Computer Organization

Unit - 1

Part - A (Digital Logic circuits)

1. Digital computers
2. Logic Gates ✓
3. Boolean Algebra
4. Map simplification ✓
5. Combinational circuits ✓
6. Flip - Flops ✓
7. Sequential circuits

Part - B (Digital Components)

1. Integrated circuits (IC)
2. Decoders ✓ (3x8)
3. Multiplexers
4. Registers ✓ (4-bit serial)
5. Shift - Registers
6. Binary counters ✗ (4-bit binary)
7. Memory unit ✓

Unit - 1

Part-A (Digital logic circuits)

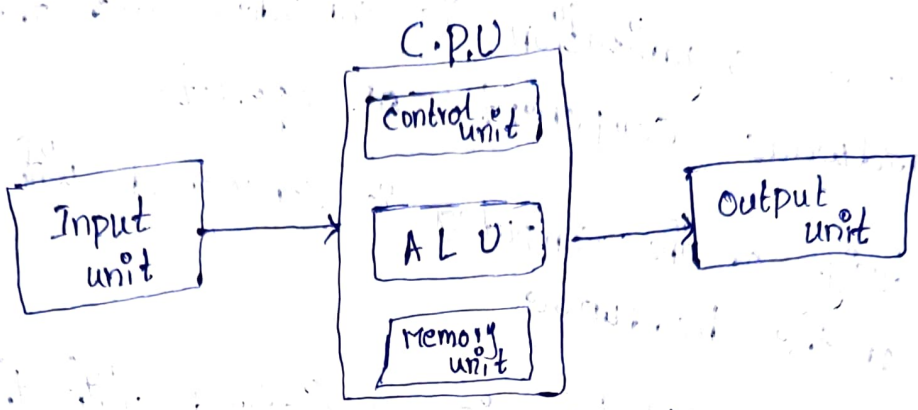
1. Digital computers

* Computer organization is consider with the way hardware components operate
* The design of the computer is concerned with the design of the hardware.
* Computer Architecture is associated with the structure and Behaviour of the computer.
* A digital computer uses binary number system that is, it uses 0's and 1's.
* The physical components of a computer system is consider to be computer hardware.

Examples :- Keyboard, Mouse, VDU (visual display unit), CPU, RAM, HDD/SSD (solid state drive) Motherboard, SMPS (switch mode power solid).

Block diagram of a computer :

* The block diagram of a digital computer is divided in to five parts.



Input unit :-

Input unit is used to take the commands & instructions from the users and sends these inputs to the CPU. Ex:- keyboard, Mouse.

Output unit :-

The output unit is used to display the results or outputs on the screen.

Example : VDU, printers.

Control unit :-

The control unit is responsible to handle the instructions of the remaining units.

Memory unit :-

Memory unit is responsible to store the information or data. Memory's are divided

into two types i.e these are Primary memory and Secondary memory.

ALU :-

ALU stands Arithmetic and Logic unit. ALU is used to perform arithmetic and logical calculations.

2. Logic Gates

* The maniplation of binary information is done by using logic circuits in the form of gates.

* The different types of logic gates which are used in a digital computer are

1. AND Gate
2. OR Gate
3. NOT Gate
4. NAND Gate
5. NOR Gate
6. XOR Gate (Exclusive OR)
7. XNOR Gate (Exclusive NOR)

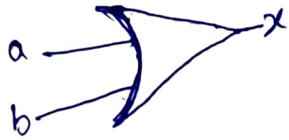
Gate name	Symbol	function	Truth table

AND

$x = a \cdot b$

a	b	x
0	0	0
0	1	0
1	0	0
1	1	1

OR



$x = a + b$

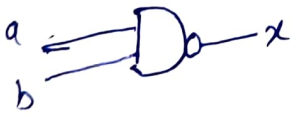
a	b	x
0	0	0
0	1	1
1	0	1
1	1	1

NOT

$x = a'$

a	x
0	1
1	0

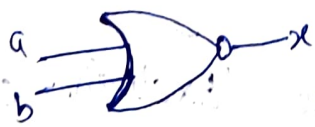
NAND



$x = (a \cdot b)'$

a	b	x
0	0	1
0	1	1
1	0	1
1	1	0

NOR

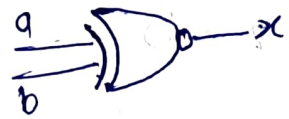


$x = (a + b)'$

a	b	x
0	0	1
0	1	0
1	0	0
1	1	0

Gate name	Symbol	function	Truth table
XOR		$x = (a+b)'$ $x = ab' + a'b$	

a	b	x
0	0	0
0	1	1
1	0	1
1	1	0

XNOR 

$x = (ab' + a'b)'$

a	b	x
0	0	1
0	1	0
1	0	0
1	1	1

3. Boolean Algebra

* Boolean Algebra is used to perform logical operations by using binary numbers & values (0's & 1's).
* Here 0 indicates False and 1 indicates True.
* The operations can be performed in boolean algebra are, AND operation.
  OR operation
  NOT operation
* To represent the relationship b/w the function and binary value, a truth table is used.

* A boolean function in boolean algebra can be represented as a logic diagram

Example:- $f = x + y'z$

number of variables (x, y, z) $n = 3$

combinations $(C) = 2^n$

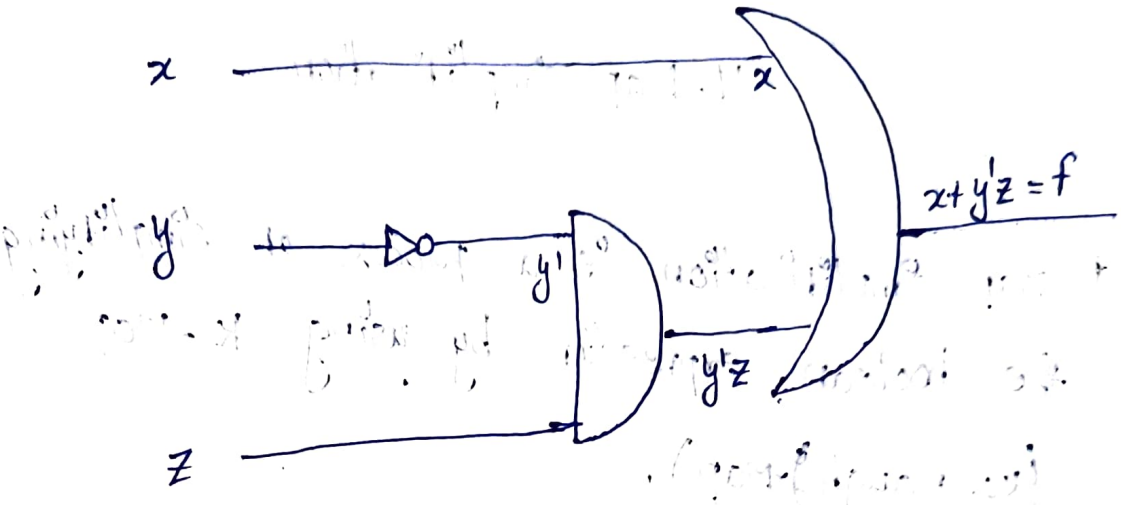
$= 2^3 = 8$

Truth table

x	y	z	y'	$y'z$	$f = x + y'z$
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	0	0	1

Logic : Diagram

$$f = x + y'z$$



* The boolean function (f) in boolean algebra can be represented in two ways. These are:

1. Truth table
2. Logic Diagram

* Basic identites of boolean algebra :

1) $x+0 = x$
2) $x.0 = 0$
3) $x+1 = 1$
4) $x.1 = x$
5) $x+x = x$
6) $x.x = x$
7) $x+x' = 1$
8) $x.x' = 0$
9) $x+y = y+x$
10) $x.y = y.x$
11) $x+(y+z) = (x+y)+z$
12) $x(y.z) = (x.y)z$
13) $x(y+z) = (x.y)+(x+z)$
14) $x+(yz) = (x+y)(x+z)$
15) $(x+y)' = x'.y'$
16) $(xy)' = x'+y'$

17) $(x')' = x$

4. Map simplification

* Map simplification is a process of simplifying the boolean expression by using K-Map (karnaugh-Map).
* A K-Map is used as a graphical representation for truth table.
* A k-map uses minimum no. of terms (Min-terms) to express the algebric functions.
* A min-term is a combination of variables in a truth table.

Example :-

If a function has three variables (x, y, z) then we have 2^3 min-terms. i.e; 8 min-terms for three ~~variables~~ variables.

* Map simplification can be of different types, they are:

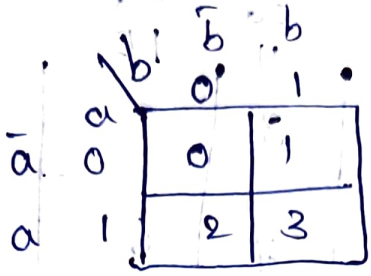
2^n

1. 2-Variable map, 2 var's, $2^2 = 4$ min terms
2. 3-Variable map, 3 var's, $2^3 = 8$ min terms
3. 4-Variable map, 4 var's, $2^4 = 16$ min terms

1. 2-var map :- var = 2, minterms = 4

$a = 1$, $\bar{a} = 0$

$b = 1$, $\bar{b} = 0$

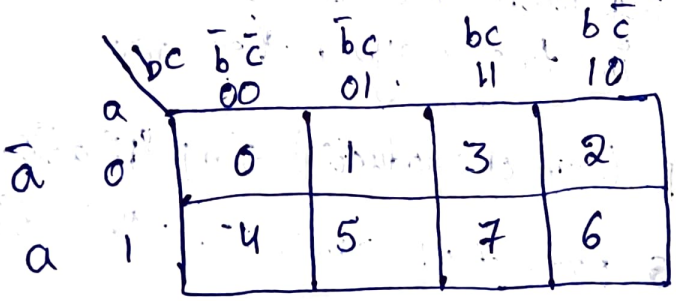


2. 3-var map :- var = 3, min-terms = 8

$a = 1$, $\bar{a} = 0$

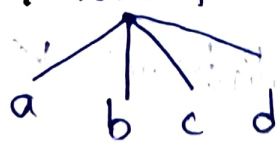
$b = 1$, $\bar{b} = 0$

$c = 1$, $\bar{c} = 0$



Binary combination
0 0 - 0
0 1 - 1
1 1 - 3
1 0 - 2

3. 4-var map :- var = 4, minterms = 16



$a=1, \bar{a}=0$
$b=1, \bar{b}=0$
$c=1, \bar{c}=0$
$d=1, \bar{d}=0$

ab \ cd	$\bar{c}\bar{d}$ 00	$\bar{c}d$ 01	cd 11	$c\bar{d}$ 10
$\bar{a}\bar{b}$ 00	0	1	3	2
$\bar{a}b$ 01	4	5	7	6
ab 11	12	13	15	14
$a\bar{b}$ 10	8	9	11	10

* The map simplification for a boolean expression can be done in two ways, they are

1. SOP (Sum of products) simplification
2. POS (Product of sums) simplification

- For sum of products, consider 1's in k-map
- For POS, consider 0's in k-map
- We can group the minterms vertically or horizontaly.
- Overlapping of minterms can also be done.
- A group must contain maximum no. of minterms during map simplification.

1. SOP (Sum of Products) Simplification (1's):-

Eg 1:- simplify the boolean function for, $f(a,b,c)$ using SOP.

$$f(a,b,c) = \Sigma(3,4,6,7)$$

Since the function has 3 variables (a,b,c),

minterms $2^n = 2^3 = 8$

a \ bc	$\bar{b}\bar{c}$	$\bar{b}c$	bc	$b\bar{c}$
$\bar{a}$			1	
a	1		1	1

$$f = \left((\bar{a}bc) + (a\,b\,c)\right) + \left((ab\bar{c}) + (a\,b\,\bar{c})\right)$$

$$f = bc(\bar{a}+a) + a\bar{c}(\bar{b}+b) \qquad (\bar{x}+x=1)$$

$$f = bc(1) + a\bar{c}(1)$$

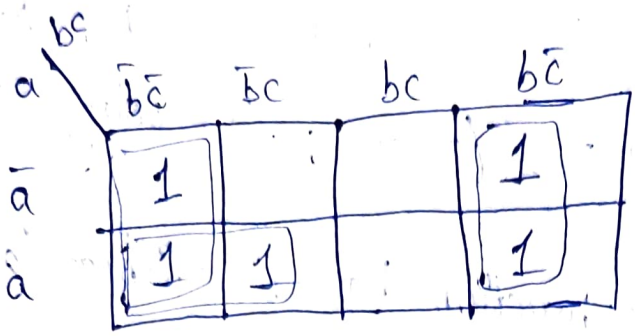
$$f = bc + a\bar{c}$$

Eg 2:- simplify the boolean function for

$$f(a,b,c) = \Sigma(0,2,4,5,6)$$ using SOP

since the function has 3 variables (a,b,c)

minterms $2^n = 2^3 = 8$



$$((\bar{a}\bar{b}\bar{c})+(a\bar{b}\bar{c}))+((a\bar{b}\bar{c})+(abc))+((\bar{a}b\bar{c})+(ab\bar{c}))$$

$$\bar{b}\bar{c}(\bar{a}+a)+a\bar{b}(\bar{c}+c)+b\bar{c}(\bar{a}+a)$$

$$\bar{b}\bar{c}(1)+a\bar{b}(1)+b\bar{c}(1)$$

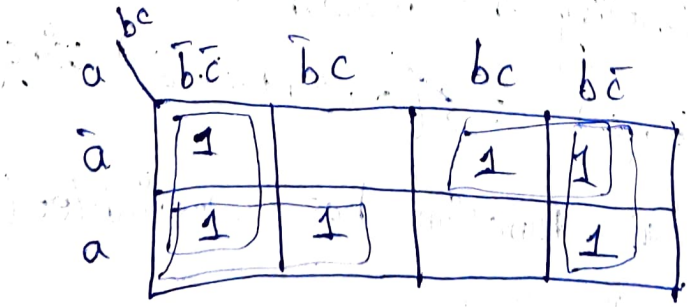
$$f = \bar{b}\bar{c}+a\bar{b}+b\bar{c} = \bar{c}(\bar{b}+b)+a\bar{b}$$
$$= \bar{c}+a\bar{b}$$

Eg 3:- simplify the boolean function for

$f(a,b,c) = \Sigma(0,2,3,4,5,6)$ using SOP

since, the function has 3 variables (a,b,c)

minterms $2^n = 2^3 = 8$.



$$\left((\bar{a}\bar{b}\bar{c}) + (a\bar{b}\bar{c})\right) + \left((a\bar{b}\bar{c}) + (ab\bar{c})\right) + \left((\bar{a}bc) + (\bar{a}b\bar{c})\right) + \left((ab\bar{c}) + (\bar{a}b\bar{c})\right)$$

$$\bar{b}\bar{c}(\bar{a}+a) + a\bar{b}(\bar{c}+c) + \{\bar{a}b(\bar{c}+c) + b\bar{c}(a+\bar{a})$$

$$f = \bar{b}\bar{c} + a\bar{b} + \bar{a}b + b\bar{c}$$

$$= \bar{c}(\bar{b}+b) + a\bar{b} + \bar{a}b$$

$$= \bar{c} + a\bar{b} + \bar{a}b$$

Eg:4 simplify the boolean function for

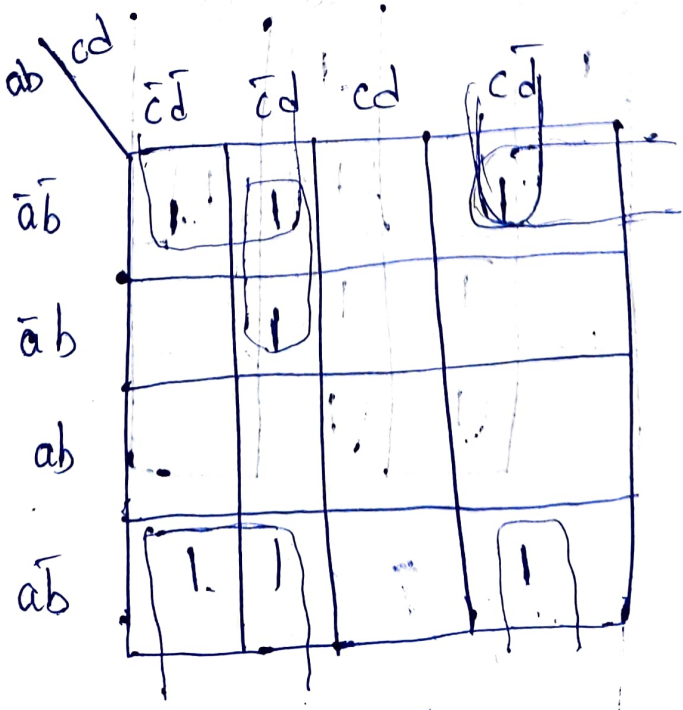
$f(a,b,c,d) = \Sigma(0,1,2,5,8,9,10)$

using SOP.

since, the function has 4 variables (a,b,c,d)

minterms $2^n = 2^4 = 16$.

