LECTURE NOTES

On

DIGITAL ELECTRONICS AND MICROPROCESSOR

On

5TH Semester of Electrical Engineering Branches

Prepared by
CHINMAYEE PANIGRAHI (PTGF Electrical)

24/09/2020

Binary Addition

There are four steps in binary addition.

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 0$ (carry 1 to the next significant bit)

eg. $(1\,0\,0\,0\,1\,0\,0\,1)_2 + (1\,0\,0\,1\,0\,1\,0\,1)_2 = (\quad)_2$

$$\begin{array}{r} \overset{1}{} \\ 1\,0\,0\,0\,1\,0\,0\,1 \\ +\ 1\,0\,0\,1\,0\,1\,0\,1 \\ \hline 1\,0\,0\,1\,1\,1\,1\,0 \end{array} \qquad \begin{array}{l} \longrightarrow (137)_{10} \\ \longrightarrow (149)_{10} \\ \hline \longrightarrow (286)_{10} \end{array}$$

eg. $(1\,1\,0\,1\,1\,1)_2 + (1\,0\,1\,0\,1\,1)_2 = (\quad)_2$

$$\begin{array}{r} \overset{1}{}\overset{1}{}\overset{1}{}\overset{1}{}\overset{1}{} \\ 1\,1\,0\,1\,1\,1 \\ +\ 1\,0\,1\,0\,1\,1 \\ \hline 1\,1\,0\,0\,0\,1\,0 \end{array} \qquad \begin{array}{l} (55)_{10} \\ +\ (43)_{10} \\ \hline (98)_{10} \end{array}$$

Binary Subtraction

$0 - 0 = 0$

$0 - 1 = 1$, borrow 1 from next more significant bit.

$1 - 0 = 1$

$1 - 1 = 0$

eg $(1\,0\,1\,1\,1\,0\,1\,1)_2 - (1\,0\,0\,1\,0\,0\,1)_2 = (\quad)_2$

$$\begin{array}{r} 1\,0\,1\,1\,1\,0\,1\,1 \\ -\ 1\,0\,0\,1\,0\,0\,1 \\ \hline 1\,1\,1\,0\,0\,1\,0 \end{array} \qquad \begin{array}{l} (187) \\ -(73)_{10} \\ \hline (114)_{10} \end{array}$$

②

eg. $(10101010)_2 - (10100010)_2 = (\quad)_2$

$$\begin{array}{r} 10101010 \\ -\ 10100010 \\ \hline 00001000 \end{array} \qquad \begin{array}{r} (170)_{10} \\ -\ (162)_{10} \\ \hline (8)_{10} \end{array}$$

Binary Multiplication

$0 \times 0 = 0$

$1 \times 0 = 0$

$0 \times 1 = 0$

$1 \times 1 = 1$ (no carry or borrow)

eg. $(1001)_2 \times (101)_2 = (\quad)_2$

$$\begin{array}{r} 1001 \\ \times\ \ 101 \\ \hline 1001 \\ 0000 \\ 1001 \\ \hline 101101 \end{array} \qquad \begin{array}{r} 9_{10} \\ \times\ 5_{10} \\ \hline 45_{10} \end{array}$$

eg. $(1111)_2 \times (11)_2 = (\quad)_2$

$$\begin{array}{r} 1111 \\ \times\ \ 11 \\ \hline 1111 \\ 1111 \\ \hline 10101 \end{array} \qquad \begin{array}{r} 15_{10} \\ \times\ 3_{10} \\ \hline 45_{10} \end{array}$$

eg. $(1101)_2 \times (101)_2 = (\quad)_2$

$$\begin{array}{r} 1101 \\ \times\ \ 101 \\ \hline 1101 \\ 0000 \\ 1101 \\ \hline 1000001 \end{array} \qquad \begin{array}{r} 13_{10} \\ \times\ 5_{10} \\ \hline 65_{10} \end{array}$$

③

Binary Division

eg. $(11010)_2 \div (101)_2 = (\quad)_2$

```
      101 ) 11010 ( 101   <- Quotient
            101
            ---
             0011
              0
             ----
             00110
               101
             -----
             00001   <- Remainder
```

```
5 | 26 | 5
   25
   --
    1
```

eg. $(11101)_2 \div (11)_2 = (\quad)_2$

```
      11 ) 11101 ( 1001
           11
           --
            001
             0
            ---
             010
              0
             ---
              101
               11
              ---
               10
```

```
3 | 29 | 9
   27
   --
    2
```

Question:

① 00110 10 + 001100 = ?

② 00110 10 − 001100 = ?

③ 00110 10 × 001100 = ?

④ 101010 ÷ 000110 = ?

```
110 | 101010 | 111
    110↓
    ----
     1001
     -110
     ----
      0110
       110
      ----
       000
```

→ 28 + 22 = 50

```
  ①①
  11100
  10110
 ------
 110010
```

63 + 14 = 77

```
 ①①①①
  111111
  001110
 -------
 1001101
```

$$\begin{array}{r} 28 \\ -\ 22 \\ \hline 06 \end{array}$$

$$\begin{array}{r} 63 \\ -\ 14 \\ \hline 49 \end{array}$$

$$\begin{array}{r} 11100 \\ -\ 10110 \\ \hline 00110 \end{array}$$

$$\begin{array}{r} 1111111 \\ -\ 001110 \\ \hline 110001 \end{array}$$

$$\begin{array}{r} 15 \\ 7 \\ \hline 105 \end{array}$$

$$\begin{array}{r} 1111 \\ 111 \\ \hline 11111 \\ 1111 \\ \hline 1101001 \end{array}$$

$$\begin{array}{r} 7 \mid 127 \mid 18 \\ 7 \\ \hline 57 \\ 56 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 111 \mid 11111110 \mid 10010 \\ 111 \\ \hline 0001 \\ 0 \\ \hline 11 \\ 0 \\ \hline 111 \\ 111 \\ \hline 0001 \\ 0 \\ \hline 001 \end{array}$$

$$\begin{array}{r} 110 \mid 1010110 \mid 111 \\ 6 \\ -\ 110 \\ \hline 1801 \\ -\ 110 \\ \hline 0110 \\ 110 \\ \hline 000 \end{array}$$

25/09/2020

Data representation

It is used to communicate with digital computer.

Data is divided in two ways

→ Unsigned data

→ Signed data.

- **Unsigned data**

In this case all bits give value.

Signed data

In this case, MSB (Most Significant bit) (or) the leftmost bit gives the sign of the data and remaining all bits give value.

* When MSB = 0, data is +ve.

MSB = 1, data is -ve.

To represent signed data, there are three method.

1. Signed magnitude notation
2. Signed 1's complement notation
3. Signed 2's complement notation

Note:

* For +ve data, no complementing is needed as +ve data is represented directly
* Always -ve data is represented in respective complement notation.
* Complements are used in digital computers for performing subtraction operation.

1's complement notation

1's complement of a binary number is another binary number obtained by toggling (or) complementing all bits i.e. transforming the bit '0' to '1' and bit '1' to 0.

eg: 1. 1's complement of 7 (0111) is 1000.

0 1 1 1
↓ ↓ ↓ ↓
1 0 0 0

i.e. $\overline{0111} = 1000$

2. 1's complement of 12 (1100) is 0011.

1 1 0 0
↓ ↓ ↓ ↓
0 0 1 1

i.e $\overline{1100} = 0011$

2's complement notation

2's complement of a binary number is another binary number obtained by adding '1' to the 1's complement of the binary number.

2's complement = 1's complement + 1

eg-1 2's complement of 7 = 1's complement of 7 + 1

i.e. $\overline{0111} + 1 = 1000 + 1$

$= 1001$

→ 2's complement of 12 (1100) = $\overline{1100}$ + 1
= 0011 + 1
= 0100

Question

Find 1's & 2's complement of following numbers.

(1) 17 (2) 127 (3) 148 (5) 123

(5) 11101111 (6) 10101010 (7) 11011011

Subtraction operation using 2's complement

$$A - B = A + (-B)$$

Minuend — Subtrahend.

In computer, for subtraction operation 2's complement method is used i.e. subtraction is done in terms of addition.

Following steps are done.

1. At first, 2's complement of the subtrahend is found.
2. Then it is added to the minuend.
3. If the final carry over of the sum is 1, it is discarded, and the result is positive.
4. If there is no carry over, the two's complement of the sum will be the result and result is negative.

