Lecture Note

by

Sasmita Das

Digital Electronics

And Microprocessor

1.1 Number Systems and Arithmetic

Signal :-

Signal is a physical quantity that depends on independent variable such as time, space, frequency etc.

→ The signal are of two types these are
(I) Analog signal
(II) Digital signal

(I) Analog Signal :-
The signal which is continous in nature that means the amplitude is change in every time.

(II) Digital signal :-
The signal which is discrete in nature that means for a particulare time period the amplitude is constant.

→ The Digital signal are represented by the help of binary number system i.e '0' & '1'

→ The number system are of four types. These are
(I) Binary number syetem
(II) Decimal number system
(III) octal number system
(IV) Hexadecimal number system.

(I) Binary number system
It is a type of number system in which the digital signal are represented by either "0" or "1".

(II) Decimal number syetem
It is a type of number system in which the digital signal are represented by 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

(III) Octal number system
It is a type of number system in which the digital signal are represented by 0, 1, 2, 3, 4, 5, 6, 7.

Hexadecimal number system :-

It is a type of number system in which the Digital signal are represented by 0, 1, 2, 3, 4 ..... 15.

→ In Hexadecimal number system 0 to 9 Digits are used and 10 is represented by A, 11 is represented by B, 12 is represented by C, 13 is represented by D, 14 is represented by E, and 15 is represented by F.

* The binary no. is represented by $(\ )_2$
* The Decimal no. is represented by $(\ )_{10}$
* The octal no. is represented by $(\ )_8$
* The Hexadecimal no. is represented by $(\ )_{16}$.

The steps for the conversion of Decimal no. into any base :-

Step-1

At the first step the given Decimal no. is converted into the base using successive division by the base or radix by living its remainder.

Step-2

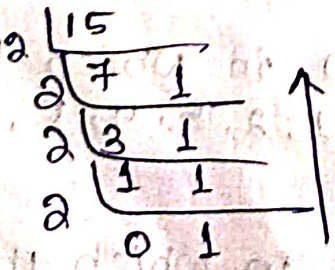
The sucessive division of quotient can be repeated until the quotient is less than the base.

Step-3

After completion of division collect the remainder from bottom to top.

Convert $(15)_{10}$ into the binary number ?

Ans :-



$(15)_{10} = (1111)_2$

* Convert $(225)_{10}$ Decimal no. into the binary no.?

2	225	
2	112	1
2	56	0
2	28	0
2	14	0
2	7	0
2	3	1
2	1	1
	0	1

$(225)_{10} = (11100001)_2$

* Convert $(0.8125)_{10}$ into the binary no.

$0.8125 \times 2 = 1.6250 \rightarrow 1$

$0.6250 \times 2 = 1.2500 \rightarrow 1$

$0.2500 \times 2 = 0.5000 \rightarrow 0$

$0.5000 \times 2 = 1.0000 \rightarrow 1$

$(0.8125)_{10} = (0.1101)_2$

* Convert $(111.001)_2$ into Decimal no.

$(111.001)_2$

$= 1\times2^2 + 1\times2^1 + 1\times2^0 + 0\times2^{-1} + 0\times2^{-2} + 0\times2^{-3}$

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

$= 7.125$

$(111.001)_2 = (7.125)_{10}$

* Convert $(1101)_2$ into Decimal no.

Ans:- $(1101)_2$

$= 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$

$= 8 + 4 + 0 + 1$

$= 13$

* Convert the binary no. $(1110.110)_2$ into the Decimal no.

Ans:- $(1110.110)_2$

$= 1\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0 + 1\times2^{-1} + 1\times2^{-2} + 0\times2^{-3}$

$= 8 + 4 + 2 + 0 + 0.5 + 0.25 + 0$

$= (14.75)_{10}$

$(1110.110)_2 = (14.75)_{10}$

* Convert the Decimal no. into octal no.

Ans:- $(225)_{10} \rightarrow (\ )_8$

8	225	
8	28	1
8	3	4
	0	3

$(225)_{10} = (341)_8$

* Convert $(1225)_{10}$ into octal no.

Ans:-

8	1225	
8	153	1
8	19	1
8	2	3
	0	2

$(1225)_{10} = (2311)_8$

* Convert $(731.151)_8$ into the Decimal no.

Ans:- $(731.151)_8$

$= 7\times8^2 + 3\times8^1 + 1\times8^0 + 1\times8^{-1} + 5\times8^{-2} + 1\times8^{-3}$

$= 448 + 24 + 1 + 0.125 + 0.078 + 0.001$

$= 473.204$

$(731.151)_8 = (473.204)_{10}$

* convert $(731)_8$ into the Decimal no.

Ans:- $(731)_8$

$= 7 \times 8^2 + 3 \times 8^1 + 1 \times 8^0$

$= 448 + 24 + 1$

$= 473$

$(731)_8 = (473)_{10}$

* convert the Decimal no. into the Hexadecimal no.

Ans:- $(1225)_{10} \rightarrow (\ )_{16}$

16	1225	
16	76	9
16	4	C
	0	4

$(1225)_{10} = (4C9)_{16}$

* convert $(1225.125)_{10}$ into Hexadecimal no.

Ans:-

16	1225	
16	76	9
16	4	C
	0	4

0.125
$\times \quad 16$
$2.000 \rightarrow 2$

$(1225.125)_{10} = (4C9.2)_{16}$

* convert $(1225.125)_{10}$ into octal

Ans:-

8	1225	
8	153	1
8	19	1
8	2	3
	0	2

0.125
$\times \quad 8$
$1.000 \rightarrow 1$

$(1225.125)_{10} = (2311.1)_8$

* convert $(11C.1)_{16}$ into Decimal no.

Ans:- $(11C.1)_{16}$

$= 1 \times 16^2 + 1 \times 16^1 + C \times 16^0 + 1 \times 16^{-1}$

$= 256 + 16 + 12 + 0.0625$

$= 284.0625$

$(11C.1)_{16} = (284.0625)_{10}$

Convert the binary no. into the octal no :-

step-1

first write the binary no.

step-2

Make groups of three bits starting from the binary points.

step-3

If the last group contain less than three bits then we assume the remaining bits to be zero.

step-4

for each three bits group find the octal digit.

step-5

To get the result place the octal digit into the same order.

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

* convert $(111001110)_2$ into octal no.

Ans :-

$$\underset{7}{111}\ \underset{1}{001}\ \underset{6}{110}$$

$$(111001110)_2 = (716)_8$$

* convert $(101111101.11101)_2$ into octal no.

Ans :-

$$\underset{2}{010}\ \underset{7}{111}\ \underset{5}{101}.\ \underset{7}{111}\ \underset{2}{010}$$

$$(101111101.11101)_2 = (275.72)_8$$

Conversion of binary no. into the Heradecimal no.

Step-1

first write the binary no.

step-2

Make groups of four bits starting from the binary points.

step-3

If the last group contains less than four bits then we assume the remaining bits to be zero.

step-4

for each four bits groups find the binary digit.

step-5

To get the result place the binary digit into the order.

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

* Convert $(1100101.110)_2$ into Hexadecimal no.

Ans:- $(1100101.110)_2$

$= \underset{6}{0110}\ \underset{5}{0101}.\ \underset{C}{1100}$

$(1100101.110)_2 = (65.C)_{16}$

* conversion of Hexadecimal no. into the Binary

Ans:- $(FC.B)_{16} \rightarrow (11111100.1011)_2$

$(BF.F)_{16} \rightarrow (10111111.1111)_2$

1.a Binary Addition

Rules :- $0+0=0$

$0+1=1$

$1+0=1$

$1+1=0$ with carry 1

Eg :- $(1101.101)_2 + (111.011)_2$

$$\begin{array}{r} 1101.101 \\ +\ 111.011 \\ \hline 10101.000 \end{array}$$

Binary Subtraction

Rules :- $0-0=0$

$1-1=0$

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

$1-0=1$

$$\begin{array}{r} 1010.010 \\ 0111.111 \\ \hline 0010.011 \end{array}$$

Binary Multiplication

Rules :-
$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 0 = 0$
$1 \times 1 = 1$

Eg:- $(101)_2 \times (101)_2$

```
      101
    x 101
    -----
      101
     000x
    101xx
    -----
    11001
```

Binary Division

Eg:- $(101101)_2 \div (110)_2 = ?$

```
110 | 101101 | 111.1
      110
      ----
      1010
       110
      ----
      1001
       110
      ----
       0110
        110
       ----
       0000
```

1.3 Complements

Complements is a method which is used for subtraction purpose.

→ usually complements are of two types

(I) $(r-1)$'s complement.

(II) r's complement.

Here $r \rightarrow$ radix

(I) **$(r-1)$'s complement :-**

In $(r-1)$'s complement all the digits are substracted from $(r-1)$

(II) **r's complement :-**

In r's complement first we have find out $(r-1)$'s complements.

→ Then we are adding '1' to get the r's complements

→ The r's complement can also find out by using the formula $\boxed{r^n - N}$.

Here r = The base

n = The numbere of digit

N = The given no.

Eg:-

* find out the 9's complement & 10's complement of $(79)_{10}$

Ans:- Here $r = 10$

$(r-1)$'s $= 9$

$$\begin{array}{r} 99 \\ -79 \\ \hline (20) \end{array}$$ 9's complement

$$\begin{array}{r} +1 \\ \hline 21 \end{array}$$ 10's complement

Or 10's complement $= r^n - N$

$= 10^2 - 79$

$= 100 - 79$

$= 21$

* find out the 1's complement and 2's complement of $(1101)_2$

Ans:- Here $r = 2$

$r - 1 = 1$

1's complement = $\begin{array}{r} 1111 \\ 1101 \\ \hline 0010 \end{array}$

2's complement = $(0010 + 1) = (0011)_2$

Or

2's complement = $r^n - N$

$= 2^4 - 1101$

$= 10000 - 1101$

$= (0011)_2$

* find out the 1's and 2's complement of $(0110)_2$

Ans:- Here $r = 2$

$r - 1 = 1$

1's complement = $\begin{array}{r} 1111 \\ 0110 \\ \hline 1001 \end{array}$

2's complement = $(1001 + 1)_2 = (1010)_2$

Or

2's complement = $r^n - N$

$= 24 - 1001 = 10000 - 1001$

$= (1010)_2$

1.4 Subtraction of binary number in 2's complement

Method

step-1

First find out the 2's complement of subtrahend

step-2

Then add the minuend with the 2's complement of subtrahend

step-3

If the end carry is 1 then discard the carry and take the result as positive.

Step-4

If their is no carry then the result of the 2's complement of that no. is taken as negative.

* Eg. Subtract $(1011)_2$ from $(1111)_2$ using 2's complement.

Ans:- 1111 → Minuend

1011 → Subtrahend

```
 1111
-1011
------
 0100  ----> 1's
+    1
------
 0101  ----> 2's
```

Add minuend + 2's complement

```
 1111
+0101
------
10100
```

$(1111)_2 - (1011)_2 = (0100)_2$

* Subtract $(1111)_2$ from $(1010)_2$ using 2's complement

Ans:- 1010 → minuend

1111 → Subtrahend

2's complement

Subtraction of two decimal no. using 10's complements:-

Step-1

first find out the 10's complement of subtrahend

Step-2

Then add the minuend with the 10's complement of subtrahend

Step-3

If the end carrey is 1 then discard the carrey and take the result as positive.

Step-4

If theire is no carrey then the result of the 2's complement of that no. is taken as negative.

* Eg. subtract $(53)_{10}$ from $(77)_{10}$

Ans:- 77 → minuend

53 → subtrahend

10's complement

$$\begin{array}{r} 99 \\ 53 \\ \hline 46 \\ +\ 1 \\ \hline 47 \end{array}$$

Add minuend + 10's complement

$$\begin{array}{r} 77 \\ +47 \\ \hline 124 \end{array}$$

$(77)_{10} - (53)_{10} = (24)_{10}$

1.5. Use of weighted and un-weighted codes & Write Binary equivalent number for a number in 8421 Excess-3 and Gray code and vice-versa.

Digital codes

If the Magnitude of decimal number is very large Then it is difficult to find out the binary number using division.

→ To overcome this problem we can use code numbers for each decimal digits and the codes are called as digital codes.

→ The digital codes are of two types.

(i) weighted codes

(ii) unweighted codes

(i) **weighted codes**:-

The codes in which each digit position is assigned with a weight is called as weighted codes.

Eg:- BCD codes.

BCD codes

→ BCD stands for Binary coded Decimal code.

→ BCD codes are also called as 8421 code.

→ In this code each decimal digit is represented by a 4 bit binary number.

→ In BCD codes for each decimal no. (0 to 9) a BCD code is assigned.

Decimal	BCD code
0 →	0000
1 →	0001
2 →	0010
3 →	0011
4 →	0100
5 →	0101
6 →	0110
7 →	0111
8 →	1000

* convert $(53)_{10}$ into BCD no.

Ans:- $(53)_{10} = (01010011)_2$

* convert $(99)_{10}$ into BCD no.

Ans:- $(99)_{10} = (10011001)_2$

* convert $(75)_{10}$ into BCD no.

Ans:- $(75)_{10} = (01110101)_2$

* Convert the BCD $(100100110100)_2$ into Decimal.

Ans:- $(100100110100)_2 = (934)_{10}$

BCD Addition

Step-1

Add two BCD number using binary addition

Step-2

If the result is greater than 9, then add binary 6 with the four bit result to get a valid BCD no.

Step-3

If the result produces end carry then the binary 6 (0110) must be added with each four bit BCD number.

Step-4

If the result is equal to or less than 9 then it is a valid BCD no.

* Add $(7)_{10}$ with $(3)_{10}$ using BCD codes?

Ans:- $(7)_{10} = 0111$

$(3)_{10} = 0011$

$$\begin{array}{r} 0111 \\ +\ 0011 \\ \hline 1010 \\ +\ 0110 \\ \hline 00010000 \\ \hline 1\quad 0 \end{array}$$

$(7)_{10} + (3)_{10} = (00010000)_2$

* Add $(88)_{10}$ with $(88)_{10}$ using BCD codes?

Ans:- $(88)_{10} = 10001000$

$(88)_{10} = 10001000$

```
  10001000
+ 10001000
-----------
 100010000
+ 01100110
-----------
 000101110110
```

0001 → 1
0111 → 7
0110 → 6

$(88)_{10} + (88)_{10} = (176)_{10}$

(ii) Unweighted code :-

The codes in which the digital bits does not have weighted position value is called as unweighted codes.

Eg:- Excess-3 codes
Gray codes

a) Excess-3 Codes

The BCD numbers are converted into the excess-3 codes by adding "3" with each BCD codes.

Decimal	BCD	Excess-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Gray code conversion

binarey no. into the Gray code.

Step-1

The MSB of binary code is same as gray code.

Step-2

To get the next gray code bit add the MSB of Binary with next bit of Binary.

Step-3

To get further next bit of Gray code add the consqutive next bit & previous bit. Repeat it until get LSB of gray code.

* Convert $(1111)_2$ into gray code

Sol:- (1111)

MSB → LSB

1 1 1 1

↓ ↓ ↓ ↓

1 0 0 0

* find the gray code of $(101101)_2$

Ans:- 1 0 1 1 0 1

↓ ↓

MSB LSB

1 0 1 1 0 1

↓ ↓ ↓ ↓ ↓

1 1 1 0 1 1

→ Gray code is also known as cyclic code.

→ This code is used for error checking & correction in digital communication.

Conversion of gray code into the binary :-

step-1

The MSB in gray code same as the MSB of binary.

step-2

To get the next binary number add the binary MSB with the next significant bit.

step-3

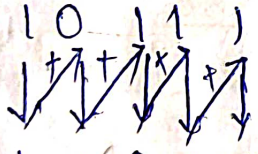
Record the result neglecting carry and continue the process until LSB is

step-4

The no. of binary bits same as the number of gray code bits.

* Find the binary code for gray code $(10111)_2$

Ans :-



$(10111)_2 = (11010)_2$

Ascii code :-

ASCII As American standard code for Information Interchange.

→ The ASCII code is seven bit code

1.6 Importance of parity bit

parity bit is an extra bit that to be added with data bits to detect the errors appears in the digital transmission.

→ A bit either 1 or 0 is added as a parity bit

→ parity bit is two types

(i) Even parity
(ii) odd parity

(i) Even parity

In Even parity the parity bit is selected as 0 if number of "1" present in the data is even, while the parity bit is selected "1" if number of "1" present in the data is odd.

(ii) odd parity

In odd parity the parity bit is selected as "0" if number of 1 is present in the data is odd, while the parity bit is selected "1" if number of "1" present in the data is even.

Data bit	Even parity	Odd parity
0000	0	1
0001	1	0
0010	1	0
0011	0	1
0100	1	0
0101	0	1
0110	0	1
0111	1	0
1000	1	0
1001	0	1
1010	0	1
1011	1	0
1100	0	1

	Even	odd
1101	1	0
1110	1	0
1111	0	1

1.7 Logic Gates :-

It is an electronics circuit having one or more than one input and only one output.

→ Logic gates are the basic building blocks of any digital system.

→ Usually 8 no. of Logic gates are used These are

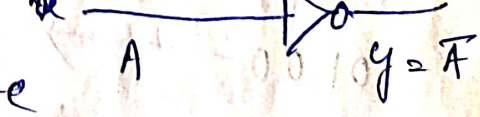
1. NOT
2. AND
3. OR
4. NAND
5. NOR
6. EX-OR
7. EX-NOR
8. Buffer

NOT Logic gates

→ In NOT gates if the input is high the output is low and if the input is low output is high

→ If the input to the NOT gate is A then the output to the NOT gate is $y = \overline{A}$

→ The symbol of NOT gate is



Truth Table for NOT gate

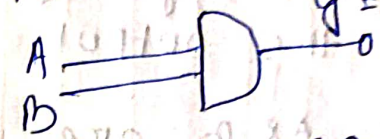
A	$y = \overline{A}$
0	1
1	0

AND gate :-

→ In AND gate two inputs are used to get the output.

→ If both the input are high then the output is high otherwise output is 0.

→ The symbol of AND gate is



→ Suppose the input to the AND gate is A & B then the output of AND gate is A · B.

Truth Table of AND gate

A	B	Y = A · B
0	0	0
0	1	0
1	0	0
1	1	1

OR gate

→ In OR gate two inputs are used. If any of the input is high then the output is high otherwise it is 0.

