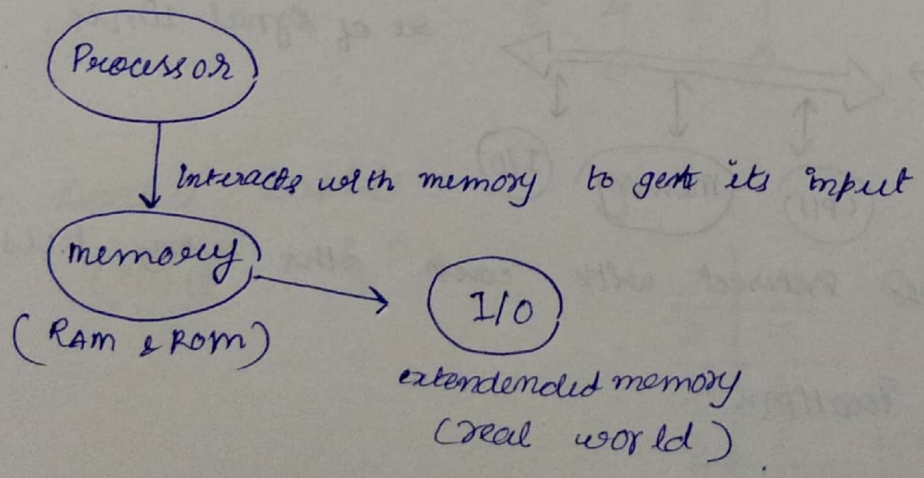
Computer System Architecture [CSEG-200

- Introduction
- ALU Design
- control unit Design.
- Memory Organization
- I/O organization.

Organization
↓
user level
s/w level

Architecture
↓
Implementation level.

Computer: An electronic machine which process data and can perform arthomatic and logical operation.



Processor + memory + I/O + S/W

(computer) Bare machine ——— computer system.

Scanned by CamScanner

Machine code (Binary bits)

↓

Assembly code (mnemonics)

Eg Add a, b ;

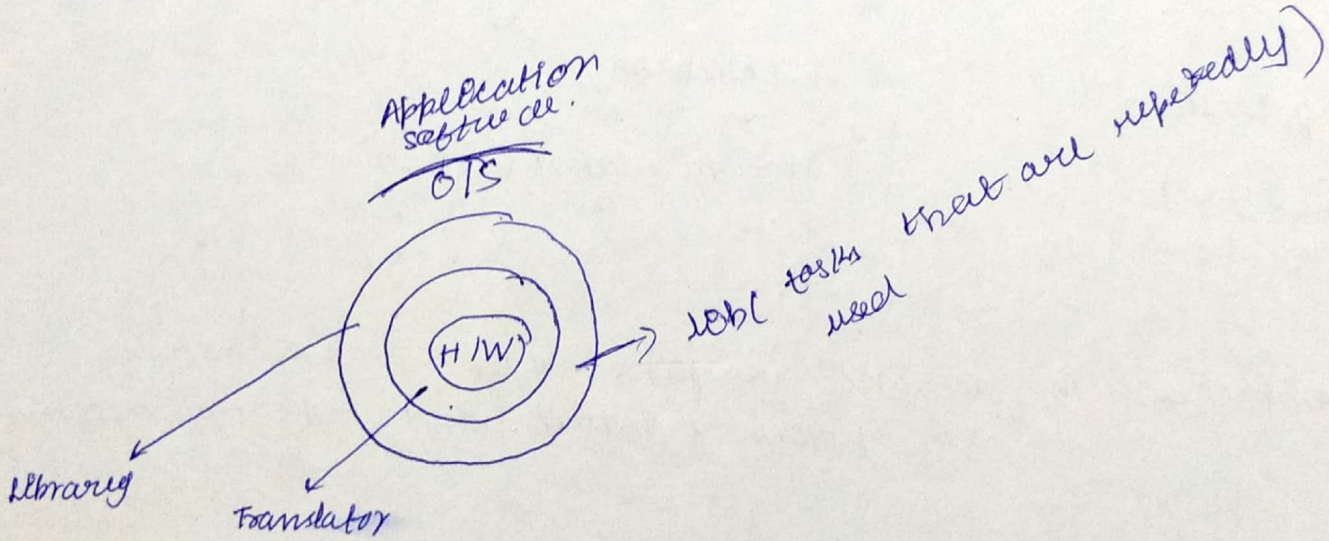
↓

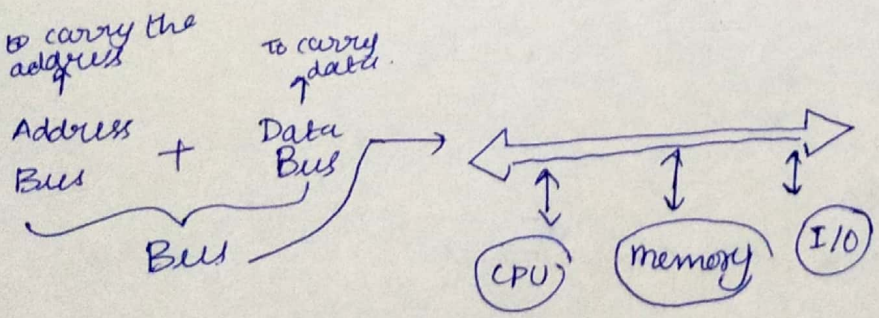
High level language

HLL → ML

Translator
- compiler (has a memory element ∴ slow)
- Interpreter → HLL → ML
- Assembler (AL → ML)

<u>Computer system</u> : combination of bare machine & S/W working together to perform com tasks





These components interact with each other using Bus.

64 bits → 2^{64} locations.

Scanned by CamScanner

* Data bus : size of Data bus is equal to size of single memory location.

* Size of Register is equal to the size of Data bus

NUMBER SYSTEM :

Decimal 0-9
Binary 0, 1
Octal 0-7 } → used to reduce the bits to store a large
Hexadecimal } no. in memory.
(0-9, A-E)

Conversions :

1) Decimal to Binary :
(i) Streamlined Method (ii) Double Dabble method.

(i) Streamline

$(10)_{10} = (?)_2$. $= (1010)_2$

using weights. →

2^3	2^2	2^1	2^0
8	4	2	1
1	0	1	0

(ii) Double Dabble.

$(10)_{10} = (?)_2$.

2	10	0 LSB
2	5	1
2	2	0
2	1	1 MSB
	0	

$= (1010)_2$

Scanned by CamScanner

Ans $(0.8125)_{10} = (?)_2$

$0.8125 \times 2 = 1.6250$ 1 MSB

$0.6250 \times 2 = 1.2500$ 1

$0.2500 \times 2 = 0.5000$ 0

$0.5000 \times 2 = 1.000$ 1 LSB

$= (0.1101)_2$

Ans $(0.95)_{10} = (?)_2$.

$0.95 \times 2 = 1.90 = 1$

$0.90 \times 2 = 1.80 = 1$

$0.80 \times 2 = 1.60 = 1$

$0.60 \times 2 = 1.20 = 1$

$0.20 \times 2 = 0.40 = 0$

$0.40 \times 2 = 0.80 = 0$

$0.80 \times 2 = 1.60 = 01.$

$= (0.111100)_2$

II) Binary to Decimal

Ans. $(1010)_2 = (?)_{10}$

$1 \times 2 = 2$

$2 + 0 = 2$

$2 \times 2 = 4$

$4 + 1 = 5$

$5 \times 2 = 10$

$10 + 0 = 10$

$= (10)_{10}$

Scanned by CamScanner

4. $(0.101)_2 = (?)_{10}$

$0.\ 1\ 0\ 1$
$2^{-1}\ 2^{-2}\ 2^{-3}$

$= 2^{-1} + 2^{-3}$
$= \frac{1}{2} + \frac{1}{8} = 0.5 + 0.125$
$= (0.625)_{10}$

$= 0.(2^{-1} \times 1 + 2^{-2} \times 0 + 2^{-3} \times 1)$
$= (0.625)_{10}$

Octal

Octal to Decimal

4. $(714)_8 = (?)_{10}$

$= 7 \times 8^2 + 1 \times 8^1 + 4 \times 8^0$
$= 7 \times 64 + 8 + 4$
$= (460)_{10}$

4. $(0.512)_8 = (?)_{10}$

$5 \times 8^{-1} + 1 \times 8^{-2} + 2 \times 8^{-3}$

$= \frac{5}{8} + \frac{1}{64} + \frac{2}{64 \times 8}$

$= 0.625 + \frac{1}{64} + \frac{1}{64 \times 4}$

$= 0.625 + 0.015625 + 0.00390$

$= (0.6445)_{10}$

Scanned by CamScanner

$(475.25)_8 = (?)_{10}$

$= 8^2 \times 4 + 8^1 \times 7 + 8^0 \times 5 \quad . \quad 2 \times 8^{-1} + 5 \times 8^{-2}$

$= 64 \times 4 + 8 \times 7 + 5 \quad . \quad 2/8 + 5/8^2$

$= 256 + 56 + 5 \quad . \quad 0.25 + 0.0507812$

$= (317.328)_{10}$

Decimal to Octal

$(20)_{10} = (?)_8$

8	20	4
8	2	2
	0	

$= (24)_8$

$(0.64025)_{10} = (?)_8$

$0.64025 \times 8 = 5.12200 \quad 5$

$0.12200 \times 8 = 0.97600 \quad 0$

$0.97600 \times 8 = 7.80800 \quad 7$

$0.80800 \times 8 = 6.46400 \quad 6$

$= (0.5076)_8$

Scanned by CamScanner

Binary to Octal

Ex. $(010111011)_2 = (?)_8$

$\underbrace{010}_{2}\ \underbrace{111}_{7}\ \underbrace{011}_{3} = (273)_8.$

~~Ex $(765)_8$~~

Octal to Decimal

$(2765)_8 = (?)_2$

$(010\ 111110\ 101)_2.$

Ex. $(725.63) = (?)_2.$

$(111010\ 1010.110011)_2$

Hexadecimal

→ Hexadecimal to Decimal

Ex $(7E4)_{16} = (?)_{10}.$

$= 7\times16^2 + E\times16 + 4\times16^0$

$= 7\times16^2 + 14\times16 + 4$

$= (2020)_{10}$

Scanned by CamScanner

Ans 0.582

$= 5\times16^{-1} + 8\times16^{-2} + 2\times16^{-3}$

$= \frac{5}{16} + \frac{8}{16^2} + \frac{2}{16^3}.$

$= 0.3125 + 0.03125 + 0.000488$

$= (0.3442)_{10}$.

Ans $(101101.1101)_2 = (?)_{16} = (?)_{10} = (?)_8$.

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

$\underbrace{0010}\underbrace{1101}$

$(2D.D)_{16}$

$1\times2^4 + 1\times2^3 + 1\times2^2 + 1\times2^0 . 1\times2^{-1} + 1\times2^{-2} + 1\times2^{-4}$

$= 32 + 8 + 4 + 1 . 8125$

$= (45.8125)_{10}$

$00\underbrace{101}\underbrace{101} . \underbrace{110}\underbrace{100}$

$(55.64)_8$

Scanned by CamScanner

Q. $(\overset{3}{2}\overset{2}{A}\overset{1}{C}\overset{0}{5}.D)_{16} = (?)_{10} = (?)_8 = (?)_2$

$= 2 \times 16^3 + 16^2 \times 10 + 16^1 \times 12 + 5 . 16^{-1} \times 13$

$= 10949.8125$

2 A C 5. D

$(0010\ 1010\ 1100\ 0101\ .\ 1101)_2$

$\underbrace{010}\underbrace{101}\underbrace{011}\underbrace{000}\underbrace{101}.\underbrace{110}\underbrace{100}$

2 5 3 0 5 . 6 4.

$(25305.64)_8$

Q. $(365)_8 = (?)_H$

$\underbrace{0111}\underbrace{0101} = F5$

Q $(22.64)_{10} = (?)_H$

$\underbrace{0001}\underbrace{0110}.\underbrace{1010}\underbrace{0011}$

$= (16.A3)_H$

0.64×2	$= 1.28$	1
0.28×2	$= 0.56$	0
0.56×2	$= 1.12$	1
0.12×2	$= 0.24$	0
0.24×2	$= 0.48$	$= 0$
0.48×2	$= 0.96$	0
0.96×2	$= 1.92$	1
	1.84	1

2	22	
2	11	0
2	5	1
2	2	1
	1	0

10110

Scanned by CamScanner

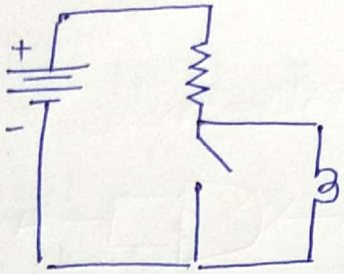
Ans $(11011.1100)_2 = (?)_H$

$(1B.C)_H$

Logic Gates :

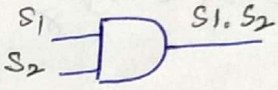
1. NOT / Inverter

7404

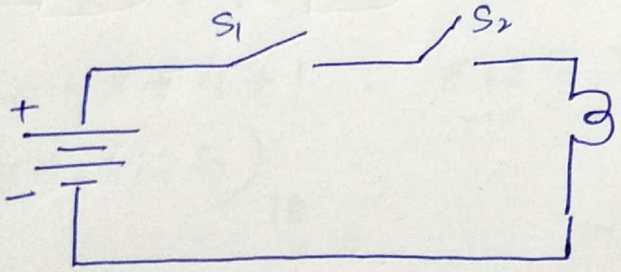


S	Y
0	1
1	0

2. AND / logical multiplicator.

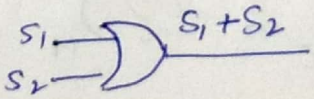


7408

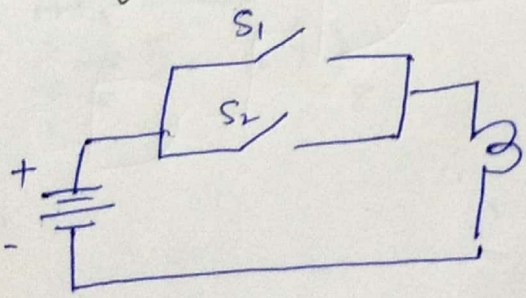


S_1	S_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

3. OR / logical addition



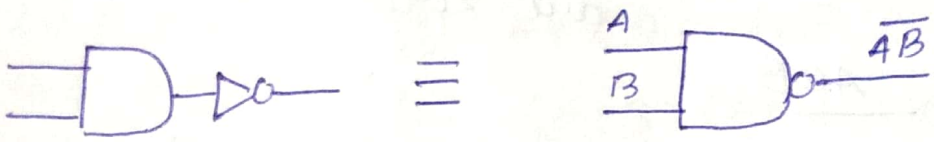
7432



S_1	S_2	Y
0	0	0
0	1	1
1	0	1
1	1	1

Scanned by CamScanner

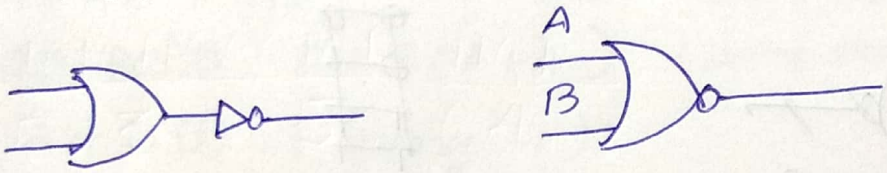
4. ~~Nand~~ NAND



7400.

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

5. NOR

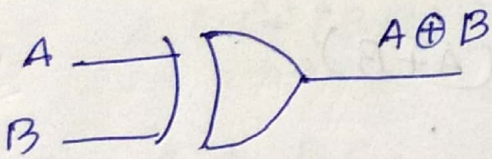


7402

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

6. Ex OR $A \oplus B$

No monolytic chip.



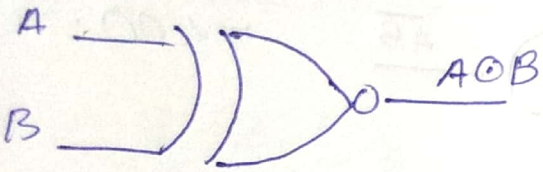
odd no. of one's = 1

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Scanned by CamScanner

7) Ex - NOR $A \odot B$

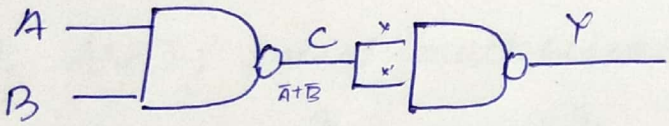
even no. of 1's or all are zero



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

- ~~NOT from NAND~~

NAND to AND.



A	B	C	Y
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

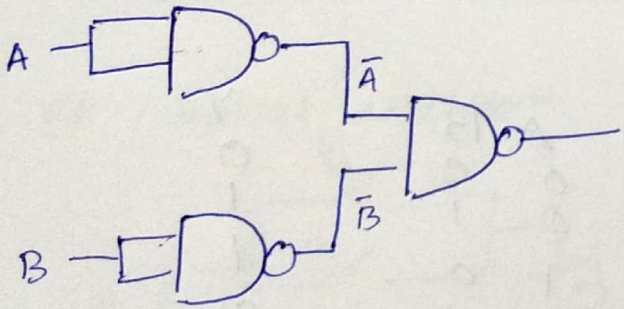
$\overline{A.B} = \overline{A}+\overline{B}$

$= \overline{x.x'} = \overline{x}+\overline{x'}$

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

$= A.B + A.B$

$= A.B.$



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

$\overline{A.B} \neq \overline{A}.\overline{B}$

$A.B + AB$

Scanned by CamScanner

1. $A + 0 = A$
2. $A + 1 = 1$
3. $A \cdot 0 = 0$
4. $A \cdot 1 = A$
5. $A + \bar{A} = 1$
6. $A \cdot \bar{A} = 0$
7. $\bar{\bar{A}} = A$
8. $A + AB = A$
9. $A + \bar{A}B = A + B$
10. $(A+B)(A+C) = (A + B.C)$

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

$A + AB + \bar{A}B$ (using identity 8).

$= \cancel{A+} \cancel{B(A+B)} \; A + B(A + \bar{A})$ (using identity 5)

$= A + B.(1)$ (using 4)

$= A + B$

10. $(A+B)(A+C) = A + B.C$

$= A.A + A.C + B.A + B.C$ {using distributive law

$= A + A.C + A.B + B.C$ {using $A.A = A$.

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

$= A(1 + (B+C)) + BC$ { $1 + (B+C) = 1$ i.e 2 }.

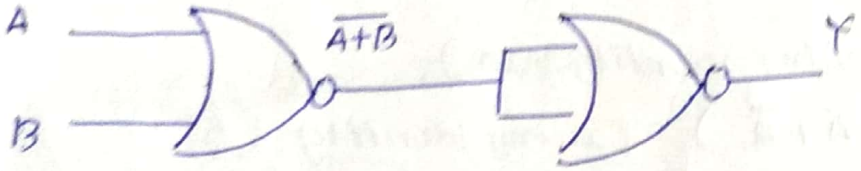
$= A + BC.$

Scanned by CamScanner

NOR to NOT

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

NOR to OR

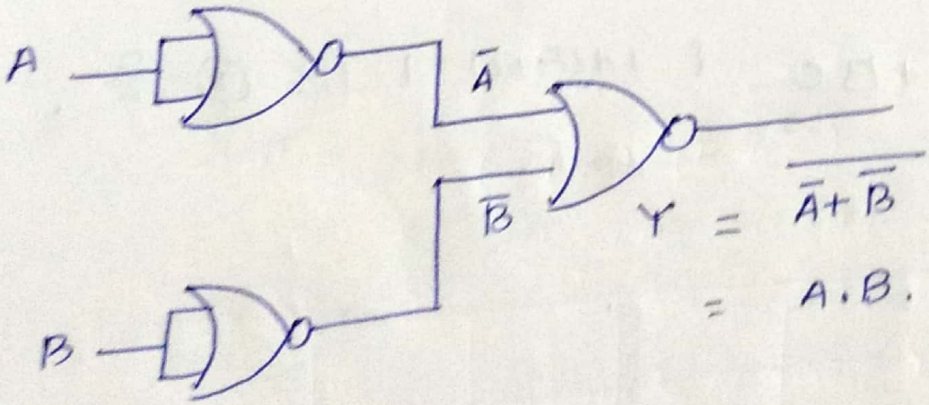


$$Y = \overline{\overline{A+B} + \overline{A+B}}$$

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

$$= A+B$$

NOR to AND



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

$$= A.B.$$

Scanned by CamScanner

i) $AB + AC + \overline{B}C = AB + \overline{B}C$

$LHS = AB + AC + \overline{B}C$

$= AB + AC(B+\overline{B}) + \overline{B}C$ $\quad [\because B+\overline{B} = 1]$

$= AB + ABC + A\overline{B}C + \overline{B}C$

$= AB(1+C) + \overline{B}C(A+1)$ $\quad [\because 1+A = 1]$

$= AB(1) + \overline{B}C(1)$ $\quad [\because A.1 = A]$

$= AB + \overline{B}C$

ii) $D(\overline{A}+B) + \overline{B}(C+AD) = D + \overline{B}C$

$LHS = D(\overline{A}+B) + \overline{B}(C+AD)$

$= \overline{A}D + BD + \overline{B}C + A\overline{B}D$

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

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

$= D(\overline{A}+B+\overline{(\overline{A}+B)}) + \overline{B}C$ $\quad$ [using Demorgan's Theorem $\because A\overline{B} = \overline{\overline{A}+B}$]

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

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

$= D + \overline{B}C$

(iii) $BC + A\overline{C} + AB + BCD = BC + A\overline{C}$

$= BC(1+D) + A\overline{C} + AB$

$= BC.(1) + A\overline{C} + AB$ $\quad [B.1 = B]$

~~$BC + A(B+\overline{C})$~~

$= BC + A\overline{C} + AB(C+\overline{C})$

$= BC + A\overline{C} + ABC + AB\overline{C}$

$= BC(1+A) + A\overline{C}(1+B)$ $\quad [\because 1+A = 1]$ $\quad [\because A.1 = A]$

$= BC.1 + A\overline{C}.1$

$= BC + A\overline{C}$

Scanned by CamScanner

ity $a[b+c(\overline{a.b+a.c})] = a.b$

$$LHS = a[b+c(\overline{a.b+a.c})]$$ { using demorgans law }

$$= a[b+c(\overline{a.b}.\overline{a.c})]$$

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

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

$$= ab + \{(a\bar{a}c + ac\bar{b}).(a\bar{a}c + ac\bar{c})\}$$ $(\because a.\bar{a} = 0)$

$$= ab + \{(0+ac\bar{b}).(0+0)\}$$ $\because 0+0=0$ & $0+u=a$

$$= ab + \{ac\bar{b}.0\}$$ $[\because a.0 = 0]$

$$= ab.$$

K-Map simplification :

Logical expression can be represented in form of

i) sum of product (SOP) Ex. $(AB+\bar{B}C+CA)$

ii) Product of sum (POS) Ex $(A+B).(B+\bar{C}).(A+C)$

canonical form : All the elements must be present in each term.

Ex. $(A+B).(B+\bar{C})$

Reduce in canonical form

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

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

↑ standar POS

Scanned by CamScanner

Ex. $AB + B\bar{C}$

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

$= ABC + AB\bar{C} + A\bar{B}\bar{C} + \bar{A}B\bar{C}$ standard SOP.

SOP (maximum value min terms) → (zeros are bared) → resultant must be one.

$AB + BC + \bar{A}C$ $\sum m(AB + BC + \bar{A}C)$

POS (minimum value max terms) → (one's are bared) → resultant must be zero

$(A+B).(B+C).(\bar{A}+C)$ $\prod m((A+B).(B+C).(\bar{A}+C))$

A, B, C

A	B	C	mintum		maxterm	
0	0	0	$\bar{A}.\bar{B}.\bar{C}$	m0	$A+B+C$	M0
0	0	1	$\bar{A}.\bar{B}.C$		$A+B+\bar{C}$	
0	1	0	$\bar{A}.B.\bar{C}$		$A+\bar{B}+C$	
0	1	1	$\bar{A}.BC$		$A+\bar{B}+\bar{C}$	
1	0	0	$A.\bar{B}.\bar{C}$		$\bar{A}+B+C$	
1	0	1	$A.\bar{B}.C$		$\bar{A}+B+\bar{C}$	
1	1	0	$A.B.\bar{C}$		$\bar{A}+\bar{B}+C$	
1	1	1	$A.B.C$		$\bar{A}+\bar{B}+\bar{C}$	

Scanned by CamScanner

2-variable K-map $2^2 = 4$ cells

A \ B	$\bar{B}_0$	B_1
$\bar{A}_0$	m_0 00	m_1 01
A_1	m_2 10	m_3 11

3-variable K-map. $2^3 = 8$ cells.

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	000 m_0	001 m_1	011 m_3	010 m_2
A	100 m_4	101 m_5	111 m_7	m_6

4-variable K-map. $2^4 = 16$ cells

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0000 m_0	0001 m_1	0011 m_3	0010 m_2
$\bar{A}B$	0100 m_4	0101 m_5	0111 m_7	0110 m_6
AB	1100 m_{12}	1101 m_{13}	1111 m_{15}	1110 m_{14}
$A\bar{B}$	1000 m_8	1001 m_9	1011 m_{11}	1010 m_{10}

Scanned by CamScanner

4s/ $\Sigma m(1,3)$

A \ B	$\bar{B}$	B
$\bar{A}$	0 (0)	1 (1)
A	0 (2)	1 (3)

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

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

$Y = B.$

4s $\Sigma m(1,3,7,6)$.

A \ BC	$\overline{BC}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0 (0)	1 (1)	1 (3)	0 (2)
A	0 (4)	0 (5)	1 (7)	1 (6)

* When 1's are covered in a group then we can neglect. Its participation in another group.

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

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

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

~~$= C$~~

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

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

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

Scanned by CamScanner

Q.

$Y = \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}D + \bar{A}BCD + \bar{A}BC\bar{D} + A\bar{B}\bar{C}\bar{D} +$
$A\bar{B}\bar{C}D + ABCD + A\bar{B}CD.$

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0000	1 0001	0011	0010
$\bar{A}B$	0100	1 0101	1 0111	1 0110
AB	1 1100	1 1101	1 1111	1110
$A\bar{B}$	1000	1001	1 1011	1010

quadrant not significant

Pairs : $\bar{A}\bar{C}D + \bar{A}BC + AB\bar{C} + ACD$

As

4.

A	B	C	Y	Min term	max term.
0	0	0	1	$\bar{A}\bar{B}\bar{C}$	
0	0	1	0		
0	1	0	0		
0	1	1	1	$\bar{A}BC$	
1	0	0	1	$A\bar{B}\bar{C}$	
1	0	1	0		
1	1	0	0		
1	1	1	1	$ABC.$	

$\bar{A}\bar{B}\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + ABC.$

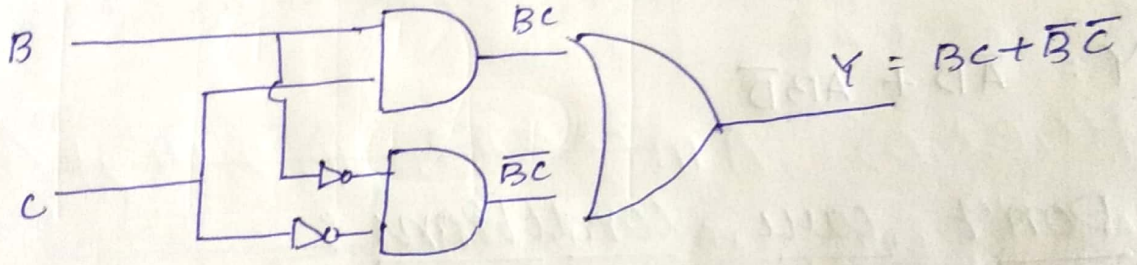
Scanned by CamScanner

A \ BC	$\overline{BC}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$	1 (000)	0 (001)	1 (011)	0 (010)
A	1 (100)	0 (101)	1 (111)	0 (110)

$$Y = \overline{BC} + BC$$

$$Y = \overline{ABC} + A\overline{BC} + ABC + \overline{A}BC$$

$$= \overline{B}\overline{C} + BC$$



4. $f(A,B,C,D) = \Sigma m(0,1,4,8,9,10)$

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

~~$Y = \overline{ABC} + \overline{ACD} + \overline{ABC} + \overline{ABD}$~~

$$Y = \overline{B}\overline{C} + \overline{A}\overline{C}\overline{D} + A\overline{B}\overline{D}$$

Scanned by CamScanner

(2). $f(A,B,C,D) = \bar{A}\bar{B}D + AB\bar{C}\bar{D} + \bar{A}BD + ABC\bar{D}$

$= \bar{A}\bar{B}CD + \bar{A}\bar{B}\bar{C}D + AB\bar{C}\bar{D} + \bar{A}BCD + \bar{A}B\bar{C}D$

$+ ABC\bar{D}$

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

$Y = \bar{A}D + AB\bar{D}$

Don't care conditions:

In some logic circuits certain input conditions never occur, therefore the corresponding output never appears. Such outputs are indicated by a cross of X or d and are called as don't care conditions.

