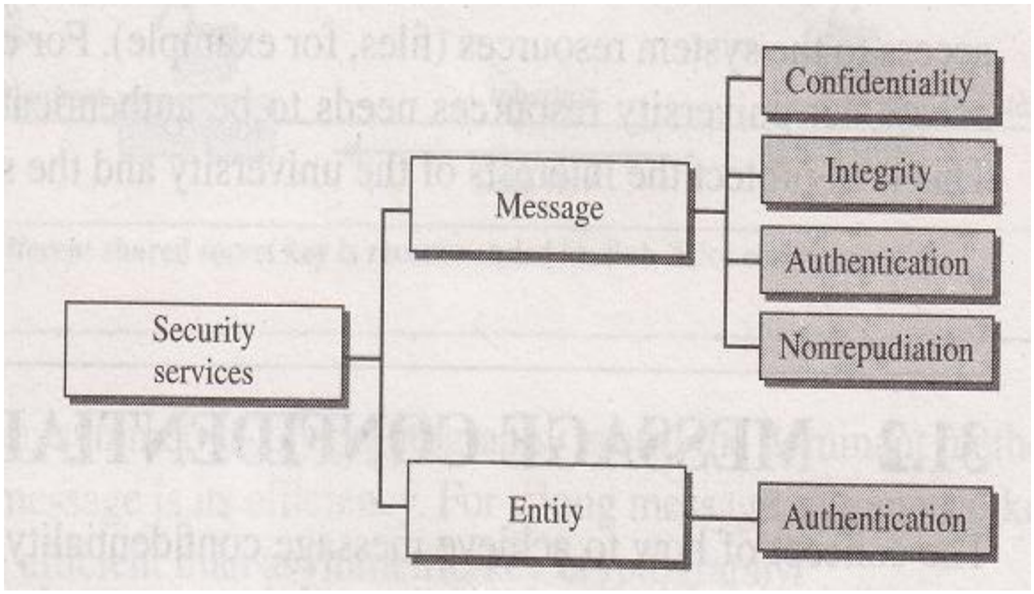


Code	*Description*	*Code*	*Description*
125	Data connection open	**250**	Request file action OK
150	File status OK	**331**	User name OK; password is needed
200	Command OK	**425**	Cannot open data connection
220	Service ready	**450**	File action not taken; file not available
221	Service closing	**452**	Action aborted; insufficient storage
225	Data connection open	**500**	Syntax error; unrecognized command
226	Closing data connection	**501**	Syntax error in parameters or arguments
230	User login OK	**530**	User not logged in

11. Write notes on Security protocol SSH in detail

Security in networking is based on cryptography (secret writing), the science and art of transforming messages to make them secure and immune to attack. Cryptography can provide confidentiality, integrity, authentication and non repudiation of messages.

Network Security can provide one of the 5 services.



- ***Message confidentiality:*** *Message confidentiality or privacy means that the sender and receiver expect confidentiality. The transmitted message must make sense to only the intended receiver. To all others, the message must be garbage.*
- ***Message integrity:*** *Message integrity means that the data must arrive at the receiver exactly as they were sent. There must be no changes during the transmission, neither accidently nor maliciously.*
- ***Message Authentication:*** *Message authentication is a service beyond message integrity. In message authentication the receiver needs to be sure of the sender's identity and that an imposter has not sent the message.*

- ***Message Non repudiation:*** *Message Non repudiation means that a sender must not be able to deny sending a message that he or she did sent. The burden of proof falls on the receiver.*

Entity Authentication: *In entity authentication or user authentication, the entity or user is verified prior to access to the system resources.*

PGP (Pretty Good Privacy)

It is a protocol to provide security at the application layer. PGP is designed to create authenticated and confidential emails.

SSH (Secure Shell)

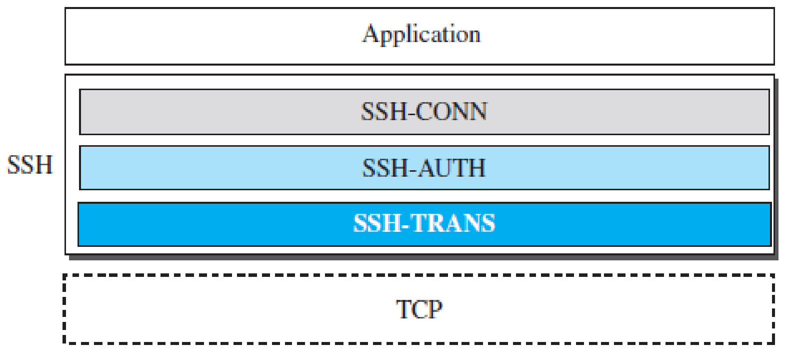
- SSH is a protocol for secure remote login and other secure network services over an insecure network. The Secure Shell (SSH) provides a remote login service and is intended to replace the less secure Telnet and rlogin programs used in the early days of the Internet.

- SSH is most often used to provide strong client/server authentication—where the SSH client runs on the user's desktop machine and the SSH server runs on some remote machine that the user wants to log into—but it also supports message integrity and confidentiality. Telnet and rlogin provide none of these capabilities.
- SSH provides a way to encrypt the data sent over these connections and to improve the strength of the authentication mechanism they use to login.

Major SSH components

- SSH Transport Layer Protocol
  - provides server authentication, confidentiality, and integrity services
  - it may provide compression
  - runs on top of any reliable transport layer (e.g., TCP)
- SSH User Authentication Protocol
  - provides client-side user authentication
  - runs on top of the SSH Transport Layer Protocol
- SSH Connection Protocol
  - multiplexes the secure tunnel provided by the SSH Transport Layer and User Authentication Protocols into several logical channels

Figure 26.25 *Components of SSH*



SSH security features

- Strong algorithms
  - uses well established strong algorithms for encryption, integrity, key exchange, and public key management
- Large key size
  - requires encryption to be used with at least 128 bit keys
  - supports larger keys
- Algorithm negotiation
  - encryption, integrity, key exchange, and public key algorithms are negotiated
  - it is easy to switch to some other algorithm without modifying the base protocol

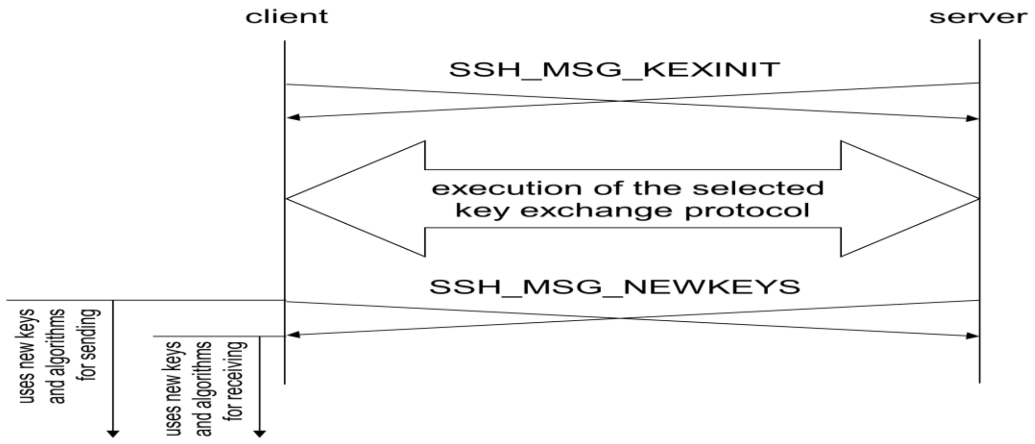
Transport Layer Protocol

- Client initiates connection to the server
- Identification string exchange
- Algorithms exchange
- Key exchanges include host key sent to client
  - Diffie-Hellman key exchange
  - Client selects a random session key

Connection setup and version string exchange

- SSH works over any 8-bit binary transport protocol, e.g., TCP
- Client initiates the connection on the port 22 on the server
- Underlying transport protocol should provide protection against transmission errors, e.g., TCP provides reliable byte stream service.
- Once the connection has been established, both client and server send a version exchange id string of the form "SSH-proto version-software version comments" followed by carriage return & new line character.
- Before the id string is sent, the server might send other strings with useful information to the client.
- The client should be capable of handling these strings and may/may not display it to the user.
- These are used by TCP wrappers to display an error message before disconnecting.
- Key exchange begins after the initial client-server version string exchange.

<u>Key exchange – Overview</u>


client
server
SSH_MSG_KEXINIT
execution of the selected key exchange protocol
SSH_MSG_NEWKEYS
uses new keys and algorithms for sending
uses new keys and algorithms for receiving

User Authentication Protocol

- the protocol assumes that the underlying transport protocol provides integrity and confidentiality (e.g., SSH Transport Layer Protocol)
- the protocol has access to the session ID
- the server should have a timeout for authentication and disconnect if the authentication has not been accepted within the timeout period
  - recommended value is 10 minutes
- the server should limit the number of failed authentication attempts a client may perform in a single session
  - recommended value is 20 attempts
- three authentication methods are supported
  - public key
  - password
  - host based

Connection Protocol

- provides
  - interactive login sessions
  - remote execution of commands
  - forwarded TCP/IP connections
- all these applications are implemented as "channels"
- all channels are multiplexed into the single encrypted tunnel provided by the SSH Transport Layer Protocol
- channels are identified by channel numbers at both ends of the connection
- channel numbers for the same channel at the client and server sides may differ

Anna university question paper

B.E/B.TECH NOVEMBER/DECEMBER 2014

2 MARKS

1. State the difference between SMTP and MIME. (Q.NO 3)
2. List down the key lengths supported by PGP (Q.NO 28)

16 MARKS

1. Write notes on URLS (16) (Q.NO 4)
2. (i) Discuss the advantages of DNS (8) (Q.NO 6)
   (ii) Explain Telnet in detail (8) (Q.NO 9)

B.E/B.Tech April May 2015

2 MARKS

1. Define SMTP (Q.NO 1)
2. What are the groups of HTTP header? (Q.NO 29)

16 MARKS

1. .i.Explain the message transfer using Simple Mail Transfer Protocol.(8) (Q.NO 2)
ii.Explain the final delivery of email to the end user using POP3.(8) (Q.NO 4)
2.Write short notes on. i.Web services (Q.NO 8) ii.SNMP(Q.NO 7)

B.E/B.Tech Nov-Dec 2015

2 MARKS

1. Mention the types of HTTP messages. (Q.NO 32)
2. What is SMTP? (Q.NO 1)

16 MARKS

1. Explain in detail about domain name system (Q.NO 6)
2. Write short notes onEmail&HTTP (Q.NO 1 &5)

B.E/B.Tech April-May 2016

2 MARKS

1. Define URL. (Q.NO 30)
2. Mention the different levels in domain name space. (Q.NO 31)

16 MARKS

1. a)Describe how SMTP protocol is used in E-mail applications. (Q.NO1,2)
   b) Explain HTTP with an example (Q.NO 5)
2. Explain in detail about Web service architecture (Q.NO 8)

B.E/B.Tech Nov-Dec 2016

2 MARKS

1. Expand POP3 and IMAP4. (Q.NO 13 & 14)
2. What is persistent HTTP. (Q.NO 16)

16 MARKS

1. Give a detailed note on DNS operation (Q.NO 6)
2. a)Explain in detail about SNMP messages. (Q.NO 7)
   b)Illustrate the role of POP3 in Electronic mail Applicatons (Q.NO 4)

B.E/B.Tech Apr-May 2017

PART A

1. State the usage of conditional get in HTTP(Q.No 33)
2. Present the information contained in a DNS resource record?(Q.No 34)

PART B

1. i) Describe how SMTP transfer message from one host to another with suitable illustration?(Q.No 2)
   ii) Explain IMAP with its state transition diagram.?(Q.No 4)
2. i) What is Domain Name System(DNS)?Explain(Q.No 6)
   ii) Brief about the importance of Simple Network Management Protocol(SNMP) (Q.No 7)

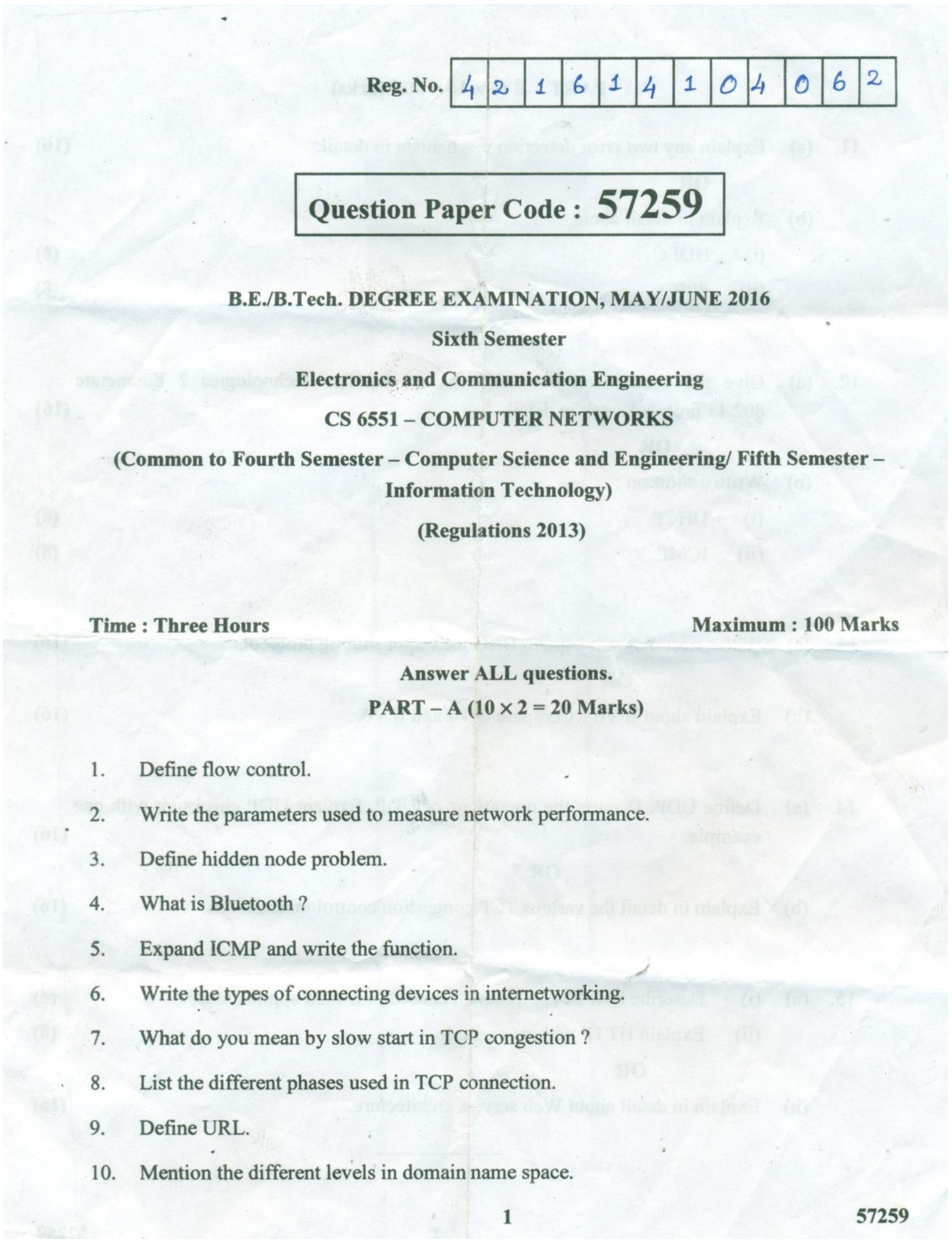
B.E/B.TechNov-Dec 2017

PART A

1. Write the use of HTTP (Q.No 35)
2. What do you meant by web services description language(WSDL)?(Q.No25)

PART B

1. i) Explain the functions of IMAP with a state transition diagram. (Q.No 4)
   ii) List and Explain the various HTTP request operations (Q.No 5)
2. i)List the element of network management and explain the operation of SNMP protocol in detail? (Q.No 7)
   ii) Discuss the function performed by of DNS . Give example. (Q.No6)



Reg. No. | 4 | 2 | 1 | 6 | 1 | 4 | 1 | 0 | 4 | 0 | 6 | 2 |

Question Paper Code : 57259

B.E./B.Tech. DEGREE EXAMINATION, MAY/JUNE 2016

Sixth Semester

Electronics and Communication Engineering

CS 6551 – COMPUTER NETWORKS

(Common to Fourth Semester – Computer Science and Engineering/ Fifth Semester – Information Technology)

(Regulations 2013)

Time : Three Hours **Maximum : 100 Marks**

Answer ALL questions.

PART – A (10 × 2 = 20 Marks)

1. Define flow control.
2. Write the parameters used to measure network performance.
3. Define hidden node problem.
4. What is Bluetooth ?
5. Expand ICMP and write the function.
6. Write the types of connecting devices in internetworking.
7. What do you mean by slow start in TCP congestion ?
8. List the different phases used in TCP connection.
9. Define URL.
10. Mention the different levels in domain name space.

1 **57259**

PART – B (5 × 16 = 80 Marks)

11. (a) Explain any two error detection mechanism in detail. **(16)**

OR

(b) Explain in detail about :

(i) HDLC **(8)**

(ii) PPP **(8)**

12. (a) Give the comparison between different wireless technologies ? Enumerate 802.11 protocol stack in detail. **(16)**

OR

(b) Write a short on :

(i) DHCP **(8)**

(ii) ICMP **(8)**

13. (a) With a neat diagram explain Distance vector routing protocol. **(16)**

OR

(b) Explain about IPV6 ? Compare IPV4 and IPV6. **(16)**

14. (a) Define UDP. Discuss the operations of UDP. Explain UDP checksum with one example. **(16)**

OR

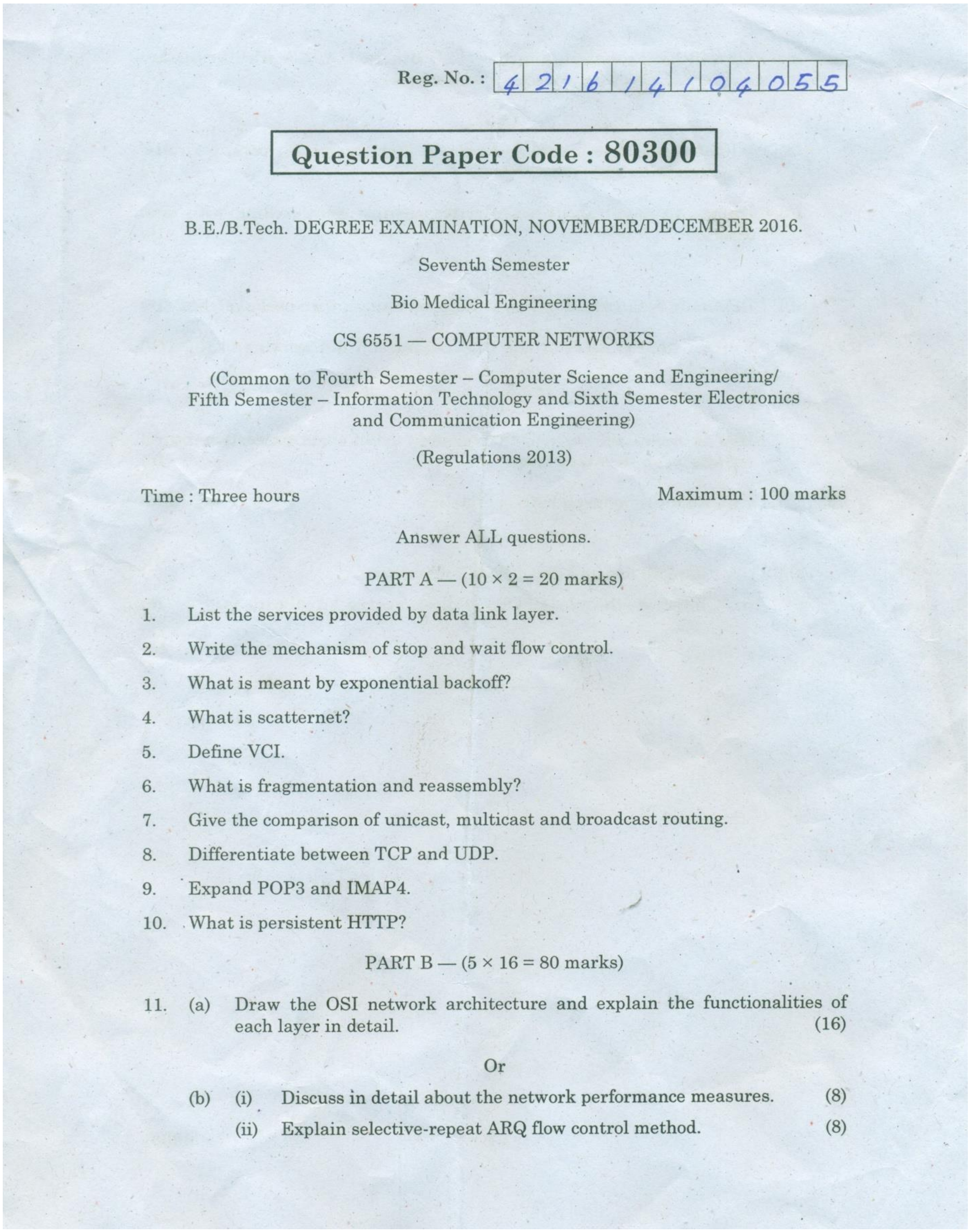
(b) Explain in detail the various TCP congestion control mechanisms. **(16)**

15. (a) (i) Describe how SMTP protocol is used in E-mail applications. **(8)**

(ii) Explain HTTP with an example. **(8)**

OR

(b) Explain in detail about Web service architecture. **(16)**



Reg. No. : 4 2 1 6 1 4 1 0 4 0 5 5

Question Paper Code : 80300

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2016.

Seventh Semester

Bio Medical Engineering

CS 6551 — COMPUTER NETWORKS

(Common to Fourth Semester – Computer Science and Engineering/ Fifth Semester – Information Technology and Sixth Semester Electronics and Communication Engineering)

(Regulations 2013)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. List the services provided by data link layer.
2. Write the mechanism of stop and wait flow control.
3. What is meant by exponential backoff?
4. What is scatternet?
5. Define VCI.
6. What is fragmentation and reassembly?
7. Give the comparison of unicast, multicast and broadcast routing.
8. Differentiate between TCP and UDP.
9. Expand POP3 and IMAP4.
10. What is persistent HTTP?

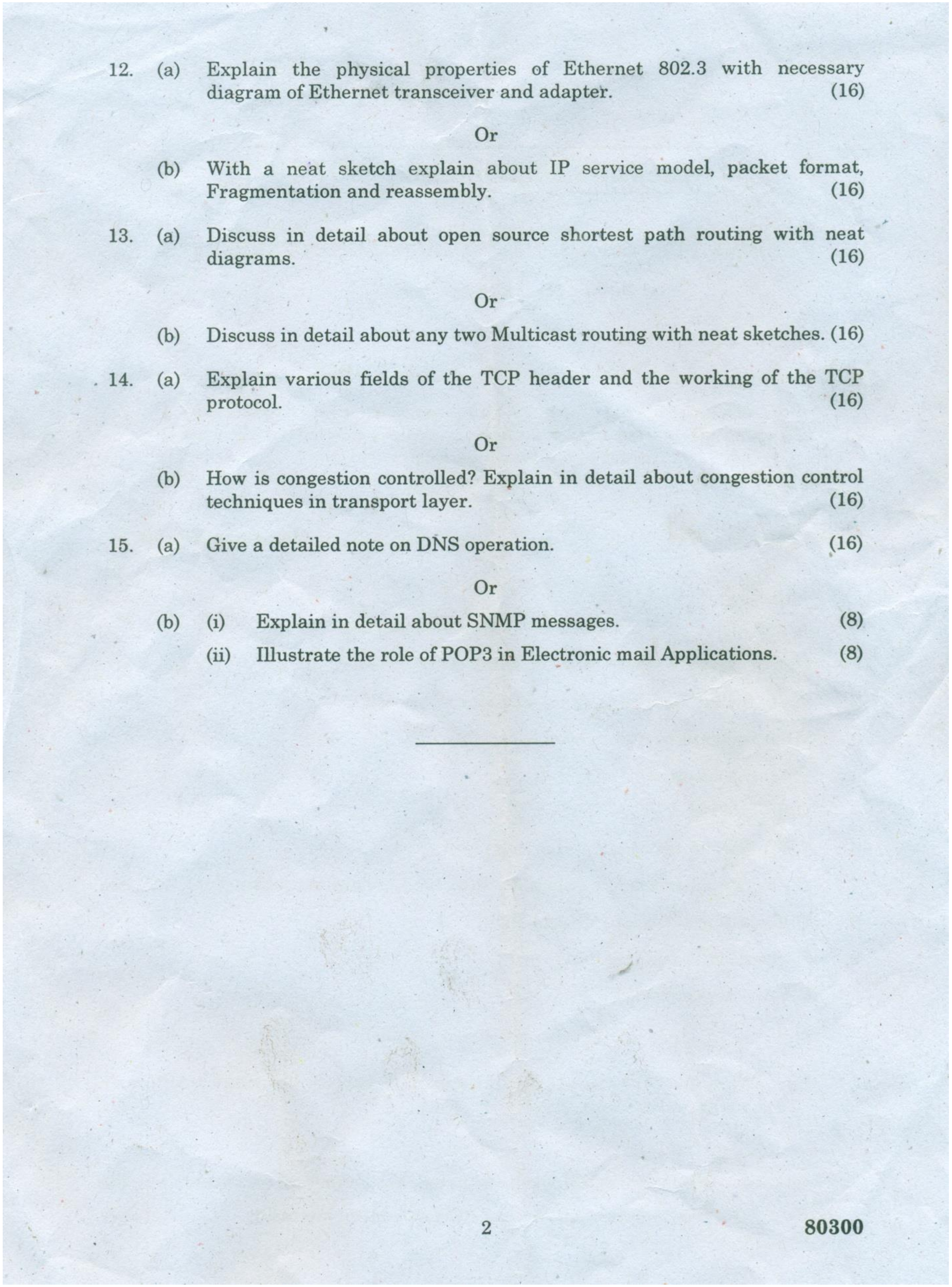
PART B — (5 × 16 = 80 marks)

11. (a) Draw the OSI network architecture and explain the functionalities of each layer in detail. (16)

Or

(b) (i) Discuss in detail about the network performance measures. (8)

(ii) Explain selective-repeat ARQ flow control method. (8)



12. (a) Explain the physical properties of Ethernet 802.3 with necessary diagram of Ethernet transceiver and adapter. (16)

Or

(b) With a neat sketch explain about IP service model, packet format, Fragmentation and reassembly. (16)

13. (a) Discuss in detail about open source shortest path routing with neat diagrams. (16)

Or

(b) Discuss in detail about any two Multicast routing with neat sketches. (16)

14. (a) Explain various fields of the TCP header and the working of the TCP protocol. (16)

Or

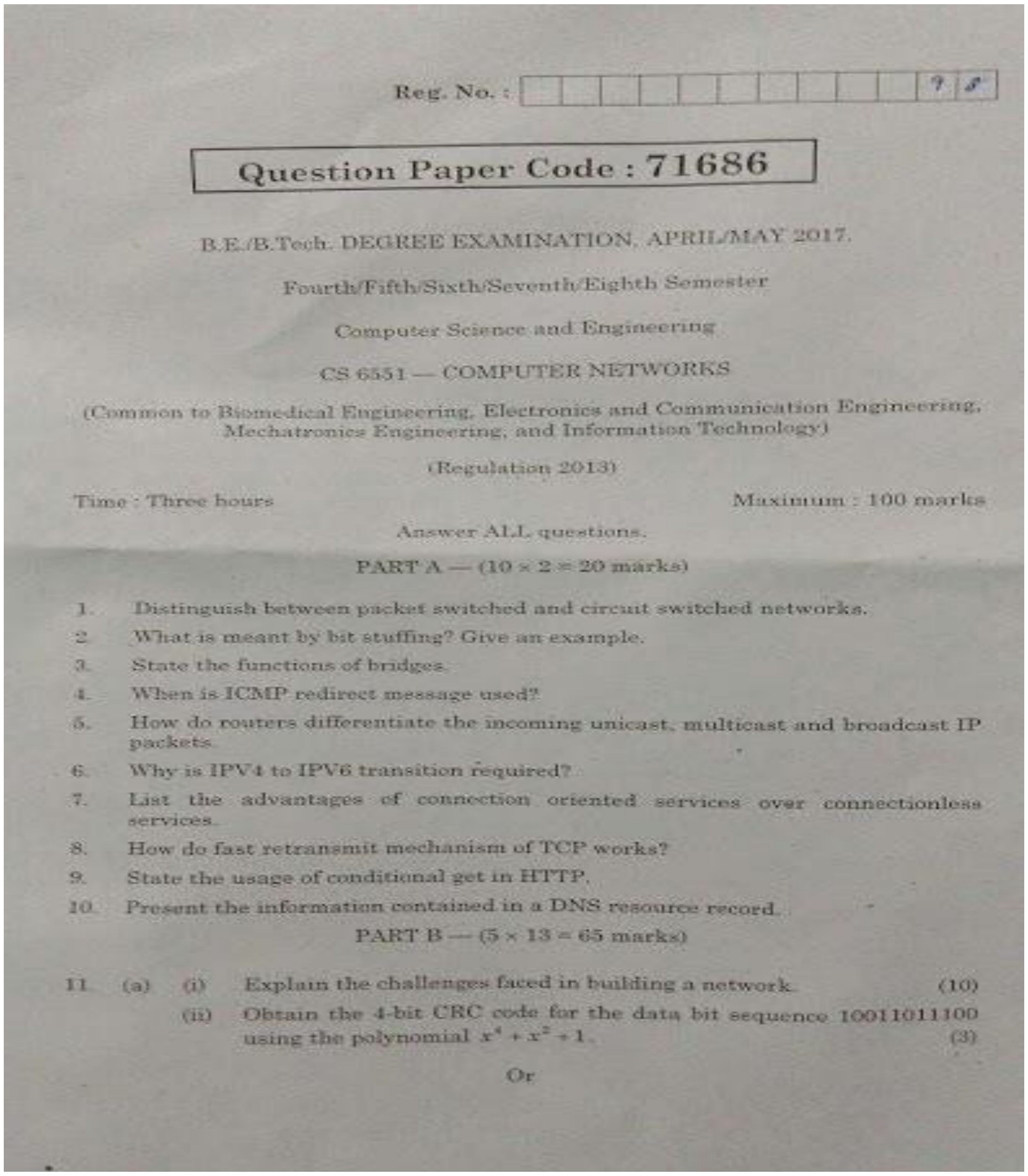
(b) How is congestion controlled? Explain in detail about congestion control techniques in transport layer. (16)

15. (a) Give a detailed note on DNS operation. (16)

Or

(b) (i) Explain in detail about SNMP messages. (8)

(ii) Illustrate the role of POP3 in Electronic mail Applications. (8)



Reg. No. : 7 8

Question Paper Code : 71686

B.E./B.Tech. DEGREE EXAMINATION, APRIL/MAY 2017.

Fourth/Fifth/Sixth/Seventh/Eighth Semester

Computer Science and Engineering

CS 6551 — COMPUTER NETWORKS

(Common to Biomedical Engineering, Electronics and Communication Engineering, Mechatronics Engineering, and Information Technology)

(Regulation 2013)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

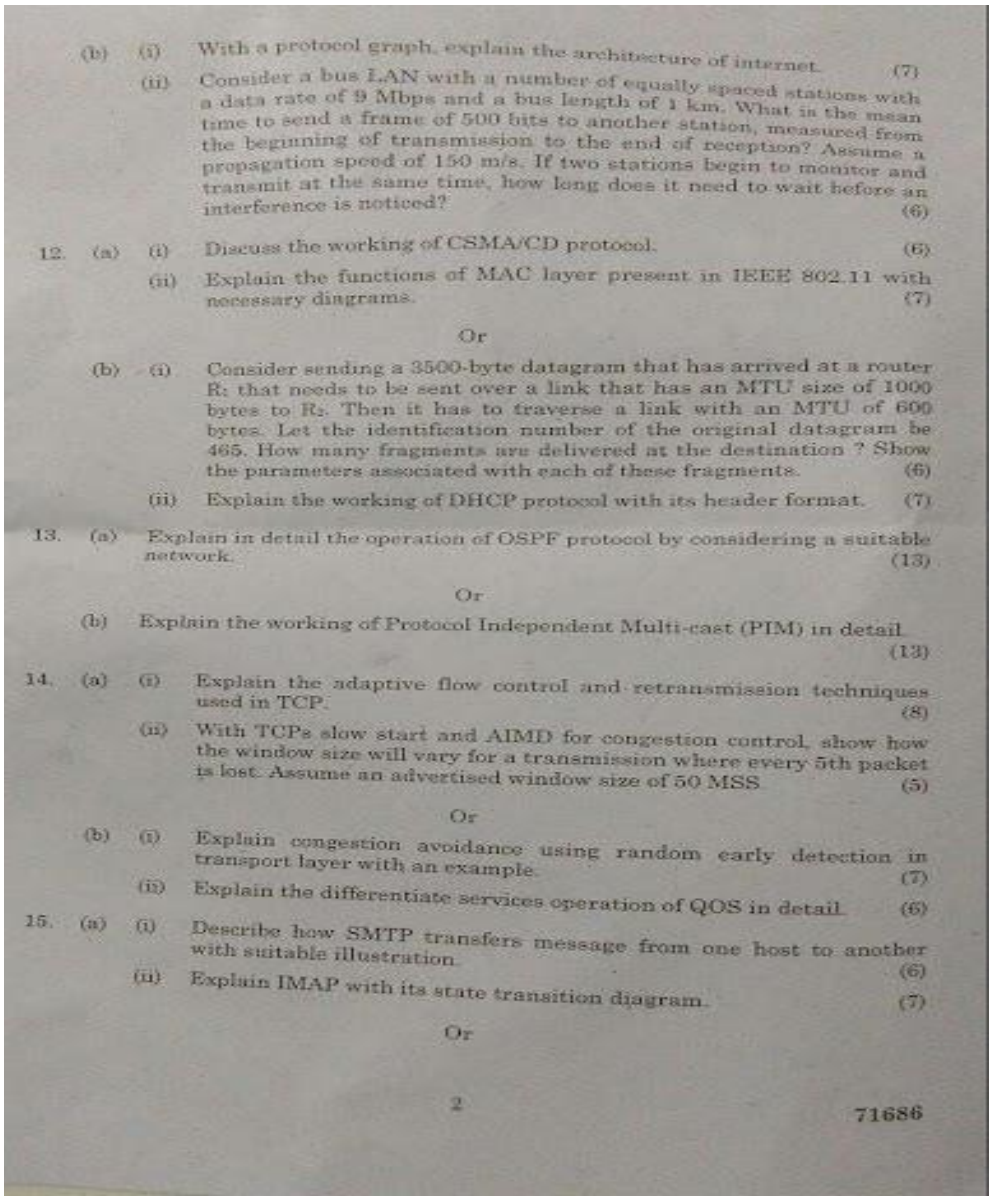
1. Distinguish between packet switched and circuit switched networks.
2. What is meant by bit stuffing? Give an example.
3. State the functions of bridges.
4. When is ICMP redirect message used?
5. How do routers differentiate the incoming unicast, multicast and broadcast IP packets.
6. Why is IPV4 to IPV6 transition required?
7. List the advantages of connection oriented services over connectionless services.
8. How do fast retransmit mechanism of TCP works?
9. State the usage of conditional get in HTTP.
10. Present the information contained in a DNS resource record.

PART B — (5 × 13 = 65 marks)

11. (a) (i) Explain the challenges faced in building a network. (10)

(ii) Obtain the 4-bit CRC code for the data bit sequence 10011011100 using the polynomial $x^4 + x^2 + 1$. (3)

Or



(b) (i) With a protocol graph, explain the architecture of internet. (7)

(ii) Consider a bus LAN with a number of equally spaced stations with a data rate of 9 Mbps and a bus length of 1 km. What is the mean time to send a frame of 500 bits to another station, measured from the beginning of transmission to the end of reception? Assume a propagation speed of 150 m/s. If two stations begin to monitor and transmit at the same time, how long does it need to wait before an interference is noticed? (6)

12. (a) (i) Discuss the working of CSMA/CD protocol. (6)

(ii) Explain the functions of MAC layer present in IEEE 802.11 with necessary diagrams. (7)

Or

(b) (i) Consider sending a 3500-byte datagram that has arrived at a router R_1 that needs to be sent over a link that has an MTU size of 1000 bytes to R_2. Then it has to traverse a link with an MTU of 600 bytes. Let the identification number of the original datagram be 465. How many fragments are delivered at the destination ? Show the parameters associated with each of these fragments. (6)

(ii) Explain the working of DHCP protocol with its header format. (7)

13. (a) Explain in detail the operation of OSPF protocol by considering a suitable network. (13)

Or

(b) Explain the working of Protocol Independent Multi-cast (PIM) in detail. (13)

14. (a) (i) Explain the adaptive flow control and retransmission techniques used in TCP. (8)

(ii) With TCPs slow start and AIMD for congestion control, show how the window size will vary for a transmission where every 5th packet is lost. Assume an advertised window size of 50 MSS. (5)

Or

(b) (i) Explain congestion avoidance using random early detection in transport layer with an example. (7)

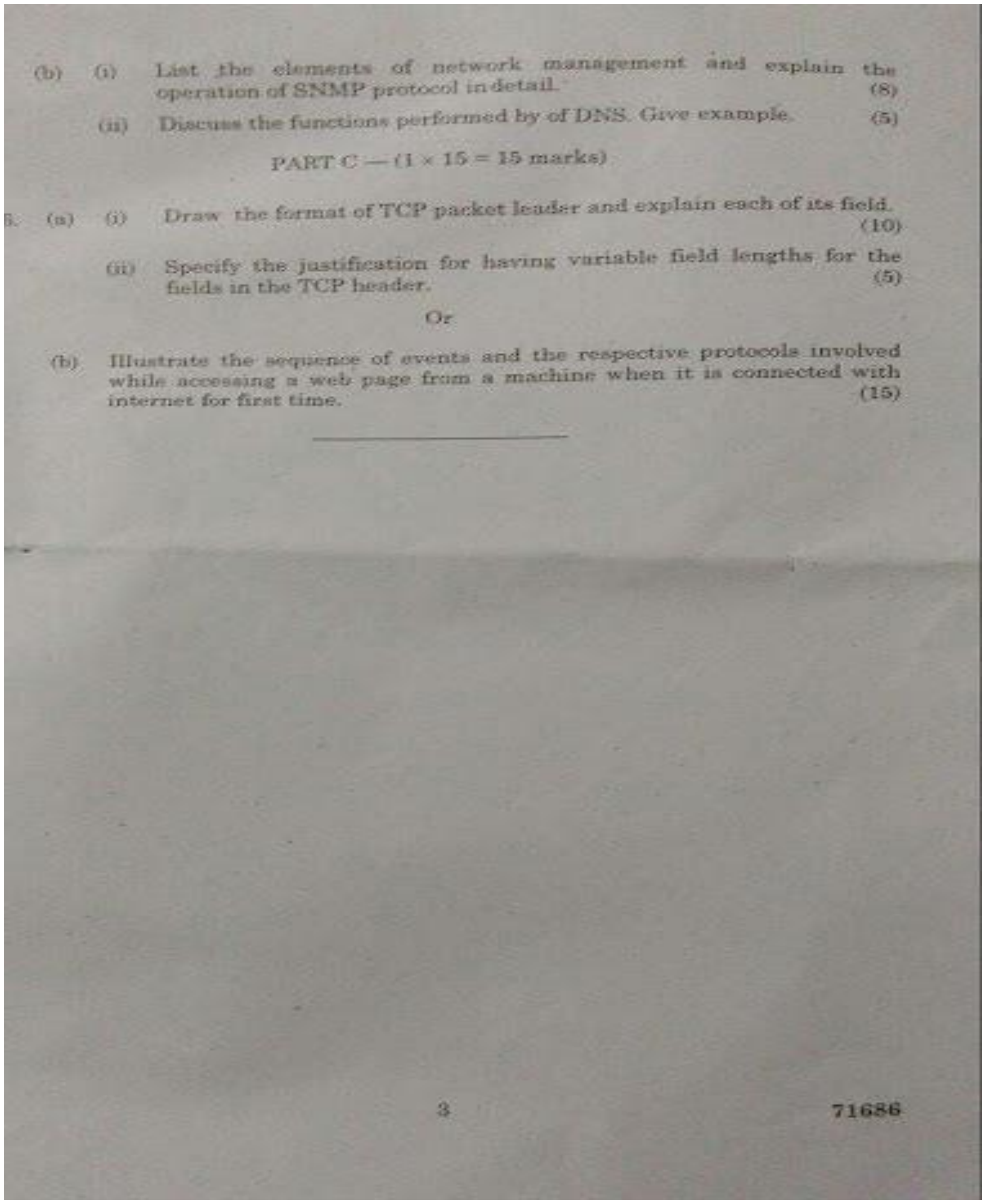
(ii) Explain the differentiate services operation of QOS in detail. (6)

15. (a) (i) Describe how SMTP transfers message from one host to another with suitable illustration. (6)

(ii) Explain IMAP with its state transition diagram. (7)

Or

2 71686



(b) (i) List the elements of network management and explain the operation of SNMP protocol in detail. (8)

(ii) Discuss the functions performed by of DNS. Give example. (5)

PART C — (1 × 15 = 15 marks)

6. (a) (i) Draw the format of TCP packet leader and explain each of its field. (10)

(ii) Specify the justification for having variable field lengths for the fields in the TCP header. (5)

Or

(b) Illustrate the sequence of events and the respective protocols involved while accessing a web page from a machine when it is connected with internet for first time. (15)

———————

3 71686

Reg. No. : | | | | | | | | | | | |

Question Paper Code : 50395

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2017

Fourth/Fifth/Sixth/Seventh/Eighth Semester

Computer Science and Engineering

CS6551 : COMPUTER NETWORKS

(Common to Biomedical Engineering, Electronics and Communication Engineering, Mechatronics Engineering, Information Technology)

(Regulations 2013)

Time : Three Hours Maximum : 100 Marks

Answer ALL questions

PART – A (10×2=20 Marks)

1. Define the terms : Bandwidth and Latency.
2. Compare Byte-oriented versus Bit-oriented protocol.
3. Show the Ethernet frame format.
4. Highlight the characteristics of datagram networks.
5. Differentiate between forwarding table and routing table.
6. What is Border Gateway Protocol (BGP) ?
7. Compare flow control versus congestion control.
8. What are the approaches used to provide a range of Quality of Service (QoS) ?
9. Write the use of Hyper Text Transfer Protocol (HTTP).
10. What do you mean by Web Services Description Language (WSDL) ?

PART – B (5×13=65 Marks)

11. a) With a neat sketch, explain the architecture of an OSI seven layer model. (13)

(OR)

b) Discuss the approaches used for error detection in networking. (13)

12. a) Explain the functions of Wi-Fi and Bluetooth in detail. (13)

(OR)

b) i) Explain the datagram forwarding in IP. (7)

ii) Show and explain the ARP packet format for mapping IP addresses into Ethernet addresses. (6)

13. a) With an example, explain the function of link state routing protocol. (13)

(OR)

b) Elaborate on multicast routing protocols. (13)

14. a) i) Draw a TCP state transition diagram for connection management. (7)

ii) Brief about approaches used for TCP congestion control. (6)

(OR)

b) Write a detailed note on congestion avoidance mechanisms used in TCP. (13)

15. a) i) Explain the function of Internet Message Access Protocol (IMAP) with a state diagram. (8)

ii) List and explain the various HTTP request operations. (5)

(OR)

b) i) What is Domain Name System (DNS) ? Explain. (8)

ii) Brief about the importance of Simple Network Management Protocol (SNMP). (5)

PART – C (1×15=15 Marks)

16. a) Outline the steps involved in building a computer network. Give the detailed description for each step. (15)

(OR)

b) For the network given in Figure 1, give global distance – vector tables when

i) Each node knows only the distances to its immediate neighbors. (5)

ii) Each node has reported the information it had in the preceding step to its immediate neighbors. (5)

iii) Step (ii) happens a second time. (5)

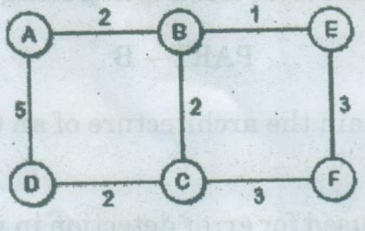


Figure 1

UNIT IV - TRANSPORT LAYER

Introduction – Transport Layer Protocols – Services – Port Numbers – User Datagram Protocol – Transmission Control Protocol – SCTP.

2 Marks

1) List the duties of Transport Layer (TL)

- Packetizing
- Connection Control
- Addressing
- Providing reliability

2) What is the difference between TCP & UDP? (NOV 2014 & 2016)

TCP	**UDP**
Connection Oriented Service	Connection less Service
Reliable	Not much reliable
Not suitable for multimedia, real time applications	used for multimedia and multicasting applications

3) What is socket? Define socket address.

Socket is the end point of a bi-directional communication flow across IP based network (Internet)

Socket address is the combination of an IP address (location of computer) and a port (application program process) into a single entity.

4) What is congestion? How to control congestion?

Congestion in network is the network is the situation in which an increase in data transmission results in a reduction in the throughput.

Throughput-amount of data passes through network congestion can be controlled using two techniques.

- Open-loop congestion control (prevention)
- Closed-loop congestion control(removal)

5) Define jitter

Jitter is the variation in delay for packets belonging to the same flow.

Example: 2ms delay for 1st packet
60ms delay for second packet.

6) What is the use of integrated services?

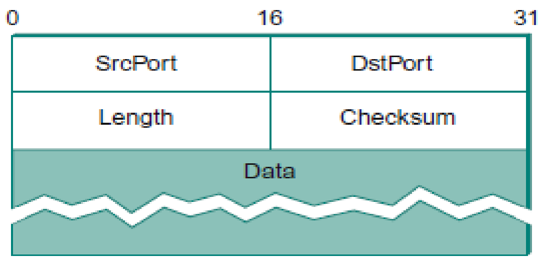
Integrated services (Instserv) is a followed based QoS model where the user creates a flow from source to direction and inform all the routers of the rource requirement.

7). Differentiate between delay and jitter.

Voice over IP (VoIP) is susceptible to network behaviors, referred to as delay and jitter, which can degrade the voice application to the point of being unacceptable to the average user. Delay is the time taken from point-to-point in a network. Delay can be measured in either one-way or round-trip delay.

Jitter is the VARIATION in delay over time from point-to-point. If the delay of transmissions varies too widely in a VoIP call, the call quality is greatly degraded. The amount of jitter tolerable on the network is affected by the depth of the jitter buffer on the network equipment in the voice path. The more jitter buffer available, the more the network can reduce the effects of jitter.

8) Draw UDP header format



9) What is traffic shaping?

Traffic shaping is a mechanism to control the amount and rate of traffic sent to the network.

10) What is the unit of data transfer in UDP and TCP?

In UDP, the Unit of data transfer is called datagram.
In TCP, Unit of data transfer is called segments.

11) List the timers used by TCP.

1) Retransmission timer
2) Persistence timer
3) Keep alive timer
4) Time waited timer

12) Define Sill window syndrome.

Sending less amount of Data (Ex. 1 byte) which is lesser than header size (20 bytes of TCP header +20 bytes of IP header) is called silly window syndrome. Here the capacity of network is used inefficiently.

13. Explain the main idea of UDP? Or Simple Demultiplexer

The basic idea is for a source process to send a message to a port and for the destination process to receive the message from a port.

14. What are the different fields in pseudo header?

- Protocol number
- Source IP address
- Destination IP addresses.

15. Define TCP? Or Reliable byte stream

TCP guarantees the reliable, in order delivery of a stream of bytes. It is a full-duplex protocol, meaning that each TCP connection supports a pair of byte streams, one flowing in each direction.

16. Define Congestion Control?

It involves preventing too much data from being injected into the network, thereby causing switches or links to become overloaded. Thus flow control is an end to an end issue, while congestion control is concerned with how hosts and networks interact.

17. State the two kinds of events trigger a state transition?

- A segment arrives from the peer.
- The local application process invokes an operation on TCP.

18. What is meant by segment?

At the sending and receiving end of the transmission, TCP divides long transmissions into smaller data units and packages each into a frame called a segment.

19. What is meant by segmentation?

When the size of the data unit received from the upper layer is too long for the network layer datagram or data link layer frame to handle, the transport protocol divides it into smaller usable blocks. The dividing process is called segmentation.

20. What is meant by Concatenation?

The size of the data unit belonging to single sessions are so small that several can fit together into a single datagram or frame, the transport protocol combines them into a single data unit. The combining process is called concatenation.

21. What is rate based design?

Rate- based design, in which the receiver tells the sender the rate-expressed in either bytes or packets per second – at which it is willing to accept incoming data.

22. Define Gateway.

A device used to connect two separate networks that use different communication protocols.

23. What are the two categories of QoS attributes?

The two main categories are,

- User Oriented
- Network Oriented

24. What is RED?

Random Early Detection in each router is programmed to monitor its own queue length and when it detects that congestion is imminent, to notify the source to adjust its congestion window.

25. What are the three events involved in the connection?

For security, the transport layer may create a connection between the two end ports. A connection is a single logical path between the source and destination that is associated with all packets in a message. Creating a connection involves three steps:

- Connection establishment
- Data transfer
- Connection release

26. Give the approaches to improve the QoS

Four common techniques are:

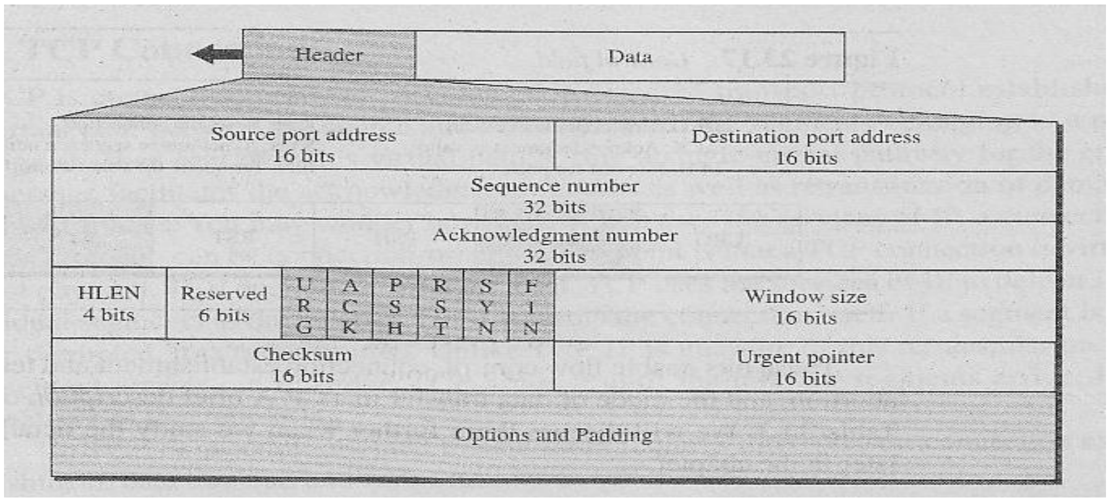
- Scheduling

- Traffic shaping
- Admission control
- Resource reservation

27. What is the difference between service point address, logical address and physical address? Service point addressing Logical addressing Physical addressing

Service point addressing	**Logical addressing**	**Physical addressing**
The transport layer header includes a type of address called a service point address or port address, which makes a data delivery from a specific process on one computer to a specific process on another computer.	If a packet passes the network boundary we need another addressing to differentiate the source and destination systems. The network layer adds a header, which indicate the logical address of the sender and receiver.	If the frames are to be distributed to different systems on the network, the data link layer adds the header, which defines the source machine' s address and the destination Machine' s address.

28. Draw TCP header format



29. How will the congestion be avoided?
The congestion may be avoided by two bits
<u>BECN</u> - Backward Explicit Congestion Notification
<u>FECN</u> - Forward Explicit Congestion Notification

30. What is the function of BECN BIT?
The BECN bit warns the sender of congestion in network. The sender can respond to this warning by simply reducing the data rate.

31. What is the function of FECN?
The FECN bit is used to warn the receiver of congestion in the network. The sender and receiver are communicating with each other and are using some types of flow control at a higher level.

32. What is meant by quality of service or QoS? (NOV 2014 & 2015)
The quality of service defines a set of attributes related to the performance of the connection. For each connection, the user can request a particular attribute each service class is associated with a set of attributes.

33. List out the user related attributes?
User related attributes are
SCR – Sustainable Cell Rate
PCR – Peak Cell Rate
MCR- Minimum Cell Rate
CVDT – Cell Variation Delay Tolerance

34. What are the networks related attributes?
The network related attributes are,
Cell loss ratio (CLR)
Cell transfer delay (CTD)
Cell delay variation (CDV)
Cell error ratio (CER)

35. Why is UDP pseudo header included in UDP checksum calculation? What is the effect of an invalid checksum at the receiving UDP?
The UDP checksum is performed over the entire payload, *and* the other fields in the header, *and* some fields from the IP header. A pseudo-header is constructed from the IP header in order to perform the calculation (which is done over this pseudo-header, the UDP header and the payload). The reason the pseudo-header is included is to catch packets that have been routed to the wrong IP address.
If the checksum validation is enabled and it detected an invalid checksum, features like packet reassembling won't be processed.

36. How can the effect of jitter be compensated? What type of application require for this compensation?
Jitter is an undesirable effect caused by the inherent tendencies of TCP/IP networks and components.
Jitter is defined as a variation in the delay of received packets. The sending side transmits \ packets in a continuous stream and spaces them evenly apart. Because of network congestion, improper queuing, or configuration errors, the delay between packets can vary instead of remaining constant. This variation causes problems for audio playback at the receiving end. Playback may experience gaps while waiting for the arrival of variable delayed packets.

When a router receives an audio stream for VoIP, it must compensate for any jitter that it detects. The playout delay buffer mechanism handles this function. Playout delay is the amount of time that elapses between the time a voice packet is received at the jitter buffer on the DSP and the time a voice packet is played out to the codec.

The playout delay buffer must buffer these packets and then play them out in a steady Stream to the DSPs. The DSPs then convert the packets back into an analog audio stream. The play out delay buffer is also referred to as the dejitter buffer.

37. What is meant by PORT or MAILBOX related with UDP?
Form of address used to identify the target process:

Process can directly identify each other with an OS-assigned process ID(pid) More commonly-processes indirectly identify each other using a port or mailbox Source sends a message to a port and destination receives the message from the port UDP port is 16 bits, so there are 64K possible ports- not enough for all Internet hosts Process is identified as a port on a particular host – a (port, host) pair.
To send a message the process learns the port in the following way:
A client initiates a message exchange with a server process. The server knows the client's port (contained in message header and can reply to it. Server accepts messages at a well known port.
Examples: DNS at port 53, mail at port 25

38. List out the various features of sliding window protocol.
The key feature of the sliding window protocol is that it permits pipelined communication. In contrast, with a simple stop-and-wait protocol, the sender waits for an acknowledgment after transmitting every frame. As a result, there is at most a single outstanding frame on the channel at any given time, which may be far less than the channel's capacity. For maximum throughput, the amount of data in transit at any given time should be equal to (channel bandwidth) X (channel delay).

39. What is the function of a router?
- Connect network segment together
- Router forwards the packet to the right path

40. What is the advantage of using UDP over TCP?
- UDP can send data in a faster way than TCP
- UDP is suitable for sending multicasting and multimedia applications

41. What is the difference between congestion control and flow control? Nov 2015 ,Nov/Dec 2017
Congestion control
It involves preventing too much data from being injected into the network, thereby causing switches or links to become overloaded. Thus flow control is an end to an end issue, while congestion control is concerned with how hosts and networks interact.

Flow control
The amount of data flowed from source to destination should be restricted. The source can send one byte at a time, but it will take long time to transmit n bytes

42. List the mechanisms used in TCP congestion control mechanism
- Additive Increase/Multiplicative Decrease
- Slow Start
- Fast Retransmit and Fast Recovery

43. List the mechanisms used in TCP congestion avoidance

- DEC bit
- RED(Random Early Detection)
- Source-based Congestion Avoidance

44. Define DEC bit

Each router monitors the load it is experiencing and explicitly notifies the end nodes when congestion is about to occur. This notification is implemented by setting a binary congestion bit in the packets that flow through the router, hence the name *DEC bit*.

45. What is meant by Source-Based Congestion Avoidance?

The general idea of these techniques is to watch for some sign from the network that some router's queue is building up and that congestion will happen soon if nothing is done about it

46. List the approaches to QoS support or what are the approaches used to provide range of Quality of services Nov/Dec 2017

***Fine-grained* approaches**, which provide QoS to individual applications or flows
***Coarse-grained* approaches,** which provide QoS to large classes of data or aggregated traffic

47. List the types of application requirements in QoS

- Real-time
- Non-real-time

48. List some of the Quality of service parameters of transport layer (May 2015)

- Reliability
- Delay
- Jitter
- Bandwidth

49. How does transport layer perform duplication control? (May 2015)
Duplication can be controlled by the use of sequence number & acknowledgment number

50. What do you mean by slow start in TCP congestion? (May 2016)

The sender starts with a very slow rate of transmission but increases the rate rapidly to reach a threshold

51. List the different phases used in TCP connection

- ✓ Connection establishment and Data transfer
- ✓ Connection termination

52. List out the advantages of connection oriented services over connectionless services. (APR 2017)

Advantage of connection oriented:

(i) In connection oriented virtual circuit,buffers can be reversed in advance
(ii) Sequencing can be guaranteed
(iii)Short-headers can be used
(iv)Troubles caused by delayed duplicate packets can be avoided

Advantage of connectionless:

(i) It can be used over subnets that do not use virtual circuit inside

(ii) No circuit setup time required
(iii) It is higher robust in the face of router failure
(iv) It is best for connectionless transport protocol because it does not impose unnecessary overhead

53. How do fast retransmit mechanism of TCP works. (APR 2017)

The transmission rate will be increases with slow start algorithm until either a loss is detected, or the receiver's advertised window (rwnd) is limiting factor, or when the slow start threshold (ssthresh) is reached.

16 MARKS

1. Discuss in detail about transport layer & its services

INTRODUCTION

The transport layer is located between the application layer and the network layer. It provides a process-to-process communication between two application layers, one at the local host and the other at the remote host. Communication is provided using a logical connection, which means that the two application layers, which can be located in different parts of the globe, assume that there is an imaginary direct connection through which they can send and receive messages.

Transport-Layer Services

Each protocol provides a different type of service and should be used appropriately.

UDP

UDP is an unreliable connectionless transport-layer protocol used for its simplicity and efficiency in applications where error control can be provided by the application-layer process.

TCP

TCP is a reliable connection-oriented protocol that can be used in any application where reliability is important.

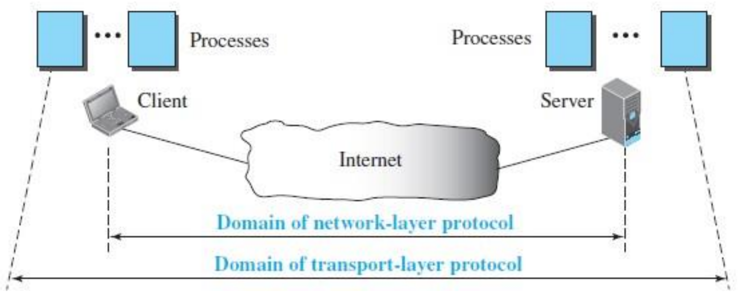
SCTP

SCTP is a new transport-layer protocol that combines the features of UDP and TCP.

Process-to-Process Communication

The first duty of a transport-layer protocol is to provide **process-to-process communication.** A process is an application-layer entity (running program) that uses the services of the transport layer. A transport-layer protocol is responsible for delivery of the message to the appropriate process

Figure 23.2 *Network layer versus transport layer*


Processes
Processes
Client
Server
Internet
Domain of network-layer protocol
Domain of transport-layer protocol

Addressing: Port Numbers

Although there are a few ways to achieve process-to-process communication, the most common is through the **client-server paradigm**. A process on the local host, called a *client*, needs services from a process usually on the remote host, called a *server*.

The local host and the remote host are defined using IP addresses. To define the processes, we need second identifiers, called ***port numbers.*** The client program defines itself with a port number, called the ***ephemeral port number.*** The word *ephemeral* means "short-lived" and is used because the life of a client is normally short. An ephemeral port number is recommended to be greater than 1023 for some client/server programs to work properly.

Figure 23.3 *Port numbers*

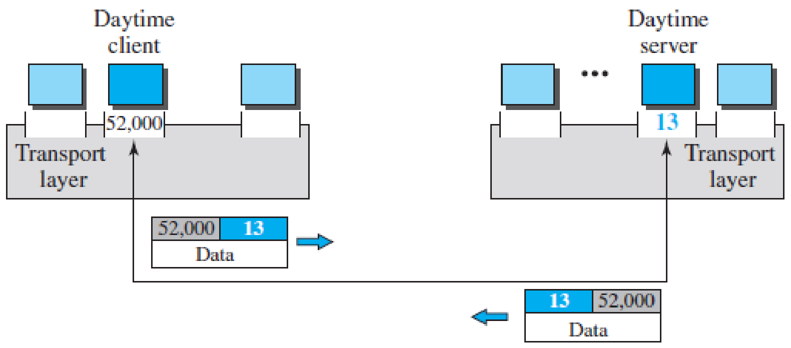


Table 24.1 *Some well-known ports used with UDP and TCP*

Port	*Protocol*	*UDP*	*TCP*	*SCTP*	*Description*
7	Echo	√	√	√	Echoes back a received datagram
9	Discard	√	√	√	Discards any datagram that is received
11	Users	√	√	√	Active users
13	Daytime	√	√	√	Returns the date and the time
17	Quote	√	√	√	Returns a quote of the day
19	Chargen	√	√	√	Returns a string of characters
20	FTP-data		√	√	File Transfer Protocol
21	FTP-21		√	√	File Transfer Protocol
23	TELNET		√	√	Terminal Network
25	SMTP		√	√	Simple Mail Transfer Protocol
53	DNS	√	√	√	Domain Name Service
67	DHCP	√	√	√	Dynamic Host Configuration Protocol
69	TFTP	√	√	√	Trivial File Transfer Protocol
80	HTTP		√	√	HyperText Transfer Protocol
111	RPC	√	√	√	Remote Procedure Call
123	NTP	√	√	√	Network Time Protocol
161	SNMP-server	√			Simple Network Management Protocol
162	SNMP-client	√			Simple Network Management Protocol

ICANN Ranges

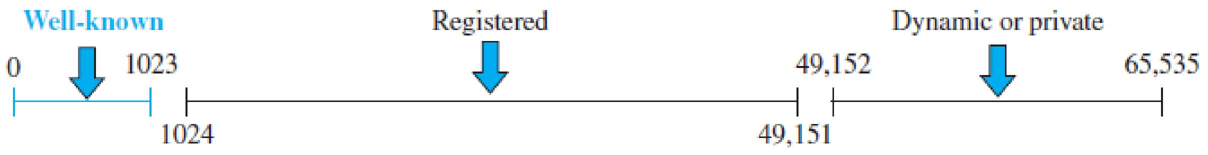
ICANN has divided the port numbers into three ranges: well-known, registered, and dynamic (or private), as shown in Figure 23.5

❑ ***Well-known ports.*** The ports ranging from 0 to 1023 are assigned and controlled by ICANN. These are the well-known ports.

❑ ***Registered ports.*** The ports ranging from 1024 to 49,151 are not assigned or controlled by ICANN. They can only be registered with ICANN to prevent duplication.

❑ ***Dynamic ports.*** The ports ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used as temporary or private port numbers.

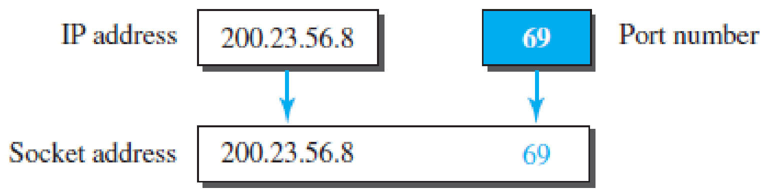
Figure 23.5 *ICANN ranges*



Socket Addresses

A transport-layer protocol in the TCP suite needs both the IP address and the port number, at each end, to make a connection. The combination of an IP address and a port number is called a ***socket address.*** The client socket address defines the client process uniquely just as the server socket address defines the server process uniquely

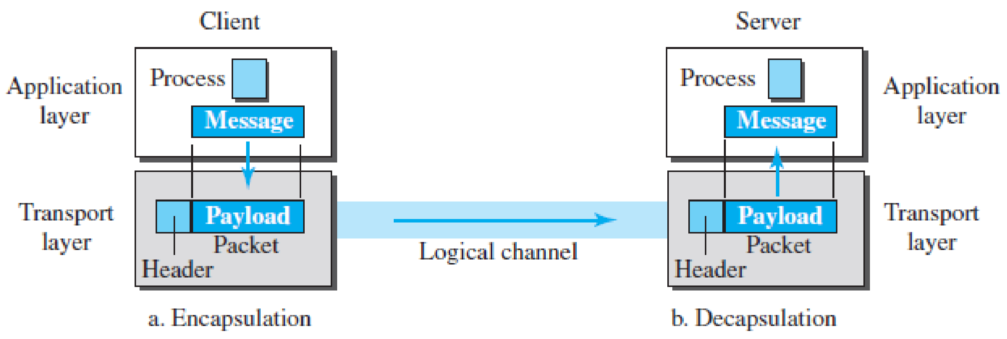
Figure 23.6 *Socket address*



Encapsulation and Decapsulation

To send a message from one process to another, the transport-layer protocol encapsulates and decapsulates messages.

Figure 23.7 *Encapsulation and decapsulation*



Multiplexing and Demultiplexing

Whenever an entity accepts items from more than one source, this is referred to as ***multiplexing*** (many to one); whenever an entity delivers items to more than one source, this is referred to as ***demultiplexing*** (one to many). The transport layer at the source performs multiplexing; the transport layer at the destination performs demultiplexing

Flow Control

Whenever an entity produces items and another entity consumes them, there should be a balance between production and consumption rates

Error Control

Error control at the transport layer is responsible for

1. Detecting and discarding corrupted packets.
2. Keeping track of lost and discarded packets and resending them.
3. Recognizing duplicate packets and discarding them.
4. Buffering out-of-order packets until the missing packets arrive.

Congestion Control

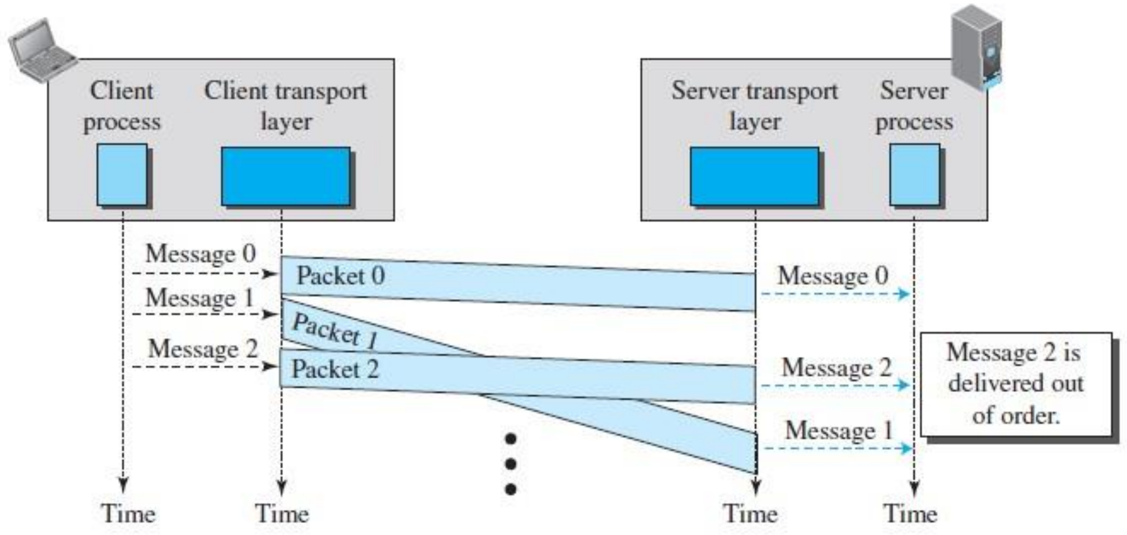
Congestion control refers to the mechanisms and techniques that control the congestion and keep the load below the capacity

Connectionless and Connection-Oriented Protocols

Connectionless Service

In a connectionless service, the source process (application program) needs to divide it message into chunks of data of the size acceptable by the transport layer and deliver them to the transport layer one by one

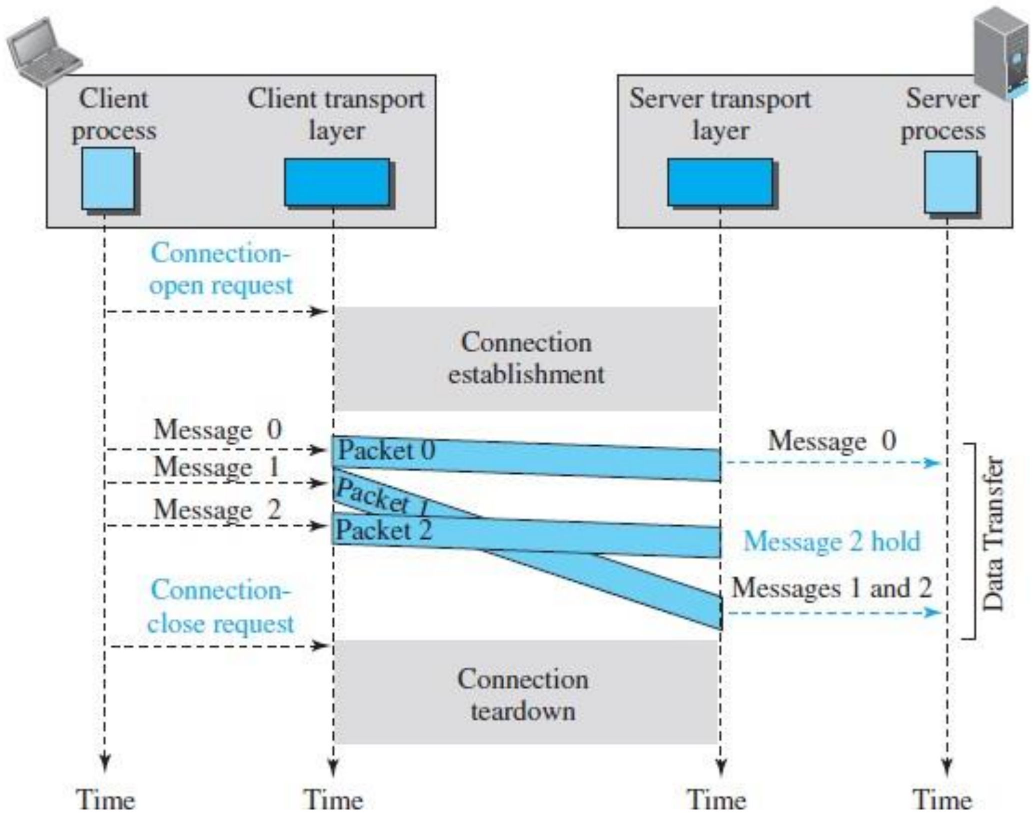
Figure 23.14 *Connectionless service*



Connection-Oriented Service

In a connection-oriented service, the client and the server first need to establish a logical connection between themselves. The data exchange can only happen after the connection establishment. After data exchange, the connection needs to be torn down

Figure 23.15 *Connection-oriented service*



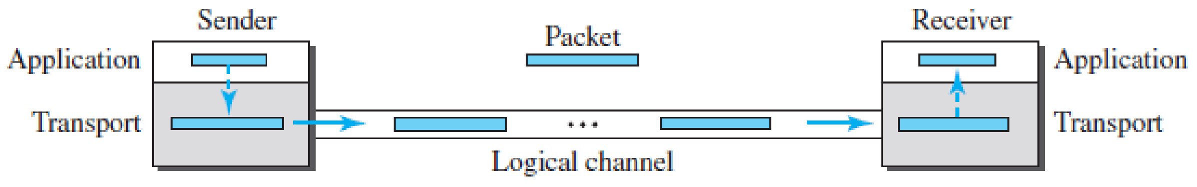
2. List & explain the protocols used in transport layer.

- Simple Protocol
- Stop-and-Wait Protocol
- Go-Back-*N* Protocol (GBN)
- Selective-Repeat Protocol
- Bidirectional Protocols: Piggybacking

Simple Protocol

Our first protocol is a simple connectionless protocol with neither flow nor error control. We assume that the receiver can immediately handle any packet it receives. In other words, the receiver can never be overwhelmed with incoming packets.

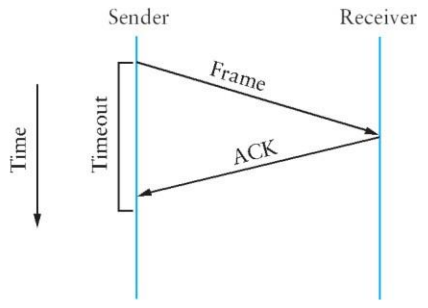
Figure 23.17 *Simple protocol*

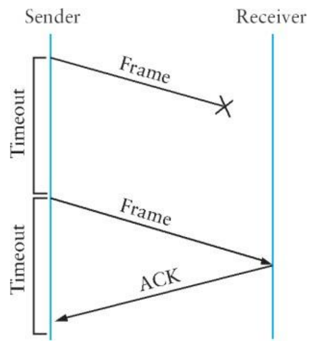


Stop and Wait Protocol

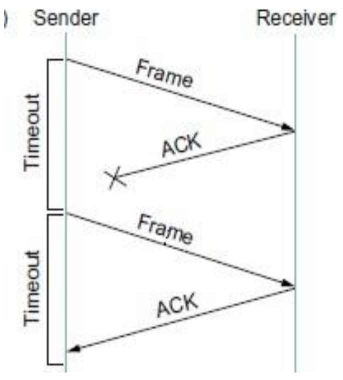
➢ After transmitting one frame, <u>the sender waits for an acknowledgment before transmitting the next frame.</u>

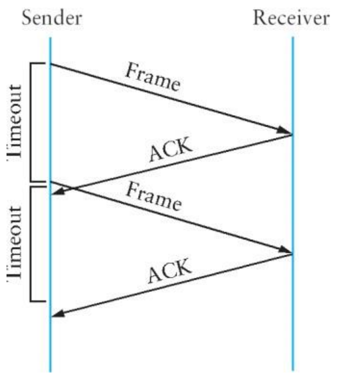
➢ If the acknowledgment does not arrive after a certain period of time, the sender times out and retransmit the original frame.





a) The ACK is received before the timer expires **b) The original frame is lost**





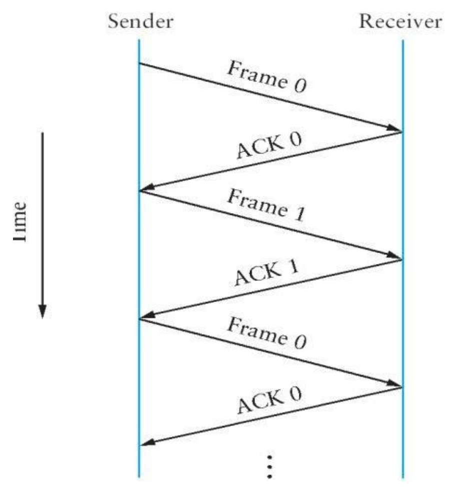
c) The ACK is lost **d)The timeout fires too soon**

Fig: illustrates four different scenarios that result from this basic algorithm. The sending side is represented on the left, the receiving side is depicted on the right, and time flows from top to bottom.

➢ In Fig (a) ACK is received before the timer expires, (b) and (c) show the situation in which the original frame and the ACK, respectively, are lost, and (d) shows the situation in which the timeout fires too soon..

➢ Suppose the sender sends a frame and the receiver acknowledges it, but the acknowledgment is either lost or delayed in arriving. This situation is in (c) and (d). In both cases, the sender times out and retransmit the original frame, but the receiver will think that it is the next frame, since it correctly received and acknowledged the first frame.

➢ This makes the receiver to receive the duplicate copies. To avoid this two sequence numbers (0 and 1) must be used alternatively.



- The main drawback of the stop-and-wait algorithm is that it allows the sender have only one outstanding frame on the link at a time.

Sliding Window Protocol

- The sender can transmit several frames before needing an acknowledgement.
- Frames can be sent one right after another meaning that the link can carry several frames at once and it s capacity can be used efficiently.
- The receiver acknowledges only some of the frames, using a single ACK to confirm the receipt of multiple data frames
- Sliding Window refers to imaginary boxes at both the sender and the receiver.
- Window can hold frames at either end and provides the upper limit on the number of frames that can be transmitted before requiring an acknowledgement.
- Frames are numbered modulo-n which means they are numbered from o to n-1
- For eg. If n=8 the frames are numbered 0,1,2,3,4,5,6,7. i.e the size of the window is n -1.
- When the receiver sends ACK it includes the number of the next frame it expects to receive.
- When the sender sees an ACK with the number 5, it knows that all frames up through number 4 have been received.

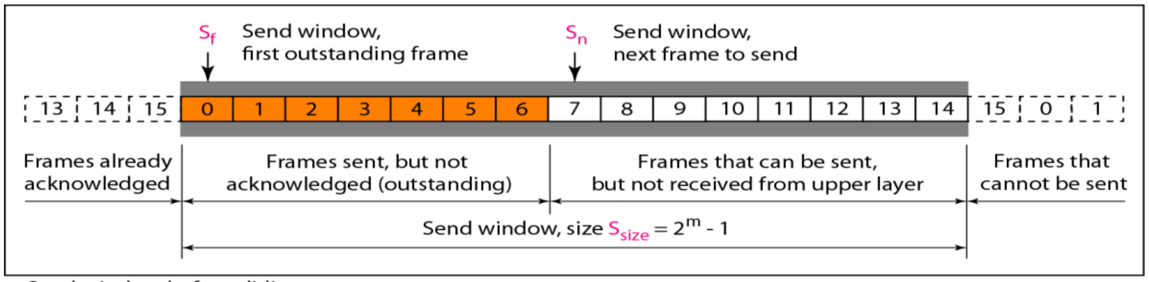
There are two methods to retransmit the lost frames

- GO-Back N
- Selective Repeat

Go – Back N Protocol

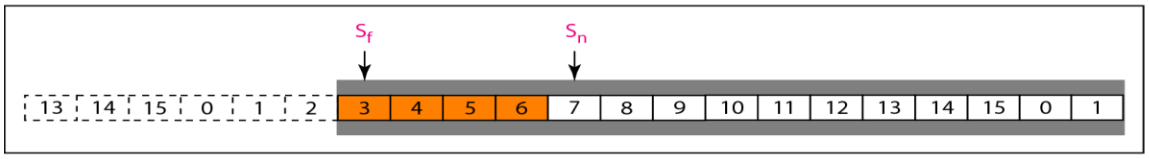
Sender Window

- At the beginning of transmission, the sender window contains n-1 frames. As frames are sent out, the left boundary of the window moves inward, shrinking the size of the window
- If size of window is W if three frames have been transmitted since the last acknowledgement then the number of frames left in the window is w -3.
- Once an ACK arrives, the window expands to allow in a number of new frames equal to the number of frames acknowledged by that ACK.



S_f Send window, first outstanding frame
S_n Send window, next frame to send
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already acknowledged
Frames sent, but not acknowledged (outstanding)
Frames that can be sent, but not received from upper layer
Frames that cannot be sent
Send window, size $S_{size} = 2^m - 1$

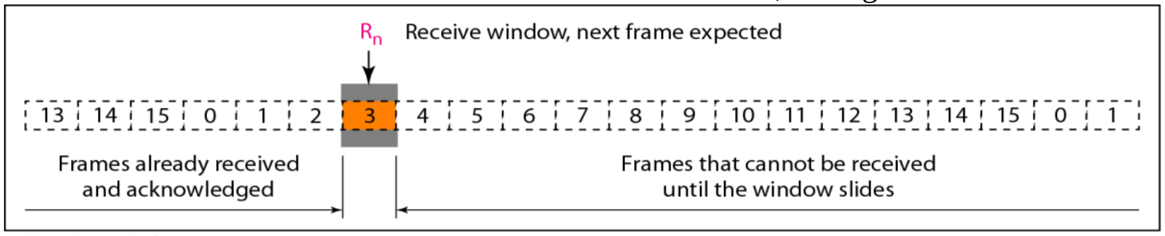
a. Send window before sliding



b. Send window after sliding

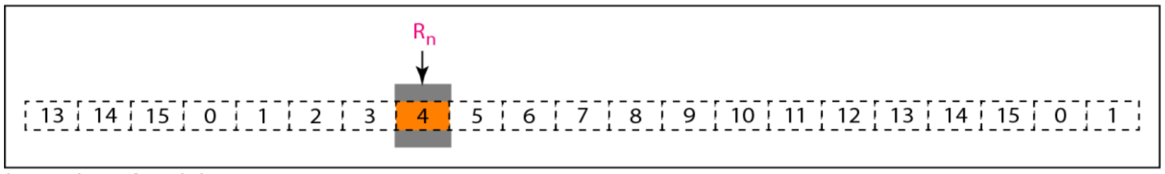
Receiver Window

- ➢ The receive window is an abstract concept defining an imaginary box of size 1 with one single variable Rn.
- ➢ The window slides when a correct frame has arrived, sliding occurs one slot at a time.



R_n Receive window, next frame expected
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already received and acknowledged
Frames that cannot be received until the window slides

a. Receive window

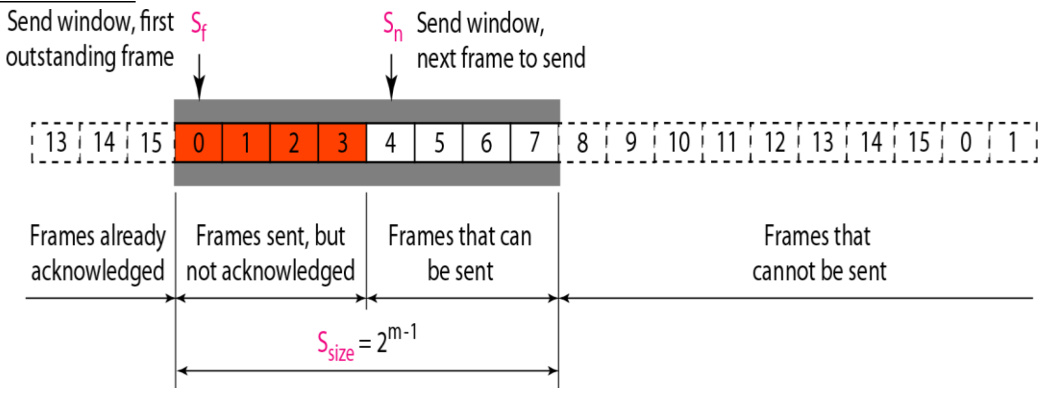


b. Window after sliding

When the timer expires, the sender resends all outstanding frames. For example, suppose the sender has already sent frame 6, but the timer for frame 3 expires. This means that frame 3 has not been acknowledged; the sender goes back and sends frames 3, 4,5, and 6 again. That is why the protocol is called *Go-Back-N*.

Selective Repeat Protocol

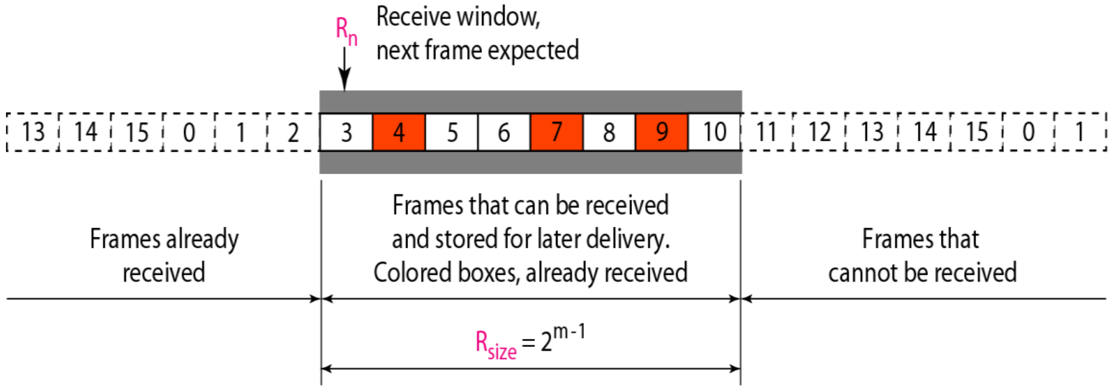
Sender Window



Send window, first outstanding frame S_f
S_n Send window, next frame to send
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already acknowledged
Frames sent, but not acknowledged
Frames that can be sent
Frames that cannot be sent
$S_{size} = 2^{m-1}$

Receiver window

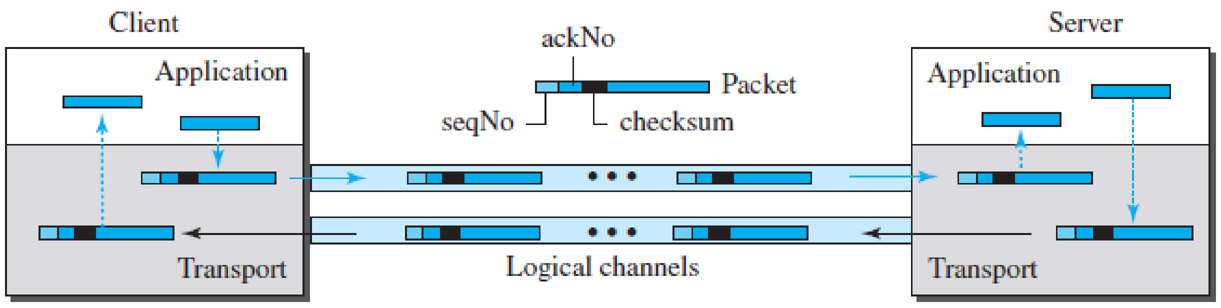
- The Selective Repeat Protocol allows as many frames as the size of the receive window to arrive out of order and be kept until there is a set of in-order frames to be delivered to the network layer.
- Because the sizes of the send window and receive window are the same, all the frames in the send frame can arrive out of order and be stored until they can be delivered.
- If any frame lost, sender has to retransmit only that lost frames.


R_n
Receive window, next frame expected
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already received
Frames that can be received and stored for later delivery. Colored boxes, already received
Frames that cannot be received
$R_{size} = 2^{m-1}$

Bidirectional Protocols: Piggybacking

The four protocols we discussed earlier in this section are all unidirectional: data packets flow in only one direction and acknowledgments travel in the other direction. In real life, data packets are normally flowing in both directions: from client to server and from server to client. This means that acknowledgments also need to flow in both directions. A technique called **piggybacking** is used to improve the efficiency of the bidirectional protocols. When a packet is carrying data from A to B, it can also carry acknowledgment feedback about arrived packets from B; when a packet is carrying data from B to A, it can also carry acknowledgment feedback about the arrived packets from A.

Figure 23.37 *Design of piggybacking in Go-Back-N*



3. Explain the working of USER DATAGRAM PROTOCOL (UDP) or Simple Demultiplexer (May 2016)

The UDP is called a connection less, unreliable transport protocol. The purpose of UDP is to break up a stream of data into datagram, add a source and destination port information, a length and a checksum. It is the receiving application's responsibility to detect and recover lost or damaged packets, as UDP doesn't take care of this.

Advantages:

- It is a very simple protocol using a minimum of overhead.
- If a process wants to send a small message and doesn’t care much about reliability, it can use UDP.
- It is a convenient protocol for multimedia and multicasting applications.
- Sending a small message by using UDP takes much less interaction between the sender and receiver than using TCP.

Basic Properties of UDP (services):

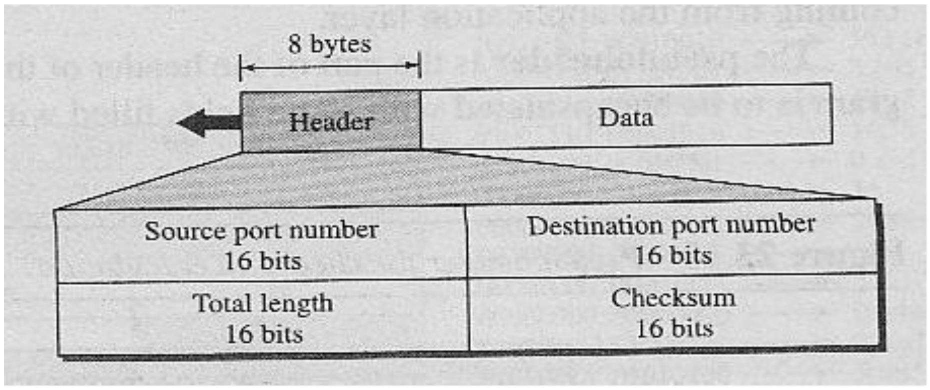
- UDP is a connectionless transport protocol.
  - A UDP application sends messages without establishing and then closing a connection.
  - UDP has a smaller overhead then TCP, especially when the total size of the messages is small.
- UDP does not guarantee reliable data delivery.
  - UDP messages can be lost or duplicated, or they may arrive out of order; and they can arrive faster than the receiver can process them.
  - The application programmers using UDP have to consider and tackle these issues themselves.
  - Not buffered -- UDP accepts data and transmits immediately (no buffering before transmission)
  - Full duplex -- concurrent transfers can take place in both directions
- UDP has no mechanism for flow control.
- Multiplexing and Demultiplexing
  - This is many to one relationship used in sender side.
  - The protocol accepts messages from different processes, differentiated by their assigned port numbers.
  - Demultiplexing is one to many relationship used in receiver side.
- Encapsulation and Decapsulation
  - To send a message from one processes to another, the UDP protocol encapsulate and decapsulate messages in an IP datagram
  - Encapsulate each UDP message in an IP datagram, and use IP to deliver this datagram across the internet.

UDP Message Format

User Datagram:

- UDP packets called user datagram which has a fixed size header of 8 bytes.

Format of User Datagram:



User datagram has the following fields.

- **Source Port Number**
  The source port number used by the processes running on the <u>source host</u> (local computer). It is 16 bits long, which means that the port number can range from 0 to 65535. If the source host is the client, the port number requested by the processes and chosen by the UDP software running on the source host.

- **Destination Port Number**
  This is the port number used by the processes running on the <u>destination host</u>. It is also 16 bits long. The Destination port is usually a 'well known port number' such as 69 for trivial file transfer protocol, or 53 for DNS. These port numbers allow the remote machine to recognize a request for a particular type of service. If the destination host is a client, the server copies the port number it has received in the request packet.

- **Length**
  This is a 16 bits field that defines the <u>total length of the user data gram</u>, header plus data. The 16 bits can defined a total length of 0 to 65535 bytes.

- **Checksum**
  This field is used to <u>detect errors</u> over the entire user datagram. The calculation of checksum and its inclusion in the user datagram are optional.

<u>Process communication in UDP</u>

The next issue is how a process learns the port for the process to which it wants to send the message.

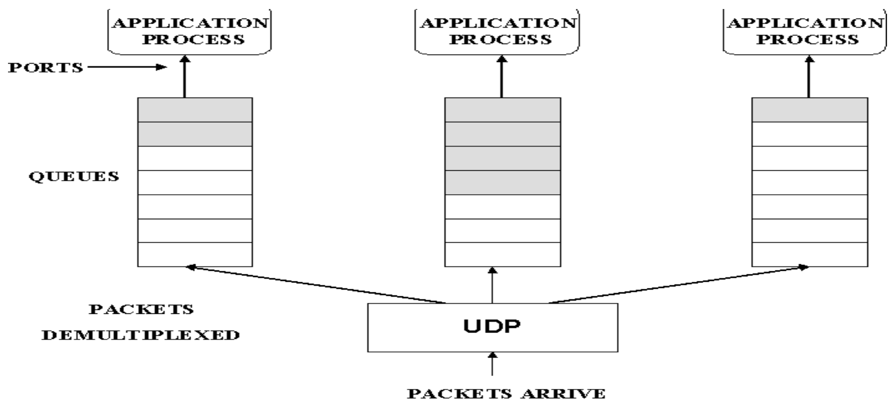
<u>Typically a client process initiates a message exchange with a server process. Once a client has contacted a server, the server knows the client's port (it was contained in the message header) and can reply to it.</u>

The real problem, therefore, is how the client learns the server's port in the first place. A common approach is for the server to accept messages at a well known port.

That is, each <u>server receives its messages at some fixed port that is widely published, much like the emergency telephone service available at the well-known number 911</u>.

In the Internet, for example, the domain name server (DNS) receives messages at well-known port 53 on each host, the mail service listens of messages at port 25,and the Unix talk program accepts messages at well known port 517,and so on.

UDP Message queue



This mapping is published periodically in an RFC and is available on the most Unix systems in file /etc/services. Sometimes a well-known port to agree on some other port that they will use for subsequent communication leaving the well-known port free for other clients.

A port is purely an abstraction. Exactly how it is implemented differs from system to system, or more precisely, from OS to OS.

For example, the socket API is an example implementation of ports. Typically, a port is implemented by a message queue.

When a message arrives, the protocol l(eg.UDP) appends the message to the end of the queue. Should the queue be full, the message is discarded.

There is no flow-control mechanism that tells the sender to slow down. When an application process wants to receive a message, one is removed from the front of the queue. If the queue is empty, the process blocks until a message becomes available.

Uses or applications of UDP

- UDP is used for process with simple request-response communication with little concern for and error control.
- UDP is suitable for a process with internal flow and error control mechanism.
- UDP is suitable for multicasting. Multicasting capabilities are embedded in UDP software but not in TCP software.
- UDP is used for some route updating protocols, such as routing information protocol (RIP).
- UDP is used for management processes such as SNMP.

4. Describe in detail about TCP segment (Header) format (NOV 2013, 2014)(May & Nov 2015) or Draw the format of TCP Packet header and explain each of its field and Specify the justification for having variable field length for the field in TCP header. Apr 2017

A Packet in TCP is called a segment. The below diagram shows the format of the segment.
The segment consists of a 20 to 60 byte header, followed by data from the application program.
The header is 20 bytes if there are no options and up to 60 bytes if it contains options.

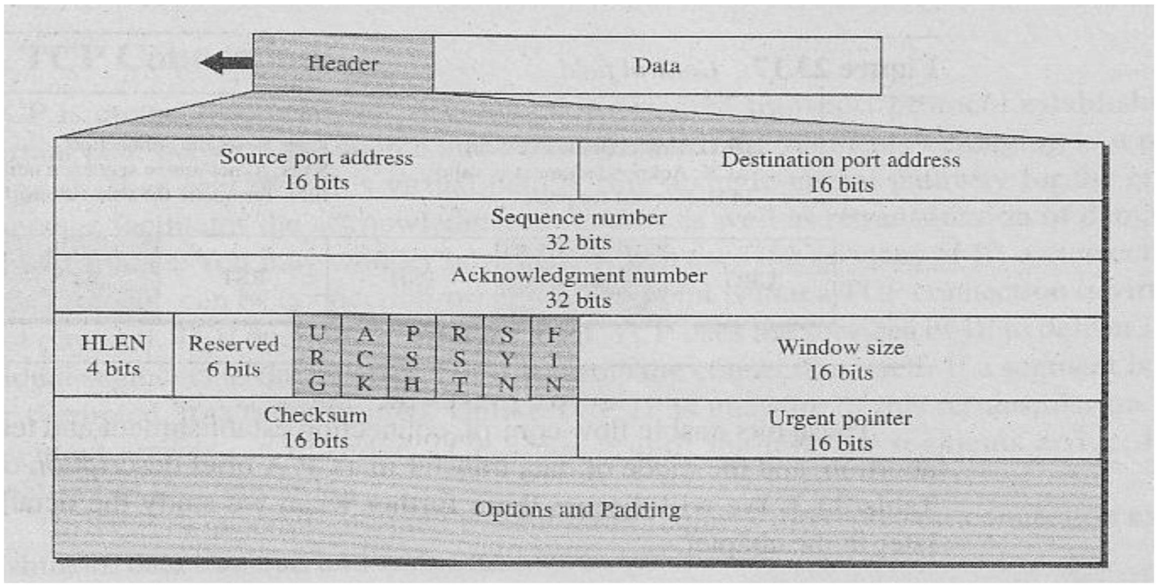
- *Header*

– The header is composed of a 20-byte fixed part and an optional part with a variable length. The total size of the header (in 32-bit words) is specified in HLEN.

- *Data -* The data can have a variable size, which can be up to 65535 – 20 = 65515 bytes.
- *Source port number (16 bits)*
  – The *SOURCE PORT* field identifies the *TCP process* which sent the datagram.

- *Destination port number (16 bits)*

The *DESTINATION PORT* field identifies the *TCP process* which is receiving the datagram



- *Sequence number (32 bits)*
  – The *SEQUENCE NUMBER* field identifies the first byte of the outgoing data. The receiver uses this to re-order segments arriving out of order and to compute an acknowledgement number.

- *Acknowledgement number (32 bits)*
  – Contains the next sequence number that the sender of the acknowledgement expects to receive which is the sequence number plus 1 (plus the number of bytes received in the last message). This number is used only if the ACK flag is on. The *ACKNOWLEDGEMENT NUMBER* field identifies the sequence number of the incoming data that is expected next.
- *Header Length*
  – This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes.

- *Reserved*
  – This is a 6-bit field reserved for future use.

- *Code bits*
  – The *CODE BITS* (or *FLAGS*) field contains one or more 1-bit flags
  – Control bits to indicate end of stream, acknowledgement field being valid, connection reset, urgent pointer field being valid, etc.

[CONTROL]: URG (1)-	Urgent Bit validates the Urgent Pointer field.
[CONTROL]: ACK (1)-	Acknowledge Bit, set if the Acknowledge Number field is being used.
[CONTROL]: PSH (1)-	Push Bit tells the sender that a higher throughput is required.
[CONTROL]: RST (1)-	Reset Bit resets the connection when there's conflicting sequence numbers.
[CONTROL]: SYN (1)-	Sequence Number Synchronization. Used in 3 types of segments: connection request, connection confirmations (with ACK) and confirmation termination (with FIN) in 3 types of segments: terminal request, terminal confirmation (with ACK) and acknowledgement of terminal confirmation (with ACK).
[CONTROL]: FIN (1)-	Used with SYN to confirm termination of connections

- *Window Size*(16 bit)
  - The *WINDOW* field identifies how much buffer space is available for incoming data.
  - During piggybacking, how much data a receiver is willing to accept.

Note: The process of sending data along with the acknowledgment is called piggybacking

- *Checksum*(16 bit)
  - The *CHECKSUM* field contains a simple checksum over the TCP segment header and data.
- *Urgent Pointer* (16 bit)
  - This 16-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines the number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment.
- *Options*
  - There can be up to 40 bytes of optional information in the TCP header.

5. Explain in detail about TCP connection establishment & termination (TCP Connection Management) (NOV 2013) (May & Nov 2015) Nov 2017

TCP is connection-oriented. A connection-oriented transport protocol establishes a virtual path between the source and destination. All the segments belonging to a message are then sent over this virtual path. Using a single virtual pathway for the entire message facilitates the acknowledgement process as well as retransmission of damaged or lost frames. TCP, which uses the services of IP, a connection-less protocol, can be connection-oriented. TCP uses the services of IP to deliver individual segments to the receiver, but it controls the connection itself. If a segment is lost or corrupted, it is retransmitted.

In TCP connection-oriented transmission requires two phases:

- ✓ Connection establishment and Data transfer
- ✓ Connection termination

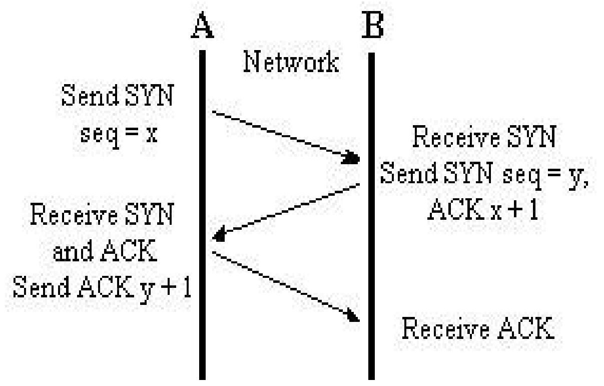
Connection establishment:

TCP transmits data in full-duplex mode. When two TCP's in two machines or connected, they are able to send segments to each other simultaneously.

Three-way handshaking.

The connection establishment in TCP is called three way handshaking. The process starts with the server. The server program tells its TCP that it is ready to accept a connection. This is called a request for a passive open.

The client program issues a request for an active open. A client that wishes to connect to an open server tells its TCP that it needs to be connected to that particular server. TCP can now start the three-way handshaking process. Each segment has the sequence number the acknowledgement number, the control flags, and the window size, if not empty.



The three steps in this phase are as follows.

1. The client sends the first segment, a SYN segment, in which only the SYN flag is set. This segment is for synchronization of sequence numbers. It consumes one sequence number. When the data transfer starts, the sequence number is incremented by 1. A SYN segment cannot carry data, but it consumes one sequence number.

2. The server sends the second segment, a SYN + ACK segment, with 2 flag bits set: SYN and ACK. This segment has a dual pupose. It is a SYN segment for communication in the other direction and serves as the acknowledgement for the SYN segment. It consumes one sequence number.

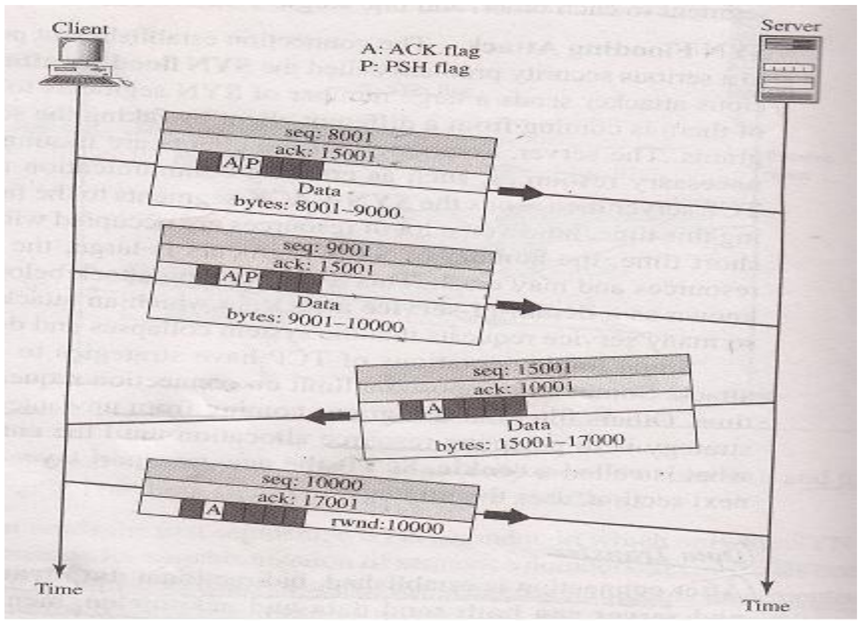
3. The client sends the third segment. This is just an ACK segment. It acknowdeges the receipt of the second segmant with the ACK flag and acknowledgment number field.

Data Transfer

After connection is established, bidirectional data transfer can take place. The client and server can both send data and acknowledgements.

The below figure shows an example. In this example, after connection is established, the client sends 2000 bytes of data in two segments. The server then sends 2000 bytes in one segment. The client sends one more segment. The first three segments carry both data and acknowledgment, but the last segment carries only an acknowledgement because there are no more data to be sent.

The data segments sent by the client have the PSH (push) flag set so that the server TCP knows to deliver data to the server process as soon as they are received.



Pushing Data: The sending TCP uses a buffer to store the stream of data coming from the sending application program. The sending TCP can select the segment size. The receiving TCP also buffers the data when they arrive and delivers them to the application program when the application program is ready or when it is convenient for the receiving TCP. This type of flexibility increases the efficiency of TCP.

The application program at the sending site can request a push operation. This means that the sending TCP must not wait for the window to be filled. It must create a segment and send it immediately. The sending TCP must also set the push bit (PSH) to let the receiving TCP know that the segment includes data that must be delivered to the receiving application program as soon as possible and not to wait for more data to come.

TCP Connection Release (or) Connection Termination

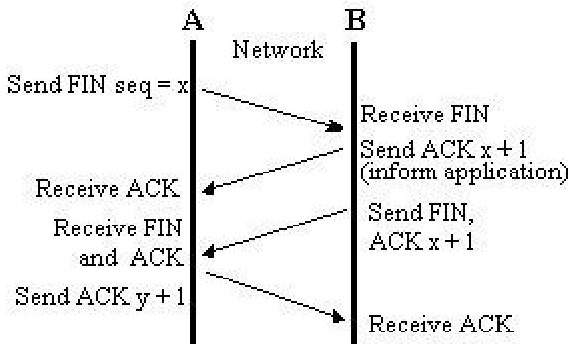
Any of the two parties involved in exchanging data (client or server) can close the connection, although it is usually initiated by the client. Most implementations today allow two options for connection termination: three-way handshaking and four-way handshaking with a half-close option.

Three-way handshaking

In a normal situation, the client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment in which the FIN flag is set. The FIN segment consumes one sequence number if it does not carry data.

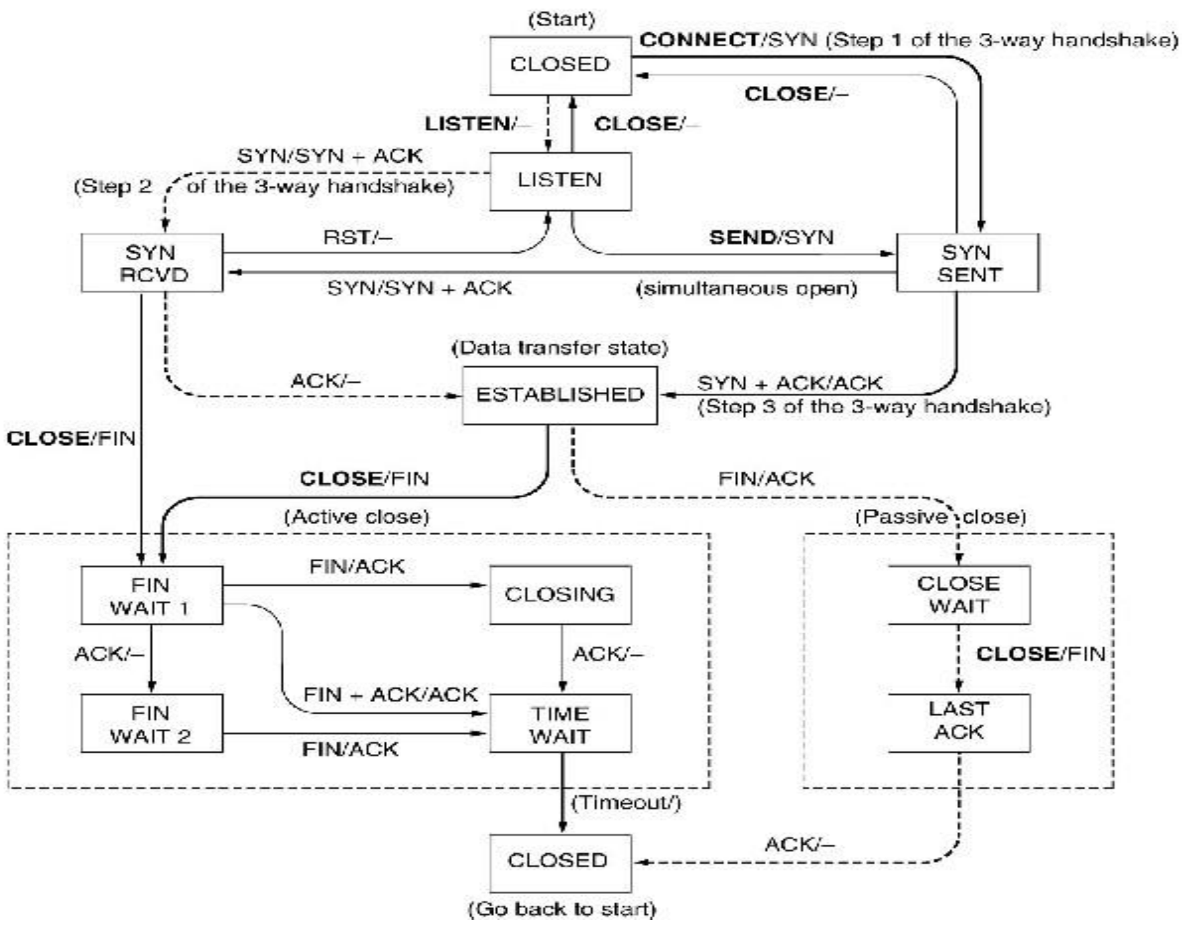
1. The server TCP, after receiving the FIN segment, informs its process of the situation and sends the second segment, a FIN + ACK segment, to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction. This segment can also contain the last chunk of data from the server. The FIN + ACK segment consumes one sequence number if it does not carry data.

2. The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server. This segment contains the acknowledgement number, which is 1 plus the sequence number received in the FIN segment from the server. This segment cannot carry data and consumes no sequence number.



TCP Connection Management (State transition diagram) (NOV/DEC 2013)

The steps to manage a TCP connection can be represented in a finite state machine with the eleven states listed in Figure



State	Description
CLOSED	No connection is active or pending
LISTEN	The server is waiting for an incoming call
SYN RCVD	A connection request has arrived; wait for ACK
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state
FIN WAIT 1	The application has said it is finished
FIN WAIT 2	The other side has agreed to release
TIMED WAIT	Wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	Wait for all packets to die off

Client Diagram:

- The client TCP starts in the CLOSED state.
- While in this state, the client TCP can receive an active open request from the client application program. It sends a SYN segment to the server TCP and goes to the SYN-SENT state.
- While in this state, the client TCP can receive a SYN + ACK segment from other TCP. It sends an ACK segment to the other TCP and goes to the ESTABLISHED state. This is the data transfer state. The client remains in this state as long as it is sending and receiving data.
- While in this state, the client TCP can receive a close request from the client application program. It sends a FIN segment to the other TCP and goes to the FIN-WAIT1 state.
- While in this state, the client TCP waits to receive an ACK from the server TCP. When ACK is received, it goes to the FIN-WAIT2 state. Now the connection is closed in one direction.
- The client remains in this state, waiting for the server to close the connection. If it receives a FIN segment, it sends an ACK segment and goes to the TIME-WAIT state.
- When the client is in this state, it starts a timer and waits until this timer goes off. After the time-out, the client goes to the CLOSED state, where it began.

Server Diagram:

- The server TCP starts in the CLOSED state.
- While in this state, the server TCP can receive an open request from the server application program, it goes to the LISTEN state.
- While in this state, the server TCP can receive a SYN segment. It sends a SYN + ACK segment to the client and goes to the SYN-RCVD state.
- While in this state, the server TCP receives an ACK and goes to ESTABLISHED state. This is the data transfer state. The server remains in this state as long as it is receiving and sending data.
- While in this state, the server TCP can receive a FIN segment from the client. It can send an ACK and goes to the CLOSE-WAIT state.
- While in this state, the server waits until it receives a close request from the server program. It then sends a FIN segment and goes to LAST-ACK state.
- While in this state, the server waits for the last ACK segment and goes to the CLOSED state.

6. Discuss TCP Services & Features

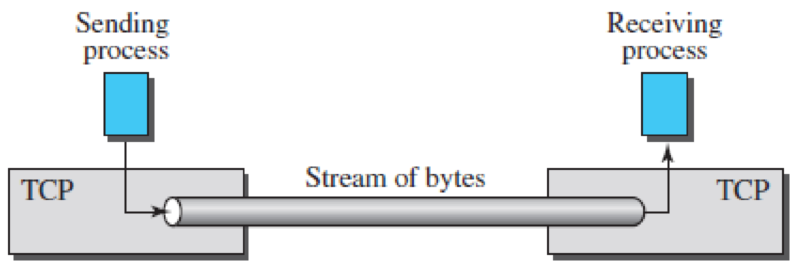
Transmission Control Protocol (TCP) is a connection-oriented, reliable protocol.
TCP explicitly defines connection establishment, data transfer, and connection teardown phases to provide a connection-oriented service

<u>TCP Services</u>

Process-to-Process Communication
As with UDP, TCP provides process-to-process communication using port numbers

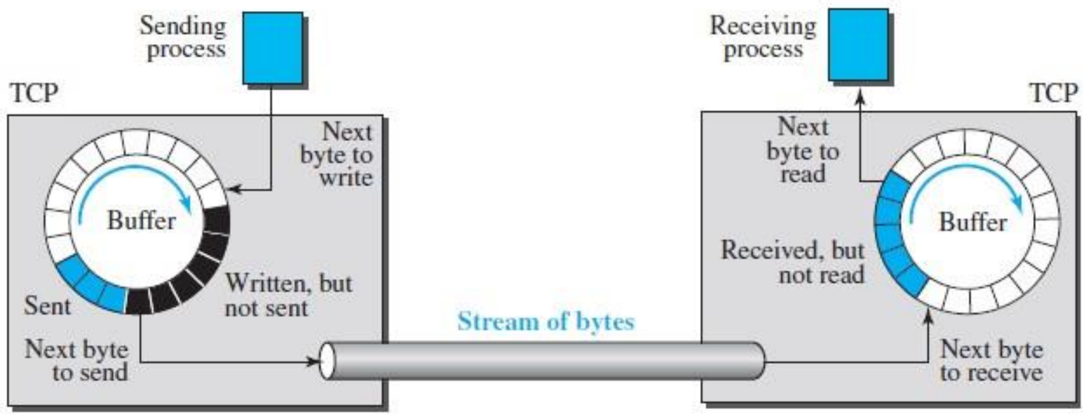
Stream Delivery Service

Figure 24.4 *Stream delivery*



Sending and Receiving Buffers

Figure 24.5 *Sending and receiving buffers*



Full-Duplex Communication
TCP offers *full-duplex service,* where data can flow in both directions at the same time.
Each TCP endpoint then has its own sending and receiving buffer, and segments move in both directions

Multiplexing and Demultiplexing
Like UDP, TCP performs multiplexing at the sender and demultiplexing at the receiver.

However, since TCP is a connection-oriented protocol, a connection needs to be established for each pair of processes.

Connection-Oriented Service

TCP, unlike UDP, is a connection-oriented protocol. When a process at site A wants to send to and receive data from another process at site B, the following three phases occur:
1. The two TCP's establish a logical connection between them.
2. Data are exchanged in both directions.
3. The connection is terminated.

Reliable Service

TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data. We will discuss this feature further in the section on error control.

TCP Features

- *Numbering System*
- *Byte Number*
- *Sequence Number*
- *Acknowledgment Number*

Windows in TCP:

Send Window

The sender maintains three variables:

- ✓ The *send window size*, denoted SWS, gives the upper bound on the number of outstanding (unacknowledged) frames that the sender can transmit;
- ✓ LAR denotes the sequence number of the *last acknowledgment received*; and
- ✓ LFS denotes the sequence number of the *last frame sent*.

The sender also maintains the following invariant:

$$\boxed{\mathbf{LFS - LAR \leq SWS}}$$

When an acknowledgment arrives, the sender moves LAR to the right, thereby allowing the sender to transmit another frame. Also, the sender associates a timer with each frame it transmits, and it retransmits the frame should the timer expire before an ACK is received. Notice that the sender has to be willing to buffer up to SWS frames since it must be prepared to retransmit them until they are acknowledged.

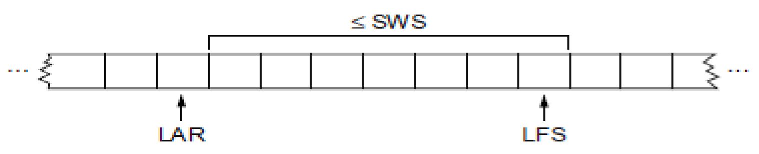
Receive Window

The receiver maintains the following three variables:

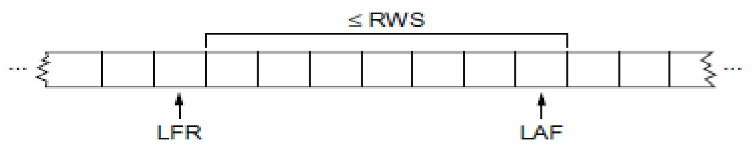
- ✓ The *receive window size*, denoted RWS, gives the upper bound on the number of outof- order frames that the receiver is willing to accept;
- ✓ LAF denotes the sequence number of the *largest acceptable frame*; and
- ✓ LFR denotes the sequence number of the *last frame received.*

The receiver also maintains the following invariant

$$\mathbf{LAF - LFR \leq RWS}$$



■ **FIGURE 2.20** Sliding window on sender.



■ **FIGURE 2.21** Sliding window on receiver.

When a frame with sequence number SeqNum arrives, the receiver takes the following action. If SeqNum ≤ LFR or SeqNum > LAF, then the frame is outside the receiver's window and it is discarded. If LFR < SeqNum ≤ LAF, then the frame is within the receiver's window and it is accepted.
Now the receiver needs to decide whether or not to send an ACK. Let SeqNumToAck denote the largest sequence number not yet acknowledged, such that all frames with sequence numbers less than or equal to SeqNumToAck have been received.
The receiver acknowledges the receipt of SeqNumToAck, even if higher numbered packets have been received. This acknowledgment is said to be cumulative. It then sets LFR = SeqNumToAck and adjusts LAF = LFR+RWS.

7. Explain in detail about TCP flow control OR TCP Adaptive flow control (NOV/DEC 2013, 2014) APR 2017

TCP uses a sliding *window* mechanism to control the flow of data. When a connection is established, each end of the connection allocates a buffer to hold incoming data, and sends the size of the buffer to the other end. As data arrives, the receiver sends acknowledgements together with the amount of buffer space available called a *window advertisement.*

Sliding window algorithm serves several purposes.

1. It guarantees the reliable delivery of data
2. It ensures that data is delivered in order.
3. It enforces flow control between, the sender and the receiver.

Reliable and Ordered Delivery

To see how the sending and receiving sides of TCP interact with each other to implement reliable and ordered delivery, consider the situation illustrated in Figure 5.8. TCP on the sending side maintains a send buffer. This buffer is used to store data that has been sent but not yet acknowledged, as well as data that has been written by the sending application but not transmitted. On the receiving side, TCP maintains a receive buffer. This buffer holds data that arrives out of order, as well as data that is in the correct order (i.e., there are no missing bytes earlier in the stream) but that the application process has not yet had the chance to read.

To make the following discussion simpler to follow, we initially ignore the fact that both the buffers and the sequence numbers are of some finite size and hence will eventually wrap around. Also, we do not distinguish between a pointer into a buffer where a particular byte of data is stored and the sequence number for that byte

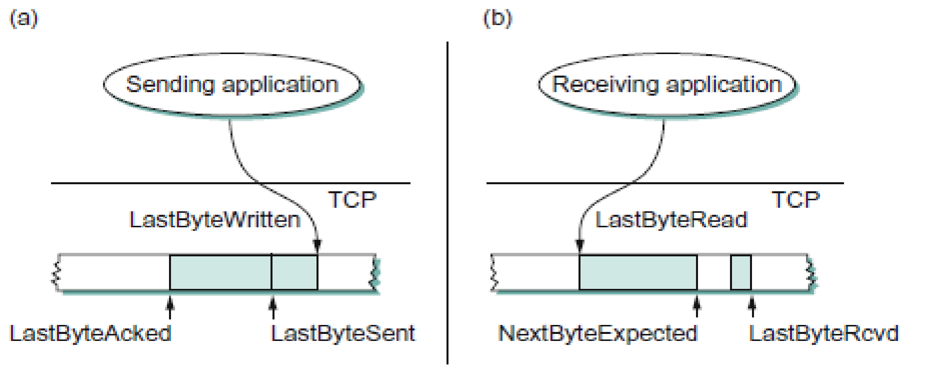


FIGURE 5.8 Relationship between TCP send buffer (a) and receive buffer (b).

Looking first at the sending side, three pointers are maintained into the send buffer, each with an obvious meaning: LastByteAcked, LastByteSent, and LastByteWritten. Clearly

$$\text{LastByteAcked} \le \text{LastByteSent}$$

since the receiver cannot have acknowledged a byte that has not yet been sent, and

$$\text{LastByteSent} \le \text{LastByteWritten}$$

since TCP cannot send a byte that the application process has not yet written. Also note that none of the bytes to the left of LastByteAcked need to be saved in the buffer because they have already been acknowledged, and none of the bytes to the right of LastByteWritten need to be buffered because they have not yet been generated.

A similar set of pointers (sequence numbers) are maintained on the receiving side: LastByteRead, NextByteExpected, and LastByteRcvd. The inequalities are a little less intuitive, however, because of the problem of out-of-order delivery. The first relationship

$$\text{LastByteRead} < \text{NextByteExpected}$$

is true because a byte cannot be read by the application until it is received *and* all preceding bytes have also been received. NextByteExpected points to the byte immediately after the latest byte to meet this criterion. Second,

$$\text{NextByteExpected} \le \text{LastByteRcvd}+1$$

since, if data has arrived in order, NextByteExpected points to the byte after LastByteRcvd, whereas if data has arrived out of order, then NextByteExpected points to the start of the first gap in the data, as in Figure 5.8.Note that bytes to the left of LastByteRead need not be buffered because they have already been read by the local application process, and bytes to the right of LastByteRcvd need not be buffered because they have not yet arrived.

Flow Control

Recall that in a sliding window protocol, the size of the window sets the amount of data that can be sent without waiting for acknowledgment from the receiver. Thus, the receiver throttles the sender by advertising a window that is no larger than the amount of data that it can buffer. Observe that TCP on the receive side must keep

$$\text{LastByteRcvd} - \text{LastByteRead} \leq \text{MaxRcvBuffer}$$

to avoid overflowing its buffer. It therefore advertises a window size of

$$\text{AdvertisedWindow} = \text{MaxRcvBuffer} - ((\text{NextByteExpected} - 1) - \text{LastByteRead})$$

which represents the amount of free space remaining in its buffer. As data arrives, the receiver acknowledges it as long as all the preceding bytes have also arrived. In addition, LastByteRcvd moves to the right (is incremented), meaning that the advertised window potentially shrinks. Whether or not it shrinks depends on how fast the local application process is consuming data. If the local process is reading data just as fast as it arrives (causing LastByteRead to be incremented at the same rate as LastByteRcvd), then the advertised window stays open (i.e., AdvertisedWindow = MaxRcvBuffer). If, however, the receiving process falls behind, perhaps because it performs a very expensive operation on each byte of data that it reads, then the advertised window grows smaller with every segment that arrives, until it eventually goes to 0.

TCP on the send side must then adhere to the advertised windowit gets from the receiver. This means that at any given time, it must ensure that

$$\text{LastByteSent} - \text{LastByteAcked} \leq \text{AdvertisedWindow}$$

Said another way, the sender computes an *effective* window that limits how much data it can send:

$$\text{EffectiveWindow} = \text{AdvertisedWindow} - (\text{LastByteSent} - \text{LastByteAcked})$$

All the while this is going on, the send side must also make sure that the local application process does not overflow the send buffer—that is, that

$$\text{LastByteWritten} - \text{LastByteAcked} \leq \text{MaxSendBuffer}$$

If the sending process tries to write y bytes to TCP, but

$$(\text{LastByteWritten} - \text{LastByteAcked}) + y > \text{MaxSendBuffer}$$

then TCP blocks the sending process and does not allow it to generate more data.

8. Discuss TCP adaptive retransmission APR 2017

TCP ADAPTIVE RETRANSISSION

TCP copies with the loss of packets using a technique called *retransmission*. When TCP data arrives an *acknowledgement* is sent back to the sender. Whenever a TCP segment is transmitted, a copy of it is also placed on the retransmission queue. When TCP data is sent, a timer is started this starts from a particular value and counts down to zero. If the timer expires before an acknowledgement arrives, TCP retransmits the data.

Three algorithm of adaptive retransmission

- ✓ Simple algorithm (Original algorithm)
- ✓ Kern/Partridge algorithm
- ✓ Jacobson/Karels algorithm

Original Algorithm

We begin with a simple algorithm for computing a timeout value between a pair of hosts. This is the algorithm that was originally described in the TCP specification—and the following description presents it in those terms—but it could be used by any end-to-end protocol.

The idea is to keep a running average of the RTT and then to compute the timeout as a function of this RTT. Specifically, every time TCP sends a data segment, it records the time. When an ACK for that segment arrives, TCP reads the time again, and then takes the difference between these two times as a SampleRTT. TCP then computes an EstimatedRTT as a weighted average between the previous estimate and this new sample. That is,

$$\text{EstimatedRTT} = \alpha \times \text{EstimatedRTT} + (1 - \alpha) \times \text{SampleRTT}$$

The parameter _ is selected to *smooth* the EstimatedRTT. A small _ tracks changes in the RTT but is perhaps too heavily influenced by temporary fluctuations. On the other hand, a large α is more stable but perhaps not quick enough to adapt to real changes. The original TCP specification recommended a setting of α between 0.8 and 0.9. TCP then uses EstimatedRTT to compute the timeout in a rather conservative way:

$$\textbf{TimeOut} = \textbf{2} \times \textbf{EstimatedRTT}$$

Karn/Partridge Algorithm

After several years of use on the Internet, a rather obvious flaw was discovered in this simple algorithm. The problem was that an ACK does not really acknowledge a transmission; it actually acknowledges the receipt of data. In other words, whenever a segment is retransmitted and then an ACK arrives at the sender, it is impossible to determine if this ACK should be associated with the first or the second transmission of the segment for the purpose of measuring the sample RTT.

It is necessary to know which transmission to associate it with so as to compute an accurate SampleRTT. As illustrated in Figure 5.10, if you assume that the ACK is for the original transmission but it was really for the second, then the SampleRTT is too large (a); if you assume that the ACK is for the second transmission but it was actually for the first, then the SampleRTT is too small (b).

The solution, which was proposed in 1987, is surprisingly simple. Whenever TCP retransmits a segment, it stops taking samples of the RTT; it only measures SampleRTT for segments that have been sent only once. This solution is knownas the Karn/Partridge algorithm, after its inventors.

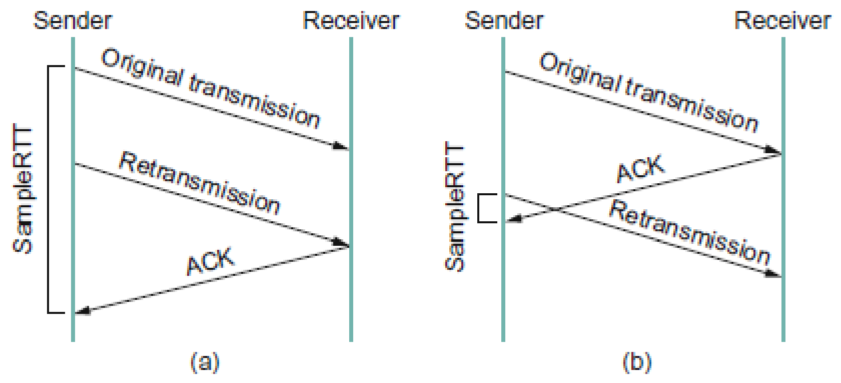


FIGURE 5.10 Associating the ACK with (a) original transmission versus (b) retransmission.

Their proposed fix also includes a second small change to TCP's timeout mechanism. Each time TCP retransmits, it sets the next timeout to be twice the last timeout, rather than basing it on the last EstimatedRTT. That is, Karn and Partridge proposed that TCP use exponential backoff, similar to what the Ethernet does. The motivation for using exponential backoff is simple: Congestion is the most likely cause of lost segments, meaning that the TCP source should not react too aggressively to a timeout. In fact, the more times the connection times out, the more cautious the source should become

Jacobson/Karels Algorithm

The Karn/Partridge algorithm was introduced at a time when the Internet was suffering from high levels of network congestion. Their approach was designed to fix some of the causes of that congestion, but, although it was an improvement, the congestion was not eliminated. The following year (1988), two other researchers—Jacobson and Karels—proposed a more drastic change to TCP to battle congestion. The bulk of that proposed change is described in Chapter 6. Here, we focus on the aspect of that proposal that is related to deciding when to time out and retransmit a segment.

As an aside, it should be clear how the timeout mechanism is related to congestion—if you time out too soon, you may unnecessarily retransmit a segment, which only adds to the load on the network. As we will see in Chapter 6, the other reason for needing an accurate timeout value is that a timeout is taken to imply congestion, which triggers a congestion-control mechanism. Finally, note that there is nothing about the Jacobson/Karels timeout computation that is specific to TCP. It could be used by any endto- end protocol.

The main problem with the original computation is that it does not take the variance of the sample RTTs into account. Intuitively, if the variation among samples is small, then the EstimatedRTT can be better trusted and there is no reason for multiplying this estimate by 2 to compute the timeout. On the other hand, a large variance in the samples suggests that the timeout value should not be too tightly coupled to the EstimatedRTT.

In the new approach, the sender measures a new SampleRTT as before. It then folds this new sample into the timeout calculation as follows:

$$\text{Difference} = \text{SampleRTT} - \text{EstimatedRTT}$$

$$\text{EstimatedRTT} = \text{EstimatedRTT} + (\delta \times \text{Difference})$$

$$\text{Deviation} = \text{Deviation} + \delta(|\text{Difference}| - \text{Deviation})$$

where δ is a fraction between 0 and 1. That is, we calculate both the mean RTT and the variation in that mean.

TCP then computes the timeout value as a function of both Estimated- RTT and Deviation as follows:

$$\text{TimeOut} = \mu \times \text{EstimatedRTT} + \phi \times \text{Deviation}$$

where based on experience, μ is typically set to 1 and ϕ is set to 4. Thus, when the variance is small, TimeOut is close to EstimatedRTT; a large variance causes the Deviation termto dominate the calculation.

9. Explain in detail about TCP congestion control mechanisms OR Brief about approaches used for TCP congestion control (NOV 2013, 2014, 2015, 2016,2017) OR With TCPs slow start and AIMD for congestion control,show how the window size will vary for a transmission where every 5TH Packet is lost.Assume an advertised window size of 50 MSS (APR 2017)

Congestion, in a network may occur if the load on the network – the number of packets sent to the network is greater than the capacity of the network – the number of packets a network can handle.

- Congestion control refers to the mechanisms and techniques to control the congestion and keep the load below the capacity that can either prevent congestion before it happens or remove congestion, after it has happened.
- There are two categories of congestion control
  - **Open-loop congestion control (prevention):** are applied to prevent congestion before it happens. In this, congestion control is handled by either the source or the destination.
  - **Closed-loop congestion control (removal):** try to remove congestion after it happens.

TCP CONGESTION CONTROL

Too many sources sending too much data too fast for network to handle. TCP uses congestion control to avoid congestion or remove congestion in the network.