→ Suppose the input to the OR gate is A & B the the output of OR gate is A + B

→ The symbol of OR gate is

Truth Table of OR gate

A	B	y = A + B
0	0	0
0	1	1
1	0	1
1	1	1

NAND gate

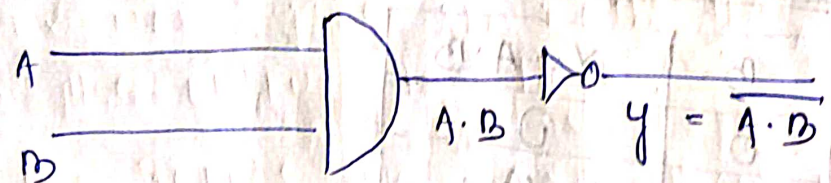
NAND gate = NOT gate + AND gate

→ In NAND gate we have two input & one output

→ NAND logic gates if any of the input is low then the output is high.

→ IF A & B are the inputs to the NAND gate then the output y is $(\overline{A \cdot B})$

→ the symbol of NAND gate is



→

Truth Table of NAND gate

A	B	A·B	Y = $\overline{A \cdot B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

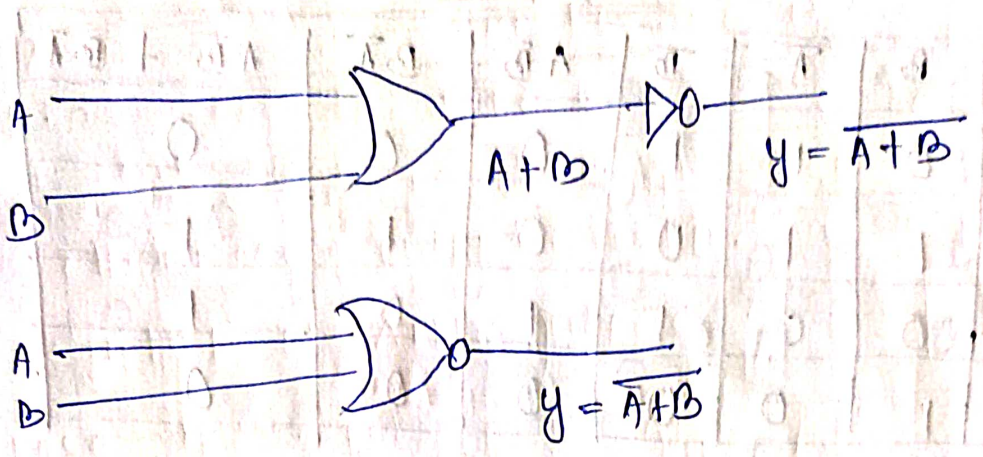
NOR gate:-

In NOR gate if both the inputs are low then the output is high.

→ In NOR gate we have two inputs and one outputs.

→ If the inputs are A & B then the output is $y = \overline{A + B}$

→ The symbol of NOR gate is



Truth Table fore NOR gate

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

EX-OR gate :-

In EX-OR gate if the eithere input is high then the output is high.

→ when the inputs are A&B then the output of the EX-OR gate is

$$y = A \oplus B$$

that means $y = A\bar{B} + B\bar{A}$

→ The symbol of EX-OR gate is

$y = A \oplus B$

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

Truth Table for EX-OR gate:-

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

EX-NOR gate:-

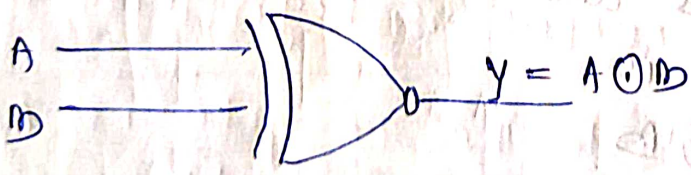
→ In EX-NOR gate, if both the inputs are equal then the output is high.

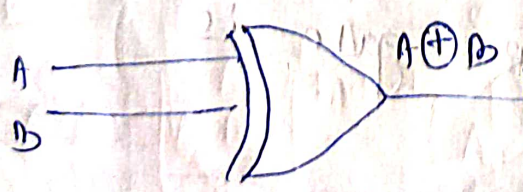
→ If the inputs are A & B then the output of the EX-NOR gate is

$$y = A \odot B$$

$$y = A\bar{B} + \bar{A}\bar{B}$$

→ The symbol of EX-NOR gate is





$$y = \overline{A \oplus B}$$

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

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

$$= \bar{A}\bar{B} + AB$$

Truth Table for EX-NOR gate :-

A	B	$\bar{A}$	$\bar{B}$	AB	$\bar{A}\bar{B}$	$AB + \bar{A}\bar{B}$
0	0	1	1	0	1	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	1	0	1

Buffer gate :-

→ In Buffer gate what ever inputs are given the same outputs are taken.

→ The symbol of Buffer gate is

A $\quad y = A$

If $\bar{A} \quad y = \bar{A}$

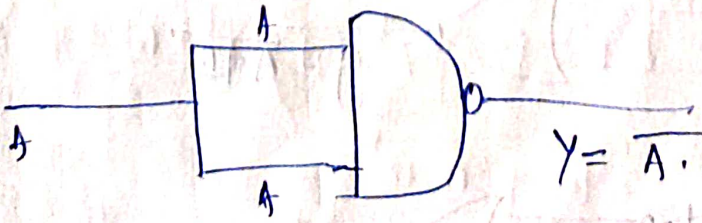
1.8 Realize AND, OR, NOT operations using NAND NOR gates.

Universal gate :-

→ It is a gate by which we can realised all the gates.

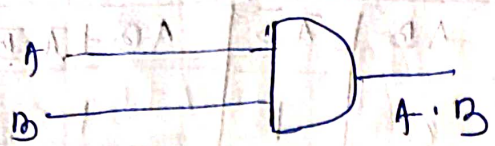
→ Usually the NAND gate and NOR gate are the universal gate.

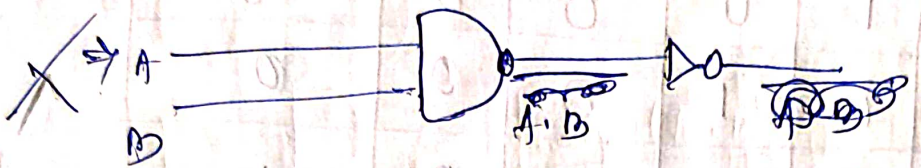
Realization of NOT gate using NAND gate :-

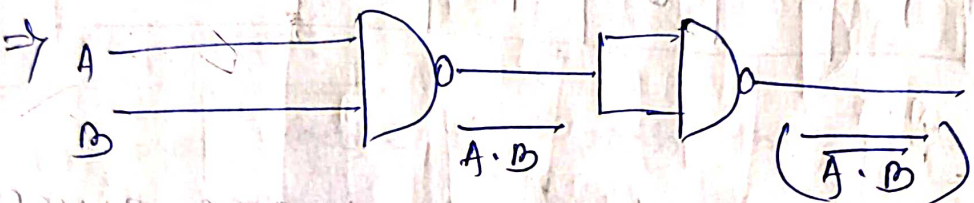


$Y = \overline{A \cdot A} = \bar{A}$

Realization of AND gate using NAND :-

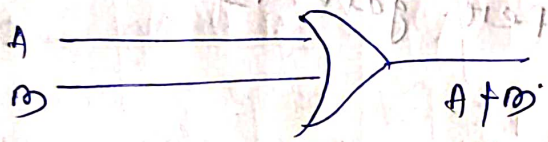


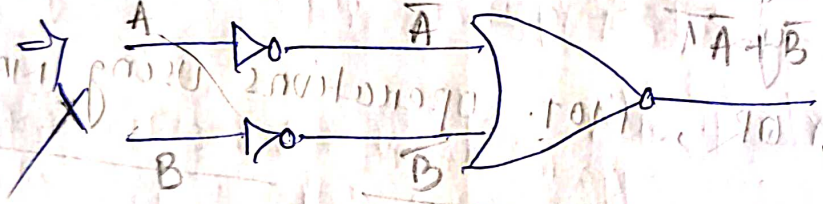




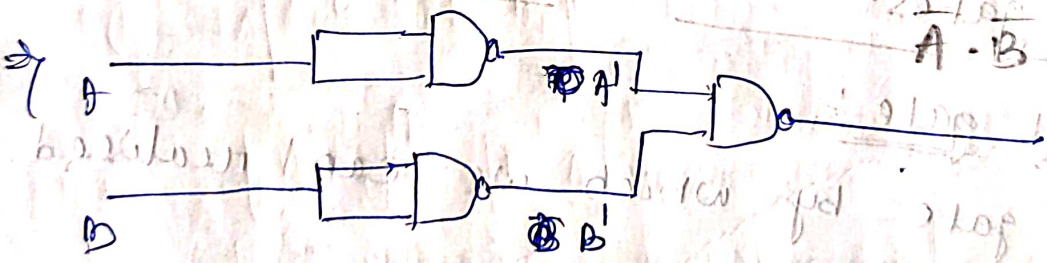
$= A \cdot B$

Realization of OR gate using NAND :-





$\overline{\bar{A} + \bar{B}} = \bar{\bar{A}}\,\bar{\bar{B}} = A \cdot B$

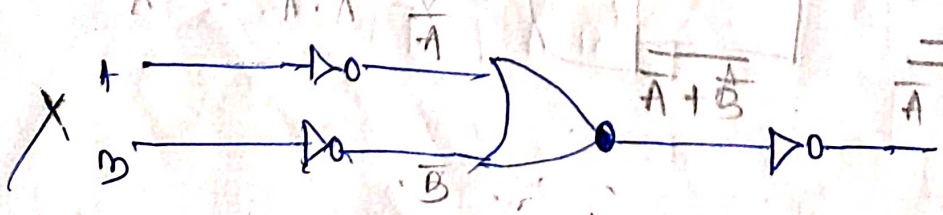


$\overline{A \cdot B} = \bar{A} + \bar{B}$

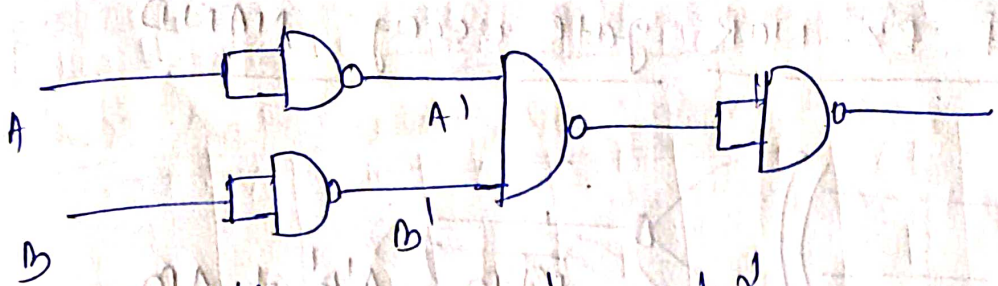
$A + B$

$[A'B']' = (A')' + (B')' = A + B$

Realization of NOR gate using NAND :-



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



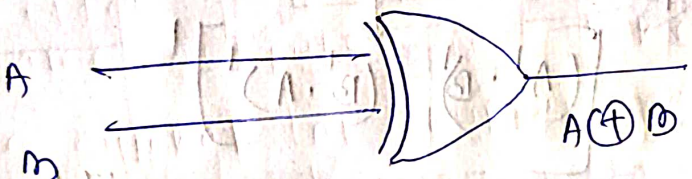
$$[A'B']' = (A')' + (B')'$$

$$= A + B$$

$$y = (A+B)'$$

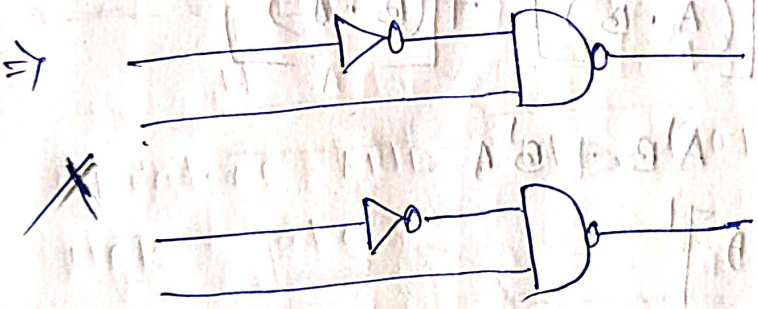
$$= \overline{A+B}$$

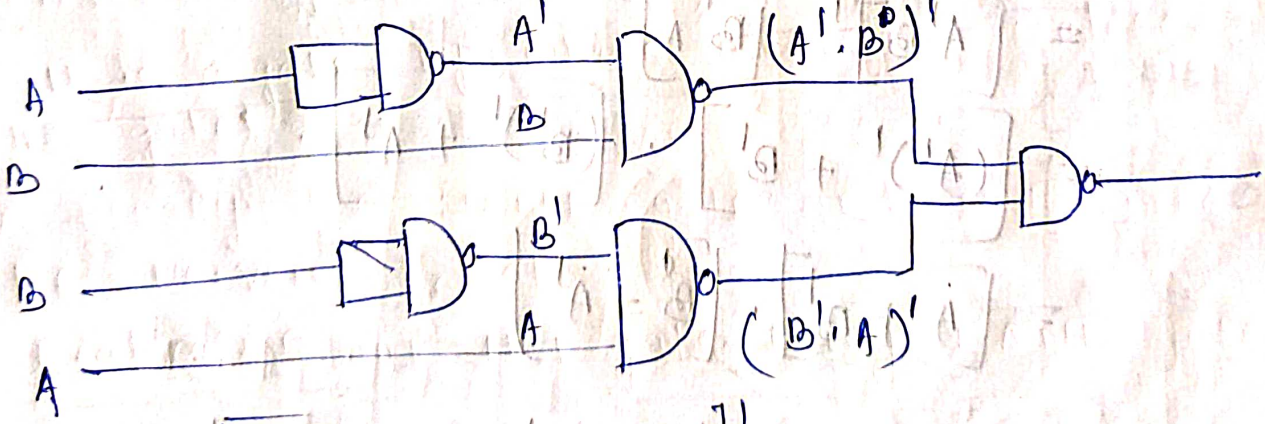
Realization of EX-OR gate using NAND



$$A \oplus B = A'B + AB'$$

$\Rightarrow$





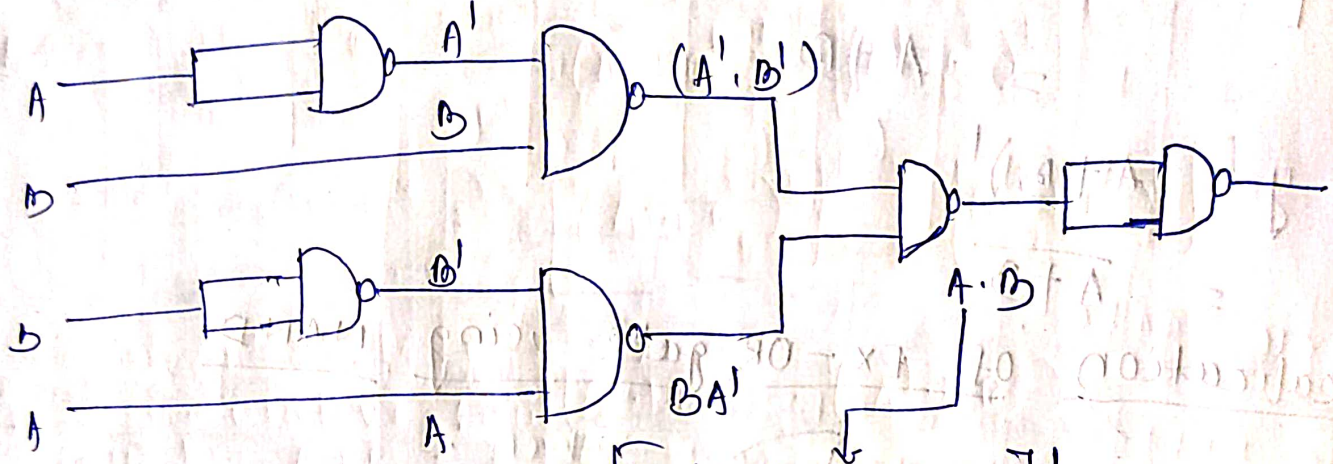
$$[(A'.B)'.(B'.A)']'$$

$$= [(A'.B)']' + [(B'.A)']'$$

$$= A'B + B'A$$

Realization of EX-NOR gate using NAND

$A \oplus B = A'B' + AB$



$$\left[(A'\cdot B')\,(B'\cdot A)'\right]'$$

$$= \left[(A'\cdot B)'\right]' + \left[(B'\cdot A)'\right]'$$

$$= A'B + B'A$$

$$y = \left[A'B + B'A\right]'$$

$$= \left[A'B\right]' \cdot \left[B'A\right]'$$

$$= \left[(A')' + B'\right] \cdot \left[(B')' + A'\right]$$

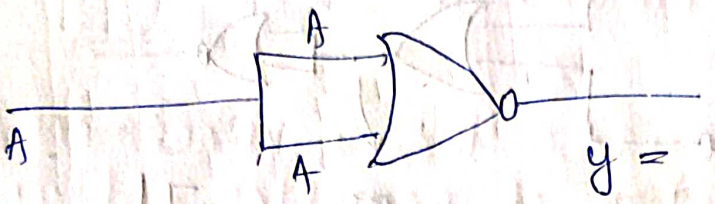
$$= \left[A + B'\right] \cdot \left[B + A'\right]$$

$$= A\cdot B + A\cdot A' + B'\cdot B + A'\cdot B'$$

$$= AB + A'B'$$

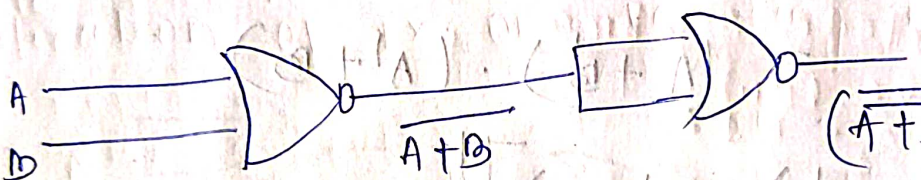
Realization of basic gate using NOR gate :-

