GANESH INSTITUTE OF ENGINEERING AND TECHNOLOGY (GIET),
Jagannath Prasad, Andharua, BHUBANESWAR

DIGITAL ELECTRONICS & MICROPROCESSOR

(Th- 03)

(As per the 2019-20 syllabus of the SCTE&VT, Bhubaneswar, Odisha)

Fourth Semester

Electrical Engg.

Prepared By: Er. SWAGATIKA DAS

DIGITALELECTRONICS & MICROPROCESSOR

CHAPTER-WISE DISTRIBUTION OF PERIODS & MARKS

Sl. No.	Chapter No.	Topics	Periods as per syllabus	Periods actually needed	Expected marks
1	1	**BASICS OF DIGITAL ELECTRONICS**	**12**	**12**	**25**
2	2	**COMBINATIONAL LOGIC CIRCUITS**	**10**	**10**	**20**
3	3	**SEQUENTIAL LOGIC CIRCUITS**	**09**	**09**	**15**
4	4	**8085 MICROPROCESSORS**	**15**	**15**	**25**
5	5	**INTERFACING AND SUPPORT CHIPS**	**04**	**04**	**05**
Total			**60**	**60**	**110**

Chapter-01

INTRODUCTION

LEARING OBJECTIVES:

1.1-Binary, Octal, Hexadecimal number systems and compare with Decimal system.
1.2-Binary addition, subtraction, multiplication and division.
1.3-1's compliment and 2's complement numbers for a binary number.
1.4-Subtraction of binary numbers in 2's complement method.
1.5-Use of weighted and Un-weighted codes & binary equivalent number for a number in 8421, Excess-3 code and gray code and vice-versa.
1.6-Importance of parity bit.
1.7-Logic gates: AND, OR, NOT, NOR, NAND, EX-OR Gate with truth table.
1.8-Realize AND, OR, NOT OPERATION using NAND, NOR gates.
1.9-Different postulates and De-Morgan's theorems in Boolean algebra.
1.10-Use of Boolean Algebra for simplification of logic expression.
1.11-Karnaugh map for 2,3,4 variable, Simplification of SOP and POS logic expression.

INTRODUCTION: -

- The term digital refers to a process that is achieved by using discrete unit.
- In number system there are different symbols and each symbol has an absolute value and also has place value.

RADIX OR BASE: -

- The radix or base of a number system is defined as the number of different digits which can occur in each position in the number system.

1.1-Binary, Octal, Hexadecimal number system and compare with decimal system

NUMBER SYSTEM: -

TYPES OF NUMBER SYSTEM: -

There are four types of number systems. They are

1. Decimal number system
2. Binary number system
3. Octal number system
4. Hexadecimal number system

DECIMAL NUMBER SYSTEM: -

- The decimal number system contains ten unique symbols 0,1,2,3,4,5,6,7,8 and 9. In decimal system 10 symbols are involved, so the base or radix is 10.

- It is a positional weighted system.
- The value attached to the symbol depends on its location with respect to the decimal point. In general,

dn dn-1 dn-2 …… d0 . d -1 d -2 ..... d - m is given by

$(dn \times 10^{n}) + (dn\text{-}1 \times 10^{n-1}) + (dn\text{-}2 \times 10^{n-2}) + \ldots + (d0 \times 10^{0}) + (d\text{-}1 \times 10^{-1}) + (d\text{-}2 \times 10^{-2}) + \ldots + (d\,m \times 10^{-m})$

For example: $9256.26 = 9 \times 1000 + 2 \times 100 + 5 \times 10 + 6 \times 1 + 2 \times (1/10) + 6 \times (1/100)$

$= 9 \times 10^{3} + 2 \times 10^{2} + 5 \times 10^{1} + 6 \times 10^{0} + 2 \times 10^{-1} + 6 \times 10^{-2}$

BINARYNUMBER SYSTEM: -

- The binary number system is a positional weighted system. The base or radix of this number system is 2.
- It has two independent symbols. The symbols used are 0 and 1. A binary digit is called as bit.

OCTAL NUMBER SYSTEM: -

- It is also a positional weighted system. Its base or radix is 8.
- It has 8 independent symbols 0,1,2,3,4,5,6 and 7.
- Its base $8 = 2^3$, every 3- bit group of binary can be represented by an octal digit.

HEXADECIMALNUMBER SYSTEM: -

- The hexadecimal number system is a positional weighted system. The base or radix of this number system is 16.
- The symbols used are 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E and F
- The base 16 = 24 , every 4 – bit group of binary can be represented by an hexadecimal digit.

CONVERSION FROM ONE NUMBER SYSTEM TO ANOTHER

1-BINARY NUMBER SYSTEM: -

Binary to decimal conversion: -

In this method, each binary digit of the number is multiplied by its positional weight and the product terms are added to obtain decimal number. **For example:**

- Convert (10101)2 to decimal.

Solution:

(Positional weight) $2^4\ 2^3\ 2^2\ 2^1\ 2^0$ Binary number 10101

$= (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$

$= 16 + 0 + 4 + 0 + 1$

$= (21)10$

- Convert (111.101)2 to decimal. Solution:

$(111.101)2 = (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3})$

$= 4 + 2 + 1 + 0.5 + 0 + 0.125$

$= (7.625)10$

Binary to Octal conversion: -

- For conversion binary to octal the binary numbers are divided into groups of 3 bits each, starting at the binary point and proceeding towards left and right.

Octal	Binary	Octal	Binary
0	000	4	100
1	001	5	101
2	010	6	110
3	011	7	111

For example:

(i) Convert (101111010110.110110011)2 into octal. Solution---

Group of 3 bits are 101 111 010 110 . 110 110 011
Convert each group into octal = 5 7 2 6 . 6 6 3
The result is (5726.663)8

(i) Convert (101111010110.110110011)2 into octal. Solution---

Group of 3 bits are 101 111 010 110 . 110 110 011
Convert each group into octal = 5 7 2 6 . 6 6 3
The result is (5726.663)8

Binary to Hexadecimal conversion: -

For conversion binary to hexadecimal number the binary numbers starting from the binary point, groups are made of 4 bits each, on either side of the binary point.

Hexadecimal	Binary 0000	Hexadecimal	Binary
0		8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

For example--
Convert (1011011011)2 into hexadecimal.

Solution

Given Binary number 10 1101 1011

Group of 4 bits are 0010 1101 1011

Convert each group into hex = 2 D B

The result is (2DB)16

Convert (01011111011.011111)2 into hexadecimal. Solution:

Given Binary number 010 1111 1011 . 0111 11

Group of 3 bits are = 0010 1111 1011 . 0111 1100

Convert each group into octal = 2 F B . 7 C

The result is (2FB.7C)16

2-DECIMAL NUMBER SYSTEM: -

Decimal to binary conversion: -

In the conversion the integer number are converted to the desired base using successive division by the base or radix.

For example:

Convert (52)10 into binary Solution-

Divide the given decimal number successively by 2 read the integer part remainder upwards to get equivalent binary number. Multiply the fraction part by 2. Keep the integer in the product as it is and multiply the new fraction in the product by 2. The process is continued and the integer are read in the products from top to bottom.

2 I52
2 l26 — 0
2 l13 — 0
2 l6 — 1
2 l3 — 0
2 l1 — 1
0 — 1

Result of (52)10 is (110100)2

Convert (105.15)10 into binary.

Integer part	Fraction part
2 I105	0.15 x 2 = 0.30
2 l 52 — 1	0.30 x 2 = 0.60
2 l 26 — 0	0.60 x 2 = 1.20
2 l13 — 0	0.20 x 2 = 0.40
2 l6 — 1	0.40 x 2 = 0.80
2 l3 — 0	0.80 x 2 = 1.60

2 11 — 1
0 — 1

Result of (105.15)10 is (1101001.001001)2

Decimal to octal conversion: -

To convert the given decimal integer number to octal, successively divide the given number

by 8 till the quotient is 0. To convert the given decimal fractions to octal successively multiply the decimal fraction and the subsequent decimal fractions by 8 till the product is 0 or till the required accuracy is obtained.

For example:

(i) Convert (378.93)10 into octal.

Solution:

8 I378		0.93 x 8 = 7.44 8
l 47	— 2	0.44 x 8 = 3.52
8 l 5	— 7	0.52 x 8 = 4.16
0	— 5	0.16 x 8 = 1.28

Result of (378.93)10 is (572.7341)8

Decimal to hexadecimal conversion: -

The decimal to hexadecimal conversion is same as octal.

For example:

(i) Convert (2598.675)10 into hexadecimal

Remainder	Decimal	Hex		Hex
16 I2598			0.675 x 16 = 10.8	A
16 l 162	— 6	6	0.800 x 16 = 12.8	C
16 l 10	— 2	2	0.800 x 16 = 12.8	C
0	— 10	A	0.800 x 16 = 12.8	C

Result of (2598.675)10 is (A26.ACCC) 16

3-OCTAL NUMBERS SYTEM: -

Octal to binary conversion: -

To convert a given a octal number to binary, replace each octal digit by its 3- bit binary equivalent.

For example:

Convert (367.52)8 into binary.

Solution:

Given Octal number is 3 6 7 . 5 2

Convert each group octal to = 011 110 111 . 101 010 binary

Result of (367.52)8 is (011110111.101010)2

Octal to decimal conversion: -

For conversion octal to decimal number, multiply each digit in the octal number by the weight of its position and add all the product terms .

For example: -

Convert (4057.06) 8 to decimal Solution:

$(4057.06)_8 = 4 \times 8^3 + 0 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 + 0\ 8^{-1} + 6 \times 8^{-2} \times$

$= 2048 + 0 + 40 + 7 + 0 + 0.0937 =$

(2095. 0937)10

For example: -

Convert (B9F.AE)16 to octal.

Solution:-

Given hexadecimal no.is	B	F	.	A	E	
Convert each hex. digit to binary	= 1011	1111	.	1010	1110	
Group of 3 bits are	= 101	011 111	.	101	011	100
Convert 3 bits group to octal.	= 5	3 7	.	5	4	

Result is (5637.534)8

Octal to hexadecimal conversion: -

 For conversion of octal to Hexadecimal, first convert the given octal number to binary and then binary number to hexadecimal.

For example: -

Convert (756.603)8 to hexadecimal.

Solution:-

Given octal no.		7	5	6	.	6	0	3
Convert each octal digit to binary	=	111	101	110	.	110	000	011
Group of 4bits are	=	0001	1110	1110	.	1100	0001	1000
Convert 4 bits group to hex.	=	1	E	E	.	C	1	8

Result is (1EE.C18)16

4 HEXADECIMAL NUMBER SYSTEM: -

Hexadecimal to binary conversion: -

 For conversion of hexadecimal to binary, replace hexadecimal digit by its 4 bit binary group.

For example:

Convert $(3A9E.B0D)_{16}$ into binary.

Solution:

Given Hexadecimal number is 3 A 9 E .B 0 D

Convert each hexadecimal= 0011 1010 1001 1110. 1011 0000 1101 digit to 4 bit binary

Result of (3A9E.B0D)8 is (0011101010011110.101100001101)2

Hexadecimal to decimal conversion: -

 For conversion of hexadecimal to decimal, multiply each digit in the hexadecimal number by its position weight and add all those product terms.

For example: -

Convert (A0F9.0EB)16 to decimal Solution:

$(A0F9.0EB)16 = (10 \times 16^3) + (0 \times 16^2) + (15 \times 16^1) + (9 \times 16^0) + (0 \times 16^{-1}) + (14 \times 16^{-2}) + (11 \times 16^{-3})$

$= 40960 + 0 + 240 + 9 + 0 + 0.0546 + 0.0026$

$= (41209.0572)10$

Result is (41209.0572)10

Hexadecimal to Octal conversion: -

 For conversion of hexadecimal to octal, first convert the given hexadecimal number to binary and then binary number to octal.

1.2 Binary addition, subtraction, multiplication and division

Arithmetic operation: -

BINARY ADDITION: -

The binary addition rules are as follows

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 10, i.e. 0 with a carry of 1

For example: -

Add (100101)2 and (1101111)2.

Solution: -

```
    1 0 0 1 0 1
+   1 1 0 1 1 1 1
-------------------
  1 0 0 1 0 1 0 0
```

Result is (10010100)2

2-BINARY SUBTRACTION: -

The binary subtraction rules are as follows

0 - 0 = 0

1 - 1 = 0

1 - 0 = 1, 0 - 1 = 1, with a borrow of 1

For example: -

Subtract (111.111)2 from (1010.01)2.

Solution: -

```
    1 0 1 0. 0 1 0
-     1 1 1. 1 1 1
-------------------
    0 0 1 0 .0 1 1
```

Result is (0010.011)2

BINARY MULTIPLICATION: -

The binary multiplication rules are as follows

0 x 0 = 0

1 x 1 = 1

1 x 0 = 0 and

0 x 1 = 0

For example: -

Multiply (1101)2 by (110)2.
Solution: -

```
  1 1 0 1
                          x   1 1 0
      0 0 0 0
    1 1 0 1
                        + 1 1 0 1
                        1 0 0 1 1 1 0
```

Result is $(1001110)_2$

BINARY DIVISION: -

- The binary division is very simple and similar to decimal number system. The division by '0' is meaningless. So we have only 2 rules

$0 \div 1 = 0$
$1 \div 1 = 1$

For example: -
Divide (10110)2 by (110)2.

```
110) 101101 (111.1  - 110
       1010
        110
         1001
         110
           110
         110        00
```

Result is (111.1)2

1.3-1s complement and 2s complement numbers for binary number

1's complement representation-

- The 1's complement of a binary number is obtained by changing each 0 to 1 and each 1 to 0.

For example: -
Find (1100)2 1's complement. Solution: -

Given	1	1	0	0
1's complement is	0	0	1	1

Result is (0011)2

2’s complement representation: -

 The 2’s complement of a binary number is a binary number which is obtained by adding

	= 84
=	- 84
	0

1 to the 1's complement of a number i.e. 2's complement = 1's complement + 1

For example: - Find (1010)2 2's complement

Given		1	0	1	0
1's complement is		0	1	0	1
	+				1
2's complement		0	1	1	0

Result is (0110)2

1.4-Subtraction of binary numbers in 1s and 2s complement method

Subtraction using complement method: -

1s complement: -

- In 1's complement subtraction, add the 1's complement of subtrahend to the minuend. If there is a carry out, then the carry is added to the LSB. This is called end around carry. If the MSB is 0, the result is positive. If the MSB is 1, the result is negative and is in its 1's complement form. Then take its 1's complement to get the magnitude in binary.

Subtract (10000)2 from

(11010)2 using 1's complement.

Solution: -

```
  1 1 0 1 0                  1 1 0 1 0                    =  26
- 1 0 0 0 0      =>        + 01111      (1's complement)   = - 16
          Carry  →    1 0 1 0 0 1                          + 10
                    +           1
                      01010      = +10
```

Subtraction of binary numbers in 2s complement method

Subtraction using complement method:- 2's complement: -

- In 2's complement subtraction, add the 2's complement of subtrahend to the minuend. If there is a carry out, ignore it. If the MSB is 0, the result is positive.
- If the MSB is 1, the result is negative and is in its 2's complement form.
- Then take its 2's complement to get the magnitude in binary

For example: -

Subtract (1010100)2 from (1010100)2 using 2's complement.

```
  1 0 1 0 1 0 0                  1 0 1 0 1 0 0
- 1 0 1 0 1 0 0      =>       + 0 1 0 1 1 0 0 (2's complement)
                              1 0 0 0 0 0 0 0 ( Ignore the carry)
                            =                 0 (result =
```

Hence MSB is 0. The answer is positive. So it is +0000000 =0

1.5-Use of weighted and unweighted codes and write binary equivalent

⊕

numbers in 8421,Excess-3 and gray code and vice versa

Digital codes: -

- In practice the digital electronics requires to handle data which may be numeric, alphabets and special characters.
- This requires the conversion of the incoming data into binary format before it can be processed. There is various possible ways of doing this and this process is called encoding.
- To achieve the reverse of it, we use decoders.

Weighted and non-weighted codes: -

There are two types of binary codes

1-Weighted binary codes

2-non-weighted binary codes

Weighted binary codes

- In weighted codes, for each position (or bit), there is specific weight attached.
- For example, in binary number, each bit is assigned particular weight 2n where 'n' is the bit number for n = 0,1,2,3,4 the weights are 1,2,4,8,16 respectively.
- Example: - BCD CODE, 2421

Non- weighted binary codes

- Non-weighted codes are codes which are not assigned with any weight to each digit position, i.e., each digit position within the number is not assigned fixed value.
- Example: - Excess – 3 (XS -3) code and gray codes **Excess three (xs-3) code:-**
- The Excess-3 code, also called XS-3, is a non- weighted BCD code.
- This derives it name from the fact that each binary code word is the corresponding 8421 code word plus 0011(3). It is a sequential code. It is a self-complementing code.

Gray code: -

- The gray code is a non-weighted code. It is not a BCD code. It is cyclic code because successive words in this differ in one bit position only i.e. it is a unit distance code.
- Gray code is used in instrumentation and data acquisition systems where linear or angular displacement is measured. They are also used in shaft encoders, I/O devices, A/D converters and other peripheral equipment.

Binary-to–gray conversion: -

- If an n-bit binary number is represented by Bn Bn-1 - - - - - B1 and its gray code equivalent by Gn Gn-1 G1, where Bn and Gn are the MSBs , then gray code bits are obtained from the binary code as

⊕follows Gn= Bn

Gn-1 = Bn EX-OR Bn-1

.

.

.

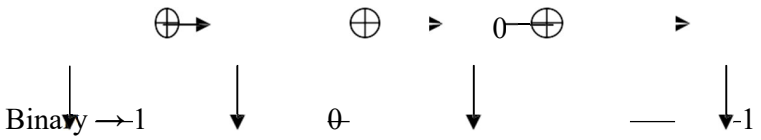
.

G1 = B2 EX-ORB1

For example

Convert the binary 1001 to the gray code.

7 bits of data	(count of 1bits)	8 bits including parity	
		even	odd
0000000	0	0000000	0000001


0
Binary
1
0
-1

Gray → 1 1

The gray code is 1101

Gray-to-binary conversion: -

- If an n-bit gray number is represented by Gn Gn-1 ------- G1 and its binary equivalent by Bn Bn1 ⊕⊕B1, then binary bits are obtained from gray bits as follows: Bn = Gn Bn-1 = Bn EX-ORGn-1 B1 = B2 EX-ORG1

For example: -

$(11100)_{\text{Gray Code}} = (\_\_?\_\_)_2$ Solution:

Gray code : 11100 (Gray code to Binary) $b4=g4=1$ $b3=b4 \oplus g3=1 \oplus 1=0$ $b2=b3 \oplus g2=0 \oplus 1=1$ $b1=b2 \oplus g1=1 \oplus 0=1$ $b0=b1 \oplus g0=1 \oplus 0=1$

∴ Binary : 10111

1.6-Importance of Parity bit

- A parity bit, or check bit, is a bit added to a string of binary code. Parity bits are a simple form of error detecting code. Parity bits are generally applied to the smallest units of a communication protocol, typically 8-bit octets (bytes), although they can also be applied separately to an entire message string of bits.

10100 01	3	1**1**	1**0**
11010 01	4	1**0**	1**1**
11111 11	7	1**1**	1**0**

- The parity bit ensures that the total number of 1-bits in the string is even or odd.[1] Accordingly, there are two variants of parity bits: even parity bit and odd parity bit. In the case of even parity, for a given set of bits, the occurrences of bits whose value is 1 are counted.
- If that count is odd, the parity bit value is set to 1, making the total count of occurrences of 1s in the whole set (including the parity bit) an even number.
- If the count of 1s in a given set of bits is already even, the parity bit's value is 0. In the case of odd parity, the coding is reversed.

1.7-Logic Gates And, Or, Not,Nand,Nor And Ex-Or Gate With

Truth Table:-

- Logic gates are the fundamental building blocks of digital systems. There are 3 basic types of gates AND, OR and NOT.
- Logic gates are electronic circuits because they are made up of a number of electronic devices and components.
- Inputs and outputs of logic gates can occur only in 2 levels. These two levels are termed HIGH and LOW, or TRUE and FALSE, or ON and OFF or simply 1 and 0.
- The table which lists all the possible combinations of input variables and the corresponding outputs is called a truth table.

Different types of logic gates: not gate (inverter):-

- A NOT gate, also called and inverter, has only one input and one output. It is a device whose output is always the complement of its input.
- The output of a NOT gate is the logic 1 state when its input is in logic 0 state and the logic 0 state when its inputs is in logic 1 state.
- IC No. :- 7404

Logic Symbol

Truth table

INPUT A	OUTPUT A'
0	1
1	0

AND Gate: -

- An AND gate has two or more inputs but only one output.
- The output is logic 1 state only when each one of its inputs is at logic 1 state. The output is logic 0 state even if one of its inputs is at logic 0 state.
- IC No. :- 7408

Logic Symbol

of inputs, the output is logic 1.

INPUT		OUTPUT
A	B	Y=A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR gate:-

- ✓ An OR gate may have two or more inputs but only one output.
- ✓ The output is logic 1 state, even if one of its input is in logic 1 state.
- ✓ The output is logic 0 state, only when each one of its inputs is in logic state.
- ✓ IC No.:- 7432

Logic Symbol

INPUT		OUTPUT
A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1

NAND gate:-

- ✓ NAND gate is a combination of an AND gate and a NOT gate.
- ✓ The output is logic 0 when each of the input is logic 1 and for any other combination

- ✓ IC No.:- 7400
- ✓ two input NAND gate 7410
- ✓ three input NAND gate 7420
- ✓ four input NAND gate 7430

Logic Symbol-

INPUT		OUTPUT
A	B	Y=A.B
0	0	0
0	1	1
1	0	1
1	1	1

NOR GATE:-

- ✓ NOR gate is a combination of an OR gate and a NOT gate.
- ✓ The output is logic 1, only when each one of its input is logic 0 and for any other combination of inputs, the output is a logic 0 level.

- IC No.:- 7402 two input NOR gate
- 7427 three input NOR gate
- 7425 four input NOR gate

INPUT		OUTPUT
A	B	$\overline{Q = A + B}$
0	0	1
0	1	0
1	0	0

Input		Output
A	B	Y
0	0	0
0	1	1

1	0	1
1	1	0

Logic Symbol **Truth Table**

1 1 0

EXCLUSIVE – OR (X-OR) GATE:-

- An X-OR gate is a two input, one output logic circuit.
- The output is logic 1 when one and only one of its two inputs is logic 1. When both the inputs is logic 0 or when both the inputs is logic 1, the output is logic 0.
- IC No.:- 7486

Logic Symbol

INPUTS are A and B OUTPUT is Q = A $\oplus$ B
= A' B + A B'

Truth Table

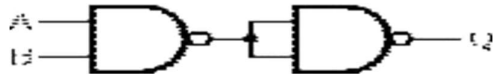
EXCLUSIVE – NOR (X-NOR) GATE:-

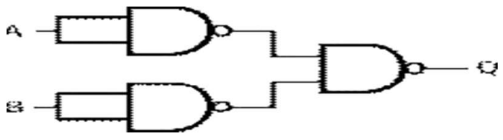
- An X-NOR gate is the combination of an X-OR gate and a NOT gate. An X-NOR gate is a two input, one output logic circuit.
- The output is logic 1 only when both the inputs are logic 0 or when both the inputs is 1. The output is logic 0 when one of the inputs is logic 0 and other is 1.
- IC No.:- 74266

Logic Symbol

Input		Output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

OUT =A’ B’ + A B





= A XNOR B

1.8-UNIVERSAL GATES & ITS REALISATION:-

Introduction

- There are 3 basic gates AND, OR and NOT, there are two universal gates NAND and NOR, each of which can realize logic circuits single handedly.
- The NAND and NOR gates are called universal building blocks. Both NAND and NOR gates can perform all logic functions i.e. AND, OR, NOT, EXOR and EXNOR

NAND GATE:-

Inverter from NAND gate

Input = A Output Q = A'

AND gate from NAND gate

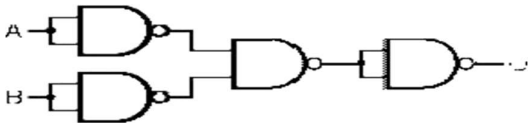
Input s are A and B Output Q = A.B

OR gate from NAND gate

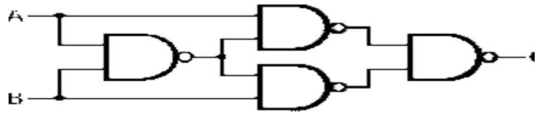
Inputs are A and B Output Q = A+B

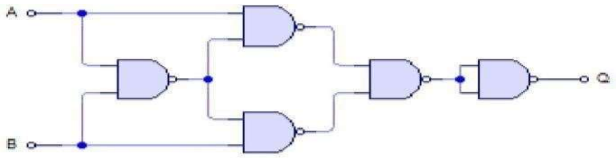
NOR gate from NAND gate

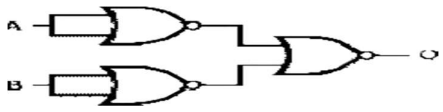
Inputs are A and B
Output Q = A+B

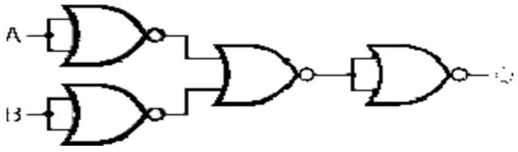


EX-OR gate from NAND gate









Inputs are A and B
Output Q = A B + AB

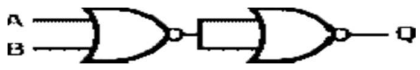
EX-NOR gate From NAND gate
Inputs are A and B Output Q = A B + A B

NOR GATE

Inverter from NOR gate
Input = A
Output Q = A

AND gate from NOR gate
Input s are A and B
Output Q = A.B

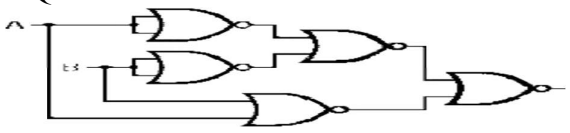
OR gate from NOR gate
Inputs are A and B Output Q = A+B



NAND gate from NOR gate

Inputs are A and B Output Q = (A.B)' **EX-OR gate from NOR gate**

Inputs are A and B Output Q = A' B + A B'



EX-NOR gate From NOR gate

Inputs are A and B

Output Q = A' B' + A B

1.9-Different postulates and De-Morgan's thermos in Boolean algebra

Introduction

- Switching circuits are also called logic circuits, gates circuits and digital circuits. Switching algebra is also called Boolean algebra.
- Boolean algebra is a system of mathematical logic. It is an algebraic system consisting of the set of elements (0, 1), two binary operators called OR and AND and unary operator called NOT.
- It is a way to express logic functions algebraically.
- Any complex logic can be expressed by a Boolean function.

1.10-Use of Boolean algebra for simplification of logic expression

- Axioms or postulates of Boolean algebra are set of logical expressions that are accepted without proof and upon which we can build a set of useful theorems.
- Actually, axioms are nothing more than the definitions of the three basic logic operations AND, OR and INVERTER.
- Each axiom can be interpreted as the outcome of an operation performed by a logic gate.

AND Rule

0.0=0

0.1=0

1.0=0

1.1=1

OR Rule

0+0=0

0+1=1

1+0=1

1+1=1

NOT Rule **1=0**

0=1

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

B	A	B+ A
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A . B
0	0	0
0	1	0
1	0	0
1	1	1

B	A	B. A
0	0	0
0	1	0
1	0	0
1	1	1

Complementation Laws:-

The term complement simply means to invert, i.e. to changes 0s to 1s and 1s to 0s. The five laws of complementation are as follows:

Law 1: 0 '= 1
Law 2: 1' = 0
Law 3: if A = 0, then A' = 1
Law 4: if A = 1, then A' = 0
Law 5: A'' = 0 (double complementation law

OR Laws:-

The four OR laws are as follows

Law 1: A + 0 = A(Null law)
Law 2: A + 1 = 1(Identity law)
Law 3: A + A = A

Law 4: $A + \bar{A} = 1$

AND Laws:-

The four AND laws are as follows

Law 1: A . 0 = 0(Null law)
Law 2: A . 1 = 1(Identity law)
Law 3: A . A = A
Law 4: A .A = 0

Commutative Laws:-

Commutative laws allow change in position of AND or OR variables. There are two commutative laws.

Law 1: A + B = B + A

Proof

=

Law 2: A . B = B . A

Proof

=

This law can be extended to any number of variables. For example A.B. C = B. C. A = C. A. B = B.

A	B	C	A+B	(A+B)+C
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

A	B	C	B+C	A+(B+C)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

A	B	C	AB	(AB)C
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

A	B	C	B.C	A(B.C)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

A	B	C	AB	AC	A+(B+C)
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0

1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

A	B	C	B+C	A(B+C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1

1	1	0	1	1
1	1	1	1	1

Associative Laws:-

The associative laws allow grouping of variables. There are 2 associative laws. Law 1: (A + B) + C = A + (B + C)

Proof

=

Law 2: (A .B) C = A (B .C)

This law can be extended to any number of variables.
For example A(BCD) = (ABC)D = (AB) (CD)

Distributive Laws:-

The distributive laws allow factoring or multiplying out of expressions. There are two distributive laws.
Law 1: A (B + C) = AB + AC

Proof

=

A	B	A + B	$\overline{A + B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

A	B	$\overline{A}$	$\overline{B}$	$\overline{A}\,\overline{B}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

A	B	A . B	$\overline{A \cdot B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

A	B	$\overline{A}$	$\overline{B}$	$\overline{A + B}$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

1. $0' = 1$ $\quad 1' = 0$
2. $0 \cdot 1 = 0$ $\quad 1 + 0 = 1$
3. $0 \cdot 0 = 0$ $\quad 1 + 1 = 1$
4. $1 \cdot 1 = 1$ $\quad 0 + 0 = 0$
5. $A \cdot 0 = 0$ $\quad A + 1 = 1$
6. $A \cdot 1 = A$ $\quad A + 0 = A$
7. $A \cdot A = A$ $\quad A + A = A$
8. $A \cdot A = 0$ $\quad A + A = 1$
9. $A \cdot B = B \cdot A$ $\quad A + B = B + A$
10. $A \cdot (B \cdot C) = (A \cdot B) \cdot C$ $\quad A + (B + C) = (A + B) + C$
11. $A \cdot (B + C) = AB + AC$ C) $\quad A + BC = (A + B) (A +$
12. $A(A + B) = A$ $\quad A + AB = A$

De Morgan's Theorem:-

De Morgan's theorem represents two laws in Boolean algebra. Law 1:(A + B)' =A'· B

~~Proof~~

This law states that the complement of a sum of variables is equal to the product of their individual complements.
~~Law 2~~: (A· B)' = A' + B' Proof

This law states that the complement of a product of variables is equal to the sum of their individual complements.