SOP = 1's



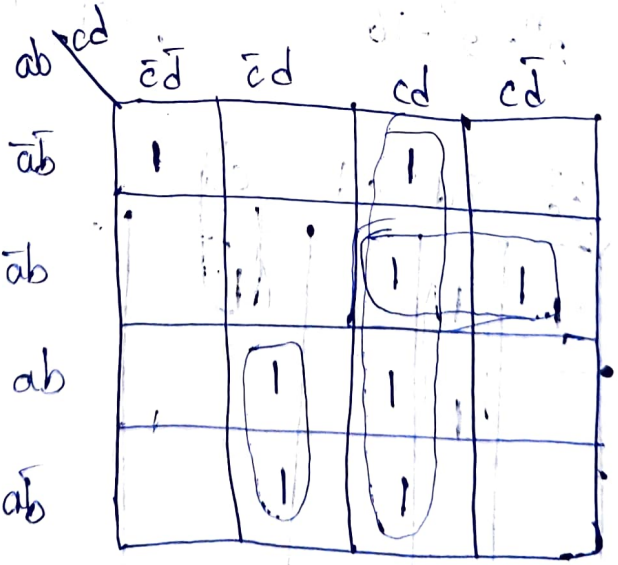
$$= (\bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}\bar{c}d + ab\bar{c}\bar{d} + a\bar{b}\bar{c}d) + (\bar{a}\bar{b}cd + \bar{a}b\bar{c}d) + (\bar{a}\bar{b}c\bar{d} + abc\bar{d})$$

$$= \left(\bar{a}\bar{b}\bar{c}(\bar{d}+d) + ab\bar{c}(\bar{d}+d)\right) + \bar{a}\bar{c}d(\bar{b}+b) + \bar{b}c\bar{d}(\bar{a}+a)$$

$$= \bar{b}\bar{c}(\bar{a}+a) + \bar{a}\bar{c}d + \bar{b}c\bar{d}$$

$$= \bar{b}\bar{c} + \bar{a}\bar{c}d + \bar{b}c\bar{d}$$

Eg 5 :- simplify the boolean function for $f(a,b,c,d) = \Sigma(0,3,6,7,9,11,13,15)$ using



$f = \bar{a}\bar{b}\bar{c}\bar{d} + (\bar{a}\bar{b}cd + \bar{a}bcd + abcd + a\bar{b}cd) + (\bar{a}bcd + \bar{a}bc\bar{d}) + (ab\bar{c}d + a\bar{b}\bar{c}d)$

$= \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}bc(\bar{d}+d) + a\bar{c}d(b+\bar{b}) + \bar{a}cd(\bar{b}+b) + acd(b+\bar{b})$

$= \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}bc + a\bar{c}d + \bar{a}cd + acd$

$= \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}bc + a\bar{c}d + cd(\bar{a}+a)$

$f = \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}bc + a\bar{c}d + cd$

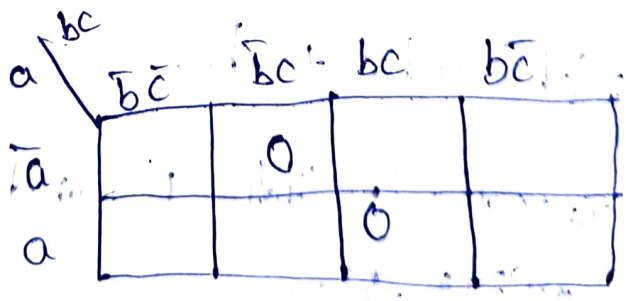
Product of sums (POS):

Eg: simplify the boolean function for

$F(a,b,c) = \Sigma(0,2,3,4,5,6)$ using POS.

since the function has 3 variables (a,b,c)

minterms : $2^3 = 8$.



simplify using sop

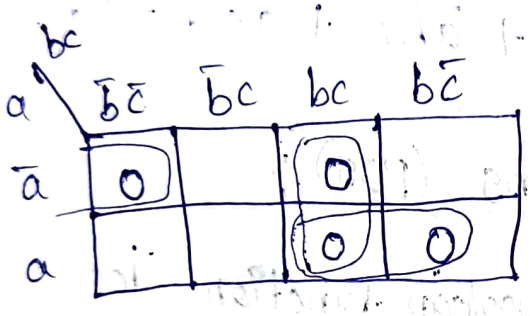
$(\bar{a}bc) + (abc)$

for POS, convert • into + and + into •, along with inverse.

$f = (a+b+\bar{c}) \cdot (\bar{a}+\bar{b}+\bar{c})$.

Ex-2:-

$f = (a,b,c) = \Sigma(1,2,4,5)$ using POS:



simplify using SOP

$= \bar{a}\bar{b}\bar{c} + (\bar{a}b\bar{c} + abc) + (abc + ab\bar{c})$

$= \bar{a}\bar{b}\bar{c} + bc(\bar{a}+a) + ab(c+\bar{c})$

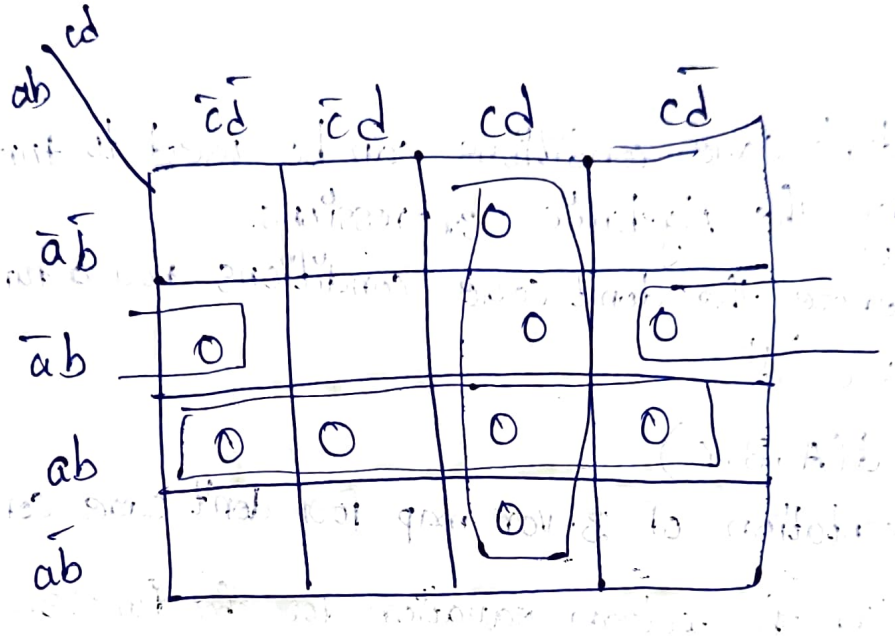
$= \bar{a}\bar{b}\bar{c} + bc + ab$.

for pos, convert '.' into '+' and + into '.' along with inverse.

$f = (a+b+c) \cdot (\bar{b}+\bar{c}) \cdot (\bar{a}+\bar{b})$

Ex:3 simplify the boolean function for

$f(a,b,c,d) = \Sigma(0,1,2,5,8,9,10)$ by using pos.



simplify using SOP

$= (\bar{a}b\bar{c}\bar{d} + \bar{a}bc\bar{d}) + (ab\bar{c}\bar{d} + ab\bar{c}d + abcd + abc\bar{d}) + (\bar{a}\bar{b}cd + \bar{a}bcd + abcd + a\bar{b}cd)$

$= \bar{a}b\bar{d}(\bar{c}+c) + ab\bar{c}(\bar{d}+d) + abc(d+\bar{d}) + \bar{a}cd(\bar{b}+b) + acd(b+\bar{b})$

$= \bar{a}b\bar{d} + ab\bar{c} + abc + \bar{a}cd + acd$

$= \bar{a}b\bar{d} + ab(\bar{c}+c) + cd(\bar{a}+a)$

$= \bar{a}b\bar{d} + ab + cd$

For pos

$= (\bar{a}+\bar{b}) + (\bar{c}+\bar{d}) \cdot (a+\bar{b}+d)$

Don't care conditions

* The min terms that may produced either 0 or 1, for the function are said to be don't care conditions where, these are represented by using "X"

* The don't care conditions can be used to further simplify the Algebraic expressions.
* To denote the don't care conditions for a function we use "d".

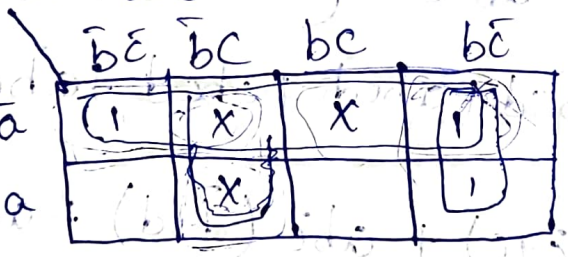
i.e., $d(A,B,C)$

representation of 3-var map for don't care condition

simplify the boolean equation for the function

$F(A,B,C) = \Sigma(0,2,6)$ and $d(A,B,C) = \Sigma(1,3,5)$

$F(A,B,C) = \Sigma(0,2,6) + d(A,B,C) = \Sigma(1,3,5)$



SOP $F = (\bar{A}\bar{C}) + (B\bar{C})$ [without using d]

$F = \bar{A} + B\bar{C}$ [By considering]

From the above example the expression is further simplified which is an sop by considering the minterms (1,3) of the Don't care conditions.

$= (\bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + ab\bar{c} + \bar{a}b\bar{c}) + (\bar{a}b\bar{c} + ab\bar{c})$

$= \bar{a}(\bar{b}+b)$

$= \bar{a}b(\bar{c}+c) + a\bar{b}(c+\bar{c}) + b\bar{c}(\bar{a}+a)$

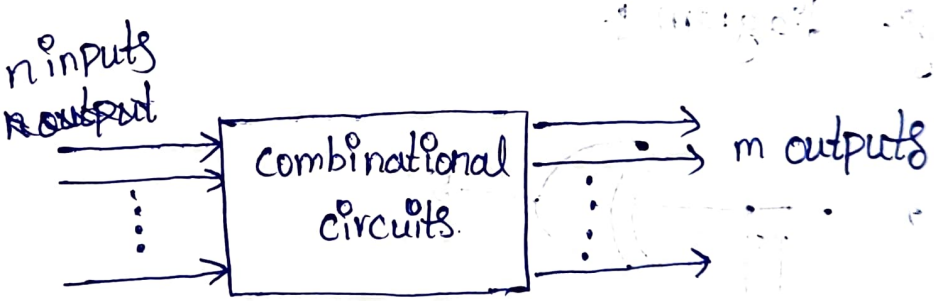
$= \bar{a}b + a\bar{b} + b\bar{c}$

$= \bar{a}(b+b) + b\bar{c}$

$= \bar{a} + b\bar{c}$

5. Combinational circuit

* A combinational circuit is a collection of logic gates which are connected to each other & produces m outputs by taking n inputs
* A combinational circuit is mostly used in digital computers.
* A combinational circuit can also be specified with a boolean function



* The combinational circuit is used to perform "arithmetic operations".
* To perform arthimeatic operations in a combina-tional circuits, we use
    1. Half Adder
    2. Full adder
* The half adder is used to perform arthimetic operation on two inputs.

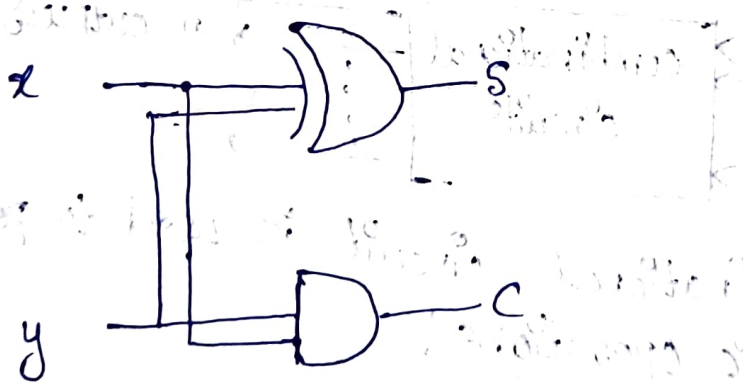
* A full adder is used to perform arthimetic operations on more than two inputs.

Half Adder :-

Truth table $2^2 = 4$ combinations

x	y	Sum S	Carry C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Logic diagram :-



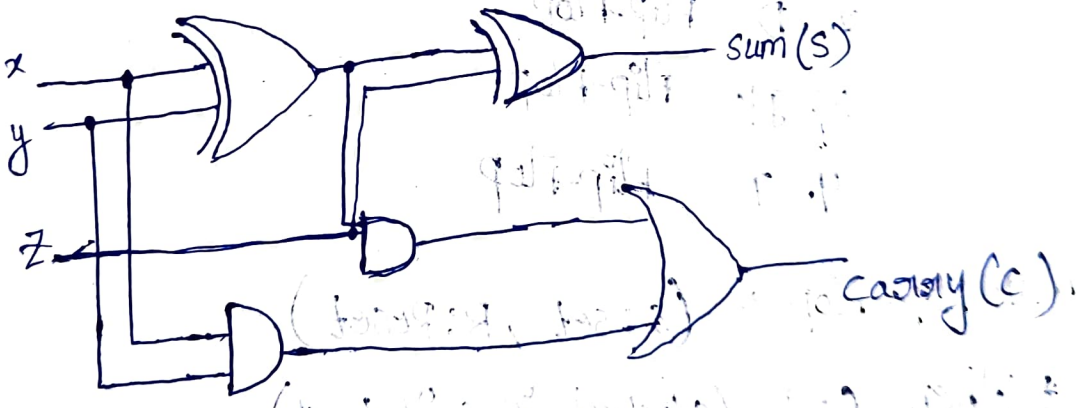
Full adder :-

Truth table $2^3 = 8$ combinations

x	y	z	Sum S	Carry C
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

Logic diagram :- I/P = 3 ← x, y, z

O/P = 2 ← S, C



6. Flip-Flops

* A Flip-Flop is a clocked (c) sequential circuit which is used to store an element (bit) in the form of a binary cell.

* A flip-flop has two outputs, where one output is the value and the other output the compliment of the value.

* AFlip-Flop maintains a binary state until a clock pulse is switched

* There are four types of flip-flops, they are

1. SR Flip-Flop
2. D Flip-flop
3. JK Flip-Flop
4. T Flip-Flop

1. SR Flip-Flop :- (S: set, R: Reset)

* When C=1, (clocked is said to 1)

→ if S=0, R=0, there is no change in this state

→ if S=0, R=1, the flip-flop goes to clear state (i.e, 0)

→ if S=1, R=0, the Flip-Flop goes to set state

→ if S=R=1 the flip-flop goes to intermediate state. (any state)

S	R	state Q(t+1)
0	0	Q(t)
0	1	0 (clear)
1	0	1
1	1	intermediate

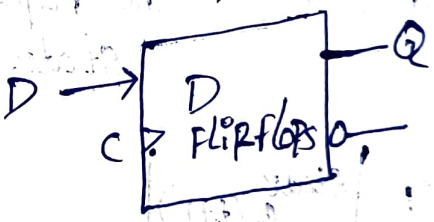
2. D Flip-flop :- (Data flip-flop)

When C=0,

→ if D=0 the Flip-flop goes to clear state (i.e. 0)

→ if D=1, the flip-flop goes to set state (i.e. 1)

D	state (Q(t+1))
0	0
1	1



3. Jk Flip-flop :- (Jack, Kilby)

when C=1

→ if J=0, K=0 then there is no change in state

→ if J=0, K=1 the flip-flop goes to clear state (0)

→ if J=1, K=0 the flip-flop goes to set state (1)

→ if J=K=1, the flip-flop goes to compliment state (inverse)

J	K	State Q(t+1)
0	0	Q(t)
0	1	0 (clear)
1	0	1
1	1	Q'(t)

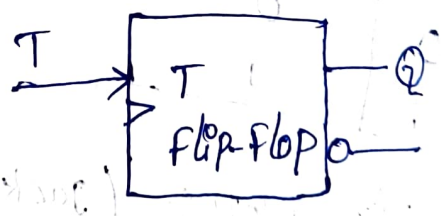
4. T-flip-flop :- (Toggle)

when C = 1

→ if T=0, then there is no change in state

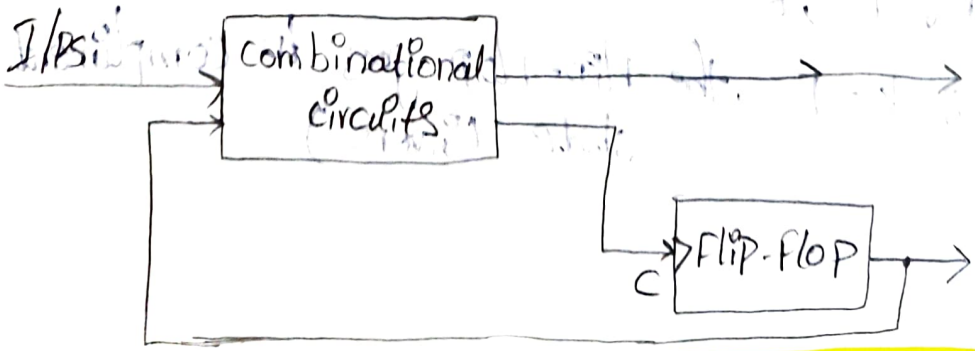
→ if T=1, the flip-flop goes to complement state.

