

# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 1

**Introduction**



# 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

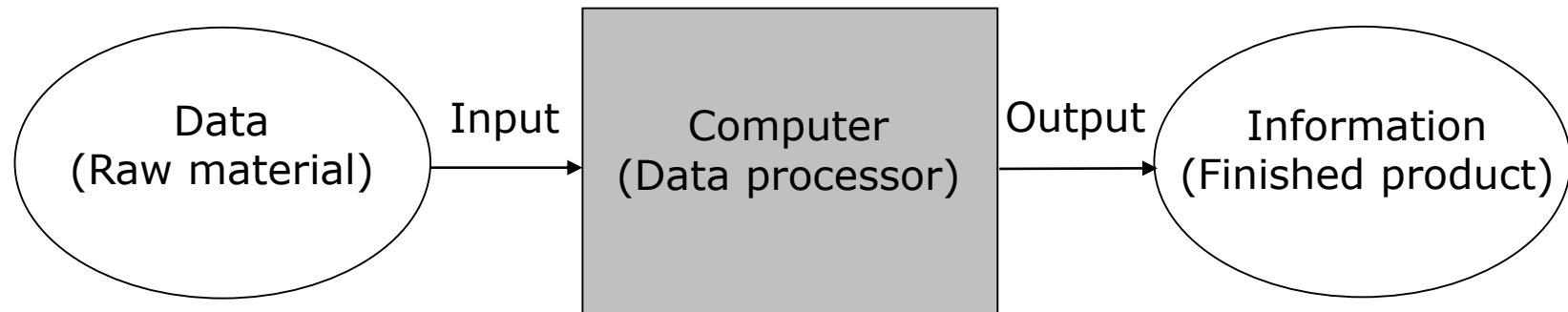


- The word computer comes from the word “compute”, which means, “to calculate”
- Thereby, a computer is an electronic device that can perform arithmetic operations at high speed
- A computer is also called a *data processor* because it can store, process, and retrieve data whenever desired

# Data Processing



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

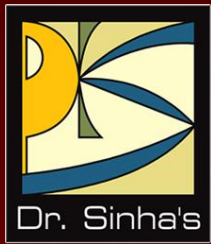


*Data* is raw material used as input to data processing and *information* is processed data obtained as output

# Characteristics of Computers



Sr. No.	Characteristics	Description
1	Automatic	It carries out a job normally without any human intervention
2	Speed	It can perform several billion ( $10^9$ ) simple arithmetic operations per second
3	Accuracy	It performs every calculation with the same accuracy
4	Diligence	It is free from monotony, tiredness, and lack of concentration
5	Versatility	It can perform a wide variety of tasks
6	Memory	It can store huge amount of information and can recall any piece of this information whenever required
7	No I. Q.	It cannot take its own decisions, and has to be instructed what to do and in what sequence
8	No Feelings	It cannot make judgments based on feelings and instincts



# Evolution of Computers



# 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

(Continued on next slide)

# Evolution of Computers



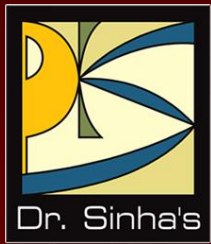
- *Charles Babbage* is considered to be the father of modern digital computers
  - He designed “Difference Engine” in 1822
  - He designed a *fully automatic analytical engine* in 1842 for performing basic arithmetic functions
  - His efforts established a number of principles that are fundamental to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



# Some Well Known Early Computers



- The Mark I Computer (1937-44)
- The Atanasoff-Berry Computer (1939-42)
- The Electronic Numerical Integrator And Calculator (ENIAC) (1943-46)
- The Electronic Discrete Variable Automatic Computer (EDVAC) (1946-52)
- The Electronic Delay Storage Automatic Calculator (EDSAC) (1947-49)
- Manchester Mark I (1948)
- The Universal Automatic Computer (UNIVAC) I (1951)
- IBM 701 (1952)
- IBM 650 (1953)



# Computer Generations



# 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



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 commercial production difficult and costly</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 core memory</li> <li>▪ Magnetic tapes</li> <li>▪ Disks 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>

# Computer Generations



Generation (Period)	Key hardware technologies	Key software technologies	Key Characteristics	Some representative systems
Third (1964-1975)	<ul style="list-style-type: none"> <li>▪ ICs with SSI and MSI technologies</li> <li>▪ Larger magnetic core memory</li> <li>▪ Larger capacity magnetic disks and tapes secondary storage</li> <li>▪ Minicomputers</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



Generation (Period)	Key hardware technologies	Key software technologies	Key Characteristics	Some representative 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>▪ 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>▪ Multiprocessor operating systems and concurrent programming languages</li> <li>▪ UNIX operating system</li> <li>▪ C and C++ programming languages</li> <li>▪ PC-based applications; network-based applications</li> <li>▪ Object-oriented software design</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</li> <li>▪ General purpose machines</li> <li>▪ Easier to produce commercially</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)*

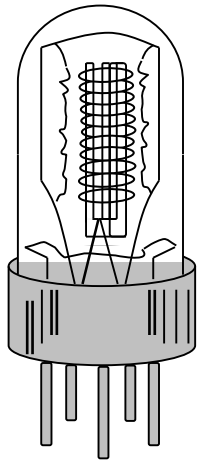
# Computer Generations



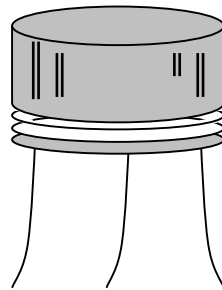
Generation (Period)	Key hardware technologies	Key software technologies	Key Characteristics	Some representative systems
Fifth (1989-Present)	<ul style="list-style-type: none"> <li>▪ ICs with ULSI technology</li> <li>▪ Multicore processor chips</li> <li>▪ Larger capacity main memory and hard disks</li> <li>▪ Optical disks as portable read-only storage media</li> <li>▪ Solid state disks</li> <li>▪ Notebook computers</li> <li>▪ Powerful desktop PCs and workstations</li> <li>▪ Very powerful mainframes</li> <li>▪ Supercomputers based on parallel processing</li> <li>▪ Internet</li> </ul>	<ul style="list-style-type: none"> <li>▪ World Wide Web</li> <li>▪ Multimedia, Internet-based applications</li> <li>▪ Microkernel, multithreading, multicore OS</li> <li>▪ JAVA, Python and other programming languages</li> <li>▪ MPI and PVM libraries for parallel programming</li> </ul>	<ul style="list-style-type: none"> <li>▪ Portable computers</li> <li>▪ Hand-held mobile smart devices</li> <li>▪ More powerful, cheaper, reliable, and easier to use desktop machines</li> <li>▪ Very powerful mainframes</li> <li>▪ Very high uptime due to hot-pluggable components</li> <li>▪ General purpose machines</li> <li>▪ Easier to produce commercially</li> </ul>	<ul style="list-style-type: none"> <li>▪ iPhone</li> <li>▪ iPad</li> <li>▪ IBM notebooks</li> <li>▪ Pentium PCs</li> <li>▪ Windows PC</li> <li>▪ Apple PC</li> <li>▪ SUN Workstations</li> <li>▪ IBM SP/2</li> <li>▪ SGI Origin 2000</li> <li>▪ PARAM supercomputers</li> </ul>

*(Continued on next slide)*

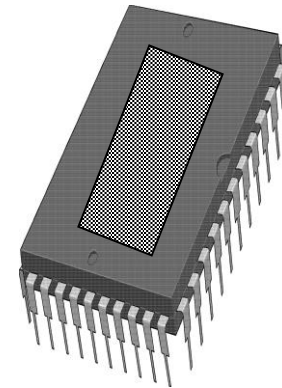
# Electronic Devices Used in Computers of Different Generations



(a) A Vacuum tube



(b) A Transistor



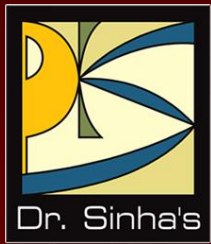
(c) An IC chip



# Key Words/Phrases



- Computer
- Computer generations
- Computer Supported Cooperative Working (CSCW)
- Data
- Data processing
- Data processor
- First-generation computers
- Second-generation computers
- Third-generation computers
- Fourth-generation computers
- Fifth-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



# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 2

## Basic Computer Organization



# 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



- 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

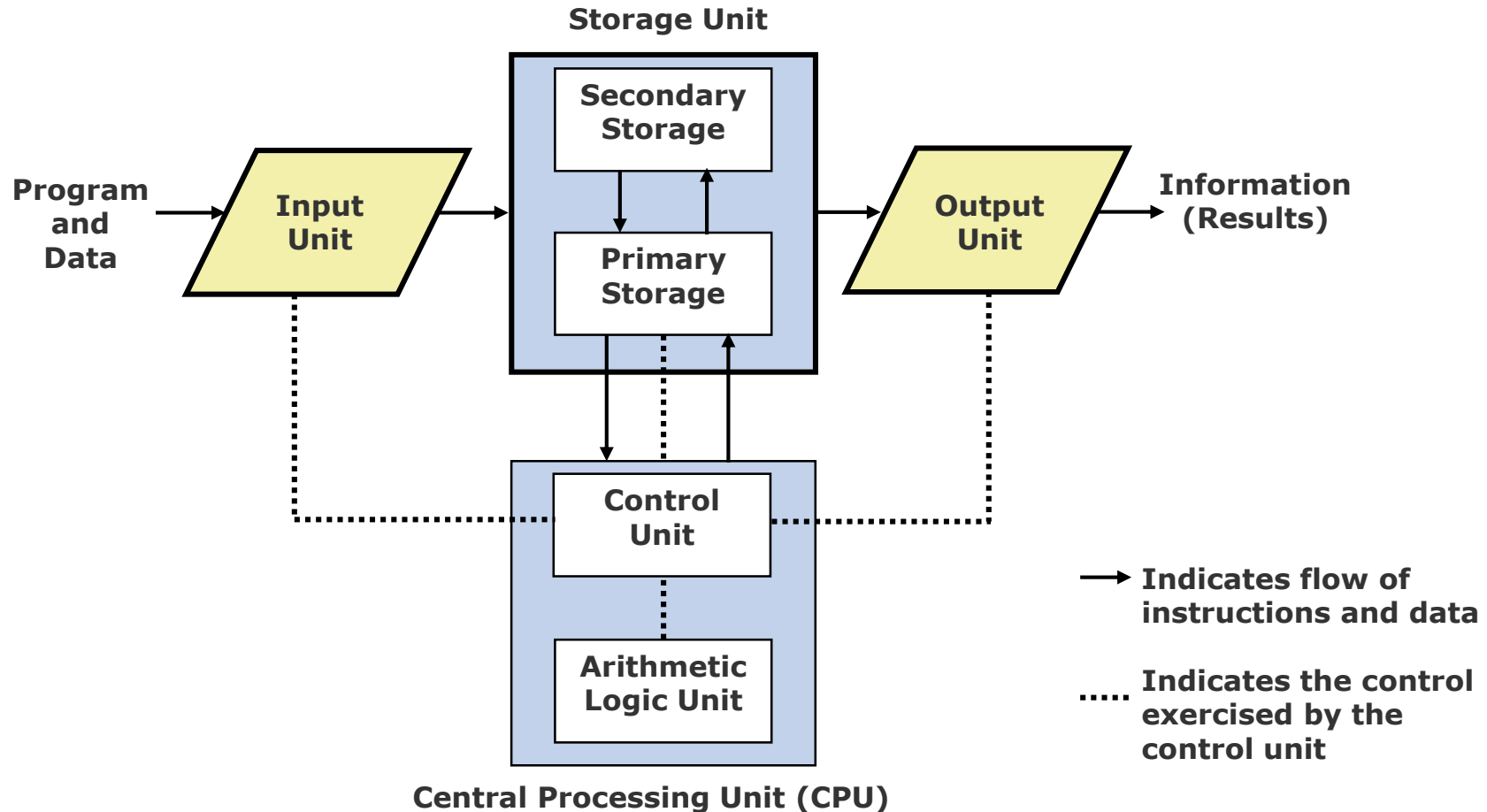
*(Continued on next slide)*

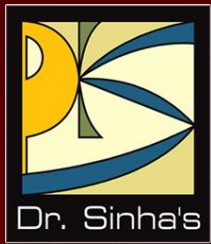
# The Five Basic Operations of a Computer System



- **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





# Main Units and Their Functions





# 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

# Types of Storage



**The broad categories of storage are:**

1. Primary storage
2. Secondary 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)*

# 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

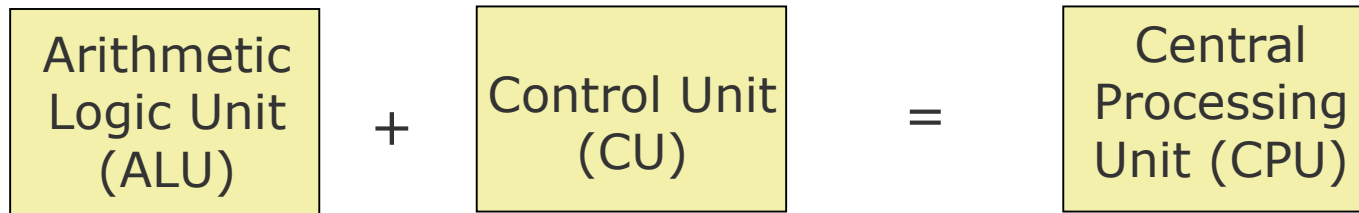
# Control Unit (CU)



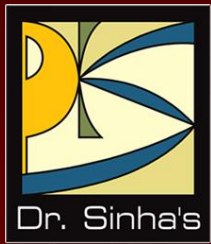
Control Unit of a computer system manages and coordinates the operations of all other components of the computer system



# 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



# The System Concept



**A system has following three characteristics:**

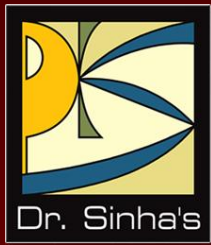
1. A system has more than one element
2. All elements of a system are logically related
3. All elements of a system are controlled in a manner to achieve the system goal

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
- Primate storage
- Processing
- Secondary storage
- Storage unit
- Storing
- System



# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 3

**Number Systems**



# 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



- 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

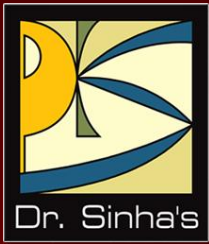
# Number Systems



**Two types of number systems are:**

- Non-positional number systems
- Positional number systems





# Basics and Few Popular 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



- 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



## Example

$$\begin{aligned} 2586_{10} &= (2 \times 10^3) + (5 \times 10^2) + (8 \times 10^1) + (6 \times 10^0) \\ &= 2000 + 500 + 80 + 6 \end{aligned}$$

# 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



## 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 **b**inary 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



- 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)*

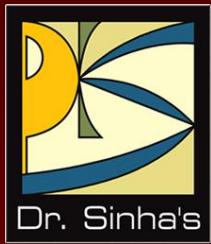
# Hexadecimal Number System



- 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 from One Number System to Another



# 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



## Example

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

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

Column 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



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



## Example

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

## Solution:

8	952	Remainders
	119	0
	14	7
	1	6
	0	1

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

# Converting from a Base Other Than 10 to Another Base Other Than 10



## 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 from a Base Other Than 10 to Another Base Other Than 10



## 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 from a Base Other Than 10 to Another Base Other Than 10



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



## Example

$$1101010_2 = ?_8$$

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

$$\underline{001} \quad \underline{101} \quad \underline{010}$$

Step 2: Convert each group into one octal digit

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

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

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

$$\text{Hence, } 1101010_2 = 152_8$$

# 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



## 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

$$562_8 = \begin{array}{ccc} \underline{101} & \underline{110} & \underline{010} \\ 5 & 6 & 2 \end{array}$$

$$\text{Hence, } 562_8 = 101110010_2$$

# 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



## 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

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

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10} = D_{16}$$

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



## 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

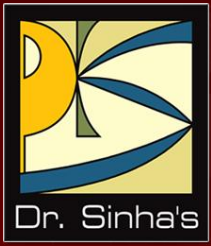


Step 2: Combine the binary groups

$$2AB_{16} = \begin{array}{ccc} \underline{0010} & \underline{1010} & \underline{1011} \\ 2 & A & B \end{array}$$

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





# Fractional Numbers



# 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)



	Binary Point										
						↓					
Position	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		$1/2$	$1/4$	$1/8$	$1/16$	

*(Continued on next slide)*

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



## 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)



	Octal Point							
	↓							
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		$1/8$	$1/64$	$1/512$

*(Continued on next slide)*

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



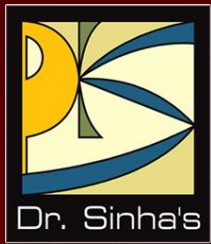
## Example

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

# 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



# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 4

**Computer Codes**



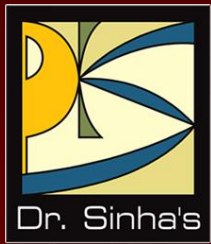


# Learning Objectives



## 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 and Their Binary Representation



# 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



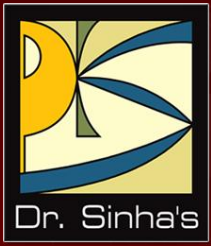
- Computer codes are used for internal representation of data in computers
- As computers use binary numbers for internal data representation, computer codes use binary coding schemes
- In binary coding, every symbol that appears in the data is represented by a group of bits
- The group of bits used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

*(Continued on next slide)*

# Computer Codes



- 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



- BCD stands for **B**inary **C**oded **D**ecimal
- 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



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	0000	00

# 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

<u>110010</u>	<u>110001</u>	<u>010010</u>	<u>110101</u>
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

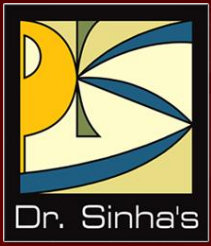
G = 67 in BCD octal notation

I = 71 in BCD octal notation

T = 23 in BCD octal notation

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

<u>64</u>	<u>71</u>	<u>67</u>	<u>71</u>	<u>23</u>
D	I	G	I	T



# EBCDIC



# EBCDIC



- EBCDIC stands for **E**xtended **B**inary **C**oded **D**ecimal **I**nterchange **C**ode
- 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



Character	EBCDIC Code		Hexadecimal 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



<b>Numeric Value</b>	<b>EBCDIC</b>	<b>Sign Indicator</b>
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



<b>Numeric Value</b>	<b>EBCDIC</b>	<b>Sign Indicator</b>
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

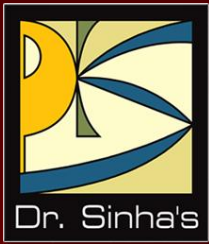
I = 1100 1001 in EBCDIC binary notation

T = 1110 0011 in EBCDIC binary notation

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

<u>11000010</u>	<u>11001001</u>	<u>11100011</u>
B	I	T

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



# ASCII



# 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



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



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

<u>1000010</u>	<u>1001111</u>	<u>1011001</u>
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

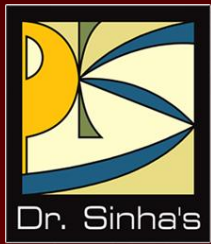
K = 01001011 in ASCII-8 binary notation

Y = 01011001 in ASCII-8 binary notation

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

<u>01010011</u>	<u>01001011</u>	<u>01011001</u>
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



# Unicode

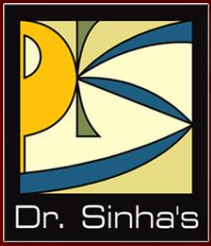


- **Why Unicode:**
  - No single encoding system supports all languages
  - Different encoding systems conflict
  
- **Unicode features:**
  - Provides a consistent way of encoding multilingual plain text
  - Defines codes for characters used in all major languages of the world
  - Defines codes for 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



- 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  
( $A < B < C \dots < Z$ )
  2. Digits are considered in numeric order  
( $0 < 1 < 2 \dots < 9$ )



# 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:

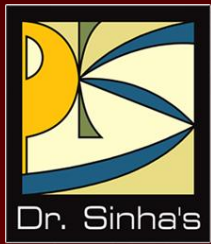
$$1A < 23 < 2a < A1 < Aa < a2 < aA$$

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



# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 5

**Computer  
Arithmetic**

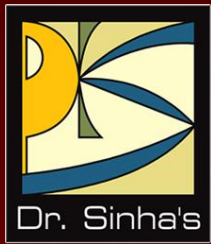


# Learning Objectives



## 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 (/)



# Why Binary?







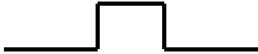

# Binary over Decimal



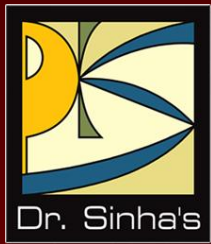
- Information is handled in a computer by electronic/electrical components
- Electronic components operate in binary mode (can only indicate two states – on (1) or off (0))
- Binary number system has only two digits (0 and 1), and is suitable for expressing two possible states
- In binary system, computer 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



- 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:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 0 \text{ plus a carry of } 1 \text{ to next higher column}$$

# Binary Addition (Example 1)



## Example

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

## Solution

Binary	Decimal
carry 11 10011 +1001 <hr/> 11100 <hr/>	carry 1 19 +9 <hr/> 28 <hr/>

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

	<b>Binary</b>	<b>Decimal</b>
carry	11111	carry 1
	100111	39
	+11011	+27
	<hr/>	<hr/>
	1000010	66
	<hr/>	<hr/>

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:

$$0 - 0 = 0$$

$$0 - 1 = 1 \text{ with a borrow from the next column}$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

# Binary Subtraction (Example)



## Example

Subtract  $01110_2$  from  $10101_2$

## Solution

$$\begin{array}{r}
 \left\{ \begin{array}{l} 12 \\ 0202 \end{array} \right. \\
 10101 \\
 -01110 \\
 \hline
 00111 \\
 \hline
 \end{array}$$

Note: Go through explanation given in the book

# Complement of a Number



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

The diagram illustrates the formula for the complement of a number. It shows the equation  $C = B^n - 1 - N$  with arrows pointing from descriptive text to each term:
 

- An arrow points from "Complement of the number" to  $C$ .
- An arrow points from "Base of the number" to  $B^n$ .
- An arrow points from "The number" to  $N$ .
- An arrow points from "Number of digits in the number" to the exponent  $n$  in  $B^n$ .



# 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

$$\begin{aligned} \text{Step 1: Complement of } 56_{10} \\ = 10^2 - 1 - 56 = 99 - 56 = 43_{10} \end{aligned}$$

$$\begin{aligned} \text{Step 2: } 92 + 43 \text{ (complement of 56)} \\ = 135 \text{ (note 1 as carry)} \end{aligned}$$

$$\text{Step 3: } 35 + 1 \text{ (add 1 carry to sum)}$$

$$\text{Result} = 36$$

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:

$$\begin{array}{r} 18 \\ + 64 \text{ (complement} \\ \hline \phantom{+ 64} \text{of } 35) \\ \hline 82 \\ \hline \end{array}$$

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

$$\begin{aligned} \text{Result} &= -(99 - 82) \\ &= -17 \end{aligned}$$

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

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

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

└───┬───> 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

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

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

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



# 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
<hr/>	
1010	Partial Product
0000	Partial Product
0000	Partial Product
1010	Partial Product
<hr/>	
1011010	Final Product
<hr/>	

*(Continued on next slide)*

# Binary Multiplication (Example 2)



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,

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

# Binary Division



Table for binary division is as follows:

$0 \div 0 =$  Divide by zero error

$0 \div 1 = 0$

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

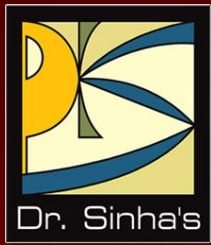
Total subtractions = 6

Remainder = 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



# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 6

**Boolean Algebra  
and Logic Circuits**

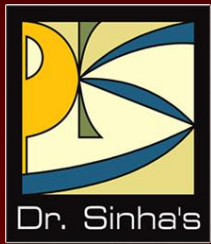


# Learning Objectives



## 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 their design



# Boolean Algebra



# 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**
  - Variables used in Boolean equations can have either of two possible values, 0 or 1
- **Logical addition**
  - Symbol '+', also known as 'OR' operator, is used for logical addition. It follows law of binary addition
- **Logical multiplication**
  - Symbol '.', also known as 'AND' operator, is used for logical multiplication. It follows law of binary multiplication
- **Complementation**
  - Symbol '-', also known as 'NOT' operator, is used for complementation. It follows law of binary complement

# Truth Tables for Boolean Operators



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

Truth Table for OR (+)

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

Truth Table for AND (.)

Input	Output
A	$\bar{A}$
0	1
1	0

Truth Table for NOT (-)



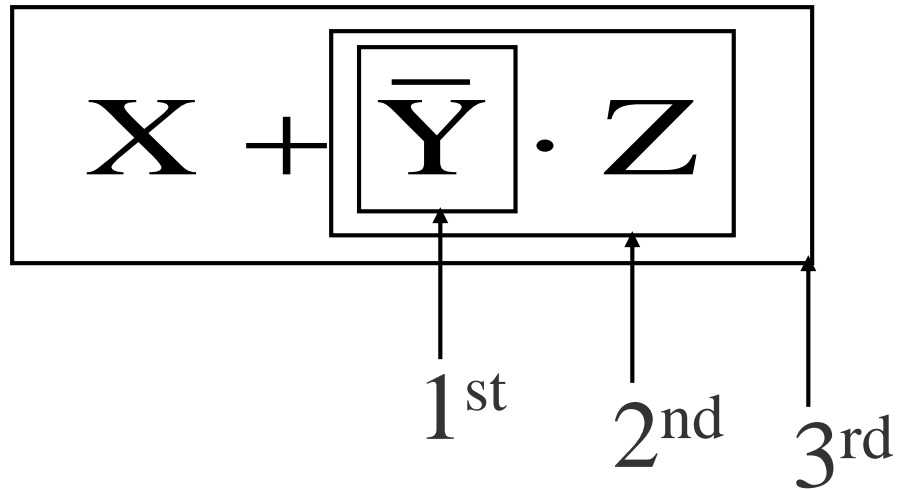
# Operator Precedence



- 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



# 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



## ***Postulate 4: Associative Law***

$$(a) \quad x + (y + z) = (x + y) + z$$

$$(b) \quad x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

## ***Postulate 5: Distributive Law***

$$(a) \quad x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$

$$(b) \quad x + (y \cdot z) = (x + y) \cdot (x + z)$$

## ***Postulate 6:***

$$(a) \quad x + \bar{x} = 1$$

$$(b) \quad x \cdot \bar{x} = 0$$

# The Principle of Duality



There is a precise duality between the operators  $\cdot$  (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 ' $\cdot$ ' 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	$\overline{x+y} = \overline{x} \cdot \overline{y}$	$\overline{x \cdot y} = \overline{x} + \overline{y}$	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.

$$= x + x \cdot y$$

$$= x \cdot 1 + x \cdot y$$

$$= x \cdot (1 + y)$$

$$= x \cdot (y + 1)$$

$$= x \cdot 1$$

$$= x$$

$$= \text{R.H.S.}$$

by postulate 2(b)

by postulate 5(a)

by postulate 3(a)

by theorem 2(a)

by postulate 2(b)



# Proving a Theorem by Perfect Induction (Example)



**Theorem:**

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

<b>x</b>	<b>y</b>	<b>x · y</b>	<b>x + x · y</b>
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.

$$= x + x$$

$$= (x + x) \cdot 1 \quad \text{by postulate 2(b)}$$

$$= (x + x) \cdot (x + \bar{x}) \quad \text{by postulate 6(a)}$$

$$= x + x \cdot \bar{x} \quad \text{by postulate 5(b)}$$

$$= x + 0 \quad \text{by postulate 6(b)}$$

$$= x \quad \text{by postulate 2(a)}$$

$$= \text{R.H.S.}$$

*(Continued on next slide)*

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



## ***Dual Theorem:***

$$X \cdot X = X$$

## ***Proof:***

L.H.S.

$$= X \cdot X$$

$$= X \cdot X + 0 \quad \text{by postulate 2(a)}$$

$$= X \cdot X + X \cdot \bar{X} \quad \text{by postulate 6(b)}$$

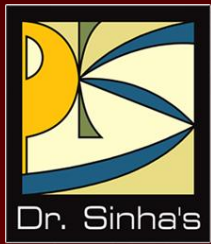
$$= X \cdot (X + \bar{X}) \quad \text{by postulate 5(a)}$$

$$= X \cdot 1 \quad \text{by postulate 6(a)}$$

$$= X \quad \text{by postulate 2(b)}$$

$$= \text{R.H.S.}$$

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



# 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

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

*(Continued on next slide)*

# Representation as a Truth Table



- 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



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

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

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

$F_2$  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



<b>x</b>	<b>y</b>	<b>z</b>	<b>F<sub>1</sub></b>	<b>F<sub>2</sub></b>
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  $F_1$  and  $F_2$  produce the same result

# Try out some Boolean Function Minimization



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

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

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

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

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

# 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  $\overline{F_1}$ , 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

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

# 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	$\bar{x} \cdot \bar{y} \cdot \bar{z}$	$m_0$	$x + y + z$	$M_0$
0	0	1	$\bar{x} \cdot \bar{y} \cdot z$	$m_1$	$x + y + \bar{z}$	$M_1$
0	1	0	$\bar{x} \cdot y \cdot \bar{z}$	$m_2$	$x + \bar{y} + z$	$M_2$
0	1	1	$\bar{x} \cdot y \cdot z$	$m_3$	$x + \bar{y} + \bar{z}$	$M_3$
1	0	0	$x \cdot \bar{y} \cdot \bar{z}$	$m_4$	$\bar{x} + y + z$	$M_4$
1	0	1	$x \cdot \bar{y} \cdot z$	$m_5$	$\bar{x} + y + \bar{z}$	$M_5$
1	1	0	$x \cdot y \cdot \bar{z}$	$m_6$	$\bar{x} + \bar{y} + z$	$M_6$
1	1	1	$x \cdot y \cdot z$	$m_7$	$\bar{x} + \bar{y} + \bar{z}$	$M_7$

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 <sub>1</sub>
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 in case of F<sub>1</sub> :

001, 100, and 111

*(Continued on next slide)*

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



- 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

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

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

# 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 <sub>1</sub>
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 in case of F<sub>1</sub> :  
 000, 010, 011, 101, and 110

*(Continued on next slide)*

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



- Their corresponding maxterms are:

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

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

- Taking the AND of these maxterms, we get:

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

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

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



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

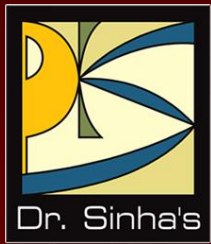


To convert from one canonical form to another, interchange the symbol and list those numbers missing from the original form.

## 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



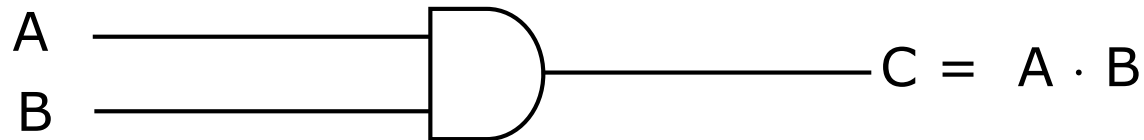
- Logic gates are electronic circuits that operate on one or more input signals to produce standard output signal
- Are the building blocks of all the circuits in a computer
- Some of the most basic and useful logic gates are AND, OR, NOT, NAND and NOR gates

# 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)



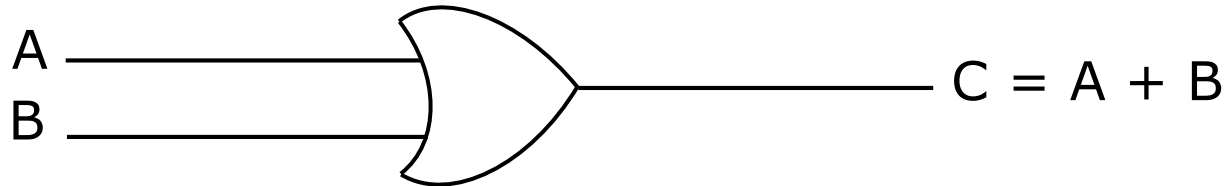
Inputs		Output
A	B	$C = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

# 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

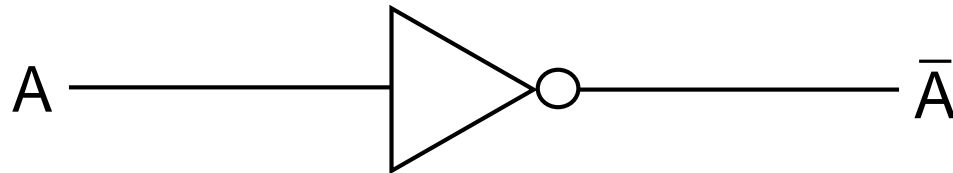
# NOT Gate



- 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)



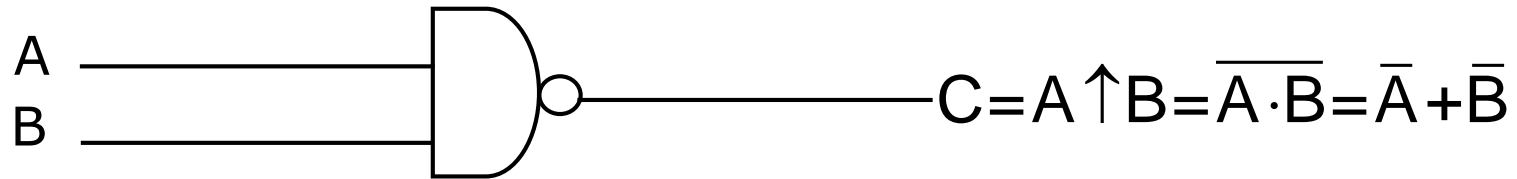
Input	Output
A	$\bar{A}$
0	1
1	0

# 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)



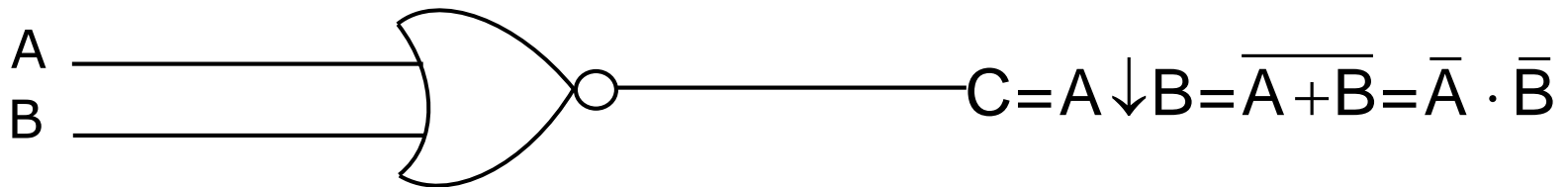
Inputs		Output
A	B	$C = \overline{A} + \overline{B}$
0	0	1
0	1	1
1	0	1
1	1	0

# NOR Gate

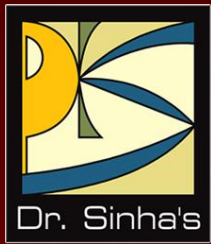


- Complemented OR gate
- Generates an output signal of:
  - 1 only when all inputs are 0
  - 0 if any one of inputs is a 1

# 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 Cicruits

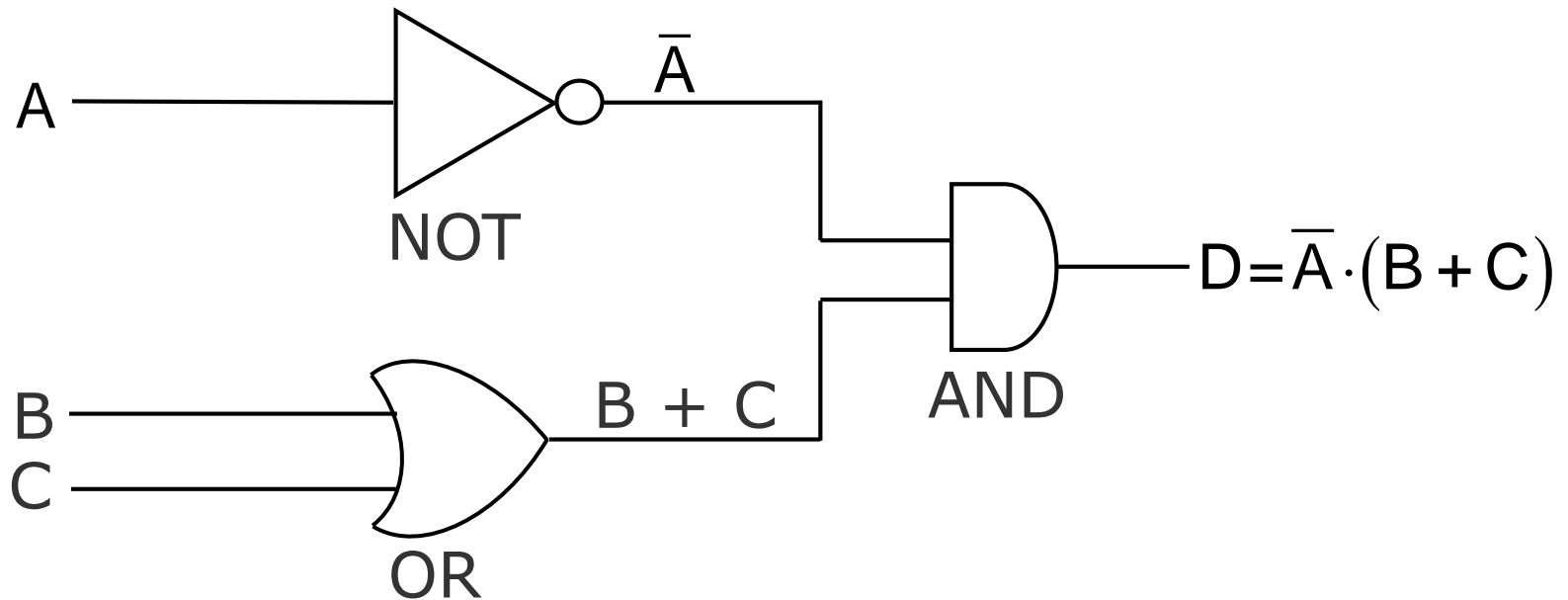


# Logic Circuits



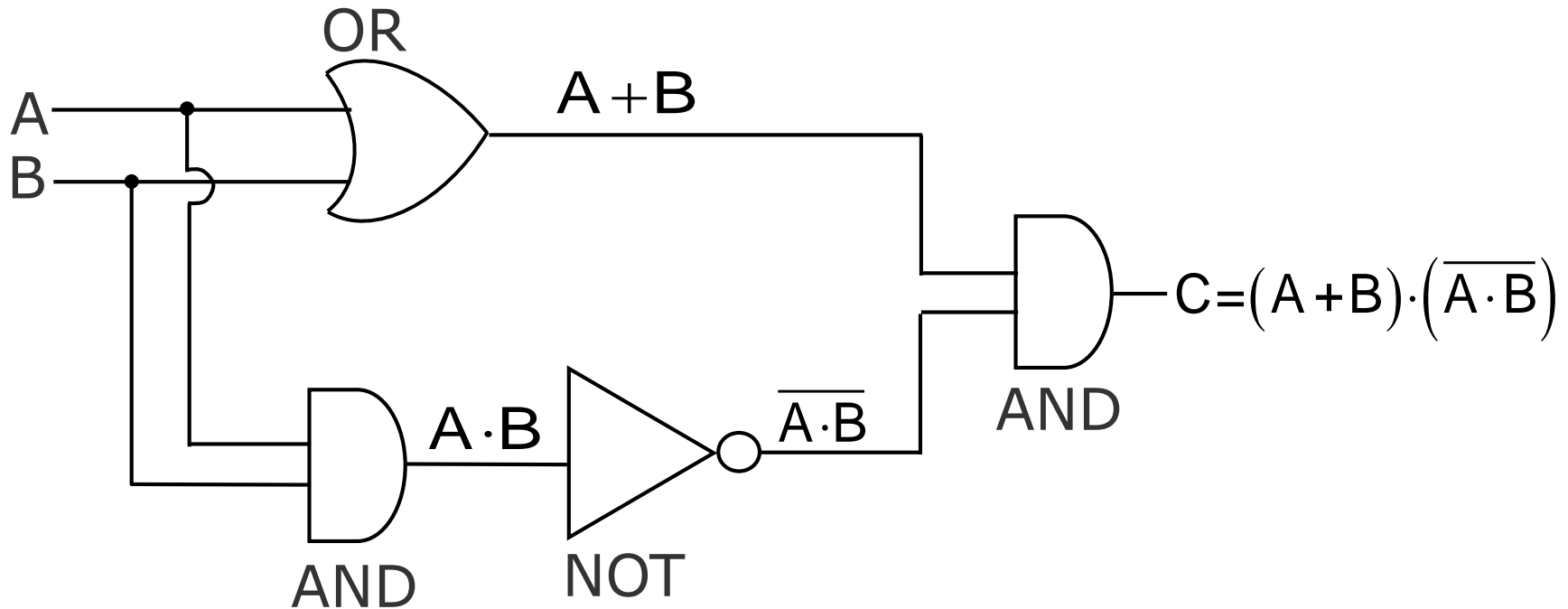
- When logic gates are interconnected to form a gating / logic network, it is known as a *combinational logic circuit*
- The Boolean algebra expression for a given logic circuit can be derived by systematically progressing from input to output on the gates
- The three logic gates (AND, OR, and NOT) are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





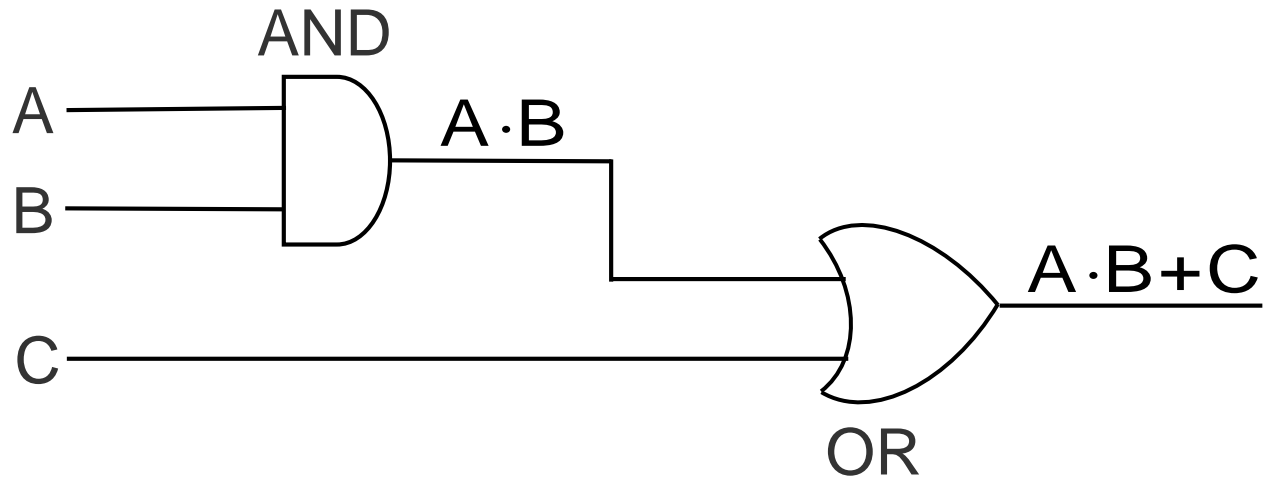
# Finding Boolean Expression of a Logic Circuit (Example 2)



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



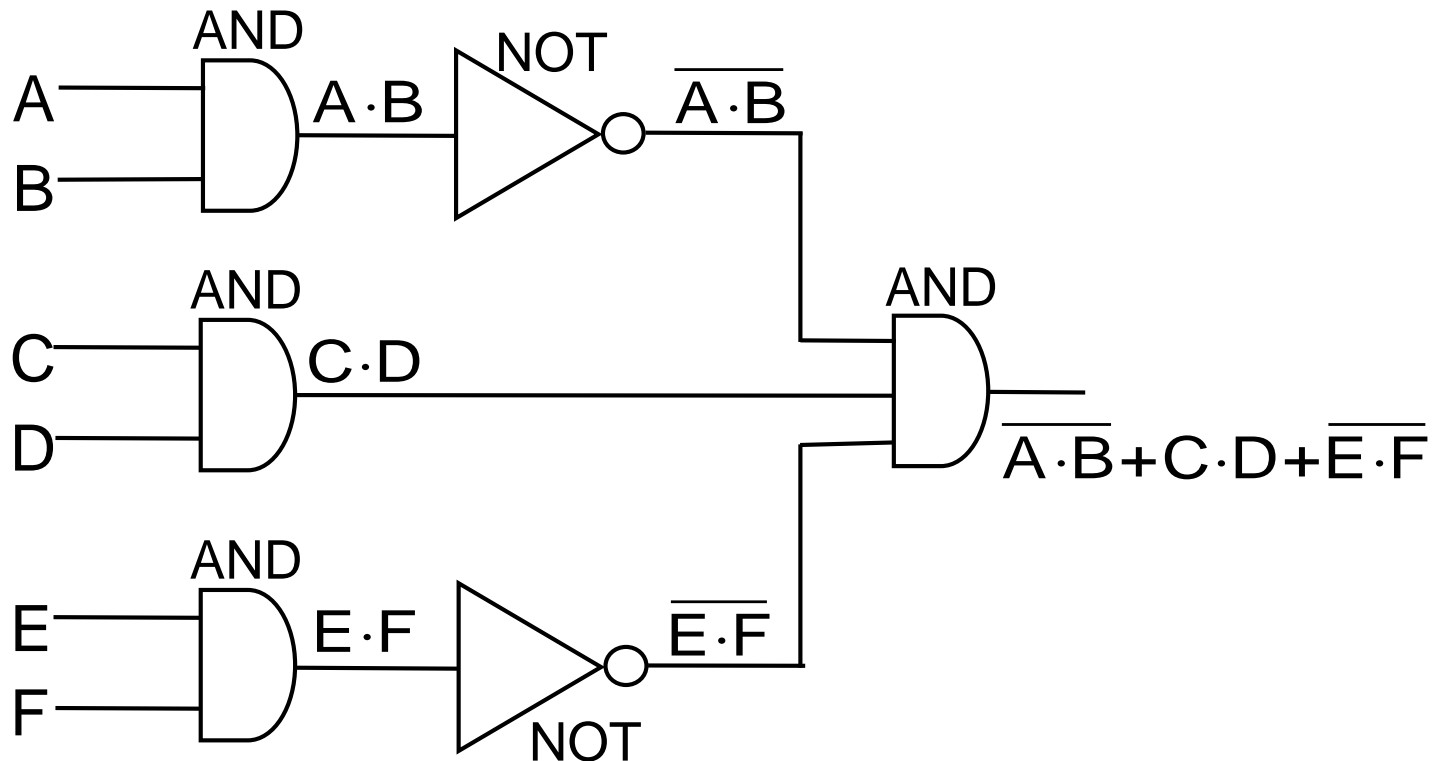
Boolean Expression =  $A \cdot B + C$



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



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

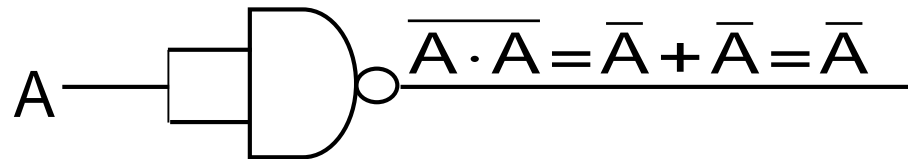


# Universal NAND Gate

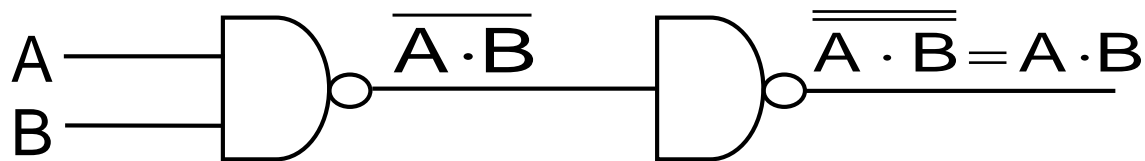


- NAND gate is an universal gate, it is alone sufficient to implement any Boolean expression
- **To understand this, consider:**
  - Basic logic gates (AND, OR, and NOT) are logically complete
  - Sufficient to show that AND, OR, and NOT gates can be implemented with NAND gates

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



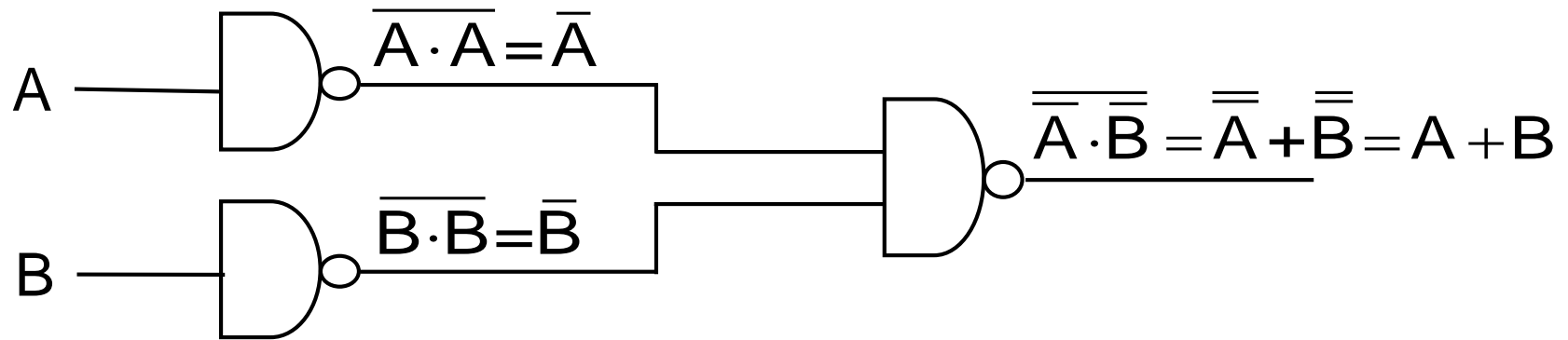
(a) NOT gate implementation.



(b) AND gate implementation.

*(Continued on next slide)*

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



(c) OR gate implementation.

# Method of Implementing a Boolean Expression with Only NAND Gates

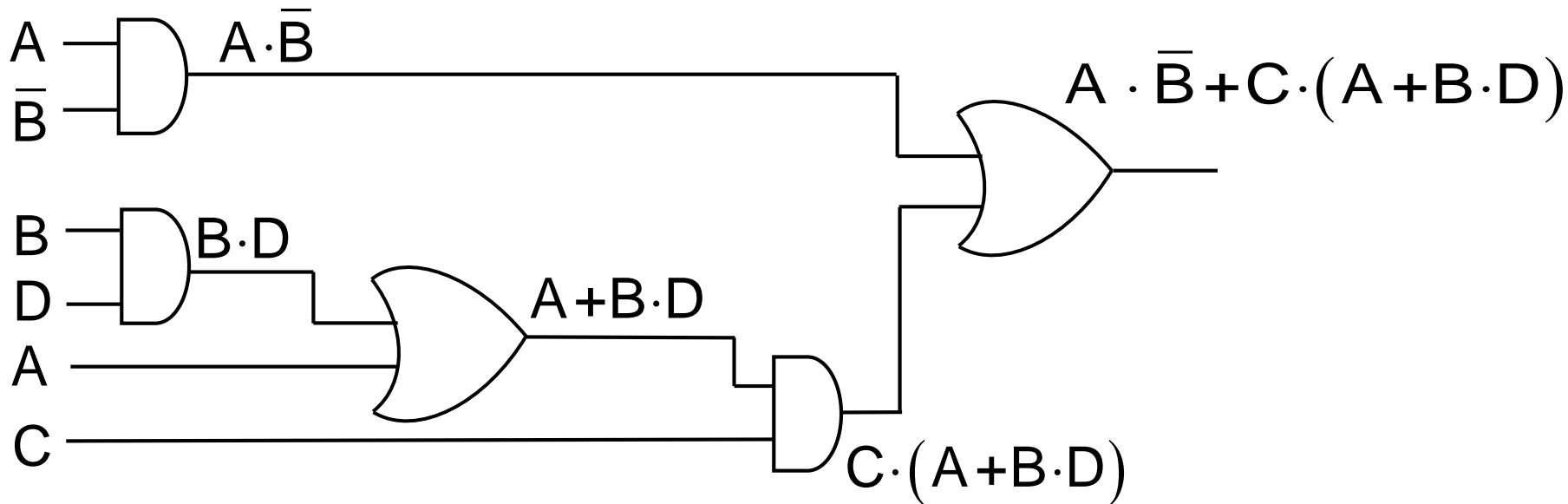


- Step 1: From the given algebraic expression, draw the logic diagram with AND, OR, and NOT gates. Assume that both the normal ( $A$ ) and complement ( $\overline{A}$ ) inputs are available
- Step 2: Draw a second logic diagram with the equivalent NAND logic substituted for each AND, OR, and NOT gate
- 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)



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

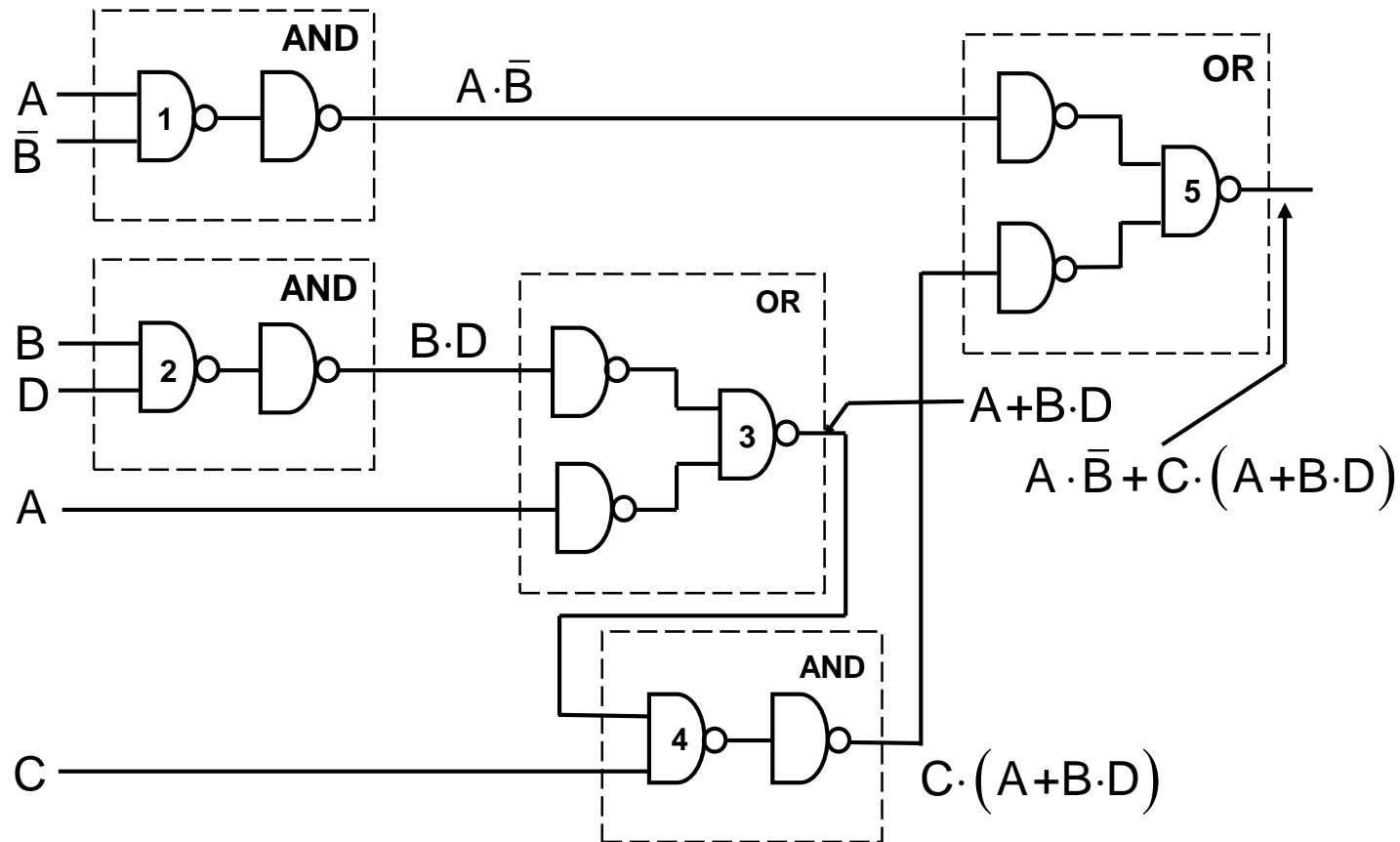


(a) **Step 1:** AND/OR implementation

*(Continued on next slide)*



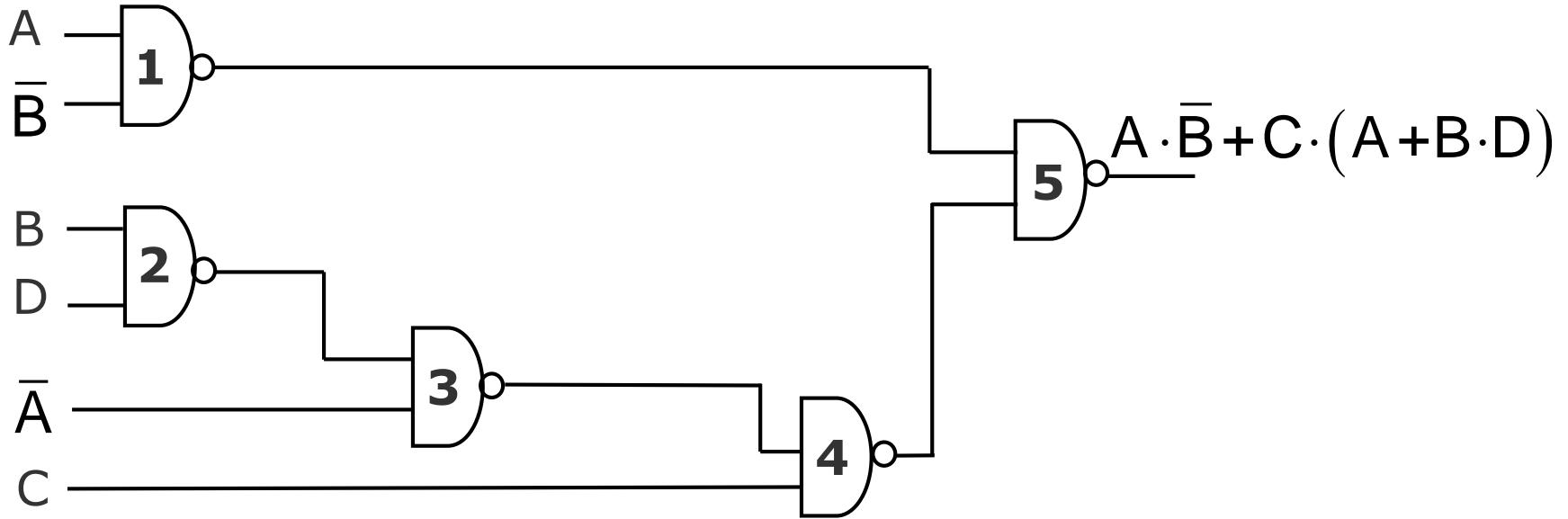
# Implementing a Boolean Expression with Only NAND Gates (Example)



(b) **Step 2:** Substituting equivalent NAND functions

*(Continued on next slide)*

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



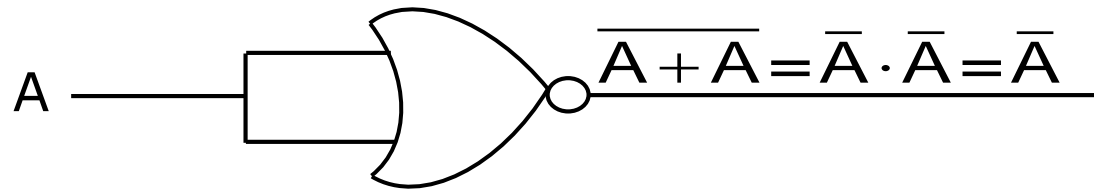
(c) **Step 3:** NAND implementation.

# Universal NOR Gate

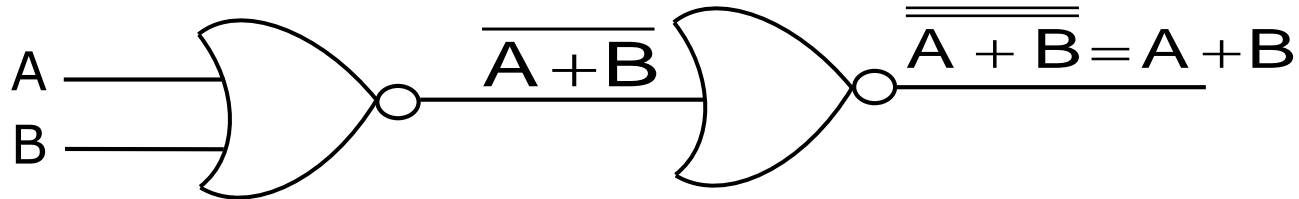


- NOR gate is an universal gate, it is alone sufficient to implement any Boolean expression
- To understand this, consider:
  - Basic logic gates (AND, OR, and NOT) are logically complete
  - Sufficient to show that AND, OR, and NOT gates can be implemented with NOR gates

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



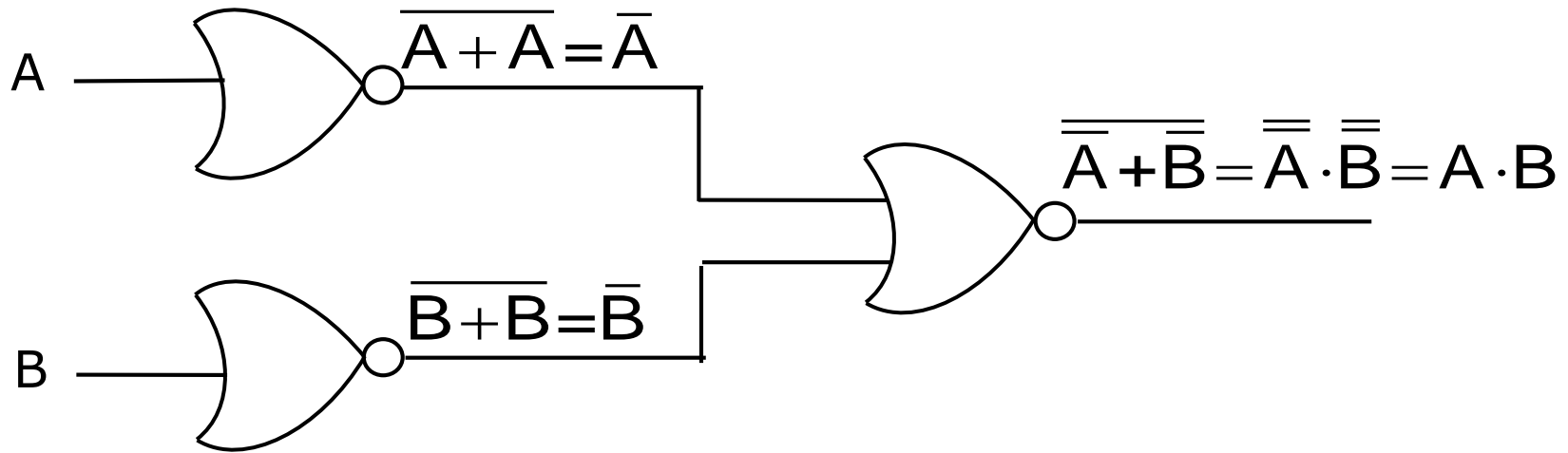
(a) NOT gate implementation.



(b) OR gate implementation.

(Continued on next slide)

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



(c) AND gate implementation.

# Method of Implementing a Boolean Expression with Only NOR Gates

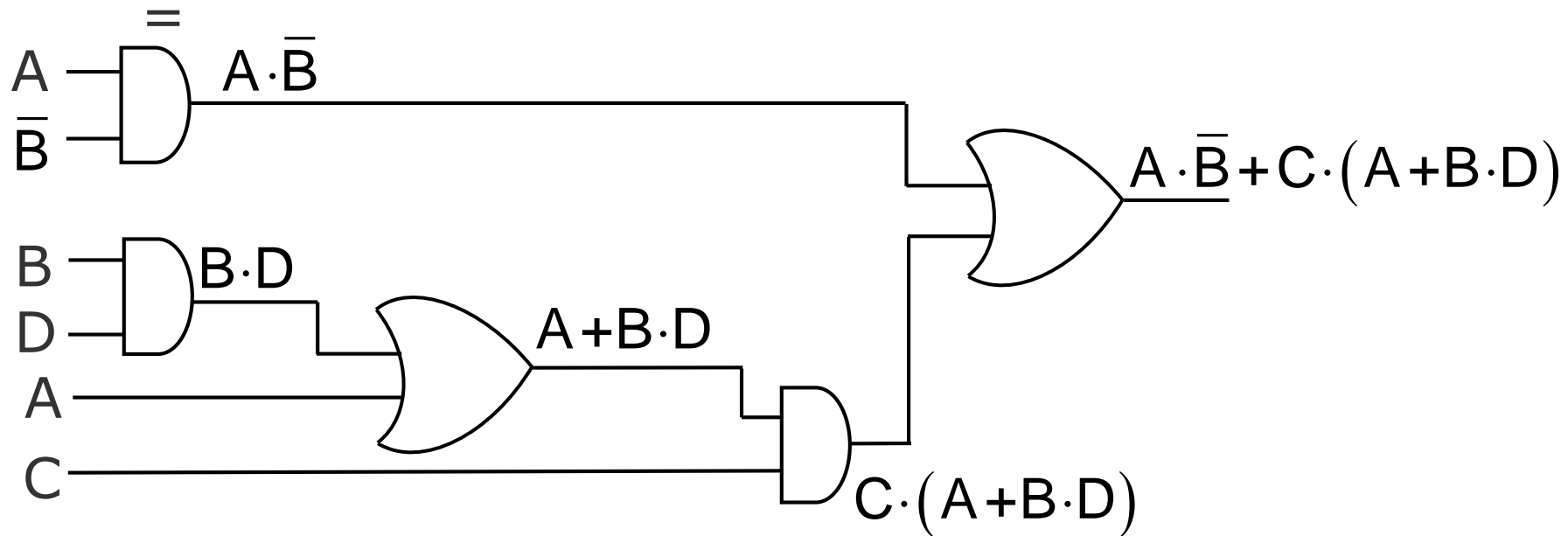


- Step 1: For the given algebraic expression, draw the logic diagram with AND, OR, and NOT gates. Assume that both the normal ( $A$ ) and complement ( $\bar{A}$ ) inputs are available
- Step 2: Draw a second logic diagram with equivalent NOR logic substituted for each AND, OR, and NOT gate
- Step 3: 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)



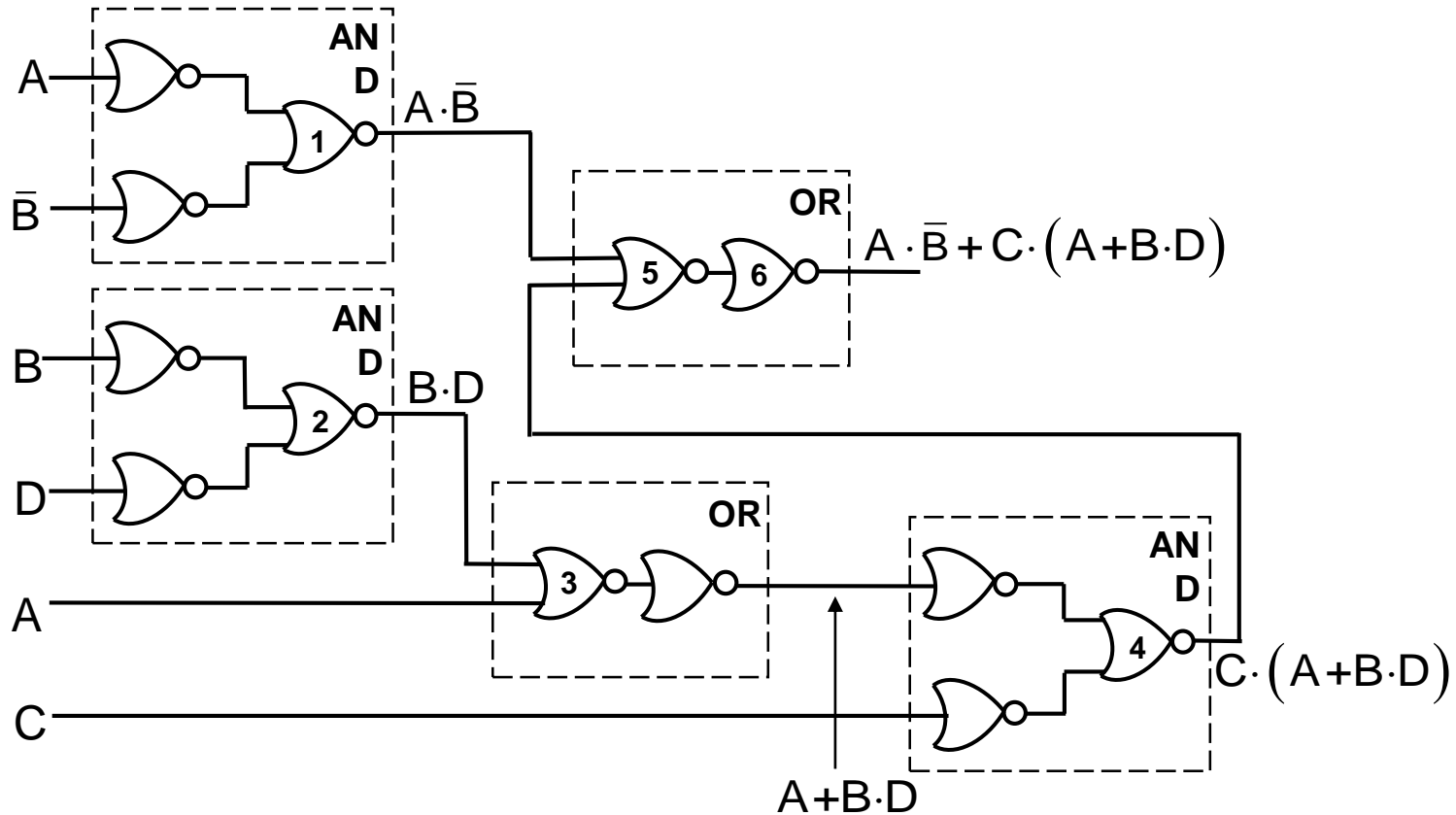
Boolean Expression  $A \cdot \bar{B} + C \cdot (A + B \cdot D)$



(a) **Step 1:** AND/OR implementation.

*(Continued on next slide)*

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

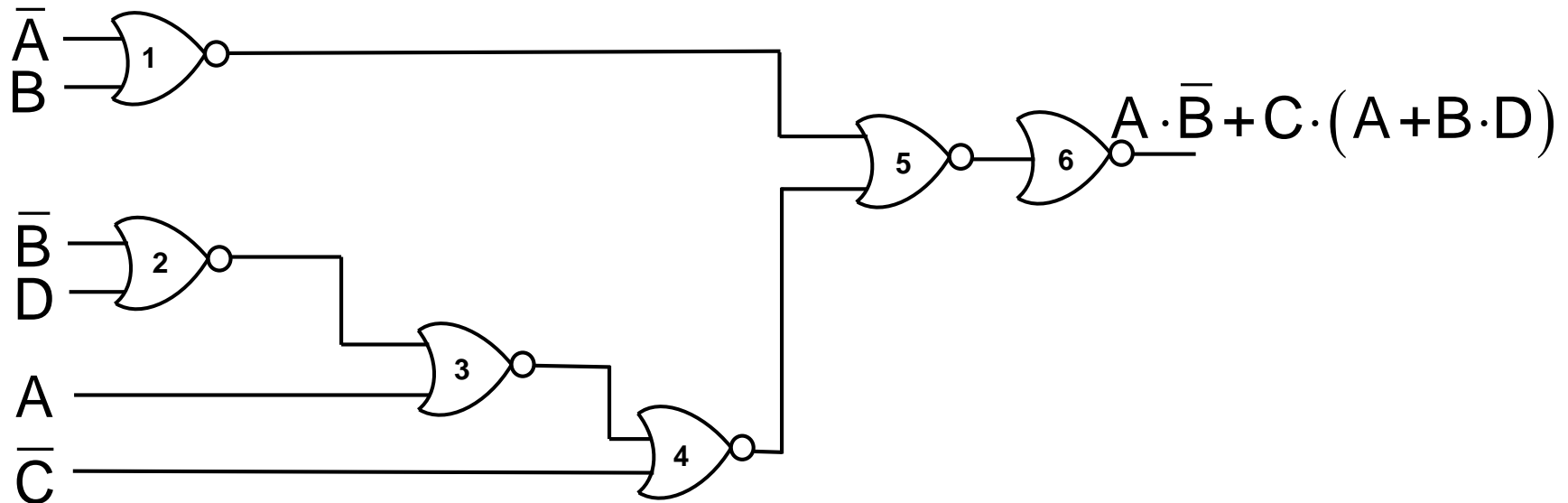


(b) **Step 2:** Substituting equivalent NOR functions.

(Continued on next slide)



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

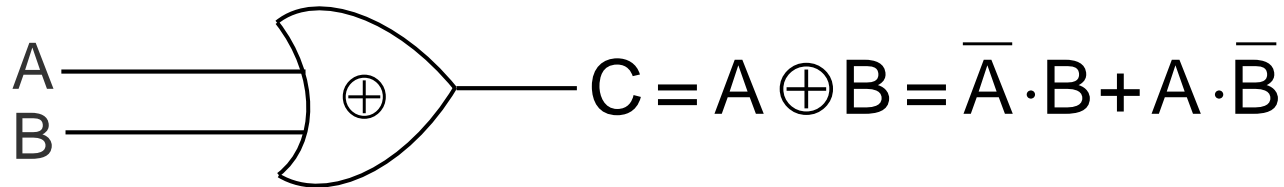
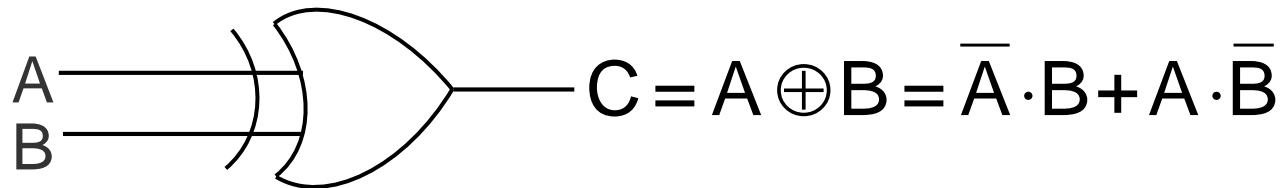


(c) **Step 3:** NOR implementation.