DUALITY

Given expression

13. $\overline{A} \cdot (\overline{A} \cdot B) = A \cdot B$ $\overline{A} + \overline{A} + \overline{B} = A + B$

14. AB = A + B A + B = A B

15. (A + B) AC (A+ C) (B + C) = (A+ B)(A + C) AB + AC + BC = AB +

16. A + BC = (A + B)(A + C) A(B+ C) = A B +A C

17. (A+C)(A+B) = AB+AC AC+AB=(A+B) (A+C)

18. (A+B)(C+D) = AC + AD + BC + BD (AB+CD) = (A+C)(A+D)(B+C)(B+D)

19. A+B=AB+ AB + AB AB=(A+B)(A+B) (A+B)

20. AB + A + AB = 0 A + B · A · (A + B) = 1

1.11-Karlaugh Map for 2, 3, 4 variables, simplification of SOP and POS logic expression using K-Map

- This is also called disjunctive Canonical Form (DCF) or Expanded Sum of Products Form or Canonical Sum of Products Form.
- In this form, the function is the sum of a number of products terms where each product term contains all variables of the function either in complemented or un complemented form.
- This can also be derived from the truth table by finding the sum of all the terms that corresponds to those combinations for which 'f ' assumes the value 1. **For example**

f(A, B, C) = A'B + B'C = A'B (C + C') + B'C (A + A')
= A BC + ABC + ABC + ABC

- The product term which contains all the variables of the functions either in complemented or un complemented form is called a minterm.
- The minterm is denoted as mo, m1, m2
- An 'n' variable function can have 2n minterms.
- Another way of representing the function in canonical SOP form is the showing the sum of minterms for which the function equals to 1. For exam**pl**e

f (A, B, C) = m1 + m2+ m3 + m5 or f (A, B, C) =∑ m (1, 2, 3, 5) where ∑m represents the sum of all the minterms whose decimal codes are given the parenthesis.

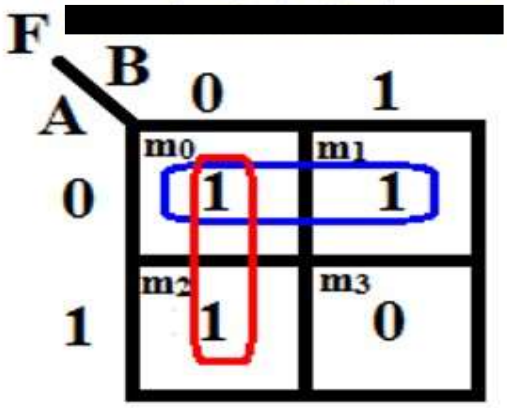
Product- of-sums form:-

- This form is also called as Conjunctive Canonical Form (CCF) or Expanded Product of – Sums Form
  or Canonical Product Of Sums Form.
- This is by considering the combinations for which f = 0 Each term is a sum of all the variables.

The function f (A, B, C) = (A + B + C·C) + (A + B + C·C)
= (A + B + C) (A + B + C) (A + B + C) (A + B + C)

- The sum term which contains each of the 'n' variables in either complemented or un complemented form is called a maxterm.
- Maxterm is represented as M0, M1, M2,
- Thus CCF of 'f' may be written as f(A, B, C)= M0 · M4 · M6· M7 or f(A, B, C) = (0, 4, 6, 7)

- Where represented the product of all maxterms.



Conversion between canonical form:-

 The complement of a function expressed as the sum of minterms equals the sum of minterms missing from theoriginal function. Example:-

f(A, B, C) = ∑m(0,2,4,6,7)

This has a complement that can be expressed as f (A, B, C)' =∑ m(1, 3, 5) = m1 + m3 + m5

If we complement f by De- Morgan's theorem we obtain 'f' in a form. f =(m1+ m3 + m5)' = m1'. m3'. m5'

= M1 M3 M5 =∏ M(1, 3 ,5)

Karnaugh map or k-map & minimisation of logical expressions, don't Care conditions:-

 The K- map is a chart or a graph, composed of an arrangement of adjacent cells, each representing a particular combination of variables in sum or product form.

 The K- map is systematic method of simplifying the Boolean expression.

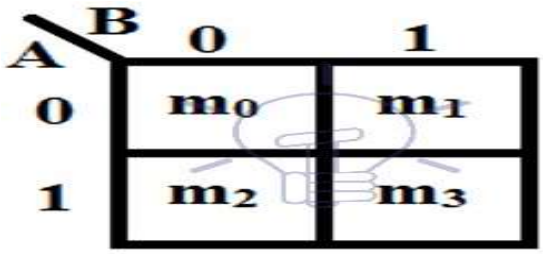
Two variable k-map:-

 A two variable expression can have $2^2 = 4$ possible combinations of the input variables A and B.

Mapping of SOP Expression:-

 The 2 variable K-map has $2^2 = 4$ squares. These squares are called cells.

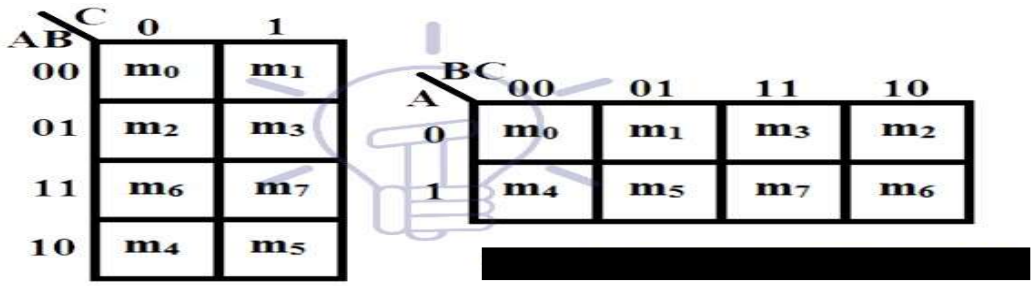
 A '1' is placed in any square indicates that corresponding minterm is included in the output expression, and a 0 or no entry in any square indicates that the corresponding min term does not appear in the expression for output.

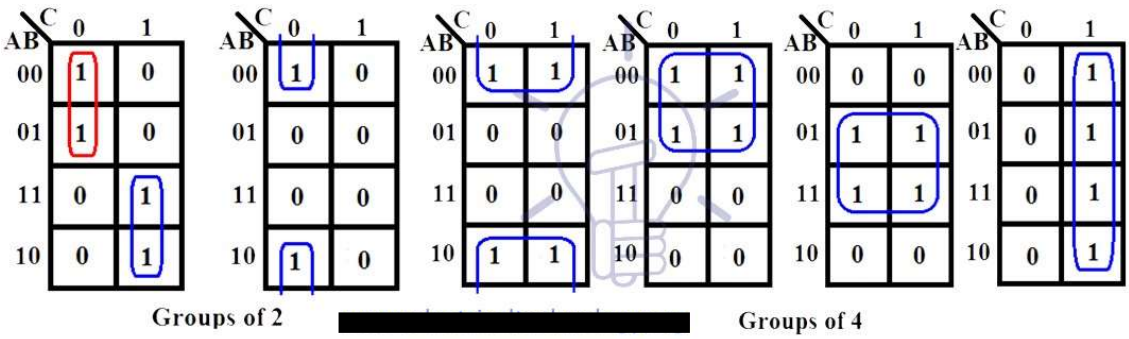


Example of 2 Variable K-Map Function F (A, B)

$F = \sum (m_0, m_1, m_2) = \bar{A}\,\bar{B} + \bar{A}\,B + A\bar{B}$

The simplifies expression will be the sum of these two terms as given below,





Example of 3 Variable K-Map

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

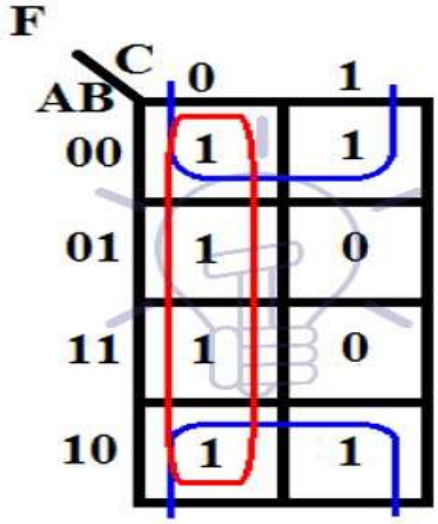
3-Variable K-Map

 variables make $2^n=2^3=8$ min terms, so the Karnaugh map of 3 variables will have 8 squares(cells) as shown in the figure given below.

Some examples of grouping:

We can make groups of 2, 4 & 8 cells having same 1s or 0s.

$F(A,B,C) = \sum (m_0, m_1, m_2, m_4, m_5, m_6)$

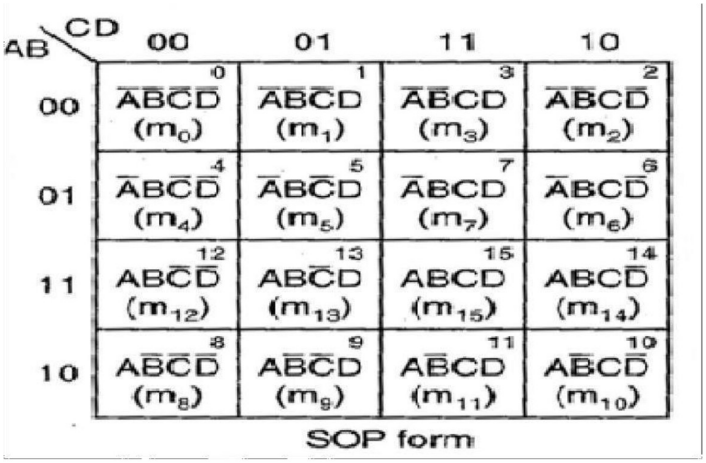


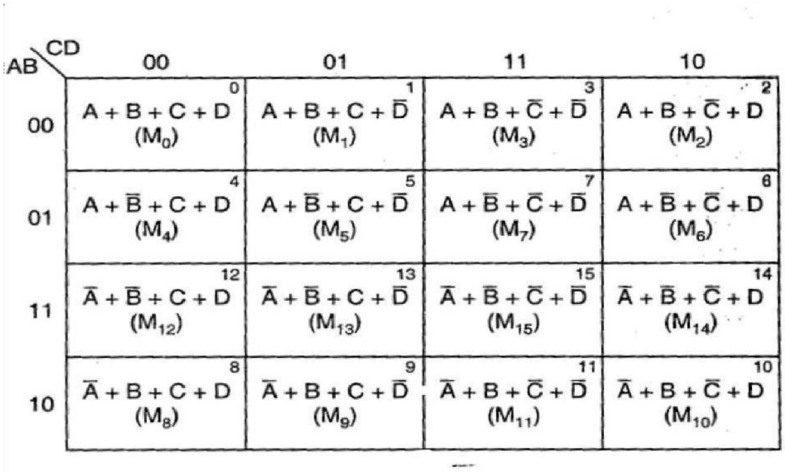
The sum of these two terms will make the simplified expression of the function is $F = \overline{B} + \overline{C}$

4-variable K-Map

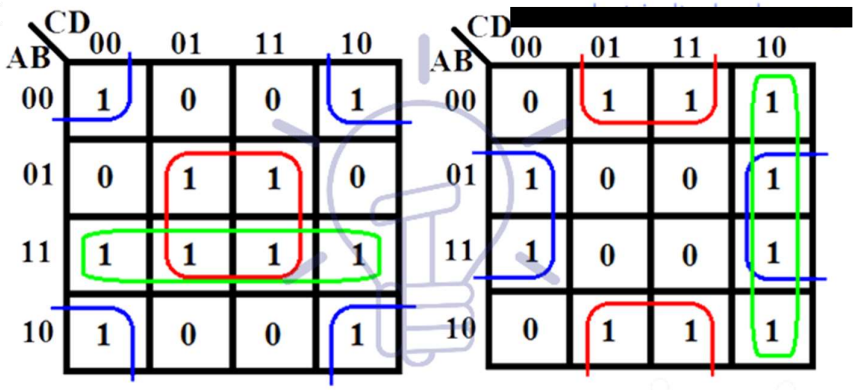
- A four variable (A, B, C, D) expression can have $2^4 = 16$ possible combinations of input variables.
- A four variable K-map has $2^4 = 16$ squares or cells and each square on the map represents either a minterm or a maxterm as shown in the figure below.
- The binary number designations of the rows and columns are in the gray code.
- The binary numbers along the top of the map indicate the conditions of C and D along any column and binary numbers along left side indicate the conditions of A and B along any row.
- The numbers in the top right corners of the squares indicate the minterm or maxterm designations.

- 4 variables have $2^n=2^4=16$ minterms. So a 4-variable k-map will have 16 cells as shown in the figure given below.

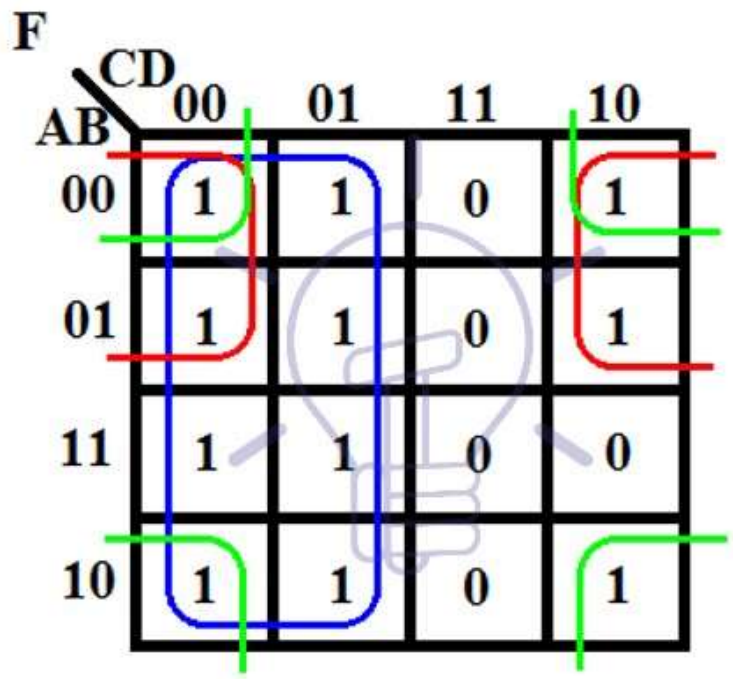




Some example of grouping in 4-variable k-map is given .


CD
AB
00 01 11 10
00 1 0 0 1
01 0 1 1 0
11 1 1 1 1
10 1 0 0 1
CD
AB
00 01 11 10
00 0 1 1 1
01 1 0 0 1
11 1 0 0 1
10 0 1 1 1

$F(A,B,C,D) = \sum(m_0, m_1, m_2, m_4, m_5, m_6, m_8, m_9, m_{12}, m_{13}, m_{14})$

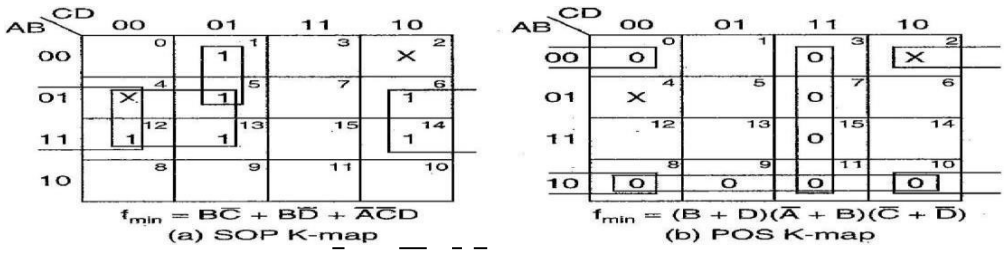


So the expression will be
$F = \bar{C} + \bar{B}\,\bar{D} + \bar{A}\,\bar{D}$

DON'T CARE COMBINATIONS: -

- The combinations for which the values of the expression are not specified are called don't care combinations or optional combinations and such expression stand incompletely specified.
- The output is a don't care for these invalid combinations. The don't care terms are denoted by d or X.
- During the process of designing using SOP maps, each don't care is treated as 1 to reduce the map otherwise it is treated as 0 and left alone.
- During the process of designing using POS maps, each don't care is treated as 0 to reduce the map otherwise it is treated as 1 and left alone.
- A standard SOP expression with don't cares can be converted into standard POS form by keeping the don't cares as they are, and the missing minterms of the SOP form

are written as the maxterms of the POS form.



- Similarly, to convert a standard POS expression with don't cares can be converted into standard SOP form by keeping the don't cares as they are, and the missing maxterms of the POS form are written as the minterms of the SOP form.

Example:-

Reduce the expression f = ∑ m(1, 5, 6, 12, 13, 14) + d(2, 4) using K- map.

Solution:-

The given expression in SOP form is f = ∑ m (1, 5, 6, 12, 13, 14) + d(2, 4)

The given expression in POS form is f = π M (0, 3, 7, 8, 9, 10, 11,15) + d(2, 4)

The minimal of SOP expression is fmin = BC + BD +ACD

The minimal of POS expression is fmin = $(B + D)(\bar{A} + B)(C + D)$

Possible Short Type Questions With Answers

1. Convert (10101)2 to decimal.

Solution :

(Positional weight) $2^4\ 2^3\ 2^2\ 2^1\ 2^0$ Binary number 10101

$= (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$

= 16 + 0+ 4+ 0+ 1

= (21)10

2. Convert (1011011011)2 into hexadecimal. Solution:

Given Binary number 10 1101 1011

Group of 4 bits are 0010 1101 1011

Convert each group into hex = 2 D B

The result is (2DB)16

3. What do you mean by radix of a number. (W-20 Exam)

Solution-

The base of each number system is also called the radix. The radix of a decimal number is ten, and the radix of binary is **two**.

4. Find 2's complement of $(110101.01)_2$. (W-20 Exam)

Solution

1's complement is 001010.10

2's complement is 001010.10

\+ 1

00101011

5. Subtract (111.111)2 from (1010.01)2

.Solution :-

1 0 1 0 . 0 1 0

1 1 1 . 1 1 1

0 0 1 0 .0 1 1

Result is (0010.011)2

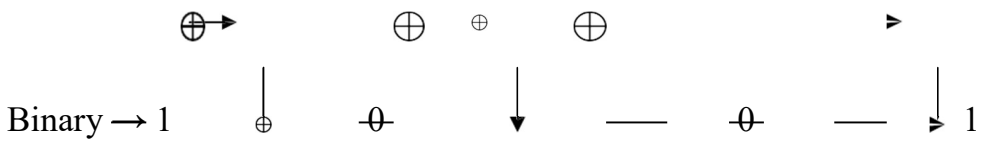
6. Find (1100)2 1's complement. Solution :-

Given 1 1 0 0

1's complement is 0 0 1 1

7. Convert the binary 1001 to the Gray code.

Solution :-`



Gray → 1 1 0 1

The gray code is 1101

8. Define Don't care condition

The combinations for which the values of the expression are not specified are called don't care combinations or optional combinations and such expression stand incompletely specified.

The output is a don't care for these invalid combinations. The don't care terms are denoted by d or X.

Possible Long Questions

1- Find out the logic expression for f=m(0,1,2,3,5,7,8,9,10,12,13) using KMap.

2- What are the universal gate? Derive other gate using any one of the universal gate.

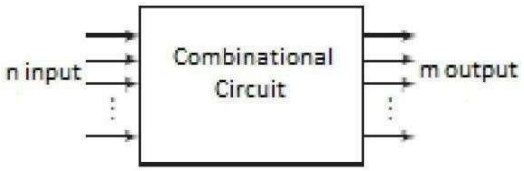
3- Draw the logic diagram of the following Boolean expression. Y=AB(C+BD).

4 - Which gates are referred to a universal gate and why?

5- State and prove De-Morgan's Theorem. [W-20]

6- Minimize the four variable logic expression using K-MAP. [W-20]

CHAPTER 2



COMBINATIONAL LOGIC CIRCUITS

LEARING OBJECTIVES:

2.1-Give the concept of Combinational circuits

2.2-Half adder circuit and verify its functionality using truth table.

2.3-Realize a half adder using NAND gate only and NOR gate only.

2.4-Full adder circuit and explain its operation with truth table.

2.5-Realize full adder using two half adder and OR gate and write truth table.

2.6-Full subtractor circuit and explain its operation with truth table.

2.7-Operation of 4X1 Multiplexers and 1X4 demultiplexer.

2.8-Working of Binary-Decimal Encoder and 3x8 Decoder.

2.9-Working of Two-bit magnitude comparator.

2.1. Give the Cocept Of Combinational circuits

- A combinational circuit consists of logic gates whose outputs at any time are determined from only the present combination of inputs. .
- The n input binary variables come from an external source; the m output variables are produced by the internal combinational logic circuit and go to an external destination.
- Each input and output variable exists physically as an analog signal whose values are interpreted to be a binary signal that represents logic 1and logic 0.

2.2-Half adder circuit and verify its functionality using truth table.

Half Adder

- This circuit needs two binary inputs and two binary outputs.
- The input variables designate the augend and addend bits; the output variables produce the sum and carry. Symbols x and y are assigned to the two inputs and S (for sum) and C (for carry) to the outputs.
- The C output is 1 only when both inputs are 1. The S output represents the least significant bit of the sum.
- The simplified Boolean functions for the two outputs can be obtained directly

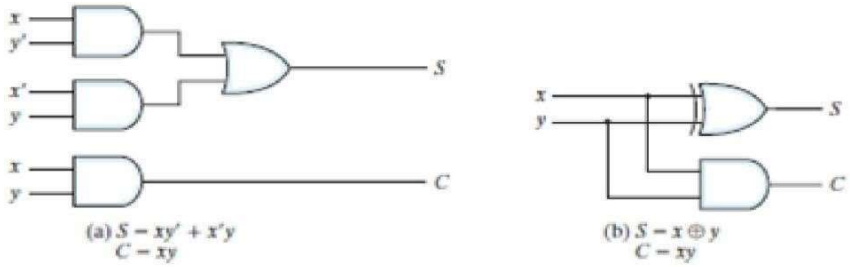
. The simplified sum-of-pro

x	y	D	B
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Truth Table

S = x'y + xy' C = xy

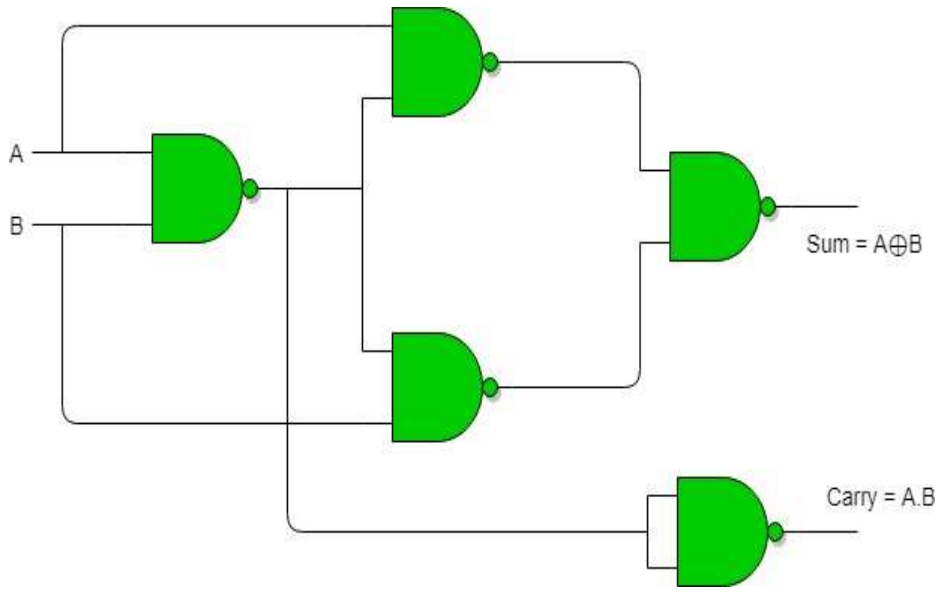
- The logic diagram of the half adder implemented in sum of products is shown in the below figure. It can be also implemented with an exclusive-OR and an AND gate



2.3-Realize a half adder using NAND gate only and NOR gate only

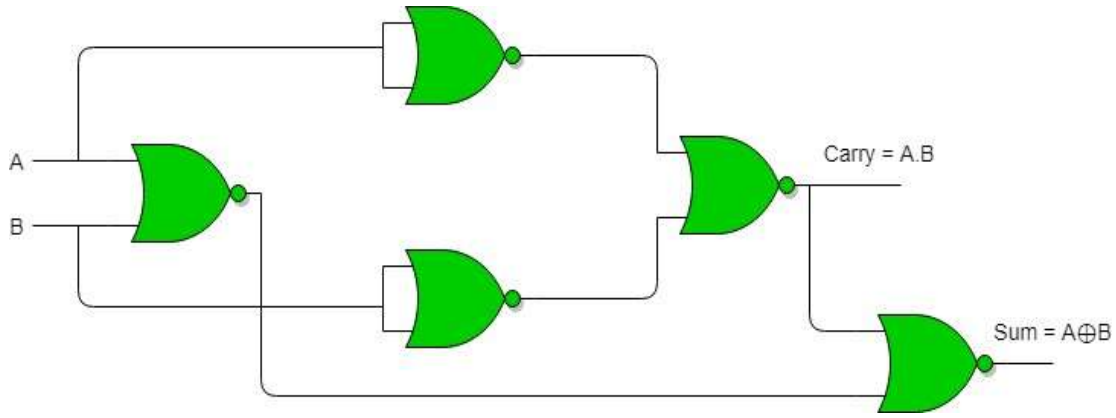
Half adder using NAND gate only

- As we know that NAND and NOR are called universal gates as any logic system can be implemented using these two.
- The Half Adder Circuit can also be implemented using them. We know that a half adder circuit has one Ex – OR gate and one AND gate.



Half Adder using NOR gate only

Five NOR gates are required in order to design a half adder. The circuit to realize half adder using NOR gates is shown below



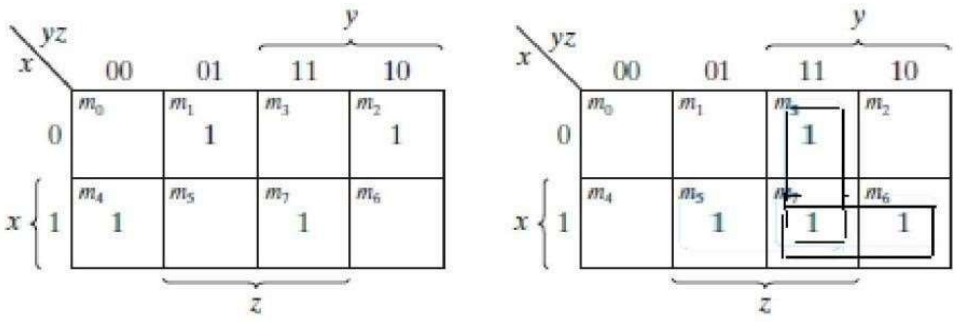
2.4-Full adder circuit and explain its operation with truth table

Full adder

- Full adder is a combinational circuit design to add more than two single bit number with carry
- It consists of three inputs and two outputs. Two of the input variables, denoted by x and y , represent the two significant bits to be added.
- The third input, z , represents the carry from the previous lower significant position. The two output are designated by the symbol S for sum and for carry.