These don't care conditions can be used to increase the no. of groups as their participation will not affect the final result.

Ans $f(A,B,C,D) = \Sigma m(1,3,7,11,15) + \Sigma d(0,2,4)$

Scanned by CamScanner

4.

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

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

4. $f(A,B,C,D) = \Sigma m(0,7,8,9,10,12) + \Sigma d(2,5,13)$

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

$Y = \bar{B}\bar{D} + A\bar{C} + \bar{A}BD.$ Ans

4. $f(A,B,C,D) = \Pi m(0,2,3,8,9,12,13,15).$

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

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

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

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

Scanned by CamScanner

Ans. $f(A,B,C,D) = \pi M(0,2,3,8,9,12,13,15)$.

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0		0	0
$\bar{A}B$				
AB	0	0	0	
$A\bar{B}$	0	0		

$\bar{f} = \bar{A}\bar{B}\bar{D} + ABD + A\bar{C} + \bar{A}\bar{B}C$

$f = (\bar{A}+C)(A+B+D) + (\bar{A}+\bar{B}+\bar{D}) + (A+B+\bar{C})$.

Ans. $f(A,B,C,D) = \pi M(0,3,4,7,8,10,12,14) + d(2,6)$

A+B \ C+D	$C+D$	$C+\bar{D}$	$\bar{C}+\bar{D}$	$\bar{C}+D$
$A+B$	0 (0)	(1)	0 (3)	X (2)
$A+\bar{B}$	0 (4)	(5)	0 (7)	X (6)
$\bar{A}+\bar{B}$	0 (2)	(13)	(15)	0 (14)
$\bar{A}+B$	0 (8)	(9)	(11)	0 (10)

octate = $m_0 . m_4 m_2 m_6$
$m_2 m_6 m_{14} m_{10}$

quad = $m_3 . m_2 m_7 m_6$.

~~Y = (D+D).(Ā+C̄).(C̄+A)~~.

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

$= D.(A+\bar{C})$.

Scanned by CamScanner

4s 5-variable K-Map

$f(A,B,C,D,E) = \Sigma m(0,5,6,8,9,10,11,16,20,24,25,26,27,29,31)$

A = 0

BC \ DE	$\bar{D}\bar{E}$	$\bar{D}E$	DE	$D\bar{E}$
$\bar{B}\bar{C}$	1 (0)	(1)	(3)	(2)
$\bar{B}C$	(4)	1 (5)	(7)	1 (6)
BC	(12)	(13)	(15)	(14)
$B\bar{C}$	1 (8)	1 (9)	1 (11)	1 (10)

A = 1

BC \ DE	$\bar{D}\bar{E}$	$\bar{D}E$	DE	$D\bar{E}$
$\bar{B}\bar{C}$	1 (16)	(17)	(19)	(18)
$\bar{B}C$	1 (20)	(21)	(23)	(22)
BC	(28)	1 (29)	1 (31)	(30)
$B\bar{C}$	1 (24)	1 (25)	1 (27)	1 (26)

Octate : $(m_8\, m_9\, m_{11}\, m_{10}\, m_{24}\, m_{25}\, m_{27}\, m_{26})$

quadrent : $m_{29}\, m_{31}\, m_{25}\, m_{27}$
$m_0\, m_8\, m_{16}\, m_{24}$

ones = m_5 and m_6.

$$Y = B\bar{C} + ABE + A\bar{B}\bar{D}\bar{E} + \bar{C}\bar{D}\bar{E} + \bar{A}\bar{B}C\bar{D}E + A\bar{B}CD\bar{E}$$

4s. $f(ABCDE) = \Sigma m(0,2,4,6,9,11,13,15,17,21,25,27,29,31)$

A = 0

BC \ DE	$\bar{D}\bar{E}$	$\bar{D}E$	DE	$D\bar{E}$
$\bar{B}\bar{C}$	1 (0)	(1)	(3)	1 (2)
$\bar{B}C$	1 (4)	(5)	(7)	1 (6)
BC	(12)	1 (13)	1 (15)	(14)
$B\bar{C}$	(8)	1 (9)	1 (11)	(10)

A = 1

BC \ DE	$\bar{D}\bar{E}$	$\bar{D}E$	DE	$D\bar{E}$
$\bar{B}\bar{C}$	(16)	1 (17)	(19)	(18)
$\bar{B}C$	(20)	1 (21)	(23)	(22)
BC	(28)	1 (29)	1 (31)	(30)
$B\bar{C}$	(24)	1 (25)	1 (27)	(26)

$$Y = \bar{A}\bar{B}\bar{E} + BE + A\bar{D}E$$

Scanned by CamScanner

Combinational circuits :

① Half Adder. (2 bits addition

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

SUM = $\bar{A}B + A\bar{B}$ = $A \oplus B$.

CARRY = AB

Next step: K-MAP simplification.

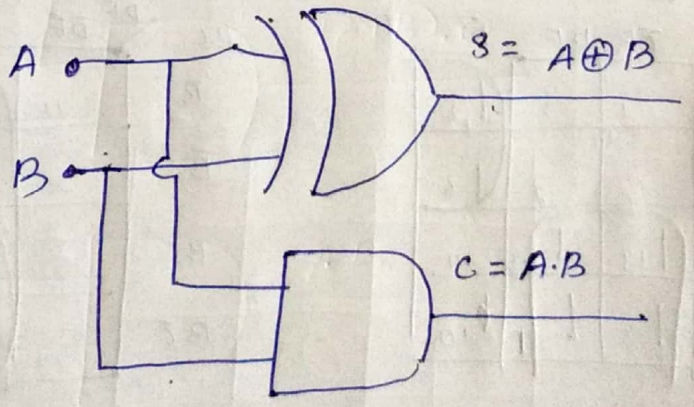
SUM

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

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

or $A \oplus B$.

CARRY = AB.



Scanned by CamScanner

② FULL ADDER. (3 bits addition).

A	B	C_{in}	sum	carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\text{sum} = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC.$$

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

$$= \underbrace{\bar{A}(B \oplus C) + A(\overline{B \oplus C})}_{\text{XOR expression}}$$

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

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

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

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

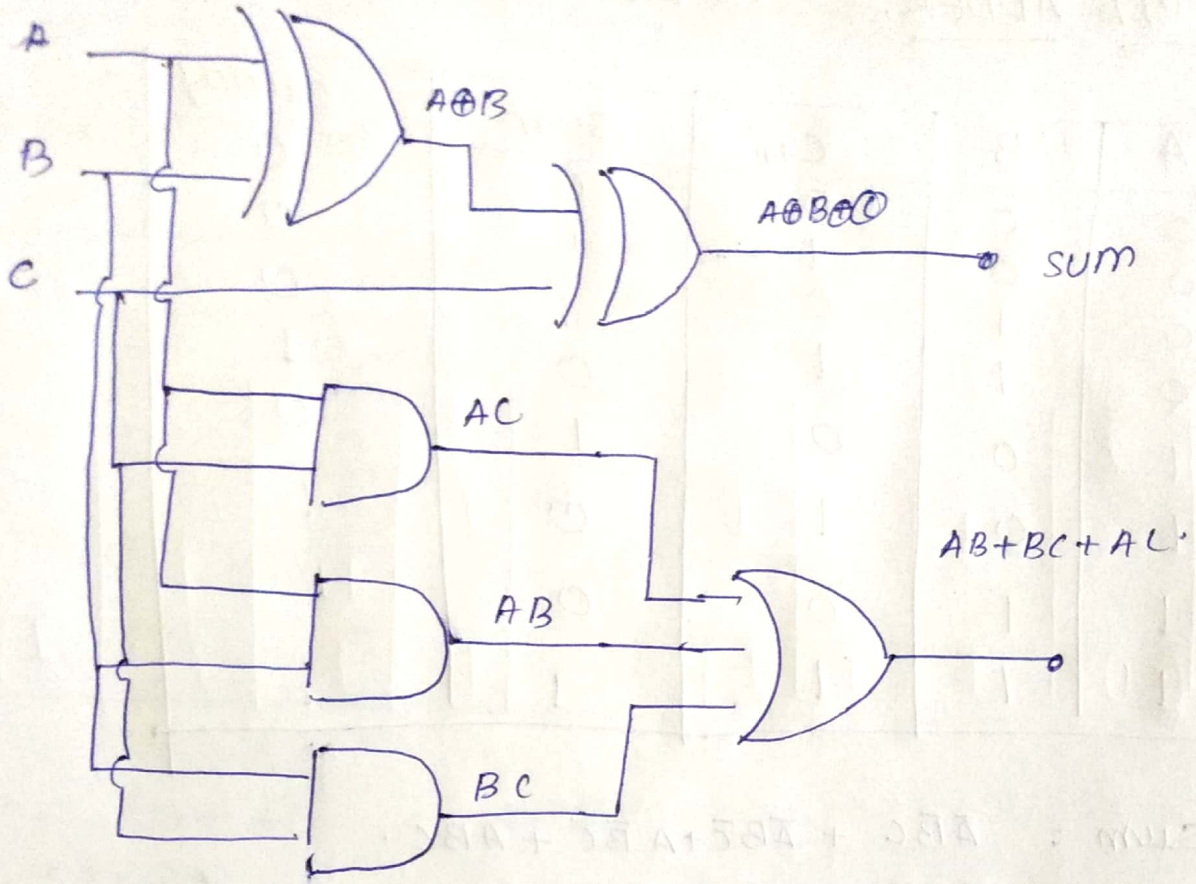
KMAP: sum

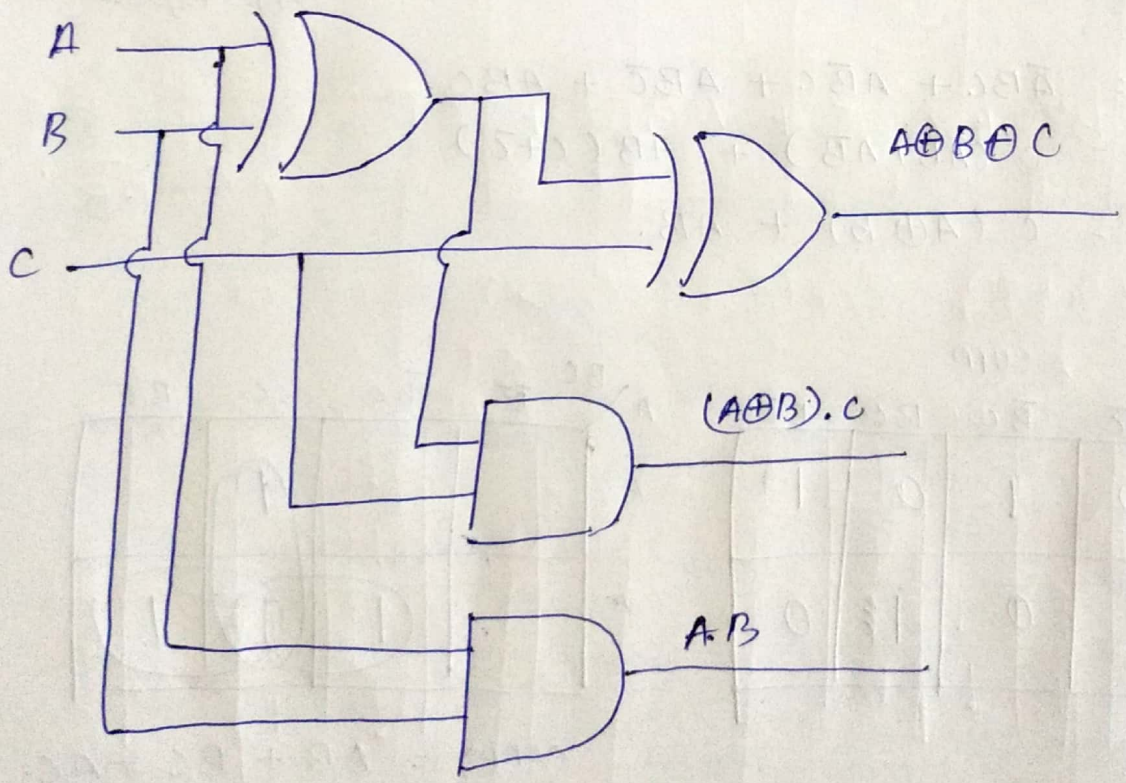
A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0	1	0	1
A	1	0	1	0

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$			1	
A		1	1	1

$$\text{CARRY} = AB + BC + AC.$$

Scanned by CamScanner





Scanned by CamScanner

Half Subtractor : combinational circuit (2 bits subtraction)

A	B	DIFFERENCE	BORROW
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

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

BORROW $= \bar{A}B$

K-map simplification

DIFFERENCE

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

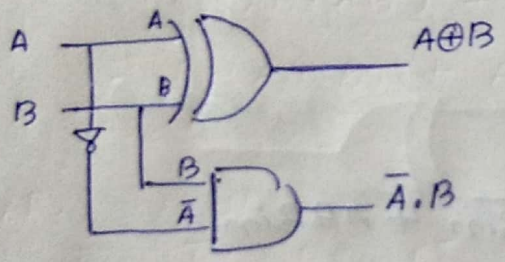
$= \bar{A}B + A\bar{B}$

$= A \oplus B$

BORROW

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

$= \bar{A}B$



Scanned by CamScanner

FULL SUBTRACTOR :

A	B	C_{in} (B_{in})	DIFFERENCE	BORROW (B_{out})
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

DIFFERENCE $= \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + ABB_{in}$

$B_{out} = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + \bar{A}BB_{in} + ABB_{in}$.

K-Map simplification:

A \ BB_{in}	$\bar{B}\bar{B}_{in}$	$\bar{B}B_{in}$	BB_{in}	$B\bar{B}_{in}$
$\bar{A}$		1		1
A	1		1	

$Diff = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + ABB_{in}$.

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

$= \bar{B}_{in}(A \oplus B) + B_{in}(A \odot B)$

$= \bar{B}_{in}(A \oplus B) + B_{in}(\overline{A \oplus B})$

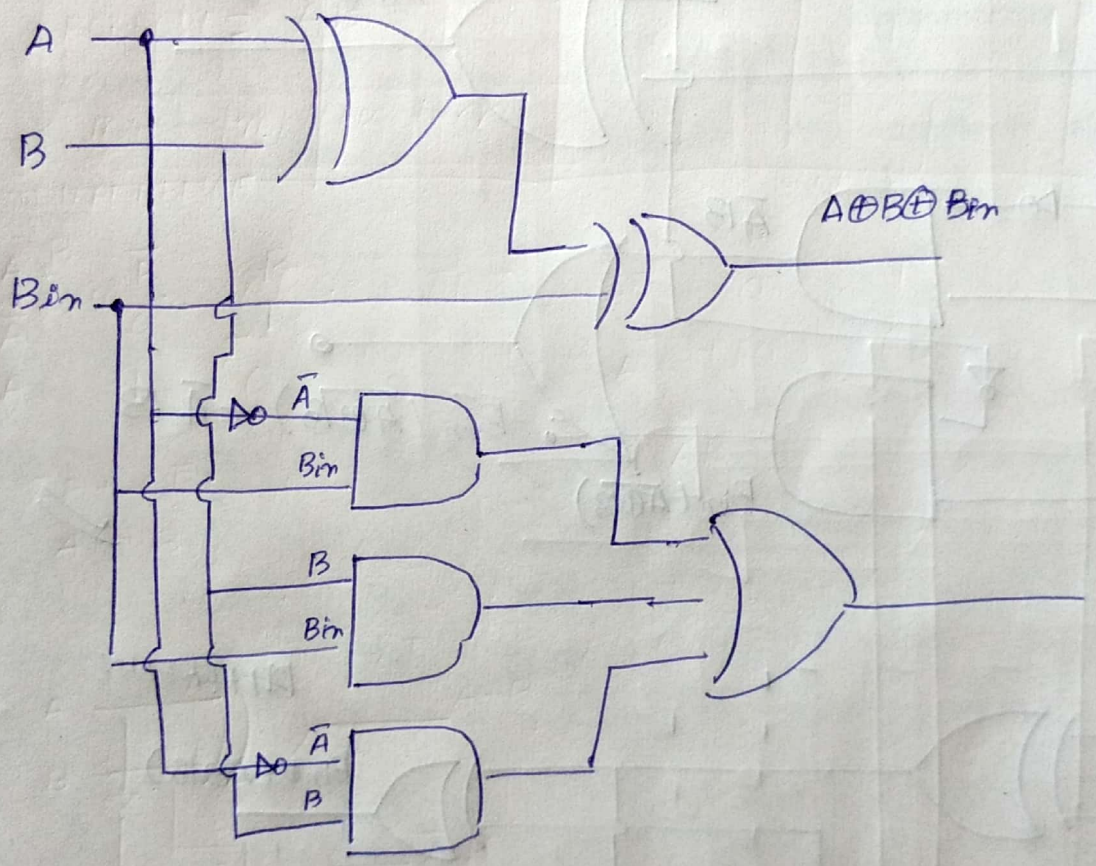
$= B_{in} \oplus (A \oplus B)$

Scanned by CamScanner

Borrow :

A \ $B B_{in}$	$\bar{B}\bar{B}_i$	$\bar{B}B_i$	BB_i	$B\bar{B}_i$
$\bar{A}$		1	1	1
A			1	

$$B_{out} = \bar{A}B_{in} + BB_{in} + \bar{A}B.$$



$$B_{out} = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + \bar{A}BB_{in} + ABB_{in}$$

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

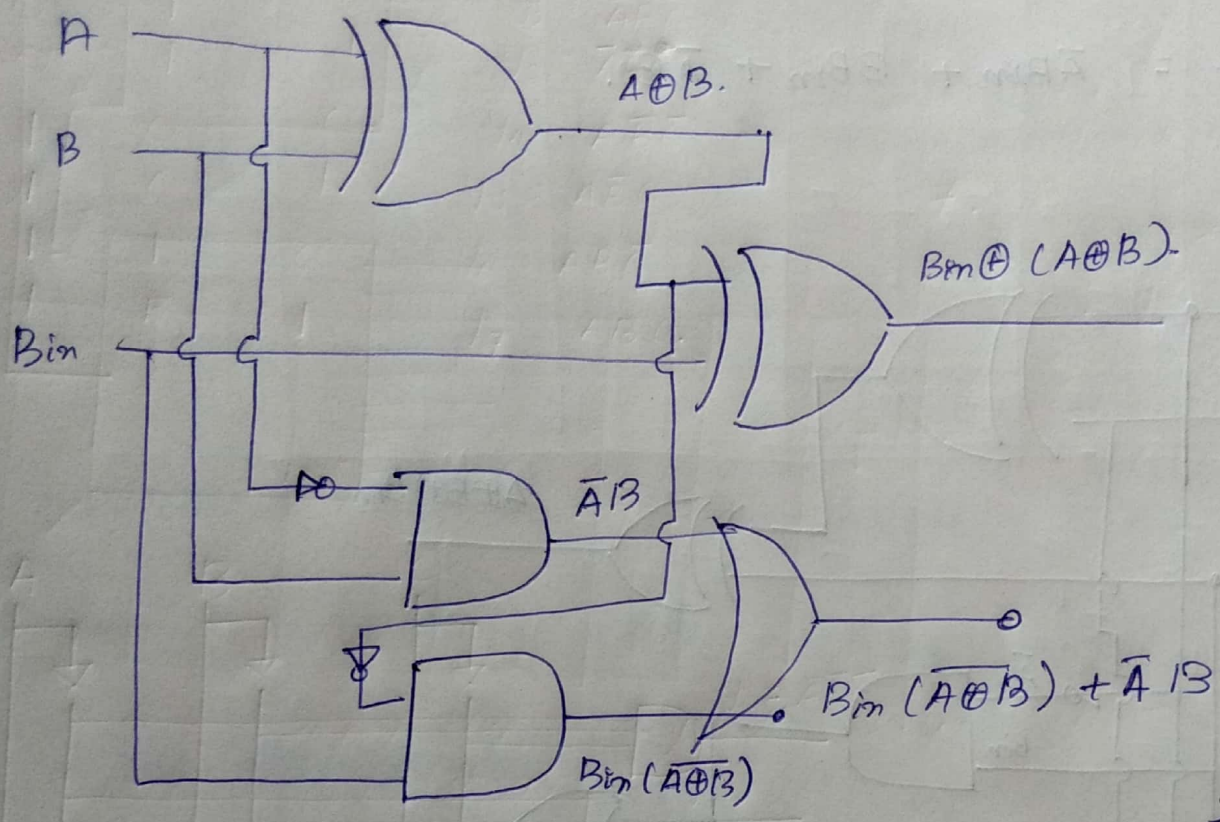
$$= B_{in}(A \odot B) + \bar{A}B.$$

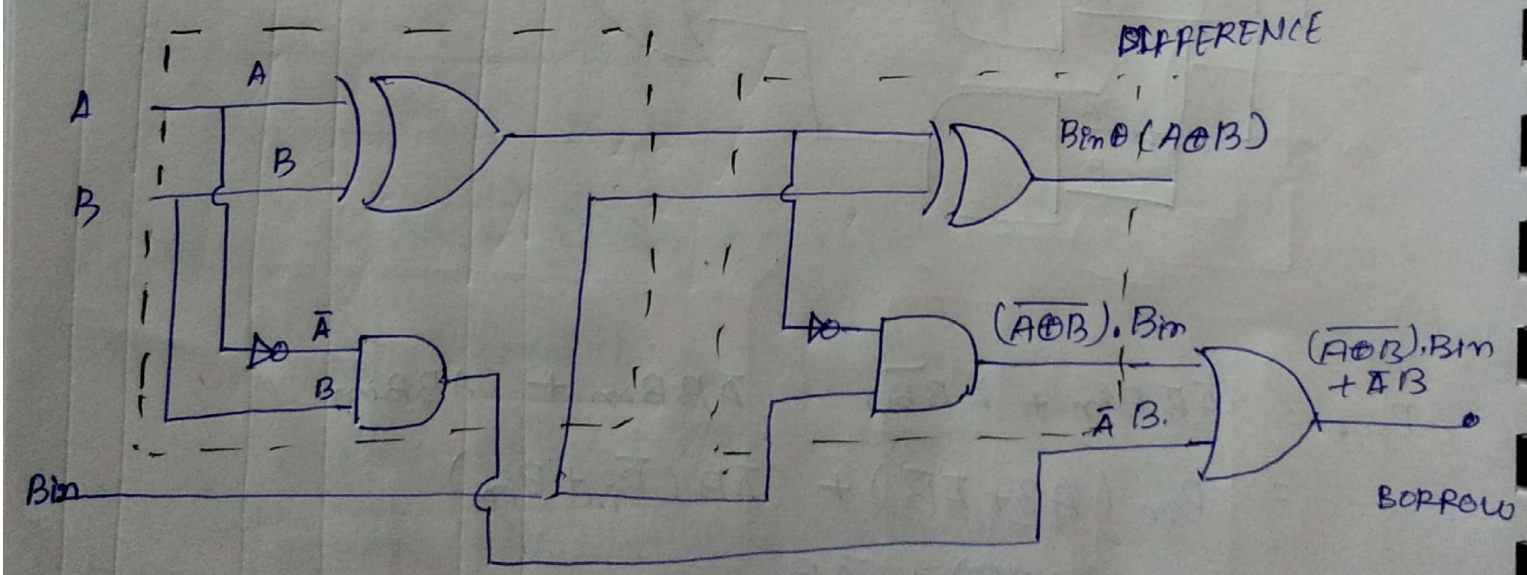
$$= B_{in}(\overline{A \oplus B}) + \bar{A}B.$$

Scanned by CamScanner

$D = B_{in} \oplus (A \oplus B)$

$B_{out} = B_{in}(\overline{A \oplus B}) + \bar{A}B$





$(\overline{A \oplus B}).B_{in} + \bar{A}B$

$= B_{in}(A \odot B) + \bar{A}B$

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

$= \bar{A}\bar{B}B_{in} + ABB_{in} + \bar{A}B$

Scanned by CamScanner

$= \bar{A}\bar{B}B_{in} + AB_{in} + \bar{A}B(1+B_{in})$

$= \bar{A}\bar{B}B_{in} + AB_{in} + \bar{A}B + \bar{A}BB_{in}$

$= B\,B_{in}(A+\bar{A}) + \bar{A}\bar{B}B_{in} + \bar{A}B.$

$= B\,B_{in} + \bar{A}\bar{B}B_{in} + \bar{A}B(1+B_{in})$

$= BB_{in} + \underline{\bar{A}\bar{B}B_{in}} + \bar{A}B + \underline{\bar{A}BB_{in}}$

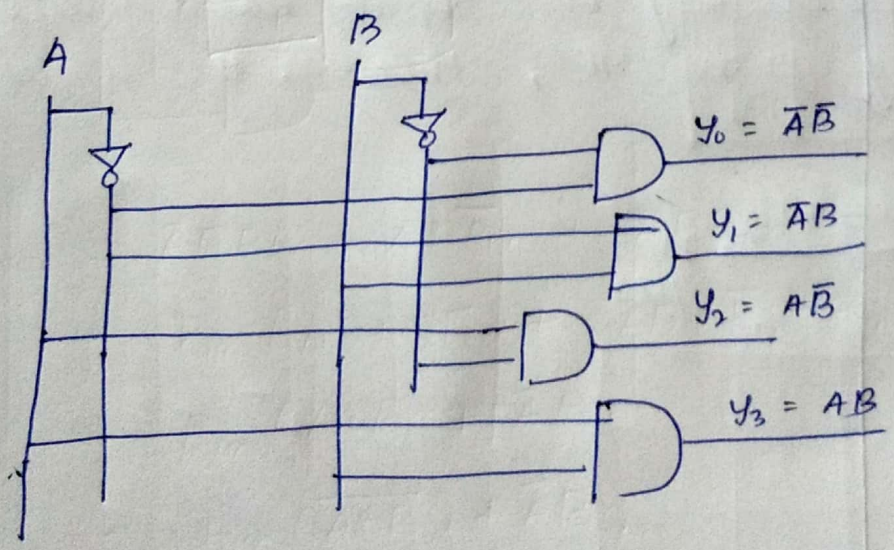
$= BB_{in} + \bar{A}B_{in}(B+\bar{B}) + \bar{A}B$

$= BB_{in} + \bar{A}B_{in} + \bar{A}B. \quad = B_{out}.$

Decoder: If we pass 'n' inputs (input lines) to a decoder then it will produce the output of 2^n output lines.

$$\underset{\text{I/ps}}{n} \longrightarrow \underset{\text{O/ps.}}{2^n}$$

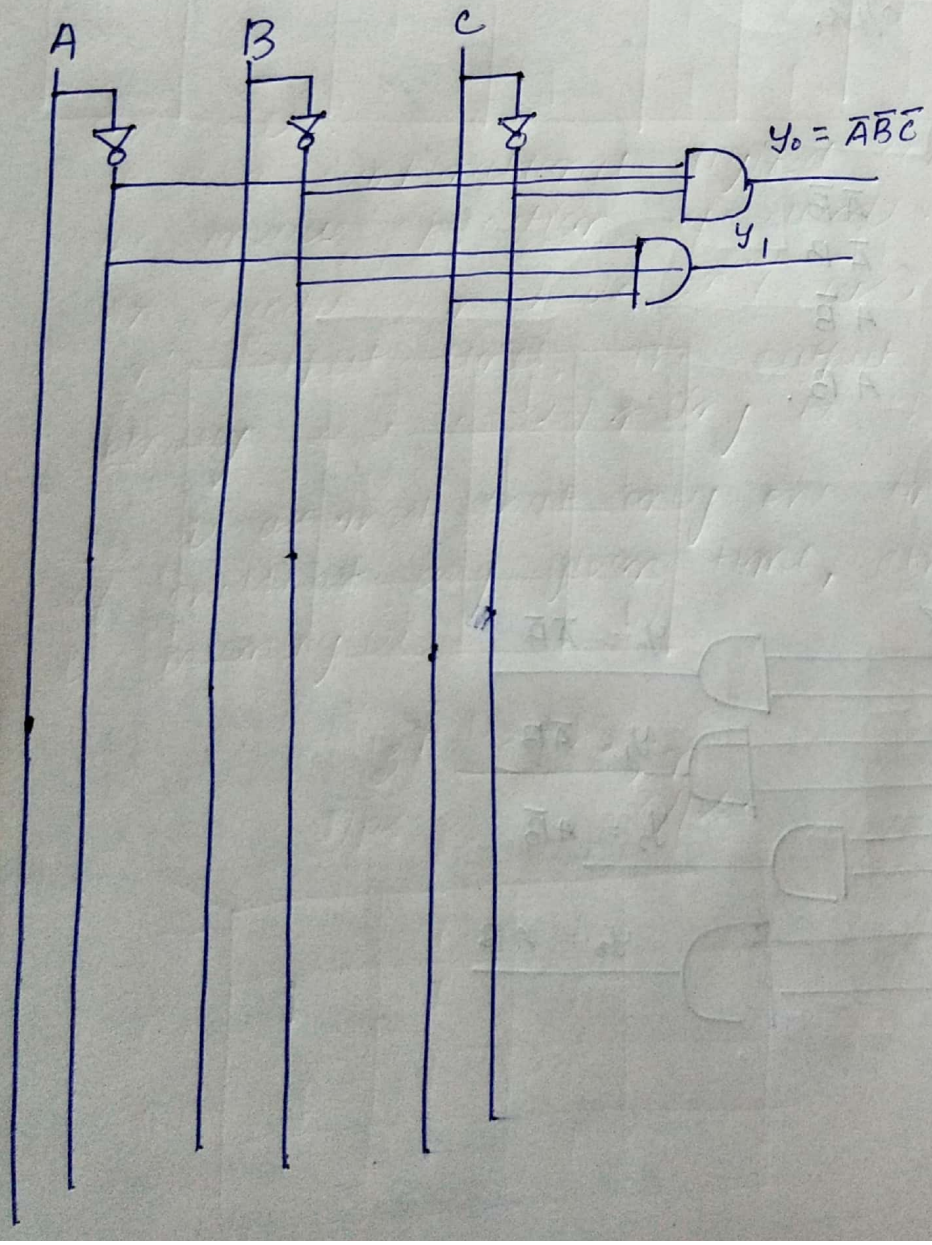
A	B	O/P.	
0	0	y_0	$\bar{A}\bar{B}$
0	1	y_1	$\bar{A}B$
1	0	y_2	$A\bar{B}$
1	1	y_3	$AB.$



Scanned by CamScanner

3 : 8 Line Decoder :

A	B	C	O/P
0	0	0	$Y_0 = \bar{A}\bar{B}\bar{C}$
0	0	1	$Y_1 = \bar{A}\bar{B}C$
0	1	0	$Y_2 = \bar{A}B\bar{C}$
0	1	1	$Y_3 = \bar{A}BC$
1	0	0	$Y_4 = A\bar{B}\bar{C}$
1	0	1	$Y_5 = A\bar{B}C$
1	1	0	$Y_6 = AB\bar{C}$
1	1	1	$Y_7 = ABC.$



Scanned by CamScanner

A Decoder is a combinational circuit that converts binary information from 'n' input lines to a maximum of '2^n' unique output lines.