NOT gate using NOR gate :-



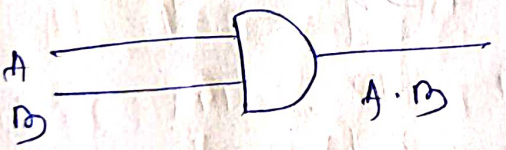
$$y = \overline{A + A} = \overline{A}$$

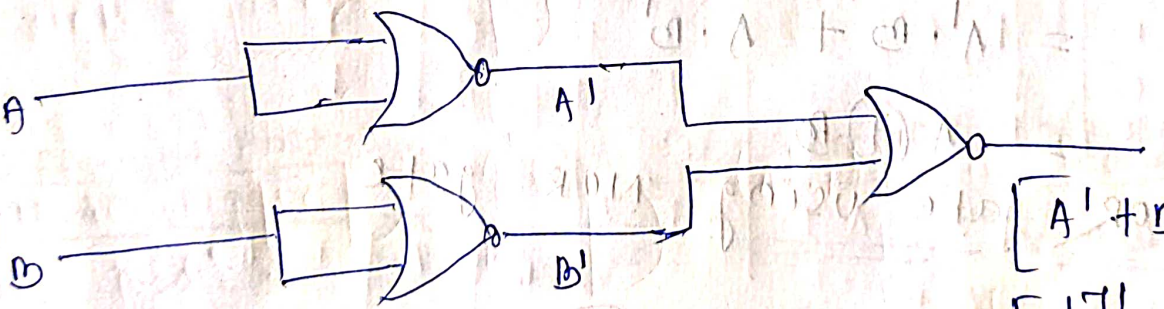
OR gate using NOR gate :-



$$\overline{(A+B)} = A+B$$

AND gate using NOR gate :-



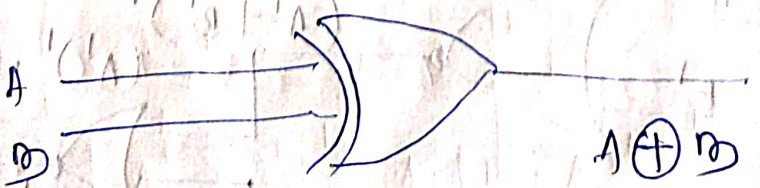


$$[A' + B']'$$

$$= [A']' \cdot [B']'$$

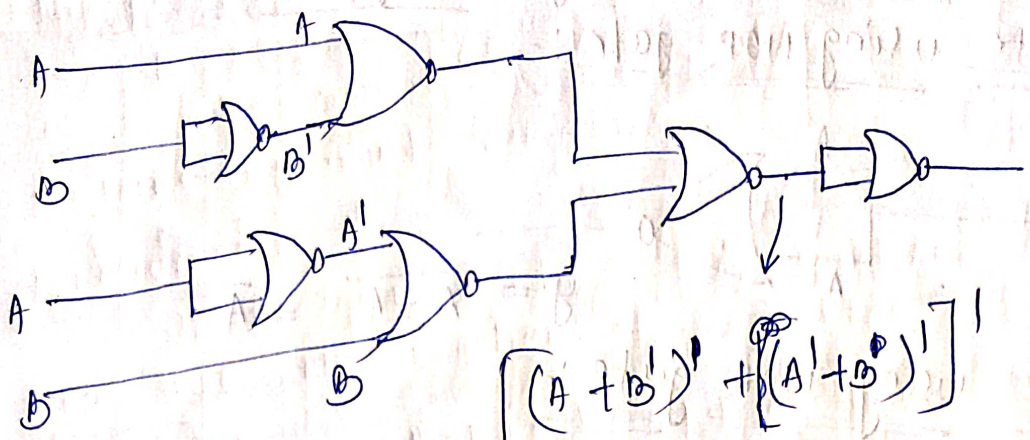
$$= A \cdot B$$

EX-OR gate using NOR gate :-



$$A \oplus B$$

$$= A'B + AB'$$



$$\left[(A+B')' + (A'+B')'\right]'$$

$$= \left[(A+B')'\right]' \cdot \left[(A'+B')'\right]'$$

$$= (A+B') \cdot (A'+B)$$

$$Y = \left[(A+B') \cdot (A'+B)\right]'$$

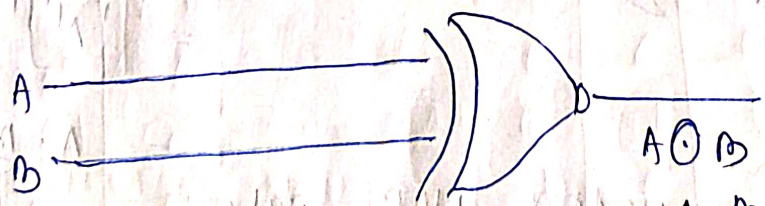
$$= (A+B')' + (A'+B)'$$

$$= (A+B')'$$

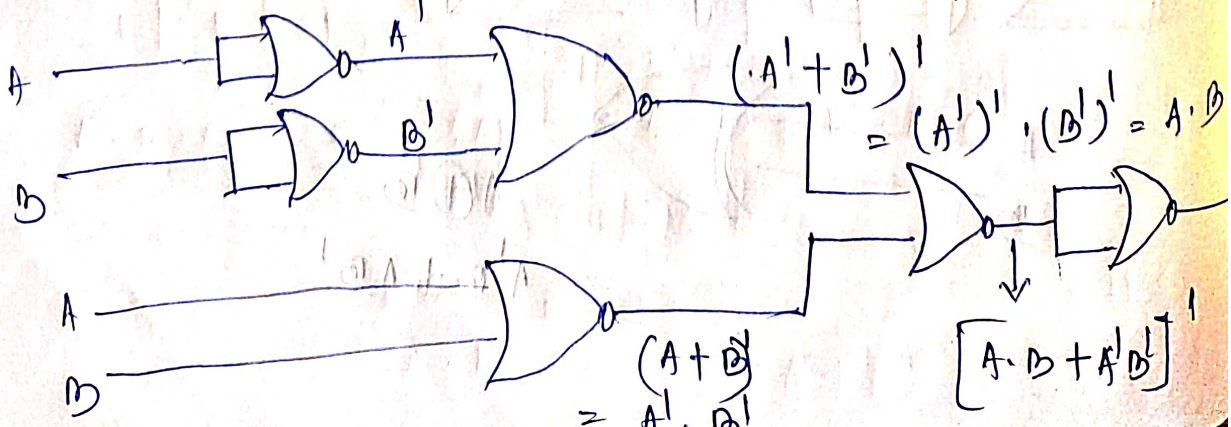
$$= A' \cdot B + A \cdot B'$$

$$= A \oplus B$$

Ex-NOR gate using NOR gate



$$= A \cdot B + A' \cdot B'$$

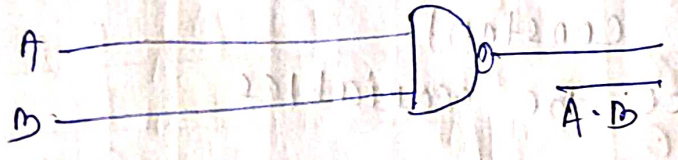


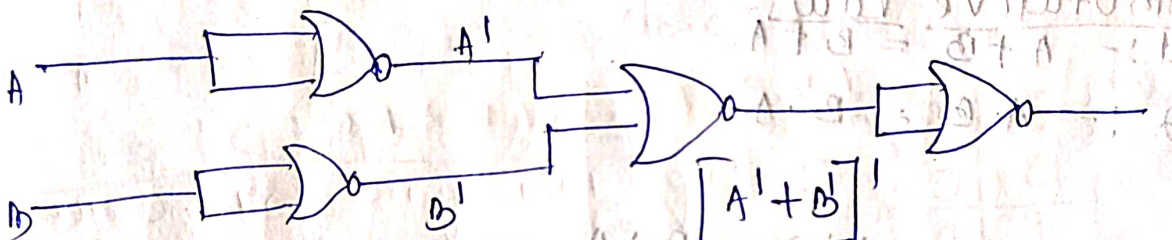
$$y = \left[\left[\overline{A \cdot B} + A'B'\right]'\right]'$$

$$= A \cdot B + A'B'$$

$$= A \odot B$$

NAND gate using NOR gate





$$y = \left[\left[A' + B'\right]'\right]'$$

$$= (A' + B')$$

$$= (A \cdot B)'$$

1.9 Different postulates and Demorgan's Theorem in Boolean Algebra

The Boolean Algebra are used to represent the digital signal in terms of Algebra.

→ The Boolean Algebra consists of three parts

(i) constant
(ii) variable
(iii) function

constant

In Boolean Algebra the constants are always same.

~~i.e. it~~ The constant in Boolean Algebra may be 0 or 1.

Variable :-

In Boolean Algebra the variables are changes that means the output of the boolean algebra may contains different variables.

function :-

In Boolean Algebra the functions contains more than one number of variables and constant.

Ex :- $F(A, B, C) = 1 + AB + BC$

Here 1 is the constant

A, B, C are the variables

Laws of Boolean Algebra

① Commutative law

Law 1 :- $A + B = B + A$

Law 2 :- $A \cdot B = B \cdot A$

Proof

Law-1 :- $A + B = B + A$

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

Law-2 :- $A \cdot B = B \cdot A$

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

$A \cdot B \cdot C = B \cdot C \cdot A = C \cdot A \cdot B = B \cdot A \cdot C$

(2) Associative laws :-

Law-1 :- $(A+B)+C = A+(B+C)$

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

Law-2 :- $(A \cdot B)C = A(B \cdot C)$

A	B	C	AB	(A·B)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

If 4 no. of variables are present then

$$A(BCD) = (ABC)D = (AB)(CD)$$

(3) Distributive Laws :-

Law 1 :- $A(B+C) = AB + AC$

Law 2 :- $A + BC = (A+B)(A+C)$

Proof Law :- 1 $A(B+C) = AB + AC$

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

A	B	C	AB	AC	AB + AC
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

Law :- 2

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

R.H.S $= (A+B)(A+C)$

$= AA + AC + AB + BC$

$= A + AC + AB + BC$

$= A(1 + C + B) + BC$ ∵ L.H.S = R.H.S

$= A(1 + B) + BC$

$= A \cdot 1 + BC = A + BC$ (Proved)

(4) AND Laws :-

$A \cdot 0 = 0$

$A \cdot 1 = A$

$A \cdot A = A$

$A \cdot \overline{A} = 0$

(5) OR Laws

$A + 0 = A$

$A + 1 = 1$

$A + A = A$

$A + \overline{A} = 1$

(6) NOT operation (Inversion Law) :-

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

(7) Absorption Law

Law 1 = $A + A \cdot B = A$

$A + A \cdot B$

$= A(1 + B)$

$= A \cdot 1$

$= A$

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

Law-2 = $A(A + B) = A$

$A(A + B)$

$= A \cdot A + A \cdot B$

$= A + AB$

$= A(1 + B)$

$= A \cdot 1$

$= A$

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

(8) Identity Law:-

$A + A = A$

$A \cdot A = A$

(9) Demorgan's Law:-

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

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

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

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

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

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

1.10 Use of Boolean Algebra for simplification of Logic Expression.

Logic Expressions are expressed by the help of Boolean Expression.

→ The expression may contains constants (0 & 1), variables & function.

→ usually the Boolean expression are expressed in two ways. There are

(I) sum of product (SOP)

(II) product of sum (POS)

(I) Sum of product (SOP)

It is an expression in which the product term are sum together.

Ex:- $A\bar{B} + A\bar{B}C + \bar{B}C$

Here $A \cdot \bar{B}$, $A \cdot \bar{B} \cdot C$, $\bar{B} \cdot C$ are the product term, Then the product terms are solve together by summed ~~them~~.

(II) Product of sums: (POS)

It is an Expression in which the sum terms are product together.

Ex:- $F = (A + B + C) \cdot (A + \bar{B} + C)$

Here $A+B+C$ & $A+\bar{B}+C$ are the summing term, then the summing terms are solve together by producted ~~them~~.

Min term:-

Each individual term in the standard SOP form is called Minterm.

Eg:- $F = \underbrace{A\bar{B}C} + \underbrace{A\bar{B}\bar{C}} + \underbrace{\bar{A}\bar{B}C}$

→ Minterms

→ It is represented by "mi"

Max term:-

Each individual term in the standard POS form is called Maxterm.

Eg:- $f = \underbrace{(A+\bar{B})}\ \underbrace{(A+\bar{B})}$

→ Max terms

→ It is represented by "Mi"

A	B	Minterms m_i	Maxterms M_i
0	0	$\bar{A}\bar{B} \rightarrow m_0$	$A+B \rightarrow M_0$
0	1	$\bar{A}B \rightarrow m_1$	$\bar{A}+\bar{B} \rightarrow M_1$
1	0	$A\bar{B} \rightarrow m_2$	$\bar{A}+B \rightarrow M_2$
1	1	$AB \rightarrow m_3$	$\bar{A}+\bar{B} \rightarrow M_3$

SOP → A = 1, $\bar{A}$ = 0 → Minterm → product term

POS → A = 0, $\bar{A}$ = 1 → Maxterm → sum term

standard SOP

~~when the function is a three variable function~~

A	B	C	Minterms	Maxterms
0	0	0	$\bar{A}\bar{B}\bar{C}$ m_0	$A+B+C$, M_0
0	0	1	$\bar{A}\bar{B}C$	$A+B+\bar{C}$, M_1
0	1	0	$\bar{A}B\bar{C}$	$A+\bar{B}+C$, M_2
0	1	1	$\bar{A}BC$	$A+\bar{B}+\bar{C}$, M_3
1	0	0	$A\bar{B}\bar{C}$	$\bar{A}+B+C$, M_4
1	0	1	$\bar{A}\bar{B}C$	$\bar{A}+B+\bar{C}$, M_5
1	1	0	$AB\bar{C}$	$\bar{A}+\bar{B}+C$, M_6
1	1	1	ABC	$\bar{A}+\bar{B}+\bar{C}$, M_7

* $Y = \sum m(3, 6, 7)$

$= m_3 + m_6 + m_7$

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

* $Y = \pi(4, 5, 7)$

$= M_4 \cdot M_5 \cdot M_7$

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

Canonical form / standard form :-

Boolean expression where each term contains all Boolean variables in their true or complemented form.

→ These product terms are nothing but the minterms.

→ sum of all minterms of "f" for which "f" assumes 1 is called canonical sop.

Ex:- $f = \bar{x}\bar{y}z + \bar{x}yz + x\bar{y}z + xyz$

$f = \sum m(1, 3, 5, 7)$

$= \sum(m_1 + m_3 + m_5 + m_7)$

Conversion of sop into canonical sop form :-

Step-1

Determine the maximum variable.

Step-2

Multiply "1" where term is missing.

Step-3

Simplify the boolean expression using Boolean Theorm.

* Convert $AB + A'C$ into canonical form ?

Ans:- (i) A, B, C

(ii) $A \cdot B \cdot 1 + A' \cdot 1 \cdot C$

$F = A \cdot B(C + \bar{C}) + A' \cdot (B + \bar{B}) \cdot C \quad \left(\because \begin{matrix} C + \bar{C} \\ B + \bar{B} \end{matrix} \right\} = 1)$

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

* Convert $AB + C$ into canonical form ?

Ans:- (i) A, B, C

(ii) $AB \cdot 1 + 1 \cdot 1 \cdot C$

(iii) $f(A, B, C) = AB \cdot 1 + 1 \cdot 1 \cdot C$

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

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

$= ABC + ABC' + C[AB + A\bar{B} + \bar{A}B + \bar{A}\bar{B}]$

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

Conversion of POS into canonical POS form :-

Step-1

First write the boolean expression and determine the maximum number of variables.

Step-2

Add 0 where the variables terms are missing.

Step-3

Simplify the boolean expression using Boolean Theorm.

* Convert the $F = (A'+B')(B'+C)(A+C)$ Boolean expression into canonical POS form.

Ans :- $F = (A'+B')(B'+C)(A+C)$

A, B, C

$F = (A'+B'+0)(0+B'+C)(A+0+C)$

$= (A'+B'+C\cdot C')(A\cdot A'+B'+C)(A+B\cdot B'+C)$

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

$(A+B+C)(A+B'+C)$

Conversion of SOP into the canonical POS :

Step-1

First write the boolean expression and determine the Maximum variable.

Step-2

Take the complement of given expression and simplify.

Step-3

Take once again complement.

* convert $F = AB' + BA'$ into canonical POS form.

Ans :- $F = AB' + BA'$

$$F' = [AB' + BA']'$$

$$= (AB')' \cdot (BA')'$$

$$= (A' + B)(B' + A)$$

$$(F')' = F = [(A' + B) \cdot (B' + A)]'$$

$$= [A'(B' + A) + B(B' + A)]'$$

$$= [A'B' + A'A + BB' + AB]'$$

$$= [A'B' + AB]'$$

$$= [A'B']' \cdot [AB]'$$

$$= (A + B)(A' + B')$$

~~* conversion of pos into sop~~

K-Map (Karnaugh Map)

→ TO simplifying the boolean expressions K-map Method is used.

→ K-map is the graphical representation of boolean expression.

→ for a Boolean expression consisting of n-variables number of cell required in K-map = 2^n cell.

Rules for K-Map

~~(I) No zeros allowed.~~

(I) groups can be vertical or Horizontal but can't be diagonals.

(II) overlapping allowed.

(III) Group should be as large as possible.

(IV) Group must contain 2^n cells.

Two Variable K-Map

For 2 variable K-map there are 2^n, is equals to $2^2 = 4$ number of squares is required.

A \ B	0	1
0	00	01
1	10	11

A \ B	0	1
0	$A'B'$	$A'B$
1	AB'	AB

* $F = \sum(0, 1)$

Ans :-

A \ B	0	1
A' 0	1	1
A 1	0	0

$A'B' + A'B$

$= A'(B' + B)$

$= A'$

Three - variable K-Map

For 3 variables there are $2^3 = 8$ number of squares is required.