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Truth Table



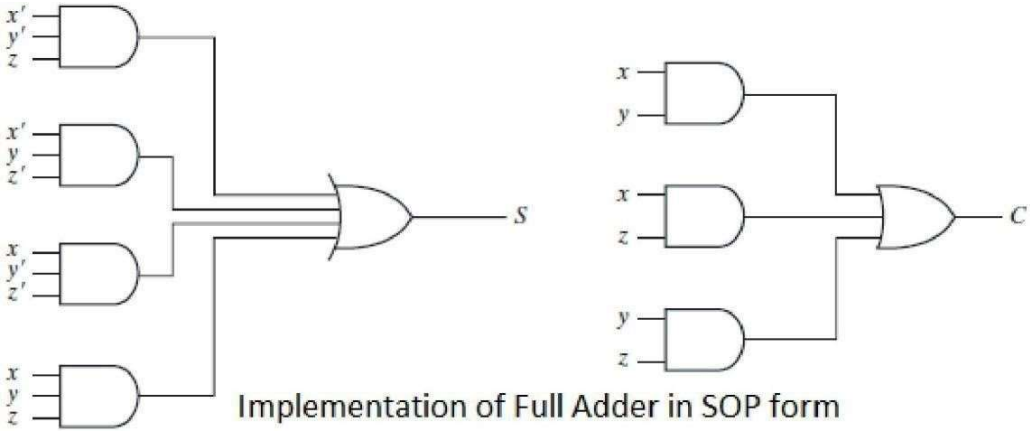
(a) $S = x'y'z + x'yz' + xy'z' + xyz$ (b) $C = xy + xz + yz$

K-Map for full adder

The simplified expressions are

S = x'y'z + x'yz' + xy'z' + xyz

C = xy + xz + yz



Implementation of Full Adder in SOP form

2.5-Realize full adder using two half adder and OR gate and write truth table

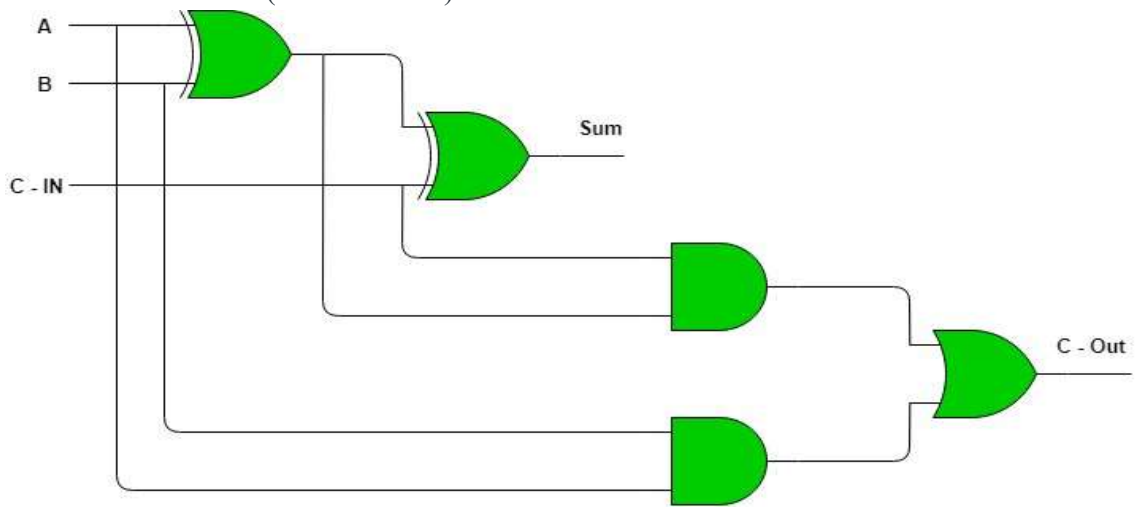
Full Adder using two Half Adder **Truth Table for full Adder**

Inputs			Outputs	
A	B	C – IN	Sum	C - Out
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

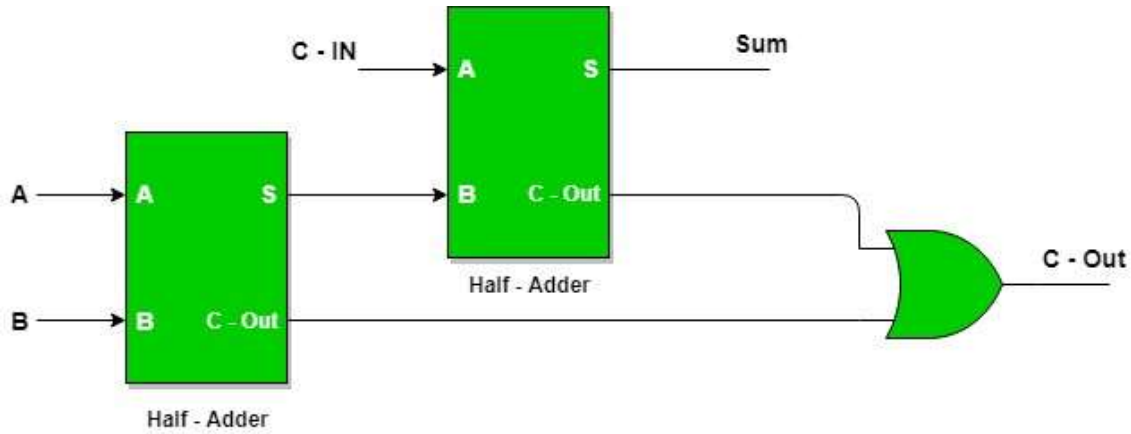
Logical Expression for SUM:

= A’ B’ C-IN + A’ B C-IN’ + A B’ C-IN’ + A B C-IN
= C-IN (A’ B’ + A B) + C-IN’ (A’ B + A B’)
= C-IN XOR (A XOR B)
= (1,2,4,7)

Logical Expression for C-OUT:
= A’ B C-IN + A B’ C-IN + A B C-IN’ + A B C-IN
= A B + B C-IN + A C-IN = (3,5,6,7)
Another form in which C-OUT can be implemented:
= A B + A C-IN + B C-IN (A + A’)
= A B C-IN + A B + A C-IN + A’ B C-IN
= A B (1 +C-IN) + A C-IN + A’ B C-IN
= A B + A C-IN + A’ B C-IN
= A B + A C-IN (B + B’) + A’ B C-IN
= A B C-IN + A B + A B’ C-IN + A’ B C-IN
= A B (C-IN + 1) + A B’ C-IN + A’ B C-IN
= A B + A B’ C-IN + A’ B C-IN
= AB + C-IN (A’ B + A B’)
Therefore COUT = AB + C-IN (A EX – OR B)



Block Diagram of full Adder using Two half Adder



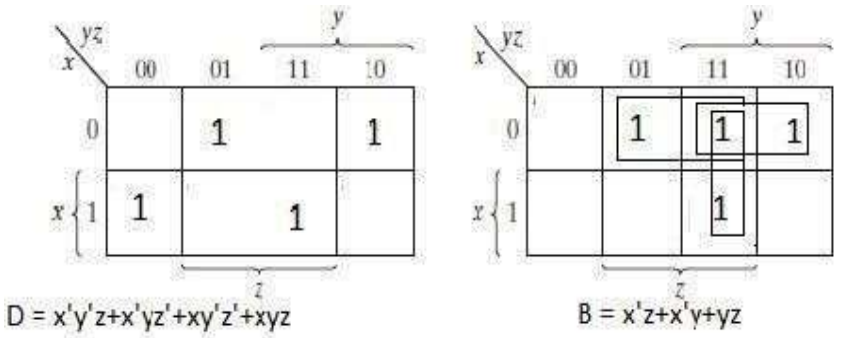
2.6-Full subtractor circuit and explain its operation with truth table

Full subtractor

- A full subtractor is a combinational circuit that forms the arithmetic subtraction operation of three bits.
- It consists of three inputs and two outputs. Two of the input variables, denoted by x and y , represent the two significant bits to be subtracted. The third input, z , is , subtracted from the result 0f the first subtraction
- The two outputs are designated by the symbols D for difference and B for borrow.
- The binary variable D gives the value of the least significant bit of the difference. The binary variable B gives the output borrow formed during the subtraction process.

x	y	z	D	B
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

Truth Table



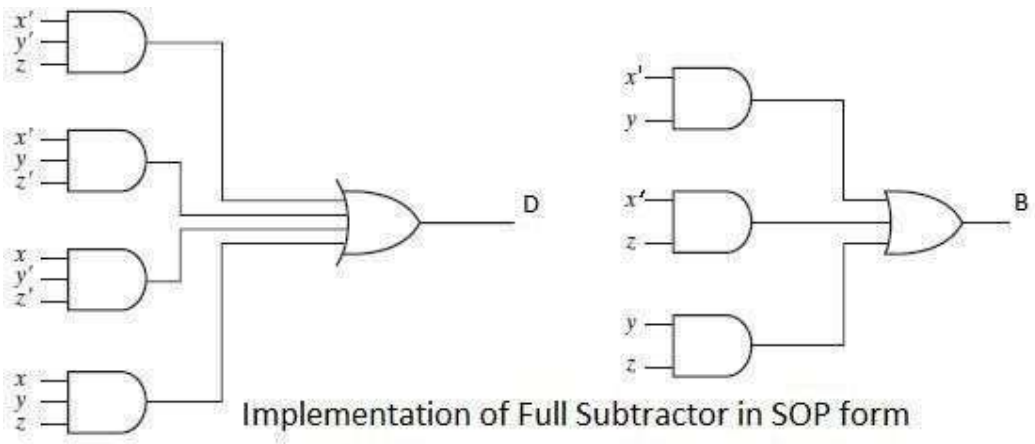
K-Map for full Subtractor

The simplified expressions are

D = x’y’z + x’yz’ + xy’z’ + xyz

B = x’z + x’y + yz

Logic Diagram

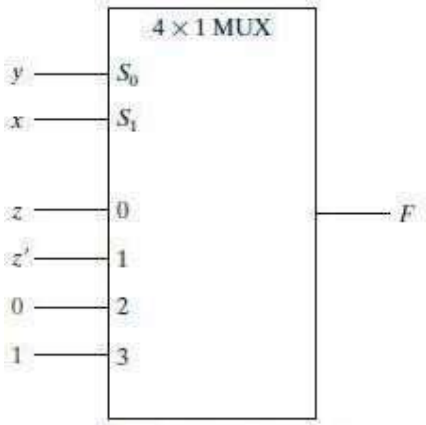


Implementation of Full Subtractor in SOP form

2.7-Operation of 4X1 Multiplexers and 1X4 demultiplexer

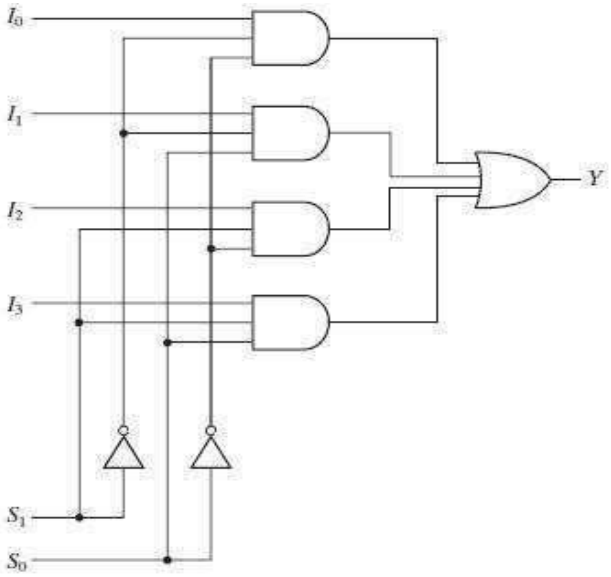
Multiplexer

- A multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.
- The selection of a particular input line is controlled by a set of selection lines.
- Normally, there are 2^n input lines and n selection lines whose bit combinations determine which input is selected.
- A multiplexer is also called a data selector, since it selects one of many inputs and steers the binary information to the output



(b) Multiplexer implementation

From the Truth Table Output Y=S0'S1'I0+S0' S1 I1+S0 S1' I2+S0 S1 I3



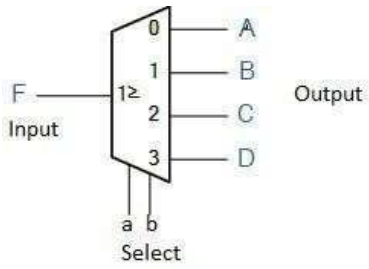
Logic diagram

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Truth table

1x4 De-MUX

$F = a'b'A + a'bB + ab'C + abD$



Data Input	Select Inputs		Outputs			
D	S_1	S_0	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0

- The data distributor, known more commonly as a Demultiplexer or "Demux" for short, is the exact opposite of the Multiplexer.
- The demultiplexer takes one single input data line and then switches it to any one of a number of individual output lines one at a time.
- The demultiplexer converts a serial data signal at the input to a parallel data at its output lines
- The Boolean expression for this 1-to-4 demultiplexer above with outputs A to D and data select lines a, b is given as:

According to Truth table

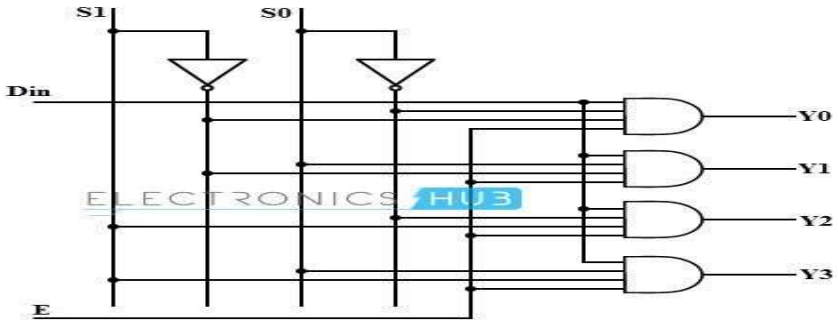
Y0=E S1’ S0’ Din

Y1=E S1’ S0 Din

Y2=E S1 S0’ Din

Y3=E S1 S0 Din

Logic Diagram



2.8-Working of Binary-Decimal Encoder and 3x8 Decoder Binary-Decimal Encoder

- Encoders are used as code converters in computer systems.
- These are available as IC’s in the market.
- To convert a decimal number into binary a Decimal to BCD Encoder is used.
- In the BCD system, the decimal number is represented as the four-digit binary.
- It can convert the decimal numbers from 0 to 9 into the binary stream.The encoder is a combinational logic circuit.
- The reverse of the encoder is a decoder that performs the reverse action.
- The truth table of Decimal to BCD encoder is given below,

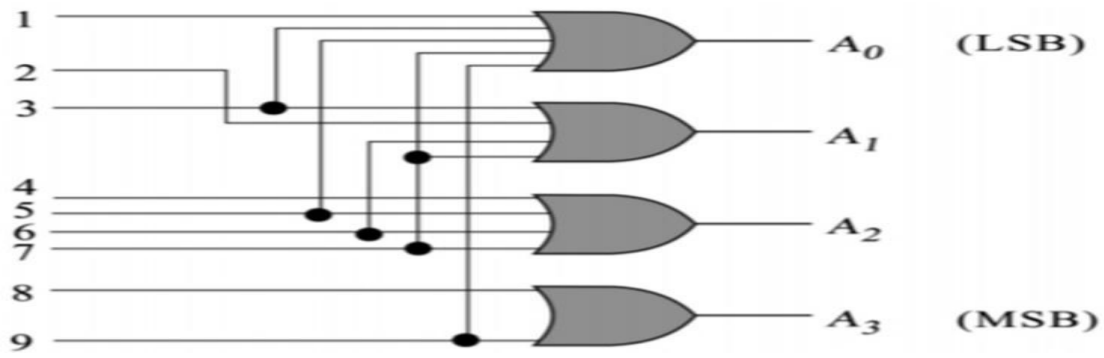
Decimal Digit	BCD Code			
	A3	A2	A1	A0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

From the truth table above form the equations for the words A3, A2, A1, A0. Thus the logical

equations are as below-

A3 = 8+9: A2 = 4+5+6+7 : A1 =2+3+6+7 : A0= 1+3+5+7+9

Now, considering the logic equations above, form the combinational circuit with OR gates.



3x8 Decoder

- A decoder is a combinational logic circuit that is used to change the code into a set of signals. It is the reverse process of an encoder.
- A decoder circuit takes multiple inputs and gives multiple outputs. A decoder circuit takes binary data of 'n' inputs into '2^n' unique output.
- In 3 to 8 line decoder, it includes three inputs and eight outputs. Here the inputs are represented through A, B & C whereas the outputs are represented through D0, D1, D2…D7.
- The selection of 8 outputs can be done based on the three inputs. So, the truth table of this 3 line to 8 line decoder is shown below.
- From the following truth table, we can observe that simply one of 8 outputs from DO – D7 can be selected depending on 3 select inputs.

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	1	0	0	0	0	0	0	1

From the above truth table of 3 lines to 8 line decoder, the logic expression can be defined as

D0 = A'B'C'

D1= A'B'C

D2 = A'BC'

D3 = A'BC

D4 = AB'C'

D5= AB'C

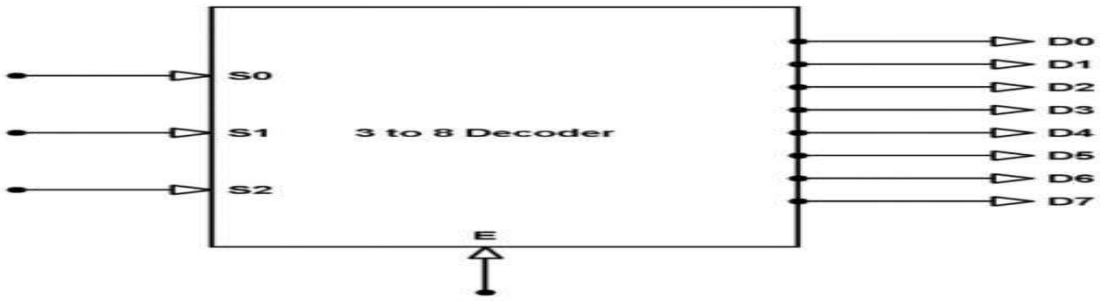
D6 = ABC'

D7 = ABC

From the above Boolean expressions, the implementation of 3 to 8 decoder circuit can be done with the help of three NOT gates & 8-three input AND gates.

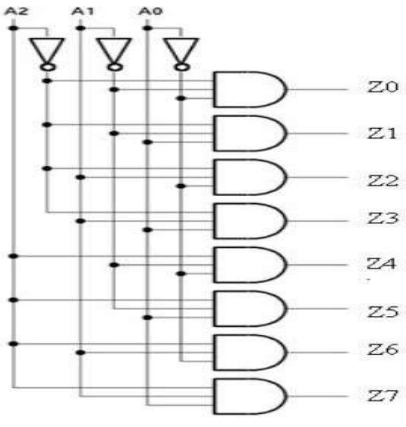
In the above circuit, the three inputs can be decoded into 8 outputs, where every output represents one of the midterms of the three input variables.

3 Line to 8 Line Decoder Block Diagram



3 to 8 Line Decoder Block Diagram

The decoder circuit works only when the Enable pin (E) is high. S0, S1 and S2 are three different inputs and D0, D1, D2, D3. D4. D5. D6. D7 are the eight outputs. The logic diagram of the 3 to 8 line **decoder** is shown below.



3 to 8 Decoder Circuit

2.9-Working of Two-bit magnitude comparator

Comparator

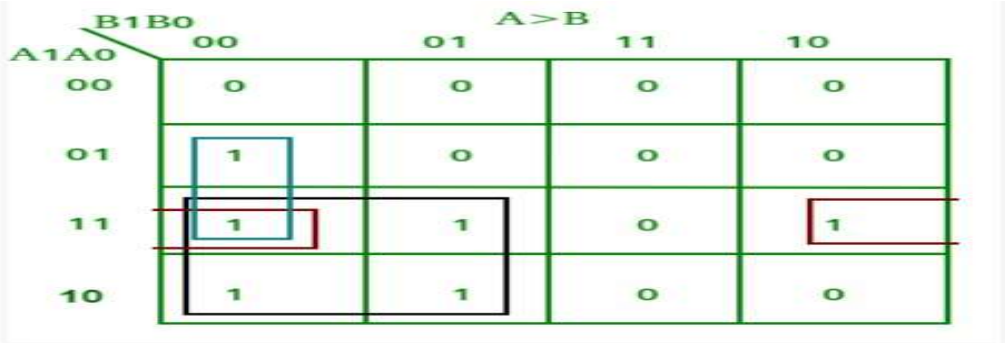
- A magnitude digital Comparator is a combinational circuit that **compares two digital or binary numbers** in order to find out whether one binary number is equal, less than or greater than the other binary number.
- We logically design a circuit for which we will have two inputs one for A and other for B and have three output terminals, one for A > B condition, one for A = B condition and one for A < B condition.

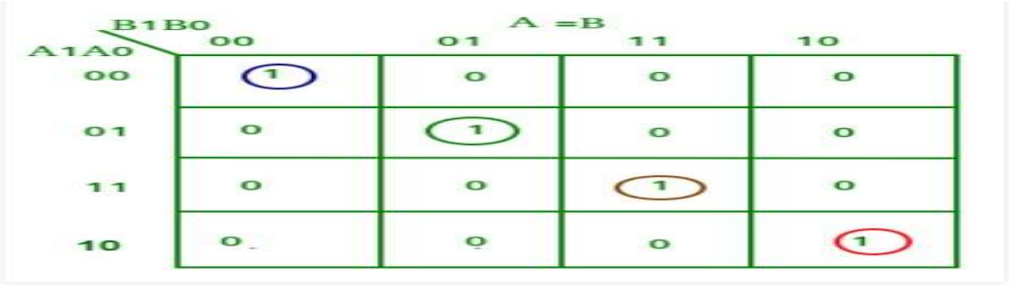
Two-bit magnitude comparator

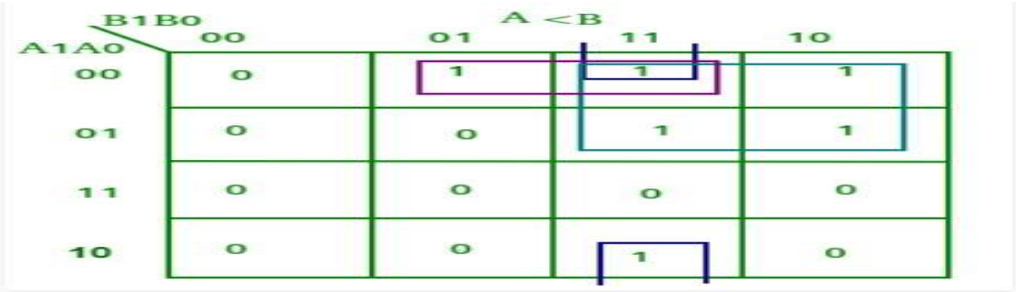
- A comparator used to compare two binary numbers each of two bits is called a 2-bit Magnitude comparator.
- It consists of four inputs and three outputs to generate less than, equal to and greater than between two binary numbers.
- The truth table for a 2-bit comparator is given below:

INPUT				OUTPUT		
A1	A0	B1	B0	A<B	A=B	A>B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

From the above truth table K-map for each output can be drawn as follows:




A =B
B1B0
A1A0
00
01
11
10
00 1 0 0 0
01 0 1 0 0
11 0 0 1 0
10 0 0 0 1


A <B
B1B0
A1A0
00
01
11
10
00 0 1 1 1
01 0 0 1 1
11 0 0 0 0
10 0 0 1 0

From the above K-maps logical expressions for each output can be expressed as follows:

A>B:A1B1’ + A0B1’B0’ + A1A0B0’

A=B: A1’A0’B1’B0’ + A1’A0B1’B0 + A1A0B1B0 + A1A0’B1B0’

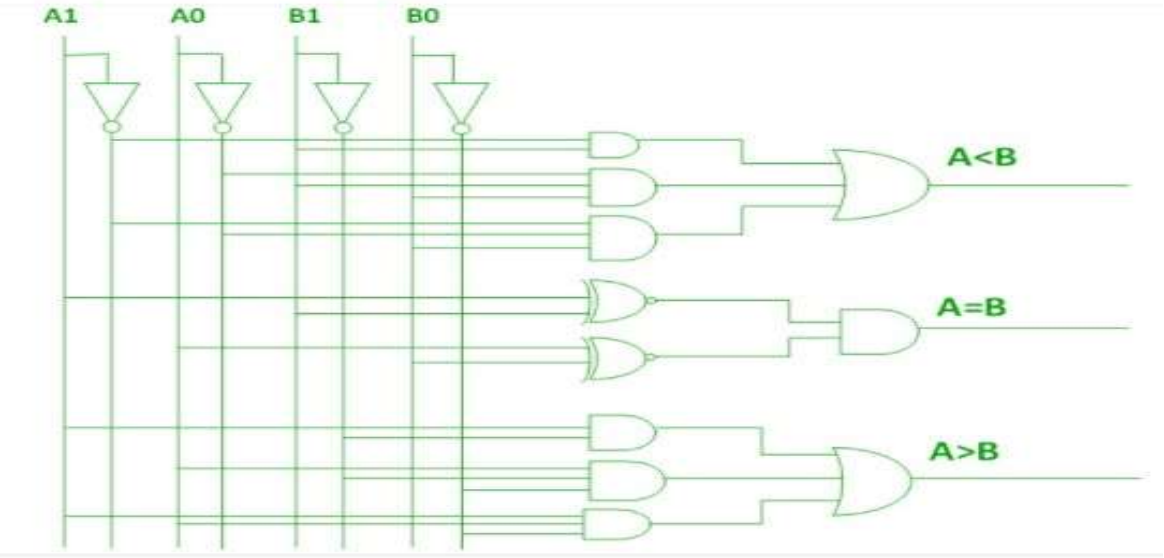
: A1’B1’ (A0’B0’ + A0B0) + A1B1 (A0B0 + A0’B0’)

: (A0B0 + A0’B0’) (A1B1 + A1’B1’)

: (A0 Ex-Nor B0) (A1 Ex-Nor B1)

A<B:A1’B1 + A0’B1B0 + A1’A0’B0

By using these Boolean expressions, we can implement a logic circuit for this comparator as given below:



Possible Short Type Questions with Answers

1. What is combinational circuit? Give an example ?

Ans: A combinational circuit consists of logic gates whose outputs at any time are determined from the present combination of inputs.
Examples of combinational circuits are adder, coder, magnitude comparator etc.

2. What are the universal gates ?
Ans: NAND and NOR are universal gates, because they replace all the other gates in a circuit.

3. What is demux ?
Ans: Demultiplexer is a circuit that receives information on a single line and transmits this information on one of 2n possible output lines. A demultiplexer is a decoder with an enable input.

4. Write the procedural steps for the design of combinational circuits. Ans: The design of combinational circuit starts from a specification of the problem culminates in a logic diagram or set of Boolean equations from which the logic diagram can be obtained. The procedure involves the following steps:

- From the specifications of the circuit, determine the required number of inputs and outputs, and assign a letter symbol to each.
- Derive the truth table that defines the required relation ship between inputs and outputs.
- Obtain the simplified Boolean functions of each output as function of the input variables.
- Draw the logic diagram.

5. What is the difference between combinational and Sequential Logic Circuit ? (w-20)

Ans : <u>Combinationa logic Circuit</u>

- In this output depends only upon present input.
- Speed is fast.
- It is designed easy.
- There is no feedback between input and output.

<u>Sequential Logic Circuit</u>

- In this output depends upon present as well as past input.
- Speed is slow.
- It is designed tough as compared to combinational circuits.
- There exists a feedback path between input and output.

6. Distinguish between a Multiplexer and Demultiplexer . (w-20) Ans : A

multiplexer (Mux) is a combinational circuit that uses several data inputs to generate a single output.
A **demultiplexer (Demux)** is also a combinational circuit that uses single input that can be directed throughout several outputs.

Possible Long Type Questions

1. What is Half Adder ? Design a Full Adder Circuit using Half Adder and OR Gate. (w-20)
2. Explain the function of 1:4 Demux circuit with a neat diagram and write its truth table. (w-20)
3. Design a 2 bit magnitude comparator circuit and explain its operation. (w20)
4. Realize full subtractor circuit and explain its operation with truth table.

CHAPTER 3-
SEQUENTIAL LOGIC CIRCUITS

LEARING OBJECTIVES:

3.1- Give the idea of Sequential Logic Circuits.

3.2- State the necessary of clock and give the concept of level clocking and edge triggering.

3.3- Clocked SR flipflop with preset and clear inputs.
3.4- Construct level clocked JK flipflop using SR flipflop and explain with truth table.
3.5- Concept of Race around condition and study of master slave JK flipflop.
3.6- Give the truth tables of edge triggered D and T flipflop and draw their symbols.
3.7- Applications of flipflops.
3.8- Define modulus of a counter.
3.9- 4 bit asynchronous counter and its timing diagram.
3.10- Asynchronous decade counter.
3.11- 4 bit Asynchronous counter.
3.12- Distinguish between Synchronous and Asynchronous Counter.
3.13- State the need for a Register and lists the four types of Registers.
3.14- Working of SISO, SIPO, PISO, PIPO Register with truth table using flipflop.

3.1-Give the idea of sequential logic circuit

SEQUENTIAL CIRCUIT : -

It is a circuit whose output depends upon the present input, previous output and the sequence in which the inputs are applied.

DIFFERENCE BETWEEN SLC AND CLC :-

- In combinational circuit output depends upon present input at any instant of time and do not use memory.
- Hence previous input does not have any effect on the circuit. But sequential circuit has memory and depends upon present input and previous output.
- Sequential circuits are slower than combinational circuits and these sequential circuits are harder to design.
- The data stored by the memory element at any given instant of time is called the present state of sequential circuit.

3.2-State the necessity of clock and give the concept of level clocking and edge triggering;-

Principle Of F/F Operation, Its Types types:-

Sequential logic circuits (SLC) are classified as

- Synchronous SLC
- Asynchronous SLC

The SLC that are controlled by clock are called synchronous SLC and those which are not controlled by a clock are asynchronous SLC.

Clock:- A recurring pulse is called a clock.

FLIP-FLOP AND LATCH:-

- A flip-flop or latch is a circuit that has two stable states and can be used to store information.
- A flip-flop is a binary storage device capable of storing one bit of information. In a stable state, the output of a flip-flop is either 0 or 1.
- **Latch** is a non-clocked flip-flop and it is the building block for the flip-flop.
- **Storage element that** operate with signal level are called latches and those operate with clock transition are called as flip-flops.

A flip-flop is called so because its output either flips or flops meaning to switch back and forth. ,

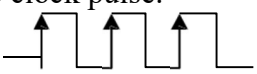
A flip-flop is also called a bi-stable multi-vibrator as it has two stable states. The input signals which command the flip-flop to change state are called excitations.

, Flip-flops are can store 1 or 0.

, Flip-flops using the clock signal are called clocked flip-flops. Control signals are effective only if they are applied in synchronization with the clock signal.

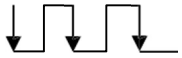
, Clock-signals may be positive-edge triggered or negative-edge triggered.

, Positive-edge triggered flip-flops are those in which state transitions take place only at positive-going edge of the clock pulse.



Negative-edge triggered flip-flops are those in which state transition take place only at negative- going

edge of the clock pulse.

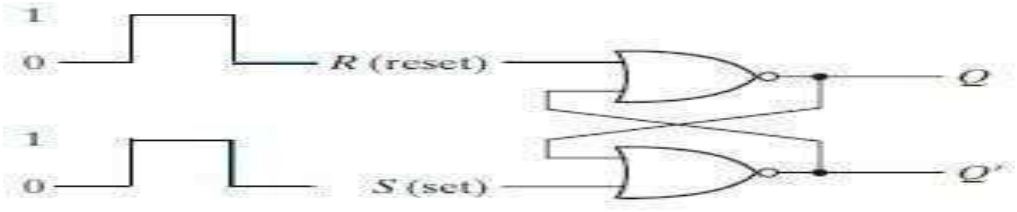


Some common type of flip-flops include

- SR (set-reset) F-F
- D (data or delay) F-F
- T (toggle) F-F and
- JK F-F

3.3-Clocked SR Flip-Flop with preset and clear input SR F/F USING NAND,NOR LATCH (UNCLOCKED)

- The SR latch is a circuit with two cross-coupled NOR gates or two cross-coupled NAND gates.
- It has two outputs labeled Q and Q'. Two inputs are there labeled S for set and R for reset. The latch has two useful states.
- When Q=0 and Q'=1 the condition is called reset state and when Q=1 and Q'=0 the condition is called set state.
- Normally Q and Q' are complement of each other.
- The figure represents a SR latch with two cross-coupled NOR gates.
- We know if any one of the input for a NOR gate is HIGH then its output will be LOW and if both the inputs are LOW then only the output will be HIGH.



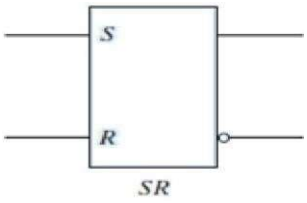
,

- The first condition (S = 1, R = 0) is the action that must be taken by input S to bring the circuit to the set state. Removing the active input from S leaves the circuit in the same state.
- After both inputs return to 0, it is then possible to shift to the reset state by momentary applying a 1 to the R input. The 1 can then be removed from R, whereupon the circuit remains in the reset state.

- When both inputs S and R are equal to 0, the latch can be in either the set or the reset state, depending on which input was most recently a 1.
- If a 1 is applied to both the S and R inputs of the latch, both outputs go to 0.
- This action produces an undefined next state, because the state that results from the input transitions depends on the order in which they return to 0.
- It also violates the requirement that outputs be the complement of each other.
- In normal operation, this condition is avoided by making sure that 1's are not applied to both inputs simultaneously.

Truth table for SR latch designed with NOR gates is shown below.