# Exclusive-OR Function



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



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)

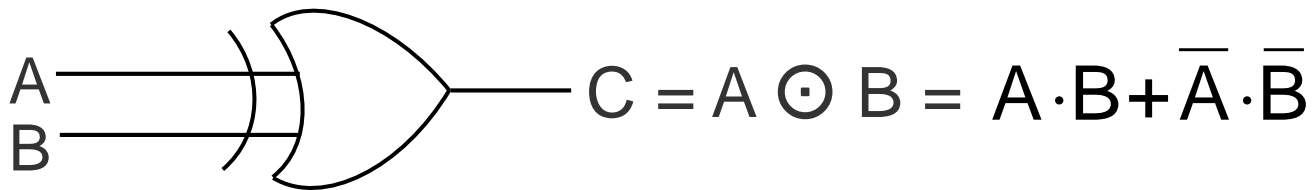


Inputs		Output
A	B	$C = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

# Equivalence Function with Block Diagram Symbol



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



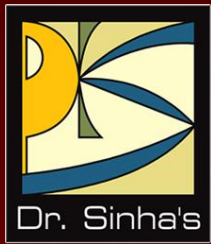
$$\text{Also, } (A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$$

*(Continued on next slide)*

# Equivalence Function (Truth Table)



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



# Design of Combinational Circuits



# 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

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

$$C = A \cdot B$$

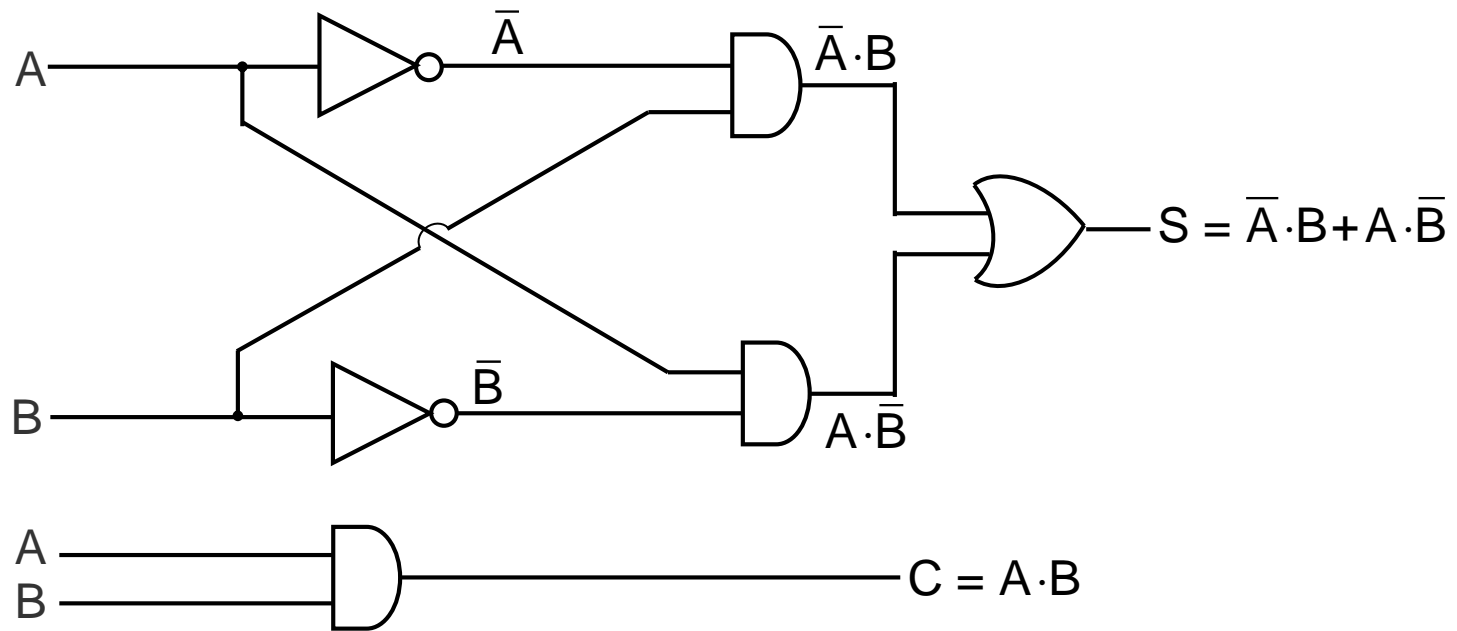


Boolean functions for the two outputs.



# Designing a Combinational Circuit

## Example 1 – Half-Adder Design



Logic circuit diagram to implement the Boolean functions

# Designing a Combinational Circuit

## Example 2 – Full-Adder Design



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

Truth table for a full adder

*(Continued on next slide)*

# Designing a Combinational Circuit

## Example 2 – Full-Adder Design



**Boolean functions for the two outputs:**

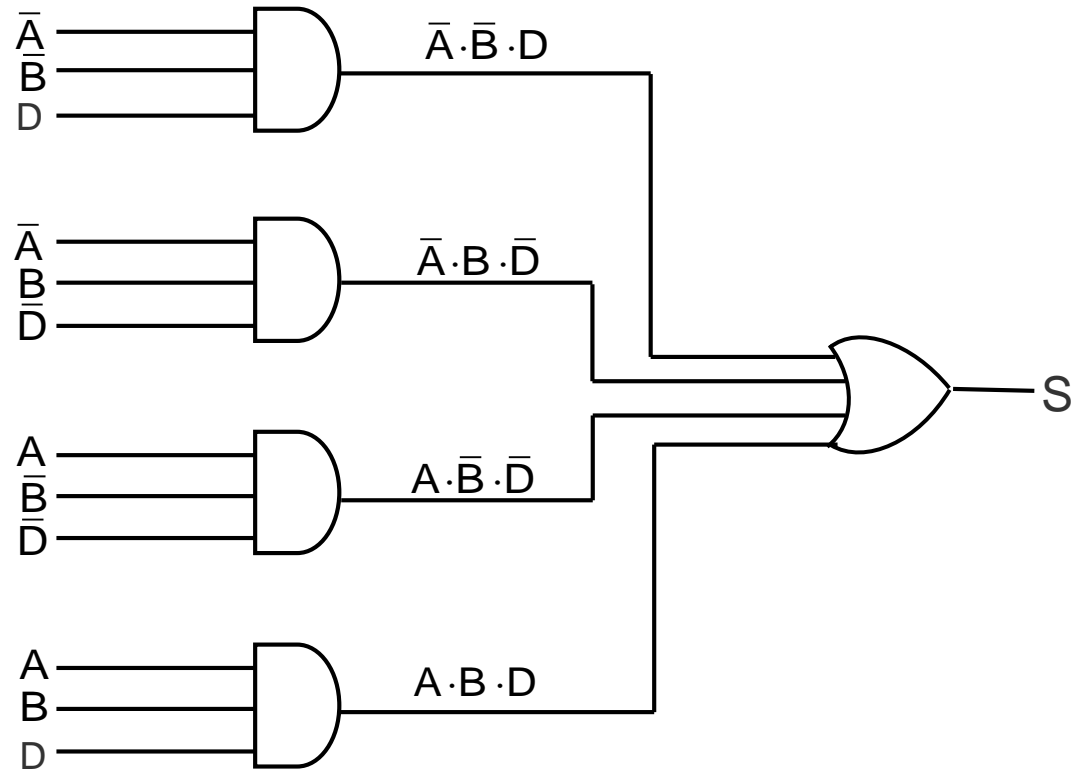
$$S = \bar{A} \cdot \bar{B} \cdot D + \bar{A} \cdot B \cdot \bar{D} + A \cdot \bar{B} \cdot \bar{D} + A \cdot B \cdot D$$

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

*(Continued on next slide)*

# Designing a Combinational Circuit

## Example 2 – Full-Adder Design

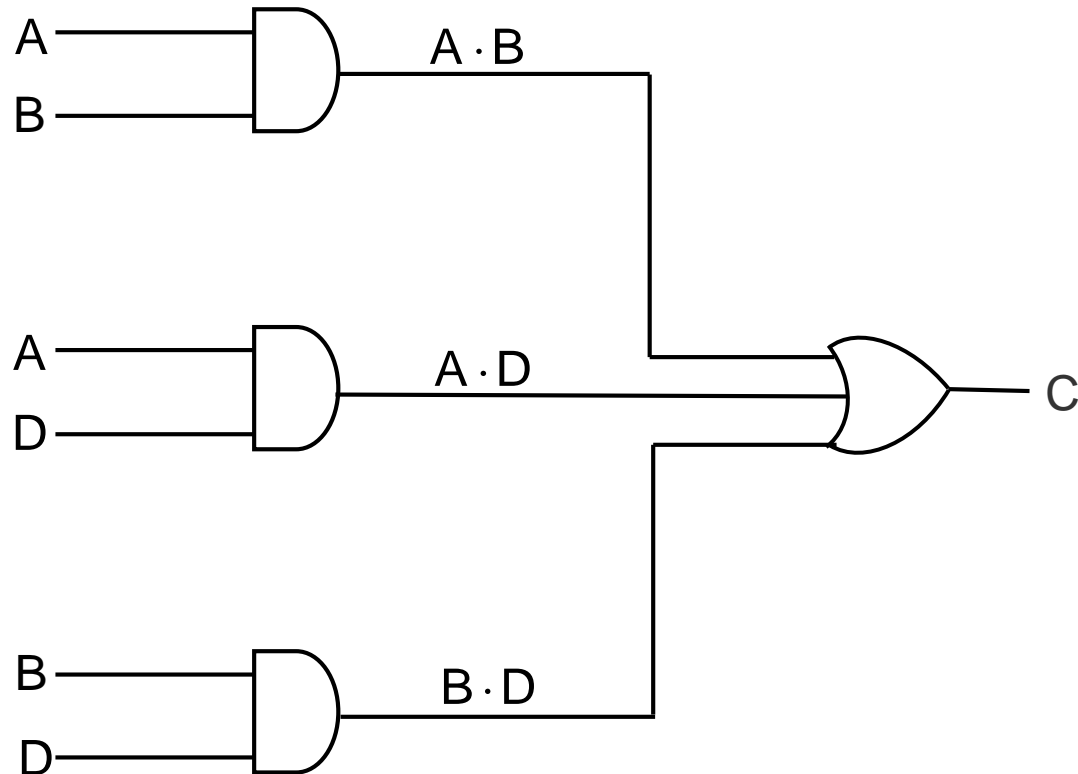


(a) Logic circuit diagram for sums

*(Continued on next slide)*

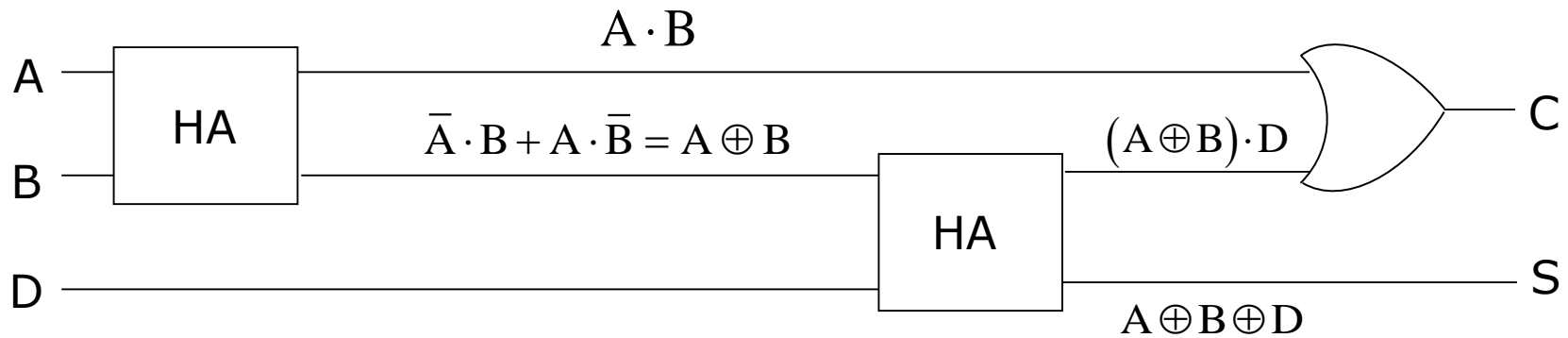
# Designing a Combinational Circuit

## Example 2 – Full-Adder Design

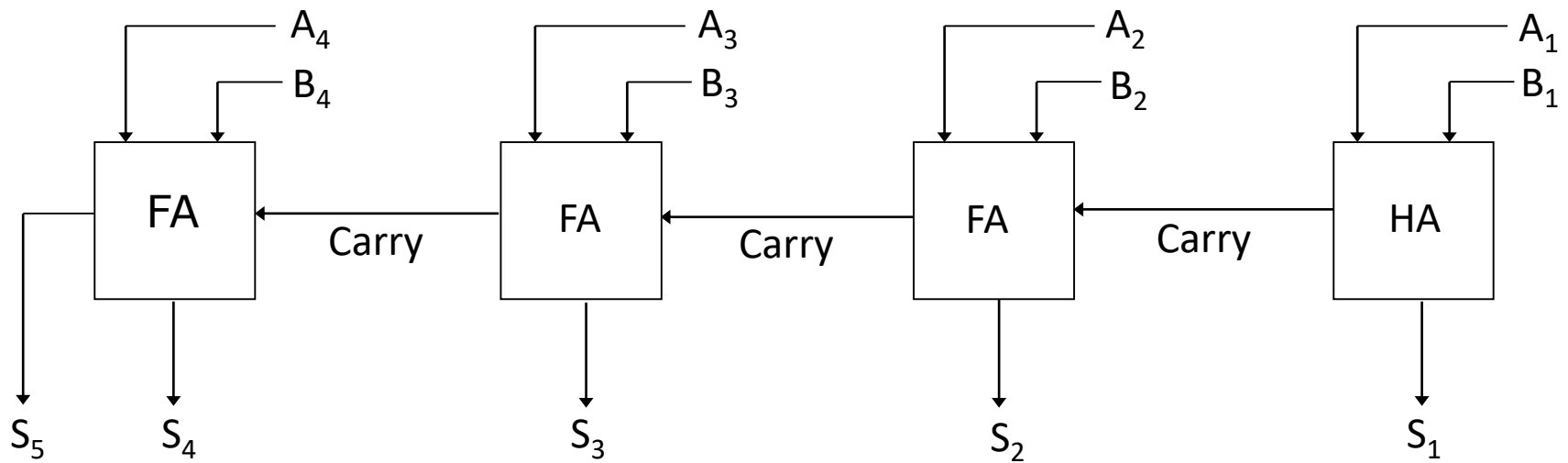


(b) Logic circuit diagram for carry

# Designing a Combinational Circuit: Full-Adder Design using Half-Adders



# Parallel Binary Adder

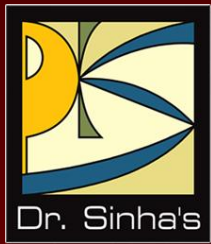


# Key Words/Phrases



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





# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 7

**Processor and  
Memory**



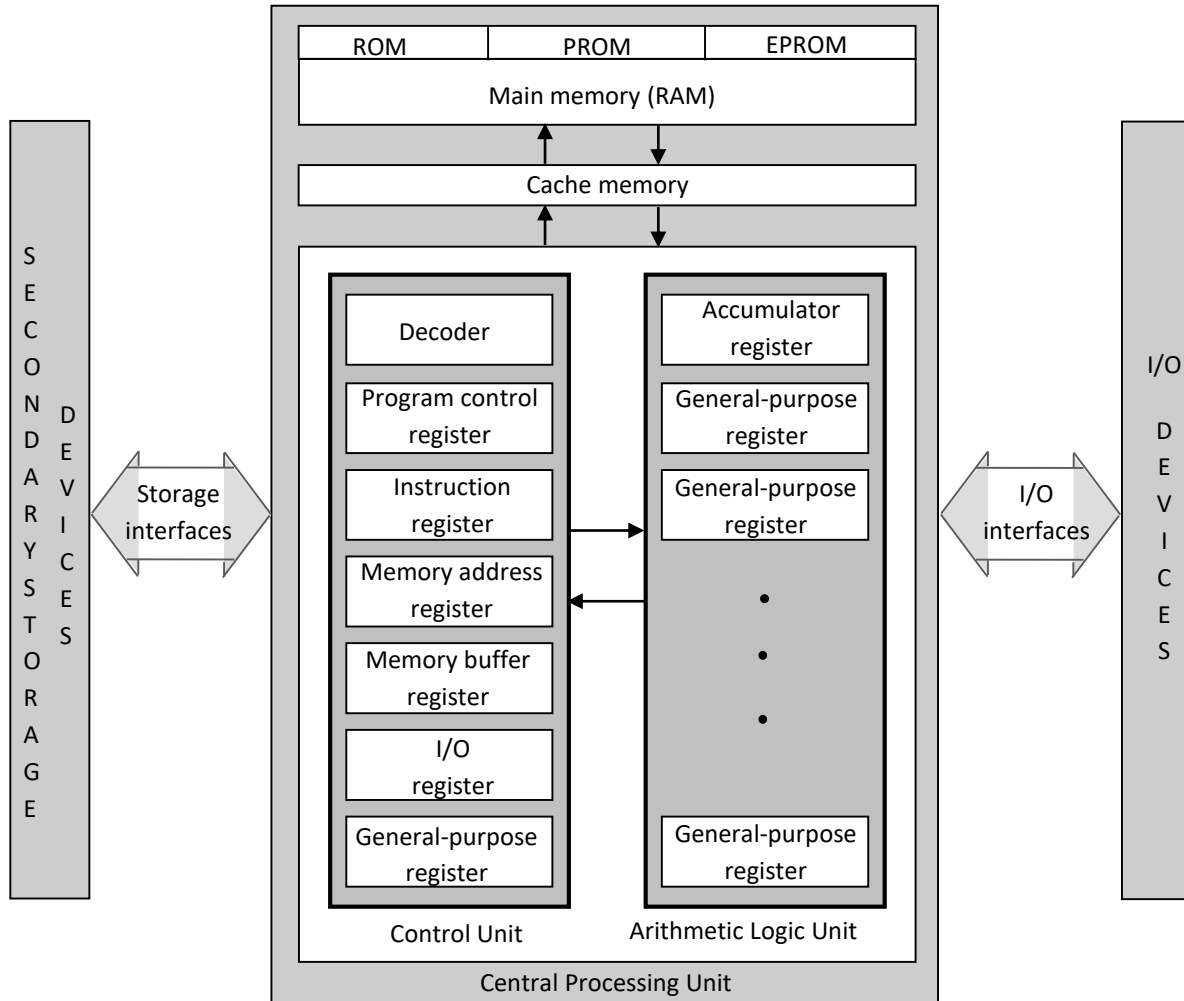
# Learning Objectives

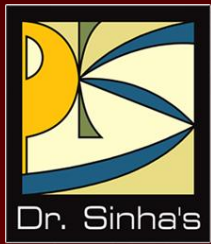


## In this chapter you will learn about:

- 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)



# 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 information on its way to and from memory
3	Program Control (PC)	Holds address of the next instruction to be executed
4	Accumulator (A)	Accumulates results and data to be operated upon
5	Instruction (I)	Holds an instruction while it is being executed
6	Input/Output (I/O)	Communicates with I/O devices

# Execution of Instructions



- Control unit takes address of the next program instruction to be executed from program control register and reads the instruction from corresponding memory address into the instruction register
- Control unit then sends the operation and address parts of the instruction to the decoder and memory address register
- Decoder interprets the instruction and accordingly the control unit sends command signals to the appropriate unit for carrying out the task specified in the instruction
- As each instruction is executed, address of next instruction is loaded and steps are repeated

# 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)
- We measure processing speed of PCs in MHz or GHz, of workstations and servers in MIPS or BIPS, and of supercomputers in GFLOPS (gigaflops, which refers to a billion FLOPS) or TFLOPS (teraflops, which refers to  $10^{12}$  FLOPS), or PFLOPS (petaflops, which refers to  $10^{15}$  FLOPS)

# Types of Processors



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 Processors



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 Processors



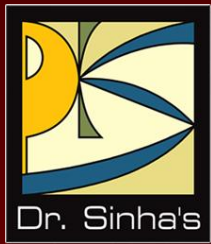
Type of Architecture	Features	Usage
Multi-Core Processor	<ul style="list-style-type: none"><li>Processor chip has multiple cooler-running, more energy-efficient processing cores</li><li>Improve overall performance by handling more work in parallel</li><li>can share architectural components, such as memory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

# Power-Efficient Processors



- Manufacturers of computing systems have made attempts to reduce power consumption of systems
- New processor architectures to reduce power consumption right at processor level
- latest processor offers a technology called *Demand Based Switching (DBS)* for reduced power consumption
- Processors based on DBS technology are designed to run at multiple frequency and voltage settings
- processors automatically switch to and operate at the lowest setting that is consistent with optimal application performance





# Main Memory



# Main Memory



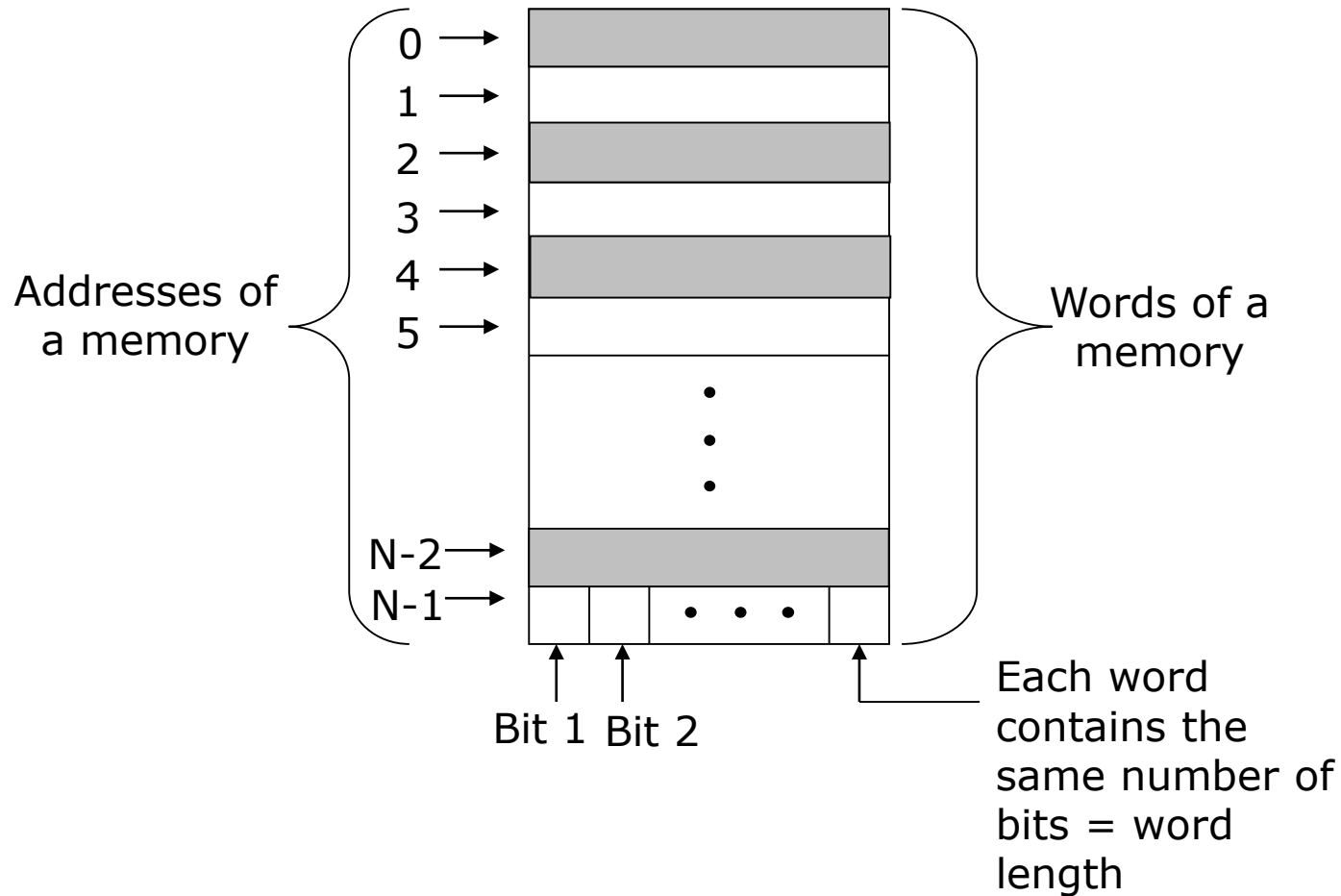
- Every computer has a temporary storage built into the computer hardware
- It stores instructions and data of a program mainly when the program is being executed by the CPU
- This temporary storage is known as main memory, primary storage, or simply *memory*
- Physically, it consists of some chips either 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



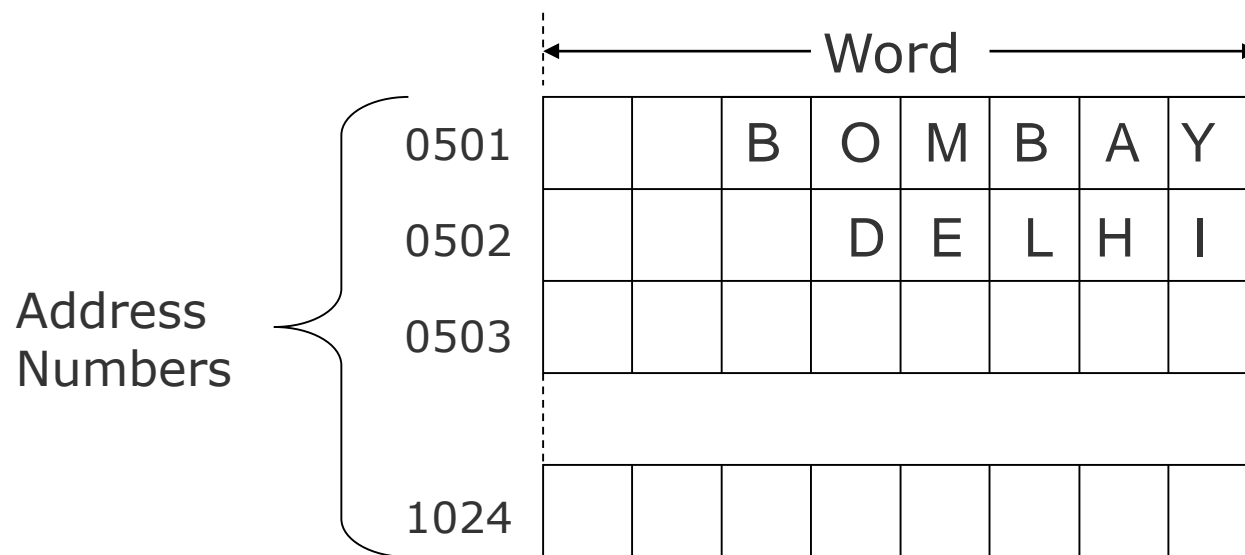
*(Continued on next slide)*

# Main Memory Organization



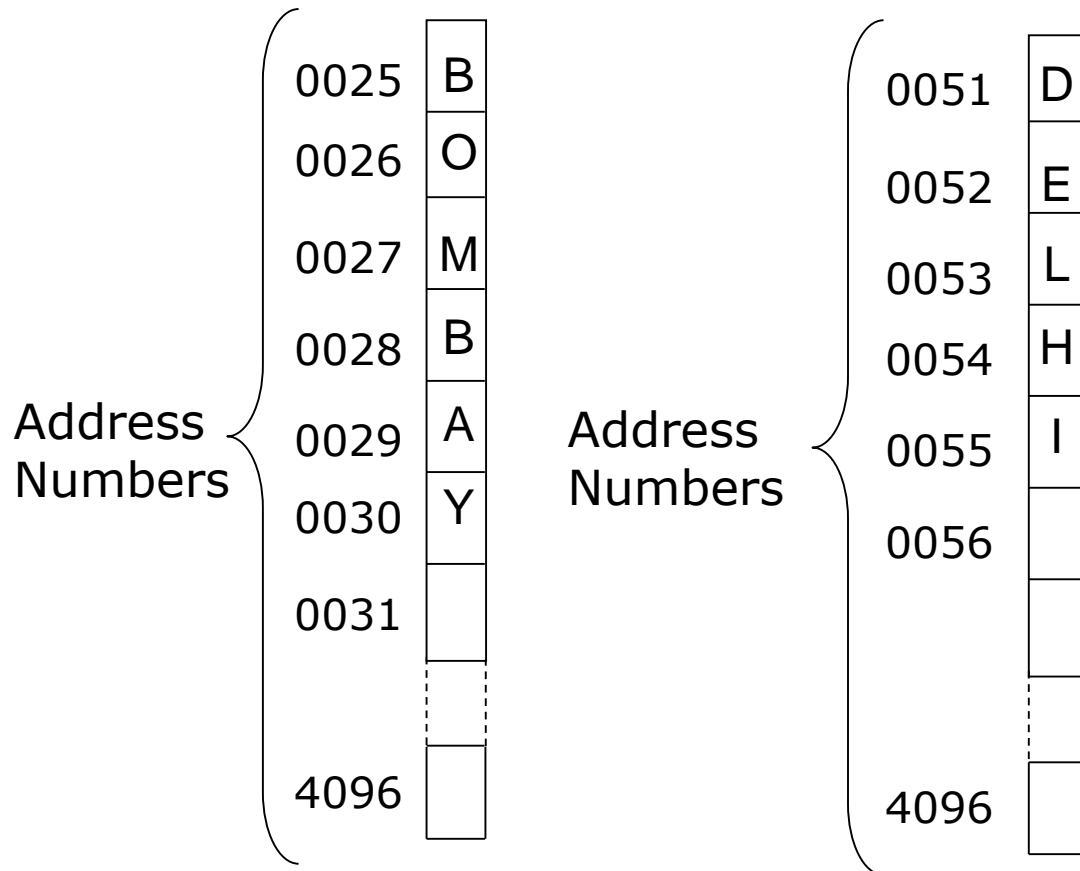
- Machines having smaller word-length are slower in operation than machines having larger word-length
- A *write* to a memory location is destructive to its previous contents
- A *read* from a memory location is non-destructive to its previous contents

# 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



- Each memory location can store only a single character
- Slower in speed of calculation than fixed word-length memory
- Used in small business computers for optimizing the use of storage space

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

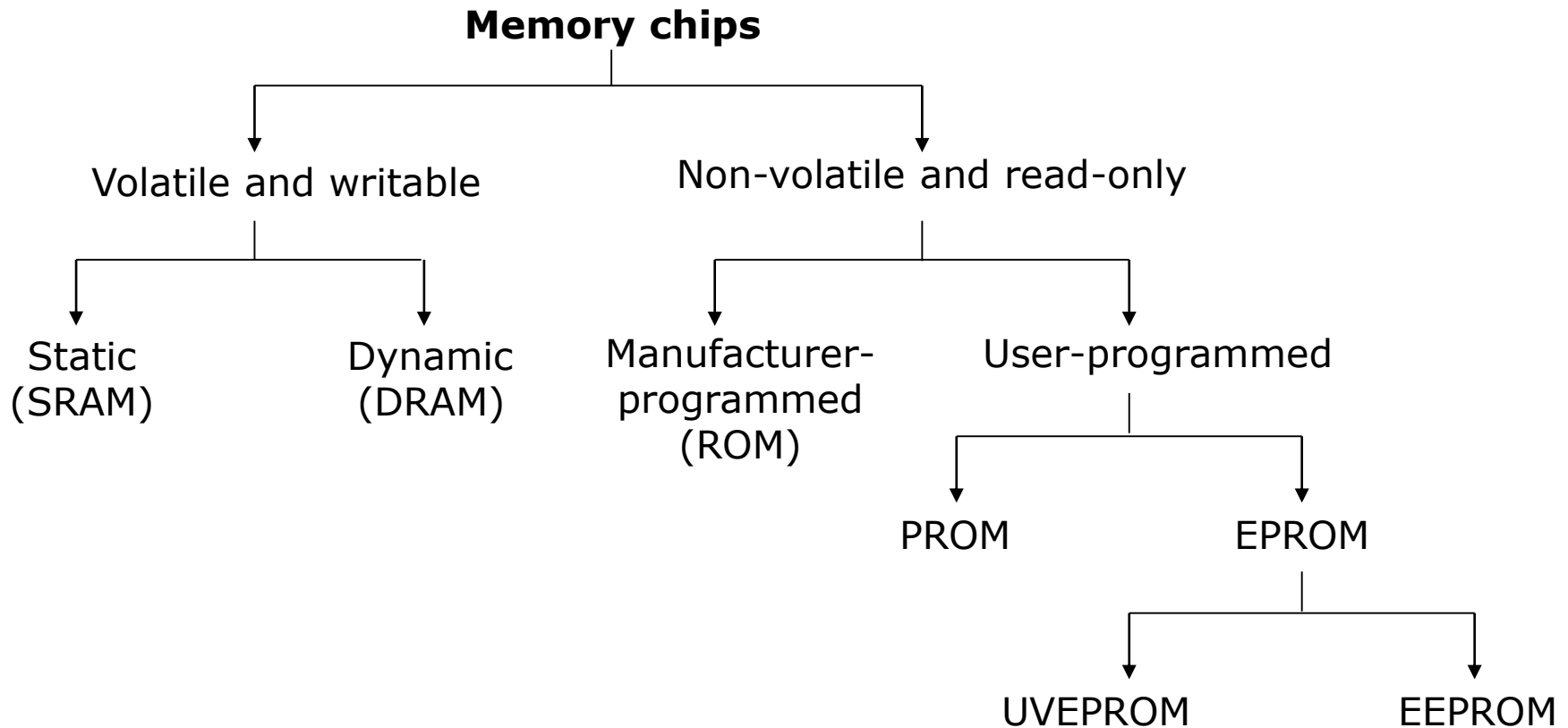
# Memory Capacity



- 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



# Types of Memory Chips



# 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



Type	Usage
Ultra Violet EPROM (UVEPROM)	A type of EPROM 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 EPROM 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)

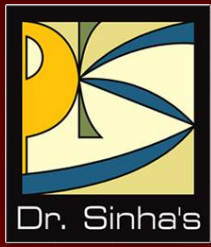
*(Continued on next slide)*

# Key Words/Phrases



- 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





# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 8

## Secondary Storage Devices



# 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



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

# Limitations of Primary Storage



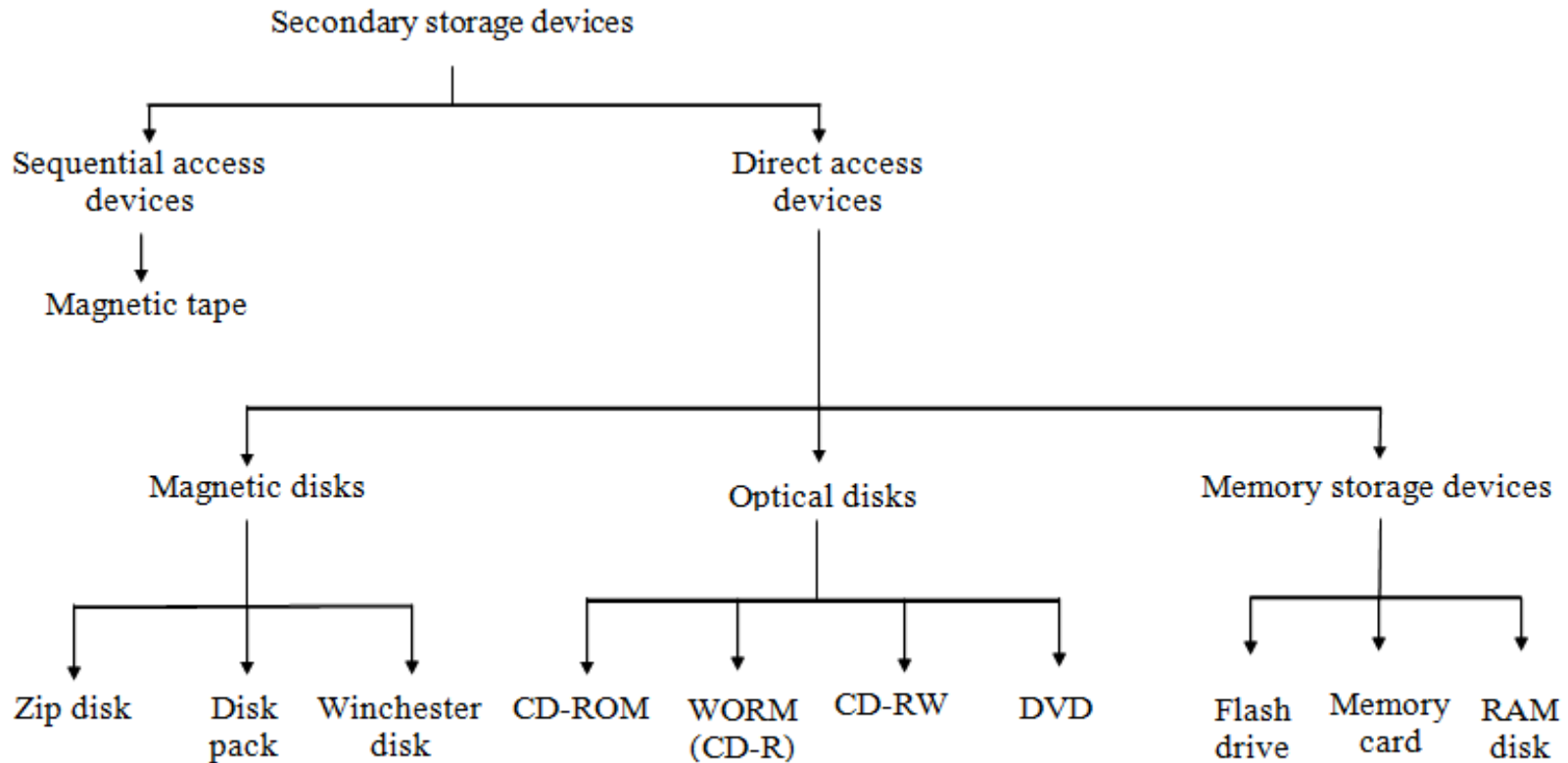
- Limited capacity because the cost per bit of storage is high
- Volatile - data stored in it is lost when the electric power is turned off or interrupted

# Secondary Storage

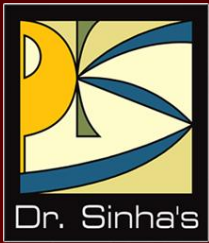


- Used in a computer system to overcome the limitations of primary storage
- Has virtually unlimited capacity because the cost per bit of storage is very low
- Has an operating speed far slower than that of the primary storage
- Used to store large volumes of data on a permanent basis
- Also known as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

# Classification of Commonly Used Secondary Storage Devices



**Figure 8.1.** Commonly used secondary storage devices and their classification.



# Sequential and Direct-Access Storage Devices



# Sequential-access Storage Devices



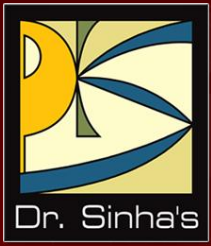
- Arrival at the desired storage location may be preceded by sequencing through other locations
- Data can only be retrieved in the same sequence in which it is stored
- Access time varies according to the storage location of the information being accessed
- Suitable for sequential processing applications 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 Tapes

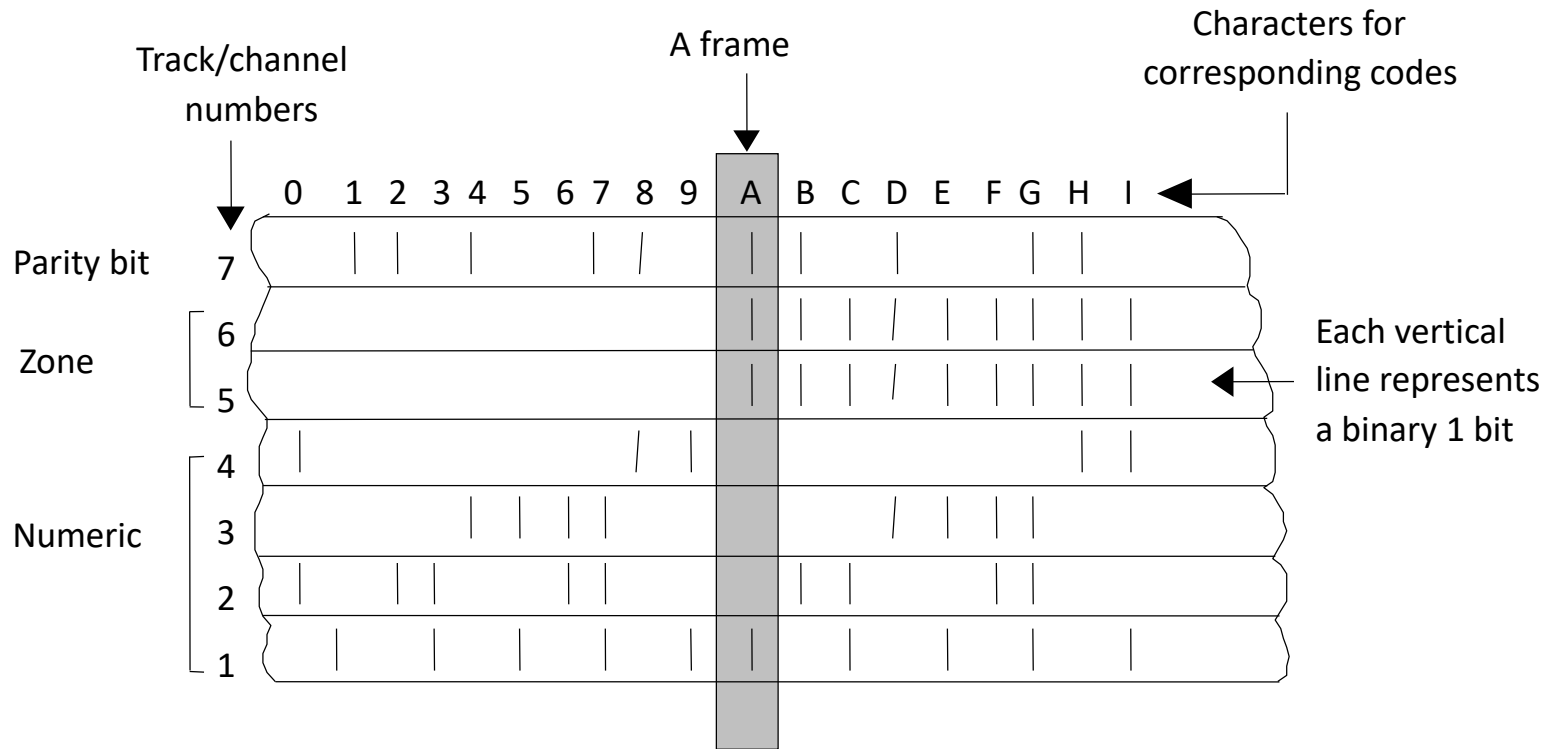


# Magnetic Tape Basics



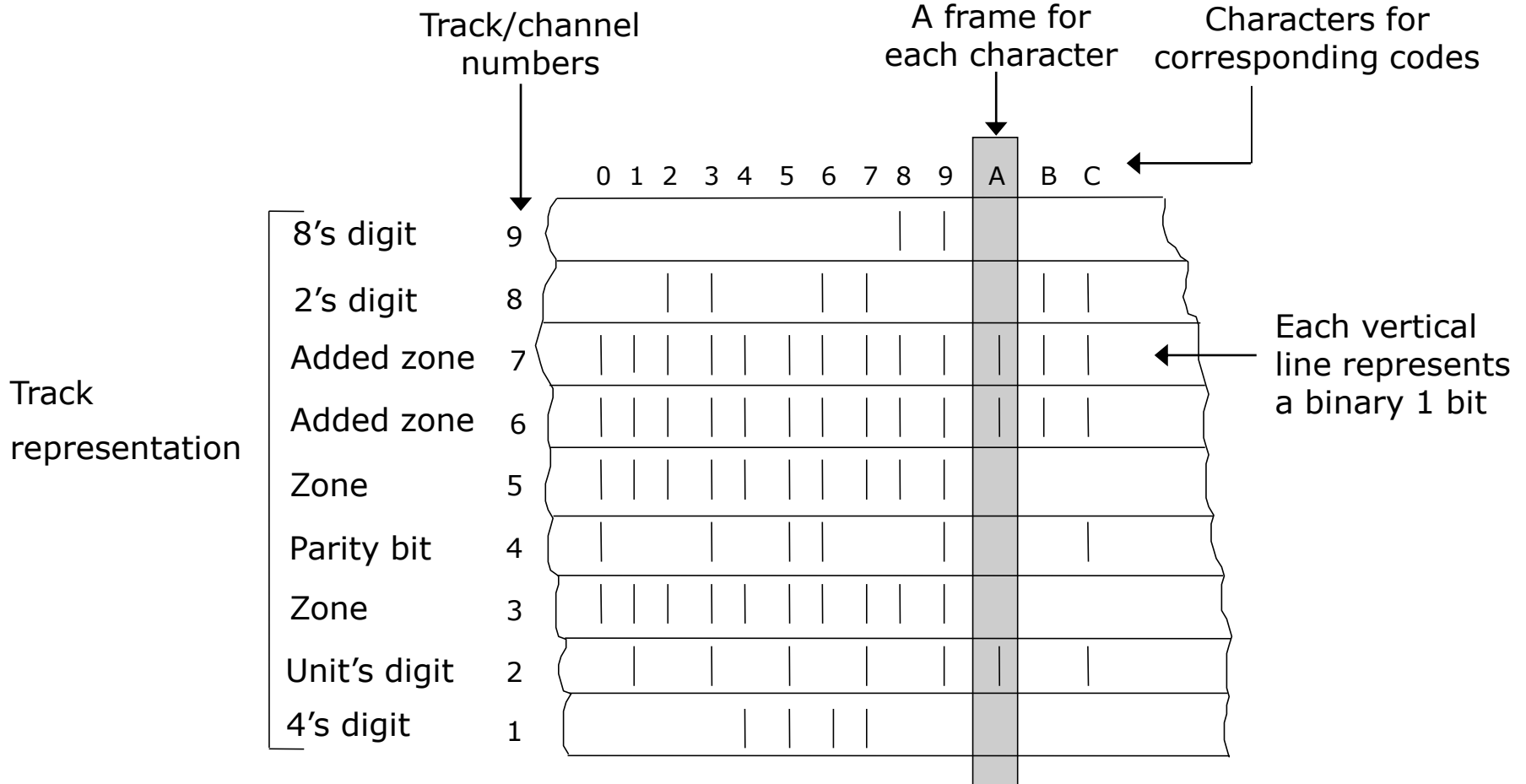
- Commonly used sequential-access secondary storage device
- Physically, the tape medium is a plastic ribbon, which is usually  $\frac{1}{2}$  inch or  $\frac{1}{4}$  inch wide and 50 to 2400 feet long
- Plastic ribbon is coated with a magnetizable recording material such as iron-oxide or chromium dioxide
- 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 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



<i>Forward space one block</i>	forwards the tape to the beginning of next block
<i>Forward space one file</i>	forwards the tape to the beginning of next file
<i>Rewind</i>	fully rewinds the tape
<i>Unload</i>	releases the tape drive's grip so that the tape spool can be unmounted from the tape drive

# Types of Magnetic Tape



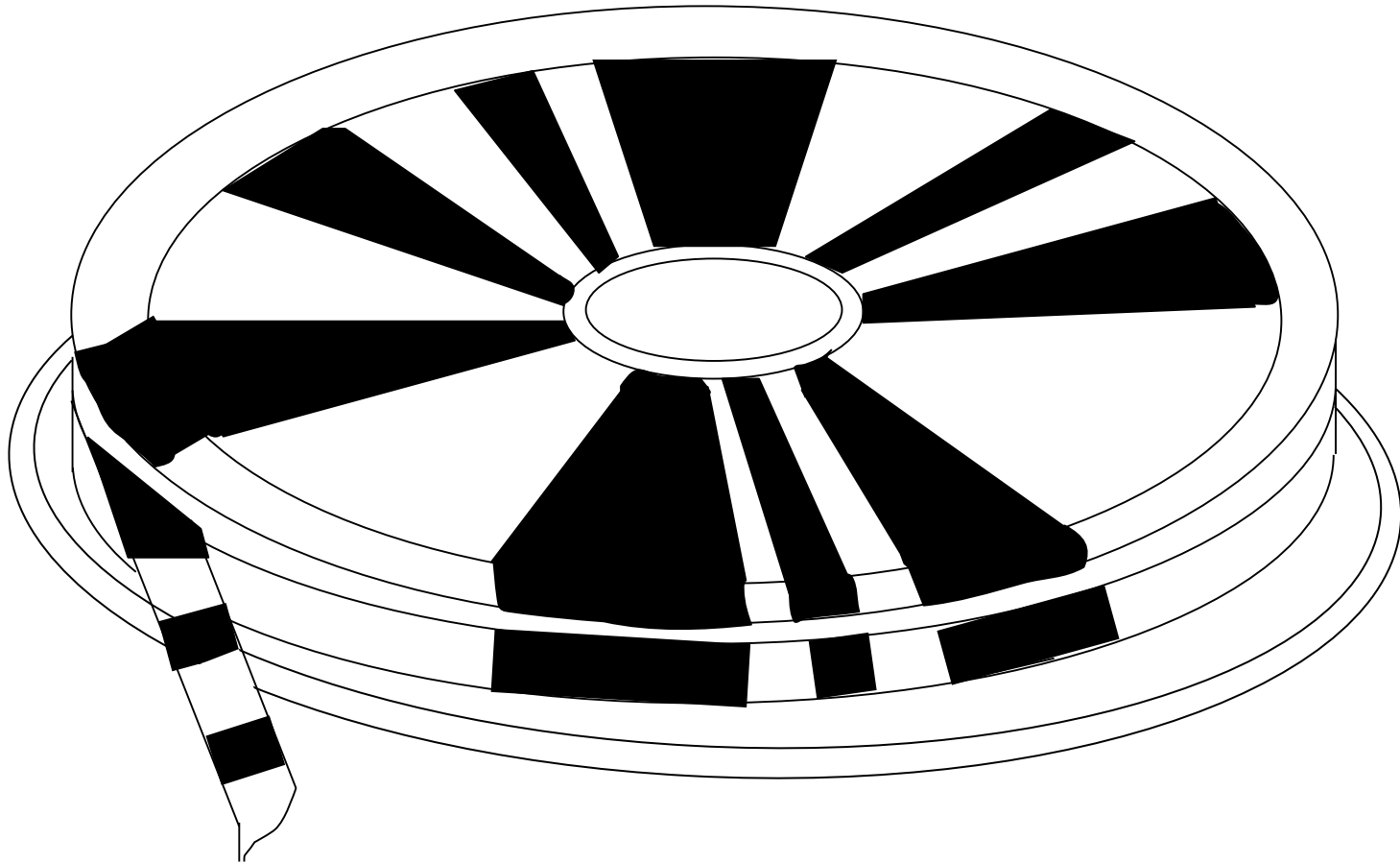
- 1/2-inch tape reel
- 1/2-inch tape cartridge
- 1/4-inch streamer tape
- 4-mm digital audio tape (DAT)

# Half-inch Tape Reel

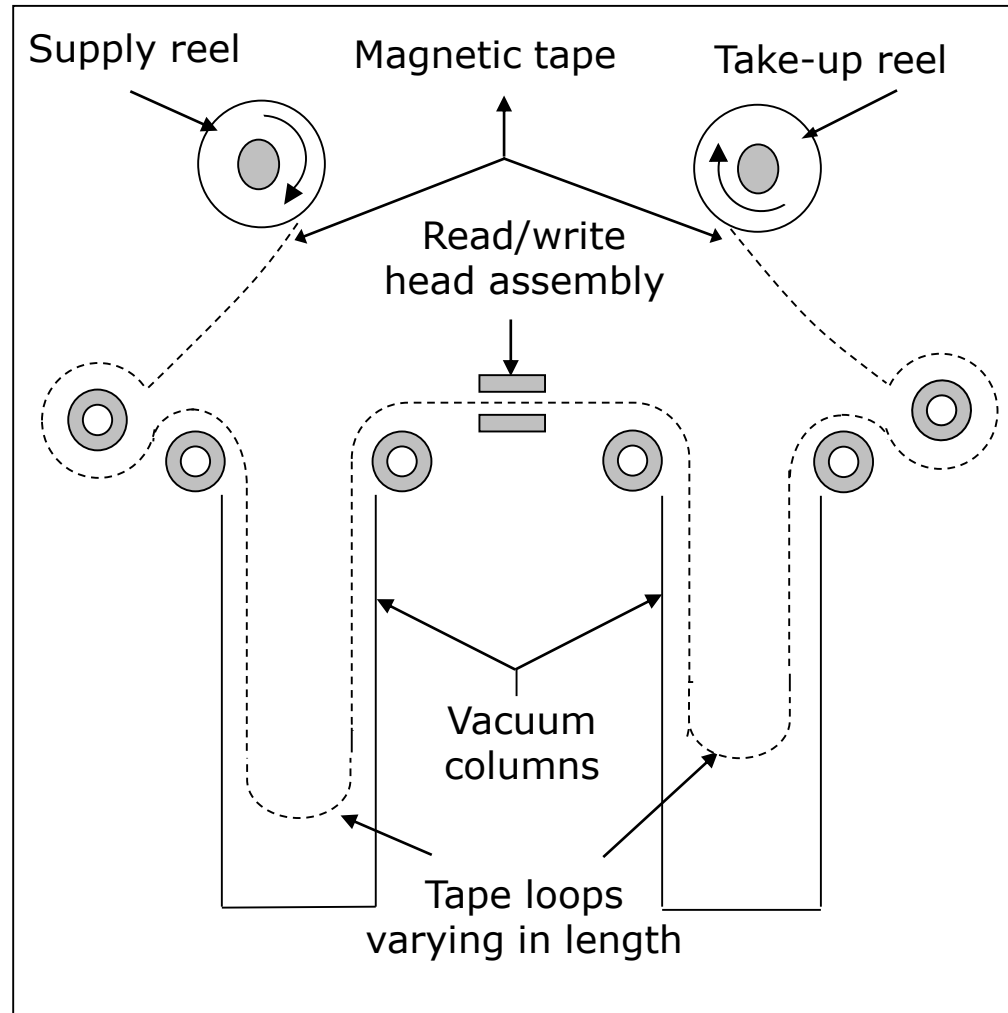


- Uses 1/2 inch wide tape ribbon stored on a tape reel
- Uses parallel representation method of storing data, in which data are read/written a byte at a time
- Uses a read/write head assembly that has one read/write head for each track
- Commonly used as archival storage for off-line storage of data 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

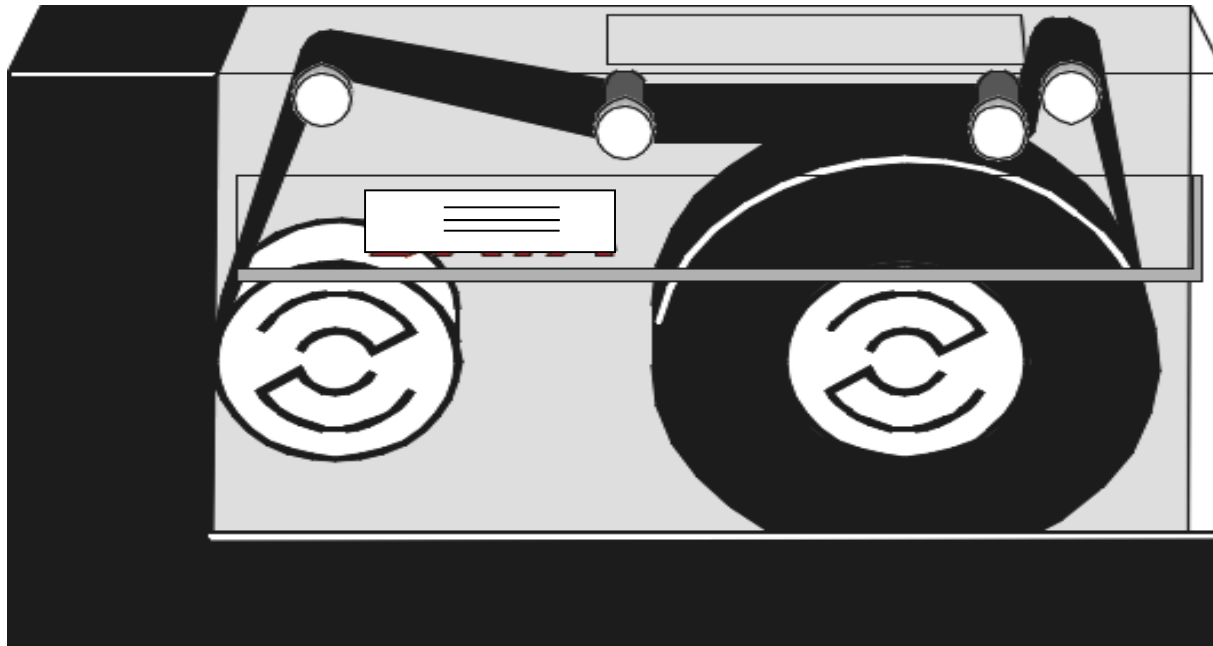


# Half-inch Tape Cartridge



- Uses ½ 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 ¼ 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

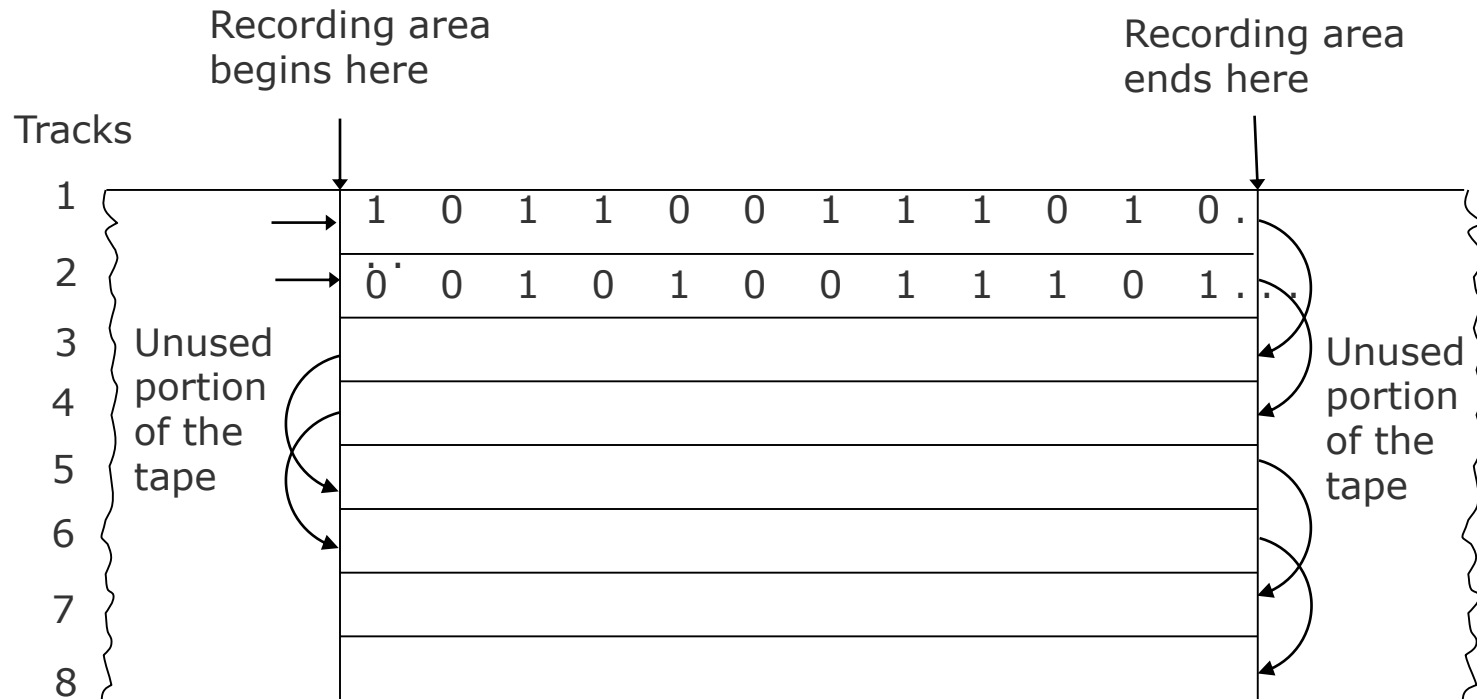
*(Continued on next slide)*

# Quarter-inch Streamer Tape



- 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)

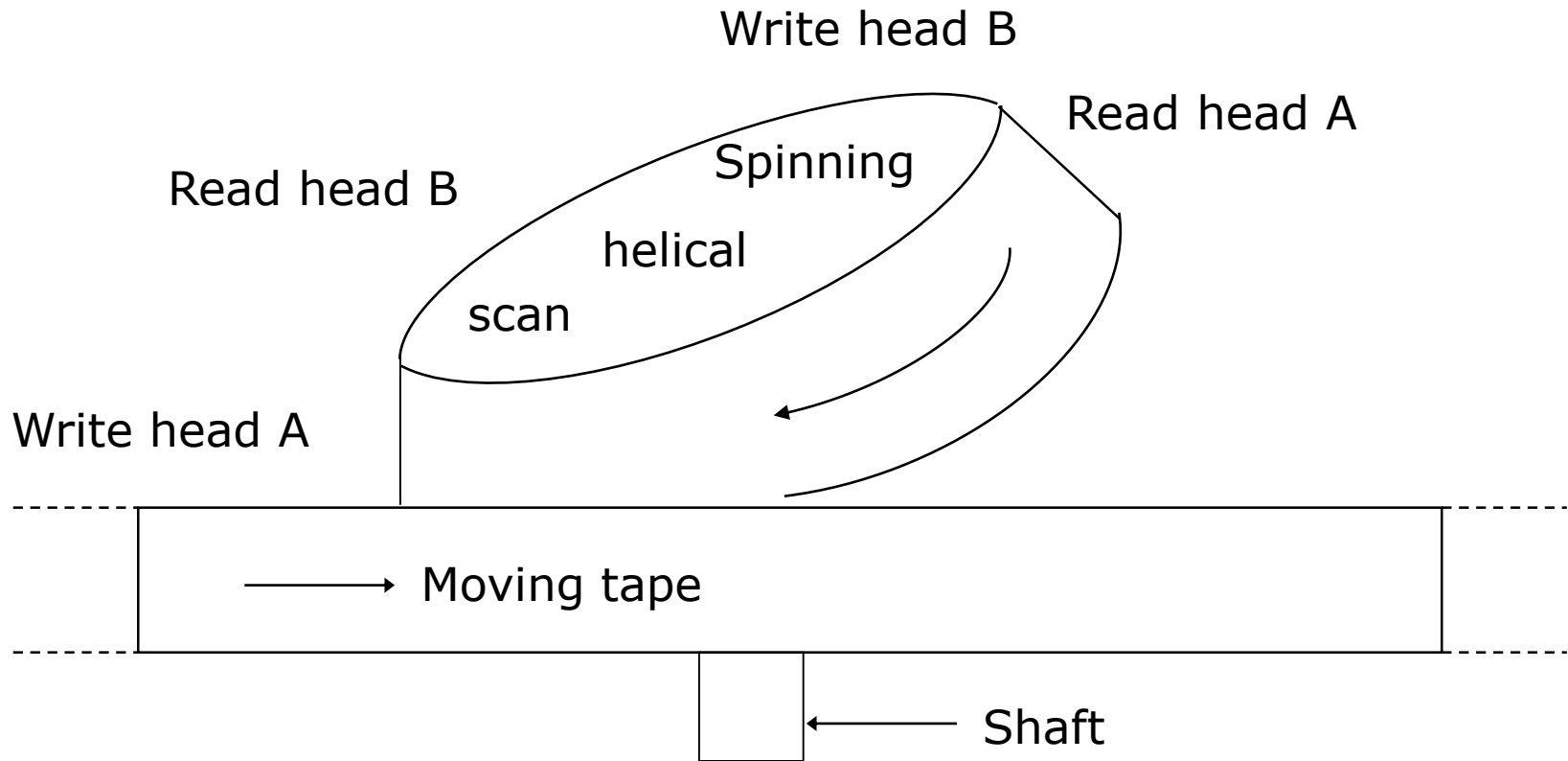


# 4mm Digital Audio Tape (DAT)



- 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



- 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

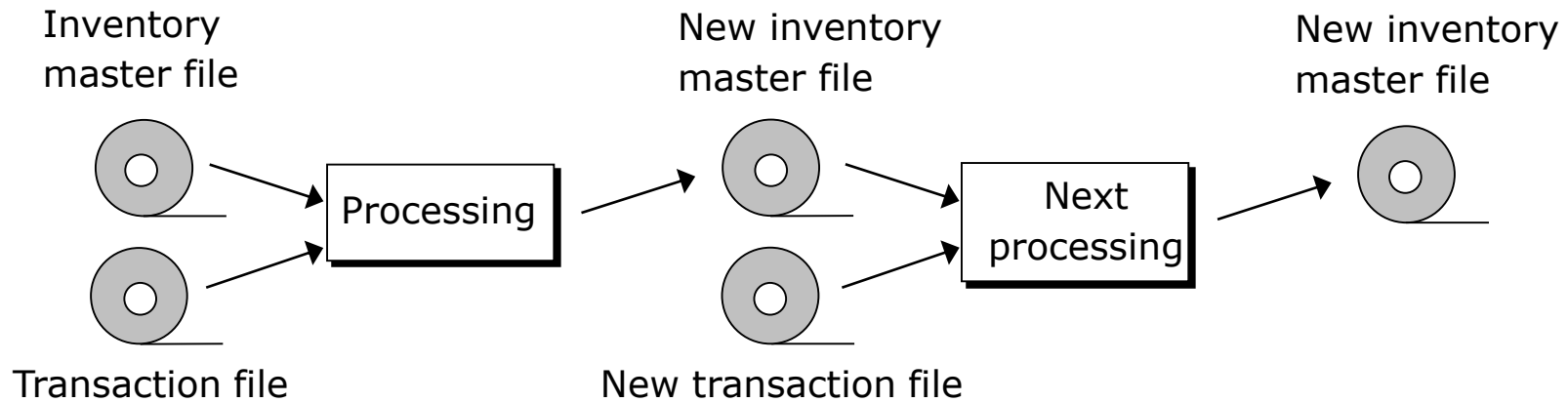


# Uses of Magnetic Tapes

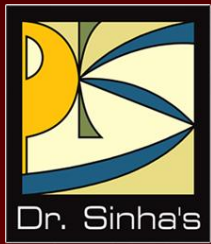


- For applications that are based on sequential data processing
- Backing up of data for off-line storage
- Archiving of infrequently used data
- Transferring of data from one computer to another that are not linked together
- As a distribution media for software by vendors

# Uses of Magnetic Tapes



**Illustrating the use of tapes in a sequential application.**



# Magnetic Disks

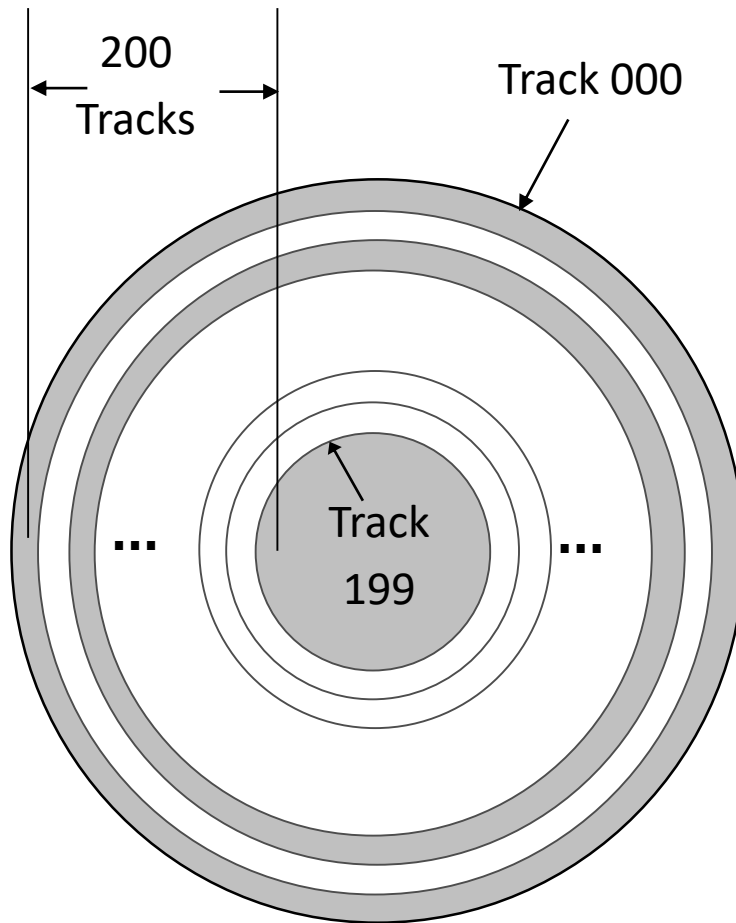


# Magnetic Disk - Basics



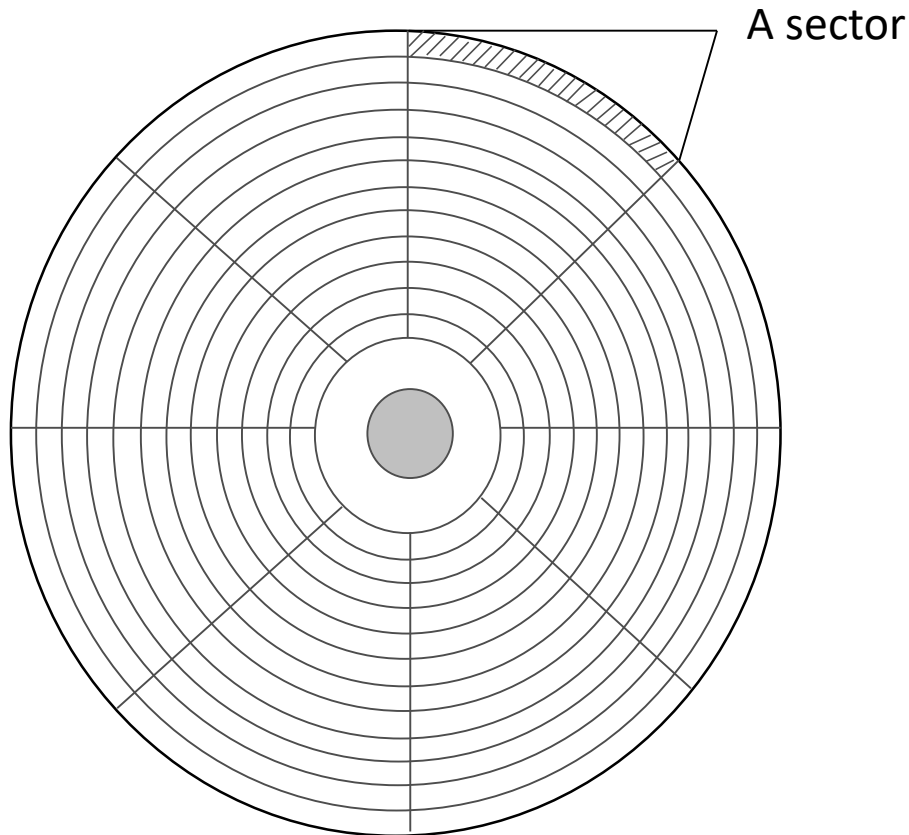
- Commonly used direct-access secondary storage device.
- Physically, a magnetic disk is a thin, circular plate/platter made of metal or plastic that is usually coated on both sides with a magnetizable recording material such as iron-oxide
- Data are recorded on the disk in the form of tiny invisible 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



- 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

# Disk Pack



- Data is accessed from a disk by specifying its disk address
- It is comprised of sector number, track number, and surface number
- Operating systems combine two or more sectors to form a *cluster*
- In this case, the smallest unit of data access from a disk is a cluster, instead of a sector
- Read/write operations read/write a whole cluster at a time
- Multiple disks are stacked together as a disk pack
- Disk pack is sealed and mounted on a disk drive consisting of a motor to rotate the disk pack about its axis
- Disk drive has an access arms assembly having separate read/write heads for each surface of the disk pack