A \ BC	00	01	11	10
0	000 $A'B'C'$ 0	001 $A'B'C$ 1	011 $A'BC$ 3	010 $A'BC'$ 2
1	100 $AB'C'$ 4	101 $AB'C$ 5	111 ABC 7	110 ABC' 6

* $F = \sum(0, 1, 2, 3)$

Ans :-

A \ BC	00	01	11	10
0	1 0	1 1	1 3	1 2
1	0 4	0 5	0 7	0 6

$F = A'$

* $F = A'B'C' + AB'C'$

Ans:-

A \ BC	00	01	11	10
0	1 (0)	0 (1)	0 (3)	0 (2)
1	1 (4)	0 (5)	0 (7)	0 (6)

$F = B'C'$

* $f(A, B, C) = \Sigma(0, 2, 4, 5, 6)$

Ans:-

A \ BC	00	01	11	10
0	1 (0)	0 (1)	1 (3)	1 (2)
1	1 (4)	1 (5)	1 (7)	1 (6)

$f = C' + AB'$

4- variable K-map :-

for 4- variable the number of squares are $2^n = 2^4 =$ 16 number of square is required.

AB \ CD	00	01	11	10
00	0000 $A'B'C'D'_0$	0001 $A'B'C'D_1$	0011 $A'B'CD_3$	0010 $A'B'CD'_2$
01	0100 $A'BC'D'_4$	0101 $A'BC'D_5$	0111 $A'BCD_7$	0110 $A'BCD'_6$
11	1100 $ABC'D'_{12}$	1101 $ABC'D_{13}$	1111 $ABCD_{15}$	1110 $ABCD'_{14}$
10	1000 $AB'C'D'_8$	1001 $A'BCD'_9$	1011 $AB'CD_{11}$	1010 $AB'CD'_{10}$

* $F = \Sigma(0, 1, 3, 5, 8, 9, 12, 13)$

Ans:-

AB \ CD	00	01	11	10
00	1 (0)	1 (1)	1 (3)	(2)
01	(4)	1 (5)	(7)	(6)
11	1 (12)	1 (13)	(15)	(14)
10	1 (8)	1 (9)	(10)	(11)

$F = AC' + C'D + AB'C' + A'B'CD$

* $F = A'BC'D + ABC'D + A'BCD + ABCD + A'B'C'D'$

Ans:-

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

$F = AB'C'D' + BD$

K-Map for POS form

(i) Two variable $2^n = 2^2 = 4$

A \ B	0	1
0	$A+B$ (0)	$A+B'$ (1)
1	$A'+B$ (2)	$A'+B'$ (3)

* $F = (A'+B)(A+B)$

Ans:-

A \ B	0	1
0	0 (0)	(1)
1	0 (2)	(3)

$F = B$

$F = (A'+B)(A+B)$

$= A'.A + A'B + BA + B$

$= 0 + A'B + BA + B$

$= B(A'+A) + B$

$= B.1 + B$

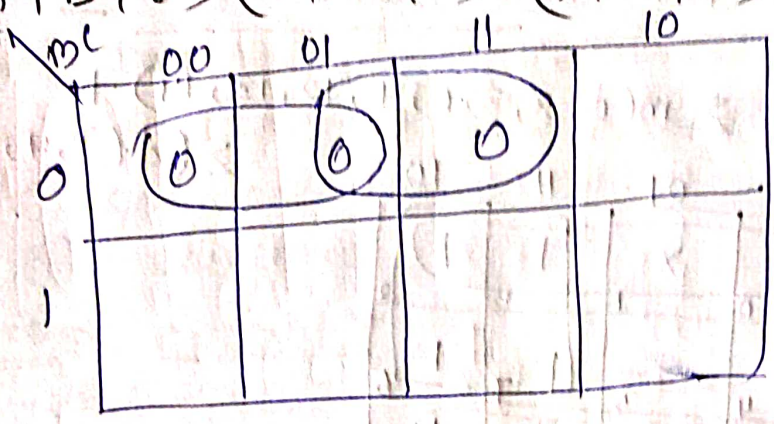
$= B$

(ii) Three variable $2^n = 2^3 = 8$

A \ BC	00	01	11	10
0	$A+B+C$ (0)	$A+B+C'$ (1)	$A+B'+C'$ (3)	$A+B'+C$ (2)
1	$A'+B+C$ (4)	$A'+B+C'$ (5)	$A'+B'+C'$ (7)	$A'+B'+C$ (6)

* $F = (A+B+\bar{C})(A+B'+C')(A+B+C)$

Ans:-



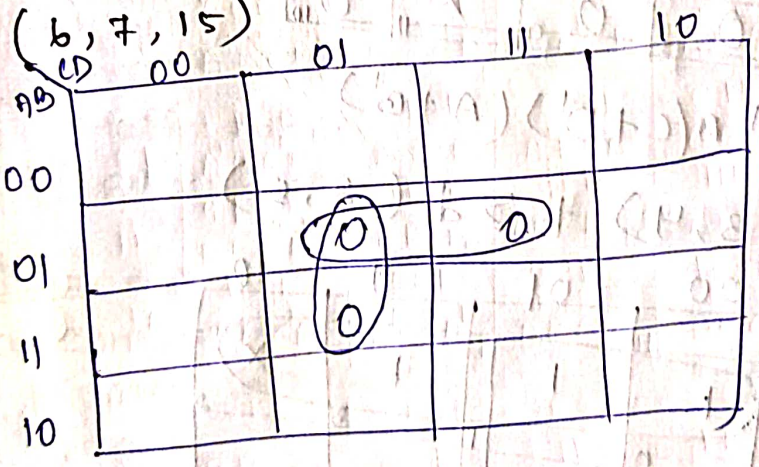
$f = (A+B)(A+C')$

(III) four variable $2^n = 2^4 = 16$

AB \ CD	00	01	11	10
00	A+B+C+D (0)	A+B+C+D' (1)	A+B+C'+D (3)	A+B+C'+D' (2)
01	A+B'+C+D (4)	A+B'+C+D' (5)	A+B'+C'+D (7)	A+B'+C'+D' (6)
11	A'+B'+C+D (12)	A'+B'+C+D' (13)	A'+B'+C'+D (15)	A'+B'+C'+D' (14)
10	A'+B+C+D (8)	A'+B+C+D' (9)	A'+B+C'+D (11)	A'+B+C'+D' (10)

* $f = \Pi(6, 7, 15)$

Ans:-



$f = (A+B'+C')\ (C'+D'+B')$

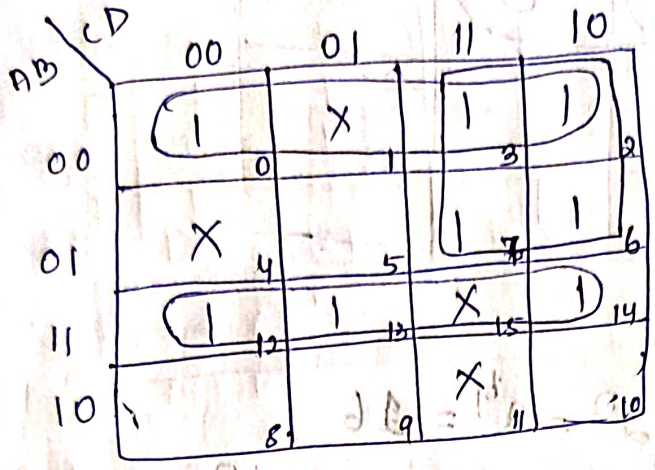
Don't care condition

→ when the place is not specified either it is one or zero, then Don't care condition is used.

→ It is usually denoted by "x" Mark - that means the value may be "0" or "1".

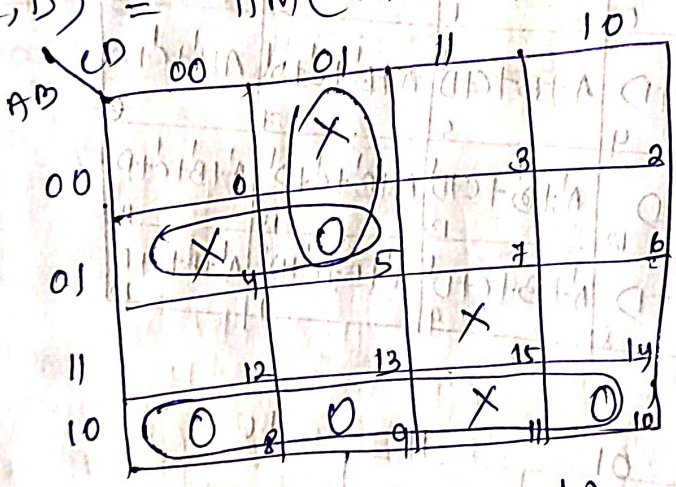
K-map simplification with Don't care condition :-

* $F(A, B, C, D) = \sum m(0, 2, 3, 6, 7, 12, 13, 14) + \sum d(1, 4, 11, 15)$



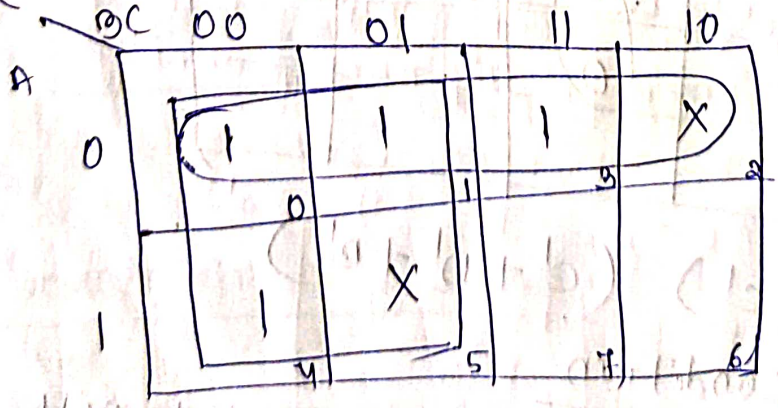
$f = A'B' + AD + A'C$

* $F(A, B, C, D) = \prod M(5, 8, 9, 10) \cdot \prod D(1, 4, 11, 15)$



$f = (A'+B) \cdot (C+D') \cdot (A+B')$

* $f = \sum(0, 1, 3, 4) + \sum d(2, 5)$



$F = A' + B'$

2. Combinational Logic Circuits

2.1 Give the concept of Combinational logic circuits.

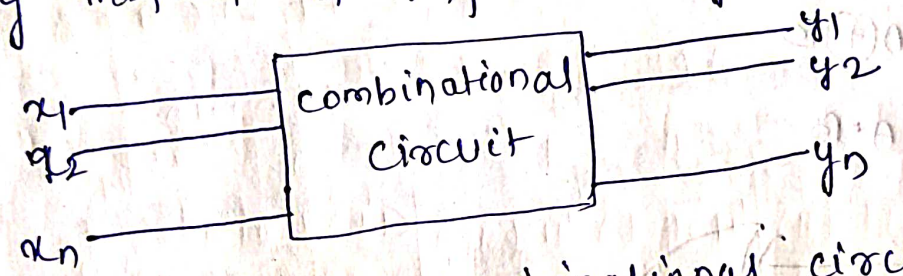
The Logic circuits are of two types. these are

(I) Combinational circuit

(II) sequential circuit

Combinational circuit:-

Combinational circuit is the type of logic ckt where at any instant of Input we get the output.



→ The Examples of combinational circuits are ADDER, SUBTRACTOR, DECODER, ENCODER, MULTIPLEXER, DEMULTIPLEXER,

ADDER

It is a combinational logic ckt which adds two bit or three bit binary numbers and produce the output sum and carry.

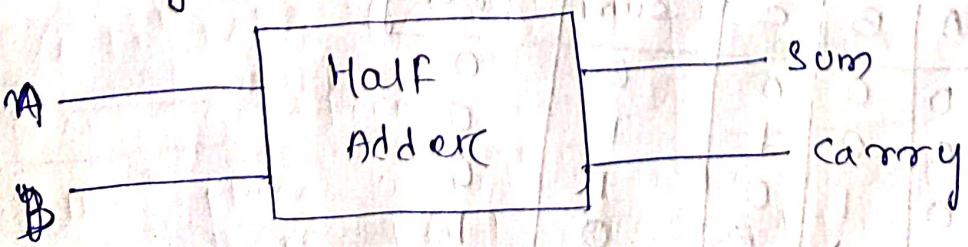
→ ADDER circuit are of two types such as

(I) Half Adder

(II) Full Adder

Half Adder

→ It is a combinational logic circuit which adds the two bit binary numbers and produce the output sum and carry.



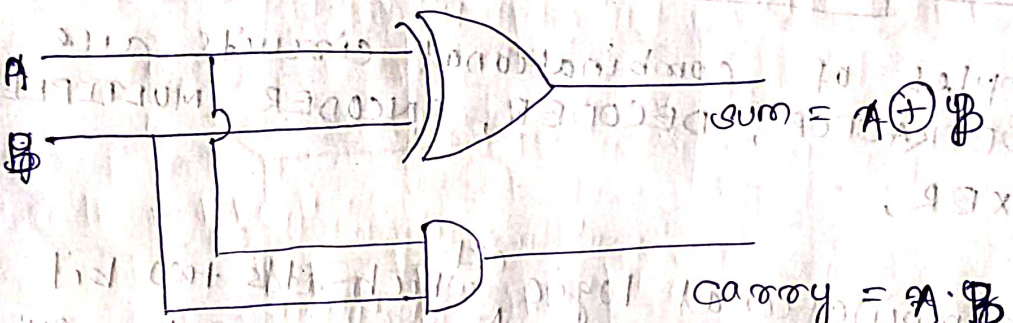
Truth Table

A	B	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$sum = A'B + AB'$

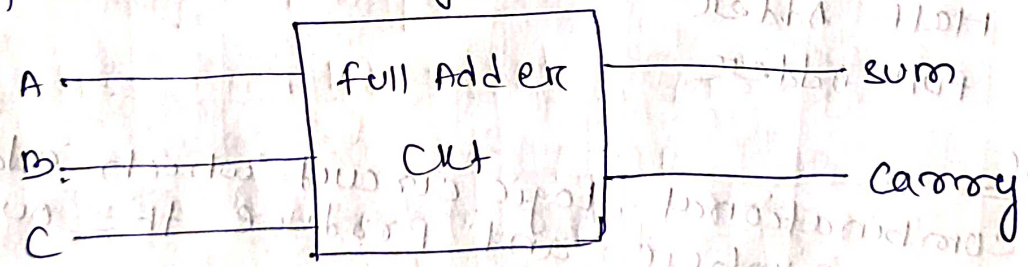
$= A \oplus B$

$carry = A \cdot B$



Full Adder :-

It is a combinational circuit that performs addition of three bit binary number and provides output sum and carry.



Truth Table

A	B	C	sum	carry
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

$$\text{Sum} = A'B'C + A'BC' + AB'C' + ABC$$

$$= A'(B'C + BC') + A(B'C' + BC)$$

$$= A'(B \oplus C) + A(B \odot C)$$

$$= A'\underbrace{(B \oplus C)}_{x} + A\underbrace{(B \oplus C)}_{x'}$$

$$= A'x + Ax'$$

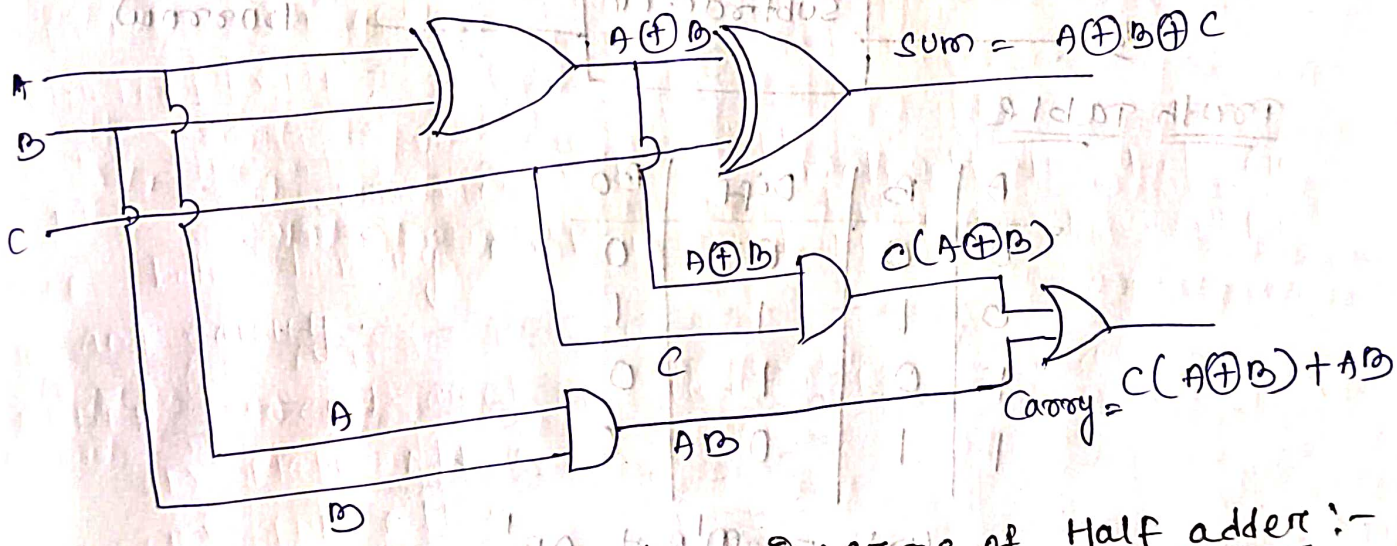
$$= A \oplus x$$

$$= A \oplus B \oplus C$$

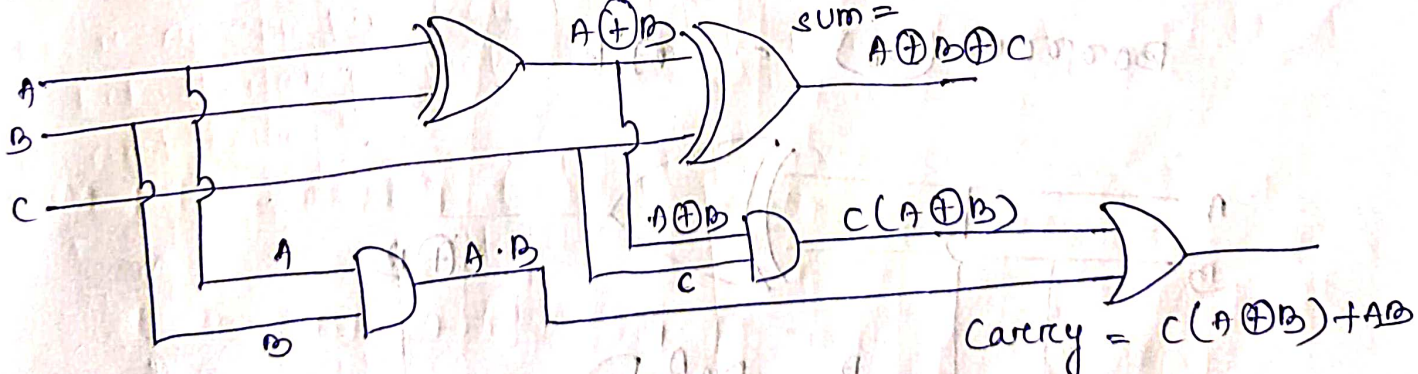
$$\text{carry} = A'BC + AB'C + ABC' + ABC$$

$$= C(A'B + AB') + AB(C' + C)$$

$$= C(A \oplus B) + AB$$



Representation of full Adder interms of Half adder :-



Subtractor

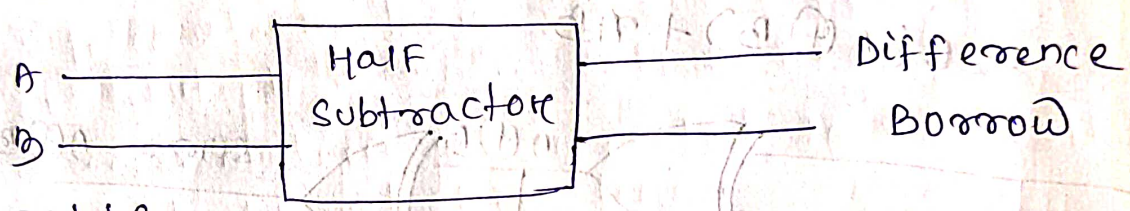
It is a combinational logic ckt which subtract two bit or three bit number and produce the output difference and Borrow.

→ subtractor are of two types such as

(i) Half subtractor
(ii) full subtractor

(i) Half subtractor

Half subtractor is a combinational logic ckt which subtract two bit number and produce the output difference and borrow.



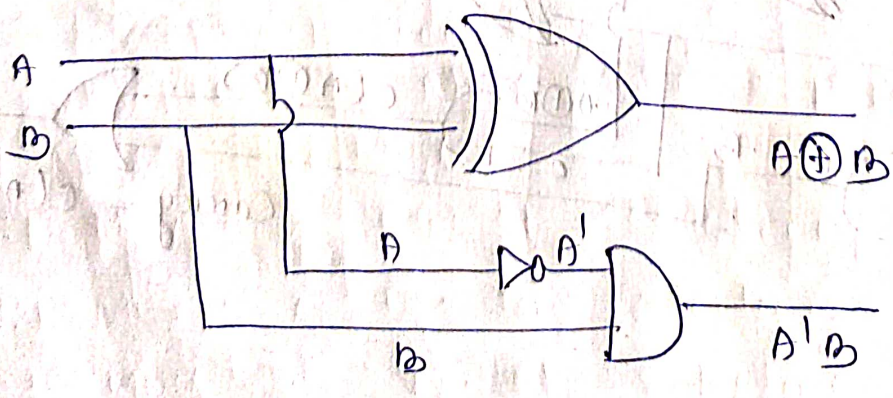
Truth Table

A	B	Diff	Bo
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Difference (D_{iff}) = $A'B + AB'$

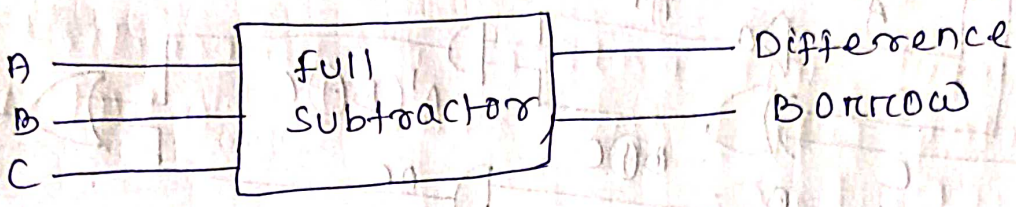
$= A \oplus B$

Borrow (B_0) = $A'B$



full subtractor :-

It is a combinational circuit that perform subtraction of three bit binary number and provide the output Difference and Borrow.



Truth Table

A	B	C	Diff	Bo
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

$$Diff = A'B'C + A'BC' + AB'C' + ABC$$

$$= A'(B'C + BC') + A(B'C' + BC)$$

$$= A'(B \oplus C) + A(B \odot C)$$

$$= A'\underline{(B \oplus C)} + A\underline{\overline{(B \oplus C)}}$$
$$\quad\ x \qquad\qquad\quad x$$

$$= A'x + Ax'$$

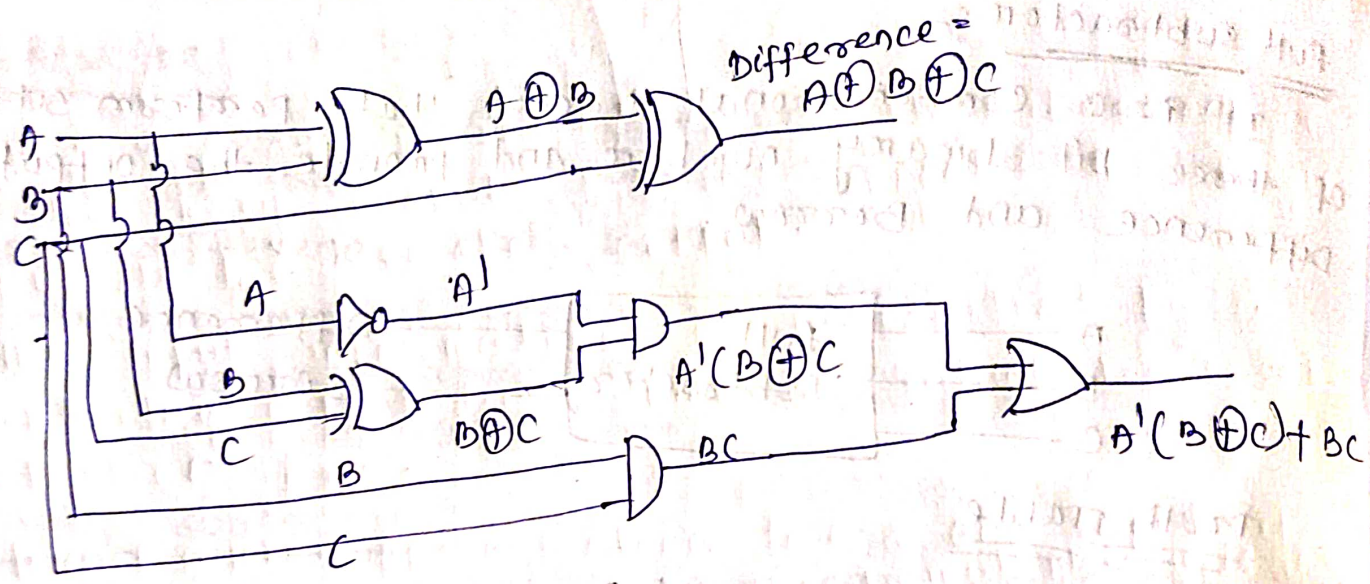
$$= A \oplus x$$

$$= A \oplus B \oplus C$$

$$Bo = A'B'C + A'BC' + A'BC + ABC$$

$$= A'(B'C + BC') + BC(B' + A)$$

$$= A'(B \oplus C) + BC$$

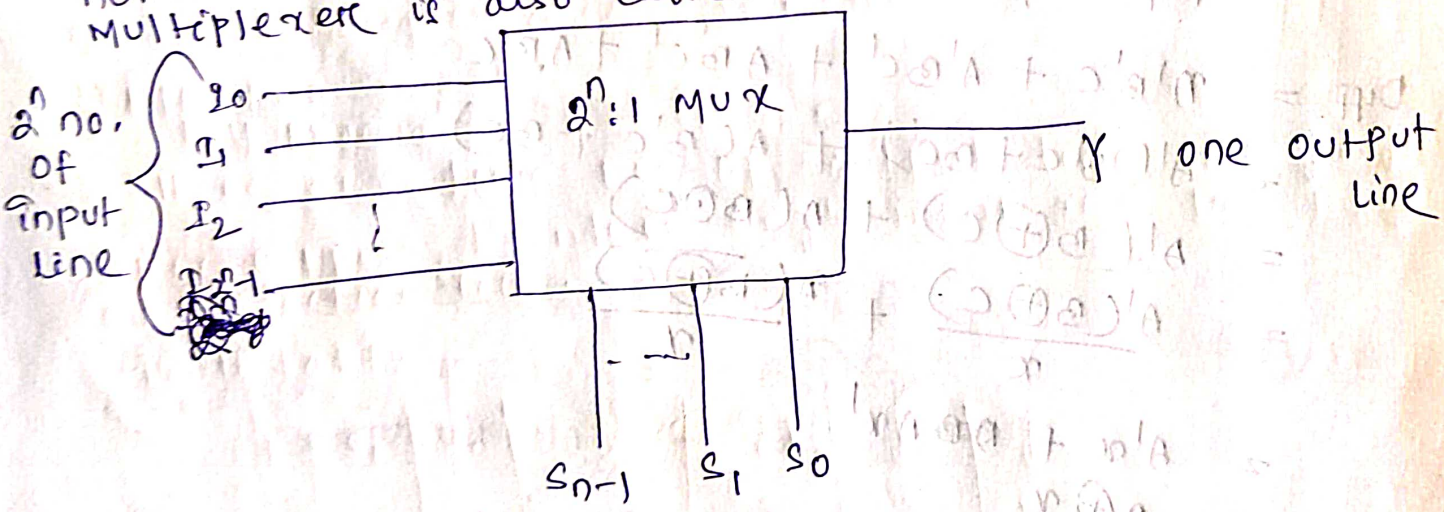


Multiplexer (MUX) :-

It is a combinational Logic circuit that receives binary information from several inputs and transmits the information to a single output line.

→ The input information is connected to the output lines by a set of selection line.

→ In order to select 2^n no. of input lines the number of selection line required is "n". The Multiplexer is also called a data selector.



n = no. of selection line

→ The examples of MUX are 2:1 MUX, 4:1 MUX, 16:1 MUX, 8:1 MUX

2:1 MUX → no. of selection Line is 1

4:1 MUX → no. of selection Line is 2

8:1 MUX → no. of selection Line is 3

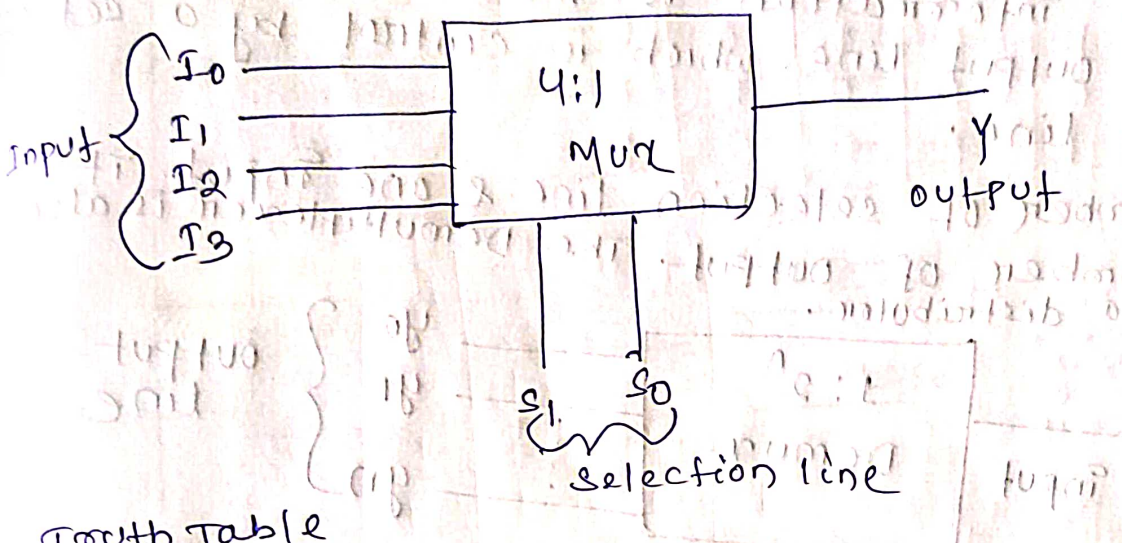
16:1 MUX → no. of selection Line is 4

4:1 Mux :-

→ In 4:1 Mux there are four no. of inputs and there is one output.

→ The output depends on the number of selection line.

→ Here the no. of selection line is 2.

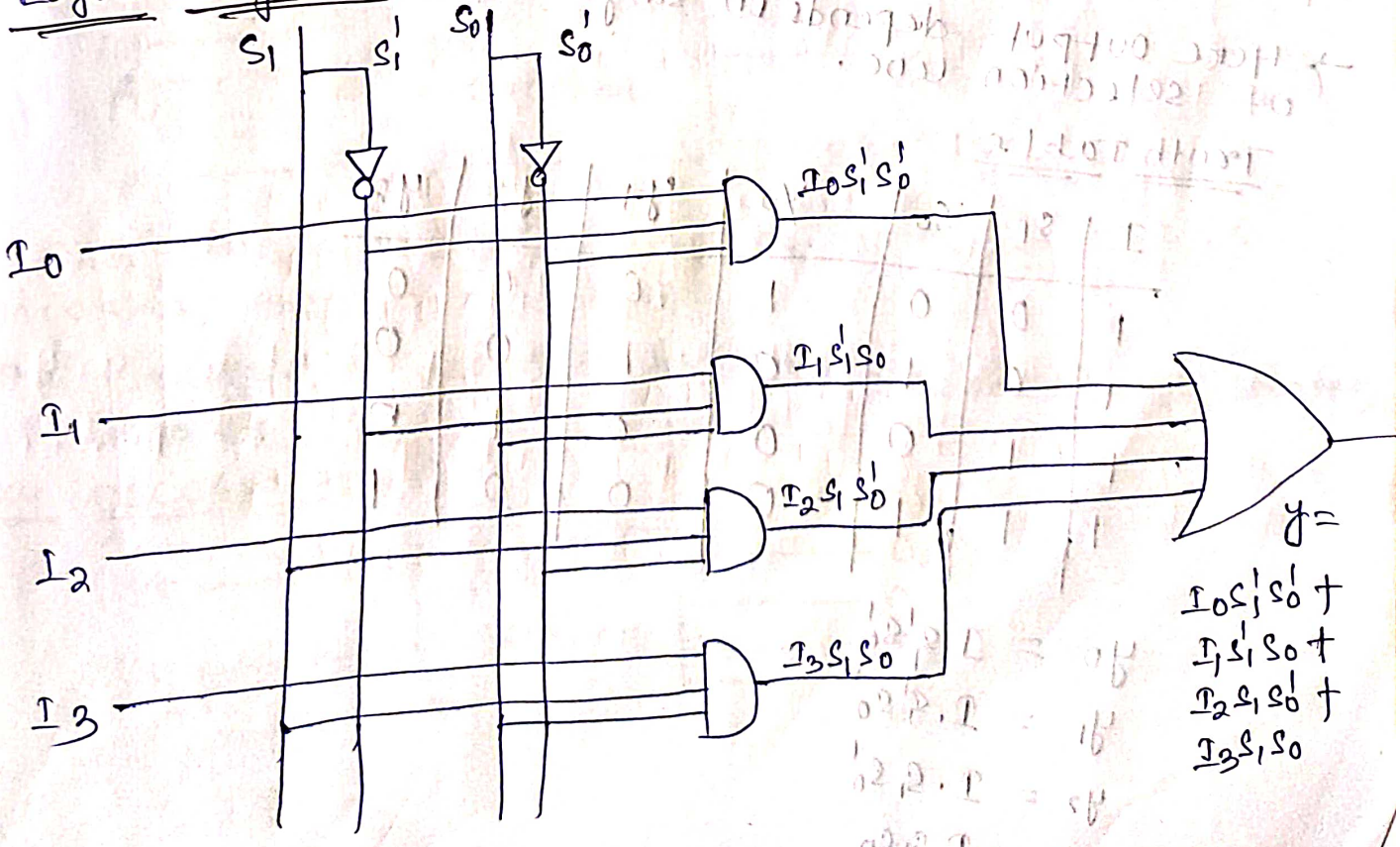


Truth Table

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

$Y = I_0 S_1' S_0' + I_1 S_1' S_0 + I_2 S_1 S_0' + I_3 S_1 S_0$

Logic Diagram

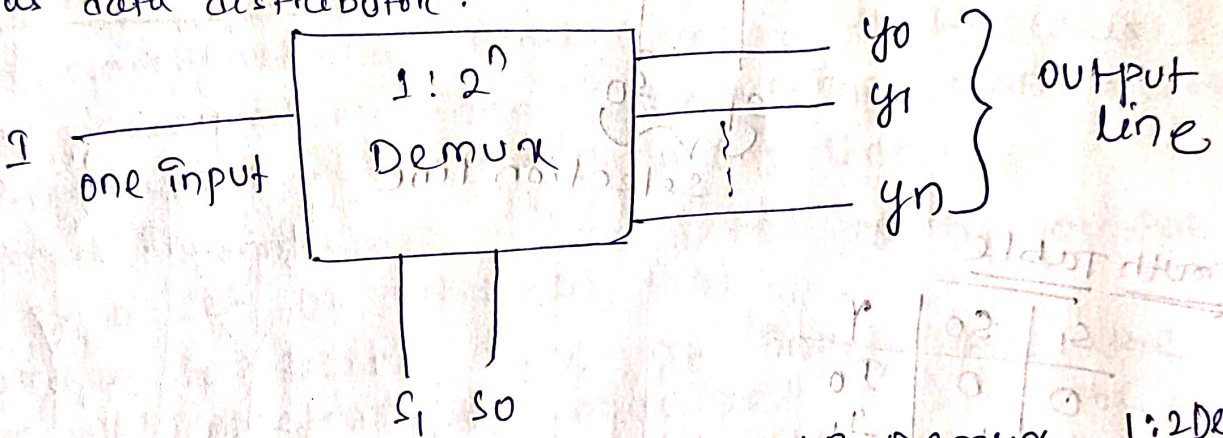


Demultiplexer (DeMux) :-

Demux is a combinational logic circuit that select one of the many output line and connects the single input to the selected output.