Input		Output				Comment
S	R	Q	Q'	QNext	Q'Next	
0	0	0	1	0	1	No change
0	0	1	0	1	0	
0	1	0	1	0	1	Reset
0	1	1	0	0	1	
1	0	0	1	1	0	Set
1	0	1	0	1	0	
1	1	0	1	X	X	Prohibited
1	1	1	0	X	X	state



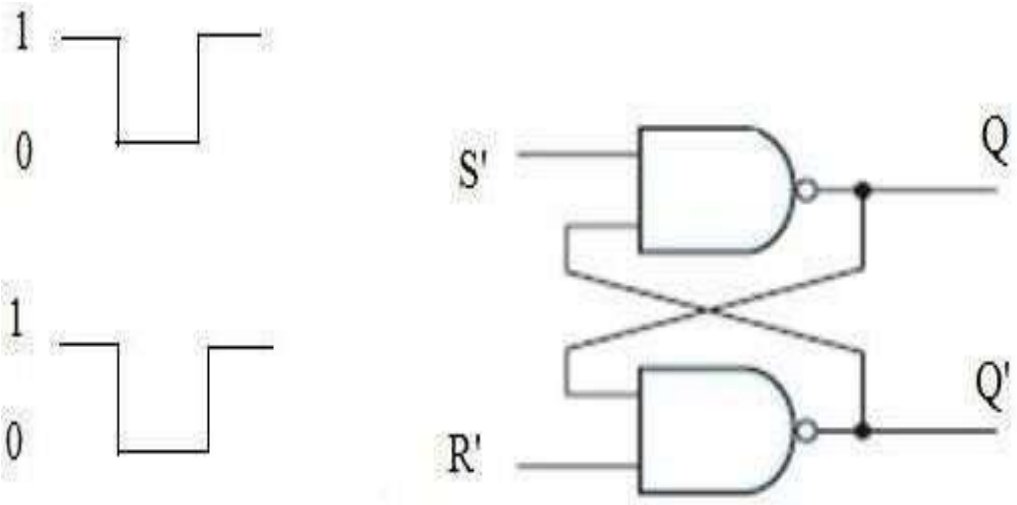
Symbol for SR NOR Latch

Racing Condition:-

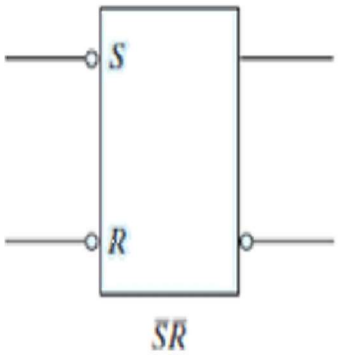
In case of a SR latch when S=R=1 input is given both the output will try to become 0. This is called Racing condition.

SR latch using NAND gate:-

- The below figure represents a SR latch with two cross-coupled NAND gates.
- The circuit has NAND gates and as we know if any one of the input for a NAND gate is LOW then its output will be HIGH and if both the inputs are HIGH then only the output will be LOW.
- It operates with both inputs normally at 1, unless the state of the latch has to be changed. The application of 0 to the S input causes output Q to go to 1, putting the latch in the set state.
- When the S input goes back to 1, the circuit remains in the set state. After both inputs go back to 1, we are allowed to change the state of the latch by placing a 0 in the R input.
- This action causes the circuit to go to the reset state and stay there even after both inputs return to 1.



- The condition that is forbidden for the NAND latch is both inputs being equal to 0 at the same time, an input combination that should be avoided.
- In comparing the NAND with the NOR latch, note that the input signals for the NAND require the complement of those values used for the NOR latch.
- Because the NAND latch requires a 0 signal to change its state, it is sometimes referred to as an S'R' latch.
- The primes (or, sometimes, bars over the letters) designate the fact that the inputs must be in their complement form to activate the circuit.



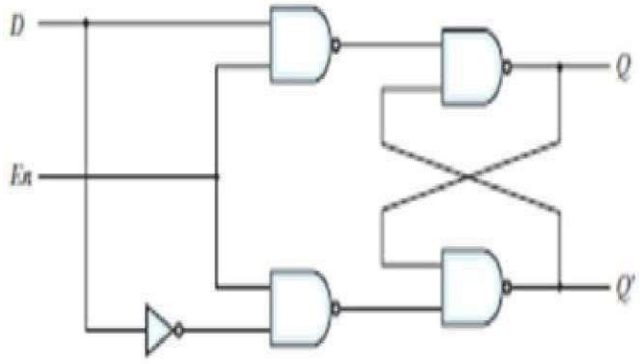
- The above represents the symbol for inverted SR latch or SR latch using NAND gate. Truth table for SR latch using NAND gate or Inverted SR latch

S	R	Q_{next}	Q'_{next}
0	0	Race	Race
0	1	0	1 (Reset)
1	0	1	0 (Set)
1	1	Q (No change)	Q' (No change)

3.4-Construct level clocked JK Flip-Flop using SR Flip-Flop and explain with truth table

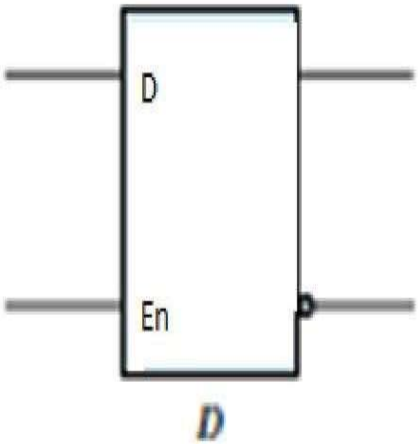
Clocked SR, D, JK, T, MSJK Flip-Flop Symbol, Logic Circuit, Truth Table, and Applications

One way to eliminate the undesirable condition of the indeterminate state in the SR latch is to ensure that inputs S and R are never equal to 1 at the same time.



(a) Logic diagram

- This is done in the D latch. This latch has only two inputs: D (data) and En (enable).
- The D input goes directly to the S input, and its complement is applied to the R input.



(Symbol for D-Latch)

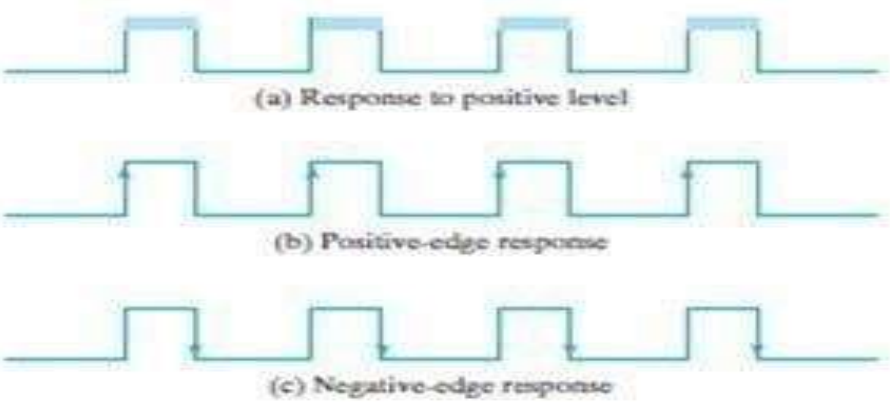
- As long as the enable input is at 0, the cross-coupled SR latch has both inputs at the 1 level and the circuit can't change state regardless of the value of D.
- The below represents the truth table for the D-latch.

En	D	Next State of Q
0	X	No change
1	0	Q=0;Reset State
1	1	Q=1;Set State

The D input is sampled when En = 1. If D = 1, the Q output goes to 1, placing the circuit in the set state. If D = 0, output Q goes to 0, placing the circuit in the reset state. This situation provides a path from input D to the output, and for this reason, the circuit is often called a TRANSPARENT latch.

TRIGGERING METHODS:-

- The state of a latch or flip-flop is switched by a change in the control input. This momentary change is called a trigger, and the transition it causes is said to trigger the flip-flop.
- Flip-flop circuits are constructed in such a way as to make them operate properly when they are part of a sequential circuit that employs a common clock.
- The problem with the latch is that it responds to a change in the level of a clock pulse. For proper operation of a flip-flop it should be triggered only during a signal transition.
- This can be accomplished by eliminating the feedback path that is inherent in the operation of the sequential circuit using latches. A clock pulse goes through two transitions: from 0 to 1 and the return from 1 to 0.
- A ways that a latch can be modified to form a flip-flop is to produce a flip-flop that triggers only during a signal transition (from 0 to 1 or from 1 to 0) of the synchronizing signal (clock) and is disabled during the rest of the clock pulse.

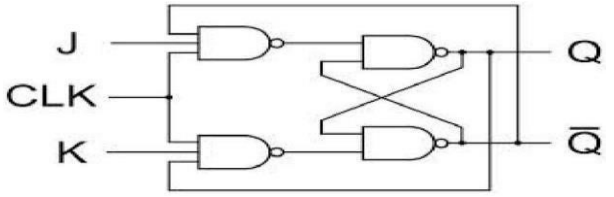


3.5-Concept of race around condition and study of master slave JK Flip-Flop

JK FLIP-FLOP:-

- The JK flip-flop can be constructed by using basic SR latch and a clock. In this case the outputs Q and Q' are returned back and connected to the inputs of NAND gates.
- This simple JK flip Flop is the most widely used of all the flip-flop designs and is considered to be a universal flip-flop circuit.
- The sequential operation of the JK flip flop is exactly the same as for the previous SR flip-flop with the same "Set" and "Reset" inputs.
- The difference this time is that the "JK flip flop" has no invalid or forbidden input states of the SR Latch even when S and R are both at logic "1".

(The below diagram shows the circuit diagram of a JK flip-flop)



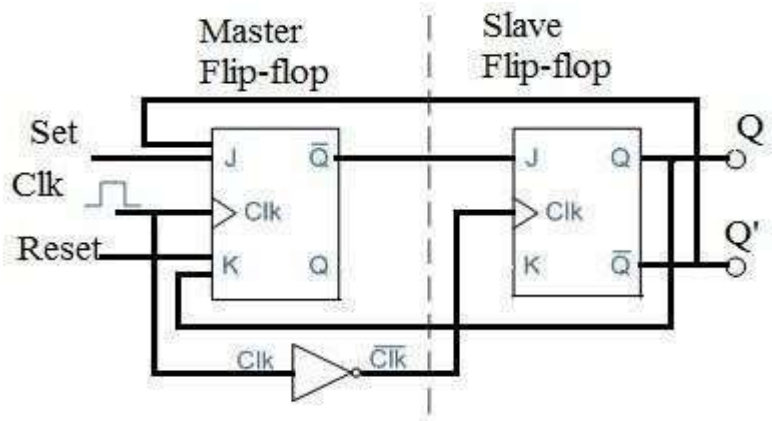
- The JK flip flop is basically a gated SR Flip-flop with the addition of a clock input circuitry that prevents the illegal or invalid output condition that can occur when both inputs S and R are equal to logic level “1”.

- Due to this additional clocked input, a JK flip-flop has four possible input combinations, “logic 1”, “logic 0”, “no change” and “toggle”.

MASTER-SLAVE JK FLIP-FLOP:-

- The Master-Slave Flip-Flop is basically two gated SR flip-flops connected together in a series configuration with the slave having an inverted clock pulse.
- The outputs from Q and Q’ from the “Slave” flip-flop are fed back to the inputs of the “Master” with the outputs of the “Master” flip flop being connected to the two inputs of the “Slave” flip flop.
- This feedback configuration from the slave’s output to the master’s input gives the characteristic toggle of the JK flip flop as shown below.

- The Master-Slave JK Flip Flop

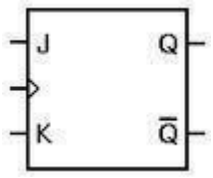


- The input signals J and K are connected to the gated “master” SR flip flop which “locks” the input condition while the clock (Clk) input is “HIGH” at logic level “1”.
- As the clock input of the “slave” flip flop is the inverse (complement) of the “master” clock input, the “slave” SR flip flop does not toggle.
- The outputs from the “master” flip flop are only “seen” by the gated “slave” flip flop when the clock input goes “LOW” to logic level “0”.
- When the clock is “LOW”, the outputs from the “master” flip flop are latched and any additional changes to its inputs are ignored.
- The gated “slave” flip flop now responds to the state of its inputs passed over by the “master” section.
- Then on the “Low-to-High” transition of the clock pulse the inputs of the “master” flip flop are fed through to the gated inputs of the “slave” flip flop and on the “High-to-Low” transition the same inputs are reflected on the output of the “slave” making this type of flip flop edge or pulse-triggered.
- Then, the circuit accepts input data when the clock signal is “HIGH”, and passes the data to the output on the falling-edge of the clock signal.
- In other words, the Master-Slave JK Flip flop is a “Synchronous” device as it only passes data with the timing of the clock signal.

RACING CONDITION ;-

In JK F/F when J=K=1, and clock =1 for a longer period of time, then Q output will toggle as long as CLK=1 HIGH, which makes the output of the f/f unstableor uncertain. This problem is called race around condition. In can be avoided by using MSJK F/F.

The symbol for a JK flip flop is similar to that of an SR bistable latch except the clock input.



(The above diagram shows the symbol of a JK flip-flop.)

- Both the S and the R inputs of the SR bi-stable have now been replaced by two inputs called the J and K inputs, respectively after its inventor Jack and Kilby. Then this equates to: J = S and K = R.
- The two 2-input NAND gates of the gated SR bi-stable have now been replaced by two 3-input NAND gates with the third input of each gate connected to the outputs at Q and Q'.
- This cross coupling of the SR flip-flop allows the previously invalid condition of S = "1" and R = "1" state to be used to produce a "toggle action" as the two inputs are now interlocked.
- If the circuit is now "SET" the J input is inhibited by the "0" status of Q' through the lower NAND gate. If the circuit is "RESET" the K input is inhibited by the "0" status of Q through the upper NAND gate. As Q and Q' are always different we can use them to control the input.

Truth table for JK flip- flop

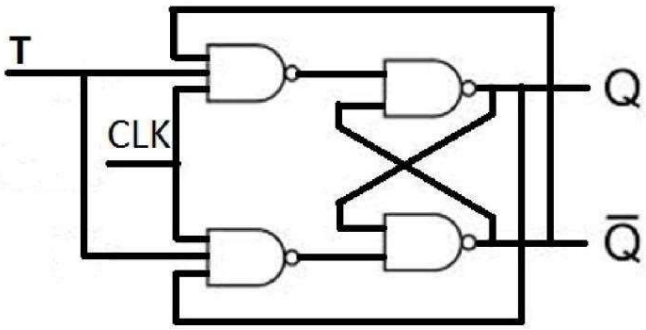
Inp		Outp		Comment
J	K	Q	Qn	
0	0	0	0	No change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Toggle
1	1	1	0	

When both inputs J and K are equal to logic "1", the JK flip flop toggles.

3.6-Give the truth table of edge trigger D and T Flip-Flop and draw their symblos

T FLIP-FLOP:-

- Toggle flip-flop or commonly known as T flip-flop.
- This flip-flop has the similar operation as that of the JK flip-flop with both the inputs J and K are shorted i.e. both are given the common input.



. Hence its truth table is same as that of JK flip-flop when J=K= 0 and J=K=1.So its truth table is as follows.

T	Q	$Q_{n\ ext}$	Comment
0	0	0	No change
	1	1	
1	0	1	Toggles
	1	0	

CHARACTERISTIC TABLE:-

- A characteristic table defines the logical properties of a flip-flop by describing its operation in tabular form.
- The next state is defined as a function of the inputs and the present state.
- Q (t) refers to the present state and Q (t + 1) is the next.
- Thus, Q (t) denotes the state of the flip-flop immediately before the clock edge, and Q(t + 1) denotes the state that results from the clock transition.
- The characteristic table for the JK flip-flop shows that the next state is equal to the present state when inputs J and K are both equal to 0. This condition can be expressed as Q (t + 1) = Q (t), indicating that the clock produces no change of state. **Characteristic Table Of JK Flip-Flop**

J	K	Q(t+1)
0	0	Q(t) No change
0	1	0 Reset
1	0	1 Set
1	1	Q'(t) Complement

- When K = 1 and J = 0, the clock resets the flip-flop and Q(t + 1) = 0. With J = 1 and K = 0, the flip-flop sets and Q(t + 1) = 1. When both J and K are equal to 1, the next state changes to the complement of the present state, a transition that can be expressed as Q(t + 1) = Q'(t). The characteristic equation for JK flip-flop is represented as Q(t+1)= JQ' + K'Q

Characteristic Table of D Flip-Flop

D	Q(t+1)
0	0
1	1

The next state of a D flip-flop is dependent only on the D input and is independent of the present state. This can be expressed as Q (t + 1) = D. It means that the next-state value is equal to the value of D. Note that the D flip-flop does not have a “no-change” condition and its characteristic equation is written as Q(t+1)=D.

Characteristic Table of T Flip-Flop

T	Q(t+1)
0	Q(t) No change
1	Q’(t) Complement

 The characteristic table of T flip-flop has only two conditions: When T = 0, the clock edge does not

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

change the state; when T = 1, the clock edge complements the state of the flip-flop and the eqn is

3.7-Application of Flip-Flop

Registers

- Registers are the devices which are meant to store the data. As known, each flip-flop can store a single-bit of information.
- This means that by cascading n flip-flops, one can store n bits of information. Such an arrangement is called an n-bit register.
- For example by cascading three D flip-flops as shown in Figure 1, one can store three bits of information (B3, B2 and B1), thus forming a 3-bit buffer register.
- The data stored in the registers can be moved stage-wise within the registers and/or in/out of the register by applying clock pulses. Such a register is called shift register.
- There are various kinds of shift registers depending on the mode of data-shift viz., serial-in serial-out register, serial-in parallel-out register, parallel-in serial-out register, parallel-in parallel- out register.
- Further depending on the direction of data movement they can be either left-shift and/or right-shift in nature

3.8- Define modulus of a counter

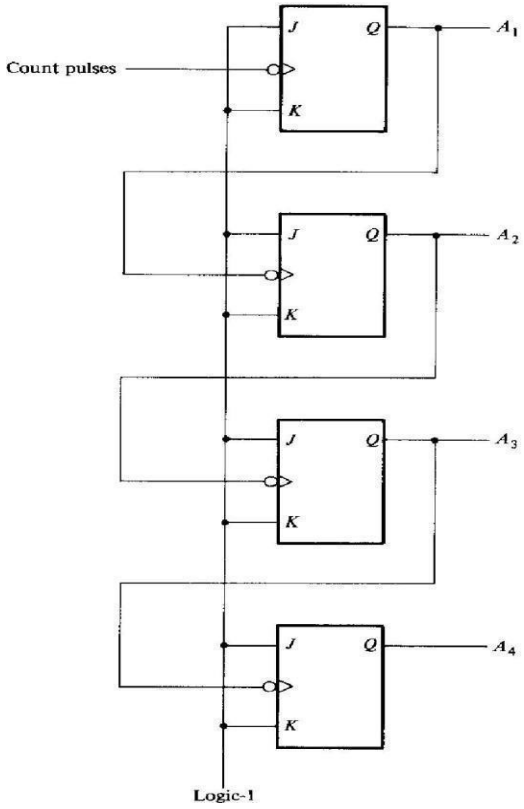
Modulus Counter

- A modulus counter is that which produces an output pulse after a certain number of input pulses is applied.
- In modulus counter the total count possible is based on the number of stages, i.e., digit positions

- Modulus counters are used in digital computers.
- A binary modulo-8 counter with three flip-flops, i.e., three stages, will produce an output pulse, i.e., display an output one-digit, after eight input pulses have been counted, i.e., entered or applied.
- This assumes that the counter started in the zero-condition.

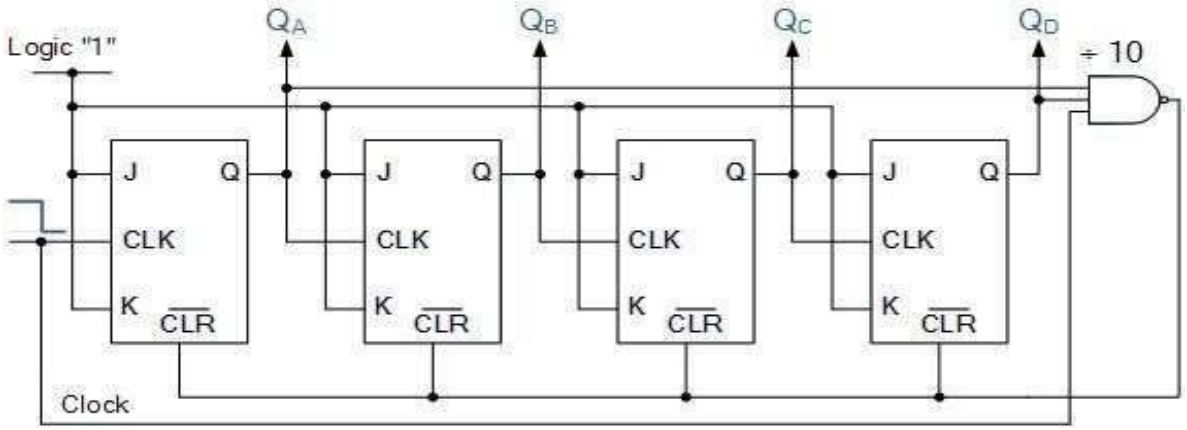
3.9- Four bit Asynchronous Counter and its timing diagram Asynchronous Counter

- An asynchronous (ripple) counter is a single d-type flip-flop, with its J (data) input fed from its own inverted output.
- This circuit can store one bit, and hence can count from zero to one before it overflows (starts over from 0).



- This counter will increment once for every clock cycle and takes two clock cycles to overflow, so every cycle it will alternate between a transition from 0 to 1 and a transition from 1 to 0.
- This creates a new clock with a 50% duty cycle at exactly half the frequency of the input clock.
- Additional flip-flops can be added, by always inverting the output to its own input, and using the output from the previous flip-flop as the clock signal. The result is called a ripple counter, which can count to $2^n - 1$, where n is the number of bits (flip-flop stages) in the counter.
- Ripple counters suffer from unstable outputs as the over flows "ripple" from stage to stage, but they find application as dividers for clock signals.

3.10- Asynchronous Decade Counter Asynchronous Decade Counter



A decade counter can count from BCD “0” to BCD “9”.

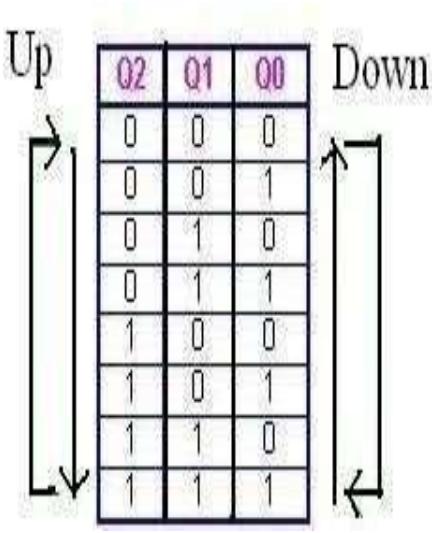
- A decade counter requires resetting to zero when the output count reaches the decimal value of 10, ie. when DCBA = 1010 and this condition is fed back to the reset input.
- A counter with a count sequence from binary “0000” (BCD = “0”) through to “1001” (BCD = “9”) is generally referred to as a BCD binary-coded-decimal counter because its ten state sequence is that of a BCD code but binary decade counters are more common.
- This type of asynchronous counter counts upwards on each leading edge of the input clock signal starting from 0000 until it reaches an output 1001 (decimal 9).
- Both outputs QA and QD are now equal to logic “1” and the output from the NAND gate changes state from logic “1” to a logic “0” level and whose output is also connected to the CLEAR (CLR) inputs of all the J-K Flip-flops.
- This signal causes all of the Q outputs to be reset back to binary 0000 on the count of 10. Once QA and QD are both equal to logic “0” the output of the NAND gate returns back to a logic level “1” and the counter restarts again from 0000. We now have a decade or Modulo-10 counter.

Decade Counter Truth Table

Clock Count	Output bit Pattern				Decimal Value
	QD	QC	QB	QA	
1	0	0	0	0	0
2	0	0	0	1	1
3	0	0	1	0	2
4	0	0	1	1	3
5	0	1	0	0	4
6	0	1	0	1	5
7	0	1	1	0	6
8	0	1	1	1	7
9	1	0	0	0	8
10	1	0	0	1	9
11	Counter Resets its Outputs back to Zero				

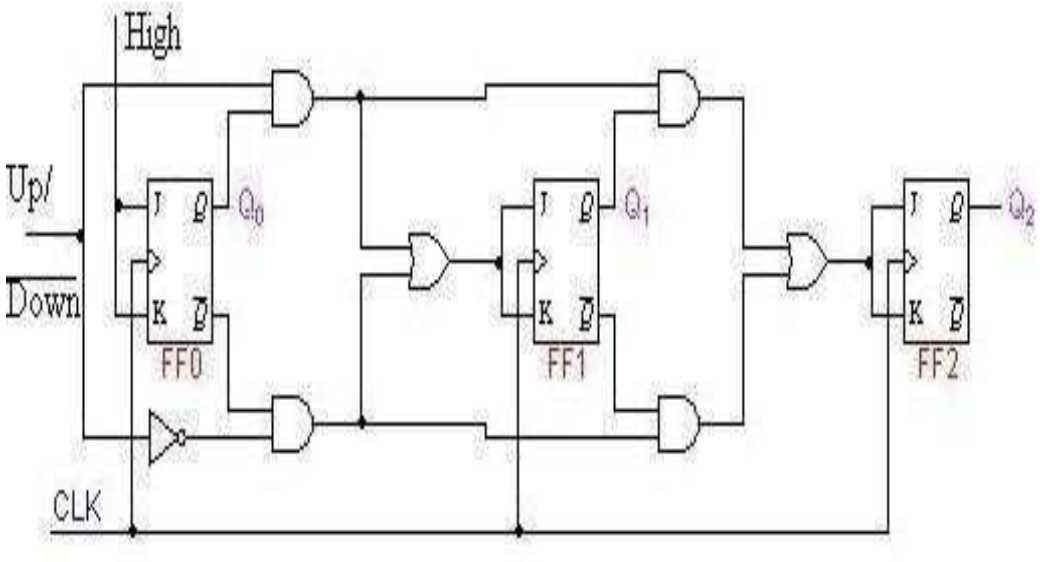
Up/Down Counter

- In a synchronous up-down binary counter the flip-flop in the lowest-order position is complemented with every pulse.
- A flip-flop in any other position is complemented with a pulse, provided all the lower-order pulse equal to 0.
- Up/Down counter is used to control the direction of the counter through a certain sequence.



From the sequence table we can observe that:

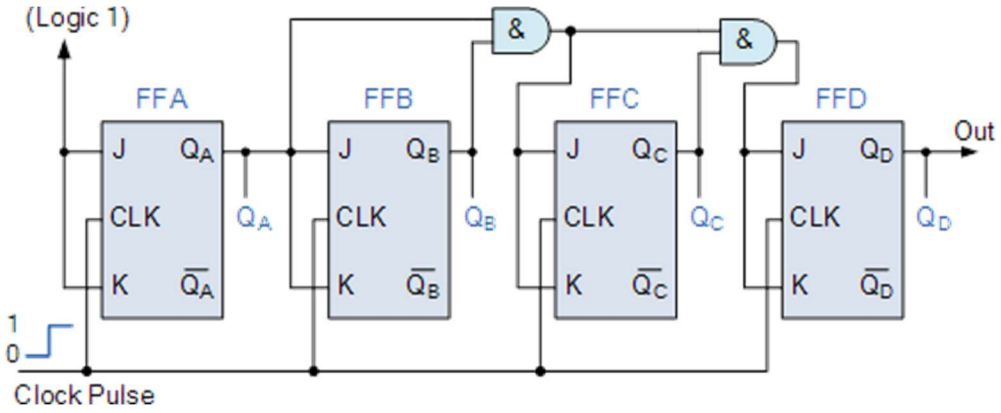
- For both the UP and DOWN sequences, Q0 toggles on each clock pulse.
- For the UP sequence, Q1 changes state on the next clock pulse when Q0=1.
- For the DOWN sequence, Q1 changes state on the next clock pulse when Q0=0.
- For the UP sequence, Q2 changes state on the next clock pulse when Q0=Q1=1.
- For the DOWN sequence, Q2 changes state on the next clock pulse when Q0=Q1=0.



- These characteristics are implemented with the AND, OR & NOT logic connected as shown in the logic diagram above.

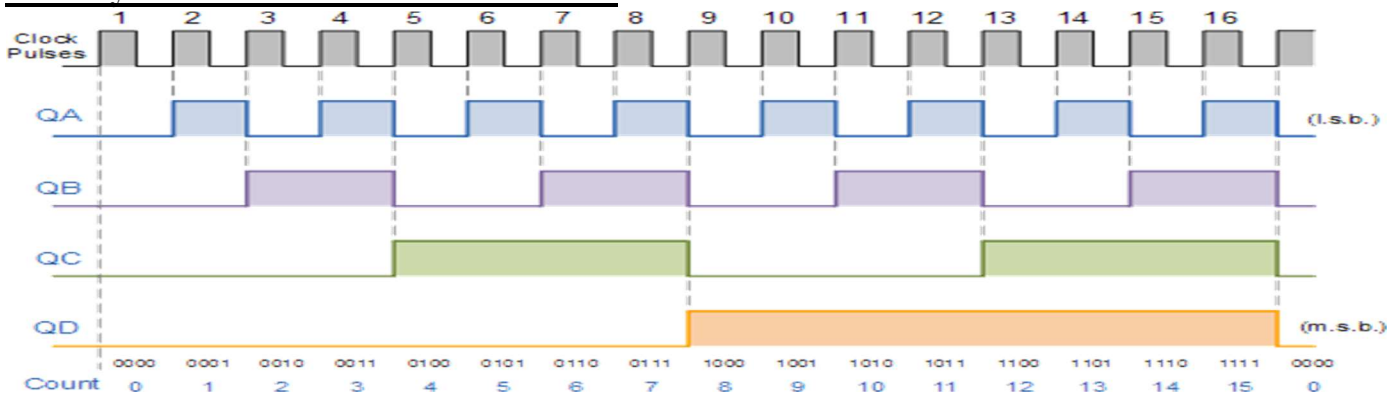
3.11-Four bit synchronous counter

Binary 4-bit Synchronous Up Counter



- It can be seen above, that the external clock pulses (pulses to be counted) are fed directly to each of the J-K flip-flops in the counter chain and that both the J and K inputs are all tied together in toggle mode, but only in the first flip-flop, flip- flop FFA (LSB) are they connected HIGH, logic "1" allowing the flip-flop to toggle on every clock pulse. Then the synchronous counter follows a predetermined sequence of states in response to the common clock signal, advancing one state for each pulse.
- The J and K inputs of flip-flop FFB are connected directly to the output QA of flip-flop FFA, but the J and K inputs of flip- flops FFC and FFD are driven from separate AND gates which are also supplied with signals from the input and output of the previous stage. These additional AND gates generate the required logic for the JK inputs of the next stage.
- If we enable each JK flip-flop to toggle based on whether or not all preceding flip-flop outputs (Q) are "HIGH" we can obtain the same counting sequence as with the asynchronous circuit but without the ripple effect, since each flip-flop in this circuit will be clocked at exactly the same time.
- Then as there is no inherent propagation delay in synchronous counters, because all the counter stages are triggered in parallel at the same time, the maximum operating frequency of this type of frequency counter is much higher than that for a similar asynchronous counter circuit.