Factors of congestion:

- Two senders, two receivers
- One router, infinite buffers
- No retransmission
- One router, finite buffers, reliable data transfer
- Sender retransmission of lost packet

Congestion Window

The sender’s window size is determined by the receiver and also by congestion in the network. The sender has two pieces of information:

i) The receiver – advertised window size (rwnd)
ii) The congestion window size (cwnd)

The actual size of the window is the minimize of these two
Max window = min (Congestion window, advertised window)
Effective window = Max window – (LastByteSend – LastByteAcked)
LastByteSend – LastByteAcked <= CongWin

Congestion Policy:
TCP handles congestion is based on three phases

i) Slow start (Exponential Increase)
ii) Additive Increase / Multiplicative Decrease
iii) Fast Retransmit and Fast Recovery

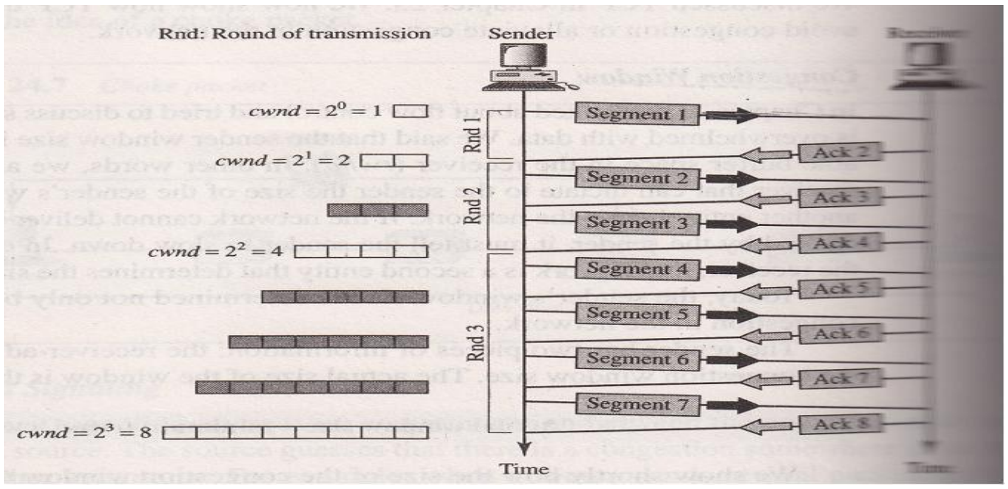
i) Slow Start

In this, the sender starts with a very slow rate of transmission but increases the rate rapidly to reach a threshold.

Slow start adds another window to the sender's TCP: the congestion window, called "cwnd". When a new connection is established with a host on another network, the congestion window is initialized to one segment. Each time an ACK is received, the congestion window is increased by one segment. The sender can transmit up to the minimum of the congestion window and the advertised window. The congestion window is flow control imposed by the sender, while the advertised window is flow control imposed by the receiver. The former is based on the sender's assessment of perceived network congestion; the latter is related to the amount of available buffer space at the receiver for this connection.

The sender starts by transmitting one segment and waiting for its ACK. When that ACK is received, the congestion window is incremented from one to two, and two segments can be sent. When each of those two segments is acknowledged, the congestion window is increased to four. This provides an exponential growth, although it is not exactly exponential because the receiver may delay its ACKs, typically sending one ACK for every two segments that it receives.

At some point the capacity of the internet can be reached, and an intermediate router will start discarding packets. This tells the sender that its congestion window has gotten too large.



Early implementations performed slow start only if the other end was on a different network. Current implementations always perform slow start.

- The source starts with cwnd = 1.
- Every time an ACK arrives, cwnd is incremented.
- Two slow start situations:
  - At the very beginning of a connection **{cold start}**.
  - When the connection goes dead waiting for a timeout to occur (i.e, the advertized window goes to zero!)
- However, in the second case the source has more information. The current value of cwnd can be saved as a **congestion threshold.**
- This is also known as the "slow start threshold" **ssthresh**.

When the size of window in bytes reaches this threshold, slow start stops and the next phase starts.

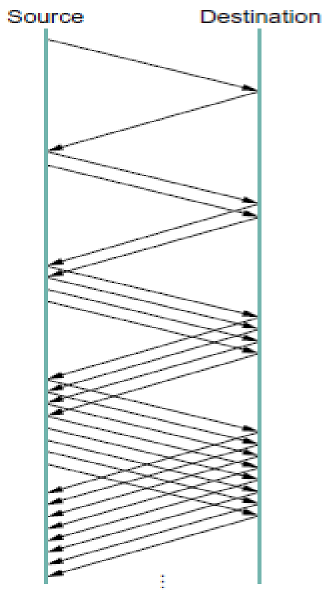
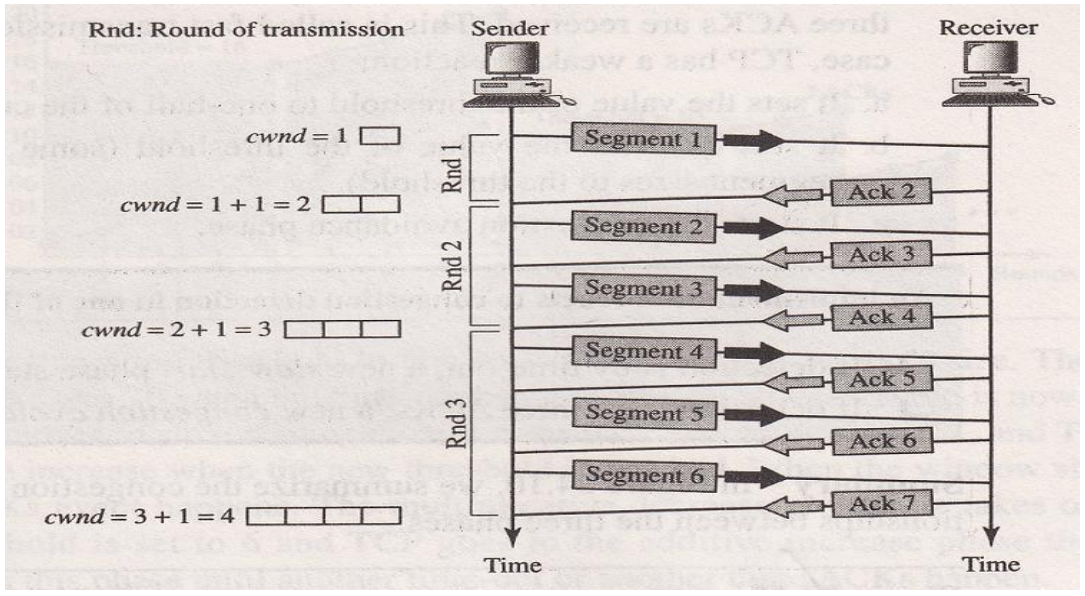


FIGURE 6.10 Packets in transit during slow start.

ii) Additive Increase (Congestion avoidance) / Multiplicative Decrease

To avoid congestion before it happens, one must slow down the exponential growth. When the size of the congestion window reaches the slow start threshold, the slow start phase steps and the additive phase begins. In this, each time the whole window of segments is acknowledged, the size of the congestion window is increased by 1.



After the sender has received acknowledgements for a complete window size of segments, the size of the congestion window increases additively until congestion is detected.
The congestion window is incremented as follows each time an ACK arrives:

Increment = MSS X (MSS / congestion window)
Congestion Window += Increment
Where MSS – Message Segment Size.

Multiplicative Decrease

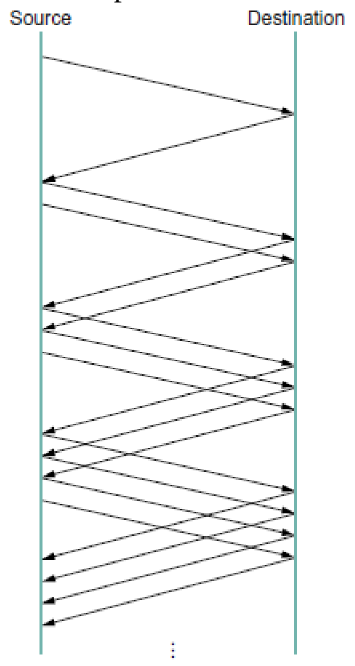
If congestion occurs, the congestion window size must be decreased. Retransmission can occur in one of two cases, when a timer times out (or) when three ACKS are received. In both cases, the size of the threshold is dropped to one-half, a multiplicative decrease.

TCP implementations have two reactions:

1. If a time-out occurs, there is a stronger possibility of congestion, a segment has probably been dropped in the network, and there is no news about the sent segments. In this, TCP reacts the following
    a. It sets the value of the threshold to one-half of the current window size.
    b. It sets cwnd to the size of one segment.
    c. It starts the slow-start phase again.
2. If three ACK's are received, there is a weaker possibility of congestion, a segment may have been dropped, but some segments after that may have arrived safely since three ACKs are received. This is called fast transmission and fast recovery.

    In this, TCP reacts the following:

a. It sets the value of the threshold to one-half of the current window size.
b. It sets cwnd to the value of the threshold.
c. It starts the congestion avoidance phase



■ FIGURE 6.8 Packets in transit during additive increase, with one packet being added each RTT.

iii) Fast Retransmit & Fast Recovery

Every time a data packet arrives at the receiving side, the receiver responds with an acknowledgement. When a packet arrives out of order, TCP resends the same acknowledgement is sent the last time. This second transmission of the same acknowledgement is called a duplicate ACK.

When the sending side sees a duplicate ACK, it knows that the other side must have received a packet out of order. The sender waits until it sees some no. of duplicate ACK's and then retransmit the missing packet. TCP waits until it has seen three duplicate ACK's before retransmitting the packet.

In this diagram, the destination receives packets 1 & 2, but packet 3 is lost in the network. Thus the destination will send a duplicate ACK for packet 2 when packet 4 arrives, again when packet 5 arrives & so on. When the sender sees the third duplicate ACK for packet 2, the receiver had gotten packet 6, it retransmits packet s. When the retransmitted copy of packet 3 arrives at the destination, the receiver then sends a cumulative ACK for everything up to and including packet 6 back to the sends.

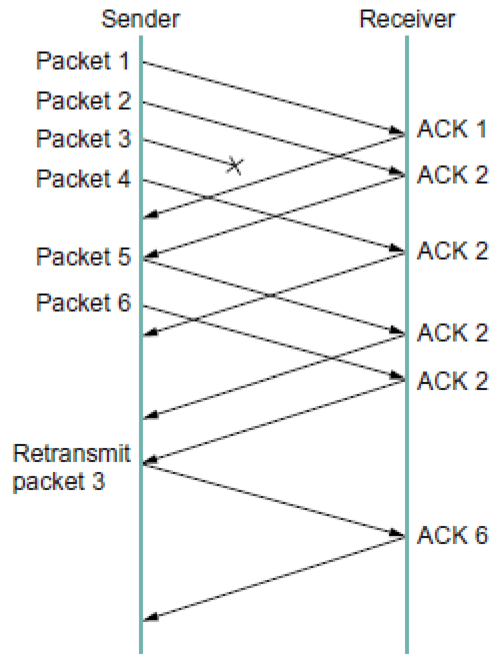


FIGURE 6.12 Fast retransmit based on duplicate ACKs.

Fast Recovery

<u>After fast retransmit sends what appears to be the missing segment, congestion avoidance, but not slow start is performed. This is the fast recovery algorithm. It is an improvement that allows high throughput under moderate congestion, especially for large windows.</u>

The reason for not performing slow start in this case is that the receipt of the duplicate ACKs tells TCP more than just a packet has been lost. Since the receiver can only generate the duplicate ACK when another segment is received, that segment has left the network and is in the receiver's buffer. That is, there is still data flowing between the two ends, and TCP does not want to reduce the flow abruptly by going into slow start.

The fast retransmit and fast recovery algorithms are usually implemented together as follows.

1. When the third duplicate ACK in a row is received, set ssthresh to one-half the current congestion window, cwnd, but no less than two segments. Retransmit the missing segment. Set cwnd to ssthresh plus 3 times the segment size. This inflates the congestion window by the number of segments that have left the network and which the other end has cached.

2. Each time another duplicate ACK arrives, increment cwnd by the segment size. This inflates the congestion window for the additional segment that has left the network. Transmit a packet, if allowed by the new value of cwnd.

3. When the next ACK arrives that acknowledges new data, set cwnd to ssthresh (the value set in step 1). This ACK should be the acknowledgment of the retransmission from step 1, one round-trip time after the retransmission. Additionally, this ACK should acknowledge all the intermediate segments sent between the lost packet and the receipt of

the first duplicate ACK. This step is congestion avoidance, since TCP is down to one-half the rate it was at when the packet was lost.

When fast retransmit detects three duplicate ACKs, start the recovery process from congestion avoidance region and use ACKs in the pipe to pace the sending of packets.

10. Write a detailed note on congestion avoidance mechanism used in TCP. NOV 2017
Or Explain congestion avoidance using random early detection in transport layer with example APR 2017

1. DEC Bit, 2. RED & 3. Source based Congestion Avoidance

DECbit

It is a first mechanism

- The idea here is to more evenly split the responsibility for congestion control between the routers and the end nodes.
- Each router monitors the load it is experiencing and explicitly notifies the end nodes when congestion is about to occur.
- This notification is implemented by setting a **binary congestion bit** in the packets that flow through the router: hence the name DECbit.
- The destination host then copies this congestion bit into the ACk it sends back to the source.
- Finally, the source adjusts its sending rate so as to avoid congestion.

How it is functioning:

- A single congestion bit is added to the packet header. A router sets this bit in a packet if its average queue length is grater than or equal to 1 at the time the packet arrives.
- This average queue length is measured over a time interval that distance the last bust + idle cycle, plus the current busy cycle. (The router is busy when it is transmitting and idles when it is not).
- The above figure shows the queue length at a router as a function of time. Essentially, the router calculates the area under the curve and divides this value by the time interval to compute the average queue length

If less than 50% of the packets had the bit set, then the source increases its congestion window by one packet. If 50% or more of the last window's worth of packets had the congestion bit set, the source decreases its congestion window to 0.875 times the previous value.

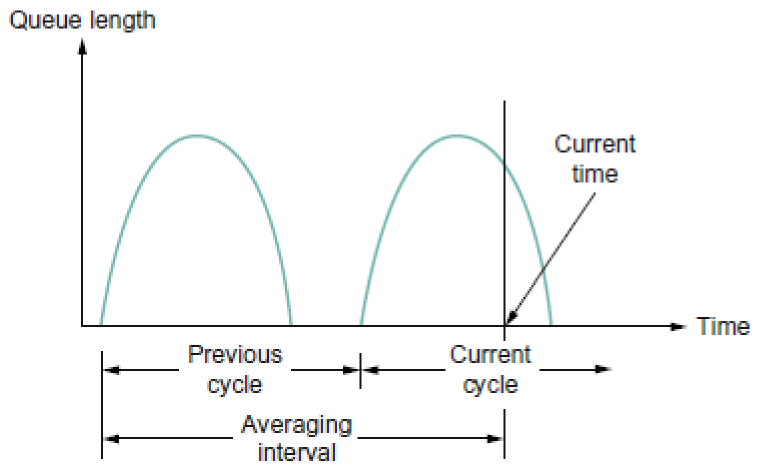


FIGURE 6.14 Computing average queue length at a router.

Random Early Detection (RED)

A second mechanism, called random early detection (RED), is similar to the DECbit scheme in that each router is programmed to monitor its own queue length, and when it detects that congestion is imminent (forthcoming), to notify the source to adjust its congestion window.

- The first is that rather than explicitly sending a congestion notification message to the source, RED is most commonly implemented such that it implicitly notifies the source of congestion by dropping one of its packets.
- The source is, effectively notified by the subsequent timeout or duplicates ACK. In case you haven't already guessed, RED is designed to be used in conjunction with TCP, which currently detects congestion by means of timeouts.
- As the "early" part of the RED acronym suggests, the gateway drops the packet earlier than it would have to, so as to notify the source that it should decrease its congestion window sooner than it would normally have.
- In other words, the router drops a few packets before it has exhausted its buffer space completely, so as to cause the source to slow down, with the hope that this will mean it does not have to drop lots of packets later on.
- Note that RED could easily be adapted to work with an explicit feedback scheme simply by marking a packet instead of dropping it, as discussed in the sidebar on Explicit Congestion Notification.

Source-Based Congestion Avoidance

- A strategy for detecting the initial stages of congestion – before losses occur – from the end hosts.

- The general idea of these techniques is to watch for some sign from the network that some router's queue is building up and that congestion will happen soon if nothing is done about it.

- A **first Scheme** the congestion window normally increases as in TCP, but every two round-trip delays the algorithm checks to see if the current RTT is greater than the average of the minimum and maximum RTT's seen so far. If it is, then the algorithm decreases the congestion window by one-eighth.

- A **second algorithm** is the decision as to whether or not to change the current window size is based on changes to both the RTT and the window size. The window is adjusted once every two round-trip delays based on the product

  (CurrentWindow – OldWindow) X (CurrentRTT – OldRTT)

  If the result is positive, the source decreases the window size by one-eighth; if the result is negative or 0, the source increases the window by one maximum packet size.
- A **third scheme**, Every RTT, it increases the window size by one packet and compares the throughput achieved to the throughput when the window was one packet smaller. If the difference is less than one-half the throughput achieved when only one packet was in transit. If the difference is greater than the algorithm decreases the window by one packet. This scheme calculates the throughput by dividing the number of bytes outstanding in the network by the RTT.

- A **fourth mechanism**, it looks at changes in the throughput rate, or more specifically, changes in the sending rate.

  It compares the measured throughput rate with an expected throughput rate. The algorithm, which is called **TCP Vegas**.
  TCP Vegas uses this idea to measure and control the amount of extra data this connection has in transit, where by "extra data" we mean that the source would not have transmitted had it been trying to match exactly the available bandwidth of the network. The goal of TCP Vegas is to maintain the "right" amount of extra data in the network.
  Obviously, if a source is sending too much extra data, it will cause long delays and possibly lead to congestion. Less obviously, if a connection is sending too little extra data, it cannot respond rapidly enough to transient increases in the available network bandwidth.
  - ✓ TCP Vegas sets BaseRTT to the minimum of all measured round-trip times; it is commonly the RTT of the first packet sent by the connection, before the router queues increase due to traffic generated by this flow. If we assume that we are not overflowing the connection, then the expected throughput is give by

    ExpectedRate = CongestionWindow / BaseRTT

    Where CongestinWindow is the TCP congestion window, which we assume (for the purpose of this discussion) to be equal to the number of bytes in transit.
  - ✓ Second TCP Vegas calculates the current sending rate, ActualRate. This is done by recording the sending time for a distinguished packet, recording how many

bytes are transmitted between the time that packet is sent and when its acknowledgment is received, computing the sample RTT for the distinguished packet when its acknowledgment arrives, and dividing the number of bytes transmitted by the sample RTT. This calculation is done once per round-trip time.

- ✓ Third, TCP Vegas compares ActualRate to ExpectedRate and adjusts the window accordingly. We let Diff = ExpectedRate – ActualRate. Note that Diff is positive or 0 by definition, since ActualRate > ExpectedRate implies that we need to change BaseRTT to the latest sampled RTT.

We also define two thresholds, $\alpha < \beta$, roughly corresponding to having too little and too much extra data in the network, respectively. When Diff < α, TCP Vegas increases the congestion window linearly during the next RTT, and when Diff > β, TCP Vegas decreases the congestion window linearly during the next RTT. TCP Vegas leaves the congestion window unchanged when $\alpha <$ Diff $< \beta$.

11. Explain SCTP (Stream Control Transmission Protocol) in detail

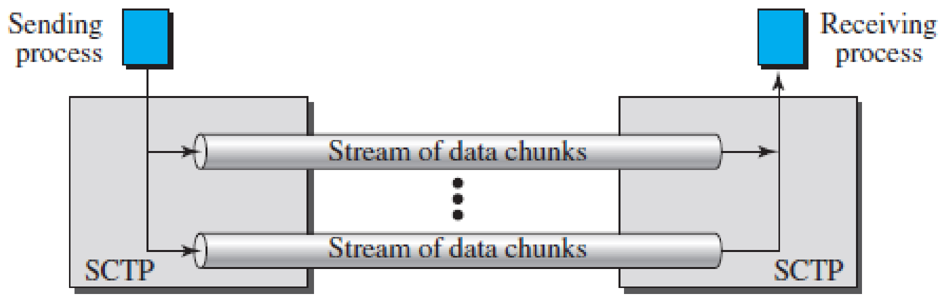
Stream Control Transmission Protocol (SCTP) is a new transport-layer protocol designed to combine some features of UDP and TCP in an effort to create a better protocol for multimedia communication.

SCTP Services

- ***Process-to-Process Communication***
- ***Multiple Streams***

SCTP allows **multistream service** in each connection, which is called ***association*** in SCTP terminology. If one of the streams is blocked, the other streams can still deliver their data.

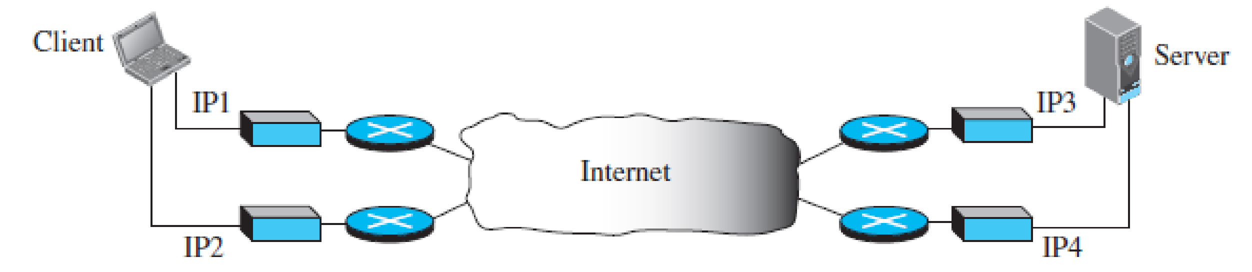
Figure 24.38 *Multiple-stream concept*



- ***Multihoming***

The sending and receiving host can define multiple IP addresses in each end for an association. In this fault-tolerant approach, when one path fails, another interface can be used for data delivery without interruption. This fault-tolerant feature is very helpful when we are sending and receiving a real-time payload such as Internet telephony.

Figure 24.39 *Multihoming concept*



- ***Full-Duplex Communication***
- ***Connection-Oriented Service***
- ***Reliable Service***

SCTP Features

Transmission Sequence Number (TSN)
The unit of data in SCTP is a data chunk, which may or may not have a one-to-one relationship with the message coming from the process because of fragmentation . Data transfer in SCTP is controlled by numbering the data chunks.
SCTP uses a **transmission sequence number (TSN)** to number the data chunks. In other words, the TSN in SCTP plays a role analogous to the sequence number in TCP.

Stream Identifier (SI)
In SCTP, there may be several streams in each association. Each stream in SCTP needs to be identified using a **stream identifier (SI).** Each data chunk must carry the SI in its header so that when it arrives at the destination, it can be properly placed in its stream. The SI is a 16-bit number starting from 0.

Stream Sequence Number (SSN)
When a data chunk arrives at the destination SCTP, it is delivered to the appropriate stream and in the proper order. This means that, in addition to an SI, SCTP defines each data chunk in each stream with a **stream sequence number (SSN).**

Acknowledgment Number
TCP acknowledgment numbers are byte-oriented and refer to the sequence numbers. SCTP acknowledgment numbers are chunk-oriented. They refer to the TSN. A second difference between TCP and SCTP acknowledgments is the control information

SCTP Packet Format

An SCTP packet has a mandatory general header and a set of blocks called chunks. There are two types of chunks: control chunks and data chunks. A control chunk controls and maintains the association; a data chunk carries user data. In a packet, the control chunks come before the data chunks. Figure 24.42 shows the general format of an SCTP packet.

General Header

The *general header* (packet header) defines the end points of each association to which the packet belongs, guarantees that the packet belongs to a particular association, and preserves the integrity of the contents of the packet including the header itself. The format of the general header is shown in Figure 24.43.

There are four fields in the general header. The source and destination port numbers are the same as in UDP or TCP. The verification tag is a 32-bit field that matches a packet to an association. This prevents a packet from a previous association from being mistaken as a packet in this association. It serves as an identifier for the association; it is repeated in every packet during the association. The next field is a checksum. However, the size of the checksum is increased from 16 bits (in UDP, TCP, and IP) to 32 bits in SCTP to allow the use of the CRC-32 checksum.

Figure 24.42 *SCTP packet format*

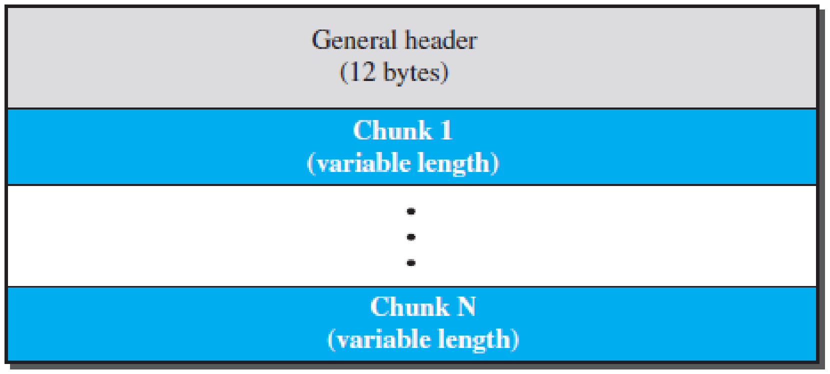
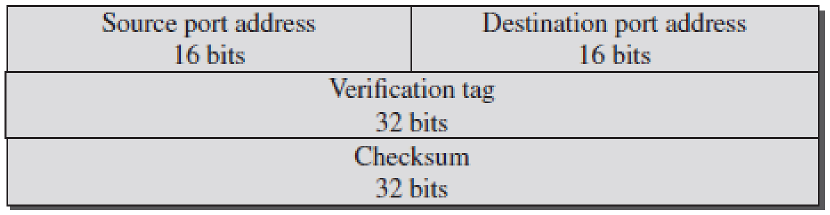


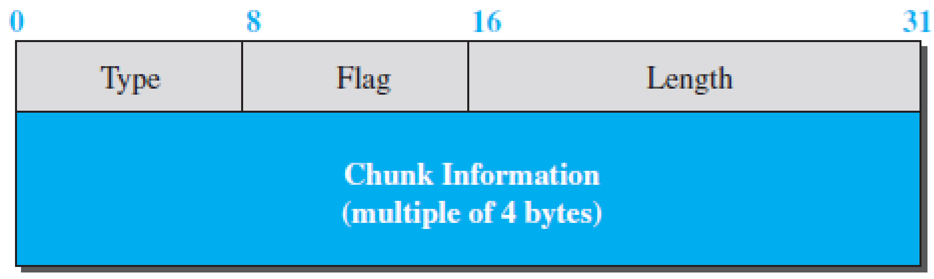
Figure 24.43 *General header*



Chunks

Control information or user data are carried in chunks. Chunks have a common layout, The first three fields are common to all chunks; the information field depends on the type of chunk

Figure 24.44 *Common layout of a chunk*



Types of Chunks
SCTP defines several types of chunks

Table 24.3 *Chunks*

Type	*Chunk*	*Description*
0	DATA	User data
1	INIT	Sets up an association
2	INIT ACK	Acknowledges INIT chunk
3	SACK	Selective acknowledgment
4	HEARTBEAT	Probes the peer for liveliness
5	HEARTBEAT ACK	Acknowledges HEARTBEAT chunk
6	ABORT	Aborts an association
7	SHUTDOWN	Terminates an association
8	SHUTDOWN ACK	Acknowledges SHUTDOWN chunk
9	ERROR	Reports errors without shutting down
10	COOKIE ECHO	Third packet in association establishment
11	COOKIE ACK	Acknowledges COOKIE ECHO chunk
14	SHUTDOWN COMPLETE	Third packet in association termination
192	FORWARD TSN	For adjusting cumulating TSN

An SCTP Association
SCTP, like TCP, is a connection-oriented protocol. However, a connection in SCTP is called an *association* to emphasize multihoming

- ***Association Establishment***
- ***Data Transfer***
    - *Multihoming Data Transfer*
    - *Multistream Delivery*
    - *Fragmentation*
- ***Association Termination***

Figure 24.45 *Four-way handshaking*

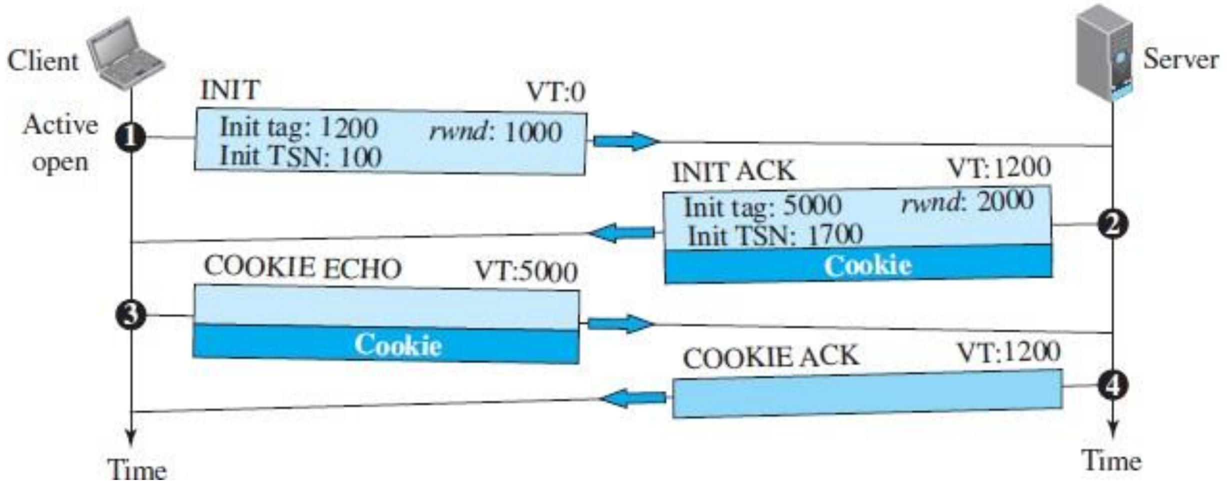
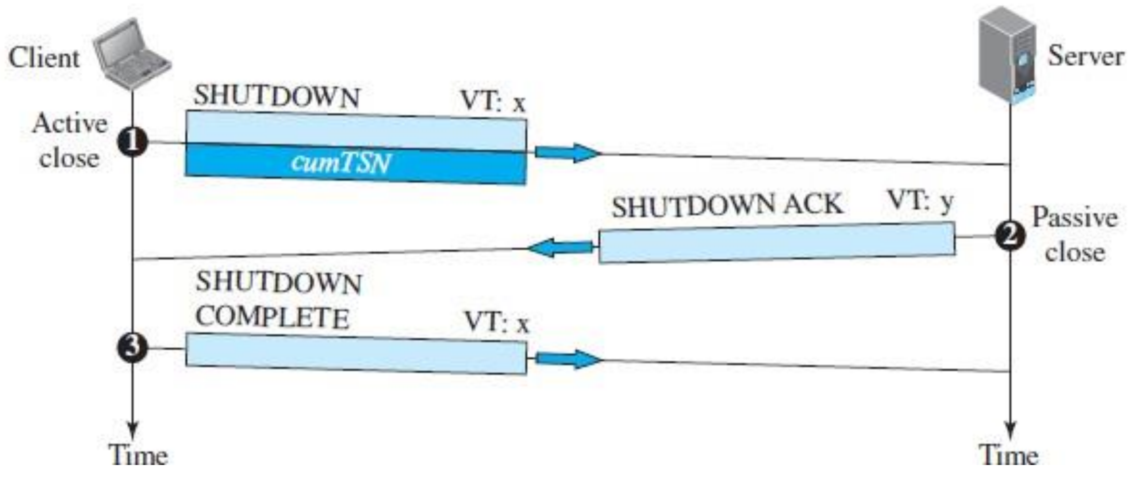


Figure 24.46 *Association termination*



UNIVERSITY QUESTIONS

B.E/B.TECH NOVEMBER/DECEMBER 2014(2008 Regulation)

2 MARKS

1. Differentiate TCP and UDP. (Q.NO. 2)
2. What is QOS? (Q.NO.32)

16 MARKS

1. Explain the following
   (i) TCP header (8) (Q.NO. 2)
   (ii) Adaptive flow control (8) (Q.NO. 4)
2. How is congestion controlled? Explain in detail the TCP congestion control (16)(Q.NO. 6)

B.E/B.Tech April May 2015

2 MARKS

1. List some of the Quality of service parameters of transport layer (Q.NO. 48)
2. How does transport layer perform duplication control? (Q.NO. 49)

16 MARKS

1. Explain the various fields of TCP header and the working of TCP protocol (16) (Q.NO. 2 & 3)
2 (i) i.Explain the three way handshake protocol to establish the transport level connection (8) (Q.NO. 3)
(ii) List the various congestion control mechanisms. Explain any one in detail (8) (Q.NO. 6)

B.E/B.Tech Nov-Dec 2015

2 MARKS

1. What is the difference between congestion control and flow control? (Q.NO 41)
2. What do you mean by QoS? (Q.NO 32)

16 MARKS

1. With a neat architecture, explain TCP in detail (Q.NO 2 & 3)
2. Explain TCP congestion control methods (Q.NO 6)

B.E/B.Tech April-May 2016

2 MARKS

1. What do you mean by slow start in TCP congestion? (Q.NO 50)
2. List the different phases used in TCP connection (Q.NO 51)

16 MARKS

1. Define UDP. Discuss the operations of UDP. Explain UDP checksum withone example. (Q.NO 1)
2. Explain in detail the various TCP congestion control mechanisms (Q.NO 6)

B.E/B.Tech Nov-Dec 2016

2 MARKS

1. Differentiate between TCP and UDP. **(Q.NO 2)**

16 MARKS

1. Explain various fields of TCP header and the working of the TCP protocol **(Q.NO 2)**

2. How is Congestion controlled? Explain in detail about congestion control techniques in transport layer **(Q.NO 6)**

B.E/B.Tech April-May 2017

2 MARKS

1. List out the advantages of connection oriented services over connectionless services. **(Q.NO 52)**

2. How do fast retransmit mechanism of TCP works. **(Q.NO 53)**

16 MARKS

1. (i) Explain the adaptive flow control and retransmission technique used in TCP **(Q.NO 4,5)**

(ii) With TCPs slow start and AIMD for congestion control, show how the window size will vary for a transmission where every 5th Packet is lost. Assume an advertised window size of 50 MSS. **(Q.NO 6)**

2. (i) Explain congestion avoidance using random early detection in transport layer with example**(Q.NO.7)**

(ii) Explain the different services operation of QOS in detail. **(Q.NO 9)**

Part-C

1. (i)Draw the format of TCP Packet header and explain each of its field. **(Q.NO 2)**

(ii)Specify the justification for having variable field length for the field in TCP header.

B.E/B.Tech Nov-Dec 2017

2 MARKS

1. Compare flow control versus congestion control? **(Q.NO 41)**

2. What are the approaches used to provide range of quality of services? **(Q.NO 46)**

16 MARKS

1. (i)Draw the TCP state transition diagram for connection management. **(Q.NO 3)**

(ii)Brief about approaches used for TCP congestion control. **(Q.NO 6)**

2. Write a detailed note on congestion avoidance mechanism used in TCP. **(Q.NO 7)**

UNIT II - DATA-LINK LAYER & MEDIA ACCESS

Introduction – Link-Layer Addressing – DLC Services – Data-Link Layer Protocols – HDLC – PPP - Media Access Control - Wired LANs: Ethernet - Wireless LANs – Introduction – IEEE 802.11, Bluetooth – Connecting Devices.

PART A

1. What are the functions of MAC?

MAC sub layer resolves the contention for the shared media. It contains synchronization, flag, flow and error control specifications necessary to move information from one place to another, as well as the physical address of the next station to receive and route a packet.

2. What is Ethernet?

Ethernet is a multiple-access network, meaning that a set of nodes send and receive frames over a shared link.

3. Define Repeater?

A repeater is a device that forwards digital signals, much like an amplifier forwards analog signals. However, no more than four repeaters may be positioned between any pairs of hosts, meaning that an Ethernet has a total reach of only 2,500m.

4. Why Ethernet is said to be a I-*persistent* protocol?

An adaptor with a frame to send transmits with probability ‘1 ‘whenever a busy line goes idle.

5. What is exponential back off? (Nov 2016)

Once an adaptor has detected a collision and stopped its transmission, it waits a certain amount of time and tries again. Each time it tries to transmit but fails, the adaptor doubles the amount of time it waits before trying again. This strategy of doubling the delay interval between each transmission attempt is a general technique known as exponential back off.

6. What are the four prominent wireless technologies?

- Bluetooth
- Wi-Fi(formally known as 802.11)
- WiMAX(802.16)
- Third generation or 3G cellular wireless.

7. Define Bluetooth? (May 2016)

Bluetooth fills the niche of very short-range communication between mobile phones, PDAs, notebook computers, and other personal or peripheral devices. For example, Bluetooth can be used to connect mobile phones to a headset, or a notebook computer to a printer.

8. Explain the term handoff?

If the phone is involved in a call at the time , the call must be transferred to the new base station in what is called a hand off.

9. What is the use of Switch?

It is used to forward the packets between shared media LANs such as Ethernet. Such switches are sometimes known by the obvious name of LAN switches.

10. What is meant by circuit switching? (NOV/DEC 2010)

Circuit switching is a process that establishes connections on demand and permits exclusive use of those connections until released.

11. What is Spanning tree?

It is for the bridges to select the ports over which they will forward frames. A spanning tree is a subgraph of this graph that covers (spans) all the vertices but contains no cycles. That is, a spanning tree keeps all of the vertices of the original graph but throws out some of the edges

12. What are the three pieces of information in the configuration messages?

1. The ID for the bridge that is sending the message.
2. The ID for what the sending bridge believes to the root bridge.
3. The distance, measured in hops, from the sending bridge to the root bridge.

13. What is broadcast?

Broadcast is simple – each bridge forwards a frame with a destination broadcast address out on each active (selected) port other than the one on which the frame was received.

14. What is multicast?

It can be implemented with each host deciding for itself whether or not to accept the message.

15. How does a given bridge learn whether it should forward a multicast frame over a given port?

It learns exactly the same way that a bridge learns whether it should forward a unicast frame over a particular port- by observing the source addresses that it receives over that port.

16. Differentiate fast Ethernet and gigabit Ethernet. (NOV/DEC 2012)

Fast Ethernet cards connect to networks at a rate of 100 Mbps while Gigabit network cards can connect at speeds up to 1000mb/s. The main difference between the two is speed. A fast Ethernet card can run on bandwidths at 100mb/s while a gigabit Ethernet can run at ten times that speed. However, the existence of FDDIs around made this technology more like a stepping stone to something better – enter the gigabit card. Gigabit networks are made to run the best at Layer 3 switching meaning it has more route functionality than the 100mbs fast Ethernet.

17. What is Transceiver?

Transceiver is a device which connects host adaptor to Ethernet Cable. It receives and sends signal.

18. What is the difference between switch and bridge? (NOV/DEC 2012)

Bridge	Switch
A bridge is device which operates at the data link layer. It may be used to join two	A bridge with more than two interface (ports)is also known as a switch

LAN segment(A,B),Constructing a larger LAN	
Bridges receive Ethernet frames then forward all frames, like a repeater	A switch, on the other hand ,forward the frame to only the required interfaces
Bridges learns the association between the system MAC addresses and the interface ports.	The switch reduces the number the number of packets on the other LAN segments, by sending the packet only where it need to go.

.

19. Define bridge and switch. (NOV/DEC 2012)

- Bridges are software based ,while switches are hardware based
- Bridges can only have one spanning –tree instance per bridge, while switches can have many
- Bridges can only have up to 16 ports, whereas a switch can have hundreds.

20. State the difference between token ring and FDDI? (NOV/DEC 2010)

Token ring	**FDDI**
• It uses shielded twisted pair cables • It uses Manchester encoding • It supports data rate upto 16Mbps • It is implemented as a ring, switch or multistation access unit	It uses fibre optic cables. It uses 4B/5B before NRZ-1 for encoding It supports data rate upto100 Mpbs It is implemented as dual ring, nodes with single attachment station and dual attachment station with concentrator.

21. Define a Bridge. (NOV/DEC 2010)

A network bridge is an abstract device that connects multiple network segments along the data link layer. A concrete example of a bridge in a computer network is the network switch.

22. A network with bandwidth of 10 Mbps can pass only an average of 12,000 frames per minute with each frame carrying an average of 10,000 bits. What is the throughput of this network? (APRIL/MAY 2011)

Throughput= (12,000*10,000)/60=2Mbps.

It is 1/5th of bandwidth.

23. What is the role of VCI?

A virtual channel identifier (VCI) distinguishes virtual channels (also known as circuits) created in a packet/cell switched network. A VCI has multiple circuits per communication channel and is primarily used for managing the unique identification of each created circuit.

A VCI is also known as a virtual circuit identifier (VCI).

24. List the two main limitations of bridges.Nov/Dec 2013

- Limited scalability
  - to O(1,000) hosts
  - not to global networks
- Not heterogeneous
  - no translation between frame formats

25. Define source routing. Nov/Dec 2013

Source routing allows a sender of a packet to partially or completely specify the route the packet takes through the network.

Source routing allows easier troubleshooting, improved trace route, and enables a node to discover all the possible routes to a host. It does not allow a source to directly manage network performance by forcing packets to travel over one path to prevent congestion on another.

26. Why should Ethernet frame should be 512 bytes long?

A Valid collision can only happen within the first 512 bits of frame transmission.

The 512 bits include 12 bytes of addresses, plus 2 bytes used in the type/length field ,plus 46 bytes of data,plus 4 bytes of FCS. The preamble is not considered part of the actual frame in these calculations.

27. Define ICMP? (Or) Expand ICMP and write the function (May 2016)

Internet Control Message Protocol is a collection of error messages that are sent back to the source host whenever a router or host is unable to process an IP datagram successfully

28. Define Subnetting? (Nov 2015)

Subnetting provides an elegantly simple way to reduce the total number of network numbers that are assigned. The idea is to take a single IP network number and allocate the IP address with that network to several physical networks, which are now referred to as subnets.

29. What is CIDR? (MAY/JUNE2007)

Classless Inter-Domain Routing (**CIDR**) is a methodology of allocating and routing packets. It was introduced in 1993 to replace the prior addressing architecture of design in the with the goal to slow the growth of routing tables on routers across the Internet, and to help slow the rapid of addresses, uses a syntax of specifying IP addresses for IPv4 and IPv6, using the base address of the network followed by a slash and the size of the routing prefix, e.g., 192.168.0.0/16 (IPv4), and 2001:db8::/32 (IPv6).

30. What is (Differ) ARP and RARP? (MAY/JUNE 2009)

ARP stands for Address Resolution Protocol. It is used to convert IP address to Physical address.

RARP stands for Reverse Address Resolution Protocol. It is used to convert Physical address into IP address.

31. What is DHCP? (NOV/DEC 2012)

- Dynamic Host Configuration Protocol (DHCP) is a protocol designed to provide information dynamically.
- It is a client-server program.
- DHCP is used to assign addresses to a host dynamically.
- Basically, DHCP server has two databases.
- The first database is addresses to IP addresses.

32. What are the salient features of IPV6? (NOV/DEC 2012)

- New Packet Format and Header
- Large Address Space
- State full and Stateless IPv6 address
- Multicast
- Integrated

33. Give the CIDR notation for class A, B and C. APR/MAY 2011)

Class	Binary	Dotted-Decimal	CIDR
A	11111111 00000000 00000000 00000000	255.0.0.0	/8
B	11111111 11111111 00000000 00000000	255.255.0.0	/16
C	11111111 11111111 11111111 00000000	255.255.255.0	/24

34. What is IP addressing?

Internet address or IP address is 32 bit identifier that uniquely and universally defines a host or router connected to the internet.

35. What is the need of subnetting. (NOV/DEC 2013)

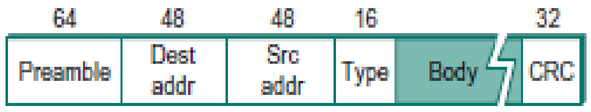
Subnetting is the technique used to break down networks into subnets. With the advent of internet, IP based networks become hugely popular. Due to this available IP addresses depleted at huge rate. To overcome this shortage concept of subnetting was introduced. Subnetting removes the classification of IP addresses according to classes and helps in creating further subnetworks from existing range of a IP network range.
For e.g A class B IP address can be broken down into further smaller networks.

36. What is the need for ARP? (NOV/DEC 2013) (Nov 2015)

Address Resolution Protocol (ARP) is a protocol for mapping an Internet Protocol address (IP address) to a physical machine address that is recognized in the local network. For example, in IP Version 4, the most common level of IP in use today, an address is 32 bits long.

In an Ethernet local area network, however, addresses for attached devices are 48 bits long. (The physical machine address is also known as a Media Access Control or MAC address.) A table, usually called the ARP cache, is used to maintain a correlation between each MAC address and its corresponding IP address. ARP provides the protocol rules for making this correlation and providing address conversion in both directions.

37. Draw Ethernet frame format (Dec 2017)



38. Define collision detection?

In Ethernet, all these hosts are competing for access to the same link, and as a consequence, they are said to be in the same collision detection.

39. What are the four steps involves in scanning?

The technique for selecting an AP is called *scanning* and involves the following four steps:

1. The node sends a Probe frame.
2. All APs within reach reply with a Probe Response frame.
3. The node selects one of the access points, and sends that AP an Association Request frame.
4. The AP replies with an Association Response frame.

40. Define Piconet

The basic Bluetooth network configuration, called a *piconet*, consists of a master device and up to seven slave devices

41. Differentiate persistent and non persistent CSMA (Nov/Dec 2014)
In 1-persistent CSMA if the medium is busy, the channel will be sensed until it is idle, then it will transmit immediately. This means that collisions are almost guaranteed to occur.

In non-persistent CSMA if the medium is busy, there will be a random delay for retransmission. This reduces the probability of collisions, but wastes the capacity.

42. State the uses of valid transmission timer (Nov/Dec 2014)

The Valid Transmission Timer (TVX) times the period between correct frame transmissions, therefore is a check for faults on the ring. If it expires then a new claim process begins

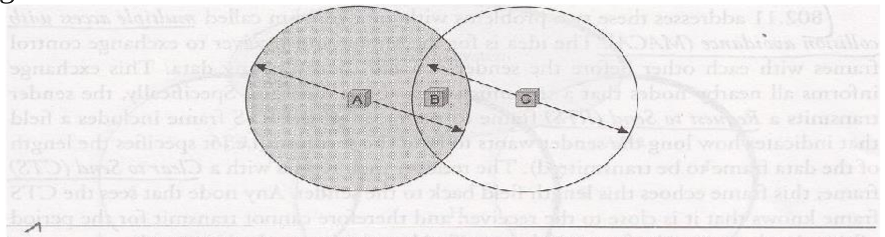
43. What do you understand by CSMA protocol? (May 2015)

Carrier Sense Multiple Access is a probabilistic Media Access control (MAC) protocol in which a node verifies the absence of other traffic before transmitting on a shared transmission medium, such as an electrical bus.

44. List the functions of bridges (May 2015) (May 2017)
1) Pass data frames between networks using MAC address
2) Break up collision domains
3) Forwards all broadcast messages

45. Define hidden node problem (May 2016)

In wireless networking, the hidden node problem or hidden terminal problem occurs when a node is visible from a wireless access point (AP), but not from other nodes communicating with that AP.



The hidden node problem. Although A and C are hidden from each other, their signals can collide at B. (B's reach is not shown.)

46. What is scatternet? (Nov 2016)

The bluetooth network consisting of one or more piconets is known as scatternet. The devices in one piconet type may function as master or slave in another piconet type of the same csatternet

47. Identify the class of the following IP address: (a) 110.34.56.45 (b) 212.208.63.23 (Nov 2015)

110.34.56.45	- Class A
212.208.63.23	- Class C

48. What is fragmentation and reassembly?

IP fragmentation is an Internet Protocol (IP) process that breaks datagrams into smaller pieces (fragments), so that packets may be formed that can pass through a link with a smaller maximum transaction unit (MTU) than the original datagram size. The fragments are reassembled by the receiving host.

49. When is ICMP redirect message used? (May 2017)

The ICMP Redirect message is generated to inform a local host that it should use a different next hop router for a certain class of traffic

50. Highlights the characteristics of datagram networks (Dec 2017)

A datagram has the following characteristics: Data is transmitted from source to destination without guarantee of delivery. Data is frequently divided into smaller pieces and transmitted without a defined route or guaranteed order of delivery.

51. What are the services offered by data link layer?

Framing, Flow control, Error control, Congestion control

52. Define bit stuffing. Give example (MAY 2011) (May 2017)

Bit stuffing is the insertion of one or more bits into a transmission unit as a way to provide signaling information to a receiver. The receiver knows how to detect and remove or disregard the stuffed bits.

e.g, Sending side - 011111**0**10

53. Define character stuffing

The problem with the sentinel approach is that the ETX character might appear in the data portion of the frame. BISYNC overcomes this problem by "escaping" the ETX character by preceding it with a DLE (data-link-escape) character whenever it appears in the body of a frame; the DLE character is also escaped (by preceding it with an extra DLE) in the frame body. This approach is called character stuffing

PART B

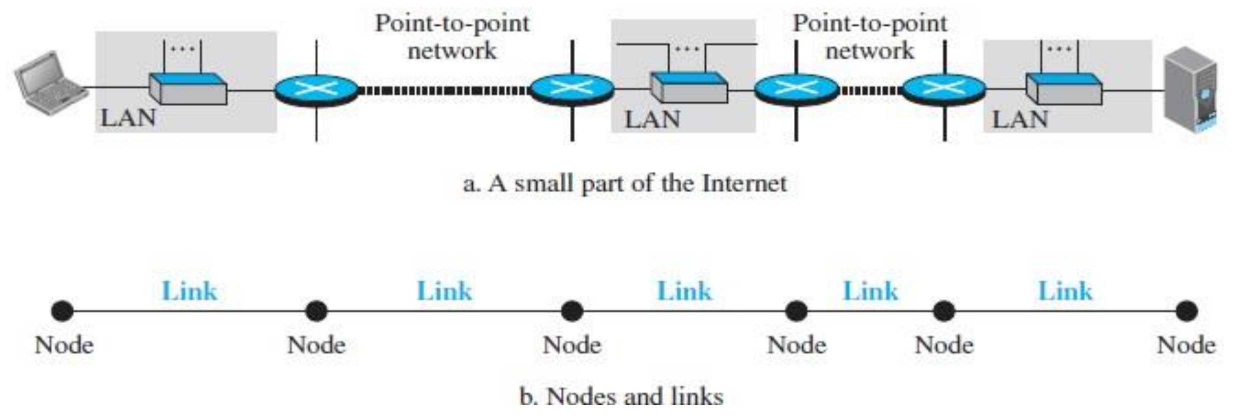
1. Discuss the services offered by data link layer

Introduction

Nodes and Links

Communication at the data-link layer is node-to-node. A data unit from one point in the Internet needs to pass through many networks (LANs and WANs) to reach another point. Theses LANs and WANs are connected by routers. It is customary to refer to the two end hosts and the routers as ***nodes*** and the networks in between as ***links.*** Figure 9.2 is a simple representation of links and nodes when the path of the data unit is only six nodes.

Figure 9.2 *Nodes and Links*



Services

The data-link layer is located between the physical and the network layers. The datalink layer provides services to the network layer; it receives services from the physical layer. Let us discuss services provided by the data-link layer.

The duty scope of the data-link layer is node-to-node. When a packet is travelling in the Internet, the data-link layer of a node (host or router) is responsible for delivering a datagram to the next node in the path. For this purpose, the data-link layer of the sending node needs to encapsulate the datagram received from the network in a frame, and the data-link layer of the receiving node needs to decapsulate the datagram from the frame.

- ***Framing***

Definitely, the first service provided by the data-link layer is **framing**. The data-link layer at each node needs to encapsulate the datagram (packet received from the network layer) in a **frame** before sending it to the next node. The node also needs to decapsulate the datagram from the frame received on the logical channel.

- ***Flow Control***

Flow control refers to a set of procedures used to restrict the amount of data. The sender can send before waiting for acknowledgment.

- ***Error Control***

Error control is used for detecting and retransmitting damaged or lost frames and to prevent duplication of frames. This is achieved through a trailer added at the end of the frame.

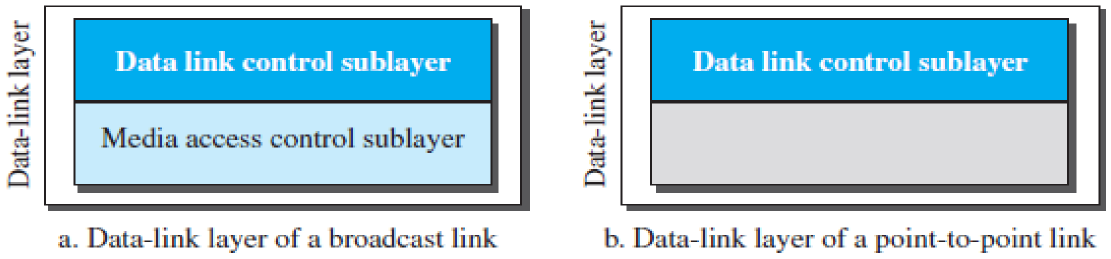
- ***Congestion Control***

It involves preventing too much data from being injected into the network, thereby causing switches or links to become overloaded. Thus flow control is an end to an end issue, while congestion control is concerned with how hosts and networks interact

Two Sublayers

To better understand the functionality of and the services provided by the link layer, we can divide the data-link layer into two sublayers: **data link control (DLC)** and **media access control (MAC).**

Figure 9.4 *Dividing the data-link layer into two sublayers*


Data-link layer
Data link control sublayer
Media access control sublayer
a. Data-link layer of a broadcast link
Data-link layer
Data link control sublayer
b. Data-link layer of a point-to-point link

2. Discuss in detail about link layer addressing

A *link-layer address* is sometimes called a *link address*, sometimes a *physical address*, and sometimes a *MAC address*

Since a link is controlled at the data-link layer, the addresses need to belong to the data-link layer. When a datagram passes from the network layer to the data-link layer, the datagram will be encapsulated in a frame and two data-link addresses are added to the frame header. These two addresses are changed every time the frame moves from one link to another

Three Types of addresses

- ***Unicast Address***

Each host or each interface of a router is assigned a unicast address. Unicasting means one-to-one communication. A frame with a unicast address destination is destined only for one entity in the link.

- ***Multicast Address***

Some link-layer protocols define multicast addresses. Multicasting means one-to-many communication. However, the jurisdiction is local (inside the link).

- ***Broadcast Address***

Some link-layer protocols define a broadcast address. Broadcasting means one-to-all communication. A frame with a destination broadcast address is sent to all entities in the link.

Address Resolution Protocol (ARP)

Introduction

- An internet consists of various types of networks and the connecting devices like routers.
- A packet starts from the source host, passes through many physical networks and finally reaches the destination host.
- At the network level, the hosts and routers are recognized by their IP addresses.

IP address

- An IP address is an internetwork address. It is a universally unique address.
- Every protocol involved in internetworking requires IP addresses.

MAC address

- The packets from source to destination hosts pass through physical networks.
- At the physical level the IP address is not useful but the hosts and routers are recognized by their MAC addresses.
- A MAC address is a local address. It is unique locally but it is not unique universally.
- The IP and MAC address are two different identifiers and both of them are needed
- Deliver a packet to a host or a router, we require two levels of addressing namely IP addressing and MAC addressing.
- Most importantly we should be able to map the IP address into a corresponding MAC address.

Mapping of IP address into a MAC address

- We have seen the need of mapping an IP address into a MAC address.
- Two types of mapping 1) Static mapping and 2) Dynamic mapping

Static Mapping

- In static mapping a table is created and stored in each machine. This table associates an IP address with a MAC address.
- If a machine knows the IP address of another machine then it can search for the corresponding MAC address in its table.
- The limitation of static mapping is that the MAC addresses can change.
- To implement static mapping, the static mapping table needs to be updated periodically.

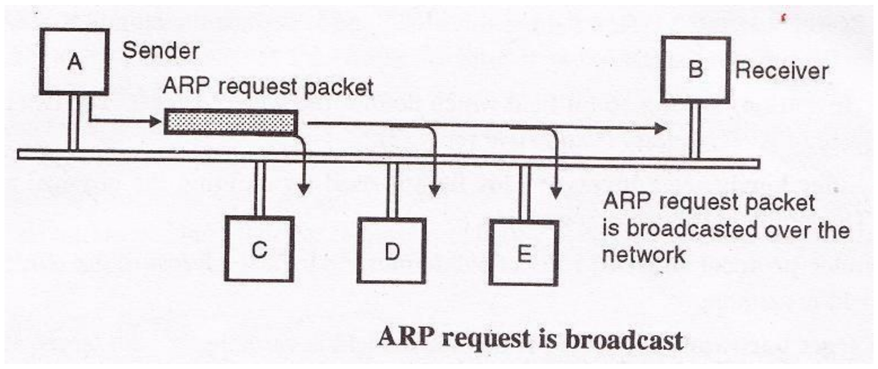
Dynamic mapping

- In dynamic mapping technique a protocol is used for finding the other address when one type of address is known.
- There are two type of dynamic mapping available.
  - Address Resolution Protocol (ARP)
  - Reverse Address Resolution Protocol (RARP)
- The ARP maps IP address to a MAC address whereas the RARP maps a MAC address to an IP address.

ARP Operation

- ARP is used for associating an IP address to its MAC address.
- For a LAN, each device has its own physical or station address as its identification. This address is imprinted on the NIC.
- Find the MAC address:

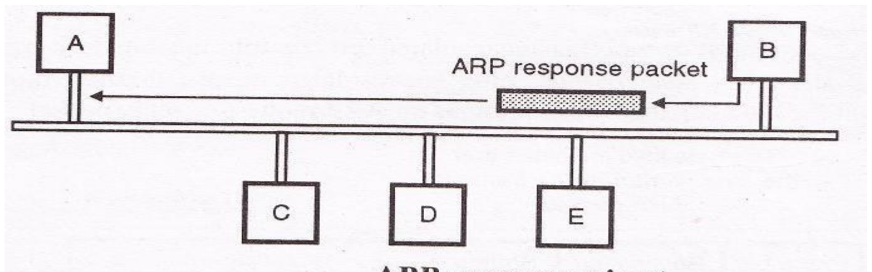
- When a router or a host needs to find the MAC address of another host or network the sequence of events taking place is as follows:
  a. The router or a host A who wants to find the MAC address of some other router, sends an ARP request packet. This packet consists of IP and MAC addresses of the sender A and the IP address of the receiver (B).
  b. This request packet is broadcasted over the network as shown the figure.
  c.



ARP request is broadcast

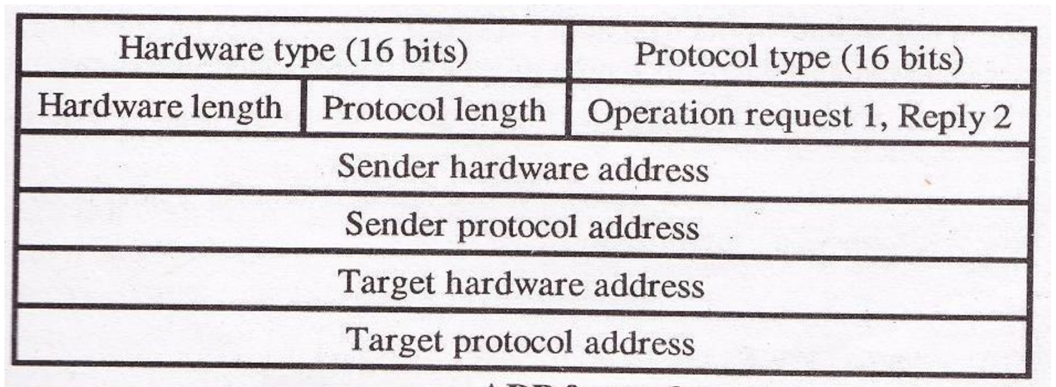
  d. Every host and router on the network receives and processes the ARP request packet. But only the intended receiver (B) recognizes its IP address in the request packet and sends back an ARP response packet.

  e. The ARP response packet contains the IP and physical addresses of the receiver (B). This packet is delivered only to A (unicast) using A’s physical address in the ARP request packet. This is shown in the following figure.



ARP response unicast

ARP Packet Format:



Hardware type (16 bits)		Protocol type (16 bits)
Hardware length	Protocol length	Operation request 1, Reply 2
Sender hardware address		
Sender protocol address		
Target hardware address		
Target protocol address		

ARP frame format

HTYPE (Hardware type):
This 16 bit field defines the type of network on which is ARP is being run. ARP can run on any physical network.

PTYPE (Protocol type):
This 16 bit field is used to define the protocol using ARP. Note that ARP can be used with any higher-level protocol such as IPv4.

HLEN (Hardware length):
It is an 8 bit field which is used for defining the length of the physical address in bytes. For example, this value is 6 for Ethernet.

PLEN (Protocol length):
This field is 8 bit long and it defines the length of the IP address in bytes. For IPv4 this value is 4.

OPER (Operation):
It is a 16 bit field which defines the type of packet. The two possible types of packets are:
ARP request (1) and ARP reply (2).

SHA (Sender Hardware Address):
This field is used for defining the physical address of the sender. The length of this field is variable.

SPA (Sender Protocol address):
This field defines the logical address of the sender. The length of this field is variable.

THA (Target hardware address):
It defines the physical address of the target. It is a variable length field. For the ARP request packet, this field contains all zeros because the sender does not know the receivers physical address.

TPA (Target Protocol address):
This field defines the logical address of the target. It is a variable length field.

3. Explain in detail about DLC services with HDLC & PPP

The **data link control (DLC)** deals with procedures for communication between two adjacent nodes—node-to-node communication—no matter whether the link is dedicated or broadcast. Data link control functions include *framing* and *flow and error control*

Framing

To transmit frames over the node it is necessary to mention start and end of each frame. There are three techniques to solve this frame

- ➢ Byte-Oriented Protocols (BISYNC, PPP, DDCMP)
- ➢ Bit-Oriented Protocols (HDLC)
- ➢ Clock-Based Framing (SONET)

Byte Oriented protocols

In this, view each frame as a collection of bytes (characters) rather than a collection of bits. Such a byte-oriented approach is exemplified by the BISYNC (Binary Synchronous Communication) protocol and the DDCMP (Digital Data Communication Message Protocol)
Sentinel Approach

The BISYNC protocol illustrates the sentinel approach to framing; its frame format is

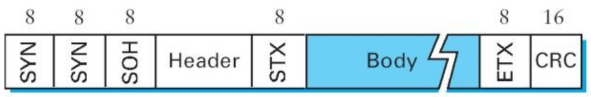


Fig: BISYNC Frame format

- ➢ The beginning of a frame is denoted by sending a special SYN (synchronization) character.
- ➢ The data portion of the frame is then contained between special sentinel characters: STX (start of text) and ETX (end of text).
- ➢ The SOH (start of header) field serves much the same purpose as the STX field.
- ➢ The frame format also includes a field labeled CRC (cyclic redundancy check) that is used to detect transmission errors.

The problem with the sentinel approach is that the ETX character might appear in the data portion of the frame. BISYNC overcomes this problem by "escaping" the ETX character by preceding it with a DLE (data-link-escape) character whenever it appears in the body of a frame; the DLE character is also escaped (by preceding it with an extra DLE) in the frame body. This approach is called character stuffing.

Byte-Counting Approach

The number of bytes contained in a frame can he included as a field in the frame header. DDCMP protocol is used for this approach. The frame format is

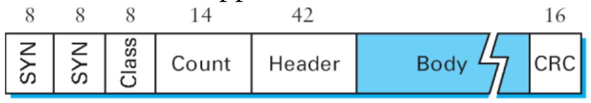
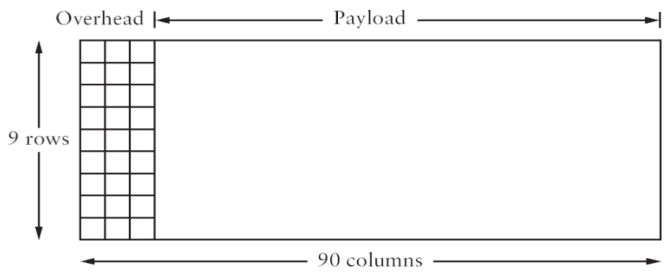


Fig: DDCMP frame format

- ➢ COUNT Field specifies how many bytes are contained in the frame's body.
- ➢ Sometime count field will be corrupted during transmission, so the receiver will accumulate as many bytes as the COUNT field indicates. This is sometimes called a framing error.
- ➢ The receiver will then wait until it sees the next SYN character.

Clock-Based Framing (SONET)

- ➢ Synchronous Optical Network Standard is used for long distance transmission of data over optical network.
- ➢ It supports multiplexing of several low speed links into one high speed links.
- ➢ An STS-1 frame is used in this method.



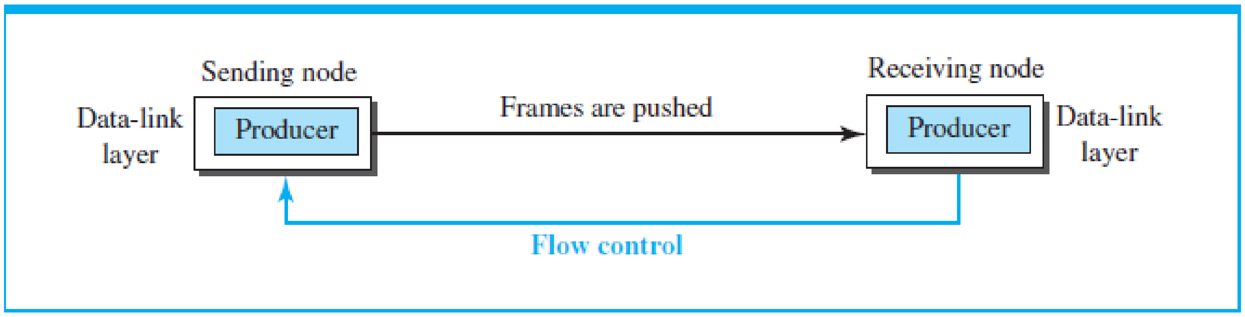
- ➢ It is arranged as nine rows of 90 bytes each, and the first 3 bytes of each row are overhead, with the rest being available for data.
- ➢ The first 2 bytes of the frame contain a special bit pattern, and it is these bytes that enable the receiver to determine where the frame starts.

➢ The receiver looks for the special bit pattern consistently, once in every 810 bytes, since each frame is 9 x 90 = 810 bytes long.

Flow and Error Control

Whenever an entity produces items and another entity consumes them, there should be a balance between production and consumption rates. If the items are produced faster than they can be consumed, the consumer can be overwhelmed and may need to discard some items. If the items are produced more slowly than they can be consumed, the consumer must wait, and the system becomes less efficient. Flow control is related to the first issue. We need to prevent losing the data items at the consumer site.

Figure 11.5 *Flow control at the data-link layer*



The figure shows that the data-link layer at the sending node tries to push frames toward the data-link layer at the receiving node. If the receiving node cannot process and deliver the packet to its network at the same rate that the frames arrive, it becomes overwhelmed with frames. Flow control in this case can be feedback from the receiving node to the sending node to stop or slow down pushing frames.

Buffers

Although flow control can be implemented in several ways, one of the solutions is normally to use two *buffers*; one at the sending data-link layer and the other at the receiving data-link layer. A buffer is a set of memory locations that can hold packets at the sender and receiver. The flow control communication can occur by sending signals from the consumer to the producer. When the buffer of the receiving data-link layer is full, it informs the sending data-link layer to stop pushing frames.

Error Control

Error control at the data-link layer is normally very simple and implemented using one of the following two methods. In both methods, a CRC is added to the frame header by the sender and checked by the receiver.

❑ In the first method, if the frame is corrupted, it is silently discarded; if it is not corrupted, the packet is delivered to the network layer. This method is used mostly in wired LANs such as Ethernet.
❑ In the second method, if the frame is corrupted, it is silently discarded; if it is not corrupted, an acknowledgment is sent (for the purpose of both flow and error control) to the sender.

Connectionless and Connection-Oriented

A DLC protocol can be either connectionless or connection-oriented.

Connectionless Protocol

In a connectionless protocol, frames are sent from one node to the next without any relationship between the frames; each frame is independent. Note that the term *connectionless* here does not mean that there is no physical connection (transmission medium) between the nodes; it means that there is no *connection* between frames. The frames are not numbered and there is no sense of ordering. Most of the data-link protocols for LANs are connectionless protocols.

Connection-Oriented Protocol

In a connection-oriented protocol, a logical connection should first be established between the two nodes (setup phase). After all frames that are somehow related to each other are transmitted (transfer phase), the logical connection is terminated (teardown phase). In this type of communication, the frames are numbered and sent in order. If they are not received in order, the receiver needs to wait until all frames belonging to the same set are received and then deliver them in order to the network layer. Connection oriented protocols are rare in wired LANs, but we can see them in some point-to-point protocols, some wireless LANs, and some WANs.

4. Discuss the protocols used in data link layer .

- Simple Protocol
- Stop-and-Wait Protocol
- Piggybacking

Point-to-Point Protocol (PPP)

The more recent Point-to-Point Protocol (PPP). The format of PPP frame is

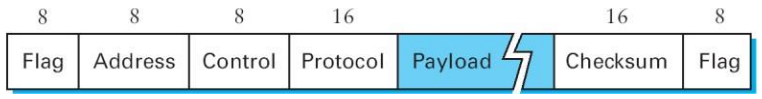


Fig: PPP Frame Format

➢ The Flag field has <u>01111110 as starting sequence</u>.
➢ The Address and Control fields usually contain default values
➢ The Protocol field is used for demultiplexing.
➢ The frame payload size can he negotiated, but it is 1500 bytes by default.
➢ The PPP frame format is unusual in that several of the field sizes are negotiated rather than fixed.
➢ Negotiation is conducted by a protocol called LCP (Link Control Protocol).
➢ LCP sends control messages encapsulated in PPP frames—such messages are denoted by an LCP identifier in the PPP Protocol.

Bit-Oriented Protocols (HDLC)

In this, frames are viewed as collection of bits. High level data link protocol is used. The format is

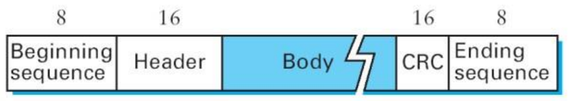


Fig: HDLC Frame Format

➢ HDLC denotes both the beginning and the end of a frame with the distinguished bit sequence 01111110.

➢ This sequence might appear anywhere in the body of the frame, it can be avoided by bit stuffing.

➢ On the sending side, any time five consecutive 1's have been transmitted from the body of the message (i.e., excluding when the sender is trying to transmit the distinguished 01111110 sequence), the sender inserts a 0 before transmitting the next bit.

➢ On the receiving side, five consecutive 1's arrived, the receiver makes its decision based on the next bit it sees (i.e., the bit following the five is).

➢ If the next bit is a 0, it must have been stuffed, and so the receiver removes it. If the next bit is a 1, then one of two things is true, either this is the end-of-frame marker or an error has been introduced into the bit stream.

➢ By looking at the next bit, the receiver can distinguish between these two cases:

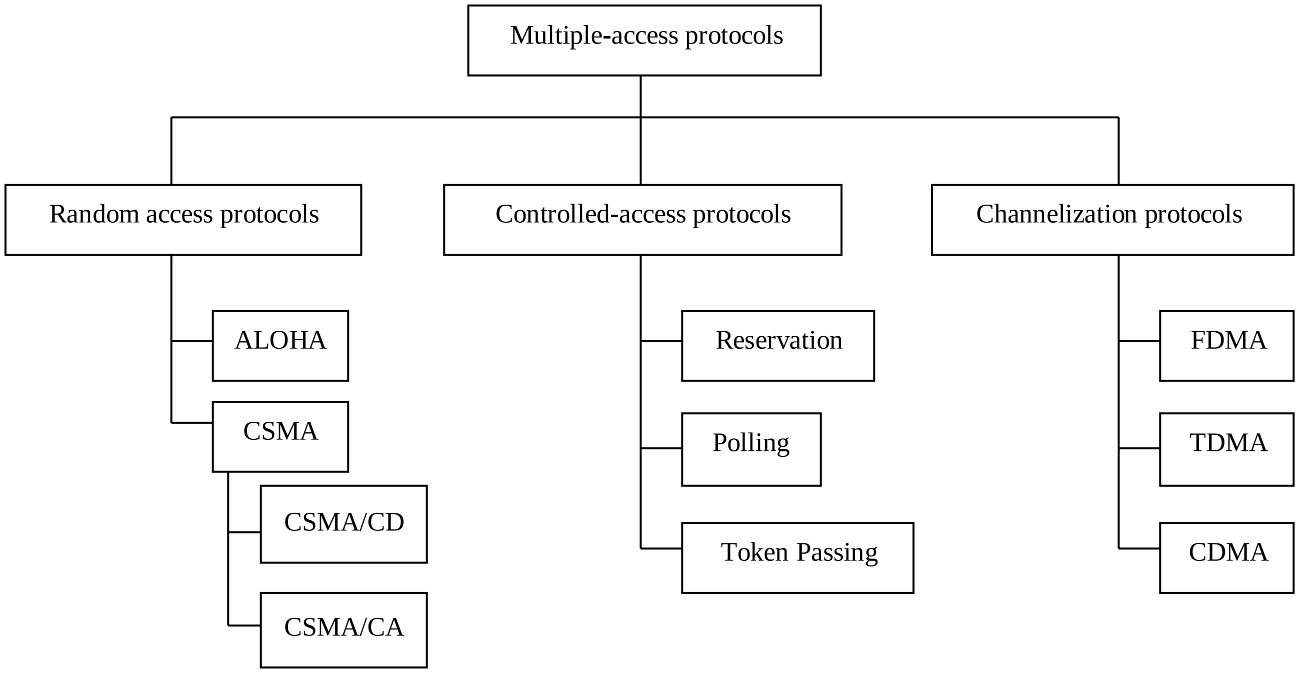
1.If it sees a 0 (i.e., the last eight bits it has looked at are 01111110), then it is the end-of- frame marker.

2.If it sees a 1 (i.e., the last eight bits it has looked at are 01111111), then there must have been an error and the whole frame is discarded.

5. Discuss Media Access Layer Protocols in detail.

- When nodes or stations are connected and use a common link, called a multipoint or broadcast link, we need a multiple-access protocol to coordinate access to the link.

TAXONOMY OF MULTIPLE ACCESS PROTOCOLS



RANDOM ACCESS

- In random access or contention methods, no station is superior to another station and none is assigned to control over another.
- No station permits, or does not permit another station to send.

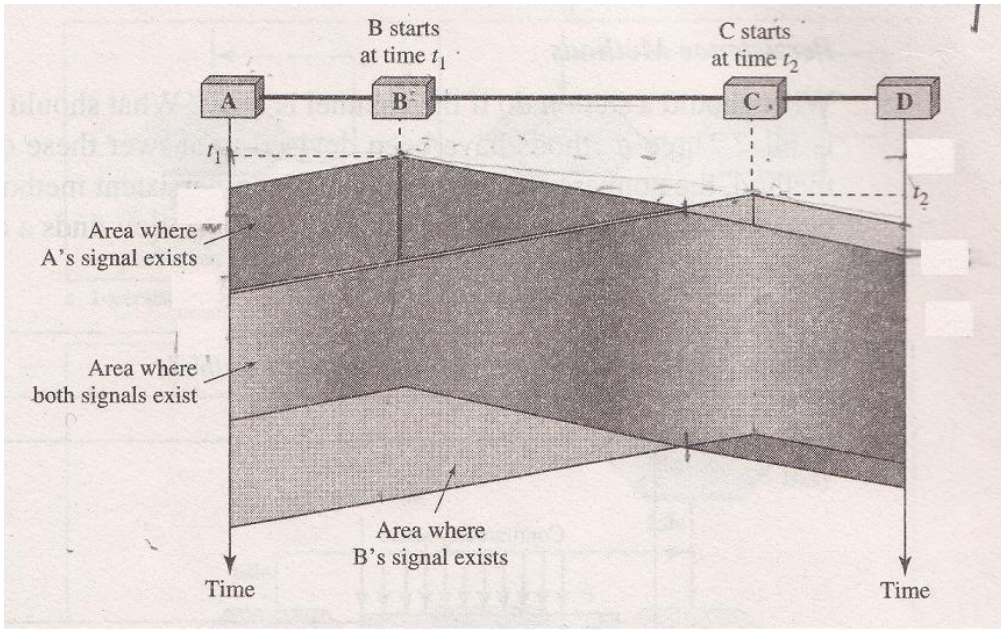
<u>*Two Features of Random Access*</u>

- There is no scheduled time for a station to transmit as the name implies. No rules specify which station should send next.
- Stations fight with one another to access the medium by a method called contention methods.

CSMA - Carrier Sense Multiple Access
CD - ***Collision Detection***
CA - ***Collision Avoidance***

<u>CSMA</u>

- To minimize the chance of collision and increase the performance CSMA method was developed.
- The chance of collision can be reduced if a station senses the medium before trying to use it.
- Carrier sense multiple access (CSMA) requires that each station first listen to the medium (or check the state of the medium) before sending.
- In other words, CSMA is based on the principle “sense before transmit” or “listen before talk.
- CSMA can reduce the possibility of collision, but it cannot eliminate it.

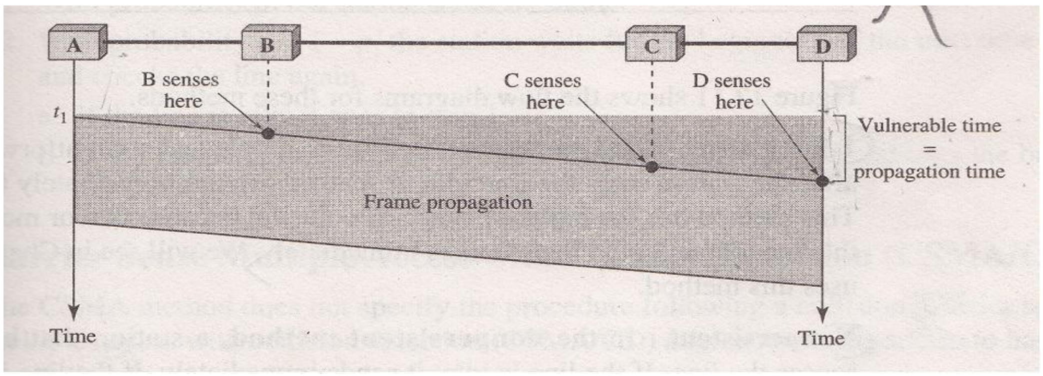

B starts at time t_1
C starts at time t_2
A
B
C
D
t_1
t_2
Area where A’s signal exists
Area where both signals exist
Area where B’s signal exists
Time
Time

<u>Space/time model of the collision in CSMA</u>

- At time t1, station B senses the medium and finds it idle, so it sends a frame.
- At time t2 (t2>t1), station C senses the medium and finds it idle because, at this time, the first bits from station B have not reached station C.
- Station C also sends a frame.
- The two signals collide and both frames are destroyed.

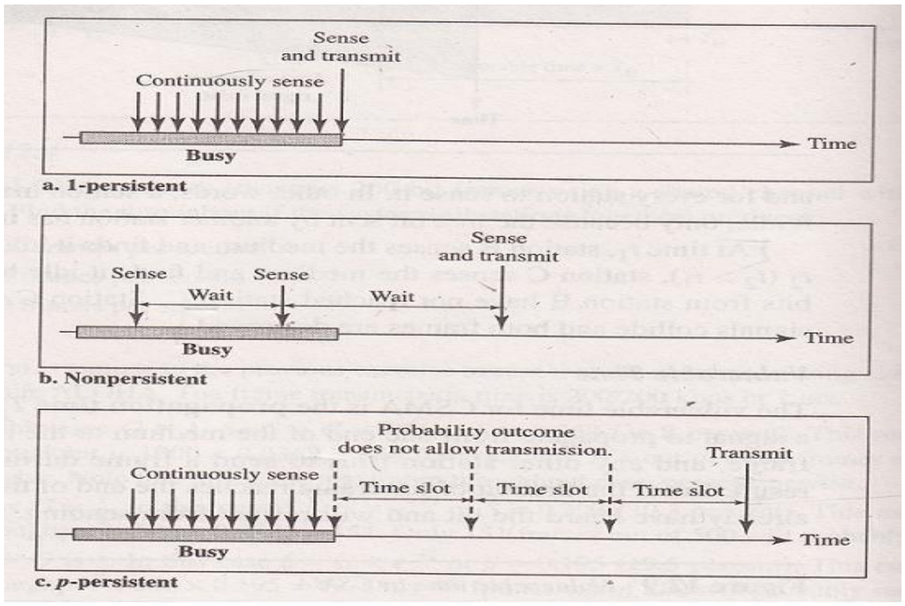
<u>Vulnerable Time</u>

- The vulnerable time for CSMA is the propagation time T_p.
- This is the time needed for a signal to propagate from one end of the medium to the other.
- When a station sends a frame, and any other station tries to send a frame during this time, a collision will result.



Vulnerable Time

<u>Persistence Methods:</u>



Behavior of three persistence methods

1-Persistent:

- The 1-persistent method is simple and straightforward.
- In this method, after the station finds the line idle, it sends its frame immediately (with probability 1).
- This method has the highest chance of collision because two or more stations may find the line idle and send their frames immediately.

Nonpersistent:

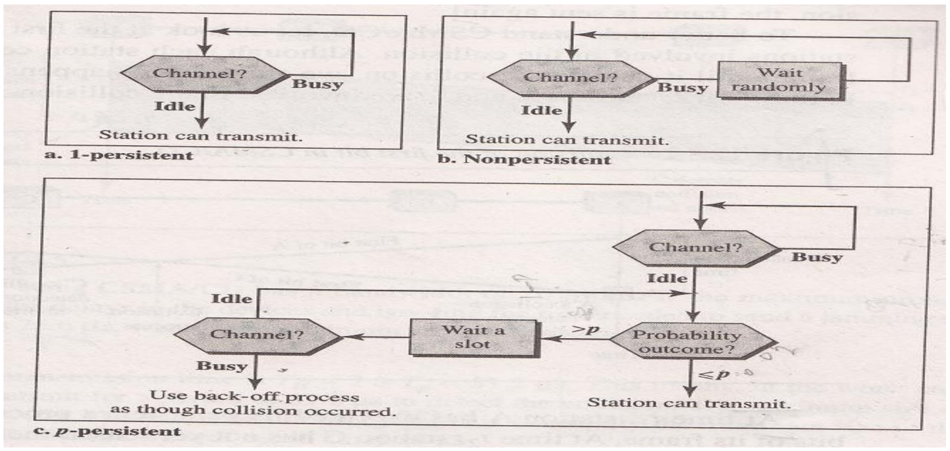
- In the nonpersistent method, a station that has a frame to send senses the line.
- If the line is idle, it sends immediately.
- If the line is not idle, it waits a random amount of time and then senses the line again.
- The nonpersistent approach reduces the chance of collision because it is unlikely that two or more stations will wait the same amount of time and retry to send simultaneously.

P-Persistent:

- The p-persistent method is used if the channel has time slots with slot duration equal to or greater than the maximum propagation time.
- The p-persistent approach combines the advantages of the other two strategies.
- It reduces the chance of collision and improves efficiency.

In this method, after the station finds the line idle it follows these steps:

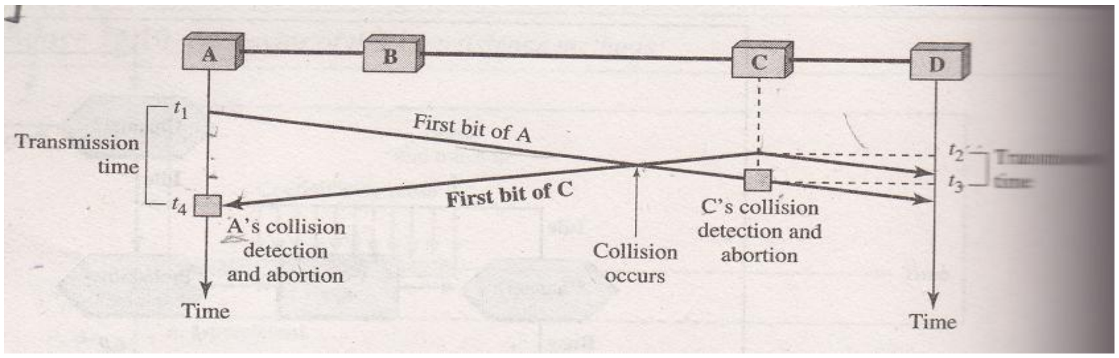
- ✓ With probability p, the station sends its frame.
- ✓ With probability q=1-p, the station waits for the beginning of the next time slot and checks the line again.
  - If the line is idle, it goes to step 1.
  - If the line is busy, it acts as though a collision has occurred and uses the backoff procedure.



CSMA/CD tells the station what to do when a collision is detected. CSMA/CA tries to avoid the collision.

Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

- Carrier sense multiple access with collision detection (CSMA/CD) augments the algorithm to handle the collision.
- A station monitors the medium after it sends a frame to see if the transmission was successful.
- If so, the station is finished. If, however, there is a collision, the frame is sent again.



- Collision of the first bit in CSMA/CD
- At time t1, station A has executed its persistence procedure and starts sending the bits of its frame.
- At time t2, station C has not yet sensed the first bit sent by A.
- Station C executes its persistence procedure and starts sending the bits in its frame, which propagate both to the left and to the right.
- The collision occurs sometime after time t2.
- Station C detects a collision at time t3 when it receives the first bit of A's frame.
- Station C immediately (or after a short time, but we assume immediately) aborts transmission.
- Station A detects collision at time t4 when it receives the first bit of C's frame; it also immediately aborts transmission.
- Looking at the figure, we can see that A transmits for the duration t4 - t1; C transmits for the duration t3 - t2.

Minimum Frame Size:

- For CSMA/CD to work, we need a restriction on the frame size.
- Before sending the last bit of the frame, the sending station must detect a collision, if any, and abort the transmission.

- This is so because the station, once the entire frame is sent, does not keep a copy of the frame and does not monitor the line for collision detection. Therefore, the frame transmission time T_{ff} must be at least two times the maximum propagation time T_p.

Example

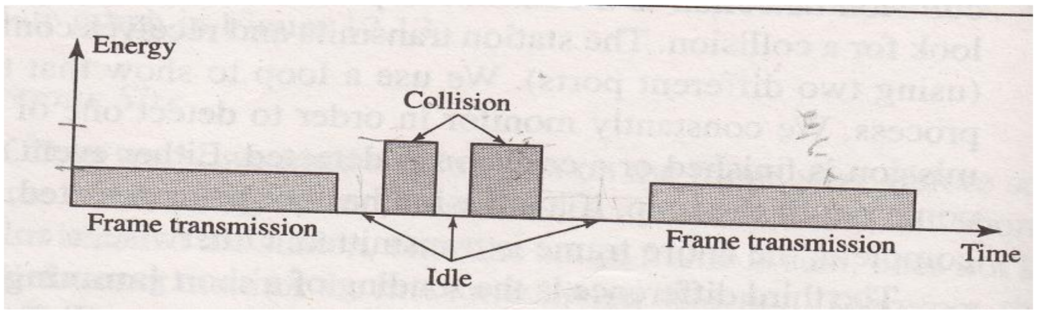
A network using CSMA/CD has a bandwidth of 10 Mbps.

If the maximum propagation time (including the delays in the devices and ignoring the time needed to send a jamming signal) is 25.6μs, what is the minimum size of the frame?

Solution: The frame transmission time is $T_{ff} = 2 X T_p$ = 51.2μs. This means, in the worst case, a station needs to transmit for a period of 51.2μs to detect the collision. The minimum size of the frame is 10 Mbps X 51.2μs = 512 bits or 64 bytes.

Energy Level:

- Level of energy in a channel can have three values: Zero, normal, and abnormal.
- At the zero level, the channel is idle.
- At the normal level, a station has successfully captured the channel and is sending its frame.
- At the abnormal level, there is a collision and the level of the energy is twice the normal level.
- A station that has a frame to send or is sending a frame needs to monitor the energy level to determine if the channel is idle, bust or in collision mode.

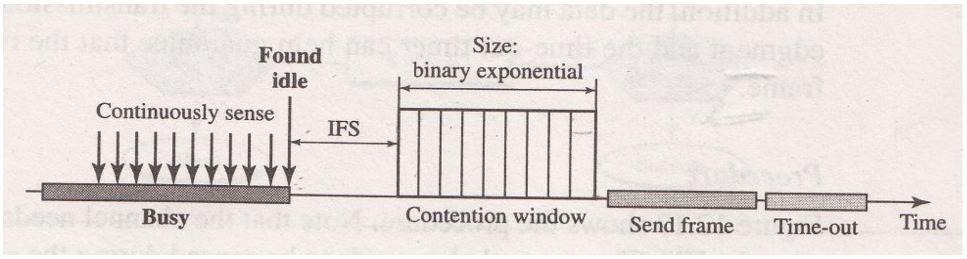


Energy level during transmission, idleness or collision

Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)

- The basic idea behind CSMA/CA is that a station needs to be able to receive while transmitting to detect a collision.
- When there is no collision, the station receives one signal: its own signal.
- When there is a collision, the station receives two signals: its own signal and the signal transmitted by a second station.

- In a wired network, the received signal has almost the same energy as the sent signal because either the length of the cable is short or there are repeaters that amplify the energy between the sender and the receiver.
- This means that in a collision, the detected energy almost doubles.
- In a wireless network, much of the sent energy is lost in transmission.
- The received signal has very little energy.
- Therefore, a collision may add only 5 to 10 percent additional energy.
- This is not useful for effective collision detection.
- To avoid collisions on wireless networks because they cannot be detected carrier sense multiple access with collision avoidance (CSMA/CA) was invented for this network.
- Collisions are avoided through the use of CSMA/CA's three strategies: the inter-frame space, the contention window, and acknowledgements.



Timing in CSMA/CA

Inter-frame Space (IFS):

- First, collisions are avoided by deferring transmission even if the channel is found idle.
- When an idle channel is found; the station does not send immediately.
- It waits for a period of time called the inter-frame space or IFS.
- Even though the channel may appear idle when it is sensed, a distance station may have already started transmitting.
- The distant station's signal has not yet reached this station.
- The IFS time allows the front of the transmitted signal by the distant station to reach this station.
- If after the IFS time the channel is still idle, the station can send, but it still needs to wait a time equal to the contention time.

In CSMA/CA, the IFS can also be used to define the priority of a station or a frame.

Contention Window:

- The contention window is an amount of time divided into slots.
- A station that is ready to send chooses a random number of slots as its wait time.
- The number of slots in the window changes according to the binary exponential back-off strategy.
- This means that it is set to one slot the first time and then doubles each time the station cannot detect an idle channel after the IFS time.
- The contention window is that the station needs to sense the channel after each time slot.
- In CSMA/CA, if the station finds the channel busy, it does not restart the timer of the contention window; it stops the timer and restarts it when the channel becomes idle.

Acknowledgment

- With all these precautions, there still may be a collision resulting in destroyed data.
- In addition, the data may be corrupted during the transmission.
- The positive acknowledgment and the time-out timer can help guarantee that the receiver has received the frame.

NOTE

Exponential backoff:

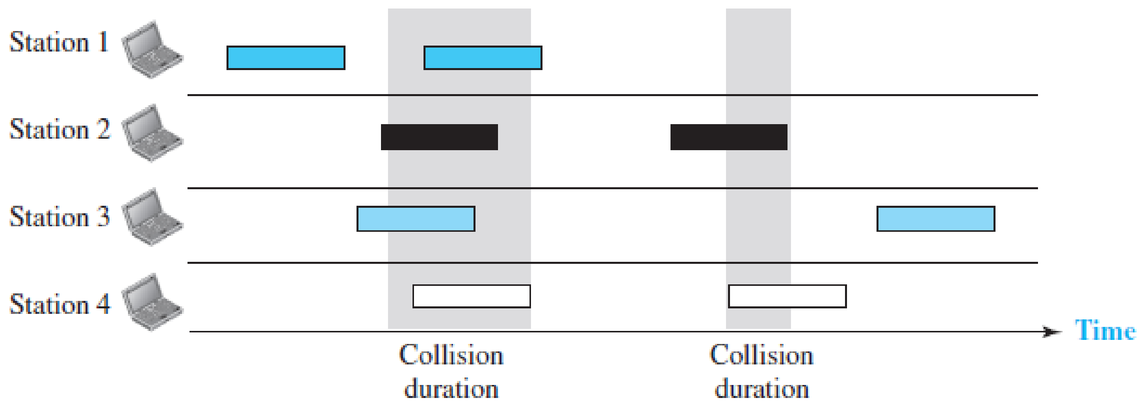
The strategy of doubling the delay interval between each retransmission attempt is a general technique known as **exponential backoff**.

<u>ALOHA</u>

- ***Pure ALOHA***

The original ALOHA protocol is called ***pure ALOHA.*** This is a simple but elegant protocol. The idea is that each station sends a frame whenever it has a frame to send (multiple access). However, since there is only one channel to share, there is the possibility of collision between frames from different stations. Figure 12.2 shows an example of frame collisions in pure ALOHA.

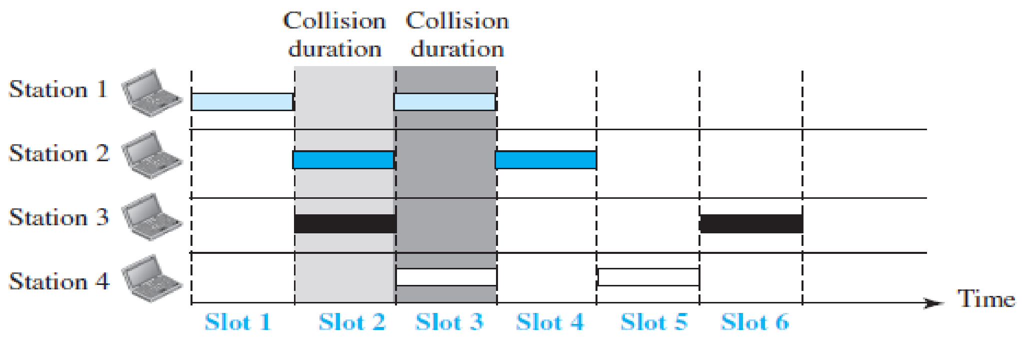
Figure 12.2 *Frames in a pure ALOHA network*



- ***Slotted ALOHA***

Pure ALOHA has a vulnerable time of 2 * *Tfr*. This is so because there is no rule that defines when the station can send. A station may send soon after another station has started or just before another station has finished. Slotted ALOHA was invented to improve the efficiency of pure ALOHA.
In **slotted ALOHA** we divide the time into slots of *Tfr* seconds and force the station to send only at the beginning of the time slot. Figure 12.5 shows an example of frame collisions in slotted ALOHA.

Figure 12.5 *Frames in a slotted ALOHA network*



6. Explain controlled access protocol in media access control

In **controlled access,** the stations consult one another to find which station has the right to send. A station cannot send unless it has been authorized by other stations. We discuss three controlled-access methods

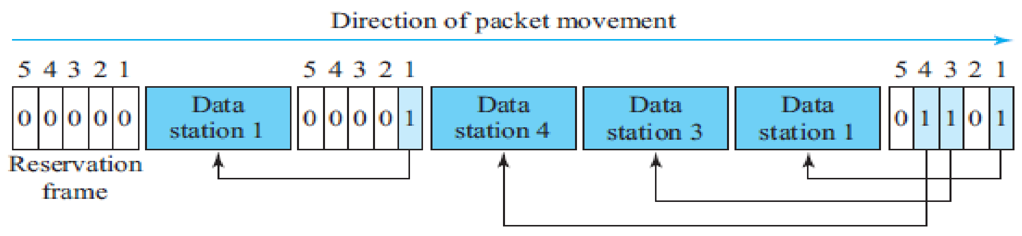
<u>Reservation</u>

In the **reservation** method, a station needs to make a reservation before sending data. Time is divided into intervals. In each interval, a reservation frame precedes the data frames sent in that interval.

If there are *N* stations in the system, there are exactly *N* reservation minislots in the reservation frame. Each minislot belongs to a station. When a station needs to send a data frame, it makes a reservation in its own minislot. The stations that have made reservations can send their data frames after the reservation frame.

Figure 12.18 shows a situation with five stations and a five-minislot reservation frame. In the first interval, only stations 1, 3, and 4 have made reservations. In the second interval, only station 1 has made a reservation

Figure 12.18 *Reservation access method*

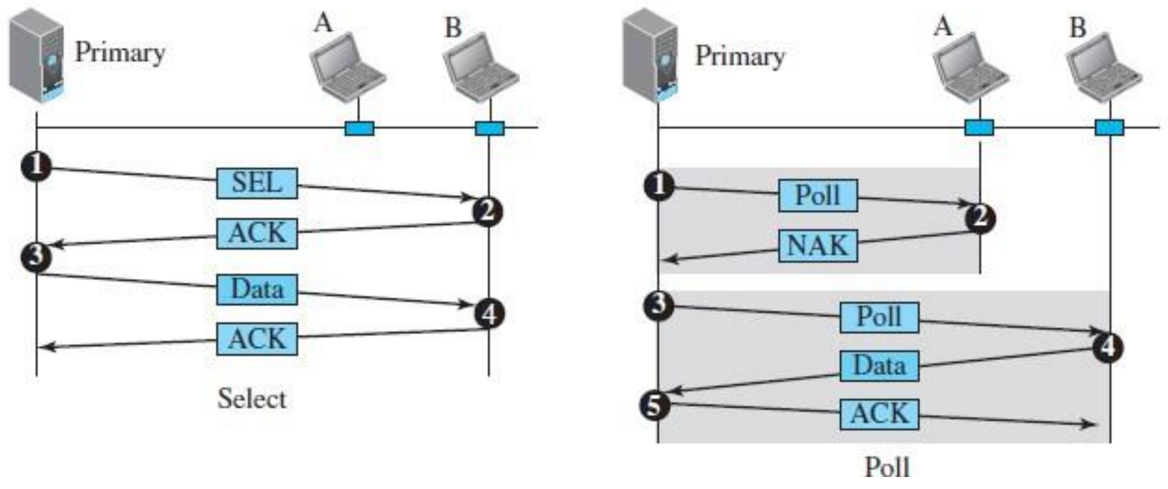


Polling

Polling works with topologies in which one device is designated as a ***primary station*** and the other devices are ***secondary stations.*** All data exchanges must be made through the primary device even when the ultimate destination is a secondary device.

The primary device controls the link; the secondary devices follow its instructions. It is up to the primary device to determine which device is allowed to use the channel at a given time. The primary device, therefore, is always the initiator of a session (see Figure 12.19). This method uses poll and select functions to prevent collisions. However, the drawback is if the primary station fails, the system goes down.

Figure 12.19 *Select and poll functions in polling-access method*



Token Passing

In the **token-passing** method, the stations in a network are organized in a logical ring. In other words, for each station, there is a *predecessor* and a *successor*. The predecessor is the station which is logically before the station in the ring; the successor is the station which is after the station in the ring. The current station is the one that is accessing the channel now. The right to this access has been passed from the predecessor to the current station. The right will be passed to the successor when the current station has no more data to send.

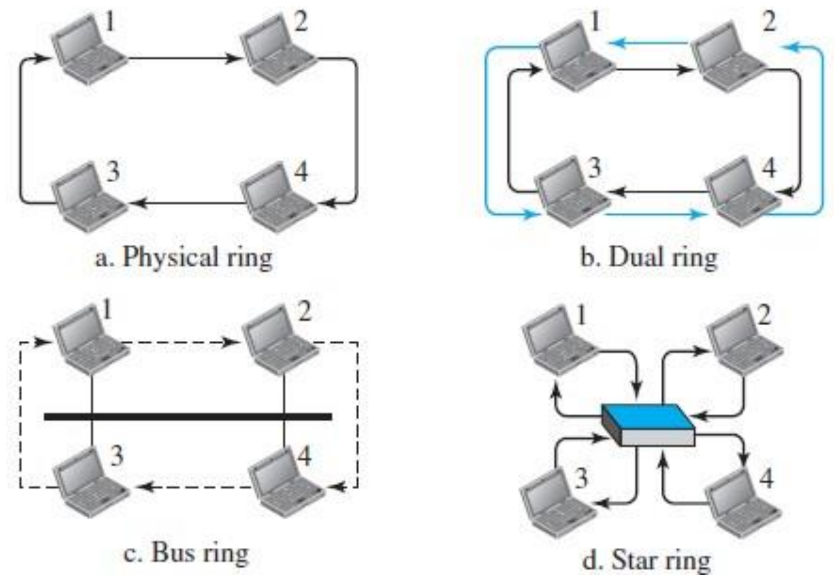
But how is the right to access the channel passed from one station to another? In this method, a special packet called a ***token*** circulates through the ring. The possession of the token gives the station the right to access the channel and send its data. When a station has some data to send, it waits until it receives the token from its predecessor. It then holds the token and sends its data. When the station has no more data to send, it releases the token, passing it to the next logical station in the ring. The station cannot send data until it receives the token again in the next round. In this process, when a station receives the token and has no data to send, it just passes the data to the next station

Token management is needed for this access method. Stations must be limited in the time they can have possession of the token. The token must be monitored to ensure it has not been lost or destroyed. For example, if a station that is holding the token fails, the token will disappear from the network. Another function of token management is to assign priorities to the stations and to the types of data being transmitted. And finally, token management is needed to make low-priority stations release the token to high-priority stations

Logical Ring

In a token-passing network, stations do not have to be physically connected in a ring; the ring can be a logical one. Figure 12.20 shows four different physical topologies that can create a logical ring.

Figure 12.20 *Logical ring and physical topology in token-passing access method*



In the physical ring topology, when a station sends the token to its successor, the token cannot be seen by other stations; the successor is the next one in line. This means that the token does not have to have the address of the next successor. The problem with this topology is that if one of the links—the medium between two adjacent stations—fails, the whole system fails.

The dual ring topology uses a second (auxiliary) ring which operates in the reverse direction compared with the main ring. The second ring is for emergencies only (such as a spare tire for a car). If one of the links in the main ring fails, the system automatically combines the two rings to form a temporary ring. After the failed link is restored, the auxiliary ring becomes idle again. Note that for this topology to work, each station needs to have two transmitter ports and two receiver ports. The high-speed Token Ring networks called *FDDI (Fiber Distributed Data Interface)* and *CDDI (Copper Distributed Data Interface)* use this topology.

In the bus ring topology, also called a token bus, the stations are connected to a single cable called a *bus*. They, however, make a logical ring, because each station knows the address of its successor (and also predecessor for token management purposes). When a station has finished sending its data, it releases the token and inserts the address of its successor in the token. Only the station with the address matching the destination address of the token gets the token to access the shared media. The Token Bus LAN, standardized by IEEE, uses this topology.

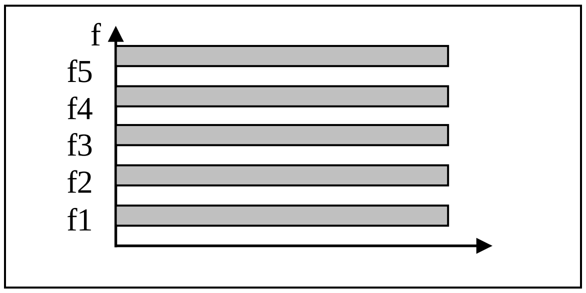
In a star ring topology, the physical topology is a star. There is a hub, however, that acts as the connector. The wiring inside the hub makes the ring; the stations are connected to this ring through the two wire connections. This topology makes the network less prone to failure because if a link goes down, it will be bypassed by the hub and the rest of the stations can operate. Also adding and removing stations from the ring is easier. This topology is still used in the Token Ring LAN designed by IBM.

7. Discuss channelization protocol in media access control

Channelization (or *channel partition,* as it is sometimes called) is a multiple-access method in which the available bandwidth of a link is shared in time, frequency, or through code, among different stations. In this section, we discuss three channelization protocols: FDMA, TDMA, and CDMA.

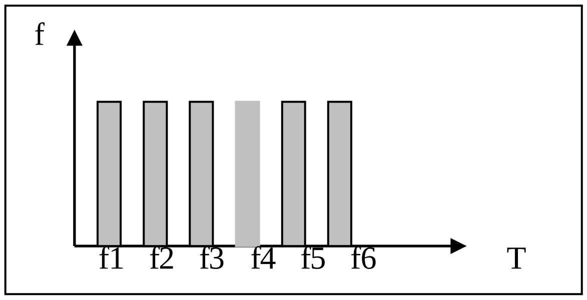
Frequency Division Multiple Access

- In frequency-division multiple access (FDMA), the available bandwidth is divided into frequency bands.
- Each station is allocated a band to send its data.
- In this method when any one frequency level is kept idle and another is used frequently leads to inefficiency.



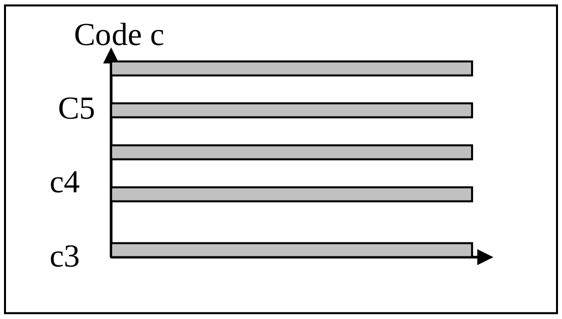
Time Division Multiple Access

- In time-division multiple access (TDMA), the stations share the bandwidth of the channel in time.
- Each station is allocated a time slot during which it can send data.
- The main problem with TDMA lies in achieving synchronization between the different stations.
- Each station needs to know the beginning of its slot and the location of its slot.



Code Division Multiple Access

- CDMA differs from FDMA because only one channel occupies the entire bandwidth of the link.
- It differs from TDMA because all stations can send data at the same time without timesharing.
- CDMA simply means communication with different codes.
- CDMA is based on coding theory. Each station is assigned a code, which is a sequence of numbers called chips.
- Chips will be added with the original data and it can be transmitted through same medium.



8. Explain in detail about wired LAN - Ethernet (IEEE 802.3) and its frame format (OR) Explain the physical properties of Ethernet 802.3 with necessary diagram (NOV 2014)(May,Nov 2015 & 2016)

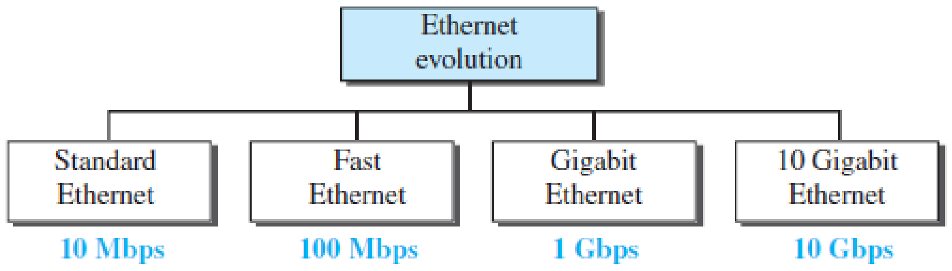
Introduction:

- The IEEE 802.3 standards committee developed a widely used LAN standard called Ethernet, which covers both the MAC layer and the physical layer.
- The Ethernet is a multiple-access network, meaning that a set of nodes send and receive frames over a shared link.
- The IEEE 802.3 standard uses CSMA for controlling media access and the 1-persistent algorithm explained earlier, although the lost time owing to collisions is very small.
- Also, IEEE 802.3 uses a back-off scheme known as binary exponential backoff.
- The use of random backoff minimizes subsequent collisions.
- This back-off scheme requires a random delay to be doubled after each retransmission.
- The user drops the frame after 16 retries.
- The combination of the 1-persistent scheme and binary exponential backoff results in an efficient scheme.
- The Ethernet versions have different data rates.

- Version 1000BaseSX, carrying 1 Gb/s, and 10GBase-T, carrying 10 Gb/s, hold the most promise for the future of high-speed LAN development

Ethernet Evolution

Figure 13.2 *Ethernet evolution through four generations*



Physical properties:

- An Ethernet segment is implemented on a <u>coaxial cable</u> of up to 500m.
- This cable is similar to the type used for cable TV, except that it typically has an impedance of 50 ohms instead of cable TV's 75 ohms.
- Hosts connect to an Ethernet segment by tapping into it; taps must be at least 2.5 m apart.
- A <u>transceiver</u> – a small device directly attached to the tap – detects when the line is idle and drives the signal when the host is transmitting.
- It also receives incoming signals.

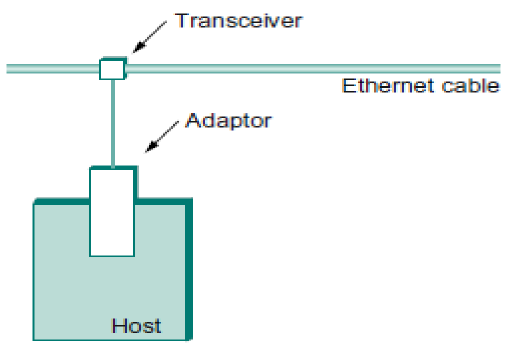


FIGURE 2.22 Ethernet transceiver and adaptor.

- The transceiver is, in turn, connected to an <u>Ethernet adaptor</u>, which is plugged into the host.

- Multiple Ethernet segments can be joined together by repeaters.
- A repeater is a device that forwards digital signals, much like an amplifier forwards analog signals.
- However, no more than four repeaters may be positioned between any pair of hosts, meaning that an Ethernet has a total reach of only 2,500 m.

- Rather than using a 50-ohm coax cable, an Ethernet can be constructed from a thinner cable known as 10Base2; the original cable is called 10Base5 (the two cables are commonly called thin-net and thick-net, respectively).
- The "10" in 10Base2 means that the network operates at 10 Mbps, "Base" refers to the fact that the cable is used in a baseband system, and the "2" means that a given segment can be no longer than 200 m.
- Today, a third cable technology is predominantly used, called 10BaseT, where the "T" stands for twisted pair.
- A 10BaseT segment is usually limited to less than 100 m in length.
- Data transmitted by any one host on the Ethernet reaches all the other hosts.
- This is the good news.
- The bad news is that all these hosts are competing for access to the same link, and as a consequence, they are said to be in the same collision domain.

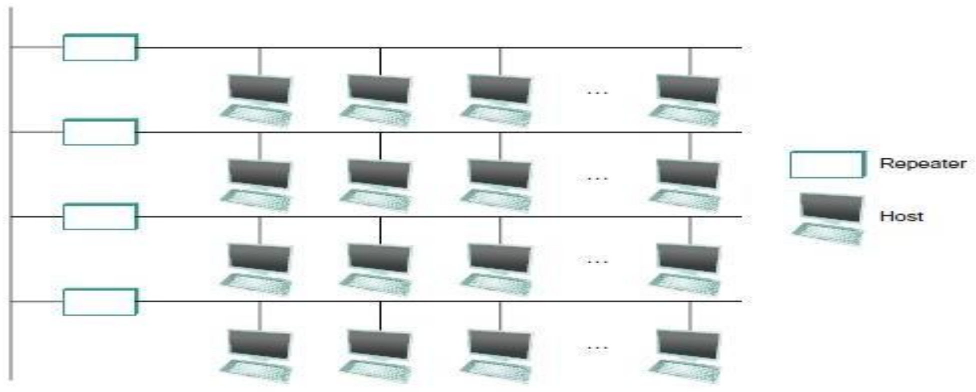


FIGURE 2.23 Ethernet repeater.

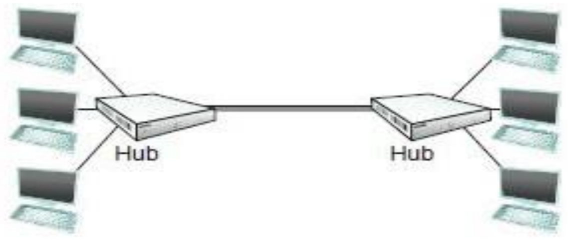


FIGURE 2.24 Ethernet hub.

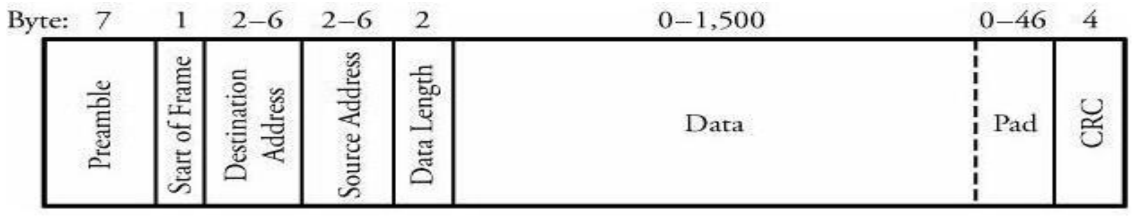
Access Protocol:

- The algorithm that controls access to the shared Ethernet link.
- This algorithm is commonly called the Ethernet's media access control (MAC). It is typically implemented in hardware on the network adaptor.

Frame Format:

A brief description of the frame fields follows and is shown in the below figure.

- Preamble is 7 bytes and consists of a pattern of alternating 0s and 1s. This field is used to provide bit synchronization.
- Start of frame consists of a 10101011 pattern and indicates the start of the frame to the receiver.
- Destination address specifies the destination MAC address.
- Source address specifies the source MAC address.
- Length/Type specifies the frame size, in bytes. The maximum Ethernet frame
- size is 1,518 bytes.
- LLC data is data from the LLC layer.
- Pad is used to increase the frame length to the value required for collision detection to work.
- Frame check sequence is 32-bit CRC for error checking.



Ethernet IEEE 802.3 LAN frame

Address

- Each host on an Ethernet – has a unique Ethernet address.
- Ethernet addresses are typically printed in a form humans can read as a sequence of six numbers separated by colons.
- Each number corresponds to 1 byte of the 6-byte address and is given by a pair of hexadecimal digits, one for each of the 4-bit nibbles in the byte; leading 0s are dropped.

- For example, 8:0:2b:e4:b1:2 is the human-readable representation of Ethernet address as follows

  00001000 00000000 00101011 11100100 10110001 00000010
- Each frame transmitted on an Ethernet is received by every **adaptor** connected to that Ethernet.
- Each **adaptor** recognizes those frames addressed to its address and passes only those frames on to the host.

An Ethernet adaptor receives all frames and accepts

- Frames addressed to its own address
- Frames addressed to the broadcast address
- Frames addressed to a multicast address, if it has been instructed to listen to that address
- All frames, if it has been placed in promiscuous mode.

It passes to the host only the frames that it accepts.

Transmitter algorithm:

1-Persistent:

- The 1-persistent method is simple and straightforward.
- In this method, after the station finds the line idle, it sends its frame immediately (with probability 1).
- This method has the highest chance of collision because two or more stations may find the line idle and send their frames immediately.

P-Persistent:

- The p-persistent method is used if the channel has time slots with a slot duration equal to or greater than the maximum propagation time.
- The p-persistent approach combines the advantages of the other two strategies.
- It reduces the chance of collision and improves efficiency.

In this method, after the station finds the line idle it follows these steps:

- With probability p, the station sends its frame.
- With probability q=1-p, the station waits for the beginning of the next time slot and checks the line again.
    - If the line is idle, it goes to step 1.
    - If the line is busy, it acts as though a collision has occurred and uses the backoff procedure.

9. Discuss the functioning (Key requirements) of wireless LAN in detail. (May 2015, Nov 2015) May 2016

Wireless technologies differ from wired links in some important ways, while at the same time sharing many common properties. Like wired links, issues of bit errors are of great concern—typically even more so due to the unpredictable noise environment of most wireless links. Framing and reliability also have to be addressed.

Unlike wired links, power is a big issue for wireless, especially because wireless links are often used by small mobile devices (like phones and sensors) that have limited access to power (e.g., a small battery). Furthermore, you can't go blasting away at arbitrarily high power with a radio transmitter—there are concerns about interference with other devices and usually regulations about how much power a device may emit at any given frequency.

Table 2.4 Overview of Leading Wireless Technologies

	Bluetooth (802.15.1)	Wi-Fi (802.11)	3G Cellular
Typical link length	10 m	100 m	Tens of kilometers
Typical data rate	2 Mbps (shared)	54 Mbps (shared)	Hundreds of kbps (per connection)
Typical use	Link a peripheral to a computer	Link a computer to a wired base	Link a mobile phone to a wired tower
Wired technology analogy	USB	Ethernet	DSL

Introduction IEEE has defined the specification for the wireless LAN called IEEE 802.11, which covers the physical and Data Link Layers.

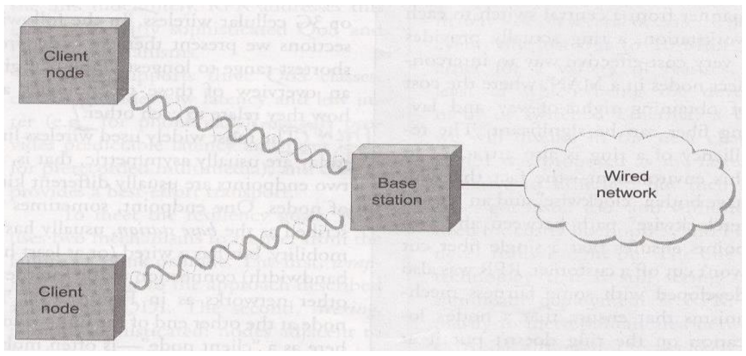
Architecture

IEEE 802.11 standard defines 2 kinds of services.

1. The basic service set (BSS)
2. The extended service set (ESS)

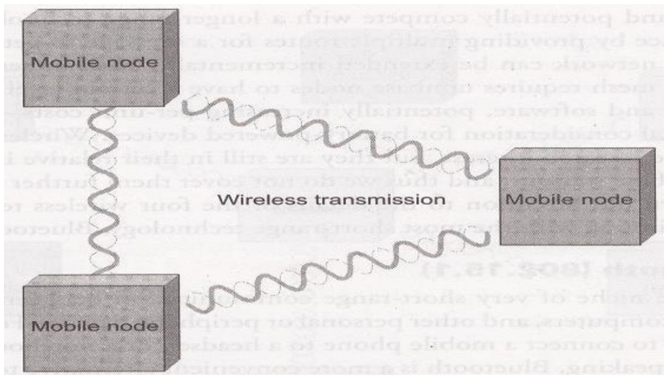
1) Basic Service Set (BSS):

A BSS is made of stationary (immobile) or mobile wireless stations and a possible central base station known as the access point AP.



Mesh or ad hoc network

The BSS without an AP is stand alone network and cannot send data to other BSS. It is called an ad hoc architecture.



2) Extended Service Set (ESS):

An ESS is made up of two or more BSS with AP. The BSS are connected through a distribution system, which is usually a wired LAN.

An ESS uses two types of stations mobile and stationary. The mobile stations are normal stations inside a BSS. The stationary stations are AP stations that are part of the wired LAN. When BSS are connected, the network is called an infrastructure network. In this the stations within reach of one another can communicate without the use of an AP. But communication between two stations in two different BSS usually occurs via two AP's.

Station Types

Three qualitatively different levels of mobility in a wireless LAN.

1. No transmission
2. BSS transition
3. ESS transition

1) No transmission:

The first level is no mobility, such as when a receiver must be in a fixed location to receive a directional transmission form the base station of a single BSS.

2) BSS transition:

It is defined as a station movement from one BSS to another BSS within the same ESS (Bluetooth).

3) ESS transition

It is defined as a station movement from a BSS in one ESS to a BSS with in another ESS. The third level is mobility between bases, as is the case with cell phones and Wi-Fi.

10. Discuss IEEE 802.11 (or) WI-FI in detail (or) MAC layer functions in IEEE802.11 (May 2015, 2016, 2017)(Dec 2017)

802.11 is designed for use in a limited geographical area (homes, office buildings, campuses), and its primary challenge is to mediate access to a shared communication medium—in this case, signals propagating through space.

Physical properties

IEEE 802.11 defines the specification for the conversion of bits to a signal in the physical layer. The IEEE 802.11 physical layer is of four types.

1. **Frequency-hopping spread spectrum (FHSS):** It is a method in which the sender sends one carrier frequency for a short amount of time, and then hops to another carrier frequency for the same amount of time, hops again to still another same amount of time and so on.

This technique makes use of 79 channels. FHSS operates in the 2.4 GHz ISM band and supports data rates of 1 Mb/s to 2 Mb/s.

   1. If the band width of the original signal is B, the allocated spread spectrum bandwidth is N X B.
   2. The amount of time spent at each sub band is called the dwell time.

2. **Direct-sequence spread spectrum (DSSS):** It uses seven channels, each supporting data rates of 1 Mb/s to 2 Mb/s. The operating frequency range is 2.4 GHz ISM band.

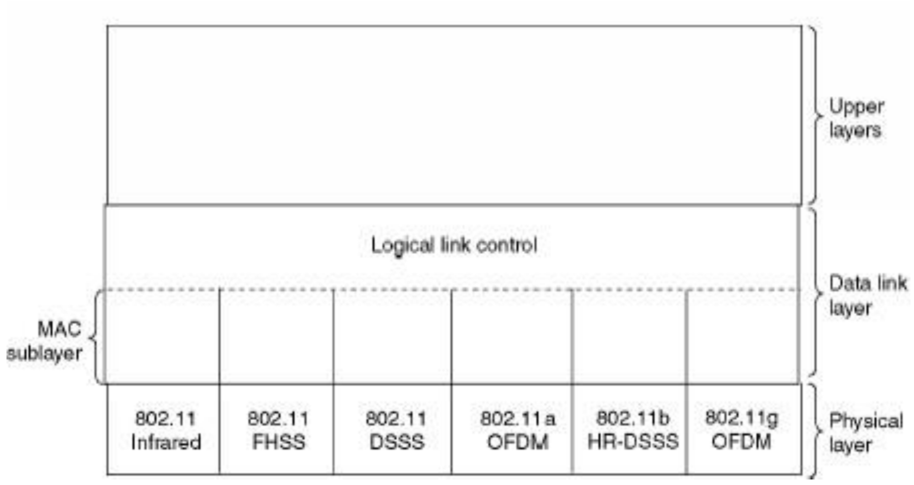
In DSSS each bit by the sender is a replaced by the sequence of bits called chip code. To avoid buffering, the time needed to send one chip code must be the same as the time needed to send one original bit.

3. **IEEE 802.11a**: Orthogonal frequency division multiplexing (OFDM): IEEE 802.11a uses OFDM, which uses 12 orthogonal channels in the 5 GHz range. All the sub bands are used by one source at a given time. The common data rates are 18 Mbps and 54 Mbps.

4. **IEEE 802.11b**: High Rate Direct-Sequence spread spectrum (HRDSSS): IEEE 802.11b operates in the 2.4 GHz band and supports data rates of 5.5 Mb/s to 11 Mb/s. It is similar to DSSS except for the encoding method which is called complementary code keying (CCK). CCK encodes four or eight bits to one CCK symbol.

5. **IEEE 802.11g**: (OFDM): IEEE 802.11g operates at 2.4 GHz and supports even higher data rates.

Protocol Stack



IEEE 802.11 MAC Layer (May 2015)

IEEE 802.11 provides several key functionalities: reliable data delivery, media access control, and security features.

The MAC layer consists of two sub layers:

1. The distributed-coordination function algorithm (DCF) and
2. The point-coordination function algorithm (PCF).

1) Point Coordination Function (PCF) Algorithm

The point-coordination function (PCF) provides a contention-free service. PCF is an optional feature in IEEE 802.11 and is built on top of the DCF layer to provide centralized media access.

2) Distributed Coordination Function (DCF) Algorithm

The DCF algorithm uses contention resolution, and its sublayer implements the CSMA scheme for media access control and contention resolution.

Begin DCF Algorithm for Wireless 802.11 MAC – **<u>MACA (NOV/DEC 2014)</u>**

1. The sender <u>senses the medium</u> for any ongoing traffic.

2. If the medium is idle, the sender waits for a time interval equal to IFS. Then the sender senses the medium again. <u>If the medium is still idle, the sender transmits the frame immediately.</u>
    1. After the station is found ideal, the station waits for a period of time, called the distributed inter-frame space (DIFS).

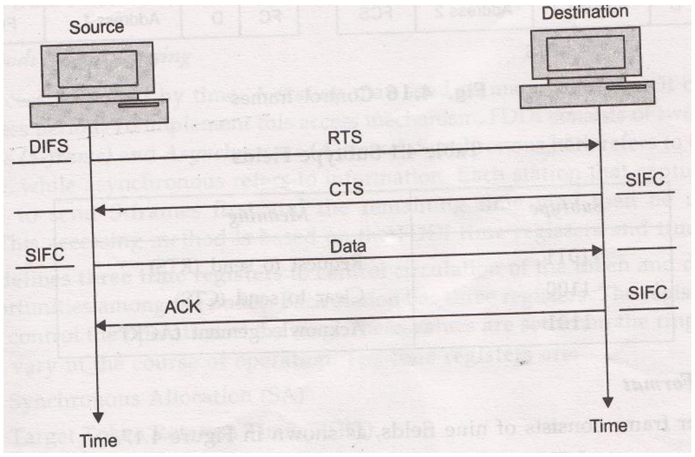
3. The station sends a <u>control frame</u> called the request to send <u>(RTS)</u>.After receiving the RTS and waiting a short period called the <u>short inter-frame space</u> (SIFS), the destination station sends a control frame called <u>clear to send</u> (CTS), to the source station. This control frame indicates that the destination station is ready to receive data.

Two or more stations made try to send RTS frames at the same time, these control frames may collide. The sender assumes there has been a collision if it has not received CTS frame from the receiver and it tries again.

4. The source station sends data after waiting an amount of time equal to SIFS.

5. The destination station after waiting for an amount of time equal to SIFS sends and acknowledgement to show that the frame has been received.

6. When a station sends an RTS frame, it includes the duration of the time that it needs to occupy the channel. The stations that are affected by this transmission create a timer called a Network Allocation Vector (NAV) that all shows how much time must pass before these stations are allowed to check the channel for idleness.



<u>**MACAW (NOV/DEC 2014)**</u>

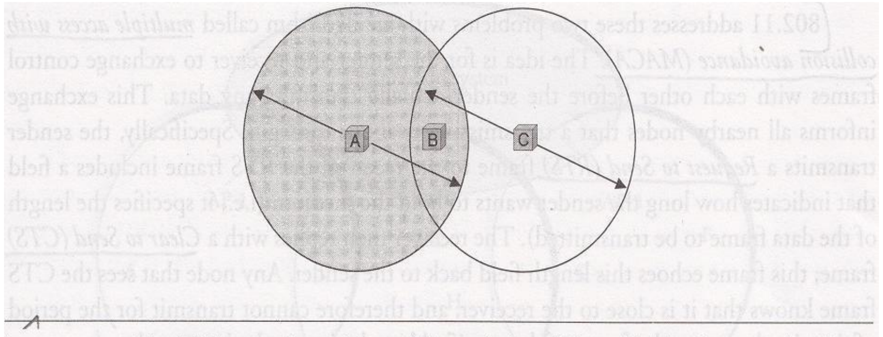
WLAN data transmission collisions can still happen, and MACA for Wireless (MACAW) is brought to extend the functionality of MACA. <u>It demands nodes to send acknowledgments after every successful frame transmission</u>. MACAW is commonly used in ad hoc networks. Moreover, it is the basis of various other MAC protocols found in wireless sensor networks (WSN).

Collision Avoidance:

- ✓ **Hidden node problem**
- ✓ **Exposed node problem**

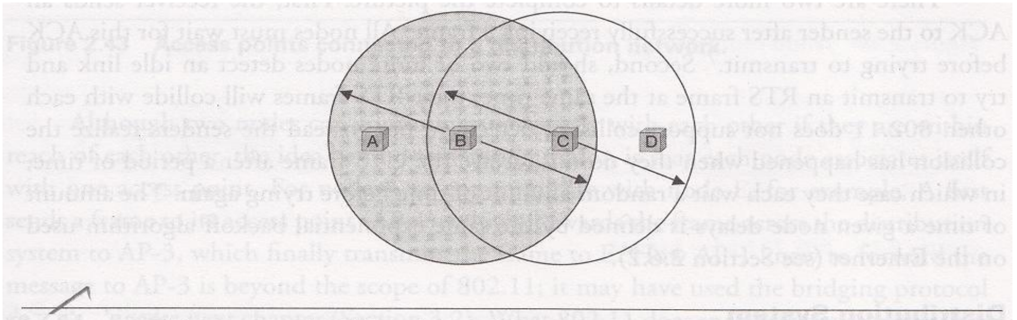
- A wireless protocol would follow the same algorithm as the Ethernet – wait until the link becomes idle before transmitting and back off should a collision occur – and to a first approximation, this is what 802.11 does.
- Consider the situation depicted in the below figure, <u>where A and C are both within range of B but not each other.</u>
- Suppose both A and C want to communicate with B and so they each send it a frame.
- A and C are unaware of each other since their signals do not carry that far.

- These two frames collide with each other at B, but unlike an Ethernet, neither A nor C is aware of this collision.
- A and C are said to be **hidden nodes** with respect to each other.



The hidden node problem. Although A and C are hidden from each other, their signals can collide at B. (B's reach is not shown.)

- A related problem, called the **exposed node problem**, occurs under the circumstances illustrated in the below figure, where each of the four nodes is able to send and receive signals that reach just the nodes to its immediate left and right.
- For example, B can exchange frames with A and C but it cannot reach D, while C can reach B and D but not A.



The exposed node problem. Although B and C are exposed to each other's signals, there is no interference if B transmits to A while C transmits to D. (A's and D's reaches are not shown.)

Distribution system

Some nodes are allowed to roam (e.g., your laptop) and some are connected to a wired network infrastructure. 802.11 calls these base stations *access points* (APs), and they are connected to each other by a so-called *distribution system*. Figure 2.32 illustrates a distribution system that connects three access points, each of which services the nodes in some region. Each access point operates on some channel in the appropriate frequency range, and each AP will typically be on a different channel than its neighbors

Although two nodes can communicate directly with each other if they are within reach of each other, the idea behind this configuration is that each node associates itself with one access point.

For node A to communicate with node E, for example, A first sends a frame to its access point (AP-1), which forwards the frame across the distribution system to AP-3, which finally transmits the frame to E. How AP-1 knew to forward the message to AP-3 is beyond the scope of 802.11; it may have used the bridging protocol described in the next chapter (Section 3.1.4). What 802.11 does specify is how nodes select their access points and, more interestingly, how this algorithm works in light of nodes moving from one cell to another.

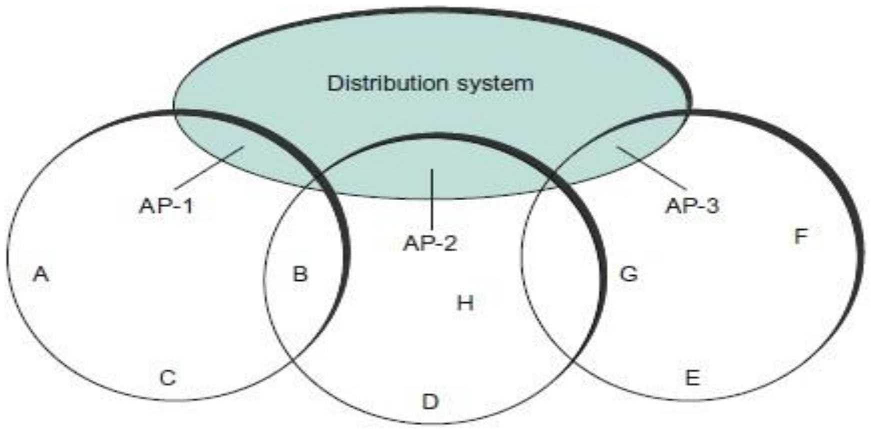


FIGURE 2.32 Access points connected to a distribution system.

The technique for selecting an AP is called *scanning* and involves the following four steps:
1. The node sends a Probe frame.
2. All APs within reach reply with a Probe Response frame.
3. The node selects one of the access points and sends that AP an Association Request frame.
4. The AP replies with an Association Response frame.

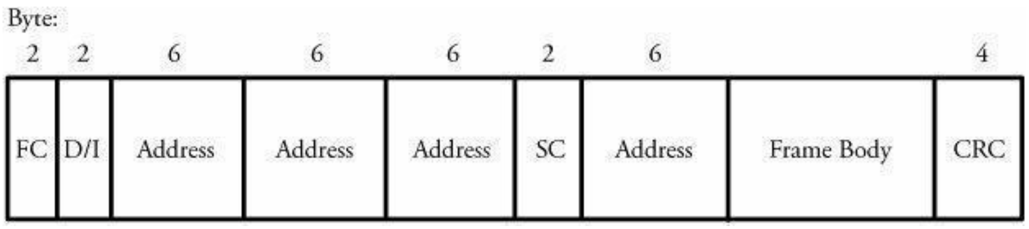
MAC Frame

The three frame types in IEEE 802.11 are control frames, data-carrying frames, and management frames.