→ The input information is transmitted by a particular output line which is control by a set of selection line.

→ for n number of selection line & one input it has 2^n number of output. The Demultiplexer is also known as data distributor.



→ The examples are 1:4 Demux, 1:8 Demux, 1:2 Demux 1:16 Demux etc.

1:4 Demux

→ There are four no. of output line i.e y_0, y_1, y_2, y_3.

→ Here output depends on single input and two number of selection line.

Truth Table

I	S_1	S_0	y_0	y_1	y_2	y_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

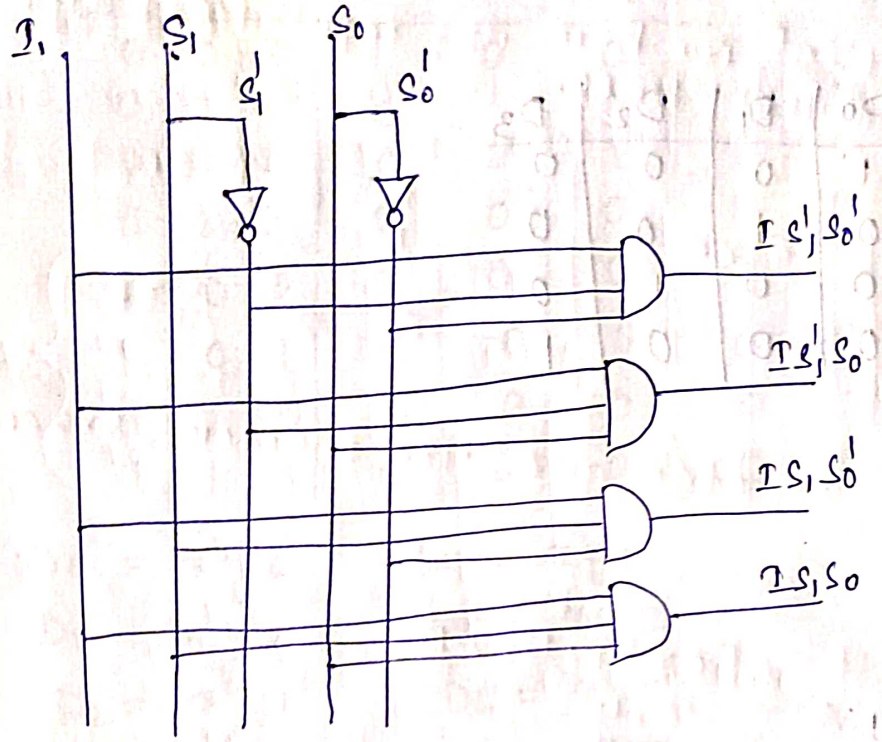
$y_0 = I S_1' S_0'$

$y_1 = I . S_1' S_0$

$y_2 = I . S_1 S_0'$

$y_3 = I . S_1 S_0$

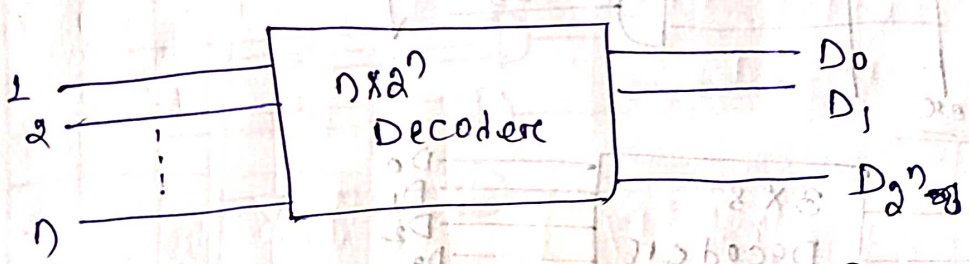
Logic Diagram



Decoder :-

A Decoder is a combinational logic circuit that convert the information from n input lines to the Maximum 2^n no. of output line.

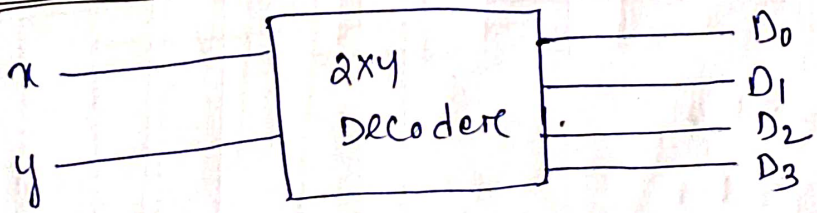
→ Hence at a time only one output line is high out of the 2^n no. of output lines & other output lines are Low i.e zero.



→ In $n \times 2^n$ line n indicates the no. of inputs & 2^n indicates the no. of output.

→ The example of decoders are 2x4 Decoder, 3x8 Decoder, 4x16 Decoder.

2x4 Decoder :-



→ Here two input lines are present i.e x & y

→ Here four output lines are present i.e D_0, D_1, D_2, D_3

Truth Table

x	y	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

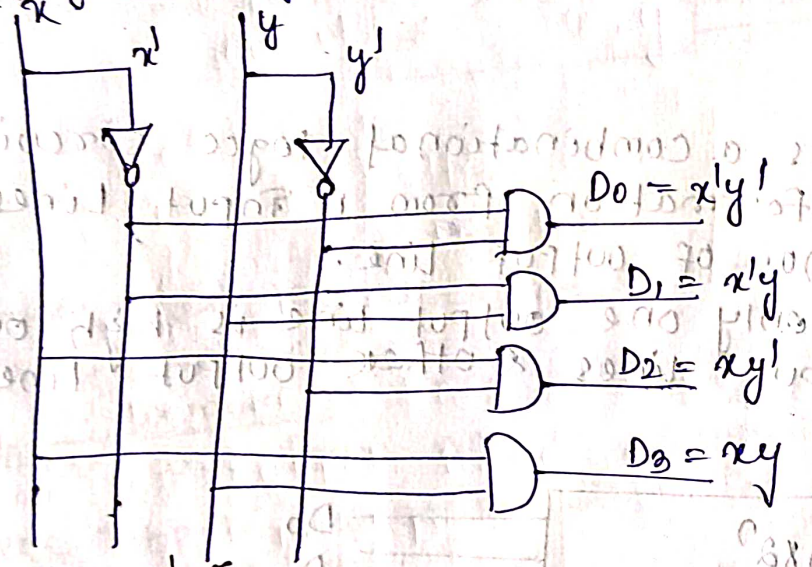
$D_0 = x'y'$

$D_1 = x'y$

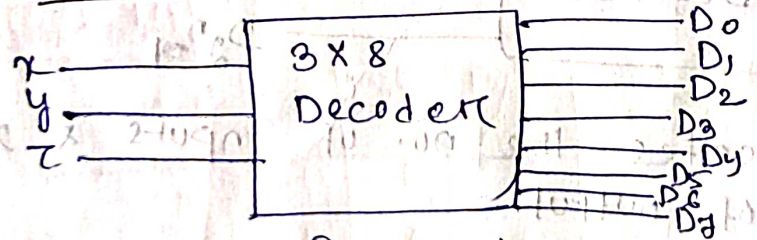
$D_2 = xy'$

$D_3 = xy$

Logic Diagram



3X8 Decoder



→ Here three input lines are present i.e x, y, & z.

→ Here eight output lines are present i.e D_0, D_1, D_2 D_3, D_4, D_5, D_6, D_7

Truth Table

x	y	z	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

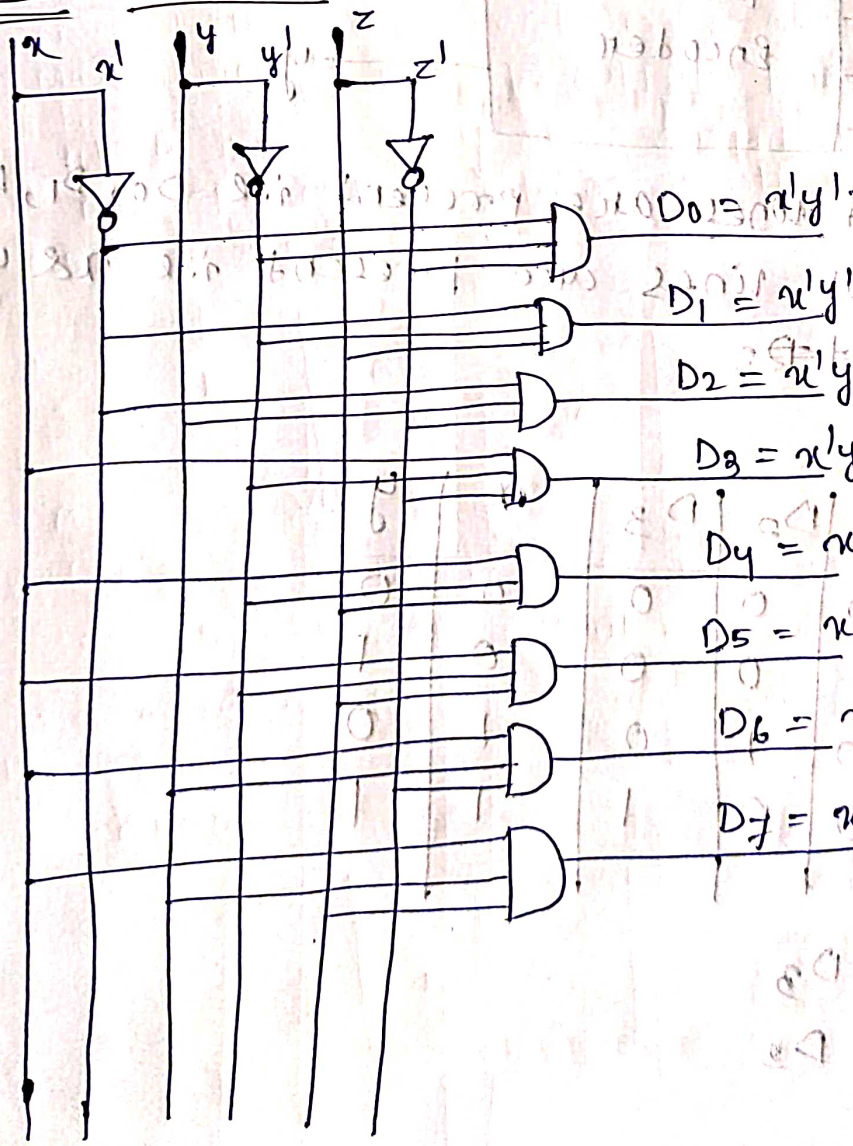
$D0 = x'y'z'$ $\quad D4 = xy'z'$

$D1 = x'y'z$ $\quad D5 = xy'z$

$D2 = x'yz'$ $\quad D6 = xyz'$

$D3 = x'yz$ $\quad D7 = xyz$

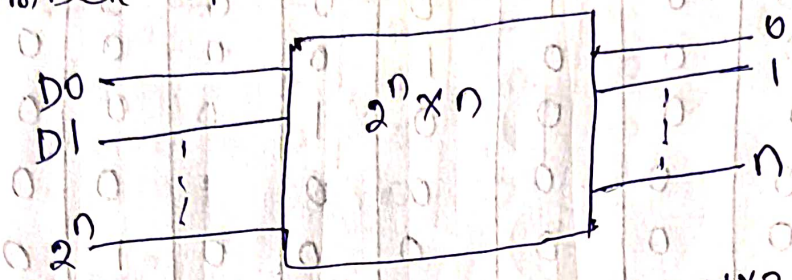
Logic Diagram



ENCODER :-

It is a combinational logic circuit that perform inverse of decoder.

→ An encoder has 2^n number of input lines and "n" number of output lines.

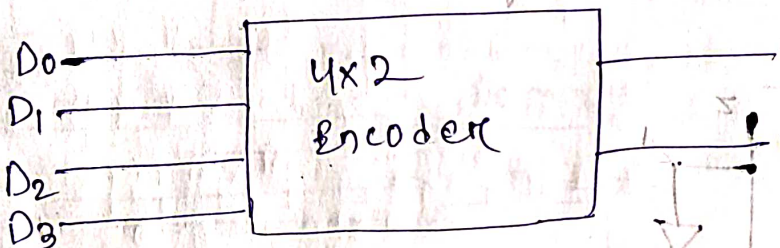


→ The example of encoder are 4×2 encoder, 8×3 encoder, 16×4 encoder.

4×2 Encoder :-

input = 4 D_0, D_1, D_2, D_3

output = 2 x, y



→ Here four input lines are present i.e D_0, D_1, D_2, D_3

→ Here two output lines are present i.e x & y

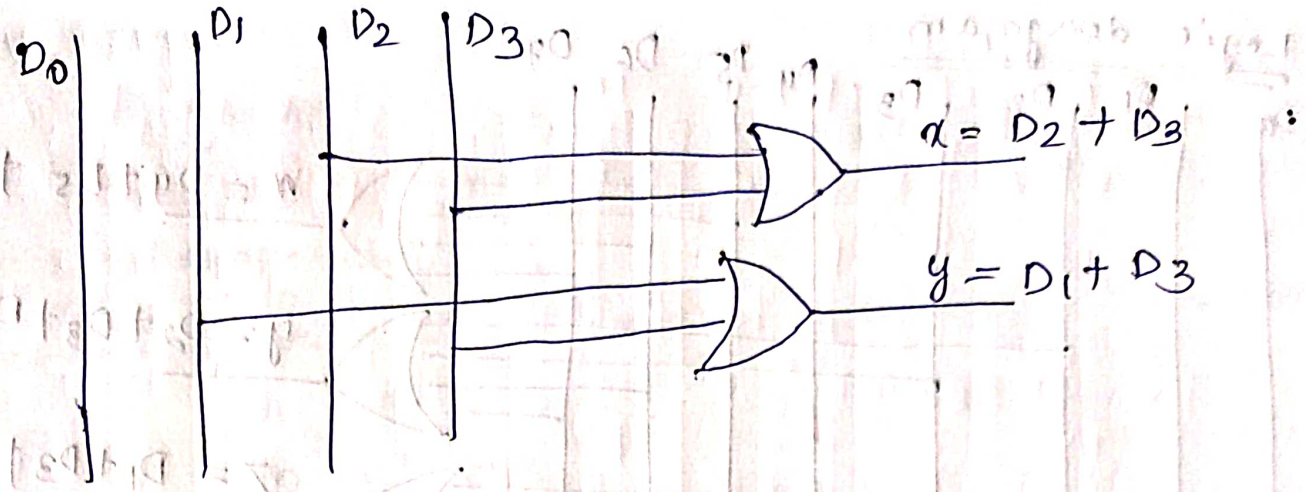
~~$x = D_2 + D_3$~~

Truth table

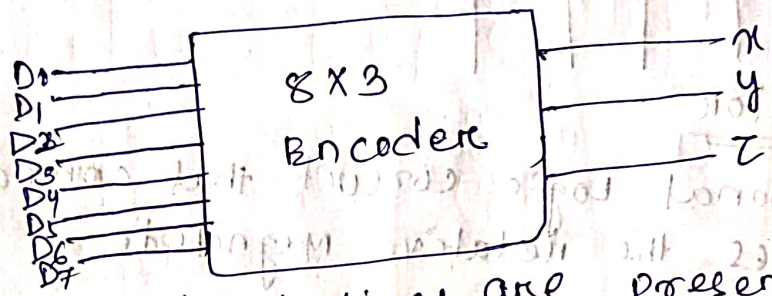
D_0	D_1	D_2	D_3	x	y
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

$x = D_2 + D_3$

$y = D_1 + D_3$



8X3 Encoder :-



→ Here eight input lines are present i.e D0, D1, D2, D3, D4, D5, D6 & D7

→ Here three outputs lines are present i.e x, y & z

Truth Table

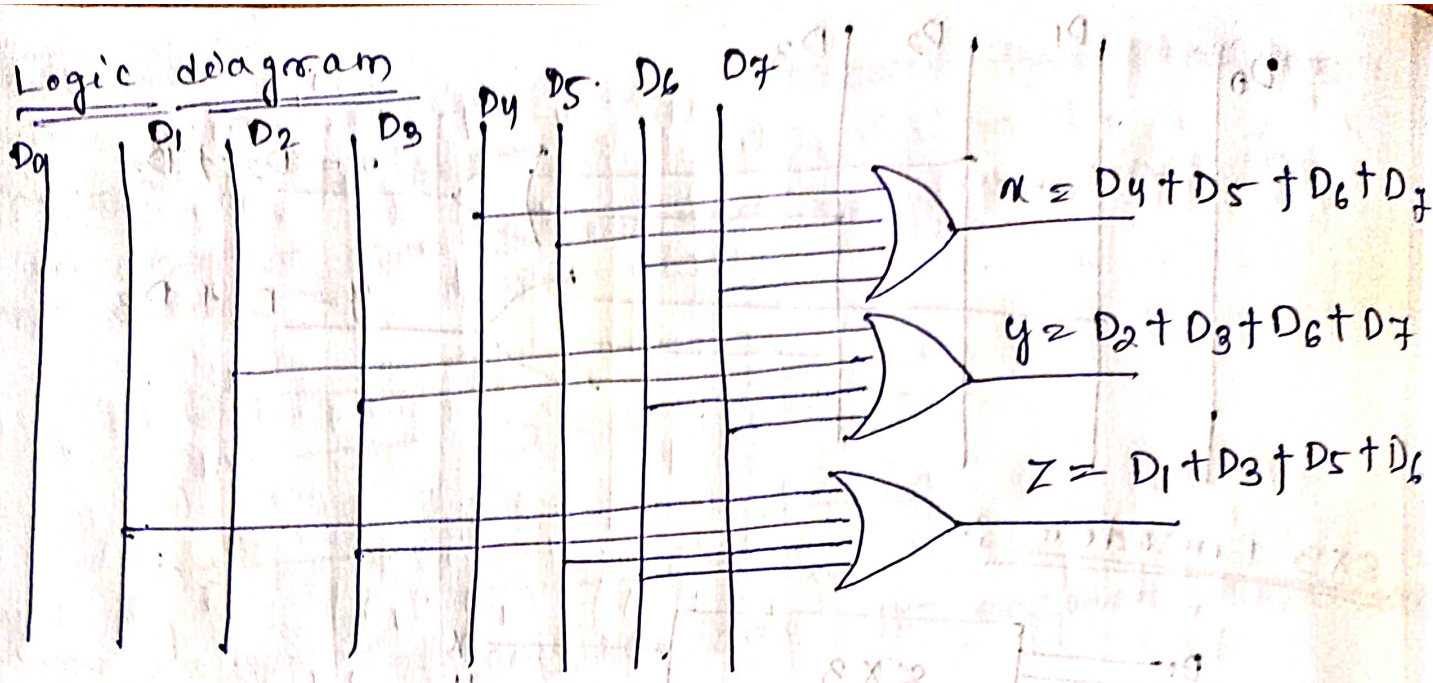
D0	D1	D2	D3	D4	D5	D6	D7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$x = D_4 + D_5 + D_6 + D_7$

$y = D_2 + D_3 + D_6 + D_7$

$z = D_1 + D_3 + D_5 + D_6$

Logic diagram



Magnitude Comparator

It is a combinational logic circuit that compares two no. & determines the relative magnitude.