4-bit Synchronous Counter Waveform



[Timing Diagram]

3.12-Distinguish between synchronous and asynchronous counters

S.NO	Synchronous Counter	Asynchronous Counter
1.	In synchronous counter, all flip flops are triggered with same clock simultaneously.	In asynchronous counter, different flip flops are triggered with different clock, not simultaneously.
2.	Synchronous Counter is faster than asynchronous counter in operation.	Asynchronous Counter is slower than synchronous counter in operation.
3.	Synchronous Counter does not produce any decoding errors.	Asynchronous Counter produces decoding error.
4.	Synchronous Counter is also called Parallel Counter.	Asynchronous Counter is also called Serial Counter.
5.	Synchronous Counter designing as well implementation are complex due to increasing the number of states.	Asynchronous Counter designing as well as implementation is very easy.
6.	Synchronous Counter will operate in any desired count sequence.	Asynchronous Counter will operate only in fixed count sequence (UP/DOWN).
7.	Synchronous Counter examples are: Ring counter, Johnson counter.	Asynchronous Counter examples are: Ripple UP counter, Ripple DOWN counter.

.

3.13-State the need for a register and list the four type of register

- Registers are used for storage and transfer of binary information in a digital system.
- A register is mostly used for the purpose of storing and shifting binary data entered into it from an external source and has no characteristics internal sequence of states.
- The storage capacity of a register is defined as the number of bits of digital data, it can store or retain. These registers are normally used for temporary storage of data.

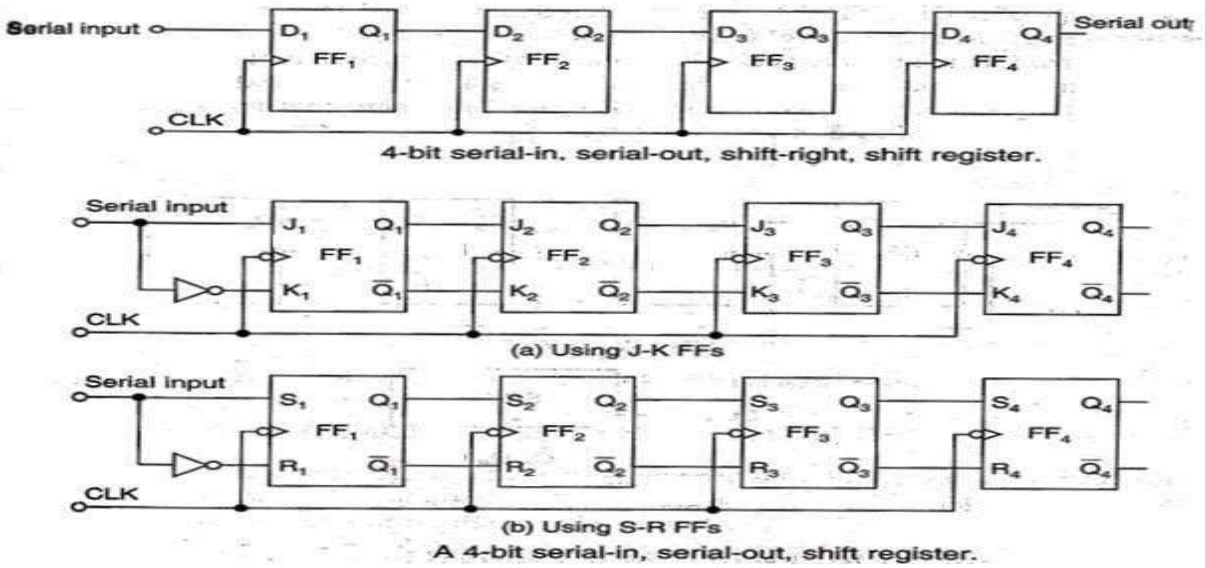
CONTROLLED BUFFER REGISTER :-

- A number of FFs connected together such that data may be shifted into and shifted out of them is called a shift register.
- Data may be shifted into or out of the register either in serial form or in parallel form. There are four basic types of shift registers
- Serial in, serial out
- Serial in, parallel out
- Parallel in, serial out Parallel in , parallel out

3.14-Working of SISO, SIPO, PISO, PIPO register with truth table using Flip-Flop

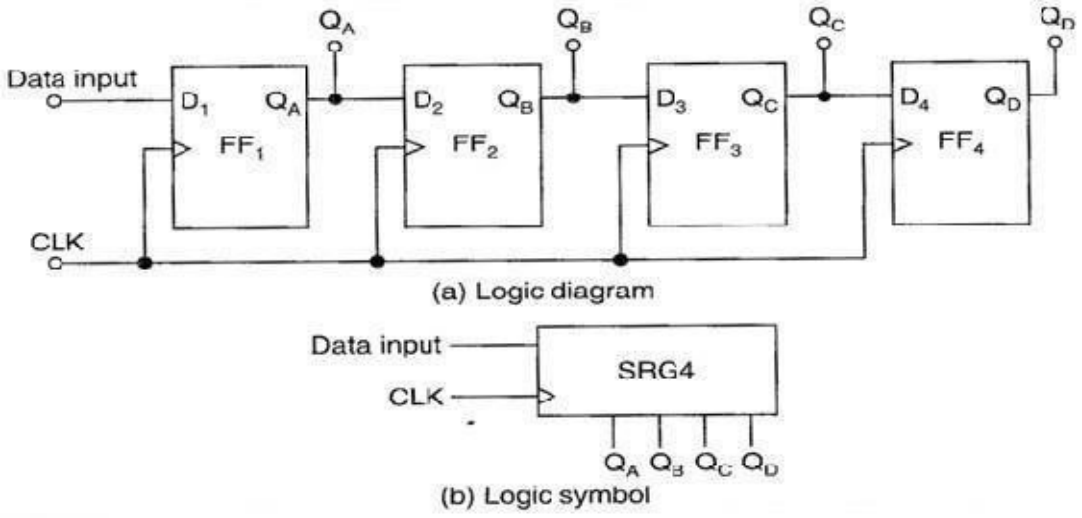
SERIAL IN,SERIAL OUTSHIFT REGISTER:-

- This type of shift register accepts data serially, i.e., one bit at a time and also outputs data serially.
- In 4 stages i.e. with 4 FFs, the register can store up to 4 bits of data.
- Serial data is applied at the D input of the first FF. The Q output of the first FF is connected to the D input of the second FF, the output of the second FF is connected to the D input of the third FF and the Q output of the third FF is connected to the D input of the fourth FF. The data is outputted from the Q terminal of the last FF.
- When a serial data is transferred to a register, each new bit is clocked into the first FF at the positive going edge of each clock pulse.
- The bit that is previously stored by the first FF is transferred to the second FF.
- The bit that is stored by the second FF is transferred to the third FF, and so on.
- The bit that was stored by the last FF is shifted out.
- A shift register can also be constructed using



SERIAL IN PARALLEL OUT SHIFT REGISTER :-

- In this type of register, the data bits are entered into the register serially, but the data stored in the register serially, but the stored in the register is shifted out in the parallel form.
- When the data bits are stored once, each bits appears on its respective output line and all bits are available simultaneously, rather than bit – by – bit basis as in the serial output.
- The serial in, parallel out shift register can be used as a serial in, serial out shift register if the output is taken from the Q terminal of the last FF.



- The logic diagram and logic symbol of a 4 bit serial in, parallel out shift register is given below.

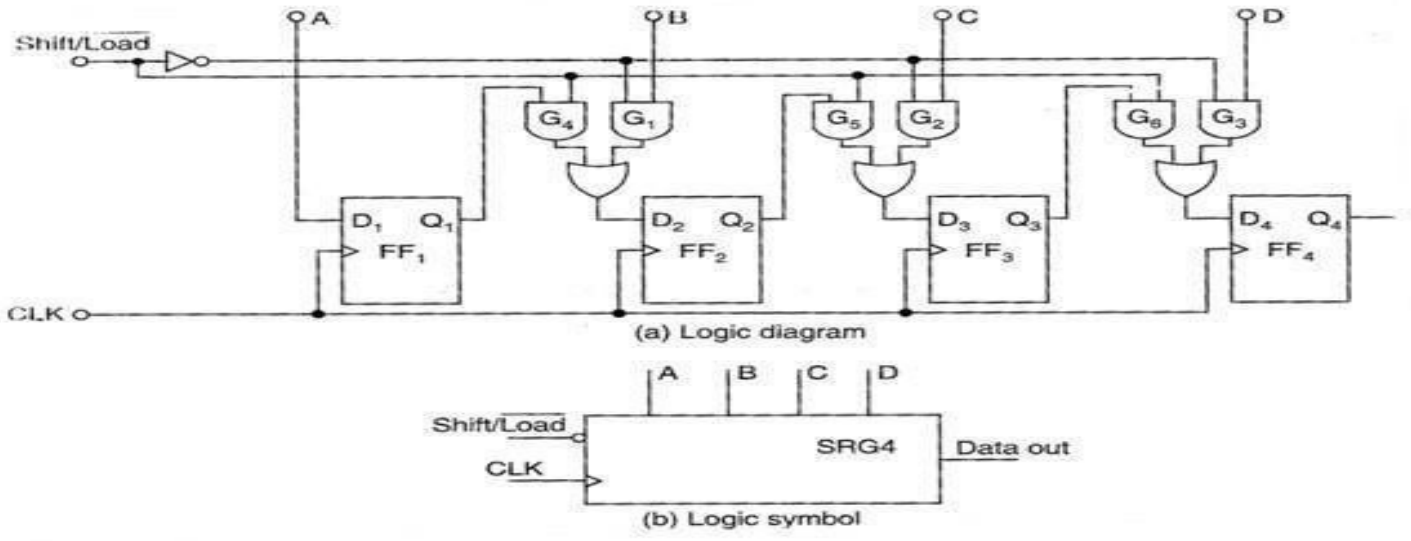
[A 4- bit serial in, parallel out shift register]

PARALLEL IN SERIAL OUT SHIFT REGISTER:-

- For parallel in, serial out shift register the data bits are entered simultaneously into their respective stages on parallel lines, but the data bits are transferred out of the register serially, i.e., on a bit by bit basis over a single line.
- The logic diagram and logic symbol of 4 bit parallel in, serial out shift register using D FFs is shown below.
- There are four data lines A, B, C and D through which the data is entered into the register in parallel form.
- The signal Shift /LOAD allows
- The data to be entered in parallel form into the register and

The data to be shifted out serially from terminal Q4.

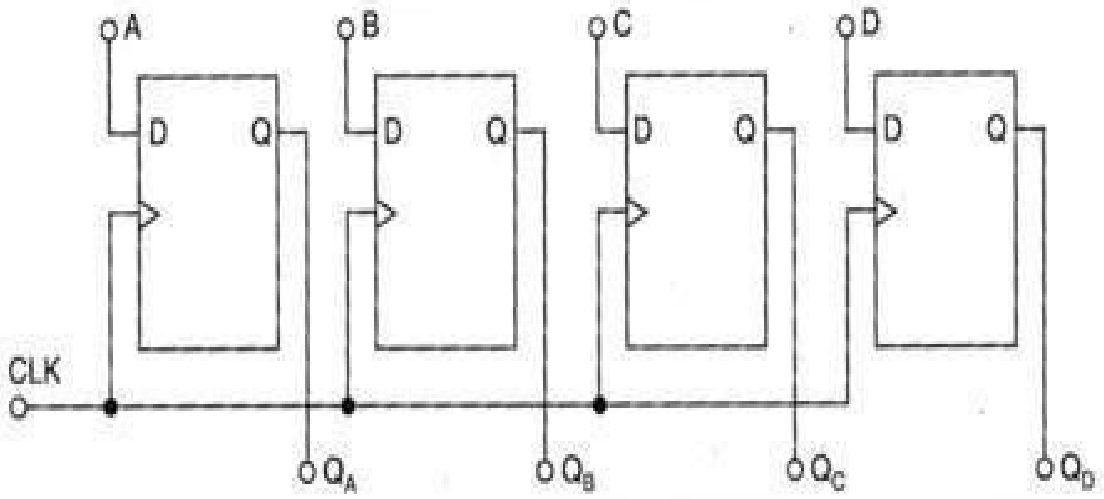
- When Shift /LOAD line is HIGH, gates G1, G2, and G3 are disabled, but gates G4, G5 and G6 are enabled allowing the data bits to shift right from one stage to next.
- When Shift /LOAD line is LOW, gates G4, G5 and G6 are disabled, whereas gates G1, G2 and G3 are enabled allowing the data input to appear at the D inputs of the respective FFs.
- When clock pulse is applied, these data bits are shifted to the Q output terminals of the FFs and therefore the data is inputted in one step.
- The OR gate allows either the normal shifting operation or the parallel data entry depending on which AND gates are enabled by the level on the Shift /LOAD input.



[A 4- bit parallel in, serial out shift register]

PARALLEL IN PARALLEL OUT SHIFT REGISTER:-

- In a parallel in, parallel out shift register, the data entered into the register in parallel form and also the data taken out of the register in parallel form. Immediately following the simultaneous entry of all data bits appear on the parallel outputs.
- The figure shown below is a 4 bit parallel in parallel out shift register using D FFs.
- Data applied to the D input terminals of the FFs.
- When a clock pulse is applied at the positive edge of that pulse, the D inputs are shifted into the Q outputs of the FFs.
- The register now stores the data.
- The stored data is available instantaneously for shifting out in parallel form .



[Logic diagram of a 4 – bit parallel in, parallel out shift register]

Possible Short Type Questions with Answers

1. What is Flip Flop ?

Ans- A flip flop is an electronic circuit with two stable states that can be used to store binary data. The stored data can be changed by applying varying inputs. Flip-flops and latches are fundamental building blocks of digital electronics systems used in computers, communications, and many other types of systems

2. Define modulus of a counter .(w-20)

Ans- The modulus of a counter is the number of states in its count sequence. The maximum possible modulus is determined by the number of flip-flops. For example, a four-bit counter can have a modulus of up to 16 (2^4).

3. List the types of shift register .

Ans- Basic shift registers are classified by structure according to the following types:

Serial-in/serial-out. Parallel-in/serial-out. Serial-in/parallel-out.
Universal parallel-in/parallel-out. Ring counter

4. Define race around condition. (w-20)

Ans- 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 output Q will toggle as long as CLK remains high which makes the output unstable or uncertain. This is called a race around condition in J-K flip-flop

5. Define SR Flipflop.

Ans- SR flip-flop is a gated set-reset flip-flop. The S and R inputs control the state of the flip-flop when the clock pulse goes from LOW to HIGH. The flip-flop will not change until the clock pulse is on a rising edge. When both S and R are simultaneously HIGH, it is uncertain whether the outputs will be HIGH or LOW.

Possible Long Type Questions

1-What is race around condition? Explain Master slave JK FF.
2-Explain working of 4 bit ripple counter and draw its timing diagram. **(w-20)**
3-Explain the working of SIPO and TIPO register with the help of suitable logic diagram.
4-Design MOD-8 counter with neat circuit diagram.
5.Draw the diagram of D-FF and explain the working with function table.
6.Explain the working of JK flipflop with the truth Table. **(w-20)**

CHAPTER-4

8085 Microprocessor

LEARING OBJECTIVES:

4.1-Introduction to Microprocessors, Microcomputers.

4.2-Architecture of Intel 8085a Microprocessor and description of each block.

4.3-Pin Diagram and description.

4.4-Stack, Stack Pointer, Stack top.

4.5-Interrupts.

4.6-Opcode and Operand.

4.7-Differentiate between one byte, two byte & three-byte instruction with example.

4.8-Instruction set of 8085 with example.

4.9-addressing mode of 8085.

5.10-Fetch cycle, Machine cycle, Instruction cycle, T-state.

5.11-Timing diagram for Memory read, Memory write, I/O read, I/O write Machine cycle.

5.12-Timing Diagram for 8085 Instruction.

4.13-Counter and time delay.

4.14-Simple assembly language programming of 8085 4.1-Introduction to Microprocessors, Microcomputers.

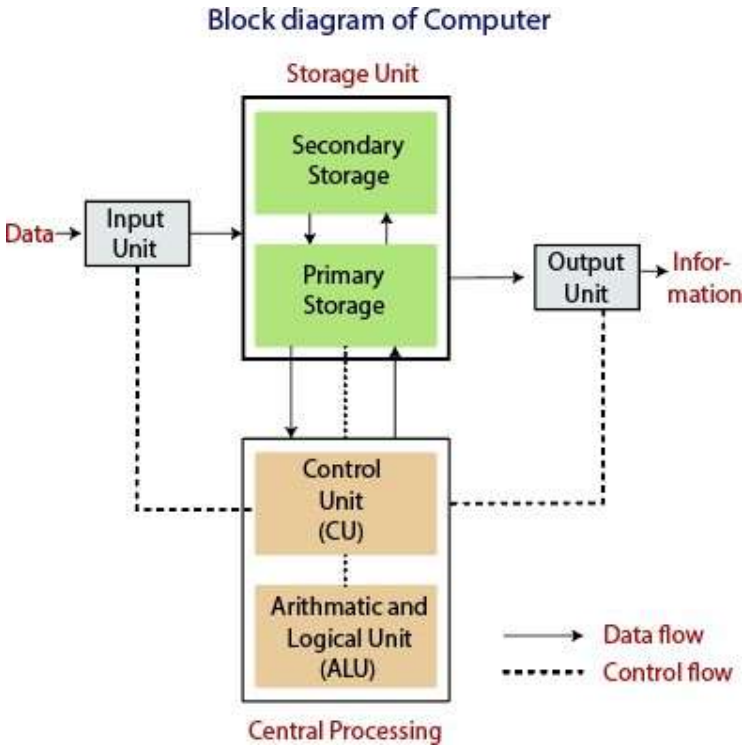
Introduction:

- A microprocessor is a programmable electronics chip that has computing and decision-making capabilities similar to central processing unit of a computer.
- Any microprocessor- based systems having limited number of resources are called microcomputers.
- Nowadays, microprocessor can be seen in almost all types of electronics devices like mobile phones, printers, washing machines etc. Microprocessors are also used in advanced applications like satellites and flights.

Microcomputer:

- Microcomputer is an Electronics Device
- A computer system generally consists of the following units

1-Input device



2-Output device
3-CPU
4- Memory Unit

- Block diagram of Digital computer.

- Input unite consists of input devices like keyboard mouse etc.
- Output unite consists of output device like Printer Monitor etc.
- Control unit(CPU) Control all the action of computer which consists of memory unit, Arithmetic & logic unit.

Microprocessor:

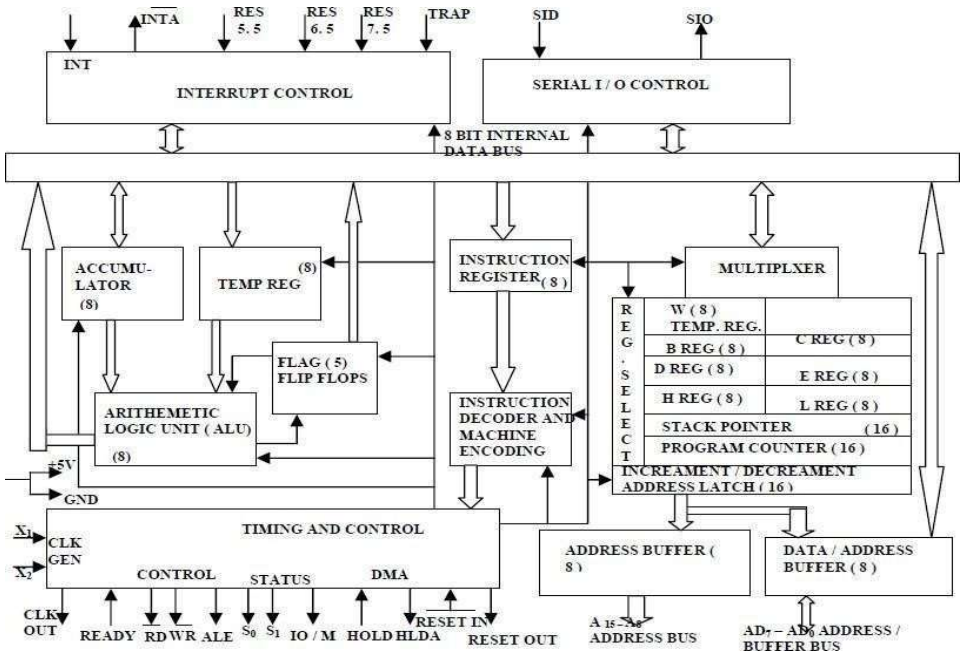
- Microprocessor is one of the most important components of Digital computer.
- It acts as a brain of the computer.
- Microprocessor is the electronic device and it is situated in the CPU.
- Using this processor, we execute the program.so it is called programable integrated circuit.

Important Term in Microprocessor:

- **Bit**: A bit is a single binary digit.
- **Word**: A word refers to the basic data size or bit size that can be processed by the arithmetic and logic unit of the processor. A 16-bit binary number is called a word ina 16-bit processor.
- **Bus**: A bus is a group of wires/lines that carry similar information.

1- Microprocessor is one component of amicrocomputer. 2- Microprocessor is a programmable integrated circuit which has its own decision making capability. Example-8085, INTEL8086,8088,8008,8080 etc.	**1-**A digital computer in which one microprocessor is used as a CPU known as microcomputer. **2-**Microcomputer uses a microprocessor for its processing operation . Example-Desktop, Laptop, Note Book etc.

- **Memory Word**: The number of bits that can be stored in a register or memory element is called a



memory word.

Difference between Microprocessor and Microcomputer:

Application of Microprocessor:

- There are many applications of Microprocessor which is used in -control seven segment LED display like Microprocessor based Traffic light control.
- Controlling of Stepper motor, TV remote, Microprocessor based home security system.

4.2-Architecture of Intel 8085a Microprocessor and description of each block.

Architecture of 8 Bit 8085 Microprocessor:

Description of Each Block:

- **In** 1975 INTEL corporation developed a more power full b bit MP by using NMOS Technology know as INTEL 8085 Microprocessor.

- It is a 40 pin dual package IC fabricated on a single LSI chip.

- The INTEL 8085 uses a single +5 volt supply for its operation and its clock speed is 3 MHZ and clock cycle is 320 nano sec.

Physical components of 8085 Microprocessor:

- Register set.
- Bus interface unit (BIU).
- Arithmetic & logic unite (ALU).
- Instruction decoder & Machine cycle encoder.
- Timing and control unit.
- Interrupt and serial communication.

Program Counter (PC)

- This 16-bit register deals with sequencing the execution of instructions.
- This register is a memory pointer.
- The microprocessor uses this register to sequence the execution of the instructions.
- The function of the program counter is to point to the memory address from which the next byte is to be fetched.
- When a byte is being fetched, the program counter is automatically incremented by one to point to the next memory location. **Increment and Decrement register**
- When the instruction is fetched the value of the program counter is increment and decrement by the increment and decrement Register.

Bus Interface unit (BIU)

- BIU consist of Address buffer or Address bus , Address and data buffer or Address and data bus.
- Address buffer or Address bus are A8-A15 and Address and Data bus are AD0-AD7.
- A8-A15 Address bus are used for MSB bits of memory Address.
- AD0-AD7, these lines are time multiplexed with address and date. They are used for LSB of memory address.

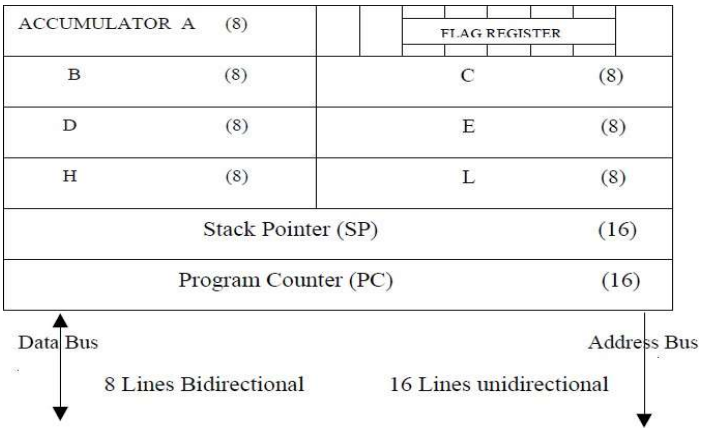
Instruction Register (IR)

- The data fetch from memory is stored by IR register.
- IR only holds that type of data which is fetch from memory.

Instruction Decoder:

- Instruction Decoder Decode the instruction by 0 or 1 **Timing and control unit:**
- Timing and control unit provide the information through control signal to all the register present in microprocessor.
- It controls the entire operation of the MP. So it is known as brain of the computer.

Arithmetic Logic Unit (ALU)



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	**Z**		**AC**		**P**		**CY**

- The ALU performs the actual numerical and logical operations such as Addition (ADD), Subtraction (SUB), AND, OR etc.
- It uses data from memory and from Accumulator to perform operations.
- The results of the arithmetic and logical operations are stored in the accumulator.

Register Set:

- The 8085 includes six registers, one accumulator and one flag register.
- It has two 16-bit registers: stack pointer and program counter.
- The 8085 has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H and L.
- They can be combined as register pairs - BC, DE and HL to perform some bit operations.
- The programmer can use these registers to store or copy data into the register by using data copy instructions.

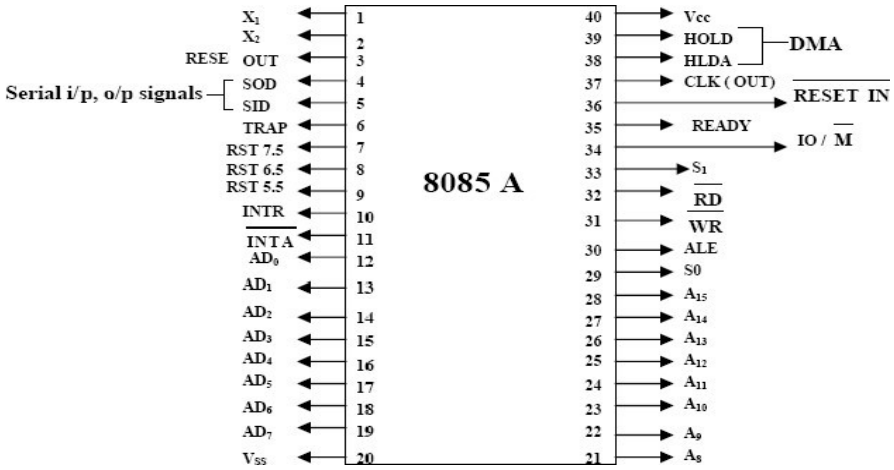
Flag register:

- The ALU includes five flip-flops, which are set or reset after an operation according to data condition of the result in the accumulator and other registers.
- The five-status flag of INTEL 8085 MP are—
- Zero (Z), Carry (CY), Sign (S), Parity (P) and Auxiliary Carry (AC) flags.
- Their bit positions in the flag register are shown in Fig.
- The microprocessor uses these flags to test data conditions.

Stack Pointer:

- The stack pointer is also a 16-bit register, used as a memory pointer.
- It points to a memory location in R/W memory, called stack.
- The beginning of the stack is defined by loading 16- bit address in the stack pointer.

- **Stack**-The Stack is a sequence of memory location set by a programmer to store different element.



So, it is known as Storage device.

- Stack works by LIFO (Last in First out) Operation.

Interrupt Control:

- INTA-Interrupt Acknowledgement.
- INTR-Interrupt Request
- If there is any Interrupt generated inside MP then the Interrupt signal send the request to the Microprocessor.
- There are 5 Interrupt signal i.e TRAP,RST 7.5, RST 6.5, RST 5.5 and INTR.

Serial input and output control:

- It has two input/output pin that is SID & SOD
- SID stands for serial input data and SOD for serial output data.

4.3-Pin Diagram and description.

Properties:

- It is a 8-bit microprocessor
- Manufactured with N-MOS technology
- 40 pin IC package
- It has 16-bit address bus and thus has 2^{16} = 64 KB addressing capability.
- Operate with 3 MHz single-phase clock.
- All the signals are classified into six Group that is

1- address bus

2-Data bus

3- Control & status signals

4- Power supply and frequency signals

5-Externally initiated signals

7-Serial I/O signals

Address and Data Buses:

- A8 – A15 (output, 3-state): Most significant eight bits of memory addresses and the eight bits of the I/O addresses. These lines enter into tri-state high impedance state during HOLD and HALT modes
- AD0 – AD7 (input/output, 3-state): Lower significant bits of memory addresses and the eight bits of the I/O addresses during first clock cycle.

Control & Status Signals:

- ALE: Address latch enable
- RD: Read control signal.
- WR: Write control signal.
- IO/M, S1 and S0 : Status signals.

Power Supply & Clock Frequency:

- Vcc: +5 V power supply
- Vss: Ground reference
- X1, X2: A crystal having frequency of 6 MHz is connected at these two pins
- CLK: Clock output.

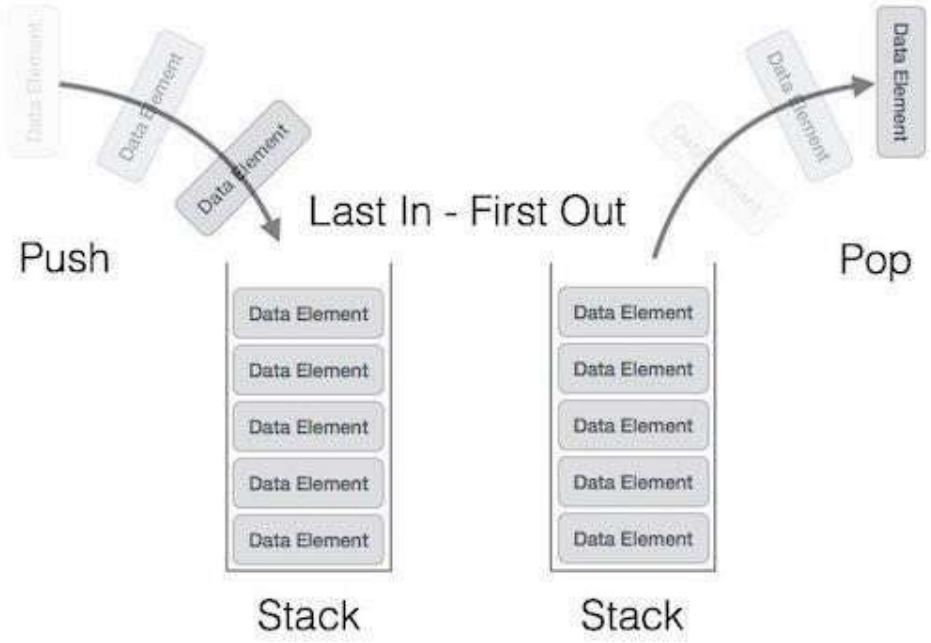
Externally Initiated and Interrupt Signals:

- RESET IN : When the signal on this pin is low, the PC is set to 0, the buses are tri- stated and the processor is reset.
- RESET OUT: This signal indicates that the processor is being reset. The signal can be used to reset other devices.
- READY: When this signal is low, the processor waits for an integral number of clock cycles until it goes high.
- HOLD: This signal indicates that a peripheral like DMA (direct memory access) controller is requesting the use of address and data bus.
- HLDA: This signal acknowledges the HOLD request.
- INTR: Interrupt request is a general-purpose interrupt.
- INTA: This is used to acknowledge an interrupt.
- RST 7.5, RST 6.5, RST 5,5 – restart interrupt: These are vectored interrupts and have highest priority than INTR interrupt.
- TRAP: This is a non-maskable interrupt and has the highest priority

Serial I/O Signals:

- SID: Serial input signal. Bit on this line is loaded to D7 bit of register A using RIM instruction.
- SOD: Serial output signal. Output SOD is set or reset by using SIM instruction.