Example

(1) $110110 - 010110 = \_\_\_\_\_$.
A B

Step-1 2's complement of 010110 (Subtrahend)

= 1's complement of 010110 + 1

= 101001 + 1

= 101010.

Step-2

$$\begin{array}{r} 110110 \\ -\ 010110 \end{array} \simeq \begin{array}{r} 110110 \\ +\ 101010 \\ \hline 1\ 100000 \end{array}$$

1 → final carry over generated.

Step-3 As carry over is generated, discarding the carry.

Result is 100000 (+ve)

$$\begin{array}{r} 54 \\ -\ 22 \\ \hline +32 \end{array}$$

+ve data.

$[(110110)_2 = 54, (010110)_2 = 22]$

(2) $10110 - 11010 = \_\_\_\_\_$.
B

Step-1 2's complement of 11010

= 11010 + 1

= 00101 + 1

= 00110

Step-II

```
  (1)(1)
   1 0 1 1 0
 + 0 0 1 1 0
 -----------
   1 1 1 0 0
```

No carry over generated.

So the result = 2's complement of

$$= \overline{11100} + 1$$

$$= 00011 + 1$$

$$= 00100$$

-ve number.

$$\begin{array}{r} 22 \\ -26 \\ \hline -(4) \end{array}$$

-ve result.

Question

① $\begin{array}{r} 127 \\ -83 \\ \hline \end{array}$

② $\begin{array}{r} 43 \\ -54 \\ \hline \end{array}$

③ $\begin{array}{r} 63 \\ -43 \\ \hline \end{array}$

Perform subtraction operation of the following questions using 2's complement.

28/09/2020

Codes

Numerical code
eg. 8421
5421
2421
Binary
Gray
Excess-3

Alphanumeric code
eg. ASCII

Numerical Code

Weighted Code

Non-weighted code.

→ Codes in digital are broadly classified into two types.

(I) Numerical code.
(II) Alphanumerical code.

→ Numerical code is again divided in two groups

(I) Weighted code
(II) Nonweighted code.

Weighted Code.

Weighted binary codes are those binary codes in which each digit is assigned

a specific weight according to its position.

→ The sum of all digits multiplied by a weight gives the total amount being represented.

eg. 8421 (or) Binary coded decimal (BDC)
5421
2421

Nonweighted code

Nonweighted binary codes are not positionally weighted ie codes that are not assigned with any weight to each digit position.

eg. Gray code
Excess-3 code.

8421 code

8421 is a weighted code in which each decimal digit 0 to 9 is represented by a four bit code word.

→ The bit positions in each code word are assigned weights, 8, 4, 2 and 1 from left to right.

→ It is also called Binary Coded Decimal (BCD) code.

Decimal digit	8421 BCD code.			
	8	4	2	1
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

Excess-3 code (XS3)

→ It is a non-weighted code and selfcomplementary BCD code used to represent decimal number.

→ Excess-3 code of given BCDs self complement is equal to its 9's complement.

→ It is especially used in BCD subtraction

→ It is also called shifted binary (or) stibitz code.

→ For converting BCD to XS3, 3 (or) 0011 is added to corresponding BCD code.

eg. XS3 of 4 (or) 0100 is as 7.

$$\begin{array}{r} 0100 \\ +\,0011 \\ \hline 0111 \end{array}$$

Decimal	BCD 8 4 2 1	Excess 3 BCD + 0011
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

Gray code

→ It is also known as reflected binary code (or) gray code (or) unit distance code.

→ Always it gives one bit difference in successive combination.

→ It is used in K-map.

Binary to Gray code

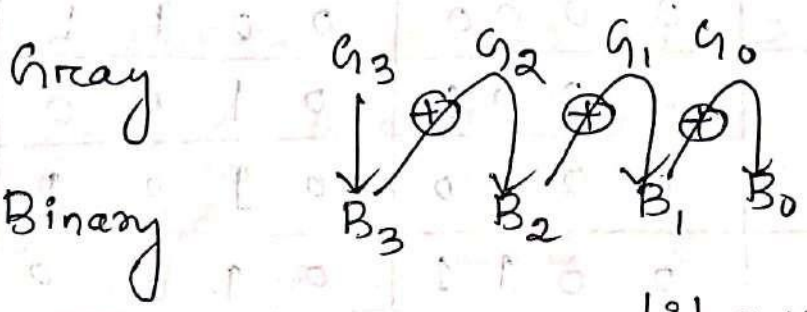
Binary → $B_3 \rightarrow \oplus \rightarrow B_2 \rightarrow \oplus \rightarrow B_1 \rightarrow \oplus \rightarrow B_0$

Gray → G_3 G_2 G_1 G_0

⊕ → Ex-OR i.e. when the bits are equal → '0'

when not equal → '1'

Gray to binary code.



⊕ → Ex-or. bit equal → '0'
bit unequal → '1'

Decimal	Binary	Gray code
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1

Questions

Q. Convert the following binary number to gray code.

(i) 1 1 0 1 1 1 1 0

(ii) 1 1 0 1 0 1 0 1

(iii) 1 0 1 0 1 1 0 1

(iv) 1 0 1 1 0 1 1 1

Q. Convert the following gray code to binary code.

(i) 1 0 1 0 1 1 0 1 (ii) 1 1 0 1 1 1 1 0

(iii) 1 0 1 1 1 1 1 (iv) 1 1 1 0 1 1 0 0

Importance of parity bit

→ A parity bit also known as a check bit is a single bit that can be appended to a binary data string.

→ It is set to either '1' or '0' to make the total number of '1' bit either even ("even parity") or odd ("odd parity").

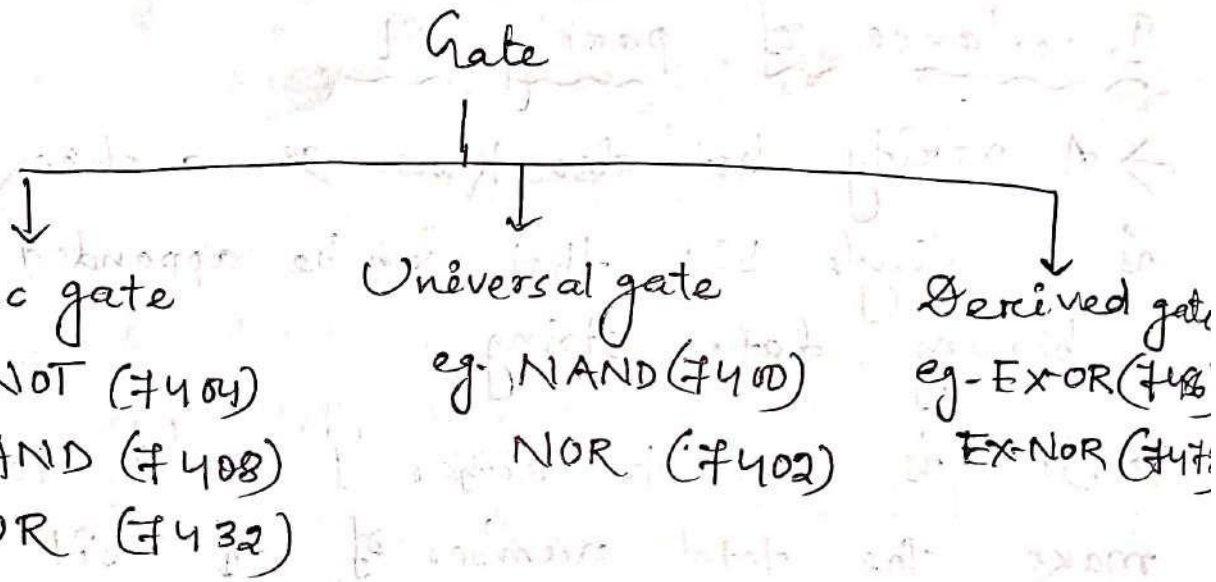
→ The purpose of parity bit is to check the error at receiving end.

P_{odd}	A	B	C	P_{even}
1	0	0	0	0
0	0	0	1	1
0	0	1	0	1
1	0	1	1	0
0	1	0	0	1
1	1	0	1	0
1	1	1	0	0
0	1	1	1	1

29/09/2020

Logic Gate

It is an electronic circuit which permits one or more I/ps and delivers single o/p.