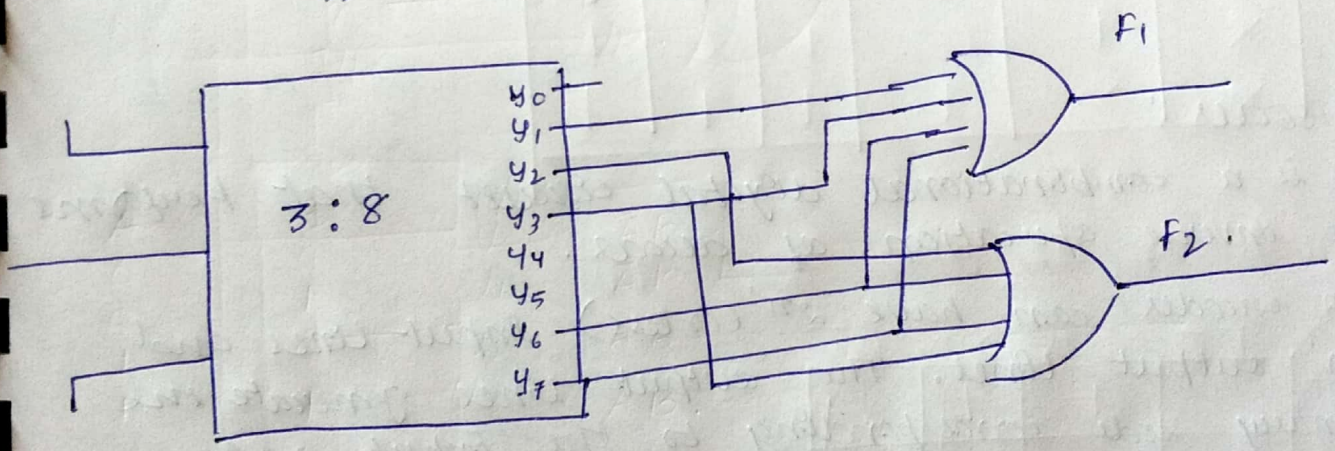
Ans. Implement the following boolean ~~circuit~~ functions using 3:8 decoders and external gates.

$F_1(A,B,C) = \Sigma m(1,3,5,7)$

$F_2(A,B,C) = \Sigma m(2,3,6,7)$

$$F_1 = \underset{y_1}{\bar{A}\bar{B}C} + \underset{y_3}{\bar{A}BC} + \underset{y_5}{A\bar{B}C} + \underset{y_7}{ABC}.$$

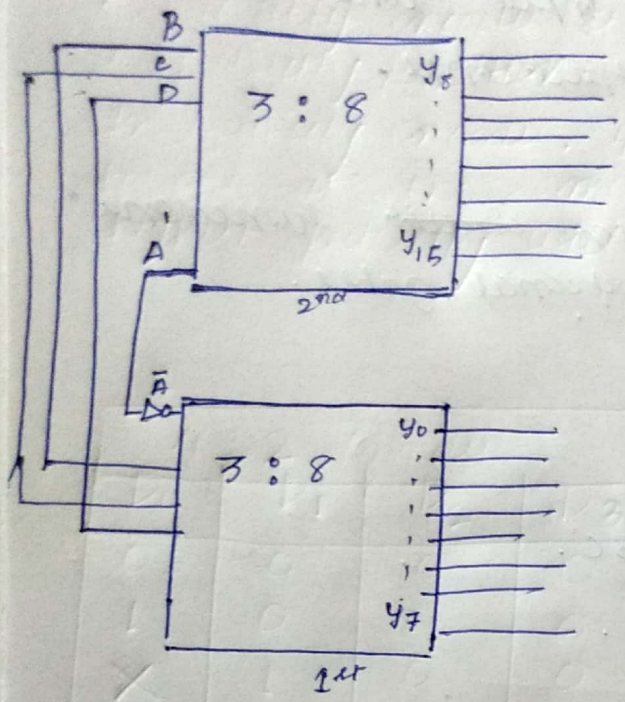
$$F_2 = \underset{y_2}{\bar{A}B\bar{C}} + \underset{y_3}{\bar{A}BC} + \underset{y_6}{AB\bar{C}} + \underset{y_7}{ABC}.$$



Scanned by CamScanner

Design:

4:16 decoder using 3:8 decoder



A will act as Enable

i.e for 1st decoder (3:8) A will give 0 ($\bar{A}$) which will generate 0 - 7.

for 2nd decoder (3:8) A will give 1 (A) which will generate 8 - 15.

Encoder:

It is a combinational digital circuit that performs the inverse operation of decoder.

An encoder can have '2^n' (or less) input lines and 'n' output lines. The output lines generate the binary code corresponding to the input value.

It is assumed that only one input has a value of one (1) at any given time, otherwise the circuit is meaningless

$$\underset{\text{I/ps}}{2^n} \longrightarrow \underset{\text{O/ps.}}{n}$$

Scanned by CamScanner

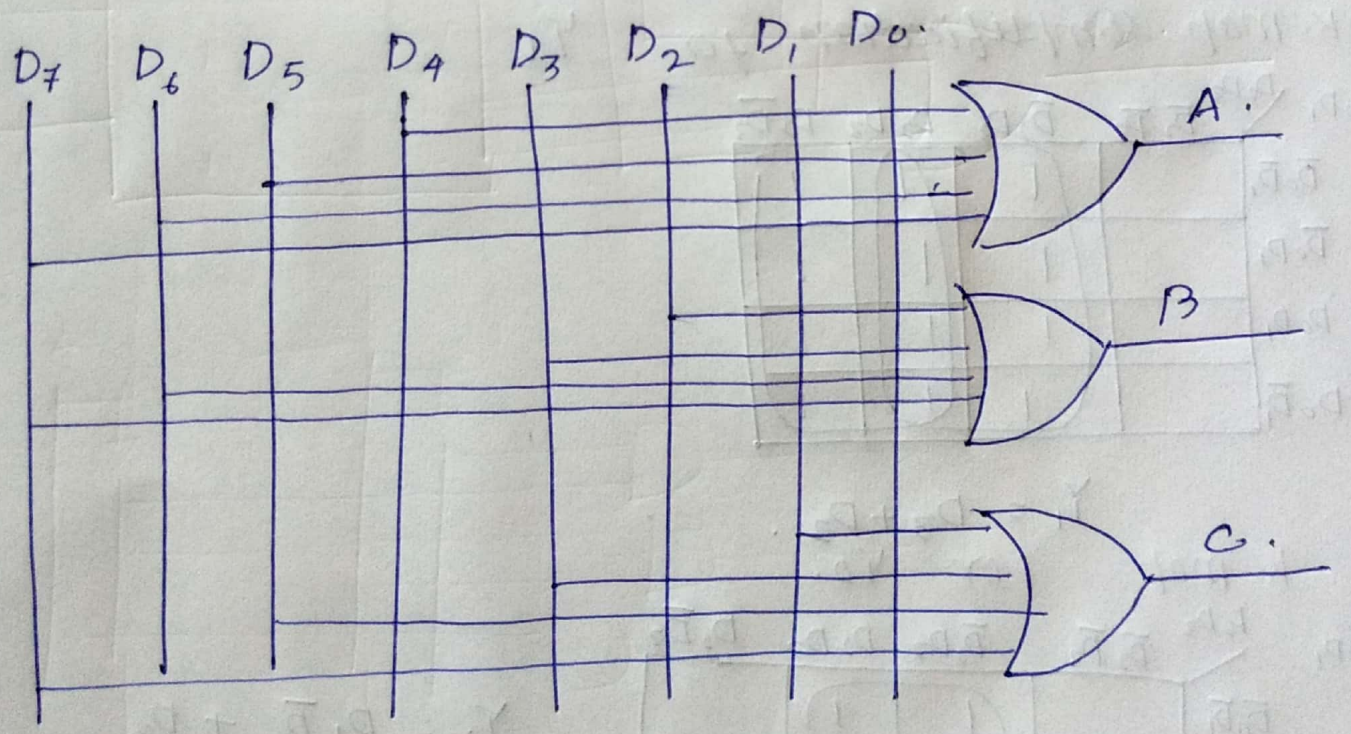
8:3 encoder:

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
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

$A = D_4 + D_5 + D_6 + D_7$

$B = D_2 + D_3 + D_6 + D_7$

$C = D_1 + D_3 + D_5 + D_7$.



Scanned by CamScanner

Priority encoder :

A priority encoder is an encoder circuit than includes a priority function.

In Priority encoder, if two or more inputs are equal to one at the same time, the input having the highest priority will take the precedence.

4 : 2 Encoder				O/P		Valid
D_0	D_1	D_2	D_3	Y_1	Y_0	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
~~X~~	1	0	0	0	1	1
X	X	1 D_2	0 $\bar{D}_3$	1	0	1
X	X	X	1 D_3	1	1	1

K-map simplification for Y_1 :

D_0D_1 \ D_2D_3	$\bar{D}_2\bar{D}_3$	$\bar{D}_2D_3$	D_2D_3	$D_2\bar{D}_3$
$\bar{D}_0\bar{D}_1$		1	1	1
$\bar{D}_0D_1$		1	1	1
D_0D_1		1	1	1
$D_0\bar{D}_1$		1	1	1

$$Y_1 = D_3 + D_2 .$$

K-map for Y_0.

D_0D_1 \ D_2D_3	$\bar{D}_2\bar{D}_3$	$\bar{D}_2D_3$	D_2D_3	$D_2\bar{D}_3$
$\bar{D}_0\bar{D}_1$		1	1	
$\bar{D}_0D_1$	1	1	1	
D_0D_1	1	1	1	
$D_0\bar{D}_1$		1	1	

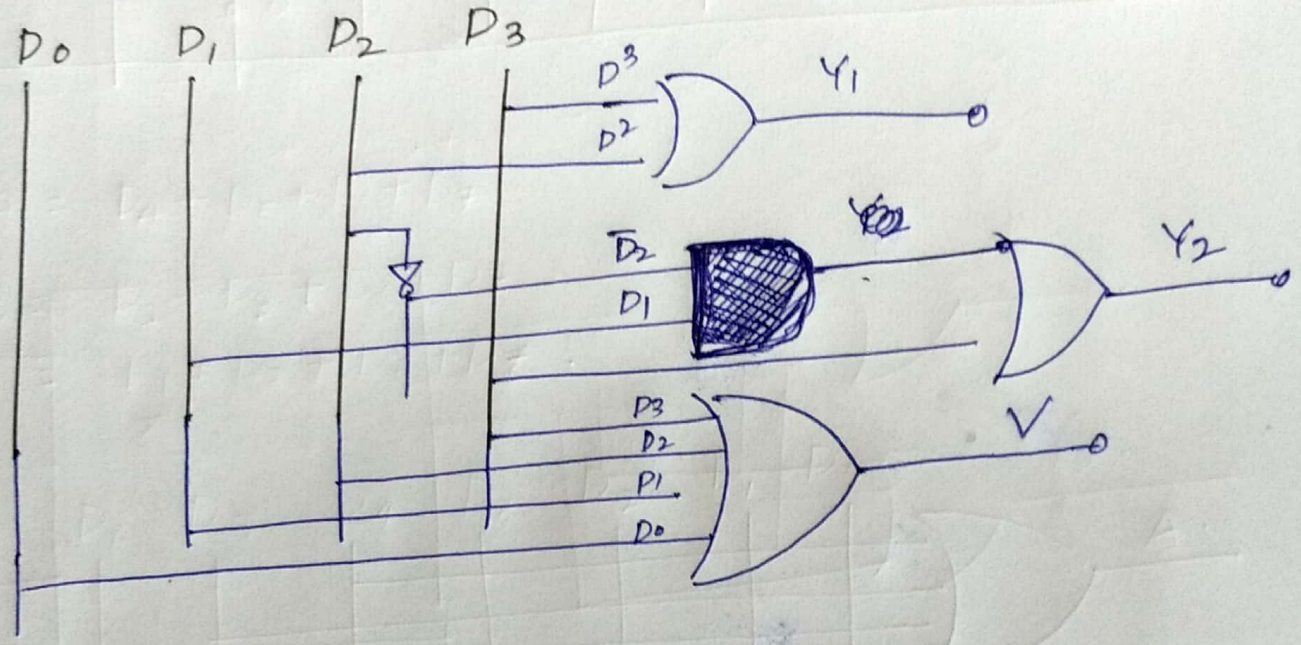
$$Y_2 = D_1\bar{D}_2 + D_3$$

Scanned by CamScanner

K-map for valid:

	$\overline{D_2}\,\overline{D_3}$	$\overline{D_2}D_3$	D_2D_3	$D_2\overline{D_3}$
$\overline{D_0}\,\overline{D_1}$	0	1	1	1
$\overline{D_0}D_1$	1	1	1	1
D_0D_1	1	1	1	1
$D_0\overline{D_1}$	1	1	1	1

$$V = D_2 + D_3 + D_1 + D_0$$



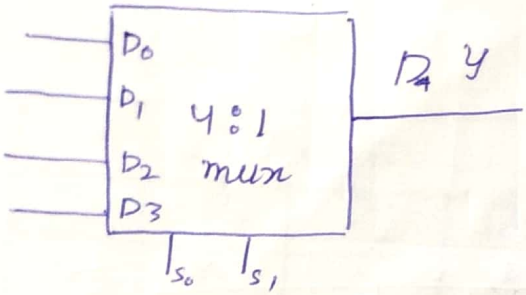
Scanned by CamScanner

Multiplexer: (many to one)

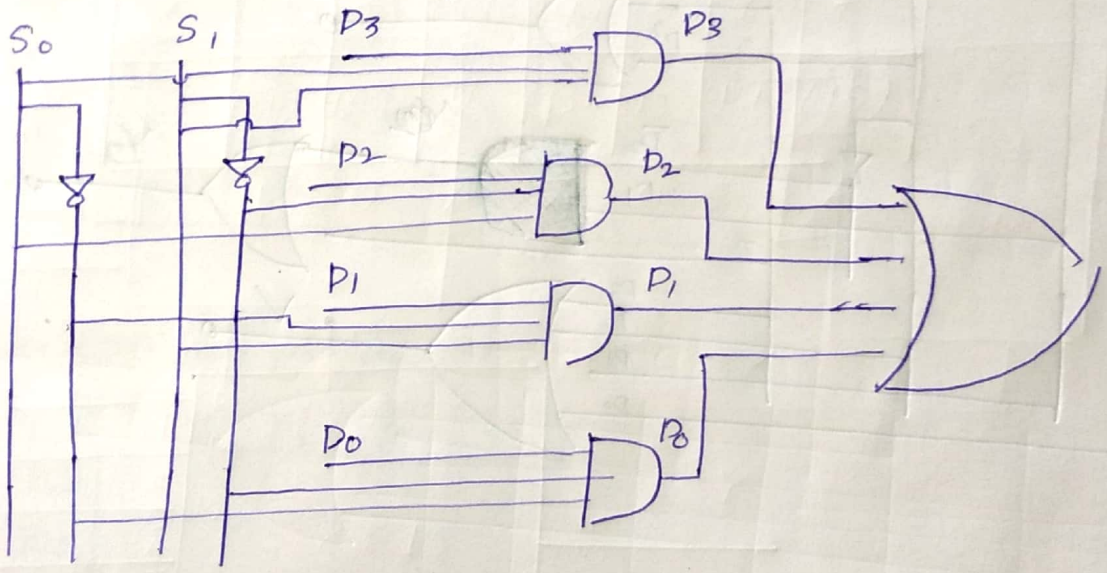
$2^n \rightarrow 1$

i/ps o/p

4:1 mux.

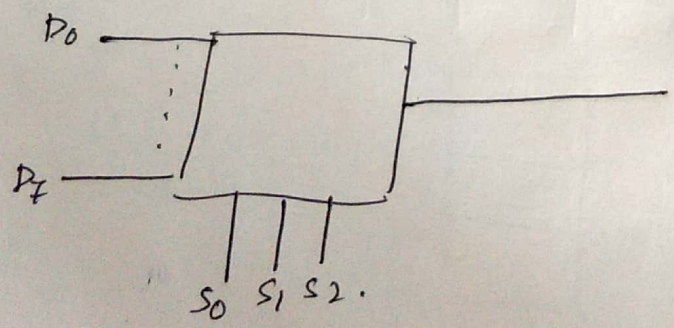


S_0	S_1	y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

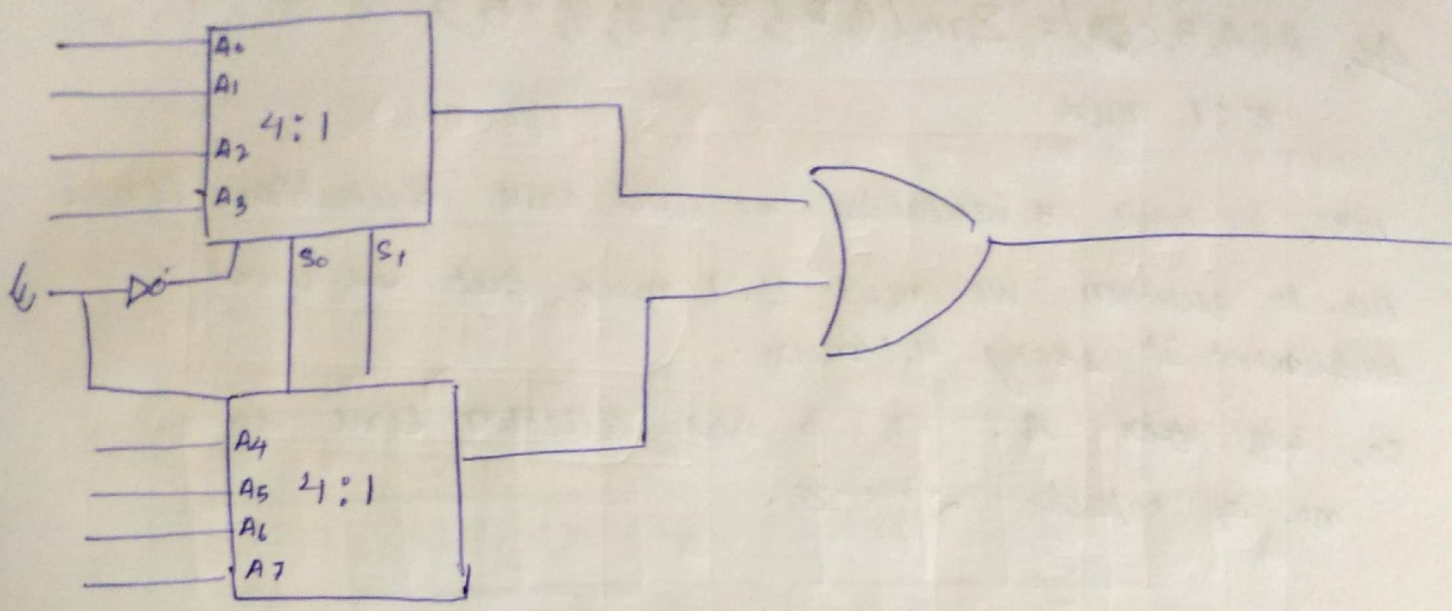


Q Implement 8:1 mux using 4:1 mux. (2)

8:1 mux.



Scanned by CamScanner



Ans $F(A, B, C) = \Sigma m(1,3,5,6)$.

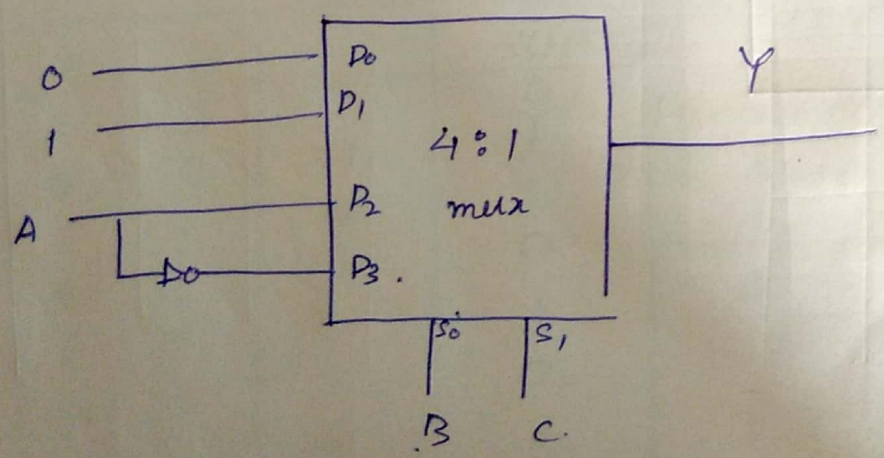
Fixed. S_0, S_1.

here variable will act as ~~selection~~ ~~Input~~ selection line.

Ans. 4:1 mux

	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
	0	1	A	$\bar{A}$

when there is no circuit = 0
both " = 1



Scanned by CamScanner

4. $F(A,B,C,D) = \Sigma m(0,1,3,4,8,9,15)$.

8 : 1 mux

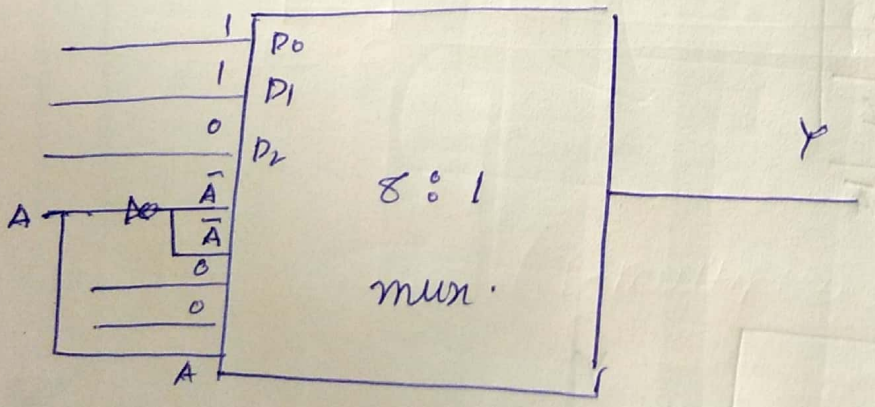
Here, we have 3 variables we will have 3 selection lines

Acc. to question we need 16:1 mux but we will implement it using 4:1 mux.

So, we ~~fix~~ A. B C D are selection line so

no. of inputs $2^3 = 8$.

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



Scanned by CamScanner

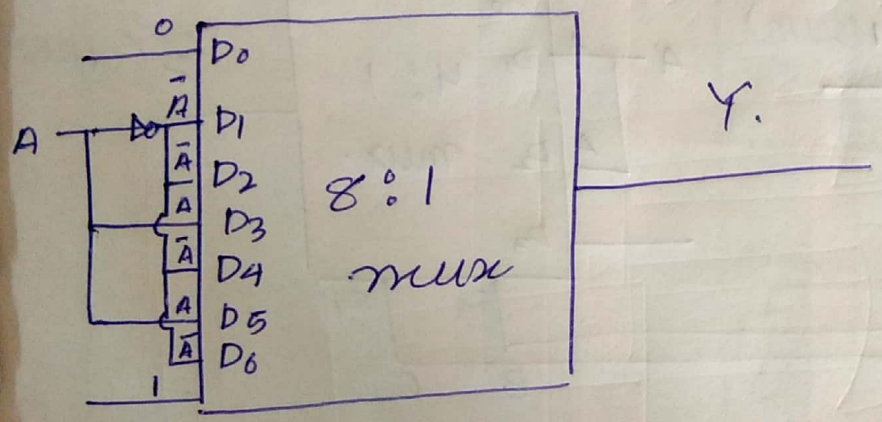
Ans. $F(A, B, C, D) = \pi M(0, 3, 5, 8, 9, 10, 12, 14)$.

Negation of M will be m

$\Rightarrow \Sigma m(1, 2, 4, 6, 7, 11, 13, 15)$.

We fix A.

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



Ans. Implement the full adder using two 4:1 mux?

