

# Chapter 01

# Introduction to Computers

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

In this chapter you will learn about:

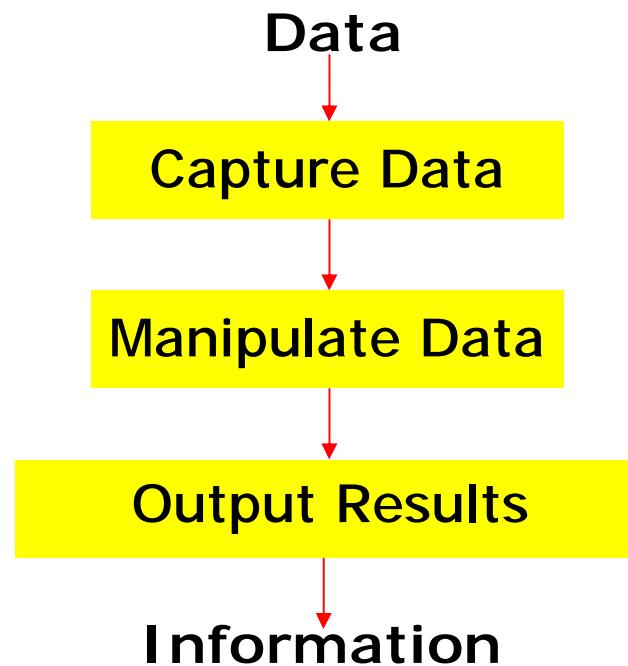
- § Computer
- § Data processing
- § Characteristic features of computers
- § Computers' evolution to their present form
- § Computer generations
- § Characteristic features of each computer generation

# Computer

- § The word computer comes from the word “compute”, which means, “to calculate”
- § Thereby, a computer is an electronic device that can perform arithmetic operations at high speed
- § A computer is also called a *data processor* because it can store, process, and retrieve data whenever desired

# Data Processing

The activity of processing data using a computer is called *data processing*



*Data* is raw material used as input and *information* is processed data obtained as output of data processing

# Characteristics of Computers

- 1) Automatic:** Given a job, computer can work on it automatically without human interventions
- 2) Speed:** Computer can perform data processing jobs very fast, usually measured in microseconds ( $10^{-6}$ ), nanoseconds ( $10^{-9}$ ), and picoseconds ( $10^{-12}$ )
- 3) Accuracy:** Accuracy of a computer is consistently high and the degree of its accuracy depends upon its design. Computer errors caused due to incorrect input data or unreliable programs are often referred to as *Garbage-In-Garbage-Out (GIGO)*

(Continued on next slide)

# Characteristics of Computers

(Continued from previous slide..)

- 4) Diligence:** Computer is free from monotony, tiredness, and lack of concentration. It can continuously work for hours without creating any error and without grumbling
- 5) Versatility:** Computer is capable of performing almost any task, if the task can be reduced to a finite series of logical steps
- 6) Power of Remembering:** Computer can store and recall any amount of information because of its secondary storage capability. It forgets or loses certain information only when it is asked to do so

(Continued on next slide)

# Characteristics of Computers

(Continued from previous slide..)

- 7) No I.Q.:** A computer does only what it is programmed to do. It cannot take its own *decision* in this regard
  
- 8) No Feelings:** Computers are devoid of emotions. Their judgement is based on the instructions given to them in the form of programs that are written by us (human beings)

(Continued on next slide)

# Evolution of Computers

- § Blaise Pascal invented the first *mechanical adding machine* in 1642
- § Baron Gottfried Wilhelm von Leibniz invented the first *calculator for multiplication* in 1671
- § *Keyboard machines* originated in the United States around 1880
- § Around 1880, Herman Hollerith came up with the concept of *punched cards* that were extensively used as input media until late 1970s

# Evolution of Computers

(Continued from previous slide..)

- § Charles Babbage is considered to be the father of modern digital computers
- § He designed “Difference Engine” in 1822
- § He designed a *fully automatic analytical engine* in 1842 for performing basic arithmetic functions
- § His efforts established a number of principles that are fundamental to the design of any digital computer

(Continued on next slide)

# Some Well Known Early Computers

- § The Mark I Computer (1937-44)
- § The Atanasoff-Berry Computer (1939-42)
- § The ENIAC (1943-46)
- § The EDVAC (1946-52)
- § The EDSAC (1947-49)
- § Manchester Mark I (1948)
- § The UNIVAC I (1951)

# Computer Generations

- § “*Generation*” in computer talk is a step in technology. It provides a framework for the growth of computer industry
- § Originally it was used to distinguish between various hardware technologies, but now it has been extended to include both hardware and software
- § Till today, there are five computer generations

(Continued on next slide)

# Computer Generations

(Continued from previous slide..)

Generation (Period)	Key hardware technologies	Key software technologies	Key characteristics	Some representative systems
First (1942-1955)	<ul style="list-style-type: none"> <li>§ Vacuum tubes</li> <li>§ Electromagnetic relay memory</li> <li>§ Punched cards secondary storage</li> </ul>	<ul style="list-style-type: none"> <li>§ Machine and assembly languages</li> <li>§ Stored program concept</li> <li>§ Mostly scientific applications</li> </ul>	<ul style="list-style-type: none"> <li>§ Bulky in size</li> <li>§ Highly unreliable</li> <li>§ Limited commercial use and costly</li> <li>§ Difficult commercial production</li> <li>§ Difficult to use</li> </ul>	<ul style="list-style-type: none"> <li>§ ENIAC</li> <li>§ EDVAC</li> <li>§ EDSAC</li> <li>§ UNIVAC I</li> <li>§ IBM 701</li> </ul>
Second (1955-1964)	<ul style="list-style-type: none"> <li>§ Transistors</li> <li>§ Magnetic cores memory</li> <li>§ Magnetic tapes</li> <li>§ Disks for secondary storage</li> </ul>	<ul style="list-style-type: none"> <li>§ Batch operating system</li> <li>§ High-level programming languages</li> <li>§ Scientific and commercial applications</li> </ul>	<ul style="list-style-type: none"> <li>§ Faster, smaller, more reliable and easier to program than previous generation systems</li> <li>§ Commercial production was still difficult and costly</li> </ul>	<ul style="list-style-type: none"> <li>§ Honeywell 400</li> <li>§ IBM 7030</li> <li>§ CDC 1604</li> <li>§ UNIVAC LARC</li> </ul>

(Continued on next slide)

# Computer Generations

(Continued from previous slide..)

Generation (Period)	Key hardware technologies	Key software technologies	Key characteristics	Some rep. systems
Third (1964-1975)	<ul style="list-style-type: none"> <li>§ ICs with SSI and MSI technologies</li> <li>§ Larger magnetic cores memory</li> <li>§ Larger capacity disks and magnetic tapes secondary storage</li> <li>§ Minicomputers; upward compatible family of computers</li> </ul>	<ul style="list-style-type: none"> <li>§ Timesharing operating system</li> <li>§ Standardization of high-level programming languages</li> <li>§ Unbundling of software from hardware</li> </ul>	<ul style="list-style-type: none"> <li>§ Faster, smaller, more reliable, easier and cheaper to produce</li> <li>§ Commercially, easier to use, and easier to upgrade than previous generation systems</li> <li>§ Scientific, commercial and interactive on-line applications</li> </ul>	<ul style="list-style-type: none"> <li>§ IBM 360/370</li> <li>§ PDP-8</li> <li>§ PDP-11</li> <li>§ CDC 6600</li> </ul>

(Continued on next slide)

# Computer Generations

(Continued from previous slide..)

Generation (Period)	Key hardware Technologies	Key software technologies	Key characteristics	Some rep. systems
Fourth (1975-1989)	<ul style="list-style-type: none"> <li>§ ICs with VLSI technology</li> <li>§ Microprocessors; semiconductor memory</li> <li>§ Larger capacity hard disks as in-built secondary storage</li> <li>§ Magnetic tapes and floppy disks as portable storage media</li> <li>§ Personal computers</li> <li>§ Supercomputers based on parallel vector processing and symmetric multiprocessing technologies</li> <li>§ Spread of high-speed computer networks</li> </ul>	<ul style="list-style-type: none"> <li>§ Operating systems for PCs with GUI and multiple windows on a single terminal screen</li> <li>§ Multiprocessing OS with concurrent programming languages</li> <li>§ UNIX operating system with C programming language</li> <li>§ Object-oriented design and programming</li> <li>§ PC, Network-based, and supercomputing applications</li> </ul>	<ul style="list-style-type: none"> <li>§ Small, affordable, reliable, and easy to use PCs</li> <li>§ More powerful and reliable mainframe systems and supercomputers</li> <li>§ Totally general purpose machines</li> <li>§ Easier to produce commercially</li> <li>§ Easier to upgrade</li> <li>§ Rapid software development possible</li> </ul>	<ul style="list-style-type: none"> <li>§ IBM PC and its clones</li> <li>§ Apple II</li> <li>§ TRS-80</li> <li>§ VAX 9000</li> <li>§ CRAY-1</li> <li>§ CRAY-2</li> <li>§ CRAY-X/MP</li> </ul>

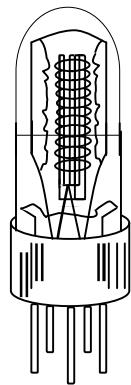
(Continued on next slide)

# Computer Generations

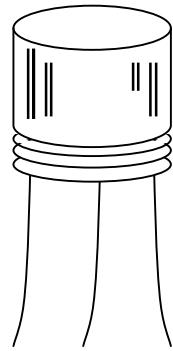
(Continued from previous slide..)

Generation (Period)	Key hardware technologies	Key software technologies	Key characteristics	Some rep. systems
Fifth (1989- Present)	<ul style="list-style-type: none"> <li>§ ICs with ULSI technology</li> <li>§ Larger capacity main memory, hard disks with RAID support</li> <li>§ Optical disks as portable read-only storage media</li> <li>§ Notebooks, powerful desktop PCs and workstations</li> <li>§ Powerful servers, supercomputers</li> <li>§ Internet</li> <li>§ Cluster computing</li> </ul>	<ul style="list-style-type: none"> <li>§ Micro-kernel based, multithreading, distributed OS</li> <li>§ Parallel programming libraries like MPI &amp; PVM</li> <li>§ JAVA</li> <li>§ World Wide Web</li> <li>§ Multimedia, Internet applications</li> <li>§ More complex supercomputing applications</li> </ul>	<ul style="list-style-type: none"> <li>§ Portable computers</li> <li>§ Powerful, cheaper, reliable, and easier to use desktop machines</li> <li>§ Powerful supercomputers</li> <li>§ High uptime due to hot-pluggable components</li> <li>§ Totally general purpose machines</li> <li>§ Easier to produce commercially, easier to upgrade</li> <li>§ Rapid software development possible</li> </ul>	<ul style="list-style-type: none"> <li>§ IBM notebooks</li> <li>§ Pentium PCs</li> <li>§ SUN Workstations</li> <li>§ IBM SP/2</li> <li>§ SGI Origin 2000</li> <li>§ PARAM 10000</li> </ul>

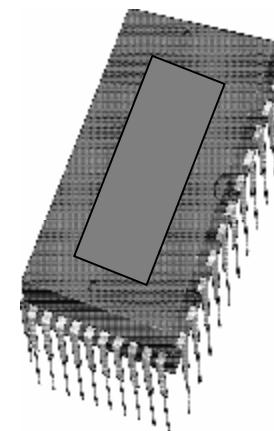
## Electronic Devices Used in Computers of Different Generations



(a) A Vacuum Tube



(b) A Transistor



(c) An IC Chip

# Key Words/Phrases

- § Computer
- § Computer generations
- § Computer Supported Cooperative Working (CSCW)
- § Data
- § Data processing
- § Data processor
- § First-generation computers
- § Fourth-generation computers
- § Garbage-in-garbage-out (GIGO)
- § Graphical User Interface (GUI)
- § Groupware
- § Information
- § Integrated Circuit (IC)
- § Large Scale Integration (VLSI)
- § Medium Scale Integration (MSI)
- § Microprocessor
- § Personal Computer (PC)
- § Second-generation computers
- § Small Scale Integration (SSI)
- § Stored program concept
- § Third-generation computers
- § Transistor
- § Ultra Large Scale Integration (ULSI)
- § Vacuum tubes

## Chapter 02

# Basic Computer Organization

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

In this chapter you will learn about:

- § Basic operations performed by all types of computer systems
- § Basic organization of a computer system
- § Input unit and its functions
- § Output unit and its functions
- § Storage unit and its functions
- § Types of storage used in a computer system

(Continued on next slide)

# Learning Objectives

(Continued from previous slide..)

§ Arithmetic Logic Unit (ALU)

§ Control Unit (CU)

§ Central Processing Unit (CPU)

§ Computer as a system

# The Five Basic Operations of a Computer System

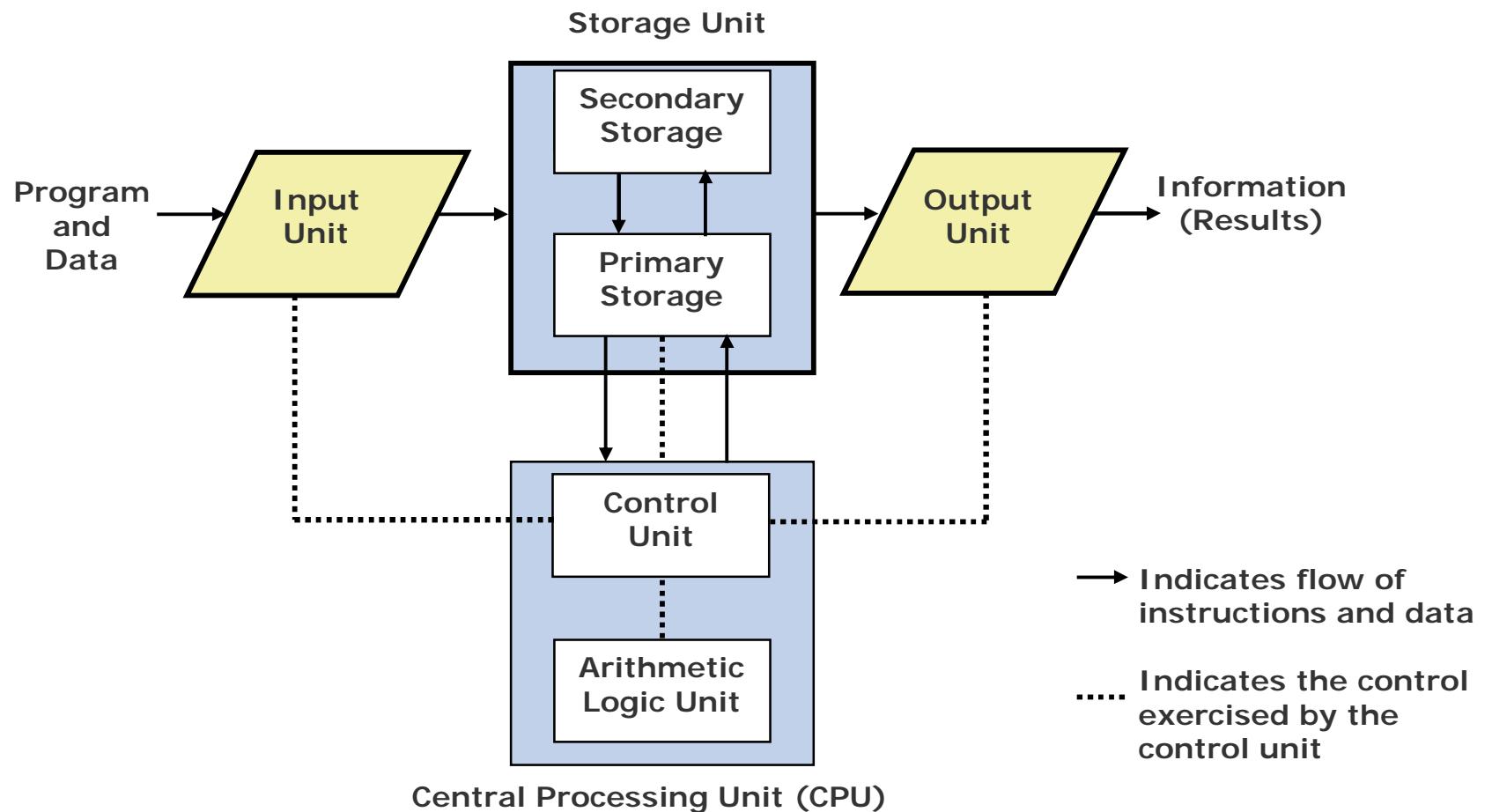
- § **Inputting.** The process of entering data and instructions into the computer system
- § **Storing.** Saving data and instructions to make them readily available for initial or additional processing whenever required
- § **Processing.** Performing arithmetic operations (add, subtract, multiply, divide, etc.) or logical operations (comparisons like equal to, less than, greater than, etc.) on data to convert them into useful information

(Continued on next slide)

# The Five Basic Operations of a Computer System

- § **Outputting.** The process of producing useful information or results for the user such as a printed report or visual display
- § **Controlling.** Directing the manner and sequence in which all of the above operations are performed

# Basic Organization of a Computer System



# Input Unit

An input unit of a computer system performs the following functions:

1. It accepts (or reads) instructions and data from outside world
2. It converts these instructions and data in computer acceptable form
3. It supplies the converted instructions and data to the computer system for further processing

# Output Unit

An output unit of a computer system performs the following functions:

1. It accepts the results produced by the computer, which are in coded form and hence, cannot be easily understood by us
2. It converts these coded results to human acceptable (readable) form
3. It supplies the converted results to outside world

# Storage Unit

The storage unit of a computer system holds (or stores) the following :

1. Data and instructions required for processing (received from input devices)
2. Intermediate results of processing
3. Final results of processing, before they are released to an output device

# Two Types of Storage

## § Primary storage

- § Used to hold running program instructions
- § Used to hold data, intermediate results, and results of ongoing processing of job(s)
- § Fast in operation
- § Small Capacity
- § Expensive
- § Volatile (loses data on power dissipation)

(Continued on next slide)

# Two Types of Storage

(Continued from previous slide..)

## § Secondary storage

- § Used to hold stored program instructions
- § Used to hold data and information of stored jobs
- § Slower than primary storage
- § Large Capacity
- § Lot cheaper than primary storage
- § Retains data even without power

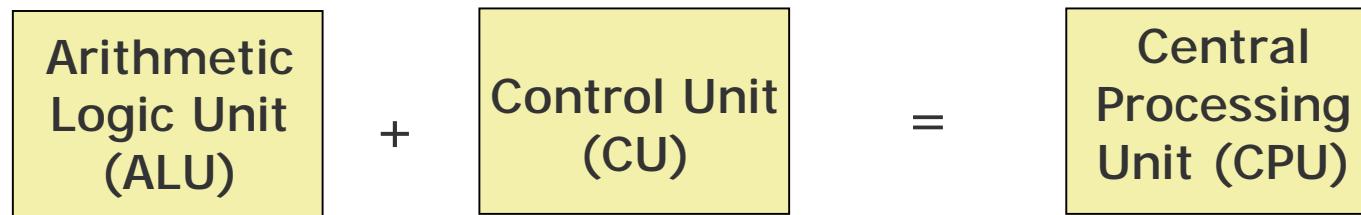
# Arithmetic Logic Unit (ALU)

Arithmetic Logic Unit of a computer system is the place where the actual executions of instructions takes place during processing operation

# Control Unit (CU)

Control Unit of a computer system manages and coordinates the operations of all other components of the computer system

# Central Processing Unit (CPU)



- § It is the brain of a computer system
- § It is responsible for controlling the operations of all other units of a computer system

# The System Concept

A system has following three characteristics:

1. A system has more than one element
2. All elements of a system are logically related
3. All elements of a system are controlled in a manner to achieve the system goal

A computer is a system as it comprises of integrated components (input unit, output unit, storage unit, and CPU) that work together to perform the steps called for in the executing program

# Key Words/Phrases

- § Arithmetic Logic Unit (ALU)
- § Auxiliary storage
- § Central Processing Unit (CPU)
- § Computer system
- § Control Unit (CU)
- § Controlling
- § Input interface
- § Input unit
- § Inputting
- § Main memory
- § Output interface
- § Output unit
- § Outputting
- § Primary storage
- § Processing
- § Secondary storage
- § Storage unit
- § Storing
- § System

# Chapter 03

# Number Systems

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

In this chapter you will learn about:

- § Non-positional number system
- § Positional number system
- § Decimal number system
- § Binary number system
- § Octal number system
- § Hexadecimal number system

(Continued on next slide)

# Learning Objectives

(Continued from previous slide..)

- § Convert a number's base
  - § Another base to decimal base
  - § Decimal base to another base
  - § Some base to another base
- § Shortcut methods for converting
  - § Binary to octal number
  - § Octal to binary number
  - § Binary to hexadecimal number
  - § Hexadecimal to binary number
- § Fractional numbers in binary number system

# Number Systems

Two types of number systems are:

- § Non-positional number systems
- § Positional number systems

# Non-positional Number Systems

## § Characteristics

- § Use symbols such as I for 1, II for 2, III for 3, IIII for 4, IIIII for 5, etc
- § Each symbol represents the same value regardless of its position in the number
- § The symbols are simply added to find out the value of a particular number

## § Difficulty

- § It is difficult to perform arithmetic with such a number system

# Positional Number Systems

## § Characteristics

- § Use only a few symbols called digits
- § These symbols represent different values depending on the position they occupy in the number

(Continued on next slide)

# Positional Number Systems

(Continued from previous slide..)

§ The value of each digit is determined by:

1. The digit itself
2. The position of the digit in the number
3. The base of the number system

(base = total number of digits in the number system)

§ The maximum value of a single digit is always equal to one less than the value of the base

# Decimal Number System

## Characteristics

- § A positional number system
- § Has 10 symbols or digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Hence, its base = 10
- § The maximum value of a single digit is 9 (one less than the value of the base)
- § Each position of a digit represents a specific power of the base (10)
- § We use this number system in our day-to-day life

(Continued on next slide)

# Decimal Number System

(Continued from previous slide..)

## Example

$$2586_{10} = (2 \times 10^3) + (5 \times 10^2) + (8 \times 10^1) + (6 \times 10^0)$$

$$= 2000 + 500 + 80 + 6$$

# Binary Number System

## Characteristics

- § A positional number system
- § Has only 2 symbols or digits (0 and 1). Hence its base = 2
- § The maximum value of a single digit is 1 (one less than the value of the base)
- § Each position of a digit represents a specific power of the base (2)
- § This number system is used in computers

(Continued on next slide)

# Binary Number System

(Continued from previous slide..)

## Example

$$\begin{aligned}10101_2 &= (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) \\&= 16 + 0 + 4 + 0 + 1 \\&= 21_{10}\end{aligned}$$

# Representing Numbers in Different Number Systems

In order to be specific about which number system we are referring to, it is a common practice to indicate the base as a subscript. Thus, we write:

$$10101_2 = 21_{10}$$

# Bit

- § Bit stands for **binary digit**
- § A bit in computer terminology means either a 0 or a 1
- § A binary number consisting of  $n$  bits is called an  $n$ -bit number

# Octal Number System

## Characteristics

- § A positional number system
- § Has total 8 symbols or digits (0, 1, 2, 3, 4, 5, 6, 7).  
Hence, its base = 8
- § The maximum value of a single digit is 7 (one less than the value of the base)
- § Each position of a digit represents a specific power of the base (8)

(Continued on next slide)

# Octal Number System

(Continued from previous slide..)

§ Since there are only 8 digits, 3 bits ( $2^3 = 8$ ) are sufficient to represent any octal number in binary

## Example

$$\begin{aligned}2057_8 &= (2 \times 8^3) + (0 \times 8^2) + (5 \times 8^1) + (7 \times 8^0) \\&= 1024 + 0 + 40 + 7 \\&= 1071_{10}\end{aligned}$$

# Hexadecimal Number System

## Characteristics

- § A positional number system
- § Has total 16 symbols or digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F). Hence its base = 16
- § The symbols A, B, C, D, E and F represent the decimal values 10, 11, 12, 13, 14 and 15 respectively
- § The maximum value of a single digit is 15 (one less than the value of the base)

(Continued on next slide)

# Hexadecimal Number System

(Continued from previous slide..)

- § Each position of a digit represents a specific power of the base (16)
- § Since there are only 16 digits, 4 bits ( $2^4 = 16$ ) are sufficient to represent any hexadecimal number in binary

## Example

$$\begin{aligned}1AF_{16} &= (1 \times 16^2) + (A \times 16^1) + (F \times 16^0) \\&= 1 \times 256 + 10 \times 16 + 15 \times 1 \\&= 256 + 160 + 15 \\&= 431_{10}\end{aligned}$$

# Converting a Number of Another Base to a Decimal Number

## Method

- Step 1: Determine the column (positional) value of each digit
- Step 2: Multiply the obtained column values by the digits in the corresponding columns
- Step 3: Calculate the sum of these products

(Continued on next slide)

# Converting a Number of Another Base to a Decimal Number

(Continued from previous slide..)

## Example

$$4706_8 = ?_{10}$$

$$\begin{aligned} 4706_8 &= 4 \times 8^3 + 7 \times 8^2 + 0 \times 8^1 + 6 \times 8^0 \\ &= 4 \times 512 + 7 \times 64 + 0 + 6 \times 1 \\ &= 2048 + 448 + 0 + 6 \leftarrow \text{Sum of these products} \\ &= 2502_{10} \end{aligned}$$

Common values multiplied by the corresponding digits

# Converting a Decimal Number to a Number of Another Base

## Division-Remainder Method

- Step 1: Divide the decimal number to be converted by the value of the new base
- Step 2: Record the remainder from Step 1 as the rightmost digit (least significant digit) of the new base number
- Step 3: Divide the quotient of the previous divide by the new base

(Continued on next slide)

# Converting a Decimal Number to a Number of Another Base

(Continued from previous slide..)

Step 4: Record the remainder from Step 3 as the next digit (to the left) of the new base number

Repeat Steps 3 and 4, recording remainders from right to left, until the quotient becomes zero in Step 3

Note that the last remainder thus obtained will be the most significant digit (MSD) of the new base number

(Continued on next slide)

# Converting a Decimal Number to a Number of Another Base

(Continued from previous slide..)

## Example

$$952_{10} = ?_8$$

## Solution:

8	952	Remainder
	119	S 0
	14	7
	1	6
	0	1

$$\text{Hence, } 952_{10} = 1670_8$$

# Converting a Number of Some Base to a Number of Another Base

## Method

- Step 1: Convert the original number to a decimal number (base 10)
- Step 2: Convert the decimal number so obtained to the new base number

(Continued on next slide)

# Converting a Number of Some Base to a Number of Another Base

(Continued from previous slide..)

## Example

$$545_6 = ?_4$$

Solution:

Step 1: Convert from base 6 to base 10

$$\begin{aligned}545_6 &= 5 \times 6^2 + 4 \times 6^1 + 5 \times 6^0 \\&= 5 \times 36 + 4 \times 6 + 5 \times 1 \\&= 180 + 24 + 5 \\&= 209_{10}\end{aligned}$$

(Continued on next slide)

# Converting a Number of Some Base to a Number of Another Base

(Continued from previous slide..)

Step 2: Convert  $209_{10}$  to base 4

4	209	Remainders
	52	1
	13	0
	3	1
	0	3

Hence,  $209_{10} = 3101_4$

So,  $545_6 = 209_{10} = 3101_4$

Thus,  $545_6 = 3101_4$

# Shortcut Method for Converting a Binary Number to its Equivalent Octal Number

## Method

- Step 1: Divide the digits into groups of three starting from the right
- Step 2: Convert each group of three binary digits to one octal digit using the method of binary to decimal conversion

(Continued on next slide)

# Shortcut Method for Converting a Binary Number to its Equivalent Octal Number

(Continued from previous slide..)

## Example

$$1101010_2 = ?_8$$

Step 1: Divide the binary digits into groups of 3 starting from right

001      101      010

Step 2: Convert each group into one octal digit

$$001_2 = 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 1$$

$$101_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 5$$

$$010_2 = 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 2$$

Hence,  $1101010_2 = 152_8$

# Shortcut Method for Converting an Octal Number to Its Equivalent Binary Number

## Method

- Step 1: Convert each octal digit to a 3 digit binary number (the octal digits may be treated as decimal for this conversion)
- Step 2: Combine all the resulting binary groups (of 3 digits each) into a single binary number

(Continued on next slide)

# Shortcut Method for Converting an Octal Number to Its Equivalent Binary Number

*(Continued from previous slide..)*

## Example

$$562_8 = ?_2$$

**Step 1:** Convert each octal digit to 3 binary digits

$$5_8 = 101_2, \quad 6_8 = 110_2, \quad 2_8 = 010_2$$

## Step 2: Combine the binary groups

$$562_8 = \underline{101} \quad \underline{110} \quad \underline{010}$$

5            6            2

Hence,  $562_8 = 101110010_2$

# Shortcut Method for Converting a Binary Number to its Equivalent Hexadecimal Number

## Method

- Step 1: Divide the binary digits into groups of four starting from the right
- Step 2: Combine each group of four binary digits to one hexadecimal digit

(Continued on next slide)

# Shortcut Method for Converting a Binary Number to its Equivalent Hexadecimal Number

(Continued from previous slide..)

## Example

$$111101_2 = ?_{16}$$

Step 1: Divide the binary digits into groups of four starting from the right

$$\underline{0011} \quad \underline{1101}$$

Step 2: Convert each group into a hexadecimal digit

$$0011_2 = 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 3_{10} = 3_{16}$$

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 3_{10} = D_{16}$$

$$\text{Hence, } 111101_2 = 3D_{16}$$

# Shortcut Method for Converting a Hexadecimal Number to its Equivalent Binary Number

## Method

Step 1: Convert the decimal equivalent of each hexadecimal digit to a 4 digit binary number

Step 2: Combine all the resulting binary groups (of 4 digits each) in a single binary number

(Continued on next slide)

# Shortcut Method for Converting a Hexadecimal Number to its Equivalent Binary Number

(Continued from previous slide..)

## Example

$$2AB_{16} = ?_2$$

Step 1: Convert each hexadecimal digit to a 4 digit binary number

$$2_{16} = 2_{10} = 0010_2$$

$$A_{16} = 10_{10} = 1010_2$$

$$B_{16} = 11_{10} = 1011_2$$

# Shortcut Method for Converting a Hexadecimal Number to its Equivalent Binary Number

(Continued from previous slide..)

Step 2: Combine the binary groups

$$2AB_{16} = \begin{array}{c} \underline{0010} \quad \underline{1010} \quad \underline{1011} \\ 2 \qquad \qquad A \qquad \qquad B \end{array}$$

Hence,  $2AB_{16} = 001010101011_2$

# Fractional Numbers

*Fractional numbers* are formed same way as decimal number system

In general, a number in a number system with base  $b$  would be written as:

$$a_n a_{n-1} \dots a_0 . a_{-1} a_{-2} \dots a_{-m}$$

And would be interpreted to mean:

$$a_n \times b^n + a_{n-1} \times b^{n-1} + \dots + a_0 \times b^0 + a_{-1} \times b^{-1} + a_{-2} \times b^{-2} + \dots + a_{-m} \times b^{-m}$$

The symbols  $a_n, a_{n-1}, \dots, a_{-m}$  in above representation should be one of the  $b$  symbols allowed in the number system

# Formation of Fractional Numbers in Binary Number System (Example)

Position	Binary Point									
	4	3	2	1	0	.	-1	-2	-3	-4
Position Value	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$		$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$
Quantity Represented	16	8	4	2	1		$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$

(Continued on next slide)

# Formation of Fractional Numbers in Binary Number System (Example)

(Continued from previous slide..)

## Example

$$\begin{aligned}110.101_2 &= 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \\&= 4 + 2 + 0 + 0.5 + 0 + 0.125 \\&= 6.625_{10}\end{aligned}$$

# Formation of Fractional Numbers in Octal Number System (Example)

Position	3	2	1	0	.	-1	-2	-3
Position Value	$8^3$	$8^2$	$8^1$	$8^0$	$8^{-1}$	$8^{-2}$	$8^{-3}$	
Quantity Represented	512	64	8	1	$\frac{1}{8}$	$\frac{1}{64}$	$\frac{1}{512}$	

(Continued on next slide)

# Formation of Fractional Numbers in Octal Number System (Example)

(Continued from previous slide..)

## Example

$$\begin{aligned}127.54_8 &= 1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 5 \times 8^{-1} + 4 \times 8^{-2} \\&= 64 + 16 + 7 + \frac{5}{8} + \frac{4}{64} \\&= 87 + 0.625 + 0.0625 \\&= 87.6875_{10}\end{aligned}$$

# Key Words/Phrases

- § Base
- § Binary number system
- § Binary point
- § Bit
- § Decimal number system
- § Division-Remainder technique
- § Fractional numbers
- § Hexadecimal number system

- § Least Significant Digit (LSD)
- § Memory dump
- § Most Significant Digit (MSD)
- § Non-positional number system
- § Number system
- § Octal number system
- § Positional number system

# Chapter 04

# Computer Codes

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

In this chapter you will learn about:

- § Computer data
- § Computer codes: representation of data in binary
- § Most commonly used computer codes
- § Collating sequence

# Data Types

- § **Numeric Data** consists of only numbers 0, 1, 2, ..., 9
- § **Alphabetic Data** consists of only the letters A, B, C, ..., Z, in both uppercase and lowercase, and blank character
- § **Alphanumeric Data** is a string of symbols where a symbol may be one of the letters A, B, C, ..., Z, in either uppercase or lowercase, or one of the digits 0, 1, 2, ..., 9, or a special character, such as + - \* / , . ( ) = etc.

# Computer Codes

- § Computer codes are used for internal representation of data in computers
- § As computers use binary numbers for internal data representation, computer codes use binary coding schemes
- § In binary coding, every symbol that appears in the data is represented by a group of bits
- § The group of bits used to represent a symbol is called a byte

(Continued on next slide)

# Computer Codes

(Continued from previous slide..)

- § As most modern coding schemes use 8 bits to represent a symbol, the term byte is often used to mean a group of 8 bits
- § Commonly used computer codes are BCD, EBCDIC, and ASCII

# BCD

- § BCD stands for Binary Coded Decimal
- § It is one of the early computer codes
- § It uses 6 bits to represent a symbol
- § It can represent 64 ( $2^6$ ) different characters

# Coding of Alphabetic and Numeric Characters in BCD

Char	BCD Code		Octal
	Zone	Digit	
A	11	0001	61
B	11	0010	62
C	11	0011	63
D	11	0100	64
E	11	0101	65
F	11	0110	66
G	11	0111	67
H	11	1000	70
I	11	1001	71
J	10	0001	41
K	10	0010	42
L	10	0011	43
M	10	0100	44

Char	BCD Code		Octal
	Zone	Digit	
N	10	0101	45
O	10	0110	46
P	10	0111	47
Q	10	1000	50
R	10	1001	51
S	01	0010	22
T	01	0011	23
U	01	0100	24
V	01	0101	25
W	01	0110	26
X	01	0111	27
Y	01	1000	30
Z	01	1001	31

(Continued on next slide)

# Coding of Alphabetic and Numeric Characters in BCD

(Continued from previous slide..)

Character	BCD Code		Octal Equivalent
	Zone	Digit	
1	00	0001	01
2	00	0010	02
3	00	0011	03
4	00	0100	04
5	00	0101	05
6	00	0110	06
7	00	0111	07
8	00	1000	10
9	00	1001	11
0	00	1010	12

# BCD Coding Scheme (Example 1)

## Example

Show the binary digits used to record the word BASE in BCD

## Solution:

B = 110010 in BCD binary notation

A = 110001 in BCD binary notation

S = 010010 in BCD binary notation

E = 110101 in BCD binary notation

So the binary digits

110010    110001    010010    110101  
    B            A            S            E

will record the word BASE in BCD

# BCD Coding Scheme (Example 2)

## *Example*

Using octal notation, show BCD coding for the word DIGIT

### **Solution:**

D = 64 in BCD octal notation

I = 71 in BCD octal notation

G = 67 in BCD octal notation

I = 71 in BCD octal notation

T = 23 in BCD octal notation

Hence, BCD coding for the word DIGIT in octal notation will be

$$\begin{array}{ccccc} \underline{64} & \underline{71} & \underline{67} & \underline{71} & \underline{23} \\ D & I & G & I & T \end{array}$$

# EBCDIC

- § EBCDIC stands for Extended Binary Coded Decimal Interchange Code
- § It uses 8 bits to represent a symbol
- § It can represent 256 ( $2^8$ ) different characters

# Coding of Alphabetic and Numeric Characters in EBCDIC

Char	EBCDIC Code		Hex
	Digit	Zone	
A	1100	0001	C1
B	1100	0010	C2
C	1100	0011	C3
D	1100	0100	C4
E	1100	0101	C5
F	1100	0110	C6
G	1100	0111	C7
H	1100	1000	C8
I	1100	1001	C9
J	1101	0001	D1
K	1101	0010	D2
L	1101	0011	D3
M	1101	0100	D4

Char	EBCDIC Code		Hex
	Digit	Zone	
N	1101	0101	D5
O	1101	0110	D6
P	1101	0111	D7
Q	1101	1000	D8
R	1101	1001	D9
S	1110	0010	E2
T	1110	0011	E3
U	1110	0100	E4
V	1110	0101	E5
W	1110	0110	E6
X	1110	0111	E7
Y	1110	1000	E8
Z	1110	1001	E9

(Continued on next slide)

# Coding of Alphabetic and Numeric Characters in EBCDIC

(Continued from previous slide..)

Character	EBCDIC Code		Hexadecima l Equivalent
	Digit	Zone	
0	1111	0000	F0
1	1111	0001	F1
2	1111	0010	F2
3	1111	0011	F3
4	1111	0100	F4
5	1111	0101	F5
6	1111	0110	F6
7	1111	0111	F7
8	1111	1000	F8
9	1111	1001	F9

# Zoned Decimal Numbers

- § Zoned decimal numbers are used to represent numeric values (positive, negative, or unsigned) in EBCDIC
- § A sign indicator (C for plus, D for minus, and F for unsigned) is used in the zone position of the rightmost digit
- § Zones for all other digits remain as F, the zone value for numeric characters in EBCDIC
- § In zoned format, there is only one digit per byte

# Examples Zoned Decimal Numbers

Numeric Value	EBCDIC	Sign Indicator
345	F3F4F5	F for unsigned
+345	F3F4C5	C for positive
-345	F3F4D5	D for negative

# Packed Decimal Numbers

§ Packed decimal numbers are formed from zoned decimal numbers in the following manner:

Step 1: The zone half and the digit half of the rightmost byte are reversed

Step 2: All remaining zones are dropped out

§ Packed decimal format requires fewer number of bytes than zoned decimal format for representing a number

§ Numbers represented in packed decimal format can be used for arithmetic operations

# Examples of Conversion of Zoned Decimal Numbers to Packed Decimal Format

Numeric Value	EBCDIC	Sign Indicator
345	F3F4F5	345F
+345	F3F4C5	345C
-345	F3F4D5	345D
3456	F3F4F5F6	03456F

# EBCDIC Coding Scheme

## Example

Using binary notation, write EBCDIC coding for the word BIT. How many bytes are required for this representation?

### Solution:

B = 1100 0010 in EBCDIC binary notation

I = 1100 1001 in EBCDIC binary notation

T = 1110 0011 in EBCDIC binary notation

Hence, EBCDIC coding for the word BIT in binary notation will be

11000010    11001001    11100011  
      B              I              T

3 bytes will be required for this representation because each letter requires 1 byte (or 8 bits)

# ASCII

- § ASCII stands for American Standard Code for Information Interchange.
- § ASCII is of two types – ASCII-7 and ASCII-8
- § ASCII-7 uses 7 bits to represent a symbol and can represent 128 ( $2^7$ ) different characters
- § ASCII-8 uses 8 bits to represent a symbol and can represent 256 ( $2^8$ ) different characters
- § First 128 characters in ASCII-7 and ASCII-8 are same

# Coding of Numeric and Alphabetic Characters in ASCII

Character	ASCII-7 / ASCII-8		Hexadecimal Equivalent
	Zone	Digit	
0	0011	0000	30
1	0011	0001	31
2	0011	0010	32
3	0011	0011	33
4	0011	0100	34
5	0011	0101	35
6	0011	0110	36
7	0011	0111	37
8	0011	1000	38
9	0011	1001	39

(Continued on next slide)

# Coding of Numeric and Alphabetic Characters in ASCII

(Continued from previous slide..)

Character	ASCII-7 / ASCII-8		Hexadecimal Equivalent
	Zone	Digit	
A	0100	0001	41
B	0100	0010	42
C	0100	0011	43
D	0100	0100	44
E	0100	0101	45
F	0100	0110	46
G	0100	0111	47
H	0100	1000	48
I	0100	1001	49
J	0100	1010	4A
K	0100	1011	4B
L	0100	1100	4C
M	0100	1101	4D

(Continued on next slide)

# Coding of Numeric and Alphabetic Characters in ASCII

(Continued from previous slide..)

Character	ASCII-7 / ASCII-8		Hexadecimal Equivalent
	Zone	Digit	
N	0100	1110	4E
O	0100	1111	4F
P	0101	0000	50
Q	0101	0001	51
R	0101	0010	52
S	0101	0011	53
T	0101	0100	54
U	0101	0101	55
V	0101	0110	56
W	0101	0111	57
X	0101	1000	58
Y	0101	1001	59
Z	0101	1010	5A

# ASCII-7 Coding Scheme

## Example

Write binary coding for the word BOY in ASCII-7. How many bytes are required for this representation?

### Solution:

B = 1000010 in ASCII-7 binary notation

O = 1001111 in ASCII-7 binary notation

Y = 1011001 in ASCII-7 binary notation

Hence, binary coding for the word BOY in ASCII-7 will be

1000010    1001111    1011001  
      B              O              Y

Since each character in ASCII-7 requires one byte for its representation and there are 3 characters in the word BOY, 3 bytes will be required for this representation

# ASCII-8 Coding Scheme

## Example

Write binary coding for the word SKY in ASCII-8. How many bytes are required for this representation?

## Solution:

S = 01010011 in ASCII-8 binary notation

K = 01001011 in ASCII-8 binary notation

Y = 01011001 in ASCII-8 binary notation

Hence, binary coding for the word SKY in ASCII-8 will be

01010011    01001011    01011001  
              S              K              Y

Since each character in ASCII-8 requires one byte for its representation and there are 3 characters in the word SKY, 3 bytes will be required for this representation

# Unicode

## § Why Unicode:

- § No single encoding system supports all languages
- § Different encoding systems conflict

## § Unicode features:

- § Provides a consistent way of encoding multilingual plain text
- § Defines codes for characters used in all major languages of the world
- § Defines codes for special characters, mathematical symbols, technical symbols, and diacritics

# Unicode

## § Unicode features (continued):

- § Capacity to encode as many as a million characters
- § Assigns each character a unique numeric value and name
- § Reserves a part of the code space for private use
- § Affords simplicity and consistency of ASCII, even corresponding characters have same code
- § Specifies an algorithm for the presentation of text with bi-directional behavior

## § Encoding Forms

- § UTF-8, UTF-16, UTF-32

# Collating Sequence

- § Collating sequence defines the assigned ordering among the characters used by a computer
- § Collating sequence may vary, depending on the type of computer code used by a particular computer
- § In most computers, collating sequences follow the following rules:
  1. Letters are considered in alphabetic order  
(A < B < C ... < Z)
  2. Digits are considered in numeric order  
(0 < 1 < 2 ... < 9)

# Sorting in EBCDIC

## Example

Suppose a computer uses EBCDIC as its internal representation of characters. In which order will this computer sort the strings 23, A1, 1A?

## Solution:

In EBCDIC, numeric characters are treated to be greater than alphabetic characters. Hence, in the said computer, numeric characters will be placed after alphabetic characters and the given string will be treated as:

A1 < 1A < 23

Therefore, the sorted sequence will be: A1, 1A, 23.

# Sorting in ASCII

## Example

Suppose a computer uses ASCII for its internal representation of characters. In which order will this computer sort the strings 23, A1, 1A, a2, 2a, aA, and Aa?

## Solution:

In ASCII, numeric characters are treated to be less than alphabetic characters. Hence, in the said computer, numeric characters will be placed before alphabetic characters and the given string will be treated as:

$$1A < 23 < 2a < A1 < Aa < a2 < aA$$

Therefore, the sorted sequence will be: 1A, 23, 2a, A1, Aa, a2, and aA

# Key Words/Phrases

- § Alphabetic data
- § Alphanumeric data
- § American Standard Code for Information Interchange (ASCII)
- § Binary Coded Decimal (BCD) code
- § Byte
- § Collating sequence
- § Computer codes
- § Control characters
- § Extended Binary-Coded Decimal Interchange Code (EBCDIC)
- § Hexadecimal equivalent
- § Numeric data
- § Octal equivalent
- § Packed decimal numbers
- § Unicode
- § Zoned decimal numbers

# Chapter 05

# Computer Arithmetic

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

In this chapter you will learn about:

- § Reasons for using binary instead of decimal numbers
- § Basic arithmetic operations using binary numbers
  - § Addition (+)
  - § Subtraction (-)
  - § Multiplication (\*)
  - § Division (/)

# Binary over Decimal

- § Information is handled in a computer by electronic/electrical components
- § Electronic components operate in binary mode (can only indicate two states – on (1) or off (0))
- § Binary number system has only two digits (0 and 1), and is suitable for expressing two possible states
- § In binary system, computer circuits only have to handle two binary digits rather than ten decimal digits causing:
  - § Simpler internal circuit design
  - § Less expensive
  - § More reliable circuits
- § Arithmetic rules/processes possible with binary numbers

# Examples of a Few Devices that work in Binary Mode

Binary State	On (1)	Off (0)
Bulb		
Switch		
Circuit Pulse		

# Binary Arithmetic

- § Binary arithmetic is simple to learn as binary number system has only two digits – 0 and 1
  
- § Following slides show rules and example for the four basic arithmetic operations using binary numbers

# Binary Addition

Rule for binary addition is as follows:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$1 + 1 = 0$  plus a carry of 1 to next higher column

# Binary Addition (Example 1)

## Example

Add binary numbers 10011 and 1001 in both decimal and binary form

## Solution

Binary	Decimal
carry 11	carry 1
10011	19
+1001	+9
—————	—————
11100	28
—————	—————

In this example, carry are generated for first and second columns

# Binary Addition (Example 2)

## Example

Add binary numbers 100111 and 11011 in both decimal and binary form

## Solution

Binary	Decimal
carry 11111	carry 1
100111	39
+11011	+27
—————	—————
<u>1000010</u>	<u>66</u>

The addition of three 1s can be broken up into two steps. First, we add only two 1s giving 10 ( $1 + 1 = 10$ ). The third 1 is now added to this result to obtain 11 (a 1 sum with a 1 carry). Hence,  $1 + 1 + 1 = 1$ , plus a carry of 1 to next higher column.

# Binary Subtraction

Rule for binary subtraction is as follows:

$$0 - 0 = 0$$

$0 - 1 = 1$  with a borrow from the next column

$$1 - 0 = 1$$

$$1 - 1 = 0$$

# Binary Subtraction (Example)

## Example

Subtract  $01110_2$  from  $10101_2$

## Solution

$$\begin{array}{r} 12 \\ 0202 \\ 10101 \\ -01110 \\ \hline 00111 \end{array}$$

Note: Go through explanation given in the book

# Complement of a Number

$$C = B^n - 1 - N$$

Number of digits  
in the number

$\downarrow$

$C$       =       $B^n$       -      1      -       $N$

$\uparrow$

Complement  
of the number      Base of the  
number      The number

# Complement of a Number (Example 1)

## Example

Find the complement of  $37_{10}$

## Solution

Since the number has 2 digits and the value of base is 10,

$$(\text{Base})^n - 1 = 10^2 - 1 = 99$$

$$\text{Now } 99 - 37 = 62$$

Hence, complement of  $37_{10} = 62_{10}$

# Complement of a Number (Example 2)

## Example

Find the complement of  $6_8$

## Solution

Since the number has 1 digit and the value of base is 8,

$$(\text{Base})^n - 1 = 8^1 - 1 = 7_{10} = 7_8$$

$$\text{Now } 7_8 - 6_8 = 1_8$$

Hence, complement of  $6_8 = 1_8$

# Complement of a Binary Number

Complement of a binary number can be obtained by transforming all its 0's to 1's and all its 1's to 0's

## Example

Complement of      1 0 1 1 0 1 0      is  
                        ↓ ↓ ↓ ↓ ↓ ↓ ↓  
                        0 1 0 0 1 0 1

Note: Verify by conventional complement

# Complementary Method of Subtraction

Involves following 3 steps:

- Step 1: Find the complement of the number you are subtracting (subtrahend)
- Step 2: Add this to the number from which you are taking away (minuend)
- Step 3: If there is a carry of 1, add it to obtain the result; if there is no carry, recomplement the sum and attach a negative sign

Complementary subtraction is an additive approach of subtraction

# Complementary Subtraction (Example 1)

## Example:

Subtract  $56_{10}$  from  $92_{10}$  using complementary method.

## Solution

Step 1: Complement of  $56_{10}$   
 $= 10^2 - 1 - 56 = 99 - 56 = 43_{10}$

Step 2:  $92 + 43$  (complement of 56)  
 $= 135$  (note 1 as carry)

Step 3:  $35 + 1$  (add 1 carry to sum)

Result = 36

The result may be verified using the method of normal subtraction:

$$92 - 56 = 36$$

# Complementary Subtraction (Example 2)

## Example

Subtract  $35_{10}$  from  $18_{10}$  using complementary method.

## Solution

Step 1: Complement of  $35_{10}$   
 $= 10^2 - 1 - 35$   
 $= 99 - 35$   
 $= 64_{10}$

Step 2:

18	
+ 64 (complement of 35)	
<hr/>	
82	

Step 3: Since there is no carry,  
 re-complement the sum and  
 attach a negative sign to  
 obtain the result.

Result  $= -(99 - 82)$   
 $= -17$

The result may be verified using normal subtraction:

$$18 - 35 = -17$$

# Binary Subtraction Using Complementary Method (Example 1)

## Example

Subtract  $0111000_2$  ( $56_{10}$ ) from  $1011100_2$  ( $92_{10}$ ) using complementary method.

## Solution

$$\begin{array}{r} 1011100 \\ + 1000111 \quad (\text{complement of } 0111000) \\ \hline \end{array}$$

$$\begin{array}{r} 10100011 \\ \hline \end{array}$$

→ 1 (add the carry of 1)

$$\begin{array}{r} 0100100 \\ \hline \end{array}$$

$$\text{Result} = 0100100_2 = 36_{10}$$

# Binary Subtraction Using Complementary Method (Example 2)

## Example

Subtract  $100011_2$  ( $35_{10}$ ) from  $010010_2$  ( $18_{10}$ ) using complementary method.

## Solution

$$\begin{array}{r} 010010 \\ +011100 \quad (\text{complement of } 100011) \\ \hline 101110 \end{array}$$

Since there is no carry, we have to complement the sum and attach a negative sign to it. Hence,

$$\begin{aligned} \text{Result} &= -010001_2 \quad (\text{complement of } 101110_2) \\ &= -17_{10} \end{aligned}$$

# Binary Multiplication

Table for binary multiplication is as follows:

$$0 \times 0 = 0$$

$$0 \times 1 = 0$$

$$1 \times 0 = 0$$

$$1 \times 1 = 1$$

# Binary Multiplication (Example 1)

## Example

Multiply the binary numbers 1010 and 1001

## Solution

1010	Multiplicand
x1001	Multiplier
1010	Partial Product
0000	Partial Product
0000	Partial Product
1010	Partial Product
1011010	Final Product

(Continued on next slide)

# Binary Multiplication (Example 2)

(Continued from previous slide..)

Whenever a 0 appears in the multiplier, a separate partial product consisting of a string of zeros need not be generated (only a shift will do). Hence,

$$\begin{array}{r} 1010 \\ \times 1001 \\ \hline 1010 \\ 1010 \text{SS } (\text{S = left shift}) \\ \hline 1011010 \end{array}$$

# Binary Division

Table for binary division is as follows:

$0 \div 0 = \text{Divide by zero error}$

$0 \div 1 = 0$

$1 \div 0 = \text{Divide by zero error}$

$1 \div 1 = 1$

As in the decimal number system (or in any other number system), division by zero is meaningless

The computer deals with this problem by raising an error condition called 'Divide by zero' error

# Rules for Binary Division

1. Start from the left of the dividend
2. Perform a series of subtractions in which the divisor is subtracted from the dividend
3. If subtraction is possible, put a 1 in the quotient and subtract the divisor from the corresponding digits of dividend
4. If subtraction is not possible (divisor greater than remainder), record a 0 in the quotient
5. Bring down the next digit to add to the remainder digits. Proceed as before in a manner similar to long division

# Binary Division (Example 1)

## Example

Divide  $100001_2$  by  $110_2$

**Solution** 0101 (Quotient)

$110$	$\overline{)100001}$	(Dividend)
$110$	$1 \leftarrow$	Divisor greater than 100, so put 0 in quotient
$1000$	$2 \leftarrow$	Add digit from dividend to group used above
$110$	$3 \leftarrow$	Subtraction possible, so put 1 in quotient
$100$	$4 \leftarrow$	Remainder from subtraction plus digit from dividend
$110$	$5 \leftarrow$	Divisor greater, so put 0 in quotient
$1001$	$6 \leftarrow$	Add digit from dividend to group
$110$	$7 \leftarrow$	Subtraction possible, so put 1 in quotient
$11$	Remainder	

# Additive Method of Multiplication and Division

Most computers use the additive method for performing multiplication and division operations because it simplifies the internal circuit design of computer systems

## Example

$$4 \times 8 = 8 + 8 + 8 + 8 = 32$$

# Rules for Additive Method of Division

- § Subtract the divisor repeatedly from the dividend until the result of subtraction becomes less than or equal to zero
- § If result of subtraction is zero, then:
  - § quotient = total number of times subtraction was performed
  - § remainder = 0
- § If result of subtraction is less than zero, then:
  - § quotient = total number of times subtraction was performed minus 1
  - § remainder = result of the subtraction previous to the last subtraction

# Additive Method of Division (Example)

## Example

Divide  $33_{10}$  by  $6_{10}$  using the method of addition

### Solution:

$$33 - 6 = 27$$

$$27 - 6 = 21$$

$$21 - 6 = 15$$

$$15 - 6 = 9$$

$$9 - 6 = 3$$

$$3 - 6 = -3$$

Since the result of the last subtraction is less than zero,

Quotient = 6 - 1 (ignore last subtraction) = 5

Total subtractions = 6

Reminder = 3 (result of previous subtraction)

# Key Words/Phrases

- § Additive method of division
- § Additive method of multiplication
- § Additive method of subtraction
- § Binary addition
- § Binary arithmetic
- § Binary division
- § Binary multiplication
- § Binary subtraction
- § Complement
- § Complementary subtraction
- § Computer arithmetic

# Chapter 06

## Boolean Algebra and Logic Circuits

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

In this chapter you will learn about:

- § Boolean algebra
- § Fundamental concepts and basic laws of Boolean algebra
- § Boolean function and minimization
- § Logic gates
- § Logic circuits and Boolean expressions
- § Combinational circuits and design

# Boolean Algebra

- § An algebra that deals with binary number system
- § George Boole (1815-1864), an English mathematician, developed it for:
  - § Simplifying representation
  - § Manipulation of propositional logic
- § In 1938, Claude E. Shannon proposed using Boolean algebra in design of relay switching circuits
- § Provides economical and straightforward approach
- § Used extensively in designing electronic circuits used in computers

# Fundamental Concepts of Boolean Algebra

## § Use of Binary Digit

§ Boolean equations can have either of two possible values, 0 and 1

## § Logical Addition

§ Symbol '+', also known as 'OR' operator, used for logical addition. Follows law of binary addition

## § Logical Multiplication

§ Symbol '.', also known as 'AND' operator, used for logical multiplication. Follows law of binary multiplication

## § Complementation

§ Symbol '−', also known as 'NOT' operator, used for complementation. Follows law of binary compliment

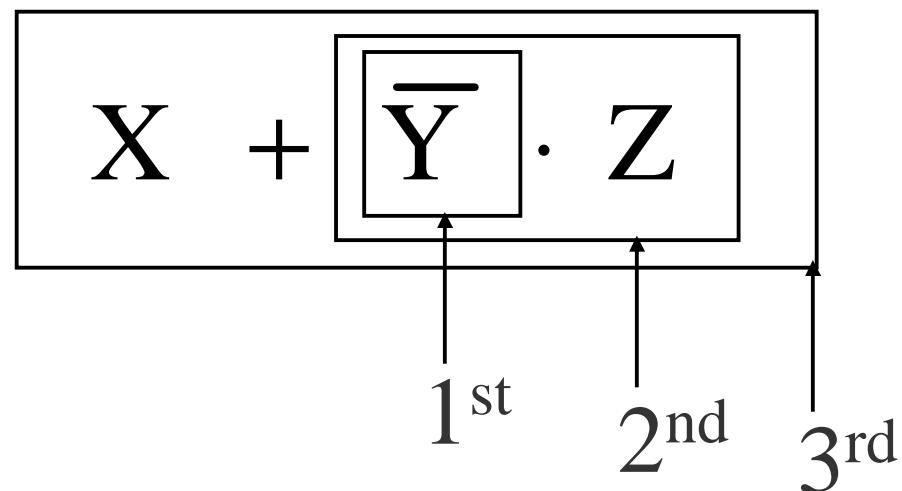
# Operator Precedence

- § Each operator has a precedence level
- § Higher the operator's precedence level, earlier it is evaluated
- § Expression is scanned from left to right
- § First, expressions enclosed within parentheses are evaluated
- § Then, all complement (NOT) operations are performed
- § Then, all ‘.’ (AND) operations are performed
- § Finally, all ‘+’ (OR) operations are performed

(Continued on next slide)

# Operator Precedence

(Continued from previous slide..)



# Postulates of Boolean Algebra

## ***Postulate 1:***

- (a)  $A = 0$ , if and only if, A is not equal to 1
- (b)  $A = 1$ , if and only if, A is not equal to 0

## ***Postulate 2:***

- (a)  $x + 0 = x$
- (b)  $x \cdot 1 = x$

## ***Postulate 3: Commutative Law***

- (a)  $x + y = y + x$
- (b)  $x \cdot y = y \cdot x$

*(Continued on next slide)*

# Postulates of Boolean Algebra

(Continued from previous slide..)

## Postulate 4: Associative Law

- (a)  $x + (y + z) = (x + y) + z$
- (b)  $x \cdot (y \cdot z) = (x \cdot y) \cdot z$

## Postulate 5: Distributive Law

- (a)  $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
- (b)  $x + (y \cdot z) = (x + y) \cdot (x + z)$

## Postulate 6:

- (a)  $x + \bar{x} = 1$
- (b)  $x \cdot \bar{x} = 0$

# The Principle of Duality

There is a precise duality between the operators . (AND) and + (OR), and the digits 0 and 1.

For example, in the table below, the second row is obtained from the first row and vice versa simply by interchanging '+' with '.' and '0' with '1'

	Column 1	Column 2	Column 3
Row 1	$1 + 1 = 1$	$1 + 0 = 0 + 1 = 1$	$0 + 0 = 0$
Row 2	$0 \cdot 0 = 0$	$0 \cdot 1 = 1 \cdot 0 = 0$	$1 \cdot 1 = 1$

Therefore, if a particular theorem is proved, its dual theorem automatically holds and need not be proved separately

# Some Important Theorems of Boolean Algebra

Sr. No.	Theorems/ Identities	Dual Theorems/ Identities	Name (if any)
1	$x + x = x$	$x \cdot x = x$	Idempotent Law
2	$x + 1 = 1$	$x \cdot 0 = 0$	
3	$x + x \cdot y = x$	$x \cdot x + y = x$	Absorption Law
4	$\overline{\overline{x}} = x$		Involution Law
5	$x \cdot \overline{x} + y = x \cdot y$	$x + \overline{x} \cdot y = x + y$	
6	$\overline{x+y} = \overline{x} \cdot \overline{y}$	$\overline{x \cdot y} = \overline{x} \cdot \overline{y} +$	De Morgan's Law

# Methods of Proving Theorems

The theorems of Boolean algebra may be proved by using one of the following methods:

1. By using postulates to show that L.H.S. = R.H.S
2. By *Perfect Induction* or *Exhaustive Enumeration* method where all possible combinations of variables involved in L.H.S. and R.H.S. are checked to yield identical results
3. By the *Principle of Duality* where the dual of an already proved theorem is derived from the proof of its corresponding pair

# Proving a Theorem by Using Postulates (Example)

**Theorem:**

$$x + x \cdot y = x$$

**Proof:**

L.H.S.

$$\begin{aligned} &= x + x \cdot y \\ &= x \cdot 1 + x \cdot y && \text{by postulate 2(b)} \\ &= x \cdot (1 + y) && \text{by postulate 5(a)} \\ &= x \cdot (y + 1) && \text{by postulate 3(a)} \\ &= x \cdot 1 && \text{by theorem 2(a)} \\ &= x && \text{by postulate 2(b)} \\ &= \text{R.H.S.} \end{aligned}$$

# Proving a Theorem by Perfect Induction (Example)

**Theorem:**

$$x + x \cdot y = x$$

————— = —————

x	y	$x \cdot y$	$x + x \cdot y$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

# Proving a Theorem by the Principle of Duality (Example)

*Theorem:*

$$x + x = x$$

*Proof:*

L.H.S.

$$\begin{aligned} &= x + x \\ &= (x + x) \cdot 1 && \text{by postulate 2(b)} \\ &= (x + x) \cdot (x + \bar{x}) && \text{by postulate 6(a)} \\ &= x + x \cdot \bar{x} && \text{by postulate 5(b)} \\ &= x + 0 && \text{by postulate 6(b)} \\ &= x && \text{by postulate 2(a)} \\ &= \text{R.H.S.} \end{aligned}$$

(Continued on next slide)

# Proving a Theorem by the Principle of Duality (Example)

(Continued from previous slide..)

## Dual Theorem:

$$x \cdot x = x$$

## Proof:

L.H.S.

$$= x \cdot x$$

$$= x \cdot x + 0$$

$$= x \cdot x + x \cdot \bar{x}$$

$$= x \cdot (x + \bar{x})$$

$$= x \cdot 1$$

$$= x$$

= R.H.S.

by postulate 2(a)

by postulate 6(b)

by postulate 5(a)

by postulate 6(a)

by postulate 2(b)

Notice that each step of the proof of the dual theorem is derived from the proof of its corresponding pair in the original theorem

# Boolean Functions

§ A Boolean function is an expression formed with:

- § Binary variables
- § Operators (OR, AND, and NOT)
- § Parentheses, and equal sign

§ The value of a Boolean function can be either 0 or 1

§ A Boolean function may be represented as:

- § An algebraic expression, or
- § A truth table

# Representation as an Algebraic Expression

$$W = X + \bar{Y} \cdot Z$$

- § Variable W is a function of X, Y, and Z, can also be written as  $W = f(X, Y, Z)$
- § The RHS of the equation is called an *expression*
- § The symbols X, Y, Z are the *literals* of the function
- § For a given Boolean function, there may be more than one algebraic expressions

# Representation as a Truth Table

X	Y	Z	W
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(Continued on next slide)

# Representation as a Truth Table

(Continued from previous slide..)

- § The number of rows in the table is equal to  $2^n$ , where  $n$  is the number of literals in the function
- § The combinations of 0s and 1s for rows of this table are obtained from the binary numbers by counting from 0 to  $2^n - 1$

# Minimization of Boolean Functions

- § Minimization of Boolean functions deals with
  - § Reduction in number of literals
  - § Reduction in number of terms
- § Minimization is achieved through manipulating expression to obtain equal and simpler expression(s) (having fewer literals and/or terms)

(Continued on next slide)

# Minimization of Boolean Functions

(Continued from previous slide..)

$$F_1 = \overline{x} \cdot \overline{y} \cdot z + \overline{x} \cdot y \cdot z + x \cdot \overline{y}$$

$F_1$  has 3 literals (x, y, z) and 3 terms

$$F_2 = x \cdot \overline{y} + \overline{x} \cdot z$$

$F_2$  has 3 literals (x, y, z) and 2 terms

$F_2$  can be realized with fewer electronic components, resulting in a cheaper circuit

(Continued on next slide)

# Minimization of Boolean Functions

(Continued from previous slide..)

x	y	z	F <sub>1</sub>	F <sub>2</sub>
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	0	0

Both F<sub>1</sub> and F<sub>2</sub> produce the same result

# Try out some Boolean Function Minimization

(a)  $x + \overline{x} \cdot y$

(b)  $x \cdot (\overline{x} + y)$

(c)  $\overline{x} \cdot \overline{y} \cdot z + \overline{x} \cdot y \cdot z + x \cdot \overline{y}$

(d)  $x \cdot y + \overline{x} \cdot z + y \cdot z$

(e)  $(x + y) \cdot (\overline{x} + z) \cdot (y + z)$

# Complement of a Boolean Function

- § The complement of a Boolean function is obtained by interchanging:
  - § Operators OR and AND
  - § Complementing each literal
- § This is based on *De Morgan's theorems*, whose general form is:

$$\overline{A_1 + A_2 + A_3 + \dots + A_n} = \bar{A}_1 \cdot \bar{A}_2 \cdot \bar{A}_3 \cdot \dots \cdot \bar{A}_n$$
$$\overline{A_1 \cdot A_2 \cdot A_3 \cdot \dots \cdot A_n} = \bar{A}_1 + \bar{A}_2 + \bar{A}_3 + \dots + \bar{A}_n$$

# Complementing a Boolean Function (Example)

$$F_1 = \bar{x} \cdot y \cdot \bar{z} + \bar{x} \cdot \bar{y} \cdot z$$

To obtain  $\bar{F}_1$ , we first interchange the OR and the AND operators giving

$$(\bar{x} + y + \bar{z}) \cdot (\bar{x} + \bar{y} + z)$$

Now we complement each literal giving

$$\bar{F}_1 = (x + \bar{y} + z) \cdot (x + y + \bar{z})$$

# Canonical Forms of Boolean Functions

**Minterms** :  $n$  variables forming an AND term, with each variable being primed or unprimed, provide  $2^n$  possible combinations called *minterms* or *standard products*

**Maxterms** :  $n$  variables forming an OR term, with each variable being primed or unprimed, provide  $2^n$  possible combinations called *maxterms* or *standard sums*

# Minterms and Maxterms for three Variables

Variables			Minterms		Maxterms	
x	y	z	Term	Designation	Term	Designation
0	0	0	$\overline{x} \cdot \overline{y} \cdot \overline{z}$	$m_0$	$x + y + z$	$M_0$
0	0	1	$\overline{x} \cdot \overline{y} \cdot z$	$m_1$	$x + y + \overline{z}$	$M_1$
0	1	0	$\overline{x} \cdot y \cdot \overline{z}$	$m_2$	$x + \overline{y} + z$	$M_2$
0	1	1	$\overline{x} \cdot y \cdot z$	$m_3$	$x + \overline{y} + \overline{z}$	$M_3$
1	0	0	$x \cdot \overline{y} \cdot \overline{z}$	$m_4$	$\overline{x} + y + z$	$M_4$
1	0	1	$x \cdot \overline{y} \cdot z$	$m_5$	$\overline{x} + y + \overline{z}$	$M_5$
1	1	0	$x \cdot y \cdot \overline{z}$	$m_6$	$\overline{x} + \overline{y} + z$	$M_6$
1	1	1	$x \cdot y \cdot z$	$m_7$	$\overline{x} + \overline{y} + \overline{z}$	$M_7$

Note that each minterm is the complement of its corresponding maxterm and vice-versa

# Sum-of-Products (SOP) Expression

A sum-of-products (SOP) expression is a product term (minterm) or several product terms (minterms) logically added (ORed) together. Examples are:

$$x$$

$$x + y$$

$$x + y \cdot z$$

$$x \cdot y + z$$

$$x \cdot \bar{y} + \bar{x} \cdot y$$

$$\bar{x} \cdot \bar{y} + x \cdot \bar{y} \cdot z$$

## Steps to Express a Boolean Function in its Sum-of-Products Form

1. Construct a truth table for the given Boolean function
2. Form a minterm for each combination of the variables, which produces a 1 in the function
3. The desired expression is the sum (OR) of all the minterms obtained in Step 2

# Expressing a Function in its Sum-of-Products Form (Example)

x	y	z	$F_1$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

The following 3 combinations of the variables produce a 1:  
 001, 100, and 111

(Continued on next slide)

# Expressing a Function in its Sum-of-Products Form (Example)

(Continued from previous slide..)

§ Their corresponding minterms are:

$$\bar{x} \cdot \bar{y} \cdot z, \quad x \cdot \bar{y} \cdot \bar{z}, \quad \text{and} \quad x \cdot y \cdot z$$

§ Taking the OR of these minterms, we get

$$F_1 = \bar{x} \cdot \bar{y} \cdot z + x \cdot \bar{y} \cdot \bar{z} + x \cdot y \cdot z = m_1 + m_4 + m_7$$

$$F_1(x \cdot y \cdot z) = \sum(1, 4, 7)$$

# Product-of Sums (POS) Expression

A product-of-sums (POS) expression is a sum term (maxterm) or several sum terms (maxterms) logically multiplied (ANDed) together. Examples are:

$$x \quad (x + \bar{y}) \cdot (\bar{x} + y) \cdot (\bar{x} + \bar{y})$$

$$\bar{x} + y \quad (x + y) \cdot (\bar{x} + y + z)$$

$$(\bar{x} + \bar{y}) \cdot z \quad (\bar{x} + y) \cdot (x + \bar{y})$$

## Steps to Express a Boolean Function in its Product-of-Sums Form

1. Construct a truth table for the given Boolean function
2. Form a maxterm for each combination of the variables, which produces a 0 in the function
3. The desired expression is the product (AND) of all the maxterms obtained in Step 2

# Expressing a Function in its Product-of-Sums Form

x	y	z	$F_1$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

§ The following 5 combinations of variables produce a 0:

000,      010,      011,      101,      and      110

(Continued on next slide)

# Expressing a Function in its Product-of-Sums Form

(Continued from previous slide..)

§ Their corresponding maxterms are:

$$(x+y+z), (x+\bar{y}+z), (x+\bar{y}+\bar{z}),$$

$$(\bar{x}+y+\bar{z}) \text{ and } (\bar{x}+\bar{y}+z)$$

§ Taking the AND of these maxterms, we get:

$$F_1 = (x+y+z) \cdot (x+\bar{y}+z) \cdot (x+\bar{y}+\bar{z}) \cdot (\bar{x}+y+\bar{z}) \cdot$$

$$(\bar{x}+\bar{y}+z) = M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$$

$$F_1(x,y,z) = \prod(0,2,3,5,6)$$

# Conversion Between Canonical Forms (Sum-of-Products and Product-of-Sums)

To convert from one canonical form to another, interchange the symbol and list those numbers missing from the original form.

## Example:

$$F(x,y,z) = \prod(0,2,4,5) = \sum(1,3,6,7)$$

$$F(x,y,z) = \prod(1,4,7) = \sum(0,2,3,5,6)$$

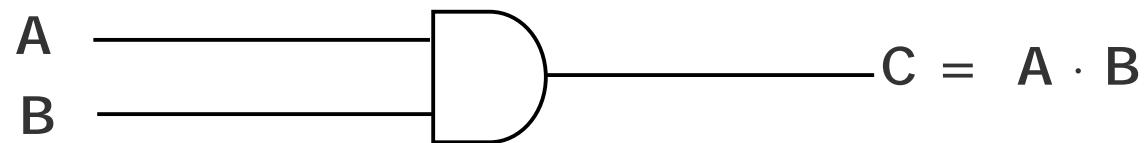
# Logic Gates

- § Logic gates are electronic circuits that operate on one or more input signals to produce standard output signal
- § Are the building blocks of all the circuits in a computer
- § Some of the most basic and useful logic gates are AND, OR, NOT, NAND and NOR gates

# AND Gate

- § Physical realization of logical multiplication (AND) operation
- § Generates an output signal of 1 only if all input signals are also 1

# AND Gate (Block Diagram Symbol and Truth Table)

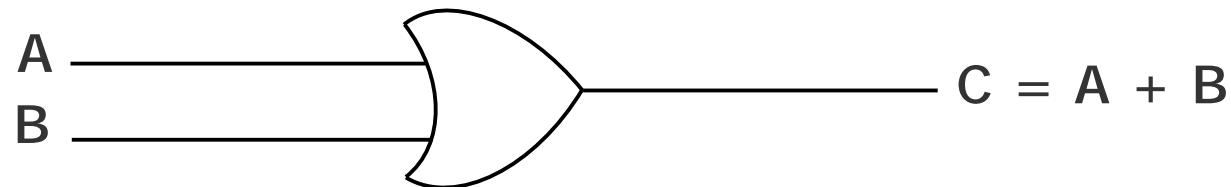


Inputs		Output
A	B	$C = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

# OR Gate

- § Physical realization of logical addition (OR) operation
- § Generates an output signal of 1 if at least one of the input signals is also 1

# OR Gate (Block Diagram Symbol and Truth Table)

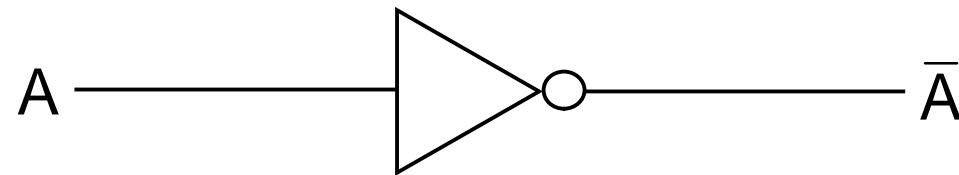


Inputs		Output
A	B	$C = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

# NOT Gate

- § Physical realization of complementation operation
- § Generates an output signal, which is the reverse of the input signal

# NOT Gate (Block Diagram Symbol and Truth Table)

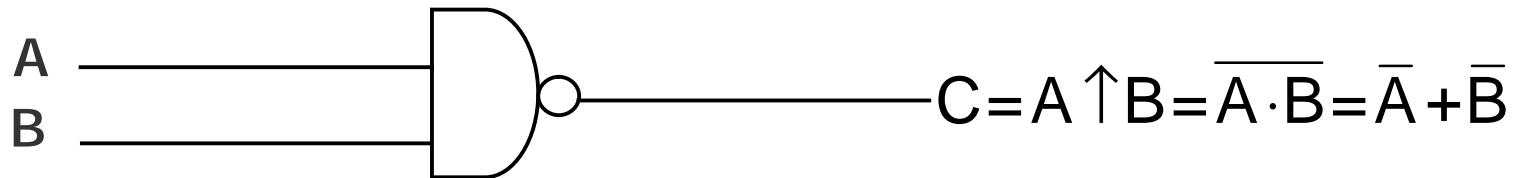


Input	Output
A	Ā
0	1
1	0

# NAND Gate

- § Complemented AND gate
- § Generates an output signal of:
  - § 1 if any one of the inputs is a 0
  - § 0 when all the inputs are 1

# NAND Gate (Block Diagram Symbol and Truth Table)

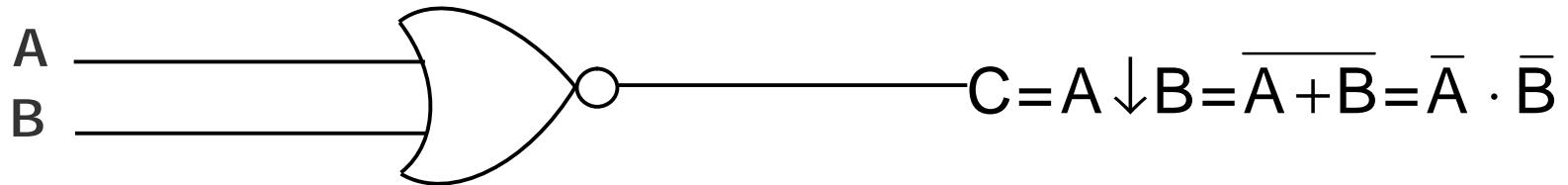


Inputs		Output
A	B	$C = \overline{A} + \overline{B}$
0	0	1
0	1	1
1	0	1
1	1	0

# NOR Gate

- § Complemented OR gate
- § Generates an output signal of:
  - § 1 only when all inputs are 0
  - § 0 if any one of inputs is a 1

# NOR Gate (Block Diagram Symbol and Truth Table)

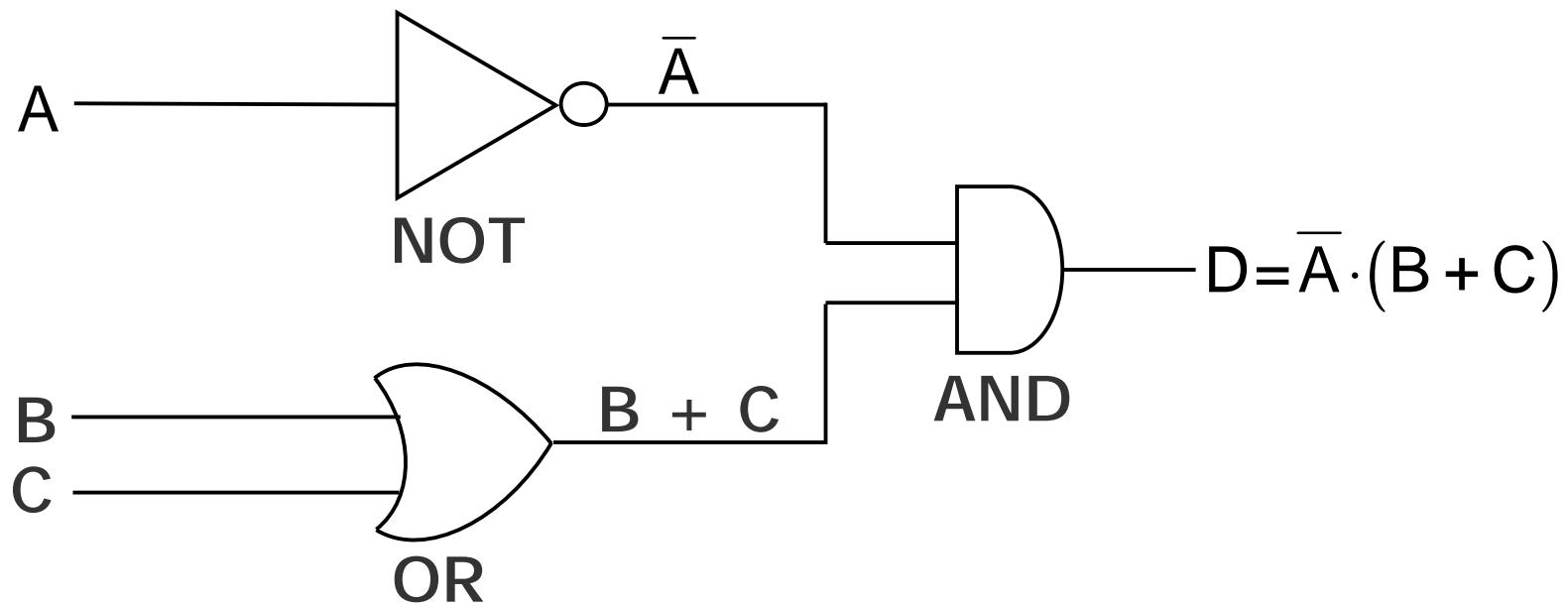


Inputs		Output
A	B	$C = \overline{A} \cdot \overline{B}$
0	0	1
0	1	0
1	0	0
1	1	0

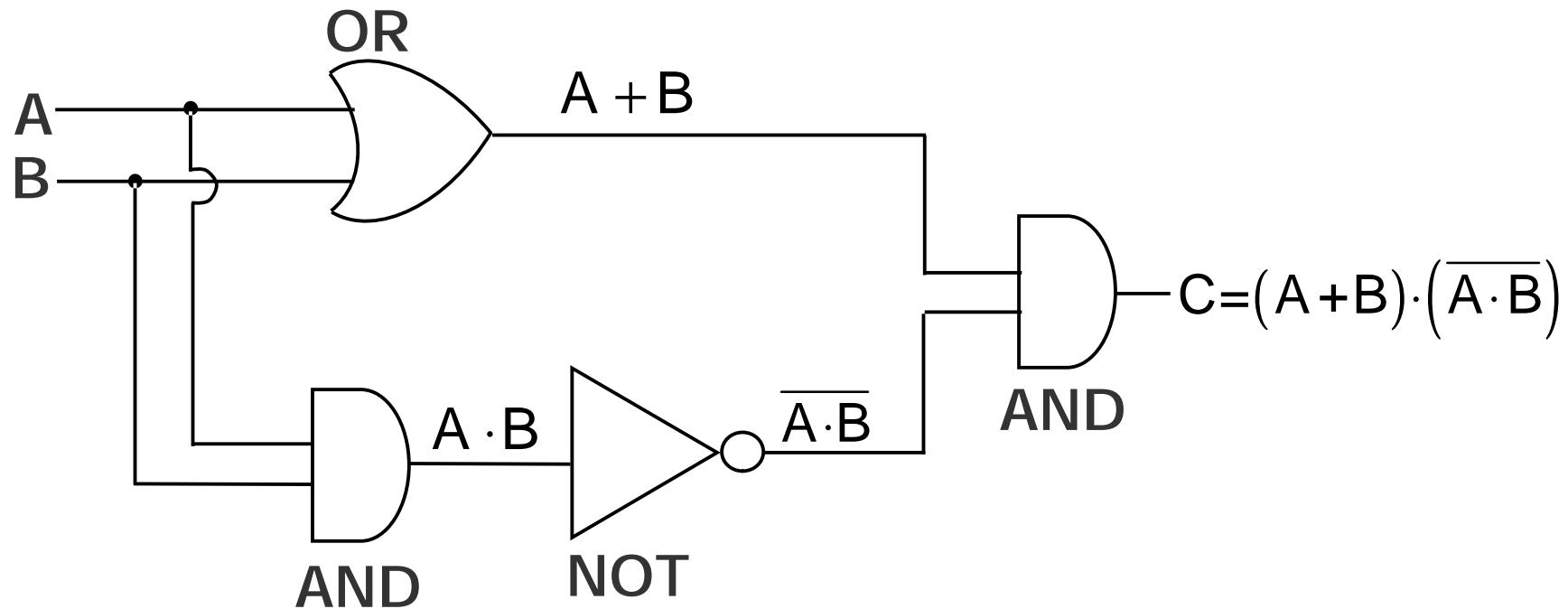
# Logic Circuits

- § When logic gates are interconnected to form a gating / logic network, it is known as a *combinational logic circuit*
- § The Boolean algebra expression for a given logic circuit can be derived by systematically progressing from input to output on the gates
- § The three logic gates (AND, OR, and NOT) are logically complete because any Boolean expression can be realized as a logic circuit using only these three gates

# Finding Boolean Expression of a Logic Circuit (Example 1)

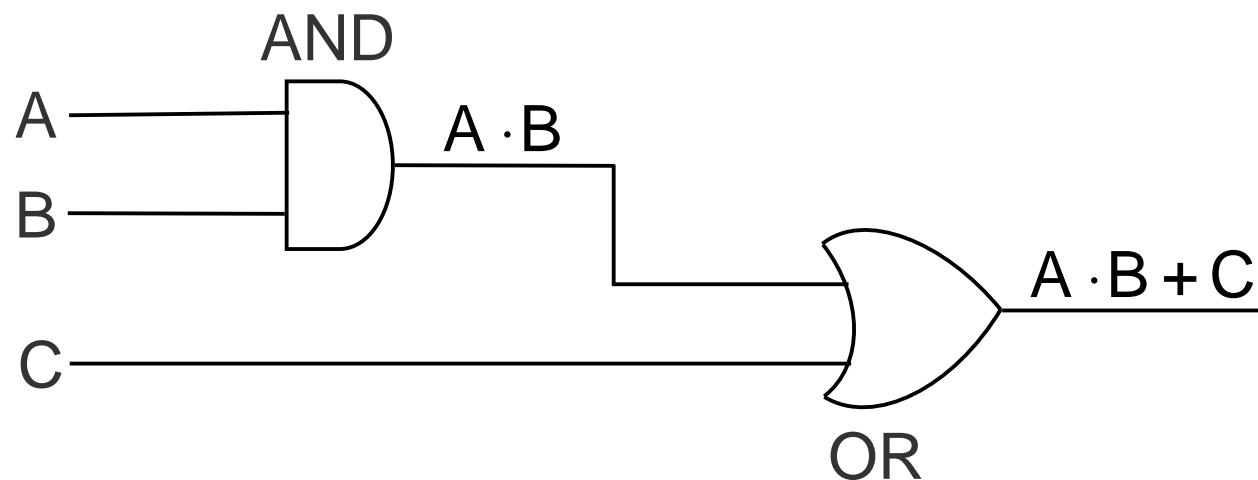


# Finding Boolean Expression of a Logic Circuit (Example 2)



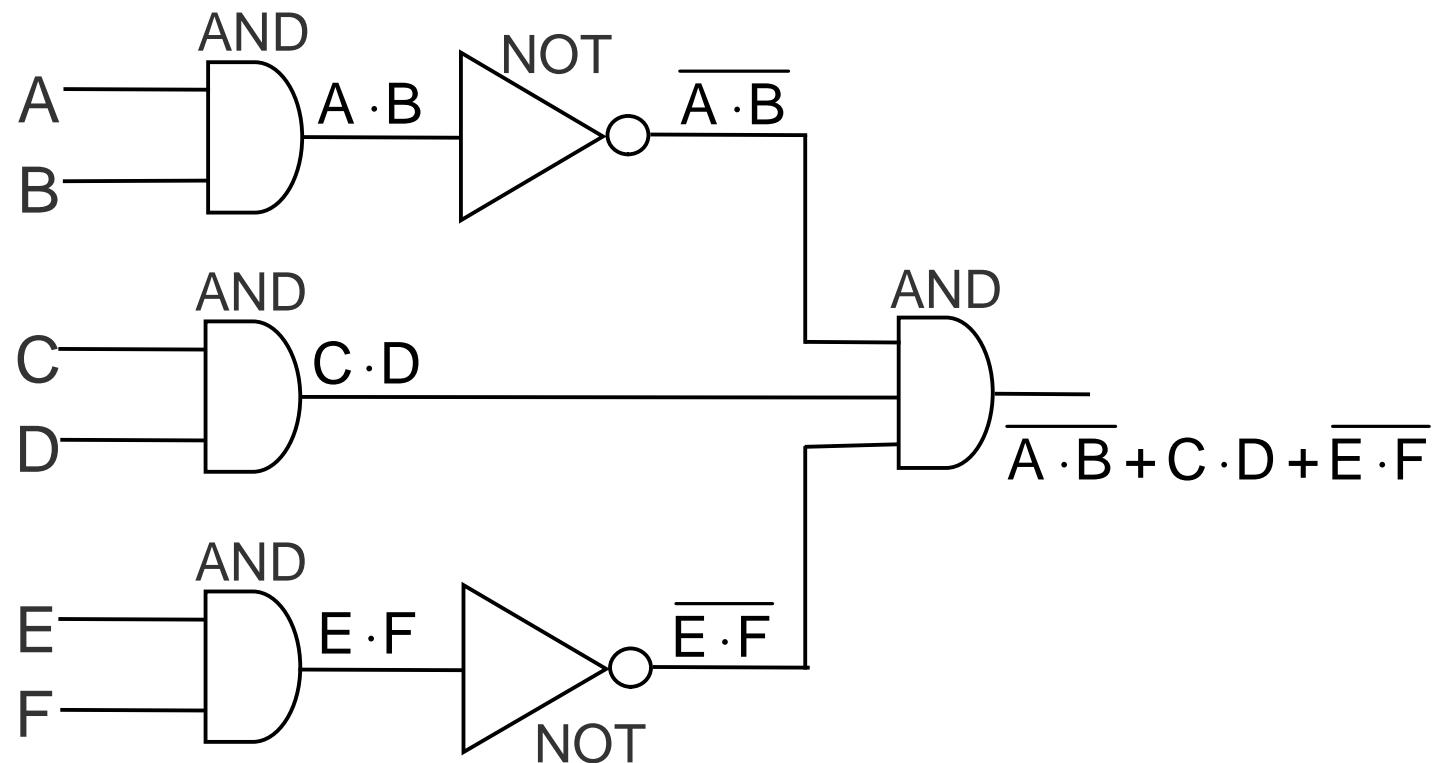
# Constructing a Logic Circuit from a Boolean Expression (Example 1)

Boolean Expression =  $A \cdot B + C$



# Constructing a Logic Circuit from a Boolean Expression (Example 2)

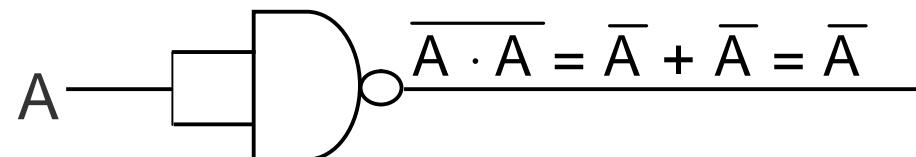
Boolean Expression =  $\overline{A \cdot B} + C \cdot D + \overline{E \cdot F}$



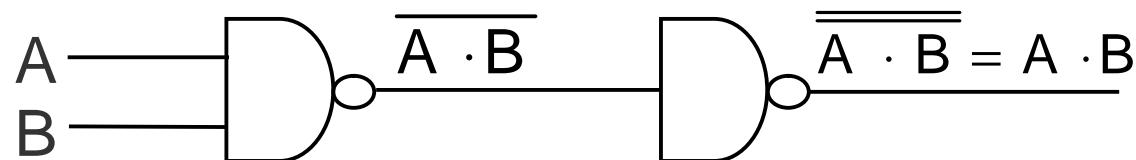
# Universal NAND Gate

- § NAND gate is an universal gate, it is alone sufficient to implement any Boolean expression
- § To understand this, consider:
  - § Basic logic gates (AND, OR, and NOT) are logically complete
  - § Sufficient to show that AND, OR, and NOT gates can be implemented with NAND gates

# Implementation of NOT, AND and OR Gates by NAND Gates



(a) NOT gate implementation.