4.4-Stack, Stack Pointer, Stack top.



Stack

- Stack is used to store and retrieve return addresses during function calls.
- It is also used to transfer arguments to a function. On a microprocessor it is also used to store the status register contents before a context switch. The stack is a temporary store for data.
- There are two types of stacks they are register stack and the memory stack.

Stack Pointer:

- The stack pointer is also a 16-bit register, used as a memory pointer.
- It points to a memory location in R/W memory, called stack.
- The beginning of the stack is defined by loading 16- bit address in the stack pointer.
- A stack (also called a pushdown stack) operates in a last-in/first-out sense.
- When a new data item is entered or "pushed" onto the top of a stack, the stack pointer increments to the next physical memory address, and the new item is copied to that address.
- When a data item is "pulled" or "popped" from the top of a stack, the item is copied from the address of the stack pointer, and the stack pointer decrements to the next available item at the top of the stack.

Stack top:

- The stack is a LIFO (last in, first out) data structure implemented in the RAM area and is used to store addresses and data.
- when the microprocessor branches to a subroutine..... The Stack Pointer register will hold the address of the top location of the stack is known as Stack top.

4.5-Interrupt

Interrupt Structure:

- Interrupt is the mechanism by which the processor is made to transfer control from its current program execution to another program having higher priority.
- The interrupt signal may be given to the processor by any external peripheral device **Types of Interrupts:**
- Interrupts are classified based on their mask ability, IVA and source. They are classified as:
- Vectored and Non-Vectored Interrupts
- Vectored interrupts require the IVA to be supplied by the external device thatgives the interrupt signal. This technique is vectoring, is implemented in number of ways.
- Non-vectored interrupts have fixed IVA for ISRs of different interrupt signals.

Maskable and Non-Maskable Interrupts

- Maskable interrupts are interrupts that can be blocked. Masking can be done by software or hardware means.
- Non-maskable interrupts are interrupts that are always recognized; the corresponding ISRs are executed.

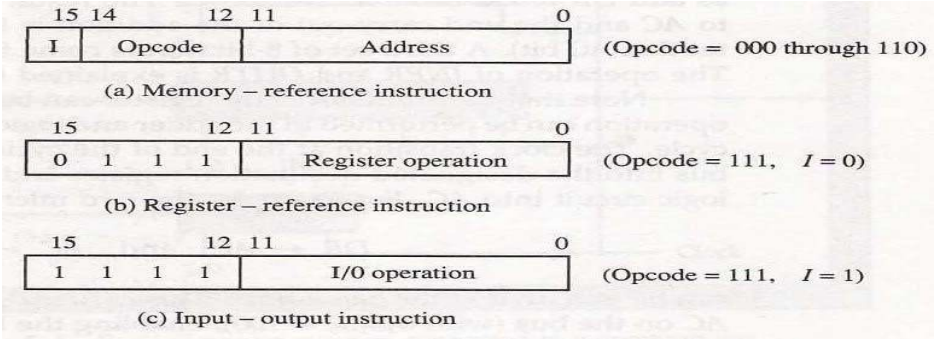
Software and Hardware Interrupts

- Software interrupts are special instructions, after execution transfer the control to predefined ISR.
- Hardware Interrupts and Priorities:
- 8085 have five hardware interrupts – INTR, RST 5.5, RST 6.5, RST 7.5 and TRAP.

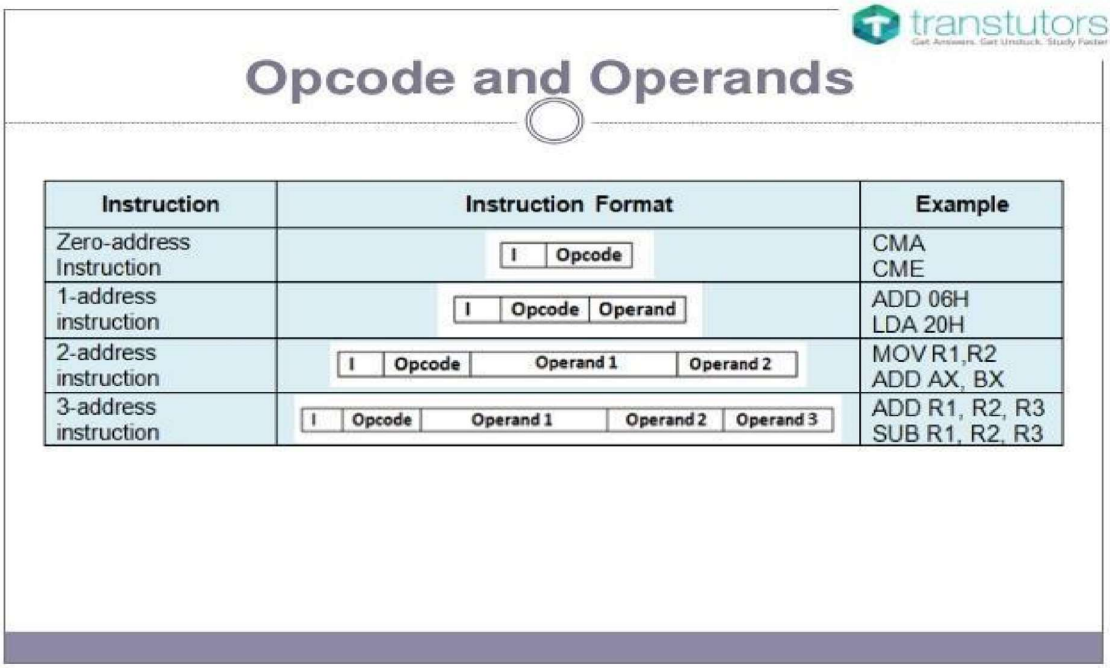
4.6-opcode and Operand

Opcode:

- In computing, an opcode (abbreviated from **operation code**, also known as instruction machine code, instruction code is the portion of a machine language instruction that specifies the operation to be performed.
- Opcodes mean “operation codes”. An opcode is the first part of an instruction that tells the computer what function to perform. Every computer has an operation code or opcode for each of its functions.



- ✓ An operand is the second part of the instruction, which tells the computer where to find or store the data or instructions.
- ✓ The number of operands varies among computers.
- ✓ Each instruction tells the Control Unit of the CPU what to do and how to do it. The operations are Arithmetic, Logical, Branch operation, etc depending upon the problem that is given to the computers.



4.7-Differentiate between one byte, two byte & three-byte instruction with example.

One byte, two byte & three-byte instruction with example.

- ✓ According to the length, the instruction of 8085 microprocessor is classified into 3 types.
- ✓ 1-Single byte instruction
- ✓ 2-Two-byte instruction
- ✓ 3-Three-byte instruction

Single byte instruction

- ✓ **In** single byte instruction only opcode is present, there is no operand.
- ✓ Examples-MOV A, B, ADD B, CMAIn these instructions only opcode is present so these are the single byte instruction.
- ✓ The length of this instruction is 8 bits. Each instruction requires one memory location.

Operand:

Two-byte instruction

- In two-byte instruction the first 8 bit indicates the opcode and next 8 bit indicates the operand
- .Example MVI A,32H, ADI A,08H
- The length of this instruction is 16 bit that is the opcode is 8 bit and operand is 8 bits. Each instruction requires two memory location. **Three-byte instruction**
- Three instruction is the type of instruction in which the first 8 bit indicates the opcode and next 2 bytes specified the operand which is 16 b it address.
- The low order address is represented in 2nd byte and the higherorders address represented in the 3rd byte.
- Example-LBA 2050H, JMP 2085H
- This instruction would require 3 memory location to store the binary code.

4.8-Instruction set of 8085 with example.

Instruction set of 8085:

Instruction sets are instruction codes to perform some task. It is classified into five categories.

1-Control Instruction

2-Logical Instruction

3-Branching Instruction

4-Arithmatic Instruction

5- Data transfer Instruction

Control Instruction:

Opcode	Operand	Meaning	Explanation
NOP	None	No operation	No operation is performed, i.e., the instruction is fetched and decoded.
HLT	None	Halt and enter wait state	The CPU finishes executing the current instruction and stops further execution. An interrupt or reset is necessary to exit from the halt state.
DI	None	Disable interrupts	The interrupt enable flip-flop is reset and all the interrupts are disabled except TRAP.

EI	None	Enable interrupts	The interrupt enable flip-flop is set and all the interrupts are enabled.

RIM	None	Read interrupt mask	This instruction is used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit.
SIM	None	Set interrupt mask	This instruction is used to implement the interrupts 7.5, 6.5, 5.5, and serial data output.

Logical Instruction:

Opcode	Operand	Meaning	Explanation
CMP	R M	Compare the register or memory with the accumulator	The contents of the operand (register or memory) are M compared with the contents of the accumulator.
CPI	8-bit data	Compare immediate with the accumulator	The second byte data is compared with the contents of the accumulator.
ANA	R M	Logical AND register or memory with the accumulator	The contents of the accumulator are logically AND with M the contents of the register or memory, and the result is placed in the accumulator.

ANI	8-bit data	Logical AND immediate with the accumulator	The contents of the accumulator are logically AND with the 8-bit data and the result is placed in the accumulator.
XRA	R M	Exclusive OR register or memory with the accumulator	The contents of the accumulator are Exclusive OR with M the contents of the register or memory, and the result is placed in the accumulator.
XRI	8-bit data	Exclusive OR immediate with the accumulator	The contents of the accumulator are Exclusive OR with the 8-bit data and the result is placed in the accumulator.
ORA	R M	Logical OR register or memory with the accumulator	The contents of the accumulator are logically OR with M the contents of the register or memory, and result is placed in the accumulator.

	ORI	8-bit data	Logical OR immediate with the accumulator	The contents of the accumulator are logically OR with the 8-bit data and the result is placed in the accumulator.

RLC	None	Rotate the accumulator left	Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D0 as well as in the Carry flag. CY is modified according to bit D7.
RRC	None	Rotate the accumulator right	Each binary bit of the accumulator is rotated right by one position. Bit D0 is placed in the position of D7 as well as in the Carry flag. CY is modified according to bit D0.
RAL	None	Rotate the accumulator left through carry	Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7.
RAR	None	Rotate the accumulator right through carry	Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0.
CMA	None	Complement accumulator	The contents of the accumulator are complemented. No flags are affected.
CMC	None	Complement carry	The Carry flag is complemented. No other flags are affected.
STC	None	Set Carry	Set Carry

Branching Instruction:

Opcode	Operand	Meaning	Explanation
JMP	16-bit address	Jump unconditionally	The program sequence is transferred to the memory address given in the operand.

	<table><tr><th>Opcode</th><th>Description</th><th>Flag Status</th></tr><tr><td>JC</td><td>Jump on Carry</td><td>CY=1</td></tr><tr><td>JNC</td><td>Jump on no Carry</td><td>CY=0</td></tr><tr><td>JP</td><td>Jump on positive</td><td>S=0</td></tr><tr><td>JM</td><td>Jump on minus</td><td>S=1</td></tr><tr><td>JZ</td><td>Jump on zero</td><td>Z=1</td></tr><tr><td>JNZ</td><td>Jump on no zero</td><td>Z=0</td></tr><tr><td>JPE</td><td>Jump on parity even</td><td>P=1</td></tr><tr><td>JPO</td><td>Jump on parity odd</td><td>P=0</td></tr></table>	16-bit address	Jump conditionally	The program sequence is transferred to the memory address given in the operand based on the specified flag of the PSW.
	<table><tr><th>Opcode</th><th>Description</th><th>Flag Status</th></tr><tr><td>CC</td><td>Call on Carry</td><td>CY=1</td></tr><tr><td>CNC</td><td>Call on no Carry</td><td>CY=0</td></tr><tr><td>CP</td><td>Call on positive</td><td>S=0</td></tr></table>	16-bit address	Unconditional subroutine call	The program sequence is transferred to the memory address given in the operand. Before transferring, the address of the next instruction after CALL is pushed onto the stack.

<table><tr><td>CM</td><td>Call on minus</td><td>S=1</td></tr><tr><td>CZ</td><td>Call on zero</td><td>Z=1</td></tr><tr><td>CNZ</td><td>Call on no zero</td><td>Z=0</td></tr><tr><td>CPE</td><td>Call on parity even</td><td>P=1</td></tr><tr><td>CPO</td><td>Call on parity odd</td><td>P=0</td></tr></table>			
RET	None	Return from subroutine unconditionally	The program sequence is transferred from the subroutine to the calling program.

Opcode	Description	Flag Status			
RC	Return on Carry	CY=1	None	Return from subroutine conditionally	The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW and the program execution begins at the new address.
RNC	Return on no Carry	CY=0			
RP	Return on positive	S=0			
RM	Return on minus	S=1			
RZ	Return on zero	Z=1			

RNZ	Return on no zero	Z=0			
RPE	Return on parity even	P=1			
RPO	Return on parity odd	P=0			

<table>
<tr>
<td>PCHL</td>
<td>None</td>
<td>Load the program counter with HL contents</td>
<td>The contents of registers H & L are copied into the program counter. The contents of H are placed as the high-order byte and the contents of L as the low order byte.</td>
</tr>
<tr>
<td>RST</td>
<td>0-7</td>
<td>Restart</td>
<td>The RST instruction is used as software instructions in a program to transfer the program execution to one of the following eight locations.
<table>
<tr><th>Instruction</th><th>Restart Address</th></tr>
<tr><td>RST 0</td><td>0000H</td></tr>
<tr><td>RST 1</td><td>0008H</td></tr>
<tr><td>RST 2</td><td>0010H</td></tr>
<tr><td>RST 3</td><td>0018H</td></tr>
<tr><td>RST 4</td><td>0020H</td></tr>
<tr><td>RST 5</td><td>0028H</td></tr>
<tr><td>RST 6</td><td>0030H</td></tr>
</table>
</td>
</tr>
</table>

			<table><tr><td>RST 7</td><td>0038H</td></tr></table> The 8085 has additionally 4 interrupts, which can generate RST instructions internally and doesn't require any external hardware. Following are those instructions and their Restart addresses − <table><tr><td>Interrupt</td><td>Restart Address</td></tr><tr><td>TRAP</td><td>0024H</td></tr><tr><td>RST 5.5</td><td>002CH</td></tr><tr><td>RST 6.5</td><td>0034H</td></tr><tr><td>RST 7.5</td><td>003CH</td></tr></table>

Arithmetic Instruction:

Opcode	Operand	Meaning	Explanation
ADD	R M	Add register or memory, to the accumulator	The contents of the register or memory are added to the contents of the accumulator and the result is stored in the accumulator. Example − ADD K.
ADC	R M	Add register to the accumulator with carry	The contents of the register or memory & M the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. Example − ADC K
ADI	8- bit data	Add the immediate to the accumulator	The 8-bit data is added to the contents of the accumulator and the result is stored in the accumulator.

			Example − ADI 55K
ACI	8-bit data	Add the immediate to the accumulator with carry	The 8-bit data and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. Example − ACI 55K
LXI	Reg. pair, 16bit data	Load the register pair immediate	The instruction stores 16-bit data into the register pair designated in the operand. Example − LXI K, 3025M
DAD	Reg. pair	Add the register pair to H and L registers	The 16-bit data of the specified register pair are added to the contents of the HL register. Example − DAD K
SUB	R M	Subtract the register or the memory from the accumulator	The contents of the register or the memory are subtracted from the contents of the accumulator, and the result is stored in the accumulator. Example − SUB K
SBB	R M	Subtract the source and borrow from the accumulator	The contents of the register or the memory & M the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. Example − SBB K
SUI	8-bit data	Subtract the immediate from the accumulator	The 8-bit data is subtracted from the contents of the accumulator & the result is stored in the accumulator. Example − SUI 55K

XCHG	None	Exchange H and L with D and E	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. Example − XCHG

INR	R M	Increment the register or the memory by 1	The contents of the designated register or the memory are incremented by 1 and their result is stored at the same place. Example − INR K
INX	R	Increment register pair by 1	The contents of the designated register pair are incremented by 1 and their result is stored at the same place. Example − INX K
DCR	R M	Decrement the register or the memory by 1	The contents of the designated register or memory are decremented by 1 and their result is stored at the same place. Example − DCR K
DCX	R	Decrement the register pair by 1	The contents of the designated register pair are decremented by 1 and their result is stored at the same place. Example − DCX K
DAA	None	Decimal adjust accumulator	The contents of the accumulator are changed from a binary value to two 4-bit BCD digits. If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits. If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits. Example − DAA

Data transfer Instruction:

Opcode	Operand	Meaning	Explanation
MOV	Rd, Sc M, Sc Dt, M	Copy from the source (Sc) to the destination(Dt)	This instruction copies the contents of the source register into the destination register without any alteration. Example − MOV K, L
MVI	Rd, data M, data	Move immediate 8-bit	The 8-bit data is stored in the destination register or memory. Example − MVI K, 55L
LDA	16-bit address	Load the accumulator	The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. Example − LDA 2034K
LDAX	B/D Reg. pair	Load the accumulator indirect	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. Example − LDAX K
LXI	Reg. pair, 16-bit data	Load the register pair immediate	The instruction loads 16-bit data in the register pair designated in the register or the memory. Example − LXI K, 3225L

LHLD	16-bit address	Load H and L registers direct	The instruction copies the contents of the memory location pointed out by the address into register L and copies the contents of the next memory location into register H. Example − LHLD 3225K
STA	16-bit address	16-bit address	The contents of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the highorder address.

			Example − STA 325K
STAX	16-bit address	Store the accumulator indirect	The contents of the accumulator are copied into the memory location specified by the contents of the operand. Example − STAX K
SHLD	16-bit address	Store H and L registers direct	The contents of register L are stored in the memory location specified by the 16- bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the highorder address. Example − SHLD 3225K
XCHG	None	Exchange H and L with D and E	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. Example − XCHG
SPHL	None	Copy H and L registers to the stack pointer	The instruction loads the contents of the H and L registers into the stack pointer register. The contents of the H register provide the high-order address and the contents of the L register provide the low- order address. Example − SPHL
XTHL	None	Exchange H and L with top of stack	The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1).

			Example − XTHL
PUSH	Reg. pair	Push the register pair onto the stack	The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The stack pointer register is decremented and the contents of the high order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location. Example − PUSH K
POP	Reg. pair	Pop off stack to the register pair	The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1. Example − POPK
OUT	8-bit port address	Output the data from the accumulator to a port with 8bit address	The contents of the accumulator are copied into the I/O port specified by the operand. Example − OUT K9L
IN	8-bit port address	Input data to accumulator from a port with 8-bit address	The contents of the input port designated in the operand are read and loaded into the accumulator. Example − IN5KL

4.9-Adressing mode of 8085 Microprocessor

- The process of specifying the data to be operated on by the instruction is called addressing.
- The various formats for specifying operands are called addressing modes. The 8085 has thefollowing five types of addressing modes.
    - Immediate addressing
    - Memory direct addressing
    - Register direct addressing
    - Indirect addressing
    - Implicit addressing

Immediate Addressing:

- . In this mode, the operand given in the instruction - a byte or word – transfers to the destination register or memory location. Ex: MVI A, 9AH
- The operand is a part of the instruction.
- The operand is stored in the register mentioned in the instruction.

Memory Direct Addressing:

- Memory direct addressing moves a byte or word between a memory location and register.
- The memory location address is given in the instruction.

Ex: LDA 850FH

- This instruction is used to load the content of memory address 850FH in the accumulator.

Register Direct Addressing:

- Register direct addressing transfer a copy of a byte or word from source register to destination register. Ex: MOV B, C
- It copies the content of register C to register B.

Indirect Addressing:

- Indirect addressing transfers a byte or word between a register and a memory location.

Ex: MOV A, M

- Here the data is in the memory location pointed to by the contents of HL pair. The data ismoved to the accumulator.

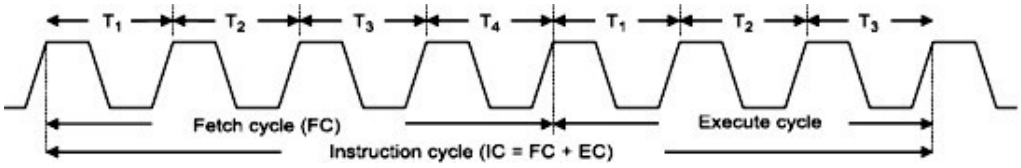
Implicit Addressing:

- In this addressing mode the data itself specifies the data to be operated upon.

Ex: CMA

- The instruction complements the content of the accumulator.
- No specific data or operand is mentioned inthe instruction.

4.10-Fetch cycle, Machine cycle, Instruction cycle, T-State



Fetch cycle:

- The first byte of an instruction is its op-code.
- An instruction may be more than one byte long.
- The other bytes are data or operand address.
- The program counter (PC) keeps the memory address of the next instruction to be executed.
- In the beginning of a fetch cycle the content of the program counter, which is the address of the memory location where op-code is available, is sent to the memory.
- ·The memory places the op-code on the data bus so as to transfer it to the microprocessor.
- The entire operation of fetching an op-code takes three clock cycles.

Machine Cycle:

- Machine cycle is defined as the time required for completing the operation of accessing either memory or I/O device.
- In the 8085, the machine cycle may consist of three to six T states.
- The T-state is defined as one sub-division of the operation performed in one clock period.
- ·In every machine cycle the first operation is op-code fetch and the remaining will be read or write from memory or IO devices.

Instruction Cycle:

- ·An instruction is a command given to the microprocessor to perform a specific operation on the given data.
- ·Sequence of instructions written for a processor to perform a particular task is called a program.
- The microprocessor fetches one instruction from the memory at a time & executes it.
  It executes all the instructions of the program one by one to produce the final result.
- In other words, an instruction cycle is defined as the time required completing the execution of an instruction.
- An instruction cycle consists of a fetch cycle and an execute cycle.
- The time required to fetch an opcode (fetch cycle) is a fixed slot of time while the time required to execute an instruction (execute cycle) is variable which depends on the type of instruction to be executed.

Instruction cycle (IC) = Fetch cycle (FC) Execute cycle (EC)

T-state:

- One time period of frequency of microprocessor is called t-state.
- A t-state is measured from the falling edge of one clock pulse to the falling edge of the next clock pulse.
- Fetch cycle takes four t-states and execution cycle takes three t-states

4.11-Timing Diagram for Opcode Fetch, Memory Read, Memory Write, I/O read and I/O write.

Various operation of 8085 MP

To execute a program, 8085 performs various operations as:

- Opcode fetch
- Operand fetch
- Memory read/write
- I/O read/write

Opcode Fetch Machine Cycle:

- It is the first step in the execution of any instruction.
- The following points explain the various operations that take place and the signals that are changed during the execution of opcode fetch machine cycle: T1 clock cycle
- The content of PC is placed in the address bus; AD0 - AD7 lines contains lower bit address and A8 – A15 contains higher bit address.
- IO/M signal is low indicating that a memory location is being accessed. S1 and S0 also changed to the levels.
- ALE is high, indicates that multiplexed AD0 – AD7 act as lower order bus.

T2 clock cycle

- Multiplexed address bus is now change to Data bus.
- The RD signal is made low by the processor. This signal makes the memory device load the data bus with the contents of the location addressed by the processor.

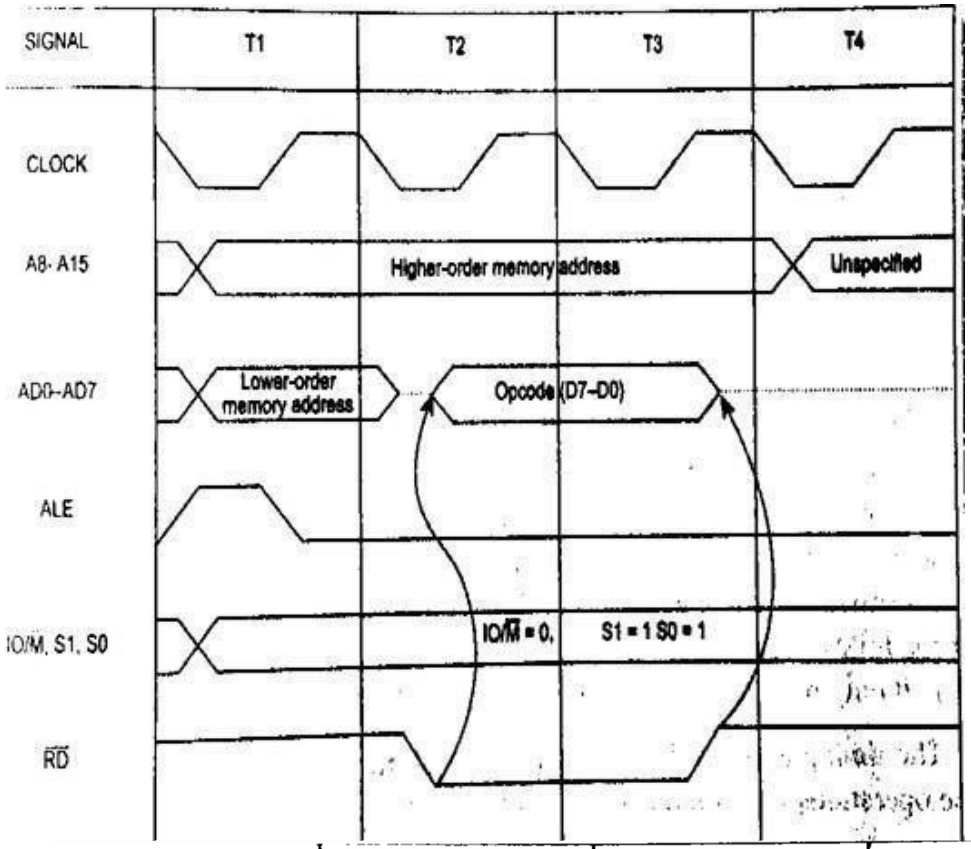
T3 clock cycle

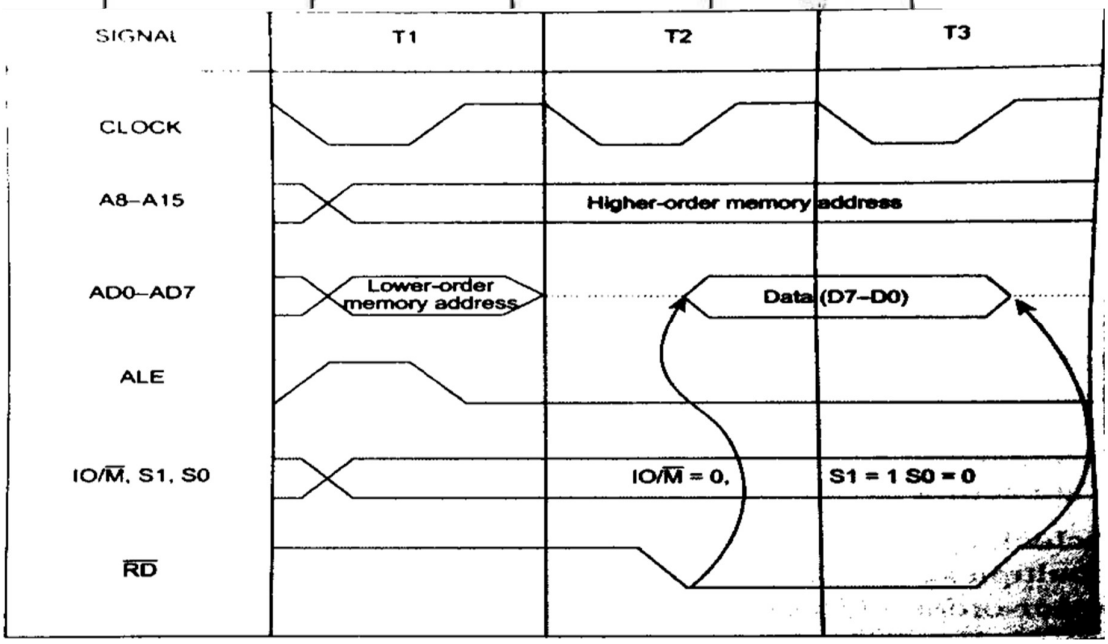
- The opcode available on the data bus is read by the processor and moved to the instruction register.
- The RD signal is deactivated by making it logic 1.

T4 clock cycle

- The processor decode the instruction in the instruction register and generate the necessary control signals to execute the instruction.
- Based on the instruction further operations such as fetching, writing into memory etc takes place.

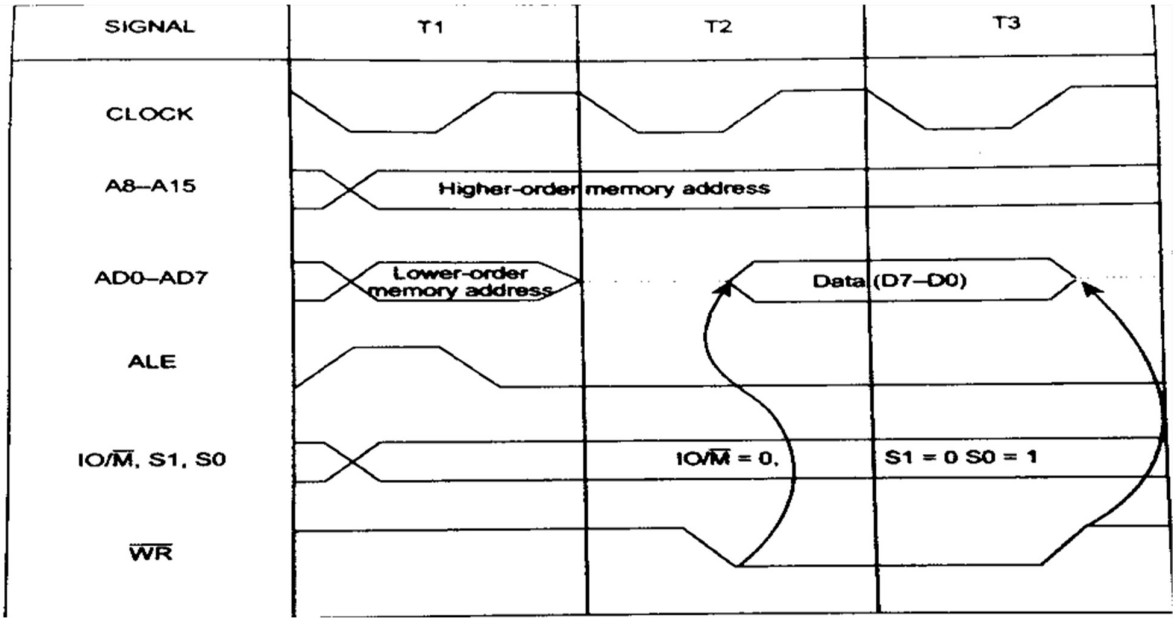
Me


SIGNAL
T1
T2
T3
T4
CLOCK
A8- A15
Higher-order memory address
Unspecified
AD0-AD7
Lower-order memory address
Opcode (D7–D0)
ALE
IO/M, S1, S0
IO/M = 0,
S1 = 1 S0 = 1
RD


SIGNAL
T1
T2
T3
CLOCK
A8–A15
Higher-order memory address
AD0–AD7
Lower-order memory address
Data (D7–D0)
ALE
IO/M, S1, S0
IO/M = 0,
S1 = 1 S0 = 0
RD

- The memory read cycle is executed by the processor to read a data byte from memory.
- The machine cycle isexactly same to opcode fetch except: a) It has three T-states b) The S0 signal is set to 0.
- The timing diagramof this cycle is given in Fig.

