# 2. Digital Logic and Microprocessor



Er. Pralhad Chapagain

# Syllabus

**2.1 Digital logic**: Number Systems, Logic Levels, Logic Gates, Boolean algebra, Sum-of-Products Method, Product-of-Sums Method, Truth Table to Karnaugh Map. (AExE0201)

**2.2 Combinational and arithmetic circuits**: Multiplexetures, Demultiplexetures, Decoder, Encoder, Binary Addition, Binary Subtraction, operation on Unsigned and Signed Binary Numbers. (AExE0202)

**2.3 Sequential logic circuit**: RS Flip-Flops, Gated Flip-Flops, Edge Triggered Flip-Flops, Mater- Slave Flip-Flops. Types of Registers, Applications of Shift Registers, Asynchronous Counters, Synchronous Counters. (AExE0203)

**2.4 Microprocessor**: Internal Architecture and Features of microprocessor, Assembly Language Programming. (AExE0204)

**2.5 Microprocessor system**: Memory Device Classification and Hierarchy, Interfacing I/O and Memory Parallel Interface. Introduction to Programmable Peripheral Interface (PPI), Serial Interface, Synchronous and Asynchronous Transmission, Serial Interface Standards. Introduction to Direct Memory Access (DMA) and DMA Controllers. (AExE0205)

**2.6 Interrupt operations**: Interrupt, Interrupt Service Routine, and Interrupt Processing. (AExE0206)

**2.1 Digital logic**: Number Systems, Logic Levels, Logic Gates, Boolean algebra, Sum-of-Products Method, Product-of-Sums Method, Truth Table to Karnaugh Map. (AExE0201)

# Number System

- Binary
- Octal
- Decimal
- Hexadecimal

- Conversions

# POSITIVE AND NEGATIVE LOGIC

## POSITIVE LOGIC:

- When we use binary 1 for high voltage and binary 0 for low voltage then it is called positive logic.

## NEGATIVE LOGIC:

- When we use binary 0 for high voltage and binary 1 for low voltage then it is called Negative logic.

Truth Table

| A | Y=A’ |
|---|---|
| 1 | 0 |
| 0 | 1 |

Negative Logic NOT gate

Truth Table

| A | B | Y=A+B |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

Negative Logic OR gate

Truth Table

| A | B | Y=A.B |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

Negative Logic AND gate

- Positive logic AND gate is equivalent to Negative logic OR gate and vice versa.
- Positive logic NAND gate is equivalent to Negative logic NOR gate and vice versa.
- Positive logic XOR gate is equivalent to Negative logic XNOR gate and vice versa.

# Logic Gates

- Basic Gates (NOT, AND, OR)
- Universal Gates (NAND, NOR)
- Exclusive Gates (XOR , XNOR)

# Boolean Algebra

- Boolean Algebra is used to analyze and simplify the digital (logic) circuits. It uses only the binary numbers i.e. 0 and 1. It is also called as Binary Algebra or logical Algebra. Boolean algebra was invented by George Boole in 1854.
- Boolean Laws

## Commutative law

Any binary operation which satisfies the following expression is referred to as commutative operation.

(i) $A.B = B.A$ (ii) $A + B = B + A$

# Boolean Algebra

## Associative law

This law states that the order in which the logic operations are performed is irrelevant as their effect is the same.

(i) $(A.B).C = A.(B.C)$ (ii) $(A + B) + C = A + (B + C)$

## Distributive law

Distributive law states the following condition.

$$A.(B + C) = A.B + A.C$$

## AND law

These laws use the AND operation. Therefore they are called as **AND** laws.

(i) $A.0 = 0$ (ii) $A.1 = A$

(iii) $A.A = A$ (iv) $A.\overline{A} = 0$

# Boolean Algebra

## OR law

These laws use the OR operation. Therefore they are called as **OR** laws.

$$(i)\ A + 0 = A \qquad (ii)\ A + 1 = 1$$

$$(iii)\ A + A = A \qquad (iv)\ A + \overline{A} = 1$$

## INVERSION law

This law uses the NOT operation. The inversion law states that double inversion of a variable results in the original variable itself.

$$\overline{\overline{A}} = A$$

# DE MORGANS THEOREM

## Theorem 1

$$\overline{A.B} = \overline{A} + \overline{B}$$

NAND = Bubbled OR

## Theorem 2

$$\overline{A + B} = \overline{A}.\overline{B}$$

NOR = Bubbled AND



FOR MCQ: https://www.sanfoundry.com/discrete-mathematics-questions-answers-de-morgan-laws/

# BOOLEAN ALGEBRA

## Dual Theorem:

- Starting with a Boolean relation, we can derive another Boolean relation, called its dual by the following steps:
  - Changing each OR sign into AND sign
  - Changing each AND sign into OR sign

Ex

| S.N. | Given Expression | Dual of given expression |
|---|---|---|
| 1 | A + AB = A | A. (A + B) = A |
| 2 | A + A'B = A + B | A. (A' + B) = A.B |
| 3 | A + A' = 1 | A.A' = 0 |
| 4 | (A + B)(A + C) = A + BC | A.B + A.C = A. (B + C) |

# STANDARD FORM AND CANONICAL FORM

## CANONICAL FORM:

- Max Term
- Min Term

## Max Term:

- Each Max term is obtained from an OR logic of n variables with each variable being unprimed if the corresponding bit is zero (0) and primed if one (1).

| A | B | C | Max term | designation |
|---|---|---|---|---|
| 0 | 0 | 0 | A+B+C | $M_0$ |
| 0 | 0 | 1 | A+B+C’ | $M_1$ |
| 0 | 1 | 0 | A+B’+C | $M_2$ |
| 0 | 1 | 1 | A+B’+C’ | $M_3$ |
| 1 | 0 | 0 | A’+B+C | $M_4$ |
| 1 | 0 | 1 | A’+B+C’ | $M_5$ |
| 1 | 1 | 0 | A’+B’+C | $M_6$ |
| 1 | 1 | 1 | A’+B’+C’ | $M_7$ |

# STANDARD FORM AND CANONICAL FORM

## Min Term:

- Each Min term is obtained from an AND logic of n variables with each vari…nding bit is one (1) and primed if …

| A | B | C | Min term | designation |
|---|---|---|---|---|
| 0 | 0 | 0 | A’B’C’ | $m_0$ |
| 0 | 0 | 1 | A’B’C | $m_1$ |
| 0 | 1 | 0 | A’BC’ | $m_2$ |
| 0 | 1 | 1 | A’BC | $m_3$ |
| 1 | 0 | 0 | AB’C’ | $m_4$ |
| 1 | 0 | 1 | AB’C | $m_5$ |
| 1 | 1 | 0 | ABC’ | $m_6$ |
| 1 | 1 | 1 | ABC | $m_7$ |

# STANDARD FORM AND CANONICAL FORM

## STANDARD FORMS:

- In standard form the terms that form the function may contain one, two or any number of literals/ variables. There are two types of standard forms.
  - Sum of Product (SOP)
  - Product of Sum (POS)

## Sum of Product (SOP)

- SOP is a Boolean expression containing terms with AND logic of 1 or more literals.
- E.g. F=XYZ + X'YZ + X'Y'Z

## Product of Sum(POS)

- POS is a Boolean expression containing terms with OR logic of 1 or more literals.
- E.g. F=( X + Y + Z ) (X' + Y + Z) (X' +Y' +Z)

# Boolean Algebra - MCQ

1. Algebra of logic is termed as _____
a) Numerical logic
b) Boolean algebra
c) Arithmetic logic
d) Boolean number

2. Boolean algebra can be used _____
a) For designing of the digital computers
b) In building logic symbols
c) Circuit theory
d) Building algebraic functions

# KARNAUGH MAP (K-MAP) (SOP)

- K-MAP is regarded as a diagrammatic or pictorial form of a truth table.
- The map is a diagram made up of squares.
- Each square represents one min/ max term.
- The MAP represents a visual diagram of all possible ways of function, may ne expressed in a standard form.

## Basic K-MAP

| B / A | | B' | B |
|---|---|---|---|
| | | 0 | 1 |
| A' | 0 | A'B' | A'B |
| A | 1 | AB' | AB |

Fig: Two variable K-MAP

# KARNAUGH MAP (K-MAP) (SOP)

## Three variable K-MAP

- There are 8 min terms for 3 binary variables.
- A MAP consists of 8 squares.
- The min terms are arranged not in a binary sequence but in sequence similar to gray code.
- The characteristics of the sequence is that only one bit is changes from one sequence to another.

| BC / A | | B'C' | B'C | BC | BC' |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| A' | 0 | $m_0$ | $m_1$ | $m_3$ | $m_2$ |
| A | 1 | $m_4$ | $m_5$ | $m_7$ | $m_6$ |

Fig: Three variable K-MAP

# KARNAUGH MAP (K-MAP) (SOP)

## Four variable K-MAP

- There are 16 min terms for 4 binary variables.
- A MAP consists of 16 squares.

| CD / AB | | C'D' | C'D | CD | CD' |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| A'B' | 00 | $m_0$ | $m_1$ | $m_3$ | $m_2$ |
| A'B | 01 | $m_4$ | $m_5$ | $m_7$ | $m_6$ |
| AB | 11 | $m_{12}$ | $m_{13}$ | $m_{15}$ | $m_{14}$ |
| AB' | 10 | $m_8$ | $m_9$ | $m_{11}$ | $m_{10}$ |

Fig: Four variable K-MAP

## Simplification:

- One square box represents one min term giving a term of four literals.
- Two adjacent square box represents a term of three literals
- Four adjacent square box represents a term of two literals.
- Eight square box represents one min term giving a term of one literals.
  - Sixteen adjacent square box represents a function 1
  - Zero square box represents a function 0.

# KARNAUGH MAP (K-MAP) (SOP)

## DON'T CARE CONDITION:

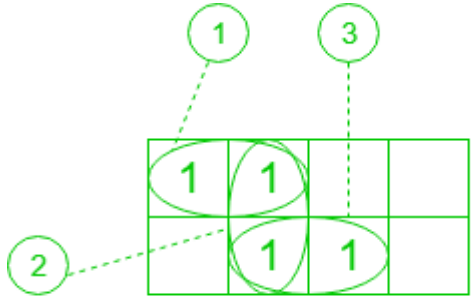
- There are some condition of inputs for which output is not specified and such output does not affect the whole system, which are known as Don't Care condition.
- The Don't care min terms are denoted by 'X' sign.

## IMPLICANTS IN K-MAP

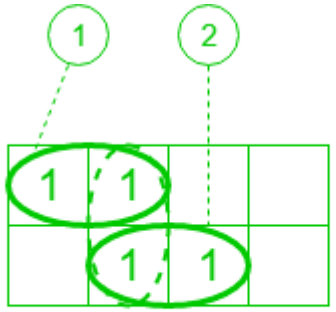
- **Prime Implicants**
  - A group of square or rectangle made up of bunch of adjacent minterms which is allowed by definition of K-Map are called prime implicants(PI) i.e. all possible groups formed in K-Map.
- **Essential Prime Implicants**
  - These are those sub cubes (groups) which cover at least one minterm that can't be covered by any other prime implicant. Essential prime implicants(EPI) are those prime implicants which always appear in final solution.

# KARNAUGH MAP (K-MAP) (SOP)

E.G.




No. of Prime Implicants = 3




No. of Essential Prime Implicants = 2

# KARNAUGH MAP (K-MAP) (SOP)

## K-MAP EXAMPLE:

- Simplify using K-MAP and design a logic circuit.

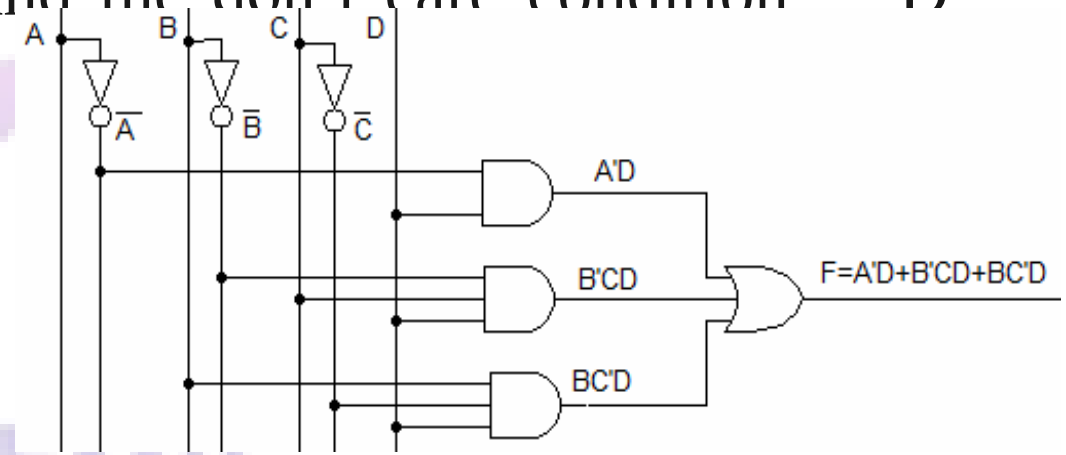
F (A, B, C, D) = ∑(1,3,7,11,13) and the don't care condition D (

| CD / AB | | C'D' | C'D | CD | CD' |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| A'B' | 00 | X | 1 | 1 | X |
| A'B | 01 | 0 | X | 1 | 0 |
| AB | 11 | 0 | 1 | 0 | 0 |
| AB' | 10 | 0 | 0 | 1 | 0 |

A'D

B'CD

BC'D




F = A'D + B'CD + BC'D

Fig: Combinational Design

# Boolean Algebra - MCQ

4. F(X,Y,Z,M) = X`Y`Z`M`. The degree of the function is _____
a) 2
b) 5
c) 4
d) 1

5. A _____ value is represented by a Boolean expression.
a) Positive
b) Recursive
c) Negative
d) Boolean

7. What are the canonical forms of Boolean Expressions?
a) OR and XOR
b) NOR and XNOR
c) MAX and MIN
d) SOM and POM

8. Which of the following is/are the universal logic gates?
a) OR and NOR
b) AND
c) NAND and NOR
d) NOT

9. The logic gate that provides high output for same inputs _____
a) NOT
b) X-NOR
c) AND
d) XOR



10. The _____ of all the variables in direct or complemented from is a maxterm.
a) addition
b) product
c) moduler
d) subtraction

# Boolean Algebra – MCQ –SET B

1. Which of the following bits is the negation of the bits “010110”?
a) 111001
b) 001001
c) 101001
d) 111111

2. Which of the following option is suitable, if A is “10110110”, B is”11100000” and C is”10100000”?
a) C=A or B
b) C=~A
c) C=~B
d) C=A and B

3. How many bits string of length 4 are possible such that they contain 2 ones and 2 zeroes?
a) 4
b) 2
c) 5
d) 6

4. If a bit string contains {0, 1} only, having length 5 has no more than 2 ones in it. Then how many such bit strings are possible?
a) 14
b) 12
c) 15
d) 16

5. If A is “001100” and B is “010101” then what is the value of A (Ex-or) B?
a) 000000
b) 111111
c) 001101
d) 011001

6. The Ex-nor of this string “01010101” with “11111111” is?
a) 10101010
b) 00110100
c) 01010101
d) 10101001

7. What is the one’s complement of this string “01010100”?
a) 10101010
b) 00110101
c) 10101011
d) 10101001

8. What is the 2’s complement of this string “01010100”?
a) 10101010
b) 00110100
c) 10101100
d) 10101001

# Boolean Algebra – MCQ –SET B

9. If in a bits string of {0,1}, of length 4, such that no two ones are together. Then the total number of such possible strings are?

a) 1

b) 5

8 7

d) 4

10. Let A: "010101", B=?, If { A (Ex-or) B } is a resultant string of all ones then which of the following statement regarding B is correct?

a) B is negation of A

b) B is 101010

c) {A (and) B} is a resultant string having all zeroes

) All of the mentioned

# Boolean Algebra – MCQ –SET C

1. The terms in SOP are called _____
a) max terms
b) min terms
c) mid terms
d) sum terms

2. A sum of products expression is a product term (min term) or several min terms ANDed together.
a) True
b) False

3. Which of the following is an incorrect SOP expression?
a) x+x.y
b) (x+y)(x+z)
c) x
d) x+y

4. The corresponding min term when x=0, y=0 and z=1.
a) x.y.z'
b) X'+Y'+Z
c) X+Y+Z'
d) x'.y'.z

5. LSI stands for _____
a) Large Scale Integration
b) Large System Integration
c) Large Symbolic Instruction
d) Large Symbolic Integration

6. Which operation is shown in the following expression: (X+Y').(X+Z).(Z'+Y')
a) NOR
b) ExOR
c) SOP
d) POS

# Boolean Algebra – MCQ –SET C

7. The number of min terms for an expression comprising of 3 variables?

a) 8
b) 3
c) 0
d) 1

8. The number of cells in a K-map with n-variables.

a) 2n
b) $n^2$
c) $2^n$
d) n

9. The output of AND gates in the SOP expression is connected using the _____ gate.

a) XOR
b) NOR
c) AND
d) OR



10. The expression A+BC is the reduced form of _____

a) AB+BC
b) (A+B)(A+C)
c) (A+C)B
d) (A+B)C

# Boolean Algebra – MCQ –SET D

1. A Karnaugh map (K-map) is an abstract form of _____ diagram organized as a matrix of squares.
a) Venn Diagram
b) Cycle Diagram
c) Block diagram
d) Triangular Diagram

2. There are _____ cells in a 4-variable K-map.
a) 12
b) 16
c) 18
d) 8

3. Each product term of a group, w'.x.y' and w.y, represents the _____ in that group.
a) Input
b) POS
c) Sum-of-Minterms
d) Sum of Maxterms

4. It should be kept in mind that don't care terms should be used along with the terms that are present in _____
a) Minterms
b) Expressions
c) K-Map
d) Latches

5. The prime implicant which has at least one element that is not present in any other implicant is known as _____
a) Essential Prime Implicant
b) Implicant
c) Complement
d) Prime Complement

6. Product-of-Sums expressions can be implemented using _____
a) 2-level OR-AND logic circuits
b) 2-level NOR logic circuits
c) 2-level XOR logic circuits
d) Both 2-level OR-AND and NOR logic circuits

# Boolean Algebra – MCQ –SET D

7. Each group of adjacent Minterms (group size in powers of twos) corresponds to a possible product term of the given _____
a) Function
b) Value
c) Set
d) Word

8. Don't care conditions can be used for simplifying Boolean expressions in _____
a) Registers
b) Terms
c) K-maps
d) Latches

**2.2 Combinational and arithmetic circuits**: Multiplexetures, Demultiplexetures, Decoder, Encoder, Binary Addition, Binary Subtraction, operation on Unsigned and Signed Binary Numbers. (AExE0202)

# ADDER - HALF

- Binary adders nay be of two types:
  - Half Adder
  - Full Adder

Sum ar

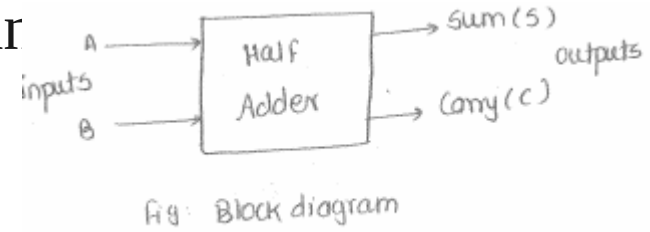


Fig: Block diagram

## Half Adder:

- Half adder is a combinational logic circuit with two inputs and two outputs.
- It is the basic building block for addition of two single bit numbers
- This circuit has two outputs namely,
- Tr

| Inputs | | Outputs | |
|---|---|---|---|
| A | B | Sum | Carry |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

# ADDER - HALF

K-map for sum

| A \ B | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

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

K-map for carry

| A \ B | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

$Carry = AB$

Combinational circuit is:

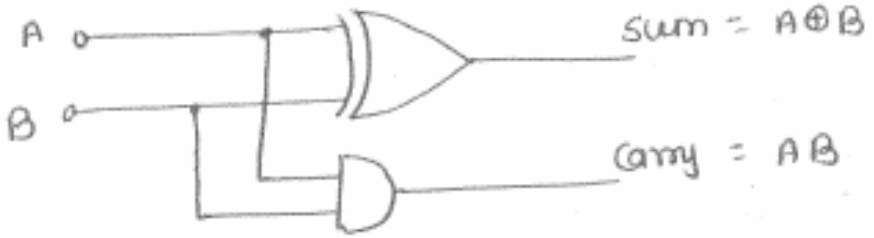



Fig: An Half adder circuit

Half adder using basic gates:

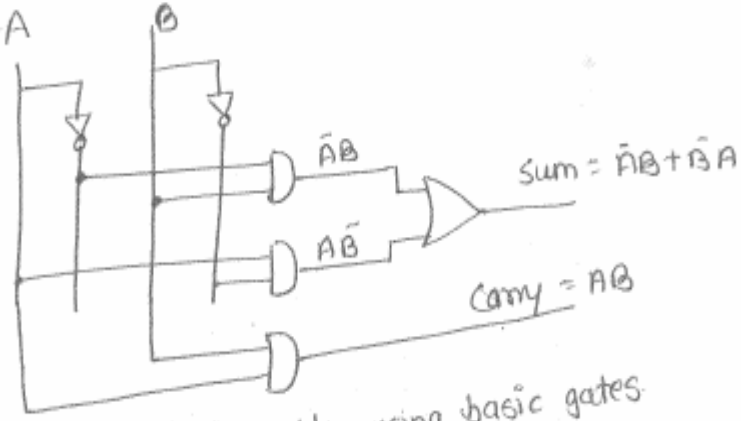



Fig: An Half adder using basic gates

- Limitations:
  - The addition of three bits is not possible to perform by using an half adder circuit.

# ADDER - FULL

## Full Adder:

- To overcome the drawback of an half adder circuit, a 3-single bit adder circuit called full adder is developed.
- Basically, a full adder is a three input and two output combinational circuit.

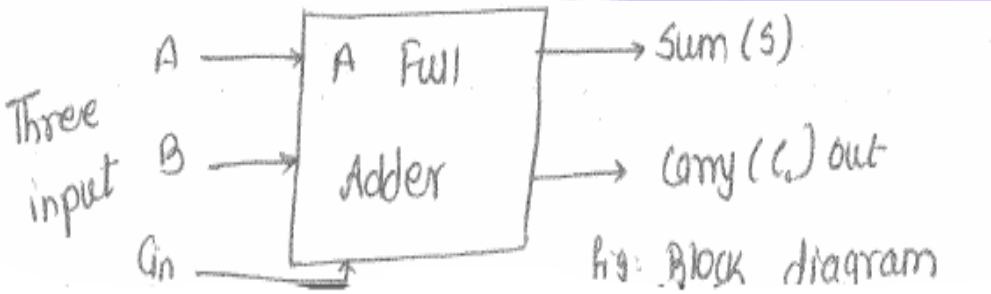


Fig: Block diagram

- T

| Inputs | | | Outputs | |
|---|---|---|---|---|
| A | B | $C_{in}$ | Sum | Carry ($C_o$) |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

ding block of the 4 bit/ 8 bit binary/ BCD adder Ics such as 7483.

# ADDER - FULL

K-map for sum (S)

| A \ BCin | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |

$S = A'B'C_{in} + A'BC_{in}' + AB'C_{in}' + ABC_{in}$

$= A \oplus B \oplus C_{in}$

K-map for carry out ($C_o$)

| A \ BCin | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 |

$C_o = AC_{in} + BC_{in} + AB$

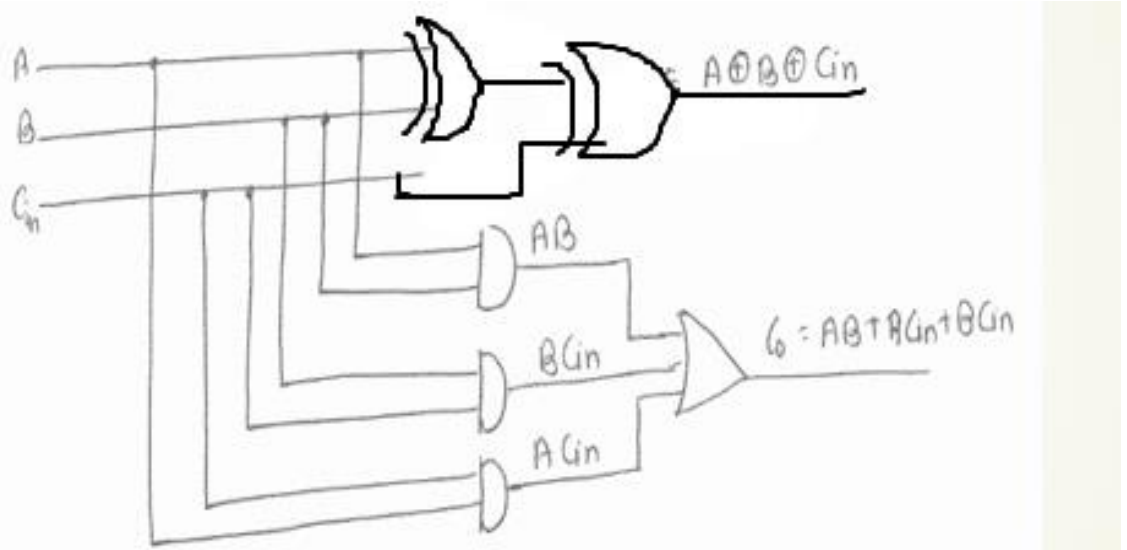



Fig: A full adder circuit

# SUBTRACTOR - HALF

- Binary Subtractor nay be of two types:
  - Half Subtractor
  - Full Subtractor
- In subtraction (A-B), A is called m… S…

| Inputs | | Outputs | |
|---|---|---|---|
| A | B | Difference (A-B) | Borrow ($B_o$) |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

## Half Subtractor:

- Half subtractor may be defined as a combinational circuit with two inputs and two outputs (i.e. difference and borrow)

K-map for Difference (D)

- Truth

| A \ B | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

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

# SUBTRACTOR - HALF

K-map for borrow o/p

| A \ B | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |

Borrow = $\bar{A}B$

Logic diagram

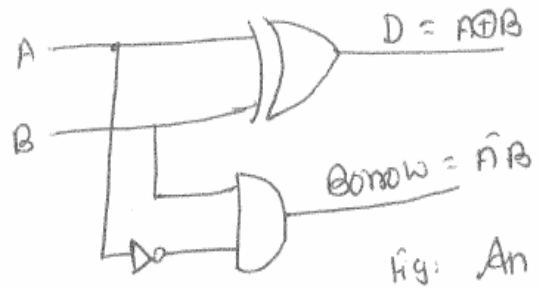



Fig: An half subtractor circuit.

Using Basic gate:

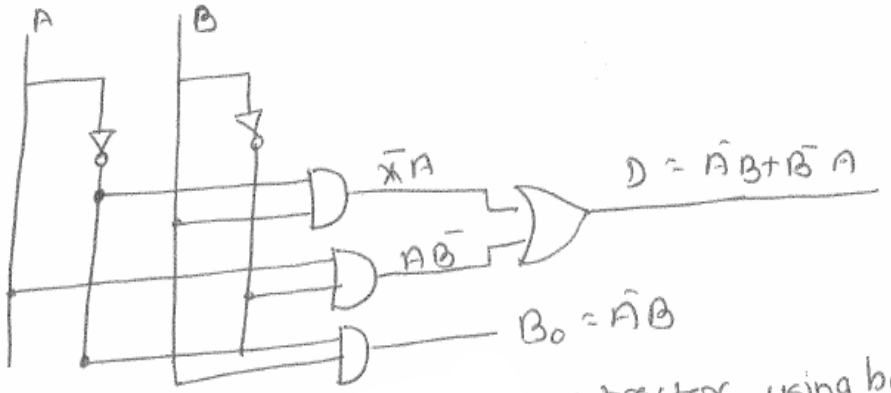



Fig: Half subtractor using basic gates

# SUBTRACTOR - FULL

## Full Subtractor:

- A Full Subtractor is a combinational circuit with three inputs A, B, $B_{in}$ and two outputs D (Difference) and Borrow ($B_{o.}$)
- Here A is the minuend, B is the subtrahend, $B_{in}$ is the borrow produced by the previous stage, D is the difference output and $B_o$ is the borrow output.
- T

| Inputs | | | Outputs | |
|---|---|---|---|---|
| A | B | $B_{in}$ | Difference (A-B-$B_{in}$) | Borrow ($B_o$) |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

# SUBTRACTOR - FULL

K-map

for difference output

| A \ BBin | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |

$$D = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + ABB_{in}$$

$$= B_{in}(\bar{A}\bar{B} + AB) + \bar{B}_{in}(\bar{A}B + A\bar{B})$$

$$= B_{in}(\overline{A \oplus B}) + \bar{B}_{in}(A \oplus B)$$

$$= A \oplus B \oplus B_{in}$$

for Borrow output

| A \ BBin | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |

$$B_0 = \bar{A}B_{in} + \bar{A}B + BB_{in}$$

# SUBTRACTOR - FULL

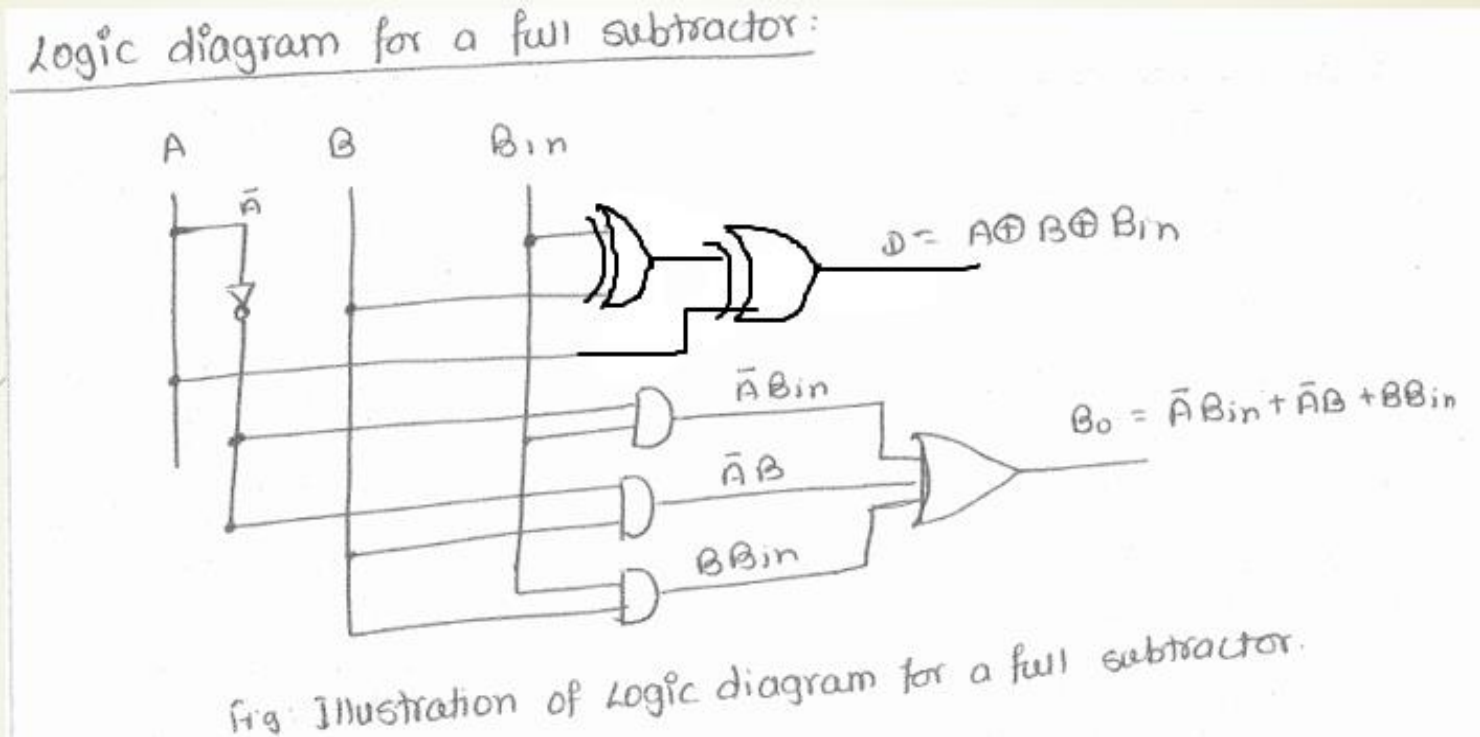



Fig: Illustration of Logic diagram for a full subtractor.

# BINARY PARALLEL ADDER

- A full adder is capable of adding only two single digit binary numbers along with a carry input.
- But, in practice, we need to add binary numbers which are much longer than just one bit.
- To add two n- bit binary numbers, we need to use the n-bit parallel adder.
- It makes use of a number of full adders in cascade.
- Th[illegible]ıs full adder is connected to the carry input of [illegible]

| | | | | |
|---|---|---|---|---|
| Input Carry | 0 | 1 | 1 | 0 | Ci |
| Augend | 1 | 0 | 1 | 1 | Ai |
| Addend | 0 | 0 | 1 | 1 | Bi |
| Sum | 1 | 1 | 1 | 0 | Si |
| output Carry | 0 | 0 | 1 | 1 | Ci+1 |

## A 4-bit Binary Parallel Adder:

# BINARY PARALLEL ADDER

- A binary parallel adder is a digital function that produces the arithmetic sum of two binary numbers in parallel.
- It consists of full-adder connected in cascade. with the output carry from c rry of the next full-adder.

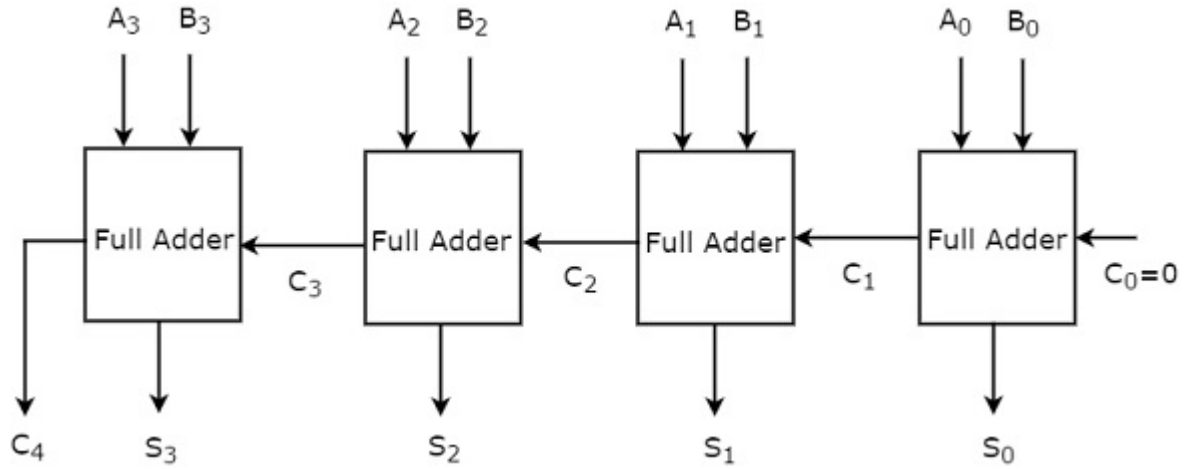
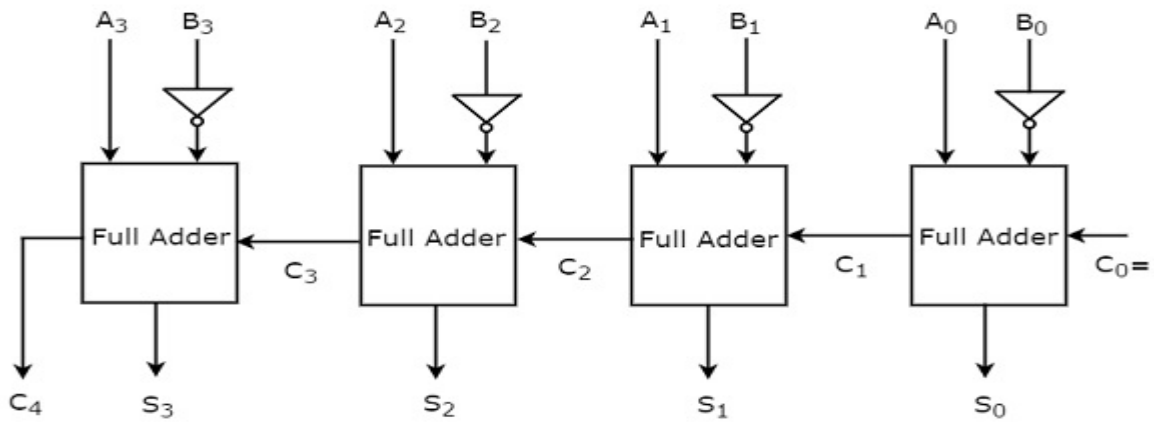



Fig: 4-bit Binary Parallel adder

# 4-BIT BINARY PARALLEL SUBTRACTOR

- The 4-bit binary subtractor produces the subtraction of two 4-bit numbers.
- Let the 4 bit binary numbers, $A=A_3A_2A_1A_0$ and $B=B_3B_2B_1B_0$.
- Internally, the operation of 4-bit Binary subtractor is similar to that of 4-bit Binary adder.
- If the normal bits of binary number A, complemented bits of binary number B [illegible] plied to 4-bit Binary adder, ther[illegible]
- The block [illegible]vn in the following figure.




Note: A-B = A+B'+1

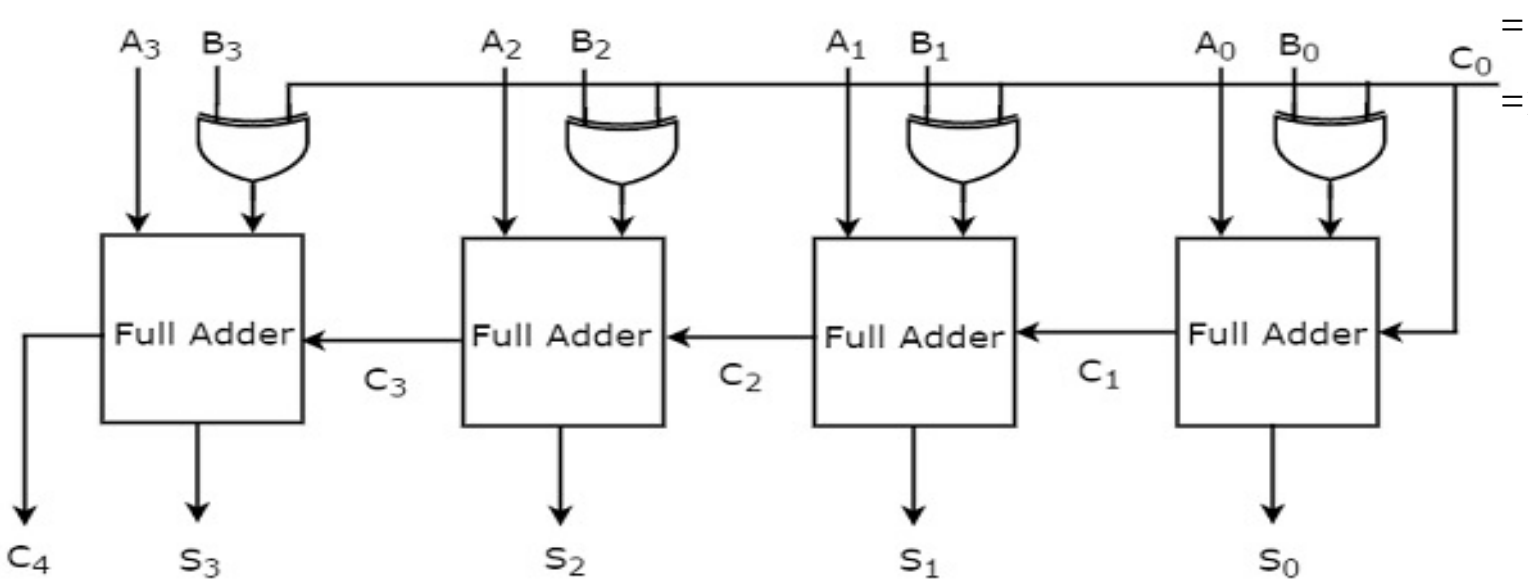
# 4-BIT BINARY PARALLEL SUBTRACTOR

- This 4-bit binary subtractor produces an output, which is having at most 5 bits.
- If Binary number A is greater than Binary number B, then MSB of the output is zero and the remaining bits hold the magnitude of A-B.
- If Binary number A is less than Binary number B, then MSB of the output is one. So, take the 2's complement of output in order to get the magnitude of A-B

# 4-BIT BINARY PARALLEL ADDER/SUBTRACTOR

- The circuit, which can be used to perform either addition or subtraction of two binary numbers at any time is known as Binary Adder / subtractor.
- Both, Binary adder and Binary subtractor contain a set of Full adders, which are cascaded.
- The input bits of binary number A are directly applied in both Binary adder and Binary subtractor.
- The input bits of binary number B are directly applied to Full adders in Binary adder, whereas the complemented bits of binary number B are applied to Full adders in Binary subtractor.
- The initial carry, $C_0 = 0$ is applied in 4-bit Binary adder, whereas the initial carry borrow, $C_0 = 1$ is applied in 4-bit Binary subtractor.
- We know that a 2-input Ex-OR gate produces an output, which is same as that of first input when other input is zero. Similarly, it produces an output, which is complement of first input when other input is one.

# 4-BIT BINARY PARALLEL ADDER/SUBTRACTOR




- If initial carry, $C_0$ is zero, then each full adder gets the normal bits of binary numbers A & B. So, the 4-bit binary adder / subtractor produces an output, which is the addition of two binary numbers A & B.
- If initial borrow, $C_0$ is one, then each full adder gets the normal bits of binary number A & complemented bits of binary number B. So, the 4-bit binary adder / subtractor produces an output, which is the subtraction of two binary numbers A & B.

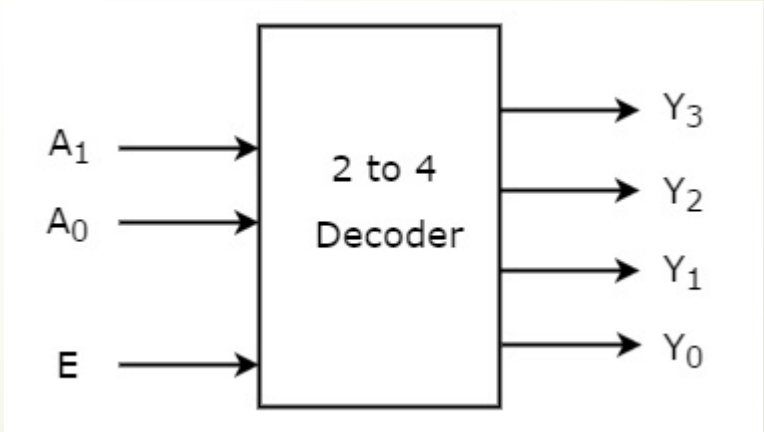
# DECODERS



- Decoder is a combinational circuit that has ‘n’ input lines and maximum of $2^n$ output lines.
- One of these outputs will be active High based on the combination of inputs present, when the decoder is enabled.
- That means decoder detects a particular code. The outputs of the decoder are nothing but the min terms of ‘n’ input variables lines, when it is enabled.

**2 to 4 Decoder**

- Let 2 to 4 Decoder has two inputs A1 & A0 and four outputs Y3, Y2, Y1 & Y0. The block diagram of 2 to 4 decoder is shown in the following figure.



# DECODERS



One of these four outputs will be ‘1’ for each combination of inputs when enable, E is ‘1’. The Truth table of 2 to 4 decoder is shown below.

| Enable | Inputs | | Outputs | | | |
|---|---|---|---|---|---|---|
| E | $A_1$ | $A_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| 0 | x | x | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

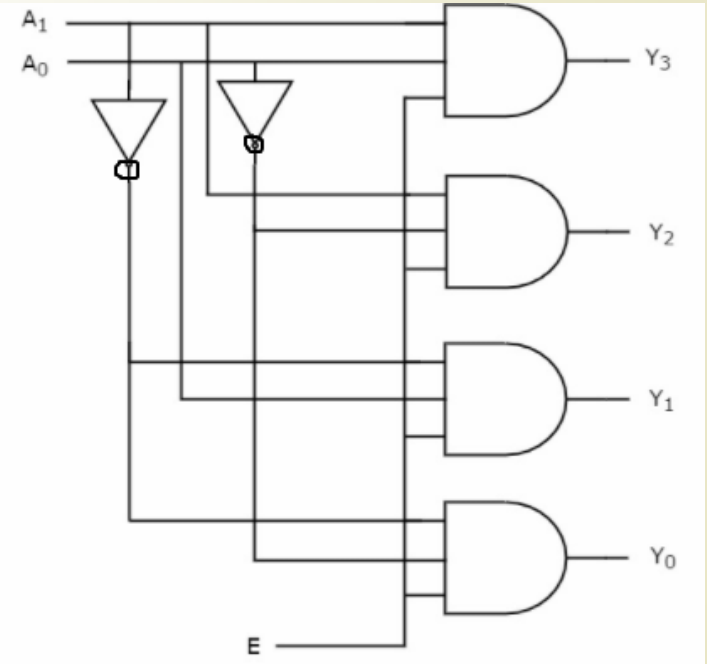
From Truth table, we can write the **Boolean functions** for each output as

$$Y_3 = E.A_1.A_0$$

$$Y_2 = E.A_1.A_0'$$

$$Y_1 = E.A_1'.A_0$$

$$Y_0 = E.A_1'.A_0'$$



# DECODERS

## 3 X 8 DECODER:

### Truth Table

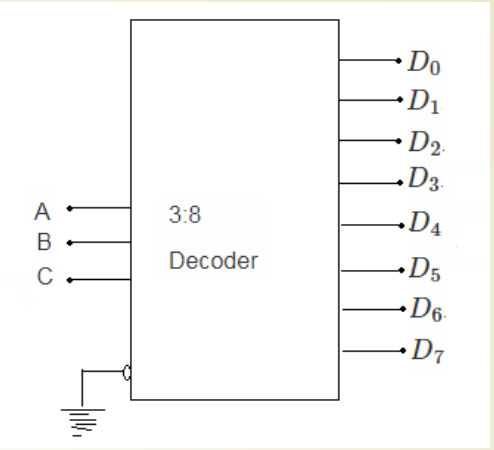
| A | B | C | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### Function

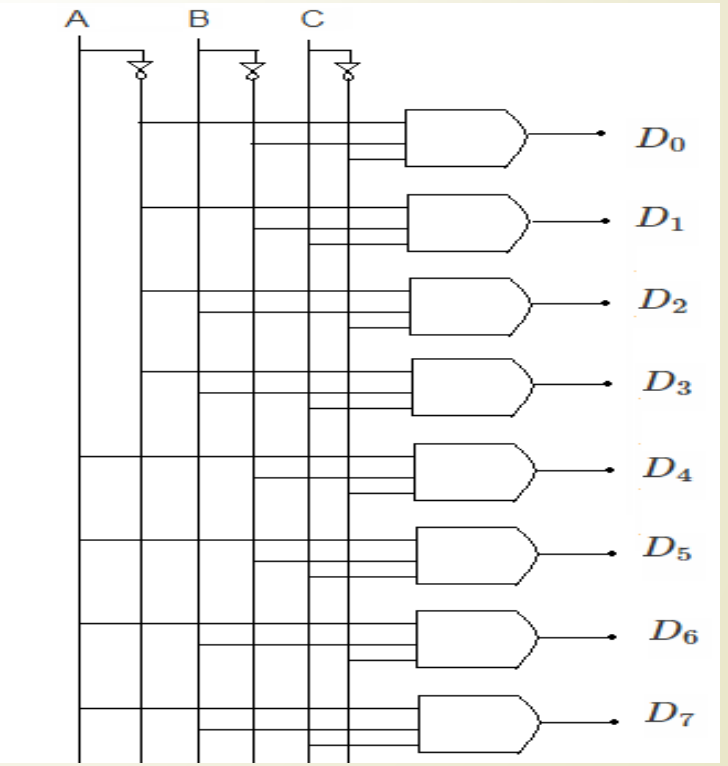
$$D_0 = \bar{A}\bar{B}\bar{C}, \quad D_1 = \bar{A}\bar{B}C, \quad D_2 = \bar{A}B\bar{C},$$

$$D_3 = \bar{A}BC, \quad D_4 = A\bar{B}\bar{C}, \quad D_5 = A\bar{B}C,$$

$$D_6 = AB\bar{C}, \quad D_7 = ABC$$




Block Diagram

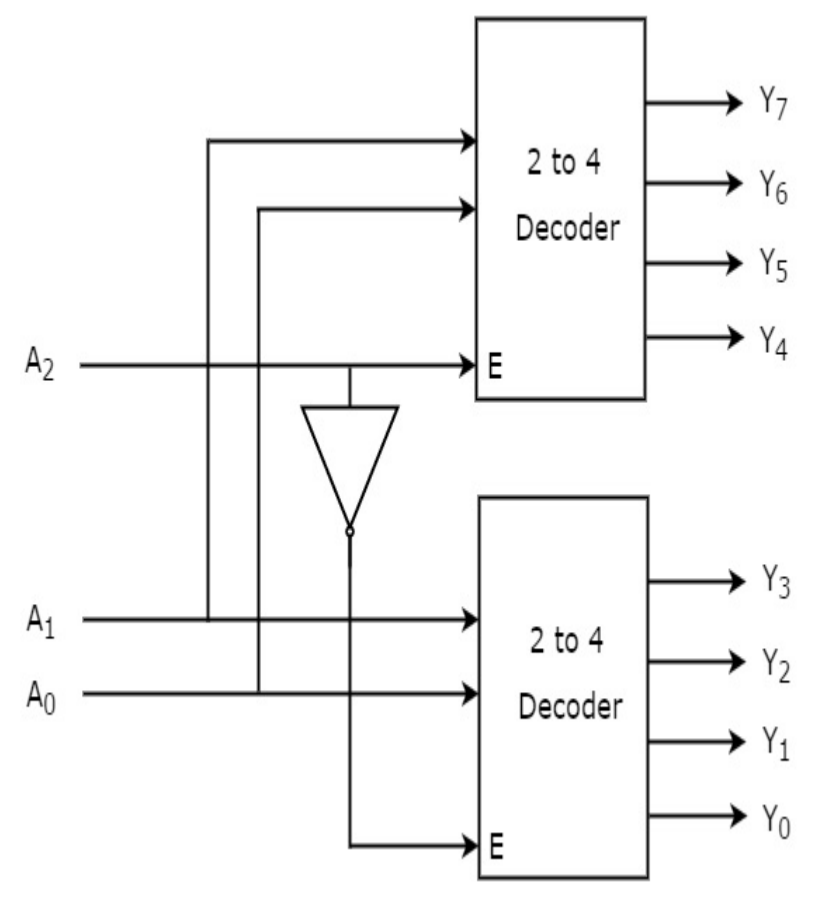



Logic Diagram

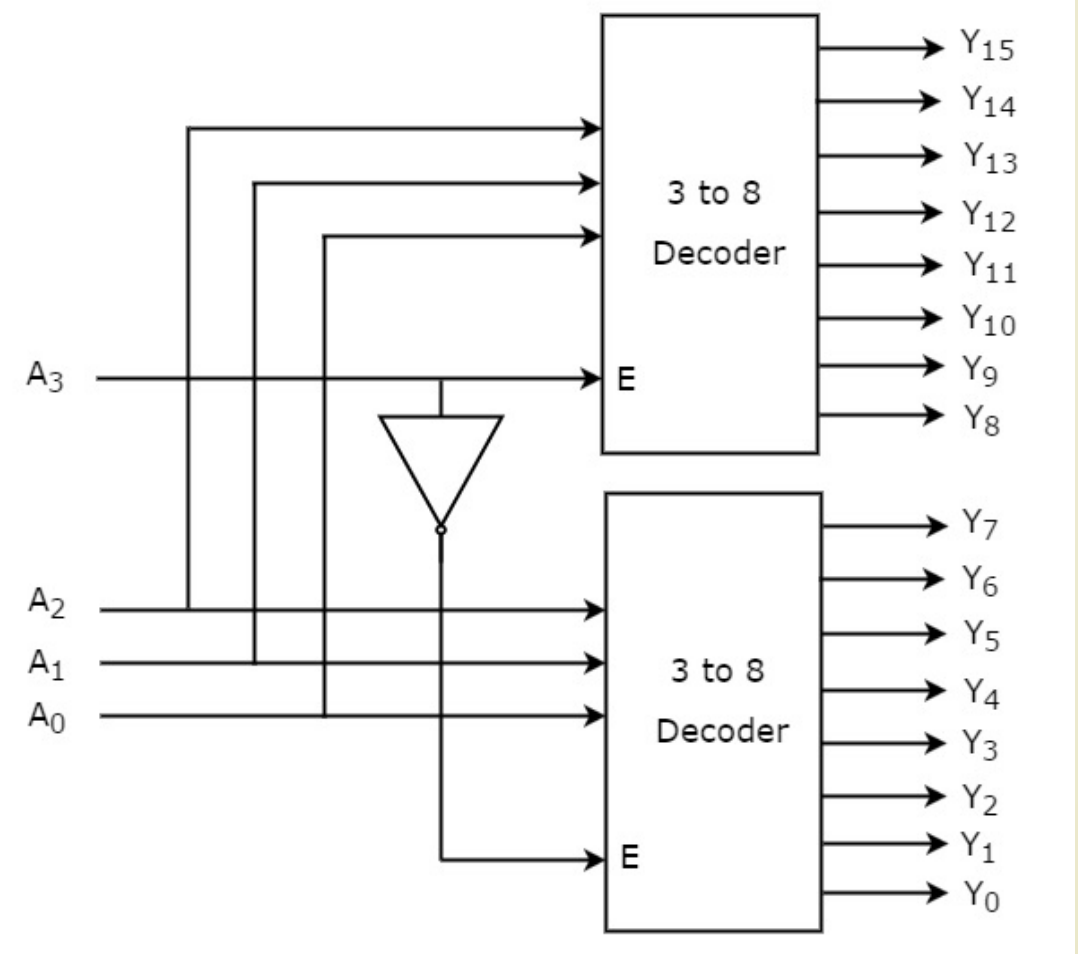
# IMPLEMENTATION OF HIGHER ORDER DECODERS



3 to 8 decoder using 2 to 4 decoder



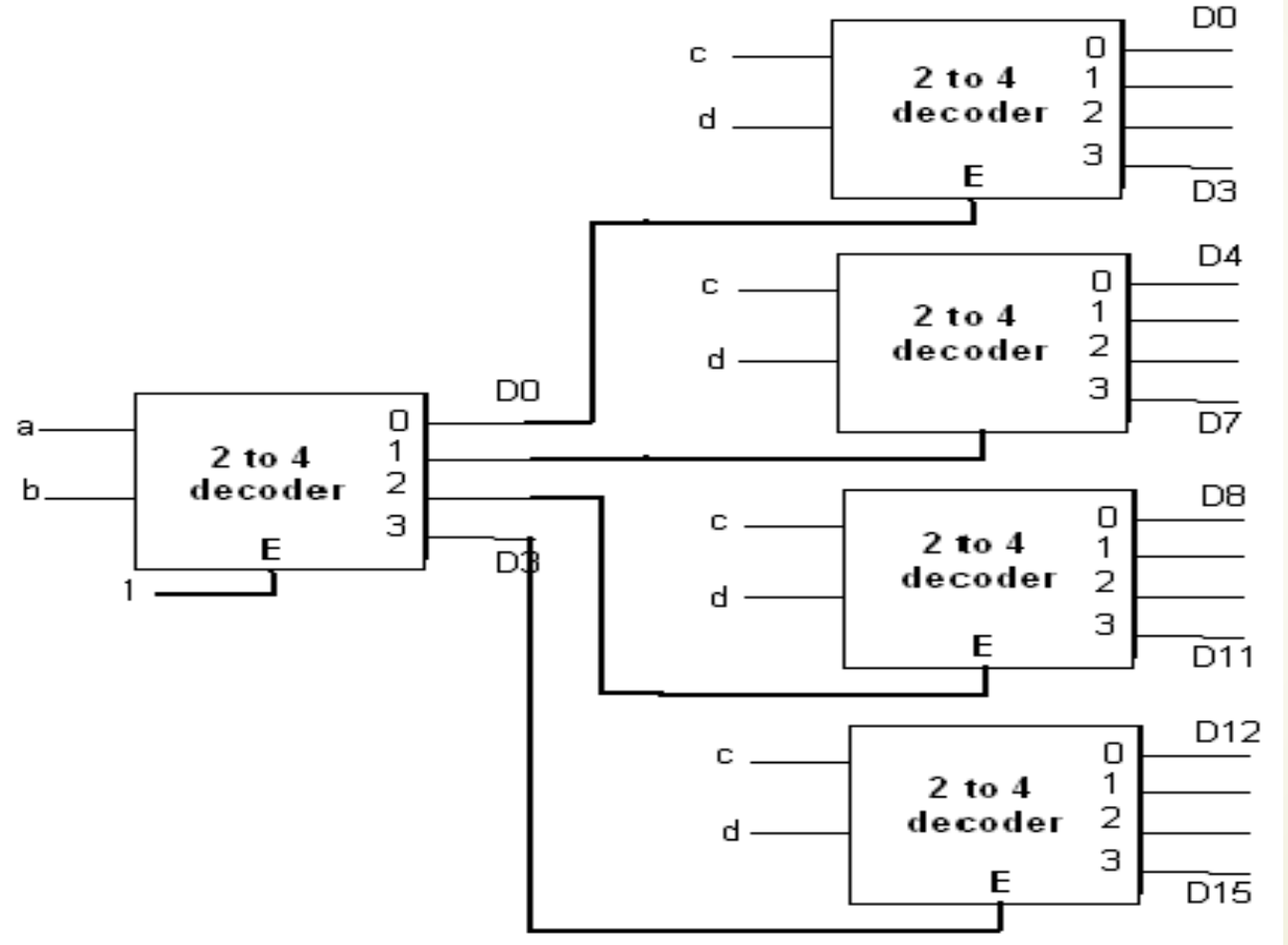

4 to 16 decoder using 3 to 8 decoder



# IMPLEMENTATION OF HIGHER ORDER DECODERS



4 to 16 decoder using 2 to 4 decoder



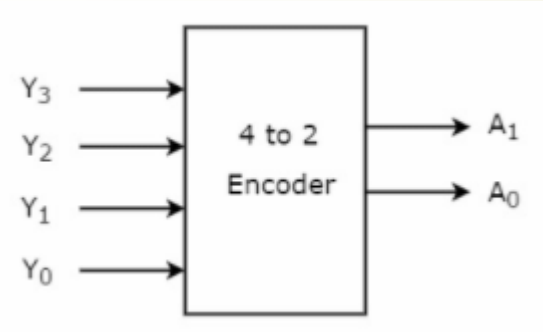
# ENCODERS



- An Encoder is a combinational circuit that performs the reverse operation of Decoder.
- It has maximum of $2^n$ input lines and 'n' output lines.
- It will produce a binary code equivalent to the input, which is active High.
- Therefore, the encoder encodes $2^n$ input lines with 'n' bits. It is optional to represent the enable signal in encoders.

## 4 to 2 Encoder

- Let 4 to 2 Encoder has four inputs $Y_3$, $Y_2$, $Y_1$ & $Y_0$ and two outputs $A_1$ & $A_0$. The block diagram of 4 to 2 Encoder is shown in the following figure.



# ENCODERS



At any time, only one of these 4 inputs can be '1' in order to get the respective binary code at the output.