T	State Q(t+1)
0	Q(t)
1	Q'(t)



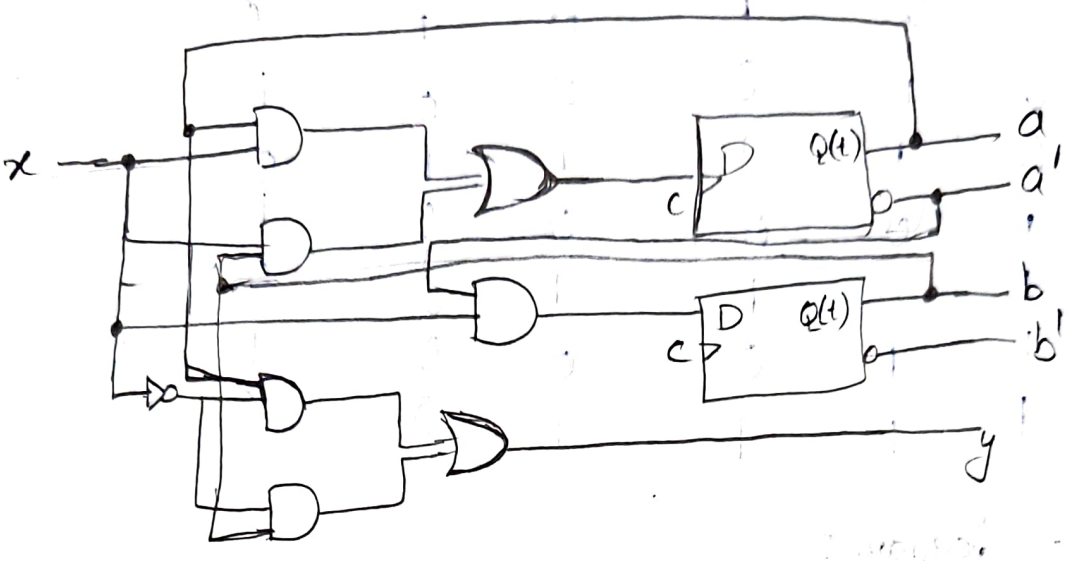
7. Sequential Circuit

* A sequential circuit is a combination of combinational circuits and clocked flip-flops.



Sequential circuit Diagram:-

The sequential circuit diagram given below consists of 5 AND gates, 2 OR gates and two D flip-flops.



State table:-

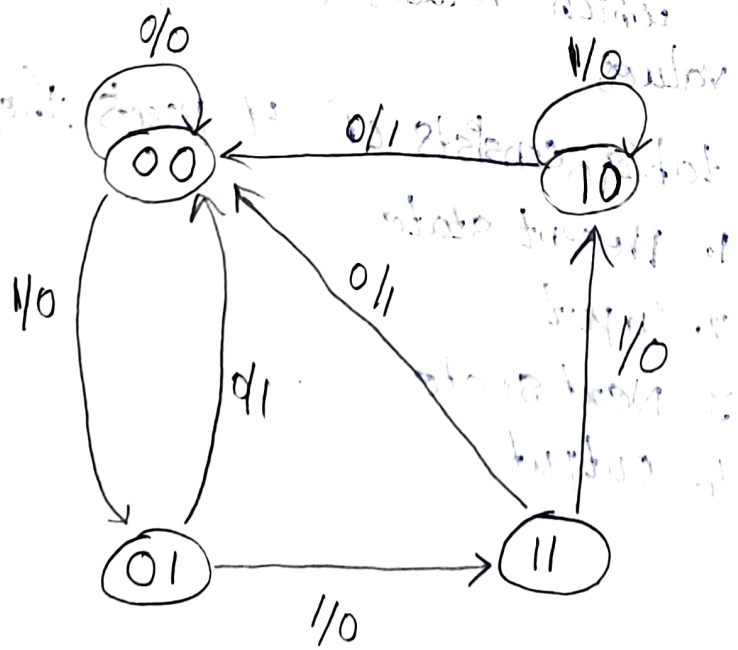
* A state table is used to display the behaviour of sequential circuit which is based on inputs, outputs and the states of flip-flops.
* A sequential circuit is specified by using a state table which relates the outputs and the next state values
* The state table consists of 4 parts. they are
  1. Present state
  2. Input
  3. Next state
  4. output

Present state		Input	next state		Output
a	b	x	a	b	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

State diagram :-

A state diagram is a graphical representation of state table.

In a state diagram, each state is represented by a circle (O) and the transition b/w each state is represented by directed graph.



Part-B

Digital Components

Integrated circuits :-

→ Integrated circuits (IC) are used to construct digital circuit.

→ An IC contains several electronic components to process digital gates

→ An IC can also be called chip which is made of small silicon semiconductor crystal

→ Various logic gates are interconnected to each other in side the chip.

→ IC can be used in

SSI –

MSI

LSI

VLSI

→ SSI stands for small scale Integration, where the devices contain several independent gates in a single unit (<10)

→ MSI stands for Medium scale Integration, where the devices contain several independent gates in a single unit (10-200)

→ LSI stands for large scale Integration, where the devices contain several independent gates in a single unit (200 - 1000 gates)

→ VLSI stands for very large scale integration which where the devices contain several independent gates in a single unit (>1000).

→ The various integrated circuits use TTL

1. TTL (Transistor - Transistor - logic)
2. ECL (Emitter - coupled - logic)
3. MOS (Metal - Oxide - semiconductor)
4. CMOS (complimentary - metal - Oxide - semi conductor)

2. Decoders

→ A decoder is a combinational circuits which converts the binary information from 'n' inputs to 2^n outputs.

→ The decoders are termed as n-to-m line decoder

(or) nxm decoders

Example :-

1. 2-to-4 line decoders

or

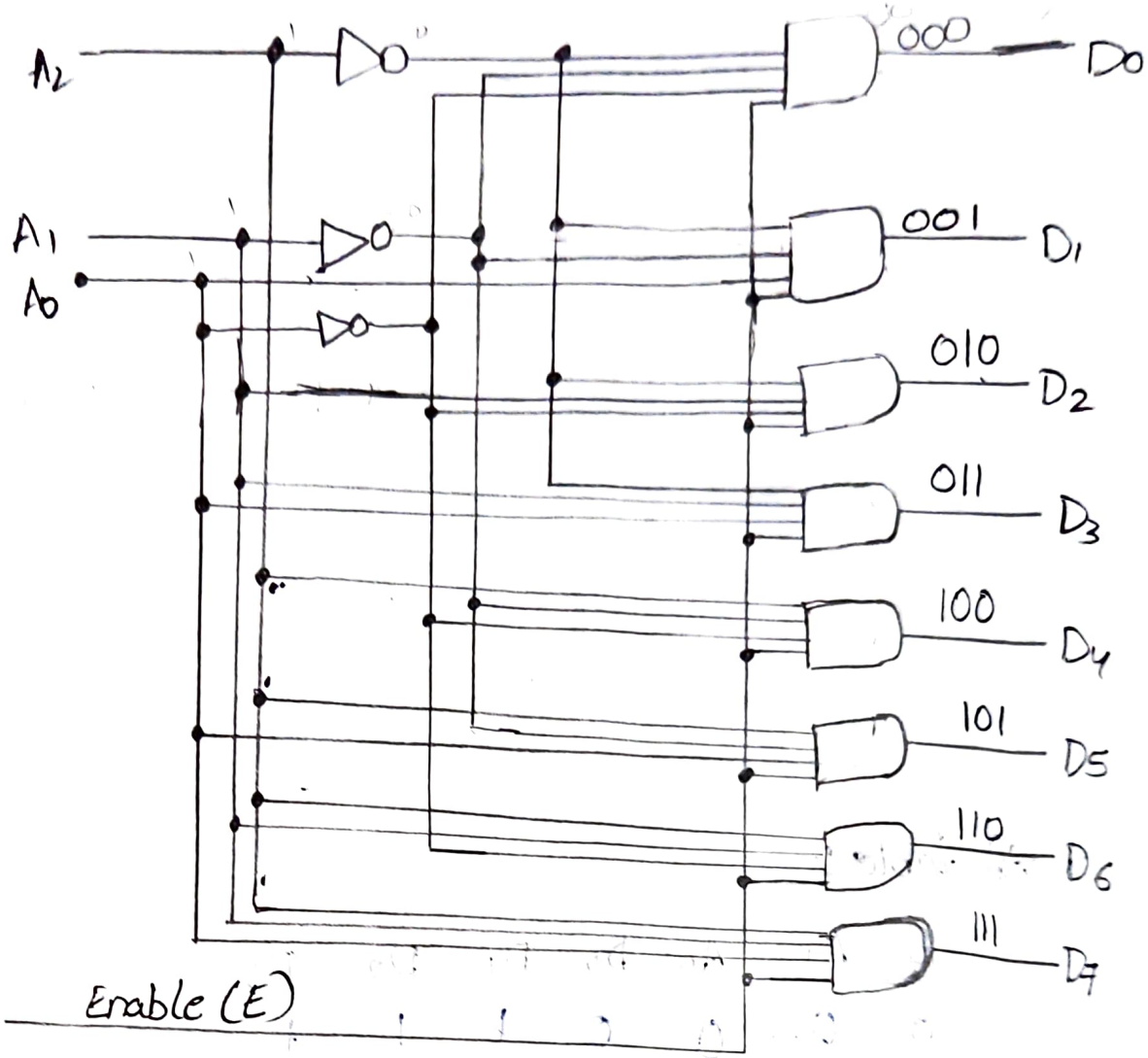
2x4 decoders

2. 3-to-8 line decoder

or

3x8 decoders

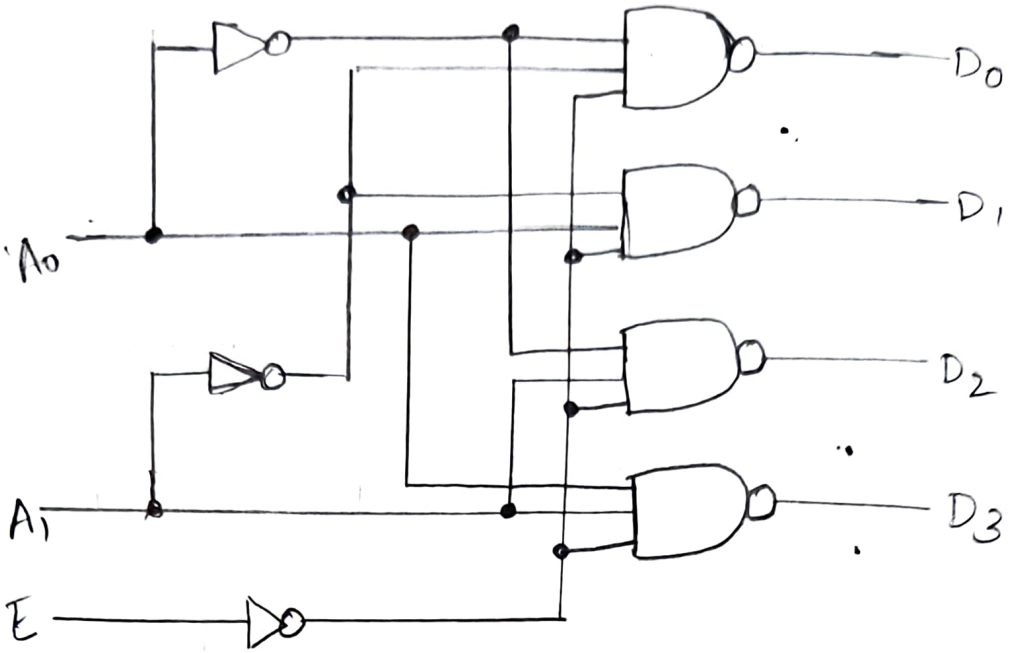
3-to-8 line decoder (3x8 decoder) :-



Enable	Inputs			outputs							
E	A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

2-to-4 line decoder :- (with NAND gates)

Logic diagram :-



Truth table :-

E	A_1	A_0	D_0	D_1	D_2	D_3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	X	X	1	1	1	1

Encoders :-

* An Encoder is a digital circuit which peaforms the inverse operation of the decoder.
* The encoder takes 2^n inputs and produces n output lines
* The output lines generate the binary code corresponding to the input value

N*

Inputs $2^n=8$								Outputs n=3		
D7	D6	D5	D4	D3	D2	D1	D0	A2	A1	A0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

3. Multiplexer

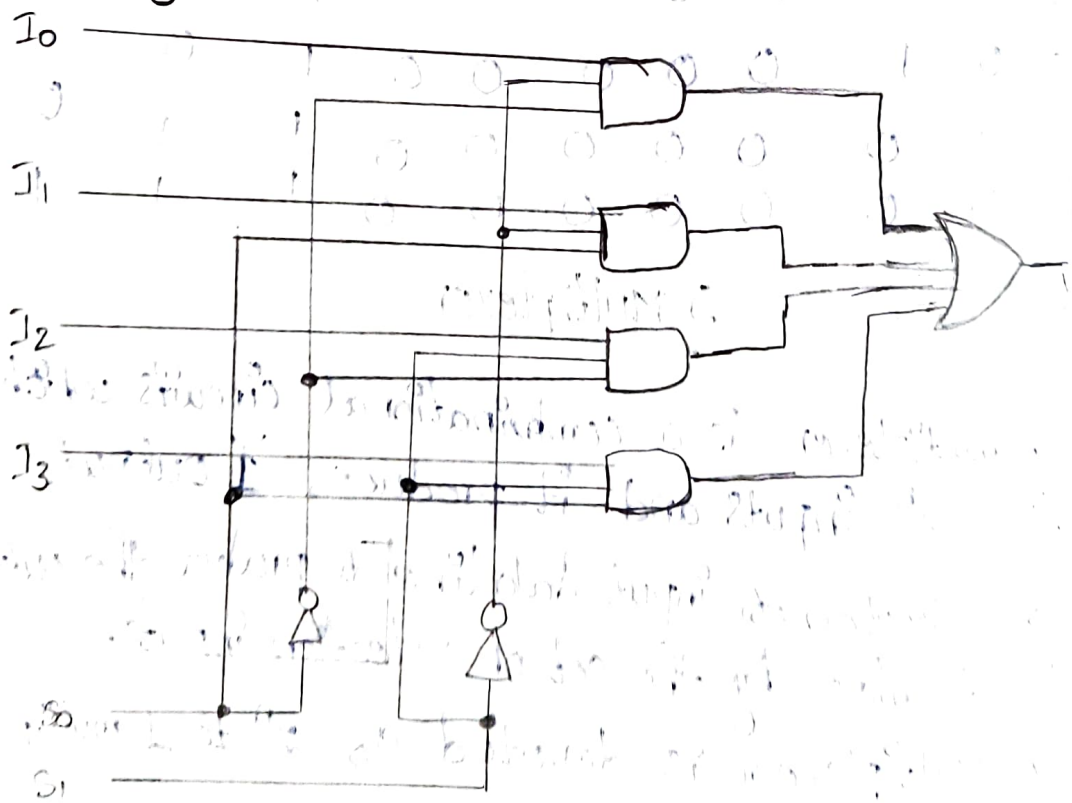
* A multiplexer is a combinational circuits which takes 2^n inputs and it produces 1 output.
* The selection of input data line to produce the output is determine by the set of selection inputs.
* A Multiplexer is denoted as 2^n to 1 multiplexer

Function table :- (for n=2)

* A function table for 4 to 1 line multiplexer consists of n selection inputs and 1 output.
* For n=2, then the selection inputs will be two (S_0, S_1) where as the outputs are represented 'y' (I_0, I_1, I_2, I_3)

selection I/p		o/p
S_1	S_0	y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Logic diagram :-



From the above 4-to-1 line multiplexer, each of the four data inputs $[I_0, I_1, I_2, I_3]$ is applied to one input of an AND gate.

The two selection inputs $[s_0, s_1]$ are decoded to select a particular AND gate.

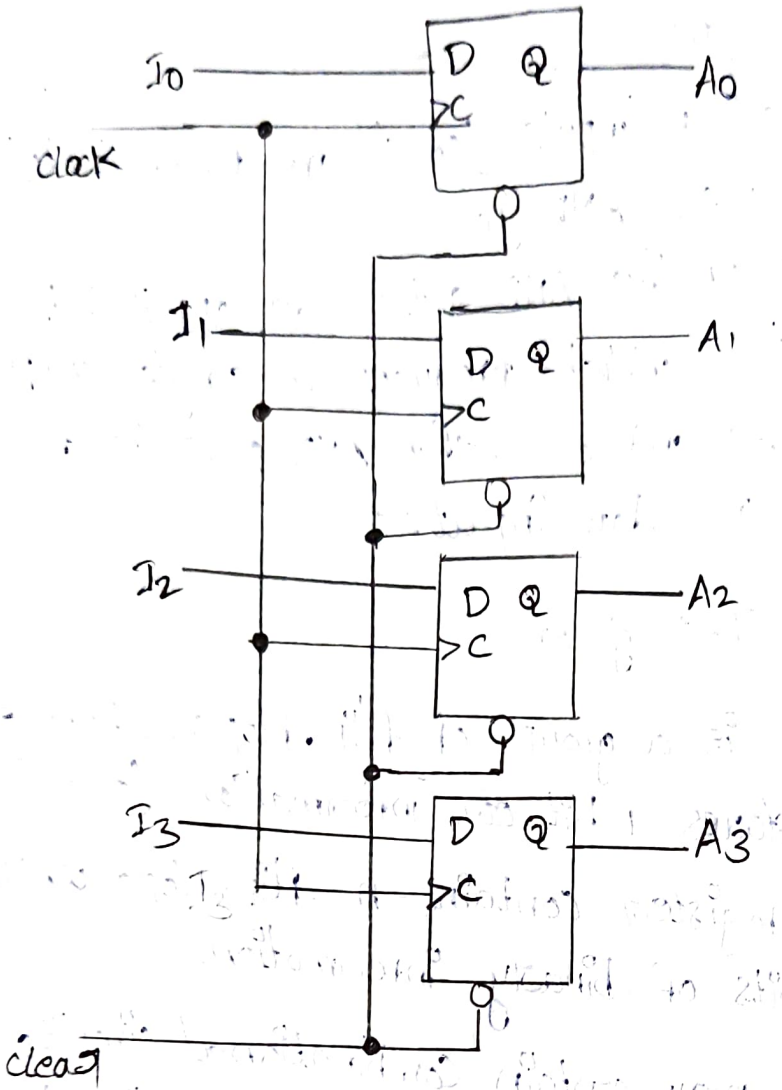
The outputs of the AND gates are applied to a single OR gate which produces a single output.

Ex: when $s_1 = 1$ and $s_0 = 0$, the AND gate is associated with I_2 data input.

4. Registers

* A Register is a group of flip-flops, where each flip-flop stores 1 bit of information (0 or 1)
* A n-bit-register contains n flip-flops which stores n bits of binary information.
* A Register may contain combinational circuits which can perform certain data processing task.
* The flip-flops in the register contains binary information and gates are used to control the new information when transfer to a register.

4-bit-Register: (serial)

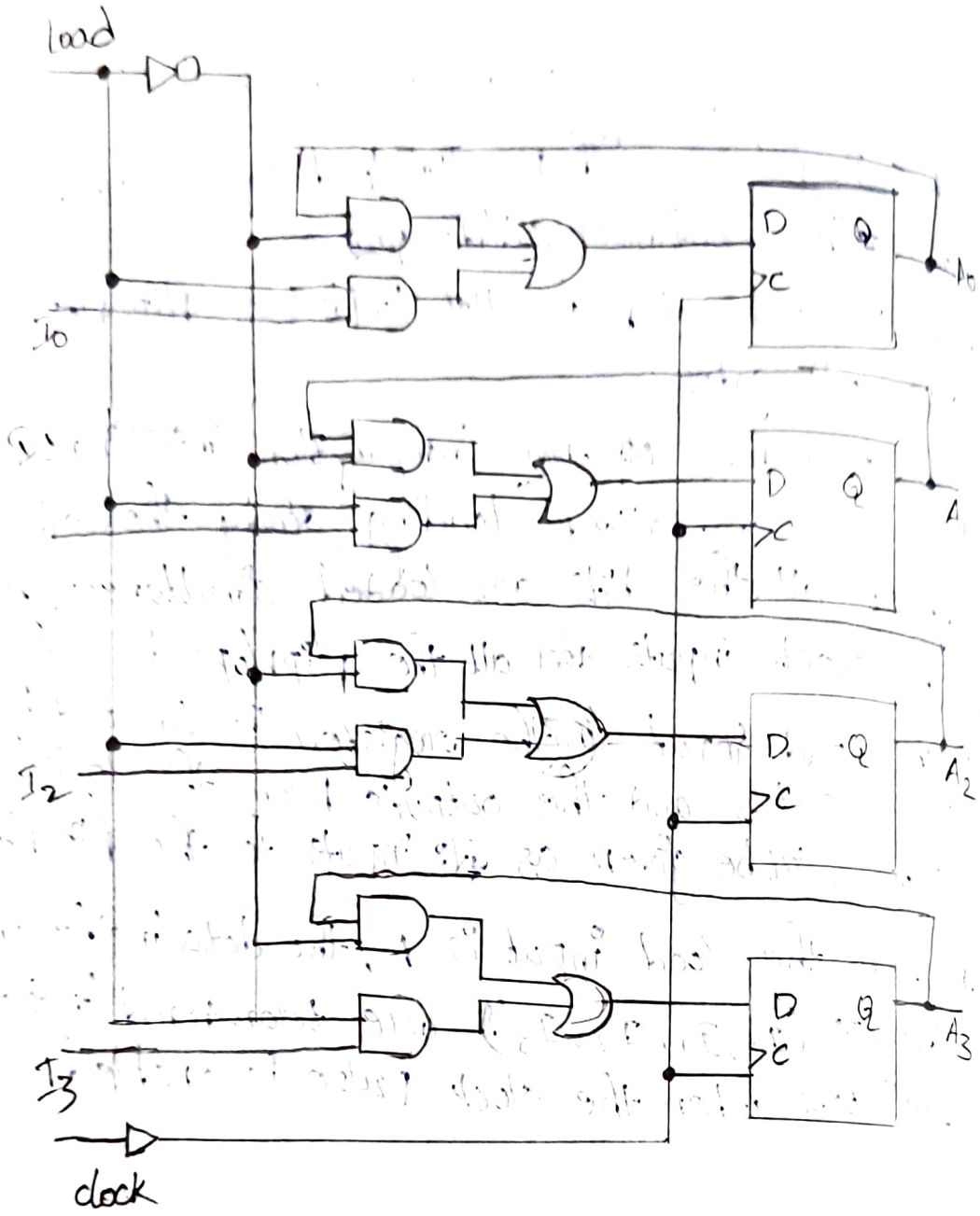


* From the above 4-bit register we use 4 flip flops without external gates
* A common clock and clear are used for the four D-flip-flops
* The data input for the four D-flip-flops are denoted by I_0, I_2, I_3 where as the outputs are denoted by A_0, A_1, A_2, A_3.
* when the clock is enabled (1), we can input the D-flip-flops, else the clock is set to 0

* the clear input is set to 1, else set to 0.

Parallel or Registers with parallel Load:-

* the Registers with parallel load uses a clock that supplies continuous data pulses
* the Register with parallel loads uses flip-flops and also logic gates.
* The transfer of new information into a new register is called "loading" study the register where all the bits are loaded simultaneously
* the clock inputs for all the flipflops is same
* The load input in the register is directed by using gates and the outputs from this logic gates will be given as the inputs to the flip-flops.
* when the load input is 1, the data in the 4 inputs (I_0, I_1, I_2, I_3) are transfered into the register when the clock pulse is enabled



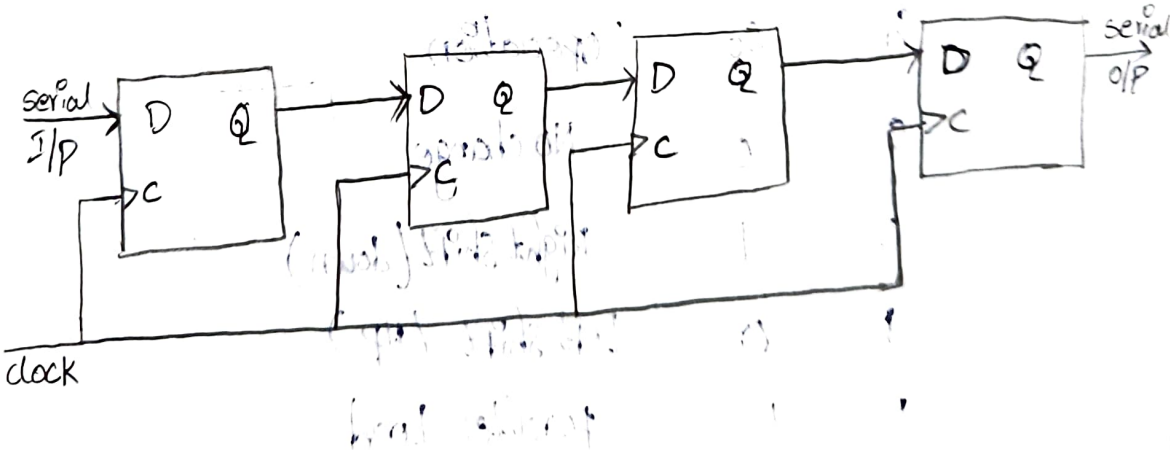
5. Shift Registers

* A Register which is capable of shifting the binary information in both directions is called shift Registers
* The shift Registers contains several flip-flops which are connected to each other

* the output of one flip-flop will be the input to other flip-flop.
* the flip-flops will have a common clock pulse which initiates the shift operation.
* The shift Registers can be used with serial load or parallel load
* A Register which is capable of shifting in one direction only is called unidirectional registers
* A Register which is capable of shifting in both directions is called Bi directional registers

Unidirectional shift Register :-

* Consider the unidirectional shift Register with serial load having four data flipflops (D)
* the data input is only one and the output is also one

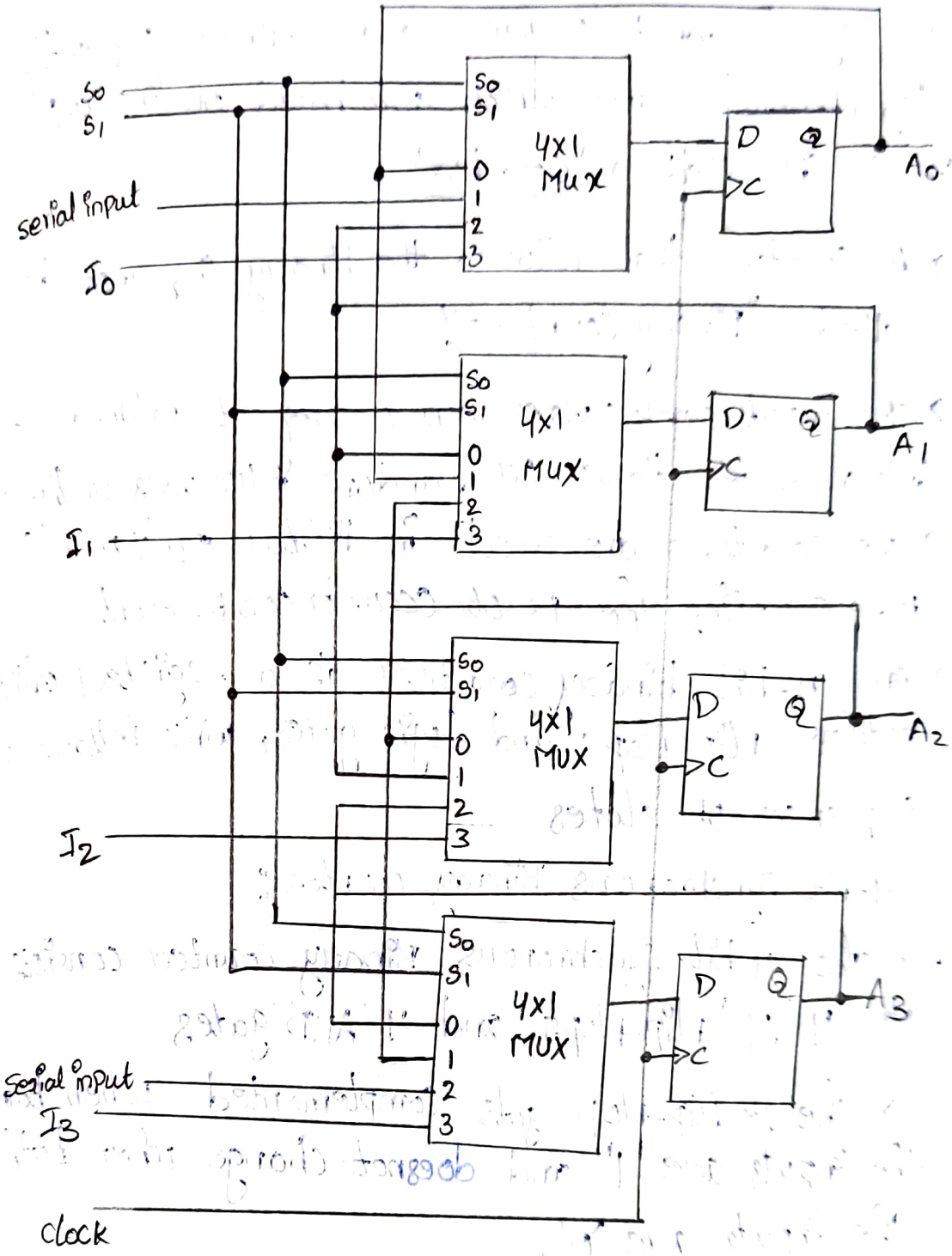


Bidirectional shift Registers :- [Parallel]

* A Bidirectional shift Register with parallel load constructed with '4-D flip-flops' and with 4 by multiplexers.
* It has two selection inputs, denoted by S_0, which controls the mode of operation [left shift or Right ship] of the Registers
* The data inputs are denoted by I_0, I_1, I_2, I_3 which are connected to multiplexers individual
* The serial inputs are connected to first multiplexer and Last multiplexer
* The clock pulse for all the flip-flops is same
* The outputs obtained by the 4 bit bidirectional shift Register with parallel load are 4 which are denoted by A_0, A_1, A_2, A_3.

Function table :-

S_1	S_0	operation
0	0	No change
0	1	Right shift (down)
1	0	Left shift (up)
1	1	Parallel Load


S0
S1
serial input
I0
4x1 Mux
0
1
2
3
D
Q
C
A0
I1
A1
I2
A2
I3
A3
clock

6. Binary Counters

→ A Register which performs a predefined sequential (synchronous) states which are based on input pulse (clock) is called a 'Counter'

→ A Counter that follows the binary sequence is called "Binary Counters".

→ the input clock pulses can occur at uniform intervals of time or random intervals of time

→ the counters are used in digital equipments for counting the no. of occurrences of event

→ An 'n'-bit binary counter is a register which contains flip-flops and logic gates, which follows a sequence of states

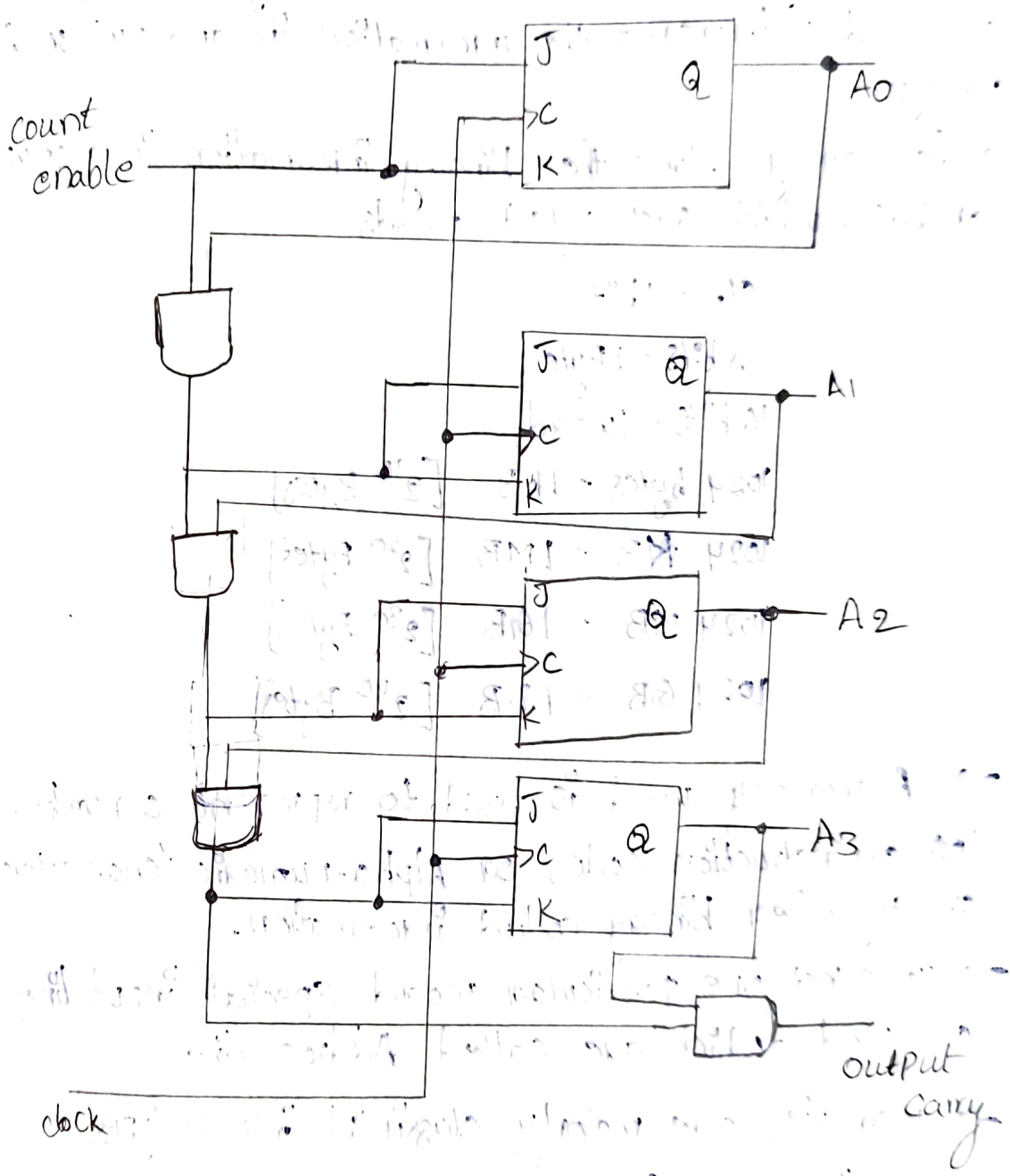
4-bit synchronous Binary counter :-

→ the 4-bit synchronous Binary counter consists of 4 'Jk' flip-flops and 4 'AND' gates

→ The Jk flip-flop gets complemented when both the inputs are '1' and doesnot change when both the inputs are '0'.

→ when the clock is enabled, the binary counter is enabled.

→ The 4 Jk Flip-flops Produce 4 Outputs (A_0, A_1, A_2, A_3) along with a Carry


count
enable
J
Q
C
K
A0
A1
A2
A3
output
carry
clock

7. Memory unit

A memory unit is a collection of storage cells which are used to transfer the information in and out of the storage

→ The memory stores the binary information in a group of bits, which are called words

0/1 = 1 bit
8 bits = 1 Byte
16 bits = 1 word
1024 bytes = 1 kB $[2^{10}$ Bytes$]$
1024 kB = 1 MB $[2^{20}$ Bytes$]$
1024 MB = 1 GB $[2^{30}$ Bytes$]$
1024 GB = 1 T.B $[2^{40}$ Bytes$]$

→ A memory word is used to represent a number (or) an instruction code, or Alpha-numeric characters or any other binary coded information.

→ To select one particular word special input lines are used which are called Address line.

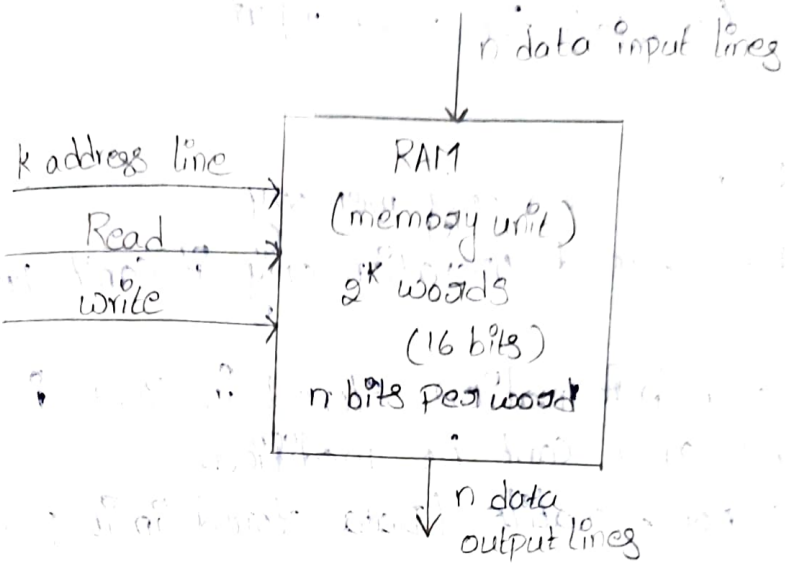
→ Memories are majorly classified into 2 types
  i) RAM [Random Access memory]
  ii) ROM [Read only memory].

i) RAM :-

* It stands for Random Access Memory
* In RAM, the memory cells can be accessed for information transfer from any desire random
* The communication b/w the memory cell and

environmental is achieved through data input and data output lines.

* the block diagram of RAM is given below.



* The n data input lines provides the information to be stored in the memory
* The n data output lines supplies the out of the
* k address line provides a binary No. of k bits which specifies a particular word choosen among the 2^k available inside the memory.
* The true control unit specify the direction of the transfer
* The main operations of RAM are Read & write operations

To transfer (write) a new word into memory)

a) Apply the binary address of desired word into address line

b) Apply the data bits that must be stored in memory into data input lines

c) Activate (Enable) a the write input.

To transfer (read) a stored word out of memory.

a) Apply the binary address of desired word into address line.

b) Activate (enable) the read input

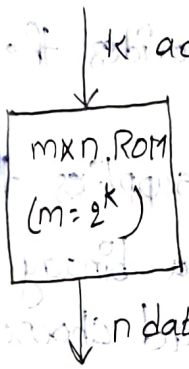
ROM :-

→ It stands for Read only memory

→ ROM performs - read operation only it can't perform write operation.

→ the Binary information stored in ROM is Peamanent and cant be modified.

→ ROM is non-volatile (Data stored in it permanent)



→ An mxn ROM is an array of binary cells original into m words of n bits each

→ ROM has k address lines to select one of $2^k = m$ words of memory and n o/p lines, one for each bit of the word

→ In ROM, the output lines automatically provides the n-bits of the word selected by the address value.

→ ROM is used in design for control units for digital Computers.

→ A central unit that utilizes ROM to store binary control information is called a Microprogrammed

control unit.

Different types of ROM are

i) PROM :-

programmable read-only memory

It allows users to program to achieve desired relationship b/w input address and stored words.

ii) EPROM :-

Erasable programmable read only memory

The fixed pattern in the program can be altered using ultra violet light.

iii) EEPROM :-

Electrically Erasable programmable read only memory

The fixed pattern in the program can be altered using electrical signals.

NOTE :- ROM, PROM (can't modify data)

EPROM, EEPROM (can modify data).

Unit-2

Part-A (Data Representation)

1. Data types ✗ mid
2. Complements ✓ (ALL)
3. Fixed-point Representation ✓ } Endexam 1½
4. Floating-Point Representation ✓ }

Part-B (Register Transfer and Micro Operations)

1. Register Transfer Language
2. Register Transfer ✓
3. Bus and Memory Transfer ✓ [Bus transfer]
4. Arithmetic Micro operations ✓
5. Assembly Language Instructions
6. 8085 Architecture ✓ (components)
7. 8085 Instruction set

Part A [Data Representation]

1. Data types

* The data types are used in "Registers" of digital computers.
* The data types can be numbers, Alphabets and special characters
* The data types are represented in BCD (Binary coded Decimal) form
* To Perform arthimetic operations we use various numbers system in binary form.

Number system :-

The various number systems are

Type	Numbers	Radix	Eg
Binary	0, 1	2	$1011_{(2)}$
Decimal	0, 1, 2, ... 9	10	$25_{(10)}$
Octal	0, 1, 2, ... 7	8	$65_{(8)}$
Hexadecimal	0, 1, 2, ... 9, a, b, c, d, e, f.	16	$C5_{(16)}$

* The radix (r) = 2 represents binary number system which has two value (0, 1)

* Radix (r=10), represents decimal number system which has 10 value [0,1,2,....9]

* Radix (r=8), represents Octal number system which has 8 values [0,1,2,...7]

* Radix (r)= 16, represents Hexa decimal number system which has the values (0....9, a,b,c...f)

Conversion :-

* Conversion of one number into another Number System is done, based on the value and Radix (r)

* While converting, the process should be done individually on integer part and ~~fact~~ Fractional Part.

* The decimal, octal, or hexa decimal values can be converted into binary coded values.

Eg:- $45_{(10)}$ is equivalent to 101101. (2)

$478_{(10)}$ is equivalent to 736 (8)

$243_{(10)}$ is " " F3 (16)

$41.6875_{(10)}$ is equivalent to $101001.1011_{(2)}$

12 (8) is equivalent to 1010 (2)

62 (8) " " " 110010 (2)

32 (16) " " " 00110010 (2)

F8 (16) " " " (2)

1, $45_{(10)} \Rightarrow 101101_{(2)}$

2 | 45
2 | 22 - 1
2 | 11 - 0
2 | 5 - 1
2 | 2 - 1
1 - 0

$1\,0\,1\,1\,0\,1$
$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$32+0+8+4+0+1=45$

2, $478_{(10)} \Rightarrow 736_{(8)}$

8 | 478
8 | 59 - 6
7 - 3

$7\ \ 3\ \ 6$
$8^2\ \ 8^1\ \ 8^0$

$448+24+6=478$

4, $41.6875_{(10)} \Rightarrow$ 101001.1011

41.6875

2 | 41
2 | 20 - 1
2 | 10 - 0
2 | 5 - 0
2 | 2 - 1
1 - 0

6875×2
1.3750 (1st)
3750×2
0.7500 (2nd)
7500×2
1.5000 (3rd)
5000×2
1.0000 (4th value)

$\Rightarrow 101001.1011_{(2)} = 41.6875_{(10)}$

1011

formula:-

$\left(\text{val } 1 \times \frac{1}{2^1}\right) + \left[\text{val } 2 \times \frac{1}{2^2}\right] + \left[\text{val } 3 \times \frac{1}{2^3}\right] + \left[\text{val } 4 \times \frac{1}{2^4}\right]$

$= \left(1 \times \frac{1}{2}\right) + \left(0 + \frac{1}{4}\right) + \left(1 \times \frac{1}{8}\right) + \left(1 \times \frac{1}{16}\right)$

$= 0.5 + 0 + 0.125 + 0.0625$

$= 0.6875$

101001
$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$
$32+0+8+0+0+1$
41

5, $12_{(8)} \Rightarrow 1010_{(2)}$

i) $12_{(8)} = 10_{(10)} = 1010_{(2)}$

1 2

8^1 8^0

$1\times8 + 1\times2 = 10$

2|10
2|5 - 0
2|2 - 1
1 - 0

ii) $12_{(8)}$

1 | 2
2|2
1 - 0

$1_{(2)} = 10_{(2)}$

$001 + 010 = 001010$

6, $62_{(8)} \Rightarrow 110010_{(2)}$

2|6 2|2
2|3 - 0 1 - 1
1 - 1

$110 + 010 = 110010_{(2)}$

7, $32_{(16)} \Rightarrow 00110010_{(2)}$

3 2

11 10

011 010

011010

8, $F8_{(16)} \Rightarrow 11111000_{(2)}$

F → 15

2|15
2|7 - 1
2|3 - 1
1 - 1

8

2|8
2|4 - 0
2|2 - 0
1 - 0

1111 1000

2. Complements

→ Complements are used in digital computers for simplifying the subtraction operation.

→ Complements are of two types, based on Radix (r)
these are;

1. r's complement
2. $(r-1)^{s}$ complement

→ For decimal notation r = 10 then
r's complement is 10's complement
(r-1)'s is 9's complement.

→ For Binary notation r = 2 then
r's complement is 2's complement
$(r-1)^{s}$ is 1's complement.

Decimal Notation

$(r-1)^{s}$ complement (9's complement):-
The $(r-1)^{s}$ complement of number (N) for n digits is given by

9's → $\boxed{(r^n - 1) - N}$

Eg:-

r = 10, let n = 2, and N = 20

$= (10^2 - 1) - 20$

$= 79$

2. r^s Complement (10^s complement):-

The r^s. complement of number (N) for n digits is given by.

$10's \rightarrow \boxed{[(r^n-1)-N]+1}$

Ex:- $r=10$, Let $n=2$, and $N=20$

$=[(10^2-1)-20]+1$

$=(100-1)-20+1$

$=80$

Binary Notation

$(r-1)^s$ complement (1's complement):-

The $(r-1)^s$ complement is obtain by changing

0's to 1's & 1's to 0's

Eg:- 1011001

i) The 1^s complement for 1011001 is 0100110

ii) Manual checking

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

r's Complement (2's complement):

The 2's complement is obtained by changing 0's to 1's & 1's to 0's and then add one (1) to the LSB (Least significant Bit)

Eg: 1,011001

→ the 1's complement for 1011001 is
0100110

→ the 2's complement of above 1's complement is 0100110
+ 1
0100111

Decimal Number:

Eg: M-N = 72532 - 13250
= 59282

M = 72532
N = 13250

a) 10's complement (N)
9's complement (N)
99999
- 13250
86749

2) 10's complement (N):
9's complement + 1
86749 + 1
= 86750.

Eg: 13250 - 72532
= -59282

M = 13250
N = 72532

a) 10's complement (N)
9's com (N): 99999
- 72532
27467
10's com (N) + 1
27468

b)

b) 10's complement(N) + M

72532
+ 86750

if carry, 1 59282
-100000
then - 1 discard 059282

b) M + 10's comp(N)

13250
+27468

← 40718

No carry → 10's comp, -ve sign

c) 10's comp (result) M+N

40718

a) 9's comp

99999
- 40718
59281

b) 10's comp

+1
59282

c) Add -ve sign

-59282

Binary Numbers :-

Eg:- M-N = 1010100 - 1000011 = 17

a) 2's complement(N)

a) 1's complent (N): 1000011
1's com
0111100

b) 2's complement + 1
0111101

Eg:- 1000011 - 1010100 = -17

a) 2's comp(N)

1) 1's comp(N): 1010100
1's com
0101011

b) 2's complement + 1
0101100

No carry.

→ 1's comp, -ve sign

c) 10's comp result

b) m + 2's comp(N)

```
  1010100
+ 0111101
-----------
1 0010001
-10000000
-----------
0 0010001 = 17(10)
```

If carry, then discard

b) 1's comp m + 2's comp(N)

```
  1000011
  0101100
-----------
1 1101111 - result
```

No carry

c) 2's comp(result), -ve sign

a) 1's comp : 0010000

b) 2's comp : 0010001

c) sign -0010001 ≈ -17(10)

3. Fixed-point Representation

→ A digital computer represents a positive (+) number with '0' and a negative (-) number with '1' in the negative (-) number with MSB (Left)

Eg:- sign | magnitude

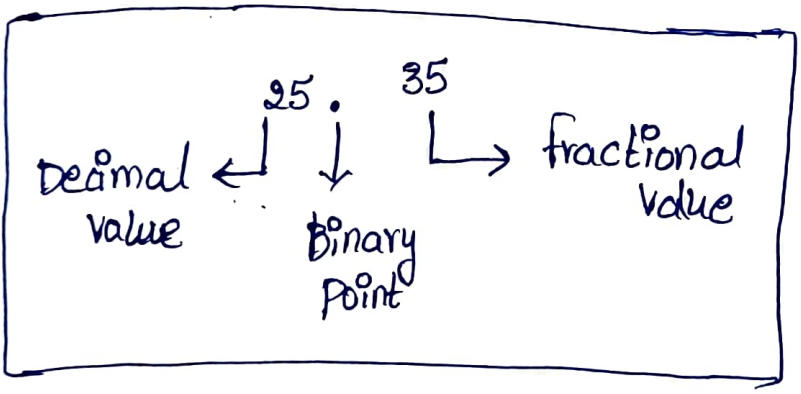
0 | 1101 = +13

1 | 0110 = -6

→ To specify the position of binary point in a register, there are 2 types of representation, they are;

a) Fixed-point Representation

b) Floating-point Representation



Fixed-point Representation :-

→ Fixed-point representation assumes that the binary point is always fixed on one position

→ The positions of fixed-point can be extreme left (or) extreme right

Ex:- Extreme left (.2345)

↓

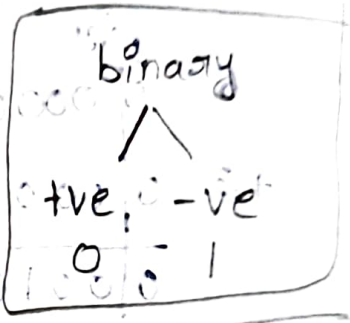
Fractional value

Extreme right (567.)
↓
decimal value

Integer Representation :-

→ The signed integer numbers can be represented in 3 ways, they are

+ve 1, signed - magnitude
-ve { 2, signed 1's complement
3, signed 2's complement



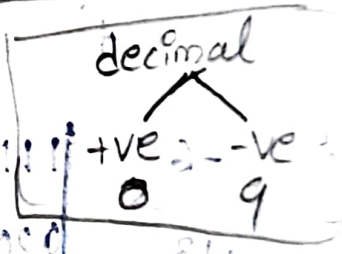
1, Signed - Magnitude :-

Eg:- $14_{(10)} \approx 1110_{(2)}$

↓

sign
0 | 000 1110
↓ ↓
+ve sign magnitude

$= 1111\ 0010_{(2)}$



Eg:- $-14_{(10)} \approx$ 2's complement of $14_{(10)}$

0000 1110
sign magnitude.

a) 1's complement 1111 0001 (signed 1's complement)

b) 2's complement sign . 1 (signed 2's comp)
(1's compt+1) 1|111 0010 = -ve
= 111 0010

Example 1:-

Perform arithmetic addition of two numbers with signed magnitude of binary values.

Sol: Let the two values be 6 and 13 we get 4 Possibilities to perform addition, they are

6+13 -6+13

6-13 -6-13

1, 6 → sign 0 | 0000110

+13 0 | 0001101

0 | 0010011

+

-6 1 | 1111010

+13 0 | 0001101

1 | 0000111

↓ sign

+

-6 → 2's complement

(+6) 0 0000110

1's 1 1 1 1 1 0 0 1

2's 1 1 1 1 0 1 0

3, 6 0 0000110

-13 1 1 1 1 0 0 1 1

1 1 1 1 1 0 0 1

↓ sign -

-13 → 2's complement

(+13) 1's 0 0 0 0 1 1 0 1

2's 1 1 1 1 0 0 1 0

1

1 1 1 1 0 0 1 1

4, −6 +1111010
−13 00001011

11101101

↓

-sign.

Ex-2:-

Perform arithmetic addition of two numbers with signed magnitude of decimal values

Let the two values be 375, -240.

375 (-240)

↓

375 + 10's com (240)

a) 10's comp (240) = 760

1. 9's comp : 999
   -240

   759

2. 10's comp (9's+1) :- +1

   760

Note:-

The positive sign is indicated by '0' & negative sign is indicated by '9' in decimal magnitude.

b) (375) + (-240)

↓ 10's comp (240)

+ve 760

+ve

↓ ↓

(0375 + 9760)

sign	magnitude
0	375
9	760
discard 1	0135

↓

+ sign.

4. Floating - Point Representation

→ A Floating point representation uses two Registers

→ the first Register stores the number, which is called "Mantissa / Fraction" (m)

→ The second Register stores the position of the binary point of the first Register, which is called "Exponent."(e)

→ The floating point number is given by

$m \times r^e$, where m denotes Mantissa

e denotes Exponent and r denotes Radix(r=

Example:-

$m = +6132.789$, $e = 4$

The above values are represented as

+ve ← BP → -ve

6132.789

Mantissa(m)	Exponent(e)
0.6132789	+4

The binary number for the above mantissa (m) are 0.6132789 & exponent (e) = 4 is represented as

Mantissa(M/F)	Exponent(e)
01001110	sin 0|00100

sign 0|1001110

+ 0.6132789

1st ×2 → 1.2265578

2nd ×2 → 0.4531156

3rd ×2 → 0.9062312

4th ×2 → 1.8124624

5th ×2 → 1.6259248

6th ×2 → 1.2498496

7th ×2 → 0.4996992

2 | 4
2 | 2 - 0
2 | 1 - 0
0 - 1

→ The Floating point representation for binary values will be 8 bits for mantissa (m) and 6 bits exponent (e).

PART-B

1. Register Transfer Language :-

* A Register transfer language is represented using symbolic Notations which is used to describe the micro operation transfers between registers.
* A Micro operation is an elementary operation which is used to store information in one or more registers.
* the result of the operation may replace the Previous binary information of a register.
* the Micro operations like increment & load are performed on a binary counter with parallel load.
* A bi-directional shift register is capable of performing shift-Left, and shift-Right operation.
* the internal hardware organisation of a digital computer can be specified by:
  a) the set of registers it contains and their function.
  b) the sequence of Micro operations performed on the binary information which is stored in the registers
  c) the control that initiates the sequence

PART-B

1. Register Transfer Language :-

* A Register transfer language is represented by using symbolic Notations which is used to describe the micro operation transfers between registers.
* A Micro operation is an elementary operation which is used to store information in one or more registers.
* the result of the operation may replace the Previous binary information of a register.
* the Micro operations like increment & load are performed on a binary counter with parallel load.
* A bi-directional shift register is capable of performing shift-Left, and shift-Right operations.
* The internal hardware organisation of a digital computer can be specified by:

a) the set of registers it contains and their function.

b) the sequence of Micro operations Performed on the binary information which is stored in the registers

c) the control that initiates the sequence

Micro operations.

2. Register Transfer

* Registers are denoted with capital letters [Alphabets] followed by numeric value

Eg:- R1 ; R2 , Memory address Register , IR , PC

R_1 → process Register 1

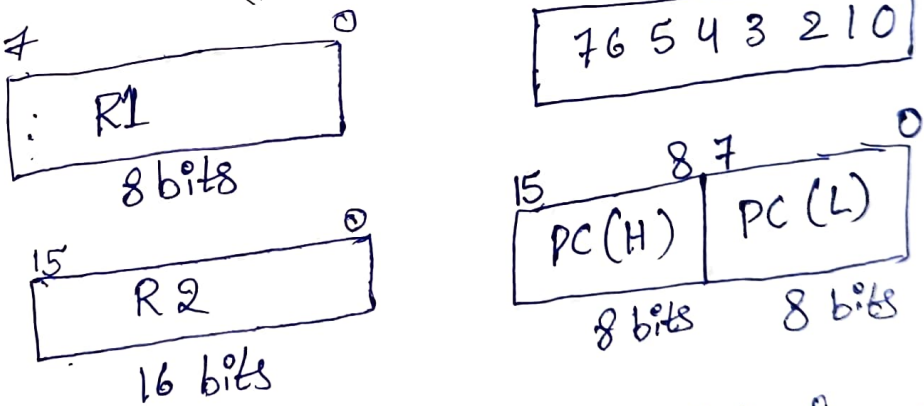
R_2 → process Register 2

MAR stands for Memory address Register

IR stands for Instruction Register

PC stands for Program counter.

Block diagram of Register:-



From the above example PC(L) is referred to "lower-order byte", whereas PC(H) is refered has "Higher-order byte"

→ By using replacement operator ←, the information (or) data from one register is transformed to another register.

Eg:- $R_2 \leftarrow R_1$

→ A control function which is denoted by 'P' uses a boolean variable.

Ex:- if (P==1) then $R_2 \leftarrow R_1$

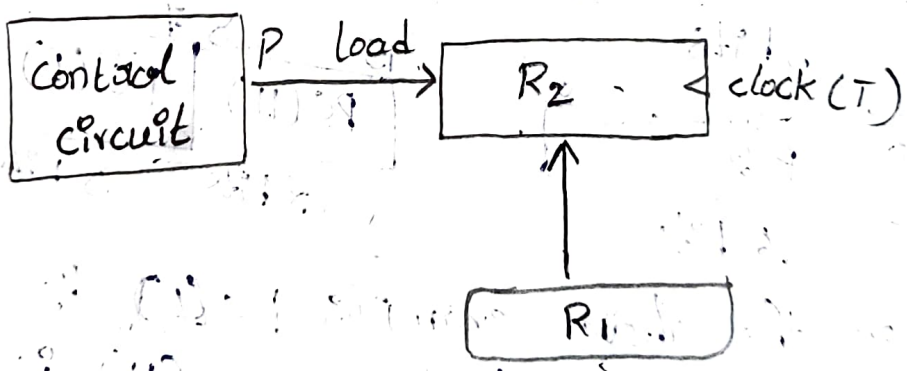
The above statement can be return in register transfer language as $P: R_2 \leftarrow R_1$ (P-control variable)

→ A common clock can be used to perform multiple operations, which is denoted by 'T'

Eg:- clock

$T: R_2 \leftarrow R_1, R_1 \leftarrow R_2$

The block diagram for the transfer from R_1 to R_2 when P=1



3. Bus and Memory Transfer

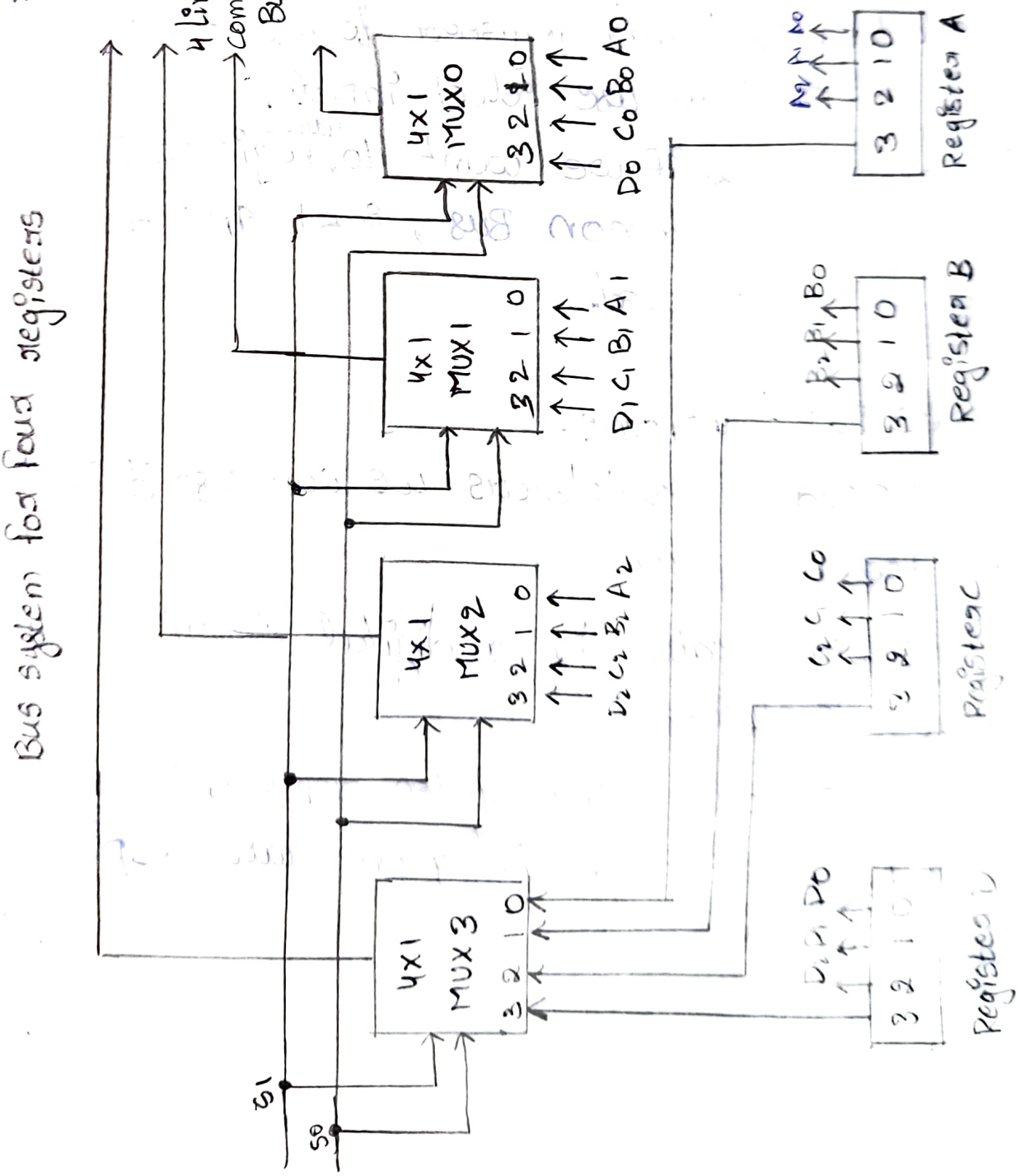
Bus transfer:-

* A digital computer has many registers which transfer information from one register to another registers using a common bus system

* A bus structure consists of set of common lines, one for each bit of a register.
* control signals determines which register is selected by the bus.
* To construct a common bus system, we use multiplexers.
* The multiplexers select the source register whose binary information is placed on the bus

Bus system for Four Registers:-

*



Function table :-

S_1	S_0	Register
0	0	A
0	1	B
1	0	C
1	1	D

* To select a particular register to use the common bus, we need to use select inputs.
* For example, if we want to select register B to use the common Bus, $S_0 = 1$, and $S_1 = 0$ are to be selected.

3-state Bus transfer :-

* Instead of multiplexers we can use 3 state gates for bus system.
* A 3-state gate is a digital circuit that produces 3-states
* In this 3 states, two states represent signals (0 or 1) and the remaining one state represents open circuit

Memory Transfer :-

A memory word is denoted by M

* One memory word is selected by the memory address during the transfer.
* The transfer of information form a meaning/ memory word to outside environment is called "read operation".
* The transfer of new information to be stored into the memory word is called "write" operation.
* It is neccesary to specify the address of M when writing memory transfer operations. It is done by enclosing the address in square brackets.

Ex:- Read : DR ← M[AR]

* AR : Address register DR : Data regists.

From above example, the information transfers from M to DR. where M is selected based on AR

Ex:- Write : M[AR] ← R,

From the above example, transfer of information from R_1 to M based on AR will be done.

4. Arithmetic Micro Operations

* A micro operation is an elementary operation performed with the data that is stored in a registers.
* Micro operations in a digital computer are classified into Four distinct categories, they are

a) Register transfer Micro operation.
[Transfer data from one register to another register]

b) Arithematic Micro operation [arithmetic operation]

c) Logic Micro Operation [bit Manipulation]

d) shift Micro operations [Perform shift operation]

Arithmetic Micro operation :-

* The basic arithmetic Micro operations are

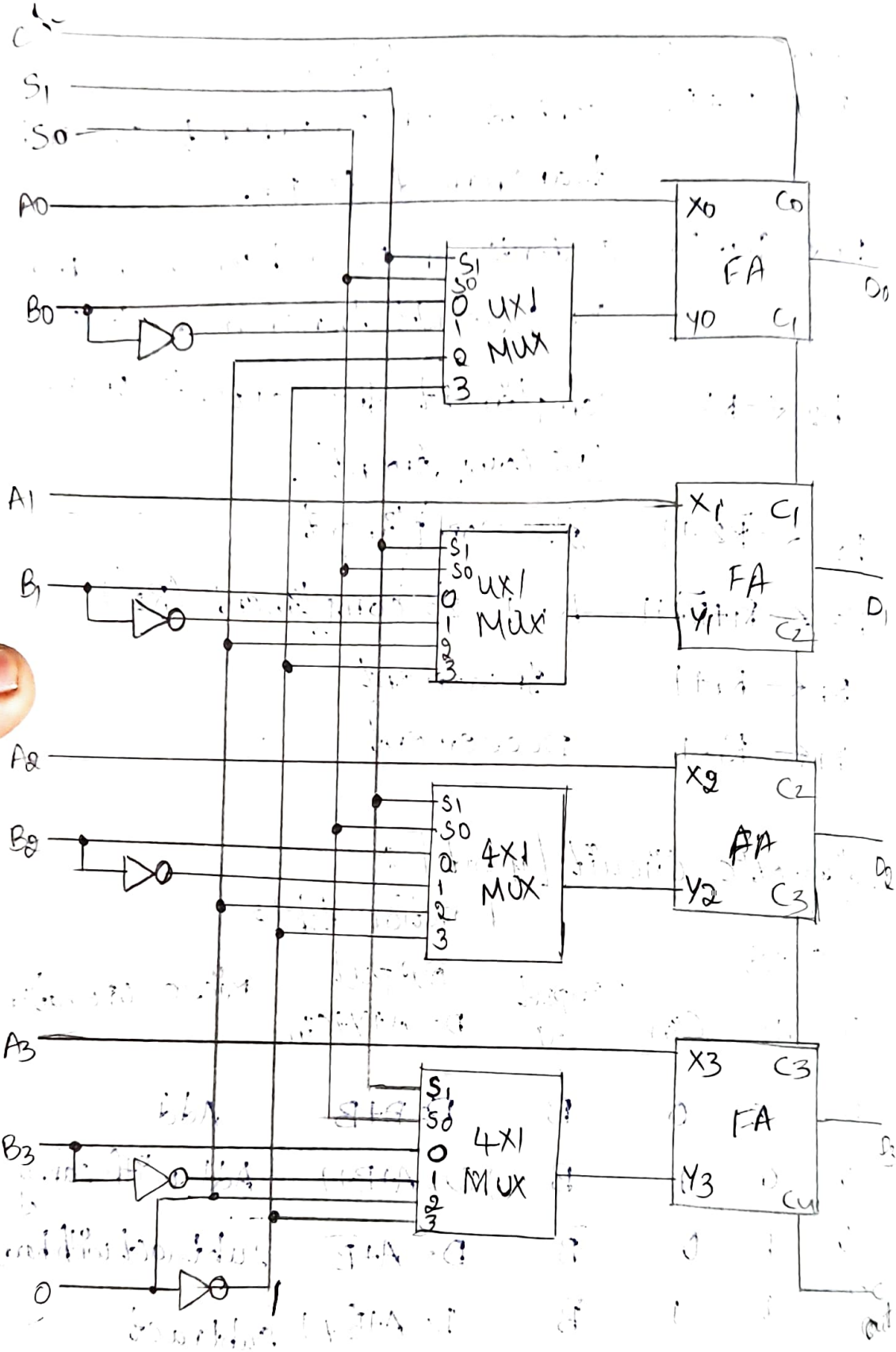
a. Addition
b. Substraction
c. Increement
d. Decrement
e. shift

Arithmetic MO	Description
$R_3 \leftarrow R_1 + R_2$	contents of R_1 added to R_2 and transferred to R_3.
$R_3 \leftarrow R_1 - R_2$	contents of R_2 substraction from R_1 & result transferred to R_3
$R_2 \leftarrow \overline{R2}$	complement the contents of R_2 (1's complement).
$R_2 \leftarrow \overline{R2} + 1$	2's compliment
$R_3 \leftarrow R_1 + \overline{R_2} + 1$	R_1 + 2's compliment (R_2)
$R_1 \leftarrow R_1 + 1$	Increament
$R_1 \leftarrow R_1 - 1$	Decrement

Arithmetic Circuit [4 bit] :-

Function table

Select			Input	output	Micro operation
S_1	S_0	C_{in}	Y	$D = A + Y + C_{in}$	
0	0	0	B	$D = A + B$	Add
0	0	1	B	$D = A + B + 1$	Add with carry
0	1	0	$\overline{B}$	$D = A + \overline{B}$	Subtract with borrow
0	1	1	$\overline{B}$	$D = A + \overline{B} + 1$	subtract
1	0	0	0	$D = A$	Transfer A
1	0	1	0	$D = A + 1$	Increament A
1	1	0	1	$D = A - 1$	Decrement A
1	1	1	1	$D = A$	Transfer A



Eg:-

* To perform addition operation, $S_1 = 0$, $S_0 = 0$ & $C_{in} = 0$

5. Assembly Language Instructions

* Assembly language is low-level language which communicates directly with the computer hardware [CPU].
* An assembly language contains several assembly language instructions.
* "Mnemonics" are used in assembly language instructions to represent a particular operation

Examples:-

ADD	R_1	R_2
SUB	R_2	R_1

Mnemonics

From the above example, the mnemonics are ADD and SUB.

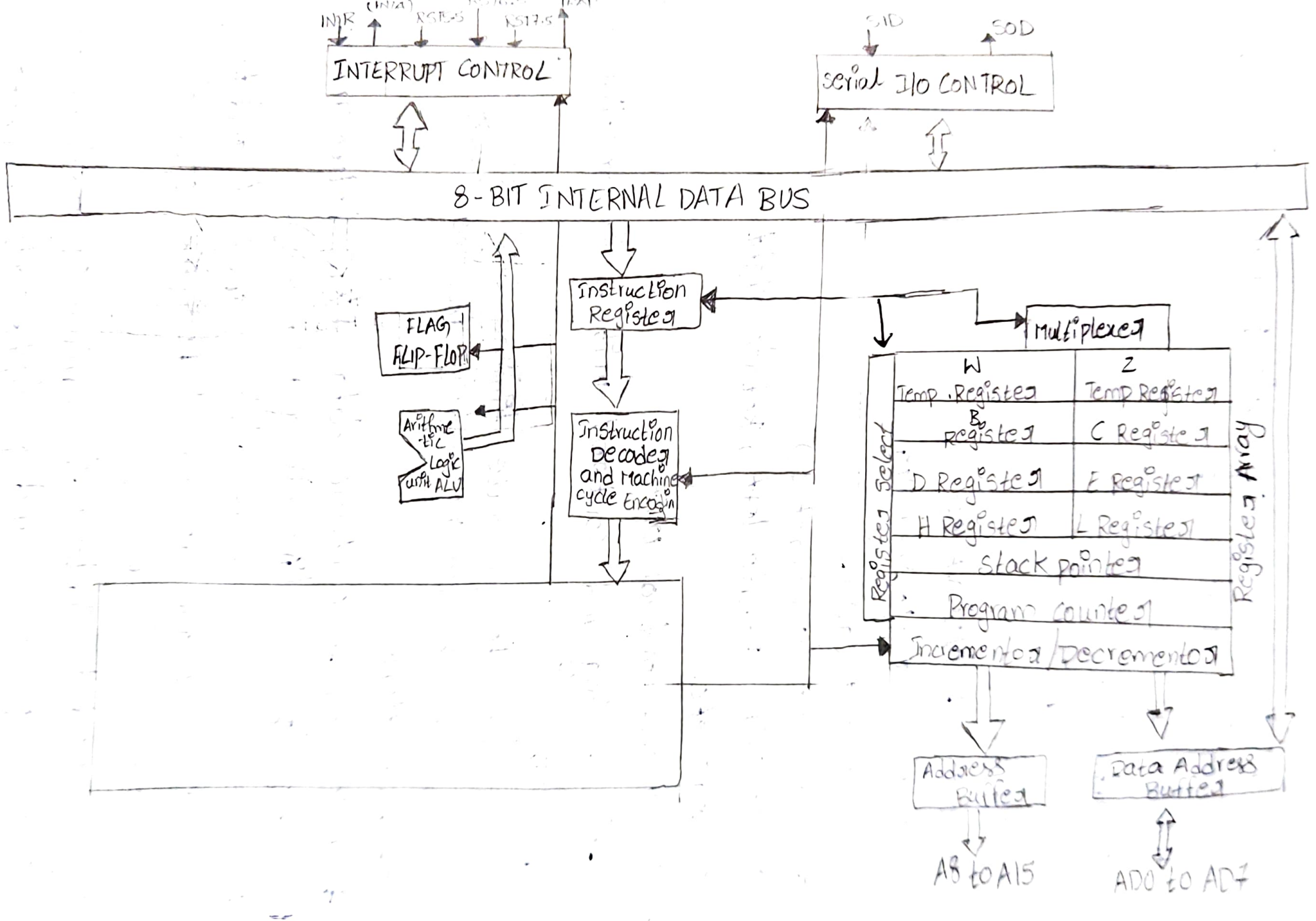
* Mnemonics can also be called as "OP-code" [operation code]
* Assembly language instruction consists of Mnemonics followed by operands.
* Once the assembly language instruction is executed, the result can be transferred to memory location.

6. 8085 Architecture

→ 8085 is an 8-bit Microprocessor designed by Intel in 1977.

8085 Architecture configuration :-

* It is a 40 pin I.C. package fabricated on a single LSI chip.
* the intel 8085 uses a single +5V d.c. supply for its operation.
* Intel 8085s clock speed is about 3 MHz; the clock cycle is of 320ns.
* 8 bit data bus. Address bus is of 16-bit, which can address up to 64kB
* 16-bit stack pointer, 16 bit PC (program counter)
* six 8-bit registers are arranged in pairs: BC, DE, HL.


INTR
(INTA)
RST5.5
RST7.5
INTERRUPT CONTROL
SID
SOD
serial I/O CONTROL
8-BIT INTERNAL DATA BUS
Instruction Register
FLAG FLIP-FLOP
Arithmetic Logic unit ALU
Instruction Decoder and Machine cycle Encoder
Multiplexer
W
Z
Temp. Register
Temp Register
B Register
C Register
D Register
E Register
H Register
L Register
Stack pointer
Program counter
Incrementor/Decrementor
Register Select
Register Array
Address Buffer
Data Address Buffer
A8 to A15
AD0 to AD7

ALU :-

The Arithmetic and Logic unit performs the arithmetic and logical operations :

* Addition
* subtraction
* Logical AND
* Logical OR
* Logical EXCLUSIVE OR
* complement (Logical NOT)
* Increment (add 1)
* Decrement (subtract 1)
* Left shift, Rotate left, Rotate right
* clear, etc.

Timing and Control unit :-

The timing and control unit is the section of the cpu.

* It is used to generate timing and control signals which are necessary for the execution of instructions.
* It is used to control data flow between cpu and peripherals (including memory).
* It is used to provide status, control and timing signals which are required for the operation of memory and I/o devices.
* It is used to Control the entire operations of the microprocessor and peripherals connected to it.

Thus we can see that the control unit of the CPU acts as the brain of the computer system.

Registers:-

Registers are used for temporary storage and manipulation of data and instructions by the microprocessor. Data remain in the registers till they are sent to the I/O devices or memory. Intel 8085 Microprocessor has the following registers:

* One 8-bit accumulator (ACC) i.e. register A
* six general purpose registers of 8-bit, these are B, C, D, E, H and L.
* one 16-bit stack pointer, SP
* one 16-bit program counter, PC
* Instruction register.
* Temporary register.

In addition to the above mentioned registers the 8085 microprocessor contains a set of five flip-flops which serve as flags (or status flags) A flag is a flip-flop which indicates some conditions which arises after the execution of an arithmetic or logical instruction.

1. Accumulator (ACC):-

The accumulator is an 8-bit register associated with the ALU. The register 'A' is an accumulator in the 8085. It is used to hold one of the operands of an arithmetic and logical operation.

The final result of an arithmetic or logical

operation is also placed in the accumulator.

2. General-purpose Registers:-

The 8085 Microprocessor contains six 8-bit general purpose registers. They are: B, D, C, E, H and L register.

To hold data of 16-bit a combination of two 8-bit registers can be employed.

The combination of two 8-bit registers is called register pair. The valid register pairs in the 8085 are: D-E, B-C and H-L. The H-L pair is used to act as a memory pointer.

3. Program counter (PC):-

It is a 16-bit special purpose register. It is used to hold the address of memory of the next instruction to be executed. It keeps the track of the instruction in a program while they are being executed.

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

4. Stack pointer (SP):-

It is a 16-bit special function register used as memory pointer. A stack is nothing but a portion of RAM. In the stack, the contents of only those registers are saved, which is being are needed in the later part of the program.

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

the address of the top element of data stored in the stack.

5. Instruction Register :-

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

6. Temporary Register :-

It is an 8-bit register associated with the ALU. It holds data during an arithmetic / logical operation. It is used by the microprocessor it is not accessible to programmer.

7. Flags :

The Intel 8085 microprocessor contains five flip-flops to serve as a status flags. The flip-flops are reset or set according to the conditions which arise during an arithmetic or logical operation.

The five status flags of Intel 8085 are :-

* Carry flag (CS)
* parity flag (P)
* Auxiliary carry flag (AC)
* Zero flag (Z)
* Sign flag (S)

If a flip-flop for a particular flag is set, then it indicates 1. when it is reset, it indicates 0.

Data and Address Bus :-

* The Intel 8085 is an 8-bit microprocessor. Its 'data bus' is 8-bit wide and therefore, 8 bits of data can be transmitted in parallel from or to the Microprocessor..
* The Intel 8085 requires an address bus of which addresses are of 16-bits.

7. 8085 Instruction set

* Instruction sets are instruction codes to perform some task.
* A digital computer understands instructions written in binary codes (machine codes)
* An instruction (instruction format) is a command to the microprocessor to perform a given task on a particular data.
* Each instruction (instruction format) is of two parts.
* one is task to be performed, called operation code (opcode) and the second one is the data to be operated on called the 'operand'.
* The operands or data can be specified in different ways.
* It may include an 8-bit or 16-bit data, an internal register, a memory location, or 8-bit or 16-bit address.

* In some instructions, the operand is implicit.

The 8085 instruction set is of three groups according to word size:

* one-word or one-Byte instructions.
* Two-word or two-Byte instructions.
* three-word or three-Byte instructions.

In the 8085 microprocessor, byte and words are synonymous because it is an 8-bit micro processor. But, instructions are commonly referred to in terms of bytes rather than words.

One-Byte instructions:-

A one-byte instruction includes a opcode and a operand in the same byte. Operand(s) are internal registers and are in the instruction in form of codes. If there is no numerical present in the instruction then that instruction coill be of one-byte, opcode operand.

MOV C,A (copy the content of accumulator in the register C)

ADD B (Add the contents of register B to the contents of the accumulator)

Two-Byte instructions:-

In a two-byte instruction, the first byte specifies the operation code and second byte specifies the operand. Source operand is a data byte and immediately following the opcode. If an 8-bit numeral is present in the instruction

then that instruction will be of two-byte. Here, the numeral may be a data or an address.

opcode	operand
MVI	A 35H (Load an 8-bit data byte in the accumulator.)

Three-Byte instructions :-

In a three-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit operand. The second byte is the low-order operand and the third byte is the high-order operand. If a 16-bit numeral is present in the instruction then that instruction will be of three bytes.

opcode	Operand
JMP	2550H (Transfer the program sequence to memory location 2085h).

Unit-3

Part-A (Design)

1. Instruction codes
2. Computer Registers
3. Computer Instructions
4. Timing & Control
5. Instruction cycle
6. Memory-Reference instructions
7. Input-output and Interrupt.

Part-B (CPU)

1. Introduction
** 2. General Register organisation.
3. Stack organisation.
4. Instruction formats
*** 5. Addressing modes.

Part-A (Design)

1. Instruction Codes

* An instruction code is a group of bits that instruct the computer to perform a specific task.
* A program is a set of instructions
* A computer instruction specifies a sequence of micro operations.
* The computer reads each instruction from the memory and stores a registers.
* Every computer has its own unique instruction set, which has the ability to store and execute instructions.
* An operation code two parts; they are
    1. Operation code (opcode / mnemonic)
    2. Address of data / data

1. Opcode :-

The various operations can be provided by various opcodes, they are

1. ADD
2. SUB
3. MUL

4. SHIFT

5 COMPLEMENT

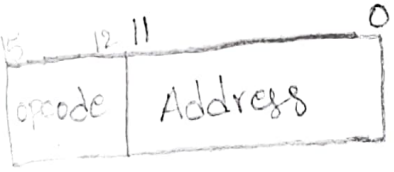
* To perform an operation, a computer needs an Instruction and one processor register.

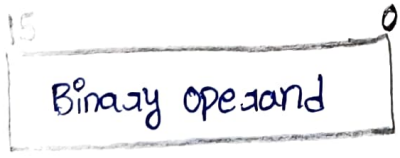
2 Address of Data :-

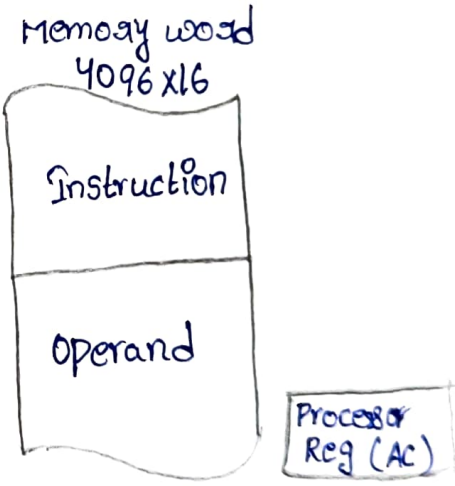
* The address instructs the control where to find operand in the memory, which is used as the data which will be stored in the processor register.

→ The memory unit stores both the instructions and operands in a memory word.

→ A memory word size is 16 bits

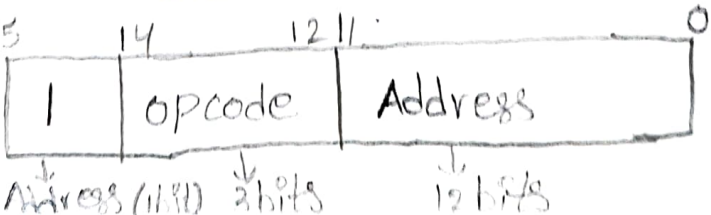






→ The instruction format contains two types of addresses, they are:

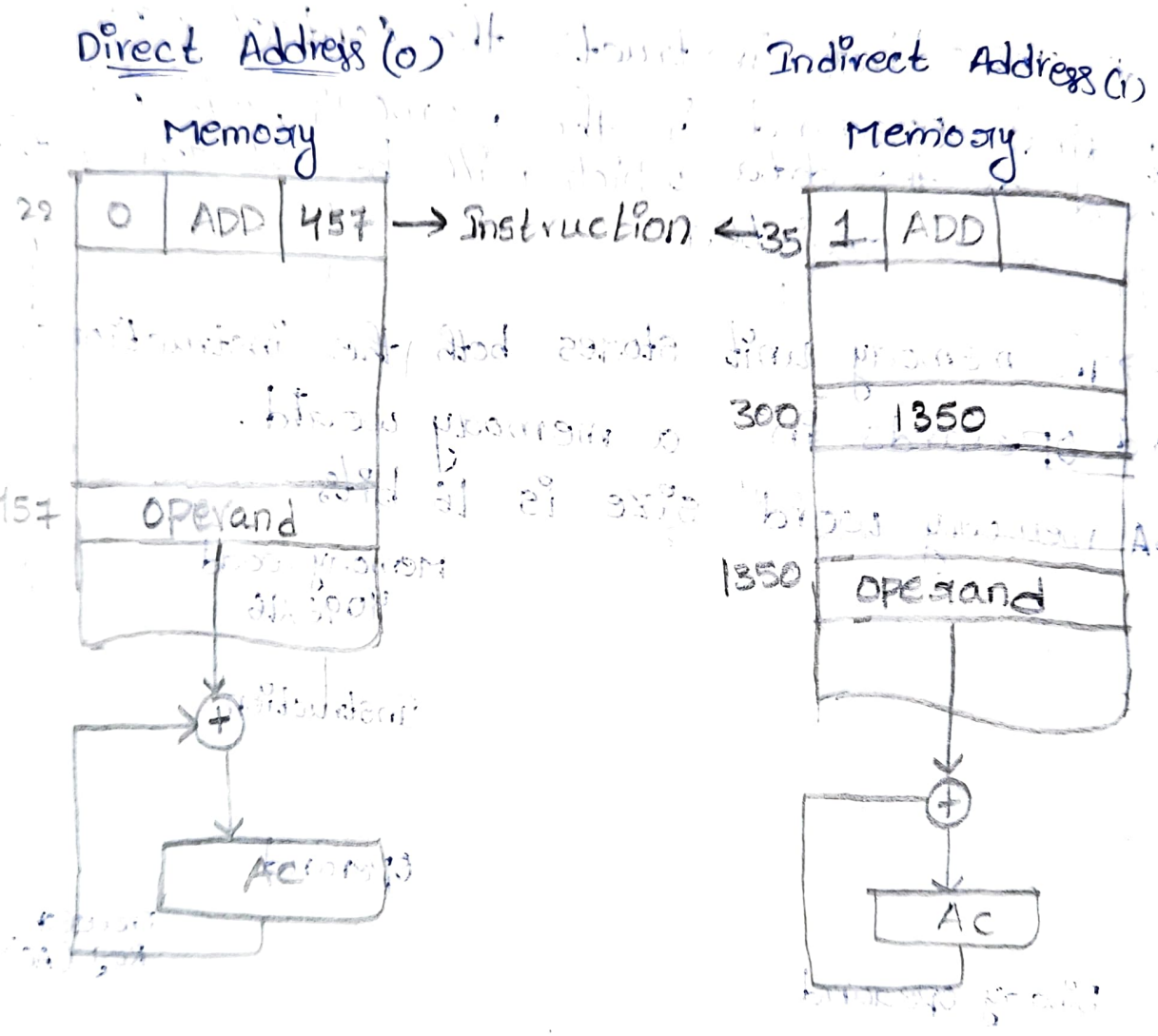
1. Direct Address
2. Indirect Address



(Instruction format)

→ From the above instruction format uses direct (or) Indirect address.

→ A direct address is denoted by '0' and indirect address is denoted by '1'.



2. Computer Registers

* A basic computer consists of several registers, they are

Register Name	Register symbol	No. of bits (size)
Accumulator	AC	16 ≈ 2 Bytes
Address Register	AR	12
Data Register	DR	16
Instruction Register	IR	16
Program Counter	PC	12
Temporary Register	TR	16
Input Register	INPR	8
output register	OUTR	8

Accumulator (AC) :-

→ Accumulator is a processor Register

→ The symbol of Accumulator is 'AC'.

→ The size of Accumulator is 16 bits.

Address Register :-

→ Address Register is used to store the address of memory location.

→

→

Data Register :-

→ Data Register is used to store the operands value.

→ the symbol of data Register is "DR"

→ the size of data Register is "16 bits"

Instruction Register :-

→ Instruction Register stores the instruction which is read from the memory

→ the symbol of d Instruction Register is "IR"

→ the size of instruction Register is "16 bits"

Program counter :-

→ program counter is used to store the address of next instruction to be read from the memory.

→

→

Temporary Register :-

→ Temporary Register is used to store temporary data during the processing of instruction.

→

→

Input Register :-

→ Input Register is used to receive and store the data from input device

→ the symbol of Input register is "INP"

→ The size of input register is "8 bits".

Output Register:-

→ output register is used to send the data to an output device from the memory.

→ The symbol of output Register is "OUTR"

→ The size of output Register is "8 bits"

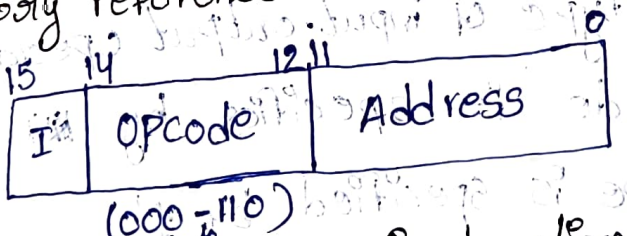
3. Computer Instructions

→ A basic computer uses three types of instructions

→ The basic computer instruction formats are

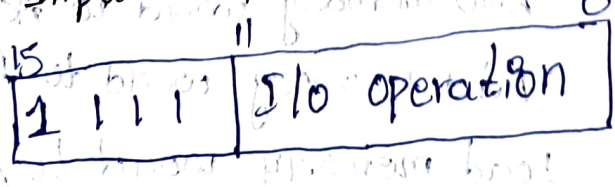
1. Memory-reference instruction
2. Register-reference instruction
3. Input-output instruction.

1. Memory reference instruction:-



2. Register-reference instruction:-

3. Input-output instruction:-



1. Memory reference instruction :-

The memory reference instructions use 12 bits to specify an address, 3 bits to specify the opcode, 1 bit is used to specify the type of address. (0 = Direct & 1 = Indirect).

2. Register reference instruction :-

The Register reference instruction specifies a Register operation, which is 12 bits. Here the opcode is specified by 111 and address is 5) type is '0'.

3. Input - Output Instruction :-

The Input - output instruction uses 12 bits to specify the type of input-output operations Here the opcode is specifies by 111 and address type is specified by '1'.

→ The basic computer instructions which are used in a computer are

Symbol	Description
AND	AND memory word to AC
ADD	Add memory word to AC
LDA	Load memory word to AC
STA	store content of AC in memory

BUN	Branch unconditionally
BSA	Branch and save return address
ISZ	Increment and skip if zero
CLA	Clear AC
CLE	clear E
CMA	complement AC
CME	Complement E
CIR	circulate right AC and E
CIL	Circulate Left Ac and E
INC	Increment AC
SPA	skip next instruction if AC positive
SNA	skip next instruction if AC negative
SZA	skip next instruction if AC zero
SZE	skip next instruction if E is 0
HLT	Halt computer.
INP	Input Character to AC
OUT	Output character from AC
SKI	skip on input flag
SKO	skip on output flag
ION	Interrupt on
IOF	Interrupt off

→ An instruction set is complete, if the computer satisfies below conditions, they are

1. Arithematic, logical and shift instruction.
2. Move information from memory & Registers.
3. Control instructions [program]

4. Input and output instructions

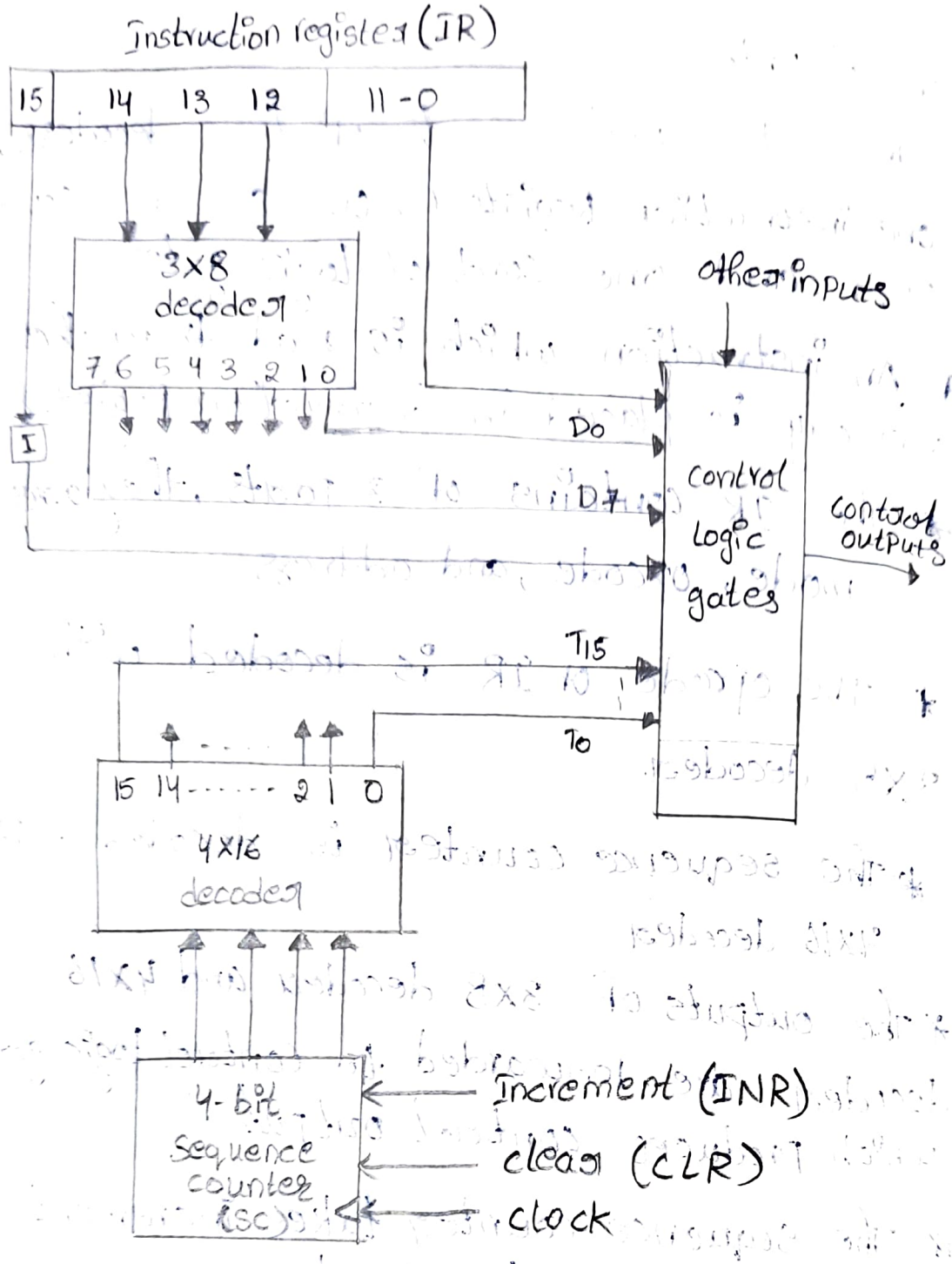
4. Timing and Control

* The timing for all registers in a basic computer is controlled by a master clock.
* The clock pulses are applied to all the flip-flops and registers in the system.
* The clock pulses cannot change the state of a register until the Register is enable.
* The control signals are generated in the control unit which provides inputs to multiplexers
* The control unit can be organised in two ways, they are:
    1. Hardwired control
    2. Microprogrammed control
* In hardwired organization, the control logical is implemented by using gates, flip-flops, decoders or any other circuits.
* In microprogrammed organization, the control information is stored in control memory, which is programmed to initiate the required

instructions.

Control unit :-

* The control unit consists of two decoders, one instruction Register, one sequence counter (SC), and one control logic gates
* An instruction which is read from the memory is placed in instruction Register (IR)
* The IR contains of 3 parts, they are mode, opcode, and address
* The opcode of IR is decoded with 3X8 decoder.
* The sequence counter is decoded with 4x16 decoder
* The outputs of 3x8 decoder and 4x16 decoder are forwarded to control logic gates, which produces control output.
* The sequence counter takes increment, clear, and clock has input



Control unit of basic computer

5. Instruction cycle

* A program in the memory unit of a computer contains a sequence of instructions.
* A program is executed in the computer through a cyclic process for each instruction.
* Each instruction cycle can be divided into 4 phases, they are :

1. Fetch an instruction from the Memory.
2. Decode the instruction:
3. Read the effective address from the memory
* [Eg:- Indirect Address]
4. Execute the instruction

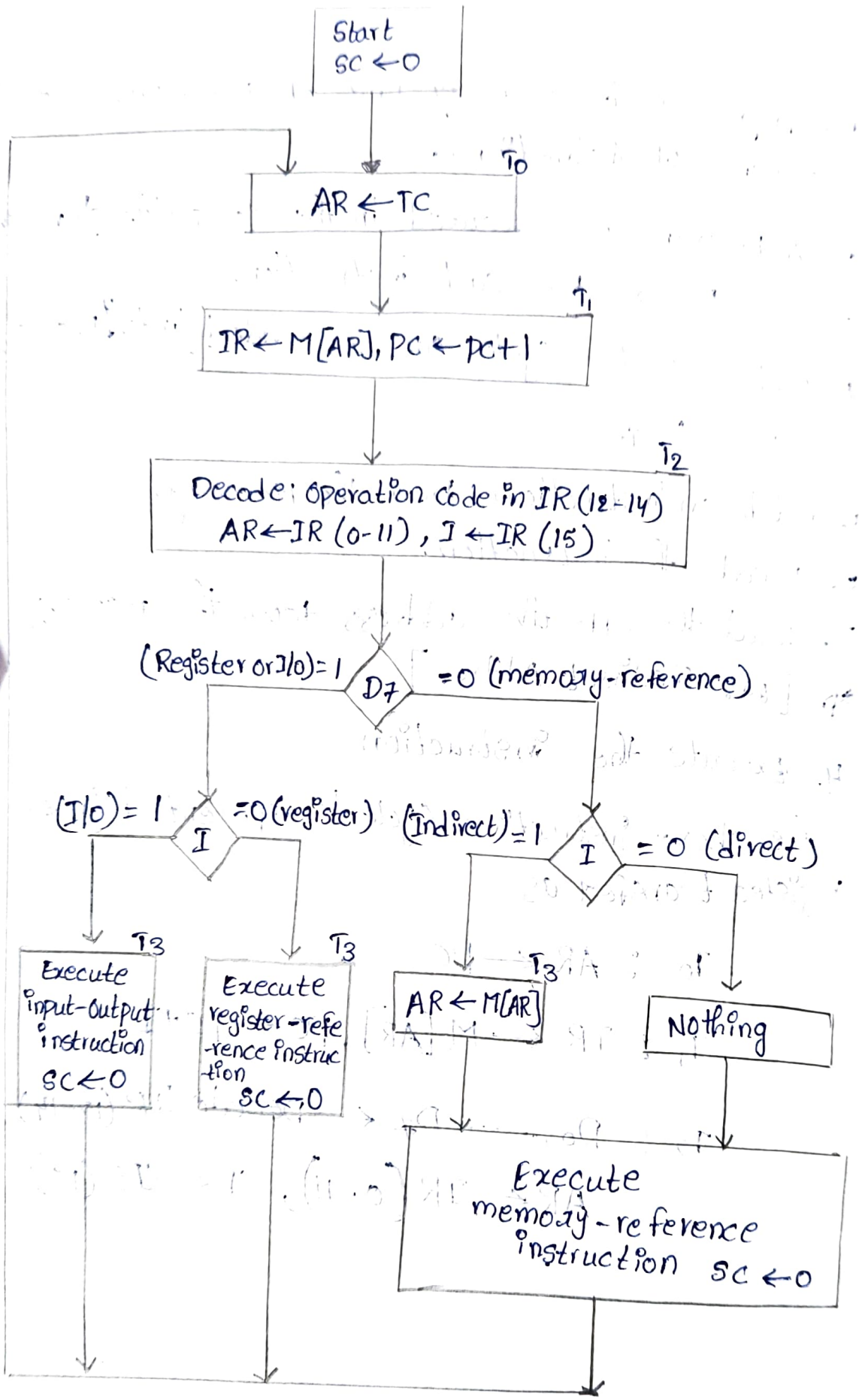
The above instruction cycle is represented in the register transfer as

T_0 : $AR \leftarrow PC$

T_1 : $IR \leftarrow M[AR], PC \leftarrow PC+1$

T_2 : $D_0, \ldots D_7 \leftarrow$ Decode $IR(12\text{-}14)$,
$AR \leftarrow IR(0\text{-}11), I \leftarrow IR(15)$

Instruction cycle Flowchart



Register Reference Instruction :-

* The execution of Register reference instruction is done when $D_7 = 1$ and $I = 0$
* The register reference instruction gets executed when decode operation code in IR (12-14).
  $AR \leftarrow IR(0,11)$, $I \leftarrow IR(15)$
* The list of micro operations needed to execute memory register reference instruction are:

	r: $SC \leftarrow 0$	clear SC
CLA	rB_{11} : $AC \leftarrow 0$	clear AC
CLE	rB_{10} : $E \leftarrow 0$	clear E
CMA	rB_9 : $AC \leftarrow \overline{AC}$	complement AC
CME	rB_8 : $E \leftarrow \bar{E}$	complement E
CIR	rB_7 : $AC \leftarrow shr\ AC$, $AC(15) \leftarrow E$, $E \leftarrow AC(0)$.	circulate right
CIL	rB_6 : $AC \leftarrow shL\ AC$, $AC(0) \leftarrow E$, $E \leftarrow AC(15)$	circulate left
INC	rB_5 : $AC \leftarrow AC + 1$	Increment AC.
SPA	rB_4 : If $(AC(15) = 0)$ then $(PC \leftarrow PC + 1)$	skip if positive
SNA	rB_3 : If $(AC(15) = 1)$ then $(PC \leftarrow PC + 1)$	skip if negative

SZA rB_2: If (AC=0) then (pc ← pc+1) — Skip if AC zero

SZE rB_1: If (E=0) then (pc ← pc+1) — Skip if E zero

HLT rB_0: S ← 0 (S is a start-stop flip-flop) — Halt computer