	A	B	C	sum	CARRY
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	1
3	0	1	1	0	0
4	1	0	0	1	1
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

Scanned by CamScanner

SUM = $\Sigma m(1, 2, 4, 7)$. $f(A, B, C)$

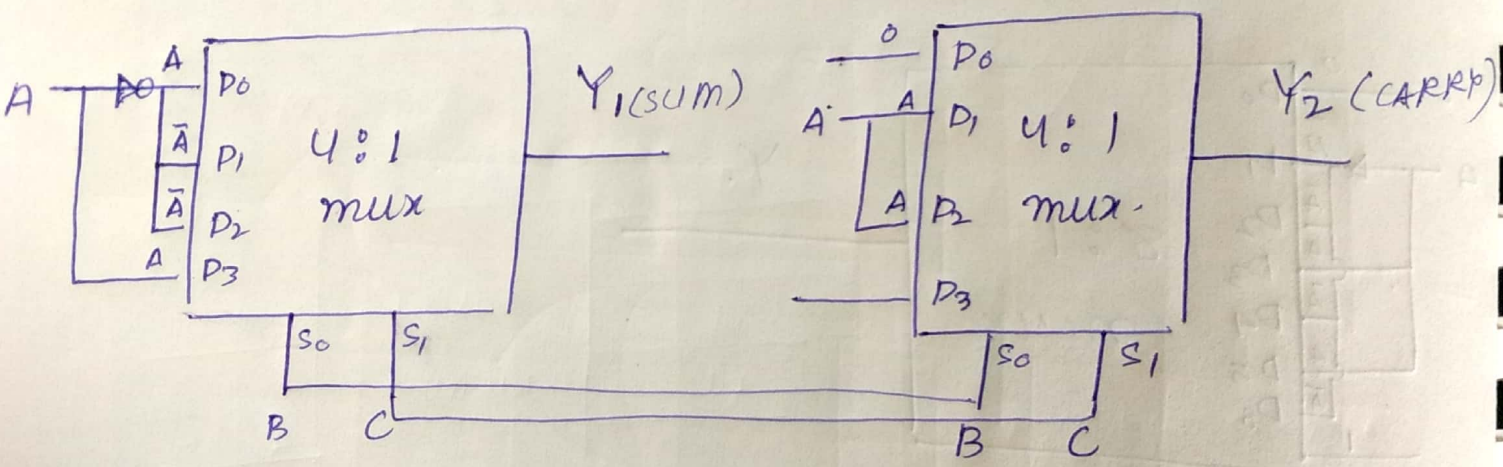
CARRY = $\Sigma m(3, 5, 6, 7)$. $f(A, B, C)$.

SUM

	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
	A	$\bar{A}$	$\bar{A}$	A

CARRY

	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
	0	A	A	1



Demultiplexer (D-multiplexer).

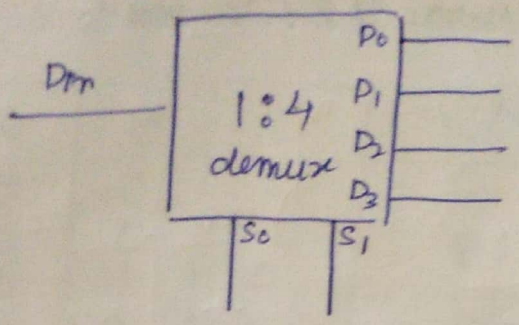
A demultiplexer is a circuit that recieves information on a single line and transmitt this information on one of 2^n possible output lines.

The selection of specific output line is controlled by the values of n selection line.

$1 \longrightarrow 2^n$ selection line

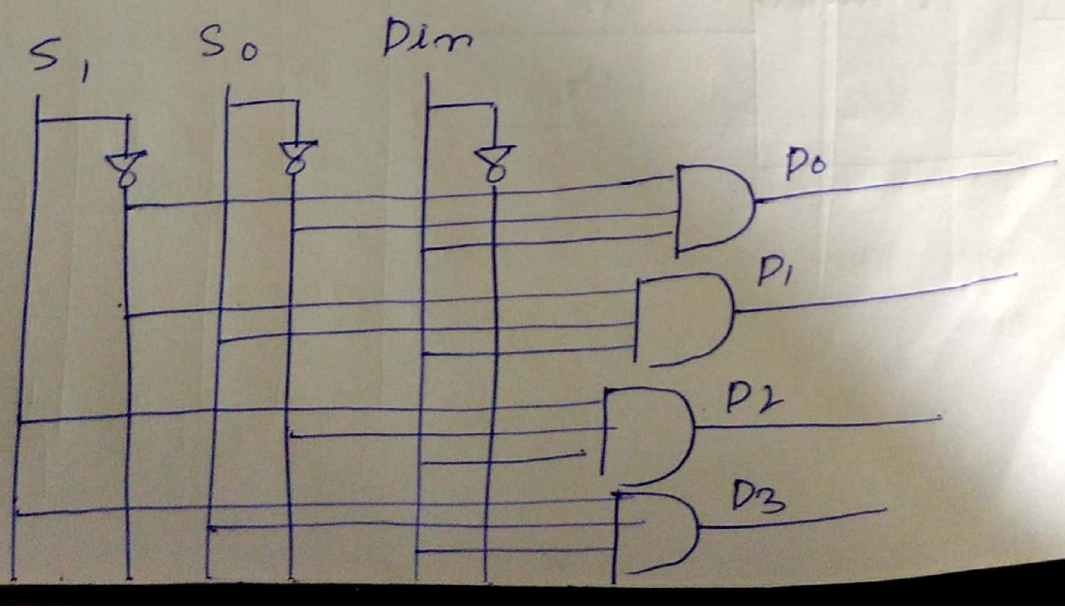
input outputs. (n).

Scanned by CamScanner



here selection lines will decide where the output will appear.
Din will alter the result at output side.

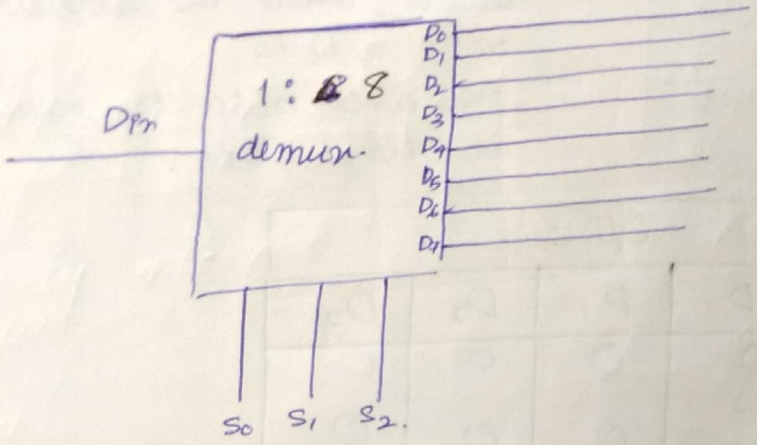
	VPs			O/Ps.			
	S_1	S_0	Din	D_0	D_1	D_2	D_3
D_0	0	0	0	0	0	0	0
	0	0	1	1	0	0	0
D_1	0	1	0	0	0	0	0
	0	1	1	0	1	0	0
D_2	1	0	0	0	0	0	0
	1	0	1	0	0	1	0
D_3	1	1	0	0	0	0	0
	1	1	1	0	0	0	1

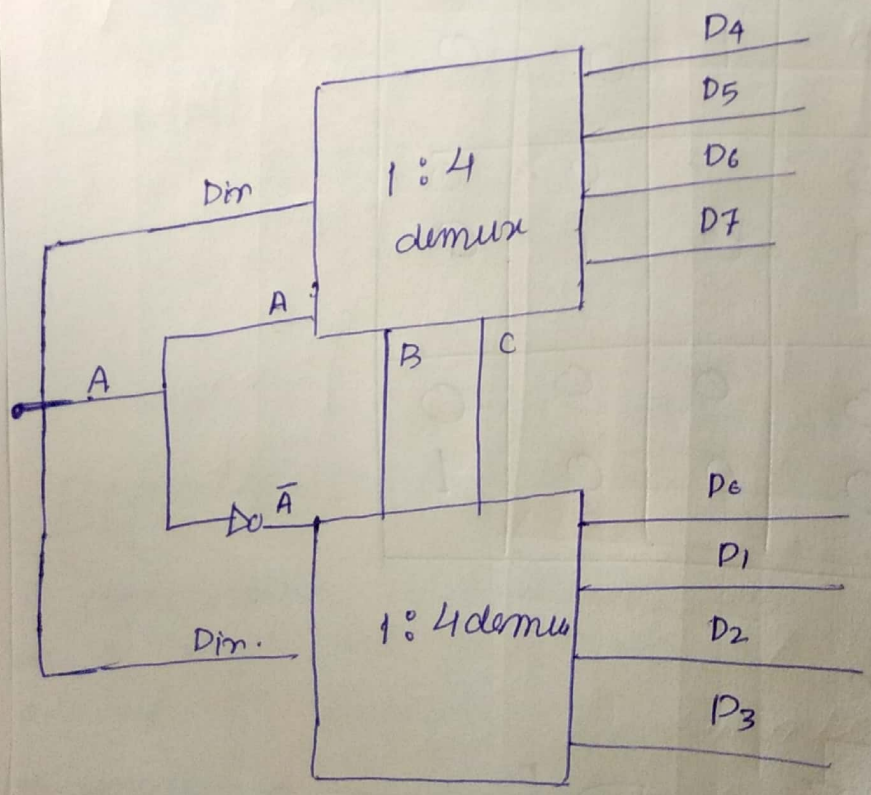


Scanned by CamScanner

Implement 1:8 demux using two 4:1 demux.

1:8 → n = 3 signal lines.
1 inputs.





A act as enable.

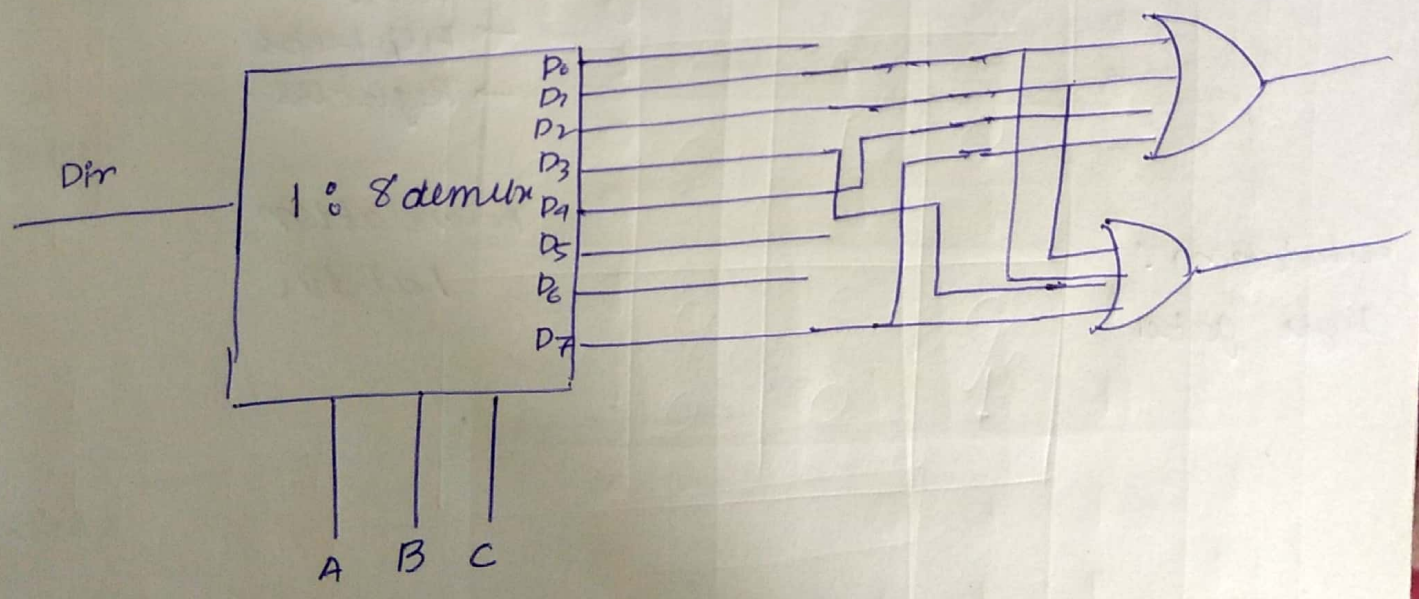
Scanned by CamScanner

Implement full subtractor using demultiplexer ?

A	B	C	SUB	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

SUB = $\Sigma m(1, 2, 4, 7)$

BORROW = $\Sigma m(1, 2, 3, 7)$.



Scanned by CamScanner

Difference b/w sequential and combinational circuit.

Combinational	Sequential
Output variable at all times are dependent on the input variable	They depend not only on the present input variable but also on the past history of these input variable
Memory unit not required	Required
faster	slower beoz of the involvement of memory.
Easy to design.	complexity harder to design.
Ex. parallel adder	Ex. Serial adder. → Latches → Flip flops → Registers.
Foundation: Logic gates	Foundation: Latches

Latches are bi-stable elements which are used as building blocks of most sequential circuits

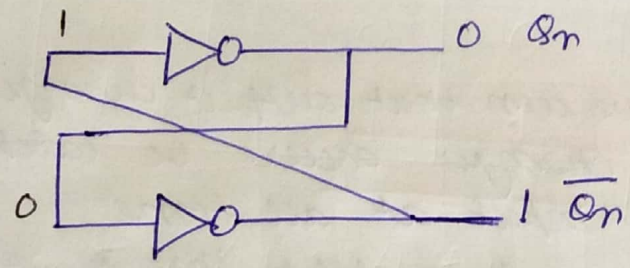
The main diff. b/w latches & flipflops is in the methods used for changing their state.

Scanned by CamScanner

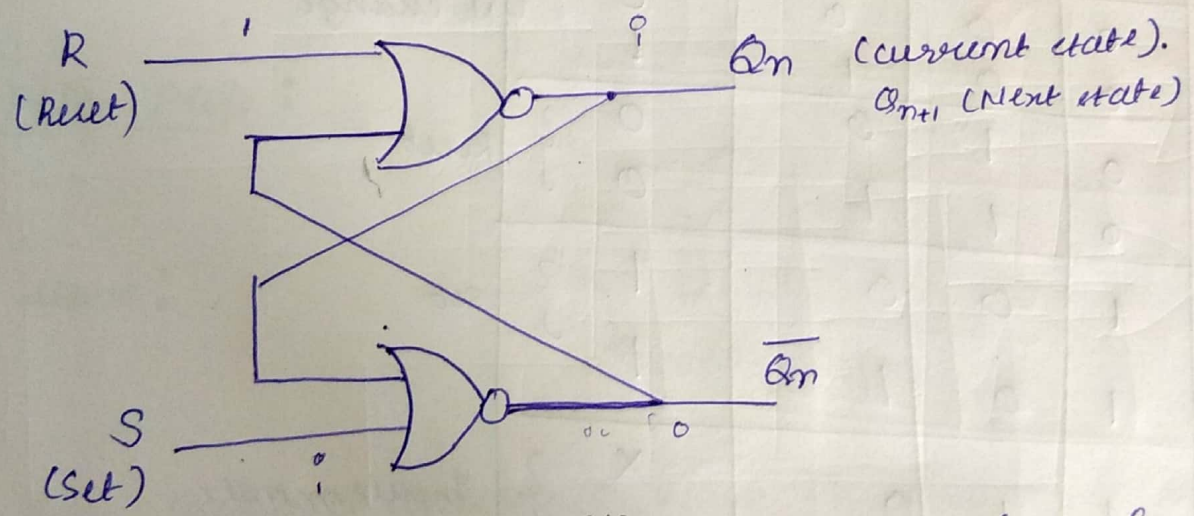
→ Flip flops normally samples its inputs and changes its output at times determined by clocking signal.

→ Latch checks all of its inputs continiously & changes its output accordingly at any time independent of the clocking signal.

Latch.



S.R Latch :



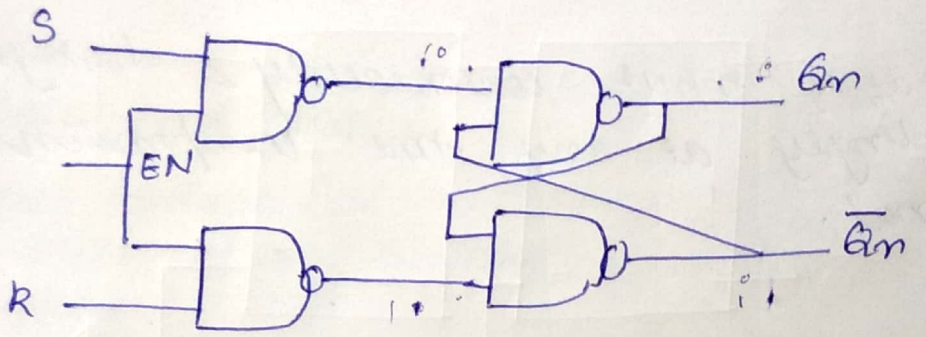
O/P.

Qn Qn+1
0

S	R	Qn	Qn+1	
0	0	0	0	no change
0	0	1	1	
0	1	0	0	Reset.
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	X	Indeterminant state.
1	1	1	X	

Scanned by CamScanner

Gated S-R Latch :



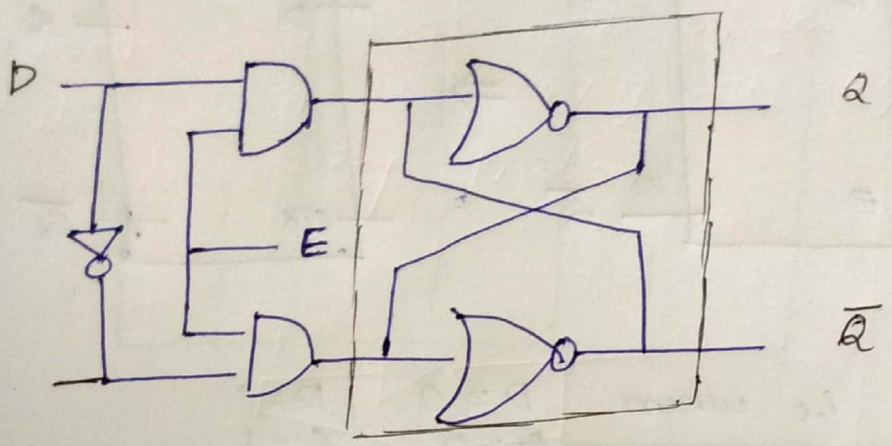
In the S-R Latch we have seen that output changes occurs quickly after the input changes occurs i.e latch is sensitive to S-and-R input at all time, However it can be used to create a latch that is sensitive to these input only one the Enable is active.

EN	S	R	Qn	Qn+1	
1	0	0	0	0	No change
1	0	0	1	1	
1	0	1	0	0	Reset
1	0	1	1	0	
1	1	0	0	1	set
1	1	0	1	1	
1	1	1	0	X	Indeterminate.
1	1	1	1	X	
0	X	X	0	0	copy previous state.
0	X	X	1	1	

Scanned by CamScanner

D - Latch :

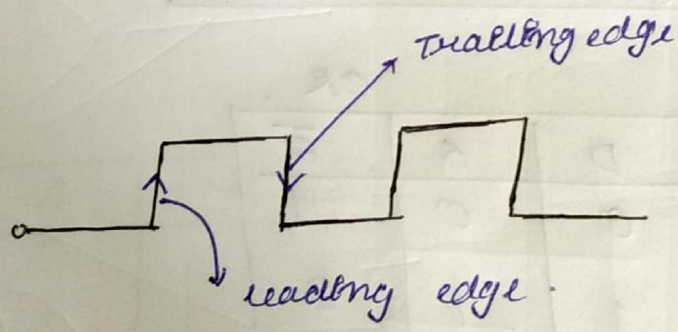
A latch will not enter on the indeterminant state



E	D	Q	$\overline{Q}$
0	X	0	0
1	1	1	0
1	0	0	1

Flip Flop :

Clock :



Edge Triggered circuits : When the output is optained on b/w the high or low signal.
i.e during leading or Trailing edge are called edge biggred circuits. Ex. Flip flops

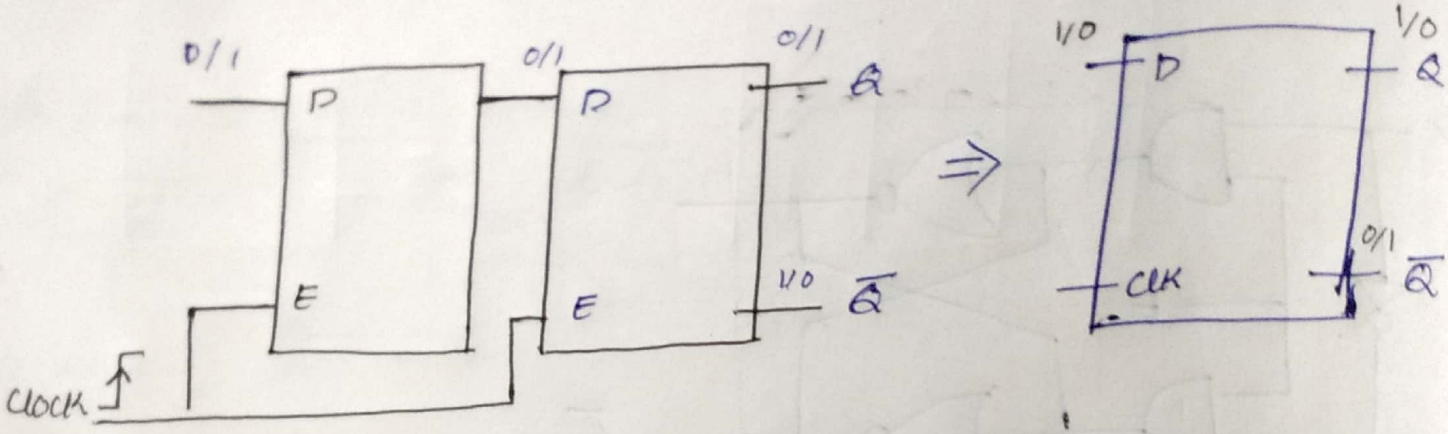
Level Triggered circuits : When the output is optained when the signal is either high or low.
then it is level biggred circuits.

Ex. Latch.

Scanned by CamScanner

D - Flip Flop :

Here Enable is high when clock is leading edge.



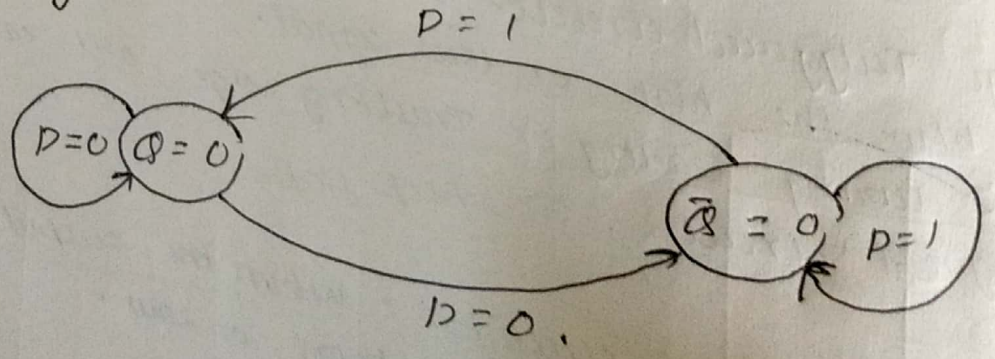
It is same as D-Latch i.e when $D = 0$, $Q = 0$ | $D = 1$, $Q = 1$

INPUT			OUTPUT	
CLK / E	D	Q	Q_{next}	$\bar{Q}_{next}$
0	X	0	0	1
1	X	0	0	1
↑	1	X	1	0
↑	0	X	0	1

OR

CLK	D	Q	$\bar{Q}$
↑	0	0	1
↑	1	1	0

State Diagram

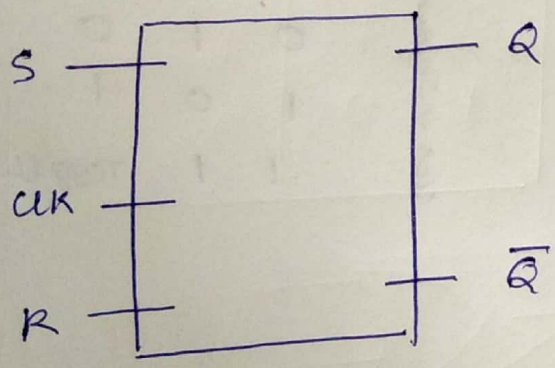


Scanned by CamScanner

Excitation Table :

Q	Qnext	D
0	0	0
0	1	1
1	0	0
1	1	1

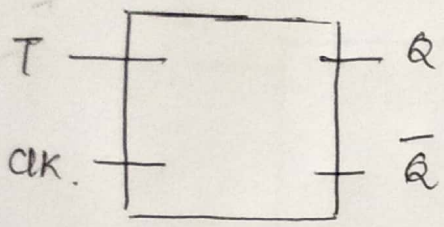
S-R Flip Flops :



S	R	Q	Qnext
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

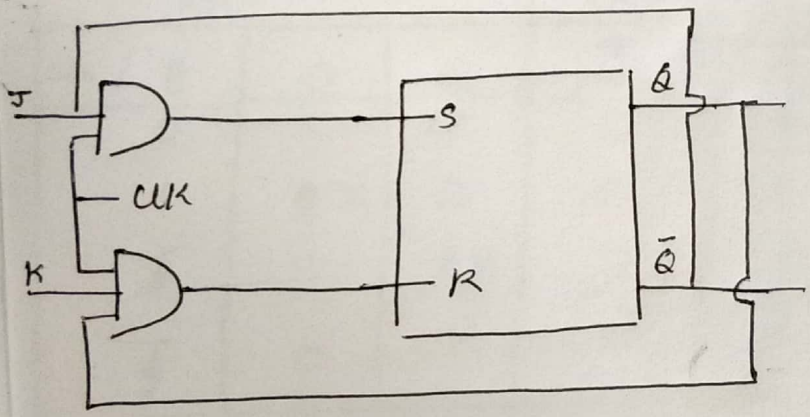
Scanned by CamScanner

T- Flip Flop : (Toggle → switching)



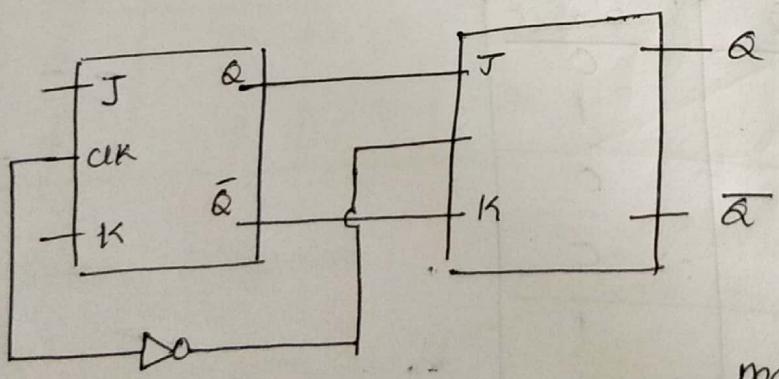
T	Q	$\bar{Q}$
0	1	0
1	0	1

JKSR Flip Flop.

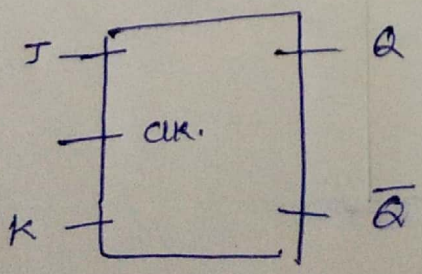


Clock is Triggered.

clk	J	K	Q
	0	0	Hold
	0	1	0
	1	0	1
	1	1	Toggle



master.



slave.

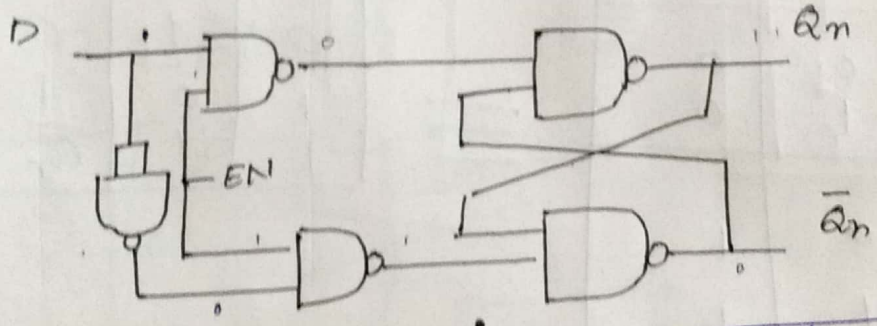
Scanned by CamScanner

Gated D-Latch:

	S	R	Q_{n+1}
1	0	0	No change
2	0	1	0 (Reset)
3	1	0	1 (Set)
4	1	1	* (Indeterminate)

In SR Latch 1 an 4 conditions are not required as these condition don't have practicle implementation.