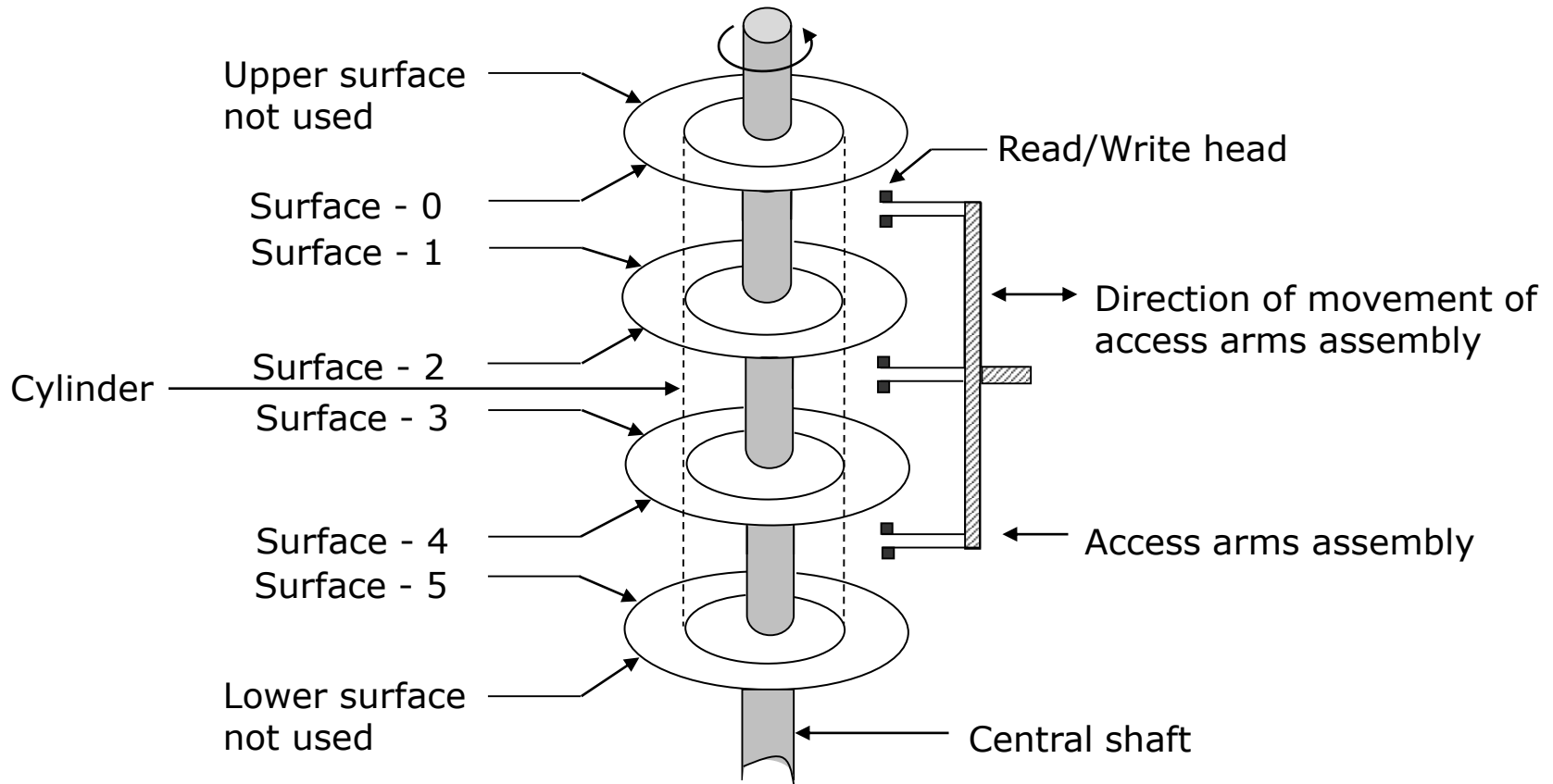
# Cylinder-Head-Sector (CHS)



- Corresponding tracks on all recording surfaces of a disk pack together form a cylinder
- Addressing scheme is called *CHS* (*Cylinder-Head-Sector*) addressing
- It is also known 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



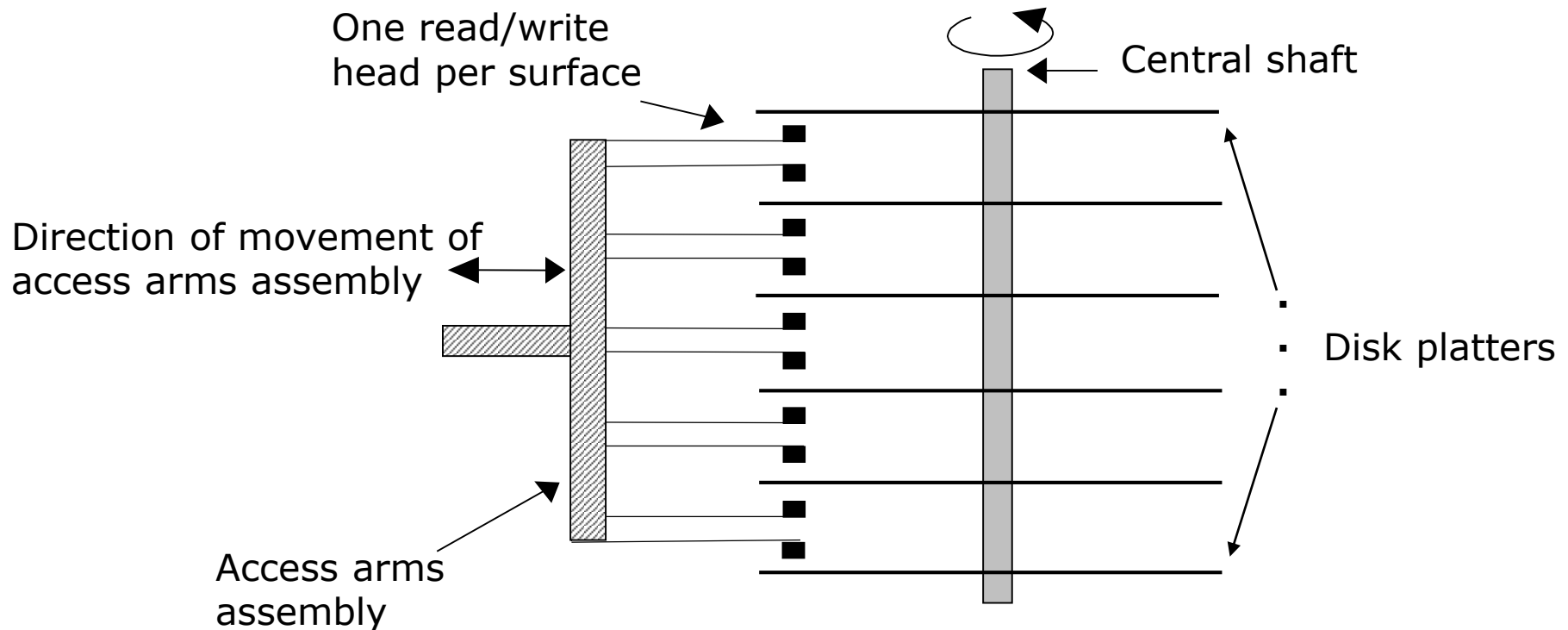
Storage capacity of a disk system = Number of recording surfaces  
× Number of tracks per surface  
× Number of sectors per track  
× Number of bytes per sector

# Access Mechanism



- Data is recorded on the tracks of a spinning disk surface and read from the surface by one or more read/write heads
- Read/write heads are mounted on an access arms assembly
- Access arms assembly moves in and out
- Read/write heads move horizontally across the surfaces of the disks
- In case of a disk pack all heads move together
- Read/write heads are of flying type
- They do not have direct contact with disk surfaces

# 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

= Average seek time + 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



- 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



- 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

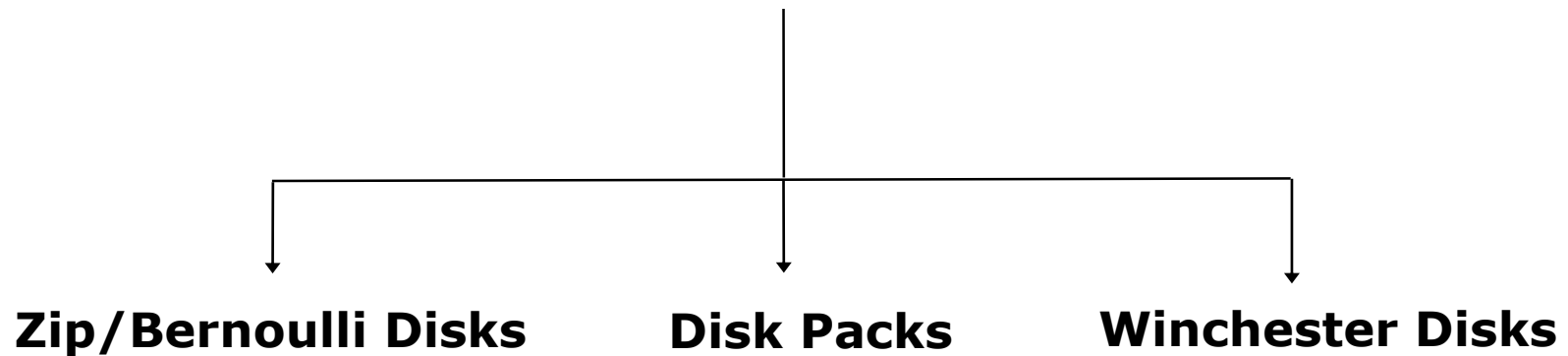


- Disk drive is connected to and controlled by a disk controller, which interprets the commands for operating the disk drive
- Typically supports only *read* and *write* commands, which need disk address (surface number, cylinder/track number, and sector number) as parameters
- Connected to and controls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

# Types of Magnetic Disks



## Magnetic Disks



# 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



- 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



- 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



- 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



- 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



- 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



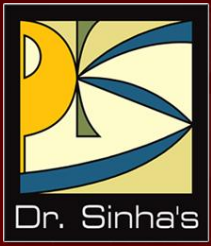
- For applications that are based on random data processing
- As a shared on-line secondary storage device. Winchester disks and disk packs are often used for this purpose
- As a backup device for off-line storage of data. Floppy disks, zip disks, and disk packs are often used for this purpose

*(Continued on next slide)*

# Uses of Magnetic Disks



- 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. Zip disks are often used for this purpose



# Optical Disks



# Optical Disk – Basics

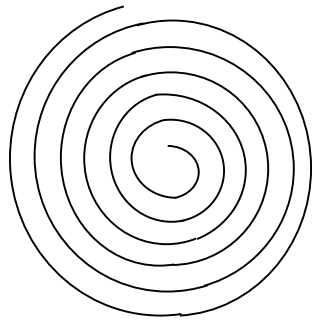


- Consists of a circular disk, which is coated with a thin metal or some other material that is highly reflective
- Laser beam technology is used for recording/reading of data on the disk
- Also known as laser disk / optical laser disk, due to the use of laser beam technology
- Proved to be a promising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

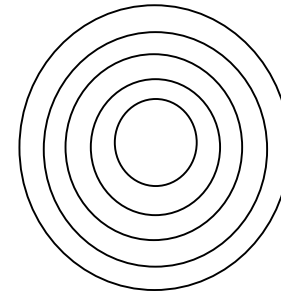
# Optical Disk – Storage Organization



- 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

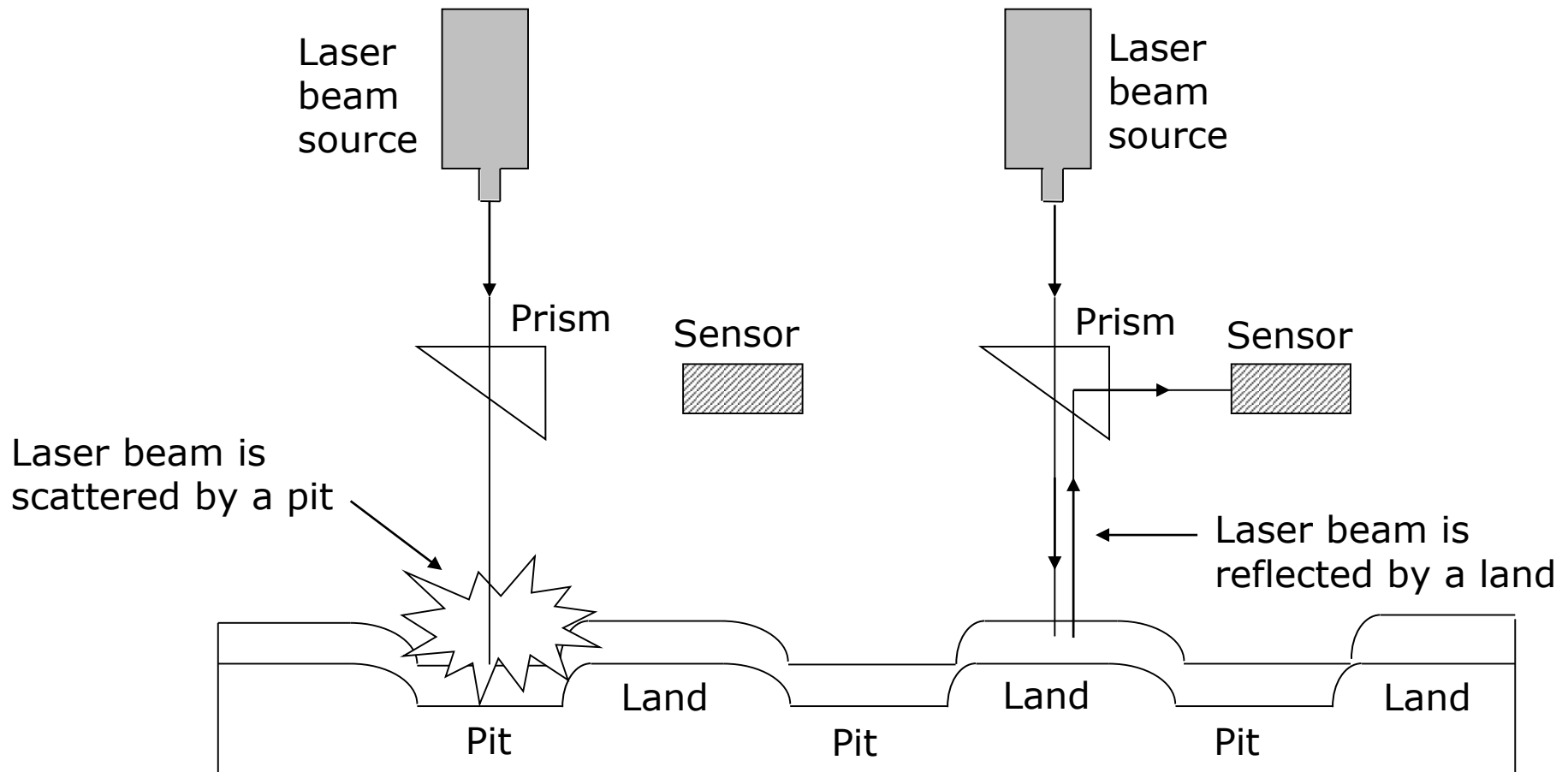


*Storage capacity of an optical disk*

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

The most popular optical disk uses a disk of 5.25 inch diameter with formatted 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

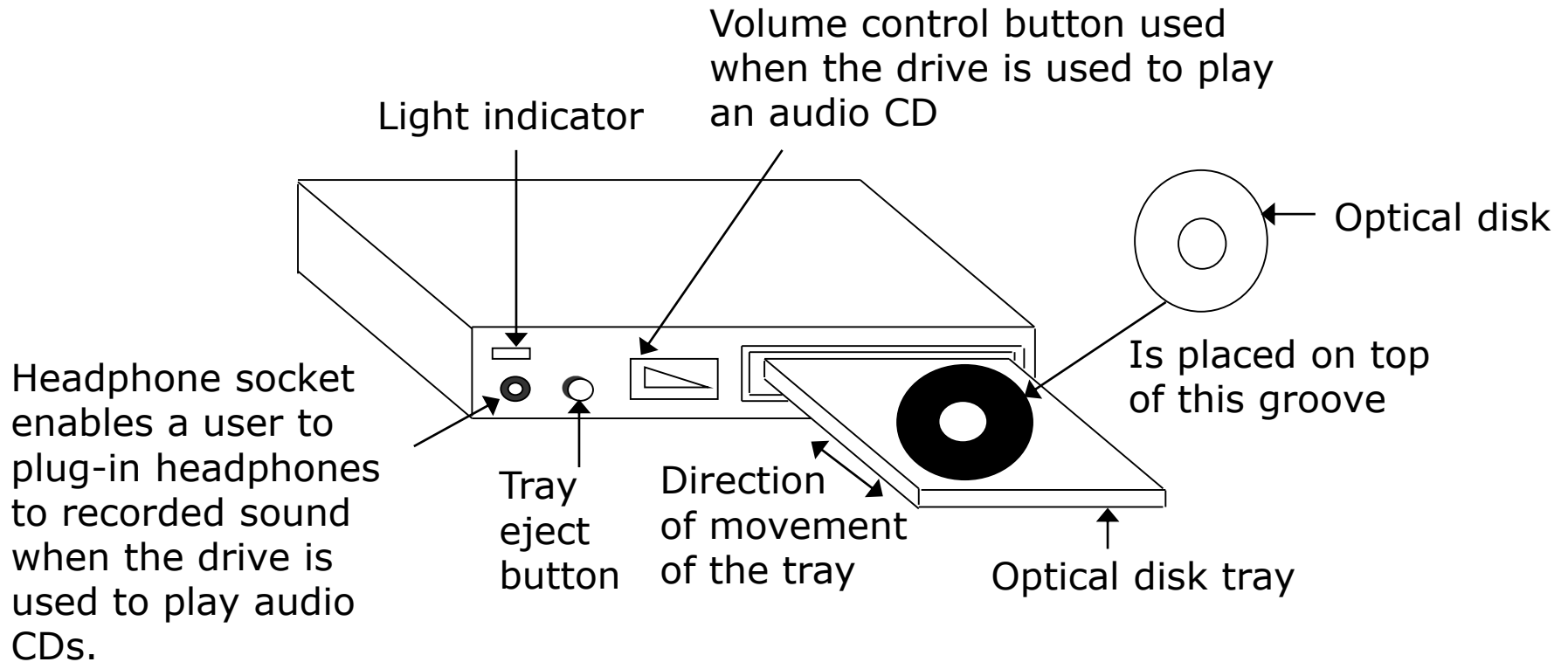


# Optical Disk Drive



- 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 5¼ 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



- 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

*(Continued on next slide)*

# 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



- 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

*(Continued on next slide)*

# 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

*(Continued on next slide)*

# Types of Optical Disks



## ■ Digital Video (or Versatile) Disk (DVD)

- DVD is a standard format for distribution and interchange of digital content
- Pits are about 4½ times as dense on a DVD as on a CD
- Stores about seven times more data per side
- Greater density is due to a more efficient data modulation scheme and error correction method
- DVD follows Eight-to-Fourteen Modulation Plus (EFMPlus) encoding
- Two variants of DVD – single-layer disk and double-layer disk
- Single-layer disk has storage capacity of 4.7 GB
- Double-layer disk has storage capacity of 8.5 GB *(Continued on next slide)*



# Types of Optical Disks



- Physical layer specification defines following types of physical media:
  - *DVD-ROM*: Used for mass distribution of pre-recorded software programs and multimedia
  - *DVD-RAM*: Used for general read-and-write applications
  - *DVD-R*: Used for low-cost, write-once recordable media
  - *DVD-RW*: Rewritable version of DVD-R

(Continued on next slide)

# Types of Optical Disks



- Logical layer specification defines recording formats for video and audio data
  - **DVD-video**
    - Now the most dominant movie storage format
    - Allows storage of video in 4:3 or 16:9 aspect ratio in MPEG-2 video format
    - Audio is usually Dolby Digital (AC-3) or Digital Theater System (DTS)
    - Can be either monaural or 5.1 surround sound
    - It has multiple selectable language soundtracks and subtitles

*(Continued on next slide)*

# Types of Optical Disks



- **DVD-audio**
  - Offers multiple choices of sampling rate and number of bits per sample
  - Specification also supports up to six channels of multichannel
  - Surround sound with 24-bit sampling at a 48 KHz rate
  - Provision for visual menus, still images, and video to accompany the audio program

# Advantages of Optical Disks



- Cost-per-bit of storage for optical disks is very low
- Use of a single spiral track makes optical disks an ideal storage medium for reading large blocks of sequential data
- Optical disk drives do not have any mechanical read/write heads
- Optical disks have data storage life in excess of 30 years
- Data once stored on CD-ROM/WROM disks becomes permanent, there is no danger of losing stored data
- Due to their compact size and lightweight, optical disks are easy to handle
- Can use a computer having a CD-ROM drive, a sound card, and speakers as a music system
- Can use a computer having a DVD drive to play DVDs

# Limitations of Optical Disks

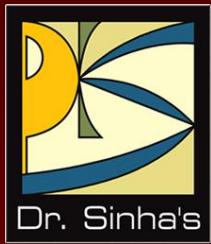


- Data access speed of optical disks is slower than that of magnetic disks
- Optical disks require more complicated drive mechanism
- Optical disk is prone to scratches, dust, sticky prints etc. while handling
- Optical disks must be labeled properly

# 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 audio such as songs
- For distribution of new or upgraded versions of software products
- For storage and distribution of wide variety of multimedia applications
- For archiving of data not used frequently
- To make permanent storage of proprietary information
- DVDs have become a popular medium for distribution of movies



# Memory Storage Devices



# Memory Storage Devices



- A new breed of electronic secondary storage devices
- They are faster and more reliable than conventional secondary storage devices because they do not use any mechanical component
- Some popular ones are:
  - Solid State Drive (SSD)
  - Flash Drive (Pen Drive)
  - Memory Card (SD/MMC)



# Solid State Drive (SSD)



- Is a new alternative to Hard Disk Drive (HDD) based secondary storage in computer systems
- It consists of interconnected flash memory chips
- It is built entirely out of semiconductors (hence the name solid state) and does not have any moving parts
- It is more expensive than HDD of same capacity, but is more reliable, rugged, compact and faster in data access speed

# Inside View of HDD and SSD



(a) Hard Disk Drive (HDD)



(b) Solid State Drive (SSD)

**Figure 8.16.** Inside view of HDD and SSD. When packaged in an enclosure, both may look similar from outside, but internally their design is very different.

# Memory Storage Devices



## Flash Drive (Pen Drive)

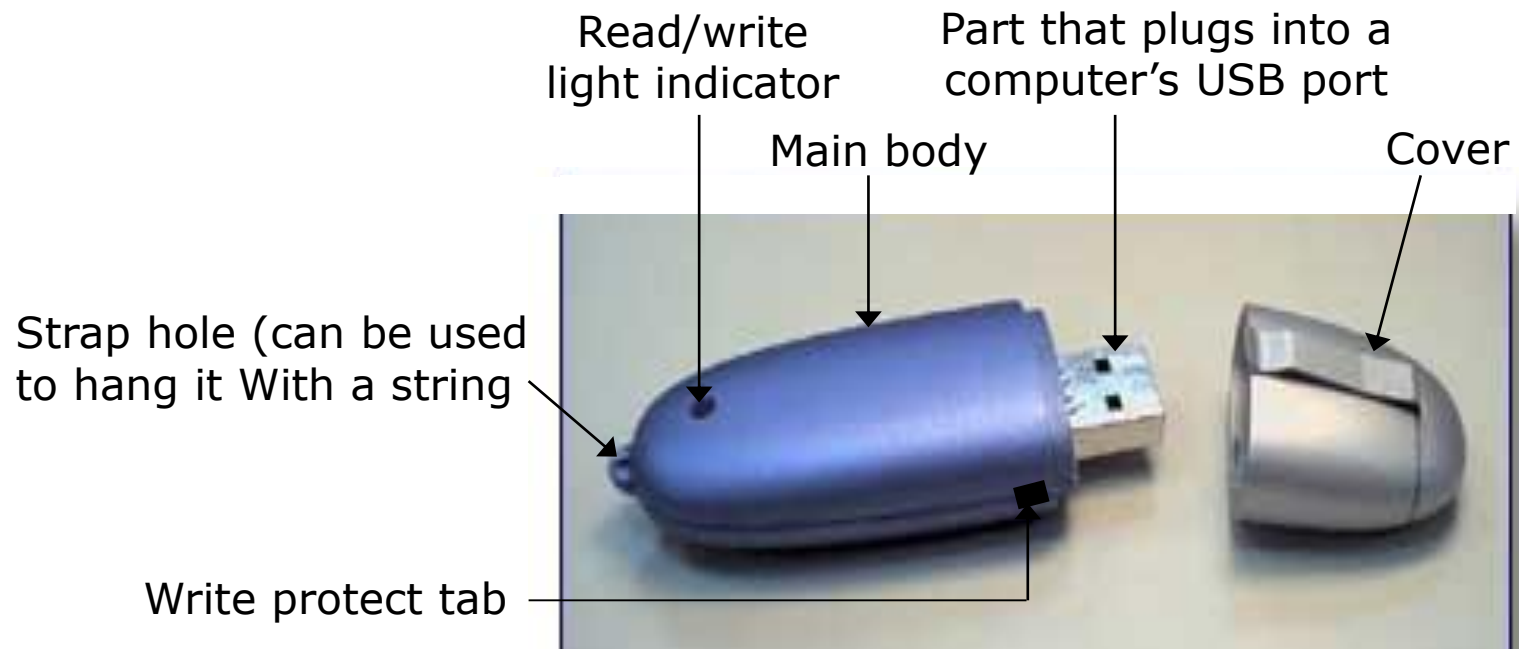
- Flash drive is a compact device of the size of a pen
- Comes in various shapes and stylish designs
- May have different added features
- It is a plug-and-play device
- Plugs into a USB (Universal Serial Bus) port
- Computer detects it automatically as a removable drive
- Once done, it can be simply plugged out of the USB port
- Flash drive does not require any battery, cable, or software
- It is the most preferred external data storage

# Memory Storage Devices



- It is based on flash memory storage technology
- Flash memory is non-volatile, Electrically Erasable Programmable Read Only Memory (EEPROM) chip
- It is a highly durable solid-state storage having data retention capability of more than 10 years

# A Flash Drive (Pen Drive)



# Memory Storage Devices



## Memory Card (SD/MMC)

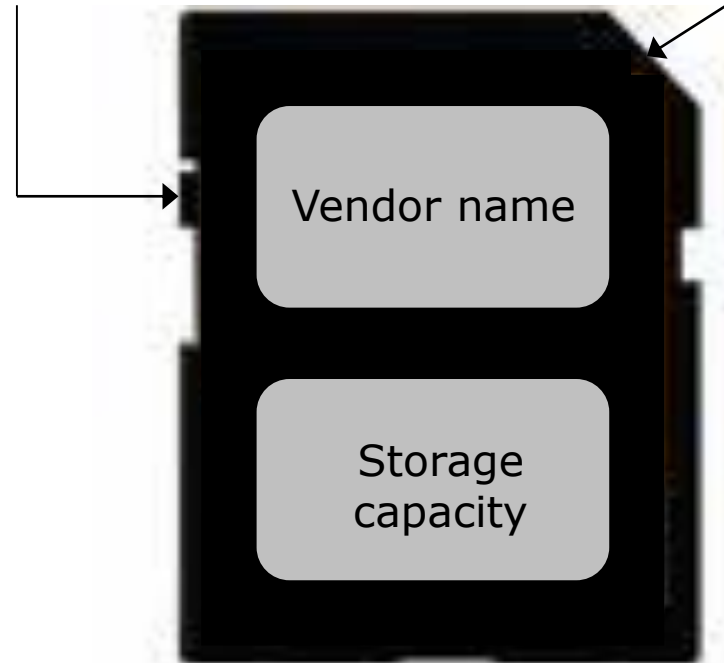
- Flash memory based cards are available as removable storage device in different types of electronic equipment
- The most popular ones are Secure Digital (SD) and Multimedia Card (MMC)
- These cards are used in various types of digital devices
- Each of these cards has its own interface and specific design features

# Flash Memory Based Card



Write-protect lock  
(slides down/up to lock/  
unlock write operation)

Direction notch  
(guides which side of the  
card should be inserted first  
in the electronic equipment)

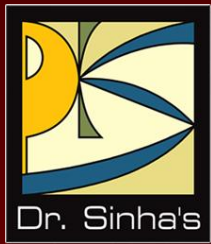


# Hybrid Secondary Storage Drives



- Both SSD and HDD coexist in such systems
- The two commonly used configurations are:
  - Dual-drive system
  - SSD as cache and HDD as normal secondary storage





# Mass Storage Devices



# 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



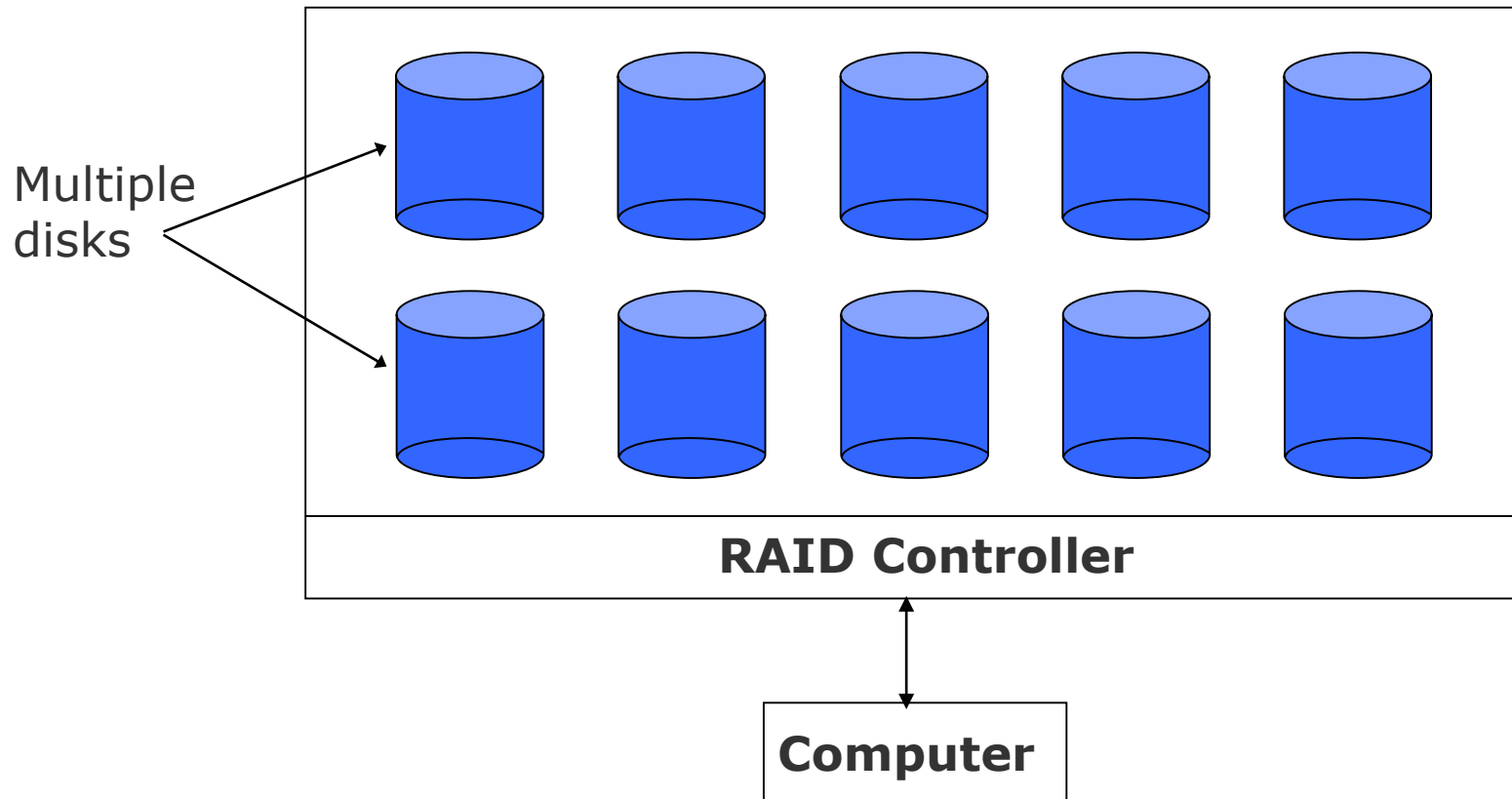
- Set of hard disks and hard disk drives with a controller mounted in a single box, forming a single large storage unit
- It is commonly known as a *RAID (Redundant Array of Inexpensive Disks)*
- As a secondary storage device, provides enhanced storage capacity, enhanced performance, and enhanced reliability

# 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

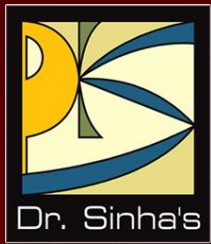


- Set of magnetic tapes and magnetic tape drives with a controller mounted in a single box, forming a single large storage unit
- Large tape library can accommodate up to several hundred high capacity magnetic tapes bringing the storage capacity of the storage unit to several terabytes
- Typically used 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



# Data Backup





# What is Data Backup?



- Data backup is the process of creating a copy of some data from an on-line storage device to a secondary storage backup device
- Copy is used for off-line storage
- It can be retrieved from the backup device and stored back on the on-line storage device

# Why Backup Data?



- On-line storage device can be damaged or lost in any of the following ways
  - A disk crash
  - A virus attack
  - A hardware malfunction
  - An unintended accidental deletion of useful files
  - A natural disaster like fire or earthquake damaging the computer system
- Amount of time it would take us to re-create all files is far greater than the few minutes it takes to back them up

# Types of Backup



## ■ Full backup

- All data on an on-line secondary storage device is copied on a backup device at the time of backup
- Full backup is simpler and a little safer
- To restore a particular file we need to search for it at only one place

## ■ Incremental backup

- Only newly created files or files that have been changed since the last backup are copied on backup device from on-line secondary storage device at the time of backup
- Incremental backup is faster than full backup
- To restore a particular file we need to search for it on several backups

# Backup Policy



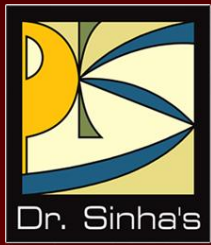
- What is the periodicity of backup?
  - Normally, periodicity depends on the nature of usage and criticality of stored data
- Whether to take full or incremental backup?
  - If a user creates only a few files, daily/weekly incremental backup is sufficient
  - If a user frequently creates many files, weekly full backups are safer
- What storage media to use for backup?
  - Small backups are taken normally on CD/pen drives
  - Whereas larger backups are taken on magnetic tapes

*(Continued on next slide...)*

# Backup Policy



- Who takes backup?
  - In case of PCs, it is the PC user
  - In case of large computer centers, normally the administrators of the system
- Where to store the backup media with the backed up data?
  - It is suggested to store the backup media in a building that is away from the building where backup was taken to protect the data against natural calamities



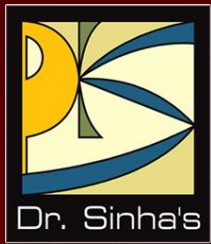
# On-line, Near-line, and Off-line Storage



# On-line, Near-line, and Off-line Storage



- **On-line storage**
  - On-line storage device is under direct control of the processing unit of the computer
- **Near-line storage**
  - Near-line storage device is an intermediate storage between on-line storage and off-line storage
- **Off-line storage**
  - Off-line storage is not under direct control of any computer

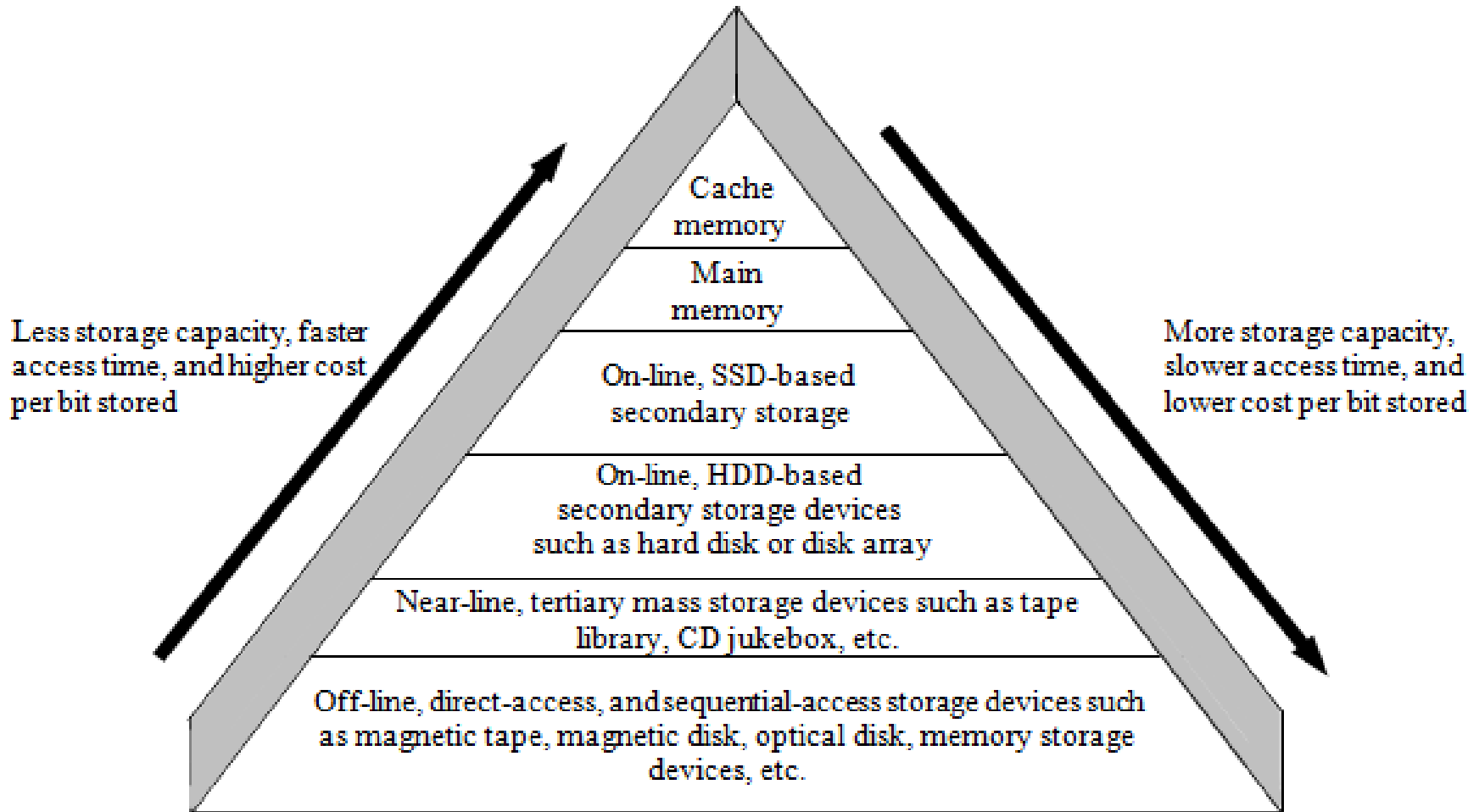


# Hierarchical Storage System (HSS)





# Hierarchical Storage System (HSS)



# Hierarchical Storage Management (HSM)



- HSM software is an integral part of an HSS
- It automatically moves the more frequently used data (files) to higher layers of the HSS (having faster access time) from its lower layers (having slower access time)
- For this, it continuously monitors the data access pattern by the users of HSS

# Hierarchical Storage Management (HSM)



- Important issue in HSS is how to decide which data should reside on which storage tier
- This is handled automatically by a software called Hierarchical Storage Management (HSM)
- All HSSs have HSM software, which monitors the way data is used, and makes decisions as to which data can be moved safely to slower devices
- Frequently used data are stored on on-line storage
- Moved eventually by HSM to near-line storage if they are not accessed for a certain period
- Users do not need to know or keep track of where the accessed data is stored and how to get it back
- Computer retrieves the data automatically

# Key Words/Phrases



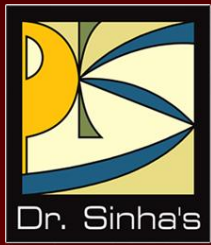
- Automated tape library
- Auxiliary memory
- Block
- Blocking
- Blocking factory
- CD-ROM
- CD-ROM jukebox
- Check bit
- Cylinder
- Data transfer rate
- Direct access device
- Disk array
- Disk controller
- Disk drive
- Disk formatting
- Disk pack
- DVD
- Even parity
- File Allocation Tube (FAT)
- Floppy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

*(Continued on next slide)*

# Key Words/Phrases



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



# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 9

## Input-Output Devices



# 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

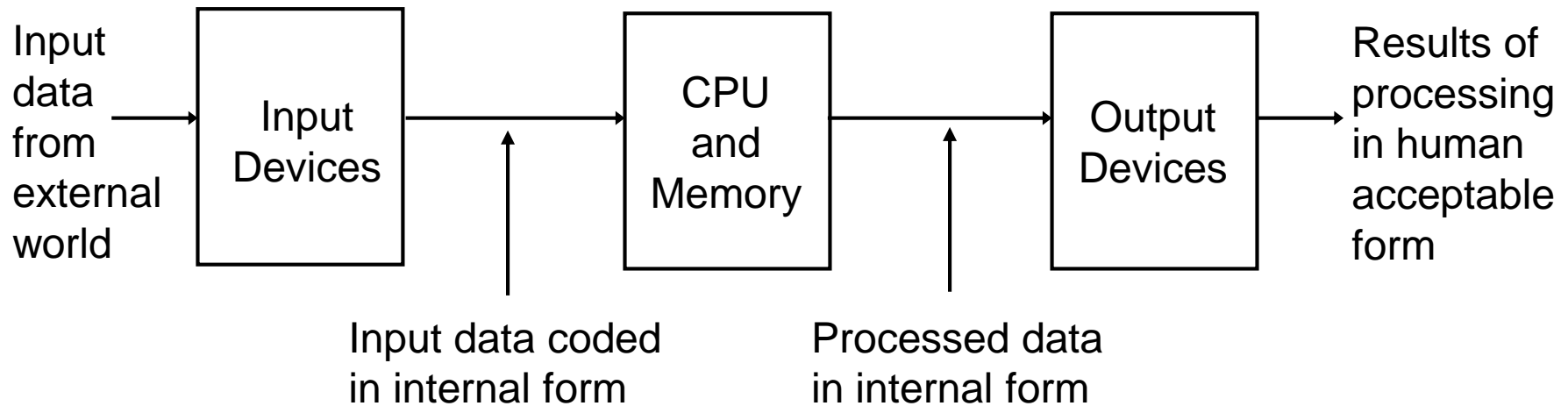
# I/O Devices

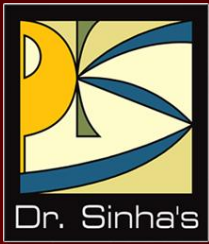


- Provide means of communication between a computer and outer world
- Also known as peripheral devices because they surround the CPU and memory of a computer system
- Input devices are used to enter data from the outside world into primary storage
- Output devices supply results of processing from primary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



# Role of I/O Devices





# Input Devices



# Commonly Used Input Devices



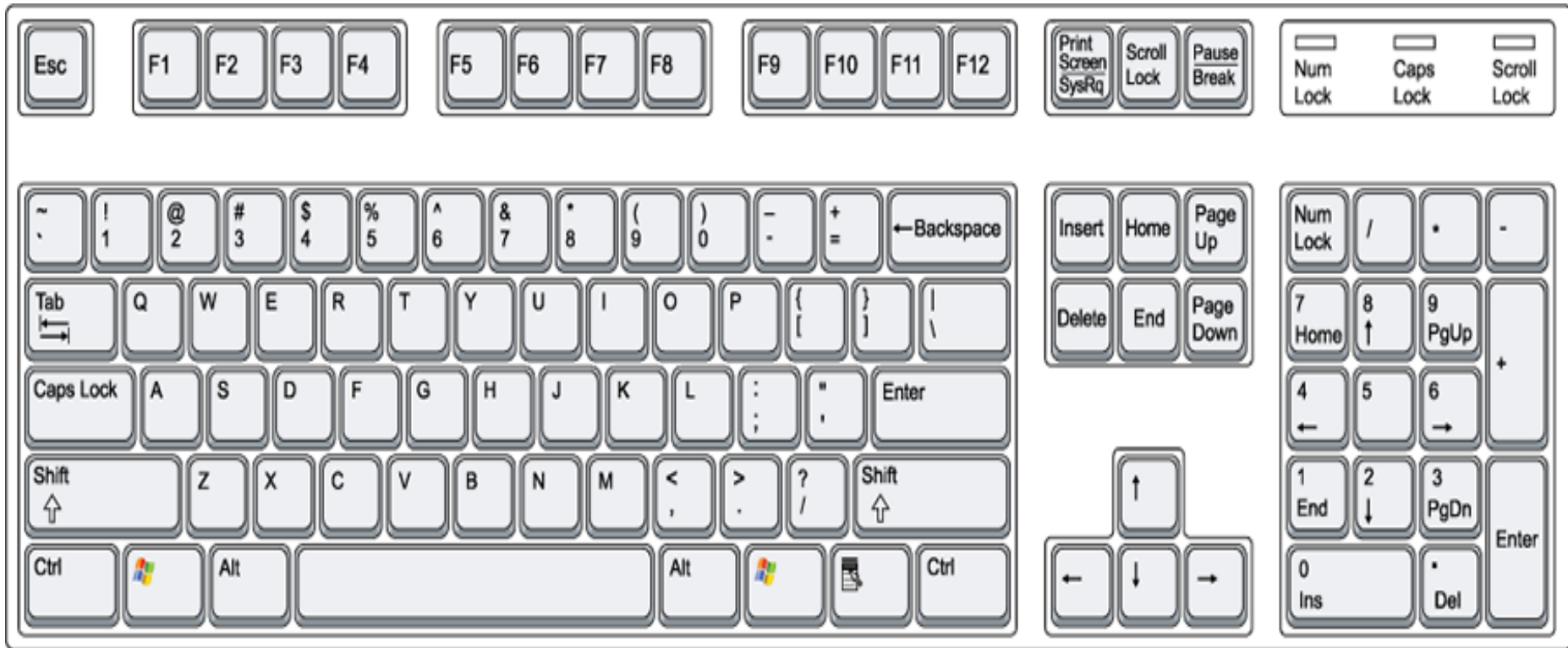
- Keyboard devices
- Point-and-draw devices
- Data scanning devices
- Digitizer
- Electronic cards based devices
- Speech recognition devices
- Vision based devices

# 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



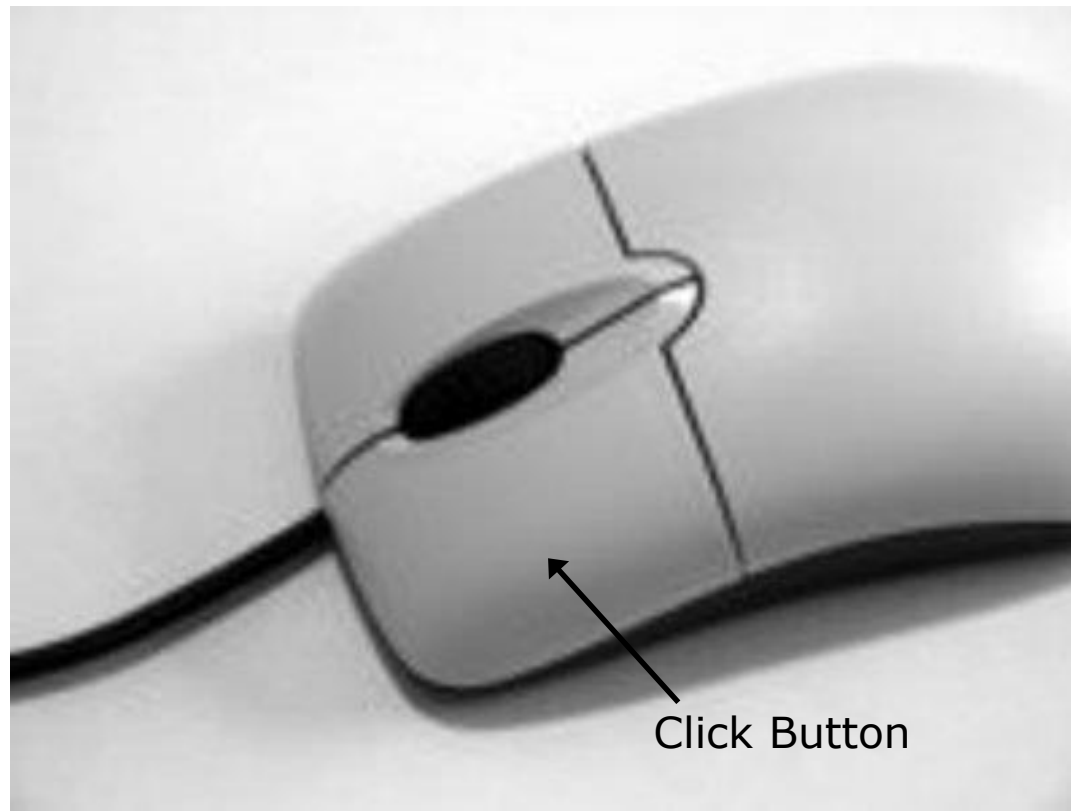
- Used to rapidly point to and select a graphic icon or menu item from multiple options displayed on the *Graphical User Interface (GUI)* of a screen
- Used to create graphic elements on the screen such as lines, curves, and freehand shapes
- Some commonly used point-and-draw devices are mouse, track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

# Mouse



- Mouse is the most popular point-and-draw device
- Mouse is a small hand-held device that fits comfortably in a user's palm
- It rolls on a small bearing and has one or more buttons on the top
- When a user rolls a mouse on a flat surface, a *graphics cursor* moves on the terminal screen in the direction of the mouse's movement
- Different applications display the graphics cursor as different symbols
- Graphics cursor, irrespective of its size and shape, has a pixel-size point that is the point of reference to decide the position of the cursor on the screen. This point is called *hot spot*

# Mouse





# Types of Mouse



## ■ Mechanical mouse

- Mechanical mouse has a ball inside it that partially projects out through an opening in its base
- Ball rolls due to surface friction when the mouse is moved on a flat surface
- On two sides of the ball are two small wheels that spin to match the speed of the ball. Each wheel of the mouse is connected to a sensor
- As the mouse ball rolls when a user moves the mouse, the sensors detect how much each wheel spins and send this information to the computer in the form of changes to the current position

*(Continued on next slide...)*

# Types of Mouse



## ■ **Optical mouse**

- An optical mouse has no mechanical parts like the ball and wheels
- It has a built-in photo-detector
- When a user moves the mouse on a special pad with gridlines, the photo-detector senses each horizontal and vertical line on the pad, and sends this information to the computer in the form of changes to the current position

## ■ **One, Two, and Three buttons mouse**

- Mouse can have one, two, or three buttons
- With a mouse having multiple buttons, the leftmost button is the main button that allows for most mouse operations
- A user can configure another button as main button

*(Continued on next slide...)*

# Types of Mouse



- **Serial and bus mouse**

- A serial mouse plugs into a serial port
- A bus mouse requires a special electronic card, which provides a special port just for connecting the mouse to a computer

- **Wired and cordless mouse**

- Wired mouse is connected to a computer with a small cord
- A cordless mouse operates by transmitting a low-intensity radio or infrared signal

# Trackball



- A trackball is a pointing device similar to a mechanical mouse
- Roller ball is placed on the top along with the buttons
- We have to roll the ball with hand
- Trackball requires less space than a mouse for operation
- Trackball is a preferred device for CAD/CAM applications

# Trackball



# Joystick



- Joystick is a pointing device that works on the same principle as a trackball
- To make the movements of the spherical ball easier, it is placed in a socket with a stick mounted on it
- User holds the stick in his/her hand and moves it around to move the spherical ball
- User can move the stick forward or backward, left or right, to move and position the graphics cursor at a desired position
- Joysticks use potentiometers to sense stick and ball movements
- A button on top of the stick enables a user to select the option pointed to by the cursor

# Joystick



# Electronic Pen



## ▪ **Light pen**

- Uses a photoelectric cell and an optical lens mounted in a pen-shaped case
- It focuses on to it any light in its field of view
- It detects the light emitted from a limited field of view of the monitor's display
- System transmits this electric response to a processor, which identifies the menu item or icon that is triggering the photocell
- Pen has a finger-operated button

## ▪ **Writing pen with pad**

- This type of electronic pen comes with a special type of writing pad
- User writes on the pad with the electronic pen whatever data he/she wants to input to the computer
- This input device with handwriting recognition software is used often as an easy way to input text and freehand drawings into computer



# 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



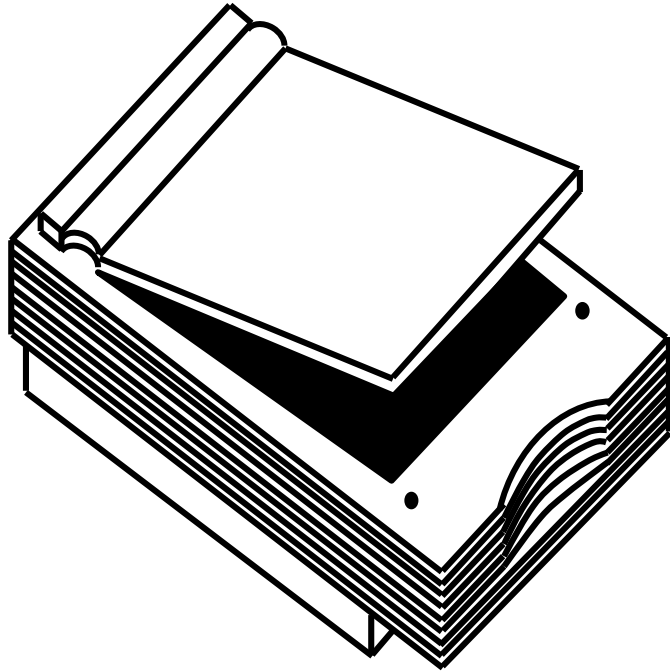
- Input devices that enable direct data entry into a computer system from source documents
- Eliminate the need to key in text data into the computer
- Due to reduced human effort in data entry, they improve data accuracy and also increase the timeliness of the information processed
- Demand high quality 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

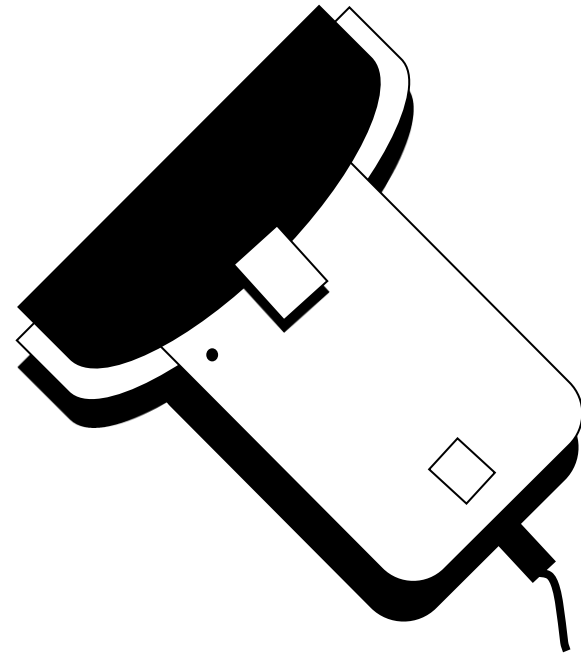


- Input device that translates paper documents into an electronic format for storage in a computer
- Electronic format of a scanned image is its bit map representation
- Stored image can be altered or manipulated with an image-processing software

# 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

Indicates direction in which the sheet should be fed to the OMR

1.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
	a	b	c	d
2.	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
	a	b	c	d
3.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	a	b	c	d

(b) Pre-printed answer sheet

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

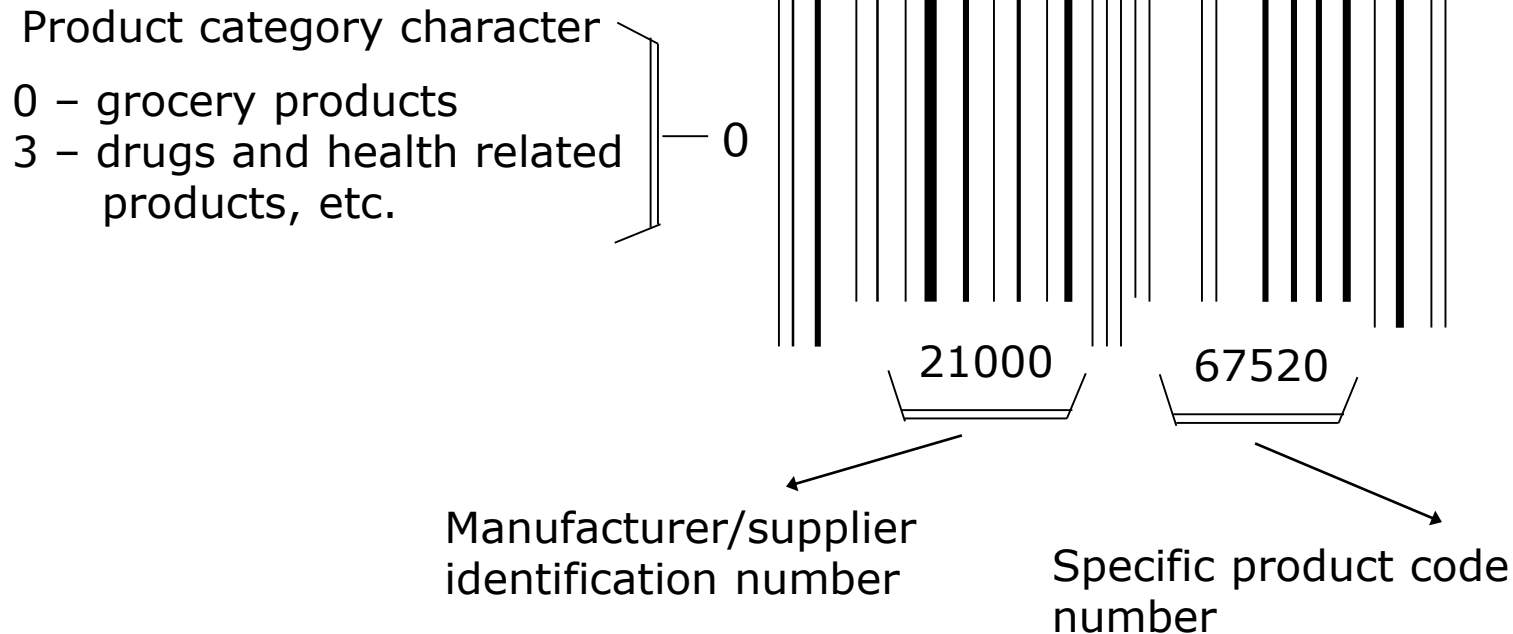
# Bar-code Reader



- 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

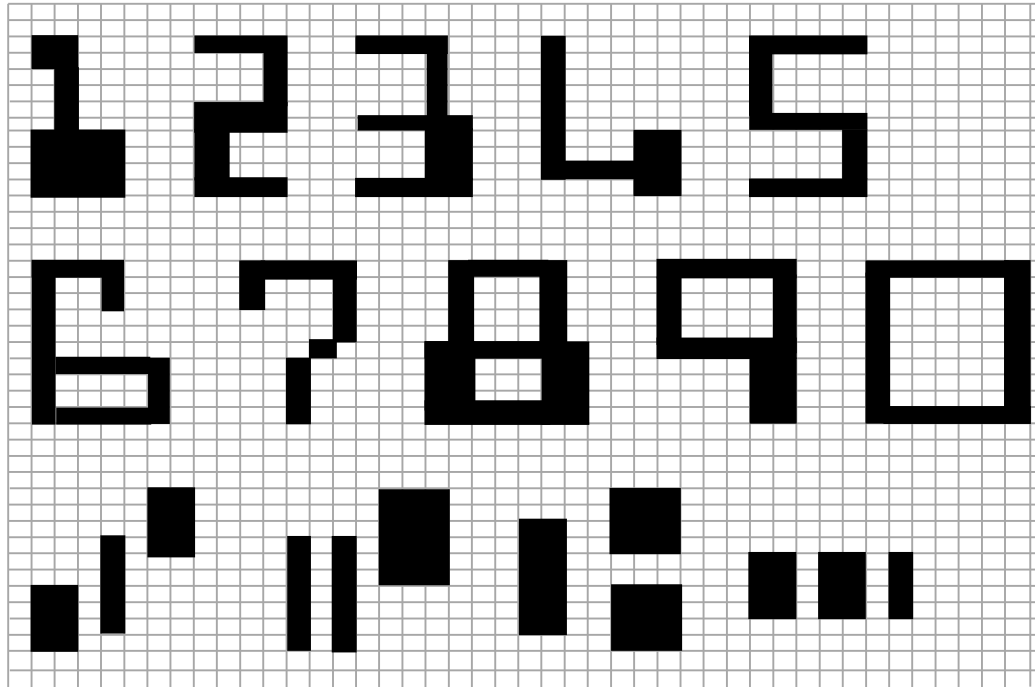


# Magnetic-Ink Character Recognition (MICR)



- 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

# 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



- *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

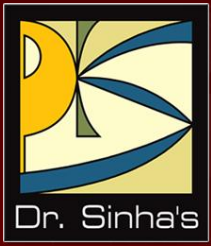


- For inputting data to a computer system by a person in situations where his/her hands are busy, or his/her eyes must be fixed on a measuring instrument or some other object
- For data input by dictation of long text or passage for later editing and review
- For authentication of a user by a computer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



# Output Devices



# 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**

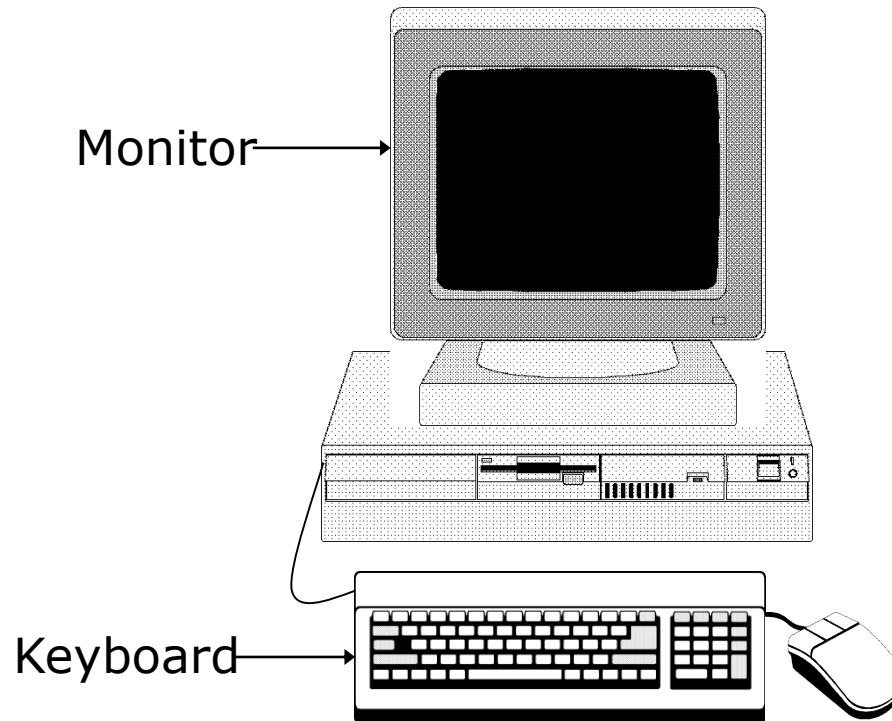
- Produced on a paper or some material that can be touched and carried for being shown to others
- Permanent in nature and can be kept in paper files or can be looked at a later time when the person is not using the computer
- Examples are output produced by printers or plotters on paper

# 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
- LCD (Liquid Crystal Display) 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.



# CRT and LCD Monitors



CRT Monitor



LCD Monitor

# Printers



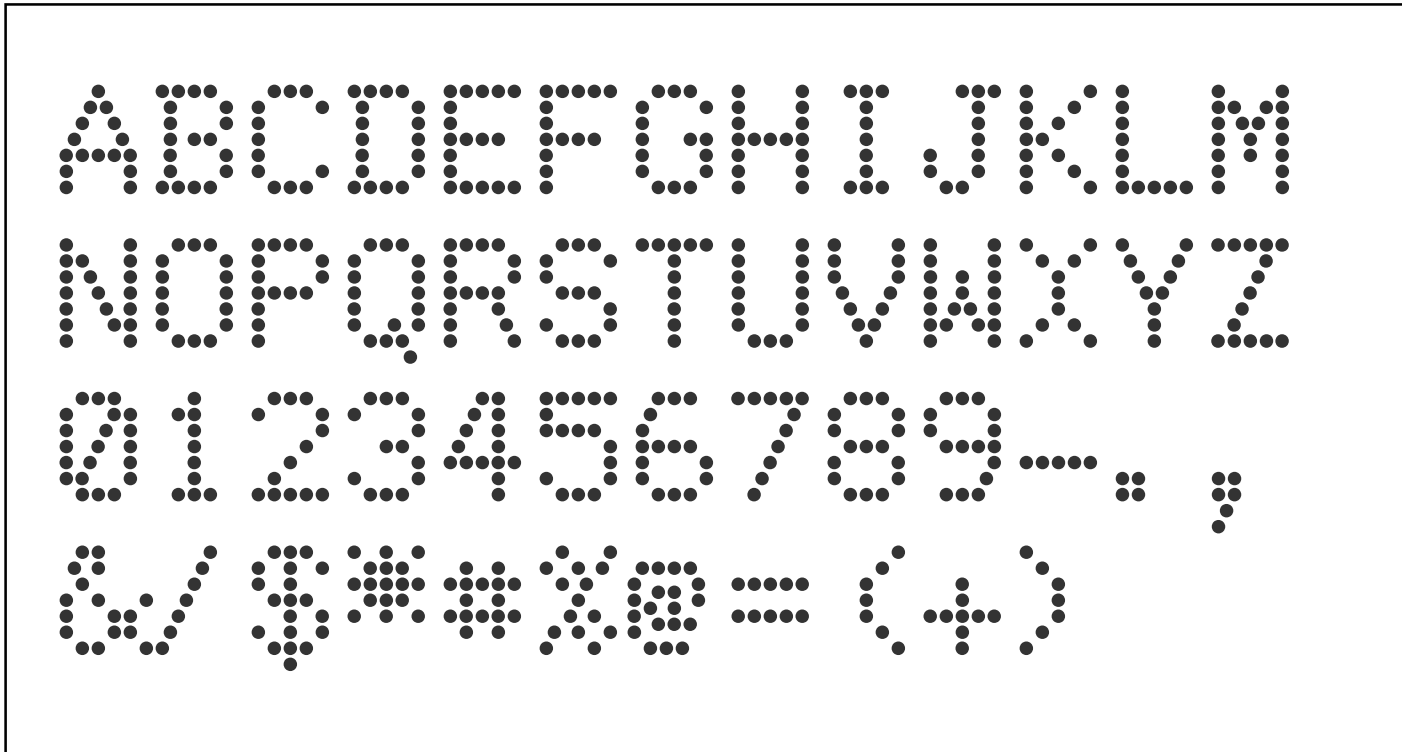
Most common output devices for producing hard-copy output

# Dot-Matrix Printers

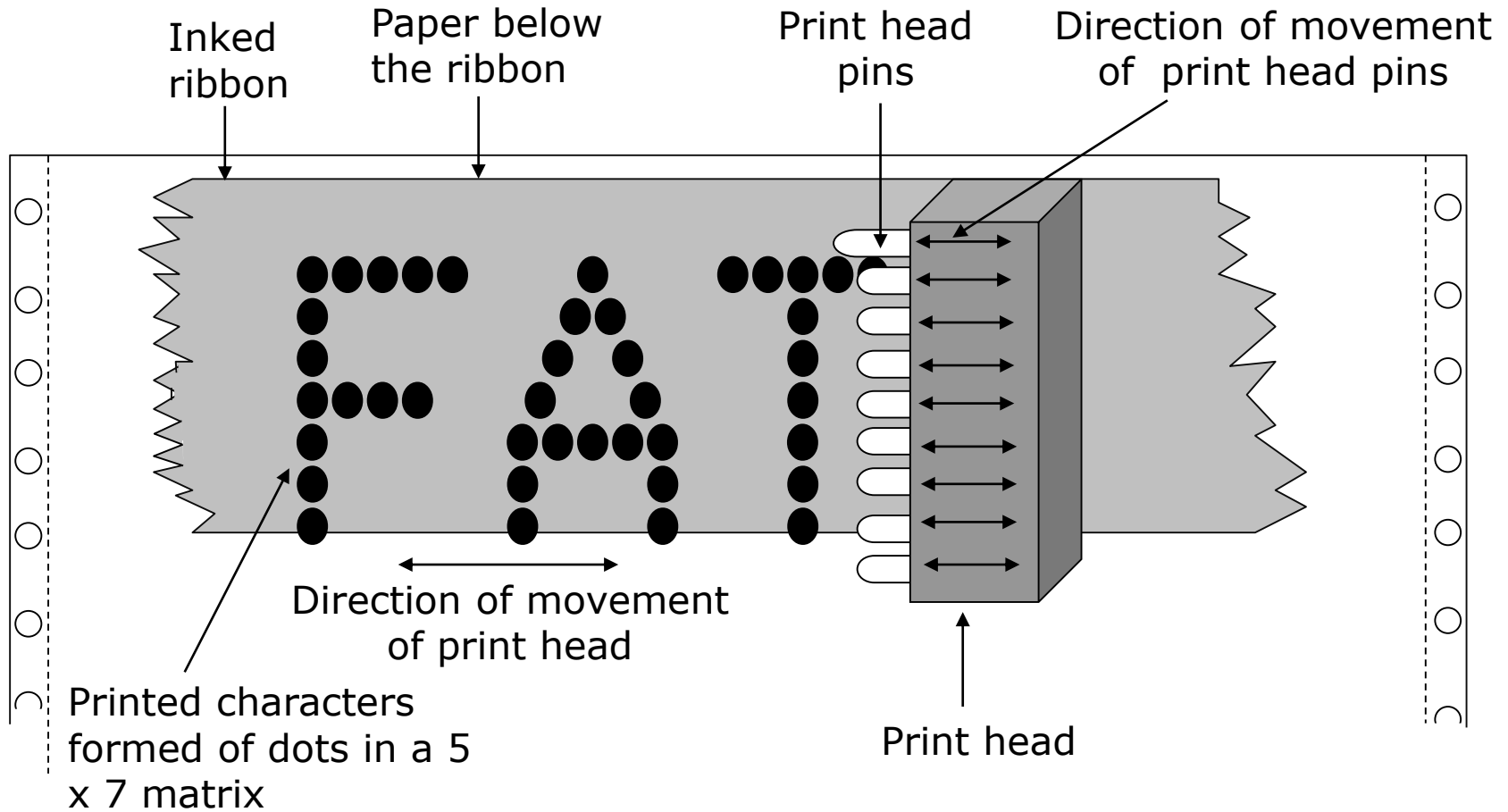


- Character printers that form characters and all kinds of images as a pattern of dots
- Print many special characters, different sizes of print and graphics such as charts and graphs
- Impact printers can be used for generating multiple copies by using carbon paper or its equivalent
- Slow, with speeds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

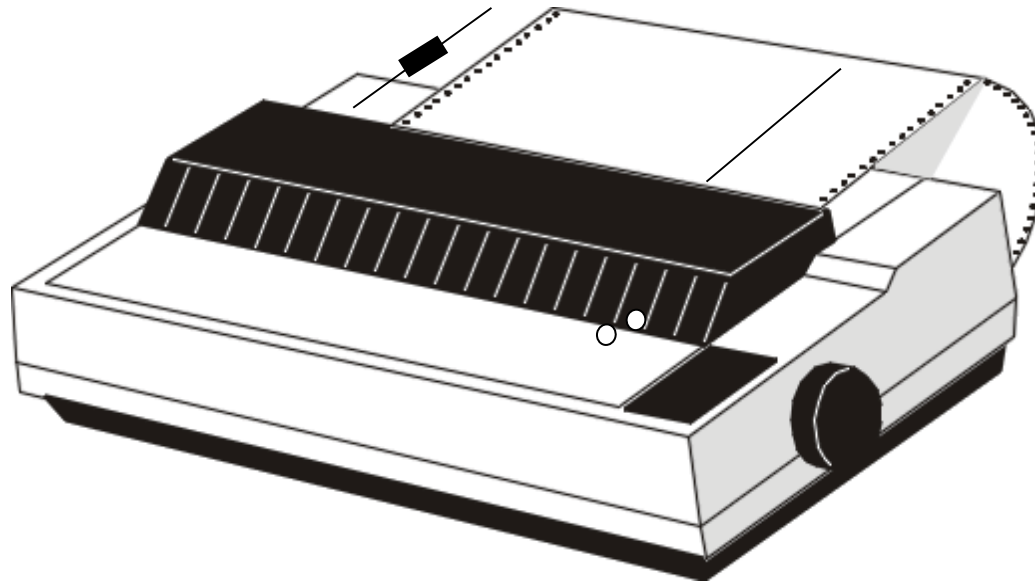
# Formation of Characters as a pattern of dots



# Dot Matrix Printer Printing Mechanism



# 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)*

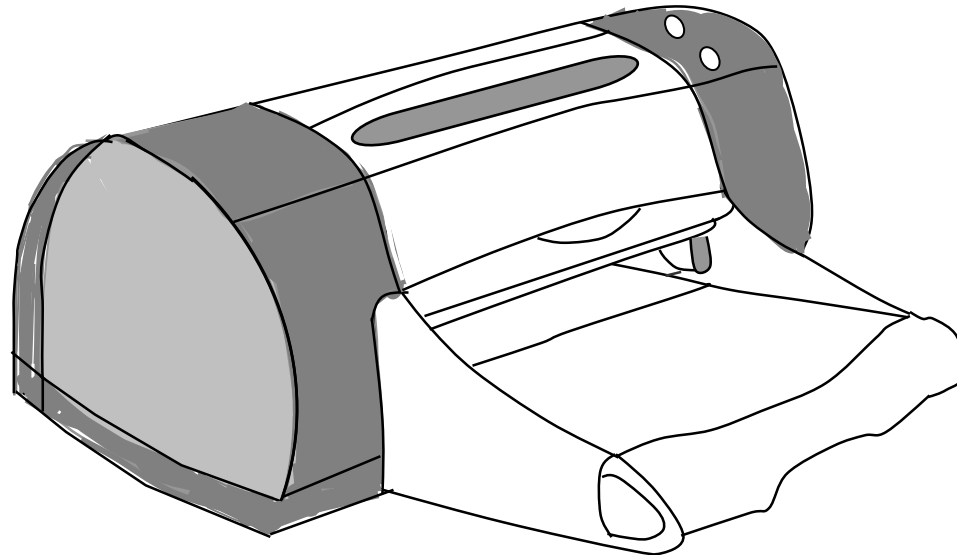
# Inkjet Printers



- 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 Printer

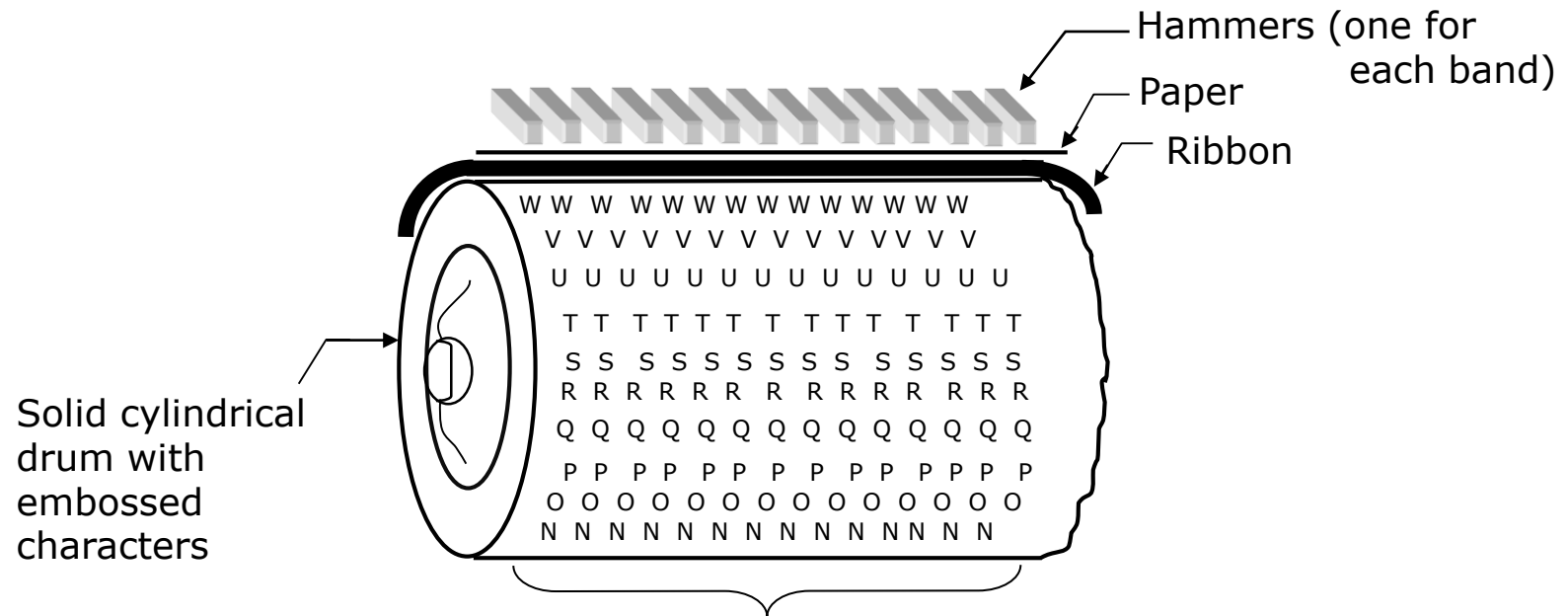


# Drum Printers



- 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



Total number of bands is equal to the maximum number of characters (print positions) on a line. Each band has all characters supported by the printer.