→ In Magnitude comparator there are three condition
i.e. (i) $A = B$
(ii) $A < B$
(iii) $A > B$.

Single bit Comparator

A	B	A = B	A < B	A > B
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	1	0	0

for $A = B$

$$y = A'B' + AB$$
$$= A \odot B$$

when $A \odot B = 1$, then $A = B$

for $A < B$

$$y = A'B$$

when $A'B = 1$, then $A < B$

For A > B

$y = AB'$

when $AB' = 1$ then A > B

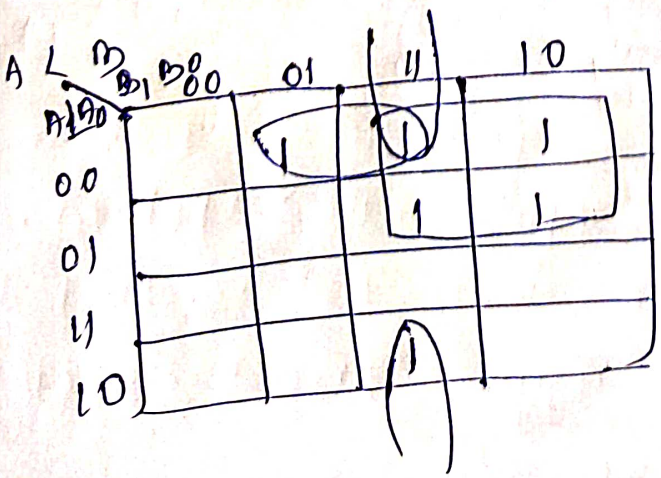
Two bit Comparator :-

consider two no. A and B with two digits each.

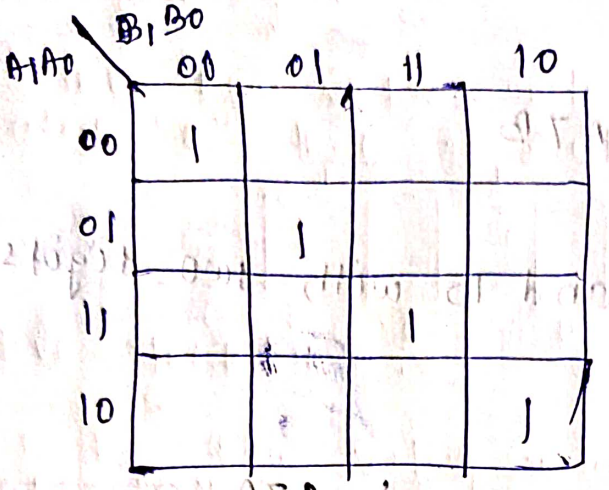
$A = A_1 A_0$

$B = B_1 B_0$

A_1	A_0	B_1	B_0	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

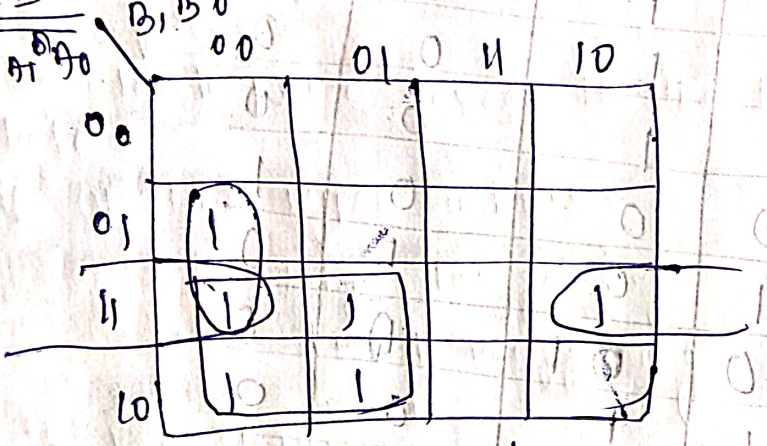


$A_1' B_1 + A_1' A_0' B_0 + A_0' B_1 B_0$



$A = B = (A_1 \odot B_1)\,(A_0 \odot B_0)$

$A > B$



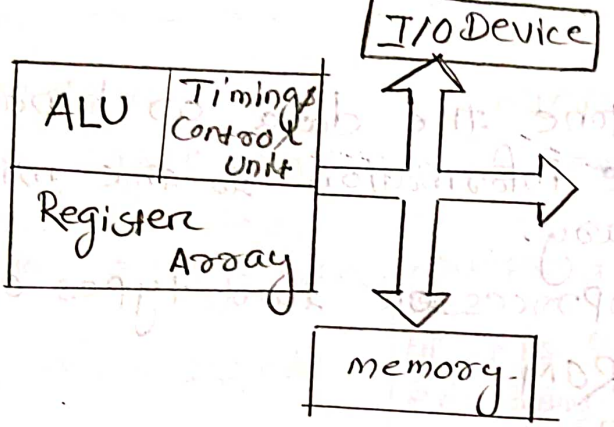
$A_1 B_1' B_0 + A_1 A_0 B_0 + A_1 B_1'$

New → {8085 MICROPROCESSOR} CH-4

Introduction to microprocessor, Microcomputer:

Microprocessor: It is a semiconduton device which takes digital data Processes according to the instruction and gives the result in digital from ~~with~~, with in small interval of time.

Block diagram of microprocessor:



Arithmatic & logic unit (ALU):

→ ALU Perforons arithmatic & logic operation.
⇒ The arithmatic operations are addition, subtraction, Multiplication, division,
⇒ It also Peforms the logical operation like OR, AND, NOR, NAND, etc.

Time & Control Unit:

⇒ It acts as a brain of a Computer.
⇒ It controls all the operation of micro Processer. It also Controls input device, output device & all omer device Connected to CPU.
⇒ It Controls flow of data between microprocessor memory & generates the signals which are required from all the operation to be Peform by CPU & required for Control of input device. These signals are called timing.

Timing & control unit :-

→ It acts as a brain of a computer.
→ It controls all the operation of Microprocessor. It also controls input device, output device & all other device connected to CPU.
→ It controls flow of data between Microprocessor memory. It generates the signal which are required for all the operation to be perform by CPU & required for control of input device. These signals are called timing signals.

Register Array :-

→ Registers are used to store the data temporarily during the execution of program.
→ Register Array is the no. of registers present in Microprocessor. Register Array are Accumulator, general purpose register, special purpose register.

Memory :-

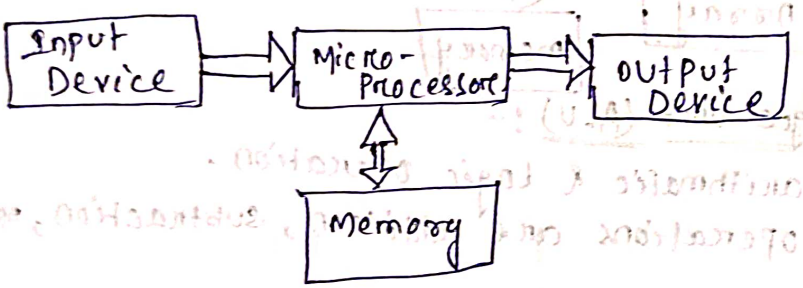
→ The Memory store the data or binary information.
→ It provide the information to the Microprocessor when it is necessary.
→ usually in Microprocessor two types of memory are used
  ① ROM
  ② RAM

I/O Device :-

→ I/O device is a input & output device.
→ I/O device are used to communicate with outside the Microprocessor.
→ The I/O devices are also called as peripheral devices.
Ex:- printer, Scanner

Microcomputer

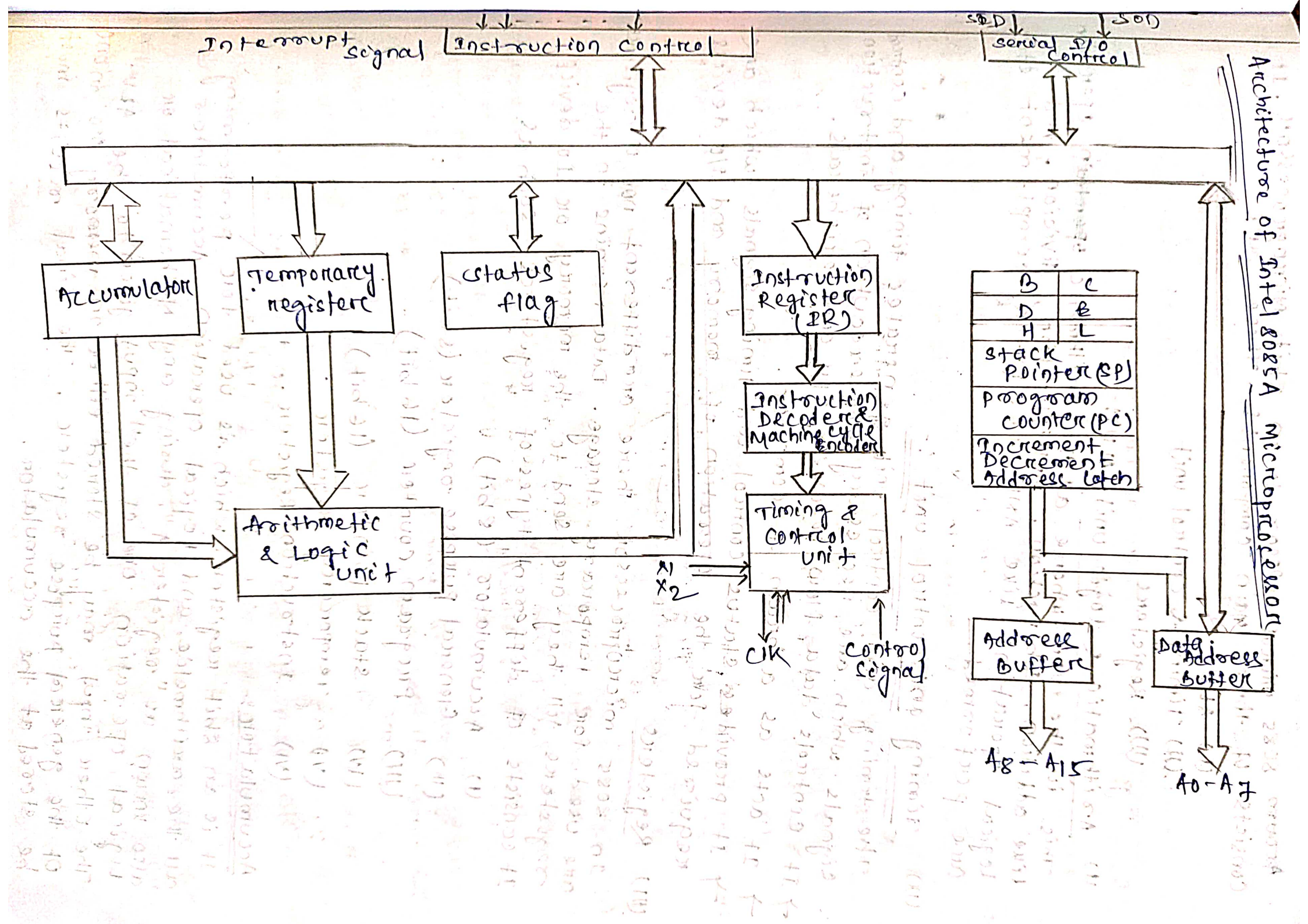
Micro computer is a computer which has only one microprocessor in its CPU.
Ex:- Laptop, PC



→ The device through which input are given is called input device.
Ex:- keyboard.
→ The device through which output are taken is called as output device.
Ex:- monitor

Distinguish between Microprocessor and Microcomputer

Microprocessor	Microcomputer
(i) It is cpu on a single chip. It includes the ALU, small memory register & arrays on a single chip. It is of 8 bit, 16 bit or 32 bit type processor.	(i) It designed using a microprocessor as its CPU along with input, output, & memory.
(ii) It is used in instructions, automatic testing product, speed control of motors, traffic light control, light control of furnaces etc.	(ii) It is used in business and education including word processing spreadsheets and Logical operations.
(iii) Its clock frequency is about 1GHz.	(iii) It use clock frequency is up to 50MHz.
(iv) Its cost is high.	(iv) Its cost is low.


Interrupt signal
Instruction control
SID
SOD
serial I/O control
Accumulator
Temporary register
status flag
Instruction Register (IR)
Instruction Decoder & Machine cycle encoder
Timing & control unit
Arithmetic & Logic unit
X1
X2
CLK
control signal
B C
D E
H L
stack Pointer (SP)
program counter (PC)
Increment Decrement Address latch
Address buffer
Data Address buffer
A8 - A15
A0 - A7
Architecture of Intel 8085A Microprocessor

Accurm 8085 Microprocessor is an 8bit Microprocessor. It consists of three main components. These are

(I) ALU
(II) Timing & control unit
(III) Registers

(I) Arithmatic & logic unit (ALU):-

This is the unit where all the arithmetic operation like addition, subtraction, Multiplication, division, and logical operations like AND, OR, EX-OR, complement etc, are performed.

(II) Timing and control unit :-

The timing and control unit generates timing and control signals which are necessary for execution of instruction.

→ It controls data flow between CPU and Perciphercals.
→ It acts as a brain of computer.
→ It provides status control and timing signals which are required for the operation of memory and I/O devices.

(III) Registers

In 8085 microprocessor there are different no. of registers are used for temporary storage. Data remains in the registers till they are send to the memory or I/O device. It consists of different types of registers such as

(I) Accumulator (8bit)
(II) General purpose register (8bit)
(III) Program counter (16 bit)
(IV) Stack pointer (16 bit)
(V) Temporary Register.
(VI) Instruction Register.

Accumulator :-

It is an 8bit register which is used for performing all the arithmetic and Logical operation. Accumulator is also known as "register A", during any Mathematical or Logical operation one of the input given to the ALU. The other input may be stored either in memory or in one of the general purpose register. The final results must be stored at the accumulator.

(II) General purpose Register :-

→ 8085 has 6 different general purpose register. These are B, C, D, E, H, L.

→ Each registers are 8bit.

→ To hold 16 bit data a combination of two 8bit register can be used the combination of two 8bit register is called as register pair.

→ The valid register pairs are B-C, D-E, H-L. Always H-L register pair are used for the memory.

(III) program counter :-

→ It is a 16bit special purpose register It is used to hold the memory location of the next instruction to be execute.

→ Initially it indicates towards the starting address of the program but after the first instruction is fetched the program counter automatically gets incremented by one and points towards the next instruction. This process continues till the end of program.

(IV) stack pointer :-

→ It is a 16 bit special purpose register. It basically serves two purposes.

(a) It indicates the beginning of the stack memory, whenever some thing is added to the stack, the stack pointer is decremented and whenever something is removed from the stack, the stack pointer always points to the top of the stack.

(b) stack pointer also points towards the memory location where the programer to store the content of register during the execution of program.

(V) Instruction Register :-

Instruction register holds the opcodes of instruction which is being decoded & executed.

(VI) Temporary Register :-

The temporary register are used by the Microprocessor for storing the data temporarily during execution of a program. They are 8bit registers and are not accessible to the users.

Flags :-

The flags is concede of flip flop. The flip flops are set (1) or reset (0) according to the condition during execution of arithmatic & logical operation.

→ flags are of 5 types. These are

(I) Carry flag
(II) parity flag
(III) Auxiliarcy carry flag
(IV) Zero flag
(V) Sign flag

(I) Carry flag :- (CS)

After the execution of arithmatic or logical operation if the carry is produce then the carry flag is set to 1. when there is no carry it said to zero.

(II) Parity flag :- (P)

The parity flag is set to 1 when there is even no. of 1 in the result of arithmatic or logical operation. It is reset if the result contains odd no. of one.

(III) Auxiliary carry flag :- (AC)

The auxiliary carry flag holds the carry out of bit no. 3 to bit no. 4 durcing the execution of arithmatic or logical operation.

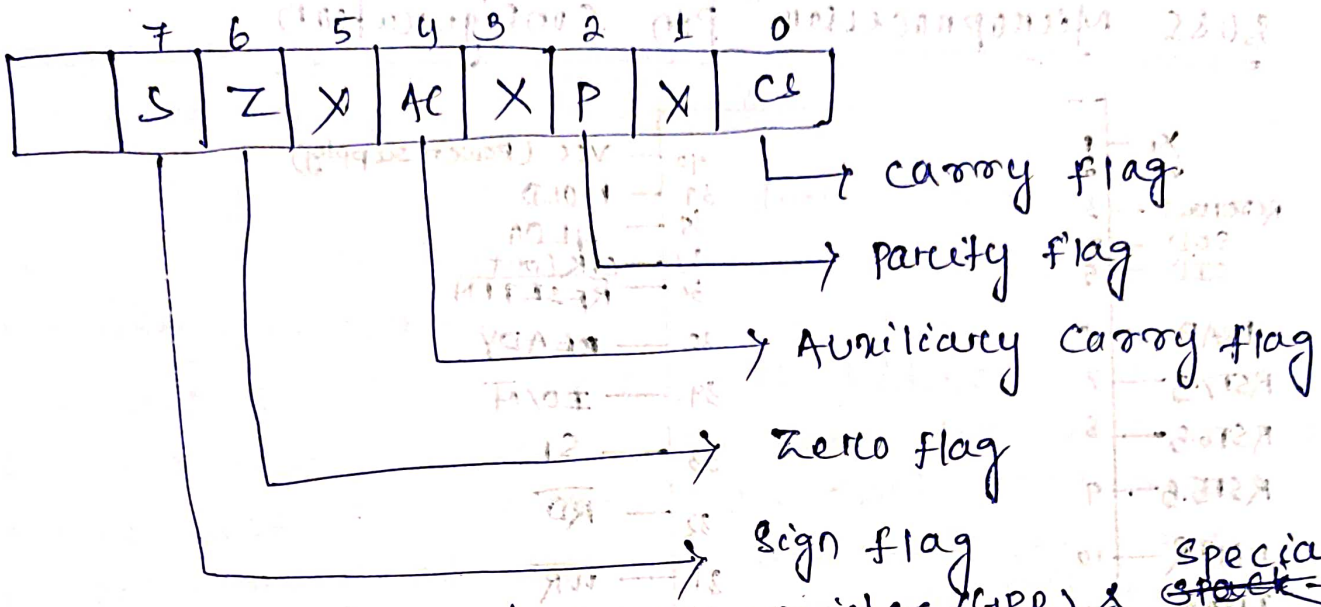
→ when there is carry from bit no. 3 to bit no. 4 that means auxiliarcy carcy set to 1 or otherwise reset (0).

