6/1/25

COA

i) Digital circuits :
ii) Microprocessor 8086
iii) C.O.A.

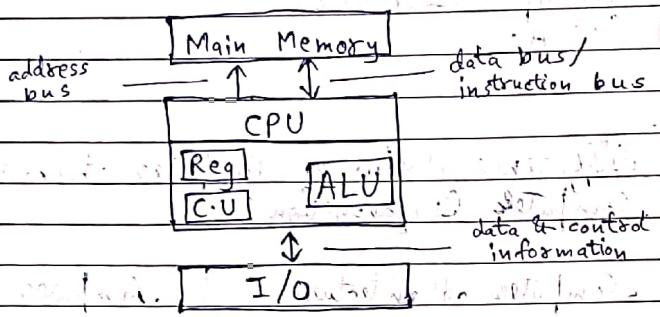
9/1/25

Q. Difference between Computer Architecture & Computer Organization.

Computer Architecture	Computer Organization
→ Explains what a computer does.	→ Explains how computer actually does.
→ Focuses on functional behavior.	→ Focuses on structural relationship & internal behaviour.
→ Architecture deals with high level design.	→ Organization deals with low level design.
→ It comes before computer organization.	→ It comes after computer architecture.
→ It is also called as ISA (Instruction Set Architecture).	→ CO is micro architecture
→ Covers logical function such as register, data type, instruction set & addressing modes.	→ It covers physical unit like peripherals, circuit design & adder.

9/1/25

* Von Neumann Model (Stored Memory architecture/ Stored Memory Program):



Von Neumann Model works on stored program concept.
Execution of instructions in Von Neumann machine is carried out in a sequential fashion from one instruction to the next.

13/1/25

* Performance Measure of computer architecture
  i) Better Hardware
  ii) Innovative Architecture
  iii) Efficient Resource management
  iv)
  i) Better Algorithm
  ii) Data Structure
  iii) Language efficiency

1) Sequential execution time (T_1):
Time required for a program to be executed on a sequential system. (Uni-processor)
2) Parallel execution time [$T(n)$]:
Time required for a program to be executed on a n-parallel processor system.

13/1/25

3) Speed up:
Ratio of time required to execute a program on a sequential system to that of parallel system.
$S(n) = T(1)/T(n)$
4) Efficiency [η or $E(n)$]:
$\eta = \frac{\text{Actual speed up}}{\text{Ideal speed up}} = \frac{T(1)}{nT(n)}$
5) Clock per instruction (CPI):
→ ~~Clock~~ a measure of clock pulses required per instruction. ~~or~~
→ It is a ratio of clock cycles required for a program to the number of instructions in the program.
Time required to execute a program (T) can be given by:
$T = I_c \times C.P.I. \times \tau$
where τ = time for one clock pulse.
I_c = number of instructions in the program.
6) MIPS (Million Instruction per second):
→ Count of instruction executed per second is in millions.
→ Thus, number of instructions executed per second:
~~Into~~ Instruction per second $= \frac{1}{CPI \times \tau}$

$MIPS = \frac{1}{CPI \times \tau \times 10^6}$

$\tau = \frac{1}{f}$ $\quad MIPS = \frac{f}{CPI \times 10^6}$

13/1/25

7) Million floating point instructions per second (MFLOPS):
→ It is similar to MIPS only difference being is floating-point instructions are taken into account.

8) Throughput (W_s)
Not Number of program executed per unit time.
$W_s = \frac{\text{Number of programs}}{\text{Time in seconds}}$

9) Scalability
A parallel system is said to be scalable if the same efficiency is obtained by increasing the number of processors.

* Amdahl's law:
→ This law points out some limitations of parallel processing. Programs contain ~~some~~ certain computations that are inherently sequential & thus cannot be speeded up through parallel processing.
→ If f represents the fraction of a program running time due to such unparallelizable computation, even assuming that the remainder of the program enjoys the perfect speed up of p, when run on a 'p' processors. The overall speed up would be:

$$S = \frac{1}{f + (1-f)/p} \leq \min(p, 1/f)$$

num- 1 is original running time of the program
den- improved execution with p parallel processor.

↑ sum of running time for unparallelizable block which runs p times as fast

16/1/25

Evolution of Computers:

1680 ~~s~~ 1822 Transistor

Generation	I	Vaccum tube 1945-1954
Generation	II	Semiconductor (transistor)
Generation	III	IC (Integrated circuit)
Generation	IV	Microprocessor

16/1/25

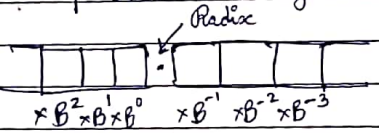
Number System

Base / Radix: The number of values that a digit can assume is equal to the base of a system.

→ Ex: In decimal system, base is 10. Hence every digit can assume 10 values [0..9]

→ Largest value is always one less than base.

→ Numbers have positional weights:



→ Various numbers system:

Base	Name	Digits
2	Binary	0, 1
8	Octal	0, 1, 2, 3, 4, 5, 6, 7
10	Decimal	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
12	Duo-binary	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B
16	Hexadecimal	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

16/1/25

Decimal	Binary	Octal	Duo-binary	Hexadecimal
0	0000	00	0	0
1	0001	01	1	1
2	0010	02	2	2
3	0011	03	3	3
4	0100	04	4	4
5	0101	05	5	5
6	0110	06	6	6
7	0111	07	7	7
8	1000	$10	8	8
9	1001	11	9	9
10	1010	12	A	A
11	1011	13	B	B
12	1100	14	10	C
13	1101	15	11	D
14	1110	16	12	E
15	1111	17	13	F

MSB → bit to the leftmost side
LSB → bit to the rightmost side

Name	Size (bits)	eg
bit	1	1
Nibble	4	0010
Byte	8	0010 1011
word	16	1010 1011 0100 1000
Double word	32	1101 0110 1111 1011 1010 0001 1011 0001

16/1/25

* Conversion of number systems:

i) From any base to decimal:

a) Binary to decimal: (Streamlined method)

1011.01_2

$2^3\times1+2^2\times0+2^1\times1+2^0\times1\cdot0\times2^{-1}+1\times2^{-1}$

$8+0+2+1\cdot0+0.25$

$= 11.25_{10}$

b) Octal to decimal

365.24_8

$3\times8^2+6\times8^1+5\times8^0\cdot2\times8^{-1}+4\times8^{-2}$

245.3125_{10}

c)

$4C8.2$

$4\times16^2+12\times16^1+8\times16^0\cdot2\times16^{-1}$

1224.125_{16}

ii) From decimal to any base:

a) Decimal to binary: (Babble method)

2	105	
2	52	1
2	26	0
2	13	0
2	6	1
2	3	0
	1	1

$105_{10} = 1101001_2$

16/1/25

b) Decimal to Octal:

8	204	
8	25	4
	3	1
		3

$(204)_{10} = 314_8$

2	210	2
2	26	2
	3	3

c) Decimal to Hexadecimal:

259

16	259	3
	16	0
	1	1

$259_{10} = 103_{HH}$

17/1/25

i) $(105.42)_{10} = (\quad)_2$

$0.42\times2 = 0.84$

$0.84\times2 = 1.68$

$0.68\times2 = 1.36$

$0.36\times2 = 0.72$

ii) $(204.6234)_{10} = (\quad)_8$

$0.6234\times8 = 4.9872$

$0.9872\times8 = 7.8976$

$0.8976\times8 = 7.1808$

$0.1808\times8 = 1.4464$

iii) $(259.122)_{10} = (\quad)_{16}$

$0.122\times16 = 1.952$

$0.952\times16 = F.232$

17/1/25

i) Binary to decimal :

ii) Binary to octal :

$(110100010)_2 = (\quad)_8$

$(128+64+16+2)+2 = (212)_8$

0610 0100 010 = 322

iii) Binary to Hexadecimal :

1010 1111 1011 0010

1010 1111 1011 0010

A F B 2 $= (AFB2)_H$

iv) Octal to Binary :

$(364)_8 = (011\ 110\ 100)_2$

$(298)_8$

∵ given number is not octal

v) Octal to Hexadecimal :

$(436)_8 = (11E)_{16}$

8	3567	7
8	445	5
8	55	7
	6	6

6757

vi) Hex to Binary :

$(ABC)_{16} = (\quad)_2$

1010 1011 1100

$(DEF)_{16} =$ 1101 1110 1111

$(6757)_8$

17/1/25

* Binary addition :

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

10111
11001
110000
↑
CY

* Binary subtraction :

A	B	Difference	Borrow
0	0	0	0
1	0	1	0
1	1	0	0
0	1	1	1

011
101
010 2

3 - 5 = ?

* Unsigned Binary numbers :

All data is +ve

Range $- (0-255)_{10}$ & $(00-FF)_{16}$

* Signed Magnitude number :

If data is +ve or -ve, then we use signed magnitude number.

MSB of ~~signed~~ binary no. is used to represent sign & remaining bits are used to represent magnitude.

Range becomes : -127 to 127

20/1/25

* 1's complement:

→ Replace every 0's with 1's & 1's with 0's to get 1's complement.

Ex: 1010 $\xrightarrow{\text{1's comp}}$ 0101

11010111 ⟶ 00101000

* 2's complement:

→ add 1 to 1's complement.

1010 ⟶ 0101 + 1 = 0110

1011 ⟶ 0100 + 1 = 0101

1011 0010 ⟶ 0100 1101 + 1 = 0100 1110

* Signed Binary number:

Decimal no.	Sign magnitude	2's complement	1's complement
7	0 111	1 0001	
6	0 110	1 0010	
5	0 101	1 0011	
4	0 100	1 0110	
3	0 011	1 101	
2	0 010	1 1010	
1	0 001	1 110	
0	0 000	1 111	
-0	1 000	0 111	
-1	1 001	0 110	
-2	1 010	0 101	
-3	1 011	0 100	
-4	1 100	0 011	
-5	1 101	0 010	
-6	1 110	0 001	
-7	1 111	0 000	

20/1/25

* From 2's complement method, ambiguity associated with +0 & -0 is eliminated. We can represent -8 also using 2's complement.

Q. Represent +40 & -40 using 2's complement

$(40)_{10} = 101000 \Rightarrow 0010\ 1000$

$(-40)_{10} = (1101\ 1000)_2$

2 | 40 0
2 | 20 0
2 | 10 0
2 | 5 1
2 | 2 0
1

Q. Perform following using 1's complement

9 → 1001 → 1001

−4 → +1100 → +1011

10101 ⇒ +5 → 10100 → 0101

* Subtraction using 1's complement:

→ Take complement of subtrahend / represent -ve no. as complement of 1's.

→ In the result add the MSB to LSB to get the answers.