# Chain/Band Printers



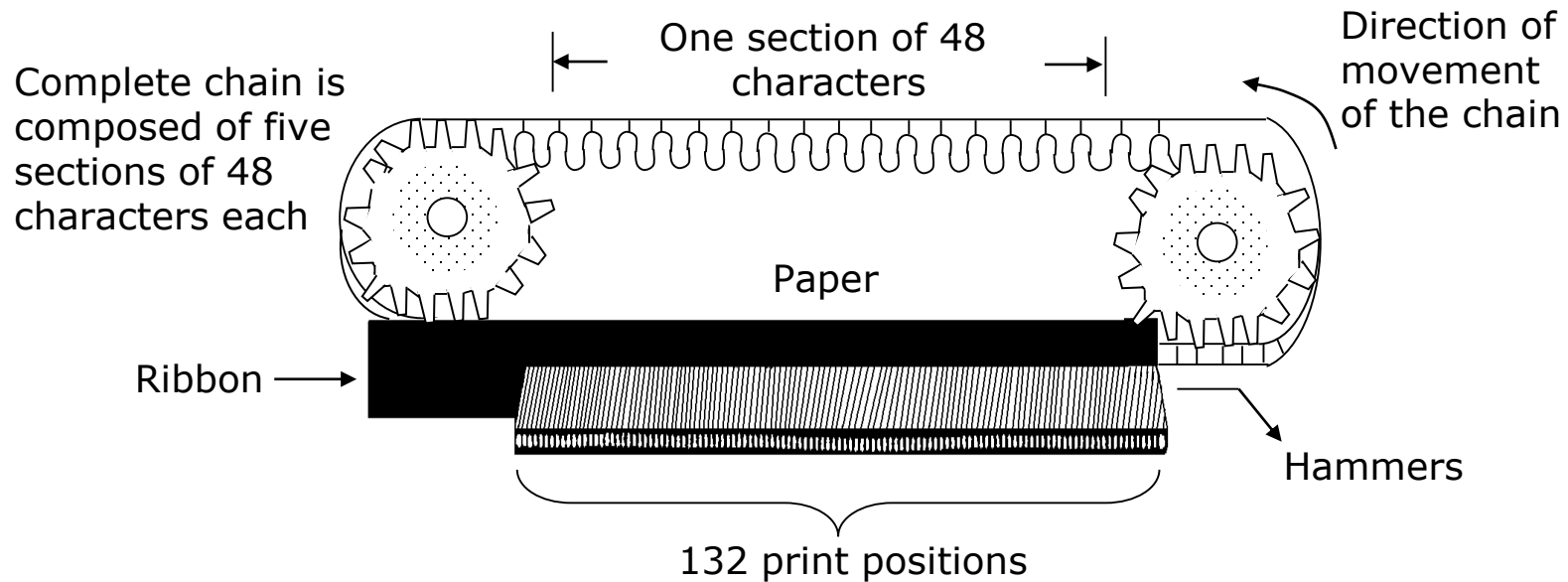
- Line printers that print one line at a time
- Consist of a metallic chain/band on which all characters of the character set supported by the printer are embossed
- Also have a set of hammers mounted in front of the chain/band in such a manner that an inked ribbon and paper can be placed between the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

# Printing Mechanism of a Chain/Band Printer



# 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 electrostatically 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)*

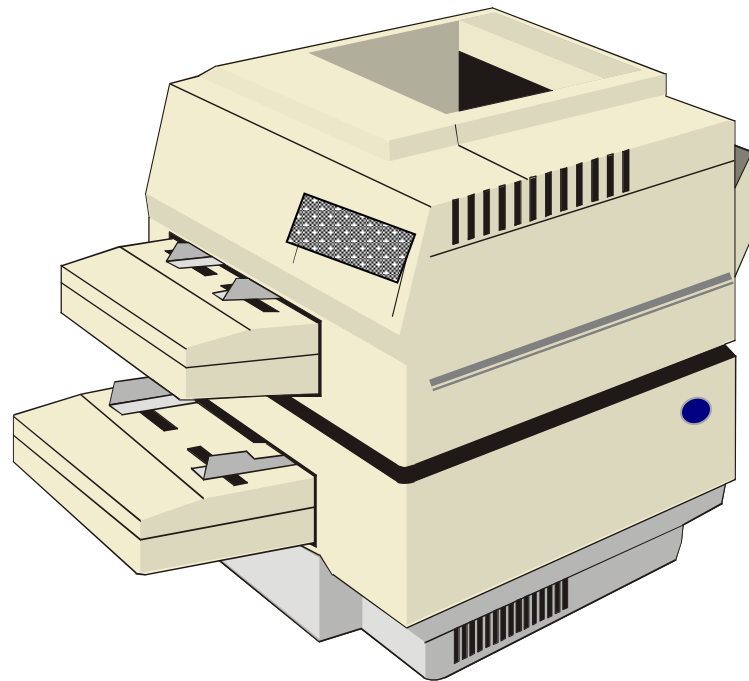
# Laser Printers



- 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 Printer

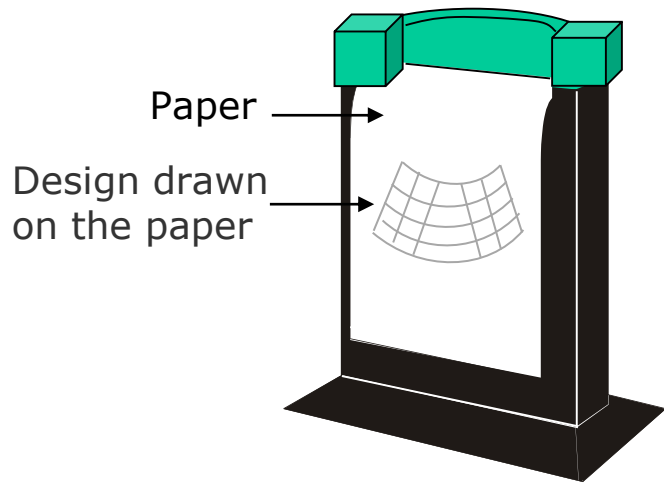


# Plotters

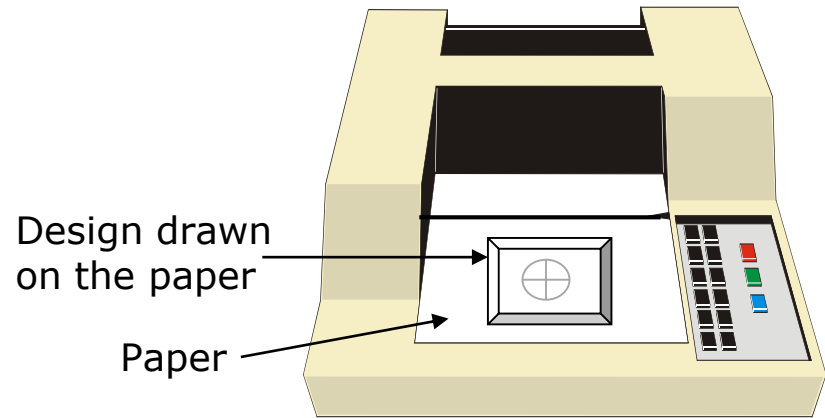


- Plotters are an ideal output device for architects, engineers, city planners, and others who need to routinely generate high-precision, hard-copy graphic output of widely varying sizes
- Two commonly used types of plotters are:
  - *Drum plotter*, in which the paper on which the design has to be made 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

# Plotters



**A drum plotter**



**A flatbed plotter**

# 3D Printers



- They print (create) 3D objects
- They use Additive Manufacturing technique
- They follow a 3-step process:
  1. First create a 3D digital file (3D model) of the object
  2. Then slice the 3D model into horizontal layers
  3. Then upload the sliced file and print

# Types of 3D Printers



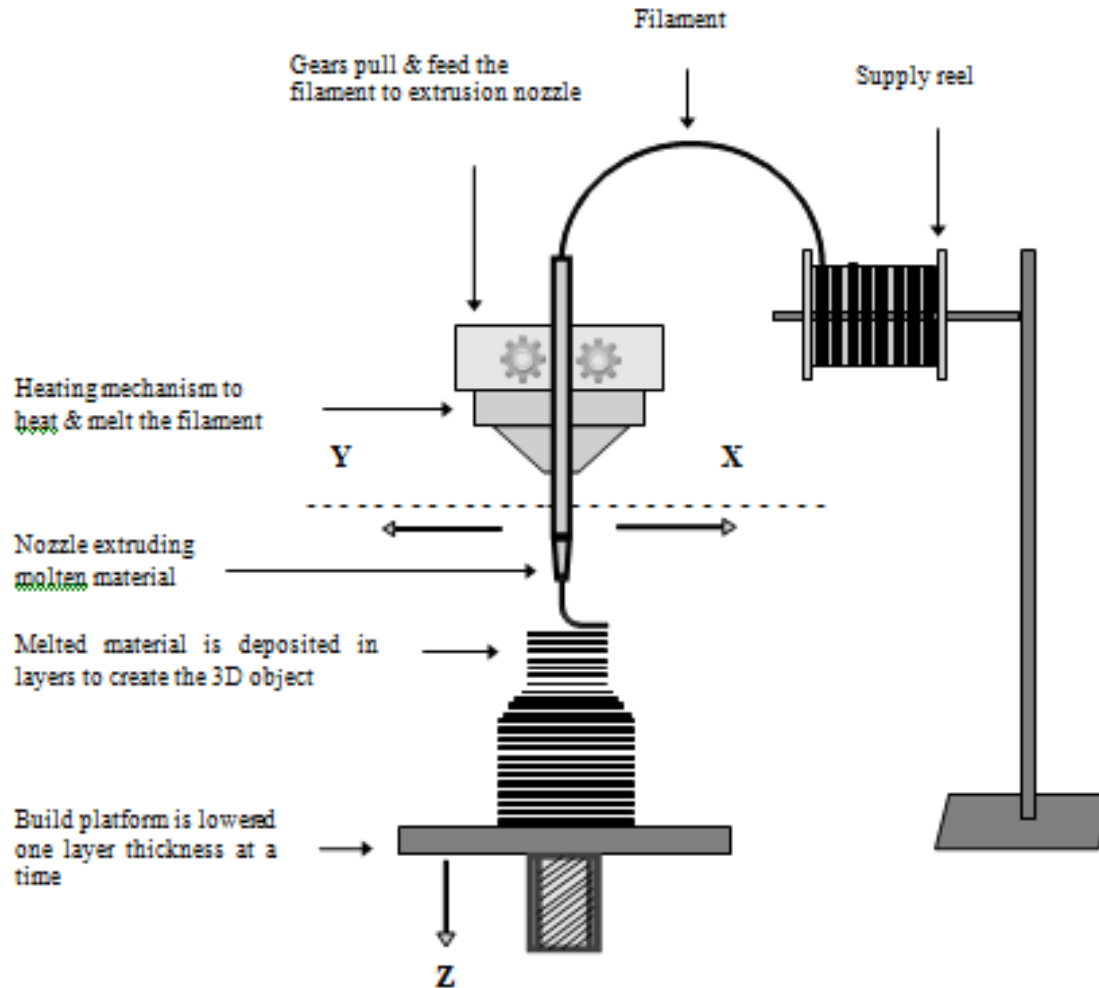
- 3D printers mainly differ in the way they build an object layer-by-layer and the material they use for manufacturing objects
- Some commonly used 3D printing technologies are:
  - Fused Deposition Modeling (FDM)
  - STereoLithography (STL)
  - Digital Light Processing (DLP)
  - Selective Layer Sintering (SLS)

# Applications of 3D Printers



- Rapid prototyping
- Rapid manufacturing of customized designs
- Manufacturing body/engine parts
- Manufacturing implants and prosthetics
- Manufacturing hearing aid and dental appliances
- Manufacturing museum souvenirs

# Fused Deposition Modeling (FDM)



# 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



# Screen Image Projector



# 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



- 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

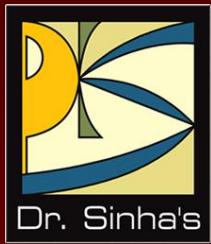


- 3D Printer
  - Bard code reader
  - Cathode Ray Tube (CRT)
  - Chain/Band printer
  - Data scanning device
  - Digital Light Processing (DLP)
  - Digitizer
  - Digitizing tablet
  - Dot-Matrix printer
  - Drum plotter
  - Drum printer
  - Electronic card reader
  - Electronic Pen
  - Flatbed plotter
  - Flatbed Scanner
  - Fused Deposition Modeling (FDM)
  - Graphical User Interface
  - Hand-held scanner
  - Hard-copy output
  - Image Scanner
  - Information Kiosk
  - Inkjet printer
  - Input/Output device
  - Joystick
  - Keyboard device
  - Laser printer
  - Magnetic-Ink Character Recognition (MICR)
  - Monitor
  - Mouse
  - Optical Character Recognition (OCR)
  - Optical Mark Reader (OMR)
  - Peripheral device
  - Phonemes
  - Plotter
  - Point-and-draw device
  - Printer
  - QWERTY keyboard
  - Screen Image Projector
- (Continued on next slide)*

# Key Words/Phrases



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



# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 10

**Computer Software**



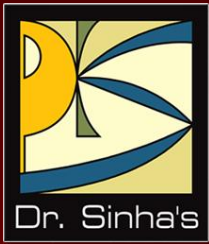
# Learning Objectives



## 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
- Software Engineering and CASE tools
- Firmware and Middleware





# Software, Firmware and Middleware



# Software



- **Hardware** refers to the physical devices of a computer system.
- **Software** refers to a collection of programs, procedures, and associated documents describing the programs, and how they are to be used.
- **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 (for example, word-processing package)

# 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



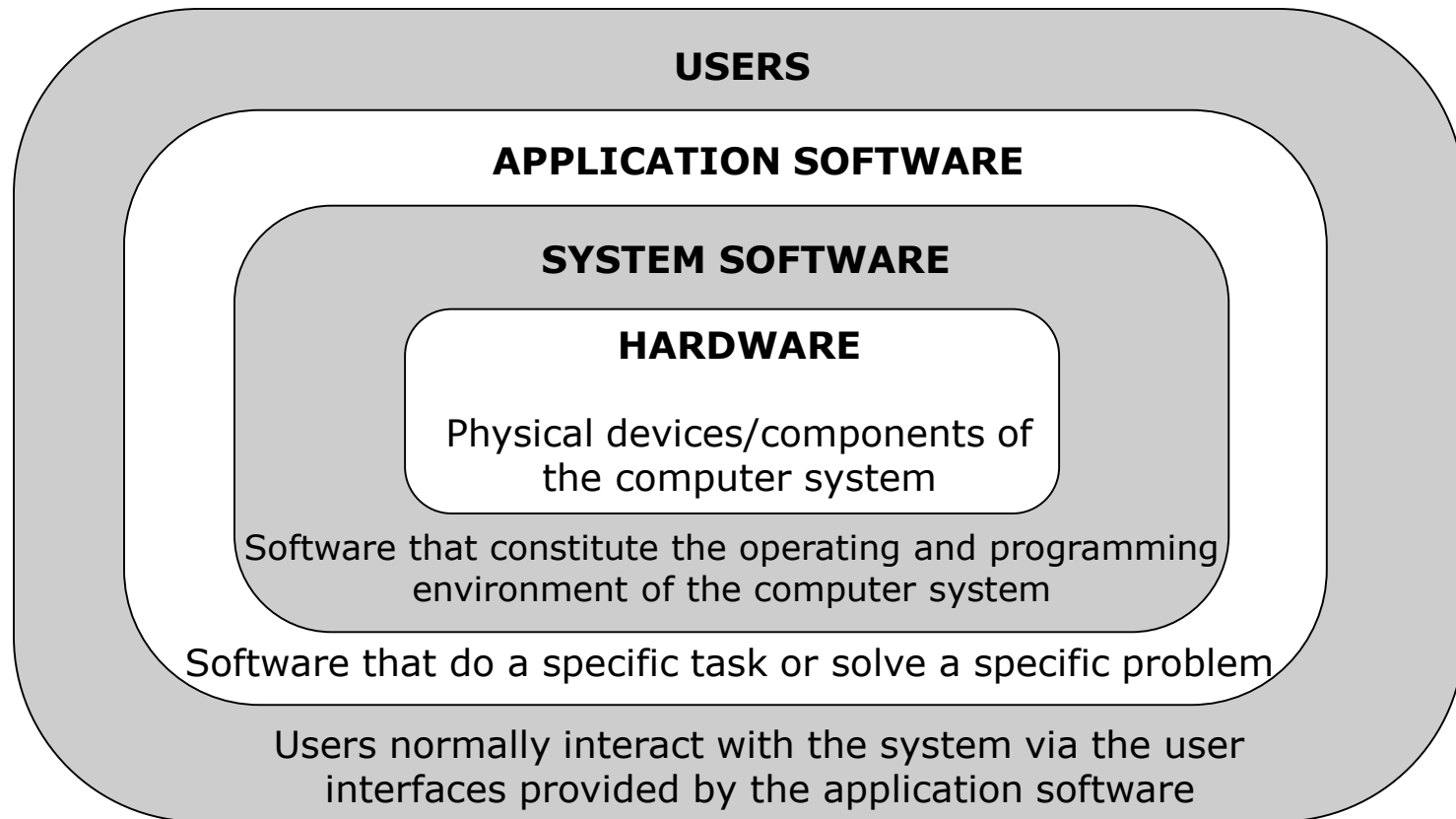
- Make the operation of a computer system more effective and efficient
- Help hardware components work together and provide support for the development and execution of application software
- Programs included in a system software package are called ***system programs*** and programmers who prepare them 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



# Firmware



- *Firmware* refers to a sequence of instructions (software) substituted for hardware
- This software is stored in a read-only memory (ROM) chip of the computer
- Initially, vendors supplied only system software in the form of firmware
- Many vendors now supply even application programs as firmware
- Firmware is frequently a cost-effective alternative to wired electronic circuits
- Firmware has today made it possible to produce smart machines of all types

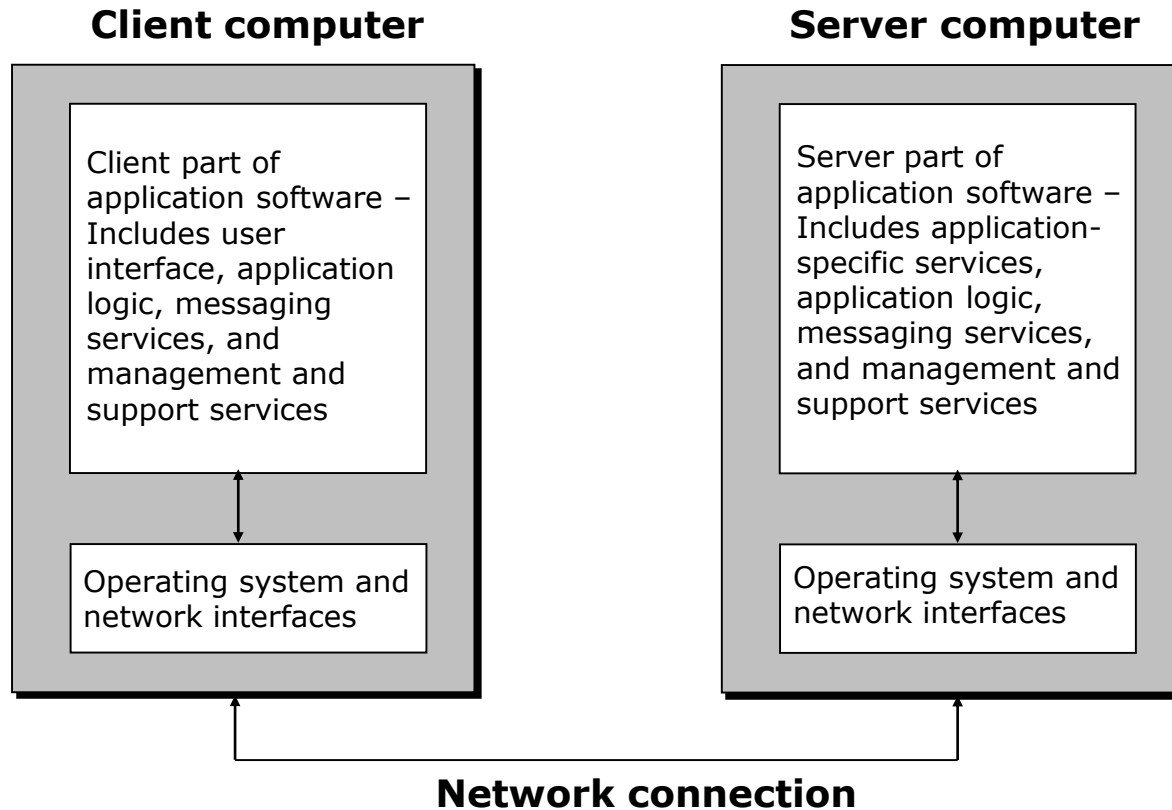


# Middleware

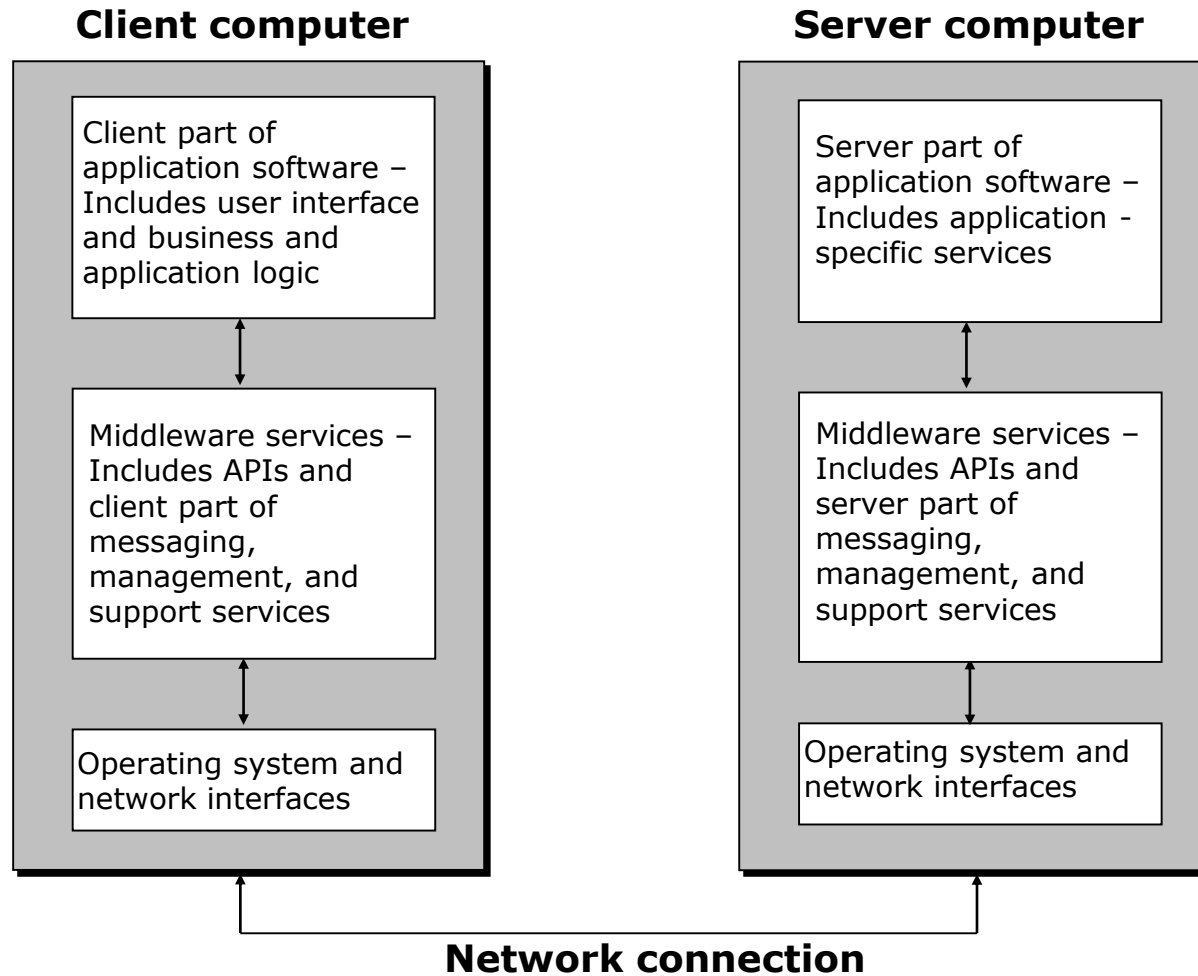


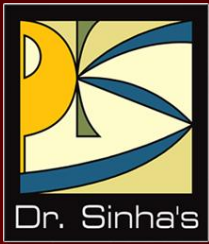
- Basic idea here is to have a separate software layer that acts as “glue” between the client and server parts of an application
- Provides a programming abstraction as well as masks the heterogeneity of underlying networks, hardware, and operating systems
- This software layer is known as middleware because it sits in the middle, between the operating system and applications
- *Middleware* is defined as a set of tools and data that helps applications use networked resources and services

# Two-tier, Client-server Architecture



# Three-tier, Client-server Architecture





# Acquiring Software



# 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

# Buying Pre-written Software



- Prepare a list of all available software packages that can perform the desired task
- Select those software packages only that meet the system specifications
- Choose the best one
- Find out the source from where you can purchase the finally selected software at the cheapest price

# 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 is not as good for pre-written software packages as for in-house developed software packages

# Ordering Customized Software



- If none of the available pre-written software packages meet the specific requirements of a user it becomes necessary for the user to create a customized software package
- If a team does not exist in-house, the user must get it created by another organization by placing an order for it
- Following steps are followed for this:
  - User prepares a list of all user requirements carefully
  - User then floats a tender for inviting quotations for creation of the requisite software
  - After receiving the quotations, the user selects a few of them for further interaction based on the cost quoted by them, their reputation in the market, their submitted proposal, etc.
  - User then personally interacts with the representative(s) of each of the selected vendors



# Ordering Customized Software



- User makes a final choice of the vendor to offer the contract for creation of the requisite software
- Selected vendor then creates the software package and delivers it to the user
- Vendor need not develop everything from scratch
- User may choose to place the order for both hardware and software to a single vendor
- Vendor develops the software on the chosen hardware, and delivers the software along with the hardware to the user
- This is referred to as an *end-to-end solution* or a *turnkey solution*

# 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

# Developing Customized Software



- If no pre-written software package meets the specific requirements, and if the organization has in-house software development team, it may choose to develop a customized software package in-house
- Following steps are followed for in-house development of a software package:
  - Organization first constitutes a project team to develop the software
  - Team studies the requirements and plans functional modules
  - It then analyzes which of the functional modules need to be developed, and which of the functional modules' requirements are met with existing pre-written software
  - Team next plans their programs and does coding, testing, debugging, and documentation

*(Continued on next slide...)*

# Developing Customized Software



- Team then tests all the modules in an integrated manner
- Team then deploys the software for use by users
- Users then use the software and the project team members responsible for maintenance activities maintain it

# 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

# Downloading Public-domain Software

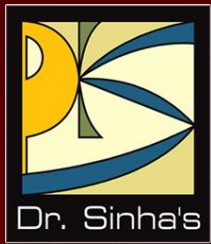


- *Public-domain software* is software available free or for a nominal charge from the bulletin boards or user-group libraries on the Internet
- Public-domain software is also referred to as *shareware/freeware*
- They are also known as *community-supported software* as mostly the authors do not support the product directly and users of the software support and help each other
- *Open Source Software (OSS)*: Usually, OSS allows a user to download, view, modify, and distribute modified source code to others

# 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

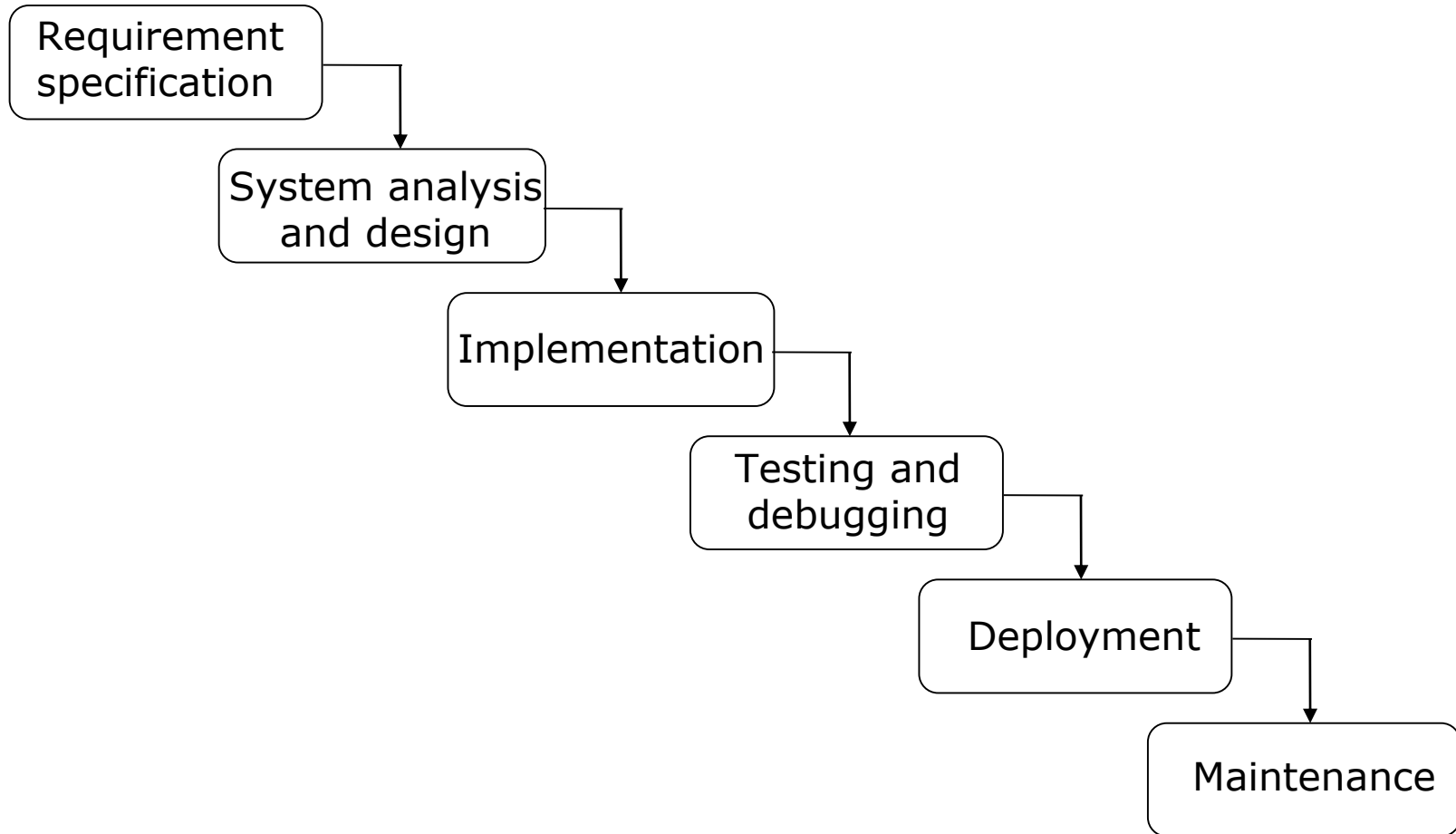


# Software Development Life Cycle (SDLC)





# Software Development Life Cycle (SDLC)



*(Continued on next slide...)*

# Software Development Life Cycle (SDLC)



- **Requirement specification**
  - Team defines all possible requirements of the software in this phase
- **System analysis and design**
  - Team studies the requirements specified in the first phase with respect to available hardware and software technologies and prepares a system design document
- **Implementation**
  - This phase is also known as *construction* or *code generation* because in this phase, the team constructs the various software components specified in system design document

# Software Development Life Cycle (SDLC)



## ■ **Testing and debugging**

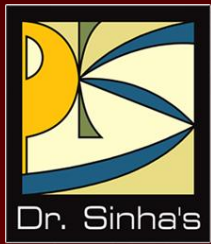
- Team integrates the independently developed and tested modules into a complete system
- Team then tests the integrated system to check if all modules coordinate properly with each other

## ■ **Deployment**

- Team deploys the software at user(s) site on (or along with) associated hardware for use by intended user

## ■ **Maintenance**

- Team fixes problems in the system in this phase



# Software Engineering



# What is Software Engineering?



- *Software* is a set of computer programs, procedures, and associated documents
- *Engineering* is systematic application of scientific knowledge in creation and building of cost-effective solutions
- *Software engineering* is systematic application of principles of computer science and mathematics in creation and building of cost-effective software solutions

# Need for Software Engineering



- With software products growing in scale and complexity, number of software developers involved in a software development project has been increasing proportionately
- Managing the development of large software products and maintaining them is a difficult task
- Progressively larger software products in sensitive applications are being used
- Required correctness and reliability of software products is increasing
- Quality and productivity demands for software products led to the introduction of systematic practices (later on known as *software engineering practices*)

# Goals of Software Engineering



- **Correctness should be of very high degree**
  - Correctness refers to the degree to which a software product performs its intended functions properly, consistently, and predictably
- **Usability should be of very high degree**
  - Usability refers to the ease with which a software product and its associated documentation are usable
- **Should be cost-effective**
  - Cost-effectiveness means that the total development and operational costs of a software product should be as low as possible

# Principles of Software Engineering



- **Precise requirements definition**
  - Designer of a software product must define its requirements precisely
- **Modular structure**
  - Designer of a software product must structure it in a modular fashion
  - Modular design helps in distribution of development task of different modules to different programmers
- **Abstraction**
  - Software product should use abstraction and information hiding
  - Object-oriented programming takes care of this aspect of software engineering
  - Abstraction helps in easy reusability of existing modules



# Principles of Software Engineering



## ■ **Uniformity**

- Software product should maintain uniformity in design, coding, documentation, etc. Uniformity ensures consistency
- **CASE (Computer Aided Software Engineering) tools** provide a wide range of features for creation of better and more reliable software products
  - *Design specification tools*
    - Allow programmers to design visually screens, menus, database tables, reports, dialog boxes, and other key components of a program
  - *Code generation tools*
    - Generate source code (programs) from the design specification of a software product

(Continued on next slide...)

# Principles of Software Engineering

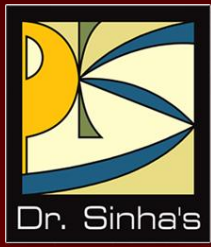


- *Testing and debugging tools*
  - Help programmers in testing and debugging of their programs
- *Source-code analysis tools*
  - Help in optimizing a program by pointing out unreachable lines of code and functions that the program never uses (calls)
- *Documentation tools*
  - Assist in automatic generation of technical documents for a software product

# Key Words/Phrases



- Application programmers
- Application programs
- Application software
- CASE tools
- 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 Development Life Cycle (SDLC)
- Software Engineering
- Software package
- Spreadsheet
- System programmers
- System programs
- System software
- Turnkey solution
- User-supported software
- Utilities
- Waterfall model
- Word-processing



# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 11

## Planning the Computer Program

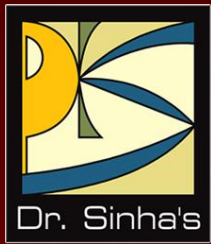


# Learning Objectives



## In this chapter you will learn about:

- Programs must be planned before they are written
- Planning the logic of a computer program
- Commonly used tools for program planning
- Algorithm
- Flowchart
- Pseudocode



# Purpose of Program Planning and Algorithm



# 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

# What is an Algorithm?



- *Algorithm* refers to the logic of a program
- It is a step-by-step description of how to arrive at a solution to a given problem
- Is defined as a sequence of instructions that when executed in the specified sequence, the desired results are obtained
- Instructions must possess following characteristics:
  - Each instruction should be precise and unambiguous
  - Each instruction should be executed in a finite time
  - No instruction should be repeated infinitely. This ensures that the algorithm terminates ultimately
  - After executing the instructions (when the algorithm terminates), the desired results are 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.

*(Continued on next slide...)*

# Sample Algorithm (Example 1)



- 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.

*(Continued on next slide...)*

# Sample Algorithm (Example 2)



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

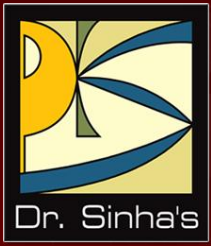
# Representation of Algorithms



- 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



# Flowcharts



# What is a Flowchart?



- *Flowchart* is a pictorial representation of an algorithm
- Programmers often use it as a program-planning tool
- It uses boxes of different shapes to denote different types of instructions
- Process of drawing a flowchart for an algorithm is known as *flowcharting*

# Why Use Flowcharts?



- Algorithm is first represented as a flowchart
- Flowchart is then expressed in a programming language
- Main advantages of this two-step approach in program writing are:
  - While drawing a flowchart, a programmer can concentrate fully on the logic of the solution
  - Since a flowchart shows the flow of operations in pictorial form, a programmer can detect any error in the logic
  - Once the flowchart is ready, the programmer can concentrate on coding the operations in each box of the flowchart as statements of the programming language

*(Continued on next slide...)*



# Why Use Flowcharts?



- This two-step approach ensures an error-free program
- It is a good practice to have a flowchart along with a computer program because the flowchart often serves as a document for the computer program

# Basic Flowchart Symbols



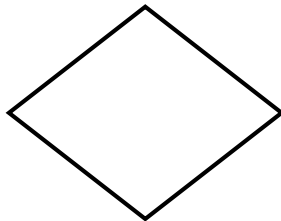
Terminal



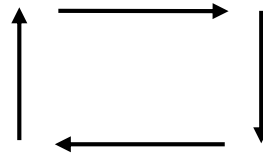
Input/Output



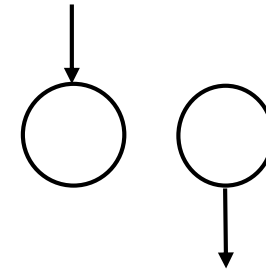
Processing



Decision

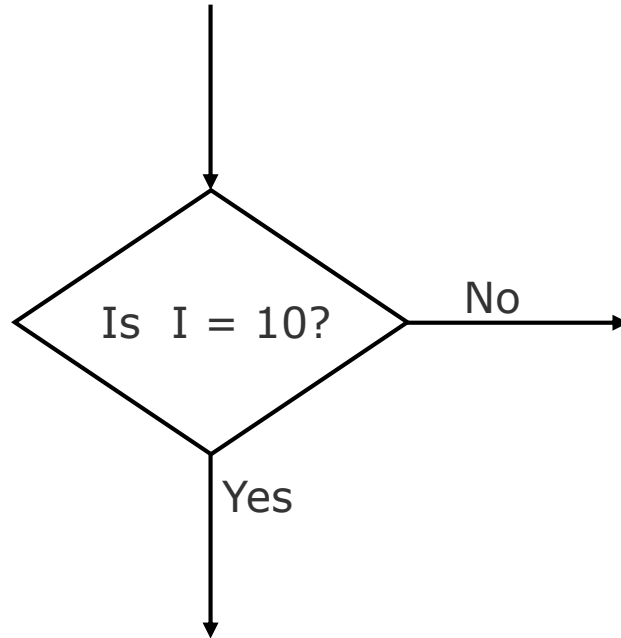


Flow lines

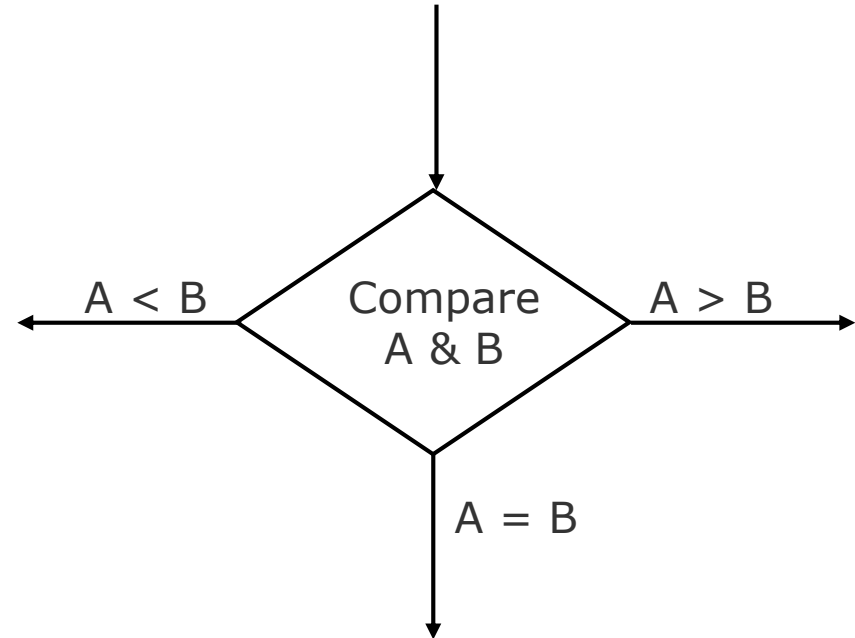


Connectors

# Examples of Decision Symbol



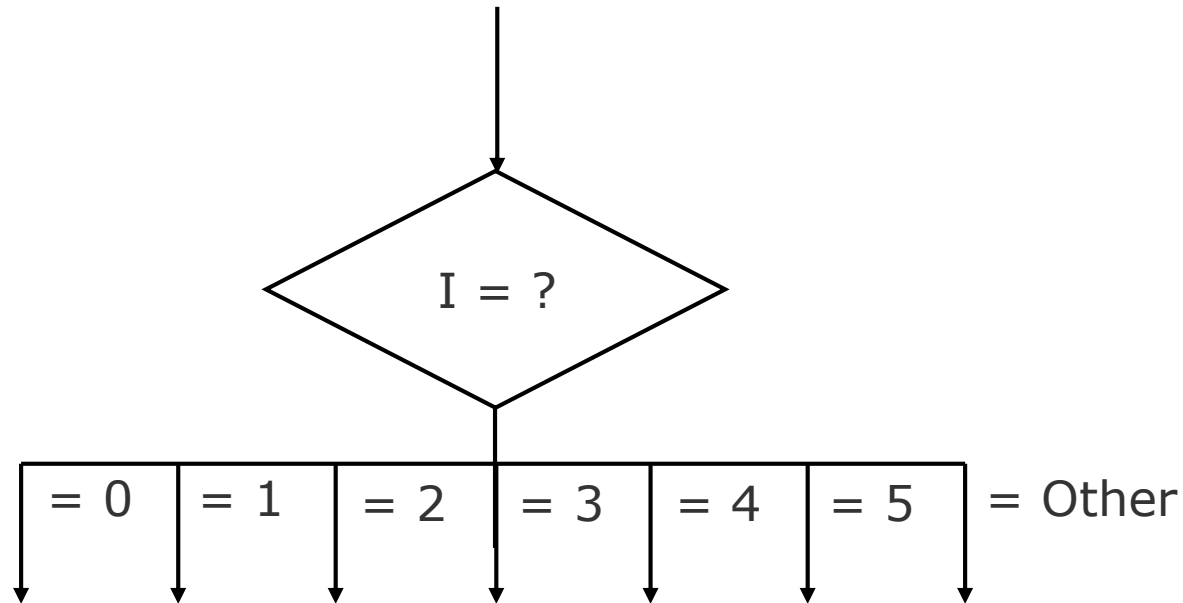
(a) A two-way branch decision.



(b) A three-way branch decision.

*(Continued on next slide...)*

# Examples of Decision Symbol



(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.

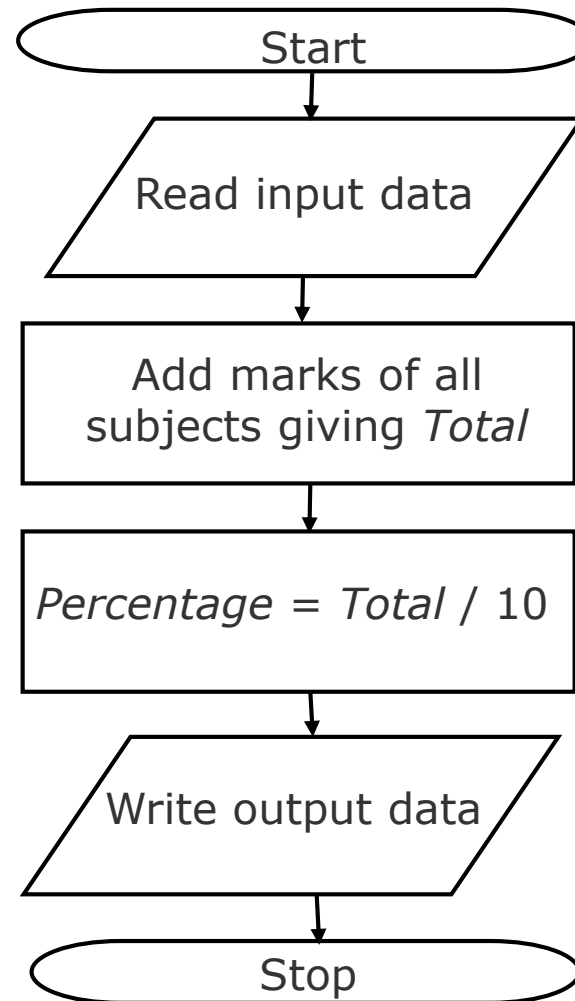
The roll number of the student, his/her name, and the marks obtained by him/her in various subjects are supplied as input data.

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.

*(Continued on next slide...)*

# Sample Flowchart (Example 3)



# Sample Flowchart (Example 4)

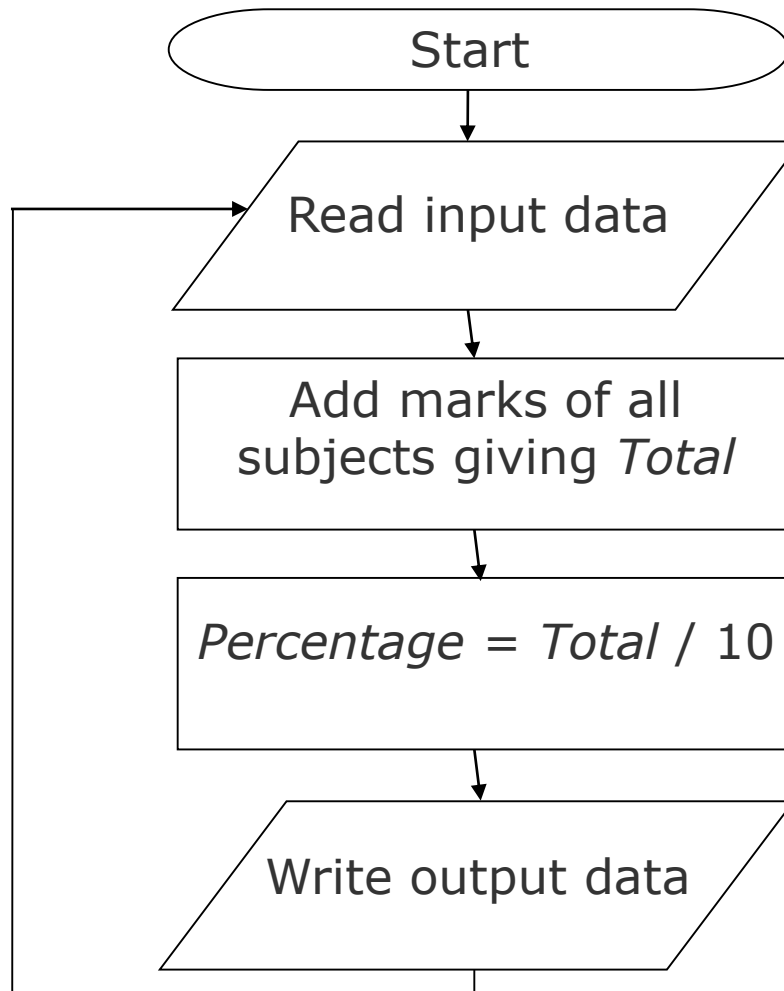


50 students of a class appear in the examination of Example 3.

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.

*(Continued on next slide...)*

# Sample Flowchart (Example 4)

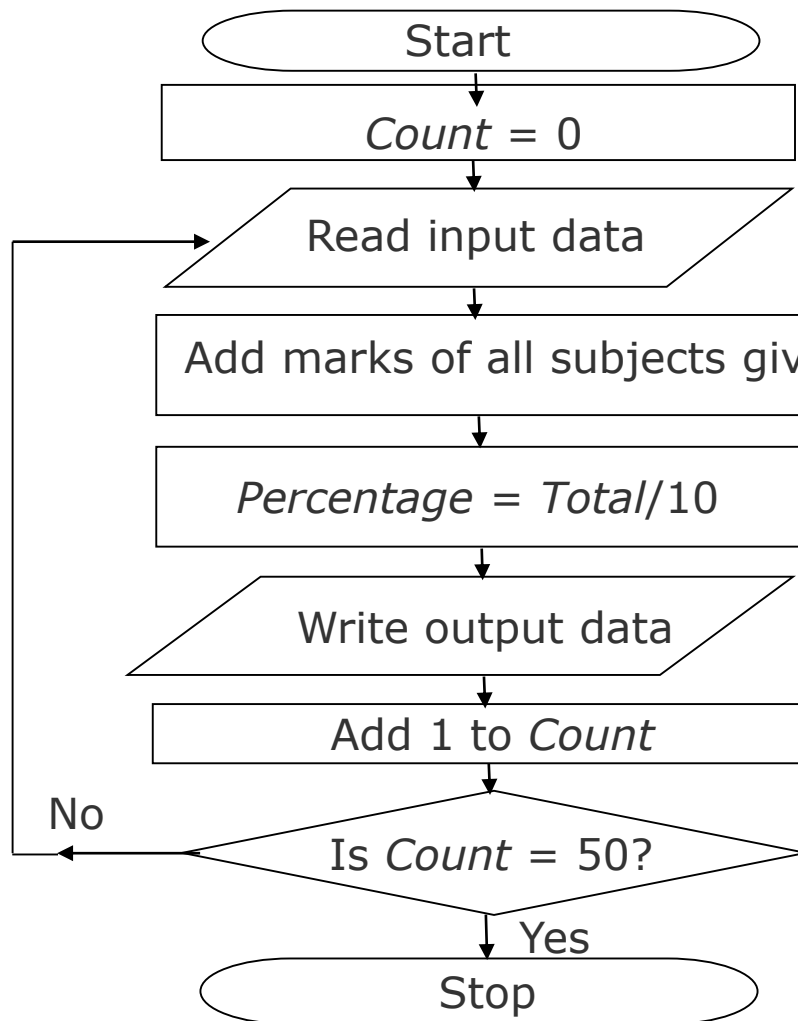


Flowchart for the solution of Example 4 with an infinite (endless) process loop.

(Continued on next slide...)



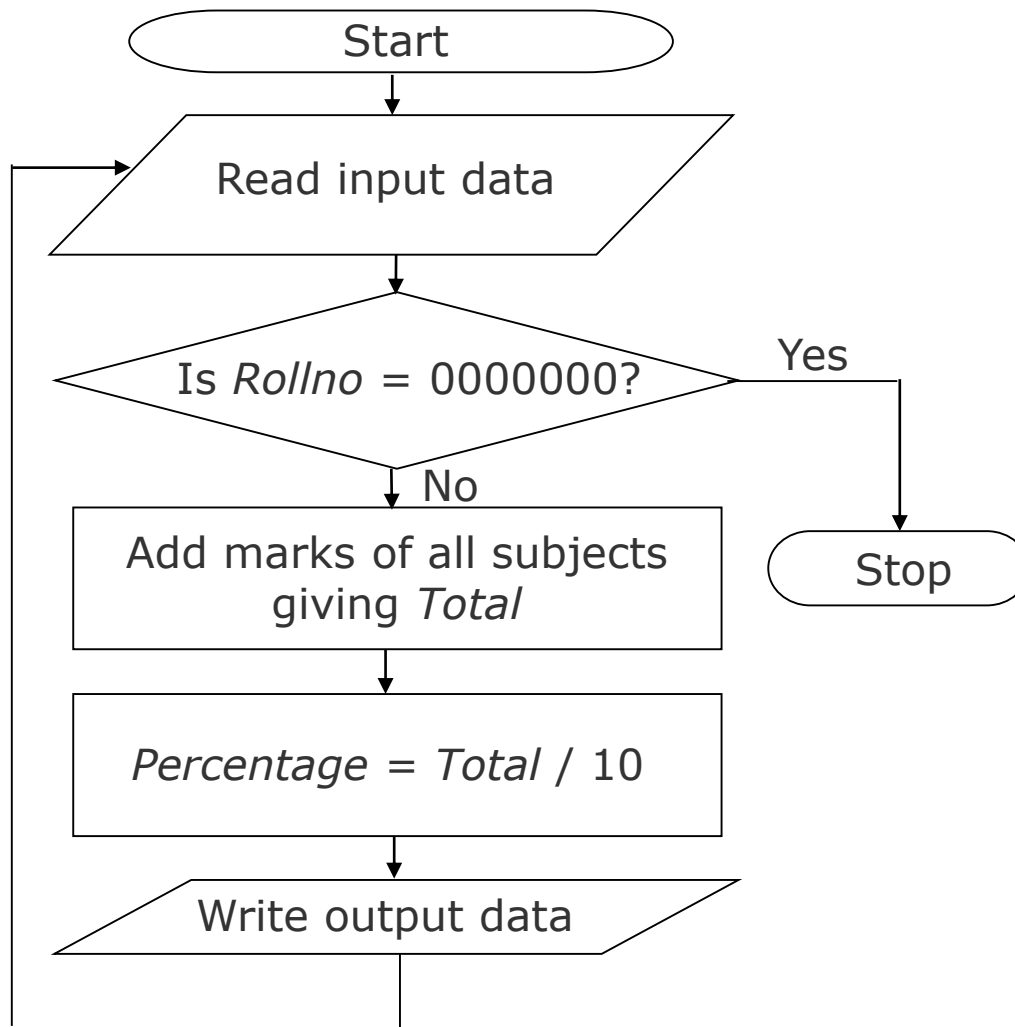
# Sample Flowchart (Example 4)



Flowchart for the solution of Example 4.

(Continued on next slide...)

# Sample Flowchart (Example 4)



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)



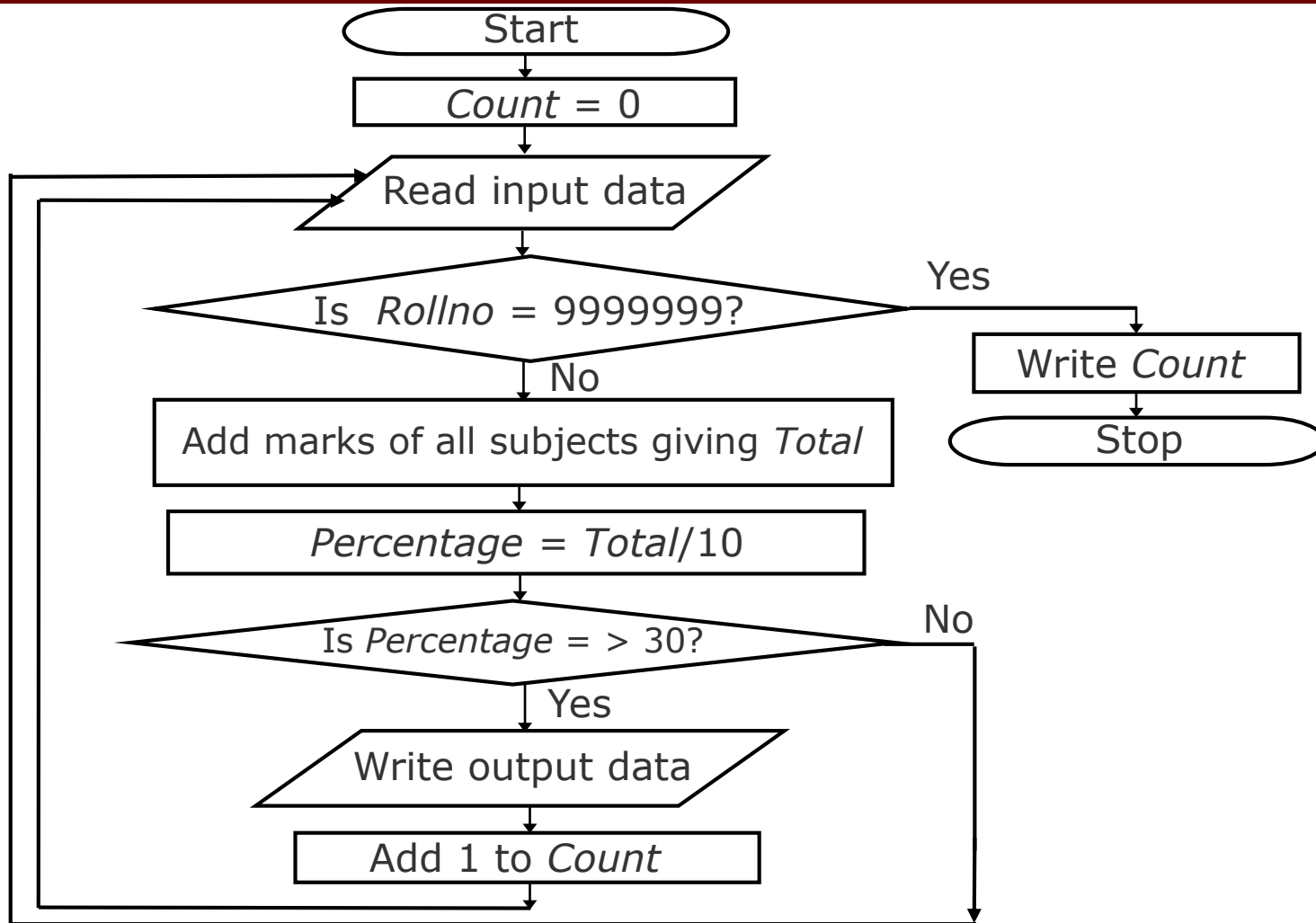
For the examination of Example 3, we want to make a list of only those students who have passed (obtained 30% or more marks) in the examination.

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.

*(Continued on next slide...)*

# Sample Flowchart (Example 5)



# Sample Flowchart (Example 6)



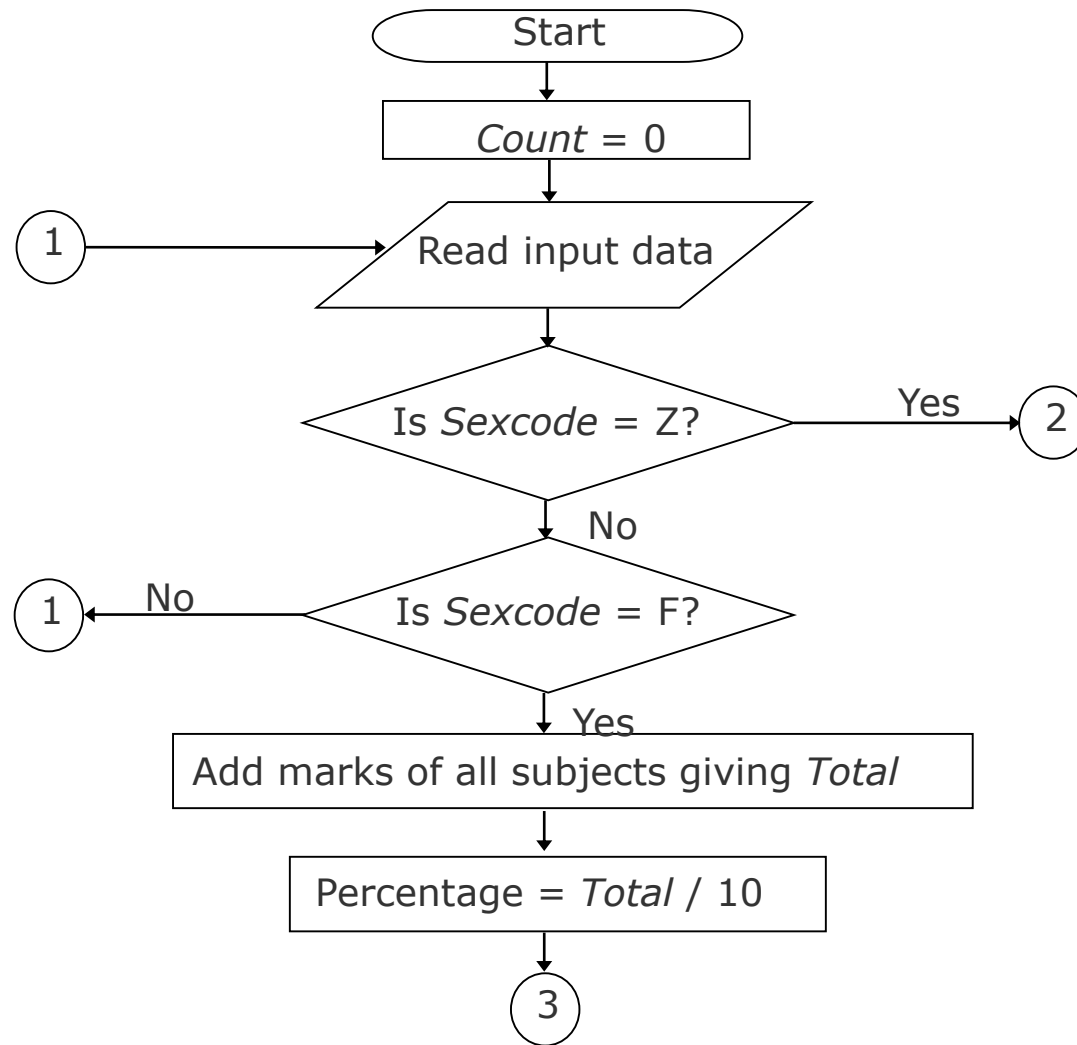
Suppose the input data of each student for the examination of Example 3 also contains information regarding the sex of the candidate in the field named *Sexcode* having values M (for male) or F (for female).

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.

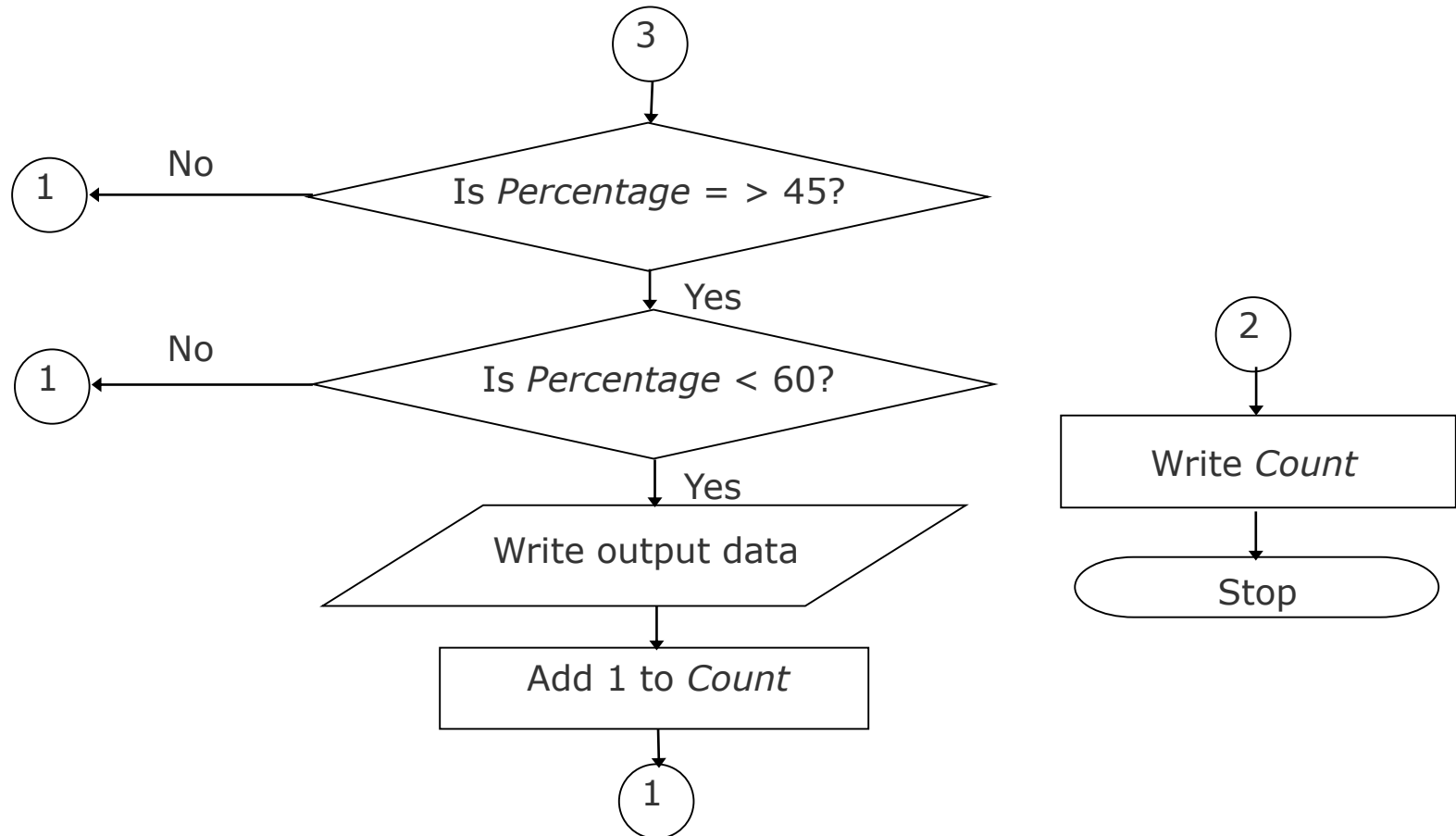
Assuming that the input data of all the students is terminated by a trailer record, which has a sentinel value of Z for *Sexcode*, draw a flowchart for the algorithm to do this.

# Sample Flowchart (Example 6)



(Continued on next slide...)