The Truth table of 4 to 2 encoder is shown below.

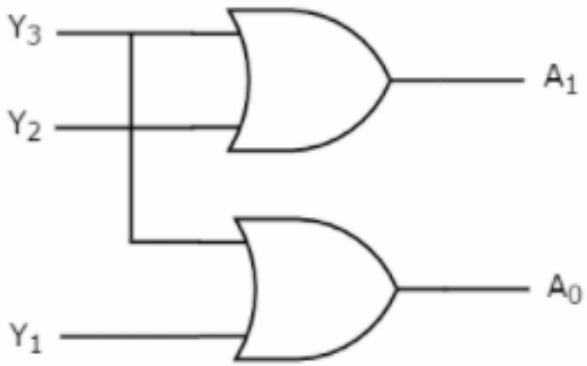
| Inputs | | | | Outputs | |
|---|---|---|---|---|---|
| $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ | $A_1$ | $A_0$ |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |

From Truth table, we can write the **Boolean functions** for each output as

$$A_1 = Y_3 + Y_2$$

$$A_0 = Y_3 + Y_1$$

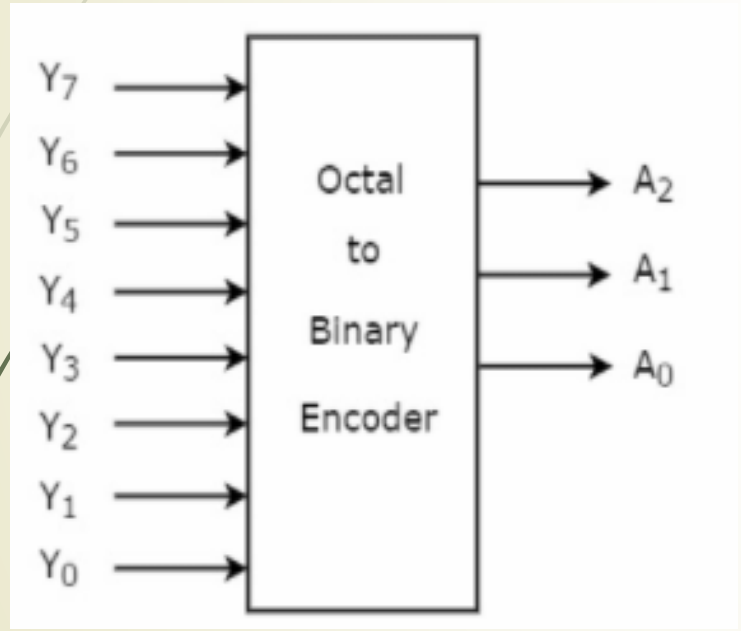
Logic Diagram



# ENCODERS – OCTAL TO BINARY ENCODER



- Octal to binary Encoder has eight inputs, Y7 to Y0 and three outputs A2, A1 & A0.
- Octal to binary encoder is nothing but 8 to 3 encoder. The block diagram of octal to binary Encoder is shown in the following figure




Truth Table and Function

| Inputs | | | | | | | | Outputs | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $Y_7$ | $Y_6$ | $Y_5$ | $Y_4$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ | $A_2$ | $A_1$ | $A_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

From Truth table, we can write the **Boolean functions** for each output as

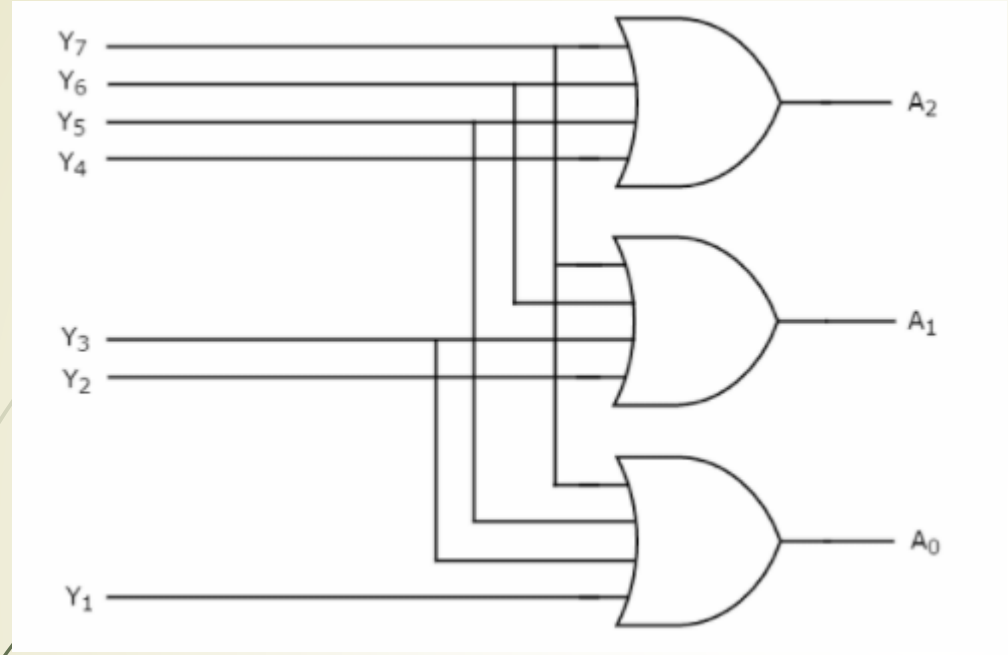
$$A_2 = Y_7 + Y_6 + Y_5 + Y_4$$

$$A_1 = Y_7 + Y_6 + Y_3 + Y_2$$

$$A_0 = Y_7 + Y_5 + Y_3 + Y_1$$

# ENCODERS – OCTAL TO BINARY ENCODER






Logic Diagram

**Drawbacks of Encoder**

- There is an ambiguity, when all outputs of encoder are equal to zero
- If more than one input is active High, then the encoder produces an output, which may not be the correct code.

# ENCODERS – PRIORITY ENCODER



We considered one more output, V in order to know, whether the code available at outputs is valid or not.

- If at least one input of the encoder is ‘1’, then the code available at outputs is a valid one. In this case, the output, V will be equal to 1.
- If all the inputs of encoder are ‘0’, then the code available at outputs is not a valid one. In this case, the output, V will be equal to 0.

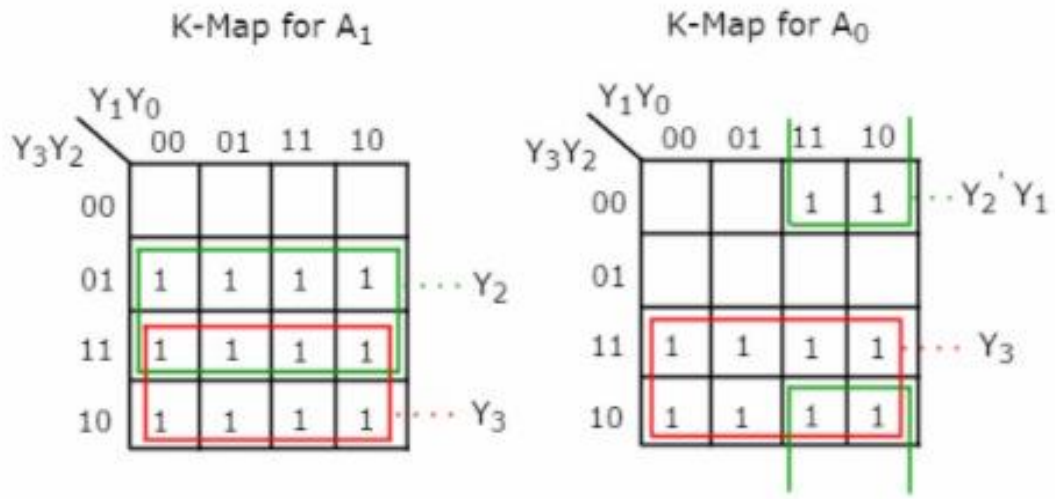
Truth Table

| Inputs | | | | Outputs | | |
|---|---|---|---|---|---|---|
| $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ | $A_1$ | $A_0$ | V |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | x | 0 | 1 | 1 |
| 0 | 1 | x | x | 1 | 0 | 1 |
| 1 | x | x | x | 1 | 1 | 1 |

# ENCODERS – PRIORITY ENCODER



K-MAP and Function




The simplified **Boolean functions** are

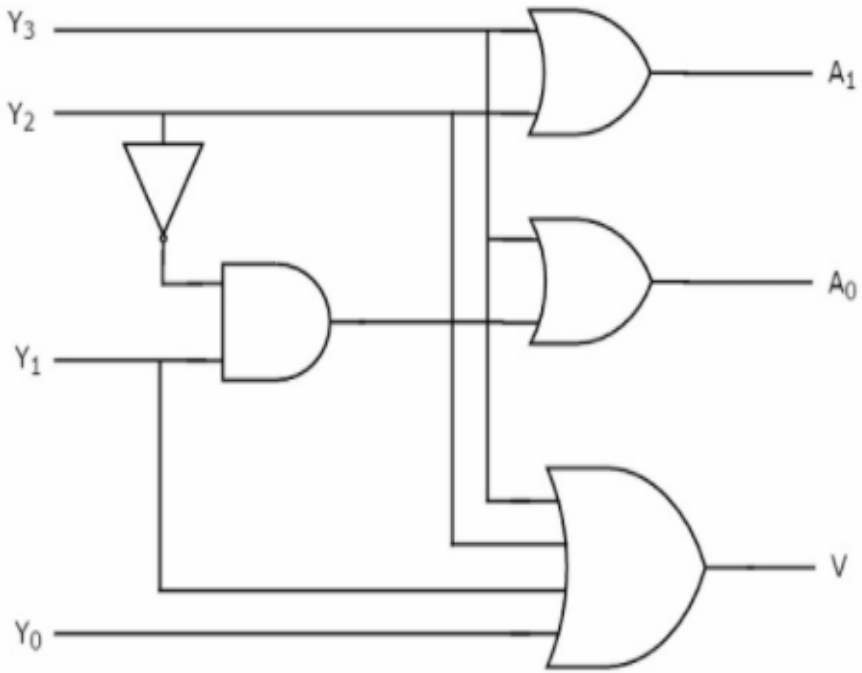
$$A_1 = Y_3 + Y_2$$

$$A_0 = Y_3 + Y_2'Y_1$$

Similarly, we will get the Boolean function of output, V as

$$V = Y_3 + Y_2 + Y_1 + Y_0$$

Logic Diagram



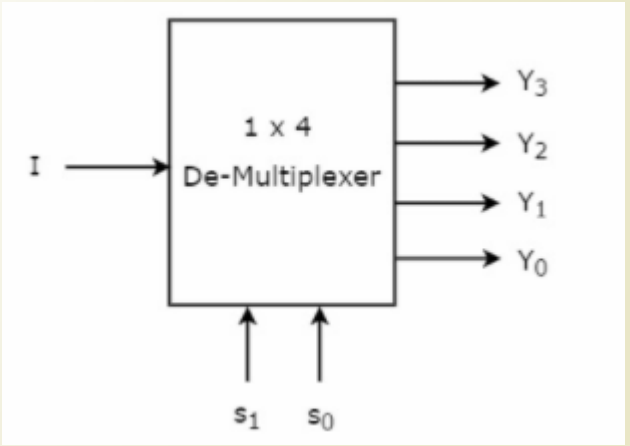
# DEMULTIPLEXERS (DEMUX)



- A demultiplexer (or demux) is a device that takes a single input line and routes it to one of several digital output lines.
- A demultiplexer of $2^n$ outputs has n select lines, which are used to select which output line to send the input.
- A demultiplexer is also called a data distributor.

## 1x4 De-Multiplexer

- 1x4 De-Multiplexer has one input I, two selection lines, s1 & s0 and four outputs Y3, Y2, Y1 &Y0. The block diagram of 1x4 De-Multiplexer is shown in the following figure.



# DEMULTIPLEXERS (DEMUX)



Truth Table

| Selection Inputs | | Outputs | | | |
|---|---|---|---|---|---|
| $S_1$ | $S_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| 0 | 0 | 0 | 0 | 0 | I |
| 0 | 1 | 0 | 0 | I | 0 |
| 1 | 0 | 0 | I | 0 | 0 |
| 1 | 1 | I | 0 | 0 | 0 |

From the above Truth table, we can directly write the **Boolean functions** for each output as

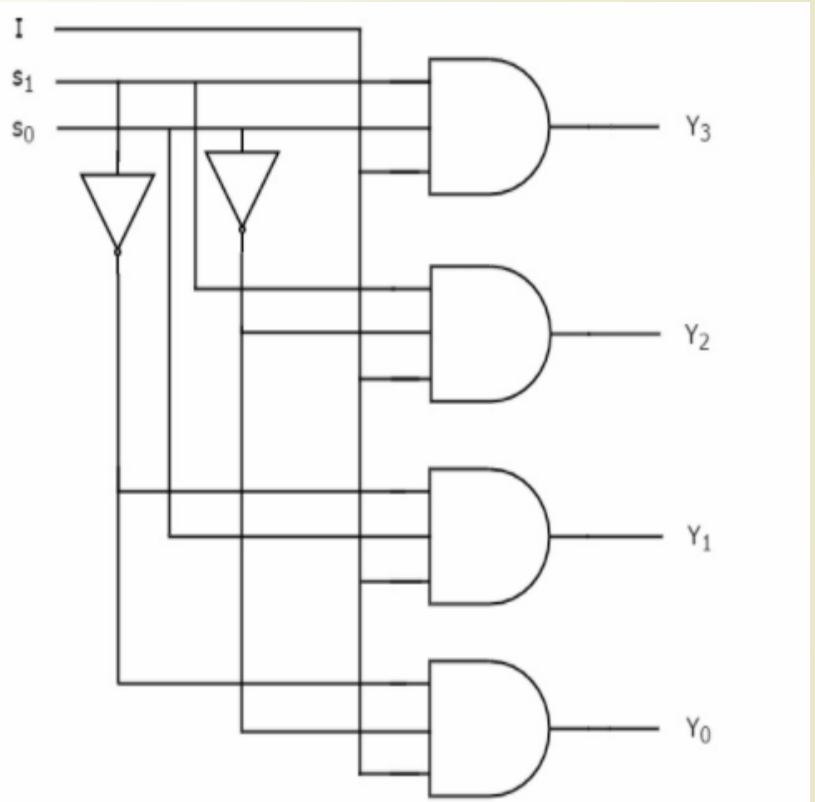
$$Y_3 = s_1 s_0 I$$

$$Y_2 = s_1 s_0' I$$

$$Y_1 = s_1' s_0 I$$

$$Y_0 = s_1' s_0' I$$

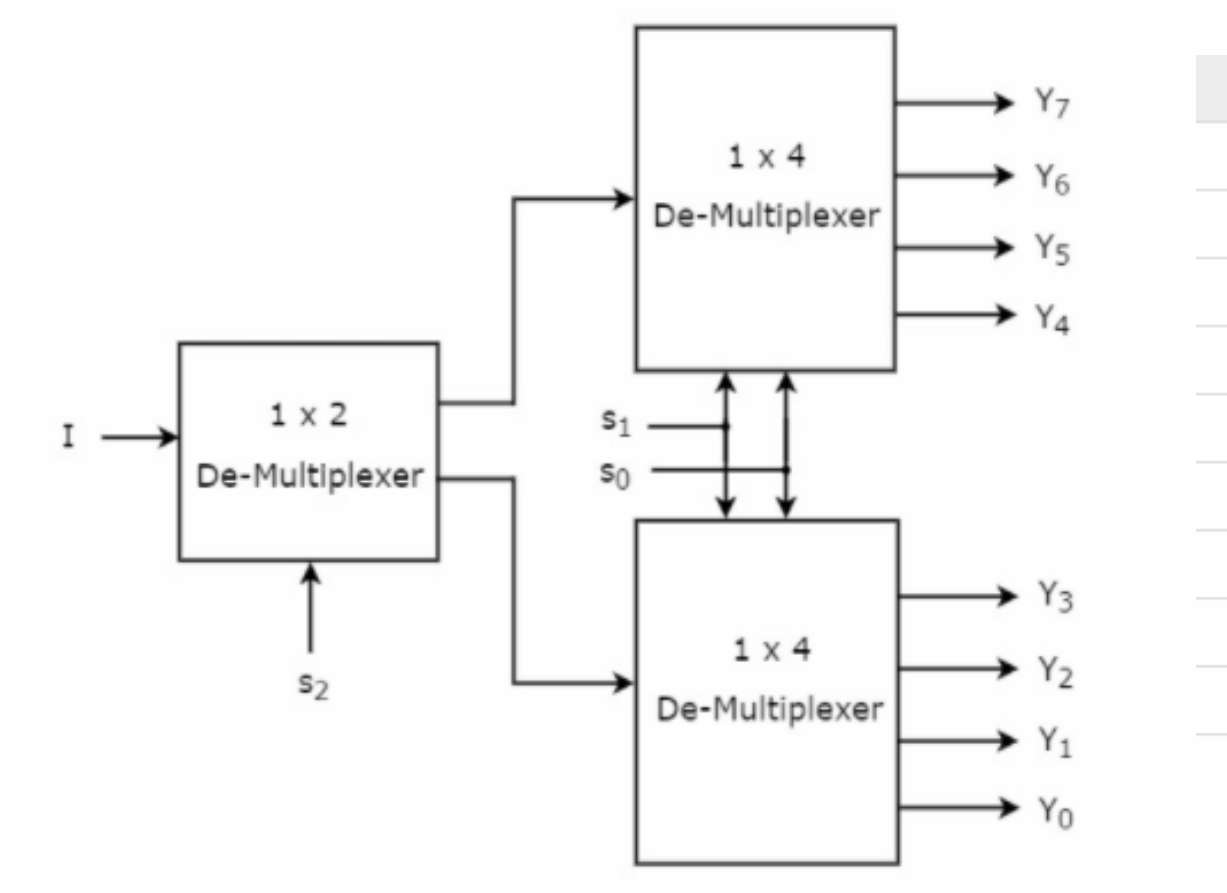
Logic Diagram



# IMPLEMENTATION OF HIGHER ORDER DE-MULTIPLEXERS



**1 X 8 DE-MUX using 1 X 4 and 1 X 2 DE-MUX**




Logic Diagram

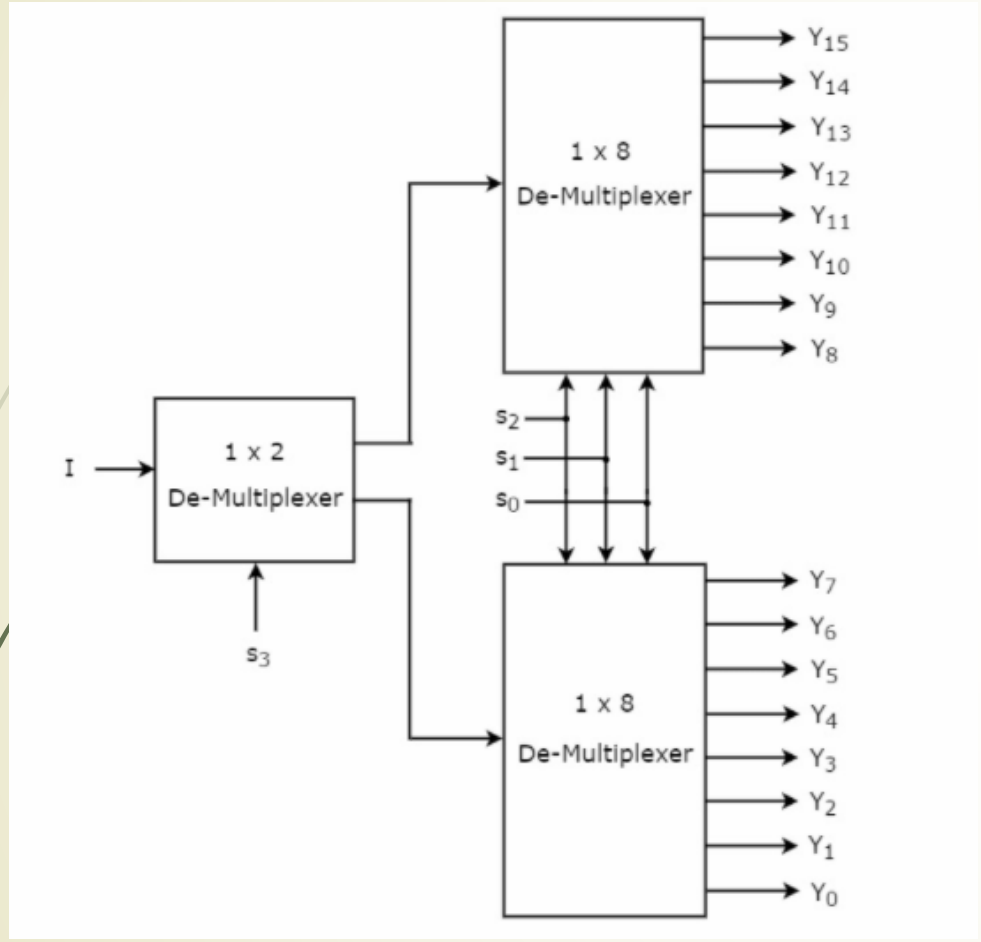
Truth Table

| Selection Inputs | | | Outputs | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $s_2$ | $s_1$ | $s_0$ | $Y_7$ | $Y_6$ | $Y_5$ | $Y_4$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | I |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | I | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | I | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | I | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | I | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | I | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | I | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | I | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# IMPLEMENTATION OF HIGHER ORDER DE-MULTIPLEXERS



## 1 X 16 DE-MUX using 1 x 8 and 1 x 2 DE-MUX




Logic Diagram

## 1 X 16 DE-MUX using 1 x 4

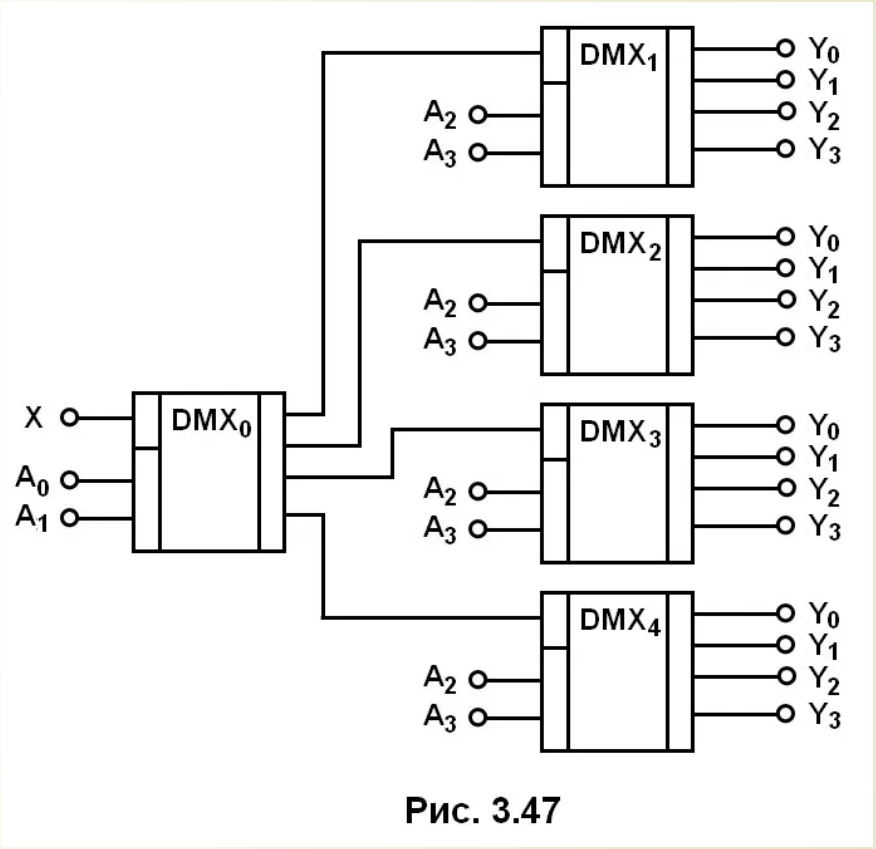



Рис. 3.47

X = INPUT, $A_3$,$A_2$,$A_1$,$A_0$ are selection line

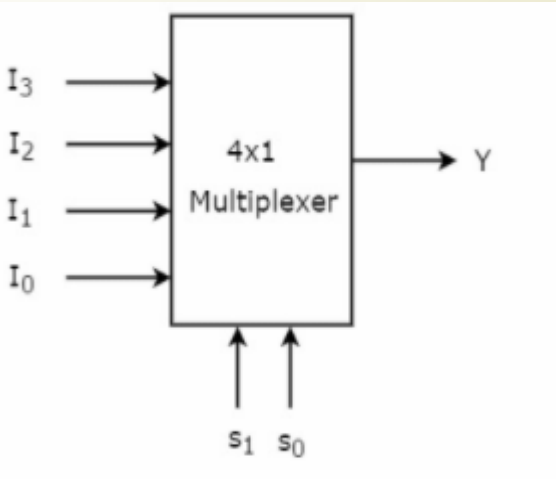
# MULTIPLEXERS (MUX)



- Multiplexer is a combinational circuit that has maximum of $2^n$ data inputs, ‘n’ selection lines and single output line.
- One of these data inputs will be connected to the output based on the values of selection lines.
- Since there are ‘n’ selection lines, there will be $2^n$ possible combinations of zeros and ones.
- So, each combination will select only one data input. Multiplexer is also called as Mux.

## 4x1 Multiplexer

- 4x1 Multiplexer has four data inputs $I_3$, $I_2$, $I_1$ & $I_0$, two selection lines $s_1$ & $s_0$ and one output Y. The block diagram of 4x1 Multiplexer is shown in the following figure.



# MULTIPLEXERS (MUX)



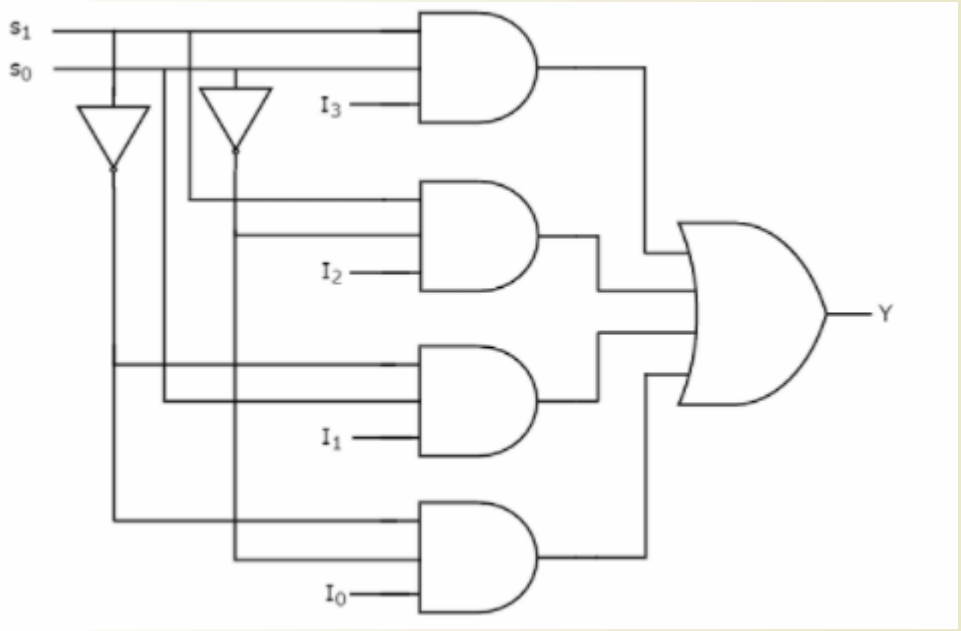
**Truth table** of 4x1 Multiplexer is shown below.

| Selection Lines | | Output |
|---|---|---|
| $S_1$ | $S_0$ | Y |
| 0 | 0 | $I_0$ |
| 0 | 1 | $I_1$ |
| 1 | 0 | $I_2$ |
| 1 | 1 | $I_3$ |

From Truth table, we can directly write the **Boolean function** for output, Y as

$$Y = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3$$

Logic Diagram



# MULTIPLEXERS (MUX)



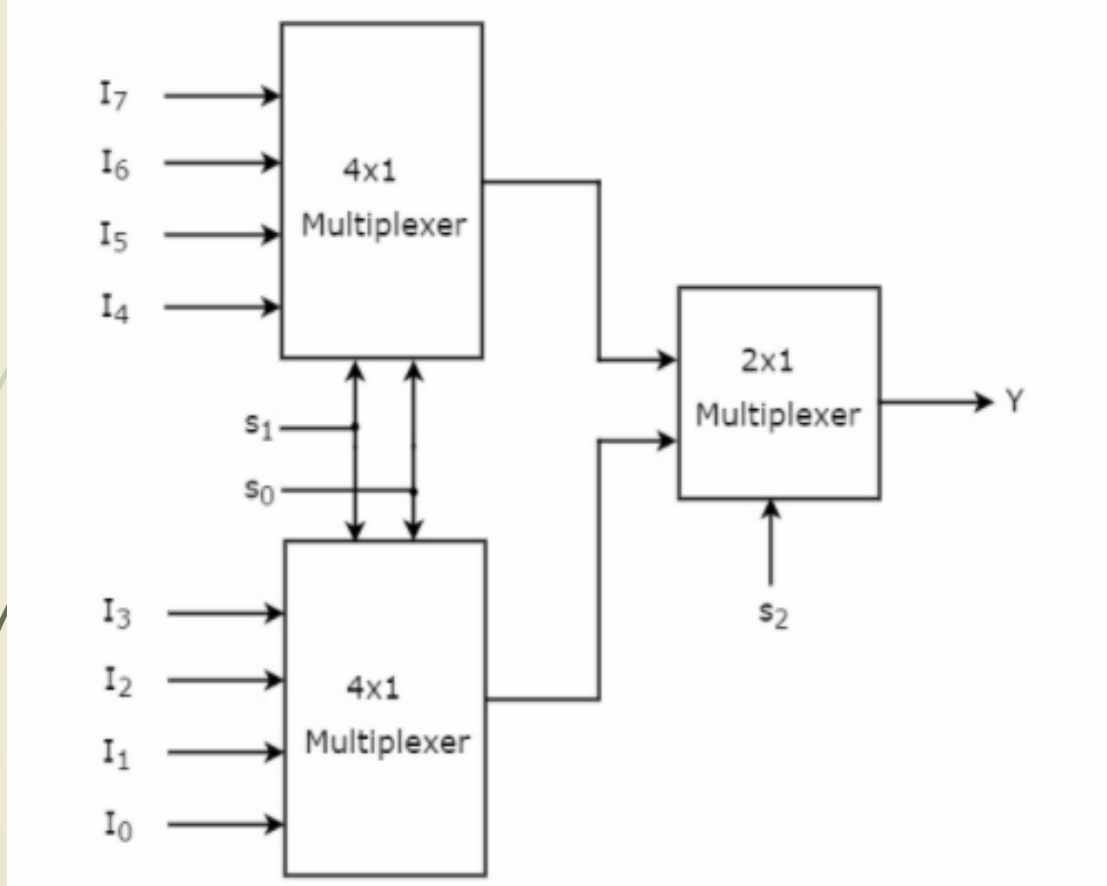
## Application

- Communication system
- Telephone network
- Computer memory
- Transmission from the computer system of a satellite
- Data acquisition system

# IMPLEMENTATION OF HIGHER ORDER MULTIPLEXERS



**8 X 1 MUX using 4 x 1 and 2 x 1 MUX**



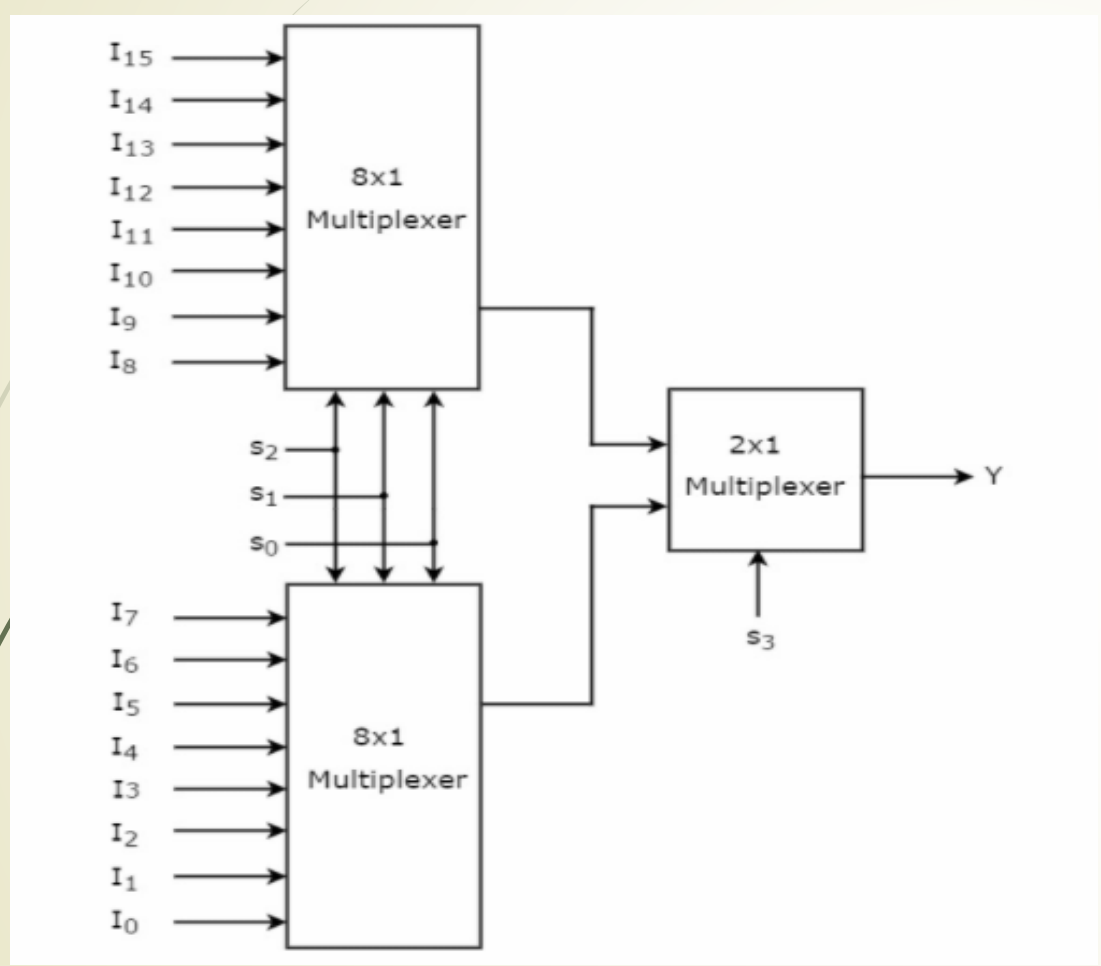

Truth Table

| Selection Inputs | | | Output |
|---|---|---|---|
| $S_2$ | $S_1$ | $S_0$ | Y |
| 0 | 0 | 0 | $I_0$ |
| 0 | 0 | 1 | $I_1$ |
| 0 | 1 | 0 | $I_2$ |
| 0 | 1 | 1 | $I_3$ |
| 1 | 0 | 0 | $I_4$ |
| 1 | 0 | 1 | $I_5$ |
| 1 | 1 | 0 | $I_6$ |
| 1 | 1 | 1 | $I_7$ |

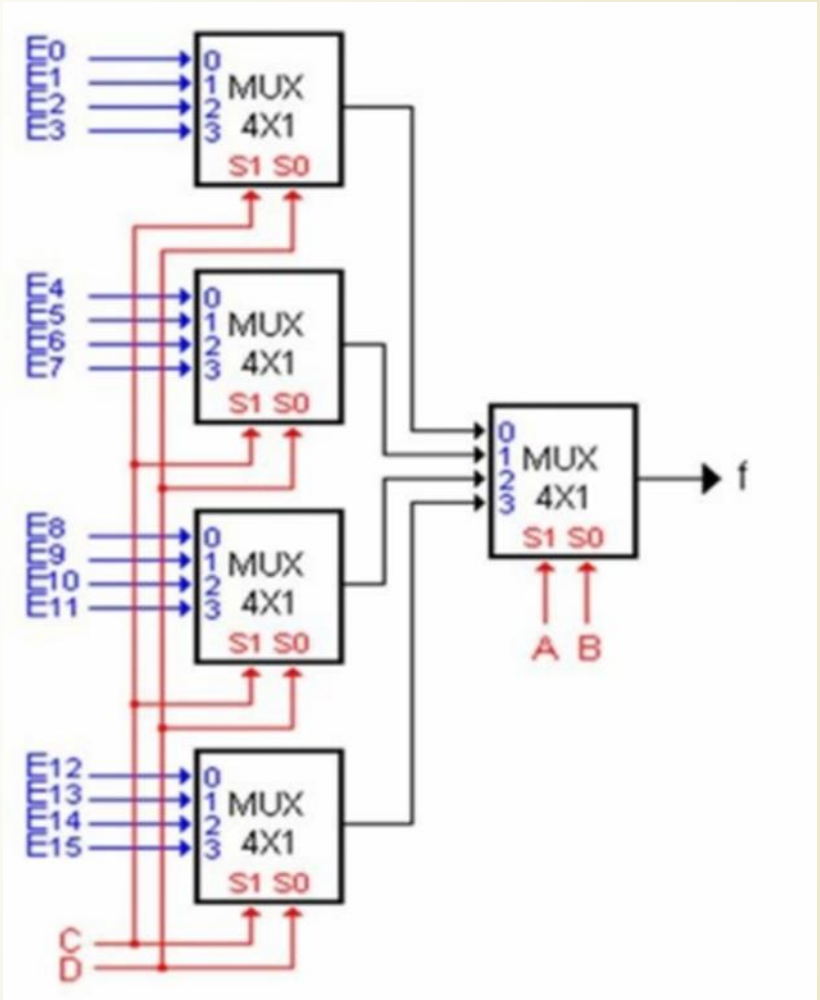
# IMPLEMENTATION OF HIGHER ORDER MULTIPLEXERS



**16 X 1 MUX using 8 x 1 and 2 x 1 MUX**




**16 X 1 MUX using 4 x 1**



# BINARY ADDITION



## Binary Addition Rules

| | | Carry Over | Result |
|---|---|---|---|
| 1. | 0 + 0 | 0 | 0 |
| 2. | 0 + 1 | 0 | 1 |
| 3. | 1 + 0 | 0 | 1 |
| 4. | 1 + 1 | 1 | 0 |
| 5. | 1 + 1 + 1 | 1 | 1 |

# BINARY SUBTRACTION

## Rules of Binary Subtraction

$\Rightarrow 1 - 0 = 1$

$\Rightarrow 1 - 1 = 0$

$\Rightarrow 0 - 0 = 0$

$\Rightarrow 0 - 1 = 1$

(This can not be done directly, hence we borrow one digit from the digit to the left or the next higher order digit.)

# SIGNED AND UNSIGNED BINARY NUMBERS



## Unsigned Numbers

As we already know, the unsigned numbers don't have any sign for representing negat numbers. So the unsigned numbers are always positive. By default, the decimal numb representation is positive. We always assume a positive sign in front of each decimal digit.

## Signed Numbers

The signed numbers have a sign bit so that it can differentiate positive and negative integer numbers. The signed binary number technique has both the sign bit and the magnitude of the number. For representing the negative decimal number, the corresponding symbol in front of the binary number will be added.

# SIGNED AND UNSIGNED BINARY NUMBERS



1. **Sign-Magnitude form**

   In this form, a binary number has a bit for a sign symbol. If this bit is set to 1, the number will be negative else the number will be positive if it is set to 0. Apart from this sign-bit, the n-1 bits represent the magnitude of the number.

2. **1's Complement**

   By inverting each bit of a number, we can obtain the 1's complement of a number. The negative numbers can be represented in the form of 1's complement. In this form, the binary number also has an extra bit for sign representation as a sign-magnitude form.

3. **2's Complement**

   By inverting each bit of a number and adding plus 1 to its least significant bit, we can obtain the 2's complement of a number. The negative numbers can also be represented in the form of 2's complement. In this form, the binary number also has an extra bit for sign representation as a sign-magnitude form.

# Combinational and Arithmetic Circuits – SET A

1. Which of the following is correct for multiplexer?
a) Several inputs and single output
b) Single input and several outputs
c) Single input and single output
d) Several inputs and several outputs

2. Multiplexers work with _____
a) Analog signal
b) Digital signal
c) Both analog and digital signal
d) None of the mentioned

3. TDM stands for _____
a) Time direct measurement
b) Time division multiplexing
c) Time direct multiplexing
d) Time division measurement

4. Which of the following is analogous to multiplexer?
a) Data selector
b) Data multiplexer
c) Data filter
d) None of the mentioned

5. Which of the following represent multiple input single output switch?
a) Multiplexer
b) De multiplexer
c) Both multiplexer and demultiplexer
d) None of the mentioned

6. Schematic symbol of multiplexer is _____
a) Isosceles triangle
b) Isosceles trapezoid
c) Equilateral triangle
d) Rectangle

# Combinational and Arithmetic Circuits – SET A

7. In digital multiplexer selector line is _____
a) Analog value
b) Digital value
c) Unpredictable
d) None of the mentioned

8. Which of the following is not a multiplexer?
a) 8-to-1 line
b) 16-to-1 line
c) 4-to-1 line
d) 1-to-4 line

# Combinational and Arithmetic Circuits – SET B

1. What is a multiplexer?
a) It is a type of decoder which decodes several inputs and gives one output
b) A multiplexer is a device which converts many signals into one
c) It takes one input and results into many output
d) It is a type of encoder which decodes several inputs and gives one output

2. Which combinational circuit is renowned for selecting a single input from multiple inputs & directing the binary information to output line?
a) Data Selector
b) Data distributor
c) Both data selector and data distributor
d) DeMultiplexer

3. The enable input is also known as _____
a) Select input
b) Decoded input
c) Strobe
d) Sink

4. Which is the major functioning responsibility of the multiplexing combinational circuit?
a) Decoding the binary information
b) Generation of all minterms in an output function with OR-gate
c) Generation of selected path between multiple sources and a single destination
d) Encoding of binary information

5. What is the function of an enable input on a multiplexer chip?
a) To apply Vcc
b) To connect ground
c) To active the entire chip
d) To active one half of the chip

6. One multiplexer can take the place of _____
a) Several SSI logic gates
b) Combinational logic circuits
c) Several Ex-NOR gates
d) Several SSI logic gates or combinational logic circuits

# Combinational and Arithmetic Circuits – SET B

7. A digital multiplexer is a combinational circuit that selects _____
a) One digital information from several sources and transmits the selected one
b) Many digital information and convert them into one
c) Many decimal inputs and transmits the selected information
d) Many decimal outputs and accepts the selected information

8. In a multiplexer, the selection of a particular input line is controlled by _____
a) Data controller
b) Selected lines
c) Logic gates
d) Both data controller and selected lines

9. If the number of n selected input lines is equal to 2^m then it requires ____ select lines.
a) 2
b) m
c) n
d) $2^n$

10. How many select lines would be required for an 8-line-to-1-line multiplexer?
a) 2
b) 4
c) 8
d) 3

12. How many NOT gates are required for the construction of a 4-to-1 multiplexer?
a) 3
b) 4
c) 2
d) 5

# Combinational and Arithmetic Circuits – SET C

3. On subtracting (01010)2 from (11110)2 using 1's complement, we get _____
a) 01001
b) 11010
c) 10101
d) 10100

4. On subtracting (010110)2 from (1011001)2 using 2's complement, we get _____
a) 0111001
b) 1100101
c) 0110110
d) 1000011

5. On subtracting (001100)2 from (101001)2 using 2's complement, we get _____
a) 1101100
b) 011101
c) 11010101
d) 11010111

6. On addition of 28 and 18 using 2's complement, we get _____
a) 00101110
b) 0101110
c) 00101111
d) 1001111

7. On addition of +38 and -20 using 2's complement, we get _____
a) 11110001
b) 100001110
c) 010010
d) 110101011

8. On addition of -46 and +28 using 2's complement, we get _____
a) -10010
b) -00101
c) 01011
d) 0100101

# Combinational and Arithmetic Circuits – SET D

1. A decoder converts n inputs to _____ outputs
a) n
b) $n^2$
c) $2^n$
d) $n^n$

2. Which of the following are building blocks of encoders
a) NOT gate
b) OR gate
c) AND gate
d) NAND gate

3. Which of the following can be represented for decoder?
a) Sequential circuit
b) Combinational circuit
c) Logical circuit
d) None of the mentioned

4. BCD to seven segment conversion is a _____
a) Decoding process
b) Encoding process
c) Comparing process
d) None of the mentioned

7. Invalid BCD can be made to valid BCD by adding with _____
a) 0101
b) 0110
c) 0111
d) 1001

8. Decoder is constructed from _____
a) Inverters
b) AND gates
c) Inverters and AND gates
d) None of the mentioned

# Combinational and Arithmetic Circuits – SET D

9. Which of the following represents a number of output lines for a decoder with 4 input lines?

a) 15
b) 16
c) 17
d) 18

**7). Which of the following is the output "A,B" for an encoder with 4bits "Y1,Y2,Y3,Y4" as " 0010"?**

- 00
- 01
- 10
- 11

**15). A 4 to 2 encoder requires ___ number of logic gates?**

- 2
- 3
- 4
- 5

**17). Which of the following is the output for 8 to 3 type encoder for input D [7] to I [0] "00000001"?**

- XXX
- 000
- 001
- 010

**26). An encoder generates ___ type code on each input?**

- Hexa
- Binary
- Octal
- ASCII

**83). Which of the following is the output of 3to8 type decoder when input is 1,100?**

- A *B bar * C bar
- A * B* C
- A+B bar +C
- A bar +B +C

**2.3 Sequential logic circuit**: RS Flip-Flops, Gated Flip-Flops, Edge Triggered Flip-Flops, Mater- Slave Flip-Flops. Types of Registers, Applications of Shift Registers, Asynchronous Counters, Synchronous Counters. (AExE0203)

# SEQUENTIAL LOGIC



- Sequential circuit produces an output based on current input and previous input variables.
- That means sequential circuits include memory elements which are capable of storing binary information.
- That binary information defines the state of the sequential circuit at that time.
- A latch is capable of storing one bit of information.
- As shown in figure there are two types of input to the combinational logic :
  - External inputs which not controlled by the circuit.
  - Internal inputs which are a function of a previous output states.

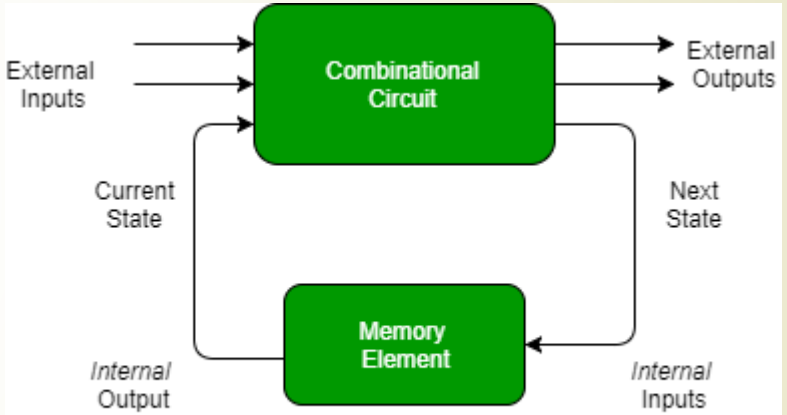



**Figure:** Sequential Circuit

# SEQUENTIAL LOGIC - TYPES



## Asynchronous sequential circuit

- These circuit do not use a clock signal but uses the pulses of the inputs.
- These circuits are faster than synchronous sequential circuits because they change their state immediately when there is a change in the input signal.
- We use asynchronous sequential circuits when speed of operation is important and independent of internal clock pulse.
- But these circuits are more difficult to design and their output is uncertain.

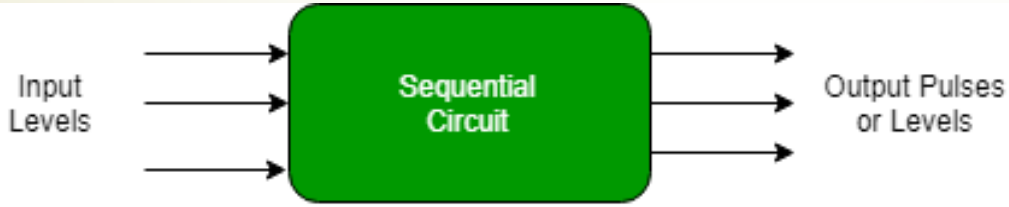



**Figure:** Asynchronous Sequential Circuit

# SEQUENTIAL LOGIC - TYPES



## Synchronous sequential circuit

- These circuit uses clock signal and level inputs (or pulsed) with restrictions on pulse width and circuit propagation.
- The output pulse is the same duration as the clock pulse for the clocked sequential circuits.
- Since they wait for the next clock pulse to arrive to perform the next operation, so these circuits are bit slower compared to asynchronous.
- Level output changes state at the start of an input pulse and remains in that until the next input or clock pulse.
- We use synchronous sequential circuit in synchronous counters,

  flip flops, and in the design of

  state management machines.

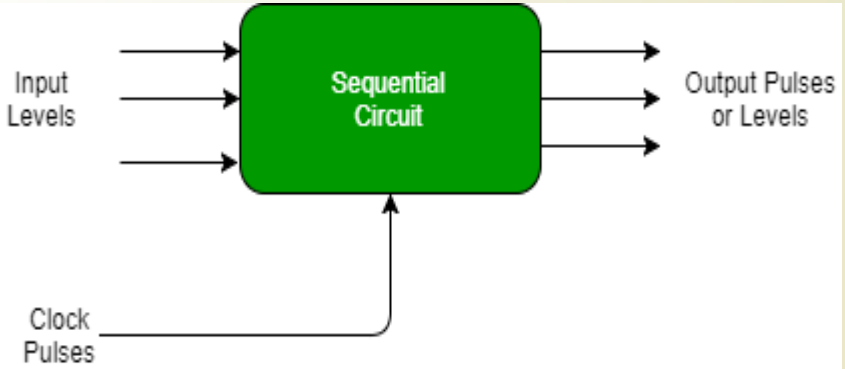


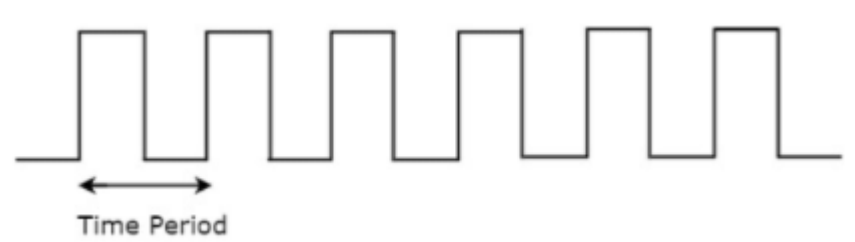

**Figure:** Synchronous Sequential Circuit

# CLOCK SIGNAL AND TRIGGERING

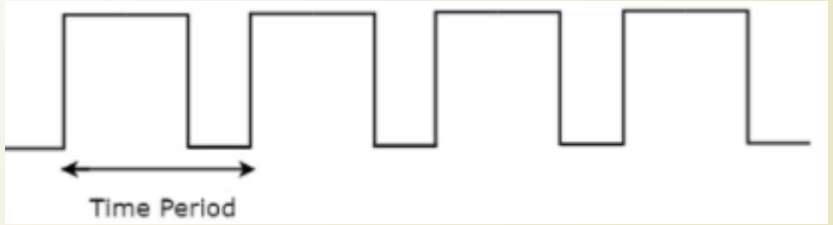


**Clock signal:**

- Clock signal is a periodic signal and its ON time and OFF time need not be the same.
- We can represent the clock signal as a square wave, when both its ON time and OFF time are same.
- The pattern repeats with some time period. In this case, the time period will be equal to either twice of ON time or twice of OFF time.




- We can represent the clock signal as train of pulses, when ON time and OFF time are not same
- In this case, the time period will be equal to sum of ON time and OFF time.



# CLOCK SIGNAL AND TRIGGERING



## Types of Triggering

- Level triggering
- Edge triggering

## Level triggering

- There are two levels, namely logic High and logic Low in clock signal.
- Following are the two types of level triggering.
  - Positive level triggering
  - Negative level triggering

- If the sequential circuit is operated with the clock signal when it is in Logic High, then that type of triggering is known as **Positive level triggering**. It is highlighted in above figure.

# CLOCK SIGNAL AND TRIGGERING



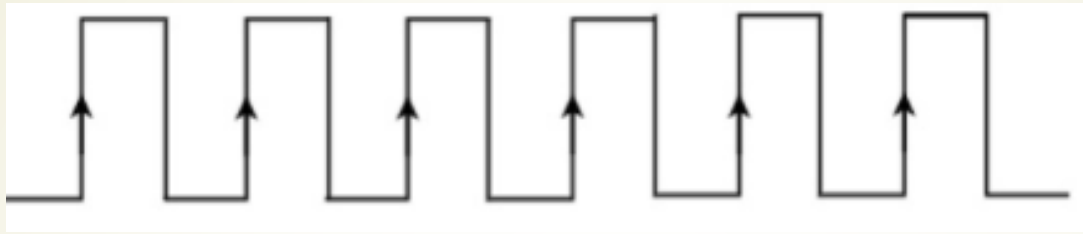
- If the sequential circuit is operated with the clock signal when it is in Logic Low, then that type of triggering is known as **Negative level triggering**. It is highlighted in the following figure.

## Edge triggering

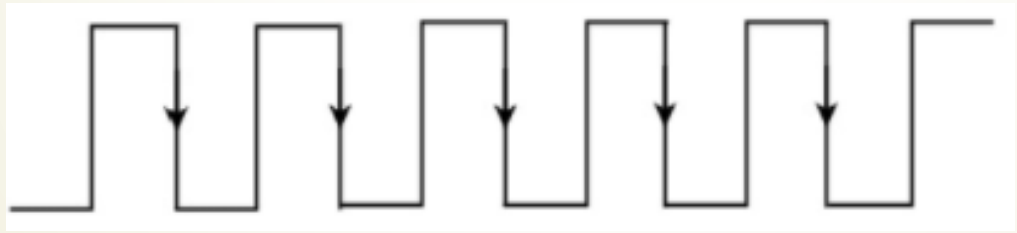
- There are two types of transitions that occur in clock signal. That means, the clock signal transitions either from Logic Low to Logic High or Logic High to Logic Low..
  - Positive edge triggering
  - Negative edge triggering
- If the sequential circuit is operated with the clock signal that is transitioning from Logic Low to Logic High, then that type of triggering is known as **Positive edge triggering**. It is also called as rising edge triggering. It is shown in the following figure.

# CLOCK SIGNAL AND TRIGGERING





- If the sequential circuit is operated with the clock signal that is transitioning from Logic High to Logic Low, then that type of triggering is known as **Negative edge triggering**. It is also called as falling edge triggering. It is shown in the following figure.



# LATCHES

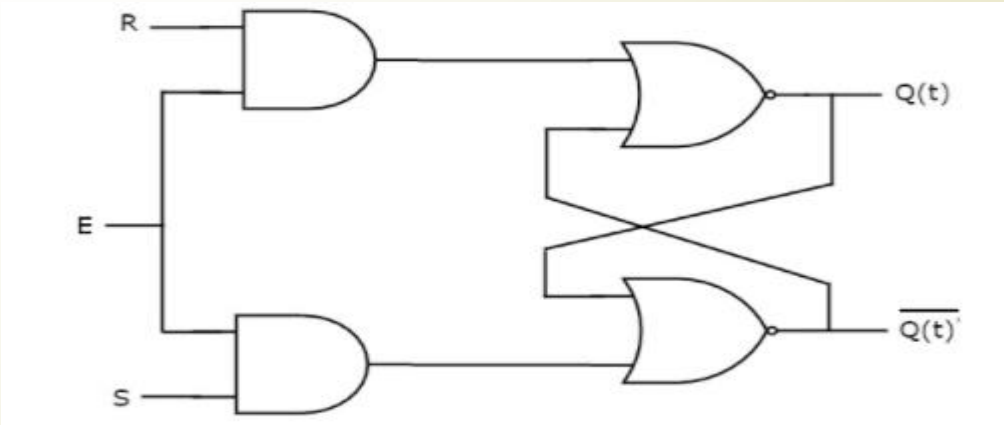


- There are two types of memory elements based on the type of triggering that is suitable to operate it.
  - Latches
  - Flip-flops
- Latches operate with enable signal, which is level sensitive. Whereas, flip-flops are edge sensitive.

## SR Latch

- SR Latch is also called as Set Reset Latch.
- This latch affects the outputs as long as the enable, E is maintained at '1'. The circuit diagram of SR Latch is shown in the following figure

| S | R | Q $t+1$ |
|---|---|---|
| 0 | 0 | Q $t$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | - |



# FLIP-FLOP

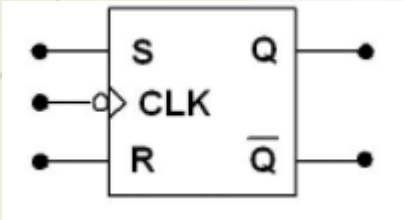


- Flip-flop is a basic digital memory circuit, which stores one bit of information.
- Flip flops are the fundamental blocks of most sequential circuits.
- Flip-flops are used as memory elements in clocked sequential circuit.
- Flip-flop circuit has two outputs, one for the normal value and one for the complement value of the bit stored in it.
- Binary information can be enter a flip-flop in a variety of ways, a fact, which gives rise to different types of flip-flops.
  - SR Flip-Flop
  - D Flip-Flop
  - JK Flip-Flop
  - T Flip-Flop

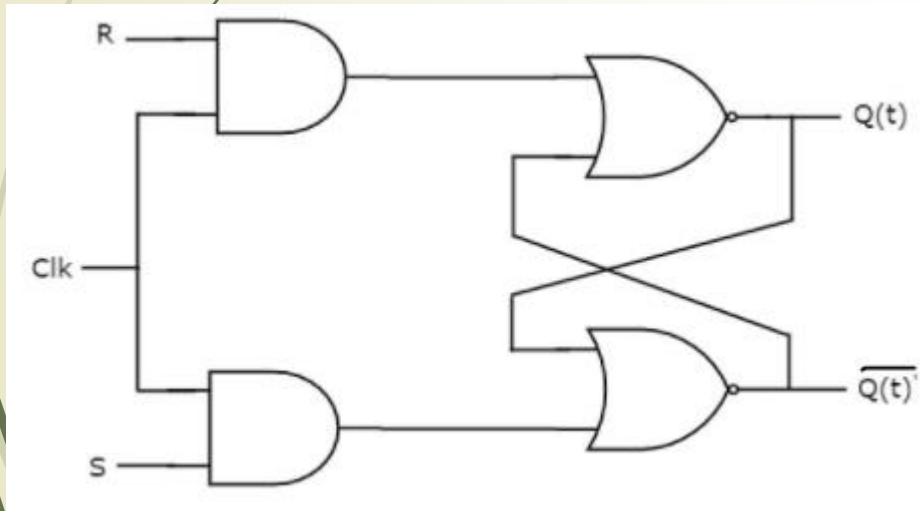
# FLIP-FLOP – SR FLIP-FLOP



SR (Set-Reset) flip-flop operates with only positive clock transitions or negative clock transitions. Whereas, SR latch operates with enable signal. The circuit diagram of SR flip-flop is shown in the following figure.




Graphical Symbol




Logic diagram

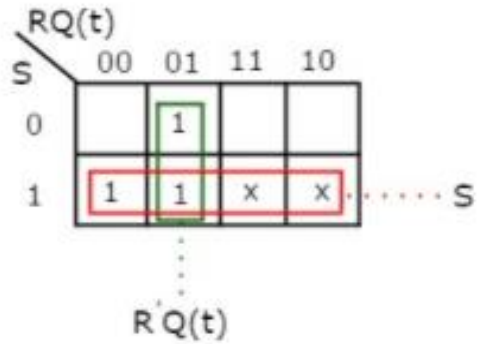
| Clock | S | R | Q(t) | Q(t+1) | Comment |
|---|---|---|---|---|---|
| ↓ | 0 | 0 | 0 | 0 | Hold |
| ↓ | 0 | 0 | 1 | 1 | Hold |
| ↓ | 0 | 1 | 0 | 0 | Reset |
| ↓ | 0 | 1 | 1 | 0 | Reset |
| ↓ | 1 | 0 | 0 | 1 | Set |
| ↓ | 1 | 0 | 1 | 1 | Set |
| ↓ | 1 | 1 | 0 | X | Indeterminant |
| ↓ | 1 | 1 | 1 | X | Indeterminant |

Characteristic Table

# FLIP-FLOP – SR FLIP-FLOP



**K-Map** for next state, Q $t+1$ is shown in the following figure.

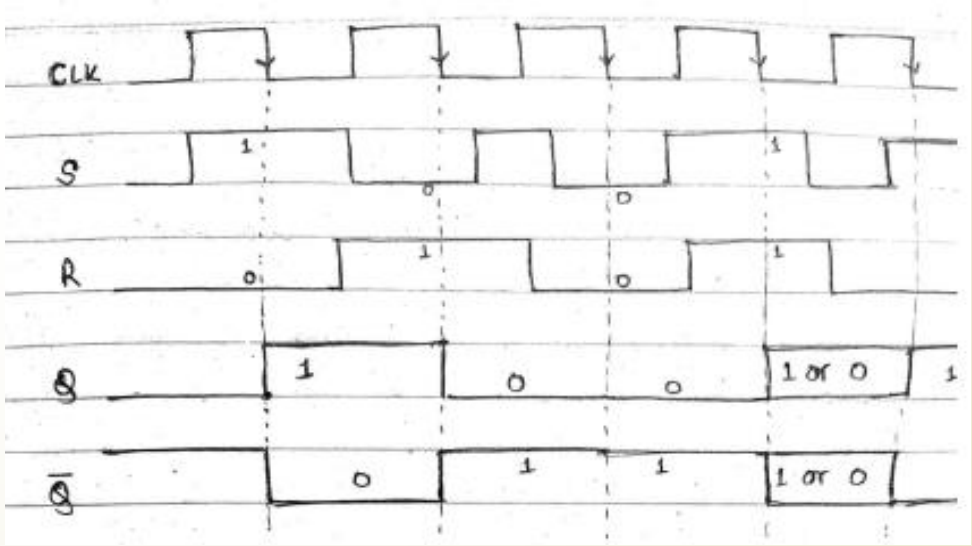



$$Q(t+1) = S + R'Q(t)$$

K-MAP and Boolean function

| SR Flip-flop | | | |
|---|---|---|---|
| Q(t) | Q(t+1) | S | R |
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | X | 0 |

Excitation Table

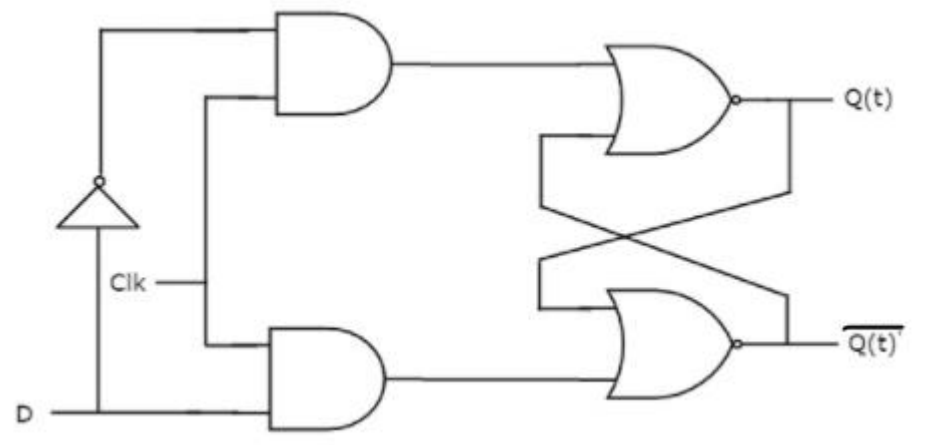



Timing diagram

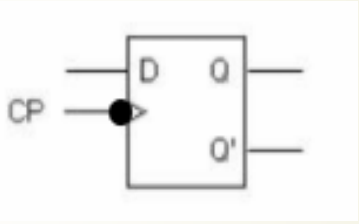
# FLIP-FLOP – D FLIP-FLOP



- D (Data) flip-flop operates with only positive clock transitions or negative clock transitions. Whereas, D latch operates with enable signal.
- That means, the output of D flip-flop is insensitive to the changes in the input, D except for active transition of the clock signal. The circuit diagram of D flip-flop is shown in the following figure.




Logic diagram




Graphical Symbol

| Clock | D | Q(t) | Q(t+1) | Comment |
|---|---|---|---|---|
| ↓ | 0 | 0 | 0 | Reset |
| ↓ | 0 | 1 | 0 | Reset |
| ↓ | 1 | 0 | 1 | Set |
| ↓ | 1 | 1 | 1 | Set |

Characteristic Table

# FLIP-FLOP – D FLIP-FLOP

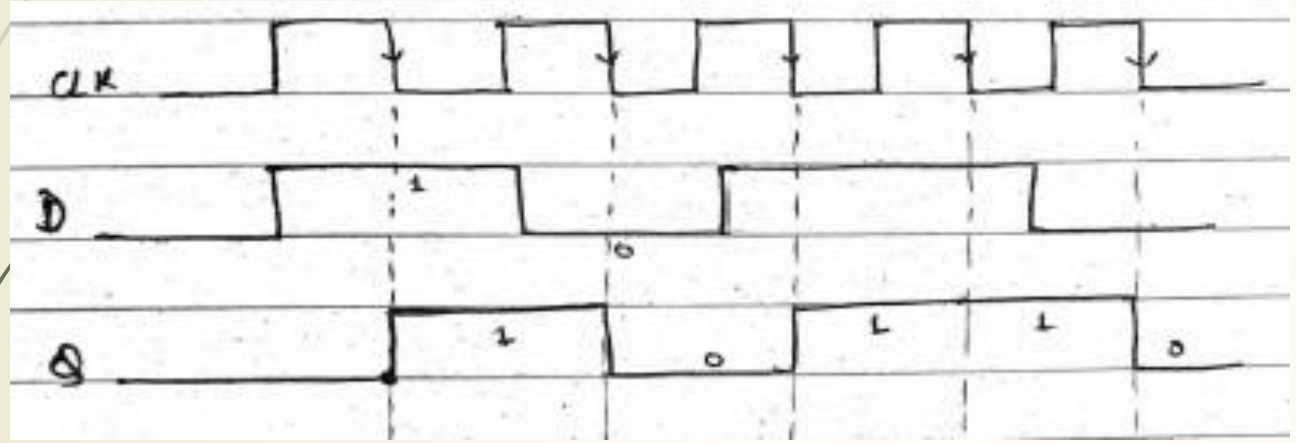


$$Q(t+1) = D$$

Boolean function

| D Flip-flop | | |
|---|---|---|
| Q(t) | Q(t+1) | D |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Excitation Table

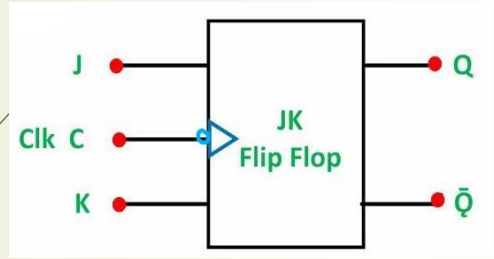



Timing diagram

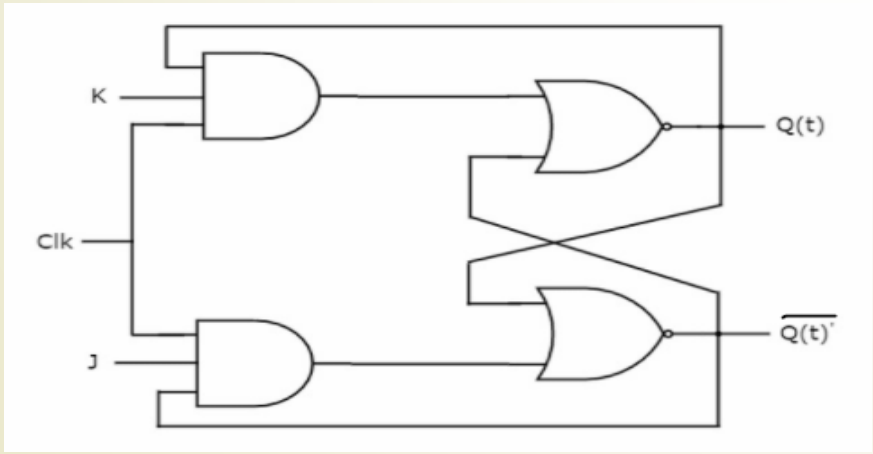
# FLIP-FLOP – JK FLIP-FLOP



- JK flip-flop is the modified version of SR flip-flop.
- It operates with only positive clock transitions or negative clock transitions. The circuit diagram of JK flip-flop is shown in the following figure.
- The indeterminant state of RS flip-flop is defined in JK flip-flop.




Graphical Symbol



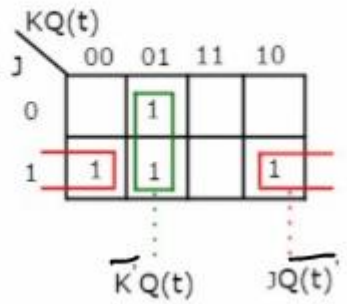

Logic diagram

| Clock | J | K | Q(t) | Q(t+1) | Comment |
|---|---|---|---|---|---|
| ↓ | 0 | 0 | 0 | 0 | Hold |
| ↓ | 0 | 0 | 1 | 1 | Hold |
| ↓ | 0 | 1 | 0 | 0 | Reset |
| ↓ | 0 | 1 | 1 | 0 | Reset |
| ↓ | 1 | 0 | 0 | 1 | Set |
| ↓ | 1 | 0 | 1 | 1 | Set |
| ↓ | 1 | 1 | 0 | 1 | Complement |
| ↓ | 1 | 1 | 1 | 0 | Complement |

Characteristic Table

# FLIP-FLOP – JK FLIP-FLOP

**Three variable K-Map** for next state, $Q(t+1)$ is shown in the following figure.

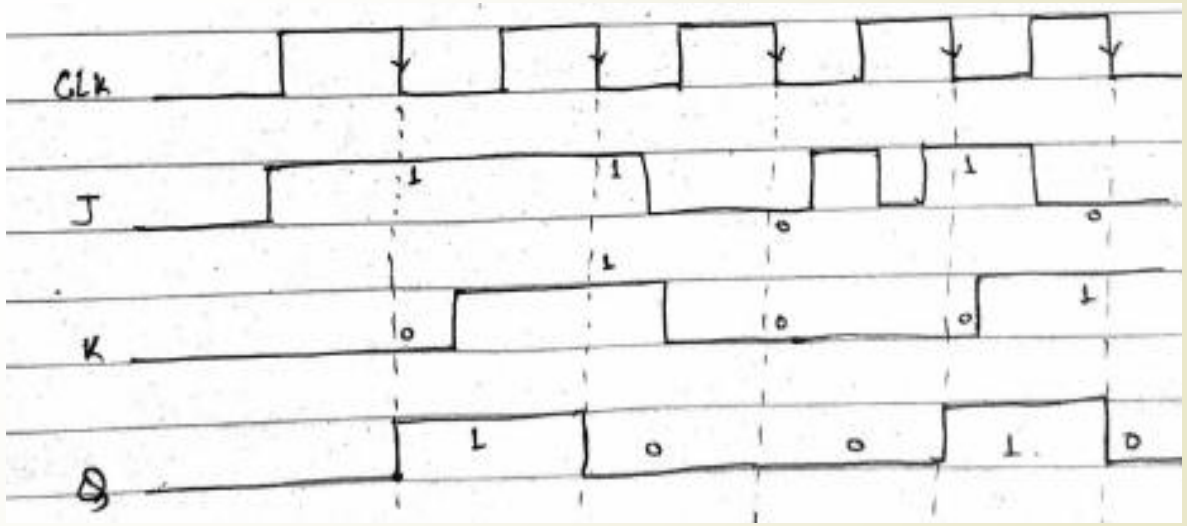



$$Q(t+1) = JQ(t)' + K'Q(t)$$

K-MAP and Boolean function

| JK flip-flop | | | |
|---|---|---|---|
| Q(t) | Q(t+1) | J | K |
| 0 | 0 | 0 | x |
| 0 | 1 | 1 | x |
| 1 | 0 | x | 1 |
| 1 | 1 | x | 0 |

Excitation Table

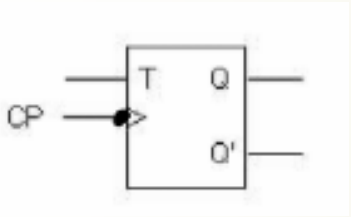



Timing diagram

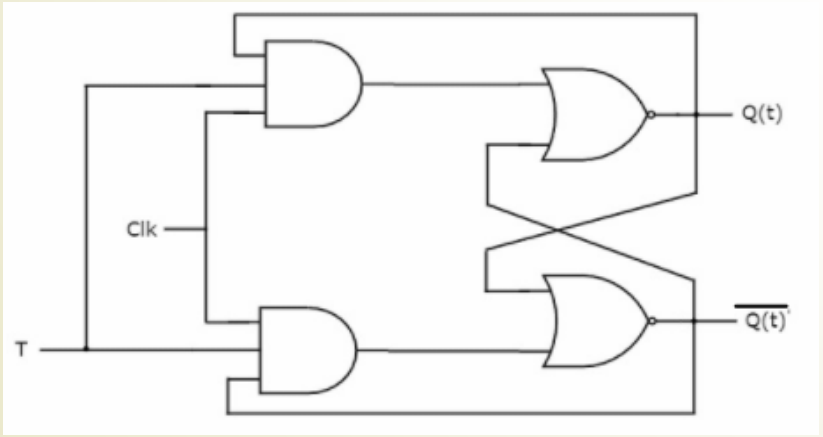
# FLIP-FLOP – T FLIP-FLOP



- The T flipflop is a single input version of the JK flipflop.
- The T flipflop is obtained from JK type if both inputs are tied together.
- The designation T comes from the ability of the flipflop to “Toggle”, or change state. Regardless of the present state of the flipflop, it assumes the complement state when the clock pulse occurs while input T is in logic 1.




Graphical Symbol




Logic diagram

| Clock | T | Q(t) | Q(t+1) | Comment |
|---|---|---|---|---|
| ↓ | 0 | 0 | 0 | Hold |
| ↓ | 0 | 1 | 1 | Hold |
| ↓ | 1 | 0 | 1 | Complement |
| ↓ | 1 | 1 | 0 | Complement |

Characteristic Table

# FLIP-FLOP – T FLIP-FLOP



From the above characteristic table, we can directly write the **next state equation** as

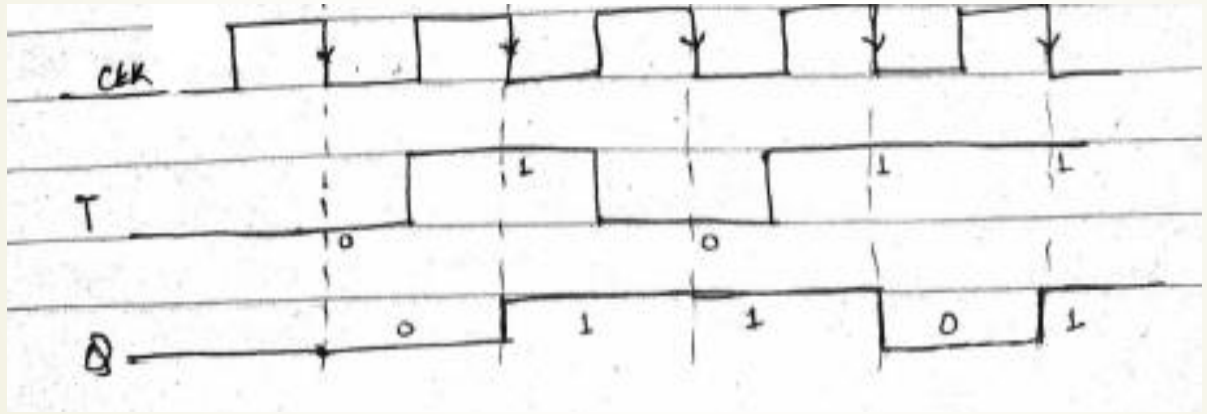
$$Q(t+1) = T'Q(t) + TQ(t)'$$

$$\Rightarrow Q(t+1) = T \oplus Q(t)$$

Boolean function

| T flip-flop | | |
|---|---|---|
| Q(t) | Q(t+1) | T |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Excitation Table




Timing diagram

# MASTER – SLAVE JK FLIPFLOP



## Race Around Condition In JK Flip-flop

- For J-K flip-flop, if J=K=1, and if clk=1 for a long period of time, then Q output will toggle as long as CLK is high, which makes the output of the flip-flop unstable or uncertain.
- This problem is called race around condition in J-K flip-flop.
- This problem (Race Around Condition) can be avoided by ensuring that the clock input is at logic “1” only for a very short time.
- To eliminate race around condition:
  - Use edge- triggered flip-flop
  - Use **Master Slave JK flip flop**.

# MASTER – SLAVE JK FLIPFLOP

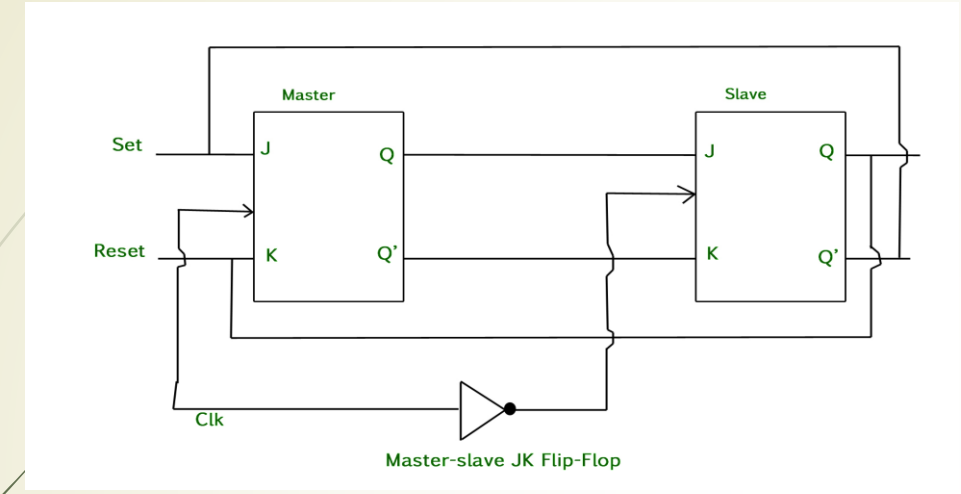


**Master Slave JK flip flop**

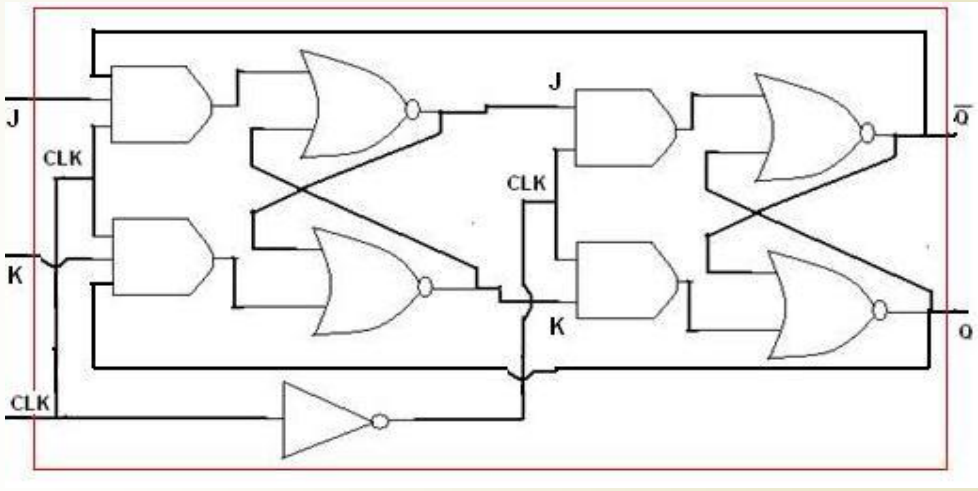
- The Master-Slave Flip-Flop is basically a combination of two JK flip-flops connected together in a series configuration.
- Out of these, one acts as the “master” and the other as a “slave”.
- The output from the master flip flop is connected to the two inputs of the slave flip flop whose output is fed back to inputs of the master flip flop.
- In addition to these two flip-flops, the circuit also includes an inverter.
- The inverter is connected to clock pulse in such a way that the inverted clock pulse is given to the slave flip-flop.
- In other words if CP=0 for a master flip-flop, then CP=1 for a slave flip-flop and if CP=1 for master flip flop then it becomes 0 for slave flip flop.

# MASTER – SLAVE JK FLIPFLOP

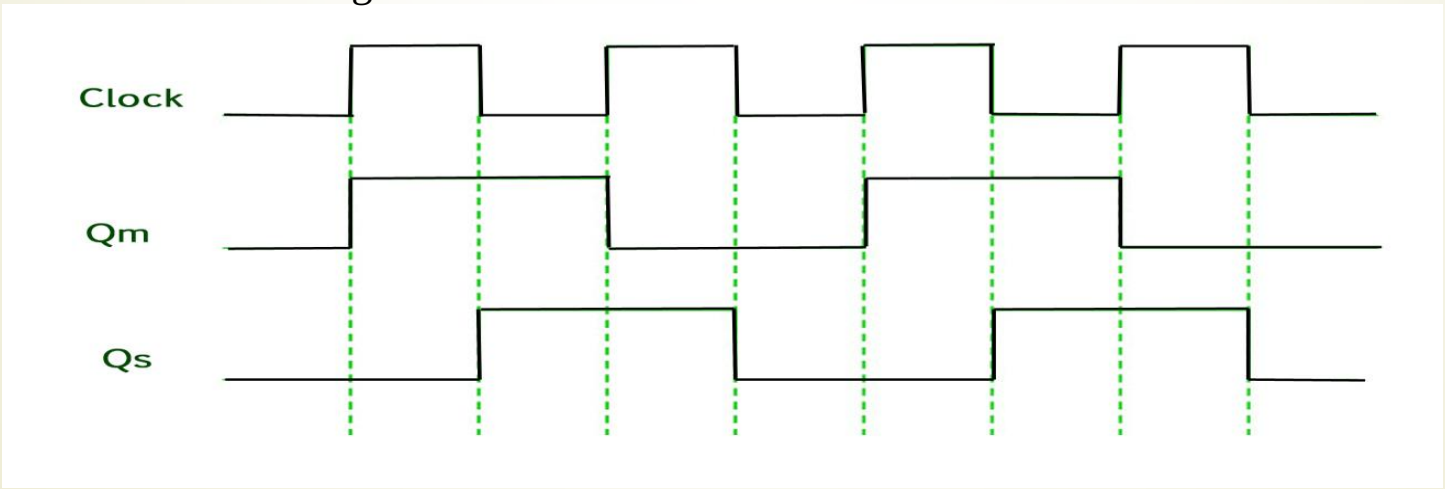





Block diagram




Logic diagram




Timing diagram

# COUNTERS



- Counter is a sequential circuit.
- A digital circuit which is used for a counting pulses is known counter.
- Counter is the widest application of flip-flops. It is a group of flip-flops with a clock signal applied.
- Counters are of two types.
- **Asynchronous or ripple counters.**
  - In asynchronous counter we don't use universal clock, only first flip flop is driven by main clock and the clock input of rest of the following flip flop is driven by output of previous flip flops
- **Synchronous counters.**
  - Unlike the asynchronous counter, synchronous counter has one global clock which drives each flip flop so output changes in parallel.
  - The one advantage of synchronous counter over asynchronous counter is, it can operate on higher frequency than asynchronous counter as it does not have cumulative delay because of same clock is given to each flip flop

# SYNCHRONOUS VS ASYNCHRONOUS COUNTERS

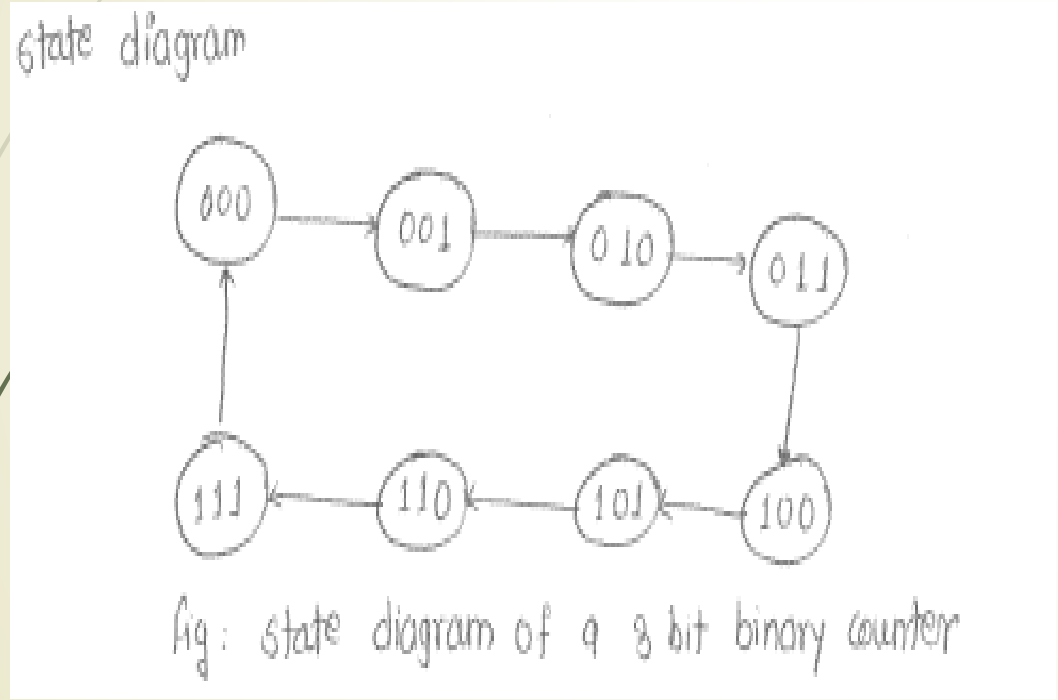
| Sr. No. | Parameter | Asynchronous counter | Synchronous counter |
|---|---|---|---|
| 1. | Circuit complexity | Logic circuit is simple. | With increase in number of states, the logic circuit becomes complicated. |
| 2. | Connection pattern | Output of the preceding FF, is connected to clock of the next FF. | There is no connection between output of preceding FF and CLK of next one. |
| 3. | Clock input | All the FFs are not clocked simultaneously. | All FFs receive clock signal simultaneously. |
| 4. | Propagation delay | P.D. = n * (td) where n is number of FF and td is p.d. per FF. | P.D. = n * (td)FF + (td)gate. It is much shorter than that of asynchronous counter. |
| 5. | Maximum frequency of operation | Low because of the long propagation delay. | High due to shorter propagation delay. |

# ASYNCHRONOUS COUNTERS



## BINARY UP COUNTER

- An ‘N’ bit Asynchronous binary up counter consists of ‘N’ T flip-flops. It counts from 0 to $2^N - 1$.
- **EXAMPLE: 3-BIT BINARY UP COUNTER**




State diagram

| No of negative edge of Clock | $Q_2$ $MSB$ | $Q_1$ | $Q_0$ $LSB$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 |

Count Sequence

# REGISTERS



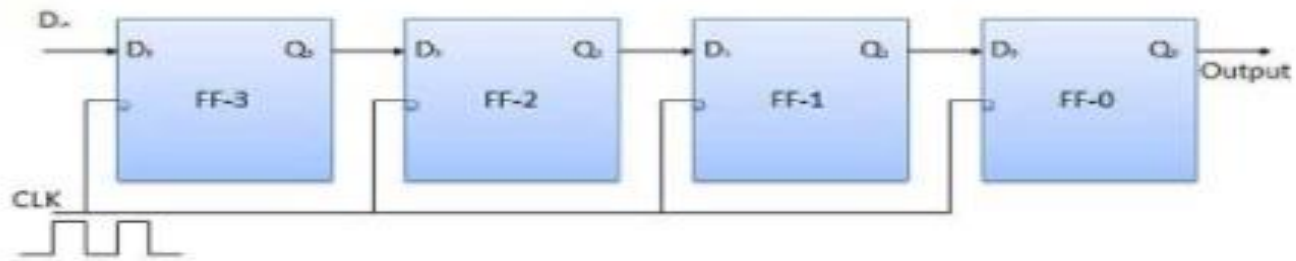
- Flip-flop is a 1 bit memory cell which can be used for storing the digital data.
- To increase the storage capacity in terms of number of bits, we have to use a group of flip-flop.
- Such a group of flip-flop is known as a Register.
- The n-bit register will consist of n number of flip-flop and it is capable of storing an n-bit word.
- The binary data in a register can be moved within the register from one flip-flop to another.
- The registers that allow such data transfers are called as shift registers. There are four mode of operations of a shift register.
  - Serial Input Serial Output
  - Serial Input Parallel Output
  - Parallel Input Serial Output
  - Parallel Input Parallel Output

# REGISTERS

## 101 SERIAL INPUT SERIAL OUTPUT

- Let all the flip-flop be initially in the reset condition i.e. Q3 = Q2 = Q1 = Q0 = 0. If an entry of a four bit binary number 1 1 1 1 is made into the register, this number should be applied to Din bit with the LSB bit applied first. The D input of FF-3 i.e. D3 is connected to serial data input Din. Output of FF-3 i.e. Q3 is connected to the input of the next flip-flop i.e. D2 and so on.
- Also known as, shift right register.

Block Diagram



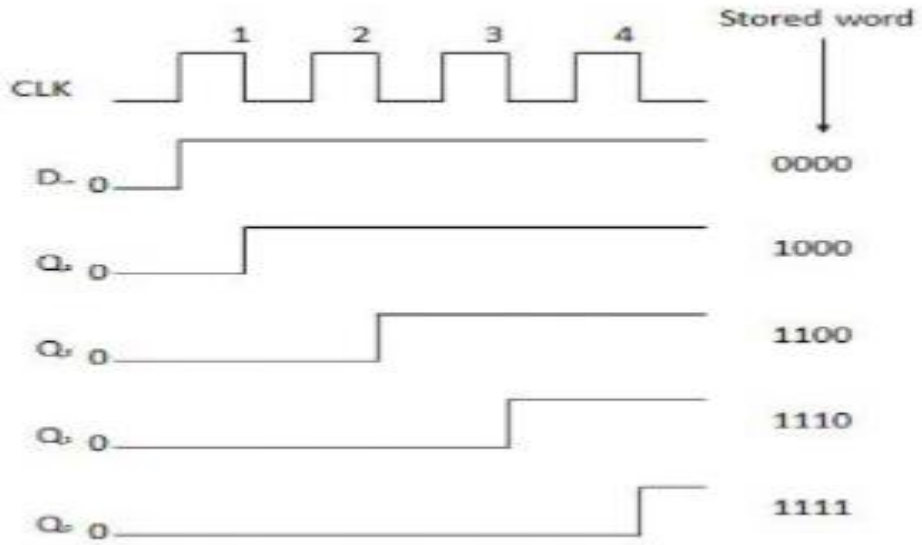
# REGISTERS

Truth Table

| | CLK | $D_0 = Q_3$ | $Q_3 = D_2$ | $Q_2 = D_1$ | $Q_1 = D_0$ | $Q_0$ |
|---|---|---|---|---|---|---|
| Initially | | | 0 | 0 | 0 | 0 |
| (i) | ↓ | 1 | 1 | 0 | 0 | 0 |
| (ii) | ↓ | 1 | 1 | 1 | 0 | 0 |
| (iii) | ↓ | 1 | 1 | 1 | 1 | 0 |
| (iv) | ↓ | 1 | 1 | 1 | 1 | 1 |

→ Direction of data travel

Waveforms



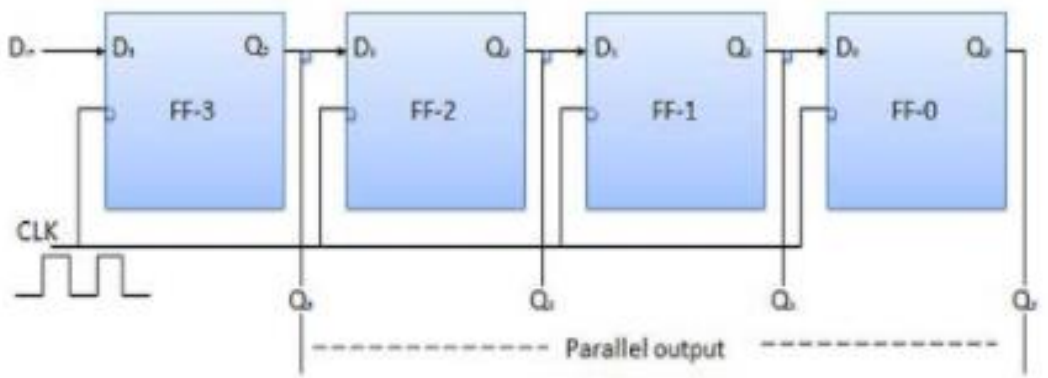
# REGISTERS



## SERIAL INPUT PARALLEL OUTPUT

- In such types of operations, the data is entered serially and taken out in parallel fashion.
- Data is loaded bit by bit. The outputs are disabled as long as the data is loading.
- As soon as the data loading gets completed, all the flip-flops contain their required data, the outputs are enabled so that all the loaded data is made available over all the output lines at the same time.
- 4 clock cycles are required to load a four bit word. Hence the speed of operation of SIPO mode is same as that of SISO mode.

Block Diagram

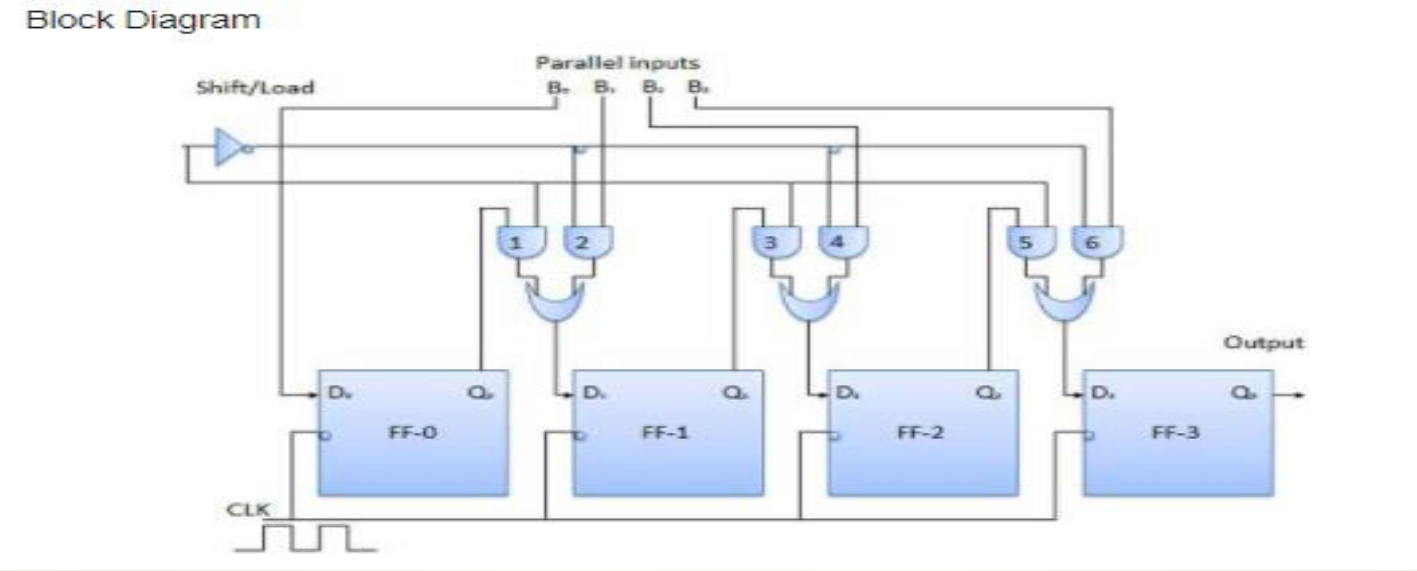


# REGISTERS



## PARALLEL INPUT SERIAL OUTPUT (PISO)

- Data bits are entered in parallel fashion.
- The circuit shown below is a four bit parallel input serial output register.
- Output of previous Flip Flop is connected to the input of the next one via a combinational circuit.
- The binary input word B0, B1, B2, B3 is applied though the same combinational circuit.
- There are two modes in which this circuit can work namely - shift mode or load mode.



# REGISTERS



## Load mode

- When the shift/load bar line is low (0), the AND gate 2, 4 and 6 become active they will pass B1, B2, B3 bits to the corresponding flip-flops. On the low going edge of clock, the binary input B0, B1, B2, B3 will get loaded into the corresponding flip-flops. Thus parallel loading takes place.

## Shift mode

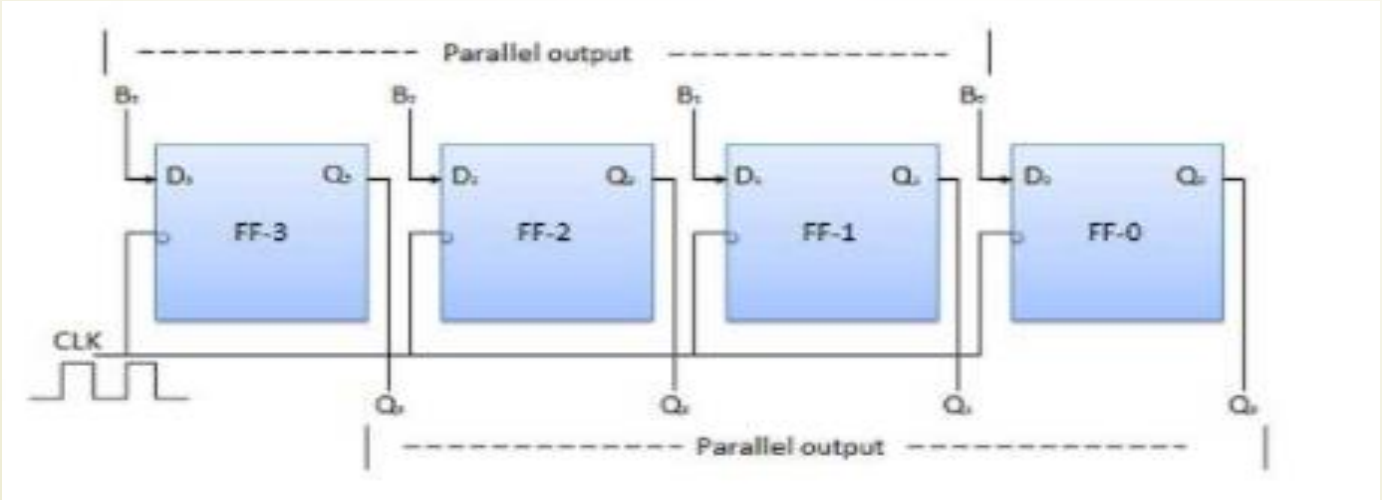
- When the shift/load bar line is high (1), the AND gate 2, 4 and 6 become inactive. Hence the parallel loading of the data becomes impossible. But the AND gate 1,3 and 5 become active. Therefore the shifting of data from left to right bit by bit on application of clock pulses. Thus the parallel in serial out operation takes place.

# REGISTERS



## PARALLEL INPUT PARALLEL OUTPUT (PIPO)

- In this mode, the 4 bit binary input B0, B1, B2, B3 is applied to the data inputs D0, D1, D2, D3 respectively of the four flip-flops.
- As soon as a negative clock edge is applied, the input binary bits will be loaded into the flip-flops simultaneously. The loaded bits will appear simultaneously to the output side. Only clock pulse is essential to load all the bits.



# SHIFT REGISTERS



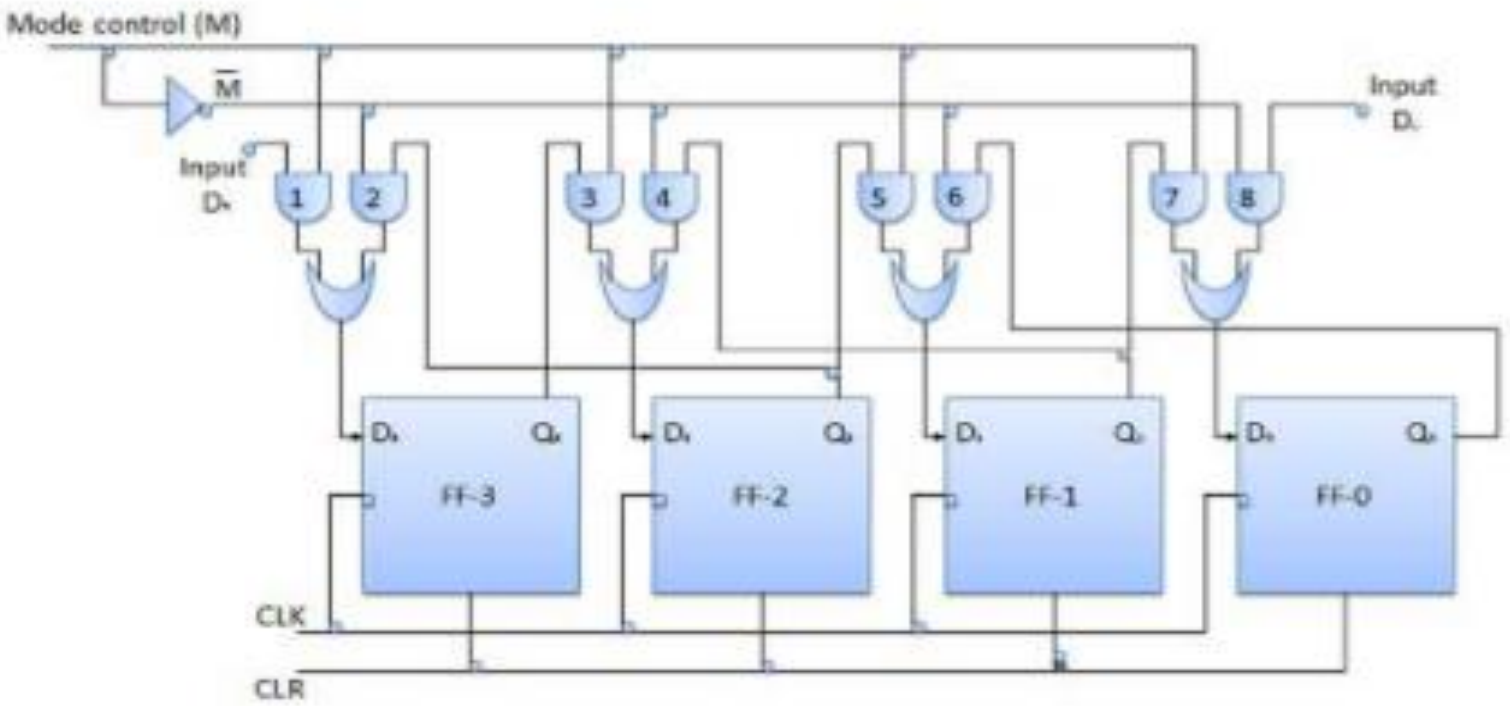
## Bidirectional Shift Register

- If a binary number is shifted left by one position then it is equivalent to multiplying the original number by 2. Similarly if a binary number is shifted right by one position then it is equivalent to dividing the original number by 2.
- Hence if we want to use the shift register to multiply and divide the given binary number, then we should be able to move the data in either left or right direction.
- Such a register is called bi-directional register. A four bit bi-directional shift register is shown in fig.
- There are two serial inputs namely the serial right shift data input DR, and the serial left shift data input DL along with a mode select input (M).

# SHIFT REGISTERS



Block Diagram



# SHIFT REGISTERS



## With M = 1 – Shift right operation

- If M = 1, then the AND gates 1, 3, 5 and 7 are enabled whereas the remaining AND gates 2, 4, 6 and 8 will be disabled.
- The data at DR is shifted to right bit by bit from FF-3 to FF-0 on the application of clock pulses. Thus with M = 1 we get the serial right shift operation

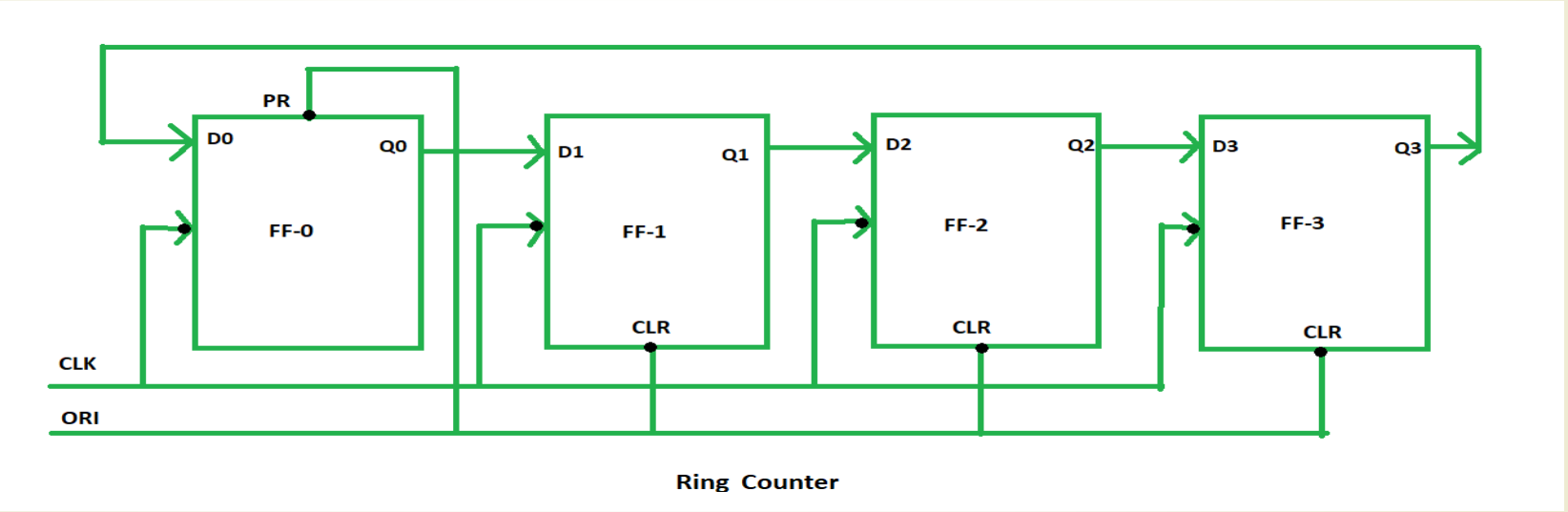
## With M = 0 – Shift left operation

- When the mode control M is connected to 0 then the AND gates 2, 4, 6 and 8 are enabled while 1, 3, 5 and 7 are disabled.
- The data at DL is shifted left bit by bit from FF-0 to FF-3 on the application of clock pulses. Thus with M = 0 we get the serial right shift operation.

# RING COUNTERS



- Ring counter is a typical application of Shift resister.
- Ring counter is almost same as the shift counter.
- The only change is that the output of the last flip-flop is connected to the input of the first flip-flop in case of ring counter but in case of shift resister it is taken as output. Except this all the other things are same.



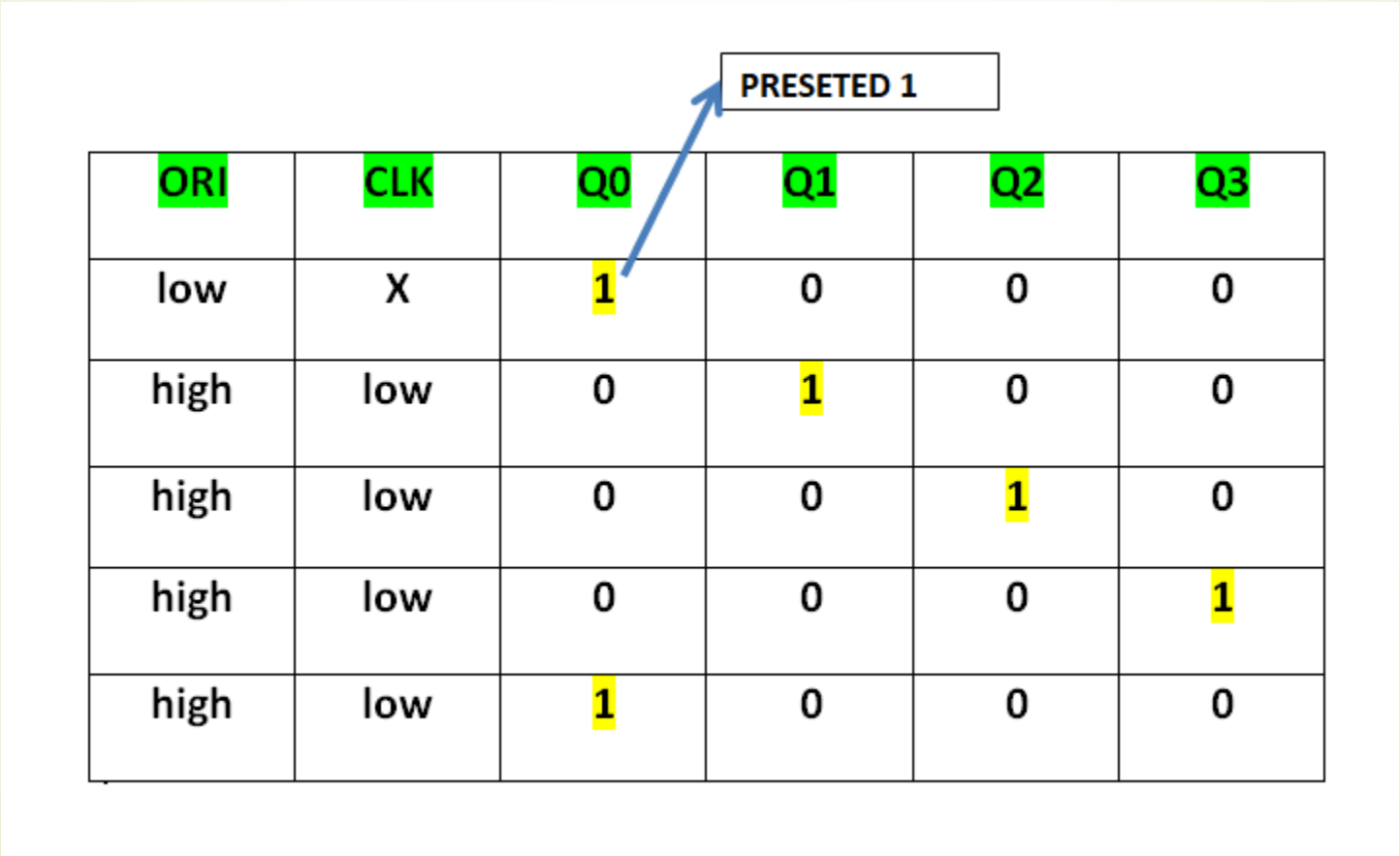

Ring Counter

- In this diagram, we can see that the clock pulse (CLK) is applied to all the flip-flop simultaneously. Therefore, it is a Synchronous Counter.

# RING COUNTERS



- Also, here we use Overriding input (ORI) to each flip-flop. Preset (PR) and Clear (CLR) are used as ORI.
- When PR is 0, then the output is 1. And when CLR is 0, then the output is 0. Both PR and CLR are active low signal that is always works in value 0.




| ORI | CLK | Q0 | Q1 | Q2 | Q3 |
|---|---|---|---|---|---|
| low | X | 1 | 0 | 0 | 0 |
| high | low | 0 | 1 | 0 | 0 |
| high | low | 0 | 0 | 1 | 0 |
| high | low | 0 | 0 | 0 | 1 |
| high | low | 1 | 0 | 0 | 0 |

# RING COUNTERS



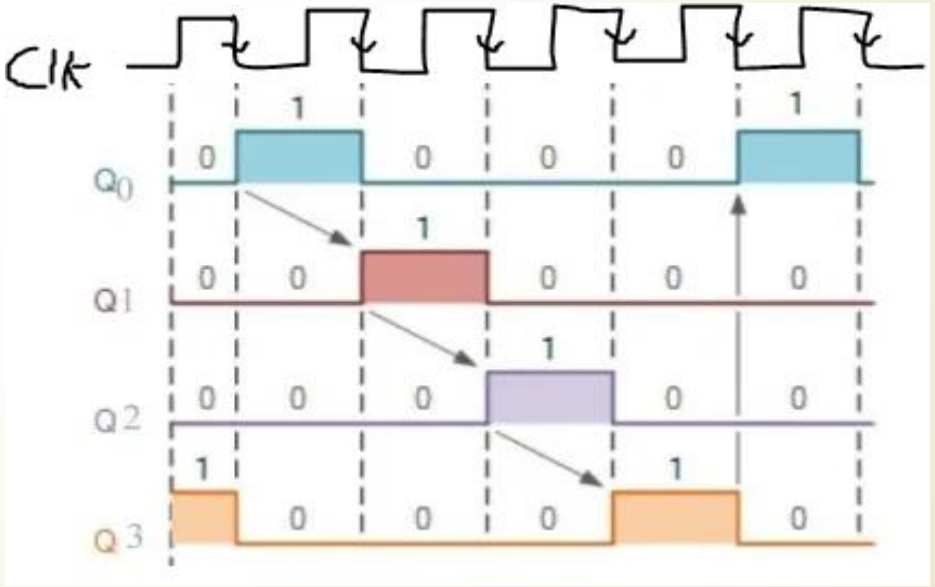
- This Preseted 1 is generated by making ORI low and that time Clock (CLK) becomes don't care.
- After that ORI made to high and apply low clock pulse signal as the Clock (CLK) is negative edge triggered.
- After that, at each clock pulse the preseted 1 is shifted to the next flip-flop and thus form Ring.
- From the above table, we can say that there are 4 states in 4-bit Ring Counter.
- 4 states are:

  1 0 0 0

  0 1 0 0

  0 0 1 0

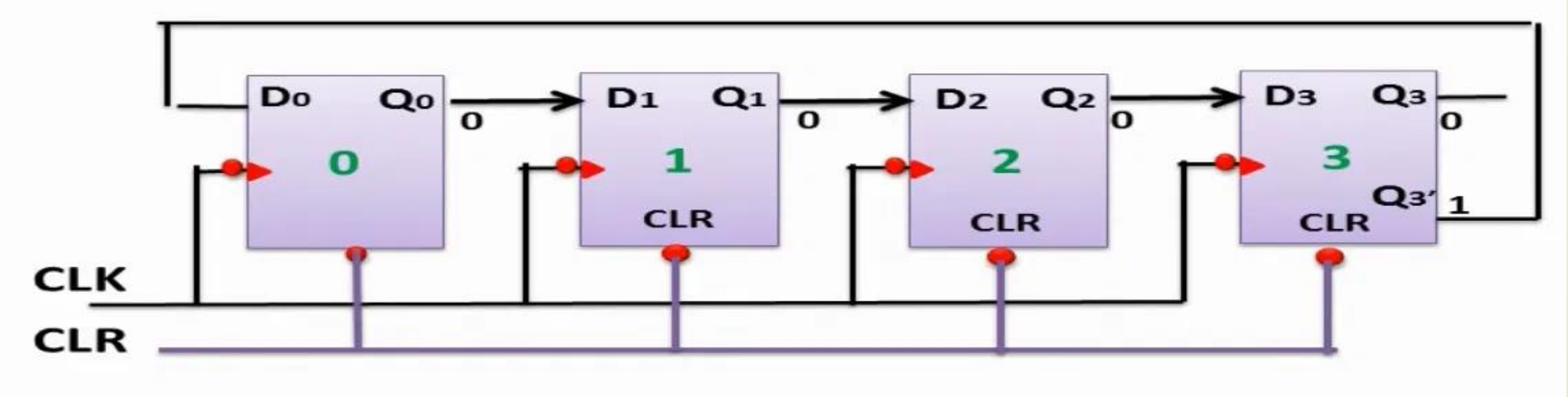
  0 0 0 1



# JOHNSON COUNTERS



- In the ring counter we given the output of the last flip flop into the input of the first flip but in the Johnson counter the last flip flop complemented output is given to the input of the first flip flop.
- In Johnson counter the number of states is equal to twice the number of flip flops.
- So if we use 4 flip flops we will have 8 states so the number of the states are double.
- We applied clock simultaneously to all flip flops.
- The clear input is applied to all the flip flops.



# JOHNSON COUNTERS

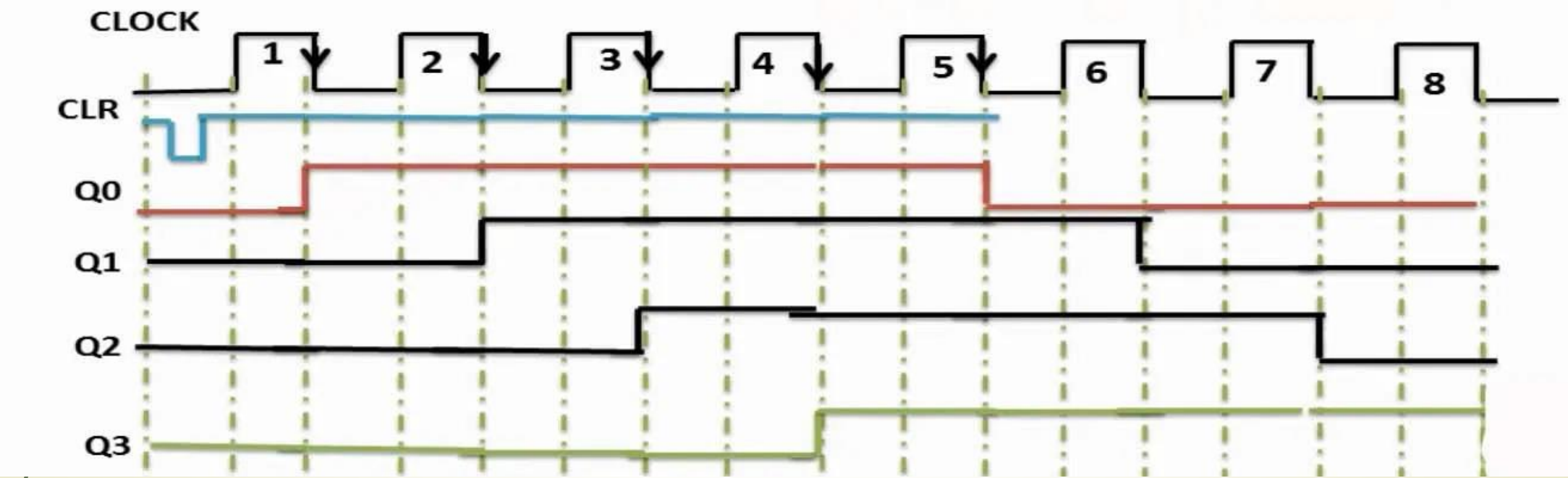


- The output of the first flip flop which is Q0 is given at the input of the second flip flop D1 and the output of the second flip flop which is Q2 is given to input of the third flip flop which is D2 and the complemented output (Q3') will be given to the input of the first flip D0.
- The difference between the ring counter and Johnson counter is that it does not require pre-set.

| Clear/ Pre-set | Clock | Q0 | Q1 | Q2 | Q3 |
|---|---|---|---|---|---|
| 0 | No clock | 0 | 0 | 0 | 0 |
| 1 | ↓ | 1 | 0 | 0 | 0 |
| 1 | ↓ | 1 | 1 | 0 | 0 |
| 1 | ↓ | 1 | 1 | 1 | 0 |
| 1 | ↓ | 1 | 1 | 1 | 1 |
| 1 | ↓ | 0 | 1 | 1 | 1 |
| 1 | ↓ | 0 | 0 | 1 | 1 |
| 1 | ↓ | 0 | 0 | 0 | 1 |
| 1 | ↓ | 0 | 0 | 0 | 0 |

# JOHNSON COUNTERS





# Sequential Circuit – SET A

1. Latches constructed with NOR and NAND gates tend to remain in the latched condition due to which configuration feature?
a) Low input voltages
b) Synchronous operation
c) Gate impedance
Cross coupling

3. The truth table for an S-R flip-flop has how many VALID entries?
a) 1
b) 2
3
d) 4

5. Which of the following is correct for a gated D-type flip-flop?
The Q output is either SET or RESET as soon as the D input goes HIGH or LOW
b) The output complement follows the input when enabled
c) Only one of the inputs can be HIGH at a time
d) The output toggles if one of the inputs is held HIGH

4. When both inputs of a J-K flip-flop cycle, the output will _____
a) Be invalid
b) Change
Not change
d) Toggle

6. A basic S-R flip-flop can be constructed by cross-coupling of which basic logic gates?
a) AND or OR gates
b) XOR or XNOR gates
NOR or NAND gates
d) AND or NOR gates

7. The logic circuits whose outputs at any instant of time depends only on the present input but also on the past outputs are called _____
a) Combinational circuits
Sequential circuits
c) Latches
d) Flip-flops

# Sequential Circuit – SET A

8. Whose operations are more faster among the following?
a) Combinational circuits
b) Sequential circuits
c) Latches
d) Flip-flops

9. How many types of sequential circuits are?
a) 2
b) 3
c) 4
d) 5

12. In S-R flip-flop, if Q = 0 the output is said to be _____
a) Set
b) Reset
c) Previous state
d) Current state

13. The output of latches will remain in set/reset untill _____
a) The trigger pulse is given to change the state
b) Any pulse given to go into previous state
c) They don't get any pulse more
d) The pulse is edge-triggered

14. What is a trigger pulse?
a) A pulse that starts a cycle of operation
b) A pulse that reverses the cycle of operation
c) A pulse that prevents a cycle of operation
d) A pulse that enhances a cycle of operation

The data sheet of a certain flip-flop specifies that the minimum HIGH time $t_w(H)$ for the clock pulse is 16 nanoseconds and the minimum LOW time $t_w(L)$ is 29 nanoseconds. What is the maximum operating frequency for the given flip-flop?

1. 62·50 MHz
2. 31·25 MHz
3. 22·22 MHz
4. 11·11 MHz

# Sequential Circuit – SET A

**In which flip flop the present input will be the next output?**

S-R

J-K

D

T

**The clear input is used to make output _____**

- Q=1
- Q=0
- Invalid
- No change

**A flip flop is an _____**

- Edge sensitive device
- Synchronous device
- Both a and b
- None of the above

**The preset input is used to make output _____**

- Q=1
- Q=0
- Invalid
- No change

**The shift registers are categorized into _____**

- One
- Two
- Three
- Four

**_____ are the applications of flip flop**

- Registers
- Counters
- Storage devices
- All of the above

# Sequential Circuit – SET A

**The number of flip-flops in a shift register is dependent upon?**

- The modulus of the counter
- The number of stages in the counter
- The number of digits to be stored
- None of the above

**. A shift register is a digital circuit that _____**

- Stores data.
- Shifts the data from left to right.
- Shifts the data from right to left.
- all of the above.

**What type of register is a shift register?**

- Digital
- Analog
- Both 1 and 2
- Neither 1 nor 2

**. In which mode can we provide data to all the flip-flops simultaneously?**

- Loopback mode
- Serial input mode
- Parallel input mode
- All of the above

**What is a shift register?**

- An adder circuit
- A memory circuit
- A combinational circuit
- A decoder circuit

**Which of the following is true about shift registers?**

- It is not used to store multi-bit data.
- They are available only in the parallel mode of operation.
- They are useful for data transfer from one location to another.
- All of the above.

# Sequential Circuit – SET A

1. Based on how binary information is entered or shifted out, shift registers are classified into _____ categories.
a) 2
b) 3
c) 4
d) 5

2. The full form of SIPO is _____
a) Serial-in Parallel-out
b) Parallel-in Serial-out
c) Serial-in Serial-out
d) Serial-In Peripheral-Out

3. A shift register that will accept a parallel input or a bidirectional serial load and internal shift features is called as?
a) Tristate
b) End around
c) Universal
d) Conversion

4. How can parallel data be taken out of a shift register simultaneously?
a) Use the Q output of the first FF
b) Use the Q output of the last FF
c) Tie all of the Q outputs together
d) Use the Q output of each FF

6. The group of bits 11001 is serially shifted (right-most bit first) into a 5-bit parallel output shift register with an initial state 01110. After three clock pulses, the register contains _____
a) 01110
b) 00001
c) 00101
d) 00110

7. Assume that a 4-bit serial in/serial out shift register is initially clear. We wish to store the nibble 1100. What will be the 4-bit pattern after the second clock pulse? (Right-most bit first)
a) 1100
b) 0011
c) 0000
d) 1111

# Sequential Circuit – SET A

A serial in/parallel out, 4-bit shift register initially contains all 1s. The data nibble 0111 is waiting to enter. After four clock pulses, the register contains _____

a) 0000
b) 1111
0111
d) 1000

9. With a 200 kHz clock frequency, eight bits can be serially entered into a shift register in _____

a) 4 µs
40 µs
c) 400 µs
d) 40 ms

1. How many flip-flops are required to make a MOD-32 binary counter?

Ⓐ 3
Ⓑ 45
5
Ⓓ 6

2. Using four cascaded counters with a total of 16 bits, how many states must be deleted to achieve a modulus of 50,000?

Ⓐ 50,000
Ⓑ 65,536
Ⓒ 25,536
15,536

3. A MOD-16 ripple counter is holding the count $1001_2$. What will the count be after 31 clock pulses?

$1000_2$
Ⓑ $1010_2$
Ⓒ $1011_2$
Ⓓ $1101_2$

4. The terminal count of a modulus-11 binary counter is _____.

1010
Ⓑ 1000
Ⓒ 1001
Ⓓ 1100

An eight-stage ripple counter uses a flip-flop with propagation delay of 75 nanoseconds. The pulse width of the strobe is 50ns. Frequency of the input signal which can be used for proper operation of the counter is approximately

2 points

- a. 1MHz
- b. 2MHz
- c. 500MHz
- d. 4MHz

**Concept:**

In case of ripple or asynchronous counter total propagation delay $T_{pd} = n\ t_{pd}$

Where n = no. of flip flops

$t_{pd}$ = propagation delay of each flip flop

If the strobe signal is given, the pulse width of the strobe is also added, i.e.

$T_{pd} = n\ t_{pd} + T_s$

.. A 4-bit ripple counter consists of flip-flop that each have propagations delay form clock to Q output of 12ns. For the counter to recycle from 1111 to 0000, it takes a total of

2 points

- a. 12ns
- b. 48ns
- c. 24ns
- d. 36ns

**Concept:**

- For an n-bit ripple counter, the MSB is generated only when the carry from all the previous flip-flip is propagated to the MSB flip flop.
- So, the maximum time(Worst-Case delay) taken for the output of the **Ripple counter to be stable = n × $t_d$** (where $t_d$ is the propagation delay of each flip flop)

63. The divide by 60 counter in digital clock is implemented by using two cascading counters 2 points

- [x] Mod-6, Mod-10
- [ ] Mod-50, Mod-10
- [ ] Mod-10. Mod-50
- [ ] Mod-50, Mod-6

64. In a certain digital waveform, the period is twice the pulse width. the duty cycle is ___ % 2 points

- [ ] 100
- [x] 50
- [ ] 200
- [ ] 25

**2.4 Microprocessor**: Internal Architecture and Features of microprocessor, Assembly Language Programming. (AExE0204)

# INTRODUCTION TO MICROPROCESSOR

- Microprocessor is a multipurpose, programmable, clock-driven, register-base, electronic device that reads binary instructions from a storage device called memory, accepts binary data as input and processes data according to instructions, and provides results as output.
  - Input: Digital input from I/O or Memory
  - Process: Process the instruction
  - Output: Digital output to I/O or memory
- A typical programmable machine can be represented with four components: microprocessor, memory, input, and output. The physical components of this system are called hardware. A set of instructions written for the microprocessor to perform a task is called a program, and a group of program is called software.

# INTRODUCTION TO MICROPROCESSOR

- In general microprocessor performs three basic tasks:
  1. Data transfer between itself and memory or I/O unit
  2. Arithmetical or Logical operation on data, and
  3. Switching and decision making
- Operation on above tasks are based on the instruction provided to the microprocessor. So, a critical steps in performing any task is retrieval of instruction, understanding the command and executing it.
- This can be listed as:
  1. Fetch the instruction
  2. Decode instruction
  3. Execute the command from the instruction

# INTRODUCTION TO MICROPROCESSOR

- A microprocessor incorporates the functions of a computer's central processing unit (CPU) on a single integrated chip (IC).

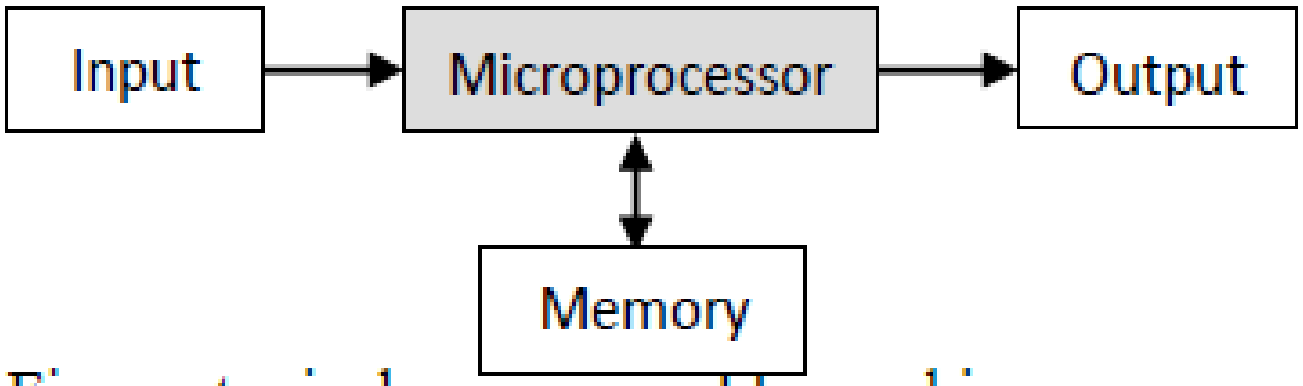



Figure: typical programmable machine

# MICROCONTROLLER

- A microcontroller is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications.

# MICROPROCESSOR VS MICROCONTROLLER

| Microprocessor | Micro Controller |
|---|---|
|  |  |
| Microprocessor is heart of Computer system. | Micro Controller is a heart of embedded system. |
| It is just a processor. Memory and I/O components have to be connected externally | Micro controller has external processor along with internal memory and i/O components |
| Since memory and I/O has to be connected externally, the circuit becomes large. | Since memory and I/O are present internally, the circuit is small. |
| Cannot be used in compact systems and hence inefficient | Can be used in compact systems and hence it is an efficient technique |
| Cost of the entire system increases | Cost of the entire system is low |
| Due to external components, the entire power consumption is high. Hence it is not suitable to used with devices running on stored power like batteries. | Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries. |
| Most of the microprocessors do not have power saving features. | Most of the micro controllers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further. |
| Since memory and I/O components are all external, each instruction will need external operation, hence it is relatively slower. | Since components are internal, most of the operations are internal instruction, hence speed is fast. |
| Microprocessor have less number of registers, hence more operations are memory based. | Micro controller have more number of registers, hence the programs are easier to write. |
| Microprocessors are based on von Neumann model/architecture where program and data are stored in same memory module | Micro controllers are based on Harvard architecture where program memory and Data memory are separate |
| Mainly used in personal computers | Used mainly in washing machine, MP3 players |

# ORGANIZATION OF MICROPROCESSOR BASED SYSTEM

- The basic structure of microprocessor based system includes microprocessor, I/O and memory (ROM & R/WM). These components are organized around a common communication path called bus.

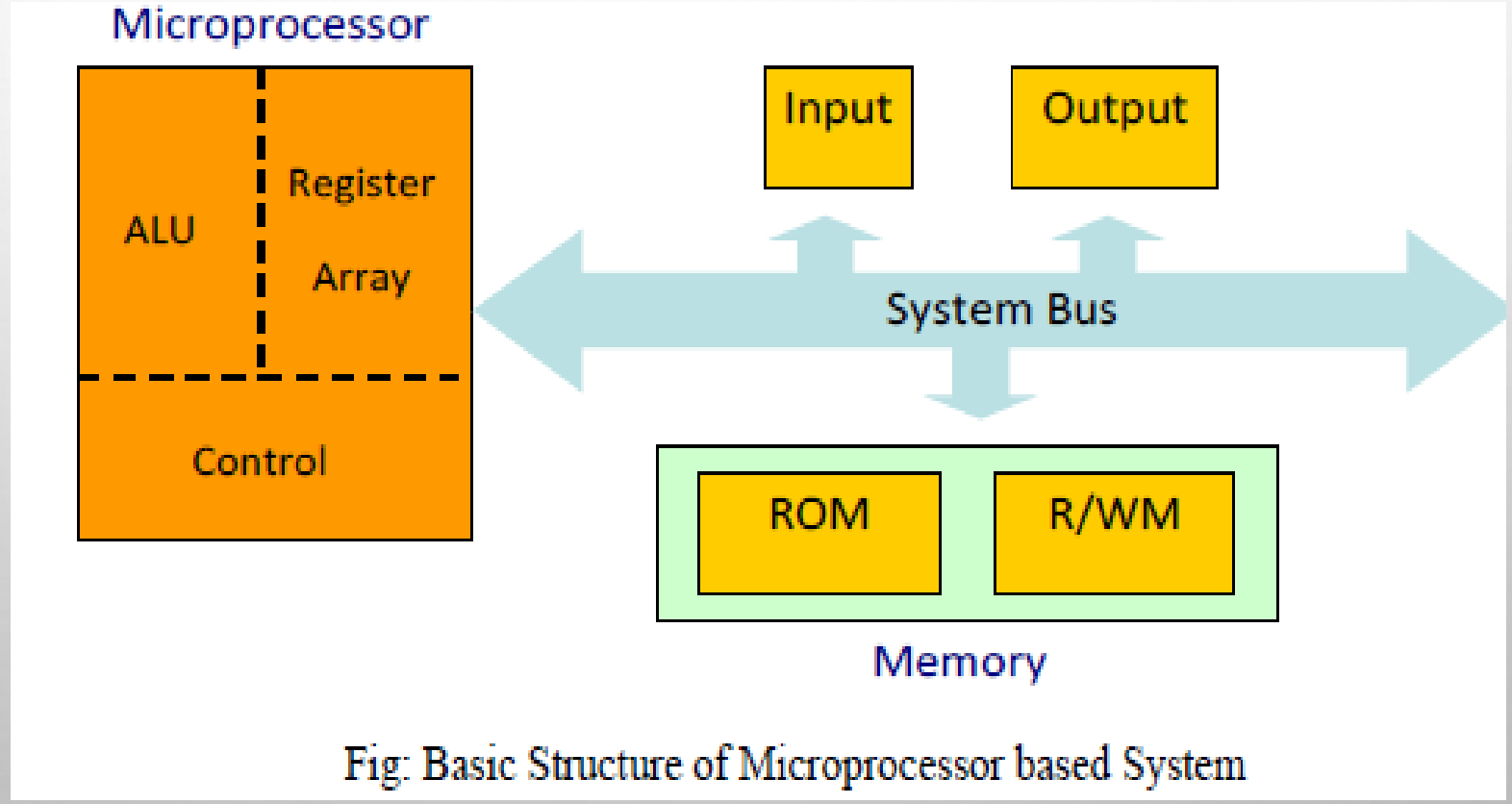



Fig: Basic Structure of Microprocessor based System

# BUS ORGANIZATION/ 3-BUS ARCHITECTURE

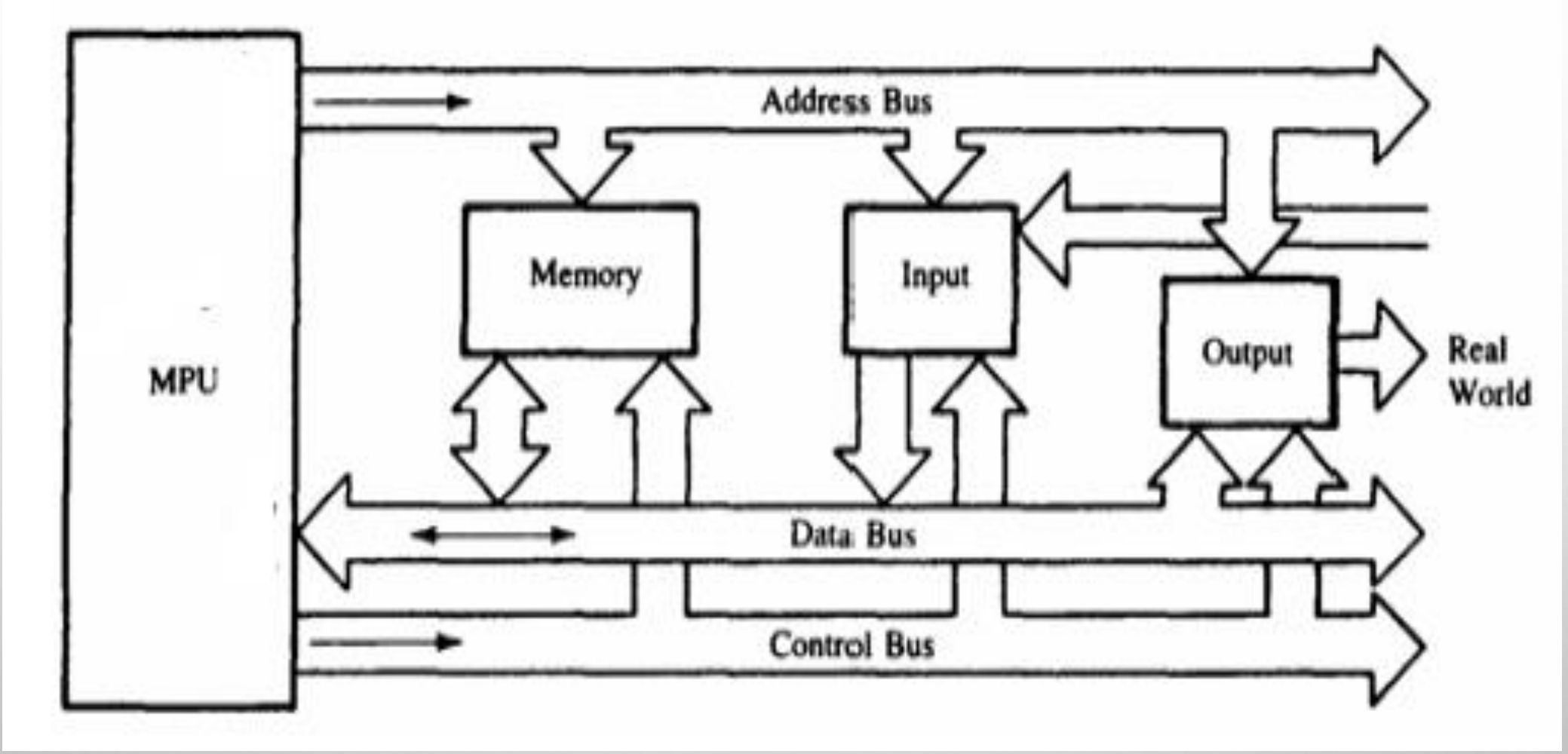



Fig: 3-Bus Architecture of Microprocessor

# BUS ORGANIZATION/ 3-BUS ARCHITECTURE

**Address bus:**

- The address bus consists of 16, 20, 24 or 32 parallel signal lines that is used to specify a physical address.
- On these lines the CPU sends out the address of the memory location that is to be written to or from.
- The address bus is unidirectional: bit flow in one direction- from the MPU to peripheral devices.
- The width of the address bus determines the amount of memory a system can address.
- For example, a system with a 16-bit address bus can address $2^{16}$ (64 KB) memory locations.

# BUS ORGANIZATION/ 3-BUS ARCHITECTURE

**Data bus:**

- The data bus consist of 8, 16 or 32 parallel signal lines and are bidirectional that carries the actual data being processed.
- CPU can read data in from memory and send data out to memory on these lines.

# BUS ORGANIZATION/ 3-BUS ARCHITECTURE

**Control bus:**

- The control bus is comprised of various single lines that carry synchronization signals.
- The MPU uses such lines to provide timing signals.
- These are no a group of lines like data and address buses, but individual lines that provide a pulse to indicate an MPU operation.
- The MPU generates specific control signals for every operations (such as Memory read or I/O write) it performs.
- These signals are used to identify a device type with which the MPU intended to communicate.

# STORED PROGRAM CONCEPT AND VON NEUMANN MACHINE

- The task of entering and altering the programs for the ENIAC (electronic numerical integrator and computer) was extremely tedious.
- The programming concept could be facilitated if the program could represent in a form suitable for storing in memory alongside the data.
- Then a computer could get its instruction by reading them form the memory and a program could be set or altered by setting the values of a portion of memory.
- This approach is known “stored program concept”, was first adopted by John Von Neumann and hence the architecture of computer he proposed is name as Von-Neumann’s architecture.
- (Note: 8085 μp uses this architecture)

# STORED PROGRAM CONCEPT AND VON NEUMANN MACHINE

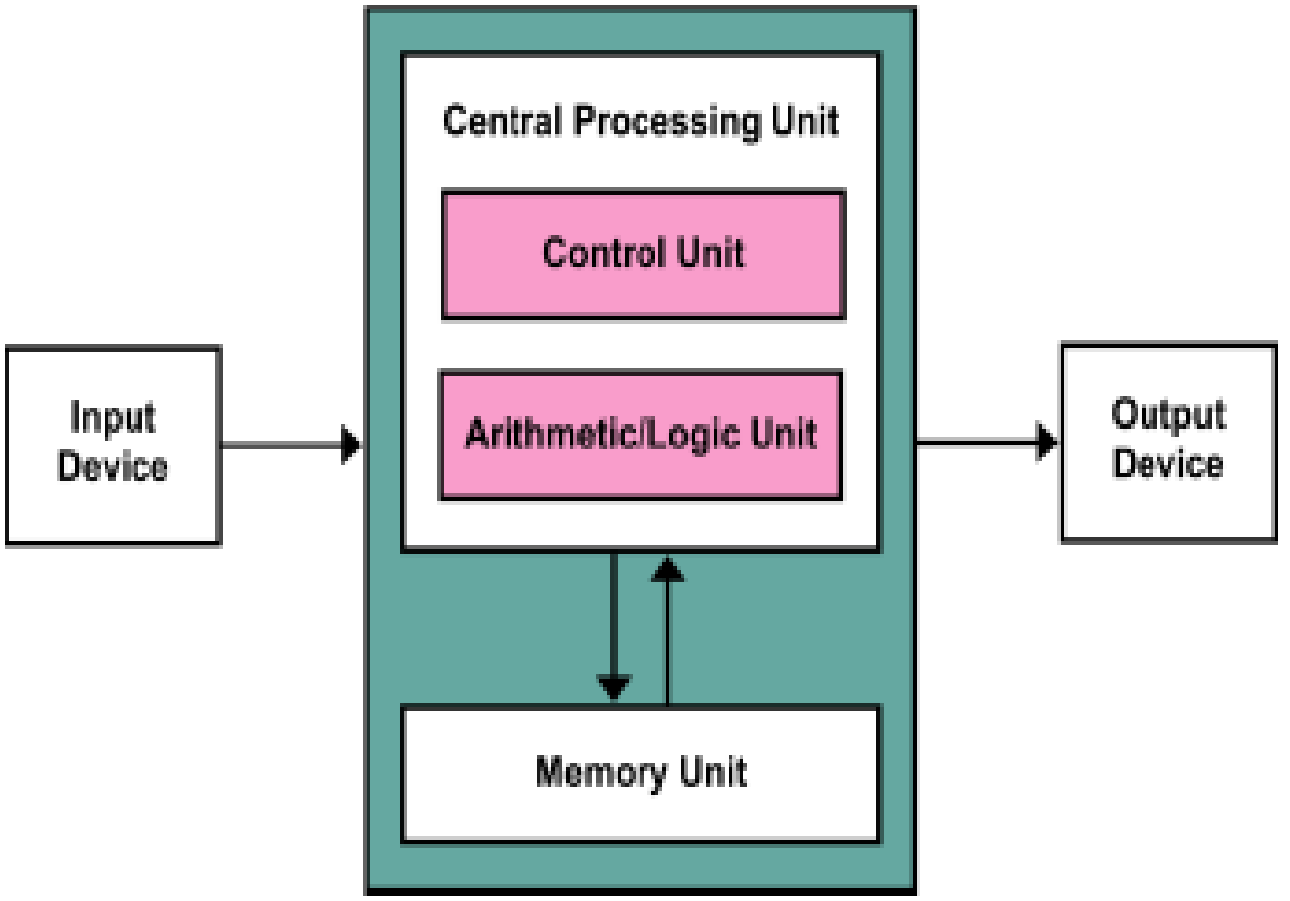



Fig: Von-Neumann Architecture

# STORED PROGRAM CONCEPT AND VON NEUMANN MACHINE

- Main memory is used to store both data and instructions.
- The ALU is capable for performing arithmetic and logical operation on binary data.
- The control unit (CU) interprets the instruction in memory and causes them to be executed.
- The I/O unit gets operated from the control unit. The input/output unit helps inputting data and getting results.
- The Von-Neumann's Architecture is the fundamental basis for the architecture of today's digital computers.

# STORED PROGRAM CONCEPT AND VON NEUMANN MACHINE

- The memory of Von-Neumann machine consists of thousand storage location called words of 40 binary digits (bits).
- Both data and instruction are stored in it.
- The storage locations of control unit and ALU are called registers.
- The various registers of this model are MBR, MAR, IR, IBR, PC, AC.
- **Memory Buffer Register (MBR):**
  - It consists of a word to be stored in memory or is used to receive a memory or is used to receive a word from memory.
- **Memory address Register (MAR):**
  - It contains the address in memory of the world to be written from or read into the MBR.

# STORED PROGRAM CONCEPT AND VON NEUMANN MACHINE

- **Instruction register (IR):**
  - Contain the 8 bit op-code (operation code) instruction being executed.
- **Instruction buffer register (IBR):**
  - It is used to temporarily hold the instruction from a word in memory.
- **Program counter (PC):**
  - It contains address of next instruction to be fetched from memory.
- **Ac (Accumulator) and MQ (multiplier quotient):**
  - They are employed to temporarily hold operands and results of ALU operations.

# STORED PROGRAM CONCEPT AND VON NEUMANN MACHINE

**Advantages:**

- Computer can handle instruction as easily as data
- Ease of loading program into memory
- Efficient use of memory
- Cost effective due to same program and data memory

**Disadvantages:**

- Required special hardware protection mechanism to protect instruction and data being overlapped by each other.
- Low speed because concurrent fetching of data and instruction was not possible.

# HARVARD ARCHITECTURE

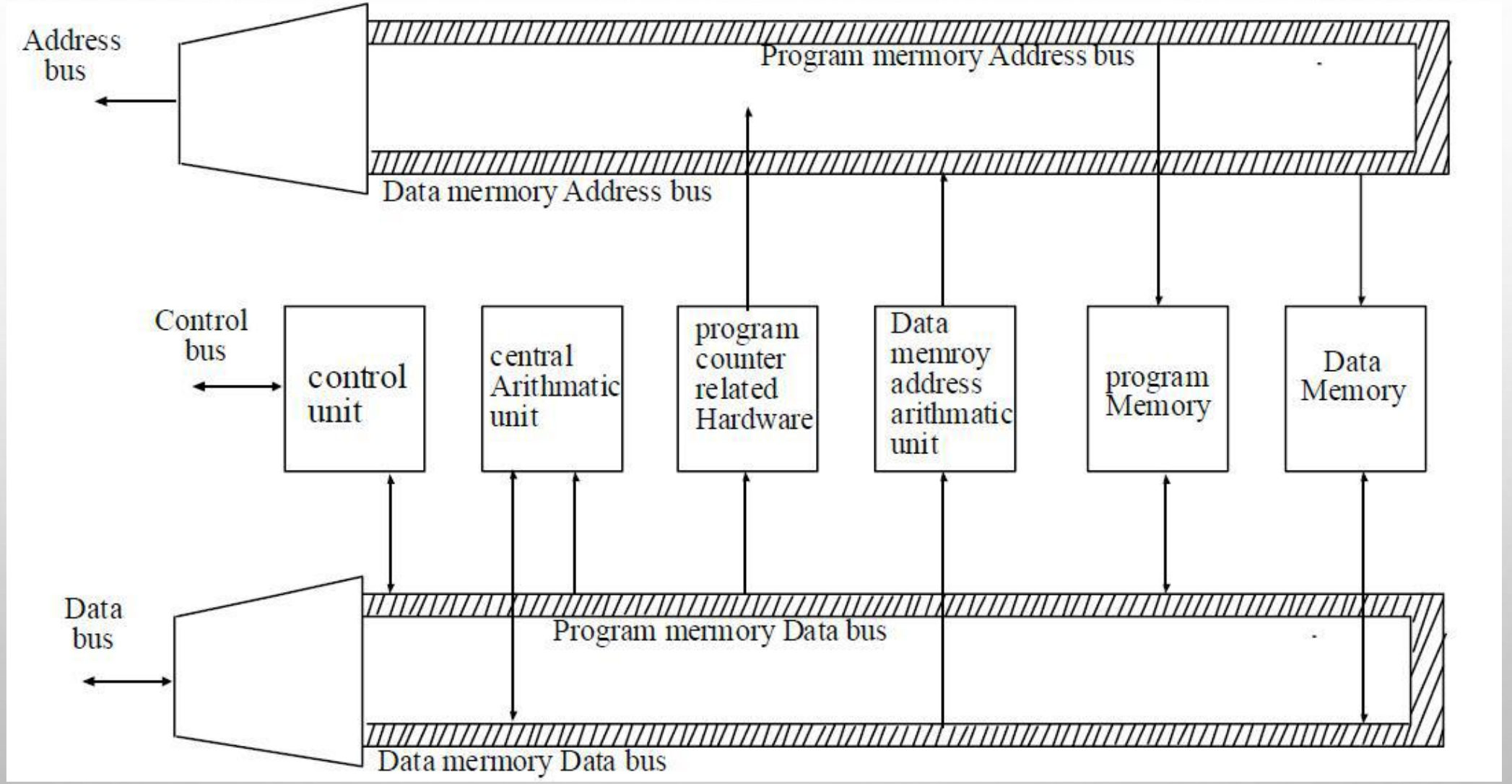





Fig. Block diagram of the Harvard architecture

# HARVARD ARCHITECTURE

- Harvard Architecture based computer consist so separate memory spaces for the programs (instruction) and data.
- Each memory space has its own address and data bus.
- Thus both instruction and data can fetch from memory concurrently.
- From the figure it is seen that there are two data and two address buses for the program and data memory spaces respectively.
- The program memory data bus and data memory data are multiplexed to form single data bus where as program memory Address and data memory address are multiplexed to form single address bus.

# HARVARD ARCHITECTURE

- Hence there are two blocks of RAM chip: One for program memory and another for data memory space.
- Data memory address arithmetic unit generates data memory address.
- The data memory address bus carries the memory address of data where as program memory address bus carries the memory address of the instruction.
- Central arithmetic logic unit consists of the ALU, multiplier, Accumulator, etc.
- The Program Counter is used to address program memory.
- PC always contains the address of next instruction to be fetched. Control unit control the sequence of operations to be executed.
- The data and control bus are bidirectional where as address bus is unidirectional.

# HARVARD ARCHITECTURE

**Advantages**:

- Concurrent fetching of data and instruction was possible so it provide higher speed
- No overwriting of program and data

**Disadvantages**:

- Methods or mechanism of storing program into program memory and data into data memory had to be developed
- Higher cost due to separate program and data memory
- No optimum use of memory.

# INTRODUCTION TO REGISTER TRANSFER LANGUAGE (RTL)

FETCH – REGISTERS

- **Memory Address Register (MAR)**
  - Connected to address bus
  - Specifies address for read or write op
- **Memory Buffer Register (MBR)**
  - Connected to data bus
  - Holds data to write or last data read
- **Program Counter (PC)**
  - Holds address of next instruction to be fetched
- **Instruction Register (IR)**
  - Holds last instruction fetched

# INTRODUCTION TO 8085 MICROPROCESSOR

- The 8085 μp is an 8 bit general purpose microprocessor having 16 bit address lines (capable of addressing $2^{16}$= 65536 bytes= 64 KB of memory).
- The device has 40 pins, requires a +5V single power supply and can operate with 3 MHZ single phase clock.

# INTERNAL ARCHITECTURE AND FEATURE OF 8085 MICROPROCESSOR

- The main components of 8085 μp, as shown in functional block diagram, are the arithmetic/logic unit (ALU), Register array, timing and control unit, instruction register and decoder, interrupt control and serial I/O control. These are linked by an internal data bus.

# INTERNAL ARCHITECTURE AND FEATURE OF 8085 MICROPROCESSOR

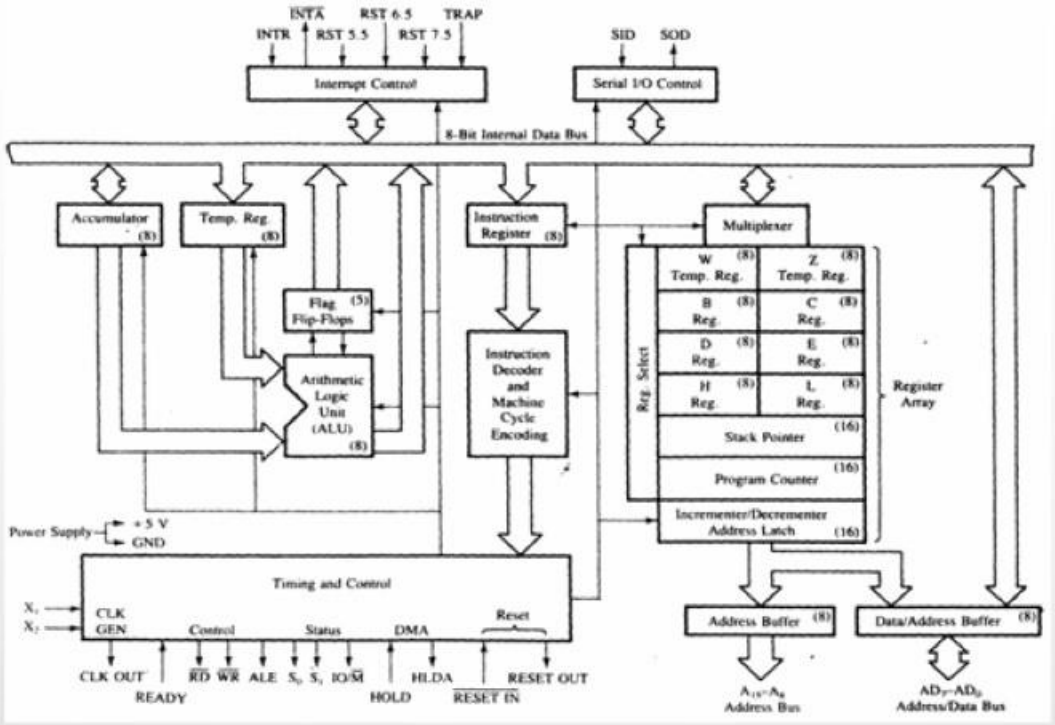
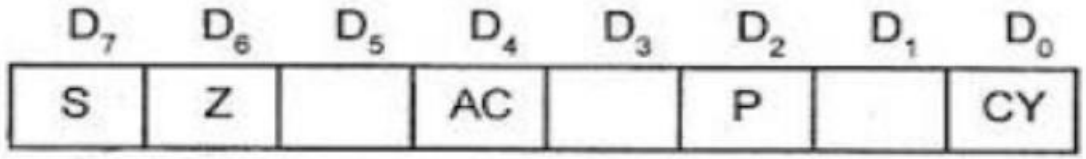



Figure: functional block diagram of 8085 microprocessor

# INTERNAL ARCHITECTURE AND FEATURE OF 8085 MICROPROCESSOR




- **S- Sign flag:**
  - After the execution of arithmetic or logic operation, if D7 bit (MSB: most significant bit) of the result (usually in accumulator) is 1, the sign flag (S) is set. Otherwise it is reset. (For signed number operation D7 bit indicates the sign(positive or negative) of number).
- **Z- Zero flag:**
  - Set i.e. 1, if the result of last operation is zero, and the flag is reset i.e. 0 if the result is not 0. This flag is often used in loop control and in searching for particular data value.

# INTERNAL ARCHITECTURE AND FEATURE OF 8085 MICROPROCESSOR

- **AC- Auxiliary Carry:**
  - In an arithmetic operation, when a carry is generated by digit D3 and passed on to digit D4, the AC flag is set. The flag is used only internally for BCD operations and is not available for programmer.
- **P- Parity flag:**
  - Set i.e. 1, if the number of 1 in the result of the last operation is even. If the result has an odd number of 1's the flag is reset i.e. 0.
- **C- Carry flag:**
  - Flag is set i.e. 1, if the result of the last operation generates a carry, otherwise it is reset i.e. 0.

# INTERNAL ARCHITECTURE AND FEATURE OF 8085 MICROPROCESSOR

**Instruction register and decoder :**

- The instruction register receives the operation codes of instructions from the internal data bus and passes it to the instruction decoder and machine cycle encoder circuit.
- The decoder decodes the instruction and establishes the sequence of events to follow.
- The instruction register is not accessible to the programmer.

**Register array :**

- The 8085 μp has 6 general purpose registers to hold 8 bit data; these are B, C, D, E, H, and L.
- They can also be combined as register pairs BC, DE, And HL to perform 16 bit operation.
- These registers are accessible to programmer i.e. programmable.
- In addition, the register H and L are utilized in indirect addressing mode. In this mode, the memory location whose address is specified by contents of the register pair.

# INTERNAL ARCHITECTURE AND FEATURE OF 8085 MICROPROCESSOR

- Register array also includes two additional register W & Z, called temporary registers, each used to hold 8 bit data during the execution of some instructions.
- However, they are not available to programmer.
- Program counter (PC) and Stack Pointer (SP) are two 16 bit registers used to hold memory addresses.
- A stack is an area of R/W memory set aside for the purpose of storing data, by an operation known as stacking.
- The beginning of stack is defined by loading a 16 bit address in the stack pointer register.
- A computer program consists of the sequence of coded instructions. These instructions are stored sequentially in the memory location.

# INTERNAL ARCHITECTURE AND FEATURE OF 8085 MICROPROCESSOR

**Interrupt controls:**

- The various interrupt controls signals (INTR, RST 5.5, RST 6.5, RST 7.5 and TRAP) are used to interrupt a microprocessor.

**Serial I/O controls:**

- Two serial I/O control signals (SID and SOD) are used to implement the serial data transmission.

# INTERNAL ARCHITECTURE AND FEATURE OF 8085 MICROPROCESSOR

**The salient features of 8085 µp are:**

- It is a 8 bit microprocessor.
- It is manufactured with N-MOS technology.
- It has 16-bit address bus and hence can address up to $2^{16} = 65536$ bytes (64KB) memory locations through $A_0$-$A_{15}$.
- The first 8 lines of address bus and 8 lines of data bus are multiplexed $AD_0 - AD_7$.
- Data bus is a group of 8 lines $D_0 - D_7$.
- It supports external interrupt request.

# INTERNAL ARCHITECTURE AND FEATURE OF 8085 MICROPROCESSOR

- A 16 bit program counter (PC)
- A 16 bit stack pointer (SP)
- Six 8-bit general purpose register arranged in pairs: BC, DE, HL.
- It requires a signal +5V power supply and operates at 3 MHZ single phase clock.
- It is enclosed with 40 pins DIP (Dual in line package).
- It has 74 operation codes with total 246 instructions.

# INSTRUCTION FORMAT

- An instruction is a binary pattern designed inside a μp to perform a specific function.
- The entire group of instructions, called the instruction set.
- Each instruction has two parts: one is the task to be performed, called the operation code (Op-code), and the other is data to be operated on, called the operand.
- Operand may include 8 bit or 16 bit data, and internal register, a memory location, or an 8 (or 16 bit) address.
- In some instructions, the operand is implicit.
- The 8085 instruction set is classified into 3 groups according to word or byte size.

# INSTRUCTION FORMAT

- 1- byte instructions
- 2- byte instructions
- 3- byte instructions

**One byte instructions**

- It includes the op-code and operand in the same byte. These instructions are stored in 8 bit binary format in memory; each requires one byte memory location.

e.g.

| Memory address | Op-code | Operand | Hex code | Description |
|---|---|---|---|---|
| 3000 | MOV | C,A | 4F H | Copy the contents of the Accumulator in register C |
| 3001 | ADD | B | 80 H | Add the contents of register B to the contents of the Accumulator |
| 3002 | CMA | | 2F H | Invert (compliment) each bit in the Accumulator |

# INSTRUCTION FORMAT

**Two byte instructions**

- In this type of instruction, first byte specifies the operation code and the second byte specified the operand.
- These instructions would require two memory locations each to store the binary codes.

e.g.

| Memory address | Op-code | Operand | Hex code | Description |
|---|---|---|---|---|
| 3100 | MVI | A,32 H | 3E H | Load an one byte data |
| 3101 | | | 32 H | (32H) in the accumulator |
| 8080 | MVI | B, F2 H | 06 H | Load an one byte data |
| 8081 | | | F2 H | (F2H) in reg. B |

# INSTRUCTION FORMAT

## Three byte instructions

- In 3 byte instructions, the first byte specifies the Op-code, and the following two bytes specify the 16 bit address or data. (Note: second byte is the low-order address and the third byte is the high-order address).
- These instructions would require 3 memory locations each to store the binary codes.

e.g.

| Memory address | Op-code | Operand | Hex code | Description |
|---|---|---|---|---|
| 3000 | LDA | 2050 H | 3A H | Load contents of memory |
| 3001 | | | 50 H | 2050H into accumulator |
| 3002 | | | 20 H | |
| | | | | |
| 8000 | JMP | 2085 H | C3 H | Transfer the program |
| 8001 | | | 85 H | sequence to memory |
| 8002 | | | 20 H | location 2085H |

# ADDRESS MODES OF 8085

- To perform any operation, we have to give the corresponding instructions to the microprocessor.
- In each instruction, we have to specify the following three things:
  - Operation to be performed.
  - Address of source of data.
  - Address of destination of result.
- The method by which the address of source of data or the address of destination of result is given in the instruction is called Addressing Mode.
- The term addressing mode refers to the way in which the operand of the instruction is specified.

# ADDRESS MODES OF 8085

## Types of Addressing Modes

- Intel 8085 uses the following addressing modes:
  - 1. Direct Addressing Mode
  - 2. Register Direct Addressing Mode
  - 3. Register Indirect Addressing Mode
  - 4. Immediate Addressing Mode
  - 5. Implied Addressing Mode

# ADDRESS MODES OF 8085

## Direct Addressing Mode

- Instructions using this mode specify the effective address as part of instruction.
- Instructions using this mode may contain 2 or 3 bytes, with first byte as the Op-code followed by 1 or 2 bytes or address of data.
- In this mode, the address of the operand is given in the instruction itself.

LDA 2500 H  Load the contents of memory location 2500 H in accumulator.

*LDA is the operation. 2500 H is the address of source. Accumulator is the destination.*

IN 41 H  Reads the data at port 41 H and store data at the accumulator

*IN is the operation. 41 H is the address of source. Accumulator is the destination.*

# ADDRESS MODES OF 8085

## Register Direct Addressing Mode

- In this mode, the operand is in general purpose register i.e. data is provided through registers.
  - MOV A, B  Move the contents of register B to A.

    MOV is the operation. B is the source of data. A is the destination.

## Register Indirect Addressing Mode

- In this mode, the address part of instruction specifies the memory location whose content is the address of the operand. In 8085 μp, wherever the instruction uses the HL pointer the address is called indirect addressing.
  - MOV A, M  Move data from memory location specified by H-L pair to accumulator.

    MOV is the operation. M is the memory location specified by H-L register pair. A is the destination.

    **LDAX B, STAX D**

# ADDRESS MODES OF 8085

## Immediate Addressing Mode

- In this mode, the operand is specified within the instruction itself. This mode of instructions uses first byte as the Op-code and following 1 or 2 byte data itself.
  - MVI A, 05 H  Move 05 H in accumulator.

    MVI is the operation. 05 H is the immediate data (source). A is the destination.
  - LXI H, 7A21 H  Loads register H with 7A H and register L with 21 H

    LXI is the operation. 7A21 H is the immediate data (Source). H & L registers are the destination.
- So in both cases, the actual data is part of the instruction, and hence called immediate addressing.

# ADDRESS MODES OF 8085

## Implied Addressing Mode

- If address of source of data as well as address of destination of result is fixed, then there is no need to give any operand along with the instruction.
- So, instructions of such mode don't have operands.
  - CMA Complement accumulator.

    CMA is the operation. A is the source. A is the destination.

# REGISTERS IN 8085

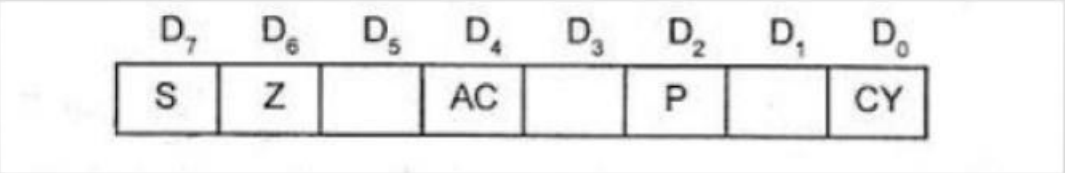
**Registers in 8085**

- General purpose Register
- Temporary Register
- Special purpose Register
- 16 bit Register

- **General purpose register**
  - B, C, D, E, H, L are general purpose register They can store 8 bit of data at a time
  - 16 bit of data can also be stored by using combination of BC, DE and HL
  - The general purpose register is available for the user to manipulate

# REGISTERS IN 8085

- **Temporary Register**
  - W and Z are each of 8 bit and are used by processor for its internal operation and not available to the user or programmer
- **Special purpose register**
  - Accumulator and flag are the two special purpose register each of 8 bit.
  - Accumulator is extensively used to store results of an arithmetic and logical operation It is also used in input output operations.
  - Flag register holds the properties of an arithmetical and logical operation



# REGISTERS IN 8085

- **16 bit register**
  - Stack/ stack pointer and program counter are 16 bit registers
  - Stack stores the value of flag and program counter during interrupt operation or subroutine call
  - Stack pointer holds the address of top of the stack
  - Program counter stores the address of next instruction to be executed

# PROGRAMMING MODEL OF 8085

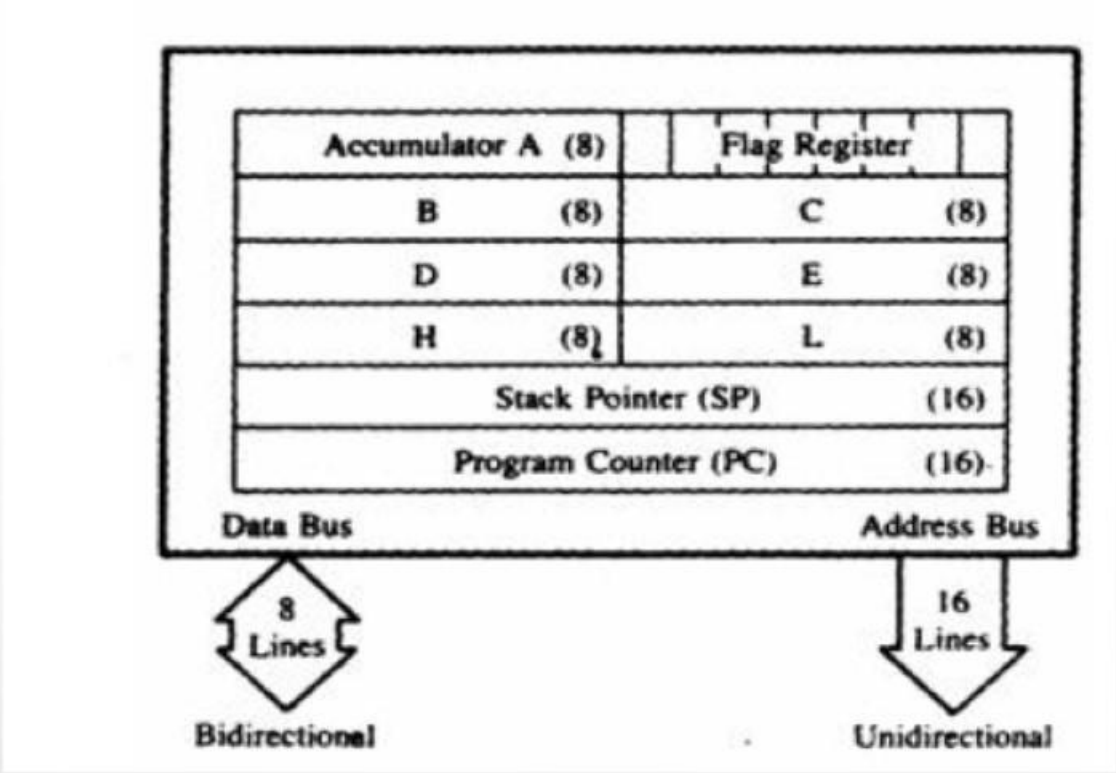



Fig: Programming model of 8085

Note: Explain Registers (Flags, Accumulator, Temporary Register and Register pair), Stack Pointer, Program Counter, Address and Data Bus

# Introduction to 8086 Microprocessor

- 8086 Microprocessor is an enhanced version of 8085 Microprocessor that was designed by Intel in 1978.
- It is a 16 bit Microprocessor having 20 address lines and 16 data lines that provides up to 1 MB storage.
- It consists of powerful instruction set, which provides operations like multiplication and division easily.

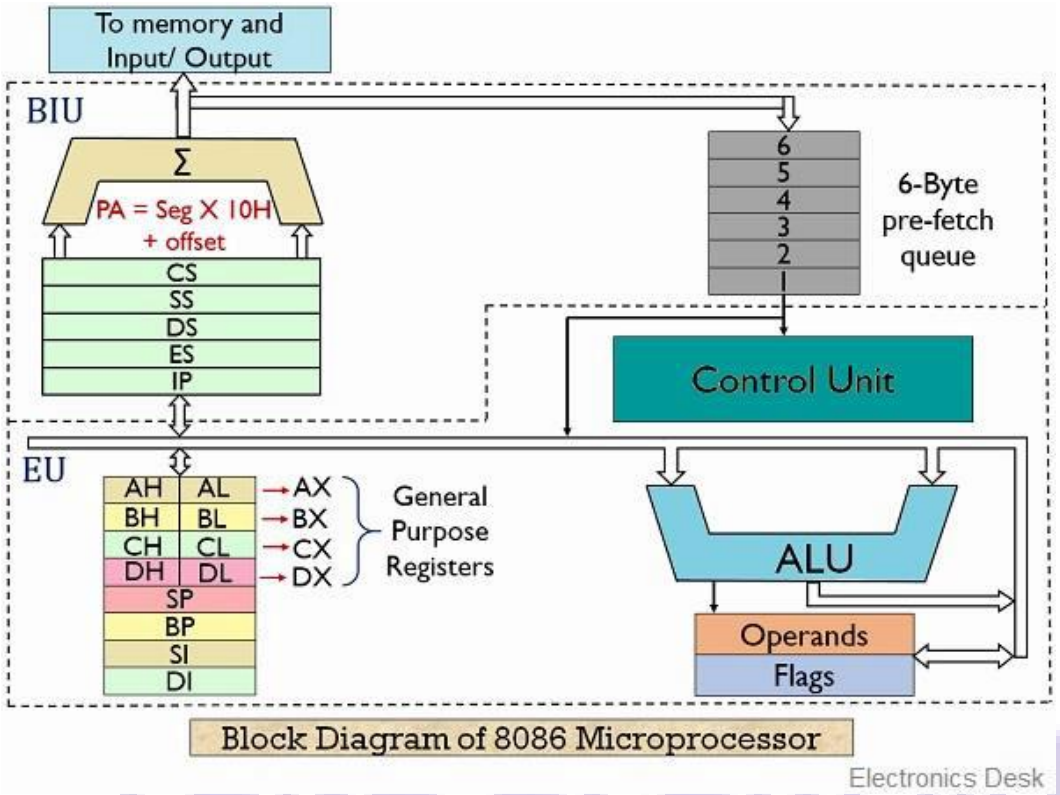
# Features 8086 Microprocessor

- It has an instruction queue, which is capable of storing six instruction bytes from the memory resulting in faster processing
- It was the first 16 bit processor having 16 bit ALU, 16 bit registers, internal data bus, and 16 bit external data bus resulting in faster processing
- It is available at different clock frequencies 5 MHz, 8 MHz and 10 MHz
- It uses two stages of pipelining, i e Fetch Stage and Execute Stage, which improves performance
- Fetch stage can prefetch up to 6 bytes of instructions and stores them in the queue
- Execute stage executes these instructions
- It has 256 vectored interrupts
- It consists of 29 000 transistors

# Comparison with 8085

- **Size:** 8085 is 8 bit microprocessor, whereas 8086 is 16 bit microprocessor
- **Address Bus :** 8085 has 16 bit address bus while 8086 has 20 bit address bus
- **Memory:** 8085 can access up to 64 KB, whereas 8086 can access up to 1 MB of memory
- **Instruction Queue:** 8085 doesn't have an instruction queue, whereas 8086 has an instruction queue
- **Pipelining:** 8085 doesn't support a pipelined architecture while 8086 supports a pipelined architecture
- **I/O:** 8085 can address $2^8$=256 I/O's, whereas 8086 can access $2^{16}$ =65 536 I/O's
- **Cost:** The cost of 8085 is low whereas that of 8086 is high

# Internal Architecture of 8086



# Execution unit (EU) and its components

- **Index register**
  - The two index registers SI (Source index) and DI (Destination Index) are used in indexed addressing. The instructions that process data strings use the SI and DI index register together with DS and ES respectively, in order to distinguish between the source and destination address.

- **Flag register**
  - The 8086 has nine 1 bit flags. Out of 9 six are status and three are control flags. The control bits in the flag register can be set or reset by the programmer.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | X | X | X | O | D | I | T | S | Z | X | Ac | X | P | X | Cy |

# Execution unit (EU) and its components

**Control flags:**

- **D-Direction Flag**
  - This is used by string manipulation instructions. If this flag bit is '0' , the string is processed beginning from the lowest address to the higher address, i.e. auto incrementing mode otherwise the string is processed from the highest address towards the lowest address, i.e. autodecrementing mode.
- **I-Interrupt flag**
  - If this flag is set the maskable interrupts are recognized by the CPU, otherwise they are ignored.
- **T- Trap flag**
  - If this flag is set the processor enters the single step execution mode. In other words, a trap interrupt is generated after execution of each instruction. The processor executes the current instruction and the control is transferred to the Trap interrupt service routine.

# Execution unit (EU) and its components

**Status Flags:**

- **O- Overflow flag**
  - This flag is set if an arithmetic overflow occurs, i.e. if the result of a signed operation is large enough to be accommodated in a destination register.
- **S - Sign flag**
  - This flag is set when the result of any computation is negative. For signed computations, the sign flag equals the MSB of the result.
- **Z- Zero**
  - This flag is set when the result of the computation is or comparison performed by the previous instruction is zero. 1 for zero result, 0 fir nonzero result

# Execution unit (EU) and its components

- **$A_{c-}$ Auxiliary Carry**
  - This is set if there is a carry from the lowest nibble, i.e. bit three during the addition or borrow for the lowest nibble i.e. bit three, during subtraction.
- **P- Parity flag**
  - This flag is set to 1 if the lower byte of the result contains even number of 1s otherwise reset.
- **$C_y$-Carry flag:**
  - This flag is set when there is a carry out of MSB in case of addition or a borrow in case of subtraction.

# Segment and offset address

- Segments are special areas defined in a program for containing the code, data and stack.
- A segment begins on a paragraph boundary.
- A segment register is of 16 bits in size and contains the starting address of a segment.
- A segment begins on a paragraph boundary, which is an address divisible by decimal 16 or hex 10.
- Consider a DS that begins at location 038EOH. In all cases, the rightmost hex digit is zero, the computer designers decided that it would be unnecessary to store the zero the zero digit in the segment register.
- Thus 038E0H is stores in register as 038EH.

# Segment and offset address

- The distance in bytes from the segment address to another location within the segment is expressed as an offset or displacement.
- Suppose the offset of 0032H for above example of data segment. Processor combines the address of the data segment with the offset as:
- SA: OA (segment address: offset address)
- 038E: 0032 H = 038E * 10 +0032 = 038E0 + 0032
- Physical address = 03912H

# Addressing modes in 8086

- Addressing modes describe types of operands and the way in which they are accessed for executing an instruction.
- An operand address provides source of data for an instruction to process an instruction to process.
- An instruction may have from zero to two operands. For two operands first is destination and second is source operand.
- The basic modes of addressing are register, immediate and memory which are described below.

# Addressing modes in 8086

**Register Addressing:**

- For this mode, a register may contain source operand, destination operand or both.
- E.g. MOV AH, BL MOV DX, CX

**Immediate Addressing**

- In this type of addressing, immediate data is a part of instruction, and appears in the form of successive byte or bytes.
- This mode contains a constant value or an expression.
- E.g. MOV AH, 35H MOV BX, 7A25H

# Addressing modes in 8086

**Direct memory addressing:**

- In this type of addressing mode, a 16-bit memory address (offset) is directly specified in the instruction as a part of it.
- One of the operand is the direct memory and other operand is the register.
- E.g. ADD AX, [5000H]
- Note: Here data resides in a memory location in the data segment, whose effective address may be computed using 5000H as the Offset address and content of DS as segment address.
- The effective address, here, is 10H*DS + 5000H.

# Addressing modes in 8086

**Direct offset addressing**

- In this addressing, a variation of direct addressing uses arithmetic operators to modify an address.
- E.g. ARR DB 15, 17, 18, 21
- MOV AL, ARR [2] ; MOV AL, 18
- ADD BH, ARR+3 ; ADD BH, 21

# Addressing modes in 8086

**Indirect memory addressing:**

- Indirect addressing takes advantage of computer's capability for segment: offset addressing.
- The registers used for this purpose are base register (BX and BP) and index register (DI and SI) E.g. MOV [BX],AL

  ADD CX, [SI]

# Addressing modes in 8086

**Base displacement addressing:**

- This addressing mode also uses base registers (BX and BP) and index register (SI and DI), but combined with a displacement (a number or offset value) to form an effective address.
- E.g. MOV BX, OFFSET ARR ; LEA BX, ARR
- MOV AL, [BX +2]
- ADD TBL [BX], CL ;TBL [BX] [BX + TBL] e.g. [BX + 4]

# Addressing modes in 8086

**Base index addressing:**

- This addressing mode combines a base registers (BX or BP) with an index register (SI or DI) to form an effective address.
- E.g. MOV AX, [BX +SI]
- ADD [BX+DI], CL

**Base index with displacement addressing**

- This addressing mode, a variation on base- index combines a base register, an index register, and a displacement to form an effective address.
- E.g. MOV AL, [BX+SI+2]
- ADD TBL [BX +SI], CH

# Addressing modes in 8086

**String addressing:**

- This mode uses index registers, where SI is used to point to the first byte or word of the source string and DI is used to point to the first byte or word of the destination string, when string instruction is executed.
- The SI or DI is automatically incremented or decremented to point to the next byte or word depending on the direction flag (DF).
- E.g. MOVS, MOVSB, MOVSW

# Coding in assembly language

- Assembly language programming has taken its place in between the machine language (low level) and the high level language.
  - High level language's one statement may generate many machine instructions.
  - Low level language consists of either binary or hexadecimal operation. One symbolic statement generates one machine level instructions.

# Coding in assembly language

- **Advantage of ALP**
  - They generate small and compact execution module.
  - They have more control over hardware.
  - They generate executable module and run faster.
- **Disadvantages of ALP**
  - Machine dependent.
  - Lengthy code
  - Error prone (likely to generate errors).

# Coding in assembly language

| S.NO | HIGH LEVEL LANGUAGE | LOW LEVEL LANGUAGE |
|---|---|---|
| 1. | It is programmer friendly language. | It is a machine friendly language. |
| 2. | High level language is less memory efficient. | Low level language is high memory efficient. |
| 3. | It is easy to understand. | It is tough to understand. |
| 4. | It is simple to debug. | It is complex to debug comparatively. |
| 5. | It is simple to maintain. | It is complex to maintain comparatively. |
| 6. | It is portable. | It is non-portable. |
| 7. | It can run on any platform. | It is machine-dependent. |
| 8. | It needs compiler or interpreter for translation. | It needs assembler for translation. |
| 9. | It is used widely for programming. | It is not commonly used now-a-days in programming. |

# assembly language features

- The main features of ALP are program comments, reserved words, identifies, statements and directives which provide the basic rules and framework for the language.

**Program comments:**

- The use of comments throughout a program can improve its clarity.
- It starts with semicolon (;) and terminates with a new line.
- E.g. ADD AX, BX ; Adds AX & BX

# assembly language features

## Reserved words

- Certain names in assembly language are reserved for their own purpose to be used only under special conditions and includes
  - **Instructions** : Such as MOV and ADD (operations to execute)
  - **Directives**: Such as END, SEGMENT (information to assembler)
  - **Operators**: Such as FAR, SIZE
  - **Predefined symbols**: such as @DATA, @ MODEL

# assembly language features

**Identifiers:**

- An identifier (or symbol) is a name that applies to an item in the program that expects to reference.
- Two types of identifiers are Name and Label.
  - Name refers to the address of a data item such as NUM1 DB 5, COUNT DB 0
  - Label refers to the address of an instruction.
- E. g: MAIN PROC FAR
- L1: ADD BL, 73

# assembly language features

**Statements:**

- ALP consists of a set of statements with two types
- **Instructions**, e. g. MOV, ADD
- **Directives**, e. g. define a data item

| | Identifiers | operation | operand | comment |
|---|---|---|---|---|
| Directive: | COUNT | DB | 1 | ; initialize count |
| Instruction: | L30: | MOV | AX, 0 | ; assign AX with 0 |

# assembly language features

**Directives:**

- The directives are the number of statements that enables us to control the way in which the source program assembles and lists.
- These statements called directives act only during the assembly of program and generate no machine-executable code. The different types of directives are:

# assembly language features

## 1) The page and title listing directives:

- The page and title directives help to control the format of a listing of an assembled program.
- This is their only purpose and they have no effect on subsequent execution of the program.
- The page directive defines the maximum number of lines to list as a page and the maximum number of characters as a line.
- PAGE [Length] [Width]
- Default : Page [50][80]
- TITLE gives title and place the title on second line of each page of the program.
- TITLE text [comment]

# assembly language features

## 2) SEGMENT directive

- It gives the start of a segment for stack, data and code.

  Seg-name  Segment  [align] [combine][‘class’]

  Seg-name  ENDS

- Segment name must be present, must be unique and must follow assembly language naming conventions.
- An ENDS statement indicates the end of the segment.
- Align option indicates the boundary on which the segment is to begin; PARA is used to align the segment on paragraph boundary.
- Combine option indicates whether to combine the segment with other segments when they are linked after assembly. STACK, COMMON, PUBLIC, etc are combine types.
- Class option is used to group related segments when linking. The class code for code segment, stack for stack segment and data for data segment.

# assembly language features

## 3) PROC Directives

- The code segment contains the executable code for a program, which consists of one or more procedures, defined initially with the PROC directives and ended with the ENDP directive.
- PROC - name PROC [FAR/NEAR]
- ……
- PROC – name ENDP
- FAR is used for the first executing procedure and rest procedures call will be NEAR.
- Procedure should be within segment.

# assembly language features

## 4) END Directive

- An END directive ends the entire program and appears as the last statement.
- ENDS directive ends a segment and ENDP directive ends a procedure.
- END PROC-Name

## 5) ASSUME Directive

- An .EXE program uses the SS register to address the stack, DS to address the data segment and CS to address the code segment.
- Used in conventional full segment directives only.
- Assume directive is used to tell the assembler the purpose of each segment in the program.
- Assume SS: Stack name, DS: Data Segname CS: code segname

# assembly language features

## 6) Processor directive

- Most assemblers assume that the source program is to run on a basic 8086 level computer.
- Processor directive is used to notify the assembler that the instructions or features introduced by the other processors are used in the program.
- E.g. .386 - program for 386 protected mode.

# assembly language features

## 7) Dn Directive (Defining data types)

- Assembly language has directives to define data syntax: [name] Dn expression
- The Dn directive can be any one of the following:
- DB Define byte 1 byte
- DW Define word 2 bytes
- DD Define double 4 bytes
- DF defined farword 6 bytes
- DQ Define quadword 8 bytes
- DT Define 10 bytes 10 bytes

```
VAL1 DB 25
ARR    DB    21, 23, 27, 53
MOV    AL, ARR [2]          or
MOV AL, ARR + 2 ;           Moves 27 to AL register
```

# assembly language features

## 8) The EQU directive

- It can be used to assign a name to constants.
- E.g. FACTOR EQU 12

## 9) DUP Directive

- It can be used to initialize several locations to zero. e. g. SUM DW 4 DUP(0)
- Reserves four words starting at the offset sum in DS and initializes them to Zero.
- Also used to reserve several locations that need not be initialized. In this case (?) is used with DUP directives. E. g. PRICE DB 100 DUP(?)
- Reserves 100 bytes of uninitialized data space to an offset PRICE.

# assembly language features

## 10) DOSSEG

- There is a standard order for placing the stack, code and data segments One can place the segments in any order as well.
- This directives tells the assembler to place the segments in standard order

## 11) MODEL

- This directive determines the size of each segment .All of the program models except tiny result in the creation of exe program. The tiny model creates a com program.

# assembly language features

| Model | Description |
|---|---|
| TINY | Code and data together may not be greater than 64K |
| SMALL | Neither code nor data may be greater than 64K |
| MEDIUM | Only the code may be greater than 64K |
| COMPACT | Only the data may be greater than 64K |
| LARGE | Both code and data may be greater than 64K |
| HUGE | All available memory may be used for code and data |

# assembly language features

**ORG:**

- It changes the starting offset address of the data.

**STARTUP**

- It generates the instruction to initialize the segment register

**EXIT**

- It generates the INT 21H function, 4CH instruction for exiting program.

**Global:**

- The GLOBAL directive is used to make the symbol available to the other modules.

**EXTERN:**

- It is used to tell the assembler that the name or label following the directive are in some other assembly module

# Microprocessor– SET A

1. In 8085 microprocessor, how many interrupts are maskable.

a. Two
b. Three
● Four
d. Five

2. Which stack is used in 8085 microprocessors?

a. FIFO
b. FILO
● LIFO
d. LILO

3. In the instruction of the 8085 microprocessor, how many bytes are present?

a. One or two
● One, two or three
c. One only
d. Two or three

4. Which one of the following addressing technique is not used in 8085 microprocessor?

a. Register
b. Immediate
c. Register indirect
● Relative

5. Which one of the following register of 8085 microprocessor is not a part of the programming model?

a. Instruction register
b. Memory address register
● Status register
d. Temporary data register

6. The program counter in 8085 microprocessor is a 16-bit register, because

a. It counts 16 bits at a time
● There are 16 address times
c. It facilitates the users storing 16-bit data temporarily
d. It has to fetch two 8-bit data at a time.

# Microprocessor– SET A

7. A direct memory access (DMA) transfer replies

a. Direct transfer of data between memory and accumulator
b. Direct transfer of data between memory and I/O devices without the use of microprocessor
c. Transfer of data exclusively within microprocessor registers
d. A fast transfer of data between microprocessor and I/O devices

4. The microprocessor of a computer can operate on any information if it is present in _____ only.

a) Program Counter
b) Flag
c) Main Memory
d) Secondary Memory

9. In a Microprocessor, the address of the new next instruction to be executed is stored in

a. Stack pointer
b. address latch
c. Program counter
d. General purpose register

12. Which one of the following statements is correct regarding the instruction CMP A ?

a. compare accumulator with register A
b. compare accumulator with memory
c. compare accumulator with register H
d. This instruction does not exist

13. The instruction JNC 16-bit refers to jump to 16-bit address if ?

a. sign flag is set
b. carry flag is reset
c. zero flag is set
d. parity flag is reset

15. XCHG instruction of 8085 exchanges the content of ?

a. top of stack with contents of register pair
b. BC and DE register pairs
c. HL and DE register pairs
d. None of the above

# Microprocessor– SET A

16. Direction flag is used with

a. string instructions
b. stack instructions
c. arithmetic instructions
d. branch instructions

18. Following is a 16-bit register for 8085 microprocessor

a. Stack pointer
b. Accumulator
c. Register B
d. Register C

19. The register which holds the information about the nature of results of arithmetic of logic operations is called as

a. Accumulator
b. Condition code register
c. Flag register
d. Process status registers

20. When referring to instruction words, a mnemonic is

a. a short abbreviation for the operand address.
b. a short abbreviation for the operation to be performed.
c. a short abbreviation for the data word stored at the operand address.
d. Shorthand for machine language.

**12). Which of the following interrupt is non-vectored in 8085?**

- RST 5.5
- TRAP
- INTR
- RST 7

**11). What kind of interrupts are RST0 to RST7 in the 8085 microprocessor?**

- Logical interrupts
- Hardware interrupts
- Conditional interrupts
- Software interrupts

# Microprocessor– SET A

10. Which of the following is true about microprocessors?
a) It has an internal memory
b) It has interfacing circuits
c) It contains ALU, CU, and registers
d) It uses Harvard architecture

11. Which of the following is the correct sequence of operations in a microprocessor?
a) Opcode fetch, memory read, memory write, I/O read, I/O write
b) Opcode fetch, memory write, memory read, I/O read, I/O write
c) I/O read, opcode fetch, memory read, memory write, I/O write
d) I/O read, opcode fetch, memory write, memory read, I/O write

12. The _____ directive instructs the assembler to begin memory allocation for a segment/block/code from the stated address.
a) GROUP
b) OFFSET
c) ORG
d) LABEL

14. Which of the following is not a property of TRAP interrupt in microprocessor?
a) It is a non-maskable interrupt
b) It is of highest priority
c) It uses edge-triggered signal
d) It is a vectored interrupt

16. Which of the following flag is used to mask INTR interrupt?
a) zero flag
b) auxiliary carry flag flag
c) interrupt flag
d) sign flag

18. Which of the following circuit is used as a special signal to demultiplex the address bus and data bus?
a) Priority Encoder
b) Decoder
c) Address Latch Enable
d) Demultiplexer

# Microprocessor– SET A

19. How many flip-flops are there in a flag register of 8085 microprocessor?
a) 4
b) 5
c) 7
d) 10

20. Which of the following flag condition is used for BCD arithmetic operations in microprocessor?
a) Sign flag
b) Auxiliary carry flag
c) Parity flag
d) Zero flag

21. Whenever a non-maskable interrupt occurs in 8085 microprocessor, which of the following data line contains the data?
a) 2C H
b) 3C H
c) 36 H
d) 24 H

22. What does a microprocessor understand after decoding opcode?
a) Perform ALU operation
b) Go to memory
c) Length of the instruction and number of operations
d) Go to the output device



23. How many address lines are present in 8086 microprocessor?
a) 16
b) 20
c) 32
d) 40

24. Which of the following is not a status flag in microprocessor?
a) Overflow flag
b) Direction flag
c) Interrupt flag
d) Index flag

# Microprocessor– SET A

25. Which of the following is not a condition flag?
a) Trap flag
b) Auxiliary carry flag
c) Parity flag
d) Zero flag

26. Which of the following register is not used in opcode fetch operations?
a) Program counter
b) Memory address register
c) Memory data register
d) Flag register

27. A memory connected to a microprocessor has 20 address lines and 16 data lines. What will be the memory capacity?
a) 8 KB
b) 2 MB
c) 16 MB
d) 64 KB

28. What is the word length of the Pentium-II microprocessor?
a) 8-bit
b) 32-bit
c) 64-bit
d) 16-bit



29. Which of the following is not true about 8085 microprocessor?
a) It is an 8-bit microprocessor
b) It is a 40 pin DIP chip
c) It is manufactured using PMOS technology
d) It has 16 address lines

31. Which of the following is true?
a) Every instruction has two parts i.e. opcode and operands
b) MOV B, C is a two-byte instruction
c) MVI A, 90H is a three-byte instruction
d) Maximum number of T-states possible for the execution of an instruction is 16

# Microprocessor– SET A

34. If a 90 GB memory has to be connected to a microprocessor, minimum how many address lines are required?
a) 36
b) 39
c) 32
d) 37

35. Which of the following is a software interrupt?
a) TRAP
b) INTR
c) RST-6.5
d) RST-5

37. Which of the following is true about stack pointer?
a) Stack pointer contains the address of the top of the stack memory
b) Stack pointer is an 8-bit register
c) Stack pointer stores data permanently
d) Stack pointer is initialized after stack operation

38. How many address lines are required to connect a 4 KB RAM to a microprocessor?
a) 10
b) 16
c) 12
d) 20

39. Which of the following is true about MOV A, B instruction?
a) It means move the content of register A to register B
b) It uses immediate addressing mode
c) It doesn't affect the flag register
d) It is a 2-byte instruction

40. Which of the following is false about LDA instruction?
a) It is a 3-byte instruction
b) It uses indirect addressing mode
c) It has 13 T-states
d) It doesn't affect any flags

# Microprocessor– SET A

42. DAA instruction is used to perform which type of addition?

a) BCD addition
b) Excess-3 addition
c) Binary addition
d) Octal addition

44. Suppose registers 'A' and 'B' contain 50H and 40H respectively. After instruction MOV A, B, what will be the contents of registers A and B?

a) 40H, 40H
b) 50H, 40H
c) 50H, 50H
d) 60H, 40H

1. _____ converts the programs written in assembly language into machine instructions.

a) Machine compiler
b) Interpreter
c) Assembler
d) Converter

2. The instructions like MOV or ADD are called as _____

a) OP-Code
b) Operators
c) Commands
d) None of the mentioned

5. The assembler directive EQU, when used in the instruction: Sum EQU 200 does _____

a) Finds the first occurrence of Sum and assigns value 200 to it
b) Replaces every occurrence of Sum with 200
c) Re-assigns the address of Sum by adding 200 to its original address
d) Assigns 200 bytes of memory starting the location of Sum

15. To overcome the problems of the assembler in dealing with branching code we use _____

a) Interpreter
b) Debugger
c) Op-Assembler
d) Two-pass assembler

**2.5 Microprocessor system**: Memory Device Classification and Hierarchy, Interfacing I/O and Memory Parallel Interface. Introduction to Programmable Peripheral Interface (PPI), Serial Interface, Synchronous and Asynchronous Transmission, Serial Interface Standards. Introduction to Direct Memory Access (DMA) and DMA Controllers. (AExE0205)

# MICROPROCESSOR SYSTEM

- A microcomputer consists of a set of components or modules of three basic types CPU memory and I/O units which communicate with each other.

# PIN CONFIGURATION OF 8085

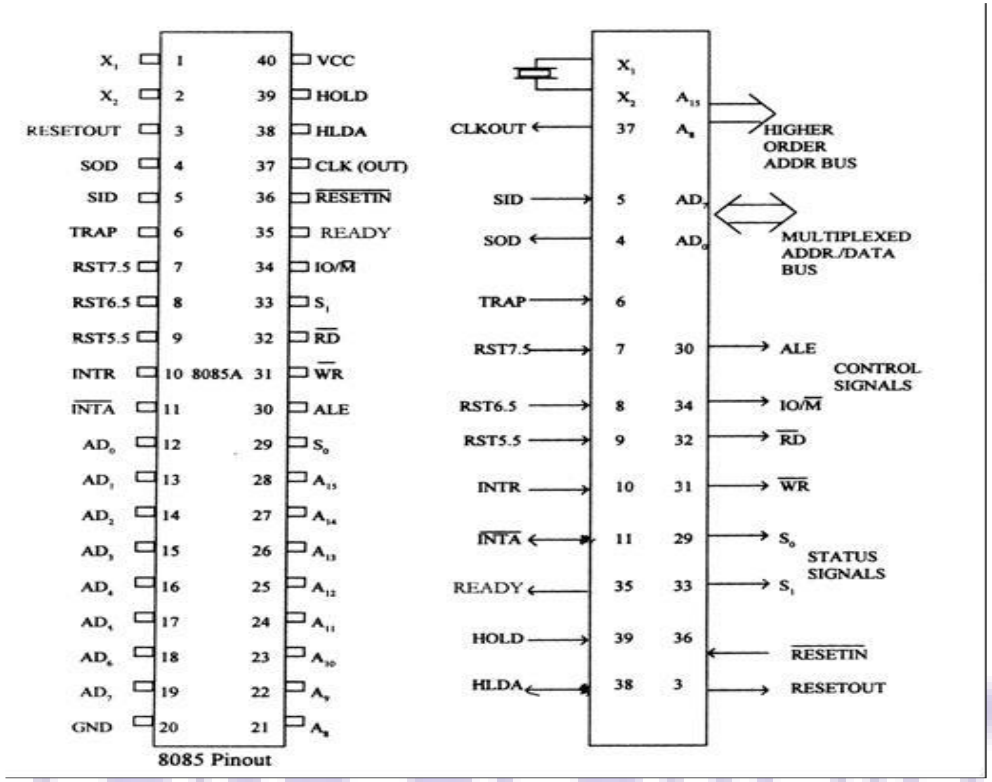



Fig (a) - Pin Diagram of 8085 & Fig(b) - logical schematic of Pin diagram

# PIN CONFIGURATION OF 8085

- The microprocessor is a clock-driven semiconductor device consisting of electronic logic circuits manufactured by using either a large-scale integration (LSI) or very-large-scale integration (VLSI) technique.
- The microprocessor is capable of performing various computing functions and making decisions to change the sequence of program execution.
- In large computers, a CPU implemented on one or more circuit boards performs these computing functions.
- The microprocessor is in many ways similar to the CPU, but includes the logic circuitry, including the control unit, on one chip.

# PIN CONFIGURATION OF 8085

- The microprocessor can be divided into three segments for the sake clarity, arithmetic/logic unit (ALU), register array, and control unit.
- 8085 is a 40 pin IC, DIP package. The signals from the pins can be grouped as follows
  - 1. Power supply and clock signals
  - 2. Address bus
  - 3. Data bus
  - 4. Control and status signals
  - 5. Interrupts and externally initiated signals
  - 6. Serial I/O ports

# PIN CONFIGURATION OF 8085

**Power supply and Clock frequency signals:**

- **Vcc** : + 5 volt power supply
- **Vss :** Ground
- **X1, X2** : Crystal or R/C network or LC network connections to set the frequency of internal clock generator.
- The frequency is internally divided by two. Since the basic operating timing frequency is 3 MHz, a 6 MHz crystal is connected externally.
- **CLK (output) :**Clock Output is used as the system clock for peripheral and devices interfaced with the microprocessor.

# PIN CONFIGURATION OF 8085

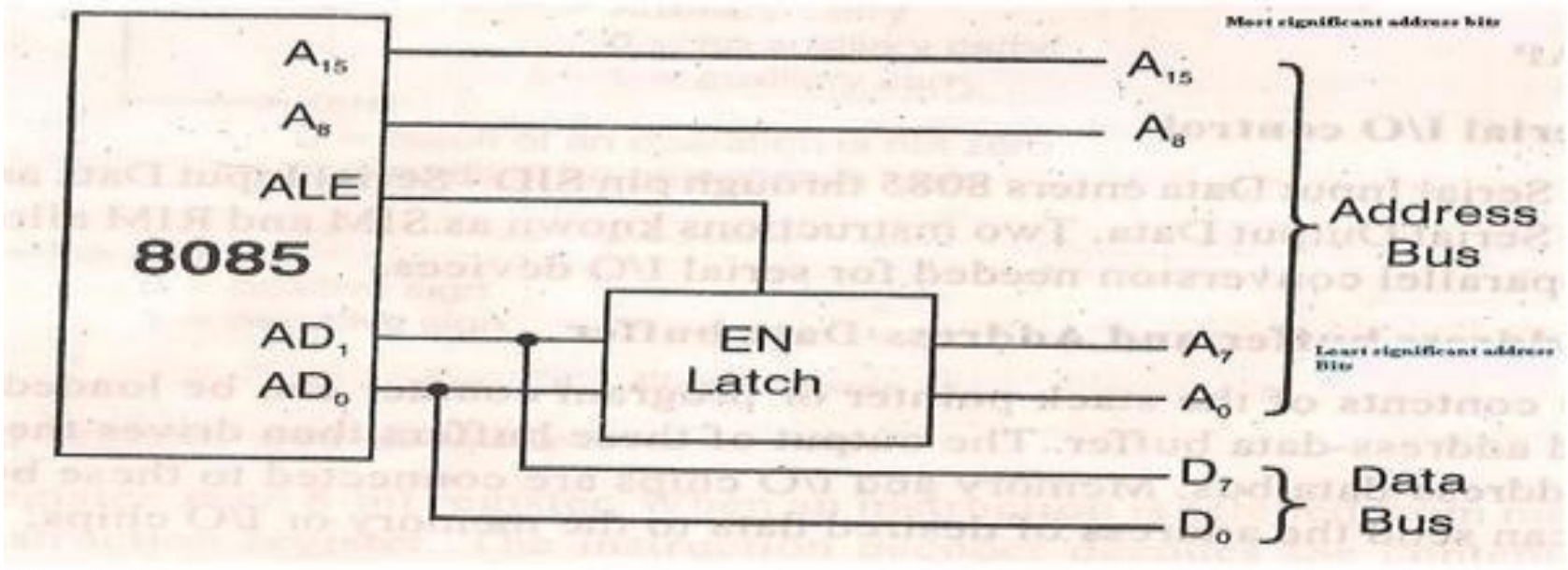
**2. Address Bus:**

- **A8 - A15**
- It carries the most significant 8 bits of the memory address or the 8 bits of the I/O address.

**3. Multiplexed Address / Data Bus:**

- **AD0 - AD7**
- These multiplexed set of lines used to carry the lower order 8 bit address as well as data bus.
- During the opcode fetch operation, in the first clock cycle, the lines deliver the lower order address A0 - A7.
- In the subsequent IO / memory, read / write clock cycle the lines are used as data bus.
- The CPU may read or write out data through these lines.

# PIN CONFIGURATION OF 8085



# PIN CONFIGURATION OF 8085

**Control and Status signals:**

- These signals include two control signals (RD & WR) three status signals (IO/M, S1 and So) to identify the nature of the operation and one special signal (ALE) ti indicate the beginning of the operations.
- **ALE (output) -** Address Latch Enable.
  - This signal helps to capture the lower order address presented on the multiplexed address / data bus. When it is the pulse, 8085 begins an operation. It generates AD0 - AD7 as the separate set of address lines A0 –A7.
- **RD (active low)** - Read memory or IO device.
  - This indicates that the selected memory location or I/O device is to be read and that the data bus is ready for accepting data from the memory or I/O device.

# PIN CONFIGURATION OF 8085

- **WR (active low)** - Write memory or IO device.
  - This indicates that the data on the data bus is to be written into the selected memory location or I/O device.
- **IO/M' (output)** - Select memory or an IO device.
  - This status signal indicates that the read / write operation relates to whether the memory or I/O device.
  - It goes high to indicate an I/O operation.
  - It goes low for memory operations.

# PIN CONFIGURATION OF 8085

**Status Signals:**

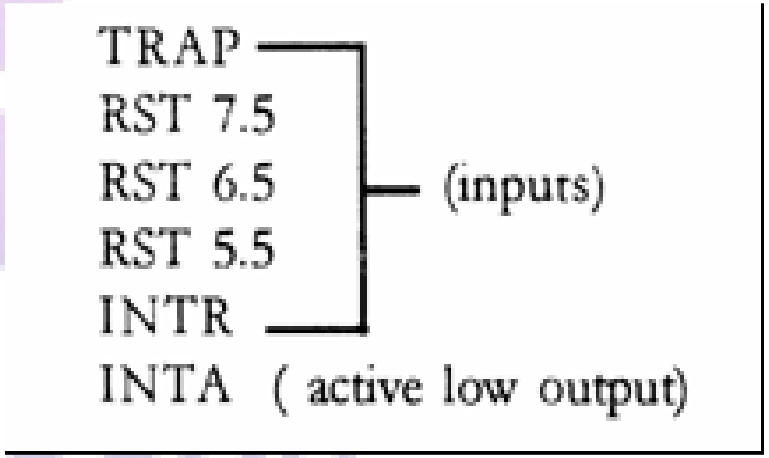
- It is used to know the type of current operation of the microprocessor.

| IO/M(Active Low) | S1 | S2 | Data Bus Status (Output) |
|---|---|---|---|
| 0 | 0 | 0 | Halt |
| 0 | 0 | 1 | Memory WRITE |
| 0 | 1 | 0 | Memory READ |
| 1 | 0 | 1 | IO WRITE |
| 1 | 1 | 0 | IO READ |
| 0 | 1 | 1 | Opcode fetch |
| 1 | 1 | 1 | Interrupt acknowledge |

# PIN CONFIGURATION OF 8085

**Interrupts and Externally initiated operations:**

- They are the signals initiated by an external device to request the microprocessor to do a particular task or work.
- There are five hardware interrupts called,




- On receipt of an interrupt, the microprocessor acknowledges the interrupt by the active low INTA (Interrupt Acknowledge) signal.

# PIN CONFIGURATION OF 8085

- **Hold (Input)**
  - This indicates peripheral controller requesting the bus.
- **HLDA (Output)**
  - This indicates the acknowledgement for the Hold request.
- **READY (Input)**
  - It is used to delay the microprocessor read and write cycles until a slow responding peripheral is ready to send or accept data.
  - Memory and I/O devices will have slower response compared to microprocessors.
  - Before completing the present job such a slow peripheral may not be able to handle further data or control signal from CPU.
  - The processor sets the READY signal after completing the present job to access the data.
  - The microprocessor enters into WAIT state while the READY pin is disabled.

# PIN CONFIGURATION OF 8085

- **Reset In (input, active low)**
  - This signal is used to reset the microprocessor.
  - The program counter inside the microprocessor is set to zero.
  - The buses are tri-stated.
- **Reset Out (Output)**
  - It indicates CPU is being reset.
  - Used to reset all the connected devices when the microprocessor is reset.

**Single Bit Serial I/O ports:**

- **SID (input) -** Serial input data line
- **SOD (output) -** Serial output data line
- These signals are used for serial communication.

# Bus structure

- A microcomputer consists of a set of components or modules of three basic types CPU memory and I/O units which communicate with each other.
- A bus is a communication pathway between two or more such components.
- A bus actually consists of multiple communication pathway or lines. Each line is capable of transmitting signals representing binary 1 and 0.
- Several lines of the bus can be used to transmit binary data simultaneously.
- The bus that connects major microcomputer components such as CPU, memory or I/O is called the system bus.
- System bus consists of number of separate lines. Each line assigned a particular function.
- Fundamentally in any system bus the lines can be classified into three group buses.

# Bus structure

**Data Bus:**

- Data bus provides the path for monitoring data between the system modules. The bus has various numbers of separate lines like 8, 16, 32, or 64. Which referred as the width of data bus .These number represents the no. of bits they can carry because each carry 1 bit.

**Address Bus:**

- Each Lines of address bus are used to designate the source or destination of the data on data bus.
- For example, if the CPU requires reading a word (8, 16, 32) bits of data from memory, it puts the address of desired word on address bus. The address bus is also used to address I/O ports. Bus width determines the total memory the up can handle.

# Bus structure

**Control Bus:**

- The control bus is a group of lines used to control the access to control signals and the use of the data and address bus. The control signals transmit both command and timing information between the system modules. The timing signals indicate the validity of data and address information, where as command signals specify operations to be performed. Some of the control signals are:
- **Memory Write (MEMW):** It causes data on the bus to be loaded in to the address location.
- **Memory Read (MEMR):** It causes data from the addressed location to be placed on the data bus.
- **I/O Write (IOW):** It causes the data on the bus to be output to the addressed I/O port.
- **I/O Read (IOR):** It causes the data from the addressed I/O port to be placed on the bus.
- **Transfer Acknowledge:** This signal indicates that data have been accepted from or placed on the bus.

# Bus structure

- **Bus Request:** It is used to indicate that a module wants to gain control of the bus.
- **Bus Grant**: It indicates that a requesting module has been granted for the control of bus.
- **Interrupt Request**: It indicates that an interrupt has been pending.
- **Interrupt Acknowledge**: it indicates that the pending interrupt has been recognized.

# Bus types

**Synchronous Bus:**

- In a synchronous bus, The Occurrence of the events on the bus is determined by a clock .
- The clock transmits a regular sequence of 0's & 1's of equal duration. All the events start at beginning of the clock cycle.

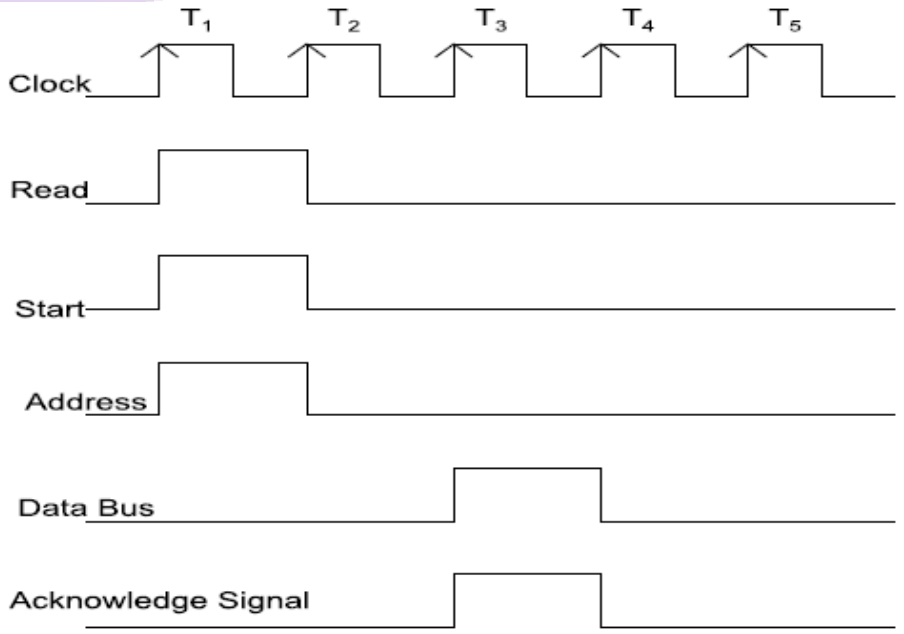



Fig: Synchronous Read Operation

# Bus types

- Here the CPU issues a START signal to indicate the presence of address and control information on the bus.
- Then it issues the memory read signal and places the memory address on the address bus.
- The addressed memory module recognizes the address and after a delay of one clock cycle it places the data and acknowledgment signal on the buses.
- In synchronous bus, all devices are tied to a fixed rate, and hence the system can not take advantage of device performance but it is easy to implement.

# Bus types

## Asynchronous Bus:

- In an asynchronous bus, the timing is maintained in such way that occurrence of one event on the bus follows and depends on the occurrence of previous event.

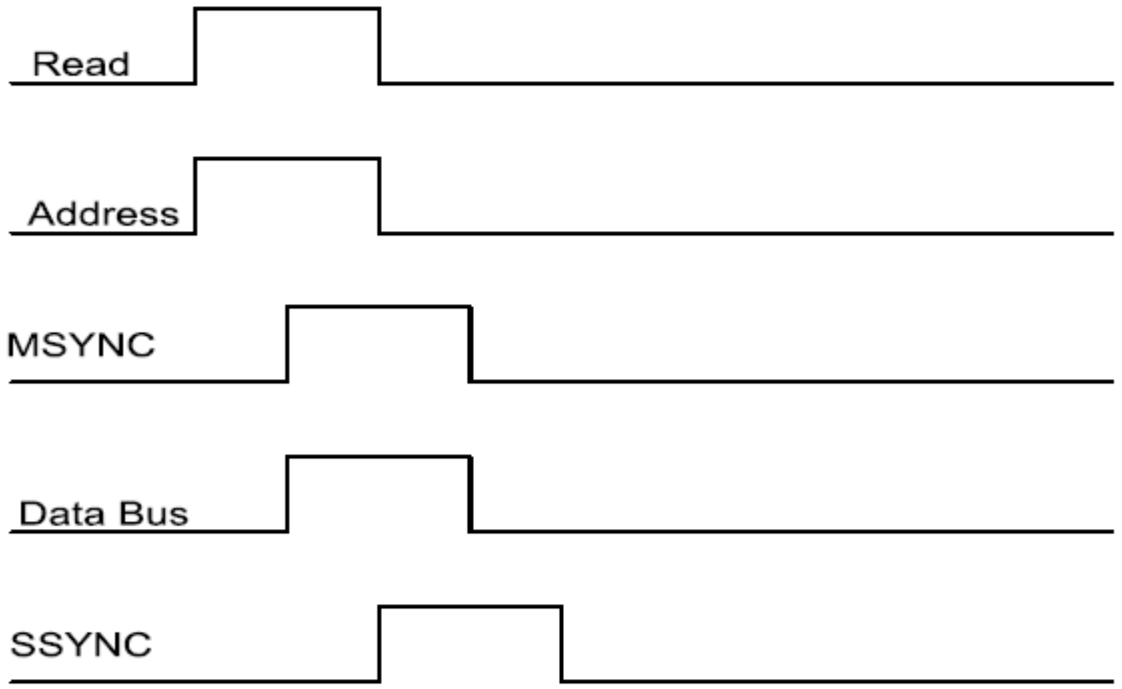


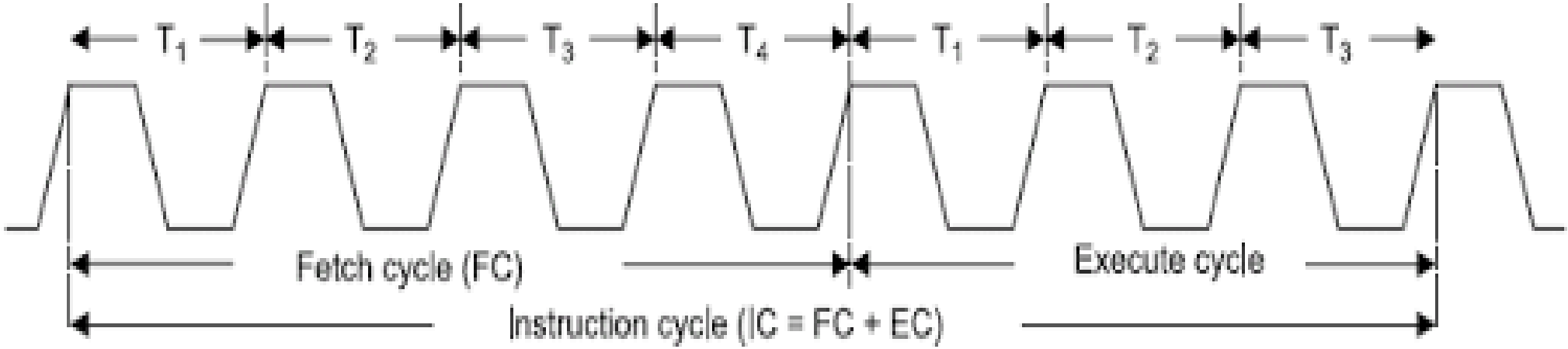

Fig: Asynchronous Read Operation

# Bus types

- Here the CPU places Memory Read (Control) and address signals on the bus.
- Then it issues master synchronous signal (MSYNC) to indicate the presence of valid address and control signals on the bus.
- The addressed memory module responds with the data and the slave synchronous signal (SSYNC)

# Timing diagram

- It is a graphical representation. It represents the execution time taken by each instruction in a graphical format. The execution time is represented in T-state.



# Instruction cycle

- It is defined as the time required to complete the execution of an instruction.
- The necessary steps that the CPU carries out to fetch an instruction and necessary data from the memory and to execute it constitute an instruction cycle.
- An instruction cycle consists of fetch cycle and execute cycle.
- In fetch cycle CPU fetches op-code from the memory.
- The necessary steps which are carried out to get data if any from the memory and to perform the specific operation specified in instruction constitute an execute cycle.
- The total time required to execute an instruction is given by IC=FC+EC
- The 8085 consists of 1-5 machine cycle or operation.

# fetch cycle

- The first byte of an instruction is its op-code.
- The content of the program counter, which is the address of the memory location where op-code is available, is send to the memory.
- The memory places the op-code on the data bus so as to transfer it to CPU.
- The entire process takes 3 clock cycle and then the instruction is decoded in next one clock cycle.

# Execute cycle

- The op-code from the memory goes to the IR, from the IR it goes to the decoder which decodes instruction. After the instruction is decoded execution begins.
- If the operand is in general purpose register, execution is performed immediately.
- If an instruction contains data or operand address, then CPU has to perform some read operations to get the desired data.
- In some instruction write operation is performed. In write cycle data are sent from the CPU to the memory of an o/p device.
- In some cases execute cycle may involve one or more read or write cycle or both.

# machine cycle

- It is defined as the time required to complete one operation of accessing memory i/p, o/p or acknowledging the external request. This cycle may consists of 3 to 6 T states.
- Op-code Fetch Cycle
- Memory Read Cycle (3T)
- Memory Write Cycle (3T)
- I/O Read Cycle (3T)
- I/O Write Cycle (3T)
- Interrupt acknowledge
- Bus idle





**T-state:**

- It is defined as one sub-division of the operation performed in one clock period. These sub-division are internal states synchronized with system clock and each T state precisely equal to one clock period.

# Opcode fetch machine cycle

- The first machine cycle of every instruction is opcode fetch cycle in which the 8085 finds the nature of the instruction to be executed.
- In this machine cycle, the microprocessor places the contents of PC on the address bus then by reading operation it reads the op-cod of an instruction from determined memory location. The length of this cycle is not fixed.

**Step1: (T1 state)**

- The 8085 processor places the contents of program counter on the address bus, activate the ALE and send the status signals IO/M', S1, and S0 with logical status (0 1 1) respectively.



**Step 2: (T2 state)**

- The low order address disappears from AD0-AD7 lines. Also, 8085 processor activates the RD signals to enable the addressed memory location which places its contents on the data bus (AD0-AD7).

# Opcode fetch machine cycle

**Step 3: (T3 state)**

- The processor loads the contents of data bus on its Instruction Register and deactivates the RD signal to disables the memory devices.

**Step4: (T4 state)**

- Microprocessor decodes the instruction and performed the task specified into instruction.

**Step5: (T5 & T6 states)**

- The processor performs stack write, internal 16 bits, or conditional return operations depending upon the type of instruction.
- One byte instructions those operate on 16 bit data are executed in T5 & T6. For example DCX H, PCHL, SPHL, INX H, etc.

# Opcode fetch machine cycle

**Instruction that takes 6-T for opcode fetch**

*CRISP*

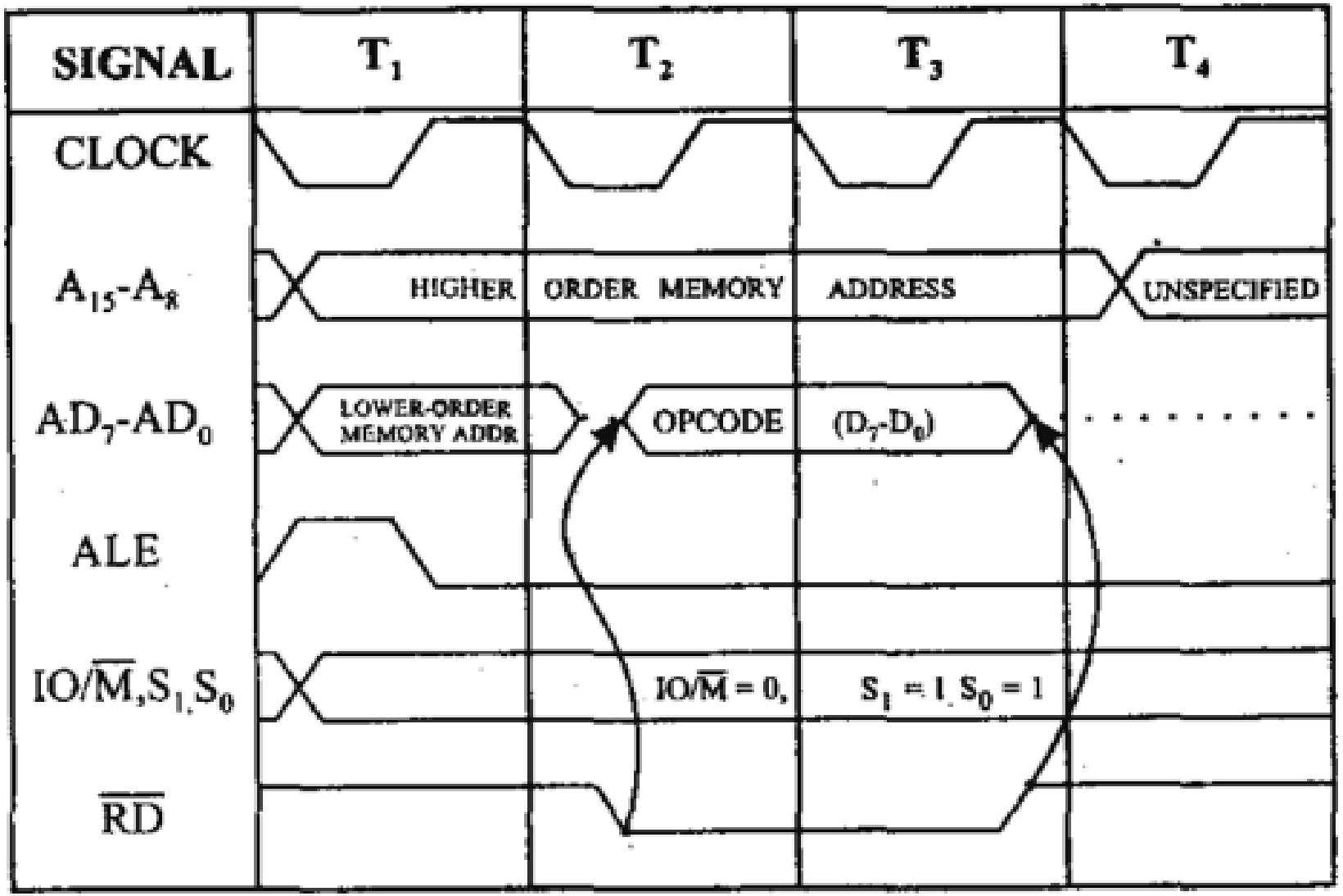
C- CALL, Conditional RET.

R- RST

I- INX ,DCX

S- SPHL, PCHL

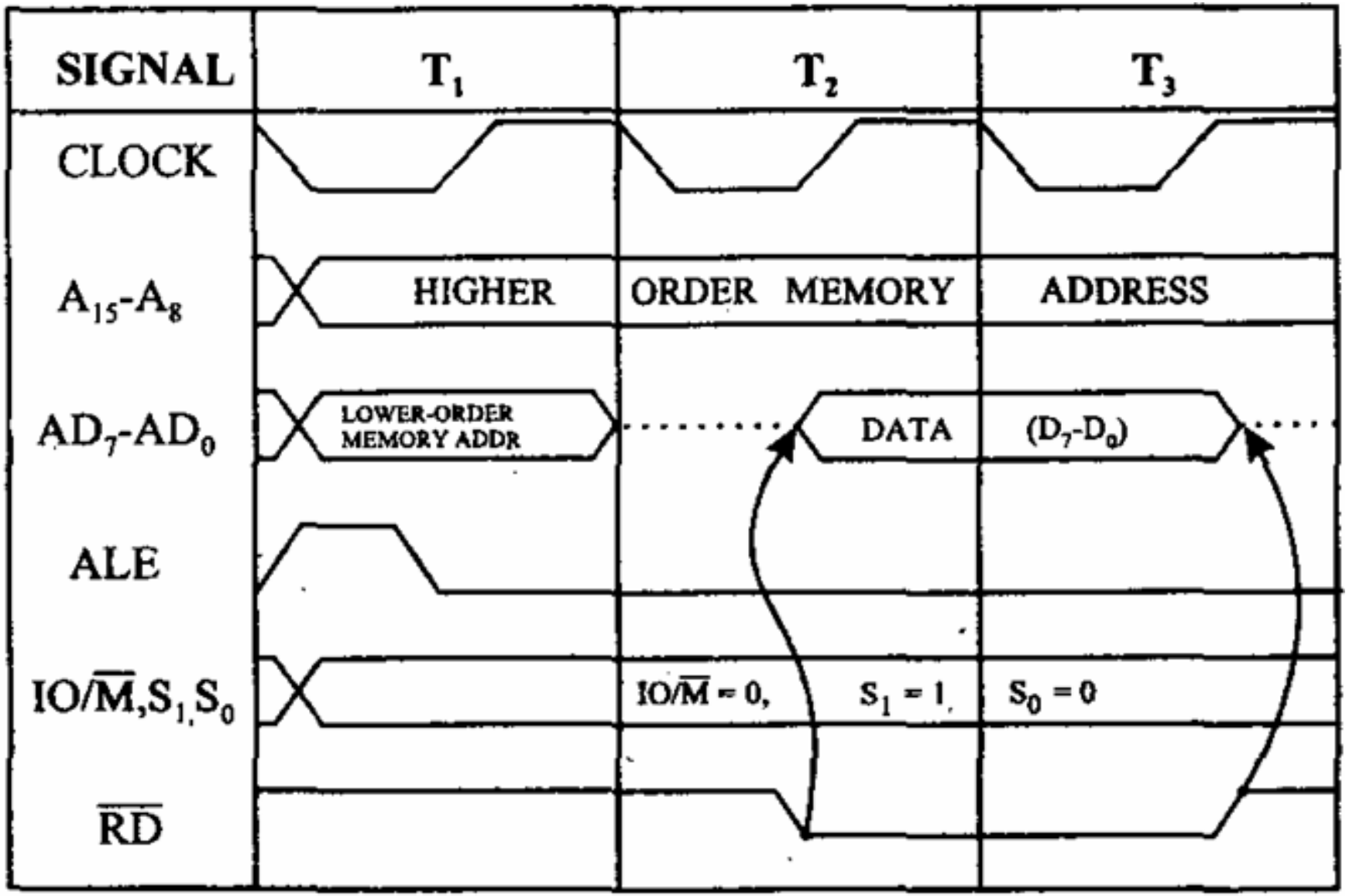
P- PUSH



# MEMORY READ machine cycle

- The microprocessor executes the memory read cycle to read the data from RAM or ROM memory. 8085 processor executes this machine cycle in 3 T-states. Steps below show the details of this machine cycle:
- **Step1 (T1 state):**
  - processor places the address on the address lines from SP, Rp, or PC and activates ALE in order to latch low-order of address. Also, it sends the status signals IO/M', S1, and S0 with logical status (0 1 0) for memory read machine cycle.
- **Step2 (T2 state):**
  - 8085 processor activates the RD' signals to enable the addressed memory location which places its contents on the data bus (AD0-AD7).
- **Step 3: (T3 state)**
  - The processor loads the contents of data bus on specified register (F, A, B, C, D, E, H, and L) and deactivates the RD' signal to disables the memory devices.

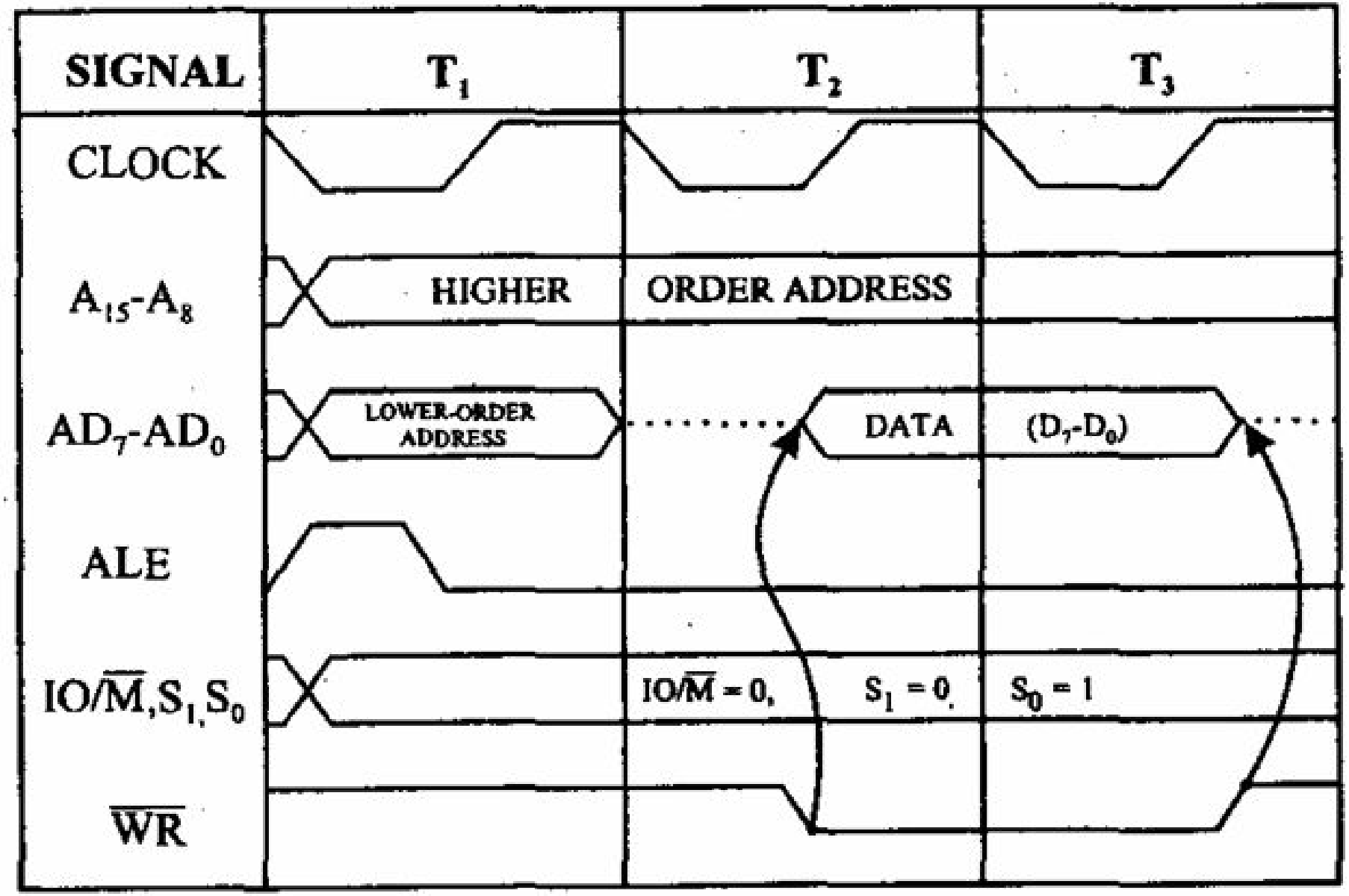
# MEMORY READ machine cycle



# MEMORY WRITE machine cycle

- The microprocessor executes the memory write cycle to store the data into RAM or stack memory. 8085 processor executes this machine cycle in 3 T-states. Steps below show the details of this machine cycle:
- **Step1 (T1 state):**
  - processor places the address on the address lines from SP or Rp and activates ALE in order to latch low-order of address. Also, it sends the status signals IO/M', S1, and S0 with logical status (0 0 1) for memory write machine cycle.
- **Step2 (T2 state):**
  - 8085 processor places the data on data bus and activates the WR' signal to writing data into addressed memory location.
- **Step 3: (T3 state)**
  - The processor deactivates the WR' signal which disables the memory device and terminates the write operation.

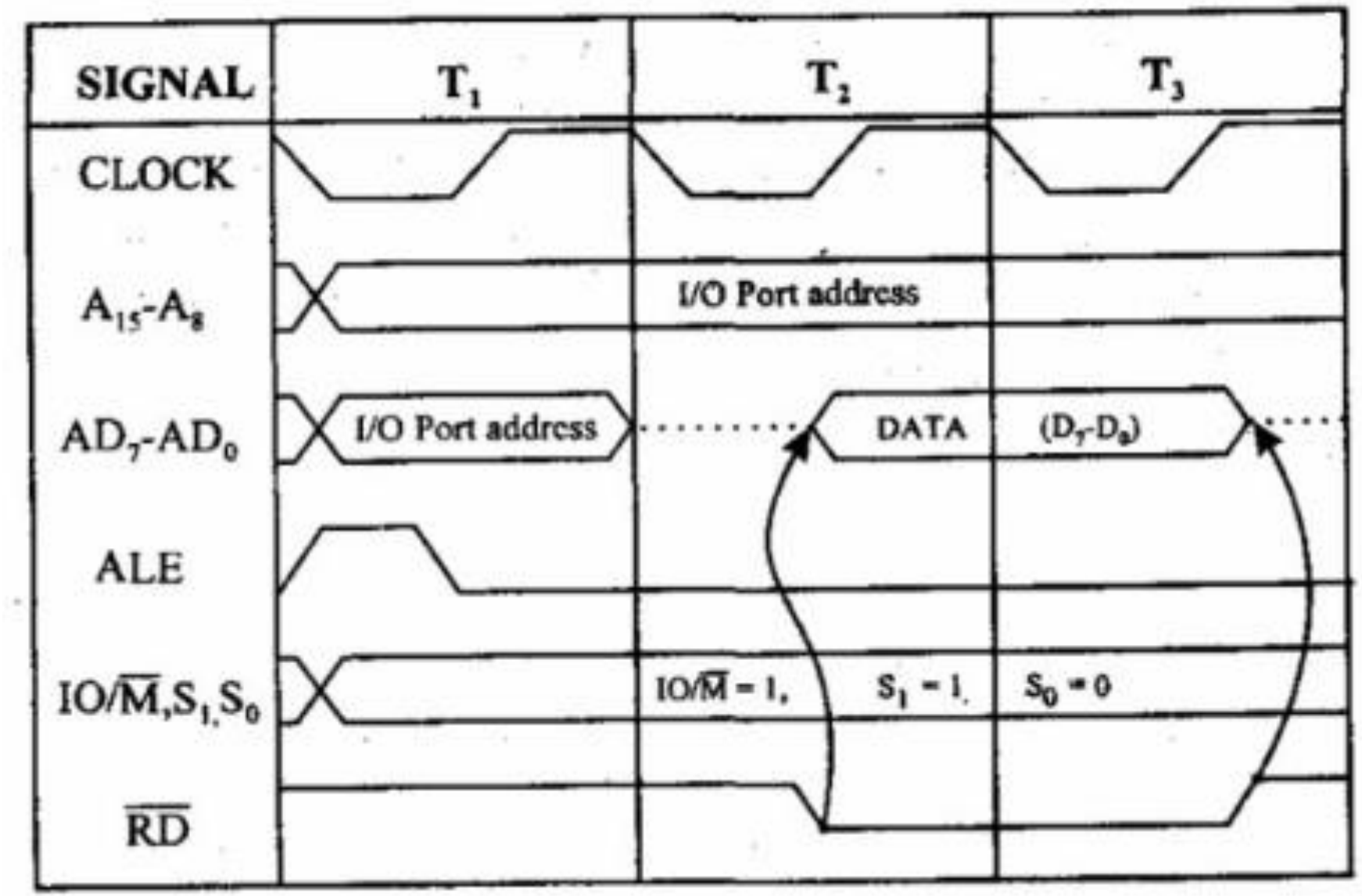
# MEMORY WRITE machine cycle



# IO READ machine cycle

- The microprocessor executes the IO read cycle to read the data from input device. 8085 processor executes this machine cycle in 3 T-states. Steps below show the details of this machine cycle:
- **Step1 (T1 state):**
  - processor places the address on the address lines from SP, Rp, or PC and activates ALE in order to latch low-order of address. Also, it sends the status signals IO/M', S1, and S0 with logical status (1 1 0) for IO read machine cycle.
- **Step2 (T2 state):**
  - 8085 processor activates the RD' signals to enable the addressed input device which places its contents on the data bus (AD0-AD7).
- **Step 3: (T3 state)**
  - The processor loads the contents of data bus on specified register (F, A, B, C, D, E, H, and L) and deactivates the RD' signal to disables the input device.