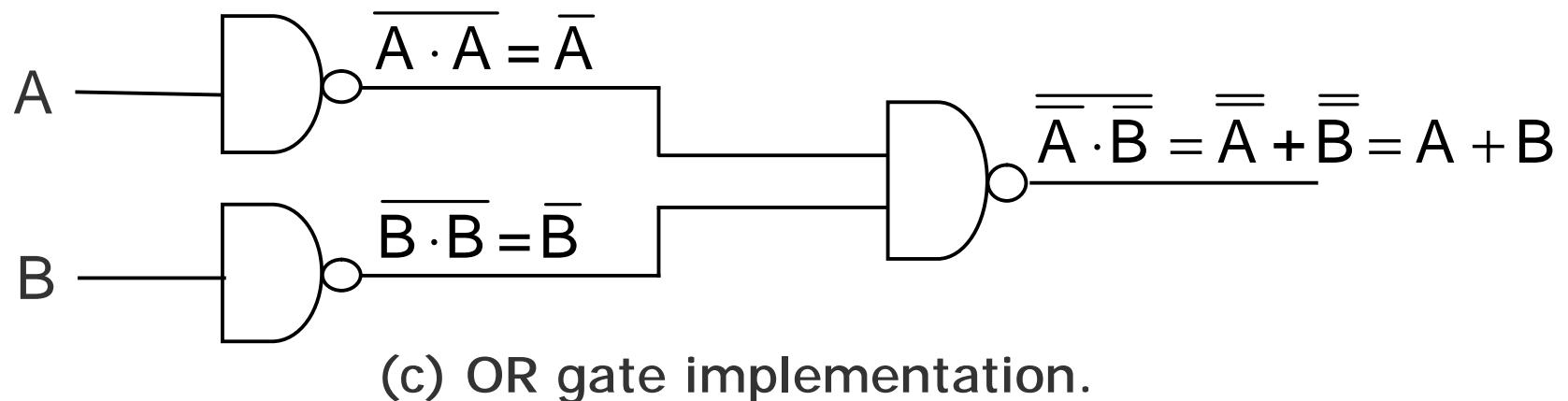


(b) AND gate implementation.

(Continued on next slide)

# Implementation of NOT, AND and OR Gates by NAND Gates

(Continued from previous slide..)

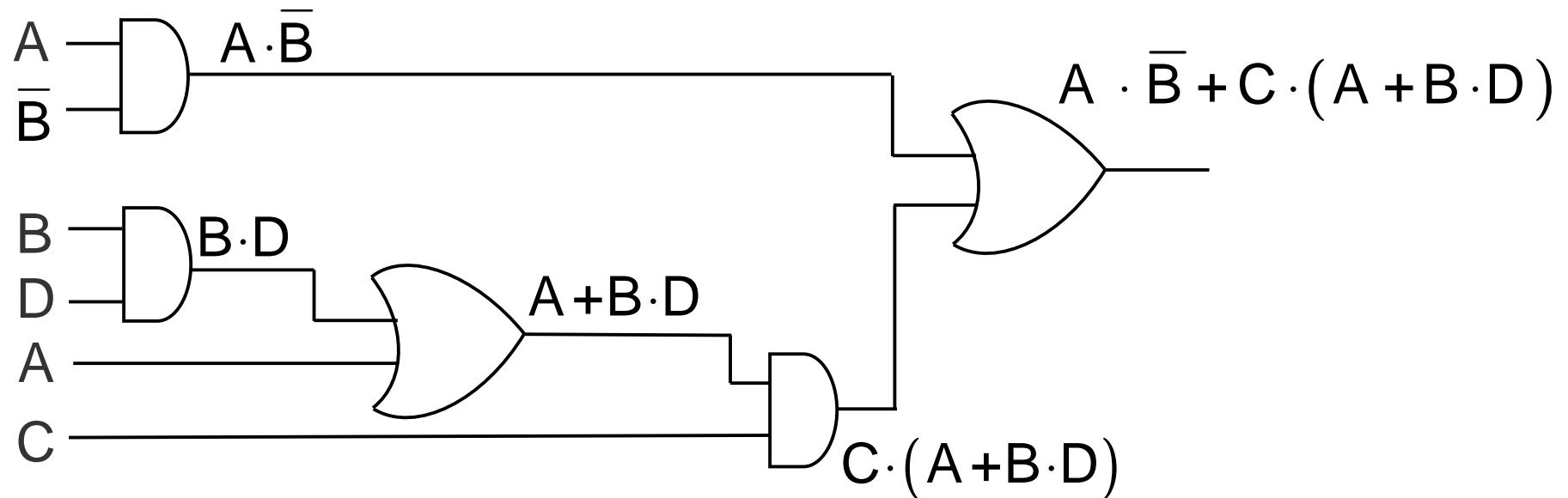


# Method of Implementing a Boolean Expression with Only NAND Gates

- Step 1: From the given algebraic expression, draw the logic diagram with AND, OR, and NOT gates. Assume that both the normal (A) and complement ( $\bar{A}$ ) inputs are available
- Step 2: Draw a second logic diagram with the equivalent NAND logic substituted for each AND, OR, and NOT gate
- Step 3: Remove all pairs of cascaded inverters from the diagram as double inversion does not perform any logical function. Also remove inverters connected to single external inputs and complement the corresponding input variable

# Implementing a Boolean Expression with Only NAND Gates (Example)

$$\text{Boolean Expression} = A \cdot \bar{B} + C \cdot (A + B \cdot D)$$

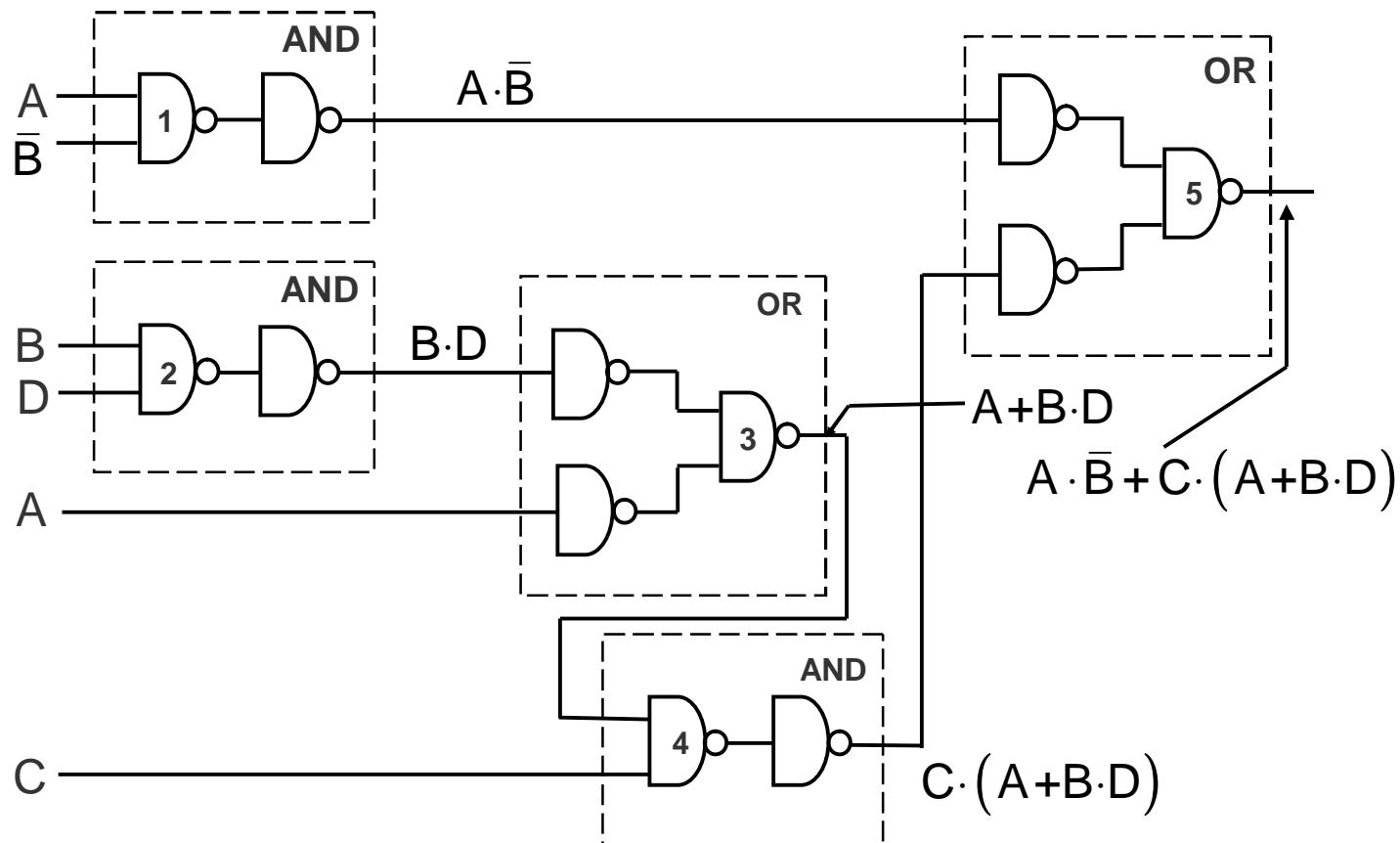


(a) Step 1: AND/OR implementation

(Continued on next slide)

# Implementing a Boolean Expression with Only NAND Gates (Example)

(Continued from previous slide..)

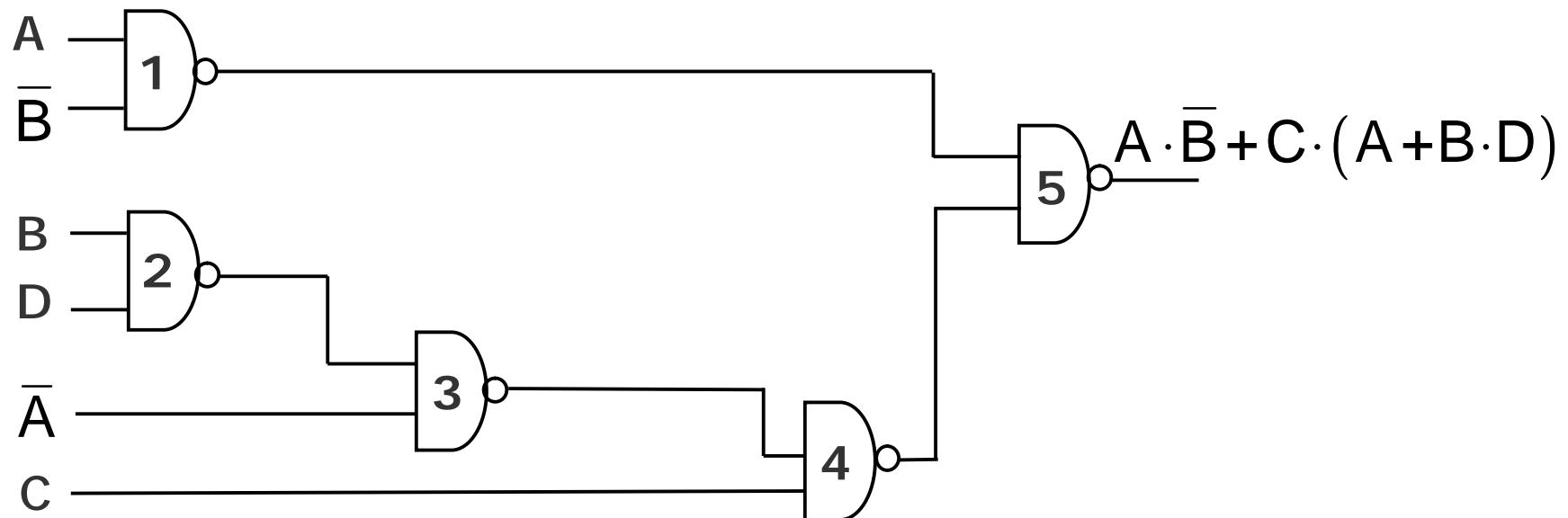


(b) Step 2: Substituting equivalent NAND functions

(Continued on next slide)

# Implementing a Boolean Expression with Only NAND Gates (Example)

(Continued from previous slide..)

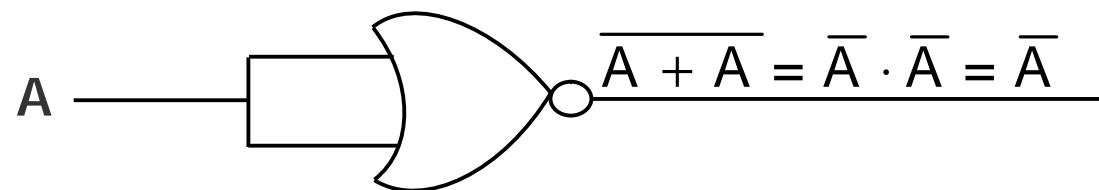


(c) Step 3: NAND implementation.

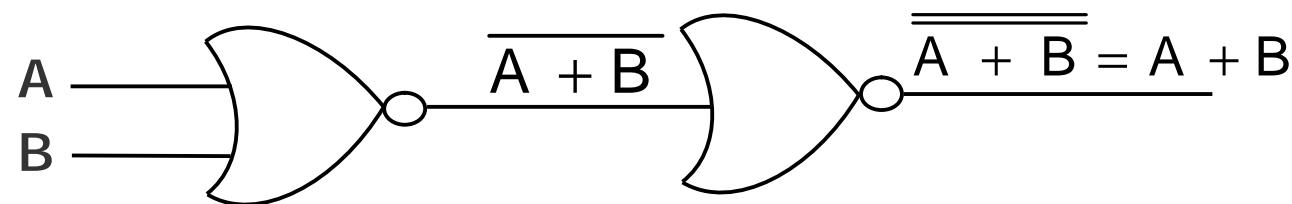
# Universal NOR Gate

- § NOR gate is an universal gate, it is alone sufficient to implement any Boolean expression
- § To understand this, consider:
  - § Basic logic gates (AND, OR, and NOT) are logically complete
  - § Sufficient to show that AND, OR, and NOT gates can be implemented with NOR gates

# Implementation of NOT, OR and AND Gates by NOR Gates



(a) NOT gate implementation.

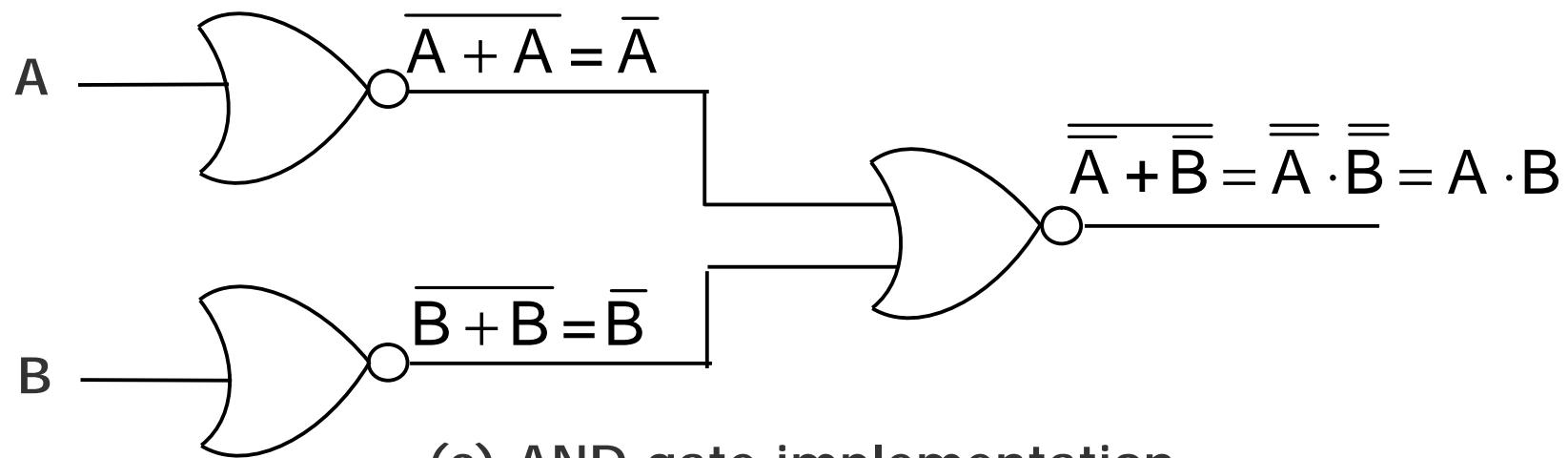


(b) OR gate implementation.

(Continued on next slide)

# Implementation of NOT, OR and AND Gates by NOR Gates

(Continued from previous slide..)



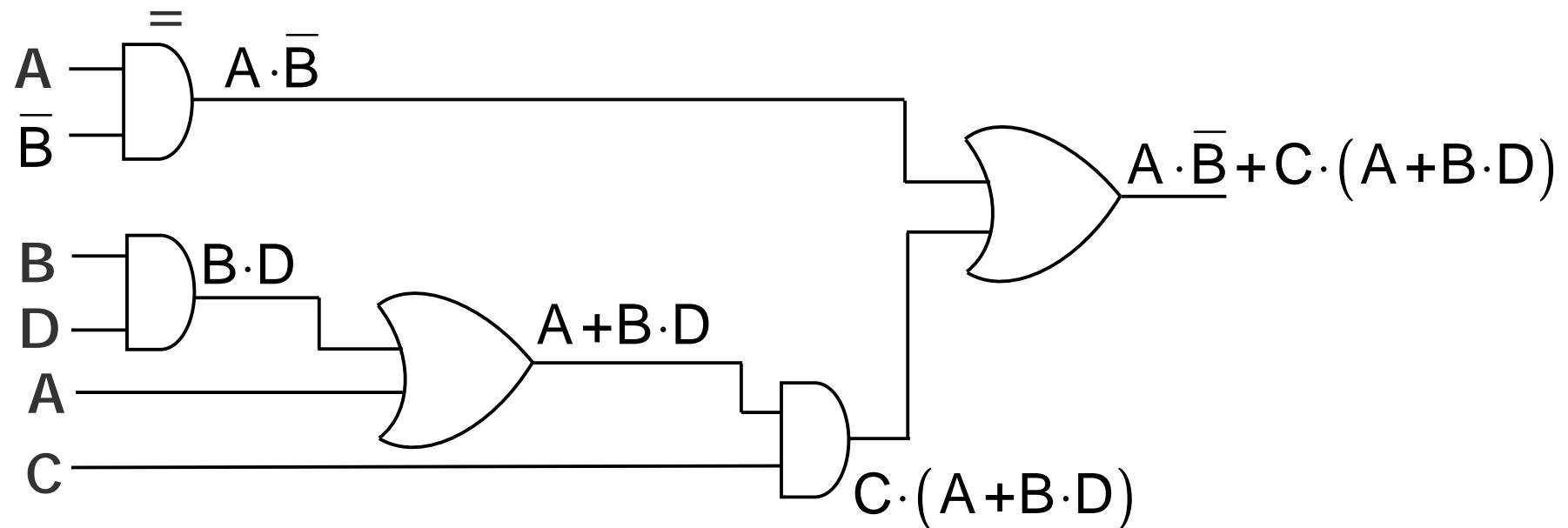
# Method of Implementing a Boolean Expression with Only NOR Gates

- Step 1: For the given algebraic expression, draw the logic diagram with AND, OR, and NOT gates. Assume that both the normal ( $A$ ) and complement ( $\bar{A}$ ) inputs are available
- Step 2: Draw a second logic diagram with equivalent NOR logic substituted for each AND, OR, and NOT gate
- Step 3: Remove all parts of cascaded inverters from the diagram as double inversion does not perform any logical function. Also remove inverters connected to single external inputs and complement the corresponding input variable

# Implementing a Boolean Expression with Only NOR Gates (Examples)

(Continued from previous slide..)

Boolean Expression  $A \cdot \bar{B} + C \cdot (A + B \cdot D)$

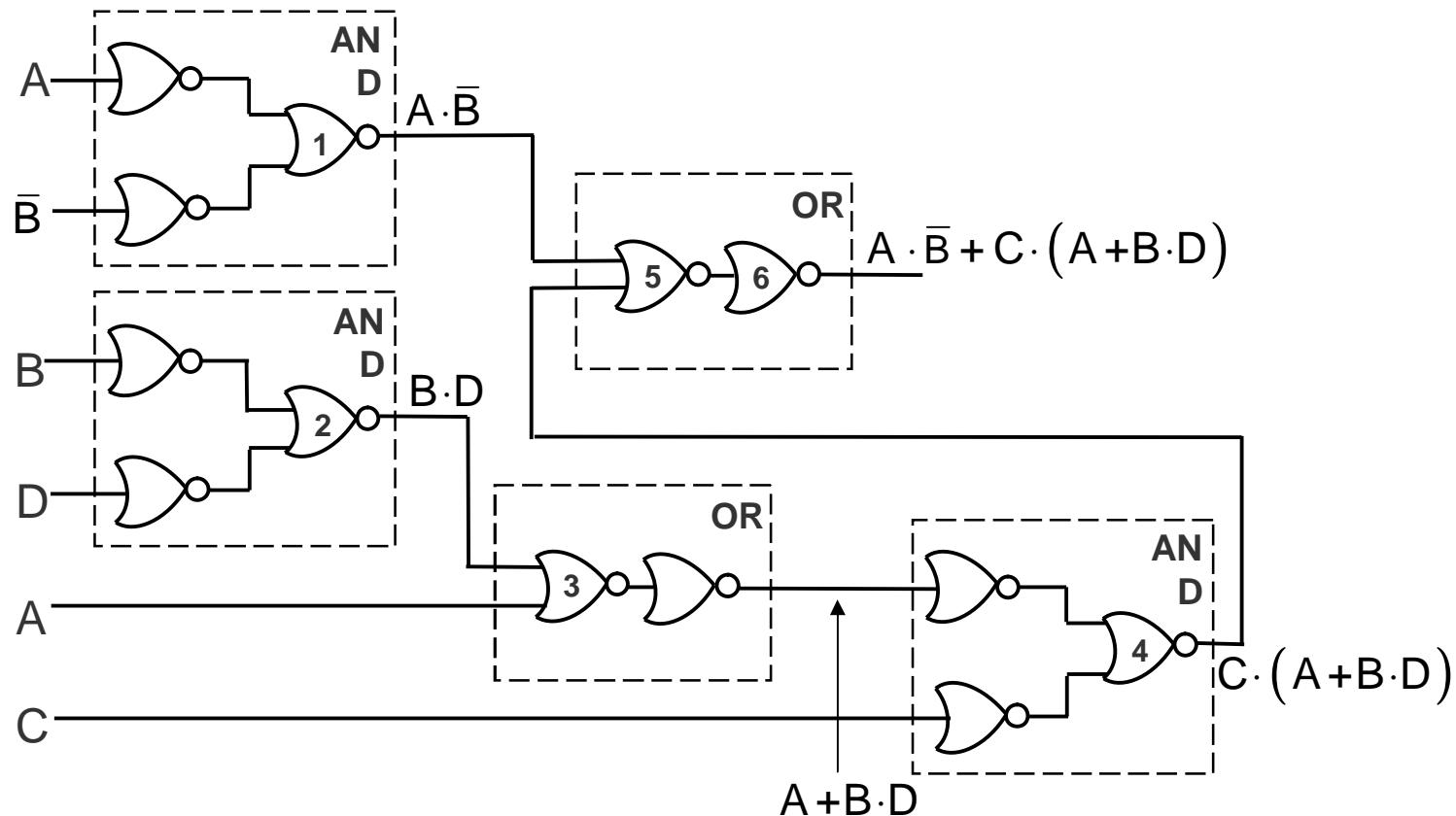


(a) Step 1: AND/OR implementation.

(Continued on next slide)

# Implementing a Boolean Expression with Only NOR Gates (Examples)

(Continued from previous slide..)

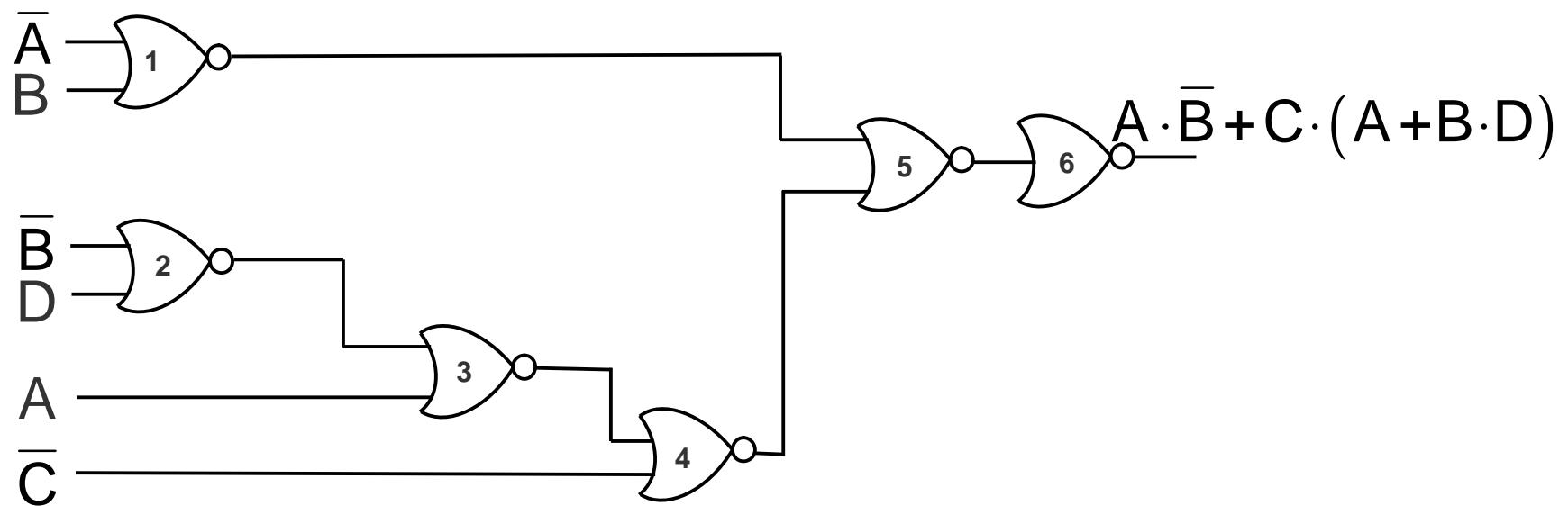


(b) Step 2: Substituting equivalent NOR functions.

(Continued on next slide)

# Implementing a Boolean Expression with Only NOR Gates (Examples)

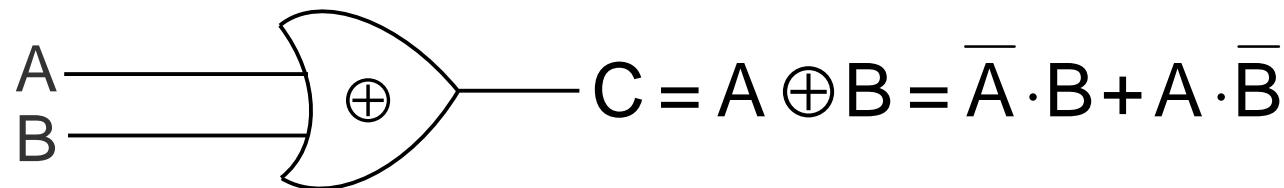
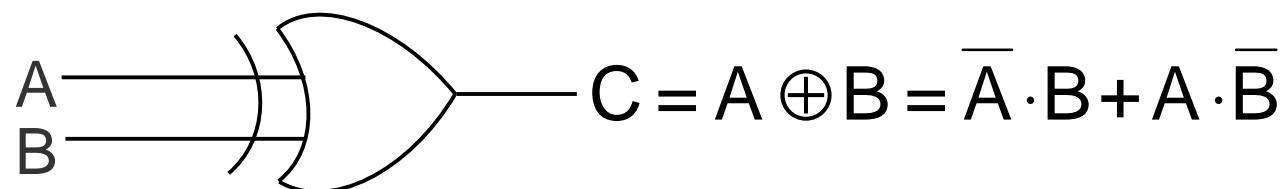
(Continued from previous slide..)



(c) Step 3: NOR implementation.

# Exclusive-OR Function

$$A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B$$



$$\text{Also, } (A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$$

(Continued on next slide)

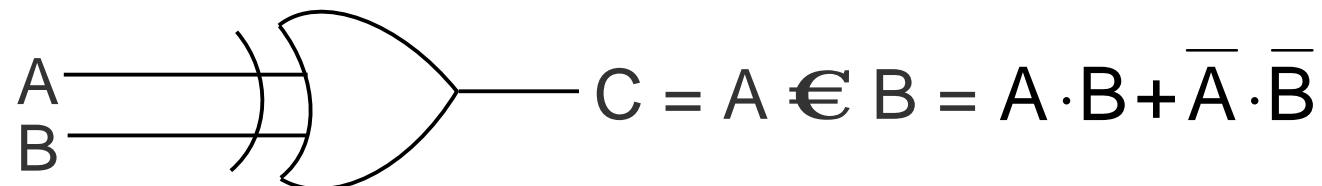
# Exclusive-OR Function (Truth Table)

(Continued from previous slide..)

Inputs		Output
A	B	$C = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

# Equivalence Function with Block Diagram Symbol

$$A \oplus B = A \cdot B + \bar{A} \cdot \bar{B}$$



$$\text{Also, } (A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$$

(Continued on next slide)

# Equivalence Function (Truth Table)

Inputs		Output
A	B	$C = A \Leftrightarrow B$
0	0	1
0	1	0
1	0	0
1	1	1

# Steps in Designing Combinational Circuits

1. State the given problem completely and exactly
2. Interpret the problem and determine the available input variables and required output variables
3. Assign a letter symbol to each input and output variables
4. Design the truth table that defines the required relations between inputs and outputs
5. Obtain the simplified Boolean function for each output
6. Draw the logic circuit diagram to implement the Boolean function

# Designing a Combinational Circuit

## Example 1 – Half-Adder Design

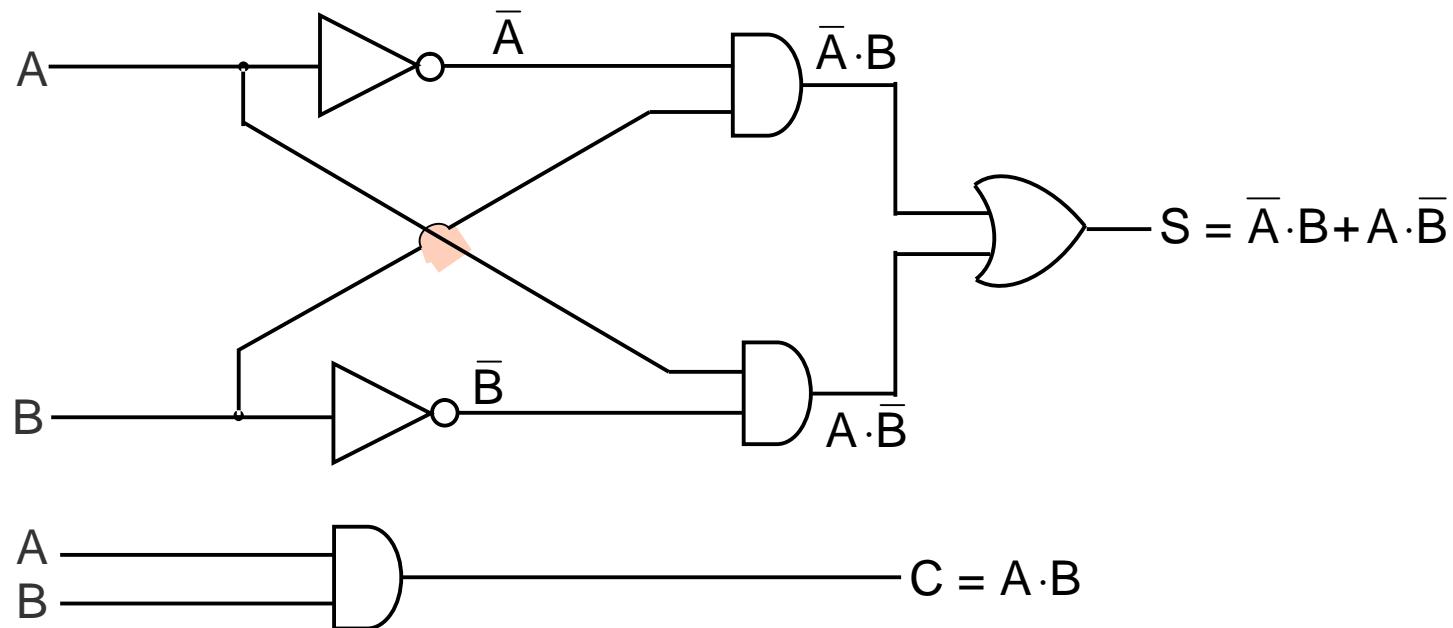
Inputs		Outputs	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$\left. \begin{array}{l} S = \overline{A} \cdot B + A \cdot \overline{B} \\ C = A \cdot B \end{array} \right\} \text{Boolean functions for the two outputs.}$$

# Designing a Combinational Circuit

## Example 1 – Half-Adder Design

(Continued from previous slide..)



Logic circuit diagram to implement the Boolean functions

# Designing a Combinational Circuit

## Example 2 – Full-Adder Design

Inputs			Outputs	
A	B	D	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Truth table for a full adder

(Continued on next slide)

# Designing a Combinational Circuit

## Example 2 – Full-Adder Design

(Continued from previous slide..)

**Boolean functions for the two outputs:**

$$S = \bar{A} \cdot \bar{B} \cdot D + \bar{A} \cdot B \cdot \bar{D} + A \cdot \bar{B} \cdot \bar{D} + A \cdot B \cdot D$$

$$C = \bar{A} \cdot B \cdot D + A \cdot \bar{B} \cdot D + A \cdot B \cdot \bar{D} + A \cdot B \cdot D$$

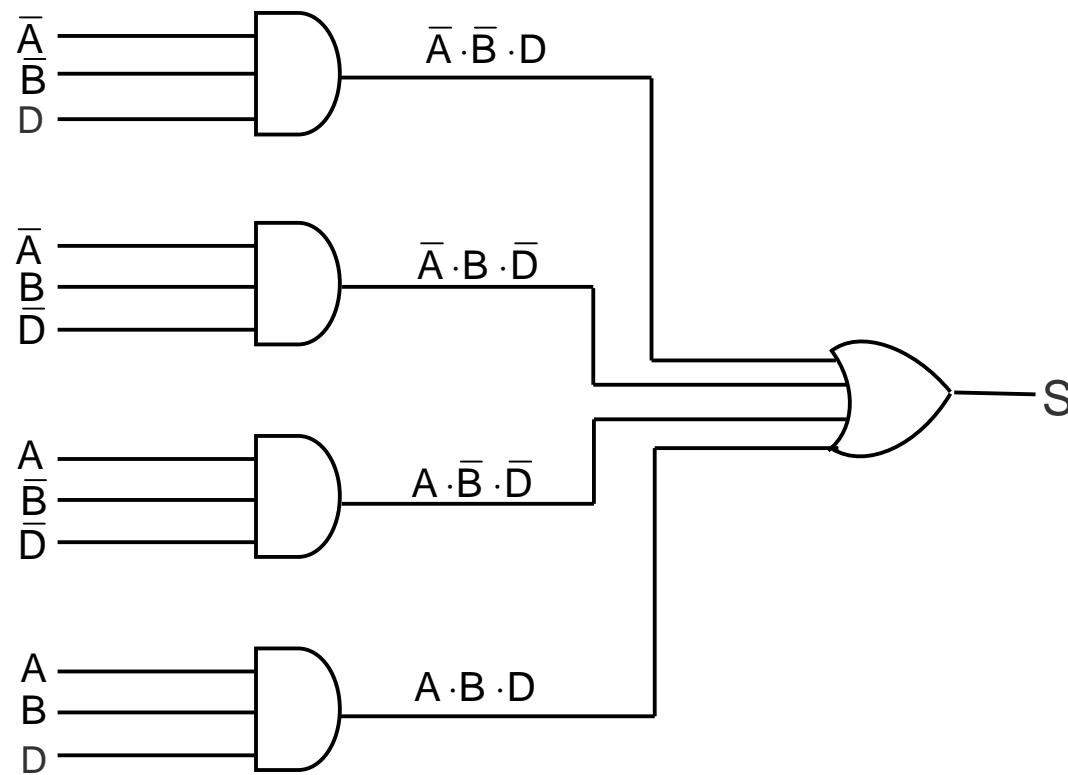
$$= A \cdot B + A \cdot D + B \cdot D \quad (\text{when simplified})$$

(Continued on next slide)

# Designing a Combinational Circuit

## Example 2 – Full-Adder Design

(Continued from previous slide..)



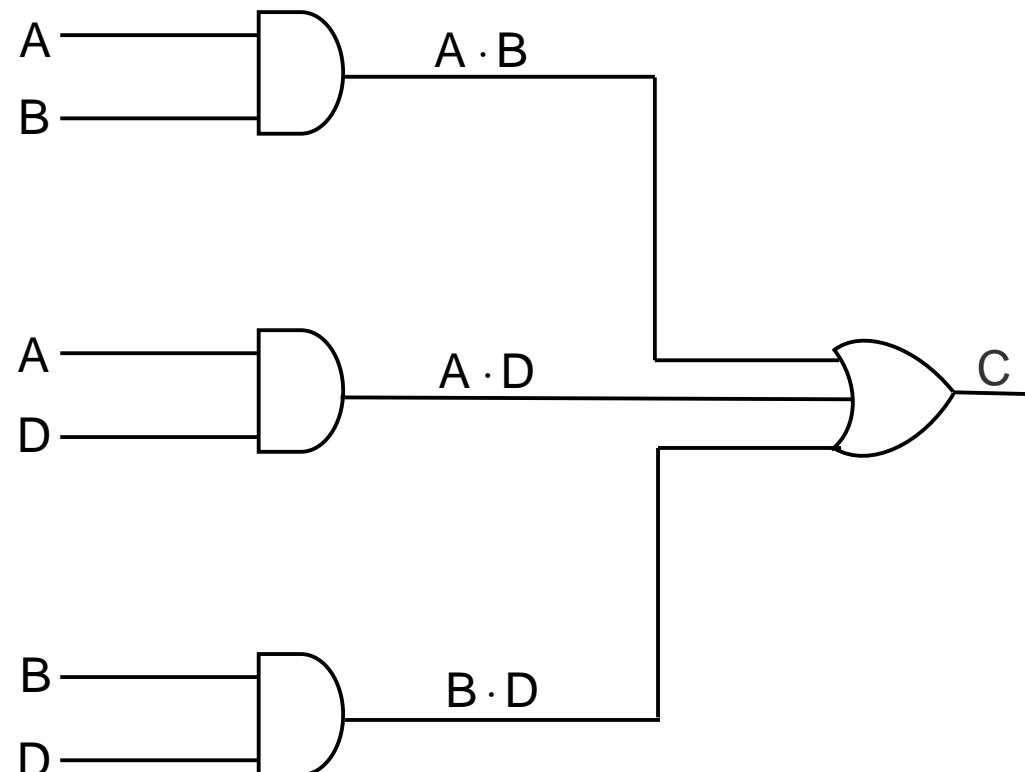
(a) Logic circuit diagram for sums

(Continued on next slide)

# Designing a Combinational Circuit

## Example 2 – Full-Adder Design

(Continued from previous slide..)



(b) Logic circuit diagram for carry

# Key Words/Phrases

- |   |                                     |                                 |
|---|-------------------------------------|---------------------------------|
| § Absorption law                        | § Equivalence function              | § NOT gate                      |
| § AND gate                              | § Exclusive-OR function             | § Operator precedence           |
| § Associative law                       | § Exhaustive enumeration method     | § OR gate                       |
| § Boolean algebra                       | § Half-adder                        | § Parallel Binary Adder         |
| § Boolean expression                    | § Idempotent law                    | § Perfect induction method      |
| § Boolean functions                     | § Involution law                    | § Postulates of Boolean algebra |
| § Boolean identities                    | § Literal                           | § Principle of duality          |
| § Canonical forms for Boolean functions | § Logic circuits                    | § Product-of-Sums expression    |
| § Combination logic circuits            | § Logic gates                       | § Standard forms                |
| § Cumulative law                        | § Logical addition                  | § Sum-of Products expression    |
| § Complement of a function              | § Logical multiplication            | § Truth table                   |
| § Complementation                       | § Maxterms                          | § Universal NAND gate           |
| § De Morgan's law                       | § Minimization of Boolean functions | § Universal NOR gate            |
| § Distributive law                      | § Minterms                          |                                 |
| § Dual identities                       | § NAND gate                         |                                 |

## Chapter 07

# Processor and Memory

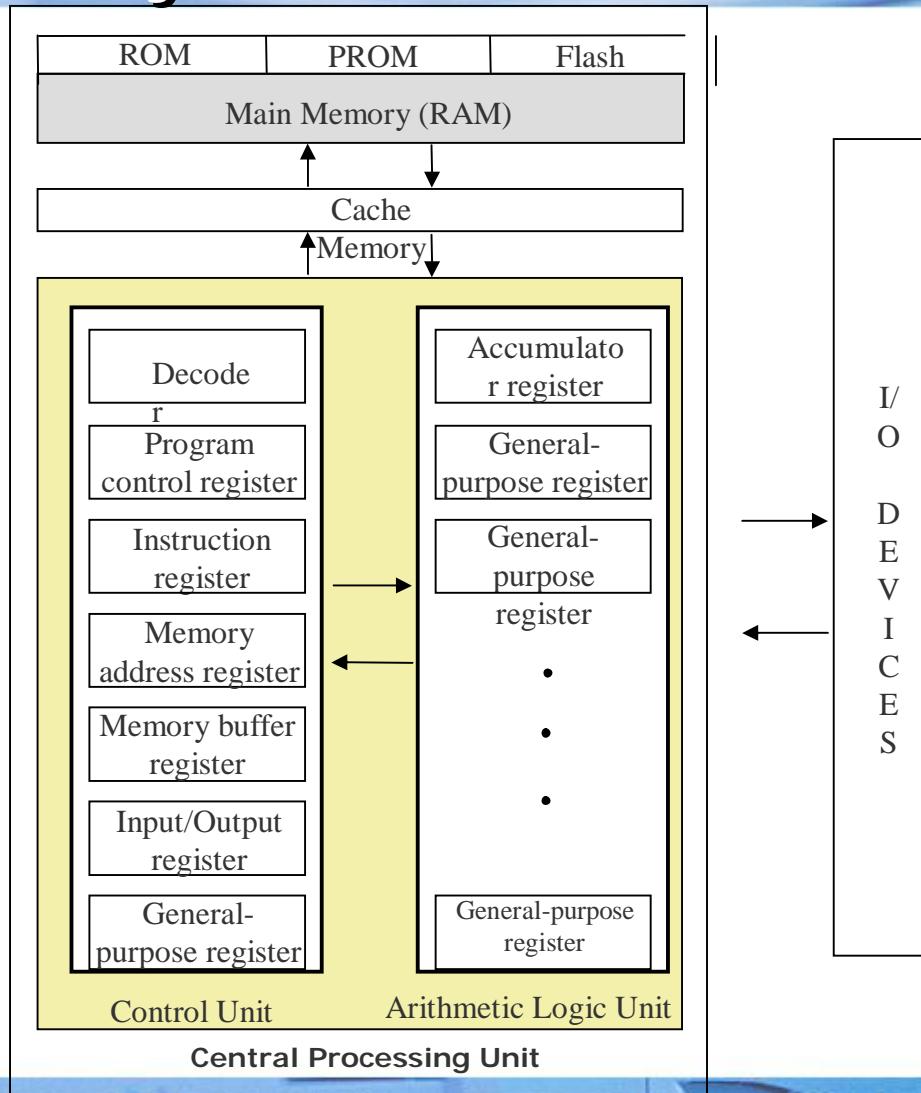
Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

In this chapter you will learn about:

- § Internal structure of processor
- § Memory structure
- § Determining the speed of a processor
- § Different types of processors available
- § Determining the capacity of a memory
- § Different types of memory available
- § Several other terms related to the processor and main memory of a computer system

# Basic Processor & Memory Architecture of a Computer System



# Central Processing Unit (CPU)

- § The *brain* of a computer system
- § Performs all major calculations and comparisons
- § Activates and controls the operations of other units of a computer system
- § Two basic components are
  - § Control Unit (CU)
  - § Arithmetic Logic Unit (ALU)
- § No other single component of a computer determines its overall performance as much as the CPU

# Control Unit (CU)

- § One of the two basic components of CPU
- § Acts as the central nervous system of a computer system
- § Selects and interprets program instructions, and coordinates execution
- § Has some special purpose registers and a decoder to perform these activities

# Arithmetic Logic Unit (ALU)

- § One of the two basic components of CPU.
- § Actual execution of instructions takes place in ALU
- § Has some special purpose registers
- § Has necessary circuitry to carry out all the arithmetic and logic operations included in the CPU instruction set

# Instruction Set

- § CPU has built-in ability to execute a particular set of machine instructions, called its *instruction set*
- § Most CPUs have 200 or more instructions (such as add, subtract, compare, etc.) in their instruction set
- § CPUs made by different manufacturers have different instruction sets
- § Manufacturers tend to group their CPUs into “families” having similar instruction sets
- § New CPU whose instruction set includes instruction set of its predecessor CPU is said to be *backward compatible* with its predecessor

# Registers

- § Special memory units, called registers, are used to hold information on a temporary basis as the instructions are interpreted and executed by the CPU
- § Registers are part of the CPU (not main memory) of a computer
- § The length of a register, sometimes called its *word size*, equals the number of bits it can store
- § With all other parameters being the same, a CPU with 32-bit registers can process data twice larger than one with 16-bit registers

# Functions of Commonly Used Registers

Sr. No.	Name of Register	Function
1	Memory Address (MAR)	Holds address of the active memory location
2	Memory Buffer (MBR)	Holds contents of the accessed (read/written) memory word
3	Program Control (PC)	Holds address of the next instruction to be executed
4	Accumulator (A)	Holds data to be operated upon, intermediate results, and the results
5	Instruction (I)	Holds an instruction while it is being executed
6	Input/Output (I/O)	Used to communicate with the I/O devices

# Processor Speed

- § Computer has a built-in *system clock* that emits millions of regularly spaced electric pulses per second (known as *clock cycles*)
- § It takes one cycle to perform a basic operation, such as moving a byte of data from one memory location to another
- § Normally, several clock cycles are required to fetch, decode, and execute a single program instruction
- § Hence, shorter the clock cycle, faster the processor
- § Clock speed (number of clock cycles per second) is measured in Megahertz ( $10^6$  cycles/sec) or Gigahertz ( $10^9$  cycles/sec)

# Types of Processor

Type of Architecture	Features	Usage
CISC (Complex Instruction Set Computer)	<ul style="list-style-type: none"> <li>§ Large instruction set</li> <li>§ Variable-length instructions</li> <li>§ Variety of addressing modes</li> <li>§ Complex &amp; expensive to produce</li> </ul>	Mostly used in personal computers
RISC (Reduced Instruction Set Computer)	<ul style="list-style-type: none"> <li>§ Small instruction set</li> <li>§ Fixed-length instructions</li> <li>§ Reduced references to memory to retrieve operands</li> </ul>	Mostly used in workstations

(Continued on next slide)

# Types of Processor

(Continued from previous slide..)

Type of Architecture	Features	Usage
EPIC (Explicitly Parallel Instruction Computing)	<ul style="list-style-type: none"> <li>§ Allows software to communicate explicitly to the processor when operations are parallel</li> <li>§ Uses tighter coupling between the compiler and the processor</li> <li>§ Enables compiler to extract maximum parallelism in the original code, and explicitly describe it to the processor</li> </ul>	Mostly used in high-end servers and workstations

(Continued on next slide)

# Types of Processor

(Continued from previous slide..)

Type of Architecture	Features	Usage
Multi-Core Processor	<ul style="list-style-type: none"><li>§ Processor chip has multiple cooler-running, more energy-efficient processing cores</li><li>§ Improve overall performance by handling more work in parallel</li><li>§ can share architectural components, such as memory elements and memory management</li></ul>	Mostly used in high-end servers and workstations

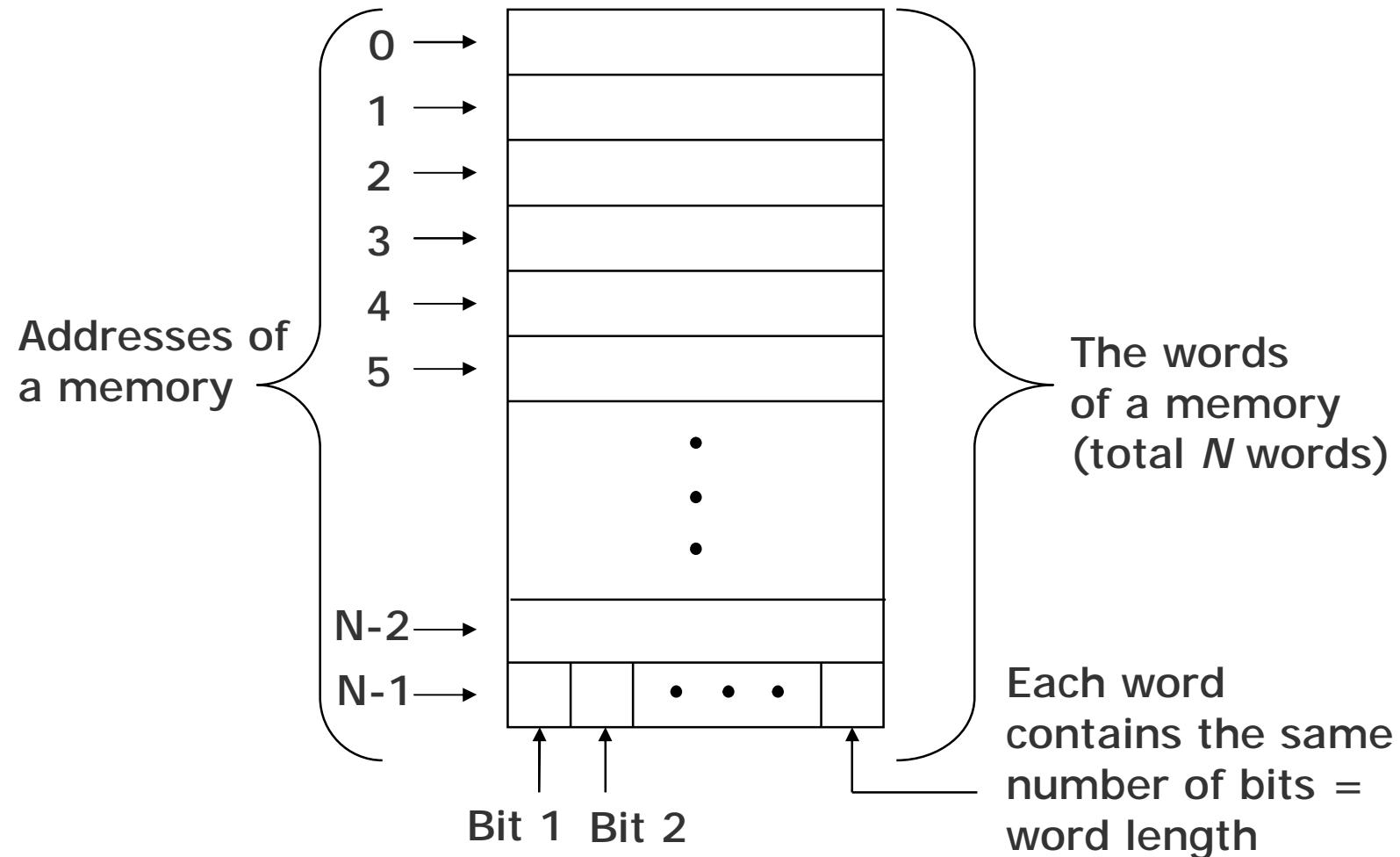
# Main Memory

- § Every computer has a temporary storage built into the computer hardware
- § It stores instructions and data of a program mainly when the program is being executed by the CPU.
- § This temporary storage is known as main memory, primary storage, or simply *memory*.
- § Physically, it consists of some chips either on the motherboard or on a small circuit board attached to the motherboard of a computer
- § It has random access property.
- § It is volatile.

# Storage Evaluation Criteria

Property	Desirable	Primary storage	Secondary storage
Storage capacity	Large storage capacity	Small	Large
Access Time	Fast access time	Fast	Slow
Cost per bit of storage	Lower cost per bit	High	Low
Volatility	Non-volatile	Volatile	Non-volatile
Access	Random access	Random access	Pseudo-random access or sequential access

# Main Memory Organization



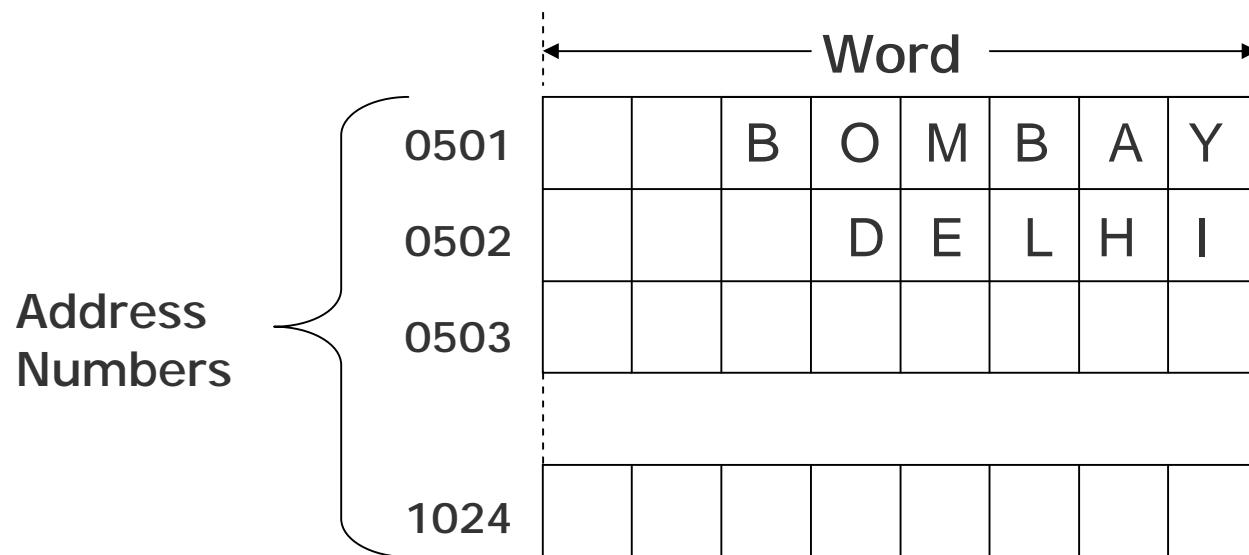
(Continued on next slide)

# Main Memory Organization

(Continued from previous slide..)

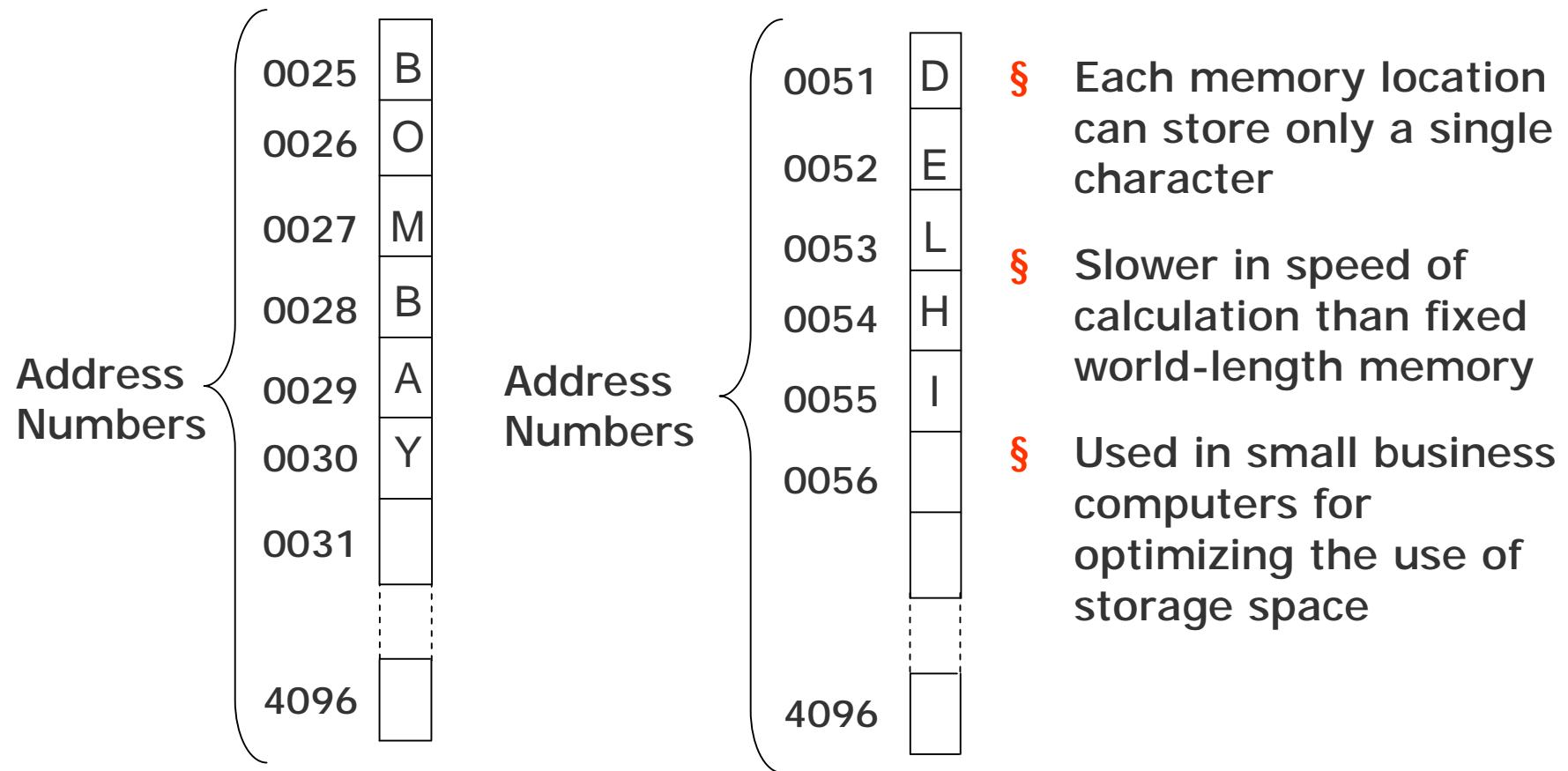
- § Machines having smaller word-length are slower in operation than machines having larger word-length
- § A *write* to a memory location is destructive to its previous contents
- § A *read* from a memory location is non-destructive to its previous contents

# Fixed Word-length Memory



- § Storage space is always allocated in multiples of word-length
- § Faster in speed of calculation than variable word-length memory
- § Normally used in large scientific computers for gaining speed of calculation

# Variable Word-length Memory



**Note:** With memory becoming cheaper and larger day-by-day, most modern computers employ fixed-word-length memory organization

# Memory Capacity

- § Memory capacity of a computer is equal to the number of bytes that can be stored in its primary storage
- § Its units are:

Kilobytes (KB) :  $1024 (2^{10})$  bytes

Megabytes (MB) :  $1,048,576 (2^{20})$  bytes

Gigabytes (GB) :  $1,073,741,824 (2^{30})$  bytes

# Random Access Memory (RAM)

- § Primary storage of a computer is often referred to as RAM because of its random access capability
- § RAM chips are volatile memory
- § A computer's motherboard is designed in a manner that the memory capacity can be enhanced by adding more memory chips
- § The additional RAM chips, which plug into special sockets on the motherboard, are known as *single-in-line memory modules (SIMMs)*

# Read Only Memory (ROM)

- § ROM a non-volatile memory chip
- § Data stored in a ROM can only be read and used – they cannot be changed
- § ROMs are mainly used to store programs and data, which do not change and are frequently used. For example, system boot program

# Types of ROMs

Type	Usage
Manufacturer-programmed ROM	Data is burnt by the manufacturer of the electronic equipment in which it is used.
User-programmed ROM or Programmable ROM (PROM)	The user can load and store “read-only” programs and data in it
Erasable PROM (EPROM)	The user can erase information stored in it and the chip can be reprogrammed to store new information

(Continued on next slide)

# Types of ROMs

(Continued from previous slide..)

Type	Usage
Ultra Violet EPROM (UVEPROM)	A type of EEPROM chip in which the stored information is erased by exposing the chip for some time to ultra-violet light
Electrically EPROM (EEPROM) or Flash memory	A type of EEPROM chip in which the stored information is erased by using high voltage electric pulses

# Cache Memory

- § It is commonly used for minimizing the memory-processor speed mismatch.
- § It is an extremely fast, small memory between CPU and main memory whose access time is closer to the processing speed of the CPU.
- § It is used to temporarily store very active data and instructions during processing.

*Cache is pronounced as “cash”*

# Key Words/Phrases

- § Accumulator Register (AR)
- § Address
- § Arithmetic Logic Unit (ALU)
- § Branch Instruction
- § Cache Memory
- § Central Processing Unit (CPU)
- § CISC (Complex Instruction Set Computer) architecture
- § Clock cycles
- § Clock speed
- § Control Unit
- § Electrically EPROM (EEPROM)
- § Erasable Programmable Read-Only Memory (EPROM)
- § Explicitly Parallel Instruction Computing (EPIC)
- § Fixed-word-length memory
- § Flash Memory
- § Input/Output Register (I/O)
- § Instruction Register (I)
- § Instruction set
- § Kilobytes (KB)
- § Main Memory
- § Manufacturer-Programmed ROM
- § Megabytes (MB)
- § Memory
- § Memory Address Register (MAR)
- § Memory Buffer Register (MBR)
- § Microprogram
- § Multi-core processor
- § Non-Volatile storage Processor
- § Program Control Register (PC)
- § Programmable Read-Only Memory (PROM)
- § Random Access Memory (RAM)

(Continued on next slide)

# Key Words/Phrases

(Continued from previous slide..)

- § Read-Only Memory (ROM)
- § Register
- § RISC (Reduced Instruction Set Computer) architecture
- § Single In-line Memory Module (SIMM)
- § Ultra Violet EPROM (UVEPROM)
- § Upward compatible
- § User-Programmed ROM
- § Variable-word-length memory
- § Volatile Storage
- § Word length
- § Word size

# Chapter 08

# Secondary Storage Devices

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

In this chapter you will learn about:

- § Secondary storage devices and their need
- § Classification of commonly used secondary storage devices
- § Difference between sequential and direct access storage devices
- § Basic principles of operation, types, and uses of popular secondary storage devices such as magnetic tape, magnetic disk, and optical disk

(Continued on next slide)

# Learning Objectives

(Continued from previous slide..)

- § Commonly used mass storage devices
- § Introduction to other related concepts such as RAID, Jukebox, storage hierarchy, etc.

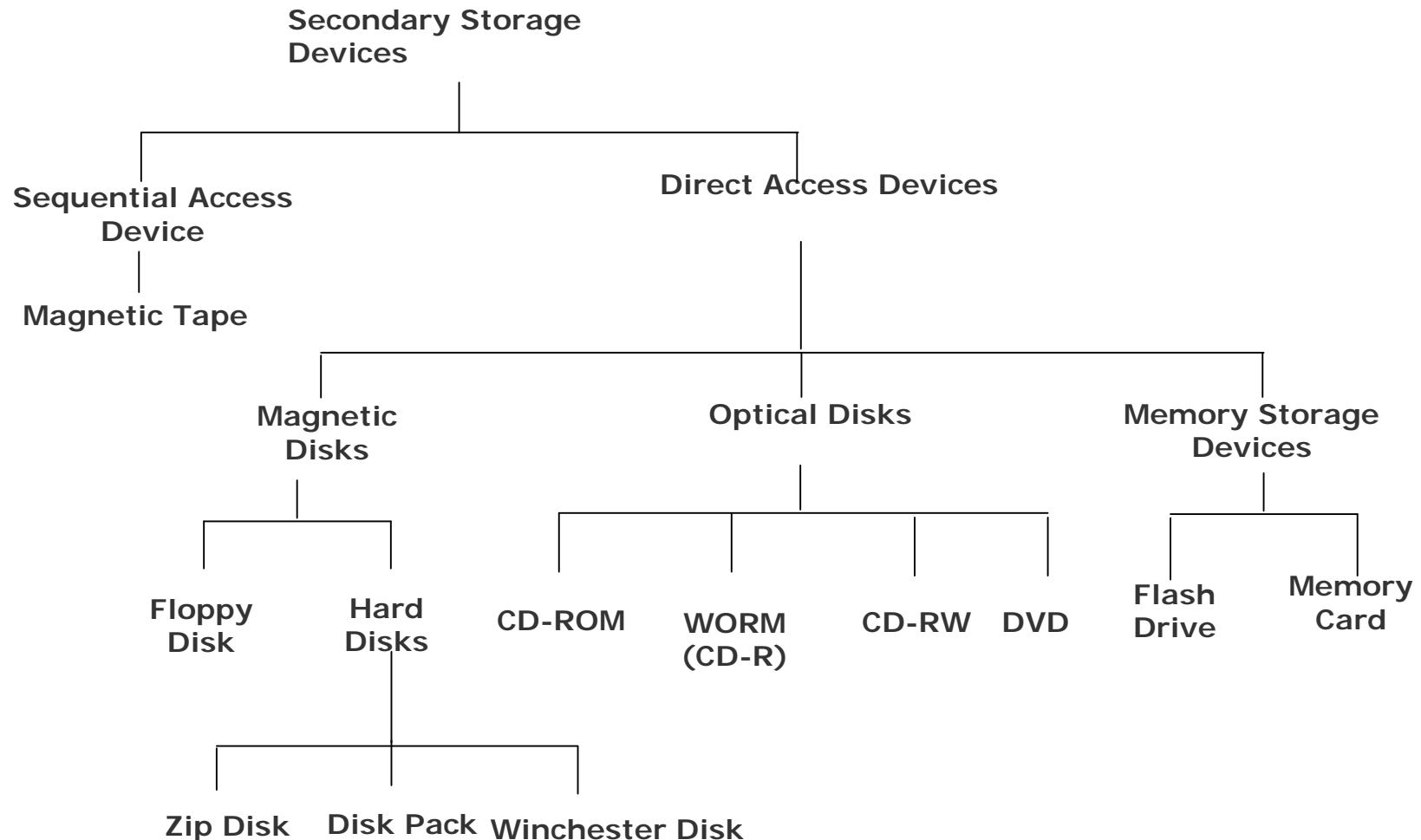
# Limitations of Primary Storage

- § Limited capacity because the cost per bit of storage is high
- § Volatile - data stored in it is lost when the electric power is turned off or interrupted

# Secondary Storage

- § Used in a computer system to overcome the limitations of primary storage
- § Has virtually unlimited capacity because the cost per bit of storage is very low
- § Has an operating speed far slower than that of the primary storage
- § Used to store large volumes of data on a permanent basis
- § Also known as *auxiliary memory*

# Classification of Commonly Used Secondary Storage Devices



# Sequential-access Storage Devices

- § Arrival at the desired storage location may be preceded by sequencing through other locations
- § Data can only be retrieved in the same sequence in which it is stored
- § Access time varies according to the storage location of the information being accessed
- § Suitable for sequential processing applications where most, if not all, of the data records need to be processed one after another
- § Magnetic tape is a typical example of such a storage device

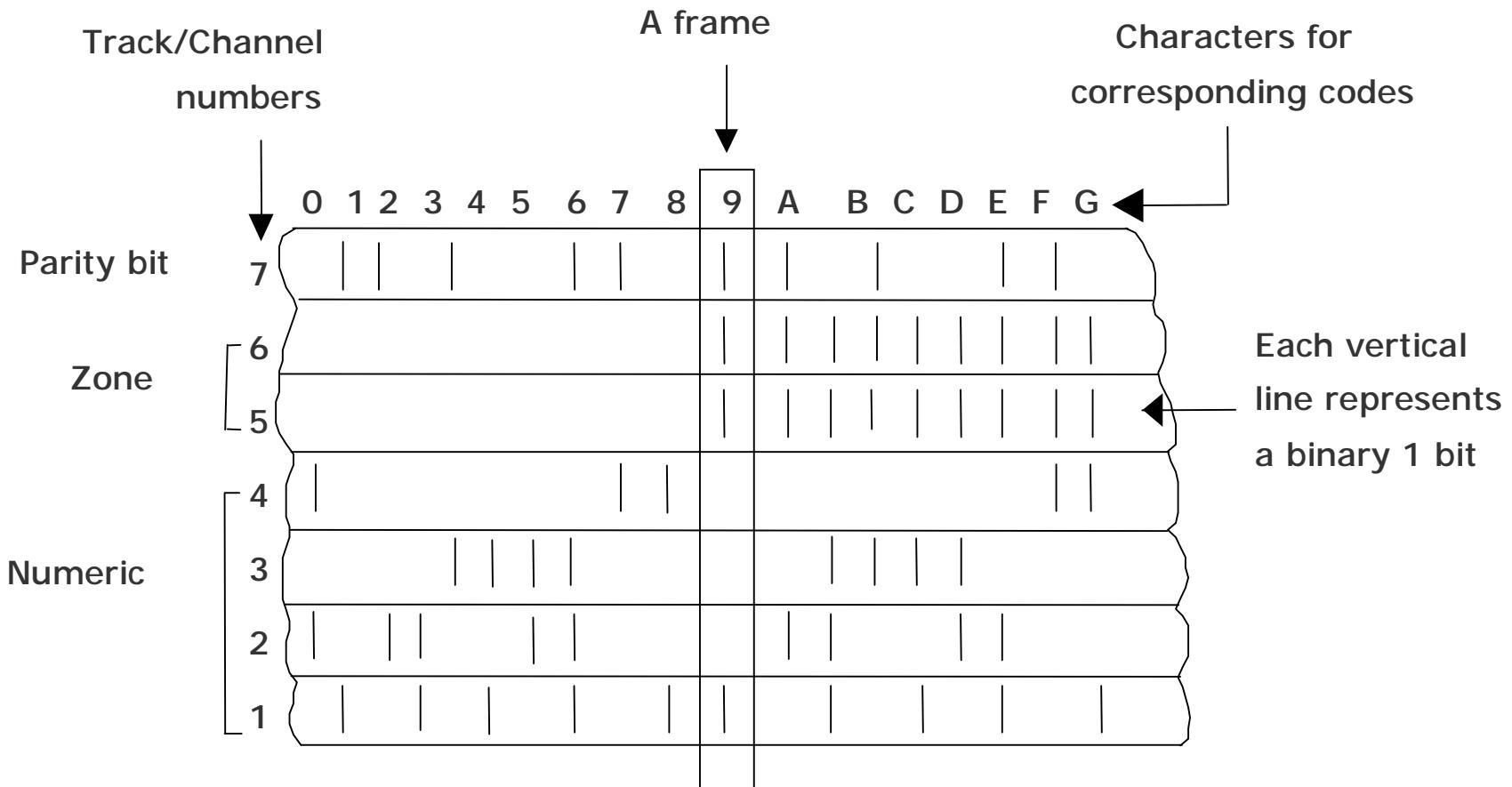
# Direct-access Storage Devices

- § Devices where any storage location may be selected and accessed at random
- § Permits access to individual information in a more direct or immediate manner
- § Approximately equal access time is required for accessing information from any storage location
- § Suitable for direct processing applications such as on-line ticket booking systems, on-line banking systems
- § Magnetic, optical, and magneto-optical disks are typical examples of such a storage device

## Magnetic Tape Basics

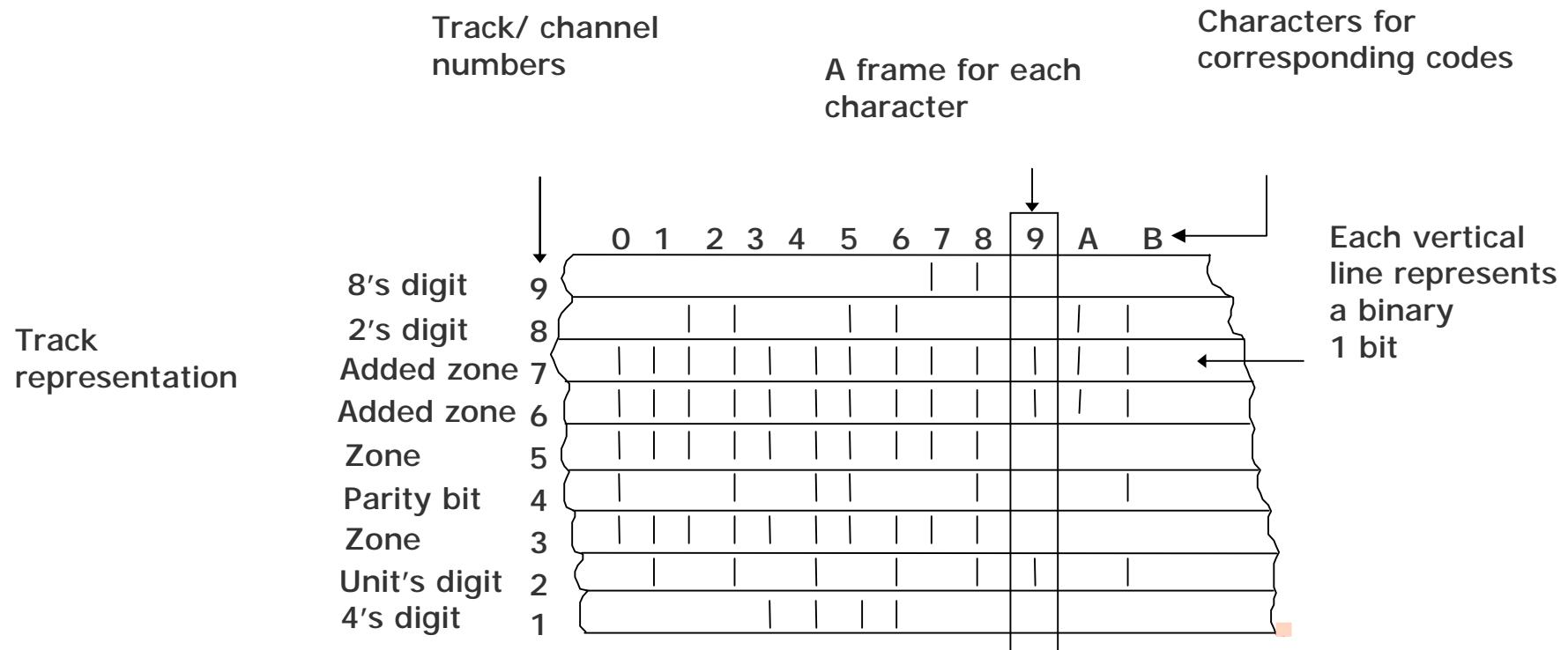
- § Commonly used sequential-access secondary storage device
- § Physically, the tape medium is a plastic ribbon, which is usually  $\frac{1}{2}$  inch or  $\frac{1}{4}$  inch wide and 50 to 2400 feet long
- § Plastic ribbon is coated with a magnetizable recording material such as iron-oxide or chromium dioxide
- § Data are recorded on the tape in the form of tiny invisible magnetized and non-magnetized spots (representing 1s and 0s) on its coated surface
- § Tape ribbon is stored in reels or a small cartridge or cassette

# Magnetic Tape - Storage Organization (Example 1)



Illustrates the concepts of frames, tracks, parity bit, and character-by-character data storage

# Magnetic Tape - Storage Organization (Example 2)

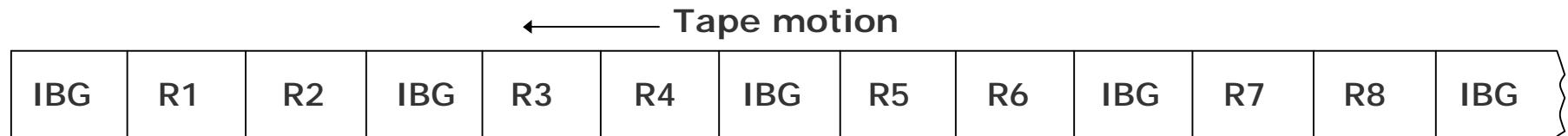


Illustrates the concepts of frames, tracks, parity bit, and character-by-character data storage

# Magnetic Tape - Storage Organization (Example 3)



(a) An unblocked tape. There is an IBG after each record.