The frame format for the 802.11 MAC is shown in the below diagram and is described as follows.

- The frame control (FC) field provides information on the type of frame: control frame, data frame, or management frame.
- Duration/connection ID (D/I) refers to the time allotted for the successful transmission of the frame.
- The addresses field denotes the 6-byte source and destination address fields.
- The sequence control (SC) field consists of 4 bits reserved for fragmentation and reassembly and 12 bits for a sequence number of frames between a particular transmitter and receiver.
- The frame body field contains a MAC service data unit or control information.
- The cyclic redundancy check (CRC) field is used for error detection.

IEEE 802.11 MAC frame



Control frames ensure reliable data delivery. The control frames are used for accessing the channel and acknowledgement frames. It consist of

FC	D	Address 1	Address 2	FCS

RTS

FC	D	Address 1	FCS

CTS or ACK

Management frames are used to monitor and manage communication among various users in the IEEE 802.11 LAN through access points.

11. Briefly discuss Bluetooth (IEEE 802.15.1) (Dec 2017)

Introduction

Bluetooth is a wireless LAN technology designed to connect devices of different functions such as telephones, notebooks, computers (desktop and laptop), cameras, printers, and even coffee makers when they are at a short distance from each other. A Bluetooth LAN is an ad hoc network, which means that the network is formed spontaneously; the devices, sometimes called gadgets, find each other and make a network called a piconet. A Bluetooth LAN can even be connected to the Internet if one of the gadgets has this capability. A Bluetooth LAN, by nature, cannot be large. If there are many gadgets that try to connect, there is chaos.

Bluetooth technology has several applications. Peripheral devices such as a wireless mouse or keyboard can communicate with the computer through this technology. Monitoring devices can communicate with sensor devices in a small health care center. Home security devices can use this technology to connect different sensors to the main security controller. Conference attendees can synchronize their laptop computers at a conference.

Bluetooth was originally started as a project by the Ericsson Company. It is named for Harald Blaatand, the king of Denmark (940-981) who united Denmark and Norway. *Blaatand* translates to *Bluetooth* in English.

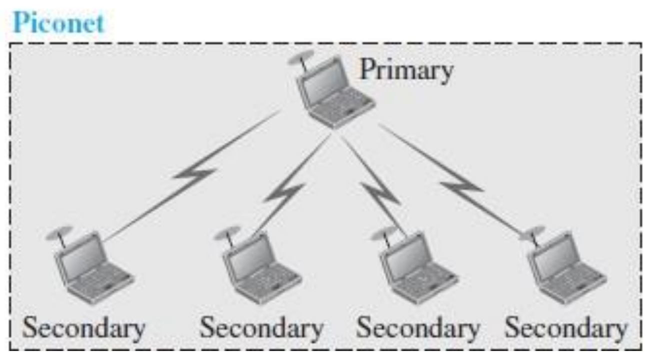
Today, Bluetooth technology is the implementation of a protocol defined by the IEEE 802.15 standard. The standard defines a wireless personal-area network (PAN) operable in an area the size of a room or a hall.

Architecture

Bluetooth defines two types of networks: **piconet and scatternet.**

- ***Piconets***

Figure 15.17 *Piconet*



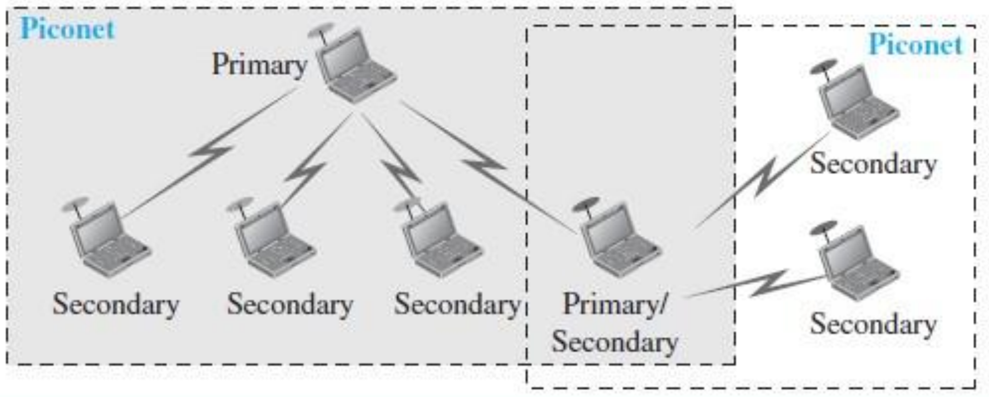
A Bluetooth network is called a ***piconet,*** or a small net. A piconet can have up to eight stations, one of which is called the *primary;* the rest are called *secondaries.* All the secondary stations synchronize their clocks and hopping sequence with the primary. Note that a piconet can have only one primary station. The communication between the primary and secondary stations can be one-to-one or one-to-many. Figure 15.17 shows a piconet.

Although a piconet can have a maximum of seven secondaries, additional secondaries can be in the *parked state*. A secondary in a parked state is synchronized with the primary, but cannot take part in communication until it is moved from the parked state to the active state. Because only eight stations can be active in a piconet, activating a station from the parked state means that an active station must go to the parked state.

- ***Scatternet***

Piconets can be combined to form what is called a ***scatternet.*** A secondary station in one piconet can be the primary in another piconet. This station can receive messages from the primary in the first piconet (as a secondary) and, acting as a primary, deliver them to secondaries in the second piconet. A station can be a member of two piconets. Figure 15.18 illustrates a scatternet.

Figure 15.18 *Scatternet*



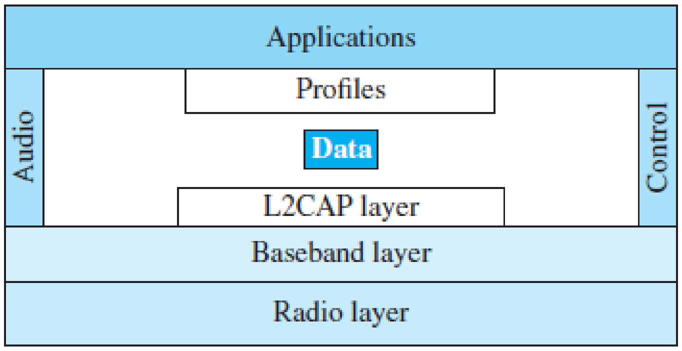
Bluetooth Devices

A Bluetooth device has a built-in short-range radio transmitter. The current data rate is 1 Mbps with a 2.4-GHz bandwidth. This means that there is a possibility of interference between the IEEE 802.11b wireless LANs and Bluetooth LANs.

Bluetooth Layers

Bluetooth uses several layers that do not exactly match those of the Internet model we have defined in this book. Figure 15.19 shows these layers.

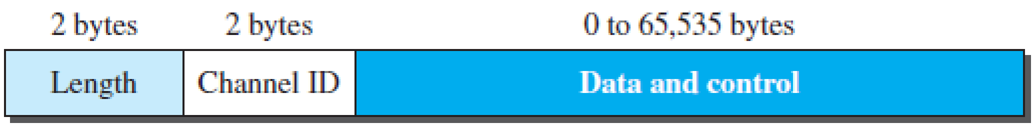
Figure 15.19 *Bluetooth layers*



L2CAP

The **Logical Link Control and Adaptation Protocol,** or **L2CAP** (L2 here means LL), is roughly equivalent to the LLC sublayer in LANs. It is used for data exchange on an ACL link; SCO channels do not use L2CAP. Figure 15.20 shows the format of the data packet at this level.

Figure 15.20 *L2CAP data packet format*

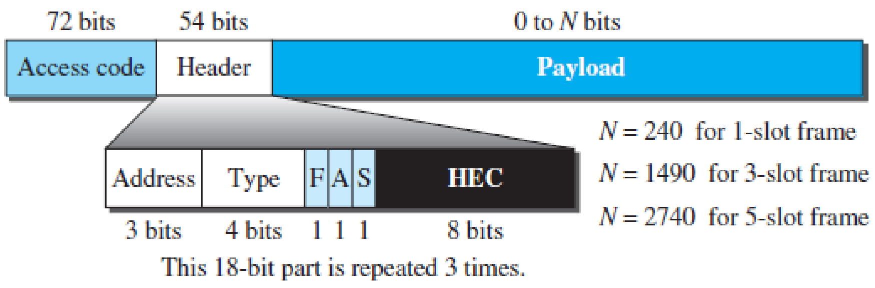


Baseband Layer

The baseband layer is roughly equivalent to the MAC sublayer in LANs. The access method is TDMA

Frame Format

Figure 15.23 *Frame format types*



Radio Layer

The radio layer is roughly equivalent to the physical layer of the Internet model. Bluetooth devices are low-power and have a range of 10 m.

Band

Bluetooth uses a 2.4-GHz ISM band divided into 79 channels of 1 MHz each.

FHSS

Bluetooth uses the **frequency-hopping spread spectrum (FHSS)** method in the physical layer to avoid interference from other devices or other networks

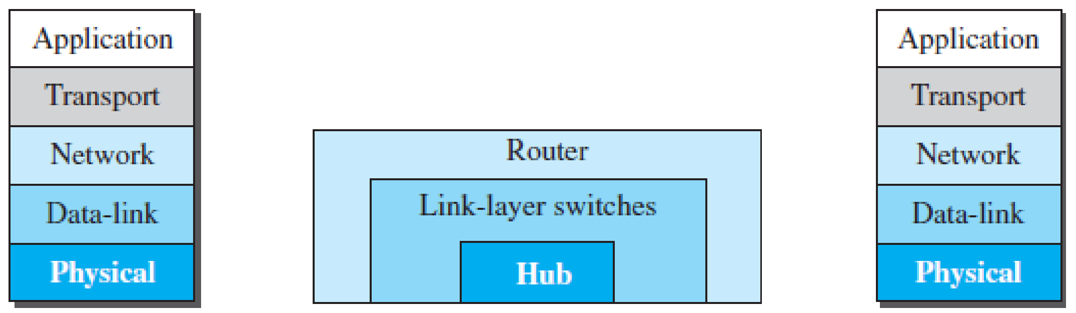
Modulation

To transform bits to a signal, Bluetooth uses a sophisticated version of FSK, called GFSK

12. Explain in detail about connecting devices in network

Hosts and networks do not normally operate in isolation. We use **connecting devices** to connect hosts together to make a network or to connect networks together to make an internet. Connecting devices can operate in different layers of the Internet model. We discuss three kinds of *connecting devices*; **hubs, link-layer switches, and routers**. Hubs today operate in the first layer of the Internet model. Link-layer switches operate in the first two layers. Routers operate in the first three layers

Figure 17.1 *Three categories of connecting devices*



Hubs

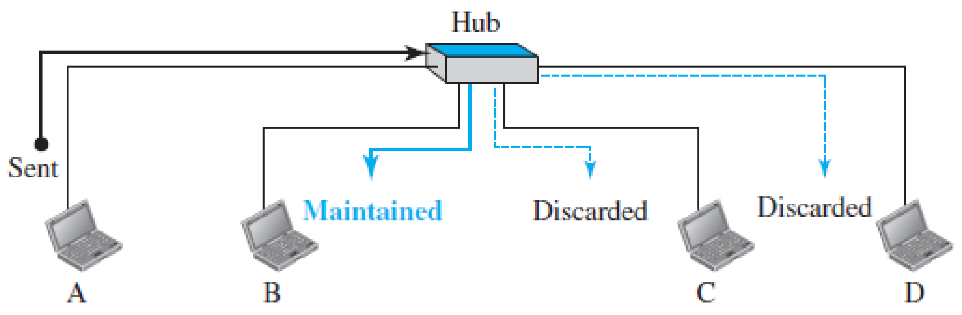
A **hub** is a device that operates only in the physical layer. Signals that carry information within a network can travel a fixed distance before attenuation endangers the integrity of the data. A **repeater** receives a signal and, before it becomes too weak or corrupted, *regenerates* and *retimes* the original bit pattern.

The repeater then sends the refreshed signal. In the past, when Ethernet LANs were using bus topology, a repeater was used to connect two segments of a LAN to overcome the length restriction of the coaxial cable. Today, however, Ethernet LANs use star topology. In a star topology, a repeater is a multiport device, often called a *hub*, that can be used to serve as the connecting point and at the same time function as a repeater.

Figure 17.2 shows that when a packet from station A to station B arrives at the hub, the signal representing the frame is regenerated to remove any possible corrupting noise, but the hub forwards the packet from all outgoing ports except the one from which the signal was received. In other words, the frame is broadcast. All stations in the LAN receive the frame, but only station B keeps it. The rest of the stations discard it.

Figure 17.2 shows the role of a repeater or a hub in a switched LAN. The figure definitely shows that a hub does not have a filtering capability; it does not have the intelligence to find from which port the frame should be sent out. A hub or a repeater is a physical-layer device. They do not have a link-layer address and they do not check the link-layer address of the received frame. They just regenerate the corrupted bits and send them out from every port.

Figure 17.2 *A hub*



Link-Layer Switches

Introduction

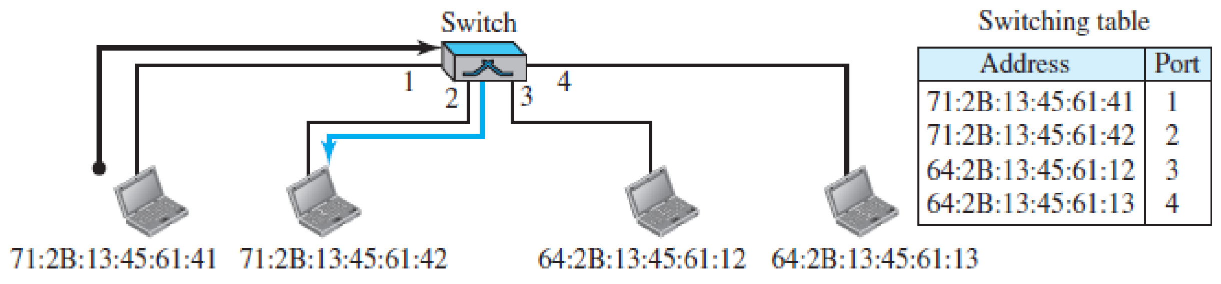
- A switch is a combination of a hub and a bridge.
- It can interconnect two or more workstations, but like a bridge, it observes traffic flow and learns.
- When a frame arrives at a switch, the switch examines the destination address and forwards the frame out the one necessary connection.
    1. Workstations that connect to a hub are on a *shared segment*.
    2. Workstations that connect to a switch are on a *switched segment*.
- The backplane of a switch is fast enough to support multiple data transfers at one time.
- A switch that employs a *cut-through architecture* is one that passes on the frame before the entire frame has arrived at the switch.
- Multiple workstations connected to a switch use dedicated segments. This is a very efficient way to isolate heavy users from the network.
- A switch can allow simultaneous access to multiple servers, or multiple simultaneous connections to a single server.

A **link-layer switch** (or ***switch***) operates in both the physical and the data-link layers. As a physical-layer device, it regenerates the signal it receives. As a link-layer device, the link-layer switch can check the MAC addresses (source and destination) contained in the frame.

Filtering

One may ask what the difference in functionality is between a link-layer switch and hub. A link-layer switch has **filtering** capability. It can check the destination address of a frame and can decide from which outgoing port the frame should be sent.

Figure 17.3 *Link-layer switch*



Address	Port
71:2B:13:45:61:41	1
71:2B:13:45:61:42	2
64:2B:13:45:61:12	3
64:2B:13:45:61:13	4

Transparent Switches

A **transparent switch** is a switch in which the stations are completely unaware of the switch's existence. If a switch is added or deleted from the system, reconfiguration of the stations is unnecessary. According to the IEEE 802.1d specification, a system equipped with transparent switches must meet three criteria:

❑ Frames must be forwarded from one station to another.
❑ The forwarding table is automatically made by learning frame movements in the network.
❑ Loops in the system must be prevented

Major role of Switches

- Isolating traffic patterns and providing multiple accesses.
- This design is usually done by the network manager.

Switches are easy to install and have components that are hot-swappable.

Advantages of switches

1. Switches divide a network into several isolated channels or collision domains
2. Reduce the possibility of collision
3. Each channel has its own network capacity
4. Connecting Heterogenous Devices

Limitations of switches

1. Although contains buffers to accommodate bursts of traffic, can become overwhelmed by heavy traffic
2. Device cannot detect collision when buffer full
3. Some higher level protocols do not detect error

<u>**Spanning Tree Algorithm**</u>

The preceding strategy works just fine until the extended LAN has a loop in it, in which case it fails in a horrible way—frames potentially loop through the extended LAN forever.

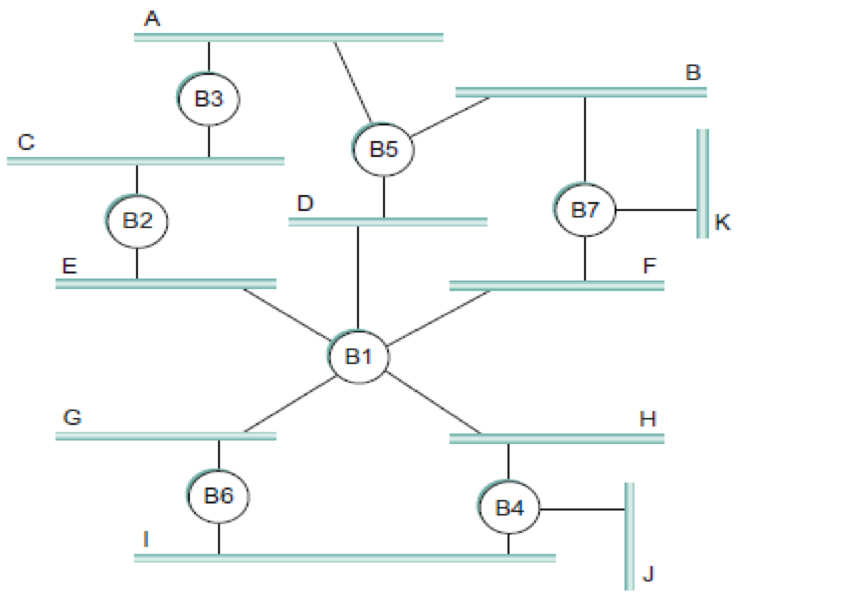


FIGURE 3.10 Extended LAN with loops.

This is easy to see in the example depicted in Figure 3.10, where, for example, bridges B1, B4, and B6 form a loop.
Whatever the cause, bridges must be able to correctly handle loops.

This problem is addressed by having the bridges run a distributed *spanning tree* algorithm.

If you think of the extended LAN as being represented by a graph that possibly has loops (cycles), then a spanning tree is a subgraph of this graph that covers (spans) all the vertices but contains no cycles. That is, a spanning tree keeps all of the vertices of the original graph but throws out some of the edges. For example, Figure 3.11 shows a cyclic graph on the left and one of possibly many spanning trees on the right.

The idea of a spanning tree is simple enough: It's a subset of the actual network topology that has no loops and that reaches all the LANs in the extended LAN. The hard part is how all of the bridges coordinate their decisions to arrive at a single view of the spanning tree. After all, one topology is typically able to be covered by multiple spanning trees. The answer lies in the spanning tree protocol.

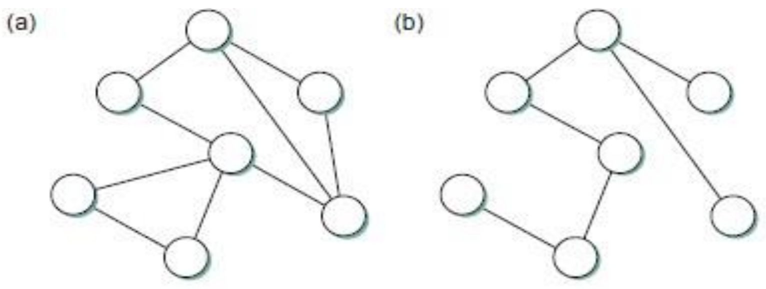


FIGURE 3.11 Example of (a) a cyclic graph; (b) a corresponding spanning tree.

The main idea of the spanning tree is for the bridges to select the ports over which they will forward frames.

The algorithm selects ports as follows.

- Each bridge has a unique identifier; for our purposes, we use the labels B1, B2, B3, and so on.
- The algorithm first elects the bridge with the smallest ID as the root of the spanning tree;
- Next, each bridge computes the shortest path to the root and notes which of its ports is on this path.
- Finally, all the bridges connected to a given LAN elect a single *designated* bridge that will be responsible for forwarding frames toward the root bridge

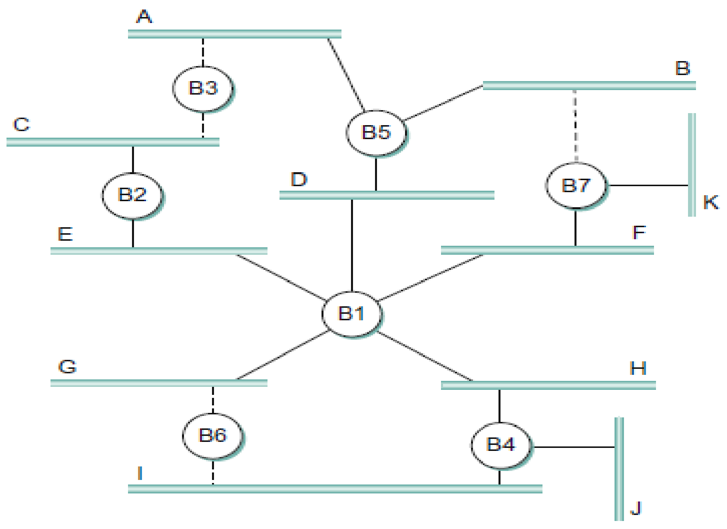


FIGURE 3.12 Spanning tree with some ports not selected.

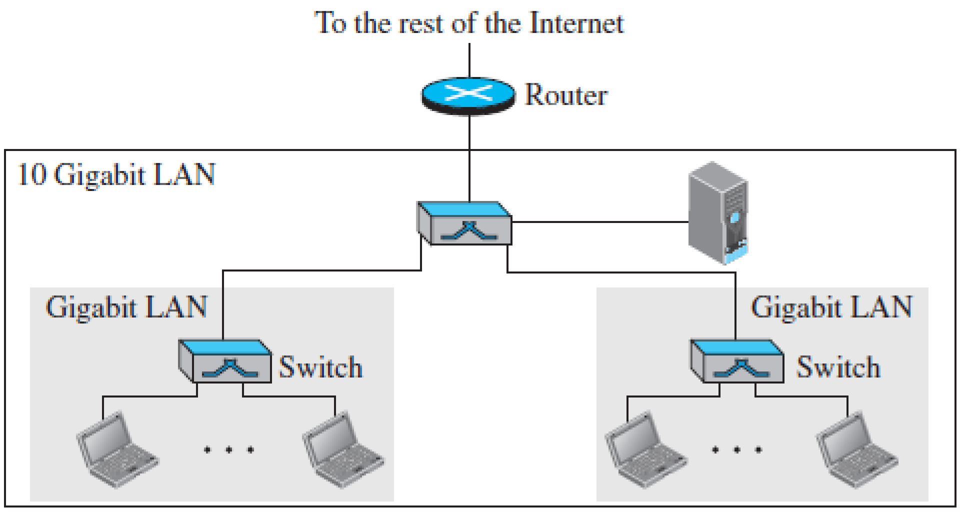
Routers

A **router** is a three-layer device; it operates in the physical, data-link, and network layers. As a physical-layer device, it regenerates the signal it receives. As a link-layer device, the router checks the physical addresses (source and destination) contained in the packet. As a network-layer device, a router checks the network-layer addresses.

A router can connect networks. In other words, a router is an internetworking device; it connects independent networks to form an internetwork. According to this definition, two networks connected by a router become an internetwork or an internet. There are three major differences between a router and a repeater or a switch.
1. A router has a physical and logical (IP) address for each of its interfaces.
2. A router acts only on those packets in which the link-layer destination address matches the address of the interface at which the packet arrives.
3. A router changes the link-layer address of the packet (both source and destination) when it forwards the packet.

Let us give an example. In Figure 17.9, assume an organization has two separate buildings with a Gigabit Ethernet LAN installed in each building. The organization uses switches in each LAN. The two LANs can be connected to form a larger LAN using 10 Gigabit Ethernet technology that speeds up the connection to the Ethernet and the connection to the organization server. A router then can connect the whole system to the Internet.

Figure 17.9 *Routing example*



UNIVERSITY QUESTIONS

B.E/B.TECH NOVEMBER/DECEMBER 2014 (2008 Regulation)

2 MARKS

1. Differentiate persistent and non persistent CSMA (Q.NO. 41)
2. State the uses of valid transmission timer (Q.NO. 42)

16 MARKS

1.Explain and differentiate FDDI and Ethernet (16) (Q.NO. 2,13 & 14)
2.Write short notes on.
(i) Transparent bridges (8) (Q.NO. 6)
(ii) MACA and MACAW (8) (Q.NO. 4)

B.E/B.Tech April May 2015

2 MARKS

1. What do you understand by CSMA protocol? (Q.NO. 43)
2. List the functions of bridges (Q.NO. 44)

16 MARKS

1. Explain in detail about access method and frame format used in Ethernet and token ring (16) (Q.NO. 2 & 16)
2. (i) Discuss the MAC layer functions of IEEE802.11 (8) (Q.NO. 4)
(ii) Briefly define key requirements of wireless LAN (8) (Q.NO. 3)

B.E/B.Tech Nov-Dec 2015

2 MARKS

1. Define sub-netting. (Q.NO 28)
2. What is the need of ARP? (Q.NO 36)
3. Identify the class of the following IP address: (a) 110.34.56.45 (b) 212.208.63.23 (Q.NO 47)

16 MARKS

1. Write short notes on Ethernet & Wireless LAN (8+8) (Q.NO 2 & 3)
2. Explain in detail ARP, DHCP, ICMP (16) (Q.NO 8, 9 & 10)

B.E/B.Tech April-May 2016

2 MARKS

1. Define hidden node problem. (Q.NO 45)
2. What is Bluetooth? (Q.NO 7)
3. Expand ICMP and write the function (Q.NO 27)

16 MARKS

1. Give the comparison between different wireless technologies? Enumerate 802.11 protocol stack in detail (16) (Q.NO 3 & 4)
2. Write short notes on DHCP & ICMP (8+8) (Q.NO 9 & 10)

B.E/B.Tech Nov-Dec 2016

2 MARKS

1. What is meant by exponential backoff? (Q.NO 5)
2. What is scatternet? (Q.NO 46)
3. What is fragmentation and reassembly? (Q.NO 48)

16 MARKS

1. Explain the physical properties of Ethernet 802.3 with necessary diagram of Ethernet transceiver and adapter (16) (Q.NO 2)
2. With a neat sketch explain about IP service model,packet format,Fragmentation and reassembly.(16) (Q.NO 11)

B.E/B.Tech April-May 2017

PART A

1. State the functions of bridges. (Q.NO 44)
2. When is ICMP redirect message used? (Q.NO 49)

PART B

1.i) Discuss the working of CSMA/CD protocol (6) (Q.NO 15)
ii) Explain the functions of MAC layer present in IEEE802.11 with necessary diagrams (7) (Q.NO 4)
2. Explain the working of DHCP protocol with its header format (Q.NO 9)

B.E/B.Tech Nov-Dec 2017

PART A

1. Show the Ethernet frame format (Q.NO 37)
2. Highlights the characteristics of datagram networks (Q.NO 50)

PART B

1. Explain the functions of Wi-Fi & Bluetooth in detail (13) (Q.NO 4 & 5)
2. i)Explain the datagram forwarding in IP (Q.NO 11)
   ii) Show and explain the ARP packet format for mapping IP addresses into Ethernet addresses (Q.NO 8)

UNIT III - NETWORK LAYER

Network Layer Services – Packet switching – Performance – IPV4 Addresses – Forwarding of IP Packets - Network Layer Protocols: IP, ICMP v4 – Unicast Routing Algorithms – Protocols – Multicasting Basics – IPV6 Addressing – IPV6 Protocol.

PART A

1. List the various services provided in the Network Layer.

- Packetizing
- Routing and Forwarding
- Other Services
  - *Error Control*
  - *Flow Control*
  - *Congestion Control*
  - *Quality of Service*
  - *Security*

2. Define packetizing.

- **Packetizing:** encapsulating the payload (data received from upper layer) in a network-layer packet at the source and decapsulating the payload from the network-layer packet at the destination.

3. Define Routing and Forwarding.

Routing

- The network layer is responsible for routing the packet from its source to the destination.

Forwarding

- *Forwarding* can be defined as the action applied by each router when a packet arrives at one of its interfaces.

4. Define packet switched network and list the different approaches to route the packet.

Packet Switched Network:

- Packet switching is used at the network layer because the unit of data at this layer is a packet.
- At the network layer, a message from the upper layer is divided into manageable packets and each packet is sent through the network.
- A packet-switched network can use two different approaches to route the packets:
  - The *datagram approach* - Connectionless Service
  - The *virtual circuit approach* - Connection-Oriented Service

5. Narrate how the performance of a network or network layer can be measured ?

- The performance of a network can be measured in terms of
  - *delay,*
    - transmission delay,
    - propagation delay,
    - processing delay,
    - queuing delay.
  - *throughput,*
  - *packet loss.*
  - *Congestion Control*

6. Define Transmission delay

- A sender needs to put the bits in a packet on the line one by one.
- The transmission delay is longer for a longer packet and shorter if the sender can transmit faster.
- In other words, the transmission delay is

$$\textbf{Delay}_{\textbf{tr}} = \textbf{(Packet length) / (Transmission rate).}$$

7. **Define *Propagation Delay***

- Propagation delay is the time it takes for a bit to travel from point A to point B in the transmission media.

$$\textbf{Delay}_{\textbf{pg}} = \textbf{(Distance) / (Propagation speed).}$$

8. **Define *Processing Delay***

- The processing delay is the time required for a router or a destination host to receive a packet from its input port, remove the header, perform an error detection procedure, and deliver the packet to the output port (in the case of a router) or deliver the packet to the upper-layer protocol (in the case of the destination host).

$$\textbf{Delay}_{\textbf{pr}} = \textbf{Time required to process a packet in a router or a destination host}$$

9. Queuing Delay

- The queuing delay for a packet in a router is measured as the time a packet waits in the input queue and output queue of a router.

$$\textbf{Delay}_{\textbf{qu}} = \textbf{The time a packet waits in input and output queues in a router}$$

10. Define Throughput.

- Throughput at any point in a network is defined as the number of bits passing through the point in a second, which is actually the transmission rate of data at that point.

- In a path from source to destination, a packet may pass through several links (networks), each with a different transmission rate.

Throughput = minimum {TR1, TR2, . . . TRn}.

11. Define Congestion Control and mention its types.

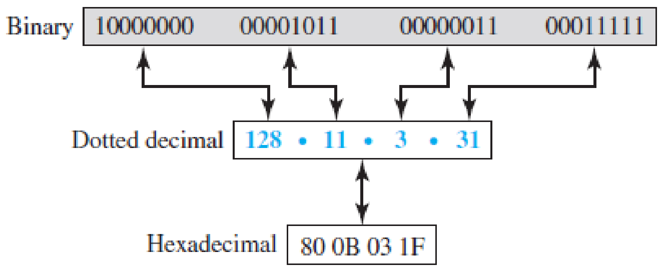
- Congestion control is a mechanism for improving performance.
- Two broad categories:
    - open-loop congestion control (prevention)
    - closed-loop congestion control (removal).

12. Define IPv4 Address and list the various types of notations.

- An IPv4 address is a 32-bit address that uniquely and universally defines the connection of a host or a router to the Internet.
- IPv4 addresses are unique. If a device has two connections to the Internet, via two networks, it has two IPv4 addresses.

Notation

- There are three common notations to show an IPv4 address: binary notation (base 2), dotted-decimal notation (base 256), and hexadecimal notation (base 16).



13. Define and Differentiate Classful and Classless Addressing.

Classful Addressing

- An IPv4 address was designed with a fixed-length prefix. The whole address space was divided into five classes (class A, B, C, D, and E). This scheme is referred to as classful addressing.

Classless Addressing

- In addressing, the whole address space is divided into variable length classless blocks.
- The prefix in an address defines the block (network); the suffix defines the node (device).
- A prefix length ranges from 0 to 32.

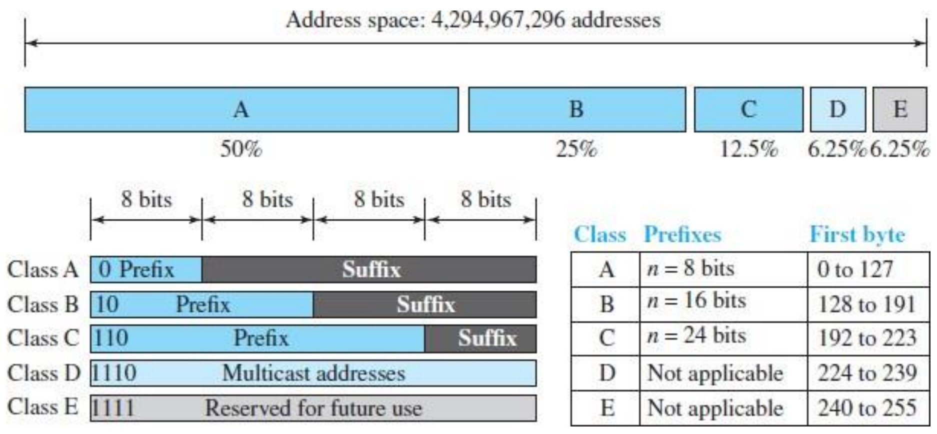
Difference between Classless Addressing and Classful Addressing

Classful Addressing	Classless Addressing
An IP Address allocation method that allocates IP addresses according to five major classes.	An IP Address allocation method that is designed to replace classful addressing to minimize the rapid exhaustion of IP addresses.
Less practical and useful.	More practical and useful.
Network ID and host ID changes depending on the classes.	There is no boundary on Network ID and host ID
Addresses have three parts: network, subnet, and host.	Addresses have two parts: subnet or prefix, and host.
IP forwarding process is restricted in how it uses the default route	IP forwarding process has no restrictions on using the default route
Routing protocol does not advertise masks nor support VLSM; RIP-1 and IGRP	Routing protocol does advertise masks and support VLSM; RIP-2, EIGRP, OSPF.

14. Define Address Masking.

- The address mask is a 32-bit number in which the n leftmost bits are set to 1s and the rest of the bits (32 − n) are set to 0s.
- To extract the information in a block, using the three bit-wise operations NOT, AND, and OR.
  1. The number of addresses in the block N = NOT (mask) + 1.
  2. The first address in the block = (Any address in the block) AND (mask).
  3. The last address in the block = (Any address in the block) OR [(NOT (mask)].

15. Specify the various types of Classes and its range in Classful Addressing.



Class	Prefixes	First byte
A	$n = 8$ bits	0 to 127
B	$n = 16$ bits	128 to 191
C	$n = 24$ bits	192 to 223
D	Not applicable	224 to 239
E	Not applicable	240 to 255

16. A classless address is given as 167.199.170.82/27. Find the number of addresses, First address and last address of the block.

Solution:

- The **number of addresses** in the network is $2^{32} - n = 2^5 =$ **32 addresses**.
- The **first address** can be found by keeping the first 27 bits and changing the rest of the bits to 0s.

  Address:

  167.199.170.82/**27** 10100111 11000111 10101010 01010010

  First address:

  167.199.170.64/**27** 10100111 11000111 10101010 010**00000**
- The **last address** can be found by keeping the first 27 bits and changing the rest of the bits to 1s.

  Address:

  167.199.170.82/**27** 10100111 11000111 10101010 01011111

  Last address:

  167.199.170.95/27 10100111 11000111 10101010 010**11111**

17.An organization is granted a block of addresses with the beginning address 14.24.74.0/24. The organization needs to have 3 subblocks of addresses to use in its three subnets: one subblock of 10 addresses, one subblock of 60 addresses, and one subblock of 120 addresses. Design the subblocks.

Solution

- There are $2^{32} - {}^{24}$ = **256 addresses** in this block.
- The first address is **14.24.74.0/24**; the last address is **14.24.74.255/24**.

Subblock with 120 addresses:

- The number of addresses in the largest subblock, which requires 120 addresses, is not a power of 2. We allocate **128 addresses**.
- The subnet mask for this subnet can be found as n1 = $32 - \log_2 128$ = **25**. The first address in this block is **14.24.74.0/25;** the last address is **14.24.74.127/25**.

Subblock with 60 addresses:

- The number of addresses in the second largest subblock, which requires 60 addresses, is not a power of 2 either. We allocate **64 addresses**.
- The subnet mask for this subnet can be found as n2 = $32 - \log_2 64$ = **26.**
- The first address in this block is **14.24.74.128/26**; the last address is **14.24.74.191/26**.

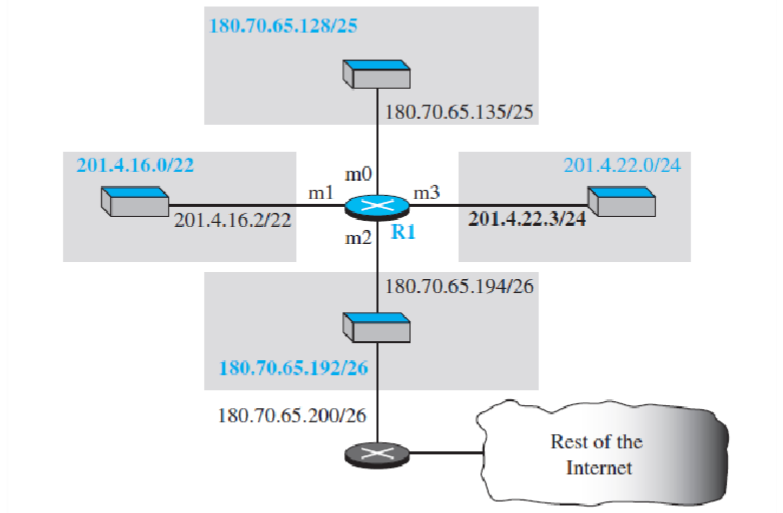
Subblock with 60 addresses:

- The number of addresses in the smallest subblock, which requires 10 addresses, is not a power of 2 either. We allocate **16 addresses**.
- The subnet mask for this subnet can be found as n3 = $32 - \log_2 16$ = **28**.
- The first address in this block is **14.24.74.192/28**; the last address is

14.24.74.207/28.

- If we add all addresses in the previous subblocks, the result is **208** addresses. The first address in this range is **14.24.74.208**. The last address is **14.24.74.255**.

18. Make a forwarding table for router R1 using the configuration in Figure



Solution:

Network address/mask	*Next hop*	*Interface*
180.70.65.192/26	—	m2
180.70.65.128/25	—	m0
201.4.22.0/24	—	m3
201.4.16.0/22	—	m1
Default	180.70.65.200	m2

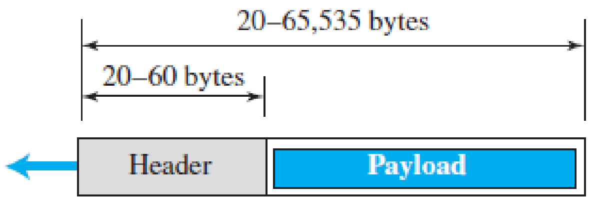
19. How the IP packets are forwarded?

FORWARDING OF IP PACKETS

- Forwarding Based on Destination Address
- Forwarding Based on Label

20. Define Datagram.

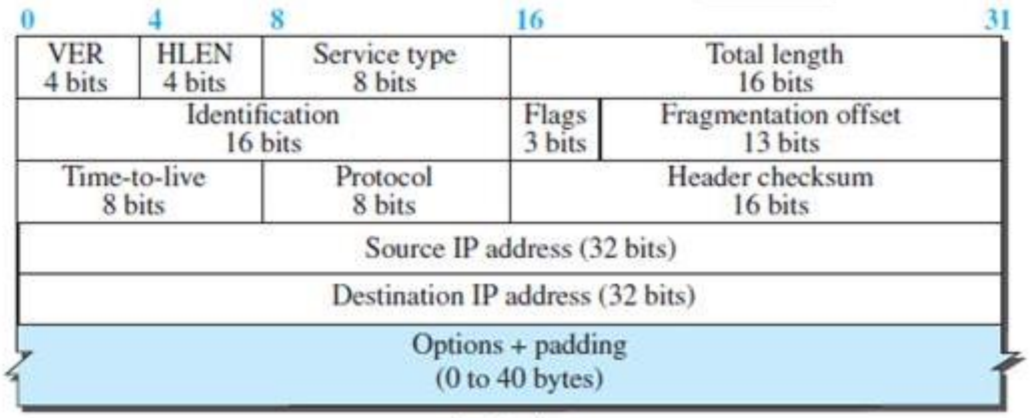
- Packets used by the IP are called *datagrams*.
- A datagram is a variable-length packet consisting of two parts: header and payload (data).
- The header is 20 to 60 bytes in length and contains information essential to routing and delivery.



a. IP datagram

21. What is IPv4 and mention its IPv4 packet format.

- The Internet Protocol version 4 (IPv4), is responsible for packetizing, forwarding, and delivery of a packet at the network layer.



b. Header

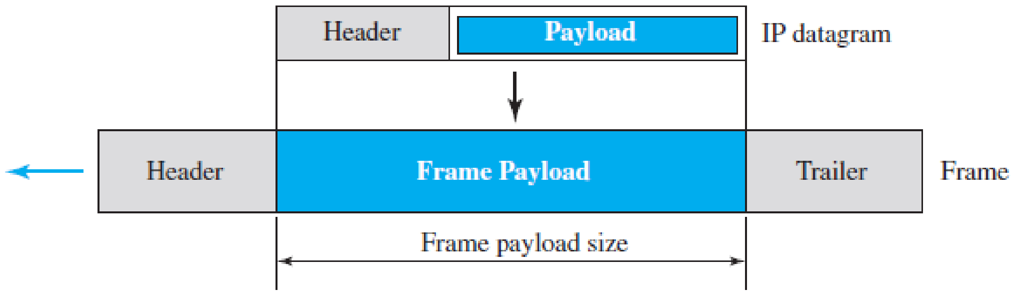
22. Define fragmentation and explain how it is performed.

➢ **Fragmentation**

- The division of a packet into smaller units to accommodate a protocol's MTU.

Maximum Transfer Unit (MTU)

- The largest size data unit a specific network can handle.



MTU: Maximum size of frame payload

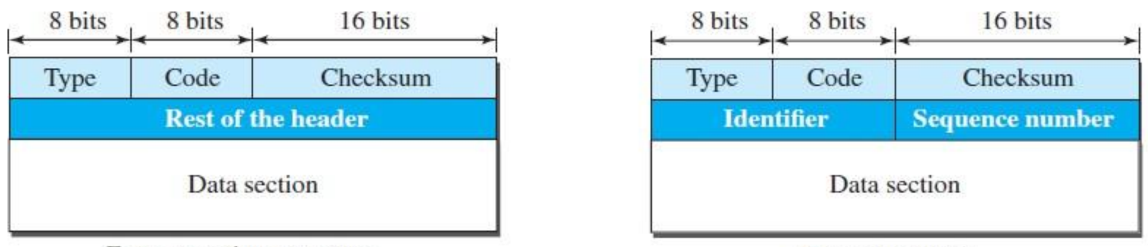
- When a datagram is fragmented, each fragment has its own header with most of the fields repeated, but some have been changed.
- A datagram may be fragmented several times before it reaches the final destination.

23.Narrate the purpose of Internet Control Message Protocol version 4 (ICMPv4) message.

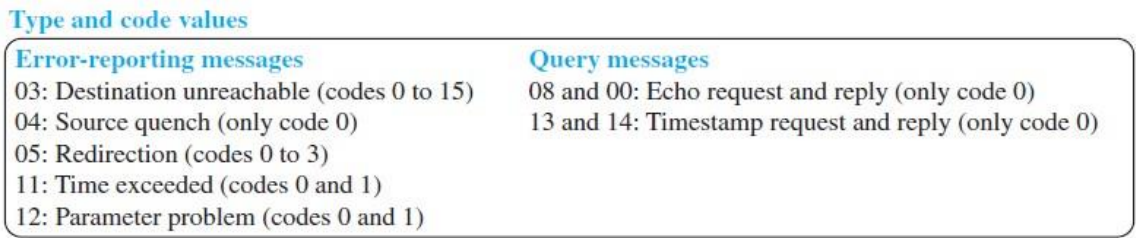
- The Internet Control Message Protocol version 4 (ICMPv4) helps IPv4 to handle some errors that may occur in the network-layer delivery.
- ICMP is used to report some errors that may occur during the processing of the IP datagram. ICMP does not correct errors, it simply reports them.

24. List the various ICMPv4 Error Messages.

- ICMP messages are divided into two broad categories: *error-reporting messages* and *query messages*



Error-reporting messages Query messages


Type and code values

Error-reporting messages	Query messages
03: Destination unreachable (codes 0 to 15)	08 and 00: Echo request and reply (only code 0)
04: Source quench (only code 0)	13 and 14: Timestamp request and reply (only code 0)
05: Redirection (codes 0 to 3)	
11: Time exceeded (codes 0 and 1)	
12: Parameter problem (codes 0 and 1)	

25. List and define the two debugging tools used in ICMPv4 messages. Or Define Ping and Traceroute

- Two debugging tools: *ping* and *traceroute*.

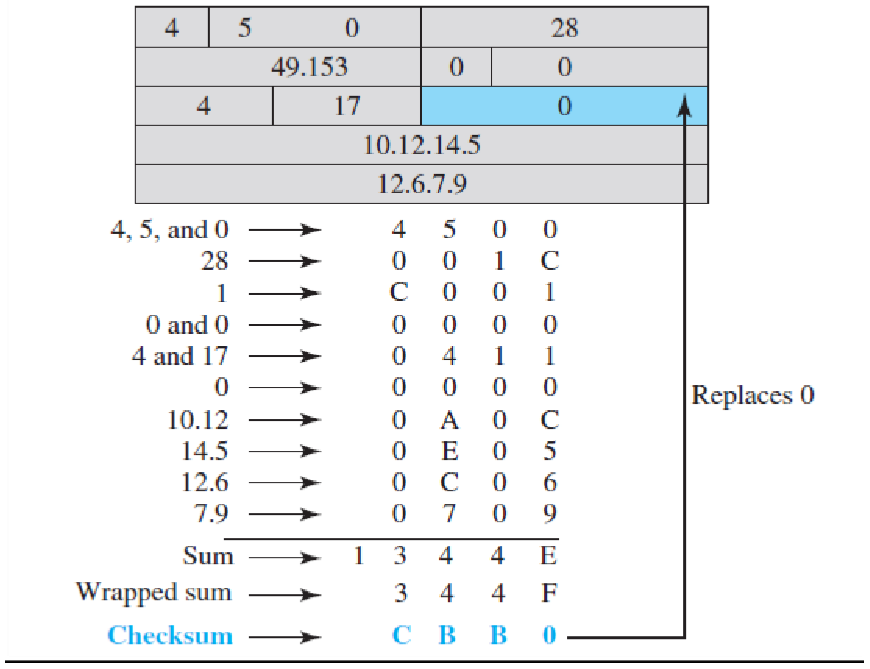
Ping

- *Ping* program is used to find if a host is alive and responding.
- The source host sends ICMP echo-request messages; the destination, if alive, responds with ICMP echo-reply messages.
- The ping program gets help from two query messages;

Traceroute or Tracert

- The *traceroute* program in UNIX or *tracert* in Windows can be used to trace the path of a packet from a source to the destination.
- It can find the IP addresses of all the routers that are visited along the path.
- The *traceroute* program gets help from two error-reporting messages: time-exceeded and destination-unreachable.

26.An example of a checksum calculation for an IPv4 header without options. The header is divided into 16-bit sections. All the sections are added and the sum is complemented after wrapping the leftmost digit. The result is inserted in the checksum field.



27. List the security issues practically applicable to IP Datagrams.

- There are three security issues that are particularly applicable to the IP protocol:
  - packet sniffing,
  - packet modification,
  - IP spoofing.

Packet Sniffing

- Packet sniffing is a passive attack, in which the attacker does not change the contents of the packet.
- This type of attack is very difficult to detect because the sender and the receiver may never know that the packet has been copied.

Packet Modification

- The attacker intercepts the packet, changes its contents, and sends the new packet to the receiver.
- The receiver believes that the packet is coming from the original sender.
- This type of attack can be detected using a data integrity mechanism.

IP Spoofing

- An attacker can masquerade as somebody else and create an IP packet that carries the source address of another computer.
- An attacker can send an IP packet to a bank pretending that it is coming from one of the customers.
- This type of attack can be prevented using an origin authentication mechanism

28. How IP packets are protected from various security issues.

IPSec

- The IP packets today can be protected from the security attacks using a protocol called IPSec (IP Security).

IPSec provides the following four services:

- *Defining Algorithms and Keys*.
- *Packet Encryption*.
- *Data Integrity*.
- *Origin Authentication*.

29. List the various routing algorithms or unicast routing algorithms in detail.

- **ROUTING ALGORITHMS**
  - Distance-Vector Routing
  - Link-State Routing
  - Path-Vector Routing

30. **Define *Bellman-Ford Equation***

- This equation is used to find the least cost (shortest distance) between a source node, ***x***, and a destination node, ***y***, through some intermediary nodes (**a, b, c,** . . .).
- The following shows the general case in which D***ij*** is the shortest distance and c***ij*** is the cost between nodes ***i*** and ***j***.

$$D_{xy} = \min\{(c_{xa} + D_{ay}), (c_{xb} + D_{by}), (c_{xc} + D_{cy}), \ldots\}$$

- In distance-vector routing, normally we want to update an existing least cost with a least cost through an intermediary node, such as ***z***, if the latter is shorter.

$$D_{xy} = \min\{D_{xy}, (c_{xz} + D_{zy})\}$$

31. Define Border Gateway Protocol (BGP)

- The Border Gateway Protocol version 4 (BGP4) is the only interdomain routing protocol, based on the path-vector algorithm.
- BGP allows routers to carry specific policies or constraints that they must meet.
- In BGP, two contributing (casual) routers can exchange routing information even if they are located in two different autonomous systems.

32. Write the keys for understanding the distance vector routing?

The three keys for understanding the algorithm are,

- Knowledge about the whole networks
- Routing only to neighbors
- Information sharing at regular intervals

33. Write the keys for understanding the link state routing?

The three keys for understanding the algorithm are,

- Knowledge about the neighborhood.
- Routing to all neighbors.
- Information sharing when there is a range.

34. How the packet cost referred in distance vector and link state routing?

- In distance vector routing, cost refer to hop count while in case of link state routing, cost is a weighted value based on a variety of factors such as security levels, traffic or the state of the link.

35. What are the features in OSPF?

- Authentication of routing messages.
- Additional hierarchy.
- Load balancing.

36. Define Sub netting?

- Sub netting provides an elegantly simple way to reduce the total number of network numbers that are assigned.
- The idea is to take a single IP network number and allocate the IP address with that network to several physical networks, which are now referred to as subnets.

37. What is DHCP? (NOV/DEC 2012)

- Dynamic Host Configuration Protocol (DHCP) is a protocol designed to provide information dynamically.
- It is a client-server program.
- DHCP is used to assign addresses to a host dynamically.
- Basically, DHCP server has two databases.
- The first database is addresses to IP addresses.

38. What are the salient features of IPV6? (NOV/DEC 2012)

- New Packet Format and Header
- Large Address Space
- State full and Stateless IPv6 address
- Multicast
- Integrated

37. What are the different routing techniques available to manage routing table entries?

1. Next hop routing.
2. Network specific routing
3. Host specific routing
4. Default routing

38. WhatisIPv6?

- **Internet Protocol version 6** (**IPv6**) is the latest revision of the Internet Protocol (IP), the communications that provides an identification and location system for computers on networks and routes traffic across the Internet.
- IPv6 was developed by the Internet Engineering Task Force (IETF) to deal with the long-anticipated problem of IPv4 address exhaustion.

39. Discuss Congestion avoidance in network layer.

- Congestion occurs in a computer network when the resource demands exceed the capacity. Packets may be lost due to too much queuing in the network.
- During congestion, the network throughput may drop and the path delay may become very high.
- A congestion control scheme helps the network to recover from the congestion state.
- A congestion avoidance scheme allows a network to operate in the region of low delay and high throughput. Such schemes prevent a network from entering the congested state.
- Congestion avoidance is a prevention mechanism while congestion control is a recovery mechanism.

40. What is the need of sub netting? (NOV/DEC 2013& 2015)

- When we divide a network into several subnets, we have three levels of hierarchy
  - o The netid is the first level, defines the site.
  - o The subnetid is the 2nd level, defines the physical subnetwork.
  - o The hostid is the 3rd level defines the connection of the host to the subnetwork.

41. What is a hostid and netid?

- **Netid** – The portion of the IP address that identifies the network called the netid.
- **Hostid** – The portion of the IP address that identifies the host or router on the network is called the hostid.

42. What is the difference between boundary level masking and non-boundary level masking.

- **Boundary level Masking:**

  If the masking is at the boundary level, the mask numbers are either 255 or 0, finding the subnetwork address is very easy.
- **Non Boundary level Masking**

  If the masking is not at the boundary level, the mask numbers are not just 255 or 0, finding the subnetwork address involves using the bitwise AND operators.

43. How does a router differ from a bridge?

- Routers provide links between two separate but same type LANs and are most active at the network layer.
- Whereas bridges utilize addressing protocols and can affect the flow control of a single LAN; most active at the data link layer.

44. Identify the class and default subnet mask of the IP address 217.65.10.7.

It belongs to class C.
Default subnet mask – 255.255.255.192

45. What is the time to live field in IP header?

- Time to live field is counter used to limit packet lifetimes counts in second and default value is 255 sec.

46. What are the main disadvantages of distance vector routing?

1. Split horizon
2. Count to infinity problem

47. What are the desirable properties of a routing algorithms?

1. Correctness
2. Simplicity
3. Robustness
4. Stability
5. Fairness
6. Optimality

PART B

1. Explain in detail about Network Layer Services.

➢ **Packetizing**

- **Packetizing:** encapsulating the payload (data received from upper layer) in a network-layer packet at the source and decapsulating the payload from the network-layer packet at the destination.
- The source host receives the payload from an upper-layer protocol, adds a header that contains the source and destination addresses and some other information that is required by the network-layer protocol and delivers the packet to the data-link layer.
- The destination host receives the network-layer packet from its data-link layer, decapsulates the packet, and delivers the payload to the corresponding upper-layer protocol.

- The routers in the path are not allowed to decapsulate the packets they received unless the packets need to be fragmented.
- The routers are not allowed to change source and destination addresses either.

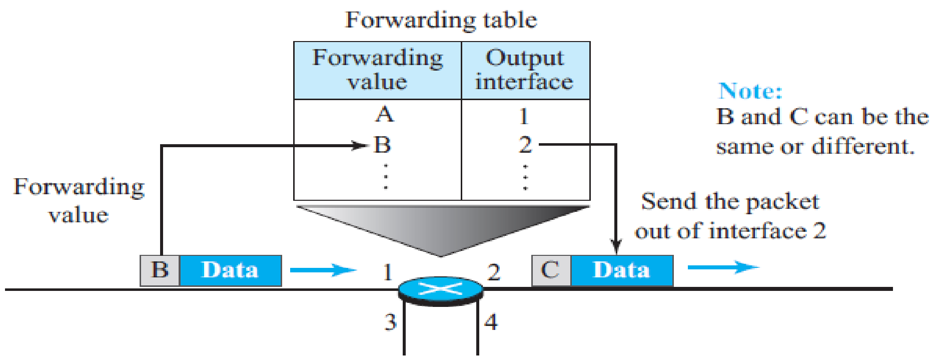
➢ Routing and Forwarding

Routing

- The network layer is responsible for routing the packet from its source to the destination.
- A physical network is a combination of networks (LANs and WANs) and routers that connect them.
- The network layer is responsible for finding the best one among these possible routes.

Forwarding

- *Forwarding* can be defined as the action applied by each router when a packet arrives at one of its interfaces.
- The decision-making table a router normally uses for applying this action is sometimes called the *forwarding table* and sometimes the *routing table*.
- When a router receives a packet from one of its attached networks, it needs to forward the packet to another attached network or to some attached networks.



➢ Other Services

1. *Error Control*

- The designers of the network layer, however, have added a checksum field to the datagram to control any corruption in the header, but not in the whole datagram.
- This checksum may prevent any changes or corruptions in the header of the datagram.
- The Internet uses an auxiliary protocol, ICMP, that provides some kind of error control if the datagram is discarded or has some unknown information in the header.

2. *Flow Control*

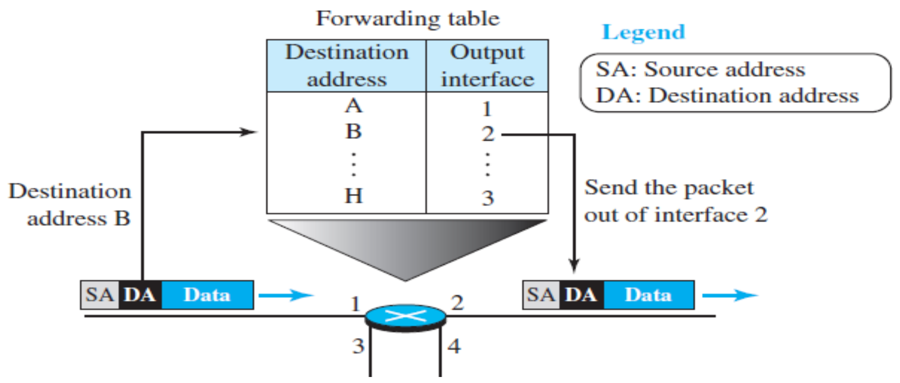
- Flow control regulates the amount of data a source can send without overwhelming the receiver.
- To control the flow of data, the receiver needs to send some feedback to the sender to inform the latter that it is overwhelmed with data.
- The network layer in the Internet, however, does not directly provide any flow control.
-

3. ***Congestion Control***
   - Another issue in a network-layer protocol is congestion control. Congestion in the network layer is a situation in which too many datagrams are present in an area of the Internet.
   - Congestion may occur if the number of datagrams sent by source computers is beyond the capacity of the network or routers.
   - If the congestion continues, sometimes a situation may reach a point where the system collapses and no datagrams are delivered.
4. ***Quality of Service***
   - As the Internet has allowed new applications such as multimedia communication, the quality of service (QoS) of the communication has become more and more important.
5. ***Security***
   - The network layer was designed with no security provision.
   - To provide security for a connectionless network layer, we need to have another virtual level that changes the connectionless service to a connection-oriented service. This virtual layer, called IPSec.

.

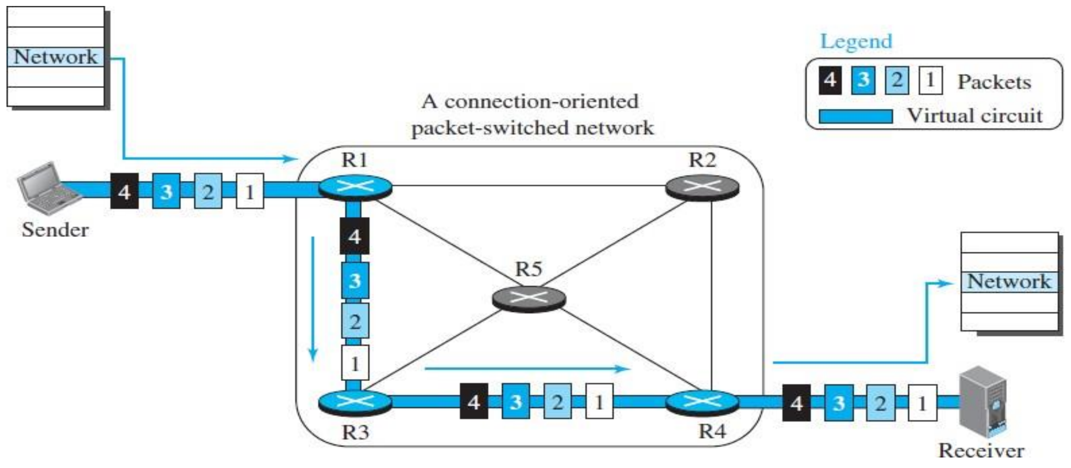
2. Explain in detail about Packet Switching in Network Layer.

- **PACKET SWITCHING**
  - Packet switching is used at the network layer because the unit of data at this layer is a packet.
  - At the network layer, a message from the upper layer is divided into manageable packets and each packet is sent through the network.
  - A packet-switched network can use two different approaches to route the packets: the *datagram approach* and the *virtual circuit approach*
- **Datagram Approach: Connectionless Service**
  - The network layer was designed to provide a connectionless service in which the network-layer protocol treats each packet independently.
  - The network layer is only responsible for delivery of packets from the source to the destination. The switches in this type of network are called *routers*.
  - Each packet is routed based on the information contained in its header: source and destination addresses.
  - The destination address defines where it should go; the source address defines where it comes from.



➢ **Virtual-Circuit Approach: Connection-Oriented Service**

- In a connection-oriented service (also called *virtual-circuit approach*), there is a relationship between all packets belonging to a message.
- Before all datagrams in a message can be sent, a virtual connection should be set up to define the path for the datagrams.
- After connection setup, the datagrams can all follow the same path.
- In this type of service, not only must the packet contain the source and destination addresses, it must also contain a flow label, a virtual circuit identifier that defines the virtual path the packet should follow.

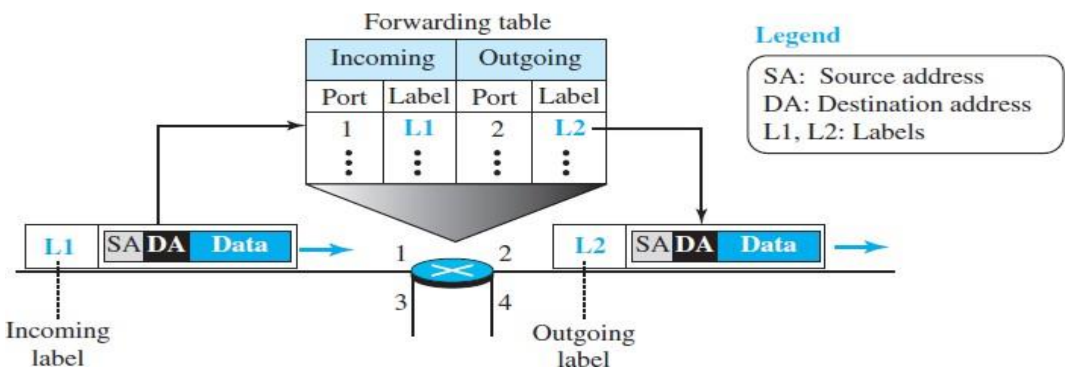


To create a connection-oriented service, a three-phase process is used:

- setup,
- data transfer
- teardown.

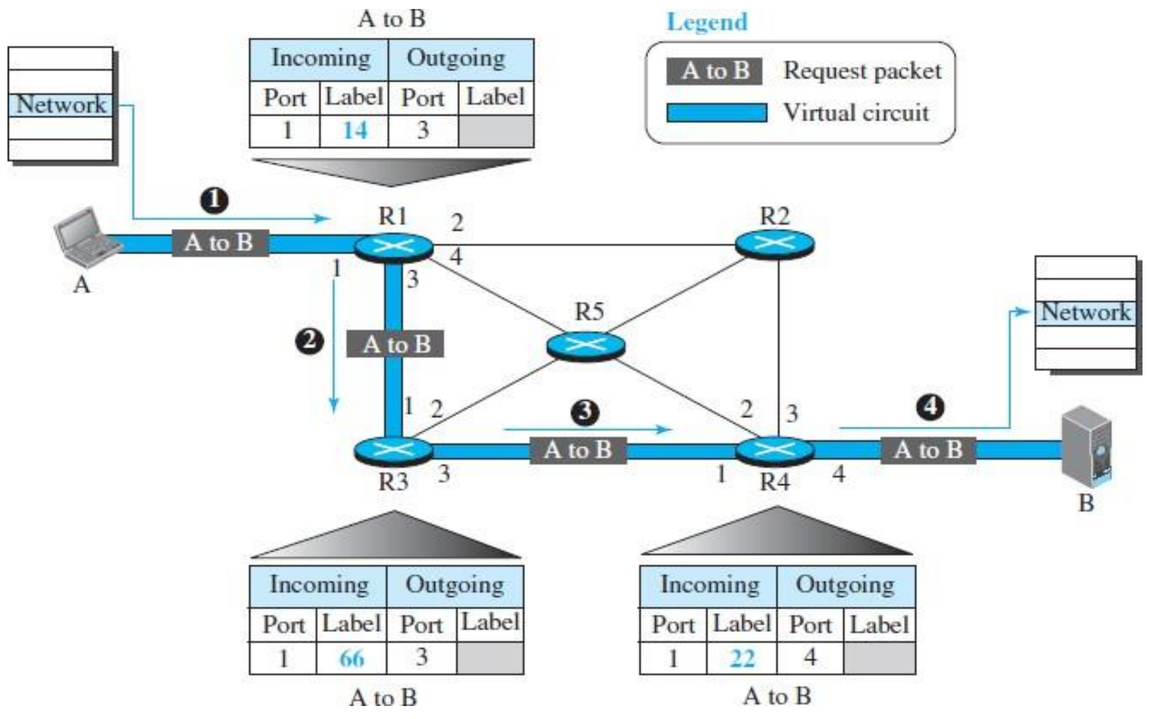
1. Setup Phase

- In the setup phase, a router creates an entry for a virtual circuit.
- For example, suppose source A needs to create a virtual circuit to destination B.
- Two auxiliary packets need to be exchanged between the sender and the receiver: the request packet and the acknowledgment packet.



Request packet

A request packet is sent from the source to the destination.

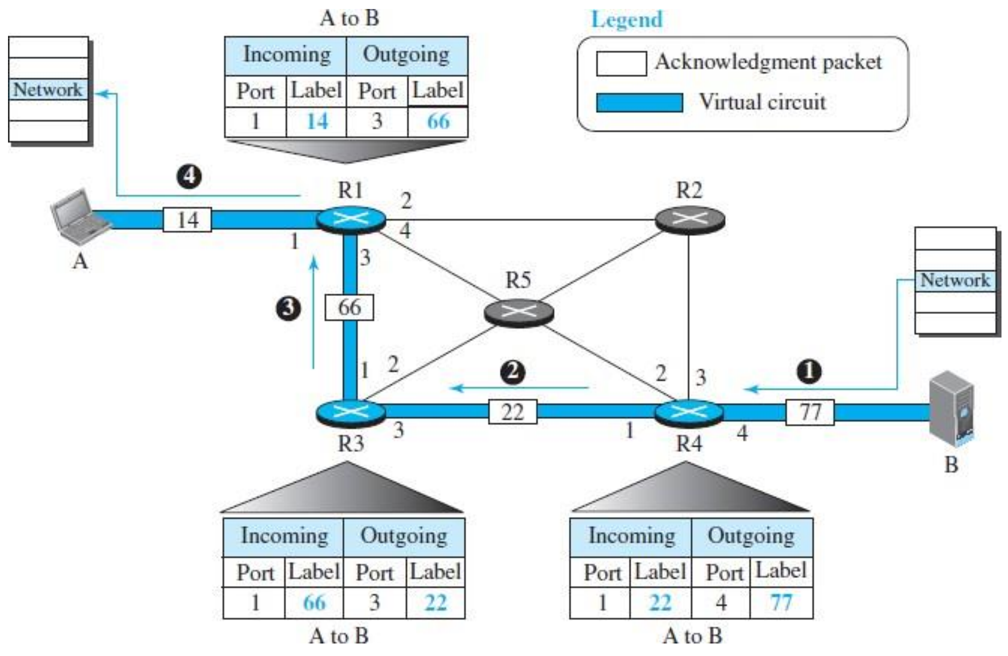


1. Source A sends a request packet to router R1.
2. Router R1 receives the request packet. It knows that a packet going from A to B goes out through port 3. The router creates an entry in its table for this virtual circuit. The router assigns the incoming port (1) and incoming label (14) and the outgoing port (3). The router then forwards the packet through port 3 to router R3.
3. Router R3 receives the setup request packet. The same events happen here as at router R1; three columns of the table are completed: in this case, incoming port (1), incoming label (66), and outgoing port (3).
4. Router R4 receives the setup request packet. Again, three columns are completed: incoming port (1), incoming label (22), and outgoing port (4).
5. Destination B receives the setup packet, and if it is ready to receive

packets from A, it assigns a label to the incoming packets that come from A, in this case 77. This label lets the destination know that the packets Come from A, and not from other sources.

Acknowledgment Packet

A special packet, called the acknowledgment packet, completes the entries in the switching tables.



1. The destination sends an acknowledgment to router R4. The acknowledgment carries the global source and destination addresses so the router knows which entry in the table is to be completed. The packet also carries label 77, chosen by the destination as the incoming label for packets from A. Router R4 uses this label to complete the outgoing label column for this entry. Note that 77 is the incoming label for destination B, but the outgoing label for router R4.

2. Router R4 sends an acknowledgment to router R3 that contains its incoming label in the table, chosen in the setup phase. Router R3 uses this as the outgoing label in the table.

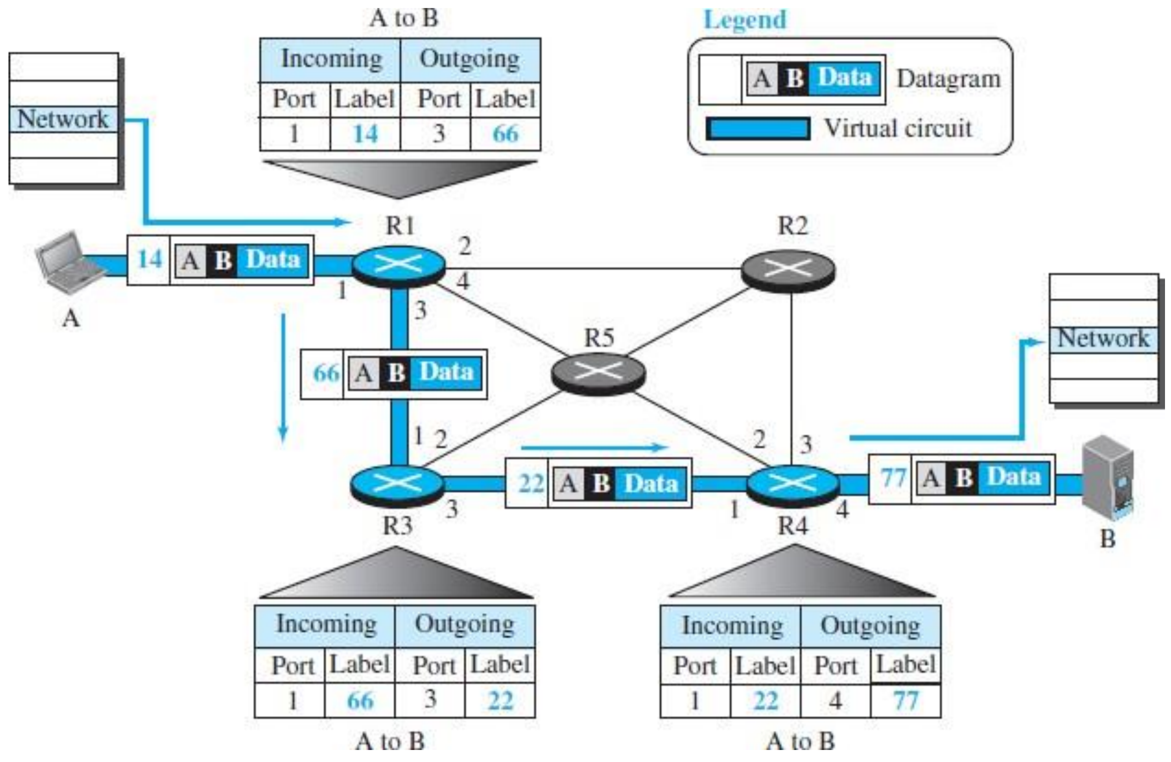
3. Router R3 sends an acknowledgment to router R1 that contains its incoming label in the table, chosen in the setup phase. Router R1 uses this as the outgoing label in the table.

4. Finally router R1 sends an acknowledgment to source A that contains its Incoming label in the table, chosen in the setup phase.

5. The source uses this as the outgoing label for the data packets to be sent to destination B.

2. Data-Transfer Phase

- The second phase is called the data-transfer phase.
- The source computer uses the label 14, which it has received from router R1 in the setup phase. Router R1 forwards the packet to router R3, but changes the label to 66.
- Router R3 forwards the packet to router R4, but changes the label to 22.
- Finally, router R4 delivers the packet to its final destination with the label 77.
- All the packets in the message follow the same sequence of labels, and the packets arrive in order at the destination.



3. Teardown Phase

- In the teardown phase, source A, after sending all packets to B, sends a special packet called a teardown packet.
- Destination B responds with a confirmation packet.
- All routers delete the corresponding entries from their tables.

3. Explain the performance of network layer in detail.

- The performance of a network can be measured in terms of
  - *delay,*
  - *throughput,*
  - *packet loss.*
- *Congestion control* is an issue that can improve the performance.

1 Delay

- All of us expect instantaneous response from a network, but a packet, from its source to its destination, encounters delays.
- The delays in a network can be divided into four types:
    - transmission delay,
    - propagation delay,
    - processing delay,
    - queuing delay.

➢ ***Transmission Delay***

- A sender needs to put the bits in a packet on the line one by one.
- If the first bit of the packet is put on the line at time t1 and the last bit is put on the line at time t2, transmission delay of the packet is (t2 − t1).
- The transmission delay is longer for a longer packet and shorter if the sender can transmit faster. In other words, the transmission delay is

$$\textbf{Delay}_{\textbf{tr}} = \textbf{(Packet length) / (Transmission rate).}$$

➢ ***Propagation Delay***

- Propagation delay is the time it takes for a bit to travel from point A to point B in the transmission media.
- The propagation delay depends on the propagation speed of the media, which is 3 × 108 meters/second in a vacuum and normally much less in a wired medium; it also depends on the distance of the link.

$$\textbf{Delay}_{\textbf{pg}} = \textbf{(Distance) / (Propagation speed).}$$

➢ ***Processing Delay***

- The processing delay is the time required for a router or a destination host to receive a packet from its input port, remove the header, perform an error detection procedure, and deliver the packet to the output port (in the case of a router) or deliver the packet to the upper-layer protocol (in the case of the destination host).

$$\textbf{Delay}_{\textbf{pr}} = \textbf{Time required to process a packet in a router or a destination host}$$

➢ ***Queuing Delay***

- The queuing delay for a packet in a router is measured as the time a packet waits in the input queue and output queue of a router.

$$\textbf{Delay}_{\textbf{qu}} = \textbf{The time a packet waits in input and output queues in a router}$$

➢ ***Total Delay***

- If we have *n* routers, we have (*n* + 1) links.
- Therefore, we have (*n* + 1) transmission delays related to *n* routers and the source, (*n* + 1) propagation delays related to (*n* + 1) links, (*n* + 1) processing delays related to *n* routers and the destination, and only *n* queuing delays related to *n* routers.

$$\textbf{Total delay} = (n + 1)\,(\text{Delay}_{tr} + \text{Delay}_{pg} + \text{Delay}_{pr}) + (n)\,(\text{Delay}_{qu})$$

2 Throughput

- Throughput at any point in a network is defined as the number of bits passing through the point in a second, which is actually the transmission rate of data at that point.
- In a path from source to destination, a packet may pass through several links (networks), each with a different transmission rate.

Throughput = **minimum {TR1, TR2,** . . . **TRn}.**

3 Packet Loss

- When a router receives a packet while processing another packet, the received packet needs to be stored in the input buffer waiting for its turn.
- A router, however, has an input buffer with a limited size. A time may come when the buffer is full and the next packet needs to be dropped.
- This effect is packet loss.

4 Congestion Control

- Congestion control is a mechanism for improving performance.
- Congestion control refers to techniques and mechanisms that can either prevent congestion before it happens or remove congestion after it has happened.
- Two broad categories:
  - open-loop congestion control (prevention)
  - closed-loop congestion control (removal).

➢ ***Open-Loop Congestion Control***

- In open-loop congestion control, policies are applied to prevent congestion before it happens.
- In these mechanisms, congestion control is handled by either the source or the destination.
- The policies are Retransmission Policy, Window Policy, Acknowledgment Policy, Discarding Policy, Admission Policy.

➢ ***Closed-Loop Congestion Control***.

- Closed-loop congestion control mechanisms try to alleviate congestion after it happens.
- Several mechanisms have been used by different protocols.
  - Backpressure
  - Choke Packet
  - Implicit Signalling
  - Explicit Signalling

4. Explain in detail IPv4 Addresses.

IPV4 ADDRESSES
1. Address Space
2. Classful Addressing
3. Classless Addressing

IPV4 ADDRESSES

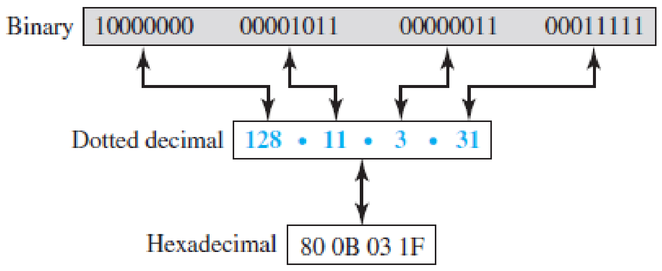
- The identifier used in the IP layer of the TCP/IP protocol suite to identify the connection of each device to the Internet is called the Internet address or IP address.
- The IP address is the address of the connection, not the host or the router, because if the device is moved to another network, the IP address may be changed.
- An IPv4 address is a 32-bit address that uniquely and universally defines the connection of a host or a router to the Internet.
- IPv4 addresses are unique. If a device has two connections to the Internet, via two networks, it has two IPv4 addresses.

1. Address Space

- A protocol like IPv4 that defines addresses has an address space.
- An address space is the total number of addresses used by the protocol.
- IPv4 uses 32-bit addresses, which means that the address space is 232 or 4,294,967,296 (more than four billion

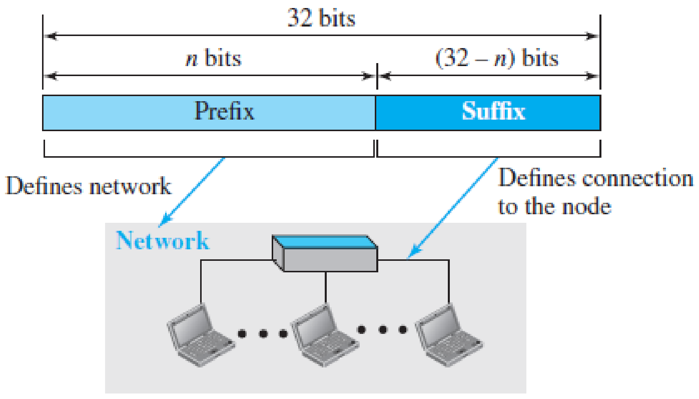
Notation

- There are three common notations to show an IPv4 address: binary notation (base 2), dotted-decimal notation (base 256), and hexadecimal notation (base 16).
- In binary notation, an IPv4 address is displayed as 32 bits.
- Dotted-decimal notation is decimal point (dot) separating the bytes.
- IPv4 address in hexadecimal notation. Each hexadecimal digit is equivalent to four bits. This means that a 32-bit address has 8 hexadecimal digits.



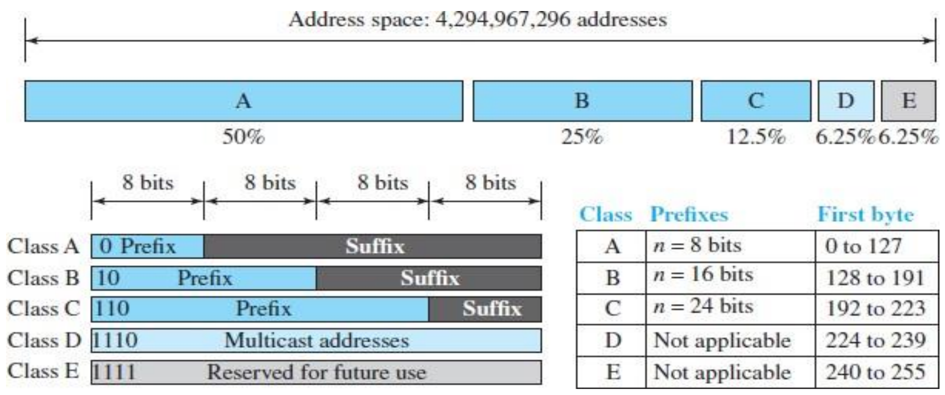
Hierarchy in Addressing

- A 32-bit IPv4 address is also hierarchical, but divided only into two parts. The first part of the address, called the prefix, defines the network; the second part of the address, called the suffix, defines the node (connection of a device to the Internet).
- The prefix length is n bits and the suffix length is (32 − n) bits. A prefix can be fixed length or variable length.

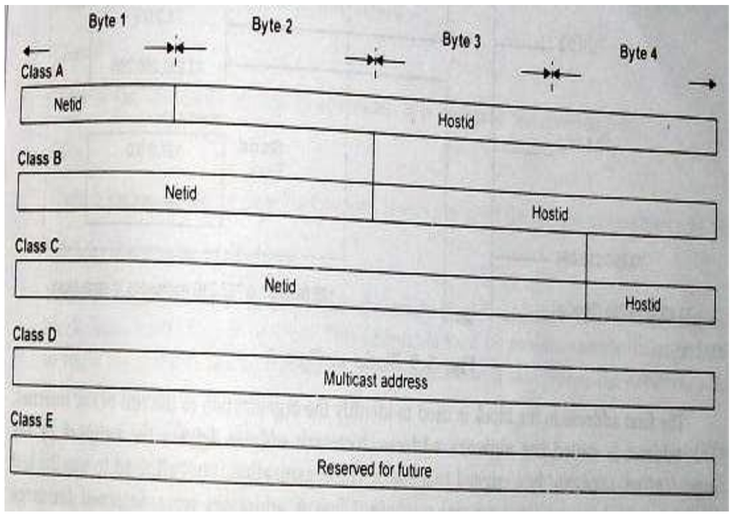


2 Classful Addressing

An IPv4 address was designed with a fixed-length prefix. The whole address space was divided into five classes (class A, B, C, D, and E). This scheme is referred to as classful addressing.



Class	Prefixes	First byte
A	$n = 8$ bits	0 to 127
B	$n = 16$ bits	128 to 191
C	$n = 24$ bits	192 to 223
D	Not applicable	224 to 239
E	Not applicable	240 to 255



- Addresses in classes A, B and C are for unicast communication, from one source to one destination.

- Addresses in class D are for multicast communication, from one source to a group of destination. A multicast address is used only in destination addresses.
- Addresses in class E are reserved. The original idea was to use them for special purpose.

Subnetting and Supernetting

- To alleviate address depletion, two strategies were proposed and implemented: subnetting and supernetting.
- In subnetting, a class A or class B block is divided into several subnets. Each subnet has a larger prefix length than the original network.
- Subnetting allows the addresses to be divided among several organizations.
- Supernetting was devised to combine several class C blocks into a larger block to be attractive to organizations that need more than the 256 addresses available in a class C block.

Advantage of Classful Addressing

Given an address, we can easily find the class of the address and, since the prefix length for each class is fixed, we can find the prefix length immediately.

Examples:

1. Find the class for the following IP addresses. (i) 205.55.43.11 and (ii) 100.23.28.65

Solution:

i) Class C (First byte 205 between 192 to 223)
ii) Class A (First byte 100 between 0 to 127)

2. Find the class for the following IP address:

i) 11110111 11110010 10000011 10101010 - Class E (First byte starts with 1111)
ii) 01111111 11110000 01010111 00001100 - Class A (First byte starts with 0)

3. Find the netid and hostid for the following:

i) 19.34.1.5 - netid = 19 hostid = 34.1.5
ii) 190.3.70.10 - netid = 190.3 hostid = 70.10
iii) 246.3.4.10 - No netid and no hostid because 246.3.4.10 is the class E address.
i) 201.2.4.2 - netid = 201.2.4 hostid = 2

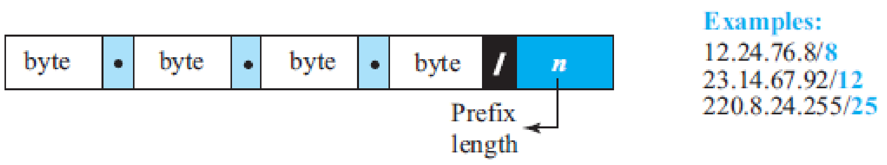
3 Classless Addressing

- In addressing, the whole address space is divided into variable length classless blocks.
- The prefix in an address defines the block (network); the suffix defines the node (device).
- A prefix length ranges from 0 to 32. The size of the network is inversely

- proportional to the length of the prefix. A small prefix means a larger network; a large prefix means a smaller network.

Prefix Length: Slash Notation

- The prefix length, n, is added to the address, separated by a slash.
- The notation is informally referred to as slash notation and formally as classless interdomain routing or CIDR strategy.
- An address in classless addressing can then be represented as shown in



Address Mask

- The address mask is a 32-bit number in which the n leftmost bits are set to 1s and the rest of the bits (32 − n) are set to 0s.
- To extract the information in a block, using the three bit-wise operations NOT, AND, and OR.

1. The number of addresses in the block N = NOT (mask) + 1.
2. The first address in the block = (Any address in the block) AND (mask).
3. The last address in the block = (Any address in the block) OR [(NOT (mask)].

Network Address

- The network address is actually the identifier of the network; because it is used in routing a packet to its destination network.

5. Explain the forwarding of IP packets in detail.

FORWARDING OF IP PACKETS

- Forwarding Based on Destination Address
- Forwarding Based on Label
- Routers as Packet Switches

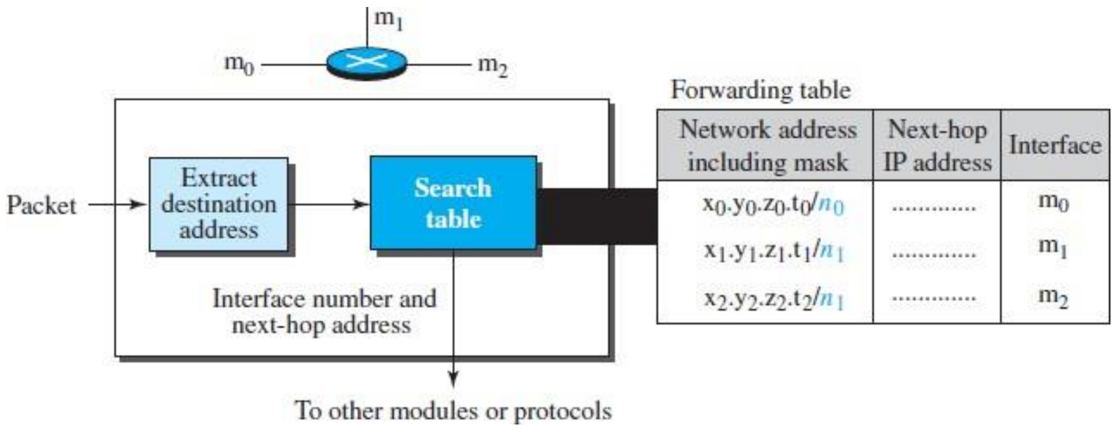
➢ **FORWARDING OF IP PACKETS**

- Forwarding means to place the packet in its route to its destination.
- When IP is used as a connectionless protocol, forwarding is based on the destination address of the IP datagram; when the IP is used as a connection-oriented protocol, forwarding is based on the label attached to an IP datagram.

➢ **Forwarding Based on Destination Address**

- Forwarding requires a host or a router to have a forwarding table.
- When a host has a packet to send or when a router has received a packet to be forwarded, it looks at this table to find the next hop to deliver the packet to.

- The table needs to be searched based on the network address.
- Unfortunately, the destination address in the packet gives no clue about the network address.
- To solve the problem, we need to include the mask (/*n*) in the table.
- A classless forwarding table needs to include four pieces of information: the mask, the network address, the interface number, and the IP address of the next router.
- For example, if *n* is 26 and the network address is 180.70.65.192, then one can combine the two as one piece of information: 180.70.65.192/**26**.



m_1
m_0
m_2
Forwarding table
Packet
Extract destination address
Search table
Interface number and next-hop address
To other modules or protocols

Network address including mask	Next-hop IP address	Interface
$x_0.y_0.z_0.t_0/n_0$	....	m_0
$x_1.y_1.z_1.t_1/n_1$	....	m_1
$x_2.y_2.z_2.t_2/n_1$	....	m_2

➢ **Forwarding Based on Label**

- In a connection-oriented network (virtual-circuit approach), a switch forwards a packet based on the label attached to the packet.
- When the forwarding algorithm gets the destination address of the packet, it needs to apply the mask to find the destination network address.
- It then needs to check the network addresses in the table until it finds the match.
- The router then extracts the next-hop address and the interface number to be delivered to the data-link layer.

➢ **Routers as Packet Switches**

- The packet switches that are used in the network layer are called routers.
- Routers can be configured to act as either a datagram switch or a virtual-circuit switch.

6. Explain about INTERNET PROTOCOL (IP) in detail. [May/June 2014]

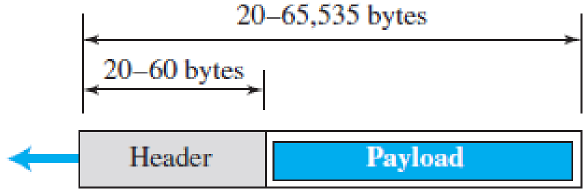
➢ **INTERNET PROTOCOL (IP)**

- The Internet Protocol version 4 (IPv4), is responsible for packetizing, forwarding, and delivery of a packet at the network layer.

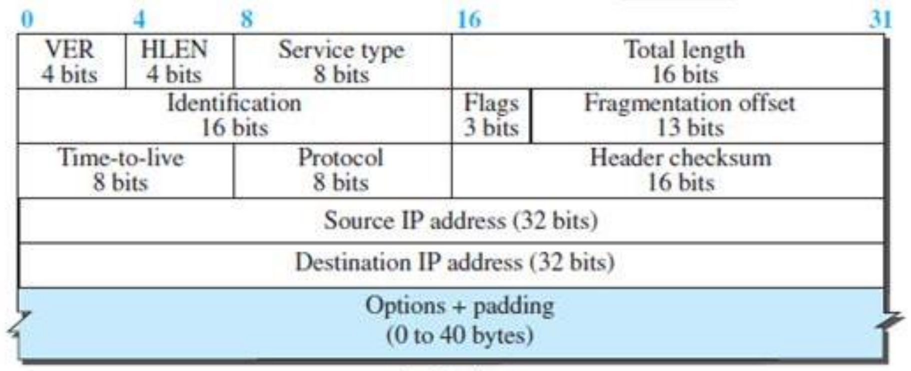
Datagram Format

- Packets used by the IP are called *datagrams*.
- A datagram is a variable-length packet consisting of two parts: header and payload (data).

- The header is 20 to 60 bytes in length and contains information essential to routing and delivery.



a. IP datagram



b. Header

- ***Version Number.***
  - The 4-bit version number (VER) field defines the version of the IPv4 protocol, which, obviously, has the value of 4.
- ***Header Length.***
  - The 4-bit header length (HLEN) field defines the total length of the datagram header in 4-byte words. The IPv4 datagram has a variable-length header.
- ***Service Type.***
  - **T**ype of service (TOS), defines how the datagram should be handled.
- ***Total Length.***
  - This 16-bit field defines the total length (header plus data) of the IP datagram in bytes
- ***Identification, Flags, and Fragmentation Offset.***
  - These three fields are related to the fragmentation of the IP datagram when the size of the datagram is larger than the underlying network can carry.
- ***Time-to-live.***
  - The time-to-live (TTL) field is used to control the maximum number of hops (routers) visited by the datagram. When a source host sends the datagram, it stores a number in this field.
  - Each router that processes the datagram decrements this number by one. If this value, after being decremented, is zero, the router discards the datagram.
- ***Protocol.***
  - In TCP/IP, the data section of a packet, called the *payload*, carries the whole packet from another protocol.

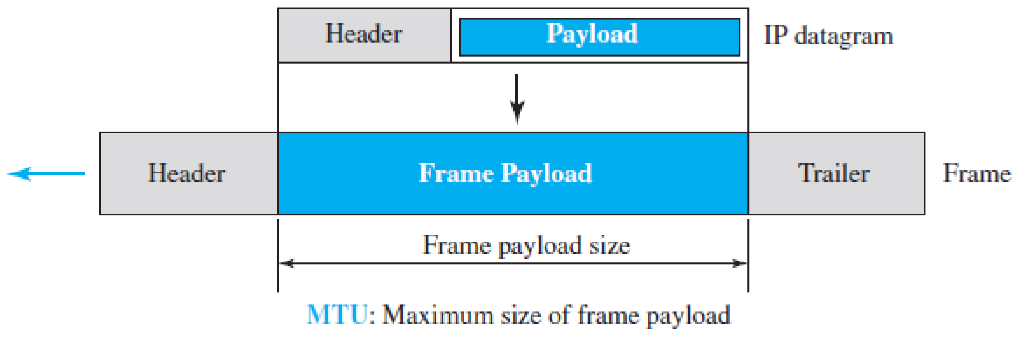
  - A datagram, for example, can carry a packet belonging to any transport-layer protocol such as UDP or TCP.
- ***Header checksum.***
  - IP adds a header checksum field to check the header for error control, but not the payload.
  - Since the value of some fields, such as TTL, which are related to fragmentation and options, may change from router to router, the checksum needs to be recalculated at each router.
- ***Source and Destination Addresses.***
  - These 32-bit source and destination address fields define the IP address of the source and destination respectively.
  - The source host should know its IP address.
  - The destination IP address is either known by the protocol that uses the service of IP or is provided by the DNS.
- ***Options.***
  - A datagram header can have up to 40 bytes of options. Options can be used for network testing and debugging.
- ***Payload.***
  - Payload is the packet coming from other protocols that use the service of IP.

Fragmentation

- The division of a packet into smaller units to accommodate a protocol's MTU.

Maximum Transfer Unit (MTU)

- The largest size data unit a specific network can handle.



- The value of the MTU differs from one physical network protocol to another.
- When a datagram is fragmented, each fragment has its own header with most of the fields repeated, but some have been changed.
- A datagram may be fragmented several times before it reaches the final destination.
- The host or router that fragments a datagram must change the values of three fields: flags, fragmentation offset, and total length.

Three fields in an IP datagram are related to fragmentation:

identification, flags, and *fragmentation offset.*

- The 16-bit *identification field* identifies a datagram originating from the source host.
- The 3-bit *flags field* defines three flags.
  - The leftmost bit is reserved (not used).
  - The second bit (D bit) is called the *do not fragment* bit.
    - If its value is 1, the machine must not fragment the datagram.
    - If its value is 0, the datagram can be fragmented if necessary.
  - The third bit (M bit) is called the *more fragment bit*.
    - If its value is 1, it means the datagram is not the last fragment; there are more fragments after this one.
    - If its value is 0, it means this is the last or only fragment.
- The 13-bit *fragmentation offset field* shows the relative position of this fragment with respect to the whole datagram.

7. Explain about Internet Control Message Protocol version 4 (ICMPv4) in detail.

ICMPv4
- MESSAGES
- Debugging Tools
- ICMP Checksum

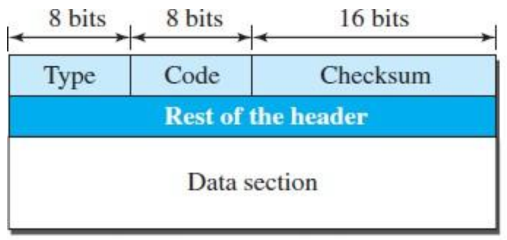
- The Internet Control Message Protocol version 4 (ICMPv4) helps IPv4 to handle some errors that may occur in the network-layer delivery.
- ICMP is used to report some errors that may occur during the processing of the IP datagram. ICMP does not correct errors, it simply reports them.

➢ **MESSAGES**

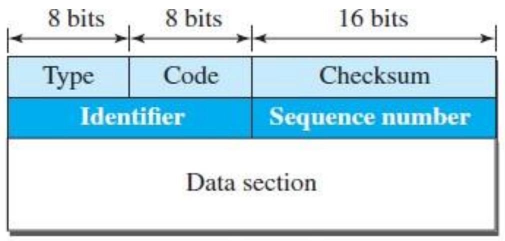
- ICMP messages are divided into two broad categories:

  error-reporting messages **and** ***query messages***

- An ICMP message has an 8-byte header and a variable-size data section.
- The first field, ICMP type, defines the type of the message.
- The code field specifies the reason for the particular message type.
- The last common field is the checksum field.
- The rest of the header is specific for each message type.
- The data section in error messages carries information for finding the original packet that had the error.
- In query messages, the data section carries extra information based on the type of query.



Error-reporting messages



Query messages

Type and code values

Error-reporting messages
03: Destination unreachable (codes 0 to 15)
04: Source quench (only code 0)
05: Redirection (codes 0 to 3)
11: Time exceeded (codes 0 and 1)
12: Parameter problem (codes 0 and 1)

Query messages
08 and 00: Echo request and reply (only code 0)
13 and 14: Timestamp request and reply (only code 0)

Error Reporting Messages

- The error-reporting messages report problems that a router or a host (destination) may encounter when it processes an IP packet.
- To make the error-reporting process simple, ICMP follows some rules in
- reporting messages.
  - First, no error message will be generated for a datagram having a multicast address or special address (such as *this host* or *loopback*).
  - Second, no ICMP error message will be generated in response to a datagram carrying an ICMP error message.
  - Third, no ICMP error message will be generated for a fragmented datagram that is not the first fragment.

1. Destination Unreachable

- The most widely used error message is the destination unreachable (type 3).
- This message uses different codes (0 to 15) to define the type of error message and the reason why a datagram has not reached its final destination.

2. Source Quench

- It informs the sender that the network has encountered congestion and the datagram has been dropped; the source needs to slow down sending more datagrams.

3. Redirection Message

- The *redirection message* (type 5) is used when the source uses a wrong router to send out its message.
- The router redirects the message to the appropriate router, but informs the source that it needs to change its default router in the future.

***4.* Time Exceeded**

- When the TTL value becomes 0, the datagram is dropped by the visiting router and a *time exceeded* message (type 11) with code 0 is sent to the source to inform it about the situation.

- The time-exceeded message (with code 1) can also be sent when not all fragments of a datagram arrive within a predefined period of time.

5. Parameter Problem

- A *parameter problem message* (type 12) can be sent when either there is a problem in the header of a datagram (code 0) or some options are missing or cannot be interpreted (code 1).

Query Messages

- Query messages are used to probe or test the liveliness of hosts or routers in the Internet.
- The query messages come in pairs: request and reply.
- The *echo request* (type 8) and the *echo reply* (type 0) pair of messages are used by a host or a router to test the liveliness of another host or router. A host or router sends an echo request message to another host or router; if the latter is alive, it responds with an echo reply message.

➢ **Debugging Tools**

- Two debugging tools: *ping* and *traceroute*.

Ping

- *Ping* program is used to find if a host is alive and responding.
- The source host sends ICMP echo-request messages; the destination, if alive, responds with ICMP echo-reply messages.
- The ping program gets help from two query messages;

Traceroute or Tracert

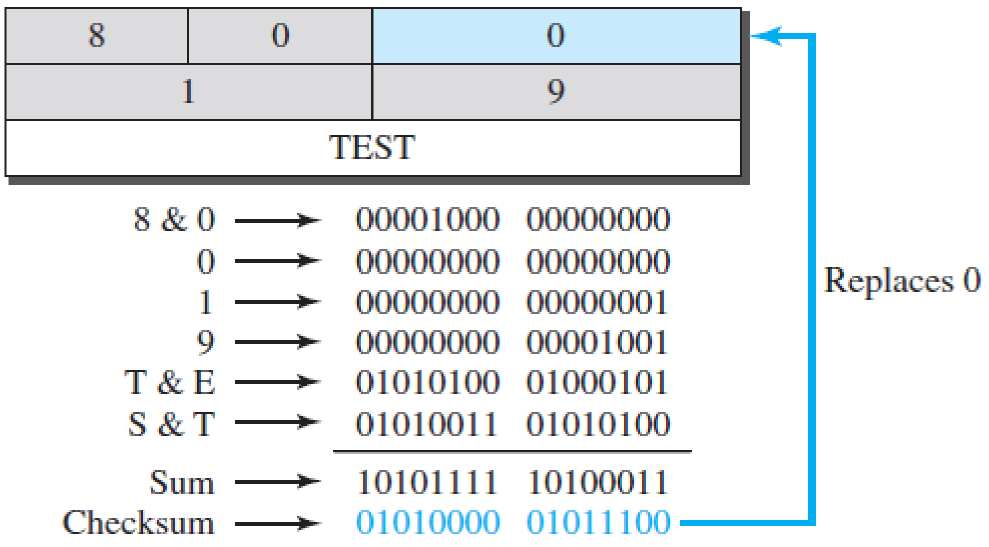
- The *traceroute* program in UNIX or *tracert* in Windows can be used to trace the path of a packet from a source to the destination.
- It can find the IP addresses of all the routers that are visited along the path.
- the *traceroute* program gets help from two error-reporting messages: time-exceeded and destination-unreachable.

➢ **ICMP Checksum**

- In ICMP the checksum is calculated over the entire message (header and data).

Example

An example of checksum calculation for a simple echo-request message. We randomly chose the identifier to be 1 and the sequence number to be 9. The message is divided into 16-bit (2-byte) words. The words are added and the sum is complemented. Now the sender can put this value in the checksum field.



8. Elaborate the various routing algorithms or unicast routing algorithms in detail.

- **ROUTING ALGORITHMS**
  - Distance-Vector Routing
  - Link-State Routing
  - Path-Vector Routing

Distance-Vector (DV) Routing

- In distance-vector routing, the first thing each node creates is its own least-cost tree with the rudimentary information it has about its immediate neighbours.
- The incomplete trees are exchanged between immediate neighbours to make the trees more and more complete and to represent the whole internet.

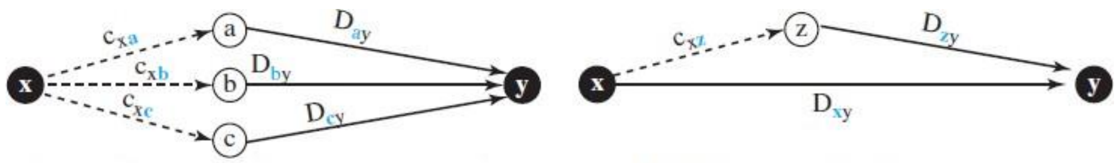
Bellman-Ford Equation

- This equation is used to find the least cost (shortest distance) between a source node, ***x***, and a destination node, ***y***, through some intermediary nodes (**a, b, c,** . . .).
- The following shows the general case in which D***ij*** is the shortest distance and c***ij*** is the cost between nodes ***i*** and ***j***.

$$D_{xy} = \min\{(c_{xa} + D_{ay}), (c_{xb} + D_{by}), (c_{xc} + D_{cy}), \ldots\}$$

- In distance-vector routing, normally we want to update an existing least cost with a least cost through an intermediary node, such as ***z***, if the latter is shorter.

$$D_{xy} = \min\{D_{xy}, (c_{xz} + D_{zy})\}$$

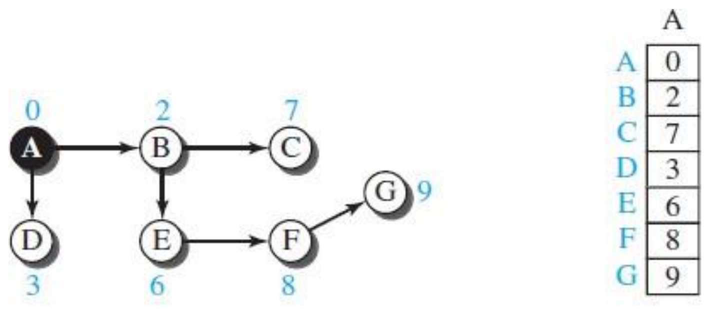


a. General case with three intermediate nodes b. Updating a path with a new route

(a→y), (b→y), and (c→y) as previously established least-cost paths and (x→y) as the new least-cost path.

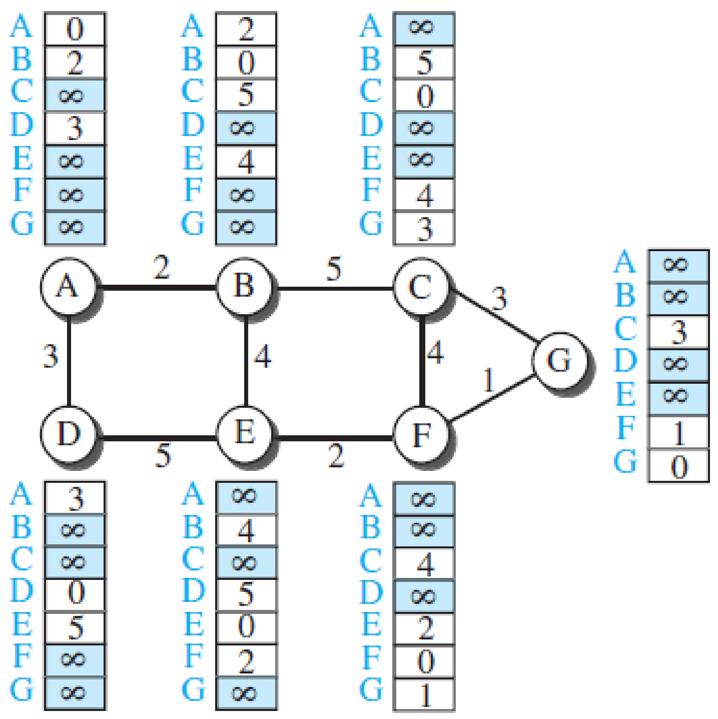
Distance Vectors

- A least-cost tree is a combination of least-cost paths from the root of the tree to all destinations.
- These paths are graphically glued together to form the tree.



	A
A	0
B	2
C	7
D	3
E	6
F	8
G	9

a. Tree for node A b. Distance vector for node A



	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

Distance-Vector Routing Algorithm

```
1       Distance_Vector_Routing ( )
2       {
3         // Initialize (create initial vectors for the node)
4               D[myself ] = 0
5               for (y = 1 to N)
6               {
7                    if (y is a neighbour)
8                         D[y] = c[myself ][y]
9                    else
10                        D[y] = ∞
11              }
12              send vector {D[1], D[2], …, D[N]} to all neighbours

13      // Update (improve the vector with the vector received from a
        neighbour)
14              repeat (forever)
15              {
16                   wait (for a vector Dw from a neighbour w or any change
                  in the link)
17                   for (y = 1 to N)
18                   {
19                        D[y] = min [D[y], (c[myself ][w] + Dw[y ])]
                                                  // Bellman-Ford equation
20                   }
21                   if (any change in the vector)
22              send vector {D[1], D[2], …, D[N]} to all neighbours23    }

24        } // End of Distance Vector
```

- Lines 4 to 11 initialize the vector for the node.
- Lines 14 to 23 show how the vector can be updated after receiving a vector from the immediate neighbour.
- The *for* loop in lines 17 to 20 allows all entries (cells) in the vector to be updated after receiving a new vector.
- Note that the node sends its vector in line 12, after being initialized, and in line 22, after it is updated.

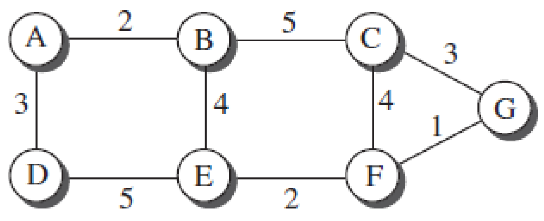
➢ **Link-State Routing**

- This method uses the term *link-state* to define the characteristic of a link (an edge) that represents a network in the internet.

- Links with lower costs are preferred to links with higher costs; if the cost of a link is infinity, it means that the link does not exist or has been broken.

Link-State Database (LSDB)

- The LSDB can be represented as a two-dimensional array(matrix) in which the value of each cell defines the cost of the corresponding link.



a. The weighted graph

	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Link state database

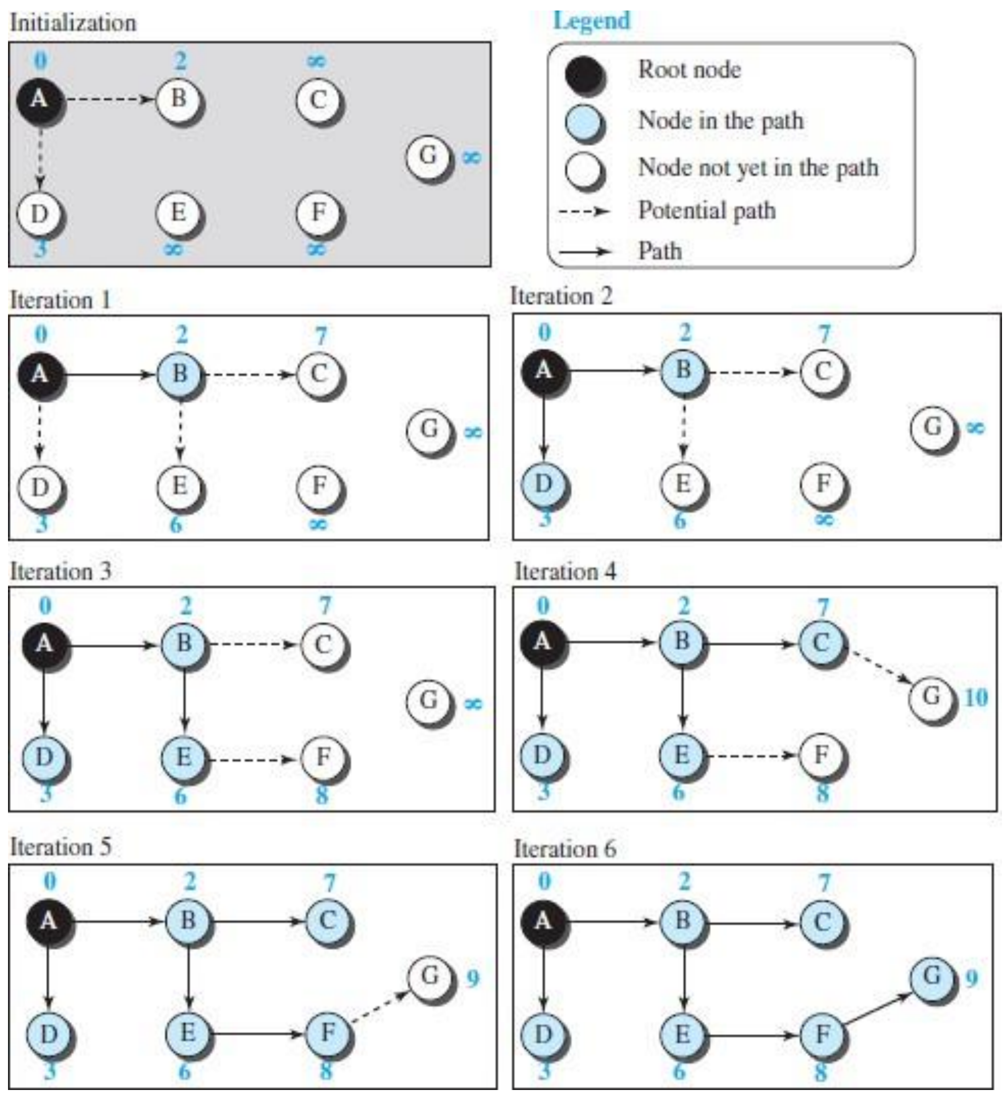
Formation of Least-Cost Trees

- To create a least-cost tree for itself, using the shared LSDB, each node needs to run the famous Dijkstra Algorithm.
- This iterative algorithm uses the following steps:
  1. The node chooses itself as the root of the tree, creating a tree with a single node, and sets the total cost of each node based on the information in the LSDB.
  2. The node selects one node, among all nodes not in the tree, which is closest to the root, and adds this to the tree. After this node is added to the tree, the cost of all other nodes not in the tree needs to be updated because the paths may have been changed.
  3. The node repeats step 2 until all nodes are added to the tree.

Dijkstra’s Algorithm

```
Dijkstra’s Algorithm ( )
{
    // Initialization
    Tree = {root} // Tree is made only of the root
    for (y = 1 to N) // N is the number of nodes
    {
        if (y is the root)
            D[y] = 0
        // D[y] is shortest distance from root to node y
        else if (y is a neighbour)
            D[y] = c[root][y]
        // c[x][y] is cost between nodes x and y in LSDB
        else
            D[y] = ∞
    }
    // Calculation
    repeat
    {
        find a node w, with D[w] minimum among all nodes not
        in the Tree
        Tree = Tree ∪ {w} // Add w to tree
        // Update distances for all neighbours of w
        for (every node x, which is a neighbour of w and not in the Tree)
        {
            D[x] = min{D[x], (D[w] + c[w][x])}
        }
    } until (all nodes included in the Tree)
} // End of Dijkstra
```

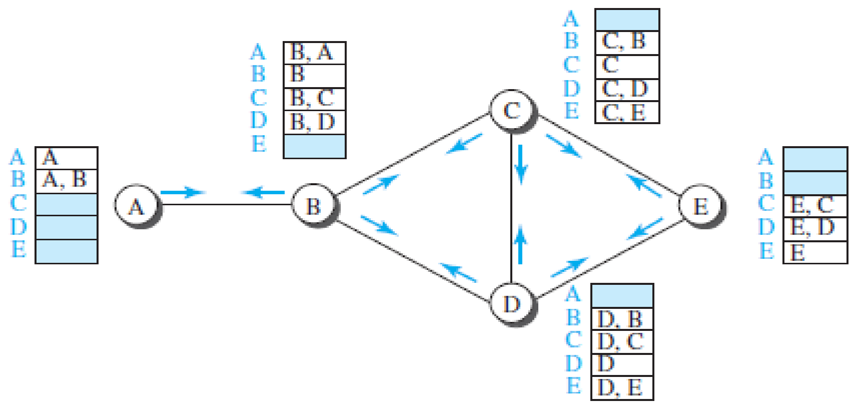
Example:



➢ **Path-Vector Routing**

- In path-vector routing, the path from a source to all destinations is also determined by the best spanning tree.
- The best spanning tree, however, is not the least-cost tree; it is the tree determined by the source when it imposes its own policy.
- Path-vector routing, is an asynchronous and distributed routing algorithm.

$$\text{Path}(x, y) = \text{best}\,\{\text{Path}(x, y), [(x + \text{Path}(v, y)]\} \quad \text{for all } v\text{'s in the internet.}$$



Path-vector algorithm for a node

```
Path_Vector_Routing ( )
{
// Initialization
    for (y = 1 to N)
    {
        if (y is myself)
            Path[y] = myself
        else if (y is a neighbour)
            Path[y] = myself + neighbour node
        else
            Path[y] = empty
    }
    Send vector {Path[1], Path[2], …, Path[y]} to all neighbours
// Update
    repeat (forever)
    {
        wait (for a vector Path from a neighbour w)
        for (y = 1 to N)
        {
        if (path includes myself)
            discard the path // Avoid any loop
        else
            Path[y] = best {Path[y], (myself + Path w[y])}
        }
        If (there is a change in the vector)
            Send vector {Path[1], Path[2], …, Path[y]} to all
            neighbours
    }
} // End of Path Vector
```

9. Discuss in detail about Routing algorithms. [May/June 2014]-Nov-15

ROUTING ALGORITHMS PROTOCOL

- Routing Information Protocol (RIP), based on the distance-vector algorithm,
- Open Shortest Path First (OSPF), based on the link-state algorithm,
- Border Gateway Protocol (BGP), based on the path-vector algorithm.

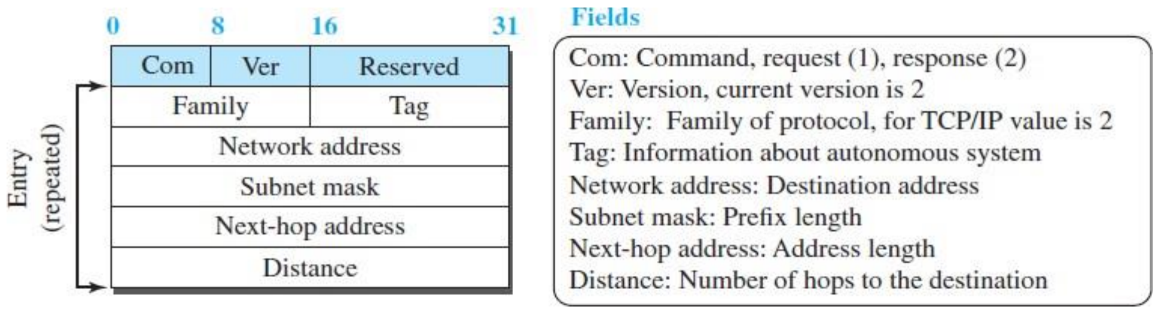
➢ **Routing Information Protocol (RIP)**

- The **Routing Information Protocol** (**RIP**) is one of the most widely used intradomain routing protocols based on the distance-vector routing algorithm.

RIP Implementation

- RIP is implemented as a process that uses the service of UDP on the well-known port number 520. RIP messages are encapsulated inside UDP user datagrams, which in turn are encapsulated inside IP datagrams.

RIP Messages

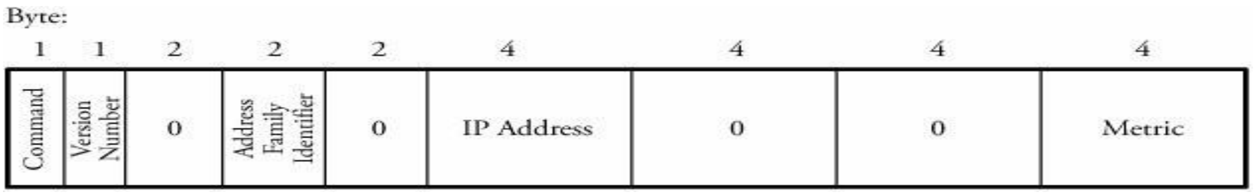


Com: Command, request (1), response (2)
Ver: Version, current version is 2
Family: Family of protocol, for TCP/IP value is 2
Tag: Information about autonomous system
Network address: Destination address
Subnet mask: Prefix length
Next-hop address: Address length
Distance: Number of hops to the destination

- RIP has two types of messages: request and response.
- A request message is sent by a router that can ask about specific entries or all entries.
- A response (or update) message can be either solicited or unsolicited. A solicited response message is sent only in answer to a request message. It contains information about the destination specified in the corresponding request message.
- An unsolicited response message, on the other hand, is sent periodically, every 30 seconds or when there is a change in the forwarding table.

RIP Packet Format

- ▪ Each packet consists of several address distances.
- ▪ The header format contains the following specifications for the first address distance.



- **Command** indicates a request with value 1 or a reply with value 2.
- **Version number** specifies the version: RIP-1 or RIP-2.
- **Address family identifier** shows the type of address, such as an IP address.

- **IP address** provides the IP address in a particular network.
- **Metric** identifies the distance from a router to a specified network.

Timers in RIP

- RIP uses three timers to support its operation.
  - The *periodic timer* controls the advertising of regular update messages.
  - The *expiration timer* governs the validity of a route.
  - The *garbage collection timer* is used to purge a route from the forwarding table.

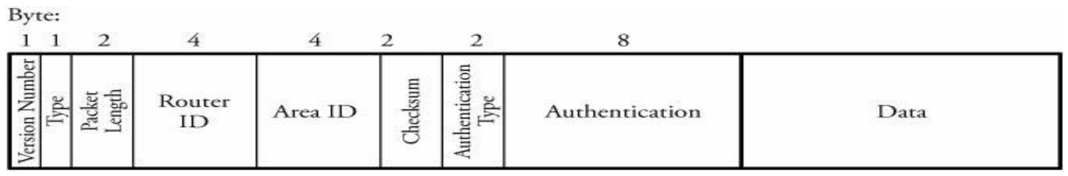
Performance

- ***Update Messages.*** The update messages in RIP have a very simple format and are sent only to neighbours; they are local.
- ***Convergence of Forwarding Tables*****.** RIP uses the distance-vector algorithm, which can converge slowly if the domain is large, but, since RIP allows only 15 hops in a domain, there is normally no problem in convergence.
- ***Robustness*****.** If there is a failure or corruption in one router, the problem will be propagated to all routers and the forwarding in each router will be affected.

➢ Open Shortest Path First (OSPF)

Open Shortest Path First (**OSPF**) is also an intradomain routing protocol. OSPF is an *open* protocol, which means that the specification is a public document.

OSPF Packet Format



All OSPF packets use a 24-byte header as follows:

- Version number indicates the version of OSPF.
- Type is one of the five types of packets for OSPF to choose from: hello, database description, link-state request, link-state update, and link-state acknowledgment.
- Packet length specifies the length of the OSPF packet.
- Router ID specifies the packet's source router ID.
- Area ID refers to the area that the source router belongs to.
- Checksum specifies the standard IP checksum of the packet contents.
- Authentication type identifies which authentication method to choose.
- Authentication specifies the authentication method.

OSPF Messages

- OSPF is a very complex protocol; it uses five different types of messages.
- The ***hello*** message (type 1) is used by a router to introduce itself to the neighbours and announce all neighbours that it already knows.
- The ***database description*** message (type 2) is normally sent in response to the hello message to allow a newly joined router to acquire the full LSDB.
- The ***link-state request*** message (type 3) is sent by a router that needs information about a specific LS.
- The ***link-state update*** message (type 4) is the main OSPF message used for building the LSDB.
- The ***link-state acknowledgment*** message (type 5) is used to create reliability in OSPF; each router that receives a link-state update message needs to acknowledge it.

OSPF Algorithm

- OSPF implements the link-state routing.

***Performance* of OSPF**:

- Update Messages.
- Convergence of Forwarding Tables.
- Robustness.

➢ *Border Gateway Protocol (BGP)*

- The Border Gateway Protocol version 4 (BGP4) is the only interdomain routing protocol, based on the path-vector algorithm.
- BGP allows routers to carry specific policies or constraints that they must meet.
- In BGP, two contributing (casual) routers can exchange routing information even if they are located in two different autonomous systems.

Details of BGP

- BGP has three functional components:
    - Neighbor relationship
    - Neighbor maintenance
    - Network maintenance
- The neighbor relationship refers to an agreement between two routers in two different autonomous systems to exchange routing information on a regular basis.
- A router may reject its participation in establishing a neighbor relationship for several reasons, such as the rule of the domain, overload, or a temporary malfunctioning of external links.
- Neighbor maintenance is a process of maintaining the neighbor relationship already established.
- For this reason, two routers send keep-alive messages to each other. The last BGP process is network maintenance.

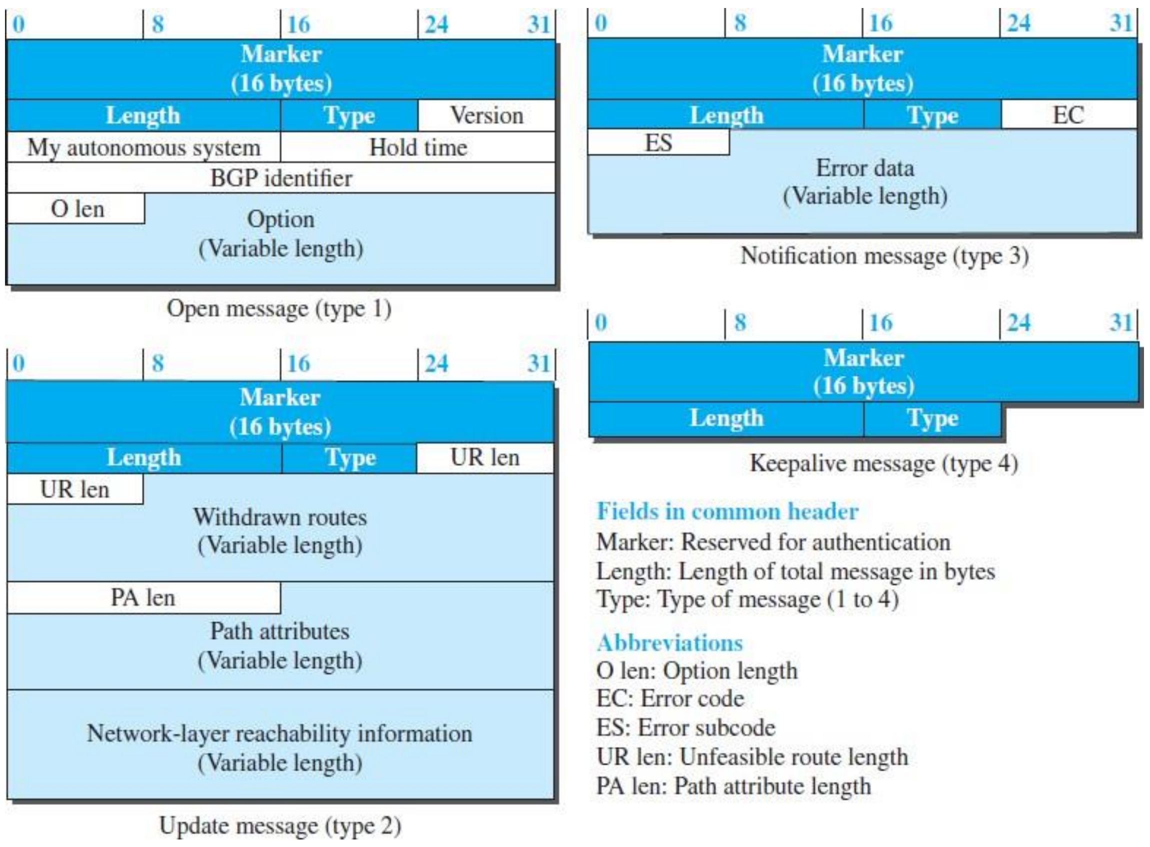
- Each router keeps the database of the subnetworks that it can reach and tries to get the best route for that subnetwork.

BGP Packets / Messages

- **Open packet.** This packet requests establishment of a relationship between two routers.
- **Update packet**. This packet conveys update information about routes.
- **Keep-alive packet**. Once a relationship between two routers is established, this packet confirms its neighbor relationship frequently.
- **Notification packet.** This packet is used when an error occurs.

Performance

- BGP speakers exchange a lot of messages to create forwarding tables, but BGP is free from loops and count-to-infinity.



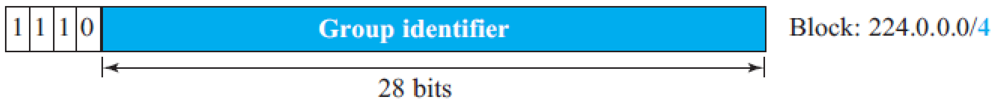
10. Explain in detail about multicasting basics.

- **MULTICASTING BASICS**
  - Multicast Addresses
  - Delivery at Data-Link Layer
  - Collecting Information about Groups
  - Multicast Forwarding
  - Two Approaches to Multicasting

➢ **Multicast Addresses**

- In multicast communication, the sender is only one, but the receiver is many.

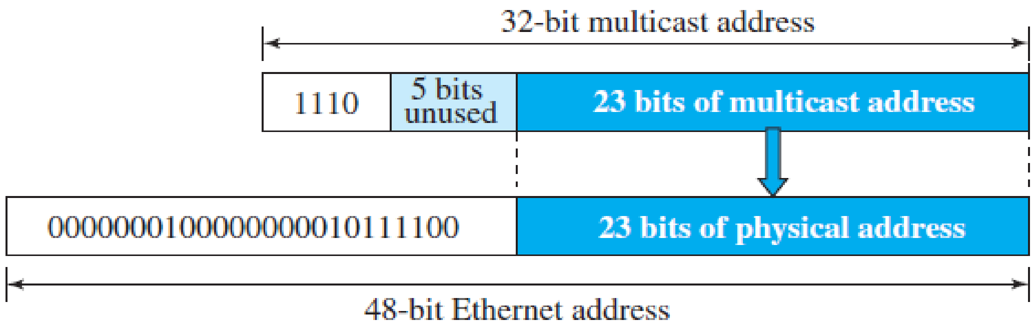
- A multicast address defines a group of recipients, not a single one.
- In other words, a multicast address is an identifier for a group.
- A host, which is a member of *n* groups, actually has (*n* + 1) addresses: one unicast address that is used for source or destination address in unicast communication and *n* multicast addresses that are used only for destination addresses to receive messages sent to a group.
- In classful addressing, all of class D was composed of these addresses; classless addressing used the same block, but it was
- referred to as the block 224.0.0.0/**4** (from 224.0.0.0 to 239.255.255.255).



➢ **Delivery at Data-Link Layer**

- Data-link layer multicast addresses are also needed to deliver a multicast packet encapsulated in a frame.

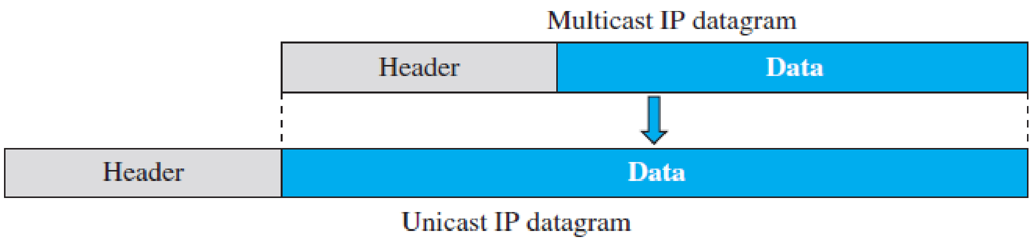
Mapping class D to Ethernet physical address



- Most LANs support physical multicast addressing. Ethernet is one of them.
- An Ethernet physical address (MAC address) is six octets (48 bits) long.
- If the first 25 bits in an Ethernet address are 00000001 00000000 01011110 0, this identifies a physical multicast address for the TCP/IP protocol.
- The remaining 23 bits can be used to define a group.
- To convert an IP multicast address into an Ethernet address, the multicast router extracts the least significant 23 bits of a multicast IP address and inserts them into a multicast Ethernet physical address

Network with No Multicast Support

- To send a multicast packet through WANs, a process called *tunnelling* is used. In **tunnelling,** the multicast packet is encapsulated in a unicast packet and sent through the network, where it emerges from the other side as a multicast packet



➢ **Collecting Information about Groups**

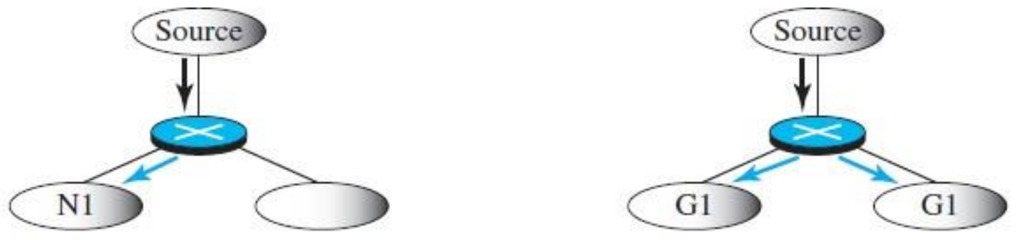
Creation of forwarding tables in both unicast and multicast routing involves two steps:

1. A router needs to know to which destinations it is connected.
2. Each router needs to propagate information obtained in the first step to all Other routers so that each router knows to which destination each other router is connected.