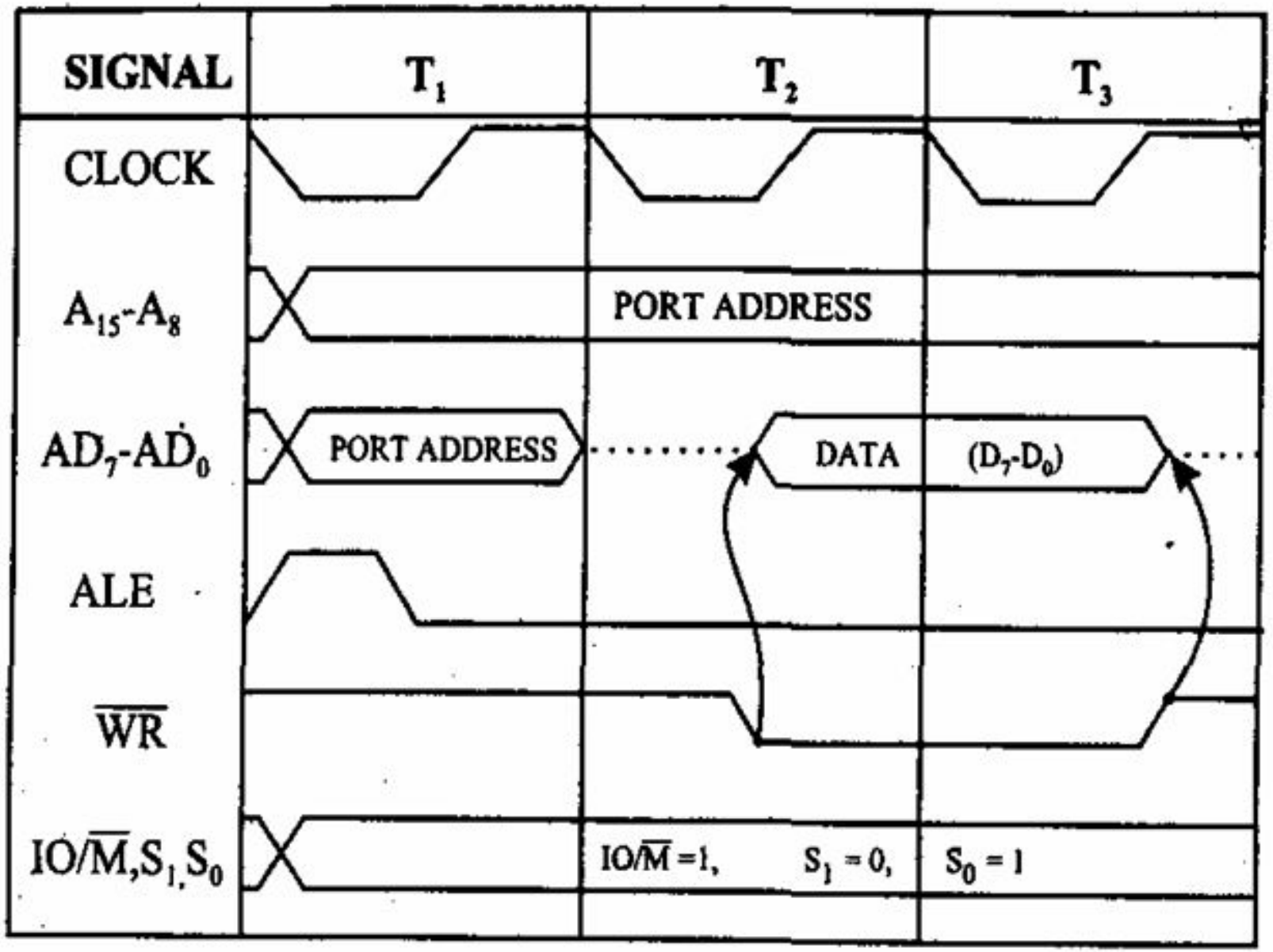
# IO READ machine cycle



# IO write machine cycle

- The microprocessor executes the IO write cycle to store the data into output device. 8085 processor executes this machine cycle in 3 T-states. Steps below show the details of this machine cycle:
- **Step1 (T1 state):**
  - processor places the address on the address lines from SP or Rp and activates ALE in order to latch low-order of address. Also, it sends the status signals with logical status (1 0 1) for IO write machine cycle.
- **Step2 (T2 state):**
  - 8085 processor places the data on data bus and activates the WR’ signal to writing data into addressed output device.
- **Step 3: (T3 state)**
  - The processor deactivates the WR’ signal which disables the output device and terminates the writing operation.

# IO write machine cycle



# memory

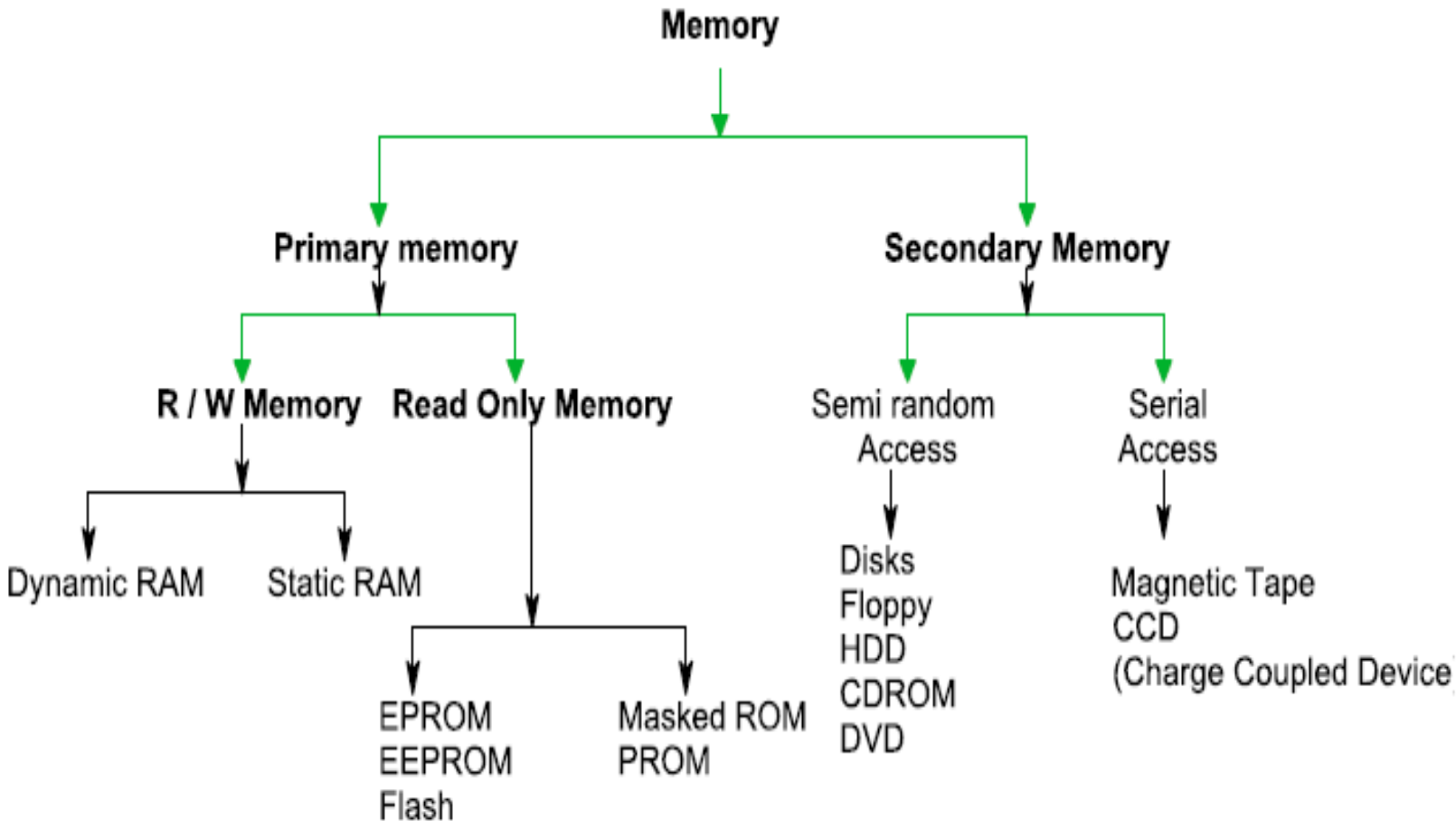



Fig: Classfication of memory system

# memory

- Memory is an essential component of the microcomputer system.
- It is used to store both instructions and data.
- Memory is made up of registers and the number of bits stored in a register is called memory word .
- Memory word is identified by an address .
- If microprocessor uses 16 bit address , then there will be maximum of $2^{16}$= 65536 memory addresses ranging from 0000H to FFFFH.
- There are various types of memory which can be classified in to two main groups i.e. Primary memory and Secondary memory.

# memory

**Processor Memory**

- Processor memory refers to a set of microprocessor registers.
- They are used to hold temporary results when a computation is in progress Although use of such registers enhances the execution speed
- The cost involved in the approach forces a microcomputer designer to include only a few registers include the processor.
- In 8085 we have registers like A, B, C, D, E, H, L, SP, PC etc. to store data temporarily

**Primary Memory**

- It is the storage area where all programs are executed. The microprocessor can directly access only those items that are stored in the primary memory
- Hence, all programs and data must be within the primary memory prior to execution.
- Usually, the size of the primary memory is much larger than that of processor memory and its operating speed is much slower than processor's registers.

# memory

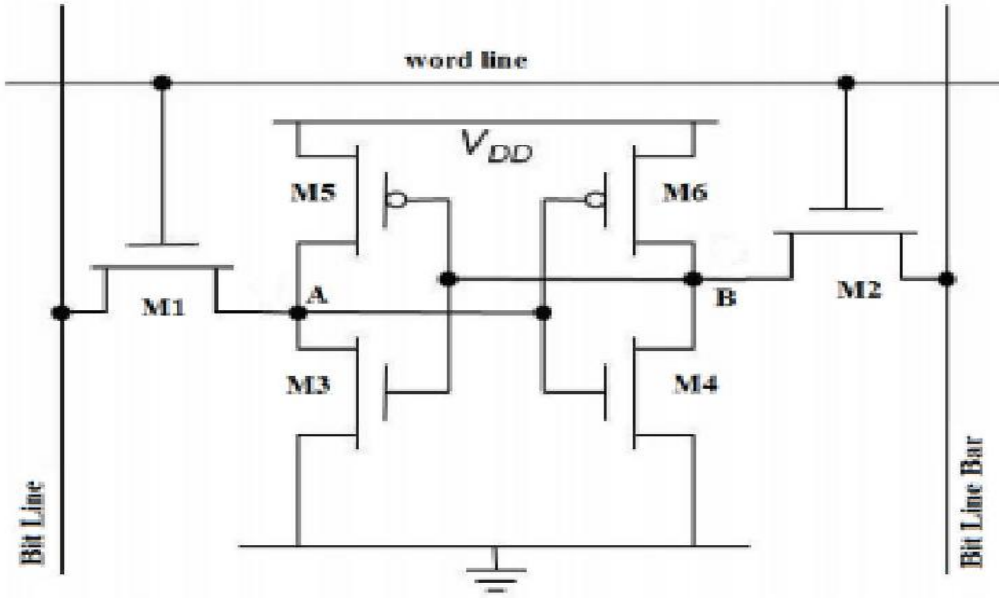
- Primary memories can be divided into two main groups
  - Read only memory (ROM)
  - Random Access memory .(RAM)

**RAM**

- It is used primarily for information that is likely to be altered, such as writing programs or receiving data.
- This memory is volatile Two types of RAM are available
- **Static RAM**
  - This memory is made up of flip flops and stores the bits as voltage.

# memory

- Each memory cell requires six transistors.
- This memory is more expensive and power consuming than dynamic memory.
- It is called ' because the information doesn't need a constant update.
- These memories are commonly used for cache memory.
- This types of memory is very fast with access time is 15 to 30 nanoseconds but is physically bulky.

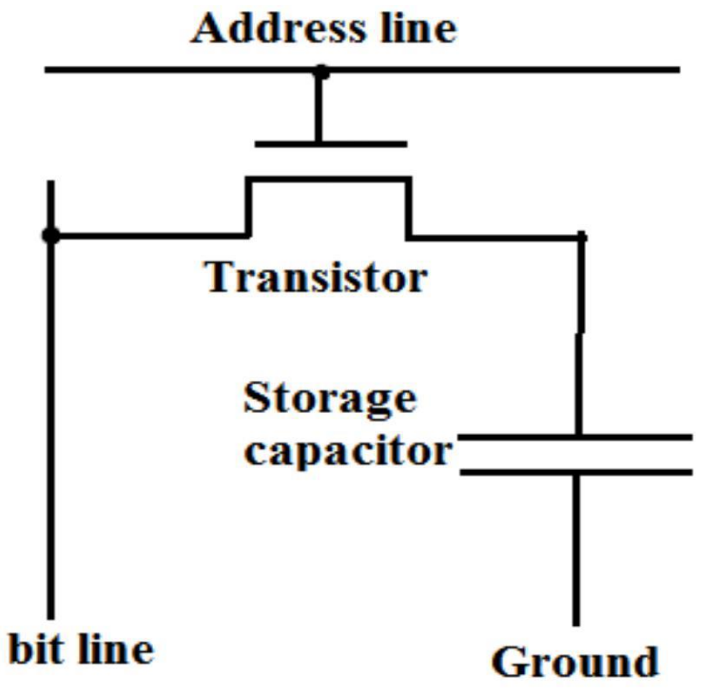


# memory

## Dynamic RAM (DRAM)

- Dynamic random access memory is an improvement over the expensive and bulky SRAM.
- DRAM uses a different approach to store data Information is stored as charges in a very small capacitor.
- If a charge exists in a capacitor, it's interpreted as 1 The absence of a charge will be interpreted as 0.
- Because DRAM uses capacitors there is a chance of leakage of charge.
- Thus it needs to use a constant refresh signal to keep the information in the memory (every few millisecond)
- DRAM technology allows several memory units, called cells to be packed at very high density.
- Therefore, these chips can hold very large amount of information.
- Most PCs today use DRAM.
- Access time for DRAM is 80 nanoseconds or more, slower than SRAM, and two or three times faster than ROM

# memory




DRAM

| SRAM | DRAM |
|---|---|
| 1. SRAM has lower access time, so it is faster compared to DRAM. | 1. DRAM has higher access time, so it is slower than SRAM. |
| 2. SRAM is costlier than DRAM. | 2. DRAM costs less compared to SRAM. |
| 3. SRAM requires constant power supply, which means this type of memory consumes more power. | 3. DRAM offers reduced power consumption, due to the fact that the information is stored in the capacitor. |
| 4. Due to complex internal circuitry, less storage capacity is available compared to the same physical size of DRAM memory chip. | 4. Due to the small internal circuitry in the one-bit memory cell of DRAM, the large storage capacity is available. |
| 5. SRAM has low packaging density. | 5. DRAM has high packaging density. |

# memory

**Read only memory (ROM)**

- ROM is a non volatile memory and can be read only.
- It is used to store data and programs that are not to be altered
- Among other things ROM is needed for storing an initial program called boot strap loader.
- The bootstrap loader is a program whose function is to start the computer software operating when power is turned on.
- Since RAM is volatile, its contents are destroyed when power is turned off . The contents of ROM remain unaltered after power is turned off and on again
- The startup of a computer consists of turning the power on and starting the execution of an initial program.
- Thus when power is turned on, the hardware of the computer sets the program counter to the first address of the bootstrap loader.
- The bootstrap program loads a portion of the operation of the operating system from disk to main memory and control is then transferred to the operating system, which prepares the computer for general use.

memory

## Masked ROM

- They are permanent ROM recorded by masking Generally manufacturers use this process to produce ROM in large numbers

## PROM

- These are un programmed ROM The fuses on the ROM are not burned.
- A programmer can program this ROM according to his needs.
- The information stored is permanent.

# memory

## EPROM

- These ROM can be reprogrammed and erased. Two types of such EPROM are available

## UV EPROM

- The memory of such ROM can be erased by exposing the chip via a lid or window on the chip to ultraviolet light.
- The erase time generally varies between 10 to 30 minutes.
- The EPROM can be programmed by inserting the chip into a socket of the PROM programmer and providing proper addresses.
- The programming time varies from 1 to 2 minutes.



## EEPROM

- This does not require UV rays to be erased It can be completely erased or have certain byes changed, using electrical pulses
- Writing to EEPROM is slower than writing to RAM, so it can not be used in high speed circuits.

memory

## FLASH MEMORY

- This is a modified EEPROM. The difference is the erasure procedure.
- EEPROM can be erased at a register level, but flash memory must be erased either in its entirety or at the sector (block) level

## Secondary Memory

- Secondary memory are storage devices. These devices have high data holding capacity.
- They store programs that are not frequently used by the processor.
- They are slow and have larger size

memory

## Performance of memory

- **Access time ($t_a$)**
  - Read access time: It is the average time required to read the unit of information from memory
  - Write access time: It is the average time required to write the unit of information on memory
  - Access rate $r_a = 1 / t_a$
- **Cycle time ($t_c$)**
  - It is the average time that lapses between two successive read operation Cycle rate ($r_c$) = bandwidth = $(1/t_c)$

memory

- **Access modes of memory**

- **Random access**
  - In random access mode, the ta is independent of the location from which the data is accessed like MOS memory

- **Sequential access**
  - In that mode, the ta is dependent of the location form which the data is accessed like magnetic type

- **Semi random access**
  - The semi random access combines these two For e.g. In magnetic disk, any track can be accessed at random But the access within the track must be in serial fashion

# Memory hierarchy

- Capacity, cost and speed of different types of memory play a vital role while designing a memory system for computers.
- If the memory has larger capacity, more application will get space to run smoothly.
- It's better to have fastest memory as far as possible to achieve a greater performance. Moreover for the practical system, the cost should be reasonable.a
- There is a tradeoff between these three characteristics cost, capacity and access time. One cannot achieve all these quantities in same memory module because
  - If capacity increases, access time increases (slower) and due to which cost per bit decreases.
  - If access time decreases (faster), capacity decreases and due to which cost per bit increases.
- The designer tries to increase capacity because cost per bit decreases and the more application program can be accommodated. But at the same time, access time increases and hence decreases the performance.

# Memory hierarchy

- **So the best idea will be to use memory hierarchy.**
- Memory Hierarchy is to obtain the highest possible access speed while minimizing the total cost of the memory system.
- Not all accumulated information is needed by the CPU at the same time.
- Therefore, it is more economical to use low-cost storage devices to serve as a backup for storing the information that is not currently used by CPU
- The memory unit that directly communicate with CPU is called the main memory
- Devices that provide backup storage are called auxiliary memory
- The memory hierarchy system consists of all storage devices employed in a computer system from the slow by high-capacity auxiliary memory to a relatively faster main memory, to an even smaller and faster cache memory

# Memory hierarchy

- The main memory occupies a central position by being able to communicate directly with the CPU and with auxiliary memory devices through an I/O processor
- A special very-high-speed memory called cache is used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate
- CPU logic is usually faster than main memory access time, with the result that processing speed is limited primarily by the speed of main memory
- The cache is used for storing segments of programs currently being executed in the CPU and temporary data frequently needed in the present calculations
- The memory hierarchy system consists of all storage devices employed in a computer system from slow but high capacity auxiliary memory to a relatively faster cache memory accessible to high speed processing logic. The figure below illustrates memory hierarchy.

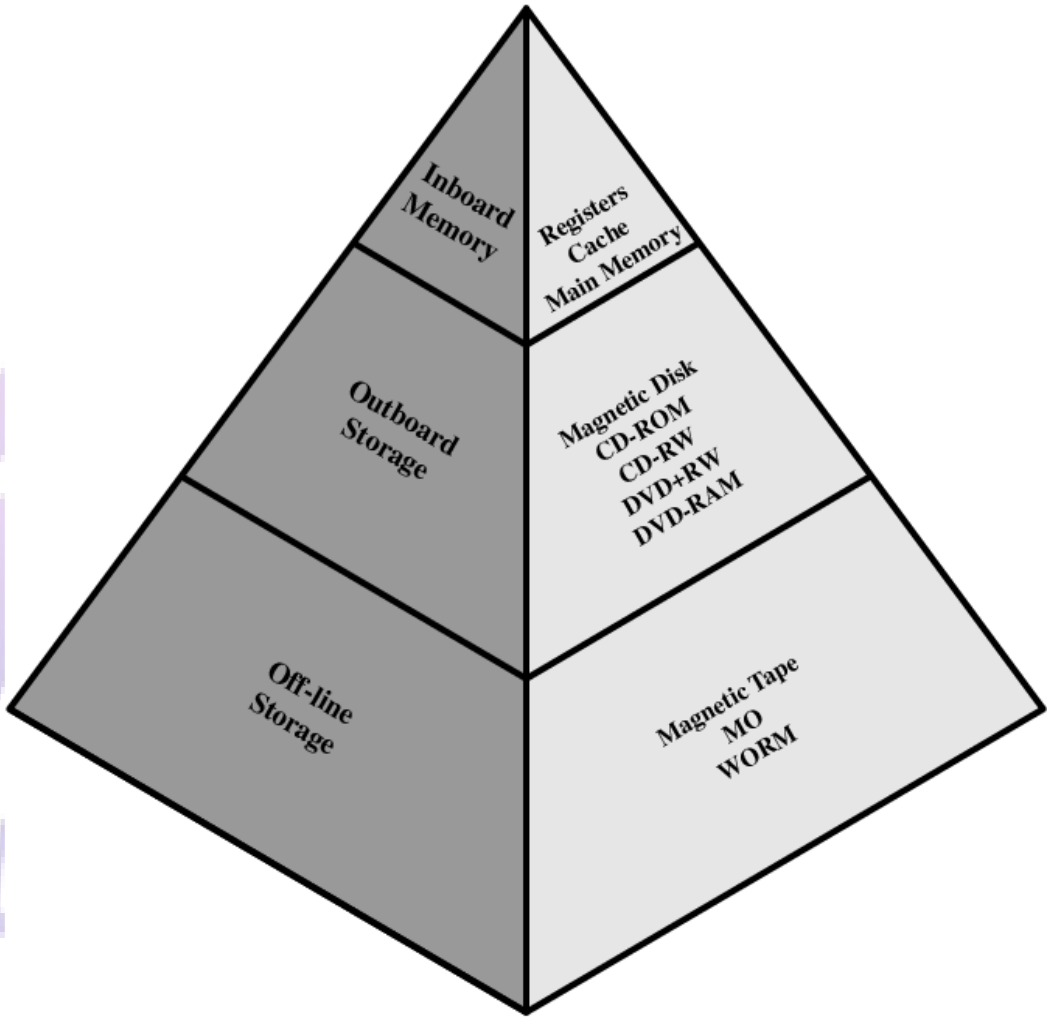
# Memory hierarchy

**As we go down in the hierarchy**

- Cost per bit decreases
- Capacity of memory increases
- Access time increases
- Frequency of access of memory by processor also decreases.

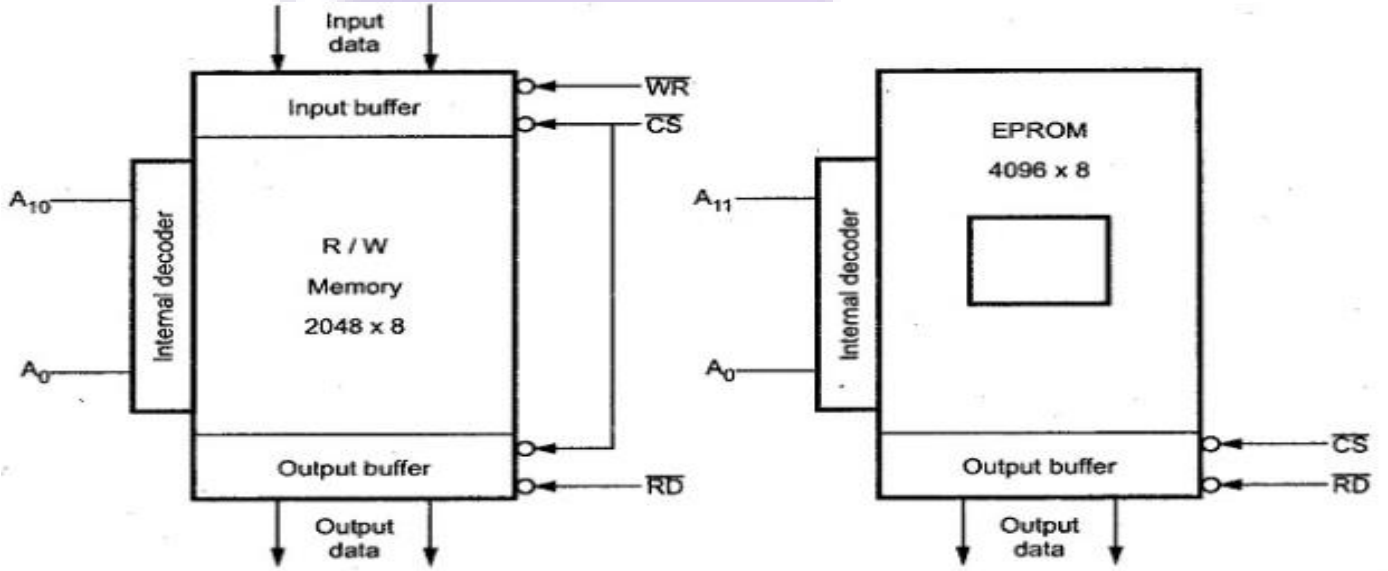
**Hierarchy List**

- Registers
- L1 Cache
- L2 Cache
- Main memory
- Disk cache
- Disk
- Optical
- Tape



# Memory structure and its requirements

- Internally a memory consists of address decoder, input buffer, output buffer, registers with address lines, data lines, and ($RD$') ($WR$'), ($CS$') control lines.
- The number of address lines will be determined by the memory capability.
- The number of data lines will be determined by memory size.
- For $2^n$ X m memory capability, the number of address line =n and the number of data lines =m.



# Address decoding

| Memory mapped I/O | I/O mapped I/O |
|---|---|
| 1. In this device address is 16 bit. Thus $A_0$ to $A_{15}$ lines are used to generate device address. | 1. In this I/O device address is 8 bit. Thus $A_0$ to $A_7$ or $A_8$ to $A_{15}$ lines are used to generate device address. |
| 2. $\overline{MEMR}$ and $\overline{MEMW}$ control signals are used to control read and write I/O operations. | 2. $\overline{IOR}$ and $\overline{IOW}$ control signals are used to control read and write I/O operations. |
| 3. Instructions available are LDA addr, STA addr, LDAX rp, STAX rp, MOV M,R, MOV R,M ADD M, CMP M etc. | 3. Instructions available are IN and OUT. |
| 4. Data transfer is between any register and I/O device. | 4. Data transfer is between accumulator and I/O device. |
| 5. Maximum number of I/O devices are 65536 (theoretically). | 5. Maximum number of I/O devices are 256. |
| 6. Execution speed using LDA addr, STA addr is 13 T-state and 7 T-states for MOV M, r and MOV r, M instructions. | 6. Execution speed is 10 T-states. |
| 7. Decoding 16 bit address may require more hardware. | 7. Decoding 8 bit address will require less hardware. |

# Serial interface

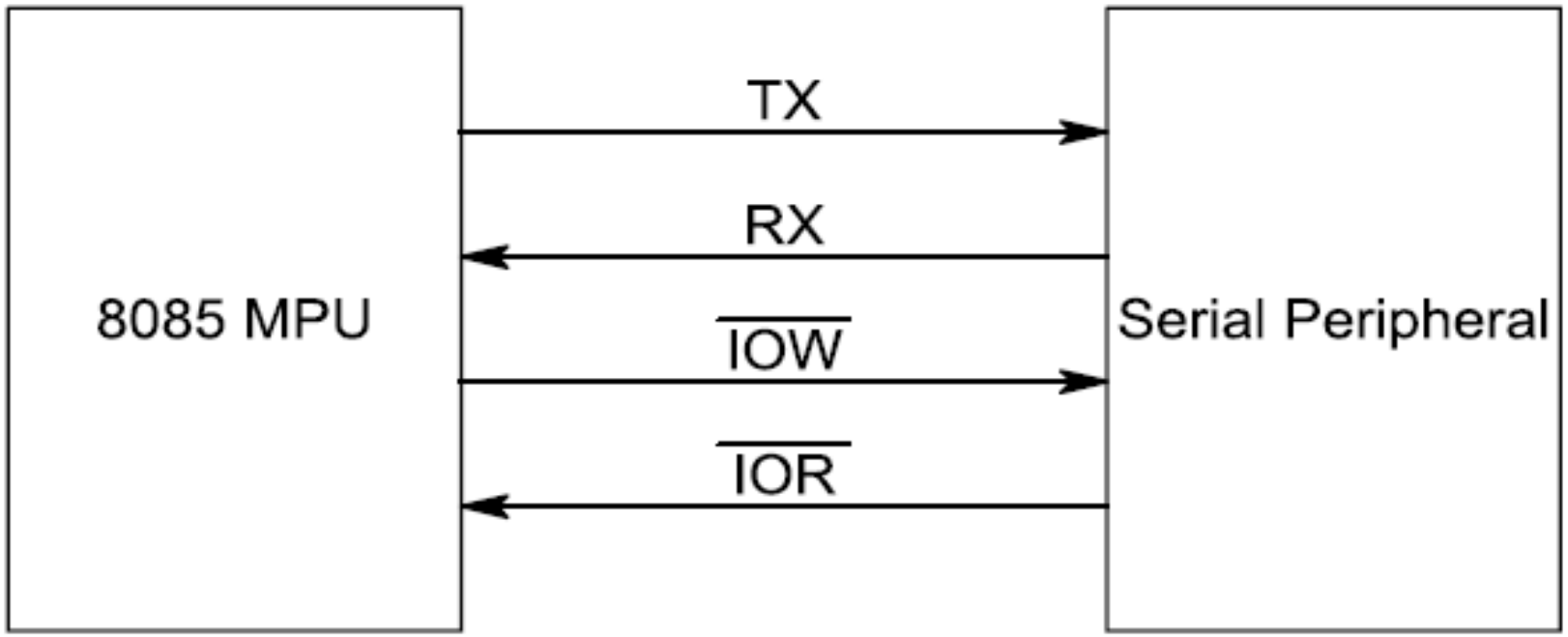
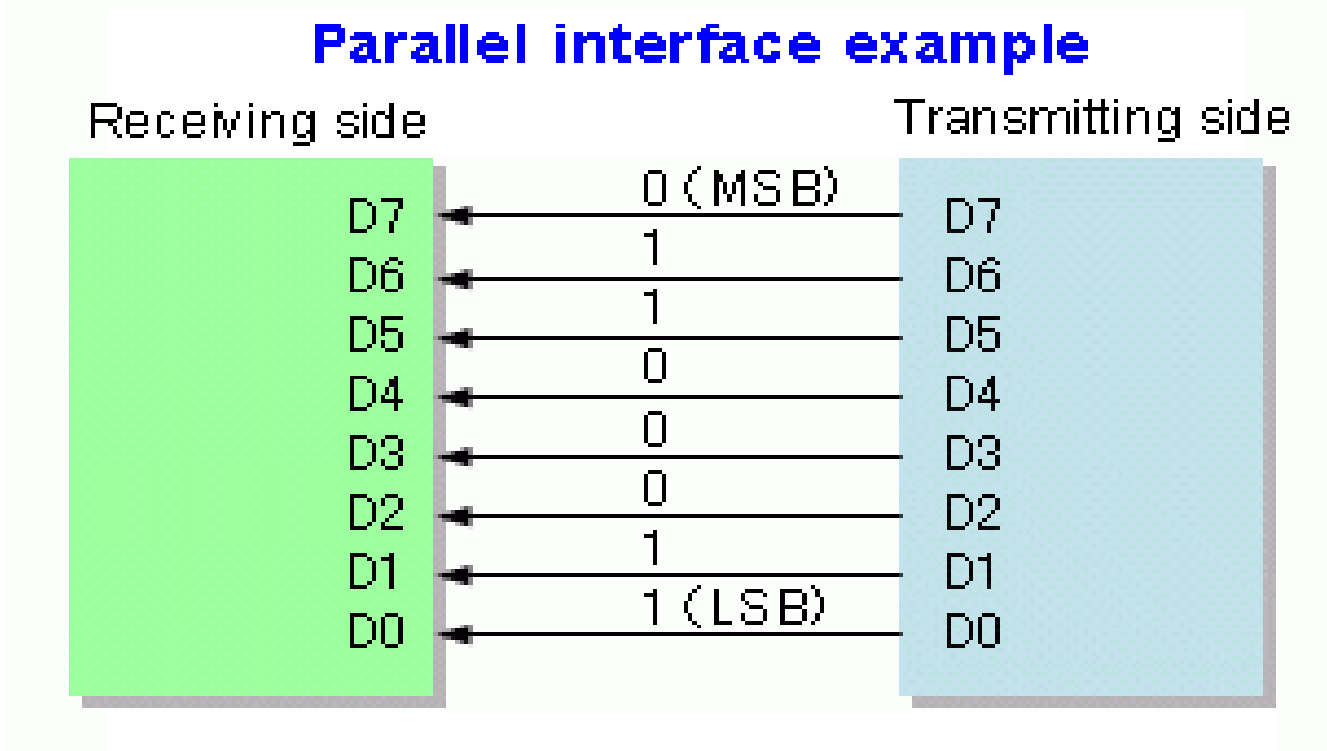



Fig: Serial Interface between microprocessor and I/O device.

# Parallel interface

- The device which can handle data at higher speed cannot support with serial interface.
- N bits of data are handled simultaneously by the bus and the links to the device directly.
- Achieves faster communication but becomes expensive due to need of multiple wires.



# Synchronizing the computer with peripherals

- The information exchanged between a microprocessor and an I/O interface circuit consists of input or output data and control information.
- The status information enable the microprocessor monitor the device and when it is ready then send or receive data.
- Control information is the command by microprocessor to cause I/O device to take some action.
- If the device operates at different speeds, then microprocessor can be used to select a particular speed of operation of the device.
- The techniques used to transfer data between different speed devices and computer is called synchronizing. Different techniques under synchronizing are:

# Synchronizing the computer with peripherals

## Simple I/O:

- For simple I/O, the buffer switch and latch switches i. e. LED are always connected to the input and output ports.
- The devices are always ready to send or receive data.
- Here cross line indicate the time for new valid data.

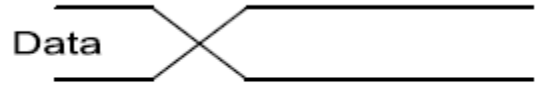



Fig: Simple I /O

## Wait Interface( Simple strobe I/O)

- In this technique, microprocessor need to wait until the device is ready for the operation.

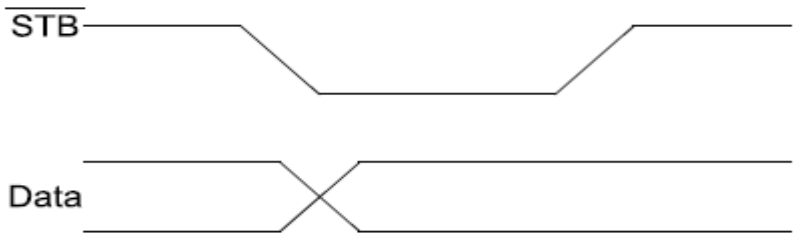



Fig: Simple Strobe I/O

# Synchronizing the computer with peripherals

- Consider a simple keyboard consisting of 8 switches connected to a microprocessor through a parallel interface circuit (Tri-state buffer).
- The switch is of dip switches.
- In order to use this keyboard as an input device the microprocessor should be able to detect that a key has been activated.
- This can be done by observing that all the bits are in required order.
- The processor should repeatedly read til it finds the right order of bits i.e. at least 1 bit of 8
- Consider the tri-state A/D converter

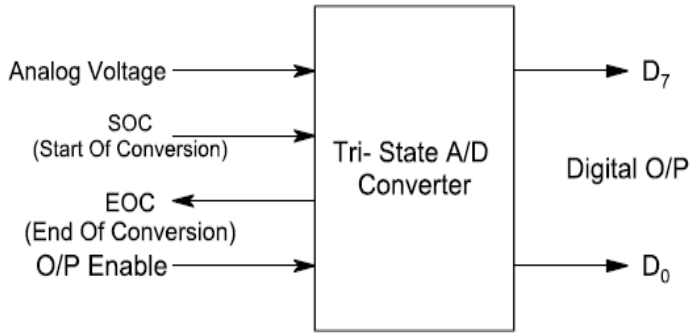



Fig: Tri-State A/D Converter

# Synchronizing the computer with peripherals

- Used to convert analog to digital data which can be read by I/O unit of MP
- When SOC appears 1, I/O unit should ready for reading binary data/digital data.
- When EOC's status is 1, then I/O unit should stop to read data.
- Strobe signal indicates the time at which data is being activated to transmit.

# Synchronizing the computer with peripherals

**Single Handshaking:**

- The peripheral outputs some data and send STB' signal to microprocessor. "here is the data for you."
- Microprocessor detects asserted STB' signal, reads the data and sends an acknowledge signal (ACK) to indicate data has been read and peripheral can send next '
- Microproc pheral is ready.

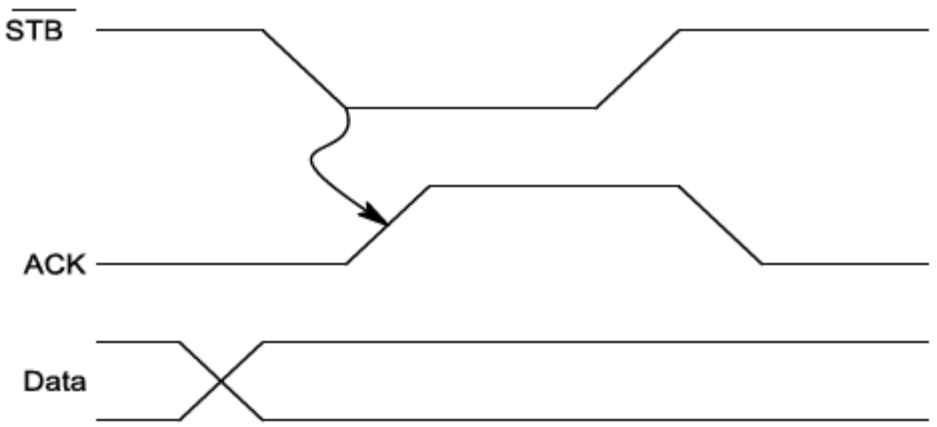



Fig: Single Handshaking

# Synchronizing the computer with peripherals

**Double Handshaking:**

- The peripheral asserts its STB' line low to ask MP "Are you ready?"
- The MP raises its ACK line high to say " I am ready".
- Peripheral then sends data and raises its STB' line low to say "Here is some valid data for you."
- MP then reads the data ave the data, thank you, and I await your requ

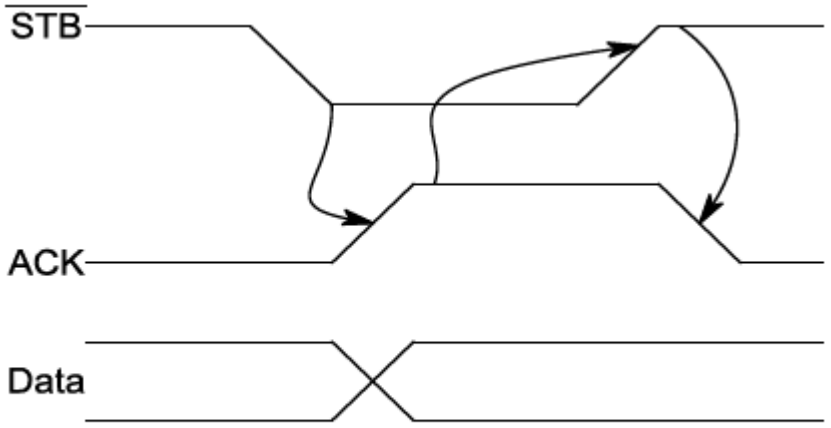



Fig: Double Handshaking

# Programmable peripheral interface (ppi) – 8255A

- The Intel 8255A is a general purpose programmable I/O device designed for use with Intel microprocessors.
- It has 24 I/O pins that can be grouped primarily in two 8-bit parallel ports: A and B, with the remaining bits as port C.
- The 8-bits of port C can be used as individual bits or be grouped in two 4-bits ports: $C_{upper}$ ($C_u$) and $C_{lower}$ ($C_l$).
- The functions of these ports are defined by writing a control word in the control register.

# Programmable peripheral interface (ppi) – 8255A

## Block diagram:

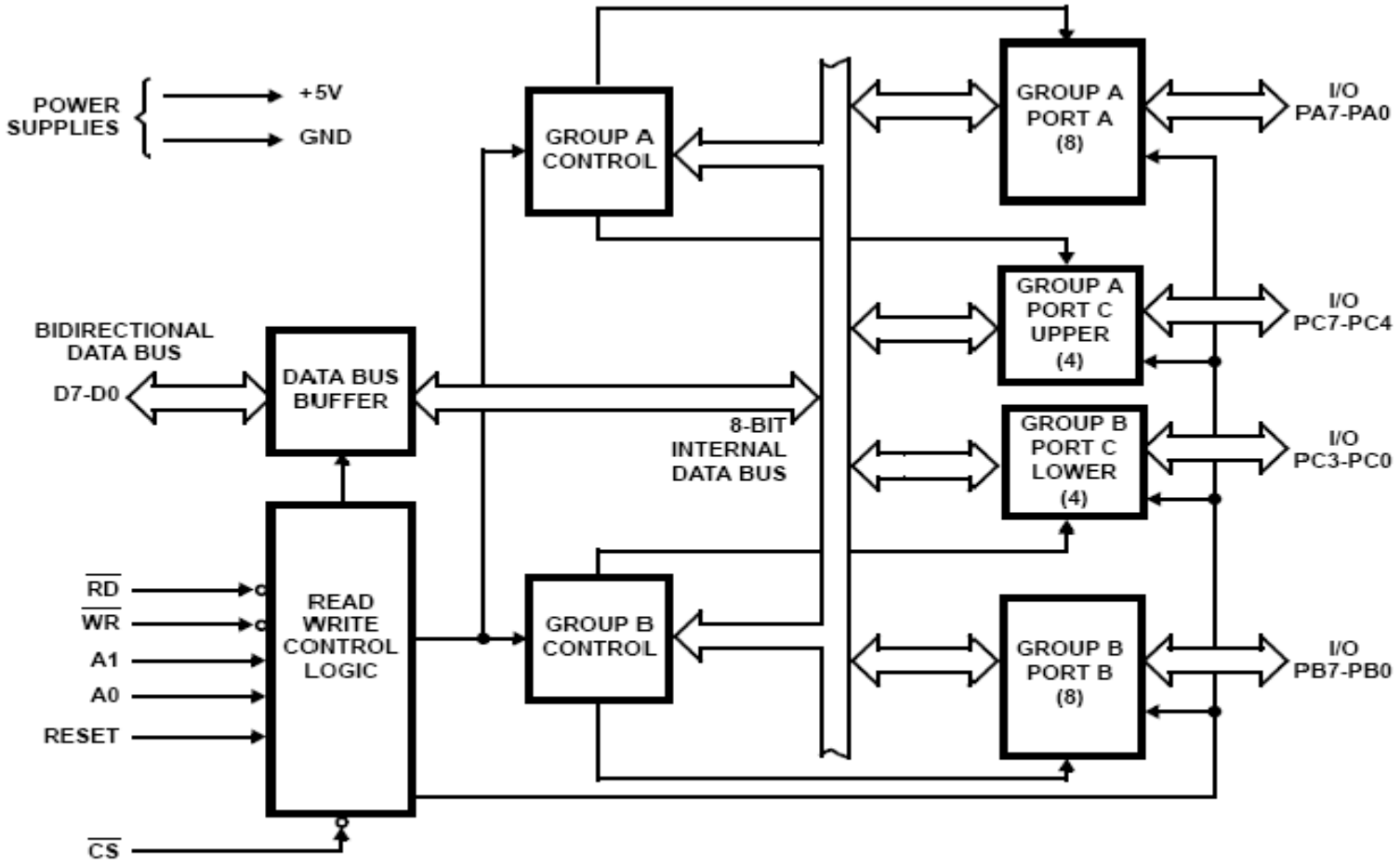



**Fig2: Internal Block Diagram of 8255**

# Programmable peripheral interface (ppi) – 8255A

**Data Bus Buffer**

- The 3-state bidirectional 8-bit buffer is used to interface the 8255A to the system data bus.
- Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU.
- Control words and status information are also transferred through the data bus buffer.

**Read/Write Control Logic**

- The function of the block is to manage all of the internal and external transfers of both data and control or status words.
- It accepts inputs from the CPU address and control buses and in turn, issues commands to both of the control groups.
- Chip Select (CS'): A "low" on this pin enables the communications between the 8255A and the CPU.

# Programmable peripheral interface (ppi) – 8255A

- **Read (RD'):** A "low" on this input enables the 8255A to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to read from the 8255A.
- **Write (WR'):** A "low" on this input pin enables the CPU to write data or control words into the 8255A.
- **Reset (RESET):** A "high" to this pin clears the control register and sets all ports (A, B and C) in the input mode.
- **A0 and A1:** These input signals controls the selection of one of the three ports or the control word register. They are connected to the least significant bits of the address bus.
- The **CS'** signal is the master chip select, and A0 and A1 specify one of the I/O ports or the control register as given below.

# Programmable peripheral interface (ppi) – 8255A

| CS' | $A_1$ | $A_0$ | Selected |
|---|---|---|---|
| 0 | 0 | 0 | Port A |
| 0 | 0 | 1 | Port B |
| 0 | 1 | 0 | Port C |
| 0 | 1 | 1 | Control Register |
| 1 | X | X | 8255A is not selected |

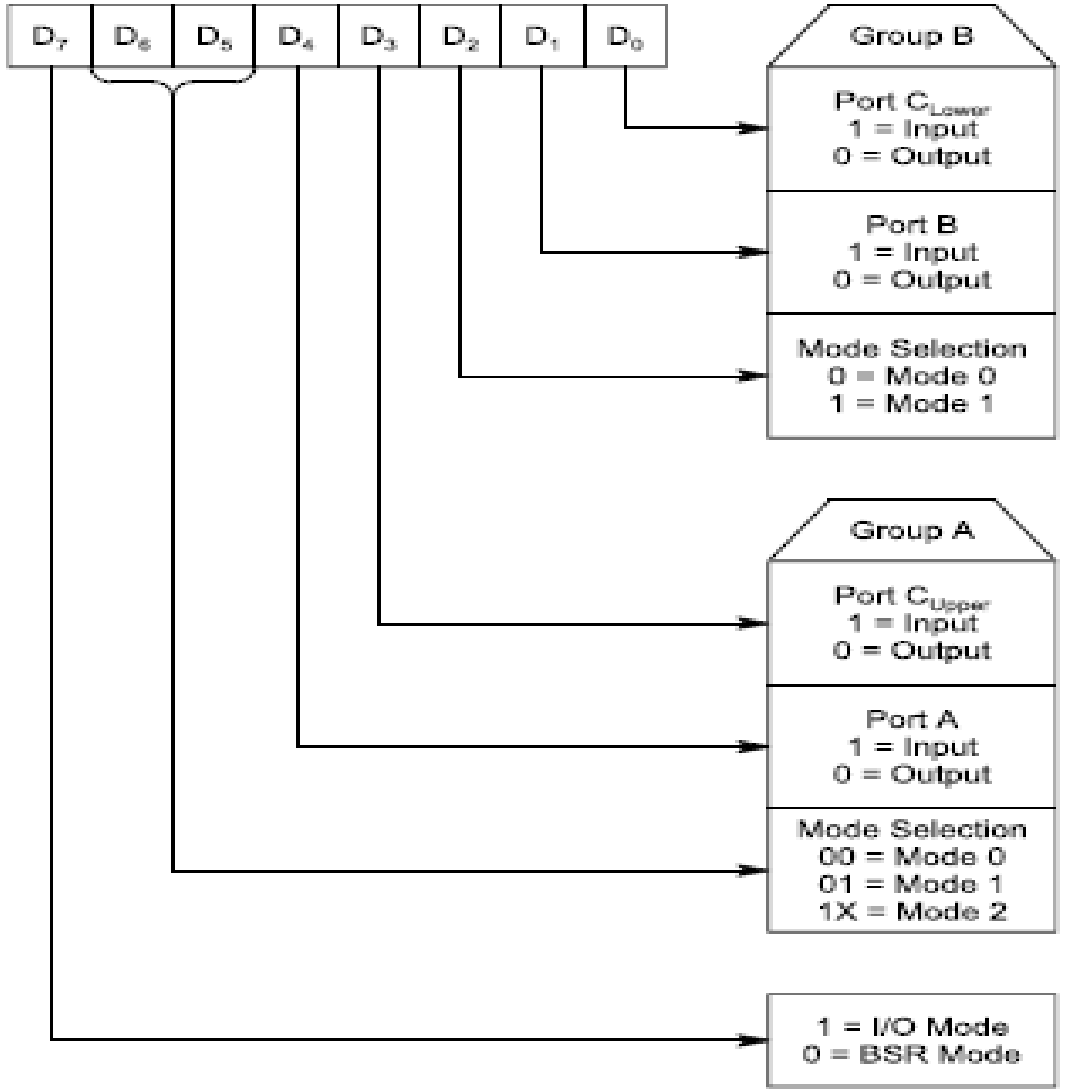
- **Group A and Group B controls**
  - Each of the control blocks (Group A and Group B) accepts "commands" from the Read/Write control logic, receives control word from the internal data bus and issues the proper commands to its associated ports.
  - Control Group A – Port A and Port $C_{Upper}$ (C7 – C4)
  - Control Group B – Port B and Port $C_{Lower}$ (C3 – C0)

# Programmable peripheral interface (ppi) – 8255A

**Control Word**

- When A0 and A1 pins have value 1, the mapped address addresses the control register which is the 8-bit register to write the specific content according to the port conditions although it cannot be read. The content of this register is called control word which specifies an I/O function for each port.
- To communicate with peripherals through 8255, following steps are necessary.
- Determine the Port addresses of Ports A, B and C and of the control register, according to Chip Select logic and address lines A1 and A0.
- Write a control word in control register.
- Write I/O instructions to communicate with peripherals through Ports A, B and C.

# Programmable peripheral interface (ppi) – 8255A




8255A Control Word Format for I/O Mode

# Programmable peripheral interface (ppi) – 8255A

**Operating modes:**

- **Bit Set/Reset mode:** The BSR mode is used to set or reset the bits in port C.
- **I/O mode:** The I/O mode is further divided into three modes: mode 0, mode 1 and mode 2.
- In **mode 0,** all ports function as simple I/O ports.
- **Mode 1** is a handshake mode whereby ports A and/or B use bits from port C as handshake signals. In the handshake mode, two types of I/O data transfer can be implemented: status check and interrupt.
- In **mode 2**, port A can be set up for bidirectional data transfer using handshake signals from port C and port B can be set up either in mode 0 or mode 1.

# Programmable peripheral interface (ppi) – 8255A

**BSR Mode (Bit Set/Reset)**

- BSR mode is concerned only with eight bits of port C, which can be set or reset by writing an appropriate control word in the control register.
- A control word with bit D7=0 is recognized as a control word and it does not alter any previously transmitted control word with bit D7=1; thus the I/O operations of ports A and B are not affected by a BSR control word.
- In the BSR mode individual bits of port C can be used for applications such as On/Off switch

**BSR Control Word:**

- This control word, when written in control register, sets or resets one bit at a time, as specified in figure

# Programmable peripheral interface (ppi) – 8255A

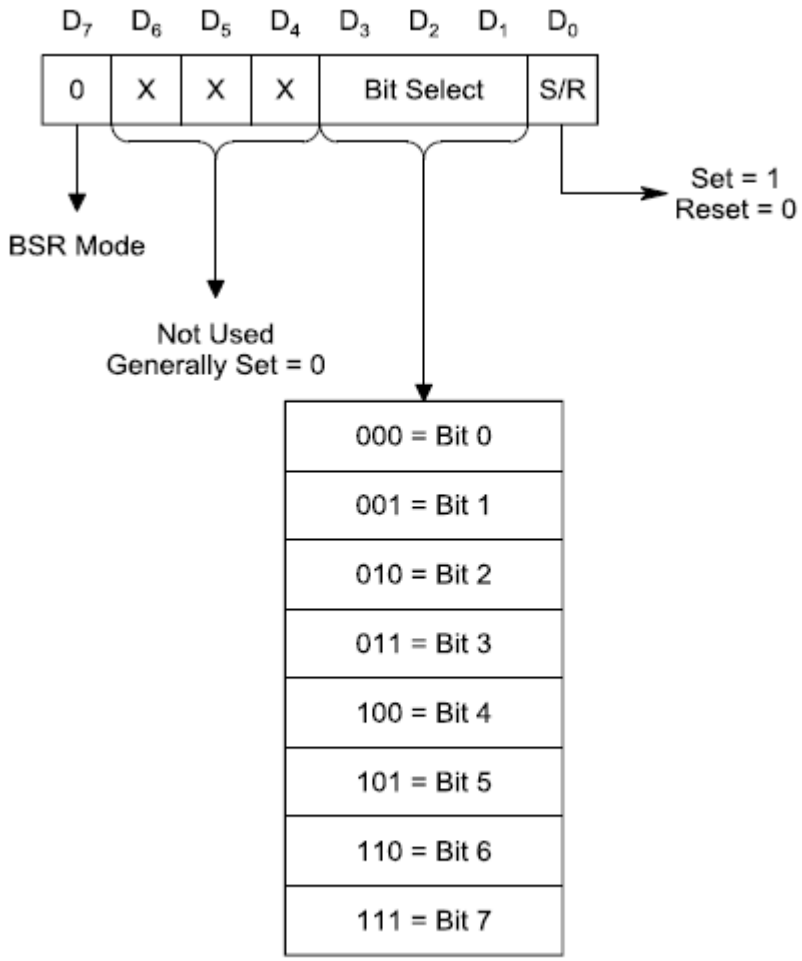



Fig: 8255A Control Word Format for BSR Mode

# Programmable peripheral interface (ppi) – 8255A

**Mode 0 (Basic Input/output)**

- This functional configuration provides simple input and output operation for each of the three ports.

  No ‘handshaking” is required; data is simply written to or read from a specified port..

- Mode 0 basic functional definitions:
  - Two 8-bit ports and two 4-bit ports
  - Any port can be input or output
  - Outputs are latched
  - Inputs are not latched
  - 16 different input/output configurations are possible in this mode.

# Programmable peripheral interface (ppi) – 8255A

**Mode 1 (Strobe Input/output)**

- The functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or handshaking signals.
- In mode 1, port A and port B use the lines of port C to generate or accept these handshaking signals.
- Mode 1 basic functional definitions:
  - Two groups (Group A and Group B)
  - Each group contains one 8-bit data port and one 4-bit control/data port
  - The 8-bit data port can be either input or output. Both inputs and outputs are latched.
  - The 4-bit port is used for control and status of the 8-bit data port.

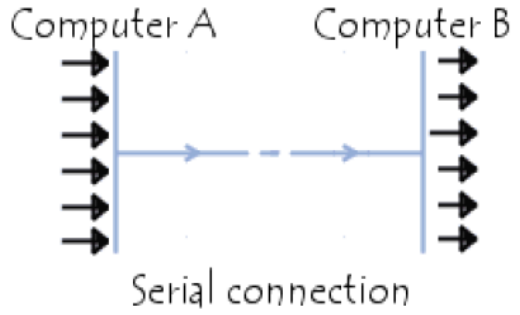
# Programmable peripheral interface (ppi) – 8255A

**Mode 2 (Strobe Bidirectional Bus I/O)**

- The functional configuration provides a means for communicating with a peripheral device or a structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O).
- "Handshaking Signals" are provided to maintain proper bus flow discipline in a similar manner to Mode 1.
- Interrupt generation and enable/disable functions are also available.
- Mode 2 basic functional definitions:
  - Used in Group A only
  - One 8-bit bidirectional bus port (Port A) and a 5-bit control port (Port C)
  - Both inputs and outputs are latched
  - The 5-bit control port (Port C) is used for control and status for the 8-bit, bidirectional bus port (Port A)

# Serial data transmission

- Data are sent one bit at a time over the transmission channel.
- However, since most processors process data in parallel, the transmitter needs to transform incoming parallel data into serial data and the receiver needs to do the opposite.
- Cost of communication hardware is considerable reduced since only a single wire or channel is require for transmission.
- Slow as compared to parallel transmission.
- Serial data can be sent synchronously or asynchronously.



# Serial data transmission - advantages

- Data transmission over longer distance because voltage loss is not much a problem in serial communication.
- Serial; 1 → -3V to -25V
- 0 → +3V to +25V
- Parallel; 1→ +5V
- 0 → 0V
- Requires less number of wires than parallel and so cheaper to transmit data.
- Crosstalk is less of an issue because there are fewer conductors' compared to that of parallel cables.
- Many IC and peripherals have serial interface
- Clock skew between different cables is not an issue
- Serials can be clocked at higher data rate
- Serial cable can be longer than parallel
- Cheaper to implement
- *But in serial mode of transfer, only one bit of a word is transferred at a time so that data transfer rate is very slow; it is the one of the demerit over parallel data transfer*

# Serial data transmission

**Serial Synchronous Data Transmission**

- Data is transmitted or received based on a clock pulse (i.e. one bit at each clock pulse)
- In order to interpret the data correctly, the receiving device must know the start and end of each data unit.
- The transmitter must know the number of data units to be transferred and the receiver must be synchronized with the data boundaries.
- Therefore, there must be synchronization between the transmitter and receiver.
- Usually one or more SYNC characters are used to indicate the start of each synchronous data stream or frame of data.
- Transmitter sends a large block of data characters one after the other with no time between characters.

# Serial data transmission

- Transmitting device sends data continuously to the receiving device.
- If the data is not ready to be transmitted, the line is held in marking condition.
- To indicate the start of transmission, the transmitter sends out one or more SYNC characters or a unique bit pattern called a flag, depending on the system being used.
- The receiving device waits for data, when it finds the SYNC characters or the flag then starts interpreting the data which shifts the data following the SYNC characters and converts them to parallel form so they can be read in by a computer.

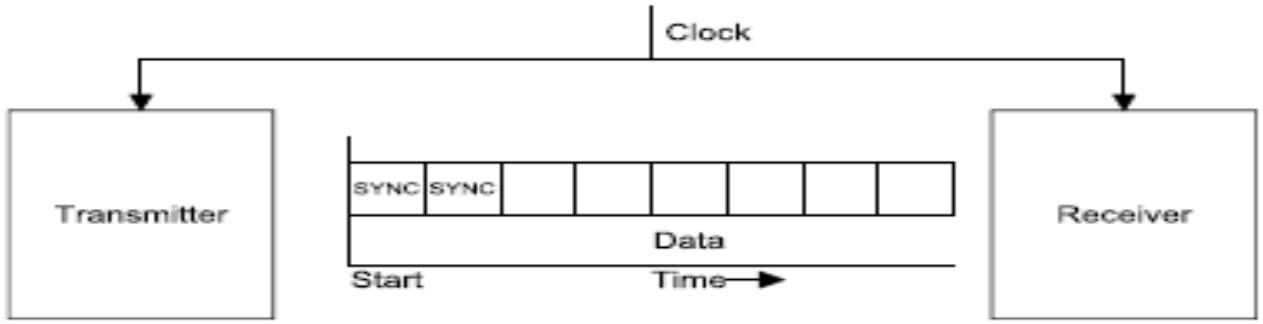



Fig: Synchronous Serial Transmission Format

# Serial data transmission

- **Advantages and Disadvantages of Synchronous Communication**
- Main advantage of Synchronous data communication is the high speed. The synchronous communications require high-speed peripherals/devices and a good-quality, high bandwidth communication channel.
- The disadvantage includes the possible in accuracy. Because when a receiver goes out of Synchronization, loosing tracks of where individual characters begin and end. Correction of errors takes additional time.

# Serial data transmission

**Serial Asynchronous Data Transmission**

- The receiving device does not need to be synchronized with the transmitting device.
- The transmitting device can send one or more data units when it is ready to send data.
- Each data unit must be formatted i.e. must contain start and stop bits for indicating beginning and the end of data unit. It also includes one parity bit to identify odd or even parity of data.
- To send ASCII character, the framing of data should contain:
  - 1 start bit: Beginning of data
  - 8 bit character: Actual data transferred
  - 1 or 2 stop bits: End of data
- When no data is being sent, the signal line is in a constant high or marking state.

# Serial data transmission

- The beginning of the data character is indicated by the line going low for 1 bit time and this bit is called a start bit.
- The data bits are then sent out on the line one after the other where the least significant bit is sent out first.
- Parity bit should contain to check for errors in received data.
- After the data bit and a parity bit, the signal line is returned high for at least 1 bit time to identify the end of the character, this always high bit is referred to as a stop bit. Some older systems use 2 stop bits.
- Asynchronous communication is used when slow speed peripherals communicate with the computer.
- The main disadvantage of asynchronous communication is slow speed transmission.

# Serial data transmission

- Asynchronous communication however, does not require the complex and costly hardware equipment's as is required for synchronous transmission.

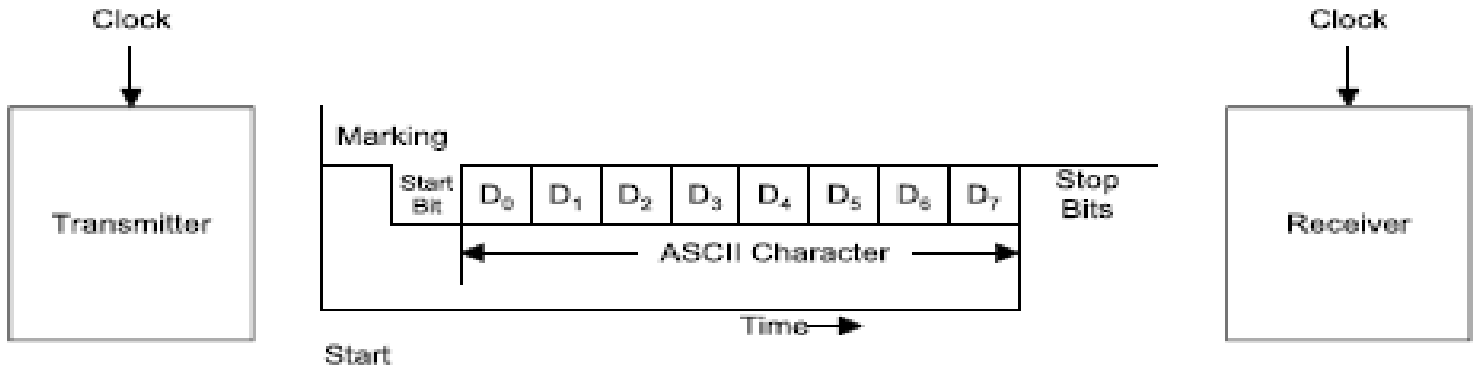



Fig: Asynchronous Serial Transmission Format

# Serial data transmission

- Synchronous versus Asynchronous serial data transmission

| S.N. | Parameter | Asynchronous | Synchronous |
|---|---|---|---|
| 1. | Fundamental | Transmission does not based on clock signal | Transmission based on clock signal |
| 2. | Data Format | One character at a time | Group of characters i.e. a block of characters |
| 3. | Speed | Low (< 20 kbps) | High (> 20 kbps) |
| 4. | Framing Information | Start and stop bits are sent with each character. | SYNC characters are sent with each character. |
| 5. | Implementation | Hardware / Software | Hardware |

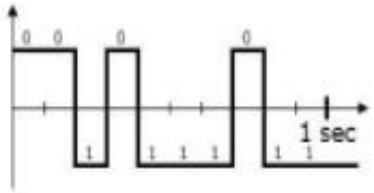
# Bit rate and baud rate

- **Bit Rate:**
  - Measure of no of data bits transmitted per sec.
  - E.g. 2400 bits per sec means 2400 zeros or ones can be transmitted in one sec.
- **Baud Rate:**
  - No of times a signal in a communication channel changes state.
  - Change state means change from 0 to 1 or from 1 to 0 up to 2400 times per sec.
  - E.g. 2400 baud rate means "the channel can change states up to 2400 time per sec"

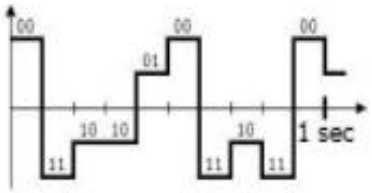
**Baud** → How many times a signal changes per second

**Bit rate** → How many bits can be sent per time unit (usually per second)

Bit rate is controlled by baud and number of signal levels




Baud = 10
Bit rate = 10 bps




Baud = 10
Bit rate = 20 bps

# Standards in serial i/o

- The serial I/O technique is commonly used to interface different peripheral terminals such as printers, modems with microcomputers which are designed and manufactured by various manufacturers.
- Therefore, a common understanding must exist, among various manufacturing and user groups that can ensure compatibility among different equipment.
- The standard is defined as the understanding which is accepted in industry and by users.
- A standard is normally defined by a professional organizations such as IEEE (Institute of Electrical and Electronics Engineers), EIA (Electronic Industries Association) as a de jure standard. However, a widespread practice can become a de facto standard.
- In serial I/O, data can be transmitted as either current or voltage.

# Rs-232c

- RS-232C is an interface developed to standardize the interface between data terminal equipment (DTE) and data communication equipment (DCE) employing serial binary data exchange.
- Modem and other devices used to send serial data are called data communication equipment (DCE).
- The computers or terminals that are sending or receiving the data are called data terminal equipment (DTE).
- RS- 232C is the interface standard developed by electronic industries Association (EIA) in response to the need for the signal and handshake standards between the DTE and DCE.

# Rs-232c

**RS-232C has following standardize features.**

- It uses 25 pins (DB – 25P) or 9 Pins (DE – 9P) standard, where 9 pins standard does not use all signals i.e. data, control, timing and ground.
- It describes the voltage levels, impendence levels, rise and fall times, maximum bit rate and maximum capacitance for all signal lines.
- It specifies that DTE connector should be male and DCE connector should be female.
- It can send 20kBd for a distance of 50 ft.
- The voltage level for RS-232 are:
- A logic high or 1 or mark, -3V to -15V
- A logic low or 0 or space, +3v to +15v
- Normally ±12V voltage levels are used

# Rs-232c

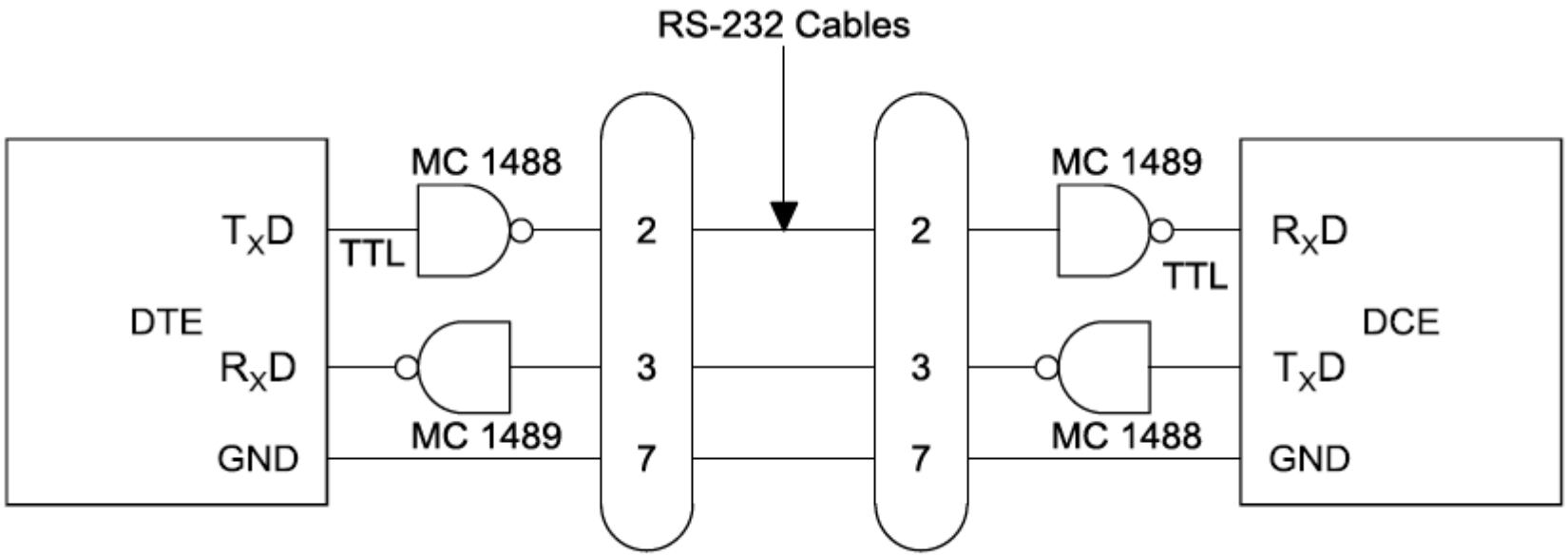



Fig: Connection of DTE and DCE through RS-232C Interface

- Mc1488 line driver converts logic 1 to -9V Logic 0 to +9v
- Mc1489 line receiver converts RS – 232 to TTL
- Signal levels of RS-232 are not compatible with that of the DTE and DCE which are TTL signals for that line driver such as M 1488 and line receiver MC1489 are used.

# Rs-232c

**RS- 232 signals used in handshaking:**

| Signal Flow | DE-9P | DB-25P | Signal | Description |
|---|---|---|---|---|
| - | 1 | - | Protective Ground | - |
| DTE to DCE | 3 | 2 | TxD | Transmitted Data |
| DCE to DTE | 2 | 3 | RxD | Received Data |
| DTE to DCE | 7 | 4 | $\overline{RTS}$ | Request To Send |
| DCE to DTE | 8 | 5 | $\overline{CTS}$ | Clear To Send |
| DCE to DTE | 6 | 6 | $\overline{DSR}$ | Data Set Ready |
| Common Ref | 5 | 7 | GND | Signal Ground |
| DCE to DTE | 1 | 8 | $\overline{DCD}$ | Data Carrier Detect |
| DTE to DCE | 4 | 20 | $\overline{DTR}$ | Data Terminal Ready |
| DCE to DTE | 9 | 22 | RI | Ring Indicator |
| DCE to DTE | - | 23 | DSRD | Data Signal Rate Detector |

# Rs-232c

- **Data Terminal Ready (DTR):**
  - After the terminal power is turned on and terminal runs any self checks, it asserts data terminal ready (DTR') signal to tell the modem that it is ready.
- **Data Set Ready (DSR):**
  - When the MODEM is powered up and ready to transmit or receive data, it will assert data set ready (DSR') to the terminal. Under manual control or terminal control, modem then dials up the computer. If the computer is available, it will send back a specified tone.
- **Request to send (RTS):**
  - When a terminal has a character ready to send, it will assert a request-to-send (RTS') signal to the modem.
- **Data Carrier Detect (DCD):**
  - The modem will then assert its data-carrier-detect (DCD') signal to the terminal to indicate that it has established connection with the computer.

# Rs-232c

- **Clear to send (CTS):**
  - When the modem is fully ready to receive data, it asserts the clear-to-send (CTS') signal back to the terminal.
- **Ring indicator (RI):**
  - It indicates that a ring has occurred at modem. Deactivating DTR or DSR breaks the connection but RI works independently of DTR i.e. a modem may activate RI signal even if DTR is not active.
- **Transmitted Data (TxD):**
  - The terminal then sends serial data characters to the modem.
- **Received Data (RxD):**
  - Modem will receive data from terminal through this line.
- **Data Signal Rate Detect (DSRD):**
  - It is used for switching different baud rate.

# Rs-232c

**Digital Data Transmission Using Modem and standard Phone Lines**

- Standard telephone system can be used for sending serial data over long distances.
- However, telephone lines are designed to handle voice, bandwidth of telephone lines ranges from 300 HZ
- to 3400 HZ.
- Digital signal requires a bandwidth of several megahertz. Therefore, data bits should be converted into audio tones, this is accomplished through modems.

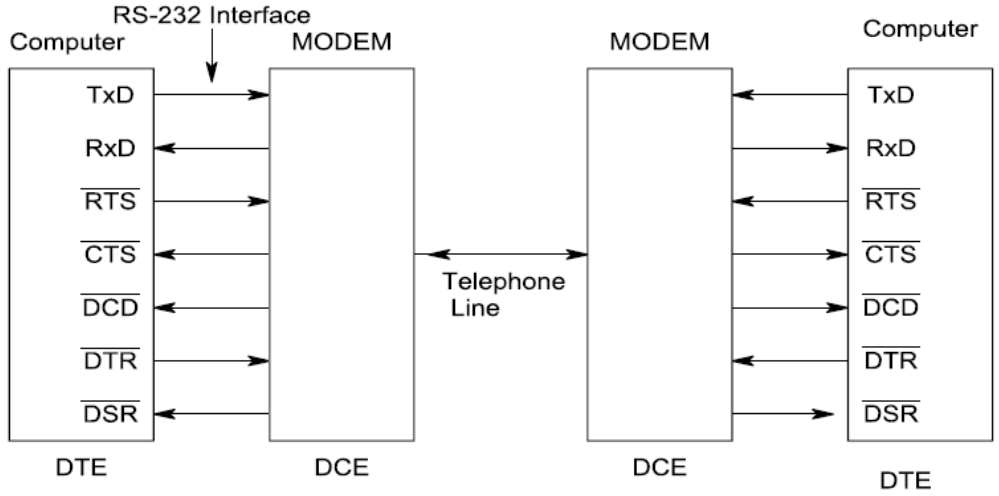



Fig: Digital Data transmission using MODEM and Telephone Line

# Rs-232c

- DTE asserts DTR’ to tell the modem it is ready.
- Then DCE asserts DSR’ signal to the terminal and dials up.
- DTE asserts RTS’ signal to the modem.
- Modem then asserts DCD’ signal to indicate that it has established connection with the computer.
- DCE asserts CTS’ signals, then DTE sends serial data.
- When sending completed, DTE asserts RTS’ high, this causes modem to un assert its CTS’ signal and stop transmitting similar handshake taken between DCE and DTE other side.
- To communicate from serial port of a computer to serial port of another computer without modem, null-modem is used.

# Rs-232c – null modem connection

- A zero modem serves for data exchange between DTEs.
- Since both the computers are configured as DTEs, directly connecting them by means of the conventional serial interface cable is impossible; not even the plug fits into the jack of the second terminal.
- Also the TxD meets TxD and RxD meets RxD, DTR meets DTR and DSR meets DSR etc.
- This means that outputs are connected to outputs and inputs are connected to inputs. With this convention, no data transfer is possible.
- For the transmission of data, it is required to twist the TxD and RxD lines.
- In this way, the transmitted data of one terminal (PC) becomes received data of other and vice versa.
- As shown in figure, activation of RTS to begin a data transfer gives rise to an activation of CTS on same DTE and to an activation of DCD on other DTE.

# Rs-232c – null modem connection

- Further, an activation of DTR leads to rise of DSR and RI on other DTE. Hence for every DTE, it is simulated that a DCE is on the end of line, although a connection between two DTEs is actually present.
- Zero modem can be operated with standard BIOS and DOS functions.

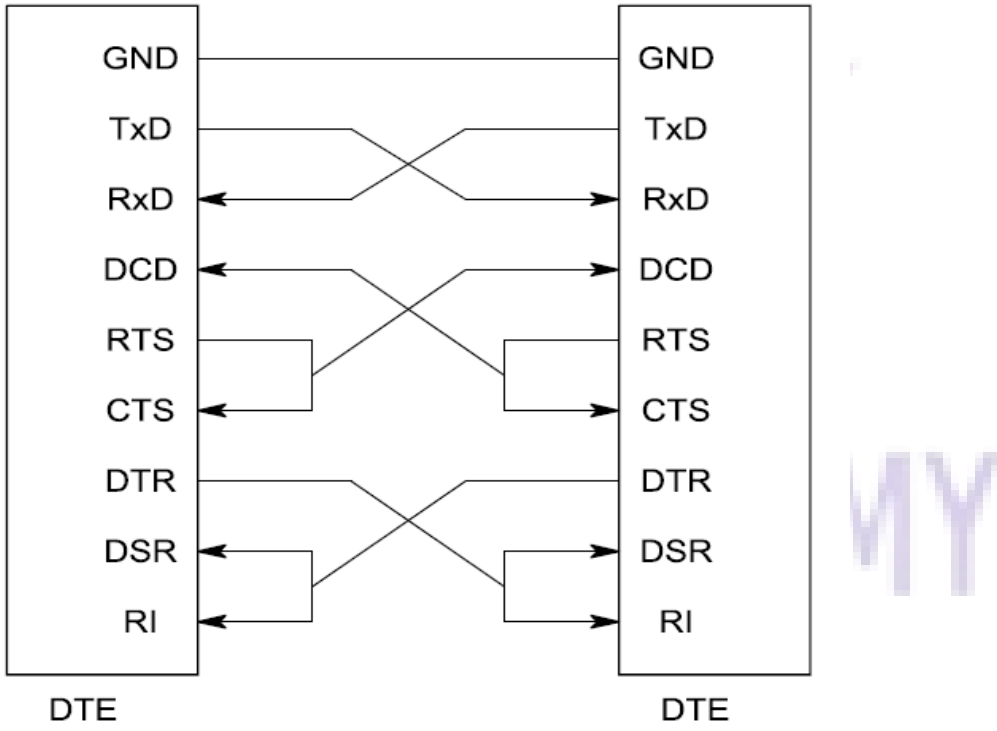



Fig: Null MODEM Connection for RS-232 Terminals

# Rs-232c – connection to printer

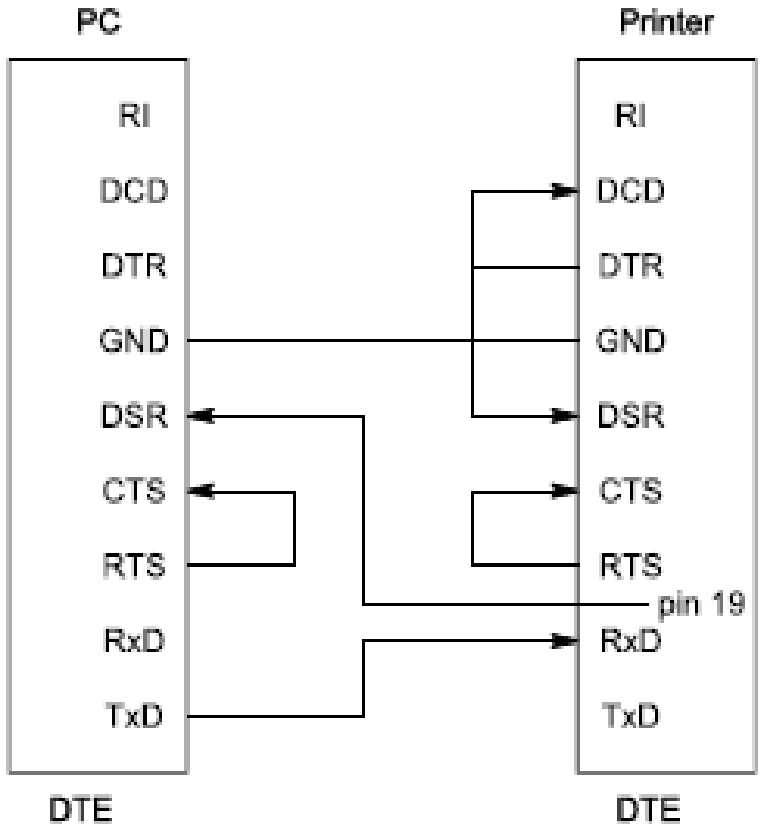



Fig: Connection to Printer



- PC may send data faster than the printer can acknowledge it.
- Therefore, pin 19 (Buffer Full) of printer is connected to DSR of PC side.

# Rs-232c – connection to printer

- An overflow of data deactivates the DSR signal and communication halts
- On PC RTS and CTS are connected to each other so that a transmission request from PC immediately enables the transmission.
- Printer as DTE refers to print anything as long as no active signal is present at inputs. Of CTS, DSR and DCD.
- This problem is resolved by connecting RTS with CTS and DTR with DCD and DSR.

# Rs-423a

- A major problem with RS-232C is that it can only transmit data reliably for about 50 ft at its maximum rate of 20Kbd.
- If longer lines are used the transmission rate has to be drastically reduced due to open signal lines with a common signal ground.
- Another EIA standard which is improvement over RS-232C is RS-423A.
- The standardize features of RS-423 are:
  - This standard specifies a low impendence single ended signal which can be sent over 50 ohm coaxial cable and partially terminated at the receiving end to prevent reflection.
  - Voltage levels
    - Logic High 4V - 6V negative
    - Logic Low 4V - 6V positive
  - It allows a maximum data rate of 100 Kbd over 40 ft line or a maximum baud rate of 1 Kbd over 4000 ft line.

# Rs-422a

It is a newer standard for serial data transfer. It specifies that each signal will be sent differentially over two adjacent wires in a ribbon cable or a twisted pair of wires uses differential amplifier to reject noise.

- The term differential in this standard means that the signal voltage is developed between two signal lines rather than between signal line and ground as in RS-232C and RS-423A.
- Any electrical noise induced in one signal line will be induced equally in the other signal line.
- A differential line receiver MC3486 responds only to the voltage difference between its two inputs so any noise voltage that is induced equally on two inputs will not have any effect on the output of the differential receiver.
- RS-422A has following standardized features:
  - Logic high is transmitted by making ‘b’ line more positive than ‘a’ line.
  - Logic low is transmitted by making ‘a’ line more positive than ‘b’ line.
  - The voltage difference between the two lines must be greater than 0.4V but less than 12V.

# comparison

**Comparison of Serial I/O Standards**

| S.N. | Specifications | RS-232C | RS-423A | RS-422A |
|---|---|---|---|---|
| 1. | Speed | 20 Kbaud | 100 Kbaud at 40 ft<br>1 kbaud at 4000 ft | 10 Mbaud at 40 ft<br>100 kbaud at 4000 ft |
| 2. | Distance | 50 ft | 4000 ft | 4000 ft |
| 3. | Logic 0 | +3 V to +25 V | +4 V to +6 V | B line > A line |
| 4. | Logic 1 | -3 V to -25 V | -4 V to –6 V | A line > B line |
| 5. | Receiver Input Voltage | ±15V | ±12V | ±7V |
| 6. | Mode of Operation | Single ended input and output | Differential input and single ended output | Differential input and output |
| 7. | Noise Immunity | 2.0 V | 3.4 V | 1.8 V |
| 8. | Input Impedance | 3-7 KOhm and 2500 pf | >4 KOhm | >4 KOhm |
| 9. | Short circuit current | 500 mA | 150 mA | 150 mA |

# DMA (DIRECT MEMORY ACCESS)

- The data transfer between a fast storage media such as magnetic disk and memory unit is limited by the speed of the CPU
- Thus we can allow the peripherals directly communicate with each other using the memory buses, removing the intervention of the CPU This type of data transfer technique is known as DMA or direct memory access
- During DMA the CPU is idle and it has no control over the memory buses
- The DMA controller takes over the buses to manage the transfer directly between the I/O devices and the memory unit

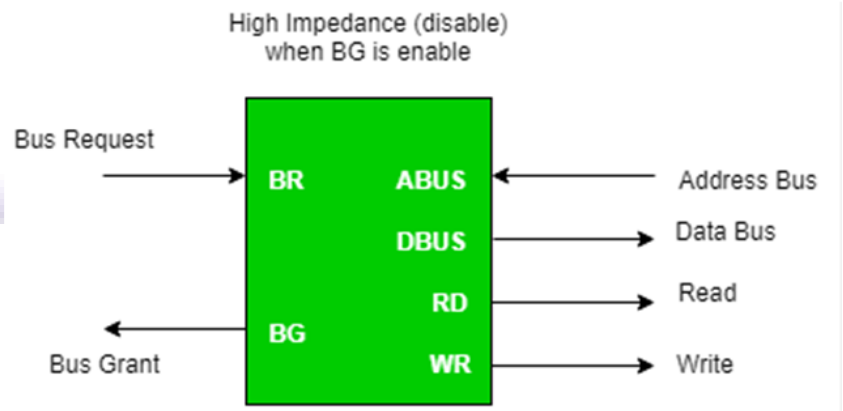



**Figure** - CPU Bus Signals for DMA Transfer

# DMA (DIRECT MEMORY ACCESS)

- **Bus Request** It is used by the DMA controller to request the CPU to relinquish the control of the buses
- **Bus Grant** It is activated by the CPU to inform the external DMA controller that the buses are in high impedance state and the requesting DMA can take control of the buses Once the DMA has taken the control of the buses it transfers the data
- DMA transfer uses two signal
  - **HOLD**
  - **HLDA**

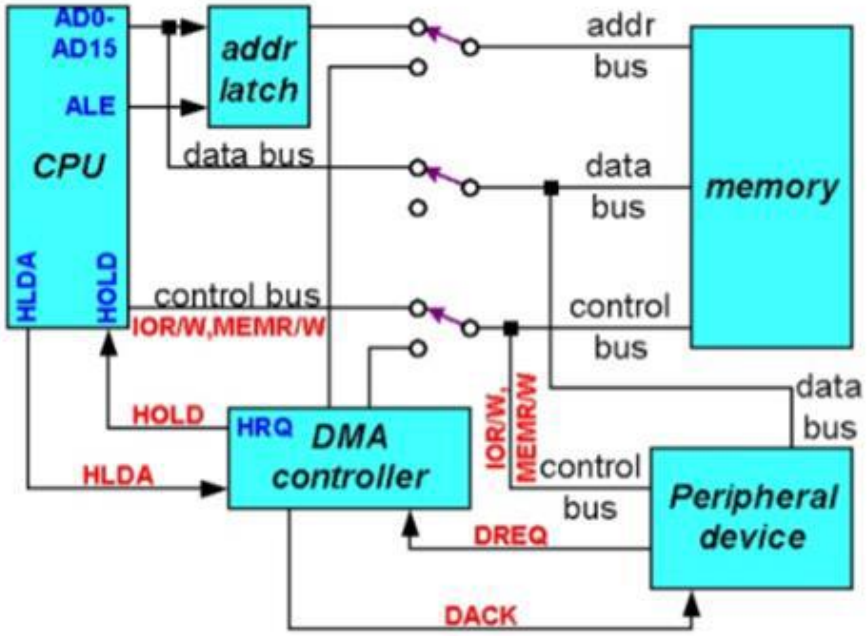
# DMA (DIRECT MEMORY ACCESS)

- **HOLD**
  - Active high input signal to 8085 from another master requesting the use of address and data bus
  - After receiving the HOLD request, the MPU relinquishes the buses in the following machine cycle
  - All buses are tri stated and HOLD acknowledge signal is sent out
  - MPU regains the control of the buses after HOLD goes low
- **HLDA**
  - This is an active high output signal indicting that MPU is relinquishing control of the buses
  - A DMA controller uses these signals as if it were a peripheral requesting the MPU for the control of the buses

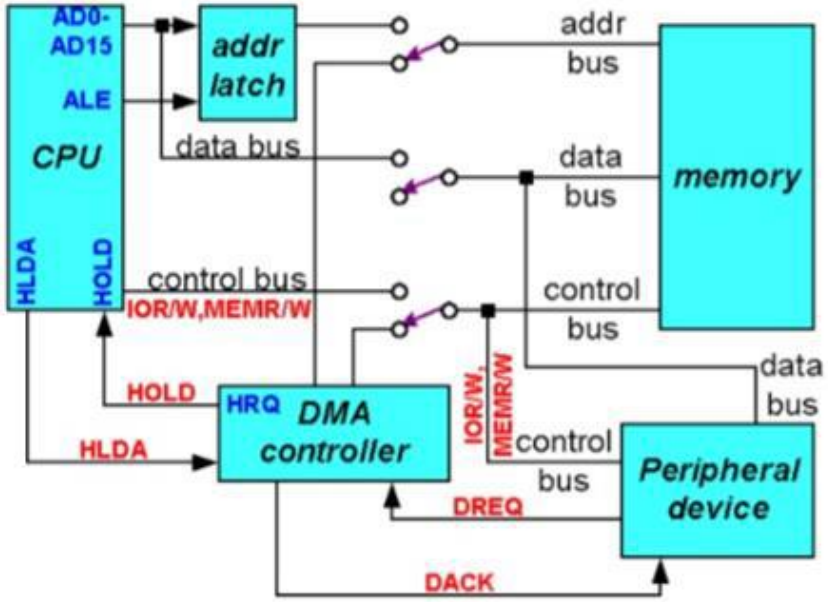
# DMA (DIRECT MEMORY ACCESS)

The sequence of DMA Transfer

CPU having the control over the bus:




When DMA operates:



# DMA (DIRECT MEMORY ACCESS)

**The sequence of DMA Transfer**

- Originally, microprocessor is connected to the memory as shown in fig with switches closed for address, data and control buses When peripheral wants to transfer data using DMA transfer, it sends DMA request, DREQ, signal to the DMA controller
- If the input ( of the DMA controller is unmasked, the DMA controller will send a hold request, HRQ signal to the microprocessors HOLD input
- The microprocessor finishes the current machine cycle and floats its buses, sending out a hold acknowledge signal, HLDA, to the DMA controller

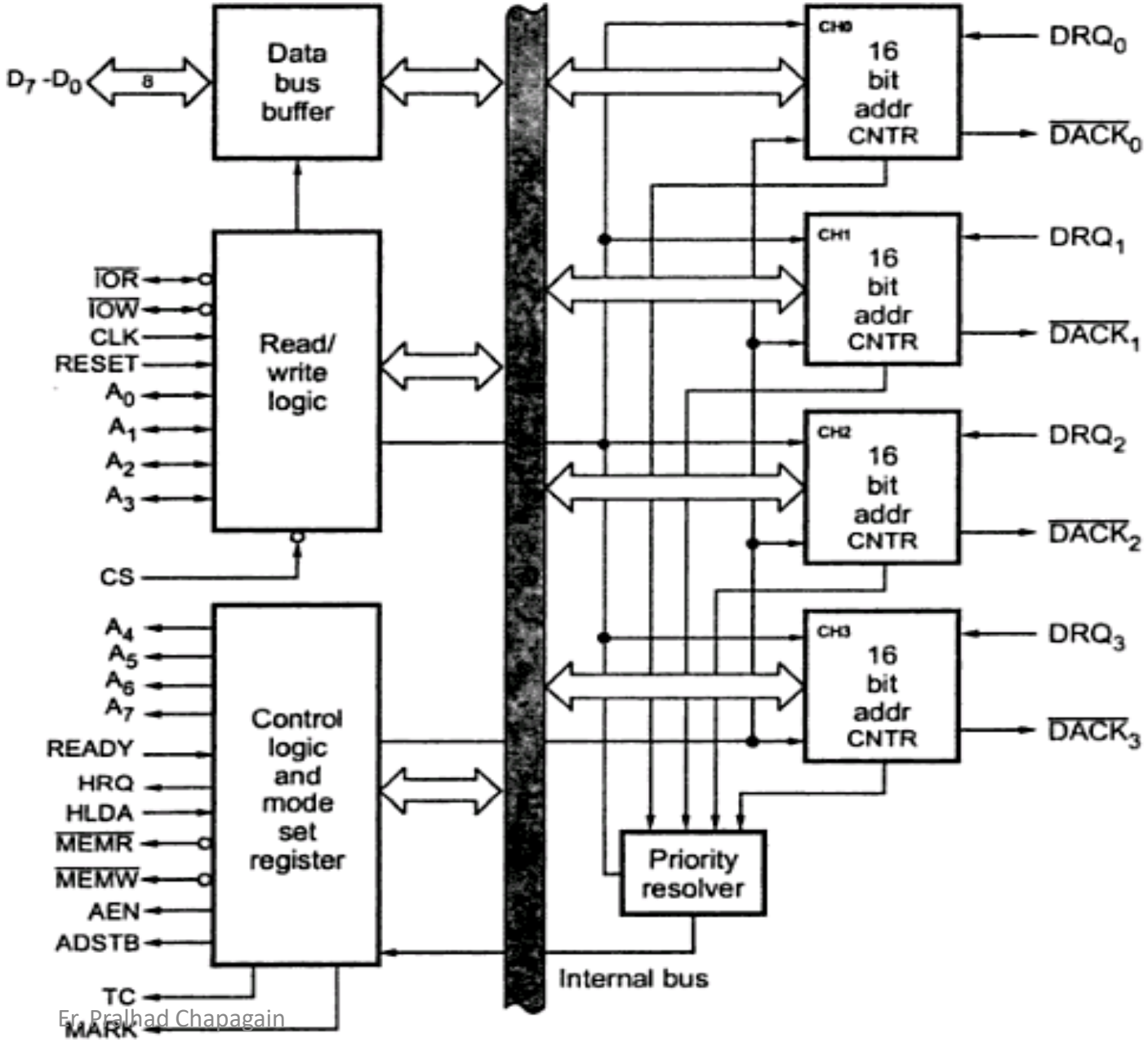
# DMA (DIRECT MEMORY ACCESS)

- When DMA controller receives HLDA signal, it will send out a control signal which disconnects the processors from buses and connects DMA controller to the buses Now DMA controller sends out the address of the byte to be transferred and send out DMA acknowledge ( to the peripheral device to tell it to get ready to output the byte
- Then the DMA transfer begins and finally when the data transfer is complete, the DMA controller un asserts its hold request signal to the processor and releases the buses

# DMA (DIRECT MEMORY ACCESS)

- DMA performs data transfer operation The different DMA transfer modes are as follows
  - Burst or Block transfer DMA
  - Cycle steal or Single byte transfer DMA
  - Transparent DMA

# PROGRAMMABLE DMA CONTROLLER – INTEL 8257



# PROGRAMMABLE DMA CONTROLLER – INTEL 8257

- It is a device to transfer the data directly between IO device and memory without through the CPU. So it performs a high-speed data transfer between memory and I/O device.
- The features of 8257 is,
  - The 8257 has four channels and so it can be used to provide DMA to four I/O devices.
  - Each channel can be independently programmable to transfer up to 64kb of data by DMA.
  - Each channel can be independently perform read transfer, write transfer and verify transfer.
- The functional blocks of 8257 as shown in the above figure are data bus buffer, read/write logic, control logic, priority resolver and four numbers of DMA channels.

# PROGRAMMABLE DMA CONTROLLER – INTEL 8257

## Operation of 8257 DMA Controller

- Each channel of 8257 has two programmable 16-bit registers named as address register and count register.
- Address register is used to store the starting address of memory location for DMA data transfer.
- The address in the address register is automatically incremented after every read/write/verify transfer.
- The count register is used to count the number of byte or word transferred by DMA.

# PROGRAMMABLE DMA CONTROLLER – INTEL 8257

- In read transfer the data is transferred from memory to I/O device.
- In write transfer the data is transferred from I/O device to memory.
- Verification operations generate the DMA addresses without generating the DMA memory and I/O control signals.
- The 8257 has two eight bit registers called mode set register and status register.

# Microprocessor System

1. Which of the following is the smallest entity of memory?

(a) Block
(b) Cell
(c) Instance
(d) Set

2. The primary memory (also called main memory) of a personal computer consists of

(a) RAM only
(b) ROM only
(c) both RAM and ROM
(d) Cache memory

3. The Boot sector files of the system are stored in which computer memory?

(a) RAM
(b) ROM
(c) Cache
(d) Register

4. Which of the following statements are not correct about the main memory of a computer?

(a) In main memory, data gets lost when power is switched off.
(b) Main memory is faster than secondary memory but slower than registers.
(c) They are made up of semiconductors.
(d) All are correct

9. Which of the following is the lowest in the computer memory hierarchy?

(a) Cache
(b) RAM
(c) Secondary memory
(d) CPU registers

10. Which of the following has the fastest speed in the computer memory hierarchy?

(a) Cache
(b) Register in CPU
(c) Main memory
(d) Disk cache

# Microprocessor System

11. Which memory acts as a buffer between CPU and main memory?

(a) RAM
(b) ROM
(c) Cache
(d) Storage

12. Which of the following statements are not correct about cache memory?

(a) Cache memory is used to store data temporarily.
(b) It holds that data and program which has to be executed within a short period of time.
(c) It consumes less access time as compared to the RAM.
(d) All are correct.

15. Which computer memory chip allows simultaneous both read and write operations?

(a) ROM
(b) RAM
(c) PROM
(d) EEPROM

16. In which type of memory, once the program or data is written, it cannot be changed?

(a) EPROM
(b) PROM
(c) EEPROM
(d) None of these

17. In which type of ROM, data can be erased by ultraviolet light and then reprogrammed by the user or manufacturer?

(a) PROM
(b) EPROM
(c) EEPROM
(d) Both a and b

19. How many types of RAM are available?

(a) 4
(b) 3
(c) 2
(d) 5

# Microprocessor System

1. What is true about memory unit?

A. A memory unit is the collection of storage units or devices together.
B. The memory unit stores the binary information in the form of bits.
C. Both A and B
D. None of the above

5. What is the formula for Hit Ratio?

A. Hit/(Hit + Miss)
B. Miss/(Hit + Miss)
C. (Hit + Miss)/Miss
D. (Hit + Miss)/Hit

9. Which of the following is correct refreshed rate for DRAM?

A. 10~1000 ms
B. 10~50 ms
C. 10~100 ms
D. 10~500 ms

5. The memory which is used to store the copy of data or instructions stored in larger memories, inside the CPU is called _____

a) Level 1 cache
b) Level 2 cache
c) Registers
d) TLB

13. Suppose that a certain semiconductor memory chip has a capacity of 8K × 8. How many bytes could be stored in this device?

a) 8,000
b) 65,536
c) 8,192
d) 64,000

15. What is the difference between static RAM and dynamic RAM?

a) Static RAM must be refreshed, dynamic RAM does not
b) There is no difference
c) Dynamic RAM must be refreshed, static RAM does not
d) SRAM is slower than DRAM

# Microprocessor System

1. The device that enables the microprocessor to read data from the external devices is
a) printer
b) joystick
c) display
d) reader

3. The input and output operations are respectively similar to the operations,
a) read, read
b) write, write
c) read, write
d) write, read

4. The operation, IOWR (active low) performs
a) write operation on input data
b) write operation on output data
c) read operation on input data
d) read operation on output data

10. In memory-mapped scheme, the devices are viewed as
a) distinct I/O devices
b) memory locations
c) only input devices
d) only output devices

2. If a location is selected, then all the bits in it are accessible using a group of conductors called
a) control bus
b) address bus
c) data bus
d) either address bus or data bus

3. To address a memory location out of N memory locations, the number of address lines required is
a) log N (to the base 2)
b) log N (to the base 10)
c) log N (to the base e)
d) log (2N) (to the base e)

# Microprocessor System

4. If the microprocessor has 10 address lines, then the number of memory locations it is able to address is
a) 512
b) 1024
c) 2048
d) none

8. To obtain 16-bit data bus width, the two 4K*8 chips of RAM and ROM are arranged in
a) parallel
b) serial
c) both serial and parallel
d) neither serial nor parallel

2. Whenever a large memory is required in a microcomputer system, the memory subsystem is generally designed using
a) Static RAM
b) Dynamic RAM
c) Both static and dynamic RAM
d) ROM

3. If a typical static RAM cell requires 6 transistors then corresponding dynamic RAM requires
a) 1 transistor along with capacitance
b) 2 transistors along with resistance
c) 3 transistors along with diode
d) 2 transistors along with capacitance

5. The process of refreshing the data in the RAM to reduce the possibility of data loss is known as
a) data cycle
b) regain cycle
c) retain cycle
d) refresh cycle

10. If 'n' denotes the number of rows that are to be refreshed in a single refresh interval, 'td' denotes the range of time it may take then, refresh time (tr) can be defined as
a) n*td
b) td/n
c) n/td
d) $td^n$

# Microprocessor System

2. Port C of 8255 can function independently as
a) input port
b) output port
either input or output ports
d) both input and output ports

3. All the functions of the ports of 8255 are achieved by programming the bits of an internal register called
a) data bus control
b) read logic control
control word register
d) none of the mentioned

4. The data bus buffer is controlled by
a) control word register
read/write control logic
c) data bus
d) none of the mentioned

5. The input provided by the microprocessor to the read/write control logic is
a) RESET
b) A1
c) WR(ACTIVE LOW)
All of the mentioned

6. The device that receives or transmits data upon the execution of input or output instructions by the microprocessor is
a) control word register
b) read/write control logic
3-state bidirectional buffer
d) none of the mentioned

8. If A1=0, A0=1 then the input read cycle is performed from
a) port A to data bus
port B to data bus
c) port C to data bus
d) CWR to data bus

# Microprocessor System

10. The pin that clears the control word register of 8255 when enabled is
a) CLEAR
b) SET
RESET
d) CLK

**Question 1: How many pins does the 8255 PPI IC contains?**

a. 24
b. 20
c. 32
40

**Question 2: In which mode do all the Ports of the 8255 PPI work as Input-Output units for data transfer?**

a. BSR mode
. Mode 0 of I/O mode
c. Mode 1 of I/O mode
d. Mode 2 of I/O mode

**Question 3: Which of the following pins are responsible for handling the on the Read Write control logic unit of the 8255 PPI?**

a. CS'
b. RD'
c. WR'
ALL of the above

**Question 4: In which of the following modes is the 8255 PPI capable of transferring data while handshaking with the interfaced device?**

a. BSR mode
b. Mode 0 of I/O mode
. Mode 1 of I/O mode
d. Mode 2 of I/O mode

**Question 5: How many bits of data can be transferred between the 8255 PPI and the interfaced device at a time? or What is the size of internal bus of the 8255 PPI?**

a. 16 bits
b. 12 bits
8 bits
d. None of the above

# Microprocessor System

**Question 6: Which port of the 8255 PPI is capable of performing the handshaking function with the interfaced devices?**

a. Port A
b. Port B
c. Port C
d. All of the above

**Question 7: In which of the following modes of the 8255 PPI, only port C is taken into consideration?**

a. BSR mode
b. Mode 0 of I/O mode
c. Mode 1 of I/O mode
d. Mode 2 of I/O mode

**Question 8: In mode 2 of I/O mode, which of the following ports are capable of transferring the data in both the directions?**

a. Port A
b. Port B
c. Port C
d. All of the above

**Question 9: In which of the following modes we do not consider the D6, D5 and D4 bits of the control word?**

a. BSR mode
b. Mode 0 of I/O mode
c. Mode 1 of I/O mode
d. Mode 2 of I/O mode

**Question 13: The 8255 ports works in the I/O mode,**

a. Programmable I/O ports
b. Set pins
c. Reset pins
d. None of these

**Question 16: Strobed input/output mode is also known as -**

a. Mode 0
b. Mode 1
c. Mode 2
d. None of these

# Microprocessor System

1. The serial communication is
a) cheaper communication
b) requires less number of conductors
c) slow process of communication
d) all of the mentioned

2. The serial communication is used for
a) short distance communication
b) long distance communication
c) short and long distance communication
d) communication for a certain range of distance

1. In _____ transmission of data, a group of n bits are sent simultaneously.

A. parallel

B. synchronous

C. asynchronous

D. serial

3. Serial transmission can be _____

A. parallel

B. synchronous

C. asynchronous

D. b or c

4. _____ transmission features start bits, stop bits, and gaps between data units

A. Parallel

B. Synchronous

C. Asynchronous

D. Virtual

6. Data transmission between the keyboard and the computer is usually _____

A. parallel

B. synchronous

C. asynchronous

D. virtual

# Microprocessor System

7. Serial transmission without stop bits, start bits, or gaps is called _____ transmission

A. parallel

B. synchronous

C. asynchronous

D. virtual

8. In parallel transmission, if we are sending 80 Kbps (on each line), we send _____ characters per second

B. 80,000

B. 10,000

C. 40,000

D. none of above

5. The task of converting the byte into serial form and transmitting it bit by bit along with start, stop and parity bits is carried out by

a) reception unit

b) serial communication unit

c) transmission unit

d) all of the mentioned

**1. Which of the following can be used for long distance communication?**

A. I2C

B. Parallel port

C. SPI

D. RS232

**4. Which of the following can provide hardware handshaking?**

A. RS232

B. Parallel port

C. Counter

D. Timer

1. RS-232 is now known as _____

A. OSI-232

B. EIA-232

C. ITU-232

D. IEEE-232

# Microprocessor System

12. A null modem has _____ connector(s).

A. one male and one female

B. two male

two female

D. only one

7. In the DB-25 implementation, the majority of the EIA-232 interface wires are used for _____

A. data transmission

. control and timing

C. test mode

D. future assignment

2. The most popular implementation of EIA-232 is a _____-pin interface

A. 23

25

C. 232

D. 9

11. In a null modem, the _____ pins are rewired so that DTEs can directly connect to each other

A. data transfer

B. set up

C. timing

. all of the above

13. The _____ interface standard specifies a 4000-feet transmission distance

A. EIA-232

B. RS-422

C. EIA-530

D. X.21

5. The voltage level and the signal type are _____ specifications of the EIA-232 interface standard

A. mechanical

electrical

C. functional

D. any of the above

# Microprocessor System

2. The DMA transfers are performed by a control circuit called as _____
a) Device interface
b) DMA controller
c) Data controller
d) Overlooker

1. Which of the following provides an efficient method for transferring data from a peripheral to memory?
A. dma controller
B. serial port
C. parallel port
D. dual port

**4). For transfer of data, devices inform the DMA through——?**

- Bus request signal.
- DRQ.
- HLDA
- DAK

**5). DMA acts as—— to the CPU for data transfer.**

- Master
- Slave.
- Both a and b.
- None of the above



3. In DMA transfers, the required signals and addresses are given by the _____
a) Processor
b) Device drivers
c) DMA controllers
d) The program itself

4. After the completion of the DMA transfer, the processor is notified by ____
a) Acknowledge signal
b) Interrupt signal
c) WMFC signal
d) None of the mentioned

**2.6 Interrupt operations**: Interrupt, Interrupt Service Routine, and Interrupt Processing. (AExE0206)

# INTERRUPTS

- Interrupt is signals send by an external device to the processor, to request the processor to perform a particular task or work.
- Mainly in the microprocessor based system the interrupts are used for data transfer between the peripheral and the microprocessor.
- The processor will check the interrupts always at the 2nd T-state of last machine cycle.
- If there is any interrupt it accept the interrupt and send the INTA (active low) signal to the peripheral.
- The vectored address of particular interrupt is stored in program counter.
- The processor executes an interrupt service routine (ISR) addressed in program counter.
- It returned to main program by IRET instruction.

# INTERRUPTS

- **Need for Interrupt:**
  - Interrupts are particularly useful when interfacing I/O devices that provide or require data at relatively low data transfer rate.

# INTERRUPT OPERATIONS

- The transfer of data between the microprocessor and input /output devices takes place using various modes of operations like programmed I/O, interrupt I/O and direct memory access.
- In programmed I/O, the processor has to wait for a long time until I/O module is ready for operation.
- So the performance of entire system degraded.
- To remove this problem CPU can issue an I/O command to the I/O module and then go to do some useful works. The I/O device will then interrupt the CPU to request service when it is ready to exchange data with CPU.
- In response to an interrupt, the microprocessor stops executing its current program and calls a procedure which services the interrupt.

# INTERRUPT OPERATIONS

- The interrupt is a process of data transfer whereby an external device or a peripheral can inform the processor that it is ready for communication and it requests attention.
- The response to an interrupt request is directed or controlled by the microprocessor.

# INTERRUPT OPERATIONS

- **Process of interrupt Operation**
- **From the point of view of I/O unit**
  - I/O device receives command from CPU
  - The I/O device then processes the operation
  - The I/O device signals an interrupt to the CPU over a control line.
  - The I/O device waits until the request from CPU.
- **From the point of view of processor**
  - The CPU issues command and then goes off to do its work.
  - When the interrupt from I/O device occurs, the processor saves its program counter & registers of the current program and processes the interrupt.
  - After completion for interrupt, processor requires its initial task.

# POLLING VERSUS INTERRUPT

- Each time the device is given a command, for example ``move the read head to sector 42 of the floppy disk" the device driver has a choice as to how it finds out that the command has completed. The device drivers can either poll the device or they can use interrupts.
- Polling the device usually means reading its status register every so often until the device's status changes to indicate that it has completed the request.
- Polling means the CPU keeps checking a flag to indicate if something happens.
- An interrupt driven device driver is one where the hardware device being controlled will cause a hardware interrupt to occur whenever it needs to be serviced.

# POLLING VERSUS INTERRUPT

- With interrupt, CPU is free to do other things, and when something happens, an interrupt is generated to notify the CPU. So it means the CPU does not need to check the flag.
- Polling is like picking up your phone every few seconds to see if you have a call. Interrupts are like waiting for the phone to ring.
- Interrupts win if processor has other work to do and event response time is not critical.
- Polling can be better if processor has to respond to an event ASAP; may be used in device controller that contains dedicated secondary processor.

# POLLING VERSUS INTERRUPT

- **Advantages of interrupt over Polling**
- Interrupts are used when you need the fastest response to an event. For example, you need to generate a series of pulses using a timer. The timer generates an interrupt when it overflows and within 1 or 2 sec, the interrupt service routine is called to generate the pulse. If polling were used, the delay would depend on how often the polling is done and could delay response to several msecs. This is thousands times slower.
- Interrupts are used to save power consumption. In many battery powered applications, the microcontroller is put to sleep by stopping all the clocks and reducing power consumption to a few micro amps. Interrupts will awaken the controller from sleep to consume power only when needed. Applications of this are hand held devices such as TV/VCR remote controllers.
- Interrupts can be a far more efficient way to code. Interrupts are used for program debugging.

# INTERRUPT STRUCTURES

- A processor is usually provided with one or more interrupt pins on the chip.
- Therefore a special mechanism is necessary to handle interrupts from several devices that share one of these interrupt lines. There are mainly two ways of servicing multiple interrupts which are polled interrupts and daisy chain (vectored) interrupts.

# POLLED INTERRUPT

- Polled interrupts are handled by using software which is slower than hardware interrupts.
- Here the processor has the general (common) interrupt service routine (ISR) for all devices.
- The priority of the devices is determined by the order in which the routine polls each device.
- The processor checks the starting with the highest priority device.
- Once it determines the source of the interrupt, it branches to the service routine for that device.

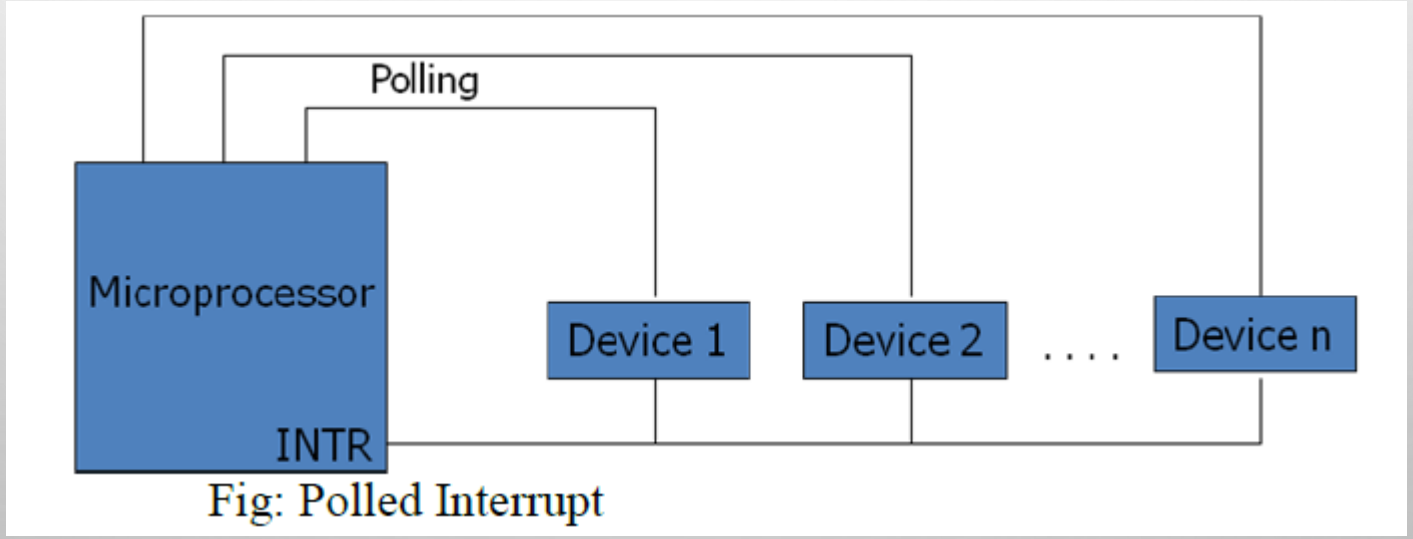


Fig: Polled Interrupt

# POLLED INTERRUPT

- Here several eternal devices are connected to a single interrupt line (INTR) of the microprocessor.
- When INTR signal goes up, the processor saves the contents of PC and other registers and then branches to an address defined by the manufactures of the processor.
- The user can write a program at this address to find the source of the interrupt by starting the polled from highest priority device.

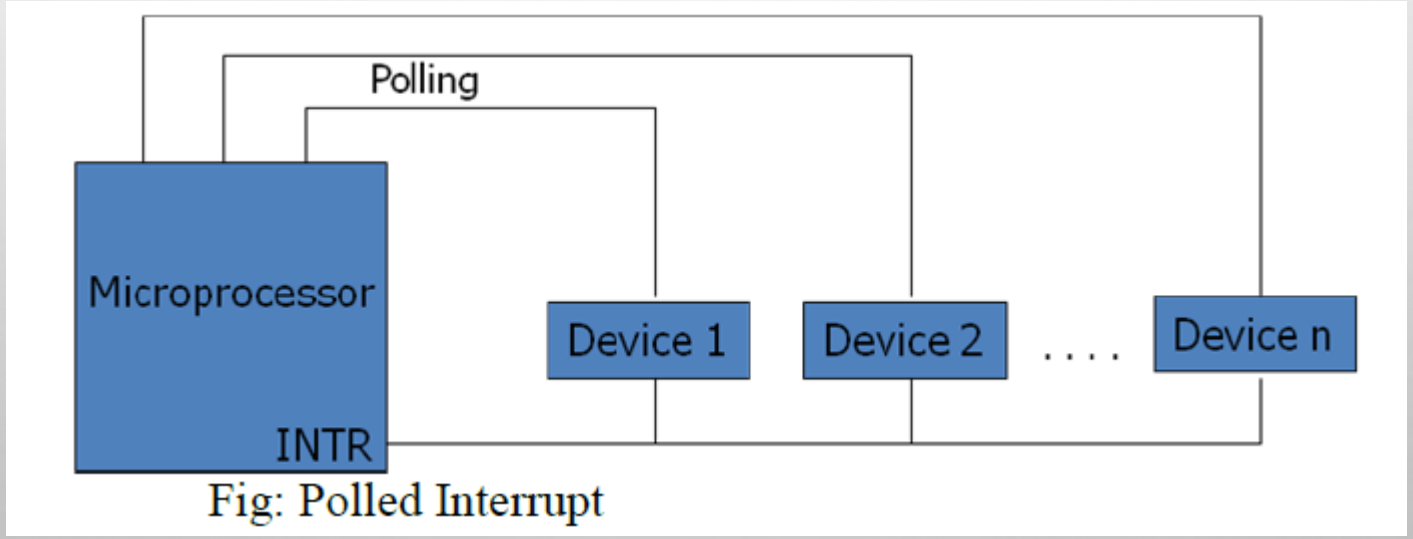


Fig: Polled Interrupt

# DAISY CHAIN (VECTORED) INTERRUPT

- In polled interrupt, the time required to poll each device may exceed the time to service the device through software.
- To improve this, the faster mechanism called vectored or daisy chain interrupt is used.
- Here the devices are connected in chain fashion.
- When INTR pin goes up, the processor saves its current status and then generates INTA signal to the highest priority device.
- If this device has generated the interrupt, it will accept the INTA; otherwise it will push INTA to the next priority device until the INTA is accepted by the interrupting device

# DAISY CHAIN (VECTORED) INTERRUPT

- When INTA is accepted, the device provides a means to the processor for findings the interrupt address vector using external hardware.
- The accepted device responds by placing a word on the data lines which becomes the vector address with the help of any hardware through which the processor points to appropriate device service routine.
- Here no general interrupt service routine need first that means appropriate ISR of the device will be called.

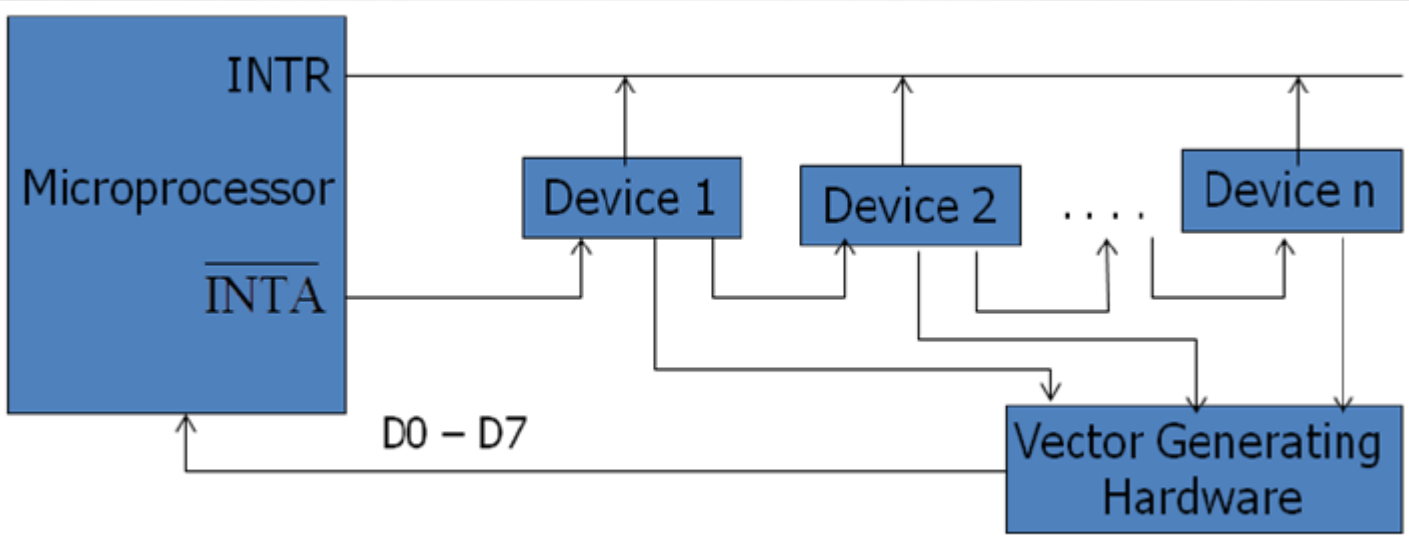


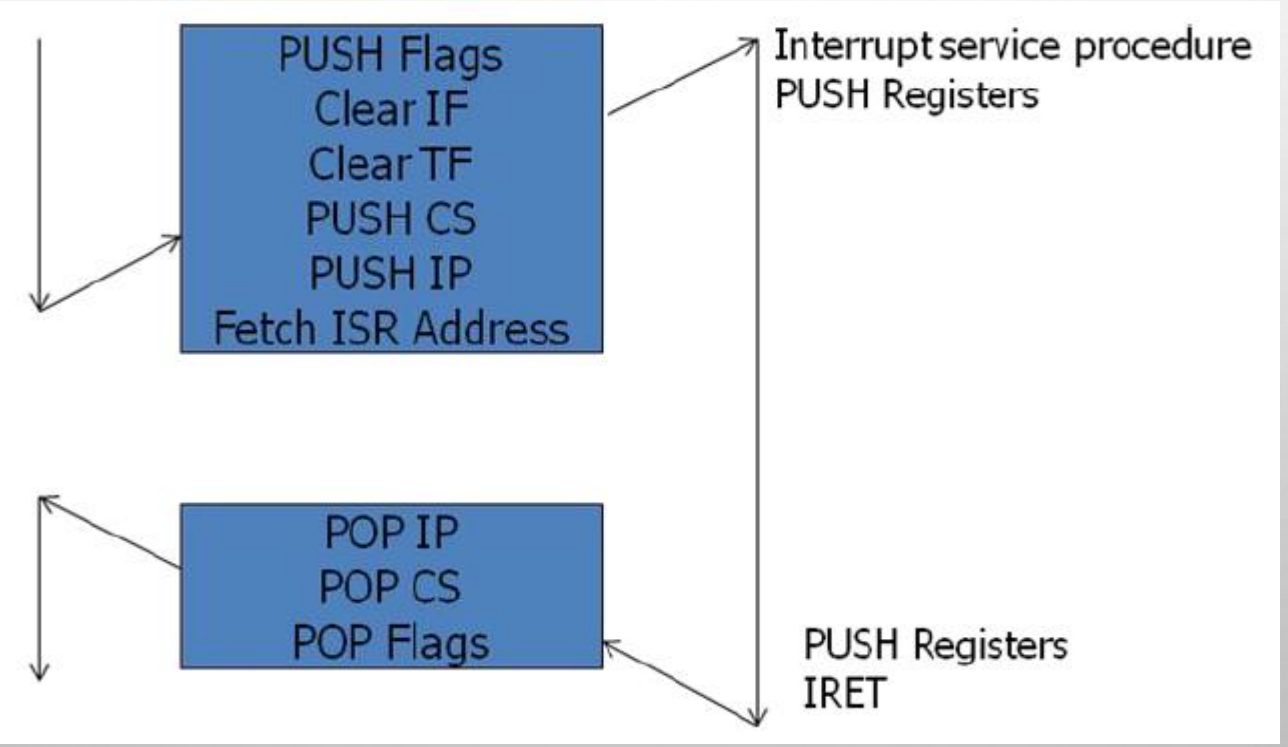

Fig: Vectored (Daisy Chain) Interrupt

# INTERRUPT PROCESSING SEQUENCE

- The occurrence of interrupt triggers a number of events, both in processor hardware and in software. The interrupt driven I/O operation takes the following steps.
  - The I/O unit issues an interrupt signal to the processor for exchange of data between them.
  - The processor finishes execution of the current instruction before responding to the interrupt.
  - The processor sends an acknowledgement signal to the device that it issued the interrupt.
  - The processor transfers its control to the requested routine called “Interrupt Service Routine (ISR)” by saving the contents of program status word (PSW) and program counter (PC).
  - The processor now loads the PC with the location of interrupt service routine and the fetches the instructions. The result is transferred to the interrupt handler program.
  - When interrupt processing is completed, the saved register’s value are retrieved from the stack and restored to the register.
  - Finally it restores the PSW and PC values from the stack.

# INTERRUPT PROCESSING SEQUENCE

- The figure summarizes these steps.
- The processor pushes the flag register on the stack, disables the INTR input and does essentially an indirect call to the interrupt service procedure.
- An IRET function at the end of interrupt service procedure returns execution to the main program.



# INTERRUPT - TYPES

## 1. External interrupts:

- These interrupts are initiated by external devices such as A/D converters and classified on following types.
- Maskable interrupt :
  - It can be enabled or disabled by executing instructions such as EI and DI. In 8085, EI sets the interrupt enable flip flop and enables the interrupt process. DI resets the interrupt enable flip flop and disables the interrupt.
- Non-maskable interrupt:
  - It has higher priority over maskable interrupt and cannot be enabled or disabled by the instructions.

# INTERRUPT - TYPES

**2. Internal interrupts:**

- These are indicated internally by exceptional conditions such as overflow, divide by zero, and execution of illegal op-code.
- The user usually writes a service routine to take correction measures and to provide an indication in order to inform the user that exceptional condition has occurred.
- There can also be activated by execution of TRAP instruction. This interrupt means TRAP is useful for operating the microprocessor in single step mode and hence important in debugging.
- These interrupts are used by using software to call the function of an operating system. Software interrupts are shorter than subroutine calls and they do not need the calling program to know the operating system's address in memory.

# MULTIPLE INTERRUPTS DEALING APPROACHES:

- If the processor gets multiple interrupts, then we need to deal these interrupts one at a time and the dealing approaches are:

**Sequential processing of interrupts**

- When user program is executing and an interrupt occurs interrupts are disabled immediately. After the interrupt service routine completes, interrupts are enabled before resuming the user program and the processor checks to see if additional interrupts have occurred.

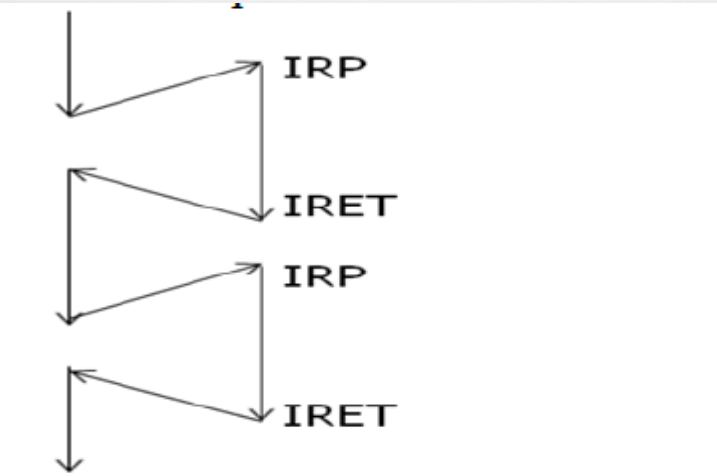


Fig: Sequential Interrupt Service

# MULTIPLE INTERRUPTS DEALING APPROACHES:

**Priority wise processing of interrupts:**

- The drawback of sequential processing is that it does not take account of relative priority or time critical needs.
- The alternative form of this is to define priorities for interrupts and to allow an interrupt of higher priority to cause a lower priority interrupts pause until high priority interrupt completes its function.

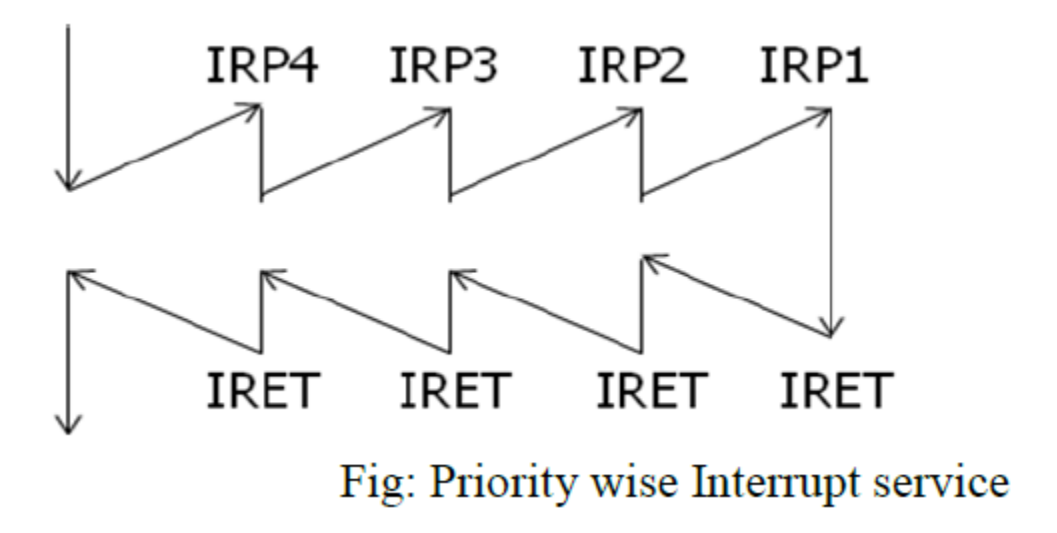


Fig: Priority wise Interrupt service

# INTERRUPT SERVICE ROUTINE

- An interrupt service routine (ISR) is a software routine that hardware invokes in response to an interrupt.
- ISRs examine an interrupt and determine how to handle it.
- ISRs handle the interrupt, and then return a logical interrupt value.
- Its central purpose is to process the interrupt and then return control to the main program.
- An ISR must perform very fast to avoid slowing down the operation of the device and the operation of all lower priority ISRs.
- As in procedures, the last instruction in an ISR should be ret.

# INTERRUPT SERVICE ROUTINE

**ISR is responsible for doing the following things:**

- **Saving the processor context**
  - Because the ISR and main program use the same processor registers, it is the responsibility of the ISR to save the processor's registers before beginning any processing of the interrupt. The processor context consists of the instruction pointer, registers, and any flags. Some processors perform this step automatically.
- **Acknowledging the interrupt**
  - The ISR must clear the existing interrupt, which is done either in the peripheral that generated the interrupt, in the interrupt controller, or both.
- **Restoring the processor context**
  - After interrupt processing, in order to resume the main program, the values that were saved prior to the ISR execution must be restored. Some processors perform this step automatically.

# INTERRUPT PROCESSING IN 8085

- Interrupt is signals send by an external device to the processor, to request the processor to perform a particular task or work.
- Mainly in the microprocessor based system the interrupts are used for data transfer between the peripheral and the microprocessor.
- The processor will check the interrupts always at the 2nd T-state of last machine cycle.
- If there is any interrupt it accept the interrupt saves the value of PSW and PC into the stack and send the INTA (active low) signal to the peripheral.
- The vectored address of particular interrupt is stored in program counter.
- The processor executes an interrupt service routine (ISR) addressed in program counter
- It returned to main program by RET instruction.

# TYPES OF INTERRUPT

- It supports two types of interrupts.
  - Hardware
  - Software
- **Software interrupts:**
  - The software interrupts are program instructions. These instructions are inserted at desired locations in a program.
  - The 8085 has eight software interrupts from RST 0 to RST 7. The vector address for these interrupts can be calculated as follows.
  - Interrupt number * 8 = vector address
  - For RST 5; 5 * 8 = 40 = 28H
  - Vector address for interrupt RST 5 is 0028H
  - The Table shows the vector addresses of all interrupts

| Interrupt | Vector address |
|---|---|
| RST 0<br>RST 1 | $0000_H$<br>$0008_H$ |
| RST 2<br>RST 3 | $0010_H$<br>$0018_H$ |
| RST 4<br>RST 5 | $0020_H$<br>$0028_H$ |
| RST 6<br>RST 7 | $0030_H$<br>$0038_H$ |

# TYPES OF INTERRUPT

- **Hardware interrupts (Interrupt Pins and Priorities)**
  - An external device initiates the hardware interrupts and placing an appropriate signal at the interrupt pin of the processor.
  - If the interrupt is accepted then the processor executes an interrupt service routine.
  - The 8085 has five hardware interrupts
  - (1) TRAP (2) RST 7.5 (3) RST 6.5 (4) RST 5.5 (5) INTR

| Interrupt type | Trigger | Priority | Maskable | Vector address |
|---|---|---|---|---|
| TRAP | Edge and Level | 1st | No | 0024H |
| RST 7.5 | Edge | 2nd | Yes | 003CH |
| RST 6.5 | Level | 3rd | Yes | 0034H |
| RST 5.5 | Level | 4th | Yes | 002CH |
| INTR | Level | 5th | Yes | - |

# TYPES OF INTERRUPT

- **TRAP:**
  - This interrupt is a non-maskable interrupt. It is unaffected by any mask or interrupt enable.
  - TRAP bas the highest priority and vectored interrupt.
  - TRAP interrupt is edge and level triggered. This means hat the TRAP must go high and remain high until it is acknowledged.
  - In sudden power failure, it executes a ISR and send the data from main memory to backup memory.
  - The signal, which overrides the TRAP, is HOLD signal. (i.e., If the processor receives HOLD and TRAP at the same time then HOLD is recognized first and then TRAP is recognized).
  - There are two ways to clear TRAP interrupt.
    - By resetting microprocessor (External signal)
    - By giving a high TRAP ACKNOWLEDGE (Internal signal)

# TYPES OF INTERRUPT

- **RST 7.5:**
  - The RST 7.5 interrupt is a maskable interrupt.
  - It has the second highest priority.
  - It is edge sensitive. i.e. Input goes to high and no need to maintain high state until it recognized.
  - Maskable interrupt. It is disabled by,
    - 1. DI instruction
    - 2. System or processor reset.
    - 3. After reorganization of interrupt.
  - Enabled by EI instruction.

# TYPES OF INTERRUPT

- **RST 6.5 and 5.5:**
  - The RST 6.5 and RST 5.5 both are level triggered. . ie. Input goes to high and stay high until it recognized.
  - Maskable interrupt. It is disabled by,
    - DI, SIM instruction
    - System or processor reset.
    - After reorganization of interrupt.
  - Enabled by EI instruction.
  - The RST 6.5 has the third priority whereas RST 5.5 has the fourth priority.

# TYPES OF INTERRUPT

- **INTR:**
  - INTR is a maskable interrupt.
  - It is disabled by,
    - 1. DI, SIM instruction
    - 2. System or processor reset.
    - 3. After reorganization of interrupt.
  - Enabled by EI instruction.
  - Non- vectored interrupt. After receiving INTA (active low) signal, it has to supply the address of ISR.
  - It has lowest priority.
  - It is a level sensitive interrupts. ie. Input goes to high and it is necessary to maintain high state until it recognized.

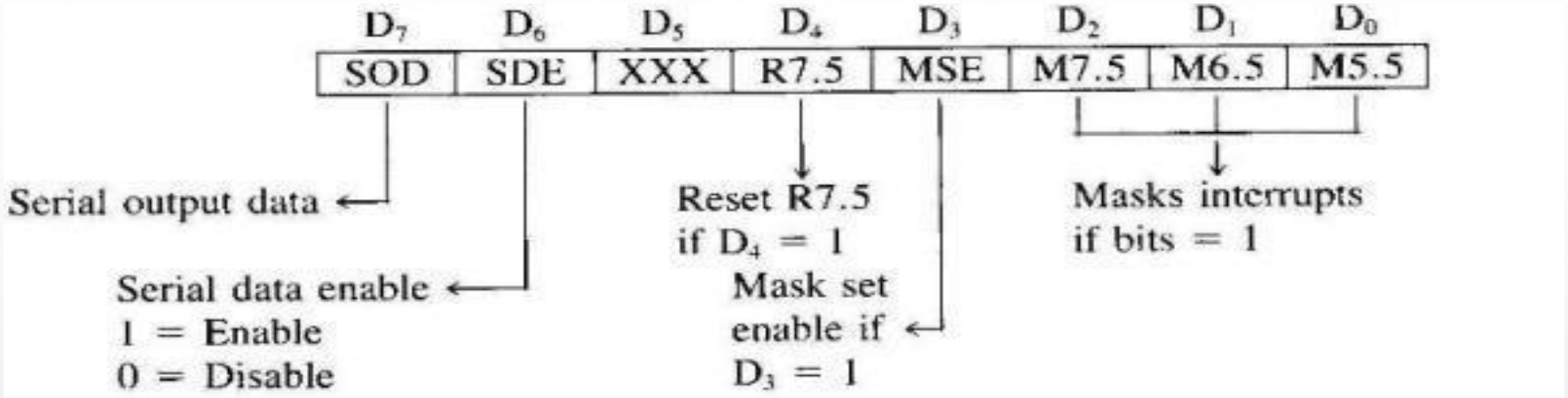
# TYPES OF INTERRUPT

- **The following sequence of events occurs when INTR signal goes high.**
  - **1**. The 8085 checks the status of INTR signal during execution of each instruction.
  - 2. If INTR signal is high, then 8085 complete its current instruction and sends active low interrupt acknowledge signal, if the interrupt is enabled.
  - 3. In response to the acknowledge signal, external logic places an instruction OPCODE on the data bus. In the case of multibyte instruction, additional interrupt acknowledge machine cycles are generated by the 8085 to transfer the additional bytes into the microprocessor.
  - 4. On receiving the instruction, the 8085 save the address of next instruction on stack and execute received instruction**.**

# INTERRUPT INSTRUCTIONS

- **SIM (Set Interrupt Mask) instruction:**
  - The 8085 provide additional masking facility for RST 7.5, RST 6.5 and RST 5.5 using SIM instruction.
  - This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output.
  - The masking or unmasking of RST 7.5, RST 6.5 and RST 5.5 interrupts can be performed by moving an 8-bit data to accumulator and then executing SIM instruction.
  - The format of the 8-bit data is shown below.

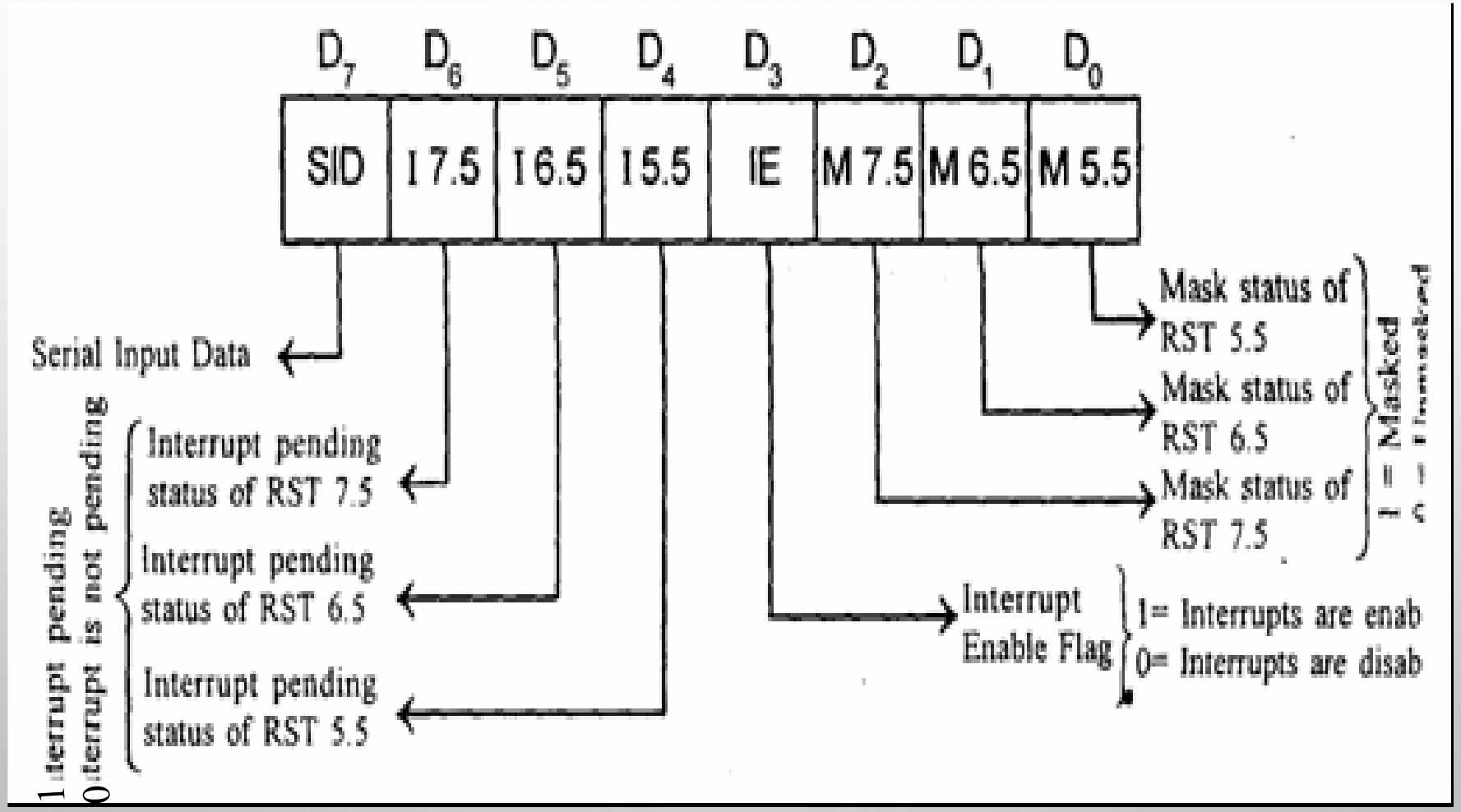
# INTERRUPT INSTRUCTIONS




- SOD — Serial Output Data: Bit $D_7$ of the accumulator is latched into the SOD output line and made available to a serial peripheral if bit $D_6 = 1$.
- SDE — Serial Data Enable: If this bit = 1, it enables the serial output. To implement serial output, this bit needs to be enabled.
- XXX — Don't Care
- R7.5 — Reset RST 7.5: If this bit = 1, RST 7.5 flip-flop is reset. This is an additional control to reset RST 7.5.
- MSE — Mask Set Enable: If this bit is high, it enables the functions of bits $D_2$, $D_1$, $D_0$. This is a master control over all the interrupt masking bits. If this bit is low, bits $D_2$, $D_1$, and $D_0$ do not have any effect on the masks.
- M7.5 — $D_2$ = 0, RST 7.5 is enabled.
  = 1, RST 7.5 is masked or disabled.
- M6.5 — $D_1$ = 0, RST 6.5 is enabled.
  = 1, RST 6.5 is masked or disabled.
- M5.5 — $D_0$ = 0, RST 5.5 is enabled.
  = 1, RST 5.5 is masked or disabled.

# INTERRUPT INSTRUCTIONS

- **RIM (Read Interrupt Mask) instruction:**



# INTERRUPT INSTRUCTIONS

- The status of pending interrupts can be read from accumulator after executing RIM instruction.
- This is a multipurpose instruction used to read the status of RST 7.5, 6.5, 5.5 and read serial data input bit.
- When RIM instruction is executed an 8-bit data is loaded in accumulator, which can be interpreted as shown in above fig.
- **Bits 0-2** show the current setting of the mask for each of RST 7.5, RST 6.5 and RST 5.5. They return the contents of the three masks flip flops. They can be used by a program to read the mask settings in order to modify only the right mask.
- **Bit 3** shows whether the maskable interrupt process is enabled or not. It returns the contents of the Interrupt Enable Flip Flop. It can be used by a program to determine whether or not interrupts are enabled.
- **Bits 4-6** show whether or not there are pending interrupts on RST 7.5, RST 6.5, and RST 5.5. Bits 4 and 5 return the current value of the RST5.5 and RST6.5 pins. Bit 6 returns the current value of the RST7.5 memory flip flop.
- **Bit 7** is used for Serial Data Input. The RIM instruction reads the value of the SID pin on the microprocessor and returns it in this bit.

# INTERRUPT INSTRUCTIONS

- **DI**
  - Disable interrupts
  - The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected.
  - 1 byte instruction
  - Example: DI
- **EI**
  - Enable interrupts
  - The interrupt enable flip-flop is set and all interrupts are enabled.
  - No flags are affected.
  - After a system reset or the acknowledgement of an interrupt, the interrupt enable flip flop is reset, thus disabling the interrupts.
  - This instruction is necessary to enable the interrupts (except TRAP).
  - 1 byte instruction
  - Example: EI

# USING PROGRAMMABLE INTERRUPT CONTROLLER (PIC)

- **Priority interrupt controller (PIC)**
- The INTR pin can be used for multiple peripherals and to determine priorities among these devices when two or more peripherals request interrupt service simultaneously, PIC is used.
- If there are simultaneous requests, the priorities are determined by the encoder, it responds to the higher level input, ignoring the lower level input.
- The drawback of the scheme is that the interrupting device connected to input I7 always has the highest priority.
- The PIC includes a status register and a priority comparator in addition to a priority encoder.

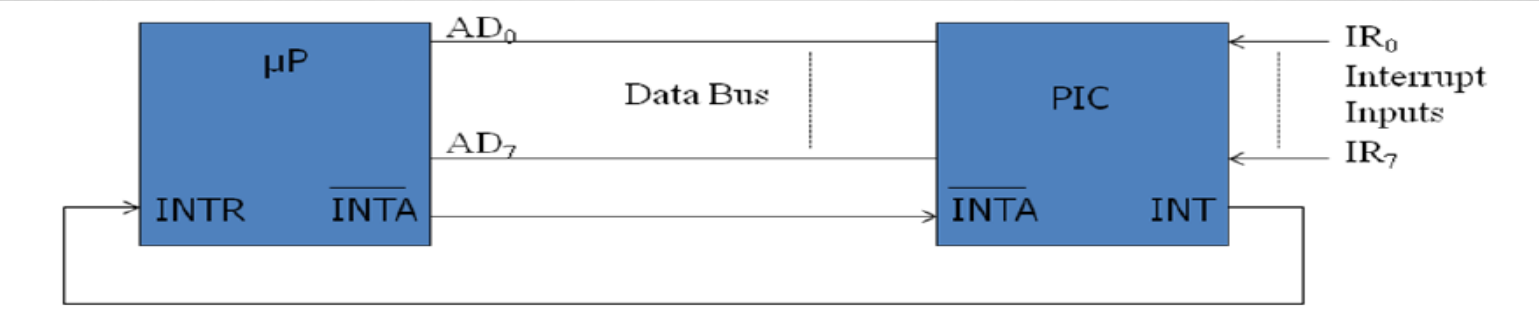
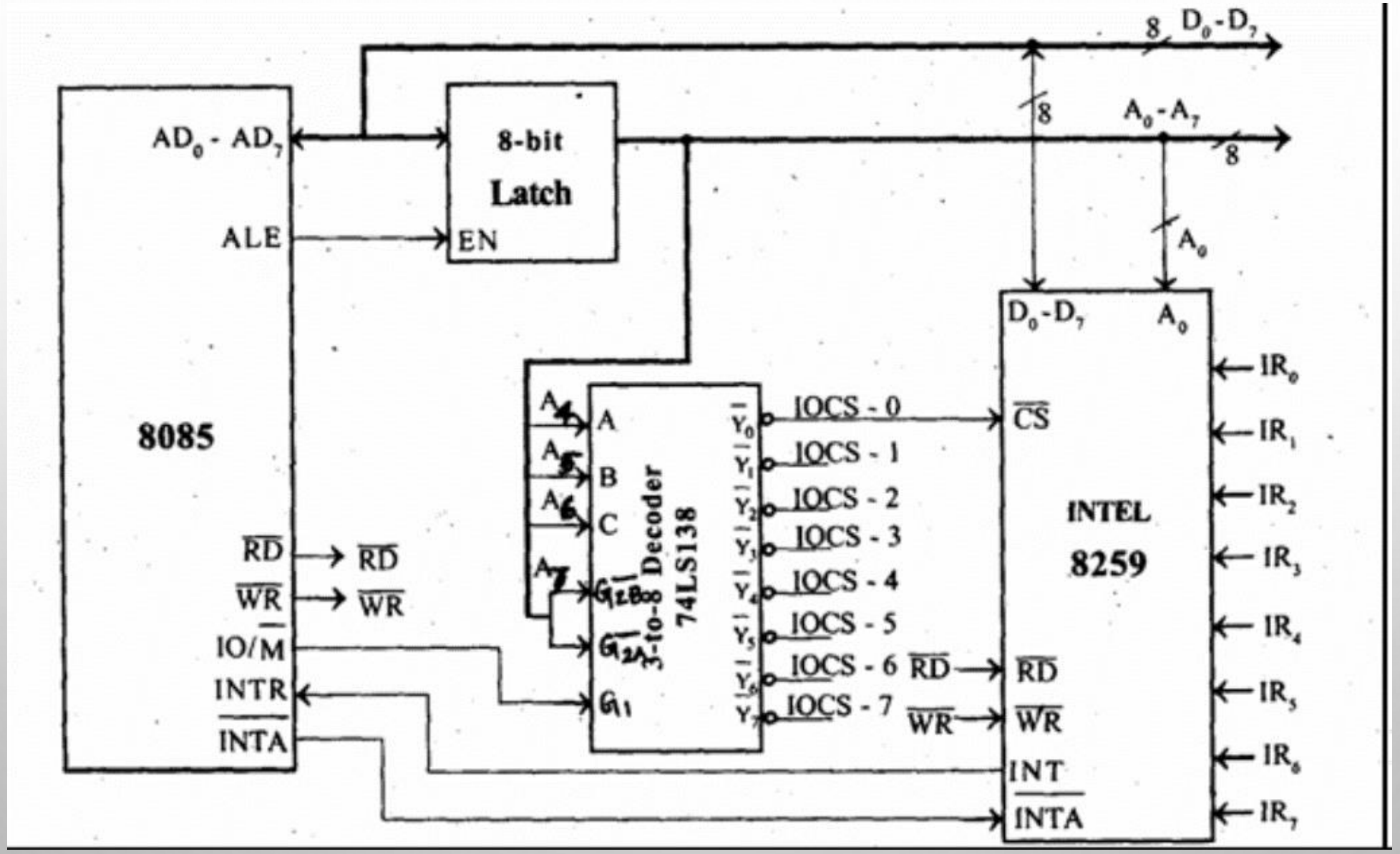


Fig: Multiple Interrupts using PIC

# USING PROGRAMMABLE INTERRUPT CONTROLLER (PIC)

- Today this device is replaced by a more versatile one called a programmable interrupt controller 8259A.
- When an 8259A receives an interrupt signal on one of its IR inputs, it sends an interrupt request signal to the INTR input of the μP.
- Then INTA pulses will cause the PIC to release vectoring information onto the data bus.

# USING PROGRAMMABLE INTERRUPT CONTROLLER (PIC)



# USING PROGRAMMABLE INTERRUPT CONTROLLER (PIC)

- It requires two internal address and they are A =0 or A = 1.
- It can be either memory mapped or I/O mapped in the system. The interfacing of 8259 to 8085 is shown in figure is I/O mapped in the system.
- The low order data bus lines D0-D7 are connected to D0-D7 of 8259.
- The address line A0 of the 8085 processor is connected to A0 of 8259 to provide the internal address.
- The 8259 require one chip select signal. Using 3-to-8 decoder generates the chip select signal for 8259.
- The address lines A4, A5 and A6 are used as input to decoder.
- The control signal IO/M (low) is used as logic high enables for decoder and the address line A7 is used as logic low enable for decoder.
- The I/O addresses of 8259 are shown in table below.

# USING PROGRAMMABLE INTERRUPT CONTROLLER (PIC)

| | Binary Address | | | | | | | | Hexa address |
|---|---|---|---|---|---|---|---|---|---|
| | Decoder input/ enable | | | Input to address pin of 8259 | | | | | |
| | $A_7$ | $A_6$ | $A_5$ | $A_4$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ | |
| For $A_0$ of 8259 to be zero | 0 | 0 | 0 | 0 | x | x | x | 0 | 00 |
| For $A_0$ of 8259 to be one | 0 | 0 | 0 | 0 | x | x | x | 1 | 01 |

**Note :** *Don't care "x" is considered as zero.*

# USING PROGRAMMABLE INTERRUPT CONTROLLER (PIC)

- **Working of 8259 with 8085 processor:**

- First the 8259 should be programmed by sending Initialization Command Word (ICW) and Operational Command Word (OCW). These command words will inform 8259 about the following,
  - 1. Type of interrupt signal (Level triggered / Edge triggered).
  - 2. Type of processor (8085/8086).
  - 3. Call address and its interval (4 or 8)
  - 4. Masking of interrupts.
  - 5. Priority of interrupts.
  - 6. Type of end of interrupts.
- Once 8259 is programmed it is ready for accepting interrupt signal. When it receives an interrupt through any one of the interrupt lines IR0-IR7 it checks for its priority and also checks whether it is masked or not.

# USING PROGRAMMABLE INTERRUPT CONTROLLER (PIC)

- If the previous interrupt is completed and if the current request has highest priority and unmasked, then it is serviced.
- For servicing this interrupt the 8259 will send INT signal to INTR pin of 8085.
- In response it expects an acknowledge INTA (low) from the processor.
- When the processor accepts the interrupt, it sends three INTA (low) one by one.
- In response to first, second and third INTA (low) signals, the 8259 will supply CALL opcode, low byte of call address and high byte of call address respectively. Once the processor receives the call opcode and its address, it saves the content of program counter (PC) in stack and load the CALL address in PC and start executing the interrupt service routine stored in this call address.

# USING PROGRAMMABLE INTERRUPT CONTROLLER (PIC)

- **How INTR pin is used in 8085:**
- The microprocessor checks INTR, one clock period before the last T- state of an instruction cycle.
- In the 8085, the call instructions require 18 T-states; therefore the INTR pulse should be high at least for 17.5 T-states.
- The INTR can remain high until the interrupt flip flop is set by the EI instruction in the service routine.
- If it remains high after the execution of the EI instruction, the processor will be interrupted again, as if it was a new interrupt.

# INTERRUPT PROCESSING IN 8086

- The meaning of ‘interrupts’ is to break the sequence of operation.
- While the CPU is executing a program, on ‘interrupt’ breaks the normal sequence of execution of instructions, diverts its execution to some other program called Interrupt Service Routine (ISR).
- After executing ISR , the control is transferred back again to the main program.
- Interrupt processing is an alternative to polling.

# INTERRUPT PROCESSING IN 8086

## Interrupt Pins

- **INTR and NMI**
  - INTR is a maskable hardware interrupt. The interrupt can be enabled/disabled using STI/CLI instructions or using more complicated method of updating the FLAGS register with the help of the POPF instruction.
  - When an interrupt occurs, the processor stores FLAGS register into stack, disables further interrupts, fetches from the bus one byte representing interrupt type, and jumps to interrupt processing routine address of which is stored in location 4 * <interrupt type>. Interrupt processing routine should return with the IRET instruction.
  - NMI is a non-maskable interrupt. Interrupt is processed in the same way as the INTR interrupt. Interrupt type of the NMI is 2, i.e. the address of the NMI processing routine is stored in location 0008h. This interrupt has higher priority than the maskable interrupt.
  - – Ex: NMI, INTR.

# INTERRUPT VECTOR TABLE AND ITS ORGANIZATION

- An interrupt vector is a pointer to where the ISR is stored in memory.
- All interrupts (vectored or otherwise) are mapped onto a memory area called the Interrupt Vector Table (IVT).
  - The IVT is usually located in memory page 00 (0000H - 00FFH).
  - The purpose of the IVT is to hold the vectors that redirect the microprocessor to the right place when an interrupt arrives.
- Interrupt Vector Table (IVT) is a 1024 bytes sized table that contains addresses of interrupts.
- Each address is of 4 bytes long of the form offset:segment, which represents the address of a routine to be called when the CPU receives an interrupt.
- IVT can hold maximum of 256 addresses (0 to 255).
- The interrupt number is used as an index into the table to get the address of the interrupt service routine.

# INTERRUPT VECTOR TABLE AND ITS ORGANIZATION

- IVT act as pointers, unlike function call IVT need number as an argument then as a result IVT point us to interrupt service routine (ISR).
- ISR executes its code, when ISR finished then returns back to original statement. Interrupt vector table is a global table situated at the address 0000:0000H.
- The interrupt vector table is a feature of the Intel 8086/8088 family of microprocessors.

# INTERRUPT VECTOR TABLE AND ITS ORGANIZATION

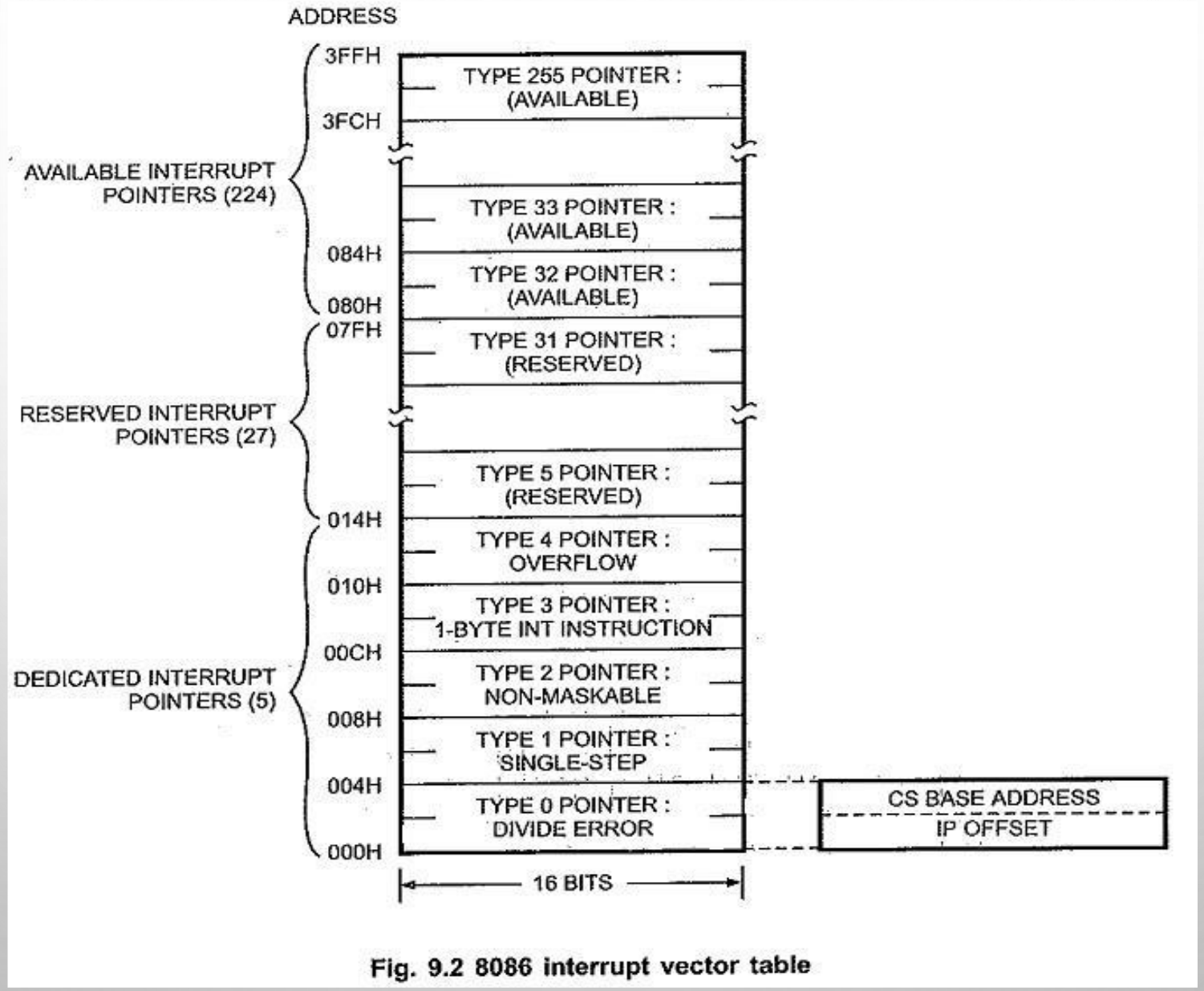



**Fig. 9.2 8086 interrupt vector table**

# INTERRUPT VECTOR TABLE AND ITS ORGANIZATION

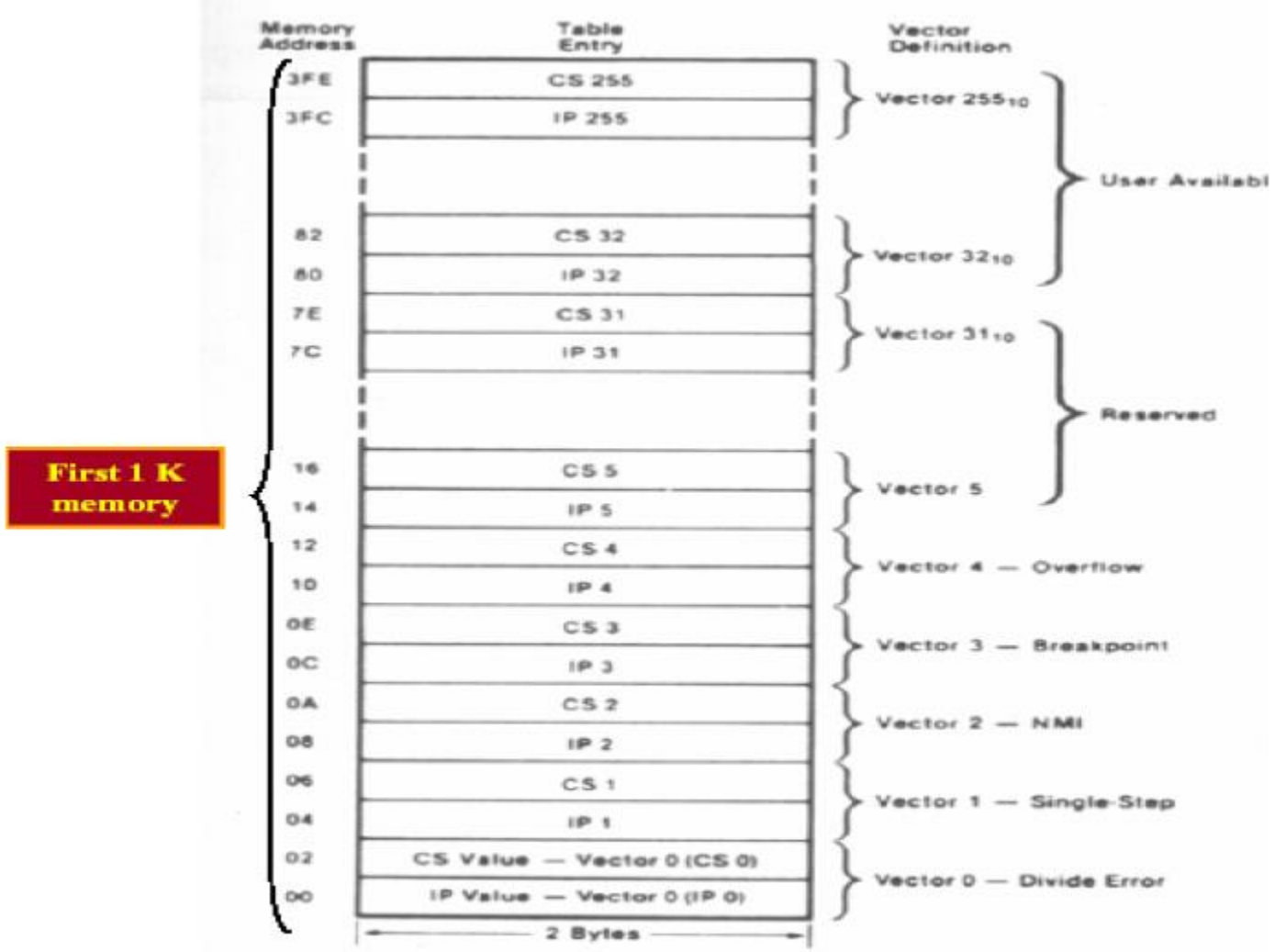



Fig: IVT Structure (Organization)

# INTERRUPT VECTOR TABLE AND ITS ORGANIZATION

| INT Number | Physical Address |
|---|---|
| INT 00 | 00000 |
| INT 01 | 00004 |
| INT 02 | 00008 |
| : | : |
| : | : |
| INT FF | 003FC |

# DEDICATED INTERRUPTS

**INT 00 (divide error)**

- INT00 is invoked by the microprocessor whenever there is an attempt to divide a number by zero.
- ISR is responsible for displaying the message “Divide Error” on the screen

**INT 01 (Single step interrupt)**

- For single stepping the trap flag must be 1
- After execution of each instruction, 8086 automatically jumps to 00004H to fetch 4 bytes for CS: IP of the ISR.
- The job of ISR is to dump the registers on to the screen

# DEDICATED INTERRUPTS

**INT 02 (Non maskable Interrupt)**

- When ever NMI pin of the 8086 is activated by a high signal (5v), the CPU Jumps to physical memory location 00008 to fetch CS:IP of the ISR associated with NMI.

**INT 03 (break point)**

- A break point is used to examine the cpu and memory after the execution of a group of Instructions.
- It is one byte instruction whereas other instructions of the form “INT nn” are 2 byte instructions.

**INT 04 ( Signed number overflow)**

- There is an instruction associated with this INT 0 (interrupt on overflow).
- If INT 0 is placed after a signed number arithmetic as IMUL or ADD the CPU will activate INT 04 if OF = 1. (OF = Overflow Flag)



- In case where OF = 0 , the INT 0 is not executed but is bypassed and acts as a NOP.

# SOFTWARE AND HARDWARE INTERRUPT

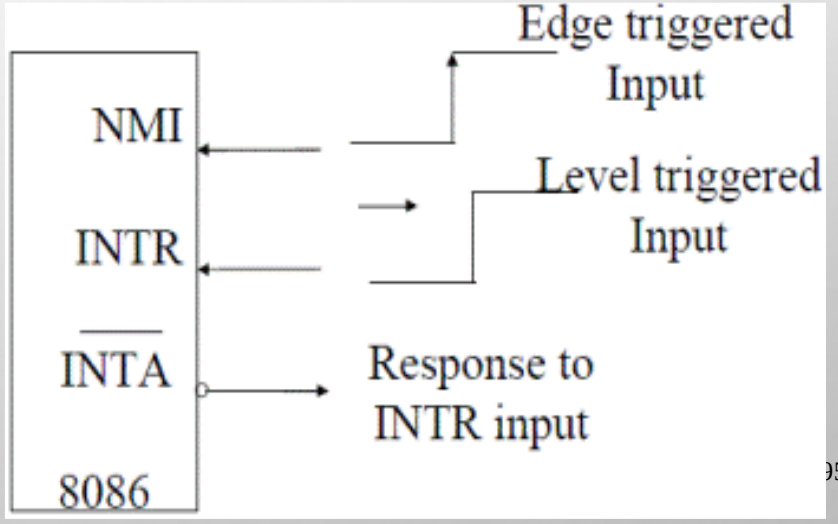
- **Types of Interrupts**:
  - There are two types of Interrupts in 8086. They are:

## 1) Hardware Interrupts (External Interrupts).

- The Intel microprocessors support hardware interrupts through:
  - Two pins that allow interrupt requests, INTR and NMI
  - One pin that acknowledges, INTA, the interrupt requested on INTR.

## Performance of Hardware Interrupts

- NMI : Non maskable interrupts - TYPE 2 Interrupt
- INTR : Interrupt request - Between 20H and FFH



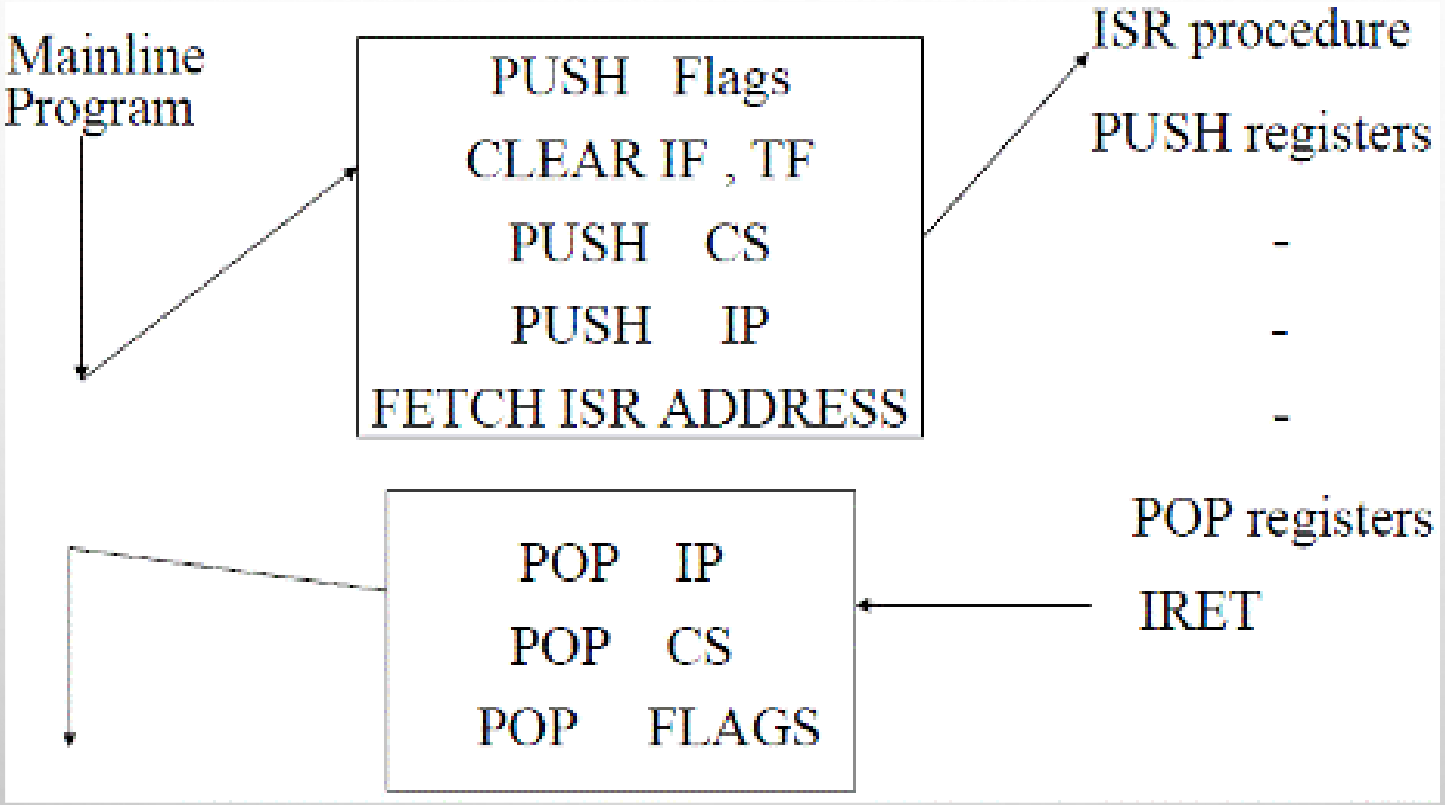
# SOFTWARE AND HARDWARE INTERRUPT

## 2) Software Interrupts (Internal Interrupts and Instructions) .

- Software interrupts can be caused by:
- INT instruction - breakpoint interrupt. This is a type 3 interrupt.
- INT <interrupt number> instruction - any one interrupt from available 256 interrupts.
- INT0 instruction - interrupt on overflow
- Single-step interrupt - generated if the TF flag is set. This is a type 1 interrupt. When the CPU processes this interrupt it clears TF flag before calling the interrupt processing routine.
- Processor exceptions: Divide Error (Type 0), Unused Opcode (type 6) and Escape opcode (type 7).
- Software interrupt processing is the same as for the hardware interrupts.
- Ex: INT n (Software Instructions)
- Control is provided through:
  - IF and TF flag bits
  - IRET and IRETD

# SOFTWARE AND HARDWARE INTERRUPT

**Performance of Software Interrupts**



# SOFTWARE AND HARDWARE INTERRUPT

## Performance of Software Interrupts

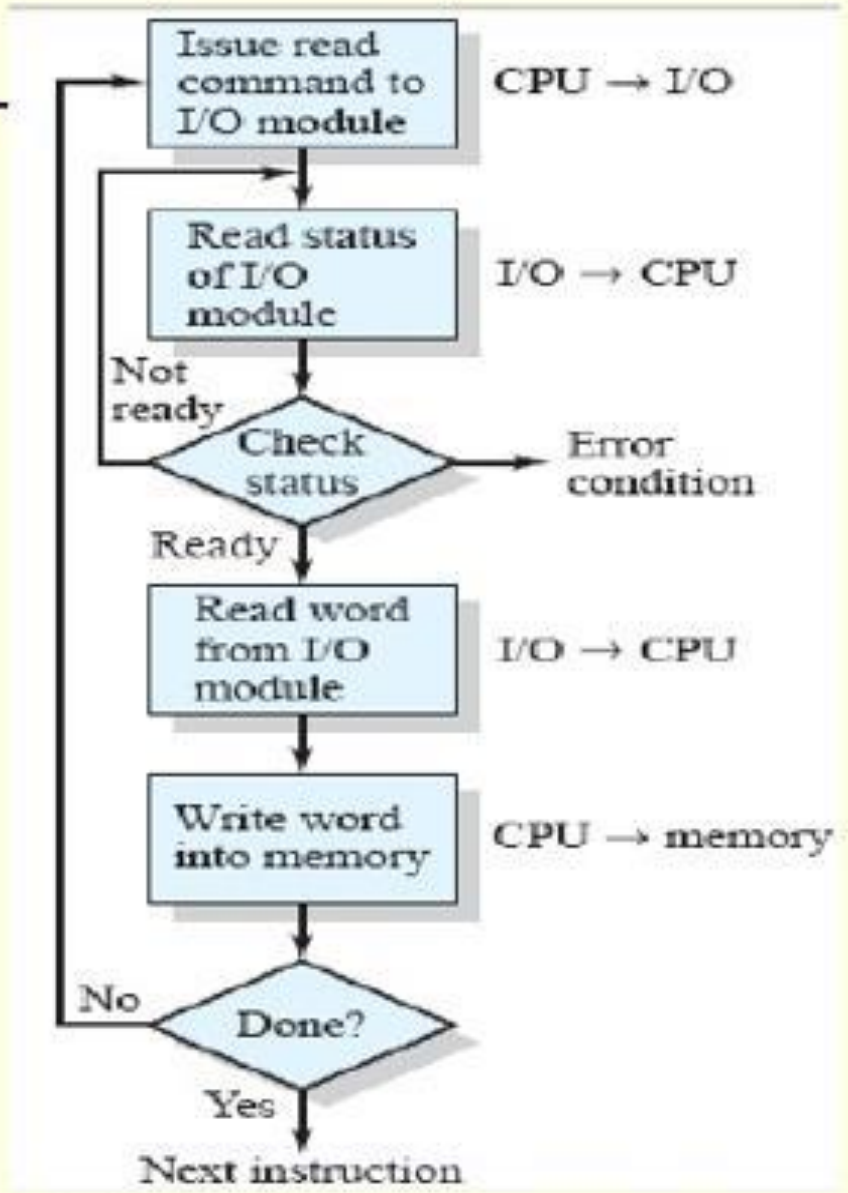
- It decrements SP by 2 and pushes the flag register on the stack.
- Disables INTR by clearing the IF.
- It resets the TF in the flag Register.
- It decrements SP by 2 and pushes CS on the stack.
- It decrements SP by 2 and pushes IP on the stack.
- Fetch the ISR address from the interrupt vector table.

**Interrupt Priorities**

| Interrupt | Priority |
|---|---|
| Divide Error, INT(n),INTO | Highest |
| NMI | ↓ |
| INTR | ↓ |
| Single Step | Lowest |

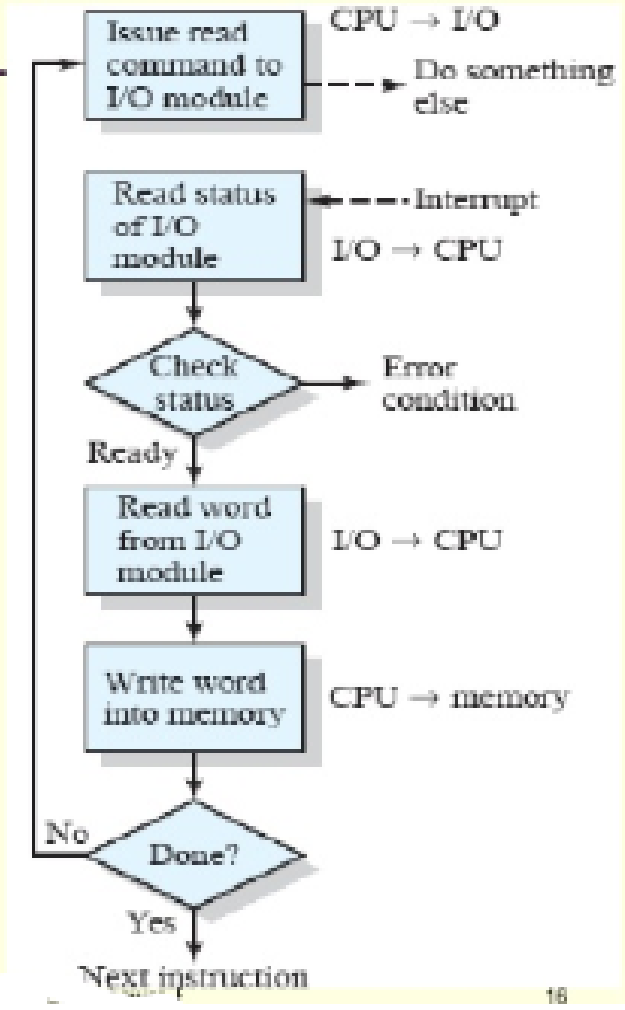
# Programmed I/O

- CPU while executing a program encounters an I/O instruction
- CPU issues I/O command to I/O module
- I/O module performs the requested action & set status registers
- CPU is responsible to check status registers periodically to see if I/O operation is complete. **SO**
- No Interrupt to alert the processor



# Interrupt-Driven I/O

- Similar to direct I/O but processor not required to poll device.
- I/O module will interrupt CPU for data exchange when ready



# Interrupt operations

1. While CPU is executing a program, an interrupt exists then it
a) follows the next instruction in the program
b) jumps to instruction in other registers
c) breaks the normal sequence of execution of instructions
d) stops executing the program

2. An interrupt breaks the execution of instructions and diverts its execution to
a) Interrupt service routine
b) Counter word register
c) Execution unit
d) control unit

3. While executing the main program, if two or more interrupts occur, then the sequence of appearance of interrupts is called
a) multi-interrupt
b) nested interrupt
c) interrupt within interrupt
d) nested interrupt and interrupt within interrupt

4. Whenever a number of devices interrupt a CPU at a time, and if the processor is able to handle them properly, it is said to have
a) interrupt handling ability
b) interrupt processing ability
c) multiple interrupt processing ability
d) multiple interrupt executing ability



5. NMI stands for
a) nonmaskable interrupt
b) nonmultiple interrupt
c) nonmovable interrupt
d) none of the mentioned

7. If any interrupt request given to an input pin cannot be disabled by any means then the input pin is called
a) maskable interrupt
b) nonmaskable interrupt
c) maskable interrupt and nonmaskable interrupt
d) none of the mentioned

# Interrupt operations

8. The INTR interrupt may be
a) maskable
b) nonmaskable
c) maskable and nonmaskable
d) none of the mentioned

9. The Programmable interrupt controller is required to
a) handle one interrupt request
b) handle one or more interrupt requests at a time
c) handle one or more interrupt requests with a delay
d) handle no interrupt request

10. The INTR interrupt may be masked using the flag
a) direction flag
b) overflow flag
c) interrupt flag
d) sign flag

1. If an interrupt is generated from outside the processor then it is an
a) internal interrupt
b) external interrupt
c) interrupt
d) none of the mentioned

2. If the interrupt is generated by the execution of an interrupt instruction then it is
a) internal interrupt
b) external interrupt
c) interrupt-in-interrupt
d) none of the mentioned

3. Example of an external interrupt is
a) divide by zero interrupt
b) keyboard interrupt
c) overflow interrupt
d) type2 interrupt

# Interrupt operations

4. Example of an internal interrupt is
a) divide by zero interrupt
b) overflow interrupt
c) interrupt due to INT
d) all of the mentioned

5. The interrupt request that is independent of IF flag is
a) NMI
b) TRAP
c) Divide by zero
d) All of the mentioned

7. During the execution of an interrupt, the data pushed into the stack is the content of
a) IP
b) CS
c) PSW
d) All of the mentioned

9. At the end of ISR, the instruction should be
a) END
b) ENDS
c) IRET
d) INTR

2. The register that stores all the interrupt requests in it in order to serve them one by one on a priority basis is
a) Interrupt Request Register
b) In-Service Register
c) Priority resolver
d) Interrupt Mask Register

3. The register that stores the bits required to mask the interrupt inputs is
a) In-service register
b) Priority resolver
c) Interrupt Mask register
d) None

# Interrupt operations

4. The interrupt control logic
a) manages interrupts
b) manages interrupt acknowledge signals
c) accepts interrupt acknowledge signal
d) all of the mentioned

5. In a cascaded mode, the number of vectored interrupts provided by 8259A is
a) 4
b) 8
c) 16
d) 64

9. In the application where all the interrupting devices are of equal priority, the mode used is
a) Automatic rotation
b) Automatic EOI mode
c) Specific rotation
d) EOI

8. When non-specific EOI command is issued to 8259A it will automatically
a) set the ISR
b) reset the ISR
c) set the INTR
d) reset the INTR