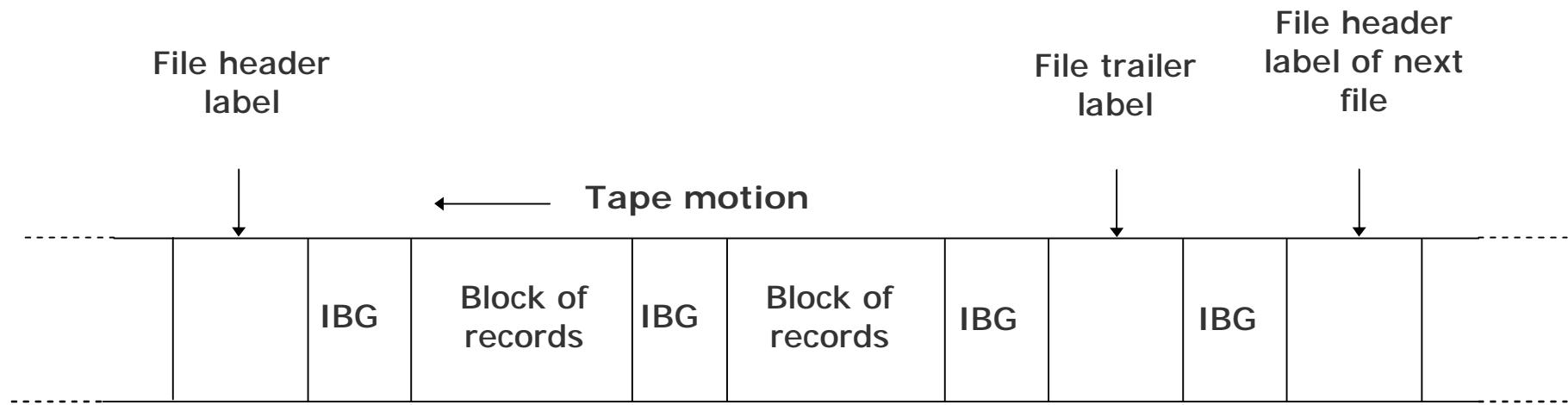
(b) A tape which uses a blocking factor of two. There is an IBG after every two records.



(c) A tape which uses a blocking factor of three. There is an IBG after every three records.

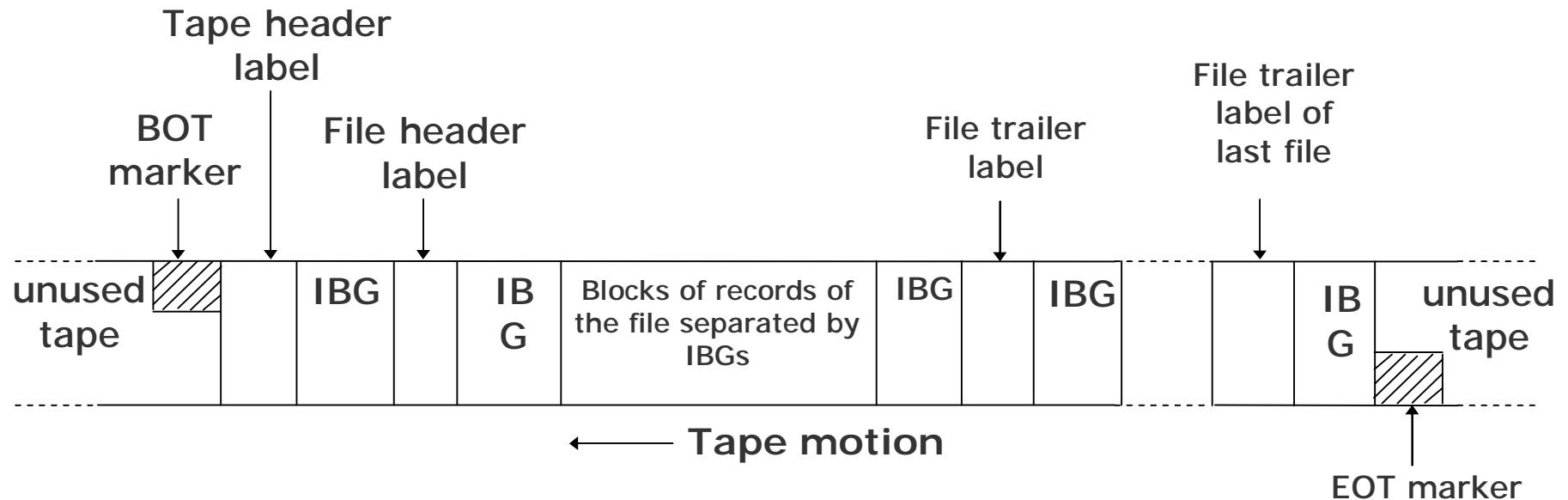
Illustrates the concepts of blocking of records, inter-block gap (IBG), and blocking factor

# Magnetic Tape - Storage Organization (Example 4)



Illustrates the concepts of multiple blocks of records forming a file that is separated from other files by a file header label in the beginning and a file trailer label at the end of the file

# Magnetic Tape-Storage Organization (Example 5)



Illustrates the concepts of Beginning of Tape (BoT) and End of Tape (EoT) markers, and tape header label

# Magnetic Tape Storage Capacity

- § Storage capacity of a tape =  
Data recording density x Length
- § Data recording density is the amount of data that can be stored on a given length of tape. It is measured in bytes per inch (bpi)
- § Tape density varies from 800 bpi in older systems to 77,000 bpi in some of the modern systems
- § Actual storage capacity of a tape may be anywhere from 35% to 70% of its total storage capacity, depending on the storage organization used

# Magnetic Tape – Data Transfer Rate

- § Refers to characters/second that can be transmitted to the memory from the tape
- § Transfer rate measurement unit is bytes/second (bps)
- § Value depends on the data recording density and the speed with which the tape travels under the read/write head
- § A typical value of data transfer rate is 7.7 MB/second

# Magnetic Tape – Tape Drive

- § Used for writing/reading of data to/from a magnetic tape ribbon
- § Different for tape reels, cartridges, and cassettes
- § Has read/write heads for reading/writing of data on tape
- § A magnetic tape reel/cartridge/cassette has to be first loaded on a tape drive for reading/writing of data on it
- § When processing is complete, the tape is removed from the tape drive for off-line storage

# Magnetic Tape – Tape Controller

- § Tape drive is connected to and controlled by a tape controller that interprets the commands for operating the tape drive
- § A typical set of commands supported by a tape controller are:

<i>Read</i>	reads one block of data
<i>Write</i>	writes one block of data
<i>Write tape header label</i>	used to update the contents of tape header label
<i>Erase tape</i>	erases the data recorded on a tape
<i>Back space one block</i>	rewinds the tape to the beginning of previous block

(Continued on next slide)

# Magnetic Tape – Tape Controller

(Continued from previous slide..)

*Forward space one block*

forwards the tape to the beginning of next block

*Forward space one file*

forwards the tape to the beginning of next file

*Rewind*

fully rewinds the tape

*Unload*

releases the tape drive's grip so that the tape spool can be unmounted from the tape drive

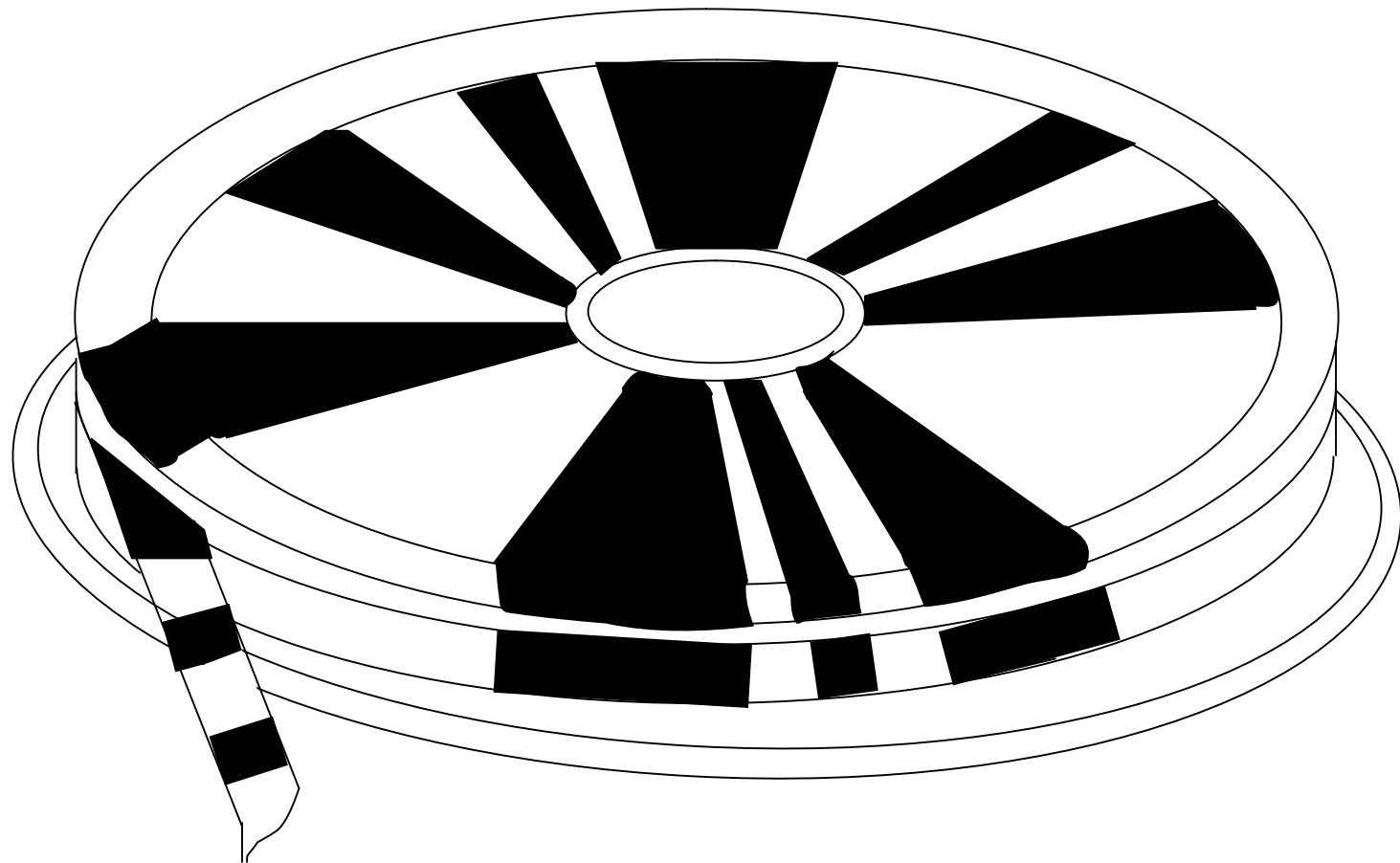
# Types of Magnetic Tape

- § ½-inch tape reel
- § ½-inch tape cartridge
- § ¼-inch streamer tape
- § 4-mm digital audio tape (DAT)

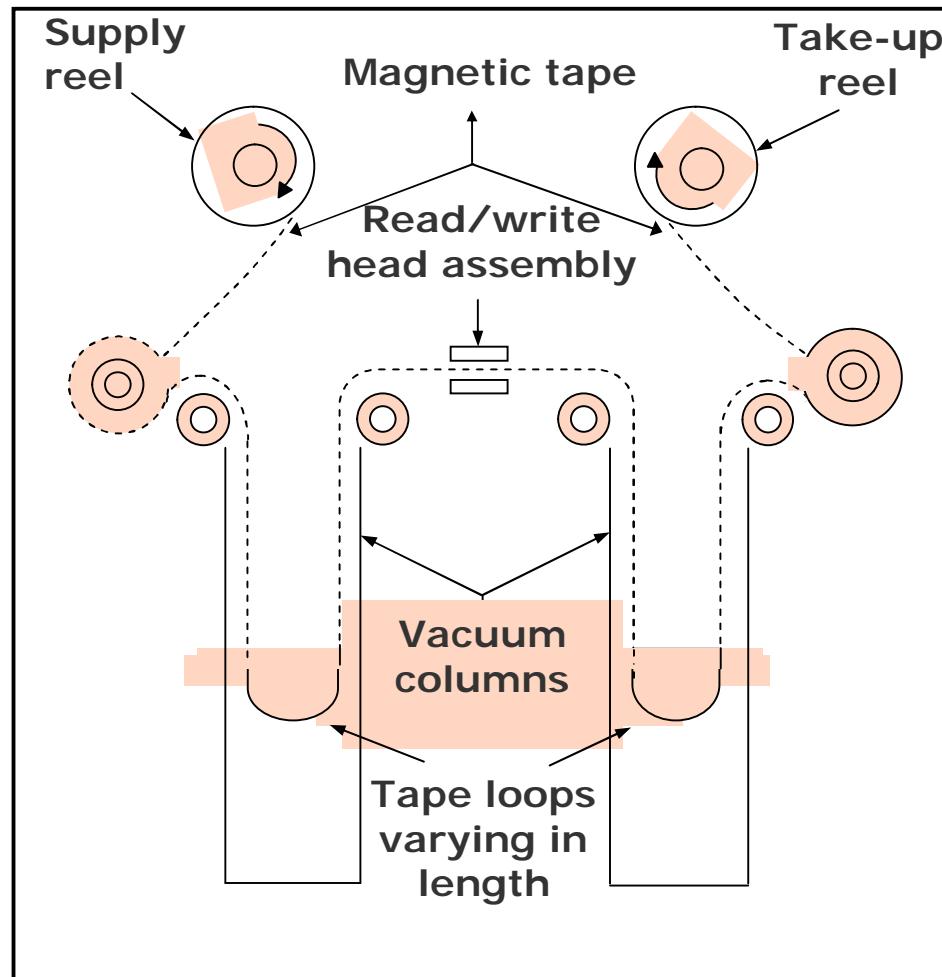
# Half-inch Tape Reel

- § Uses  $\frac{1}{2}$  inch wide tape ribbon stored on a tape reel
- § Uses parallel representation method of storing data, in which data are read/written a byte at a time
- § Uses a read/write head assembly that has one read/write head for each track
- § Commonly used as archival storage for off-line storage of data and for exchange of data and programs between organizations
- § Fast getting replaced by tape cartridge, streamer tape, and digital audio tape they are more compact, cheaper and easier to handle

# Half-inch Tape Reel



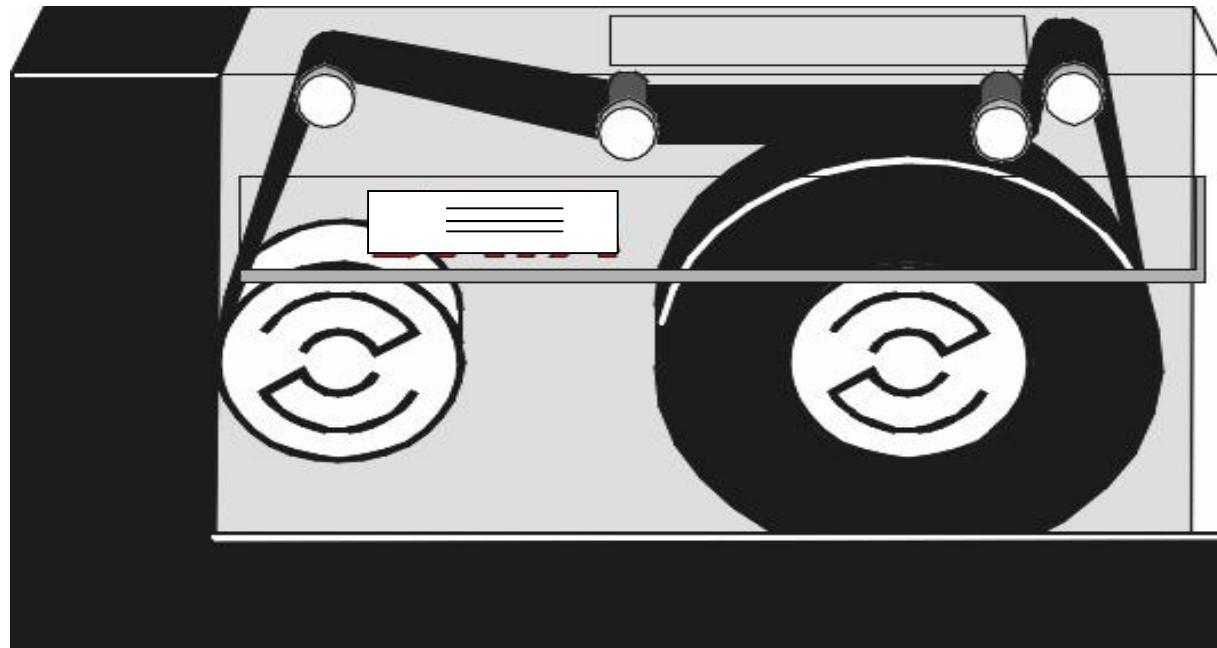
# Tape Drive of Half-inch Tape Reel



# Half-inch Tape Cartridge

- § Uses  $\frac{1}{2}$  inch wide tape ribbon sealed in a cartridge
- § Has 36 tracks, as opposed to 9 tracks for most half-inch tape reels
- § Stores data using parallel representation. Hence, 4 bytes of data are stored across the width of the tape. This enables more bytes of data to be stored on the same length of tape
- § Tape drive reads/writes on the top half of the tape in one direction and on the bottom half in the other direction

# Half-inch Tape Cartridge



# Quarter-inch Streamer Tape

- § Uses  $\frac{1}{4}$  inch wide tape ribbon sealed in a cartridge
- § Uses serial representation of data recording (data bits are aligned in a row one after another in tracks)
- § Can have from 4 to 30 tracks, depending on the tape drive
- § Depending on the tape drive, the read/write head reads/writes data on one/two/four tracks at a time
- § Eliminates the need for the start/stop operation of traditional tape drives

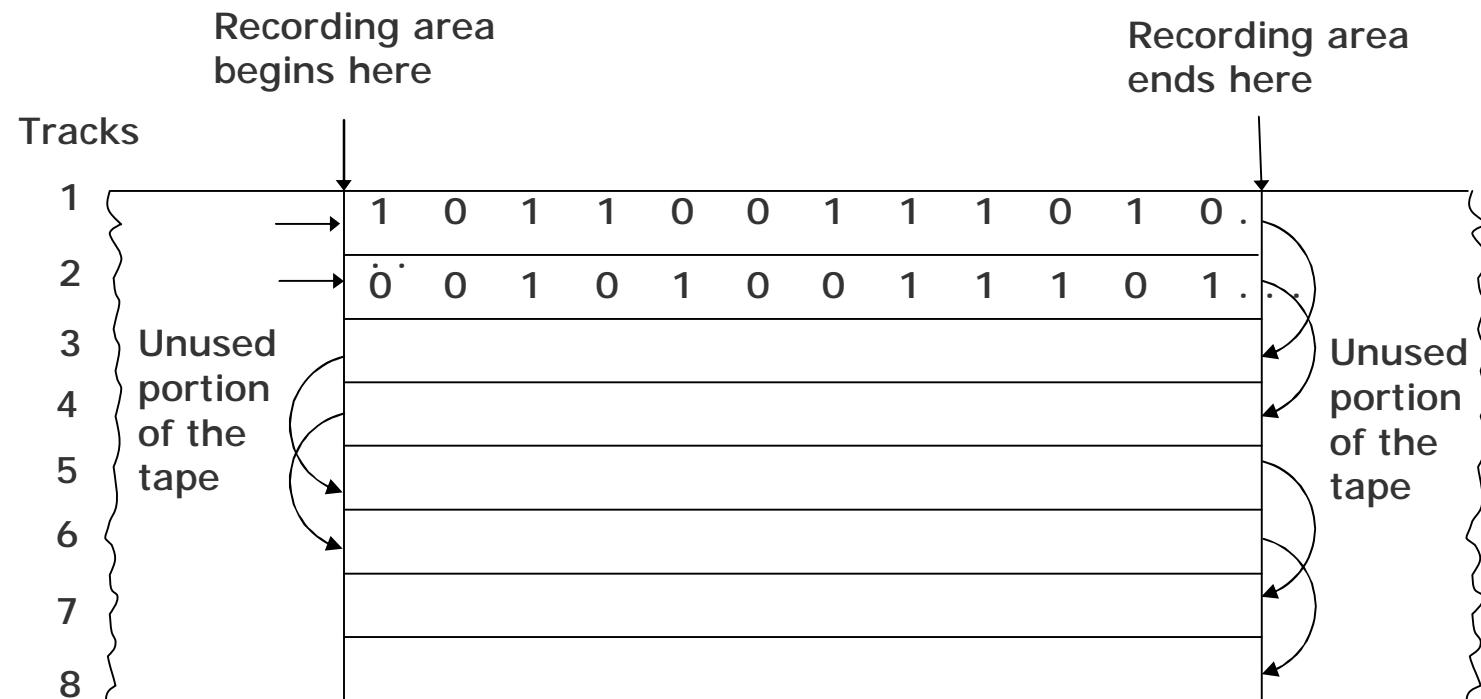
(Continued on next slide)

# Quarter-inch Streamer Tape

(Continued from previous slide..)

- § Can read/write data more efficiently than the traditional tape drives because there is no start/stop mechanism
- § Make more efficient utilization of tape storage area than traditional tape drives because IBGs are not needed
- § The standard data formats used in these tapes is known as the QIC standard

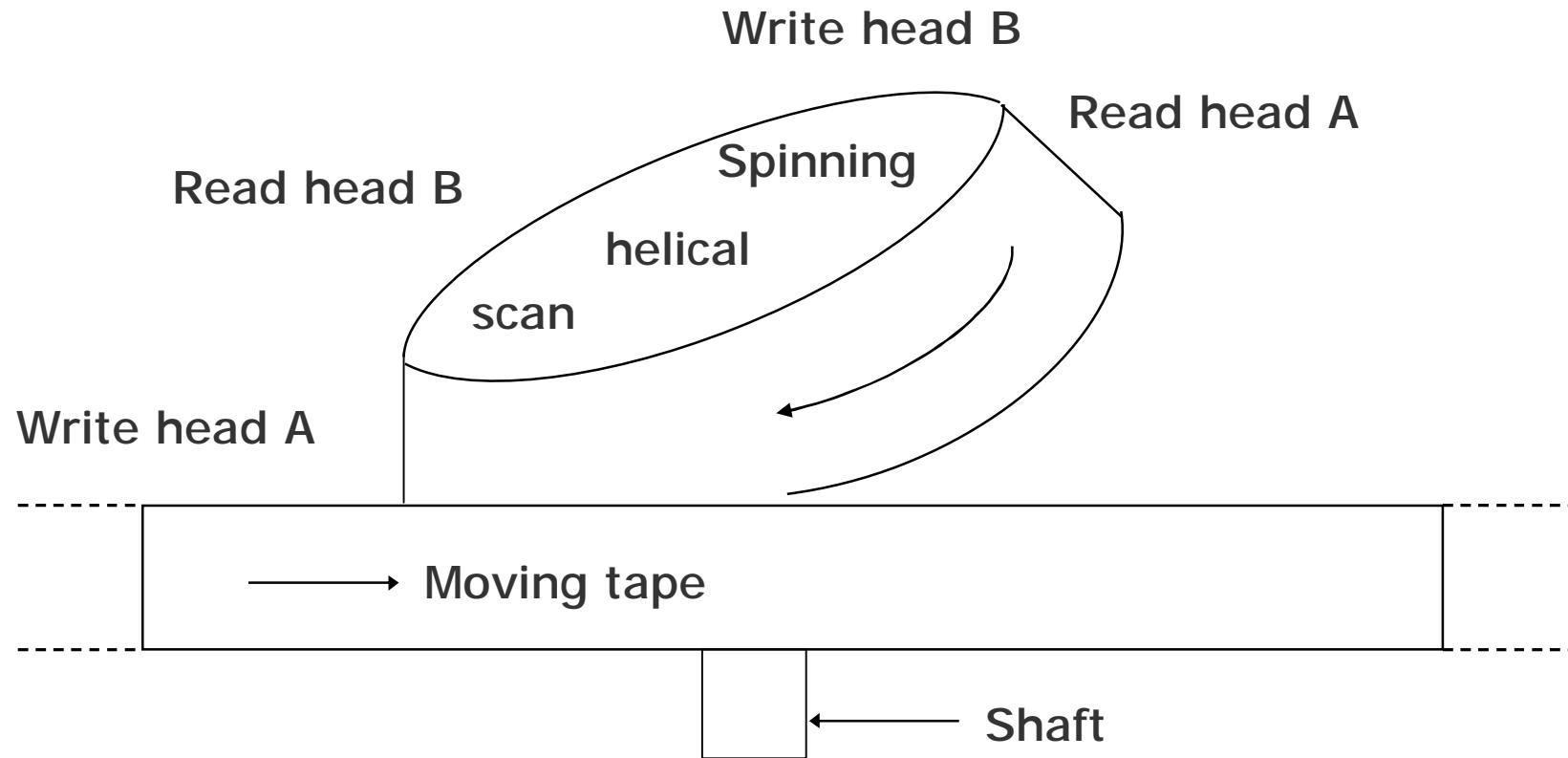
# Quarter-inch Streamer Tape (Example)



# 4mm Digital Audio Tape (DAT)

- § Uses 4mm wide tape ribbon sealed in a cartridge
- § Has very high data recording density
- § Uses a tape drive that uses helical scan technique for data recording, in which two read heads and two write heads are built into a small wheel
- § DAT drives use a data recording format called Digital Data Storage (DDS), which provides three levels of error-correcting code
- § Typical capacity of DAT cartridges varies from 4 GB to 14 GB

# The Helical Scan Techniques Used in DAT Drives



# Advantages of Magnetic Tapes

- § Storage capacity is virtually unlimited because as many tapes as required can be used for storing very large data sets
- § Cost per bit of storage is very low for magnetic tapes.
- § Tapes can be erased and reused many times
- § Tape reels and cartridges are compact and light in weight
- § Easy to handle and store.
- § Very large amount of data can be stored in a small storage space

(Continued on next slide)

# Advantages of Magnetic Tapes

(Continued from previous slide..)

- § Compact size and light weight
- § Magnetic tape reels and cartridges are also easily portable from one place to another
- § Often used for transferring data and programs from one computer to another that are not linked together

# Limitations of Magnetic Tapes

- § Due to their sequential access nature, they are not suitable for storage of those data that frequently require to be accessed randomly
- § Must be stored in a dust-free environment because specks of dust can cause tape-reading errors
- § Must be stored in an environment with properly controlled temperature and humidity levels
- § Tape ribbon may get twisted due to warping, resulting in loss of stored data
- § Should be properly labeled so that some useful data stored on a particular tape is not erased by mistake

# Uses of Magnetic Tapes

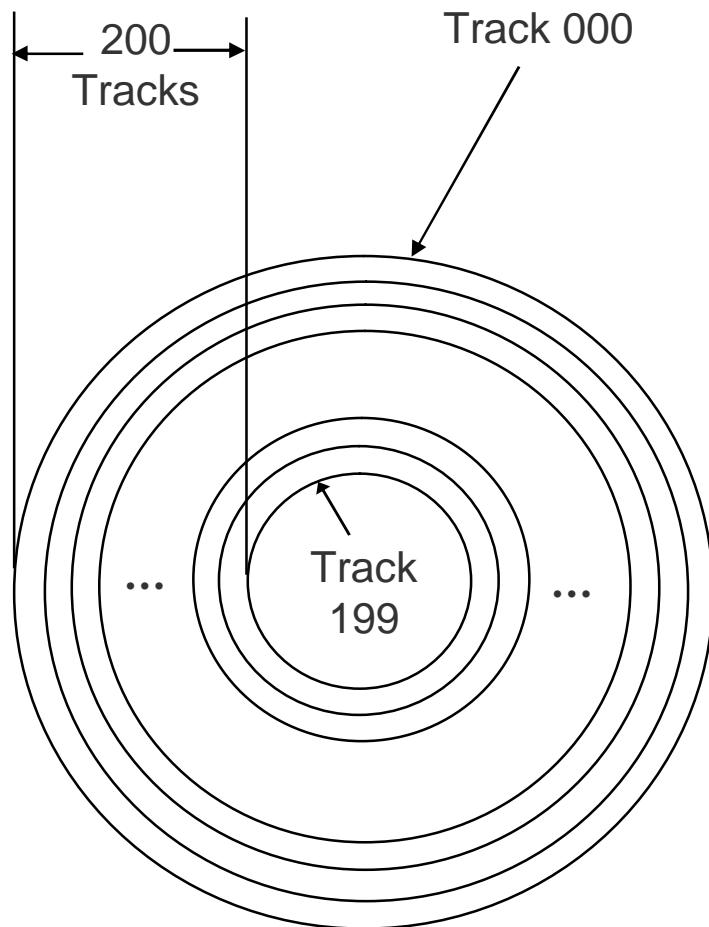
- § For applications that are based on sequential data processing
- § Backing up of data for off-line storage
- § Archiving of infrequently used data
- § Transferring of data from one computer to another that are not linked together
- § As a distribution media for software by vendors

# Magnetic Disk - Basics

- § Commonly used direct-access secondary storage device.
- § Physically, a magnetic disk is a thin, circular plate/platter made of metal or plastic that is usually coated on both sides with a magnetizable recording material such as iron-oxide
- § Data are recorded on the disk in the form of tiny invisible magnetized and non-magnetized spots (representing 1s and 0s) on the coated surfaces of the disk
- § The disk is stored in a specially designed protective envelope or cartridge, or several of them are stacked together in a sealed, contamination-free container

# Magnetic Disk – Storage Organization

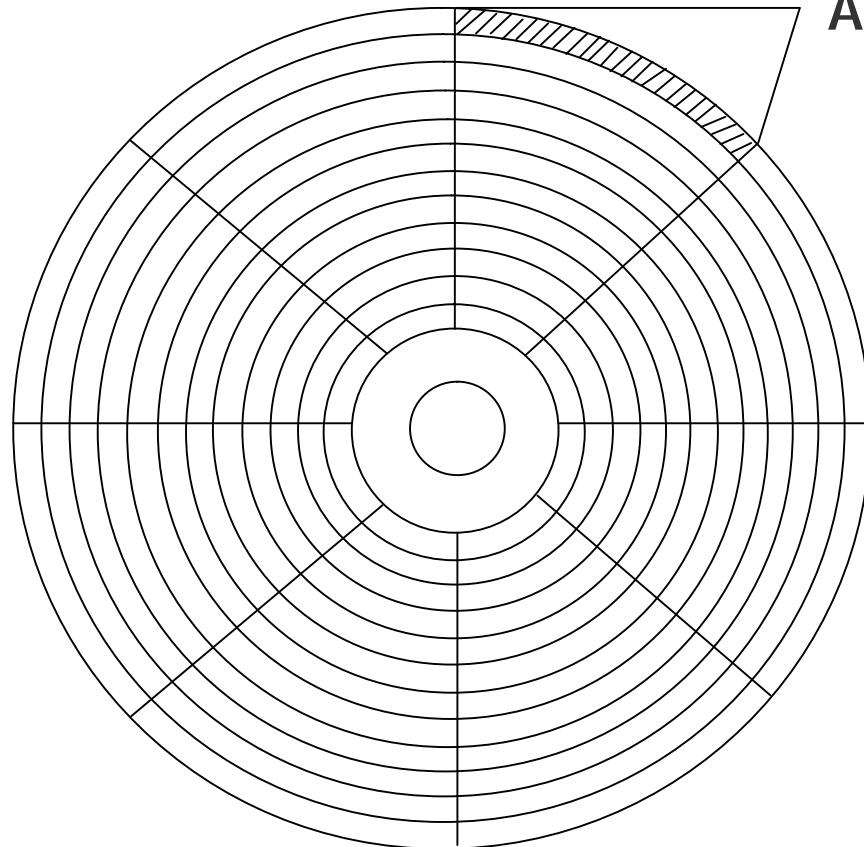
## Illustrates the Concept of Tracks



- § A disk's surface is divided into a number of invisible concentric circles called tracks
- § The tracks are numbered consecutively from outermost to innermost starting from zero
- § The number of tracks on a disk may be as few as 40 on small, low-capacity disks, to several thousand on large, high-capacity disks

# Magnetic Disk – Storage Organization

## Illustrates the Concept of Sectors

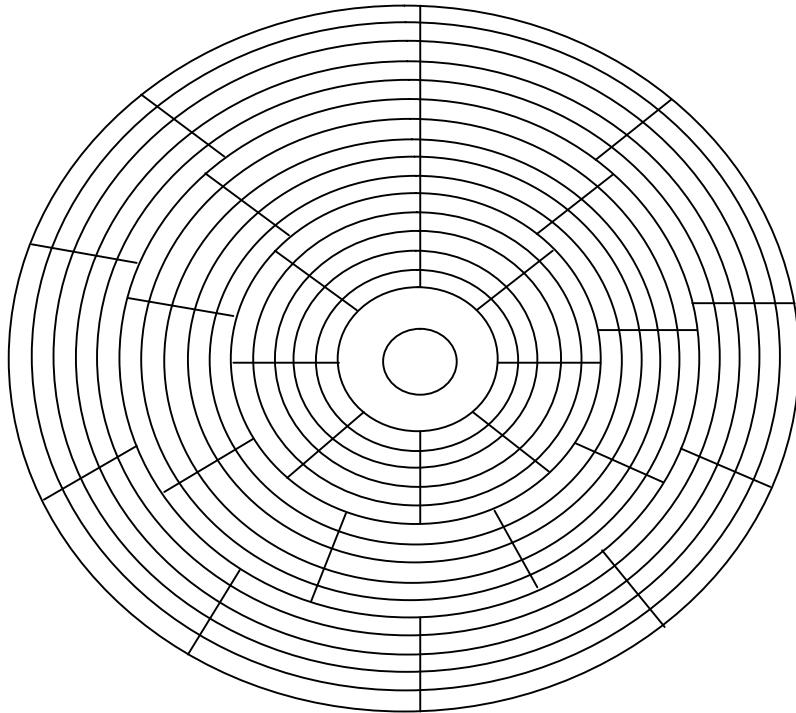


A sector

- § Each track of a disk is subdivided into sectors
- § There are 8 or more sectors per track
- § A sector typically contains 512 bytes
- § Disk drives are designed to read/write only whole sectors at a time

# Magnetic Disk – Storage Organization

Illustrates Grouping of Tracks and Use of Different Number of Sectors in Tracks of Different Groups for Increased Storage Capacity



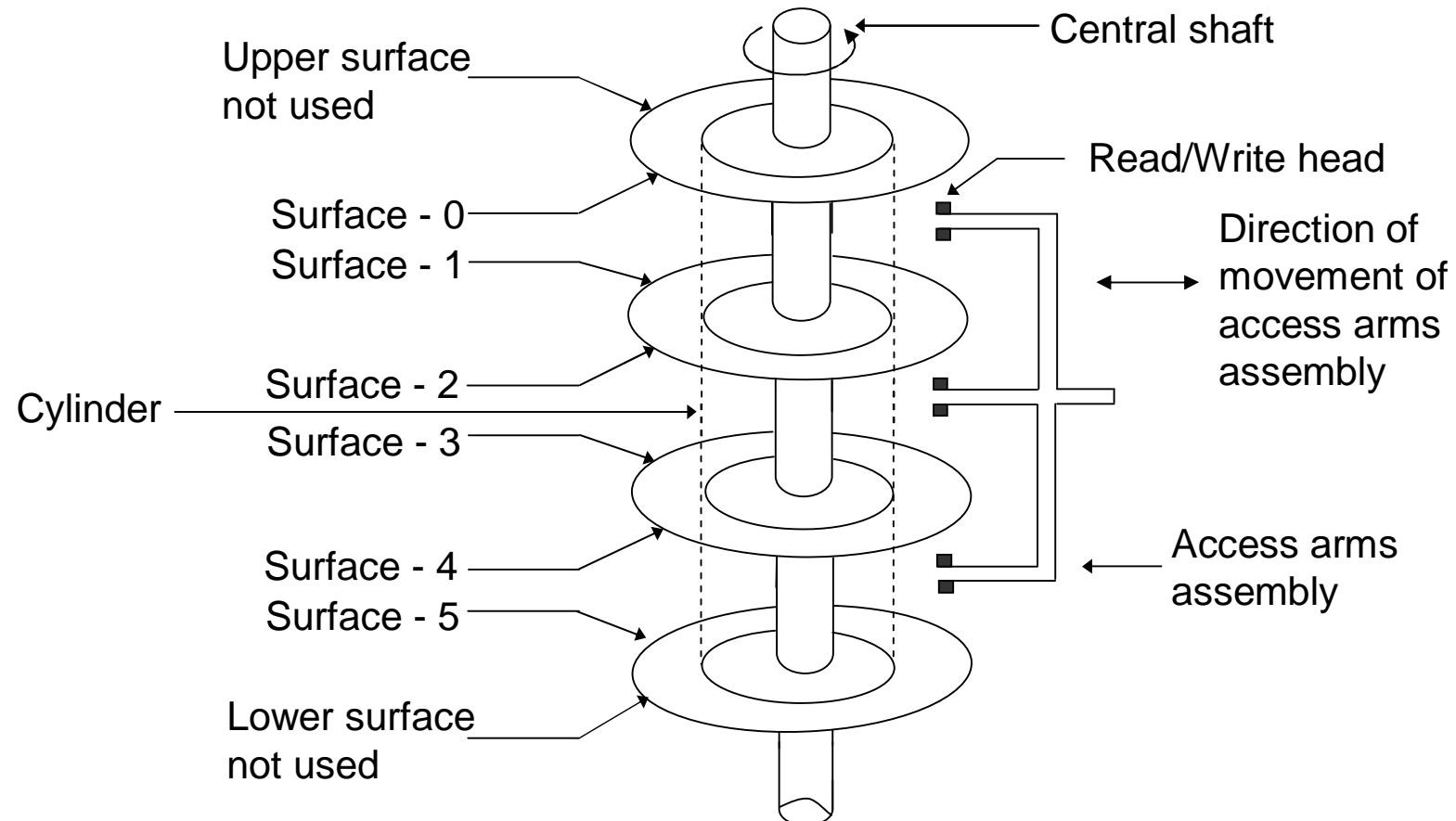
- § Innermost group of tracks has 8 sectors/track
- § Next groups of tracks has 9 sectors/track
- § Outermost group of tracks has 10 sectors/track

# Magnetic Disk – Disk Address or Address of a Record on a Disk

- § Disk address represents the physical location of the record on the disk
- § It is comprised of the sector number, track number, and surface number (when double-sided disks are used)
- § This scheme is called the *CHS addressing* or *Cylinder-Head-Sector* addressing. The same is also referred to as *disk geometry*

# Magnetic Disk – Storage Organization

(Illustrates the Concept of Cylinder)

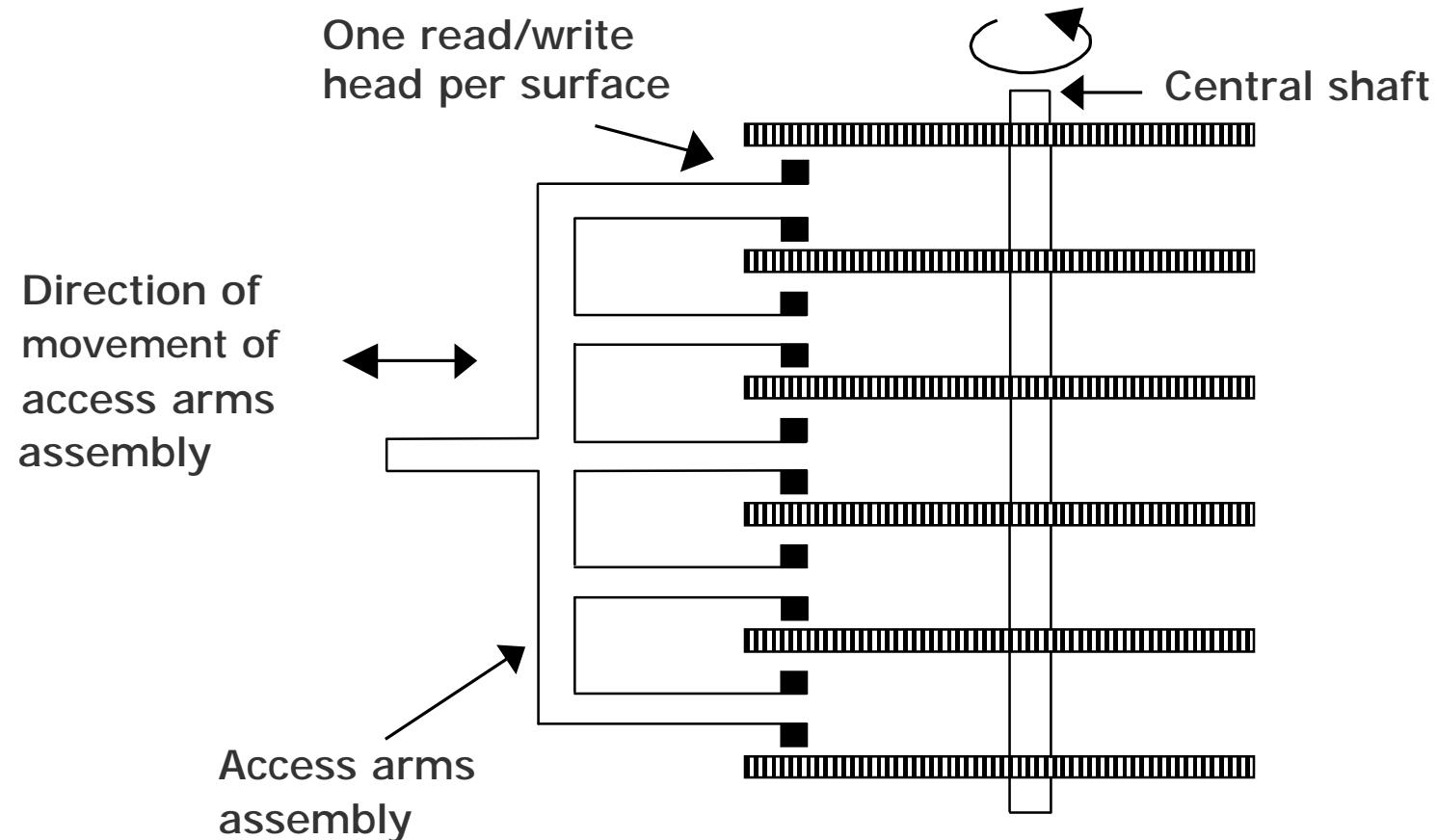


No. of disk platters = 4, No. of usable surfaces = 6. A set of corresponding tracks on all the 6 surfaces is called a cylinder.

# Magnetic Disk – Storage Capacity

Storage capacity of a disk system = Number of recording surfaces  
                                  × Number of tracks per surface  
                                  × Number of sectors per track  
                                  × Number of bytes per sector

# Magnetic Disk Pack – Access Mechanism



Vertical cross section of a disk system. There is one read/write head per recording surface

# Magnetic Disk – Access Time

- § *Disk access time* is the interval between the instant a computer makes a request for transfer of data from a disk system to the primary storage and the instant this operation is completed
- § Disk access time depends on the following three parameters:
  - *Seek Time*: It is the time required to position the read/write head over the desired track, as soon as a read/write command is received by the disk unit
  - *Latency*: It is the time required to spin the desired sector under the read/write head, once the read/write head is positioned on the desired track

# Magnetic Disk – Access Time

- *Transfer Rate:* It is the rate at which data are read/written to the disk, once the read/write head is positioned over the desired sector
- § As the transfer rate is negligible as compared to seek time and latency,

Average access time

$$= \text{Average seek time} + \text{Average latency}$$

# Disk Formatting

- § Process of preparing a new disk by the computer system in which the disk is to be used.
- § For this, a new (unformatted) disk is inserted in the disk drive of the computer system and the disk formatting command is initiated
- § Low-level disk formatting
  - § Disk drive's read/write head lays down a magnetic pattern on the disk's surface
  - § Enables the disk drive to organize and store the data in the data organization defined for the disk drive of the computer

(Continued on next slide)

# Disk Formatting

(Continued from previous slide..)

## § OS-level disk formatting

- § Creates the File Allocation Table (FAT) that is a table with the sector and track locations of data
- § Leaves sufficient space for FAT to grow
- § Scans and marks bad sectors
- § One of the basic tasks handled by the computer's operating system
- § Enables the use of disks manufactured by third party vendors into one's own computer system

# Magnetic Disk – Disk Drive

- § Unit used for reading/writing of data on/from a magnetic disk
- § Contains all the mechanical, electrical and electronic components for holding one or more disks and for reading or writing of information on to it

(Continued on next slide)

# Magnetic Disk – Disk Drive

(Continued from previous slide..)

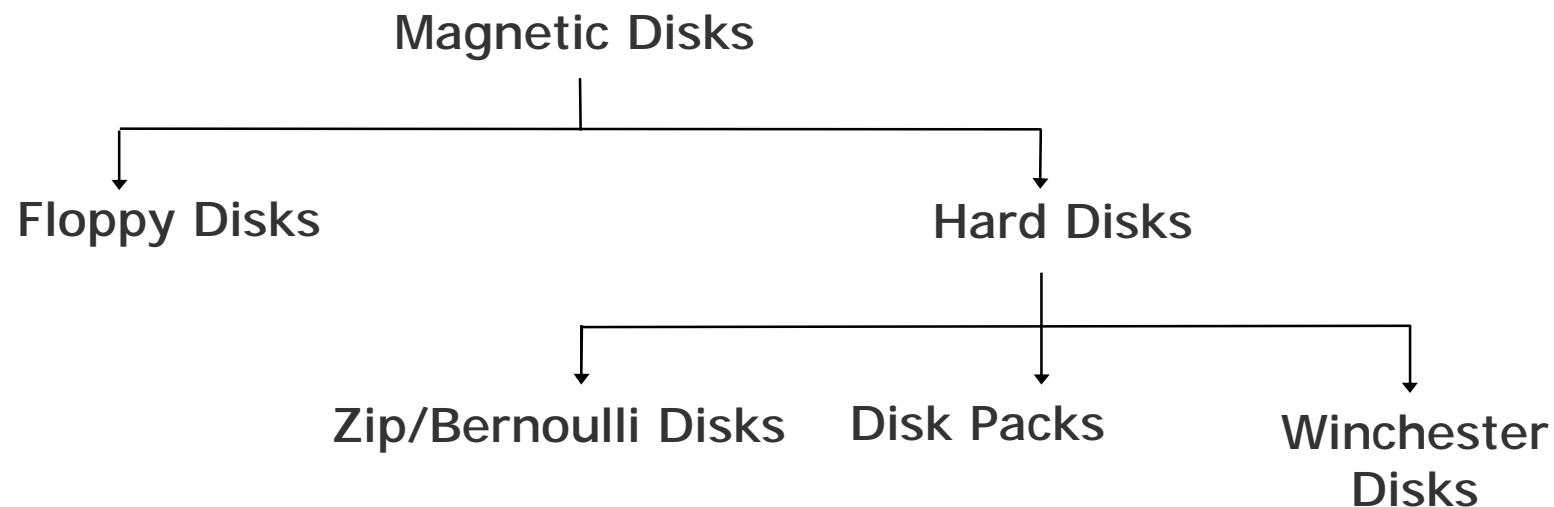
§ Although disk drives vary greatly in their shape, size and disk formatting pattern, they can be broadly classified into two types:

- *Those with interchangeable magnetic disks*, which allow the loading and unloading of magnetic disks as and when they are needed for reading/writing of data on to them
- *Those with fixed magnetic disks*, which come along with a set of permanently fixed disks. The disks are not removable from their disk drives

# Magnetic Disk – Disk Controller

- § Disk drive is connected to and controlled by a disk controller, which interprets the commands for operating the disk drive
- § Typically supports only *read* and *write* commands, which need disk address (surface number, cylinder/track number, and sector number) as parameters
- § Connected to and controls more than one disk drive, in which case the disk drive number is also needed as a parameter of *read* and *write* commands

# Types of Magnetic Disks



# Floppy Disks

- § Round, flat piece of flexible plastic disks coated with magnetic oxide
- § So called because they are made of flexible plastic plates which can bend
- § Also known as floppies or diskettes
- § Plastic disk is encased in a square plastic or vinyl jacket cover that gives handling protection to the disk surface

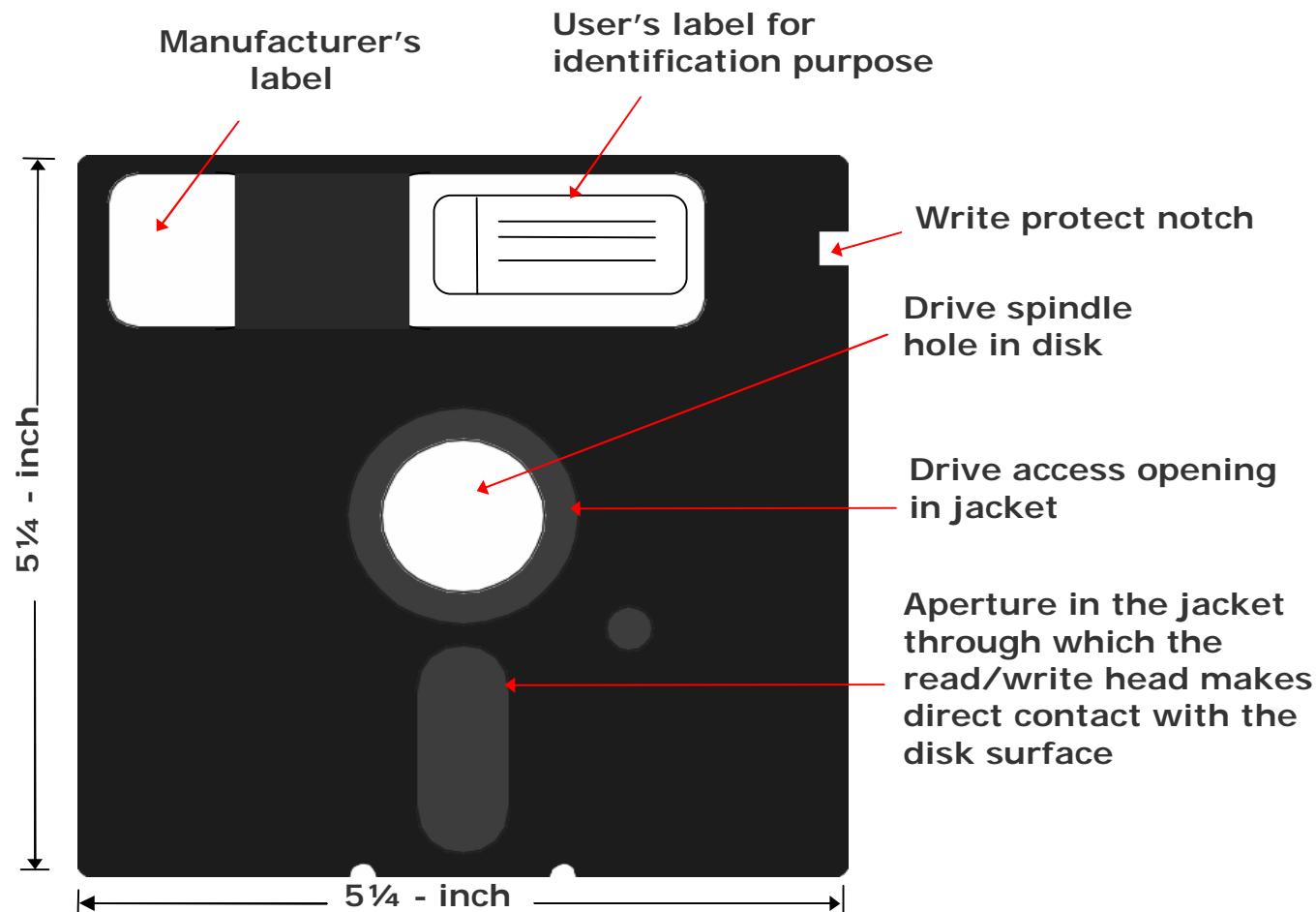
(Continued on next slide)

# Floppy Disks

(Continued from previous slide..)

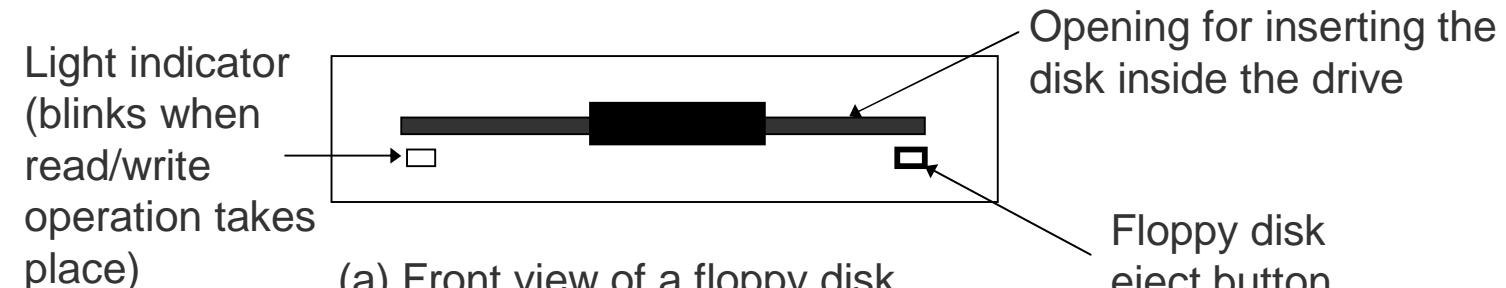
- § The two types of floppy disks in use today are:
  - § 5½-inch diskette, whose diameter is 5½-inch.  
It is encased in a square, flexible vinyl jacket
  - § 3½-inch diskette, whose diameter is 3½-inch.  
It is encased in a square, hard plastic jacket
- § Most popular and inexpensive secondary storage medium used in small computers

# A 5½-inch Floppy Disk

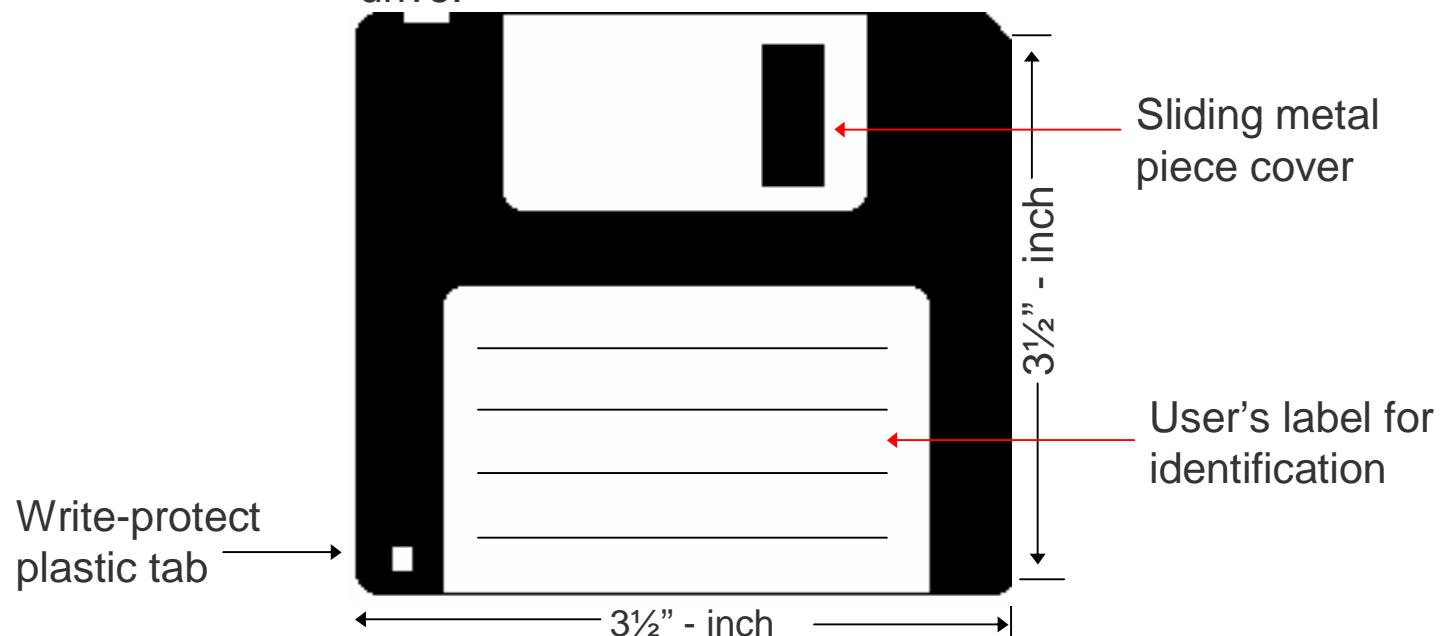


A 5½-inch floppy disk enclosed within jacket. The drive mechanism clamps on to a portion of the disk exposed by the drive access opening in the jacket

# A 3½-inch Floppy Disk



(a) Front view of a floppy disk drive.



(b) A 3½ - inch floppy disk.

# Storage Capacities of Various Types of Floppy Disks

Size (Diameter in inches)	No. of surfaces	No. of tracks	No. of sectors/track	No. of bytes/sector	Capacity in bytes	Approximate capacity
5¼	2	40	9	512	3,68,640	360 KB
5¼	2	80	15	512	12,28,800	1.2 MB
3½	2	40	18	512	7,37,280	720 KB
3½	2	80	18	512	14,74,560	1.4 MB
3½	2	80	36	512	29,49,120	2.88 MB

# Hard Disks

- § Round, flat piece of rigid metal (frequently aluminium) disks coated with magnetic oxide
- § Come in many sizes, ranging from 1 to 14-inch diameter.
- § Depending on how they are packaged, hard disks are of three types:
  - § Zip/Bernoulli disks
  - § Disk packs
  - § Winchester disks
- § Primary on-line secondary storage device for most computer systems today

# Zip/Bernoulli Disks

- § Uses a single hard disk platter encased in a plastic cartridge
- § Disk drives may be portable or fixed type
- § Fixed type is part of the computer system, permanently connected to it
- § Portable type can be carried to a computer system, connected to it for the duration of use, and then can be disconnected and taken away when the work is done
- § Zip disks can be easily inserted/removed from a zip drive just as we insert/remove floppy disks in a floppy disk drive

# Disk Packs

- § Uses multiple (two or more) hard disk platters mounted on a single central shaft
- § Disk drives have a separate read/write head for each usable disk surface (the upper surface of the top-most disk and the lower surface of the bottom most disk is not used)
- § Disks are of removable/interchangeable type in the sense that they have to be mounted on the disk drive before they can be used, and can be removed and kept off-line when not in use

# Winchester Disks

- § Uses multiple (two or more) hard disk platters mounted on a single central shaft
- § Hard disk platters and the disk drive are sealed together in a contamination-free container and cannot be separated from each other

(Continued on next slide)

# Winchester Disks

(Continued from previous slide..)

§ For the same number of disks, Winchester disks have larger storage capacity than disk packs because:

- All the surfaces of all disks are used for data recording

They employ much greater precision of data recording, resulting in greater data recording density

§ Named after the .30-30 Winchester rifle because the early Winchester disk systems had two 30-MB disks sealed together with the disk drive

# Advantages of Magnetic Disks

- § More suitable than magnetic tapes for a wider range of applications because they support direct access of data
- § Random access property enables them to be used simultaneously by multiple users as a shared device. A tape is not suitable for such type of usage due to its sequential-access property
- § Suitable for both on-line and off-line storage of data

(Continued on next slide)

# Advantages of Magnetic Disks

(Continued from previous slide..)

- § Except for the fixed type Winchester disks, the storage capacity of other magnetic disks is virtually unlimited as many disks can be used for storing very large data sets
- § Due to their low cost and high data recording densities, the cost per bit of storage is low for magnetic disks.
- § An additional cost benefit is that magnetic disks can be erased and reused many times
- § Floppy disks and zip disks are compact and light in weight. Hence they are easy to handle and store.
- § Very large amount of data can be stored in a small storage space

(Continued on next slide)

# Advantages of Magnetic Disks

- § Due to their compact size and light weight, floppy disks and zip disks are also easily portable from one place to another
- § They are often used for transferring data and programs from one computer to another, which are not linked together
- § Any information desired from a disk storage can be accessed in a few milliseconds because it is a direct access storage device

(Continued on next slide)

# Advantages of Magnetic Disks

(Continued from previous slide..)

- § Data transfer rate for a magnetic disk system is normally higher than a tape system
- § Magnetic disks are less vulnerable to data corruption due to careless handling or unfavorable temperature and humidity conditions than magnetic tapes

# Limitations of Magnetic Disks

- § Although used for both random processing and sequential processing of data, for applications of the latter type, it may be less efficient than magnetic tapes
- § More difficult to maintain the security of information stored on shared, on-line secondary storage devices, as compared to magnetic tapes or other types of magnetic disks

(Continued on next slide)

# Limitations of Magnetic Disks

(Continued from previous slide..)

- § For Winchester disks, a disk crash or drive failure often results in loss of entire stored data. It is not easy to recover the lost data. Suitable backup procedures are suggested for data stored on Winchester disks
- § Some types of magnetic disks, such as disk packs and Winchester disks, are not so easily portable like magnetic tapes
- § On a cost-per-bit basis, the cost of magnetic disks is low, but the cost of magnetic tapes is even lower

(Continued on next slide)

# Limitations of Magnetic Disks

(Continued from previous slide..)

- § Must be stored in a dust-free environment
- § Floppy disks, zip disks and disk packs should be labeled properly to prevent erasure of useful data by mistake

# Uses of Magnetic Disks

- § For applications that are based on random data processing
- § As a shared on-line secondary storage device. Winchester disks and disk packs are often used for this purpose
- § As a backup device for off-line storage of data. Floppy disks, zip disks, and disk packs are often used for this purpose

(Continued on next slide)

# Uses of Magnetic Disks

(Continued from previous slide..)

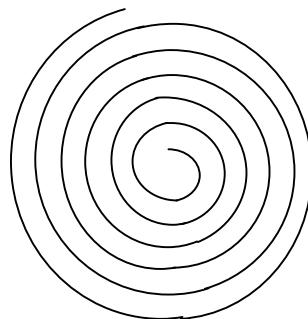
- § Archiving of data not used frequently, but may be used once in a while. Floppy disks, zip disks, and disk packs are often used for this purpose
- § Transferring of data and programs from one computer to another that are not linked together. Floppy disks and zip disks are often used for this purpose
- § Distribution of software by vendors. Originally sold software or software updates are often distributed by vendors on floppy disks and zip disks

# Optical Disk – Basics

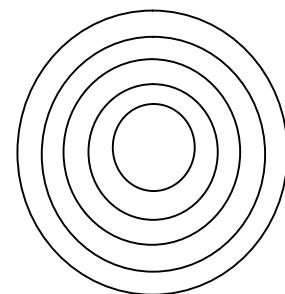
- § Consists of a circular disk, which is coated with a thin metal or some other material that is highly reflective
- § Laser beam technology is used for recording/reading of data on the disk
- § Also known as laser disk / optical laser disk, due to the use of laser beam technology
- § Proved to be a promising random access medium for high capacity secondary storage because it can store extremely large amounts of data in a limited space

# Optical Disk – Storage Organization

- § Has one long spiral track, which starts at the outer edge and spirals inward to the center
- § Track is divided into equal size sectors



(a) Track pattern on an optical disk



(b) Track pattern on a magnetic disk

Difference in track patterns on optical and magnetic disks.

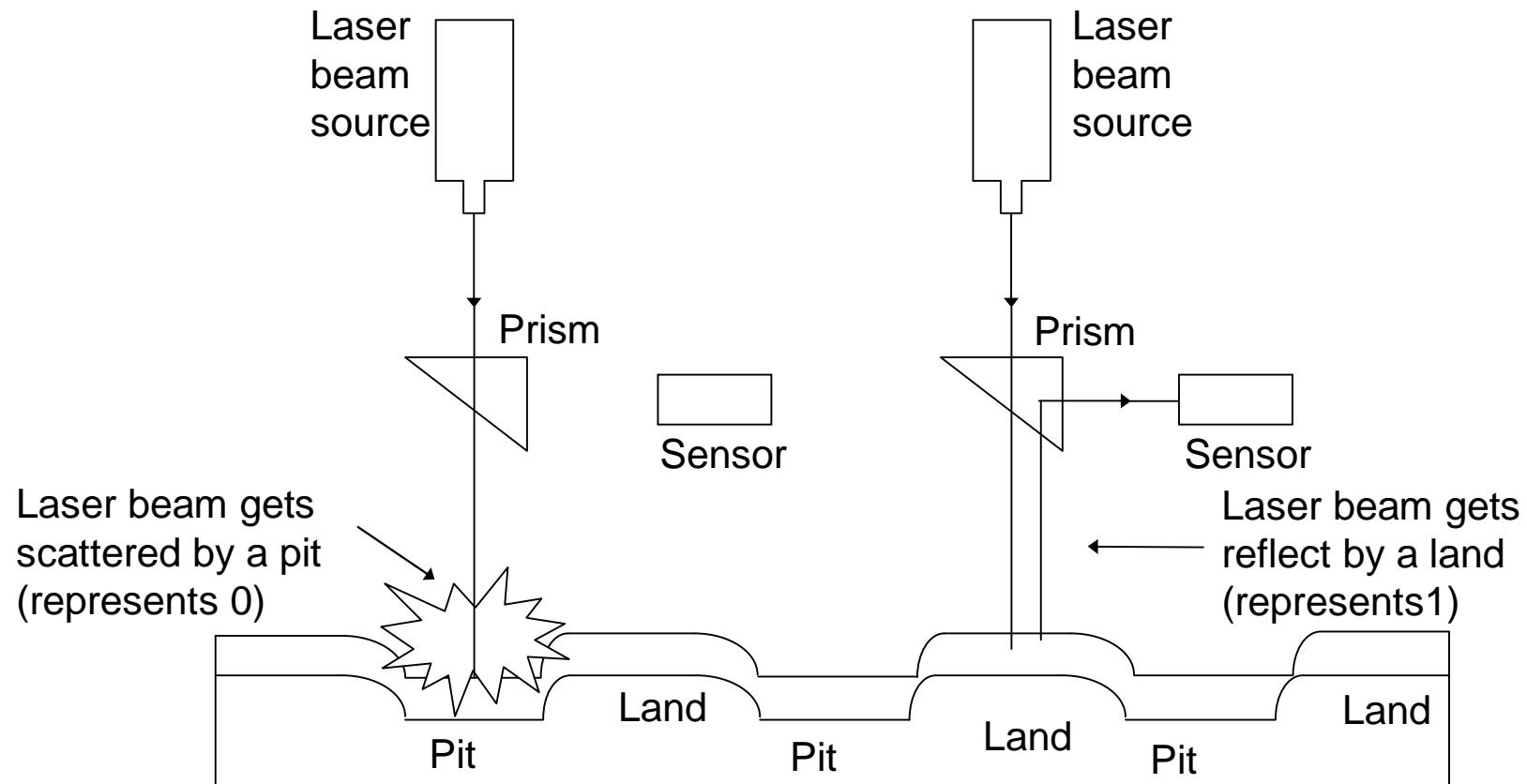
# Optical Disk – Storage Capacity

*Storage capacity of an optical disk*

$$\begin{aligned} &= \text{Number of sectors} \\ &\times \text{Number of bytes per sector} \end{aligned}$$

The most popular optical disk uses a disk of 5.25 inch diameter with storage capacity of around 650 Megabytes

# Optical Disk – Access Mechanism



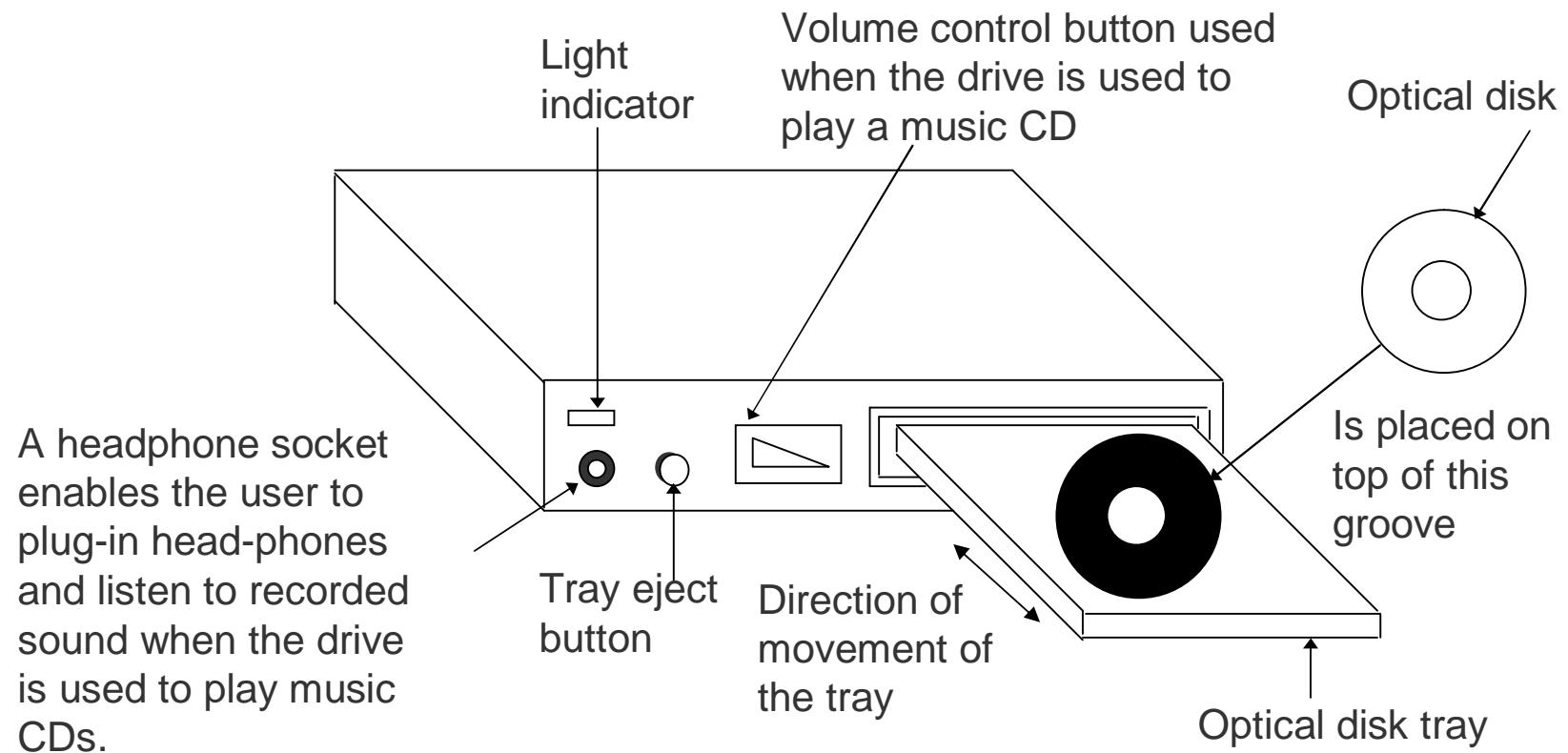
# Optical Disk – Access Time

- § With optical disks, each sector has the same length regardless of whether it is located near or away from the disk's center
- § Rotation speed of the disk must vary inversely with the radius. Hence, optical disk drives use a constant linear velocity (CLV) encoding scheme
- § Leads to slower data access time (larger access time) for optical disks than magnetic disks
- § Access times for optical disks are typically in the range of 100 to 300 milliseconds and that of hard disks are in the range of 10 to 30 milliseconds

# Optical Disk Drive

- § Uses laser beam technology for reading/writing of data
- § Has no mechanical read/write access arm
- § Uses a constant linear velocity (CLV) encoding scheme, in which the rotational speed of the disk varies inversely with the radius

# Optical Disk Drive



# Types of Optical Disks

The types of optical disks in use today are:

## CD-ROM

- § Stands for Compact Disk-Read Only Memory
- § Packaged as shiny, silver color metal disk of  $5\frac{1}{4}$  inch (12cm) diameter, having a storage capacity of about 650 Megabytes
- § Disks come pre-recorded and the information stored on them cannot be altered
- § Pre-stamped (pre-recorded) by their suppliers, by a process called *mastering*

(Continued on next slide)

# Types of Optical Disks

(Continued from previous slide..)

- § Provide an excellent medium to distribute large amounts of data in electronic form at low cost.
- § A single CD-ROM disk can hold a complete encyclopedia, or a dictionary, or a world atlas, or biographies of great people, etc
- § Used for distribution of electronic version of conference proceedings, journals, magazines, books, and multimedia applications such as video games
- § Used by software vendors for distribution of software to their customers

# Types of Optical Disks

## WORM Disk / CD-Recordable (CD-R)

- § Stands for Write Once Read Many. Data can be written only once on them, but can be read many times
- § Same as CD-ROM and has same storage capacity
- § Allow users to create their own CD-ROM disks by using a CD-recordable (CD-R) drive that can be attached to a computer as a regular peripheral device
- § Data to be recorded can be written on its surface in multiple recording sessions

(Continued on next slide)

# Types of Optical Disks

(Continued from previous slide..)

- § Sessions after the first one are always additive and cannot alter the etched/burned information of earlier sessions
- § Information recorded on them can be read by any ordinary CD-ROM drive
- § They are used for data archiving and for making a permanent record of data. For example, many banks use them for storing their daily transactions

# Types of Optical Disks

## CD-Read/Write (CD-RW)

- § Same as CD-R and has same storage capacity
- § Allow users to create their own CD-ROM disks by using a CD-recordable (CD-R) drive that can be attached to a computer as a regular peripheral device
- § Data to be recorded can be written on its surface in multiple recording sessions
- § Made of metallic alloy layer whose chemical properties are changed during burn and erase
- § Can be erased and written afresh

# Types of Optical Disks

## Digital Video / Versatile Disk (DVD)

- § Looks same as CD-ROM but has capacity of 4.7 GB or 8.5 GB
- § Designed primarily to store and distribute movies
- § Can be used for storage of large data
- § Allows storage of video in 4:3 or 16:9 aspect-ratios in MPEG-2 video format using NTSC or PAL resolution
- § Audio is usually Dolby® Digital (AC-3) or Digital Theater System (DTS) and can be either monaural or 5.1 Surround Sound

# Advantages of Optical Disks

- § The cost-per-bit of storage for optical disks is very low because of their low cost and enormous storage density.
- § The use of a single spiral track makes optical disks an ideal storage medium for reading large blocks of sequential data, such as music.
- § Optical disk drives do not have any mechanical read/write heads to rub against or crash into the disk surface. This makes optical disks a more reliable storage medium than magnetic tapes or magnetic disks.
- § Optical disks have a data storage life in excess of 30 years. This makes them a better storage medium for data archiving as compared to magnetic tapes or magnetic disks.

# Advantages of Optical Disks

- § As data once stored on an optical disk becomes permanent, danger of stored data getting inadvertently erased/overwritten is removed
- § Due to their compact size and light weight, optical disks are easy to handle, store, and port from one place to another
- § Music CDs can be played on a computer having a CD-ROM drive along with a sound board and speakers. This allows computer systems to be also used as music systems

# Limitations of Optical Disks

- § It is largely read-only (permanent) storage medium. Data once recorded, cannot be erased and hence the optical disks cannot be reused
- § The data access speed for optical disks is slower than magnetic disks
- § Optical disks require a complicated drive mechanism

# Uses of Optical Disks

- § For distributing large amounts of data at low cost
- § For distribution of electronic version of conference proceedings, journals, magazines, books, product catalogs, etc
- § For distribution of new or upgraded versions of software products by software vendors

(Continued on next slide)

# Uses of Optical Disks

(Continued from previous slide..)