21/1/25

Q. Add two numbers using signed complement form.

A = +7 & B = +12

0000 0111

0000 1100

0001 0011 → +19

A = -7, B = +12

1 0000 0111 → 1111 1001

0000 1100 → 0000 1100

1 0001 0011 → 0000 0101 → 5

2/1/25

A = +7, B = -12

0000 0111 → 1111 0011

1111 0100 (+1) → 1111 0101

2's complement

-5 → 0000 101

Q. Perform (9-4) using 1's complement.

9 → 1001

-4 → 1011

1 0100 → 0101 → +5

(+1)

Q. Subtract 9 from 4 using 1's complement

0110

0100

0 1010 → 1's comp → 0101 → -5

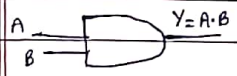
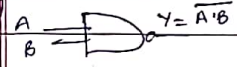
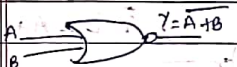
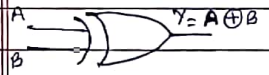
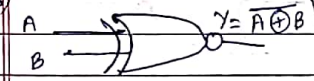
-ve

*	Combinational circuit	Sequential circuit
→	Output depends only on present input	Output depends on previous or past input
→	Speed is fast	Speed is slow
→	Easy to design	Difficult to design
→	No feedback	There is feedback
→	Time independent	Time dependent
→	Basic building block: logic gates	Basic building block: Flip flop
→	Clock independent, hence no triggering required	Clock dependent, & requires triggering
→	Ex: encoder, decoder, MUX, DEMUX, gates	Ex: Flip flops, counters, shift registers.

2/1/25

7486

(IC symbol)

	Name of gate	Boolean expression	Logic operation	Truth table
(IC-7404) HEX INVERTOR	1 NOT 	$Y = \bar{A}$	Inverter	A | Y=Ā 0 | 1 1 | 0
(IC-7432)	2 OR 	$Y = A+B$	Logical addition	A B | Y=A+B 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
(IC-7408)	3 AND 	$Y = A \cdot B$	Logical multiplication	A B | Y=A·B 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
(IC-7400)	4 NAND 	$Y = \overline{A \cdot B}$		A B | Y=Ā·B̄ 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
(IC-7402)	5 NOR 	$Y = \overline{A+B}$		A B | Y=Ā+B̄ 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
(IC-7486)	6 XOR 	$Y = A \oplus B$ $Y = \bar{A} \cdot B + A \cdot \bar{B}$		A B | Y=A⊕B 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
(IC-7486)	7 XNOR 	$Y = \overline{A \oplus B}$ $Y = \overline{(\bar{A} \cdot B + A \cdot \bar{B})}$		A B | Y=Ā⊕B (or A⊙B) 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

23/1/25

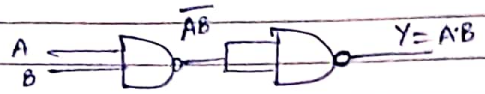
De Morgan's theorem:

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

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

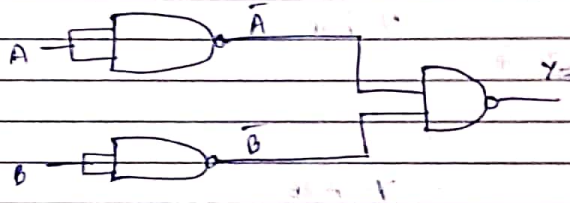
$\overline{\overline{A}} = A$

* AND gate using NAND gate:

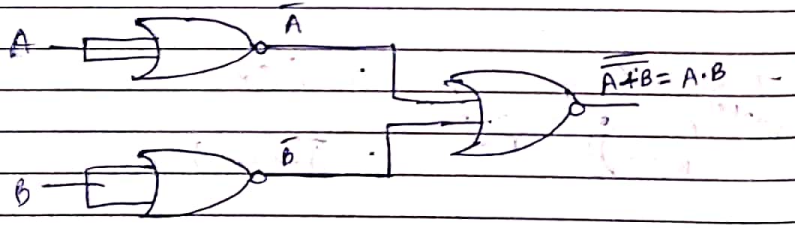


* NOT gate using NAND gate:

* OR gate using NAND gate:

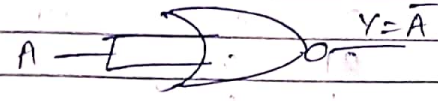


* AND using NOR gate:

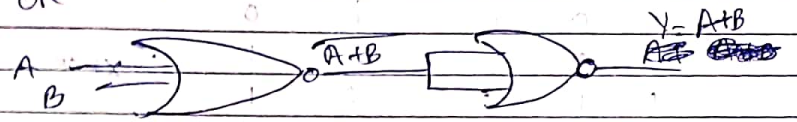


23/1/25

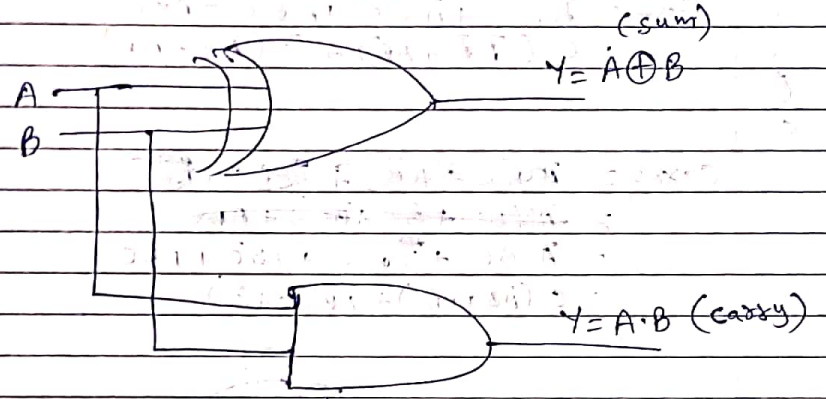
* NOT using NOR:



* OR using NOR:

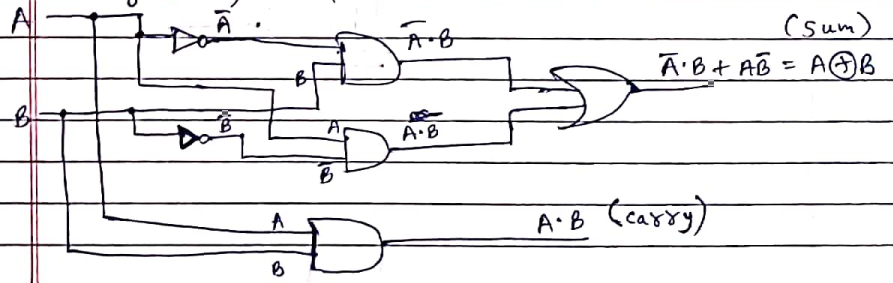


* Half adder:



Using AND & XOR ↑

Using NOT, AND & OR:



23/1/25

Full adder:

A	B	C	Sum	carry		
0	0	0	0	0	XOR	AND
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	XNOR	OR
1	1	0	0	1		
1	1	1	1	1		

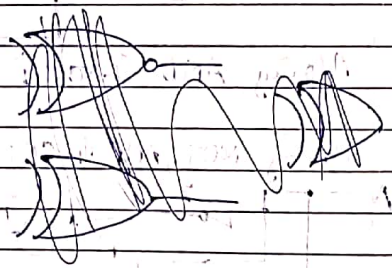
$Sum = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

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

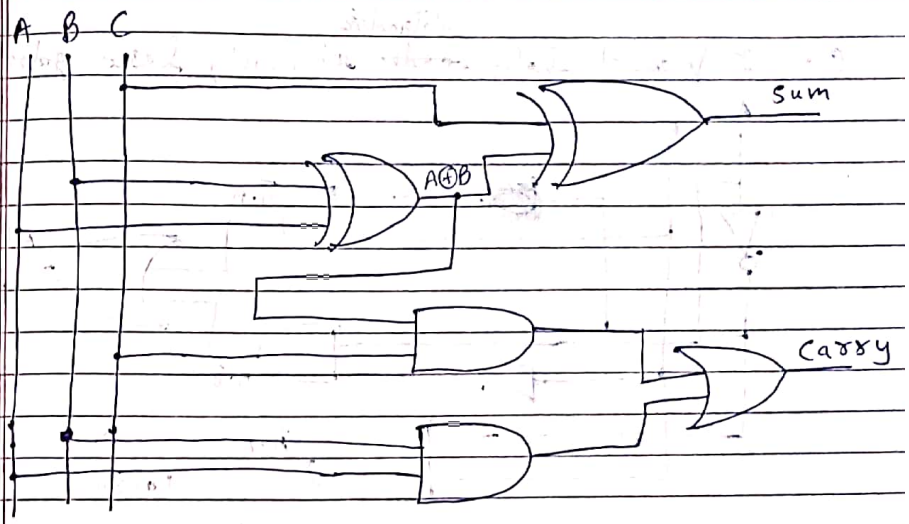
$Carry = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

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

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



24/1/25

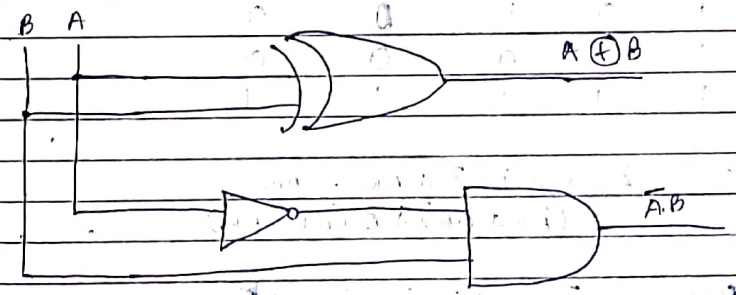


Half Subtractor:

A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	10	0

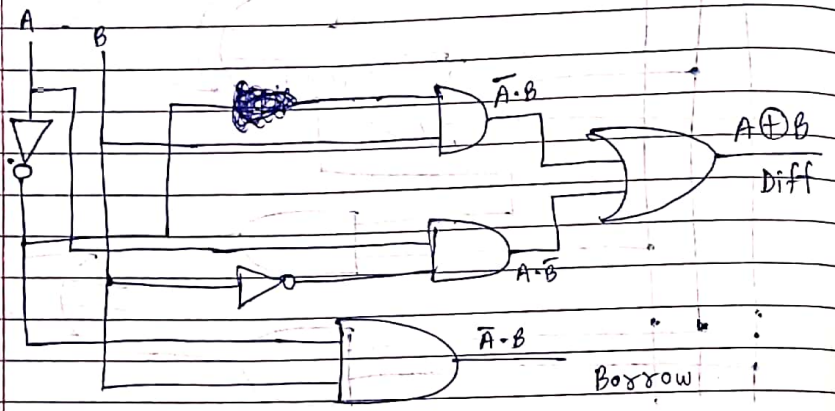
$Diff = \bar{A}B + A\bar{B} = A \oplus B$

$Borrow = \bar{A}B$



24/11/25

Q. Implement half ~~adder~~ subtractor using only basic gates



* Full Subtractor:

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

~~Diff = $\bar{A}\bar{B}\bar{C}$~~

$Diff = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

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

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

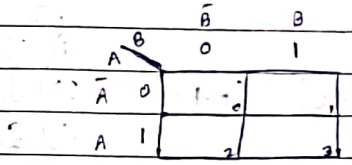
$= BC(\bar{A}+A) + \bar{A}(\bar{B}C + B\bar{C})$

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

24/11/25

* 2-variable k-map:

→



Note: minterm (m) — S.O.P. (sum of products) Σm

Note: maxterm (M) — P.O.S. (products of sum) ΠM

* 3-variable k-map:

→

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

* 4-variable k-map:

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

30/1/25

* Karnough map (K-map):