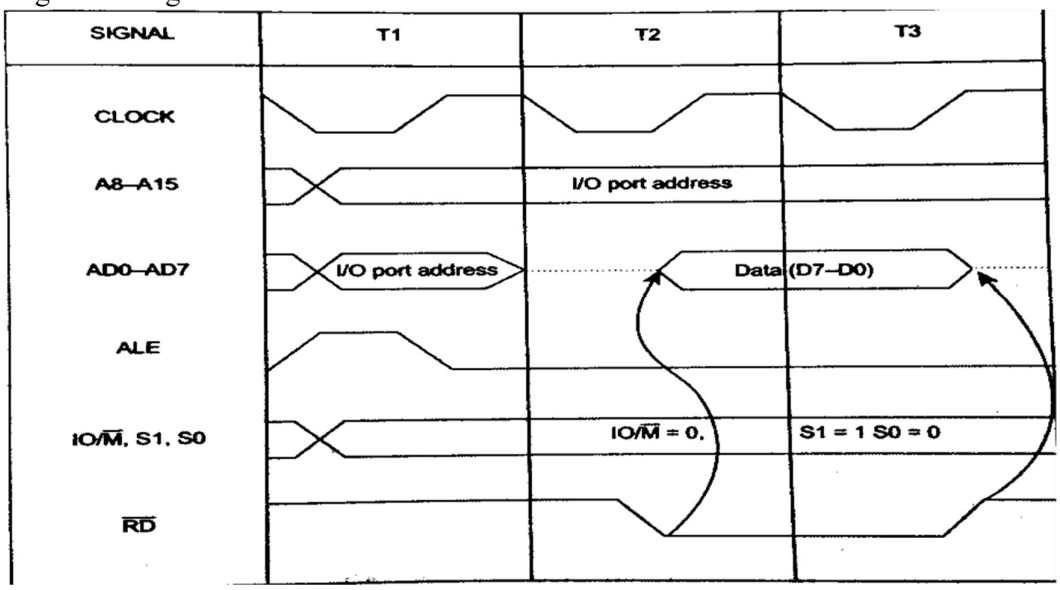
Memory Write Machine cycle:



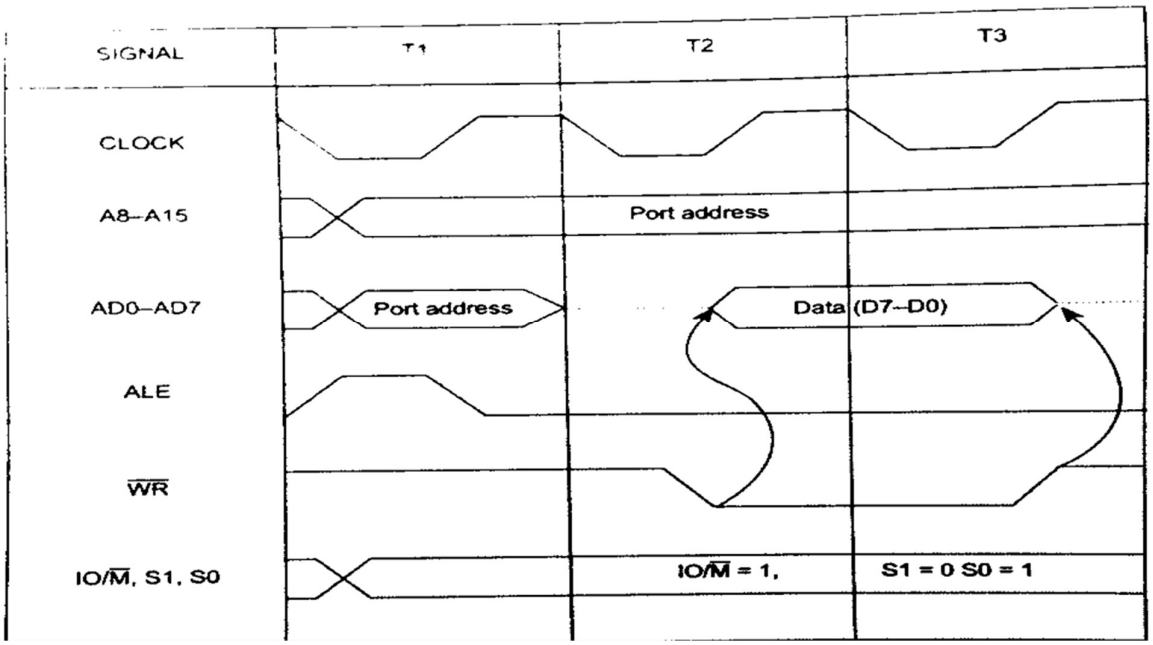
- The memory write cycle is executed by the processor to write a data byte in a memory location.
- The processor takes three T-states and WR signal is made low.
- The timing diagram of this cycle is given in Fig

I/O Read Cycle:

- The I/O read cycle is executed by the processor to read a data byte from I/O port or from peripheral, which is I/O mapped in the system.
- The 8-bit port address is placed both in the lower and higher order address bus.
- The processor takes three T-states to execute this machine cycle. The timing diagram of this cycle is given in Fig.



I/O Write Cycle:



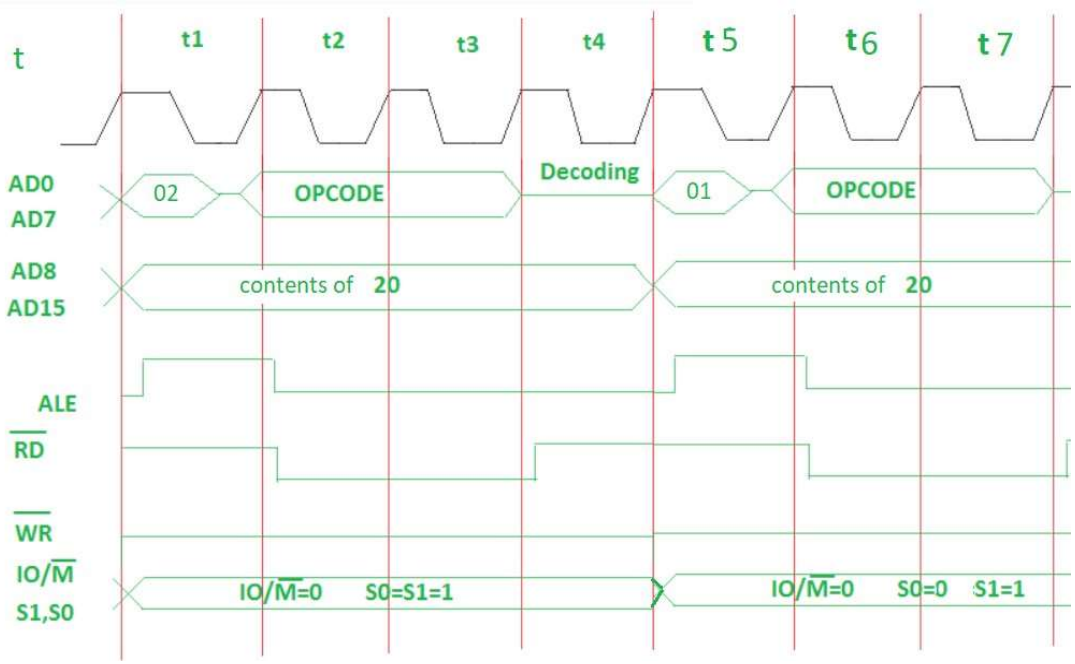
- The I/O write cycle is executed by the processor to write a data byte to I/O port or to a peripheral, which is I/O mapped in the system.
- The processor takes three T-states to executethis machine cycle.
- The timing diagram of this cycle is given in Fig.

4.12-Timing Diagram for 8085 Instruction

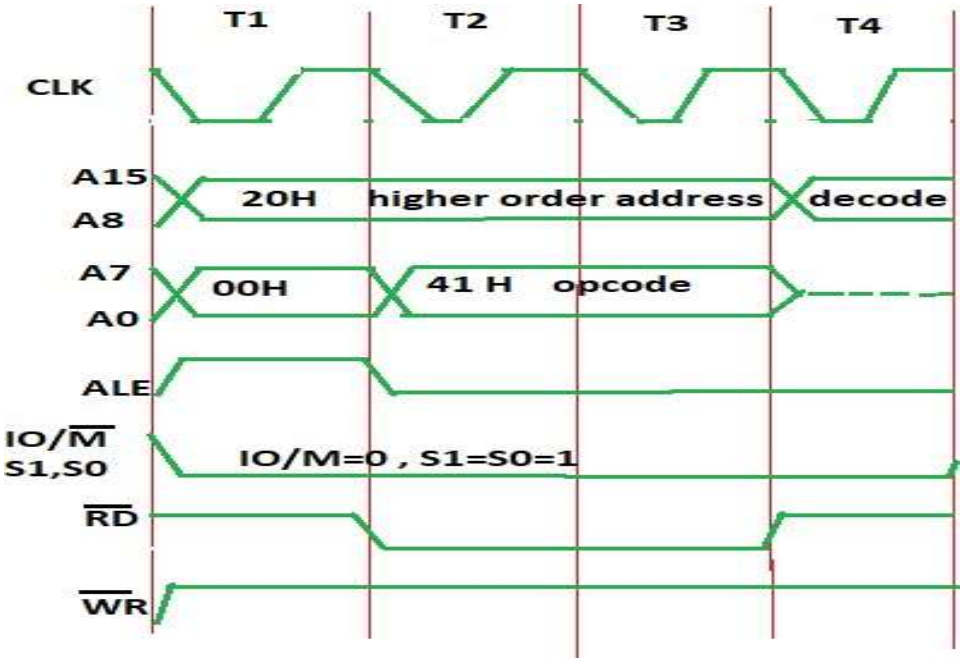
Draw the timing diagram of the following code, MVI B,

45

- Opcode: MVI
- Operand: B is the destination register and 45 is the source data which needs to be transferred to the register.
- 45’ data will be stored in the B register.
- The opcode fetch will be same in all the instructions.
- Only the read instruction of the opcode needs to be added in the successive T states.
- For the opcode fetch the IO/M (low active) = 0, S1 = 1 and S0 = 1. Also, 4 T states will be required to fetch the opcode from memory.
- For the opcode read the IO/M (low active) = 0, S1 = 1 and S0 = 0. Also, only 3 T states will be required to read data from memory.


t
t1
t2
t3
t4
t5
t6
t7
AD0
AD7
02
OPCODE
Decoding
01
OPCODE
AD8
AD15
contents of 20
contents of 20
ALE
RD
WR
IO/M
S1,S0
IO/M=0 S0=S1=1
IO/M=0 S0=0 S1=1

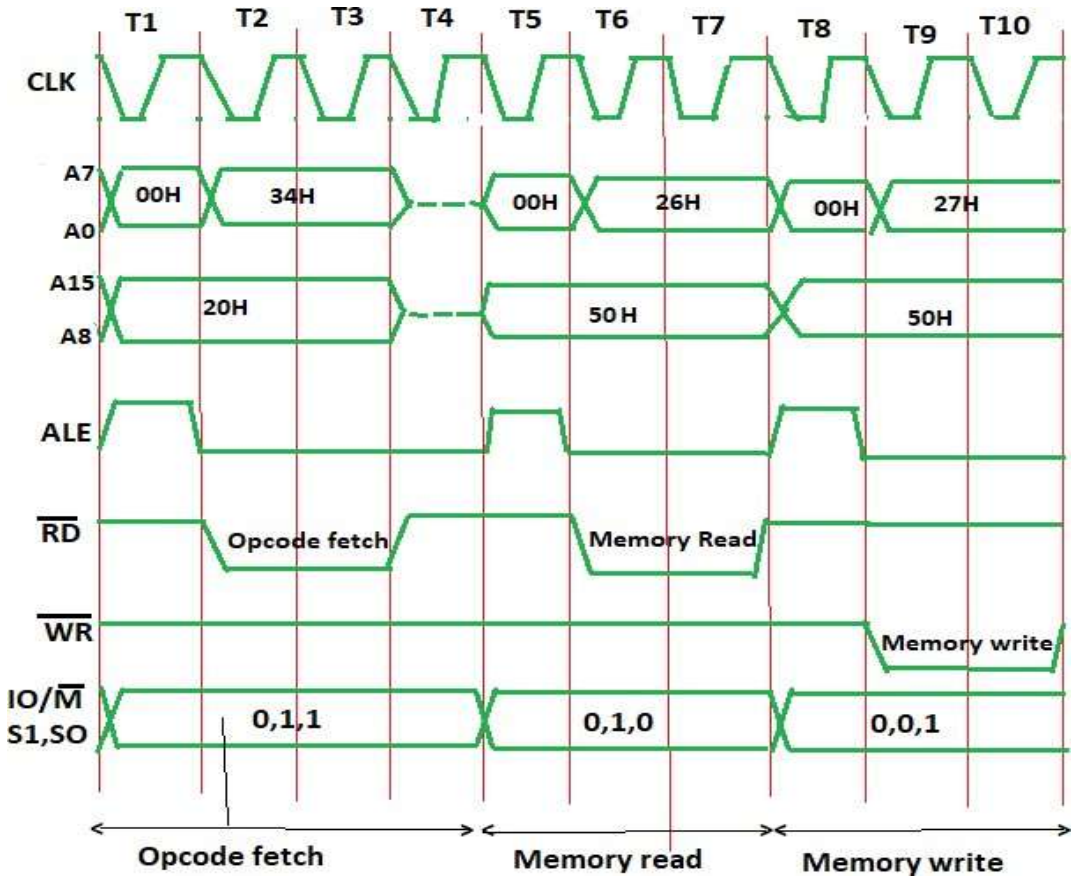
Draw the timing diagram of the given instruction in 8085, MOV B,



C

- In this instruction Only opcode fetching is required for this instruction and thus we need 4 T states for the timing diagram.
- For the opcode fetch the IO/M (low active) = 0, S1 = 1 and S0 = 1.

In Opcode fetch (t1-t4 T states):



- 00 – lower bit of address where opcode is stored, i.e., 00 20 – higher bit of address where opcode is stored, i.e., 20.
- ALE – provides signal for multiplexed address and data bus. Only in t1 it used as address bus to fetch lower bit of address otherwise it will be used as data bus.
- RD (low active) – signal is 1 in t1 & t4 as no data is read by microprocessor. Signal is 0 in t2 & t3 because here the data is read by microprocessor.
- WR (low active) – signal is 1 throughout, no data is written by microprocessor.
- IO/M (low active) – signal is 1 in throughout because the operation is performing on memory.
- S0 and S1 – both are 1 in case of opcode fetching.

Draw the timing diagram of the given instruction in 8085,

INR M

- The instruction INR M is of 1 byte; therefore, the complete instruction will be stored in a single memory address.
- The opcode fetch will be same as for other instructions in first 4 T states.
- Only the Memory read and Memory Write need to be added in the successive T states.

In Opcode fetch (t1-t4 T states) –

- 00: lower bit of address where opcode is stored, i.e., 00 20: higher bit of address where opcode is stored, i.e., 20.
- ALE: provides signal for multiplexed address and data bus. Only in t1 it used as address bus to fetch lower bit of address otherwise it will be used as data bus.
- RD (low active): signal is 1 in t1 & t4 as no data is read by microprocessor. Signal is 0 in t2 & t3 because here the data is read by microprocessor.
- WR (low active): Signal is 1 throughout, no data is written by microprocessor.
- IO/M (low active): Signal is 0 in throughout because the operation is performing on memory.
- S0 and S1: both are 1 in case of opcode fetching.

In Memory read (t5-t7 T states) –

- 00: lower bit of address where opcode is stored, i.e, 00 50: higher bit of address where opcode is stored, i.e, 50.
- ALE: provides signal for multiplexed address and data bus. Only in t5 it used as address bus to fetch lower bit of address otherwise it will be used as data bus.
- RD (low active): signal is 1 in t5, no data is read by microprocessor. Signal is 0 in t6 & t7, data is read by microprocessor.
- WR (low active): signal is 1 throughout, no data is written by microprocessor.
- IO/M (low active): signal is 0 in throughout; operation is performing on memory.
- S0 and S1 – S1=1 and S0=0 for Read operation. **In Memory write (t8-t10 T states) –**

- 00: lower bit of address where opcode is stored, i.e, 00 50: higher bit of address where opcode is stored, i.e, 50.
- ALE: provides signal for multiplexed address and data bus. Only in t8 it used as address bus to fetch lower bit of address otherwise it will be used as data bus.
- RD (low active): signal is 1 throughout, no data is read by microprocessor.
- WR (low active): signal is 1 in t8, no data is written by microprocessor. Signal is 0 in t9 & t10, data is written by microprocessor.
- IO/M (low active): signal is 0 in throughout; operation is performing on memory.
- S0 and S1 – S1=0 and S0=1 for write operation.

4.13-Counter and Time Delay

Counter in 8085 MP:

- The function of the program counter is to point to the memory address from which the next byte is to be fetched.
- When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location.
- Stack Pointer: It is used as a memory pointer.
- The Program Counter (PC) is a register structure that contains the address pointer value of the current instruction.

- Each cycle, the value at the pointer is read into the instruction decoder and the program counter is updated to point to the next instruction.

Time Delay in 8085 MP:

- The delay will be used in different places to simulate clocks, or counters or some other area.
- When the delay subroutine is executed, the microprocessor does not execute other tasks.
- For the delay we are using the instruction execution times. executing some instructions in a loop, the delay is generated.

4.14-Simple assembly language programming of 8085

Assembly language program:

Write 8085 Assembly language program to perform 8-bit addition without carry. The numbers are stored at F100, and F101. Result will be stored at F102.

Input

Address	Data
…	…
F100	CE
F101	21

Program

Address	HEX Codes	Labels	Mnemonics	Comments
F000	21, 01, F1		LXI H,F100H	Point to get the numbers
F003	7E		MOV A,M	Load first number to A
F004	23		INX H	Point to next operand
F005	86		ADD M	Add M with A
F006	23		INX H	Point to next location

F007	77		MOV M,A	Store result

Address	**HEX Codes**	**Labels**	**Mnemonics**	**Comments**
F008	76		HLT	Terminate the program

Output

Address	**Data**
…	…
F102	EF

Write 8085 Assembly language program to subtract two 8-bit numbers and store the result at locations 8050H and 8051H. Input

first input

Address	**Data**
. . .	. . .

8000	78
8001	5D
. . .	. . .

second input

Address	Data
. . .	. . .
8000	23

Address	Data
8001	CF

Program

Address	HEX Codes	Labels	Mnemonics	Comments
F000	0E,00		MVIC,00H	Clear C register

F002	21,00, 80		LXIH,8000H	Load initial address to get operand
F005	7E		MOVA, M	Load Acc with the memory element
F006	23		INX H	Point to next location
F007	46		MOVB, M	Load B with the second operand
F008	90		SUB B	Subtract B from A
F009	D2,0D, F0		JNC STORE	When CY = 0, go to STORE
F00C	0C		INR C	Increase C by 1
F00D	21,50, 80	STORE	LXIH,8050H	Load the destination address
F010	77		MOVM, A	Store the result
F011	23		INX H	Point to next location
F012	71		MOVM, C	Store the borrow
F013	76		HLT	Terminate the program

Output

first output

Address	Data
. . .	. . .
8050	1B
8051	00
. . .	. . .

second output

Address	Data
. . .	. . .
8050	54
8051	01

Write 8085 Assembly language program to multiply two 8-bit numbers stored in memory location and store the 16-bit results into the memory.

Input

Address	Data
. .	. .

.	.

Address	Data
8000	DC
8001	AC

Program

Address	HEX Codes	Labels	Mnemonics	Comments
F000	21, 00, 80		LXI H,8000H	Load first operand address
F003	46		MOV B, M	Store first operand to B
F004	23		INX H	Increase HL pair
F005	AF		XRA A	Clear accumulator
F006	4F		MOV C, A	Store 00H at register C
F007	86	LOOP	ADD M	Add memory element with Acc
F008	D2, 0C, F0		JNC SKIP	When Carry flag is 0, skip next task

F00B	0C		INR C	Increase C when carry is 1
F00C	05	SKIP	DCR B	Decrease B register
F00D	C2, 07, F0		JNZ LOOP	Jump to loop when Z flag is not 1
F010	21, 50, 80		LXI H,8050H	Load Destination address
F013	71		MOV M, C	Store C register content into memory
F014	23		INX H	Increase HL Pair
F015	77		MOV M, A	Store Acc content to memory

Address	HEX Codes	Labels	Mnemonics	Comments
F016	76		HLT	Terminate the program

Output

Address	Data
. . .	. . .
8050	93
8051	D0

Short Question and Answer:

1- Define Microprocessor and Microcomputer.

Ans- **Microcomputer:**

- Microcomputer is an Electronics Device
- A computer system generally consists of the following units
  1-Input device
  2-Output device
  3-CPU
  4-Memory Unit

Microprocessor:

- Microprocessor is one of the most important components of Digital computer.
- It acts as a brain of the computer.
- Microprocessor is the electronic device and it is situated in the CPU.
- Using this processor, we execute the program.so it is called programable integrated circuit.

2- Difference between Microprocessor and Microcomputer

1-Microprocessor is one component of amicrocomputer. microprocessor is used as a CPU known as	**1**-A digital computer in which one
2-Microprocessor is a programmable integrated circuit mi making capability.	rocomputer. which has its own decision
Example-8085, INTEL8086,8088,8008,8080 etc. for its	**2**-Microcomputer uses a microprocessor processing operation . Example-Desktop, Laptop, Note Book etc.

3-**Define Program counter.**

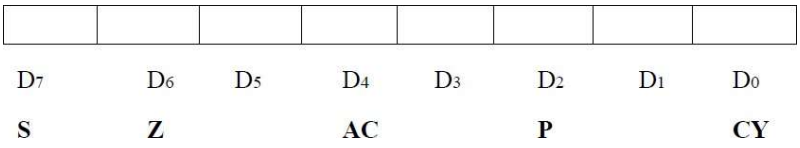
- This 16-bit register deals with sequencing the execution of instructions.
- This register is a memory pointer.
- The microprocessor uses this register to sequence the execution of the instructions
- When a byte is being fetched, the program counter is automatically incremented by one to point to the next memory location.

4-**Define ALU in microprocessor.**

- The ALU performs the actual numerical and logical operations such as Addition (ADD), Subtraction (SUB), AND, OR etc.
- It uses data from memory and from Accumulator to perform operations.
- The results of the arithmetic and logical operations are stored in the accumulator.

5-**Define Flag register in 8085 Microprocessor.**

- The ALU includes five flip-flops, which are set or reset after an operation according to data condition of the result in the accumulator and other registers.
- The five-status flag of INTEL 8085 MP are—
- Zero (Z), Carry (CY), Sign (S), Parity (P) and Auxiliary Carry (AC) flags.
- Their bit positions in the flag register are shown in Fig.
- The microprocessor uses these flags to test data conditions.



6-**Define Sack Pointer.**

- The stack pointer is also a 16-bit register, used as a memory pointer.
- It points to a memory location in R/W memory, called stack.
- The beginning of the stack is defined by loading 16- bit address in the stack pointer.
- **Stack**-The Stack is a sequence of memory location set by a programmer to store different element. So, it is known as Storage device.
- Stack works by LIFO (Last in First out) Operation.

7- How many interrupt signal are present in 8085 Microprocessor.

- ✓ There are 5 Interrupt signal i.e TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR.

8- How many Instructions set are present in 8085 Microprocessor.

- ✓ Instruction sets are instruction codes to perform some task. It is classified into five categories.

1-Control Instruction

2-Logical Instruction

3-Branching Instruction

4-Arithmatic Instruction

5- Data transfer Instruction

9- Define Addressing mode. How many Addressing mode are present in 8085.

- ✓ The process of specifying the data to be operated on by the instruction is called addressing.
- ✓ The various formats for specifying operands are called addressing modes.
- ✓ The 8085 has thefollowing five types of addressing modes.
  - o Immediate addressing
  - o Memory direct addressing
  - o Register direct addressing
  - o Indirect addressing
  - o Implicit addressing

10- Define T-state.

- ✓ One time period of frequency of microprocessor is called t-state.
- ✓ A t-state is measured from the falling edge of one clock pulse to the falling edge of the next clock pulse.
- ✓ Fetch cycle takes four t-states and execution cycle takes three t-states

Long Question:

1- Explain the Architecture of 8085 Microprocessor.

2- Draw the pin Diagram of 8085 and explain each pin,

3- Differenciate between one byte, two-byte, three-byte instruction with Example.

5-Draw the Timing diagram of MVI instruction.

6-Write the Assembly language program for Addition of two 8-bit number.

CHAPTER NUMBER -5

INTERFACING AND SUPPORT CHIPS

LEARING OBJECTIVES:

5.1. Basic interfacing concepts, memory mapping and I/O mapping.
5.2. Functional Block Diagram and description of each block of programmable peripheral interface intel 8255.
5.3. Application using 8255 :Seven Segment LED Display, Square Wave Generator, Traffic Light Control.

5.1-Basic interfacing concepts, memory mapping and I/O mapping

- The programs and data that are executed by the microprocessor have to be stored inROM/EPROM and RAM, which are basically semiconductor memory chips.
- The programs and data that are stored in ROM/EPROM are not erased even when power supply to the chip is removed. Hence, they are called non-volatile memory.
- They can be used to store permanent programs. In a RAM, stored programs and data are erased when the power supply to the chip is removed.
- Hence, RAM is called volatile memory. RAM can be used to store programs and data that include, programs written during software development for a microprocessor based system, program written when one is learning assembly language programming and data enter while testing these programs. Input and output devices, which are interfaced with 8085, are essential in any microprocessor based system.
- They can be interfaced using two schemes: I/O mapped I/O and memory-mapped I/O. In the I/O mapped I/O scheme, the I/O devices are treated differently from memory. In the memory-mapped I/O scheme, each I/O device is assumed to be a memory location.

Memory mapping and I/O mapping

- Like the memory locations 8085 microprocessor gets addressed by the processor which are called memory-mapped Input Output ports. There is a set of instructions for this memory-mapped I/O operations.
- Register A is an 8-bit register used in 8085 to perform arithmetic, logical, I/O & LOAD/STORE Operations Memory-mapped I/O uses the same address space to address both memory and I/O devices.

- The memory and registers of the I/O devices are mapped to (associated with) address values. So when an address is accessed by the CPU, it may refer to a portion of physical RAM, or it can instead refer to memory of the I/O device.
- Thus, the CPU instructions used to access the memory can also be used for accessing devices. Each I/O device monitors the CPU's address bus and responds to any CPU access of an address assigned to that device, connecting the data bus to the desired device's hardware register.
- To accommodate the I/O devices, areas of the addresses used by the CPU must be reserved for I/O and must not be available for normal physical memory.

IO/M	RD’	WR’	Operation
0	0	1	8085 reads data from memory
0	1	0	8085 writes data into memory

- The reservation may be permanent, or temporary (as achieved via bank switching).

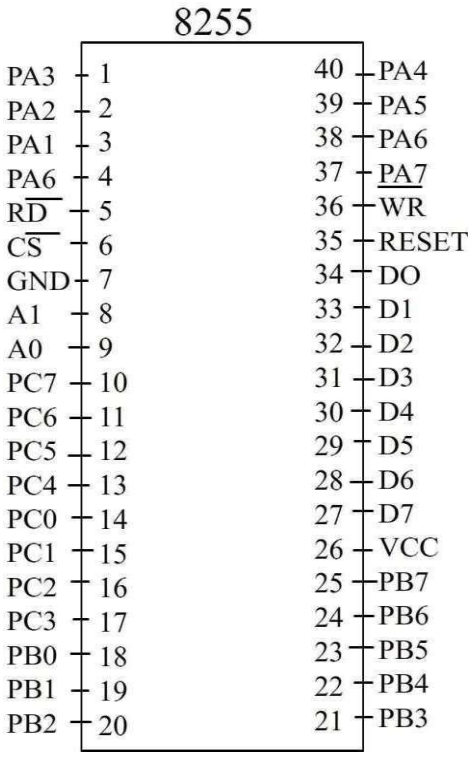
- The 8085 initiates set of signals such as IO/M , RD' and WR' when it wants to read from and write into memory. Similarly, each memory chip has signals such as CE or CS' (chip enable or chip select), OE or RD' (output enable or read) and WE or WR (write enable or write) associated with it.
- When the 8085 wants to read from and write into memory, it activates IO/M , RD' and WR' signals as shown in Table.

- Status of IO/M , RD' and WR' signals during memory read and write operations

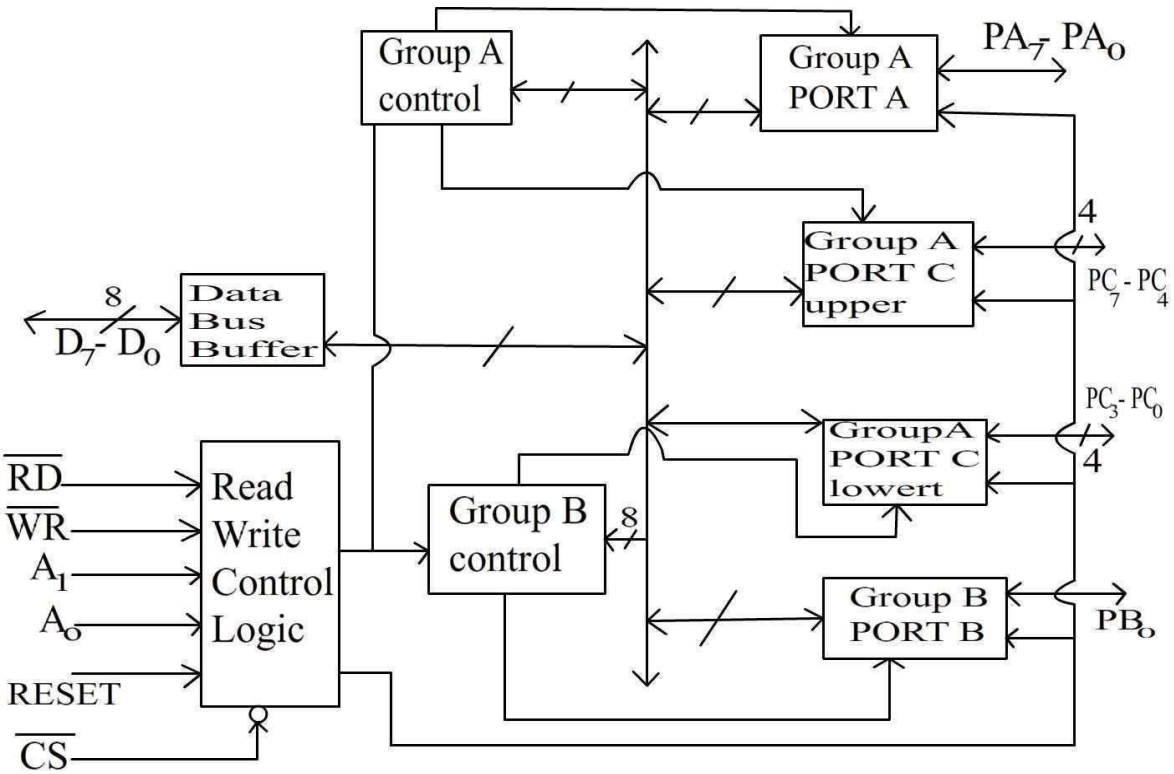
- Using IO/M , RD' and WR' signals, two control signals MEMR (memory read) and MEMW (memory write) are generated.