- § For storage and distribution of a wide variety of multimedia applications
- § For archiving of data, which are not used frequently, but which may be used once in a while
- § WORM disks are often used by end-user companies to make permanent storage of their own proprietary information

# Memory Storage Devices

## Flash Drive (Pen Drive)

- § Relatively new secondary storage device based on flash memory, enabling easy transport of data from one computer to another
- § Compact device of the size of a pen, comes in various shapes and stylish designs and may have different added features
- § Plug-and-play device that simply plugs into a USB (Universal Serial Bus) port of a computer, treated as removable drive
- § Available storage capacities are 8MB, 16MB, 64MB, 128MB, 256MB, 512MB, 1GB, 2GB, 4GB, and 8GB

# Memory Storage Devices

## Memory Card (SD/MMC)

- § Similar to Flash Drive but in card shape
- § Plug-and-play device that simply plugs into a port of a computer, treated as removable drive
- § Useful in electronic devices like Camera, music player
- § Available storage capacities are 8MB, 16MB, 64MB, 128MB, 256MB, 512MB, 1GB, 2GB, 4GB, and 8GB

# Mass Storage Devices

- § As the name implies, these are storage systems having several trillions of bytes of data storage capacity
- § They use multiple units of a storage media as a single secondary storage device
- § The three commonly used types are:
  1. *Disk array*, which uses a set of magnetic disks
  2. *Automated tape library*, which uses a set of magnetic tapes
  3. *CD-ROM Jukebox*, which uses a set of CD-ROMs
- § They are relatively slow having average access times in seconds

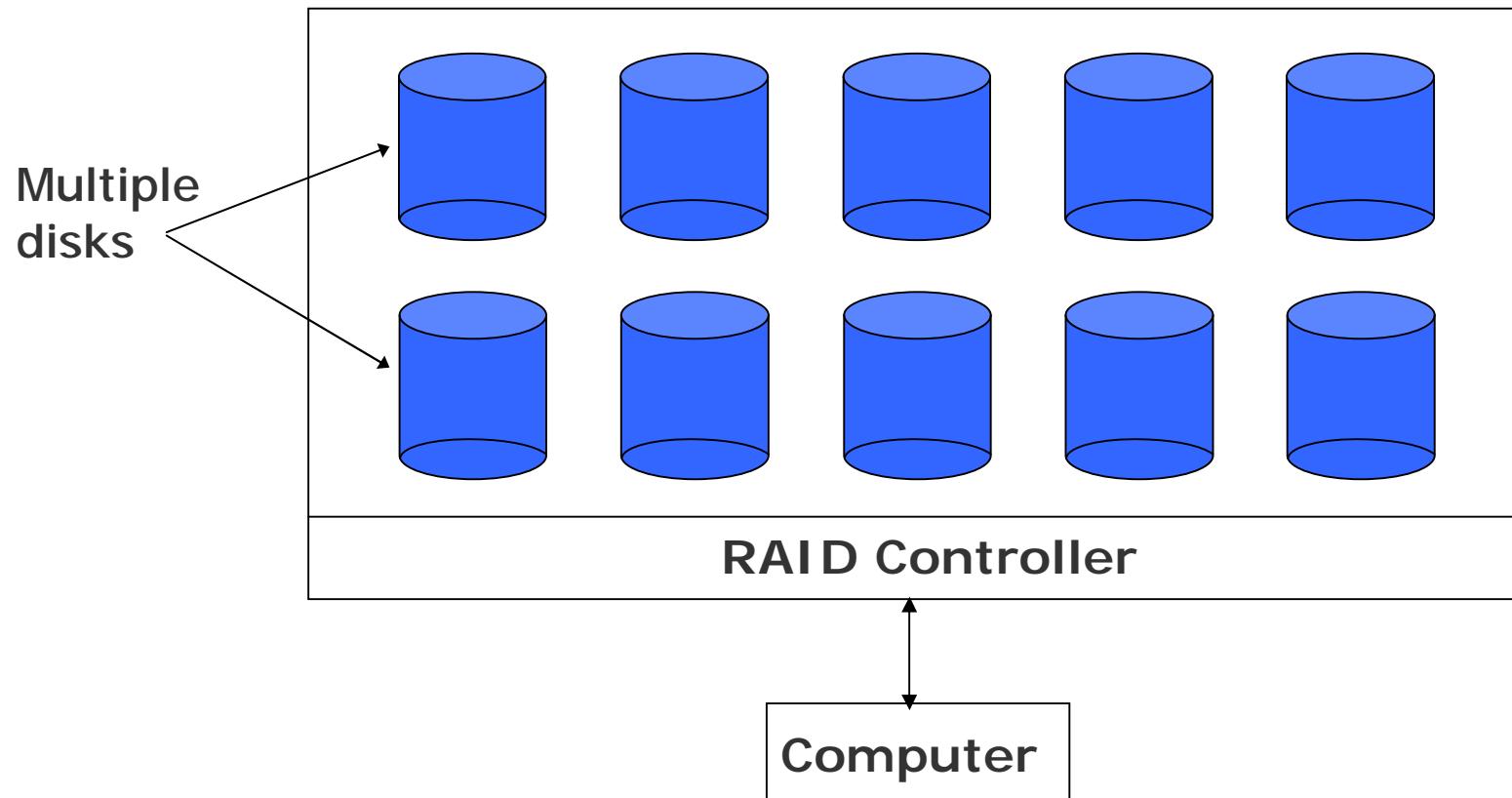
# Disk Array

- § Set of hard disks and hard disk drives with a controller mounted in a single box, forming a single large storage unit
- § It is commonly known as a *RAID (Redundant Array of Inexpensive Disks)*
- § As a secondary storage device, provides enhanced storage capacity, enhanced performance, and enhanced reliability

# Disk Array

- § Enhanced storage capacity is achieved by using multiple disks
- § Enhanced performance is achieved by using parallel data transfer technique from multiple disks
- § Enhanced reliability is achieved by using techniques such as mirroring or striping
- § In *mirroring*, the system makes exact copies of files on two hard disks
- § In *striping*, a file is partitioned into smaller parts and different parts of the file are stored on different disks

# A RAID Unit



# Automated Tape Library

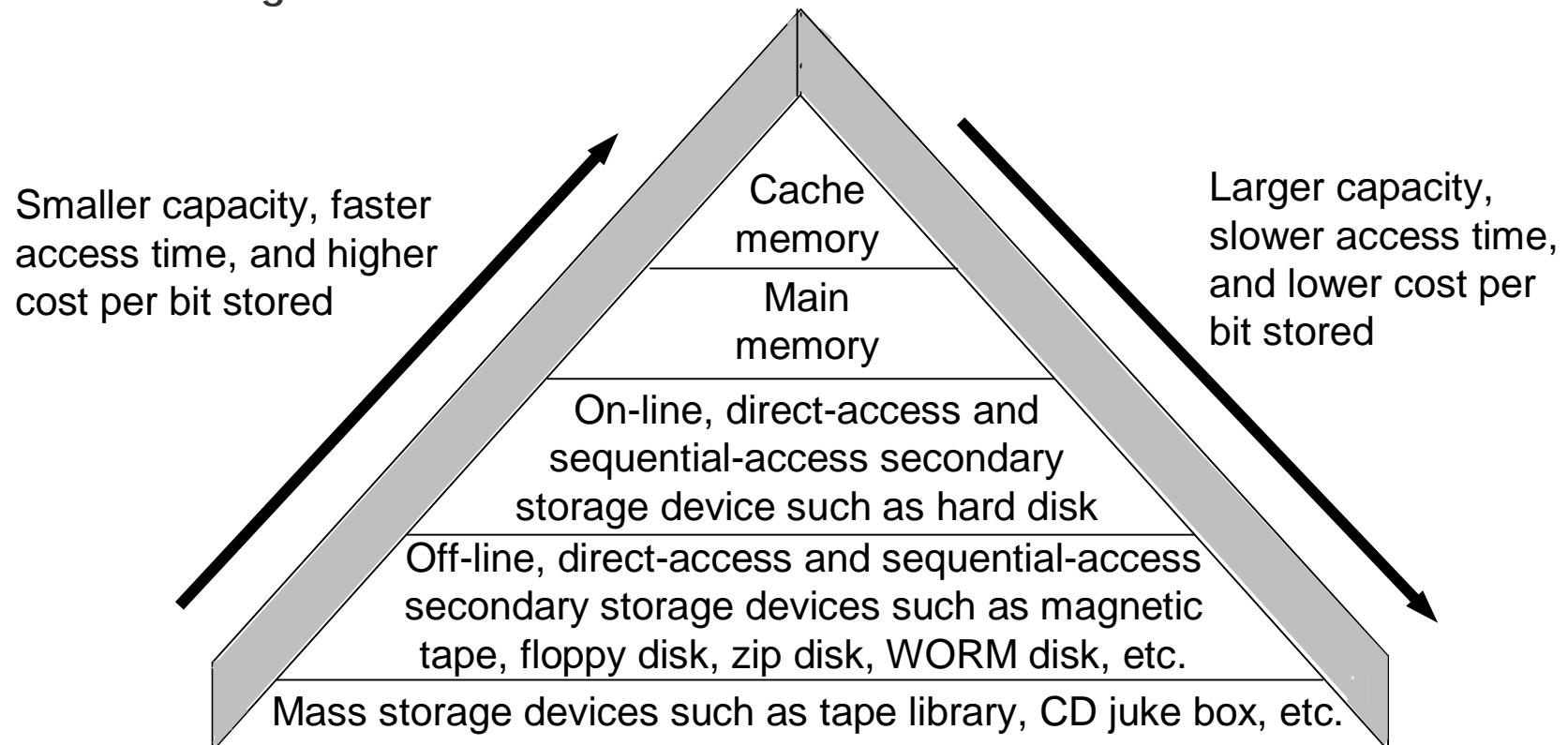
- § Set of magnetic tapes and magnetic tape drives with a controller mounted in a single box, forming a single large storage unit
- § Large tape library can accommodate up to several hundred high capacity magnetic tapes bringing the storage capacity of the storage unit to several terabytes
- § Typically used for data archiving and as on-line data backup devices for automated backup in large computer centers

# CD-ROM Jukebox

- § Set of CD-ROMs and CD-ROM drives with a controller mounted in a single box, forming a single large storage unit
- § Large CD-ROM jukebox can accommodate up to several hundred CD-ROM disks bringing the storage capacity of the storage unit to several terabytes
- § Used for archiving read-only data in such applications as on-line museums, on-line digital libraries, on-line encyclopedia, etc

# Storage Hierarchy

As a single type of storage is not superior in speed of access, capacity, and cost, most computer systems make use of a hierarchy of storage technologies as shown below.



# Key Words/Phrases

- |                              |                           |
|------------------------------|---------------------------|
| § Automated tape library     | § Floppy disk             |
| § Auxiliary memory           | § Hard disk               |
| § Block                      | § Inter-block gap (IBG)   |
| § Blocking                   | § Inter-record gap (IRG)  |
| § Blocking factory           | § Land                    |
| § CD-ROM                     | § Latency                 |
| § CD-ROM jukebox             | § Magnetic disk           |
| § Check bit                  | § Magnetic tape           |
| § Cylinder                   | § Magnetic tape drive     |
| § Data transfer rate         | § Mass storage devices    |
| § Direct access device       | § Master file             |
| § Disk array                 | § Odd parity              |
| § Disk controller            | § Off-line storage        |
| § Disk drive                 | § On-line storage         |
| § Disk formatting            | § Optical disk            |
| § Disk pack                  | § Parallel representation |
| § DVD                        | § Parity bit              |
| § Even parity                | § Pit                     |
| § File Allocation Tube (FAT) |                           |

(Continued on next slide)

# Key Words/Phrases

(Continued from previous slide..)

- § QIC Standard
- § Record
- § Redundant Array of Inexpensive Disks (RAID)
- § Secondary storage
- § Sector
- § Seek time
- § Sequential access device
- § Storage hierarchy
- § Tape controller
- § Track
- § Transaction file
- § Winchester disk
- § WORM disk
- § Zip disk

# Chapter 09

# Input-Output Devices

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

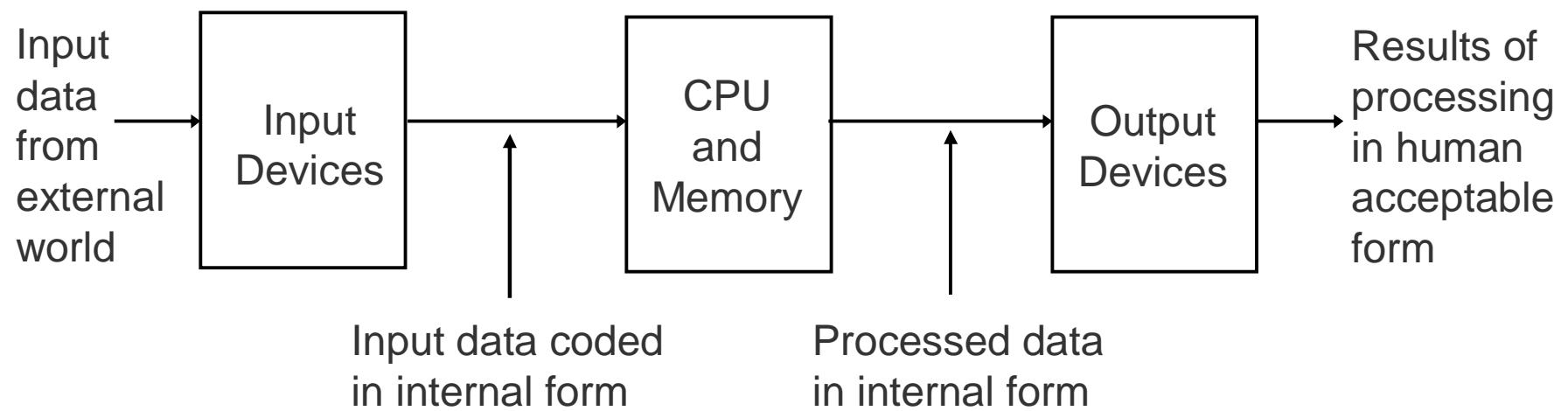
In this chapter you will learn about:

- § Input/Output (I/O) devices
- § Commonly used input devices
- § Commonly used output devices
- § Other concepts related to I/O devices

# I/O Devices

- § Provide means of communication between a computer and outer world
- § Also known as peripheral devices because they surround the CPU and memory of a computer system
- § Input devices are used to enter data from the outside world into primary storage
- § Output devices supply results of processing from primary storage to users

# Role of I/O Devices



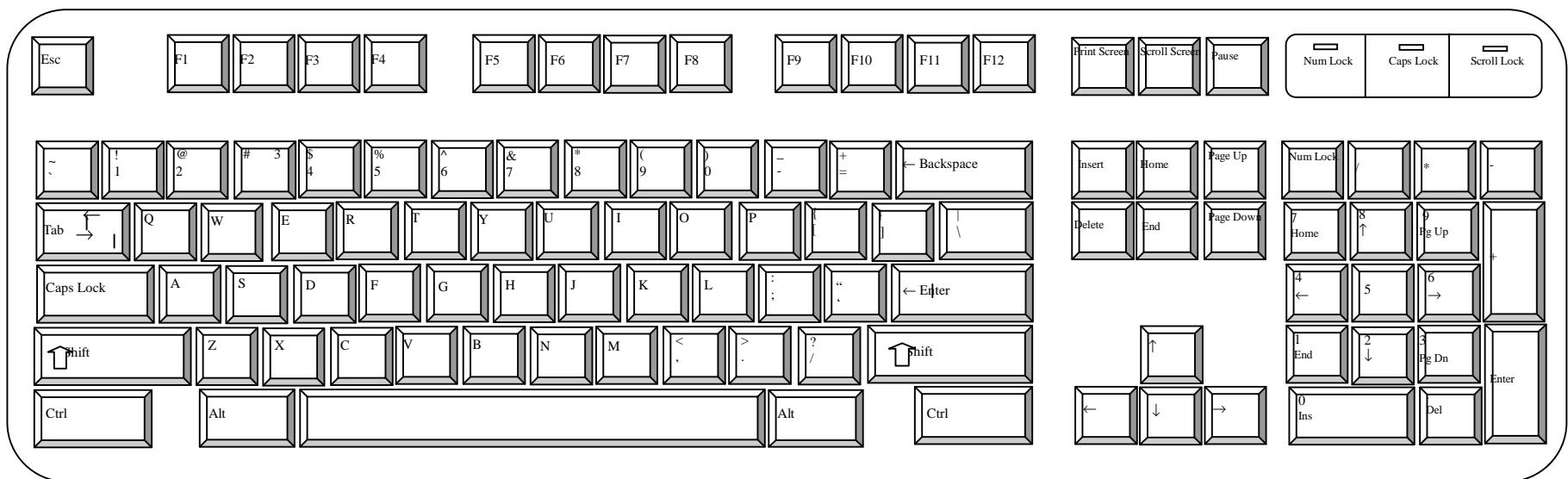
# Commonly Used Input Devices

- § Keyboard devices
- § Point-and-draw devices
- § Data scanning devices
- § Digitizer
- § Electronic cards based devices
- § Speech recognition devices
- § Vision based devices

# Keyboard Devices

- § Allow data entry into a computer system by pressing a set of keys (labeled buttons) neatly mounted on a keyboard connected to a computer system
- § 101-keys QWERTY keyboard is most popular

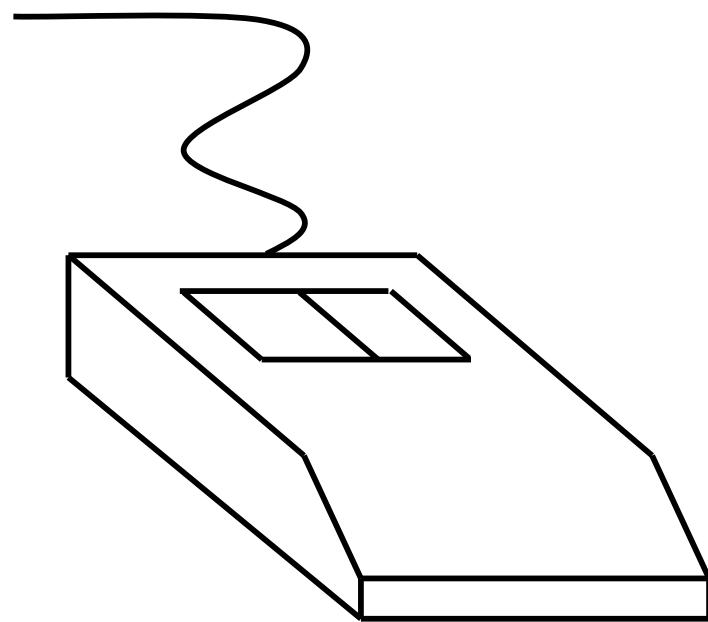
# The Layout of Keys on a QWERTY Keyboard



# Point-and-Draw Devices

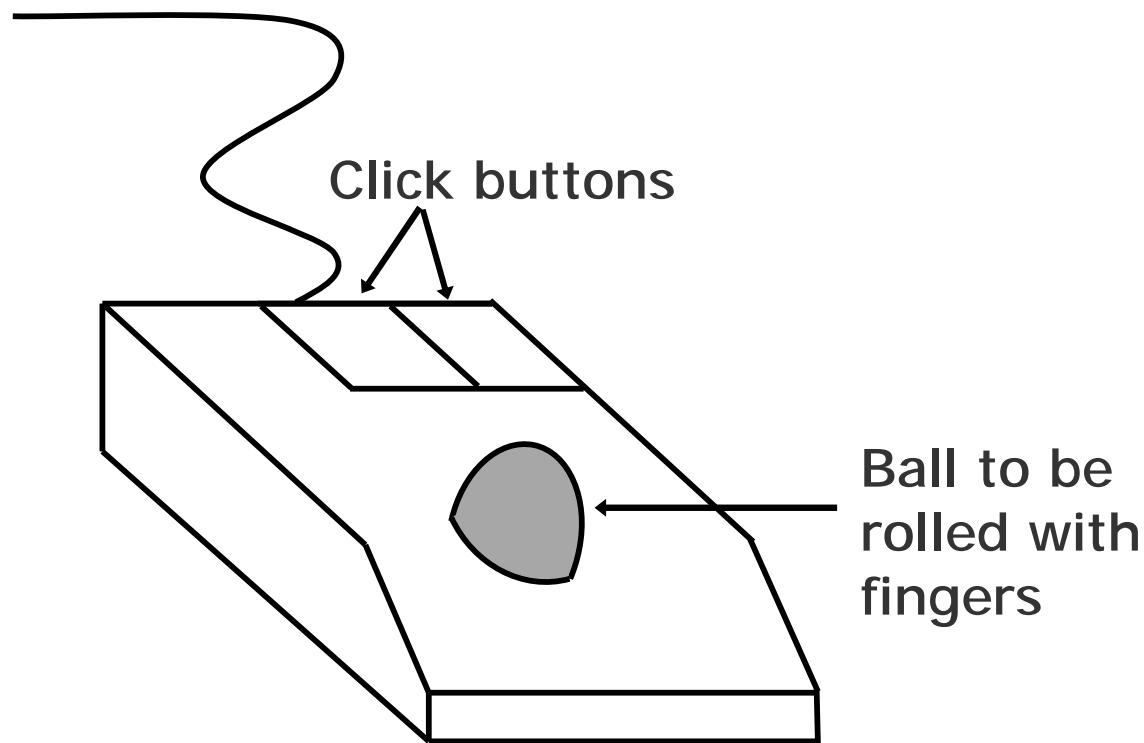
- § Used to rapidly point to and select a graphic icon or menu item from multiple options displayed on the *Graphical User Interface (GUI)* of a screen
- § Used to create graphic elements on the screen such as lines, curves, and freehand shapes
- § Some commonly used point-and-draw devices are mouse, track ball, joy stick, light pen, and touch screen

# Mouse



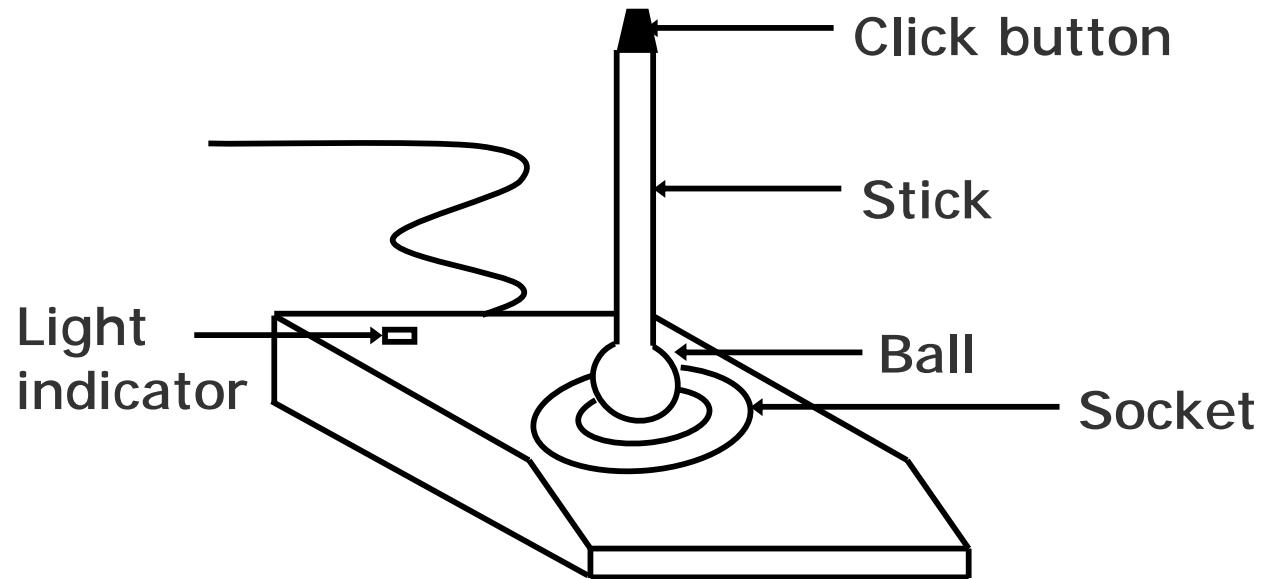
Commonly used in personal computers and workstations

# Trackball



Commonly used in laptop (notebook) computers

# Joystick



Commonly used for video games, flight simulators, training simulators, and for controlling industrial robots

# Electronic Pen

- § Pen-based point-and-draw device
- § Used to directly point with it on the screen to select menu items or icons or directly draw graphics on the screen
- § Can write with it on a special pad for direct input of written information to a system
- § Pressure on tip of a side button is used to cause same action as *right-button-click* of a mouse

# Touch Screen

- § Most simple, intuitive, and easiest to learn of all input devices
- § Enables users to choose from available options by simply touching with their finger the desired icon or menu item displayed on the screen
- § Most preferred human-computer interface used in *information kiosks* (unattended interactive information systems such as automatic teller machine or ATM)

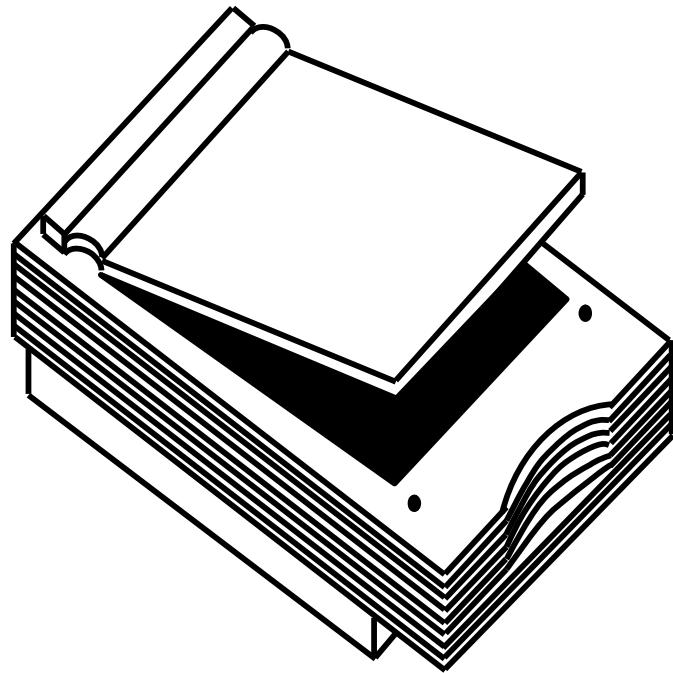
# Data Scanning Devices

- § Input devices that enable direct data entry into a computer system from source documents
- § Eliminate the need to key in text data into the computer
- § Due to reduced human effort in data entry, they improve data accuracy and also increase the timeliness of the information processed
- § Demand high quality of input documents
- § Some data scanning devices are also capable of recognizing marks or characters
- § Form design and ink specification usually becomes more critical for accuracy

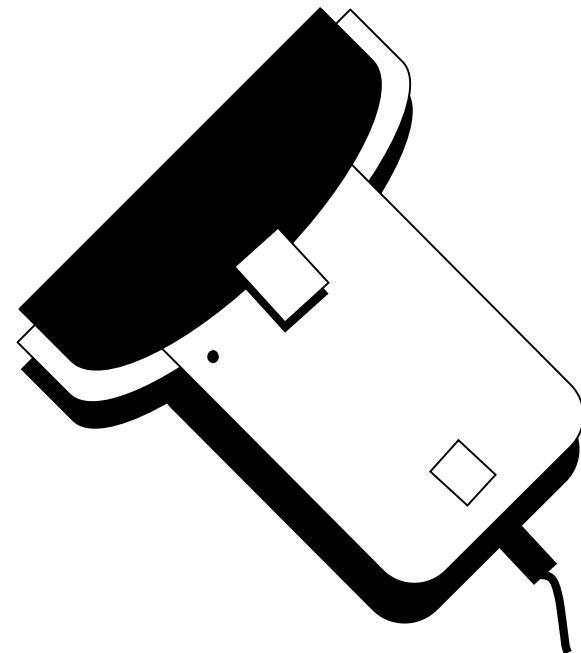
# Image Scanner

- § Input device that translates paper documents into an electronic format for storage in a computer
- § Electronic format of a scanned image is its bit map representation
- § Stored image can be altered or manipulated with an image-processing software

## Two Common Types of Image Scanners



A flat-bed scanner



A hand-held scanner

# Optical Character Recognition (OCR) Device

- § Scanner equipped with a character recognition software (called OCR software) that converts the bit map images of characters to equivalent ASCII codes
- § Enables word processing of input text and also requires less storage for storing the document as text rather than an image
- § OCR software is extremely complex because it is difficult to make a computer recognize an unlimited number of typefaces and fonts
- § Two standard OCR fonts are OCR-A (American standard) and OCR-B (European standard)

## Optical Mark Reader (OMR)

- § Scanner capable of recognizing a pre-specified type of mark by pencil or pen
- § Very useful for grading tests with objective type questions, or for any input data that is of a choice or selection nature
- § Technique used for recognition of marks involves focusing a light on the page being scanned and detecting the reflected light pattern from the marks

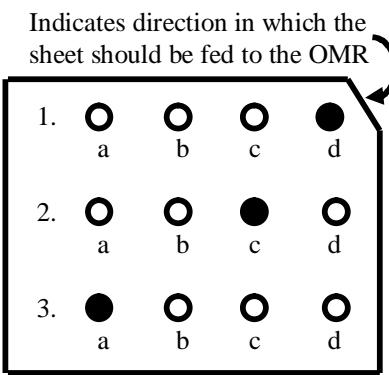
# Sample Use of OMR

For each question, four options are given out of which only one is correct. Choose the correct option and mark your choice against the corresponding question number in the given answer sheet by darkening the corresponding circle with a lead pencil.

1. The binary equivalent of decimal 4 is:
  - a) 101
  - b) 111
  - c) 001
  - d) 100
  
2. The full form of CPU is:
  - a) Cursor Positioning Unit
  - b) Central Power Unit
  - c) Central Processing Unit
  - d) None of the above
  
3. Which is the largest unit of storage among the following:
  - a) Terabyte
  - b) Kilobyte
  - c) Megabyte
  - d) Gigabyte

(a) Question sheet

A sample use of OMR for grading tests with objective type questions



(b) Pre-printed answer sheet

# Bar-code Reader

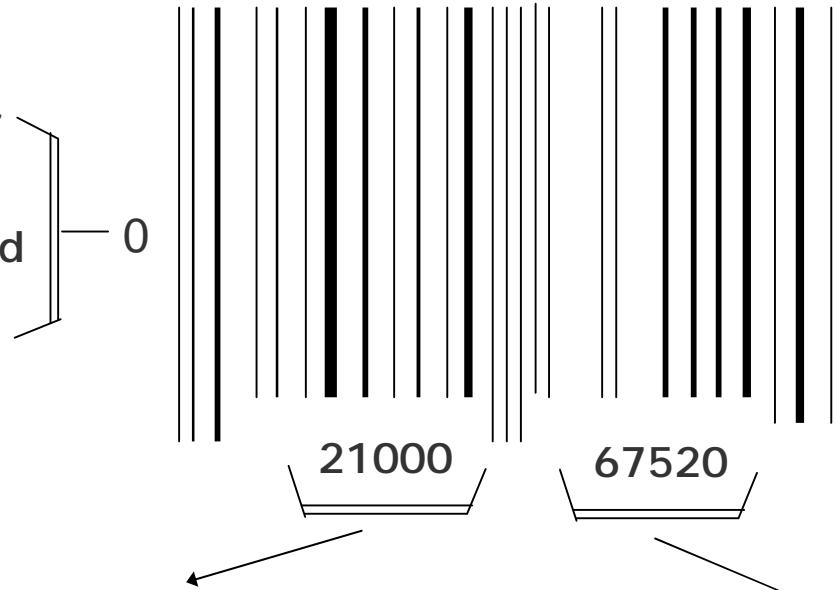
- § Scanner used for reading (decoding) bar-coded data
- § Bar codes represent alphanumeric data by a combination of adjacent vertical lines (bars) by varying their width and the spacing between them
- § Scanner uses laser-beam to stroke across pattern of bar code. Different patterns of bars reflect the beam in different ways sensed by a light-sensitive detector
- § Universal Product Code (UPC) is the most widely known bar coding system

# An Example of UPC Bar Code

Product category character

0 – grocery products

3 – drugs and health related products, etc.



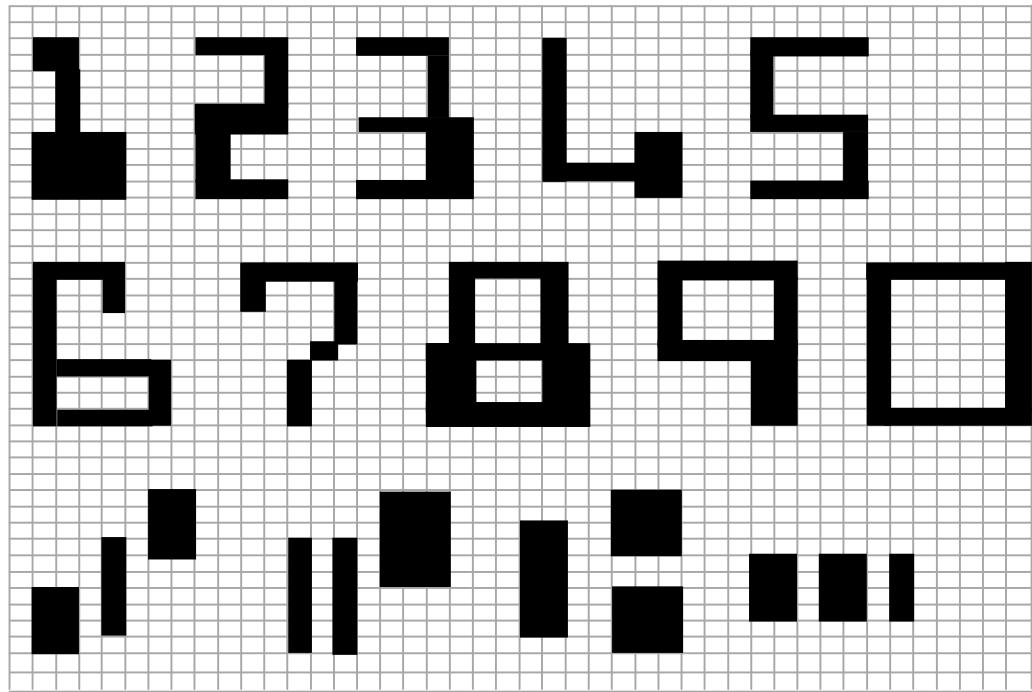
Manufacturer/supplier identification number

Specific product code number

## Magnetic-Ink Character Recognition (MICR)

- § MICR is used by banking industry for faster processing of large volume of cheques
- § Bank's identification code (name, branch, etc.), account number and cheque number are pre-printed (encoded) using characters from a special character set on all cheques
- § Special ink is used that contains magnetizable particles of iron oxide
- § MICR reader-sorter reads data on cheques and sorts them for distribution to other banks or for further processing

# MICR Character Set (E13B Font)

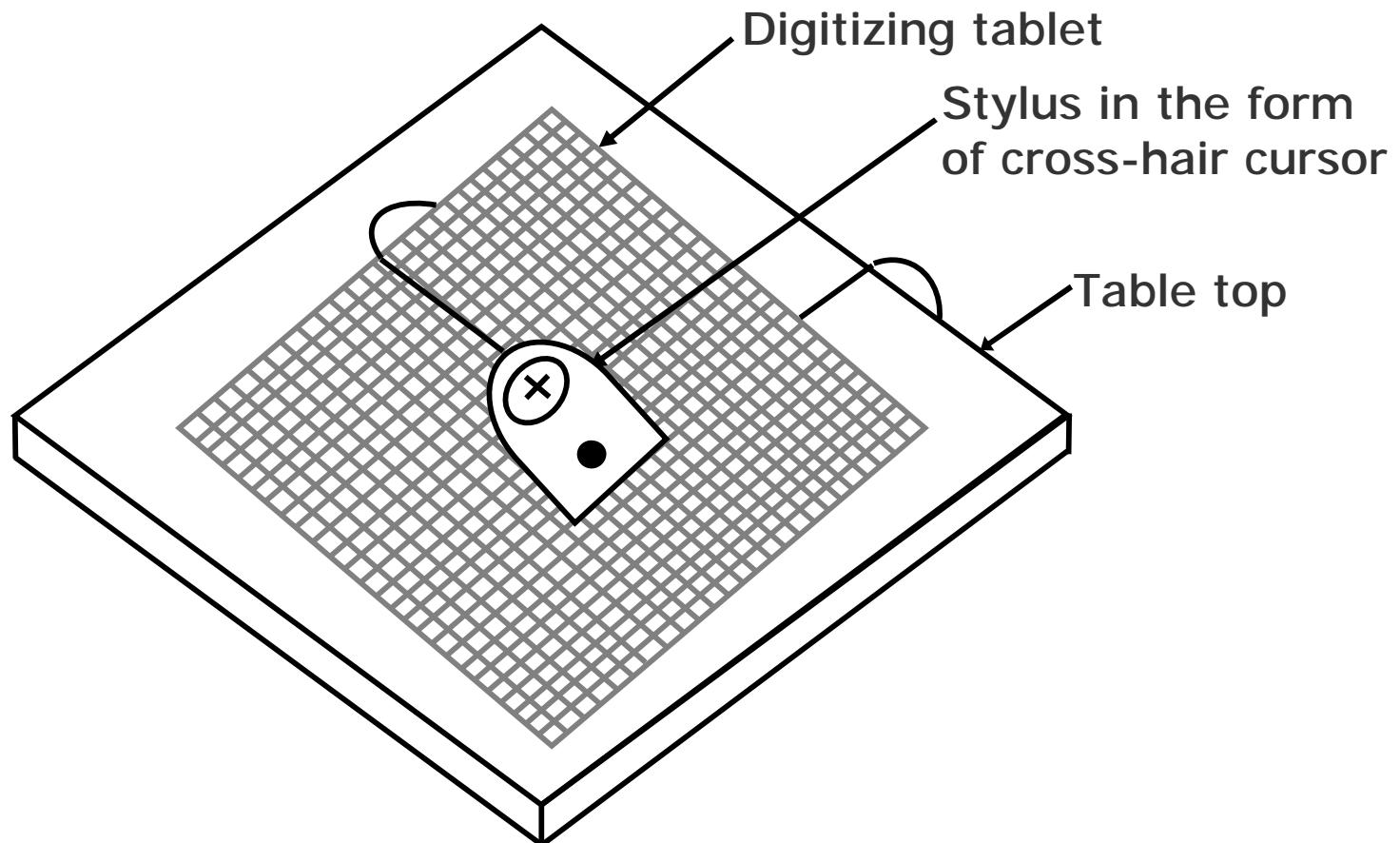


- § It consists of numerals 0 to 9 and four special characters
- § MICR is not adopted by other industries because it supports only 14 symbols

# Digitizer

- § Input device used for converting (digitizing) pictures, maps and drawings into digital form for storage in computers
- § Commonly used in the area of Computer Aided Design (CAD) by architects and engineers to design cars, buildings medical devices, robots, mechanical parts, etc.
- § Used in the area of Geographical Information System (GIS) for digitizing maps available in paper form

# A Digitizer



# Electronic-card Reader

- § Electronic cards are small plastic cards having encoded data appropriate for the application for which they are used
- § Electronic-card reader (normally connected to a computer) is used to read data encoded on an electronic card and transfer it to the computer for further processing
- § Used together as a means of direct data entry into a computer system
- § Used by banks for use in automatic teller machines (ATMs) and by organizations for controlling access of employees to physically secured areas

# Speech Recognition Devices

- § Input device that allows a person to input data to a computer system by speaking to it
- § Today's speech recognition systems are limited to accepting few words within a relatively small domain and can be used to enter only limited kinds and quantities of data

# Types of Speech Recognition Systems

(Continued from previous slide..)

- § *Single word recognition systems* can recognize only a single spoken words, such as YES, NO, MOVE, STOP, at a time. Speaker-independent systems are mostly of this type
- § *Continuous speech recognition systems* can recognize spoken sentences, such as MOVE TO THE NEXT BLOCK. Such systems are normally speaker-dependent

# Uses of Speech Recognition Systems

- § For inputting data to a computer system by a person in situations where his/her hands are busy, or his/her eyes must be fixed on a measuring instrument or some other object
- § For data input by dictation of long text or passage for later editing and review
- § For authentication of a user by a computer system based on voice input
- § For limited use of computers by individuals with physical disabilities

# Vision-Input Systems

- § Allow computer to accept input just by seeing an object.
- § Input data is normally an object's shape and features in the form of an image
- § Mainly used today in factories for designing industrial robots that are used for quality-control and assembly processes

# Commonly Used Output Devices

- § Monitors
- § Printers
- § Plotters
- § Screen image projector
- § Voice response systems

# Types of Output

## § Soft-copy output

- § Not produced on a paper or some material that can be touched and carried for being shown to others
- § Temporary in nature and vanish after use
- § Examples are output displayed on a terminal screen or spoken out by a voice response system

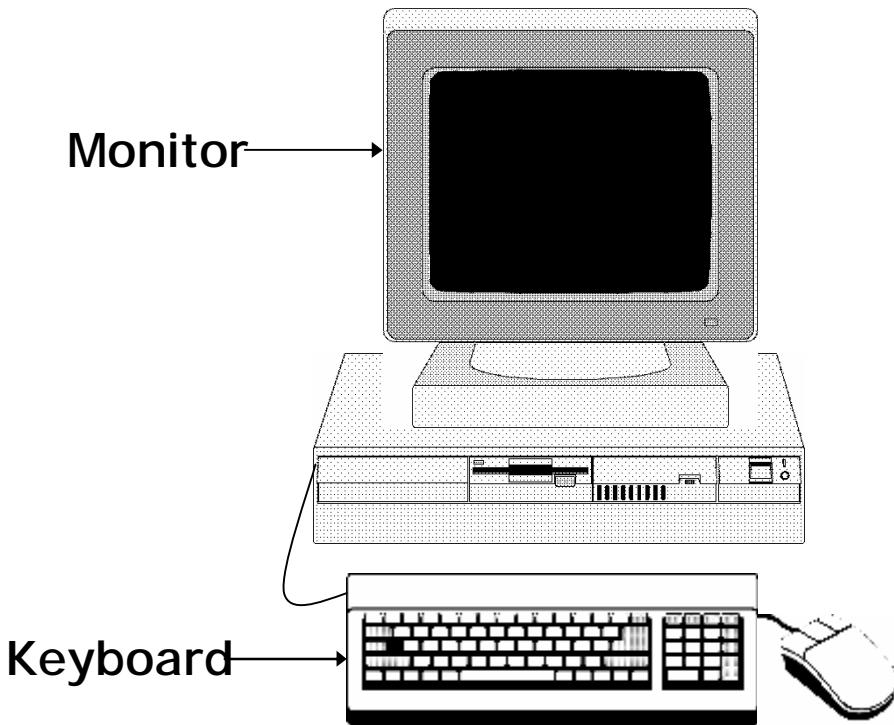
## § Hard-copy output

- § Produced on a paper or some material that can be touched and carried for being shown to others
- § Permanent in nature and can be kept in paper files or can be looked at a later time when the person is not using the computer
- § Examples are output produced by printers or plotters on paper

# Monitors

- § Monitors are the most popular output devices used for producing soft-copy output
- § Display the output on a television like screen
- § Monitor associated with a keyboard is called a video display terminal (VDT). It is the most popular I/O device

# Monitors



A video display terminal consists of a monitor and a keyboard

# Types of Monitors

- § Cathode-ray-tube (CRT) monitors look like a television and are normally used with non-portable computer systems
- § Flat-panel monitors are thinner and lighter and are commonly used with portable computer systems like notebook computers. Now they are also used with non-portable desktop computer systems because they occupy less table space.

# Printers

Most common output devices for producing hard-copy output

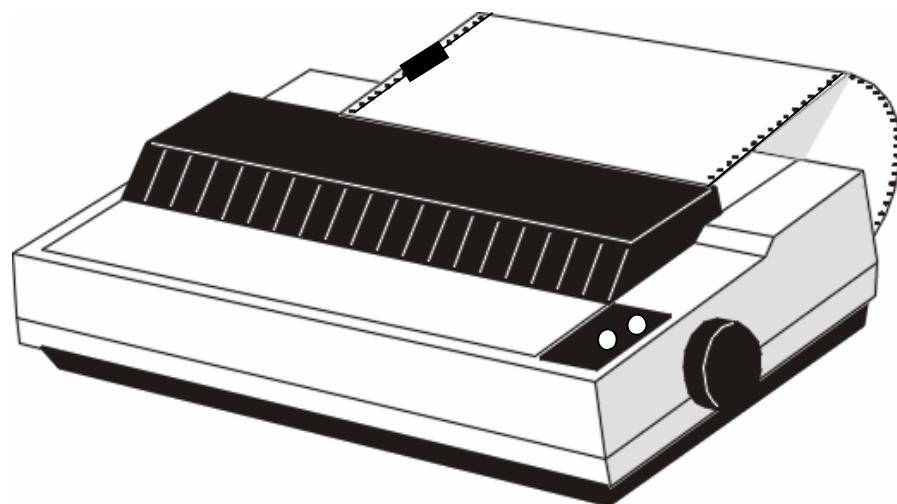
# Dot-Matrix Printers

- § Character printers that form characters and all kinds of images as a pattern of dots
- § Print many special characters, different sizes of print and graphics such as charts and graphs
- § Impact printers can be used for generating multiple copies by using carbon paper or its equivalent
- § Slow, with speeds usually ranging between 30 to 600 characters per second
- § Cheap in both initial cost and cost of operation

# Formation of Characters as a pattern of dots

ABCDEFGHIJKLMNO  
PQRSTUVWXYZ  
0123456789-.,  
&/\${\*#%@=(+)

# A Dot Matrix Printer



# Inkjet Printers

- § Character printers that form characters and all kinds of images by spraying small drops of ink on to the paper
- § Print head contains up to 64 tiny nozzles that can be selectively heated up in a few micro seconds by an integrated circuit register
- § To print a character, the printer selectively heats the appropriate set of nozzles as the print head moves horizontally
- § Can print many special characters, different sizes of print, and graphics such as charts and graphs

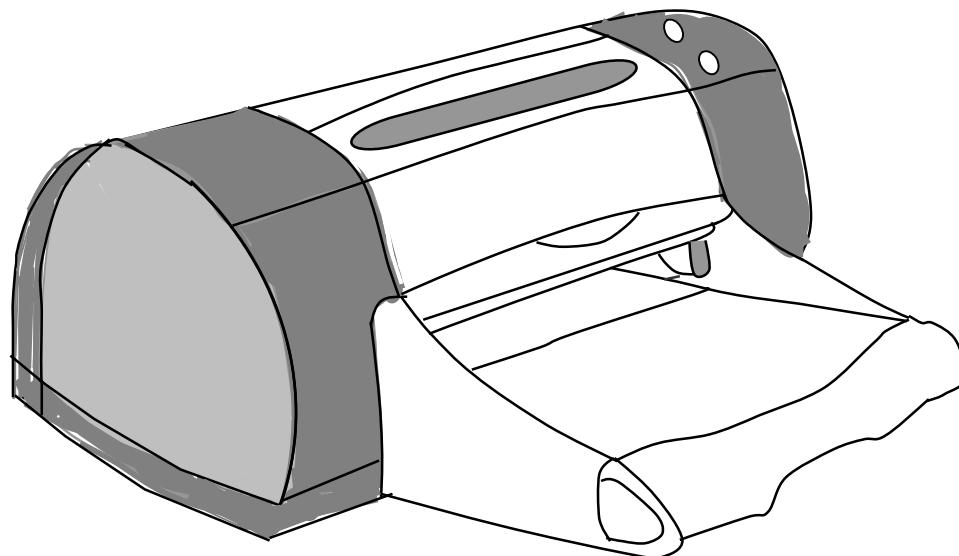
(Continued on next slide)

# Inkjet Printers

(Continued from previous slide..)

- § Non-impact printers. Hence, they cannot produce multiple copies of a document in a single printing
- § Can be both monochrome and color
- § Slower than dot-matrix printers with speeds usually ranging between 40 to 300 characters per second
- § More expensive than a dot-matrix printer

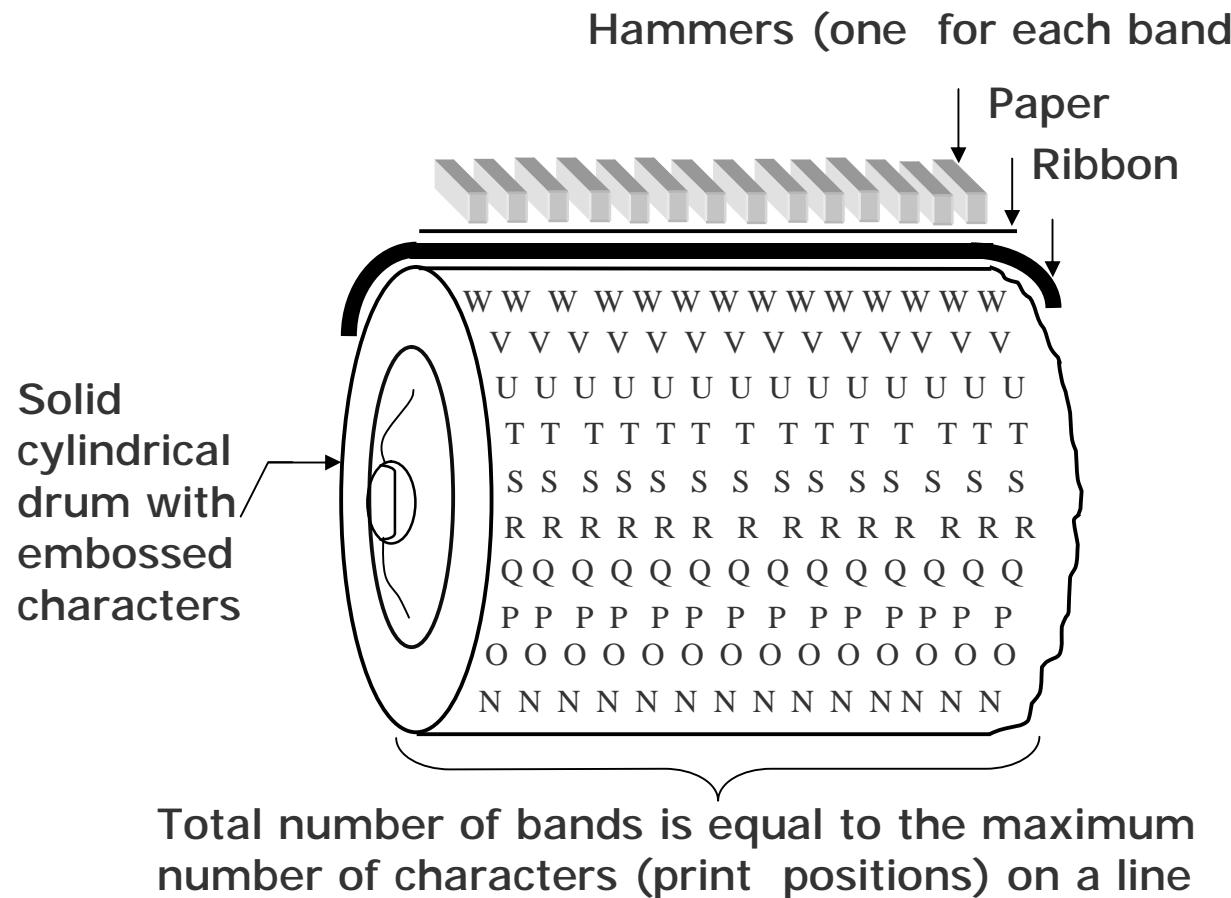
# An Inkjet Printers



# Drum Printers

- § Line printers that print one line at a time
- § Have a solid cylindrical drum with characters embossed on its surface in the form of circular bands
- § Set of hammers mounted in front of the drum in such a manner that an inked ribbon and paper can be placed between the hammers and the drum
- § Can only print a pre-defined set of characters in a pre-defined style that is embossed on the drum
- § Impact printers and usually monochrome
- § Typical speeds are in the range of 300 to 2000 lines per minute

# Printing Mechanism of a Drum Printer



# Chain/Band Printers

- § Line printers that print one line at a time
- § Consist of a metallic chain/band on which all characters of the character set supported by the printer are embossed
- § Also have a set of hammers mounted in front of the chain/band in such a manner that an inked ribbon and paper can be placed between the hammers and the chain/band

# Chain/Band Printers

- § Can only print pre-defined sets of characters that are embossed on the chain/band used with the printer
- § Cannot print any shape of characters, different sizes of print, and graphics such as charts and graphs
- § Are impact printers and can be used for generating multiple copies by using carbon paper or its equivalent
- § Are usually monochrome
- § Typical speeds are in the range of 400 to 3000 lines per minute

# Laser Printers

- § Page printers that print one page at a time
- § Consist of a laser beam source, a multi-sided mirror, a photoconductive drum and toner (tiny particles of oppositely charged ink)
- § To print a page, the laser beam is focused on the electro statically charged drum by the spinning multi-sided mirror
- § Toner sticks to the drum in the places the laser beam has charged the drum's surface.
- § Toner is then permanently fused on the paper with heat and pressure to generate the printer output
- § Laser printers produce very high quality output having resolutions in the range of 600 to 1200 dpi

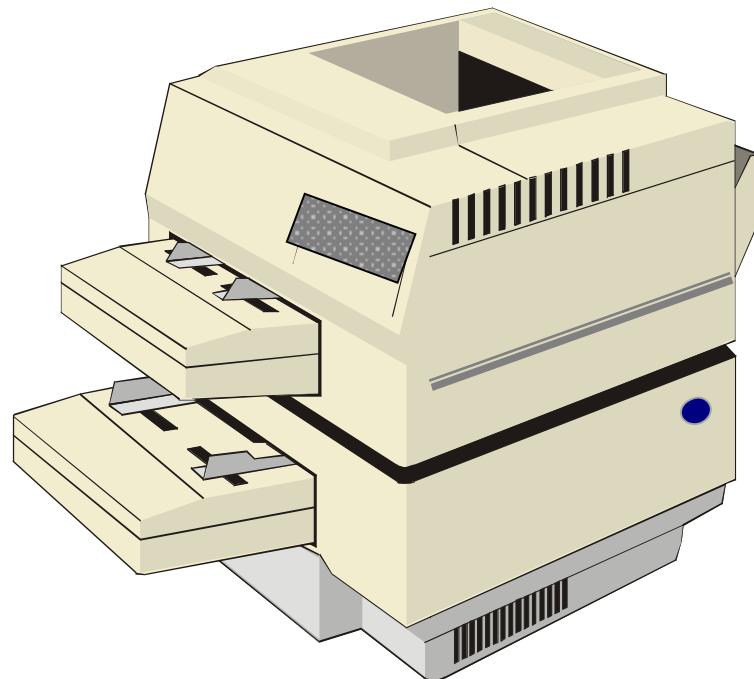
(Continued on next slide)

# Laser Printers

(Continued from previous slide..)

- § Can print many special characters, different sizes of print, and graphics such as charts and graphs
- § Are non-impact printers
- § Most laser printers are monochrome, but color laser printers are also available
- § Low speed laser printers can print 4 to 12 pages per minute. Very high-speed laser printers can print 500 to 1000 pages per minute
- § More expensive than other printers

# A Laser Printers



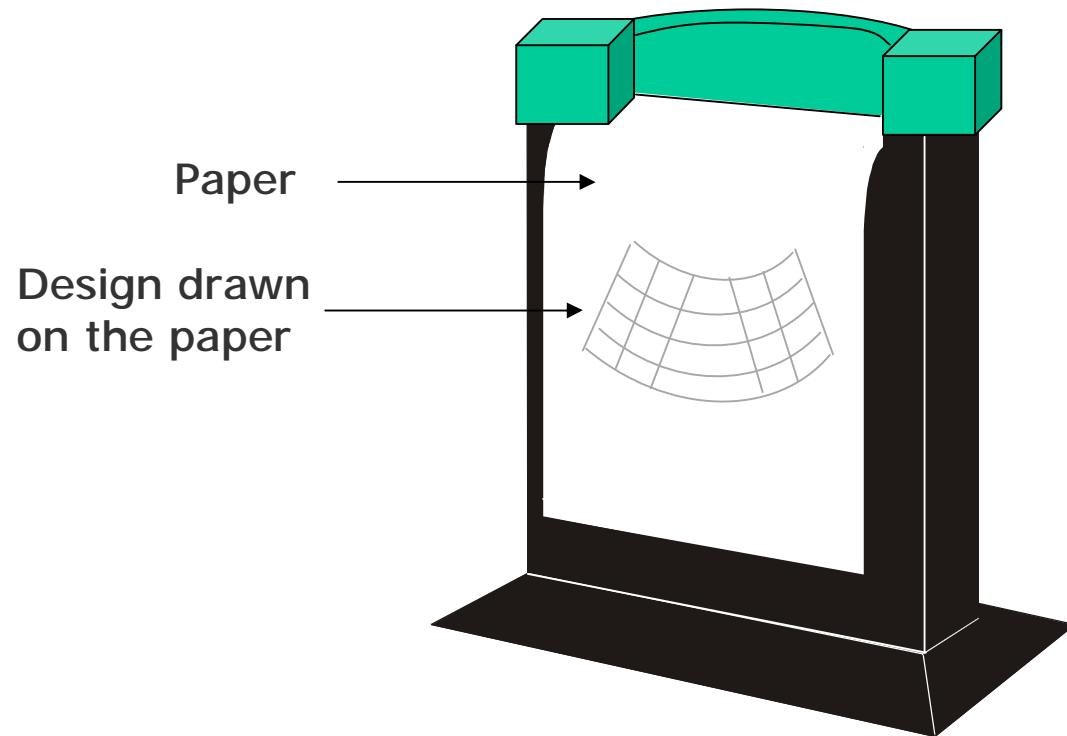
# Plotters

§ Plotters are an ideal output device for architects, engineers, city planners, and others who need to routinely generate high-precision, hard-copy graphic output of widely varying sizes

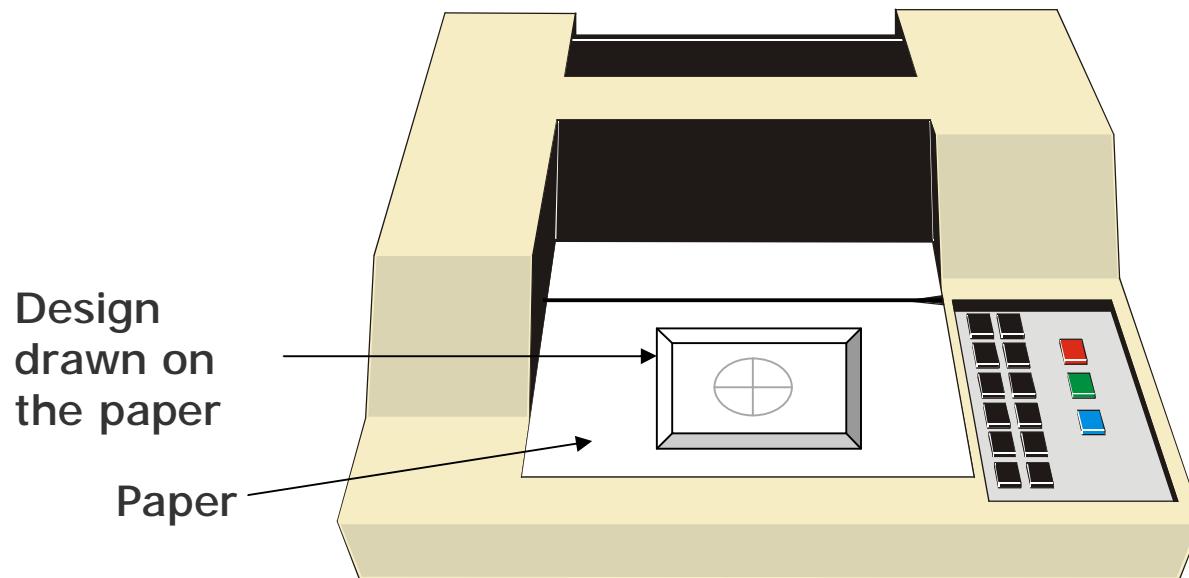
§ Two commonly used types of plotters are:

- *Drum plotter*, in which the paper on which the design has to be made is placed over a drum that can rotate in both clockwise and anti-clockwise directions
- *Flatbed plotter*, in which the paper on which the design has to be made is spread and fixed over a rectangular flatbed table

# A Drum Plotter



# A Flatbed Plotter



# Screen Image Projector

- § An output device that can be directly plugged to a computer system for projecting information from a computer on to a large screen
- § Useful for making presentations to a group of people with direct use of a computer
- § Full-fledged multimedia presentation with audio, video, image, and animation can be prepared and made using this facility

# Voice Response Systems

- § Voice response system enables a computer to talk to a user
- § Has an audio-response device that produces audio output
- § Such systems are of two types:
  - § Voice reproduction systems
  - § Speech synthesizers

(Continued on next slide)

# Voice Reproduction Systems

(Continued from previous slide..)

- § Produce audio output by selecting an appropriate audio output from a set of pre-recorded audio responses
- § Applications include audio help for guiding how to operate a system, automatic answering machines, video games, etc.

# Speech Synthesizers

- § Converts text information into spoken sentences
- § Used for applications such as:
  - § Reading out text information to blind persons
  - § Allowing those persons who cannot speak to communicate effectively
  - § Translating an entered text into spoken words in a selected language

# Key Words/Phrases

- |                            |   |
|----------------------------|---|
| § Bard code reader         | § Information Kiosk                         |
| § Cathode Ray Tube (CRT)   | § Inkjet printer                            |
| § Chain/Band printer       | § Input/Output device                       |
| § Data scanning device     | § Joystick                                  |
| § Digitizer                | § Keyboard device                           |
| § Digitizing tablet        | § Laser printer                             |
| § Dot-Matrix printer       | § Magnetic-Ink Character Recognition (MICR) |
| § Drum plotter             | § Monitor                                   |
| § Drum printer             | § Mouse                                     |
| § Electronic card reader   | § Optical Character Recognition (OCR)       |
| § Electronic Pen           | § Optical Mark Reader (OMR)                 |
| § Flatbed plotter          | § Peripheral device                         |
| § Flatbed Scanner          | § Phonemes                                  |
| § Graphical User Interface | § Plotter                                   |
| § Hand-held scanner        | § Point-and-draw device                     |
| § Hard-copy output         | § Printer                                   |
| § Image Scanner            | § QWERTY keyboard                           |
|                            | § Screen Image Projector                    |

(Continued on next slide)

# Key Words/Phrases

(Continued from previous slide..)

- § Soft-copy output
- § Speech synthesizer
- § Stylus
- § Touch Screen
- § Trackball
- § Universal Product Code (UPC)
- § Video Display Terminal (VDT)
- § Vision-input system
- § Voice recognition device
- § Voice reproduction system
- § Voice response system

# Chapter 10

# Computer Software

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

In this chapter you will learn about:

- § Term “Software” and its relationship with “Hardware”
- § Various types of software and their examples
- § Relationship among hardware, system software, application software, and users of a computer system
- § Different ways of acquiring software
- § Various steps involved in software development
- § Firmware
- § Middleware

# Software

- § **Hardware** refers to the physical devices of a computer system.
- § **Software** refers to a collection of programs
- § **Program** is a sequence of instructions written in a language that can be understood by a computer
- § **Software package** is a group of programs that solve a specific problem or perform a specific type of job

# Relationship Between Hardware and Software

- § Both hardware and software are necessary for a computer to do useful job. They are complementary to each other
- § Same hardware can be loaded with different software to make a computer system perform different types of jobs
- § Except for *upgrades*, hardware is normally a one-time expense, whereas software is a continuing expense
- § Upgrades refer to renewing or changing components like increasing the main memory, or hard disk capacities, or adding speakers, modems, etc.

# Types of Software

Most software can be divided into two major categories:

- § ***System software*** are designed to control the operation and extend the processing capability of a computer system
- § ***Application software*** are designed to solve a specific problem or to do a specific task

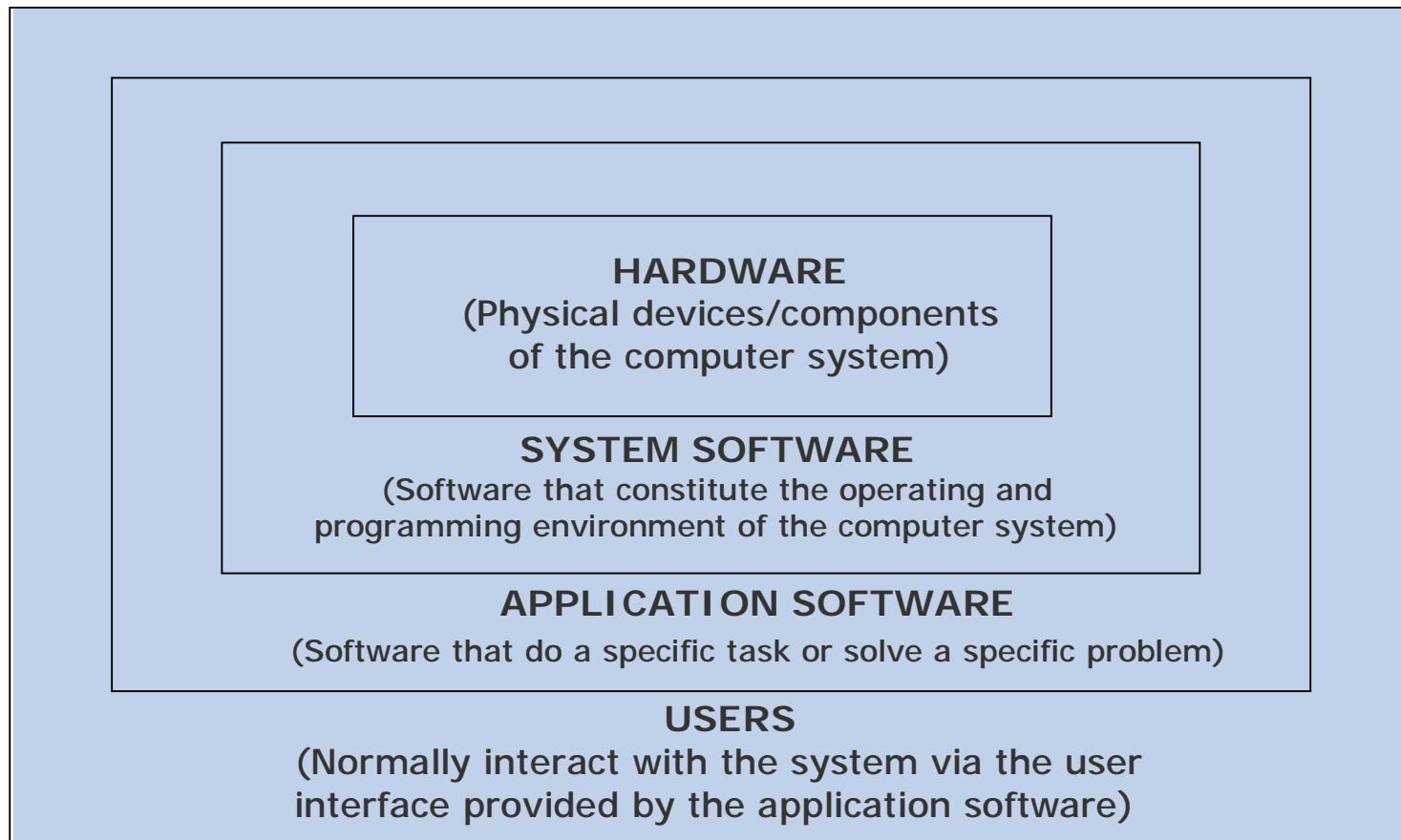
# System Software

- § Make the operation of a computer system more effective and efficient
- § Help hardware components work together and provide support for the development and execution of application software
- § Programs included in a system software package are called *system programs* and programmers who prepare them are called *system programmers*
- § Examples of system software are operating systems, programming language translators, utility programs, and communications software

# Application Software

- § Solve a specific problem or do a specific task
- § Programs included in an application software package are called *application programs* and the programmers who prepare them are called *application programmers*
- § Examples of application software are word processing, inventory management, preparation of tax returns, banking, etc.

# Logical System Architecture



Relationship among hardware, system software, application software, and users of a computer system.

# Ways of Acquiring Software

- § Buying pre-written software
- § Ordering customized software
- § Developing customized software
- § Downloading public-domain software

Each of these ways of acquiring software has its own advantages and limitations

# Advantages and Limitations of Buying Pre-written Software

- § Usually costs less
- § Planned activity can be started almost immediately
- § Often, operating efficiency and the capability to meet specific needs of user more effectively in not as good for pre-written software packages as for in-house developed software packages

# Advantages & Limitations of Ordering Customized Software

- § User need not maintain its own software development team, which is an expensive affair
- § User needs to always depend on the vendor for carrying out the changes and the vendor may separately charge for every request for change

# Advantages & Limitations of Developing Customized Software

- § Easier to carry out changes in the software, if it is developed in-house
- § Developing software in-house means a major commitment of time, money, and resources
- § In-house software development team needs to be maintained and managed

# Advantage & Limitations of Downloading Public-domain Software

- § Available for free or as shareware, and are usually accompanied with source code
- § Usually community-supported as author does not support users directly
- § Can be downloaded and used immediately
- § They may not be properly tested before release
- § Open Source Software (OSS) are becoming popular due to:
  - § Allows any user to download, view, modify, and redistribute
  - § User can fix bugs or change software to suit needs
  - § Copyright is protected for both original and subsequent authors
- § Not all open source software are free and vice-versa

# Software Development Steps

Developing a software and putting it to use is a complex process and involves following steps:

- § Analyzing the problem at hand and planning the program(s) to solve the problem
- § Coding the program(s)
- § Testing, debugging, and documenting the program(s)
- § Implementing the program(s)
- § Evaluating and maintaining the program(s)

# Firmware

- § Firmware is software substituted for hardware and stored in read-only memory
- § Firmware technology has enabled production of various types of smart machines having microprocessor chips with embedded software

# Middleware

- § Basic idea is to have a *separate software layer* to:
  - § Act as “glue” between client and server parts of application
  - § Provide programming abstraction
  - § Mask heterogeneity of underlying network, hardware, and OS
- § Encourages *three-tier* software architecture against two-tier popularized by Server-Client architecture

# Key Words/Phrases

- § Application programmers
- § Application programs
- § Application software
- § Computer program
- § Customized software
- § Database
- § Education software
- § End-to-end solution
- § Entertainment software
- § Firmware
- § Graphics software
- § Hardware
- § Middleware
- § Open Source Software
- § Personal assistance software
- § Pre-written software
- § Public-domain software
- § Shareware
- § Software
- § Software package
- § Spreadsheet
- § System programmers
- § System programs
- § System software
- § Turnkey solution
- § User-supported software
- § Utilities
- § Word-processing

# Chapter 11

# Planning the Computer Program

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

In this chapter you will learn about:

- § Programs must be planned before they are written
- § Algorithm
- § Flowchart
- § Pseudocode
- § Plan the logic of a computer program
- § Commonly used tools for program planning and their use

# Purpose of Program Planning

- § To write a correct program, a programmer must write each and every instruction in the correct sequence
- § Logic (instruction sequence) of a program can be very complex
- § Hence, programs must be planned before they are written to ensure program instructions are:
  - § Appropriate for the problem
  - § In the correct sequence

# Algorithm

- § Refers to the logic of a program and a step-by-step description of how to arrive at the solution of a given problem
- § In order to qualify as an algorithm, a sequence of instructions must have following characteristics:
  - § Each and every instruction should be precise and unambiguous
  - § Each instruction should be such that it can be performed in a finite time
  - § One or more instructions should not be repeated infinitely. This ensures that the algorithm will ultimately terminate
  - § After performing the instructions, that is after the algorithm terminates, the desired results must be obtained

# Sample Algorithm (Example 1)

There are 50 students in a class who appeared in their final examination. Their mark sheets have been given to you.

The division column of the mark sheet contains the division (FIRST, SECOND, THIRD or FAIL) obtained by the student.

Write an algorithm to calculate and print the total number of students who passed in FIRST division.

# Sample Algorithm (Example 1)

(contd...)

Step 1: Initialize Total\_First\_Division and  
Total\_Marksheets\_Checked to zero.

Step 2: Take the mark sheet of the next student.

Step 3: Check the division column of the mark sheet to see if it is  
FIRST, if no, go to Step 5.

Step 4: Add 1 to Total\_First\_Division.

Step 5: Add 1 to Total\_Marksheets\_Checked.

Step 6: Is Total\_Marksheets\_Checked = 50, if no, go to Step 2.

Step 7: Print Total\_First\_Division.

Step 8: Stop.

# Sample Algorithm (Example 2)

There are 100 employees in an organization. The organization wants to distribute annual bonus to the employees based on their performance. The performance of the employees is recorded in their annual appraisal forms.

Every employee's appraisal form contains his/her basic salary and the grade for his/her performance during the year. The grade is of three categories – 'A' for outstanding performance, 'B' for good performance, and 'C' for average performance.

It has been decided that the bonus of an employee will be 100% of the basic salary for outstanding performance, 70% of the basic salary for good performance, 40% of the basic salary for average performance, and zero for all other cases.

Write an algorithm to calculate and print the total bonus amount to be distributed by the organization.

# Sample Algorithm (Example 2)

Step 1: Initialize Total\_Bonus and Total\_Employees\_Checked to zero.

(contd...)

Step 2: Initialize Bonus and Basic\_Salary to zero.

Step 3: Take the appraisal form of the next employee.

Step 4: Read the employee's Basic\_Salary and Grade.

Step 5: If Grade = A, then Bonus = Basic\_Salary. Go to Step 8.

Step 6: If Grade = B, then Bonus = Basic\_Salary x 0.7. Go to Step 8.

Step 7: If Grade = C, then Bonus = Basic\_Salary x 0.4.

Step 8: Add Bonus to Total\_Bonus.

Step 9: Add 1 to Total\_Employees\_Checked.

Step 10: If Total\_Employees\_Checked < 100, then go to Step 2.

Step 11: Print Total\_Bonus.

Step 12: Stop.

# Representation of Algorithms

- § As programs
- § As flowcharts
- § As pseudocodes

When an algorithm is represented in the form of a programming language, it becomes a program

Thus, any program is an algorithm, although the reverse is not true

# Flowchart

- § *Flowchart* is a pictorial representation of an algorithm
- § Uses symbols (boxes of different shapes) that have standardized meanings to denote different types of instructions
- § Actual instructions are written within the boxes
- § Boxes are connected by solid lines having arrow marks to indicate the exact sequence in which the instructions are to be executed
- § Process of drawing a flowchart for an algorithm is called *flowcharting*

# Basic Flowchart Symbols



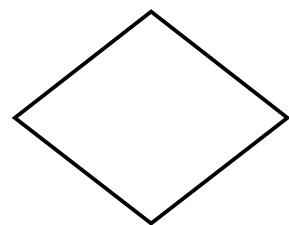
Terminal



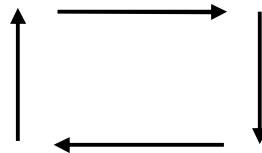
Input/Output



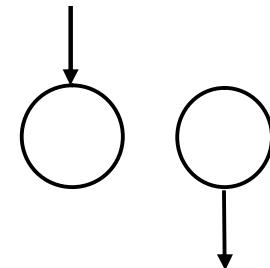
Processing



Decision

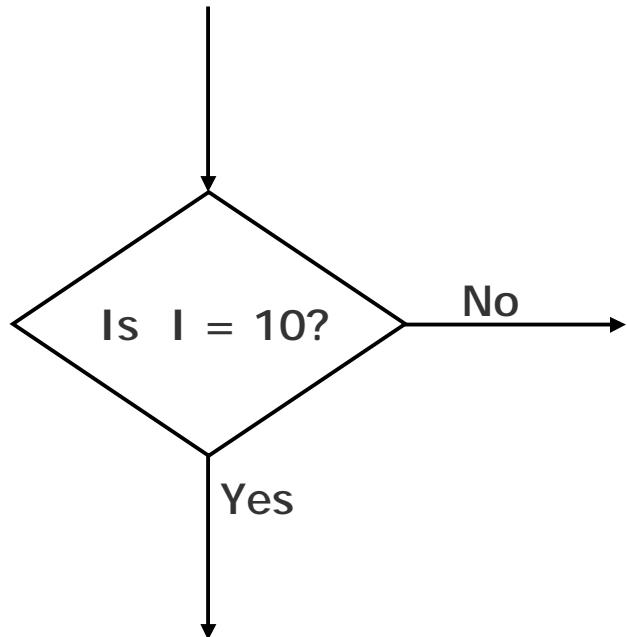


Flow lines

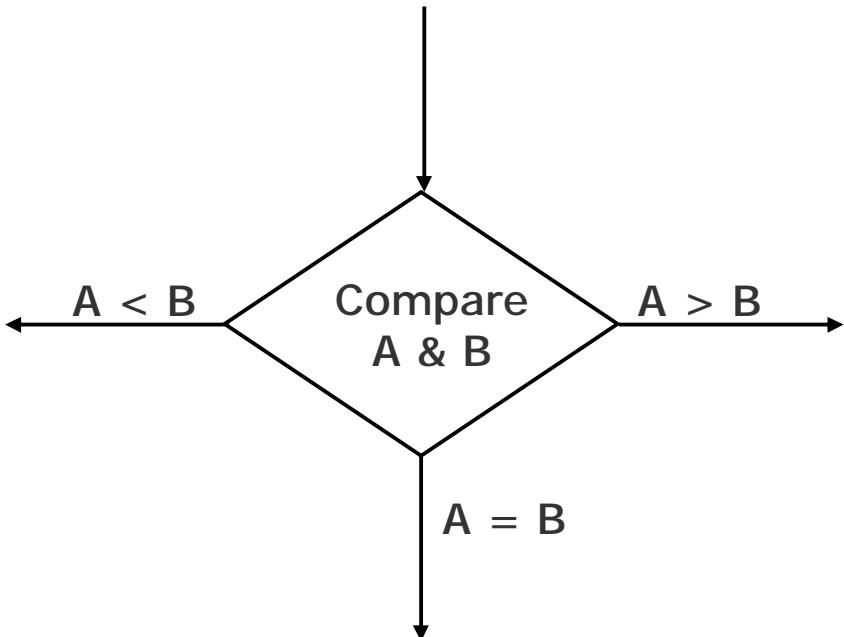


Connectors

# Examples of Decision Symbol



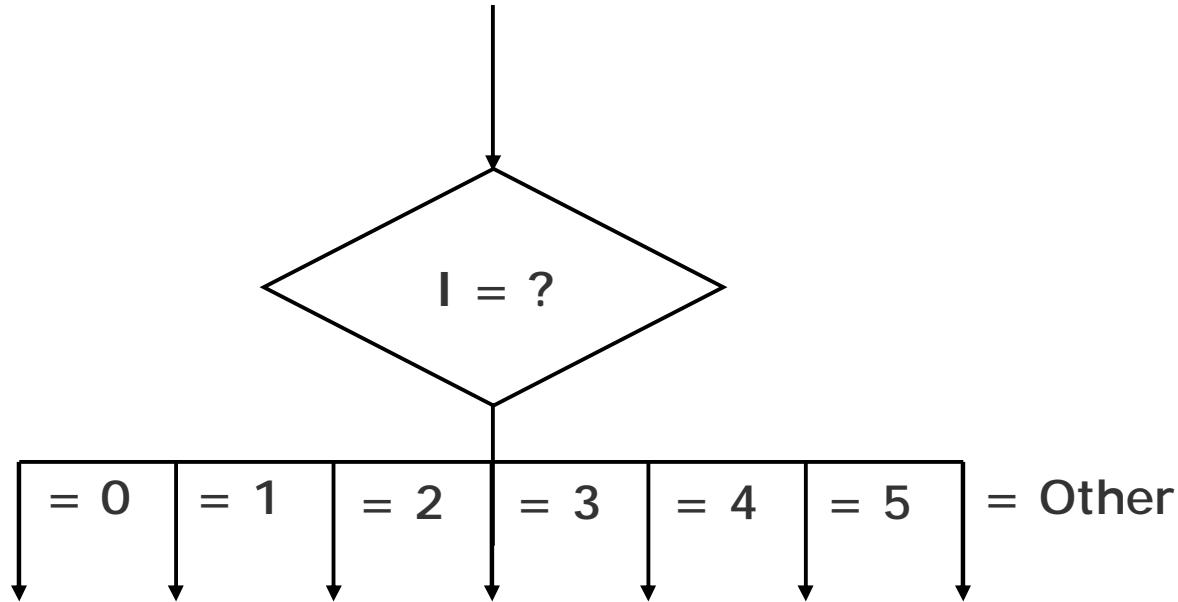
(a) A two-way branch decision.



(b) A three-way branch decision.

# Examples of Decision Symbol

(contd...)



(c) A multiple-way branch decision.

# Sample Flowchart (Example 3)

A student appears in an examination, which consists of total 10 subjects, each subject having maximum marks of 100.

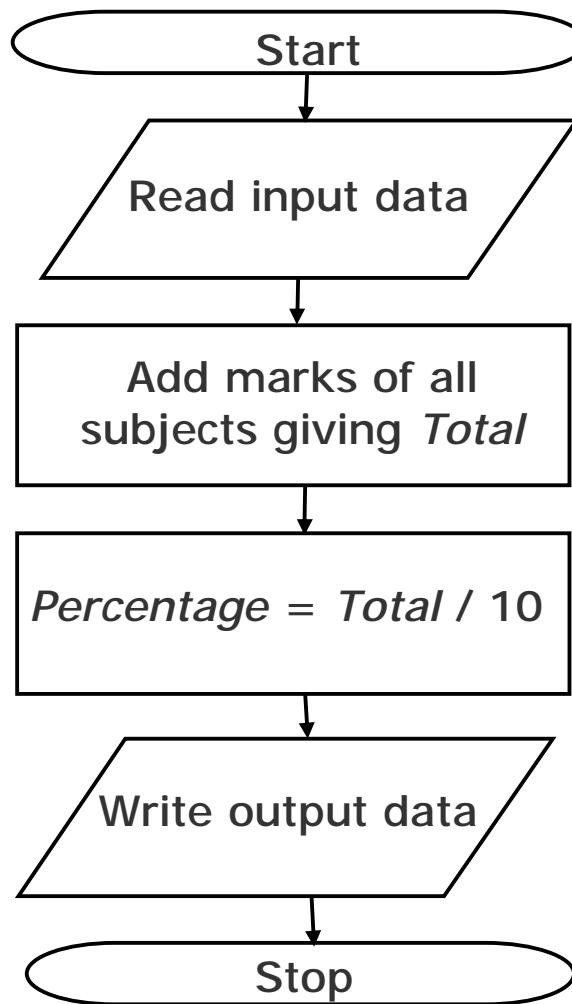
The roll number of the student, his/her name, and the marks obtained by him/her in various subjects are supplied as input data.

Such a collection of related data items, which is treated as a unit is known as a record.

Draw a flowchart for the algorithm to calculate the percentage marks obtained by the student in this examination and then to print it along with his/her roll number and name.

# Sample Flowchart (Example 3)

(contd...)



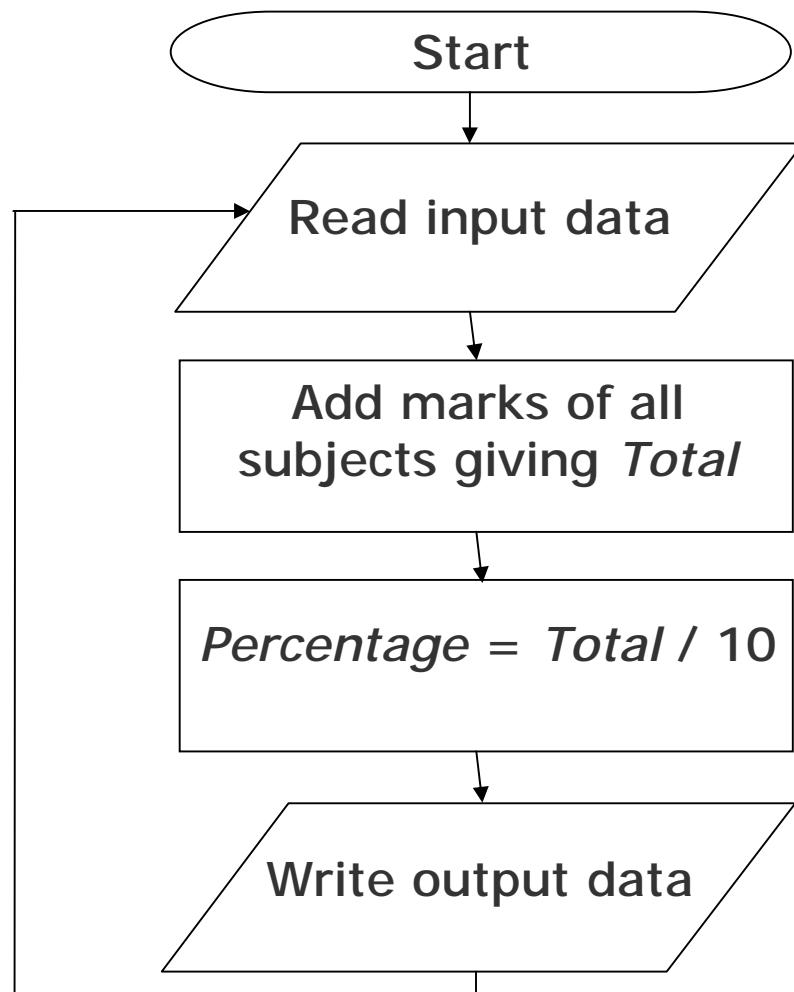
# Sample Flowchart (Example 4)

50 students of a class appear in the examination of Example 3.

Draw a flowchart for the algorithm to calculate and print the percentage marks obtained by each student along with his/her roll number and name.

# Sample Flowchart (Example 4)

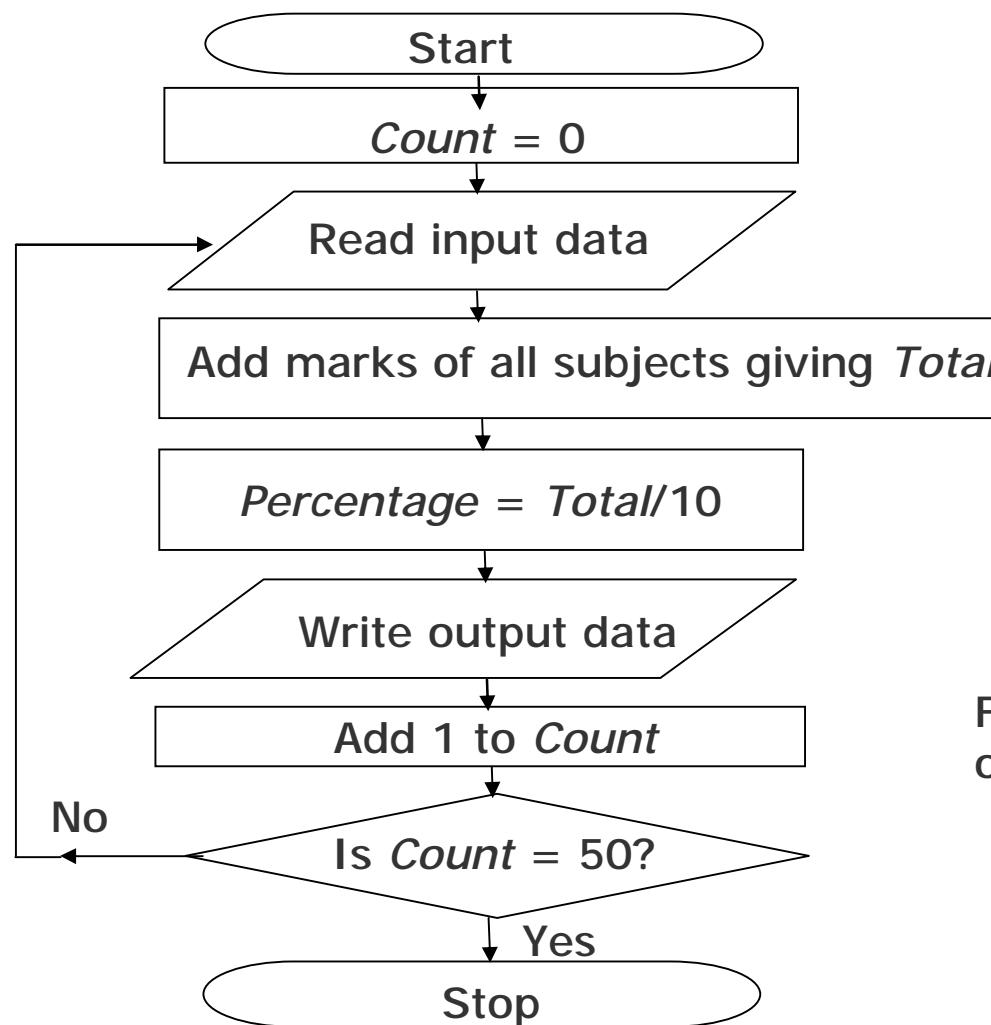
(contd...)



Flowchart for the solution of Example 4 with an infinite (endless) process loop.

# Sample Flowchart (Example 4)

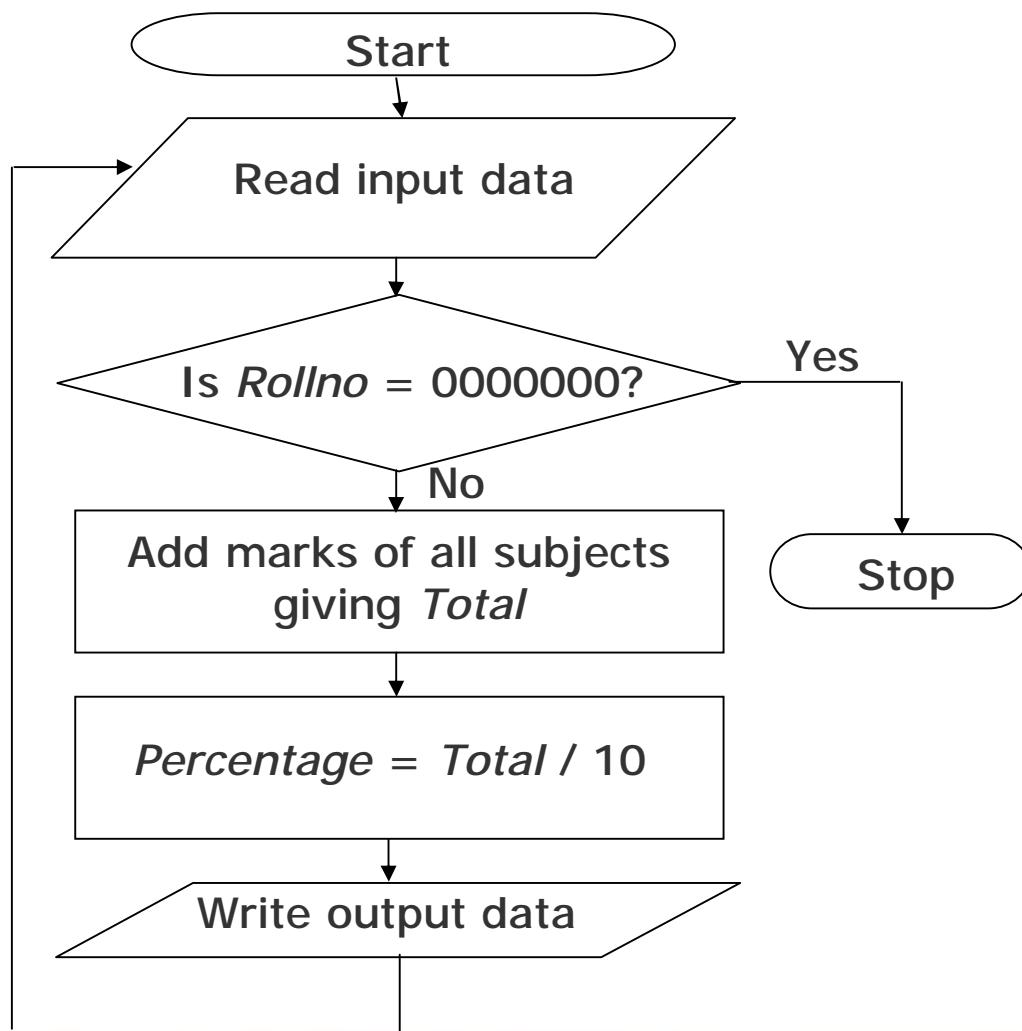
(contd...)



Flowchart for the solution  
of Example 4.

# Sample Flowchart (Example 4)

(contd...)



Generalized flowchart for the solution of Example 4 using the concept of trailer record. Here the process loop is terminated by detecting a special non-data record.

# Sample Flowchart (Example 5)

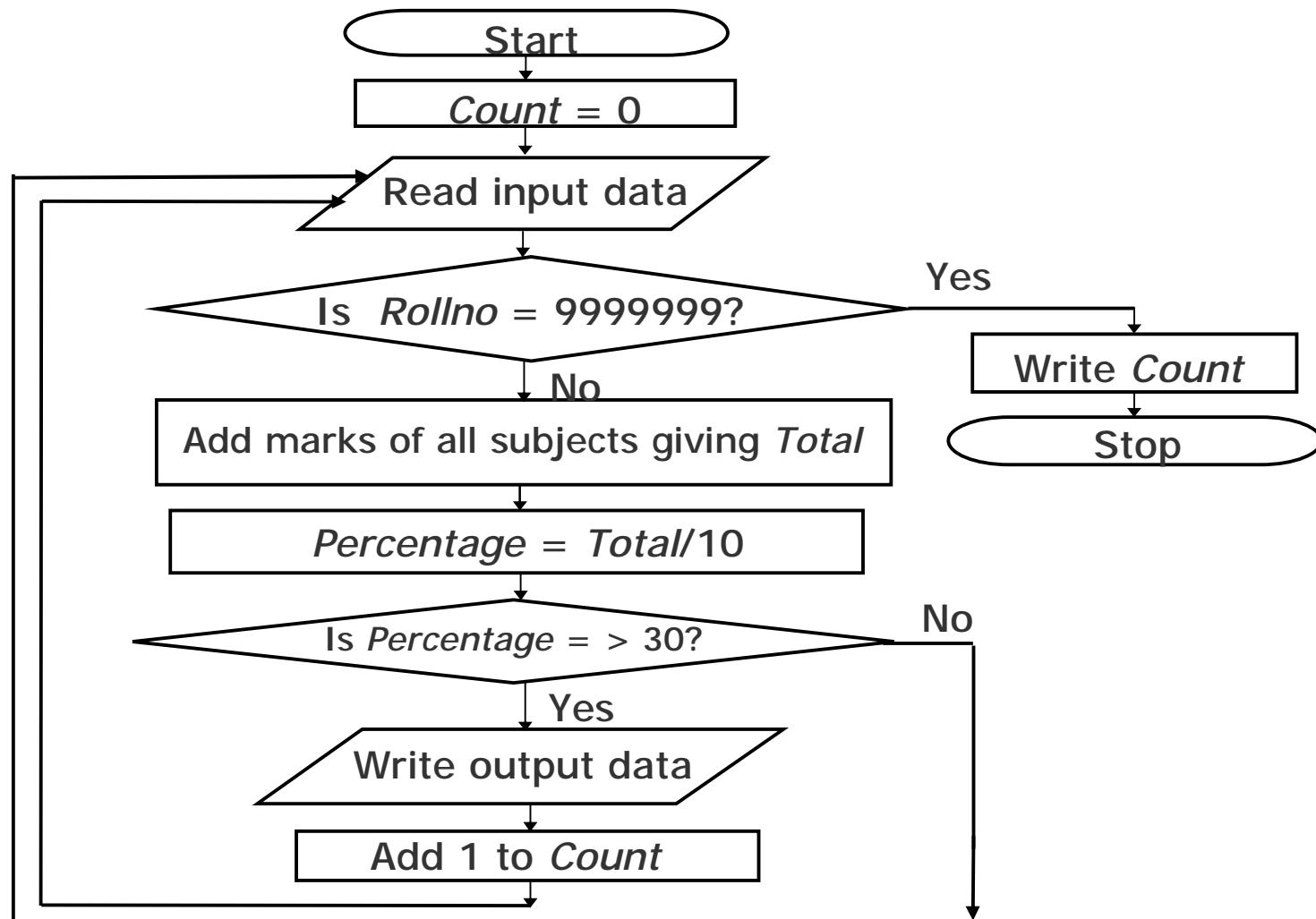
For the examination of Example 3, we want to make a list of only those students who have passed (obtained 30% or more marks) in the examination.

In the end, we also want to print out the total number of students who have passed.

Assuming that the input data of all the students is terminated by a trailer record, which has sentinel value of 9999999 for Rollno, draw a flowchart for the algorithm to do this.

# Sample Flowchart (Example 5)

(contd...)



# Sample Flowchart (Example 6)

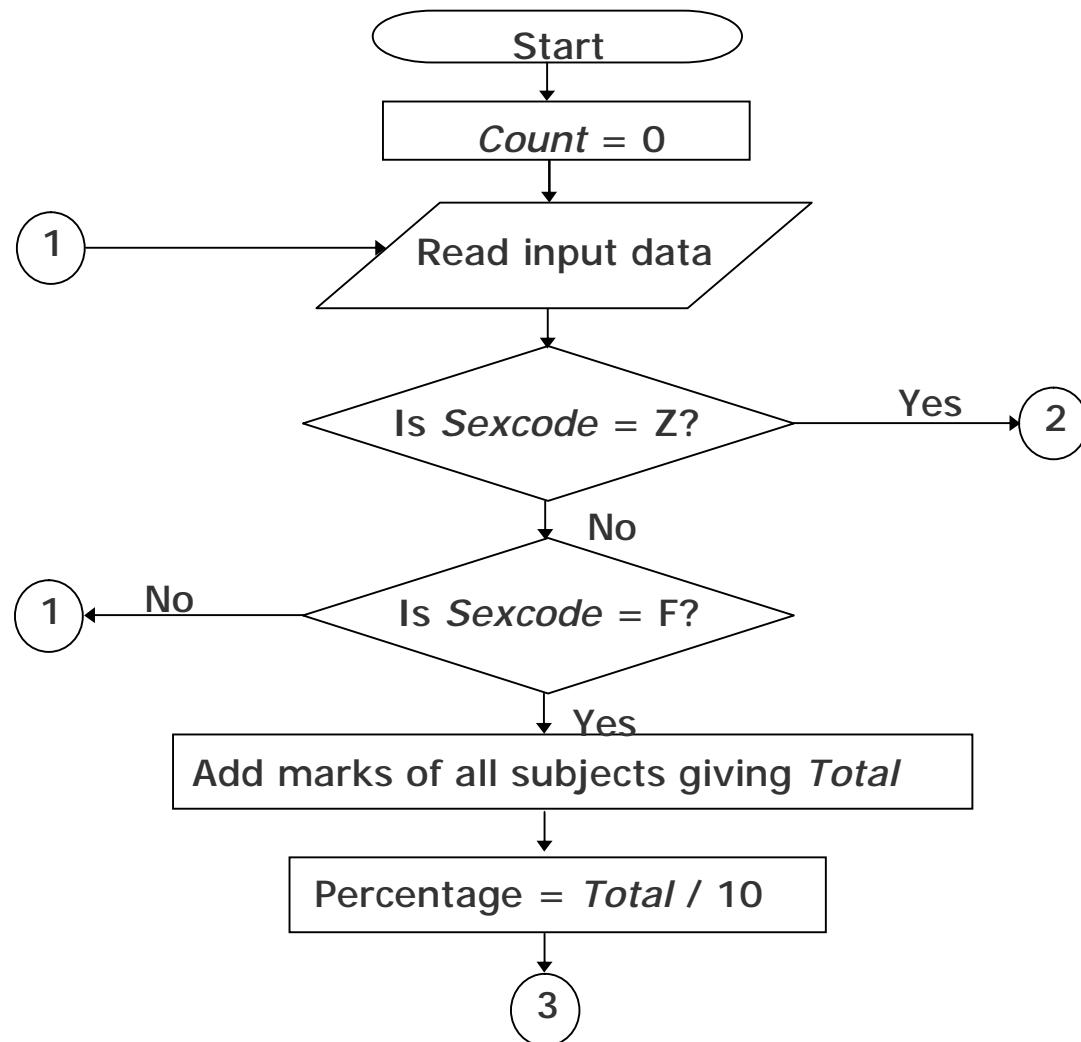
Suppose the input data of each student for the examination of Example 3 also contains information regarding the sex of the candidate in the field named *Sexcode* having values M (for male) or F (for female).

We want to make a list of only those female students who have passed in second division (obtained 45% or more but less than 60% marks).

In the end, we also want to print out the total number of such students.

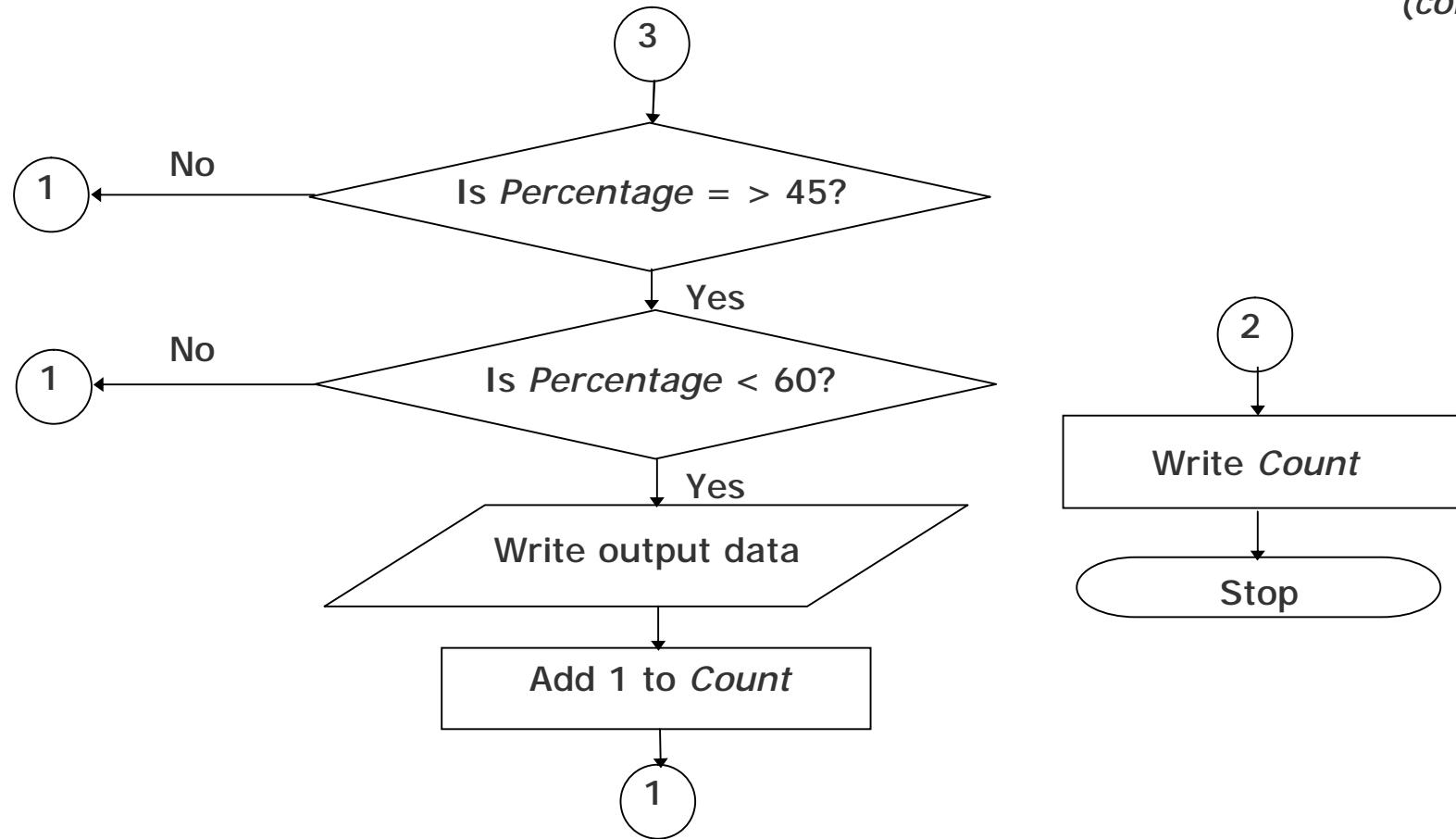
Assuming that the input data of all the students is terminated by a trailer record, which has a sentinel value of Z for *Sexcode*, draw a flowchart for the algorithm to do this.

# Sample Flowchart (Example 6)



# Sample Flowchart (Example 4)

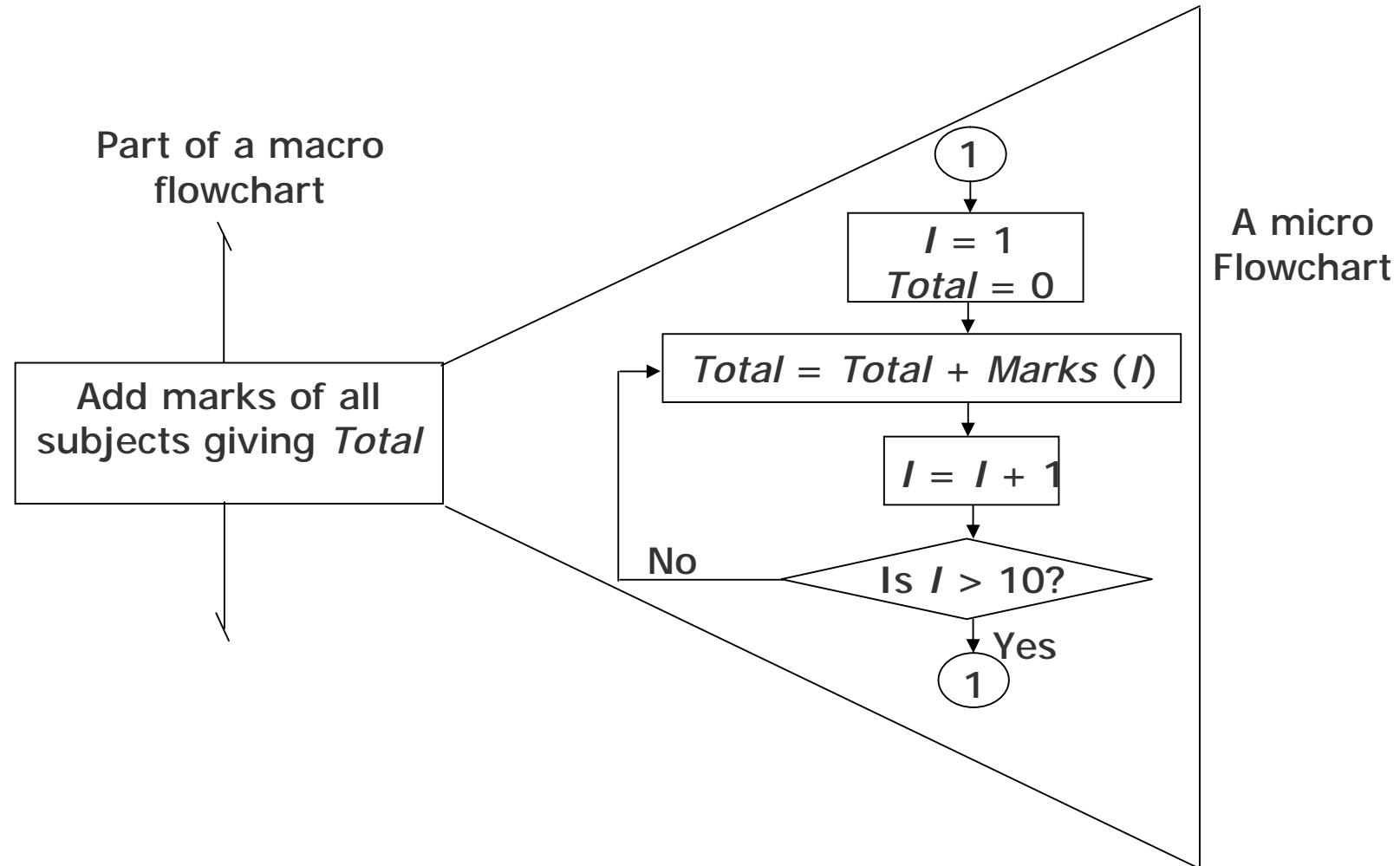
(contd...)



# Levels of Flowchart

- § Flowchart that outlines the main segments of a program or that shows less details is a *macro flowchart*
- § Flowchart with more details is a *micro flowchart*, or detailed flowchart
- § There are no set standards on the amount of details that should be provided in a flowchart

# Example of Micro Flowchart



# Flowcharting Rules

- § First chart the main line of logic, then incorporate detail
- § Maintain a consistent level of detail for a given flowchart
- § Do not chart every detail of the program. A reader who is interested in greater details can refer to the program itself
- § Words in the flowchart symbols should be common statements and easy to understand

# Flowcharting Rules

(contd...)

- § Be consistent in using names and variables in the flowchart
- § Go from left to right and top to bottom in constructing flowcharts
- § Keep the flowchart as simple as possible. Crossing of flow lines should be avoided as far as practicable
- § If a new flowcharting page is needed, it is recommended that the flowchart be broken at an input or output point.
- § Properly labeled connectors should be used to link the portions of the flowchart on different pages

# Advantages of Flowchart

- § Better Communication
- § Proper program documentation
- § Efficient coding
- § Systematic debugging
- § Systematic testing

# Limitations of Flowchart

- § Flowcharts are very time consuming and laborious to draw (especially for large complex programs)
- § Redrawing a flowchart for incorporating changes/modifications is a tedious task
- § There are no standards determining the amount of detail that should be included in a flowchart

# Pseudocode

- § A program planning tool where program logic is written in an ordinary natural language using a structure that resembles computer instructions
- § “Pseudo” means imitation or false and “Code” refers to the instructions written in a programming language. Hence, pseudocode is an imitation of actual computer instructions
- § Because it emphasizes the design of the program, pseudocode is also called *Program Design Language (PDL)*

# Basic Logic (Control) Structures

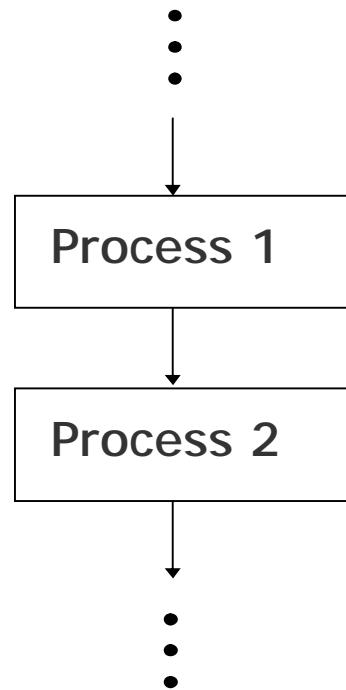
Any program logic can be expressed by using only following three simple logic structures:

1. Sequence logic,
2. Selection logic, and
3. Iteration (or looping) logic

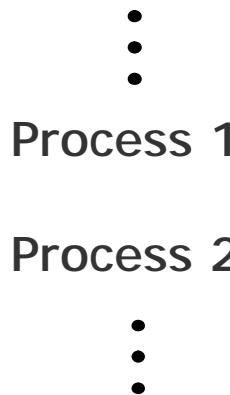
Programs structured by using only these three logic structures are called ***structured programs***, and the technique of writing such programs is known as ***structured programming***

# Sequence Logic

**It is used for performing instructions one after another in sequence.**



### (a) Flowchart

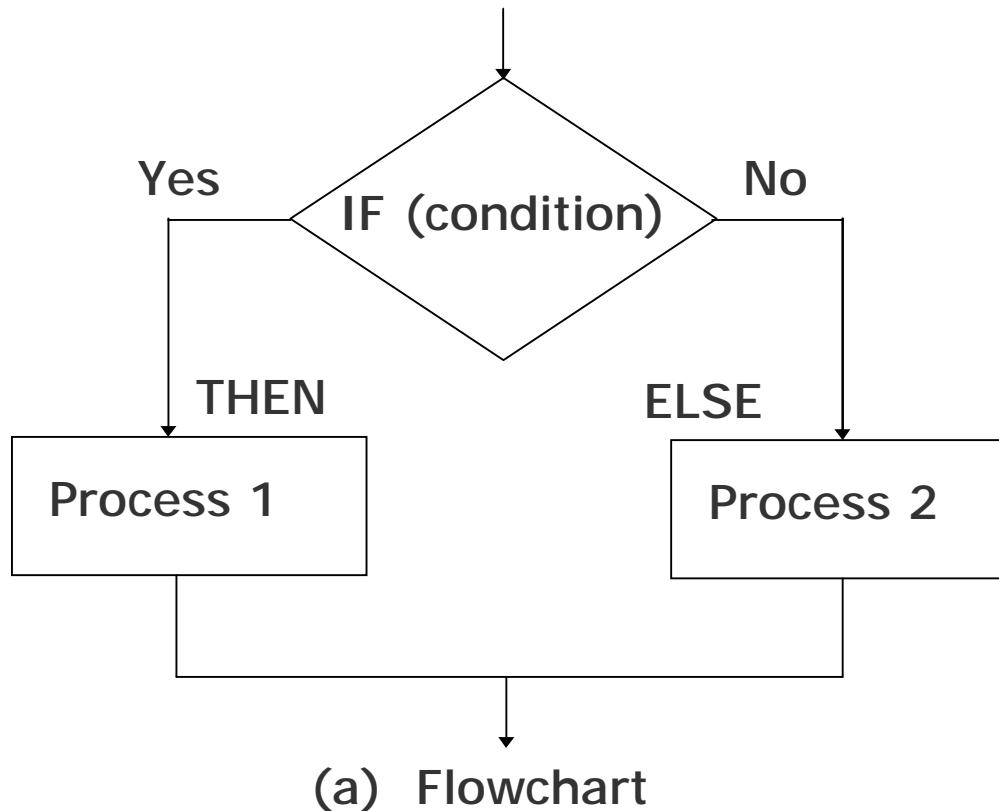


## (b) Pseudocode

# Selection Logic

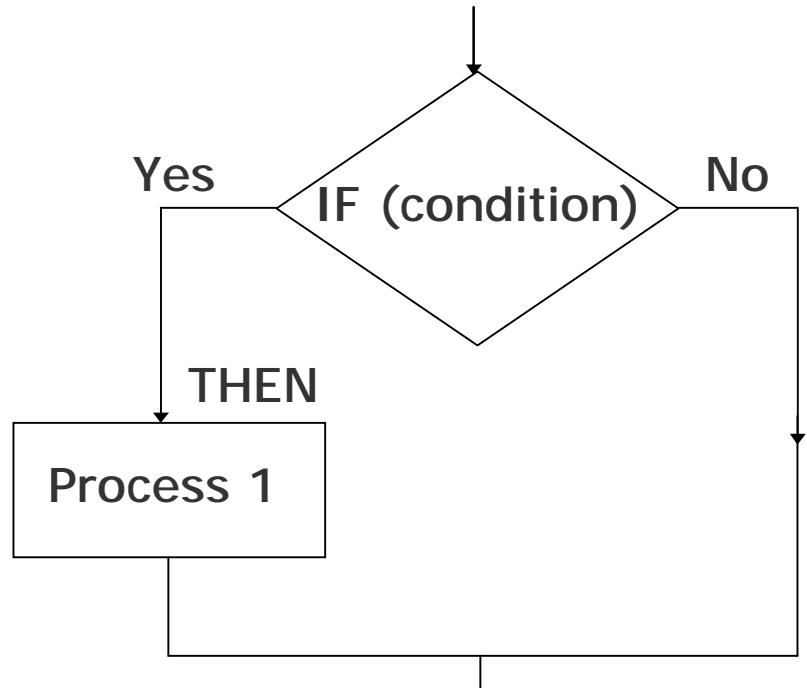
- Also known as decision logic, it is used for making decisions
- Three popularly used selection logic structures are
  1. IF...THEN...ELSE
  2. IF...THEN
  3. CASE

# Selection Logic (IF...THEN...ELSE Structure)



IF Condition  
THEN      Process 1  
ELSE      Process 2  
ENDIF  
⋮  
(b) Pseudocode

# Selection Logic (IF...THEN Structure)



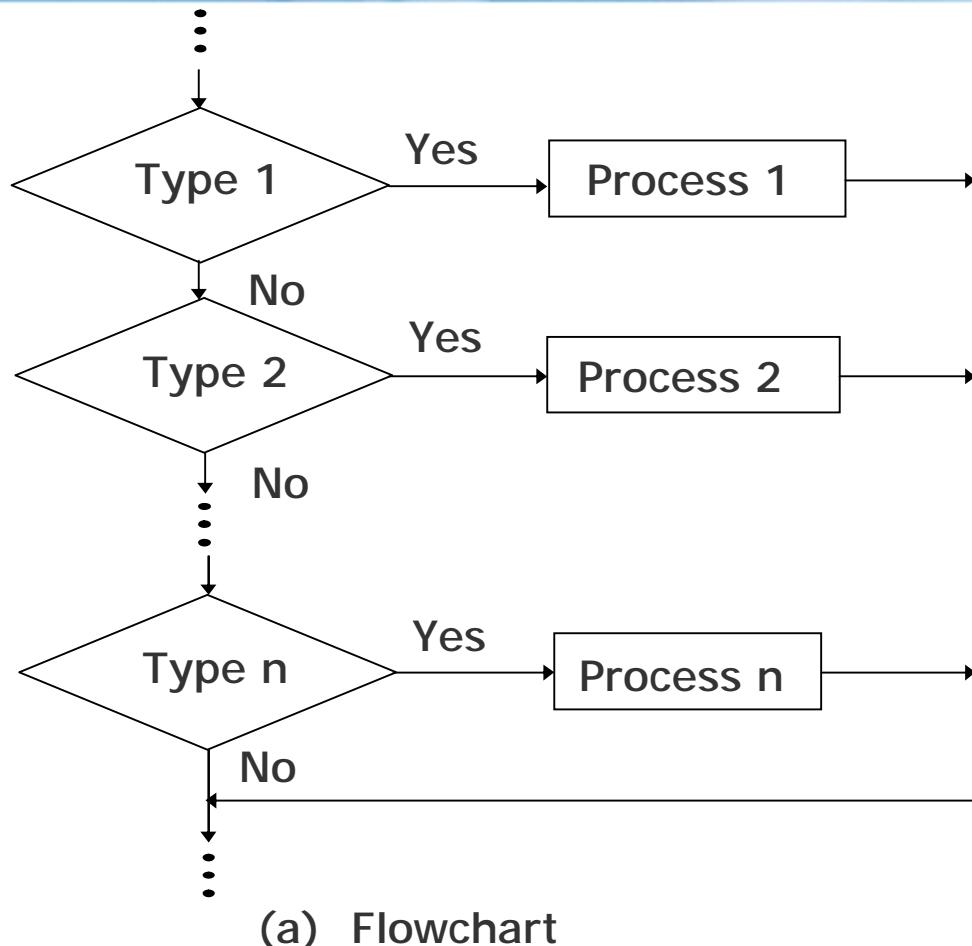
(a) Flowchart

IF Condition  
THEN      Process 1  
ENDIF

•  
•

(b) Pseudocode

# Selection Logic (CASE Structure)



•  
•  
•  
CASE Type  
Case Type 1: Process 1  
Case Type 2: Process 2  
•  
Case Type n: Process n

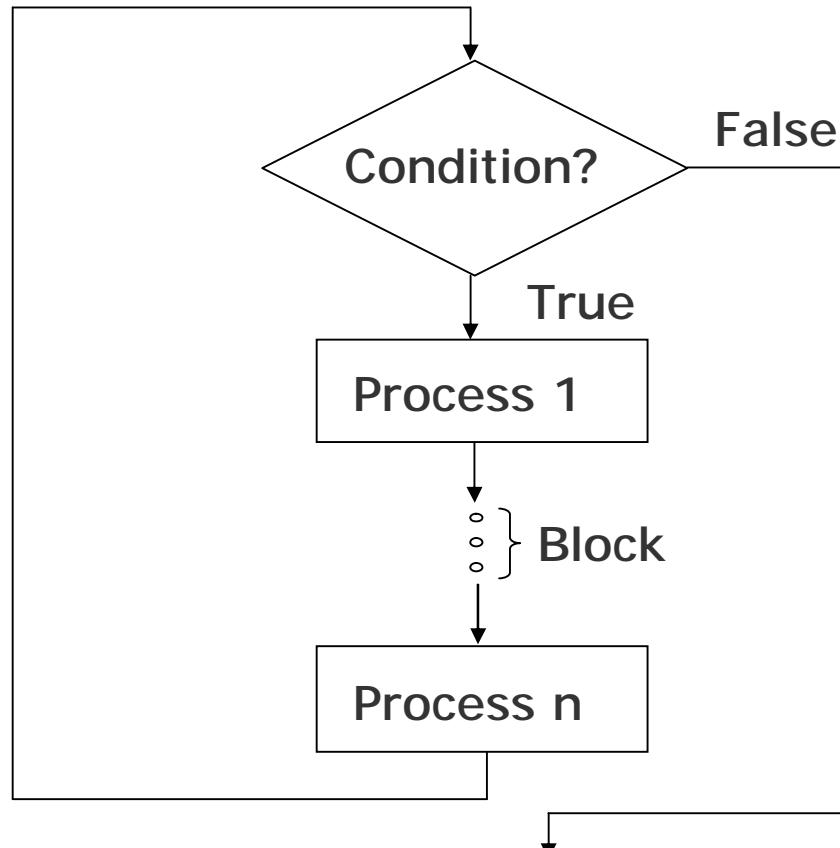
ENDCASE

•  
•  
(b) Pseudocode

# Iteration (or Looping) Logic

- § Used to produce loops in program logic when one or more instructions may be executed several times depending on some conditions
- § Two popularly used iteration logic structures are
  1. DO...WHILE
  2. REPEAT...UNTIL

# Iteration (or Looping) Logic (DO...WHILE Structure)

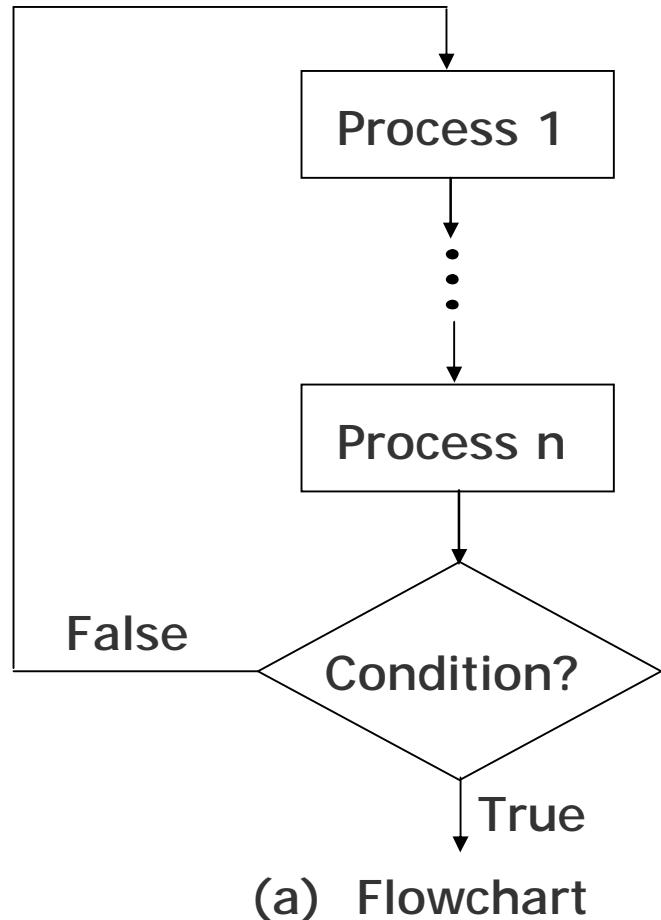


(a) Flowchart

DO WHILE Condition  
Process 1  
⋮  
Process n  
ENDDO  
⋮

(b) Pseudocode

# Iteration (or Looping) Logic (REPEAT...UNTIL Structure)



REPEAT  
Process 1  
•  
Process n  
UNTIL Condition  
•  
(b) Pseudocode

## Sample Pseudocode (for Example 6)

```
Set Count to zero
Read first student record
DO WHILE Sexcode is not equal to Z
    IF Sexcode = F THEN
        Calculate Percentage
        IF Percentage = > 45 THEN
            IF Percentage < 60 THEN
                Write output data
                Add 1 to Count
            ENDIF
        ENDIF
    ENDIF
    Read next student record
ENDDO
Write Count
Stop
```

# Advantages of Pseudocode

- § Converting a pseudocode to a programming language is much more easier than converting a flowchart to a programming language
- § As compared to a flowchart, it is easier to modify the pseudocode of a program logic when program modifications are necessary
- § Writing of pseudocode involves much less time and effort than drawing an equivalent flowchart as it has only a few rules to follow

# Limitations of Pseudocode

- § In case of pseudocode, a graphic representation of program logic is not available
- § There are no standard rules to follow in using pseudocode
- § Different programmers use their own style of writing pseudocode and hence communication problem occurs due to lack of standardization
- § For a beginner, it is more difficult to follow the logic of or write pseudocode, as compared to flowcharting

# Key Words/Phrases

- § Algorithm
- § Basic logic structures
- § Control structures
- § Flowchart
- § Iteration logic
- § Looping logic
- § Micro flowchart
- § Macro flowchart
- § Pseudocode
- § Program Design Language (PDL)
- § Sequence logic
- § Selection logic
- § Sentinel value
- § Structured programming
- § Trailer record

# Chapter 12

# Computer Languages

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

In this chapter you will learn about:

- § Computer languages or programming languages
- § Three broad categories of programming languages – machine, assembly, and high-level languages
- § Commonly used programming language tools such as assembler, compiler, linker, and interpreter
- § Concepts of object-oriented programming languages
- § Some popular programming languages such as FORTRAN, COBOL, BASIC, Pascal, C, C++, C#, Java, RPG, LISP and SNOBOL
- § Related concepts such as Subprogram, Characteristics of a good programming language, and factors to consider while selecting a language for coding an application

# Broad Classification of Computer Languages

- § Machine language
- § Assembly language
- § High-level language

# Machine Language

- § Only language of a computer understood by it without using a translation program
- § Normally written as strings of binary 1s and 0s
- § Written using decimal digits if the circuitry of the computer being used permits this

# A Typical Machine Language Instruction Format

<b>OPCODE</b> (operation code)	<b>OPERAND</b> (Address/Location)
-----------------------------------	--------------------------------------

- § OPCODE tells the computer which operation to perform from the instruction set of the computer
- § OPERAND tells the address of the data on which the operation is to be performed

# A Sample Machine Language Program

00100000000001100111001	10001471
00110000000001000100001	14002041
011000000000011100101110	30003456
10100011111011100101110	50773456
00000000000000000000000000	00000000

**In Binary**

(Difficult to read and understand)

**In Decimal**

(Easier to read and understand)

# Advantages & Limitations of Machine Language

## Advantage

- § Can be executed very fast

## Limitations

- § Machine Dependent
- § Difficult to program
- § Error prone
- § Difficult to modify

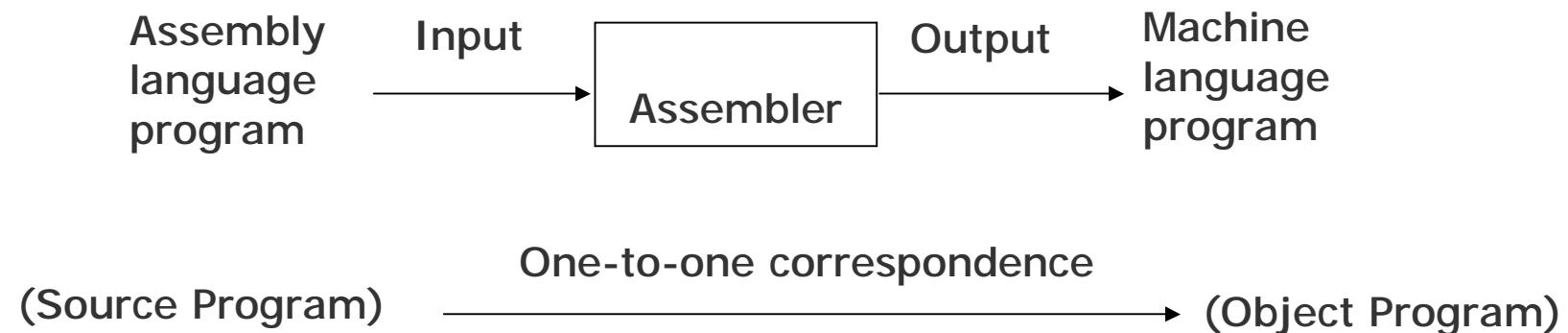
# Assembly/Symbolic Language

Programming language that overcomes the limitations of machine language programming by:

- § Using alphanumeric mnemonic codes instead of numeric codes for the instructions in the instruction set  
e.g. using ADD instead of 1110 (binary) or 14 (decimal) for instruction to add
- § Allowing storage locations to be represented in form of alphanumeric addresses instead of numeric addresses  
e.g. representing memory locations 1000, 1001, and 1002 as FRST, SCND, and ANSR respectively
- § Providing pseudo-instructions that are used for instructing the system how we want the program to be assembled inside the computer's memory  
e.g. START PROGRAM AT 0000; SET ASIDE AN ADDRESS FOR FRST

# Assembler

- § Software that translates an assembly language program into an equivalent machine language program of a computer



# An Example of Assembly Language Program

Mnemonic	Opcode	Meaning
HLT	00	Halt, used at the end of program to stop
CLA	10	Clear and add into A register
ADD	14	Add to the contents of A register
SUB	15	Subtract from the contents of A register
STA	30	Store A register

A subset of the set of instructions supported by a computer

# An Example of Assembly Language Program

```
START PROGRAM AT 0000
START DATA AT 1000
SET ASIDE AN ADDRESS FOR FRST
SET ASIDE AN ADDRESS FOR SCND
SET ASIDE AN ADDRESS FOR ANSR
CLA FRST
ADD SCND
STA ANSR
HLT
```

Sample assembly language program for adding two numbers and storing the result

# An Example of Assembly Language Program

Symbolic name	Memory location
FRST	1000
SCND	1001
ANSR	1002

Mapping table set up by the assembler for the data items of the assembly language program

# An Example of Assembly Language Program

Memory location	Contents		Comments
	Opcode	Address	
0000	10	1000	Clear and add the number stored at FRST to A register
0001	14	1001	Add the number stored at SCND to the contents of A register
0002	30	1002	Store the contents of A register into ANSR
0003	00		Halt
-			
-			
-			
1000			Reserved for FRST
1001			Reserved for SCND
1002			Reserved for ANSR

Equivalent machine language program for the assembly language program

# Advantages of Assembly Language Over Machine Language

- § Easier to understand and use
- § Easier to locate and correct errors
- § Easier to modify
- § No worry about addresses
- § Easily relocatable
- § Efficiency of machine language

# Limitations of Assembly Language

- § Machine dependent
- § Knowledge of hardware required
- § Machine level coding

## Typical Uses of Assembly Language

- § Mainly used today to fine-tune important parts of programs written in a high-level language to improve the program's execution efficiency

# Assembly Languages with Macro Instructions

- § Any assembly language instruction that gets translated into several machine language instructions is called a **macro instruction**
- § Several assembly languages support such macro instructions to speed up the coding process
- § Assemblers of such assembly languages are designed to produce multiple machine language instructions for each macro instruction of the assembly language

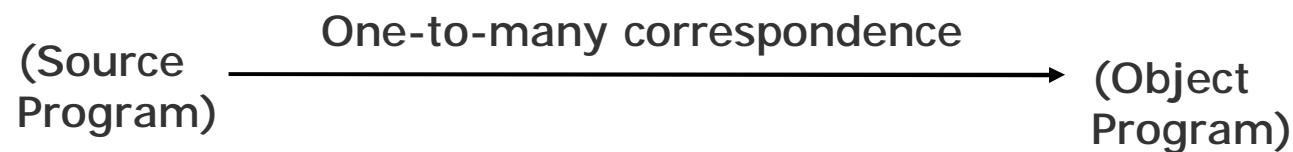
# High-Level Languages

- § Machine independent
- § Do not require programmers to know anything about the internal structure of computer on which high-level language programs will be executed
- § Deal with high-level coding, enabling the programmers to write instructions using English words and familiar mathematical symbols and expressions

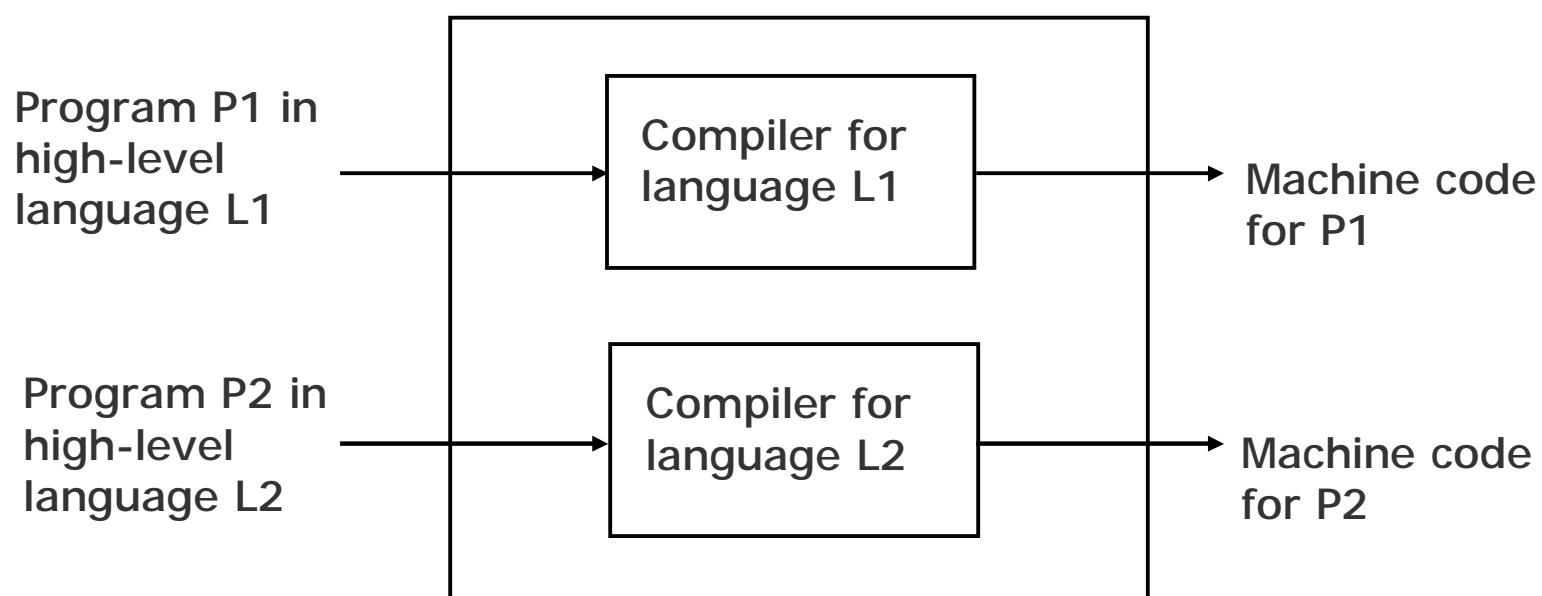
# Compiler

- § Translator program (software) that translates a high-level language program into its equivalent machine language program
- § Compiles a set of machine language instructions for every program instruction in a high-level language

# Compiler



# Compiler



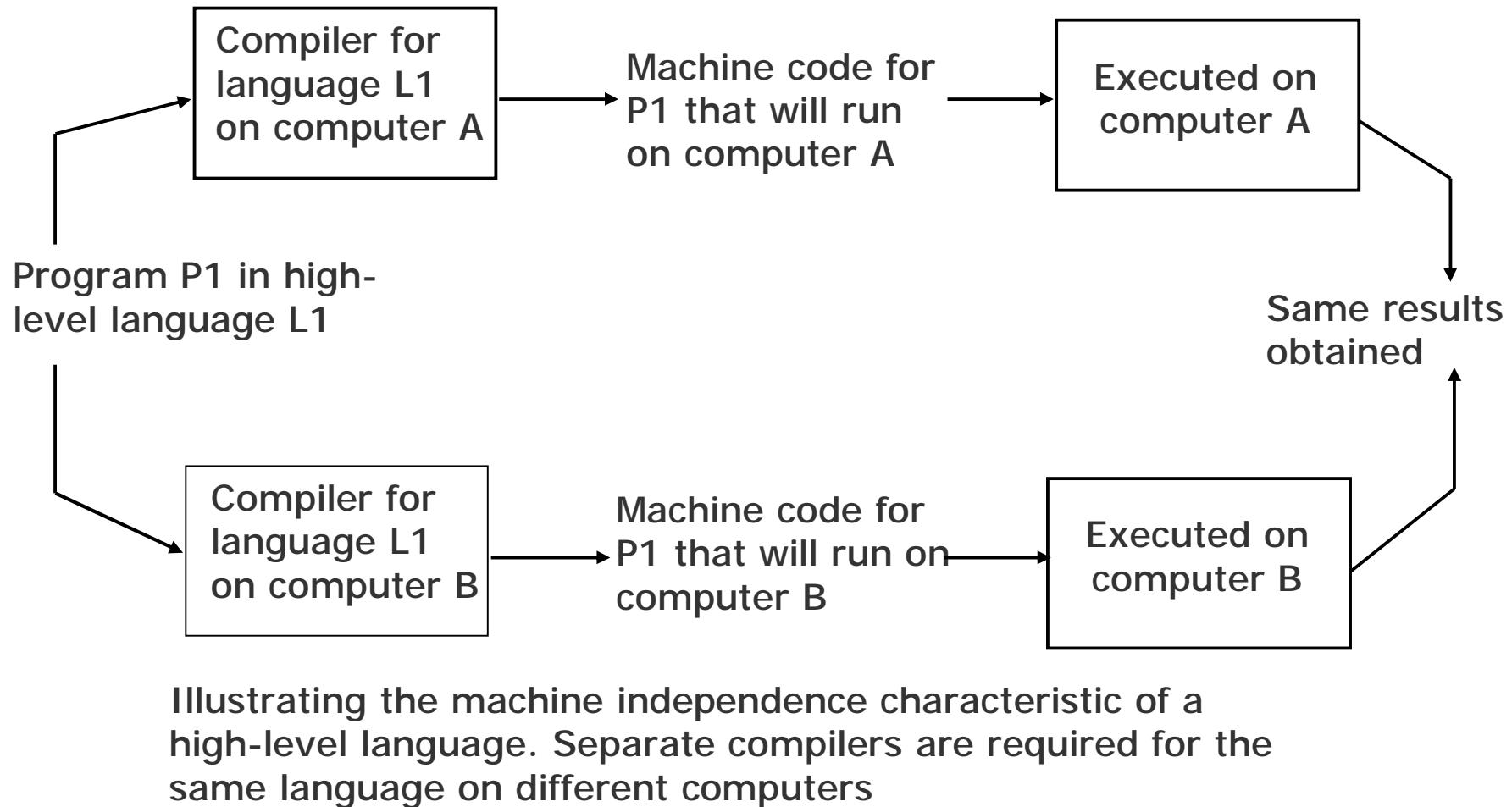
A computer supporting languages L1 and L2

Illustrating the requirement of a separate compiler for each high-level language supported by a computer

(Continued on next slide)

# Compiler

(Continued from previous slide..)



# Syntax Errors

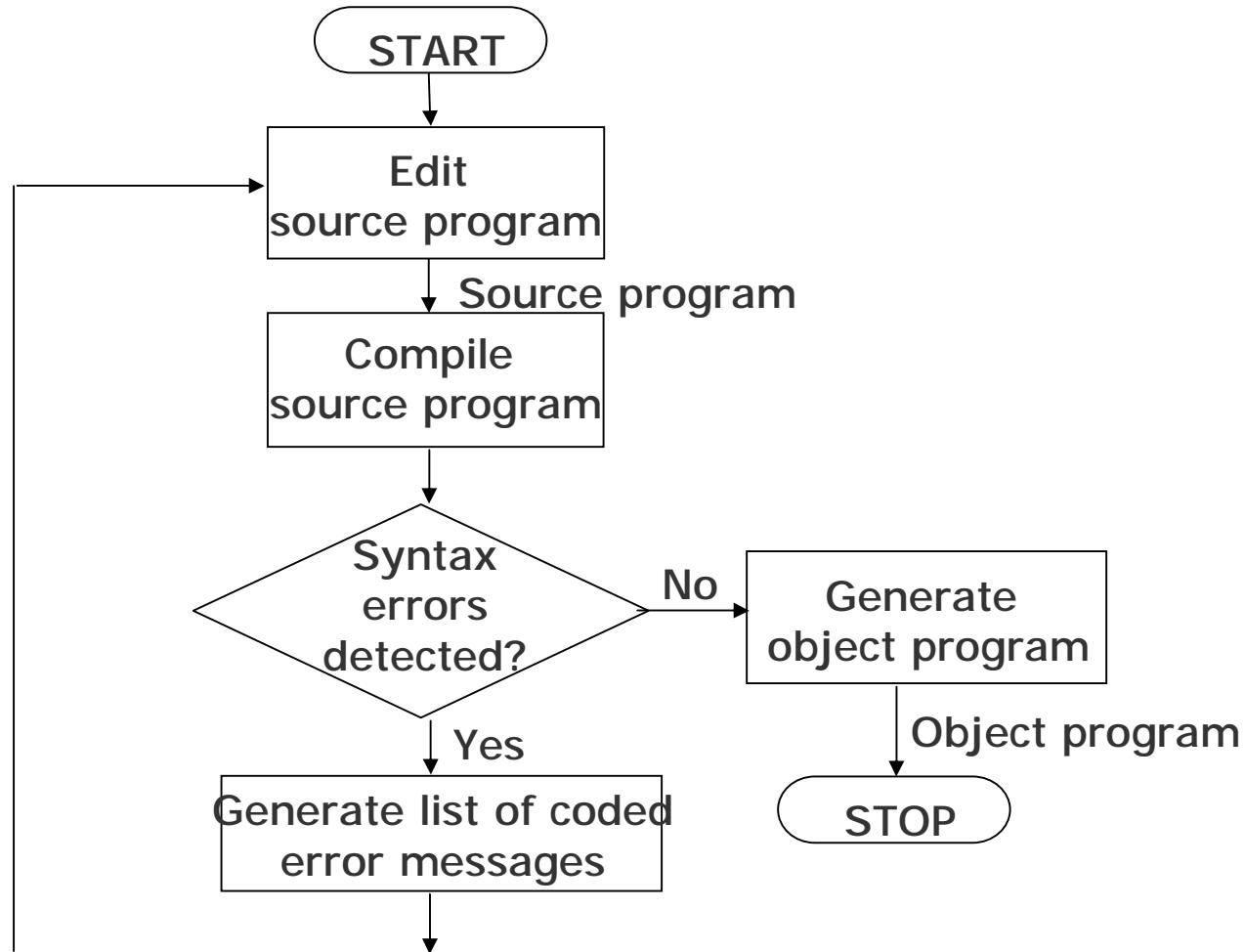
In addition to doing translation job, compilers also automatically detect and indicate syntax errors

Syntax errors are typically of following types:

- § Illegal characters
- § Illegal combination of characters
- § Improper sequencing of instructions in a program
- § Use of undefined variable names

*Note : A compiler cannot detect logic errors in a program*

# The Process of Removing Syntax Errors From A Source Program



# Linker

- § For a large software, storing all the lines of program code in a single source file will be:
  - Difficult to work with
  - Difficult to deploy multiple programmers to concurrently work towards its development
  - Any change in the source program would require the entire source program to be recompiled
- § Hence, a modular approach is generally adapted to develop large software where the software consists of multiple source program files
- § No need to write programs for some modules as it might be available in library offering the same functionality

(Continued on next slide)

# Linker

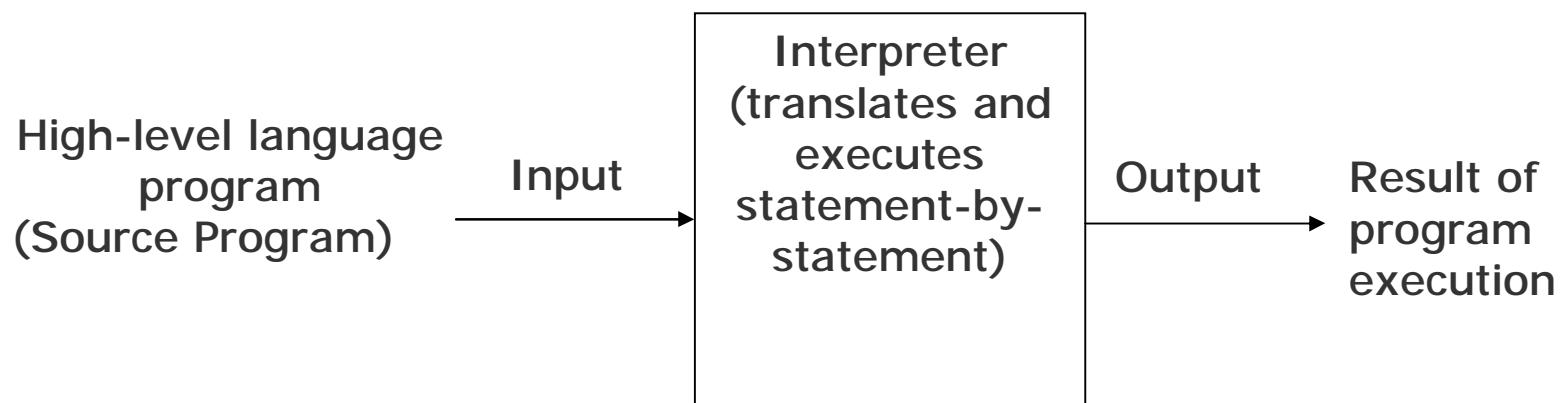
(Continued from previous slide..)

- § Each source program file can be independently modified and compiled to create a corresponding object program file
- § Linker program (software) is used to properly combine all the object program files (modules)
- § Creates the final executable program (load module)

# Interpreter

- § Interpreter is a high-level language translator
- § Takes one statement of a high-level language program, translates it into machine language instructions
- § Immediately executes the resulting machine language instructions
- § Compiler simply translates the entire source program into an object program and is not involved in its execution

# Role of an Interpreter



# Intermediate Language Compiler & Interpreter

- § New type of compiler and interpreter combines the speed, ease, and control of both compiler and interpreter
- § Compiler first compiles the source program to an *intermediate* object program
- § Intermediate object program is not a machine language code but written in an intermediate language that is virtually machine independent
- § Interpreter takes intermediate object program, converts it into machine language program and executes it

# Benefits of Intermediate Language Compiler & Interpreter

- § Intermediate object program is in compiled form and thus is not original source code, so safer and easier to share
- § Intermediate object program is based on a standard Intermediate Definition Language (IDL)
- § Interpreter can be written for any computer architecture and operating system providing virtual machine environment to the executing program
- § Newer Interpreter compiles intermediate program, in memory, into final host machine language program and executes it
- § This technique is called *Just-In-Time (JIT) Compilation*

# Advantages of High-Level Languages

- § Machine independent
- § Easier to learn and use
- § Fewer errors during program development
- § Lower program preparation cost
- § Better documentation
- § Easier to maintain

# Limitations of High-Level Languages

- § Lower execution efficiency
- § Less flexibility to control the computer's CPU, memory and registers

# Object-Oriented Programming Languages

- § Programming languages are used for simulating real-world problems on computers
- § Much of the real world is made up of objects
- § Essence of OOP is to solve a problem by:
  - § Identifying the real-world objects of the problem
  - § Identifying processing required of them
  - § Creating simulations of objects, processes, and their communications

# FORTRAN

- § Stands for FORmula TRANslation
- § Originally developed by John Backus and his team at IBM followed by several revisions
- § Standardized by ANSI as FORTRAN-77 and FORTRAN-90
- § Designed for solving scientific & engineering problems
- § Oriented towards solving problems of a mathematical nature
- § Popular language amongst scientists and engineers

# A Sample FORTRAN Program

```
C FORTRAN PROGRAM TO COMPUTE
C THE SUM OF 10 NUMBERS
SUM = 0
DO 50 I = 1, 10
READ (5, 10) N
SUM = SUM + N
50 CONTINUE
WRITE (6, 20) SUM
10 FORMAT (F6.2)
20 FORMAT (1X, 'THE SUM OF GIVEN NUMBERS = ',
F10.2)
STOP
END
```

# COBOL

- § Stands for COmmon Business Oriented Language
- § Originally developed started under Grace Hopper followed by COnference on DAta SYstems Languages (CODASYL)
- § Standardized by ANSI as COBOL-74, COBOL-85, and COBOL-2002
- § Designed for programming business data processing applications
- § Designed to have the appearance and structure of a business report written in English, hence often referred to as a self-documenting language

# A Sample COBOL Program

```
IDENTIFICATION DIVISION.  
PROGRAM_ID. SUMUP.  
AUTHOR. P K SINHA.  
* THIS PROGRAM COMPUTES AND PRINTS  
* THE SUM OF GIVEN NUMBERS.
```

```
ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.  
SOURCE_COMPUTER. BURROUGHS_6700.  
OBJECT_COMPUTER. BURROUGHS_6700.  
INPUT_OUTPUT SECTION.  
FILE_CONTROL.  
    SELECT DATA_FILE ASSIGN TO DISK.  
    SELECT OUTPUT_FILE ASSIGN TO PRINTER.
```

```
DATA DIVISION.  
FILE SECTION.
```

(Continued on next slide)

# A Sample COBOL Program

(Continued from previous slide..)

**FD DATA\_FILE**

RECORD CONTAINS 80 CHARACTERS

LABEL RECORD IS OMITTED

DATA RECORD IS INPUT\_DATA\_RECORD.

01 INPUT\_DATA\_RECORD.

  05 NPICTURE 9(6)V99.

  05 FILLER PICTURE X(72).

**FD OUTPUT\_FILE**

RECORD CONTAINS 132 CHARACTERS

LABEL RECORD IS OMITTED

DATA RECORD IS OUTPUT\_RECORD.

(Continued on next slide)

# A Sample COBOL Program

(Continued from previous slide..)

```
01      OUTPUT_RECORD.  
        05 FILLER PICTURE X.  
        05 TITLE PICTURE X(25).  
        05 SUM PICTURE 9(10)V99.  
        05 FILLER PICTURE X(94).  
WORKING_STORAGE SECTION.  
77MESSAGE PICTURE X(25)  
VALUE IS "THE SUM OF GIVEN NUMBERS=".  
  
PROCEDURE DIVISION.  
OPEN_FILES.  
    OPEN INPUT DATA_FILE.  
    OPEN OUTPUT OUTPUT_FILE.  
  
INITIALIZATION.  
    MOVE SPACES TO OUTPUT_RECORD.  
    MOVE ZERO TO SUM.
```

(Continued on next slide)

# A Sample COBOL Program

(Continued from previous slide..)

PROCESS\_LOOP.

    READ DATA\_FILE AT END GO TO  
PRINT PARA.

    ADD N TO SUM.

    GO TO PROCESS\_LOOP.

PRINT PARA.

    MOVE MESSAGE TO TITLE.

    WRITE OUTPUT\_RECORD.

END\_OF\_JOB.

    CLOSE DATA\_FILE.

    CLOSE OUTPUT\_FILE.

    STOP RUN.

# BASIC

- § Stands for Beginners All-purpose Symbolic Instruction Code
- § Developed by Professor John Kemeny and Thomas Kurtz at Dartmouth College in the United States
- § Standardized by ANSI as BASIC-78
- § Designed to be an interactive language and to use an interpreter instead of a compiler
- § Simple to implement, learn and use language. Hence, it is a widely used language on personal computers
- § Flexible and reasonably powerful language and can be used for both business and scientific applications

# A Sample BASIC Program

```
5 REM PROGRAM TO COMPUTE
6 REM THE SUM OF 10 NUMBERS
10 LET S = 0
20 FOR I = 1 TO 10
30 READ N
40 LET S = S + N
50 NEXT I
60 PRINT "THE SUM OF GIVEN NUMBERS = "; S
70 DATA 4, 20, 15, 32, 48
80 DATA 12, 3, 9, 14, 44
90 END;
```

# Pascal

- § Named after the famous seventeenth-century French mathematician Blaise Pascal
- § Developed by Professor Nicklaus Wirth of Federal Institute of Technology in Zurich
- § Encourages programmers to write well-structured, modular programs, instills good program practices
- § Recognized as an educational language and is used to teach programming to beginners
- § Suitable for both scientific & business applications
- § Has features to manipulate numbers, vectors, matrices, strings, sets, records, files, and lists

# A Sample Pascal Program

PROGRAM SUMNUMS (INPUT, OUTPUT);  
(\* PROGRAM TO COMPUTE THE SUM OF 10 NUMBERS \*)

(\* DECLARATION OF VARIABLES \*)

VAR SUM, N : REAL;

VAR I : INTEGER;

(\* MAIN PROGRAM LOGIC STARTS HERE \*)

BEGIN

    SUM := 0;

    FOR I := 1 TO 10 DO

        BEGIN

            READ (N);

            SUM := SUM + N;

        END;

        WRITELN ('THE SUM OF GIVEN NUMBERS=', SUM);

    END;

# C

- § Developed in 1972 at AT&T's Bell laboratories, USA by Dennis Ritchie and Brian Kernighan
- § Standardized by ANSI and ISO as C89, C90, C99
- § High-level programming languages (mainly machine independence) with the efficiency of an assembly language
- § Language of choice of programmers for portable systems software and commercial software packages like OS, compiler, spreadsheet, word processor, and database management systems

# A Sample C Program

```
/* PROGRAM TO COMPUTE THE SUM OF 10 NUMBERS */
/* Directives to include standard library and header */
#include <stdlib.h>
#include <stdio.h>
/* Main function starts here */
void main ()
{
    /* Declaration of variables */
    float Sum = 0.0, N = 0.0;
    int Count = 0;
    for (Count = 0; Count < 10; Count++)
    {
        printf("\nGive a number:");
        scanf("%f", &N);
        Sum += N;
    }
    printf("THE SUM OF GIVEN NUMBERS = %f", &Sum);
}
```

# C++

- § Named C++ as ++ is increment operator and C language is incremented to its next level with C++
- § Developed by Bjarne Stroustrup at Bell Labs in the early 1980s
- § Contains all elements of the basic C language
- § Expanded to include numerous object-oriented programming features
- § Provides a collection of predefined classes, along with the capability of user-defined classes

# Java

- § Development started at Sun Microsystems in 1991 by a team led by James Gosling
- § Developed to be similar to C++ with fewer features to keep it simple and easy to use
- § Compiled code is machine-independent and developed programs are simple to implement and use
- § Uses *just-in-time* compilation
- § Used in embedded systems such as hand-held devices, telephones and VCRs
- § Comes in two variants – Java Runtime Engine (JRE) and Java Software Development Kit (SDK)

# C# (*C Sharp*)

- § Object-oriented programming language developed by Anders Hejlsberg and released by Microsoft as part of Microsoft's .NET technology initiative
- § Standardized by ECMA and ISO
- § Syntactically and semantically very close to C++ and adopts various object-oriented features from both C++ and Java
- § Compilers target the Common Language Infrastructure (CLI) implemented by Common Language Runtime (CLR) of .NET Framework
- § CLR provides important services such as, memory management, exception handling, and security

# RPG

- § Stands for Report Program Generator
- § Developed by IBM to meet customer requests for an easy and economic mechanism for producing reports
- § Designed to generate the output reports resulting from the processing of common business applications
- § Easier to learn and use as compared to COBOL
- § Programmers use very detailed coding sheets to write specifications about input, calculations, and output

# LISP

- § Stands for LISt Processing
- § Developed in 1959 by John McCarthy of MIT
- § Designed to have features for manipulating non-numeric data, such as symbols and strings of text
- § Due to its powerful list processing capability, it is extensively used in the areas of pattern recognition, artificial intelligence, and for simulation of games
- § Functional programming language in which all computation is accomplished by applying functions to arguments

# SNOBOL

- § Stands for String Oriented symbolic Language
- § Used for non-numeric applications
- § Powerful string manipulation features
- § Widely used for applications in the area of text processing

# Characteristics of a Good Programming Language

- § Simplicity
- § Naturalness
- § Abstraction
- § Efficiency
- § Structured Programming Support
- § Compactness
- § Locality
- § Extensibility
- § Suitability to its environment

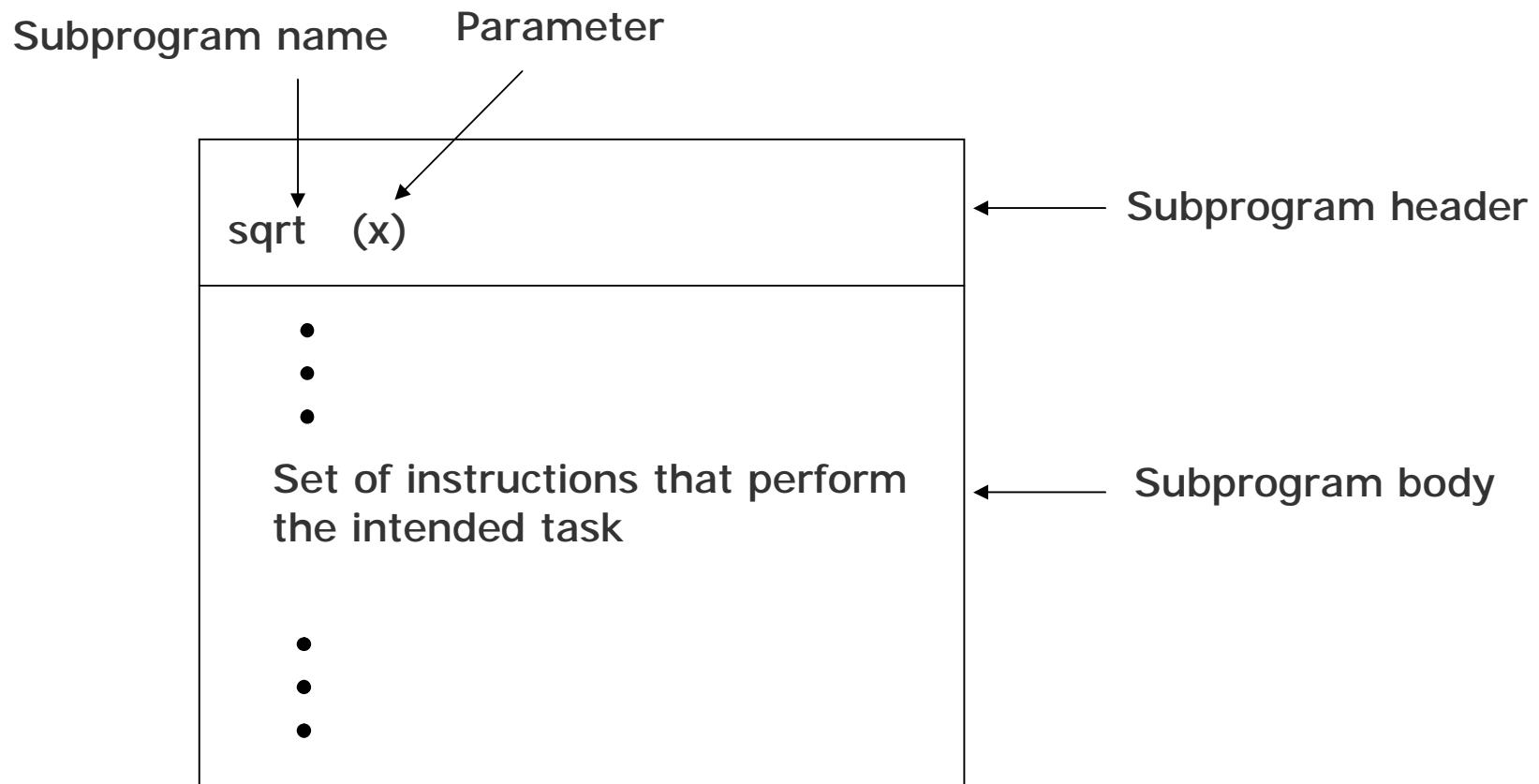
# Factors for Selecting a Language for Coding an Application

- § Nature of the application
- § Familiarity with the language
- § Ease of learning the language
- § Availability of program development tools
- § Execution efficiency
- § Features of a good programming language

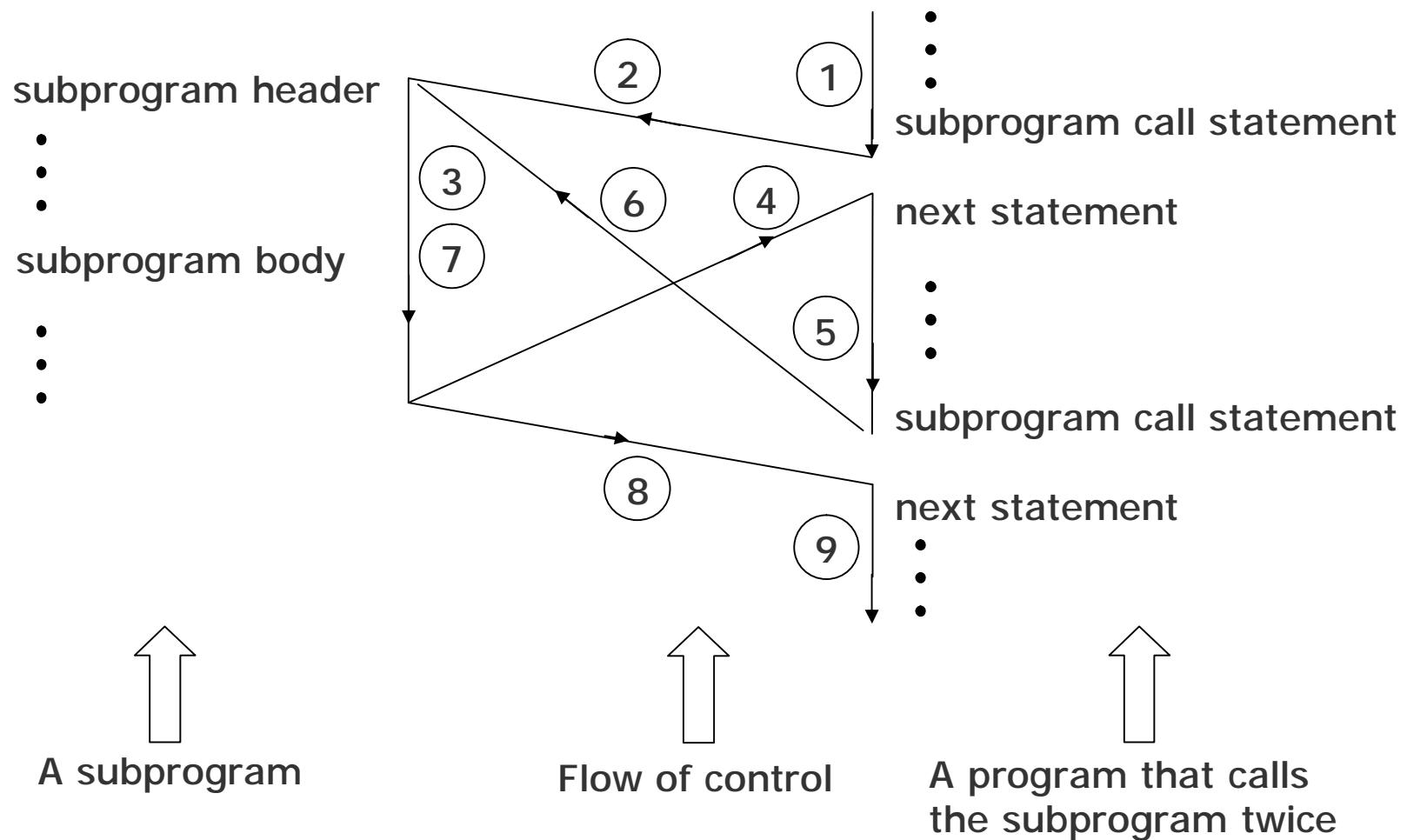
# Subprogram

- § Program written in a manner that it can be brought into use in other programs and used whenever needed without rewriting
- § Also referred to as *subroutine*, *sub-procedure*, or *function*
- § Subprogram call statement contains the name of the subprogram followed by a list of parameters enclosed within a pair of parentheses
- § *Intrinsic subprograms* (also called *built-in-functions*) are those provided with the programming language
- § *Programmer-written subprograms* are written and used as and when they are needed

# Structure of a Subprogram



# Flow of Control in Case of Subprogram Calls



# Key Words/Phrases

- |                             |                               |
|-----------------------------|-------------------------------|
| § Assembler                 | § Just-in-time compilation    |
| § Assembly language         | § Language processor          |
| § BASIC                     | § Linker                      |
| § Built-in function         | § LISP                        |
| § C                         | § Load module                 |
| § C++                       | § Logic error                 |
| § C#                        | § Low-level language          |
| § COBOL                     | § Machine language            |
| § Coding                    | § Macro instructions          |
| § Compiler                  | § Object program              |
| § Computer language         | § Object-oriented programming |
| § FORTRAN                   | § Opcode                      |
| § Function                  | § Operand                     |
| § High-level language       | § Pascal                      |
| § HotJava Interpreter       | § Programmer                  |
| § Intrinsic subprogram      | § Programming                 |
| § Intermediate compiler and | § Programming language        |
| § Interpreter               | § Pseudo instruction          |
| § Java                      | § RPG                         |
|                             | § Self-documenting language   |

(Continued on next slide)

# Key Words/Phrases

(Continued from previous slide..)

- § SNOBOL
- § Source program
- § Sub-procedure
- § Subprogram
- § Subroutine
- § Symbolic language
- § Syntax error
- § Syntax rules
- § Written subprograms

# Chapter 13

## System Implementation and Operation

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

In this chapter you will learn about:

- § Main activities of implementation and operation phase
- § Testing and debugging of programs
- § Complete documentation of the system
- § Change over to the new system
- § System evaluation and
- § System maintenance

# Testing and Debugging

- § Program errors are known as *bugs*
- § Process of detecting and correcting these errors is called *debugging*
- § *Testing* is the process of making sure that the program performs the intended task
- § *Debugging* is the process of locating and eliminating program errors

# Types of Program Errors

## § *Syntax errors*

- § Occurs when the rules or syntax of the programming language are not followed
- § For example, incorrect punctuation, incorrect word sequence, undefined terms, and misuse of terms
- § Syntax errors are detected by a language processor

## § *Logic errors*

- § Occurs due to errors in planning a program's logic
- § Such errors cause the program to produce incorrect output.
- § These errors cannot be detected by a language processor

# Testing of a Program

- § Testing procedure involves running program to process input test data, and comparing obtained results with correct results
- § Test data must test each logical function of the program, and should include all types of possible valid and invalid data
- § Program internally released for testing is known as *alpha version* and the test conducted on it is called *alpha testing*
- § Program released for additional testing to a selected set of external users is *beta version* and test conducted on it called is *beta testing*

# Debugging a Program for Syntax Errors

- § Relatively easier to detect and correct syntax errors than logic errors in a program
- § Language processors are designed to automatically detect syntax errors
- § Single syntax error often causes multiple error messages to be generated by the language processor
- § Removal of the syntax error will result in the removal of all associated error messages

# Debugging a Program for Logic Errors

- § Logic errors are more difficult to detect than syntax errors as computer does not produce any error message for such errors
- § One or more of following methods are commonly used for locating logic errors:
  - § Doing hand simulation of the program code
  - § Putting print statements in the program code
  - § Using a debugger (a software tool that assists a programmer in following the program's execution step-by-step)
  - § Using memory dump (printout of the contents of main memory and registers)

# Difference Between Testing and Debugging

Sr. No.	Testing	Debugging
1	<ul style="list-style-type: none"> <li>§ Testing is the process of validating the correctness of a program</li> <li>§ Its objective is to demonstrate that the program meets its design specifications</li> </ul>	<ul style="list-style-type: none"> <li>§ Debugging is the process of eliminating errors in a program</li> <li>§ Its objective is to detect the exact cause and remove known errors in the program</li> </ul>
2	<ul style="list-style-type: none"> <li>§ Testing is complete when all desired verifications against specifications have been performed</li> </ul>	<ul style="list-style-type: none"> <li>§ Debugging is complete when all known errors in the program have been fixed</li> <li>§ Note that debugging process ends only temporarily as it must be restarted whenever a new error is found in the program</li> </ul>

(Continued on next slide)

# Difference Between Testing and Debugging

(Continued from previous slide..)

Sr. No.	Testing	Debugging
3	<ul style="list-style-type: none"> <li>§ Testing is a definable process which can and should be planned and scheduled properly</li> </ul>	<ul style="list-style-type: none"> <li>§ Debugging being a reactive process cannot be planned ahead of time</li> <li>§ It must be carried out as and when errors are found in a program</li> </ul>
4	<ul style="list-style-type: none"> <li>§ Testing can begin in the early stages of software development.</li> <li>§ Although the test runs of a program can be done only after the program is coded, but the decision of what to test, how to test, and with what kind of data to test, can and should be done before the coding is started</li> </ul>	<ul style="list-style-type: none"> <li>§ Debugging can begin only after the program is coded</li> <li>§ The approach used for debugging largely depends on the personal choice of the programmer and the type of problem in the program</li> </ul>

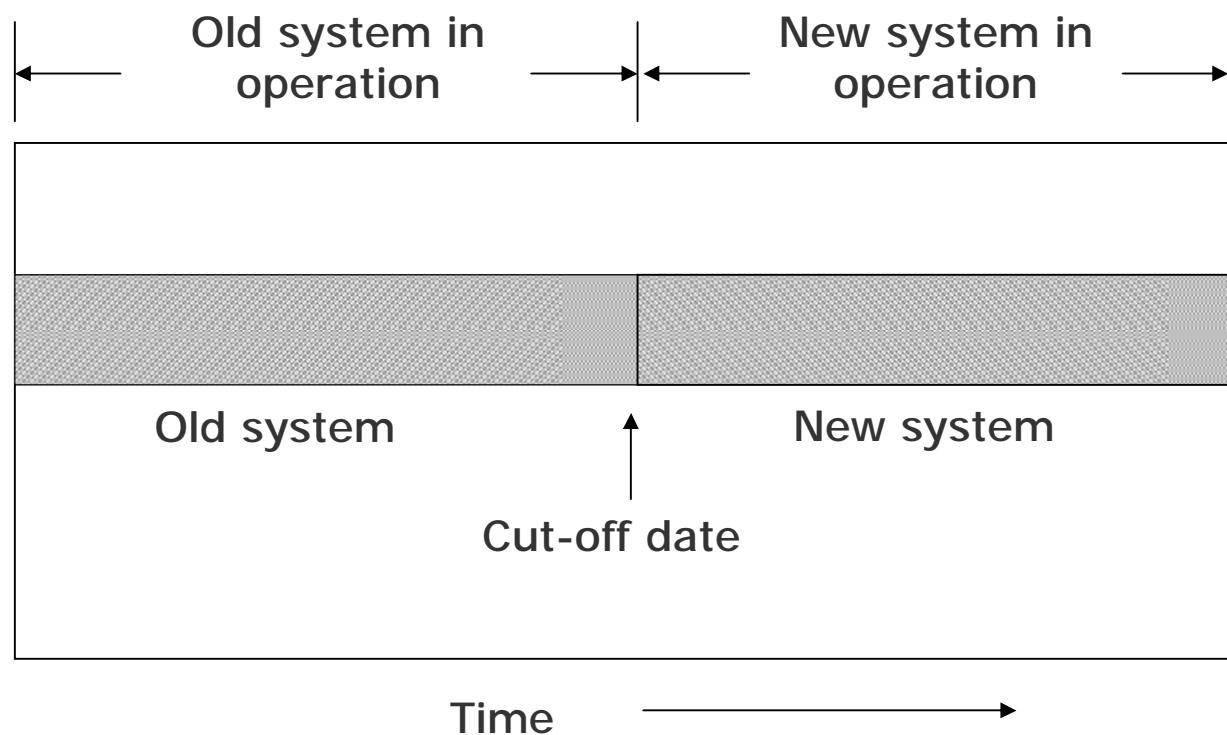
# Documentation

- § Process of collecting, organizing, storing, and otherwise maintaining a complete historical record of programs and other documents used or prepared during the different phases of the life cycle of a software
- § Three commonly used forms of documentation are:
  - § Program comments
  - § System manual
  - § User manual

# Changeover to the New System

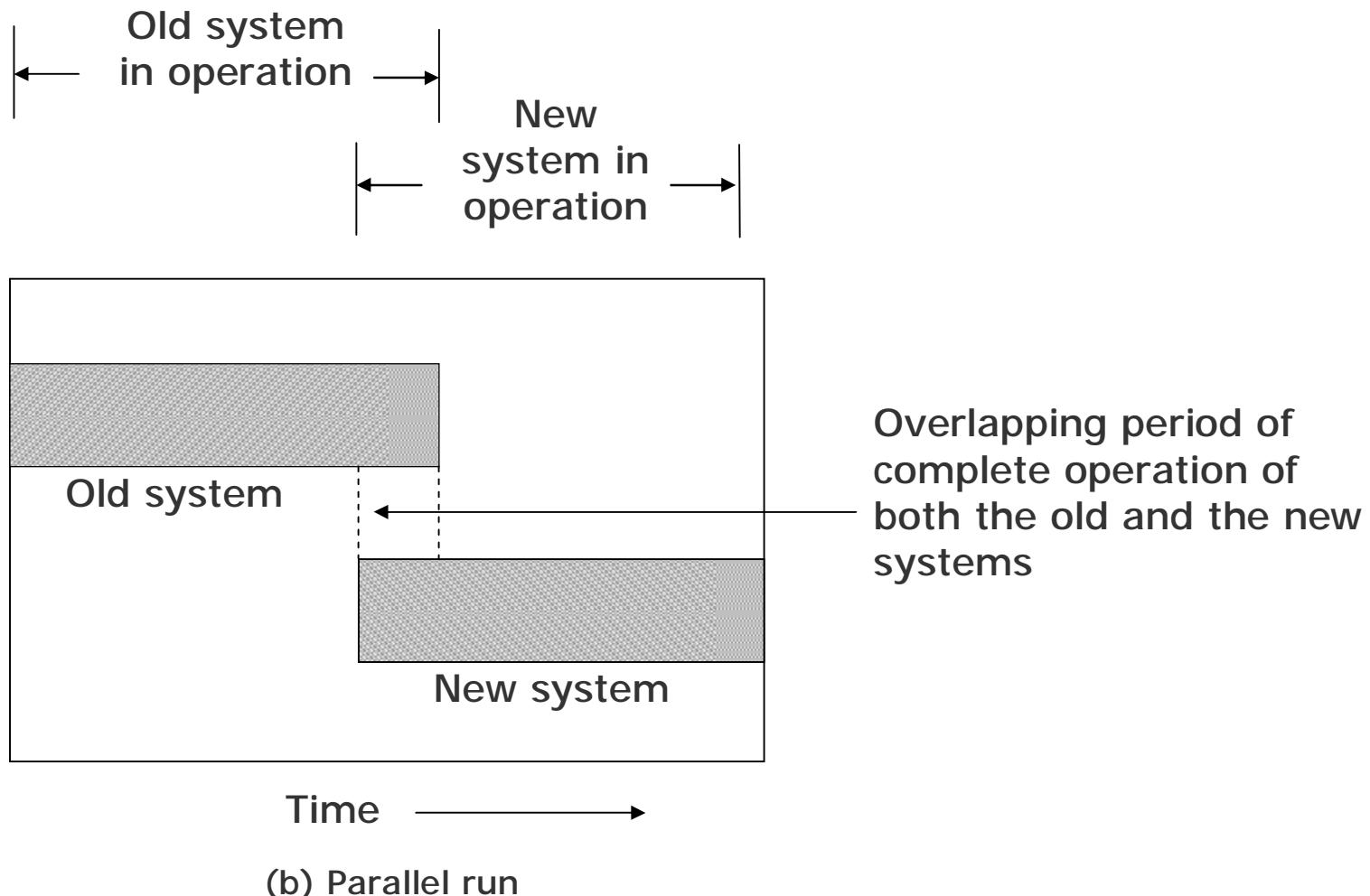
- § When a software is ready for use, it is deployed at site for use by the intended users
- § At this stage, a changeover from the old system of operation to the new system takes place
- § Three normally followed methods to carry out the changeover process are:
  - § Immediate changeover
  - § Parallel run
  - § Phased conversion

# Changeover to the New System

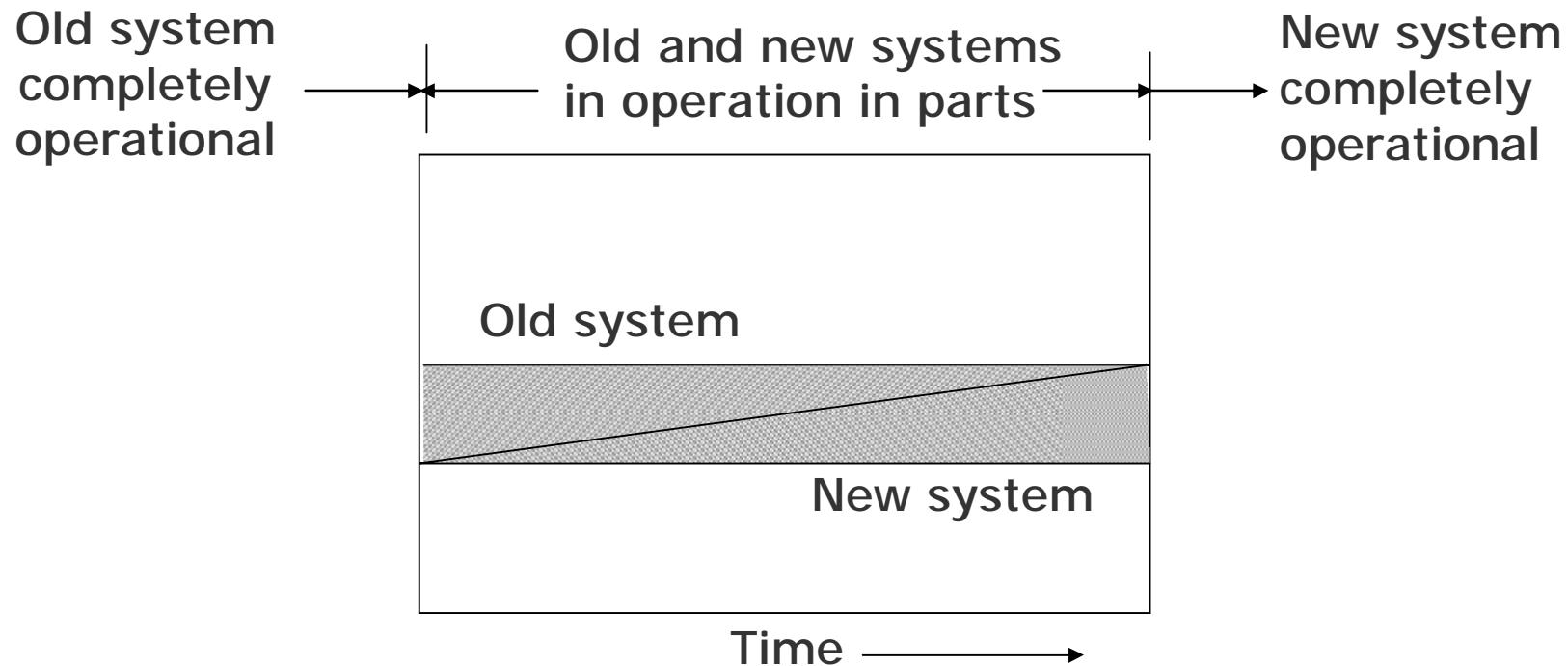


(a) Immediate changeover

# Changeover to the New System



# Changeover to the New System



(c) Phased conversion

# System Evaluation

- § Process of evaluating a system (after it is put in operation) to verify whether or not it is meeting its objectives
- § Points normally considered for evaluating a system are:
  - § Performance evaluation
  - § Cost analysis
  - § Time analysis
  - § User satisfaction
  - § Ease of modification
  - § Failure rate

# System Maintenance

- § Process of incorporating changes in an existing system to enhance, update, or upgrade its features
- § On an average, maintenance cost of a computerized system is two to four times more than the initial development cost

# Key Words/Phrases

- § Beta testing
- § Bugs
- § Comments
- § Debugger
- § Debugging
- § Documentation
- § Immediate changeover
- § Logic errors
- § Memory dump
- § Parallel run
- § Phased conversion
- § Syntax errors
- § System evaluation
- § System maintenance
- § System manual
- § Testing
- § User manual

# Chapter 14

# Operating Systems

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

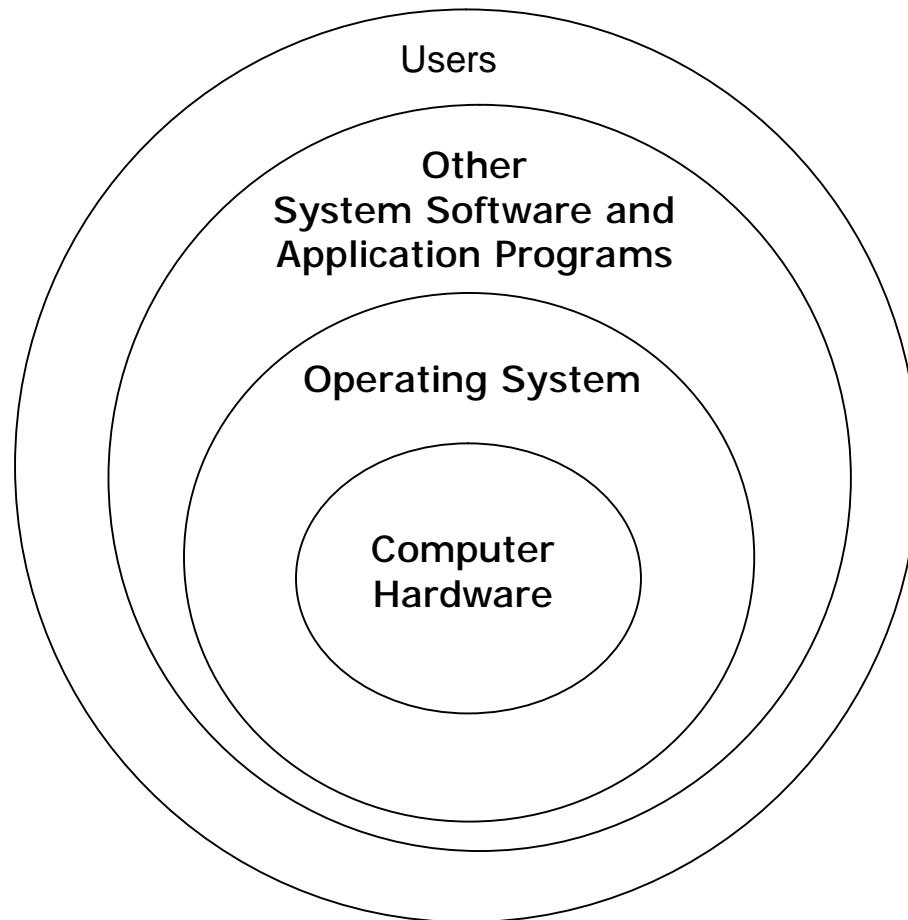
In this chapter you will learn about:

- § Definition and need for operating system
- § Main functions of an operating system
- § Commonly used mechanisms for:
  - § Process management
  - § Memory management
  - § File management
  - § Security
  - § Command interpretation module
- § Some commonly used OS capability enhancement software
- § Some popular operating systems

# Definition and Need for OS

- § Integrated set of programs that controls the resources (the CPU, memory, I/O devices, etc.) of a computer system
- § Provides its users with an interface or virtual machine that is more convenient to use than the bare machine
- § Two primary objectives of an OS are:
  - § Making a computer system convenient to use
  - § Managing the resources of a computer system

# Logical Architecture of a Computer System



The operating system layer hides the details of the hardware from the programmer and provides the programmer with convenient interface for using the system

# Main Functions of an OS

- § Process management
- § Memory management
- § File management
- § Security
- § Command interpretation

# Parameters for Measuring System Performance

- § **Throughput:** Amount of work that the system is able to do per unit time
- § **Turnaround time:** Interval from the time of submission of a job to the system for processing to the time of completion of the job
- § **Response time:** Interval from the time of submission of a job to the system for processing to the time the first response for the job is produced by the system

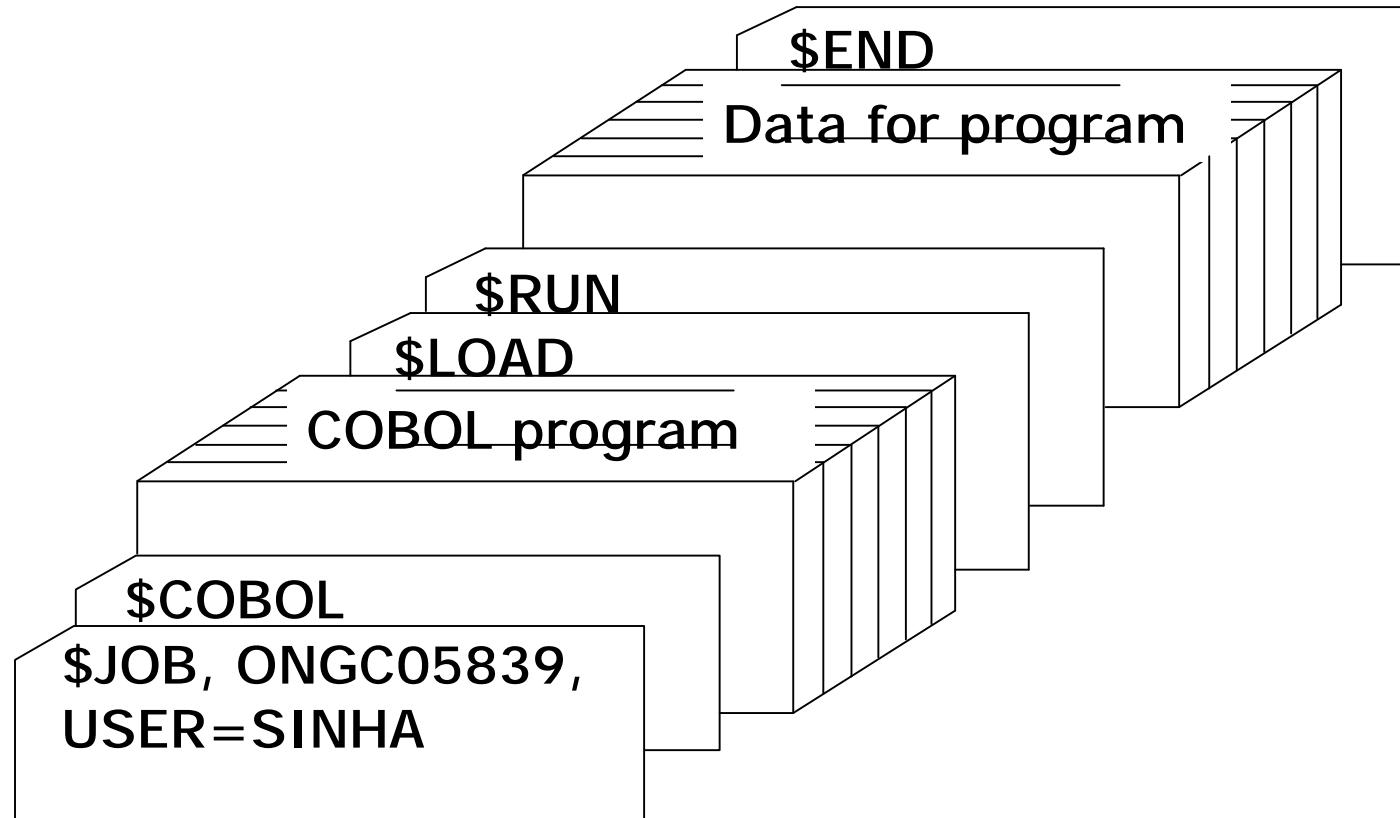
# Process Management

- § A process (also called job) is a program in execution
- § Process management manages the processes submitted to a system in a manner to minimize *idle time* of processors (CPUs, I/O processors, etc.) of the system

# Process Management Mechanisms in Early Systems

- § **Manual loading mechanism:** Jobs were manually loaded one after another in a computer by the computer operator
- § **Batch processing mechanism:** Batch of jobs was submitted together to the computer and job-to-job transition was done automatically by the operating system
- § **Job Control Language (JCL):** Control statements were used to facilitate job loading and unloading

# Use of Job Control Statements in Batch Processing (An Example)



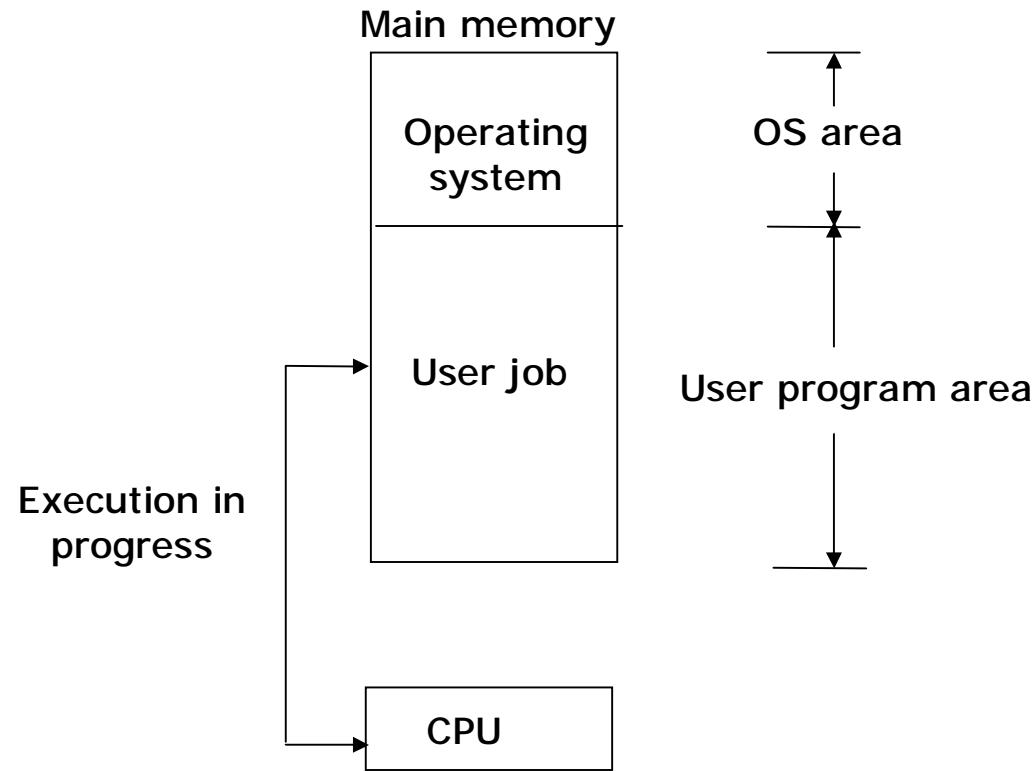
# Multiprogramming

- § **Uniprogramming:** Only one job is processed at a time and all system resources are available exclusively for the job until its completion
- § **Multiprogramming:** Interleaved execution of two or more different and independent programs by a computer
- § **Types of Multiprogramming:**
  - § ***Multiprogramming with fixed tasks (MFT)*:** Fixed number of jobs can be processed concurrently
  - § ***Multiprogramming with variable tasks (MVT)*:** Number of jobs can vary
- § Area occupied by each job residing simultaneously in the main memory is known as a **memory partition**

# Job

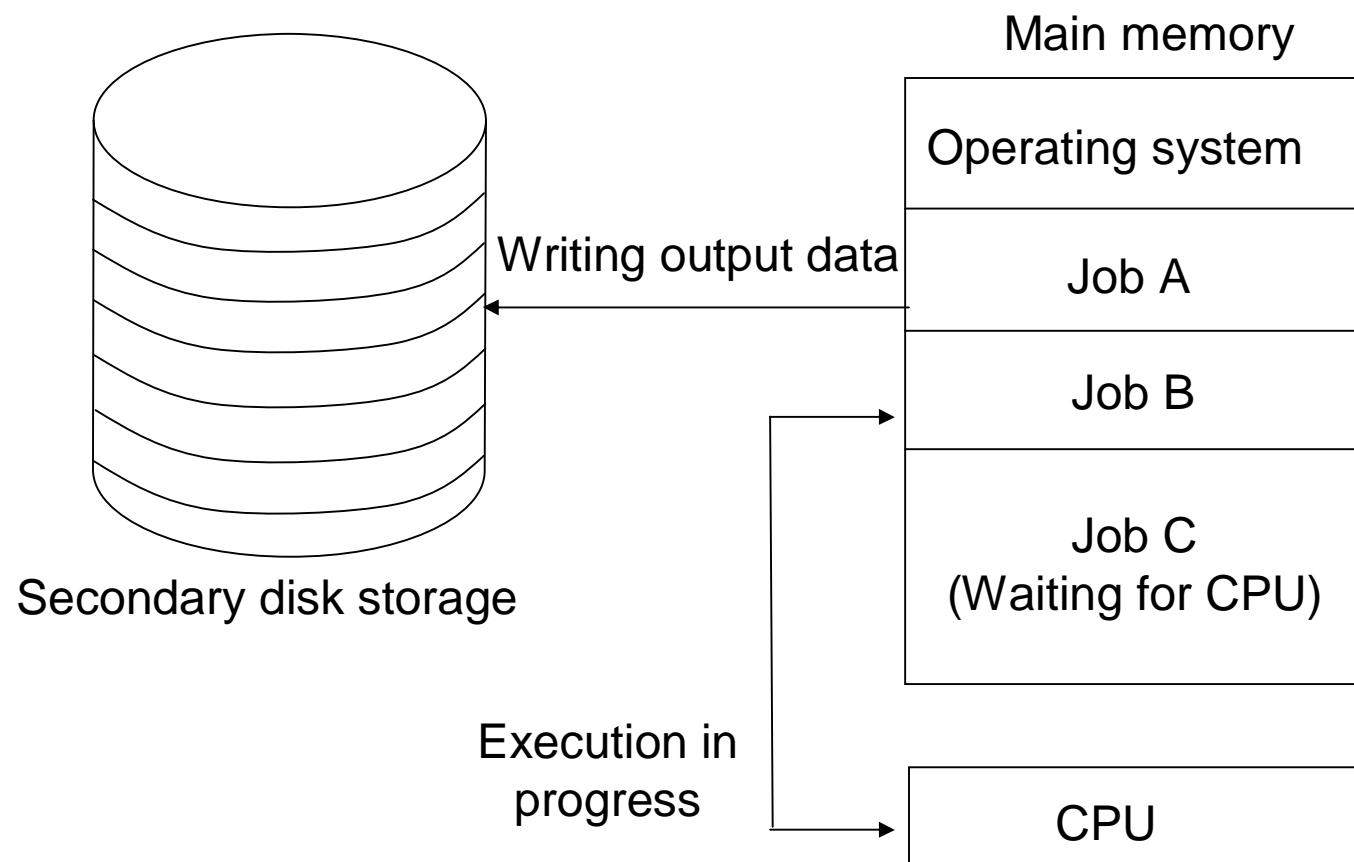
- § **CPU bound:** Mostly perform computations with little I/O operations. Scientific and engineering computations usually fall in this category
- § **I/O bound:** Mostly perform I/O operations with little computation. Commercial data processing applications usually fall in this category

# Uniprogramming System

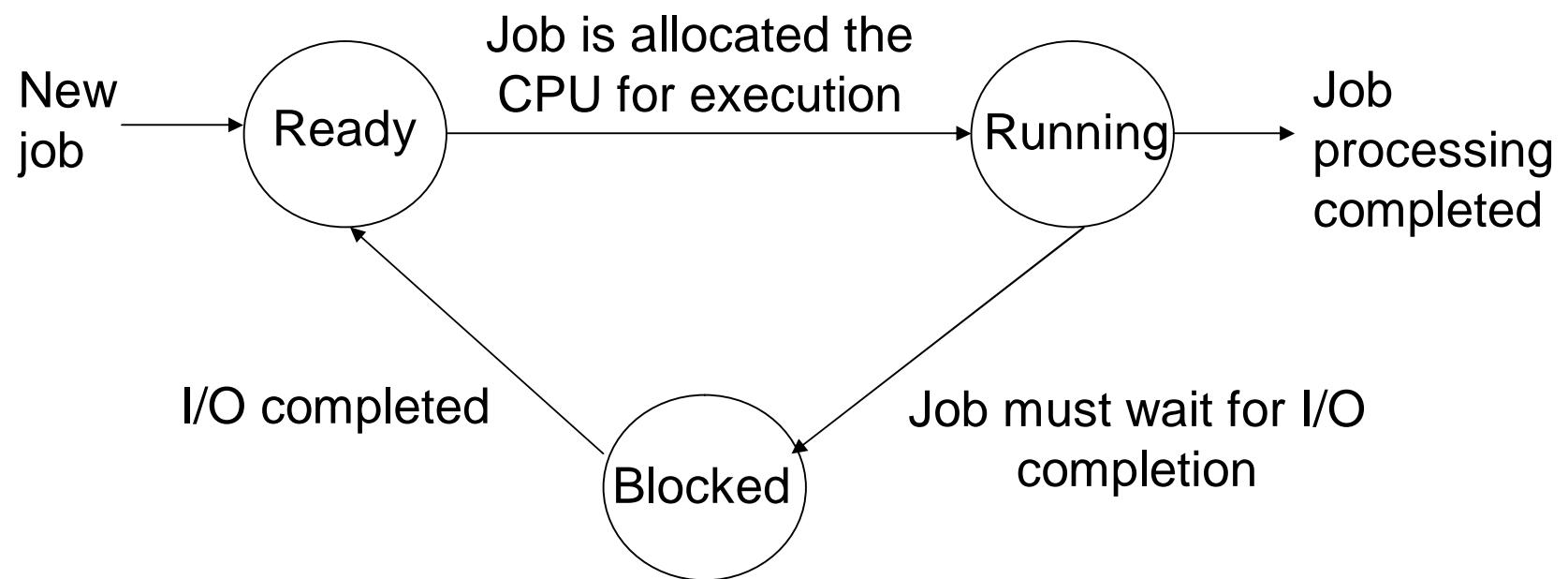


Only one job is processed by the system at a time and all the system resources are exclusively available for the job until it completes

# Multiprogramming System



# Process States in Multiprogramming



# Requirements of Multiprogramming Systems

- § Large memory
- § Memory protection
- § Job status preservation
- § Proper job mix (CPU and I/O bound jobs)
- § CPU scheduling

# Process Control Block (PCB)

process identifier
process state
program counter
values of various CPU registers
accounting and scheduling information
I/O status information
•
•
•

PCB is used to preserve the job status of each loaded process in a multiprogramming system

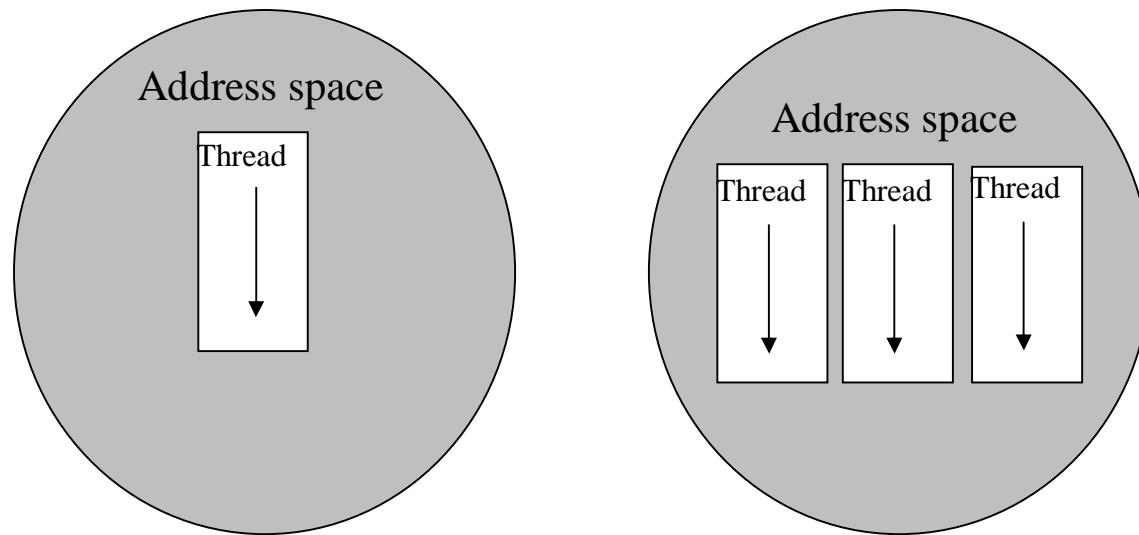
# Multitasking

- § Interleaved execution of multiple jobs (often referred to as *tasks* of same user) in a single-user system
- § Computer systems used for multitasking are uniprocessor systems (having only one CPU)
- § Treated differently from multiprogramming that refers to interleaved execution of multiple jobs in a multi-user system

# Multithreading

- § Thread is basic unit of CPU utilization. Threads share a CPU in the same way as processes do
- § All threads of a process also share the same set of operating system resources
- § All threads of a process inherit parent's address space and security parameters
- § Each thread of a process has its own program counter, its own register states, and its own stack
- § Referred as mini-process or lightweight process

# Multithreading System

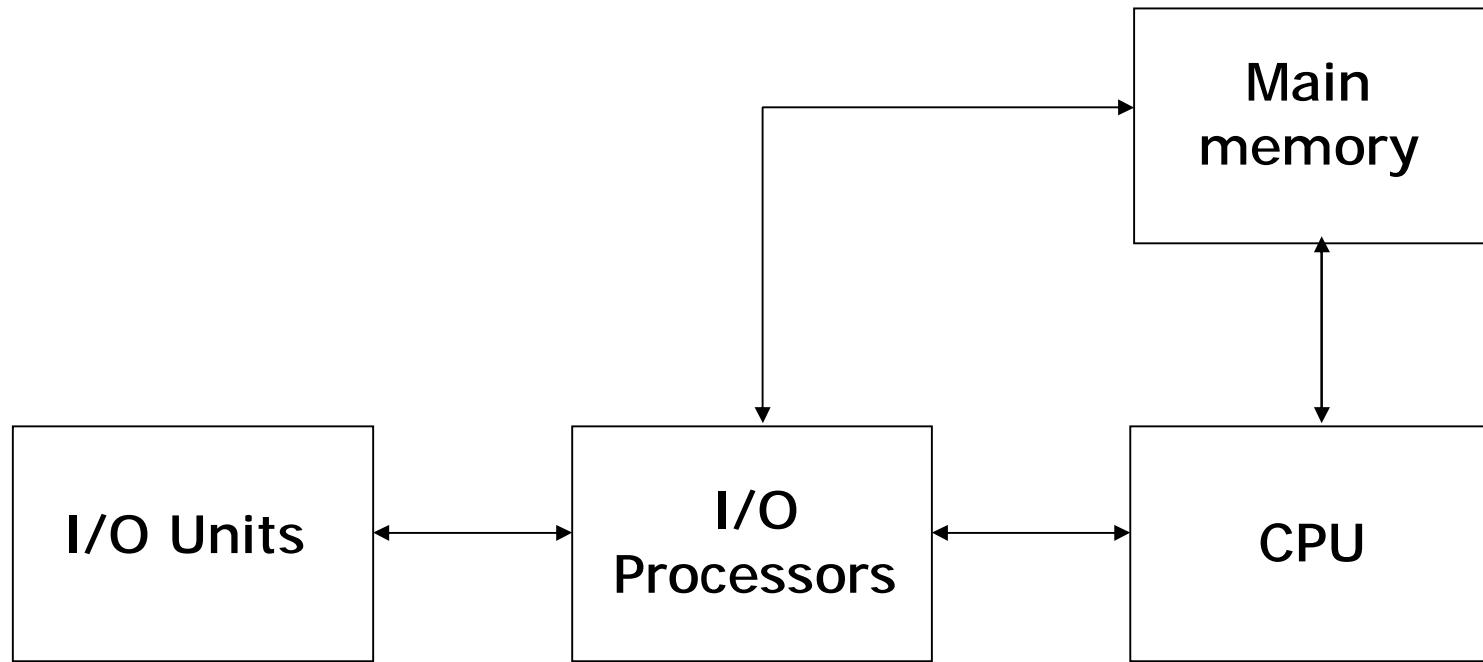


(a) Single-threaded and (b) multithreaded processes. A single-threaded process corresponds to a process of a traditional operating system. [Reproduced with permission, from the book titled *Distributed Operating Systems: Concepts and Design* by Pradeep K. Sinha. © 1997 IEEE, USA].

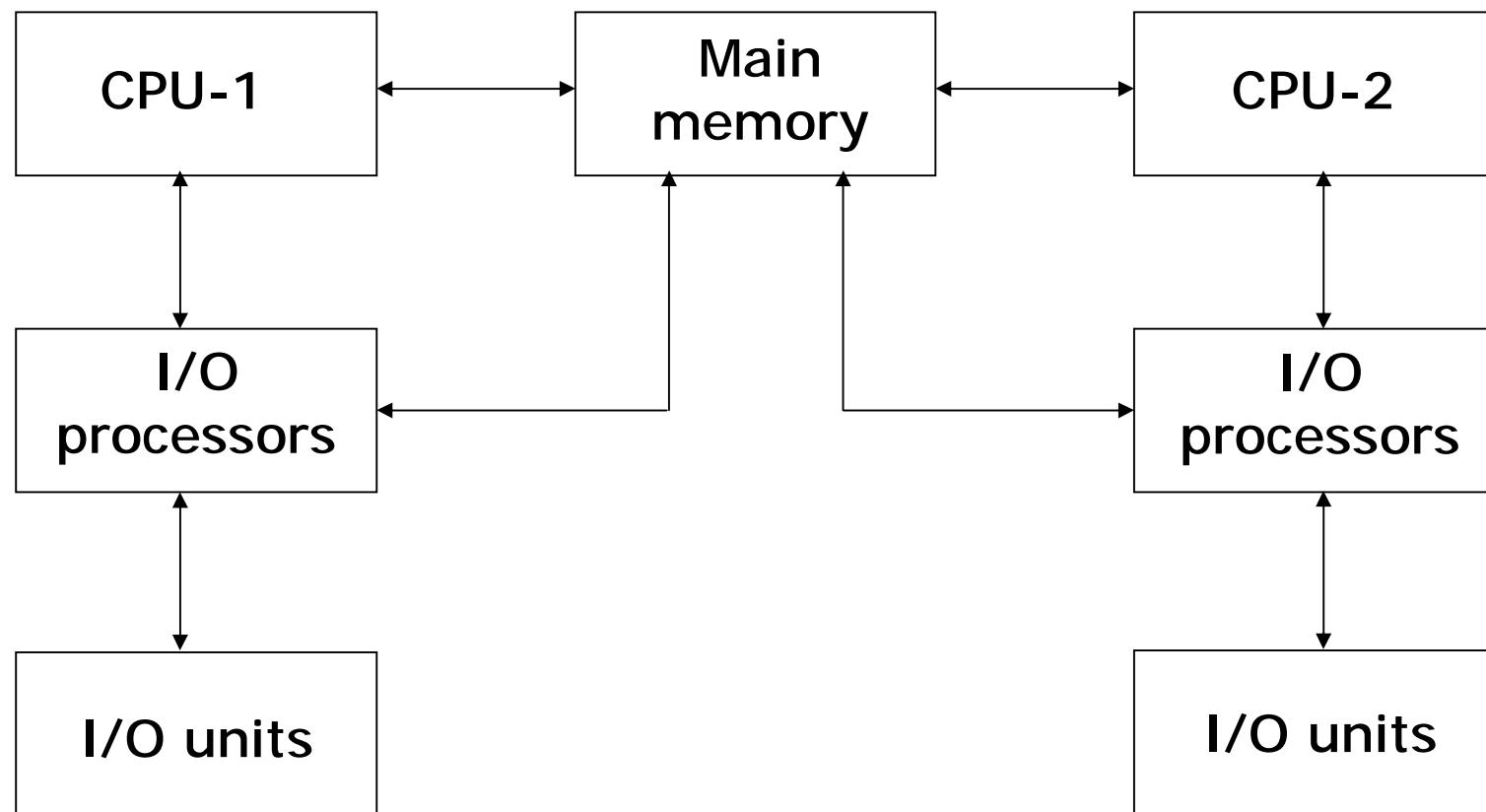
# Multiprocessing

- § System with two or more CPUs having ability to execute multiple processes concurrently
- § Multiple CPUs are used to process either instructions from different and independent programs or different instructions from the same program simultaneously
- § Types of multiprocessing:
  - § *Tightly-coupled*: Single system-wide primary memory shared by all processors
  - § *Loosely-coupled*: Each processor has its own local memory

# CPU, Memory, and I/O Processors of a Computer System



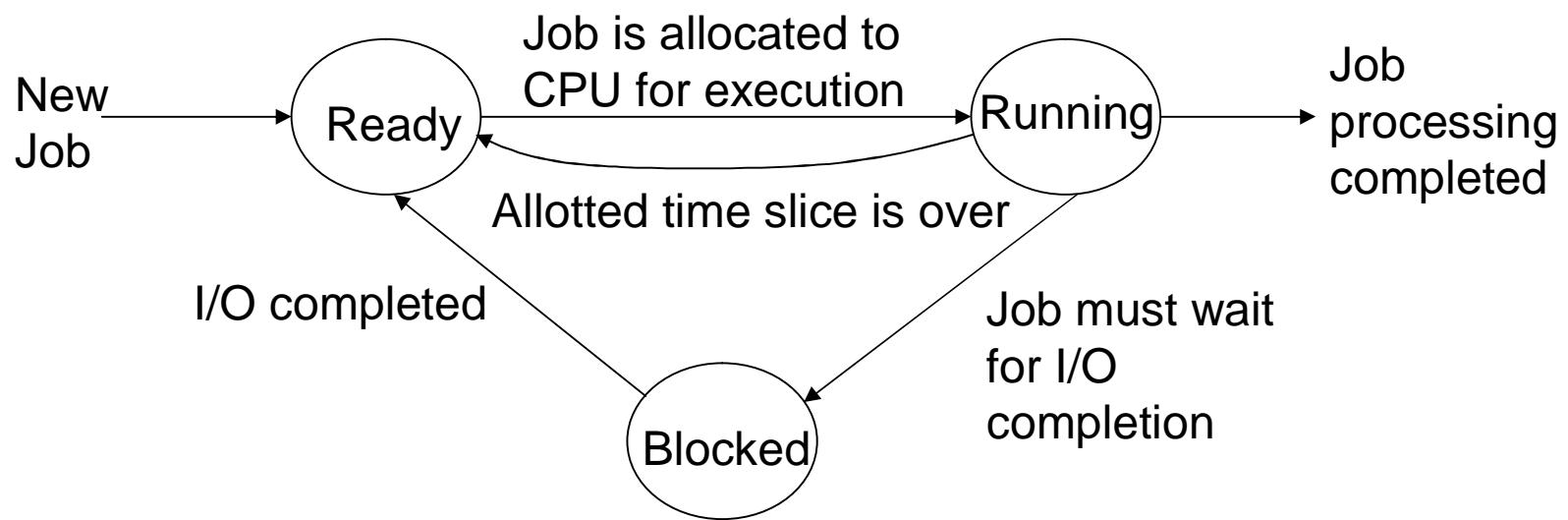
# Multiprocessing System



# Time-sharing

- § Simultaneous interactive use of a computer system by many users in such a way that each one feels that he/she is the sole user of the system
- § User terminals connected to the same computer simultaneously
- § Uses multiprogramming with a special CPU scheduling algorithm
- § Short period during which a user process gets to use CPU is known as time slice, time slot, or quantum
- § CPU is taken away from a running process when the allotted time slice expires

# Process State Diagram for a Time-Sharing System



# Advantages of Time-sharing Systems

- § Reduces CPU idle time
- § Provides advantages of quick response time
- § Offers good computing facility to small users

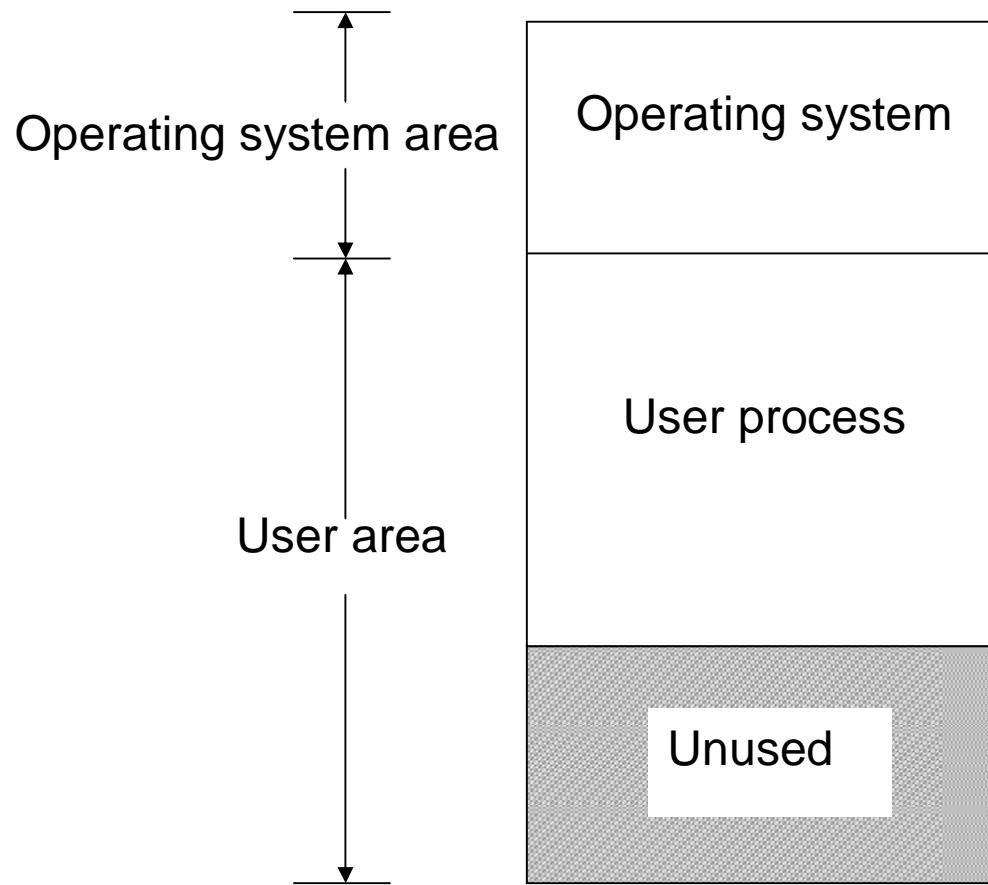
# Memory Management

- § Memory is important resource of a computer system that must be properly managed for the overall system performance
- § Memory management module:
  - § Keeps track of parts of memory in use and parts not in use
  - § Allocates memory to processes as needed and deallocates when no longer needed

# Uniprogramming Memory Model

- § Used in systems that process one job only at a time, and all system resources are available exclusively for the job until it completes
- § Simple and easy to implement
- § Does not lead to proper utilization of the main memory as unoccupied memory space by the currently active user process remains unused
- § Used only on very small or dedicated computer systems

# Uniprogramming Memory Model

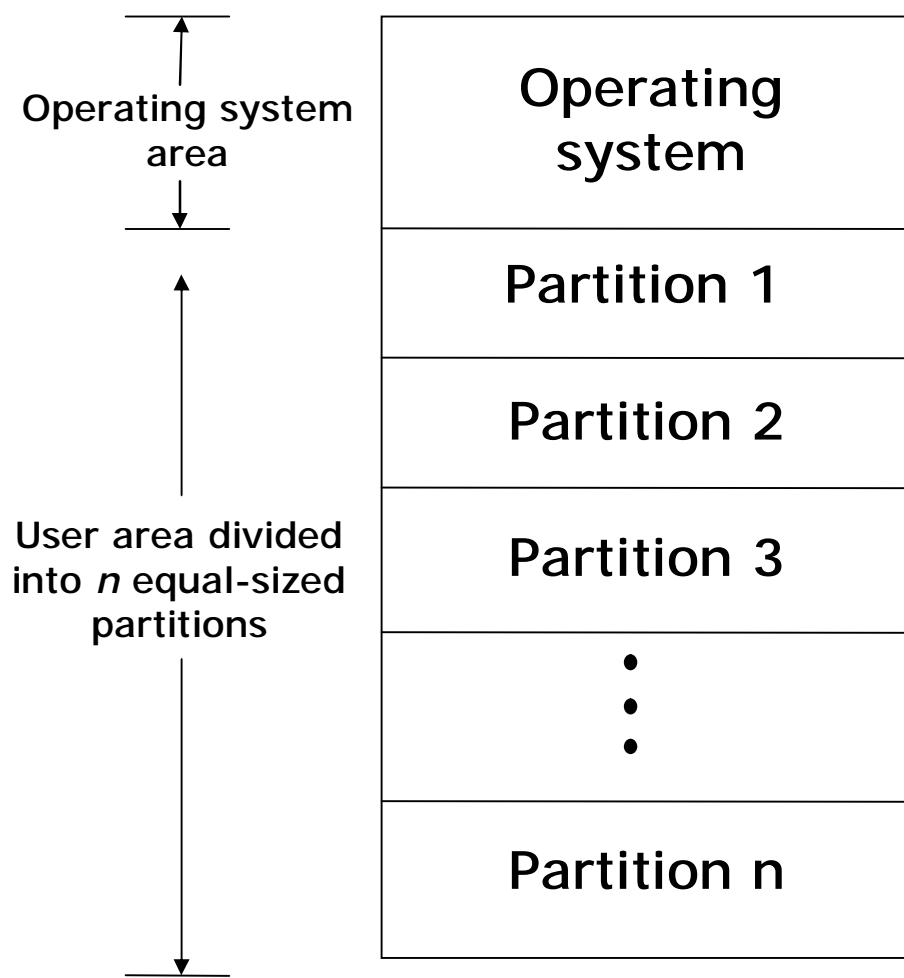


# Multiprogramming Memory Models

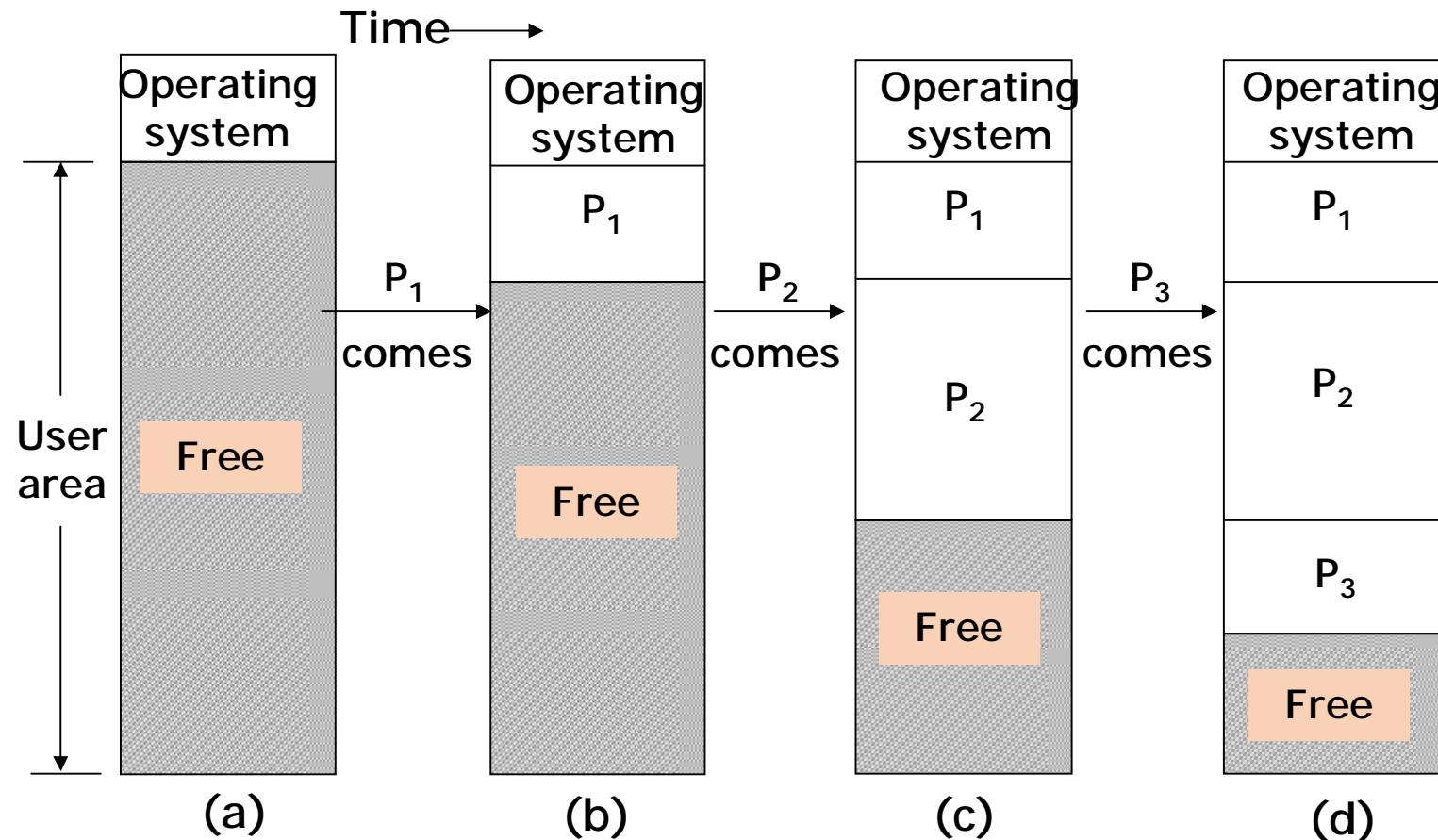
Two memory management schemes used to facilitate this are:

- § *Multiprogramming with fixed number of memory partitions*: User area of the memory is divided into a number of fixed-sized partitions
- § *Multiprogramming with variable number of memory partitions*: Number, size and location of the partitions vary dynamically as processes come and go

# Multiprogramming with Fixed Number of Memory Partition

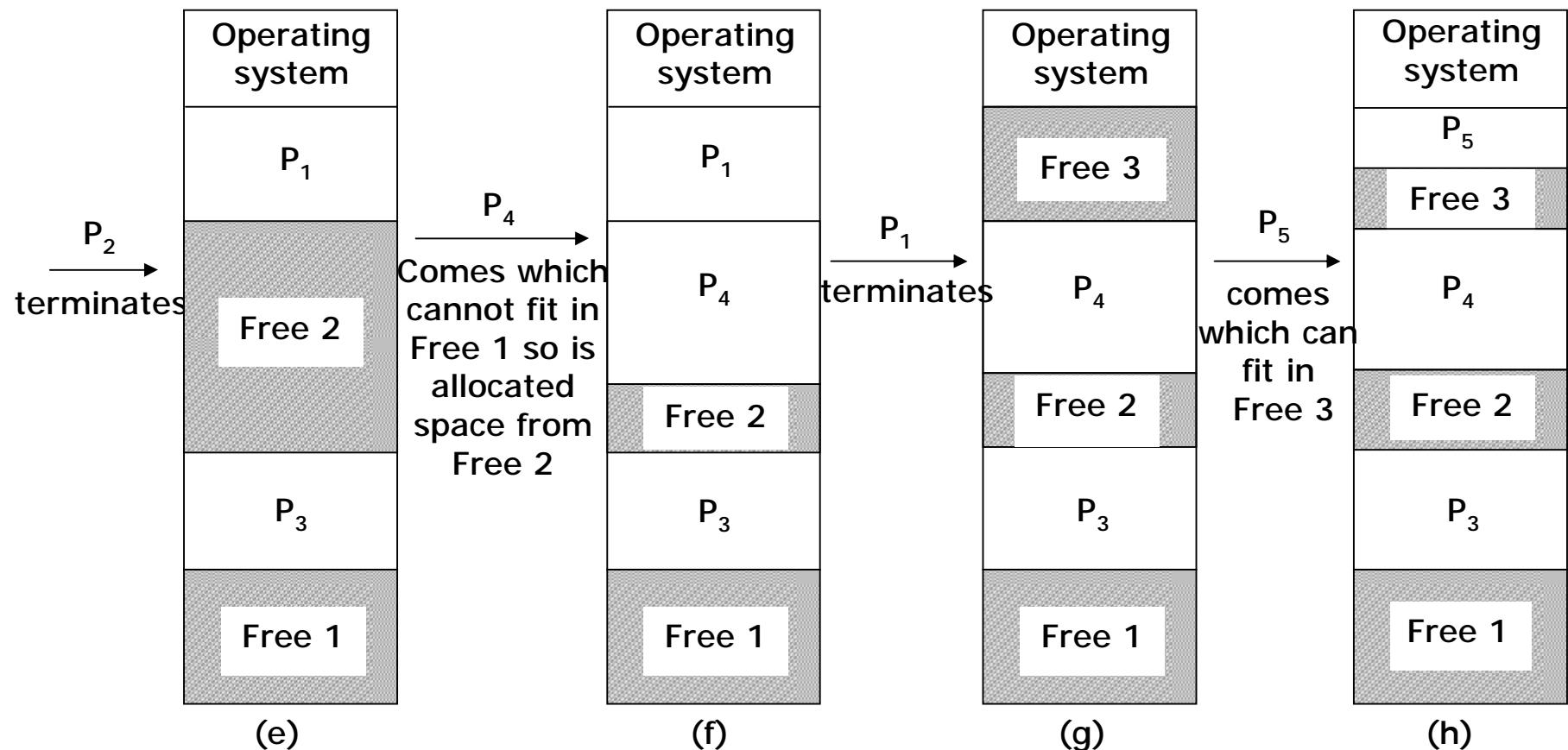


# Multiprogramming with Variable Number of Memory Partitions



The number, size, and location of the partitions vary dynamically as processes come and go. (contd...)

# Multiprogramming with Variable Number of Memory Partitions



The number, size, and location of the partitions vary dynamically as processes come and go.

# Virtual Memory

Memory management scheme that allows execution of processes that might not be completely loaded in the main memory.

It does not require the entire process to be in memory before the process can execute

# Virtual Memory Realization

Three basic concepts used for its realization are:

- § **On-line secondary storage:** Used to keep a process's address space ready to be loaded into the memory
- § **Swapping:** Process of transferring a block of data from the on-line secondary storage to main memory (swapping in) or vice-versa (swapping out)
- § **Demand paging:** Scheme of swapping in of pages of a process as and when needed during execution of the process, rather than loading all the pages before starting the process's execution

# Advantages of Virtual Memory

- § Provides a large virtual memory to programmers on a system having smaller physical memory
- § Enables execution of a process on a system whose main memory size is less than the total memory required by the process
- § Enables a process's execution to be started even when sufficient free memory for loading the entire process is not available
- § Makes programming easier there no longer need to worry about the memory size limitations
- § Often leads to less I/O activity resulting in better throughput, turnaround time, and response time

# Disadvantages of Virtual Memory

- § Difficult to implement because it requires algorithms to support demand paging
- § If used carelessly, it may substantially decrease performance due to high page fault rate

# File Management

- § A file is a collection of related information
- § Every file has a name, its data and attributes
- § File's name uniquely identifies it in the system and is used by its users to access it
- § File's data is its contents
- § File's attributes contain information such as date & time of its creation, date & time of last access, date & time of last update, its current size, its protection features, etc.
- § File management module of an operating system takes care of file-related activities such as structuring, accessing, naming, sharing, and protection of files

# File Access Methods

Two commonly supported file access methods are:

- § **Sequential access:** Information stored in a file can be accessed sequentially (in the order in which they are stored, starting at the beginning)
- § **Random access:** Information stored in a file can be accessed randomly irrespective of the order in which the bytes or records are stored

# File Operations

- § Set of commands provided by an operating system to deal with files and their contents
- § Typical file operations include create, delete, open, close, read, write, seek, get attributes, set attributes, rename, and copy

# File Naming

File naming deals with the rules for naming files in an operating system. This may include such rules as:

- § Maximum number of characters that a file name may have
- § Special characters allowed in a file name
- § Distinction between upper case and lower case letters
- § Multi-part file names allow file extensions to be part of a file name. File extensions indicate something about the file and its content
- § Used by applications to check for the intended type of file before operating on it

# File Extensions (Example)

File extension	Its meaning
.bas	Basic source program file
.c	C source program file
.ftn	Fortran source program file
.pas	Pascal source program file
.obj	Object file (compiler output, not yet linked)
.bin	Executable binary program file
.lib	Library of .obj files used by the linker
.dat	Data file
.hlp	Text file for HELP command
.man	Online manual page file

(Continued on next slide)

# File Extensions (Example)

(Continued from previous slide)

File extension	Its meaning
.man	Online manual page file
.txt	General text file
.bak	Backup file
.doc	Microsoft word document file
.wav	Microsoft windows sound file
.wk4	Lotus 1-2-3 spreadsheet file
.xls	Microsoft Excel spreadsheet file
.jpg	JPEG graphics file
.gif	GIF graphics file

# Security

- § Deals with protecting the various resources and information of a computer system against destruction and unauthorized access
- § **External security:** Deals with securing computer against external factors such as fires, floods, earthquakes, stolen disks/tapes, etc. by maintaining adequate backup, using security guards, allowing access to sensitive information to only trusted employees/users, etc.
- § **Internal security:** Deals with user authentication, access control, and cryptography mechanisms

# Security

- § **User authentication:** Deals with the problem of verifying the identity of a user (person or program) before permitting access to the requested resource
- § **Access Control:** Once authenticated, access control mechanisms prohibit a user/process from accessing those resources/information that he/she/it is not authorized to access
- § **Cryptography:** Means of encrypting private information so that unauthorized access cannot use information

# Command Interpretation

- § Provides a set of commands using which the user can give instructions to the computer for getting some job done by it
- § Commands supported by the command interpretation module are known as **system calls**

(Continued on next slide)

# Command Interpretation

(Continued from previous slide)

Two types of user interfaces supported by various operating systems are:

- § **Command-line interface:** User gives instructions to the computer by typing the commands
- § **Graphical User Interface (GUI):** User gives commands to the system by selecting icon or menu item displayed on the screen with the use of a point-and-draw device

# OS Capability Enhancement Software

- § Perform several tasks of routine nature, frequently needed by users but are not provided as part of the OS
- § They are primarily grouped into three categories:
  - § **Translating programs:** Translate a source program into an object program
  - § **Library programs:** Consist of frequently used functions and operations
  - § **Utility programs:** Assist users with system maintenance tasks such as disk formatting, data compression, data backups, antivirus utilities

# UNIX OS

- § Developed in the early 1970s at Bell Laboratories by Ken Thompson and Dennis Ritchie
- § Written in C high-level language, hence, highly portable
- § Multi-user, time-sharing OS
- § Used on a wide variety of computers ranging from notebook computers to super computers
- § Especially prevalent on RISC workstations such as those from Sun Microsystems, Hewlett-Packard, IBM, and Silicon Graphics
- § Structured in three layers – kernel, shell, and utilities

# MS-DOS

- § Stands for Microsoft Disk Operating System.
- § Single-user OS for IBM and IBM-compatible personal computers (PC)
- § Structured in three layers – BIOS (Basic Input Output System), kernel, and shell
- § Very popular in the 1980s, now not in much use and development with the launch of Microsoft Windows OS in 1990s

# Microsoft Windows

- § Developed by Microsoft to overcome limitations of MS-DOS operating system
- § Single-user, multitasking OS
- § Native interface is a GUI
- § Designed to be not just an OS but also a complete operating environment
- § OS of choice for most PCs after 1990

# Microsoft Windows NT

- § Multi-user, time-sharing OS developed by Microsoft
- § Designed to have UNIX-like features so that it can be used for powerful workstations, network, and database servers
- § Supports multiprogramming and is designed to take advantage of multiprocessing on systems having multiple processors
- § Native interface is a GUI
- § Built-in networking and communications features
- § Provides strict system security
- § Rich set of tools for software development

# Linux

- § Open-source OS enhanced and backed by thousands of programmers world-wide
- § Multi-tasking, multiprocessing OS, originally designed to be used in PCs
- § Name “Linux” is derived from its inventor Linus Torvalds
- § Several Linux distributions available (Red Hat, SuSE). Difference in distribution is mostly set of tools, number and quality of applications, documentation, support, and service

# Keywords/Phrases

- § Access control
- § Batch processing
- § Command interpretation
- § Command-line interface (CLI)
- § CPU-bound jobs
- § Cryptography
- § Demand paging
- § External security
- § File
- § File attributes
- § File extensions
- § File management
- § Graphical User Interface (GUI)
- § I/O-bound jobs
- § Internal security
- § Job control language (JCL)
- § Library programs
- § Linux
- § Loosely coupled system
- § Memory management
- § Memory partition
- § Microsoft Windows
- § Microsoft Windows NT
- § MS-DOS
- § Multiprocessing
- § Multiprogramming
- § Multiprogramming with fixed tasks (MFT)
- § Multiprogramming with variable tasks (MVT)
- § Operating systems
- § Multithreading
- § Process
- § Process Control Block (PCB) Multitasking
- § Process management
- § Random access files
- § Response time
- § Security
- § Sequential access files
- § Swapping

(Continued on next slide)

# Keywords/Phrases

(Continued from previous slide)

- § Throughput
- § Tightly coupled system
- § Time-sharing
- § Time slice
- § Time slot
- § Translating programs
- § Turnaround time
- § Unix
- § User authentication
- § Utility programs
- § Virtual machine
- § Virtual memory

## Chapter 15

# Application Software Packages

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

In this chapter you will learn about :

- § Word-processing package
- § Spreadsheet package
- § Graphics package
- § Personal assistance package

# Word-Processing Package

- § **Word-processing** describes use of hardware and software to create, edit, view, format, store, retrieve, and print documents (written material such as letters, reports, books, etc.)
- § **Word-processing package** enables us to do all these on a computer system

# Commonly Supported Features in a Word-Processing Package

- § Entering text
- § Editing text
- § Formatting page style
- § Formatting text
- § Entering mathematical symbols
- § Displaying documents
- § Saving, retrieving and deleting documents
- § Printing documents
- § Importing text, graphics and images
- § Searching and replacing text string
- § Checking spelling
- § Checking grammar and style

# Word-Processing (Few Terminologies)

- § **Style sheet:** Pre-stored page format that can be used while creating a new document or can be applied to an existing document
- § **Font:** Complete set of characters with the same style and size. A word-processing package comes with several standard fonts
- § **Points:** A point is 1/72 of an inch, and the size refers to the distance from the top of the tallest character to the bottom of the character that extends the lowest. Font size is measured in points

(Continued on next slide)

# Word-Processing (Few Terminologies)

- § Three commonly used font styles are *italic*, **bold** and underline.
- § **Justification:** Alignment of text on the left or the right margin, or on both margins. Four types of justification are:
  - § Left-justification
  - § Right-justification
  - § Center-justification
  - § Full-justification

# Different Font Types

This sentence is written in Times New Roman font.

This sentence is written in Helvetica font.

This sentence is written in Palatino font.

This sentence is written in Courier New font.

This sentence is written in Antique Olive font.

# Different Font Sizes

This sentence is written in 10 point Times New Roman font.

This sentence is written in 12 point Times New Roman font.

This sentence is written in 16 point Times New Roman font.

This sentence is written in 24 point Times New Roman font.

This sentence is written in 36 point Times New Roman font.

# Different Font Styles

*This sentence is written in italic style.*

**This sentence is written in bold style.**

This sentence is written in underline style.

You can even make individual words *italic*, **bold**, or underline.

# Different Justification Styles

The term *hardware* refers to the physical devices of a computer system. Thus, the input, storage, processing, control, and output devices are hardware.

## (a) Left Justified text

The term *hardware* refers to the physical devices of a computer system. Thus, the input, storage, processing, control, and output devices are hardware.

## (b) Right Justified text

The term *hardware* refers to the physical devices of a computer system. Thus, the input, storage, processing, control, and output devices are hardware.

## (c) Centered text

# Mathematical Symbols

$$\left\{ t^{(2)} \mid R(t) \wedge \left[ \exists u^{(u)} \right] (S(u) \wedge \neg u[1] = u[2]) \right\}$$

$$\left\{ \langle a, b, c \rangle \mid \exists \langle a, b \rangle (\langle a, b \rangle \in r \wedge \langle a, c \rangle \in s) \right\}$$

# Spreadsheet Package

- § **Spreadsheet package** is a numeric data analysis tool that allows us to create a computerized ledger
- § Useful for any numerical analysis problem whose data can be organized as rows and columns

# Uses of Spreadsheet Package

- § Maintaining and analyzing inventory, payroll, and other accounting records by accountants
- § Preparing budgets and bid comparisons by business analysts
- § Recording grades of students and carrying out various types of analysis of the grades by educators
- § Analyzing experimental results by scientists and researchers
- § Tracking stocks and keeping records of investor accounts by stockbrokers
- § Creating and tracking personal budgets, loan payments, etc. by individuals

# Common Features of Spreadsheet Package

- § Support for a large number of cells
- § Support for addressing a range of cells by the addresses of the endpoint cells
- § Support for different types of cell data (such as label, numeric value, formula, and date & time)
- § Support for use of relative and absolute cell addresses in formula
- § Support for a wide range of commands
- § Support for displaying numeric data in the form of graphs and charts

# Sample Spreadsheet

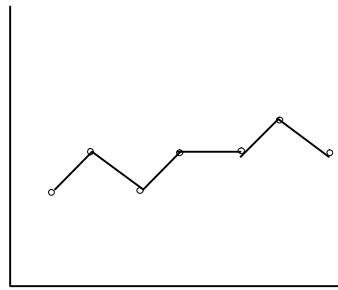
The diagram shows a spreadsheet grid with 11 rows and 7 columns. Row numbers are labeled on the left, and column letters are labeled at the top. A label 'FINAL EXAM MARKS SHEET(CLASS-X: 2001)' spans multiple columns in row 1. Row 3 contains labels for columns A through F. Row 4 contains the formula '@SUM(B9..D9)'. Row 5 contains the numeric value 91.66. Row 6 contains the numeric value 87.33. Row 7 contains the numeric value 81.00. Row 8 contains the numeric value 74.66. Row 9 contains the numeric value 94.00. Row 10 is empty. Row 11 is empty.

	A	B	C	E	E	F
1	FINAL EXAM MARKS SHEET(CLASS-X: 2001)					
2						
3	NAME	PHYS	CHEM	MATHS	TOTAL	PERCE
4						T
5	P. Davis	92	95	88	275	91.66
6	A. Raje	86	82	94	262	87.33
7	D. Rana	75	83	85	243	81.00
8	M. Ray	77	75	72	224	74.66
9	J. Smith	94	92	96	282	94.00
10						
11						

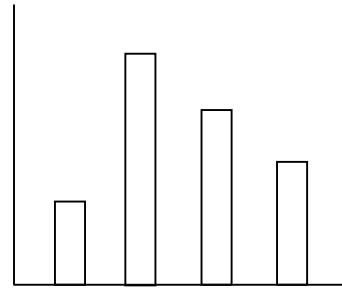
Annotations:

- Row numbers: Points to the first two rows of the grid.
- Column letters: Points to the first two columns of the grid.
- A label running across multiple columns: Points to the header in row 3.
- A label: Points to the label 'PERCE' in row 3.
- Cell F4: Points to the cell containing the formula 'T' in row 4.
- Result of the formula + E9/3: Points to the cell containing the result '94.00' in row 9.
- Result of the function @SUM(B9..D9): Points to the cell containing the result '282' in row 9.
- Cell C11: Points to the empty cell in row 11.
- Numeric Value in a Cell: Points to the cell containing the value '94' in row 9.
- Alphabetic Value in a Cell: Points to the cell containing the value 'J. Smith' in row 9.

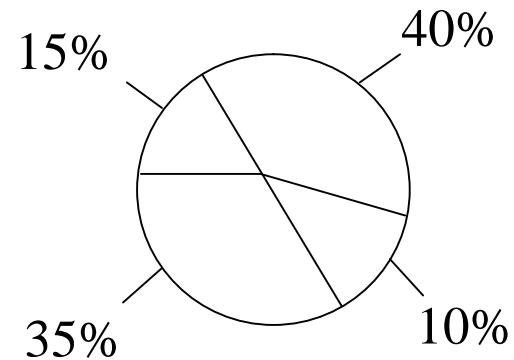
# Examples of a Line Graph, a Bar Chart and a Pie Chart



(a) A line graph



(b) A bar chart



(c) A pie chart

# Graphics Package

**Graphics package** enables us to use a computer system for creating, editing, viewing, storing, retrieving and printing designs, drawings, pictures, graphs and anything else that can be drawn in the traditional manner

# Common Features of Graphics Package

- § Drawing designs
- § Painting drawings and pictures
- § Presenting graphs and charts
- § Dragging-and-dropping graphic objects
- § Importing graphic objects
- § Capturing screen snapshots

# Computer Graphics (Few Terminologies)

- § **Computer-aided-design (CAD):** Integration of computers and graphics design packages for the purpose of automating the design and drafting process
- § **Vector graphics:** Graphic object composed of patterns of lines, points, circles, arcs and other geometric shapes that can be easily represented by few geometric parameters
- § **Raster graphics:** Graphic object composed of patterns of dots called pixels

# Personal-assistance Package

**Personal-assistance package allows individuals to:**

- § Use personal computers for storing and retrieving their personal information
- § Planning and managing their schedules, contacts, finances and inventory of important items

# Common Features of Personal Assistance Package

- § Calendar
- § To-do list
- § Address book
- § Investments book
- § Inventory bookf

# Key Words/Phrases

- § Bit-mapped image
- § Bold
- § Cell
- § Center justification
- § Clip art library
- § Computer Aided Design (CAD)
- § Font
- § Full justification
- § Graphics package
- § Italic
- § Justification
- § Landscape mode
- § Left justification
- § Personal assistance package
- § Portrait mode
- § Raster graphics
- § Right justification
- § Spreadsheet package
- § Style sheet
- § Underline
- § Vector graphics
- § What You See Is What you Get (WYSIWYG)
- § Word-processing
- § Word-processing package

# Chapter 16

# Business Data Processing

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

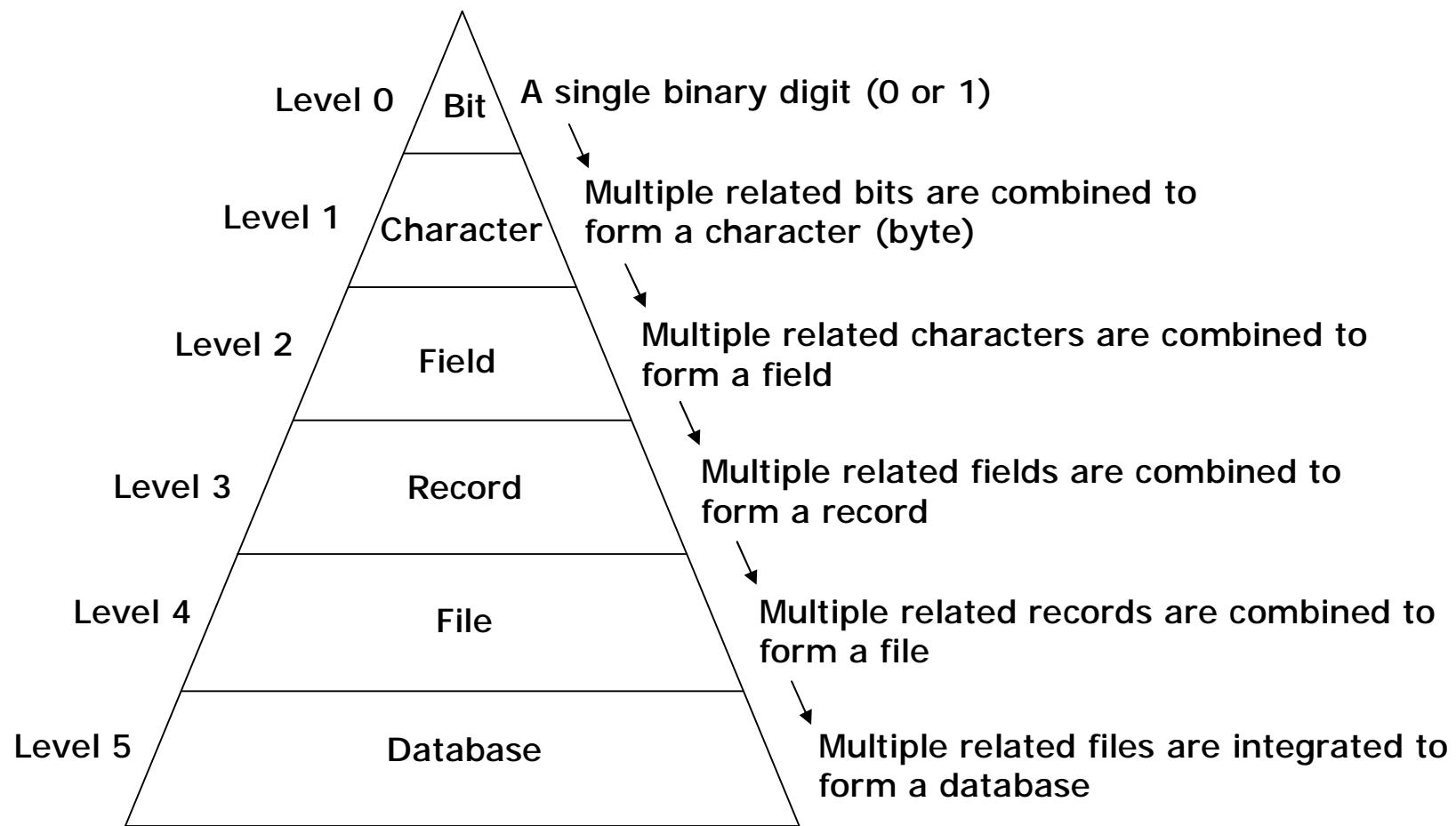
**In this chapter you will learn about:**

- § Difference between data and information
- § Data processing converts raw data into useful information
- § Data storage hierarchy commonly used to facilitate data processing
- § Standard methods of organizing data
- § Basic concepts of database systems

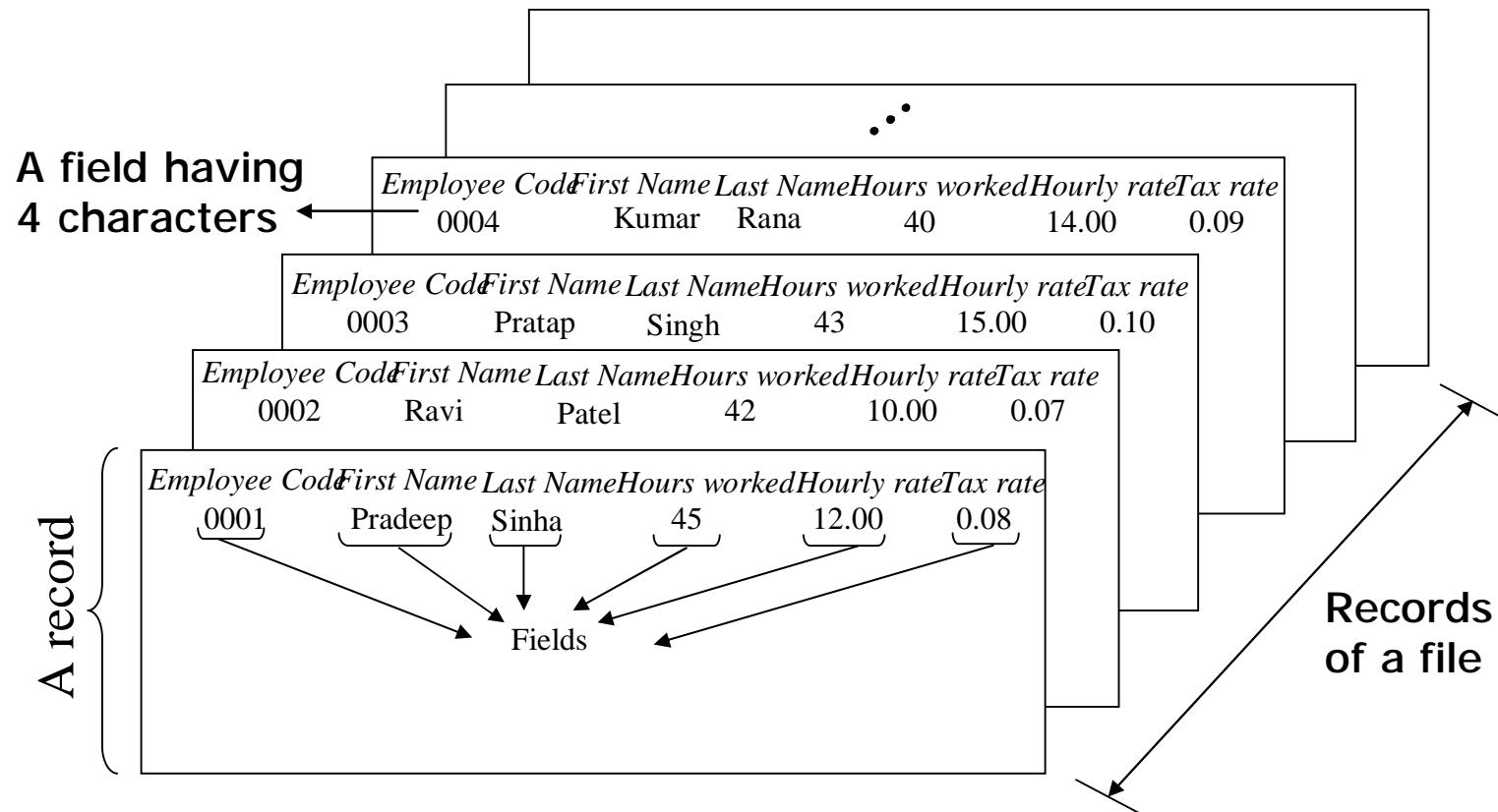
# Data Processing

- § Data is a collection of facts – unorganized but able to be organized into useful information
- § Information is data arranged in an order and form that is useful to the people who receive it
- § Data processing is a series of actions or operations that converts data into useful information
- § A data processing system includes resources such as people, procedures, and devices used to process input data for producing desirable output

# Data Storage Hierarchy



# Relationship Among Character, Field, Record, and File



# Standard Methods of Organizing Data

- § **File-oriented approach:** Application's data is organized into one or more files and application program processes them to generate the desired output
- § **Database-oriented approach:** Data from multiple related files are integrated together to form a database:
  - § Provides greater query flexibility
  - § Reduces data redundancy
  - § Solves data integrity (inconsistency) problem
  - § Makes data independent of the application programs
  - § Includes data security features at database level, record level, and field level

(Continued on next slide)

# File Management System

- § In *file-oriented approach* of organizing data, an application's data is organized into one or more files
- § Application program processes the data stored in these files to generate the desired output
- § Set of programs is provided to facilitate the users in organizing, creating, deleting, updating, and manipulating their files
- § All these programs together form a File Management System (FMS)

# File Types

A file management system supports following file types:

- § **Transaction file:** Stores input data until it can be processed
- § **Master file:** Contains all current data relevant to an application
- § **Output file:** Stores output produced by one program that is used as input to another program
- § **Report file:** Holds a copy of a report generated by an application
- § **Backup file:** Copy of a file, created as a safety precaution against loss of data

# File Organizations

- § File organization is the physical organization of the records of a file for convenience of storage and retrieval of data records
- § Three commonly used file organizations are:
  - § **Sequential:** Records are stored one after another in ascending or descending order determined by the value of the key field of the records
  - § **Direct/random:** Desired record pertaining to current transaction can be directly located by its key field value without having to navigate through sequence of other records

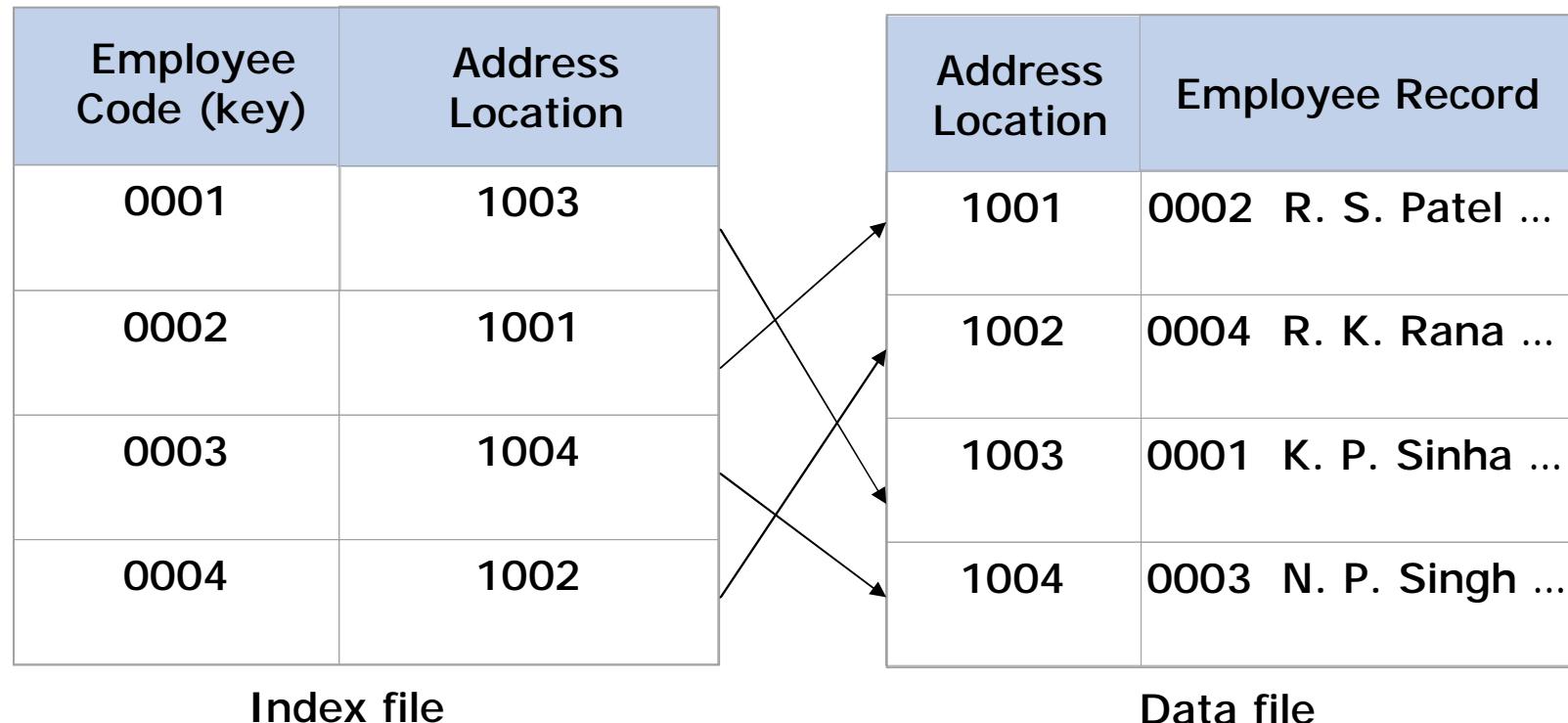
(Continued on next slide)

# File Organizations

(Continued from previous slide)

- § **Indexed sequential:** There are two files for every data file – the data file which contains the records stored in the file, and the smaller index file which contains the key and disk address of each record stored in the data file

# Organization of An Indexed Sequential File



# File Utilities

- § Routines to perform a variety of generalized operations on data files
- § Operations performed by some commonly used file utilities are Sorting, Searching, Merging, Copying, Printing, and Maintenance

# Sorting On One Key

Employee Code	Department Code	Other fields (Name, Address, Qualification, Basic Salary, etc.)
101	2	---
123	3	---
124	1	---
176	2	---
178	1	---
202	3	---
213	1	---

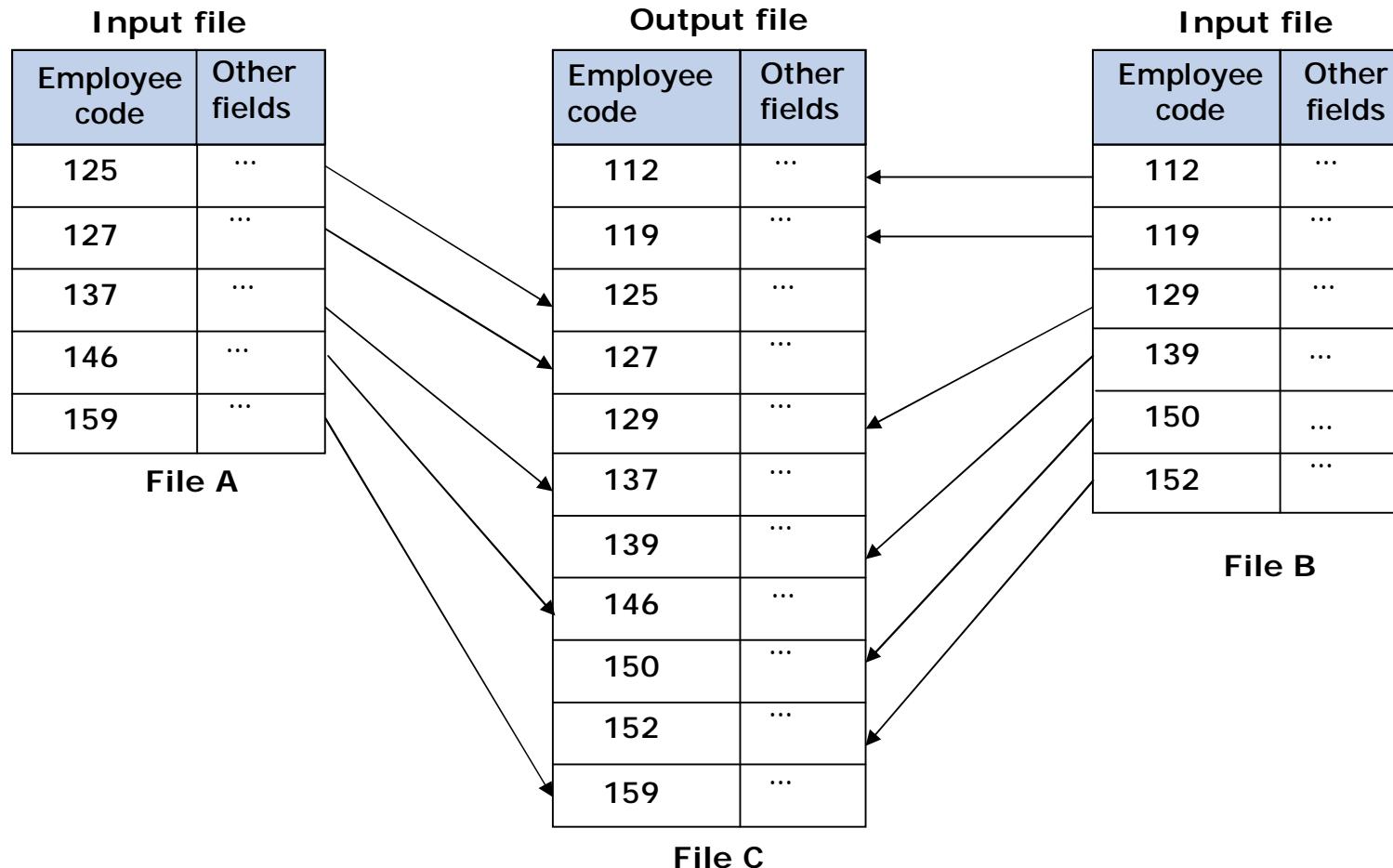
Sorting on ascending employee code sequence

# Sorting On Two Key

Employee Code	Department Code	Other fields (Name, Address, Qualification, Basic Salary, etc.)
124	1	---
178	1	---
213	1	---
101	2	---
176	2	---
123	3	---
202	3	---

Sorting on a ascending employee code (secondary key) within ascending department code (primary key)

# Merging of Two Files



Merging of files A and B to produce file C

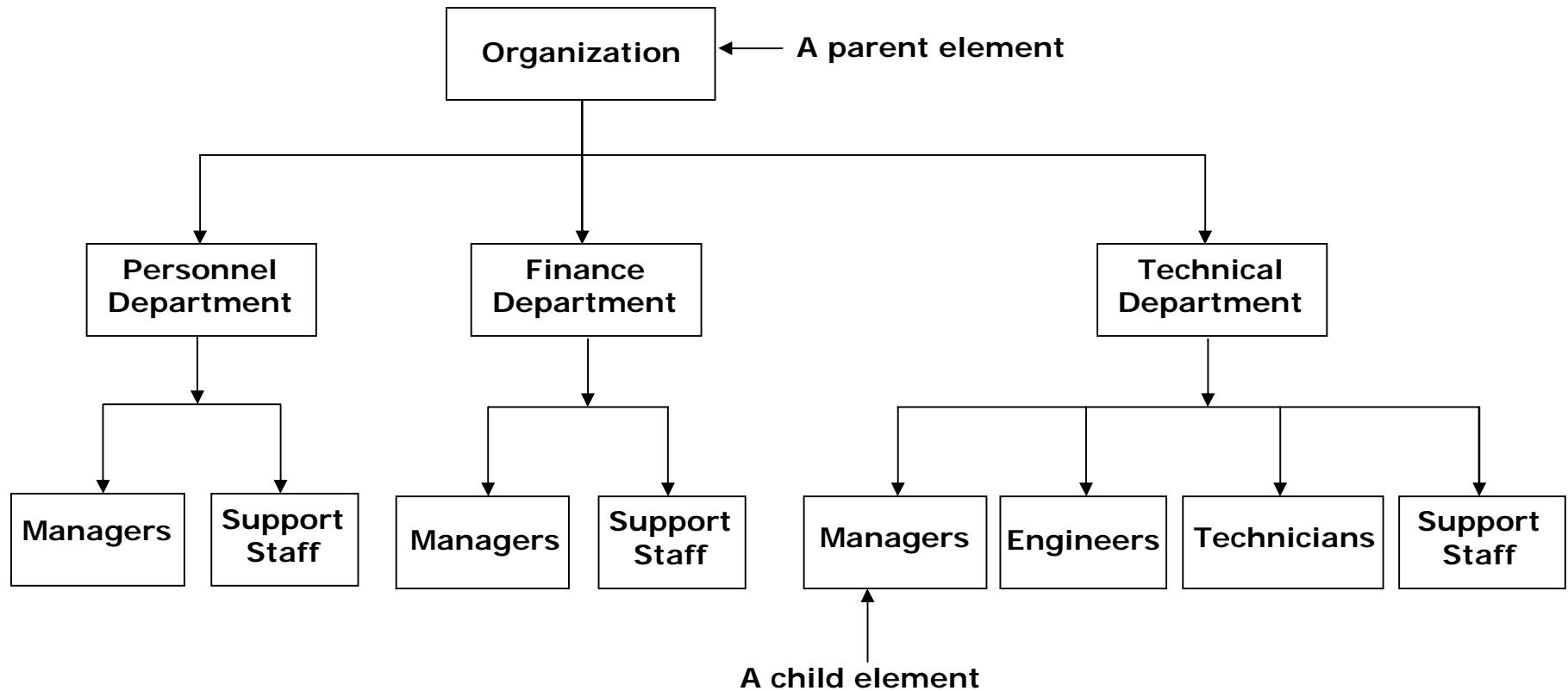
# Database Management System

- § In *database-oriented approach* of organizing data, a set of programs is provided to facilitate users in organizing, creating, deleting, updating, and manipulating data in a database
- § All these programs together form a *Database Management System (DBMS)*

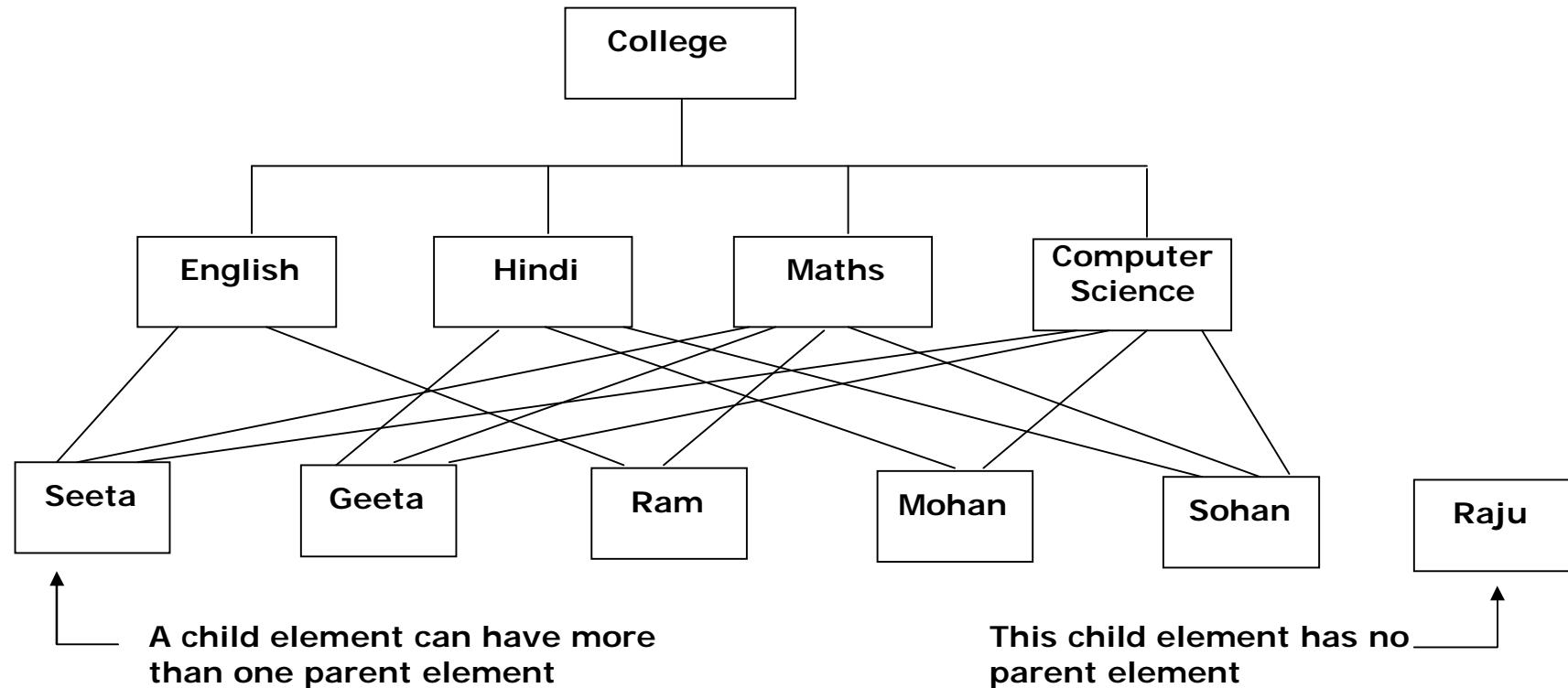
# Database Models

- § *Database model* defines the manner in which the various files of a database are linked together.
- § Four commonly used database models are:
  - § Hierarchical
  - § Network
  - § Relational
  - § Object-oriented

# Hierarchical Database



# Network Database



# Relational Database

Membership No.	Member's name	Member's Address
83569	K. N. Raina	C-15, Sarita Vihar, Pune-7
62853	D. P. Singh	A-22, Anand Park, Pune-5
12859	R. Pandey	D-18, Vrindavan, Pune-7
32228	R. S. Gupta	A-12, Nandanvan, Pune-2
23466	S. K. Ray	B-05, Royal Villa, Pune-3
11348	P. K. Sen	B-16, Anand Park, Pune-5
16185	T. N. Murli	A-11, Vrindavan, Pune-7

(a) Members data table.

Borrower (Membership No.)	Book No. (ISBN)	Due Date (DD-MM-YYYY)
12859	27-21675-2	10-12-2007
11348	89303-530-0	08-11-2007
32228	13-201702-5	10-11-2007
16185	22-68111-7	05-12-2007
12859	71606-214-0	06-11-2007
62853	13-48049-8	15-11-2007
11348	18-23614-1	12-11-2007

(b) Borrowed books data table

Book No. (ISBN)	Book Title	Author
13-201702-5	Concepts of Physics	H. C. Verma
13-48049-8	Concepts of Chemistry	S. S. Dubey
18-23614-1	Astrology for You	N. K. Sharma
22-68111-7	Fundamentals of Computers	K. Ramesh
27-21675-2	C++ Programming	R. P. Rajan
71606-214-0	Computer Networks	A. N. Rai
89303-530-0	Database Systems	P. N. Dixit

(c) Books data table

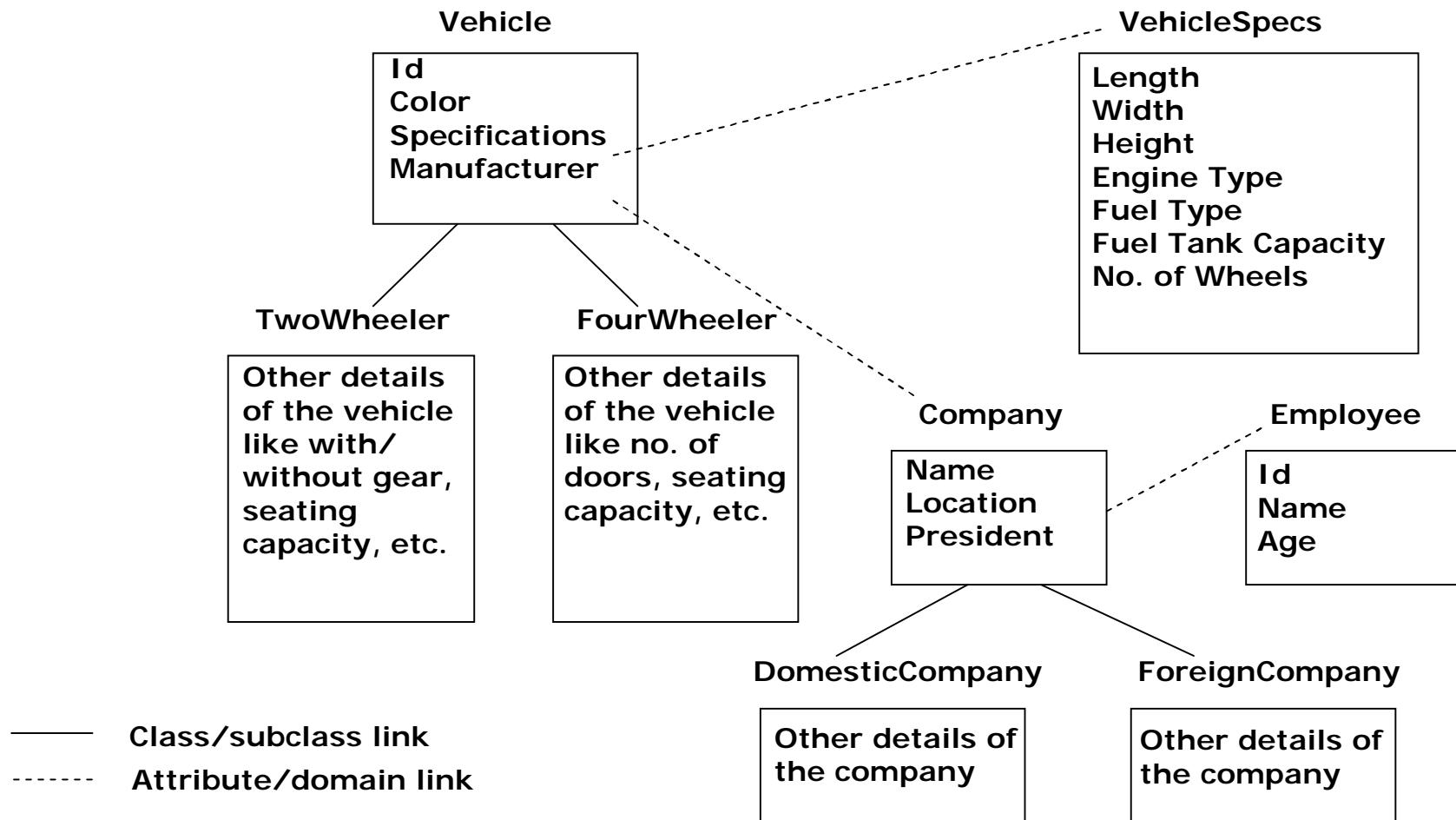
# Sample Report

## List of overdue books as on 10-11-2007

Membership No.	Member's Name	Member's Address	Due Date	Book No.	Book Title	Book Author
11348	P. K. Sen	B-16, Anand Park, Pune-5	08-11	89303-530-0	Database Systems	P. N. Dixit
32228	R. S. Gupta	A-12, Nandanvan, Pune-2	10-11	13-201702-5	Concepts of Physics	H. C. Verma
12859	R. Pandey	D-18, Vrindavan, Pune-7	06-11	71606-214-0	Computer Networks	A. N. Rai

A report of overdue books as of 10-11-2007 from the sample database of previous slide

# Object-Oriented Database



# Main Components of a DBMS

- § DBMS allows users to organize, process and retrieve selected data from a database without knowing about the underlying database structure
- § Four major components of a DBMS that enable this are:
  - § *Data Definition Language (DDL)*: Used to define the structure (schema) of a database
  - § *Data Manipulation Language (DML)*: Provides commands to enable the users to enter and manipulate the data

(Continued on next slide)

# Main Components of a DBMS

(Continued from previous slide)

- § *Query Language*: Enables users to define their requirements for extracting the desired information from the database in the form of queries
- § *Report generator*: Enables the users of a database to design the layout of a report so that it can be presented in the desired format

# Creating a Database

Creation of a database is a three step process:

- § Defining its structure (schema)
- § Designing forms (custom screens) for displaying and entering data
- § Entering the data into it

# Sample Database Form

**EMPLOYEE DATABASE DATA ENTRY FORM**

EMPLOYEE ID:	856392	SEX:	M	AGE:	42
EMPLOYEE NAME:	LAST NAME: SINHA FIRST NAME: PRADEEP MIDDLE NAME: KUMAR				
CONTACT ADDRESS:	ADDRESS 1: F/8, ANAND PARK ADDRESS 2: SOCIETY, AUNDH CITY: PUNE STATE: MH POSTAL CODE: 411007				
TELEPHONE NO.:	(020) 5680-489				
ANY OTHER INFORMATION:	IS FLUENT IN JAPANESE LANGUAGE				

# Viewing, Modifying, Deleting, and Adding Records

- § All database systems provide commands to view, modify, delete, or add records of an already established database
- § Many database systems also provide a facility to set up a filter allowing user to browse through and view only those records that meet some criterion

# Searching a Database

Commonly supported features for enabling a user to search for desired information in a database are:

- § *Find command*: Used for simple database queries
- § *Query language*: Used for more complex database queries
- § *Query By Example (QBE)*: Provides a simple user interface for specifying search criteria

# Creating Reports

- § Reports are generated by using report generator of a database system to assemble the output of a database query in desired format
- § Report generator enables user to specify layout of the report, titles & subtitles for the report, column headings for various fields, and other elements to make the report appear more presentable

# Sample Output of Report

## LIST OF EMPLOYEES WHO BELONG TO PUNE

DATE: DECEMBER 15, 2007

LAST NAME	FIRST NAME	ADDRESS-1	ADDRESS-2	TELEPHONE NUMBER
Gupta	Rajiv	A-12, Nandanvan	M. G. Road	4623-4892
Murli	Tapan	A-11, Vrindavan	Pashan Road	5863-4905
Pandey	Rupa	D-18, Vrindana	Pashan Road	5865-3236
Raina	Pushpa	C-15, Sarita Vihar	Aundh Road	5755-8328
Ray	Suhas	B-05, Royal Villa	M. G. Road	4685-6356
Sen	Prakash	B-16, Anand Park	Aundh Road	5762-3333
Singh	Deepak	A-22, Anand Park	Aundh Road	5728-6287

The report is sorted to present the list in alphabetical order of their last name

# Key Words/Phrases

- § Activity ratio
- § Backup file
- § Collision
- § Copying
- § Data
- § Data Definition Language (DDL)
- § Data dependence
- § Data dictionary
- § Data file
- § Data integrity
- § Data Manipulation Language (DML)
- § Data processing
- § Data redundancy
- § Data storage hierarchy
- § Database
- § Database administrator
- § Database Management System (DBMS)
- § Database model
- § Direct file
- § Field
- § File
- § File Management System (FMS)
- § File utilities
- § Filter
- § Hashing
- § Hashing algorithm
- § Hierarchical database
- § Index file
- § Indexed sequential file
- § Information
- § Master file
- § Merging
- § Network database
- § Output file
- § Peripheral Interchange Program
- § Primary key

(Continued on next slide)

# Key Words/Phrases

(Continued from previous slide)

- § Query By Example
- § Query language
- § Record
- § Relational database
- § Report file
- § Report Generator
- § Schema
- § Searching
- § Secondary key
- § Sequential file
- § Sorting
- § Transaction file
- § Tuple

# Chapter 17

# Data Communications and Computer Networks

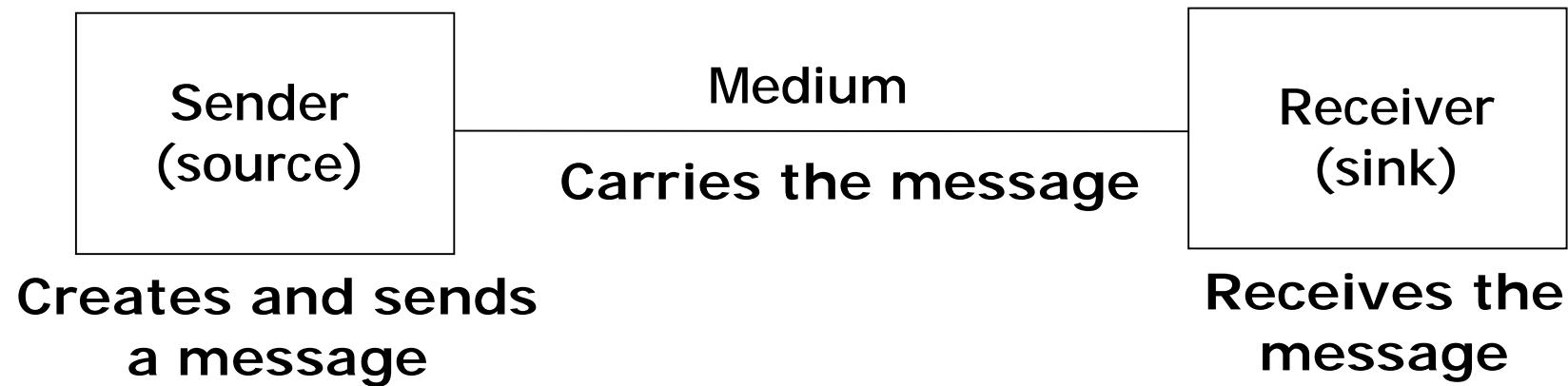
Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

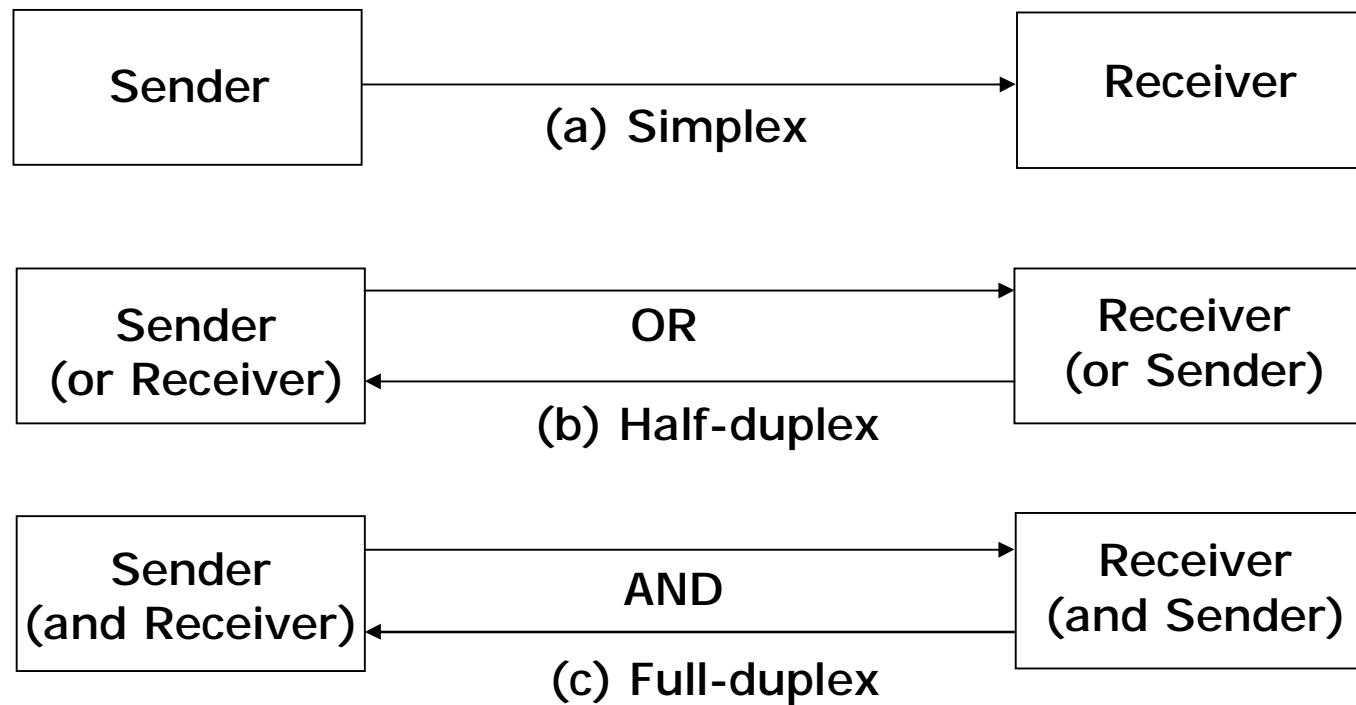
**In this chapter you will learn about:**

- § Basic elements of a communication system
- § Techniques, channels, and devices used to transmit data between distant locations
- § Types of computer networks
- § Communication protocols and their use in computer networks
- § Internetworking tools and their use in building large computer networks
- § Characteristics and advantages of distributed data processing

# Basic Elements of a Communication System



# Data Transmission Modes



# Data Transmission Speed

- § **Bandwidth:** Range of frequencies available for data transmission. It refers to data transmission rate. Higher the bandwidth, the more data it can transmit
- § **Baud:** Unit of measurement of data transfer rate. Measured in bits per second (bps)

# Data Transmission Speed Category

- § **Narrowband:** Sub-voice grade channels in range from 45 to 300 baud. Mainly used for telegraph lines and low-speed terminals
- § **Voiceband:** Voice grade channels with speed up to 9600 baud. Mainly used for ordinary telephone voice communication and slow I/O devices
- § **Broadband:** High speed channels with speed up to 1 million baud or more. Mainly used for high-speed computer-to-computer communication or for simultaneous transmission of data

# Data Transmission Media

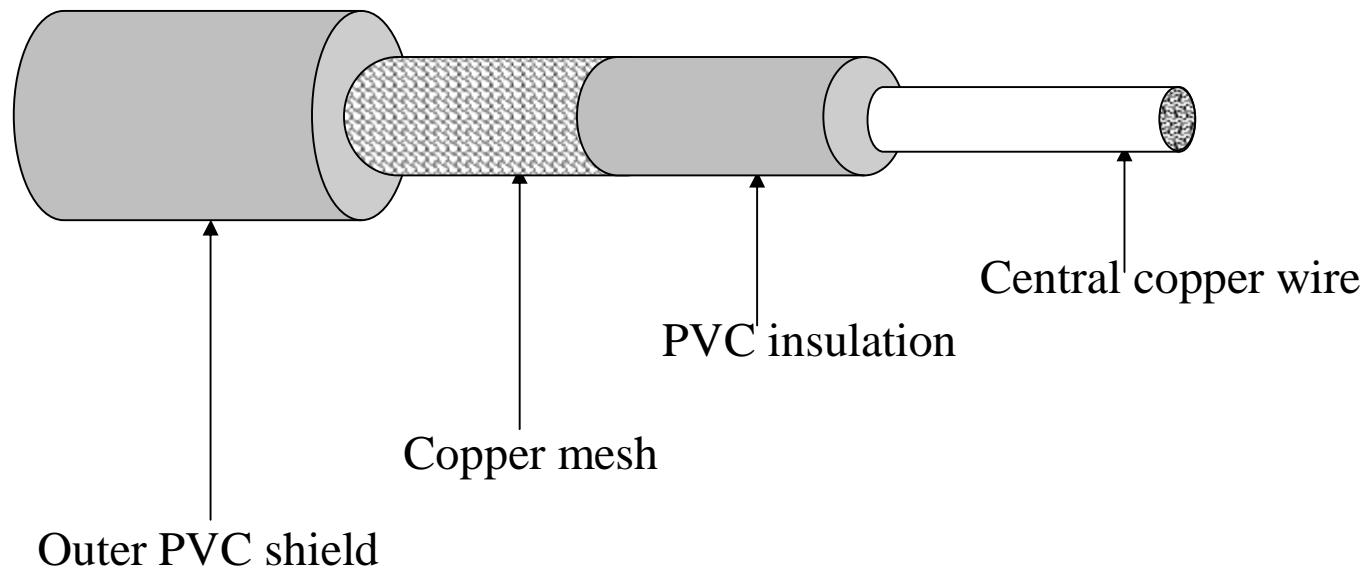
The most commonly used ones are:

- § Twisted-pair wire (UTP cable)
- § Coaxial cable
- § Microwave system
- § Communications satellite
- § Optical fibers

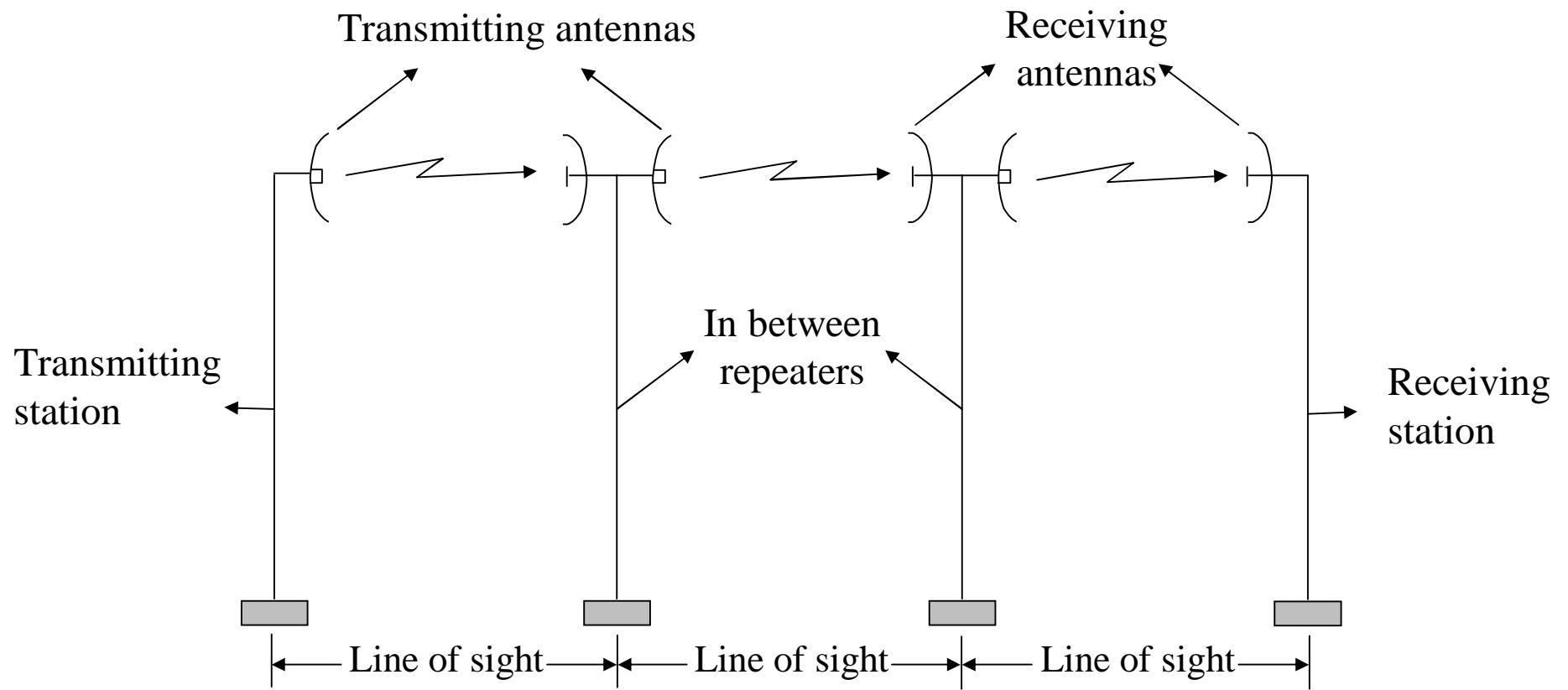
# Unshielded Twisted-Pair (UTP) Cable



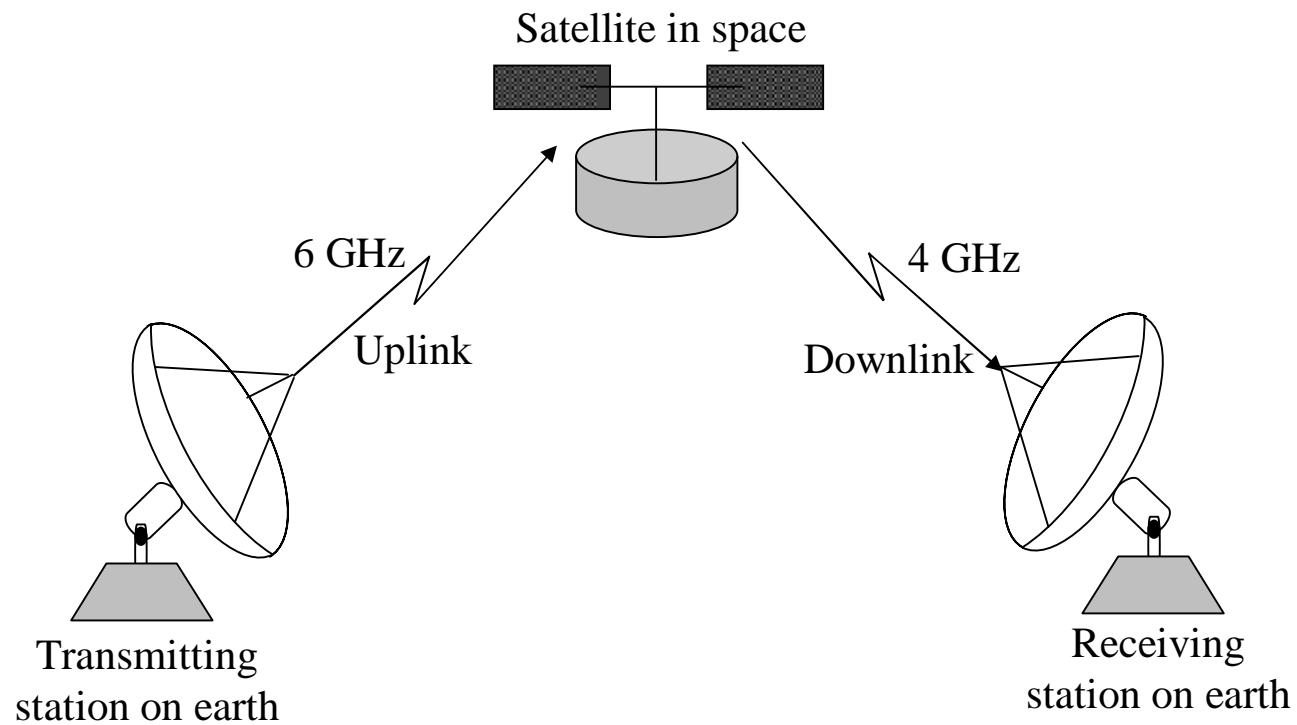
# Coaxial Cable



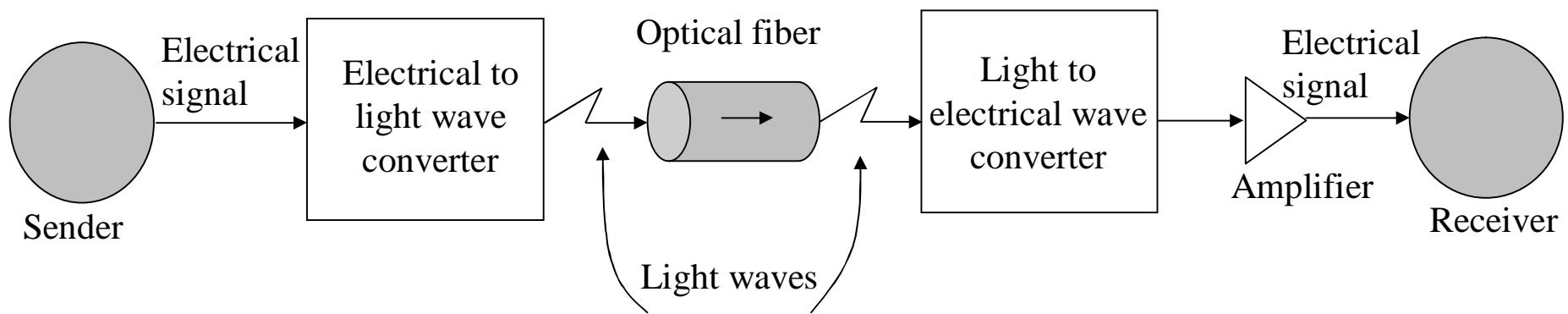
# Microwave Communication System



# Satellite Communication System



# Optical Fiber Communication System



# Digital and Analog Data Transmission

- § **Analog signal:** Transmitted power varies over a continuous range. Example: sound, light, and radio waves
- § **Digital signal:** Sequence of voltage pulses represented in binary form
- § Computer generated data signal is digital, whereas telephone lines carry analog signals

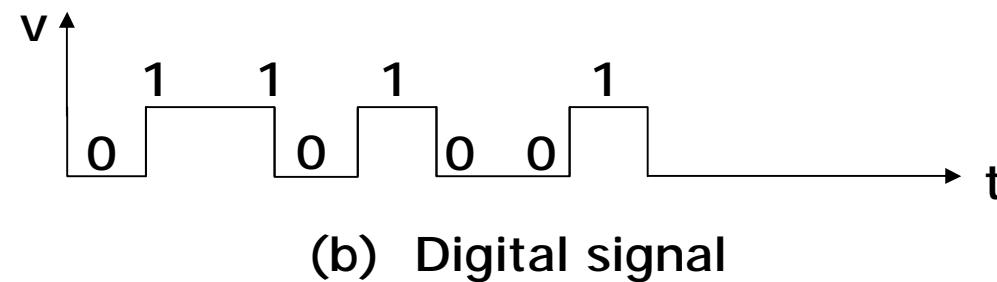
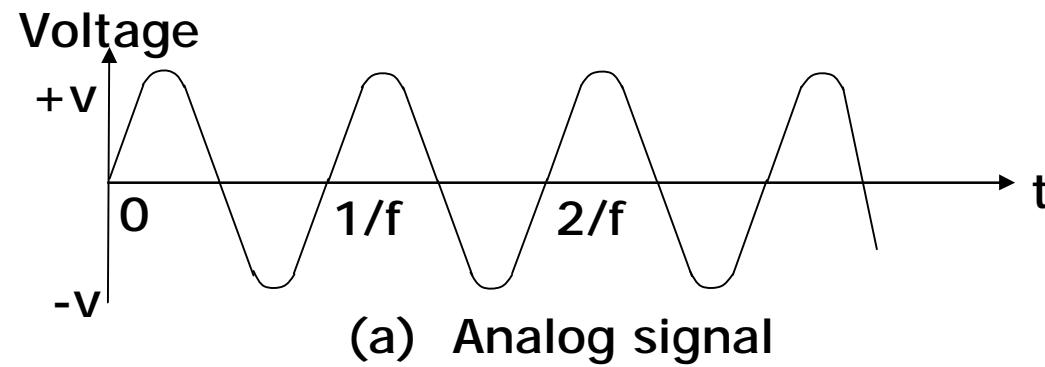
(Continued on next slide)

# Digital and Analog Data Transmission

(Continued from previous slide)

- § When digital data is to be sent over an analog facility, digital signals must be converted to analog form
- § Conversion of digital signal to analog form is known as modulation
- § Conversion of analog signal to digital form is known as demodulation
- § Digital transmission of data is preferred over analog transmission of data due to lower cost, higher transmission speeds, and lower error rate

# Analog and Digital Signals



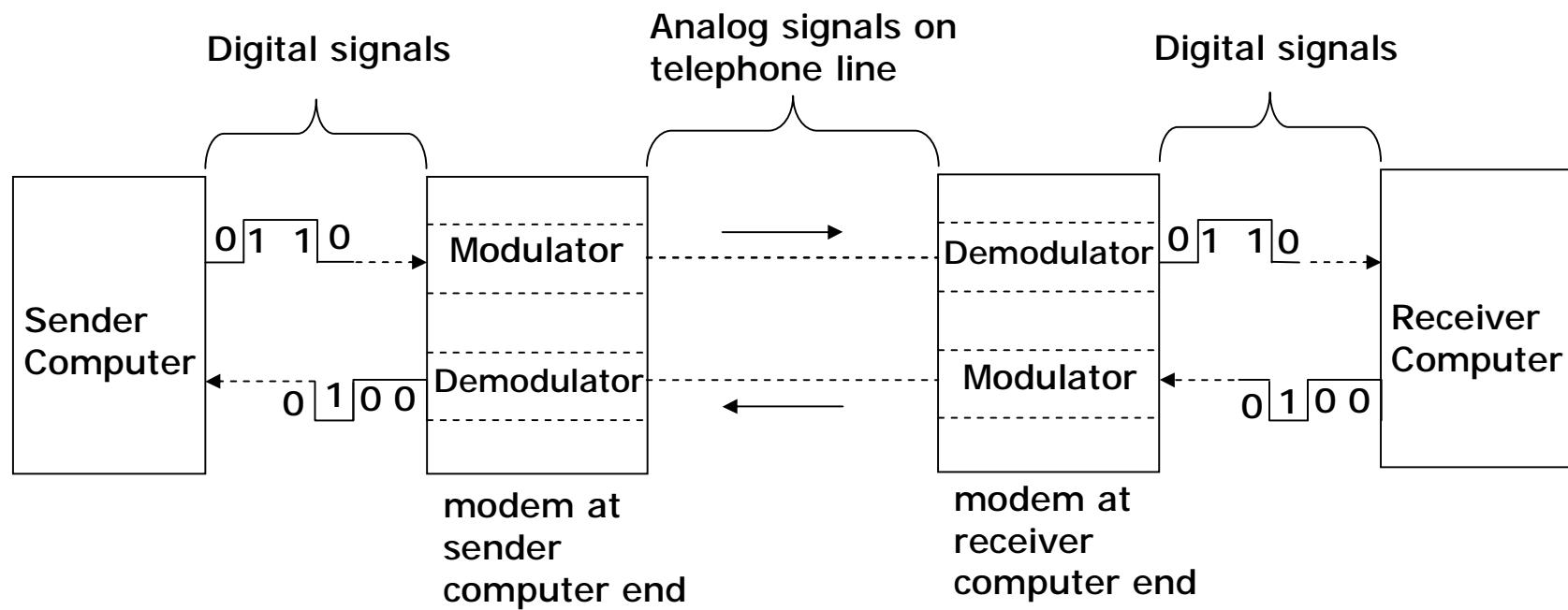
# Modulation Techniques

- § **Amplitude Modulation (AM):** Two binary values (0 and 1) of digital data are represented by two different amplitudes of the carrier signal, keeping frequency and phase constant
- § **Frequency Modulation (FM):** Two binary values of digital data are represented by two different frequencies, while amplitude and phase are kept constant
- § **Phase Modulation (PM):** Two binary values of digital data are represented by shift in phase of carrier signal

# Modems

- § Modem is short for MOdulator/DEModulator
- § Special device used for conversion of digital data to analog form (modulation) and vice-versa (demodulation)
- § Essential piece of hardware where two digital devices (say two computers) want to communicate over an analog transmission channel (say a telephone line)

# Use of Modems in Data Communications



# Factors for Modem Selection

- § Transmission speed
- § Internal versus external
- § Facsimile facility

# Data Transmission Services

- § Data transmission service providers are popularly known as *common carriers*
- § Various types of services offered by common carriers are:
  - § **Dial-up line:** Operates in a manner similar to a telephone line
  - § **Leased line:** Special conditioned telephone line that directly and permanently connects two computers
  - § **Integrated Services Digital Network (ISDN):** Telephone system that provides digital (not analog) telephone and data services

(Continued on next slide)

# Data Transmission Services

(Continued from previous slide)

- § **Value Added Network (VAN):** Provides value-added data transmission service. Value added over and above the standard services of common carriers may include e-mail, data encryption/decryption, access to commercial databases, and code conversion for communication between computers

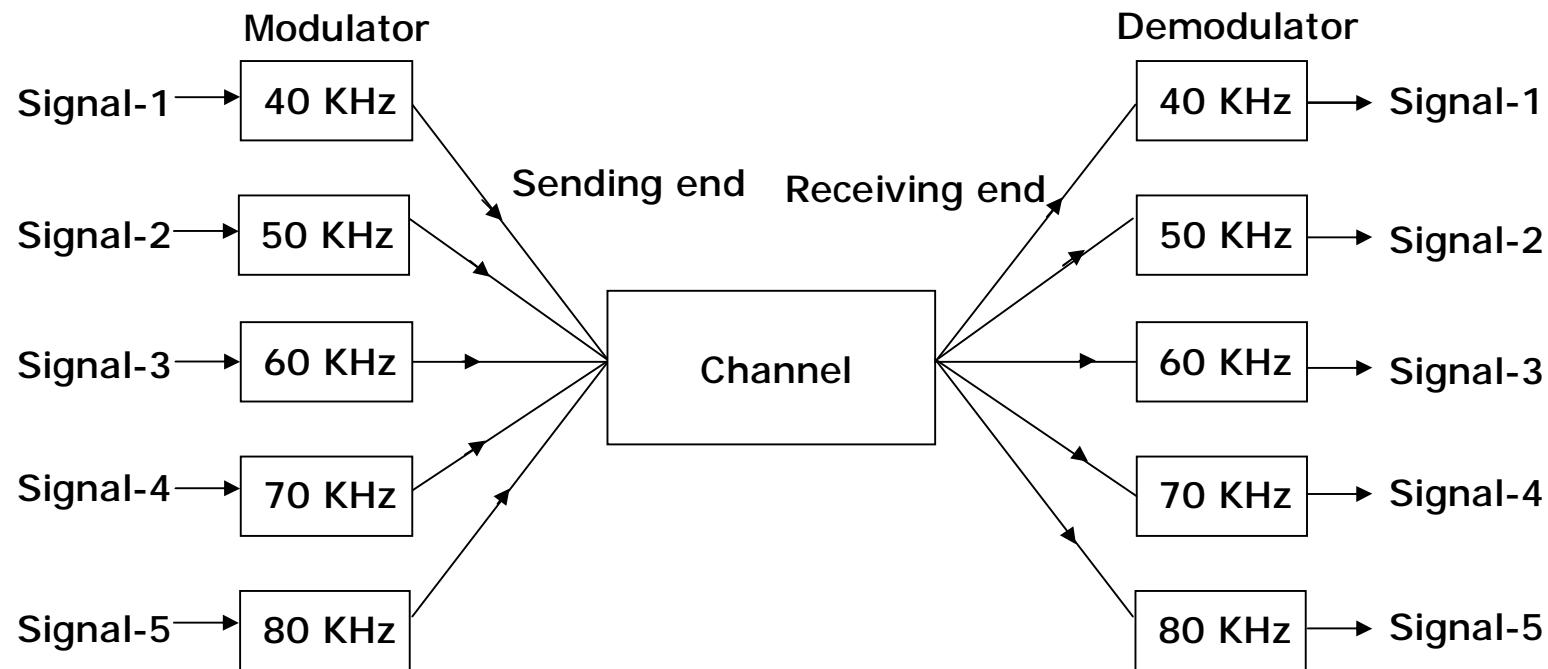
# Multiplexing

- § Method of dividing physical channel into many logical channels so that a number of independent signals may be simultaneously transmitted
- § Electronic device that performs multiplexing is known as a *multiplexer*
- § Multiplexing enables a single transmission medium to concurrently transmit data between several transmitters and receivers

## Two Basic Methods of Multiplexing

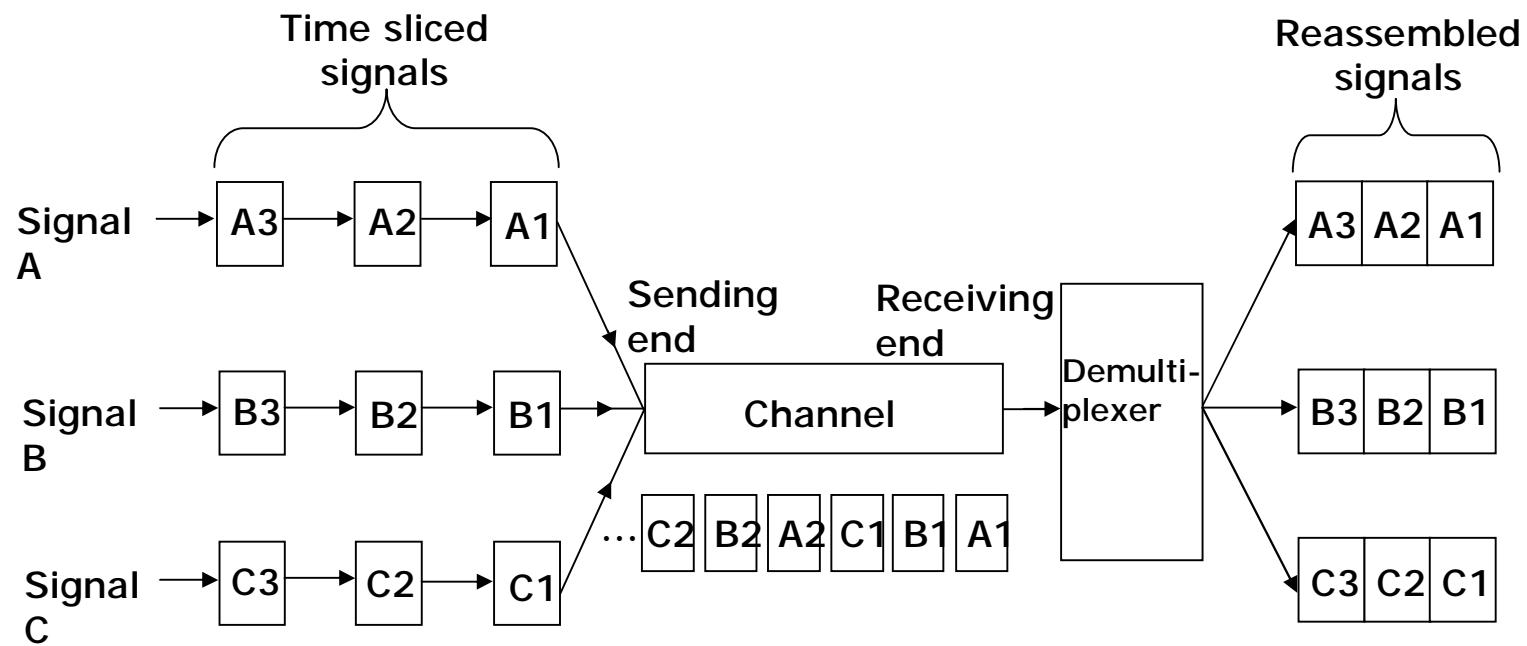
- § **Frequency-Division Multiplexing (FDM):** Available bandwidth of a physical medium is divided into several smaller, disjoint logical bandwidths. Each component bandwidth is used as a separate communication line
- § **Time-Division Multiplexing (TDM):** Total time available in a channel is divided among several users, and each user of the channel is allotted a time slice during which he/she may transmit a message

# Frequency-Division Multiplexing



Frequency-Division Multiplexing

# Time-Division Multiplexing



# Asynchronous and Synchronous Transmission

- § Two modes of data transmission on a communication line are asynchronous and synchronous
- § Asynchronous transmission
  - § Sender can send data at any convenient time and the receiver will accept it
  - § Data is transmitted character by character at irregular intervals
  - § Well suited to many keyboard type terminals

(Continued on next slide)

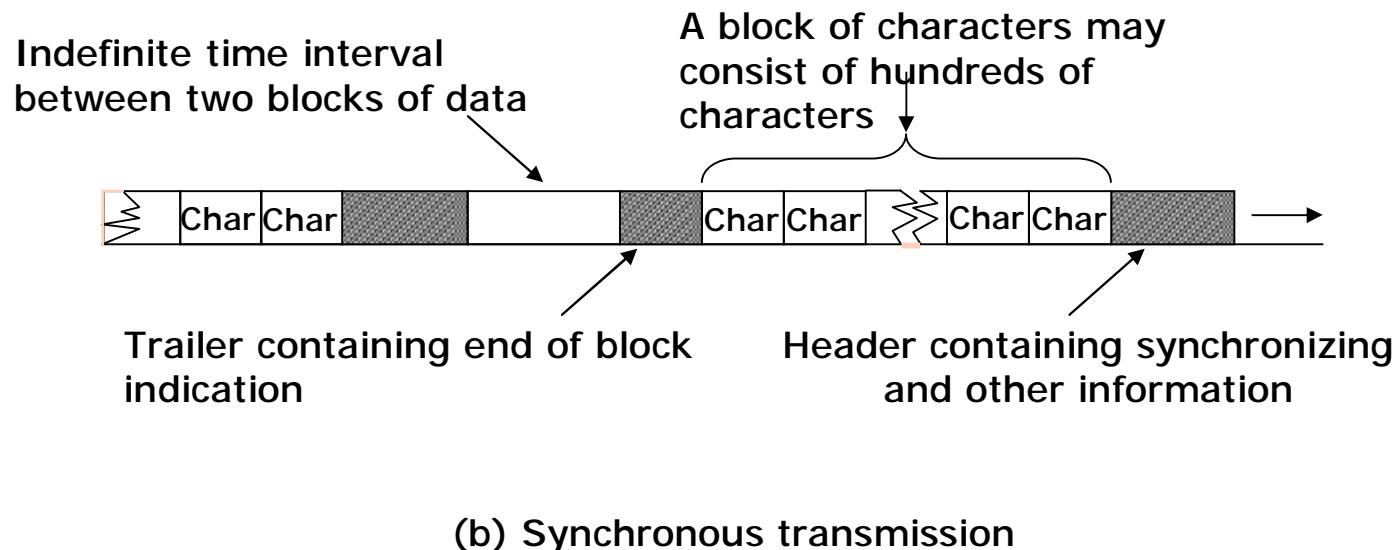
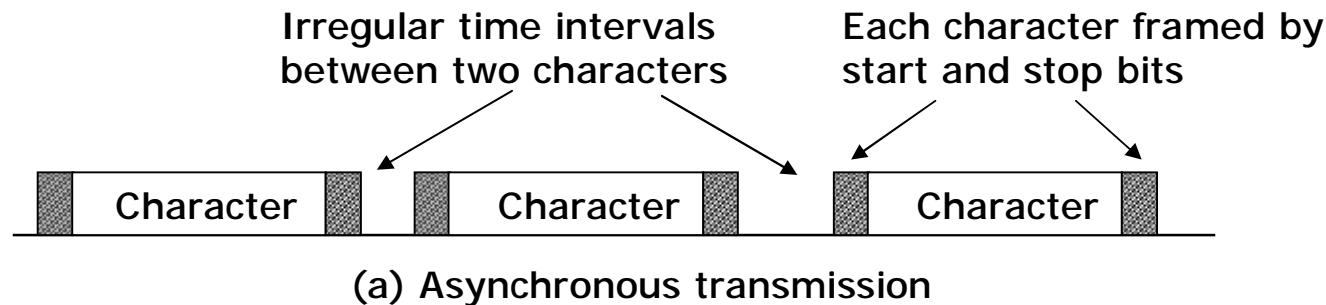
# Asynchronous and Synchronous Transmission

(Continued from previous slide)

## § Synchronous transmission

- § Sender and receiver must synchronize with each other to get ready for data transmission before it takes place
- § Entire blocks of characters are framed and transmitted together
- § Well suited to remote communication between a computer and such devices as buffered terminals and printers

# Data Transmission



# Switching Techniques

- § Data is often transmitted from source to destination through a network of intermediate nodes
- § Switching techniques deal with the methods of establishing communication links between the sender and receiver in a communication network
- § Three commonly used switching techniques are:
  - § **Circuit switching:** Dedicated physical path is established between sending and receiving stations through nodes of the network for the duration of communication

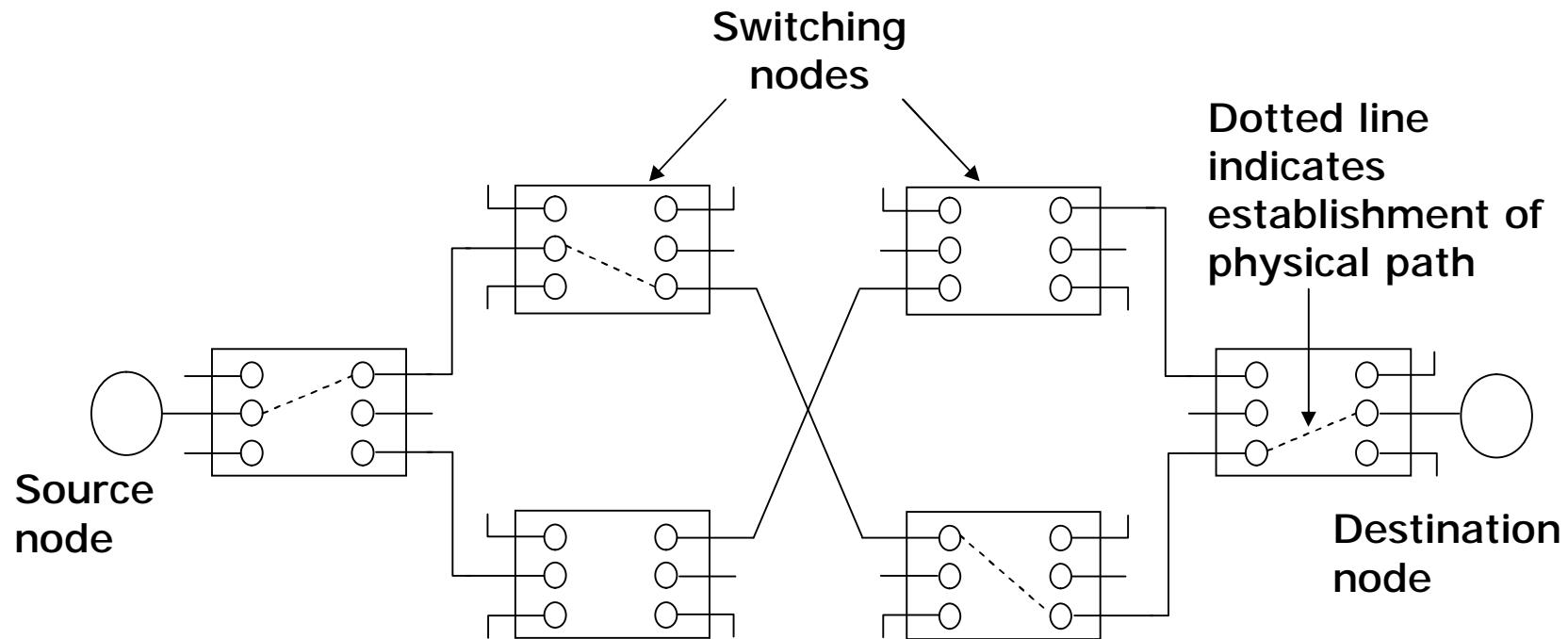
(Continued on next slide)

# Switching Techniques

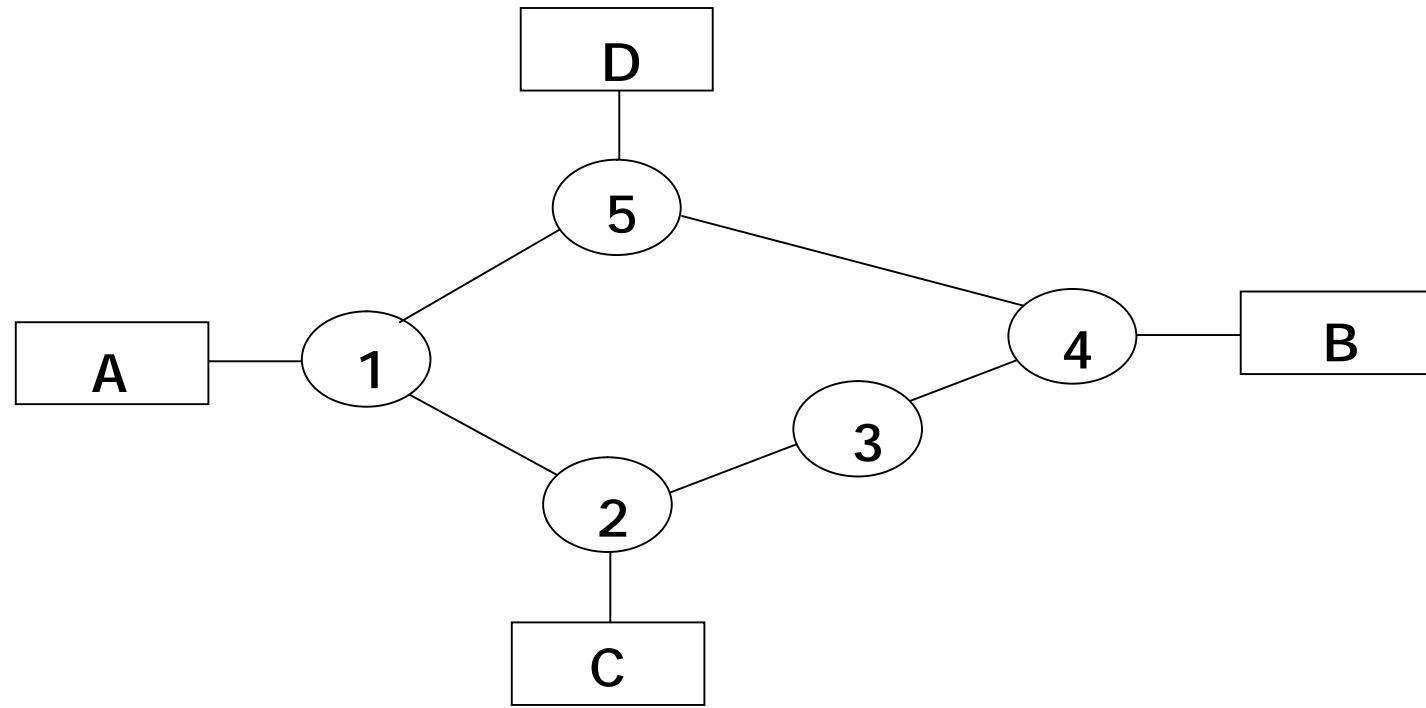
(Continued from previous slide)

- § **Message switching:** Sender appends receiver's destination address to the message and it is transmitted from source to destination either by store-and-forward method or broadcast method
- § **Packet switching:** Message is split up into fixed size packets and each packet is transmitted independently from source to destination node. Either store-and-forward or broadcast method is used for transmitting the packets. All the packets of a message are re-assembled into original message at the destination node

# Circuit Switching Method

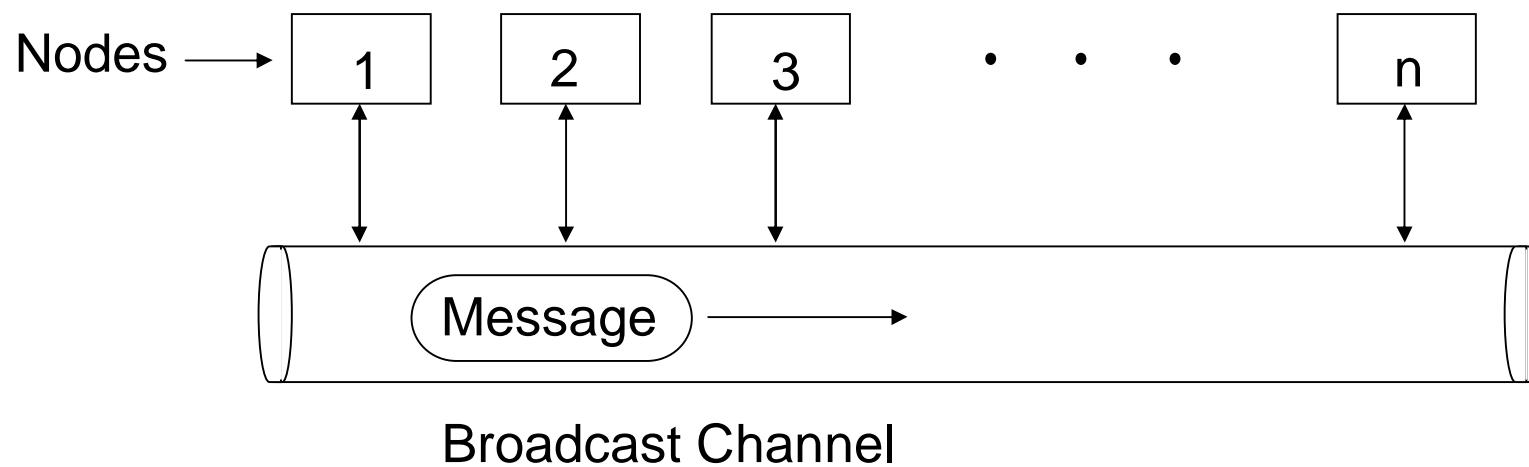


# Store-and-Forward Method of Message Switching



Either path 1-2-3-4 or 1-5-4 may be used to transmit a message from A to B.

# Broadcast Method of Message Switching



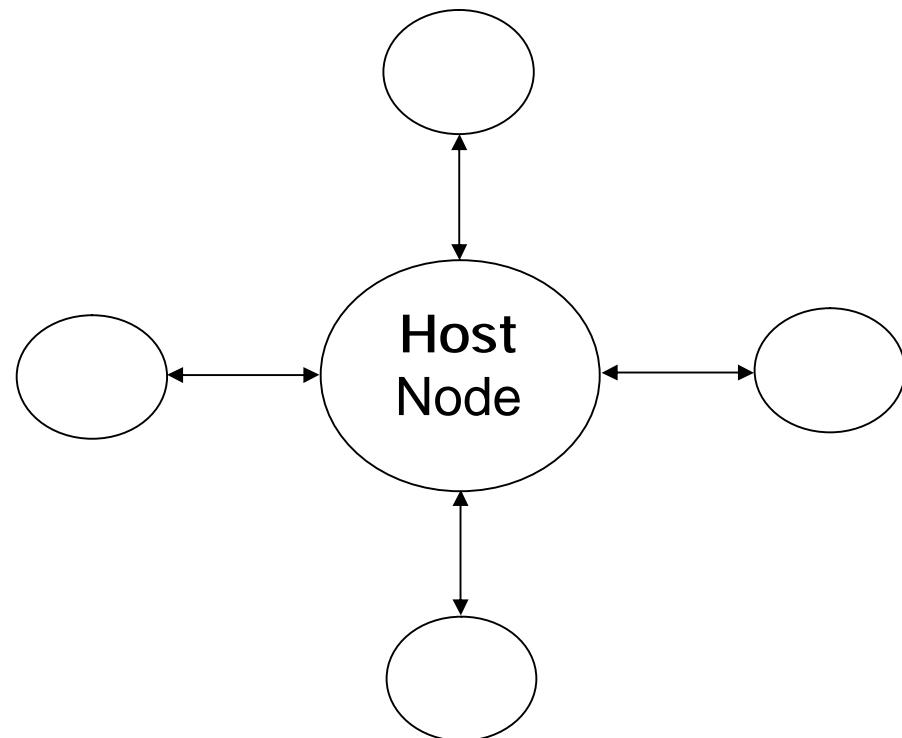
# Routing Techniques

- § In a WAN, when multiple paths exist between the source and destination nodes of a packet, any one of the paths may be used to transfer the packet
- § Selection of path to be used for transmitting a packet is determined by the routing technique used
- § Two popularly used routing algorithms are:
  - § **Source routing:** Source node selects the entire path before sending the packet
  - § **Hop-by-hop routing:** Each node along the path decides only the next node for the path

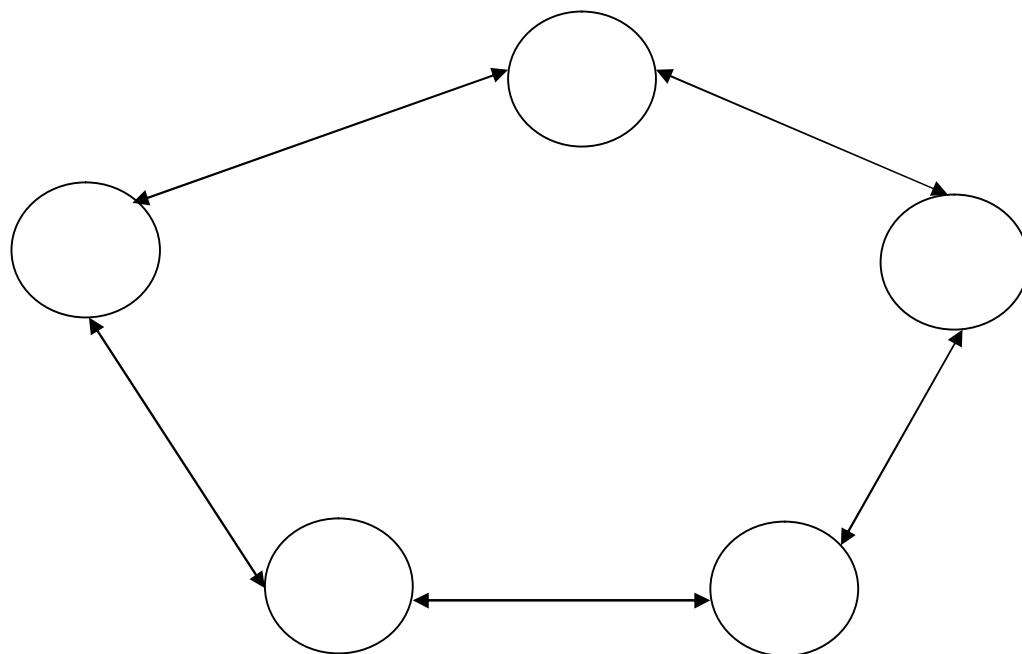
# Network Topologies

- § Term *network topology* refers to the way in which the nodes of a network are linked together
- § Although number network topologies are possible, four major ones are:
  - § Star network
  - § Ring network
  - § Completely connected network
  - § Multi-access bus network

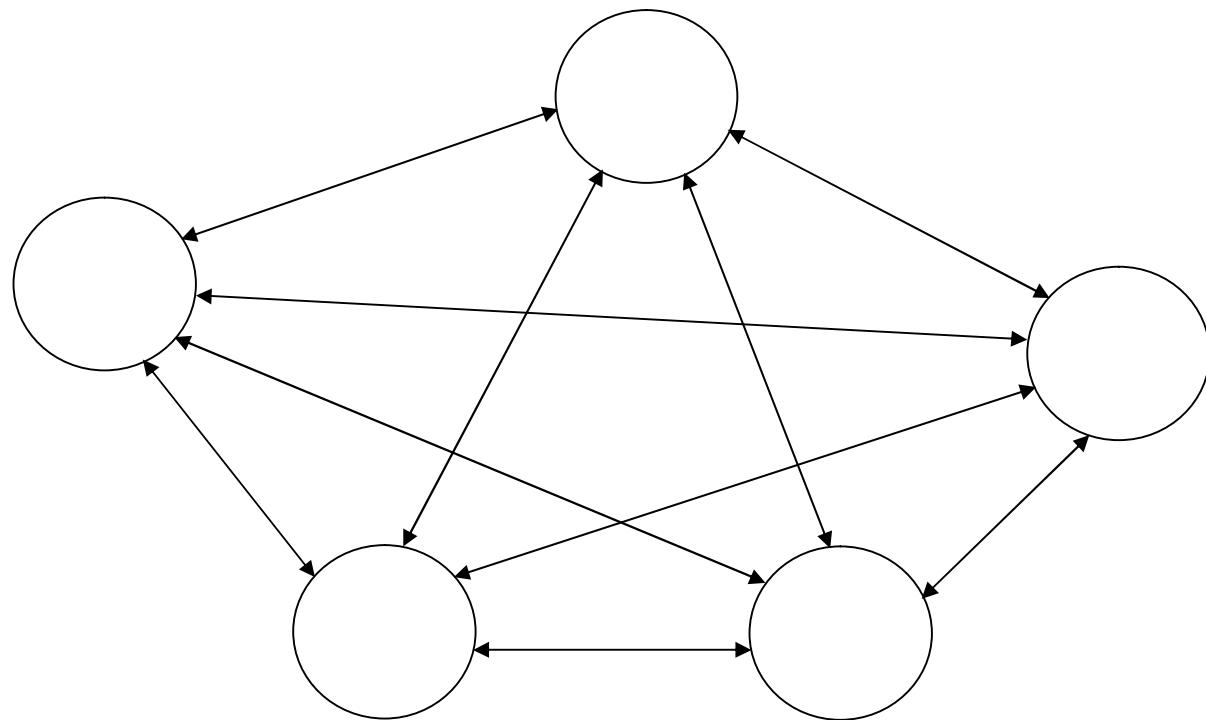
# Star Network



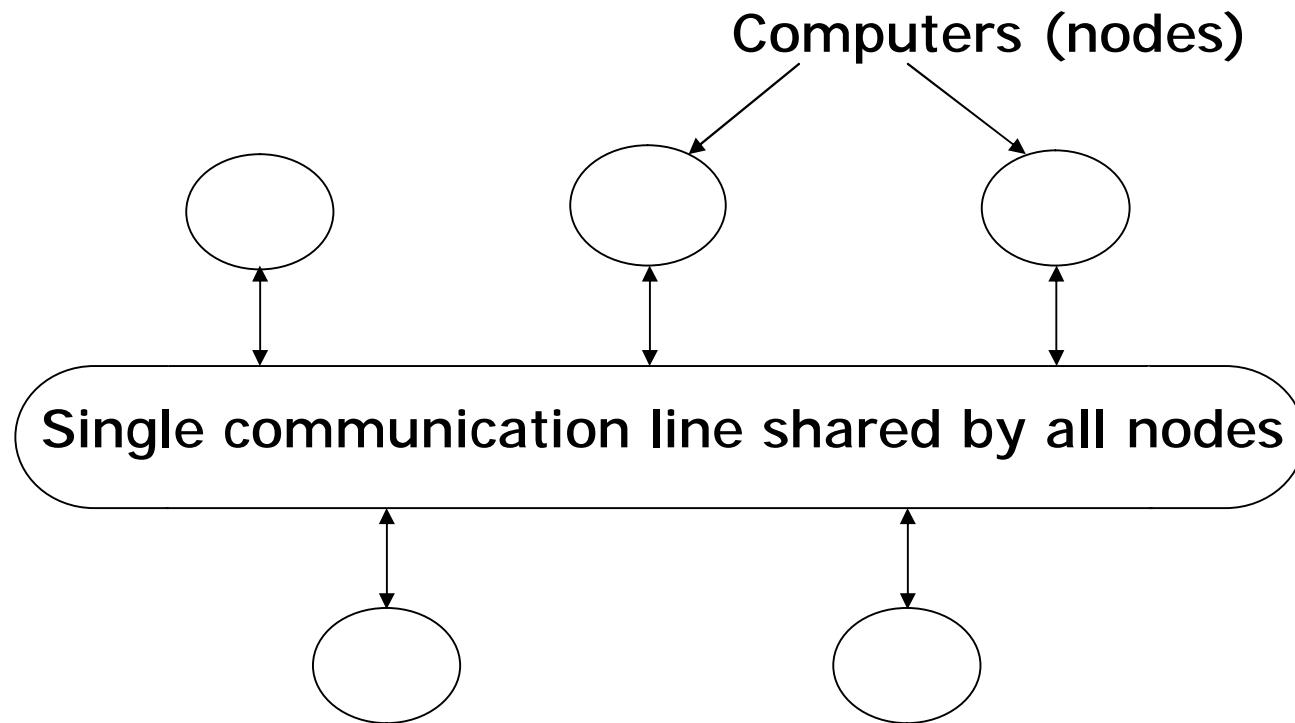
# Ring Network



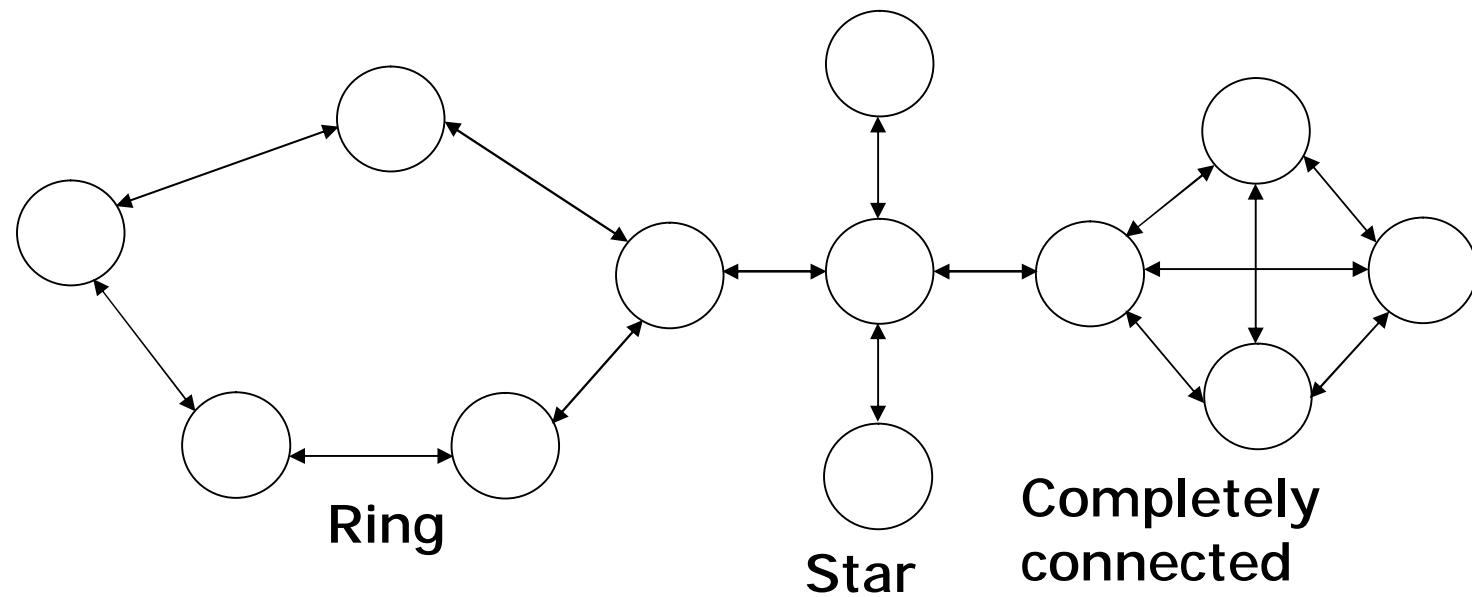
# Completely Connected Network



# Multi-Access Bus Network



# Hybrid Network



# Network Types

- § Networks are broadly classified into two types: Local Area Network (LAN) and Wide Area Network (WAN)
- § Local Area Network (LAN) as compared to WAN:
  - § Limited to a small geographic coverage
  - § Has much higher data transmission rate
  - § Experiences fewer data transmission errors
  - § Has lower data communication cost
  - § Typically owned by a single organization
- § Networks that share some of the characteristics of both LANs and WANs are referred to as Metropolitan Area Network (MAN)

# Communication Protocols

- § Protocol is a set of formal operating rules, procedures, or conventions that govern a given process
- § Communication protocol describes rules that govern transmission of data over communication networks
- § Roles of communication protocol:
  - § Data sequencing
  - § Data routing
  - § Data formatting
  - § Flow control
  - § Error control

(Continued on next slide)

# Communication Protocols

(Continued from previous slide)

- § Precedence and order of transmission
- § Connection establishment and termination
- § Data security
- § Log information.
- § Communication protocols are normally split up into a series of modules logically composed of a succession of layers.
- § Terms *protocol suite*, *protocol family*, or *protocol stack* are used to refer to the collection of protocols (of all layers) of a network system

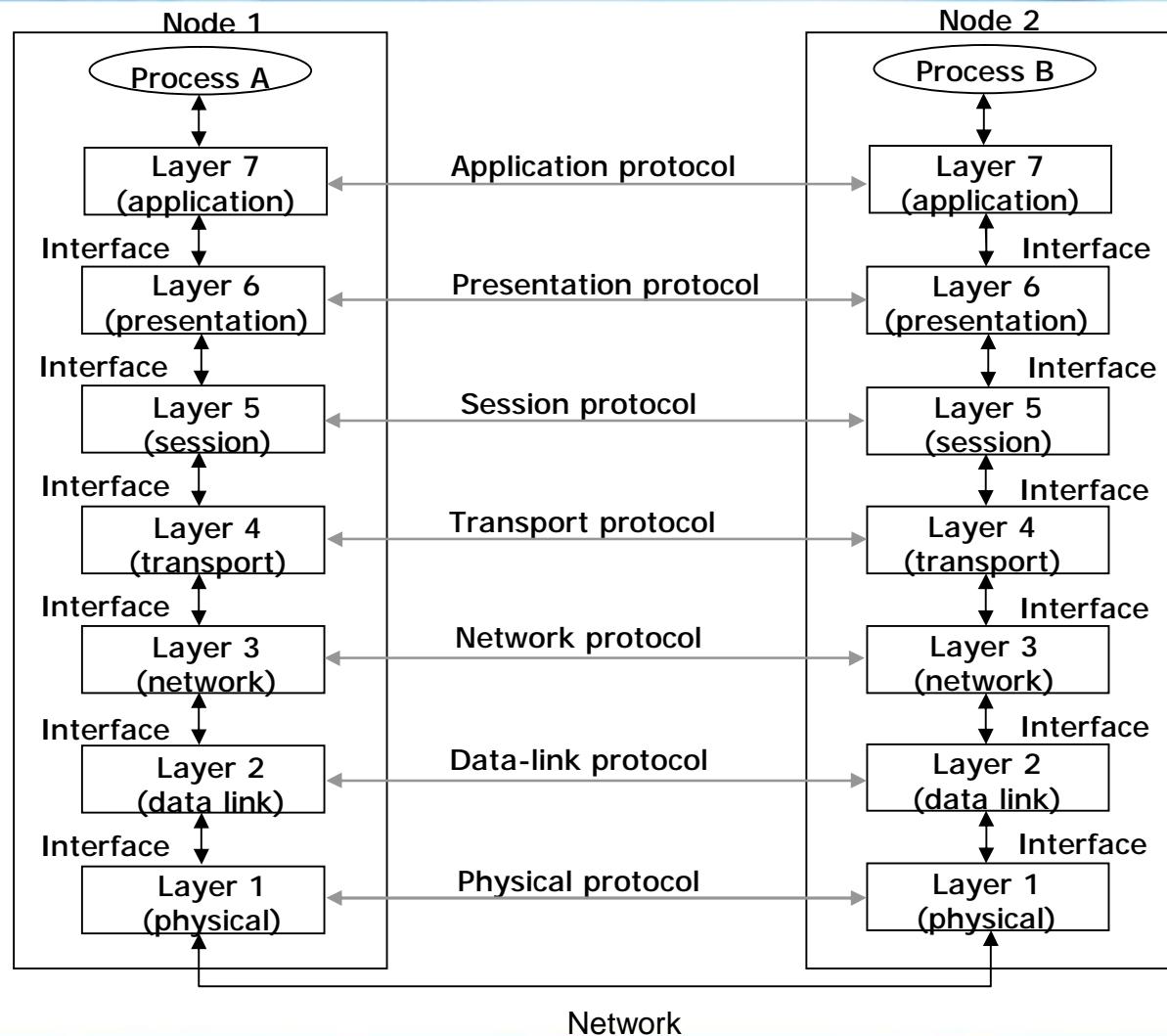
# Network Interface Card (NIC)

- § Hardware device that allows a computer to be connected to a network, both functionally and physically
- § Printed circuit board installed on to one of the expansion slots of computer
- § Provides a port on the back to which network cable is attached

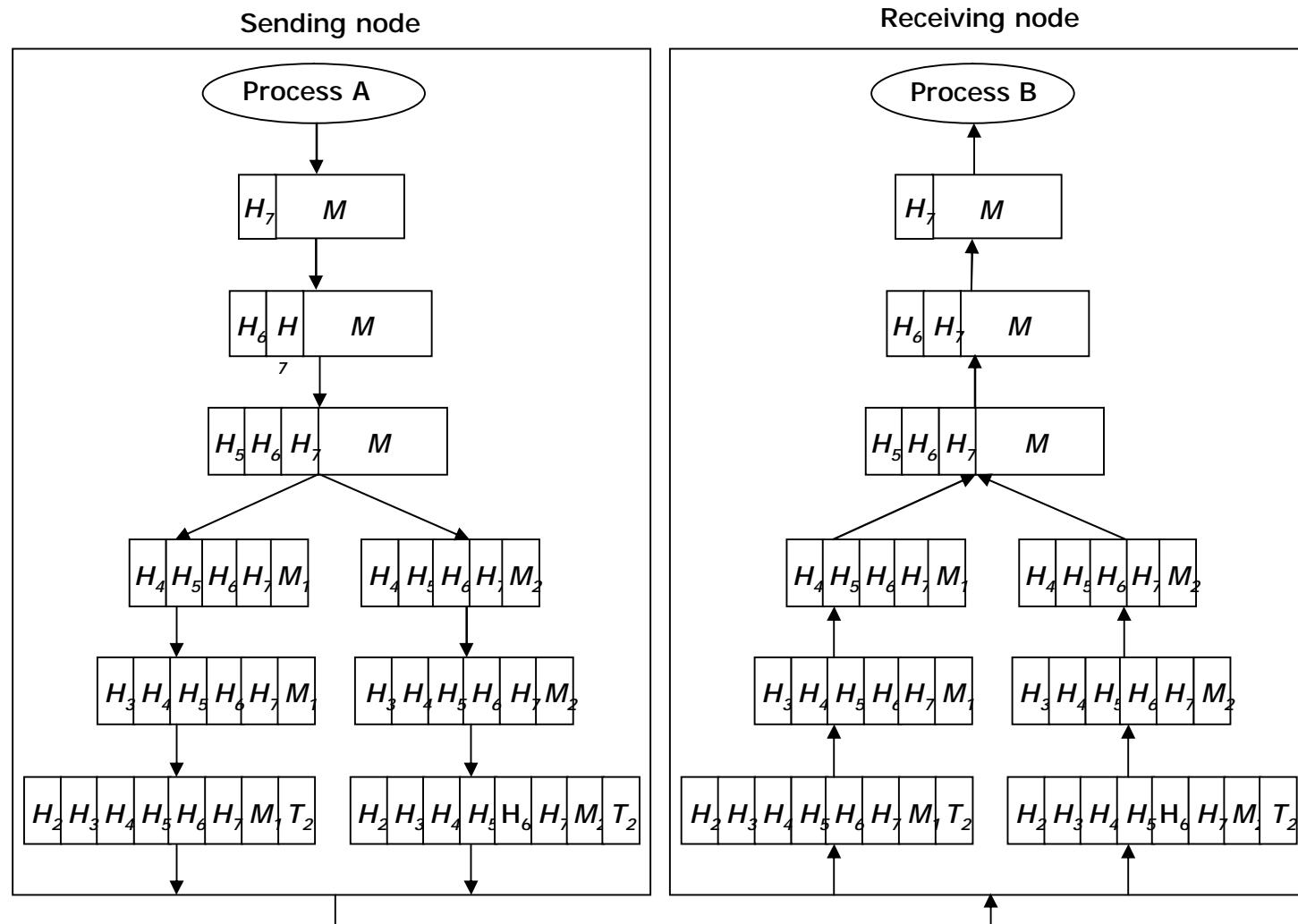
# The OSI Model

- § The Open System Interconnection (OSI) model is framework for defining standards for linking heterogeneous computers in a packet switched network
- § Standardized OSI protocol makes it possible for any two heterogeneous computer systems, located anywhere in the world, to easily communicate with each other
- § Separate set of protocols is defined for each layer in its seven-layer architecture. Each layer has an independent function

# Layers, Interfaces, and Protocols in the OSI Model



An example illustrating transfer of message M from sending node to the receiving node in the OSI model:  $H_n$ , header added by layer n;  $T_n$ , trailer added by layer n.



# Internetworking

- § Interconnecting two or more networks to form a single network is called *internetworking*, and the resulting network is called an *internetwork*
- § Goal of internetworking is to hide details of different physical networks, so that resulting internetwork functions as a single coordinated unit
- § Tools such as bridges, routers, brouters, and gateways are used for internetworking
- § The Internet is the best example of an internetwork

# Bridges

- § Operate at bottom two layers of the OSI model
- § Connect networks that use the same communication protocols above data-link layer but may use different protocols at physical and data-link layers

# Routers

- § Operates at network layer of the OSI model
- § Used to interconnect those networks that use the same high-level protocols above network layer
- § Smarter than bridges as they not only copy data from one network segment to another, but also choose the best route for the data by using routing table

# Gateways

- § Operates at the top three layers of the OSI model (session, presentation and application)
- § Used for interconnecting dissimilar networks that use different communication protocols
- § Since gateways interconnect dissimilar networks, protocol conversion is the major job performed by them

# Wireless Computing Systems

- § Wireless computing system uses wireless communication technologies for interconnecting computer systems
- § Enhances functionality of computing equipment by freeing communication from location constraints of wired computing systems
- § Wireless computing systems are of two types:
  - § **Fixed wireless systems:** Support little or no mobility of the computing equipment associated with the wireless network
  - § **Mobile wireless systems:** Support mobility of the computing equipment to access resources associated with the wireless network

# Wireless Technologies

- § 2G and 3G
- § Wireless LAN
- § WiMAX
- § Wireless Local Loop (WLL)
- § Radio-router
- § Multihop Wireless Network
- § Wireless Application Protocol (WAP)

# Distributed Computing Systems

- § Configuration where many independent computer systems are connected, and messages, processing task, programs, data, and other resources are transmitted between cooperating computer systems
- § Such an arrangement enables sharing of many hardware and software resources as well as information among several users who may be sitting far away from each other

# Main Advantages of Distributed Computing Systems

- § Inherently distributed applications
- § Information sharing among distributed users
- § Resource sharing
- § Shorter response times and higher throughput
- § Higher reliability
- § Extensibility and incremental growth
- § Better flexibility in meeting users' needs

# Keywords/Phrases

- § Amplifier
- § Amplitude Modulation (AM)
- § Application layer
- § ARPANET
- § Asynchronous transmission
- § Bandwidth
- § Baud
- § Bridge
- § Broadband
- § Broadcast
- § C-band transmission
- § Circuit switching
- § Coaxial cable
- § Common Carriers
- § Communication protocol
- § Communications satellite
- § Completely connected network
- § Computer network
- § Concentrators
- § Data-link layer
- § Demodulation
- § Dial-up line
- § Distributed Computing System
- § Ethernet
- § Fax modem
- § File Transfer Protocol (FTP)
- § Font-End Processors (FEP)
- § Frequency Modulation (FM)
- § Frequency-Division Multiplexing (FDM)
- § Full duplex
- § Gateway
- § Half duplex
- § Hop-by-hop routing
- § Hybrid network
- § Internet Protocol (IP)
- § Internetworking
- § ISDN (Integrated Services Digital Network)
- § Ku-band transmission
- § Leased line
- § Local Area Network (LAN)
- § Message switching

(Continued on next slide)

# Keywords/Phrases

- § Metropolitan Area Network (MAN)
- § Microwave system
- § Mobile computing
- § Modem
- § Modulation
- § Multi-access Bus network
- § Multiplexer
- § Narrowband
- § Network Interface Card (NIC)
- § Network layer
- § Network topology
- § Nomadic computing
- § Optical fibers
- § OSI Model
- § Packet switching
- § Phase Modulation (PM)
- § Physical layer
- § POTS (Plain Old Telephone Service)
- § Presentation layer
- § Protocol family
- § Protocol stack
- § Protocol suite (Continued from previous slide)
- § Repeater
- § Ring network
- § Router
- § Session layer
- § Simplex
- § Source routing
- § Star network
- § Store-and-forward
- § Synchronous transmission
- § Time-Division Multiplexing (TDM)
- § Transport Control Protocol (TCP)
- § Transport layer
- § Twisted-pair
- § Unshielded twisted-pair (UTP)
- § User Datagram Protocol (UDP)
- § Value Added Network (VAN)
- § Voiceband
- § VSAT (Very Small Aperture Terminals)
- § Wide Area Network (WAN)
- § Wireless network

# Chapter 18

## The Internet

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

In this chapter you will learn about:

- § The Internet
- § Evolution and basic services on Internet
- § World Wide Web (WWW)
- § WWW browsers
- § Uses of the Internet

# The Internet

- § The Internet is a network of computers that links many different types of computers all over the world
- § Network of networks sharing a common mechanism for addressing (identifying) computers, and a common set of communication protocols
- § Evolved from the basic ideas of ARPANET (the first WAN that had only four sites in 1969) for interconnecting computers
- § Initially used only by research organizations and universities to share and exchange information

(Continued on next slide)

# The Internet

(Continued from previous slide..)

- § In 1989, the US Government lifted restrictions on the use of the Internet and allowed it to be used for commercial purposes as well
- § Internet has rapidly grown and continues to grow at a rapid pace
- § Interconnects more than 30,000 networks, allowing more than 10 million computers and more than 50 million computer users in more than 150 countries to communicate with each other

# Basic Services of the Internet

- § **Electronic Mail (e-mail):** Allows user to send a mail (message) to another Internet user in any part of the world in a near-real-time manner
- § **File Transfer Protocol (FTP):** Allows user to move a file from one computer to another on the Internet
- § **Telnet:** Allows user to log in to another computer somewhere on the Internet
- § **Usenet News:** Allows group of users to exchange their views/ideas/information

# Electronic Mail

- § E-mail is a rapid and productive communication tool because:
  - § Faster than paper mail
  - § Unlike telephone, the persons communicating with each other need not be available at the same time
  - § Unlike fax documents, e-mail documents can be stored in a computer and be easily edited using editing programs

# File Transfer Protocol

- § Moving a file from a remote computer to ones own computer is known as downloading
- § Moving a file from ones own computer to a remote computer is known as uploading
- § Anonymous ftp site is a computer allowing a user to log in with a username of anonymous and password that is user's e-mail address.
- § Anonymous ftp sites are called publicly accessible sites because they can be accessed by any user on the Internet

# Telnet

Some common uses of telnet service are:

- § Using the computing power of the remote computer
- § Using a software on the remote computer
- § Accessing remote computer's database or archive
- § Logging in to ones own computer from another computer

# Usenet News

- § Several usenet news groups exist on the Internet and are called newsgroups
- § In a *moderated newsgroup* only selected members have the right to directly post (write) a message to the virtual notice board. Other members can only read the posted messages
- § In a *nonmoderated newsgroup* any member can directly post a message to the virtual notice board
- § *Netiquette* (network etiquette) deals with rules of framing messages that will not hurt others

# World Wide Web (WWW or W3)

- § Hypertext documents on the Internet are known as web pages
- § Web pages are created by using a special language called *HyperText Markup Language (HTML)*
- § WWW uses the client-server model and an Internet Protocol called *HyperText Transport Protocol (HTTP)* for interaction among the computers on the Internet
- § Any computer on the Internet that uses the HTTP protocol is called a web server and any computer that can access that server is called a web client

(Continued on next slide)

# World Wide Web (WWW or W3)

(Continued from previous slide..)

- § It uses the concept of *hypertext* for information storage and retrieval on the Internet
- § Hypertext documents enable this by using a series of links
- § Link is a special type of item in a hypertext document that connects the document to another document providing more information about the linked item

# Example of Hypertext Document

Pradeep K. Sinha has been involved in the research and development of distributed systems for almost a decade. At present Dr. Sinha is working at the Centre for Development of Advanced Computing (C-DAC), Pune, India. Before joining C-DAC, Dr. Sinha worked with the Multimedia Systems Research Laboratory, (MSRL) of Panasonic in Tokyo, Japan.

Links

# WWW Browsers

WWW browser is a special software loaded on a web client computer that normally provides following navigation facilities to users:

- § Does not require a user to remotely log in to a web server computer or to log out again when done
- § Allows user to visit the server computer's web site and to access information stored on it by specifying its *URL (Uniform Resource Locator)* address

(Continued on next slide)

# WWW Browsers

(Continued from previous slide..)

- § Allows user to create and maintain a personal *hotlist* of favorite URL addresses of server computers that user is likely to frequently visit in future
- § Allows user to download information in various formats from server computers to user's own computer

# Uses of the Internet

Some important current strategic uses of the Internet are:

- § On-line communication
- § Software sharing
- § Exchange of views on topics of common interest
- § Posting of information of general interest
- § Product promotion
- § Feedback about products
- § Customer support service
- § On-line journals and magazines
- § On-line shopping
- § World-wide video conferencing

# Keywords/Phrases

- § Anonymous ftp site
- § Browser
- § Download
- § Electronic mail (e-mail)
- § File Transfer Protocol (FTP)
- § Hypertext
- § Hypertext Transport Protocol (HTTP)
- § Internet
- § Newsgroup
- § Publicly accessible sites
- § Standard Generalized Markup Language (SGML)
- § Telnet
- § Uniform Resource Locator (URL)
- § Upload
- § Usenet
- § Web client
- § Web Server
- § World Wide Web (WWW)

# Chapter 19

## Multimedia

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

In this chapter you will learn about:

- § Multimedia
- § Multimedia computer system
- § Main components of multimedia and their associated technologies
- § Common multimedia applications

# Multimedia

- § Media is something that can be used for presentation of information.
- § Two basic ways to present some information are:
  - § **Unimedia presentation:** Single media is used to present information
  - § **Multimedia presentation:** More than one media is used to present information
- § Multimedia presentation of any information greatly enhances the comprehension capability of the user as it involves use of more of our senses

# Common Media

- § Common media for storage, access, and transmission of information are:
  - § Text (alphanumeric characters)
  - § Graphics (line drawings and images)
  - § Animation (moving images)
  - § Audio (sound)
  - § Video (Videographed real-life events)
- § Multimedia in information technology refers to use of more than one of these media for information presentation to users

# Multimedia Computer System

- § Multimedia computer system is a computer having capability to integrate two or more types of media (text, graphics, animation, audio, and video)
- § In general, size for multimedia information is much larger than plain text information
- § Multimedia computer systems require:
  - § Faster CPU
  - § Larger storage devices (for storing large data files)
  - § Larger main memory (for large data size)
  - § Good graphics terminals
  - § I/O devices to play any multimedia

(Continued on next slide)

# Text Media

(Continued from previous slide..)

- § Alphanumeric characters are used to present information in text form. Computers are widely used for text processing
- § Keyboards, OCRs, computer screens, and printers are some commonly used hardware devices for processing text media
- § Text editing, text searching, hypertext, and text importing/exporting are some highly desirable features of a multimedia computer system for better presentation and use of text information

# Graphics Media

- § *Computer graphics* deals with generation, representation, manipulation, and display of pictures (line drawings and images) with a computer
- § Locating devices (such as a mouse, a joystick, or a stylus), digitizers, scanners, digital cameras, computer screens with graphics display capability, laser printers, and plotters are some common hardware devices for processing graphics media
- § Some desirable features of a multimedia computer system are painting or drawing software, screen capture software, clip art, graphics importing, and software support for high resolution

# Animation Media

- § *Computer animation* deals with generation, sequencing, and display (at a specified rate) of a set of images (called frames) to create an effect of visual change or motion, similar to a movie film (video)
- § Animation is commonly used in those instances where videography is not possible or animation can better illustrate the concept than video
- § Animation deals with displaying a sequence of images at a reasonable speed to create an impression of movement. For a jerk-free full motion animation, 25 to 30 frames per second is required

(Continued on next slide)

# Animation Media

(Continued from previous slide..)

- § Scanners, digital cameras, video capture board interfaced to a video camera or VCR, computer monitors with image display capability, and graphics accelerator board are some common hardware devices for processing animation media
- § Some desirable features of a multimedia computer system with animation facility are animation creation software, screen capture software, animation clips, animation file importing, software support for high resolution, recording and playback capabilities, and transition effects

# Virtual Reality

§ Virtual reality is a relatively new technology using which the user can put a pair of goggles and a glove and tour a three-dimensional world that exists only in the computer, but appears realistic to the user

# Audio Media

- § *Computer audio* deals with synthesizing, recording, and playback of audio or sound with a computer
- § Sound board, microphone, speaker, MIDI devices, sound synthesizer, sound editor and audio mixer are some commonly used hardware devices for processing audio media
- § Some desirable features of a multimedia computer system are audio clips, audio file importing, software support for high quality sound, recording and playback capabilities, text-to-speech conversion software, speech-to-text conversion software, and voice recognition software

# Video Media

- § *Computer video* deals with recording and display of a sequence of images at a reasonable speed to create an impression of movement. Each individual image of such a sequence is called a frame
- § Video camera, video monitor, video board, and video editor are some of the commonly used hardware devices for processing video media
- § Some desirable features of a multimedia computer system with video facility are video clips and recording and playback capabilities

# Multimedia Applications

- § Multimedia presentation
- § Foreign language learning
- § Video games
- § Special effects in films
- § Multimedia kiosks as help desks
- § Animated advertisements
- § Multimedia conferencing

# Media Center Computer

- § There is a growing trend of owning a personal computer (PC) at home like other electronic equipment
- § New terminologies like “infotainment” and “edutainment” have evolved to refer to computers as versatile tools
- § Media center PC provides following functionalities:
  - § Server as PC, TV, radio, and music system
  - § Serve as digital photo album and digital library
  - § Server as Game station and DVD/CD Player
  - § Allows play, pause, and record of TV programs
  - § Provides Electronic Programming Guide (EPG)

# Media Center Computer



# Keywords/Phrases

- § Animation
- § Audio
- § Clip art
- § Cognitive graphics
- § Computer Aided Design (CAD)
- § Computer Aided Manufacturing (CAM)
- § Frames
- § Generative graphics
- § Graphics
- § Multimedia
- § Media Center Computer
- § Pixel
- § Refresh rate
- § Text
- § Transducer
- § Transition effects
- § Video
- § Virtual reality

# Chapter 20

## Classification of Computers

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

In this chapter you will learn about:

- § Classifications of computers
- § Common types of computers today
- § Characteristic features of various types of computers in use today

# Computer Classification

- § Traditionally, computers were classified by their size, processing speed, and cost
- § Based on these factors, computers were classified as microcomputers, minicomputers, mainframes, and supercomputers
- § However, with rapidly changing technology, this classification is no more relevant
- § Today, computers are classified based on their mode of use

# Types of Computers

Based on their mode of use, computers are classified as:

- § Notebook computers
- § Personal computers
- § Workstations
- § Mainframe systems
- § Supercomputers
- § Clients and servers
- § Handheld computers

# Notebook Computers

- § Portable computers mainly meant for use by people who need computing resource wherever they go
- § Approximately of the size of an 8½ x 11 inch notebook and can easily fit inside a briefcase
- § Weigh around 2 kg only.
- § Comfortably placed on ones lap while being used. Hence, they are also called *laptop PC*
- § Lid with display screen is foldable in a manner that when not in use it can be folded to flush with keyboard to convert the system into notebook form

(Continued on next slide)

# Notebook Computers

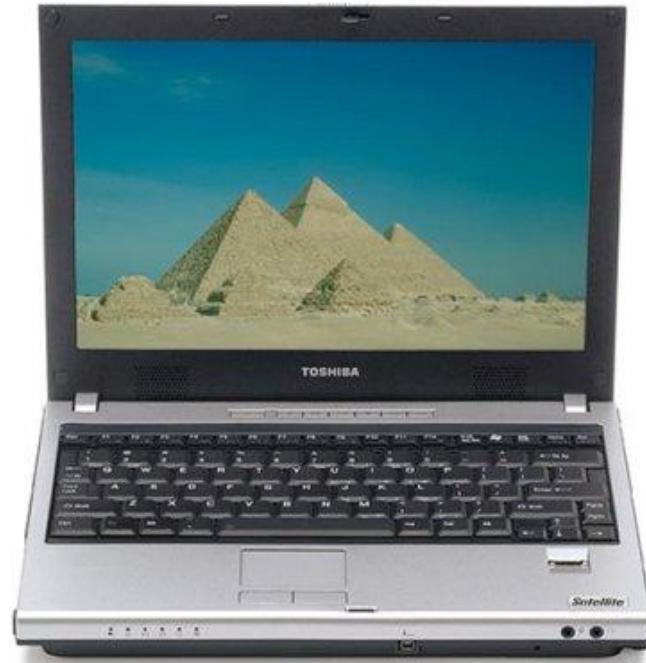
(Continued from previous slide..)

- § Designed to operate with chargeable batteries
- § Mostly used for word processing, spreadsheet computing, data entry, and power point presentations
- § Normally run MS-DOS or MS WINDOWS operating system
- § Some manufacturers are also offering models with GNU/Linux or its distributions
- § Each device of laptop is designed to use little power and remain suspended if not used

# Notebook Computers

Foldable flat screen →

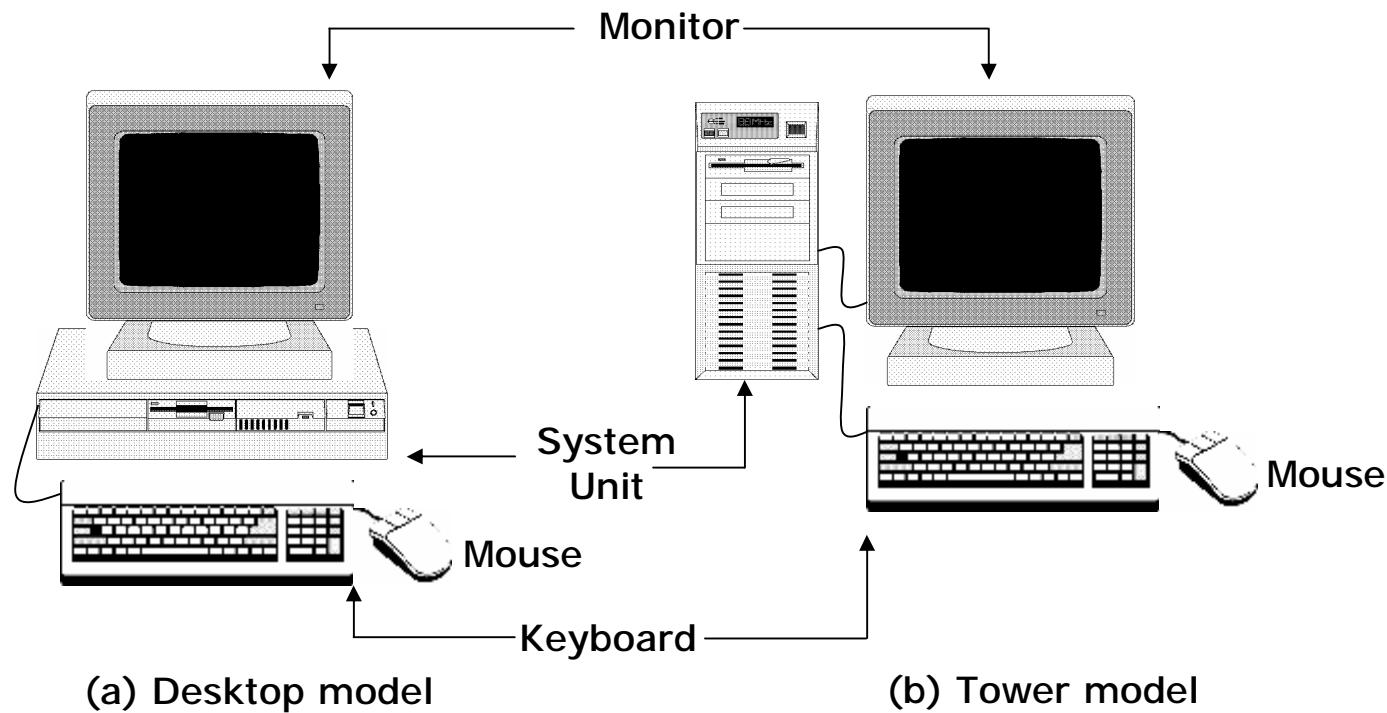
Keyboard, trackball, hard disk, floppy disk drive, etc. are in this unit →



# Personal Computers (PCs)

- § Non-portable, general-purpose computer that fits on a normal size office table
- § Designed to meet personal computing needs of individuals
- § Often used by children and adults for education and entertainment also
- § Generally used by one person at a time, supports multitasking
- § Two common models of PCs are desktop model and tower model
- § Popular OS are MS-DOS, MS-Windows, Windows-NT, Linux, and UNIX

# Common PC Models



# Workstations

- § Powerful desktop computer designed to meet the computing needs of engineers, architects, and other professionals
- § Provides greater processing power, larger storage, and better graphics display facility than PCs
- § Commonly used for computer-aided design, multimedia applications, simulation of complex scientific and engineering problems, and visualization
- § Generally run the UNIX operating system or a variation of it
- § Operating system is generally designed to support multiuser environment

# Mainframe Systems

- § Mainly used by large organizations as banks, insurance companies, hospitals, railways, etc.
- § Used for data handling and information processing requirements
- § Used in such environments where a large number of users need to share a common computing facility
- § Oriented to input/output-bound applications

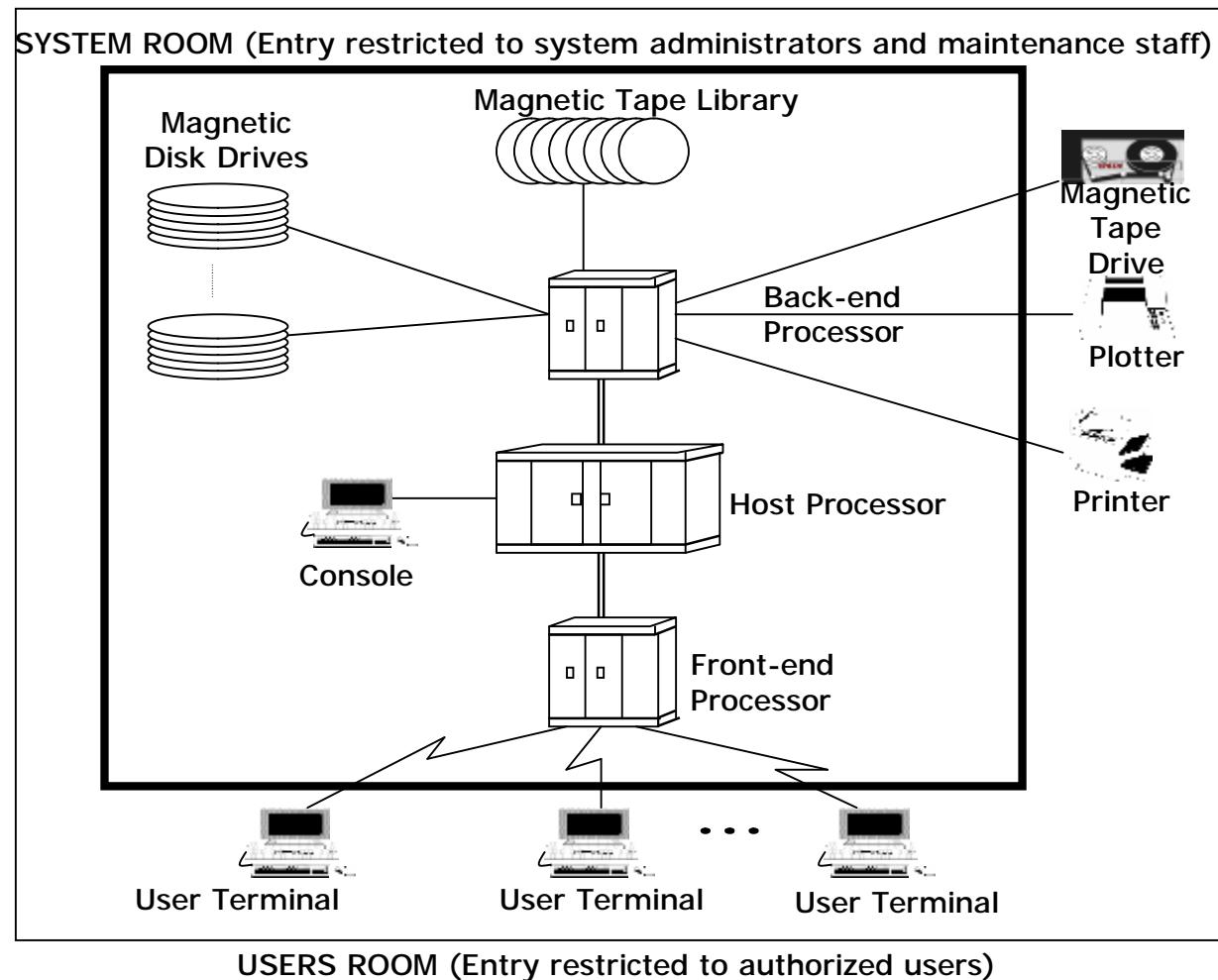
(Continued on next slide)

# Mainframe Systems

(Continued from previous slide..)

- § Typically consist of a host computer, front-end computer, back-end computer, console terminals, magnetic disk drives, tape drives, magnetic tape library, user terminals, printers, and plotters
- § Typical mainframe system looks like a row of large file cabinets and needs a large room
- § Smaller configuration (slower host and subordinate computers, lesser storage space, and fewer user terminals) is often referred to as a *minicomputer system*

# Mainframe Computer Systems



# Supercomputers

- § Most powerful and most expensive computers available at a given time.
- § Primarily used for processing complex scientific applications that require enormous processing power
- § Well known supercomputing applications include:
  - § Analysis of large volumes of seismic data
  - § Simulation of airflow around an aircraft
  - § Crash simulation of the design of an automobile
  - § Solving complex structure engineering problems
  - § Weather forecasting

(Continued on next slide)

# Supercomputers

(Continued from previous slide..)

- § Supercomputers also support multiprogramming
- § Supercomputers primarily address processor-bound applications

# Parallel Processing Systems

- § Use multiprocessing and parallel processing technologies to solve complex problems faster
- § Also known as *parallel computers* or *parallel processing systems*
- § Modern supercomputers employ hundreds of processors and are also known as *massively parallel processors*

# C-DAC's PARAM 10000 Supercomputer



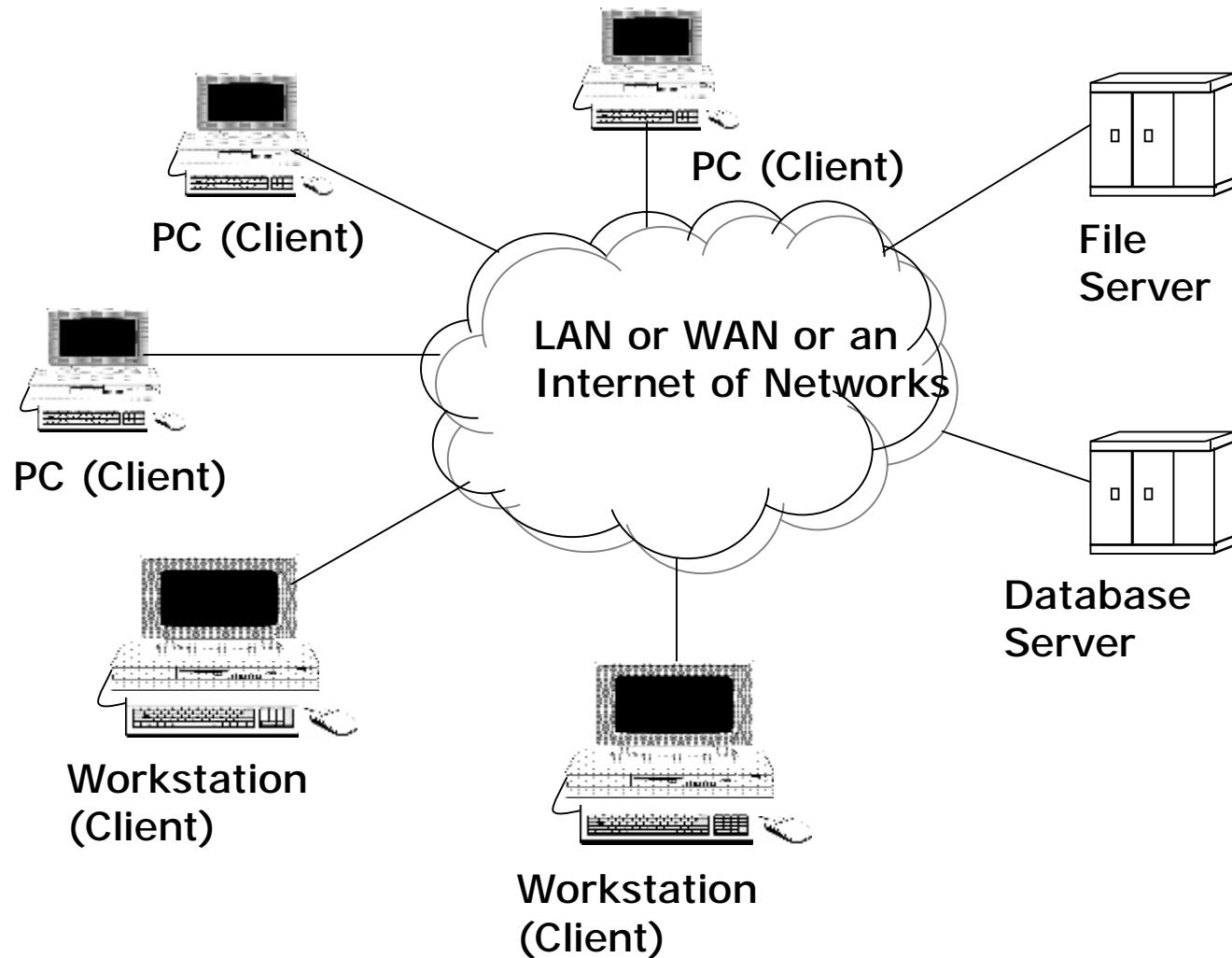
# Client and Server Computers

- § Client-server computing environment has multiple clients, one/more servers, and a network
- § *Client* is a PC/workstation with user-friendly interface running client processes that send service requests to the server
- § *Server* is generally a relatively large computer that manages a shared resource and provides a set of shared user services to the clients
- § Server runs the server process that services client requests for use of managed resources
- § *Network* may be a single LAN or WAN or an internet work

# Client-Server Computing

- § Involves splitting an application into tasks and putting each task on computer where it can be handled most efficiently
- § Computers and operating systems of a client and a server may be different
- § Common for one server to use the services of another server, and hence act both as client and server
- § Concept of client and server computers is purely role-based and may change dynamically as the role of a computer changes

# Client-Server Computing Environment



# Handheld Computers

- § Small computing device that can be used by holding in hand, also known as *palmtop*
- § Size, weight, and design are such that it can be used comfortably by holding in hand
- § Types of Handheld are:
  - § **Tablet PC:** Miniaturized laptop with light weight, screen flip, handwriting and voice recognition
  - § **PDA/Pocket PC:** Acts as PIM device with LCD touch screen, pen for handwriting recognition, PC based synchronization, and optionally mobile phone services
  - § **Smartphone:** Fully functional mobile phone with computing power, voice centric, do not have a touch screen and are smaller than PDA

# Handheld Computers



(a) Table PC



(b) PDA/Pocket PC



(c) Smartphone

# Comparison of Different Types of Computers

Types of Computers Key features	Note book	PC	Work station	Mainframe system	Super computer	Client	Server	Handheld
Size	Very small (can be placed on ones lap)	Small (can be placed on an office table)	Medium (slightly larger than PC)	Large (needs a large room)	Large (needs a large room)	Generally small (may be large if it is also play the role of a server)	Generally large	Very small (can be placed on ones palm)
Processing power	Low	Low	High	Higher	Highest	Generally low	Generally high	Low
Main memory capacity	Low	Low	High	Higher	Highest	Generally low	Generally high	Low
Hard disk storage capacity	Low	Low	High	Highest	Higher	Generally low	Generally high	Low
Has its own monitor, keyboard, and mouse for user interface	Yes	Yes	Yes	Generally no	Generally no	Yes	Generally no	No

(Continued on next slide)

# Comparison of Different Types of Computers

(Continued from previous slide..)

Types of Computers Key features	Notebook	PC	Work station	Mainframe system	Super computer	Client	Server	Handheld
Display facility	Foldable flat screen small display	Medium size display screen	Large-screen color monitor which can display high resolution graphics	Generally not available	Generally not available	Medium to large screen monitor	Generally not available	Small display
Single/multiple processors	Single	Generally single	Generally multiple	Multiple	Multiple	Generally single	Generally multiple	Single
Single/multiple – User oriented	Single	Single	Generally single	Multiple	Multiple	Single	Multiple	Single
Popular operating systems	MS-DOS, MS-Windows	MS-DOS, MS-Windows, Windows-NT, Linux, Unix	Unix or a variation of it	A variation of Unix, or proprietary	A variation of Unix, or proprietary	MS-DOS, MS-Windows, Windows-NT, Linux, Unix	Windows -NT, Unix or its variation, or proprietary	MS-Windows Mobile, Palm OS, Symbian OS, Linux, Blackberry OS

(Continued on next slide)

# Comparison of Different Types of Computers

(Continued from previous slide..)

Types of Computers Key features	Notebook	Personal Computer	Work station	Mainframe system	Super computer	Client	Server	Handheld
Popular usage	Word processing; Spreadsheet ; Data Entry; Preparing presentation materials; and Making presentations	Personal computing needs of individuals either in their working places or at their homes; and Education and entertainment of children and adults	Computing needs of engineers, architects, designers; Simulation of complex scientific and engineering problems and visualizing the results of simulation; and Multimedia applications	Data and information processing of I/O-bound applications	Large processor-bound applications like complex scientific simulations	Provide highly user-friendly interface in a client-server computing environment	Manage a shared resource and provide a set of shared user services in a client-server computing environment	Computing, Personal Information Management (PIM), cell phone, digital camera
Major vendors	IBM, Compaq, Siemens, Toshiba	IBM, Apple, Compaq, Dell, Zenith, Siemens, Toshiba, Hewlett-Packard	Sun Microsystems, IBM, DEC, Hewlett-Packard, Silicon Graphics	IBM, DEC	Cray, IBM, Silicon Graphics, Fujitsu, Intel, C-DAC	Same as PC and Workstation vendors	Same as Workstation, Mainframe System, & Super-computer vendors	Nokia, Sony, Motorola, Dell, Hewlett-Packard

# Key Words/Phrases

- § Back-end computer
- § Client computer
- § Client process
- § Front-end computer
- § Host computer
- § Handheld
- § I/O-bound application
- § Laptop PC
- § Mainframe system
- § Massively parallel processors
- § Minicomputer
- § Notebook computer
- § Parallel computers
- § Parallel processing system
- § Personal Computer (PC)
- § Processor-bound application
- § Server computer
- § Server process
- § Supercomputer
- § System board
- § Workstation

# Chapter 21

## Introduction to C

# Programming Language

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

# Learning Objectives

In this chapter you will learn about:

- § Features of C
- § Various constructs and their syntax
- § Data types and operators in C
- § Control and Loop Structures in C
- § Functions in C
- § Writing programs in C

# Features

- § Reliable, simple, and easy to use
- § Has virtues of high-level programming language with efficiency of assembly language
- § Supports user-defined data types
- § Supports modular and structured programming concepts
- § Supports a rich library of functions
- § Supports pointers with pointer operations
- § Supports low-level memory and device access
- § Small and concise language
- § Standardized by several international standards body

# C Character Set

Category	Valid Characters	Total
Uppercase alphabets	A, B, C, ..., Z	26
Lowercase alphabets	a, b, c, ..., z	26
Digits	0, 1, 2, ..., 9	10
Special characters	~ ` ! @ # % ^ & * ( ) _ - + =   \ { } [ ] : ; " ' < > , . ? /	31
		93

# Constants

- § Constant is a value that never changes
- § Three primitive types of constants supported in C are:
  - § Integer
  - § Real
  - § Character

# Rules for Constructing Integer Constants

- § Must have at least one digit
- § + or – sign is optional
- § No special characters (other than + and – sign) are allowed
- § Allowable range is:
  - § -32768 to 32767 for integer and short integer constants (16 bits storage)
  - § -2147483648 to 2147483647 for long integer constants (32 bits storage)
- § Examples are: 8, +17, -6

# Rules for Constructing Real Constants in Exponential Form

- § Has two parts – mantissa and exponent - separated by 'e' or 'E'
- § Mantissa part is constructed by the rules for constructing real constants in fractional form
- § Exponent part is constructed by the rules for constructing integer constants
- § Allowable range is -3.4e38 to 3.4e38
- § Examples are: 8.6e5, +4.3E-8, -0.1e+4

# Rules for Constructing Character Constants

- § Single character from C character set
- § Enclosed within single inverted comma (also called single quote) punctuation mark
- § Examples are:      'A'      'a'      '8'      '%'

# Variables

- § Entity whose value may vary during program execution
- § Has a name and type associated with it
- § Variable name specifies programmer given name to the memory area allocated to a variable
- § Variable type specifies the type of values a variable can contain
- § Example: In  $i = i + 5$ ,  $i$  is a variable

# Rules for Constructing Variable Names

- § Can have 1 to 31 characters
- § Only alphabets, digits, and underscore (as in *last\_name*) characters are allowed
- § Names are case sensitive (*nNum* and *nNUM* are different)
- § First character must be an alphabet
- § Underscore is the only special character allowed
- § Keywords cannot be used as variable names
- § Examples are: I saving\_2007 ArrSum

# Data Types Used for Variable Type Declaration

Data Type	Minimum Storage Allocated	Used for Variables that can contain
int	2 bytes (16 bits)	integer constants in the range -32768 to 32767
short	2 bytes (16 bits)	integer constants in the range -32768 to 32767
long	4 bytes (32 bits)	integer constants in the range -2147483648 to 2147483647
float	4 bytes (32 bits)	real constants with minimum 6 decimal digits precision
double	8 bytes (64 bits)	real constants with minimum 10 decimal digits precision
char	1 byte (8 bits)	character constants
enum	2 bytes (16 bits)	Values in the range -32768 to 32767
void	No storage allocated	No value assigned

# Variable Type Declaration Examples

int	count;
short	index;
long	principle;
float	area;
double	radius;
char	c;

# Standard Qualifiers in C

Category	Modifier	Description
Lifetime	auto register static extern	Temporary variable Attempt to store in processor register, fast access Permanent, initialized Permanent, initialized but declaration elsewhere
Modifiability	const volatile	Cannot be modified once created May be modified by factors outside program
Sign	signed unsigned	+ or – + only
Size	short long	16 bits 32 bits

# Lifetime and Visibility Scopes of Variables

- § Lifetime of all variables (except those declared as *static*) is same as that of function or statement block it is declared in
- § Lifetime of variables declared in global scope and static is same as that of the program
- § Variable is visible and accessible in the function or statement block it is declared in
- § Global variables are accessible from anywhere in program
- § Variable name must be unique in its visibility scope
- § Local variable has access precedence over global variable of same name

# Keywords

- § *Keywords* (or reserved words) are predefined words whose meanings are known to C compiler
- § C has 32 keywords
- § Keywords cannot be used as variable names

auto  
break  
case  
char  
const  
continue  
default  
do

double  
else  
enum  
extern  
float  
for  
goto  
if

int  
long  
register  
return  
short  
signed  
sizeof  
static

struct  
switch  
typedef  
union  
unsigned  
void  
volatile  
while

# Comments

- § Comments are enclosed within /\* and \*/
- § Comments are ignored by the compiler
- § Comment can also split over multiple lines
- § Example: /\* This is a comment statement \*/

# Operators

- § Operators in C are categorized into data access, arithmetic, logical, bitwise, and miscellaneous
- § **Associativity** defines the order of evaluation when operators of same precedence appear in an expression
  - §  $a = b = c = 15$ , '=' has R → L associativity
  - § First  $c = 15$ , then  $b = c$ , then  $a = b$  is evaluated
- § **Precedence** defines the order in which calculations involving two or more operators is performed
  - §  $x + y * z$  , '\*' is performed before '+'

# Arithmet ic Operators

Operator	Meaning with Example	Associativity	Precedence
<b>Arithmet ic Operators</b>			
+	Addition; $x + y$	$L \rightarrow R$	4
-	Subtraction; $x - y$	$L \rightarrow R$	4
*	Multiplication; $x * y$	$L \rightarrow R$	3
/	Division; $x / y$	$L \rightarrow R$	3
%	Remainder (or Modulus); $x \% y$	$L \rightarrow R$	3
++	Increment;		
	$x++$ means post-increment (increment the value of $x$ by 1 after using its value);	$L \rightarrow R$	1
	$++x$ means pre-increment (increment the value of $x$ by 1 before using its value)	$R \rightarrow L$	2

# Arithmetic Operators

Operator	Meaning with Example	Associativity	Precedence
<b>Arithmetic Operators</b>			
--	Decrement;		
	x-- means post-decrement (decrement the value of x by 1 after using its value);	L → R	1
	--x means pre-decrement (decrement the value of x by 1 before using its value)	R → L	2
=	x = y means assign the value of y to x	R → L	14
+=	x += 5 means x = x + 5	R → L	14
-=	x -= 5 means x = x - 5	R → L	14
*=	x *= 5 means x = x * 5	R → L	14
/=	x /= 5 means x = x / 5	R → L	14
%=	x %= 5 means x = x % 5	R → L	14

# Logical Operators

Operator	Meaning with Example	Associativity	Precedence
<b>Logical Operators</b>			
!	Reverse the logical value of a single variable; !x means if the value of x is non-zero, make it zero; and if it is zero, make it one	R → L	2
>	Greater than; x > y	L → R	6
<	Less than; x < y	L → R	6
>=	Greater than or equal to; x >= y	L → R	6
<=	Less than or equal to; x <= y	L → R	6
==	Equal to; x == y	L → R	7
!=	Not equal to; x != y	L → R	7
&&	AND; x && y means both x and y should be true (non-zero) for result to be true	L → R	11
	OR; x    y means either x or y should be true (non-zero) for result to be true	L → R	12
z?x:y	If z is true (non-zero), then the value returned is x, otherwise the value returned is y	R → L	13

# Bitwise Operators

Operator	Meaning with Example	Associativity	Precedence
<b>Bitwise Operators</b>			
$\sim$	Complement; $\sim x$ means All 1s are changed to 0s and 0s to 1s	$R \rightarrow L$	2
$\&$	AND; $x \& y$ means $x$ AND $y$	$L \rightarrow R$	8
$ $	OR; $x   y$ means $x$ OR $y$	$L \rightarrow R$	10
$\wedge$	Exclusive OR; $x \wedge y$ means $x \oplus y$	$L \rightarrow R$	9
$<<$	Left shift; $x << 4$ means shift all bits in $x$ four places to the left	$L \rightarrow R$	5
$>>$	Right shift; $x >> 3$ means shift all bits in $x$ three places to the right	$L \rightarrow R$	5
$\&=$	$x \&= y$ means $x = x \& y$	$R \rightarrow L$	14
$ =$	$x  = y$ means $x = x   y$	$R \rightarrow L$	14
$\wedge=$	$x \wedge= y$ means $x = x \wedge y$	$R \rightarrow L$	14
$<<=$	$x <<= 4$ means shift all bits in $x$ four places to the left and assign the result to $x$	$R \rightarrow L$	14
$>>=$	$x >>= 3$ means shift all bits in $x$ three places to the right and assign the result to $x$	$R \rightarrow L$	14

# Data Access Operators

Operator	Meaning with Example	Associativity	Precedence
<b>Data Access Operators</b>			
$x[y]$	Access $y^{\text{th}}$ element of array $x$ ; $y$ starts from zero and increases monotonically up to one less than declared size of array	$L \rightarrow R$	1
$x.y$	Access the member variable $y$ of structure $x$	$L \rightarrow R$	1
$x \rightarrow y$	Access the member variable $y$ of structure $x$	$L \rightarrow R$	1
$\&x$	Access the address of variable $x$	$R \rightarrow L$	2
$*x$	Access the value stored in the storage location (address) pointed to by pointer variable $x$	$R \rightarrow L$	2

# Miscellaneous Operators

Operator	Meaning with Example	Associativity	Precedence
<b>Miscellaneous Operators</b>			
x(y)	Evaluates function x with argument y	L → R	1
sizeof (x)	Evaluate the size of variable x in bytes	R → L	2
sizeof (type)	Evaluate the size of data type "type" in bytes	R → L	2
(type) x	Return the value of x after converting it from declared data type of variable x to the new data type "type"	R → L	2
x,y	Sequential operator (x then y)	L → R	15

# Statements

- § C program is a combination of statements written between { and } braces
- § Each statement performs a set of operations
- § Null statement, represented by ";" or empty {} braces, does not perform any operation
- § A simple statement is terminated by a semicolon ;
- § Compound statements, called *statement block*, perform complex operations combining null, simple, and other block statements

# Examples of Statements

```
§ a = (x + y) * 10;      /* simple statement */
```

```
§ if (sell > cost) /* compound statement  
follows */
```

```
{  
    profit = sell - cost;  
    printf ("profit is %d", profit);  
}  
else      /* null statement follows */  
{  
}
```

# Simple I/O Operations

- § C has no keywords for I/O operations
- § Provides standard library functions for performing all I/O operations

# Basic Library Functions for I/O Operations

I/O Library Functions	Meanings
getch()	Inputs a single character (most recently typed) from standard input (usually console).
getche()	Inputs a single character from console and echoes (displays) it.
getchar()	Inputs a single character from console and echoes it, but requires <i>Enter</i> key to be typed after the character.
putchar() or putch()	Outputs a single character on console (screen).
scanf()	Enables input of formatted data from console (keyboard). Formatted input data means we can specify the data type expected as input. Format specifiers for different data types are given in Figure 21.6.
printf()	Enables obtaining an output in a form specified by programmer (formatted output). Format specifiers are given in Figure 21.6. Newline character "\n" is used in <i>printf()</i> to get the output split over separate lines.
gets()	Enables input of a string from keyboard. Spaces are accepted as part of the input string, and the input string is terminated when <i>Enter</i> key is hit. Note that although <i>scanf()</i> enables input of a string of characters, it does not accept multi-word strings (spaces in-between).
puts()	Enables output of a multi-word string

# Basic Format Specifiers for *scanf()* and *printf()*

Format Specifiers	Data Types
%d	integer (short signed)
%u	integer (short unsigned)
%ld	integer (long signed)
%lu	integer (long unsigned)
%f	real (float)
%lf	real (double)
%c	character
%s	string

# Formatted I/O Example

```
/* A portion of C program to illustrate formatted input and output */
```

```
int maths, science, english, total;  
float percent;  
  
clrscr(); /* A C library function to make the screen clear */  
printf ("Maths marks = "); /* Displays "Maths marks = " */  
scanf ("%d", &maths); /* Accepts entered value and stores in variable "maths" */  
printf ("\n Science marks = "); /* Displays "Science marks = " on next line because of \n */  
scanf ("%d", &science); /* Accepts entered value and stores in variable "science" */  
printf ("\n English marks = "); /* Displays "English marks = " on next line because of \n */  
scanf ("%d", &english); /* Accepts entered value and stores in variable "english" */  
  
total = maths + science + english;  
percent = total/3; /* Calculates percentage and stores in variable "percent" */  
  
printf ("\n Percentage marks obtained = %f", percent);  
/* Displays "Percentage marks obtained = 85.66" on next line  
because of \n */
```

(Continued on next slide)

# Formatted I/O Example

(Continued from previous slide..)

## Output:

Maths marks = 92

Science marks = 87

English marks = 78

Percentage marks obtained = 85.66

# Preprocessor Directives

- § *Preprocessor* is a program that prepares a program for the C compiler
- § Three common preprocessor directives in C are:
  - § **#include** – Used to look for a file and place its contents at the location where this preprocessor directive is used
  - § **#define** – Used for macro expansion
  - § **#ifdef..#endif** – Used for conditional compilation of segments of a program

# Examples of Preprocessor Directives

```
#include <stdio.h>
#define PI 3.1415
#define AND  &&
#define ADMIT      printf ("The candidate can be admitted");
```

```
#ifdef WINDOWS
```

```
.
```

```
.
```

```
.
```

Code specific to windows operating system

```
.
```

```
.
```

```
.
```

```
#else
```

```
.
```

```
.
```

```
.
```

Code specific to Linux operating system

```
.
```

```
.
```

```
.
```

```
#endif
```

```
.
```

```
.
```

```
.
```

Code common to both operating systems

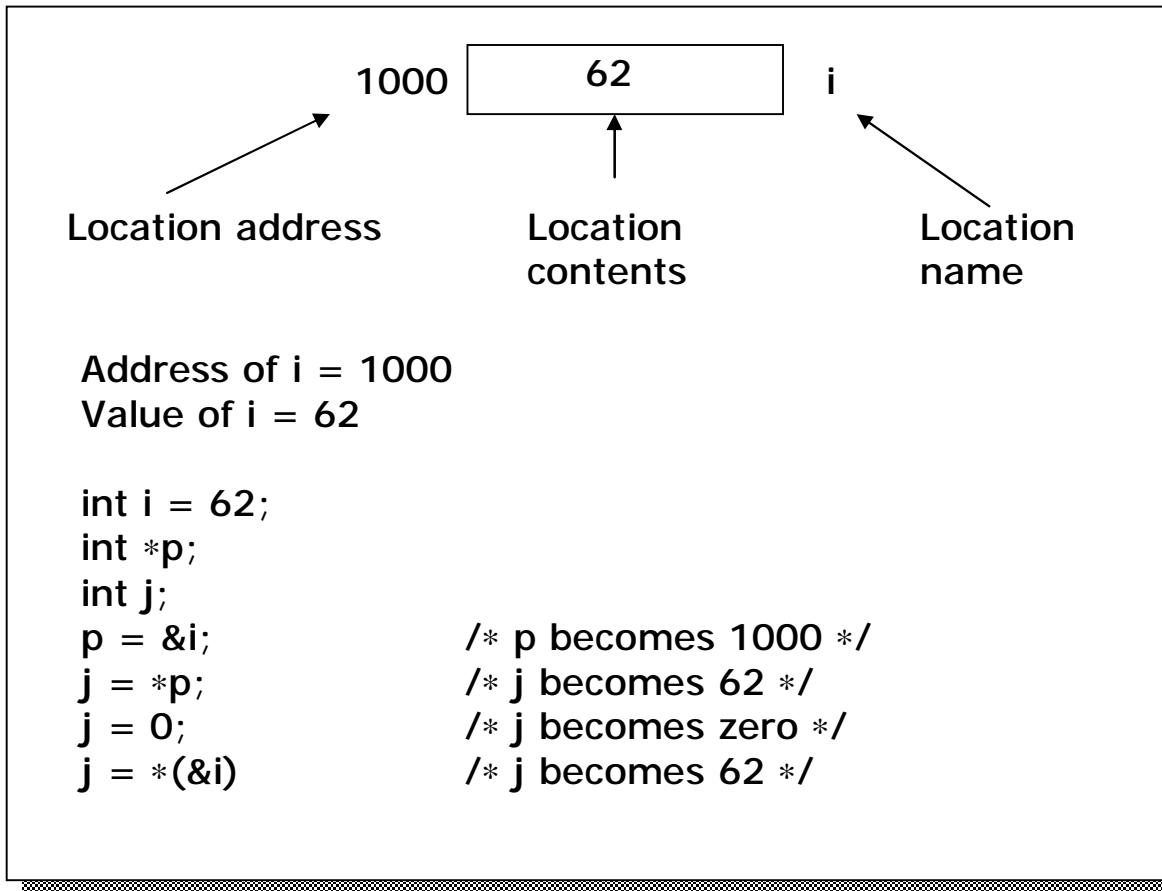
# Standard Preprocessor Directives in C

Preprocessor Directive	Meaning	Category
#	Null directive	Simple
#error <i>message</i>	Prints <i>message</i> when processed	
#line <i>linenum</i> <i>filename</i>	Used to update code line number and filename	
#pragma <i>name</i>	Compiler specific settings	
#include <i>filename</i>	Includes content of another file	File
#define <i>macro/string</i>	Define a macro or string substitution	Macro
#undef <i>macro</i>	Removes a macro definition	
#if <i>expr</i>	Includes following lines if <i>expr</i> is true	Conditional
# elif <i>expr</i>	Includes following lines if <i>expr</i> is true	
#else	Handles otherwise conditions of #if	
#endif	Closes #if or #elif block	
#ifdef <i>macro</i>	Includes following lines if macro is defined	
#ifndef <i>macro</i>	Includes following lines if macro is not defined	Operators
#	String forming operator	
##	Token pasting operator	
defined	same as #ifdef	

# Pointers

- § C pointers allow programmers to directly access memory addresses where variables are stored
- § Pointer variable is declared by adding a '\*' symbol before the variable name while declaring it.
- § If  $p$  is a pointer to a variable (e.g. int i,  $*p = i;$ )
  - § Using  $p$  means address of the storage location of the pointed variable
  - § Using  $*p$  means value stored in the storage location of the pointed variable
- § Operator '&' is used with a variable to mean variable's address, e.g.  $\&i$  gives address of variable  $i$

# Illustrating Pointers Concept



# Array

- § Collection of fixed number of elements in which all elements are of the same data type
- § Homogeneous, linear, and contiguous memory structure
- § Elements can be referred to by using their subscript or index position that is monotonic in nature
- § First element is always denoted by subscript value of 0 (zero), increasing monotonically up to one less than declared size of array
- § Before using an array, its type and dimension must be declared
- § Can also be declared as multi-dimensional such as Matrix2D[10][10]

# Illustrating Arrays Concept

1010	92		1005	Y
1008	63		1004	A
1006	82	1012	10.25	B
1004	66	1008	250.00	M
1002	84	1004	155.50	O
1000	45	1000	82.75	B
			1000	
int	marks[6];	float	price[4];	char
Each element being an int occupies 2 bytes		Each element being a float occupies 4 bytes		Each element being a char occupies 1 byte
marks[0] = 45		price[0] = 82.75		city[0] = 'B'
marks[1] = 84		price[1] = 155.50		city[1] = 'O'
.		.		.
.		.		.
.		.		.
marks[5] = 92		price[3] = 10.25		city[5] = 'Y'
(a) An array of integers having 6 elements		(b) An array of real numbers having 4 elements		(c) An array of characters having 6 elements

# String

- § One-dimensional array of characters terminated by a null character ('\0')
- § Initialized at declaration as
  - § `char name[] = "PRADEEP";`
- § Individual elements can be accessed in the same way as we access array elements such as `name[3] = 'D'`
- § Strings are used for text processing
- § C provides a rich set of string handling library functions

# Library Functions for String Handling

Library Function	Used To
strlen	Obtain the length of a string
strlwr	Convert all characters of a string to lowercase
strupr	Convert all characters of a string to uppercase
strcat	Concatenate (append) one string at the end of another
strncat	Concatenate only first n characters of a string at the end of another
strcpy	Copy a string into another
strncpy	Copy only the first n characters of a string into another
strcmp	Compare two strings
strncmp	Compare only first n characters of two strings
strcmp	Compare two strings without regard to case
strnicmp	Compare only first n characters of two strings without regard to case
strdup	Duplicate a string
strchr	Find first occurrence of a given character in a string
strrchr	Find last occurrence of a given character in a string
strstr	Find first occurrence of a given string in another string
strset	Set all characters of a string to a given character
strnset	Set first n characters of a string to a given character
strrev	Reverse a string

# User Defined Data Types (UDTs)

- § UDT is composite data type whose composition is not include in language specification
- § Programmer declares them in a program where they are used
- § Two types of UDTs are:
  - § Structure
  - § Union

# Structure

- § UDT containing a number of data types grouped together
- § Constituents data types may or may not be of different types
- § Has continuous memory allocation and its minimum size is the sum of sizes of its constituent data types
- § All elements (member variable) of a structure are publicly accessible
- § Each member variable can be accessed using “.” (dot) operator or pointer (*EmpRecord.EmpID* or *EmpRecord* ® *EmpID*)
- § Can have a pointer member variable of its own type, which is useful in creating linked list and similar data structures

# Structure (Examples)

```
struct Employee  
{  
    int EmpID;  
    char EmpName[20];  
};
```

```
Struct Employee EmpRecord;  
Struct Employee *pempRecord = &EmpRecord
```

```
struct Employee  
{  
    int EmpID;  
    char EmpName[20];  
} EmpRecord;
```

# Union

- § UDT referring to same memory location using several data types
- § Mathematical union of all constituent data types
- § Each data member begins at the same memory location
- § Minimum size of a union variable is the size of its largest constituent data types
- § Each member variable can be accessed using “.” (dot) operator
- § Section of memory can be treated as a variable of one type on one occasion, and of another type on another occasion

# Union Example

```
unionNum
{
    int intNum;
    unsigned
    unsNum'
};
union Num Number;
```

# Difference Between Structure and Union

- § Both group a number of data types together
- § Structure allocates different memory space contiguously to different data types in the group
- § Union allocates the same memory space to different data types in the group

# Control Structures

- § *Control structures* (branch statements) are decision points that control the flow of program execution based on:
  - § Some condition test (conditional branch)
  - § Without condition test (unconditional branch)
- § Ensure execution of other statement/block or cause skipping of some statement/block

# Conditional Branch Statements

- § **if** is used to implement simple one-way test. It can be in one of the following forms:
  - § if..stmt
  - § if..stmt1..else..stmt2
  - § if..stmt1..else..if..stmtn
- § **switch** facilitates multi-way condition test and is very similar to the third *if* construct when primary test object remains same across all condition tests

# Examples of “if” Construct

§ if (*i* <= 0)  
    *i*++;

§ if (*i* <= 0)  
    *i*++;  
else  
    *j*++;

§ if (*i* <= 0)  
    *i*++;  
else if (*i* >= 0)  
    *j*++;  
else  
    *k*++;

# Example of “switch” Construct

```
switch(ch)
{
    case 'A':
    case 'B':
    case 'C':
        printf("Capital");
        break;
    case 'a':
    case 'b':
    case 'c':
        printf("Small");
        break;
    default:
        printf("Not cap or small");
}
```

Same thing can be written also using *if* construct as:

```
if (ch == 'A' || ch == 'B' || ch ==
'C')
    printf("Capital");
else if (ch == 'a' || ch == 'b' || ch ==
'c')
    printf("Small");
else
    printf("Not cap or small");
```

# Unconditional Branch Statements

- § Break: Causes unconditional exit from *for*, *while*, *do*, or *switch* constructs. Control is transferred to the statement immediately outside the block in which *break* appears.
- § Continue: Causes unconditional transfer to next iteration in a *for*, *while*, or *do* construct. Control is transferred to the statement beginning the block in which *continue* appears.
- § Goto label: Causes unconditional transfer to statement marked with the label within the function.

(Continued on next slide)

# Unconditional Branch Statements

(Continued from previous slide)

- § **Return [value/variable]:** Causes immediate termination of function in which it appears and transfers control to the statement that called the function. Optionally, it provides a value compatible to the function's return data type.

# Loop Structures

- § Loop statements are used to repeat the execution of statement or blocks
- § Two types of loop structures are:
  - § Pretest: Condition is tested before each iteration to check if loop should occur
  - § Posttest: Condition is tested after each iteration to check if loop should continue (at least, a single iteration occurs)

# Pretest Loop Structures

- § **for:** It has three parts:
  - § *Initializer* is executed at start of loop
  - § *Loop condition* is tested before iteration to decide whether to continue or terminate the loop
  - § *Incrementor* is executed at the end of each iteration
- § **While:** It has a *loop condition* only that is tested before each iteration to decide whether to continue to terminate the loop

# Examples of “for” and “while” Constructs

§    **for (i=0; i < 10; i++)**  
      **printf("i = %d", i);**

§    **while (i < 10)**  
      {  
        **printf("i = %d", i);**  
        *i++;*  
      }

# Posttest Loop Construct “do...while”

- § It has a loop condition only that is tested after each iteration to decide whether to continue with next iteration or terminate the loop
- § Example of *do...while* is:

```
do {  
    printf("i = %d", i);  
    i++;  
} while (i < 10) ;
```

# Functions

- § Functions (or subprograms) are building blocks of a program
- § All functions must be declared and defined before use
- § Function declaration requires *functionname*, *argument list*, and *return type*
- § Function definition requires coding the body or logic of function
- § Every C program must have a *main* function. It is the entry point of the program

# Example of a Function

```
int myfunc ( int Val, int ModVal )  
{  
    unsigned temp;  
    temp = Val % ModVal;  
    return temp;  
}
```

This function can be called from any other place using simple statement:

```
int n = myfunc(4, 2);
```

# Sample C Program (Program-1)

```
/* Program to accept an integer from console and to display  
   whether the number is even or odd */  
  
# include <stdio.h>  
void main()  
{  
    int number, remainder;  
    clrscr(); /* clears the console screen */  
    printf ("Enter an integer: ");  
    scanf ("%d", &number);  
    remainder = number % 2;  
    if (remainder == 0)  
        printf ("\n The given number is even");  
    else  
        printf ("\n The given number is odd");  
  
    getch();  
}
```

# Sample C Program (Program-2)

```
/* Program to accept an integer in the range 1 to 7 (both inclusive) from  
console and to display the corresponding day (Monday for 1, Tuesday for  
2, Wednesday for 3, and so on). If the entered number is out of range,  
the program displays a message saying that */
```

```
# include <stdio.h>  
# include <conio.h>  
  
#define MON printf ("\n Entered number is 1 hence day is MONDAY");  
#define TUE printf ("\n Entered number is 2 hence day is TUESDAY");  
#define WED printf ("\n Entered number is 3 hence day is WEDNESDAY");  
#define THU printf ("\n Entered number is 4 hence day is THURSDAY");  
#define FRI printf ("\n Entered number is 5 hence day is FRIDAY");  
#define SAT printf ("\n Entered number is 6 hence day is SATURDAY");  
#define SUN printf ("\n Entered number is 7 hence day is SUNDAY");  
#define OTH printf ("\n Entered number is out of range");  
  
void main()  
{  
    int day;  
    clrscr();  
    printf ("Enter an integer in the range 1 to 7");  
    scanf ("%d", &day);  
    switch(day)
```

(Continued on next slide)

# Sample C Program (Program-2)

(Continued from previous slide..)

```
{  
    Case 1:  
        MON;  
        break;  
    Case 2:  
        TUE;  
        break;  
    Case 3:  
        WED;  
        break;  
    Case 4:  
        THU;  
        break;  
    Case 5:  
        FRI;  
        break;  
    Case 6:  
        SAT;  
        break;  
    Case 7:  
        SUN;  
        break;  
    default:  
        OTH;  
}  
getch();  
}
```

# Sample C Program (Program-3)

```
/* Program to accept the radius of a circle from console and to calculate  
and display its area and circumference */  
  
# include <stdio.h>  
# include <conio.h>  
# define PI 3.1415  
  
void main()  
{  
    float radius, area, circum;  
    clrscr();  
    printf ("Enter the radius of the circle: ");  
    scanf ("%f", &radius);  
    area = PI * radius * radius;  
    circum = 2 * PI * radius;  
    printf ("\n Area and circumference of the circle are %f  
           and %f respectively", area, circum);  
    getch();  
}
```

(Continued on next slide)

# Sample C Program (Program-4)

```
/* Program to accept a string from console and to display the number of
vowels in the string */

#include <stdio.h>
#include <conio.h>
#include <string.h>

void main()
{
    char input_string[50]; /* maximum 50 characters */
    int len;
    int i = 0, cnt = 0;
    clrscr();
    printf ("Enter a string of less than 50 characters: \n");
    gets (input_string);
    len = strlen (input_string);
    for (i = 0; i < len; i++)
    {
        switch (input_string[i])
```

(Continued on next slide)

# Sample C Program (Program-4)

```
{  
    case 'a':  
    case 'e':  
    case 'i':  
    case 'o':  
    case 'u':  
    case 'A':  
    case 'E':  
    case 'I':  
    case 'O':  
    case 'U':  
        cnt++  
    }  
}  
printf ("\n Number of vowels in the string are: %d", cnt);  
getch();  
}
```

# Sample C Program (Program-5)

```
/* Program to illustrate use of a user defined function. The program initializes an array of n elements from 0 to n-1 and then calculates and prints the sum of the array elements. In this example n = 10 */
```

```
#include <stdio.h>
#define SIZE 10

int ArrSum(int *p, int n);
{
    int s, tot = 0;
    for(s = 0; s < n; s++)
    {
        tot += *p;
        p++;
    }
    return tot;
}
int main()
{
    int i = 0, sum = 0;
    int nArr[SIZE] = {0};
    while(i < SIZE)
    {
        nArr[i] = i;
        i++
    }
    sum = ArrSum(nArr, SIZE);
    printf("Sum of 0 to 9 = %d\n", sum);
    return 0;
}
```

# Key Words/Phrases

- § Arithmetic operators
- § Arrays
- § Assignment operators
- § Bit-level manipulation
- § Bitwise operators
- § Branch statement
- § Character set
- § Comment statement
- § Compound statement
- § Conditional branch
- § Conditional compilation
- § Constants
- § Control structures
- § Format specifiers
- § Formatted I/O
- § Function
- § Keywords
- § Library functions
- § Logical operators
- § Loop structures
- § Macro expansion
- § Main function
- § Member element
- § Null statement
- § Operator associativity
- § Operator precedence
- § Pointer
- § Posttest loop
- § Preprocessor directives
- § Pretest loop
- § Primitive data types
- § Reserved words
- § Simple statement
- § Statement block
- § Strings
- § Structure data type
- § Unconditional branch
- § Union data type
- § User-defined data types
- § Variable name
- § Variable type declaration
- § Variables