➢ **Multicast Forwarding**

1. In unicast communication, the destination address of the packet defines one single destination.

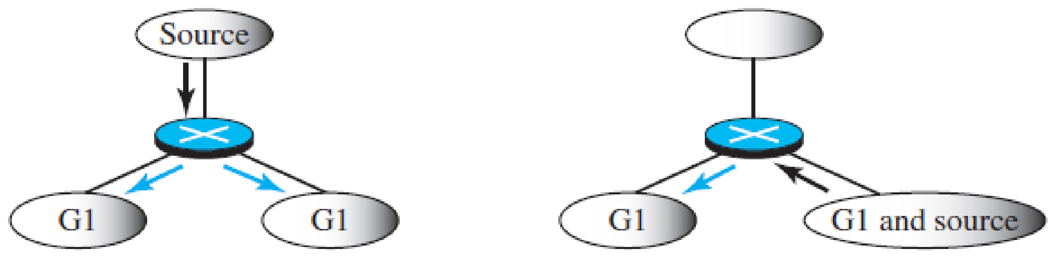
In multicast communication, the destination of the packet defines one group, but that group may have more than one member in the internet.



a. Destination in unicasting is one b. Destination in mulicasting is more than one

2. Forwarding decisions in unicast communication depend only on the Destination address of the packet.

Forwarding decisions in multicast communication depend on both the destination and the source address of the packet



a. Packet sent out of two interfaces b. Packet sent out of one interface

➢ **Two Approaches to Multicasting**

1. Source-Based Tree Approach

- In the source-based tree approach to multicasting, each router needs to create a separate tree for each source-group combination.

- If there are m groups and n sources in the internet, a router needs to create (m × n) routing trees.
- In each tree, the corresponding source is the root, the members of the group are the leaves, and the router itself is somewhere on the tree.

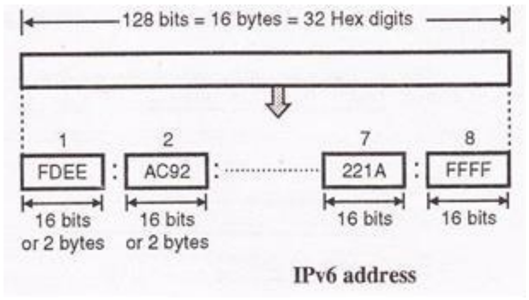
2. Group-Shared Tree Approach

- In the **group-shared tree** approach, the designated router, which is called the *core* router or the *rendezvous point* router, acts as the representative for the group.
- Any source that has a packet to send to a member of that group sends it to the core center (unicast communication) and the core center is responsible for multicasting.
- The core center creates one single routing tree with itself as the root and any routers with active members in the group as the leaves.
- In this approach, there are *m* core routers (one for each group) and each core router has a routing tree, for the total of *m* trees.
- This means that the number of routing trees is reduced from (*m* × *n*) in the source-based tree approach to *m* in this approach.

11. Explain in detail about IPv6 ADDRESSING.

➢ **IPv6 ADDRESSING**

- The main reason for migration from IPv4 to IPv6 is the small size of the address space in IPv4.
- An IPv6 address is 128 bits or 16 bytes (octets) long, four times the address length in IPv4.



➢ **Representation / Notations**

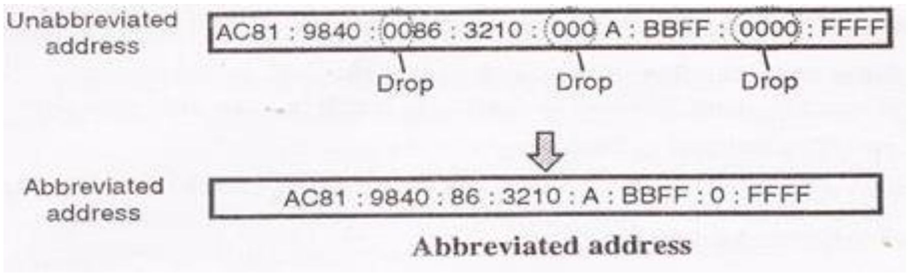
- Binary notation is used when the addresses are stored in a computer.
- The **colon hexadecimal notation** divides the address into eight sections, each made of four hexadecimal digits separated by colons.

Binary (128 bits)	1111111011110110	…	1111111100000000
Colon Hexadecimal	FEF6:BA98:7654:3210:ADEF:BBFF:2922:FF00		

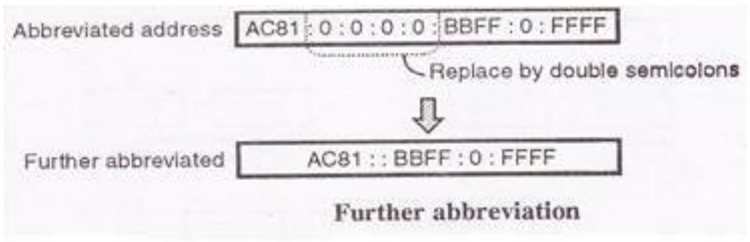
Abbreviation :

Zero Compression

- The IPv6 address, even in hexadecimal format is very long. But in this address there are many of the zero digits in it. In such a case, we can abbreviate the address. The leading zeros of a section (four digits between two colons) can be omitted.
- Note that only the leading zeros can be dropped but the trailing zeros cannot drop.


Unabbreviated address
AC81 : 9840 : 0086 : 3210 : 000A : BBFF : 0000 : FFFF
Drop
Drop
Drop
Abbreviated address
AC81 : 9840 : 86 : 3210 : A : BBFF : 0 : FFFF

Abbreviated address


Abbreviated address
AC81 : 0 : 0 : 0 : 0 : BBFF : 0 : FFFF
Replace by double semicolons
Further abbreviated
AC81 : : BBFF : 0 : FFFF

Further abbreviation

Mixed Notation

- Mixed representation of an IPv6 address: colon hex and dotted decimal notation.
- This is appropriate during the transition period in which an IPv4 address is embedded in an IPv6 address (as the rightmost 32 bits).
- For example, the address (::130.24.24.18) is a legitimate address in IPv6.

CIDR Notation

- IPv6 uses hierarchical addressing. For this, IPv6 allows slash or CIDR notation.
- For example, the following shows how we can define a prefix of 60 bits using CIDR.

FDEC::BBFF:0:FFFF/60

➢ **Address Space**

- The address space of IPv6 contains 2128 addresses. This address space is 296 times the IPv4 address.

Three Address Types

- In IPv6, a destination address can belong to one of three categories:
  - unicast,
  - anycast,
  - multicast.

1. Unicast Address

- A unicast address defines a single interface (computer or router).
- The packet sent to a unicast address will be routed to the intended recipient.

2. Anycast Address

- An **anycast address** defines a group of computers that all share a single address.

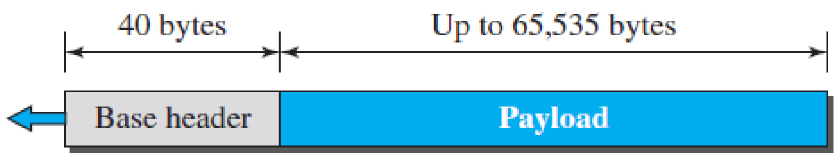
- A packet with an anycast address is delivered to only one member of the group, the most reachable one.

3. Multicast Address

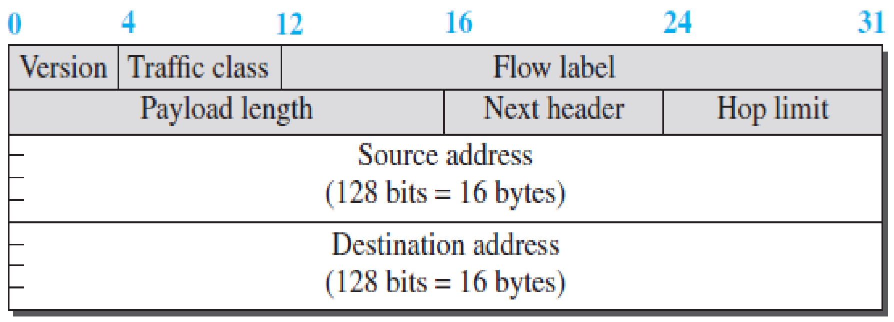
- A multicast address also defines a group of computers.
- In multicasting each member of the group receives a copy.

➢ **IPv6 Packet Format:**

- Each packet is composed of a base header followed by the payload.
- The base header occupies 40 bytes, whereas payload can be up to 65,535 bytes of information.



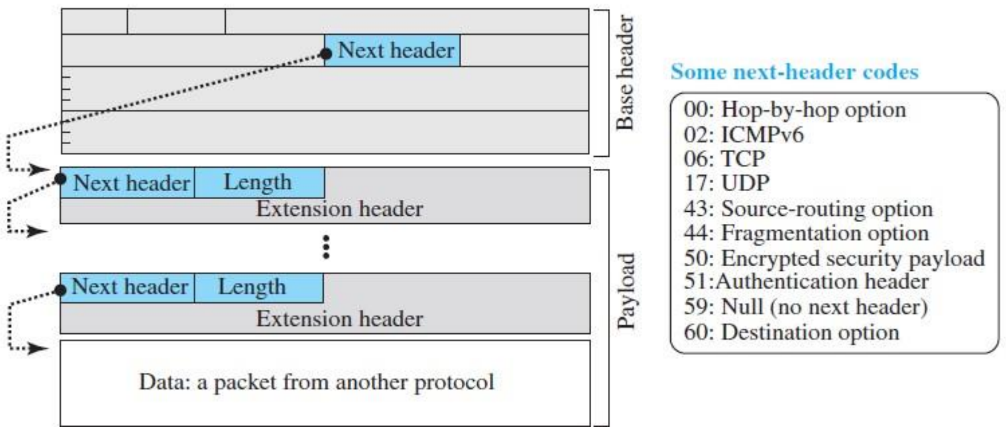
a. IPv6 packet



b. Base header

- **Version.** The 4-bit version field defines the version number of the IP. For IPv6, the value is 6.
- **Traffic class.** The 8-bit traffic class field is used to distinguish different payloads with different delivery requirements. It replaces the type-of-service field in IPv4.
- **Flow label.** The flow label is a 20-bit field that is designed to provide special handling for a particular flow of data. We will discuss this field later.
- **Payload length.** The 2-byte payload length field defines the length of the IP datagram excluding the header.
- **Hop limit.** The 8-bit hop limit field serves the same purpose as the TTL field in IPv4.
- **Source and destination addresses.** The source address field is a 16-byte (128-bit) Internet address that identifies the original source of the datagram. The destination address field is a 16-byte (128-bit) Internet address that identifies the destination of the datagram.
- **Payload.** Compared to IPv4, the payload field in IPv6 has a different format and meaning.

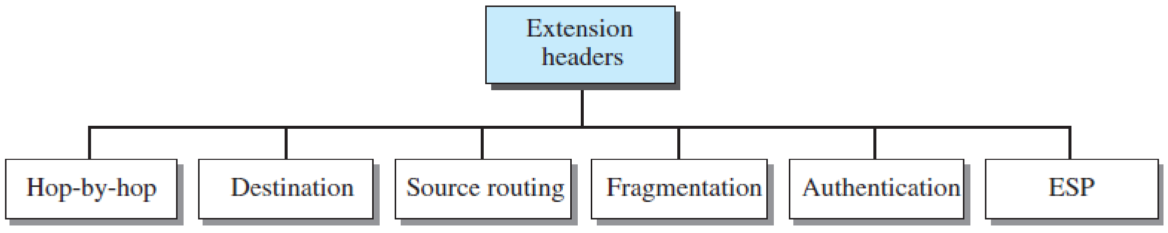
Figure 22.7 *Payload in an IPv6 datagram*



- The payload in IPv6 means a combination of zero or more extension headers (options) followed by the data from other protocols (UDP, TCP, and so on).
- Each extension header has two mandatory fields, next header and the length, followed by information related to the particular option.

➤ **Extension Header**

- An IPv6 packet is made of a base header and some extension headers.
- The length of the base header is fixed at 40 bytes.
- Extension headers are hop-by-hop option, source routing, fragmentation, authentication, encrypted security payload, and destination option.



Hop-by-Hop Option

- The hop-by-hop option is used when the source needs to pass information to all routers visited by the datagram.
- Only three hop-by-hop options have been defined:
    - Pad1, PadN, and jumbo payload.
  - Pad1. This option is 1 byte long and is designed for alignment purposes.
  - PadN. PadN is used when 2 or more bytes are needed for alignment.
  - Jumbo payload. Length of the payload in the IP datagram can be a maximum of 65,535 bytes.

Destination Option

- The destination option is used when the source needs to pass information to the destination only.

- The format of the destination option is the same as the hop-by-hop option.

Source Routing

- The source routing extension header combines the concepts of the strict source route and the loose source route options of IPv4.

Fragmentation

- The concept of fragmentation in IPv6 is the same as that in IPv4.
- In IPv6, only the original source can fragment.

Authentication

- The authentication extension header has a dual purpose: it validates the message sender and ensures the integrity of data.

Encrypted Security Payload

- The encrypted security payload (ESP) is an extension that provides confidentiality and guards against eavesdropping.

➢ **Advantages of IPv6:**

- ***Larger address space***
  - IPv6 has 128-bit address space, which is 4 times wider in bits in compared to IPv4's 32-bit address space.
- ***Better header format***
  - IPv6 uses a better header format. In its header format the options are separated from the base header.
- ***New option***
  - New options have been added in IPv6 to increase the functionality.
- ***Possibility of extension***
  - IPv6 has been designed in such a way that there is a possibility of extension of protocol if required.
- ***More security***
  - IPv6 includes security in the basic specification.
  - It includes encryption of packets (ESP: Encapsulated Security Payload) and authentication of the sender of packets (AH: Authentication Header).
- ***Support to resource allocation***
  - To implement better support for real time traffic (such as video conference), IPv6 includes flow label in the specification.
  - With flow label mechanism, routers can recognize to which end-to-end flow the packets belong.
- ***Plug and play***
  - IPv6 includes plug and play in the standard specification.
  - It therefore must be easier for novice users to connect their machines to the network, it will be done automatically.
- ***Clearer specification and optimization***

- IPv6 follows good practices of IPv4, and rejects minor flaws/obsolete items of IPv4.

➢ **Comparison of Options between IPv4 and IPv6**

- The no-operation and end-of-option options in IPv4 are replaced by Pad1 and PadN options in IPv6.
- The record route option is not implemented in IPv6 because it was not used.
- The timestamp option is not implemented because it was not used.
- The source route option is called the source route extension header in IPv6.
- The fragmentation fields in the base header section of IPv4 have moved to the fragmentation extension header in IPv6.
- The authentication extension header is new in IPv6.
- The encrypted security payload extension header is new in IPv6.

12. For the following networks, develop the datagram forwarding table for all the nodes. The links are labeled with relative costs. The tables should forward each packet via the least cost path to destination.

Solution: Initial distance stored at each node.

Information stored at node	**Distance**				
	A	**B**	**C**	**D**	**E**
A	0	∞	10	∞	2
B	∞	0	4	2	3
C	10	4	0	1	∞
D	∞	2	1	0	∞
E	2	3	∞	∞	0

Step: 1 To find Initial Routing table at node A.

Destination	**Cost**	**NextHop**
B	∞	-
C	10	C
D	∞	-
E	2	E

To find Final Routing table at node A.

Destination	**Cost**	**NextHop**
B	5	E
C	10	D
D	7	B
E	2	E

Step: 2 To find Initial Routing table at node B.

Destination	**Cost**	**NextHop**
A	∞	-
C	4	C
D	2	D
E	3	E

To find Final Routing table at node B.

Destination	**Cost**	**NextHop**
A	5	E
C	4	C
D	2	D
E	3	E

Step: 3 To find Final Routing table at node C.

Destination	**Cost**	**NextHop**
A	8	D
B	4	B
D	1	D
E	6	B

Step: 4 To find Final Routing table at node D.

Destination	**Cost**	**NextHop**
A	7	B
B	2	B
C	1	C
E	5	B

Step: 5 To find Final Routing table at node E.

Destination	**Cost**	**NextHop**
A	2	A
B	3	B
C	6	D
D	5	B

Final distance stored at each node.

CS8591 - COMPUTER NETWORKS

UNIT I - INTRODUCTION AND PHYSICAL LAYER

Networks – Network Types – Protocol Layering – TCP/IP Protocol suite – OSI Model – Physical Layer: Performance – Transmission media – Switching – Circuit-switched Networks – Packet Switching

PART A

1. List the requirements to building a network.

- ✓ Scalable Connectivity
- ✓ Cost-Effective Resource Sharing
- ✓ Support for Common Services
- ✓ Manageability

2. Write the parameters used to measure network performance (May 2016)

- ➢ Bandwidth and Latency
- ➢ Delay×Bandwidth Product
- ➢ High-Speed Networks
- ➢ Application Performance Needs

3. What are the three criteria necessary for an effective and efficient network?

The most important criteria are

- ✓ Performance
- ✓ Reliability
- ✓ Security

Performance of the network depends on number of users, type of transmission medium, and the capabilities of the connected h/w and the efficiency of the s/w. *Reliability* is measured by frequency of failure, the time it takes a link to recover from the failure and the network's robustness in a catastrophe. *Security* issues include protecting data from unauthorized access and viruses.

4. Group the OSI layers by function?

The seven layers of the OSI model belonging to three subgroups. Physical, data link and network layers are the *network support layers*; they deal with the physical aspects of moving data from one device to another. Session, presentation and application layers are the *user support layers*; they allow interoperability among unrelated software systems. The transport layer ensures *end-to-end reliable data transmission*.

5. What are the features provided by layering? (May 2013)

Two features:

- It decomposes the problem of building a network into more manageable components.
- It provides a more modular design.

6. Why are protocols needed?

In networks, communication occurs between the entities in different systems. Two entities cannot just send bit streams to each other and expect to be understood. For communication, the entities must agree on a protocol. A protocol is a set of rules that govern data communication.

7. What are the two interfaces provided by protocols?

- Service interface

- Peer interface

Service interface- defines the operations that local objects can perform on the protocol.
Peer interface- defines the form and meaning of messages exchanged between protocol peers to implement the communication service.

8. Mention the different physical media?

- Twisted pair (the wire that your phone connects to)
- Coaxial cable (the wire that your TV connects to)
- Optical fiber (the medium most commonly used for high-bandwidth, long-distance links)
- Space (the stuff that radio waves, microwaves and infra red beams propagate through)

9. Explain the two types of duplex?

- *Full duplex*-two bit streams can be simultaneously transmitted over the links at the same time, one going in each direction.
- *Half duplex*-it supports data flowing in only one direction at a time.

10. What is spread spectrum and explain the two types of spread spectrum?

Spread spectrum is to spread the signal over a wider frequency band than normal in such a way as to minimize the impact of interference from other devices.

- Frequency Hopping
- Direct sequence

11. What are the different encoding techniques?

- NRZ
- NRZI
- Manchester
- 4B/5B

12. What are the responsibilities of data link layer?

Specific responsibilities of data link layer include the following.
a) Framing b) Physical addressing c) Flow control d) Error control e) Access control.

13. Define flow control? (NOV 2011)(May 2015) (May 2016)

Flow control refers to a set of procedures used to restrict the amount of data. The sender can send before waiting for acknowledgment.

14. Mention the categories of flow control?

There are 2 methods have been developed to control flow of data across communication links.
a) Stop and wait - send one from at a time.
b) Sliding window - send several frames at a time.

15. What is a buffer?

Each receiving device has a block of memory called a buffer, reserved for storing incoming data until they are processed.

16. What is the difference between a passive and an active hub?

An active hub contains a repeater that regenerates the received bit patterns before sending them out. A passive hub provides a simple physical connection between the attached devices.

17. For n devices in a network, what is the number of cable links required for a mesh and ring topology?

- Mesh topology – n (n-1)/2
- Ring topology – n

18. What are the two types of line configuration? (NOV 2010)

- ➢ Point-to-point & Multipoint

19. What do you meant by error control? (NOV 2010)(May 2015)

Error control is used for detecting and retransmitting damaged or lost frames and to prevent duplication of frames. This is achieved through a trailer added at the end of the frame.

20. Define Error detection (NOV 2011)

Data can be corrupted during transmission. For reliable communication, errors must be detected and corrected

Types of error:

- ✓ Single bit error
- ✓ Burst error

The three error detecting techniques are:

- ➢ Parity check
- ➢ Check sum algorithm
- ➢ Cyclic Redundancy Check

21. What is the use of Two dimensional parity in error detection? (NOV 2012)

- ➢ It is based on simple parity.
- ➢ It performs calculation for each bit position across each byte in the frame.
- ➢ This adds extra parity byte for entire frame, in addition to a parity bit for each byte.

22. What are the issues (Services) in data link layer? (NOV 2012) (May 2016) (Nov 2016)

a) Services Provided to the Network Layer
b) Framing
c) Error Control
d) Flow Control

23. Define network and computer network

A **network** is any collection of independent computers that communicate with one another over a shared network medium. A **computer network** is a collection of two or more connected computers. When these computers are joined in a network, people can share files and peripherals such as modems, printers, tape backup drives, or CD-ROM drives.

24. List the components of data communication

- ✓ Message
- ✓ Sender
- ✓ Receiver
- ✓ Medium
- ✓ Protocol

25. Define bit stuffing. Give example (MAY 2011) (May 2017)

Bit stuffing is the insertion of one or more bits into a transmission unit as a way to provide signaling information to a receiver. The receiver knows how to detect and remove or disregard the stuffed bits.

e.g, Sending side - 011111**0**10

26. What are the major duties of network layer? (MAY 2012)

- **Logical addressing** - If a packet passes the n/w boundary, we need another addressing system for source and destination called logical address.
- **Routing** – The devices which connects various networks called routers are responsible for delivering packets to final destination.

27. What are the functions of application layer? (MAY 2011)

- **FTAM (file transfer, access, mgmt)** - Allows user to access files in a remote host.
- **Mail services** - Provides email forwarding and storage.
- **Directory services** - Provides database sources to access information about various sources and objects.

28. Define a layer. (Nov/Dec 2013)

The OSI (Open System Interconnection) Model breaks the various aspects of a computer network into seven distinct layers. Each successive layer envelops the layer beneath it, hiding its details from the levels above.

29. What do you mean by framing? (Nov/Dec 2013) (Nov/Dec 2014)

Frames are the small data units created by data link layer and the process of creating frames by the data link layer is known as framing

30. What is protocol? What are its key elements? (NOV/DEC 2007) (May 2016)

Set of rules that govern the data communication is protocol. The key elements are

i) Syntax ii) Semantics iii) Timing

31. Define (or) mechanism of stop and wait protocol (Nov 2016)

The idea of stop-and-wait is straightforward: After transmitting one frame, the sender waits for an acknowledgment before transmitting the next frame. If the acknowledgment does not arrive after a certain period of time, the sender times out and retransmits the original frame.

32. Define sliding window algorithm

The sender can transmit several frames before needing an acknowledgement. Frames can be sent one right after another meaning that the link can carry several frames at once and its capacity can be used efficiently. The receiver acknowledges only some of the frames, using a single ACK to confirm the receipt of multiple data frames

33. Define character stuffing

The problem with the sentinel approach is that the ETX character might appear in the data portion of the frame. BISYNC overcomes this problem by "escaping" the ETX character by preceding it with a DLE (data-link-escape) character whenever it appears in the body of a frame; the DLE character is also escaped (by preceding it with an extra DLE) in the frame body. This approach is called character stuffing.

34. List the 7 OSI layers

- Physical Layer
- Data link Layer
- Network Layer
- Transport Layer
- Session Layer
- Presentation Layer
- Application Layer

35. Define hamming distance (Nov/Dec 2014)
Hamming distance = the number of bit positions in which two code-words differ.
Eg. How to calculate ?
(Exclusive OR=XOR):

```
        10001001
        10110001
        -------------
        00111000
```

=> The number of 1's give the number of different bits

36. Write down any two differences between circuit switching and packet switching (Nov/Dec 2014) (May 2017)
Circuit switching

- <u>In circuit switching network dedicated channel has to be established before the call is made between users</u>
- The channel is reserved between the users till the connection is active

Packet switching

- In packet switching network unlike CS network, it is not required to establish the connection initially
- The connection/channel is available to use by many users.

37. Define the terms: Bandwidth & Latency (Dec 2017)
Network performance was measured in two fundamental ways: <u>*bandwidth*</u> (also called *throughput*) and <u>*latency*</u> (also called *delay*).

- The bandwidth of a network is given by the <u>number of bits that can be transmitted over the network in a certain period of time</u>
- The second performance metric, latency, corresponds to <u>how long it takes a message to travel from one end of a network to the other</u>.

38. Compare Byte oriented versus Bit-oriented protocol (Dec 2017)

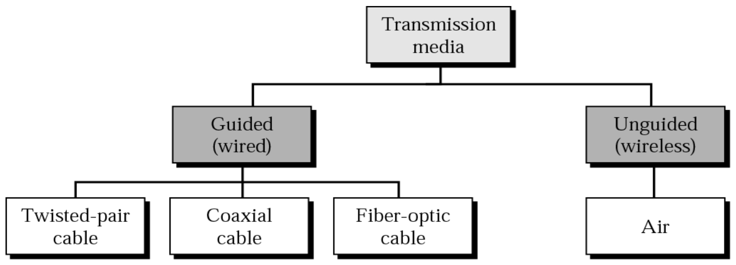
- **Bit oriented** protocol defined as it is a communication protocol it uses individual bits for control codes that bits information should be in byte.
- **Byte oriented** protocol used for framing and communication purpose, in which bytes are used for control codes

39. Define protocol layering
In data communication and networking, a protocol defines the rules that both the sender and receiver and all intermediate devices need to follow to be able to communicate effectively. When communication is simple, we may need only one simple protocol; when the communication is complex, we may need to divide the task between different layers, in which case we need a protocol at each layer, or **protocol layering.**

40. List the types of transmission media

Communication can be made by 2 ways

1. Guided (Wired)
2. Unguided (Wireless



41. Define switching & list its types

SWITCHING

- To make communication among multiple devices efficiently, a process used is called switching.
- A switched network consists of a series of interlinked nodes called switches.

Type of switching

- Circuit Switching
- Packet Switching
- Message Switching

42. Define VCI

Virtual-Circuit Identifier

The identifier that is actually used for data transfer is called the ***virtual-circuit identifier*** **(VCI)** or the ***label*****.** A VCI, unlike a global address, is a small number that has only switch scope; it is used by a frame between two switches

43. What is the use of routing table?

The destination addresses and the corresponding forwarding output ports are recorded in the tables. The routing tables are dynamic and are updated periodically.

PART B

1. Discuss the applications, advantages and disadvantages of networks

Applications of Computer Network:-

1. Business Applications
   a. Database resource
   b. Communication Medium
   c. Electronic commerce
2. Home Applications
   a. Internet Access
   b. Personal Communication
   c. Entertainment
   d. Electronic Commerce
3. Mobile Computers
   a. Wireless networks

Advantages of Network

- **Speed**. Sharing and transferring files within Networks are very rapid. Thus saving time, while maintaining the integrity of the file.
- **Cost**. Individually licensed copies of many popular software programs can be costly. Networkable versions are available at considerable savings. Shared programs, on a network allows for easier upgrading of the program on one single file server, instead of upgrading individual workstations.
- **Security**. Sensitive files and programs on a network are passwords protected or designated as "copy inhibit," so that you do not have to worry about illegal copying of programs.
- **Centralized Software Management**. Software can be loaded on one computer (the file server) eliminating that need to spend time and energy installing updates and tracking files on independent computers throughout the building.
- **Resource Sharing**. Resources such as, printers, fax machines and modems can be shared.
- **Electronic Mail**. E-mail aids in personal and professional communication.
- **Flexible Access**. Access their files from computers throughout the firm.
- **Workgroup Computing**. Workgroup software (such as Microsoft BackOffice) allows many users to work on a document or project concurrently.

Disadvantages of Network

- Server faults stop applications being available
- Network faults can cause loss of data.
- Network fault could lead to loss of resources
- User work dependent upon network
- Could become inefficient
- Could degrade in performance
- Resources could be located too far from users

2. Explain in detail about Networks & Discuss the types and connections of networks

Network :

A **network** is any collection of independent computers that communicate with one another over a shared network medium. A **computer network** is a collection of two or more connected computers. When these computers are joined in a network, people can share files and peripherals such as modems, printers, tape backup drives, or CD-ROM drives.

Network Criteria:

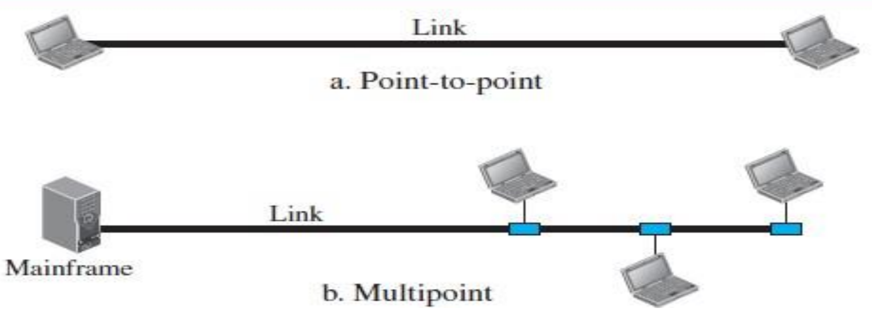
The most important criteria are
- ✓ Performance
- ✓ Reliability
- ✓ Security

Performance of the network depends on number of users, type of transmission medium, and the capabilities of the connected h/w and the efficiency of the s/w. ***Reliability*** is measured by frequency of failure, the time it takes a link to recover from the failure and the network's robustness in a catastrophe. ***Security*** issues include protecting data from unauthorized access and viruses

TYPE OF CONNECTION:
There are two types are,
1. Point to point
2. Multi point

1. Point To Point:
It provides a dedicated link between two devices of the channel. The entire capacity of the channel is reserved for transmission between those two devices.

2. Multipoint:
More than two devices can share a link by using this type of connection. It also called as multidrop. The capacity channel is shared either temporary or spatially. It simultaneously use, it is spatially shared. If it takes turns, it is time shared line configuration

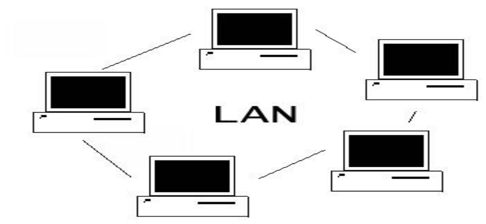


Types of Network

Computer network design is dividing into three basic types such as
- ➢ LAN (local area network),
- ➢ MAN (Metropolitan area network)

- WAN (wide area network)

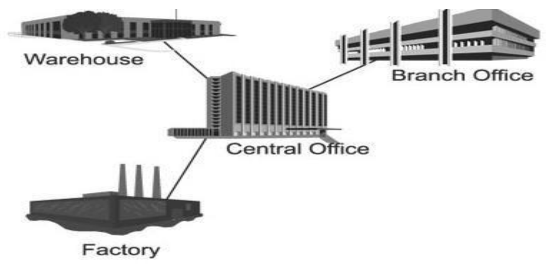
LAN (Local area networks)
Generally called **LANs**, are privately-owned **networks within a single building or campus of up to a few kilometers in size.** They are widely used to connect personal computers and workstations in company offices and factories to share resources (e.g., printers) and exchange information.



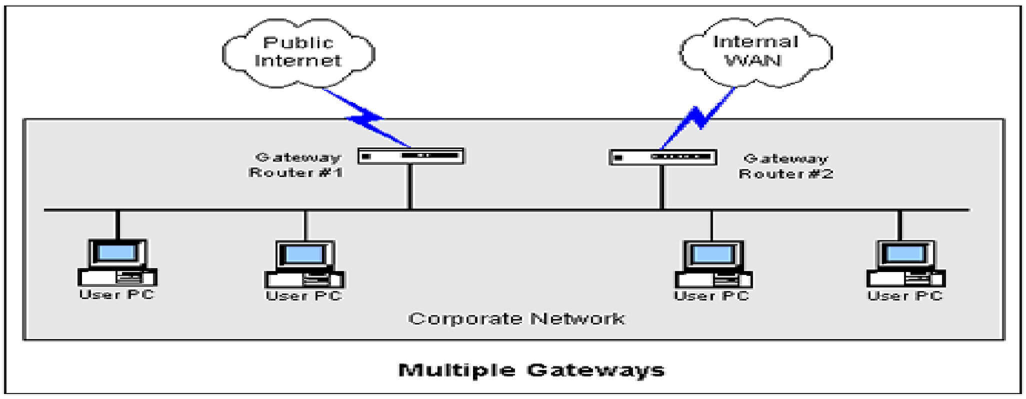
LAN configuration consists of:
- A file server
- A workstation
- Cables

MAN (Metropolitan area network)

A metropolitan area network (MAN) is a large **computer network that usually spans a city or a large campus.** A MAN usually interconnects a number of local area networks (LANs) using a high-capacity backbone technology, such as fiber-optical links, and provides up-link services to wide area networks and the Internet



WAN (Wide Area Network) A **WAN** spans a **large geographic area, such as a state, province or country.** WANs often connect multiple smaller networks, such as local area networks (LANs) or metro area networks (MANs). The world's most popular WAN is the Internet.



3. Discuss about topology and its types

Network Topologies

Topology refers to the way a network is laid out either physically or logically. Two or more devices connect to a link; two or more links form a topology. It is the geographical representation of the relationship of all the links and linking devices to each other.

1. Mesh
2. Star
3. Tree
4. Bus
5. Ring
6. Hybrid

1. Mesh Topology:

Here every device has a dedicated point to point link to every other device. A fully connected mesh can have n(n-1)/2 physical channels to link n devices. It must have n-1 IO ports.

Advantages:

1. They use dedicated links so each link can only carry its own data load. So traffic problem can be avoided.
2. It is robust. If any one link get damaged it cannot affect others
3. It gives privacy and security
4. Fault identification and fault isolation are easy.

Disadvantages:

1. The amount of cabling and the number IO ports required are very large. Since every device is connected to each other devices through dedicated links.
2. The sheer bulk of wiring is larger then the available space
3. Hardware required to connect each device is highly expensive.

Example:

A mesh network has 8 devices. Calculate total number of cable links and IO ports needed.

Solution:

Number of devices = 8

Number of links = n (n-1)/2

= 8(8-1)/2

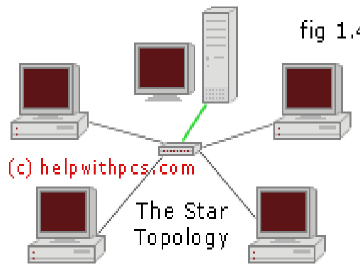
= 28

Number of port/device = n-1

= 8-1 = 7

2. STAR TOPOLOGY:

Here each device has a dedicated link to the central 'hub'. There is no direct traffic between devices. The transmission are occurred only through the central controller namely hub.

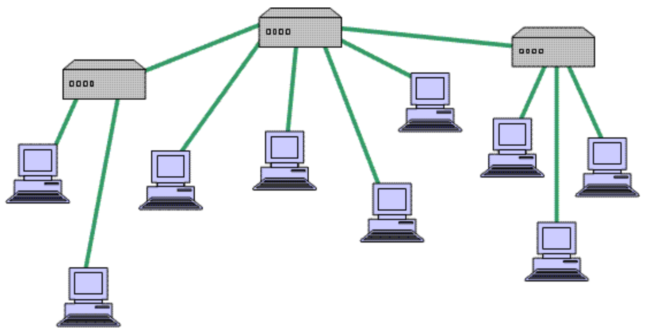


Advantages:
1. Less expensive then mesh since each device is connected only to the hub.
2. Installation and configuration are easy.
3. Less cabling is need then mesh.
4. Robustness.
5. Easy to fault identification & isolation.

Disadvantages:
1. Even it requires less cabling then mesh when compared with other topologies it still large.

3. TREE TOPOLOGY:

It is a variation of star. Instead of all devices connected to a central hub here most of the devices are connected to a secondary hub that in turn connected with central hub. The central hub is an active hub. An active hub contains a repeater, which regenerate the received bit pattern before sending.

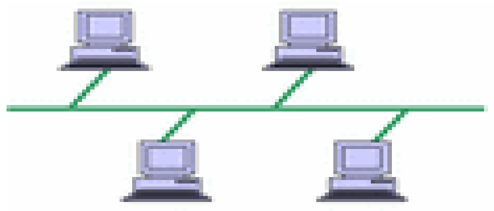


The secondary hub may be active or passive. A passive hub means it just precedes a physical connection only.

Advantages:
1. Can connect more than star.
2. The distance can be increased.
3. Can isolate and prioritize communication between different computers.

4. BUS TOPOLOGY:

A bus topology is multipoint. Here one long cable is act as a backbone to link all the devices are connected to the backbone by drop lines and taps. A drop line is the connection between the devices and the cable. A tap is the splice into the main cable or puncture the sheathing.



Advantages:

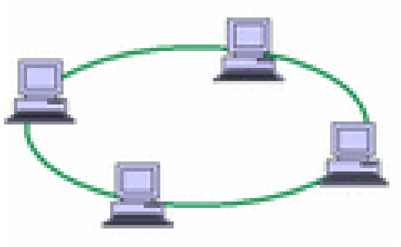
1. Ease of installation.
2. Less cabling.

Disadvantages:

1. Difficult reconfiguration and fault isolation.
2. Difficult to add new devices.
3. Signal reflection at top can degradation in quality
4. If any fault in backbone can stops all transmission

5. Ring topology

Each node is connected to exactly two other nodes, forming a ring. Can be visualized as a circular configuration. Requires at least three nodes



Advantages:

1. Easy to install.
2. Easy to reconfigure.
3. Fault identification is easy.

Disadvantages:

1. Unidirectional traffic.
2. Break in a single ring can break entire network.

6. Hybrid topology

A combination of any two or more network topologies.

4. Explain the list of requirements (challenges faced) to building a computer network (May 2017) (Nov 2017)

- ✓ Scalable Connectivity
- ✓ Cost-Effective Resource Sharing
- ✓ Support for Common Services
- ✓ Manageability

1. Scalable Connectivity

Networks (of which the Internet is the prime example) are designed to grow in a way that allows them the potential to connect all the computers in the world. A system that is designed to support growth to an arbitrarily large size is said to *scale*.

Links, Nodes, and Clouds

A network can consist of two or more computers directly connected by some physical medium, such as a coaxial cable or an optical fiber. We call such a physical medium **a *link***, and we often refer to the computers it connects as ***nodes.***

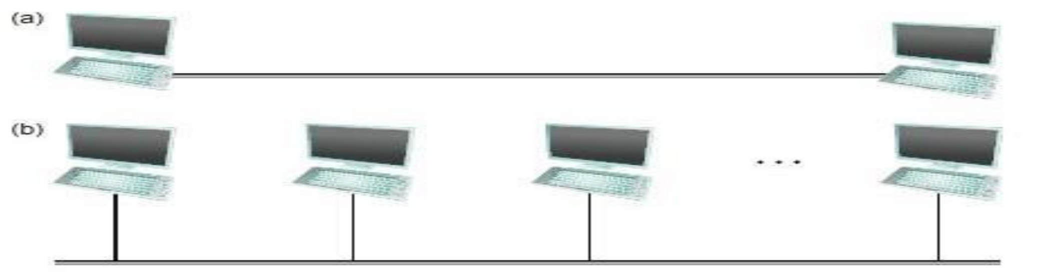


FIGURE 1.2 Direct links: (a) point-to-point; (b) multiple-access.

The cloud distinguishes between the nodes on the inside that *implement* the network (they are commonly called *switches*, and their primary function is to store and forward packets) and the nodes on the outside of the cloud that *use* the network (they are commonly called *hosts*, and they support users and run application programs).

2. Cost-Effective Resource Sharing

Given a collection of nodes indirectly connected by a nesting of networks, it is possible for any pair of hosts to send messages to each other across a sequence of links and nodes. Of course, we want to do more than support just one pair of communicating hosts—we want to provide all pairs of hosts with the ability to exchange messages.

Multiplexing can be explained by analogy to a timesharing computer system, where a single physical processor is shared (multiplexed) among multiple jobs, each of which believes it has its own private processor. Similarly, data being sent by multiple users can be multiplexed over the physical links that make up a network. There are several different methods for multiplexing multiple flows onto one physical link. One common method is *synchronous time- division multiplexing* (STDM).

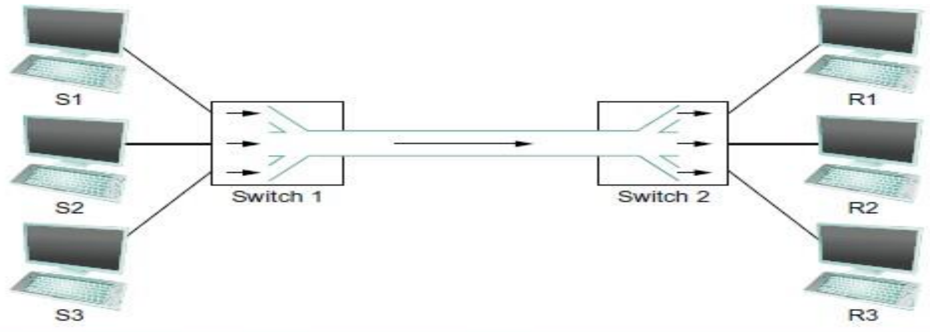


FIGURE 1.5 Multiplexing multiple logical flows over a single physical link.

3. Support for Common Services
The next requirement of a computer network is that the application programs running on the hosts connected to the network must be able to communicate in a meaningful way. From the application developer's perspective, the network needs to make his or her life easier we use a cloud to abstractly represent connectivity among a set of computers, we now think of a channel as connecting one process to another.

Diagram shows a pair of application-level processes communicating over a logical channel that is, in turn, implemented on top of a cloud that connects a set of hosts. We can think of the channel as being like a pipe connecting two applications, so that a sending application can put data in one end and expect that data to be delivered by the network to the application at the other end of the pipe

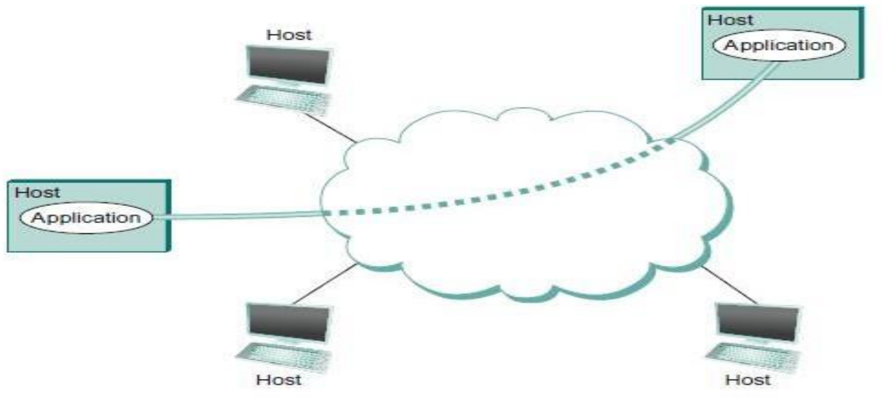


FIGURE 1.7 Processes communicating over an abstract channel.

4. Manageability

Managing a network includes making changes as the network grows to carry more traffic or reach more users, and troubleshooting the network when things go wrong or performance isn't as desired. This requirement is partly related to the issue of scalability discussed above—as the Internet has scaled up to support billions of users and at least hundreds of millions of hosts, the challenges of keeping the whole thing running correctly and correctly configuring new devices as they are added have become increasingly problematic.

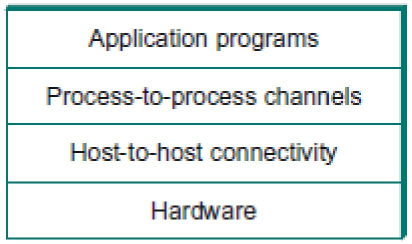
5. Discuss protocol layering in detail

Layering and Protocols

When the system gets complex, the system designer introduces another level of abstraction. It defines unifying model with important aspects of the system, encapsulated this model in interface objects and hide it from users

In network, abstraction leads to layering. Layering provides two nice features.

- It decomposes the problem of building a network into more manageable components. Rather than implementing a monolithic piece of software that does everything implement several layers, each of which solves one part of the problem.
- It provides more modular design. To add some new service, it is enough to modify the functionality at one layer, reusing the functions provided at all the other layers.



Example of a layered network system.

Protocols

A protocol is a set of rules that governs data communication. It defines what is communicated, how it is communicated, and when it is communicated. The key elements of a protocol are syntax, semantics and timing.

Each protocol defines two different interfaces.

- **Service interface** - to the other objects on the same computer that want to use its communication services. This service interface defines the operations that local objects can perform on the protocol.
- **Peer interface** - to its counterpart (peer) on another machine. It also defines the form and meaning of messages exchanged between protocol peers to implement the communication service.

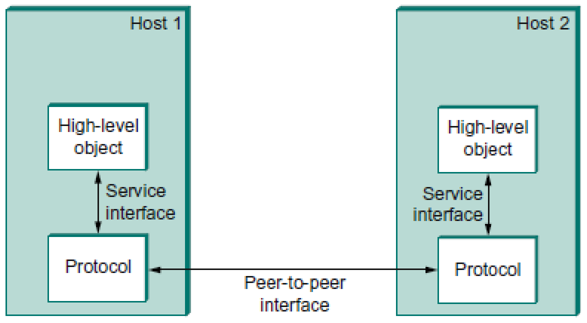


FIGURE 1.10 Service interfaces and peer interfaces.

Encapsulation

Control information must be added with the data to instruct the peer how to handle with the received message. It will be added into the header or trailer.
Header - Small data structure from few bytes to few kilobytes attached to the front of message.
Trailer – Information will be added at the end of the message
Payload or message body – Data send by the program
In this case data is encapsulated with new message created by protocol at each level.

Multiplexing and De-Multiplexing

The fundamental idea of packet switching is to multiplex multiple flows of data over a single physical link. This can be achieved by adding identifier to the header message. It is known as **demultiplexing or demux key.** It gives the address to which it has to communicate.
The messages are demultiplexed at the destination side. In some cases same demux key is used on both sides and in some cases different keys are used.

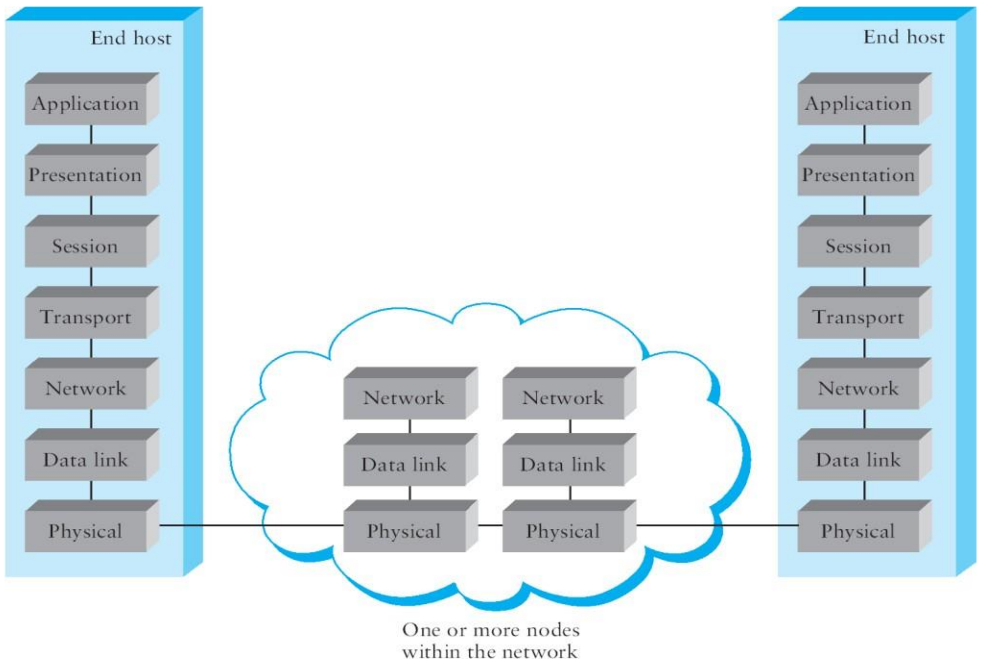
6. Explain OSI model in detail (or) Discuss ISO-OSI architecture in detail

OSI Architecture (NOV 2012) (MAY 2012)

ISO defines a common way to connect computer by the architecture called Open System Interconnection (OSI) architecture.

Network functionality is divided into seven layers.

- **Physical Layer**
- **Data link Layer**
- **Network Layer**
- **Transport Layer**
- **Session Layer**
- **Presentation Layer**
- **Application Layer**



Organization of the layers
The 7 layers can be grouped into 3 subgroups

1. **Network Support Layers**
Layers 1,2,3 - Physical, Data link and Network are the network support layers. They deal with the physical aspects of moving data from one device to another such as electrical specifications, physical addressing, transport timing and reliability.

2. **Transport Layer**
Layer4, transport layer, ensures end-to-end reliable data transmission on a single link.
3. **User Support Layers**
Layers 5,6,7 – Session, presentation and application are the user support layers. They allow interoperability among unrelated software systems

Functions of the Layers
1. **Physical Layer**
The physical layer coordinates the functions required to transmit a bit stream over a physical medium.
The physical layer is concerned with the following:
➢ **Physical characteristics of interfaces and media -** The physical layer defines the characteristics of the interface between the devices and the transmission medium.
➢ **Representation of bits -** To transmit the stream of bits, it must be encoded to signals. The physical layer defines the type of encoding.
➢ **Data Rate or Transmission rate -** The number of bits sent each second – is also defined by the physical layer.
➢ **Synchronization of bits -** The sender and receiver must be synchronized at the bit level. Their clocks must be synchronized.
➢ **Line Configuration -** In a point-to-point configuration, two devices are connected together through a dedicated link. In a multipoint configuration, a link is shared between several devices.
➢ **Physical Topology -** The physical topology defines how devices are connected to make a network. Devices can be connected using a mesh, bus, star or ring topology.

- **Transmission Mode -** The physical layer also defines the direction of transmission between two devices: simplex, half-duplex or full-duplex.

2. Data Link Layer

It is responsible for transmitting frames from one node to next node.

The other responsibilities of this layer are

- **Framing -** Divides the stream of bits received into data units called frames.
- **Physical addressing** – If frames are to be distributed to different systems on the n/w , data link layer adds a header to the frame to define the sender and receiver.
- **Flow control**- If the rate at which the data are absorbed by the receiver is less than the rate produced in the sender ,the Data link layer imposes a flow ctrl mechanism.
- **Error control**- Used for detecting and retransmitting damaged or lost frames and to prevent duplication of frames. This is achieved through a trailer added at the end of the frame.
- **Access control** -Used to determine which device has control over the link at any given time.

3. NETWORK LAYER

This layer is responsible for the delivery of packets from source to destination.

It is mainly required, when it is necessary to send information from one network to another.

The other responsibilities of this layer are

- **Logical addressing** - If a packet passes the n/w boundary, we need another addressing system for source and destination called logical address.
- **Routing** – The devices which connects various networks called routers are responsible for delivering packets to final destination.

4. TRANSPORT LAYER

- It is responsible for **Process to Process** delivery.
- It also ensures whether the message arrives in order or not.

The other responsibilities of this layer are

- **Port addressing** - The header in this must therefore include a address called port address. This layer gets the entire message to the correct process on that computer.
- **Segmentation and reassembly** - The message is divided into segments and each segment is assigned a sequence number. These numbers are arranged correctly on the arrival side by this layer.
- **Connection control** - This can either be **connectionless or connection-oriented.** The connectionless treats each segment as a individual packet and delivers to the destination. The connection-oriented makes connection on the destination side before the delivery. After the delivery the termination will be terminated.
- **Flow and error control** - Similar to data link layer, but process to process take place.

5. SESSION LAYER

This layer establishes, manages and terminates connections between applications.

The other responsibilities of this layer are

- **Dialog control** - This session allows two systems to enter into a dialog either in half duplex or full duplex.
- **Synchronization**-This allows to add checkpoints into a stream of data.

6. PRESENTATION LAYER

It is concerned with the syntax and semantics of information exchanged between two systems.

The other responsibilities of this layer are

- **Translation** – Different computers use different encoding system, this layer is responsible for interoperability between these different encoding methods. It will change the message into some common format.
- **Encryption and decryption**-It means that sender transforms the original information to another form and sends the resulting message over the n/w. and vice versa.
- **Compression and expansion**-Compression reduces the number of bits contained in the information particularly in text, audio and video.

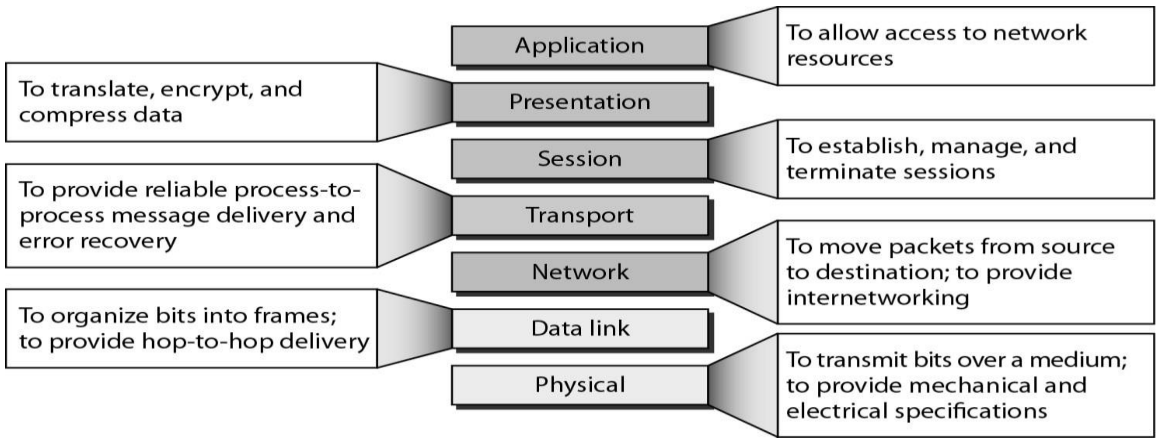
7. APPLICATION LAYER

This layer enables the user to access the n/w. This allows the user to log on to remote user.

The other responsibilities of this layer are

- **FTAM (file transfer, access, mgmt)** - Allows user to access files in a remote host.
- **Mail services** - Provides email forwarding and storage.
- **Directory services** - Provides database sources to access information about various sources and objects.

Summary of layers



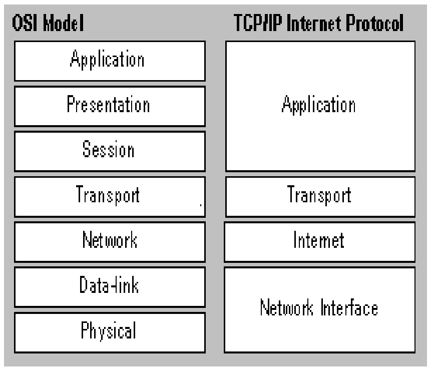
Application
To allow access to network resources
To translate, encrypt, and compress data
Presentation
Session
To establish, manage, and terminate sessions
To provide reliable process-to-process message delivery and error recovery
Transport
Network
To move packets from source to destination; to provide internetworking
To organize bits into frames; to provide hop-to-hop delivery
Data link
Physical
To transmit bits over a medium; to provide mechanical and electrical specifications

7. Explain TCP/IP protocol suite (Internet architecture) in detail (May 2015) (May 2017)

TCP/IP ARCHITECTURE

TCP/IP model is an implementation of OSI reference model. It has four layers. They are

- Network Interface Layer
- Internet Layer
- Transport (also known as Host-to-Host or Transmission) Layer
- Application Layer (known earlier as the Process Layer)



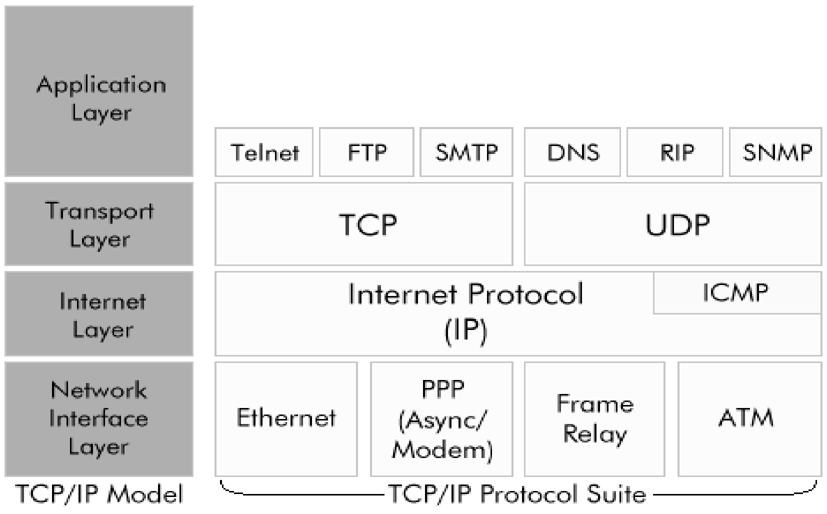


Figure 2.6 *Logical connections between layers of the TCP/IP protocol suite*

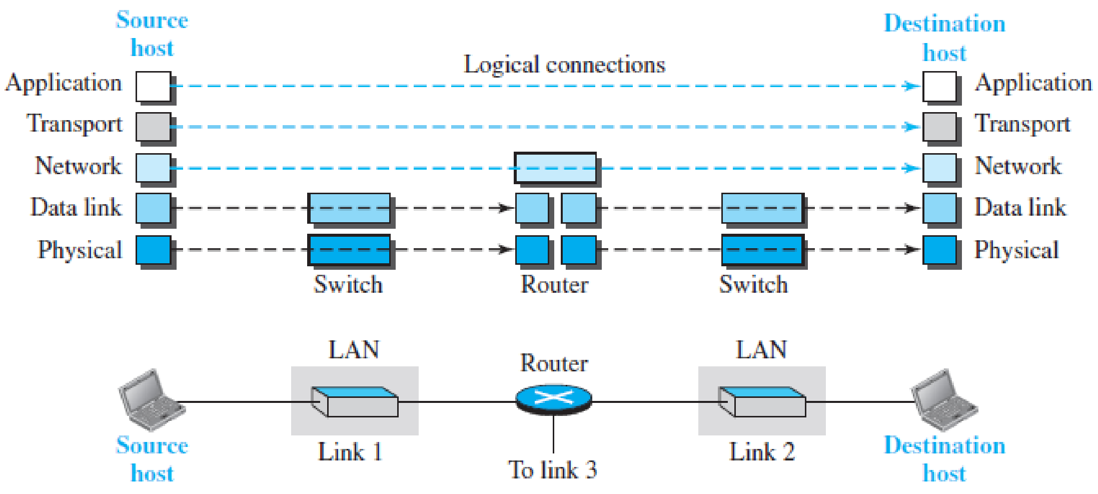
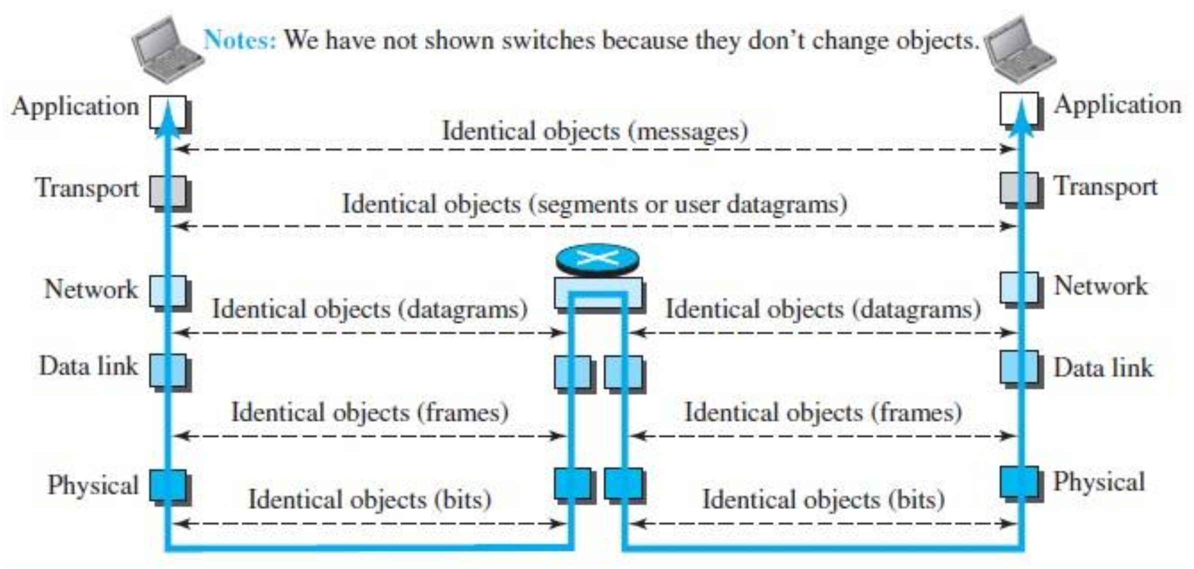


Figure 2.7 *Identical objects in the TCP/IP protocol suite*


Notes: We have not shown switches because they don't change objects.
Application
Transport
Network
Data link
Physical
Identical objects (messages)
Identical objects (segments or user datagrams)
Identical objects (datagrams)
Identical objects (datagrams)
Identical objects (frames)
Identical objects (frames)
Identical objects (bits)
Identical objects (bits)
Application
Transport
Network
Data link
Physical

1) Network interface layer (or) The Host to Network Layer:

Below the internet layer is great void. The TCP/IP reference model does not really say such about what happen here, except to point out that the host has connect to the network using some protocol so it can transmit IP packets over it. This protocol is not specified and varies from host to host and network to network.

2) Internet layer:

Packet switching network depends upon a connectionless internetwork layer. This layer is known as internet layer, is the linchpin that holds the whole design together. Its job is to allow hosts to insert packets into any network and have them to deliver independently to the destination. They may appear in a different order than they were sent in each case it is job of higher layers to rearrange them in order to deliver them to proper destination.

The internet layer specifies an official packet format and protocol known as internet protocol. The job of internet layer is to transport IP packets to appropriate destination. Packet routing is very essential task in order to avoid congestion. For these reason it is say that TCP/IP internet layer perform same function as that of OSI network layer.

3) Transport layer:

In the TCP/IP model, the layer above the internet layer is known as transport layer. It is developed to permit entities on the source and destination hosts to carry on a conversation. It specifies 2 end-to-end protocols

i) TCP (Transmission Control Protocol)
ii) UDP (User Datagram Protocol)

TCP

It is a reliable connection-oriented protocol that permits a byte stream originating on one machine to be transported without error on any machine in the internet. It divides the incoming byte stream into discrete message and passes each one onto the internet layer. At the destination, the receiving TCP process collects the received message into the output stream. TCP deals with flow control to make sure a fast sender cannot swamp a slow receiver with more message than it can handle.

UDP

It is an unreliable, connectionless protocol for applications that do not want TCP's sequencing on flow control and wish to offer their own. It is also used for client-server type request-reply queries and applications in which prompt delivery is more important than accurate delivery such as transmitting speech or video.

4) Application Layer:

In TCP/IP model, session or presentation layer are not present. Application layer is present on the top of the Transport layer. It includes all the higher-level protocols which are virtual terminal (TELNET), file transfer (FTP) and electronic mail (SMTP).

The virtual terminal protocol permits a user on one machine to log into a distant machine and work there. The file transfer protocol offers a way to move data efficiently from one machine to another. Electronic mail was used for file transfer purpose but later a specialized protocol was developed for it.

The Application Layer defines following protocols

i) File Transfer Protocol (FTP)

It was designed to permit reliable transfer of files over different platforms. At the transport layer to ensure reliability, FTP uses TCP.

FTP offers simple commands and makes the differences in storage methods across networks transparent to the user. The FTP client is able to interact with any FTP server; therefore the FTP server must also be able to interact with any FTP client.

FTP does not offer a user interface, but it does offer an application program interface for file transfer. The client part of the protocol is called as FTP and the server part of the protocol is known as FTPd. The suffix "d" means Daemon this is a legacy from Unix computing where a daemon is a piece of software running on a server that offers a service.

ii) Hyper Text Transfer Protocol

HTTP permits applications such as browsers to upload and download web pages. It makes use of TCP at the transport layer again to check reliability.

HTTP is a connectionless protocol that sends a request, receives a response and then disconnects the connection.

HTTP delivers HTML documents plus all of the other components supported within HTML such as JavaScript, Visual script and applets.

iii) Simple Mail Transfer Protocol

By using TCP, SMTP sends email to other computers that support the TCP/IP protocol suite. SMTP provides extension to the local mail services that existed in the early years of LANs. It supervises the email sending from the local mail host to a remote mail host. It is not reliable for accepting mail from local users or distributing received mail to recipients this is the responsibility of the local mail system.

SMTP makes use of TCP to establish a connection to the remote mail host, the mail is sent, any waiting mail is requested and then the connection is disconnected. It can also return a forwarding address if the intended recipient no longer receives email at that destination. To enable mail to be delivered across differing systems, a mail gateway is used.

iv) Simple Network Management Protocol

For the transport of network management information, SNMP is used as standardized protocol. Managed network devices can be cross examined by a computer running to return details about their status and level of activity. Observing software can also trigger alarms if certain performance criteria drop below acceptable restrictions. At the transport layer SNMP protocol uses UDP.

The use of UDP results in decreasing network traffic overheads.

8. Discuss in detail about the network performance measures (Nov 2016)

Like any computer system, however, computer networks are also expected to perform well. This is because the effectiveness of computations distributed over the network often depends directly on the efficiency with which the network delivers the computation's data.

While the old programming adage "first get it right and then make it fast" is valid in many settings, in networking it is usually necessary to "design for performance." It is therefore important to understand the various factors that impact network performance.

- ✓ **Bandwidth**
- ✓ **Throughput**
- ✓ **Latency (Delay)**
- ✓ **Jitter**

Bandwidth

The bandwidth of a network is given by the number of bits that can be transmitted over the network in a certain period of time.

- ***Bandwidth in Hertz***

We have discussed this concept. Bandwidth in hertz is the range of frequencies contained in a composite signal or the range of frequencies a channel can pass. For example, we can say the bandwidth of a subscriber telephone line is 4 kHz.

- ***Bandwidth in Bits per Seconds***

The term *bandwidth* can also refer to the number of bits per second that a channel, a link, or even a network can transmit. For example, one can say the bandwidth of a Fast Ethernet network (or the links in this network) is a maximum of 100 Mbps. This means that this network can send 100 Mbps.

Throughput

The **throughput** is a measure of how fast we can actually send data through a network. Although, at first glance, bandwidth in bits per second and throughput seem the same, they are different. A link may have a bandwidth of *B* bps, but we can only send *T* bps through this link with *T* always less than *B*. In other words, the bandwidth is a potential measurement of a link; the throughput is an actual measurement of how fast we can send data.

Latency or delay

The **latency** or delay defines how long it takes for an entire message to completely arrive at the destination from the time the first bit is sent out from the source. We can say that latency is made of four components: propagation time, transmission time, queuing time and processing delay.

Latency = **propagation time** + **transmission time** + **queuing time** + **processing delay**

- ***Propagation Time***

Propagation time measures the time required for a bit to travel from the source to the destination. The propagation time is calculated by dividing the distance by the propagation speed.

Propagation time = **Distance / (Propagation Speed)**

- ***Transmission Time***

In data communications we don’t send just 1 bit, we send a message. The first bit may take a time equal to the propagation time to reach its destination; the last bit also may take the same amount of time. However, there is a time between the first bit leaving the sender and the last bit arriving at the receiver. The first bit leaves earlier and arrives earlier; the last bit leaves later and arrives later. The **transmission time** of a message depends on the size of the message and the bandwidth of the channel.

Transmission time = **(Message size) / Bandwidth**

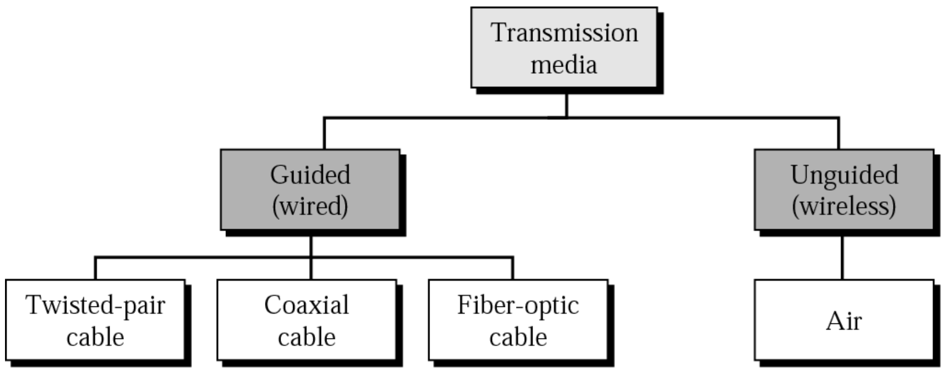
<u>Jitter</u>

Another performance issue that is related to delay is **jitter.** We can roughly say that jitter is a problem if different packets of data encounter different delays and the application using the data at the receiver site is time-sensitive (audio and video data, for example). If the delay for the first packet is 20 ms, for the second is 45 ms, and for the third is 40 ms, then the real-time application that uses the packets endures jitter.

9. Discuss physical links (or) transmission media (or) how communication made by network?

Communication can be made by 2 ways

3. Guided (Wired)
4. Unguided (Wireless)



Guided Media

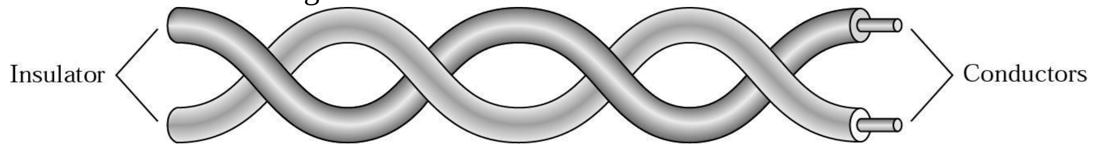
Guided media conduct signals from one device to another include Twisted-pair cable, Coaxial Cable and Fiber-optic cable. A signal traveling along any of these media is directed and contained by the physical limits of the medium.

Twisted-pair and coaxial cable use metallic (copper) conductors that accept and transport signals in the form of electric current. Optical fiber is a glass cable that accepts and transports signals in the form of light.

Twisted Pair Cable

A twisted pair consists of two conductors (normally copper) each with its own plastic insulation, twisted together.

- One of the wires is used to carry signals to the receiver
- Other is used as ground reference



Interference and cross talk may affect both the wires and create unwanted signals, if the two wires are parallel.

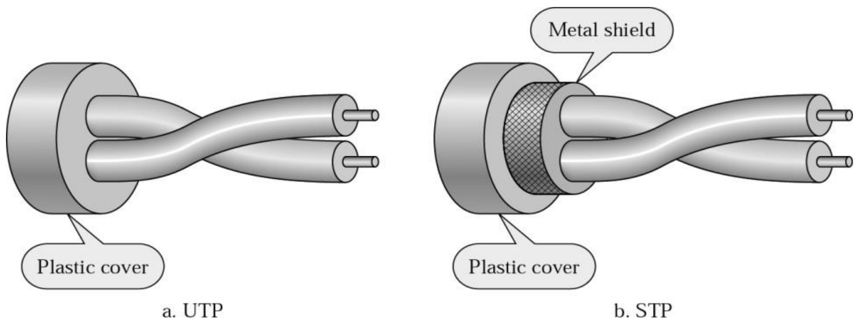
By twisting the pair, a balance is maintained. Suppose in one twist one wire is closer to noise and the other is farther in the next twist the reverse is true. Twisting makes it probable that both wires are equally affected by external influences.

Twisted Pair Cable comes into two forms:

- **Unshielded**
- **Shielded**

Unshielded versus shielded Twisted-Pair Cable

- Shielded Twisted-Pair (STP) Cable has a metal foil or braided-mesh covering that encases each pair of insulated conductors.
- Metal casing improves that quality of cable by preventing the penetration of noise or cross talk.
- It is more expensive. The following figure shows the difference between UTP and STP



Applications

- Twisted Pair cables are used in telephone lines to provide voice and data channels.
- Local area networks also use twisted pair cables.

Connectors

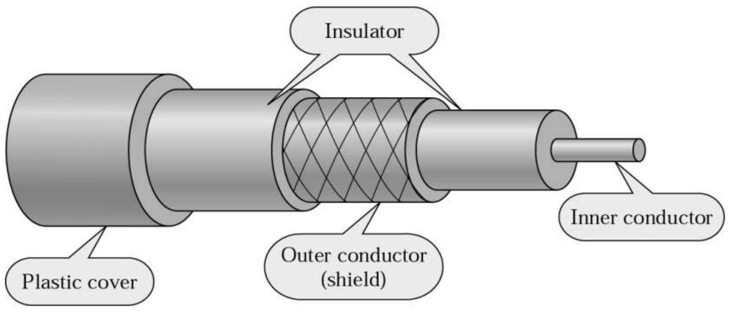
The most common UTP connector is RJ45.

Coaxial Cable

Coaxial cable (coax) carries signals of higher frequency ranges than twisted pair cable.

Instead of having two wires, coax has a central core conductor of solid or stranded wire (usually copper) enclosed in an insulating sheath, and with outer conductor of metal foil.

The outer metallic wrapping serves both as a shield against noise and as the second conductor and the whole cable is protected by a plastic cover.



]

Categories of coaxial cables

Category	Impedance	Use
RG-59	75	Cable TV
RG-58	50	Thin Ethernet
RG-11	50	Thick Ethernet

Applications

- ➢ It is used in analog and digital telephone networks
- ➢ It is also used in Cable TV networks
- ➢ It is used in Ethernet LAN

Connectors

- ➢ BNC connector – to connect the end of the cable to a device
- ➢ BNC T - to branch out network connection to computer
- ➢ BNC terminator - at the end of the cable to prevent the reflection of the signal.

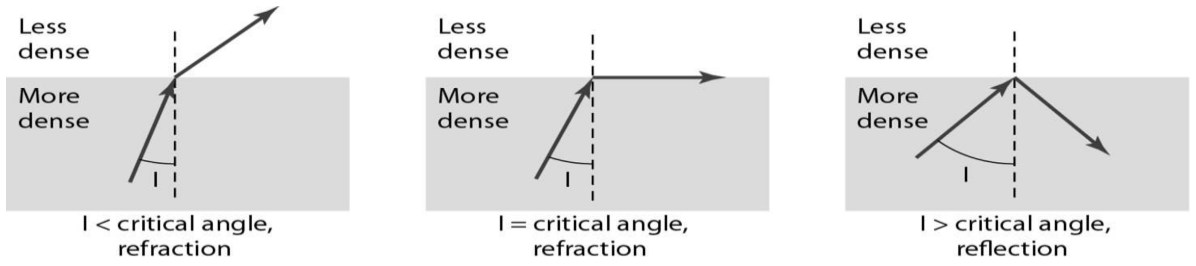
Fiber Optic Cable

A fiber-optic cable is made of glass or plastic and transmits signals in the form of light.

Properties of light

- ➢ Light travels in a straight line as long as it moves through a single uniform substance. If traveling through one substance suddenly enters another, ray changes its direction.

Bending of light ray

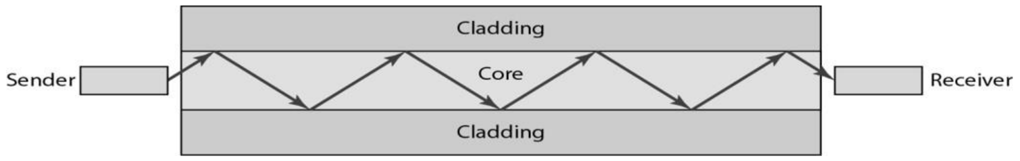

Less dense
More dense
I
I < critical angle, refraction
Less dense
More dense
I
I = critical angle, refraction
Less dense
More dense
I
I > critical angle, reflection

If the angle of incidence(the angle the ray makes with the line perpendicular to the interface between the two medium) is less than the critical angle the ray refracts and move closer to the surface.

If the angle of incidence is equal to the critical angle, the light bends along the interface.

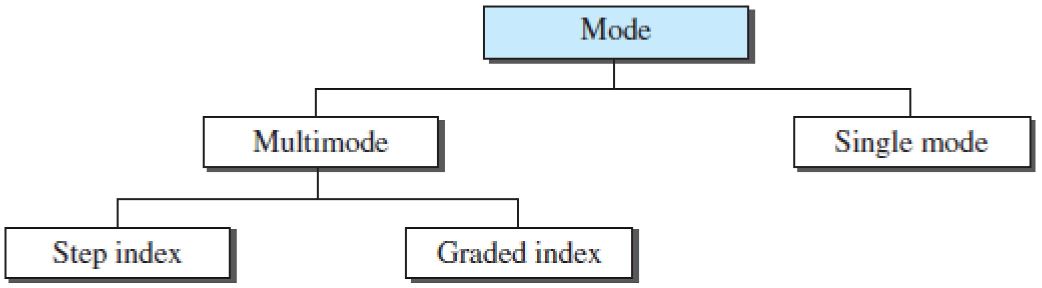
If the angle of incidence is greater than the critical angle, the ray reflects and travels again in the denser substance. Critical angle differs from one medium to another medium.

Optical fiber use reflection to guide light through a channel.



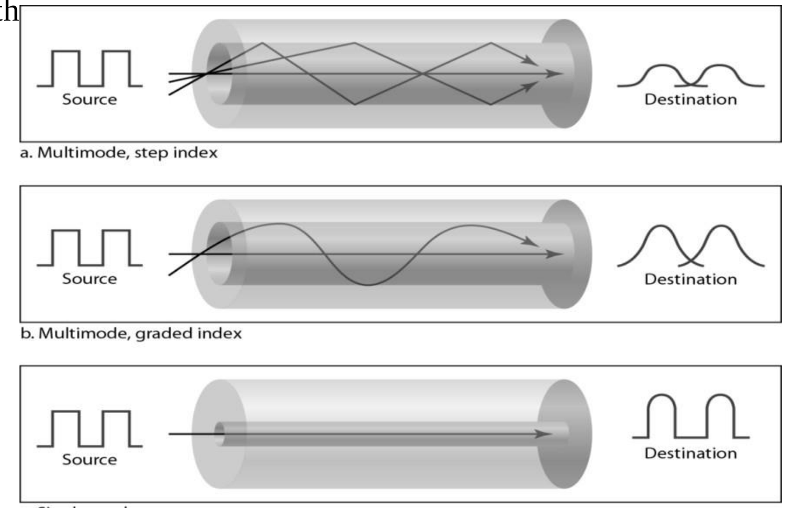
A Glass or plastic core is surrounded by a cladding of less dense glass or plastic.

Propagation Modes



<u>Multimode</u>

In the multiple mode, multiple light beams from a source move through the core in different path



a. Multimode, step index

b. Multimode, graded index

c. Single mode

- **Multimode-Step-Index fiber:** The density of core remains constant from the centre to the edge.
  A ray of light moves through this constant density in a straight line until it reaches the interface of the core and the cladding. At the interface there is an abrupt change to a lower density that changes the angle of the beam's motion.
- **Multimode- Graded -Index fiber:** The density is varying. Density is highest at the centre of the core and decreases gradually to its lowest at the edge.

<u>Single Mode</u>

Single mode uses step-index fiber and a highly focused source of light that limits beams to a small range of angles, all close to the horizontal.

The single mode fiber itself is manufactured with a much smaller diameter than that of multimedia fiber.

Connectors

- **Subscriber channel** (SC) **connector** is used for cable TV.
- **Straight-tip** (ST) **connector** is used for connecting cable to networking devices.

Advantages of Optical Fiber

- Noise resistance

- Less signal attenuation
- Light weight

Disadvantages

- Cost
- Installation and maintenance
- Unidirectional
- Fragility (easily broken)

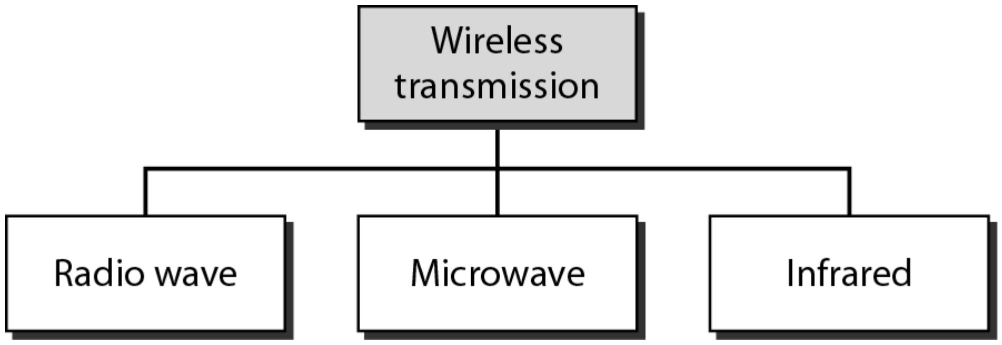
Unguided media

Unguided media transport electromagnetic waves without using a physical conductor. This type of communication is often referred to as wireless communication.

Signals are normally broadcast through air and thus available to anyone who has device capable of receiving them.

Unguided signals can travel from the source to destination in several ways:

- **Ground propagation** – waves travel through lowest portion on atmosphere.
- **Sky propagation** – High frequency waves radiate upward into ionosphere and reflected back to earth.
- **Line-of-sight propagation** – Very high frequency signals travel in a straight line



<u>Radio Waves</u>

Electromagnetic waves ranging in frequencies between 3 kHz and 1 GHz are normally called radio waves.

Properties

- Radio waves are omnidirectional. When an antenna transmits radio waves, they are propagated in all directions. This means that the sending and receiving antennas do not have to be aligned.
- A sending antenna sends waves that can be received by any receiving antenna.
- Radio waves, particularly those of low and medium frequencies, can penetrate walls.

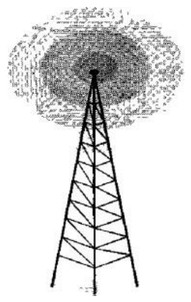


Fig:Omnidirectional antenna

Disadvantages

- The omnidirectional property has a disadvantage, that the radio waves transmitted by one antenna are susceptible to interference by another antenna that may send signals using the same frequency or band.

- As Radio waves can penetrate through walls, we cannot isolate a communication to just inside or outside a building.

Applications

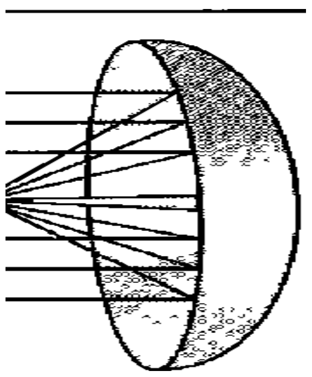
Radio waves are used for multicast communications, such as radio and television, and paging systems.

Microwaves

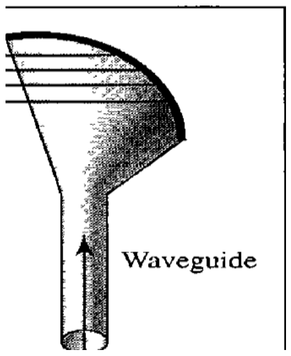
Electromagnetic waves having frequencies between 1 and 300 GHz are called microwaves.

Properties

- Microwaves are unidirectional.
- Sending and receiving antennas need to be aligned
- Microwave propagation is line-of-sight
- Very high-frequency microwaves cannot penetrate walls



a) ParabolicDish antenna



b)Horn antenna

- Parabolic Dish antenna focus all incoming waves into single point
- Outgoing transmissions are broadcast through a horn aimed at the dish.

Disadvantage

- If receivers are inside buildings, they cannot receive these waves

Applications

- Microwaves are used for unicast communication such as cellular telephones, satellite networks, and wireless LANs.

Infrared

- Electromagnetic waves with frequencies from 300 GHz to 400 THz are called infrared rays
- Infrared waves, having high frequencies, cannot penetrate walls.

Applications

- Infrared signals can be used for short-range communication in a closed area using line-of-sight propagation.

10. Discuss in detail the concepts of Packet Switched Networks (Packet Switching)

Introduction

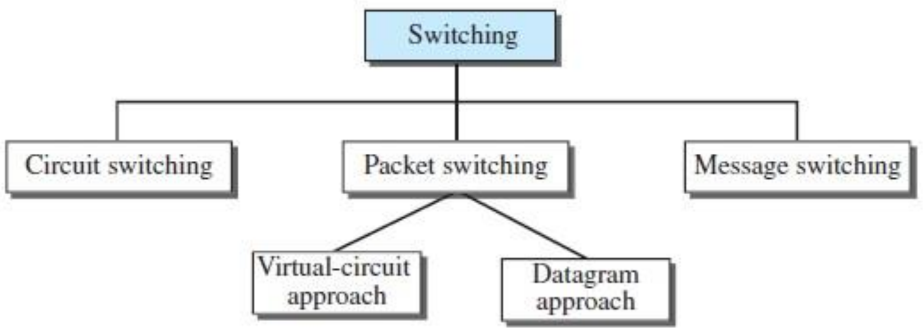
SWITCHING

- To make communication among multiple devices efficiently, a process used is called switching.
- A switched network consists of a series of interlinked nodes called switches.

Type of switching

- Circuit Switching
- Packet Switching
- Message Switching

Figure 8.2 *Taxonomy of switched networks*



Advantages of packet switching over circuit switching are as follows:

- Circuit switching is suitable for **voice communication**. When circuit switched links are used for data transmission, the link is often idle and its facilities wasted.

- The **data rate** of circuit switched connections for data transmission is very slow.

- Circuit switching is **inflexible**. Once a circuit has been established, that the path taken by all parts of the transmission whether or not it remains the most efficient.

- Circuit switching treats all transmission as equal. That means, there is no priority among the transmission of data.

The mostly widely used switching technique for data transmission *is packet switching.*
In this, the data are transmitted in the form of *packets*.
If the length of the packet is too long then it is broken-up into multiple packets.
Each packet contains data and also a header with control information.

PACKET SWITCHING:
There are two popular approaches to packet switching:

- Datagram approach and

- Virtual circuit approach

Datagram Approach:

- In the datagram approach, each packet is treated independently from all others.

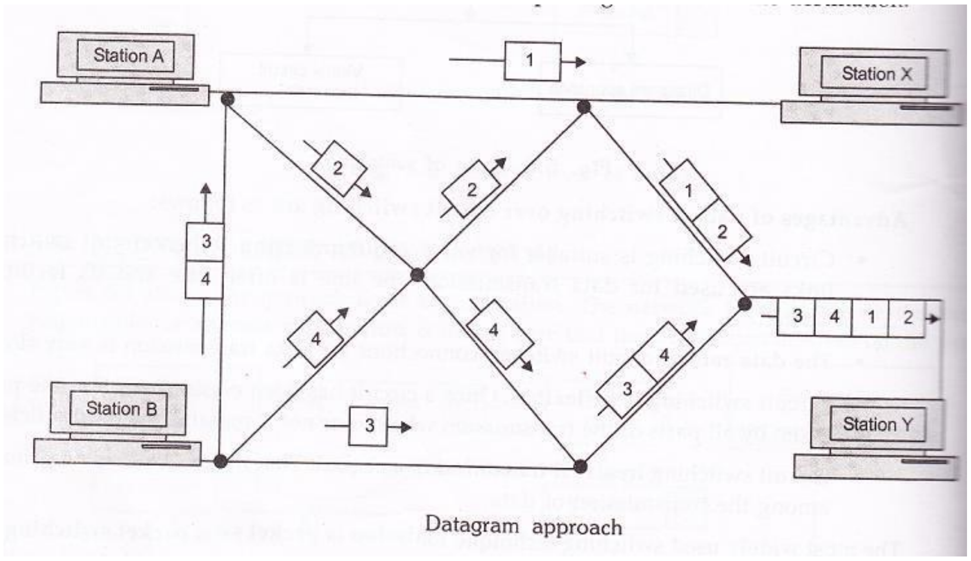
- A datagram is a multipacket of the same message and it works on the principle of 'send' and 'forget'.

The features of datagram are as follows:

- Circuit setup is not needed.
- Each packet contains both source and destination address.
- Each packet routed independently.
- Few packets are lost during crash.
- No effect or router failure.

Example

- The below figure shows how the datagram approach can be used to deliver four packets from station A to station Y.
- In this example, all four packets belong to the same message but may go by different paths to reach their destination.
- This approach can cause the datagrams of a transmission to arrive at their destination out of order.
- In most protocols, it is the responsibility of transport layer to reorder the datagrams before passing them on to the destination.

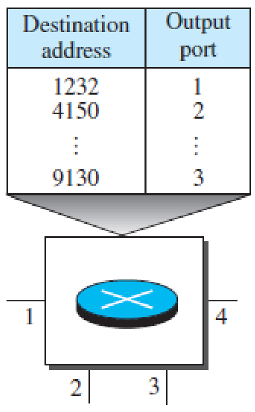


Datagram approach

Routing Table

In this type of network, each switch (or packet switch) has a routing table which is based on the destination address. The routing tables are dynamic and are updated periodically. The destination addresses and the corresponding forwarding output ports are recorded in the tables. This is different from the table of a circuit switched network (discussed later) in which each entry is created when the setup phase is completed and deleted when the teardown phase is over

Figure 8.8 *Routing table in a datagram network*



Destination address	Output port
1232	1
4150	2
⋮	⋮
9130	3

A switch in a datagram network uses a routing table that is based on the destination address.

Virtual Circuit Approach:

- In the virtual circuit approach, the relationship between all packets belonging to a message or session is preserved.
- A single route is chosen between sender and receiver at the beginning of the session.
- When the data are sent, all packets of the transmission travel one after another along that route.

Virtual circuit transmission is implemented in two formats:

- Switched Virtual Circuit (SVC)
- Permanent Virtual Circuit (PVC)

Switched Virtual Circuit (SVC)

- In the **switched virtual circuit (SVC)** method, a virtual circuit is created whenever it is needed exits only for the duration of the specific exchange.
- If the station A wants to send four packets to station X, first it requests the establishment of a connection to station X.
- Once the connection is established, the packets are sent one after another and in sequential order. When the last packet has been received, the connection is released and that virtual circuit ceases to exist.
- Only one single route exists for the duration of transmission. Each time that station A wants to communicate with station X, a new route is established.

Permanent Virtual Circuit (PVC)

- In the **permanent Virtual Circuit (PVC)** method, the same virtual circuit is provided between two users on a continuous basis.

- This circuit is dedicated to the specific users. No one else can use it, because it is always in place.
- It can be used without connection establishment and connection termination.

Two SVC users may get a different route every time they request a connection whereas two PVC users always get the same route.

Comparison between Virtual circuit and Datagram

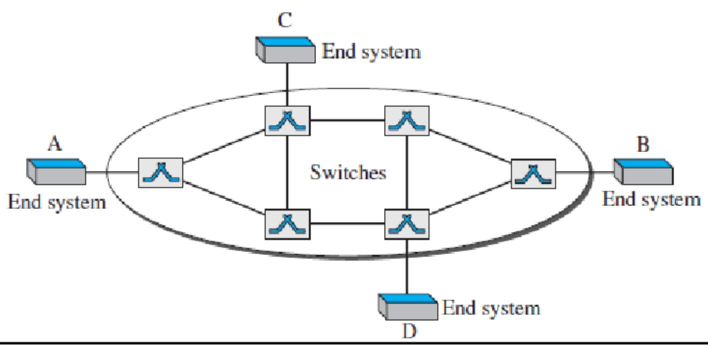
Datagram approach	Virtual circuit approach
In datagram approach, each packet is treated independently, thus they can follow different routes.	In virtual circuit approach, all packets follow the same route.
Packets can arrive at the destination in different order.	Packets should reach the destination in the same order.
Connection establishment is not required before transmission	Connection establishment is required.

Virtual-Circuit Networks – characteristics

A **virtual-circuit network** is a cross between a circuit-switched network and a datagram network. It has some characteristics of both.

1. As in a circuit-switched network, there are **setup and teardown phases** in addition to the data **transfer phase**.
2. Resources can be allocated during the setup phase, as in a circuit-switched network, or on demand, as in a datagram network.
3. As in a datagram network, data are packetized and each packet carries an address in the header. However, the address in the header has local jurisdiction (it defines what the next switch should be and the channel on which the packet is being carried), not end-to-end jurisdiction. The reader may ask how the intermediate switches know where to send the packet if there is no final destination address carried by a packet. The answer will be clear when we discuss virtual-circuit identifiers in the next section.
4. As in a circuit-switched network, all packets follow the same path established during the connection.
5. A virtual-circuit network is normally implemented in the data-link layer, while a circuit-switched network is implemented in the physical layer and a datagram network in the network layer. But this may change in the future.

Figure 8.10 *Virtual-circuit network*

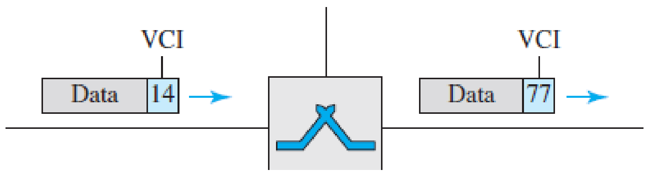


Addressing
In a virtual-circuit network, two types of addressing are involved: global and local (virtual-circuit identifier).

Global Addressing
A source or a destination needs to have a global address—an address that can be unique in the scope of the network or internationally if the network is part of an international network. However, we will see that a global address in virtual-circuit networks is used only to create a virtual-circuit identifier, as discussed next.

Virtual-Circuit Identifier
The identifier that is actually used for data transfer is called the ***virtual-circuit identifier*** **(VCI)** or the ***label.*** A VCI, unlike a global address, is a small number that has only switch scope; it is used by a frame between two switches.

Figure 8.11 *Virtual-circuit identifier*



Three Phases
As in a circuit-switched network, a source and destination need to go through three phases in a virtual-circuit network: **setup, data transfer, and teardown**.

- In the setup phase, the source and destination use their global addresses to help switches make table entries for the connection.
- In the teardown phase, the source and destination inform the switches to delete the corresponding entry.
- Data transfer occurs between these two phases.

11. Compare circuit switching with packet switching Nov/Dec 2011

Circuit Switching Vs Packet Switching
Introduction

- In circuit switching dedicated communication path is available between two stations.
- It is easier to double the capacity of a packet switched network than a circuit network.
- A circuit network is heavily dependent on the number of channel available.
- It is easier to expand a packet Switching System.
- Circuit switched technologies takes double the cost for more boxes.
- Example: Internet Traffic of the telephone network

Circuit Switching:

Advantages

- Circuit is dedicated to the call-no interference, no sharing
- Guaranteed the full bandwidth for the duration of the call

- Guaranteed quality of service

Disadvantages

- Inefficient-the equipment may be unused for a lot of the call, if no data is being sent, the dedicated still remains open
- Takes a relatively long time to set up the circuit
- During a crisis or disaster, the network may become unstable or unavailable.
- It was primarily developed for voice traffic rather than data traffic.

Packet Switching:

Advantages

- More security
- Bandwidth used to full potential
- Devices of different speeds can communicate
- Not affected by line failure(redirects signal)
- Availability-do not have to wait for a direct connection to become available
- During a crisis or disaster, when the public telephone network might stop working, e-mails and texts can still be sent via packet switching

Disadvantages

- Under heavy use there can be a delay
- Data packets can get lost or become corrupted.
- Protocols are needed for a reliable transfer
- Not so good for some types data streams.

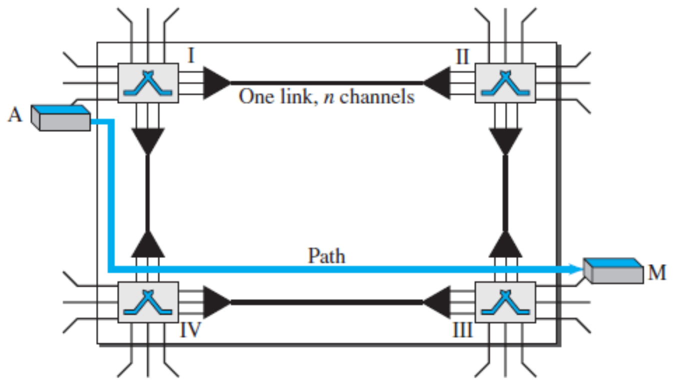
  Example: Real-Time Video streams can lose frames due to the way packets arrive out of sequence.

12. Explain in detail about circuit switched networks

A **circuit-switched network** consists of a set of switches connected by physical links. A connection between two stations is a dedicated path made of one or more links. However, each connection uses only one dedicated channel on each link.

Figure 8.3 shows a trivial circuit-switched network with four switches and four links. Each link is divided into *n* (*n* is 3 in the figure) channels by using FDM or TDM

Figure 8.3 *A trivial circuit-switched network*



We have explicitly shown the multiplexing symbols to emphasize the division of the link into channels even though multiplexing can be implicitly included in the switch fabric.

The end systems, such as computers or telephones, are directly connected to a switch. We have shown only two end systems for simplicity. When end system A needs to communicate with end system M, system A needs to request a connection to M that must be accepted by all switches as well as by M itself. This is called the **setup phase;** a circuit (channel) is reserved on each link, and the combination of circuits or channels defines the dedicated path. After the dedicated path made of connected circuits (channels) is established, the **data-transfer phase** can take place. After all data have been transferred, the circuits are torn down.

We need to emphasize several points here:

❑ Circuit switching takes place at the physical layer.

❑ Before starting communication, the stations must make a reservation for the resources to be used during the communication. These resources, such as channels (bandwidth in FDM and time slots in TDM), switch buffers, switch processing time, and switch input/output ports, must remain dedicated during the entire duration of data transfer until the **teardown phase.**

❑ Data transferred between the two stations are not packetized (physical layer transfer of the signal). The data are a continuous flow sent by the source station and received by the destination station, although there may be periods of silence.

❑ There is no addressing involved during data transfer. The switches route the data based on their occupied band (FDM) or time slot (TDM). Of course, there is end-to end addressing used during the setup phase,

Three Phases

The actual communication in a circuit-switched network requires three phases: connection setup, data transfer, and connection teardown.

Setup Phase

Before the two parties (or multiple parties in a conference call) can communicate, a dedicated circuit (combination of channels in links) needs to be established. The end systems are normally connected through dedicated lines to the switches, so connection setup means creating dedicated channels between the switches. For example, in Figure 8.3, when system A needs to connect to system M, it sends a setup request that includes the address of system M, to switch I. Switch I finds a channel between itself and switch IV that can be dedicated for this purpose. Switch I then sends the request to switch IV, which finds a dedicated channel between itself and switch III. Switch III informs system M of system A's intention at this time.

In the next step to making a connection, an acknowledgment from system M needs to be sent in the opposite direction to system A. Only after system A receives this acknowledgment is the connection established.

Data-Transfer Phase
After the establishment of the dedicated circuit (channels), the two parties can transfer data.

Teardown Phase
When one of the parties needs to disconnect, a signal is sent to each switch to release the resources.

OLD REGULATION - UNIVERSITY QUESTIONS

B.E/B.TECH NOVEMBER/DECEMBER 2014 (2008 regulation)

2 MARKS

1. What is meant by framing (Q.NO 44)
2. Define hamming distance (Q.NO 50)

16 MARKS

1. Discuss the issues in the data link layer (16) (Q.NO 9)
2. Explain in detail the error detecting codes (16) (Q.NO 10)

B.E/B.Tech April May 2015

2 MARKS

1. What do you mean by error control?(Q.NO 34)
2. Define flow control (Q.NO 27)

16 MARKS

1. Discuss in detail about Internet Architecture (16) (Q.NO 6)
2. What is the need for error detection? Explain with typical examples. Explain methods used for error detection and error correction (16) (Q.NO 10 & 13)

B.E/B.Tech Nov-Dec 2015

2 MARKS

1. State the issues of data link layer (Q.NO 37)
2. Define the term protocol (Q.NO 45)

16 MARKS

1. Draw the OSI network architecture and explain the functionalities of every layer in detail (16) (Q.NO 5)
2. Explain the various flow control mechanisms (16) (Q.NO 11)

B.E/B.Tech April-May 2016

2 MARKS

1. Define flow control. (Q.NO 27)
2. Write the parameters used to measure network performance (Q.NO 3)

16 MARKS

1. Explain any two error detection mechanism in detail (16) (Q.NO 10)
2. Explain in detail about HDLC & PPP (8+8) (Q.NO 9)

B.E/B.Tech Nov-Dec 2016

2 MARKS

1. List the services provided by data link layer (Q.NO 37)
2. Write the mechanism of stop and wait flow control (Q.NO 46)

16 MARKS

1. Draw the OSI network architecture and explain the functionalities of every layer in detail (16) (Q.NO 5)
2. a)Discuss in detail about the network performance measures (8) (Q.NO 8)
   b) Explain selective-repeat ARQ flow control method.(8) (Q.NO 11)

B.E/B.Tech April-May 2017

PART A

1. Distinguish between packet switched & circuit switched networks. (Q.NO 51)
2. What is meant by bit stuffing? Give example (Q.NO 40)

PART B

1. i) Explain the challenges faced in building a network (10) (Q.NO 4)
   ii) Obtain the 4-bit CRC code for the data bit sequence 10011011100 using the polynomial x^4+x^2+1 (3) (Q.NO 14)

2.i) With a protocol graph explain the architecture of internet (7) (Q.NO 6)
   ii) Consider a bus LAN with a number of equally spaced stations with a data rate of 9 Mbps and a bus length of 1 km. What is the mean time to send a frame of 500 bits to another station, measured from the beginning of transmission to the end of reception? Assume a propagation speed of 150 m/s. If two stations begin to monitor and transmit the same time, how long does it need to wait before interference is noticed? (6) (Q.NO 15)

B.E/B.Tech Nov-Dec 2017

PART A

1. Define the terms: Bandwidth & Latency (Q.NO 52)
2. Compare Byte oriented versus Bit-oriented protocol (Q.NO 53)

PART B

1. With a neat sketch, explain the architecture of an OSI seven layer model (13) (Q.NO 5)
2. Discuss the approaches used for error detection in networking (13) (Q.NO 10)

PART C

1. Outline the steps involved in building a computer network. Give the detailed description for each step (15) (Q.NO 4)

MADHA
Expertise | Empathy | Excellence
ENGINEERING COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COMMON FOR: DEPARTMENT OF INFORMATION TECHNOLOGY

EC8691 – MICROPROCESSORS AND MICROCONTROLLERS

R – 2017

LECTURE NOTES

UNIT-I

The 8086 Microprocessor

Introduction to 8086 – Microprocessor architecture – Addressing modes - Instruction set and assembler directives – Assembly language programming – Modular Programming - Linking and Relocation - Stacks - Procedures – Macros – Interrupts and interrupt service routines – Byte and String Manipulation.

1. **Introduction to 8086**

8086-HARDWARE ARCHITECTURE

AUQ: Explain the features of 8086 microprocessor. (May 2011, 8 Marks)

The features of 8086 are:

- The 8086 is a 16 bit processor
- The 8086 has a 16 bit data bus
- The 8086 has a 20 bit address bus
- Direct addressing capability 1M byte of memory(2^{20})
- It provides fourteen 16 bit register
- 24 operand addressing modes
- Bit, byte, word and block operations.
- 8 and 16 bit signed and unsigned arithmetic operations including multiply and divide
- Four general purpose 16 bit registers: AX, BX, CX, DX
- Two pointer group registers: stack pointer (SP), Base pointer(BP)
- Two index group registers: source index (SI), destination index (DI)
- Four segment registers: code segment (CS), Data segment (DS), Stack segment (SS), Extra segment(ES)
- 6 Status flag and 3 control flags.
- Memory is byte addressable- each address stores an 8 bit value.
- Address can be up to 32 bit long, resulting in 4GBof memory.
- Range of clock rates: 5MHZ for8086, 8MHZ for8086-2, 10MHZ for8086-1
- Multibus system compatible interface
- Available in 40 pin plastic package and lead cerdip.

2. **8086 Microprocessor Architecture:**

   AUQ: Explain the internal architecture of 8086 microprocessor.(Dec-2003,04,06,08,11,12,13, May-2003,05,07,08,10,11,15, May 2016, Dec 2016, May 2017)

The internal functions of 8086 processor are partitioned logically into two processing units.
The 8086 CPU is divided into two independent functional parts,

1. **Bus Interface Unit (BIU)**
2. **Execution Unit (EU)**

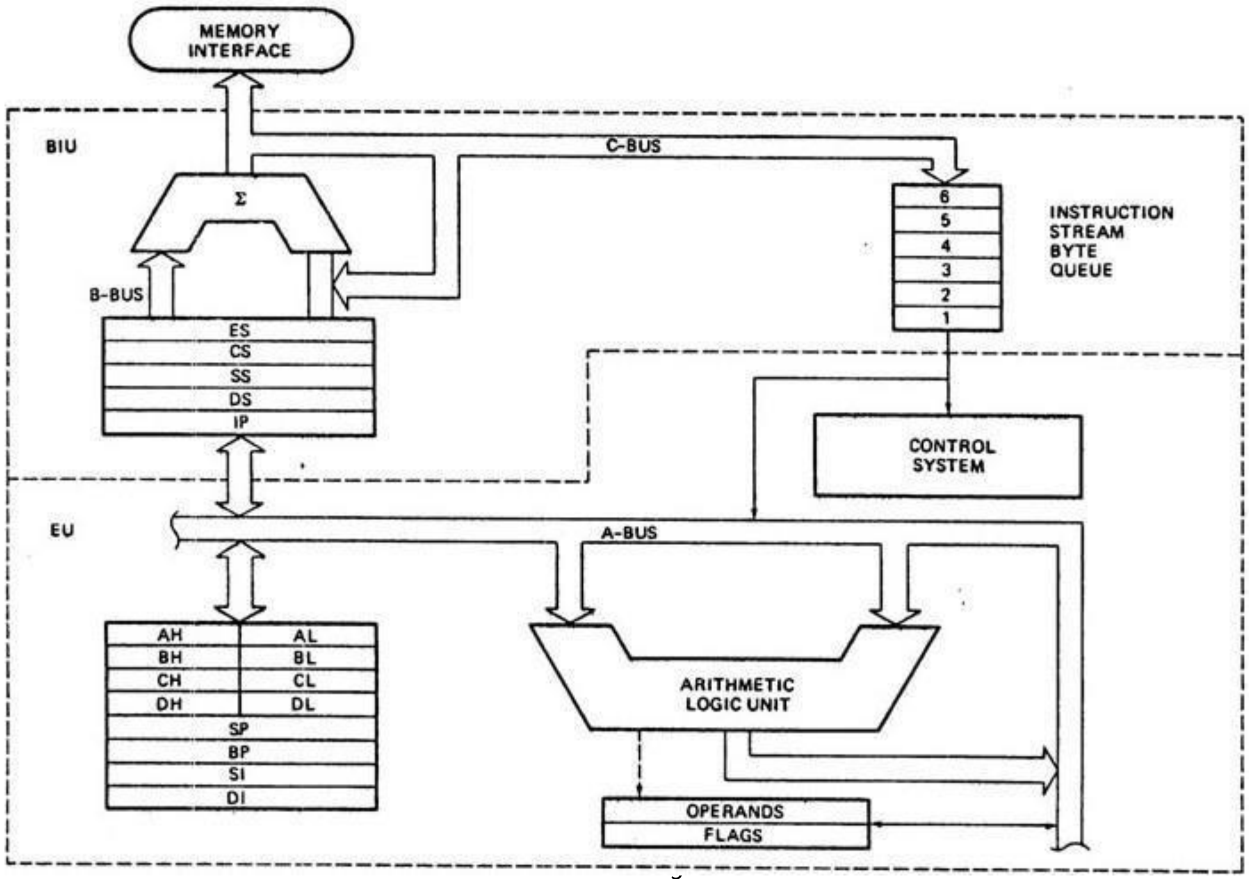
- ✓ The BIU and EU function independently.
- ✓ The BIU interface the 8086 to the outside world. The BIU fetches, reads data from memory and ports, and writes data to memory and I/O ports.
- ✓ EU receives program instruction codes and data from the BIU, executes these instructions and stores the results either in general registers or output them puts all its data through the BIU.

The BIU contains

1. Segment Registers, 2. Instruction Pointer (IP), 3.Instruction Queue

The EU contains

1. ALU
2. General purpose registers
3. Index registers
4. Pointers
5. Flag register



General Purpose Registers

All general registers of the 8086 microprocessor can be used for arithmetic and logic operations. The 16 bit general purpose registers are

1. Accumulator register (AX)
2. Base register (BX)
3. Count register (CX)
4. Data register (DX)

(i) Accumulator register (AX)

- ✓ Accumulator (AX) is a 16 bit register; consists of two 8-bit registers AL and AH.
- ✓ AL contains the low-order byte of the word, and AH contains the higher order byte.
- ✓ Accumulator can be used for Input/ Output (I/O) operations and string manipulation.

(ii) Base register (BX)

- ✓ Base register (BX) is a 16 bit register; consist of two 8-bit registers BL and BH.
- ✓ BL consist the lower order byte of the word, and BH contains the higher order byte.
- ✓ BX register contains a data pointer used for based, based indexed or register indirect addressing.

(iii) Count register (CX)

- ✓ Counter register (CX) is a 16 bit register; consists of two 8-bit registers CL and CH.
- ✓ CL register contain the low order byte of the word, and CH contains the high order byte.
- ✓ Count register can be used as a counter in string manipulation and shift/ rotate instructions.

(iv)Data register (DX)

- ✓ Data register (CX) is a 16 bit register; consists of two 8-bit registers DL and DH.
- ✓ DL register contain the low order byte of the word, and DH contains the high order byte.
- ✓ Data register can be used as a port number in I/O operations.
- ✓ In integer 32-bit multiply and divide instruction the DX register contains higher order word of the initial or resulting number.

Segment Registers

There are four different 64 KB segments for instructions, stack, data and extra data.

The segment registers are:

1. Code segment (CS)
2. Stack segment (SS)
3. Data segment (DS)
4. Extra segment (ES)

(i) Code segment (CS)

- ✓ Code segment is a 16-bit register containing address of 64 KB segment with processor instructions.
- ✓ The processor uses CS register for all accesses to instructions referenced by instruction pointer (IP).
- ✓ CS register cannot be changed directly.
- ✓ The CS register is automatically updated during FAR JUMP, FAR CALL and FAR RET instructions

(ii) Stack segment (SS)

- ✓ Stack segment is a 16-bit register containing address of 64KB segment with program stack.
- ✓ By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers are located in the stack segment.
- ✓ SS register can be changed directly using POP instruction.

(iii) Data segment (DS)

- ✓ Data segment is a 16-bit register containing address of 64KB segment with program data.
- ✓ By default, the processor assumes that all data referenced by general registers (AX, BX, CX, and DX) and index register (SI, DI) is located in the data segment.
- ✓ DS register can be changed directly using POP and LDS instructions.

(iv) Extra segment (ES)

- ✓ Extra segment is a 16-bit register containing address of 64KB segment, usually with program data.
- ✓ By default, the processor assumes that the DI register references the ES segment in string 'manipulation instructions.
- ✓ ES register can be changed directly using POP and LES instructions.
- ✓ It is possible to change default segments used by general and index registers by prefixing instructions with a CS, SS, DS or ES prefix.

Pointer Registers

(i) Stack Pointer (SP)

Stack pointer is a 16-bit register pointing to program stack.

(ii) Base Pointer (BP)

- ✓ Base pointer is a 16-bit register pointing to data in the stack segment.
- ✓ BP register is usually used for based, based indexed or register indirect addressing.

Index Registers: (i) Source Index (SI)

- ✓ Source index is a 16-bit register.
- ✓ SI is used for indexed, based indexed and register indirect addressing, as well as a source data address in string manipulation instructions.

(ii) Destination Index (DI)

- ✓ Destination index is a 16-bit register.
- ✓ DI is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions.

Instruction Pointer (IP)

- ✓ Instruction pointer is a 16-bit register. The operation is same as the program counter.
- ✓ The IP register is updated by the BIU to point to the address of the next instruction.
- ✓ Programs do not have direct access to the IP, but during execution of a program the IP can be modified or saved and restored from the stack.

Flag register

Flag register is a 16-bit register containing nine 1-bit flags:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

Six status or condition flags (OF, SF, ZF, AF, PF, CF)

Three control flags (TF, DF, IF)

- **Overflow Flag (OF)** – It is set if an overflow occurs, i.e., a result is out of range.
- **Sign Flag (SF)** – It is set if the most significant bit of the result is set.
- **Zero Flag (ZF)** – It is set if the result is zero.
- **Auxiliary carry Flag (AF)** – It is set if there is a carry out of bit 3 during addition or borrow by bit 3 during subtraction. This flag is used exclusively for BCD arithmetic.
- **Parity Flag (PF)** – It is set to 1 if the low-order 8-bits of the result contain an even number of 1s.
- **Carry Flag (CF)** – It is set if carry from or borrow to the most significant bit during last result calculation.
- **Trap Flag (TF)** – if set, a trap is executed after each instruction.
- **Direction Flag (DF)** – Used by string manipulation instructions. If set then string manipulation instructions will auto- decrement index registers. If cleared then the index registers will be auto-incremented.
- **Interrupt-enable Flag (IF)** - Setting this bit enables maskable interrupts.

Instruction Queue:

- ✓ . The instruction queue is a First-In-First-out (FIFO) group of registers where 6 bytes of instruction code is pre-fetched from memory.
- ✓ ***It is being done to speedup program execution by overlapping instruction fetch and execution. This mechanism is known as PIPELINING.***
- ✓ If the queue is full, the BIU does not perform any bus cycle. If the BIU is not full and can store atleast 2 bytes and EU does not request it to access memory, the BIU may pre-fetch instructions.
- ✓ If the BIU is interrupted by the EU for memory access while pre-fetching, the BIU first completes fetching and then services the EU. In case of JMP instruction, the BIU will reset the queue and .begin refilling after passing the new instruction to the EU.

ALU: Arithmetic and Logic Unit

ALU is a 16 bit register. It can add, subtract, increment, decrement, complement, shift numbers and performs AND, OR, XOR operations.

Control unit: "

Generates timing and control signals to perform the internal operations of the microprocessor.

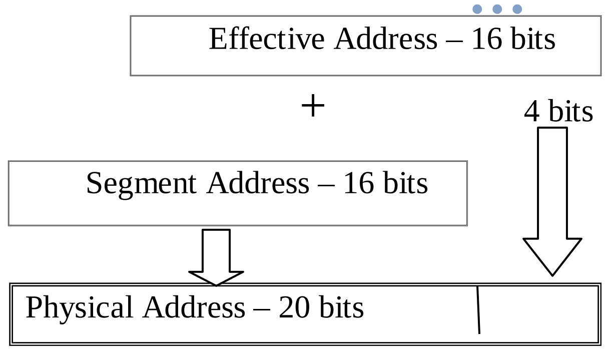
3. The 8086 Addressing Modes

AUQ: What are the addressing modes in 8086? Explain with example.(Dec-2006,07,08,10,11, May2006,07,08,09,11,15, May 2016, Dec 2016)

Addressing modes in 8086:

The 8086 memory addresses are calculated by adding the segment register contents to an offset address. The offset address calculation depends on the addressing mode being used. The total number of address lines in the 8086 is 20 whereas the segment registers are 16 bits. The actual address in memory (effective address) is calculated as per the following steps.

- The segment register contents are multiplied by 1OH, thus, shifting the contents left by 4 bits. This results in the starting address of the segment in memory.
- The offset address is calculated. The offset address is basically the offset of the actual memory location from the starting location of the segment. The calculation of this offset value depends on the addressing mode being used.
- The offset address is added to the starting address of the segment to get the effective address, i.e. the actual memory address.


Effective Address – 16 bits
+
4 bits
Segment Address – 16 bits
Physical Address – 20 bits

Suppose a segment register contents are xyzwH, and the offset value calculated is abcdH, then:

- Starting address of the segment
- Offset address
- Effective address

The addressing modes specify the location of the operand and also how its location may be determined. The following addressing modes are supported in the 8086.

- Register Addressing Mode
- Immediate Addressing Mode
- Direct Memory Addressing Mode
- Register Indirect Addressing Mode
- Base plus Index Register Addressing Mode
- Register Relative Addressing Mode
- Base plus Index Register Relative Addressing Mode
- String Addressing Mode

Register Addressing Mode

When both destination and source operands reside in registers, the addressing mode is known as register addressing mode. Following are the examples:

- ➢ MOVAX,BX

  Move the contents of BX register to AX register. The contents of BX register remain unchanged.

- ➢ AND AL, BL

  AND the contents of AL register with the contents of BL register and place the resultant contents in AL register.

Immediate Addressing Mode

When one of the operands is part of the instruction, the addressing mode is known as immediate addressing mode. Examples are given below

- MOV CX, 2346H

Copy into CX the 16-bit data 2346H.

➢ SUB AL, 24H

Subtract 24H from the contents of AL register and put the result in AL register.

Direct Memory Addressing Mode

In this mode, the 16-bit offset address is part of the instruction as displacement field. It is stored as 16-bit unsigned or 8-bit sign-extended number.

➢ MOV (4625H), *Dl*

Copy the contents of DL register into memory locations calculated from Data Segment register and offset 4625H.

➢ OR AL, (3030H)

OR the contents of AL register with the contents of memory location calculated from DS register and offset 3030H.

Register Indirect Addressing Mode

In this addressing mode, the offset address is specified through pointer register or index register.

For index register, the SI (Source Index) register or DI (Destination Index) register may be used, whereas for pointer register, BX (Base Register) register or BP (Base Pointer) register may be used. Following are some examples of the application of the register indirect addressing mode.

➢ MOV AL, (BP)

Copy into AL register the contents of memory location, whose address is calculated using offset as contents of BP register and the contents of DS register.

Base plus Index Register Addressing Mode

In this mode, both base register (BP or BX) and index register (SI or DI) are used to indirectly address the memory location. An example is given below.

➢ MOV (BX + DI), AL

Copy the contents of AL register into memory location whose address is calculated using the contents of DS (Data Segment), BX (Base Register) and DI (Destination Index) registers.

Register Relative Addressing Mode

This mode is similar to base plus index addressing mode. In this mode, the offset is calculated using either a base register (BP, BX) or an index register (SI, DI) and the displacement specified as an 8-bit or a 16-bit number, as part of the instruction.

- MOV AX, (DI + 06)

Copy to AL the contents of memory location whose address is calculated using DS (Data Segment), DI (Destination Index) register with displacement of 06, and copy to AH the contents of the next higher memory location.

Base plus Index Register Relative Addressing Mode

This addressing mode is basically the combination of base plus index register addressing mode and register relative addressing mode. To find the address of the operand in memory, a base register (BP or BX), an index register (DI or SI) and the displacement which is specified in instruction is used along with the data segment register. For example:

- MOV (BX + DI + 2), CL

Copy the contents of the CL register to the memory location whose address is calculated using DS (Data Segment), BX (Base Register) and DI (Destination Index) registers and 02 as displacement.

String Addressing Mode

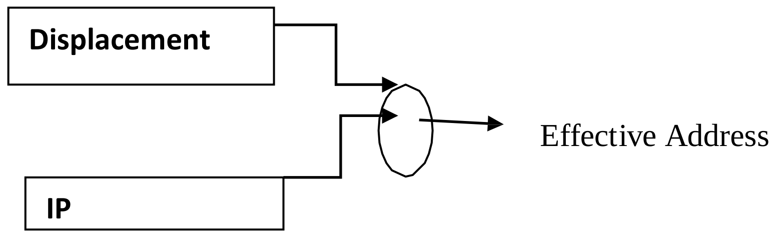
In this addressing mode, the string instruction uses index registers implicitly to access memory.

Example: MOVSB

Copy the byte from the source string location determined by DS and SI to the destination string location determined by ES and DI.

- The addressing modes for **branch related instructions** are
  - **Intrasegment direct (within the same segment)**
  - **Intrasegment Indirect**
  - **Intersegment Direct (Control transfer to different segment)**
  - **Intersegment Indirect**

Intrasegment direct (within the same segment)



If the displacement is 8 bit long, it is called short jump SJMP
If the displacement is 16 bit long, it is called Long jump LJMP.
For example **CALL NEAR**

A NEAR JMP is a jump where destination location is in the same code segment. In this case only IP is changed.

Intrasegment Indirect

The content of register or memory is accessed using any of the above data related addressing mode except immediate mode**.**

Intersegment Direct (control transfer is in different segment)

The purpose of the addressing mode is to provide a means of branching from one code segment to another .Replaces the content of IP with the part of the instruction and the contents of the CS with another part of instruction**.**

Example: FAR CALL

A FAR JMP is a jump where destination location is from a different segment. In this case both IP and CS are changed as specified in the destination.

Intersegment Indirect

The content of memory block containing 4 bytes.ie IP (LSB),IP(MSB),CS (LSB),and CS(MSB) sequentially .The starting address of the memory block may be referred using any of the addressing mode except immediate mode.

4. <u>The instruction set of 8086.</u>

Explain the instruction set of 8086 microprocessor.

Give three examples for the following 8086 microprocessor instructions: String Instructions, Process Control Instruction, Program Execution Transfer Instructions and Bit manipulation Instructions. (May 2010)(June 2016)

Explain the data transfer, arithmetic and branch instructions with examples. (June 2016)

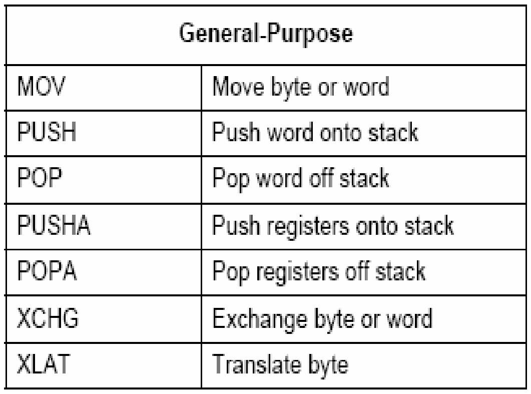
Intel 8086 has approximately 117 instructions. These instructions are used to transfer data between registers, register to memory, memory to register or register to I/O ports and other instructions are used for data manipulation.

But in Intel 8086 operations between memory to memory is not permitted. These instructions are classified in to six-groups as follows.

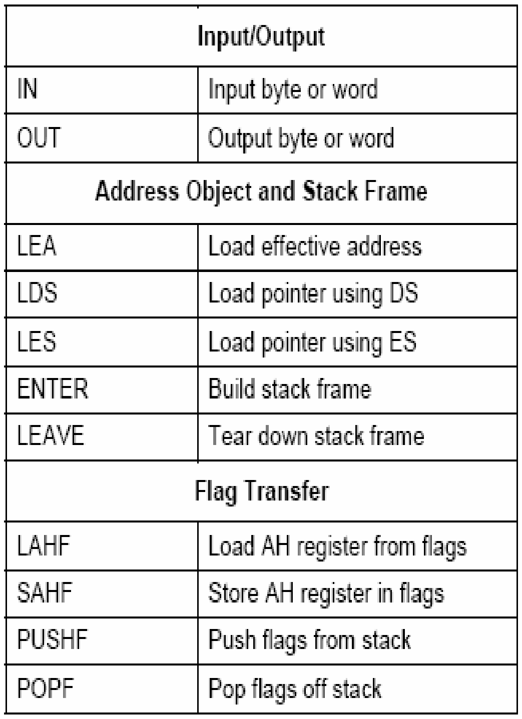
1. Data Transfer Instructions
2. Arithmetic Instructions
3. Bit Manipulation Instructions
4. String Instructions
5. Program Execution Transfer Instructions

6. Processor Control Instructions

Data Transfer Instructions



General-Purpose	
MOV	Move byte or word
PUSH	Push word onto stack
POP	Pop word off stack
PUSHA	Push registers onto stack
POPA	Pop registers off stack
XCHG	Exchange byte or word
XLAT	Translate byte



Input/Output	
IN	Input byte or word
OUT	Output byte or word
Address Object and Stack Frame	
LEA	Load effective address
LDS	Load pointer using DS
LES	Load pointer using ES
ENTER	Build stack frame
LEAVE	Tear down stack frame
Flag Transfer	
LAHF	Load AH register from flags
SAHF	Store AH register in flags
PUSHF	Push flags from stack
POPF	Pop flags off stack

1. MOV

MOV destination, source

This (Move) instruction transfers a byte or a word from the source operand to the destination operand.

(DEST) ← (SRC), DEST = Destination, SRC = Source

Example:

MOV AX, BX

MOVAX, 2150H

MOV AL, [1135]

MOV [4186], AL

MOV SS, DX

MOV [BX], DS

2. PUSH

PUSH Source

This instruction decrements SP (stack pointer) by 2 and then transfers a word from the source operand to the top of the stack now pointed to by stack pointer.

(SP) ← (SP)-2

((SP) + 1: (SP)) ← (SRC)

Example:

PUSH SI

PUSH BX

3. POP

POP destination

This instruction transfers the word at the current top of stack (pointed to by SP) **to** the destination operand and then increments SP by 2, pointing to the new top of the stack.

(DEST) ← ((SP) + 1:(SP))

(SP) ← (SP) + 2

Example:

POP DX

POP DS

4. LAHF

Load Register AH from Flags

This instruction copies Sign flag(S), Zero flag (Z), Auxiliary flag (AC). Parity flag (P) and Carry flag (C) of 8085 into bits 7, 6, 4, 2 and 0 respectively, of register AH. The content of bits 5, 3 and 1 is undefined.

AH←

S	Z	X	A	X	P	X	C

5. SAHF

Store Register AH into Flags

This instruction transfers bits 7, 6, 4, 2 and 0 from register AH into S, Z, AC, P and C flags respectively, thereby replacing the previous values.

S	Z	X	A	X	P	X	C

←AH

6. XCHG

XCHG destination, source

This (Exchange) instruction switches the contents of the source and destination operands.

(Temp) ← (DEST)

(DEST) ← (SRC)

(SRC) ← (Temp)

Example:

XCHG AX, BX

XCHG BL, AL

7. XLAT

XLAT table

- This (Translate) instruction replaces a byte in the AL register with a byte from a 256-byte, user-coded translation table.
- XLAT is useful for translating characters from one code to another like ASCII to EBCDIC. Register BX is the starting point of the table. The byte in AL is used as an index into the table and is replaced by the byte at the offset in the table corresponding to AL's binary value.

AL ← ((BX) + (AL))

Example :

XLAT ASCII_TAB

XLAT Table_3

8. LEA

LEA destination, source

This (Load Effective Address) instruction transfers the offset of the source operand (memory) to the destination operand (16-bit general register).

(REG) ← EA

Example :

LEA BX , [BP] [DI] LEA SI, [BX + 02AF H]

9. LDS

LDS destination, source

This (Load pointer using DS) instruction transfers a 32-bit pointer variable from the source operand (memory operand) to the destination operand and register DS.

(REG) ← (EA)

(DS← (EA+2)

Example:

LDS SI, [6AC1H]

10. LES

LES destination, source

This (Load pointer using ES) instruction transfers a 32-bit pointer variable from the source operand (memory operand) to the destination operand and register ES.

(REG) ← (EA)

(ES) ← (EA+2)

Example:

LES DI, [BX]

11. IN

IN accumulator, port

This (Input) instruction transfers a byte or a word from an input port to the accumulator (AL or AX).

(DEST) ← (SRC)

Example:

IN AX, DX

IN AL, 062H

12. OUT

OUT port, accumulator

This **(Output)** instruction transfers a byte or a word from the accumulator (AL or AX) to an output port.

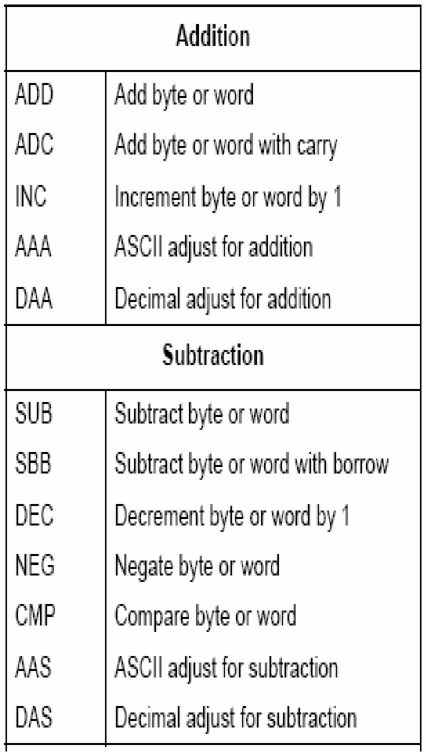
(DEST) ← (SRC)

Example:

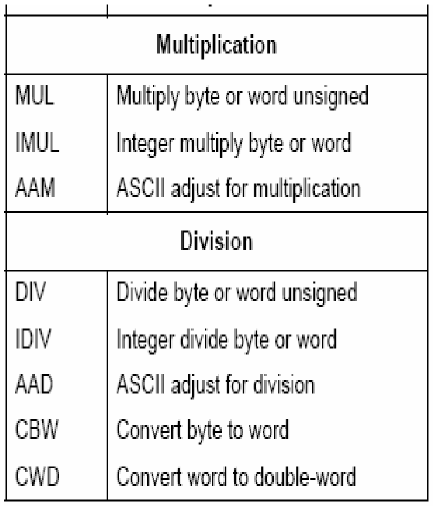
OUT DX, AL

OUT 31, AX

Arithmetic Instructions



Addition	
ADD	Add byte or word
ADC	Add byte or word with carry
INC	Increment byte or word by 1
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition
Subtraction	
SUB	Subtract byte or word
SBB	Subtract byte or word with borrow
DEC	Decrement byte or word by 1
NEG	Negate byte or word
CMP	Compare byte or word
AAS	ASCII adjust for subtraction
DAS	Decimal adjust for subtraction



Multiplication	
MUL	Multiply byte or word unsigned
IMUL	Integer multiply byte or word
AAM	ASCII adjust for multiplication
Division	
DIV	Divide byte or word unsigned
IDIV	Integer divide byte or word
AAD	ASCII adjust for division
CBW	Convert byte to word
CWD	Convert word to double-word

1. ADD

ADD destination, source

This **(Add)** instruction adds the two operands (byte or word) and stores the result in destination operand.

(DEST) ← (DEST) + (SRC)

Example:

ADD CX, DX

ADD AX, 1257 H

ADDBX, [CX]

2. ADC

ADC destination, source

This **(Add with carry)** instruction adds the two operands and adds one if carry flag (CF) is set and stores the result in destination operand.

(DEST) ← (DEST) + (SRC) + 1

Example:

ADC AX, BX

ADC AL, 8

ADC CX, [BX]

3. SUB

SUB destination, source

This (Subtract) instruction subtracts the source operand from the destination operand and the result is stored in destination operand.

(DEST) ← (DEST) - (SRC)

Example:

SUB AX, 6541 H

SUB BX, AX

SUB SI, 5780 H

4. SBB

SBB destination, source

This (**Subtract with Borrow**) instruction subtracts the source from the destination and subtracts 1 if carry flag (CF) is set. The result is stored in destination operand.

(DEST) ← (DEST) - (SRC) -1

Example:

SBB BX, CX

SBB AX, 2

5. **CMP**

CMP destination, source

This (Compare) instruction subtracts the source from the destination, but does not store the result.

(DEST) - (SRC)

Example:

CMP AX, 18

CMP BX, CX

6. **INC**

INC destination

This **(Increment)** instruction adds 1 to the destination operand (byte or word).

(DEST) ← (DEST) + 1

Example:

INC BL

INC CX

7. **DEC**

DEC destination

This **(Decrement)** instruction subtracts 1 from the destination operand.

(DEST) ← (DEST)-1

Example:

DEC BL

DEC AX

8. **NEG**

NEG destination

This (Negate) instruction subtracts the destination operand from 0 and stores the result in destination. This forms the 2's complement of the number.

(DEST) ← 0 - (DEST)

Example: NEG AX

NEG CL

9. DAA

This (Decimal Adjust for Addition) instruction converts the binary result of an ADD or ADC instruction in AL to packed BCD format.

If the auxiliary carry flag is set or the low 4 bits of AL are greater than 9, then 06 H is added to AL. If the carry flag is set or the high 4 bits of AL are greater than 9, then 60 H is added to the AL.

10. DAS

This **(Decimal Adjust for Subtraction)** instruction converts the binary result of a SUB or SBB instruction in AL to packed BCD format.

11.AAA

This **(ASCII Adjust for Addition)** instruction adjusts the binary result of ADD or ADC instruction.

If bits 0-3 of AL contain a value greater than 9, or if the auxiliary carry flag (AF) is set, the CPU adds 06 to AL and adds 1 to AH. The bits 4-7 of AL are set to zero.

(AL) ← (AL) + 6

(AH) ← (AH) +1

(AF) ←1

Example:

AAA

Before execution

AH	AL
00	0B

After execution

AH	AL
01	01

12. AAS

This **(ASCII Adjust for Subtraction)** instruction adjusts the binary result of a SUB or SBB instruction.

If D_3 - D_0 of AL>9,

(AL) ← (AL) – 6

(AH) ← (AH) – 1

(AF) ←1

13.MUL

MUL source

- This **(Multiply)** instruction multiply AL or AX register by register or memory location contents.
- Both operands are unsigned numbers.
- If the source is a byte (8 bit), then it is multiplied by register AL and the result is stored in AH and AL.
- If the source operand is a word (16 bit), then it is multiplied by register AX and the result is stored in AX and DX registers.

If 8 bit data, (AX) ← (AL) × (SRC)

If 16 bit data, (AX), (DX) ← (AX) × (SRC)

Example:

MUL25

MUL CX .

MULBL

14. IMUL

IMUL Source

This **(Integer Multiply)** instruction performs a signed multiplication of the source operand and the accumulator.

If 8 bit data, (AX) ← (AL) × (SRC)

If 16 bit data, (AX), (DX) ← (AX) × (SRC)

Example:

IMUL 250

IMUL BL

15. AAM

This (ASCII Adjust for Multiplication) instruction adjusts the binary result of a MUL instruction. AL is divided by 10(0AH) and quotient is stored in AH. The remainder is stored in AL.

(AH) ← (AL/OAH)

(AL) ← Remainder

16. DIV

DIV Source

- This (Division) instruction performs an unsigned division of the accumulator by the source operand.
- It allows a 16 bit unsigned number to be divided by an 8 bit unsigned number, or a 32 bit unsigned number to be divided by a 16 bit unsigned number.
- If byte (8-bit) operation is performed, the 8 bit quotient is stored to AL and 8 bit remainder is stored

to AH register.

- If the source operand is a word (16 bit), the 16 bit quotient is stored in AX and the remainder is stored in DX register.

For 8 bit data, AX / source

(AL) ← Quotient

(AH) ← Remainder

For 16 bit data, AX, DX / Source

(AX) ← Quotient

(DX) ← Remainder

Example:

DIV CX

DIV 321

17.IDIV

IDIV source

This (Integer Division) instruction performs a signed division of the accumulator by the source operand.

For 8 bit data, AX / Source

(AL) ← Quotient

(AH) ← Remainder

For 16 bit data, AX, DX/Source

(AX) ← Quotient

(DX) ← Remainder

Example:

IDIV CL

IDIVAX

18. AAD

This **(ASCII Adjust for Division)** instruction adjusts the unpacked BCD dividend in AX before a division operation. AH is multiplied by 10(0AH) and added to AL. AH is set to zero.

(AL) ← (AH × 0AH) + (AL)

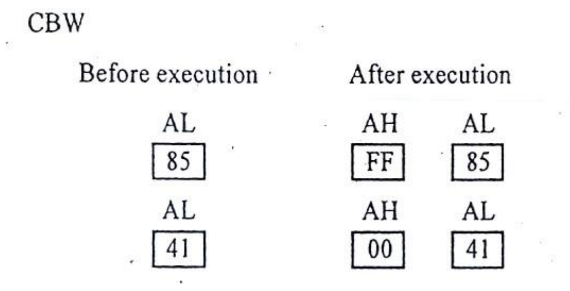
(AH) ←0

19. CBW

This **(Convert Byte to Word)** instruction converts a byte to a word. It extends the sign of the byte in register AL through register AH.

This instruction can be used for 16 bit IMUL or IDIV instruction.

Example:



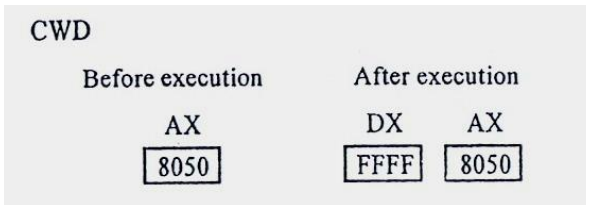
20. CWD

This (Convert Word to Double word) instruction converts a word to a double word. It extends the sign of the word in register AX through register DX.

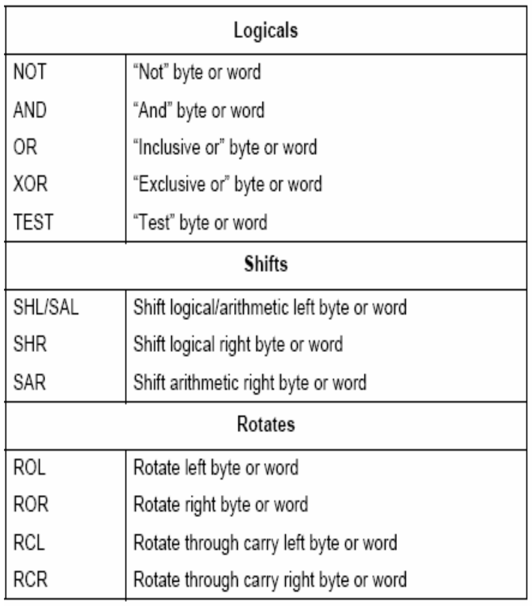
If AX < 8000 H, then DX = 0000 H

If AX > 8000 H, then DX = FFFFH

Example:



Bit Manipulation Instructions



Logicals	
NOT	"Not" byte or word
AND	"And" byte or word
OR	"Inclusive or" byte or word
XOR	"Exclusive or" byte or word
TEST	"Test" byte or word
Shifts	
SHL/SAL	Shift logical/arithmetic left byte or word
SHR	Shift logical right byte or word
SAR	Shift arithmetic right byte or word
Rotates	
ROL	Rotate left byte or word
ROR	Rotate right byte or word
RCL	Rotate through carry left byte or word
RCR	Rotate through carry right byte or word

(i) Logical Instructions : AND, OR, XOR, NOT, TEST

(ii) Shift Instructions : SHL, SAL, SHR, SAR

(iii) Rotate Instructions : ROL, ROR, RCL, RCR

1. AND

AND destination, source

This **(AND)** instruction performs the logical "AND" of the source operand with the destination operand and the result is stored in destination.

(DEST) ← (DEST) "AND" (SRC)

Example:

AND BL, CL

AND AL, 0011 1100 B

2. OR

OR destination, source

This **(OR)** instruction performs the logical "OR" of the source operand with the destination operand and the result is stored in destination.

(DEST) ← (DEST) "OR" (SRC)

Example:

OR AX, BX

OR AL, 0FH

3. XOR

XOR destination, source

This **(Exclusive OR)** instruction performs the logical "XOR" of the two operands and the result is stored in destination operand.

(DEST) ← (DEST) "XOR"(SRC)

4. NOT

NOT destination

This (NOT) instruction inverts the bits (forms the l's complement) of the byte or word.

(DEST) ← 1 's complement of (DEST)

Example:

NOT AX

5. TEST

TEST destination, source

This (TEST) instruction performs the logical "AND" of the two operands and updates the flags but does not store the result.

(DEST) "AND" (SRC)

Example:

TEST AL, 15 H

TEST SI, DI

6. SHL

SHL destination, count

This (Shift Logical Left) instruction performs the shift operation. The number of bits to be shifted is represented by a variable count, either 1 or the number contained in the CL register.

Example

SHL AL, 1

Before execution:

CF

0

AL

1	1	0	0	1	1	0	0

After execution:

CF

1

AL

1	0	0	1	1	0	0	0

7. SAL

SAL destination, count

SAL **(Shift Arithmetic Left)** and SHL **(Shift Logical Left)** instructions perform the same operation and are physically the same instruction.

Example

SAL AL, CL

SAL AL, 1

8. SHR

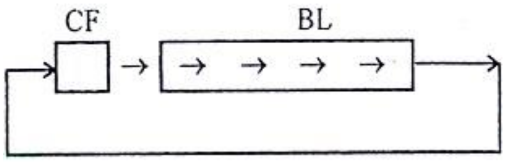
SHR destination, count

This **(Shift Logical Right)** instruction shifts the bits in the destination operand to the right by the number of bits specified by the count operand, either 1 or the number contained in the CL register.

Example

SHR BL, 1

SHR BL, CL



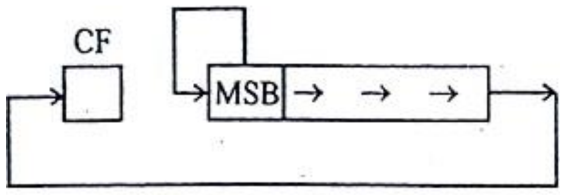
The SHR instruction may be used to divide a number by 2. For example, we can divide 32 by 2,

```
MOV  BL, 32  ;  0010  0000   (32)
SHR  BL, 1   ;  0001  0000   (16)
SHR  BL, 1   ;  0000  1000   (8)
SHR  BL, 1   ;  0000  0100   (4)
SHR  BL, I   ;  0000  0010   (2)
```

9. SAR

SAR destination, count

This **(Shift Arithmetic Right)** instruction shifts the bits in the destination operand to the right by the number of bits specified in the count operand. Bits equal to the original high-order (sign) bits are shifted in on the left, thereby preserving the sign of the original value.



Example :

SAR BL, 1

Before execution:

CF BL

CF		BL							
0		1	1	0	0	1	1	0	0

After execution:

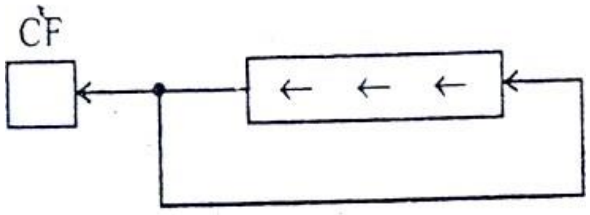
CF BL

0	1	1	1	0	0	1	1	0

10. ROL

ROL destination, count

This **(Rotate Left)** instruction rotates the bits in the byte/word destination operand to the left by the number of bits specified in the count operand.



Example:

ROL AL, 1

Before execution:

CF					AL			
0	1	1	0	0	1	1	0	0

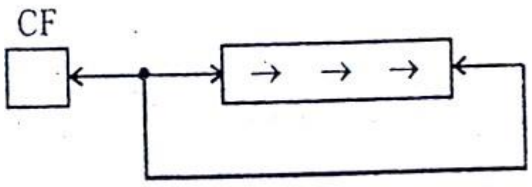
After execution:

CF					AL			
1	1	0	0	1	1	0	0	1

11. ROR

ROR destination, count

This **(Rotate Right)** instruction rotates the bits in the byte/word destination operand to the right by the number of bits specified in the count operand.



Example:

ROR AL, 1

Before execution:

CF		AL							
0		1	1	0	0	1	1	0	0

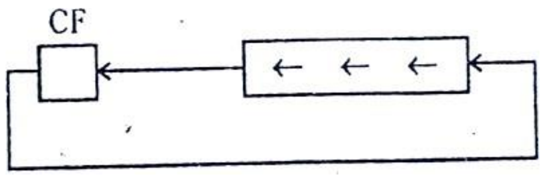
After execution:

CF		AL							
0		0	1	1	0	0	1	1	0

12. RCL

RCL destination, count

This **(Rotate through Carry Left)** instruction rotates the contents left through carry by the specified number of bits in count operand.



Example:

RCL AL, 1

Before execution:

CF		AL							
1		0	0	0	0	1	1	1	1

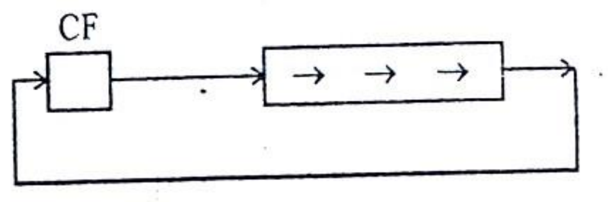
After execution:

CF		AL							
0		0	0	0	1	1	1	1	1

13.RCR

RCR destination, count

This **(Rotate through Carry Right)** instruction rotates the contents right through carry by the specified number of bits in the count operand.



Example:

RCR AL, 1

Before execution:

CF
1

AL							
1	1	0	0	0	0	1	0

After execution:

CF
0

AL							
1	1	1	0	0	0	0	1

<u>String Instructions</u>

REP	Repeat
REPE/REPZ	Repeat while equal/zero
REPNE/REPNZ	Repeat while not equal/not zero
MOVSB/MOVSW	Move byte string/word string
MOVS	Move byte or word string
INS	Input byte or word string
OUTS	Output byte or word string
CMPS	Compare byte or word string
SCAS	Scan byte or word string
LODS	Load byte or word string
STOS	Store byte or word string

1. REP

REP MOVS destination, Source

This **(Repeat)** instruction converts any string primitive instruction into a re-executing loop. It specifies a termination condition which causes the string primitive instruction to continue executing until the termination condition is met. REP is used in conjunction with the MOVS and STOS instructions.

Example:

REP MOVS CL, AL

The other Repeat instructions are:

REPE	- Repeat while Equal
REPZ	- Repeat while zero
REPNE	- Repeat while Not Equal
REPNZ	- Repeat while Not Zero

The above instructions are used with the CMPS and SCAS instructions.

Example:

REPE CMPS destination, source

REPNE SCAS destination

2.MOVS

MOVS destination - string, source-string

This **(Move String)** instruction transfers a byte/word from the source string (addressed by SI) to the destination string (addressed by DI) and updates SI and DI to point to the next string element.

(DEST) ← (SRC)

Example:

MOVS Buffer 1, Buffer 2

3. CMPS

CMPS destination-string, source-string

This **(Compare String)** instruction subtracts the destination byte/word (addressed by DI) from the source byte/word (addressed by SI). It affects the flags but does not affect the operands. -.

Example:

CMPS Buffer 1, Buffer 2

4. SCAS

SCAS destination-string

- This **(Scan String)** instruction subtracts the destination string element (addressed by DI) from the contents of AL or AX and updates the flags.

- ‘The contents of destination string or accumulator are not altered.
- After each operation, DI is updated to point to the next string element

Example:

SCAS Buffer

5. **LODS**

LODS source-string

This **(Load String)** instruction transfers the byte/word string element addressed by SI to register AL or AX and updates SI to point to the next element in the string.

(DEST)←(SRC)

Example:

LODSB name

LODSW name

6. **STOS**

STOS destination - string

This **(Store String)** instruction transfers a byte/word from register AL or AX to the string element addressed by DI and updates DI to point to the next location in the string.

(DEST) ← (SRC)

Example: STOS display

Program Transfer Instructions

Unconditional instructions : CALL, RET, JMP

Conditional instructions : JC, JZ, JA

Iteration control instructions : LOOP, JCXZ

Interrupt instructions : INT, INTO, IRET

1. CALL

- **CALL procedure - name**
- This (CALL) instruction is used to transfer execution to a subprogram or procedure.
- RET (return) instruction is used to go back to the main program.
- There are two basic types of CALL : NEAR and FAR

Intra-Segment CALL:

- A NEAR-CALL is a call to a procedure which is in the same code segment as the CALL instruction.
- When 8086 executes a NEAR-CALL instruction, it decrements the stack pointer (SP) by 2 and copies

the offset of the next instruction after the CALL on the stack.

- It loads IP with the offset of the first instruction of the procedure in same segment.
- This NEAR-CALL is known as Intra-segment CALL.

Inter-Segment CALL:

- A FAR-CALL is a call to a procedure which is in a different segment from that which contains the CALL instruction.
- When 8086 executes a FAR-CALL, it decrements the SP by 2 and copies the contents of the CS register to the stack.
- It then decrements SP by 2 again and copies the offset of the instruction after the CALL to the stack.
- Finally it loads CS with the segment base of the segment which contains the procedure and IP with the offset of the first instruction of the procedure in that segment.
- This FAR-CALL is known as Inter-segment CALL.

Example:

CALL NEAR

CALL AX

2.RET

This **(Return)** instruction will return execution from a procedure to the next instruction after the CALL instruction in the main program.

If intra-segment, IP is popped off the stack; SP =SP+2

If inter-segment, CS is popped off the stack; SP = SP+2

IP is popped off the stack; SP=SP+2

If optional POP value is used, then SP= SP + value.

Example:

RET

RET 6

3. JMP

JMP target

This **(Jump)** instruction unconditionally transfers control to the target location. The target operand may be obtained from the instruction itself (direct JMP) or from memory or a register referenced by the instruction (indirect JMP).

A NEAR-JMP (Intra-segment) is a jump where destination location is in the same code segment. In this case only IP is changed.

IP = IP + signed displacement

A FAR-JMP (Inter-segment) is a jump where destination location is from a different segment. In this case both IP and CS are changes as specified in the destination.

Example:

JMPBX

Conditional JMP

Instruction	**Operation**
JC	Jump if carry
JNC	Jump if no carry
JZ	Jump if Zero
JNZ	Jump if not zero
JS	Jump if sign or negative
JNS	Jump if positive
JP/JPE	Jump if parity/parity even
JNP/JPO	Jump if not parity/odd parity
JO	Jump if overflow
JNO	Jump if no overflow
JA/JNBE	Jump if above/not below or equal
JAE/JNB	Jump if above or equal/not below
JB/JNAE	Jump if below/not above or equal
JBE/JNA	Jump if below or equal / not above
JG/JNLE	Jump if greater/not less than nor equal
JGE/JNL	Jump if greater or equal/not less than
JL/JNGE	Jump if less/neither greater nor equal
JLE/JNG	Jump if less than or equal / not greater

5.LOOP

LOOP label

This (Loop if CX not zero) instruction decrements CX by 1 and transfers control to the target operand if CX is not zero. Otherwise the instruction following LOOP is executed.

If CX≠0, CX = CX - l

IP = IP + displacement

If CX=0, then the next sequential instruction is executed.

Example:

LOOP again

6. LOOPE/LOOPZ

LOOPE/LOOPZ label

These **(LOOP while Equal/Loop while Zero)** are different mnemonics for the same instruction.

If CX≠0, CX=CX-1 and control is transferred '.o the target operand

If CX = 0, then next sequential instruction is executed.

Example:

LOOPE again

7. LOOPNE/LOOPNZ

LOOPNE Label

These **(LOOP while Not Equal/LOOP while Not Zero)** are different mnemonics for the same instruction. CX is decremented by 1 and control is transferred to target operand if CX is not zero and if ZF=0; otherwise the next sequential instruction is executed.

Example:

LOOPNE again

8. JCXZ

JCXZ Label

This **(Jump if CX register Zero)** instruction transfers control to the target operand if CX=0. It is useful at the beginning of a loop to bypass the loop if CX=0.

Example:

JCXZ again

9. INT

INT interrupt type (0-255)

This **(Interrupt)** instruction activates the interrupt procedure specified by the interrupt-type number (0-255). The address of the interrupt pointer is calculated by multiplying the interrupt-type number by 4.

Example :

INT 7, INT 180

10. INTO

This **(Interrupt on Overflow)** instruction generates a software interrupt if the overflow flag is set. Otherwise, control proceeds to the following instruction without activating an interrupt procedure.

11. IRET

This **(Interrupt on Return)** instruction transfers control back to the point of interruption by popping IP, CS and the flags from the stack.

IRET is used to exit any interrupt procedure, whether activated by hardware or software.

Processor Control Instructions

1. HLT

This (Halt) instruction will cause the 8086 to stop fetching and executing instructions. The 8086 will enter a halt state. The ways to get the processor out of halt state are with (i) an interrupt signal on the INTR pin. (ii) An interrupt signal on the NMI pin, (iii) a reset signal on the RESET pin.

2. WAIT

This (Wait) instruction causes the 8086 to enter the wait state while its test line is not active.

3. LOCK

- The LOCK prefix allow 8086 to make sure that another processor does not take control of the system bus while it is in the middle of a critical instruction which uses the system bus.
- The LOCK prefix is put in front of the critical instruction.
- When an instruction with a LOCK prefix executes, the 8086 will assert its bus lock signal output. This signal is connected to an external bus controller device which then prevents any other processor from taking over the system bus.

4. ESC

- ✓ This **(Escape)** instruction provides a mechanism by which other coprocessors may receive their instructions from the 8086 instruction stream and make use of the 8086 addressing modes.
- ✓ The 8086 does a no operation (NOP) for the ESC instruction other than to access a memory operand and place it on the bus.

5. NOP

This **(No operation)** instruction causes the CPU to do nothing. NOP does not affect any flags.

6. Flag operations

Instruction	Operation
CLC	Clear the carry flag (CF)
CMC '	Complement the carry flag (CF)
STC	Set the carry flag (CF)
CLD	Clear the direction flag (DF)

STD	Set the direction flag (DF)
CLI	Clear the interrupt flag (IF)
STI	Set the interrupt flag (IF)

5. <u>Assembler directives</u>

AUQ: Explain the assembler directives in 8086 Microprocessor. (Dec-2006,11,12, May2007,08,10,11,13, Dec 2016)

- An assembler is a program used to convert an assembly language program into the equivalent machine code modules which may further converted into executable codes.

Some directives such as ORG, EQU, DB, DW, etc which are common in different assemblers were also described. Though a representative set of directives for the 8086 assembler is presented, it is possible that some assemblers have a few additional directives. On the other hand, some of the directives presented here may not be present or may be present in different forms.

Directives for Constant and Variable Definition

An Intel 8086 assembly program uses different types of constants like binary, decimal, octal and hexadecimal. These can be represented in the program using different suffixes like B (for binary), D (for decimal), O (for octal) and H (for hexadecimal) to the constant. For example:

- 10H is a hexadecimal number (equivalent to decimal 16).
- 27O is an octal number (equivalent to decimal 23).
- 10100B is a binary number.

A number of directives are used to define and store different kinds of constants.

The DB (Define Byte), DW (Define Word), DDW (Define Double Word) are described in Chapter The 8086 assembler uses DD as directive for double word. Some of the 8086 assemblers also provide the following additional directives.

- ➢ DQ : Define Quadword
- ➢ DT : Define Ten Bytes

In addition, these directives can be used to store strings and arrays. For example:

- NAME DB "NEHA SHIKHA". The ASCII codes of alphanumeric characters specified within double quotes are stored.

- NUM DB 5, 6, 10, 12, 7. The array of five numbers NUM is declared and the numbers 5, 6, 10, 12 and 7 are stored.

The directives DUP and EQU are used to store strings and arrays.

DUP

Using the DUP directive, several locations may be initialized and the values may be put in those locations. The format is as follows. Name Type Num DUP (value)

For example: TEMP DB 20 DUP (5)

The above directive defines an array of 20 bytes in memory and each location is initialized to 5. The array is named TEMP.

EQU

The EQU directive may be used to define a data name with immediate value or another data name. It can also be used to equate a name to a string. For example:

NUMB EQU 20H .

NAME EQU “RASHMI”

Program Location Control Directives

The directives used for program location control in the 8086 assembler are (ORG, EVEN, ALIGN and LABEL.

ORG

The ORG directive is used to set the location counter to a particular value. For example:

- ORG 2375H

  The location counter is set to 2375H. If it occurs in data segment, the next data storage will start at 2375H. If it is in code segment, the next instruction will start at 2375H.

EVEN

Using EVEN directive, the next data item or label is made to start at the even address boundary. The assembler, on encountering EVEN directive, will advance the location counter to even address boundary. For example:

- EVEN TABLE DW 20 DUP (0)

This statement declares an array named TABLE of 20 words starting from the even address. Each word is initialized to zero.

ALIGN number

This directive will force the assembler to align the next segment to an address that is divisible by 2, 4, 8 or 16. The unused bytes are filled with 0 for data and NOP instruction for code. For example:

- ALIGN 2

It will force the next segment to the next even address.

LENGTH

It is an operator which is used to tell the assembler to determine the number of elements in a data item, such as string or array. For example:

- MOV CX, LENGTH ARRAY

This statement will move the length of the array to the CX register.

OFFSET

This operator is used to determine through the assembler, the offset of a data item from the start of the segment containing it. For example:

- MOV AX, OFFSET FACT

This statement will place in the AX register the offset of the variable FACT from the start of the segment.

LABEL

The LABEL directive is used to assign a name to the current value in the location counter. The location counter is used by the assembler to keep track of the current location. The value in the location counter denotes the distance of the current location from the start of the segment.

Segment Declaration Directives

These directives help in declaring various segments with specific names. The start and the end of segments may also be specified using this directive. The directives for segment declaration include SEGMENT, ENDS, ASSUME, GROUP, CODE, DATA, STACK etc.

SEGMENT and ENDS

The SEGMENT and ENDS directives signify the start and end of a segment.

```
INST  SEGMENT
ASSUME CS: FNST, DS: DATAW
__________
__________
```

INST ENDS **ASSUME**

This directive is used to assign logical segment to physical segment at any time. It tells the assembler as to what addresses will be in segment registers at the time of execution. For example:

- ASSUME CS: CODE, DS: DATA, SS: STACK

This directive tells the assembler that the CS register will store the address of the segment whose name is CODE, and so on.

.CODE (Name)

This code directive is the shortcut used in the definition of code segments. The name is optional and is specified if there is more than one code segment in the program.

.DATA and .STACK

Similar to .CODE, the .DATA and .STACK directives are shortcuts in the definition of data segment and stack segment, respectively.

GROUP

This directive is used to tell the assembler to group all the segments in one logical group segment. This allows the contents of all the segments to be accessed from the same segment base. For example:

- PROG GROUP CODE, DATA

The above statement will group the two segments CODE and DATA into one segment named PROG. Each segment must be declared using ASSUME statement as in the following statement.

- ASSUME CS: PROG DS: PROG

Procedure and Macro-related Directives

The directives in this group relate to the declaration of procedures and macros along with the variables contained in them. The directives include PROC, ENDP, PUBLIC, MACRO,ENDM and EXTRN.

PROC and ENDP

The PROC directive is used to define the procedures. The procedure name is a *must* and it must follow the naming convention of the assembler. Along with the name of the procedure, the field NEAR or FAR also needs to be specified.

The ENDP directive is used to mark the end of the procedure. Some examples are given below.

```
FUNCT PROC FAR
_________
_________
ENDP
FACT PROC NEAR
_________
_________
ENDP
```

The first procedure FUNCT is in a segment which is different from where it is called. The second procedure FACT is in the same segment where it is called. All the statements of the procedure are between PROC and ENDP directives.

PUBLIC

It is very much possible that a variable is defined in one module, but is used in other modules. In order to facilitate the linking, such variables are declared *public*, using the PUBLIC directive in the module where they are defined. For example:

➢ PUBLIC PX, PY, PZ

MACRO and ENDM

The MACRO directive is used to define macros in the program. The ENDM directive defines the end of MACRO

Other Directives

These directives are of general nature, or they relate to more than one group described earlier. The directives described include PTR, PAGE, TITLE, NAME and END.

PTR

The PTR is an operator used in instructions to assign a specific type to a variable or a label. The PTR operator can also be used to override the declared type of variable. Following is an example of the use of PTR directive.

➢ ARRAY DW 0125H, 1630H, 9275H ...

In the above array of words, suppose we wish to move a byte from the array, we may then simply insert the PTR operator as follows

➢ MOV AL, BYTE PTR ARRAY

PAGE

This directive is used for listing of the assembled program. At the start of the program, the directive is placed to specify the maximum number of lines on a page and the maximum number of characters in a line to be placed for listing. An example is given below.

- PAGE 60, 120

The above example specifies that 60 lines are to be listed on a page with a maximum of 120 characters in each line.

TITLE

This directive is also used for the listing of the program. The title declared in this directive defines the title of the program and is listed in line 2 of each page of program listing. The maximum number of characters allowed is 60. For example:

- TITLE PROGRAM TO FIND SQUARE ROOT

NAME

The NAME directive is used to assign a specific name to each module, when the program consists of a number of modules. This helps in understanding the program logic.

END

This is the last statement of the program and it specifies the end of the program to the assembler. It must be noted that not all of the above directives are used in all programs. User may deploy them depending on the need of the program logic.

6. ASSEMBLY LANGUAGE PROGRAMS:

1. 16-BIT ADDITION USING 8086

```
     MOV DX,0000
     MOV AX,[2000]
     MOV BX,[2002]
     ADD AX,BX
     JNC L1
     INC DX
L1   MOV[2004],AX
     MOV[2006],DX
     HLT
```

2. 16-BIT SUBTRACTION USING 8086

```
     MOV DX,0000
     MOV AX,[2000]
     MOV BX,[2002]
     SUB AX,BX
     JNC L1
     INC DX
L1   MOV[2004],AX
     MOV[2006],DX
     HLT
```

3. 16 BIT MULTIPLICATION USING 8086

```
      MOV DX,0000
      MOV SI,2000
      MOV AX,SI
      MOV  CX,[2002]
      MUL  CX
      MOV SI,2100
      MOV [SI],AX
      MOV  [2102],DX
      HLT
```

4. 16 BIT DIVISION USING 8086

```
      MOV DX,0000
      MOV SI,2000
      MOV AX,SI
      MOV  CX,[2002]
      DIV  CX
      MOV SI,2100
      MOV [SI],AX
      MOV  [2102],DX
      HLT
```

5. 16 BIT ASCENDING ORDER USING 8086

```
      MOV AX,0000H
START  MOV CX,0005H
      MOV DX,0005H
      MOV SI,2000H
LABEL  MOV AX,[SI]
      CMP AX,[SI+2]
      JC LOOP
      XCHG AX,[SI+2]
      XCHG AX,[SI]
LOOP     ADD SI,0002
      LOOP LABEL
      DEC DX
      JNZ START
      HLT
```

6. 16-BIT DESCENDING ORDER USING 8086

```
      MOV AX,0000
START  MOV CX,0005
      MOV DX,0005
      MOV SI,2000
LABEL MOV AX,[SI]
      CMP AX,[SI+2]
      JNC  LOOP
      XCHG AX, [SI+2]
      XCHG AX, [SI]
LOOP      ADD SI, 0002
      LOOP LABEL
      DEC DX
      JNZ START
      HLT
```

7. LARGEST NUMBER IN AN ARRAY USING 8086

```
        MOV CX,0004
        DEC CX
        MOV SI,2000
        MOV AX,[SI]
LABEL  CMP AX,[SI+2]
        JNC LOOP1
        MOV AX,[SI+2]
LOOP 1   ADD SI,0002
        LOOP LABEL
        MOV[2500],AX
        HLT
```

8. SMALLEST NUMBER IN AN ARRAY USING 8086

```
        MOV CX,0004
        DEC CX
        MOV SI,2000
        MOV AX,[SI]
LABEL CMP AX,[SI+2]
        JC  LOOP1
        MOV AX,[SI+2]
LOOP 1  ADD SI,0002
        LOOP LABEL
        MOV[2500],AX
        HLT
```

9. Write a program based on 8086 instruction set to compute the average of 'n' numberof bytes stored in the memory.(Nov/ Dec 2012)

```
MOV     SI, 2000H
MOV     DI, 3000H
MOV     CL, [SI]
INC     SI
MOV     AX, 0000H
ADD     AL, [SI]
JNC     STEP2
INC     AH
INC     SI
LOOP    STEP1
MOV     [DI]AX
HLT
```

10. Write an 8086 ALP to sort the array of elements in ascending order. (Apr/ May 2011, May / June 2013)

```
        MOV SI, 2000H
        MOV CL,[SI]
        DEC CL
STEP1  MOV SI,2000H
        MOV CH, [SI]
        DEC CH
        INC SI
STEP2  MOV AL, [SI]
        INC SI
        CMP AL, [SI]
        JC STEP3
        XCHG AL, [SI]
        XCHG AL,[SI-1]
        DEC CH
        JNZ STEP2
STEP3  DEC CL
        JNZ STEP1
        HLT
```

11. Write an 8086 ALP to find the largest element in array elements. (Apr/ May 2011)

```
        MOV   SI.2000H
        MOV   DI, 3000H
        MOV   CL, [SI]
        INC   SI
        MOV   AL, [SI]
        DEC   CL
STEP1:  INC   SI
        MOV   BL, [SI]
        CMP   AL, BL
        JNC   STEP2:
        MOV   AL, BL
STEP2:  DEC   CL
        JNZ   STEP1:
        MOV   [DI],AL
        HLT
```

12. Write an 8086 program to convert BCD data to binary data. (**Nov/ Dec 2010)**

```
MOV BX,2000H
MOV AL,[BX]
MOV DL,AL
AND DL,0FH
AND AL,F0H
MOV CL,4
ROA AL,CL
MOV DH,OAH
MUL DH
ADD AL,DL
MOV [BX+1],AL
HLT
```

7. Linking And Relocation

Explain linking and relocation concepts in 8086 Processor.

The DOS linking program links the different object modules of a source program and function library routines to generate an integrated executable code of the source program.

The main input to the linker is the .OBJ file that contains the object modules of the source programs.

The linker program is invoked using the following options.

```
C> LINK
or
C>LINK MS.OBJ
```

The output of the link program is an executable file with the entered filename and .EXE extension. This executable filename can further be entered at the DOS prompt to execute the file.

The linked file in binary for ***run*** on a computer is commonly known as executable file or simply '.exe.' file. After linking, there has to be re-allocation of the sequences of placing the codes before actually placement of the codes in the memory.

The loader program performs the task of reallocating the codes after finding the physical RAM addresses available at a given instant. The DOS linking program links the different object modules of a source program and function library routines to generate an integrated executable code of the source program.

The loader program performs the task of reallocating the codes after finding the physical RAM addresses available at a given instant. The ***loader*** is a part of the operating system and places codes into the memory after reading the '.exe' file.

A program called *locator* reallocates the linked file and creates a file for permanent location of codes in a standard format.

Segment combination

In addition to the linker commands, the assembler provides a means of regulating the way segments in different object modules are organized by the linker.

Segments with same name are joined together by using the modifiers attached to the SEGMENT directives. SEGMENT directive may have the form Segment name SEGMENT Combination-type where the combine-type indicates how the segment is to be located within the load module.

Segments that have different names cannot be combined and segments with the same name but no combine-type will cause a linker error. The possible combine-types are:

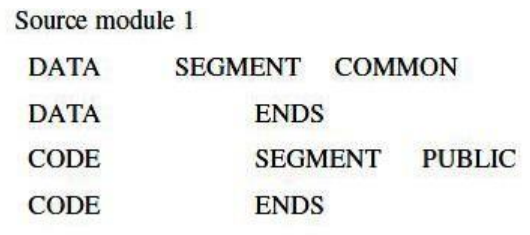
PUBLIC – If the segments in different modules have the same name and combine type PUBLIC, then they are concatenated into a single element in the load module. The ordering in the concatenation is specified by the linker command.

COMMON – If the segments in different object modules have the same name and the combine-type is COMMON, then they are overlaid so that they have the same starting address. The length of the common segment is that of the longest segment being overlaid.

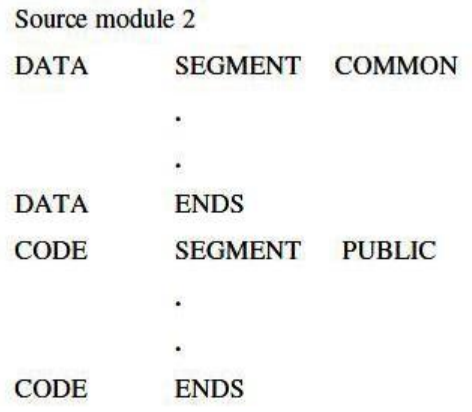
STACK – If segments in different object modules have the same name and the combine type STACK, then they become one segment whose length is the sum of the lengths of the individually specified segments. In effect, they are combined to form one large stack.

AT – The AT combine-type is followed by an expression that evaluates to a constant which is to be the segment address. It allows the user to specify the exact location of the segment in memory.

MEMORY – This combine-type causes the segment to be placed at the last of the load module. If more than one segment with the MEMORY combine-type is being linked, only the first one will be treated as having the MEMORY combine type; the others will be overlaid as if they had COMMON combine-type.



```
Source module 1
 DATA       SEGMENT   COMMON
 DATA             ENDS
 CODE             SEGMENT    PUBLIC
 CODE             ENDS
```



```
Source module 2
DATA        SEGMENT   COMMON
            .
            .
DATA        ENDS
CODE        SEGMENT    PUBLIC
            .
            .
CODE        ENDS
```

Access to External Identifiers

If an identifier is defined in an object module, then it is said to be a *local* (or *internal) identifier* relative to the module. If it is not defined in the module but is defined in one of the other modules being linked, then it is referred to as an *external* (or *global*) *identifier* relative to the module.

In order to permit other object modules to reference some of the identifiers in a given module, the given module must include a list of the identifiers to which it will allow access. Therefore, each module in multi-module programs may contain two lists, one containing the external identifiers that can be referred to by other modules.

Two lists are implemented by the EXTRN and PUBLIC directives, which have the forms:

```
EXTRN Identifier: Type..., Identifier: Type
  and
PUBLIC        Identifier,..., Identifier
```

where the identifiers are the variables and labels being declared or as being available to other modules.

8. Stacks

How stacks are accessed in 8086 processor? Explain briefly.(Dec-2007)

The stack is a block of memory that may be used for temporarily storing the contents of the registers inside the CPU. It is a top-down data structure whose elements are accessed using the stack pointer (SP) which gets decremented by two as we store a data word into the stack and gets incremented by two as we retrieve a data word from the stack back to the CPU register.

The stack is essentially *Last-In-First-Out* (LIFO) data segment. This means that the data which is pushed into the stack last will be on top of stack and will be popped off the stack first.

The stack pointer is a 16-bit register that contains the offset address of the memory location in the stack segment. T Stack Segment register (SS) contains the base address of the stack segment in the memory.

The Stack Segment register (SS) and Stack pointer register (SP) together address the stacktop as explained below:

$\text{SS} \Longrightarrow \text{5000H}$

$\text{SP} \Longrightarrow \text{2050H}$

If the stack top points to a memory location 52050H, it means that the location 52050H is already occupied with the previously pushed data. The next 16 bit push operation will decrement the stack pointer by two, so that it will point to the new stack-top 5204EH and the decremented contents of SP will be 204EH. This location will now be occupied by the recently pushed data.

Thus for a selected value of SS, the maximum value of SP=FFFFH and the segment can have maximum of 64K locations. If the SP starts with an initial value of FFFFH, it will be decremented by two whenever a 16-bit data is pushed onto the stack. After successive push operations, when the stack pointer contains 0000H, any attempt to further push the data to the stack will result in stack overflow.

After a procedure is called using the CALL instruction, the IP is incremented to the next instruction. Then the contents of IP, CS and flag register are pushed automatically to the stack. The control is then transferred to the specified address in the CALL instruction i.e. starting address of the procedure. Then the procedure is executed.

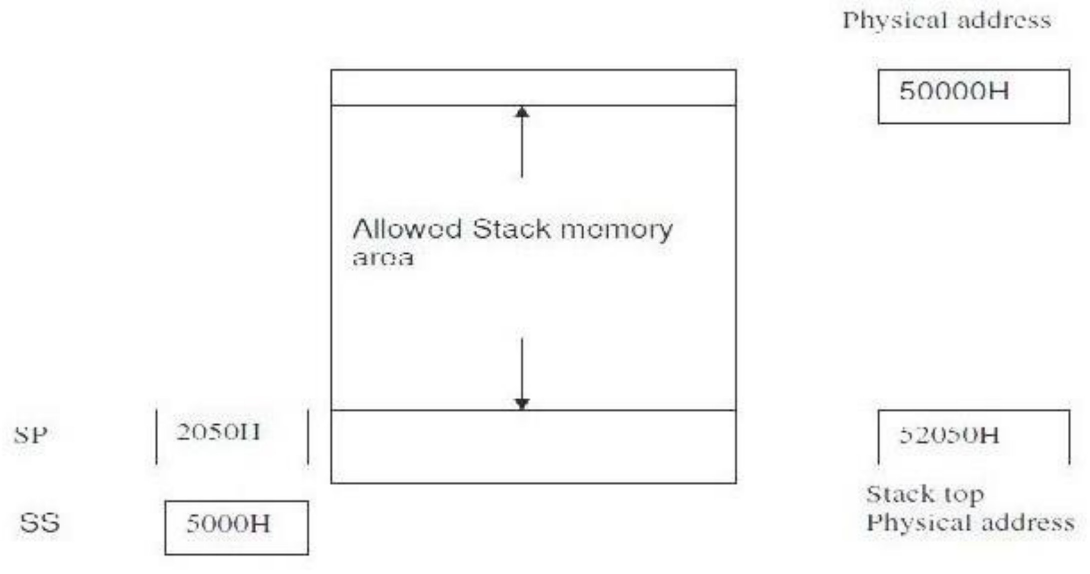


Fig. Stack-top address calculation

9. <u>PROCEDURES & MACROS</u>

<u>Macros:</u> AUQ: Define macro. Explain how macros are constructed in ASM-86 with example. (Dec-2010, May2006,10,11)

Macros look like procedures, but they exist only until our code is compiled, after compilation all macros are replaced with real instructions. If we declared a macro and never used it in out code, complier will simply ignore it.

Macro definition :

name MACRO [parameters,..]

<statements>

ENDM

Example:

My Macro MACRO P1,P2,P3

MOV AX, P1

MOV BX, P2

MOV CX, P3

ENDM

Advantages of macros

- Repeated small groups of instructions replaced by one macro
- Errors in macros are fixed only once, in the definition.
- Duplication of effort is reduced.

- In effect, new higher level instructions can be created
- Programming is made easier, less error prone
- Generally quicker in execute than subroutines.

Disadvantages of macros

In large programs, produce greater code size than procedures.

When to use Macros?

- To replaces small groups of instruction not worthy of subroutines.
- To create a higher instruction set for specific applications.
- To create compatibility with other computers,
- To replace code portions which are repeated often throughout the program.

Procedure (PROC)

This directive marks the start and end of a procedure block called label, the statements in the block can be called with they CALL instruction.

The directive PROC indicated the states of a procedure. The type of the procedure FAR of NEAR is to be specified after the directive, the type NEAR is used to call a procedure with is within the programmed module. The type FAT is used to call a procedure from some other program module. The PROC directive is used with ENDP directive to enclose a procedure.

PROC definition :

```
label  PROC [ [near / far ] ]

<Procedure instructions>

Label ENDP
```

Example:

```
WEST PROC FAR
-
-
-
WEST ENDP
```

A procedure is a set of instructions that compute some value or take some action (such as printing or reading a character value). The definition of a procedure is very similar to the definition of an algorithm.

A procedure is a set of rules to follow which, if they conclude, produce some results. An algorithm is also such a sequence, but an algorithm is guaranteed to terminate whereas a procedure offers no such guarantee.

Nested Procedures

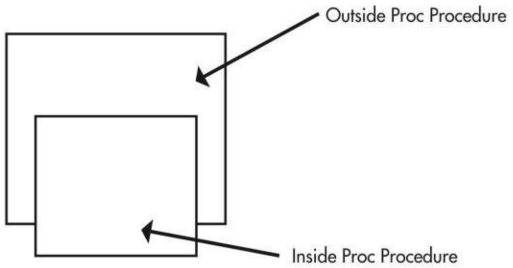
The nest procedure is one procedure definition may be totally enclosed inside another. The following is an example of such a pair of procedures:

```
OUTSIDEPROC       PROC NEAR
JMP   ENDOFOUTSIDE
INSIDEPROC PROC  NEAR
MOC  AX, 0
RET
INSIDEPROC ENDP
ENDOFOUTSIDE:    CALL INSIDEPROC
MOV  BX, 0
RET
OUTSIDEPROC       ENDP
```

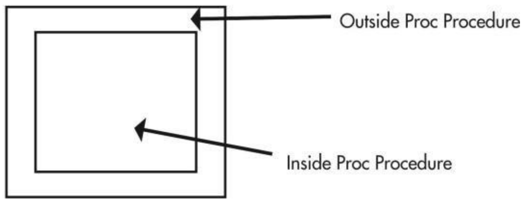
Whenever we nest one procedure within another, it must be totally contained within the nesting procedure. That is the PROC and ENDP statements for the nested procedure must lie between the PROC and ENDP directives of the outside, nesting procedure. The following is not legal.

```
OUTSIDEPROC       PROC NEAR
*
*
*
INSIDEPROC PROC  NEAR
*
*
*
OUTSIDEPROC       ENDF
*
*
*INSIDEPROC       ENDP
```

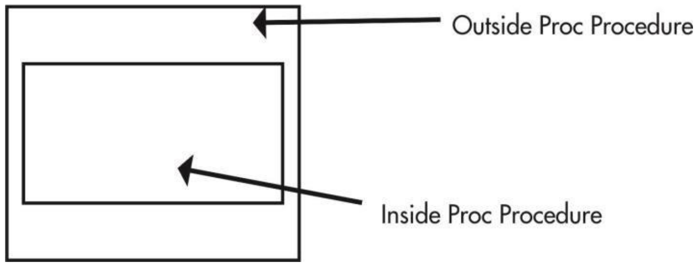
THE OUTISDE proc AND Inside PROC procedures overlap, they are not nested. If we attempt to create a set of procedures like this MASM would report a “block nesting error.



The only form acceptable to MASM



Besides fitting inside an enclosing procedure, PROC/ENDP groups must fit entirely within a segment. The ENDP directive must appear before the SEG ends statement since MyProc begins inside SEG. Therefore, procedures within segments must always take the form.



Not only can we nest procedures inside other procedures and segments, but we can nest segments inside other procedures and segments as well.

Difference between Macros and Procedures

1. To use a procedure use CALL instruction is needed. For example: CALL MyProc.

To use a macro, just type its name. For example, my Macro.

2. Procedure is located at some address in memory, and if use the same procedure 100 times, the CPU will transfer control to this part of the memory. The control will be return back to the program by RET Instruction. The stack is used to keep the return address. The CALL instruction takes about 3 bytes, so the size of the output executable file grows very insignificantly, no matter how many time the procedure is used.

Macro is expanded directly in program's code. So if use the same macro 100 times, the complier expands the macro 100 times, making the output executable file larger and larger, each time all instruction of a macro are inserted.

3. Use stack or any general purpose registers to pass parameters to procedure.

To pass parameters to macro, just type them after the macro name. For example :My Macro ,1.2.3

4. To mark the end of the macro ENDM directive is enough.

To mark the end of the procedure, type the name of the procedure before the ENDP directive.

Differentiate procedures and macros.

The difference between procedures and macros are given in the Table.

S.No.	PROCEDURES	MACROS
1	To use procedure use CALL and RET instructions are needed	To use a macro, just type its name
2	It occupies less memory	It occupies more memory
3	Stack is used	Stack is not used
4	To mark the end of the procedure, type the name of the procedure before the ENDP directive.	To mark the end of the macro ENDM directive is enough
5	Overhead time is required to call the procedure and return to the calling program	No overhead time during the execution.

10. <u>INTERRUPTS AND INTERRUPT SERVICE ROUTINES</u>

AUQ: What are the interrupts in 8086? Explain interrupt related service routines.(Dec-2007,08,12, May-2007,08,11,12,13,15, May 2016, May 2017)

Interrupts:

A signal to the processor to halt its current operation and immediately transfer control to an interrupt service routine is called as interrupt. Interrupts are triggered either by hardware as when the keyboard detects a key press, or by software, as when a program executes the INT instruction.

- Interrupts are triggered by different hardware, these are called hardware interrupts.
- To make a software interrupt there is an INT instruction, it has very simple syntax : INT Value.

Where value can be a number between 0 to 255 (or 00 to FF h).

Interrupt Service Routings (ISRs)

IST is a routing that receives processor control when a specific interrupt occurs.

The 8086 will directly call the service routing for 256 vectored interrupts without any software processing. This is in contrast to non vectored interrupts that transfer control directly to a single interrupt service routine, regardless of the interrupt source.

The 8086 provides a 256 entry interrupt vector table beginning at address 0:0 in memory. This is a 1K table containing 256 4-byte entries. Each entry in the table contains a segmented address that points at the interrupt service routing in memory. Generally, interrupts referred by their index into this table, so interrupt zero's address (vector) is at memory location 0:0 interrupt one's vector is at address 0:4, interrupt two's vector is at address 0:8, etc.

Interrupt vector table:

It is a table maintained by the operating system. It contains addresses (vectors) of current interrupt service routine. When an interrupt occurs, the CPU branches to the address in the table that corresponds to the interrupt's number.

When an interrupt occurs, regardless of source, the 8086 does the following :

1. The CPU pushes the flags register onto the stack.
2. The CPU pushes a far return address (segment: offset) onto the stack, segment value first.
3. The CPU determines the cause of the interrupt (i.e, the interrupt number) and fetches the

four byte interrupt vector from address 0 : vector x 4 (0:0, 0;4, 0:8 etc)

4. The CPU transfers control to the routine specific by the interrupt vector table entry.

After the completion of these steps, the interrupt service routine takes control. When the interrupt service routine wants to return control, it must execute an IRET (interrupt return) instruction. The interrupt return pops the far return address and the flags of the stack.

	Available Interrupt pointers (224)	3FFH	Type 255 pointer
			(Available)
			:
			:
		080H	Type 32 pointer
			(Available)
	Reserved Interrupts (27)	07FH	Type 31 pointer
			(Reserved)
			;
			Type 5 pointer
			(Reserved)
	Dedicated Interrupt Pointers(6)		Type 4 pointer
		014H	overflow

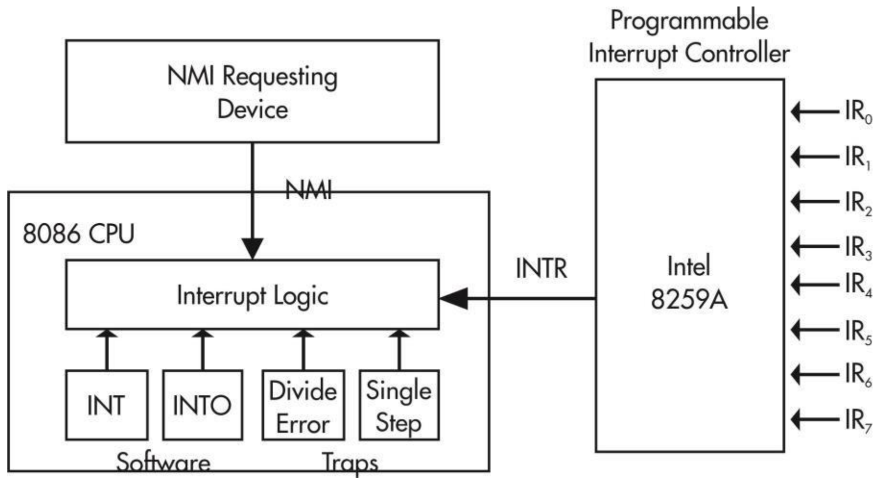
		010h	Type 3 pointer 1Byte INT instruction
		00CH	Type 2 Pointer NonMaskable
		008H	TYPE 1 Pointer Single step
CS Base address		004H	TYPE 0 Pointer Divide by zero
IP offset		000H	

Types of Interrupts :

1. Hardware Interrupt – External used INTR and NMI
2. Software Interrupt – Internal – from INT or INTO
3. Processor Interrupt – Traps and 10 Software Interrupts

External – generated outside the CPU by other hardware, (INTR, NMI)

Internal – generated within CPU as a result of an instruction of operation (INT, INTO, Divide error and single step)



Dedicated Interrupts:

(i) <u>Divide Error Interrupt (Type 0)</u>

This interrupt occurs automatically following the execution of DIV or IDIV instruction when the quotient exceeds the maximum value that the division instruction allows.

(ii) <u>Single Step Interrupt (Type 1)</u>

This interrupt occurs automatically after execution of each instruction when the Trap Flag (TF) is set of 1. It is used to execute programs one instruction at a time, after which an interrupt is requested.

Following the ISR, the next instruction is executed and another single stepping interrupt request occurs.

(iii) Non Maskable Interrupt (Type 2)

It is the highest priority hardware interrupt that triggers on the positive edge. This interrupt occurs automatically when it received a low-to-high transition on its NMI input pin. This interrupt cannot be disabled or masked. It is used to save program data or processor status in case of system power failure.

(iv) Breakpoint Interrupt (Type 3)

This interrupt is used to set break point is software debugging programs.

(v) Overflow Interrupt (Type 4)

This interrupt is initiated by INTO (Interrupt on Overflow) instruction. It is used to check overflow condition after any signed arithmetic operation in the system. The overflow flag (OF) will be set if the signed arithmetic operation generates a result whose size is larger than the size of destination register or memory location. At this time overflow interrupt is used to indicate an error condition.

Software Interrupts (INT n)

The software interrupts are non maskable interrupts. They are higher priority than hardware interrupts.

The software interrupts are called within the program using the instruction INT n. Here 'n' means value and is in the range of 0 to 255. These interrupts are useful for debugging, testing ISRs and calling procedures.

Hardware Interrupts

INTR and NMI are called hardware interrupts. INTR is maskable and NMI is non maskable interrupts.

INTR interrupts (type 0-255) can be used to interrupt a program execution. This interrupt is implemented by using two pins: INTR and INTA. This interrupts can be enabled or disabled by STI (IF=1) or (IF=0) respectively.

Interrupt Priority

The priority of interrupts of 8086 is shown in Table. The software interrupts except single step interrupt have the highest priority; followed by NMI, followed by INTR. Single step interrupt has the least priority. The 8086 checks for internal interrupts before for any hardware interrupt. Therefore software interrupts have higher priority than hardware interrupts.

Interrupt	**Priority**
INT n, INT 0, Divide Error	Highest
NMI	↓
INTR	↓
Single Step	Lowest

UNIT-II
8086 SYSTEM BUS STRUCTURE

8086 signals – Basic configurations – System bus timing –System design using 8086 – IO programming – Introduction to Multiprogramming – System Bus Structure - Multiprocessor configurations – Coprocessor, Closely coupled and loosely Coupled configurations – Introduction to advanced processors.

- ✓ The 8086 Microprocessor is a 16-bit CPU.
- ✓ Available clock rates: 5, 8 and 10MHz
- ✓ Packages: 40 pin CERDIP or plastic package
- ✓ Operates in single processor or multiprocessor configurations
- ✓ Modes of operation: Minimum mode (single processor mode) and Maximum mode (multiprocessor mode) configuration.

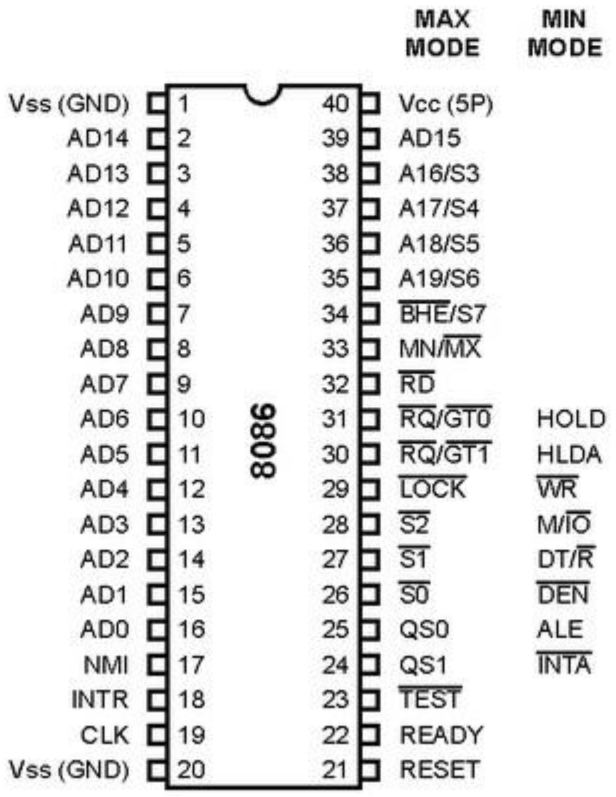
SIGNAL DESCRIPTION OF 8086

AUQ: Explain the signal used in 8086 processor. (Dec 2003,06,07,09,10,13, May 2006,07,08,09,11)

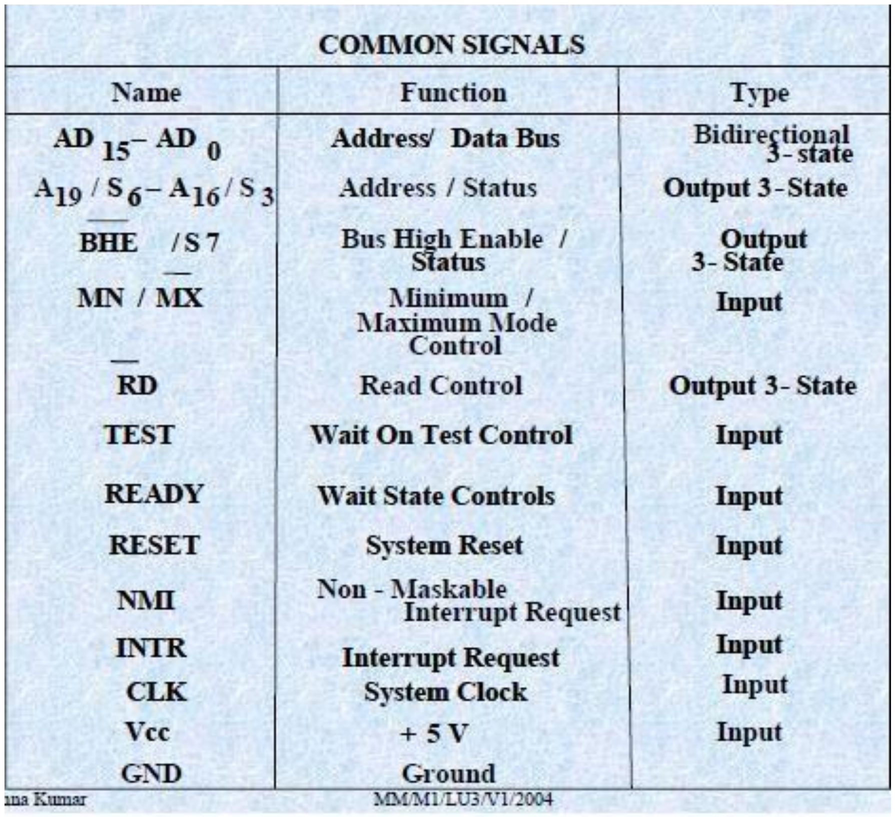
The 8086 signals can be categorized in three groups.

- ➢ Signals having common function in minimum and maximum mode.
- ➢ Signals having special functions in minimum mode
- ➢ Signals having special functions in maximum mode

PIN DIAGRAM



Signals having common function in minimum and maximum mode



COMMON SIGNALS		
Name	Function	Type
$AD_{15} - AD_0$	Address/ Data Bus	Bidirectional 3-state
$A_{19} / S_6 - A_{16} / S_3$	Address / Status	Output 3-State
$\overline{BHE}$ / S_7	Bus High Enable / Status	Output 3- State
MN / $\overline{MX}$	Minimum / Maximum Mode Control	Input
$\overline{RD}$	Read Control	Output 3- State
TEST	Wait On Test Control	Input
READY	Wait State Controls	Input
RESET	System Reset	Input
NMI	Non - Maskable Interrupt Request	Input
INTR	Interrupt Request	Input
CLK	System Clock	Input
Vcc	+ 5 V	Input
GND	Ground	

na Kumar MM/M1/LU3/V1/2004

$AD_{15} – AD_0$

- These are time multiplexed address and data lines. They act as address lines during first part of machine cycle and data lines in later part.

$A_{19} / S_6 – A_{16} / S_3$

- These are time multiplexed address and status lines. They act as address lines during first part of machine cycle and status lines in later part.
- These are most significant address lines for memory operations. During I/O operations these lines are low.
- The status signals S_4 and S_3 indicate which segment registers is being used for memory access.
- The status of interrupt enable flag bit will be displayed on S_5.
- The status line S_6 is always low.

S_4	S_3	Indications
0	0	ES
0	1	SS
1	0	CS
1	1	DS

$\overline{BHE}/S_7$- Bus high enable/ status

- Low signal on $\overline{BHE}$ indicates access to higher order memory banks, otherwise access is to only lower order memory banks.
- $\overline{BHE}$ and A_0 decide the memory bank and type of access.
- S_7 has no function.

$\overline{BHE}$	A_0	Indications
0	0	Both higher and lower order banks for word read/ write
0	1	Higher order bank for byte read/ write
1	0	Lower order bank for byte read/ write
1	1	None

$\overline{RD}$

- Read control signal
- $\overline{RD}$ is low when 8086 is receiving the data from memory or I/O.

READY

- Wait state request signal.
- A HIGH on READY input causes the 8086 to extend the machine cycle by inserting wait states.

$\overline{TEST}$

- This input is examined by WAIT instruction.
- If the $\overline{TEST}$ input goes low, execution will continue, else, the processor remains in idle state.

INTR

- This is level triggered input.
- INTR is sampled during the last clock cycle of each instruction to determine the availability of request.
- These interrupts can be masked internally by resetting the interrupt enable flag.

NMI

- NMI (Non-Maskable interrupt) is positive edge triggered non-maskable interrupt request.

CLK

- CLK is clock signal from external crystal oscillator.
- 8086 requires clock signal with 33% duty cycle.

RESET

- System reset signal must be high for atleast 4 clock periods to cause reset.
- Reset operation takes about 10 clock periods.

V_{CC}

- +5V supply with ±5% tolerance.

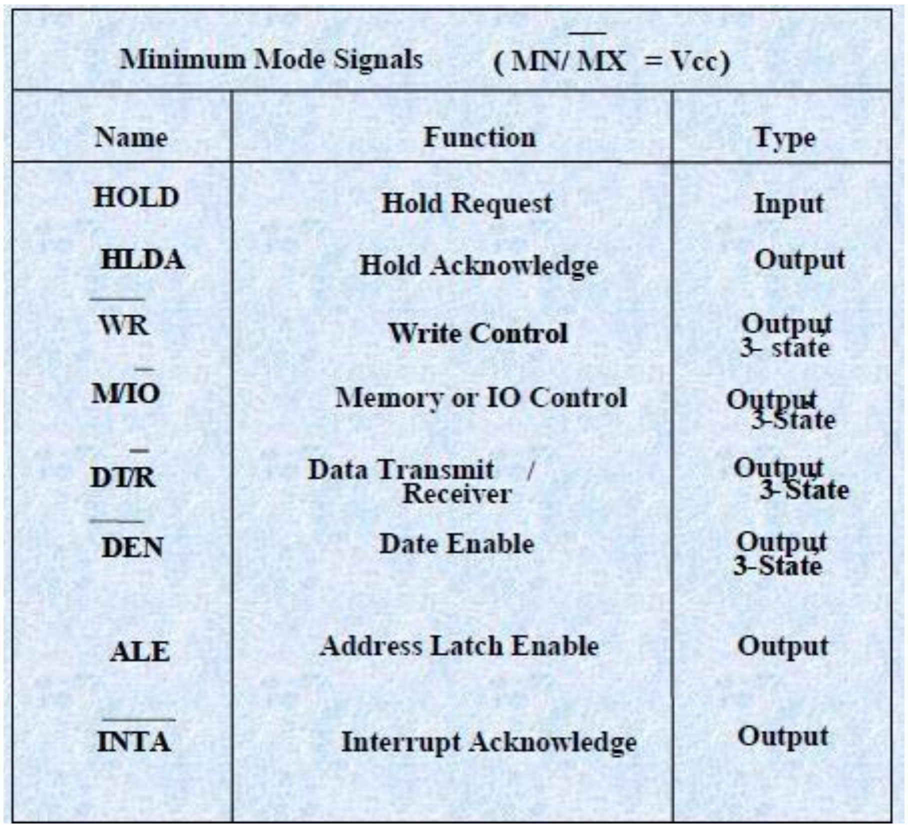
GND

- Ground for internal circuits.

$\mathrm{MN}/\overline{MX}$

- High on this pin selects minimum mode and low signal selects maximum mode.

Signals having special functions in minimum mode



Minimum Mode Signals ($\mathrm{MN}/\overline{\mathrm{MX}}$ = Vcc)		
Name	Function	Type
HOLD	Hold Request	Input
HLDA	Hold Acknowledge	Output
$\overline{\mathrm{WR}}$	Write Control	Output 3- state
$\mathrm{M}/\overline{\mathrm{IO}}$	Memory or IO Control	Output 3-State
$\mathrm{DT}/\overline{\mathrm{R}}$	Data Transmit / Receiver	Output 3-State
$\overline{\mathrm{DEN}}$	Date Enable	Output 3-State
ALE	Address Latch Enable	Output
$\overline{\mathrm{INTA}}$	Interrupt Acknowledge	Output

ALE

- Address latch enable.
- High on this pin indicates valid address on address/data bus.

$\overline{\mathrm{WR}}$

- Write control signal.
- $\overline{WR}$ is low when 8086 sends the data to memory or I/O.

$\mathrm{M}/\overline{\mathrm{IO}}$

- $\mathrm{M}/\overline{\mathrm{IO}}$ = High, indicates memory access.

- $M/\overline{IO}$ = low, indicates I/O access.

$\overline{INTA}$

- $\overline{INTA}$ is the acknowledgement for the interrupt request on INTR pin.
- It is pulsed low in two consecutive bus cycles.
- First pulse indicates interrupt acknowledgement.
- During second pulse, external logic puts the interrupt type on data bus.

$DT/\overline{R}$

- Data transmit/receive.
- This signal, when high indicates data is being transmitted by 8086.
- The low signal indicates that 8086 is receiving the data.

$\overline{DEN}$

- Data bus enable.
- This signal, when low indicates that the 8086 processors address/data bus is used as data bus.
- It is used to enable data buffers.

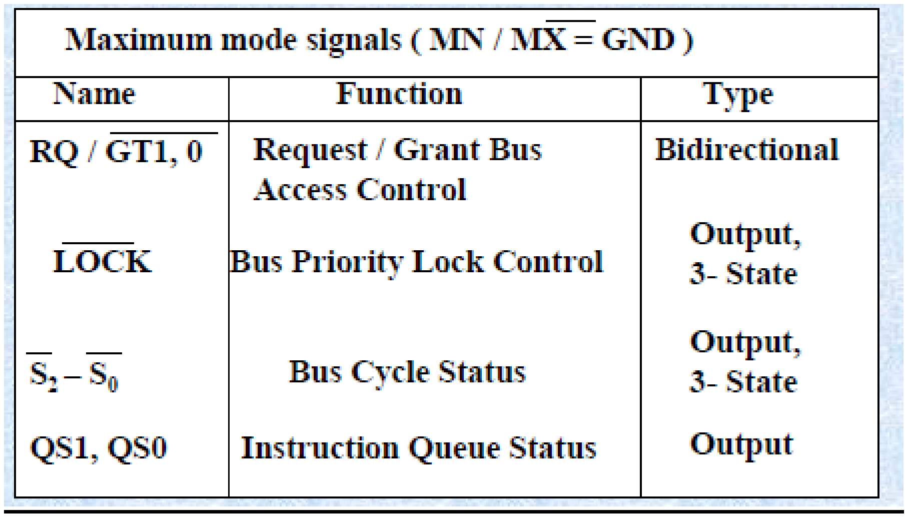
HOLD

- HOLD signal when high indicates another master has requested for direct memory access.
- When HOLD becomes low, it indicates that direct memory access is no more required.

HLDA

- The microprocessor sends high signal on HLDA to indicate acknowledgement of DMA request. It then tristates the buses and control.
- When HOLD becomes low, the microprocessor makes HLDA low and regains the control of buses.

<u>Signals having special functions in maximum mode</u>



Maximum mode signals ($MN / \overline{MX}$ = GND)

Name	Function	Type
$RQ / \overline{GT1, 0}$	Request / Grant Bus Access Control	Bidirectional
$\overline{LOCK}$	Bus Priority Lock Control	Output, 3- State
$\overline{S_2} – \overline{S_0}$	Bus Cycle Status	Output, 3- State
QS1, QS0	Instruction Queue Status	Output

$\overline{LOCK}$

- This signal indicates that an instruction with lock prefix is being executed and the bus is not to be used by any other processor.

$\overline{RQ}/\overline{GT}_1$, $\overline{RQ}/\overline{GT}_0$

- In maximum mode DMA request is received and acknowledged using these signals.

  $\overline{RQ}/\overline{GT}_0$has highest priority compared to $\overline{RQ}/\overline{GT}_1$

$\bar{S}_2$,$\bar{S}_1$, $\bar{S}_0$

- These are the status lines which indicate the type of operation being carried out by the processor.

$\bar{S}_2$	$\bar{S}_1$	$\bar{S}_0$	**Control functions**
0	0	0	Interrupt acknowledge
0	0	1	I/O read
0	1	0	I/O write
0	1	1	Halt
1	0	0	Opcode fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	No operation

$\overline{QS}_1$,$\overline{QS}_0$

- These two signals are decoded to provide instruction queue status.

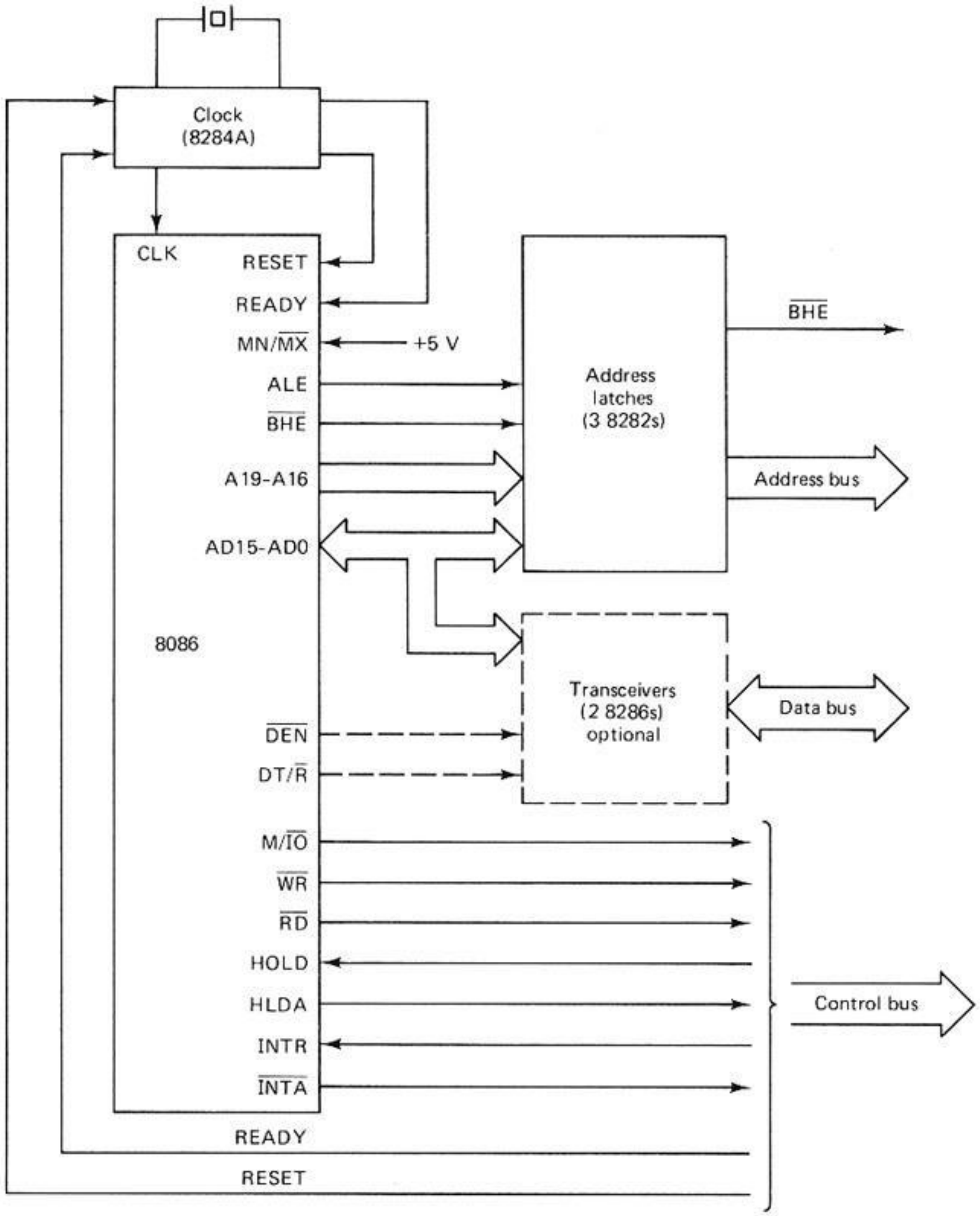
$\overline{QS}_1$	$\overline{QS}_0$	**Indications**
0	0	Queue is in idle state
0	1	First byte of opcode has entered queue
1	0	Queue empty
1	1	Subsequent byte of opcode has entered queue

<u>Basic configurations :</u>

1. Minimum Mode configuration:

AUQ: Explain with neat diagram minimum mode configuration of 8086 system. (Dec 2006,08, May 2006,07)

- ✓ A processor is in minimum mode when its MN / /MX pin is strapped to +5V. In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/MX pin to logic 1.
- ✓ In this mode, all the control signals are given out by the microprocessor chip itself.
- ✓ There is a single microprocessor in the minimum mode system.
- ✓ The remaining components in the system are latches, transreceivers, clock generator, memory and I/O devices. Some type of chip selection logic may be required for selecting memory or I/O devices, depending upon the address map of the system.
- ✓ Latches are generally buffered output D-type flip-flops like 74LS373 or 8282.
- ✓ They are used for separating the valid address from the multiplexed address/data signals and are controlled by the ALE signal generated by 8086.
- ✓ Transreceivers are the bidirectional buffers and sometimes they are called as data amplifiers. They are required to separate the valid data from the time multiplexed address/data signals.
- ✓ They are controlled by two signals namely, DEN and DT/R.
- ✓ The DEN signal indicates the direction of data, i.e. from or to the processor. The system contains memory for the monitor and users program storage.
- ✓ The clock generator generates the clock from the crystal oscillator and then shapes it and divides to make it more precise so that it can be used as an accurate timing reference for the system.
- ✓ The clock generator also synchronizes some external signal with the system clock. The general system organisation is as shown in below fig.



Note: In an 8088 system $\overline{BHE}$ is $\overline{SSO}$, M/$\overline{IO}$ is IO/$\overline{M}$, and only one 8286 is needed.

Figure 8-4 Minimum mode system.

2. Maximum Mode configuration:

AUQ: Explain with neat diagram maximum mode configuration of 8086 system. (Dec 2007)

A processor is in maximum mode when its MN / /MX pin is grounded. The maximum mode definitions of pins 24 through 31 are given in table and a typical maximum mode configuration is shown in Fig.

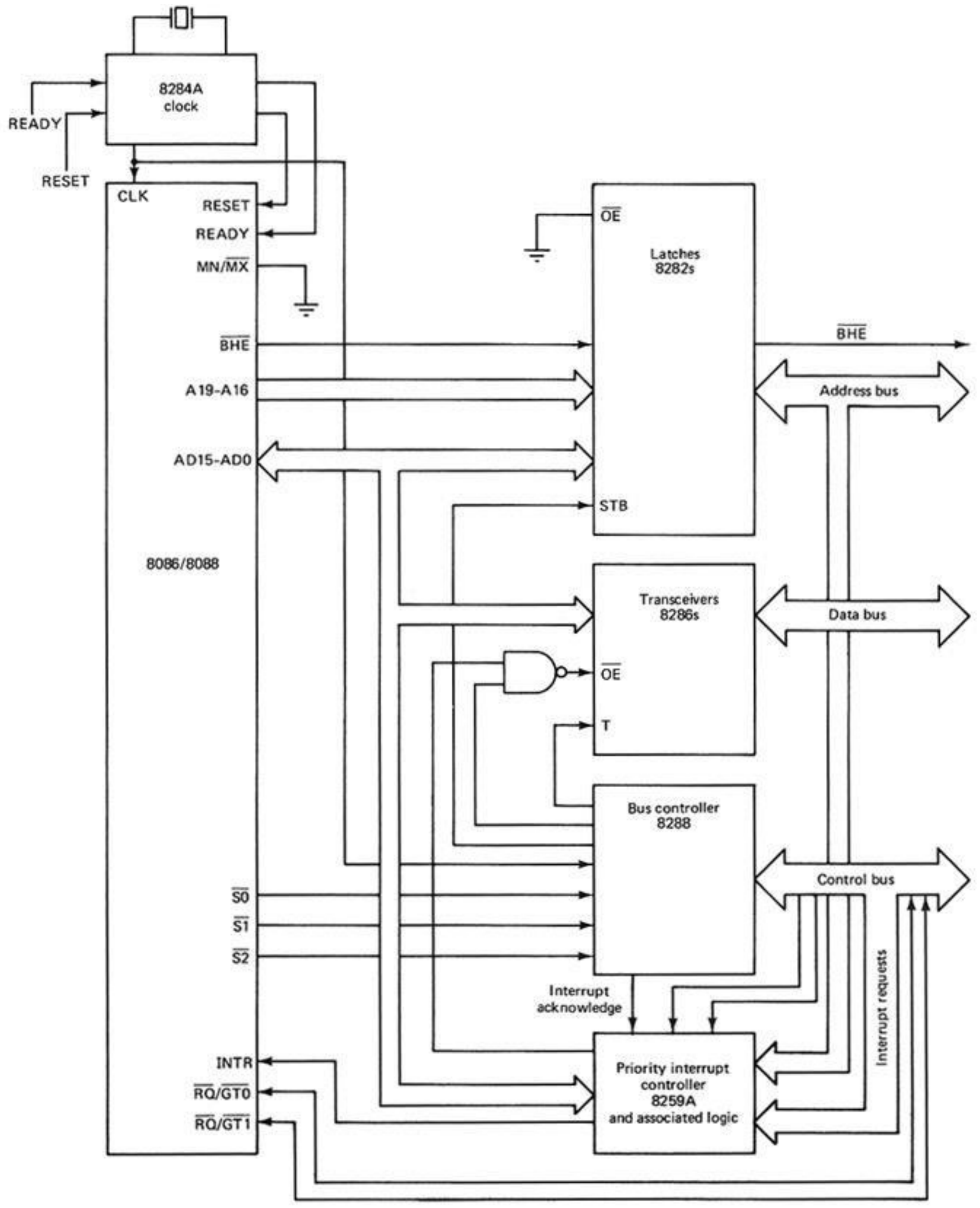
The circuitry is for converting the status bits /S0, /S1 and /S2 into the I/O and memory transfer signals needed to direct data transfers and for controlling the 8282 latches and 8286 transceivers.

It is normally implemented with an Intel 8288 bus controller. Also included in the system is an interrupt priority management device: however, its presence is optional.

- ✓ In the maximum mode, there may be more than one microprocessor in the system configuration. The components in the system are same as in the minimum mode system.
- ✓ The basic function of the bus controller chip IC8288, is to derive control signals like RD and WR (for memory and I/O devices), DEN, DT/R, ALE etc. using the information by the processor on the status lines.
- ✓ The bus controller chip has input lines S2, S1, S0 and CLK. These inputs to 8288 are driven by CPU. The process to be activated for this combination is listed below.
- ✓ It derives the outputs ALE, DEN, DT/R, MRDC, MWTC,AMWC, IORC, IOWC and AIOWC. The AEN, IOB and CEN pins are specially useful for multiprocessor systems.
- ✓ AEN and IOB are generally grounded. CEN pin is usually tied to +5V. The significance of the MCE/PDEN output depends upon the status of the IOB pin.
- ✓ If IOB is grounded, it acts as master cascade enable to control cascade 8259A, else it acts as peripheral data enable used in the multiple bus configurations.

The HOLD and HLDA pins become the /RQ / /GT0 and /RQ / /GT1 pins. Both bus requests and bus grants can be given through each of these pins. They are exactly the same except that if requests are seen on both pins at the same time, then one on /RQ / /GT0 is given higher priority. A request consists of a negative puls arriving before the start of the current bus cycle. The grant is negative puls that is issued at the beginning of the current bus cycle provided that:

1. The previous bus transfer was not the low byte of a word to or from an odd address if the CPU is an 8086. For 8088, regardless of the address alignment the grant signal will not be sent until second byte of a word reference is accessed.
2. The first pulse of an interrupt acknowledgement did not occure during the previous bus cycle.
3. An instruction with a LOCK prefix is not being executed.
4. If condition 1 or 2 is not met, then the grant will not be given until the next bus cycle and if condition 3 is not met, the grant will wait until the locked instruction is completed. In response to the grant the three-state pins are put in their high-impedance state and the next bus cycle will be given to the requesting master.



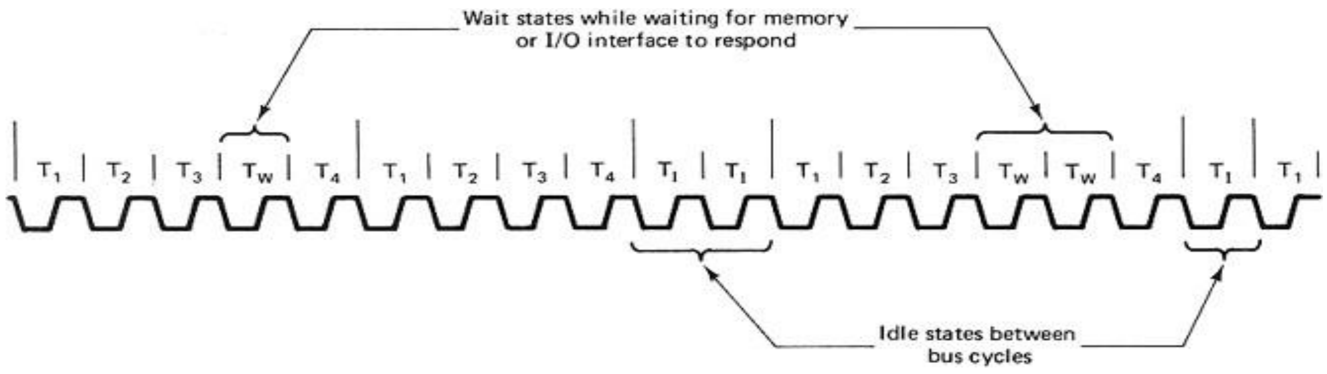
Note: $\overline{BHE}$ is not present in an 8088 system

Figure 8-9 Typical maximum mode configuration.

Read Write Timing Diagram

AUQ: Draw and explain the timing diagram of different cycle in 8086 processor. (Dec 2007, May 2009,13, Dec 2016, May 2017)

The typical sequence of bus cycles is shown below;


Wait states while waiting for memory or I/O interface to respond
T1 T2 T3 TW T4 T1 T2 T3 T4 T1 T1 T1 T2 T3 TW TW T4 T1 T1
Idle states between bus cycles

Bus timing for Minimum Mode:

- ✓ The opcode fetch and read cycles are similar. Hence the timing diagram can be categorized in two parts, the first is the timing diagram for read cycle and the second is the timing diagram for write cycle.
- ✓ The read cycle begins in T1 with the assertion of address latch enable (ALE) signal and also M / IO signal. During the negative going edge of this signal, the valid address is latched on the local bus.
- ✓ The BHE and A0 signals address low, high or both bytes. From T1 to T4 , the M/IO signal indicates a memory or I/O operation.• At T2, the address is removed from the local bus and is sent to the output. The bus is then tristated. The read (RD) control signal is also activated in T2.
- ✓ The read (RD) signal causes the address device to enable its data bus drivers. After RD goes low, the valid data is available on the data bus.
- ✓ The addressed device will drive the READY line high. When the processor returns the read signal to high level,the addressed device will again tristate its bus drivers.

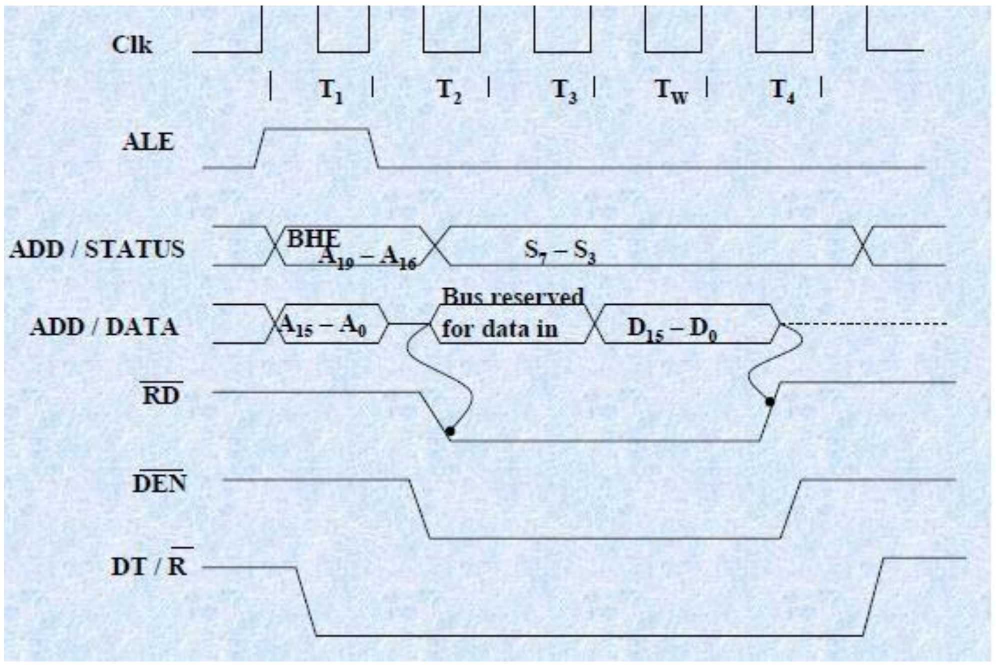


Figure:Readcycle timing diagarm of minimum mode8086

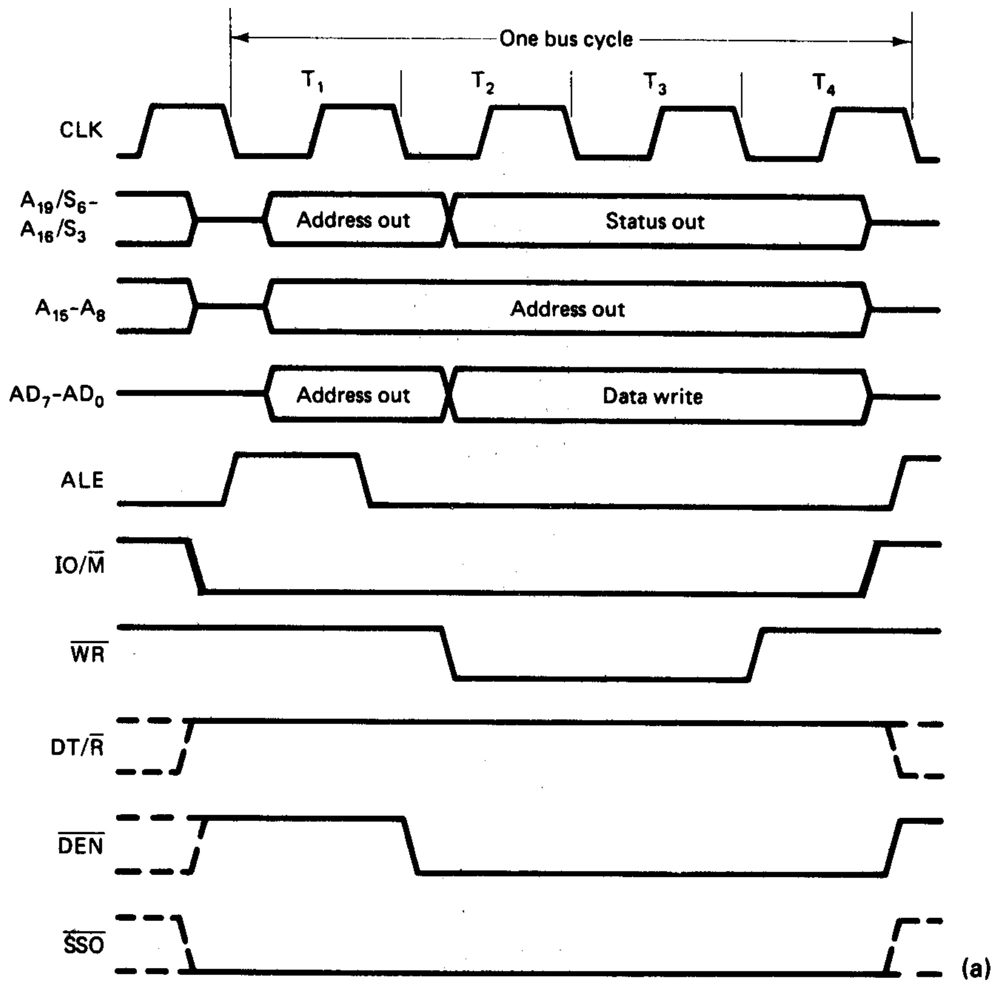
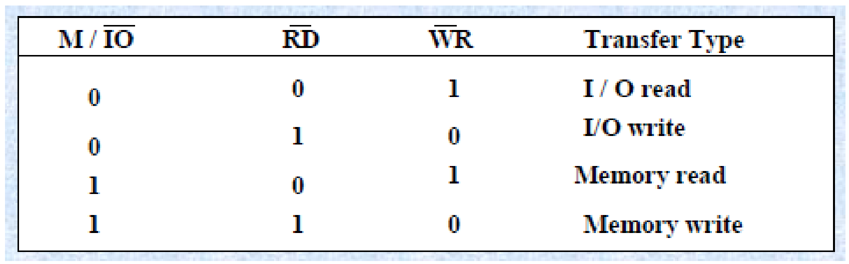


Figure: Write cycle timing diagarm of minimum mode8086

- A write cycle also begins with the assertion of ALE and the emission of the address. The M/IO signal is again asserted to indicate a memory or I/O operation. In T2, after sending the address in T1, the processor sends the data to be written to the addressed location.
- The data remains on the bus until middle of T4 state. The WR becomes active at the beginning of T2 (unlike RD is somewhat delayed in T2 to provide time for floating).
- The BHE and A0 signals are used to select the proper byte or bytes of memory or I/O word to be read or write.
- The M/IO, RD and WR signals indicate the type of data transfer as specified in table below.



M / $\overline{IO}$	$\overline{RD}$	$\overline{WR}$	Transfer Type
0	0	1	I / O read
0	1	0	I/O write
1	0	1	Memory read
1	1	0	Memory write

Bus Timing for Maximum Mode:

- ✓ The maximum mode system timing diagrams are divided in two portions as read (input) and write (output) timing diagrams.
- ✓ The address/data and address/status timings are similar to the minimum mode.
- ✓ ALE is asserted in T1, just like minimum mode. The only difference lies in the status signal used and the available control and advanced command signals.
- ✓ Here the only difference between in timing diagram between minimum mode and maximum mode is the status signals used and the available control and advanced command signals.
- ✓ R0, S1, S2 are set at the beginning of bus cycle.8288 bus controller will output a pulse as on the ALE and apply a required signal to its DT / R pin during T1.
- ✓ In T2, 8288 will set DEN=1 thus enabling transceivers, and for an input it will activate MRDC or IORC. These signals are activated until T4. For an output, the AMWC or AIOWC is activated from T2 to T4 and MWTC or IOWC is activated from T3 to T4.
- ✓ The status bit S0 to S2 remains active until T3 and become passive during T3 and T4**.**
- ✓ If reader input is not activated before T3, wait state will be inserted between T3 and T4.

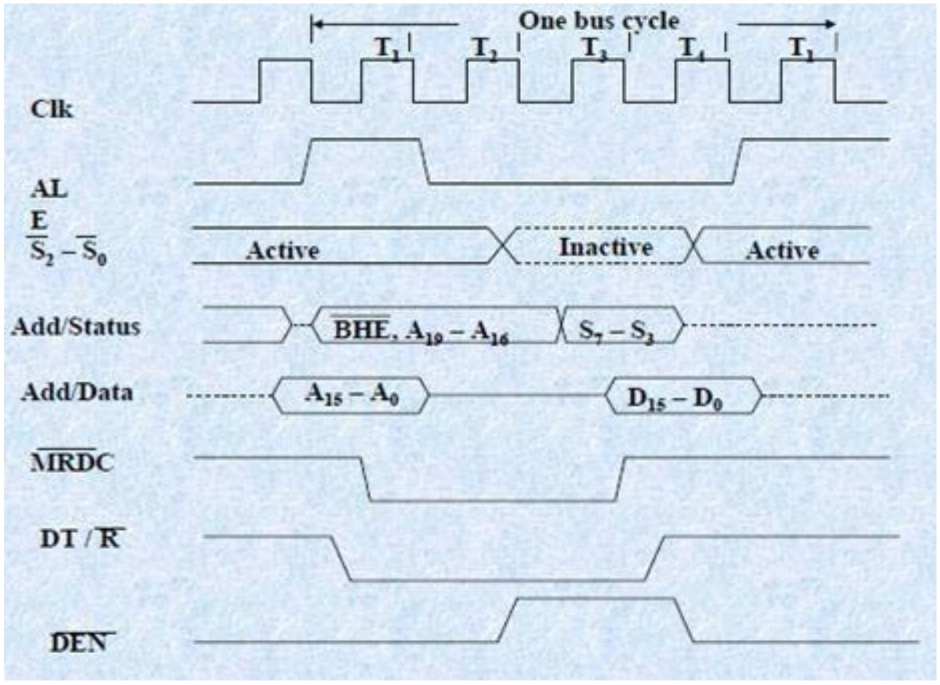


Figure:Memory read timing diagram in maximum mode

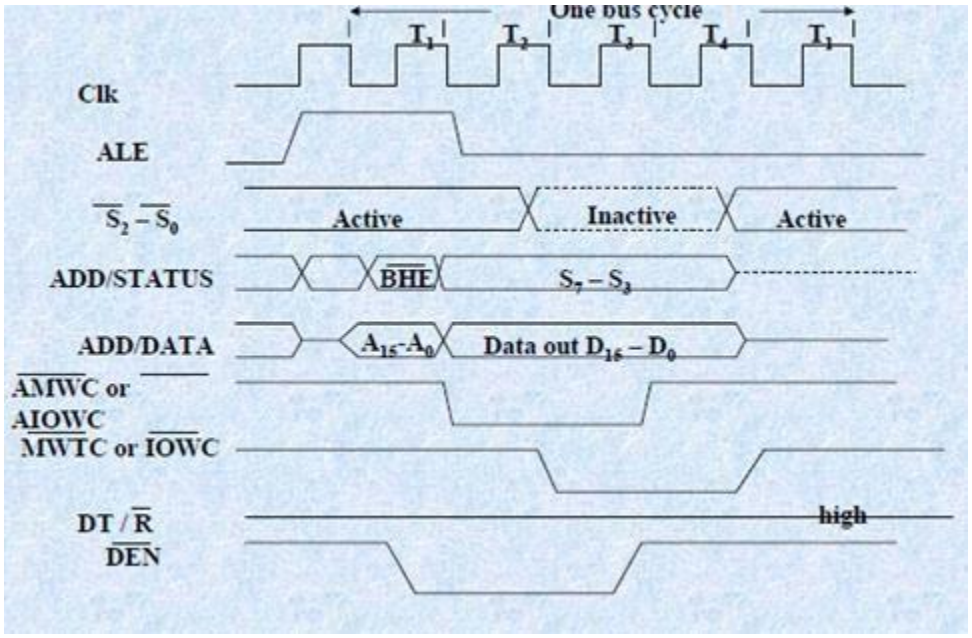


Figure: Memory write timing diagram in maximum mode

<u>IO programming</u>

With example explain the input output program concepts in 8086.

On the 8086, all programmed communications with the I/O ports is done by the IN and OUT instructions defined in Fig. 6-2.

✓ IN and OUT instructions

Name	Mnemonic and Format	Description
Input		
Long form, byte	IN AL, PORT	(AL) <- (PORT)
Long form, word	IN AX, PORT	(AX) <- (PORT+1: PORT)
Short form, byte	IN AL, DX	(AL) <- ((DX))
Short form, word	IN AX, DX	(AX) <- ((DX) + 1: (DX))
Output		
Long form, byte	OUT PORT, AL	(PORT) <- (AL)
Long form, word	OUT PORT, AX	(PORT+1: PORT) <- (AX)
Short form, byte	OUT DX, AL	((DX)) <- (AL)
Short form, word	OUT DX, AX	((DX)+1: (DX)) <- (AX)

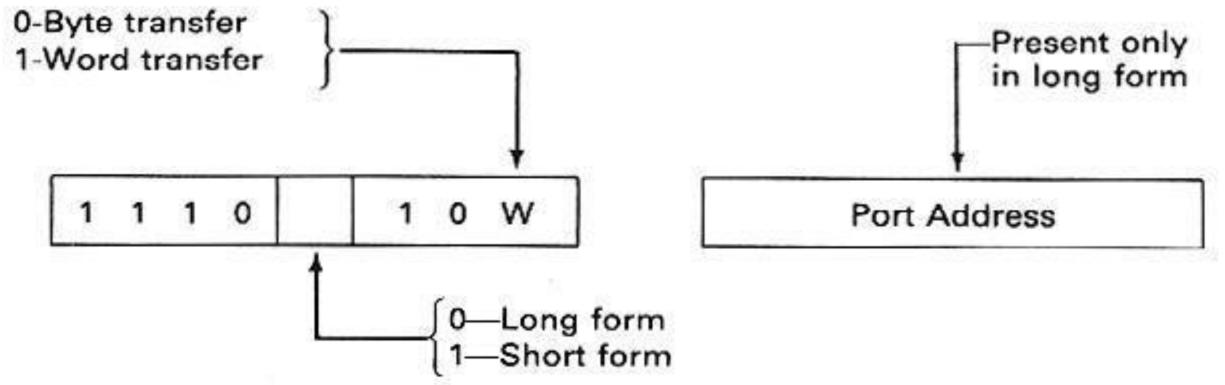
Note: PORT is a constant ranging from 0 to 255

Flags: No flags are affected

Addressing modes: Operands are limited as indicated above.

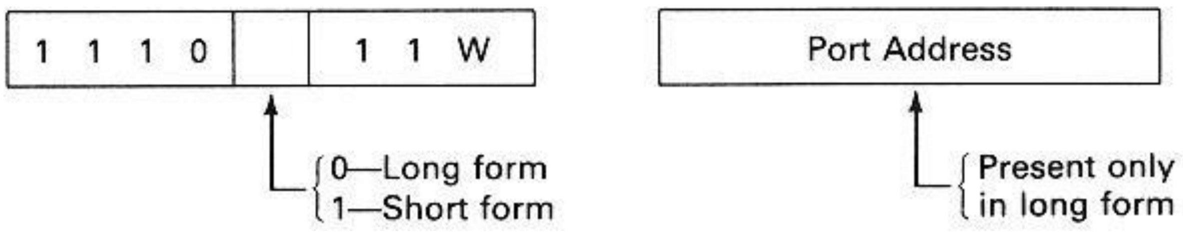
If the second operand is DX, then there is only one byte in the instruction and the contents of DX are used as the port address.

Unlike memory addressing, the contents of DX are not modified by any segment register. This allows variable access to I/O ports in the range 0 to 64K. The machine language code for the IN instruction is:



Although AL or AX is implied as the first operand in an IN instruction, either AL or AX must be specified so that the assembler can determine the W-bit.

Similar comments apply to the OUT instruction except that for it the port address is the destination and is therefore indicated by the first operand, and the second operand is either AL or AX. Its machine code is:



Note that if the long form of the IN or OUT instruction is used the port address must be in the range 0000 to 00FF, but for the short form it can be any address in the range 0000 to FFFF (i.e. any address in the I/O address space). Neither IN nor OUT affects the flags.

The IN instruction may be used to input data from a data buffer register or the status from a status register. The instructions

```
IN AX, 28H
MOV DATA_WORD, AX
```

would move the word in the ports whose address are 0028 and 0029 to the memory location DATA_WORD.

PROGRAMMED I/O

Programmed I/O consists of continually examining the status of an interface and performing an I/O operation with the interface when its status indicates that it has data to be input or its data-out buffer register is ready to receive data from the CPU.

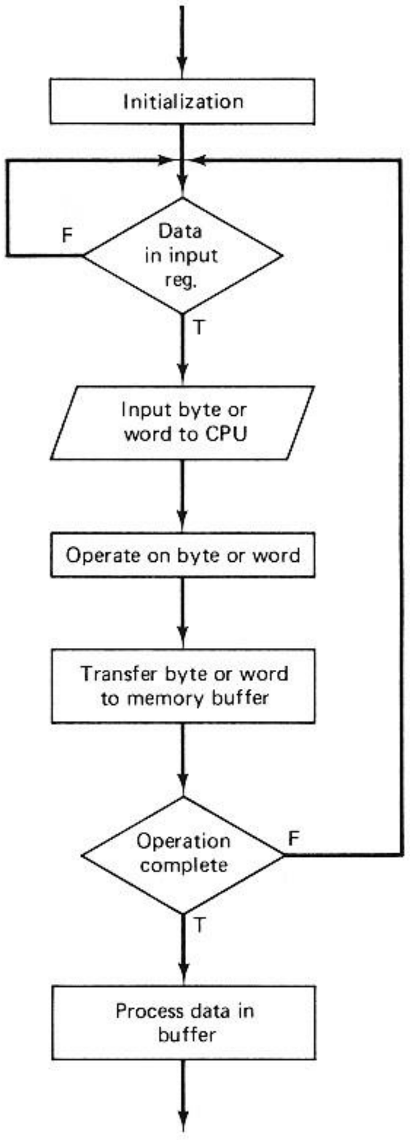


Figure 6-4 Programmed input.

As a more complete example, suppose a line of characters is to be input from a terminal to an 82-byte array beginning at BUFFER until a carriage return is encountered or more then 80 characters are input. If a carriage return is not found in the first 81 characters then the message "BUFFER OVERFLOW" is to be output to the terminal; otherwise, a line feed is to be automatically appended to the carriage return.

Because the ASCII code is a 7-bit code, the eighth bit, bit 7, is often used as parity bit during the transmission from the terminal. Let us assume that bit 7 is set according to even parity and if an odd parity byte is detected, a branch is to be made to ERROR. If there is no parity error, bit 7 is to be cleared before the byte is transferred to the memory buffer.

INTERRUPT I/O

Even though programmed I/O is conceptually simple, it can waste a considerable amount of time while waiting for ready bits to become active. In the above example, if the person typing on the terminal could type 10 characters per second and only 10 μs is required for the computer to input each character, then approximately

$$\frac{99{,}990}{100{,}000} \times 100\% = 99.99\%$$

of the time is not being utilized.

Before an 8086 interrupt sequence can begin, the currently executing instruction must be completed unless the current instruction is a HLT or WAIT instruction.

For a prefixed instruction, because the prefix is considered as part of the instruction, the interrupt request is not recognized between the prefix and the instruction.

In the case of the REP instruction, the interrupt request is recognized after the primitive operation following the REP is completed, and the return address is the location of the REP prefix.

For MOV and POP instructions in which the destination is a segment register, an interrupt request is not recognized until after the instruction following the MOV or POP instruction is executed.

For the 8086, once the interrupt request has been recognized, the interrupt sequence consists of:

1. Establishing a type N.
2. Pushing the current contents of the PSW, CS and IP onto the stack (in that order).
3. Clearing the IF and TF flags.
4. Putting the contents of the memory location 4*N into the IP and the contents of 4*N+2 into the CS.

Thus, an interrupt causes the normal program sequence to be suspended and a branch to be made to the location indicated by the double word beginning at four times the type (i.e. the *interrupt pointer*). Control can be returned to the point at which the interrupt occurred by placing an IRET instruction at the end of the *interrupt routine*.

It was mentioned that there are two classes of interrupts, internal and external interrupts, with external interrupts being caused by a signal being sent to the CPU through one of its pins, which for the 8086 is either the NMI pin or the INTR pin.

An interrupt initiated by a signal on the NMI pin is called a *nonmaskable interrupt* and will cause a type 2 interrupt regardless of the setting of the IF flag. Nonmaskable interrupt signals are normally caused by circuits for detecting catastrophic events.

An interrupt on the INTR pin is masked by the IF flag so that this flag is 0 the interrupt is not recognized until IF returns to 1.

When IF=1 and a maskable external interrupt occures, the CPU will return an acknowledgment signal to the device interface through its /INTA pin and initiate the interrupt sequence.

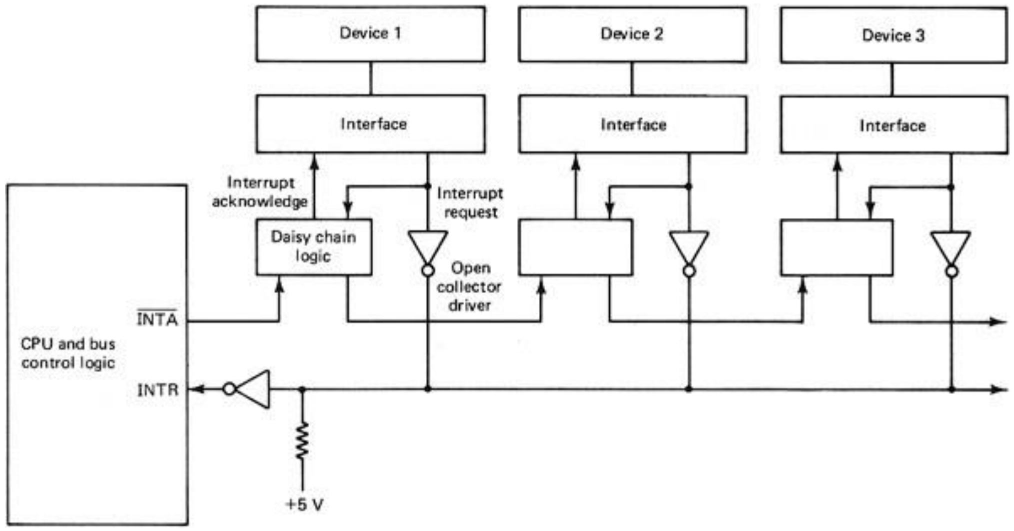
The acknowledgment signal will cause the interface that sent the interrupt signal to send to the CPU (over the data bus) the byte which specifies the type and hence the address of the interrupt pointer. The pointer, in turn, supplies the beginning address of the interrupt routine.

There are several ways of combining with interrupt I/O, some involving only software, some only hardware, and some a combination of the two. Let us consider the following means of giving priority to an interrupt system:

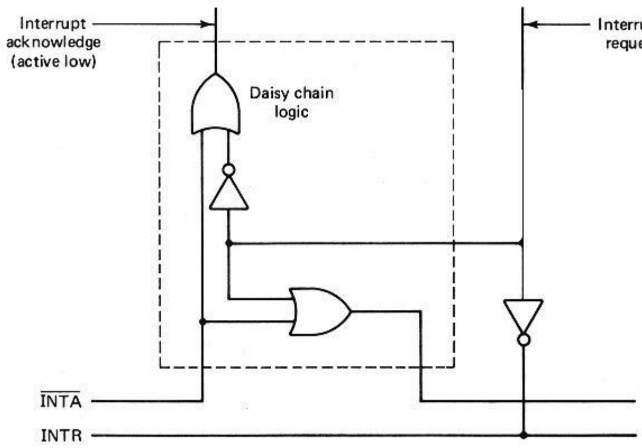
1. Polling
2. Daisy chaining
3. Interrupt priority management hardware

By putting a program sequence (similar to the one in Fig.6-7) at the beginning of the interrupt routine, the priority of the interfaces could be established by the order in which they are polled by the sequence.

Daisy chaining is a simple hardware means of attaining a priority scheme. It consists of associating a logic circuit with each interface and passing the interrupt acknowledge signal through these circuits as shown in Fig.(a). The details of daisy chain logic are shown in Fig.6-14(b). The priority of an interface is determined by its position on the daisy chain. The closer it is to the CPU the higher its priority.



(a) Daisy chain



(b) Logic

Figure 6-14 Daisy chain arrangement.

BLOCK TRANSFERS AND DMA

The activity involved in transferring a byte or word over the system bus is called a *bus cycle*. The execution of an instruction may require more than one bus cycle. For example the instruction:

MOV AL, TOTAL

would use a bus cycle to bring in the contents of TOTAL in addition to the cycle needed to fetch the instruction.

During any given bus cycle one of the system components connected to the system bus is given control of the bus. This component is said to be the *master* during that cycle and the component it is communicating with is said to be the *slave*.

The 8086 receives bus requests through its HOLD pin and issues grants from its hold acknowledge (HLDA) pin. A request is made when a potential master sends a 1 to the HOLD pin. Normally, after the current bus cycle is complete the 8086 will respond by putting a 1 on the HLDA pin.

During a block input byte transfer the following sequence occurs as the datum is sent from the interface to the memory:

1. The interface sends the controller a request for DMA service
2. The controller gains control of the bus
3. The contents of the address register are put on the address bus
4. The controller sends the interface a DMA acknowledgment which tells the interface to put data on the data bus (For an output it signals the interface to latch the next data placed on the bus)
5. The data byte is transferred to the memory location indicated by the address bus
6. The controller relinquishes the bus
7. The address register is incremented by 1
8. The byte count register is decremented by 1
9. If the byte count register is nonzero return to step 1; otherwise stop

The controller/interface design shows bidirectional address lines connected to the controller and only unidirectional address lines going to the interface.

Multiprocessor Systems

Explain the different configurations of multiprocessor systems. (May 2008)

Multiprocessor Systems refer to the use of multiple processors that execute instructions simultaneously and communicate using mailboxes and semaphores Maximum mode of 8086 is designed to implement 3 basic multiprocessor configurations:

1. Coprocessor (8087)
2. Closely coupled (dedicated I/O processor: 8089)
3. Loosely coupled (Multi bus)

Coprocessors and closely coupled configurations are similar - both the CPU and the external processor share:

- ✓ Memory
- ✓ I/O system
- ✓ Bus & bus control logic
- ✓ Clock generator

Multiprocessor configuration

Discuss about the multiprocessor system of 8086.

Explain multiprocessor system. (June 2016, Dec 2016)

Introduction:

Multiprocessor Systems refer to the use of multiple processors that execute instructions simultaneously and communicate using mailboxes and semaphores.

Maximum mode of 8086 is designed to implement 3 basic multiprocessor configurations:
1. Coprocessor (8087)
2. Closely coupled (8089)
3. Loosely coupled (Multibus)

Need for Multiprocessor system

1. Due to limited data width and lack of floating point arithmetic instructions, 8086 requires many instructions for computing even single floating point operations. For this Numeric data processor 8087 is used.
2. Some processor like DMA processor can take care of low level operations , while the 8086 CPU execute high level operations.

Advantages of Multiprocessor

- Easy to add more processor for expansion as per requirement
- When failure occurs, it is easier to replace the faulty processor
- Avoiding the expense of unneeded capabilities of a centralized system by combining several low cost processor.

6.Explain how co processor works and interacts with 8086 . (June 2016)

<u>Coprocessor configuration</u>

Coprocessors and closely coupled configurations are similar in that both the CPU and the external processor share:

- Memory
- I/O system
- Bus & bus control logic
- Clock generator

WAIT instruction allows the processor to synchronize itself with external hardware, eg., waiting for 8087 math co-processor. When the CPU executes WAIT waiting state.

TEST input is asserted (low), the waiting state is completed and execution will resume. ESC instruction: ESC opcode, operand, opcode: immediate value recognizable to a coprocessor as an instruction opcode

Coprocessor cannot take control of the bus, it does everything through the CPU.

- 8089 shares CPU and clock and bus control logic
- It communication with host CPU is by the way of shared memory

- The host sets up a message (command) in memory
- The independent processor interrupts host on completion.

Co processor adds instruction to the instruction set. An instruction to be executed by the co- processor is indicated by an escape **(ESC)** prefix or instruction.

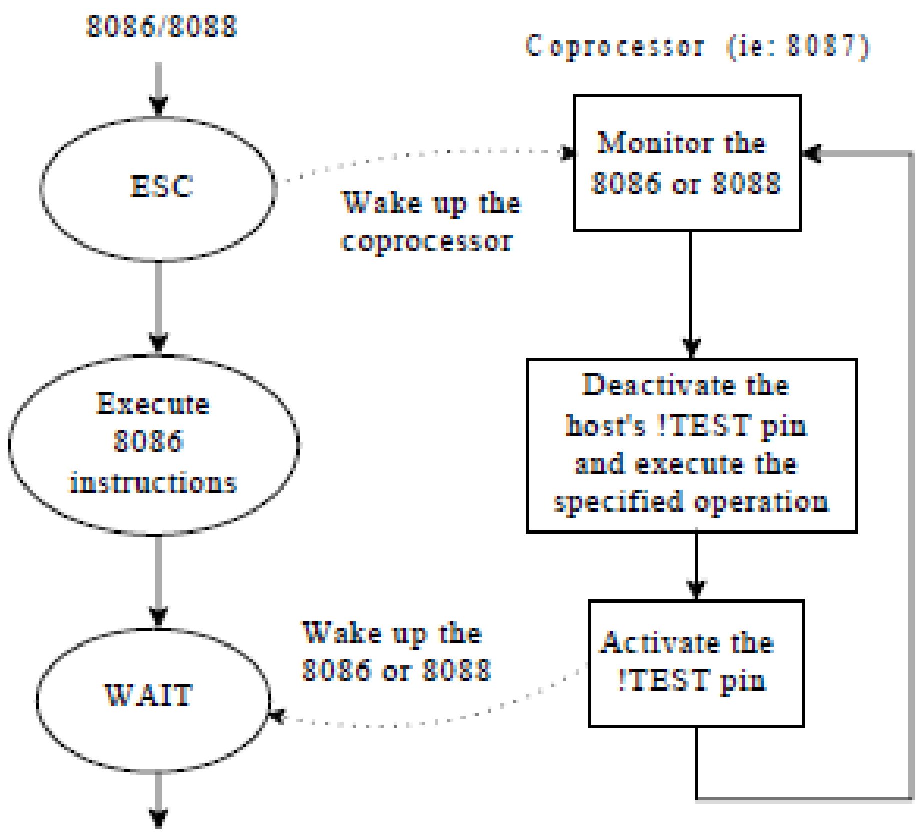


Figure: Flow diagram of coprocessor

The steps to be followed during the program execution of co processor are

1. The 8086 fetches the instruction
2. The co processor monitors the instruction sequence and captures its own instructions.
3. The ESC is decoded by the CPU and coprocessor simultaneously.
4. The CPU computes the 20 bit address of memory operand and does a dummy read. The co processor captures the address of the data and obtains control of the bus to load or store as needed.
5. The co processor sends BUSY (high) to the TEST pin
6. The CPU goes to the next instruction and if this is an 8086 instruction, the CPU and coprocessor execute in parallel.

7. If another coprocessor instruction occurs, the 8086 must wait until BUSY goes low.ie TEST pin become active. To implement this, a WAIT instruction is put in front of most 8087 instructions by the assembler.
8. The WAIT instruction does the operations ie wait until the TEST pin is active.
9. The co processor also makes use of Queue status.

****

7. Explain the closely coupled configuration of 8086 with example

. **Closely Coupled Configuration:**

The main difference between co processor and closely coupled configuration is no special instruction such as WAIT and ESC is used. The communication between 8086 and independent processor is done through memory space.

NOTE: Closely Coupled processor may take control of the bus independently. Two 8086's cannot be closely coupled.

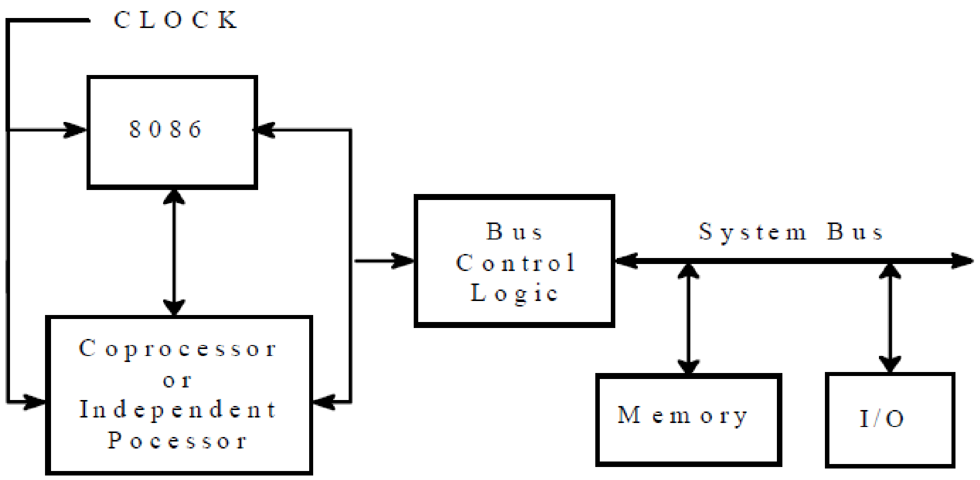


Figure: closely coupled configuration

The 8086 sets up a message in memory and wakes up independent processor by sending command to one of its ports. The independent processor then accesses the memory to execute the task in parallel with the 8086.When task is completed the external processor informs the 8086 about the completion of task by using either a status bit or an interrupt request.

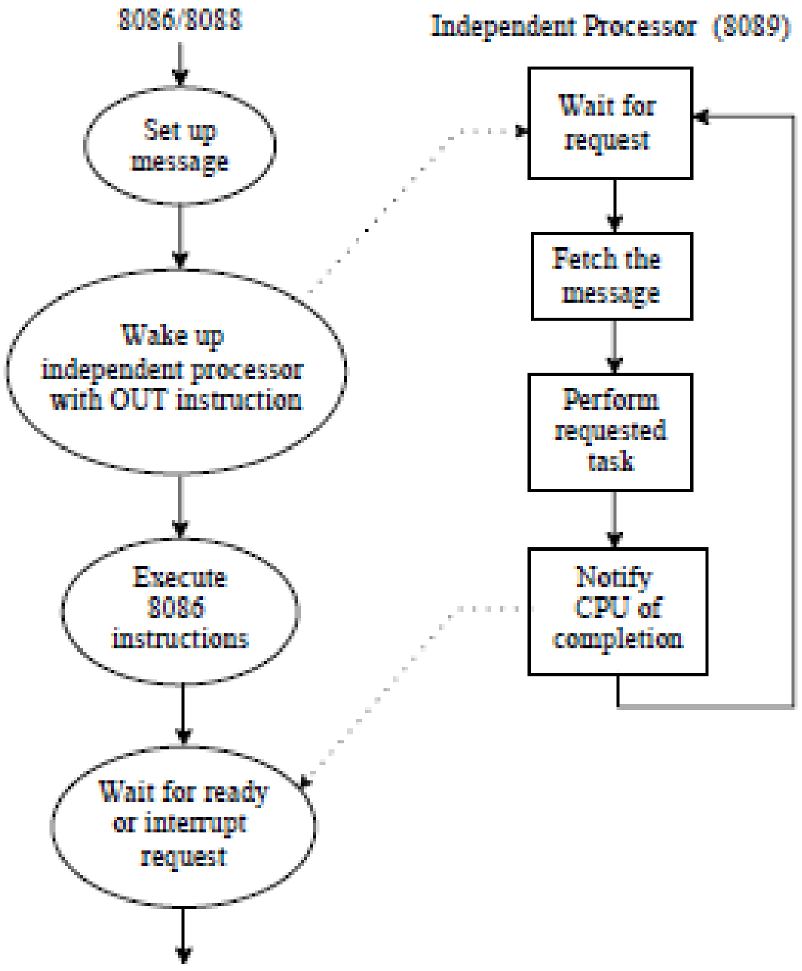

8086/8088
Set up message
Wake up independent processor with OUT instruction
Execute 8086 instructions
Wait for ready or interrupt request
Independent Processor (8089)
Wait for request
Fetch the message
Perform requested task
Notify CPU of completion

Figure:Interaction between 8086 and 8089

****

8. Write brief note on 8086 loosely coupled system configuration. (April 2006, May 2017)

<u>Loosely Coupled Configuration</u>:

- In loosely coupled configuration a number of modules of 8086 can be interfaced through a common system bus to work as a multiprocessor system.
- Each module in the loosely coupled configuration is an independent microprocessor based system with its own clock source, and its own memory and 10 devices interfaced through a local bus.
- Each module can also be a closely coupled configuration of a processor or coprocessor. The block diagram of a loosely coupled configuration of 8086 is shown in figure

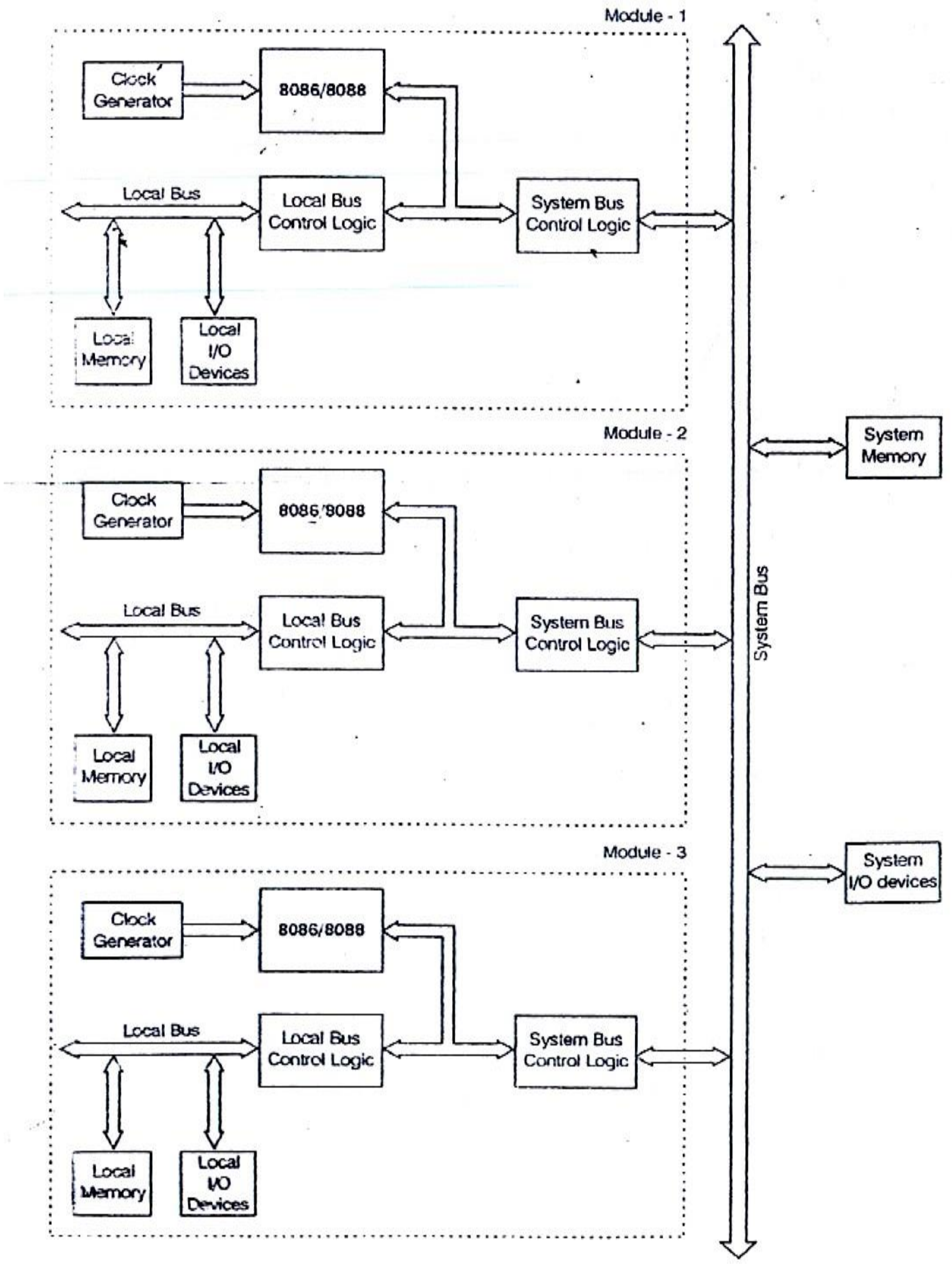


Fig loosely coupled configuration

Advantages:

1. Better system throughput by having more than one processor.
2. The system can be expanded in modular form. Each processor is an independent unit and normally on a separate PC board. One can be added or removed without affecting the others in the system.

3. A failure in one module normally does not affect the breakdown of the entire system and faulty module can be easily detected and replaced.

4. Each processor may have its own local bus to access dedicated memory or I/**O** devices so that a greater degree of parallel processing can be achieved

Disadvantages

1. **Bus Arbitration (contention**): Make sure that only 1 processor can access the bus at any given time
2. It must synchronize local and system clocks for synchronous transfer
3. It requires control chips to tie into the system bus.

****

9.Explain the basic bus access control and arbitration schemes used in multiprocessor systems. (dec 2008)

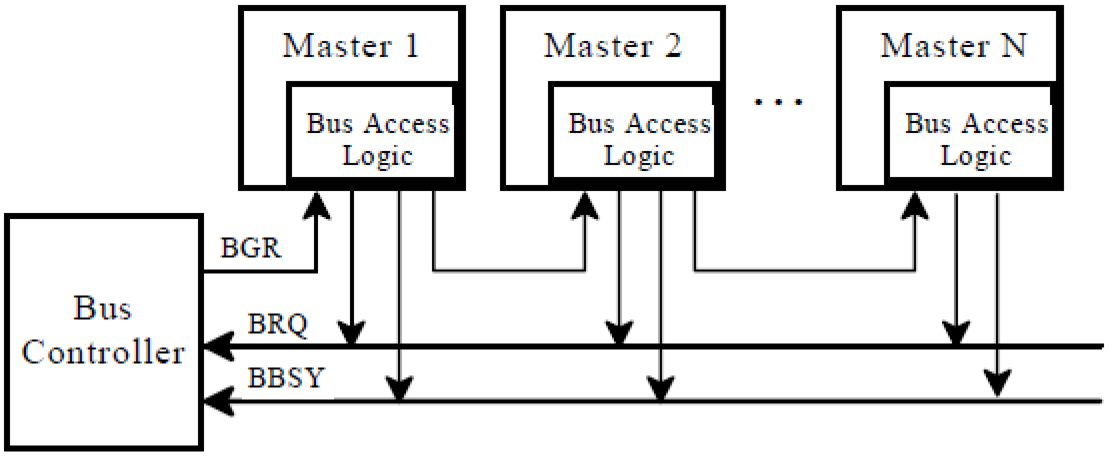
Bus allocation schemes:

- It needs some kind of priority allocation.
- It output a Bus Request (BRQ) to request the bus and BRQ line goes to some controller.
- The CPU input a Bus Grant (BGR) to gain access to bus
- The Bus access logic output a Bus Busy >BBSY= signal to hold the bus.
- To allocate the bus various methods are available.They are
    - **Daisy Chaining**
    - **Polling**
    - **Independent Priority**

Daisy Chaining:

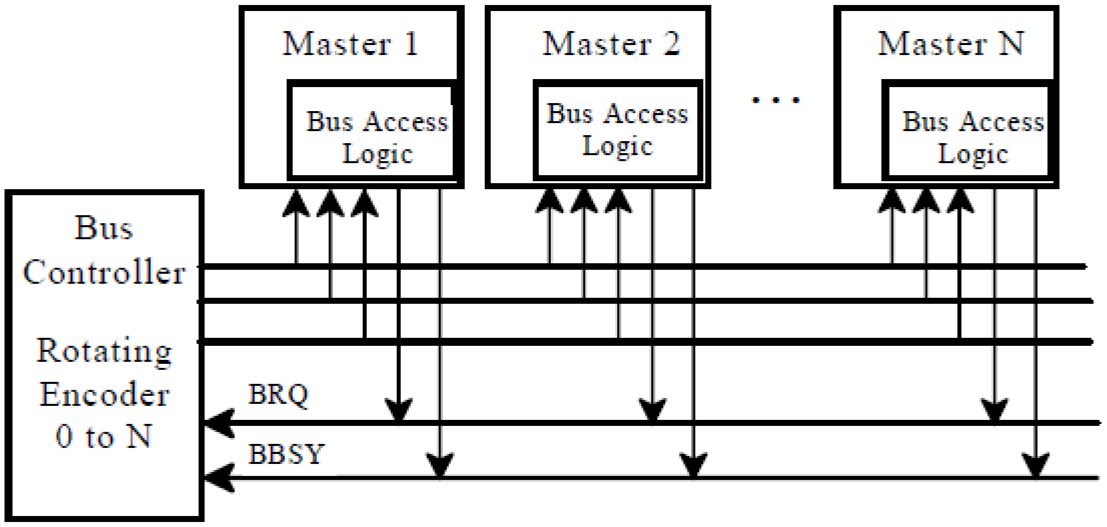
Need a bus controller to monitor bus busy and bus request signals

- It does not require any priority resolving network, rather the priorities of all the devices are essentially assumed to be in sequence.
- All the masters use a single bus request line for requesting the bus access.
- The controller sends a bus grant signal, in response to the request, if the busy signal is inactive when the bus is free.
- The bus grant pulse goes to each of the masters in the sequence till it reaches a requesting master.
- The master then receives the grant signal, activates the busy line and gains the control of the bus.
- The priority is decided by the position of the requesting master in the sequence.



Polling:

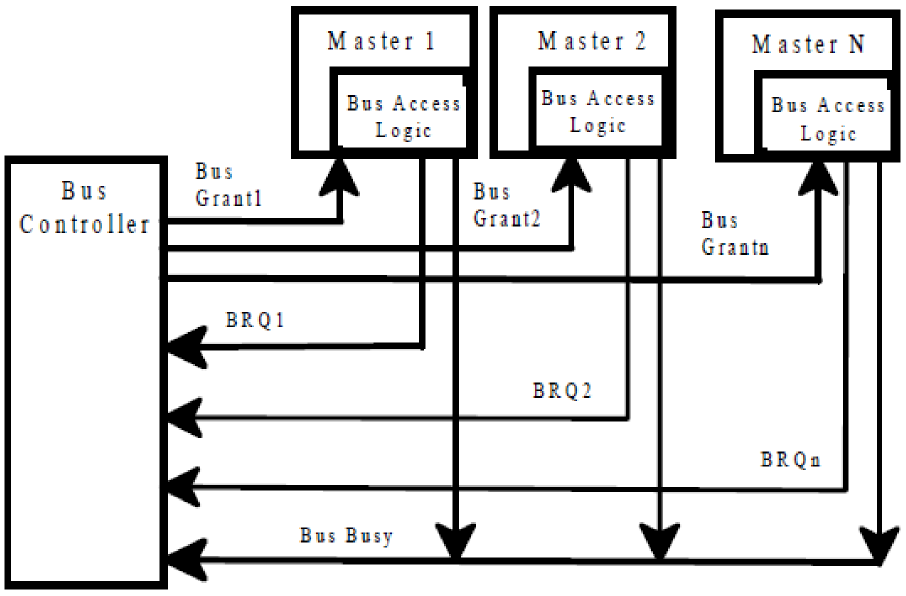
- In polling schemes, a set of address lines is driven by the controller to address each of the masters in sequence.
- When a bus request is received from a device by the controller, it generates the address on the address lines.
- If the generated address matches with that of the requesting masters, the controller activates the BUSY line.



Independent Priority

- In independent priority scheme each master has a pair of Bus request and Bus grant line and each pair has a priority assigned to it.
- The built in priority decoder within the controller selects the highest priority request a asserts the corresponding bus grant signal.
- Synchronization of the clocks must be performed once a Master is recognized.
- Master will receive a common clock from one side and pass it to the controller which will derive a clock for transfer.

- Due to separate pairs of bus request and bus grant signals, arbitration is fast.



<u>Introduction to Advanced processors: 80286 Microprocessor</u>

<u>Salient Features of 80286</u>

✓ The 80286 is the first member of the family of advanced microprocessors with memory management and protection abilities. The 80286 CPU, with its 24-bit address bus is able to address 16 Mbytes of physical memory. Various versions of 80286 are available that runs on 12.5 MHz, 10 MHz and 8 MHz clock frequencies. 80286 is upwardly compatible with 8086 in terms of instruction set.

✓ 80286 has two operating modes namely real address mode and virtual address mode. In real address mode, the 80286 can address upto 1Mb of physical memory address like 8086. In virtual address mode, it can address up to 16 Mb of physical memory address space and 1 GB of virtual memory address space.

✓ The instruction set of 80286 includes the instructions of 8086 and 80186. 80286 has some extra instructions to support operating system and memory management. In real address mode, the 80286 is object code compatible with 8086. In protected virtual address mode, it is source code compatible with 8086. The performance of 80286 is five times faster than the standard 8086.

Need for Memory Management

The part of main memory in which the operating system and other system programs are stored is not accessible to the users. It is required to ensure the smooth execution of the running process and also to ensure their protection. The memory management which is an important task of the operating system is supported by a hardware unit called memory management unit.

Swapping in of the Program

Fetching of the application program from the secondary memory and placing it in the physical memory for execution by the CPU.

Swapping out of the executable Program

Saving a portion of the program or important results required for further execution back to the secondary memory to make the program memory free for further execution of another required portion of the program.

<u>Concept of Virtual Memory</u>

Large application programs requiring memory much more than the physically available 16Mbytes of memory, may be executed by diving it into smaller segments. Thus for the user, there exists a very large logical memory space which is not actually available. Thus there exists a virtual memory which does not exist physically in a system. This complete process of virtual memory management is taken care of by the 80286 CPU and the supporting operating system.

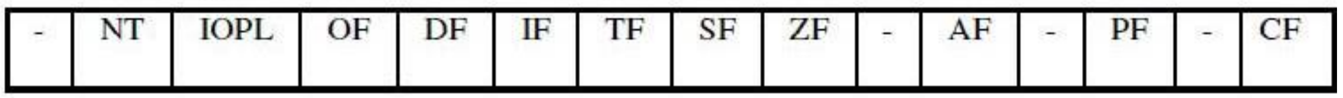
Internal Architecture of 80286

Register Organization of 80286

The 80286 CPU contains almost the same set of registers, as in 8086, namely

1. Eight 16-bit general purpose registers
2. Four 16-bit segment registers
3. Status and control registers
4. Instruction Pointer

The flag register reflects the results of logical and arithmetic instructions.



-	NT	IOPL	OF	DF	IF	TF	SF	ZF	-	AF	-	PF	-	CF

Fig. 80286 Flag Register

D_2, D_4, D_6, D_7 and D_{11} are called as status flag bits. The bits D_8 (TF) and D_9 (IF) are used for controlling machine operation and thus they are called control flags. The additional fields available in 80286 flag registers are:

1. IOPL - I/O Privilege Field (bits D12 and D13)
2. NT - Nested Task flag (bit D14)
3. PE - Protection Enable (bit D16)
4. MP - Monitor Processor Extension (bit D17)
5. EM - Processor Extension Emulator (bit D18)
6. TS – Task Switch (bit D19)

Protection Enable flag places the 80286 in protected mode, if set. This can only be cleared by resetting the CPU. If the Monitor Processor Extension flag is set, allows WAIT instruction to generate a processor extension not present exception.

Processor Extension Emulator flag if set, causes a processor extension absent exception and permits the emulation of processor extension by the CPU.

Task Switch flag if set, indicates the next instruction using extension will generate exception 7, permitting the CPU to test whether the current processor extension is for the current task.

Machine Status Word (MSW)

The machine status word consists of four flags – PE, MO, EM and TS of the four lower order bits D19 to D16 of the upper word of the flag register. The LMSW and SMSW instructions are available in the instruction set of 80286 to write and read the MSW in real address mode.

Internal Block Diagram of 80286

The CPU contain four functional blocks

1. Address Unit (AU), 2. Bus Init (BU)
3. Instruction Unit (IU), 4. Execution Unit (EU)

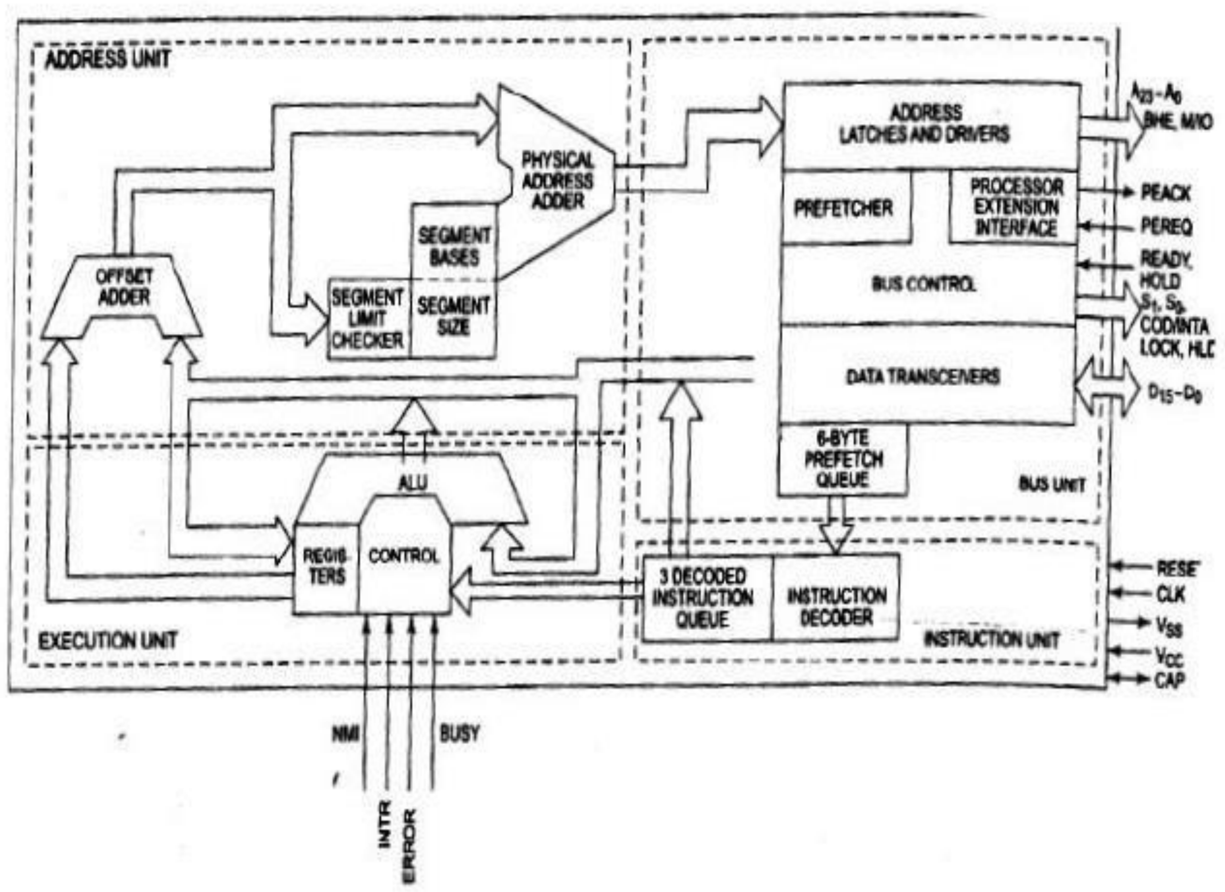
The address unit is responsible for calculating the physical address of instructions and data that the CPU wants to access. Also the address lines derived by this unit may be used to address different peripherals. The physical address computed by the address unit is handed over to the bus unit (BU) of the CPU. Major function of the bus unit is to fetch instruction bytes from the memory. Instructions are fetched in advance and stored in a queue to enable faster execution of the instructions.

The bus unit also contains a bus control module that controls the prefetcher module. These prefetched instructions are arranged in a 6-byte instructions queue. The 6-byte prefetch queue forwards the instructions arranged in it to the **instruction unit** (IU).

The instruction unit accepts instructions from the prefetch queue and an instruction decoder decodes them one by one. The decoded instructions are latched onto a decoded instruction queue. The output of the decoding circuit drives a control circuit in the **execution unit,** which is responsible for executing the instructions received from decoded instruction queue.

The decoded instruction queue sends the data part of the instruction over the data bus. The EU contains the register bank used for storing the data as scratch pad, or used as special purpose registers. The ALU, the heart of the EU, carries out all the arithmetic and logical operations and sends the results over the data bus or back to the register bank.

8.2.2 Internal Block Diagram of 80286



Interrupts of 80286

The Interrupts of 80286 may be divided into three categories,

1. External or hardware interrupts
2. INT instruction or software interrupts
3. Interrupts generated internally by exceptions

While executing an instruction, the CPU may sometimes be confronted with a special situation because of which further execution is not permitted. While trying to execute a divide by zero instruction, the CPU detects a major error and stops further execution.

In this case, we say that an exception has been generated. In other words, an instruction exception is an unusual situation encountered during execution of an instruction that stops further execution. The return address from an exception, in most of the cases, points to the instruction that caused the exception. As in the case of 8086, the interrupt vector table of 80286 requires 1Kbytes of space for storing 256, four-byte pointers to point to the corresponding 256 interrupt service routines (lSR).

Each pointer contains a 16-bit offset followed by a 16-bit segment selector to point to a particular ISR. The calculation of vector pointer address in the interrupt vector table from the (8-bit) INT type is exactly similar to 8086. Like 8086, the 80286 supports the software interrupts of type 0 (INT 00) to type FFH (INT FFH).

Maskable Interrupt INTR: This is a maskable interrupt input pin of which the INT type is to be provided by an external circuit like an interrupt controller. The other functional details of this interrupt pin are exactly similar to the INTR input of 8086.

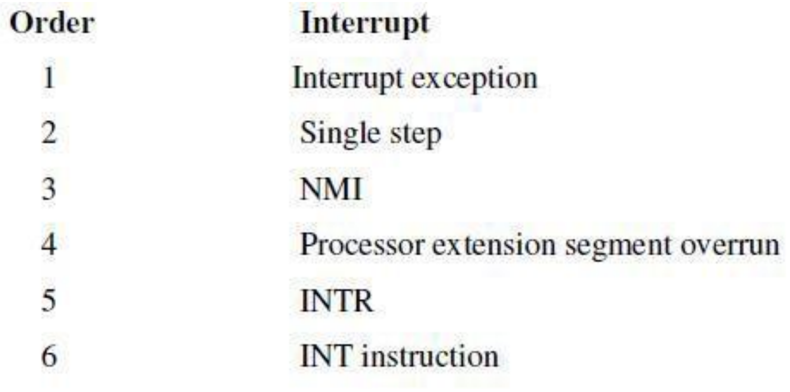
Non-Maskable Interrupt NMI: It has higher priority than the INTR interrupt. Whenever this interrupt is received, a vector value of 02 is supplied internally to calculate the pointer to the interrupt vector table. Once the CPU responds to a NMI request, it does not serve any other interrupt request (including NMI). Further it does not serve the processor extension (coprocessor) segment overrun interrupt, till either it executes IRET or it is reset. To start with, this clears the IF flag which is set again with the execution of IRET, i.e. return from interrupt.

Single Step Interrupt

As in 8086, this is an internal interrupt that comes into action, if *trap* flag (TF) of 80286 is set. The CPU stops the execution after each instruction cycle so that the register contents (including flag register), the program status word and memory, etc. may be examined at the end of each instruction execution. This interrupt is useful for troubleshooting the software. An interrupt vector type 01 is reserved for this interrupt.

Interrupt Priorities:

If more than one interrupt signals occur simultaneously, they are processed according to their priorities as shown below:



Order	Interrupt
1	Interrupt exception
2	Single step
3	NMI
4	Processor extension segment overrun
5	INTR
6	INT instruction

FUNCTION	Interrupt Number
Divide error exception	0
Single step interrupt	1
NMI interrupt	2
Breakpoint interrupt	3
INTO detected overflow exception	4
BOUND range exceeded exception	5
Invalid opcode exception	6
Processor extension not available exception	7
Intel reserved, do not use	8-15
Processor extension error interrupt	16
Intel reserved, do not use	17-31
User defined	32-255

Signal Description of 80286

CLK: This is the system clock input pin. The clock frequency applied at this pin is divided by two internally and is used for deriving fundamental timings for basic operations of the circuit. The clock is generated using 8284 clock generator.

D_{15}-D_0: These are sixteen bidirectional data bus lines. **A_{23}-A_0**: These are the physical address output lines used to address memory or I/O devices. The address lines A23 - A16 are zero during I/O transfers

BHE: This output signal, as in 8086, indicates that there is a transfer on the higher byte of the data bus (D15 – D8) .

S1 , **S0:** These are the active-low status output signals which indicate initiation of a bus cycle and with M/IO and COD/INTA, they define the type of the bus cycle.

M/ IO: This output line differentiates memory operations from I/O operations. If this signal is it “0” indicates that an I/O cycle or INTA cycle is in process and if it is “1” it indicates that a memory or a HALT cycle is in progress.

COD/ INTA: This output signal, in combination with M/ IO signal and S1 , S0 distinguishes different memory, I/O and INTA cycles.

LOCK: This active-low output pin is used to prevent the other masters from gaining the control of the bus for the current and the following bus cycles. This pin is activated by a "LOCK" instruction prefix, or automatically by hardware during XCHG, interrupt acknowledge or descriptor table access

READY This active-low input pin is used to insert wait states in a bus cycle, for interfacing low speed peripherals. This signal is neglected during HLDA cycle.

HOLD and HLDA This pair of pins is used by external bus masters to request for the control of the system bus (HOLD) and to check whether the main processor has granted the control (HLDA) or not, in the same way as it was in 8086.

INTR: Through this active high input, an external device requests 80286 to suspend the current instruction execution and serve the interrupt request. Its function is exactly similar to that of INTR pin of 8086.

NMI: The Non-Maskable Interrupt request is an active-high, edge-triggered input that is equivalent to an INTR signal of type 2. No acknowledge cycles are needed to be carried out.

PEREG and **PEACK** (**Processor Extension Request and Acknowledgement**)

Processor extension refers to coprocessor (80287 in case of 80286 CPU). This pair of pins extends the memory management and protection capabilities of 80286 to the processor extension 80287. The PEREQ input requests the 80286 to perform a data operand transfer for a processor extension. The PEACK active-low output indicates to the processor extension that the requested operand is being transferred.

BUSY and **ERROR:** Processor extension BUSY and ERROR active-low input signals indicate the operating conditions of a processor extension to 80286. The BUSY goes low, indicating 80286 to suspend the execution and wait until the BUSY become inactive.

In this duration, the processor extension is busy with its allotted job. Once the job is completed the processor extension drives the BUSY input high indicating 80286 to continue with the program execution. An active ERROR signal causes the 80286 to perform the processor extension interrupt while executing the WAIT and ESC instructions. The active ERROR signal indicates to 80286 that the processor extension has committed a mistake and hence it is reactivating the processor extension interrupt.

CAP: A 0.047 µf, 12V capacitor must be connected between this input pin and ground to filter the output of the internal substrate bias generator. For correct operation of 80286 the capacitor must be charged to its operating voltage. Till this capacitor charges to its full capacity, the 80286 may be kept stuck to reset to avoid any spurious activity.

V_{ss}: This pin is a system ground pin of 80286.

V_{cc}: This pin is used to apply +5V power supply voltage to the internal circuit of 80286. RESET The active-high RESET input clears the internal logic of 80286, and reinitializes it.

RESET The active-high reset input pulse width should be at least 16 clock cycles. The 80286 requires at least 38 clock cycles after the trailing edge of the RESET input signal, before it makes the first opcode fetch cycle.

<u>Real Address Mode</u>

- Act as a fast 8086
- Instruction set is upwardly compatible
- It address only 1 M byte of physical memory using A_0-A_{19}.
- In real addressing mode of operation of 80286, it just acts as a fast 8086. The instruction set is upward compatible with that of 8086.

The 80286 addresses only 1Mbytes of physical memory using A_0- A_{19}. The lines A_{20}-A_{23} are not used by the internal circuit of 80286 in this mode. In real address mode, while addressing the physical memory, the 80286 uses BHE along with A_0- A_{19}. The 20-bit physical address is again formed in the same way as that in 8086.

The contents of segment registers are used as segment base addresses. The other registers, depending upon the addressing mode, contain the offset addresses. Because of extra pipelining and other circuit level improvements, in real address mode also, the 80286 operates at a much faster rate than 8086, although functionally they work in an identical fashion. As in 8086, the physical memory is organized in terms of segments of 64Kbyte maximum size.

An exception is generated, if the segment size limit is exceeded by the instruction or the data. The overlapping of physical memory segments is allowed to minimize the memory requirements for a task. The 80286 reserves two fixed areas of physical memory for system initialization and interrupt vector table. In the real mode the first 1Kbyte of memory starting from address 0000H to 003FFH is reserved for interrupt vector table. Also the addresses from FFFF0H to FFFFFH are reserved for system initialization.

The program execution starts from FFFFH after reset and initialization. The interrupt vector table of 80286 is organized in the same way as that of 8086. Some of the interrupt types are reserved for exceptions, single-stepping and processor extension segment overrun, etc

When the 80286 is reset, it always starts the execution in real address mode. In real address mode, it performs the following functions: it initializes the IP and other registers of 80286, it prepares for entering the protected virtual address mode.

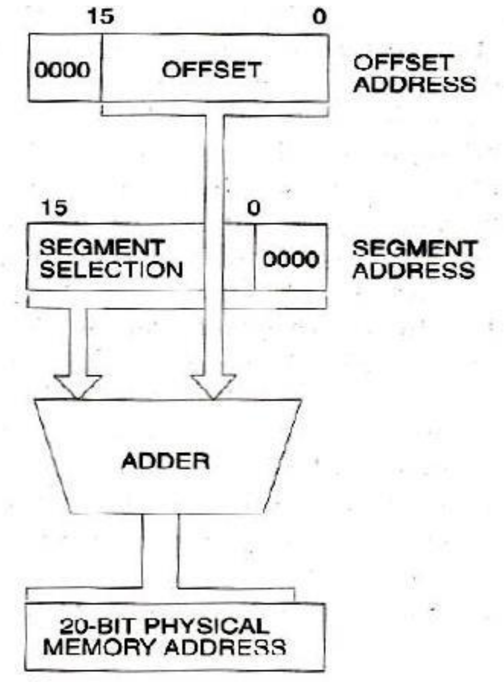


Fig. Real Address Mode Address Calculation

Protected Virtual Address Mode (PVAM)

80286 is the first processor to support the concepts of virtual memory and memory management. The virtual memory does not exist physically it still appears to be available within the system. The concept of VM is implemented using Physical memory that the CPU can directly access and secondary memory that is used as a storage for data and program, which are stored in secondary memory initially.

The Segment of the program or data required for actual execution at that instant is fetched from the secondary memory into physical memory. After the execution of this fetched segment, the next segment required for further execution is again fetched from the secondary memory, while the results of the executed segment are stored back into the secondary memory for further references. This continues till the complete program is executed.

During the execution the partial results of the previously executed portions are again fetched into the physical memory, if required for further execution. The procedure of fetching the chosen program segments or data from the secondary storage into physical memory is called *swapping*. The procedure of storing back the partial results or data back on the secondary storage is called *unswapping*. The virtual memory is allotted per task.

The 80286 is able to address 1 G byte (2^{30} bytes) of virtual memory per task. The complete virtual memory is mapped on to the 16Mbyte physical memory. If a program larger than 16Mbyte is stored on the hard disk and is to be executed, if it is fetched in terms of data or program segments of less than 16Mbyte in size into the program memory by swapping sequentially as per sequence of execution.

Whenever the portion of a program is required for execution by the CPU, it is fetched from the secondary memory and placed in the physical memory is called *swapping in* of the program. A portion of the program or important partial results required for further execution, may be saved back on secondary storage to make the PM free for further execution of another required portion of the program is called *swapping out* of the executable program.

UNIT III
I/O INTERFACING

Memory Interfacing and I/O interfacing - Parallel communication interface – Serial communication interface – D/A and A/D Interface - Timer – Keyboard /display controller – Interrupt controller – DMA controller – Programming and applications Case studies: Traffic Light control, LED display , LCD display, Keyboard display interface and Alarm Controller.

1. Explain in detail about Memory Interfacing and I/O interfacing of 8086.

8086 memory is divided into two memory banks and each memory bank size is 512K X 8 bits (Shown in fig-1)

• Low-bank holds even addressed bytes 00000H through FFFFEH

• High-bank holds odd addressed bytes 00001H through FFFFFH

• Address/data bus is demultiplexed.

• Input bus: 20-bit address bus (A_{19} through A_0), and BHE*

A1-A19, address lines select storage location

If A0 = 0 enables low memory bank

If BHE* = 0 enables high memory bank

• Input / Output bus: 16-bit data bus (D_{15} through D_0)

D7-D0	: Even addressed byte accesses
D15-D8	: Odd addressed byte accesses
D15-D0	: Word accesses

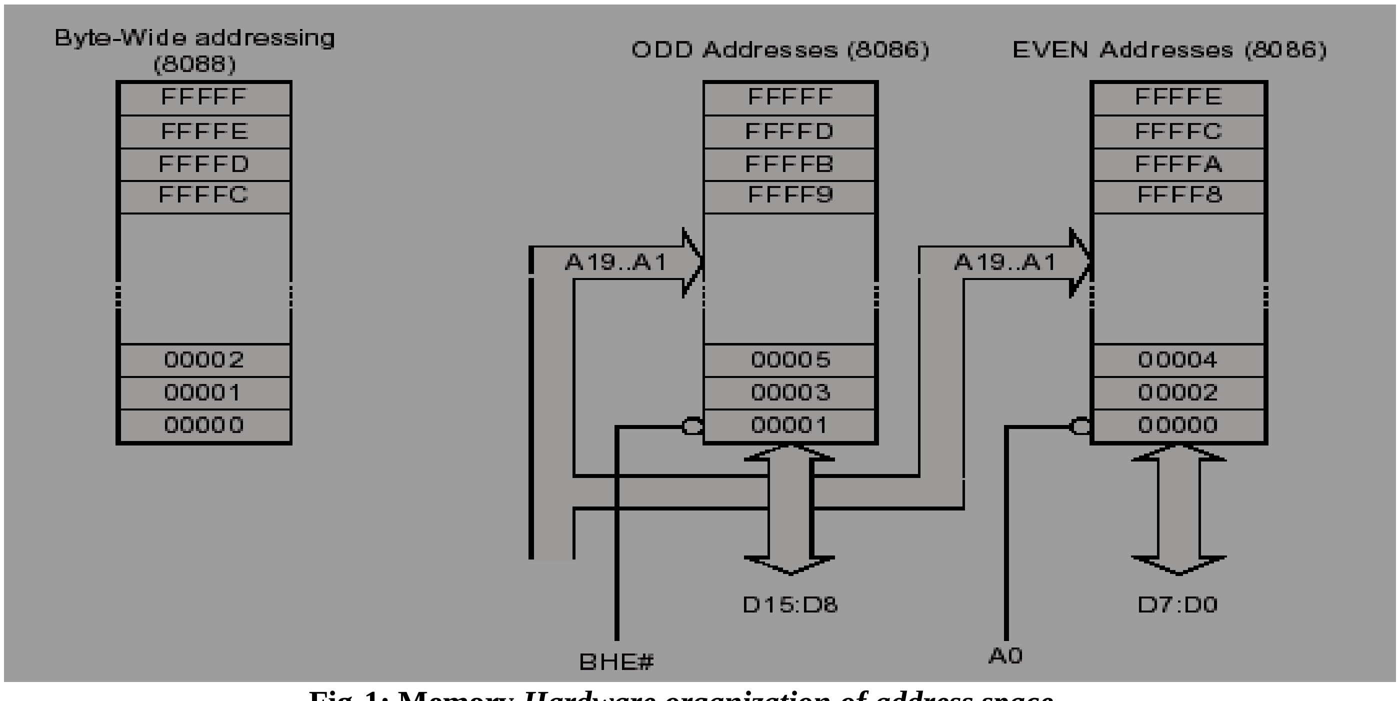


Fig-1: Memory *Hardware organization of address space*

A type of data writes that may take place:

• Byte to a storage location in the upper (odd) bank
• Byte to a storage location in the lower (even) bank
• Word to storage locations in both banks
• Write control logic must decode A0L, BHEL* and MWTC* to produce independent write signals WRU* and WRL* (Shown in fig -2)

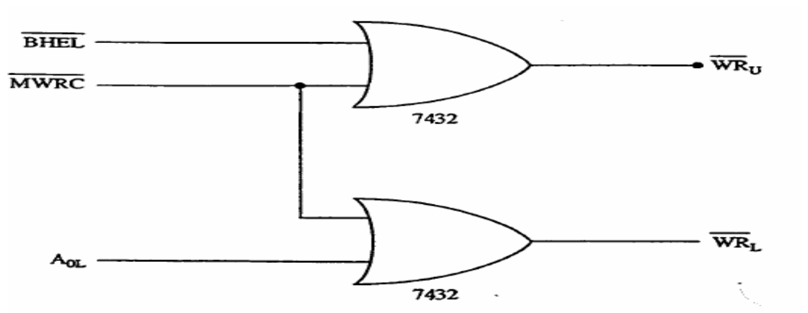


Fig -2: 2-input OR gate solution for decoding write control signals

• MWTC* =0 enables both gates

BHEL*	A_{OL}	**WRU***	**WRL***	**Bank selection**
0	0	0	0	Both banks enabled
1	0	1	0	Lower (even) bank enabled
0	1	0	1	Upper (odd) bank enabled

• All accesses take a minimum of one bus cycle of duration
@5MHz—800ns
@8MHz—500ns

During all memory accesses one of three bus cycle status code are output by the MPU (Microprocessor Unit)
• Opcode fetch
• Read memory
• Write memory

• 8288 decodes to produce appropriate control / command signals

• MRDC*	Opcode fetch/memory read
• MWTC*	Memory write
• AMWC*	Advanced memory write

Building blocks of the maximum mode 8086 memory interface

It has the following blocks shown in fig-3.:
• 8288 bus controller
• Address bus latch
• Address decoder
• Data bus transceiver/buffer
• Bank read control logic

• Bank write control logic
• Memory subsystem

Parts of address applied to address inputs of memory subsystem, address decoder, and read/ write control logic **banks**

- Bank selection is accomplished in two ways:
  - separate write signal is developed to select a write to each bank of the memory
  - separate decoders are used for each bank
- The first technique is by far the least costly approach to memory interface.
- The second technique is only used in a system that must achieve most efficient use of the power supply.

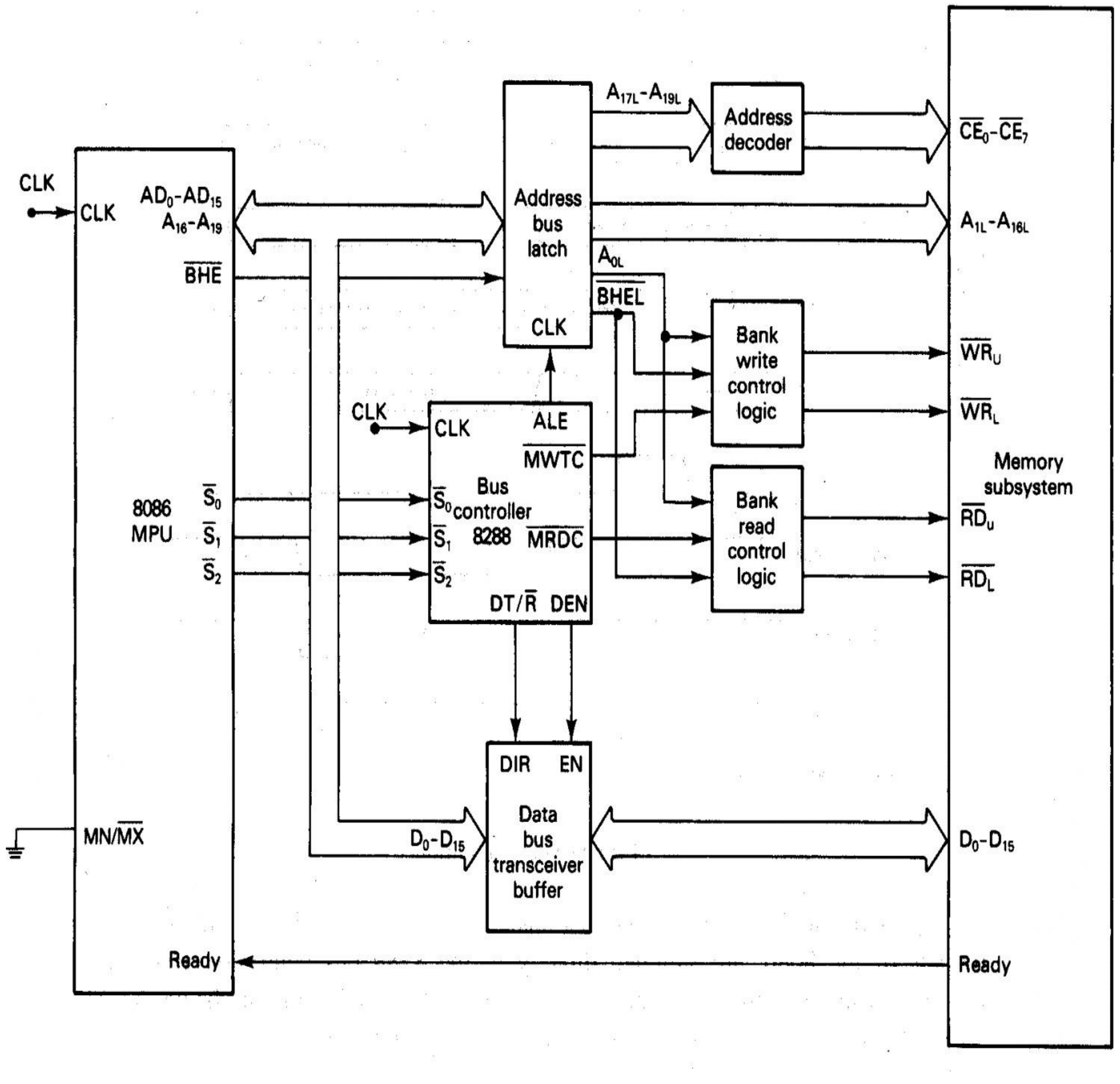


Fig-3 : Maximum mode of 8086 memory interface

I/O Interfacing

- To communicate with the outside world, microprocessor use Peripherals, I/O Devices such as keyboards, A/D converters, input devices and output devices such as CRT, Printers etc.,

- These input and output devices are called Peripherals or I/Os.
- Peripherals are connected to the microprocessor through electronic circuits known as interfacing circuits.
- These interfacing circuits convert the data available from an input device into compatible format for the computer.
- The interface associated with the output device converts the output of the microprocessor into the desired peripheral format.

There are two schemes (**Interfacing Configurations)** for the allocation of addresses to memories and input/output devices. They are

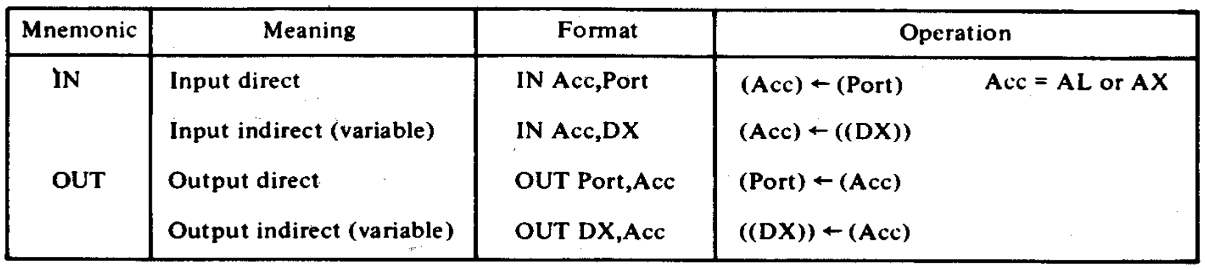
Interfacing Configurations

1. Memory mapped I/O
2. I/O mapped I/O (Isolated I/O)

Isolated I/O: It uses I/O instructions (IN & OUT) and it has its own address space for I/O ports (0000H-FFFFH), isolated from the memory address space.

Memory mapped I/O: uses memory reference instructions (e.g. MOV). So address space is shared between memory and I/O.

- Memory-mapped I/O **does not use** the **IN** or **OUT** instructions.
- It uses **any instruction** that transfers data between the microprocessor and memory.
    - treated as a memory location in memory map
- Same as interfacing 8086 memory in Minimum Mode and Maximum Mode
- I/O devices are treated **separately** from memory
- Address 0000 to 00FF is referred to **page 0**.
- Special instructions exist for this address range
- **Advantage:** Any memory transfer instruction can access the I/O device.
- **Disadvantage:** A portion of memory system is used as the I/O map and reduces memory available to applications



Mnemonic	Meaning	Format	Operation	
IN	Input direct	IN Acc,Port	(Acc) ← (Port)	Acc = AL or AX
	Input indirect (variable)	IN Acc,DX	(Acc) ← ((DX))	
OUT	Output direct	OUT Port,Acc	(Port) ← (Acc)	
	Output indirect (variable)	OUT DX,Acc	((DX)) ← (Acc)	

Memory mapped I/O

In this type of I/O interfacing, the 8086 uses 20 address lines to identify an I/O device. The I/O device is connected as memory device.

The 8086 uses same control signals and instructions to access I/O. RD and WR signals are activated indicating memory bus cycle.

I/O mapped I/O:

8086 has special instructions IN and OUT to transfer data through the input/output ports in I/O mapped I/O system.

The IN instruction copies data from an input port to the Accumulator. The OUT instruction copies a byte from AL or a word from AX to the specified port.

The M/IO signal is always low when 8086 is executing these instructions. Address of I/O device is 8-bit or 16-bit long.

Program to operate in I/O mode.

- **To write the data 00H into Output port 62H:**

```
MOV  AL,00H
OUT  62H,AL
       or
MOV  AL,00H
MOV  DX,62H
OUT  DX,AL
```

- **To read a byte from Input port address 71H:**

```
IN  AL,71H
       or
MOV  DX,71H
IN AL,DX
```

Minimum mode interface

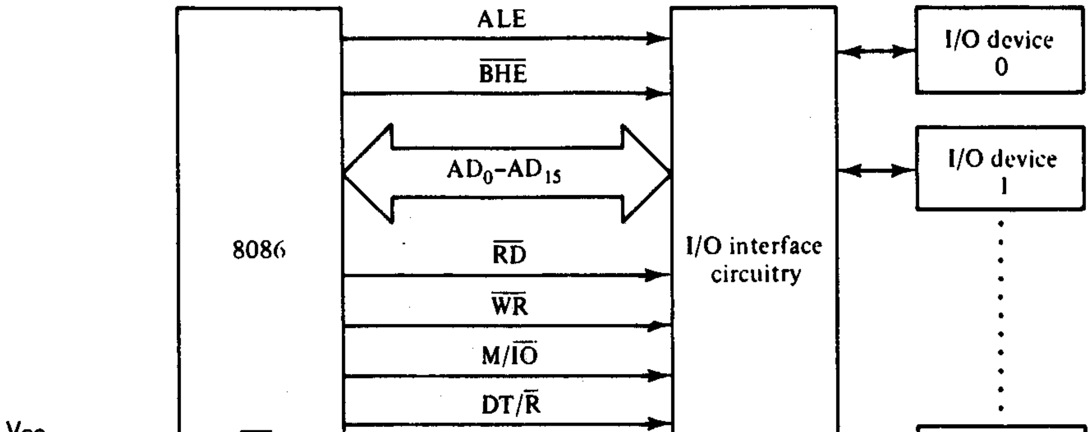


Fig-4: I/O interfacing with minimum mode

Maximum mode interface

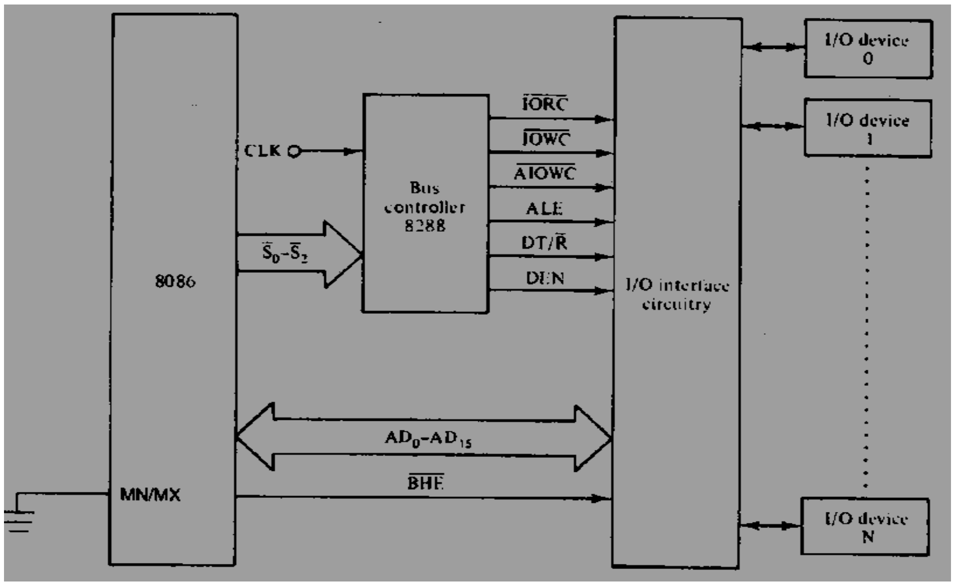


Fig-5: I/O interfacing with maximum mode

2. Describe the internal block diagram of 8255 (December 2010) (or)
Parallel communication interface (8255)
(Programmable peripheral interface)

- The 8255 is a general purpose programmable I/O device used for parallel data transfer.
- It has 24 I/O programmable pins which can be grouped into three 8 bit parallel ports of Port A , Port B and Port C. It is TTL compatible.
- The eight bit ports of PORT C can be used as individual bits or be grouped into two 4 bit ports. C_{upper} (C_u) and C_{Lower} (C_L).
- The functions of 8255 are classified according to two modes. The Bit Set/Reset mode and the I/O mode. The BSR mode is used to set or reset the bits in Port C.
- The 8-bit data bus buffer is controlled by the read/write control logic. The read/write control logic manages all of the internal and external transfers of both data and control words.
- RD , WR , A_1, A_0 and RESET are the inputs provided by the microprocessor to the READ/ WRITE control logic of 8255.
- The 8-bit, 3-state bidirectional buffer is used to interface the 8255 internal data bus with the external system data bus.
- This buffer receives or transmits data based on the execution of input or output instructions by the microprocessor. The control word is also transferred through the buffer.

Functions of Pin:
The signal descriptions of 8255 are briefly presented as follows:

• **PA_7-PA_0**: These are eight port A lines that acts as either latched output or buffered input lines depending upon the control word loaded into the control word register.

• **PC_7-PC_4** : Upper nibble of port C lines. They may act as either output latches or input buffers lines.

• This port also can be used for generation of handshake lines in mode 1 or mode 2.

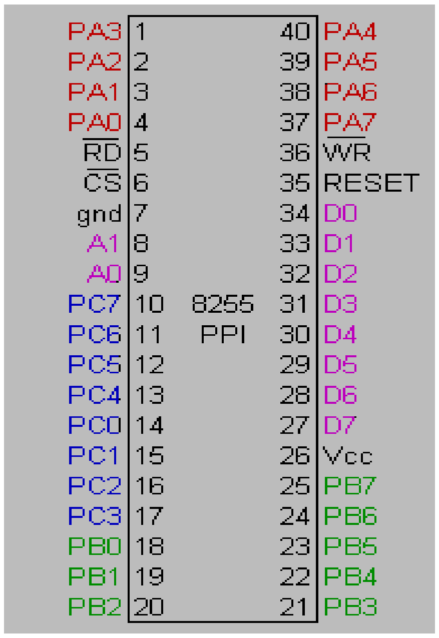
• **PC_3-PC_0** : Lower nibble of port C lines. They may act as either output latches or input buffers lines.

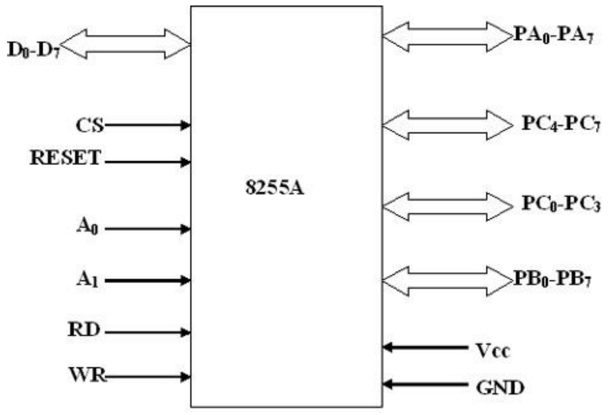
• This port also can be used for generation of handshake lines in mode 1 or mode 2.

- **PB_0-PB_7** : These are eight port B lines which are used as latched output lines or buffered input lines in the same way as port A.
- **A_1-A_0** : These are the address input lines and are driven by the microprocessor.
  - In case of 8086 systems, if the 8255 is to be interfaced with lower order data bus, the A_0 and A_1 pins of 8255 are connected with A_1 and A_2 of 8086 respectively.
  - These address lines A_1- A_0 are used for addressing any one of the four registers, i.e. three ports and a control word register as given in table below.

A1	**A0**	**Select**
0	**0**	**P_A**
0	**1**	**P_B**
1	**0**	**P_C**
1	**1**	**Control register**

- **RD** : This is the input line driven by the microprocessor and should be low to indicate read operation to 8255.
- **WR** : This is an input line driven by the microprocessor. A low on this line indicates write operation.
- **CS** : This is a chip select line. If this line goes low, it enables the 8255 to respond to RD and WR Signals.
- **D_0-D_7** : These are the data bus lines carry data or control word to/from the microprocessor.
- **RESET** : A logic high on this line, clears the control word register of 8255. All ports are set as input ports by default after reset.





Signals of 8255

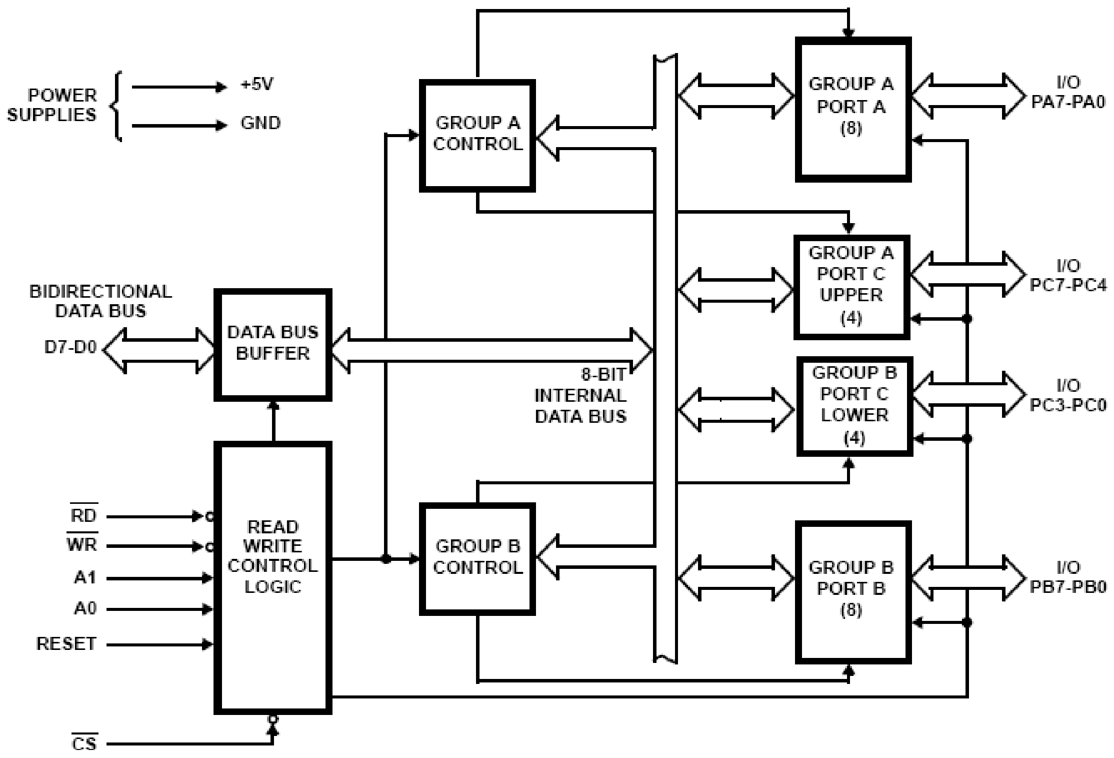


Fig-6: Pin diagram &Block diagram of 8255 Programmable Peripheral interface.

The 8255 consists of four sections namely,

- Data bus buffer
- Read/write control logic
- Group A control
- Group B control

Data Bus buffer:

- It is an 8-bit bidirectional Data bus.
- Used to interface between 8255 data bus with system bus.
- The internal data bus and Outer pins D_0-D_7 pins are connected in internally.
- The direction of data buffer is decided by Read/Control Logic.

Read/Write Control Logic:

- This is getting the input signals from control bus and Address bus
- Control signal are RD and WR.
- Address signals are A0, A1and CS.
- 8255 operation is enabled or disabled by CS.

Group A and Group B control:

- Group A and B get the Control Signals from CPU and send the command to the individual control blocks.
- Group A send the control signal to port A and Port C (Upper) PC7-PC4.
- Group B send the control signal to port B and Port C (Lower) PC3-PC0.

PORT A:

- This is an 8-bit buffered I/O latch.
- It can be programmed by mode 0 , mode 1 & mode 2 **.**

PORT B:

- This is an 8-bit buffer I/O latch.

- It can be programmed by mode 0 and mode 1.

PORT C:

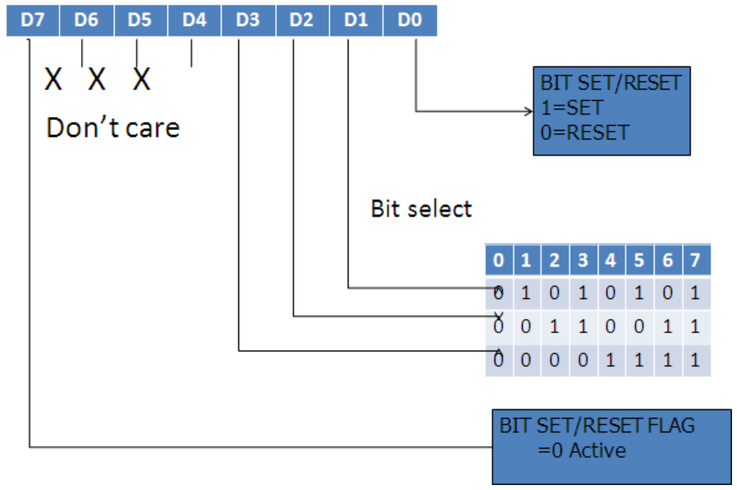
- This is an 8-bit Unlatched buffer Input and an Output latch.
- It is divided into two parts as Port C (Upper) PC7-PC4 & Port C (Lower) PC3-PC0
- It can be programmed by bit set/reset operation.

CONTROL WORD FORMATS:

There are two control word formats i) BSR mode ii) Input / Output mode

FOR BIT SET/RESET MODE:

- This is bit set/reset control word format.



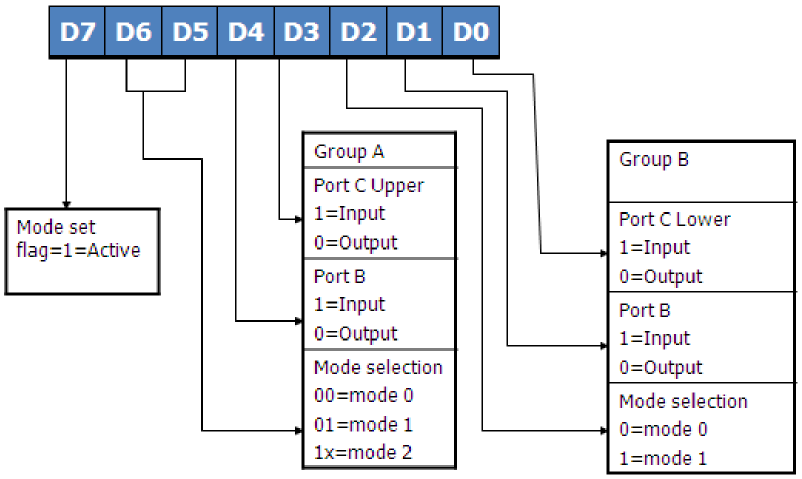
- PC0-PC7 is set or reset as per the status of D0.
- A BSR word is written for each bit of Port C

Example:

- PC3 is Set then control register will be 0XXX0111.
- PC4 is Reset then control register will be 0XXX01000.
- X is a don't care.

FOR I/O MODE

The mode format for I/O as shown in figure



- The control word for both Mode 1 and Mode 2 are same.

- Bit D7 is used for specifying whether word loaded in to Bit set/reset mode or Mode definition word.
- D7=1=Mode definition mode.
- D7=0=Bit set/Reset mode.

Steps to communicate with peripherals through the 8255:

1. Determine the addresses of Port A, port B, port C and control register according to the chip select logic and address lines A0 and A1.
2. Write a control word in the control register.
3. Write I/O instructions to communicate with peripherals through ports A, B and C.

Operation modes

BIT SET/RESET MODE:

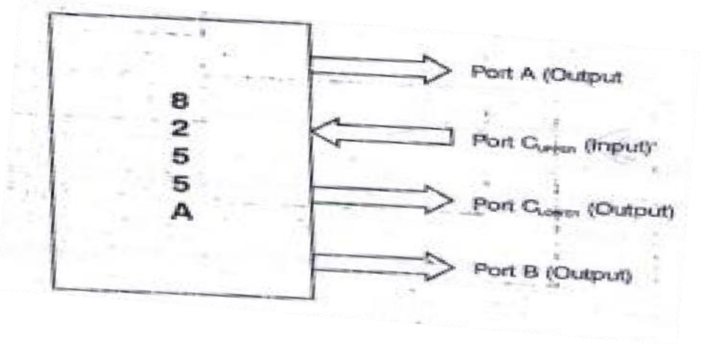
- The PORT C can be Set or Reset by sending OUT instruction to the CONTROL registers.
- In BSR mode, individual bits of Port C can be used for applications such as on/off switch.
- The control word sets or resets one bit at a time.
- BSR control word does not alter any previously transmitted control word with bit D7=1. Thus the I/O operations of Port A and Port B are not affected by a BSR control word.

I/O MODES:

- The I/O mode is divided into three modes as mode 0, mode 1, and mode 2.
  - Mode 0 – Basic I/O mode
  - Mode 1 – strobbed I/O mode
  - Mode 2 – Bidirectional data transfer mode

MODE 0 (Simple input / Output):

- In this mode , port A and port B are used as two simple 8 bit I/O ports and port C as two 4 bit ports used as individually (Simply).



Features:

- Outputs are latched, Inputs are buffered.
- Ports do not have Handshake or interrupt capability.

MODE 1 : (Input/output with Hand shake)

- In this mode, input or output is transferred by hand shaking Signals. The handshaking signals are exchanged between the microprocessor and peripherals.

The features of this mode include the following

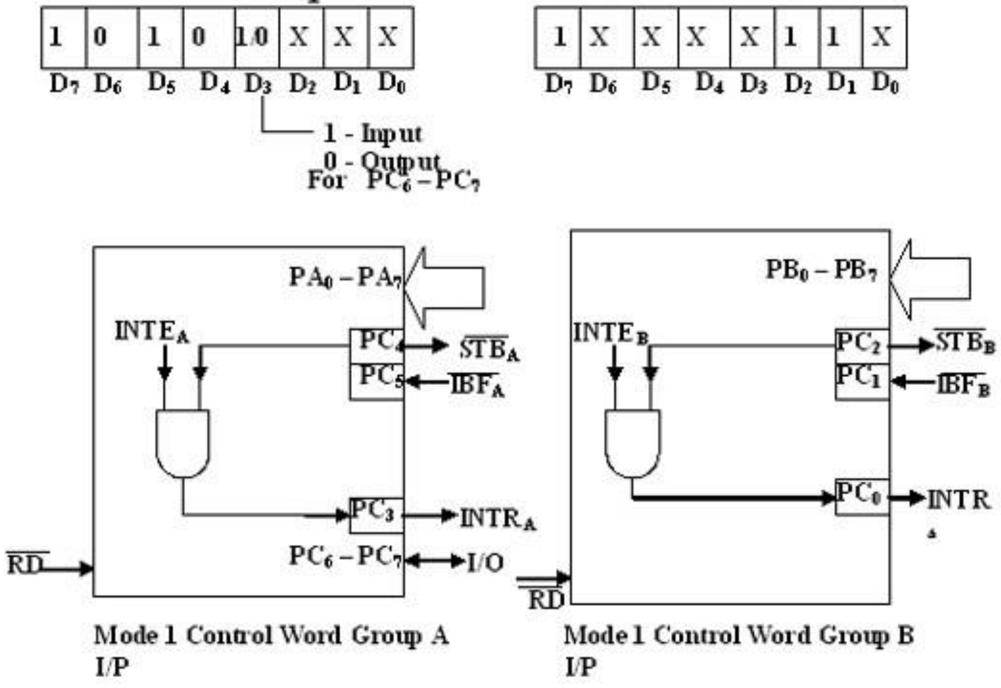
1. Two ports (A and B) function as 8 bit I/O ports .They can be configured either as input or output ports.
2. Each port uses 3 lines from port C as handshake signals. The remaining 2 lines of PORT C can be used for simple I/O operations.
3. Input and outputs data are latched.
4. Interrupt logic is supported.

In 8255, the specific lines from PORT C used for handshake signals vary according to the I/O function of a port. Therefore input and output functions in Mode 1 are discussed separately.

Input control signal definitions (Mode 1):

- **STB(Strobe input**) – If this line falls to logic low level, the data available at 8-bit input port is loaded into input latches.
- **IBF (Input buffer full)** – If this signal rises to logic 1, it indicates that data has been loaded into latches, i.e. it works as an acknowledgement.
- **INTR (Interrupt request)** – This active high output signal can be used to interrupt the CPU whenever an input device requests the service. INTR is set by a high STB pin and a high at IBF pin.
INTE is an internal flag that can be controlled by the bit set/reset mode of either PC4(INTEA) or PC2(INTEB) as shown in fig.
INTR is reset by a falling edge of RD input. Thus an external input device can be request the service of the processor by putting the data on the bus and sending the strobe signal.

Input control signal definitions in Mode 1



Mode 1 Control Word Group A I/P

Mode 1 Control Word Group B I/P

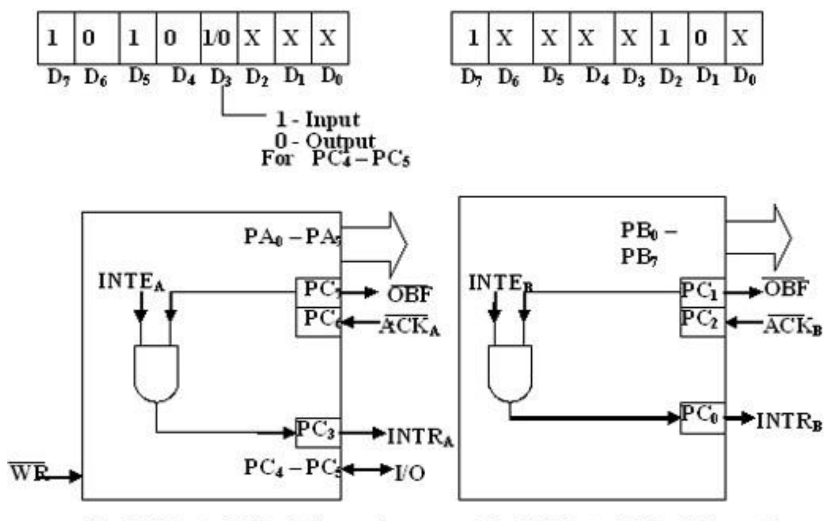
Output control signal definitions (Mode 1) :

- **OBF (Output buffer full)** – When this signal falls to low, indicates that CPU has written data to the specified output port.

• **ACK (Acknowledge input**) – ACK signal acts as an acknowledgement to be given by an output device. ACK signal, whenever low, informs the CPU that the data transferred by the CPU to the output device through the port is received by the output device.

• **INTR (Interrupt request**) – Thus an output signal that can be used to interrupt the CPU when an output device acknowledges the data received from the CPU.

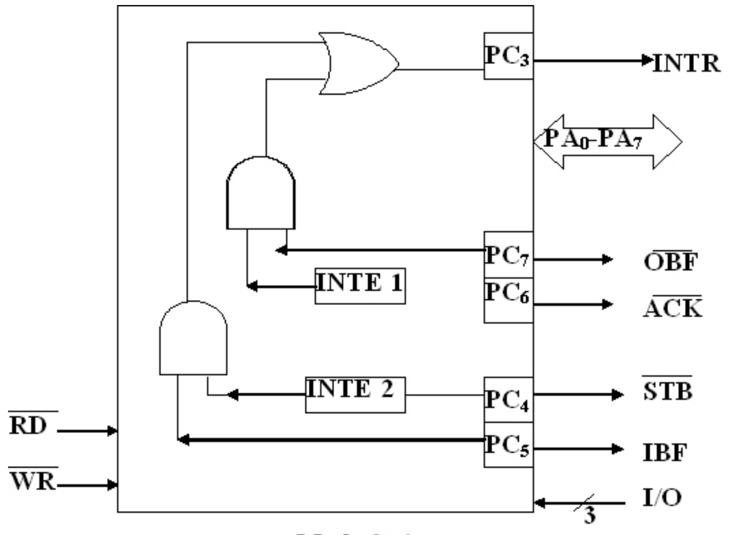
Output control signal definitions Mode 1



Mode 1 Control Word Group A Mode 1 Control Word Group B

MODE 2: Bi-directional I/O data transfer:

- This mode allows bidirectional data transfer over a single 8-bit data bus using handshake signals.
- In this mode, Port A can be configured as the bidirectional port and Port B is either in Mode 0 or Mode 1.
- Port A uses 5 signals from Port C as handshake signals for data transfer. The remaining 3 signals from Port C can be used either as simple I/O or as handshake for Port B.



Mode 2 pins

****

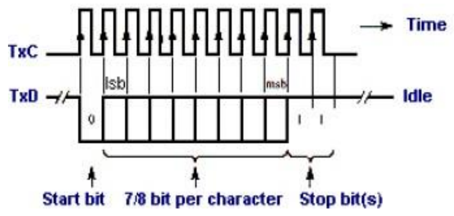
3.Explain in detail about Serial Communication Interface

Serial I/O Interfacing:

The MPU (Microprocessor Unit) selects the peripheral through chip select and uses the control signals. Read to receive data and write to transmit data.

Transmission format:

- In **synchronous format**, receiver and transmitter are synchronized with the same clock and a block of characters are transmitted along with the synchronization information. This format is generally used for high speed transmission (more than 20 Kbits/second)
- The **asynchronous format** is character oriented. Each character carries the information of the start and stop bits. Transmission starts with one start bit (low) followed by a character , and one or two stop bits (high). It is used in low speed transmission less than 20Kbits/second.



Communication Modes:

Simplex - Data are transmitted in only one direction.

Example: Transmission from a microcomputer to a printer.

Duplex - Data flow in both direction

- **Half Duplex** - If the transmission goes one way at a time it is called half duplex. Ex.: walky-talky
- **Full Duplex** – If both transmitting and receiving signals goes simultaneously, it is called full duplex. Example: Transmission between computers.

Rate of transmission

The rate at which the bits are transmitted is called bits/second or Baud rate

For example 1200 baud = 1200 bits/second

It indicates1200 bits are transmitted in a second. For 1 bit it takes 1/1200 =0.83 ms.

Programmable serial Communication Interface (8251):
Programmable serial interface

The 8251 is a programmable USART (Universal Synchronous Asynchronous Receiver Transmitter) is designed for Synchronous and Asynchronous serial communication packaged in a 28 pin DIP.

The 8251 receives parallel data from the CPU and transmits serial data after conversion. This device also receives serial data from the outside and transmits parallel data to the CPU after conversion.

Features of 8251

- Supports both synchronous and asynchronous modes of operation
- Synchronous baud rate – 0 to 64 K baud
- Asynchronous baud rate – 0 to 19.2 K baud
- Contains full duplex double buffered system
- Provides error detection to detect parity and framing errors
- 28-pin DIP package, TTL compatible
- Single +5V supply

Block diagram of 8251:

- Intel 8251 A is a programmable Serial Communication interface IC. It is available in 28 pin Dual-In-Line package. It is used for synchronous & asynchronous serial data communication. The functional block diagram is shown below. It consists of 5 sections namely,
    - Data bus buffer
    - Read/Write control logic
    - Modem control
    - Transmitter section
    - Receiver section

Data Bus Buffer:

- It is used to temporarily store the data which is to be transmitted (or) received. It consists of D_0 - D_7 signals.

Read/Write Control Logic:

- It consists of 3 registers namely data bus buffer, control register and status register.
- Reset, CLK, C/D, RD, WR, CS signals are associated with this block, If C/D is high, the control register is selected for writing control word.
- If C/D is low, then, the data buffer is selected for read/write operation.
- CS signal means chip select signal. It is generated by using unused address lines of processor. If it is low, then the chip is activated. If Reset signal is high, then 8251 is forced to enter into the idle mode.
- CLK signal is used for 8251 to communicate with CPU.
- RD and WR signal are used for read & write operations.

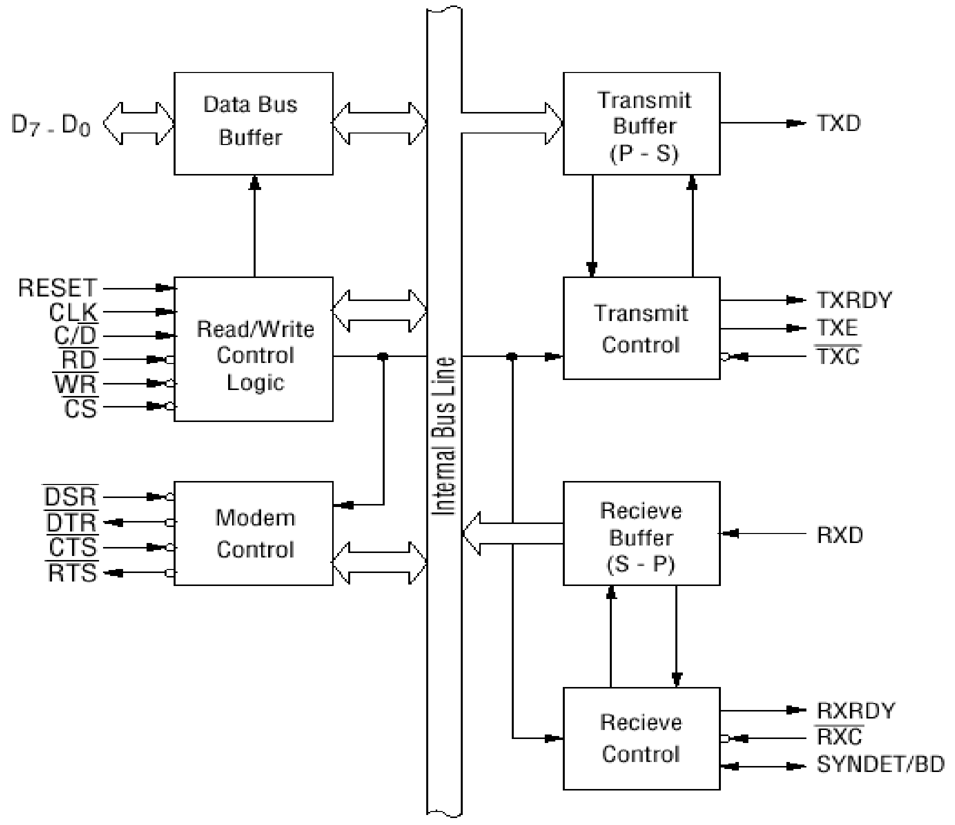


Fig: BLOCK DIAGRAM OF 8251

MODEM Control

- This block is used to interface MODEM to 8251. It is used to provide data communication through MODEM over the telephone cable.

Transmitter Section

- The data which is to be transmitted is given by using D0 - D7 signals to the data bus buffer. Then, the data is transferred to the Transmit Buffer. Here, the parallel data is converted to the serial data. It is transmitted by using the signal TXD.
- This section consists of 2 registers namely transmit buffer register & output register. Transmit buffer is used to hold the 8-bit data & output register is used to convert parallel data into serial data. If output register is empty, then the data is transferred from buffer register to output register.
- If buffer register is empty, then TXRDY signal is asserted high. If output register is empty, then TXEMPTY signal is asserted high.
- TXC signal is used to control the rate of transmission.

Receiver Section

- This section receives serial data from the signal RXD and converts that data into parallel data. It consists of 2 registers namely input register & buffer register.
- Input register receives the serial data & convert it into parallel. Buffer register is used to hold the previous converted data. If input register loads parallel data into buffer register, then, the RXRDY signal is asserted high.

- If RXD signal is low for a half of bit time, then it is assumed as start bit. So, following bits are loaded into the buffer register.
- If RXC signal is used to control the rate of reception.
- During synchronous mode, the signal SYNDET/BRKDET is used to indicate the reception of synchronous character.
- During asynchronous mode, SYNDET/BRKDET signal is used to indicate the break in the data transmission.

Pin Description:

D_0 to D_7 (Data bus Buffer)

- This is bidirectional data bus which receives control word and transmits data from the CPU and sends status words and received data to CPU.

RESET (Input terminal)

- A "High" on this input forces the 8251 into "reset status." The device waits for the writing of "mode instruction."

CLK (Input terminal)

- CLK signal is used to generate internal device timing. CLK signal is independent of RxC or TxC.

WR (Write)

- This is the "active low" input terminal which receives a signal for writing transmit data and control words from the CPU into the 8251.

RD (Read)

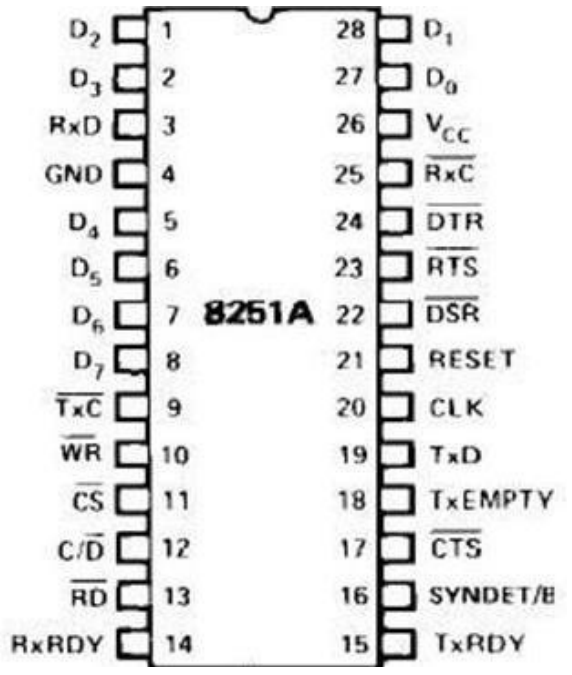
- This is the "active low" input terminal which receives a signal for reading receive data and status words from the 8251.

C/D (Control/Data)

- If C/D = low, data will be accessed. If C/D = high, command word or status word will be accessed.

CS (Chip Select)

- This is the "active low" input terminal which selects the 8251 at low level when the CPU accesses.



SYNDET/BD (Input or output terminal)

- This is a terminal whose function changes according to mode. In "internal synchronous mode." this terminal is at high level, if sync characters are received and synchronized.
- **DSR (Data set ready)**
  This is an input port for MODEM interface. This is normally used to check if the Data set is ready when communicating with a modem.
- **DTR (Data terminal ready)**
  This is an output port for MODEM interface. It is used to indicate that the device is ready to accept data when the 8251 is communicating with a modem.
- **CTS (clear to send)**
  This is an input terminal for MODEM interface which is used for controlling a transmit circuit. Data is transmittable if the terminal is at low level.
- **RTS (Request to send data)**
  This is an output port for MODEM interface. It is used to indicate the MODEM that the receiver is ready to receive a data byte from the MODEM.

Control Register

The 16 bit register for a control word consists of two independent bytes. The first byte is called the mode instruction and the second byte is called the command instruction. This register can be accessed as an output port when the C/D pin is high**.**

Status Register

This input register checks the ready status of a peripheral. This register is addressed as an input port when the C /D is high. It has the same port address as the control register.

4. Draw the block diagram of 8279 Keyboard/Display controller and explain how to interface the Hex Key Pad and 7-segment LEDs using 8279. (April 2010)

- It simultaneously drives the display of a system and interfaces a keyboard with the microprocessor.
- The keyboard display interface scans the keyboard to identify if any keys has been pressed and sends the code of the pressed key to the microprocessor.
- It also transmits the data received from microprocessor to the display device.

PIN DIAGRAM OF 8279:

DATA BUS (D7-D0)

- All data and commands between the microprocessor and 8279 are transmitted on these lines.

RD (read):

- Microprocessor reads the data/ status from 8279.

WR (write):

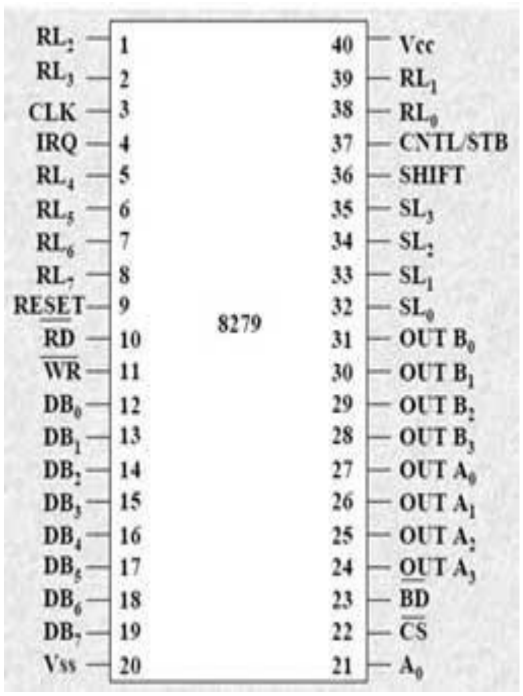
- Microprocessor writes the data to 8279

A0:

- A high signal on this line indicates that the word is a command or status. A low signal indicates the data.

RESET:

- High signal in this pin resets the 8279. After being reset, the 8279 is placed in the following modes 16 x 8 – bit character display – left entry
- Two key lock out



CS (Chip Select):

- A low signal on this input pin enables the communication between 8279 and the microprocessor.

IRQ (Interrupt Request):

- The interrupt line goes low with each FIFO/sensor RAM reads and returns high if there still information in the RAM

SL0-SL3:

- The scan lines which are used to scan the key switch or sensor matrix and the displays digits. These lines can be either encoded (1 of 16) or decoded (1 of 4)

RL0-RL7:

- Input return lines which are connected to the scan lines through the keys or sensor switches.

SHIFT:

- It has an active internal pull-up to keep it high until a switch closure pulls it low.

CNTL/STB:

- For keyboard mode, this line is used as a control input and stored like status on a key closure.
- The line is also the strobed line to enter the data into the FIFO in the strobed input.

OUT A0 – OUT A3, OUT B0 – OUT B3:

- These two ports are the outputs for the 16x4 display refresh registers. These two ports may also be considered as one 8 – bit port. The two 4 – bit ports may be blanked independently.

BD:

- This output is used to blank the display digit switching or by a display banking command.

BLOCK DIAGRAM OF 8279:

The 8279 has the following four sections.

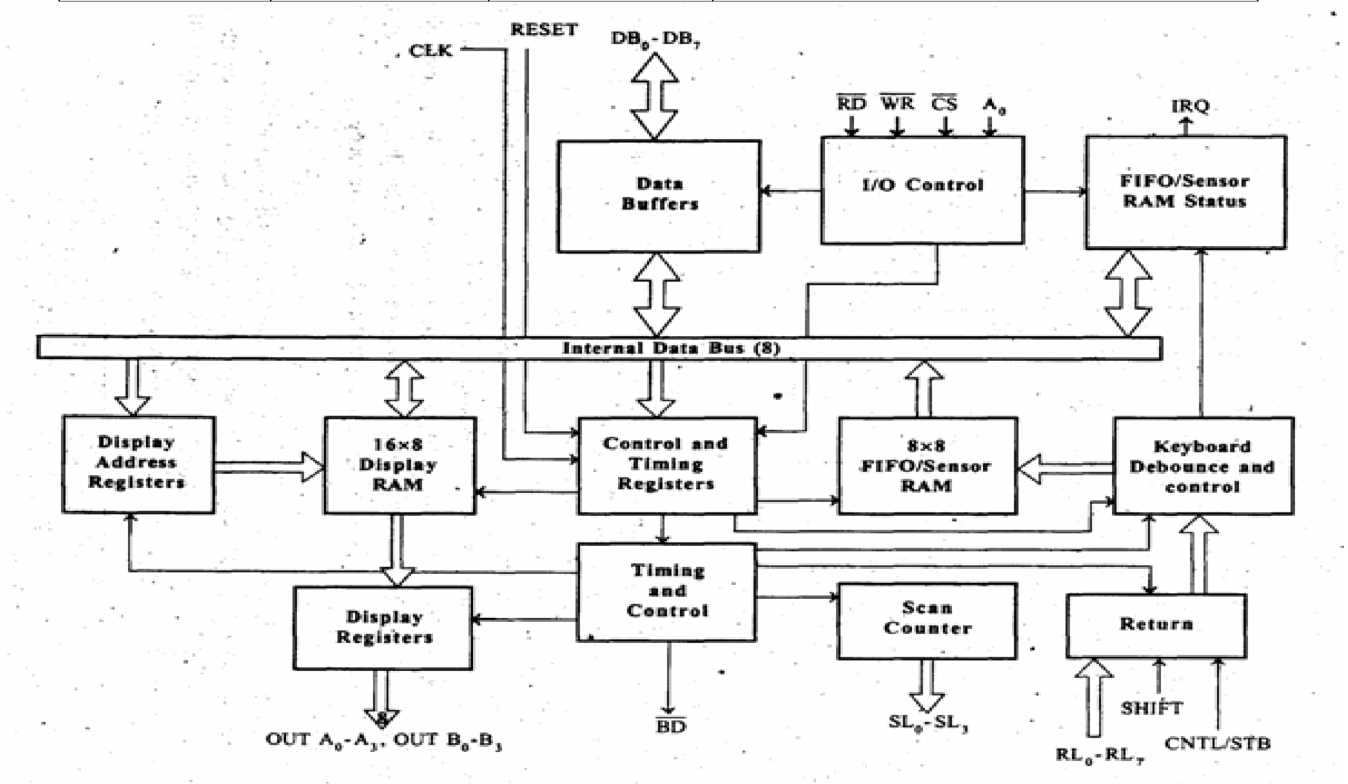
- CPU interface section
- Keyboard section
- Scan section
- Display section

CPU INTERFACE SECTION:

- This section has bi-directional data buffer (DB0 –DB7), I/O control lines (RD, WR, CS, A0) and Interrupt Request lines (IRQ).
- The A0 signal determines whether transmit/receive control word or data is used.

An active high in line IRQ is generated to interrupt the microprocessor whenever the data is available.

A0	RD	WR	Operation
0	0	0	MPU writes the data is 8279
0	0	1	MPU reads the data from 8279
1	1	0	MPU writes control word to 8279
1	0	1	MPU read status word from 8279



KEYBOARD SECTION:

- This section has keyboard debounce & control, 8X8 FIFO/sensor RAM, 8 return lines (RL0 – RL7) and CNTL/STB and shift lines.
- In the keyboard debounce and control unit, keys are automatically debounced and the keyboard can be operated in two modes.
    - Two keys lock out
    - N – key roll over

Two-Key lockout mode:

- If two keys are depressed within the debounce cycle, it is a simultaneous depression. Neither key will be recognized until one of the key is released. The final key released will be recognized and entered.

N-Key Rollover mode:

- In this mode, each key depression is treated independently. If simultaneous depression occurs, then keys are recognized and entered according to the order the keyboard scan found them.
- The 8X8 FIFO/sensor RAM consists of 8 registers that are used to store eight keyboard entries.
- The return lines (RL0-RL7) are connected to eight columns of keyboard.
- The status of shift and CNTL/STB lines are stored along with the key closure.

SCAN SECTION:

- This section has scan counter and four scan lines (SL0 – SL3).
- These lines are decoded (by using 4 to 16 decoder) to generate 16 scan lines.
- Generally SL0 – SL3 are connected with the rows of a matrix keyboard.

DISPLAY SECTION:

- This section has two groups of outputs lines A0 – A3 and B0 – B3. These lines are used to send data to display drivers.
- BD line is used blank the display. It also has 16X8 displays RAM.

<u>Modes of operations of 8279</u>

1. Input (Keyboard) modes
2. Output (Display) modes

Keyboard modes

- **Scanned keyboard mode with N key rollover**

In this mode, each key depression is treated independently. When a key is pressed, the debounce circuit waits for 2 keyboards scans and then checks whether the key is still depressed. If it is still depressed, the code is entered in FIFO RAM

- **Scanned keyboard mode with 2 key lock out.**

It Prevents 2 keys from being recognized if pressed simultaneously. If two keys are pressed within a debounce cycle (simultaneously), no key is recognized till one of them remains closed, and the other is released. The last key that remains depressed is considered as single valid key depression.

Display modes:

Left entry mode
The data is entered from the left side of the display unit.
Right entry mode
The first entry to be displayed is entered on the rightmost display.

Programming the Keyboard Interface :

- Before any keystroke is detected, the 8279 must be programmed.
- The first 3 bits of the number sent to the control port (11H) select one of the 8 different control words.

Command Words of 8279

a) **Keyboard Display mode set**

The format of the command word is to select different modes of operation of 8279

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_0
0	0	D	D	D	K	K	K	1

D_7	D_6	D_5	*Function*	*Purpose*
0	0	0	Mode set	Selects the number of display positions, type of key scan...
0	0	1	Clock	Programs internal clk, sets scan and debounce times.
0	1	0	Read FIFO	Selects type of FIFO read and address of the read.
0	1	1	Read Display	Selects type of display read and address of the read.
1	0	0	Write Display	Selects type of write and the address of the write.
1	0	1	Display write inhibit	Allows half-bytes to be blanked.
1	1	0	Clear	Clears the display or FIFO
1	1	1	End interrupt	Clears the IRQ signal to the microprocessor.

D	D	Display modes
0	0	Eight 8-bit character Left entry
0	1	Sixteen 8-bit character left entry
1	0	Eight 8-bit character Right entry
1	1	Sixteen 8-bit character Right entry

K	K	K	Keyboard modes
0	0	0	Encoded Scan, 2 key lockout (Default after reset)
0	0	1	Decoded Scan, 2 key lockout
0	1	0	Encoded Scan, N- key Roll over
0	1	1	Decoded Scan, N- key Roll over
1	0	0	Encode Scan, N- key Roll over
1	0	1	Decoded Scan, N- key Roll over
1	1	0	Strobed Input Encoded Scan
1	1	1	Strobed Input Decoded Scan

Control Word Description

First three bits given below select one of 8 control registers (opcode)

- ***<u>000DDMMM</u>***

***Mode set*: Opcode 000.**
DD sets displays mode.
MMM sets keyboard mode
DD field selects either**:**

- 8- or 16-digit display
- Whether new data are entered to the rightmost or leftmost display position.

b)Programmable clock
The clock for operation of 8279 is obtained by dividing the external clock input signal by a programmable constant called prescaler**.**

- **<u>001PPPPP</u>**

- The clock command word programs the internal clock driver**.**
- **The code PPPPP,** is a prescalar that divides the clock input pin (CLK) to achieve the desired
- operating frequency, e.g. 100 KHz requires 01010_2 .

(c) Read FIFO/Sensor RAM

- **<u>010 AI X AAA</u>**

The read FIFO control word selects the address (AAA) of a keystroke from the FIFO buffer (000 to 111).
X - don't care
AI selects auto-increment for the address

d) Read Display RAM
This command enables a programmer to read the display RAM data.

- **<u>011 AI AAAA</u>**

The display read control word selects the 4 bit address AAAA points to the 16 byte display RAM position that is to be read.
AI selects auto-increment for the address.

e) Write Display RAM

- **<u>100 AI AAAA</u>**

The display write control word selects the 4 bit address AAAA points to the 16 byte display RAM positions that is to be written.
Display. Z selects auto-increment so subsequent writes go to subsequent display positions.

f)Display with inhibit blanking

- **<u>1010WWBB</u>**

The display write inhibit control word inhibits writing to either the leftmost 4 bits of the display (left W) or rightmost 4 bits (right W).
BB works similarly except that they blank (turn off) half of the output pins.

g) Clear Display RAM

- **<u>1100CCFA</u>**

The clear control word clears the display, FIFO or both
Bit F clears FIFO and the display RAM status, and sets address pointer to 000.
If CC are 00 or 01, all display RAM locations become 00000000.
If CC is 10, --> 00100000, if CC is 11, --> 11111111.

h) End Interrupt/Error mode set

- **<u>1110E000</u>**

End of Interrupt control word is issued to clear IRQ pin to zero in sensor matrix mode

•Clock must be programmed first. If 3.0 MHz drives CLK input, PPPPP is programmed to 30 or 11110_2.

•Keyboard type is programmed next. The previous example illustrates an encoded keyboard, external decoder used to drive matrix.

•Program the operation of the FIFO.Once programmed never reprogrammed done, until a procedure is needed to read prior keyboard codes .

To determine if a character has been typed, the FIFO status register is checked.

When the control port is addressed by the IN instruction, the contents of the FIFO status word is copied into register AL:

5.Draw the functional block diagram of 8254 timer and explain the different modes of operation. (April 2010)(Nov/Dec-2013)

Programmable Interval Timer: 8254

The 8254 is a programmable interval timer/counter is used for the generation of accurate time delays ,controlling real-time events such as real-time clock, events counter, and motor speed and direction control under software control.

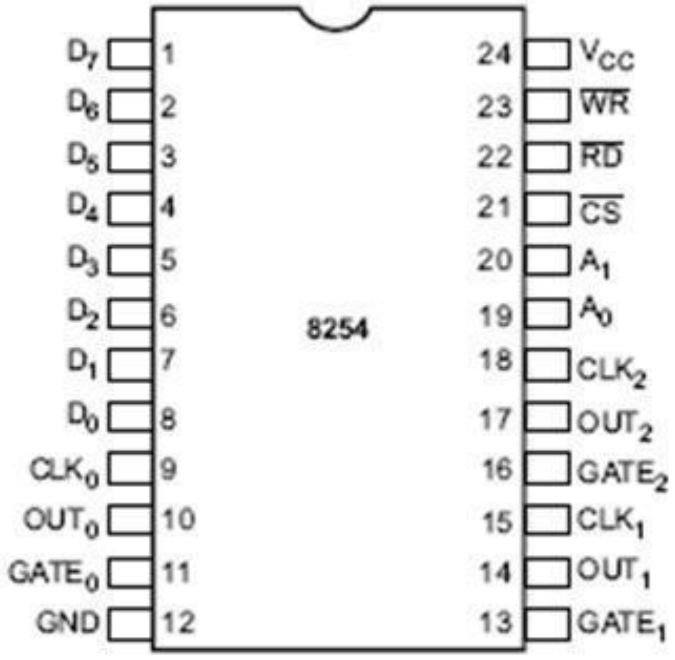
After the desired delay, the 8254 will interrupt the CPU. This makes microprocessor to be free the tasks related to the counting process and can execute the programs in memory, while the timer device may perform the counting tasks. This minimize the Software overhead on the microprocessor.

It consists of three independent 16-bit programmable counters (timers),each with capable of counting in binary or BCD with a maximum frequency of 10MHz.

Some of the other counter/timer functions common to microcomputers which can be implemented with the 8254 are:

- Real time clock
- Event-counter
- Digital one-shot
- Programmable rate generator
- Square wave generator
- Binary rate multiplier

PIN DIAGRAM OF 8254:



PIN DESCRIPTION:

A1	A2	SELECTION
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Counter 3

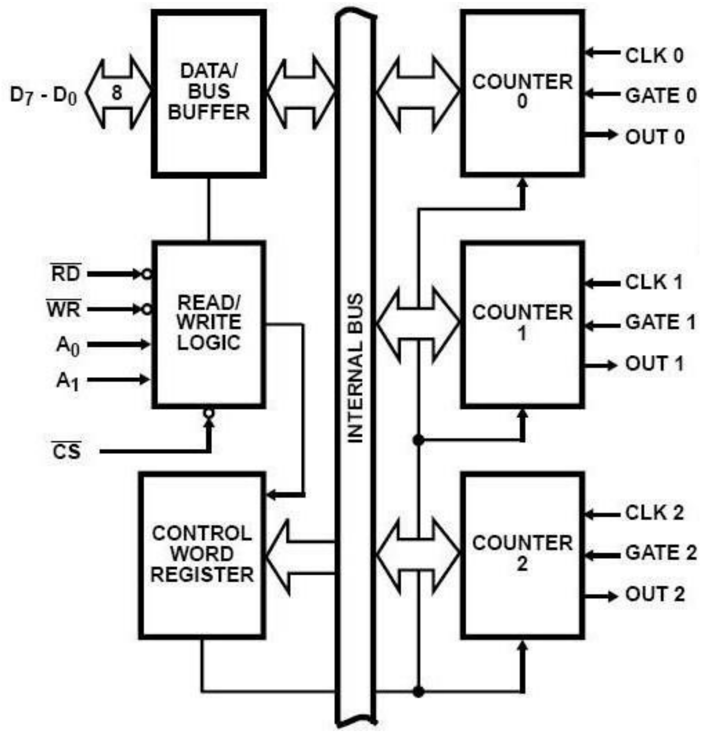
BLOCK DIAGRAM OF 8254:

DATA BUS BUFFER:

- This 3- state, bi-directional, 8-bit buffer is used to interface the 8254 to the system bus.

READ/WRITE LOGIC:

- The Read/Write logic accepts inputs from the system bus and generates control signals for the other functional blocks of the 8254.
- A1 and A0 select one of the three contents counters or the control word register to be read from/written into.
- A "low" on the RD input tells the 8254 that the CPU is reading one of the counters.
- A "low" on the WR input tells the 8254 that the CPU is writing either a control word or an initial count.
- Both RD and WR are qualified by CS; RD and WR are ignored unless than 8254 has been selected by holding CS low.



CONTROL WORD REGISTER:

- The control word register is selected by the read/write logic when A1, A0=11.
- If the CPU then does a write operation to the 8254, the data is stored in the control word register and is interpreted as a control word used to define the operation of the counters.
- The control word register can only be written to; status information is available with theRead-Back command.

COUNTER 0, COUNTER 1, COUNTER 2:

- Each is a 16 bit down counter
- The counters are fully independent. Each counter may operate in a different mode.
- Each counter has a separate clock input, count enable (gate) input lines and output lines.
- The control word register is not a part of the counter itself, but its contents determine how the counter operates.

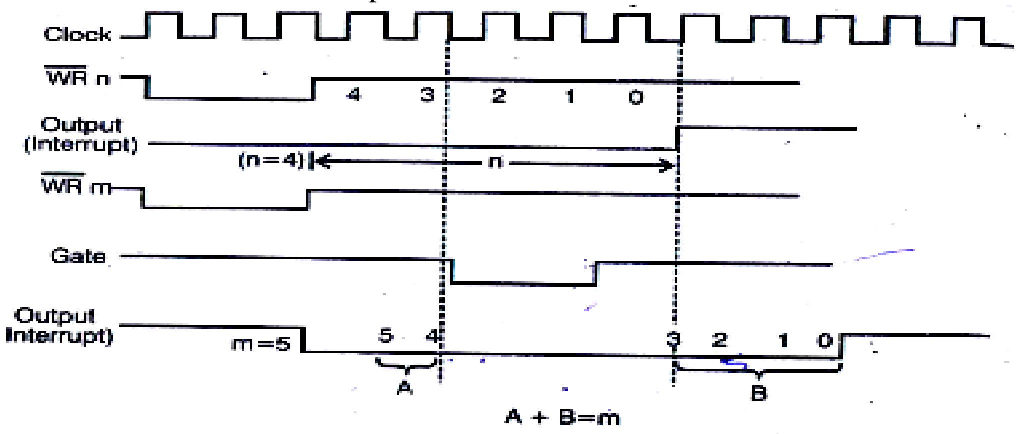
OPERATIONAL MODES OF 8254:

- The 8254 can operate in six operating modes. They are,

Mode 0: Interrupt on Terminal count:

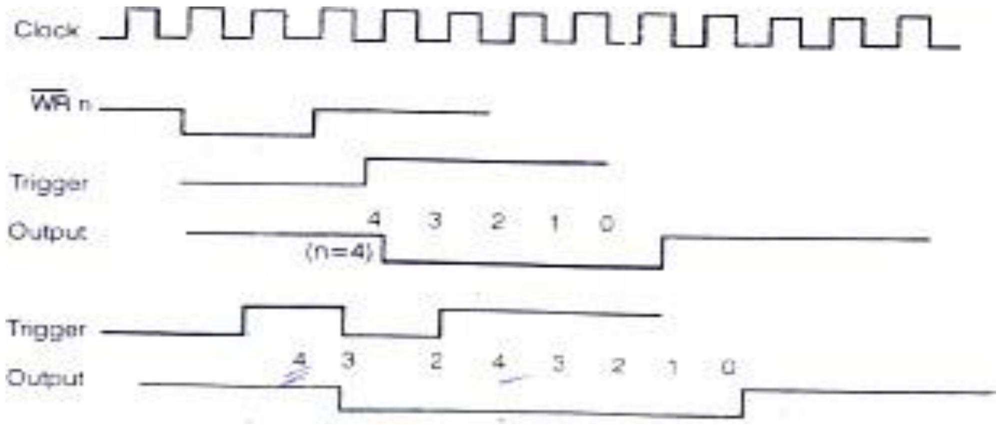
- Mode 0 is typically used for event countering.
- After the control word is written OUT is initially low, and will remain low until the counter reaches zero.
- OUT then goes high and remains high until a new count or a new mode 0 control word is written into the counter.

    - GATE = 1 enables counting;
    - GATE = 0 disables counting. GATE has no effect on OUT.
- After the control word and initial count (say n=4, m=5) are written to a counter, the initial count will be loaded on the next CLK pulse.
- This CLK pulse does not decrement the count. So far an initial count of N, OUT does not go high until N+1 CLK pulses after the initial count is written.
- This mode can be used as an interrupt.



Mode 1 – Hardware Retriggerable one-shot:

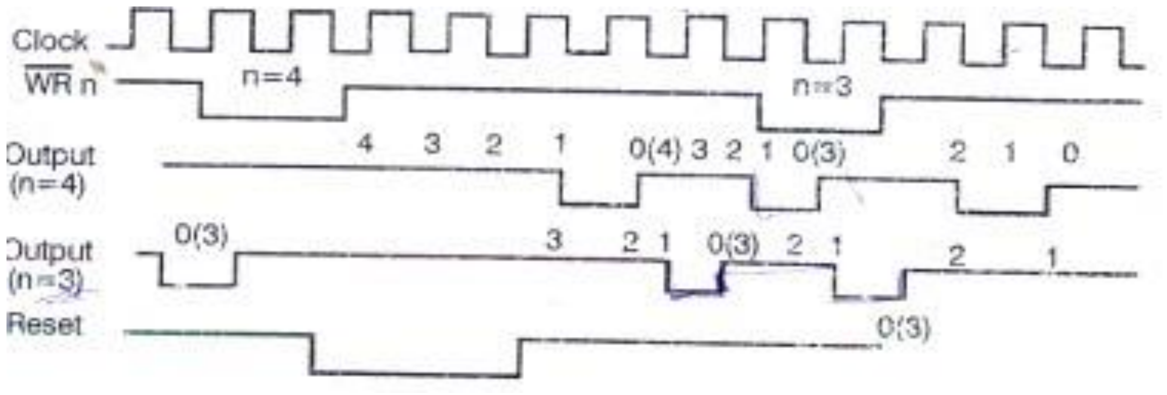
- OUT will be initially high. OUT will go low on the CLK pulse following a trigger to begin the one-shot pulse, and will remain low until the counter reaches zero.
- OUT will then go high and remain high until the CLK pulse after the next trigger. Thus generating a one- shot pulse.
- After writing the control word and initial count, the counter is armed. A trigger results in loading the counter and setting OUT low on the next CLK pulse, the starting the one-shot pulse.
- An initial count of N will result is a one-shot pulse 'N' CLK cycles in duration.



Mode 2: Rate generator

- This mode function like a device – by – N counter
- It is typically used to generate a real time clock interrupt.
- OUT will initially be high. When the initial count has decremented to 1, OUT goes low for one CLK pulse.

- Count and the process are repeated.



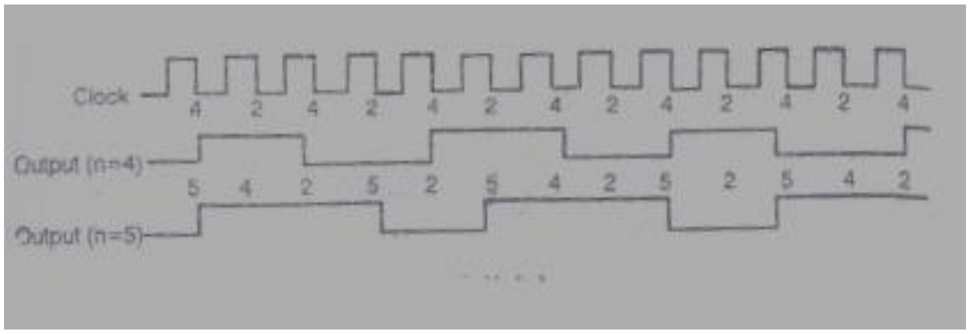
- **Mode 3: Square wave mode:**
- Mode 3 is typically used for baud rate generation.
- Mode 3 is similar to mode 2 except for the duty cycle of OUT, OUT will initially be high.
- When half the initial count has expired, OUT goes low for the reminder of the count.
- Mode 3 is periodic; the sequence above is repeated indefinitely.
- An initial count of N results in a square wave with a period of N CLK cycles.
- Mode 3 is implemented as follows:

EVEN COUNTS:

- OUT is initially high. The initial count is loaded on 1 CLK pulse and then is decremented by two on succeeding CLK pulses.
- When the count expires OUT changes value and the counter is reloaded with the initial count.
- The above process is repeated indefinitely.

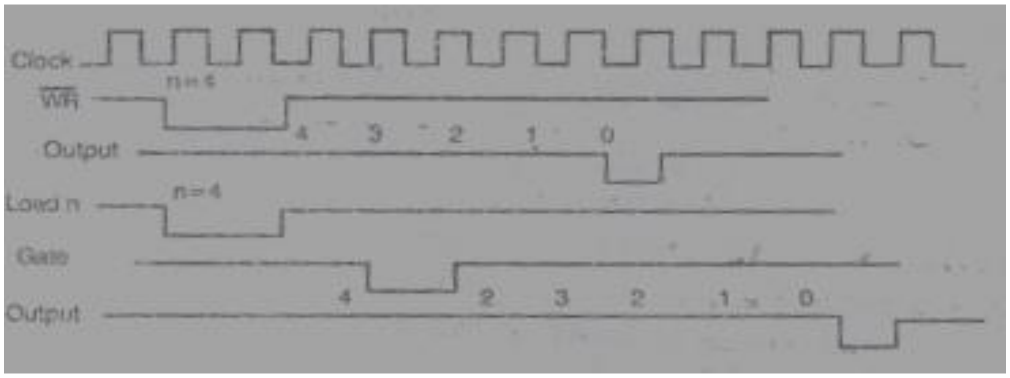
ODD COUNTS:

- For odd counts, OUT will be high for (N+1)/2 counts and low for (N-1)/2 counts.



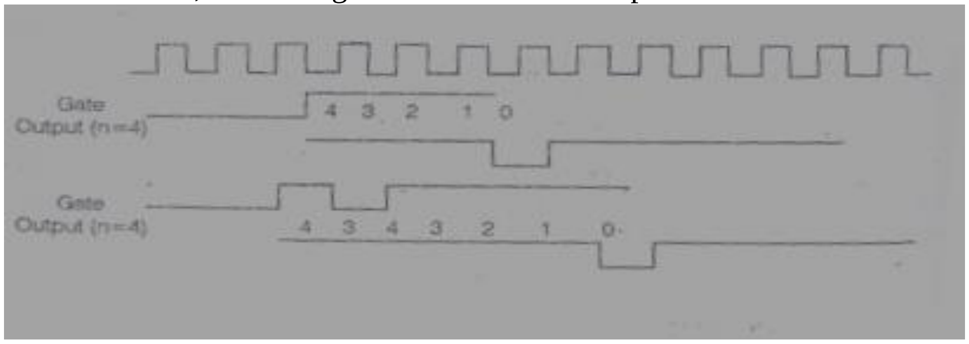
Mode 4: Software triggered Strobe

- The output goes high on setting the mode.
- After terminal count, the output goes low for one clock period and then goes high again.
- In this mode the OUT is initially high; it goes low for clock period at the end of the count.
- The count must be reloaded for subsequent outputs.

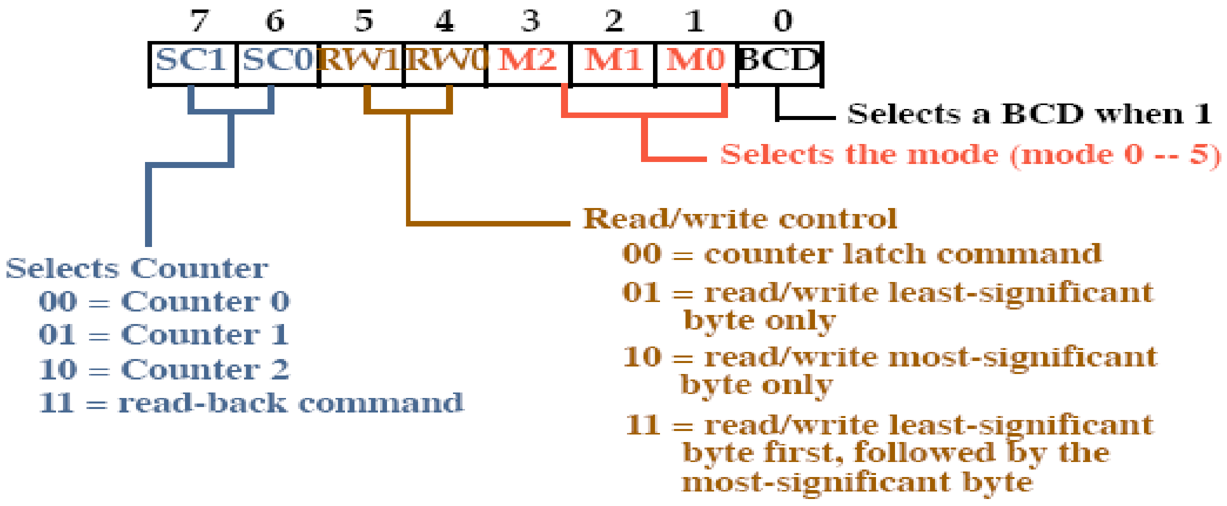


Mode 5: hardware triggered strobe

- This mode is similar to mode 4, but a trigger at the gate initiates the counting.
- This mode is similar to mode 4, except that it is triggered by the rising pulse at the gate.
- Initially the OUT is high and when the gate pulse is triggered from low to high, the count begins, at the end of the count; the OUT goes low for one clock period.



Command word of 8254

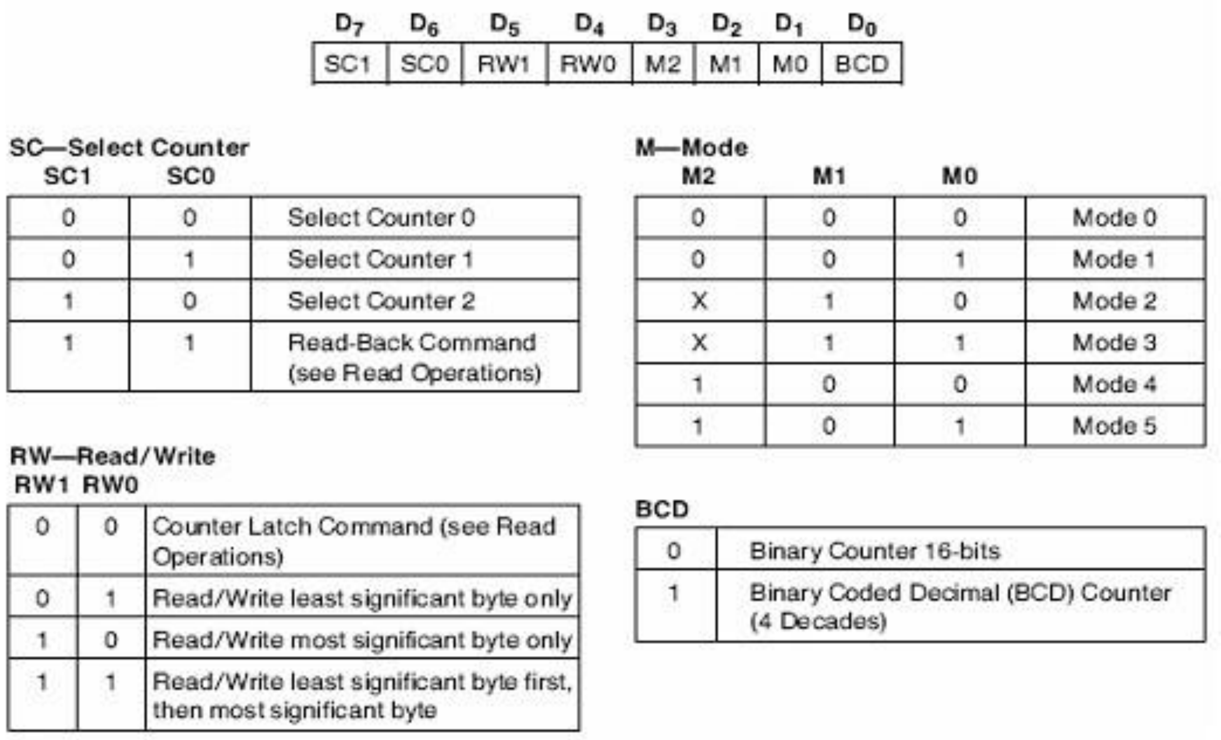


Selects a BCD when 1

Selects the mode (mode 0 -- 5)

Read/write control
00 = counter latch command
01 = read/write least-significant byte only
10 = read/write most-significant byte only
11 = read/write least-significant byte first, followed by the most-significant byte

Selects Counter
00 = Counter 0
01 = Counter 1
10 = Counter 2
11 = read-back command



D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC—Select Counter

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Read-Back Command (see Read Operations)

M—Mode

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

RW—Read/Write

RW1	RW0	
0	0	Counter Latch Command (see Read Operations)
0	1	Read/Write least significant byte only
1	0	Read/Write most significant byte only
1	1	Read/Write least significant byte first, then most significant byte

BCD

0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

NOTE: Don't care bits (X) should be 0 to insure compatibility with future Intel products.

Figure 7. Control Word Format

Each counter may be programmed with a count of 1 to FFFFH. Minimum count is 1 all modes except 2 and 3 with minimum count of 2.Each counter has a program control word used to select the way the counter operates. If two bytes are programmed, then the first byte (LSB) stops the count, and the second byte (MSB) starts the counter with the new count.

6.Discuss in detail about Programming and interfacing 8253

There may be two types of write operations in 8253

i) Writing control word into a control word register

ii) Writing a count value into a count register.

iii) The control word register accepts data from the data buffer and initializes the counter as required.

iv) The control word register contents are used for

a) Initializing operating modes(Mode 0 to Mode 4)

b) Selection of counters (Counter0 to counter3)

c) Choosing binary/BCD counters

d) Loading the counter register

<u>Read Operations</u>

There are three possible methods for reading the counters:

- a simple read operation
- the Counter Latch Command
- the Read-Back Command

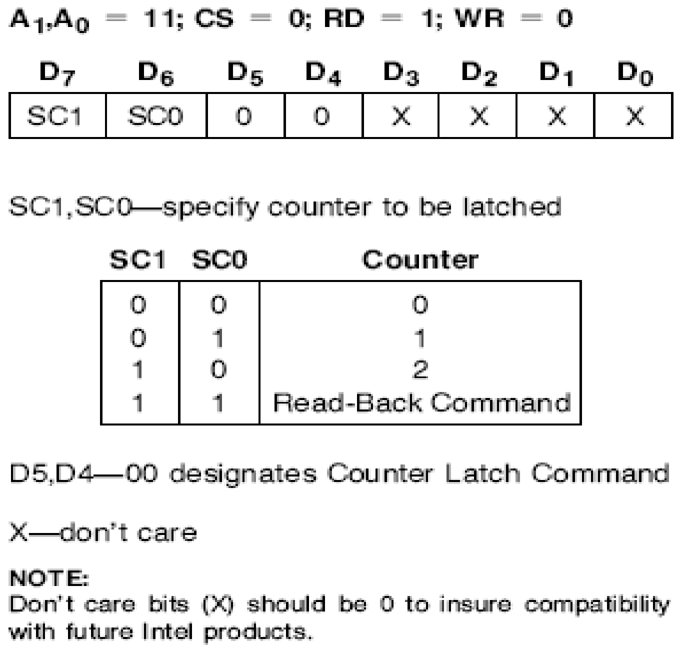
<u>Simple read operation</u> :

- The Counter which is selected with the A1, A0 inputs, the CLK input of the selected Counter must be inhibited by using either the GATE input or external logic. Otherwise, the count may be in the process of changing when it is read, giving an undefined result

<u>Counter Latch Command:</u>

- SC0, SC1 bits select one of the three Counters
- Two other bits, D5 and D4, distinguish this command from a Control Word

If a Counter is latched and then, sometime later, latched again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued**.**



A_1,A_0 = 11; CS = 0; RD = 1; WR = 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	0	0	X	X	X	X

SC1,SC0—specify counter to be latched

SC1	SC0	Counter
0	0	0
0	1	1
1	0	2
1	1	Read-Back Command

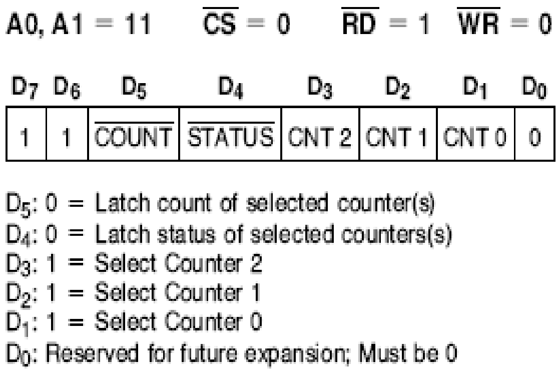
D5,D4—00 designates Counter Latch Command

X—don't care

NOTE:
Don't care bits (X) should be 0 to insure compatibility with future Intel products.

<u>Read-back control command</u>

- The read-back control, word is used when it is necessary for the contents of more than one counter to be read at a same time.
- Count : logic 0, select one of the Counter to be latched
- Status : logic 0, Status must be latched to be read status of a counter is accessed by a read from that counter

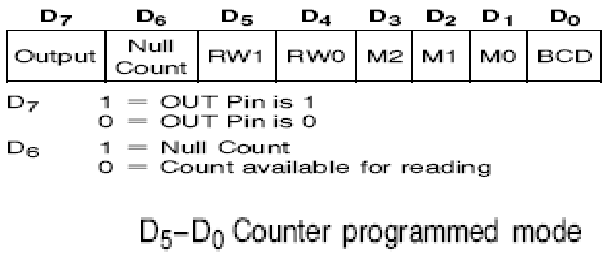

A0, A1 = 11 $\overline{CS}$ = 0 $\overline{RD}$ = 1 $\overline{WR}$ = 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	$\overline{\text{COUNT}}$	$\overline{\text{STATUS}}$	CNT 2	CNT 1	CNT 0	0

D_5: 0 = Latch count of selected counter(s)
D_4: 0 = Latch status of selected counters(s)
D_3: 1 = Select Counter 2
D_2: 1 = Select Counter 1
D_1: 1 = Select Counter 0
D_0: Reserved for future expansion; Must be 0

Status register:

- shows the state of the output pin
- check the counter is in NULL state (0) or not
- how the counter is programmed



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Output	Null Count	RW1	RW0	M2	M1	M0	BCD

D_7	1 = OUT Pin is 1
	0 = OUT Pin is 0
D_6	1 = Null Count
	0 = Count available for reading

D_5–D_0 Counter programmed mode

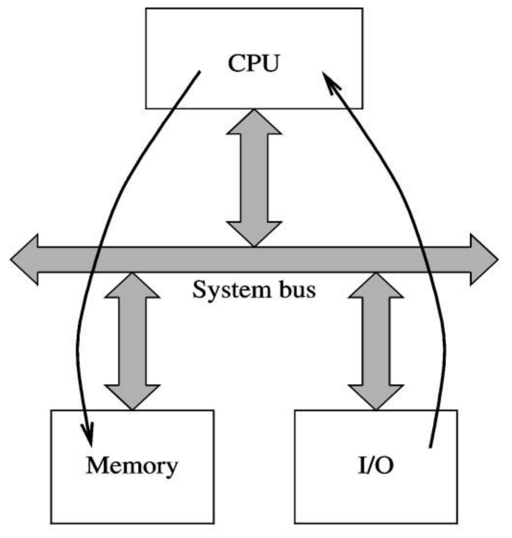
****

7. Explain in detail about Direct Memory Access (DMA Controller 8257)

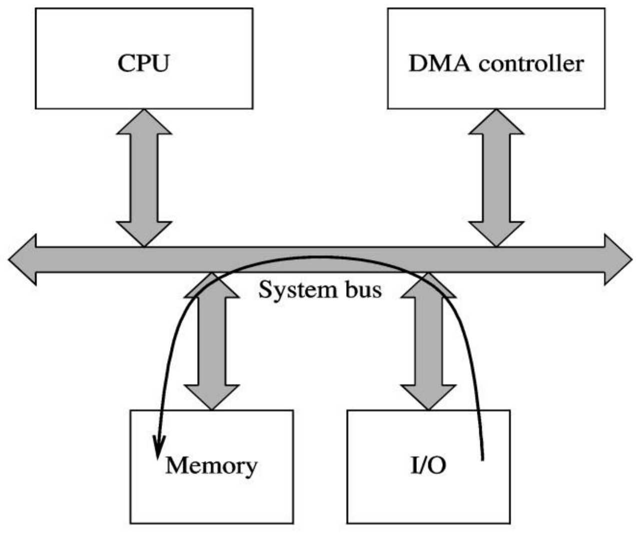
Direct memory access (**DMA**) or DMA mode of data transfer is the fastest amongst all the modes of data transfer. In this mode, the device may transfer data directly to/from memory without any interference from the CPU.

THE **DMA controller (8257)** allows certain hardware subsystems to read/write data to/from memory without microprocessor intervention, allowing the processor to do other work.

The device requests the CPU (through a DMA controller) to hold its data, address and control bus, so that the device may transfer data directly to/from memory. The DMA data transfer is initiated only after receiving HLDA signal from the CPU. For facilitating DMA type of data transfer between several devices, a DMA controller may be used.



(a) Programmed I/O transfer



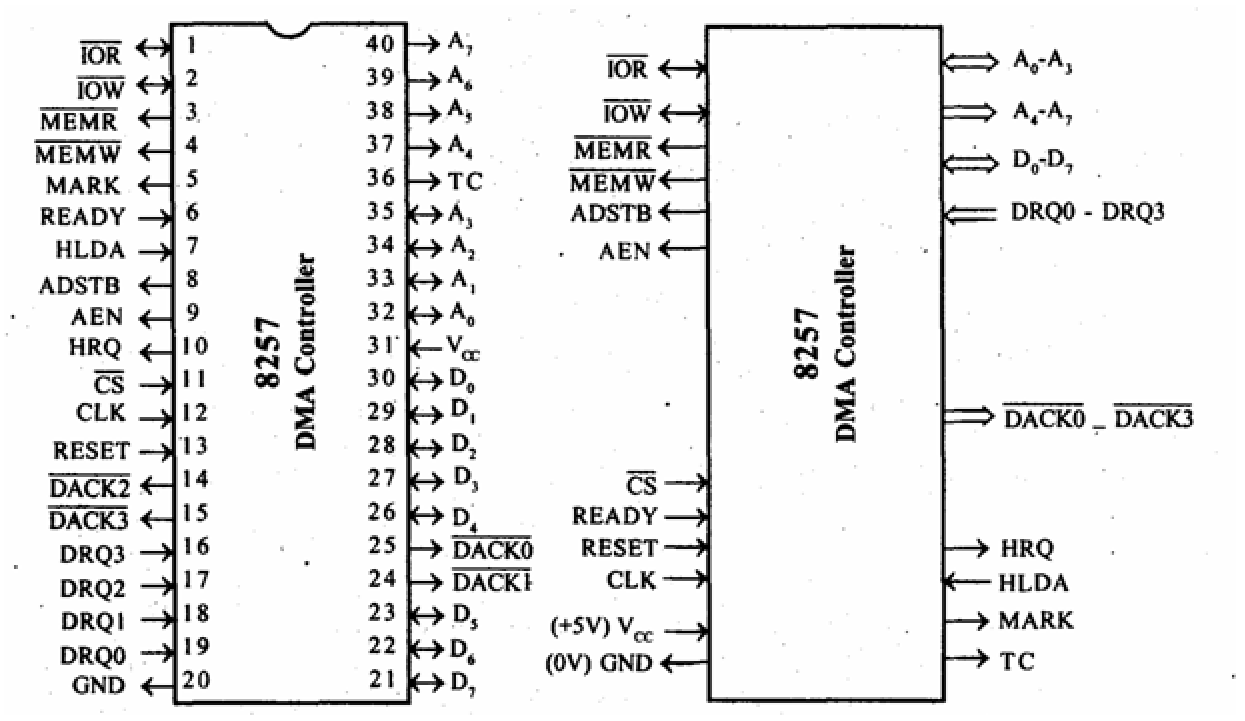
(b) DMA transfer

It is used in disk controllers, video/sound cards etc, or between memory locations. Typically, the CPU initiates DMA transfer, does other operations while the transfer is in progress, and receives an interrupt from the DMA controller once the operation is complete.

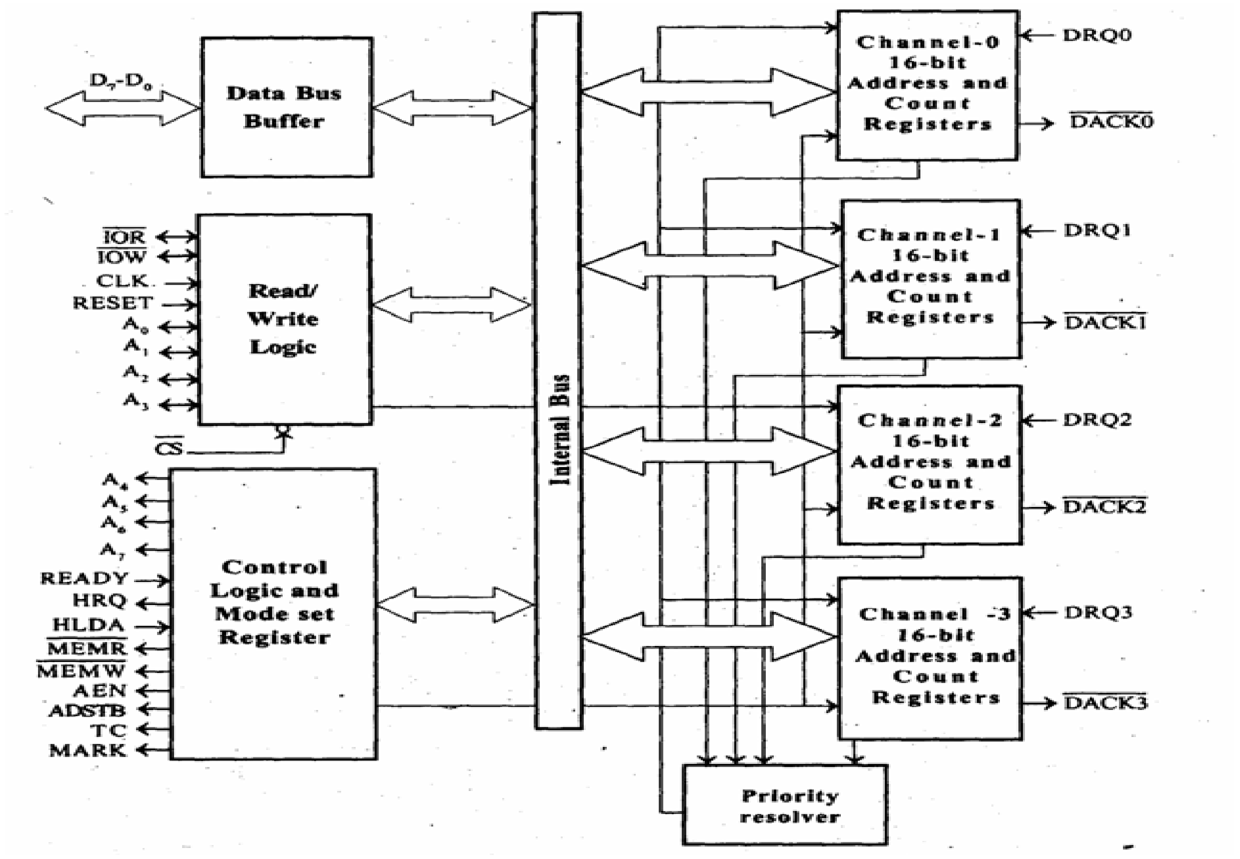
It contains Five main Blocks.

1. **Data bus buffer**
2. **Read/Control logic**
3. **Control logic block**
4. **Priority resolver**
5. **DMA channels.**

Pin diagram of 8257:



Block diagram of 8257:



D7-D0
Data Bus Buffer
Read/ Write Logic
Control Logic and Mode set Register
Internal Bus
Channel-0 16-bit Address and Count Registers
Channel-1 16-bit Address and Count Registers
Channel-2 16-bit Address and Count Registers
Channel -3 16-bit Address and Count Registers
Priority resolver
DRQ0
DACK0
DRQ1
DACK1
DRQ2
DACK2
DRQ3
DACK3
IOR
IOW
CLK
RESET
A0
A1
A2
A3
CS
A4
A5
A6
A7
READY
HRQ
HLDA
MEMR
MEMW
AEN
ADSTB
TC
MARK

DATA BUS BUFFER:

- It contains tri-state, 8 bit bi-directional buffer.
- **Slave mode**, it transfers data between microprocessor and internal data bus.
- **Master mode**, the outputs A8-A15 bits of memory address on data lines (Unidirectional).

READ/CONTROL LOGIC:

- It controls all internal Read/Write operation.
- Slave mode ,it accepts address bits and control signal from microprocessor.
- Master mode, it generates address bits and control signal.

Control logic block:

It contains ,

1. Control logic
2. Mode set register and
3. Status Register.

CONTROL LOGIC:

Master mode,

It control the sequence of DMA operation during all DMA cycles.

It generates address and control signals.

It increments 16 bit address and decrement 14 bit counter registers.

It activate a HRQ signal on DMA channel Request.

Slave mode it is disabled.

D_0 –D7

- it is a bidirectional ,tri state ,Buffered ,Multiplexed data (D0-D7)and (A8-A15).
- In the slave mode it is a bidirectional (Data is moving).

In the Master mode it is a unidirectional (Address is moving)

IOR

- It is active low, tri-state, buffered, Bidirectional lines.
- **In the slave mode it function as a input line**. IOR signal is generated by microprocessor to read the contents 8257 registers.
- **In the master mode it function as a output line.** IOR signal is generated by 8257 during write cycle

IOW

- It is active low, tri-state ,buffered ,Bidirectional control lines.
- **In the slave mode it function as a input line**. IOR signal is generated by microprocessor to write the contents 8257 registers.
- **In the master mode it function as a output line.** IOR signal is generated by 8257 during read cycle

CLK:

- It is the input line, connected with TTL clock generator.
- This signal is ignored in slave mode.

RESET:

- Used to clear mode set registers and status registers

A0-A3:

These are the tri-state, buffer, bidirectional address lines.

In slave mode, these lines are used as address inputs lines and internally decoded to access the internal registers**.**

In master mode, these lines are used as address outputs lines, A0-A3 bits of memory address on the lines.

- It is active low, Chip select input line.
- In the slave mode, it is used to select the chip.
- In the master mode, it is ignored**.**

A4-A7:

These are the tristate, buffer, output address lines.

In slave mode ,these lines are used as address input lines.

In master mode, these lines are used as address outputs lines, A0-A3 bits of memory address on the lines.

READY:

- It is an asynchronous input line.
- **In master mode,**

When ready is high it receives the signal.

When ready is low, it adds wait state between S1 and S3

- **In slave mode,** this signal is ignored.

HRQ:

It is used to **receiving the hold request** signal from the output device.

HLDA:

- It is acknowledgment signal from microprocessor.

MEMR:

- It is active low, tristate, Buffered control output line.
- In slave mode, it is tristated.
- In master mode, it activated during DMA read cycle.

MEMW:

- It is active low, tristate, Buffered control input line.
- In slave mode, it is tristated.
- In master mode, it activated during DMA write cycle.

AEN (Address enable):

- It is a control output line.
- In master mode ,it is high
- In slave mode ,it is low
- Used it isolate the system address, data, and control lines.

ADSTB: (Address Strobe)

- It is a control output line.
- Used to split data and address line.
- It is working in master mode only.
- In slave mode it is ignore.

TC (Terminal Count):

- It is a status of output line.
- It is **activated in master mode only.**
- It is high, it selected the peripheral.
- It is low, it is free and looking for a new peripheral.

MARK:

- It is a **modulo 128 MARK output line**.
- It is activated in master mode only.
- It goes high, after **transferring every 128 bytes of data block.**

DMA controller

- A DMA controller is capable of becoming the bus master and supervising a transfer between an I/O or mass storage interface and memory. While making a transfer, it must be able to place memory address on the bus and send and receive handshaking signals in a manner similar to that of the bus control logic. The purpose of a DMA controller is to perform a sequence of transfers (ie a block transfer) by stealing bus cycles.
- A DMA controller is designed to service one or more I/O mass storage interfaces, and each interface is connected to the controller by a set of conductors. A portion of a DMA controller for servicing a single interface is called a channel.
- The general organization of a one channel DMA controller and its principal connection is shown in figure. In addition to the usual control and status registers, each channel must contain an address register and a byte (or word) count register.
- Initializing the controller consists of filling these registers with the beginning (or ending) address of the memory array that is to be used as a buffer and the number of bytes (words) to be transferred .For an input to memory, each time the interface has data to transfer it makes a DMA request. The controller then makes a bus request and when it receives a bus grant, it puts the contents of the address register on the address bus, sends an acknowledgement back to the interface, and issues I/O read and memory write signals. The interface then puts the data on the data bus and drops its request. When the memory accepts the data it returns a ready signal to the controller, which then increments (or decrements) the address register, decrements the byte (word) count, and drops its bus request.
- Upon the count reaching zero, the process stops and a signal is sent to the processor as an interrupt request or to the interface to notify it that the transfers have terminated. An output is similarly executed except that the controller issues I/O write and memory read signals and the data are transferred in the other direction.

DRQ0-DRQ3 (DMA Request):

- These are the asynchronous peripheral request input signal.
- The request signals are generated by external peripheral device.

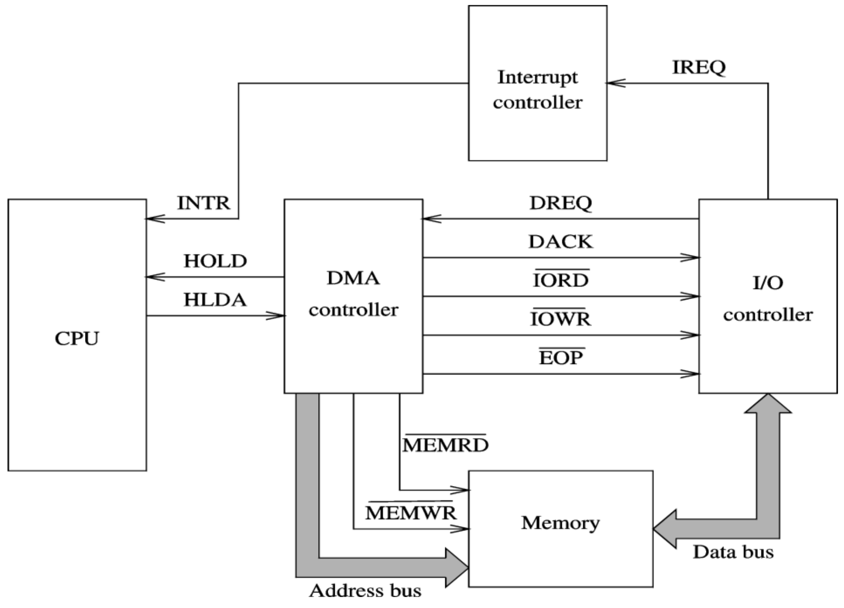
DACK0-DACK3:

- These are the active low DMA acknowledge output lines.
- Low level indicate that, peripheral is selected for giving the information (DMA cycle).

In master mode it is used for chip select

HLDA becomes active to indicate the processor has placed its buses at high-impedance state as can be seen in the timing diagram, there are a few clock cycles between the time that HOLD changes and until HLDA changes

HLDA output is a signal to the requesting device that the processor has relinquished control of its memory and I/O space one could call HOLD input a DMA request input and HLDA output a DMA grant signal



Steps in a DMA operation

- Processor initiates the DMA controller gives device number, memory buffer pointer, called ***channel initialization***
- Once initialized, it is ready for data transfer.
- When ready, I/O device informs the DMA controller .DMA controller starts the data transfer process
- Obtains bus by going through bus arbitration
- Places memory address and appropriate control signals
- Completes transfer and releases the bus
- Updates memory address and count value
- If more to read, loops back to repeat the process
- Notify the processor when done typically uses an interrupt

<u>Modes of DMA operation</u>

Each channel may be put in one of four modes, with its current mode being determined by bits 7 and6 of the channel's mode register. The four possible modes are

Single transfer mode (01)
After each transfer the controller will release the bus to the processor for at least one by cycle, but will immediately begin testing for DREQ inputs and proceed to steal another cycle as soon as a DREQ line becomes active.

Block transfer mode (10)
DREQ need only be active until DACK becomes active, after which the bus is not released until the entire block of data has been transferred.

Demand Transfer mode(00)
This is similar to the block mode except that DREQ is tested after each transfer. If DREQ is inactive, transfers are suspended until DREQ once again becomes active, at which time the block transfer continues from the point at which it was suspended. This allows the interface to stop the transfer in the event that its device cannot keep up.

Cascade Mode (11)

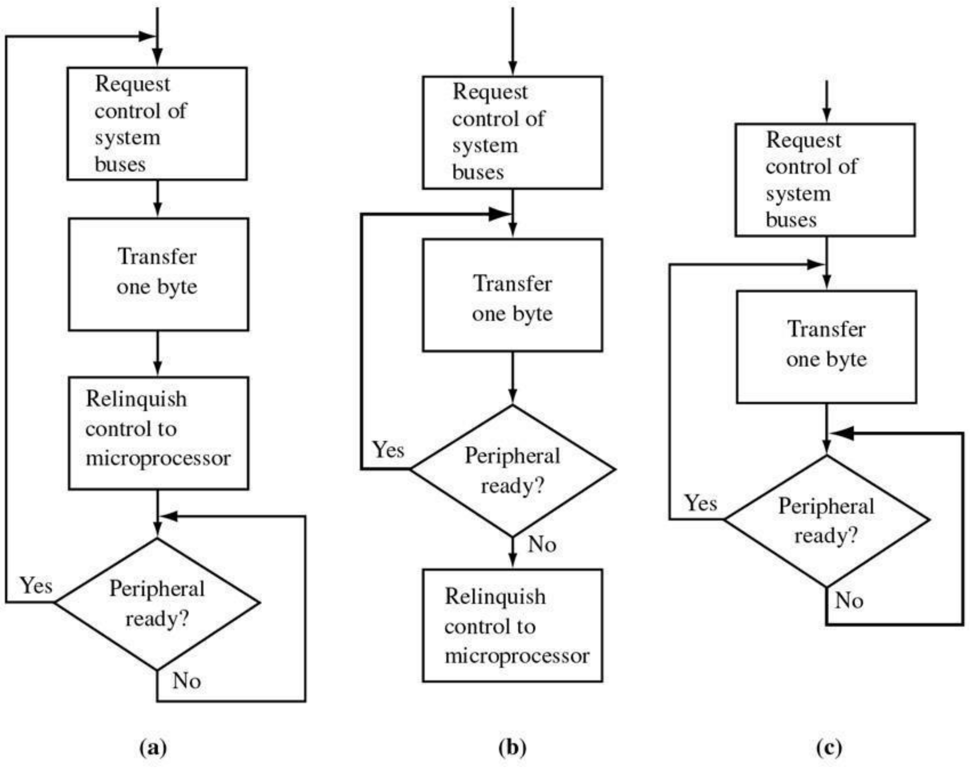
In this mode 8237s may be cascaded so that more than four channels can be included in the DMA subsystem. In cascading the controllers, those in the second level are connected to those in the first level by joining HRQ to DREQ and HLDA to DACK, To conserve space, this mode will not be considered further.

In this mode

Single-cycle mode: DMA data transfer is done one byte at a time

Burst-mode*:* DMA transfer is finished when all data has been moved

a) Byte **b) Burst** **c) Block**


Request control of system buses
Transfer one byte
Relinquish control to microprocessor
Yes
Peripheral ready?
No
(a)
Request control of system buses
Transfer one byte
Yes
Peripheral ready?
No
Relinquish control to microprocessor
(b)
Request control of system buses
Transfer one byte
Yes
Peripheral ready?
No
(c)

8. Write in detail about Analog to digital conversion (ADC)

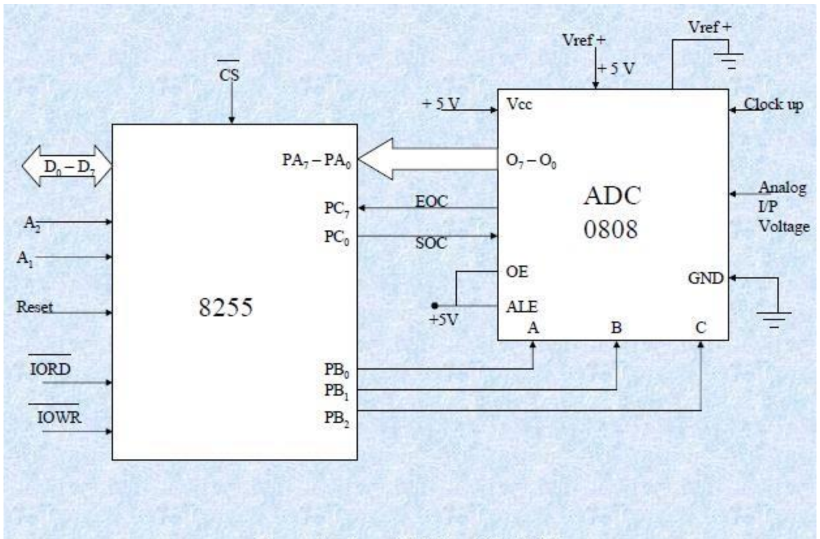
•The process of analog to digital conversion is a slow process, and the microprocessor has to wait for the digital data till the conversion is over. After the conversion is over, the ADC sends end of conversion EOC signal to inform the microprocessor that the conversion is over and the result is ready at the output buffer of the ADC. These tasks of issuing an SOC pulse to ADC, reading EOC signal from the ADC and reading the digital output of the ADC are carried out by the CPU using 8255 I/O ports.

•The time taken by the ADC from the active edge of SOC pulse till the active edge of EOC signal is called as the conversion delay of the ADC.

• It may range anywhere from a few microseconds in case of fast ADC to even a few hundred milliseconds in case of slow ADCs.

•The available ADC in the market use different conversion techniques for conversion of analog signal to digitals. Successive approximation techniques and dual slope integration techniques are the most popular techniques used in the integrated ADC chip.

General algorithm for ADC interfacing contains the following steps:
1. Ensure the stability of analog input, applied to the ADC.
2. Issue start of conversion (SOC) pulse to ADC
3. Read end of conversion signal to mark the end of conversion processes.
4. Read digital data output of the ADC as equivalent digital output.

