I - ASSIGNMENT

(Start Writing From Here)

1) Convert the following decimal numbers to the indicated bases.

(a) 7562.45 to octal.

first consider the integral part and divide it with 8.

8 | 7562
8 | 945 - 2
8 | 118 - 1
8 | 14 - 6
1 - 6.

∴ The binary number of 7562 is 16612.

Now, consider the decimal part, i.e., 0.45.

And multiply the number with 8.

⇒ 0.45 x 8 = 3.60

⇒ [∵ consider the part after the decimal part and multiply it with 8]

⇒ 0.60 x 8 = 4.8

⇒ 0.8 x 8 = 6.4

⇒ 0.4 x 8 = 3.2

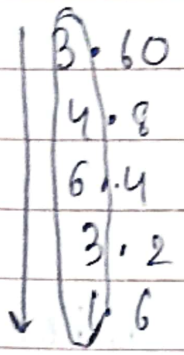
⇒ 0.2 x 8 = 1.6

[so, now we have to do the same process until we get zero]

But for some problems we get a lengthy and long solution. So we can consider upto 4 - 5 digits.

Consider the decimal before digit from the order, up to down that means top to bottom.

⇒



Consider these values and combine it with the binary Number

⇒ Binary number = 16612.

∴ 7562.45 into octal format is;

$(16612.34631)_8$

(b) 1938.257 to hexa decimal

Step ① :- Seperate the given number and write ~~binary~~ hexadecimal number for decimal part i.e.,

1938 → into bit hexadecimal

that means, divide the number with 16.

16 | 1938
16 | 121 - 2
7 - 9

∴ The hexadecimal number of 1938 is 792.

Step-②:-

Consider the floating point and multiply it with 16.

⇒ 0.257 × 16 = 4.112
⇒ 0.112 × 16 = 1.792
⇒ 0.792 × 16 = 12.672
⇒ 0.672 × 16 = 10.752

[∵ we can write, 12 as → C and 10 as → A].

So, the final answer is $(792.41CA)_{16}$

∴ 1938.257 into hexadecimal is $(792.41CA)_{16}$.

(c) 175.175 to binary.

Step-(1):-

Consider the decimal part and convert the number into binary. i.e., converting 175 into binary by multip dividing with (2)

2 | 175
2 | 87 - 1
2 | 43 - 1
2 | 21 - 1
2 | 10 - 1
2 | 5 - 0
2 | 2 - 1
1 - 0

$\Rightarrow (10101111)_2 = 175$

Step-(2):- Consider the floating point & multiply with 2.

$\Rightarrow$ $0.175 \times 2 = 0.35$
$0.35 \times 2 = 0.7$
$0.7 \times 2 = 1.4$
$0.4 \times 2 = 0.8$
$0.8 \times 2 = 1.6$

Now combine the floating point & decimal part numbers.

$= (10101111.00101)_2$.

∴ 175.175 into binary is

$\rightarrow (10101111.00101)_2$ //

(2) State and prove De-Morgan's theorem 1st and 2nd with logic gates and truth table

⇒ Here, we have two theorems in De-Morgan's theorem.

The first one is ⟹ $\overline{AB} = A' + B'$ &

The second one is ⟹ $\overline{A+B} = \bar{A} \cdot \bar{B}$

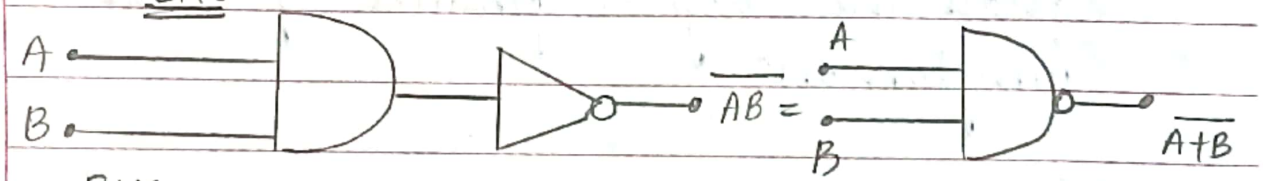
Let us consider the first one,

(i) $\overline{AB} = A' + B'$

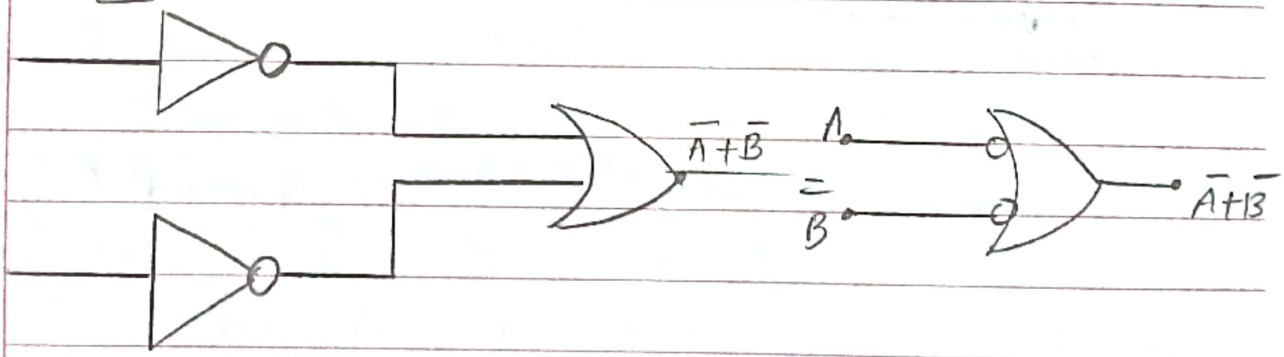
The complement of a logical product equals the logical sum of complements.

Logic Gates

LHS:-



RHS :-



Now let us look at the truth table for this theorem:-

A	B	A·B	$\overline{A \cdot B}$	A'	B'	A'+B'
0	0	0	1	1	1	1
1	0	0	1	0	1	1
0	1	0	1	1	0	1
1	1	1	0	0	0	0

$\overline{A \cdot B} \equiv A' + \bar{B}$

∴ According to our first theorem, Hence it is proved that

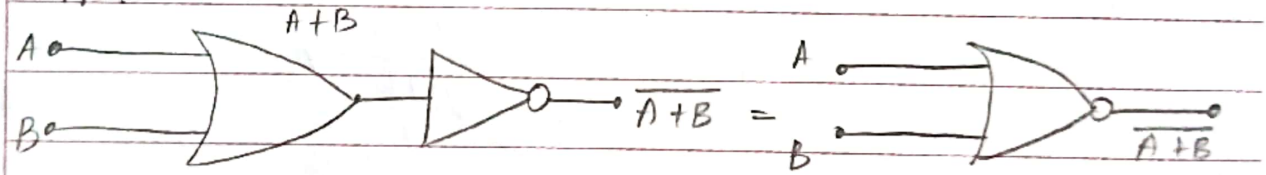
$\overline{A \cdot B} = \bar{A} + \bar{B}$

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

The complement of a logical sum equals the logical product of the complements.

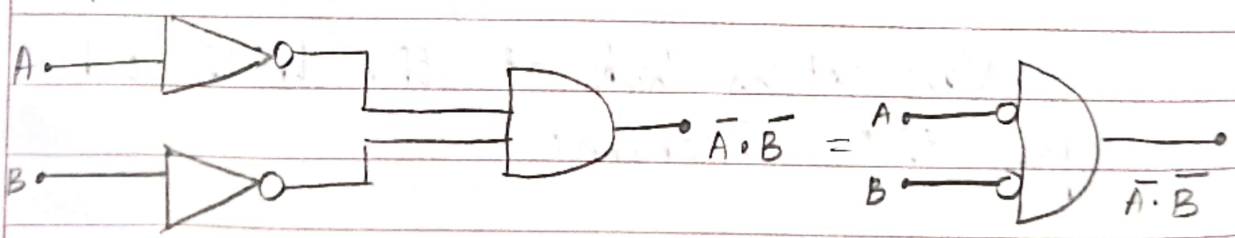
Logical Gates :-

LHS :-



NOR gate

RHS :-



Now let us look at the truth table:-

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

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

∴ According to our second theorem, Hence it is proved that,

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

---

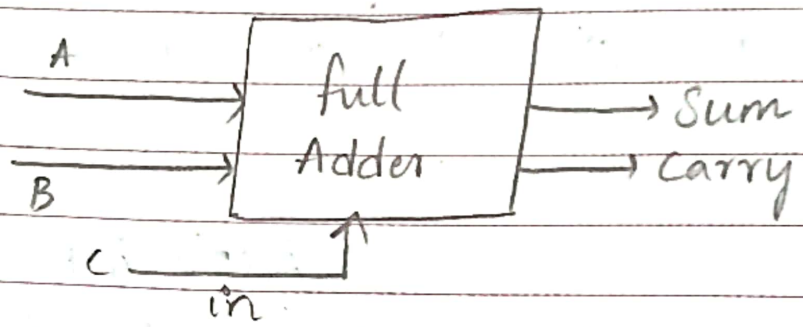
(3) What do you mean by full adder and full substractor? Design a full adder using XOR gate.

(i) Full Adder:-

- To overcome the limitations faced with half Adders, full adders are implemented.
- It is a arithmetic combinational logic circuit that performs addition of three single bits.
- It contains three inputs (A, B, Cin) and

produces two outputs (sum & Cout)

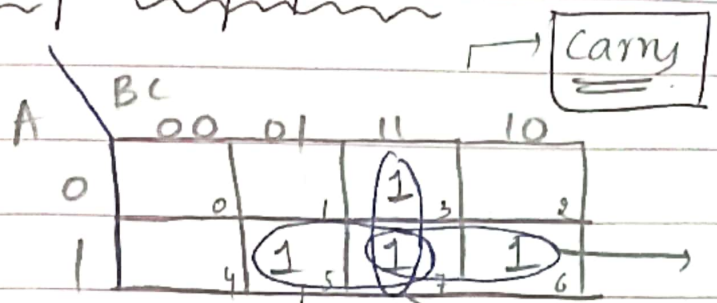
- where, C_{in} → Carry in and C_{out} → Carry out
- It is designs to add, 2 single bits with carry i.e, 3 i/p's and 2 o/p's



Truth Table :-

Inputs			Outputs	
A	B	C_{in}	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

K-map Simplification:-



G_3

A	B	C
1	1	1
1	1	0

AB

G_1

A	B	C
1	0	1
1	1	1

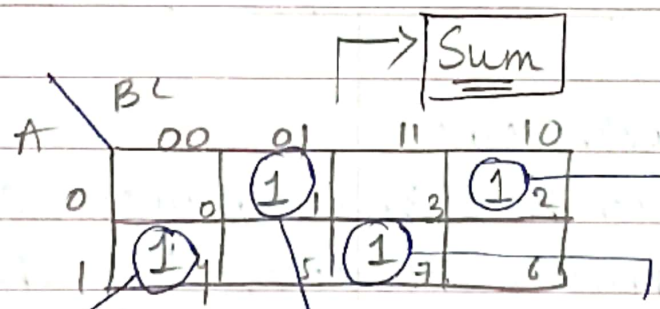
AC

G_2

A	B	C
0	1	1
1	1	1

BC

The equation obtained is ⇒ $AC + BC + AB$



$G_4 \Rightarrow$ A B C
0 1 0

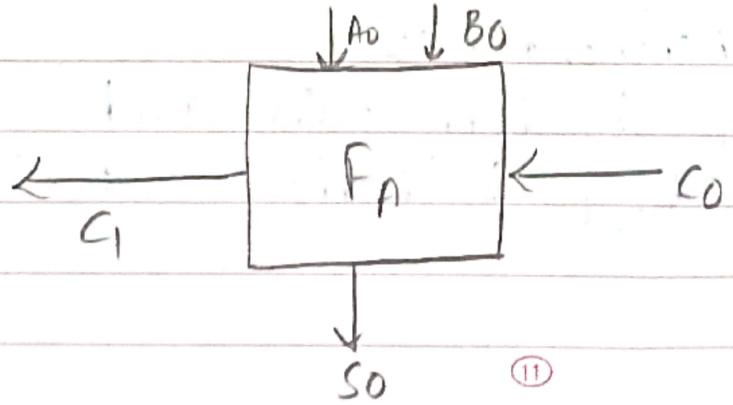
$G_1 \Rightarrow$ A B C
1 0 0

$G_2 \Rightarrow$ A B C
0 0 1

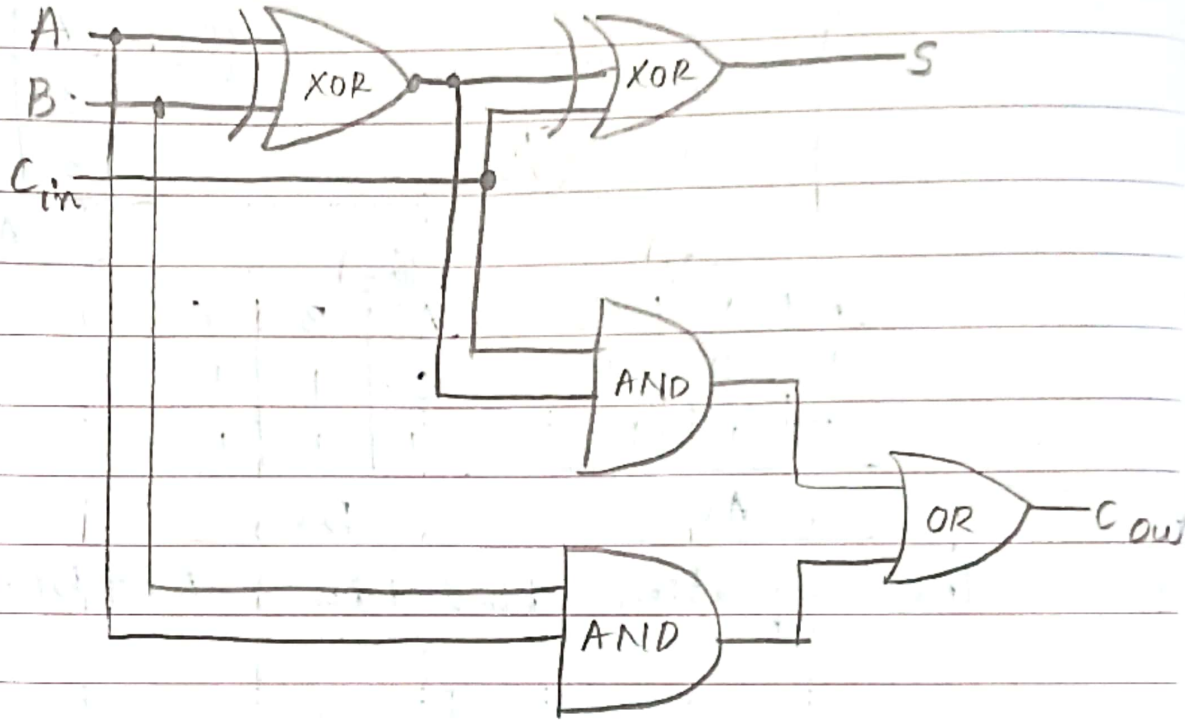
$G_3 \Rightarrow$ A B C
1 1 1

The equation obtained is ⇒

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

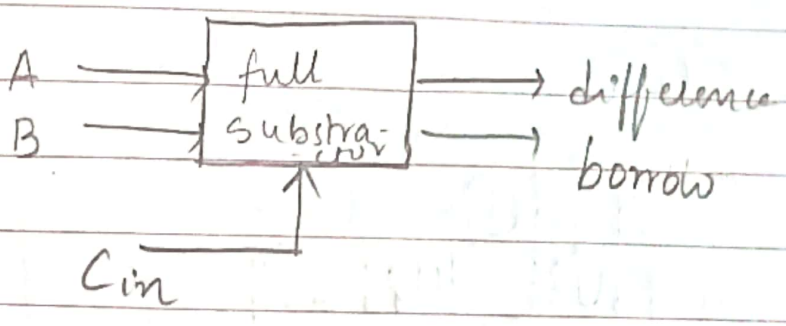


→ Full adder using XOR Gate:-



→ (ii) Full Substractor:-

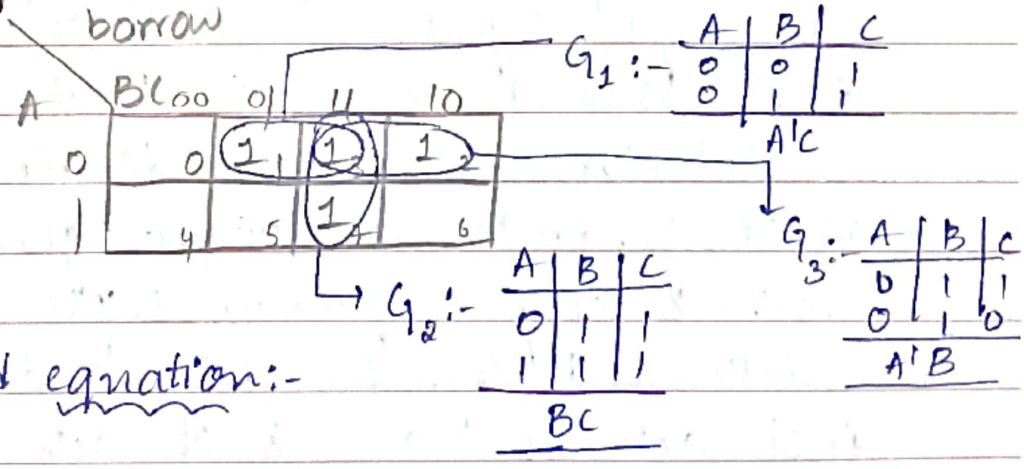
- It is a combinational logic circuit designed to perform subtraction of three single bits
- It contains three inputs and produces two outputs.
- Where, A and B are called Minuend and Subtrahend bits.
- And c_{in} is borrow-in and b_{out} is borrow-out



Truth Table:-

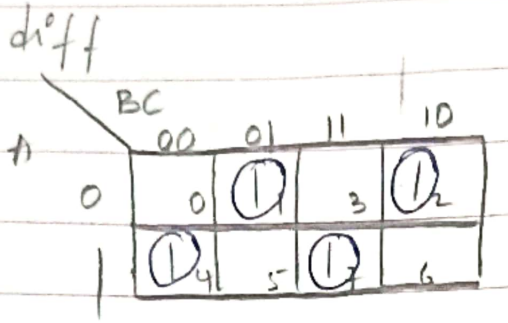
Input			Output	
A	B	C	diff	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

K-map for borrow:-



Obtained equation:-

$A'C + BC + A'B$



Obtained equation:-

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

---

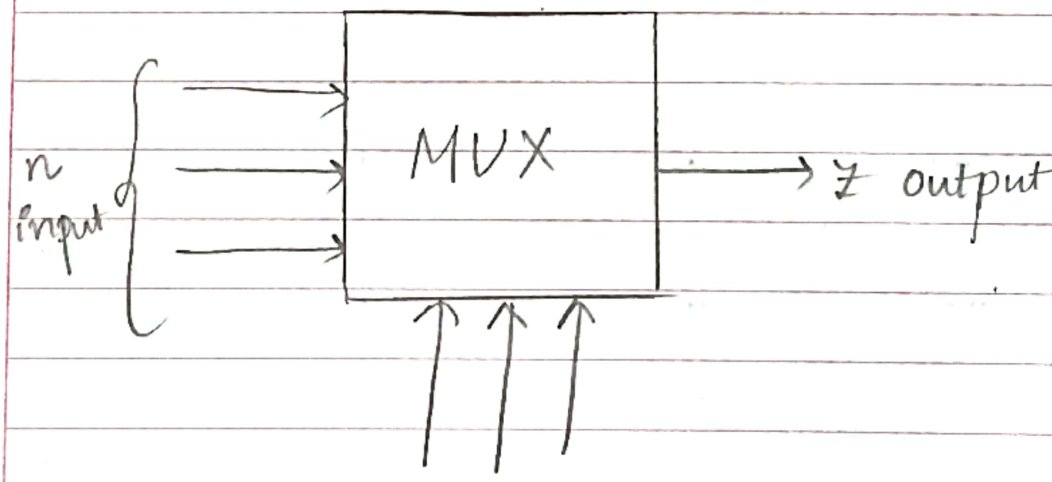
(4) Differentiate between a MUX and a DEMUX. Draw a logic circuit of 8*1 multiplexer.

(i)

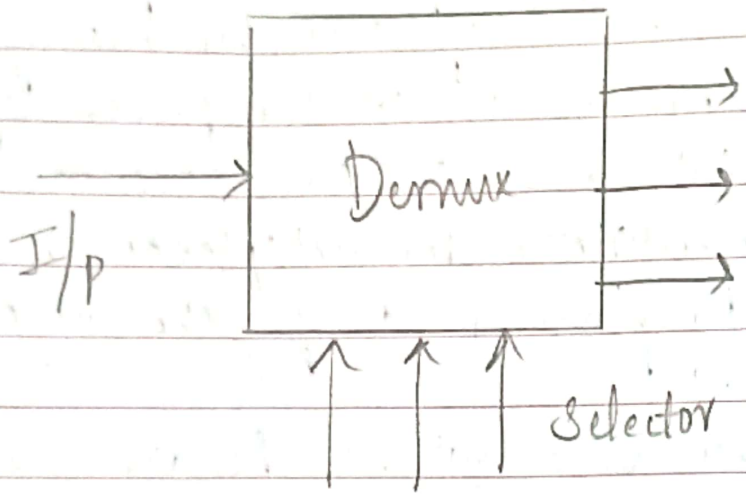
Multiplexer	Demultiplexer
i) It is a combinational circuit that accepts multiple inputs of data but provides only a single input	i) It is a combinational circuit that accepts just a single input but directs it through multiple outputs
ii) It is a data selector.	ii) It is a data distributor.
iii) It generates a single output for data and signal.	iii) It generates multiple outputs for data and signal.
iv) It acts as a digital switch.	iv) It acts as a digital circuit
v) It performs from parallel to serial	v) It performs from serial to parallel

vi, It works on an operational principal of many to one.	vi, It works on an operational principal of one to many
vii, In the process of time-division, we use a multiplexer at the end of the transmitter	vii, In the process of time-division, we use a demultiplexer at the end of the receiver.
viii, It behaves as a data selector because the multiplexer is an N to 1 device.	viii, It behaves as a data distributor because the demultiplexer is a 1 to N device.

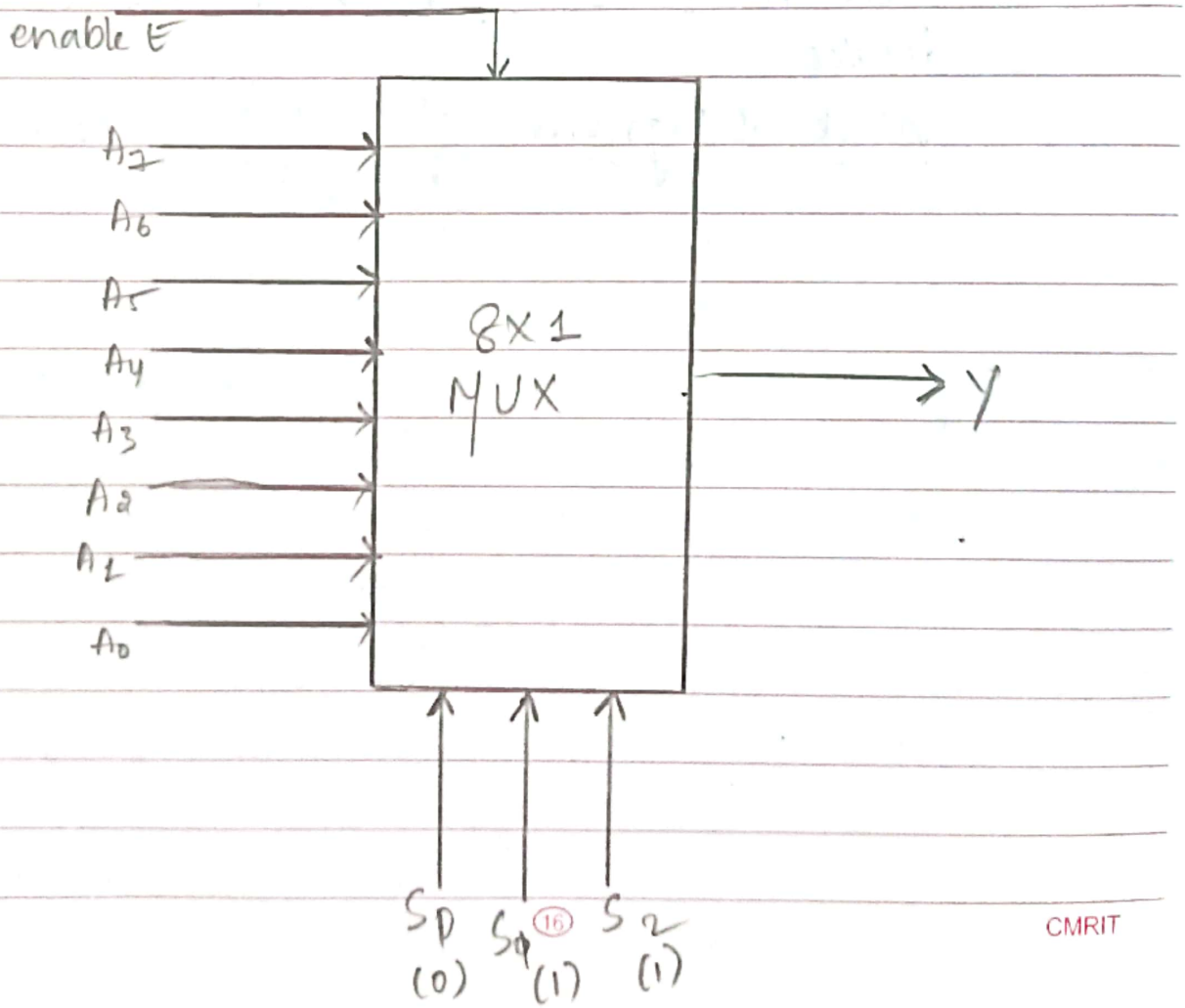
Block Diagram of Multi-plexers:-



Block Diagram of Demulti-plexer:-



(ii) Logical circuit of 8*1 multiplexer:-



Truth Table:-

Inputs			Output
S_2	S_1	S_0	Y
0	0	0	A_0
0	0	1	A_1
0	1	0	A_2
0	1	1	A_3
1	0	0	A_4
1	0	1	A_5
1	1	0	A_6
1	1	1	A_7

(5) Discuss about Instruction Codes.

→ A computer instruction is a binary code that determines the micro-operations in a sequence for a computer.

→ They are saved in the memory along with the information. Each computer has its specific group of instruction.

→ They can be categorized into two elements as Operation codes (opcodes) and Address.

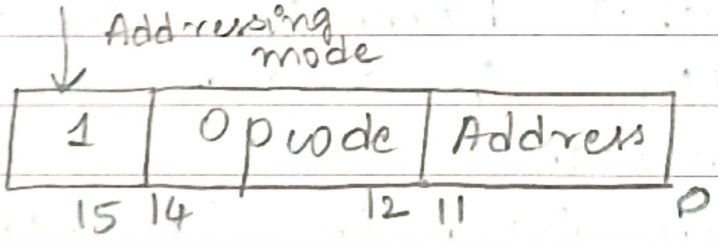
→ Opcodes specify the operation for specific instructions.

→ An address determines the registers or the areas that can be used for that operation.

→ It consists of 12 bits of memory that are required to define the address as the memory includes 4096 words.

→ The 15th bit of the instruction determines the addressing mode (where direct addressing corresponds to 0, indirect addressing corresponds to 1).

Instruction Format :-



There are three parts of the instruction format as follows :-

Addressing Modes :- This field can be represented in two different ways :-

i) Direct Addressing - It uses the address of the operand.

ii) Indirect Addressing - It facilitates the address as a pointer to the operand.

Opcodes :- It is collection of bits that represents the basic operations including add, subtract, multiply, complement and shift. It is implemented on information that is saved in processor registers or memory.

The figure shows a diagram showing direct & indirect addresses :-