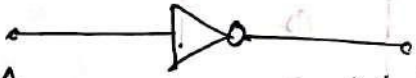
* EX-OR & EX-NOR gates are available with 2 I/ps only and these gates are 2 I/p controlled gate.

* AND, OR, NAND, NOR gates can have two or more I/ps and all these are single I/p controlled gate.

NOT gate

It is a single I/p & single o/p gate which o/p is complement of I/p.

Symbol



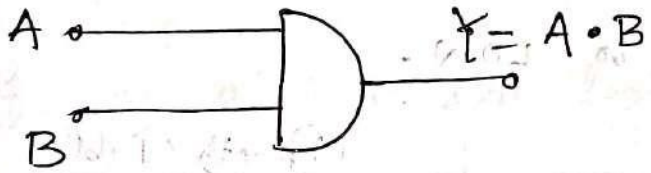
Truth Table

A	Y = Ā
0	1
1	0

AND gate

It is a multiinput logic gate in which o/p becomes '1' or 'HIGH' when all the I/ps are '1' (or) 'HIGH', otherwise o/p is zero (or) 'LOW'.

Symbol



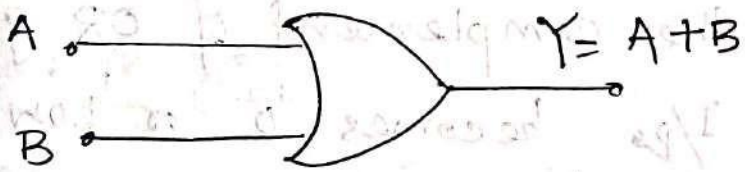
Truth Table

I/P A	I/P B	O/P Y = A • B
0	0	0
0	1	0
1	0	0
1	1	1

OR gate

It is a multiinput logic gate in which o/p becomes '1' or 'HIGH' when any one of the I/Ps becomes '1' (or) 'HIGH', otherwise o/p is '0' or 'LOW'.

Symbol



Truth Table

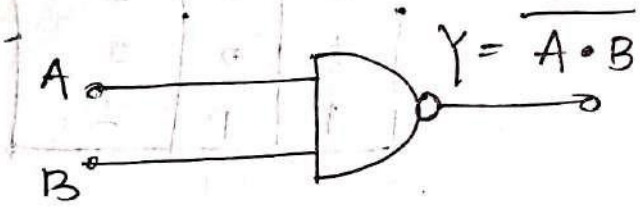
I/P A	I/P B	O/P Y = A + B
0	0	0
0	1	1
1	0	1
1	1	1

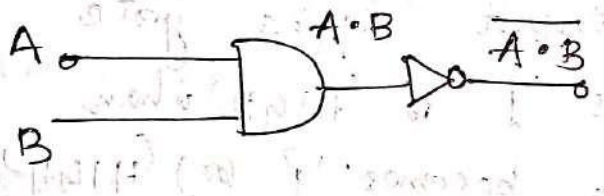
NAND gate

→ It is an universal gate.

→ It performs the complement of AND gate. i.e. if one of the I/P becomes 'Zero' or 'LOW', O/P becomes '1' or 'HIGH', or when all the I/Ps are 'HIGH' or '1', O/P becomes '0' or LOW.

Symbol





Truth Table

I/P A	I/P B	O/P $Y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

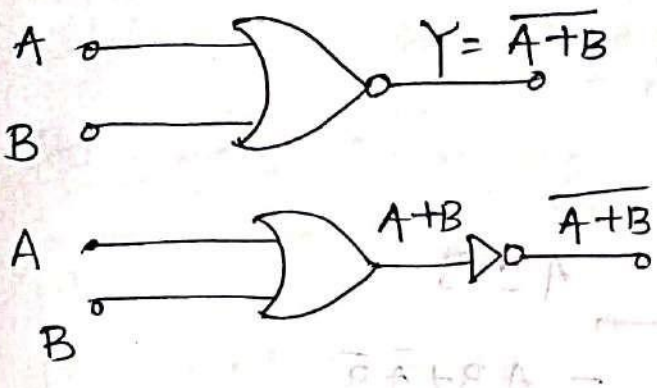
NOR gate

→ It is an universal gate.

→ It performs the complement of OR gate.

→ If all the I/Ps becomes '0' or 'LOW', the O/P is '1' or 'HIGH', otherwise the O/P becomes '0' or 'LOW'.

Symbol



Truth Table

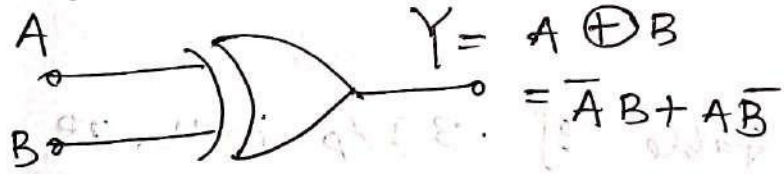
I/P A	I/P B	O/P Y = A+B
0	0	1
0	1	0
1	0	0
1	1	0

Ex-OR gate

→ It is a two I/p logic gate in which O/p is '0' or 'LOW' when its I/ps are equal ie. coincidence. So it is called Non-coincidence gate'

→ O/p becomes '1' or 'HIGH' when I/ps are unequal.

Symbol.



Truth Table

I/P A	I/P B	O/P Y = A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

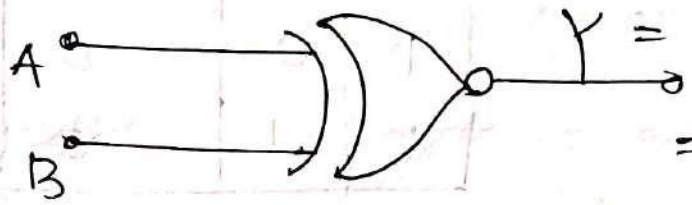
Ex-NOR gate

→ It is a two I/p logic gate.

→ The O/p becomes '1' or 'HIGH' when its I/ps are equal ie. coincidence. So it is called coincidence gate.

→ O/p becomes '0' when I/ps are not equal.

Symbol



Truth Table

I/P		O/P
A	B	Y = A⊙B
0	0	1
0	1	0
1	0	0
1	1	1

Question

Draw the truth table of 3 I/P AND, OR, NAND & NOR gate.

3/09/2020

Doubt clearing class.

01/10/2020

Boolean Algebra

→ It works with binary variable.

Various theorems of boolean algebra

(1) a) $x + 0 = x$

b) $x \cdot 1 = x$

Prove a) Let $x = 0$. $x = 1$

$0 + 0 = 0$ $1 + 0 = 1$

b) Let $x = 0$, $x = 1$

$0 \cdot 1 = 0$, $1 \cdot 1 = 1$

(2) a) $x + x' = 1$ (or) $x + \bar{x} = 1$

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

Prove b) Let $x = 0$, $x = 1$

$0 \cdot \bar{0} = 0 \cdot 1 = 0$; $1 \cdot \bar{1} = 1 \cdot 0 = 0$

a) Let $x = 0$ $x = 1$

$0 + \bar{0} = 0 + 1 = 1$ $1 + \bar{1} = 1 + 0 = 1$

(3) a) $x + x = x$

b) $x \cdot x = x$

Prove a) $x + x = (x + x) \cdot 1$

$= (x + x)(x + x')$ (from 2(a) $x + \bar{x} = 1$)

$= x \cdot x + x \cdot x' + x \cdot x + x \cdot x'$

$= x \cdot + x \cdot x'$ ($x + x = x$)

$= x \cdot + 0$ ($x \cdot x' = 0$, 2(b))

$= x$

3(b). $x \cdot x = xx + 0$ ($x+0=x$, 1a)

$= xx + x\bar{x}$ ($x \cdot \bar{x} = 0$ 2b)

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

$= x \cdot 1$ ($x+\bar{x}=1$ 2a)

$= x$

(4)

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

Prove a) $x+1 = 1 \cdot (x+1)$ ($\because 1 \cdot x = x$, 1b)

$= (x+\bar{x})(x+1)$ ($x+\bar{x}=1$, 2a)

$= x + \bar{x} \cdot 1$ ($\because (x+y)(x+z) = x+yz$)

$= x + \bar{x}$ ($1 \cdot \bar{x} = \bar{x}$)

$= 1$ ($\because x+\bar{x}=1$, 2a)

b) $x \cdot 0 = 0$.

Let $x = 0$, $0 \cdot 0 = 0$

$x = 1$, $1 \cdot 0 = 0$.

(5) $\overline{(\bar{x})} = x$

Prove

x	$\bar{x}$	$\overline{\bar{x}}$
0	1	0
1	0	1

(6) Commutative Law

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

⑦ Associative Law

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

b) $x(yz) = (xy)z$.

⑧ Distributive Law

a) $x(y+z) = xy+xz$

b) $x+yz = (x+y).(x+z)$

⑨ DeMorgan's Theorem

Theorem-1

The complement of sum of variables is equal to the product of complement of individual variables.

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

Prove

x	y	$\bar{x}$	$\bar{y}$	$x+y$	$\overline{x+y}$	$\bar{x} \cdot \bar{y}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

Theorem-2

The complement of product of variables is equal to the sum of complement of individual variable.

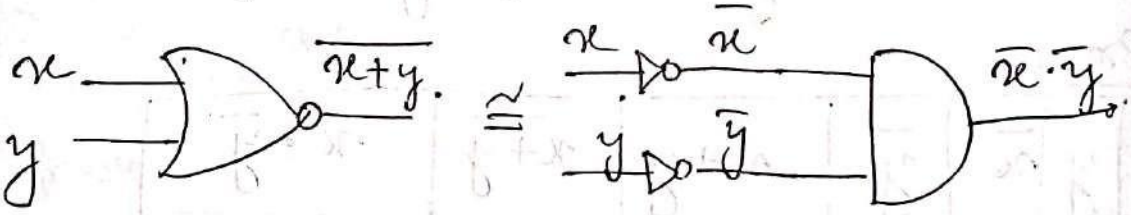
$$\overline{x \cdot y} = \bar{x} + \bar{y}$$

Prove

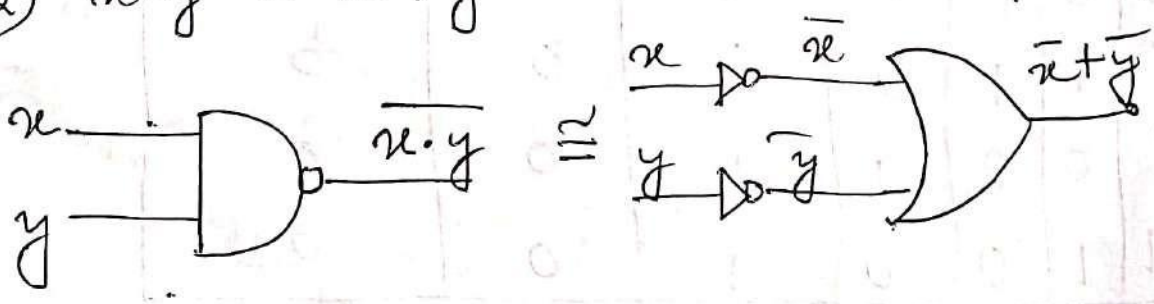
x	y	$\bar{x}$	$\bar{y}$	$x \cdot y$	$\overline{x \cdot y}$	$\bar{x} + \bar{y}$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

Gate level diagram

1) $\overline{x+y} = \bar{x} \cdot \bar{y}$



2) $\overline{x \cdot y} = \bar{x} + \bar{y}$



(10) Absorption Law

a) $x + xy = x$
b) $x(x+y) = x$

Prove

a) $x + xy = x \cdot 1 + xy$ (from 1(b))

$= x(1+y)$ (from 8(a))

$= x(y+1)$ (from 6(a))

$= x \cdot 1$ (from 4(a))

$= x$ (from 1(b))

b)

x	y	$x+y$	$x(x+y)$
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

Questions

Draw gate level diagram of following boolean expression using AOI (AND-OR-INVERT) gates.

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

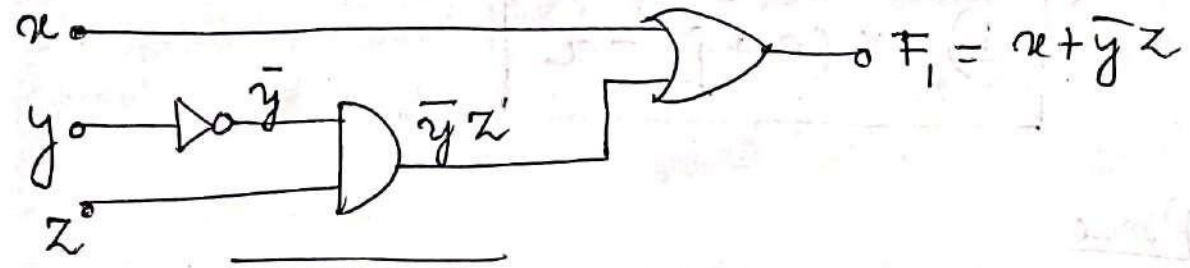
(b) $F_2 = \overline{(x + \bar{y}z)}$

(c) $F_3 = \bar{x}\bar{y}z + \bar{x}yz + x\bar{y}$

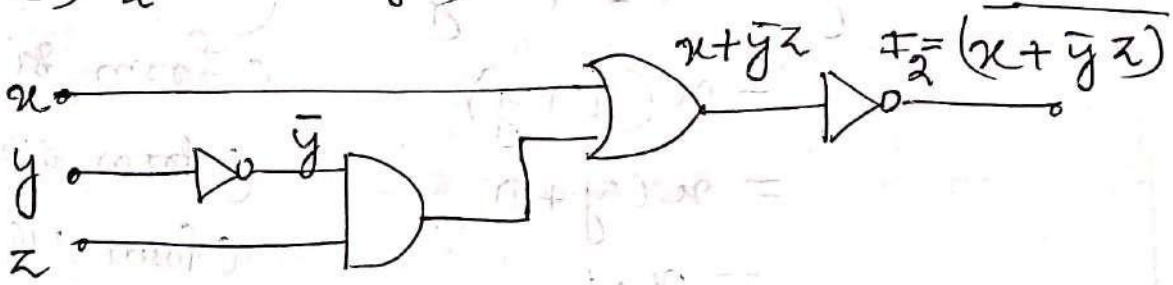
(d) $F_4 = x\bar{y} + \bar{x}z$

Solution

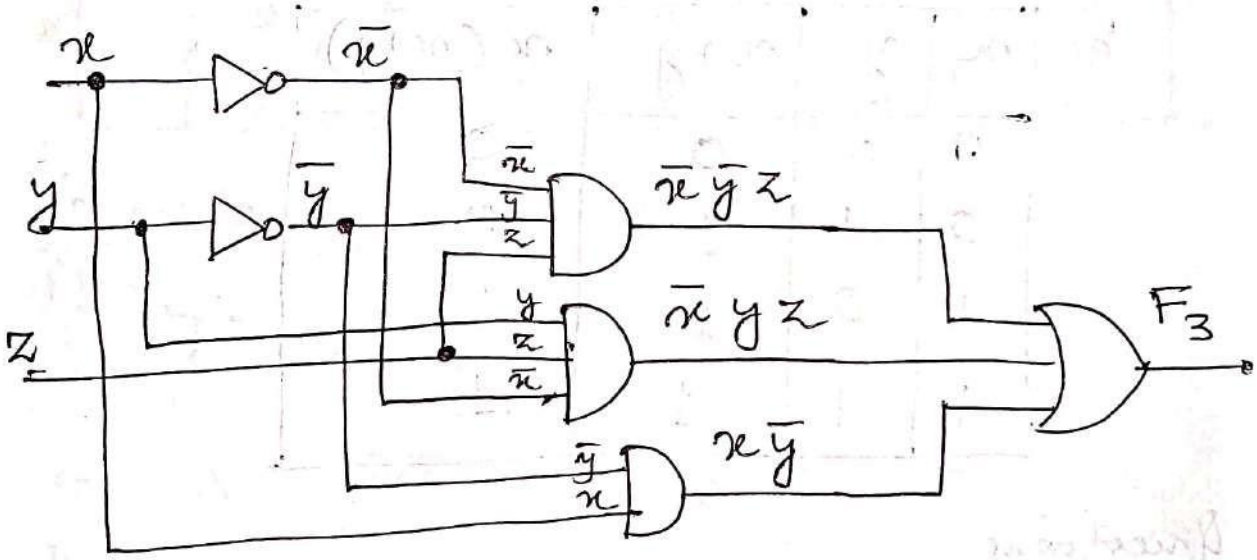
(a) $F_1 = x + \bar{y}z$



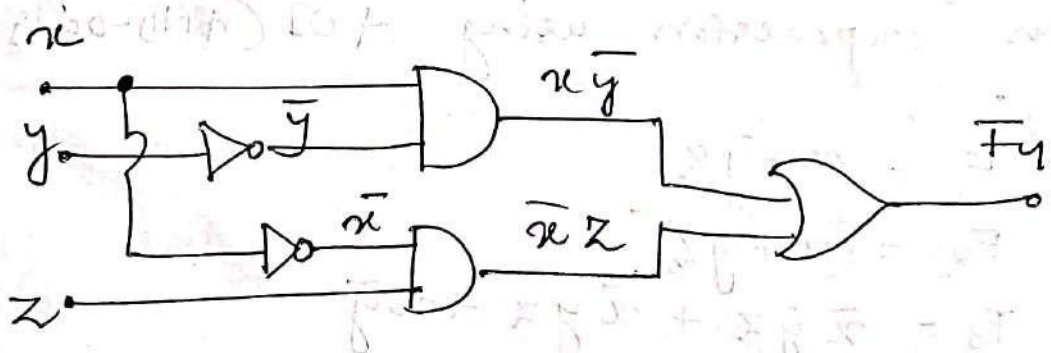
(b) $F_2 = \overline{(x + \bar{y}z)}$



(c) $F_3 = \bar{x}\bar{y}z + \bar{x}yz + x\bar{y}$



(d) $F_4 = x\bar{y} + \bar{x}z$



Question

Simplify the following boolean expressions using boolean laws.

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

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

(c) $(x+y)(x+\bar{y})$

(d) $xy+\bar{x}z+yz$

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

Solution

(a) $x(\bar{x}+y) = x\bar{x}+xy = 0+xy = xy$

(b) $x+\bar{x}y = (x+\bar{x})(x+y) = 1(x+y) = x+y$

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

$= x + x\bar{y} + xy$

$= x(1+\bar{y}+y)$

$= x\cdot 1$

$= x$

(d) $xy+\bar{x}z+yz$

$= xy+\bar{x}z+yz(x+\bar{x})$

$= xy+\bar{x}z+xyz+\bar{x}yz$

$= xy+xyz+\bar{x}z+\bar{x}yz$

$= xy(1+z)+\bar{x}z(1+y)$

$= xy+\bar{x}z$

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

$= (x+y)(\bar{x}+z)(y+z+x\bar{x})$ $(x\bar{x}=0)$

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

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

$= (x+y)(1+z)$

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

Questions

Solve the following Boolean expression to a minimum no. of variables and implement the reduced expression using Logic gates (AOI) where possible.

① $xyz + \bar{x}y + xy\bar{z}$

② $\overline{(A+B)}\,\overline{(\bar{A}+\bar{B})}$

③ $xy + x\bar{y} + \bar{x}y$

④ $ABC + \bar{A}B + AB\bar{C} + AC$

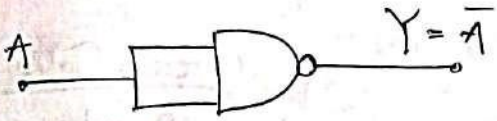
⑤ $\overline{(x+y)}\,(\bar{x}+\bar{y})$

⑥ $xy + x(wz + w\bar{z})$

Realization of other logic functions using NAND/NOR gates

Using NAND gate

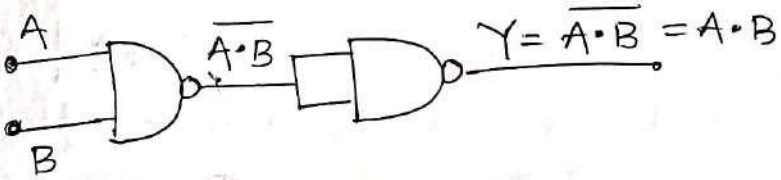
(1) NOT gate



Truth Table

A	$Y = \bar{A}$
0	1
1	0

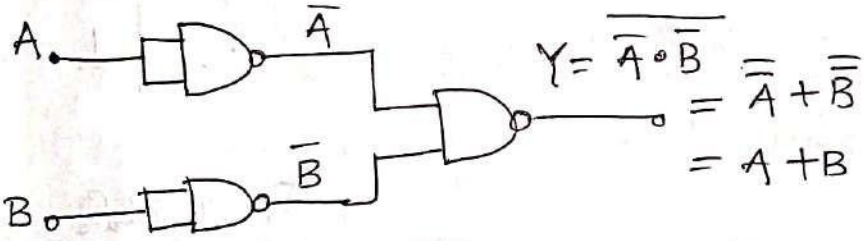
(2) AND gate



Truth Table

A	B	$Y = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

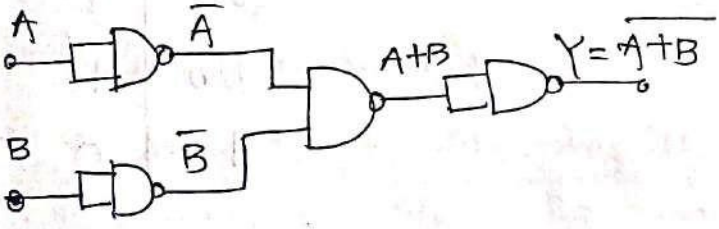
(3) OR gate



Truth Table

A	B	$Y = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

(4) NOR gate

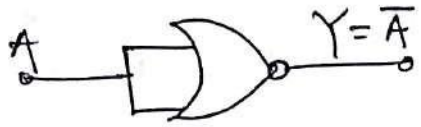


Truth Table

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

Using NOR gate

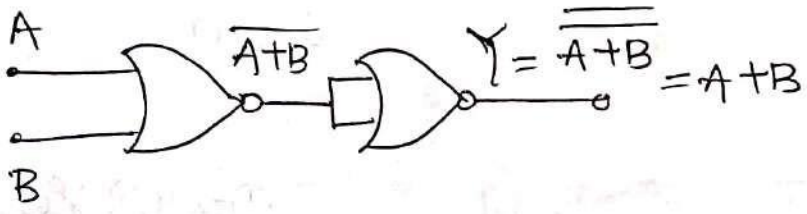
① NOT gate



Truth Table

A	Y = Ā
0	1
1	0

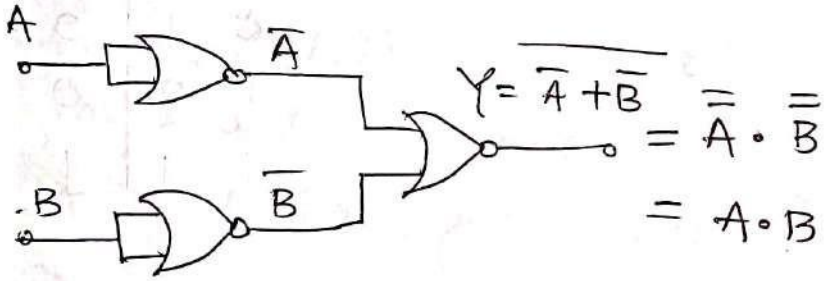
② OR gate



Truth Table

A	B	Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

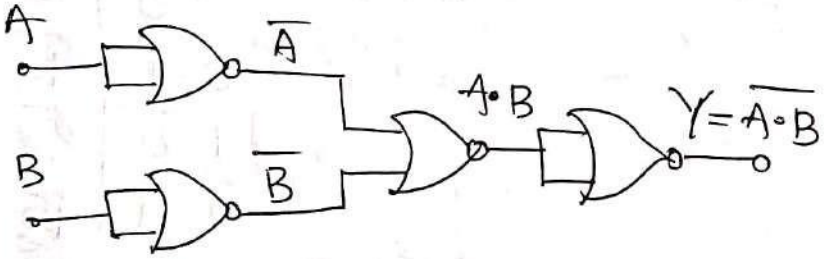
③ AND gate



Truth Table

A	B	Y = A·B
0	0	0
0	1	0
1	0	0
1	1	1

④ NAND gate



Truth Table

A	B	Y = A·B
0	0	1
0	1	1
1	0	1
1	1	0

Minterms and maxterms

→ A binary variable may appear in its normal form (x) or in its complement form (x').

→ When two binary variables x & y combined with an AND operation, there are four possible combinations $x \cdot x'y'$, $x'y$, xy', xy. Each of four AND term is called minterm (or) standard product.

→ Similarly for n variable, there are 2^n minterms.

→ If the bit is '0', then term complemented
bit is 1, then term uncomplemented.

ex:

x	y	Minterm Term	Minterm Designation
0	0	$x'y'$ (or) $\bar{x}\bar{y}$	m_0
0	1	$x'y$ (or) $\bar{x}y$	m_1
1	0	xy' (or) $x\bar{y}$	m_2
1	1	xy (or) xy	m_3

→ Similarly, n variable OR operation gives 2^n possible combination which are called maxterm (or) standard sum.

→ In this case if bit 0 → term - uncomp
bit 1 → Term complement

Maxterm

ex.

x	y	Term	Designation
0	0	$x+y$	M_0
0	1	$x+\bar{y}$	M_1
1	0	$\bar{x}+y$	M_2
1	1	$\bar{x}+\bar{y}$	M_3

Minterms and Maxterms for three variable

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

Canonical form

→ It is also known standard form.
→ It is used to get minterms / maxterms for the given reduced expression.

→ In canonical form, each term must contain all variables.

Question

Convert $F = A + \bar{B}C$ into canonical form.

Soln

$$F = A + \bar{B}C$$
$$= A \cdot 1 \cdot 1 + 1 \cdot \bar{B} \cdot C$$
$$= A(B+\bar{B})(C+\bar{C}) + (A+\bar{A})\bar{B}C$$
$$= (AB + A\bar{B})(C+\bar{C}) + A\bar{B}C + \bar{A}\bar{B}C$$
$$= ABC + AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C + \bar{A}\bar{B}C$$
$$= ABC + AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C} + \bar{A}\bar{B}C$$

$$F(A,B,C) = \overset{1\,1\,1}{ABC} + \overset{1\,1\,0}{AB\bar{C}} + \overset{1\,0\,1}{A\bar{B}C} + \overset{1\,0\,0}{A\bar{B}\bar{C}} + \overset{0\,0\,1}{\bar{A}\bar{B}C}$$
$$= m_7 + m_6 + m_5 + m_4 + m_1$$

Question

Convert $F = (\bar{x}+y)(x+z)(y+z)$ into canonical form.

Soln:

$$F = (\bar{x}+y)(x+z)(y+z)$$
$$= (\bar{x}+y+0)(x+0+z)(0+y+z)$$
$$= (\bar{x}+y+z\bar{z})(x+y\bar{y}+z)(x\bar{x}+y+z)$$
$$= (\bar{x}+y+z)(\bar{x}+y+\bar{z})(x+y+z)(x+\bar{y}+z)$$
$$(x+y+z)(\bar{x}+y+z)$$
$$= \underset{1\,0\,0}{(\bar{x}+y+z)}\,\underset{1\,0\,1}{(\bar{x}+y+\bar{z})}\,\underset{0\,0\,0}{(x+y+z)}\,\underset{0\,1\,0}{(x+\bar{y}+z)}$$

$$F(x,y,z) = M_4 \cdot M_5 \cdot M_0 \cdot M_2$$

Sum of Product (SOP)

→ For n binary variable, there are 2^n distinct minterms or product re. any boolean function can be expressed as a sum of minterms / product (SOP).

→ While using SOP, only '1' output in the truth table are considered as minterm.

→ It is expressed as Σ_m (terms).

Product of Sum (POS)

→ For n binary variable, there are 2^n distinct maxterm (or) sum re any boolean function can be expressed as a product of maxterm / sum (POS).

→ While using POS, only '0' output in the truth table are considered as maxterm.

→ It is expressed as Π_M (term).

example

$\Sigma_m (1, 2, 5, 6)$ → SOP → minterm

$\Pi_M (0, 3, 4, 5)$ → POS → maxterm.

Question

Convert the following boolean expression into SOP fom.

1) $f(x, y, z) = (xy + z)(\bar{y} + xz)$

2) $f(x, y, z) = xy + \bar{x}\bar{y} + \bar{y}z$

(3) $f(A, B, C) = AB + \bar{A}C + BC$

$m_7 \quad m_6 \quad m_3 \quad m_5$

$= xyz + xy\bar{z} + \bar{x}yz + x\bar{y}z$

$1\,1\,1 \quad 1\,1\,0 \quad 0\,1\,1 \quad 1\,0\,1$

$f(x,y,z) = \sum_m (3,5,6,7)$

② $f(x,y,z) = xy + \bar{x}\bar{y} + \bar{y}z$

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

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

$= xyz + xy\bar{z} + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z} + x\bar{y}z + \bar{x}\bar{y}z$

$= xyz + xy\bar{z} + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z} + x\bar{y}z$

$1\,1\,1 \quad 1\,1\,0 \quad 0\,0\,1 \quad 0\,0\,0 \quad 1\,0\,1$

$m_7 \quad m_6 \quad m_1 \quad m_0 \quad m_5$

$f(x,y,z) = \sum_m (0,1,5,6,7)$

③ $f(A,B,C) = AB + \bar{A}C + BC$

$= AB(C+\bar{C}) + \bar{A}(B+\bar{B})C + (A+\bar{A})BC$

$= ABC + AB\bar{C} + \bar{A}BC + \bar{A}\bar{B}C + ABC + \bar{A}BC$

$= ABC + AB\bar{C} + \bar{A}BC + \bar{A}\bar{B}C$

$1\,1\,1 \quad 1\,1\,0 \quad 0\,1\,1 \quad 0\,0\,1$

$m_7 \quad m_6 \quad m_3 \quad m_1$

$f(A,B,C) = \sum_m (1,3,6,7)$

Question

Convert the following boolean expression into POS form.

① $f(x,y,z) = (xy+z)(y+xz)$

② $f(x,y,z) = xy + \bar{x}\bar{y} + \bar{y}z$

③ $f(A,B,C) = AB + \bar{A}C + BC$

Sol^n

① $f(x,y,z) = (xy+z)(y+xz)$

$= (x+z)(y+z)$

$= (x+z)(y+z)(y+x)(y+z)$

$= (x+z)(y+z)(x+y)$

$= (x+y)(x+z)(y+z)$

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

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

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

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

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

M_0 M_1 M_2 M_4

$f(x,y,z) = \Pi_M(0,1,2,4)$

② $f(x,y,z) = xy + \bar{x}\bar{y} + \bar{y}z$

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

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

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

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

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

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

$= (\bar{x}+1)(\bar{x}+y+z)(x+\bar{y})(x+\bar{y}+z)$

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

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

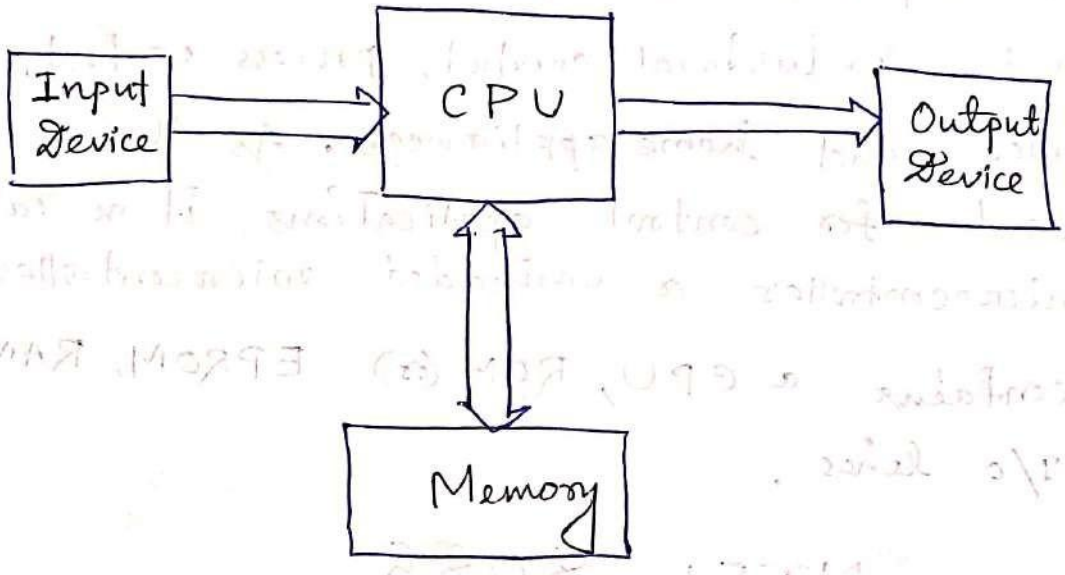
$= (\bar{x}+y+z)(x+\bar{y}+z)(x+\bar{y}+\bar{z})$

1 0 0 0 1 0 0 1 1

M_4 M_2 M_3

$f(x,y,z) = \Pi_M(2,3,4)$

Microprocessor



(Schematic diagram of a digital computer)

→ A digital computer is a programmable machine.

→ Its main components are CPU (Central Processing Unit, memory, input device & output device) as shown in above figure.

→ The central processing unit built on a single chip (IC) is called microprocessor.

→ A digital computer in which one micro processor is used as CPU is called microcomputer.

→ A microprocessor is a digital electronic device capable of performing arithmatic, logical and controlling operation for digital computer fabricated on a single Integrated Circuit (IC)

Single chip microcomputer (or) microcontroller

A digital computer built on a single IC is called single-chip microcomputer.

Such computers are used in instrumentation, automatic industrial control, process control, consumer and home appliances. As these are used for control applications, it is called as microcontroller or embedded microcontroller.

→ It contains a CPU, ROM (or) EPROM, RAM and I/O lines.

INTEL 8085

Intel 8085 is an 8-bit, NMOS microprocessor. It is a 40 pin IC package fabricated on a single LSI chip.

It uses a single +5 V dc supply.

Its clock frequency is about 3 MHz.

The clock cycle is 320 ns.

It has 80 basic instructions and 246 opcodes.

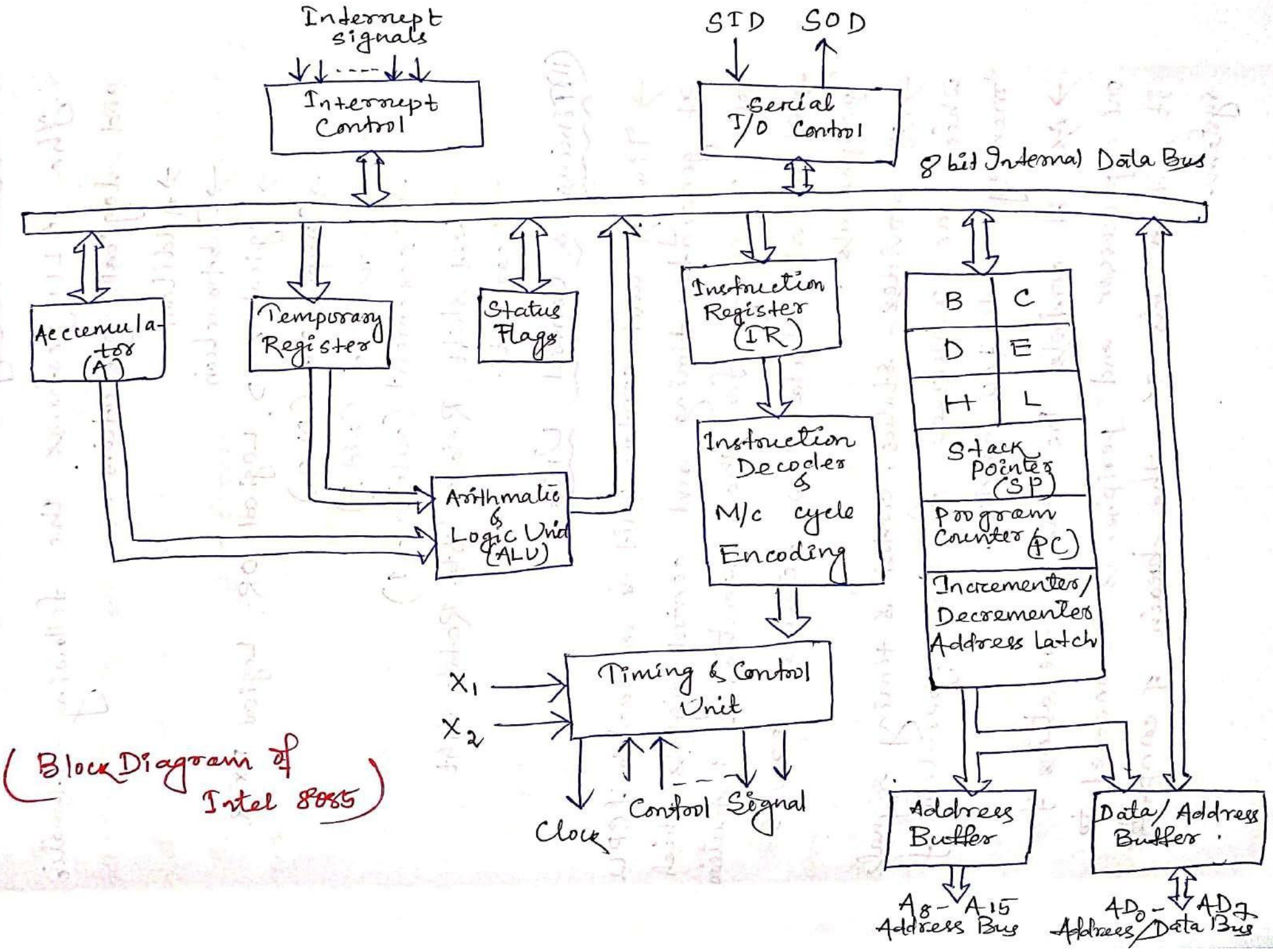
Block diagram of Intel 8085 µp.

→ It consists of three main sections,

(I) Arithmatic & logic Unit (ALU)

(II) Timing and control Unit

(III) Set of registers.



(Block Diagram of Intel 8085)

(I) Arithmatic & Logic Unit (ALU)

The ALU performs the following arithmatic and logical operations.

- → Addition
- → Subtraction
- → Logical AND, Logical OR, Logical EX OR
- → Complement (NOT)
- → Increment (add 1)
- → Decrement (subtract 1)
- → Left shift, Rotate left, Rotate right
- → Clear etc.

(II) Timing & Control Unit

→ Timing and control unit is a section of CPU. It generates timing and control signals which are necessary for the execution of instruction.

→ It controls data flow between CPU & peripherals.

→ It provides status, control & timing signals which are required for the operation of memory and I/O devices.

→ As it controls the entire operations of microprocessor and peripherals connected to it, it acts as the brain of computer system.

(iii) Registers

Registers are used by the microprocessor for temporary storage and manipulation of data and instructions.

Intel 8085 microprocessor has the following registers.

(i) One 8-bit accumulator (Acc) or register A

(ii) Six 8 bit general purpose register (B, C, D, E, H & L).

(iii) One 16-bit stack pointer, SP

(iv) One 16-bit program counter, PC

(v) Instruction Register.

(vi) Temporary Register.

Accumulator :—

The accumulator is an 8-bit register associated with ALU.

→ The register 'A' is the accumulator in 8085.

→ It is used to store one of the operand of arithmatic (or) Logical operation.

→ Other operand is stored in memory (or) in one of general purpose registers (B, C, D, E, H, L).

→ The final result of arithmatic & Logical operation is stored in accumulator.

General Purpose Registers

→ The 8085 microprocessor contains six 8-bit general purpose registers. ie B, C, D, E, H & L.

→ Two 8-bit registers are combined to hold 16 bit data known as register pair.

eg- B-C, D-E, & H-L pair.

→ The H-L pair is used to act as memory pointer & it holds the 16 bit address of a memory location.

→ The general purpose registers & accumulator are accessible to programmer.

Program Counter (PC)

→ It is a 16-bit special-purpose register.

→ It is used to hold the address of the next instruction to be executed.

→ The microprocessor increments the content of the program counter during the execution of an instruction, so it points the address of the next instruction in the program at the end of execution of an instruction.

Stack Pointer (SP)

→ It is a 16-bit special-purpose register.

→ Stack is a sequence of memory location set by a programmer to store/retrive

the content of accumulator, flag, program counter and general purpose registers during execution of a program.

→ Stack works on LIFO (Last - In - First - Out) so its operation is faster as compared to normal store/retrieve of memory location.

→ The stack pointer (SP) controls addressing of the stack.

→ The stack pointer holds the address of the top element of data stored in the stack.

→ PUSH & POP instructions are used in stack operation.

Instruction Register (IR)

The instruction register holds the opcode (operation code or instruction code) of the instruction which is being decoded and executed.