∴ In D latch we don't have any condition of ① & ④ i,e [00] and [1 1]



D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

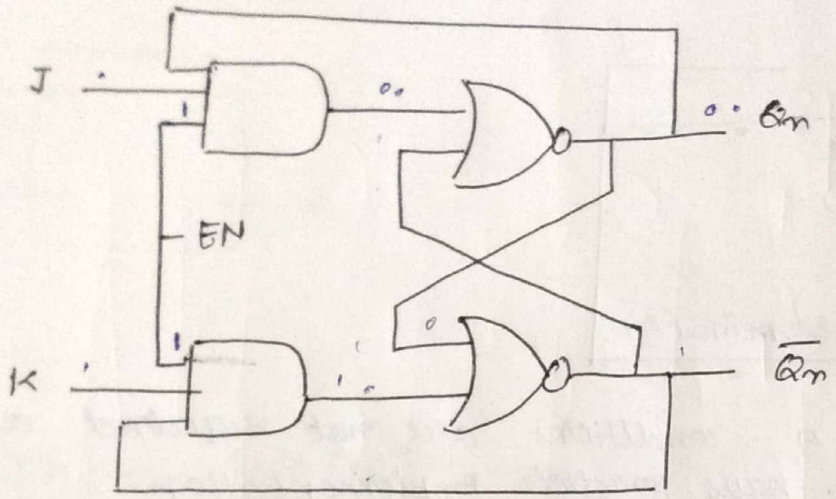
≡

EN	D	Q_{n+1}
1	0	0
1	1	1
0	X	Q_n

* Looking at the state table of S.R. Latch we can realize that when both inputs are same the output either doesn't change or it is indeterminant. In many practicle applications these input conditions are not required. Hence, in D-Latch they are avoided by making them complement of each other.

Scanned by CamScanner

J.K Latch :



J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

≡

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

J.K. Latch avoids the indeterminant input condition of SR Latch by connecting $\overline{Q_n}$ and Q_n outputs to the J.K. excitation input.

JK Latch circuit is not practical. In this circuit the output is fed back the input therefore, change on the output results change on the input. Due to this, in the +ve half of clk pulse if J & K are both high that output Toggles continuously

K-map simplification:

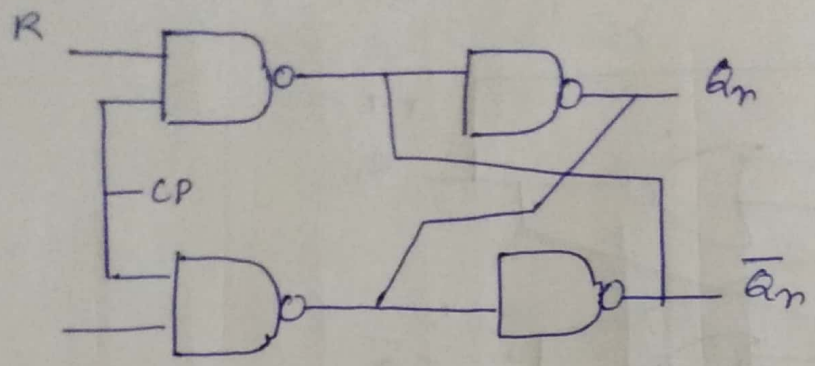
	JK	$J\overline{K}$	$\overline{J}K$	JK	$J\overline{K}$
Q_n 0	$\overline{Q_n}$	0	0	1	1
1	Q_n	1	0	0	1

$\cancel{J\overline{Q_n} + J\overline{K}\overline{Q_n}}$

$= J\overline{Q_n} + \overline{K}Q_n$

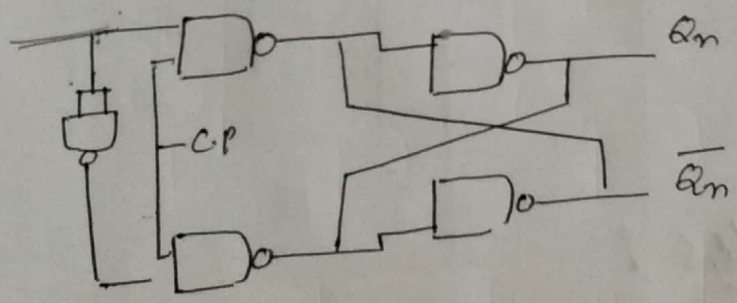
Scanned by CamScanner

S R Flip Flop :



S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	*

D - Flip Flop :



D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

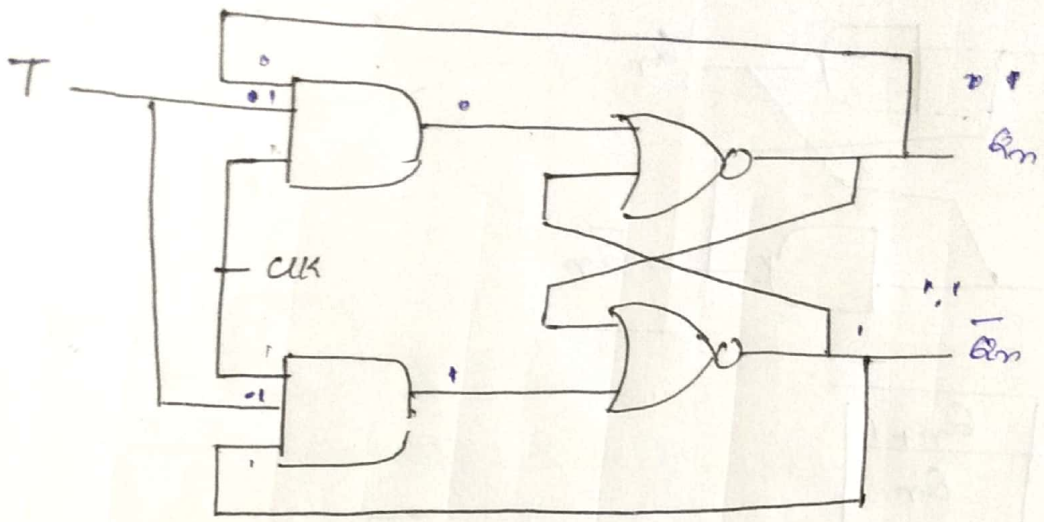
≡

D	Q_{n+1}
0	1
1	0

This condition is called as Race around condition.
(To avoid this condition we built J.K. flip flop)

Scanned by CamScanner

→ T Flip Flop (Toggle Flip Flop)



T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

≡

T	Q_{n+1}
0	Q_n
1	$\overline{Q_n}$

Excitation Tables :

(here Q_n & Q_{n+1} is given we will find inputs)

The table that lists the required inputs for a given change of state are called an excitation table.

1. S.R excitation table.

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	X

Q_n	Q_{n+1}	S	R
1	0	1	0

Scanned by CamScanner

2. JK Excitation table.

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

Q_n	Q_{n+1}	J	K
0	0	0	1
0	1	0	0
1	1	1	0
1	0	0	1

Q_n	Q_{n+1}	J	K
1	0	0	1
		1	1
1	1	0	1
		1	0

3. D - Excitation table

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

D	Q_{n+1}
0	0
1	1

4. T - excitation table

Q_n	Q_{n+1}	T
0	0	Q_n 0
0	1	$\overline{Q_n}$ 1
1	0	$\overline{Q_n}$ 1
1	1	$\overline{Q_n}$ 0

T	Q_{n+1}
0	Q_n
1	$\overline{Q_n}$

Scanned by CamScanner

Conversions of flip flops :

1) SRff to Dff.
excitation state.

D	Q_n	Q_{n+1}	S	R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0

Dff state

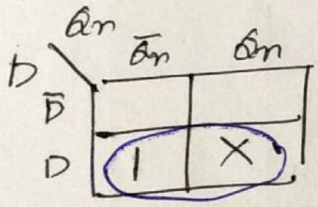
D	Q_{n+1}
0	0
1	1

SRff

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

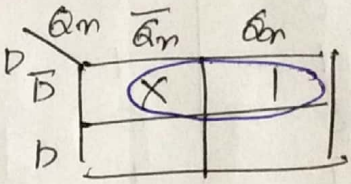
K-map simplification.

For - S

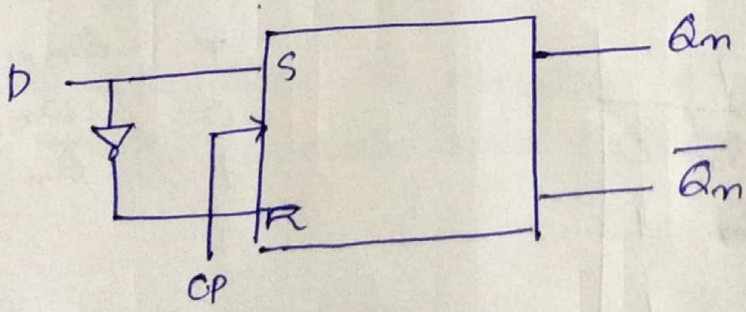


$S = D$

For R.



$R = \overline{D}$



Scanned by CamScanner

2. SRff to Tff
excitation state.

T	Q_n	Q_{n+1}	S	R
0	0	0	0	X
0	1	1	X	0
1	0	1	1	0
1	1	0	0	1

Tff state.

T	Q_{n+1}
0	Q_n
1	$\overline{Q_n}$

Sff excitation

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

K - map simplification

For S

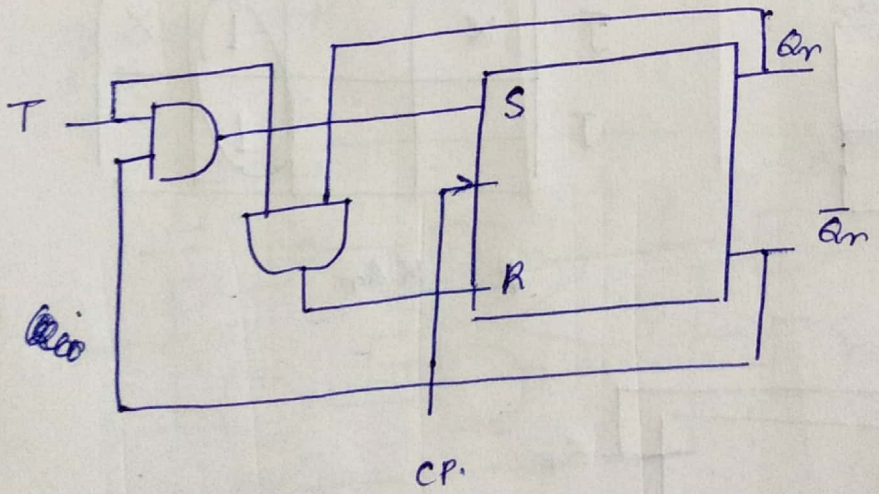
T \ Q_n	$\overline{Q_n}$	$\overline{Q_n}$
$\overline{T}$		X
T	1	

$S = T\overline{Q_n}$

For R

T \ Q_n	$\overline{Q_n}$	Q_n
$\overline{T}$	X	
T		1

$R = TQ_n$



Scanned by CamScanner

SR ff to JK ff.
excitation state

J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

JK ff state

J	K	Q_n	Q_{n+1}
0	0		Q_n
0	1		0
1	0		1
1	1		$\overline{Q_n}$

SR ff excitation

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

K- map simplification

For S.

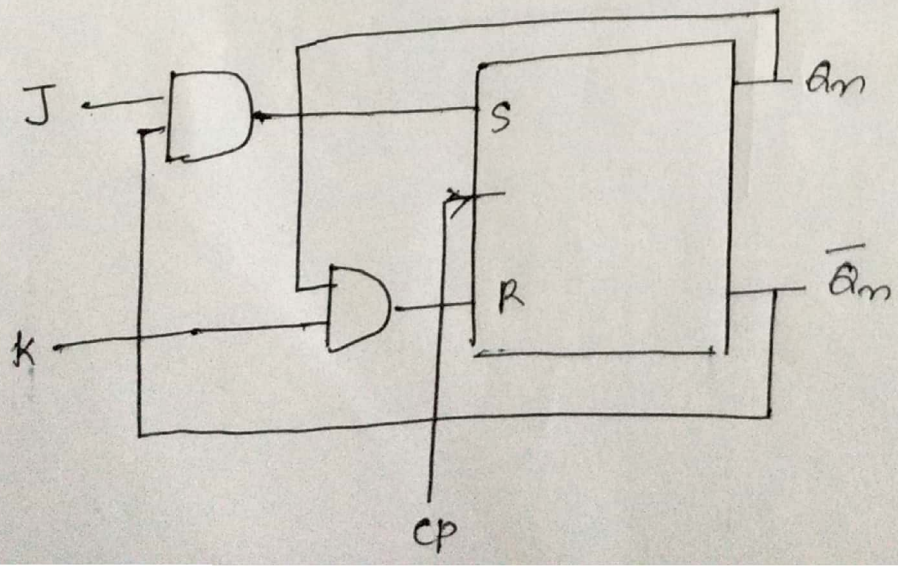
J \ $K Q_n$	$\overline{K}\overline{Q_n}$	$\overline{K}Q_n$	$K Q_n$	$K\overline{Q_n}$
$\overline{J}$		X		
J	1	X		1

$S = J\overline{Q_n}$

For R

J \ $K Q_n$	$K Q_n$	$\overline{K}Q_n$	$\overline{K}Q_n$	$K\overline{Q_n}$
$\overline{J}$	X		1	X
J			1	

$R = K Q_n$



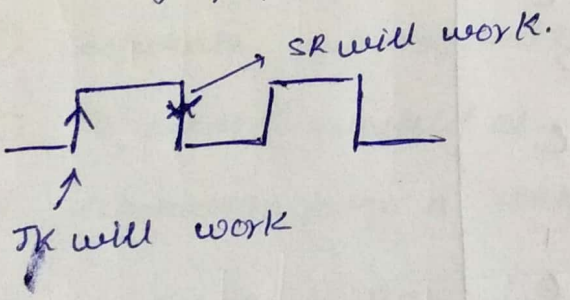
Scanned by CamScanner

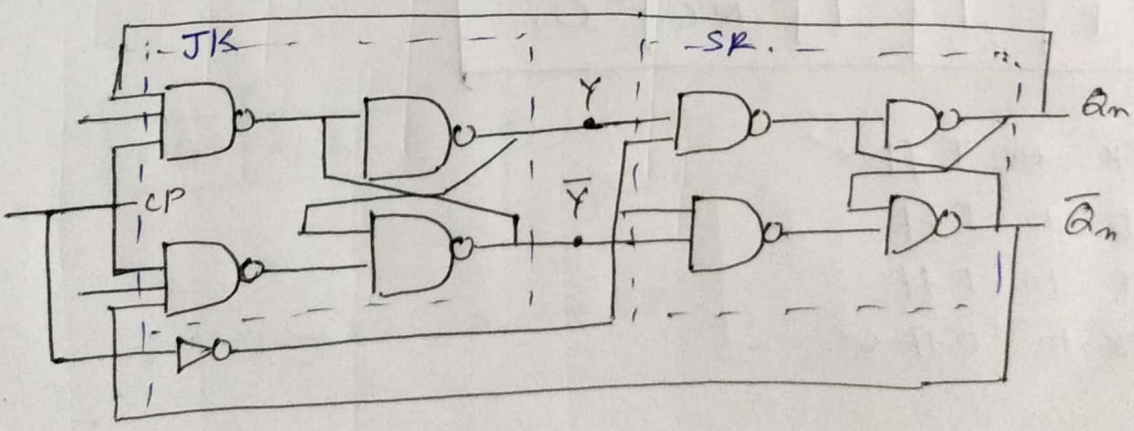
Master Slave Flip Flop :

The master slave combination can be constructed for any type of ff. It consist of clocked JK ff. as master and clock SR ff as a slave.

Clock signal is connected directly to the master but it is conected through inverter to the slave flip flop.

The information present at the J and K inputs is transmitted to the output of master flip flop on the (+)ve clock pulse and it is held there until the (-)ve clock pulse occures after which, it is allowed to pass through the output of slave flip flop.





Scanned by CamScanner

J	K	Q_n
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q}_n$

CP	Q_n	J	K	Y	Q_{n+1}
+ve	0	0	0	0	NC
-ve	0	0	0	NC	0
+ve	0	0	1	0	NC
-ve	0	0	1	NC	0
+ve	0	1	0	1	NC
-ve	0	1	0	NC	1
+ve	0	1	1	1	NC
-ve	0	1	1	NC	1
+ve	1	0	0	1	NC
-ve	1	0	0	NC	1
+ve	1	0	1	0	NC
-ve	1	0	1	NC	0
+ve	1	1	0	1	NC
-ve	1	1	0	NC	1
+ve	1	1	1	0	NC
-ve	1	1	1	NC	0.

Ans. JK to T ff ✓

Ans. D to T ff

Ans. T to D ff

Ans JK to D ff ✓

Scanned by CamScanner

Digital system works on digital system

UNIT-2

All digital computer system are composed of 3 compon

1) set of Registers (are combination of flip flops and the flip flops are +ve edge triggered)

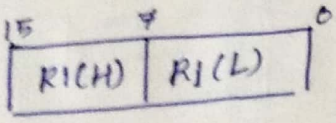
2) a set of microoperations (done on register content)

3) control signals (to control the operations)

Register transfer language (RTL)

1. Notation of Register (Alphabets / Number)
2. Transfer operation "←"
3. Seperate microoperation ","
4. To denote subfield of a Register "( )"
5. Seperation from a control function ":"

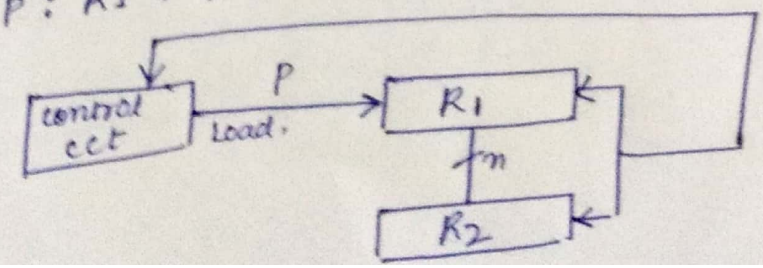
Ex. P : $R_1 \leftarrow R2$ [if (P = 1) Transfer the content of R_2 to R_1]



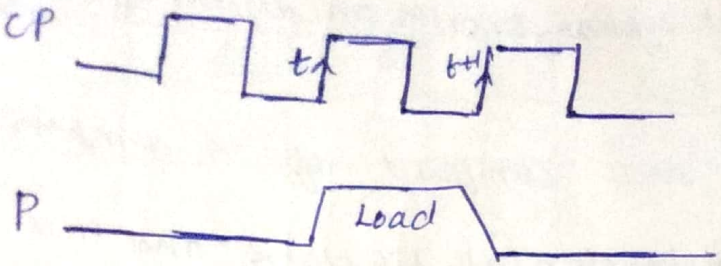
[here ( ) is denoting subfield of register R1 as R1(L) & R2(L)]

P : $R1 \leftarrow R2$, $R4 \leftarrow R3$

P : $R1 \leftarrow R2$



Scanned by CamScanner

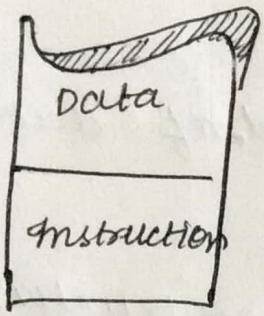


To perform any operation we need 2 things CP and Load.

When the operation is in progress i.e. from t to t+1. At that time load is also active.

Scanned by CamScanner

Stored Program Organization :



memory.

| Opcode | Address |

Instruction format.

| Binary operand |

| Accumulator |

It is assumed that our data and instructions are stored in same memory block.

INSTRUCTION FORMAT :

Opcode : operation code ; which we want to use
Ex. ADD, SUM, DIV, MUL.

Address : Gives the address of memory bits (data) on which we want to apply the operation.
It have the address of one bit operand)
$a_1 + a_2$ (a, ADDa2).

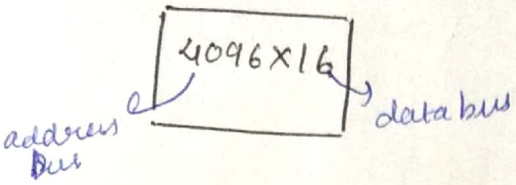
Accumulator : 2nd operand for the operation. it provided by accumulator.

It is a dedicated register

Scanned by CamScanner

Registers :

$4096 = 2^{12}$

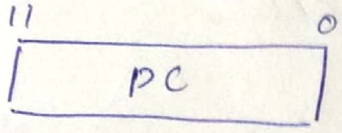


12 lines ~~are required~~ can be provide to send one instruction.

We have 16 bits to save both instruction & address.

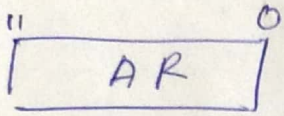
$\underbrace{\text{12 bits}}_{\text{size of address bus.}}$

PC : It holds the address of instruction.

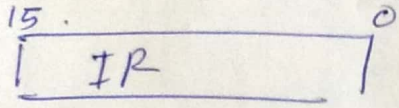


∵ size of register = size of address bus. = 12.

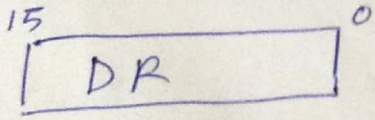
AR (Address Register) : It holds the add. of data.



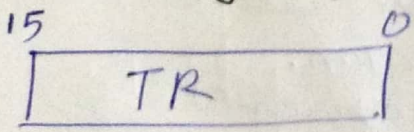
IR (Instruction register) : Holds the ~~add~~ ~~instruction~~ instruction. add. of Data operande



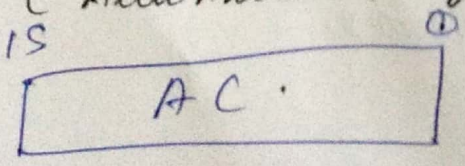
DR (Data Register) : holds the data (operands)



TR (Temprary Regis) : holds the data of temprary results.



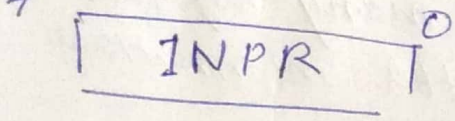
AC (Accumulator Register) : Processor register



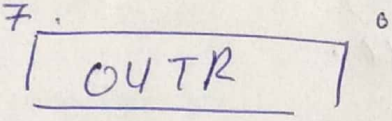
Scanned by CamScanner

INPR : (Input Register) : Holds the input character.

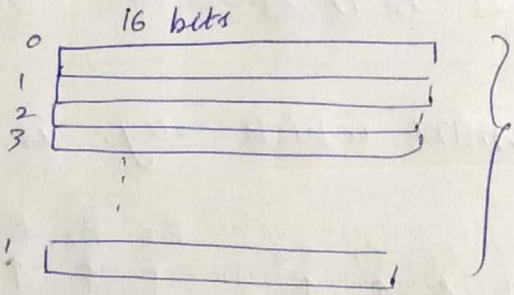
i.e 1 character = 8 bits

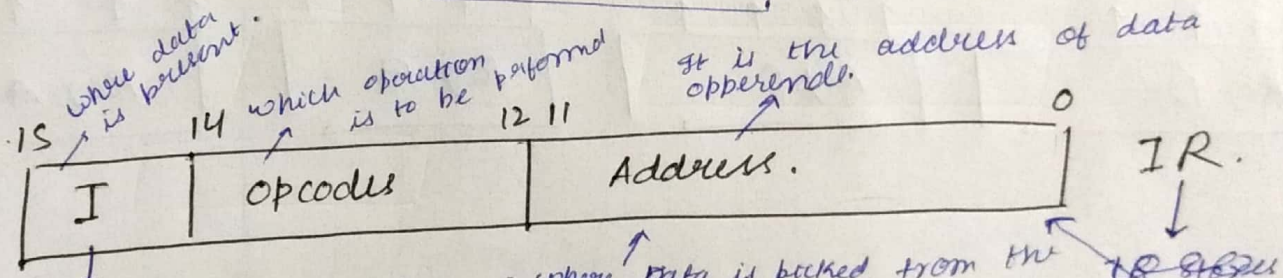


OUTR : (Output Register) : Holds the output character



4096 x 16. ⇒

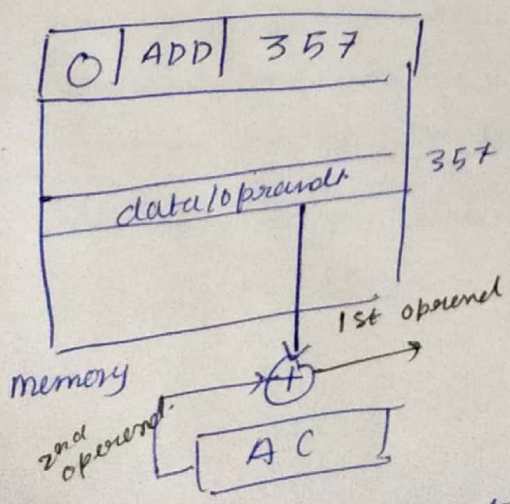




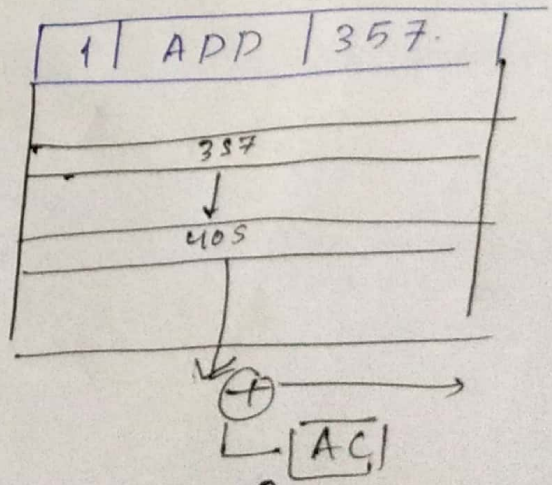
0 : direct addressing → ~~where~~ Data is picked from the address

1 : Indirect addressing → Data is present in the address which is present in the address.

I = 0 (Direct)



I = 1 (Indirect)



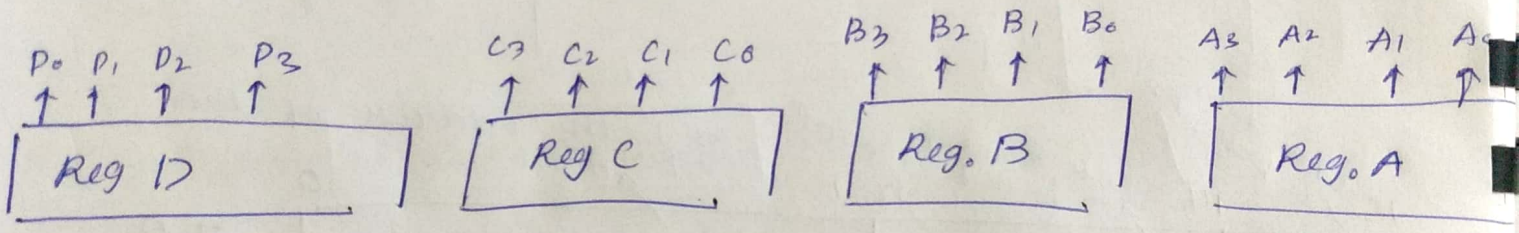
Instruction format is stored in IR).

Scanned by CamScanner

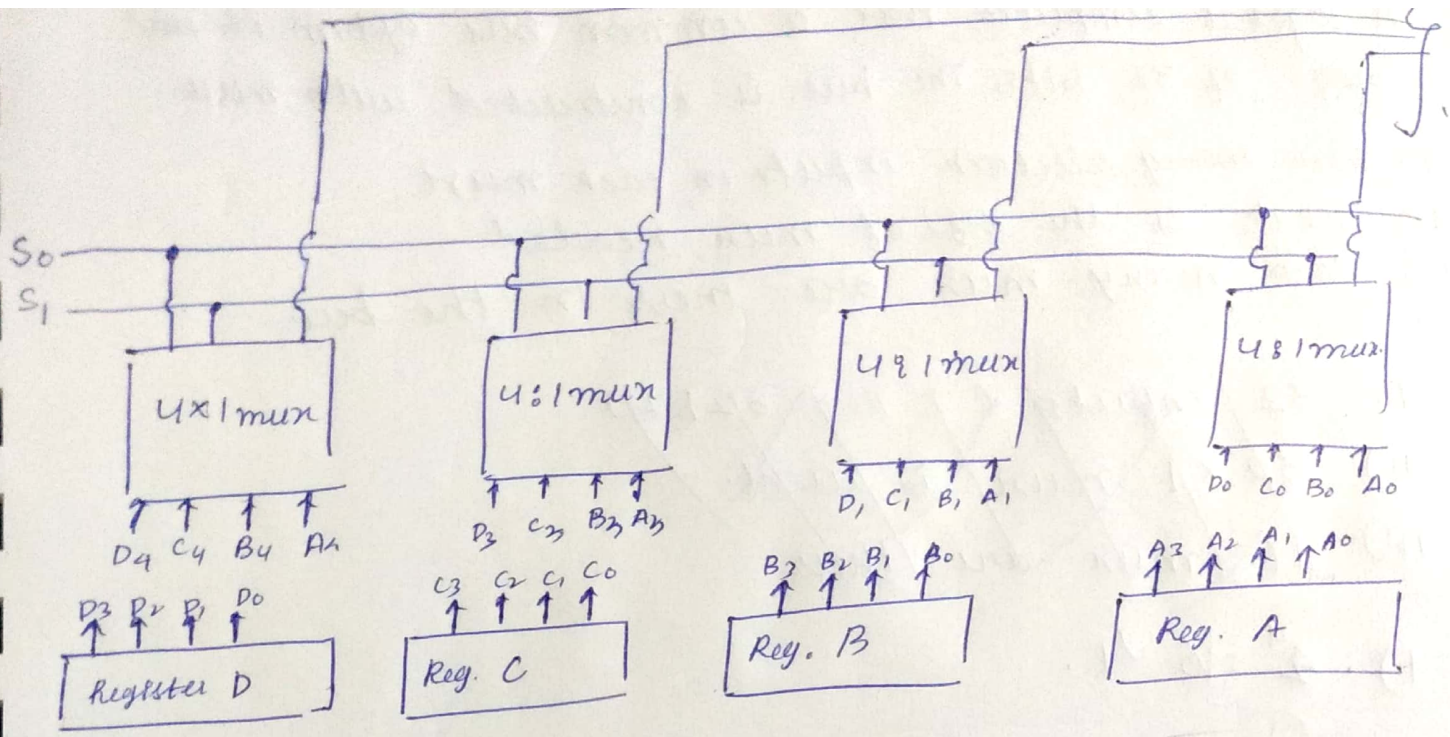
Bus : which is used to communicate b/w registers.

In a typical digital computer we have many registers and paths must be provided to transfer information from one reg. to another.
The no. of wires will be excessive if separate lines are used b/w each reg. and on other reg. in the system that is why we have a common bus.

A bus structure consists of a set of common line. one for each bit of the reg. through which binary information is transferred one at a time.
Control signal determine which reg. is to be selected.



Scanned by CamScanner



S_0	S_1	O/P
0	0	Reg. A
0	1	Reg B
1	0	Reg C
1	1	Reg D

A bus system will multiplex k reg. of n bits each to produce n-line common bus the no. of multiplexer needed to construct the bus is 'n'.
(n → no. of bits in each reg).

The size of multiplexer must be k×1 ∵ it multiplexes k data line.

Scanned by CamScanner

A digital computer has a common bus system 16 bit reg. of 32 bits. The bus is constructed with mux

i) How many selection inputs on each mux

ii) What is the size of mux needed

iii) How many mux are their in the bus.

~~i) 32 inputs (∵ Reg = 32 bits)~~

~~ii) 32 × 1 mux is used.~~

~~iii) 16 mux are their~~

1). $2^4 = 16$ 4.

2) 16 : 1 mux.

3) 32

Scanned by CamScanner

A digital computer has a common bus system 16 bit reg. of 32 bits, The bus is constructed with mux

i) How many selection inputs in each mux
ii) what is the size of mux needed
iii) How many mux are their in the bus.

~~i) 32 inputs (1 Reg = 32 bits)~~
~~ii) 32 × 1 mux is used.~~
~~iii) 16 mux are their~~

i). $2^4 = 16$ 4.

2) 16 : 1 mux.

3) 32

4s. A computer uses a memory unit 256K words. of 32 bits each. A binary instruction has 4 parts an indirect bit, opcode, reg. code part to specify 64 reg. and address part.

i) How many bits are there in opcode, reg. code and address part

ii) ~~How many~~ indicate no. of bits in each part

iii) How many bits are there in data and address inputs of the memory.

Scanned by CamScanner

Answers :

1	7'	6	18
I	opcode	reg. code	Address

← 32 bits. →

$256K \times 32$

$256 \times 1024 \times 32$

$2^{18} \times 32$.

i) ∵ IR is carring the data its size will be 32

⇒ Opcode = 32-(1+6+18)
= 7 bits

ii) 18 bits (address bus)
32 bits (data bus).

i) opcode = 7 bits , reg. code = 6 bits (∵ no. of reg = 64 = 2^6)

address part = 18 bits (∵ 256K = 2^{18}).

ii)

31	30 24	23 18	17 0
I	opcode	Reg.	Add.

Types of Instruction format :

i) memory reference Instruction format
ii) Register reference I.F.
iii) Input output I.F

These format is their to specify to CPU that on which hardware the operation / Instruction is to be performed.

i)

I	opcode	Add.

ii)

0	111	Reg. add

iii)

1	111	Perm.

Scanned by CamScanner

computer may have instructions of several different length containing varing number of addresses the no. of address fields on the instruction format of the computer depends on the internal organization of its reg. Most computers fall into the one of the three types of CPU organization.

1) Zero Address Instruction / stack organization:

$X = (A+B) * (C+D)$ (No address of any operand and operation are required)

Push A	TOP ← A
Push B	TOP ← B
Add	TOP ← (A+B)
Push C	TOP ← C
Push D	TOP ← D
Add	TOP ← (C+D)
Mul	TOP ← (A+B) * (C+D)
POP X	M[X] ← TOP.

2. One address instruction / single Accumulator organization.

$X = (A+B) * (C+D)$

Load A	AC ← m[A]
ADD B (from memory to reg.)	AC ← AC + m[B]
Store T	m[T] ← AC.
Load C	AC ← m[C].
ADD D	AC ← AC + m[D]
Mul T	AC ← AC * m[T]
Store X	m[X] ← AC.

Scanned by CamScanner

3. Two address instruction :/ (General Reg. Organization)

$X = (A+B) * (C+D)$

mov R1, A	$R_1 \leftarrow m[A]$
ADD R1, B	$R_1 \leftarrow R_1 + m[B]$
mov R2, C	$R2 \leftarrow m[C]$
ADD R2, D	$R_2 \leftarrow R2 + m[D]$
mul R_1, R_2	$R_1 \leftarrow R1 * R_2$
mov X, R_1	$m[X] \leftarrow R1$

4. Three address instruction :/ (Gen. Reg. organization)

$X = (A+B) * (C+D)$.

Add R1, A, B	$R1 \leftarrow m[A] + m[B]$
Add R2, C, D	$R2 \leftarrow m[C] + m[D]$
mul X, R1, R2	$m[X] \leftarrow R_1 * R2$

Ans. 256K × 32 [memory size]
2^{18}

PC → 18 bits

AR → 18 bits

IR → 32 bits

DR → 32 bits

TR → 32 bits

AC → 32 bits

INPR → 8 bits

OUTR → 8 bits.

Scanned by CamScanner

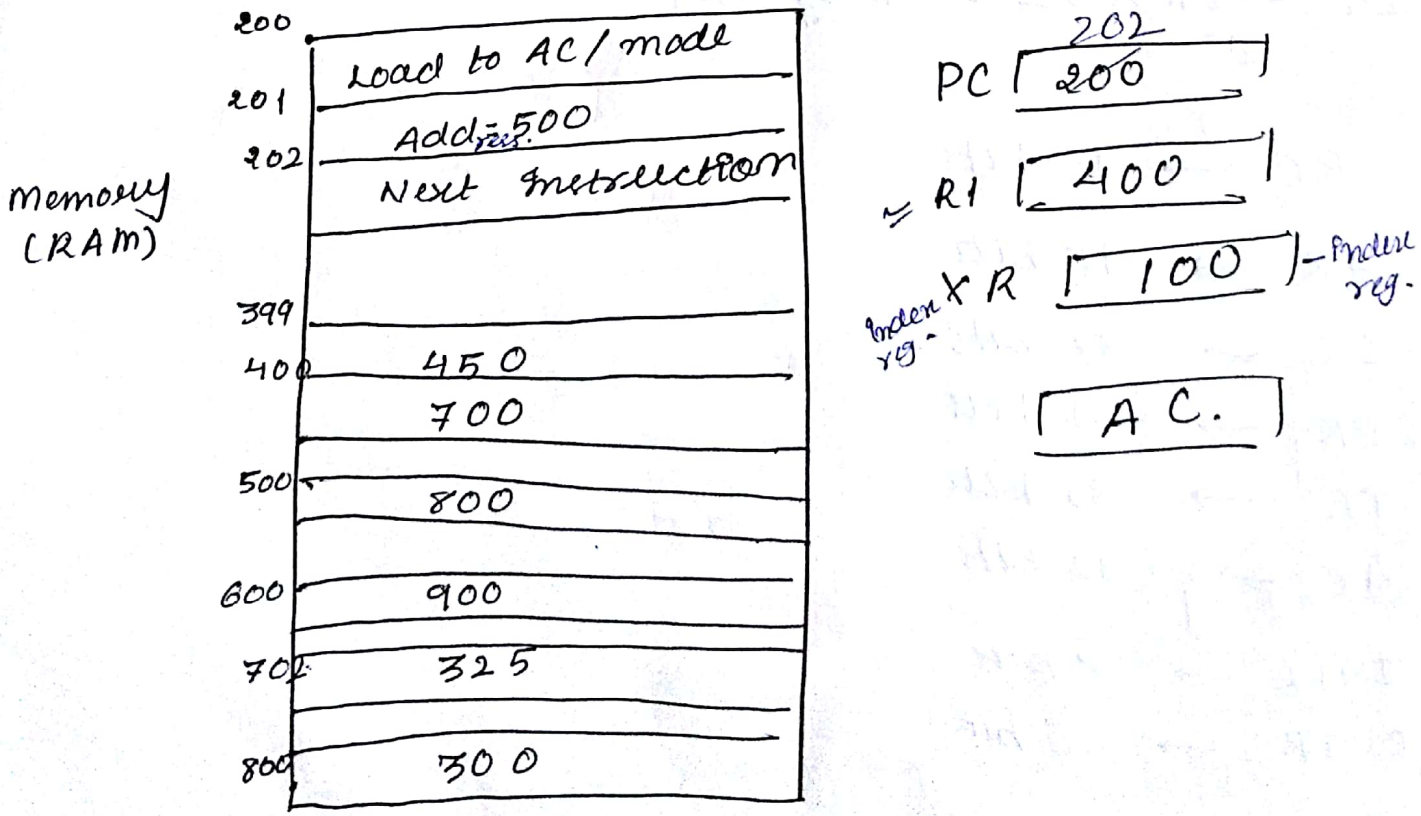
Addressing Modes :

It specifies a rout for interpreting or modifying the address ~~now~~ field of the instruction. Variety of addressing modes are supported;

1) To give programming flexibility to the user.
ii) To use the bits in the address field of the instruction efficiently.

a) Implicit mode (Where we know where the operand it we don't fetch it)
b) Immediate mode (| I | Op | operand |) we have direct value inplace of address).
c) Direct addressing mode.
d) Indirect addressing mode.
e) Register direct mode.
f) Register indirect mode.
g) Relative Addressing. (PC)
h) Index addressing.
i) Base Register addressing.
j) Auto increment / Auto Decrement.

Reg. + Add. field = EA



Scanned by CamScanner

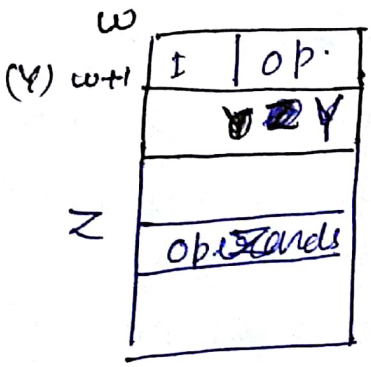
Effective address : Is the address from which we fetch the data finally.

Mode	Effective address	operands.
(a)	X	X
(b)	201	500
(c)	500	800
(d)	800	300
(e)	X	400
(f)	400	700
(g)	202 + 500 = 702	325
(h)	100 + 500 = 600	900
(i)	Base reg + 500 =	✓
(j)	i) auto increment. 400 then ++R1 = 401.	700
	ii) auto decrement at first decrement the address value and then EA is obtained 399.	450

Scanned by CamScanner

4. 2 word instruction stored at address 'W'
addr. field of instruction 'W+1' or 'Y'
Operands used during execution of instruction is stored at address symbolized by 'Z'.
Index Reg. contain value 'X'.
State how 'Z' is calculated for

a) Direct b) Indirect c) relative d) Index.



XR [X]

PC [W+2]

Operand stored at Z

	EA	Oppuands
~~Direct~~		Z = Y
Direct	Y	
Indirect	Z = m[Y]	
Relative	Z = W+2+Y	
Index.	Z = X+Y.	

Scanned by CamScanner

48

Instruction 300
address field 301
value (addr. field) = 400

R_1 [200]

PC [302]

300	I | opc
301	400
302	
400	

Processor reg (R_1) contain 200.
Evaluate EA
a) Direct b) Imediate c) Relative d) Reg Indirect
e) Index with R_1 as index reg

	EA
a) Direct	400
b) Imediate	301
c) Relative	302+400 = 702
d) Reg. Indirect	200
e) R_1 as Index	600 (200+400).

Interrupts : (generated by I/O device to let the CPU know for transferring of data)

1) External (received from outside world)
2) Internal (also called traps)
3) S/w are Interrupts

External interrupts are initiated outside of CPU and memory:
Ex Powerfailure, I/O device data transfer, time out.

Context switching - external int.

Traps / Internal: caused by currently running app.
Ex. protection violation (in terms of O/S).
when P_1 acces memory of P_2.

Reg. stack overflow

Scanned by CamScanner

iii) Divide by zero
iv) Opcode validation.

S/W Interrupt:

Internal and External interrupts are initiated by hardware. but s/w interrupts are initiated by the execution of an instruction.

Ex. Supervisor call (switch from user to kernel mode)

Priority Interrupts:

When many I/O device are requesting for data transfer at same time (collision of requests) is solved by priority interrupts:

↳ we have two approch

① Software Approch (Polling)
② Hardware Approch.

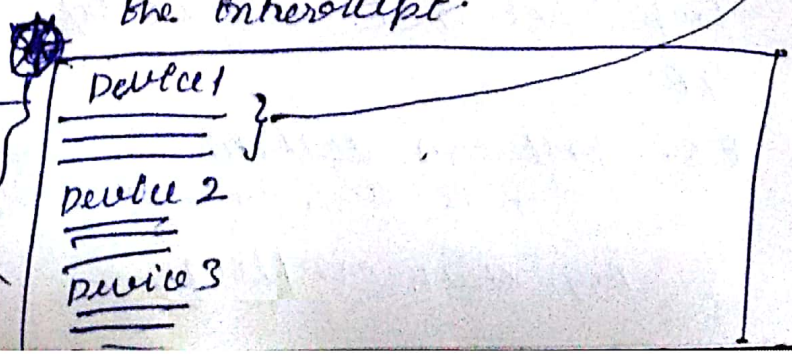
O/S have to handle the interrupts

IBR (Interrupt service routine) → located at VAD (vector address)
↓
to handle s/w interrupts.

Routine → section of code used to handle a particular interrupt.

In S/W approch we have common branch address.
↳ It has the code to handle the interrupt.

Priority is based on how it is written on this



Scanned by CamScanner

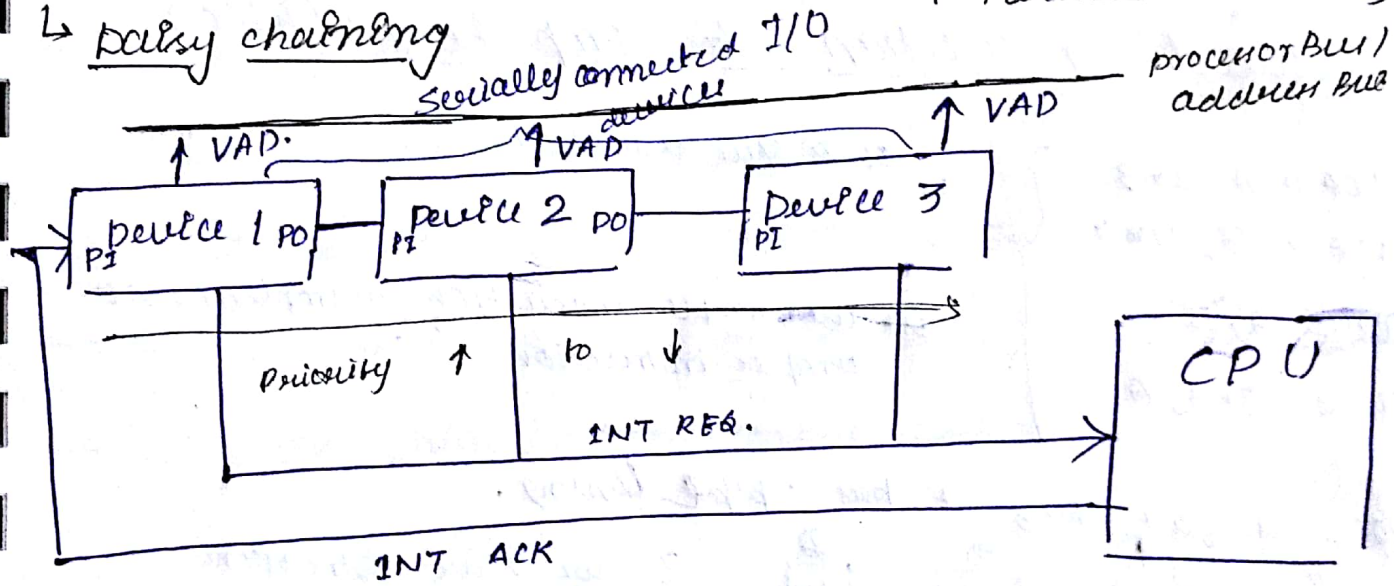
1) Polling is slow
2) flexible → we can change the code of (⊗).
3) less costly.

2) Hardware method :

1) faster
2) not flexible → we have to change whole architecture
3) more costly.

① Serial Hardware approch
↳ Daisy chaining

② // Hardware approach (parallel priority).



INTREQ → Interrupt request
INTACK → Interrupt acknoledgement.

VAD → It contain address of interrupt routine.
PI → Priority input
PO → Priority output.

→ These line/section will be holding a way to access the specific ISR.

Scanned by CamScanner

Let say INTACK sends 1 to Device 1, if Device 1 has not generated the interrupt it will receive it in PI and send same in PO.

If say Device 2 has generated the interrupt it will block PI and send 0 to next device and execute the routine.

The device in the begining has highest priority as it will receive the INTACK first.

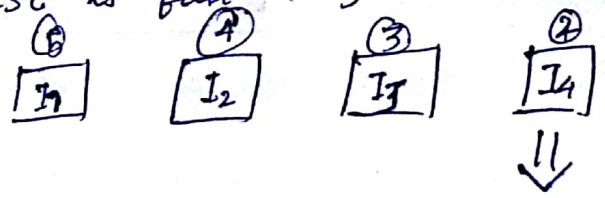
Reduced Instruction Set computers (RISC)

LOAD A, 2*3
LOAD B, 4*4
MUL A, B.
Store 2*3, A

↳ It is S/w oriented.

→ use simple instruction to implement complex instruction.

→ RISC is fast beoz it has pipelining.



} we have instruction set

↳ We divide a large tasks in multiple small tasks and they are implemented parallely.

↳ In RISC each instruction is of one clock cycle.

→ RISC has less addressing mode
→ RISC has less no. of instruction.
→ single clock cycle instruction.
↳ Assembly code is more in size in RISC.

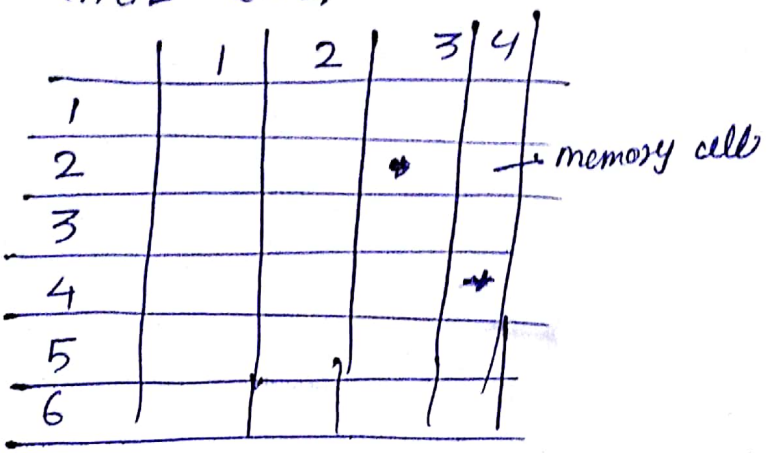
Scanned by CamScanner

CISC (complex instruction set computers)

→ CISC have instruction ~~and~~ ~~H/w~~ supported in H/w.

→ CISC is hardware oriented.

MUL 2*3, 4*4 ⟶ H/w HLL.



↳ pipe lining implementation is ~~not~~ not there.

→ CISC is slower as compared to RISC

→ CISC has more addressing modes

→ No. of instruction are more in CISC

↳ (CISC use micro program control unit).

↳ No. of Registers is less in CISC architecture as no. of transistor are more ∵ more no. of transistors are there to compute complex instruction on processor chip.

↳ multiple clock cycle instruction.

↳ ~~more~~ In CISC assembly code will be less in size thus occupying less RAM.

Scanned by CamScanner

COMMON BUS :

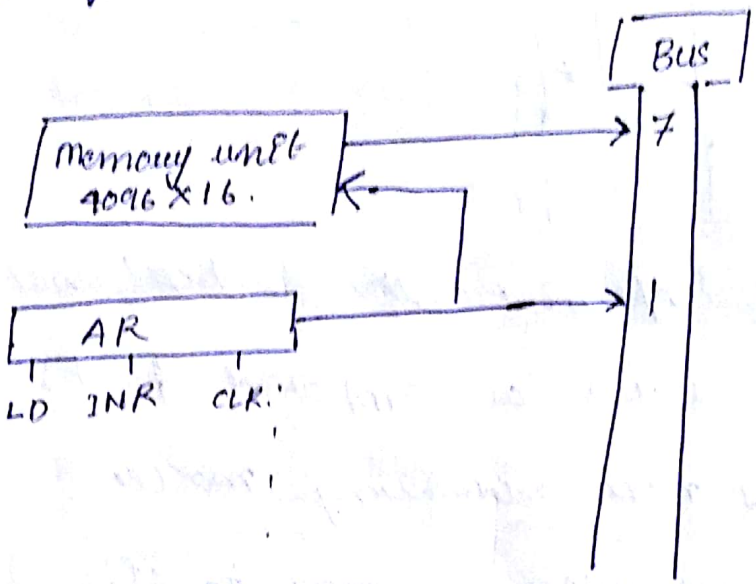
CB → Reg → Bus
(LD active) (signal line)

CLR → clear the data.

for AC we don't take data from CB.

[E] → flip-flop for computation.

IR → only have LD signal ∵ it is a instruction from memory.

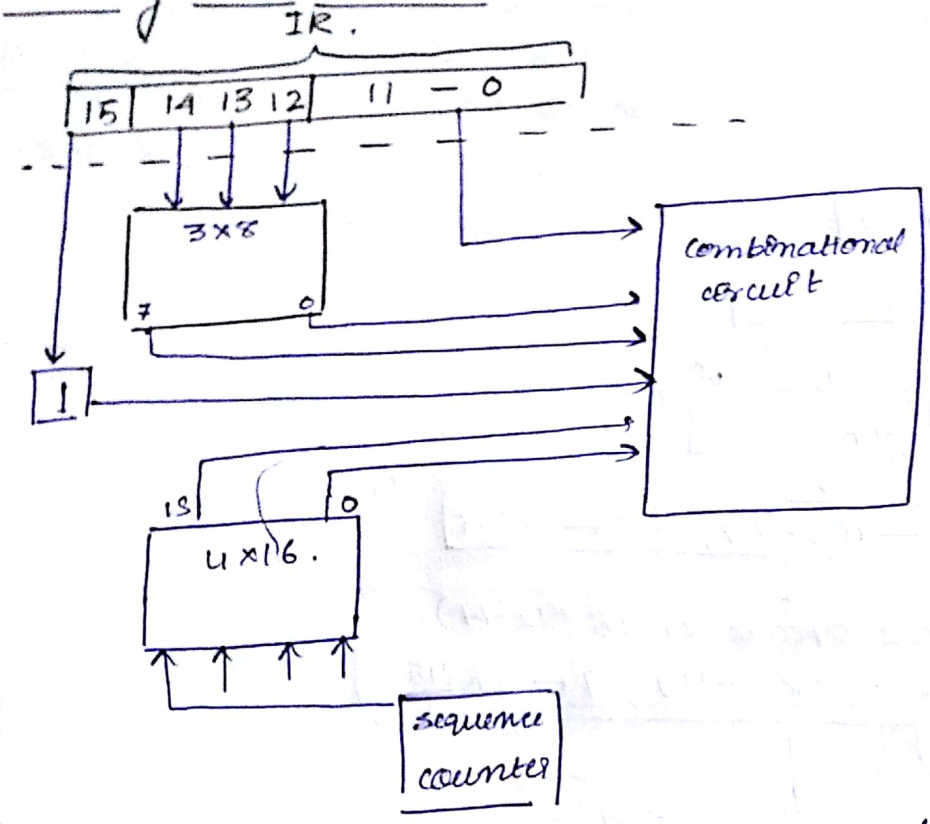


Control signal : (Read, write etc) used to tell that whether the data is to be read or written to the memory.

sequence counter generate timing signal

Scanned by CamScanner

Timing and control :



* 1-instruction have one clk tick to get execute

Instruction cycle :

1) Fetch the instruction from memory
2) Decoding the instruction
3) calculate the Effective address if there is indirect addressing.
4) Execute the instruction.

Fetch and Decode :

clock tick

T_0 : $AR \leftarrow PC$ (S_0 S_1 S_2) (0 1 0)

T_1 : $IR \leftarrow m[AR]$ $PC \leftarrow PC+1$. (S_0 S_1 S_2) (1 1 1)

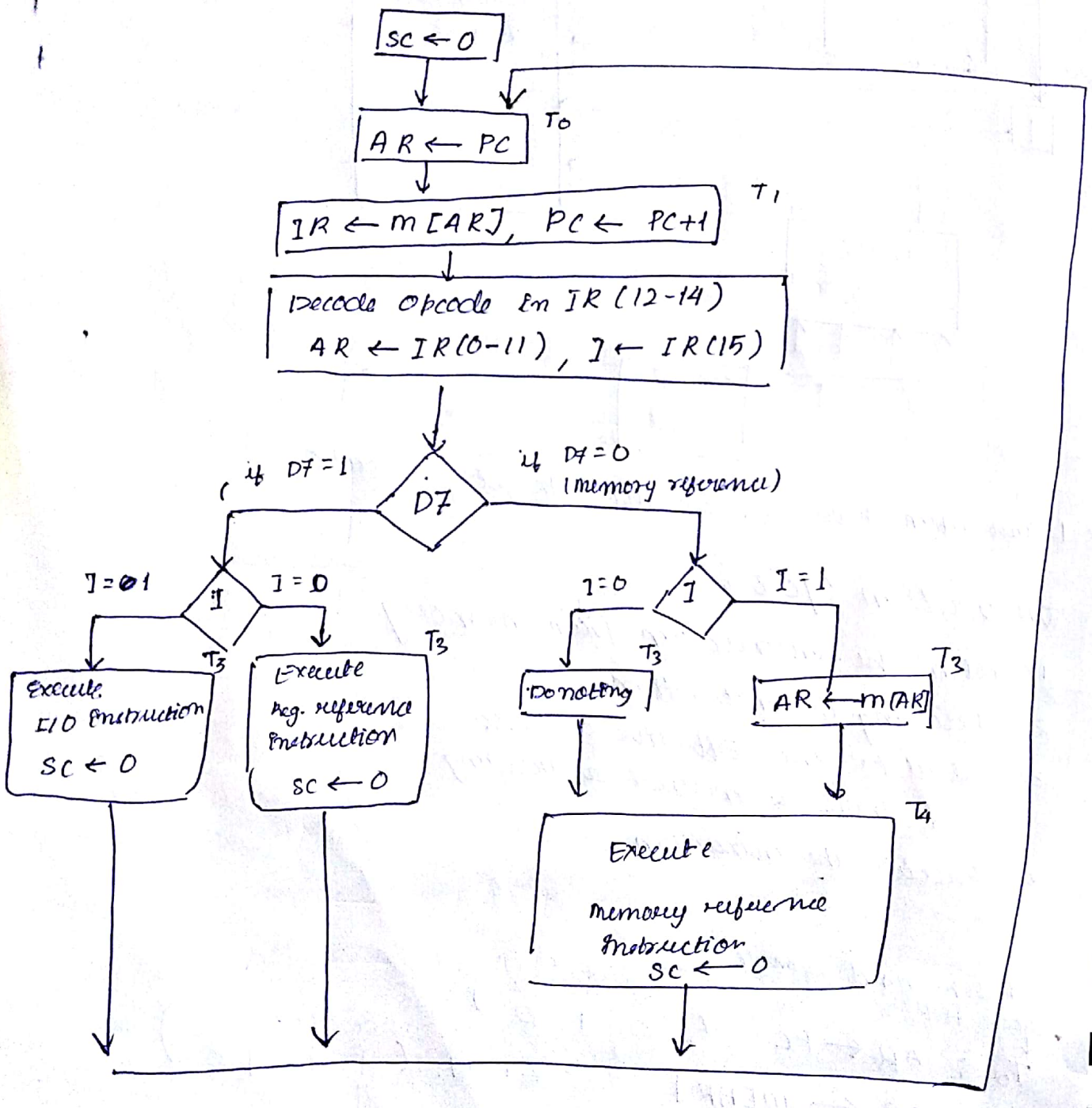
address at m[AR] is stored in IR.

as we will fetch the data from memory & memory is accessed using

Scanned by CamScanner

T_2 : $\underbrace{D_7 \ldots D_0 \leftarrow IR(12-14)}_{\text{Decode}}$, $\underbrace{AR \leftarrow IR(0-11)}_{\text{putting address of IR in AR.}}$, $\underbrace{I \leftarrow IR(15)}_{\text{storing mode bit in I-flip flop.}}$

4096 × 16 bits memory



* memory reference take one extra cycle T_4 to execute the instruction

$D_7' I T_3$: $AR \leftarrow M[AR]$

$D_7' I' T_3$: Nothing.

$D_7 I' T_3$: Execute a reg. reference instruction

$D_7 I T_3$: Execute an I/O instruction.

I. Register Reference Instruction:

15	14	12 11	0

$D_7 = 1, I = 0$

$r = D_7 I' T_3 \Rightarrow$ reg. ref. instruction.

$B_i = IR(i)$, $i = 0, 1, 2, \ldots 11$

$r B_{11}$	CLA (clear accumulator)	$AC \leftarrow 0$.
$r B_{10}$	CLE (clear E flip flop)	$E \leftarrow 0$
$r B_9$	CMA	$AC \leftarrow \overline{AC}$
$r B_8$	CME	$E \leftarrow \overline{E}$
$r B_7$	CIR (circ shift all bits)	$AC(15) \leftarrow E$, $AC \leftarrow cir(A$ $E \leftarrow AC(0)$.
$r B_6$	CIL	~~E ← AC(0)~~ $AC(0) \leftarrow E$, $AC \leftarrow cil(AC)$ $E \leftarrow AC(15)$
$r B_5$	INC (Next inst)	$AC \leftarrow AC+1$
$r B_4$	SPA next if it is (+)ve.	if $AC(15) = 0$, $PC \leftarrow PC+1$

Scanned by CamScanner

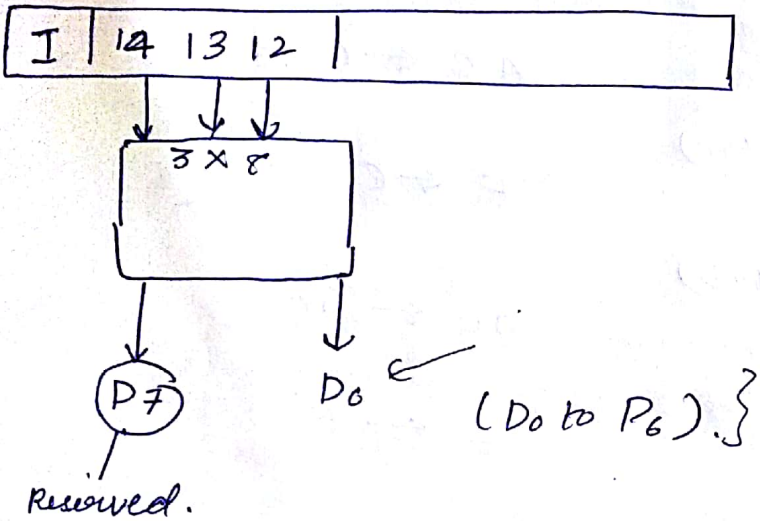
uB_3 SNA if (AC = 1) PC ← PC + 1.
(if AC is +ve no.)

uB_2 SZA if (AC = 0)
PC ← PC + 1

uB_1 SZE if (E = 0) PC ← PC + 1

uB_0 HLT SC ← 0.

Memory Reference Instruction.



(D0 to D6). } These are operation

* D0 is AND.

D0T4 : DR ← M[AR]

D0T5 : AC ← AC ∧ DR , SC ← 0

* D1 is ADD.

D1T4 : DR ← M[AR]

D1T5 : AC ← DR + AC , E ← Cout , SC ← 0
(If carry is generated)

Scanned by CamScanner

* D_2 LDA (load Into Accumulator)

D_2T_4 : $DR \leftarrow m[AR]$. N : AC ~~doesn't have direct~~ is not directly connected to bus all. to our system.

D_2T_5 : $AC \leftarrow DR$
$SC \leftarrow 0$

* D_3 STA (store).

D_3T_4 : $m[AR] \leftarrow AC$. $SC \leftarrow 0$

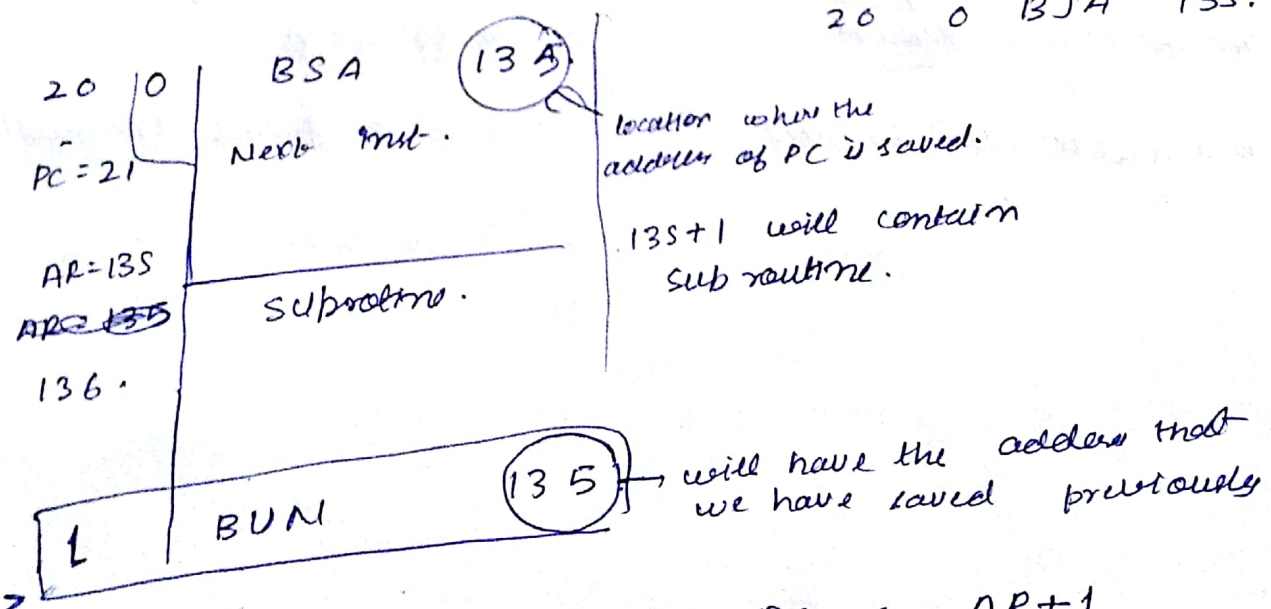
* D_4 BUN (Branch unconditionally).
(jump anywhere in the program)

D_4T_4 : $PC \leftarrow AR$ $SC \leftarrow 0$.

* D_5 BSA (Branch and save return address)

In BSA it save the address where the prog. counter will return after executing jumped instruction.

20 0 BSA 135.



D_5T_4 : $m[AR] \leftarrow PC$, $AR \leftarrow AR+1$

D_5T_5 : $PC \leftarrow AR$, $SC \leftarrow 0$

Scanned by CamScanner

* D_6 ISZ (Increment and skip if zero).

Increase the value of operand

D_6T_4 : $DR \leftarrow M[AR]$

D_6T_5 : $DR \leftarrow DR + 1$

D_6T_6 : $M[AR] \leftarrow DR$ if $(DR == 0)$ then

$PC \leftarrow PC + 1$, $SC \leftarrow 0$.

Q: What is the difference b/w BSA, BUN and Interrupt.

BSA	BUN
Branching and saving the returned address but no program modification is allowed.	Unconditional Branching and no saving of address but allows program modification.
It is used to call subroutine	It is a jump instruction

Scanned by CamScanner

FG0 → initially 1

FG1 → initially 0 [data can come from input device]
↓ input flag

- **I/O Reference Instruction :**

Program controlled input output data transfer.
(where CPU checks FG1 and FG0)

Interrupt Transfer :

15	14 13 12	11 ... 0
1	1 1 1	I/O operation

B_i where $B_{11}, B_{10}, B_9 \ldots B_6$

$P = D_7 I T_3$

PB_{11} : INP (input)

$AC(0-7) \leftarrow INPR$; $FG1 \leftarrow 0$

PB_{10} : OUT (output)

$OUTR \leftarrow AC(0-7)$ $FG0 \leftarrow 0$

PB_9 : SKI

if $(FGI = 1)$

$PC \leftarrow PC + 1$

PB_8 : SKO

if $(FGO = 1)$

$PC \leftarrow PC + 1$

PB_7 : ION

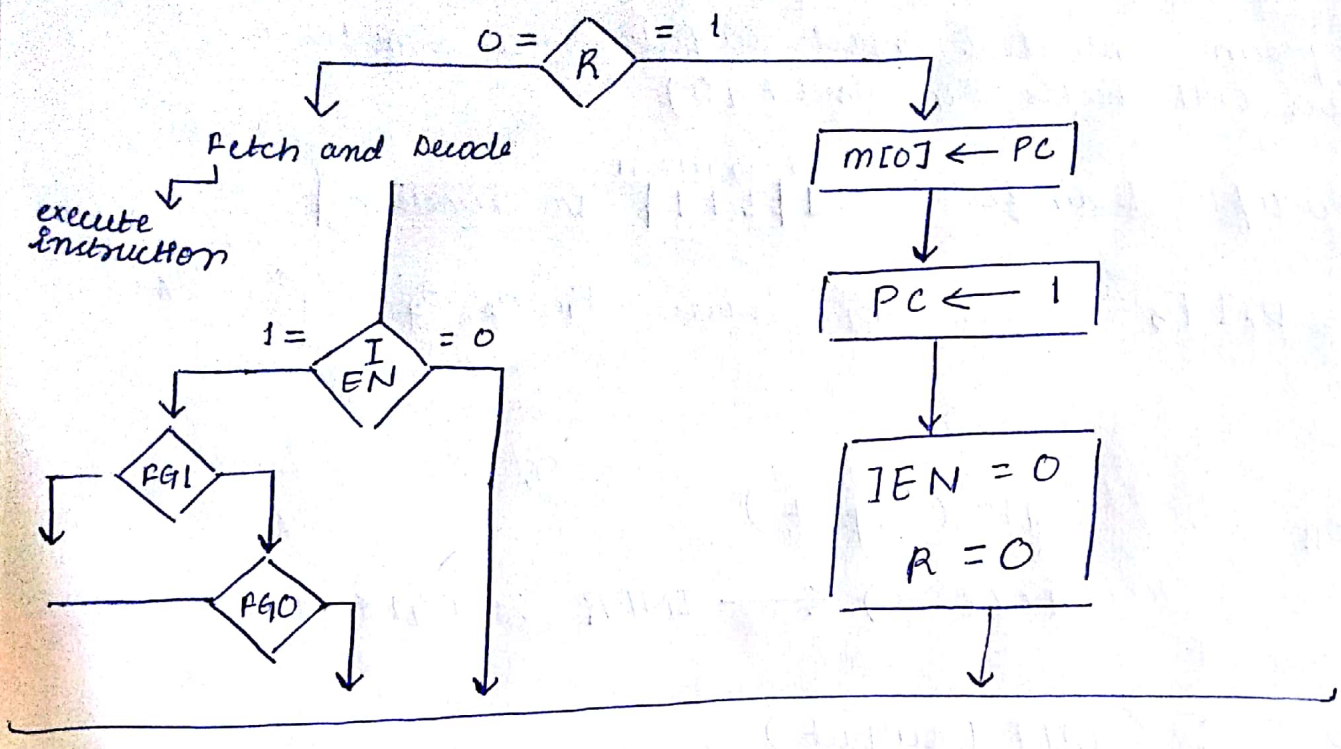
$IEN = 1$ ⟶ another flag.

PB_6 : IOF

$IEN = 0$

Scanned by CamScanner

INTERRUPT CYCLE :



Segments ⟶ Registers + combinational circuits.

1. Pipelining :

A technique of decomposing a sequential Process into sub operations, with each subprocess being executed.

Ans. 'K' segments with 'tp' time to complete 'n' tasks.

$$\underbrace{K t_p}_{\text{for } 1^{st} \text{ task}} + (n-1) t_p$$

$$t_p (K + n - 1)$$

For non-pipelined

t_n : n tasks

$n t_n$.

$$\text{Speedup ratio} = \frac{n t_n}{(K+n-1) t_p}$$

$$= \frac{t_n}{t_p} = \frac{K t_p}{t_p} = K.$$

Scanned by CamScanner

4. $t_p = 20$ nsec, $k = 4$ segments, $n = 100$

$$= \frac{100 \times 4 \times 20}{(4+99)\,20} = \frac{400}{103} = 3.88.$$

GBG
GDG

#DSC

Scanned by CamScanner

segments , $n = 100$

$$= \frac{100 \times 4 \times 20}{(4+99)\,20} = \frac{400}{103} = 3.88$$

Pipelining can increase the throughput of the system.

INSTRUCTION PIPELINE :

Six Phase in an instruction cycle (Instruction Pipeline)

1) Fetch an instruction from memory
2) Decode the instruction (done by hard wired control unit)
3) calculate the EA of operand (in case of indirect.)
4) Fetch the operands from memory
5) Execute the operation
6) store in the memory.

4 stages.

FI → Fetch instruction
DA → Decode and calculate EA
FO → Fetch operand from memory
EX → Execute and store.

Execution of 3 instruction in 4-stage pipeline.

Conventional.

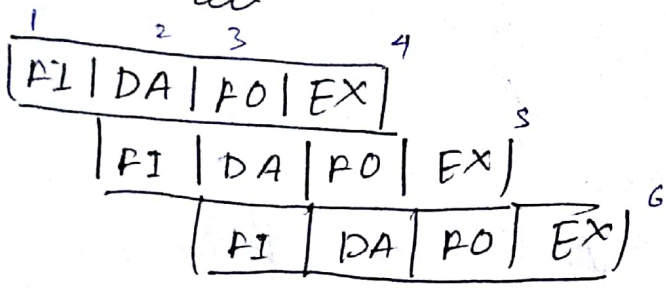
1	2	3	4	5	6	7	8	9	10	11	12
FI	DA	FO	EX								
				I2 FI	DA	FO	EX				
								I3. FI	DA	FO	EX

1 = T_4 (completed)
2 = T_8 (completed
3 = T_{12} (completed).

GB 9

Scanned by CamScanner

Pipelined



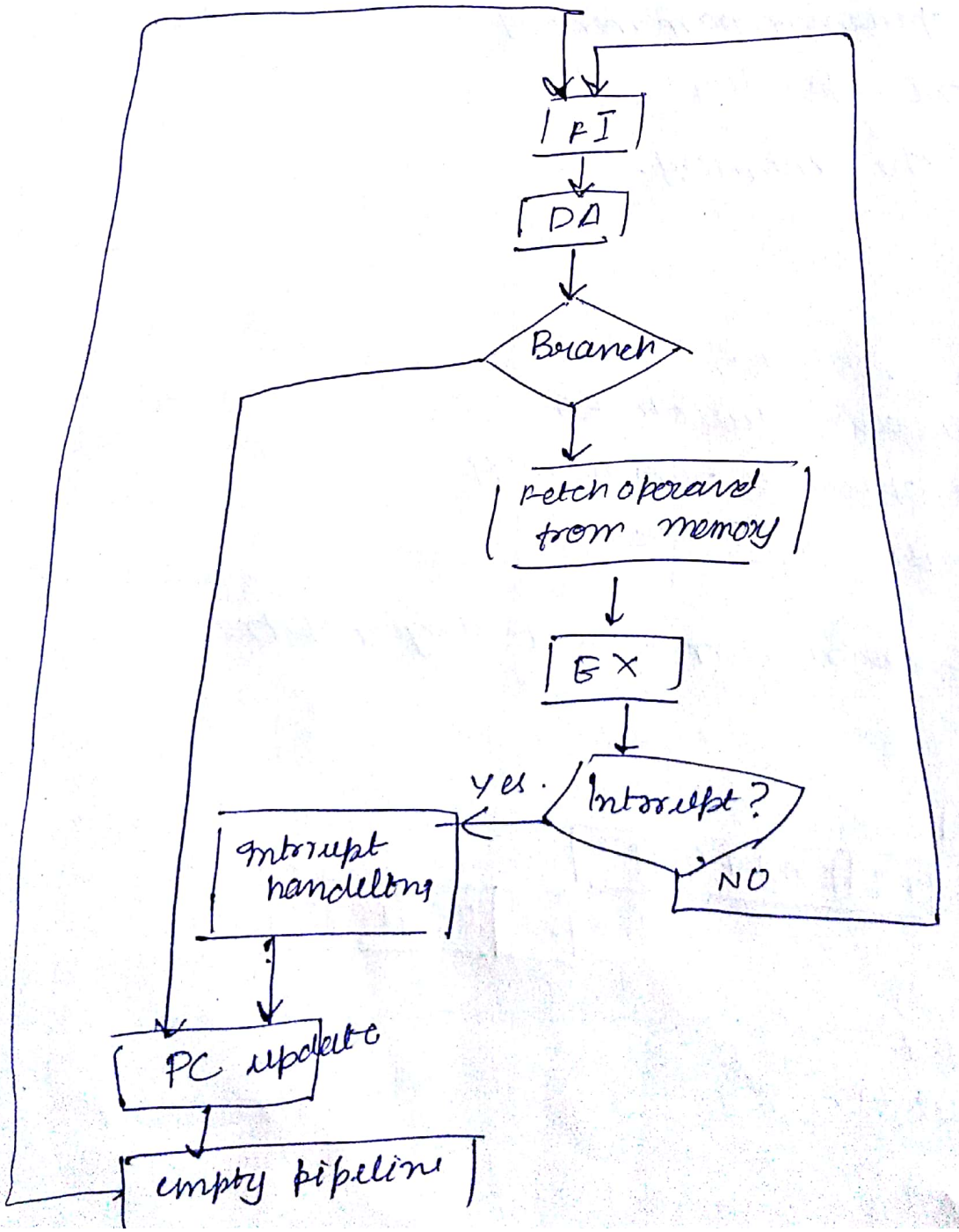
$I_1 = T_4$.

$I_2 = T_5$.

$I_3 = T_6$.

* Pipeline is not successful in case of branced instruction and in case of interrupt.

so in that case we will empty the pipeline till the branced instruction or ISR is executed.



Scanned by CamScanner

Hazards in Pipelining :

1. Structural hazards

↳ Resource conflict (hardware resources required by the instructions in simultaneous overlapped execution can't be allowed.

Stall : wasting CPU cycle as ~~we~~ in cannot perform any operation

→ with one memory port, we can either fetch operand or instruction at a time.

FI	DA	FO	EX				
	FI	DA	FO	EX			
		stall	stall	FI	DA	FO	EX

2. Data Hazard :

occurs when the execution of an instruction depends on the results of a previous instruction.

Solun : Hardware Tech.

i) Interlocks (hardware detects the data dependencies and delay the scheduling by stalling.

ii) Forwarding : (bypassing, short circuit).

we don't save the intermediate data in any reg. the ALU hold that value by removing the phase of fetching again. (ALU to ALU transfer)

Ex.

ADD R1, R2, R3

SUB R4, R1, R5.

I → Fetch.

A → Decode, Execute.

E → Store.

ADD

I	A	E						
	I			A	E			
		I	A	E				

Interlocking

Bypassing

Scanned by CamScanner

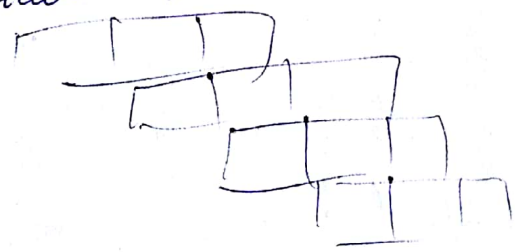
Software Tech: (compiler handle the hazard).

Delayed Load.

Ex. LOAD: $R1 \leftarrow m[X]$
$R2 \leftarrow m[Y]$.
$R3 \leftarrow R1 + R2$ } data dependency
$m[Z] \leftarrow R3$

NOP → an instruction that convey special meaning to CPU i.e NO operation (it ~~use~~ waste whole cycle)

1 NOP = 3 stalls if (I A E are instruction).



Control Hazards: Solⁿ → Delayed branch.
→ caused due to the instruction that are not sequential
(Branch instruction / interrupts) generates control hazard

Solⁿ: using special type of hardware.

Solⁿ
Associative memory or TLB (Translation Look aside buffer)
CAM. (content addressable memory)

Scanned by CamScanner

hardwired control unit ; control signals are generated using hardware
↳ Ex $D_7 I' T_3$ or $D_7' I' T_4$.

microprogrammed control unit ;
→ consist of microinstruction
microprogram is a part in control memory, and control memory is a part of ROM.
this microprogram will generate control signal.

$\boxed{CAR}$: control address reg. [same as PC]. = 10 bits.
it holds the address of next microinstruction.

$\boxed{CDR}$: control data reg. [same as IR]. = 20 bits
it holds the data about the micro instruction.

Assuming that control memory is $\boxed{\begin{array}{l} 1024 \times 20 \\ 2^{10} \times 20 \end{array}}$

$\boxed{\text{Next address Generator (sequencer)}}$: generates the address for next micro instruction to be generated
(Address sequencer)

CDR → also known as pipeline reg.
↳ It is telling the micro instruction
↳ tells the address of next instruction.

Address Sequencing :

1) address in CAR.
2) CAR is to be run in sequential order [CAR+1]

SBR (sub routine Reg) = 10 bits.
holds the address of (CAR+1) next micro instruction.
In case BSA.

Mapping logic : convert the op code of instruction into size same as the size of address of control memory, that we can address the instructions in CM.

Scanned by CamScanner

Format of CAR.

Jump → conditional [we check status bit → if (True) jump to the ~~next~~ ~~Bra~~ instruct
→ if (false) Next instruction CAR+1.

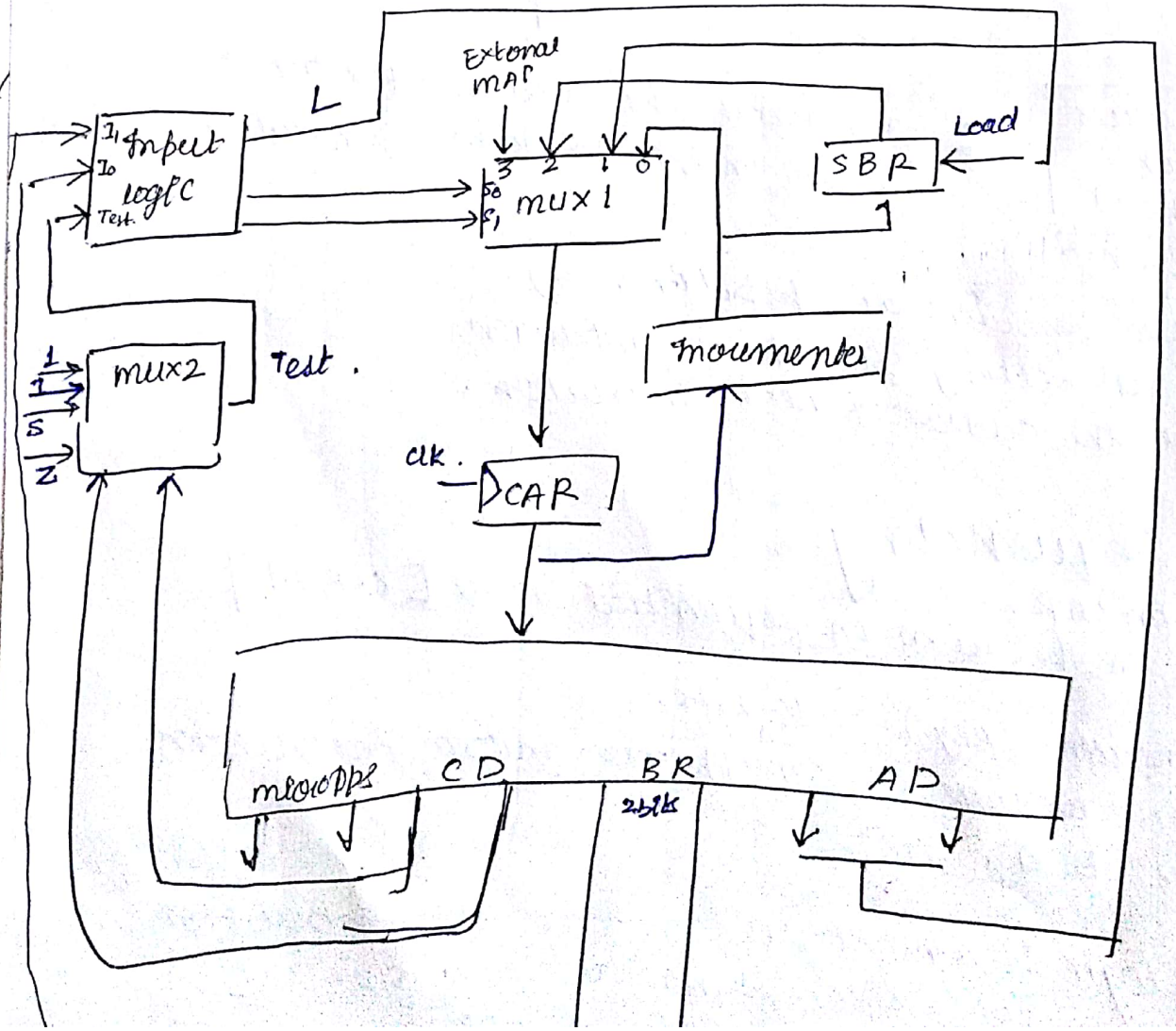
→ unconditional.