(IV) Zero flag (Z) :-

when the result is zero then the zero flag is set 1. when the result is not zero then the zero flag is reset 0.

(V) Sign flag (S) :-

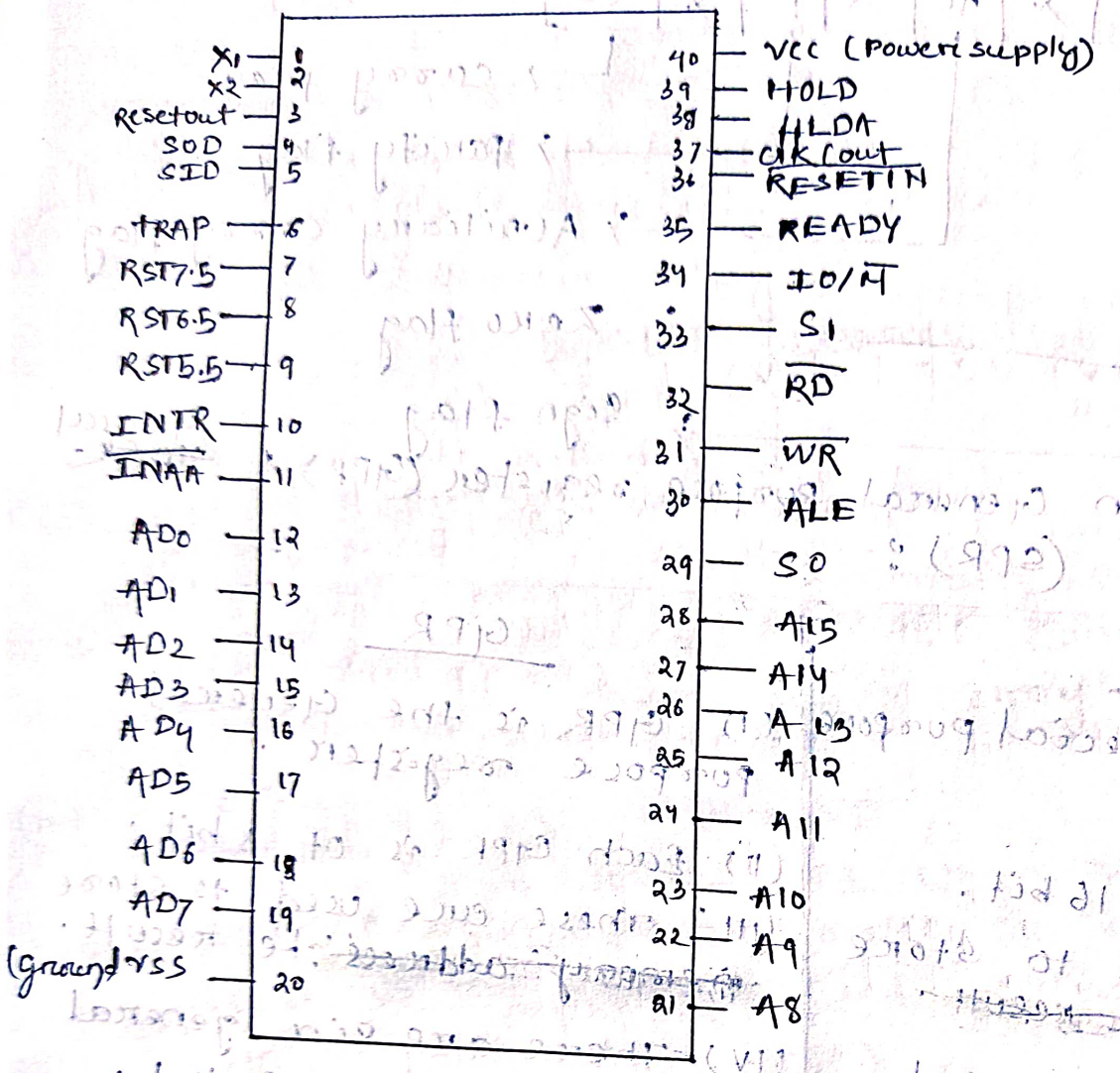
The sign flag is set to 1, if the result of arithmatic or logical operation is negative. It is positive then it is set to zero.



Different between General purpose register (GPR) & Special ~~Purpose~~ register (SPR) :-

SPR	GPR
(I) SPR is the special purpose register.	(I) GPR is the General purpose register.
(II) Each SPR is of 16 bit.	(II) Each GPR is of 8 bit.
(III) These are used to store the memory address.	(III) These are used to store the intermediate result.
(IV) There are two special purpose register in 8085 i.e stack pointer and program counter.	(IV) There are six general register i.e B, C, D, E, H, L.
(V) It cannot be used in pair form.	(V) It can be used in pair form to store 16 bit data or address i.e (B-C), (D-E) (H-L).
(VI) It is not user accessible.	(VI) These registers are user accessible.

8085 Microprocessor pin configuration



(Pin no. 21628)

→ 8085 Microprocessor has 8 bit data bus that means Do to D7 and 16 bit address bus that is A0 to A15,

→ 8085 Microprocessor is a 40 pin IC.

→ The clock period of Microprocessor is 3 MHz.

→ The 40 pin of 8085 Microprocessor are classified into 6 section.

→ These are (I) address Bus
(II) Multiplexed Data Bus
(III) control signal status.
(IV) Power supply and ground.
(V) Externally initiated signal,
(VI) Serial Input output ports.

(i) Address Bus :-

Address Bus is divided into two part. The lower 8 bit that is A0 to A7 is called lower address Bus.
→ A8 to A15 signal lines are unidirectional and used high order address bus.

(ii) Multiplex ~~Address~~ Data Bus :-

the signal line AD0 to AD7 are bidirectional. They are used for dual purpose. The data bus is a group of 8-lines used for transfer the data. They are used as low-order address Bus aswell as data Bus, therefore they are known as Multiplexed address Bus.

(iii) Control and status signal :-

This group of signals includes two control signals three status signals and one special signal. These signals are as follows.

~~Control~~.

ALE (Address Latch Enable) (pin no. 30) :-

It is a positive going pulse. generate every time to indicate beginning of operation.

$\overline{RD}$ (pin no. 32) :-

when $\overline{RD}$ goes slow it indicates that IO device or memory is reading.

$\overline{WR}$ (pin no. 31) :-

It is a control signal to write the operation. when $\overline{WR}$ goes slow it indicates that the IO device or memory is writting.

IO/$\overline{M}$ (pin no. 34) :-

It is a status signal which is used to identify between IO device and memory.
→ when IO/$\overline{M}$ goes high it indicates IO device.
→ when IO/$\overline{M}$ goes low it indicates memory device.

S_1 and S_0 (pin no. 33 and 29) :-

It is a status signal send by Microprocessor to testing use of different operation.

S_1	S_0	operation
0	0	HALT
0	1	WRITE
1	0	READ
1	1	FETCH

(iv) Power supply and ground :-

VCC (pin no. 40) :-

The VCC is used to provided +5V power supply.

Vss (pin no. 20)

It is used for ground connection.

x_1 and x_2 (pin no. 1 and 2)

The crystal is used to provide oscillation.

CLK (Pin no. 37)

The clk is used to generate the clk pulse for other device.

(v) Externally initiated signal :-

when Io device become ready to transport the data it sends a high signal to Microprocessor through a special line called as interrupt line.

→ 8085 Microprocessor has 5 different interrupt.

These are (i) INTR
(ii) Trap
(iii) RST 7.5
(iv) RST 6.5
(v) RST 5.5

INTR (Interrupt Request) (Pin no. 10) :-

This is used as a general purpose interrupt and it has lowest priority.

INTA (Interrupt Acknowledgement) (Pin no. 11) :-

It is interrupt acknowledgement send by Microprocessor after receiving interrupt request.

RST 7.5, RST 6.5, RST 5.5, Trap (pin no. 9, 8, 7, 6) :-

These are the restart interrupt that transfer the program control to a specific memory location.

TRAP has the highest priority. After that the priority order are RST 7.5, RST 6.5, RST 5.5 among the three interrupt.

HOLD :- (Pin no. 39)

It indicates another devices is requesting for address & data bus.

READY :- (Pin no. 35)

The READY signal is used by the microprocessor to check the peripheral is ready or not to transfer or receive the data.

RESET IN (Pin no. 36) :-

When RESET IN is low then microprocessor is said to 0. usually the ~~microprocessor is said~~ program counter present inside the microprocessor is RESET.

RESET OUT (Pin no. 3)

~~The READY signal is used by the Microprocessor to check the~~

RESET OUT (Pin no. 3)

It indicate that the reset out can be used to reset other device.

(VI) Serial input output ports :-

8085 has two signals to implement serial transmission.

SID (serial input data) (Pin no. 5) :-

It is data line for serial input.

SOD (Serial output data) (pin no. 4) :-

It is a data line for the serial output.

Opcode :-

The opcode or the operation code is the first part of an instruction which specifies the task to be perform by the Microprocessor.

Operand :-

The operand is the second part of the instruction which contains the data to be operated on.

Stack

Stack is an area of memory identified by the programmer for temporary storage of information. Stack works on the principle of, LIFO structure. The data to be stored or retrieved to/from the stack will always in double bytes. Stack is used to store data in register pairs BC, DE, HL.

Stack pointer

The stack is implemented with the help of special memory pointer register is called as stack pointer. The stack pointer's contents are automatically adjusted to point to the stack.

Stack TOP

The memory location that is currently pointed by the stack pointer is called stack top.

Instruction set

An instruction is a command given to the computer or Microprocessor to perform a specified operation or task on the given data.

→ The instruction set is a collection of instruction that the Microprocessor execute.

→ Different types of instruction sets are there

(I) Data Transfer group
(II) Arithmatic group
(III) Logic group
(IV) Branching group.

Instruction word size

According to the word size the instructions are classified into three types.

① 1 byte instruction/one word.
② 2 byte instruction/two word.
③ 3 byte instruction/three coord.

One byte instruction :-
In 1-byte instruction the opcode and operand of an instruction are represented in one byte. The length of these instructions is 8 bit. each required one memory location.

Two byte instruction :-
Two byte instruction is the type of instruction in which the first 8 bit indicates the opcode and the next 8 bit indicates the operand. This type of instructions need two memory location to store the binary codes.

Three byte instruction :-
Three byte instruction is the type of instruction in which the first 8 bit indicates the opcode, the next two byte specify the 16 bit address. The low order address is represented in second byte, and the high order address is represented in the third byte. This instructions need three memory location to store the binary code.

1) Data Transfer group
It is a group of instruction which is used to transfer the data between two register or register & memory.

Example :-
MOV → The content of register/memory is moved to memory/register.

```
MOV B, A ;
MOV B, M ;
MOV M, C ;
```

MVI → Move the immediate data to memory/register

MVI A, 05

MVI M, 08

LXI → This instruction loads 16 bit immediate data into register pair.

LXI H, 2500H

LDA → The content of Memory location whose address is specified by the 2nd & 3rd byte of instruction is loaded into the accumulator.

LDA 2400 H

(1) Arithmetic group

It is a group of instruction which performs the arithmatic operation such as addition, subtraction, Multiplication, division etc.

Example:-

ADD → The content of register/memory location is added to the content of accumulator The sum is placed in the accumulator.

ADD B

ADD M

ADC → The content of register/memory location and carry status are added to the content of the accumulator. The sum placed in the accumulator.

ADC B

SUB → The content of register/memory location is subtracted from the content of accumulator & result is placed in the accumulator.

SUB C

(III) Logic group :-

It is a group of instruction which performs the logical operation such as AND, OR, EX-OR etc.

ANA → The content of register/memory location is ANDed with the accumulator & result is placed in the accumulator.

ANA M
ORI Data

(IV) Branch group :-

The Branch instruction instruct the Microprocessor to get the different location.

→ The Branch instruction are classified into three types. These are (I) Jump instruction
(II) Call & return instruction
(III) Restart instruction.

Jump instruction :-

The jump instruction is used to jump from one sequence of memory location to one sequence of memory location.

→ The jump instructions are of two types such as
(I) conditional jump
(II) unconditional jump

(I) conditional jump :-

when a program jumps from one sequence of memory location to the another sequence of memory location with a condition then it is called as conditional jump.

EX :- JZ addr

The instruction indicates jump to the address if the result is zero flag.

* JNZ addr

This instruction indicates jump to the address if the result is not zero flag.

(II) Unconditional Jump :-

When a program jumps from one sequence of location to the another with out any condition, then it is called as unconditional jump.

EX :- JMP addr

Jump the address.

CALL Instruction

The call instruction is used to called a subroutine. The program jumps to subroutine starting at address specified by the label.

EX :- CALL 2600H

Subroutine :-

A subroutine is a group of instruction written separately from the main program to perform a function that occurs repeatedly in Main program.

→ for subroutine usually we use CALL & RETURN instruction.

Bus structure of 8085 Microprocessor

System Bus :-

The input & output device & memory device are connected to a CPU by group of lines is called as system Bus.

→ The System Bus are of 3 types These are

(I) Address Bus
(II) Data Bus
(III) control Bus

Address Bus :-

Address Bus is a unidirectional group of 16 lines bits flow in one direction from the microprocessor to the peripheral device.

→ In 8085 Microprocessor the Address bus is 16 bit i.e $A_0 - A_{15}$. $A_0 - A_7$ is the lower address bus & $A_8 - A_{15}$ is the higher address Bus.

Data Bus :-

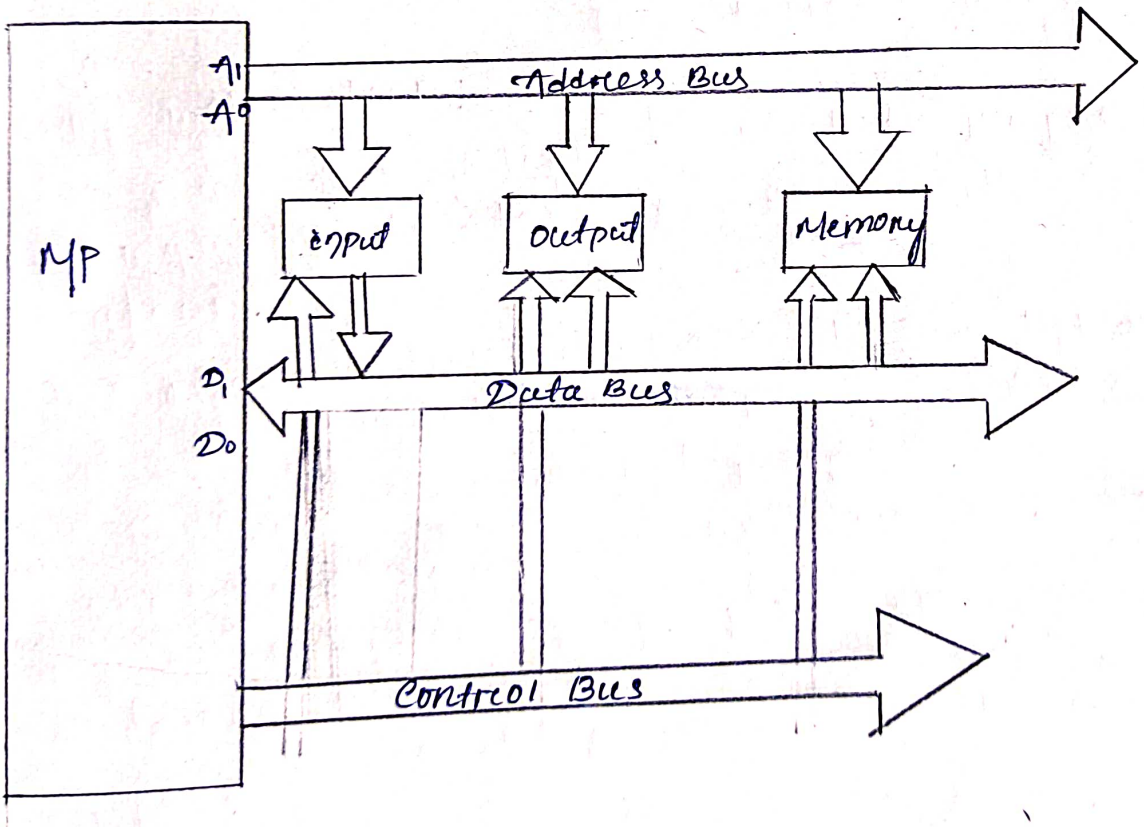
Data bus carries data in binary from between Microprocessor and peripheral device as well as memory.

→ The Data bus is bidirectional.

→ In 8085 Microprocessor data bus is 8 bit i.e AD0 - AD7

Control Bus :-

The control Bus is combination of various single lines that carry control signal. The control lines are not group links like address and data bus. Microprocessor generate ~~various~~ specific control signals for every operation it performs. It is used for transmitting and receiving the synchronized control signal between Microprocessor & other device.



Addressing Modes :-

8085 Microprocessor can able to handle 8 bit data & 16 bit of address location.

→ Each location having two parts, i.e. operand & opcodes.

→ The task be perform is called op-code & the data to be operated is called operand.

→ Each instruction required certain data on which it has to operate. There are various techniques are used to specify data for instruction.

→ The technique which is used to perform a task on operand are called Addressing Modes.

→ The addressing Mode are of five types.
These are