5.2- Functional Block Diagram and description of each block of programmable peripheral interface intel 8255

- The 8255A is a general purpose programmable I/O device designed for use with Intel microprocessors. It consists of three 8-bit bidirectional I/O ports (24I/O lines) that can be configured tomeet different system I/O needs. The three ports are PORT A, PORT B & PORT C.
- Port A contains one 8-bit output latch/buffer and one 8-bit input buffer. Port B is same as PORT A or PORT B. However, PORT C can be split into two parts PORT C lower (PC0-PC3) and PORT C upper (PC7-PC4) by the control word.
- The three ports are divided in two groups Group A (PORT A and upper PORT C) Group B (PORT B and lower PORT C). The two groups can be programmed in three different modes.
- In the first mode (mode 0), each group may be programmed in either input mode or output mode (PORT A, PORT B, PORT C lower, PORT C upper). I
- n mode 1, the second's mode, each group may be programmed to have 8-lines of input or output (PORTA or PORT B) of the remaining 4-lines (PORT C lower or PORT C upper) 3-lines are used for hand shaking and interrupt control signals.
- The third mode of operation (mode 2) is a bidirectional bus mode which uses 8-line (PORT A only for a bidirectional bus and five lines (PORT C upper 4 lines and borrowing one from other group) for handshaking.
- The 8255 is contained in a 40-pin package, whose pin out is shown below:


8255
PA3 1
PA2 2
PA1 3
PA6 4
RD 5
CS 6
GND 7
A1 8
A0 9
PC7 10
PC6 11
PC5 12
PC4 13
PC0 14
PC1 15
PC2 16
PC3 17
PB0 18
PB1 19
PB2 20
40 PA4
39 PA5
38 PA6
37 PA7
36 WR
35 RESET
34 DO
33 D1
32 D2
31 D3
30 D4
29 D5
28 D6
27 D7
26 VCC
25 PB7
24 PB6
23 PB5
22 PB4
21 PB3

Functional Block Diagram :



Data Bus Buffer:

A_1	A_0		WR'	CS'	Input operation	
0	0	0	1	0	PORT A	Data bus
0	1	0	1	0	PORT B	Data bus
1	0	0	1	0	PORT C	Data bus
0	0	1	0	0	Output operation Data bus	PORT A
0	1	1	0	0	Data bus	PORT B
1	0	1	0	0	Data bus	PORT C
1	1	1	0	0	Data bus	control

It is a tri-state 8-bit buffer used to interface the chip 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. The data lines are connected to B D B of p.

Read/Write and logic control:

The function of this block is to control the internal operation of the device and to control the transfer of data and control or status words. It accepts inputs from the CPU address and control buses a nd in turn issues command to both the control groups.

Chip Select:

A low on this input selects the chip and enables the communication between the 8255 A & the CPU. It is connected to the output of address decode circuitry to select the device when it RD ' = (Read). A low on this input enables the 8255 to send the data or status information to the CPU on the data bus.

WR' (Write):

A low on this input pin enables the CPU to write data or control words into the 8255 A.

A1, A0 port select:

These input signals, in conjunction with the RD' and WR' inputs, control the selection of one of the three ports or the control word registers. They are normally connected to the least significant bits of the address bus (A0 and A1).

- **Following Truth Table Shows the Port Selection**

PORTs A, B and C:

The 8255A contains three 8-bit ports (A, B and C). All can be configured in a variety of Functional c haracteristics by the system software.

PORTA: One 8-bit data output latch/buffer and one 8-bit data input latch.
PORT B: One 8-bit data output latch/buffer and one 8-bit data input buffer.
PORT C: One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input).

- These ports can be divided into two 4-bit ports under the mode control.
- Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signals inputs in conjunction with ports A and B.

Group A & Group B control:

- The functional configuration of each port is programmed by the system software.
- The control words outputted by the CPU configure the associated ports of the each of the two groups. Each control block a ccepts command from Read/Write content logic receives control words from the internal data bus and issues proper commands to its associated ports.
- Control Group A – Port A & Port C upper.
- Control Group B – Port B & Port C lower.
- The control word register can only be written into No read operation if the control word register is allowed.

Operational Description:

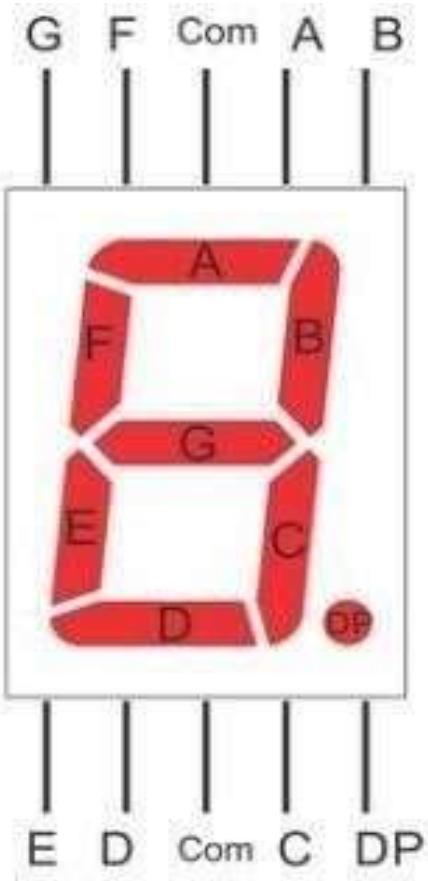
Mode selection:

There are three basic modes of operation that can be selected by the system software.
- Mode 0: Basic Input/output Mode 1: Strobes Input/output Mode 2: Bi-direction bus.

When the reset input goes HIGH all poets are set to mode'0' as input which means all 24 lines are in high impedance state and can be used as normal input.

- After the reset is removed the 8255A remains in the input mode with no additional initialization.
- During the execution of the program any of the other modes may be selected using a single output instruction.
- The modes for PORT A & PORT B can be separately defined, while PORT C is divided into two portions as required by the PORT A and PORT B definitions.
- The ports are thus divided into two groups Group A & Group B.
- All the output register, including the status flip-flop will be reset whenever the mode is changed.
- Modes of the two group may be combined for any desired I/O o peration, e.g. Group A in mode '1' and group B in mode '0'.

5.3-Application using 8255-7-segment LED display, square wave generator,

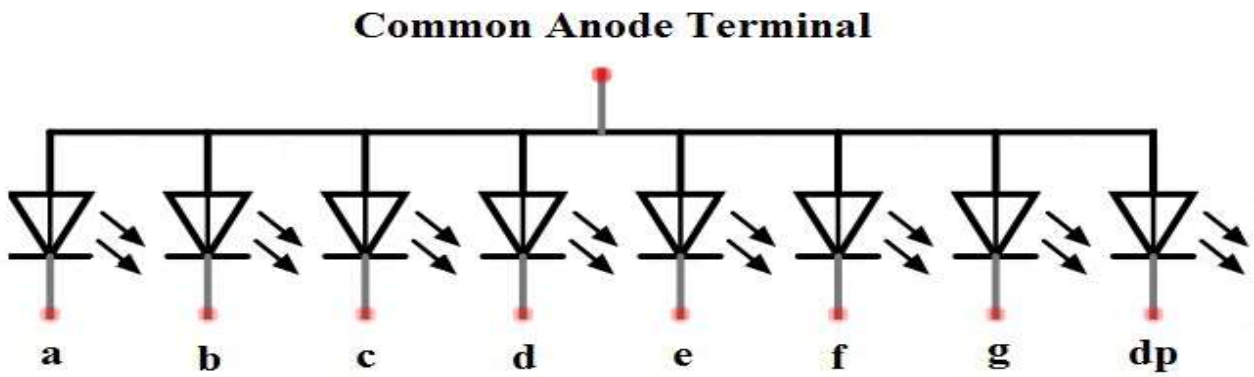


traffic light controller

Interfacing 7-segment display

- A seven segment display module is an electronic device used to display digital numbers and it is made up of seven LED segments.
- Because of the small size of the LEDs, it is really easy for a number of them to be connected together to make a unit like seven segment display.
- In the seven segment display module, seven LED s are arranged in a rectangle.
- Sometimes, an additional LED is seen in a seven segment. display unit which is meant for displaying a decimal point.
- Each LED segment has one of its pins brought out of the rectangular package.
- Other pins are connected together to a common terminal. Seven segment displays can only display 0 to 9 numbers.
- These seven LEDs indicate seven segments of the numbers and a dot point.
- Seven segment displays are seen associated with a great number of devices such as clocks,digital home appliances, signal boards on roads etc.

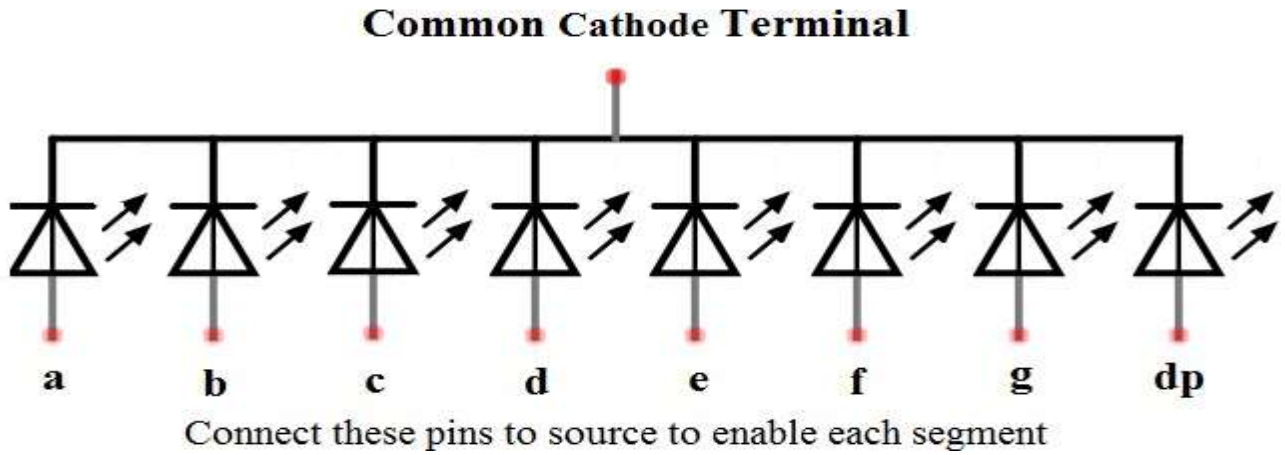
Types of Seven segment displays



- Seven segment displays come up with two different configurations. They are the common anode and a common cathode. One pin each from each segment is connected to a common terminal.
- According to the pins which are connected to the common terminal, the seven segment display is categorized as a **common anode** and **common cathode**.

Common Cathode 7-segment display

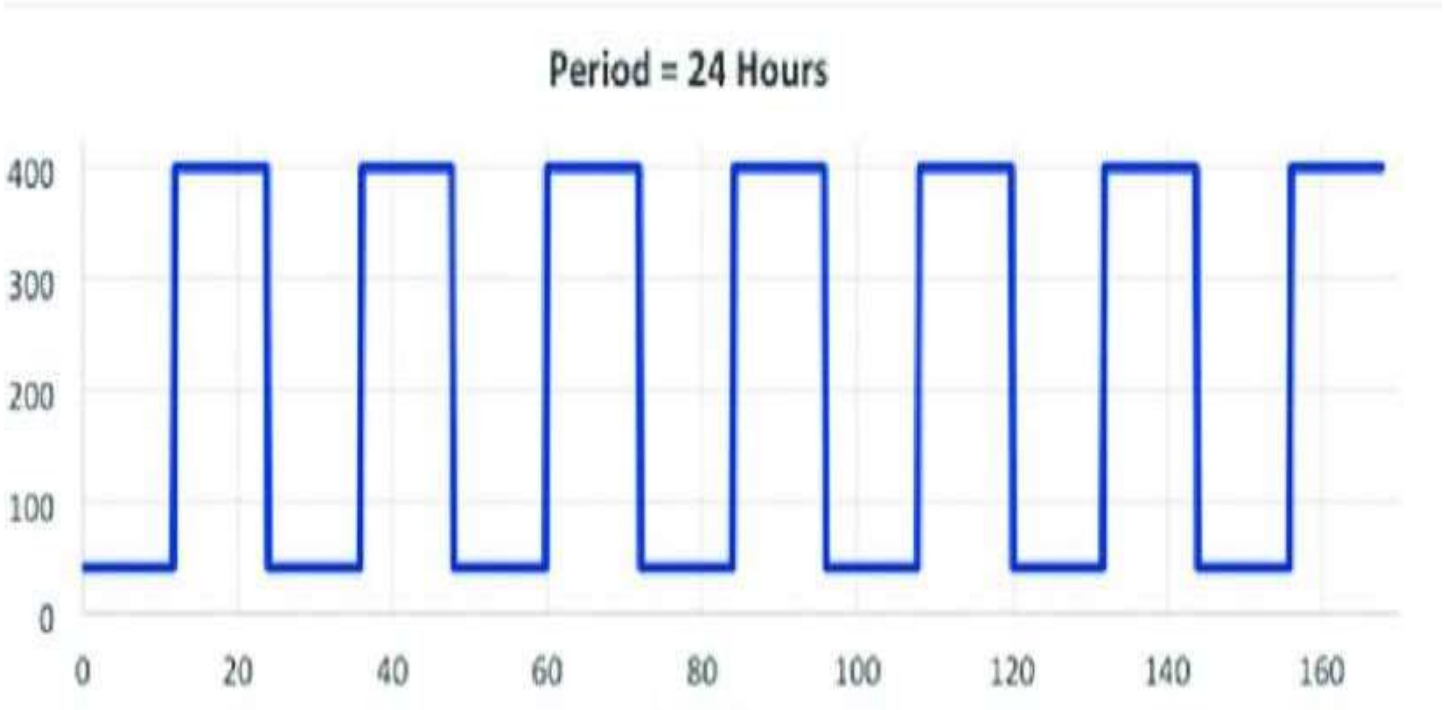
- As the name indicates, its cathode is connected to a common terminal. Below is the schematic diagram to indicate its common cathode structure.
- It should be connected to the ground while operating the display. If ahigh voltage is given to the anode, then it will turn on the corresponding segment.



Common Anode 7-segment display

- In this type, the anode is common.
- It should be connected to a high voltage (to the supply through a resistor to limit current).
- In order to turn on a particular segment, a ground level voltage is given to the corresponding pin.
- Since logic circuits can sink more current than they can source, common anode connection is used most widely.

Generate square wave on all lines of 8255



Program to interface DAC using 8255 and generate square waveform

- The following is the assembly language using DAC to interface with 8255 and generate a square wave on CRO.
- Here in the code, we use two delay elements one for the rising part of the wave and the other delay element to reach zero i.e decrement.
- Certain value chosen is delayed or sustained for a time period to form the square wave. The two loops used in the program are iterated to repeat cycles of a square wave.

Code:

MOV DX,8807 : DX is loaded with control word register address of 8255
MOV AL,80 : Move 80 to Accumulator.
OUT DX,AL : Contents of AL are transferred to port A of 8255.
MOV DX,8801 : DX is loaded with Port A.
Address of 8255 Begin MOV AL,00.
OUT DX,AL ; Contents of AL are transferred to port A of 8255.
MOV CX,00FF: Delay1 Loop Delay1
MOV AL,FF OUT DX,AL : Contents of AL are transferred to port A of 8255
MOV CX,00FF : CX is loaded with 00FFH
Delay 2 Loop Delay2 : Repeat until CX=0JMP Begin ; Repeat the same

Design interface a Traffic light control system using 8255

DESCRIPTION

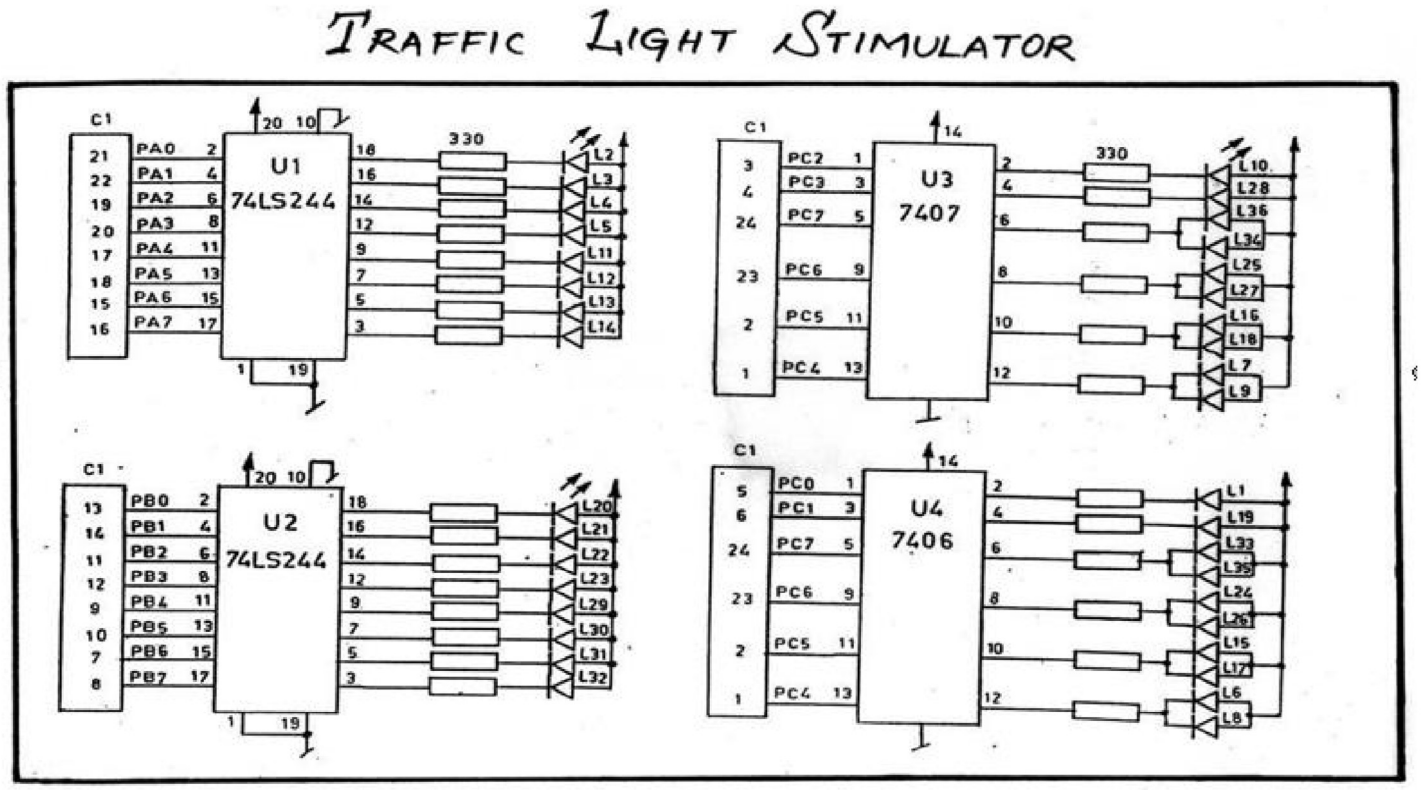
Combination of Red, Amber and Green LEDs are provided to indicate Halt, Wait and Go states for vehicles.

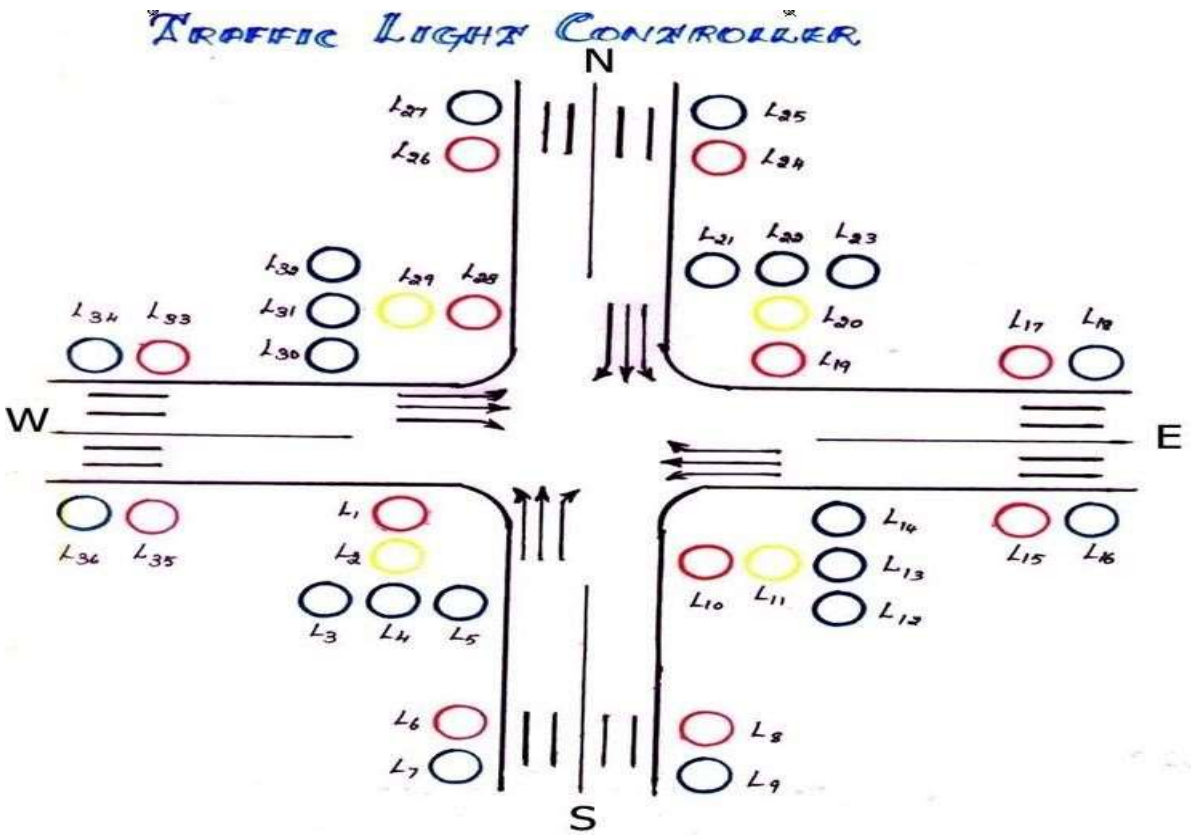
- ✓ Combination of Red and Green LEDs are provided for pedestrian crossing. 36 LEDs are arranged in the form of an intersection.
- ✓ At the left corner of each road, a group of 5 LEDs (Red, Amber and Green) are arranged in the form of a T section to control the traffic of that road.
- ✓ Each road is named as North N, South S, East E and West W. L1, L10, L19 and L28 (Red) are for stop signal for the vehicles on the road N, S, W and E respectively. L2, L11, L20 and L29 (Amber) indicate wait state for the vehicles on the road N, S, E and W respectively.
- ✓ L3,L4 and L5 (Green) are for left, straight and right turn for the vehicles on the road S. Similarly L12 - L13 - L14 , L23 - L22 - L21 and L32 - L31 - L30 simulates same function for the roads E, N & W respectively.
- ✓ A total of 16 LEDs (2 Red & 2 Green at each road) are provided for pedestrian crossing. L7 - L9, L16 - L18, L25 - L27 & L34 - L36 (Green) when on allows pedestrians to cross and L6 - L8, L15 - L17, L24 - L26 & L33 - L35 (Red) when on alarms the pedestrians to wait.
- ✓ To minimize the hardware pedestrians indicator LEDs (both Green and Red) are connected to some port lines (PC4 to PC7) with Red inverted.
- ✓ Red LED's L10 and L28are connected to port lines PC2 to PC3 while L1 and L19 are connected to lines PC0 and PC1 after inversion. All other LEDs (Amber and Green) are connected to Port A and port B.

INSTALLATION PROCEDURE

SDA_85M to NIFC_11 interface connection details:

- ✓ Connect p3 on 85M to the connector C1 on the interface using a 26 core FRC.
- ✓ Care should be taken such that, pin1 of P3 on the kit coincides with pin1 of cable [Observe the notch on the cable connector]
- ✓ Power connection: Connect +5v, GND to the interface.
- ✓ Color codes of power connection on the interface +5v - Orange, Blue,White GND – Black .
- ✓ Enter the Program.





ADDRESS	LABEL	MNEMONICS	OPCODE/OPERAND
C000		**MVI A,80H**	3E 80
C002		**OUT CWR**	D3 DB
C004	**REPEAT**	**MVI E,03H**	06 03
C006		**LXI H,C100H**	21 00 C1
C009	**NEXTSTAT**	**MOV A,M**	7E
C00A		**OUT PORTA**	D3 D8
C00C		**INX H**	23
C00D		**MOV A,M**	7E
C00E		**OUT PORTB**	D3 D9
C010		**INX H**	23
C011		**MOV A,M**	7E
C012		**OUT PORTC**	D3 DA
C014		**CALL DELAY**	CD 1F C0
C017		**INX H**	23
C018		**DCR E**	05
C019		**JNZ NEXTSTAT**	C2 09 C0
C01C		**JMP REPEAT**	C3 04 C0
C01F	**DELAY**	**LXI D,3000H**	11 00 30
C022	**L2**	**MVI C,FFH**	0E FF
C024	**L1**	**DCR C**	0D
C025		**JNZ L1**	C2 24 C0

Possible Short Type Questions with Answers

1. **Define interfacing .**

Ans- In computing, an **interface** is a shared boundary across which two or more separate components of a computer system exchange information. The exchange can be between software, computer hardware, peripheral devices, humans, and combinations of these.

2. **Define memory mapping and I/O mapping.**

Ans- Memory-mapped I/O (MMIO) and I/O mapped I/O (PMIO) are two complementary methods of performing input/output (I/O) between the central processing unit (CPU) and peripheral devices in a computer. An alternative approach is using dedicated I/O processors, commonly known as channels on mainframe computers, which execute their own instructions.

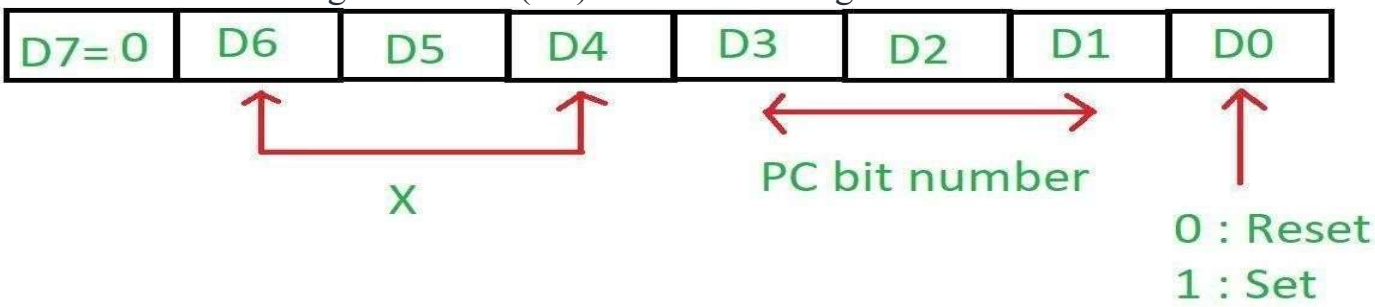
3. **Define EPROM.**

Ans- EPROM, in full erasable programmable read-only memory, Form of computer memory that **does** not lose its content when the power supply is cut off and that **can** be erased and reused.

4. **State different modes of 8255. (w-20)**

Ans- There are 2 modes in 8255 microprocessor :

a) **Bit set reset (BSR) mode –** This mode is used to set or reset the bits of port C only, and selectedwhen the most significant bit (D7) in the control register is 0.



b) **Input/output mode (I/O) –** This mode is selected when the most significant bit (D7) in the control register is 1.

Mode 0 – Simple or basic I/O mode:

Port A, B and C can work either as input function or as output function. The outputs are latched but the inputs are not latched. It has interrupt handling capability.

Mode 1 – Handshake or strobbed I/O:

In this either port A or B can work and port C bits are used to provide handshaking. The outputs as well as inputs are latched. It has interrupt handling capability. Before actual data transfer there is transmission of signal to match speed of CPU and printer.

SL. NO.	NAME OF BOOK/SOURCE	NAME OF THE AUTHOR	NAME OF THE PUBLICATION
1	Fundamentals of Digital Electronics	Ananda Kumar	PHI
2	Digital Electronics-Principal & Application	S.K. Mondal	TMH
3	Digital Electronics	B.R.Gupta & V.Singhal	S.K.Kateria
4	Digital Electronics	P.Raja	SciTech
5	Microprocessor Architecture Programming And Application with 8085	R.S.Goankar	Peneram
6	Fundamentals of Microprocessor And Microcomputers	B.Ram	Dhanpat Rai
7	Microprocessor And Interfacing	Sunetra Choudhury & S.P.Chowdhury	SciTech

Q 5- Define ADC.

Ans- In electronics, an analog-to-digital converter (ADC, A/D, or A-to-D) is a system that converts an analog signal, such as a sound picked up by a microphone or light entering a digital camera, into a digital signal. An ADC may also provide an isolated measurement such as an electronic device that converts an input analog voltage or current to a digital number representing the magnitude of the voltage or current. Typically the digital output is a two's complement binary number that is proportional to the input, but there are other possibilities.

Possible Long Type Questions

1. Draw the block diagram of 8255 . Explain the pin description of 8255.
2. Generate square wave using 8255.
3. Design and interface traffic light control using 8255**.(w-20)**
4. Design stepper motor using 8255.

REFERENCE :

Table: