I - ASSIGNMENT

(Start Writing From Here)

1 Convert the following decimal numbers to the indicated bases a. 7562.45 to octal

b. 1938.257 to hexadecimal

c. 175.175 to binary.

a) 7562.45 to octal.

The integer part of the number is divided by the base of the new number system.

8)7562(945
-7560
2

(or)

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

The fractional part of the system number is multiplied by the base of the new number system.

0.45 × 8 = 3.6
0.6 × 8 = 4.8
0.8 × 8 = 6.4
0.4 × 8 = 3.2
0.2 × 8 = 1.6
⋮

We can consider first 4 numbers if we didn't get 0 for small extent.

$(7562.45)_{10} = (16612.34631)_8$

b) 1938.257 to hexadecimal.

The integer part of the decimal number is divided by the base of the new number system

16	1938
16	121 - 2
	7 - 9

the fractional part of the number is multiplied by the base of the new number system:

$0.257 \times 16 = \underline{4}.112$

$0.112 \times 16 = \underline{1}.792$

$0.792 \times 16 = \underline{12}.672$ - C

$0.672 \times 16 = \underline{10}.752$ - A

$0.752 \times 16 = \underline{12}.032$ - C

In hexadecimal we denote 10 with A and 12 with C

$\therefore (1938.257)_{10} = 792.41CAC$

c) 175.175 to binary

The integer part of the number is divided by the base of the new number system:

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

The fractional part of the number is multiplied by the base of the new number system

$0.175 \times 2 = \underline{0}.35$
$0.35 \times 2 = \underline{0}.7$
$0.7 \times 2 = \underline{1}.4$
$0.4 \times 2 = \underline{0}.8$
$0.8 \times 2 = \underline{1}.6$ ↓
⋮

$\therefore (175.175)_{10} = (10101111.00101)_2$

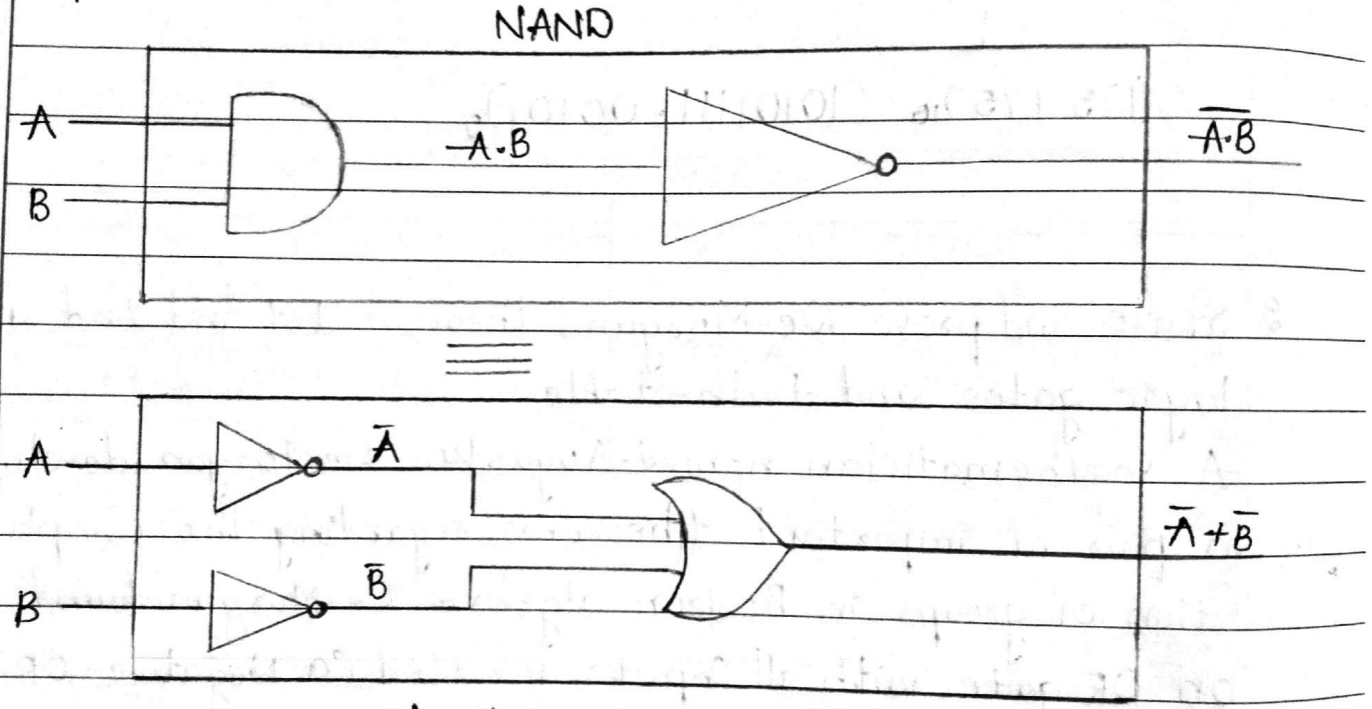
2 State and prove De-Morgan's theorem 1st and 2nd with logic gates and truth-table.

A mathematician named Augustus DeMorgan developed a pair of important theorems regarding the complementation of groups in Boolean algebra De-Morgan found that on OR gate with all inputs inverted (a Negative-OR gate) behaves the same as a NAND gate with non-inverted inputs; and on AND gate with all inputs inverted (a Negative-AND gate) behaves the same as a NOR gate with non-inverted inputs. DeMorgan's theorem states that inverting the output of any gate is the same as using the opposite type of gate with inverted inputs to put this in circuit terms. DeMorgan's theorem states that the AND gate with inverted output and the OR gate with inverted inputs are functionally equivalent

Demorgan suggested two theorems that form an important part of Boolean algebra. In the equation form, they are:

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

(i) $\overline{AB} = \bar{A} + \bar{B}$: The complement of a product is equal to the sum of the complements. This formula shows how a two-input NAND gate is "broken" to form an OR gate with two inverted inputs.



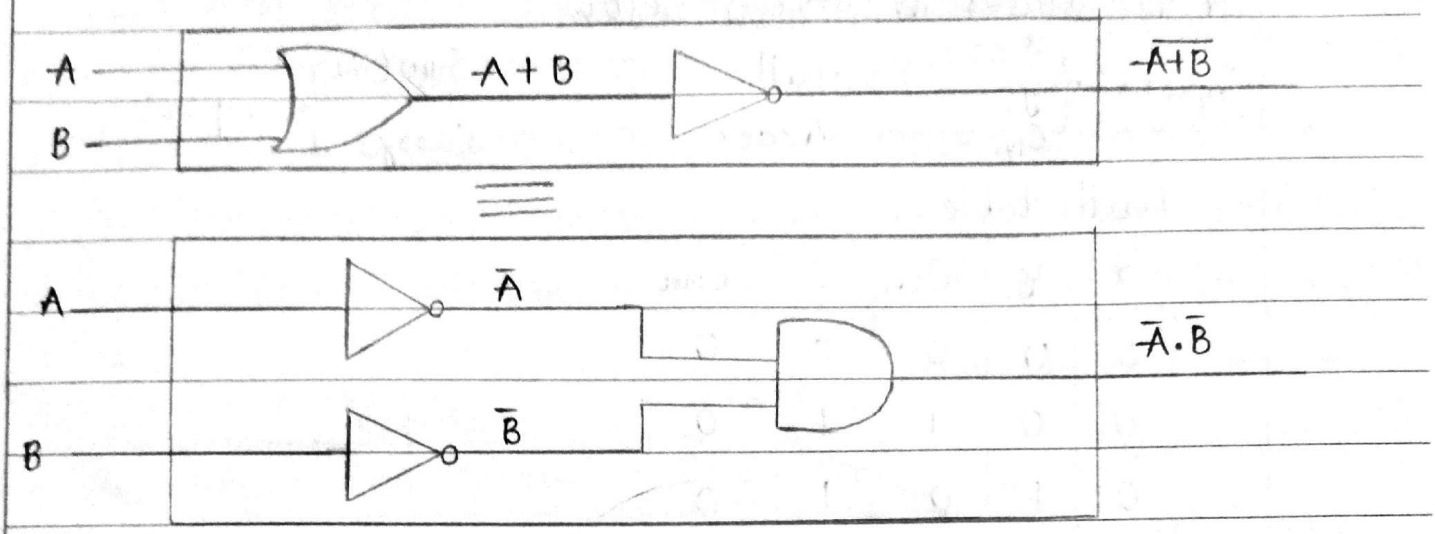
Negative-OR

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

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

ii) $\overline{A+B} = \bar{A} \cdot \bar{B}$: The complement of a sum is equal to the product

of the complement inverted inputs.



Negative-AND

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

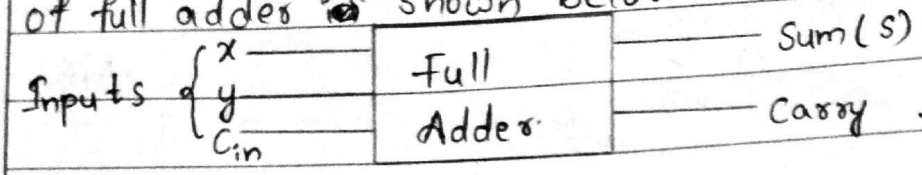
Hence $\overline{A+B} = \bar{A}.\bar{B}$

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

Full-Adder: It can be defined as a logic circuit that adds three bits i.e, two bits to be added and a carry bit from previous addition, which results in a sum and a carry is known as "Full Adder".

The full adder circuit has 3 inputs like x, y and cin which add the 3 input numbers and generate a carry and sum Full Adder is defined as a combinational circuit that performs the

addition of 2 binary inputs and a carry input. The block schematic of full adder is shown below.



Truth table.

x	y	Cin	S	Cout
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 for sum(S)

x \ yCin	00	01	11	10
0		1		1
1	1		1	

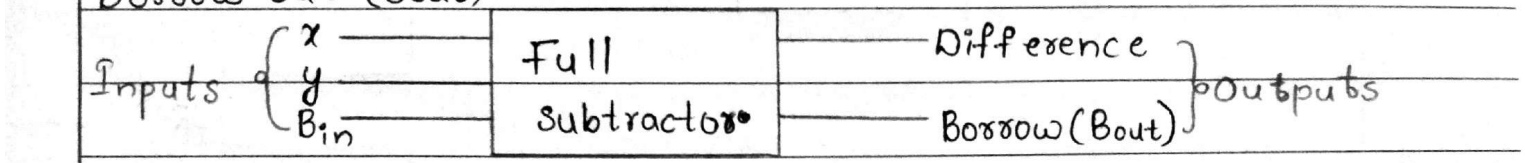
K-map for cout

x \ yCin	00	01	11	10
0			1	
1		1	1	1

$$S = \bar{x}y\bar{C}_{in} + \bar{x}y\bar{C}_{in} + x\bar{y}\bar{C}_{in} + xyC_{in}$$
$$= (\bar{x}\bar{y} + xy)C_{in} + (\bar{x}y + x\bar{y})C_{in}$$
$$= (x \odot y) + (x \oplus y)\bar{C}_{in}$$
$$= x \oplus y \oplus \bar{C}_{in}$$
$$C_{out} = xy + yC_{in} + xC_{in}$$

Full subtractor: One major disadvantage of the half subtractor circuit when used as a binary subtractor is that there is no problem/

provision for a "Borrow in" from the previous circuit when substracting multiple data bits from each other. Full subtractor was used to overcome the limitation of half subtractor circuit. Full subtractor performs the subtraction of two binary inputs and a previous generated borrow input also so therefore full substractor has 3 inputs x, y and borrow in (Bin) two outputs difference (D), borrow out (Bout).



Truth table:

Input			Output	
X	Y	Bin	D	Bout
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:

K-map for Difference (D)

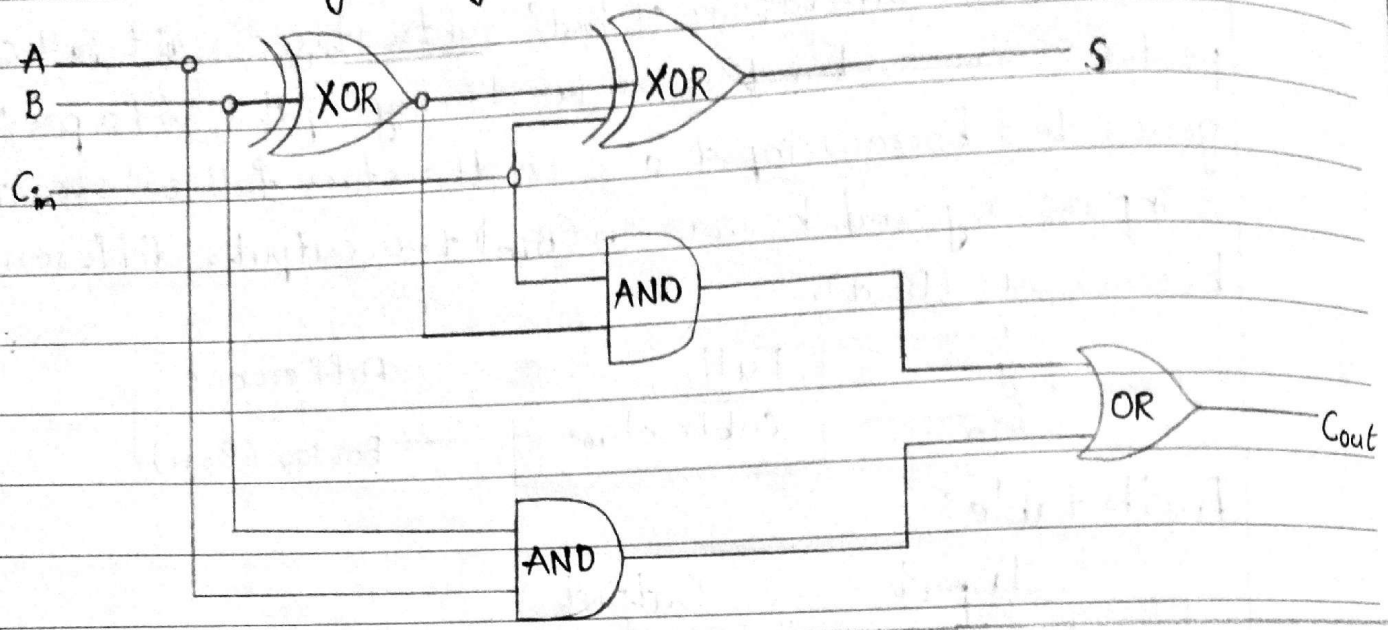
X \ YBin	00	01	11	10
0		(1)		(1)
1	(1)		(1)	

K-map for Borrow (Bout)

X \ YBin	00	01	11	10
0		1	1	1
1			1	

$D = (\bar{x} \oplus \bar{y}) B_{in} + (x \oplus y)\bar{B}_{in}$ $\quad B_{out} = \bar{X}Y + YB_{in} + \bar{X}B_{in}$

Full Adder using XOR gate:

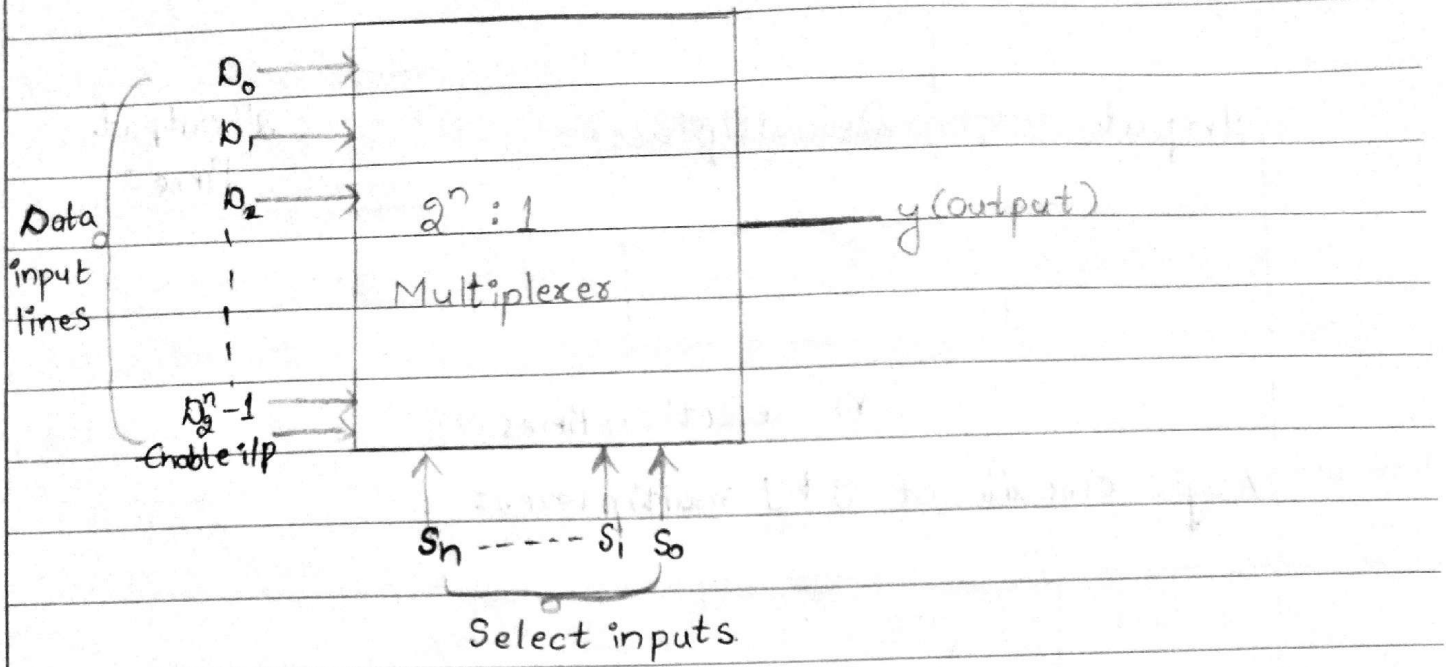


4 Differentiate between a MUX and a DEMUX. Draw a logic circuit of 8*1 multiplexer.

MUX: A multiplexer is electronic switch that can connect one out of 'n' inputs to the output. Multiplexers are mostly used as data selectors. Data selector contains 'n' data inputs lines m select digit inputs such that

$$2^m = n \text{ or } m = \log 2^n.$$

In simple terms, multiplexer selects one of several input signals and posses it on to the output. The enable input (also called stribe) can be used to cascade 2 or more multiplexer ICs to construct a multiplexer with larges numbes of inputs.



→ It is a digital switch.

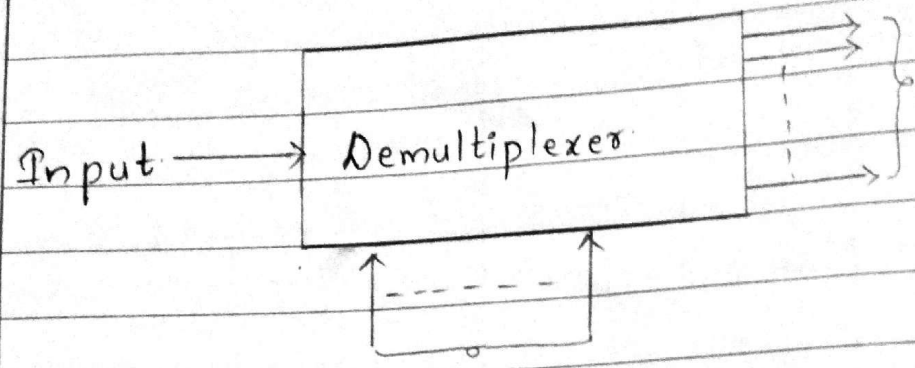
→ The basic multiplexer has several data-input lines and a single output line.

→ The selection of particular input line is controlled by a set of selection lines.

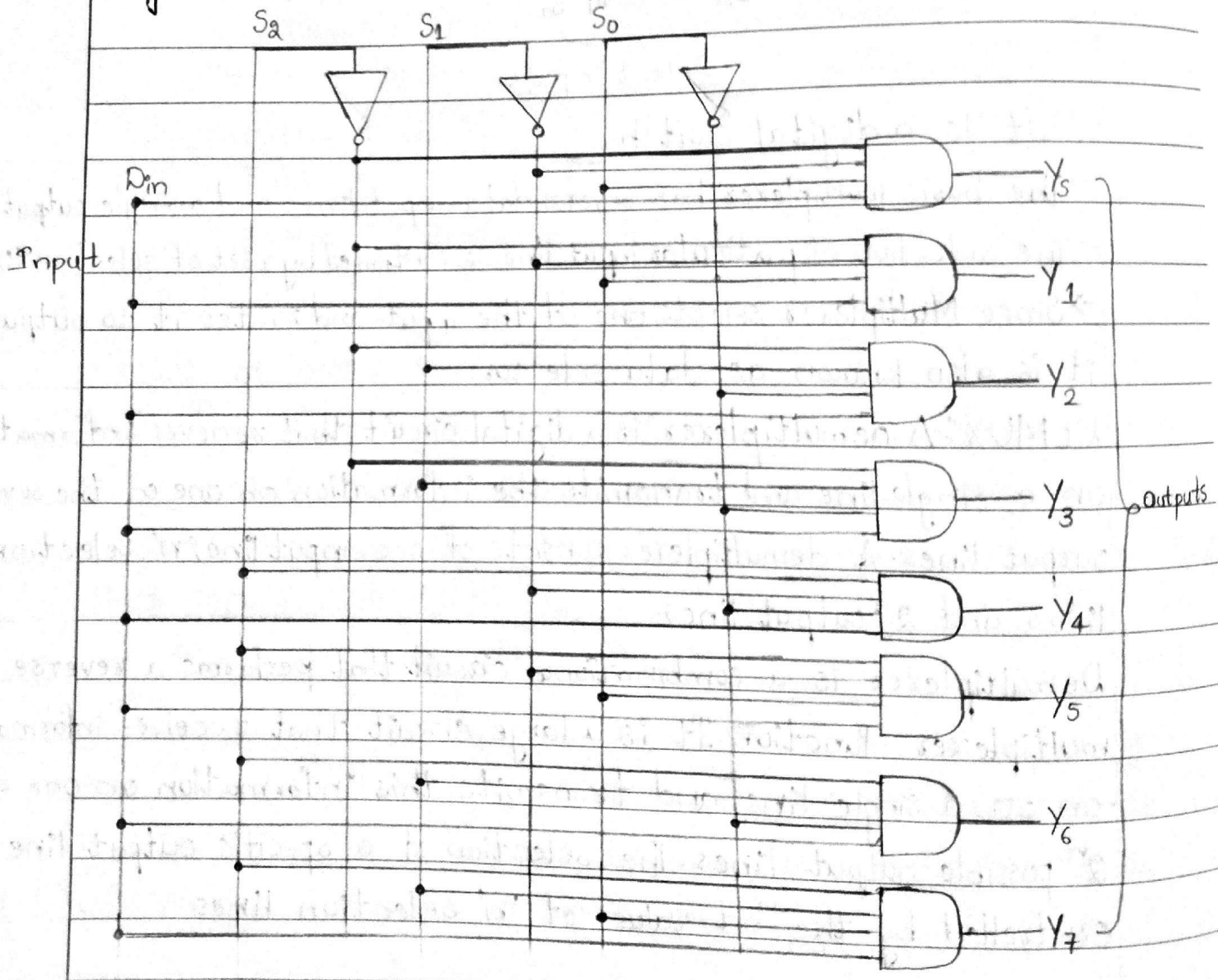
→ Since Multiplexer selects one of the input and routes it to output. it is also known as data selector.

DEMUX: A demultiplexer is a digital circuit that recieves information on a single line and transmits the information on one of the several output lines. A demultiplexer consists of one input line, 'n' selection lines and 2^n output lines.

Demultiplexer is a combinational circuit that performs a reverse multiplexer function. It is a large circuit that receives information on a single line and transmits this information on one of 2^n possible output lines. The selection of a specific output line is controlled by the bit values of 'n' selection lines.



Logic circuit of 8*1 multiplexer:



5 Discuss about Instruction codes

→ It is a group of bits that instruct computer to perform a specific operation.

→ The most basic part of instruction code is its operational part.

→ Operation code of an instruction is a group of bits that define operations as add, sub etc..

→ If operation code length is equal to 6 bits then we can perform 2^6 different operations.

→ An operation is a part of instruction stored in a memory.

→ Along with operation instruction should contain information about operands on which this operation has to be applied.

→ An instruction code must specify not only the operation but also registers or memory words where operands can be found and register or memory word where result has to be stored.

→ Memory words can be specified instruction codes by their address.

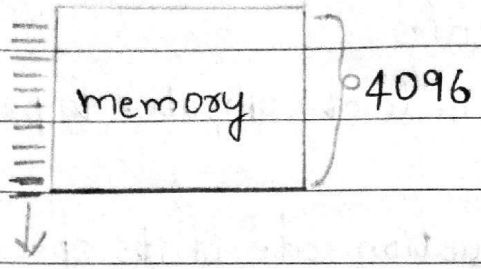
→ Each computer has its own particular instruction code format.

→ Generally instruction code has 2 parts operation and address.

→ Generally instructions are stored in 1 section of memory and data in other part.

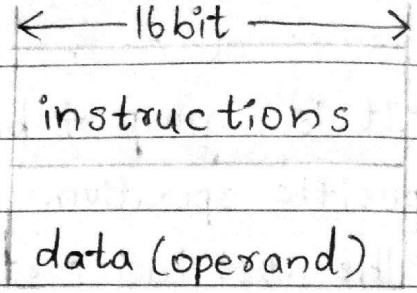
Ex: If memory unit = 4096 words then

a)

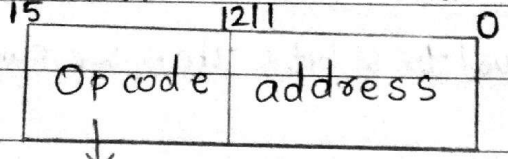


12 bit address

b)

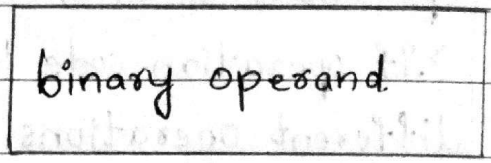


c) instruction format



Operation code.

d)



→ Instruction has direct address of operand is known as "direct address".

→ If instruction has information about operand then it is known as "Indirect address".