# Sample Flowchart (Example 4)



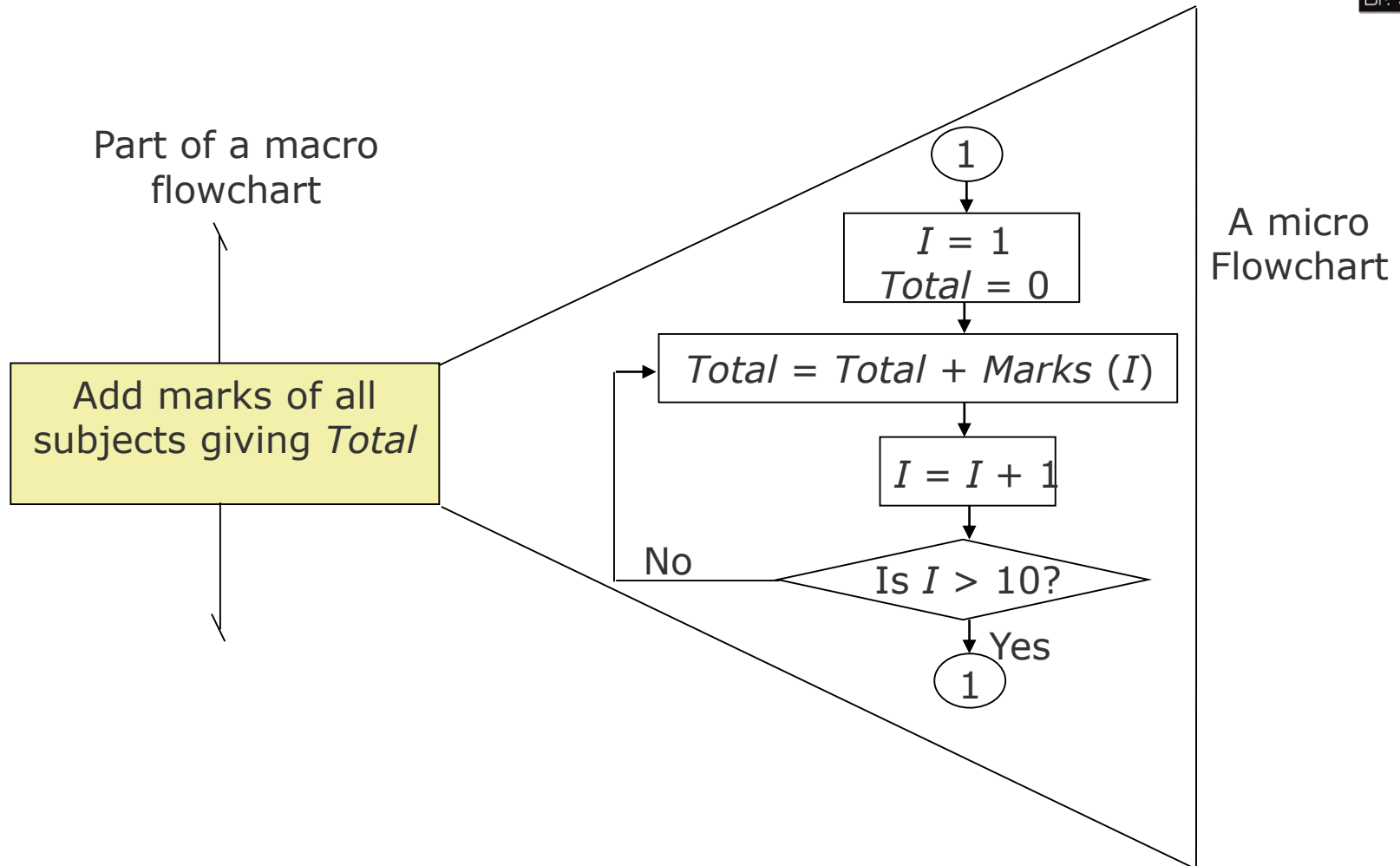
# Levels of Flowchart



- 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

*(Continued on next slide...)*

# Flowcharting Rules



- 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 Flowcharts



- **Better communication**

- Flowchart is a pictorial representation of a program

- **Effective analysis**

- Macro flowchart, which charts the main line of logic of a software system, becomes the system's model

- **Effective synthesis**

- Group of programmers are associated with the design of a big software system
- Each programmer is responsible for designing only a part of the entire system
- Flowcharts of all programmers put together can help visualize the overall system design

*(Continued on next slide...)*

# Advantages of Flowcharts



## ■ **Proper program documentation**

- Documentation involves collecting, organizing, storing, and maintaining a complete historical record of programs, and other documents associated with a system
- Flowcharts often provide valuable documentation support

## ■ **Efficient coding**

- Once a flowchart is ready, programmers find it very easy to write the corresponding program because the flowchart acts as a road map

## ■ **Systematic debugging**

- Flowchart is very helpful in detecting, locating, and removing mistakes (bugs) in a program in a systematic manner

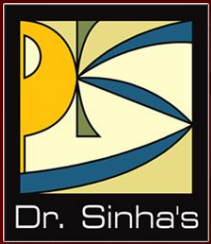
## ■ **Systematic testing**

- Flowchart is very helpful in designing test data for systematic testing of programs

# 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



# What is Pseudocode?



- *Pseudocode* is another program-planning tool used for planning program logic
- “Pseudo” means imitation or false and “Code” refers to the instructions written in a programming language
- Pseudocode is an imitation of actual computer instructions
- Pseudo-instructions are phrases written in a natural language
- Pseudocode uses a structure that resembles computer instructions
- A programmer can concentrate solely on developing the logic of a program without worrying about the syntax
- He/she can convert the pseudocode easily into a suitable programming language



# Basic Logic (Control) Structures



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

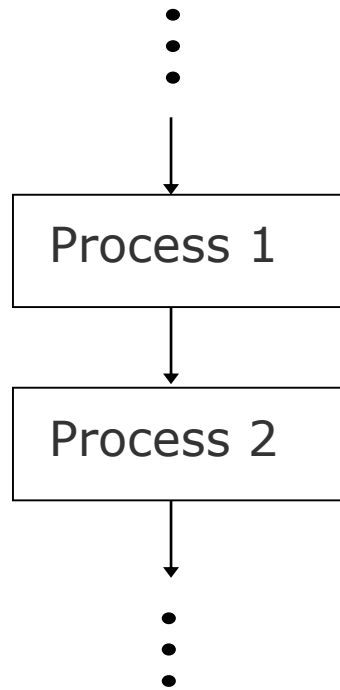
1. Sequence logic,
2. Selection logic, and
3. Iteration (or looping) logic

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***

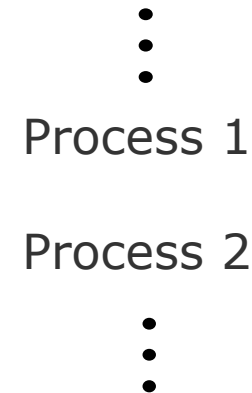
# Sequence Logic



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



(a) Flowchart



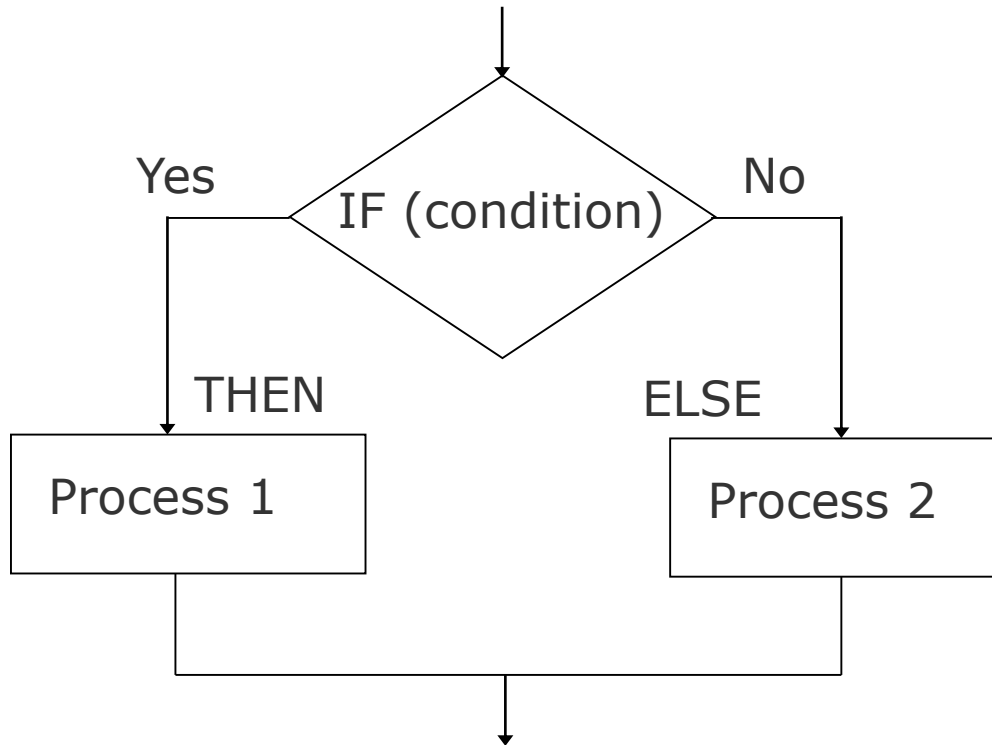
(b) Pseudocode

# 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)



(a) Flowchart

```

...
IF Condition
    THEN    Process 1
    ELSE    Process 2
ENDIF

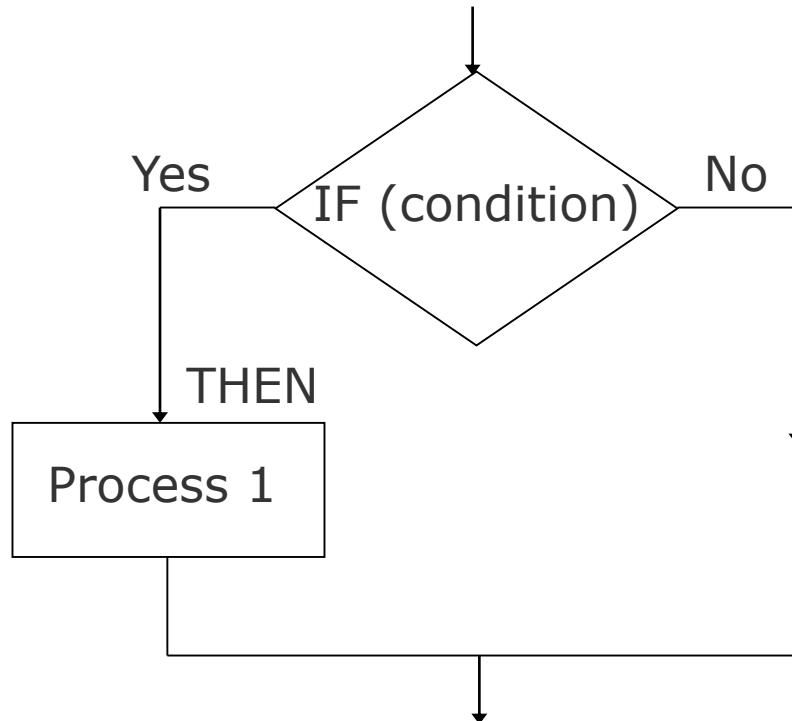
```

```

...
(b) Pseudocode

```

# Selection Logic (IF..THEN Structure)

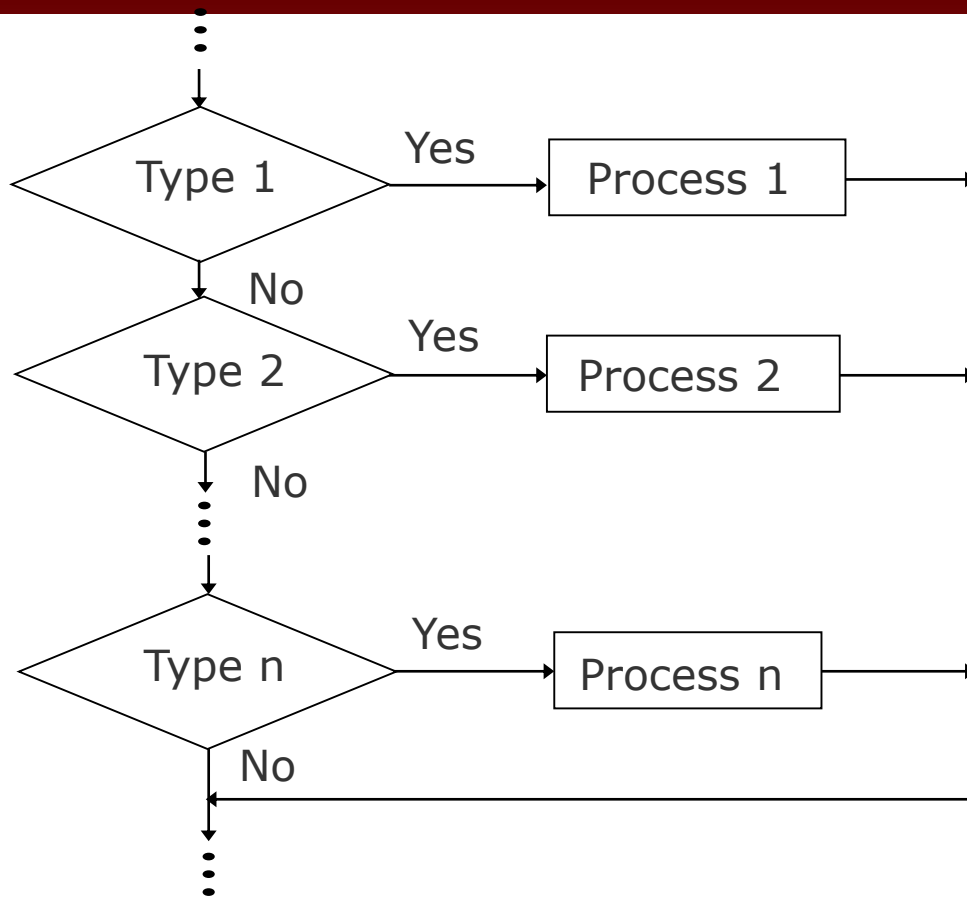


(a) Flowchart

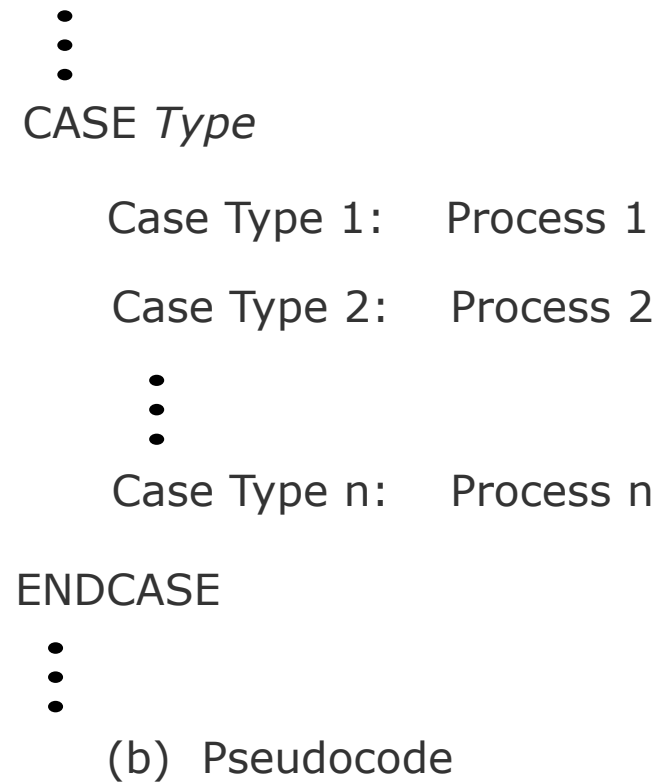
```
⋮  
IF Condition  
  
    THEN    Process 1  
  
ENDIF  
  
⋮
```

(b) Pseudocode

# Selection Logic (CASE Structure)



(a) Flowchart



(b) Pseudocode

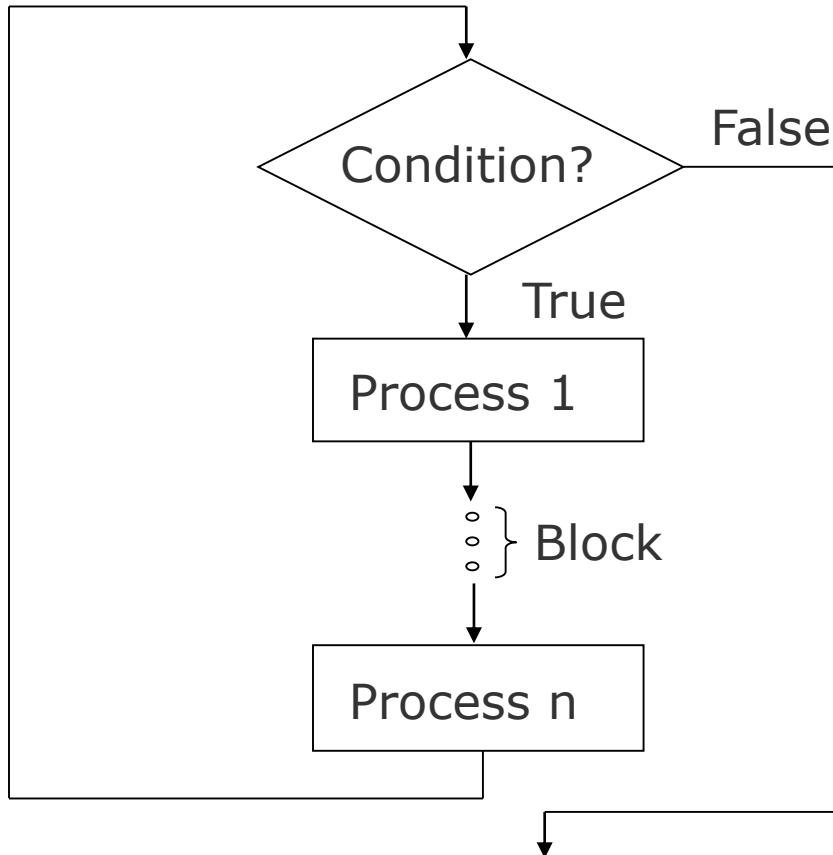
# Iteration (or Looping) Logic



- 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
:

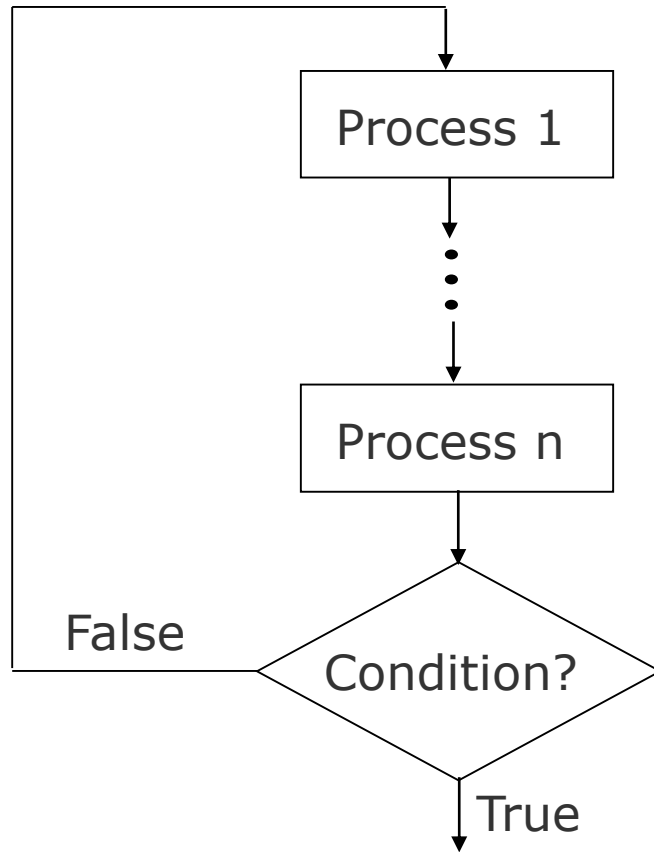
```

(b) Pseudocode



# Iteration (or Looping) Logic

(REPEAT...UNTIL Structure)



(a) Flowchart

•  
 •  
 •  
 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

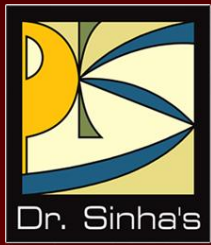


- In case of pseudocode, a graphic representation of program logic is not available
- There are no standard rules to follow in using pseudocode
- Different programmers use their own style of writing pseudocode and hence communication problem occurs due to lack of standardization
- For a beginner, it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



# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 12

**Computer  
Languages**

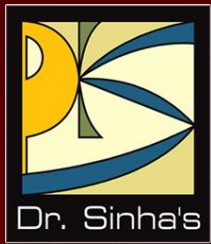


# Learning Objectives



## 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, Python, 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



# Analogy with Natural Languages





# Analogy with Natural Languages

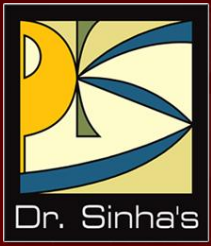


- Language is a means of communication
- Programmer uses a computer language to instruct a computer what he/she wants it to do
- Set of words allowed in a language is called its *vocabulary*
- Each word in the vocabulary has a definite unambiguous meaning
- Most computer languages use a very limited vocabulary
- Words and symbols of a computer language must be used as per the set rules, known as *syntax rules*

# Broad Classification of Computer Languages



- Machine language
- Assembly language
- High-level language



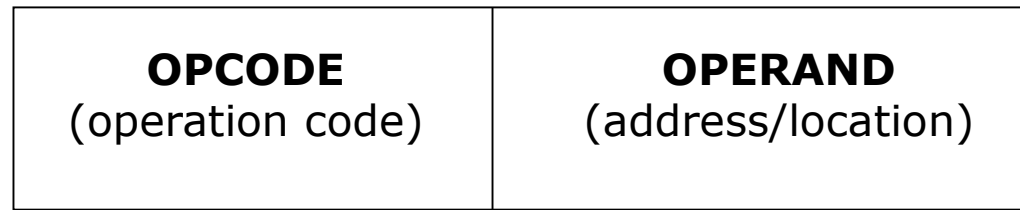
# Machine Language



# Machine Language



- Machine language programs are written normally as strings of binary 1s and 0s
- Circuitry of a computer is wired in a manner that it recognizes the machine language instructions and converts them into electrical signals needed to execute them
- A machine language instruction normally has a two-part format: *operation code* tells the computer what function to perform, and *operand* tells where to find or store the data



Instruction format

(Continued on next slide...)

# Machine Language



- Every computer has a set of operation codes called its *instruction set*
- Typical operations included in the instruction set are:
  - Arithmetic operations
  - Logical operations
  - Branch operations
  - Data movement operations between memory locations and registers
  - Data movement operations from/to input/output devices

# A Sample Machine Language Program



```
0010000000000001100111001
0011000000000010000100001
0110000000000011100101110
101000111111011100101110
00000000000000000000000000
```

In Binary

(Difficult to read and understand)

```
10001471
14002041
30003456
50773456
00000000
```

In Decimal

(Easier to read and understand)

# Advantages & Limitations of Machine Language

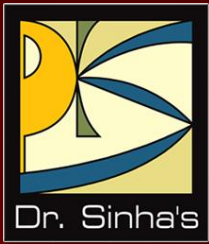


## Advantage

- Can be executed very fast

## Limitations

- Machine dependent
- Difficult to program
- Error prone
- Difficult to modify



# Assembly Language





# Assembly Language



- Assembly language programming helped in overcoming limitations of machine language programming
  - By using alphanumeric mnemonic codes instead of numeric codes for instructions in instruction set
  - Allowing use of alphanumeric names instead of numeric addresses
  - Providing additional instructions, called pseudo-instructions
- A language that allows use of letters and symbols instead of numbers for representing instructions and storage locations is called *assembly language* or *symbolic language*
- A program written in an assembly language is called *assembly language program* or *symbolic program*

# Sample Assembly Language Program

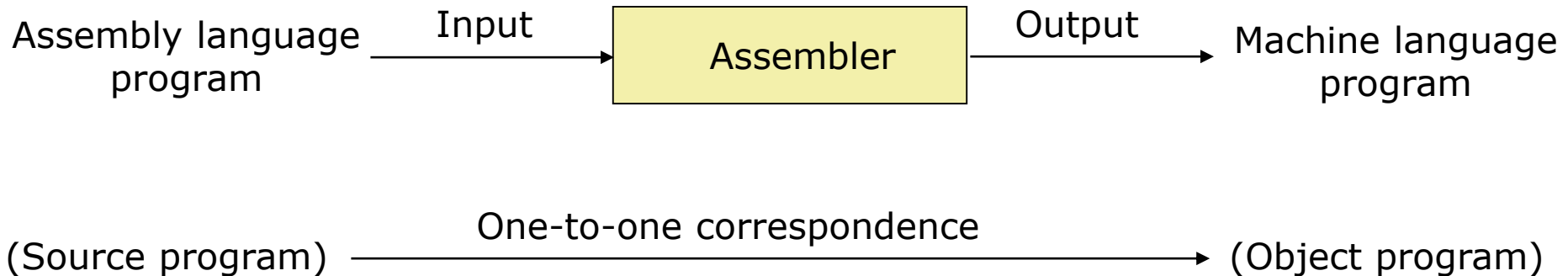


```
START PROGRAM AT 0000
START DATA AT 1000
SET ASIDE AN ADDRESS FOR FRST
SET ASIDE AN ADDRESS FOR SCND
SET ASIDE AN ADDRESS FOR ANSR
CLA FRST
ADD SCND
STA ANSR
HLT
```

# Assembler



- We must convert an assembly language program into its equivalent machine language program before executing it
- Translator program called *assembler* does this translation
- Assembler is system software supplied by computer manufacturers



# 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

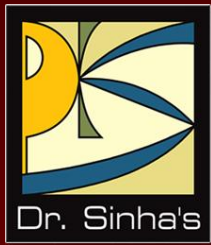


- Mainly used today to fine-tune important parts of programs written in a high-level language to improve the program's execution efficiency

# 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 Language





# High-Level Languages



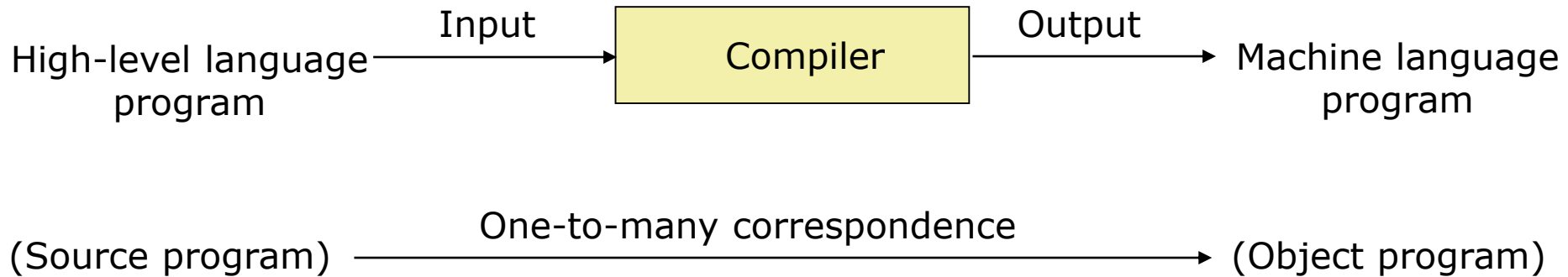
- Machine independent
- Do not require programmers to know anything about the internal structure of computer on which high-level language programs will be executed
- Deal with high-level coding, enabling the programmers to write instructions using English words and familiar mathematical symbols and expressions

# Compiler

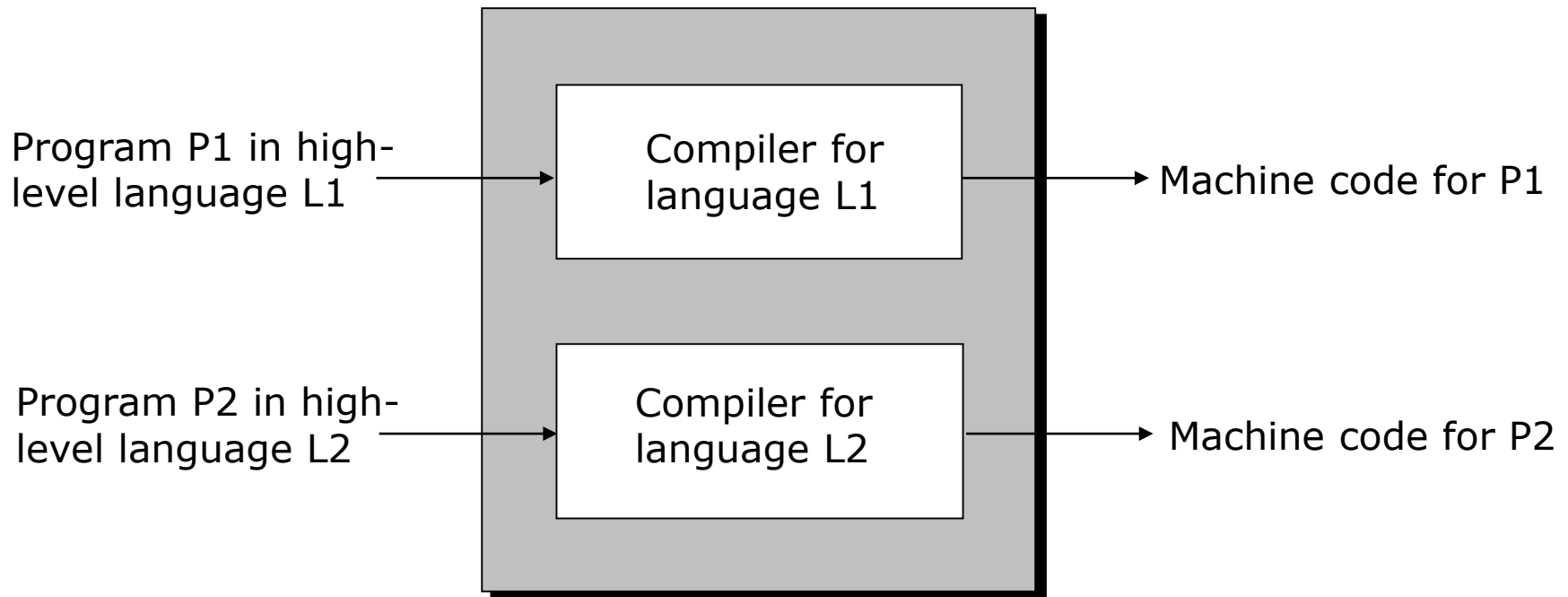


- Translator program (software) that translates a high-level language program into its equivalent machine language program
- Compiles a set of machine language instructions for every program instruction in a high-level language

# Compiler



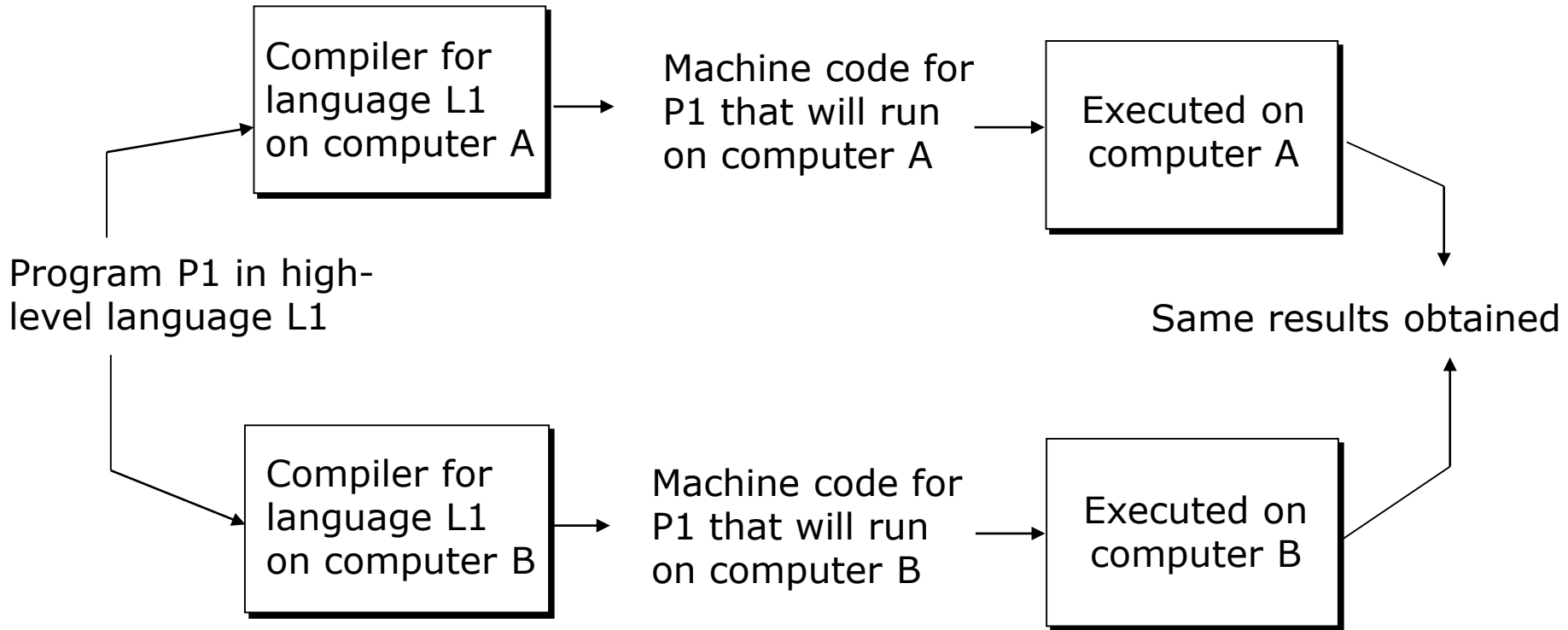
# Separate Compiler for Each High-level Language Supported



A computer supporting languages L1 and L2

*(Continued on next slide)*

# A High-level Language is Machine Independent



# Syntax Errors



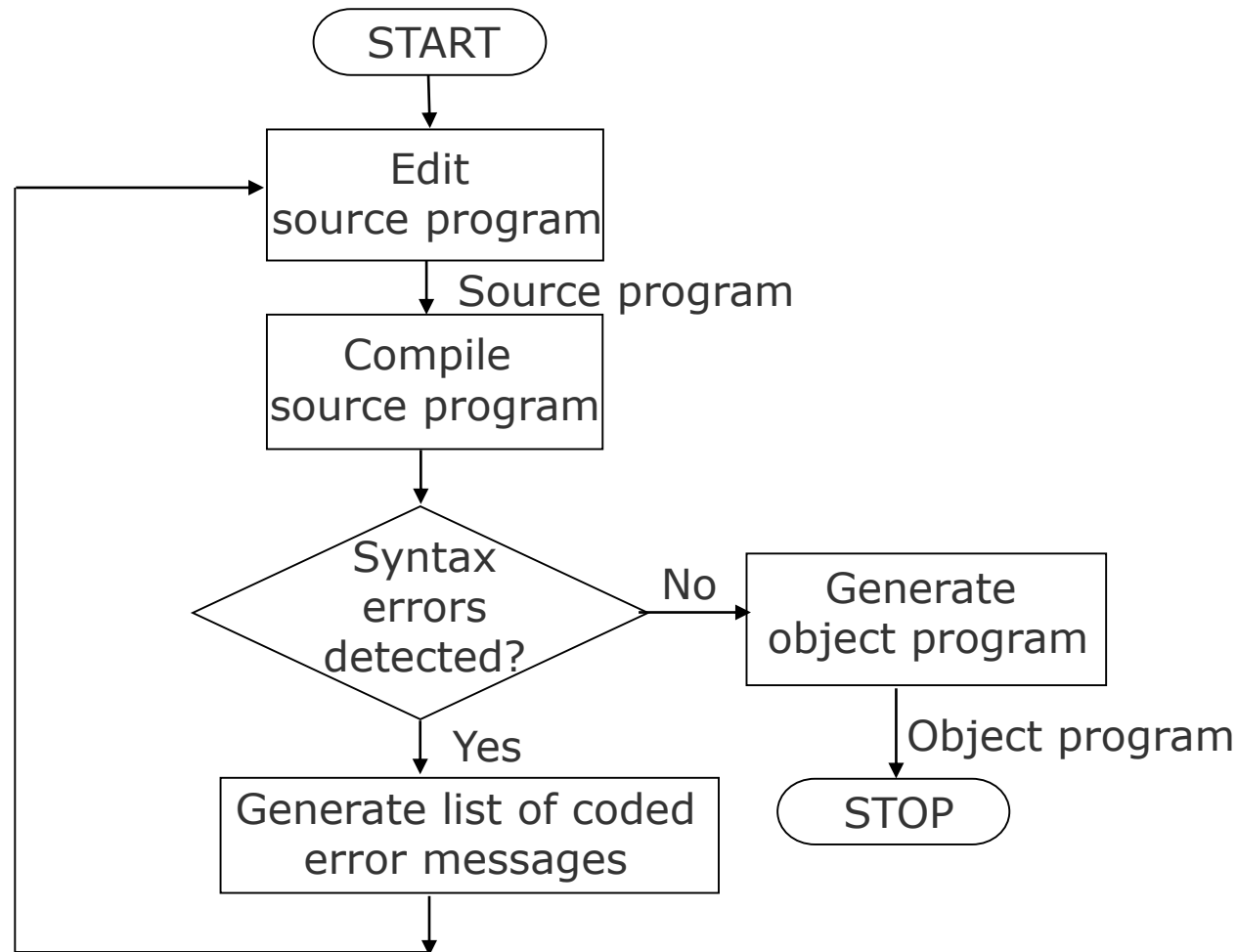
In addition to doing translation job, compilers also automatically detect and indicate syntax errors

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)*

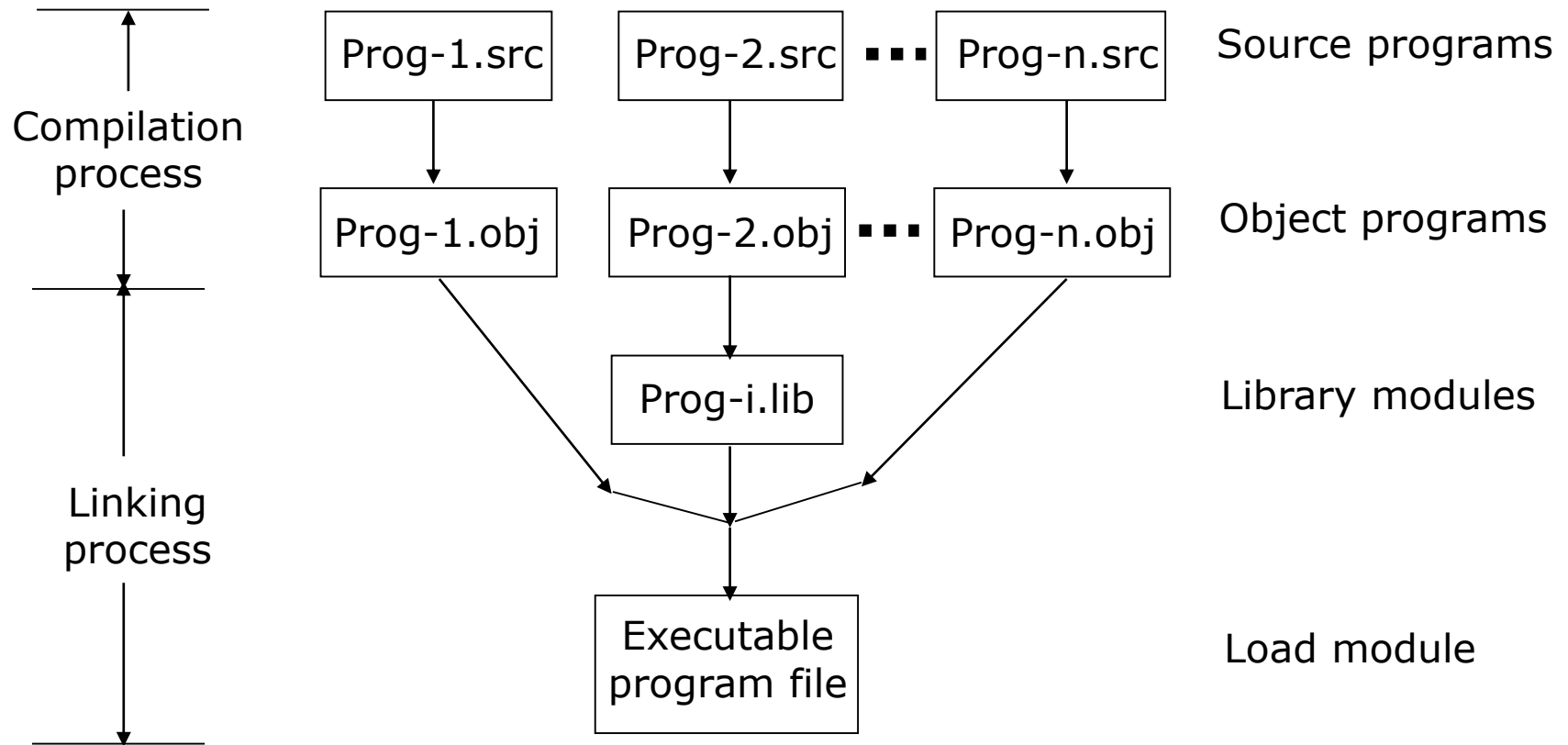


# Linker



- 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)

# Linker

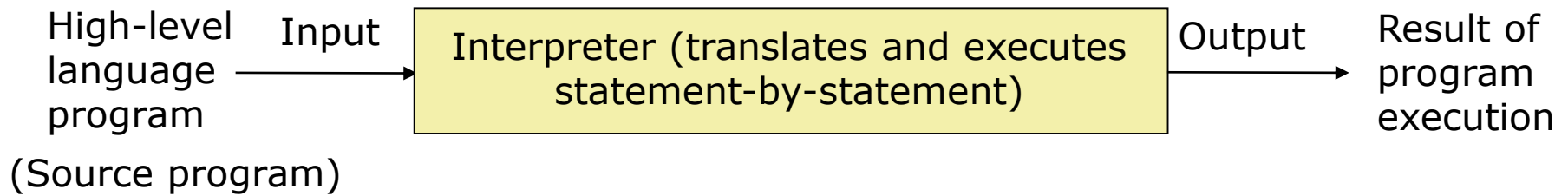


# Interpreter



- *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

# Interpreter



# Interpreter



- As compared to compilers, interpreters are easier to write
- The main advantage of interpreters over compilers is that an interpreter flags a syntax error in a program statement to a programmer as soon as it interprets the program statement
- Main disadvantage of interpreters over compilers is that they are slower than compilers
- To combine the advantages of both interpreters and compilers, computer system provides both a compiler and an interpreter for a high-level language
- Assemblers, compilers, and interpreters are also referred to as *language processors*

# 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



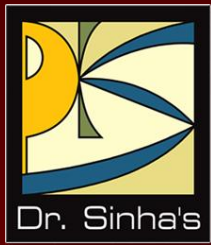
- Machine independent
- Easier to learn and use
- Fewer errors during program development
- Lower program preparation cost
- Better documentation
- Easier to maintain



# Limitations of High-Level Languages



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



# Object-Oriented Languages



# What is Object-Oriented Programming (OOP)?



- Essence of OOP is to solve a problem by:
  - Identifying the real-world objects of the problem and the processing required of those objects
  - Then creating simulations of those objects, their processes, and the required communications between the objects
- OOP makes programming simpler, easier, and faster
- OOP philosophy is now used in almost all popular programming languages

# Fundamental Concepts of OOP



## ▪ **Object**

- An object is the primitive element of a program written in an OOP language
- Each object consists of a set of procedures and some data

## ▪ **Method**

- A method of an object defines the set of operations that the object will execute when it receives a message
- Methods are like function definitions
- Entire collection of methods of an object is called the *message protocol*, or *message interface*

## ▪ **Message**

- Mechanism to support communication between objects is through messages

# Fundamental Concepts of OOP



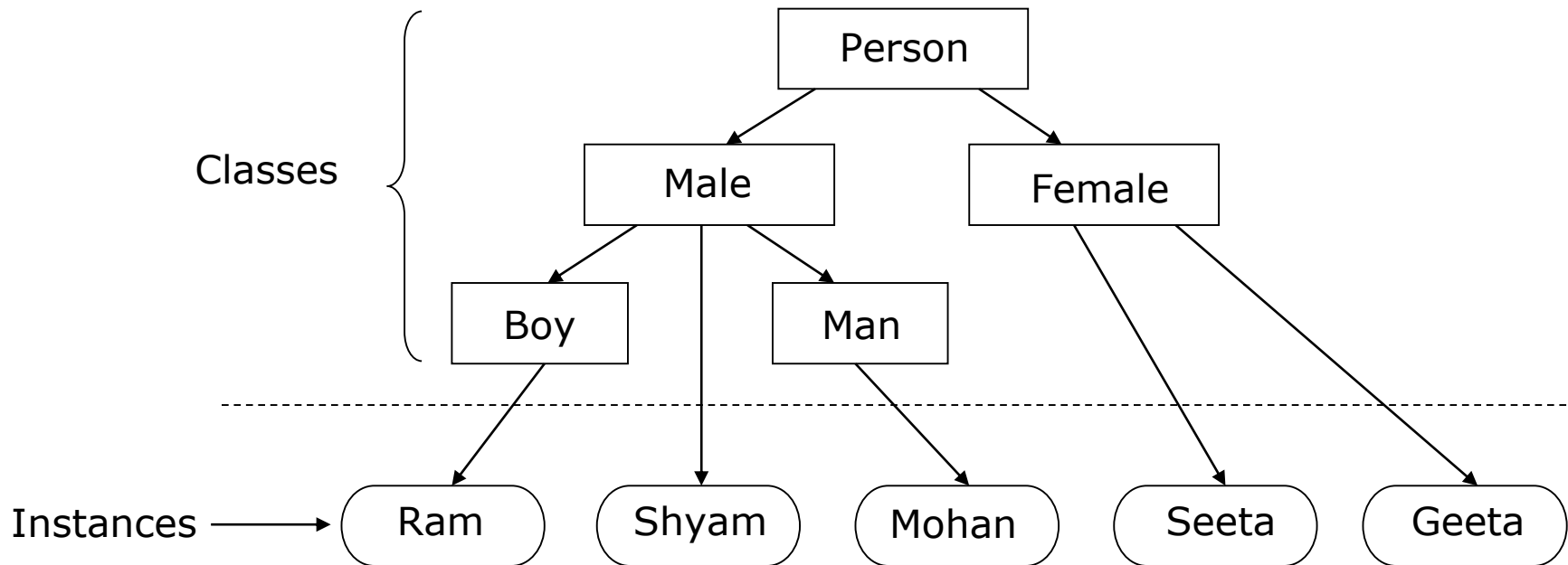
## ■ Class

- A class is a description of one or more similar objects
- A class can have multiple instances
- Each instance of a class is known as an object of the class
- *Class variables* are variables stored in the class whose values are shared by all instances
- *Instance variables* are variables for which local storage is available in the instances (objects)

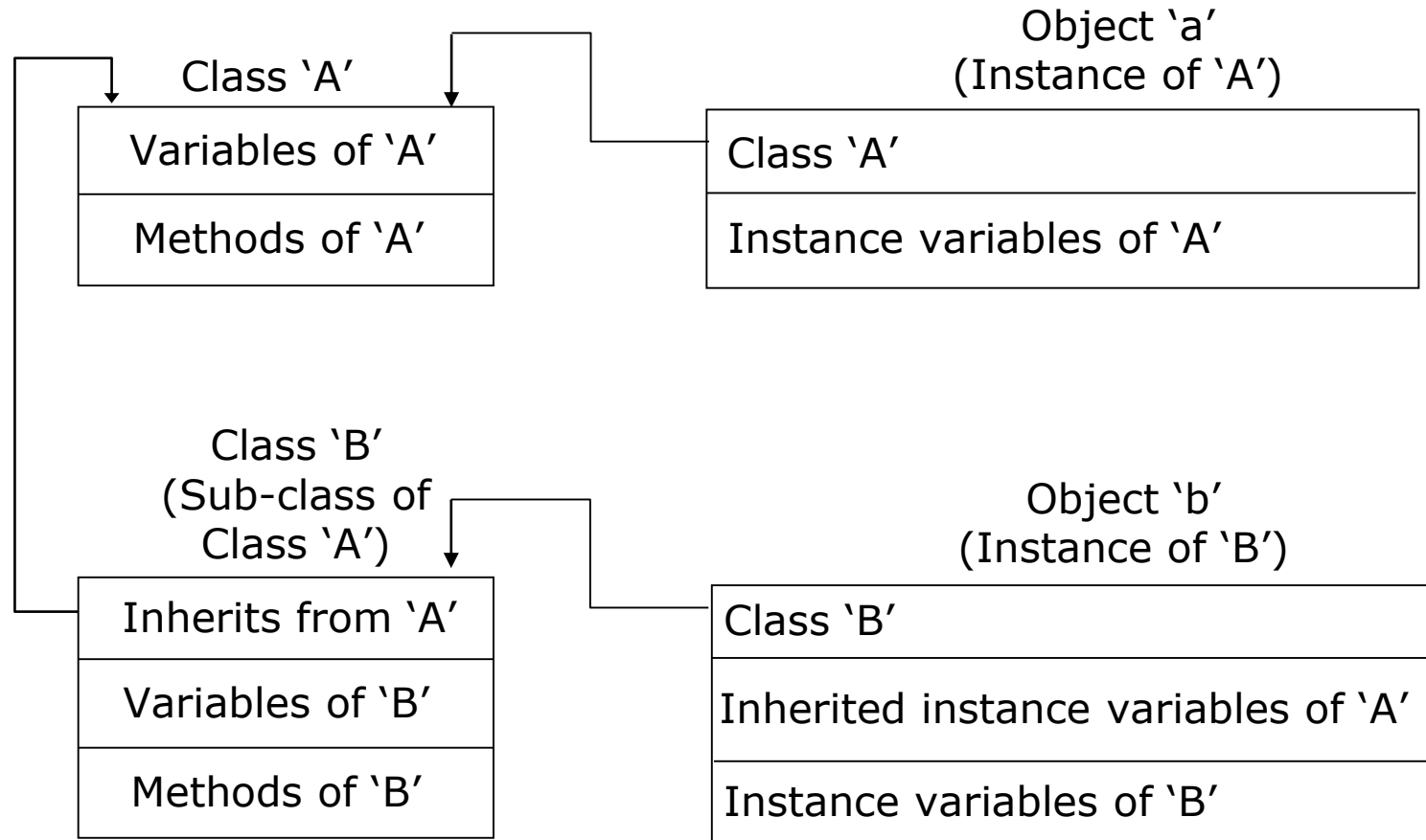
## ■ Inheritance

- Inheritance is a mechanism to share code and behavior
- A child class inherits all of the instance variables, instance methods, and class methods of its parent class
- A child class can inherit from multiple parent classes. This is called *multiple inheritance*

# Example of Class, Instance, and Inheritance



# Example of Objects, Class, and Inheritance

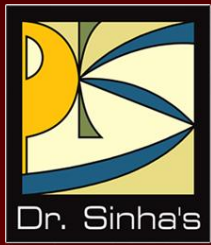


# Key Elements of Object-Oriented Paradigm



Sr. No.	Element	Meaning
1	Abstraction	<i>Abstraction</i> means that an object's characteristics are broken into manageable chunks as defined and documented in its class description.
2	Encapsulation	<i>Encapsulation</i> stipulates that code and data are stored as one unit. Encapsulation also enables selective or total information hiding, since it can make portions of the code and data inaccessible from outside the unit.
3	Modularity	<i>Modularity</i> defines the unit reuse. These units group abstractions together.
4	Hierarchy	<i>Hierarchy</i> allows an object's behaviors to be refined (subclasses) without recoding of the parent object (the superclass). Some OO languages allow an object to have more than one superclass, a feature that is known as multiple inheritance. Inheritance hierarchies enable ranking/ordering of abstractions.
5	Messages	<i>Messages</i> , similar in use to function calls, are requests to perform an operation on an instantiated object.





# Some High-level Languages



# FORTRAN



- Stands for **FORM**ula **TRAN**slation
- Originally developed by John Backus and his team at IBM followed by several revisions
- Standardized by ANSI as FORTRAN 77 and FORTRAN 90
- FORTRAN 2018 is the latest released version till date
- Designed for solving scientific & engineering problems
- Oriented towards solving problems of a mathematical nature
- Popular language amongst scientists and engineers

# COBOL



- Stands for **CO**mmon **B**usiness **O**riented **L**anguage
- Originally developed started under Grace Hopper followed by COncference on DAta SYstems Languages (CODASYL)
- Standardized by ANSI as COBOL 74, COBOL 85, and COBOL 2002
- COBOL 2014 is the latest COBOL standard as on date
- 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

# BASIC



- Stands for **B**eginners **A**ll-purpose **S**ymbolic **I**nstruction **C**ode
- 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 was a widely used language on personal computers
- Flexible and reasonably powerful language and can be used for both business and scientific applications

# 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 programming 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

## 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

# C++ (C plus plus)



- 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
- Being a superset of C, it is an extremely powerful and efficient language. However, it is more difficult to learn than C

# 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

# Python



- A high level programming language, first released in 1991 by Guido van Rossum of Netherlands
- Being a Free and Open Source Software (FOSS), it has a large, world-wide development community
- It is now managed by Python Software Foundation, a non-profit community
- It now ranks among top ten popular programming languages

# Key Features of Python Include



- Simple to learn and easy to read and understand
- Both procedure-oriented and object-oriented programming is possible
- Highly portable
- Interpreted language
- Extensible and embeddable language
- Rich library support

# LISP

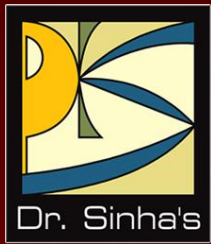


- Stands for **LIS**t **P**rocessing
- Developed in 1959 by John McCarthy of MIT
- Designed to have features for manipulating non-numeric data, such as symbols and strings of text
- Due to its powerful list processing capability, it is extensively used in the areas of pattern recognition, artificial 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



# Selecting a Language for Coding an Application



# Characteristics of a Good Programming Language



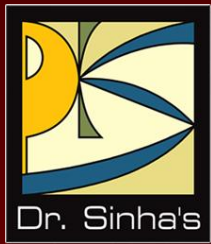
- Simplicity
- Naturalness
- Abstraction
- Efficiency
- Structured Programming Support
- Compactness
- Locality
- Extensibility
- Suitability to its environment

# 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



# Subprogram

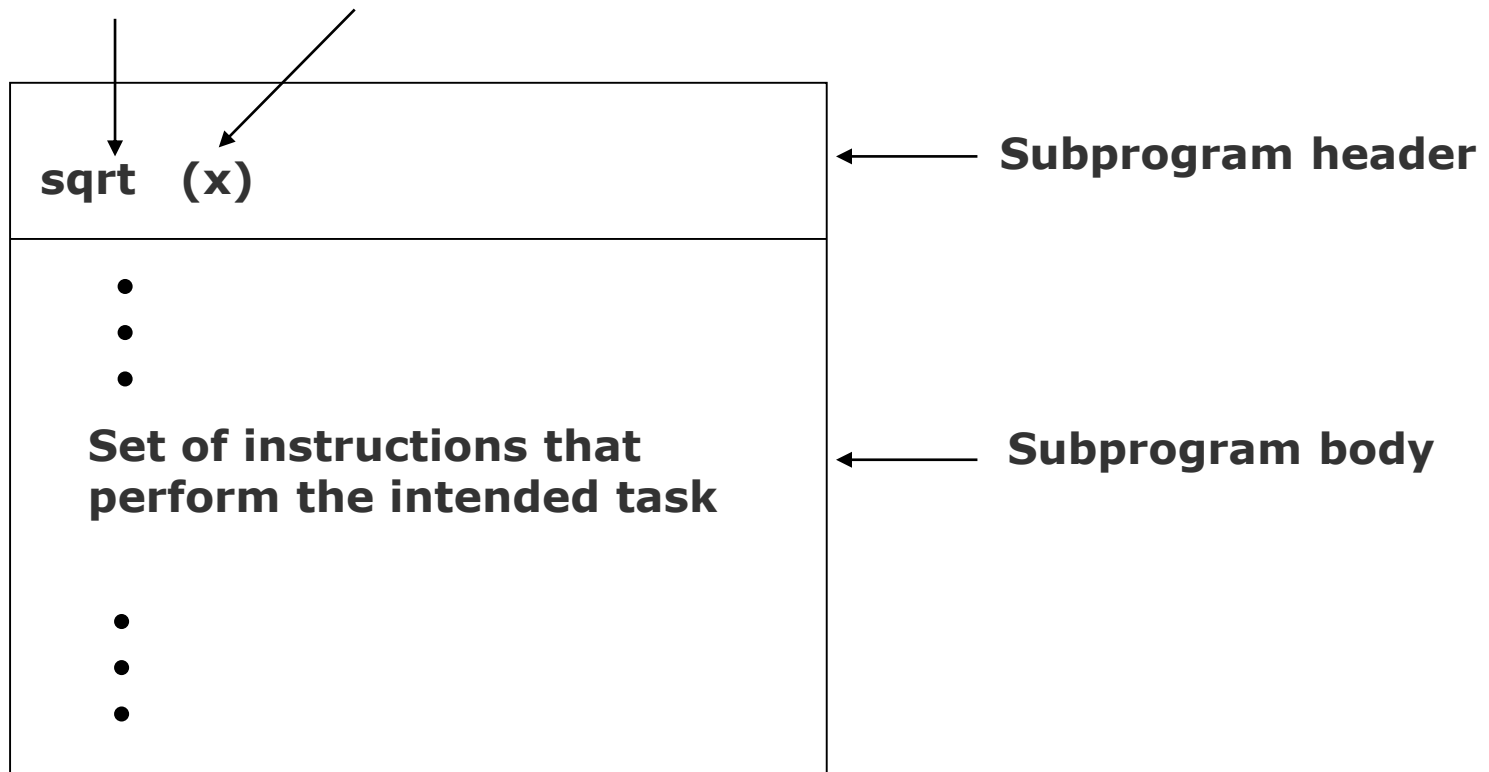


- Program written in a manner that it can be brought into use in other programs and used whenever needed without rewriting
- Also referred to as *subroutine*, *sub-procedure*, or *function*
- Subprogram call statement contains the name of the subprogram followed by a list of parameters enclosed within 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

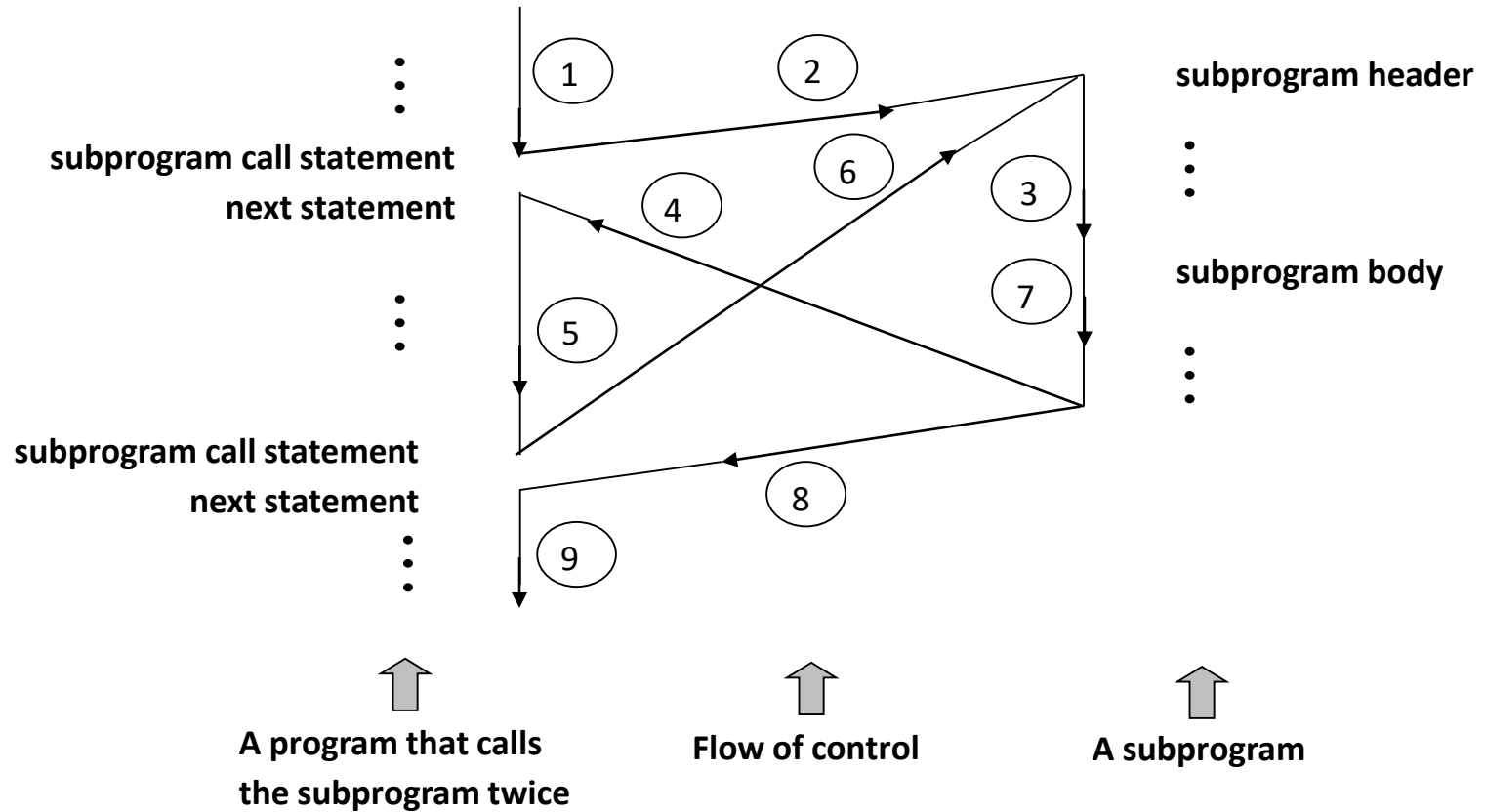
# Structure of a Subprogram



Subprogram name    Parameter



# Flow of Control in Case of Subprogram Calls



# Key Words/Phrases



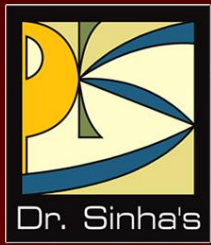
- Assembler
- Assembly language
- BASIC
- Built-in function
- C
- C++
- C#
- COBOL
- Coding
- Compiler
- Computer language
- FORTRAN
- Function
- High-level language
- HotJava Interpreter
- Intrinsic subprogram
- Intermediate compiler and Interpreter
- Java
- Just-in-time compilation
- Language processor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

*(Continued on next slide)*

# Key Words/Phrases



- SNOBOL
- Source program
- Sub-procedure
- Subprogram
- Subroutine
- Symbolic language
- Syntax error
- Syntax rules
- Programmer-written subprograms



# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 13

**System  
Implementation  
and Operation**



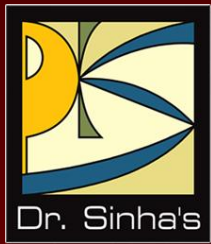
# Learning Objectives



## In this chapter you will learn about:

- Main activities of implementation and operation phase of software
- Software testing and debugging
- Software documentation
- Software deployment and changeover processes
- System evaluation and
- Software maintenance





# Program Errors, Testing and Debugging



# 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

# 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)

# Unit and Integrated Testing



- In *unit testing*, each module of software is tested independently for its functionality by the programmer who developed the module.
- In *integrated testing*, the team integrates the independently developed and tested modules into a complete system, and tests the integrated system to check if all modules coordinate properly.

# Alpha and Beta Testing



- Software internally released for testing is known as *alpha version* and the test conducted on it is called *alpha testing*
- Software released for additional testing to a selected set of external users is *beta version* and test conducted on it called is *beta testing*



# Difference Between Testing and Debugging



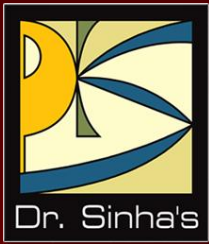
Sr. No.	Testing	Debugging
1	<ul style="list-style-type: none"> <li>▪ Testing is the process of validating the correctness of a program</li> <li>▪ Its objective is to demonstrate that the program meets its design specifications</li> </ul>	<ul style="list-style-type: none"> <li>▪ Debugging is the process of eliminating errors in a program</li> <li>▪ Its objective is to detect the cause of error 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 are completed</li> </ul>	<ul style="list-style-type: none"> <li>▪ Debugging is complete when all known errors in the program are fixed</li> <li>▪ Note that debugging process ends only temporarily because it restarts whenever a new error is detected in the program</li> </ul>

*(Continued on next slide)*

# Difference Between Testing and Debugging



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 is carried out whenever errors are detected 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 are carried out 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 ready</li> <li>▪ The approach used for debugging largely depends on the personal choice of the programmer and the type of error in the program</li> </ul>



# Software Documentation



# What is Documentation?



- It is the process of collecting, organizing, storing, and maintaining a complete historical record of programs and other documents used or prepared during the different phases of the life cycle of software
- It is an on-going process that starts as early as in the study phase of the software and continues until its implementation and operation phase
- Maintenance team has to carry out documentation from time-to-time, whenever it modifies the software

# Need for Documentation



- Solves the problem of indispensability of an individual for an organization
- Makes software easier to modify and maintain in future
- Helps in restarting a software project postponed earlier due to some reason

# Forms of Documentation



## ■ Requirement specification document

- Before developing some software, the development team must analyze its requirements and document them
- Requirement specification document is an outcome of this exercise
- Specifies the objectives of developing the software and its usefulness to various categories of users
- Specifies the functionalities, capabilities, performance, inputs, and outputs requirements of the software
- Specifies what all the software will do when developed

## ■ Design document

- Defines overall architecture of the software
- Specifies various hardware and software components of the software and interfaces between the components

*(Continued on next slide...)*

# Forms of Documentation



- Includes a hierarchy of software components, rules for component selection, and interfaces between the components
- **Comments**
  - From maintenance point of view, comments are necessary
  - All high-level languages provide the facility to write comments in the source code of a program
  - Use of this facility by programmers for proper documentation of their programs is highly recommended
  - Comments should be used intelligently to improve the quality and understandability
  - Comments should not be redundant, incorrect, incomplete, or written in a manner that is difficult to understand

*(Continued on next slide...)*

# Forms of Documentation



## ■ System manual

- Standard system manuals contain the following information:
- Specific module names along with their description and purpose
- Detailed system flow charts and program flow charts for each module
- Description of the program listings and the control procedures
- Source listing of all the programs with full details of all modifications
- Specifications of all input and output media
- Specimen of all types of input and output
- File layout
- Structure and description of all test data, test results, storage dumps, trace program printouts, etc.

*(Continued on next slide...)*



# Forms of Documentation



## ■ User manual

- User manual describes how to use the software
- User manual must contain following information:
  - Installation and operational details of software
  - Loading and unloading procedures
  - Starting, running, and terminating procedures
  - Description and example of any control statements used
  - All console commands with errors and console messages, their meaning, reply, and/or operation action
  - List of error conditions with explanation for their re-entry into the system
  - List of programs, which users must execute before and after execution of each program
  - Special checks (if any) and security measures, etc.

*(Continued on next slide...)*

# Forms of Documentation



- **Commonly used ways to describe each feature of software in a user manual are:**
  - **Commands/Functions list**
    - It lists the commands or functions of the software alphabetically or grouped logically
  - **Tutorial**
    - It contains tutorials for various functionalities or tasks of software
  - **Online help**
    - It contains online information, which users can use whenever needed while using the software without the need to refer to any other document

# Forms of Documentation



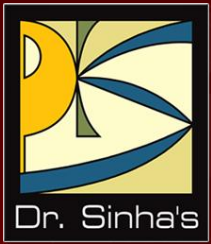
- **Marketing document:** It consists of the following:
  - Usefulness of the software product
  - Its salient features
  - Its comparison with other competing products
  - Its system requirements (hardware, operating system, etc.)

# Documentation Standard



## It deals with:

- How to do documentation?
- How to choose meaningful program variable names?
- How to design the GUI?
- How and up to what detail to include comments?
- What diagrams, charts, reports, outputs, etc. are necessary for completing documentation?



# Software Deployment



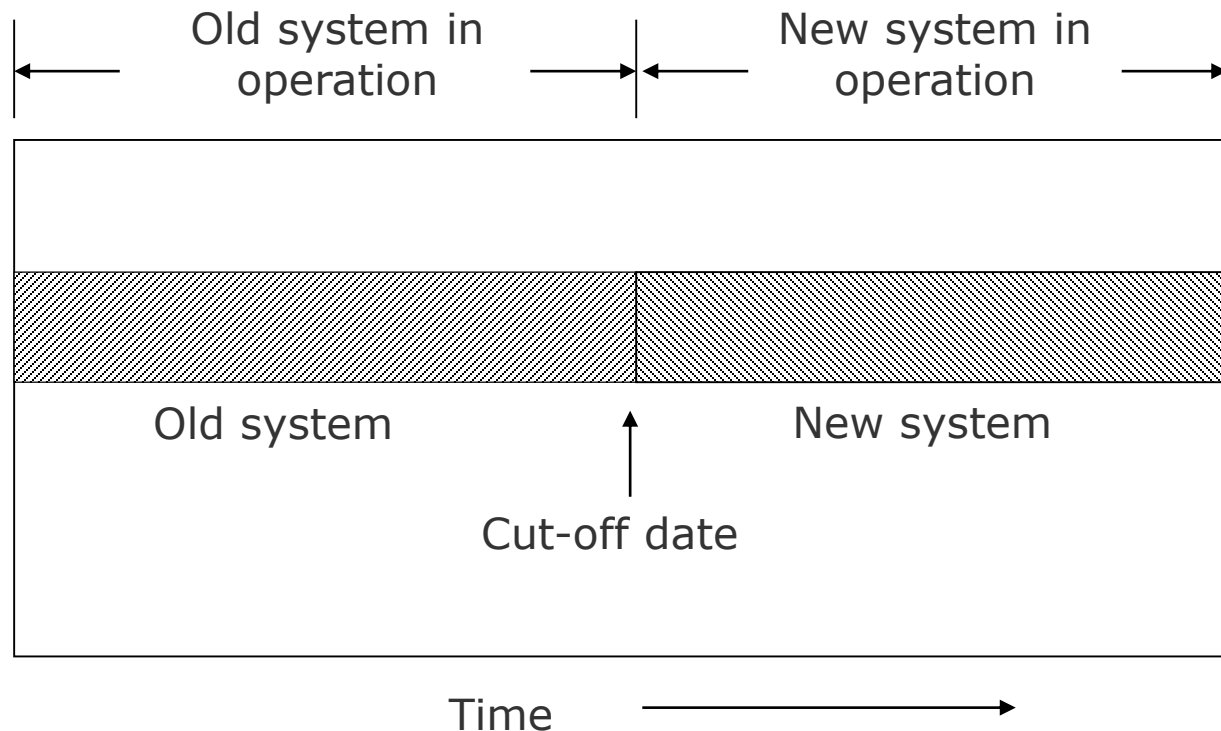
# Changeover to the New System



- 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

*(Continued on next slide)*

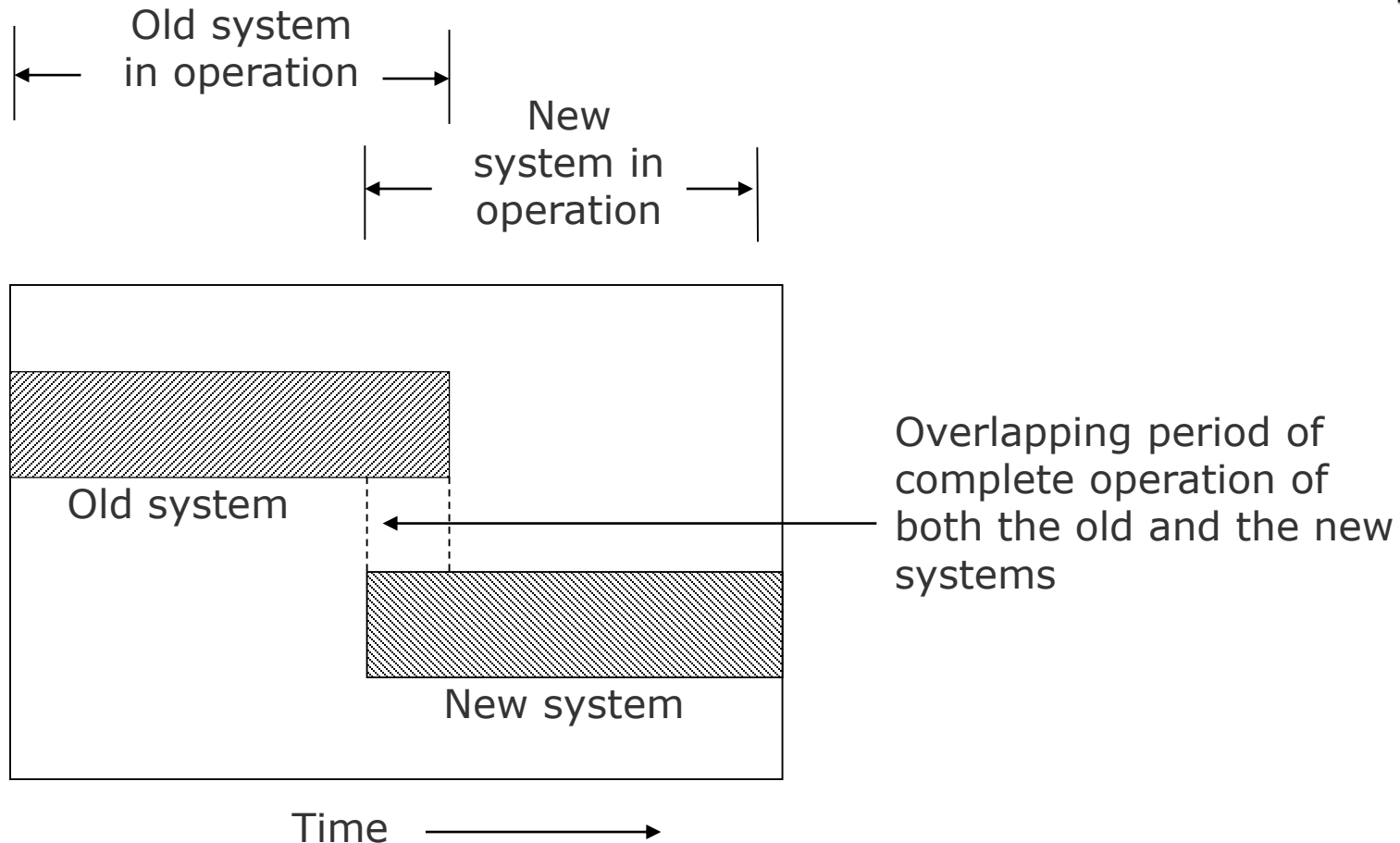
# Changeover to the New System



**(a) Immediate changeover**

*(Continued on next slide)*

# Changeover to the New System

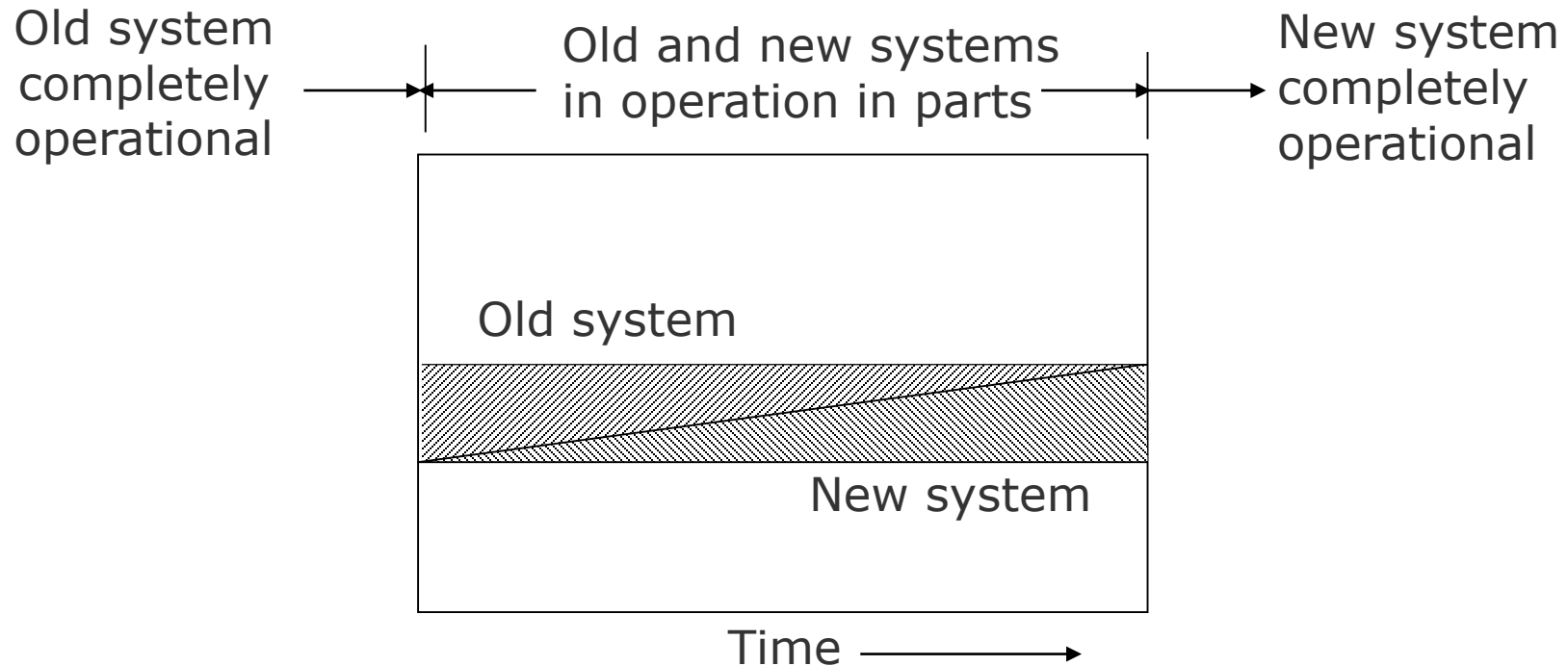


**(b) Parallel run**

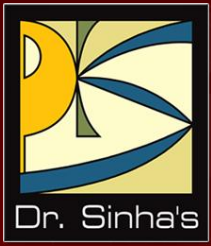
*(Continued on next slide)*



# Changeover to the New System



**(c) Phased conversion**



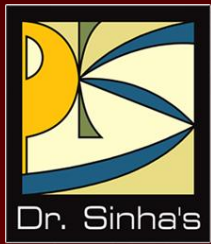
# System Evaluation



# 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



# Software Maintenance



# Definition



- Process of modifying software system or component after deployment to correct faults, add functionality, improve performance or other attributes, or adapt to a change in environment

# Need for Software Maintenance



- Any software needs modification from time-to-time due to one or more of the following reasons:
  - Changes in business conditions or operations of the organization
  - Changes in organizational policies or enforcement of new laws
  - Changes in user needs
  - Changes in technology

# Importance of Software Maintenance



- Maintenance is an important phase in SDLC
- On an average, maintenance cost of software systems is two to four times the development cost

# Controlling Modifications



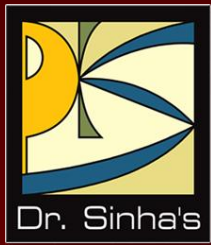
- Frequent change is disrupting and disturbing
- Some control over changes is required for which a change control board is constituted
- Change control board evaluates all requests for change and approves major changes
- It need not approve normal maintenance operations
- Major changes are those that alter the system significantly
- Whenever a programmer modifies a program, the concerned members must also modify the associated documents



# Key Words/Phrases



- Alpha testing
- Beta testing
- Bugs
- Changeover operations
- Comments
- Debugger
- Debugging
- Documentation
- Immediate changeover
- Integrated testing
- Logic errors
- Memory dump
- Parallel run
- Phased conversion
- Software maintenance
- Syntax errors
- System evaluation
- System manual
- Testing
- Unit testing
- User manual



# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 14

**Operating Systems**

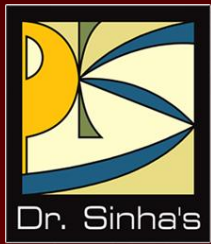


# Learning Objectives



## 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
  - Device management
  - Security
  - Command interpretation
- Some commonly used OS capability enhancement software
- Some popular operating systems



# Definition, Need and Functions of an OS

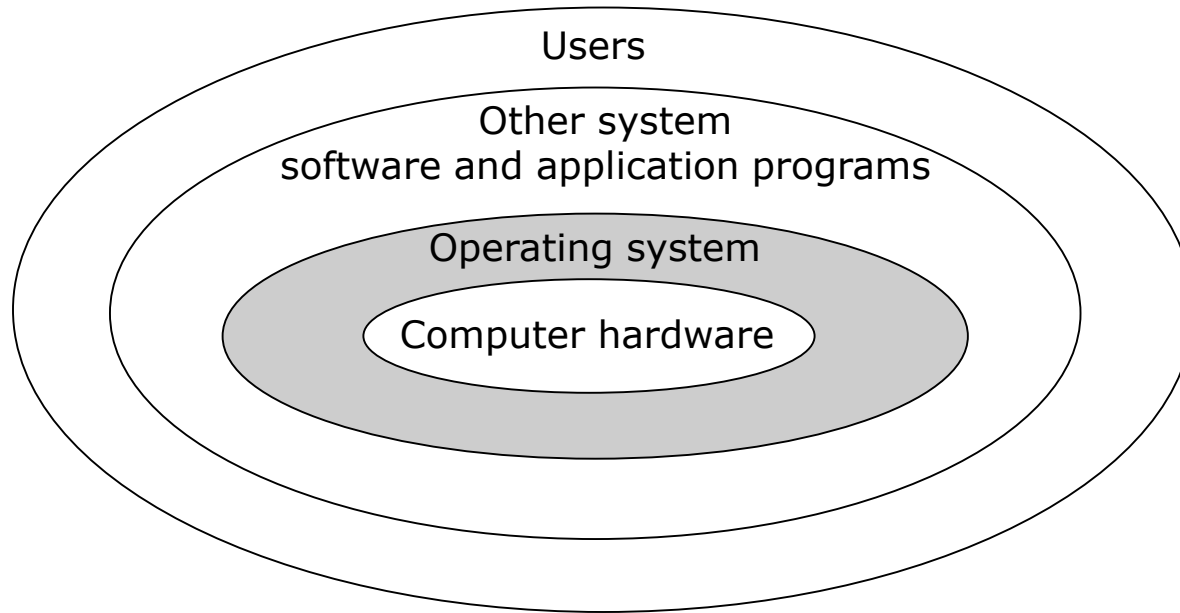


# Definition and Need for OS



- 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



Operating system layer hides details of hardware from programmers and other users and provides them with a convenient interface for using the system

# Main Functions of an OS



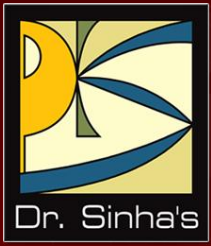
- Process management
- Memory management
- File management
- Device management
- Security
- Command interpretation

# 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



# Process Management



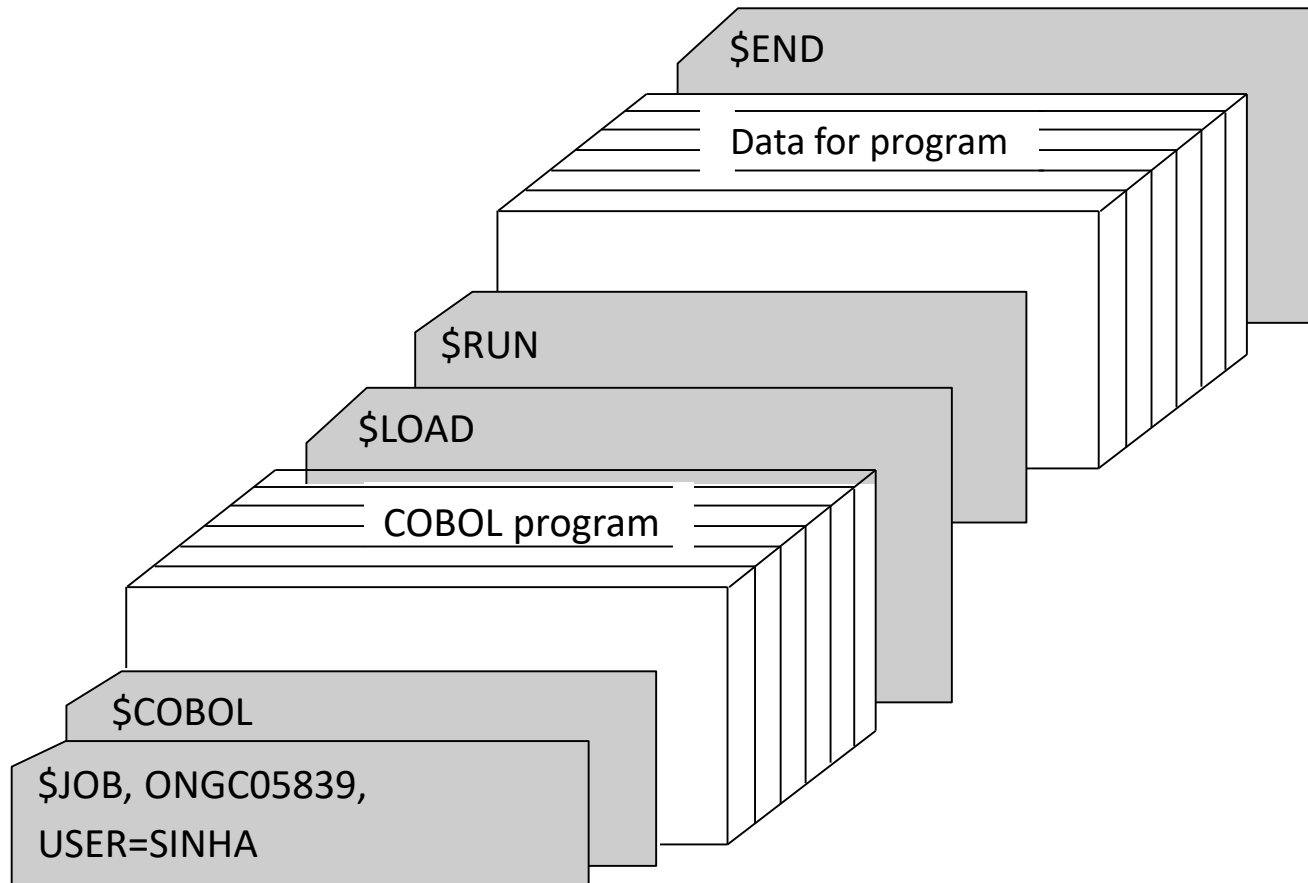
- A **process** (also called **job**) is a program in execution
- **Process management** module of an operating system manages the processes submitted to a system in a manner to minimize *idle time* of processors (CPUs, I/O processors, etc.) of the system

# Process Management Mechanisms in Early Systems

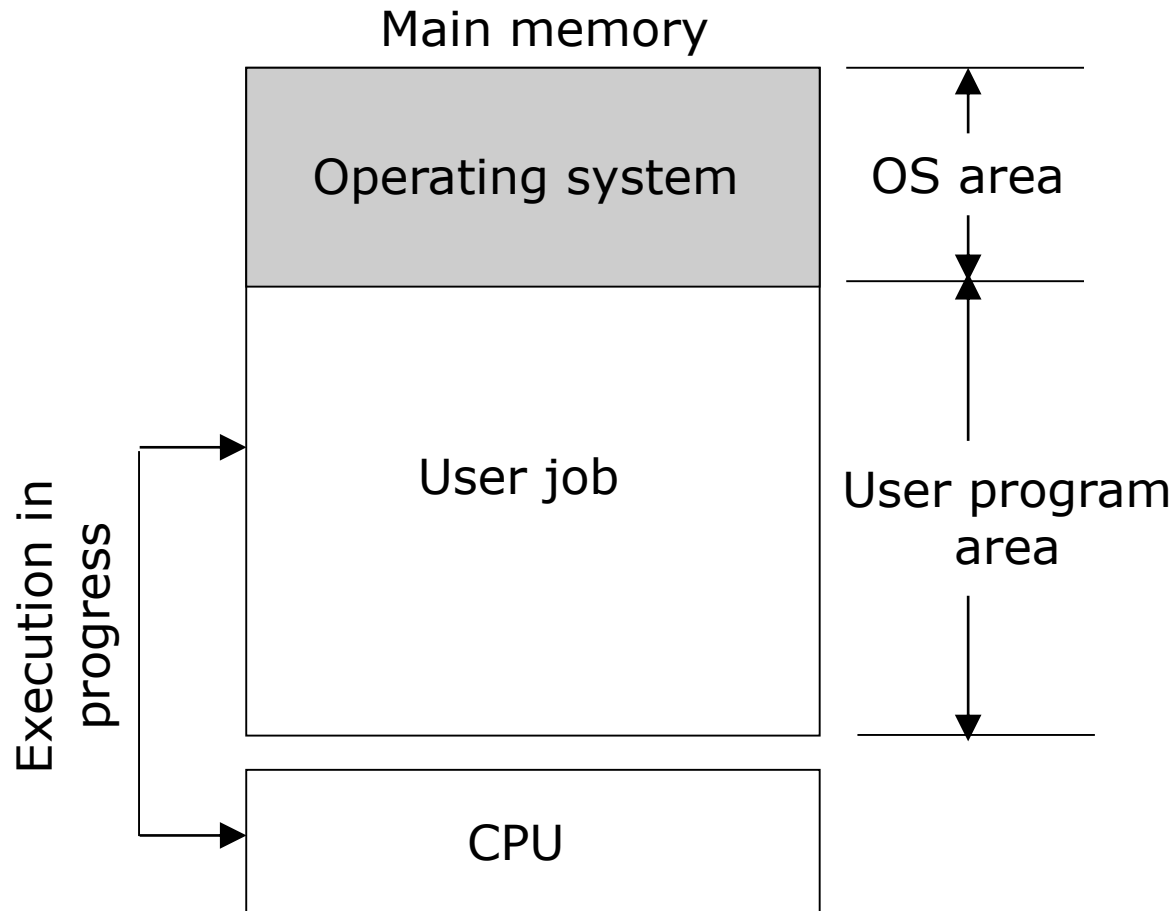


- **Manual loading mechanism:** Jobs were manually loaded one after another in a computer by the computer operator
- **Batch processing mechanism:** Batch of jobs was submitted together to the computer and job-to-job transition was done automatically by the operating system
- **Job Control Language (JCL):**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

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



# Uniprogramming



(Continued on next slide...)

# Uniprogramming



- A job does not need CPU for entire duration of its processing
- Depending on CPU utilization during the course of processing, jobs are of two types
  - *CPU-bound jobs*, which mostly perform computations with little I/O operations
  - *I/O-bound jobs*, which mostly perform I/O operations with little computation
- In a *uniprogramming system*, CPU is idle whenever the currently executing job performs I/O operations

# Multiprogramming

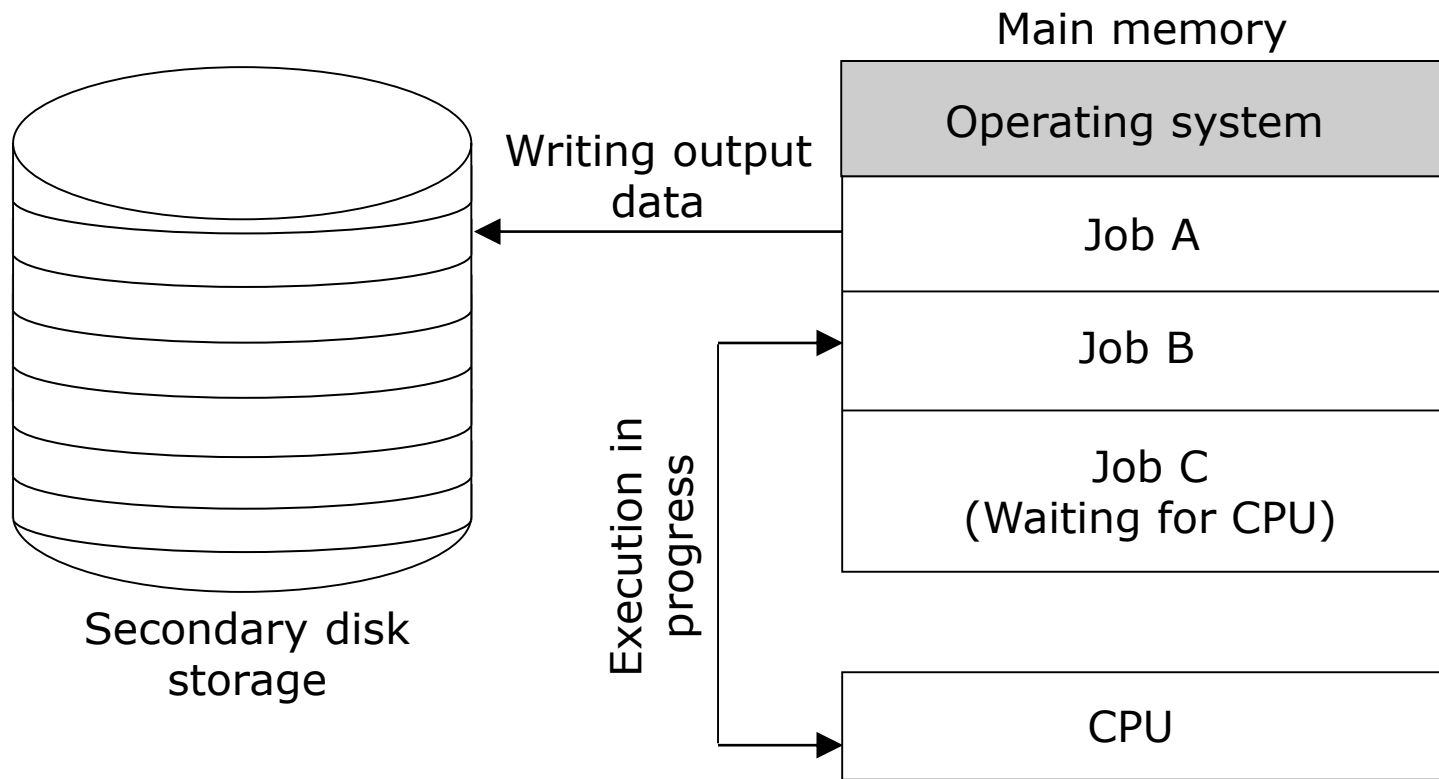


- *Multiprogramming* is interleaved execution of two or more different and independent programs by a computer
- Multiprogramming enables two or more user programs to reside simultaneously in main memory and carries out their interleaved execution
- With multiple user programs residing simultaneously in main memory, whenever a user program that was executing goes to perform I/O operations, the operating system allocates CPU to another user program in main memory
- In multiprogramming, several user programs share CPU time to keep it busy
- Note that multiprogramming does not mean execution of instructions from several programs simultaneously

*(Continued on next slide...)*



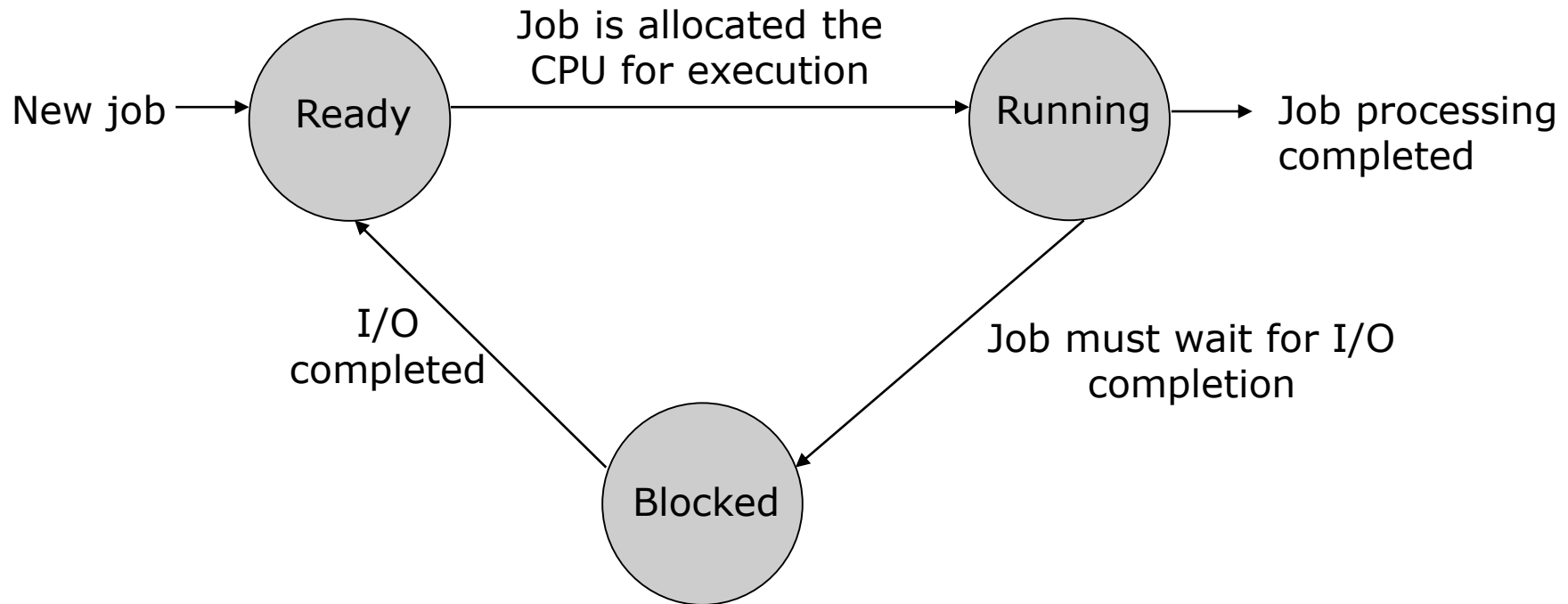
# Multiprogramming



A typical scenario of jobs in a multiprogramming system



# Multiprogramming



Three different states of jobs in main memory in a multiprogramming system

# 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



- Multitasking is single-user variation of multiprogramming concept
- Both refer to the same concept of a system's capability to work concurrently on more than one task
- Some authors prefer to use the term multiprogramming for multi-user systems and multitasking for single-user systems
- Multitasking eases user operation and saves lots of time when a user has to switch between two or more applications while performing a job
- *Multiprogramming* is interleaved execution of multiple jobs in a multi-user system, while *multitasking* is interleaved execution of multiple jobs in a single-user system

# Multithreading



- *Threads* are a popular way to improve application performance
- In traditional operating systems, the basic unit of CPU utilization is a process
- Each process has its own program counter, its own register states, its own stack, and its own address space
- In operating systems with threads facility, the basic unit of CPU utilization is a thread
- A process consists of an address space and one or more threads of control

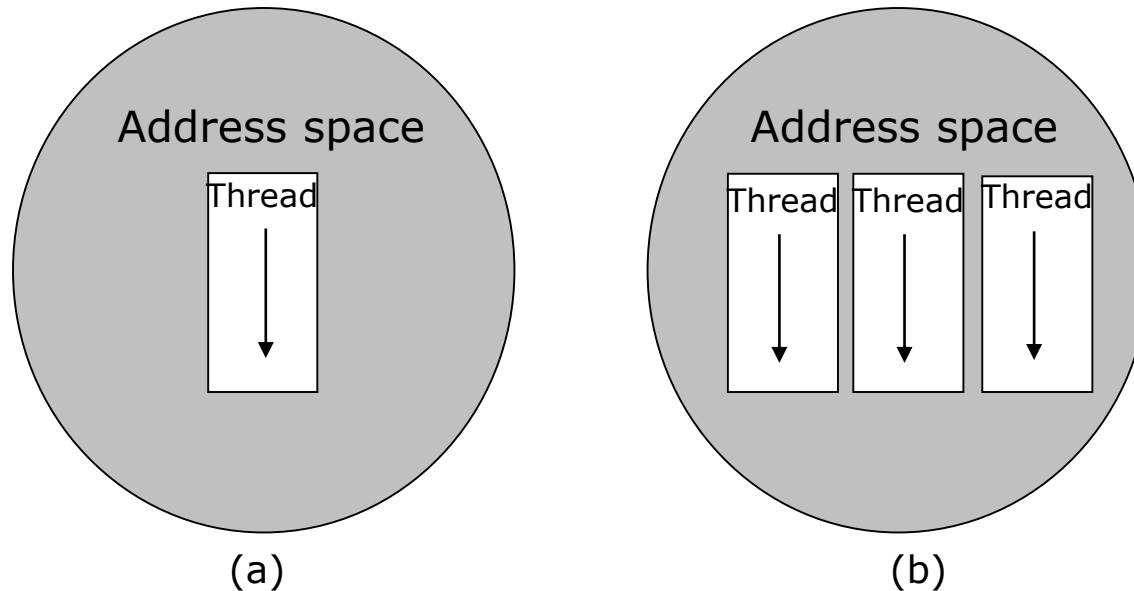
*(Continued on next slide...)*

# Multithreading



- Each thread of a process has its own program counter, its own register states, and its own stack
- All the threads of a process share the same address space
- All threads of a process also share the same set of operating system resources
- Threads are often referred to as *lightweight processes* and traditional processes are referred to as *heavyweight processes*

# 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],

# Motivations for Using Threads



- Overhead involved in creating a new process is considerably greater than that for creating a new thread within a process
- New thread uses the address space of its process
- Overhead involved in CPU switching among peer threads is very small as compared to CPU switching among processes
- Resources are shared more efficiently among multiple threads of a process than among multiple processes
- Users find the threads model more intuitive for application programming

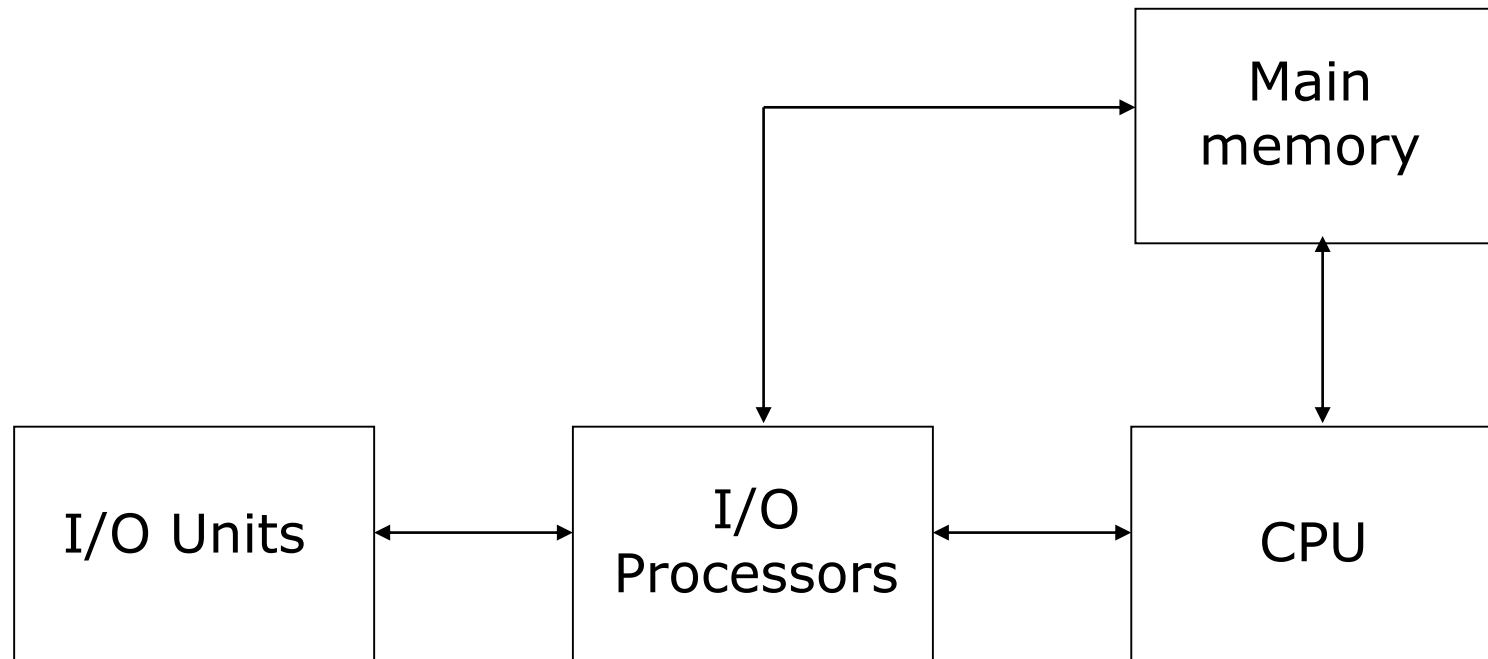


# Multiprocessing

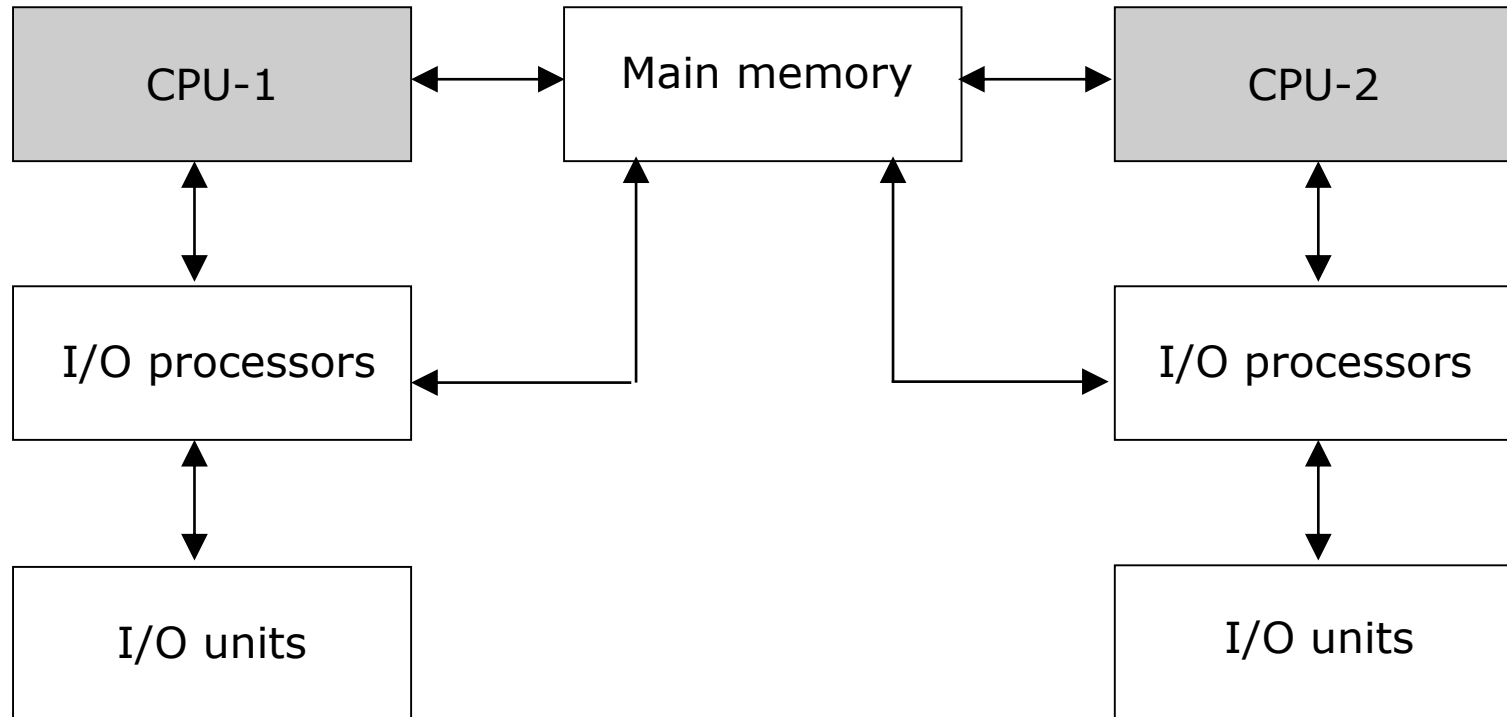


- System with two or more CPUs having ability to execute multiple processes concurrently
- Multiple CPUs are used to process either instructions from different and independent programs or different instructions from the same program simultaneously
- Types of multiprocessing:
  - *Tightly-coupled*: Single 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



# Difference between Multiprogramming and Multiprocessing



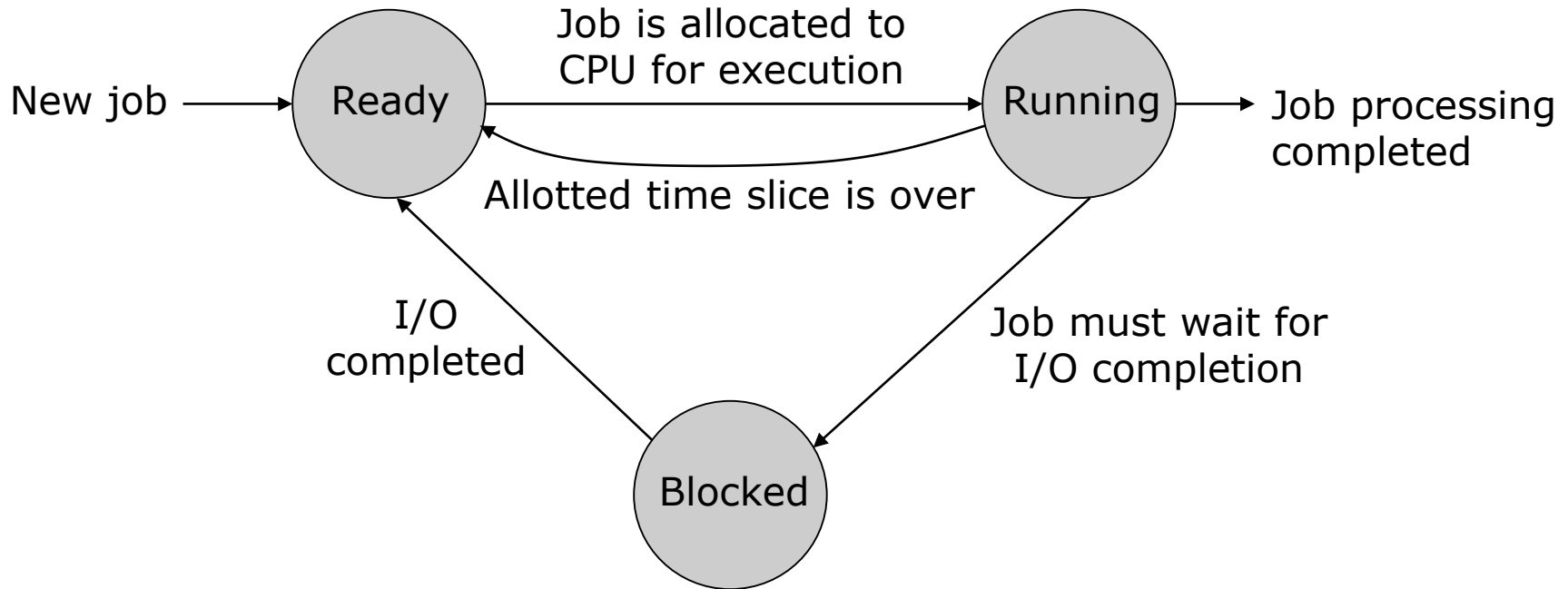
- *Multiprocessing* is simultaneous execution of two or more processes by a computer system having more than one CPU
- *Multiprogramming* is interleaved execution of two or more processes by a single-CPU system
- Multiprogramming involves execution of a portion of one program, then a portion of another, etc., in brief consecutive periods
- Multiprocessing involves simultaneous execution of several program segments of the same or different programs

# 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
- Many user terminals are 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



Process state diagram for a time-sharing system

# Requirements of Time-sharing Systems



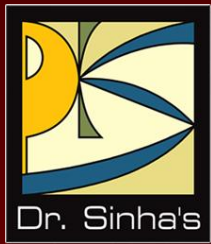
- Time-sharing systems require following additional hardware and software
  - Number of user terminals
  - Large memory to support multiprogramming
  - Memory protection mechanism to prevent a job's instructions and data from other jobs
  - Job status preservation mechanism to preserve a job's status information when the operating system takes away CPU from it, and restores this information back
  - Special CPU scheduling algorithm that allocates CPU for a short period one-by-one to each user process
  - Interrupt mechanism to send an interrupt signal to CPU after every time slice

# 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 Management



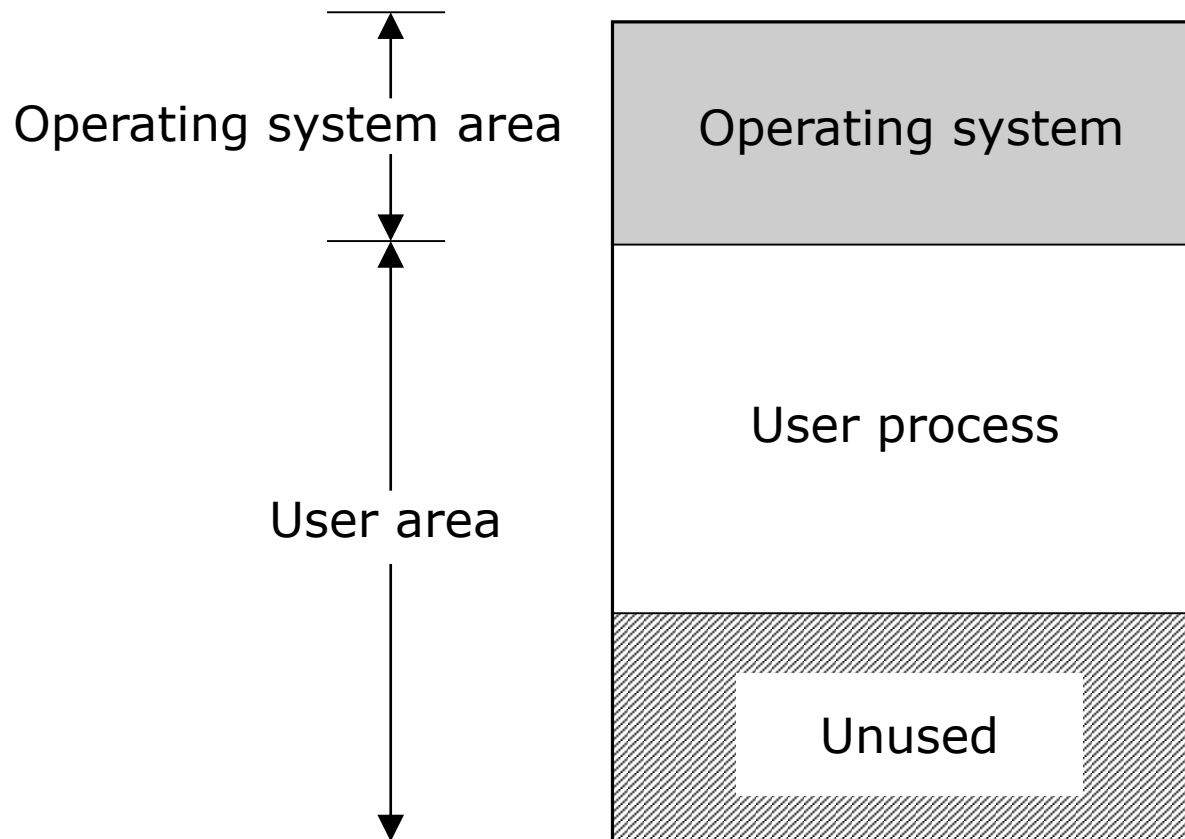
- Memory is important resource of a computer system that must be properly managed for the overall system performance
- Memory management module:
  - Keeps track of parts of memory in use and parts not in use
  - Allocates memory to processes as needed and deallocates when no longer needed

# 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

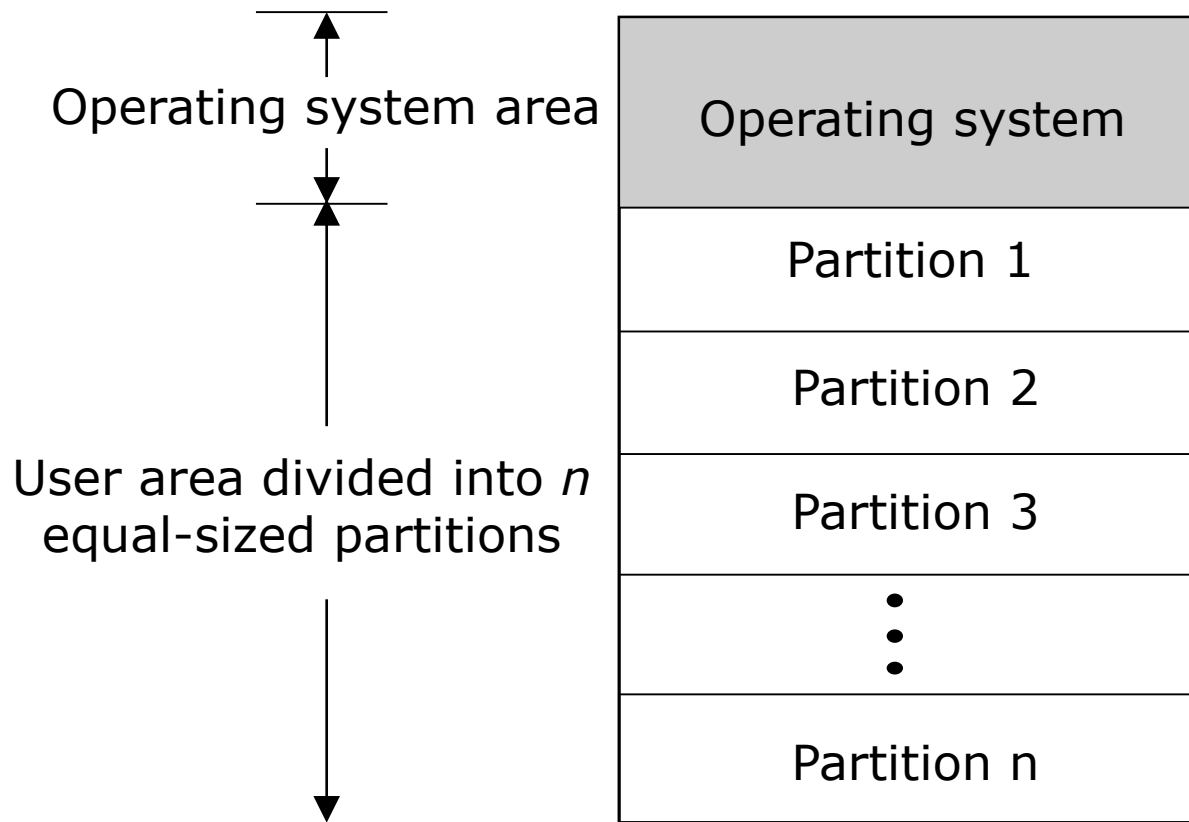


# Multiprogramming Memory Models

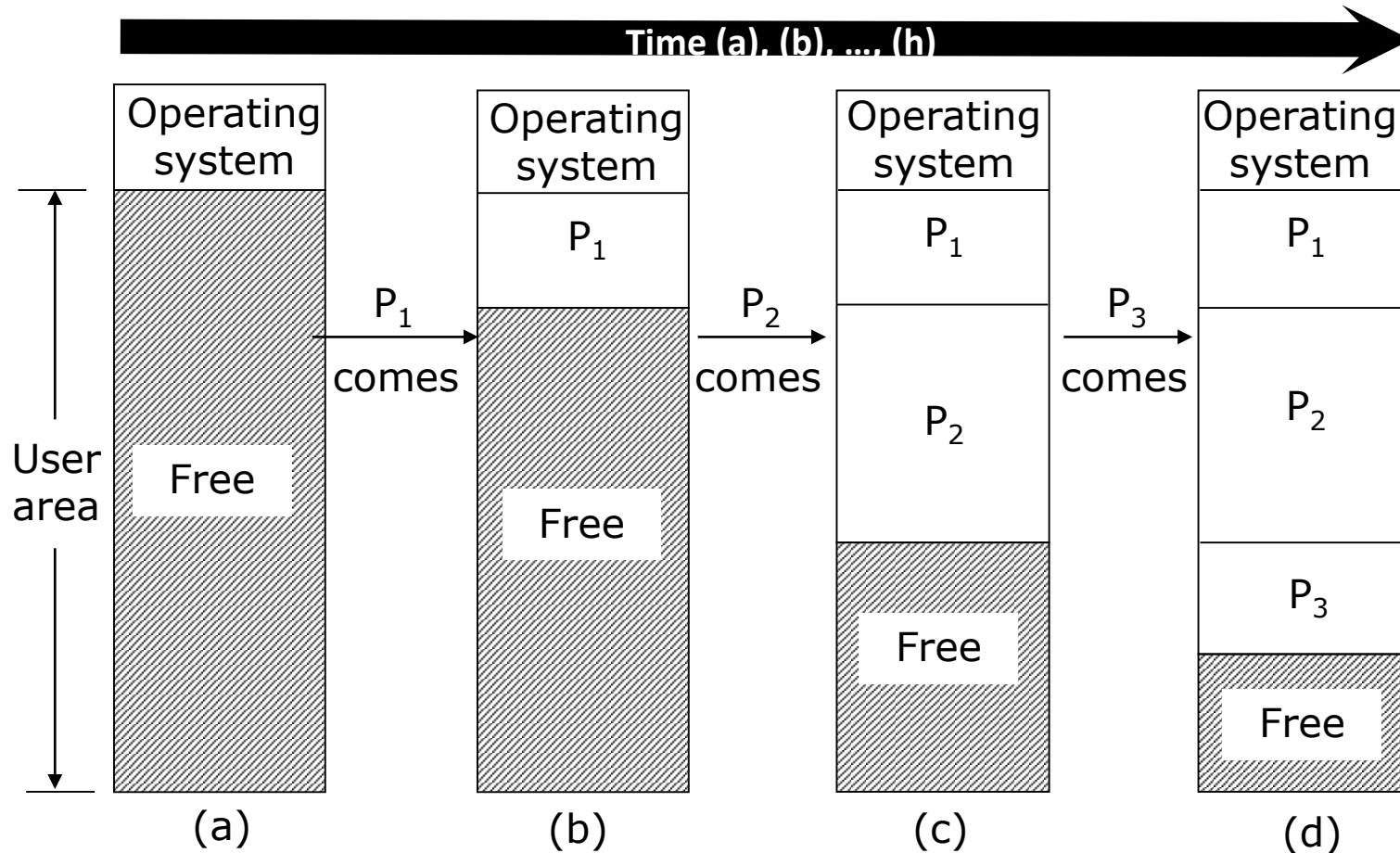


- In a multiprogramming system, multiple user processes can reside simultaneously in main memory
- Two memory management schemes used to facilitate this are:
  - *Multiprogramming with fixed number of memory partitions:* User area of the memory is divided into a number of fixed-sized partitions
  - *Multiprogramming 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

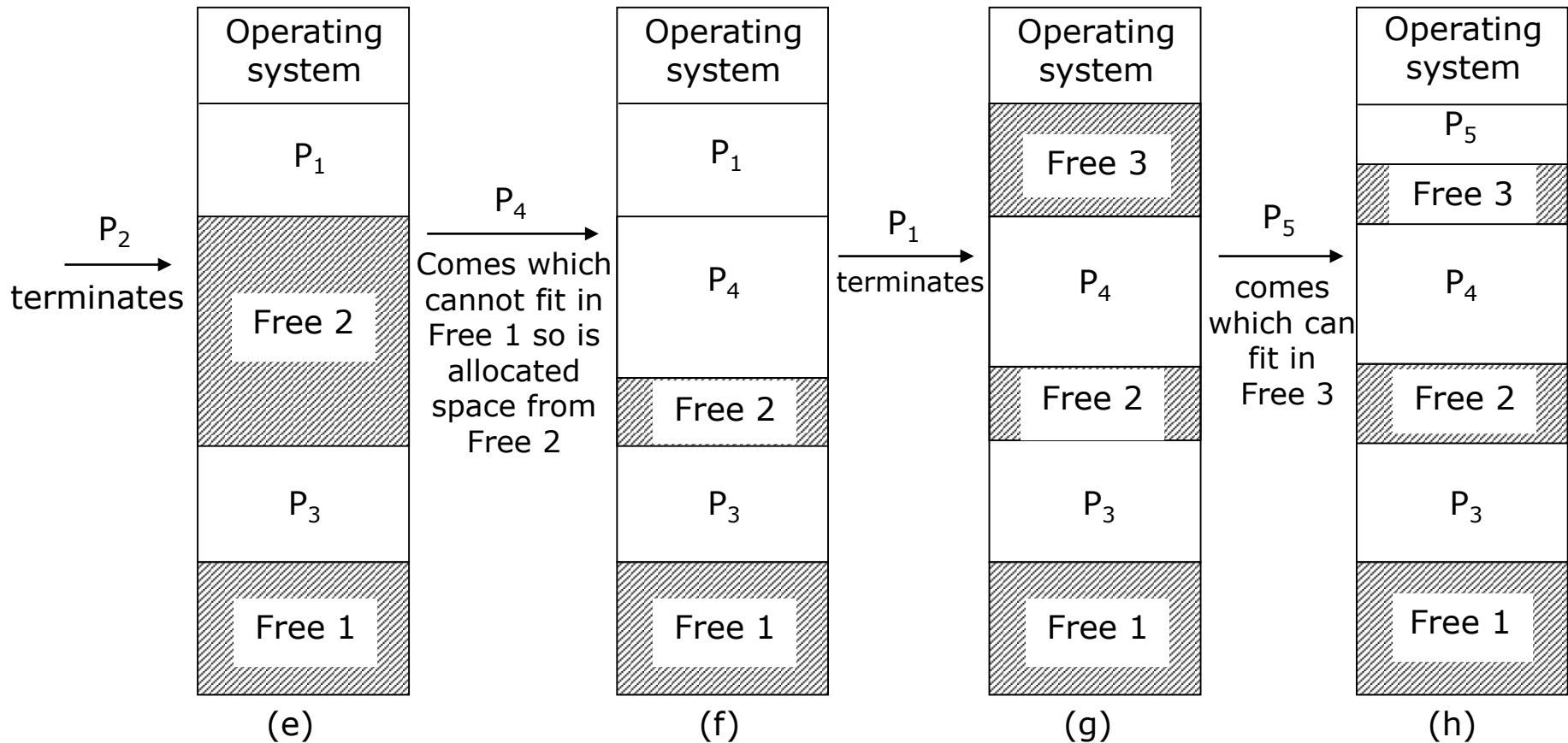


# Multiprogramming with Fixed Number of Memory Partition



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



- Conventional memory management schemes suffer from two main limitations
  - Operating system cannot load a process until sufficient free memory for loading the entire process becomes available
  - Operating system cannot load a process if main memory size is less than the total memory required
- *Virtual memory* is a memory management scheme that overcomes these limitations by allowing execution of a process without the need to load the process in main memory completely

# How is Virtual Memory Realized?



- On-line secondary storage
  - On-line secondary storage device having much larger capacity than main memory
- Swapping
  - *Swapping* is the process of transferring a block of data from on-line secondary storage to main memory or vice-versa
- Demand paging
  - Instead of loading an entire process before its execution can start, the operating system uses a swapping algorithm
  - When operating system needs to swap to continue a process's execution, it invokes a page-replacement algorithm to create one for the accessed page
  - Page replacement deals with selecting a page that is residing in memory but is not in use currently

(Continued on next slide...)

# How is Virtual Memory Realized?



- *Virtual memory* is often described as a hierarchy of two storage systems – one is a low-cost, large-capacity, low-speed system (on-line disk storage), and the other is a high-cost, small-capacity, high-speed system (main memory)

# Advantages of Virtual Memory

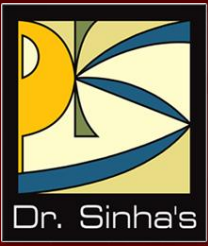


- Provides a large virtual memory to programmers on a system having smaller physical memory
- Enables execution of a process on a system whose main memory size is less than the total memory required by the process
- Enables a process's execution to be started even when sufficient free memory for loading the entire process is not available
- Makes programming easier as programmers 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



# 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



- Sequential access files
  - Operating systems use sequential access files for storage of files on sequential access storage media
  - A process can read the bytes or records in the file in the order in which they are stored
- Random access files
  - Operating systems use random access files for storage of files on random access storage media
  - Applications can access the contents of a random access file randomly, irrespective of the order in which the bytes or records are stored



# File Operations (Examples)



File operation	Usage
Create	Is used to create a new file.
Delete	Is used to delete an existing file that is no longer needed.
Open	Is used to open an existing file when a user wants to start using it.
Close	Is used to close a file when the user has finished using it.
Read	Is used to read data stored in a file.
Write	Is used to write new data in a file.
Seek	Is used with random access files to first position read/write pointer to a specific place in file so that data can be read from, or written to, that position.
Get attributes	Is used to access the attributes of a file.
Set attributes	Is used to change user-settable attributes (such as, protection mode) of a file.
Rename	Is used to change name of an existing file.
Copy	Is used to create a copy of a file, or to copy a file to an I/O device, such as a printer.

# 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)



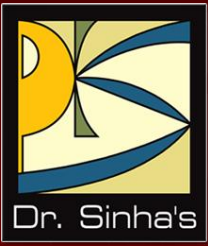
File extension	Its meaning
.bas	Basic source program file
.c	C source program file
.ftn	Fortran source program file
.pas	Pascal source program file
.obj	Object file (compiler output, not yet linked)
.bin	Executable binary program file
.lib	Library of .obj files used by the linker
.dat	Data file
.hlp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

*(Continued on next slide...)*

# File Extensions (Example)



File extension	Its meaning
.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



# Device Management



# Controlling I/O Devices



- Computer uses device controllers to connect I/O devices to it
- Each device controller is in charge of and controls a set of devices of a specific type
- Device controller maintains some local buffer storage and is responsible for moving data between an I/O device that it controls and its local buffer storage
- Device controller also has a few registers that it uses for communicating with CPU
- These registers are part of the regular memory address space
- This scheme is called *memory-mapped I/O*

(Continued on next slide...)

# Controlling I/O Devices



- Two methods to transfer data from the controller's local buffer to the appropriate memory area of the computer are:
- **Non-DMA transfer**
  - As soon as transfer of data from input device to the controller's local buffer is complete, the controller sends an interrupt signal to CPU
  - CPU then stops what it is doing currently, and transfers control of execution to the starting address of the service routine, which handles the interrupt
  - Interrupt service routine transfers the data from local buffer of the device controller to main memory

*(Continued on next slide...)*

# Controlling I/O Devices



## ■ DMA transfer

- When the operating system prepares for data transfer operation, it writes the relevant commands and their associated parameters into the controller's registers
- After the controller has read the data from the device into its buffer, it copies the data one byte or word at a time from its buffer into main memory at the specified memory address
- It does not involve the CPU
- Device controller sends an interrupt to CPU only after it completes copying the entire data



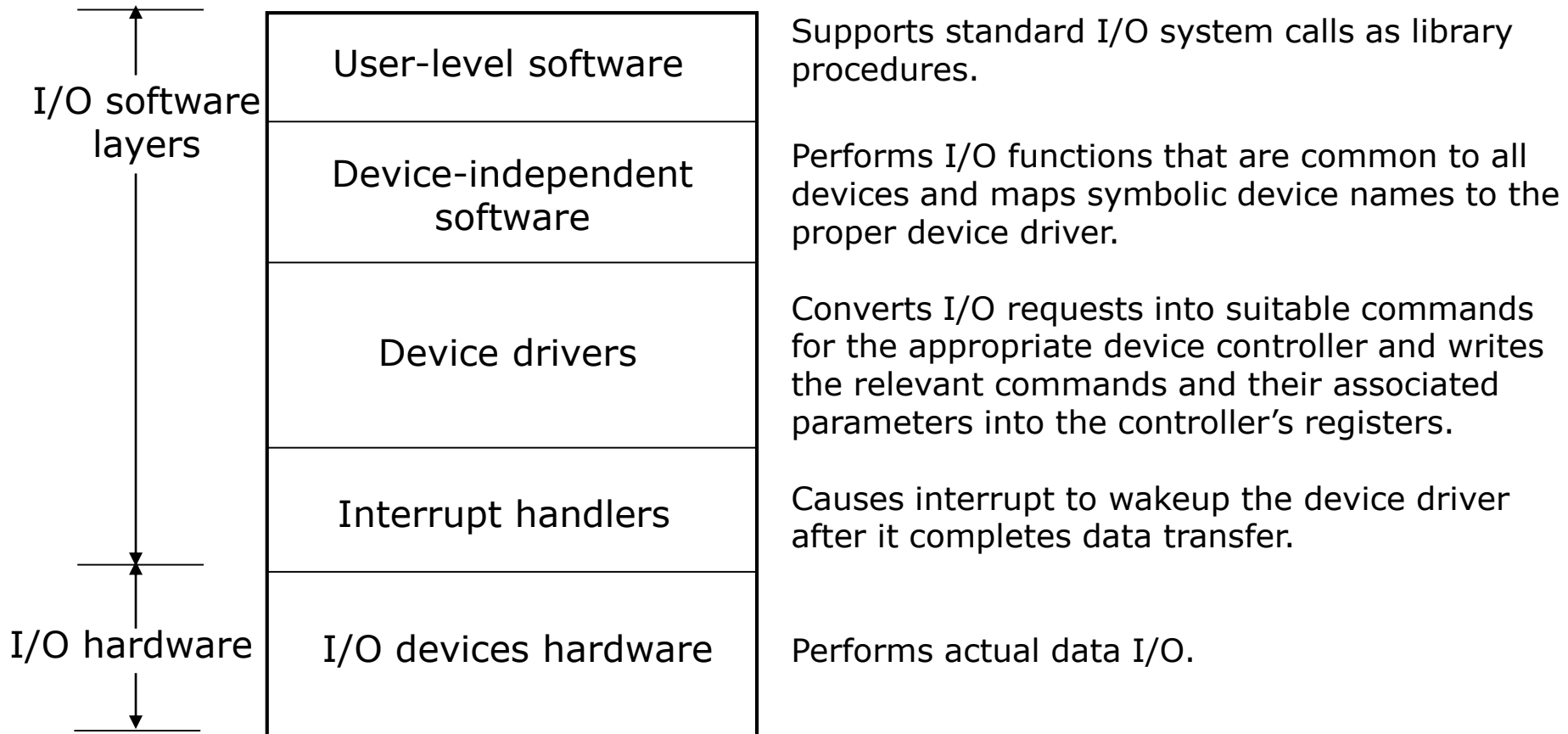
# Simple and Easy User Interface to I/O Devices

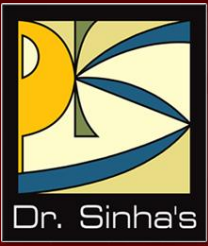


- Operating systems provide simple and easy user interface to all I/O devices
- They achieve this by organizing the software for using I/O devices as a series of layers
- Lower layers hide the internal details
- Upper layers present a nice, clean, uniform interface to the users

*(Continued on next slide...)*

# Layers of I/O System





# Security



# Security

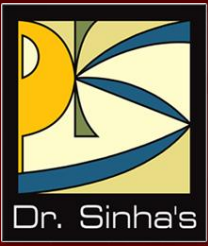


- Deals with protecting the various resources and information of a computer system against destruction and unauthorized access
- **External security:** Deals with securing computer against external factors such as fires, floods, earthquakes, stolen disks/tapes, etc. by maintaining adequate backup, using 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



# 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...)*

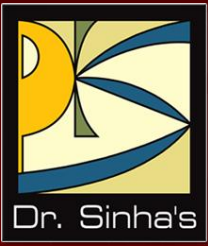
# Command Interpretation



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





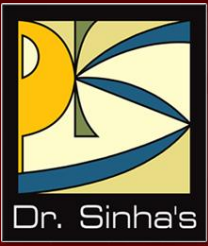
# OS Capability Enhancement Software



# 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



# Some Popular Operating Systems



# 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. Not in much use now after development 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 Server (Earlier Known as 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 multiprocessing 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
- Can run Microsoft Windows applications and many UNIX applications directly

# 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



# Mac OS



- Designed in mid 1980s by Apple Incorporation
- Main objective was 'ease of use'
- First OS to introduce the idea of GUI (Graphical User Interface), which was later adopted by almost all Operating Systems

# Key Features of Mac OS Include



- Intuitive user interface
- Interoperable with iOS and WatchOS
- Supported with a large set of built-in applications
- iCloud facility
- Privacy and security features help users work in a trusted environment
- Flexibility to work with Windows OS
- Use by differently abled users

# iOS



- Designed in 2007 by Apple Incorporation for its mobile devices such as iPad, iPhone and iPod.
- Provides a protective shell for each application (app) to prevent other apps from tampering them
- Two apps can communicate directly, if approved

# Key Features of iOS Include



- Highly intuitive user interface
- Facility to write, mark and draw
- Facility to ask for information or instruct using voice interface
- Interoperable with Mac OS and WatchOS
- Supported with a large set of applications and games
- iCloud facility
- Multitasking facility
- Camera, music and maps
- Privacy and security features help users work in a trusted environment

# WatchOS



- Designed in 2015 by Apple Incorporation for its smart wrist watch (Apple Watch)
- It is based on iOS and has many similarities with iOS in terms of features

# Key Features of WatchOS Include



- Intuitive user interface
- Health Kit
- Music
- Contact list, calendar, with notifications for important dates, appointment, etc., calculator and many other useful applications
- Ability to share data with other Apple devices

# Android OS



- An open source OS for mobile devices from Google
- Google developed it from Open Handset Alliance (OHA), a business alliance of companies to develop open standard for mobile devices
- It is Linux based, as it uses Linux kernel at its core for basic system functionalities
- It has millions of applications, making it the most versatile software environment for mobile devices

# Key Features Android OS Include



- Intuitive user interface
- Lets users choose their hardware device
- Portable across current and future hardware platforms
- Its architecture is flexible because it is component based
- It supports all Google services (Gmail, Google search, etc.)
- It supports 2D and 3D graphics, video streaming, and multilingual interface
- It supports multiple keyboards and makes them easy to install
- It supports multi-layer security



# Keywords/Phrases



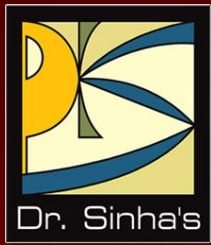
- Access control
- Android OS
- Batch processing
- Command interpretation
- Command-line interface (CLI)
- CPU-bound jobs
- Cryptography
- Demand paging
- Device management
- External security
- File
- File attributes
- File extensions
- File management
- Graphical User Interface (GUI)
- I/O-bound jobs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

*(Continued on next slide...)*

# Keywords/Phrases



- Security
- Sequential access files
- Swapping
- Throughput
- Tightly coupled system
- Time-sharing
- Time slice
- Time slot
- Translating programs
- Turnaround time
- Uniprogramming system
- Unix
- User authentication
- Utility programs
- Virtual machine
- Virtual memory
- WatchOS



# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 15

**Application  
Software Packages**

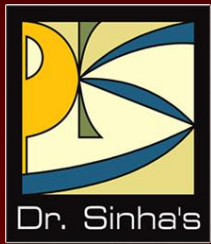


# Learning Objectives



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

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



# Word-Processing 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 (Examples)



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 (Examples)



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 (Examples)



*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 (Examples)



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.

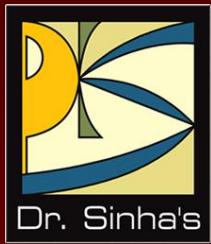
## **(c) Centered text**

# Mathematical Symbols (Examples)



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

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



# Spreadsheet Package



# Spreadsheet Package



- **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



# Few 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



Row numbers

Column letters

A label running across multiple columns

	A	B	C	D	E	F
1	FINAL EXAM MARKS SHEET(CLASS-X: 2020)					
2						
3	NAME	PHYS	CHEM	MATHS	TOTAL	PERCENT
4						
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						

A label

Cell F4

Alphabetic Value in a Cell

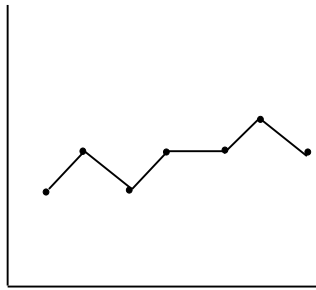
Numeric Value in a Cell

Cell C11

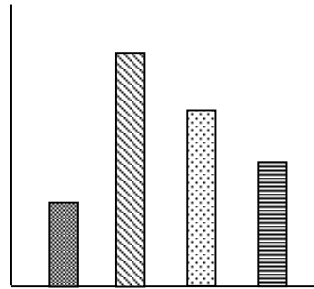
Result of the function @SUM(B9..D9)

Result of the formula + E9/3

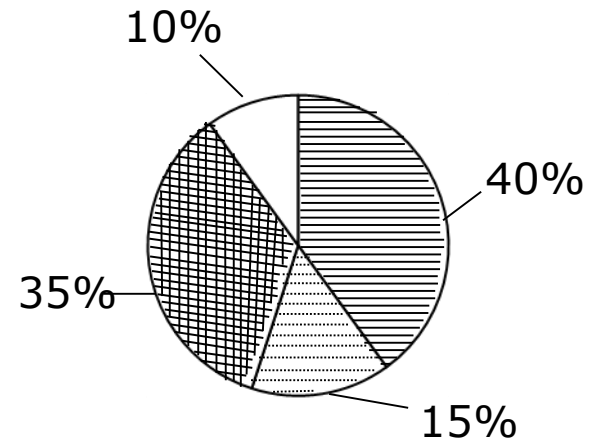
# Examples of a Line Graph, a Bar Chart and a Pie Chart



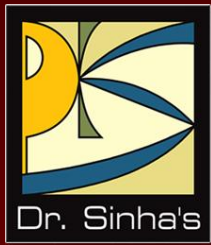
(a) A line graph



(b) A bar chart



(c) A pie chart



# Graphics Package



# Graphics Package



**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



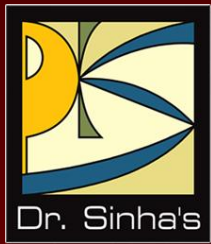
- Drawing designs
- Painting drawings and pictures
- Presenting graphs and charts
- Dragging-and-dropping graphic objects
- Importing graphic objects
- Capturing screen snapshots

# 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



**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

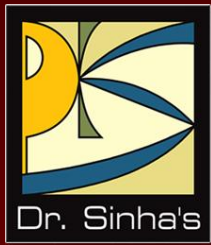


- Calendar
- To-do list
- Address book
- Investments book
- Inventory book

# 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



# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 16

**Business Data  
Processing**

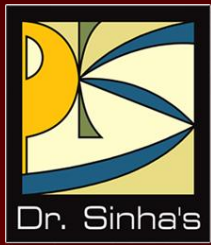


# Learning Objectives



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

- Difference between data and information
- Data storage hierarchy commonly used to facilitate data processing
- Standard methods of organizing data
- File Management System (FMS)
- Database Management System (DBMS)
- Basic concepts and main components of FMS and DBMS



# Basic Concepts



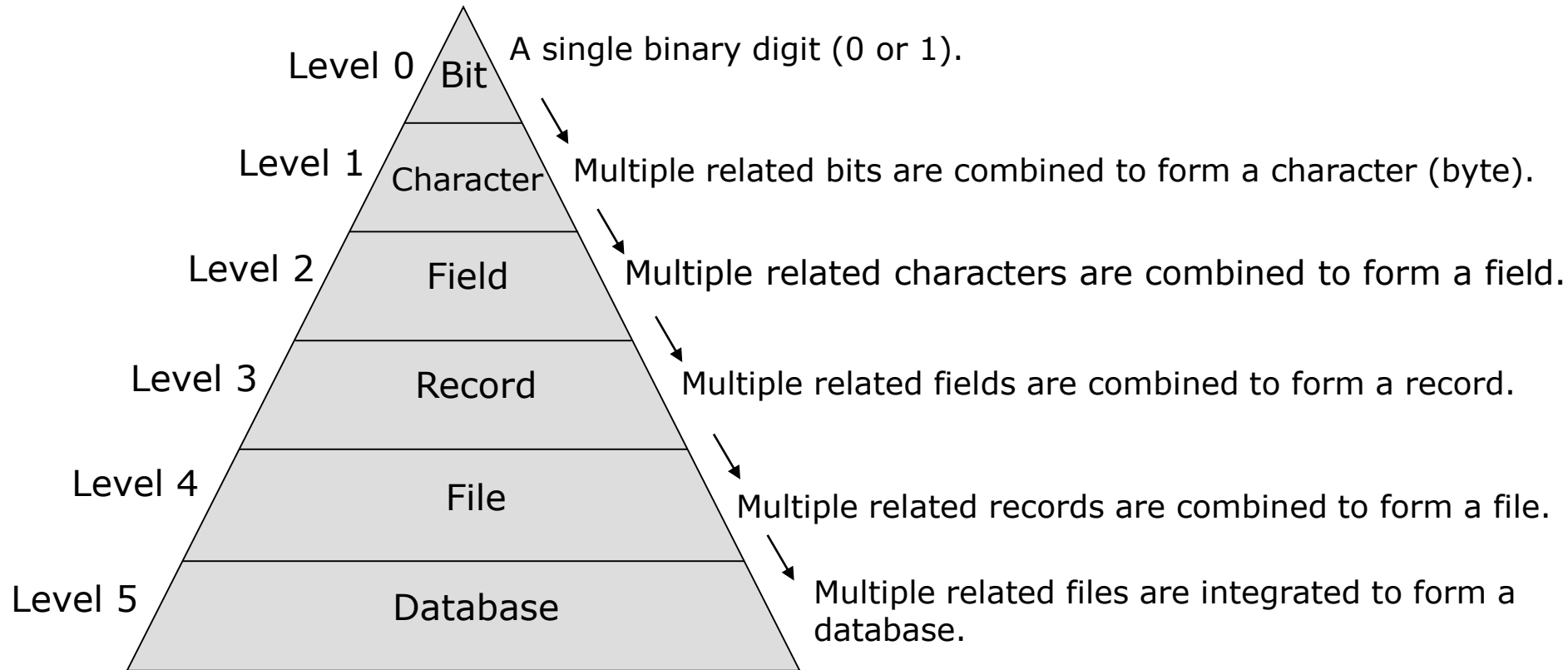
# Data Processing



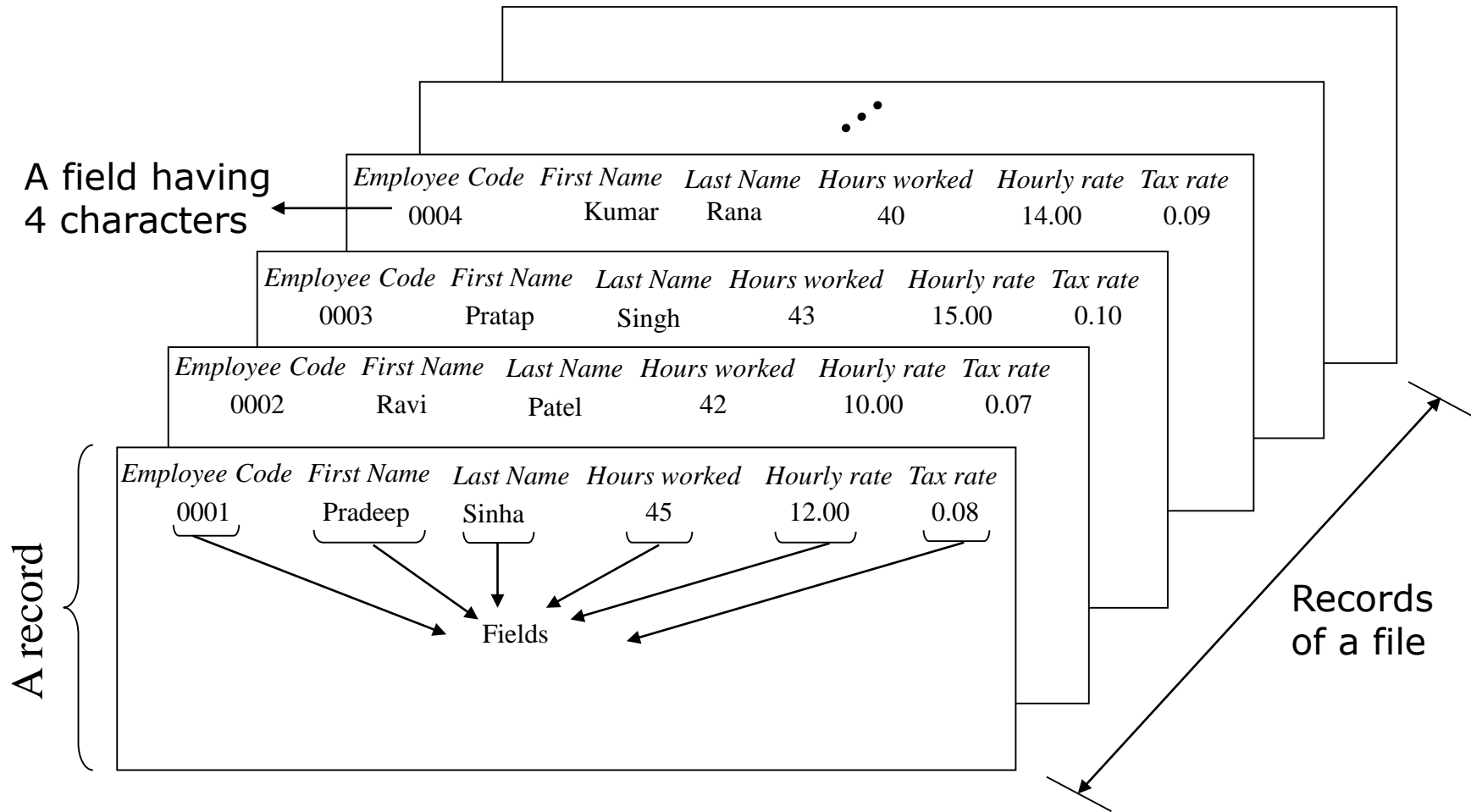
- *Data* is a collection of facts – unorganized but able to be organized into useful information
- *Information* is data arranged in an order and form that is useful to the people who receive it
- *Data processing* is a series of actions or operations that converts data into useful information
- A *data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

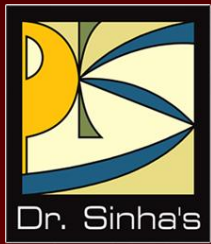


# Data Storage Hierarchy



# 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 the application program(s) processes the data stored in these files to generate desired output
- It is customary to use a master file of permanent data, and transaction files containing data of temporary nature

# Limitations of File-oriented Approach



- **Limited query flexibility**
  - When the key field is not relevant to desired information, it needs to search entire file
- **Data redundancy**
  - Repetition of same data items in more than one file is known as *data redundancy*
  - It leads to increase in cost of data entry and data storage
- **Data integrity problem**
  - *Data integrity* refers to consistency of data in all files
  - Any change in a data item must be carried out in every file containing that field

# Limitations of File-oriented Approach

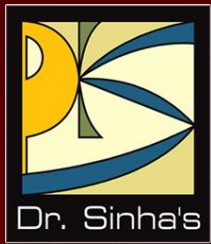


- **Lack of program/data independence**
  - An application program usually contains data format statements, which define the format of each data field precisely as the application needs it for processing
  - Data dependence occurs when data is dependent on application
- **Limited data security flexibility**
  - Offers file-level data security feature
  - It can enforce data access restrictions on an entire file only, not on a record or a field of data item

# Database-oriented Approach



- This approach integrates together data from multiple related files in the form of a database having following properties:
  - Provides greater query flexibility
  - Reduces data redundancy
  - Solves data integrity (inconsistency) problem
  - Makes data independent of application programs
  - Includes data security features at database level, record level, and even at field level



# File Management System





# What is File Management System?



- File-oriented approach of organizing data provides a set of programs to facilitate users to organize, create, delete, update, and manipulate 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 Organization



- Deals with physical organization of records of a file for convenience of their storage and retrieval
- Selection of a particular file organization depends on application type
- File organization requires use of some key field in every record in a file
- Key field value must be unique for each record of the file

# Sequential Files



- A sequential file stores its records one after another in ascending/descending order of their key field values
- A computer processes a sequential file in the order in which the file stores its records
- Sequential file organization is the most efficient and economical file organization for applications in which we have to update a large number of file records at regularly scheduled intervals
- *Activity ratio* is ratio of total number of records in transaction file and total number of records in master file

(Continued on next slide...)

# Sequential Files



## ■ Advantages

- Simple to understand and use
- Easy to organize and maintain
- Need relatively inexpensive I/O media and devices
- Efficient and economical to use for applications in which activity ratio is high

## ■ Disadvantages

- Inefficient and uneconomical to use for applications in which activity ratio is low
- Limited to batch-processing environment because of the need to accumulate transactions in batches
- Precludes possibility of up-to-the-minute data because of the need to accumulate transactions in batches
- Requires extra overhead of sorting the files before using them for processing
- Leads to data redundancy problem

# Direct Files



- Direct/random file organization is suitable for applications that directly locate any record by its key field value, without having to search through a sequence of other records
- A direct file stores each record at a location to which the address-generating function maps the record's key field value
- This mechanism is known as *hashing* and the address-generating function is called *hashing algorithm*
- Hashing algorithm sometimes maps the key values of two or more records to same storage address. This problem is known as *collision*
- To search a record, given its key value, the computer applies the hashing algorithm on the given key to generate its corresponding address
- If required, an application can process the records of a direct file sequentially in ascending/descending sequence of key field value

(Continued on next slide...)

# Direct Files



## ■ Advantages

- Can quickly locate and retrieve any record directly
- Does not require sorting of transactions
- Does not require accumulation of transactions in batches
- Can support interactive online applications
- Application can process direct file records sequentially

## ■ Disadvantages

- Require relatively expensive hardware and software resources
- Due to address generation overhead involved, they are less efficient and economical than sequential files for high activity ratio applications
- Often require special security and access synchronization mechanisms

# Indexed Sequential Files



- In indexed sequential file organization, there are two files for every data file – data file and index file
- Data file can store the records in random sequence
- Index file stores the index keys in sorted sequence on index key value
- This technique of file management is known as *Indexed Sequential Access Method (ISAM)* and files of this type are called *ISAM files*

(Continued on next slide...)



# Indexed Sequential File: Example



Employee code (key)	Address location
0001	1003
0002	1001
0003	1004
0004	1002
⋮	⋮

Index file

Address location	Employee record
1001	0002 R. S. Patel ...
1002	0004 R. K. Rana ...
1003	0001 K. P. Sinha ...
1004	0003 N. P. Singh ...
⋮	⋮

Data file

# Indexed Sequential Files



## ■ Advantages

- Applications in which activity ratio is high, can use index sequential files quite efficiently for sequential processing
- Applications in which activity ratio is low, can also use index sequential files quite efficiently for direct access processing

## ■ Disadvantages

- Require relatively expensive hardware and software resources
- Require more storage space than other types of files
- Are unsuitable for online applications requiring direct access to records

# File Utilities



- Are routines, which perform generic operations on data files
- **Sorting utility**
  - Arranges records of a file in some defined sequence
  - Keys determine the sorting sequence of the file's records
  - Enables users to specify their sequencing requirements for a file by means of input parameters
  - Reads un-sequenced records of an input file, and by means of various copying techniques, ultimately produces an output file containing records of the input file in desired sequence
- **Searching utility**
  - Finds a particular record in a file
  - Matches the specified key values with their values in each record to search the desired record
  - Efficiency of a search algorithm depends on file organization

*(Continued on next slide...)*

# File Utilities



- Searching a record from a direct or index sequential file requires much less time than searching a record from a sequential file
- **Merging utility**
  - Combines records of two or more ordered (sorted) files into a single ordered file
  - Requires records of each of the input files to be sorted in the same order, although their record layout need not be identical
  - Places records from each of the input files in their correct relative order, producing an output file having all records in the same order as input files
- **Copying utility**
  - Produces a copy of a file either from one unit of a storage device to another similar unit or from one storage medium to another

*(Continued on next slide...)*

# File Utilities



- Users often use file copying utility to take back-up copies of files
- Copying utilities are also known as *peripheral interchange programs* (PIP) since users often use them to copy a file from one peripheral device to another
- **Printing utility**
  - Printing utility prints a file on a printer to produce hard copy of its contents