Format of microinstruction format :

3	3	3	2	2	7
F1	F2	F3	CD	BR	AD

F1, F2, F3 → microoperations ($2^3+2^3+2^3$ = 24 operation)
CD → condition for branching ($2^2=4$ condition)
BR → Branch field ($2^2=4$ conditions / method)
AD → Address field

Microprogram Sequencer :



Scanned by CamScanner

BR is going to input logic.

$I_1 I_0 T$	meaning.		$S_1 S_0$	L
0 0 0	In-line.	CAR+1	00	0
0 0 1	JMP	CS(AD)	01	0
0 1 0	In-line	CAR+1	00	0
0 1 1	CALL	CS(AD) & SBR←CAR+1	01	1
1 0 x	RET	CAR←SBR.	10	0
1 1 x	MAP	DR(11-14)	11	1.

CAR(1-4), CAR(0,5,6)←0.

* In case of return we don't need the test condition

Qs1.
Size of control memory is 1024 × 32 bits each. micro instruction has 3 fields. micro operation field has 16 bits

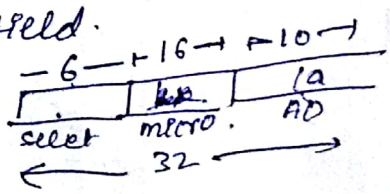
a) How many bits in the Branch address field & select field.

b) If there are 16 status bits in the system how many bits of the branch logic are used to select status bit.

c) how many bits are left to select an input for mux

Ans. $1024 \rightarrow 2^{10} \rightarrow$ 10 bits in address field.

$32 \rightarrow 2^5 \rightarrow$ 5 bits

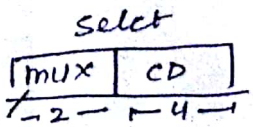


a) select field → 6 bits.
BA → 10 bits.

32 − 10 − 16

b) $16 \rightarrow 2^4 = 4$ bits/line.

c) $6 - 4 = 2$ bits/line



(mux inputs)

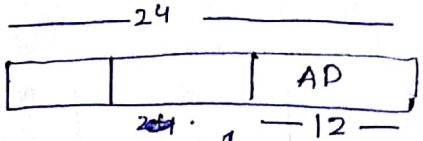
Scanned by CamScanner

48. CM is 4096 × 24 bits each.

a) how many bits are there in CAR.

(b) how many bits are there in each of the 4 input shown going into the mux.

(c) What are the no. of inputs in each mux and how many mux are needed.



Ans. a) $4096 \rightarrow 2^{12} \rightarrow 12$ bits [∵ CAR holds address]

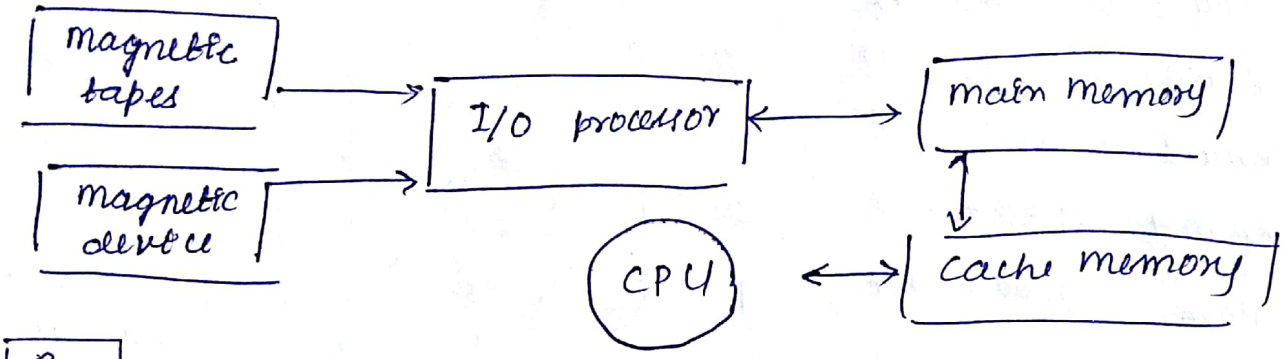
b) 12 bits.

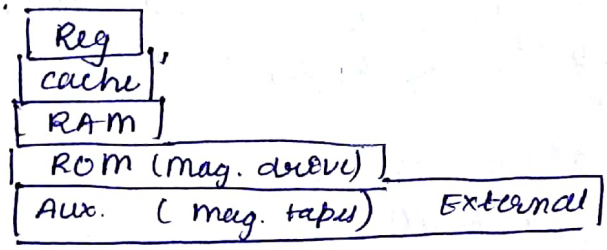
c) 4 inputs in each mux.
12 mux are required.

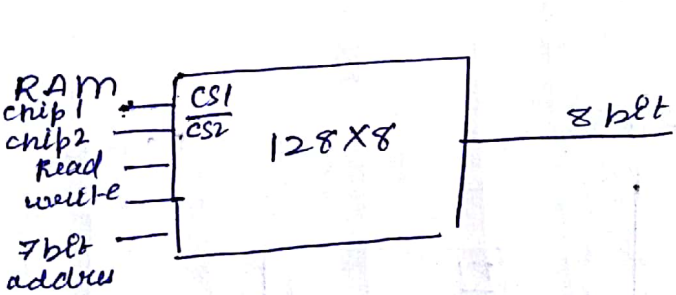
Scanned by CamScanner

Memory Organization : (UNIT-4)

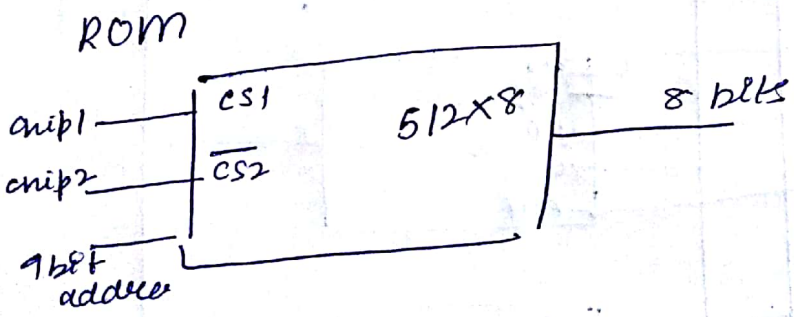
memory hierarchy is to obtain the highest possible access speed while minimizing the total cost of the memory system.







we have to specify write to Read & write



we only read from ROM hence no need to specify Read or write

* Memory address MAP.

Address space assignment to each memory chip.

Ex. 512 bytes of RAM and 512 bytes of ROM.

we have 128 X 8 bytes of RAM. In market

∴ to have a RAM of 512 bytes we will have.

4 [128X8] bytes of RAM chip.

Scanned by CamScanner

128 × 8 J → means 7 bits ~~to~~ addressing line for RAM
512 × 8 J → means 7 bits for addressing ROM.

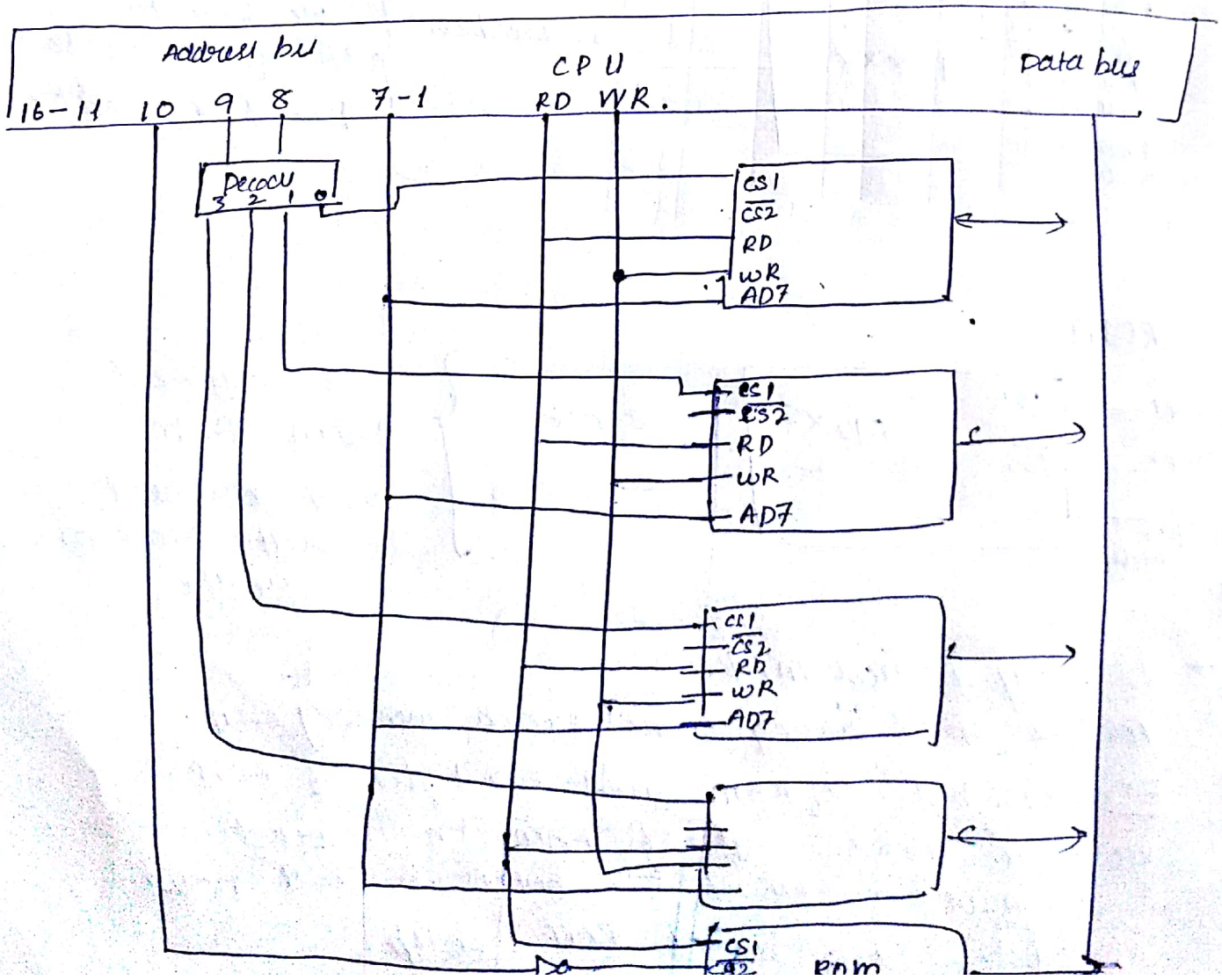
Component	Hexa address	Address bus. 10	9	8	7	6	5	4	3	2	1
RAM 1	0000-007F	0	0	0	x	x	x	x	x	x	x
RAM2	0080-00FF	0	0	1	x	x	x	x	x	x	x
RAM3	0100-017F	0	1	0	x	x	x	x	x	x	x
RAM4	0180-01FF	0	1	1	x	x	x	x	x	x	x
ROM	0200-03FF	1	x	x	x	x	x	x	x	x	x

this bit will tell us about which memory is to be used.

these bits tell us about which chip is to be used.

used to address RAM

used to address ROM.



Scanned by CamScanner

[CS1 — ROM is acting as read signal]

a) How many 128 × 8 ram chips are needed to provide a memory capacity of 2048 bytes

b) How many lines of the address bus must be used to access 2048 bytes of memory. How many of these line will be common to all chips

c) How many line must be decoded for chip select specify the size of decoder

d) Draw the memory address map for the above scenario along with the connectivity to CPU.

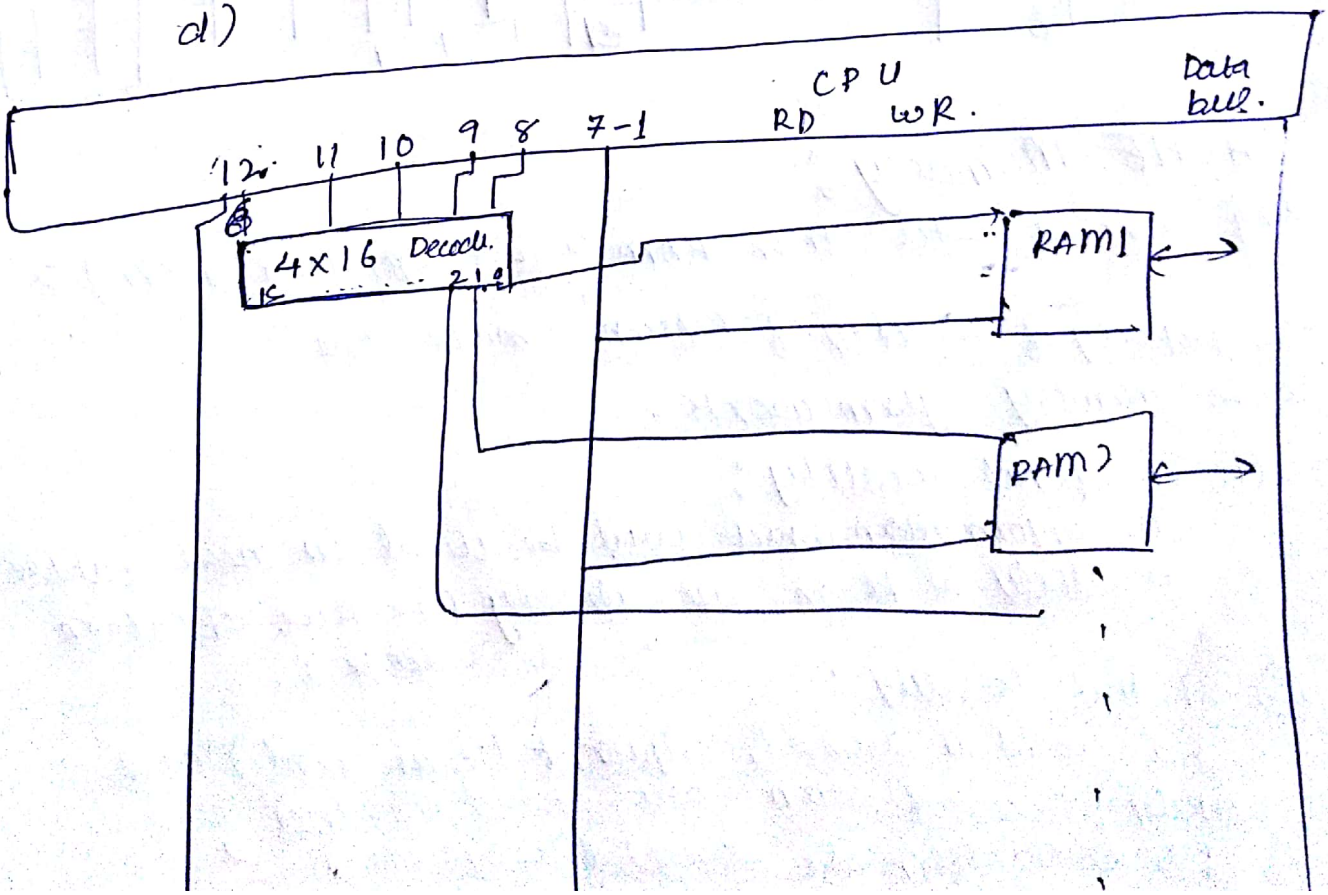
Ans. a) 16 RAM chips. 2^4

b) 11 address line in total. $(7+4)$

7 address line will be common.

c) 4 bits 4 × 16 Decoder.

d)



Scanned by CamScanner

(d).

Component	Hexa address	11	10	9	8	7	6	5	4	3	2	1
RAM 1		0	0	0	0	x	x	x	x	x	x	x
RAM 2		0	0	0	1	x	x	x	x	x	x	x
RAM 3		0	0	1	0	x	x	x	x	x	x	x
RAM 4		0	0	1	1	x	x	x	x	x	x	x
RAM 5		0	1	0	0	x	x	x	x	x	x	x
RAM 6		0	1	0	1	x	x	x	x	x	x	x
7		0	1	1	0	x	x	x	x	x	x	x
8		0	1	1	1	x	x	x	x	x	x	x
9		1	0	0	0	x	x	x	x	x	x	x
10		1	0	0	1	x	x	x	x	x	x	x
11		1	0	1	0	x	x	x	x	x	x	x
12		1	0	1	1	x	x	x	x	x	x	x
13		1	1	0	0	x	x	x	x	x	x	x
14		1	1	0	1							
15		1	1	1	0							
16		1	1	1	1							

CACHE Memory:

Why cache is faster than RAM? (is it DRAM or SRAM)?

The property of locality of Reference makes the cache memory system work.

① Temporal locality:
The information which will be used in near future is likely to be in use already (Ex. Reuse of info in loops).

② Spatial locality:
If a word is accessed, adjacent (near word) are likely accessed soon are stored together; (instructions are executed sequentially)

Cache is fast small capacity memory that should hold those information which are most likely to be accessed.

Hit → if we ~~to~~ find the data in cache

miss → if data not found in cache.

Ex. Hit Ratio = 80%

Effective access time = ?

T_c → time to search in cache

T_m → time to search in RAM.

$$EAT = \underset{\text{hit ratio}}{h} \times T_c + \underset{\text{miss ratio}}{(1-h)}(T_c + T_m)$$

$$= 0.80 \times 20ns + 0.20(20ns + 1000ns)$$

$$= 0.80 \times 20 + 0.20(1020)$$

$$=$$

$$=$$

[if given that $T_c = 20ns$, $T_m = 1000ns$]

Cache Incoherence :
Data present in cache is updated but not updated in RAM this will lead to data inconsistency.

Solution :

① Write Trough : Data is updated in cache & RAM both when there is an update in cache.

② Write Back : When the data is going to get removed from ram then it is changed in RAM.

Q. Access time of a cache is 100ns & that of m.m 1000ns. It is estimated that 80% of the memory requests are for read & remaining 20% for write. Hit ratio for read access is 0.9. A write through procedure is used

a) calculate avg. access time of system considering read cycle

b) calculate avg. access time of system for both read & write request.

Scanned by CamScanner

c) Calculate hit ratio taking only into consideration write cycle.

Ans a) Avg. access time (Read) = hit ratio of read * cache access time + miss ratio ($T_c + T_{mm}$)

$= 0.9 * 100 + 0.1 (100 + 1000)$

$= 90 + 110 = 200 ns.$

b) ∵ read operations are performed 80% of time

Avg access for read and write will be

$= \frac{80}{100} \times$ Avg. access time of read $+ \frac{20}{100}$ (Access time mm)

$= 0.8 \times 200 + 0.2 \times 1000$

$= 360.$

or $0.8 \times (200) + 0.2 (T_c + T_m)$

$= 160 + 0.2 (1100)$

$= 380 ns.$

c).

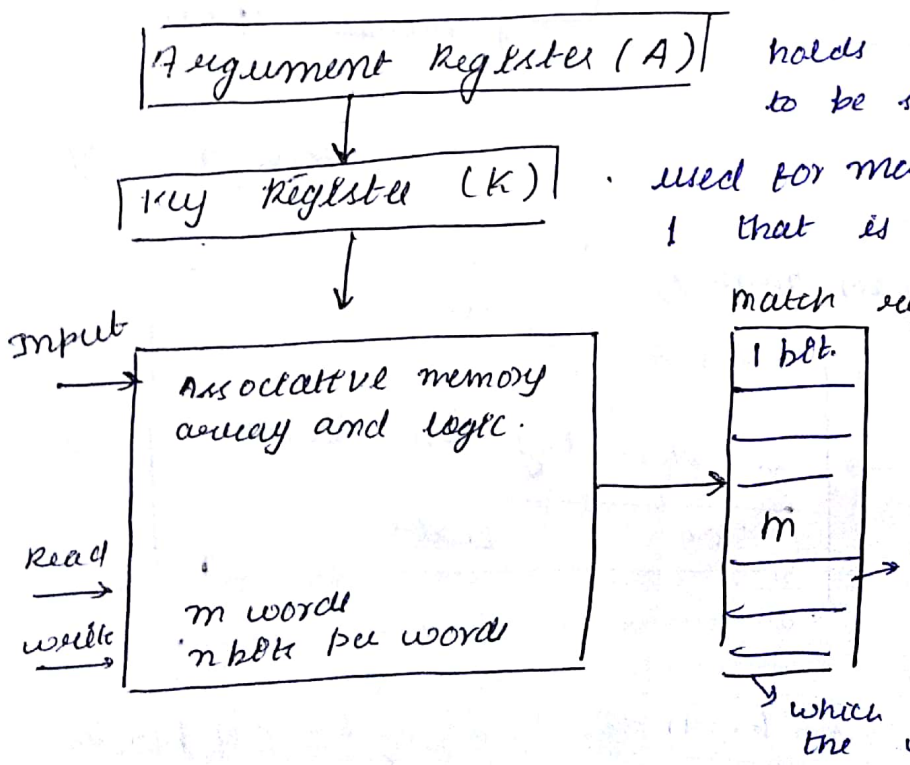
Associative mapping, Cache mapping.

Scanned by CamScanner

ASSOCIATIVE MEMORY:

* Also called content addressable memory (CAM).
* Accessed by the content of the data rather than by an address.
* Now search is parallel → all location/cells in CAM is searched parallelly.

But it expensive (as compared to cache) as it needs to have that parallel logic in it.



A → holds the argument that is to be searched.

K → used for masking. (make those bits that is to be considered or searched)

match reg. → Tells us which word is to be searched.

each word is of n bits

each bit to specify wether to consider the correspondi word or not.

A → n bits

K → n bits.

which word is matching with the words.

It is working as a faster memory than cache having parallel logic build in it.

Scanned by CamScanner

Memory and cache Mapping Technique.

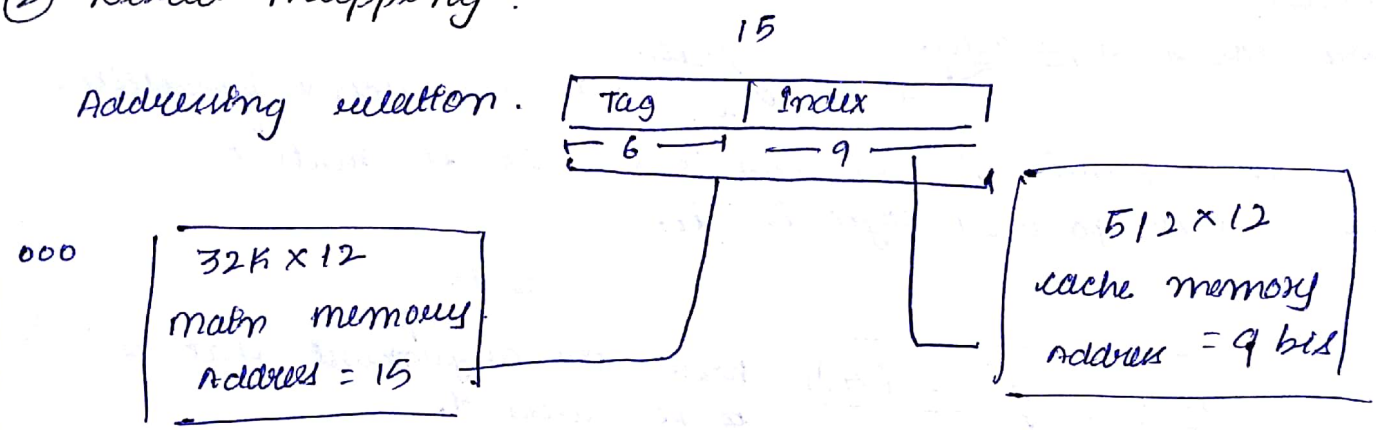
1. Associative mapping
2. Direct mapping
3. Set-associative mapping.

1. Associative mapping :

It has a extra CAM for mapping the content of memory into the cache

2. Direct mapping :

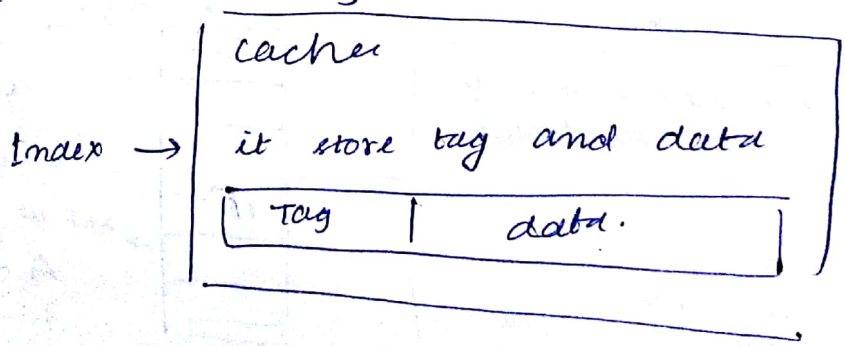
Addressing relation.

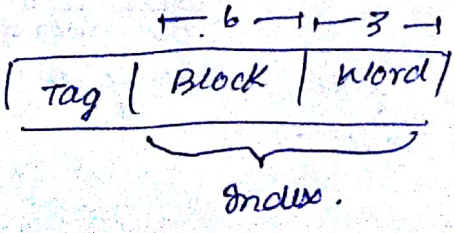


cache memory

Index → to search in cache.

Index + tag → to search in main memory





$2^6 = 64$ blocks

$2^3 = 8$ bits.

Scanned by CamScanner

Set associative.

Each memory block has a set of locations in the cache to load.

Here we ↑ the size of word in the memory.

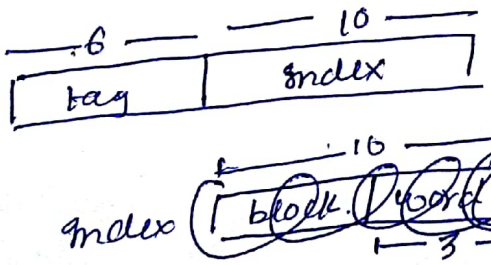
4s. A digital computer has a memory unit of 64K × 16 and a cache memory of 1K words. cache uses direct mapping with a block size of 4 words. 1) how many bits are there in the tag, index, block. and word field for the address format.

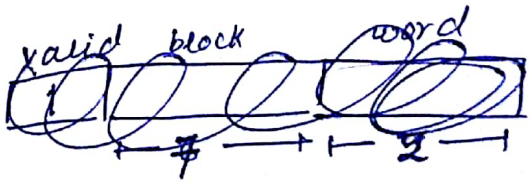
ii) How many bits are there in each word of cache & how are they divided into function, including valid bit.

iii) How many blocks can cache accomodate.

1) m.m $64K = 2^6 \times 2^{10} = 2^{16}$ 16 bits

cache $1K = 2^{10} = 10$ bits.

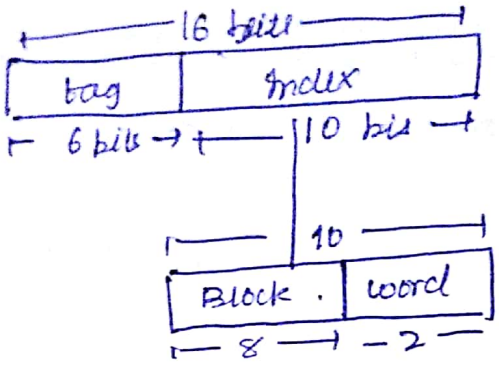




ii)

iii) $2^7 = 128$ blocks

mm
64K x 16

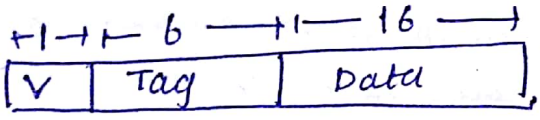


cache
1K words.

1)

we have 4-words → 2^2-bits

2)



∵ data is from mm & data is of 16 bits from mm

3) Block = 8 bits so, we have 2^8 = 256 blocks.

Scanned by CamScanner