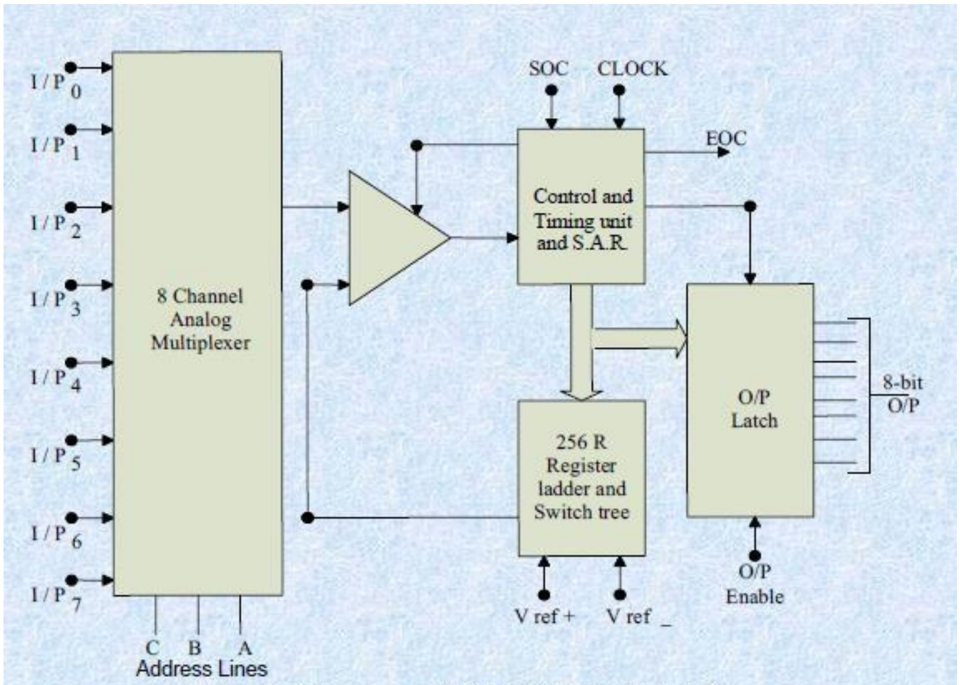


Interfacing 0808 with 8086

- Analog input voltage must be constant at the input of the ADC right from the start of conversion till the end of the conversion to get correct results. This may be ensured by a sample and hold circuit which samples the analog signal and holds it constant for specific time duration.
- The microprocessor may issue a hold signal to the sample and hold circuit. If the applied input changes before the complete conversion process is over, the digital equivalent of the analog input calculated by the ADC may not be correct.

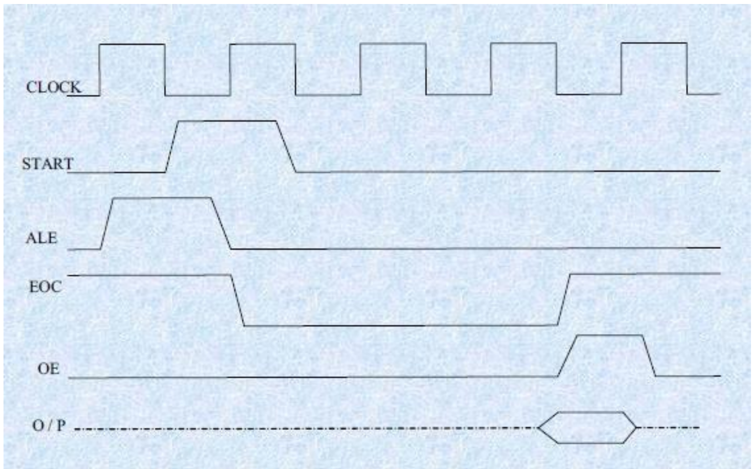
ADC 0808/0809 :
- The analog to digital converter chips 0808 and 0809 are 8-bit CMOS, successive approximation converters. This technique is one of the fast techniques for analog to digital conversion.
- The conversion delay is 100µs at a clock frequency of 640 KHz, which is quite low as compared to other converters. These converters do not need any external zero or full scale adjustments as they are already taken care of by internal circuits.
- These converters internally have a 3:8 analog multiplexer so that at a time eight different analog conversion by using address lines
- ADD A, ADD B, ADD C. Using these address inputs, multichannel data acquisition system can be designed using a single ADC. The CPU may drive these lines using output port lines in case of multichannel applications. In case of single input applications, these may be hardwired to select the proper input.
- There are unipolar analog to digital converters, i.e. they are able to convert only positive analog input voltage to their digital equivalent. These chips do not contain any internal sample and hold

circuit.If one needs a sample and hold circuit for the conversion of fast signal into equivalent digital quantities, it has to be externally connected at each of the analog inputs.



Block Diagram of ADC 0808 / 0809

• Vcc	Supply pins +5V
• GND	GND
• Vref +	Reference voltage positive +5 Volts maximum.
• Vref_	Reference voltage negative 0Volts sminimum
• I/P0–I/P7	Analog inputs
• ADD A,B,C	Address lines for selecting analog inputs.
• O7 – O0	Digital 8-bit output with O7 MSB and O0 LSB
• SOC	Start of conversion signal pin
• EOC	End of conversion signal pin
• OE	Output latch enable pin, if high enables output
• CLK	Clock input for ADC



Timing Diagram of ADC 0808

Example: Interfacing ADC 0808 with 8086 using 8255 ports. Use port A of 8255 for transferring digital data output of ADC to the CPU and port C for control signals. Assume that an analog input is present at I/P2 of the ADC and a clock input of suitable frequency is available for ADC.

• **Solution**: The analog input I/P2 is used and therefore address pins A,B,C should be 0,1,0 respectively to select I/P2. The OE and ALE pins are already kept at +5V to select the ADC and enable the outputs. Port C upper acts as the input port to receive the EOC signal while port C
lower acts as the output port to send SOC to the ADC.

Port A acts as a 8-bit input data port to receive the digital data output from the ADC. The 8255 control word is written as follows:

D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
1 0 0 1 1 0 0 0

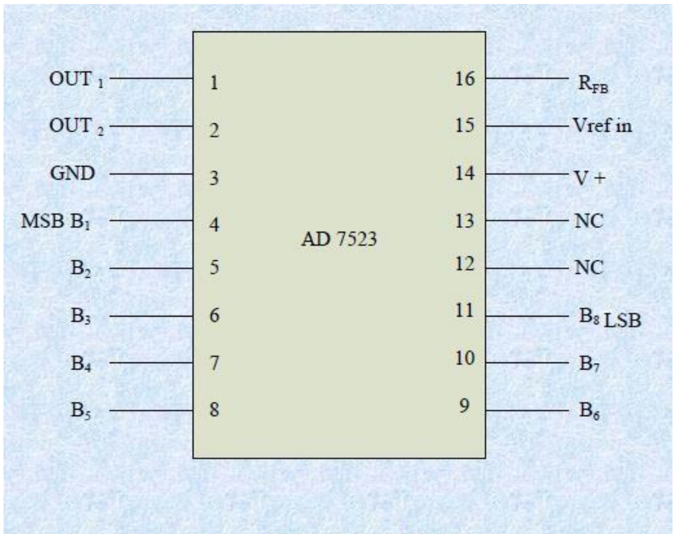
• The required ALP is as follows:

```
MOV AL, 98h           ;initialise 8255 as
OUT CWR, AL           ;discussed above.
MOV AL, 02h           ;Select I/P2 as analog
OUT Port B, AL         ;input.

MOV AL, 00h           ;Give start of conversion
OUT Port C, AL        ; pulse to the ADC
MOV AL, 01h
OUT Port C, AL
MOV AL, 00h
OUT Port C, AL
WAIT: IN AL, Port C ;Check for EOC by
RCR                    ; reading port C upper and
JNC WAIT               ;rotating through carry.
IN AL, Port A         ;If EOC, read digital equivalent
                      ;in AL
HLT                    ;Stop
```

****

9. Explain in detail about Interfacing Digital to Analog Converters

- The digital to analog converters convert binary number into their equivalent voltages. The DAC find applications in areas like digitally controlled gains, motors speed controls, programmable gain amplifiers etc.
- AD7523 8-bit Multiplying DAC : This is a 16 pin DIP, multiplying digital to analog converter, containing R-2R ladder for D-A conversion along with single pole double thrown NMOS switches to connect the digital inputs to the ladder.



Pin Diagram of AD 7523

- The pin diagram of AD7523 is shown in fig the supply range is from +5V to +15V, while Vref may be any where between -10V to +10V. The maximum analog output voltage will be any where between -10V to +10V, when all the digital inputs are at logic high state.
- Usually a zener is connected between OUT1 and OUT2 to save the DAC from negative transients. An operational amplifier is used as a current to voltage converter at the output of AD to convert the current output of AD to a proportional output voltage.
  It also offers additional drive capability to the DAC output.An external feedback resistor acts to control the gain. One may not connect any external feedback resistor, if no gain control is required.
- ***EXAMPLE***: Interfacing DAC AD7523 with an 8086 CPU running at 8MHZ and write an assembly language program to generate a saw tooth waveform of period 1ms with Vmax 5V.
- Solution: Fig shows the interfacing circuit of AD 74523 with 8086 using 8255. program gives an

ALP to generate a saw tooth waveform using circuit.

```
ASSUME CS:CODE
CODE SEGMENT
START: MOV AL,80h ;make all ports output
OUT CW, AL
AGAIN: MOV AL,00h ;start voltage for ramp
BACK : OUT PA, AL
INC AL
CMP AL, 0FFh
JB BACK
JMP AGAIN
CODE ENDS
END START
```

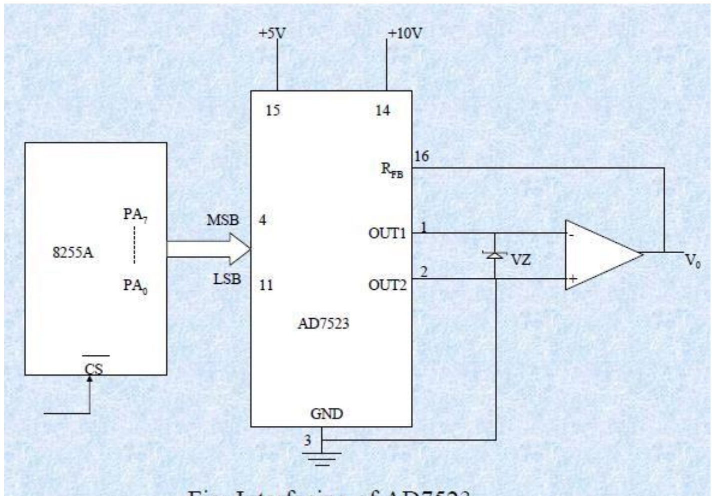


Fig: Interfacing of AD7523

- In the above program, port A is initialized as the output port for sending the digital data as input to DAC. The ramp starts from the 0V (analog), hence AL starts with 00H. To increment the ramp, the content of AL is increased during each execution of loop till it reaches F2H.
- After that the saw tooth wave again starts from 00H, i.e. 0V (analog) and the procedure is repeated. The ramp period given by this program is precisely 1.000625 ms. Here the count F2H has been calculated by dividing the required delay of 1ms by the time required for the execution of the loop once. The ramp slope can be controlled by calling a controllable delay after the OUT instruction

****

10. Draw the block diagram of 8259A and explain how to program 8259A (April 2010).

Programmable Interrupt controller (8259)

Introduction:

For applications where we have interrupts from multiple source, we use an external device called a ***priority interrupt controller*** (PIC) to the interrupt signals into a single interrupt input on the processor.

It accepts requests from the peripheral equipment, determines which of the incoming requests is of the highest importance (priority), ascertains whether the incoming request has a higher priority value than the level currently being serviced, and issues an interrupt to the CPU based on this determination.

Interrupt Request Register (IRR): The interrupts at IRQ input lines are handled by Interrupt Request internally. IRR stores all the interrupt request in it in order to serve them one by one on the priority basis.

• **In-Service Register (ISR)**: This stores all the interrupt requests those are being served, i.e. ISR keeps a track of the requests being served.

Priority Resolver : This unit determines the priorities of the interrupt requests appearing simultaneously. The highest priority is selected and stored into the corresponding bit of ISR during INTA pulse. The IR0 has the highest priority while the IR7 has the lowest one, normally in fixed priority mode. The priorities however may be altered by programming the 8259A in rotating priority mode.

• **Interrupt Mask Register (IMR)** : This register stores the bits required to mask the interrupt inputs. IMR operates on IRR at the direction of the Priority Resolver.

• **Interrupt Control Logic**: This block manages the interrupt and interrupt acknowledge signals to be sent to the CPU for serving one of the eight interrupt requests. This also accepts the interrupt acknowledge (INTA) signal from CPU that causes the 8259A to release vector address on to the data bus.
• **Data Bus Buffer** : This tristate bidirectional buffer interfaces internal 8259A bus to the microprocessor system data bus. Control words, status and vector information pass through data buffer during read or write operations.
• **Read/Write Control Logic**: This circuit accepts and decodes commands from the CPU. This block also allows the status of the 8259A to be transferred on to the data bus.
• **Cascade Buffer/Comparator**: This block stores and compares the ID's all the 8259A used in system. The three I/O pins CASO-2 are outputs when the 8259A is used as a master. The same pins act as inputs when the 8259A is in slave mode. The 8259A in master mode sends the ID of the interrupting slave device on these lines. The slave thus selected, will send its preprogrammed vector address on the data bus during the next INTA pulse.
• **CS**: This is an active-low chip select signal for enabling RD and WR operations of 8259A. INTA function is independent of CS.
• **WR:** This pin is an active-low write enable input to 8259A. This enables it to accept command words from CPU.
• **RD:** This is an active-low read enable input to 8259A. A low on this line enables 8259A to release status onto the data bus of CPU.
• **D0-D7** : These pins from a bidirectional data bus that carries 8-bit data either to control word or from status word registers. This also carries interrupt vector information.
• **CAS0 – CAS2 Cascade Lines:** A signal 8259A provides eight vectored interrupts. If more interrupts are required, the 8259A is used in cascade mode. In cascade mode, a master 8259A along with eight slaves 8259A can provide up to 64 vectored interrupt lines. These three lines act as select lines for addressing the slave 8259A.
• **PS/EN** : This pin is a dual purpose pin. When the chip is used in buffered mode, it can be used as buffered enable to control buffer transreceivers. If this is not used in buffered mode then the pin is used as input to designate whether the chip is used as a master (SP =1) or slave (SP = 0).
• **INT** : This pin goes high whenever a valid interrupt request is asserted. This is used to interrupt the CPU and is connected to the interrupt input of CPU.
• **IR0 – IR7 (Interrupt requests)** :These pins act as inputs to accept interrupt request to the CPU. In edge triggered mode, an interrupt service is requested by raising an IR pin from a low to a high state and holding it high until it is acknowledged, and just by latching it to high level, if used in level triggered mode.

A0
This input signal is used in conjunction with WR and RD signals to write commands into the various command registers, as well as reading the various status registers of the chip. This line can be tied directly to one of the address lines.

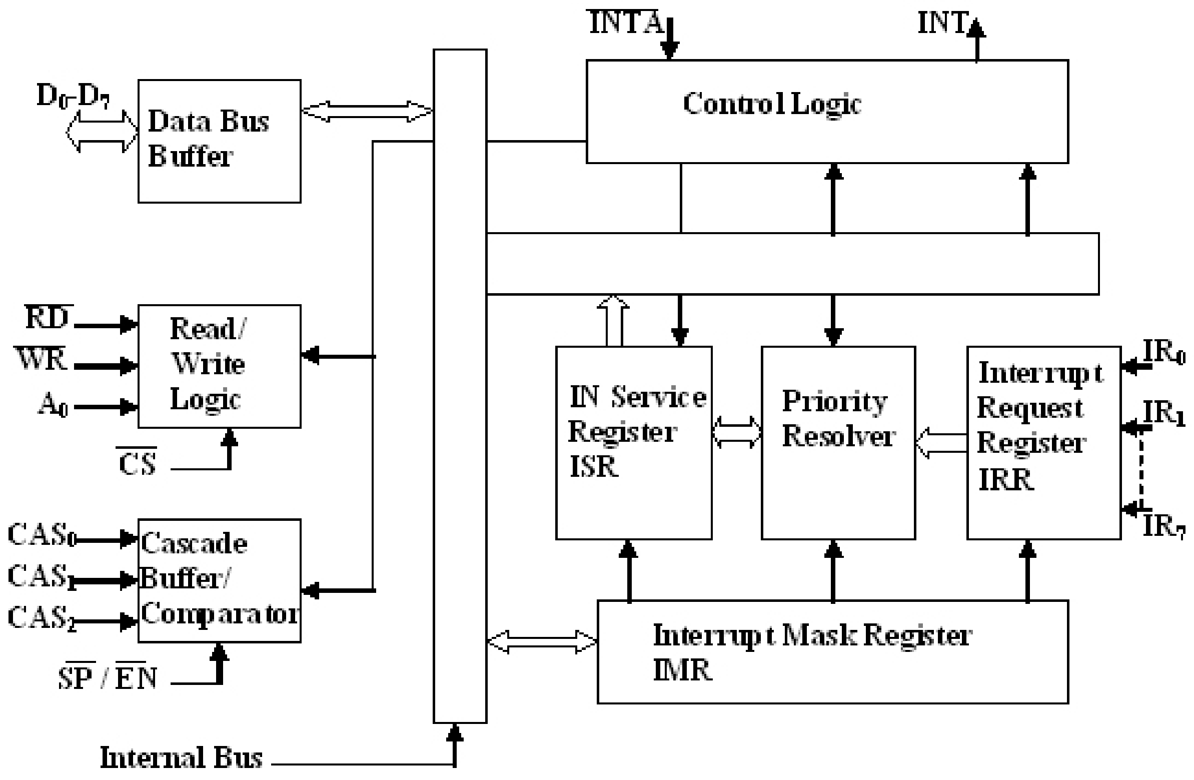


Fig:1 8259A Block Diagram

Interrupt Sequence in an 8086 system

The Interrupt sequence in an 8086-8259A system is described as follows:
1. One or more IR lines are raised high that set corresponding IRR bits.
2. 8259A resolves priority and sends an INT signal to CPU.
3. The CPU acknowledge with INTA pulse.
4. Upon receiving an INTA signal from the CPU, the highest priority ISR bit is set and the corresponding IRR bit is reset. The 8259 will also release a CALL instruction code (11001101) on to the 8 bit data bs through its D7 D0 pins.
5. The CALL instruction will initiate a second INTA pulse. During this period 8259A releases an 8-bit pointer on to a data bus from two more INTA pulses to be sent to the 8259 from the CPU group.
6.These two INTA pulses allow the 8259 to release its programmed subroutine address onto the data bits. The lower 8 bit address is released at the first INTA pulse and the higher 8 bit address is released at the second INTA pulse.
6. This completes the 3 byte CALL instruction released by the 8259. interrupt cycle. The ISR bit is reset at the end of the second INTA pulse if automatic end of interrupt (AEOI) mode is programmed. Otherwise ISR bit remains set until an appropriate EOI command is issued at the end of interrupt subroutine.

Command Words of 8259A
The 8259A accepts two types of command words generated by the CPU:

1. Initialization Command Words (ICWs):
Before normal operation can begin, each 8259A in the system must be brought to a starting pointed by a sequence of 2 to 4 bytes timed by WR pulses.

2. Operational Command Words (OCWs):

These are the command words which command the 8259A to operate in various interrupt modes. These modes are:
a. Fully nested mode
b. Rotating priority mode
c. Special mask mode
d. Polled mode
The OCWs can be written into the 8259A anytime after initialization.

INITIALIZATION COMMAND WORDS (ICWS)

Initialization Command Words (ICW): Before it starts functioning, the 8259A must be initialized by writing two to four command words into the respective command word registers. These are called as initialized command words.
• If A0 = 0 and D4 = 1, the control word is recognized as ICW1. It contains the control bits for edge/level triggered mode, single/cascade mode, call address interval and whether ICW4 is required or not.
• If A0=1, the control word is recognized as ICW2. The ICW2 stores details regarding interrupt vector addresses. The initialization sequence of 8259A is described in form of a flow chart in fig 3 below.
• The bit functions of the ICW1 and ICW2 are self explanatory as shown in fig below.

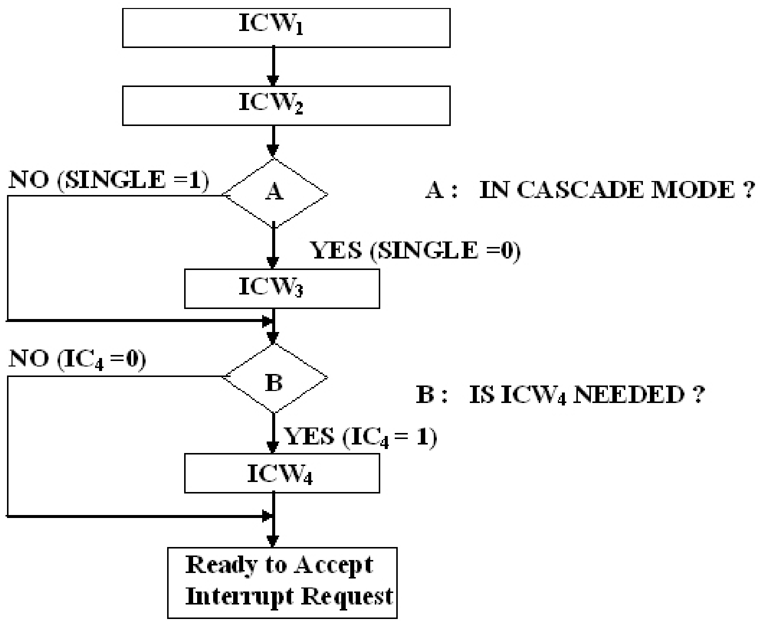


Fig 3: Initialisation Sequence of 8259A

ICW1 :

A write command issued to the 8259 with A_0 =0 and D4 =1 is interpreted as ICW1,which starts the initialization sequence It specifies

1. Single or Multiple 8259 s in the system
2. 4 or 8 bit,interval between interrupt vector locations
3. The address bits A7 A5 of the CALL instruction
4. Edge triggered or Level triggered interrupts
5. ICW4 is needed or not

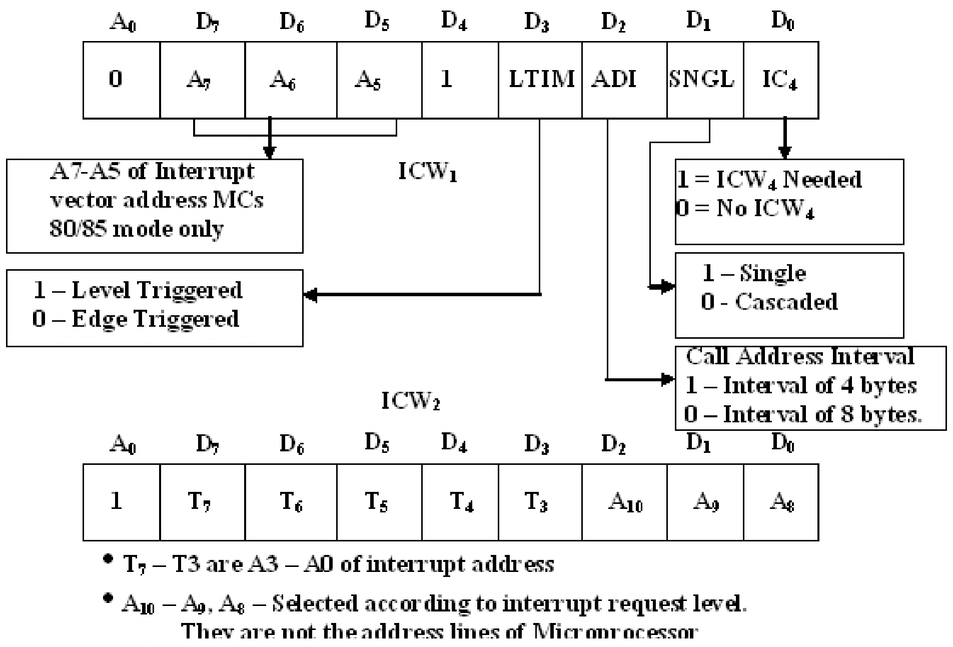

A0 D7 D6 D5 D4 D3 D2 D1 D0
0 A7 A6 A5 1 LTIM ADI SNGL IC4
A7-A5 of Interrupt vector address MCs 80/85 mode only
ICW1
1 = ICW4 Needed
0 = No ICW4
1 – Single
0 - Cascaded
1 – Level Triggered
0 – Edge Triggered
Call Address Interval
1 – Interval of 4 bytes
0 – Interval of 8 bytes.
ICW2
1 T7 T6 T5 T4 T3 A10 A9 A8
• T7 – T3 are A3 – A0 of interrupt address
• A10 – A9, A8 – Selected according to interrupt request level. They are not the address lines of Microprocessor
• A0 =1 selects ICW2

Fig 4 : Instruction Command Words ICW1 and ICW2

Once ICW1 is loaded, the following initialization procedure is carried out internally.
a. The edge sense circuit is reset, i.e. by default 8259A interrupts are edge sensitive.
b. IMR is cleared.
c. IR7 input is assigned the lowest priority.
d. Slave mode address is set to 7.
e. Special mask mode is cleared and status read is set to IRR.
f. If IC4 = 0, all the functions of ICW4 are set to zero. Master/Slave bit in ICW4 is used in the buffered mode only.

In 8086 based system A15-A11 of the interrupt vector address are inserted in place of T7 – T3 respectively and the remaining three bits A8, A9, A10 are selected depending upon the interrupt level, i.e. from 000 to 111 for IR0 to IR7.

ICW1 and ICW2 are compulsory command words in initialization sequence of 8259A as is evident from fig, while ICW3 and ICW4 are optional.

The ICW3 is read only when there are more than one 8259A in the system, cascading is used (SNGL=0).The SNGL bit in ICW1 indicates whether the 8259A in the cascade mode or not.
The ICW3 loads an 8-bit slave register. The detailed functions are as follows.

In master mode [SP = 1 or in buffer mode M/S = 1 in ICW4], the 8-bit slave register will be set bit-wise to 1 for each slave in the system

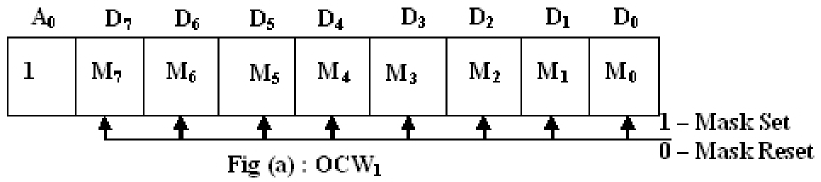
Operation Command Words:

- Once 8259A is initialized using the previously discussed command words for initialisation, it is ready for its normal function, i.e. for accepting the interrupts but 8259A has its own way of handling the received interrupts called as modes of operation.

- These modes of operations can be selected by programming, i.e. writing three internal registers called as operation command words.
- In the three operation command words OCW1, OCW2 and OCW3 every bit corresponds to some operational feature of the mode selected, except for a few bits those are either 1 or 0. The three operation command words are shown in fig with the bit selection details.

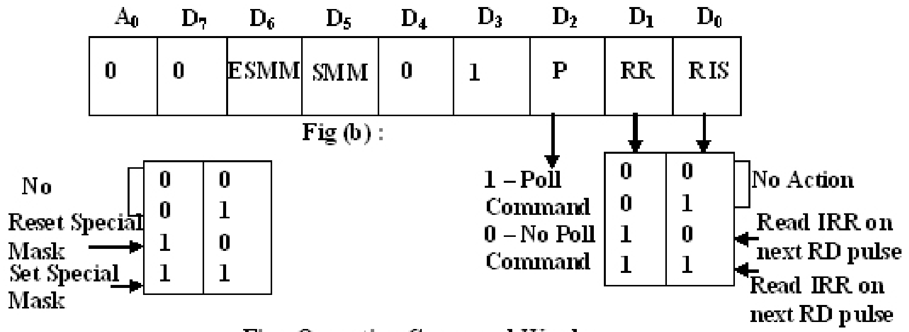
OCW1

Issued with A0= 1,used to mask the interrupts. To enable all the IR lines, the command word is 00H.



A_0 D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
1 M_7 M_6 M_5 M_4 M_3 M_2 M_1 M_0
1 – Mask Set
0 – Mask Reset

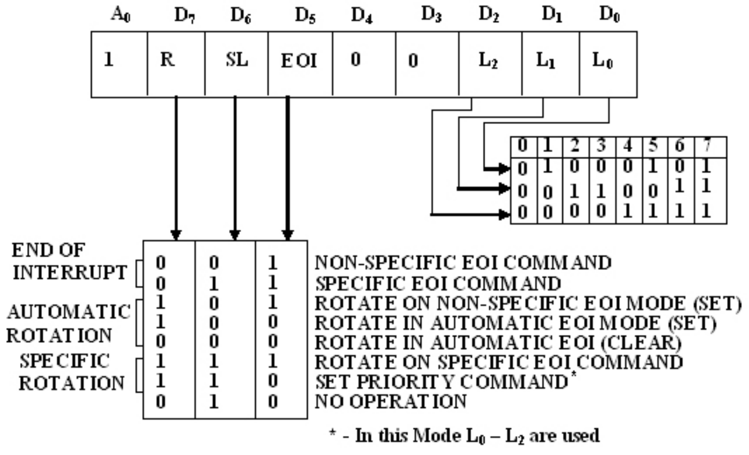
Fig (a) : OCW_1



A_0 D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
0 0 ESMM SMM 0 1 P RR RIS
Fig (b) :
No
Reset Special Mask
Set Special Mask
0 0
0 1
1 0
1 1
1 – Poll Command
0 – No Poll Command
No Action
Read IRR on next RD pulse
Read IRR on next RD pulse

Fig : Operation Command Words

Fig (c) :OCW_2



A_0 D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
1 R SL EOI 0 0 L_2 L_1 L_0
0 1 2 3 4 5 6 7
0 1 0 0 0 1 0 1
0 0 1 1 0 0 1 1
0 0 0 0 1 1 1 1
END OF INTERRUPT
AUTOMATIC ROTATION
SPECIFIC ROTATION
0 0 1 NON-SPECIFIC EOI COMMAND
0 1 1 SPECIFIC EOI COMMAND
1 0 1 ROTATE ON NON-SPECIFIC EOI MODE (SET)
1 0 0 ROTATE IN AUTOMATIC EOI MODE (SET)
0 0 0 ROTATE IN AUTOMATIC EOI (CLEAR)
1 1 1 ROTATE ON SPECIFIC EOI COMMAND
1 1 0 SET PRIORITY COMMAND*
0 1 0 NO OPERATION
* - In this Mode L_0 – L_2 are used

Fig : Operation Command Word

- In OCW2 the three bits, R, SL and EOI control the end of interrupt, the rotate mode and their combinations as shown in fig below.
- The three bits L2, L1 and L0 in OCW2 determine the interrupt level to be selected for operation, if SL bit is active i.e. 1.
- The details of OCW2 are shown in fig.
- In operation command word 3 (OCW3), if the ESMM bit, i.e. enable special mask mode bit is set to 1, the SMM bit is neglected. If the SMM bit, i.e. special mask mode. When ESMM bit is 0 the SMM bit is neglected. If the SMM bit. i.e. special mask mode bit is 1, the 8259A will enter special mask mode provided ESMM=1.

- If ESMM=1 and SMM=0, the 8259A will return to the normal mask mode. The details of bits of OCW3 are given in fig along with their bit definitions.

Priority Modes

Fully Nested Mode

This mode is entered after initialization unless another mode is programmed. The interrupt requests are ordered in priority from 0 through 7. After the initialization sequence, IR0 has the highest prioirity and IR7 the lowest. Priorities can be changed. When an interrupt is acknowledged the highest priority request is determined and its vector placed on the bus.

0	1	2	3	4	5	6	7
IR0	IR1	IR2	IR3	IR4	IR5	IR6	IR7
↓							↓

Highest priority **Lowest Priority**

Now if IR3 is made highest priority the priorities for other interrupt will also be automatically changed.

5	6	7	0	1	2	3	4
IR0	IR1	IR2	IR3	IR4	IR5	IR6	IR7
		↓	↓				

Lowest priority **Highest priority**

End of Interrupt (EOI)

The In Service (IS) bit can be reset either automatically following the trailing edge of the last in sequence INTA pulse (when AEOI bit in ICW1 is set) or by a command word that must be issued to the 8259A before returning from a service routine (EOI command). An EOI command must be issued twice if in the Cascade mode, once for the master and once for the corresponding slave.

Automatic Rotation(equal Priority): This is used in the applications where all the interrupting devices are of equal priority.

•In this mode, an interrupt request IR level receives priority after it is served while the next device to be served gets the highest priority in sequence. Once all the devices are served like this, the first device again receives highest priority.

Specific rotation mode(Specific Priority)

The programmer can change the priorities by programming the bottom priority and the fixing all other priorities.ie if IR4 is programmed as the lowest priority, then IR% will have the highest one**.**

• **Automatic EOI Mode:** Till AEOI=1 in ICW4, the 8259A operates in AEOI mode.

Special mask mode

- In the special mask mode, when a mask bit is set in OCW1,it inhibits further interrupts at that level and enables interrupts from all other levels that are not masked. Thus any interrupts may be selectively enabled by loading the mask register.

Poll command

- Service to devices is achieved by software using a poll command. So INTA sequence is not needed. It is used to expand the number of priorities levels to more than 64.

12. Explain in detail about Traffic light Control

Traffic lights, which may also be known as stop lights, traffic lamps, traffic signals, signal lights, robots or semaphore, are signaling devices positioned at road intersections, pedestrian crossings and other locations to control competing flows of traffic

ABOUT THE COLORS OF TRAFFIC LIGHT CONTROL

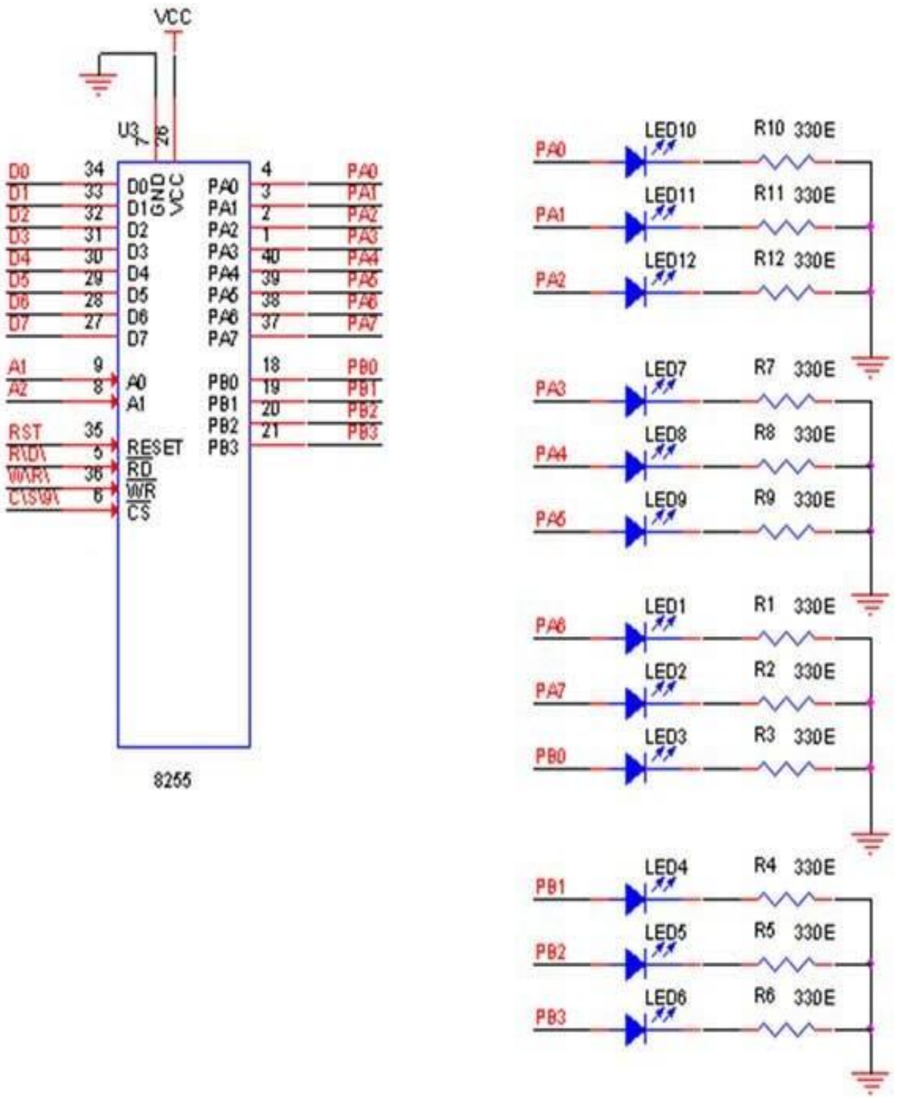
Traffic lights alternate the right of way of road users by displaying lights of a standard color (red, yellow/amber, and green), using a universal color code (and a precise sequence to enable comprehension by those who are color blind).

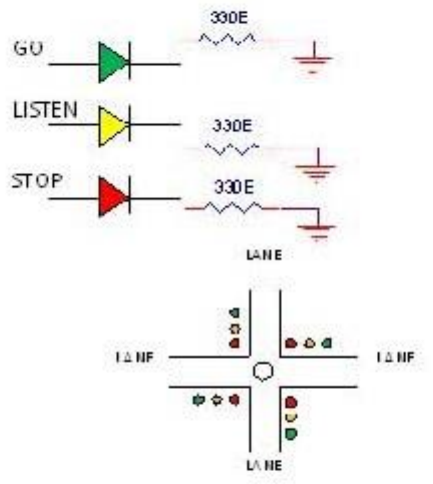
Illumination of the red signal prohibits any traffic from proceeding. Usually, the red light contains some orange in its hue, and the green light contains some blue, for the benefit of people with red-green color blindness, and "green" lights in many areas are in fact blue lenses on a yellow light (which together appear green).

INTERFACING TRAFFIC LIGHT WITH 8086

The Traffic light controller section consists of 12 Nos. point LEDs arranged by 4 Lanes in Traffic light interface card. Each lane has Go (Green), Listen (Yellow) and Stop(Red) LED is being placed

PIN ASSIGNMENT WITH 8086


VCC
U3
8255
D0
D1
D2
D3
D4
D5
D6
D7
A0
A1
RESET
RD
WR
CS
GND
VCC
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
LED1
LED2
LED3
LED4
LED5
LED6
LED7
LED8
LED9
LED10
LED11
LED12
R1 330E
R2 330E
R3 330E
R4 330E
R5 330E
R6 330E
R7 330E
R8 330E
R9 330E
R10 330E
R11 330E
R12 330E

LAN Direction	8086 LINES	MODULES	Traffic Light Controller Card
SOUTH	PA.0	GO	 330E GO LISTEN 330E STOP 330E LANE LANE LANE LANE
	PA.1	LISTEN	
	PA.2	STOP	
EAST	PA.3	GO	
	PA.4	LISTEN	
	PA.5	STOP	
NORTH	PA.6	GO	
	PA.7	LISTEN	
	PB.0	STOP	
WEST	PB.1	GO	Make high to - LED On Make low to – LED Off
	PB.2	LISTEN	
	PB.3	STOP	
	13-16	NC	
PWR	17,19	Vcc	Supply form MCU/MPU/FPGA Kits
	18,20	Gnd	

ASSEMBLY PROGRAM TO INTERFACE TRAFFIC LIGHT WITH 8086

MEMORY ADDRESS	OPCODE	MNEMONICS
1100	BB 00 11	START: MOV BX, 1200H
1103	B9 08 00	MOV CX, 0008H
1106	8A 07	MOV AL,[BX]
1108	BA 36 FF	MOV DX, CONTROL PORT
110B	EE	OUT DX, AL
110C	43	INC BX
110D	8A 07	NEXT:MOV AL,[BX]
110F	BA 30 FF	MOV DX, PORT A
1112	EE	OUT DX,AL
1113	43	INC BX
1114	8a 07	MOV AL,[BX]
1116	BA 32 FF	MOV DX,PORT B
1119	EE	OUT DX,AL
111A	E8 07 00	CALL DELAY
111D	E2 F1	INC BX
111E		LOOP NEXT
1120	EB E4	JMP START
1122	51	DELAY:PUSH CX
1124	B9 00 05	MOV CX,0005H
1126	BA FF FF	REPEAT:MOV DX,0FFFFH
1129	4A	LOOP2: DEC DX
112A	75 FD	JNZ LOOP2
112C	E2 F8	LOOP REPEAT
112E	59	POP CX
112F	C3	RET

LOOKUP TABLE

1200	80H
1201	21H,09H,10H,00H (SOUTH WAY)
1205	0CH,09H,80H,00H (EAST WAY)
1209	64H,08H,00H,04H (NOURTH WAY)
120D	24H,03H,02H,00H (WEST WAY)
1211	END

13. Discuss the following in detail

(i). Interfacing LED with 8086

(ii) Interfacing LED with 8086

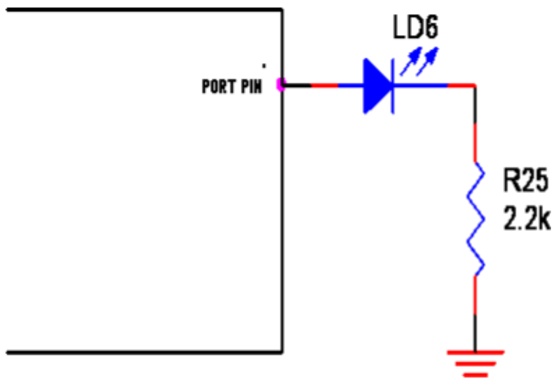
(iii) Keyboard interface

LED (LIGHT EMITTING DIODES)

Light Emitting Diodes (LED) is the most commonly used components, usually for displaying pins digital states. Typical uses of LEDs include alarm devices, timers and confirmation of user input such as a mouse click or keystroke.

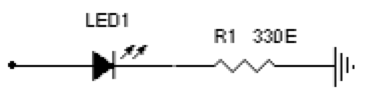
INTERFACING LED

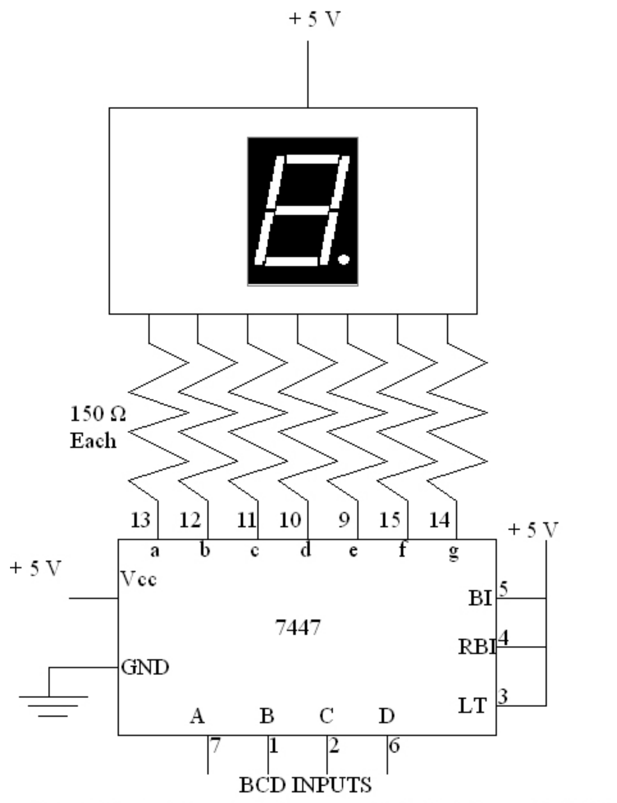
Fig. 1 shows how to interface the LED to microprocessor. As you can see the Anode is connected through a resistor to GND & the Cathode is connected to the Microprocessor pin. So when the Port Pin is HIGH the LED is OFF & when the Port Pin is LOW the LED is turned ON.



INTERFACING LED WITH 8086 We now want to flash a LED in 8086 Trainer Board. It works by turning ON a LED & then turning it OFF & then looping back to START. However the operating speed of microprocessor is very high

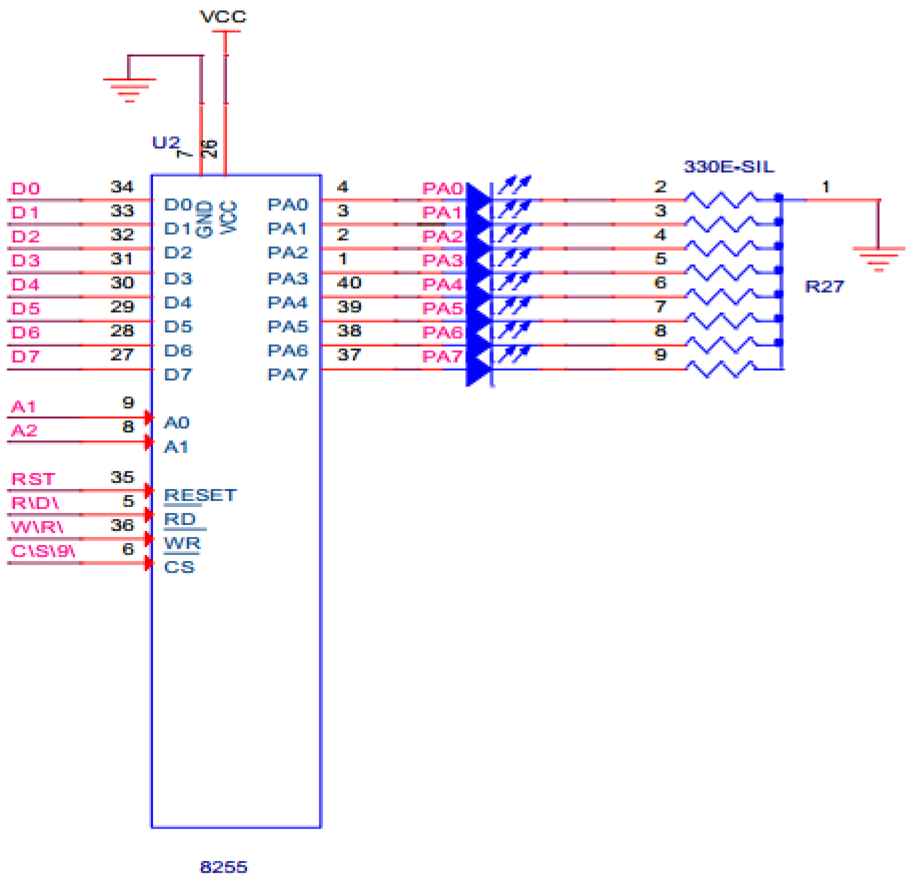
PIN ASSIGNMENT WITH 8086

	Point LEDs	8255 Lines	LED Selection
DIGITAL OUTPUTS	LD1	PA.0	
	LD2	PA.1	
	LD3	PA.2	
	LD4	PA.3	
	LD5	PA.4	
	LD6	PA.5	
	LD7	PA.6	
	LD8	PA.7	



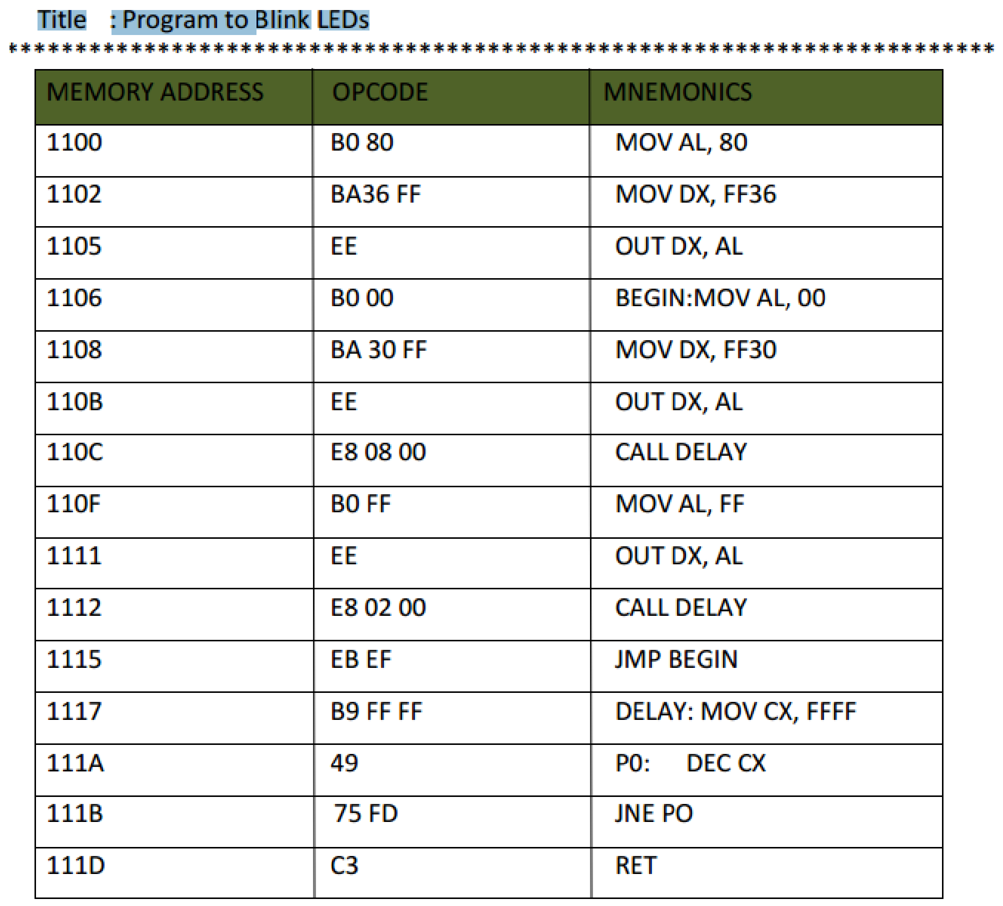
Circuit for driving single 7-segment LED display with 7447

CIRCUIT DIAGRAM TO INTERFACE LED WITH 8255



ASSEMBLY PROGRAM TO ON AND OFF LED USING 8086

Title : Program to Blink LEDs


Title : Program to Blink LEDs

****

MEMORY ADDRESS	OPCODE	MNEMONICS
1100	B0 80	MOV AL, 80
1102	BA36 FF	MOV DX, FF36
1105	EE	OUT DX, AL
1106	B0 00	BEGIN:MOV AL, 00
1108	BA 30 FF	MOV DX, FF30
110B	EE	OUT DX, AL
110C	E8 08 00	CALL DELAY
110F	B0 FF	MOV AL, FF
1111	EE	OUT DX, AL
1112	E8 02 00	CALL DELAY
1115	EB EF	JMP BEGIN
1117	B9 FF FF	DELAY: MOV CX, FFFF
111A	49	P0: DEC CX
111B	75 FD	JNE PO
111D	C3	RET

‘’’’

LCD INTERFACING

Liquid Crystal Display
Introduction:

Liquid Crystal displays are created by sandwiching a thin 10-12 μm layer of a liquid-crystal fluid between two glass plates. A transparent, electrically conductive film or backplane is put on the rear glass sheet. Transparent sections of conductive film in the shape of the desired characters are coated on the front glass plate.

When a voltage is applied between a segment and the backplane, an electric field is created in the region under the segment. This electric field changes the transmission of light through the region under the segment film.

There are two commonly available types of LCD

- **Dynamic scattering and field effect.**
- **Dynamic scattering types of LCD**: It scrambles the molecules where the field is present. This produces an etched-glass-looking light character on a dark background.

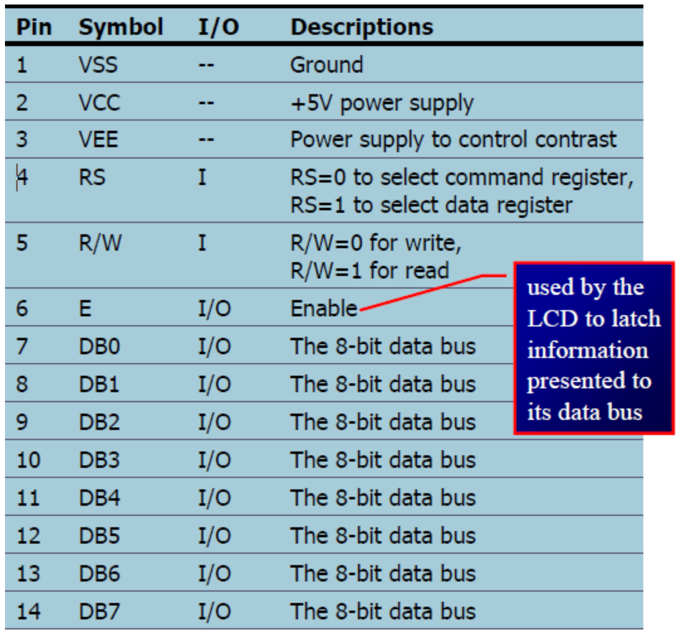
Field-effect types use polarization to absorb light where the electric field is present. This produces dark characters on a silver- gray background.

Most LCD’s require a voltage of 2 or 3 V between the backplane and a segment to turn on the segment. We cannot just connect the backplane to ground and drive the segment with the outputs of a TTL decoder. The reason for this is a steady dc voltage of more than about 50mV is applied between a segment and the To prevent a dc buildup on the segments, the segment-drive signals for LCD must be square waves with a frequency of 30 to 150 Hz.

Even if you pulse the TTL decoder, it still will not work because the output low voltage of TTL devices is greater than 50mV. CMOS gates are often used to drive LCDs.

Advantages of LCD

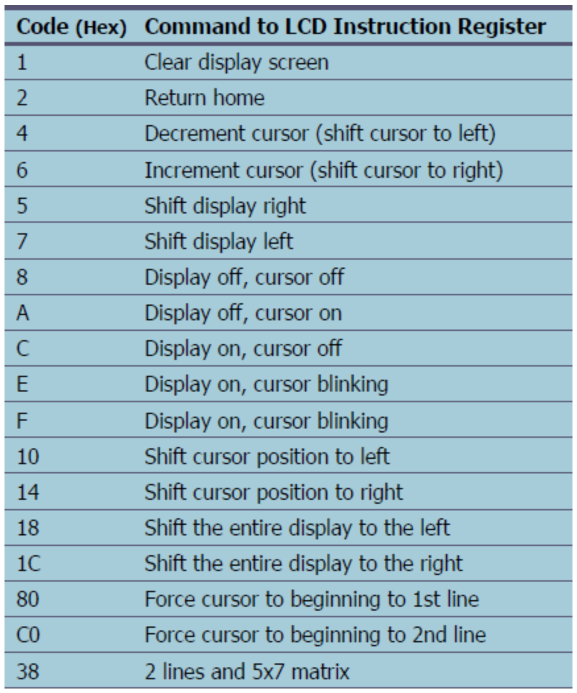
- LCD is finding widespread use replacing LEDs
- The declining prices of LCD
- The ability to display numbers, characters, and graphics
- Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD
- Ease of programming for characters and Graphics



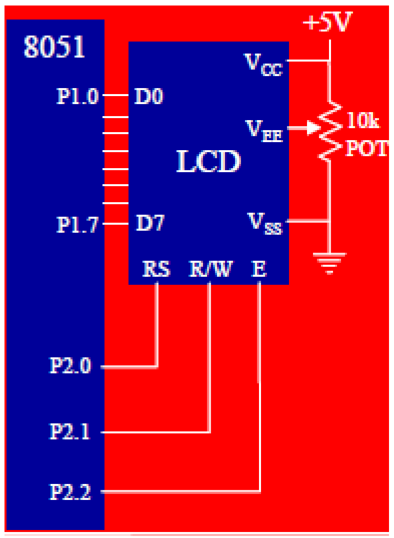
Pin	Symbol	I/O	Descriptions
1	VSS	--	Ground
2	VCC	--	+5V power supply
3	VEE	--	Power supply to control contrast
4	RS	I	RS=0 to select command register, RS=1 to select data register
5	R/W	I	R/W=0 for write, R/W=1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8-bit data bus
8	DB1	I/O	The 8-bit data bus
9	DB2	I/O	The 8-bit data bus
10	DB3	I/O	The 8-bit data bus
11	DB4	I/O	The 8-bit data bus
12	DB5	I/O	The 8-bit data bus
13	DB6	I/O	The 8-bit data bus
14	DB7	I/O	The 8-bit data bus

used by the LCD to latch information presented to its data bus

LCD Command Codes



Code (Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning to 1st line
C0	Force cursor to beginning to 2nd line
38	2 lines and 5x7 matrix



To send any of the commands to the LCD, make pin RS=0. For data, make RS=1. Then send a high-to-low pulse to the E pin to enable the internal latch of the LCD. This is shown in the code below.

```
;calls a time delay before sending next data/command
;P1.0-P1.7 are connected to LCD data pins D0-D7
;P2.0 is connected to RS pin of LCD
;P2.1 is connected to R/W pin of LCD
;P2.2 is connected to E pin of LCD
ORG 0H
MOV A,#38H ;INIT. LCD 2 LINES, 5X7 MATRIX
ACALL COMNWRT ;call command subroutine
ACALL DELAY ;give LCD some time
MOV A,#0EH ;display on, cursor on
ACALL COMNWRT ;call command subroutine
ACALL DELAY ;give LCD some time
MOV A,#01 ;clear LCD
ACALL COMNWRT ;call command subroutine
ACALL DELAY ;give LCD some time
MOV A,#06H ;shift cursor right
ACALL COMNWRT ;call command subroutine
ACALL DELAY ;give LCD some time
MOV A,#84H ;cursor at line 1, pos. 4
ACALL COMNWRT ;call command subroutine
ACALL DELAY ;give LCD some time
MOV A,#’N’ ;display letter N
ACALL DATAWRT ;call display subroutine
ACALL DELAY ;give LCD some time
MOV A,#’O’ ;display letter O
ACALL DATAWRT ;call display subroutine
AGAIN: SJMP AGAIN ;stay here
COMNWRT: ;send command to LCD
MOV P1,A ;copy reg A to port 1
CLR P2.0 ;RS=0 for command
CLR P2.1 ;R/W=0 for write
SETB P2.2 ;E=1 for high pulse
ACALL DELAY ;give LCD some time
CLR P2.2 ;E=0 for H-to-L pulse
RET
DATAWRT: ;write data to LCD
MOV P1,A ;copy reg A to port 1
SETB P2.0 ;RS=1 for data
CLR P2.1 ;R/W=0 for write
SETB P2.2 ;E=1 for high pulse
ACALL DELAY ;give LCD some time
CLR P2.2 ;E=0 for H-to-L pulse
RET
DELAY: MOV R3,#50 ;50 or higher for fast CPUs
HERE2: MOV R4,#255 ;R4 = 255
HERE: DJNZ R4,HERE ;stay until R4 becomes 0
DJNZ R3,HERE2
RET
END
```

```
;Check busy flag before sending data, command to LCD
;p1=data pin
;P2.0 connected to RS pin
;P2.1 connected to R/W pin
;P2.2 connected to E pin

ORG 0H
MOV A,#38H ;init. LCD 2 lines ,5x7 matrix
ACALL COMMAND ;issue command
MOV A,#0EH ;LCD on, cursor on
ACALL COMMAND ;issue command
MOV A,#01H ;clear LCD command
ACALL COMMAND ;issue command
MOV A,#06H ;shift cursor right
ACALL COMMAND ;issue command
MOV A,#86H ;cursor: line 1, pos. 6
ACALL COMMAND ;command subroutine
MOV A,#'N' ;display letter N
ACALL DATA_DISPLAY
MOV A,#'O' ;display letter O
ACALL DATA_DISPLAY
HERE:SJMP HERE ;STAY HERE
```

The Following fig shows how two CMOS gate outputs can be connected to drive an LCD segment and backplane.

- The off segment receives the same drive signal as the backplane. There is never any voltage between them, so no electric field is produced. The waveform for the on segment is 180 out of phase with the backplane signal, so the voltage between this segment and the backplane will always be +V.
- The logic for this signal, a square wave and its complement. To the driving gates, the segment-backplane sandwich appears as a somewhat leaky capacitor.
- The CMOS gates can be easily supply the current required to charge and discharge this small capacitance.
- Older inexpensive LCD displays turn on and off too slowly to be multiplexed the way we do LED display.
- At 0c some LCD may require as much as 0.5s to turn on or off. To interface to those types we use a non multiplexed driver device.
- More expensive LCD can turn on and off faster, so they are often multiplexed using a variety of techniques.
- In the following section we show you how to interface a non multiplexed LCD to a microprocessor such as SDK-86.
- Intersil ICM7211M can be connected to drive a 4-digit, non multiplexed, 7- segment LCD display.
- The 7211M input can be connected to port pins or directly to microcomputer bus.We have connected the CS inputs to the Y2 output of the 74LS138 port decoder.
- According to the truth table the device will then be addressable as ports with a base address of FF10H. SDK-86 system address lines A2 is connected to the digit-select input (DS2) and system address lines A1 is connected to the DS1 input. This gives digit 4 a system address of FF10H.

A8-A15	A5-A7	A4	A3	A2	A1	A0	M/IO	Y Output Selected	System Base Address	Device
1	0	0	0	X	X	0	0	00	F F 0 0	8259A #1
1	0	0	1	X	X	0	0	1	F F 0 8	8259A #2
1	0	1	0	X	X	0	0	2	F F 1 0	
1	0	1	1	X	X	0	0	3	F F 1 8	
1	0	0	0	X	X	1	0	4	F F 0 1	8254
1	0	0	1	X	X	1	0	5	F F 0 9	
1	0	1	0	X	X	1	0	6	F F 1 1	
1	0	1	1	X	X	1	0	7	F F 1 9	
ALL OTHER STATES								NONE		

Fig : Truth table for 74LS138 address decoder

Digit 3 will be addressed at FF12H, digit 2 at FF14H and digit 1 at FF16H.

- The data inputs are connected to the lower four lines of the SDK-86 data bus.The oscillator input is left open. To display a character on one of the digits, you simply keep the 4-bit hex code for that digit in the lower 4 bits of the AL register and output it to the system address for that digit.

- The ICM7211M converts the 4-bit hex code to the required 7-segment code.
- The rising edge of the CS input signal causes the 7-segment code to be latched in the output latches for the address digit.
- An internal oscillator automatically generates the segment and backplane drive waveforms as in fig . For interfacing with the LCD displays which can be multiplexed the Intersil ICM7233 can be use.

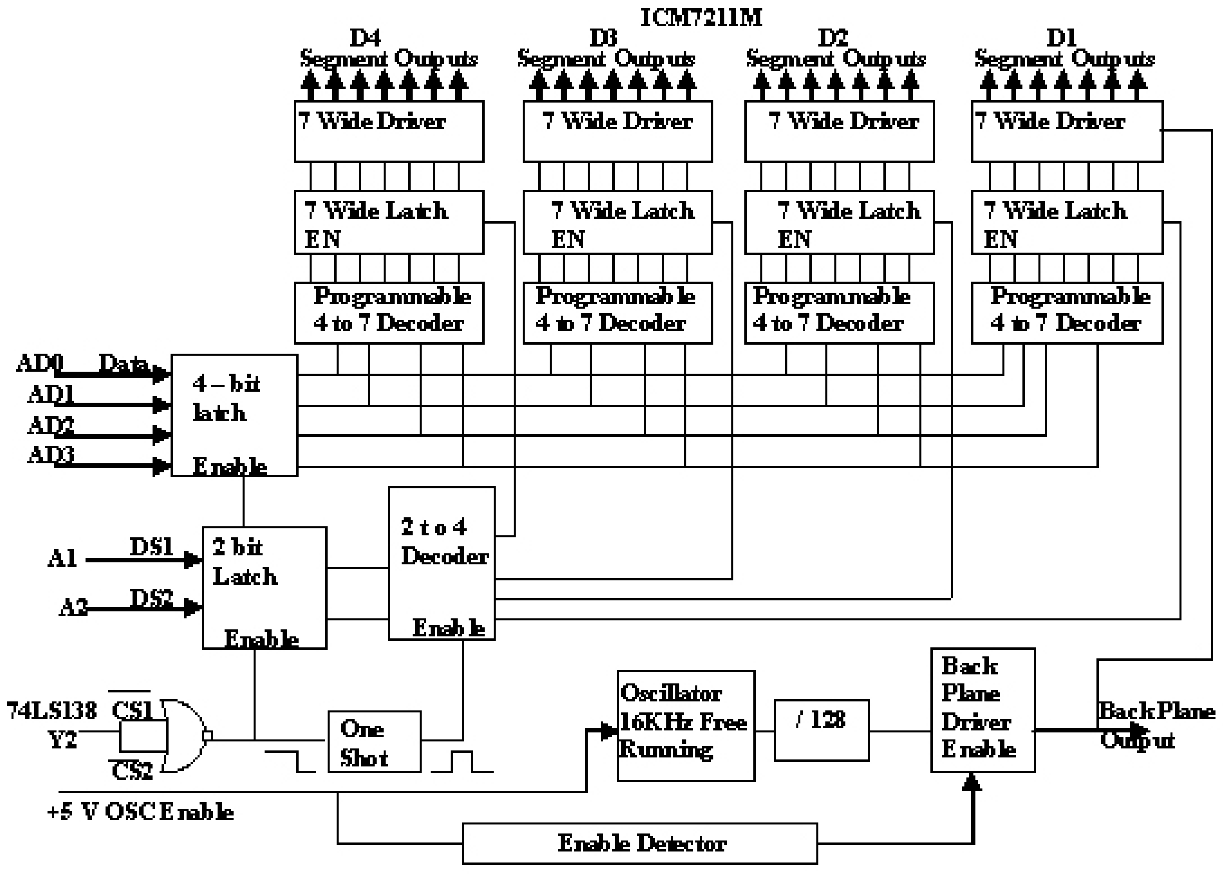


Fig : Circuit for interfacing four LCD digits to an SDK-86 bus using ICM7211M

KEYBOARD INTERFACING

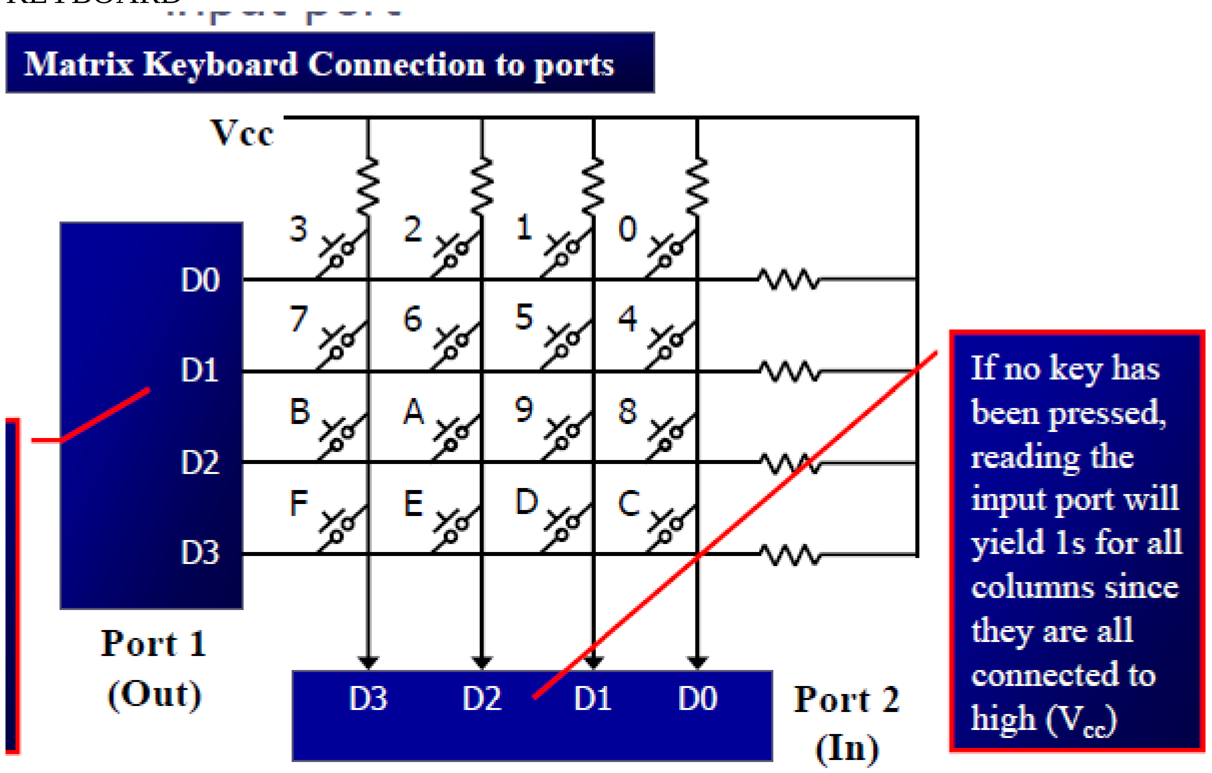
Keyboards are organized in a matrix of rows and columns

 The CPU accesses both rows and columns through ports. Therefore, with two 8-bit ports, an 8 x 8 matrix of keys can be connected to a microprocessor

 When a key is pressed, a row and a column make a contact, Otherwise, there is no connection between rows and columns

 In IBM PC keyboards, a single microcontroller takes care of hardware and software interfacing .A 4x4 matrix connected to two ports

 The rows are connected to an output port and the columns are connected to an input port KEYBOARD



Matrix Keyboard Connection to ports
Vcc
D0
D1
D2
D3
Port 1 (Out)
3
2
1
0
7
6
5
4
B
A
9
8
F
E
D
C
D3
D2
D1
D0
Port 2 (In)
If no key has been pressed, reading the input port will yield 1s for all columns since they are all connected to high (V_{cc})

It is the function of the microcontroller to scan the keyboard continuously to detect and identify the key pressed To detect a pressed key, the microcontroller grounds all rows by providing 0 to the output latch, then it reads the columns

- If the data read from columns is D3 – D0 = 1111, no key has been pressed and the process continues till key press is detected
- If one of the column bits has a zero, this means that a key press has occurred. For example, if D3 – D0 = 1101, this means that a key in the D1 column has been pressed
- After detecting a key press, microcontroller will go through the process of identifying the key Starting with the top row, the microcontroller grounds it by providing a low to row D0 only
- It reads the columns, if the data read is all 1s, no key in that row is activated and the process is moved to the next row
- It grounds the next row, reads the columns, and checks for any zero
- This process continues until the row is identified
- After identification of the row in which the key has been pressed
- Find out which column the pressed key belongs to
- Corresponding key is displayed.

‘’’’

14.Explain in detail about alarm controller.

DS12887 RTC INTERFACING
This interfacing and programming of the DS12C887 real time clock (RTC) chip.
DS12887 RTC INTERFACING

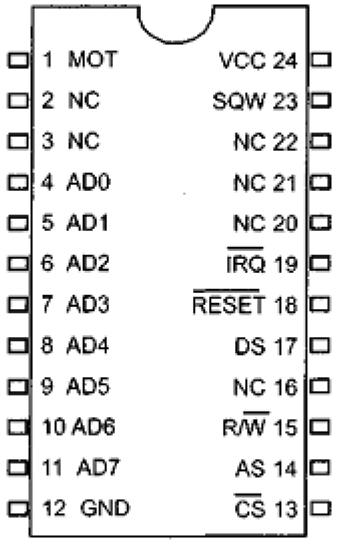


Figure 16-1. DS12887 RTC Chip

- ✓ The RTC provides accurate time and date for many applications.
- ✓ The RTC chip in the IBM PC provides time components of hour, minute and second, in addition to date / calendar components of year, month and day.
- ✓ It uses an internal battery, which keeps the time and date even when the power is off (over 10 years).
- ✓ One of the most widely used RTC chip is the DS 12887.
- ✓ It keeps track of “seconds, minutes, hours, days, day of week, date, month, and year with leap-year compensation valid up to year 2100″.
- ✓ The above information is provided in both binary (hex) and BCD formats. The DS 12887 supports both 12-hour and 24-hour clock modes with AM and PM in the 12-hour mode.
- ✓ It also supports the Daylight Savings Time option. The DS 12887 uses CMOS technology to keep the power consumption low and it has the designation DS12C887, where C is for CMOS. The DS12887 has a total of 128 bytes of nonvolatile RAM.
- ✓ It uses 14 bytes of RAM for clock/calendar and control registers, and the other 114 bytes of RAM are for general-purpose data storage.
- ✓ Next we describe the pins of the DS 12887. See Figure 16-1.

Vcc
Pin 24 provides external supply voltage to the chip. The external voltage source is +5V. When Voltage falls below the 3V level, the external source is switched off and the internal lithium battery provides power to the RTC.
GND
Pin 12 is the ground.

ADO-AD7

The multiplexed address/data pins provide both addresses and data to the chip. Addresses are latched into the DS 12887 on the falling edge of the AS (ALE) signal.

A simple way of connecting the DS 12887 to the 8051 is shown in Figure 16-2.

AS (ALE)

AS (address strobe) is an input pin. On the falling edge it will cause the addresses to be latched into the DS 12887. The AS pin is used for demultiplexing the address and data and is connected to the ALE pin of the 8051 chip.

MOT

This is an input pin that allows the choice between the Motorola and Intel microcontroller bus timings. The MOT pin is connected to GND for the Intel timing. That means when we connect DS 12887 to the 8051, MOT = GND.

DS

Data strobe or read is an input. When MOT = GND for Intel timing, the DS pin is called the RD (read) signal and is connected to the RD pin of the 8051.

R/W

Read/Write is an input pin. When MOT = GND for the Intel timing, the R/W pin is called the WR (write) signal and is connected to the WR pin of the 8051.

CS

Chip select is an input pin and an active low signal. When the DS 12887 is in write-protected state, all inputs are ignored.

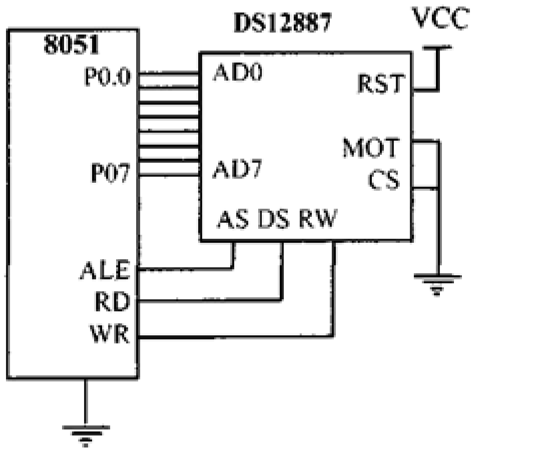


Figure 16-2. DS12887 Connection to 8051

IRQ

Interrupt request is an output pin and active low signal. To use IRQ, the interrupt-enable bits in register B must be set high. The interrupt feature of the DS12287 is discussed in Section 16.3.

SQW

Square wave is an output pin. We can program the DS 12887 to provide up to 15 different square waves. The frequency of the square wave is set by programming register A.

RESET

Pin 18 is the reset pin. It is an input and is active low (normally high). In most applications the reset pin is connected to the V_{cc} pin.

Address map of the DS12887

The DS12887 has a total of 128 bytes of RAM space with addresses 00 -7FH. The first ten locations, 00 – 09, are set aside for RTC values of time, calendar, and alarm data.

The next four bytes are used for the control and status registers. They are registers A, B, C, and D and are located at addresses 10-13 (OA – OD in hex).

The next 114 bytes from addresses OEH to 7FH are available for data storage. The entire 128 bytes of RAM are accessible directly for read or write except the following:

1. Registers C and D are read-only.
2. D7 bit of register A is read-only.
3. The high-order bit of the seconds byte is read-only.

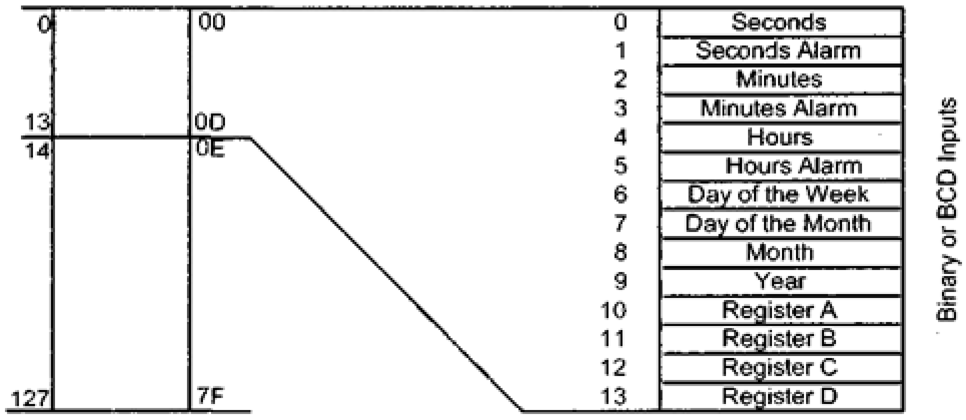


Figure 16-3. DS12887 Address Map

Figure 16.3 shows the address map of the DS 12887.

Time, calendar, and alarm address locations and modes

The byte addresses 0-9 are set aside for the time, calendar, and alarm data. Table 16-1 shows their address locations and modes. Notice the data is available in both binary (hex) and BCD formats.

Turning on the oscillator for the first time

The DS12887 is shipped with the internal oscillator turned off in order to save the lithium battery. We need to turn on the oscillator before we use the time keeping features of the DS 12887. To do that, bits D6 – D4 of register A must be set to value 010. See Figure 16-4 for details of register A.

The following code shows how to access the DS12887′s register A and is written for the Figure 16-2 connection. In Figure 16-2, the DS 12887 is using the external memory space of the 8051 and is mapped to address space of 00 – 7FH since CS = 0. See Chapter 14 for a discussion of external memory in the 8051. For the programs in this chapter, we use instruction “MOVX A, @RO” since the address is only 8-bit. In the case of a 16-bit address, we must use “MOVX A, @DPTR” as was shown in Chapter 14. Examine the following code to see how to access the DS12887 of Figure 16-2.

Table 16-1: DS12887 Address Location for Time, Calendar, and Alarm

Address Location	Function	Decimal Range	Data Mode Range Binary (hex)	Data Mode Range BCD
0	Seconds	0 - 59	00 - 3B	00 - 59
1	Seconds Alarm	0 - 59	00 - 3B	00 - 59
2	Minutes	0 - 59	00 - 3B	00 - 59
3	Minutes Alarm	0 - 59	00 - 3B	00 - 59
4	Hours, 12-Hour Mode	1 - 12	01 - 0C AM	01 - 12 AM
	Hours, 12-Hour Mode	1 - 12	81 - 8C PM	81 - 92 PM
	Hours, 24-Hour Mode	0 - 23	0 - 17	0 - 23
5	Hours Alarm, 12-Hour	1 - 12	01 - 0C AM	01 - 12 AM
	Hours Alarm, 12-Hour	1 - 12	81 - 8C PM	81 - 92 PM
	Hours Alarm, 24-Hour	0 - 23	0 - 17	0 - 23
6	Day of the Week, Sun = 1	1 - 7	01 - 07	01 - 07
7	Day of the Month	1 - 31	01 - 1F	01 - 31
8	Month	1 - 12	01 - 0C	01 - 12
9	Year	0 - 99	00 - 63	00 - 99

```
ACALL DELAY_200ms ;RTC NEEDS 200ms AFTER POWER-UP
MOV  R0,#10        ;R0=0AH,Reg A address
MOV  A,#20H        ;010 in D6-D4 to turn on osc.
MOVX @R0,A         ;send it to Reg A of DS12887
```

UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0

UIP Update in progress. This is a read-only bit.

DV2	DV1	DV0	
0	1	0	will turn the oscillator on

RS3 RS2 RS1 RS0
Provides 14 different frequencies at the SQW pin. See Section 16.3 and the DS12887 data sheet.

Figure 16-4. Register A Bits for Turning on the DS12887′s Oscillator

Setting the time

When we initialize the time or date, we need to set D7 of register B to 1. This will prevent any update at the middle of the initialization. After setting the time and date, we need to make D7 = 0 to make sure that the clock and time are updated. The update occurs once per second. The following code initializes the clock at 16:58:55 using the BCD mode and 24-hour clock mode with daylight savings time. See also Figure 16-5 for details of register B.

```
;------WAIT 200msec FOR RTC TO BE READY AFTER POWER-UP
     ACALL DELAY_200ms
;------------TURNING ON THE RTC
     MOV   R0,#10      ;R0=0AH,Reg A address
     MOV   A,#20H      ;010 in D6-D4 to turn on osc.
     MOVX  @R0,A       ;send it to Reg A of DS12887
;--------------Setting the Time mode
     MOV   R0,#11      ;Reg B address
     MOV   A,#83H ;BCD,24hrs,Daylight saving,D7=1 No update
     MOVX  @R0,A       ;send it to Reg B
;----------Setting the Time
     MOV   R0,#0       ;point to seconds address
     MOV    A,#55H     ;seconds= 55H (BCD numbers need H)
     MOVX  @R0,A       ;set seconds
     MOV   R0,#02      ;point to minutes address
     MOV   A,#58H      ;minutes= 58
     MOVX  @R0,A       ;set minutes
     MOV   R0,#04      ;point to hours address
     MOV   A,#16H      ;hours=16
     MOVX  @R0,A       ;set hours
     MOV   R0,#11      ;Reg B address
     MOV   A,#03       ;D7=0 of reg B to allow update
     MOVX  @R0,A       ;send it to reg B
```

SET	PIE	AIE	UIE	SQWE	DM	24/12	DSE

SET	SET = 0: Clock is counting once per second and time and dates are updated SET = 1: Update is inhibited (during the initialization we must make SET = 1)
PIE	Periodic Interrupt Enable. See Section 16.3.
AIE	Alarm Interrupt Enable. The AIE = 1 will allow the IRQ to be asserted, when all three bytes of time (yy:mm:dd) are the same as the alarm bytes. See Section 16.3.
UIE	See the DS12887 data sheet
SQWE	Square wave enable: See Section 16.3
DM	Data mode. DM = 0: BCD data format and DM = 1: Binary (hex) data format
24/12	1 for 24-hour mode and 0 for 12-hour mode
DSE	Daylight Saving Enable. If 1, enables the daylight saving. (The first Sunday in April and the last Sunday of October)

Figure 16-5. Some Major Bits of Register B

Setting the date

The following shows how to set the date to October 19th, 2004. Notice that when we initialize time or date, we need to set D7 of register B to 1.

```
;------------TURNING ON THE RTC
          MOV   R0,#10      ;R0=0AH,Reg A address
          MOV   A,#20H      ;010 in D6-D4 to turn on osc
          MOVX  @R0,A       ;send it to Reg A of DS12887
;--------------Setting the Time mode
          MOV   R0,#11      ;Reg B address
          MOV   A,#83H      ;BCD,24 hrs, daylight saving
          MOVX  @R0,A       ;send it to Reg B
;----------Setting the DATE
          MOV   R0,#07      ;load pointer for DAY OF MONTH
          MOV   A,#19H      ; DAY=19H (BCD numbers need H)
          MOVX  @R0,A       ;set DAY OF MONTH
          ACALL DELAY       ;
          MOV   R0,#08      ;point to MONTH
          MOV   A,#10H      ;10=OCTOBER.
          MOVX  @R0,A       ;set MONTH
          ACALL DELAY       ;
          MOV   R0,#09      ;point to YEAR address
          MOV   A,#04       ;YEAR=04 FOR 2004
          MOVX  @R0,A       ;set YEAR to 2004
          ACALL DELAY
          MOV   R0,#11      ;Reg B address
          MOV   A,#03       ;D7=0 of reg B to allow update
          MOVX  @R0,A       ;send it to reg B
```

RTCs setting, reading, displaying time and date

The following is a complete Assembly code for setting, reading, and displaying the time and date. The times and dates are sent to the screen via the serial port after they are converted from BCD to ASCII.

```
;----RTCTIME.ASM: SETTING TIME,READING AND DISPLAYING IT
         ORG 0
         ACALL DELAY_200ms ;RTC needs 200ms upon power-up
     SERIAL PORT SET-UP
         MOV   TMOD,#20H
         MOV   SCON,#50H
         MOV   TH1,#-3    ;9600
         SETB  TR1
;------------TURNING ON THE RTC
         MOV   R0,#10     ;R0=0AH,Reg A address
         MOV   A,#20H     ;010 in D6-D4 to turn on osc.
         MOVX  @R0,A      ;send it to Reg A of DS12887
;--------------Setting the Time mode
         MOV   R0,#11     ;Reg B address
         MOV   A,#83H     ;BCD, 24 hrs, daylight saving
         MOVX  @R0,A      ;send it to Reg B
;----------Setting the DATE
         MOV   R0,#07     ;load pointer for DAY OF MONTH
         MOV   A,#24H     ; DAY=24H (BCD numbers need H)
         MOVX  @R0,A      ;set DAY OF MONTH
         ACALL DELAY      ;
         MOV   R0,#08     ;point to MONTH
         MOV   A,#10H     ; 10=OCTOBER.
         MOVX  @R0,A      ;set MONTH
         ACALL DELAY      ;
         MOV   R0,#09     ;point to YEAR address
         MOV   A,#04      ;YEAR=04 FOR 2004
         MOVX  @R0,A      ;set YEAR to 2004
         ACALL DELAY
         MOV   R0,#11     ;Reg B address
         MOV   A,#03      ;D7=0 of reg B to allow update
         MOVX  @R0,A      ;send it to reg B
·--------READ Time(HH:MM:SS), CONVERT IT AND DISPLAY IT
OV1:     MOV   A,#20H     ;ASCII for SPACE
         ACALL SERIAL
         MOV   R0,#4      ;point to HR loc
         MOVX  A,@R0      ;read hours
         ACALL DISPLAY
         MOV   A,#20H     ;send out SPACE
         ACALL SERIAL
         MOV   R0,#2      ;point to minute loc
         MOVX  A,@R0      ;read minute
         ACALL DISPLAY
         MOV   A,#20H     ;send out SPACE
         ACALL SERIAL
```

```
          MOV   R0,#0       ;point to seconds loc
          MOVX  A,@R0       ;read seconds
          ACALL DISPLAY
          MOV   A,#0AH      ;send out CR
          ACALL SERIAL
          MOV   A,#0DH      ;send LF
          ACALL SERIAL
          SJMP  OV1         ;read and display forever
;---------SMALL DELAY
DELAY:
          MOV   R7,#250
D1:       DJNZ  R7,D1
          RET
;-------------CONVERT BCD TO ASCII AND SEND IT TO SCREEN
DISPLAY:
          MOV   B,A
          SWAP  A
          ANL   A,#0FH
          ORL   A,#30H
          ACALL SERIAL
          MOV   A,B
          ANL   A,#0FH
          ORL   A,#30H
          ACALL SERIAL
          RET
;-----------
SERIAL:
          MOV   SBUF,A
S1:       JNB   TI,S1
          CLR   TI
          RET
;------------
          END
```

The following shows how to read and display the date. You can replace the time display portion of the above program with the program below.

```
;--------READ DATE(YYYY:MM:MM), CONVERT IT AND DSIPLAY IT
OV2:      MOV   A,#20H      ;ASCII SPACE
          ACALL SERIAL
          MOV   A,#'2'      ;SEND OUT 2 (for 20)
          ACALL SERIAL
          MOV   A,#'0'      ;SEND OUT 0 (for 20)
          ACALL SERIAL
          MOV   R0,#09      ;point to year loc
          MOVX  A,@R0       ;read year
          ACALL DISPLAY
          MOV   A,#':'      ;SEND OUT : for yyyy:mm
          ACALL SERIAL
          MOV   R0,#08      ;point to month loc
          MOVX  A,@R0       ;read month
```

```
ACALL DISPLAY
ACALL DELAY
MOV   A,#':'      ;SEND OUT : for mm:dd
ACALL SERIAL
MOV   R0,#07      ;point to DAY loc
MOVX  A,@R0       ;read day
ACALL DISPLAY
ACALL DELAY
MOV   A,#' '      ;send out SPACE
ACALL SERIAL
ACALL DELAY
MOV   A,#' '      ;send out SPACE
ACALL SERIAL
ACALL DELAY
MOV   A,#0AH      ;send out LF
ACALL SERIAL
MOV   A,#0DH      ;send CR
ACALL SERIAL
ACALL DELAY
LJMP OV2          ;display date forever
```

UNIT – IV MICROCONTROLLER

Architecture of 8051 – Special Function Registers(SFRs) - I/O Pins Ports and Circuits - Instruction set - Addressing modes - Assembly language programming.

4.1: Introduction:

Microcontroller:

- ✓ A Microcontroller is a single chip computer.
- ✓ A CPU with all the peripherals like RAM, ROM, I/O Ports, Timers, and ADCs etc. on the same chip.

  For Ex: Motorola 6811, Intel 8051, Zilog Z8 and PIC 16X etc…

Microprocessor:

- ✓ A CPU built into a single VLSI chip is called a microprocessor.
- ✓ It is a general-purpose device and additional external circuitry is added to make it a microcomputer.
- ✓ The microprocessor contains arithmetic logic unit (ALU), Control unit, Instruction register, Program counter (PC), clock circuit (internal or external), reset circuit (internal or external) and registers.
- ✓ But the microprocessor has no on chip I/O Ports, Timers, Memory etc.
- ✓ For example, Intel 8085 is an 8-bit microprocessor and Intel 8086/8088 a 16-bit microprocessor.
- ✓ The block diagram of the Microprocessor is shown in Fig.1

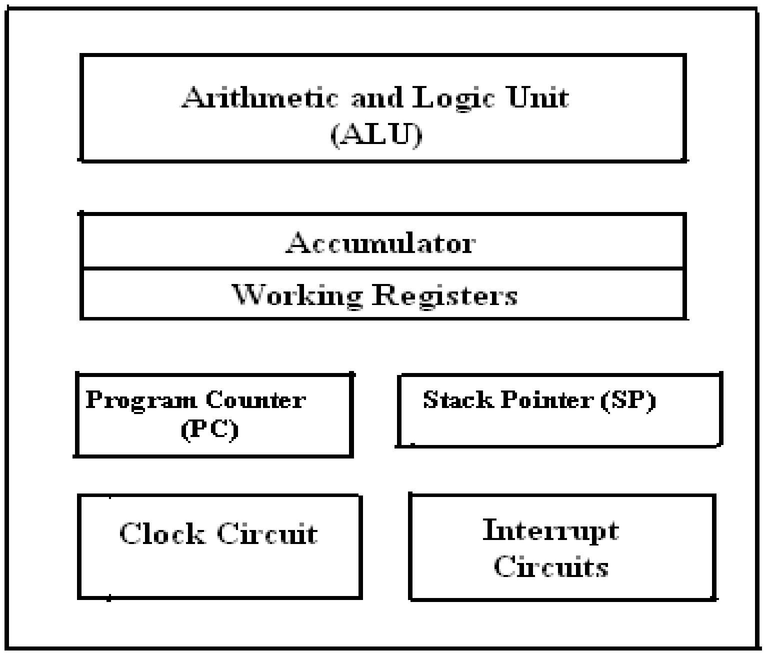


Fig.1: Block diagram of a Microprocessor.

MICROCONTROLLER :

- ✓ A microcontroller is an integrated single chip, which consists of CPU, RAM, EPROM/PROM/ROM, I/O ports, timers, interrupt controller.
- ✓ For example, Intel 8051 is 8-bit microcontroller and Intel 8096 is 16-bit microcontroller.
- ✓ The block diagram of Microcontroller is shown in Fig.2.

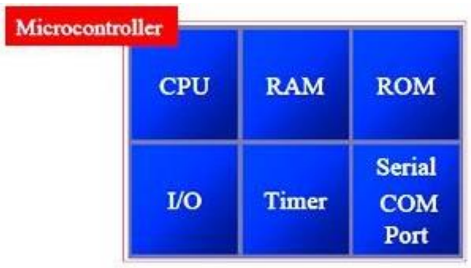


Fig.2.Block Diagram of a Microcontroller

Distinguish between Microprocessor and Microcontroller

S.No	Microprocessor	Microcontroller
1	A microprocessor is a general purpose device.	A microcontroller is a dedicated chip which is also called as single chip computer.
2	A microprocessor does not contain on chip I/O Ports, Timers, Memories etc.	A microcontroller includes RAM, ROM, serial and parallel interface, timers, interrupt circuitry in a single chip.
3	Microprocessor is used as the CPU in microcomputer system.	Microcontroller is used to perform control-oriented applications.
4	Microprocessor instructions are nibble or byte addressable	Microcontroller instructions are both bit addressable as well as byte addressable.
5	Microprocessor based system design is complex and expensive	Microcontroller based system design is simple and cost effective
6	The Instruction set of microprocessor is complex with large number of instructions.	The instruction sets are simple with less number of instructions.

****

4.2 : INTEL 8051 MICRCONTROLLER:

Draw the architectural block diagram of 8051 microcontroller and explain. (NOV 2011, MAY 2010, NOV 2009, NOV2008, May 2008, MAY 2007, MAY 2006, NOV 2016, May 2016)

Features of 8051 Microcontroller:

The 8051 is an 8-bit Controller:

- ✓ The CPU can works on only 8 bits of data at a time
- ✓ The 8051 has
  - 128 bytes of RAM
  - 4K bytes of on-chip ROM
  - Two timers
  - One serial port
  - Four I/O ports, each 8 bits wide
  - 6 interrupt sources

ARCHITECTURE & BLOCK DIAGRAM OF 8051 MICROCONTROLLER:

- ✓ It has hardware architecture with RISC (Reduced Instruction Set Computer) concept.
- ✓ The block diagram of 8051 microcontroller is shown in Fig 3.
- ✓ 8051 has 8-bit ALU.
- ✓ ALU can perform all the 8-bit arithmetic and logical operations in one machine cycle.
- ✓ The ALU is associated with two registers A & B

A and B Registers:

- ✓ The A and B registers are special function registers.
- ✓ A & B registers hold the results of many arithmetic and logical operations of 8051.
- ✓ The A register is also called the **Accumulator.**
- ✓ A register is used as a general register to accumulate the results of a large number of instructions.
- ✓ By default, it is used for all mathematical operations and data transfer operations between CPU and external memory.
- ✓ The B register is mainly used for multiplication and division operations along with A register.
  - ▪ Ex: MUL AB : DIV AB.
- ✓ It has no other function other than as a store data.

R registers:

- ✓ "R" registers are a set of eight registers that are named R0, R1, etc. up to R7.
- ✓ These registers are used as auxiliary registers in many operations.
- ✓ The "R" registers are also used to temporarily store values.

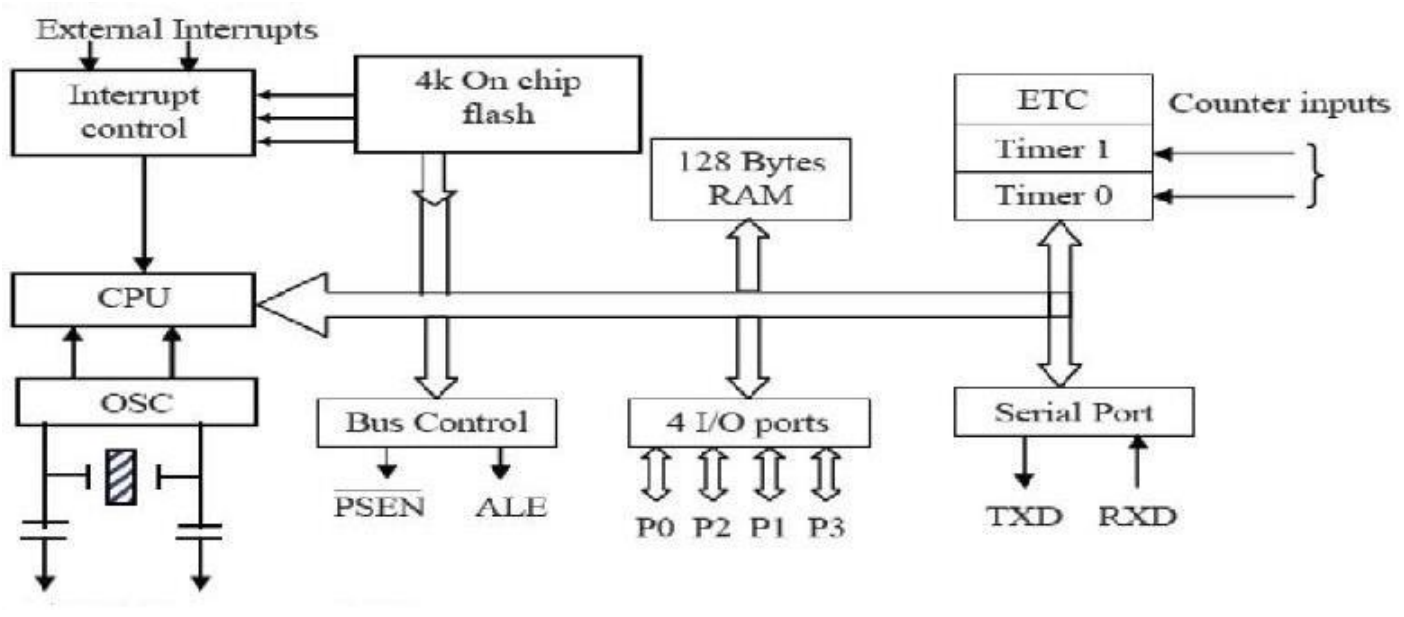


Fig.3. Block Diagram of 8051 Microcontroller

Program Counter (PC) :

- ✓ 8051 has a 16-bit program counter.
- ✓ The program counter holds address of the next instruction to be executed.
- ✓ After execution of one instruction, the program counter is incremented.

Data Pointer Register (DPTR):

- ✓ It is a 16-bit register which is the only user-accessible.
- ✓ DPTR is used to point the data. 8051 will access external memory at the address indicated by DPTR.
- ✓ DPTR can also be used as two 8-registers DPH and DPL.

Stack Pointer Register (SP) :

- ✓ It is an 8-bit register which stores the address of the stack top.
- ✓ When a value is pushed onto the stack, the 8051 first increments the value of SP and then stores the value.
- ✓ Similarly when a value is popped off the stack, the 8051 returns the value from the memory location indicated by SP and then decrements the value of SP.
- ✓ Since the SP is only 8-bit wide.
- ✓ It is incremented or decremented by two.
- ✓ SP is modified directly by the 8051 by six instructions: PUSH, POP, ACALL, LCALL, RET, and RETI.
- ✓ It is also used intrinsically whenever an interrupt is triggered.

Block Diagram

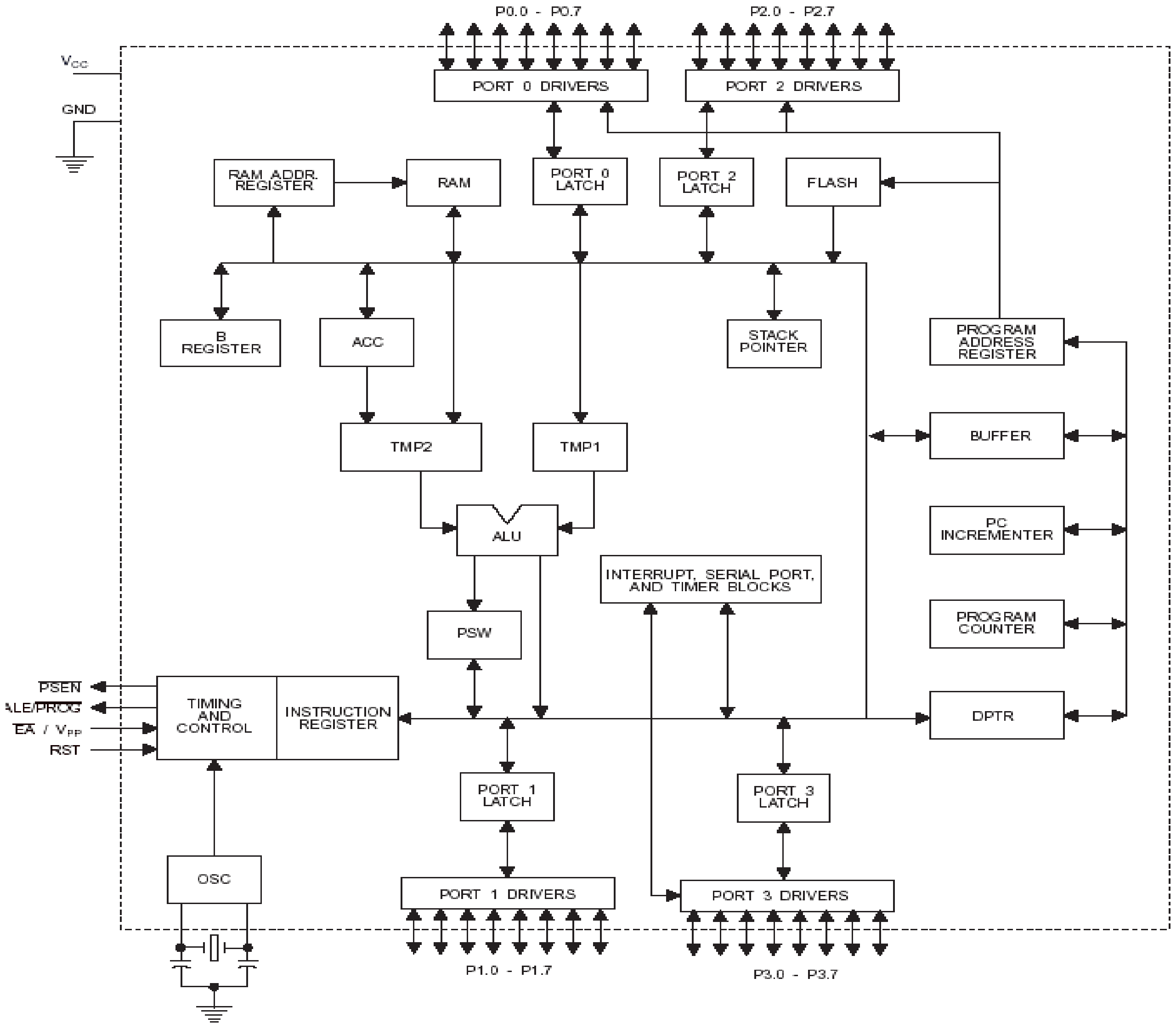


Fig 3a: Internal architecture diagram of 8051 Microcontroller

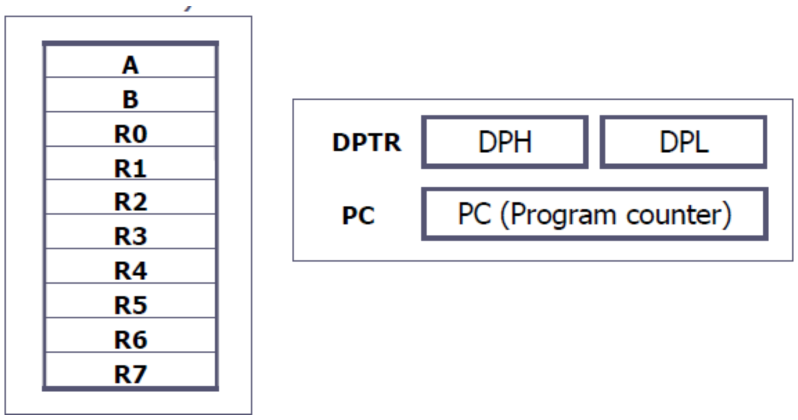


Fig: Structure of registers

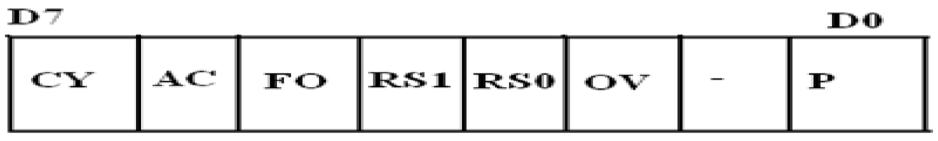
STACK in 8051 Microcontroller:

- ✓ The stack is a part of RAM used by the CPU to store information temporarily.
- ✓ This information may be either data or an address.
- ✓ The register used to access the stack is called the Stack pointer (SP).
- ✓ SP is an 8-bit register. So, it can take values of 00 to FF H.
- ✓ When the 8051 is powered up, the SP register contains the value 07.i.e the RAM location value 08 is the first location being used for the stack by the 8051 controller.
- ✓ There are two important instructions to handle stack. One is the PUSH and the other is the POP.
- ✓ The loading of data from CPU registers to the stack is done by PUSH.
- ✓ The loading of the contents of the stack back into a CPU register is done by POP.

Program Status Register (PSW):

Give PSW of 8051 and describe the use of each bit in PSW. (NOV 2015)

- ✓ The 8051 has an 8-bit PSW register which is also known as Flag register.
- ✓ In the 8-bit register only 6-bits are used by 8051.The two unused bits are user definable bits.
- ✓ In the 6-bits, four of them are conditional flags. They are Carry –CY, Auxiliary Carry-AC, Parity-P, and Overflow-OV.
- ✓ These flag bits indicate some conditions of result after an instruction was executed.



D7							D0
CY	AC	FO	RS1	RS0	OV	-	P

- ✓ The bits PSW3 and PSW4 are denoted as RS0 and RS1.
- ✓ These bits are used to select the bank registers of the RAM location.
- ✓ The meaning of various bits of PSW register is shown below.

CY	PSW.7	Carry Flag
AC	PSW.6	Auxiliary Carry Flag
FO	PSW.5	Flag 0 available for general purpose
RS1	PSW.4	Register Bank select bit 1
RS0	PSW.3	Register bank select bit 0
OV	PSW.2	Overflow flag
---	PSW.1	User definable flag
P	PSW.0	Parity flag .set/cleared by hardware.

- ✓ The selection of the register Banks and their addresses are given below.

RS1	RS0	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

RAM & ROM:

- ✓ The 8051 microcontroller has 128 bytes of Internal RAM and 4KB of on chip ROM.
- ✓ The RAM is also known as Data memory and the ROM is known as program (Code) memory.
- ✓ Code memory holds program that is to be executed.
- ✓ Program Address Register holds address of the ROM/ Flash memory.
- ✓ Data Address Register holds address of the RAM.

I/O ports:

- ✓ The 8051 microcontroller has 4 parallel I/O ports, each of 8-bits.
- ✓ So, it provides 32 I/O lines for connecting the microcontroller to the peripherals.
- ✓ The four ports are P0 (Port 0), P1 (Port1), P2 (Port 2) and P3 (Port3).

****

4.3 : Memory organization :

Explain in detail the internal memory organization of 8051 microcontroller (NOV 2014, May 2012, NOV 2011, NOV 2010, May 2010, MAY 2009, NOV 2008, NOV 2007)

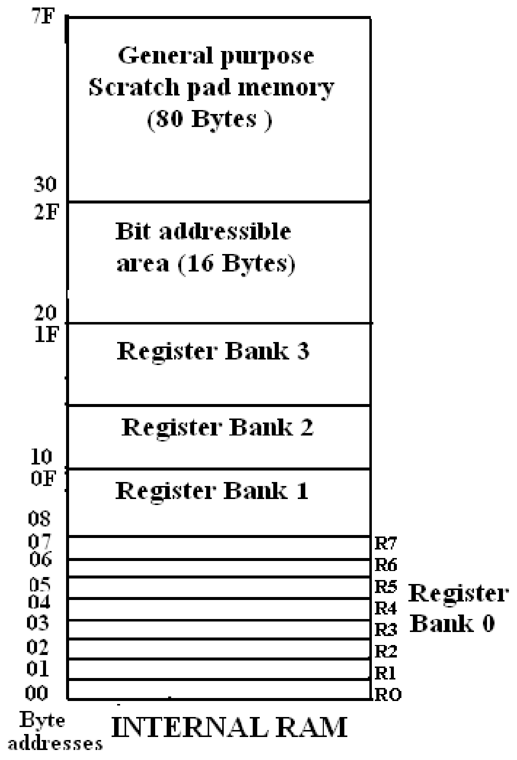
- ✓ The 8051 microcontroller has 128 bytes of Internal RAM and 4kB of on chip ROM.
- ✓ The RAM is also known as Data memory and the ROM is known as program (Code) memory.
- ✓ Code memory holds the actual 8051 program to be executed.
- ✓ In 8051, memory is limited to 64KB.
- ✓ Code memory may be found on-chip, as ROM or EPROM.
- ✓ It may also be stored completely off-chip in an external ROM / EPROM.
- ✓ The 8051 has only 128 bytes of Internal RAM but it supports 64KB of external RAM.
- ✓ Since the memory is off-chip, it is not as flexible for accessing and is also slower.

Structure of Internal RAM OF 8051(Data Memory) :

Explain the Data memory structure of 8051. (NOV 2011)

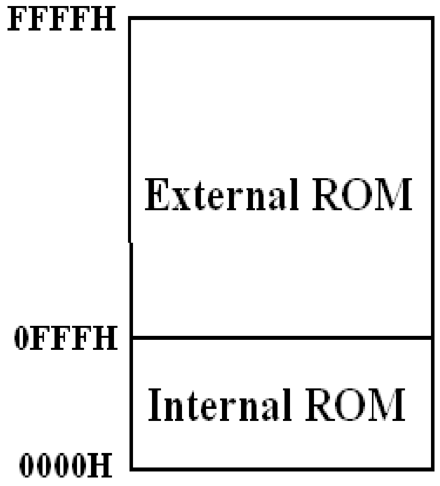
- ✓ Internal RAM is found on-chip on the 8051. So it is the fastest RAM available.
- ✓ It is flexible in terms of reading, writing and modifying its contents.
- ✓ Internal RAM is volatile.
- ✓ When the 8051 is reset, internal RAM is cleared.
- ✓ The 128 bytes of internal RAM is organized as below.
- ✓ Four register banks (Bank0, Bank1, Bank2 and Bank3) each of 8-bits (total 32 bytes).
- ✓ The default bank register is Bank0.
- ✓ The remaining Banks are selected with the help of RS0 and RS1 bits of PSW Register.
- ✓ 16 bytes of bit addressable area and
- ✓ 80 bytes of general purpose area (Scratch pad memory) of internal RAM as shown in the diagram below.
- ✓ This area is utilized by the microcontroller as a storage area for the operating stack.
- ✓ The 32 bytes of RAM from address 00 H to 1FH are used as working registers organized as four banks of eight registers each.
- ✓ The registers are named as R0-R7.
- ✓ Each register can be addressed by its name or by its RAM address.

For example: MOV A, R7 or MOV R7,#05H



Structure of Internal ROM (On –chip ROM / Program Memory / Code Memory):

- ✓ The 8051 microcontroller has 4KB of on chip ROM, but it can be extended up to 64KB.
- ✓ This ROM is also called program memory or code memory.
- ✓ The CODE segment is accessed using the program counter (PC) for opcode fetches and by DPTR for data.
- ✓ The external ROM is accessed when the EA pin is connected to ground or the contents of program counter exceeds 0FFFH.
- ✓ When the Internal ROM address is exceeded the 8051 automatically fetches the code bytes from the external program memory.



****

4.4: SPECIAL FUNCTION REGISTERS (SFRs)

Write the available special function registers in 8051. Explain each register with its format and functions. (April 2017, NOV 2015)

- ✓ In 8051 microcontroller, there are registers which uses the RAM addresses from 80h to FFh.
- ✓ They are used for certain specific operations. These registers are called Special Function Registers (SFRs).
- ✓ Most of SFRs are bit addressable and other few registers are byte addressable.
- ✓ In these SFRs, some of them are related to I/O ports (P0, P1, P2 and P3) and some of them are for control operations (TCON, SCON & PCON).
- ✓ Remaining are the auxiliary SFRs, that they don't directly configure the 8051.
- ✓ The list of SFRs and their functional names are given below.

- ✓ **The * indicates the bit addressable SFRs**

S.No	Symbol		Name of SFR	Address (Hex)
1	ACC*		Accumulator	**0E0**
2	B*		B-Register	**0F0**
3	PSW*		Program Status word register	**0DO**
4	SP		Stack Pointer Register	**81**
5	DPTR	DPL	Data pointer low byte	**82**
		DPH	Data pointer high byte	**83**
6	P0*		Port 0	**80**
	P1*		Port 1	**90**
8	P2*		Port 2	**0A**
9	P3*		Port 3	**0B**
10	IP*		Interrupt Priority control	**0B8**
11	IE*		Interrupt Enable control	**0A8**
12	TMOD		Timer mode register	**89**
13	TCON*		Timer control register	**88**
14	TH0		Timer 0 Higher byte	**8C**
15	TL0		Timer 0 Lower byte	**8A**
16	TH1		Timer 1Higher byte	**8D**
17	TL1		Timer 1 lower byte	**8B**
18	SCON*		Serial control register	**98**
19	SBUF		Serial buffer register	**99**
20	PCON		Power control register	**87**

Table: Special Function Registers

4.5 : Input / Output (I/O) ports :

What are the I/O ports available in 8051 and explain? (MAY 2014, NOV 2012, MAY2010, NOV2009)
Enumerate about the ports available in 8051. (MAY 2014)
Explain parallel port architecture of 8051 microcontroller. (NOV 2012)
Explain each PORT circuitry available in 8051. (NOV 2007)

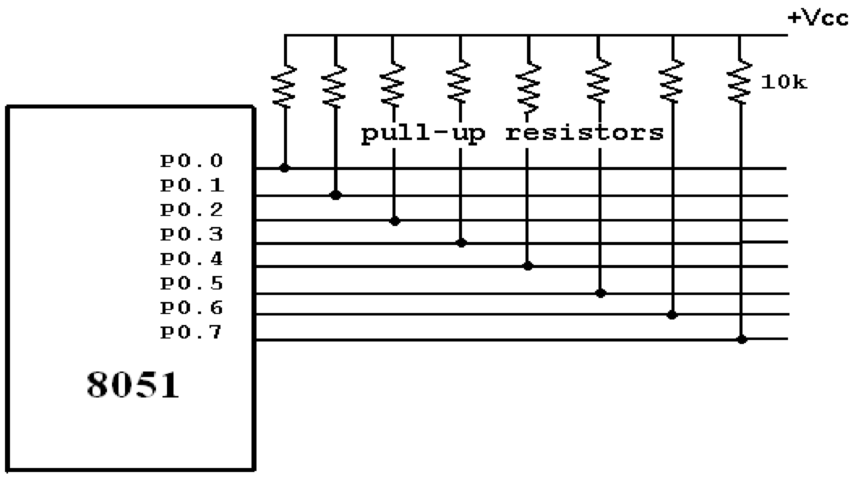
PARALLEL I /O PORTS :

- ✓ The 8051 microcontroller has 4 parallel I/O ports, each of 8-bits.
- ✓ So, it provides 32 I/O lines for connecting the microcontroller to the peripherals.
- ✓ The four ports are P0 (Port 0), P1 (Port1), P2 (Port 2) and P3 (Port3).
- ✓ When resetting, all the ports are output ports.
- ✓ In order to make them input, all the ports must be set. This is normally done by the instruction "SETB".
- ✓ Ex:

```
MOV A,#0FFH      ; A = FF
MOV P0, A        ; Make P0 an input port
```

PORT 0:

- ✓ Port 0 is an 8-bit I/O port with dual purpose.
- ✓ Port 0 does not have pull-up resistors internally, so pull-up resistors are to be connected externally as shown in the figure.



Dual role of port 0:

- ✓ If external memory is used, port 0 is used as lower order address/data bus (AD_0-AD_7), otherwise port 0 is used as input or output port.
- ✓ The 8051 multiplexes address and data through port 0 to reduce the pins.
- ✓ ALE indicates whether P0 has address or data.
- ✓ When ALE = 0, it provides data D0-D7, and when ALE =1, it provides address.

Port 0 circuit:

- ✓ Port-0 can be configured as a bidirectional I/O port or it can be used for lower order address/data lines.
- ✓ When control is '1', the port is used for address/data interfacing.
- ✓ When the control is '0', the port can be used as a bidirectional I/O port.

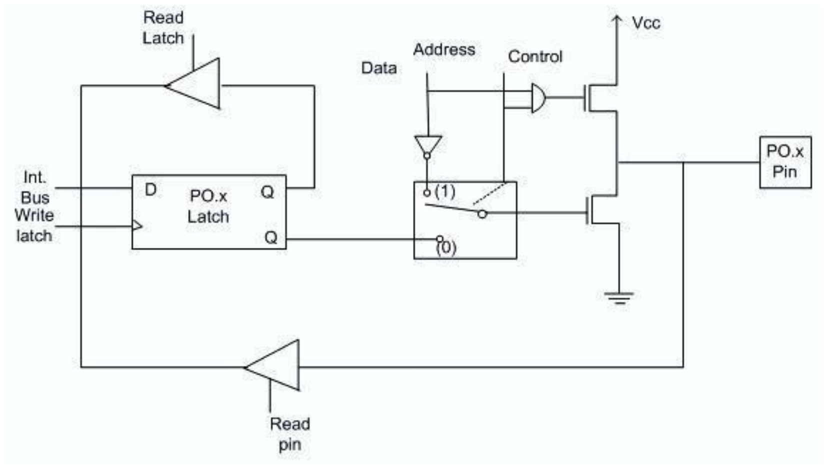


Fig: Port 0 Circuit

Port 1:

- ✓ Port 1 occupies a total of 8 pins. It has no dual application and acts only as I/O port.
- ✓ This port does not need any pull-up resistors because pull-up resistors connected internally.

Port 1 circuit:

- ✓ Port-1 does not have any alternate function i.e. it is dedicated solely for I/O interfacing.

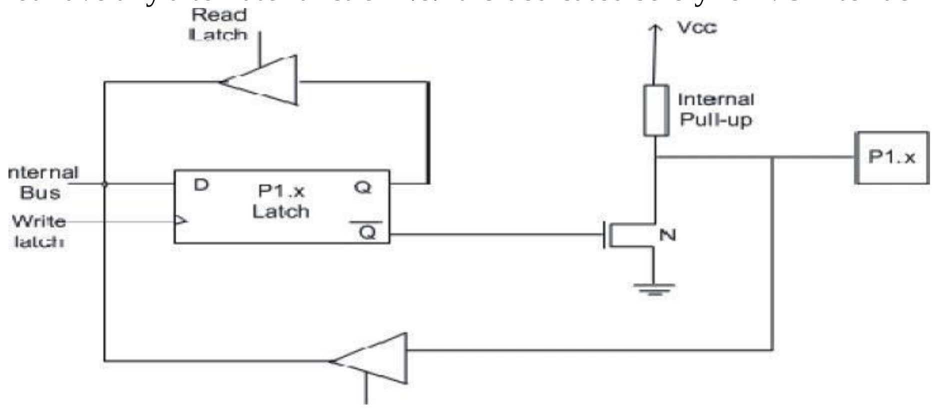


Fig: Port 1 Circuit

Port 2:

- ✓ Port 2 is an 8 bit parallel port. It can be used as input or output port.
- ✓ As this port is provided with internal pull-up resistors, it does not need any external pull-up resistors.

Dual role of port 2:

- ✓ Port 2 lines are also associated with the higher order address lines A8-A15.

- ✓ In 8051-based systems, port 2 is used along with P0 to provide the 16-bit address for the external memory.
- ✓ 8031/8051 is capable of accessing 64K bytes of external memory.
- ✓ When control is '1', the port is used for address interfacing.
- ✓ When the control is '0', the port can be used as a bidirectional I/O port.

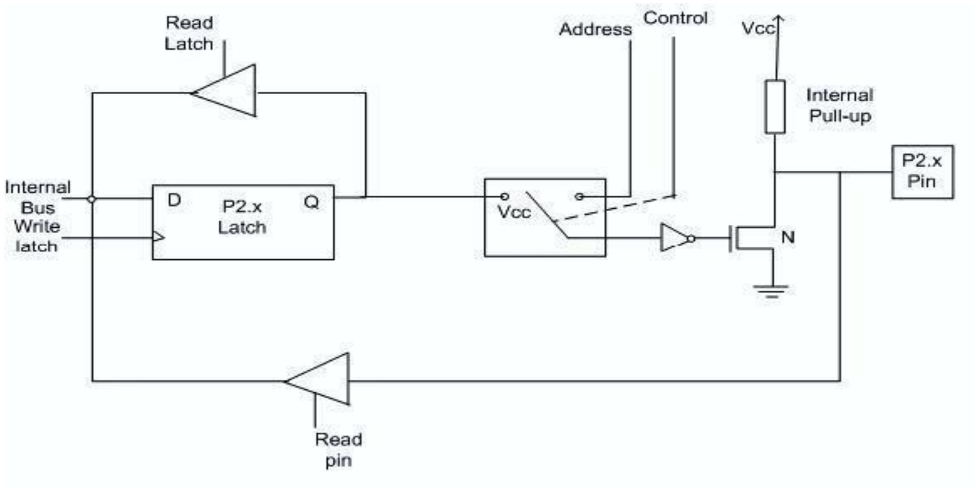


Fig: Port 2 Circuit

PORT 3:

Explain the use of port 3 of 8051 for interrupt signals. (NOV 2009)

- ✓ Port3 is an 8-bit parallel port with dual function.
- ✓ The port pins can be used for I/O operations as well as for control operations.
- ✓ Port 3 also do not need any external pull-up resistors as they are provided internally.
- ✓ The details of additional operations are given below in the table.

S.No	Port 3 bit	Pin No	Function
1	P3.0	10	RxD
2	P3.1	11	TxD
3	P3.2	12	$\overline{INT0}$
4	P3.3	13	$\overline{INT1}$
5	P3.4	14	T0
6	P3.5	15	T1
7	P3.6	16	$\overline{WR}$
8	P3.7	17	$\overline{RD}$

Table: PORT 3 alternate functions

Alternate Functions of Port 3:

- ✓ P3.0 and P3.1 are used for the RxD (Receive Data) and TxD (Transmit Data) as serial communications signals. Bits P3.2 and P3.3 are meant for external interrupts.
- ✓ Bits P3.4 and P3.5 are used for Timers 0 and Timer 1.
- ✓ P3.6 and P3.7 are used to provide the write and read signals of external memories connected in 8031/ 8051 based systems.

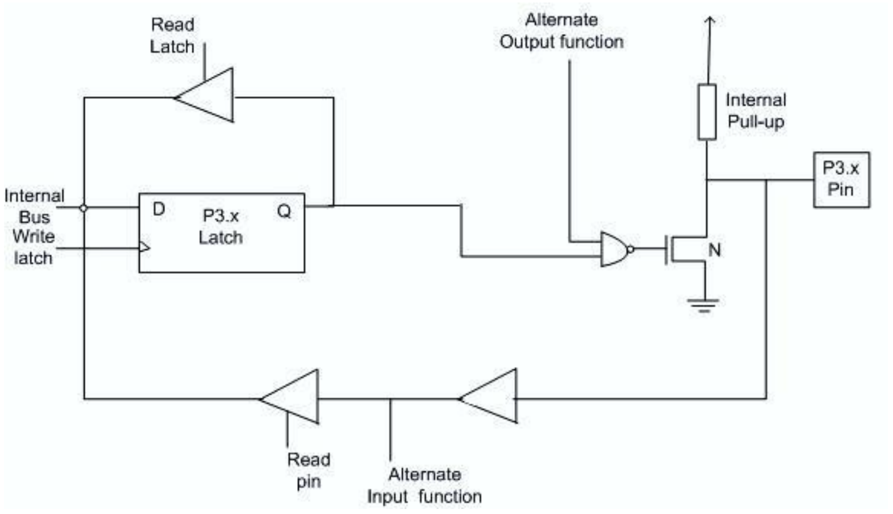


Fig: Port 3 Circuit

****

<u>4.6: Interrupt of 8051 Microcontroller:</u>

Explain interrupt structure of 8051 microcontroller. (NOV 2011, MAY 2009)

Interrupt Structure:

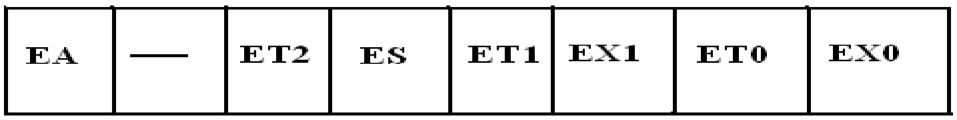
- ✓ An interrupt is an external or internal event that disturbs the microcontroller to inform it that a device needs its service.
- ✓ The program which is associated with the interrupt is called the **interrupt service routine** (ISR) or **interrupt handler**.
- ✓ Upon receiving the interrupt signal, the microcontroller finishes current operation and saves the PC on stack.
- ✓ Jumps to a fixed location in memory depending on type of interrupt.
- ✓ Starts to execute the interrupt service routine until RETI.

- ✓ Upon executing the RETI the microcontroller returns to the place where it was interrupted. Get pop PC from stack.
- ✓ The 8051 microcontroller has **FIVE** interrupts in addition to Reset. They are
  - Timer 0 overflow Interrupt
  - Timer 1 overflow Interrupt
  - External Interrupt 0(INT0)
  - External Interrupt 1(INT1)
  - Serial Port Interrupt
- ✓ Each interrupt has a specific place in code memory where program execution begins.

External Interrupt 0:	0003 H
Timer 0 overflow:	000B H
External Interrupt 1:	0013 H
Timer 1 overflow:	001B H
Serial Interrupt :	0023 H

- ✓ Upon reset all Interrupts are disabled & do not respond to the Microcontroller.
- ✓ These interrupts must be enabled by software. This is done by an 8-bit register called Interrupt Enable Register (IE).

Interrupt Enable Register :



EA	——	ET2	ES	ET1	EX1	ET0	EX0

- EA : Global enable/disable. To enable the interrupts, this bit must be set high.
- --- : Undefined-reserved for future use.
- ET2 : Enable /disable Timer 2 overflow interrupt.
- ES : Enable/disable Serial port interrupts.
- ET1 : Enable /disable Timer 1 overflow interrupt.
- EX1 : Enable/disable External interrupt1.
- ET0 : Enable /disable Timer 0 overflow interrupt.
- EX0 : Enable/disable External interrupt0

- ✓ Upon reset, the interrupts have the following priority from top to down. The interrupt with the highest PRIORITY gets serviced first.

    1. External interrupt 0 (INT0)
    2. Timer interrupt0 (TF0)
    3. External interrupt 1 (INT1)
    4. Timer interrupt1 (TF1)
    5. Serial communication (RI+TI)

- ✓ Priority can also be set to "high" or "low" by 8-bit IP register.

Interrupt priority register:

——	——	PT2	PS	PT1	PX1	PT0	PX0

- IP.7: reserved
- IP.6: reserved
- IP.5: Timer 2 interrupt priority bit (8052 only)
- IP.4: Serial port interrupt priority bit
- IP.3: Timer 1 interrupt priority bit
- IP.2: External interrupt 1 priority bit
- IP.1: Timer 0 interrupt priority bit
- IP.0: External interrupt 0 priority bit

****

4.7 : TIMERS in 8051 Microcontrollers:

Explain in detail the timer of 8051 and their associated registers. (NOV 2009, MAY2009)

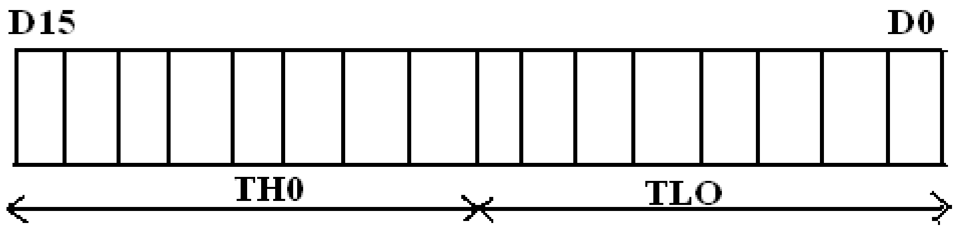
How are the timers of 8051 used to produce time delay in timer mode? (NOV 2011)

- ✓ The 8051 microcontroller has two 16-bit timers Timer 0 (T0) and Timer 1(T1), which can be used either to generate accurate time delays or as event counters.

- ✓ These timers are accessed as two 8-bit registers TLO, THO & TL1, TH1 because the 8051 microcontroller is 8-bit architecture.

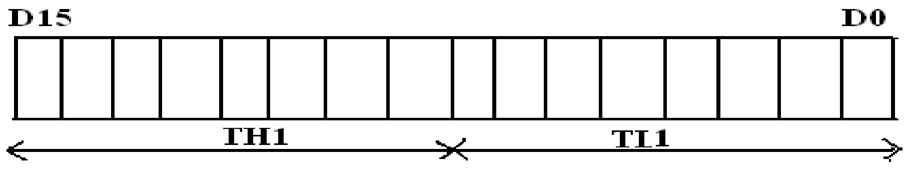
TIMER 0 :

- ✓ The Timer 0 is a 16-bit register and can be treated as two 8-bit registers (TL0 & TH0).
- ✓ These registers can be accessed similar to other registers like A, B or R1, R2, R3 etc…
- ✓ Ex : The instruction MOV TL0,#07; Moves the value 07 into lower byte of Timer0.



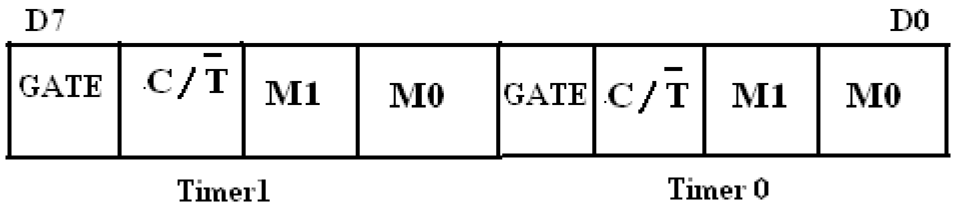
TIMER 1 :

- ✓ The Timer 1 is also a 16-bit register and can be treated as two 8-bit registers (TL1 & TH1).
- ✓ These registers can be accessed similar to any other registers like A, B or R1, R2, R3 etc…
- ✓ Ex : The instruction MOV TL1,#05: Moves the value 05 into lower byte of Timer1.



TMOD Register:

- ✓ The various operating modes of both the timers T0 and T1 are set by an 8-bit register called TMOD register**.**
- ✓ In this TMOD register the lower 4-bits are meant for Timer 0 and the higher 4-bits are meant for Timer1.



GATE:

- ✓ This bit is used to start or stop the timers by hardware.
- ✓ When GATE= 1, the timers can be started / stopped by the external sources.

- ✓ When GATE= 0, the timers can be started or stopped by software instructions like SETB TR0 or SETB TR1.

$C/\overline{T}$ **(Clock/Timer) :**

- ✓ This bit decides whether the timer is used as delay generator or event counter.
- ✓ When $C/\overline{T}$ **= 0,** the timer is used as delay generator and if $C/\overline{T}$ =1 the timer is used as an event counter.
- ✓ The clock source for the time delay is the crystal frequency of 8051.

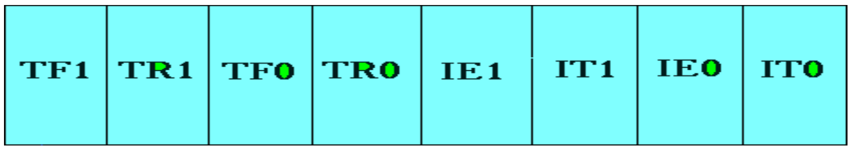
M1, M0 (Mode):

- ✓ These two bits are the timer mode bits.
- ✓ The timers of the 8051 can be configured in three modes as Mode0, Mode1 and Mode2.
- ✓ The selection and operation of the modes is shown below.

S.No	M0	M1	Mode	Operation
1	0	0	0	**13-bit Timer mode.** 8-bit Timer/counter THx with TLx as 5-bit prescaler
2	0	1	1	**16-bit Timer mode.**16-bit timer /counter THx and TLx are cascaded. There is no presacler
3	1	0	2	**8-bit auto reload.** 8-bit auto reload timer/counter. THx holds a value which is to be reloaded TLx each time it overflows
4	1	1	3	**13-bit Timer mode.** 8-bit Timer/counter THx with TLx as 5-bit prescaler

TCON (Timer control register)

- ✓ TCON (timer control) register is an 8-bit register. TCON register is a bit-addressable register.



TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

- TF1: Timer 1 overflow flag.
- TR1: Timer 1 run control bit.
- TF0: Timer 0 overflow flag.
- TR0: Timer 0 run control bit.
- IE1: External interrupt 1 edge flag.
- IT1: External interrupt 1 type flag.

- IE0: External interrupt 0 edge flag.
- IT0: External interrupt 0 type flag.

****

4.8: PIN Diagram of 8051 Microcontroller:

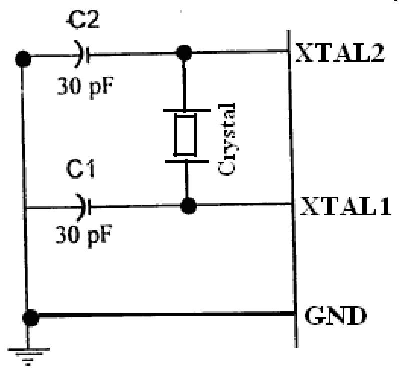
Explain Pin details of 8051 microcontroller. (MAY 2006)

Describe the functions of the following signals in 8051. RST, EA, PSEN and ALE. (NOV 2015)

- ✓ The 8051 microcontroller is available as a 40 pin DIP chip and it works at +5 volts DC.
- ✓ Among the 40 pins, a total of 32 pins are allotted for the four parallel ports P0, P1, P2 and P3 i.e each port occupies 8-pins.
- ✓ The remaining pins are VCC, GND, XTAL1, XTAL2, RST, EA ,PSEN.

XTAL1, XTAL2:

- ✓ These two pins are connected to Quartz crystal oscillator which runs the on-chip oscillator.
- ✓ The quartz crystal oscillator is connected to the two pins along with a capacitor of 30pF as shown in the circuit.
- ✓ If use a source other than the crystal oscillator, it will be connected to XTAL1 and XTAL2 is left unconnected.



RST:

- ✓ The RESET pin is an input pin and it is an active high pin.
- ✓ When a high pulse is applied to this pin, the microcontroller will reset and terminate all activities.
- ✓ Upon reset all the registers will reset to 0000 Value and SP register will reset to 0007 value.

$\overline{\text{EA}}$ (External Access):

- ✓ This pin is an active low pin.

- ✓ This pin is connected to ground when microcontroller is accessing the program code stored in the external memory.
- ✓ This pin is connected to Vcc when it is accessing the program code in the on chip memory.

$\overline{\text{PSEN}}$ (Program Store Enable):

- ✓ This is an output pin which is active low.
- ✓ When the microcontroller is accessing the program code stored in the external ROM, this pin is connected to the OE (Output Enable) pin of the ROM.

ALE (Address latch enable):

- ✓ This is an output pin, which is active high**.**
- ✓ This ALE pin will demultiplex the address and data bus.
- ✓ When the pin is high, the Address/ Data bus will act as address bus, otherwise the Address/ Data bus will act as Data bus.

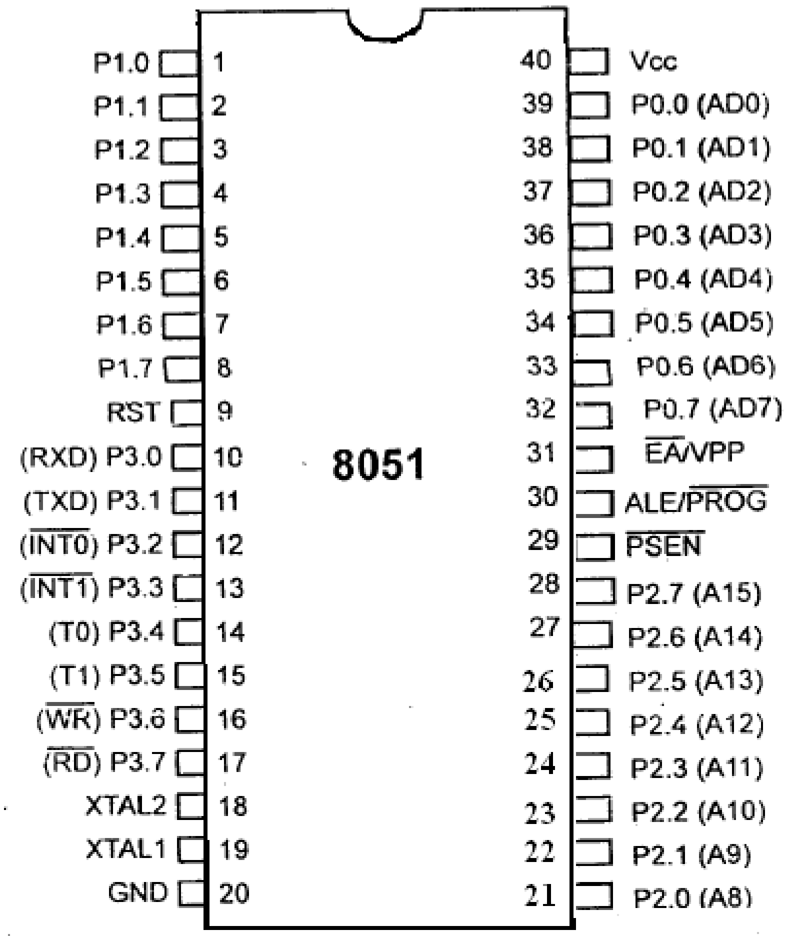


Figure: Pin diagram of 8051

P0.0- P0.7(AD0-AD7) :

- ✓ The port 0 pins multiplexed with Address/data pins.
- ✓ If the microcontroller is accessing external memory, these pins will act as address/data pins, otherwise they are used for Port 0 pins.

P2.0- P2.7 (A8-A15) :

- ✓ The port2 pins are multiplexed with the higher order address pins**.**
- ✓ When the microcontroller is accessing external memory, these pins provide the higher order address byte, otherwise they act as Port 2 pins.

P1.0- P1.7 :

- ✓ These 8-pins are dedicated to perform input or output port operations.

P3.0- P3.7:

- ✓ These 8-pins are meant for Port3 operations and also for some control operations like read, Write, Timer0, Timer1, INT0, INT1, RxD and TxD.

****

4.9 : ADDRESSING MODES OF 8051 :

Explain different types addressing modes of 8051 microcontroller. (NOV 2008, NOV 2015, April 2017)

- ✓ The way in which the data operands are specified is known as the addressing modes. There are various methods of denoting the data operands in the instruction.
- ✓ The 8051 microcontroller supports 5 addressing modes. They are
  1. Immediate addressing mode
  2. Direct Addressing mode
  3. Register addressing mode
  4. Register indirect addressing mode
  5. Indexed addressing mode

Immediate addressing mode:

- ✓ The addressing mode in which the data operand is a constant and it is a part of the instruction itself is known as Immediate addressing mode.
- ✓ Normally the data must be preceded by a # sign.
- ✓ This addressing mode can be used to transfer the data into any of the registers including DPTR.

Examples:

- ▪ MOV A, # 27 H : The data (constant) 27 is moved to the accumulator register

- ADD R1, #45 H : Add the constant 45 to the contents of the accumulator
- MOV DPTR, # 8245H : Move the data 8245 into the data pointer register.

Direct addressing mode:

- ✓ In the addressing mode, the data operand is in the RAM location (00 -7FH) and the address of the data operand is given in the instruction.
- ✓ The direct addressing mode uses the lower 128 bytes of Internal RAM and the SFRs

Examples:

- MOV R1, 42H : Move the contents of RAM location 42 into R1 register
- MOV 49H, A : Move the contents of the accumulator into the RAM location 49.
- ADD A, 56H : Add the contents of the RAM location 56 to the accumulator

Register addressing mode:

- ✓ In the addressing mode, the data operands are available in the registers.

Examples:

- MOV A,R0 : Move the contents of the register R0 to the accumulator
- MOV P1, R2 :Move the contents of the R2 register into port 1
- MOV R5, R2 : This is invalid. The data transfer between the registers is not allowed.

Register Indirect addressing mode:

- ✓ In the addressing mode, a register is used as a pointer to the data memory block.

Examples:

- MOV A,@ R0 :Move the contents of RAM location whose address is in R0 into **A** (accumulator)
- MOV @ R1 , B : Move the contents of B into RAM location whose address is held by R1
- When R0 and R1 are used as pointers, they must be preceded by @ sign

- ✓ **Advantage: It makes accessing the data more dynamic than static as in the case of direct addressing mode.**

Indexed addressing mode:

- ✓ This addressing mode is used in accessing the data elements of lookup table entries, located in program ROM.

Example: MOVC A, @ A+DPTR

- The 16-bit register DPTR and register A are used to form the address of the data element stored in on-chip ROM.

****

4.10 : Instructions Set of 8051:

Discuss in detail the 8051 instruction set. (NOV 2008)

4.10.1:Arithmetic instructions:

With example, explain arithmetic instructions in 8051 microcontroller. (NOV 2012)

- ✓ ADD
  - 8-bit addition between the accumulator (A) and a second operand.
  - The result is always in the accumulator.
  - The CY flag is set/reset appropriately.
- ✓ ADDC
  - 8-bit addition between the accumulator, a second operand and the previous value of the CY flag.
  - Useful for 16-bit addition in two steps.
  - The CY flag is set/reset appropriately.
- ✓ DAA
  - Decimal adjust the accumulator.
  - Format the accumulator into a proper 2 digit packed BCD number.
  - Operates only on the accumulator.
  - Works only after the ADD instruction.
- ✓ SUBB
  - Subtract with Borrow.
  - Subtract an operand and the previous value of a borrow (carry) flag from the accumulator.
  - A ← A - <operand> - CY.
  - The result is always saved in the accumulator.
  - The CY flag is set/reset appropriately.
- ✓ INC
  - Increment the operand by one.
  - The operand can be a register, a direct address, an indirect address, the data pointer.
- ✓ DEC
  - Decrement the operand by one.
  - The operand can be a register, a direct address, an indirect address.

- ✓ MUL AB / DIV AB
    - Multiply A by B and place result in A and B registers.
    - Divide A by B and place quotient in A register & remainder in B register.

4.10.2 : Logical instructions in 8051.

- ✓ ANL : It performs AND logical operation between two operands.
  - ➢ Work on byte sized operands or the CY flag.
    - ANL A, Rn
    - ANL A, direct
    - ANL A, @Ri
    - ANL A, #data
    - ANL direct, A
    - ANL direct, #data
    - ANL C, bit
    - ANL C, /bit
- ✓ ORL: It performs OR logical operation between two operands.
  - ➢ Work on byte sized operands or the CY flag.
    - ORL A, Rn
    - ORL A, direct
    - ORL A, @Ri
    - ORL A, #data
- ✓ XRL
  - ➢ Works on bytes only.
    - XRL A, Rn
    - XRL A, direct
- ✓ CPL / CLR
  - ➢ Complement / Clear.
  - ➢ Work on the accumulator or a bit.
    - CLR P1.2
    - CPL Rn
- ✓ RL / RLC / RR / RRC
  - ➢ Rotate the accumulator.
    - RL and RR without the carry

- RLC and RRC rotate through the carry.
- SWAP A: Swap the upper and lower nibbles of the accumulator.

4.10.3 : Data transfer instructions in 8051.

Briefly explain the data transfer instructions available in 8051 microcontroller. (NOV 2014)

MOV

- ➢ 8-bit data transfer for internal RAM and the SFR.
  - MOV A, Rn
  - MOV A, direct
  - MOV A, @Ri
  - MOV A, #data
  - MOV Rn, A
  - MOV Rn, direct
  - MOV Rn, #data
  - MOV direct, A
  - MOV direct, Rn
  - MOV direct, direct
  - MOV direct, @Ri
  - MOV direct, #data
  - MOV @Ri, A
  - MOV @Ri, direct
  - MOV @Ri, #data

- ✓ MOV
  - ➢ 1-bit data transfer involving the CY flag
    - MOV C, bit
    - MOV bit, C
- ✓ MOV
  - ➢ 16-bit data transfer involving the DPTR
    - MOV DPTR, #data
- ✓ MOVC
  - ➢ Move Code Byte
    - Load the accumulator with a byte from program memory.

    - Must use indexed addressing
    - MOVC A, @A+DPTR
    - MOVC A, @A+PC

- ✓ MOVX
  - ➢ Data transfer between the accumulator and a byte from external data memory.
    - MOVX A, @Ri
    - MOVX A, @DPTR
    - MOVX @Ri, A
    - MOVX @DPTR, A
- ✓ PUSH / POP
  - ➢ Push and Pop a data byte onto the stack.
  - ➢ The data byte is identified by a direct address from the internal RAM locations.
    - PUSH DPL
    - POP 40H
- ✓ XCH
  - ➢ Exchange accumulator and a byte operand
    - XCH A, Rn
    - XCH A, direct
    - XCH A, @Ri
- ✓ XCHD
  - ➢ Exchange lower digit of accumulator with the lower digit of the memory location specified.
    - XCHD A, @Ri
    - The lower 4-bits of the accumulator are exchanged with the lower 4-bits of the internal memory location identified indirectly by the index register.
    - The upper 4-bits of each are not modified.

4.10.4 : Boolean (or) Bit manipulation instructions in 8051.

- ✓ This group of instructions is associated with the single-bit operations of the 8051.
- ✓ This group allows manipulating the individual bits of bit addressable registers and memory locations as well as the CY flag.
    - The P, OV, and AC flags cannot be directly altered.
- ✓ This group includes:

- Set, clear, and, or complement, move.
- Conditional jumps.

✓ CLR

- Clear a bit or the CY flag.
- CLR P1.1
- CLR C

✓ SETB

- Set a bit or the CY flag.
- SETB A.2
- SETB C

✓ CPL

- Complement a bit or the CY flag.
- CPL 40H; Complement bit 40 of the bit addressable memory

✓ ORL / ANL

- OR / AND a bit with the CY flag.
- ORL C, 20H; OR bit 20 of bit addressable memory with the CY flag
- ANL C, 34H; AND bit 34 of bit addressable memory with the CY flag.

✓ MOV

- Data transfer between a bit and the CY flag.
- MOV C, 3FH; Copy the CY flag to bit 3F of the bit addressable memory.
- MOV P1.2, C; Copy the CY flag to bit 2 of P1.

✓ JC / JNC

- Jump to a relative address if CY is set / cleared.

✓ JB / JNB

- Jump to a relative address if a bit is set / cleared.
- JB ACC.2, <label>

✓ JBC - Jump to a relative address, if a bit is set and clear the bit.

4.10.5 : Branching instructions:

With example, explain branching instructions in 8051 microcontroller. (May 2010, NOV 2012)

Explain the working of program control transfer instructions of 8051. (May 2012)

✓ The 8051 provides four different types of unconditional jump instructions:

➢ Short Jump – SJMP

- Uses an 8-bit signed offset relative to the 1st byte of the next instruction.
- Long Jump – LJMP
- Uses a 16-bit address.
- 3 byte instruction capable of referencing any location in the entire 64K of program memory.

➢ Absolute Jump – AJMP
- Uses an 11-bit address.
- 2 byte instruction
- The 11-bit address is substituted for the lower 11-bits of the PC to calculate the 16-bit address of the target.
- The location referenced must be within the 2K Byte memory.

➢ Indirect Jump – JMP
- JMP @A + DPTR

✓ The 8051 provides 2 forms for the CALL instruction:

➢ Absolute Call – ACALL
- Uses an 11-bit address similar to AJMP
- The subroutine must be within the same 2K page.

➢ Long Call – LCALL
- Uses a 16-bit address similar to LJMP
- The subroutine can be anywhere.

➢ Both forms push the 16-bit address of the next instruction on the stack and update the stack pointer.

✓ The 8051 provides 2 forms for the return instruction:

➢ Return from subroutine – RET
- Pop the return address from the stack and continue execution there.

➢ Return from Interrupt Service Routine – RETI
- Pop the return address from the stack.
- Continue execution at the address retrieved from the stack.
- The PSW is not automatically restored.

✓ The 8051 supports 5 different conditional jump instructions.

➢ ALL conditional jump instructions use an 8-bit signed offset.

➢ Jump on Zero – JZ / JNZ

- Jump if the A == 0 / A != 0
    - The check is done at the time of the instruction execution.

➢ Jump on Carry – JC / JNC
- Jump if the C flag is set / cleared.

➢ Jump on Bit – JB / JNB
- Jump if the specified bit is set / cleared.
- Any addressable bit can be specified.

➢ Jump if the Bit is set then Clear the bit – JBC
- Jump if the specified bit is set.
- Then clear the bit.

✓ Compare and Jump if Not Equal – CJNE

➢ Compare the magnitude of the two operands and jump if they are not equal.
- The values are considered to be unsigned.
- The Carry flag is set / cleared appropriately.
- CJNE A, direct, rel
- CJNE Rn, #data, rel
- CJNE @Ri, #data, rel

✓ Decrement and Jump if Not Zero – DJNZ

➢ Decrement the first operand by 1 and jump to the location identified by the second operand if the resulting value is not zero.
- DJNZ Rn, rel
- DJNZ direct, rel

➢ NOP – No operation

****

Program 1: Using timers in 8051 write a program to generate square wave 100ms, 50% duty cycle. (NOV 2014, May 2016, May 2012)

```
                MOV TMOD, #01
                Here: MOV TL0, #D7
                MOV TH0, #B4
                CPL P1.3
                SETB TRO
         Again: JNB TF0, Again
                CLR TR0
                CLR TF0
                SJMP Here
```

Program 2: Write an 8051 ALP to multiply the given number 48H and 30H. (April 2017)

Mnemonics		Comments
Opcode	Operand	
MOV	A,#48	;Store data1 in accumulator
MOV	B,#30	;Store data2 in B register
MUL	AB	;Multiply both
MOV	DPTR,#4500	;Initialize memory location
MOVX	@DPTR,A	;Store lower order result
INC	DPTR	;Go to next memory location
MOV	A,B	;Store higher order result
MOVX	@DPTR,A	
L1: SJMP	L1	;Stop the program

Program 3: Write a program to add two 16 bit numbers. The numbers are 8C8D and 8D8C. Place the sum in R7 and R6. R6 should have the lower byte. (NOV 2010)

Mnemonics		Comments
Opcode	Operand	
MOV	A, #8D	;Store LSB data1 in accumulator
MOV	B, #8C	;Store LSB data2 in B register
ADD	A, B	;Add both
MOV	R6, A	;Store LSB result
MOV	A, #8C	;Store MSB data1 in accumulator
MOV	B, #8D	;Store MSB data2 in B register
ADD	A, B	;Add both
MOV	R7, A	;Store MSB result
L1: SJMP	L1	;Stop the program

UNIT V

INTERFACING MICROCONTROLLER

Programming 8051 Timers - Serial Port Programming - Interrupts Programming – LCD & Keyboard Interfacing - ADC, DAC & Sensor Interfacing - External Memory Interface- Stepper Motor and Waveform generation.

5.1: PROGRAMMING TIMERS OF 8051

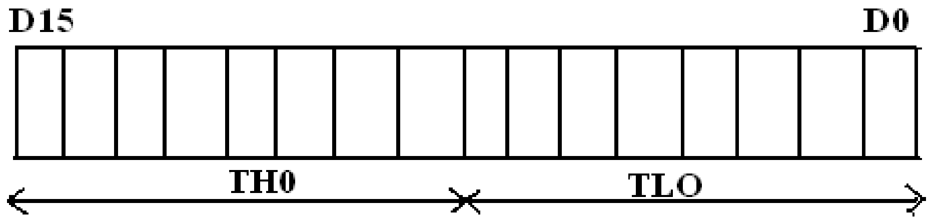
1. **Explain the different modes of operation of timers in 8051 in detail with its associated registers.**
   Describe different modes of operation of timers /counters in 8051 with its associated registers. (NOV 2009, MAY 2009. May 2007, May 2016)
   Draw and explain the functions of TCON and TMOD registers of 8051. (Dec 2008)
   Explain the on-chip timer modes of an 8051 Microcontroller. (April 2010, NOV 2016)

Timer Registers.

- ✓ The 8051 has two timers/counters, they can be used either as timers (used to generate a time delay) or as event counters.

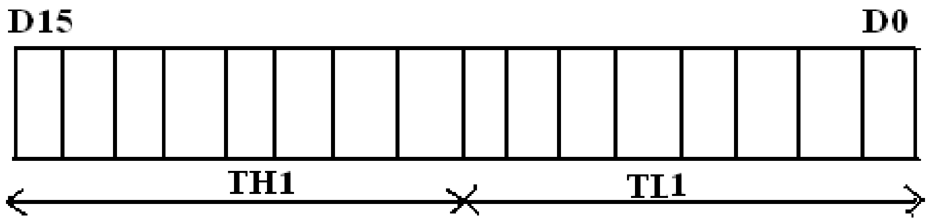
TIMER 0:

- ✓ Timer 0 is a 16-bit register and can be treated as two 8-bit registers (TL0 & TH0).
- ✓ These registers can be accessed similar to any other registers like A, B or R1 etc
- ✓ Ex : The instruction MOV TL0,#07 moves the value 07 into lower byte of Timer0.
- ✓ Similarly MOV R1, TH0 saves the contents of TH0 in the R1 register.



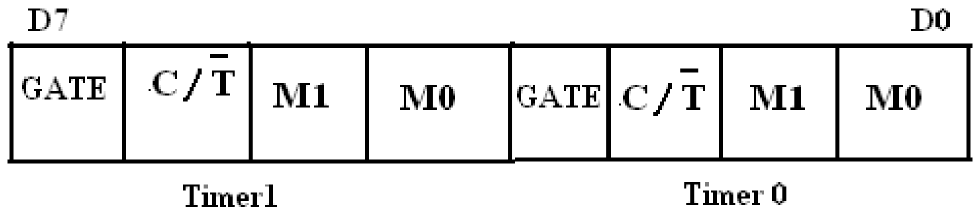
TIMER 1:

- ✓ Timer 1 is also a 16-bit register and can be treated as two 8-bit registers (TL1 & TH1).
- ✓ These registers can be accessed similar to any other registers like A, B or R1etc
- ✓ Ex : The instruction MOV TL1,#05 moves the value 05 into lower byte of Timer1.
- ✓ Similarly MOV R0,TH1 saves the contents of TH1 in the R0 register.



TMOD (Timer mode Register):

- ✓ The various operating modes of both the timers T0 and T1 are set by a TMOD register.
- ✓ TMOD is a 8-bit register.
- ✓ The lower 4 bits are for Timer 0
- ✓ The upper 4 bits are for Timer 1
- ✓ In each case,
  - The lower 2 bits are used to set the timer mode
  - The upper 2 bits to specify the operation



GATE:

- ✓ This bit is used to start or stop the timers by hardware.
- ✓ When GATE= 1, the timers can be started / stopped by the external sources.
- ✓ When GATE= 0, the timers can be started or stopped by software instructions like SETB TR_X or CLR TR_X.

C/T (Counter/Timer):

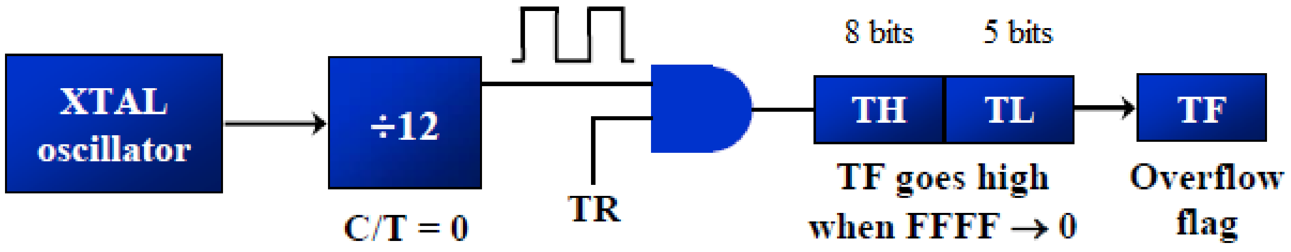
- ✓ This bit decides whether the timer is used as delay generator or event counter.
- ✓ When $C/\overline{T}$ **= 0,** timer is used as delay generator.
- ✓ When $C/\overline{T}$ **=1,** timer is used as an event counter.
- ✓ The clock source for the time delay is the crystal frequency of 8051.
- ✓ The clock source for the event counter is the external clock source.

M1, M0 (Mode):

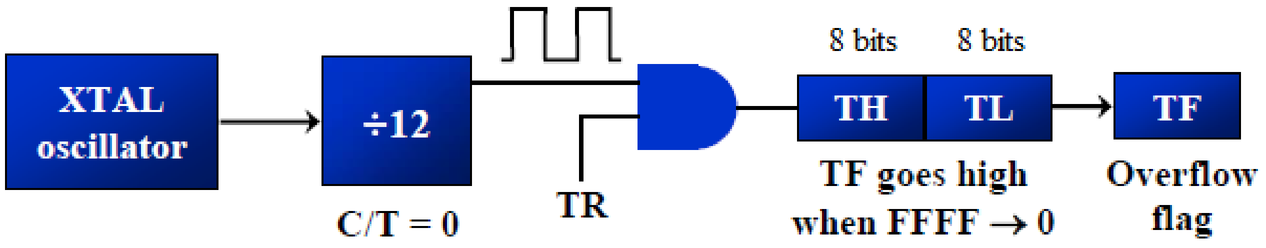
- ✓ These two bits are the timer mode bits.
- ✓ The timers of the 8051 can be configured in four modes Mode0, Mode1, Mode2 & Mode 3.
- ✓ The selection and operation of the modes is shown below.

S.No	M0	M1	Mode	Operation
1	0	0	Mode 0	**13-bit Timer mode.** 8-bit Timer/counter THx with TLx as 5-bit prescaler
2	0	1	Mode 1	**16-bit Timer mode.**16-bit timer /counter THx and TLx are cascaded. There is no presacler
3	1	0	Mode 2	**8-bit auto reload.** 8-bit auto reload timer/counter. THx holds a value which is to be reloaded TLx each time it overflows
4	1	1	Mode 3	**Split timer mode**

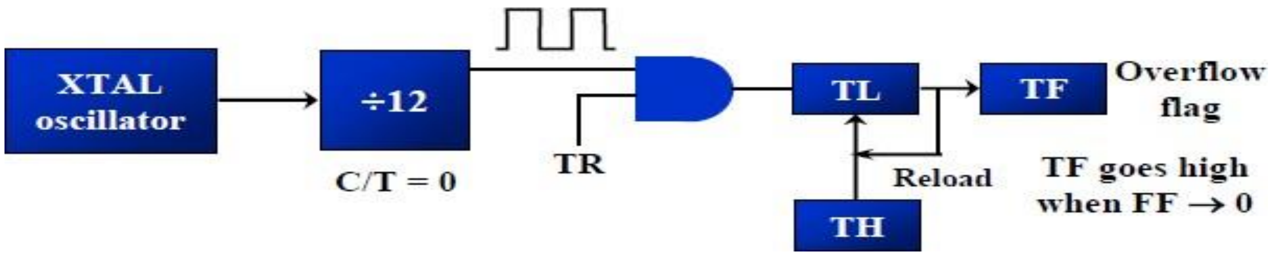
Mode 0: 13 bit Timer mode



Mode 1: 16 bit Timer mode



Mode 2: 8 bit auto reload mode



Mode 3: Split Timer mode

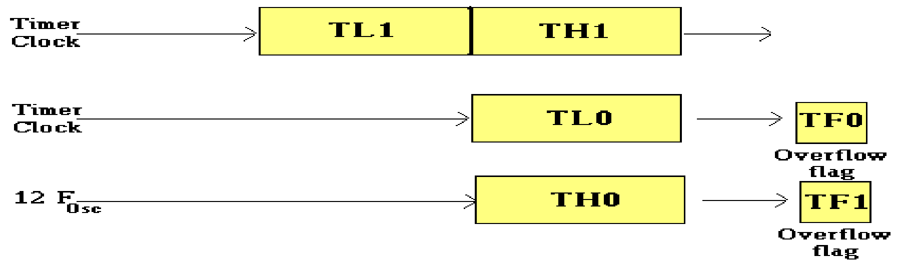


Figure: Modes of operation of Timer

TCON (Timer control register)

- ✓ TCON (timer control) register is an 8-bit register. TCON register is a bit-addressable register.

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

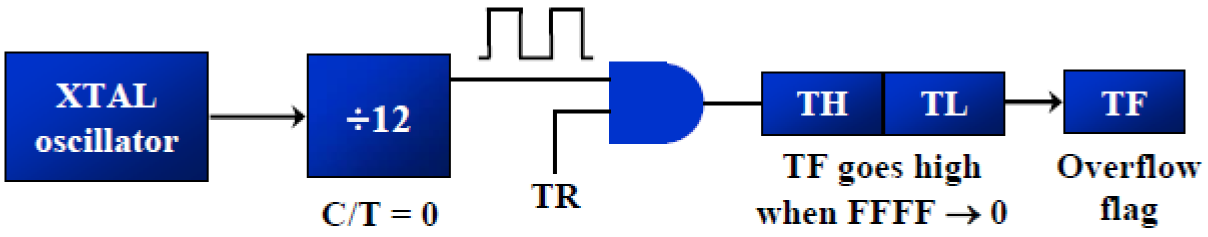
Bit Number	Bit Mnemonic	Description
7	TF1	Timer 1 overflow flag Cleared by hardware when processor vectors to interrupt routine. Set by hardware on timer/counter overflow, when the timer 1 register overflows.
6	TR1	Timer 1 run control bit Clear to turn off time/counter 1. Set to turn on timer/counter 1.
5	TF0	Timer 0 overflow flag Cleared by hardware when processor vectors to interrupt routine. Set by hardware on timer/counter overflow, when the timer 0 register overflows.
4	TR0	Timer 0 run control bit Clear to turn off time/counter 0. Set to turn on timer/counter 0.
3	IE1	External interrupt 1 edge flag. Cleared by hardware when interrupt is processed if edge-triggered. Set by hardware when external interrupt is detected on INT1 pin.
2	IT1	External interrupt 1 type control bit Clear to select low level active (level triggered) for external interrupt 1. Set to select falling edge active (edge triggered) for external interrupt 1.
1	IE0	External interrupt 0 edge flag Cleared by hardware when interrupt is processed if edge-triggered. Set by hardware when external interrupt is detected on INT0 pin.
0	IT0	External interrupt 0 type control bit Clear to select low level active (level triggered) for external interrupt 0. Set to select falling edge active (edge triggered) for external interrupt 0.

Timers of 8051 do starting and stopping by either software or hardware control

- ✓ For using software to start and stop the timer where GATE=0
- ✓ The start and stop of the timer are controlled by software using TR (timer start) bits TR_X and CLR_X
- ✓ The SETB instruction starts it, and it is stopped by the CLR instruction.
- ✓ These instructions start and stop the timers as long as GATE=0 in the TMOD register
- ✓ The hardware way of starting and stopping the timer is achieved by making GATE=1 in the TMOD register.

The following are the characteristics and operations of mode 1:

1. It is a 16-bit timer.
2. It allows value from 0000 to FFFFH.
3. Value to be loaded into the timer register TL and TH.
4. After TH and TL are loaded with a 16-bit initial value, the timer must be started
    - This is done by SETB TR0 for timer 0 and SETB TR1 for timer 1
5. After the timer is started, it starts to count up
    - It counts up until it reaches its limit of FFFFH
    - When it rolls over from FFFFH to 0000, it sets high a flag bit called TF (timer flag)
    - Each timer has its own timer flag.
    - There are TF0 for timer 0, and TF1 for timer 1.



6. Timer flag can be monitored,
    - When this timer flag is raised, to stop the timer with the CLR instructions.
    - CLR TR0 and CLR TR1, for timer 0 and timer 1 respectively.
    - After the timer reaches its limit and rolls over.
    - In order to repeat the process, TH and TL must be reloaded with the original value and TF must be reloaded to 0.

To generate a time delay

1. Load the TMOD register indicating which timer is to be used and which timer mode is selected.
2. Load registers TL and TH with initial count value.
3. Start the timer
4. Keep monitoring the timer flag (TF) with the JNB TFx , target to see if it is raised
   - Get out of the loop when TF becomes high
5. Stop the timer
6. Clear the TF flag for the next round
7. Go back to Step 2 to load TH and TL again.

Example 1:

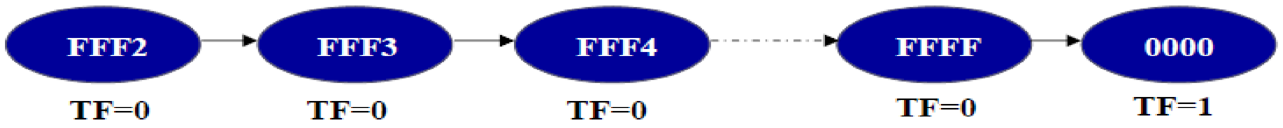
In the following program, we create a square wave of 50% duty cycle (with equal portions high and low) on the P1.5 bit. Timer 0 is used to generate the time delay. Analyze the program. (Nov 2014)

```
            MOV  TMOD,#01     ;Timer 0, mode 1(16-bit mode)
HERE:       MOV  TL0,#0F2H    ;TL0=F2H, the low byte
            MOV  TH0,#0FFH    ;TH0=FFH, the high byte
            CPL P1.5          ;toggle P1.5
            ACALL DELAY
            SJMP HERE
DELAY:      SETB TR0          ;start the timer 0
AGAIN:      JNB TF0,AGAIN     ;monitor timer flag 0 until it rolls over
            CLR TR0           ;stop timer 0
            CLR TF0           ;clear timer 0 flag
            RET
```

In the above program notice the following steps.

1. TMOD is loaded.
2. FFF2H is loaded into TH0-TL0.
3. P1.5 is toggled for the high and low portions of the pulse.
4. The DELAY subroutine using the timer is called.
5. In the DELAY subroutine, timer 0 is started by the SETB TR0 instruction.
6. Timer 0 counts up with the passing of each clock, which is provided by the crystal oscillator.
   - As the timer counts up, it goes through the states of FFF3, FFF4, FFF5, FFF6, and so on until it reaches FFFFH.
   - One more clock rolls it to 0, raising the timer flag (TF0=1). At that point, the JNB instruction falls through.
7. Timer 0 is stopped by the instruction CLR TR0.
   - The DELAY subroutine ends and the process is repeated.

Notice that to repeat the process, we must reload the TL and TH registers, and start the process is repeated.



Example 2:

In Example 1, calculate the amount of time delay in the DELAY subroutine generated by the timer. Assume XTAL = 11.0592 MHz.

Solution:

- ✓ The timer works with a clock frequency of 1/12 of the XTAL frequency, we have 11.0592 MHz / 12 = 921.6 kHz as the timer frequency.
- ✓ As a result, each clock has a period of T =1/921.6kHz,T=1.085μs.
- ✓ In other words, Timer 0 counts up each 1.085μs resulting in delay = number of counts × 1.085μs.
- ✓ The number of counts for the roll over is FFFFH – FFF2H = 0DH (13 decimal).
- ✓ Add one to 13 because of the extra clock needed when it rolls over from FFFF to 0 and raise the TF flag.
- ✓ This gives 14 × 1.085μs = 15.19μs for half the pulse. For the entire period it is T = 2 × 15.19μs = 30.38μs as the time delay generated by the timer.

(a) In hexadecimal

(FFFF – YYXX + 1) ×1.085 μs, where YYXX are TH, TL initial values respectively. Notice that value YYXX are in hex.

(b) In decimal

Convert YYXX values of the TH, TL register to decimal to get a NNNN decimal, then (65536 - NNNN) × 1.085 μs

Example 3:

In Example 1, calculate the frequency of the square wave generated on pin P1.5.

Solution:

- ✓ In the timer delay calculation of Example 1, we did not include the overhead due to instruction in the loop.
- ✓ To get a more accurate timing, we need to add clock cycles due to these instructions in the loop.
- ✓ To do that, we use the machine cycle as shown below.

```
                                        Cycles
HERE:       MOV TL0,#0F2H               2
            MOV TH0,#0FFH               2
            CPL P1.5                    1
            ACALL DELAY                 2
            SJMP HERE                   2
DELAY:      SETB TR0                    1
AGAIN:      JNB TF0, AGAIN              14
            CLR TR0                     1
            CLR TF0                     1
            RET                         2
                              Total     28
```

T = 2 × 28 × 1.085 us = 60.76 µs and F = 16458.2 Hz

Example 4:

Find the delay generated by timer 0 in the following code, using both of the Methods. Do not include the overhead due to instruction.

```
            CLR P2.3            ;Clear P2.3
            MOV TMOD,#01        ;Timer 0, 16-bitmode
HERE:       MOV TL0,#3EH        ;TL0=3Eh, the low byte
            MOV TH0,#0B8H       ;TH0=B8H, the high byte
            SETB P2.3           ;SET high timer 0
            SETB TR0            ;Start the timer 0
AGAIN:      JNB TF0,AGAIN       ;Monitor timer flag 0
            CLR TR0             ;Stop the timer 0
            CLR TF0             ;Clear TF0 for next round
            CLR P2.3
```

Solution:

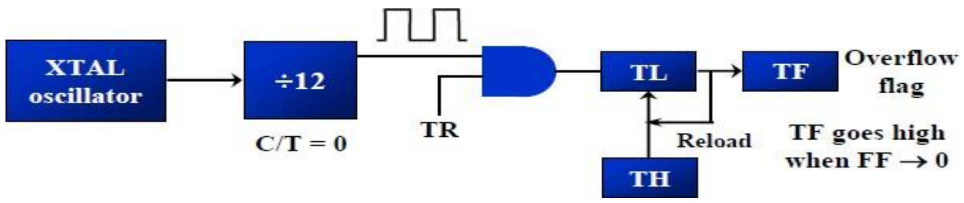
(FFFFH – B83E + 1) = 47C2H = 18370 in decimal and 18370 × 1.085 µs = 19.93145 ms

<u>The following are the characteristics and operations of mode 2:</u>

1. It is an 8-bit timer. It allows only values of 00 to FFH to be loaded into the timer register TH.
2. After TH is loaded with the 8-bit value, the 8051 copies value to TL register.
   - Then the timer must be started.
   - This is done by the instruction SETB TR0 for timer 0 and SETB TR1 for timer 1.
3. After the timer is started, it starts to count up by incrementing the TL register.
   - It counts up until it reaches its limit of FFH
   - When it rolls over from FFH to 00, it sets high the TF (timer flag)
   - When the TL register rolls from FFH to 00 and TF is set to 1.

- TL is reloaded automatically with the original value kept by the TH register.
- To repeat the process, simply clear TF.

4. This makes mode 2 an auto-reload, in contrast with mode 1 in which the programmer has to reload TH and TL



To generate a time delay

1. Load the TMOD value register indicating which timer is to be used, and the timer mode (mode 2) is selected.
2. Load the TH register with the initial count value.
3. Start timer.
4. Keep monitoring the timer flag (TF) with the JNB TFx, target, to see whether it is raised
   Get out of the loop when TF goes high
5. Clear the TF flag.
6. Go back to Step4, since mode 2 is auto reload.

Example 5:

Assume XTAL = 11.0592 MHz, find the frequency of the square wave generated on pin P1.0.

```
        MOV TMOD,#20H   ;T1/8-bit/auto reload
        MOV TH1,#5      ;TH1 = 5
        SETB TR1        ;start the timer 1
BACK:   JNB TF1,BACK    ;till timer rolls over
        CPL P1.0        ;P1.0 to hi, lo
        CLR TF1         ;clear Timer 1 flag
        SJMP BACK       ;mode 2 is auto-reload
```

Solution:

- ✓ In mode 2, no need to reload TH since it is auto-reload.
- ✓ Now (256 - 05) × 1.085 µs =251 × 1.085 µs = 272.33 µs is the high portion of the pulse.
- ✓ Since it is a 50% duty cycle square wave, the period T is twice.
- ✓ As a result T = 2 × 272.33 µs = 544.67 µs and the frequency = 1.83597 kHz

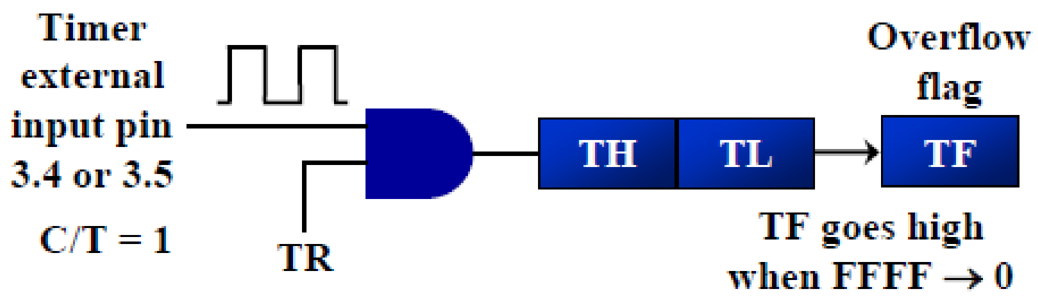
5.2: Timers as counters

Timers can also be used as counters.

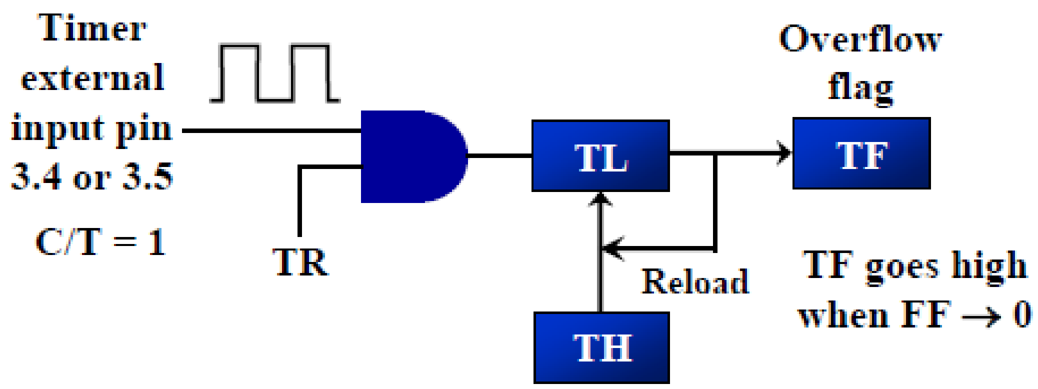
Which are used for counting events happening outside the 8051.

- When it is used as a counter, it is a pulse outside of the 8051 that increments the TH, TL register.
- TMOD and TH, TL registers are the same as in timer concept, except the source of the frequency.
- The C/T bit in the TMOD register decides the source of the clock for the timer
- When C/T = 1, the timer is used as a counter and gets its pulses from outside the 8051.
- The counter counts up as pulses are fed from pins 14 and 15.
- these pins are called T0 (timer 0 input) and T1 (timer 1 input)

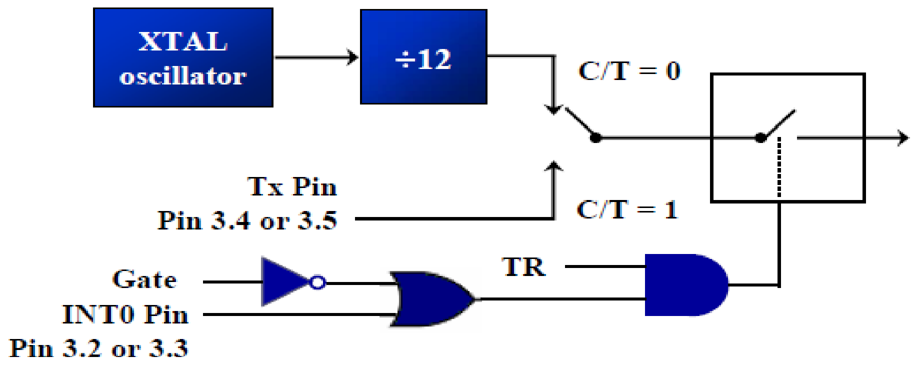
Timer with external input (Mode 1)


Timer external input pin 3.4 or 3.5
C/T = 1
TR
TH
TL
Overflow flag
TF
TF goes high when FFFF → 0

Timer with external input (Mode 2)



- ✓ If GATE = 1, the start and stop of the timer are done externally through pins P3.2 and P3.3 for timers 0 and 1, respectively
- ✓ This hardware allows starting or stopping the timer externally at any time via a simple switch



✓ The frequency for the timer is always 1/12th the frequency of the crystal attached to the 8051.

Port 3 pins used for Timers 0 and 1

Pin	Port Pin	Function	Description
14	P3.4	T0	Timer/counter 0 external input
15	P3.5	T1	Timer/counter 1 external input

Example 6:

Assuming that clock pulses are fed into pin T1, write a program for counter 1 in mode 2 to count the pulses and display the state of the TL1 count on P2, which connects to 8 LEDs.

Solution:

```
            MOV TM0D,#01100000B    ;counter 1, mode 2, C/T=1 external pulses
            MOV TH1,#0             ;clear TH1
            SETB P3.5               ;make T1 input
AGAIN:       SETB TR1               ;start the counter
BACK:       MOV A,TL1              ;get copy of TL
            MOV P2,A                ;display it on port 2
            JNB TF1,Back           ;keep doing, if TF = 0
            CLR TR1                ;stop the counter 1
            CLR TF1                ;make TF=0
            SJMP AGAIN             ;keep doing it
```

✓ Notice in the above program the role of the instruction SETB P3.5.

✓ Since ports are set up for output when the 8051 is powered up.

✓ So, we make P3.5 an input port by making it high.

✓ In other words, we must configure (set high) the T1 pin (pin P3.5) to allow pulses to be fed into it.

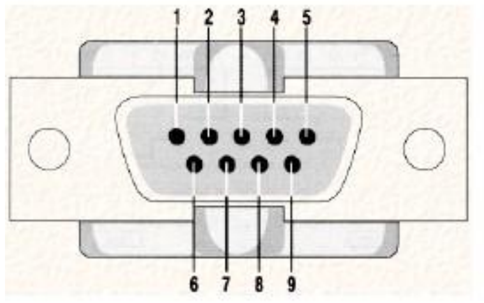
5.3: SERIAL COMMUNICATION

2. Explain the serial programming of 8051 with its associated registers. (May 2014, 2013)(Or)
Explain how to program for sending and receiving data serially using 8051 (April 2010, 2011)
Explain 8051 serial port programming with examples. (May 2016, NOV 2012)
Explain the serial modes of operation of 8051 microcontroller. (May 2007)

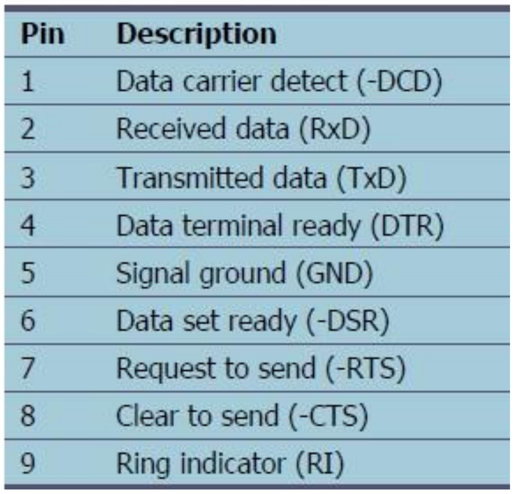
RS232

- ✓ It is an interfacing standard RS232.
- ✓ It was set by the Electronics Industries Association (EIA) in 1960.
- ✓ The standard was set long before the advent of the TTL logic family.
- ✓ Its input and output voltage levels are not TTL compatible.
- ✓ In RS232, a 0 is represented by -3 to -25 V, while a 1 bit is +3 to +25 V.
- ✓ IBM introduced the DB-9 version of the serial I/O standard.

RS232 Connector DB-9



RS232 DB-9 Pins



Pin	Description
1	Data carrier detect (-DCD)
2	Received data (RxD)
3	Transmitted data (TxD)
4	Data terminal ready (DTR)
5	Signal ground (GND)
6	Data set ready (-DSR)
7	Request to send (-RTS)
8	Clear to send (-CTS)
9	Ring indicator (RI)

<u>Handshake signals of MODEM</u>

DTR (data terminal ready)

- When DTR =1, indicate that it is ready for communication.

DSR (data set ready)

- When DSR =1, indicate that it is ready for communication.

RTS (request to send)

- It asserts RTS to signal the modem that it has a byte of data to transmit.

CTS (clear to send)

- It is to receive, it sends out signal CTS,

DCD (data carrier detect)

- The modem asserts signal DCD to inform the DTE that a valid carrier has been detected.

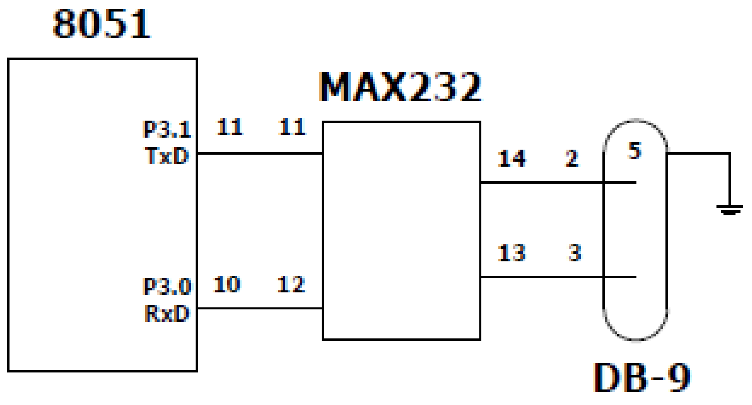
RI (ring indicator)

- An output from the modem and an input to a PC indicates that the telephone is ringing.

MAX232

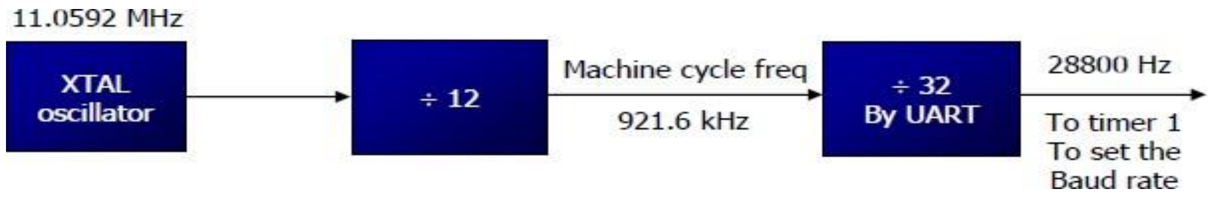
A line driver (MAX232) is required to convert RS232 voltage levels to TTL levels, and vice versa.

- 8051 has two pins that are used specifically for transferring and receiving data serially.
- These two pins are called TxD and RxD and are part of the port 3 (P3.0 and P3.1).
- These pins are TTL compatible.
- They require a line driver to make them RS232 compatible.



Baud rate:

- The baud rates in 8051 are programmable.
- 8051 divides the crystal frequency by 12 to get machine cycle frequency.
- 8051 UART circuitry divides the machine cycle frequency by 32.



- Timer 1 is used to set baud rate using TH1 register

Baud rate	TH1 (decimal)	TH1(Hex)
9600	-3	FD
4800	-6	FA
2400	-12	F4
1200	-24	E8

Explain in detail the serial communication registers of the 8051. (NOV 2009)

SBUF:

- It is an 8-bit register used for serial communication.
- For a byte data to be transferred via the TxD line:
- Byte must be placed in the SBUF register.
- Bytes are framed with the start and stop bits and transferred serially via the TxD line.
- SBUF holds the byte of data when it is received by 8051 RxD line.
- When the bits are received serially via RxD.
- 8051 de-frames byte by eliminating the stop and start bits.

SCON:

- It is an 8-bit register used to program the start bit, stop bit and data bits of data framing.

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Bit Number	Bit Mnemonic	Description
SCON.7	SM0	Serial port mode specifier
SCON.6	SM1	Serial port mode specifier
SCON.5	SM2	Used for multiprocessor communication
SCON.4	REN	Set/Cleared by software to enable/disable reception
SCON.3	TB8	Not widely used
SCON.2	RB8	Not widely used
SCON.1	TI	Transmit interrupt flag. Set by hardware at the begin of the stop bit mode 1. And cleared by software
SCON.0	RI	Receive interrupt flag. Set by hardware at the begin of the stop bit mode 1. And cleared by software

SM0, SM1: Serial port mode specifiers

SM0	**SM1**	
0	0	Serial Mode 0
0	1	Serial Mode 1; 8-bit data, 1 stop bit, 1 start bit
1	0	Serial Mode 2
1	1	Serial Mode 3

In programming the 8051 to transfer character bytes serially

1. TMOD register is loaded with the value 20H, indicating the use of timer 1 in mode 2 (8-bit auto-reload) to set baud rate.
2. The TH1 is loaded with one of the values to set baud rate for serial data transfer.
3. The SCON register is loaded with the value 50H, indicating serial mode 1, where an 8-bit data is framed with start and stop bits.
4. TR1 is set to 1 to start timer 1
5. TI is cleared by CLR TI instruction.
6. The character byte to be transferred serially is written into SBUF register.
7. The TI flag bit is monitored with the use of instruction JNB TI, xx, to see if the character has been transferred completely.
8. To transfer the next byte, go to step 5.

Write a program for the 8051 to transfer letter "A" serially at 4800 baud, continuously.

Solution:

```
            MOV  TMOD, #20H ;timer 1, mode 2 (auto reload)
            MOV   TH1, #-6        ;4800 baud rate
            MOV SCON, #50H ;8-bit, 1 stop, REN enabled
            SETB  TR1             ;start timer 1
AGAIN:      MOV   SBUF, #"A"    ;letter "A" to trtansfer
HERE:       JNB     TI, HERE        ;wait for the last bit
            CLR     TI              ;clear TI for next char
            SJMP AGAIN            ;keep sending A
```

The steps that 8051 goes through in transmitting a character via TxD

1. The byte character to be transmitted is written into the SBUF register
2. The start bit is transferred
3. The 8-bit character is transferred on bit at a time
4. The stop bit is transferred
   - It is during the transfer of the stop bit that 8051 raises the TI flag, indicating that the last character was transmitted
5. By monitoring the TI flag, we make sure that we are not overloading the SBUF

- If we write another byte into the SBUF before TI is raised, the un-transmitted portion of the previous byte will be lost.

6. After SBUF is loaded with a new byte, the TI flag bit must be forced to 0 by CLR TI in order for this new byte to be transferred

By checking the TI flag bit, we know whether or not the 8051 is ready to transfer another byte

- It must be noted that TI flag bit is raised by 8051 itself when it finishes data transfer
- It must be cleared by the programmer with instruction CLR TI
- If we write a byte into SBUF before the TI flag bit is raised, we risk the loss of a portion of the byte being transferred
- The TI bit can be checked by the instruction JNB TI,xx Using an interrupt.

Write a program for the 8051 to transfer "YES" serially at 9600 baud, 8-bit data, 1 stop bit do this continuously. (May 2006)

Solution:

```
            MOV  TMOD, #20H ;timer 1, mode 2 (auto reload)
            MOV   TH1, #-3       ;9600 baud rate
            MOV SCON, #50H ;8-bit, 1 stop, REN enabled
            SETB  TR1            ;start timer 1
AGAIN:      MOV   A, # "Y"       ;transfer "Y"
            ACALL TRANS
            MOV   A, # "E"       ;transfer "E"
            ACALL TRANS
            MOV   A, # "S"       ;transfer "S"
            ACALL TRANS
            SJMP  AGAIN          ;keep doing it
;serial data transfer subroutine
TRANS:      MOV   SBUF, A        ;load SBUF
HERE:       JNB    TI, HERE      ;wait for the last bit
            CLR    TI            ;get ready for next byte
            RET
```

In programming the 8051 to receive character bytes serially

1. TMOD register is loaded with the value 20H, indicating the use of timer 1 in mode (8-bit auto-reload) to set baud rate
2. TH1 is loaded to set baud rate
3. The SCON register is loaded with the value 50H, indicating serial mode 1, where an 8-bit data is framed with start and stop bits
4. TR1 is set to 1 to start timer 1
5. RI is cleared by CLR RI instruction
6. The RI flag bit is monitored with the use of instruction JNB RI, xx to see if an entire character has been received yet
7. When RI is raised, SBUF has the byte, its contents are moved into a safe place.
8. To receive the next character, go to step 5.

Write a program for the 8051 to receive bytes of data serially and put them in P1, set the baud rate at 4800, 8-bit data and 1 stop bit. (NOV 2016)

Solution:

```
              MOV  TMOD, #20H ;timer 1, mode 2 (auto reload)
              MOV   TH1, #-6       ;4800 baud rate
              MOV SCON, #50H ;8-bit, 1 stop, REN enabled
              SETB  TR1            ;start timer 1
HERE:         JNB    RI, HERE      ;wait for char to come in
              MOV   A, SBUF        ;saving incoming byte in A
              MOV   P1, A          ;send to port 1
              CLR   RI             ;get ready to receive next byte
              SJMP  HERE           ;keep getting data
```

In receiving bit via its RxD pin, 8051 goes through the following steps.

1. It receives the start bit
   - Indicating that the next bit is the first bit of the character byte it is about to receive
2. The 8-bit character is received one bit at time
3. The stop bit is received
   - When receiving the stop bit 8051 makes RI = 1,indicating that an entire character byte has been received.

5. After the SBUF contents are copied into a safe place.

- The RI flag bit must be forced to 0 by CLR RI in order to allow the next received character byte to be placed in SBUF.
- Failure to do this causes loss of the received character.

There are two ways to increase the baud rate of data transfer

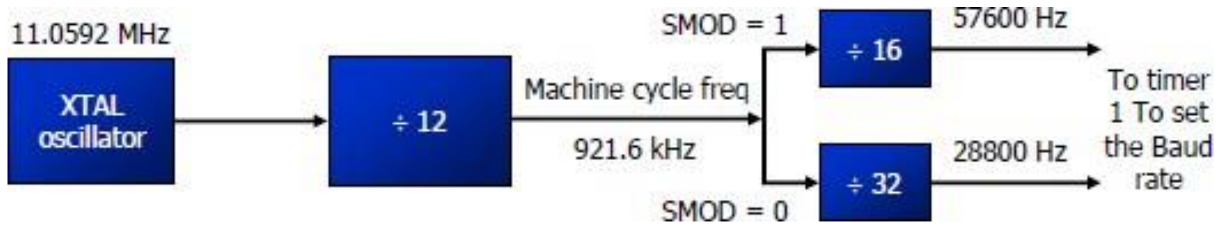
- To use a higher frequency crystal
- To change a bit in the PCON register

PCON

- PCON register is an 8-bit register
- When 8051 is powered up, SMOD is zero.
- We can set it to high by software and thereby double the baud rate.
- GF1, GF0: General flag bits
- PD: Power down mode
- IDL: Ideal mode

SMOD	--	--	--	GF1	GF0	PD	IDL

```
MOV   A,PCON       ;place a copy of PCON in ACC
SETB  ACC.7        ;make D7=1
MOV   PCON,A       ;changing any other bits
```



****

5.4 : LCD Interfacing

3. Explain how LCD is used to interface with 8051. (May 2007)

How does one interface a 16 × 2 LCD Display using 8051 Microcontroller? (May2009, May 2010)

Introduction:

- Liquid Crystal displays are created by sandwiching a thin 10-12μm layer of a liquid-crystal fluid between two glass plates.
- A transparent, electrically conductive film or backplane is put on the rear glass sheet.
- Transparent sections of conductive film in the shape of the desired characters are coated on the front glass plate.
- When a voltage is applied between a segment and the backplane, an electric field is created in the region under the segment.
- This electric field changes the transmission of light through the region under the segment film.
- Most LCD's require a voltage of 2 or 3 V between the backplane and a segment to turn on the segment.

There are two types available of LCD

- **Dynamic scattering and field effect.**

Dynamic scattering types of LCD:

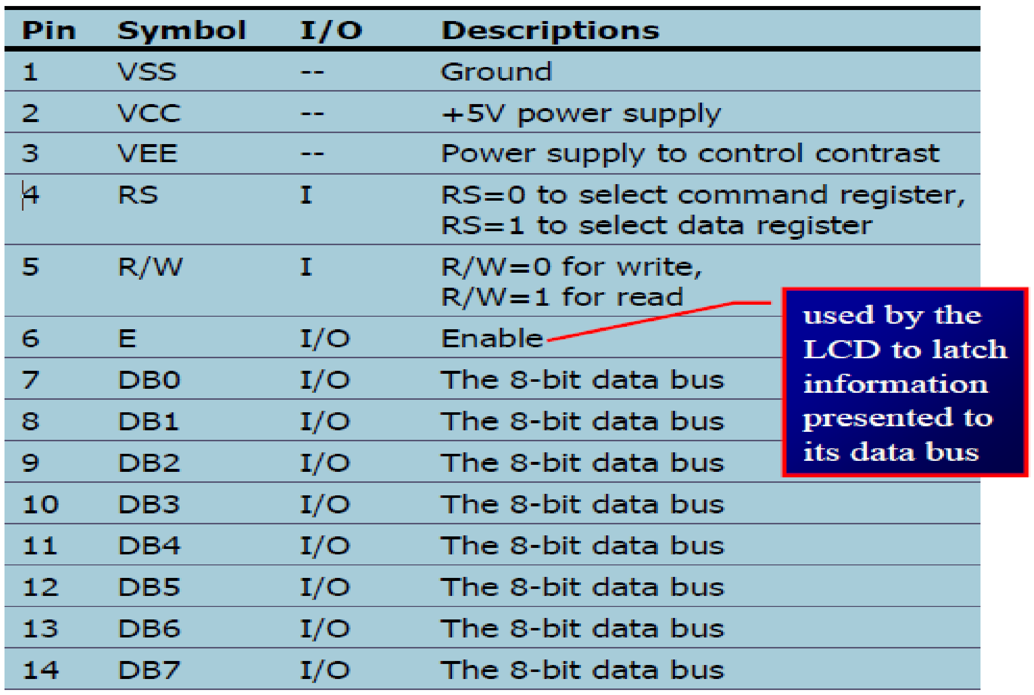
- It scrambles the molecules where the field is present.
- This produces an etched-glass-looking light character on a dark background.

Field-effect types:

- Use polarization to absorb light where the electric field is present.
- This produces dark characters on a silver- gray background.

Advantages of LCD

- LCD is finding widespread use replacing LEDs
- The declining prices of LCD
- The ability to display numbers, characters, and graphics
- Ease of programming for characters and Graphics



Pin	Symbol	I/O	Descriptions
1	VSS	--	Ground
2	VCC	--	+5V power supply
3	VEE	--	Power supply to control contrast
4	RS	I	RS=0 to select command register, RS=1 to select data register
5	R/W	I	R/W=0 for write, R/W=1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8-bit data bus
8	DB1	I/O	The 8-bit data bus
9	DB2	I/O	The 8-bit data bus
10	DB3	I/O	The 8-bit data bus
11	DB4	I/O	The 8-bit data bus
12	DB5	I/O	The 8-bit data bus
13	DB6	I/O	The 8-bit data bus
14	DB7	I/O	The 8-bit data bus

used by the LCD to latch information presented to its data bus

Fig:Pin details of LCD module

Interfacing LCD module

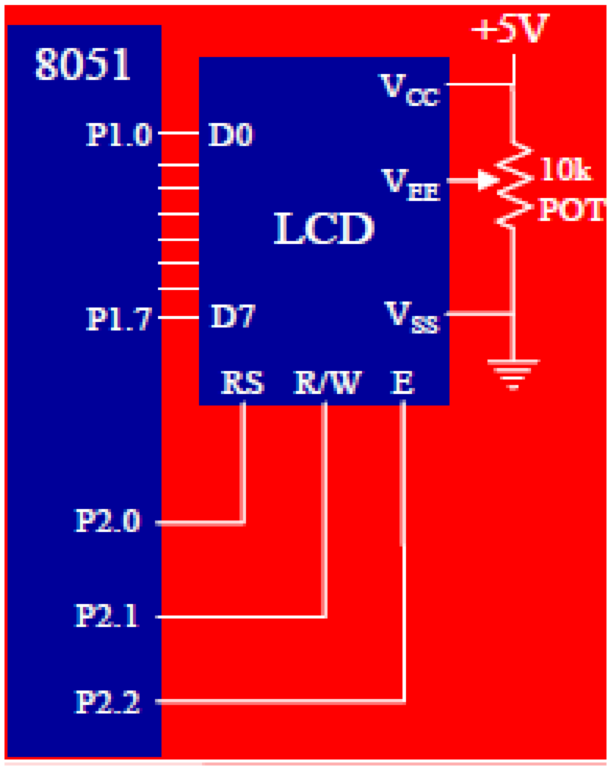
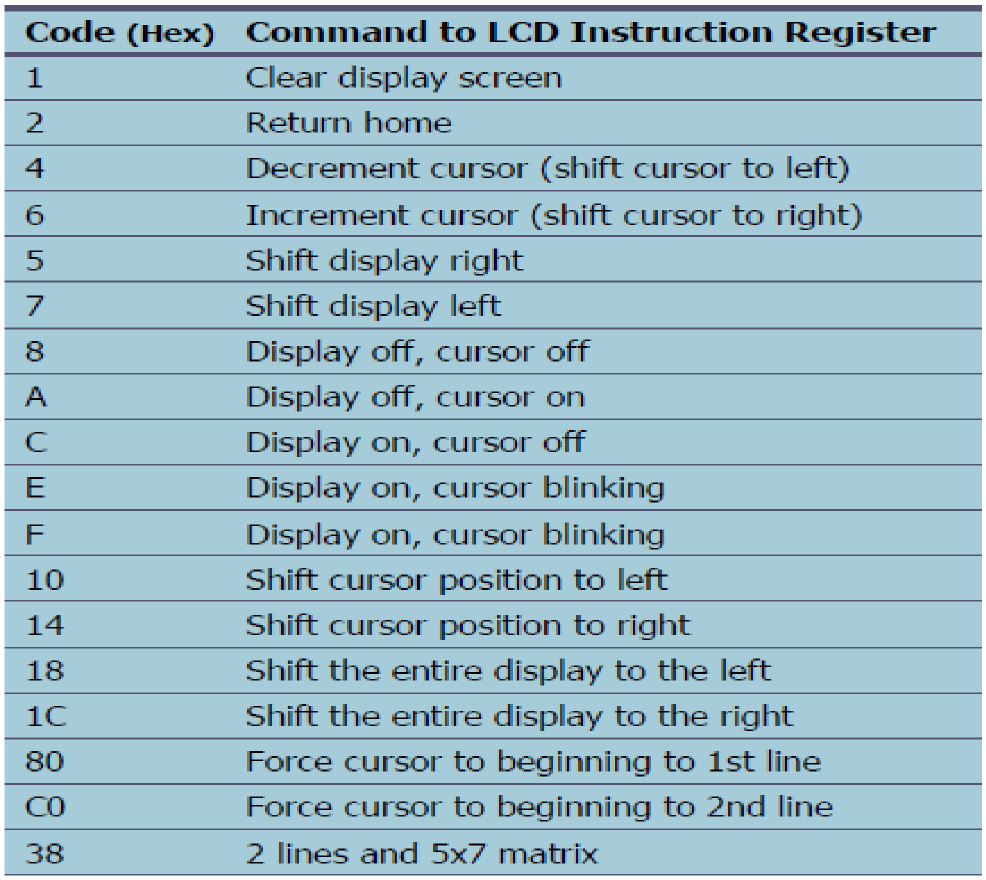


Fig: Interfacing LCD module with controller

LCD Command Codes



Code (Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning to 1st line
C0	Force cursor to beginning to 2nd line
38	2 lines and 5x7 matrix

- To send any of the commands to the LCD, make pin RS=0. For data, make RS=1.
- Then send a high-to-low pulse to the E pin to enable the internal latch of the LCD.
- This is shown in the code below.

;calls a time delay before sending next data/command

```
;P1.0-P1.7 are connected to LCD data pins D0-D7
;P2.0 is connected to RS pin of LCD
;P2.1 is connected to R/W pin of LCD
;P2.2 is connected to E pin of LCD
ORG 0H
MOV A,#38H ;INIT. LCD 2 LINES, 5X7 MATRIX
ACALL COMNWRT ;call command subroutine
ACALL DELAY ;give LCD some time
MOV A,#0EH ;display on, cursor on
ACALL COMNWRT ;call command subroutine
ACALL DELAY ;give LCD some time
MOV A,#01 ;clear LCD
```

```
ACALL COMNWRT ;call command subroutine
ACALL DELAY ;give LCD some time
MOV A,#06H ;shift cursor right
ACALL COMNWRT ;call command subroutine
ACALL DELAY ;give LCD some time
MOV A,#84H ;cursor at line 1, pos. 4
ACALL COMNWRT ;call command subroutine
ACALL DELAY ;give LCD some time
MOV A,#'N' ;display letter N
ACALL DATAWRT ;call display subroutine
ACALL DELAY ;give LCD some time
MOV A,#'O' ;display letter O
ACALL DATAWRT ;call display subroutine
AGAIN: SJMP AGAIN ;stay here
COMNWRT: ;send command to LCD
MOV P1,A ;copy reg A to port 1
CLR P2.0 ;RS=0 for command
CLR P2.1 ;R/W=0 for write
SETB P2.2 ;E=1 for high pulse
ACALL DELAY ;give LCD some time
CLR P2.2 ;E=0 for H-to-L pulse
RET
DATAWRT: ;write data to LCD
MOV P1,A ;copy reg A to port 1
SETB P2.0 ;RS=1 for data
CLR P2.1 ;R/W=0 for write
SETB P2.2 ;E=1 for high pulse
ACALL DELAY ;give LCD some time
CLR P2.2 ;E=0 for H-to-L pulse
RET
DELAY: MOV R3,#50 ;50 or higher for fast CPUs
HERE2: MOV R4,#255 ;R4 = 255
HERE: DJNZ R4,HERE ;stay until R4 becomes 0
DJNZ R3,HERE2
RET
END
```

```
;Check busy flag before sending data, command to LCD
;p1=data pin
;P2.0 connected to RS pin
;P2.1 connected to R/W pin
;P2.2 connected to E pin

ORG 0H
MOV A,#38H ;init. LCD 2 lines ,5x7 matrix
ACALL COMMAND ;issue command
MOV A,#0EH ;LCD on, cursor on
ACALL COMMAND ;issue command
MOV A,#01H ;clear LCD command
ACALL COMMAND ;issue command
MOV A,#06H ;shift cursor right
ACALL COMMAND ;issue command
MOV A,#86H ;cursor: line 1, pos. 6
ACALL COMMAND ;command subroutine
MOV A,#'N' ;display letter N
ACALL DATA_DISPLAY
MOV A,#'O' ;display letter O
ACALL DATA_DISPLAY
HERE:SJMP HERE ;STAY HERE
************************************************************************************
```

5.5: KEYBOARD INTERFACING

4. With neat circuit diagram explain how a 4 x 4 keypad is interfaced with 8051 microcontroller and write 8051 ALP for keypad scanning. (May 2013, Nov 2015, NOV 2007)

- Keyboards are organized in a matrix of rows and columns
- The CPU accesses both rows and columns through ports.
- Therefore, with two 8-bit ports, an 8 x 8 matrix of keys can be connected to a microprocessor.
- When a key is pressed, a row and a column make a contact.
- Otherwise, there is no connection between rows and columns.
- A 4x4 matrix is connected to two ports.
- The rows are connected to an output port and the columns are connected to an input port

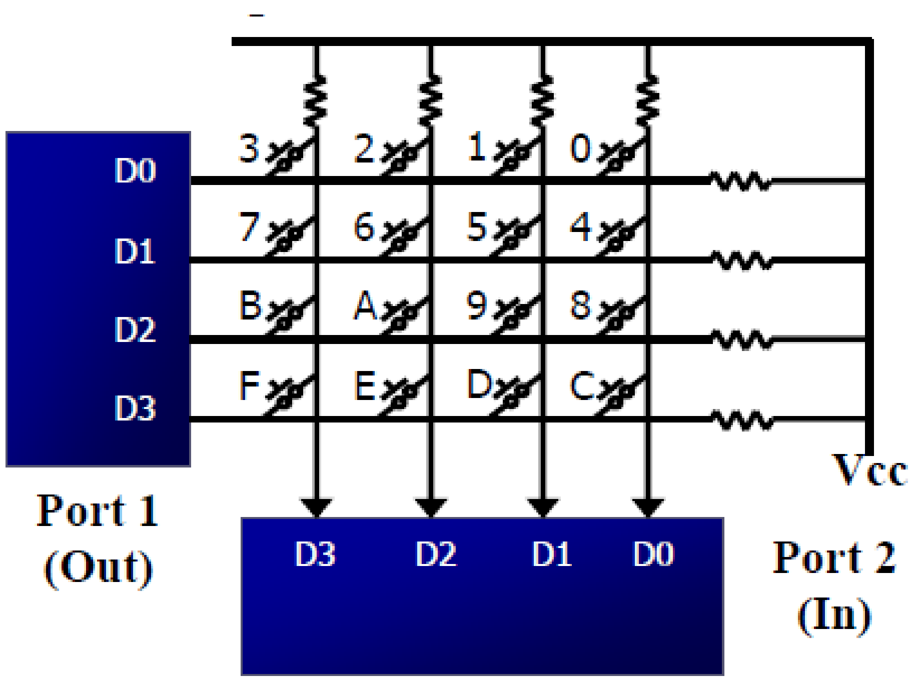
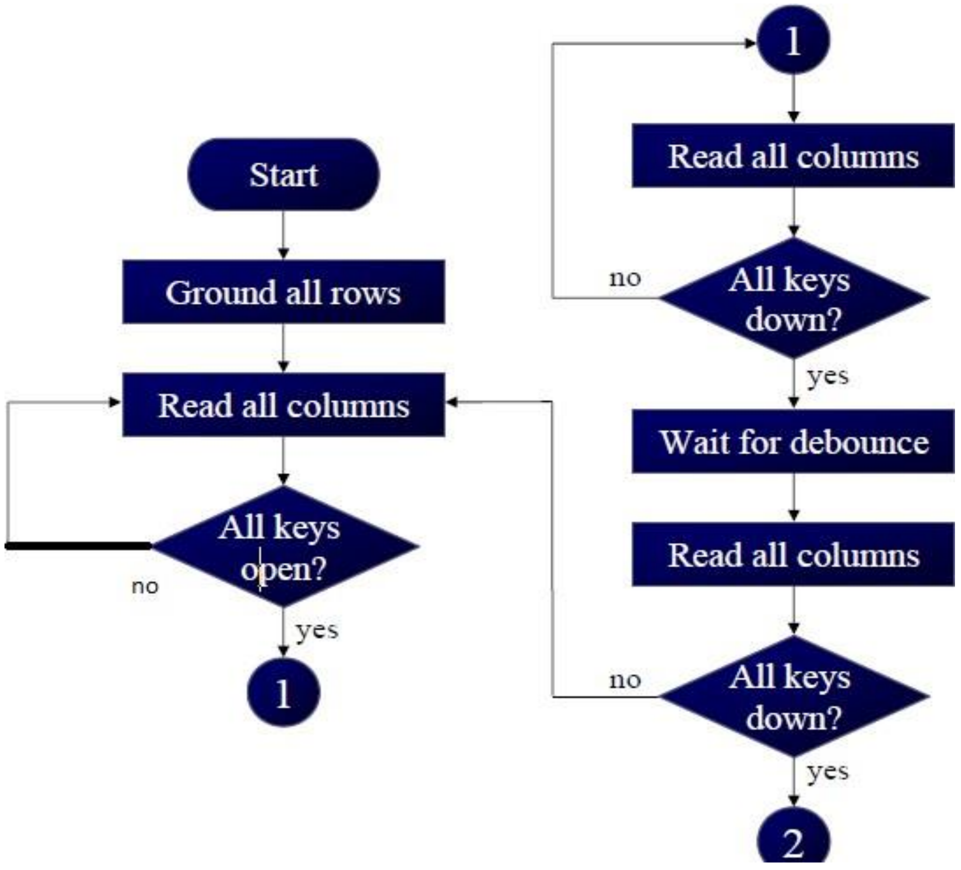


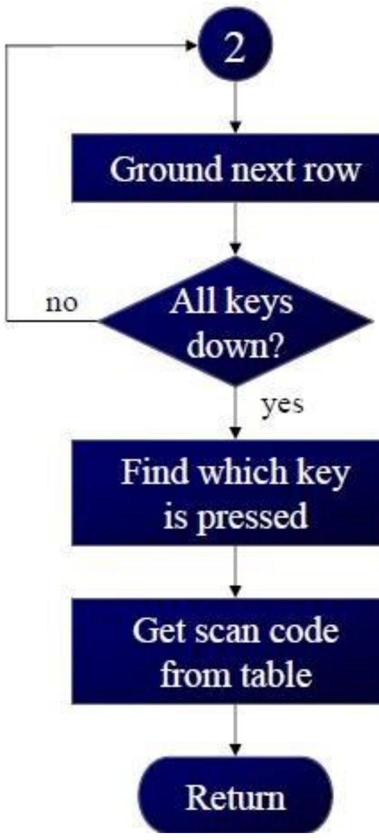
Fig: Model of 4x4 matrix KEYBOARD

- It is the function of the microcontroller to scan the keyboard continuously to detect and identify the key pressed

  <u>To detect a pressed key:</u>

- The microcontroller grounds all rows by providing 0, then it reads the columns
- Data read from columns is D3 – D0 = 1111. No key has been pressed and the process continues till key press is detected
- If one of the column bits has a zero, a key press has occurred.
- For example, if D3 – D0 = 1101, a key in the D1 column has been pressed.
- After detecting a key press, microcontroller will go through the process of identifying the key.
- Starting with the top row, the microcontroller grounds it by providing a low to row D0 only
- It reads the columns, if the data read is all 1s, no key in that row is activated.
- The process is moved to the next row
- It grounds the next row, reads the columns, and checks for any zero
- This process continues until the row is identified.
- After identification of the row in which the key has been pressed
- Find out which column the pressed key belongs to corresponding key is displayed.





Flowchart: Scan the keyboard to detect and identify the key.

5.6 : Interfacing 8051 to ADC

5. Explain how to interface an 8-bit ADC with 8051 Microcontroller.(April 2010, May 2008, Nov 2014)

ADCs (analog-to-digital converters) are widely used devices for data acquisition. A physical quantity (temperature, pressure, humidity, and velocity, etc.,) is converted to electrical (voltage, current) signals using a device called a transducer, or sensor.

- We need an analog-to-digital converter to translate the analog signals to digital numbers, so microcontroller can read them.

ADC804 IC is an analog-to-digital converter

- It works with +5 volts and has a resolution of 8 bits.
- Conversion time is defined as the time it takes the ADC to convert the analog input to a digital(binary) number.
- In ADC804 conversion time varies depending on the clocking signals applied to CLK R and CLK IN pins, but it cannot be faster than 110 μs.

CLK IN and CLK R

- CLK IN is an input pin connected to an external clock source
- To use the internal clock generator (also called self-clocking), CLK IN and CLK R pins are connected to a capacitor and a resistor, and the clock frequency is determined by

  f = 1/1.1 RC

- Typical values are R = 10K ohms and C = 150 Pf. We get f = 606 kHz and the conversion time is 110 μs

D0-D7

- The digital data output pins
- These are tri-state buffered
- The converted data is accessed only when CS = 0 and RD is forced low
- To calculate the output voltage, use the following formula

$$D_{out} = \frac{V_{in}}{step\ size}$$

Dout = digital data output (in decimal),

Vin = analog voltage and step size (resolution) is the smallest change

Vref/2

- It is used for the reference voltage
- If this pin is open (not connected), the analog input voltage is in the range of 0 to 5 volts (the same as the Vcc pin)
- If the analog input range needs to be 0 to 4 volts, Vref/2 is connected to 2 volts.

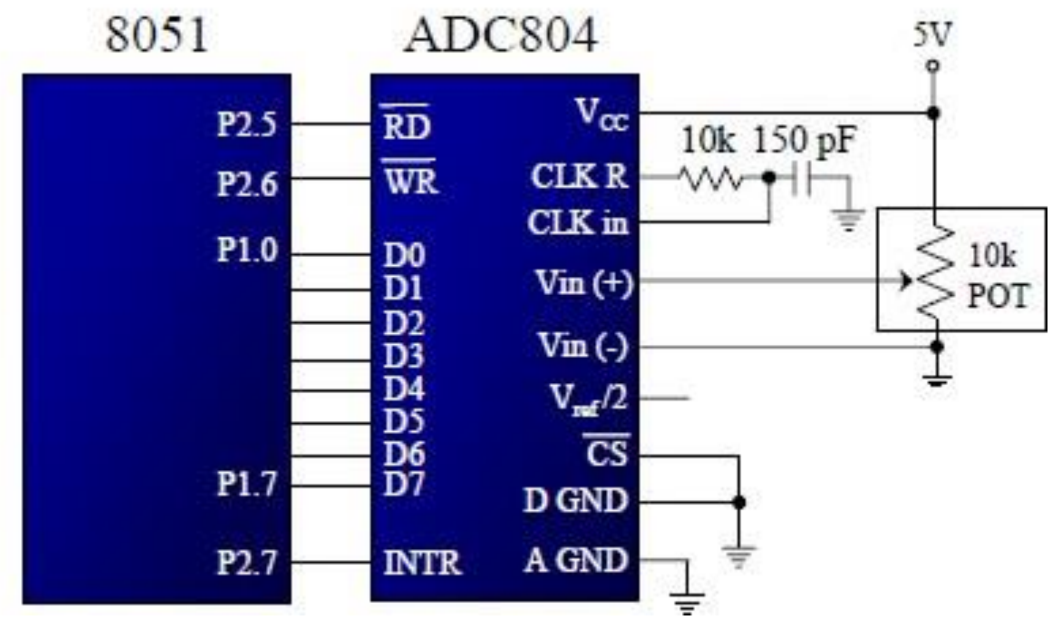
Analog ground and digital ground

- Analog ground is connected to the ground of the analog Vin
- Digital ground is connected to the ground of the Vcc pin
- To isolate the analog V_{in} signal from transient voltages caused by digital switching of the output D0 – D7. This contributes to the accuracy of the digital data output.

The following steps must be followed for data conversion by the ADC804 chip

- Make CS = 0 and send a low-to-high pulse to pin WR to start conversion
- Keep monitoring the INTR pin
- If INTR is low, the conversion is finished
- If the INTR is high, keep polling until it goes low
- After the INTR has become low, we make CS = 0 and send a high-to-low pulse to the RD pin to get the data out of the ADC804.

8051 Connection to ADC804 with Self-Clocking



Examine the ADC804 connection to the 8051 in Figure. Write a program to monitor the INTR pin and bring an analog input into register A. Then call a hex-to ACSII conversion and data display subroutines. Do this continuously. (NOV 2007)

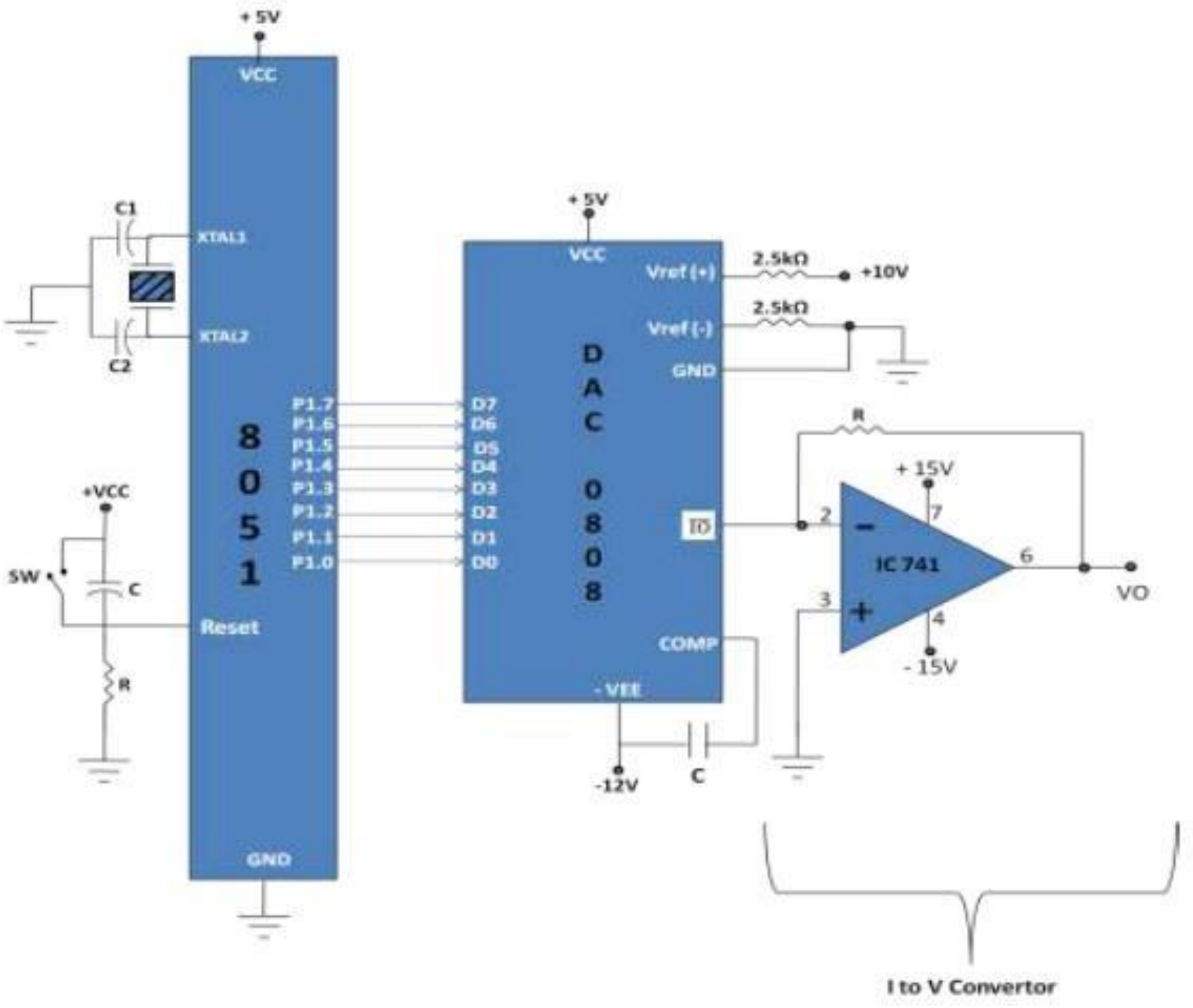
```
;p2.6=WR (start conversion needs to L-to-H pulse)
;p2.7 When low, end-of-conversion)
;p2.5=RD (a H-to-L will read the data from ADC chip)
;p1.0 – P1.7= D0 - D7 of the ADC804
            MOV P1,#0FFH            ;make P1 = input
BACK:       CLR P2.6                ;WR = 0
            SETB P2.6               ;WR = 1 L-to-H to start conversion
HERE:       JB P2.7,HERE            ;wait for end of conversion
            CLR P2.5                ;conversion finished, enable RD
            MOV A,P1                ;read the data
            ACALL CONVERSION         ;hex-to-ASCII conversion
            ACALL DATA_DISPLAY      ;display the data
            SETB p2.5                ;make RD=1 for next round
            SJMP BACK
```

5.7: DAC Interfacing with 8051

6. Explain the DAC interface with 8051. (May 2008)

Develop 8051 based system having 8Kbyte RAM to generate the triangular wave using DAC. (April 2017)

- Microcontroller is used in wide variety of applications like for measuring and control of physical quantity like temperature, pressure, speed, distance, etc.
- Microcontroller generates output which is in digital form.
- But the controlling system requires analog signal, so use DAC which converts digital data into equivalent analog voltage.
- In the figure shown, we use 8-bit DAC 0808. This IC converts digital data into equivalent analog Current.
- Hence we require an I to V converter to convert this current into equivalent voltage.



- According to theory of DAC Equivalent analog output is given as:

Ex:

1. IF data =00H [00000000], Vref= 10v

$$V0= 10\left[\frac{0}{2}+\frac{0}{4}+\frac{0}{8}+\frac{0}{16}+\frac{0}{32}+\frac{0}{64}+\frac{0}{128}+\frac{0}{256}\right]$$

Therefore, V0= 0 Volts.

2. If data is 80H [10000000], Vref= 10VTherefore, V0= 5 Volts.

$$V0=10\left[\frac{1}{2}+\frac{0}{4}+\frac{0}{8}+\frac{0}{16}+\frac{0}{32}+\frac{0}{64}+\frac{0}{128}+\frac{0}{256}\right]$$

Different Analog output voltages for different Digital signals are given as:

Data	Output Voltage
00H	0V
80H	5V
FFH	10V

Program to generate square wave:

Label	Mnemonics		Comments
	Opcode	Operand	
LOOP2 :	MOV	A,# 00	; Set Logic 0 level
	MOV	P1, AL	
	ACALL	Delay	;Generate timing delay
	MOV	A,#FF	;Set logic 1 level
	MOV	P1, AL	
	ACALL	Delay	; Generate timing delay
	SJMP	LOOP2	:Repeat to generates Square Wave
Delay:	MOV	R_O, #F0	:Delay Program
LOOP3:	DJNZ	$R_{O,}$ LOOP3	
	RET		

Program to generate Triangular wave:

Label	Mnemonics		Comments
	Opcode	Operand	
LOOP3 :	MOV	B,# 00	;Set logic 0
LOOP1:	MOV	A, B	;copy logic 0
	MOV	P1, A	
	INC	B	; Increment
	JNZ	LOOP1	; If ZF=0, jump to next
	MOV	B, #FF	Set logic 1
LOOP2:	MOV	A, B	;copy logic 1
	MOV	P1, A	
	DJNZ	B, LOOP2	; Decrement &jump to next
	SJMP	LOOP3	;Repeat

5.8: Temperature Sensors

7. Explain the interfacing of temperature sensor with 8051

- A thermistor responds to temperature change by changing resistance, but its response is not linear.

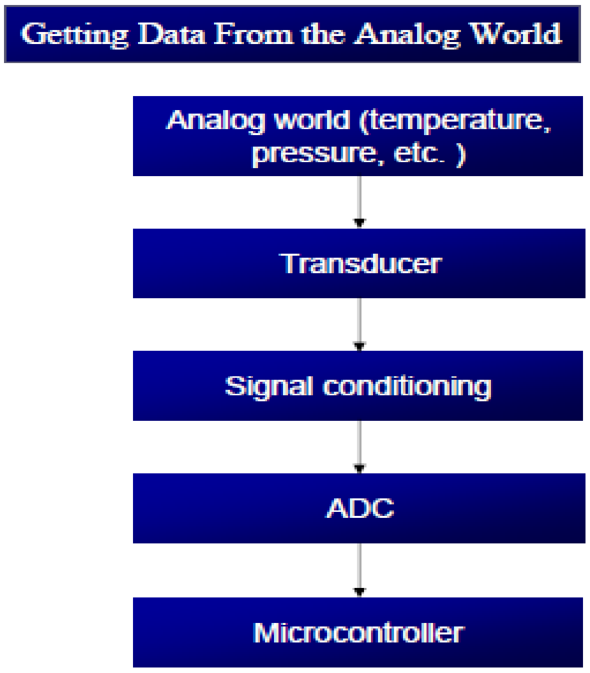
Temperature (C)	Tf (K ohms)
0	29.49
25	10
50	3.893
75	1.7
100	0.817

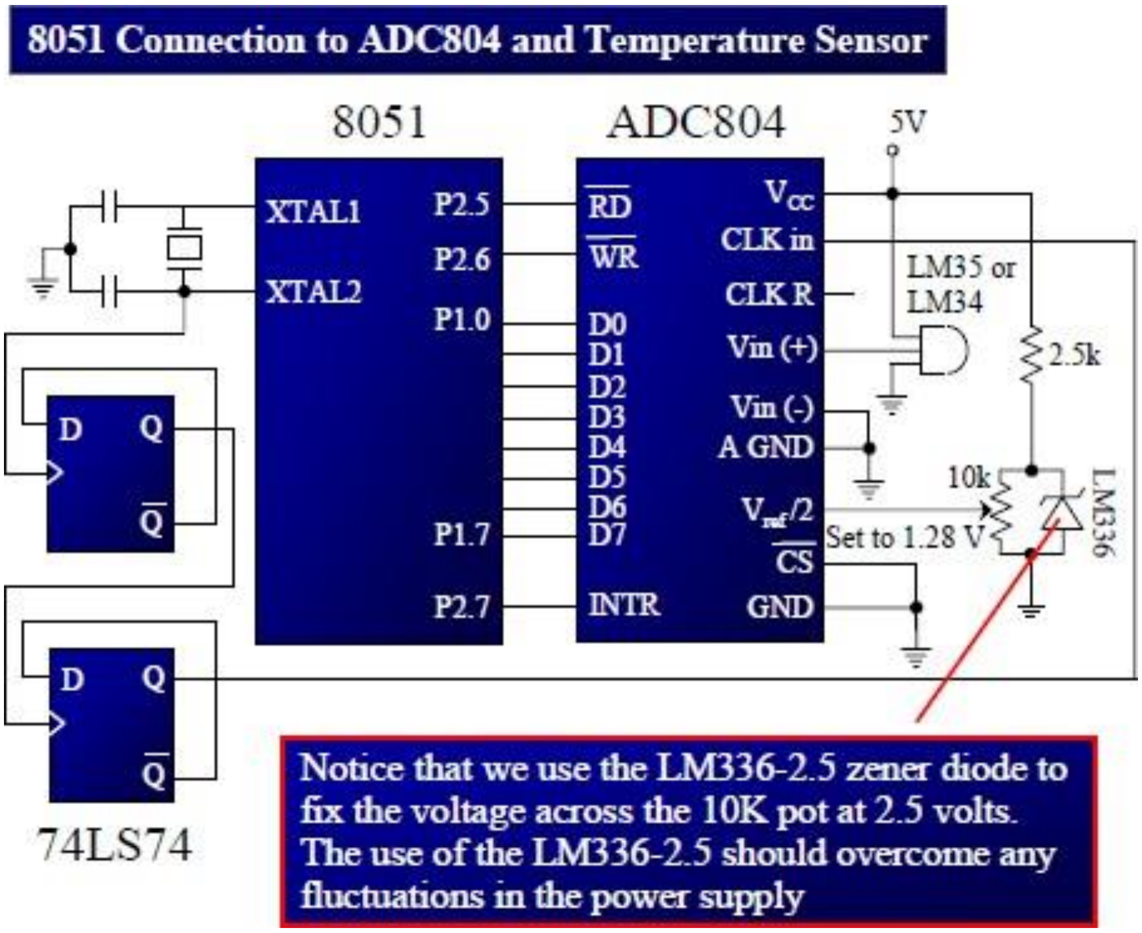
Signal conditioning is a widely used term in the world of data acquisition

- It is the conversion of the signals (voltage, current, charge, capacitance, and resistance) produced by transducers to voltage, which is sent to the input of an A to-D converter.

Signal conditioning can be a current-to voltage conversion or a signal amplification

- The thermistor changes resistance with temperature, while the change of resistance must be translated into voltage in order to be of any use to an ADC.
- The sensors of the LM34/LM35 series are precision integrated-circuit temperature sensors whose output voltage is linearly proportional to the Fahrenheit/Celsius temperature.
- The LM34/LM35 requires no external calibration since it is inherently calibrated.
- It outputs 10 mV for each degree of Fahrenheit/Celsius temperature.
- The LM336 is used to overcome any power fluctuation in the power supply.


Getting Data From the Analog World
Analog world (temperature, pressure, etc.)
Transducer
Signal conditioning
ADC
Microcontroller



8051 Connection to ADC804 and Temperature Sensor
8051
ADC804
5V
XTAL1
XTAL2
P2.5
P2.6
P1.0
P1.7
P2.7
RD
WR
D0
D1
D2
D3
D4
D5
D6
D7
INTR
Vcc
CLK in
CLK R
Vin (+)
Vin (-)
A GND
Vref/2
CS
GND
LM35 or LM34
2.5k
10k
Set to 1.28 V
LM336
D
Q
Q
74LS74
Notice that we use the LM336-2.5 zener diode to fix the voltage across the 10K pot at 2.5 volts. The use of the LM336-2.5 should overcome any fluctuations in the power supply

5.9 **: Interfacing to external memory**

8. Describe the external memory and its interfacing with 8051. (NOV 2009, May 2008)

Write short notes on memory addressing. (Nov /Dec 2007)

How a program memory and a data memory are interfaced with 8051. (NOV 2007)

5.9.1 : Connection to External Program ROM:

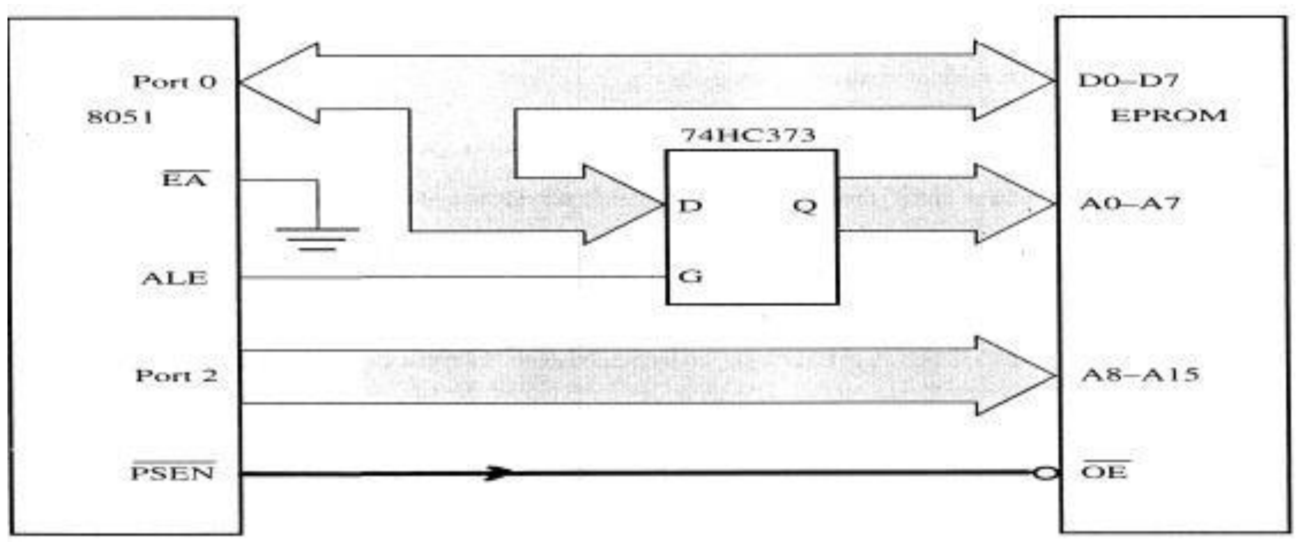
Explain how to interface ROM with the 8051. (NOV 2009)

We use RD to connect the 8051 to external ROM containing data.

For the ROM containing the program code, PSEN is used to fetch the code.

- 64K bytes are set aside for program code
- Program space is accessed using the program counter (PC) to locate and fetch instructions
- In some example we placed data in the code space and use the instruction MOVC A,@A+DPTR to get data, where C stands for code

- The other 64K bytes are set aside for data.
- The data memory space is accessed using the DPTR register and an instruction called MOVX , where X stands for external – The data memory space must be implemented externally.



- In the 8031/51, port 0 and port 2 provide the 16-bit address to access external memory.
- P0 provides the lower 8 bit address A0 – A7, and P2 provides the upper 8 bit address A8 – A15
- P0 is also used to provide the 8-bit data bus D0 – D7.
- P0.0 – P0.7 are used for both the address and data paths using address/data multiplexing.

EA (External access)

- Connect the **EA** pin to **Vcc** to indicate that the program code is stored in the microcontroller's **on-chip ROM**.
- To indicate that the program code is stored in **external ROM**, this pin must be connected to **GND.**

ALE (**address latch enable)** pin is an output pin for 8051

- ALE = 0, P0 is used for data path.
- ALE = 1, P0 is used for address path.
- To extract the address from the P0 pins we connect P0 to a 74LS373 and use the ALE pin to latch the address.

PSEN (program store enable) signal is an output signal for the 8051 microcontroller and must be connected to the OE pin of a ROM containing the program code.

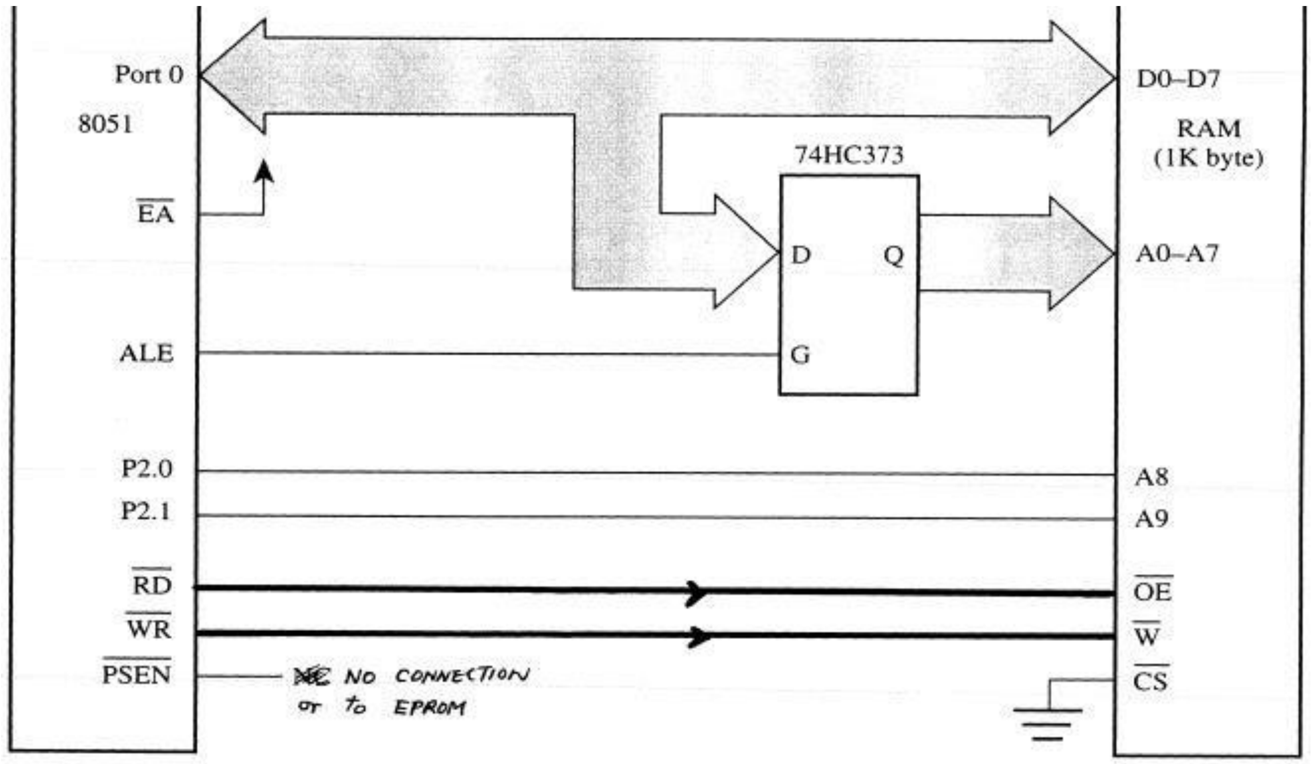
5.9.2 : Connection to External Data RAM

Interfacing external RAM with 8051

Write a brief note on external data move operations in 8051. (May 2016)

MOVX is a widely used instruction allowing access to external data memory space

- To bring externally stored data into the CPU, we use the instruction MOVX A,@DPTR
- To connect the 8051 to an external SRAM, we must use both RD (P3.7) and WR (P3.6)
- In writing data to external data RAM, we use the instruction MOVX @DPTR,A



- In the 8031/51, port 0 and port 2 provide the 16-bit address to access external memory.
- P0 provides the lower 8 bit address A0 – A7, and P2 provides the upper 8 bit address A8 – A15
- P0 is also used to provide the 8-bit data bus D0 – D7.
- P0.0 – P0.7 are used for both the address and data paths using address/data multiplexing.
- Data space is accessed using the program counter (PC) to locate, read and write data.

ALE (**address latch enable)** pin is an output pin for 8051

- ALE = 0, P0 is used for data path.
- ALE = 1, P0 is used for address path.
- To extract the address from the P0 pins we connect P0 to a 74LS373 and use the ALE pin to latch the address.
- EA & PSEN are not connected

****

5.10 : Interfacing stepper motor with 8051

9. Demonstrate the interfacing of the stepper motor with 8051 and explain its interfacing diagram and develop to rotate the motor in clock wise direction (April 2017, NOV 2016, May 2016, May 2010, May 2009, May 2008, May 2007, Nov 2015, 2014, 2013, 2011, 2010 & May 2013)
Describe in detail the microcontroller based system design with an example. (NOV 2102)

STEPPER MOTOR

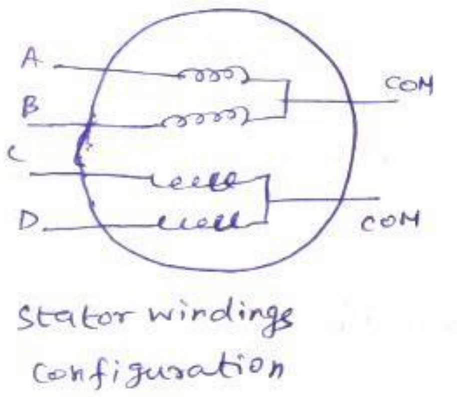
- A stepper motor is a brushless, synchronous electric motor that converts digital pulses into mechanical shaft rotation.
- Every revolution of the stepper motor is divided into a discrete number of steps, and the motor must be sent a separate pulse for each step.

Applications:

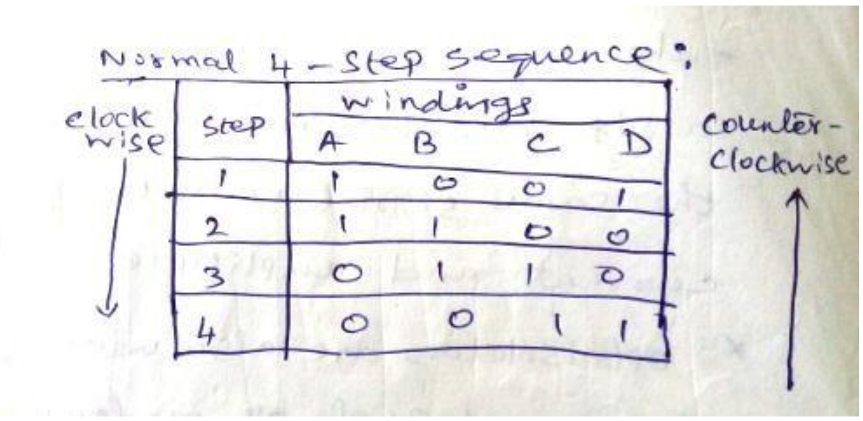
Stepper motors can be used for position control in disk drive, dot matrix printers, robotics etc.

Construction:

- The stepper motor has four stator windings that are paired with a centre tapped common.
- This type of a stepper motor is commonly referred to as a four phase unipolar stepper motor.
- The center tap allows a change of current direction in each of two coils when a winding is grounded, there by resulting in a polarity change of the stator.



Stator windings configuration


Normal 4-Step sequence:

clock wise ↓   Counter-Clockwise ↑

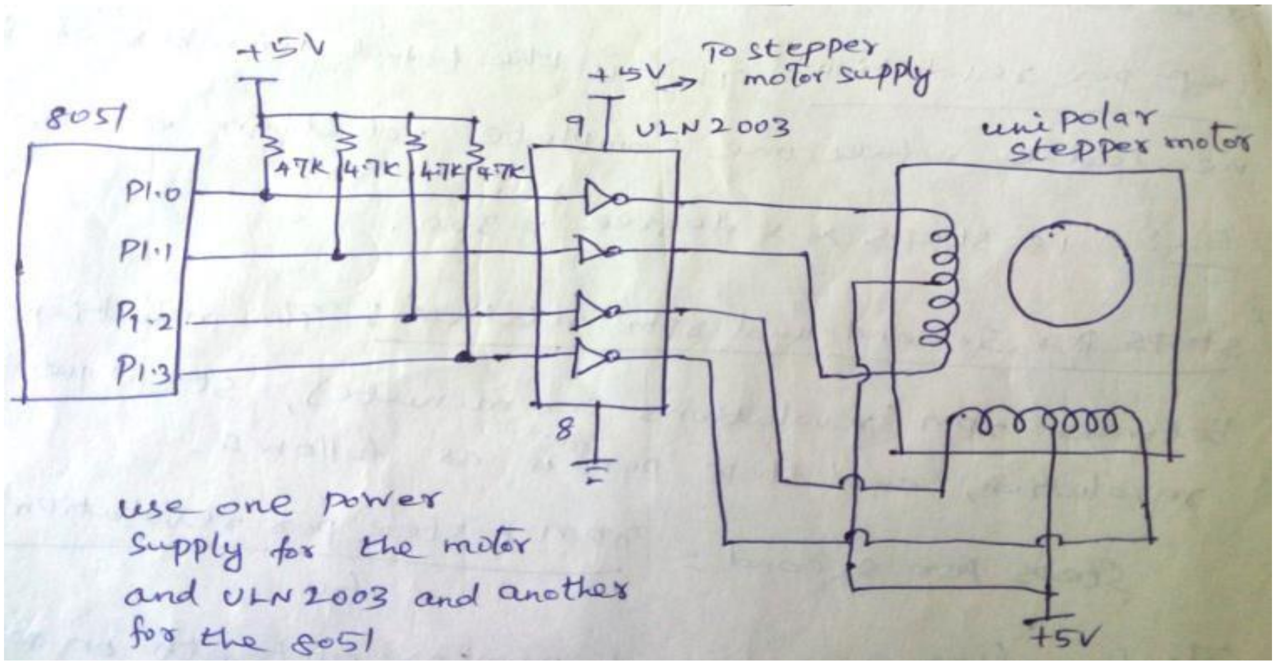
Step	Windings			
	A	B	C	D
1	1	0	0	1
2	1	1	0	0
3	0	1	1	0
4	0	0	1	1

- It has total of 6 leads: 4 leads representing the four stator windings and 2 commons for the center tapped leads.
- The stepper motor shaft moves in a fixed repeatable increment, which allows one to move it to a accurate position.
- As the direction of the current is changed, the polarity is also changed causing the reverse motion of the rotor.
- As the sequence of power is applied to each stator winding, the rotor will rotate.

- We can start with any of the sequences, once started, and must continue in the proper order.
- Step angle of the stepper motor is defined as the minimum degree of rotation associated with a single step.
- To calculate step angle, simply divide 360 by number of steps a motor takes to complete one revolution.
- Motor rotating in full mode takes 4 steps to complete a revolution ,so step angle can be calculated as step angle **ø = 360° / 4 =90.**
- By knowing the stepper motor step angle helps to move the motor in correct angular position**.**

8051 Microcontroller connection to Stepper motor:

- The stepper motor is connected with Microcontroller output port pins through a ULN2003 driver.
- The Microcontroller's pin can provide a maximum of 1-2mA current.
- Place a driver, such as the ULN 2003/ULN2803or a power transistor between the Microcontroller and the coil to energize the stator.



(Connection diagram between 8051 and Stepper motor)

- So when the microcontroller is giving pulses with particular frequency, the motor is rotated in clockwise or anticlockwise.

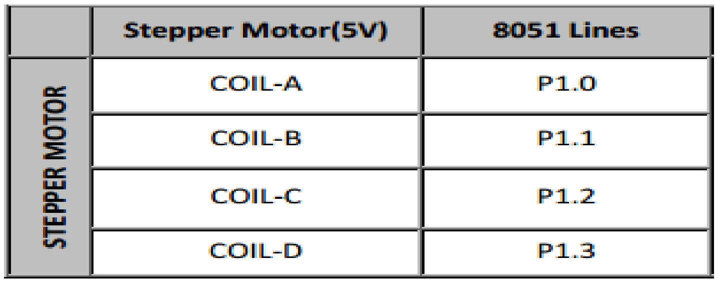
Program to interface Stepper motor with 8051:

The following steps show the 8051 connection to the stepper motor and its programming.

1. The common wires are connected to the positive side of the motor's power supply (+5V is sufficient).

2. Four stator windings are controlled by four bits of the 8051 port (P1.0 to P1.3). Driver ULN2003 used to energize the stator.

Pin assignment with 8051:



	Stepper Motor(5V)	8051 Lines
STEPPER MOTOR	COIL-A	P1.0
	COIL-B	P1.1
	COIL-C	P1.2
	COIL-D	P1.3

By giving the excitation as indicated above through port 1 we can rotate stepper motor in clockwise or anti clock wise direction.

Assembly Language Program:

To rotate motor in the clock wise direction (Forward direction):

```
         MOV A,#66H          ; Copy step value to Accumulator
BACK:    MOV P1,A            ; Copy step value from Accumulator to Port 1
         RR A                ; Rotate right to get consecutive step values
         ACALL DELAY         ; Call delay program
         SJMP BACK           ; repeat for next sequence.
DELAY:   MOV R2,#F0
DELAY2:  MOV R3,#FF
DELAY1:  DJNZ R3,# DELAY1
         DJNZ R2,# DELAY2
         RET                 ; Return to main program
```

To rotate motor in the counter (Anti) clock wise direction (Reverse direction):

```
         MOV A,#66H          ; Copy step value to Accumulator
BACK:    MOV P1,A            ; Copy step value from Accumulator to Port 1
         RL A                ; Rotate left to get consecutive step values
         ACALL DELAY         ; Call delay program
         SJMP BACK           ; repeat for next sequence.
DELAY:   MOV R2,#F0
DELAY2:  MOV R3,#FF
DELAY1:  DJNZ R3,# DELAY1
         DJNZ R2,# DELAY2
         RET                 ; Return to main program
```

To rotate motor in both forward and reverse direction:

```
          MOV A,#66H            ; Copy step value to Accumulator
FORWARD:  MOV P1,A              ; Copy step value from Accumulator to Port 1
          RR A                  ; Rotate right to get consecutive step values
          ACALL DELAY           ; Call delay program
          MOV R0,#04
          DJNZ R0,FORWARD

REVERSE:  MOV P1,A              ; Copy step value from Accumulator to Port 1
          RL A                  ; Rotate left to get consecutive step values
          ACALL DELAY           ; Call delay program
          MOV R0,#04
          DJNZ R0,REVERSE
          SJMP FORWARD          ; repeat for next sequence.

DELAY:    MOV R2,#F0            :Delay program
DELAY2:   MOV R3,#FF
DELAY1:   DJNZ R3,# DELAY1
          DJNZ R2,# DELAY2
          RET                   ; Return to main program
```

The PIC microcontroller was developed by General Instruments in 1975. PIC was developed when Microelectronics Division of General Instruments was testing its 16-bit CPU CP1600.
Although the CP1600 was a good CPU but it had low I/O performance. The PIC controller was used to offload the I/O the tasks from CPU to improve the overall performance of the system.

In 1985, General Instruments converted their Microelectronics Division to Microchip Technology.
PIC stands for Peripheral Interface Controller. The General Instruments used the acronyms Programmable Interface Controller and Programmable Intelligent Computer for the initial PICs (PIC1640 and PIC1650).

In 1993, Microchip Technology launched the 8-bit PIC16C84 with EEPROM which could be programmed using serial programming method.
The improved version of PIC16C84 with flash memory (PIC18F84 and PIC18F84A) hit the market in 1998.
Development:
Since 1998, Microchip Technology continuously developed new high performance microcontrollers with new complex architecture and enhanced in-built peripherals. PIC microcontroller is based on Harvard architecture. At present PIC microcontrollers are widely used for industrial purpose due to its high performance ability at low power consumption. It is also very famous among hobbyists due to moderate

cost and easy availability of its supporting software and hardware tools like compilers, simulators, debuggers etc. The 8-bit PIC microcontroller is divided into following four categories on the basis of internal architecture:

1. Base Line PIC
2. Mid-Range PIC
3. Enhanced Mid-Range PIC
4. PIC18

1. Base Line PIC

Base Line PICs are the least complex PIC microcontrollers. These microcontrollers work on 12-bit instruction architecture which means that the word size of instruction sets are of 12 bits for these controllers. These are smallest and cheapest PICs, available with 6 to 40 pin packaging. The small size and low cost of Base Line PIC replaced the traditional ICs like 555, logic gates etc. in industries.

2. Mid-Range PIC

Mid-Range PICs are based on 14-bit instruction architecture and are able to work up to 20 MHz speed. These controllers are available with 8 to 64 pin packaging. These microcontrollers are available with different peripherals like ADC, PWM, Op-Amps and different communication protocols like USART, SPI, I2C (TWI), etc. which make them widely usable microcontrollers not only for industry but for hobbyists as well.

3. Enhanced Mid-Range PIC

These controllers are enhanced version of Mid-Range core. This range of controllers provides additional performance, greater flash memory and high speed at very low power consumption. This range of PIC also includes multiple peripherals and supports protocols like USART, SPI, I2C and so on.

4. PIC18

PIC18 range is based on 16-bit instruction architecture incorporating advanced RISC architecture which makes it highest performer among the all 8-bit PIC families. The PIC18 range is integrated with new age communication protocols like USB, CAN, LIN, Ethernet (TCP/IP protocol) to communicate with local and/or internet based networks. This range also supports the connectivity of Human Interface Devices like touch panels etc.
MIPS stand for Millions of Instructions per Second

Besides 8-bit microcontrollers, Microchip also manufactures 16-bit and 32-bit microcontrollers. Recently Microchip developed XLP (Extreme Low Power) series microcontrollers which are based on NanoWatt technology. These controllers draw current in order of nanoamperes(nA).
PIC microcontrollers are also available with extended voltage ranges which reduce the frequency range. The operating voltage range of these PICs is 2.0-6.0 volts. The letter ‘L’ is included in controller’s name to denote extended voltage range controllers. For example, PIC16LFxxx (Operating voltage 2.0-6.0 volts).

The following section covers the PIC architecture in further detail. PIC18 series has been selected for the study because it is enhanced series of 8-bit PIC microcontroller. In this series, PIC18F4550 has been chosen to describe the architecture and other features due its moderate complexity.

Architecture:

PIC microcontrollers are based on advanced RISC architecture. RISC stands for Reduced Instruction Set Computing. In this architecture, the instruction set of hardware gets reduced which increases the execution rate (speed) of system.

PIC microcontrollers follow Harvard architecture for internal data transfer. In Harvard architecture there are two separate memories for program and data. These two memories are accessed through different buses for data communication between memories and CPU core. This architecture improves the speed of system over Von Neumann architecture in which program and data are fetched from the same memory using the same bus. PIC18 series controllers are based on 16-bit instruction set.

The question may arise that if PIC18 are called 8-bit microcontrollers, then what about them being based on 16-bit instructions set. 'PIC18 is an 8-bit microcontroller' this statement means that the CPU core can receive/transmit or process a maximum of 8-bit data at a time. On the other hand the statement 'PIC18 microcontrollers are based on 16-bit instruction set' means that the assembly instruction sets are of 16-bit. The data memory is interfaced with 8-bit bus and program memory is interfaced with 16-bit bus as depicted in the following figure

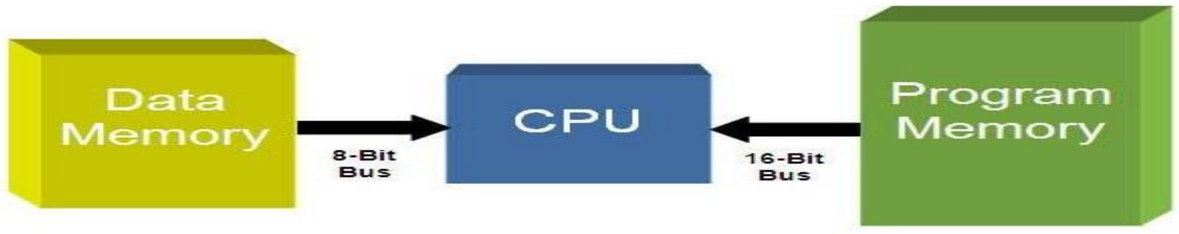
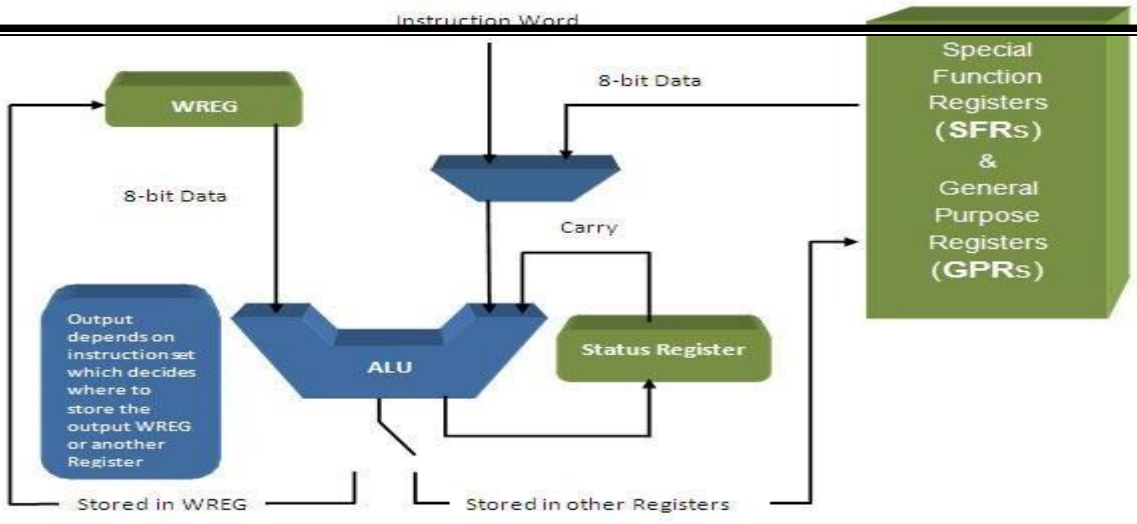


Fig. 4: Simple Block Diagram Of CPU Interfacing With Data And Program Memory In PIC

PIC18 Harvard Architecture

PIC microcontroller contains an 8-bit ALU (Arithmetic Logic Unit) and an 8-bit Working Register (Accumulator). There are different GPRs (General Purpose Registers) and SFRs (Special Function Registers) in a PIC microcontroller. The overall system performs 8-bit arithmetic and logic functions. These functions usually need one or two operands. One of the operands is stored in WREG (Accumulator) and the other one is stored in GPR/SFR. The two data is processed by ALU and stored in WREG or other registers.



Instruction Word
8-bit Data
Special Function Registers (SFRs) & General Purpose Registers (GPRs)
WREG
8-bit Data
Carry
Output depends on instruction set which decides where to store the output WREG or another Register
ALU
Status Register
Stored in WREG
Stored in other Registers

Fig. 5: Simple Block Diagram of Data Processing in PIC18 Harvard

The above process occurs in a single machine cycle. In PIC microcontroller, a single machine cycle consists of 4 oscillation periods. Thus an instruction needs 4 clock periods to be executed. This makes it faster than other 8051 microcontrollers.

Pipelining:
Early processors and controllers could fetch or execute a single instruction in a unit of time. The PIC microcontrollers are able to fetch and execute the instructions in the same unit of time thus increasing their instruction throughput. This technique is known as instruction pipelining where the processing of instructions is split into a number of independent steps.

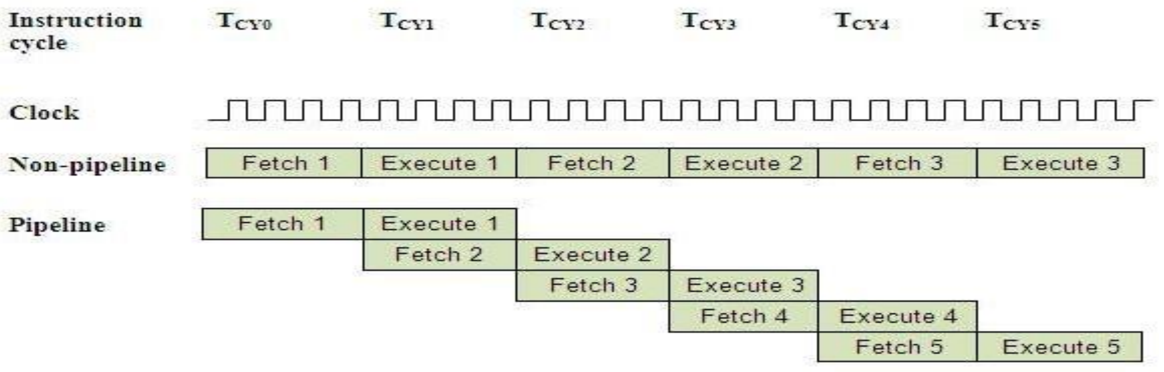


Fig. 6: Diagram Showing Instruction Pipelining Technique In PIC

Features and Peripherals

The PIC18F consists of the following features and peripherals.

Features:
- C Compiler Optimized Architecture with Optional Extended Instruction Set
- 100,000 Erase/Write Cycle Enhanced Flash
- Program Memory Typical
- 1,000,000 Erase/Write Cycle Data EEPROM Memory Typical
- Flexible oscillator option

  Four Crystal modes, including High-Precision PLL for USB

- o Two External Clock modes, Up to 48 MHz
- o Internal Oscillator: 8 user-selectable frequencies, from 31 kHz to 8 MHz
- o Dual Oscillator Options allow Microcontroller and USB module to Run at different Clock Speeds

Peripherals:
The PIC18F4550 microcontroller consists of following peripherals:

· I/O Ports: PIC18F4550 have 5 (PORTA, PORTB, PORTC, PORTD and PORTE) 8-bit input-output ports. PortB & PortD have 8 I/O pins each. Although other three ports are 8-bit ports but they do not have eight I/O pins. Although the 8-bit input and output are given to these ports, but the pins which do not exist, are masked internally.

Memory: PIC18F4550 consists of three different memory sections:

1. **Flash Memory:** Flash memory is used to store the program downloaded by a user on to the microcontroller. Flash memory is non-volatile, i.e., it retains the program even after the power is cut-off. PIC18F4550 has 32KB of Flash Memory.

2. **EEPROM:** This is also a nonvolatile memory which is used to store data like values of certain variables. PIC18F4550 has 256 Bytes of EEPROM.

3. **SRAM:** Static Random Access Memory is the volatile memory of the microcontroller, i.e., it loses its data as soon as the power is cut off. PIC18F4550 is equipped with 2 KB of internal SRAM.

· Oscillator: The PIC18F series has flexible clock options. An external clock of up to 48 MHz can be applied to this series. These controllers also consist of an internal oscillator which provides eight selectable frequency options varying from 31 KHz to 8 MHz.

· 8x8 Multiplier: The PIC18F4550 includes an 8 x 8 multiplier hardware. This hardware performs the multiplications in single machine cycle. This gives higher computational throughput and reduces operation cycle & code length.

· ADC Interface: PIC18F4550 is equipped with 13 ADC (Analog to Digital Converter) channels of 10-bits resolution. ADC reads the analog input, for example, a sensor input and converts it into digital value that can be understood by the microcontroller.

· Timers/Counters: PIC18F4550 has four timer/counters. There is one 8-bit timer and the remaining timers have option to select 8 or 16 bit mode. Timers are useful for generating precision actions, for example, creating precise time delays between two operations.

· Interrupts: PIC18F4550 consists of three external interrupts sources. There are 20 internal interrupts which are associated with different peripherals like USART, ADC, Timers, and so on.

· EUSART: Enhanced USART (Universal Synchronous and Asynchronous Serial Receiver and Transmitter) module is full-duplex asynchronous system. It can also be configured as half-duplex synchronous system. The Enhanced USART has the feature for automatic baud rate detection and calibration, automatic wake-up on Sync Break reception and 12-bit Break character transmit. These features make it ideally suited for use in Local Interconnect Network bus (LIN bus) systems.

· ICSP and ICD: PIC18F series controllers have In Circuit Serial Programming facility to program the Flash Memory which can be programmed without removing the IC from the circuit. ICD (In Circuit Debugger) allows for hardware debugging of the controller while it is in the application circuit.

· SPI: PIC18F supports 3-wire SPI communication between two devices on a common clock source. The data rate of SPI is more than that of USART.

· I2C: PIC18F supports Two Wire Interface (TWI) or I2C communication between two devices. It can work as both Master and Slave device.

ARM Processors

ARM, previously Advanced RISC Machine, originally Acorn RISC Machine, is a family of reduced instruction set computing (RISC) architectures for computer processors, configured for various environments. Arm Holdings develops the architecture and licenses it to other companies, who design their own products that implement one of those architectures—including systems-on-chips (SoC) and systems-on-modules (SoM) that incorporate memory, interfaces, radios, etc. It also designs cores that implement this instruction set and licenses these designs to a number of companies that incorporate those core designs into their own products.

Processors that have a RISC architecture typically require fewer transistors than those with a complex instruction set computing (CISC) architecture (such as the x86 processors found in most personal computers), which improves cost, power consumption, and heat dissipation. These characteristics are desirable for light, portable, battery-powered devices—including smartphones, laptops and tablet computers, and other embedded systems. For supercomputers, which consume large amounts of electricity, ARM could also be a power-efficient solution.

Arm Holdings periodically releases updates to the architecture. Architecture versions ARMv3 to ARMv7 support 32-bit address space (pre-ARMv3 chips, made before Arm Holdings was formed, as used in the Acorn Archimedes, had 26-bit address space) and 32-bit arithmetic; most architectures have 32-bit fixed-length instructions. The Thumb version supports a variable-length instruction set that provides both 32- and 16-bit instructions for improved code density. Some older cores can also provide hardware execution of Java bytecodes. Released in 2011, the ARMv8-A architecture added support for a 64-bit address space and 64-bit arithmetic with its new 32-bit fixed-length instruction set.

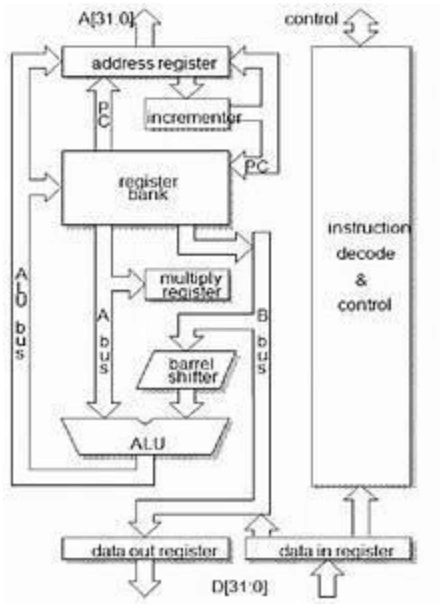
With over 100 billion ARM processors produced as of 2017, ARM is the most widely used instruction set architecture and the instruction set architecture produced in the largest quantity. Currently, the widely used Cortex cores, older "classic" cores, and specialized SecurCore cores variants are available for each of these to include or exclude optional capabilities.

The British computer manufacturer Acorn Computers first developed the Acorn RISC Machine architecture (ARM) in the 1980s to use in its personal computers. Its first ARM-based products were coprocessor modules for the BBC Micro series of computers. After the successful BBC Micro computer, Acorn Computers considered how to move on from the relatively simple MOS Technology 6502 processor to address business markets like the one that was soon dominated by the IBM PC, launched in 1981.

The Acorn Business Computer (ABC) plan required that a number of second processors be made to work with the BBC Micro platform, but processors such as the Motorola 68000 and National Semiconductor 32016 were considered unsuitable, and the 6502 was not powerful enough for a graphics-based user interface

According to Sophie Wilson, all the processors tested at that time performed about the same, with about a 4 Mbit/second bandwidth.

After testing all available processors and finding them lacking, Acorn decided it needed a new architecture. Inspired by papers from the Berkeley RISC project, Acorn considered designing its own processor. A visit to the Western Design Center in Phoenix, where the 6502 was being updated by what was effectively a single-person company, showed Acorn engineers Steve Furber and Sophie Wilson they did not need massive resources and state-of-the-art research and development facilities.



Wilson developed the instruction set, writing a simulation of the processor in BBC BASIC that ran on a BBC Micro with a 6502 second processor. This convinced Acorn engineers they were on the right track. Wilson approached Acorn's CEO, Hermann Hauser, and requested more resources. Hauser gave his approval and assembled a small team to implement Wilson's model in hardware