Temporary Register

→ It is an 8-bit register associated with ALU.

→ It holds data during arithmatic & logical operation & used by microprocessor.

It is not accessible to programmer.

Flags

The Intel 8085 microprocessor contains five flipflops to serve as status flags.

→ The flip flops are set or reset according to the conditions which arise during an arithmatic or logical operation.

→ The five status flags of Intel 8085 are

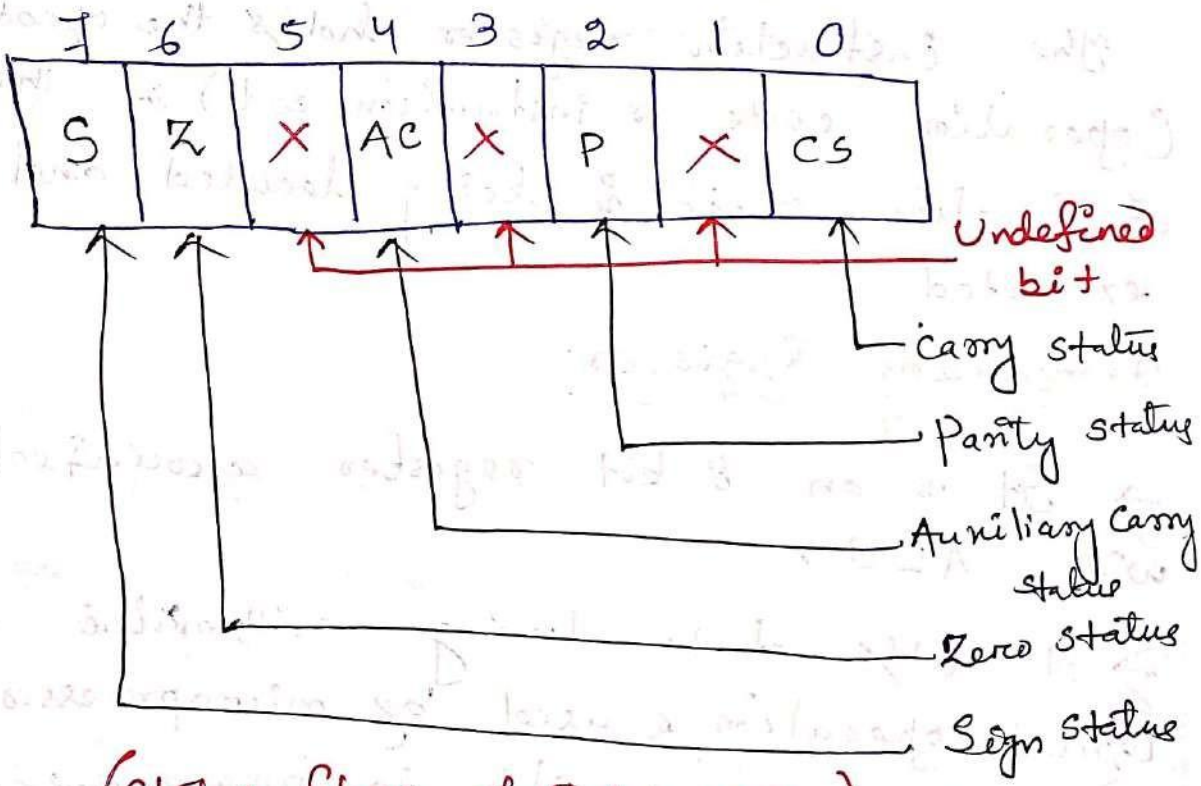
a) carry Flag (CS)

b) Parity Flag (P)

c) Auxiliary carry Flag (AC)

d) Zero Flag (Z)

e) Sign Flag (S)



(Status flags of Intel 8085)

→ If a F/F for a particular flag is set, it indicates 1, when it is reset, it indicates '0'

Carry flag (CS)

→ After the execution of an arithmatic instruction, if a carry is produced, the carry flag CS is set to 1, otherwise it is 0.

→ The carry flag holds carry out of the Most Significant Bit (MSB) resulting from execution of an arithmatic operation.

Parity Flag (P)

→ If the result of an arithmatic (or) logical operation contains even nos of '1', parity status P is set to '1'.

→ If the result contains odd no of 1, P is reset is P is '0'.

Auxiliary Carry (AC) Flag

→ The auxiliary carry flag AC holds carry out from bit no 3 to the bit no 4 resulting from execution of arithmatic operations.

→ If carry generated AC = 1, if no carry AC = 0.

Zero Flag (Z)

If the result of an arithmatic (or) logical

operation is zero, then zero status flag Z is set to '0'. If the result is nonzero, then Z = 0.

Sign flag (S)

→ If the result of arithmatic (or) logical operation is positive, the sign flag is '0'.

→ If the result is negative, sign flag is '1'.

opcode and operands

Each instruction contains two parts

(i) operation code (opcode)

(ii) Operand.

opcode –

The opcode specifies the task to be performed by the computer.

Operand –

The operand specifies the data to be operated on.

The operand given in the instruction may be of 8 bit (or) 16 bit data, 8-bit (or) 16 bit address.

Instruction word size.

Intel 8085 instructions are classified into three types

1) 1-byte instruction.
2) 2-byte instruction.
3) 3-byte instruction.

One-byte instruction

All one-byte instructions contain information regarding operands in the opcode itself.

Ex- MOV A, B ← move content of register B to register A

ADD B ← Add the content of register B to register A.

RAL ← Rotate the content of A left by one bit.

Two byte Instruction :-

In a two-byte instruction, 1st byte of the instruction is opcode and 2nd byte is either data (or) address.

Ex- MVI B, 05; Move 05 to register B.

IN 01; Read data at port B.

DB, 01, IN 01 in the code form.

Three-byte instruction

In a three byte instruction, the 1st byte of the instruction is its opcode, and 2nd and 3rd bytes are either 16 bit data or 16-bit address.

Ex-① LXI H, 2400H, Load H-L pair with 2400H.

21, 00, 24 ; LXI H, 2400H in the code form.

② LDA 2500H, get the content of the memory location 2500H into 'A'.

3A, 00, 25 ; LDA 2500H in the code form.

Instruction Cycle

An instruction is a command given to the computer to perform a specified operation.

→ A sequence of instructions which performs a specific task is called program.

→ The CPU fetches one instruction from the memory at a time and execute it.

→ The steps that CPU carries out to fetch an instruction and necessary data from the memory and execution is called instruction cycle (IC).

→ An instruction cycle consists of a fetch cycle (FC) and execute cycle (EC).

$$IC = FC + EC$$

→ Fetch cycle

The process of fetching an opcode from the memory is called fetch cycle.

→ Execute Cycle.

The process of getting data from memory and performing specific operation is called execute cycle.

→ The time required to fetch an opcode (FC) is fixed and the time required to execute an instruction (EC) is variable which depends on the type of instruction.

Addressing Modes

The techniques which specify data for instruction on which it has to perform operation is called addressing modes.

Intel 8085 has following addressing modes

1. Direct addressing
2. Register addressing
3. Register indirect addressing
4. Immediate addressing
5. Implicit addressing.

Direct Addressing

In this type of addressing the address of operand (data) is given in the instruction itself.

Ex- a) STA 2400H → Store the content of accumulator in the memory location 2400H.

b) IN 02 → Read data from the port C.
(02 is the address of port C).

Register addressing

In register addressing mode, the operand is stored in one of the general purpose register (A, B, C, D, E, H, L).

the opcode specifies the address of register(s) in addition to the operation to be performed.

Ex- a) MOV A, B → move the content of register B to register A.

b) ADD B → Add the content of register B to the content of register A.

Register Indirect Addressing

In this mode of addressing, the address of the operand is specifies by a register pair.

Ex a) LXI H, 2500 H → Load H-L pair with 2500 H.
MOV A, M → Move the content of memory location whose address is H-L pair (2500 H) to 'A'

b) LXI H, 2500 H → Load H-L pair with 2500 H.
ADD M → Add the content of memory location (2500 H) to 'A'.

Immediate Addressing

In immediate addressing mode, the operand is specified within the instruction itself.

Ex a) MVI A, 05 → move 05 to register A

b) ADI 06 → Add 06 to content of A

Implicit Addressing

In this addressing mode, the instructions operate on the content of accumulator. So such instruction doesn't require the address of the operand.

Ex- CMA, RAL, RAR.