Gray code:
i) Single
ii) Raid
iii) Quad
iv) Oct

$Y = ABC + AC + BC$

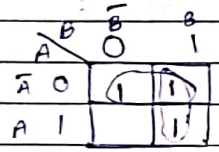
$Y = (A+B+C)(A+C)(B+C)$

SOP	POS
m	M
Σ	π
0 is bar	1 is bar

Q. $Y = F(A,B) = \bar{A}\bar{B} + AB + \bar{A}B$

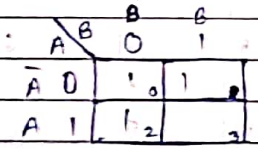
$= \bar{A}(B+\bar{B}) + AB$

$= \bar{A} + AB$



$Y = \bar{A} + B$

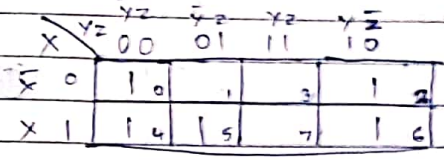
Q. $Y = F(A,B) = \Sigma m(0,1,2)$



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

30/1/25

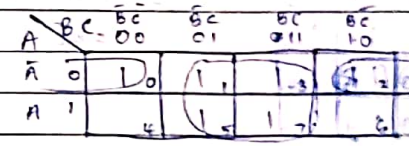
Q. $Y = F(X,Y,Z) = \bar{X}\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + X\bar{Y}Z + XYZ$



$Y = \bar{Y}\bar{Z} + X\bar{Y} + Y\bar{Z} =$

$= \bar{Z}(Y+\bar{Y}) + X\bar{Y} = \bar{Z} + X\bar{Y}$

Q. $Y = F(A,B,C) = \Sigma m(0,1,2,3,5,7)$



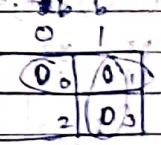
$Y = \bar{A} + \bar{A}\bar{C}$

$Y = \bar{A}$

$Y = \bar{A} + C$

Q. ~~$Y = F(A,B)$~~ $Y = f(a,b) = \pi M(0,1,3)$

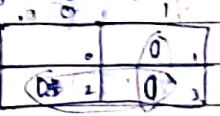
$(a+b)(a+\bar{b})(\bar{a}+\bar{b})$



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

Q. $Y = f(a,b) = (\bar{a})(\bar{b}) = (\bar{a}+\bar{b})$

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



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

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

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

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

30/1/25

Q: $Y = f(p,q,r) = \pi M(0,2,4)$

p \ qr	00	01	11	10
p	0	1	3	2
$\bar{p}$	4	5	7	6

$Y = (q+r)(p+r)$

31/1/es Q. $Y = F(a,b,c,d) = \Sigma m(1,5,6,7,11,12,13,15)$

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	1	1	1
ab 11	1	1	1	14
$a\bar{b}$ 10	8	9	1	10

$Y = acd + \bar{a}bc + \bar{a}\bar{c}d + ab\bar{c}$

$\therefore Y = acd + \bar{a}bc + \bar{a}\bar{c}d + ab\bar{c}$

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

Q. $A = f(w,x,y,z) = \pi M(1,3,4,6,9,11,12,14)$

wx \ yz	$\bar{y}\bar{z}$ 00	$\bar{y}z$ 01	yz 11	$y\bar{z}$ 10
$\bar{w}\bar{x}$ 00		0	0	2
$\bar{w}x$ 01	0	5	7	0
wx 11	0	13	15	0
$w\bar{x}$ 10	8	0	0	

$A = Y = (\bar{x}+z)(x+\bar{z})$

31/1/25

Q. Minimize following expression using K-map & realize it using basic gates:

$Y = \Sigma m(1,2,9,10,11,14,15)$

Let $f(a,b,c,d) = Y$

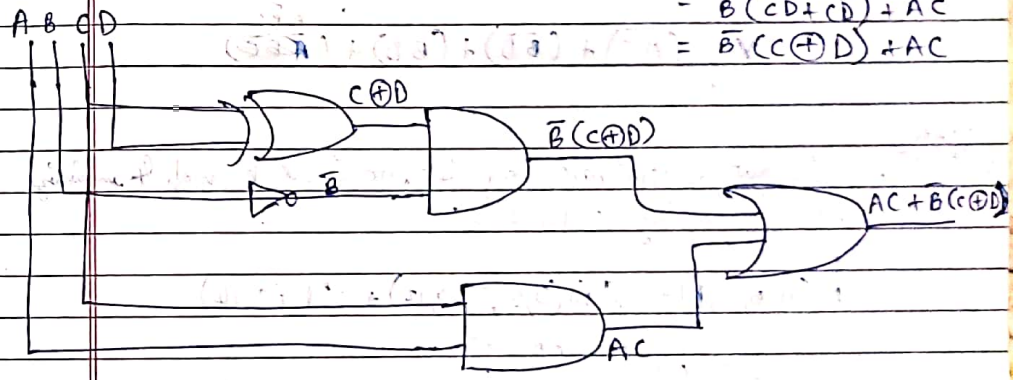
ab \ cd	$\bar{c}\bar{d}$ 00	$\bar{c}d$ 01	cd 11	$c\bar{d}$ 10
$\bar{a}\bar{b}$ 00	0	1	3	2
$\bar{a}b$ 01	4	5	7	6
ab 11	12	13	15	14
$a\bar{b}$ 10	8	9	11	10

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

$= (\bar{B}\bar{C}D) + (AC) + (B\bar{C}\bar{D})$

$= \bar{B}(\bar{C}D + C\bar{D}) + AC$

$= \bar{B}(C \oplus D) + AC$



3/1/25

Q. Reduce given boolean expression using K-map method:

$f(A,B,C,D) = \Sigma m(0,1,2,4,5,7,8,10,11,13,15)$

AB \ CD	$\bar{C}\bar{D}$ 00	$\bar{C}D$ 01	CD 11	$C\bar{D}$ 10
$\bar{A}\bar{B}$ 00	1 0	1 1	3	1 2
$\bar{A}B$ 01	1 4	1 5	1 7	6
AB 11	12	1 13	1 15	14
$A\bar{B}$ 10	1 8	9	11	1 10

~~$Y = (A\bar{C}) + (B\bar{D}) + (BD) + (A\bar{B}C)$~~

$Y = (\bar{A}\bar{C}) + (\bar{B}\bar{D}) + (BD) + (A\bar{B}C)$

=

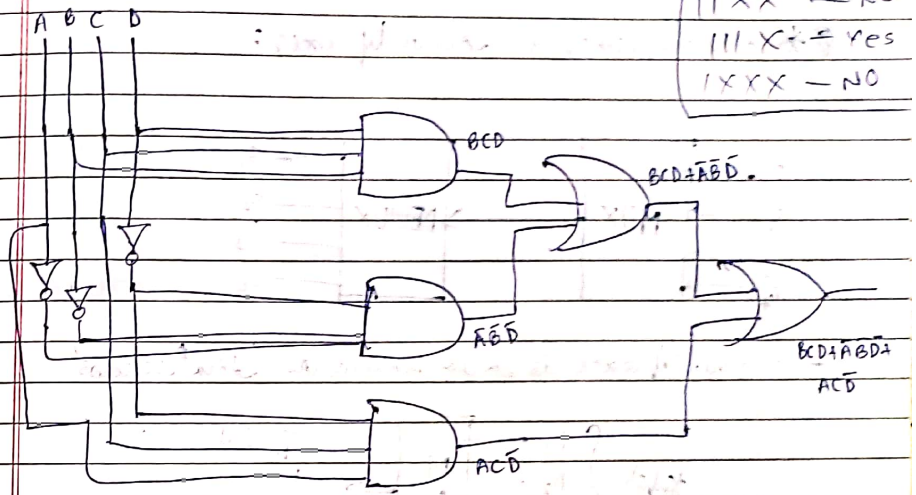
4/2/25

Q. Draw structure of 4-variable k-map & minimize the following expression.

$F(A,B,C,D) = \Sigma m(0,2,7,10,15) + \Sigma d(3,14)$

AB \ CD	$\bar{C}\bar{D}$ 00	$\bar{C}D$ 01	CD 11	$C\bar{D}$ 10
$\bar{A}\bar{B}$ 00	1 0	1	X 3	1 2
$\bar{A}B$ 01	4	5	1 7	6
AB 11	12	13	1 15	X 14
$A\bar{B}$ 10	8	9	11	1 10

$Y = \bar{A}\bar{B}\bar{D} + BCD + AC\bar{D}$



1 1 X X — NO
1 1 1 X — Yes
1 X X X — NO

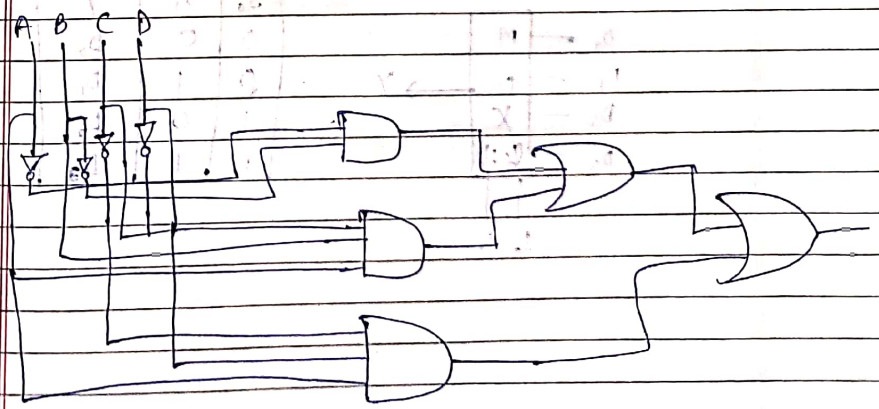
Q. $f(a,b,c,d) = \Sigma m(0,1,2,8,14,15) + \Sigma d(3,4,12,13)$

ab \ cd	$\bar{c}\bar{d}$	$\bar{c}d$	cd	$c\bar{d}$
$\bar{a}\bar{b}$	1	1	X	1
$\bar{a}b$	X			
ab	X	X	1	1
$a\bar{b}$	1			

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

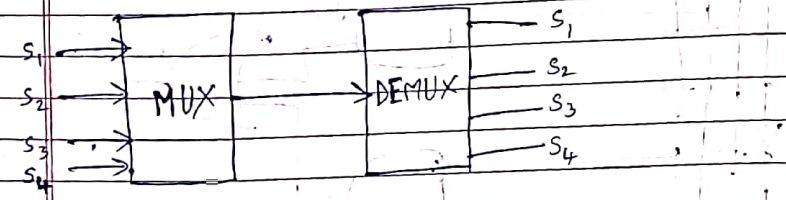
$= \bar{a}\bar{b}(\bar{c}+c) + abc + a\bar{c}\bar{d}$

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

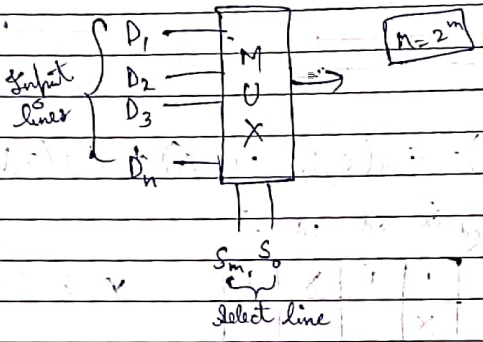


6/2/25

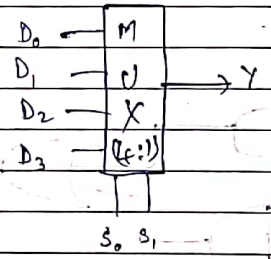
* Multiplexer & demultiplexer:



→ Multiplexer is also called as 'data selector'



4:1 MUX –



S_0	S_1	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

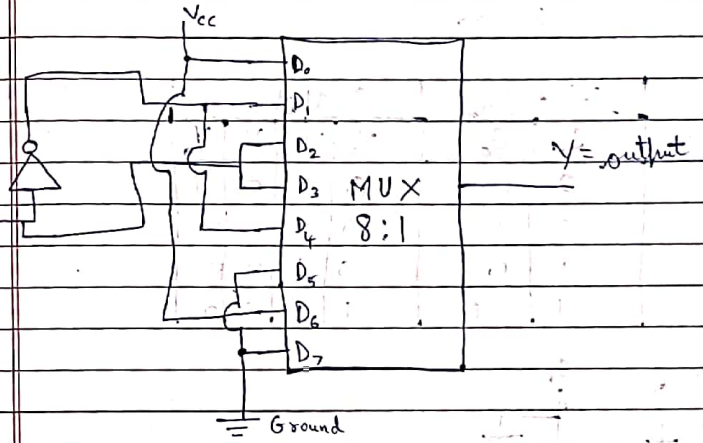
7/2/25

Q. Design & implement following expression using single 8:1 MUX.

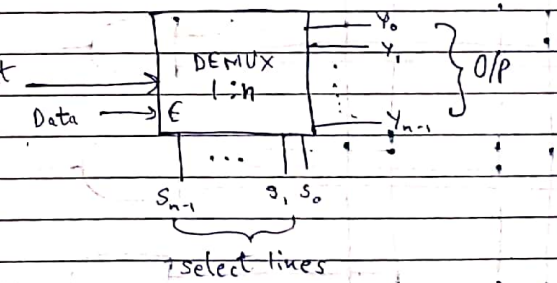
$Y(A,B,C,D) = \Sigma m(0,2,3,6,8,9,12,14)$

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

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
A	(0)	1	(2)	(3)	4	5	(6)	7
$\bar{A}$	(8)	(9)	10	11	(12)	13	(14)	15
	1	$\bar{A}$	A	A	$\bar{A}$	0	1	0



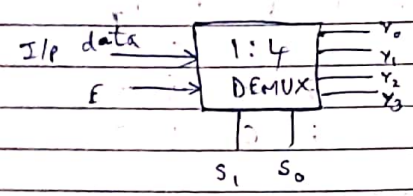
* DEMUX:



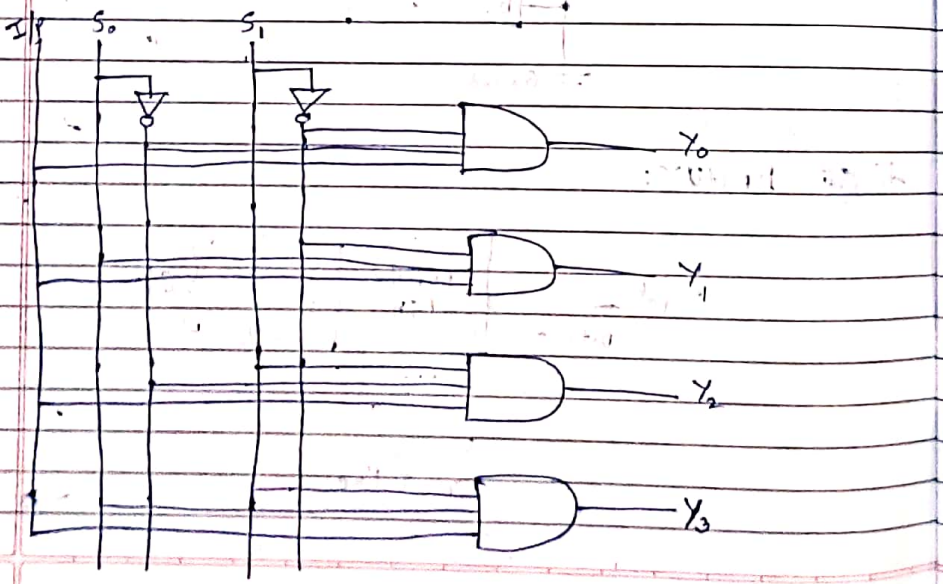
7/2/25

→ It receives data or information in a single line & transmit this information to one of the 2^n possible output lines.

1:4 DEMUX:



I/p	Select line		Output			
	S_1	S_0	Y_3	Y_2	Y_1	Y_0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0



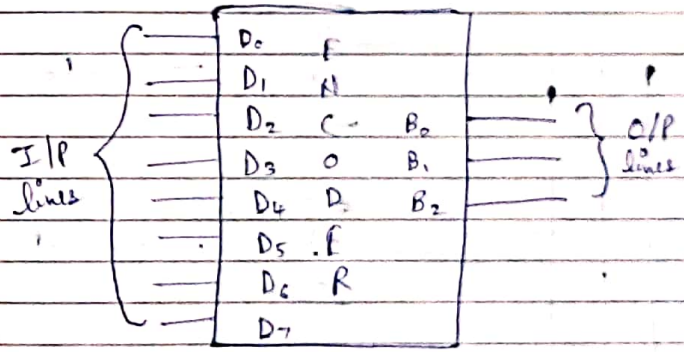
7/2/25.

Encoder:

Encoder has n number of I/P lines & m no. of O/P lines. ($n = 2^m$)
If $n = 8$ then $m = 3$.

Types of encoders:
i) Decimal to BCD
ii) Octal to binary
iii) Priority encoder

Octal to binary encoder:
→ Octal means 8 no. of inputs & 3 output lines.



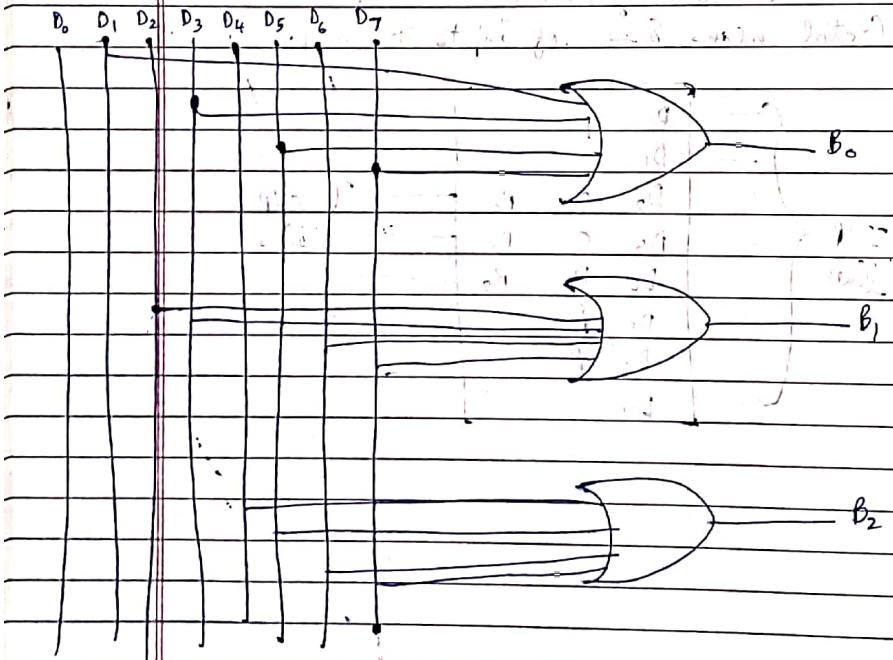
7/2/25

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	B_2	B_1	B_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$B_0 = D_1 + D_3 + D_5 + D_7$

$B_1 = D_2 + D_3 + D_6 + D_7$

$B_2 = D_4 + D_5 + D_6 + D_7$



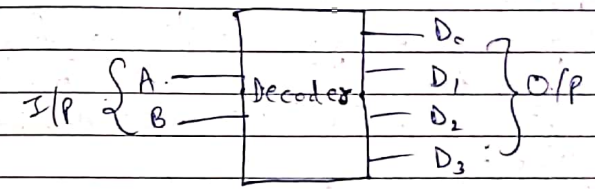
7/2/25

* Decoder:

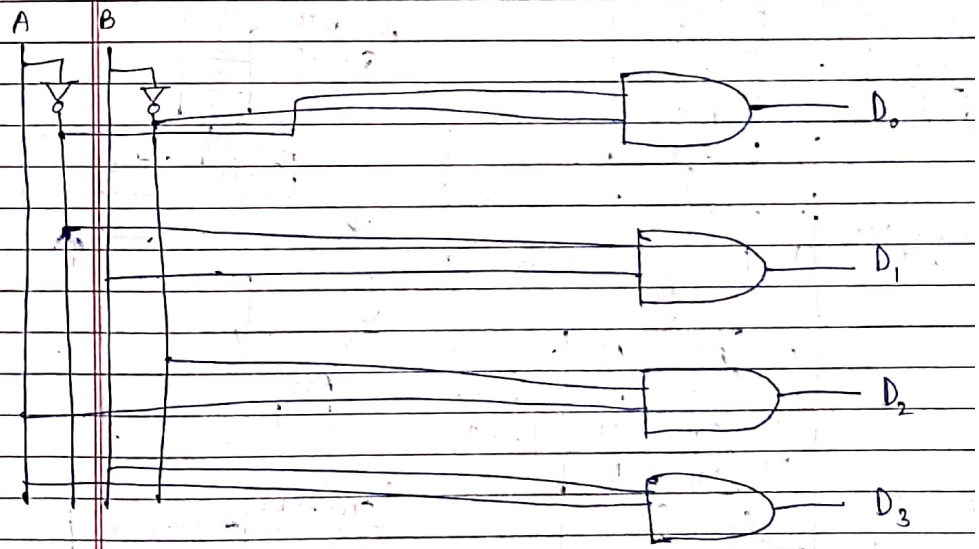
→ It is a combinational circuit.

→ It is identical to DEMUX without any data input.

→ Ex: 2:4 line decoder.



I/P		O/P			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



13/2/25

Features of 8086:

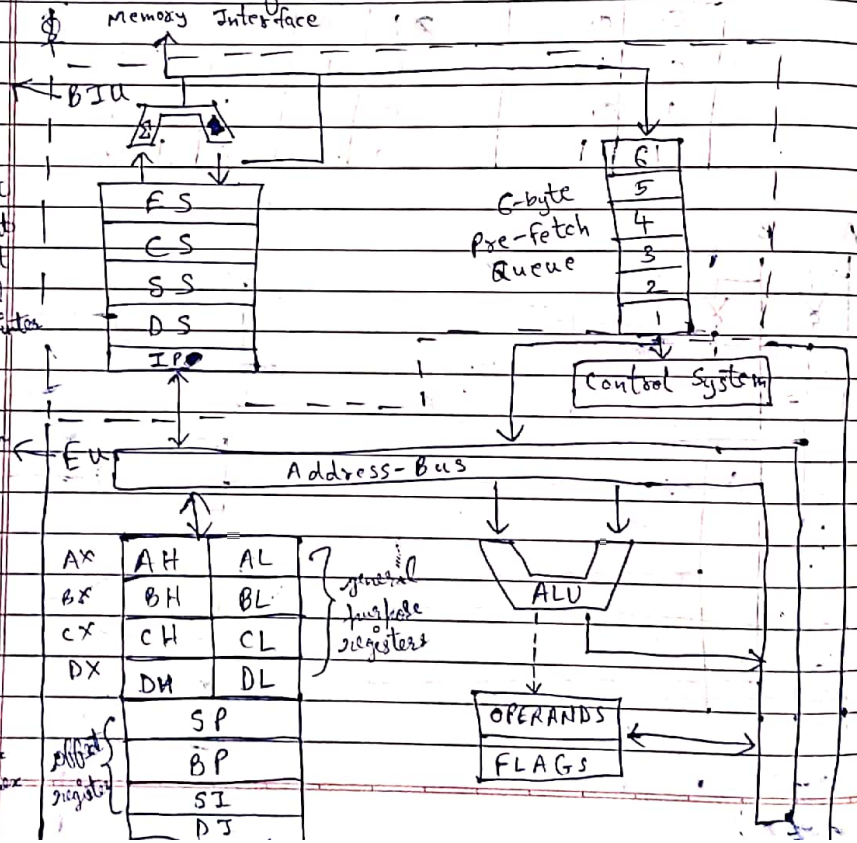
→ 16-bit processor (operates on 16-bit data)

→ Has 20-bit address line; can access $2^{20} = 1MB$ locations.

→ It is pipelined processor.
(Pipelining is fetching one instruction from memory while executing this instruction side by side, it will fetch next instruction thus saving time :)

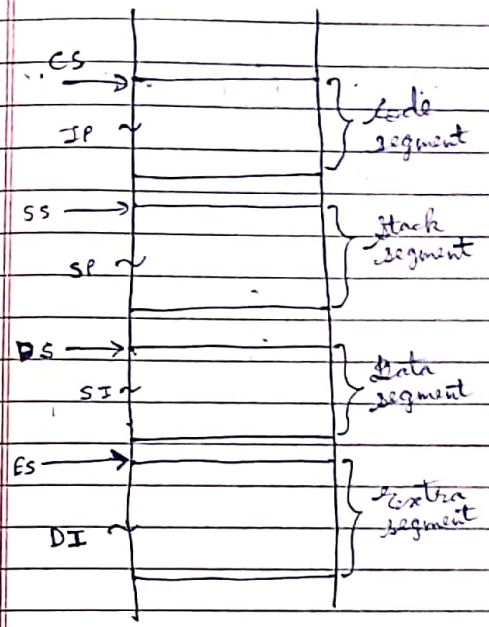
→ Operating frequency is 5MHz – 10MHz

Architecture of 8086:



13/2/25

Memory Segmentation



P.A. = segment address × 10_H + offset address

25/2/25

C.S. = 1000_H & I.P. = 2345_H. Find P.A.

→ Soln:

P.A. = seg. reg. × 10_H + offset address.
= 1000 × 10 + 2345
= 12345_H

25/2/25

Flag register: (8086)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

O – Overflow
D – Direction
I – Interrupt enable
T – Trap
S – Sign
Z – Zero
A – Auxiliary carry
P – Parity
C – Carry

(flags)

I!

* Addressing modes

i) ~~Direct addressing:~~

→ Addressing modes help us to understand the ~~&~~ types of operands & the way they are accessed while executing an instruction.

i) Immediate addressing:

→ MOV AX, $10AB_H$
MOV AL, 45_H

Immediate data is part of the instruction & appears in the form of successive byte or bytes.
Fast because no memory is referenced.

ii) Direct addressing mode:

→ 16-bit memory address is directly specified in the instruction.

Ex: MOV AX, [5000_H]

iii) Register addressing mode:

→ Data is stored in register, it can be 8-bit or 16-bit register. All registers are used in this mode (except IP)

→ MOV AL, BL & MOV AX, BX

iv) Register indirect ~~mo~~ addressing mode:

→ Address of memory location which contains data or operand is determined in indirect way using offset registers.

→ MOV AX, [BX]

v) Indexed addressing mode:

→ MOV AX, [SI]

→ Address is stored in one of the index registers either SI or DI. [Default segment for SI & DI is DS]

vi) Register relative addressing mode:

→ MOV AX, 50_H [BX]

vii) Base plus index addressing mode:

→ MOV AX, [BX] [SI]

viii) Base relative + indexed addressing mode:

→ MOV AX, 40_H [BX] [SI]

25/2/25

ix) Implict addressing mode:

→ CMA -

→ Instruction knows from where instruction has to access data.

→ CMA is complement contents of accumulator.

27/2/25

Data Representation & Arithmetic Algorithm

The number 23.678 is a decimal number with a fractional part & an integer part.

$$23.678$$
$$2\times10^{1}+3\times10^{0}+6\times10^{-1}+7\times10^{-2}+8\times10^{-3}$$

Similarly, a very large or a very small number can be represented using mantissa & exponent part

$$1234567.89 = 1.23456789\times10^{6} \leftarrow \text{exponent}$$
$$\uparrow$$
$$\text{mantissa}$$

It can be seen that the position of the radix point (.) in the number varies according to the significant digit of the number but still computers must be able to adjust automatically, hence they are known as floating point numbers.

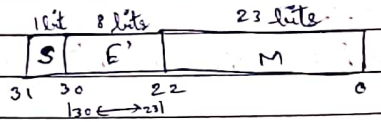
In case of numbers where position of radix remains the same, they are termed as fixed point numbers. The decimal number shown above are adjusted to have only one significant digit before the radix point. These number are said to be normalized.

Exponent - It indicates the current decimal location. The number after radix point is called mantissa.

27/2/25

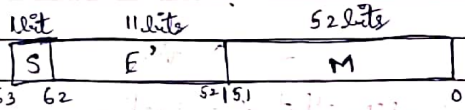
* ANSI / IEEE standard for floating point numbers:

i) Single precision format for IEEE: (32-bit)



$E' = E + 127$

ii) Double precision format



$E' = E + 1023$

Q. Represent $(1259.125)_{10}$ in single & double precision format.

→ Step 1:

2	1259	1	$(10011101011.001)_2$
2	629	1	
2	314	0	$0.125 \times 2 = 0.250$
2	157	1	$0.250 \times 2 = 0.5$
2	78	0	$0.5 \times 2 = 1.0$
2	39	1	
2	19	1	
2	9	1	
2	4	0	
2	2	0	
	1	1	

27/2/25

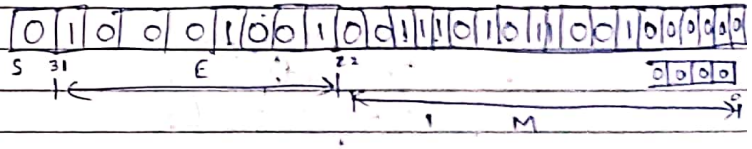
Step 2: $1.00111010111001 \times 2^{10}$

$E = 10 + 127 = 137$

$E' = (10001001)_2$

2	137	1
2	68	0
2	34	0
2	17	1
2	8	0
2	4	0
2	2	0
	1	1

Step 3:



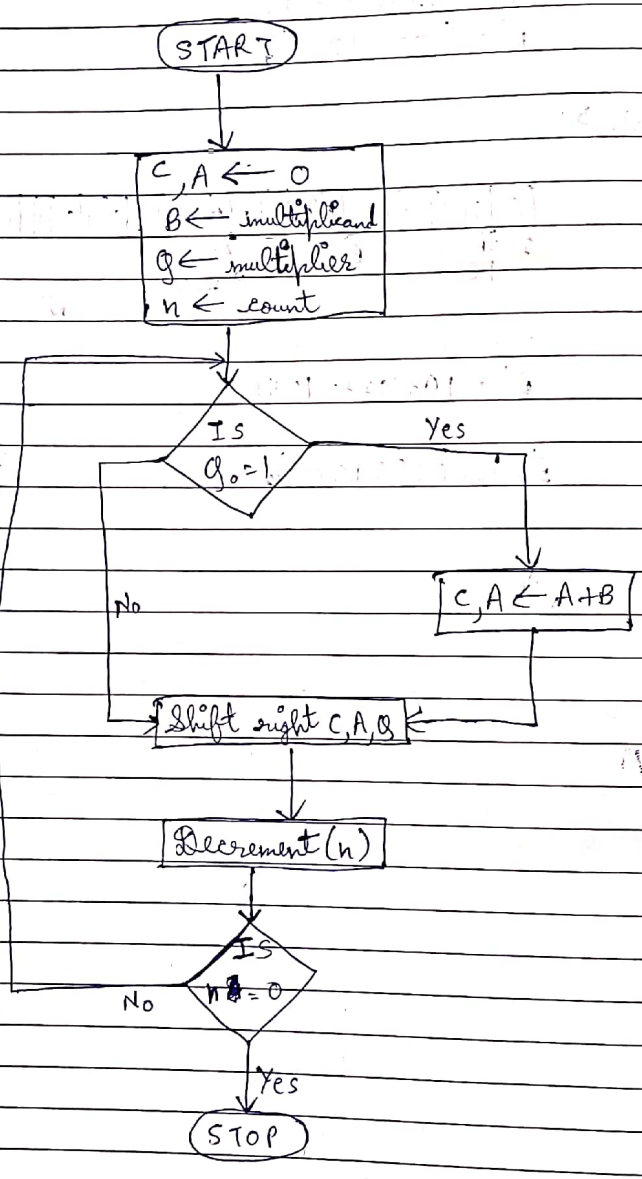
→ $E' = 10 + 1023 = 1033$

$E' = (10000001001)_2$

2	1033	1
2	516	0
2	258	0
2	129	1
2	64	0
2	32	0
2	16	0
2	8	0
2	4	0
2	2	0
	1	1

27/2/25

* Multiplication method (CAQ method):



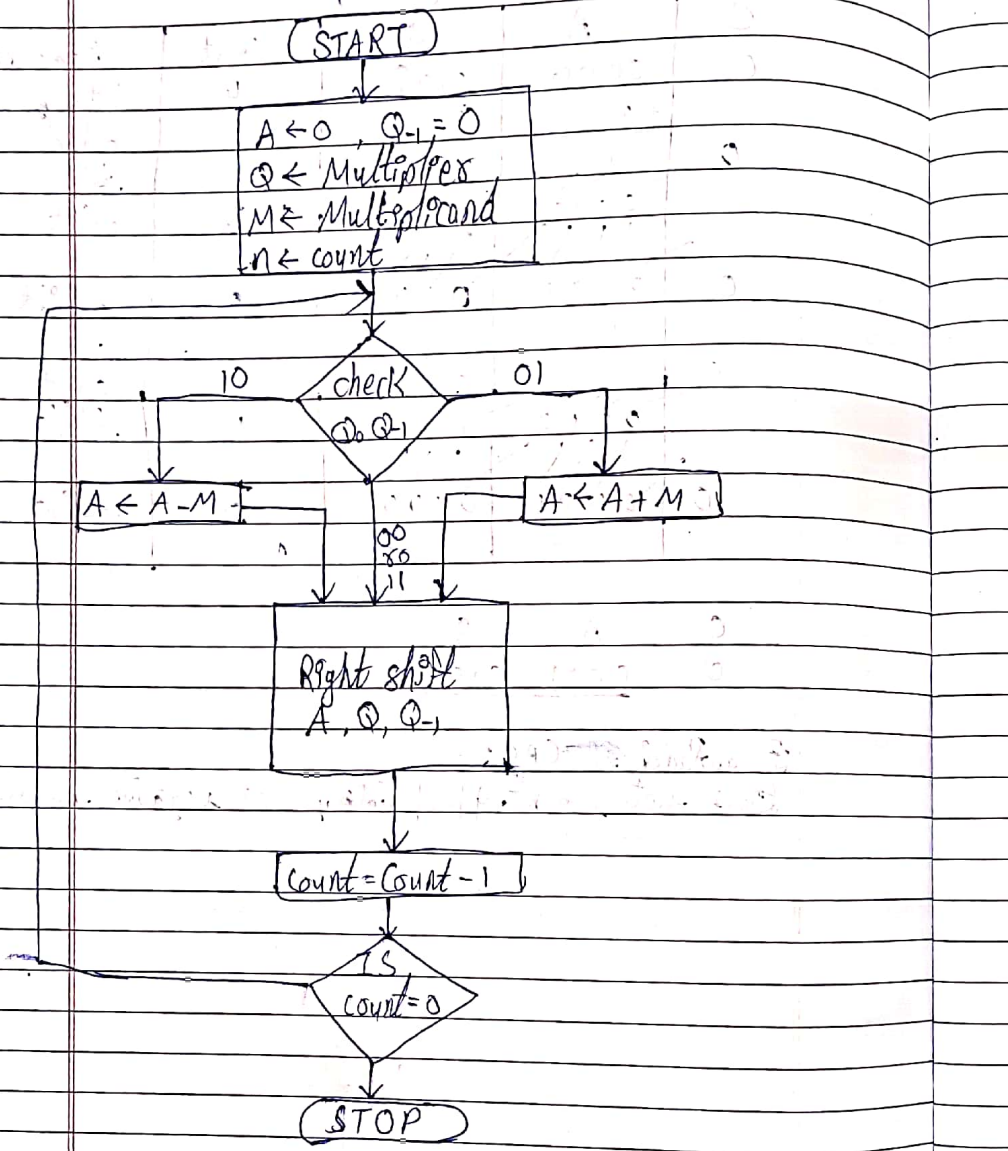
(4) (5)

B = 0100, Q = 0101, C = 0, A = 0

C	A	Q	Count	Remarks
0	0000	0101 ($q_3 q_2 q_1 q_0$)	4	Initialize
	+ 0100			Add
0	0100	0101		
	0010	0010 ($q_3 q_2 q_1 q_0$)	3	Shift
0	0001	0001 ($q_3 q_2 q_1 q_0$)	2	Shift
	+ 0100			
0	0101	0001 ($q_3 q_2 q_1 q_0$)		Add
	0010	1000	1	Shift
0	0001	0100	0	Shift

C	A	Q	
0	0001	0100	⇒ 20

Drawback of CAQ:
Does not work appropriately for negative number.


START
A ← 0 , Q-1 = 0
Q ← Multiplier
M ← Multiplicand
n ← count
check
Q0 Q-1
10
01
A ← A - M
A ← A + M
00
11
Right shift
A, Q, Q-1
Count = Count - 1
Is
Count = 0
STOP

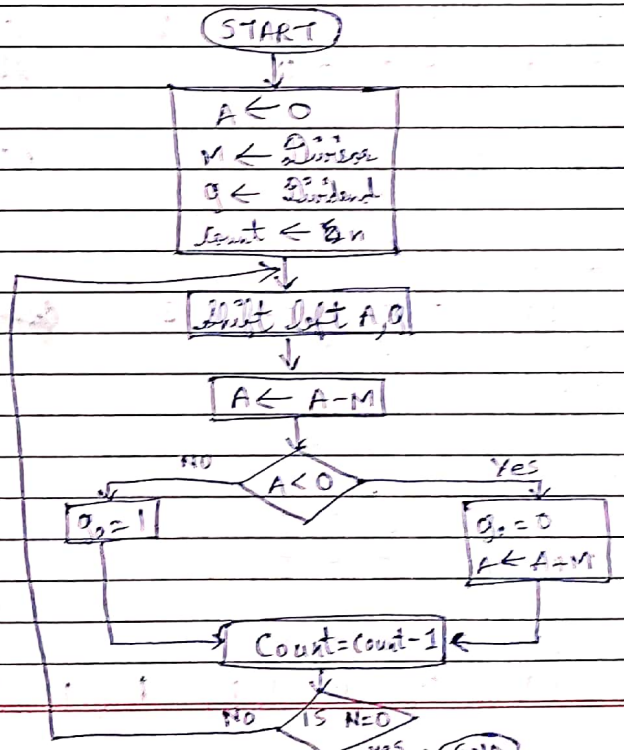
4/3/25

Division of integers:

Eg: 219/6 → 11011011 / 110

```
      100100
110 ) 11011011
      110
      ------
       00011
         110
         ---
          011
```

a) Restoring division algorithm:

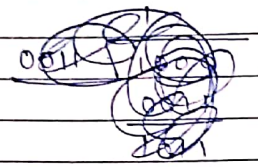


2 4/3/25

Q. Divide following unsigned numbers using restoring division method

i) 8 ÷ 3

Q = 8 → 1000 | 1000
M = 3 → 0011 | 0011



(M+1 = 5) no. of digits

Q = 1000 M = 00011 A = 0000 count = 4

A	Q	count	Remarks
00000	1000	4	Initialize
00001	0000		Shift left
+11101			A ← A-M
11110			
+00011			
00001	0000	3	A ← A+M, Q_0 = 0
00010	0000		Shift left
+11101			A ← A-M
11111			
+00011			
00010	0000	2	A ← A+M, Q_0 = 0
00100	0000		Shift left
+11101			A ← A-M
00001	0000		
00001	0001	1	Q_0 = 1

4/3/25

00010	0010		Shift left
+11101			
11111	0010		A ← A-M
+00011			1
00010	0010	0	A ← A+M, Q_0 = 0
A = 0010	Q = 0010		

ii) 17 ÷ 3

Q = 17 = 10001 1
M = 3 → 00011 -M = 11101

A	Q	count	Remarks
000000	10001	5	Initialize
000001	00010		Shift left
+11101			
11110	00010		A ← A-M
+00011			
00001	00010	4	A ← A+M, Q_0 = 0
00010	00100		Shift left
+11101			
11111	00100		A ← A-M
+00011			
00010	00100	3	A ← A+M, Q_0 = 0
00100	01000		Shift left
+11101			
00001	01000		A ← A-M
00001	01001	2	Q_0 = 1
00010	10010		Shift left
+11101			
00010	10010		A ← A-M

4/3/25

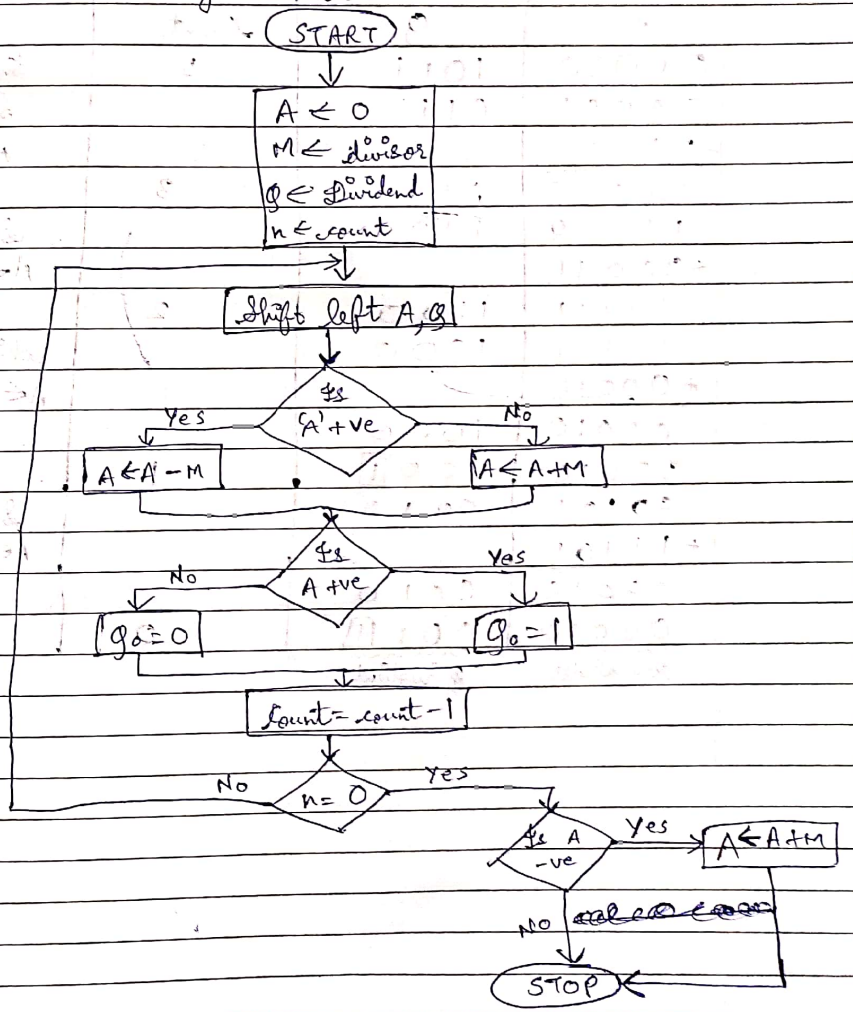
000101	10000			
+00001				
001000	10000	1		A ← A+M, q_0=0
010001	00000			Shift left
+111101				
001110	00000			
				7/3/25
000010	10010			Shift left
+111101				A ← A−M
111111				
+000011				
000010	10010	1		A ← A+M, q_0=0
000101	00100			Shift left
+111101				
000010	00100			A ← A−M
000010	00101	0		q_0 = 1
R	q			

q = 0101 = 5 & R = 0010 = 2

Compare between restoring & non-restoring

Restoring	Non-restoring
i) Needs restoring of register A if the result of subtraction is -ve.	i) Does not require restoring.
ii) In each cycle, content of register A is shifted left then the divisor is subtracted from it (A−M)	ii) In each cycle, content of A is first shifted left then divisor is added or subtracted with content of A depending upon sign of A.
iii) Needs restoring of the remainder if remainder is -ve	iii) Does not require the restoring of remainder
iv) Slower algorithm	iv) Faster algorithm.

Non restoring division:



7/3/25

Q. Using Non-restoring algorithm divide 11 by 3

11 ÷ 3

Q = 1011 M = 00011

-M = 11101

A	Q	Count	
00000	1011	4	Initialization
00001	011		Shift left
+11101			A ← A − M
11110	0110	3	$Q_0 = 0$
11100	1100		Shift left
+00011			A ← A+M
11111	1100	2	$Q_0 = 0$
11111	1000		Shift left
+00011			A ← A+M
00010	1000		
00010	1001	1	$Q_0 = 1$
00101	0010		Shift left
+11101			A ← A-M
00010	0010		
00010	0011	0	$Q_0 = 1$
Remainder	Quotient		

7/3/25

Q. Using non-restoring algorithm, divide 12 by 3

12 ÷ 3

Q = 1100 M = 00011

-M = 11101

A	Q	Count	Remarks
00000	1100	4	Initialize
00001	1000		Shift left
+11101			A ← A - M
11110	1000	3	$Q_0 = 0$
11101	0000		Shift left
+00011			A ← A+M
00000	0001	2	$Q_0 = 1$
00000	0010		Shift left
+11101			A ← A-M
11101	0010	1	$Q_0 = 0$
11010	0100		Shift left
+00011			A ← A+M
11101	0100	0	$Q_0 = 0$
+00011			A ← A+M
00000	0100		
Remainder (0)	Quotient (4)		

7/3/25

Q.

8 ÷ 5

Q = 1000 M = 00101

-M = 11011

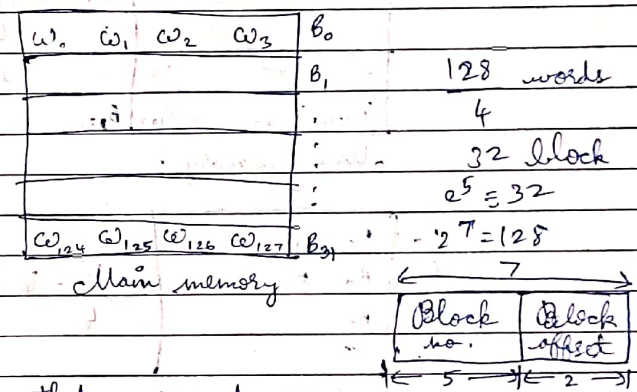
A	Q	Count	Remarks
00000	1000	4	Initialize
00001	0000		Shift left
+11011			A ← A − M
11100	000[0]	3	$Q_0 = 0$
11000	0000		Shift left
+00101			A ← A+M
11101	000[0]	2	$Q_0 = 0$
11010	0000		Shift left
+00101			A ← A+M
11111	000[0]	1	$Q_0 = 0$
11110	0000		Shift left
+00101			A ← A+M
00011	0000		
00011	000[1]	0	$Q_0 = 1$

R = 3 Q = 1

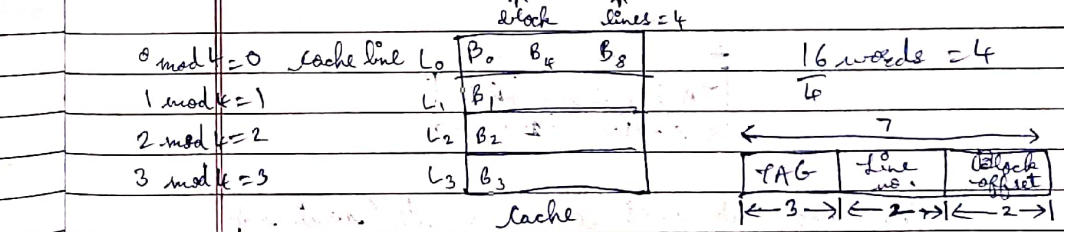
18/3/25

Q. Difference between SRAM & DRAM :

SRAM	DRAM
i) Static RAM	i) Dynamic RAM
ii) FLIP FLOP	ii) Capacitor
iii) Not ~~required~~ requires refreshing circuit	iii) requires refreshing circuit
iv) More space on chip	iv) Less space on chip
v) Bit density is less	v) Bit density is more
vi) Fast	vi) Slow
vii) Costly	vii) Cheaper
viii) Used in cache	viii) Used in main memory

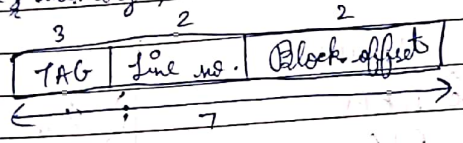


i) Direct method : k mod n



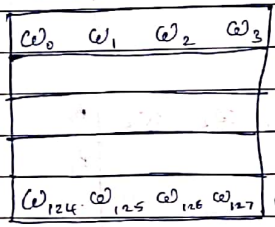
2 18/3/25

→ In cache memory, word is accessed by



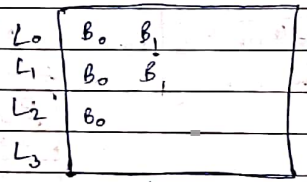
0 mod 4 = 0
1 mod 4 = 1
4 mod 4 = 0
TAG tells us which block is stored in the cache line. (Disadvantage - cache conflict)

ii) Associative Mapping : k mod n



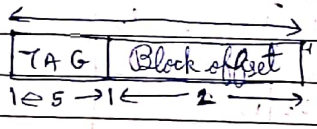
Main memory.

$\frac{128}{4} = 32$
$2^7 = 32$



$\frac{16}{4} = 4$ lines $= 2^2$

Cache memory:

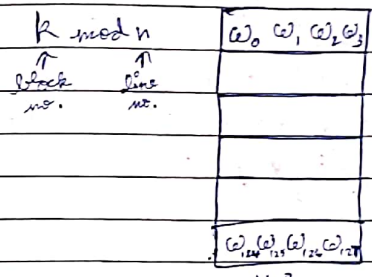


Advantage : Resolves cache conflicts (direct mapping)
Disadvantage : Requires time for searching.

2 set associative mapping

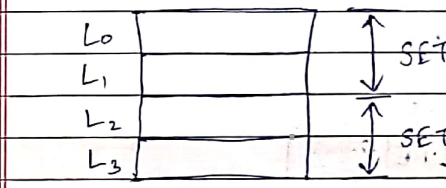
20/3/25

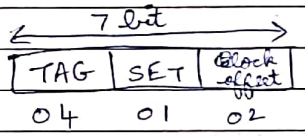
iii) Set Associative mapping : (k set associative mapping)



Main memory

$2^7 = 128$ bits
$\frac{128}{4} = 32 = 2^5$



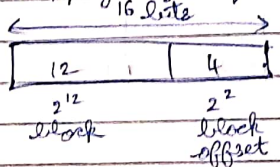


Advantages:
i) Better flexibility
ii) Search is limited

Disadvantages:
i) costly

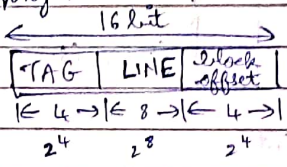
20/3/25

Q. Consider a cache consisting of 256 blocks of 16 words each for a total of 4096 words. Assume that main memory is addressable by $(4 \times 2^{10} = 2^{12})$ 16 bit address. It consist of 4k blocks. How many bits are there in each of Tag, block/set & word fields for different mapping techniques.

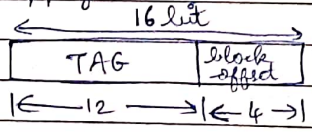


Word = 2^4 = 4 bits

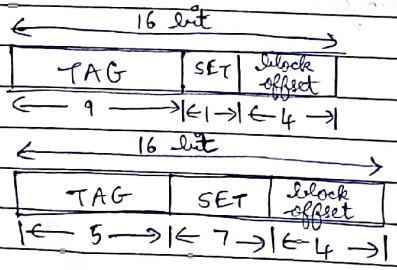
i) Direct mapping:



ii) Associative mapping



iii) Assume 2 set associative mapping



21/3/25

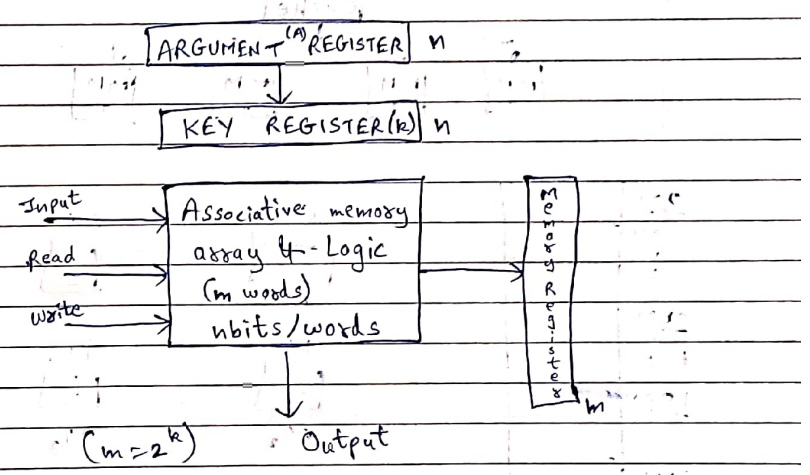
Cache coherency:

Whenever a new block is brought into the cache, the old block is flushed into memory. This is done by checking a bit known as 'dirty bit'/'altered bit'/'changed bit'. If this bit is 1 it indicates cache block is altered so the block is to be copied back to the memory.

(one in cache & main memory)

There are times when two copies are not the same. This inconsistency is known as cache coherency problem.

Associative memory: (content based

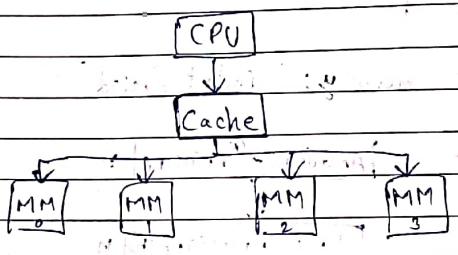


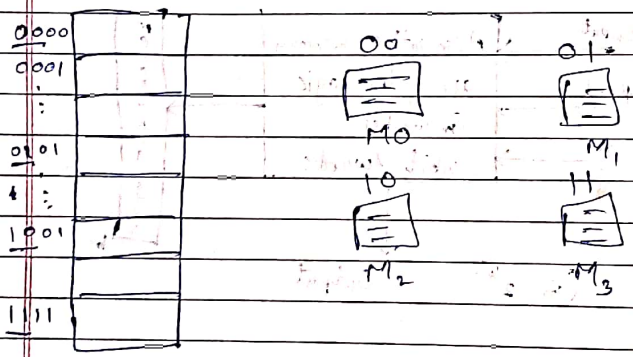
→ The time required to find an object stored in memory can be reduced considerably if objects are selected based on their contents not on their locations.

→ A memory unit accessed by the content is called as associative memory.

→ This type of memory is accessed simultaneously & in parallel.

* Inter-leaved memory :

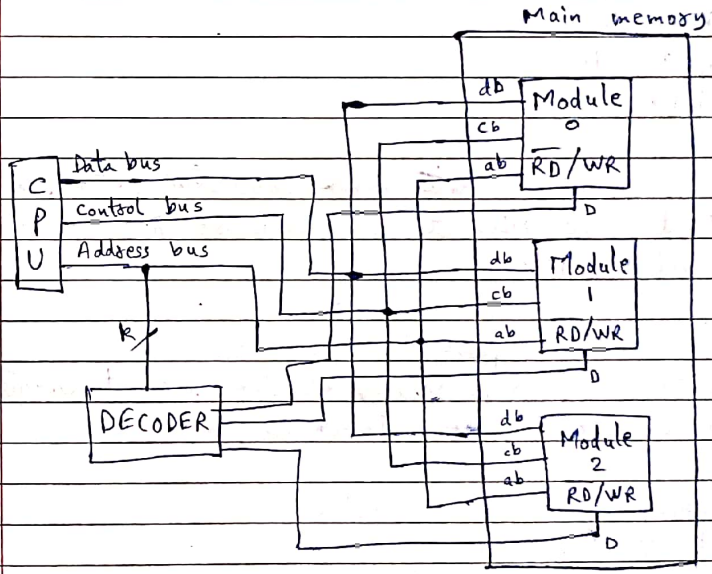




→ Main memory is divided into a number of memory modules & addresses are arranged such that the successive words in address space are placed in different modules.

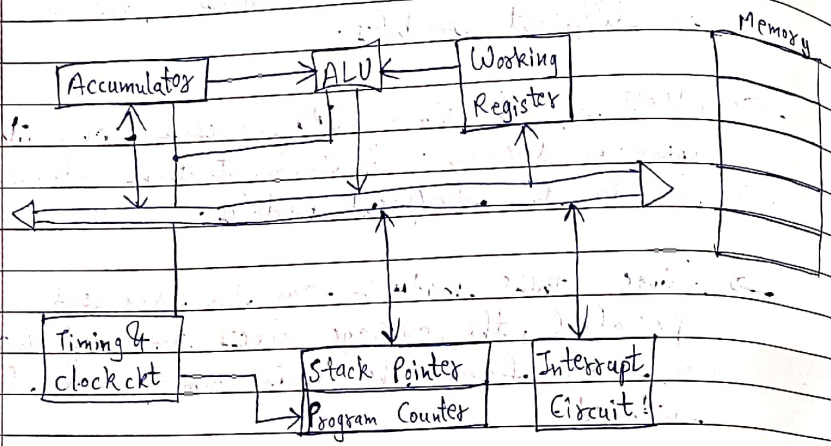
→ Most of the times, CPU accesses consecutive memory location. In such situations address will be to the different modules.

→ Since these modules can be accessed in parallel, the average access time of fetching a word from the main memory can be reduced.



25/3/25

* Processor Organization & Architecture :



i) Fetch
ii) Decode
iii) Execute

→ The processor unit when combined with control unit that supervises the sequence of micro operation is called CPU.

General purpose register
Data register, address register, segment register, index register & stack pointer.

Control & status register
Program counter
Instruction register
MAR (Memory address register)
MBR (Memory data register)

25/3/25

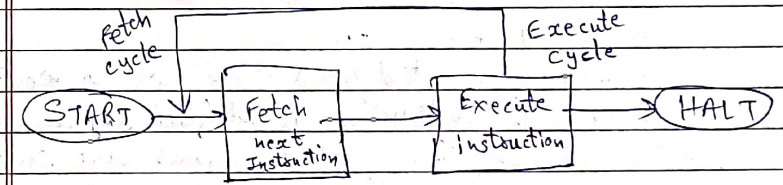
Instruction format :

→ It defines the layout of the bits of an instruction. An instruction format must include an opcode & zero or more operands in one of the addressing modes.

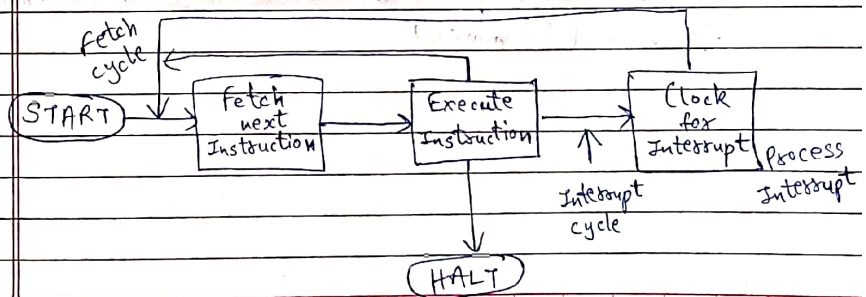
Instruction class	Example
Three address	ADD R_1, R_2, R_3
Two address	ADD R_1, R_2
One address	ADD R_1
Zero address	ADD

[JAE-2] 2M

* Basic instruction cycle :



* Instruction cycle with interrupt :



27/3/25

27/3/25

* Instruction & Instruction sequencing :

→ Instruction must be capable of performing following operation :

i) Data transfer between memory & processor registers.
Processes registers are $R_1, R_2, R_3, \ldots$.
Address of memory location : LOC PLACE MEMORY
I/O registers are represented by DATAIN/DATAOUT
Content of register or memory location are denoted by placing a square bracket $R_2 \leftarrow [LOC]$

ii) Perform arithmetic & logic operations on data.

iii) Program sequencing & control.

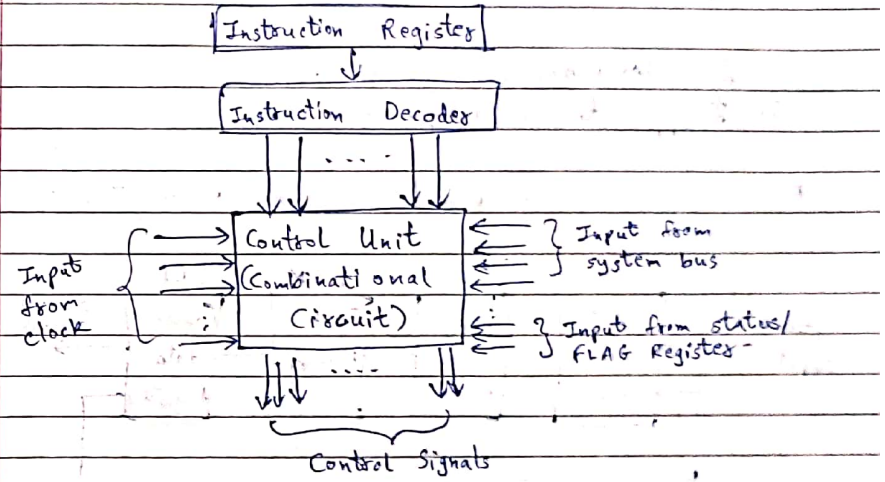
iv) I/O control.

Micro - operation :
Operations executed on data stored in register

Micro - instruction :
Every bit of a control word stands for a control signal. Each micro-instruction is defined by a set of control signals, these control signals are encoded in a control word & this is called as micro instruction

Micro Program :
A sequence of micro-instruction is called micro program (firmware)

Q. Explain hard wired control organization.



→ The control unit is implemented using hardware, i.e. gates, flip flops, etc. We need a special digital circuit that uses as inputs, the bits from opcode field in our instructions, bits from the flag register, signals from the bus & signal from the clock. It should produce as outputs the control signal to drive various components in the computer.

→ In hard wired control unit, we physically connect all of the control lines to actual machine instruction.

→ The instructions are divided up into fields & different bits in the instructions are combined through various digital logic components to drive the control line.

27/3/25

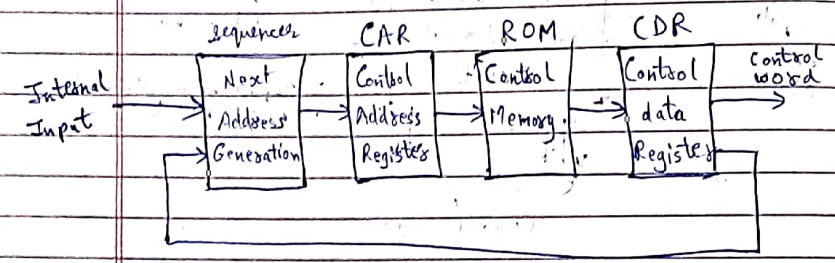
Advantage of hard wired:

i) Very fast

Disadvantage of hard wired:

i) Difficult to design & modify.

* Micro-programmed control organization:



→ A control unit whose binary control variables are stored in memory are called control memory.

→ The control variables at any given time can be represented by strings of 1's & 0's called as control word.

→ Each word in control memory contains micro-instruction within which specifies one or more micro-operations for the system.

→ A sequence of micro-instruction constitute a micro-program.

→ The control memory is assumed to be ROM within which all control informations are stored permanently.

CAR - It specifies address of micro-instruction

CDR - It holds micro instruction read from memory.

The micro instruction contains a control word that specifies one or more micro operations

Once operation are executed, control must determine next address. Next address may also be a function of external input condition.

Advantage:

→ No hardware change

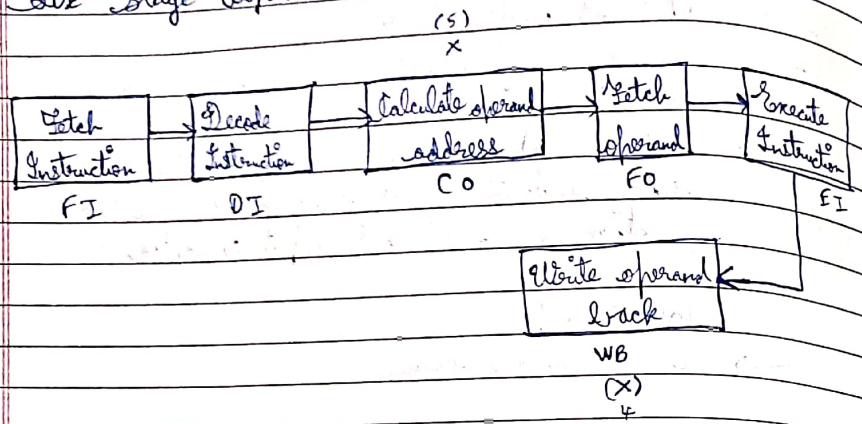
Disadvantage:

→ Slow

	Hardwired	Micro-programmed
i)	Speed is fast	i) Speed is slow.
ii)	More costly	ii) Cheaper
iii)	Occurence of error is more	iii) Occurence of error is less.
iv)	Control functions implemented in hardware	iv) Control functions implemented in software
v)	Not flexible	v) More flexible
vi)	Difficult to handle complex instruction set	vi) Easier to handle complex instruction set.
vii)	Design is harder	vii) Systematic & simple process
viii)	Complex decoding & sequencing logic	viii) Easier decoding & sequencing logic
ix)	More chip area	ix) Less chip area
x)	Used in RISC processor	x) Used in mainframes

9/4/25

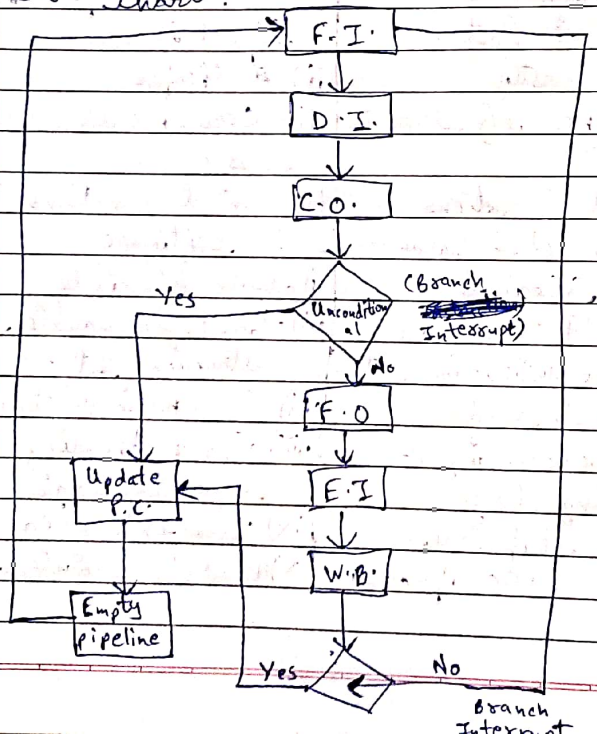
* Six stage Pipelined architecture:



4 stage → FI, DI, FO, EI

5 stage → FI, DI, FO, EI, WB

→ Flow chart:



No pipeline
6×6=36

Pipeline:
11

* Six stage pipelined processor timing diagram:

Instruction ↓ \ Time →	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}
1	FI	DI	CO	FO	EI	WB					
2		FI	DI	CO	FO	EI	WB				
3			FI	DI	CO	FO	EI	WB			
4				FI	DI	CO	FO	EI	WB		
5					FI	DI	CO	FO	EI	WB	
6						FI	DI	CO	FO	EI	WB
7											
8											
9											
⋮											

Branch in 6 stage pipeline:

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}
1	FI	DI	CO	FO	EI	WB								
2		FI	DI	CO	FO	EI	WB							
3			FI	DI	CO	FO	EI	WB						
4				FI	DI	CO	FO	-						
5					FI	DI	CO	-						
6						FI	DI	-						
7							FI	-						
8								FI	DI	CO	FO	EI	WB	
									FI	DI	CO	FO	EI	WB

I/O

Reference - William Stallings