  - Provides the facility to print file contents in different formats
  - Provides some selection and editing facilities to enable printing of parts of files
  - Provides special formats for printing files that contain program instructions rather than data

(Continued on next slide...)

# File Utilities



- **Maintenance utility**

- Copies data from one or more files to a new file selectively, or updates a file's contents selectively

# Sorting on One Key and Two Keys: Examples



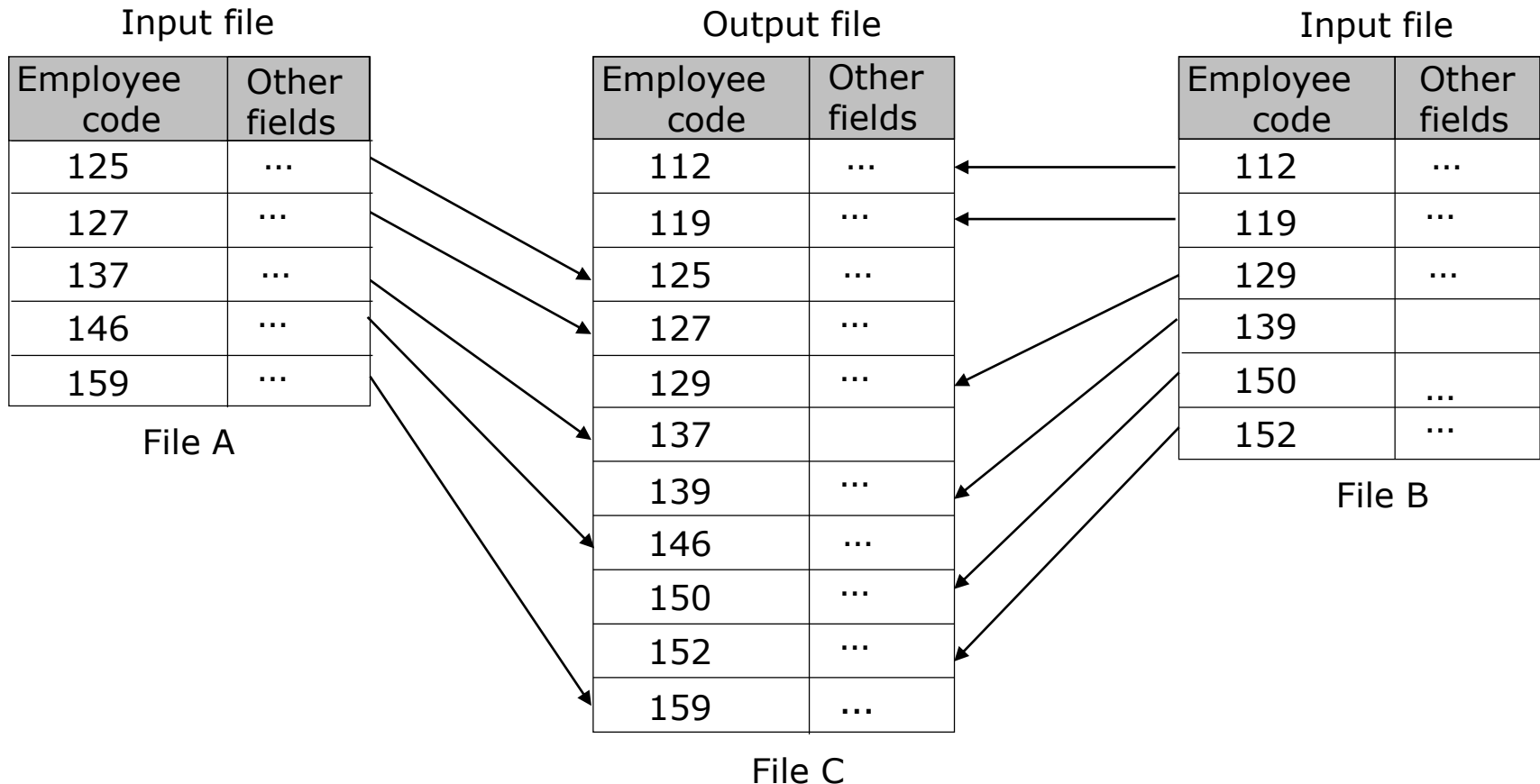
Employee code	Department code	Other fields (Name, Address, Qualification, Basic salary, etc.)
101	2	---
123	3	---
124	1	---
176	2	---
178	1	---
202	3	---
213	1	---

Sorting on one key in ascending employee-code sequence

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	---

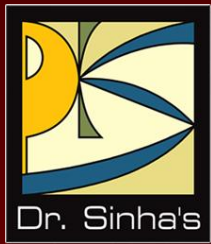
Sorting on two keys in ascending employee-code (secondary key) within ascending department-code (primary key)

# Merging Utility: Example



**Merging of files A and B to produce file C**





# Database Management System



# 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



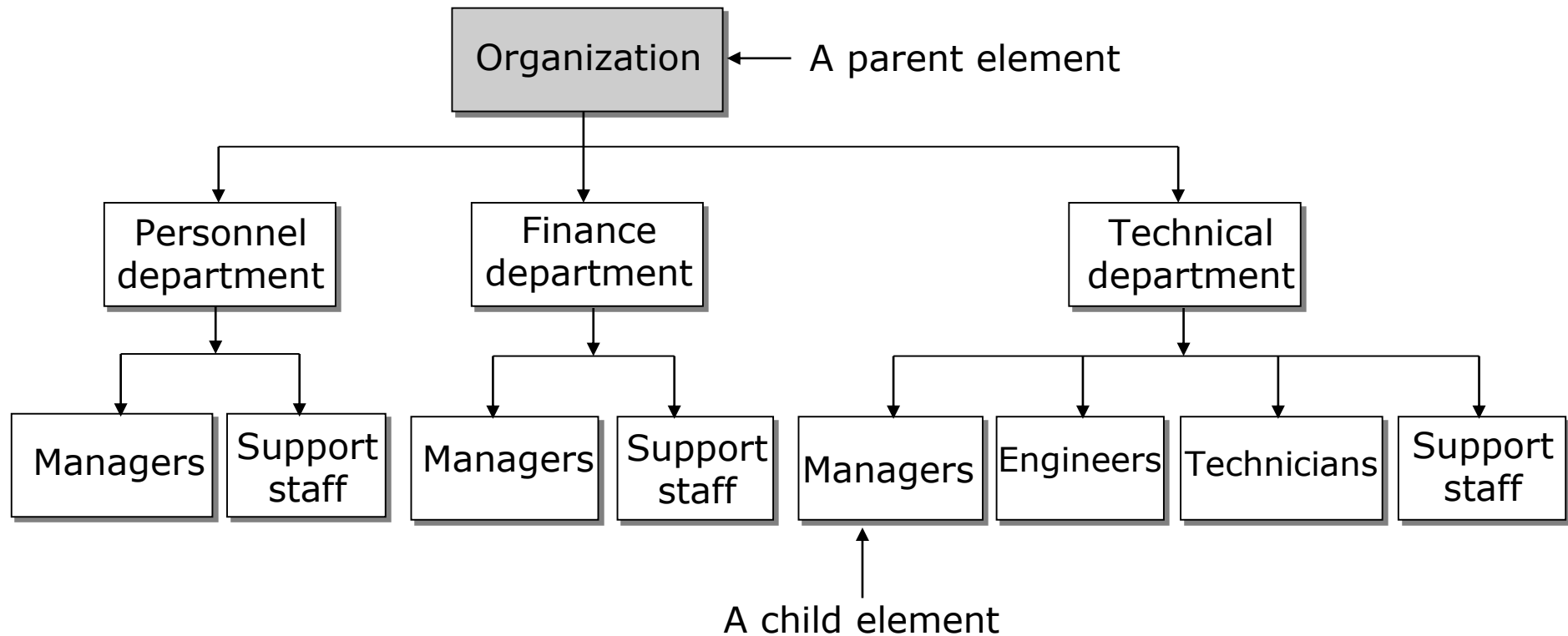
- *Database model* defines the manner in which the various files of a database are linked together.
- Four commonly used database models are:
  - Hierarchical
  - Network
  - Relational
  - Object-oriented

# Hierarchical Databases



- *Hierarchical database* links its data elements as an inverted tree structure
- Below the single-root data element are subordinate elements, each of which, in turn, has its own subordinate elements, and so on
- Tree can grow to multiple levels
- There may be many children elements under each parent element, but there can be only one parent element for any child element
- Main limitation of hierarchical database is that it does not support flexible data access
- Applications can access its data elements only by following paths formed by branches of the tree structure

# Hierarchical Database: Example

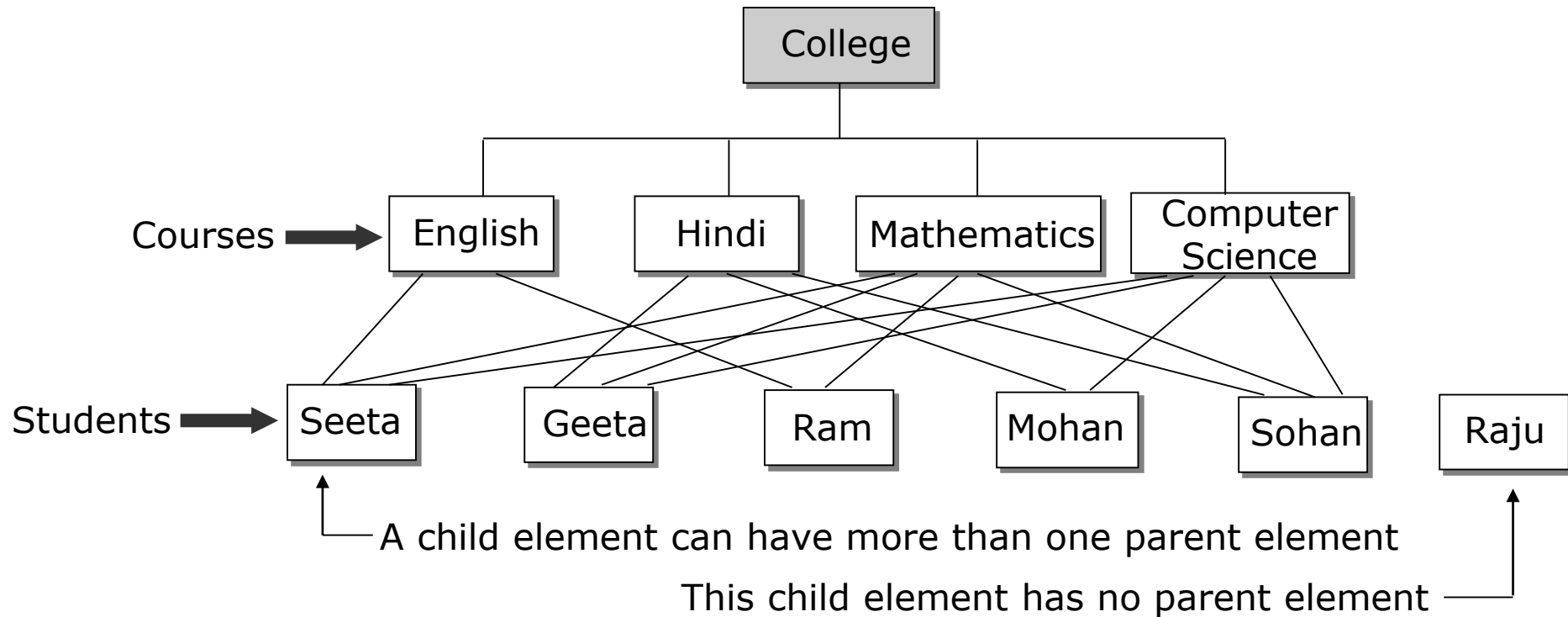


# Network Databases



- *Network database* is an extension of hierarchical database
- Organizes its data elements in such a manner that they have parent-child relationship among them
- Designer must determine all types of relationships among data elements while designing the database
- A child data element can have more than one parent element or no parent element at all
- Parent and child elements can have many-to-many relationships

# Network Database: Example



# Relational Databases



- *Relational database* organizes its data elements as multiple tables with rows and columns
- Each table column represents a data field, and each row represents a data record (also known as a *tuple*)
- Relational database model provides greater flexibility of data organization and future enhancements in database as compared to hierarchical and network database models
- Applications can organize their data elements in a relational database in a manner that is identical to real-life relationships between data elements

(Continued on next slide...)



# Relational Databases



- For adding new data to an existing relational database, there is no need to redesign the database afresh
- Users can also reorganize data elements, when necessary, to create new tables by selecting certain rows or specific columns from other tables, or by joining columns and rows from two separate tables

# Relational Database: Example



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	10-12-2020
11348	89303-530-0	08-11-2020
32228	13-201702-5	10-11-2020
16185	22-68111-7	05-12-2020
12859	71606-214-0	06-11-2020
62853	13-48049-8	15-11-2020
11348	18-23614-1	12-11-2020

(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 10-11-2020

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 10-11-2020 from the sample database of previous slide

# Object-oriented Databases



- Some key features that several applications require for effective modeling are:
  - Ability to model complex nested entities
  - Support for general data types found in object-oriented programming languages
  - Support for frequently useful object-oriented concepts such as object, class, inheritance, etc.
  - Support for proper match between object-oriented programming languages and database languages
- *Object-oriented database* is a collection of objects whose behavior, state, and relationships are in accordance with object-oriented concepts

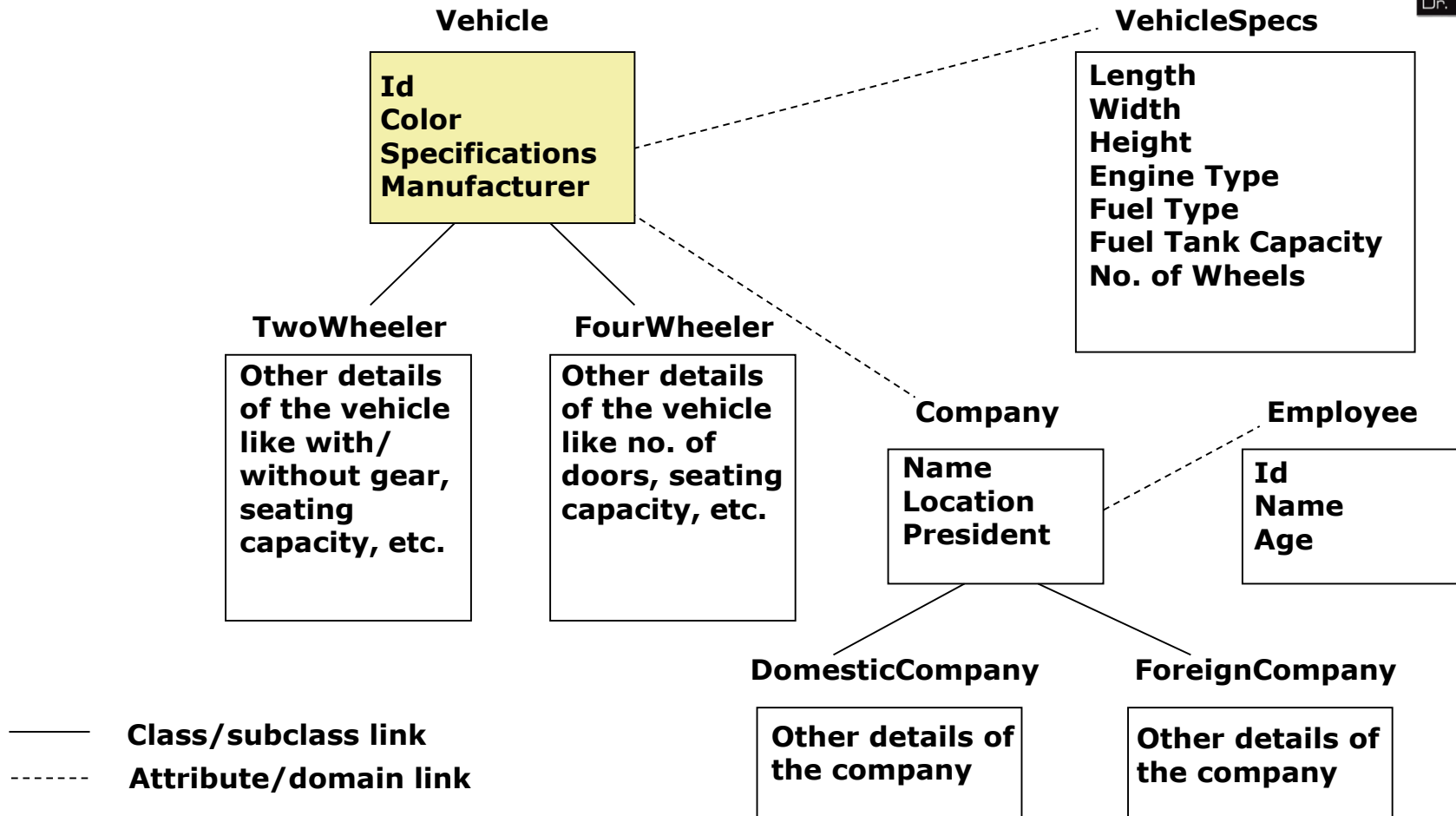
(Continued on next slide...)

# Object-oriented Databases



- *Object-oriented database management system* allows definition and manipulation of an object-oriented database
- It provides direct support for the definition and manipulation of the relationships among objects

# 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



- *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 a 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 17, 2020

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 in alphabetical order on last name of employee

# 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

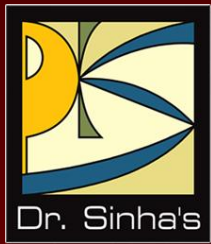
(Continued on next slide)

# Key Words/Phrases



- Query By Example
- Query language
- Record
- Relational database
- Report file
- Report Generator
- Schema
- Searching
- Secondary key
- Sequential file
- Sorting
- Transaction file
- Tuple





# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 17

**Data Communications  
& Computer Networks**

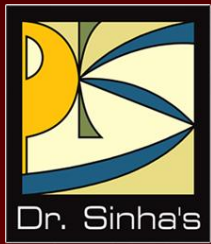


# Learning Objectives



## 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
- Wireless communications technologies and wireless networks
- Characteristics and advantages of distributed computing systems



# Basic Concepts and Terminologies



# Data Communications and Computer Networks

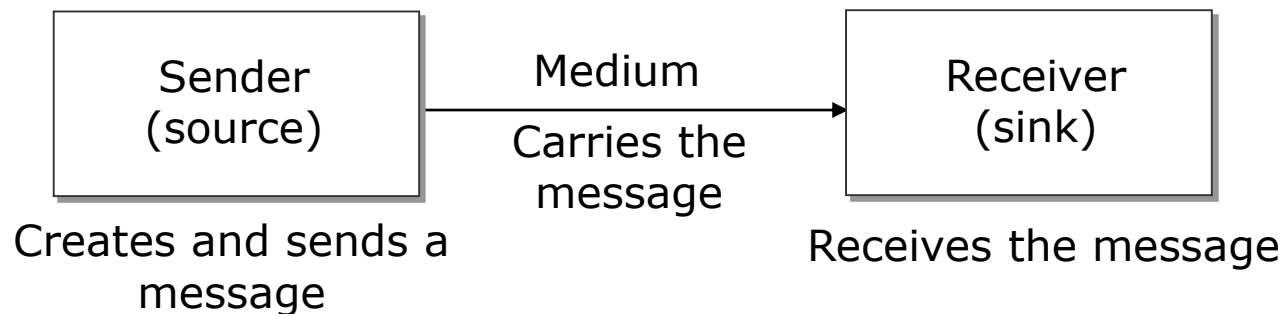


- A *computer network* is a network of computers
- It connects multiple computers in a manner to enable meaningful transmission and exchange of data among them

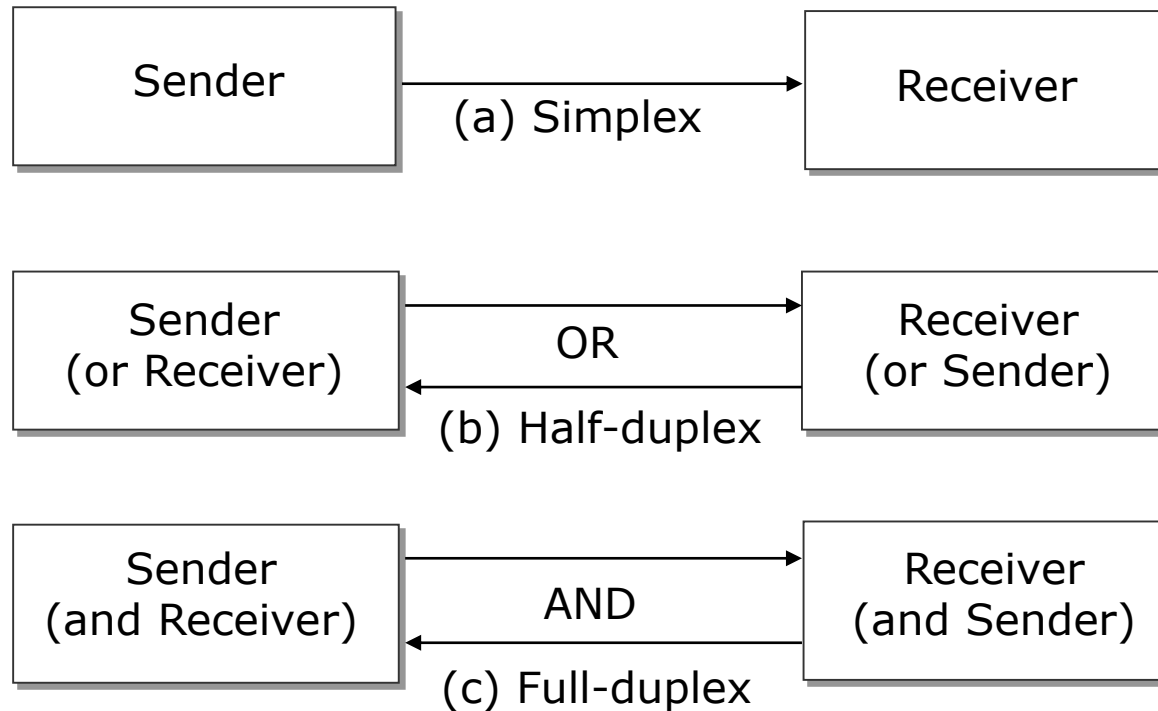
# Basic Elements of a Communication System



- *Communication* is the process of transferring a message from one point to another
- Electronic systems that transfer data from one point to another are called *data communication systems*



# Data Transmission Modes



# Data Transmission Speed



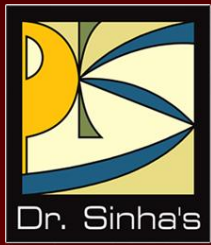
- **Bandwidth:** Range of frequencies available for data transmission. It refers to data transmission rate. Higher the bandwidth, the more data it can transmit
- **Baud:** Unit of measurement of data transfer rate. Measured in bits per second (bps)

# 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



# Data Transmission Media



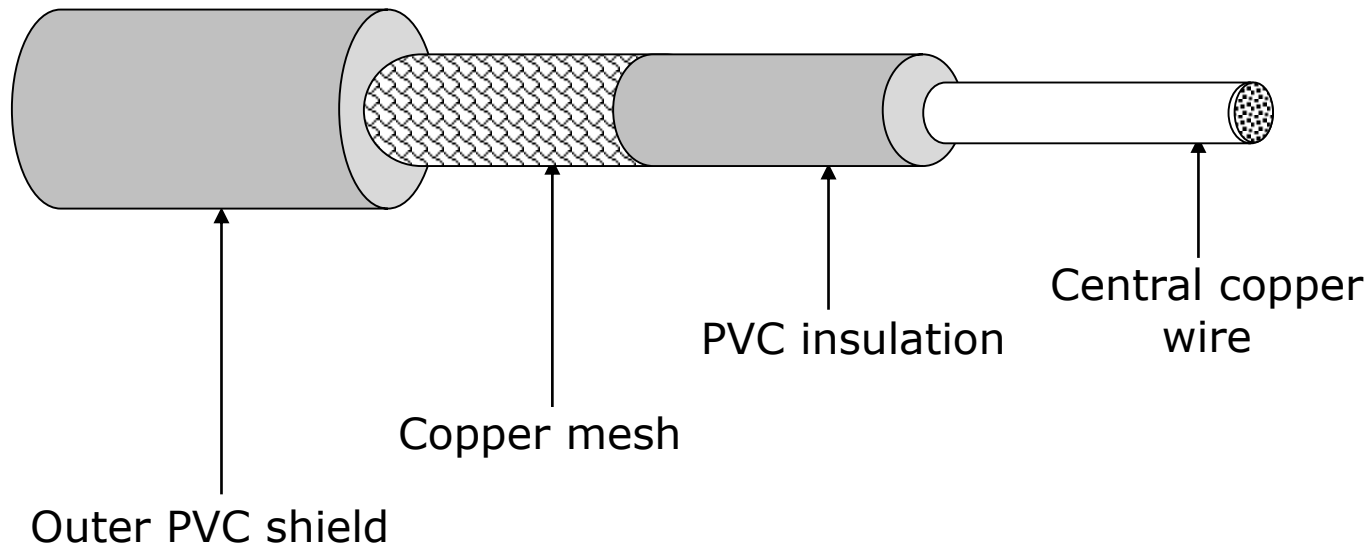
The most commonly used ones are:

- Twisted-pair wire (UTP cable)
- Coaxial cable
- Microwave system
- Communications satellite
- Optical fibers

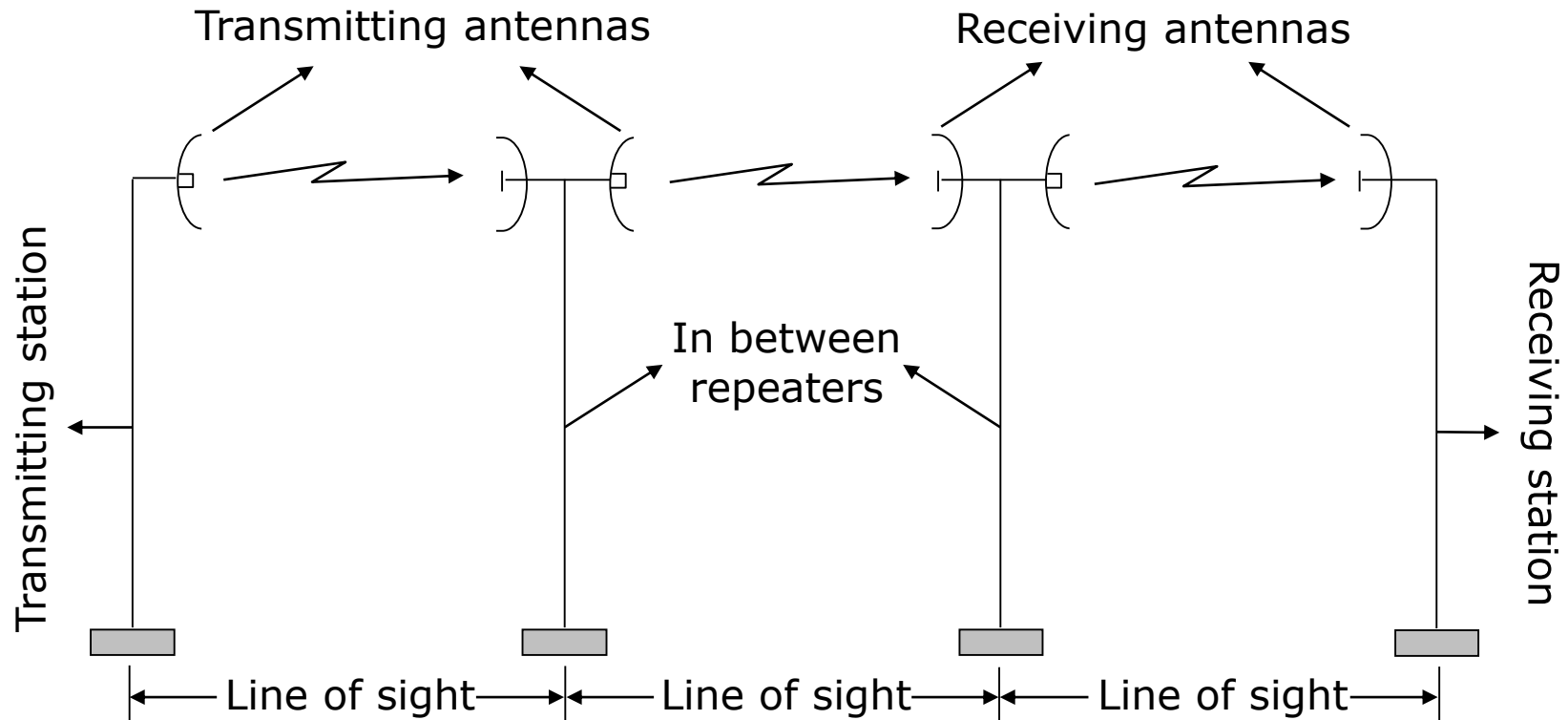
# Unshielded Twisted-Pair (UTP) Cable



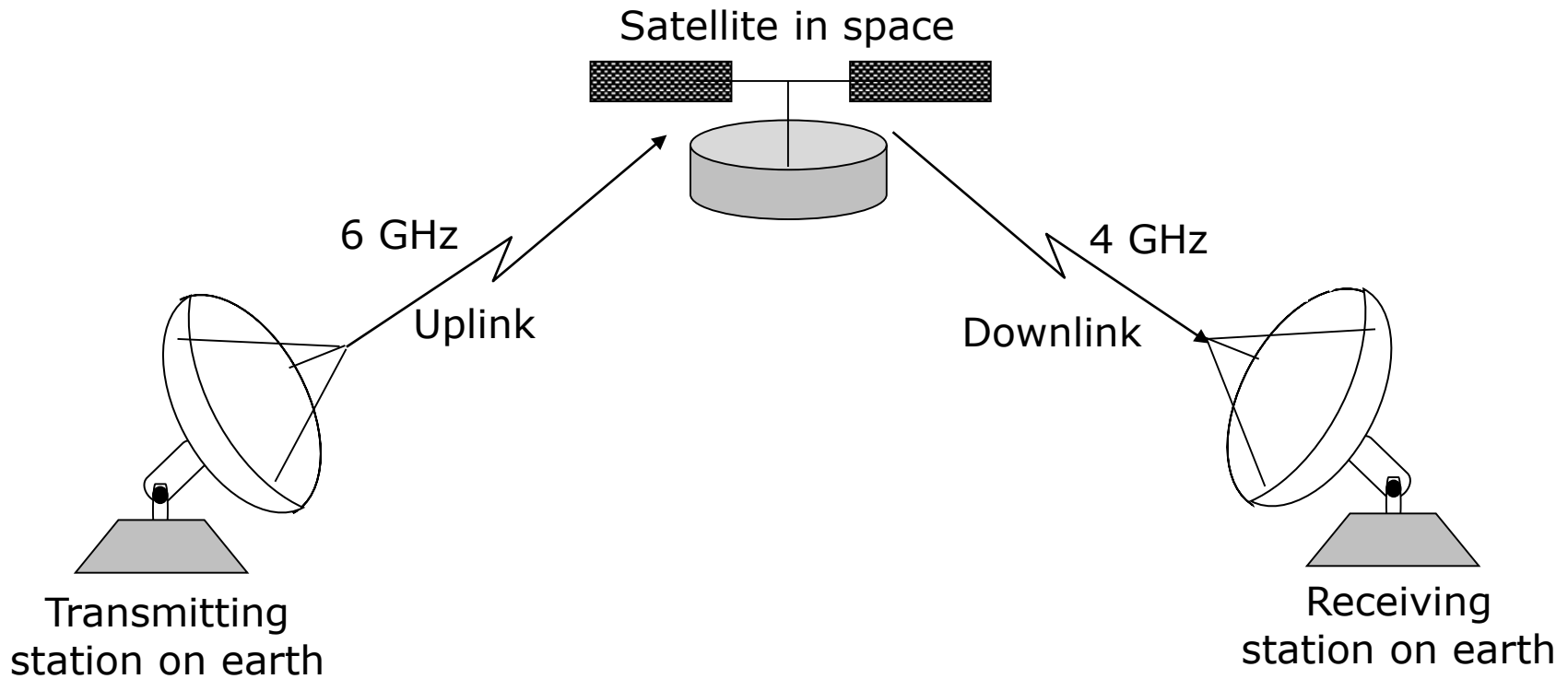
# Coaxial Cable



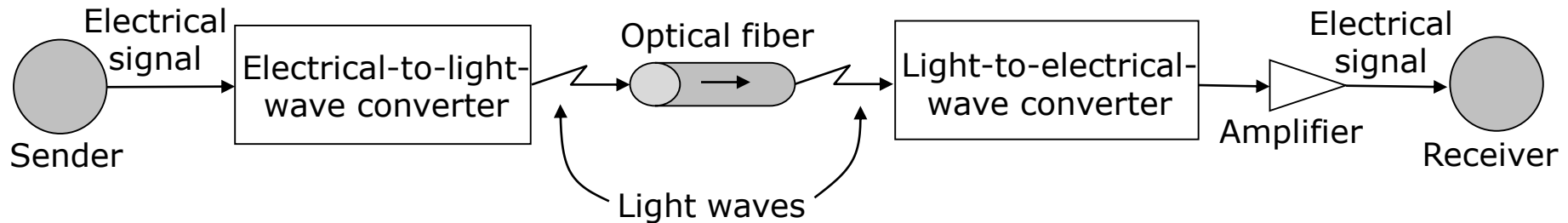
# Microwave Communication System

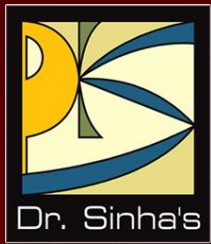


# Satellite Communication System



# Optical Fiber Communication System





# Digital and Analog Data Transmission





# 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

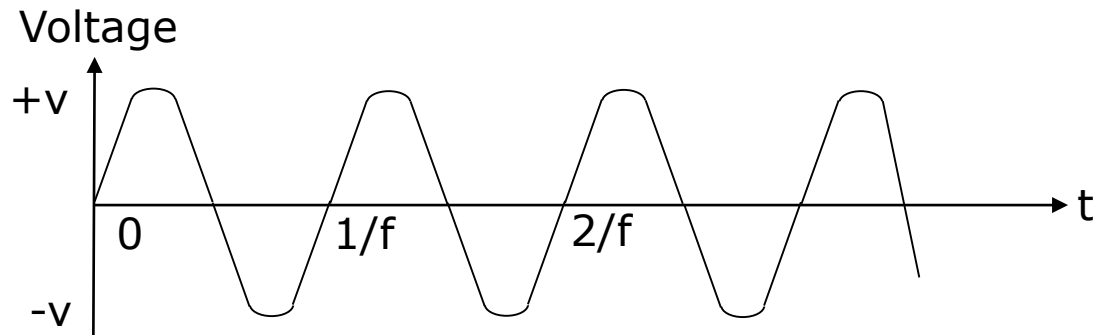
(Continued on next slide)

# Digital and Analog Data Transmission

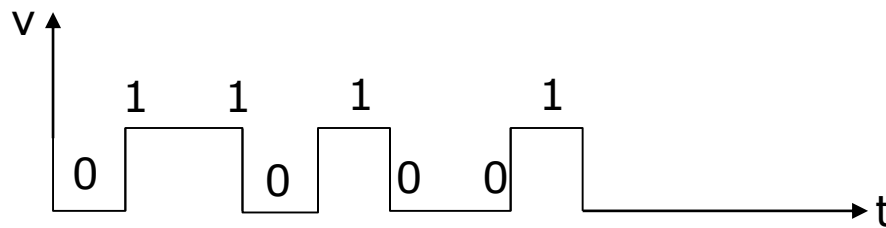


- When digital data is to be sent over an analog facility, digital signals must be converted to analog form
- Conversion of digital signal to analog form is known as *modulation*
- Conversion of analog signal to digital form is known as *demodulation*
- Digital transmission of data is preferred over analog transmission of data due to lower cost, higher transmission speeds, and lower error rate

# Analog and Digital Signals



(a) Analog signal



(b) Digital signal

# Modulation Techniques



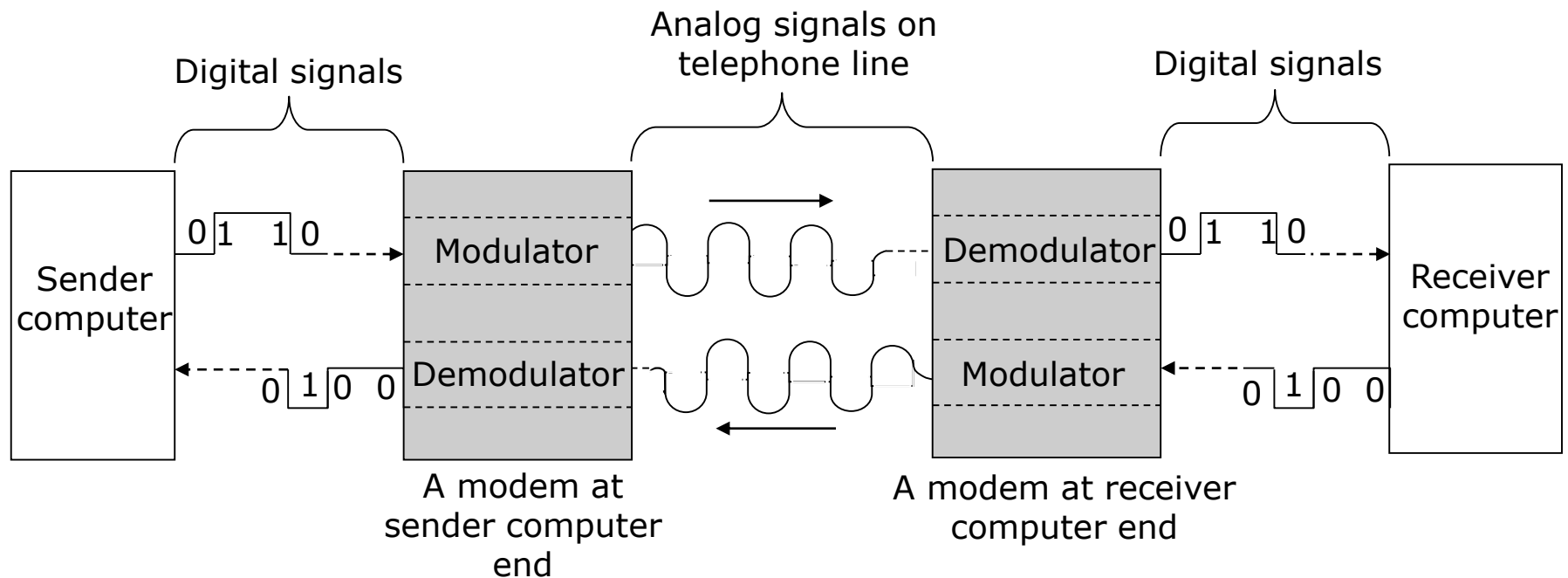
- **Amplitude Modulation (AM):** Two binary values (0 and 1) of digital data are represented by two different amplitudes of the carrier signal, keeping frequency and phase constant
- **Frequency Modulation (FM):** Two binary values of digital data are represented by two different frequencies, while amplitude 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 **MO**dulator/**DE**Modulator
- 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)

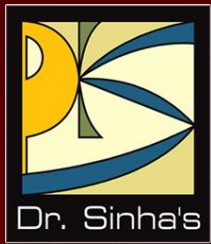
# Use of Modems in Data Communications



# Factors for Modem Selection



- Transmission speed
- Internal versus external
- Facsimile facility
- Error correction
- Data compression



# Data Transmission Services





# Data Transmission Services



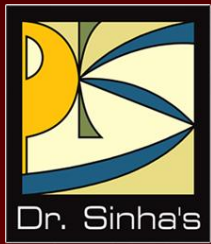
- Data transmission service providers are popularly known as *common carriers*
- Various types of services offered by common carriers are:
  - **Dial-up line:** Operates in a manner similar to a telephone line
  - **Leased line:** Special conditioned telephone line that directly and permanently connects 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



- **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 Techniques

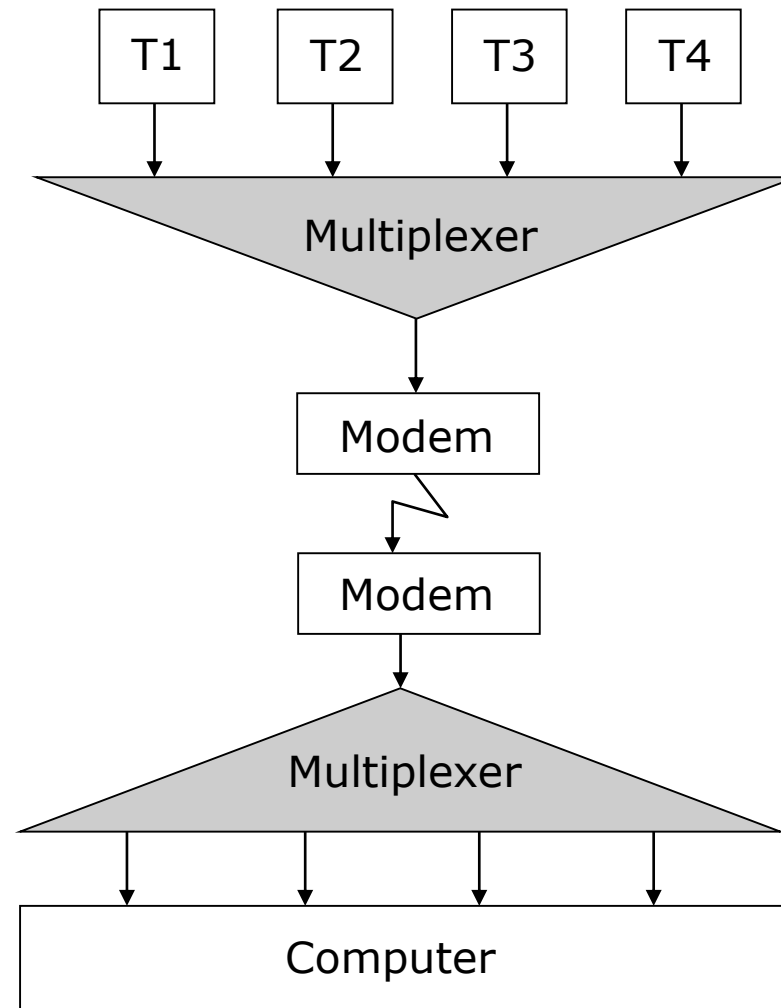


# Multiplexing



- 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

# A Multiplexed System

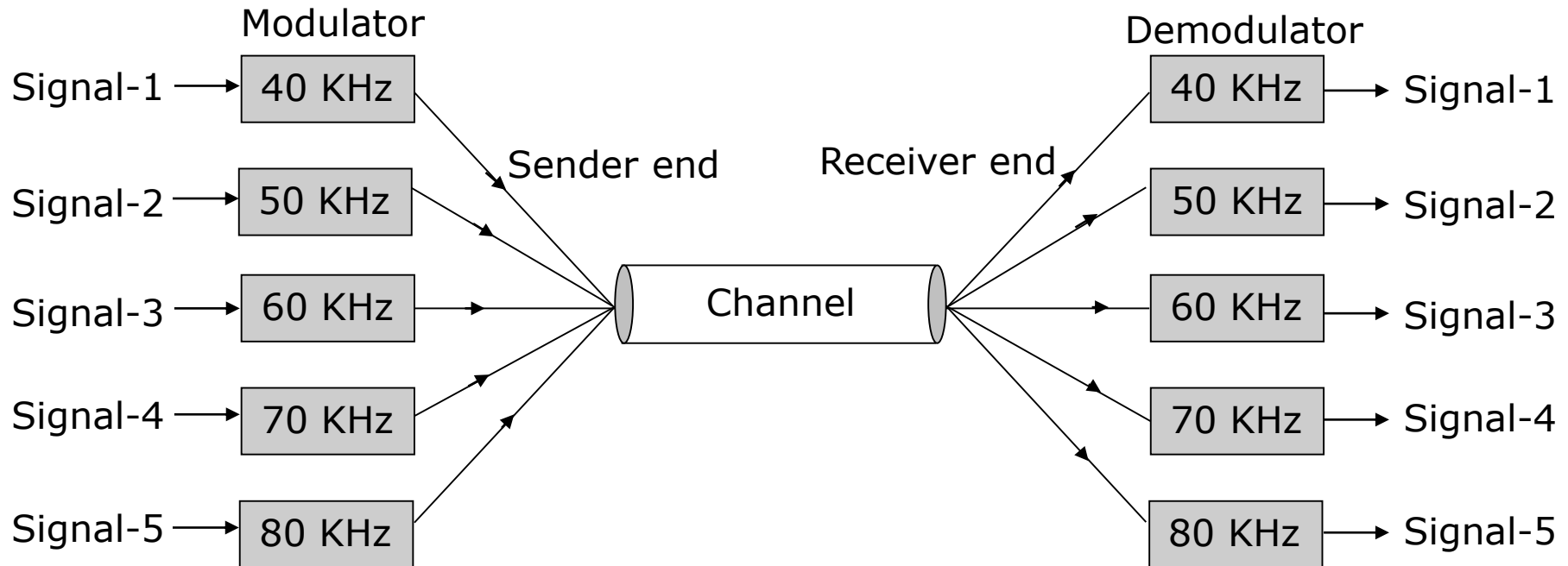


# Two Basic Methods of Multiplexing

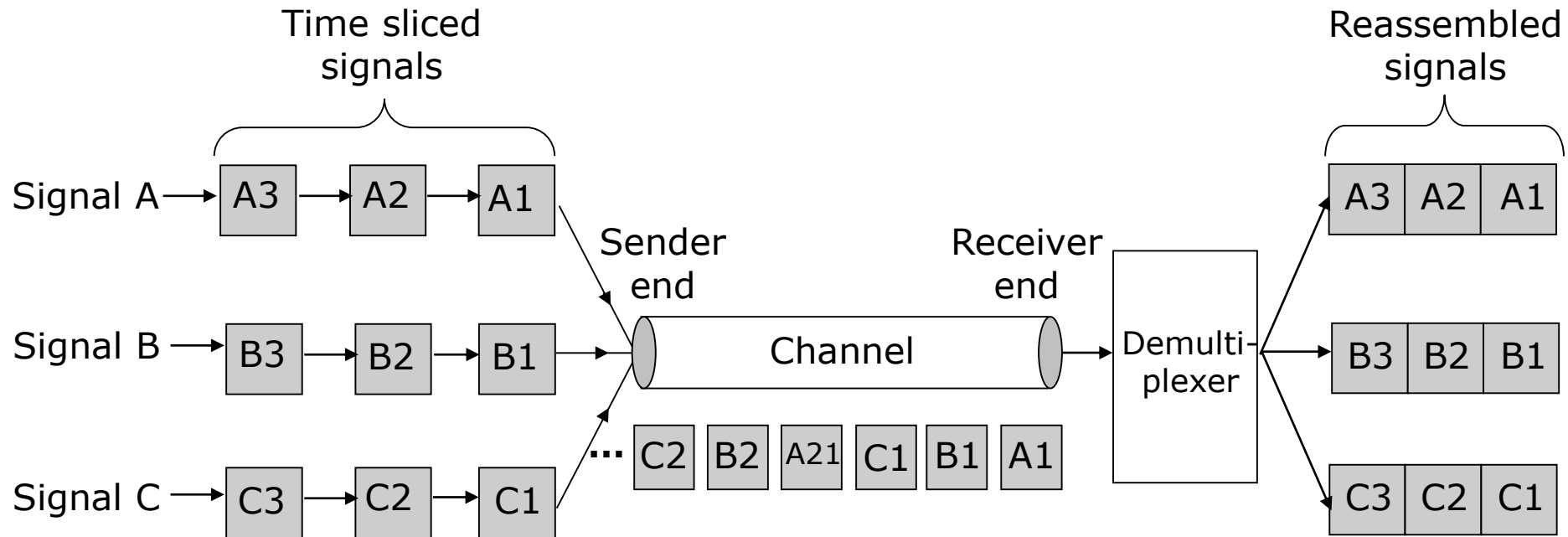


- **Frequency-Division Multiplexing (FDM):** Available bandwidth of a physical medium is divided into several smaller, disjoint logical bandwidths. Each component bandwidth is used as a separate communication line
- **Time-Division Multiplexing (TDM):** Total time available in a channel is divided among several users, and each user of the channel is allotted a time slice during which he/she may transmit a message

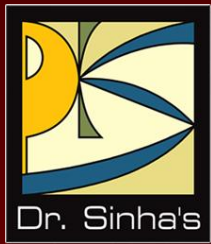
# Frequency-Division Multiplexing



# Time-Division Multiplexing







# Asynchronous and Synchronous Transmission



# 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)*

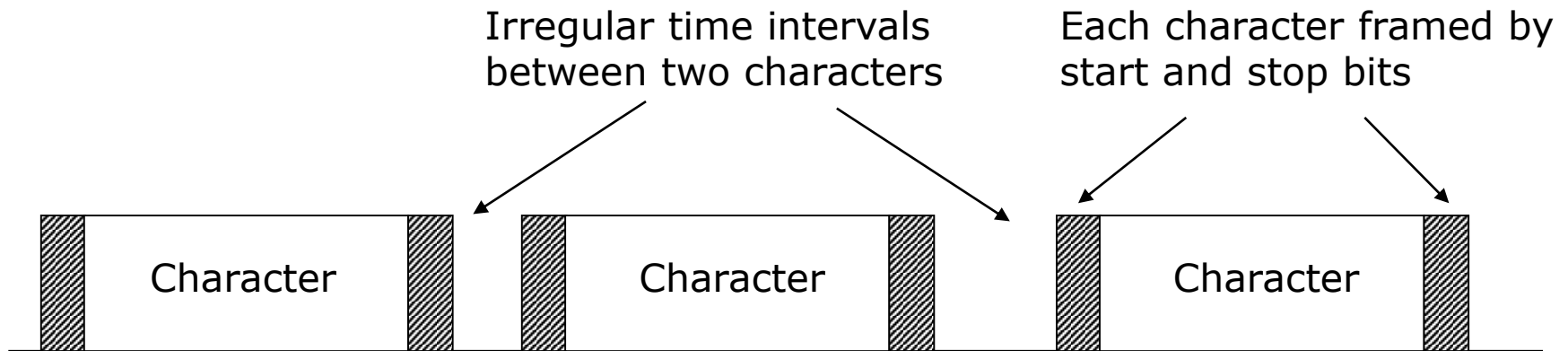
# Asynchronous and Synchronous Transmission



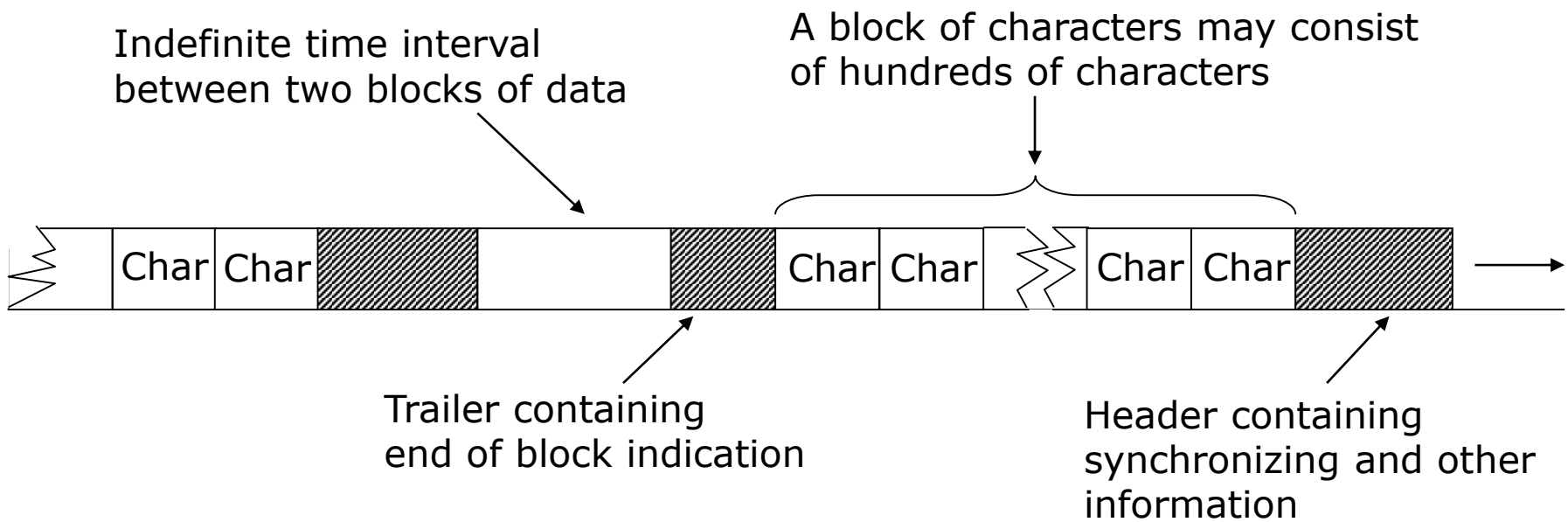
## ■ Synchronous transmission

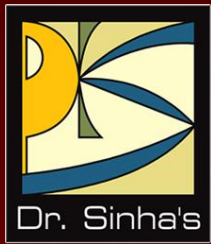
- Sender and receiver must synchronize with each other to get ready for data transmission before it takes place
- Entire blocks of characters are framed and transmitted together
- Well suited to remote communication between a computer and such devices as buffered terminals and printers

# Asynchronous Transmission



# Synchronous Transmission





# Switching & Routing Techniques



# 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)*

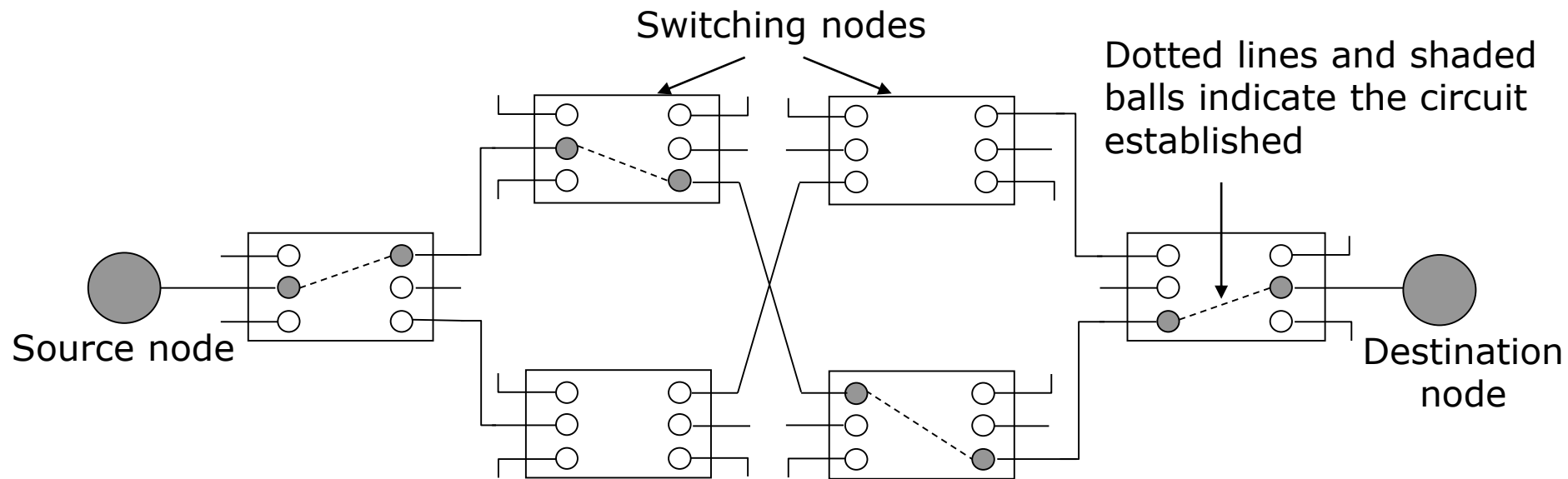
# Switching Techniques



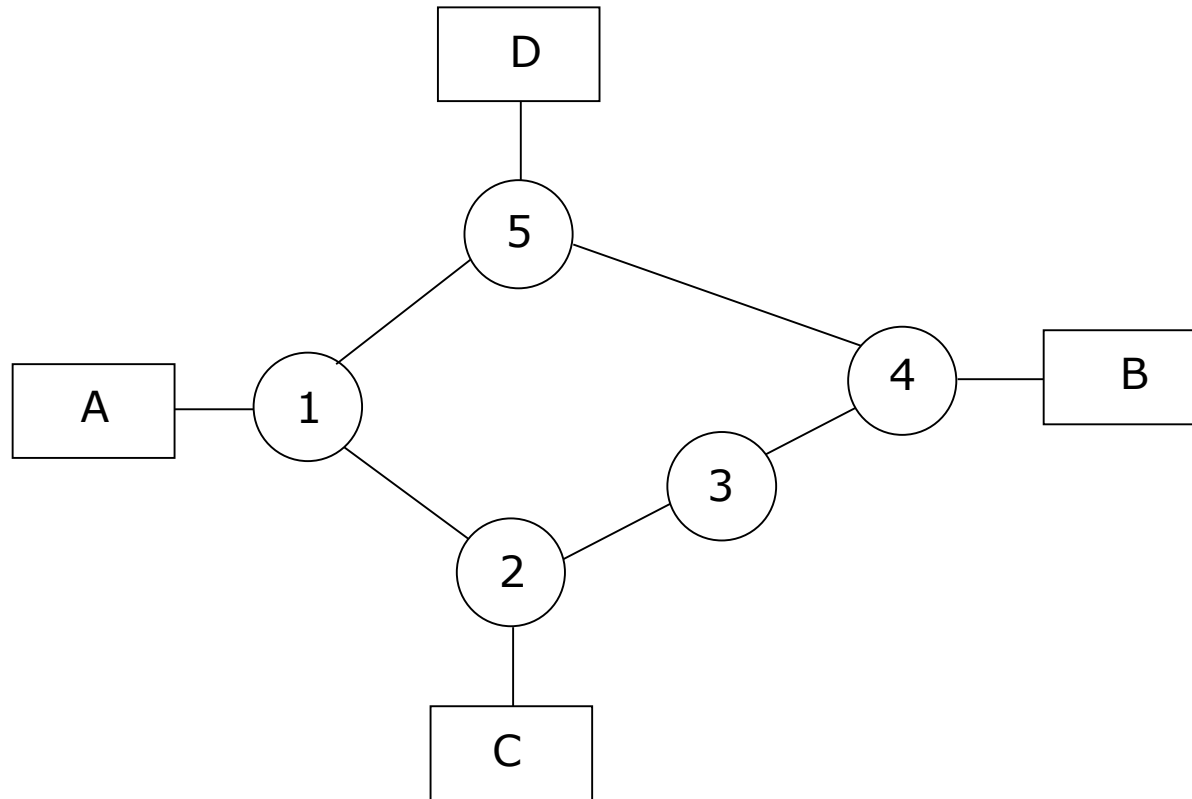
- **Message switching:** Sender appends receiver's destination address to the message and it is transmitted from source to destination either by store-and-forward method or broadcast method
- **Packet switching:** Message is split up into fixed size packets and each packet is transmitted independently 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

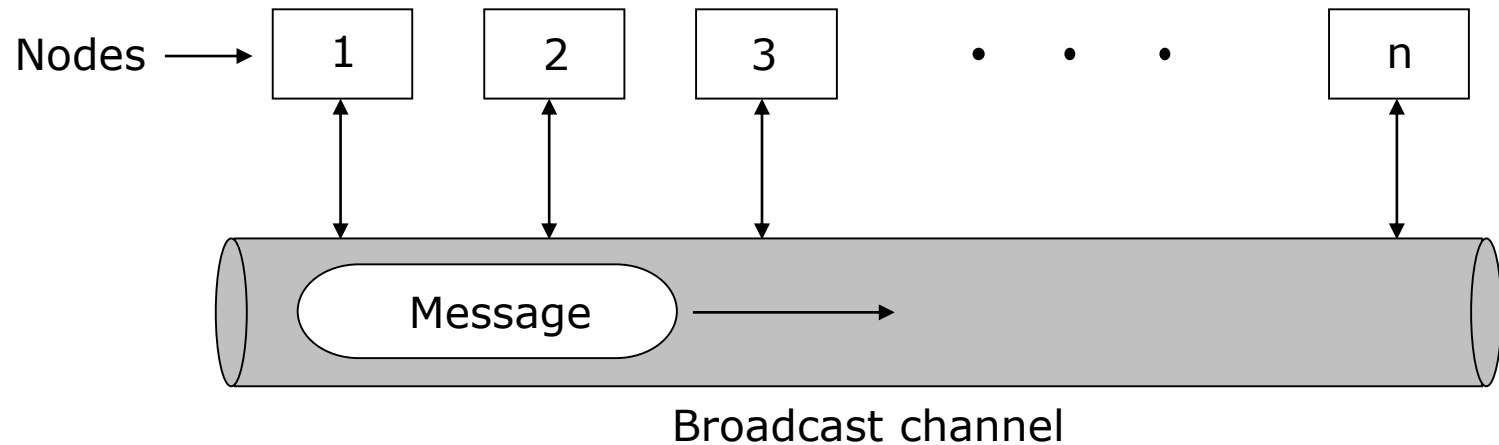


# Store-and-Forward Method of Message Switching



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



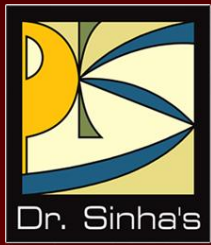
- In a WAN, when multiple paths exist between the source and destination nodes of a packet, any one of the paths may be used to transfer the packet
- Selection of path to be used for transmitting a packet is determined by the routing technique used

*(Continued on next slide)*

# Routing Techniques



- Three 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
  - **Hybrid routing:** Source node specifies only a few major intermediate nodes of the complete path, and hop-by-hop routing method is used to decide the subpaths between any two of the specified nodes.



# Network Topologies and Types



# Network Topology



- A network's topology refers to the way in which the network links its nodes
- It determines the various data paths available between any pair of nodes in the network
- Choice of a topology depends on a combination of factors such as:
  - Desired performance
  - Desired reliability
  - Size of the system
  - Expandability
  - Cost of components and services
  - Availability of communication lines
  - Acceptable delays in routing

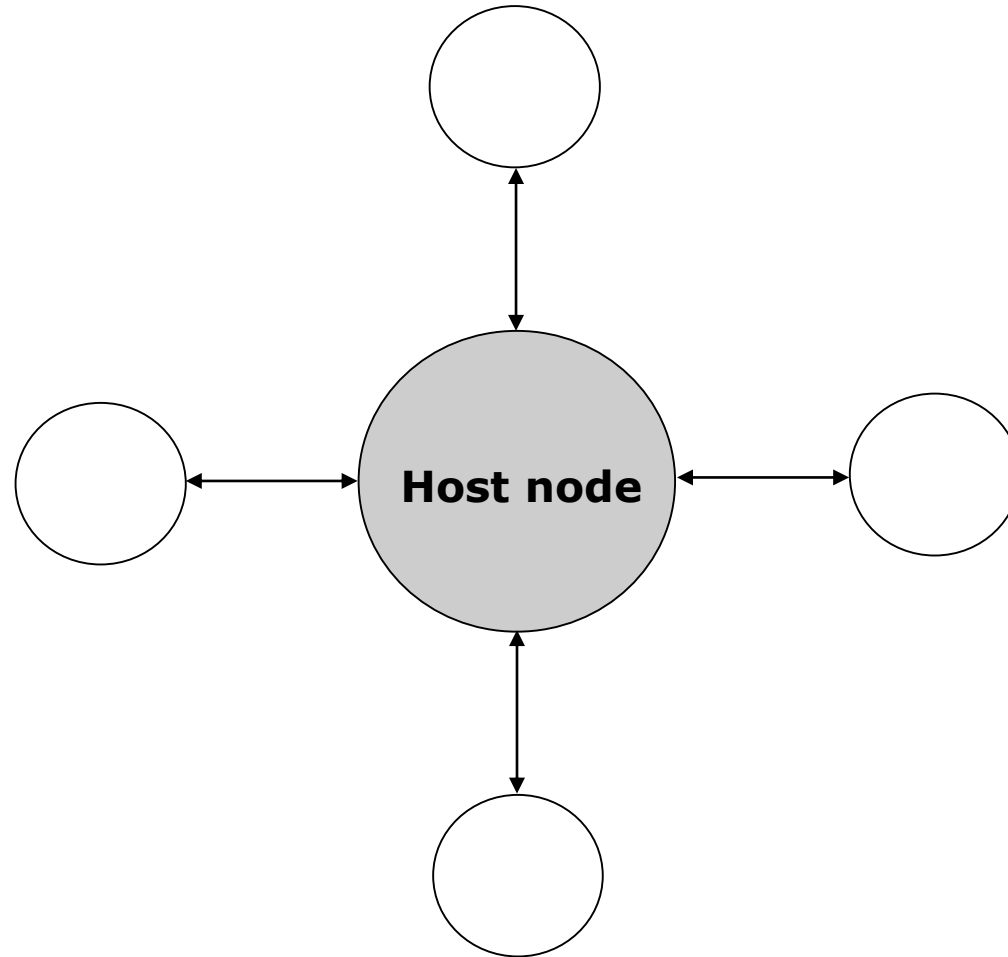
# Network Topologies



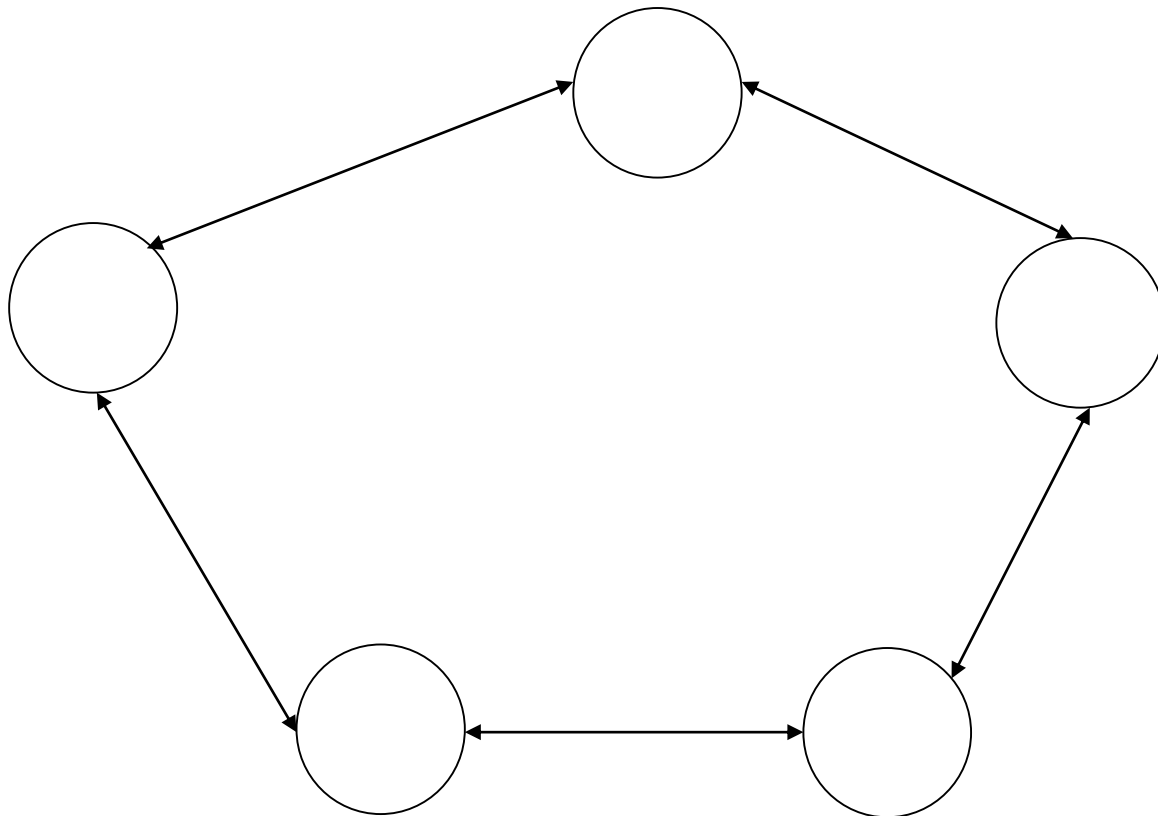
- Although number network topologies are possible, four major ones are:
  - Star network
  - Ring network
  - Completely connected network
  - Multi-access bus network



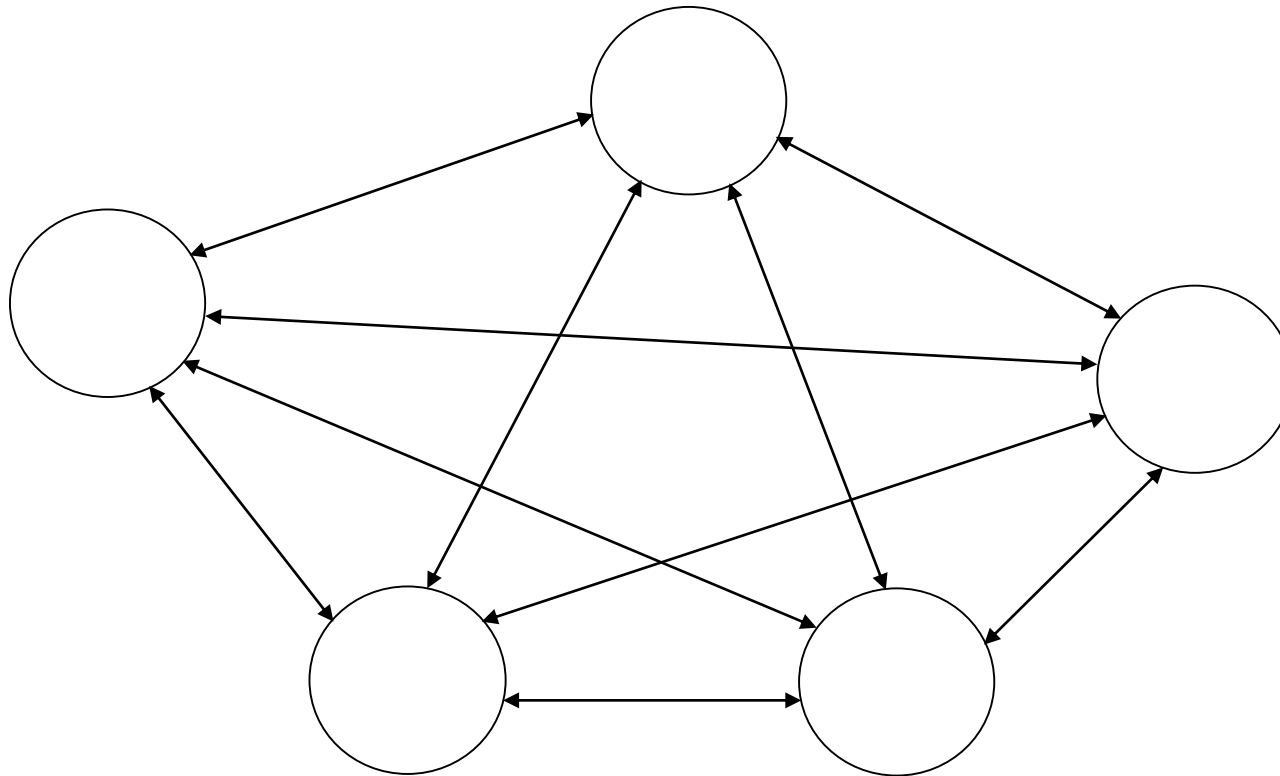
# Star Network



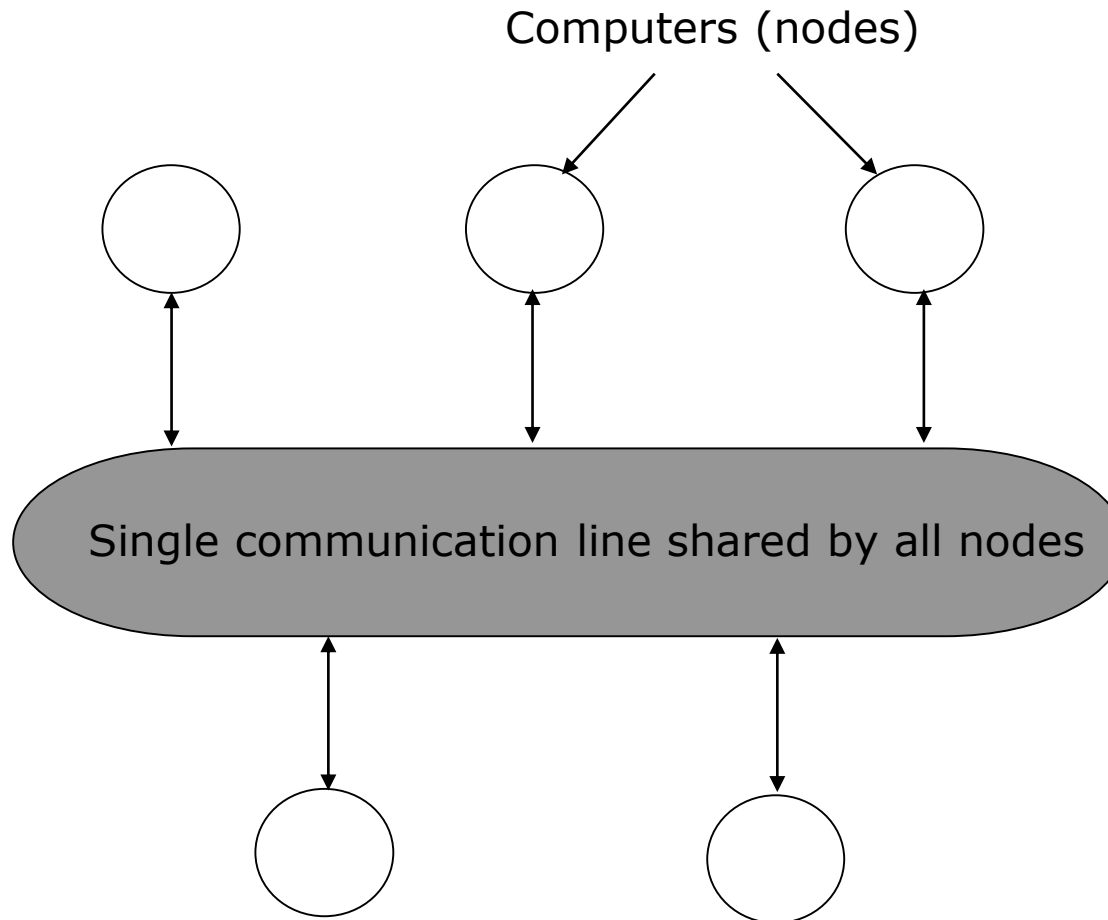
# Ring Network



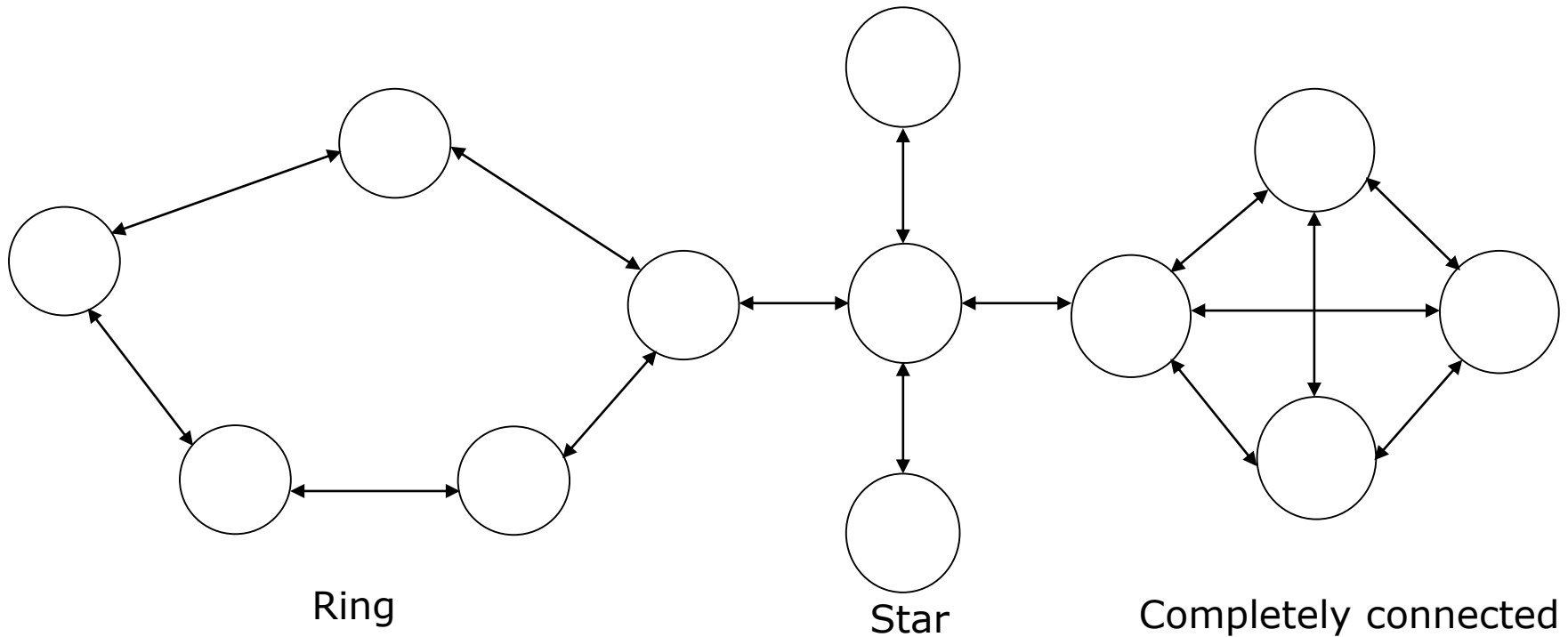
# Completely Connected Network



# Multi-Access Bus Network



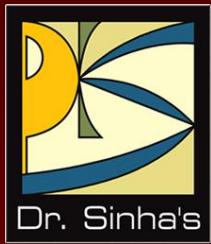
# Hybrid Network



# Network Types



- Five types of networks in common use are:
  - Personal-area networks (PANs)
  - Local-area networks (LANs)
  - Campus-area networks (CANs)
  - Metropolitan-area networks (MANs)
  - Wide-area networks (WANs)



# Communication Protocols



# Communication Protocols



- A *protocol* is a set of formal operating rules, procedures, or conventions that govern a given process
- *Communication* or *network protocol*, therefore, describes rules that govern transmission of data over communication networks



# Roles of a Communication Protocol



- **Data sequencing**

- Breaking a long message into smaller packets of fixed size
- Rules define method of numbering packets to detect loss or duplication of packets, and to identify packets

- **Data routing**

- Decide the path between source and destination

- **Data formatting**

- Define which group of bits or characters within a packet constitutes data, control, addressing, or other information

- **Flow control**

- Prevent a fast sender from flooding a slow receiver with data by regulating flow of data on communication lines

*(Continued on next slide)*

# Roles of a Communication Protocol



- **Error control**
  - Detect errors in messages to ensure transmission of correct messages
- **Precedence and order of transmission**
  - Ensure that all nodes get a chance to use communication lines and other resources
- **Connection establishment and termination**
  - How connections are established, maintained, and terminated
- **Data security**
  - Mechanisms for providing security and privacy of messages sent/received
- **Log information**
  - What types of log information the system should maintain for all jobs and data communications tasks

# Concept of Layered Protocols in Network Design



- Networks have modular design for easy and efficient handling of the system
- Consist of several modules, which are grouped into layers logically
- Each layer offers certain services to higher layers, shielding those layers from implementation details of services offered by lower layers
- Each layer has its own set of protocols
- Main reasons for using layered protocols are:
  - Layers makes their implementation more manageable
  - Provides well-defined interfaces between layers
  - Allows interaction between functionally paired layers in different locations
  - *Protocol suite, protocol family, or protocol stack* are terms used to refer to a collection of protocols of a network system

# Network Interface Card (NIC)



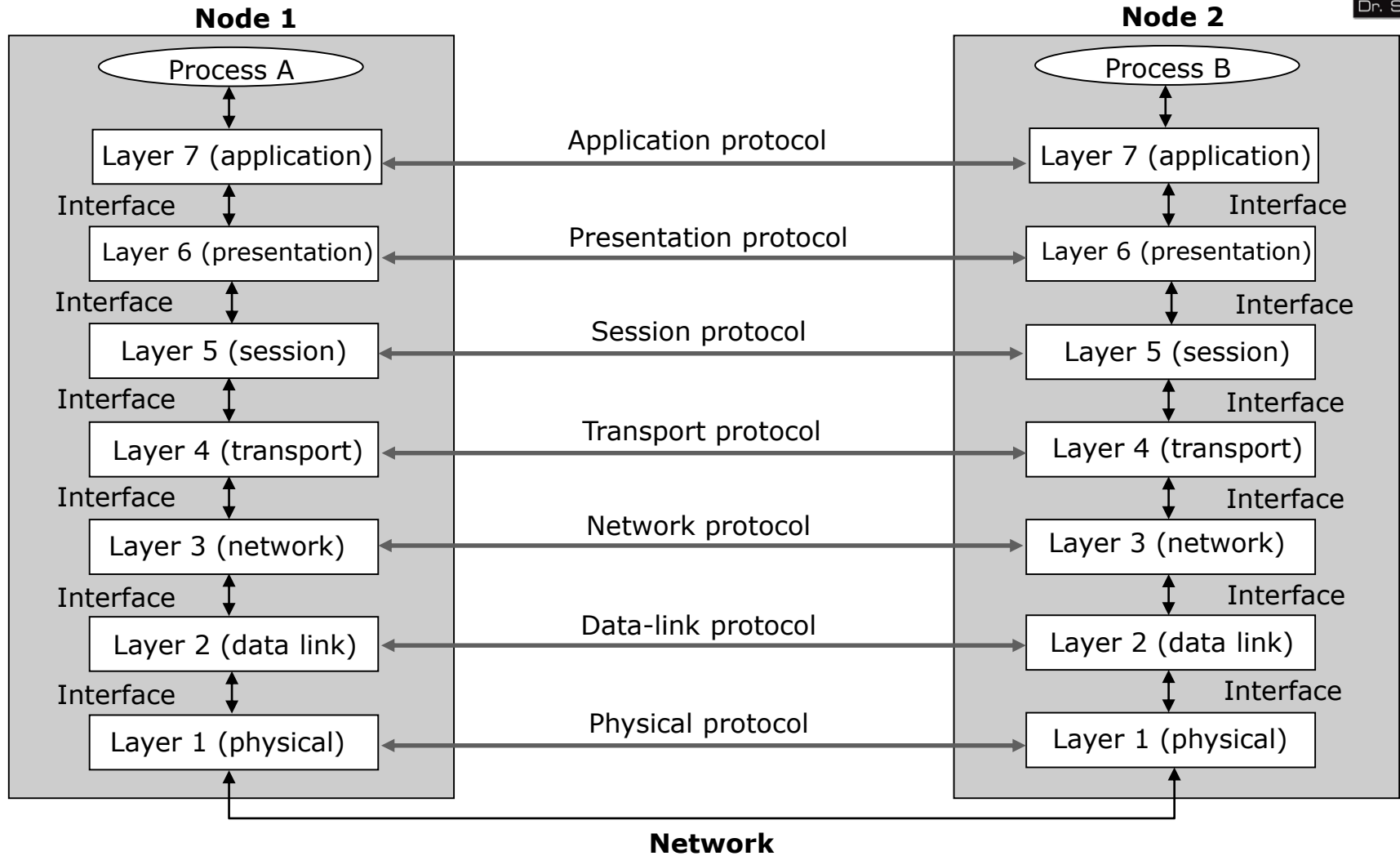
- *Network Interface Card (NIC or network card)* is a hardware device that connects a computer to a network, both functionally and physically
- It is an add-on card that is connected directly to a computer's I/O bus
- NIC's ROM has the network's physical-layer communication protocol

# 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

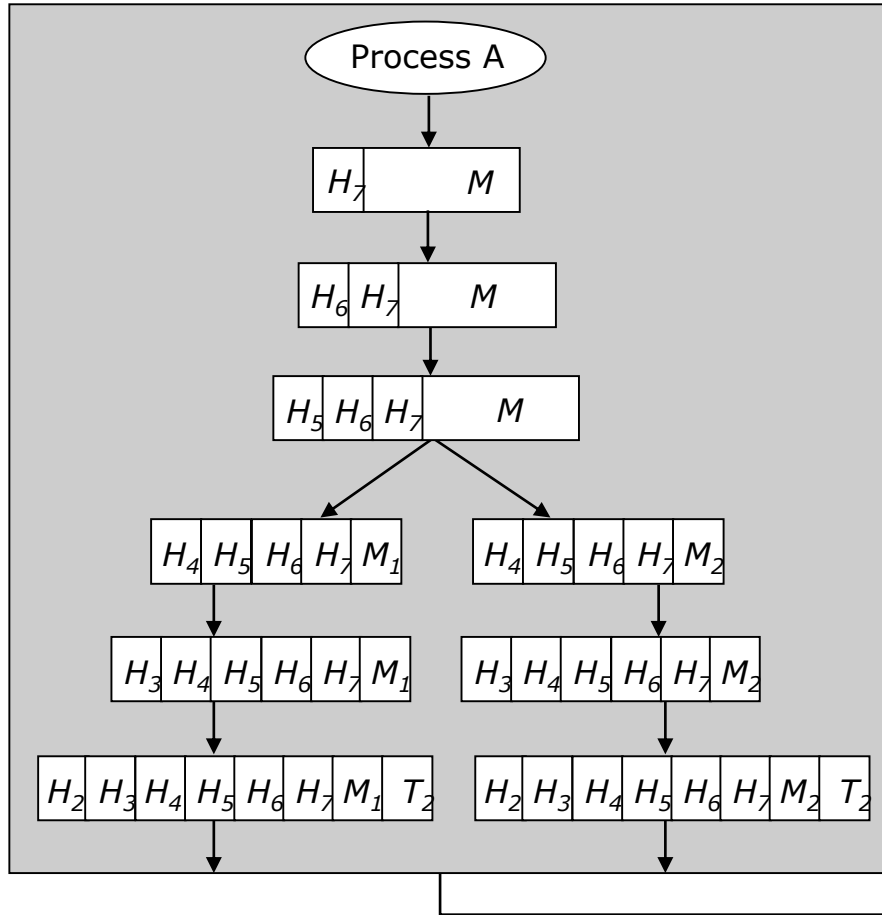
# Layers, Interfaces, and Protocols in the OSI Model



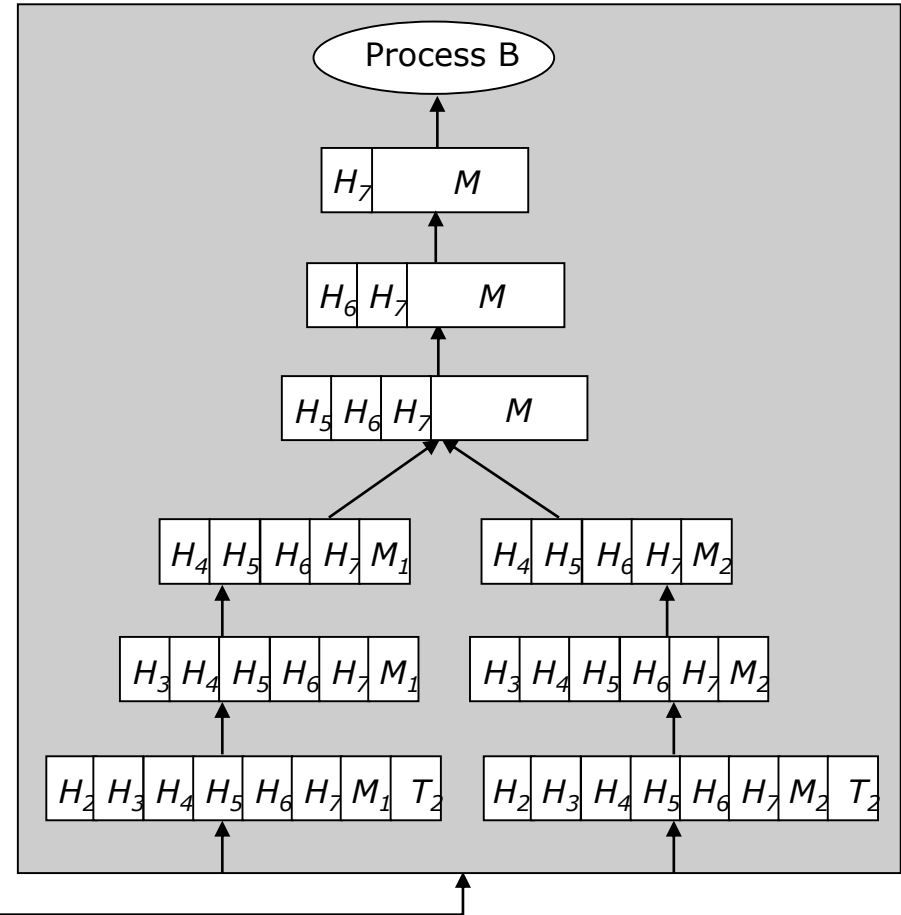
An example illustrating transfer of message  $M$  from sending node to the receiving node in the OSI model:  $H_n$ , header added by layer  $n$ ;  $T_n$ , trailer added by layer  $n$ .

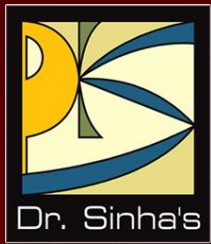


Sender node



Receiver node





# Internetworking Tools





# 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

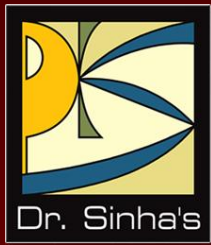


- Operates at network layer of the OSI model
- Used to interconnect those networks that use the same high-level protocols above network layer
- Smarter than bridges as they not only copy data from one network segment to another, but also choose the best route for the data by using routing table

# 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 Networks



# Wireless Computing Systems



- A wireless computing system uses wireless communication technologies for interconnecting computer systems
- It 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

# Issues in Wireless Computing Systems



- Lower bandwidth
- Variable bandwidth
- Higher error rate
- Increased security concern
- Dynamically changing network
- Lost or degraded connection
- Support for routing and location management functions
- Limited power

# Wireless Applications



- Interesting and important applications include:
  - Mobile e-commerce applications (m-commerce)
  - Web surfing
  - Access to corporate data by employees while they are traveling
  - Mobile video-on-demand
  - Location-sensitive services such as finding nearby movie theaters or restaurants



# Wireless Technologies



- 1G, 2G, 3G, 4G and 5G
- Wireless LAN
- WiMAX (Worldwide Interoperability for Microwave Access)
- Wireless Local Loop (WLL)
- Radio-router
- Multihop Wireless Network
- Wireless Application Protocol (WAP)

# Mobile Wireless Communication Technologies



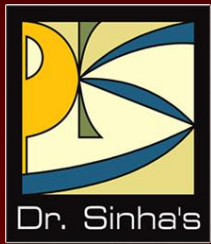
- The era of mobile wireless communication started in 1980.
- The five generations of mobile wireless communication technologies since then are 1G, 2G, 3G, 4G and 5G.
- Every new generation made mobile wireless communication faster than before and had improved or new features than earlier generation technologies



S. No.	Features	1G	2G	3G	4G	5G
1.	Year of launch	1980	1991	2000	2010	2020 (expected)
2.	Maximum communication speed	2.4 Kbps	64 Kbps, 384 Kbps with 2.5G	2 Mbps	1 Gbps	35.46 Gbps
3.	Technology	Analog	Digital, GSM, CDMA	CDMA 2000, UMTS, EDGE	LTE, MIMO, OFDM	WWWW, Massive MIMO, Millimeter Wave Mobile Communications



S. No.	Features	1G	2G	3G	4G	5G
4.	Key Services	Analog phone calls	Digital phone calls, and Messaging (SMS, MMS)	Digital phone calls, True multimedia messaging including video, Video conferencing, Mobile TV, Video games, Web-based applications	IP-based services including phone calls, Multimedia messaging, Web access, HDTV, Video conferencing, IP telephony, High-end mobile gaming, Mobile cloud computing	IP-based services including phone calls, Multimedia messaging, Web access, HDTV, Video conferencing, High-end mobile gaming, Mobile cloud computing, Interactive multimedia applications, Mobile applications based on device-to-device communication (IoT applications), etc. with far better experience than before.



# Distributed Computing Systems



# Distributed Computing Systems



- It is a configuration, which consists of many independent computer systems interconnected by a communication network
- It enables sharing of many hardware and software resources, as well as information, among several users who may be far away from each other
- It is more complex to build because its design must:
  - Enable it to use and manage a very large number of distributed resources effectively
  - Enable various nodes of the system to communicate with each other reliably
  - Include special security mechanisms to protect distributed shared resources and services against intentional or accidental security violations

# Main Advantages of Distributed Computing Systems



- Inherently distributed applications
- Information sharing among distributed users
- Resource sharing
- Better price-performance ratio
- Shorter response times and higher throughput
- Higher reliability
- Extensibility and incremental growth
- Better flexibility in meeting users' needs

# Keywords/Phrases



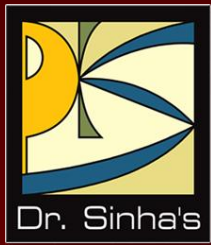
- 1G technology
  - 2G technology
  - 3G technology
  - 4G technology
  - 5G technology
  - Amplifier
  - Amplitude Modulation (AM)
  - Application layer
  - ARPANET
  - Asynchronous transmission
  - Bandwidth
  - Baud
  - Bridge
  - Broadband
  - Broadcast
  - Campus Area Network (CAN)
  - C-band transmission
  - Circuit switching
  - Coaxial cable
  - Common Carriers
  - Communication protocol
  - Communications satellite
  - Completely connected network
  - Computer network
  - Data-link layer
  - Demodulation
  - Dial-up line
  - Distributed Computing System
  - Ethernet
  - Fax modem
  - File Transfer Protocol (FTP)
  - 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)
- (Continued on next slide)*



# Keywords/Phrases



- Message switching
- 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
- 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)
- Wireless Application Protocol (WAP)
- Wide Area Network (WAN)
- WiMax
- Wireless LAN
- Wireless Local Loop (WLL)
- Wireless network



# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 18

**The Internet and  
Internet of Things**

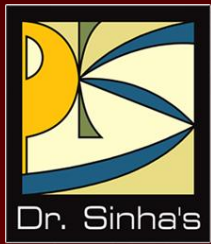


# Learning Objectives



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

- Definition and history of the Internet
- Its basic services
- The World Wide Web (WWW)
- WWW browsers
- Internet search engines
- Uses of the Internet
- Internet of Things (IoT)
- Benefits and Uses of IoT
- Challenges in Adoption of IoT



# Definition and History



# Definition and History



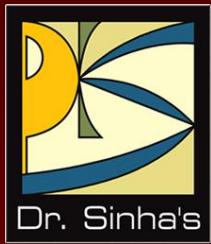
- The *Internet* is a network of computers, which links many different types of computers all over the world
- It is a network of networks sharing a common mechanism for addressing computers, and a common set of communication protocols
- The Internet has its root in the ARPANET system of the Advanced Research Project Agency of the U.S. Department of Defense
- ARPANET was the first WAN and had only four sites in 1969
- The Internet evolved from basic ideas of ARPANET for interconnecting computers

*(Continued on next slide)*

# Definition and History



- In 1989, the U.S. Government lifted restrictions on the use of the Internet, and allowed its usage for commercial purposes as well
- It now interconnects more than 50 Billion devices, and more than 5 Billion users around the world to communicate with each other
- The Internet continues to grow at a rapid pace



# Its Basic Services



# Basic Services of the Internet



- **Electronic Mail (e-mail):** Allows a 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 a user to move a file from one computer to another on the Internet
- **Telnet:** Allows a user to log in to another computer somewhere on the Internet
- **Usenet News:** Allows a group of users to exchange their views/ideas/information



# Electronic Mail



- E-mail is a rapid and productive communication tool because:
  - It is 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



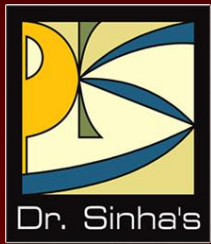
Some common uses of telnet service are:

- 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
- A moderated newsgroup ensures quality of posted messages
- *Netiquette* (network etiquette) deals with rules of framing messages that will not hurt others



# The World Wide Web (WWW)



# The World Wide Web (WWW)



- *World Wide Web (WWW or W3)* is the most popular and promising method of organizing and accessing information on the Internet
- *Hypertext* is a new way of information storage and retrieval that enables authors to structure information in novel ways
- A properly designed hypertext document can help users locate desired type of information rapidly
- Hypertext documents enable this by using a series of links
- A *link* is a special type of item in a hypertext document connecting the document to another document
- Hypertext documents on the Internet are known as *Web Pages*

(Continued on next slide...)

# The World Wide Web (WWW)



- Web page designers create Web Pages by using a special language called *HyperText Markup Language (HTML)*
- HTML is a subset of a more generalized language called *Standard Generalized Markup Language (SGML)*
- HTML is now a de-facto industrial standard for creating Web Pages
- The WWW uses client-server model, and an Internet Protocol called *HyperText Transport Protocol (HTTP)*
- Any computer on the Internet using the HTTP protocol is called a *Web Server*
- Any computer accessing that server is called a *Web Client*

# Example of Hypertext Document



Pradeep K. Sinha has been involved in the research and development of distributed systems for more than three decades. In addition to being the founding Vice Chancellor and Director of IIIT-Raipur, Dr. Sinha has worked with the **Centre for Development of Advanced Computing (C-DAC)**, Pune, India and the **Multimedia Systems Research Laboratory (MSRL) of Panasonic** in Tokyo, Japan.

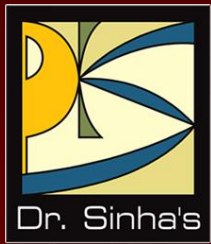
Links



# WWW Browsers



- To use a computer as a web client, a user needs to load on it a special software tool known as *WWW browser (browser)*
- Browsers provide following navigation facilities:
  - Do not require a user to log in to a server computer
  - Enable a user to visit a server computer's site directly and access information on it by specifying its URL (Uniform Resource Locator)
  - Enable a user to create and maintain a personal hotlist of favorite URL
  - Maintain a history of server computers visited by a user in a surfing session
  - Enable a user to download information in various formats



# Internet Search Engines



# Internet Search Engines



- *Internet search engine* is an application, which helps users locate web sites containing useful information and references
- To search information:
  - A user types the description of the information using the user interface of the search engine
  - The search engine then searches the requested information on the WWW and returns the results to the user
  - Results enable the user to locate the requested information quickly from the vast ocean of information available on the Internet

# Major Elements of Internet Search Engines



## ■ Search Request Interface

- Enables users to provide description of desired information to the search engine
- Search engine may allow specifications of simple keywords and phrases, combination of keywords and phrases using Boolean operators and exclusion/inclusion operators, and title and URL limiters

## ■ Information Discoverer

- Discovers information from the WWW and creates a database for the search engine
- Search engine uses the database to locate useful information during the search process
- Information discoverer accrues this information in two-ways:
  - In manual method, authors provide information about their websites

*(Continued on next slide...)*

# Major Elements of Internet Search Engines



- In automatic method, information discoverer collects the information using programs, such as web *crawlers*, *spiders*, *robots*
- **Presenter of Search Results**
  - Returns search results, ranking them in an order
  - Search engines often list search results in accordance to a relevance score
  - Relevance scores reflect the number of times a search term appears in a web page
  - Some search engines also allow users to control relevance score by giving different weights

# Some Popular Internet Search Engines



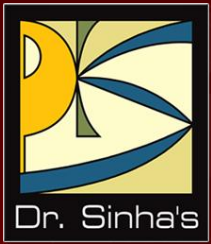
- **Google** ([www.google.com](http://www.google.com))
- **Yahoo** ([www.yahoo.com](http://www.yahoo.com))
- **Lycos** ([www.lycos.com](http://www.lycos.com))
- **Infoseek** ([www.infoseek.com](http://www.infoseek.com))
- **HotBot** ([www.hotbot.com](http://www.hotbot.com))
- **Inference Find** ([www.infind.com](http://www.infind.com))
- **Ixquick** ([www.ixquick.com](http://www.ixquick.com))

# 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
- Organization promotion
- Product promotion and feedback about products
- Customer support service
- On-line journals, magazines, encyclopedia, and dictionary
- On-line shopping
- World-wide video conferencing
- World-wide web casting



# Internet of Things





# What It Is?



IoT is enhanced form of the Internet, in which the enhancement takes care of the following two objectives:

1. Interconnection of things, in addition to computing devices, and
2. Automatic collection and transfer of data, without any help from human being

# Things in IoT?



- Refers to any object (device) with any kind of built-in sensors and having the ability to collect and transfer data over a network without manual intervention
- Also known as Smart Objects

# Benefits of IoT



- Increased accuracy
- Increased efficiency
- Reduced wastage
- Improved customer satisfaction

# IoT Applications (Uses of IoT)



- IoT is used in many sectors with millions of applications
- A few examples of IoT applications are:
  - Smart home
  - Smart city
  - Smart healthcare
  - Smart energy management
  - Smart transportation system
  - Smart agriculture/farming
  - Smart manufacturing/industry (Industry 4.0)

# Challenges in Adoption of IoT

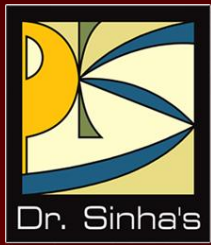


1. Interoperability issue
2. Issue of designing smart “things”
3. Unstructured data handling issue
4. Security and privacy issue
5. Reliability issue

# Keywords/Phrases



- Anonymous ftp site
- Browser
- Download
- Electronic mail (e-mail)
- File Transfer Protocol (FTP)
- Hypertext
- Hypertext Transport Protocol (HTTP)
- Industry 4.0
- Internet
- Internet of Things (IoT)
- Internet search engine
- Netiquette
- Newsgroup
- On-line shopping
- Publicly accessible sites
- Smart agriculture/farming
- Smart city
- Smart energy management
- Smart healthcare
- Smart home
- Smart manufacturing
- Smart objects
- Smart transportation system
- Standard Generalized Markup Language (SGML)
- Telnet
- Things
- Uniform Resource Locator (URL)
- Upload
- Usenet
- Video conferencing
- Web casting
- Web client
- Web crawler
- Web Server
- World Wide Web (WWW)



# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 19

**Multimedia**



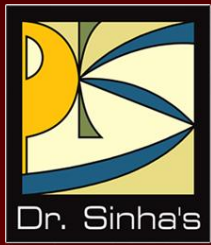
# Learning Objectives



## In this chapter you will learn about:

- What is multimedia?
- What is a multimedia computer system?
- Main components of multimedia and their associated technologies
- Common multimedia applications
- Media center computer





# What is Multimedia?



# Multimedia



- A *medium* (plural *media*) is something that a presenter can use for presentation of information
- Two basic ways to present information are:
  - **Unimedium presentation:** Presenter uses a single medium to present information
  - **Multimedia presentation:** Presenter uses more than one medium to present information
- Multimedia presentation of any information enhances comprehension capability of its user because it involves use of multiple senses by the user

# Common Media



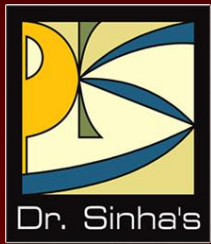
- Common media for storage, access, and transmission of information are:
  - Text (alphanumeric characters)
  - Graphics (line drawings and images)
  - Animation (moving images)
  - Audio (sound)
  - Video (Videographed real-life events)
- Multimedia in information technology refers to use of more than 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, data size for multimedia information is much larger than plain text information
- Hence, 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)*



# Multimedia Components



# Text Media



- Alphanumeric characters are used to present information in text form. Computers are widely used for text processing
- Keyboards, electronic writing pads, OCRs, terminal 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



- *Computer graphics* deals with tools and technologies for generating, representing, manipulating, and displaying pictures with a computer
- Graphics is an important component of multimedia applications because a picture is a powerful way to illustrate information

# Graphics Types



## ■ Line drawings

- Drawings are illustrations in the form of 2D and 3D pictures created from mathematical representation of simple objects like lines, circles, arcs, etc.
- Area of computer graphics that deals with this type of pictures is known as generative graphics

## ■ Images

- Computers store pixels of an image as a two-dimensional matrix
- This two-dimensional representation is called image resolution
- Each pixel is composed of three components: red (R), green (G), blue (B)
- On a display screen, each component of a pixel corresponds to a phosphor

*(Continued on next slide...)*



# Graphics Types

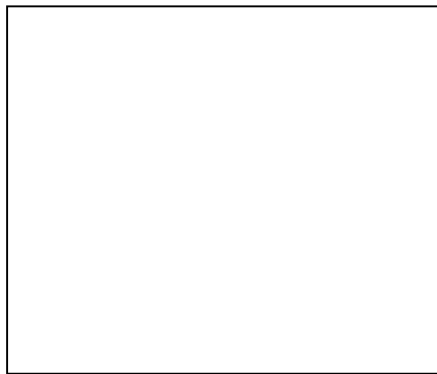


- Phosphor glows when excited by an electron gun
- Number of bits used to represent a pixel is called *color depth*
- Color depth in turn is determined by the size of video buffer in display circuitry
- Resolution and color depth determine the presentation quality and size of image storage
- Applications use image compression techniques to compress images to reduce their size before storing or transmitting them
- Area of computer graphics that deals with this type of pictures is known as *cognitive graphics*

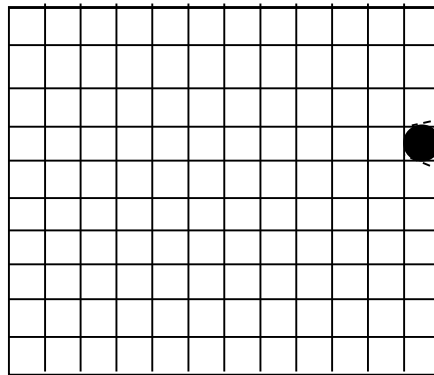
# An Image Composition



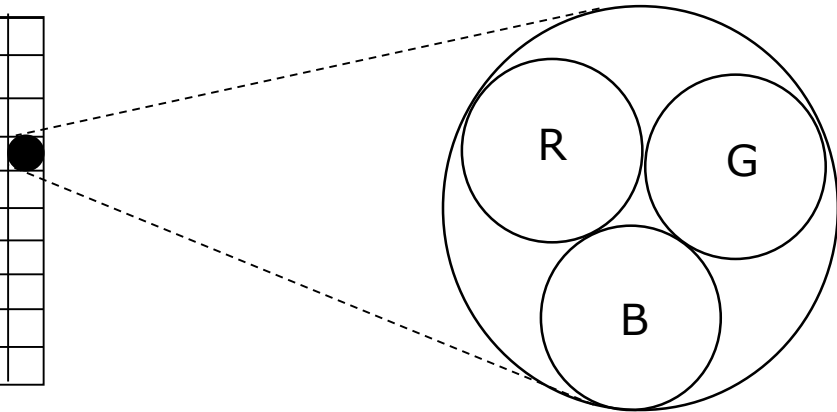
Image sample



Divided into pixels



A pixel composition



# Graphics: Hardware Requirements



- Locating device with a video display terminal and drawing software
- Flatbed or rectangular coordinate digitizer
- Scanners
- Digital camera or frame-capture hardware
- Computer screens with graphics display capability
- Laser printers
- Plotters

# Graphics: Software Requirements



- Drawing and painting software
- Screen-capture software
- Clip art
- Graphics importing
- 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

# Animation: Hardware Requirements



- Image generation tools and devices such as scanners, digital cameras, and video capture board
- Computer monitor with image display capability and a graphics accelerator board

# Animation: Software Requirements



- Animation-creation software
- Screen-capture software
- Animation clips
- Animation file importing
- Software support for high resolution
- Recording and playback capability
- 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
- Virtual reality applications use animation extensively



# Audio



- *Computer audio* deals with tools and technologies for synthesizing, recording, and playing of audio or sound with a computer

# Analog and Digital Audio



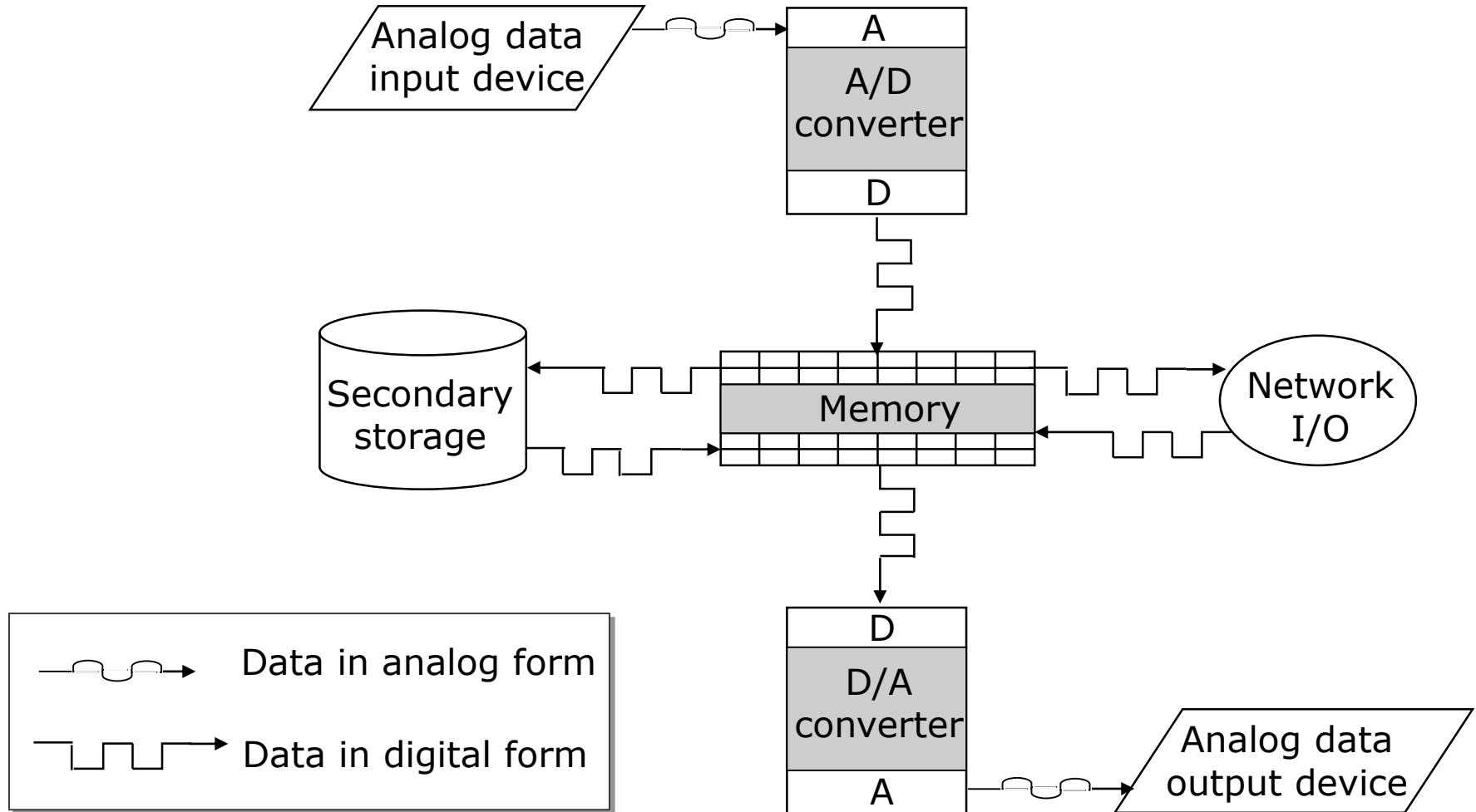
- Audio information travels in natural medium in the form of sound waves that are analog
- To enable a computer to deal with audio information, a device must convert sound waves from analog to digital form
- *Transducer* is a device that converts signals from one form to another
- Microphone is a transducer that converts sound waves to electrical signals, and loudspeaker is a transducer that converts electrical signals to sound waves
- Analog signals are continuous tone of smooth fluctuations, while digital signals are discrete values in the form of numbers

# Analog and Digital Audio



- A/D conversion transforms an analog input into a series of numeric representation by digitization
- D/A conversion is the reverse process of transforming a sequence of discrete numbers back into continuous analog signal

# Roles of A/D and D/A Converters



# Audio Processing



- Audio processing requires a sound-board which has:
  - A/D and D/A converters
  - Connector for a speaker or headphone
  - MIDI input port to input sound from an external MIDI device
  - MIDI output port to output recorded sound to the MIDI device
  - MIDI synthesizer
  - Volume control

# Audio: Hardware Requirements



- Sound-board with A/D and D/A converters
- Input device for audio input to record sound
- Output device to listen audio output
- Computer may use MIDI devices
- Computer may also enable generation of synthesized sound by using keyboard
- Sound editors enable cut and paste of sound sequences
- Audio mixers enable combining of multiple channels of sound with controls like synchronization points

# Audio: Software Requirements



- Audio clips
- Audio file importing
- Software support for high quality sound
- Recording and playback capability
- Text-to-speech conversion software
- Speech-to-text conversion software
- Voice recognition software

# Video



- *Video* deals with tools and technologies for recording and displaying a sequence of images (frames) at a reasonable speed to create an impression of motion
- For jerk-free full motion video, the display device must display 25 to 30 frames per second
- Video is an important component of multimedia applications because it is useful for illustrating concepts involving movement



# Analog and Digital Video



- Video information travels in natural medium in the form of light waves that are analog
- To enable a computer to deal with video information, a device must convert light waves from analog to digital form
- Video camera is a transducer that converts light waves to electrical signals, and monitor is a transducer that converts electrical signals to light waves
- Analog-to-digital conversion of video also involves sampling and quantization of analog video signals

# Video: Hardware Requirements

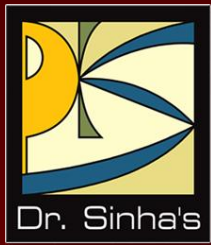


- Video camera
- Video monitor
- Video board with A/D and D/A converters, and connectors for video camera and video monitor
- Video editors

# Video: Software Requirements



- **Video clips**
  - It is a library of video clips
  - A user can import a video clip directly from this library for use in a multimedia application
- **Recording and playback capability**
  - Enables users to control recording and display of a video sequence



# Multimedia Applications

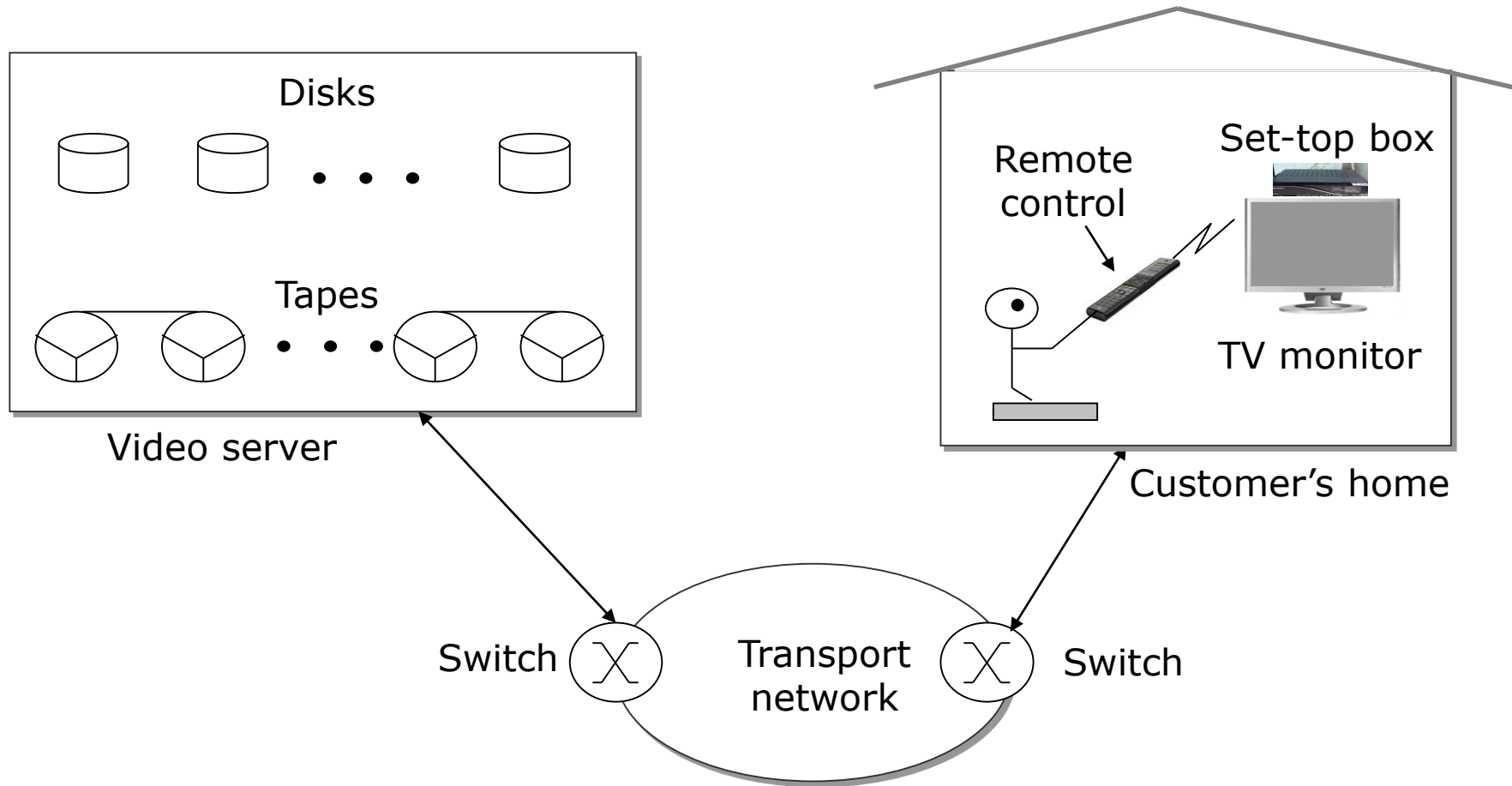


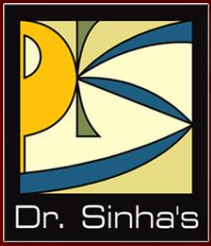
# Multimedia Applications



- Multimedia books and e-Books
- Digital library
- Multimedia presentation
- Foreign language learning
- Video games
- Special effects in movies
- Animation films
- Animated advertisements
- Multimedia kiosk
- Virtual shops and shopping malls
- Multimedia conferencing
- Smart/Interactive TV

# A Video-on-Demand System





# Media Center Computer



# 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)

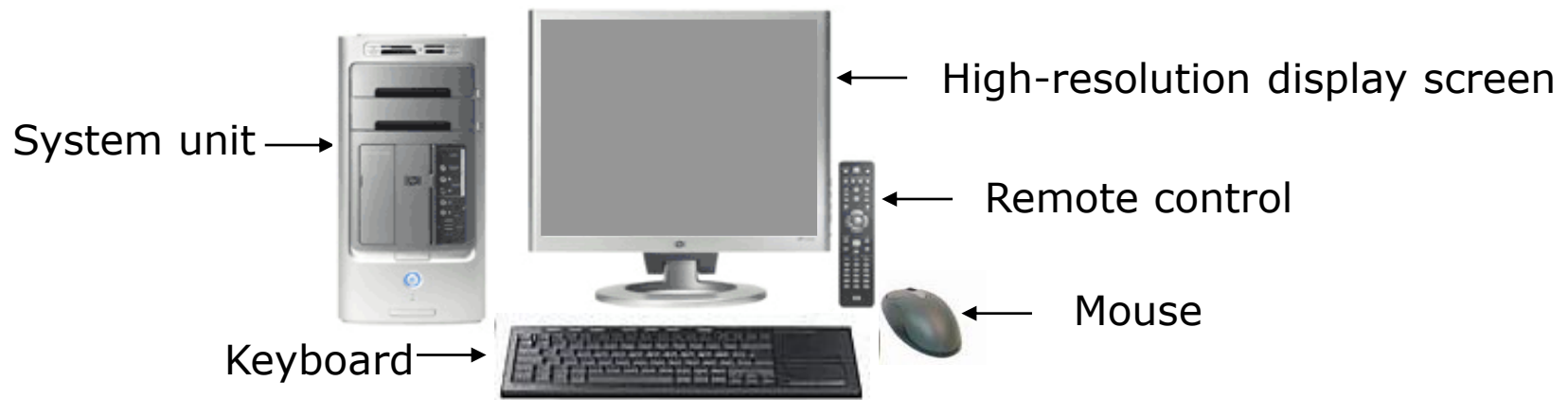
*(Continued on next slide...)*



# Media Center Computer



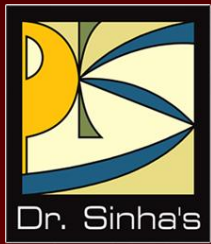
- Any PC with good graphics display card, multimedia components, and a TV tuner card can serve as a media center PC by installing media center applications on it
- Some operating systems provide features of media centre applications integrated into OS itself
- Such operating systems are called media center OS



# Keywords/Phrases



- Animation
- Audio
- Clip art
- Cognitive graphics
- Computer Aided Design (CAD)
- Computer Aided Manufacturing (CAM)
- Digital library
- Frames
- Generative graphics
- Graphics
- Interactive TV
- Multimedia
- Media Center Computer
- Pixel
- Refresh rate
- Text
- Transducer
- Transition effects
- Unimedia presentation
- Video
- Virtual reality



# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 20

## Classification of Computers



# Learning Objectives



## In this chapter you will learn about:

- Classifications of computers
- Common types of computers used today
- Characteristic features of various types of computers in use today

# Computer Classification



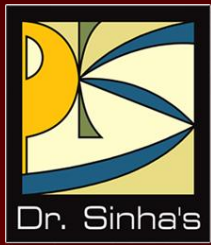
- Traditionally, computers were classified by their size, processing speed, and cost
- Based on these factors, computers were classified as microcomputers, minicomputers, mainframes, and supercomputers
- However, with rapidly changing technology, this classification is no more relevant
- Today, computers are classified based on their mode of use

# Types of Computers



Based on their mode of use, computers are classified as:

- Notebook computers (Laptops)
- Personal computers (PCs)
- Workstations
- Mainframe systems
- Supercomputers
- Client and server computers
- Handheld computers



# Notebook Computers (Laptops)



# 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 only around 2 kg or less.
- Comfortably placed on ones lap while being used. Hence, they are also called *laptop PCs*.
- 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



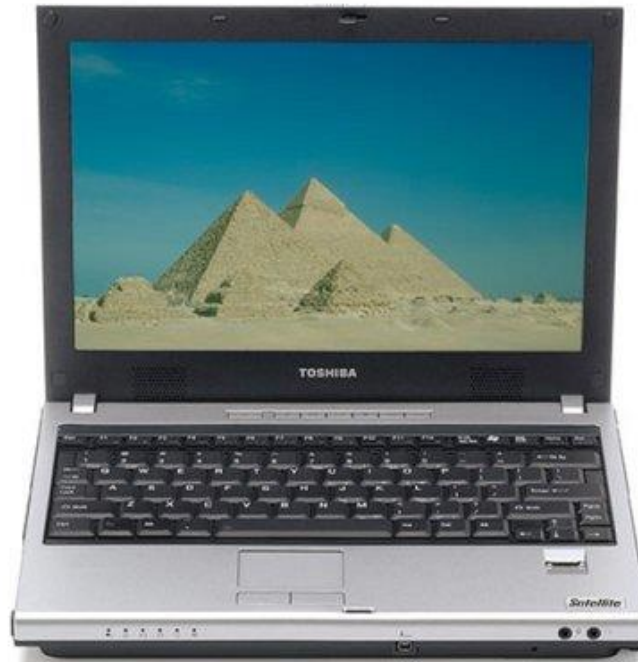
- Designed to operate with chargeable batteries
- Mostly used for word processing, spreadsheet computing, data entry, and power point presentations
- Normally run MS-DOS, MS WINDOWS Mac OS, or Linux operating system
- Some manufacturers are also offering models with GNU/Linux or its distributions
- Each device of laptop is designed to use little power and remain suspended if not used

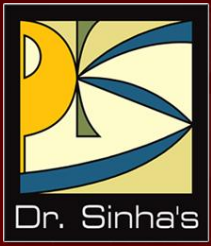
# Notebook Computers



Foldable flat screen →

Keyboard, trackball, hard disk, I/O ports, etc. are in this unit →





# Personal Computers (PCs)

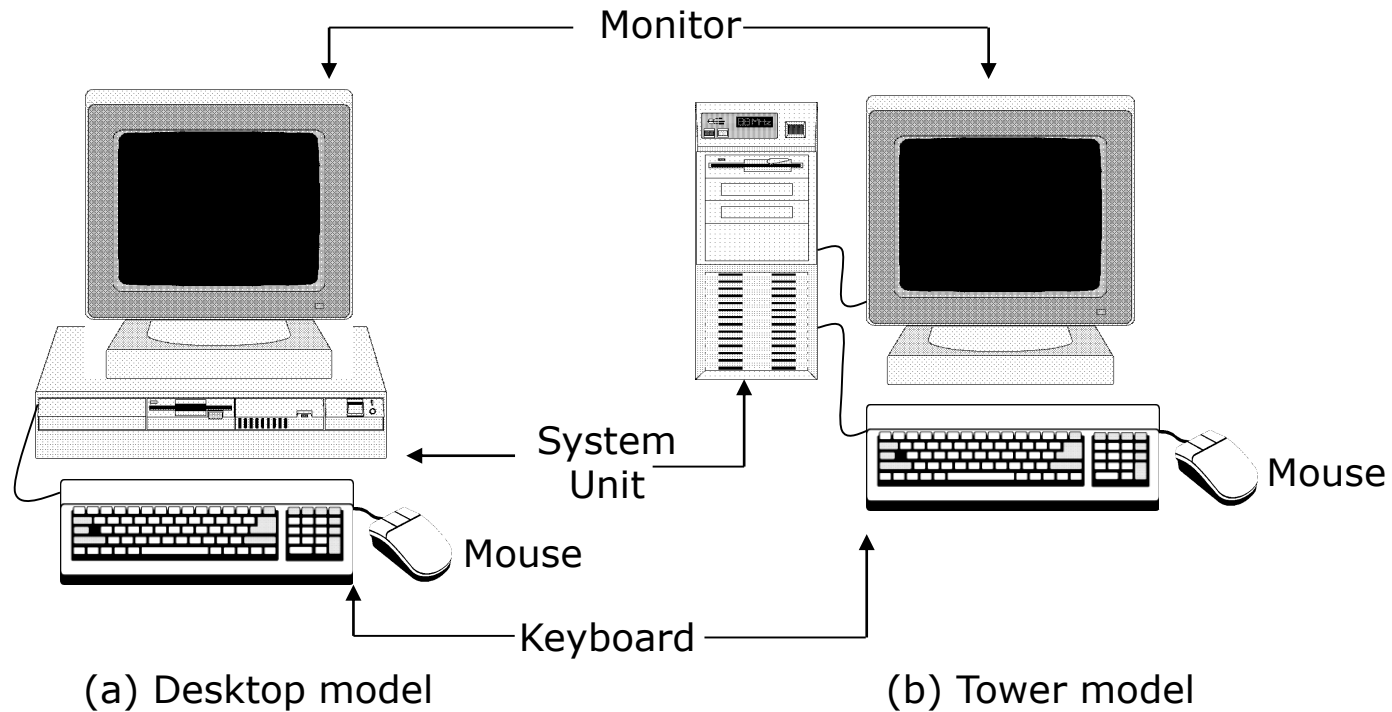


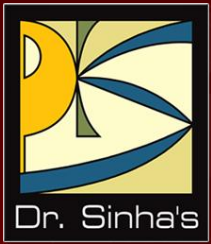
# Personal Computers (PCs)



- 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, Mac OS, Linux, and UNIX

# Common PC Models





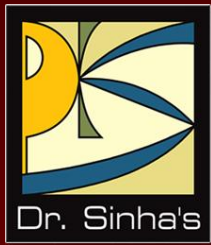
# Workstations



# 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 server version of Windows, Mac OS, Linux, or UNIX operating system
- Operating system is generally designed to support multiuser environment



# Mainframe Systems





# 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



- 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

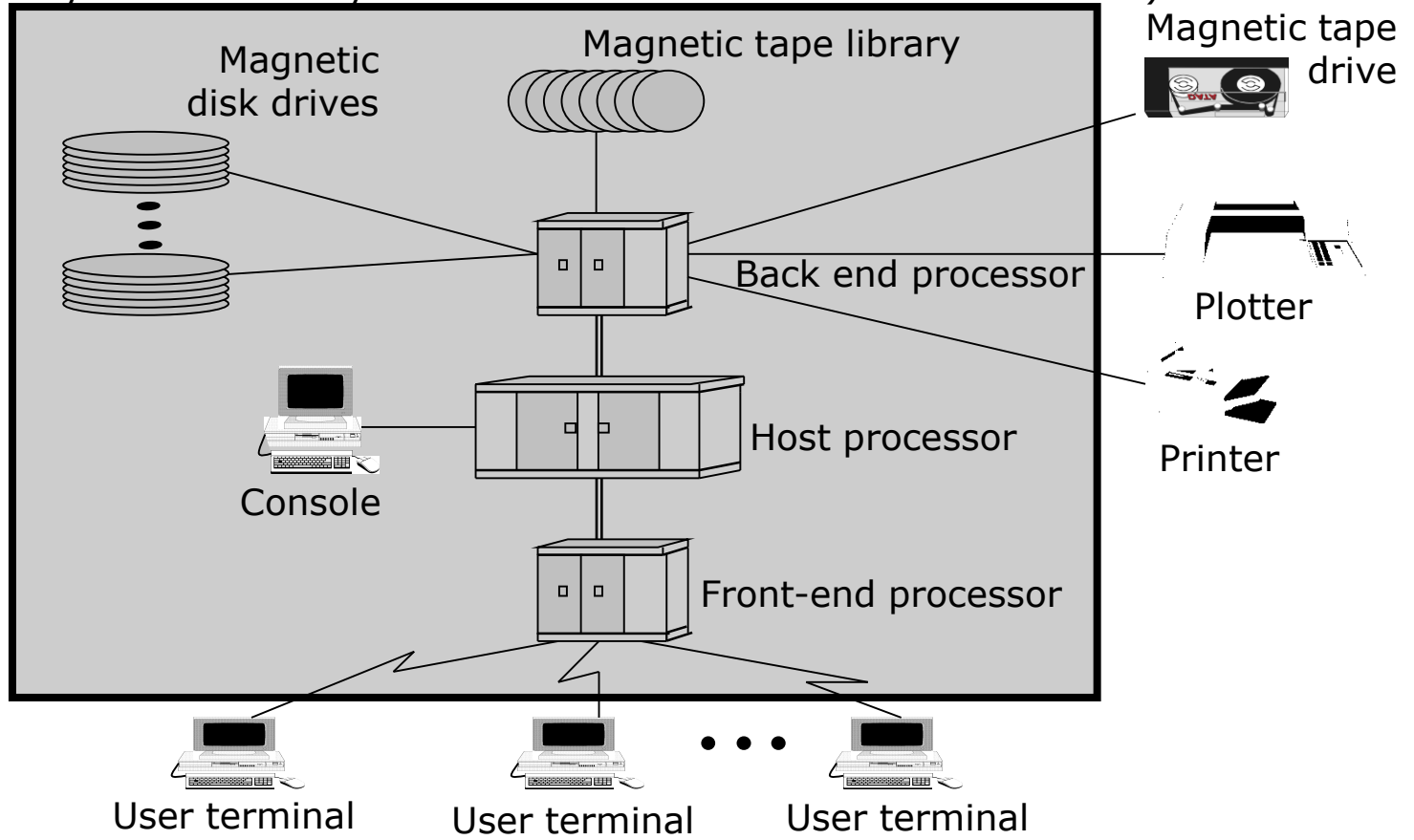


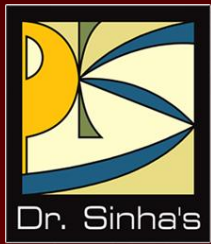
## USERS ROOM

(Entry restricted to authorized users)

## SYSTEM ROOM

(Entry restricted to system administrators and maintenance staff)





# Supercomputers



# 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
  - Creating special effects for movies and TV programs

*(Continued on next slide)*

# Supercomputers



- Supercomputers also support multiprogramming
- Supercomputers primarily address processor-bound applications

# Parallel Processing Systems

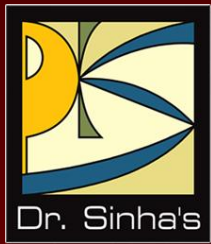


- Use multiprocessing and parallel processing technologies to solve complex problems faster
- Also known as *parallel computers* or *parallel processing systems*
- Modern supercomputers employ hundreds of processors and are also known as *massively parallel processors*

# C-DAC's PARAM Yuva Supercomputer







# Client and Server Computers



# Client and Server Computers



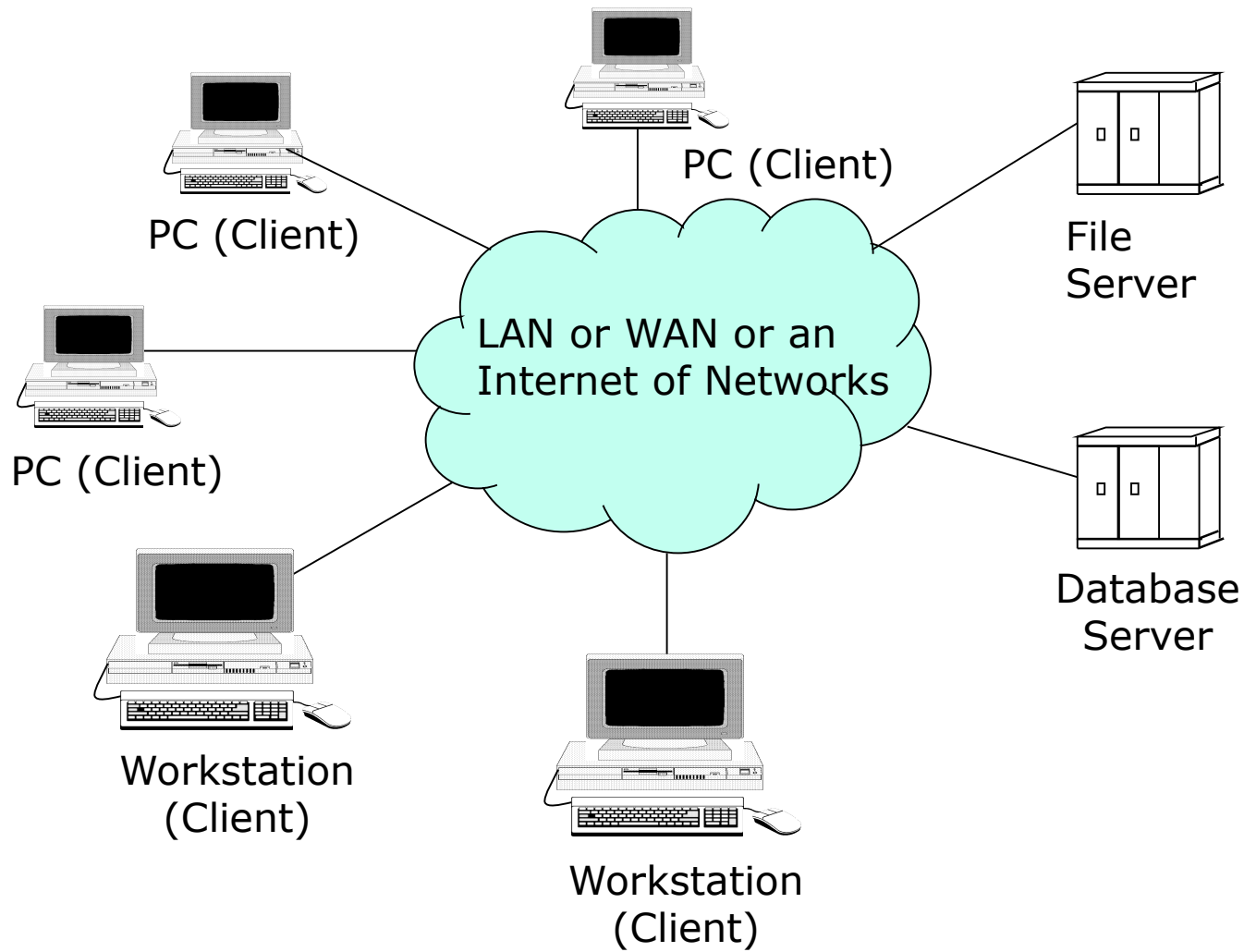
- Client-server computing environment has multiple clients, one/more servers, and a network
- *Client* is a PC/workstation with user-friendly interface running client processes that send service requests to the server
- *Server* is generally a relatively large computer that manages a shared resource 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

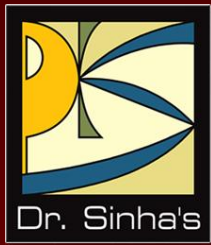
# Client-Server Computing



- 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



# 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) Tablet PC



(b) PDA/Pocket PC



(c) Smartphone

# Comparison of Different Types of Computers



Key features Types of computers	Notebook	Personal Computer	Workstation	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 also plays 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



# Comparison of Different Types of Computers



Key features Types of computers	Notebook	Personal Computer	Workstation	Mainframe system	Super computer	Client	Server	Handheld
Display facility	Foldable flat screen small display	Medium size display screen	Large-screen display that can display high-resolution graphics	Generally not available	Generally not available	Medium to large screen display	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, Mac OS, Linux	MS-DOS, MS-Windows, Windows-NT, Mac OS, Linux, Unix	Server version of Windows, Mac OS, Linux or Unix	A variant of Unix, or proprietary	A variant of Unix, or proprietary	MS-DOS, MS-Windows, Windows-NT, Mac OS, Linux, Unix	Server version of Windows, Mac OS, Linux or Unix	MS-Windows Mobile, iOS, Palm OS, Symbian OS, Linux, Blackberry OS

# Comparison of Different Types of Computers

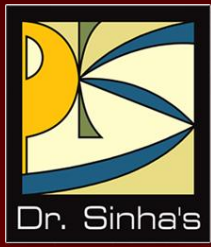


Key features Types of computers	Notebook	Personal Computer	Workstation	Mainframe system	Super computer	Client	Server	Handheld
Popular usage	Word processing; Spreadsheet; Data Entry; Preparing presentation materials; and Making presentations	Personal computing needs of individuals either at their work 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	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, Apple, Toshiba	IBM, Apple, Compaq, Dell, Siemens, Toshiba, Hewlett-Packard, Lenovo	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, and Supercomputer vendors	Nokia, Sony, Apple, Samsung, 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



# Computer Fundamentals

Pradeep K. Sinha

Priti Sinha

Chapter 21

## Introduction to C Programming Language

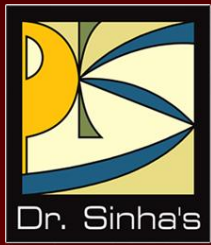


# Learning Objectives



## 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



# Basic Features and Rules



# Features of C



- 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 Fractional Form



- Must have at least one digit
- Must have one and only one decimal point
- + or - sign is optional
- No special characters (other than + and - sign) are allowed
- Examples are:      5.3,      +18.59,      -0.46

# 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



- A C variable is an entity whose value may vary during program execution
- It 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



<b>Data Type</b>	<b>Minimum Storage Allocated</b>	<b>Used for Variables that can contain</b>
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



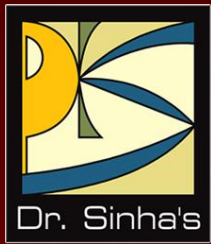
- *Keywords* (or reserved words) are predefined words whose meanings are known to C compiler
- C has 32 keywords
- Keywords cannot be used as variable names

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	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



- 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  $\rightarrow$  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 '+'

# Arithmetic Operators



Operator	Meaning with Example	Associativity	Precedence
<b>Arithmetic 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>			
~	Complement; ~x means All 1s are changed to 0s and 0s to 1s	R → L	2
&	AND; x & y means x AND y	L → R	8
	OR; x   y means x OR y	L → R	10
^	Exclusive OR; x ^ y means x ⊕ y	L → R	9
<<	Left shift; x << 4 means shift all bits in x four places to the left	L → R	5
>>	Right shift; x >> 3 means shift all bits in x three places to the right	L → R	5
&=	x &= y means x = x & y	R → L	14
=	x  = y means x = x   y	R → L	14
^=	x ^= y means x = x ^ y	R → L	14
<<=	x <<= 4 means shift all bits in x four places to the left and assign the result to x	R → L	14
>>=	x >>= 3 means shift all bits in x three places to the right and assign the result to x	R → L	14

# Data Access Operators

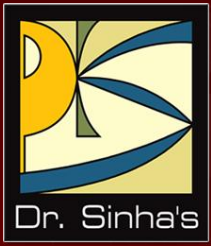


Operator	Meaning with Example	Associativity	Precedence
<b>Data Access Operators</b>			
<code>x[y]</code>	Access $y^{\text{th}}$ element of array <code>x</code> ; <code>y</code> starts from zero and increases monotonically up to one less than declared size of array	$L \rightarrow R$	1
<code>x.y</code>	Access the member variable <code>y</code> of structure <code>x</code>	$L \rightarrow R$	1
<code>x -&gt;y</code>	Access the member variable <code>y</code> of structure <code>x</code>	$L \rightarrow R$	1
<code>&amp;x</code>	Access the address of variable <code>x</code>	$R \rightarrow L$	2
<code>*x</code>	Access the value stored in the storage location (address) pointed to by pointer variable <code>x</code>	$R \rightarrow L$	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



# Statements



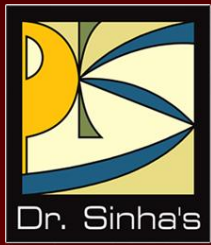
- C program is a combination of statements written between { and } braces
- Each statement performs a set of operations
- Null statement, represented by ";" or empty {} braces, does not perform any operation
- A simple statement is terminated by a semicolon ";"
- Compound statements, called *statement 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 */`  
  `{`  
  `}`





# I/O Operations



# 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



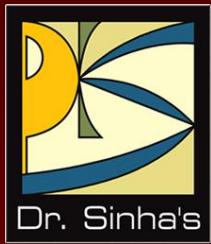
## Output:

Maths marks = 92

Science marks = 87

English marks = 78

Percentage marks obtained = 85.66



# Preprocessor Directives



# Preprocessor Directives



- *Preprocessor* is a program that prepares a program for the C compiler
- Examples of some common preprocessor directives in C are:

Preprocessor directive	Use
#include	Used to look for a file and place its contents at the location where this preprocessor directives is used
#define	Used for macro expansion
#ifdef..#endif	Used for conditional compilation of segments of a program



# Examples of Preprocessor Directives



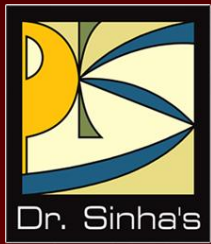
```
#include <stdio.h>
#define PI 3.1415
#define AND  &&
#define ADMIT      printf ("The candidate can be admitted");
```

```
#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 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	
#	String forming operator	Operators
##	Token pasting operator	
defined	same as #ifdef	



# Pointers, Arrays and Strings

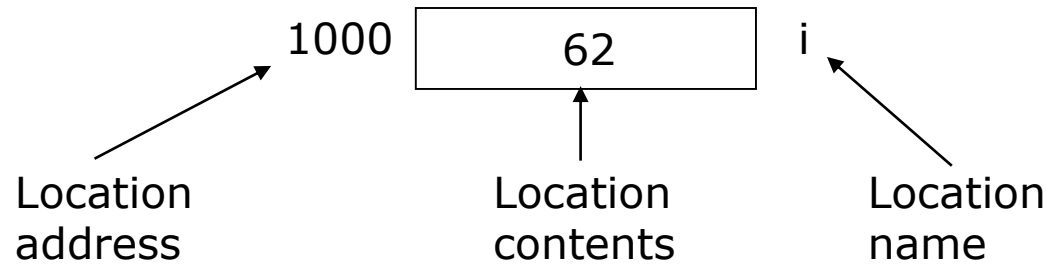


# Pointers



- 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



Address of i = 1000  
Value of i = 62

```
int i = 62;
int *p;
int j;
p = &i;           /* p becomes 1000 */
j = *p;          /* j becomes 62 */
j = 0;           /* j becomes zero */
j = *(&i)        /* j becomes 62 */
```

# Array



- An array is a collection of fixed number of elements in which all elements are of the same data type
- It is a homogeneous, linear, and contiguous memory structure
- Its 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
- An array can also be declared as multi-dimensional such as `Matrix2D[10][10]`

# Illustrating Arrays Concept



1010	92
1008	63
1006	82
1004	66
1002	84
1000	45

```
int marks[6];
```

Each element  
being an int  
occupies 2 bytes

```
marks[0] = 45
marks[1] = 84
.
.
.
marks[5] = 92
```

(a) An array of  
integers having  
6 elements

1012	10.25
1008	250.00
1004	155.50
1000	82.75

```
float price[4];
```

Each element  
being a float  
occupies 4 bytes

```
price[0] = 82.75
price[1] = 155.50
.
.
.
price[3] = 10.25
```

(b) An array of  
real numbers  
having 4 elements

1005	Y
1004	A
1003	B
1002	M
1001	O
1000	B

```
char city[6];
```

Each element  
being a char  
occupies 1 byte

```
city[0] = 'B'
city[1] = 'O'
.
.
.
city[5] = 'Y'
```

(c) An array of  
characters  
having 6 elements

# String



- A string is a one-dimensional array of characters terminated by a null character ('\0')
- It is initialized at declaration as  

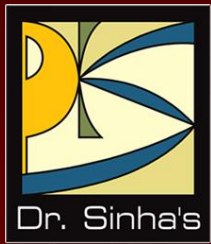
```
char name[] = "PRADEEP";
```
- Its individual elements can be accessed in the same way as we access array elements such as `name[3] = 'D'`
- Strings are used for text processing
- C provides a rich set of string handling library functions



# 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
stricmp	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



# User Defined Data Types (UDTs)



- UDT is composite data type whose composition is not included in language specification
- Programmer declares them in a program where they are used
- Two types of UDTs are:
  - Structure
  - Union

# Structure



- It is a UDT containing a number of data types grouped together
- Its constituents data types may or may not be of different types
- It 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*)
- It 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



- It is a UDT referring to same memory location using several data types
- It is a 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



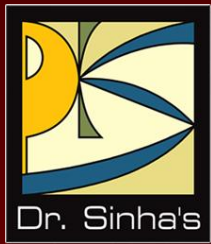
```
unionNum
{
    int intNum;
    unsigned
    unsNum'
};
union Num Number;
```

# 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



- *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)
- They 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



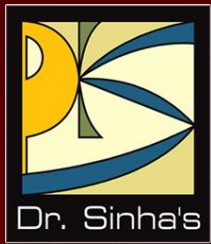
- Break: Causes unconditional exit from *for*, *while*, *do*, or *switch* constructs. Control is transferred to the statement immediately outside the block in which *break* appears.
- Continue: Causes unconditional transfer to next iteration in a *for*, *while*, or *do* construct. Control is transferred 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



- **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 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 or terminate the loop

# Examples of "for" and "while" Constructs



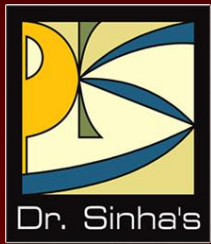
- ```
for (i=0; i < 10; i++)  
    printf("i = %d", i);
```
- ```
while (i < 10)  
{  
    printf("i = %d", i);  
    i++;  
}
```

# 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



- Functions (or subprograms) are building blocks of a program
- All functions must be declared and defined before use
- Function declaration requires *function name*, *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);
```

*(Continued on next slide)*

# Sample C Program (Program-2)



```
switch(day)
{
    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