

BOOLEAN LOGIC

10/10/11

⇒ BOOLEAN LOGIC -

x	y	f ₀	f ₁	f ₂	f ₃	f ₄	f ₅	f ₆	f ₇	f ₈	f ₉	f ₁₀	f ₁₁	f ₁₂	f ₁₃	f ₁₄	f ₁₅
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Boolean logical ideas are characterised into three ways:

- ① Two funⁿ can produce constant 0, 1
- ② four funⁿ with unary operation (complement transfer).
- ③ Ten funⁿ with binary operation (AND, OR, NAND, NOR, EXOR, EX-NOR, INHIBITION, IMPLICATION)

f ₀ = 0	NULL OPERATION
f ₁ = x.y	AND OPER.
f ₂ = x.y'	INHIBITION (x but not y) (x/y)
f ₃ = x	TRANSFER OPERATION
f ₄ = x'y	INHIBITION (y but not x) (y/x)
f ₅ = y	TRANSFER OPER.

$$\begin{aligned}
 f_6 &= XY' + X'Y = X \oplus Y && \text{EX-OR (X OR Y, BUT NOT BOTH)} \\
 f_7 &= X + Y = && \text{OR OPERATION} \\
 f_8 &= (X + Y)' = X \downarrow Y && \text{NOR OPERATION} \\
 f_9 &= XY + X'Y' = X \odot Y && \text{EX-NOR (X EQUAL Y) IDENTITY EQUIVALENCE} \\
 f_{10} &= Y' && \text{COMPLEMENTARY} \\
 f_{11} &= X + Y' = XY && \text{IMPLICATION (IF Y THEN X)} \\
 f_{12} &= X' && \text{COMPLEMENTARY} \\
 f_{13} &= X' + Y = X \supset Y && \text{IMPLICATION (IF X THEN Y)} \\
 f_{14} &= (XY)' = X \uparrow Y && \text{NAND OPERATION} \\
 f_{15} &= 1 && \text{IDENTITY OPER.}
 \end{aligned}$$

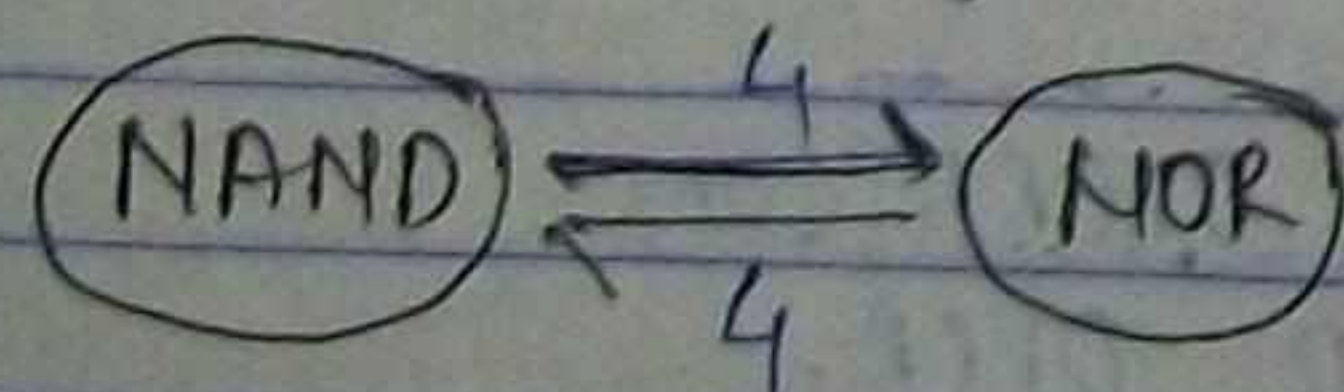
NOTE -

- To get a logic gate from a logic operation it must be either commutative or associative.

- Inhibition / Implication neither commutative nor associative.

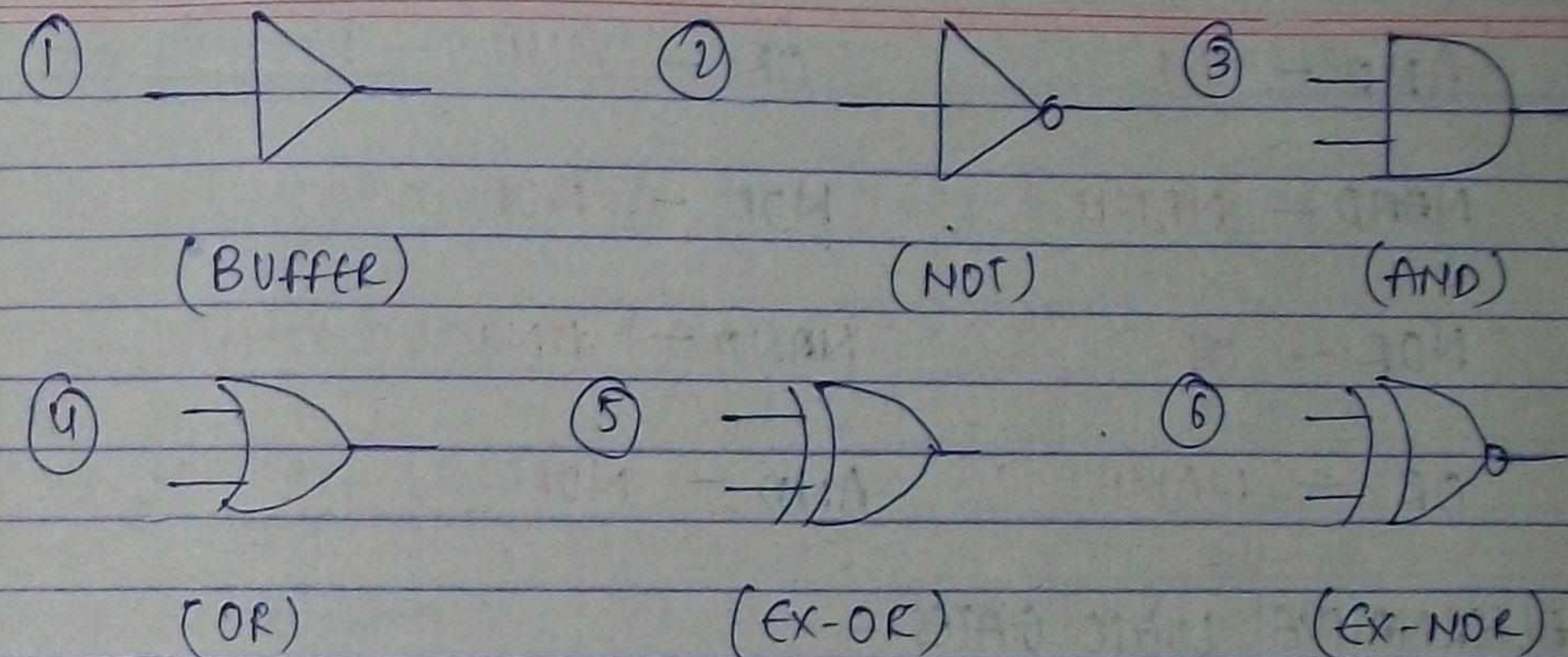
- NAND / NOR commutative but not associative.

- NAND / NOR universal gates



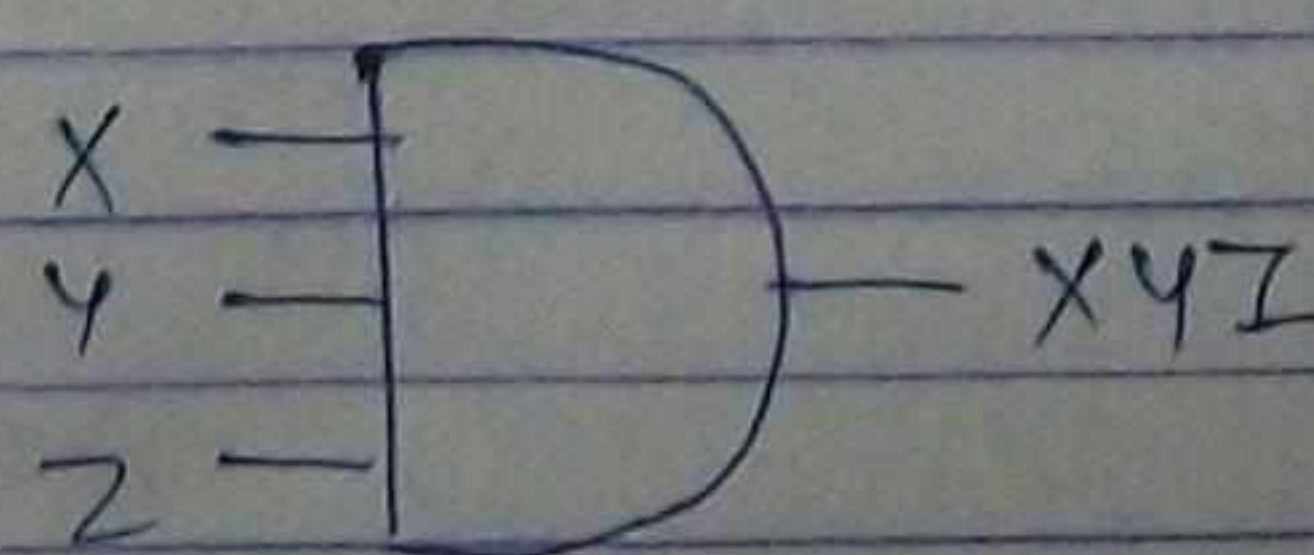
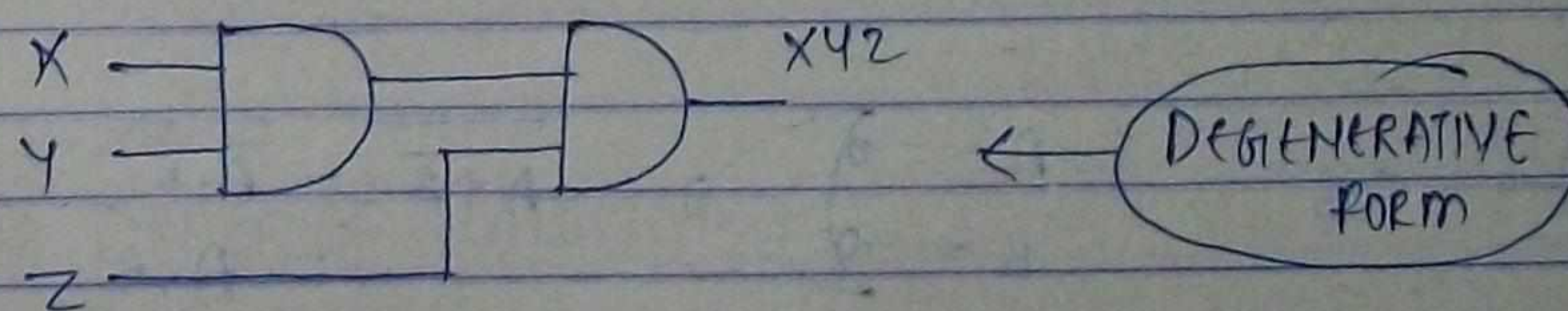
IMPLEMENT

NOT	1	1
AND	2	3
OR	3	2
EX-OR	4	5
EX-NOR	5	4



⇒ DEGENERATIVE FORMS -

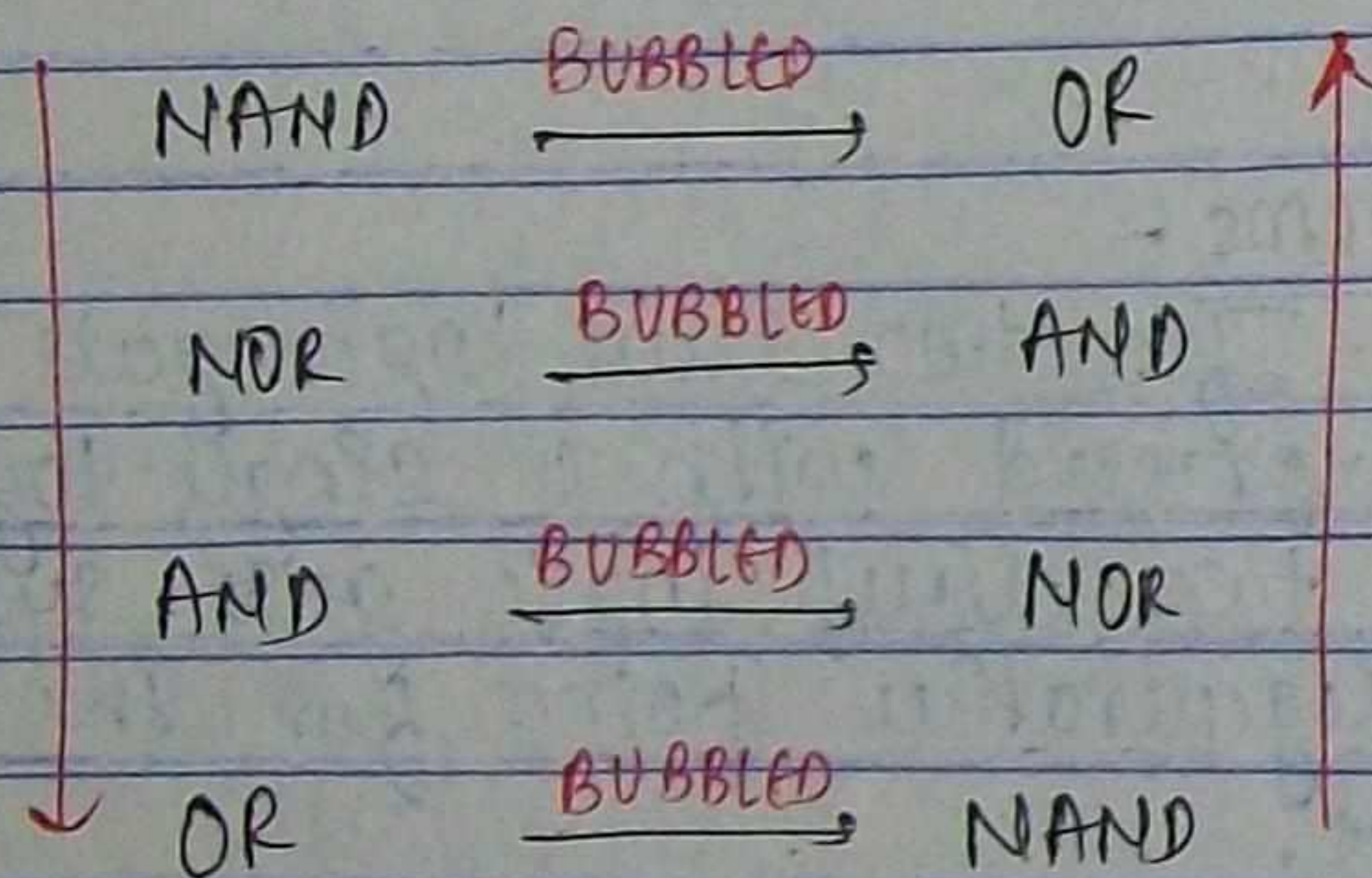
If a two level logic gate system o/p is expressed with a single logic gate then that two level logic gate system is known as degenerative form for the single logic gate.



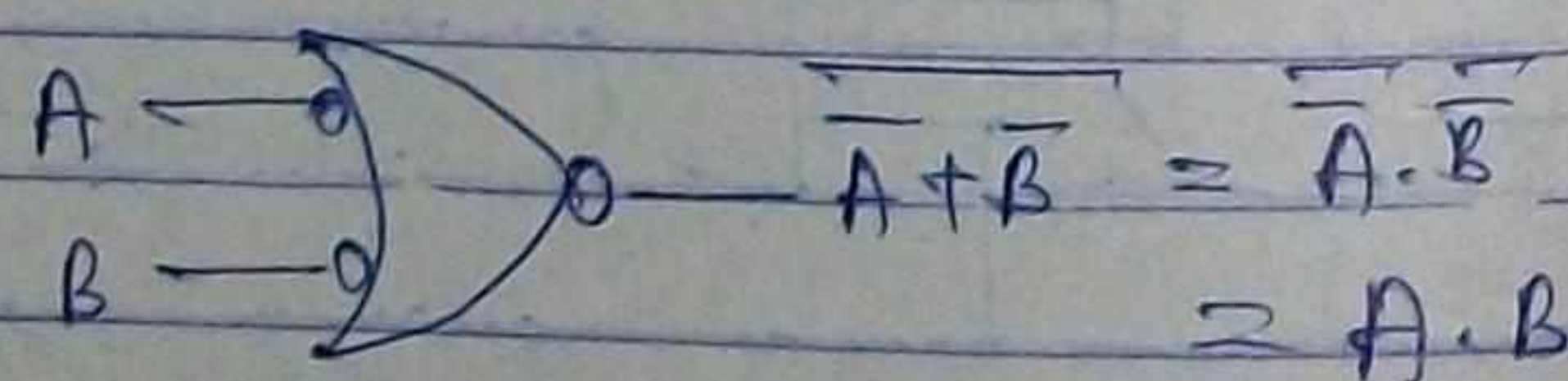
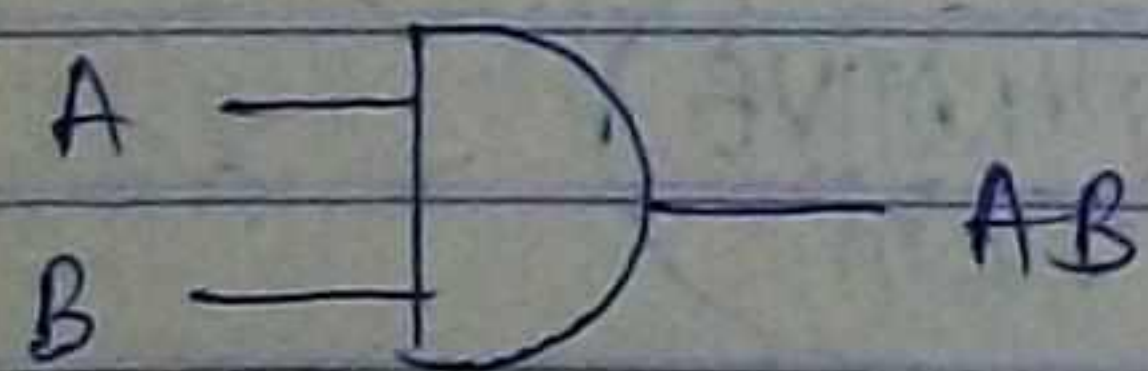
NON-DEGENERATIVE FORM -

AND - OR
 NAND - NAND
 NOR - OR
 OR - NAND
 OR - AND
 NOR - NOR
 NAND - AND
 AND - NOR

⇒ ALTERNATIVE LOGIC GATES -



Ex - AND



⇒ DUALITY -

1. Interchange ($\cdot \rightarrow +$) or ($+ \rightarrow \cdot$)
2. Interchange ($0 \rightarrow 1$) or ($1 \rightarrow 0$)

⇒ BOOLEAN LAW -

AND	$\left\{ \begin{array}{l} x \cdot x = x \\ x \cdot 0 = 0 \\ x \cdot 1 = x \\ x \cdot \bar{x} = 0 \end{array} \right.$	$x + x = x$	$\left. \right\}$ OR
		$x + 0 = x$	
		$x + \bar{x} = 1$	
		$x + 1 = 1$	

Ex - $f = \bar{x} \cdot y + z(x+y)$, $f^D = ?$

$f^D = \bar{x} + y \cdot z + (x \cdot y)$

⇒ ORDER OF IMPORTANCE -

1. []
2. NOT
3. AND
4. OR

⇒ POSITIVE - NEGATIVE LOGIC -

(+)ve LOGIC (-)ve LOGIC

HIGH $\rightarrow 1$

LOW $\rightarrow 0$

HIGH $\rightarrow 0$

LOW $\rightarrow 1$

Ex = $-7V \rightarrow 1$ = -ve logic
 $-2V \rightarrow 0$

A	B	$f = AB$	A	B	$f = A+B$
0	0	0	1	1	1
0	1	0	1	0	1
1	0	0	0	1	1
1	1	1	0	0	0

⊗ Negative AND is positive OR and vice versa

⊗ Negative NAND is positive NOR and vice versa

⇒ SPECIALITY OF EX-OR GATES -

$$X \oplus Y = X'Y + XY'$$

$$X \oplus X = 0$$

$$X \oplus X' = 1$$

$$X \oplus 1 = X'$$

$$X \oplus 0 = X$$

$$X \oplus Y$$

$$X' \oplus Y$$

$$X \oplus Y'$$

$$= X \odot Y$$

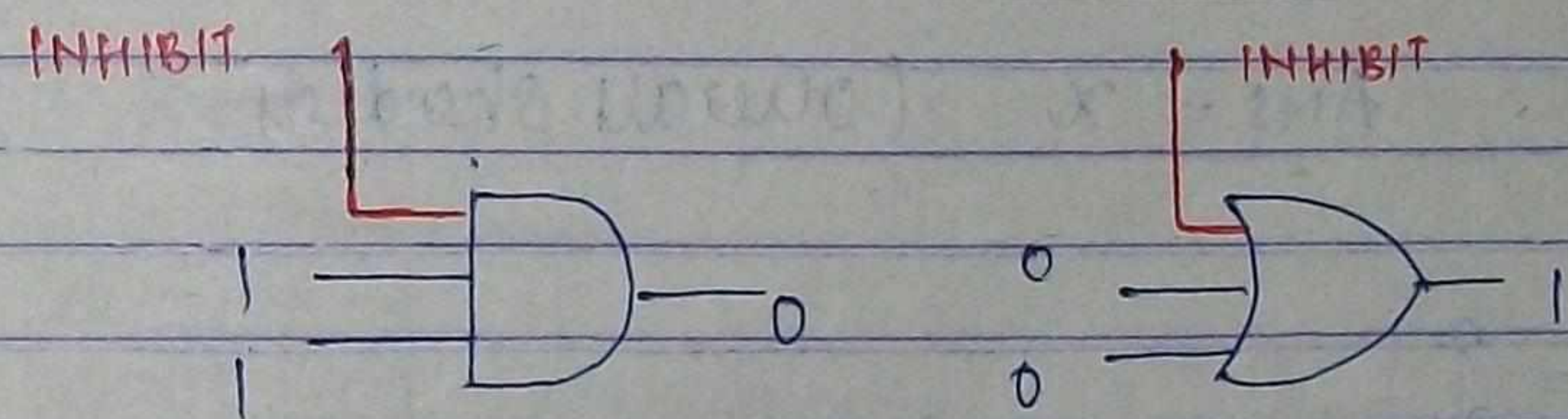
NOTE -

- In the case of symmetrical series of circuit the O/P cat:

O/P of 1st gate = O/P of every odd no. gate.

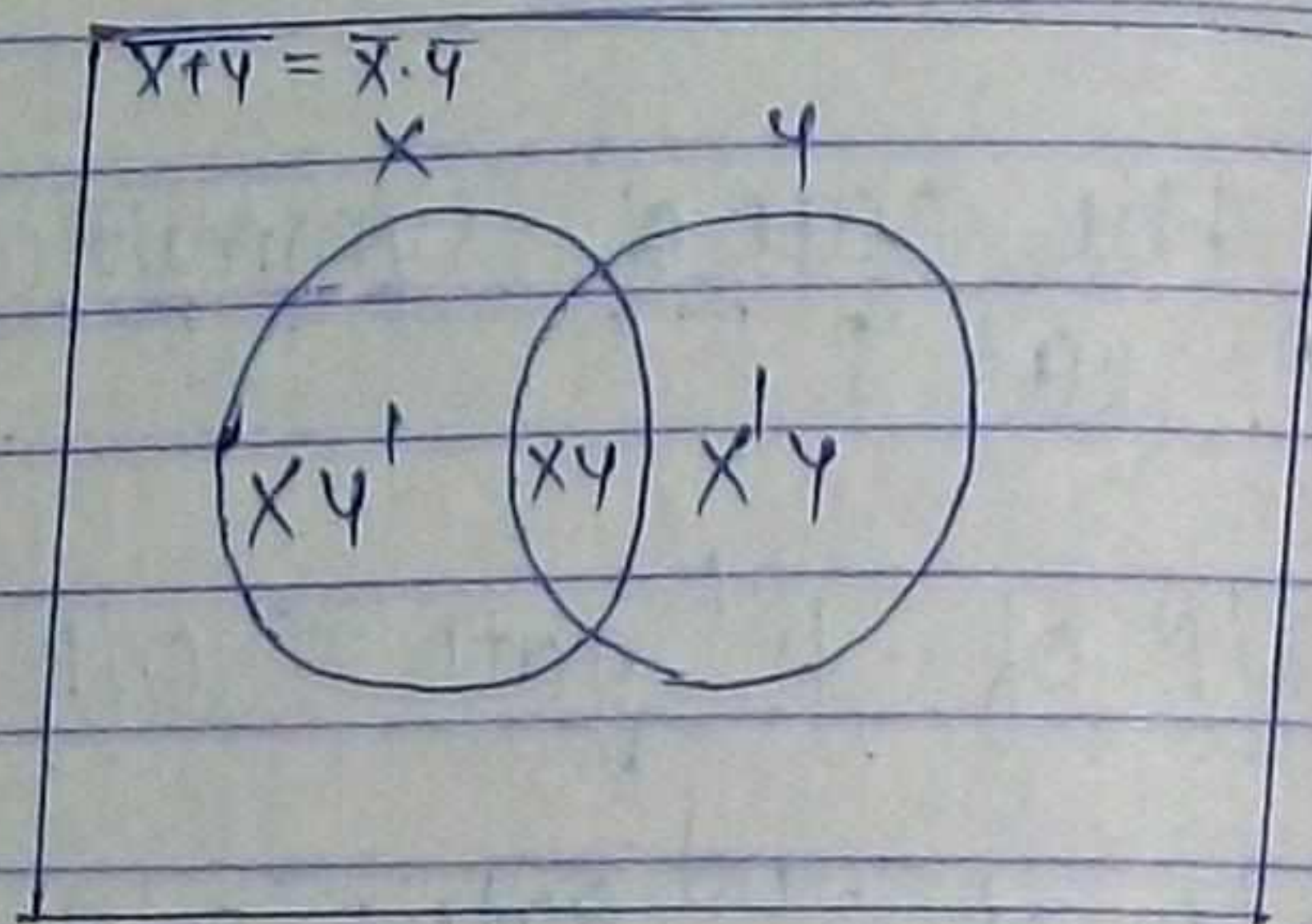
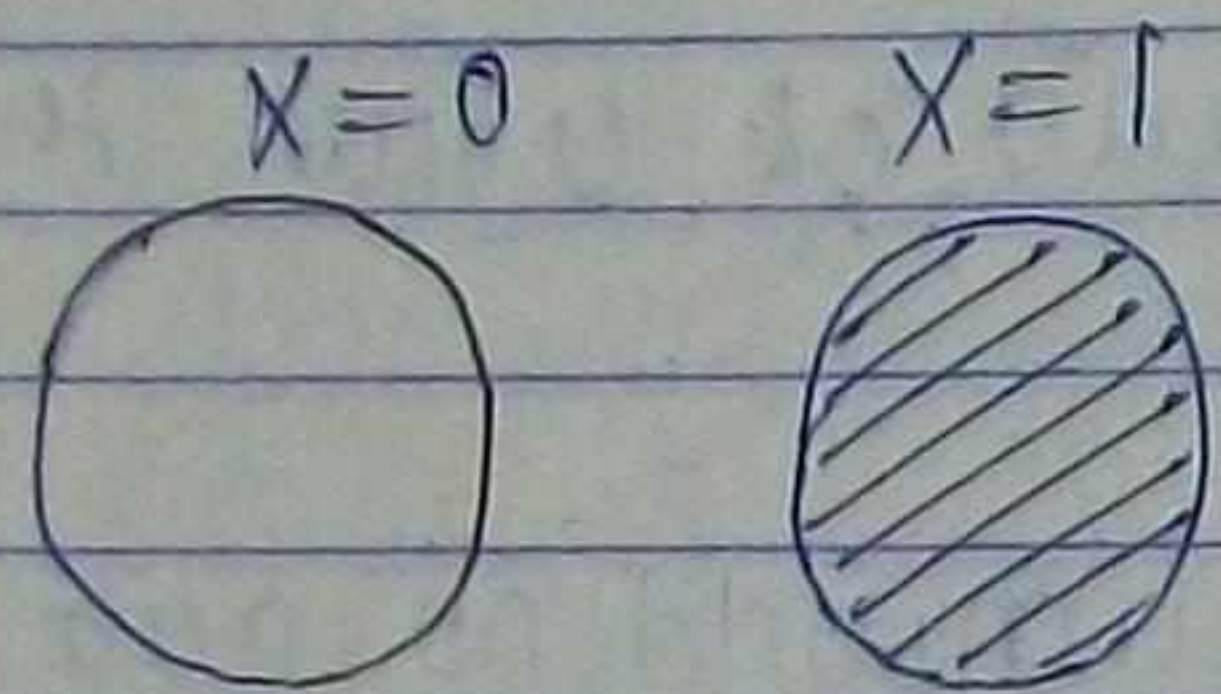
O/P of 2nd gate = O/P of every even no. gate.

⇒ INHIBITING THE LOGIC GATE -

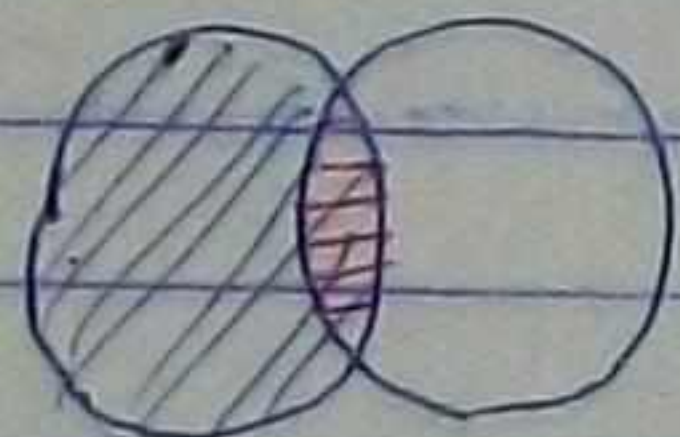


	DISABLE	ENABLE
NAND	L	H
NOR	H	L
AND	L	H
OR	H	L

⇒ VENN DIAGRAM -

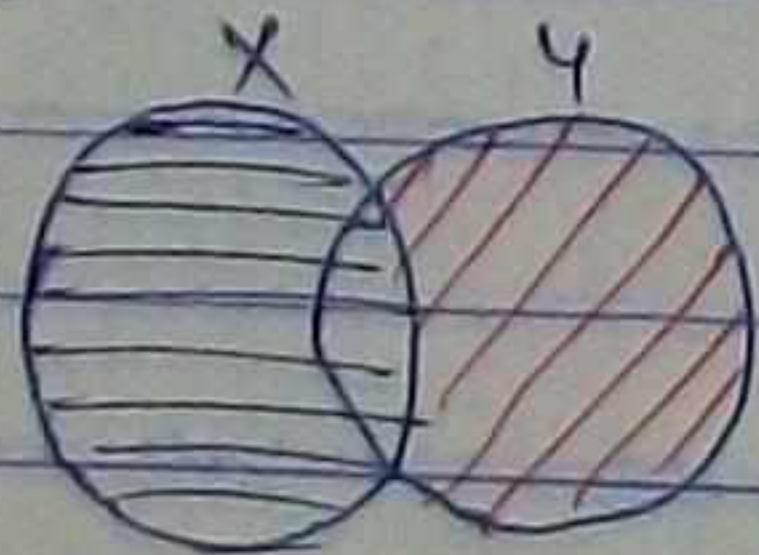


Ex - ① $x + xy = ?$

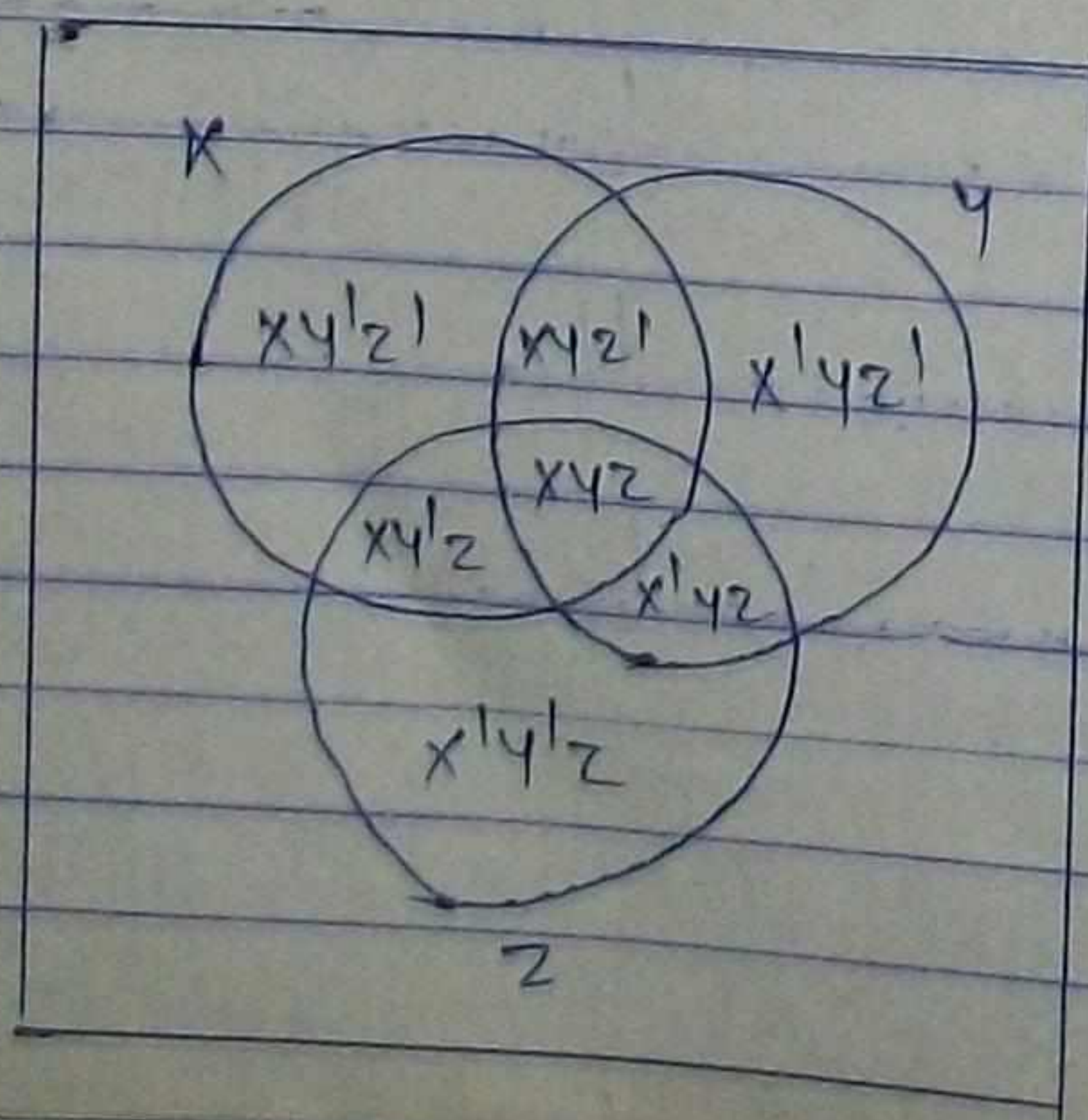


Ans = x (Overall shaded)

② $x + x'y = ?$



Ans = $x + y$ (Overall shaded)



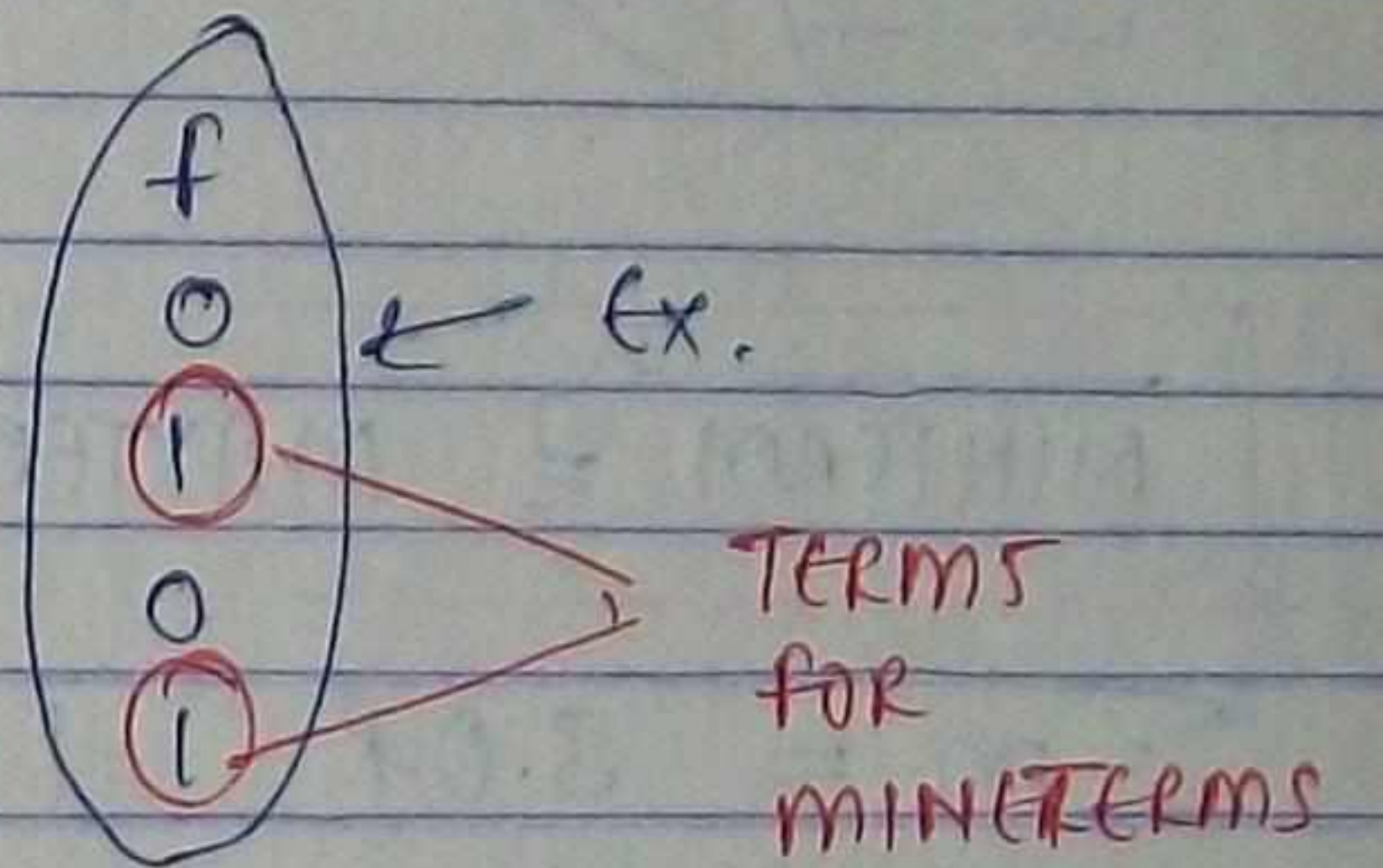
FOR THREE VARIABLE

LOGIC MINIMIZATIONS

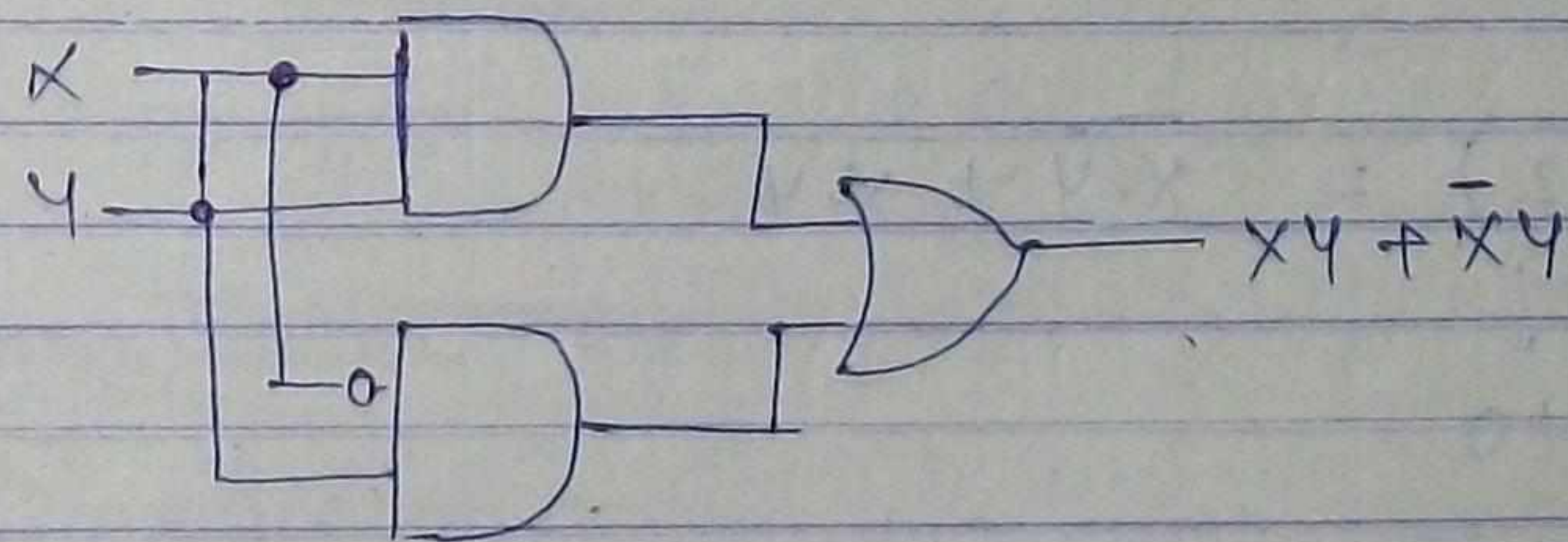
⇒ LOGIC MINIMIZATION -

① SUM OF PRODUCT (SOP) - (MINTERMS)

x	y	minterms
0	0	$\bar{x} \cdot \bar{y}$
0	1	$\bar{x} \cdot y$
1	0	$x \cdot \bar{y}$
1	1	$x \cdot y$

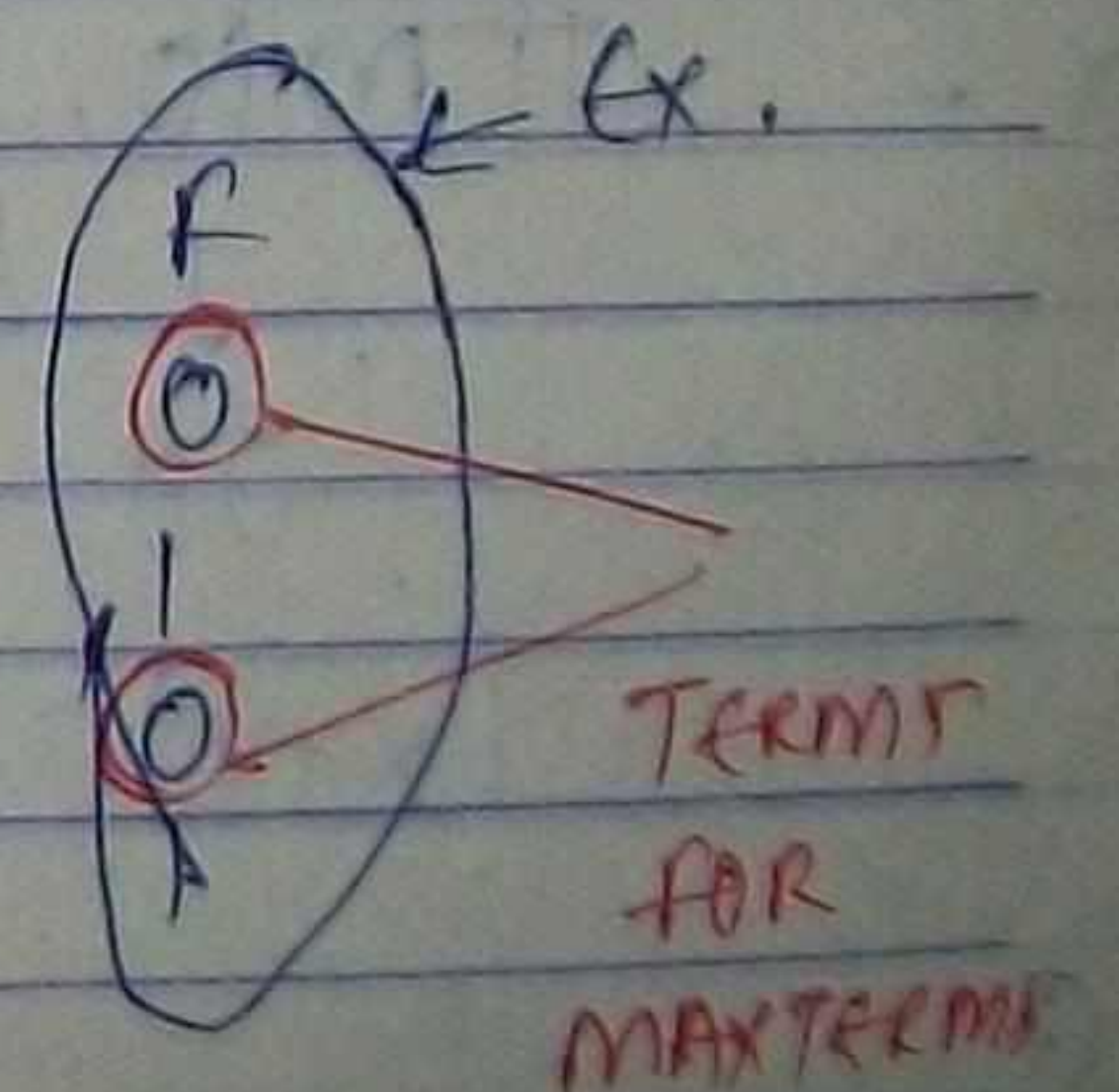


$f = \bar{x}y + xy$

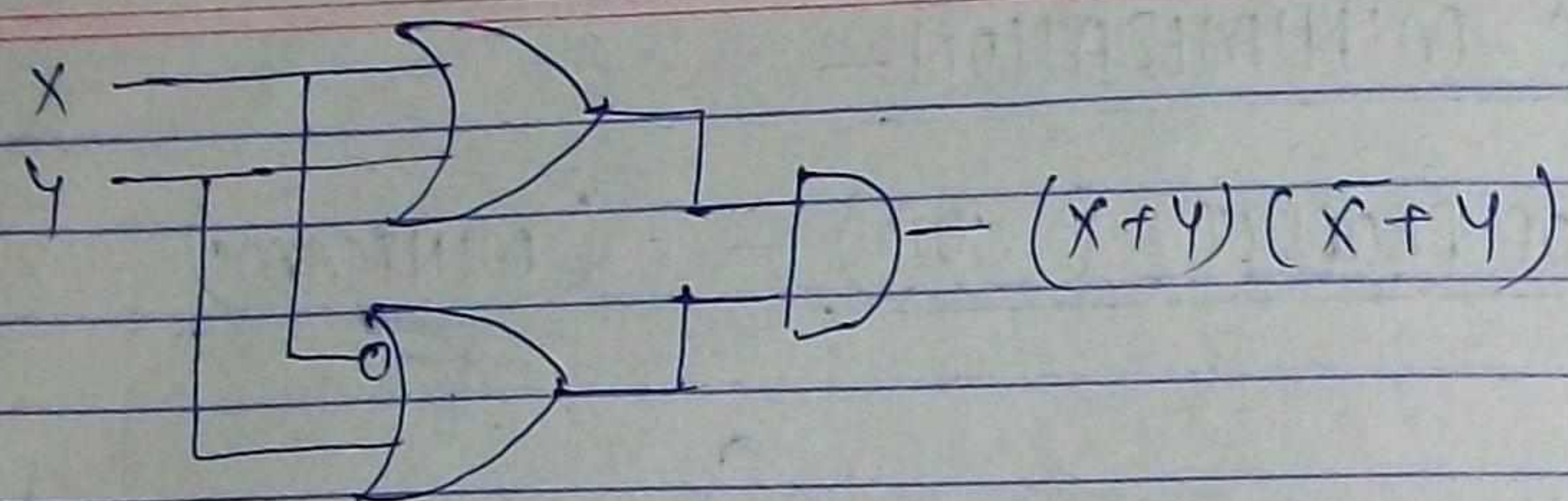


② PRODUCT OF SUM (POS) - (MAXTERMS)

x	y	minterms	maxterms
0	0	$\bar{x} \bar{y}$	$x + y$
0	1	$\bar{x} y$	$x + \bar{y}$
1	0	$x \bar{y}$	$\bar{x} + y$
1	1	xy	$\bar{x} + \bar{y}$



∴ $f = (x + y) \cdot (\bar{x} + \bar{y})$



* MINTERM = MAXTERMS (विपरीत)

$\Sigma m = \text{S.O.P}$ (min term)

$\Pi M = \text{P.O.S}$ (max term)

Ex - $\Sigma m(0, 2) = \bar{x} \cdot \bar{y} + x \cdot \bar{y}$

$\downarrow \quad \downarrow$
 00 10
 $\bar{x} \cdot \bar{y} \quad x \cdot \bar{y}$

Ex - $\Pi M(0, 2) = (x+y)(\bar{x}+y)$

$\downarrow \quad \downarrow$
 00 10
 $(x+y) \quad (\bar{x}+y)$

Ex - $\Sigma m(0, 3) = \Pi M(1, 2)$

- $\Pi M(1, 5) = \Sigma m(0, 2, 3, 4, 6, 7)$

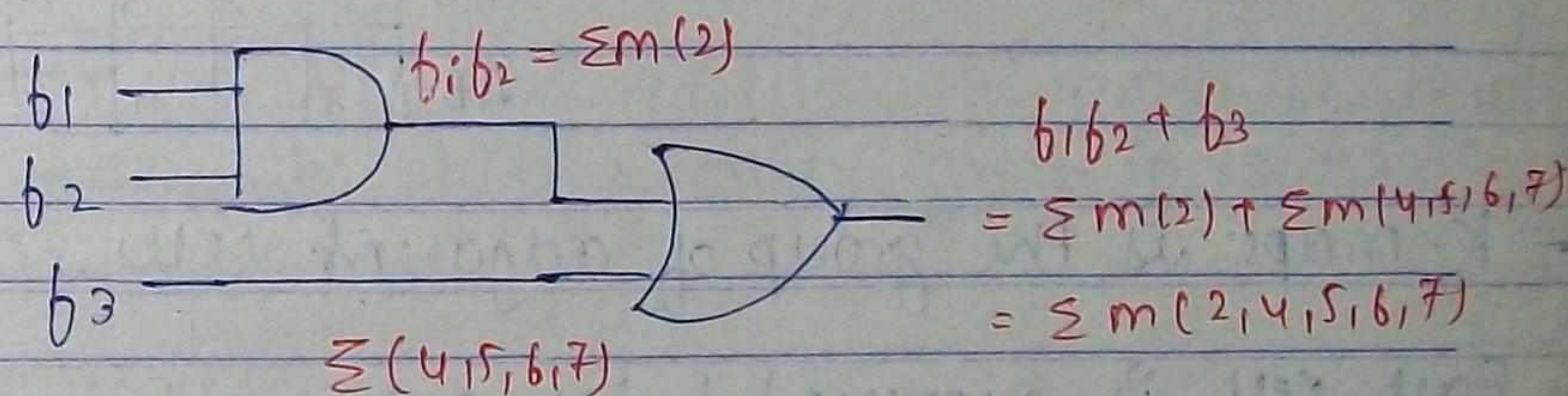
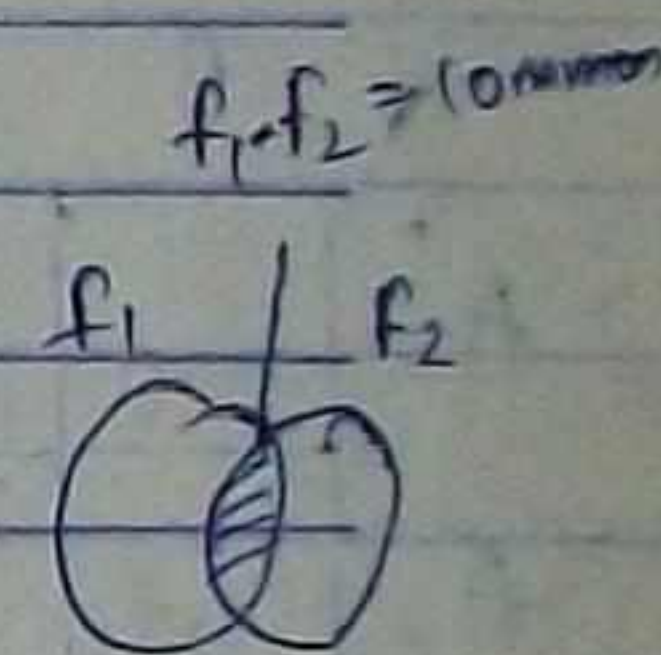
- $\Sigma m(0, 2, 11) = \Pi M(1, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15)$

Ex - $b_1 = \Sigma m(1, 2, 3)$

$b_2 = \Sigma m(0, 2, 4, 5)$

$b_3 = \Sigma m(4, 5, 6, 7)$

$b = ?$



#

$b_1' \cdot b_2 = \Sigma m(0, 4, 5)$

$b_1 b_2' = \Sigma m(1, 3)$

$b = b_1 \oplus b_2 = \Sigma m(0, 2, 4, 5)$

⇒ K-MAPS -

	B'	B
A'		
A		

	B'	B
A'	0	1
A	2	3

	B'C'	B'C	BC	BC'
A'	0	1	3	2
A	4	5	7	6

	C'D'	C'D	CD	CD'
A'B'	0	1	3	2
A'B	4	5	7	6
AB	12	13	15	14
AB'	8	9	11	10

- K-maps is the group of adjacent cells.

- Each cell is represented by group of literals.

- PAIR - 1

QUAD - 2

OCTED - 3

- In a 6 variable quad group
 $= 6 - 4 = 2$ literals.

⇒ REDUNDANT GROUP -

	B'C'	B'C	BC	BC'
A'		1	1	
A		1	1	

REDUNDANT - BC
GROUP

	C'D'	C'D	CD	CD'
A'		1		
A		1	1	1

RED. GROUP - BD.

- Generally to get the minimized circuit it is better to avoid redundant group.

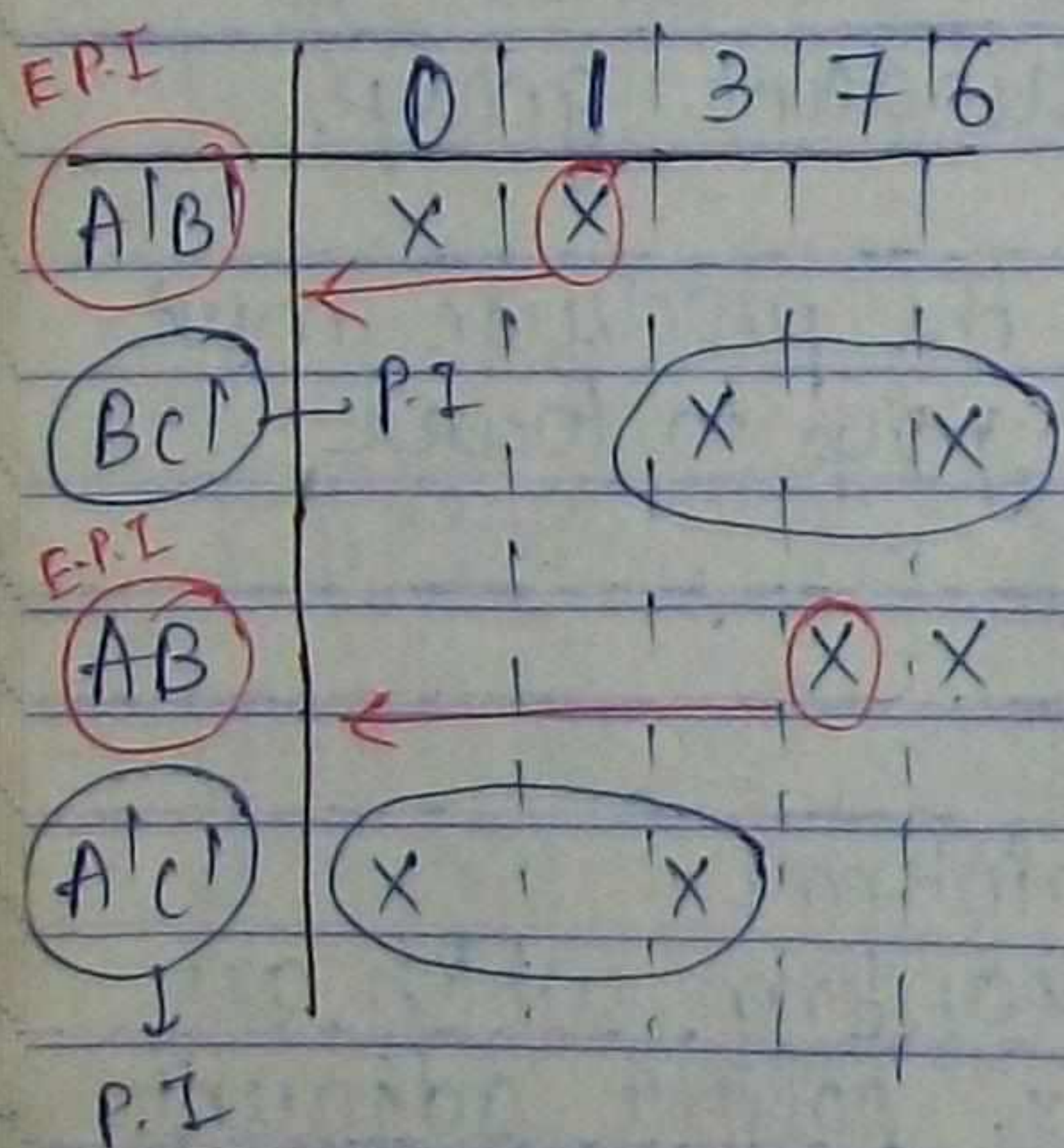
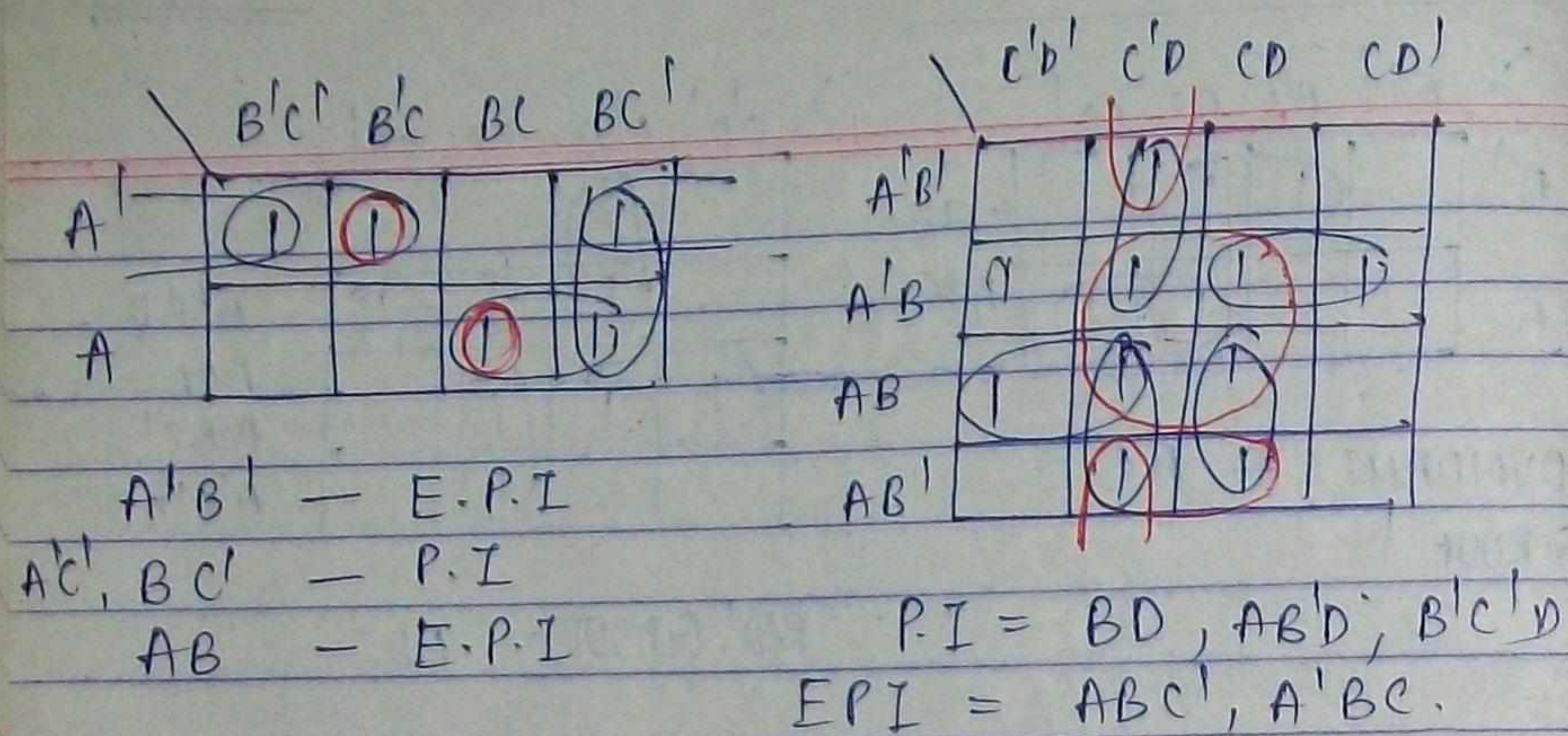
- In the minimization process, the procedure must be from the higher order group to lower order group.

⇒ PRIME IMPLICANT -

Prime Implicant is a ^{minterms} preanterm which are obtained by combining max. possible adjacent cells in K-map.

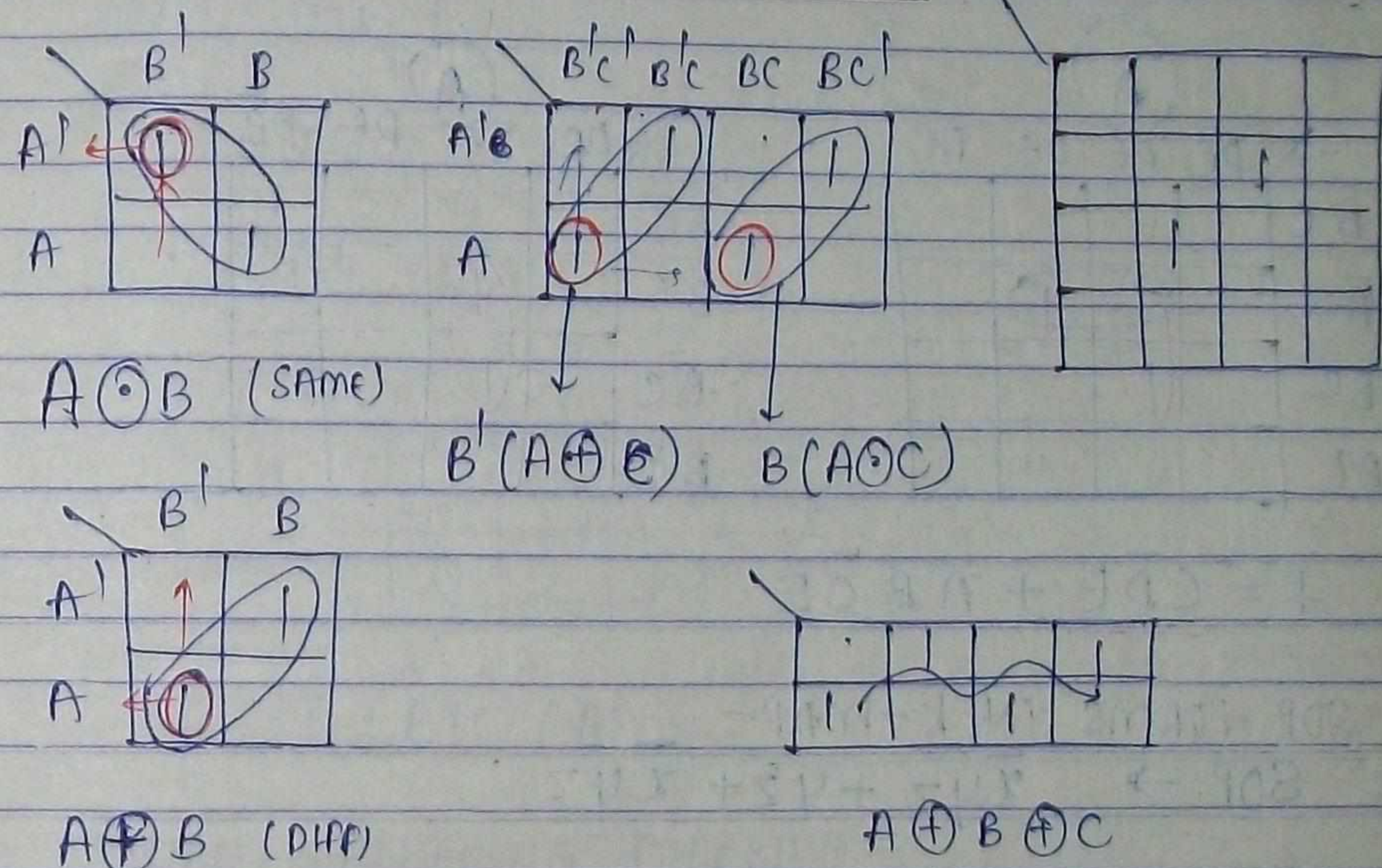
⇒ ESSENTIAL - PRIME IMPLICANT -

- An essential-P.I is a prime implicant in which one or more minterms are uniquely group, which are obtained by only one way in K-map.

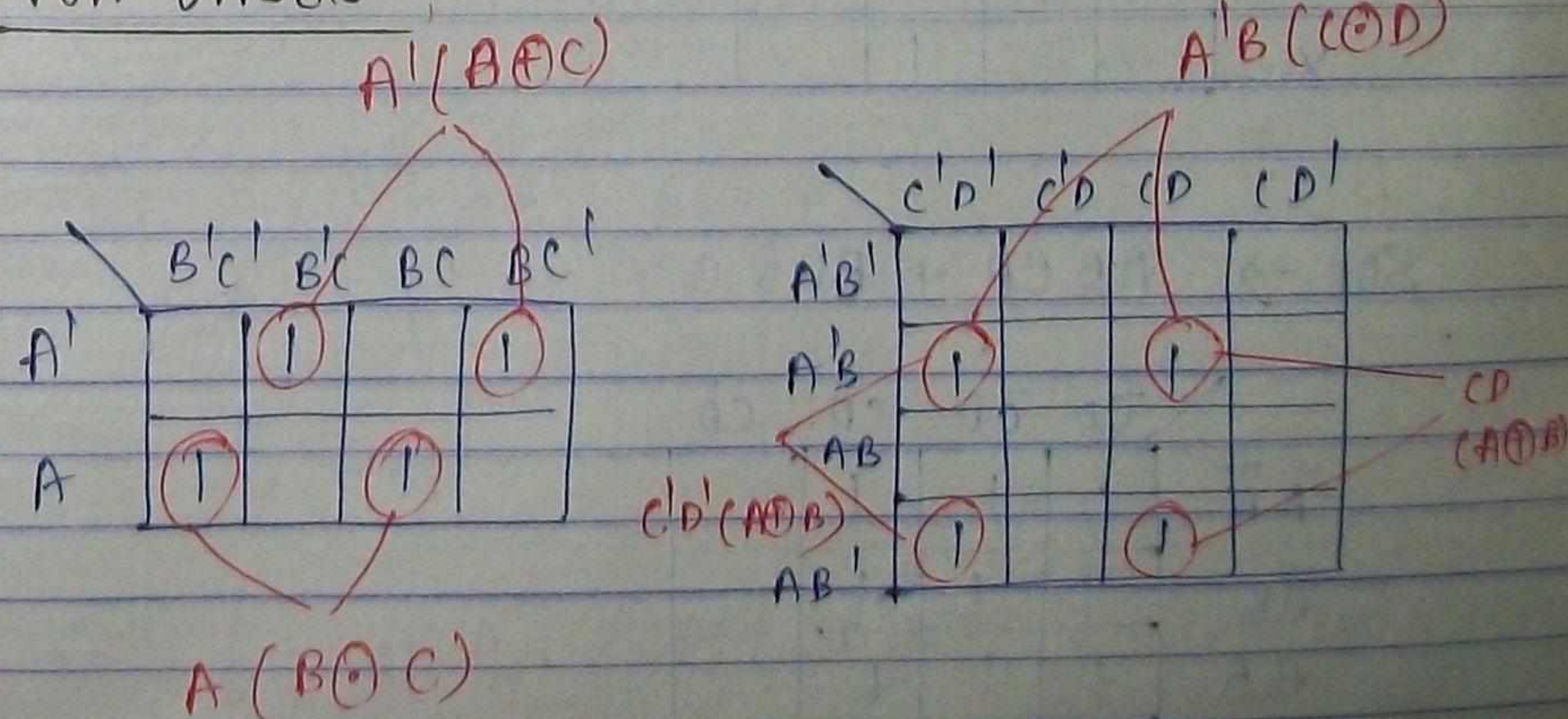


TABLATION METHOD

⇒ HYPER-TECHNIQUE IN K-MAP -



⇒ FOR OFFSETS -



⇒ FIVE VARIABLE K-MAP - (A B C D E)

	A			
	DE	DE	DE	DE
BC				
BC		1	1	
BC		1		
BC				

	A			
	DE	DE	DE	DE
BC				
BC		1		
BC		1		
BC				

$$f = CDE + ABCE$$

⇒ SOP FORMS IN K-MAP -

SOP → $\bar{x}yz + y\bar{z} + x\bar{y}z$

	$\bar{y}z$	$\bar{y}z$	yz	yz
x		1	1	
x			1	

SOP → $ABCD + BD + C$

	$\bar{C}D$	$\bar{C}D$	CD	CD
$\bar{A}\bar{B}$	1	1	1	
$\bar{A}B$	1	1	1	
AB	1	1	1	
$A\bar{B}$	1	1		

⊛ Don't care are taken as 1 if the neighbouring cells are 1.

⇒ POS FORMS IN K-MAP -

$\Pi M(0, 1, 2, 4, 5, 7)$

	$B+C$	$B+C'$	$B'+C'$	$B'+C$
A	0	0		0
A'	0	0	0	

$$f = B \cdot (A+C) \cdot (A'+C')$$

⇒ VARIABLE ENTERING MAPPING -

$f = A'B'CD + A'BCD' + A'BCD + ABCD' + ABC'D'$

	$B'C'$	$B'C$	BC	BC'
A'		0	0	0
A			0	0

∴ $f = A'CD + BCD' + ABD'$ (minimized)

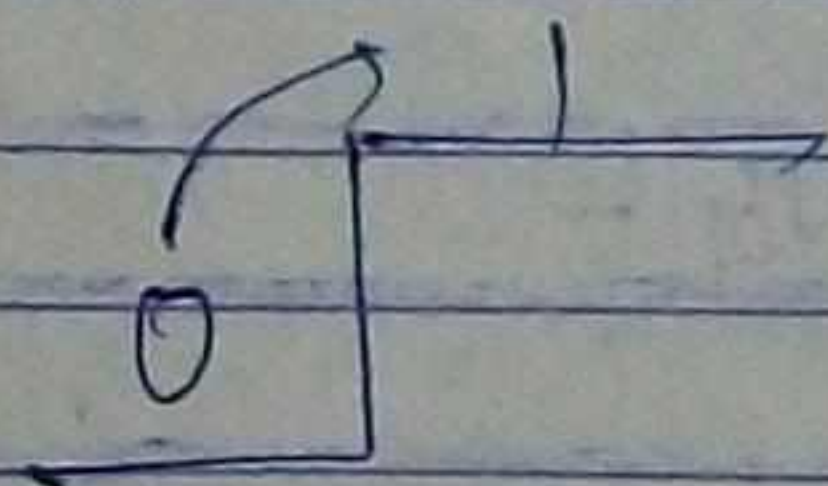
⇒ HAZARDS -

- Because of different propagation delay present in the circuit, then the O/P may not be accurate, to get the accurate O/P hazard free circuit redundant groups are involved in the circuit.

Hazards are of three type:

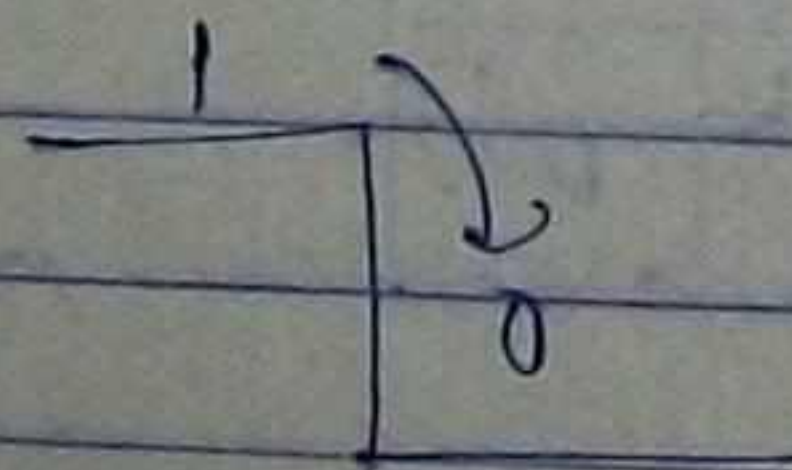
① STATIC '0' HAZARD -

If the O/P changes from 0 → 1 instead of staying at zero, then that is known as static '0' hazard.



② STATIC '1' HAZARD -

Instead of staying the O/P at '1' if it changes to '0' it is said to be static '1' hazard.



③ DYNAMIC HAZARDS -

If the O/P continuously goes on changing then it is known as dynamic hazard.



Ex -

	$B'C'$	$B'C$	BC	BC'
A'		1	1	
A			1	1

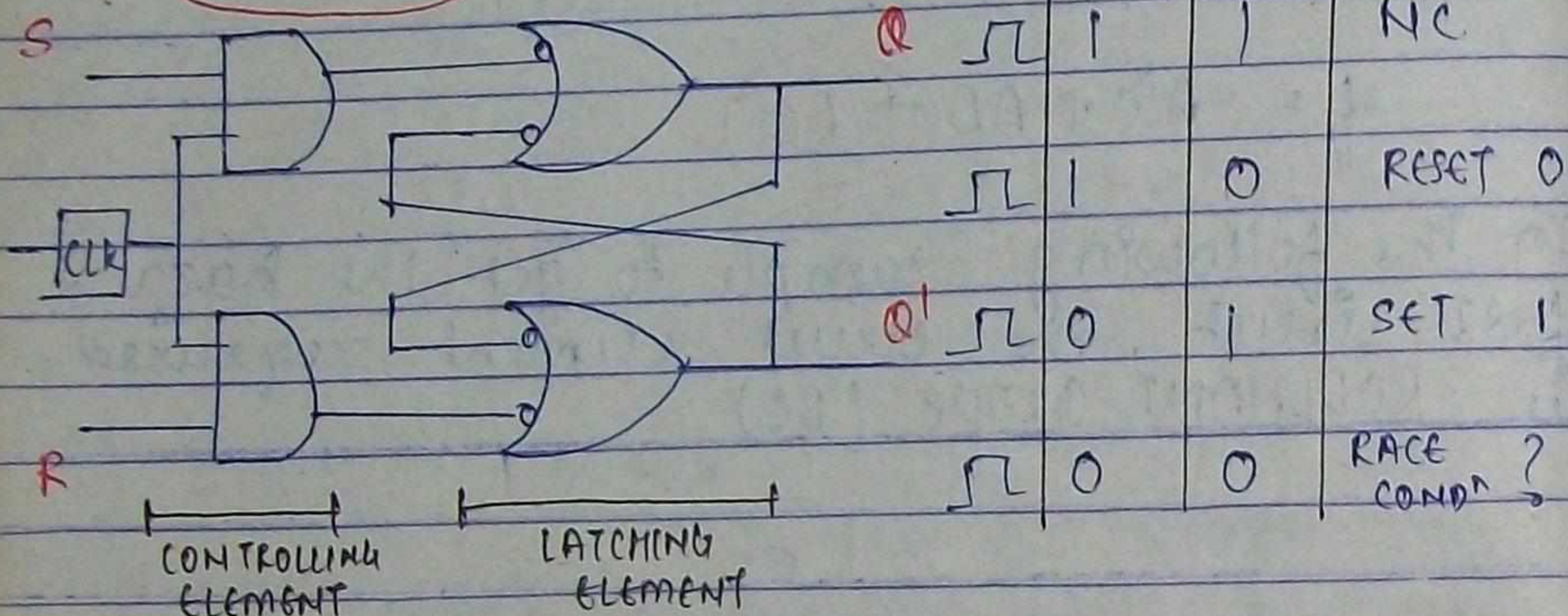
$$f = A'C + AB + \textcircled{BC}$$

In the following example to get the hazard free circuit, the circuit element required is REDUNDANT GROUP (BC).

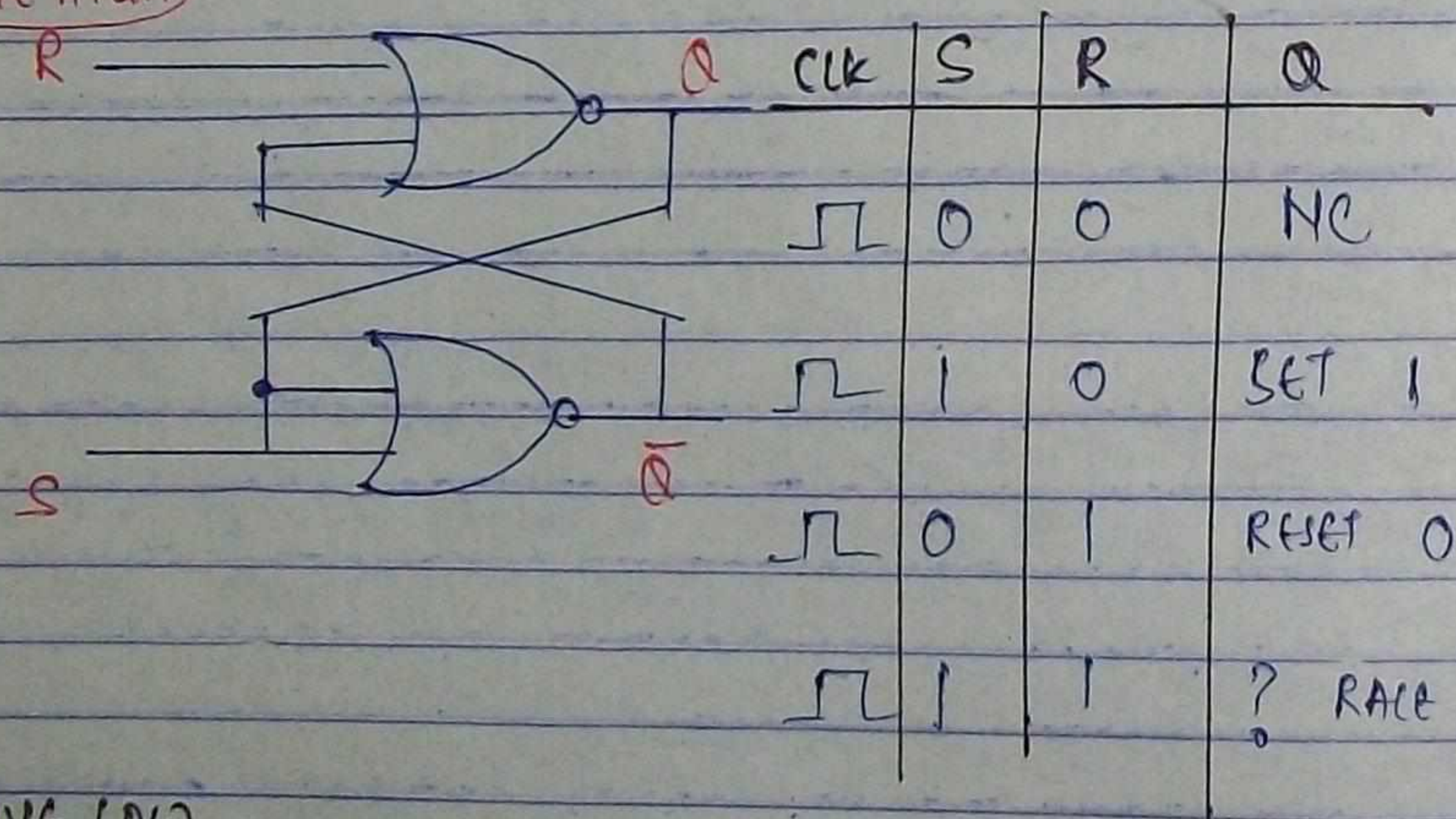
SEQUENTIAL CIRCUIT

⇒ SR-LATCH -

ACTIVE LOW



ACTIVE HIGH



ACTIVE LOW -

$S=0$
 $R=1$

$Q=1$ SET
 $Q=0$ RESET

ACTIVE HIGH -

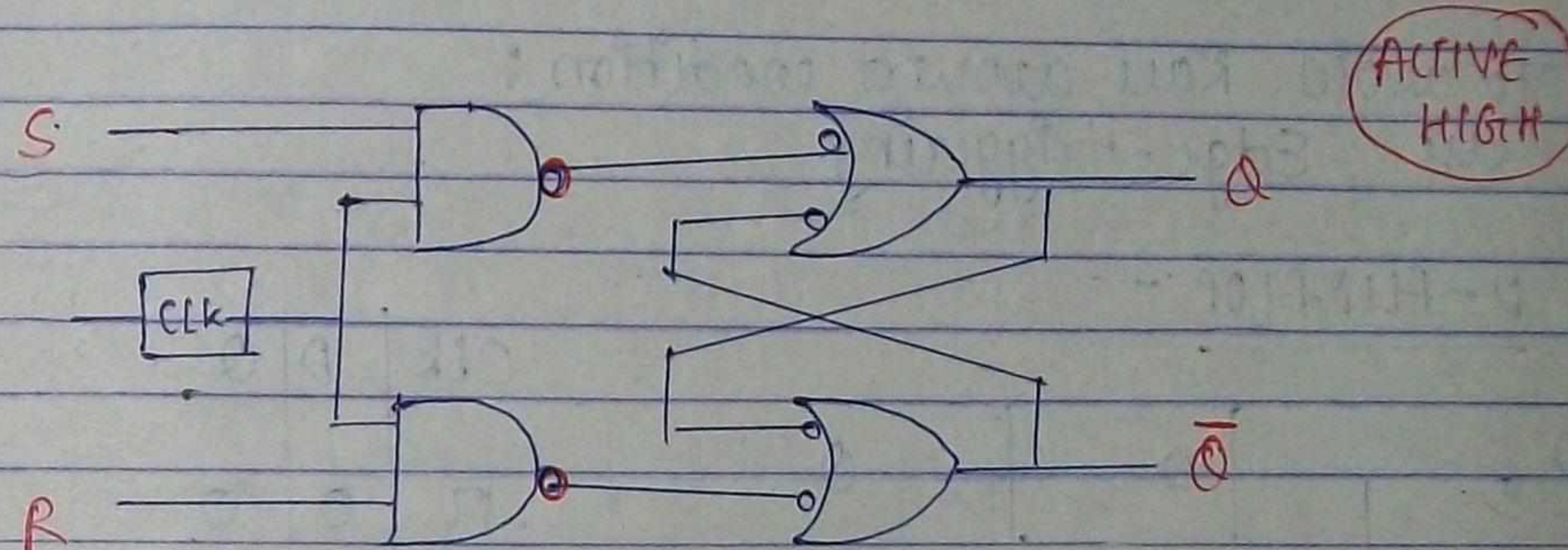
$S=1$
 $R=0$

$Q=1$
 SET

$S=1$ $R=1$

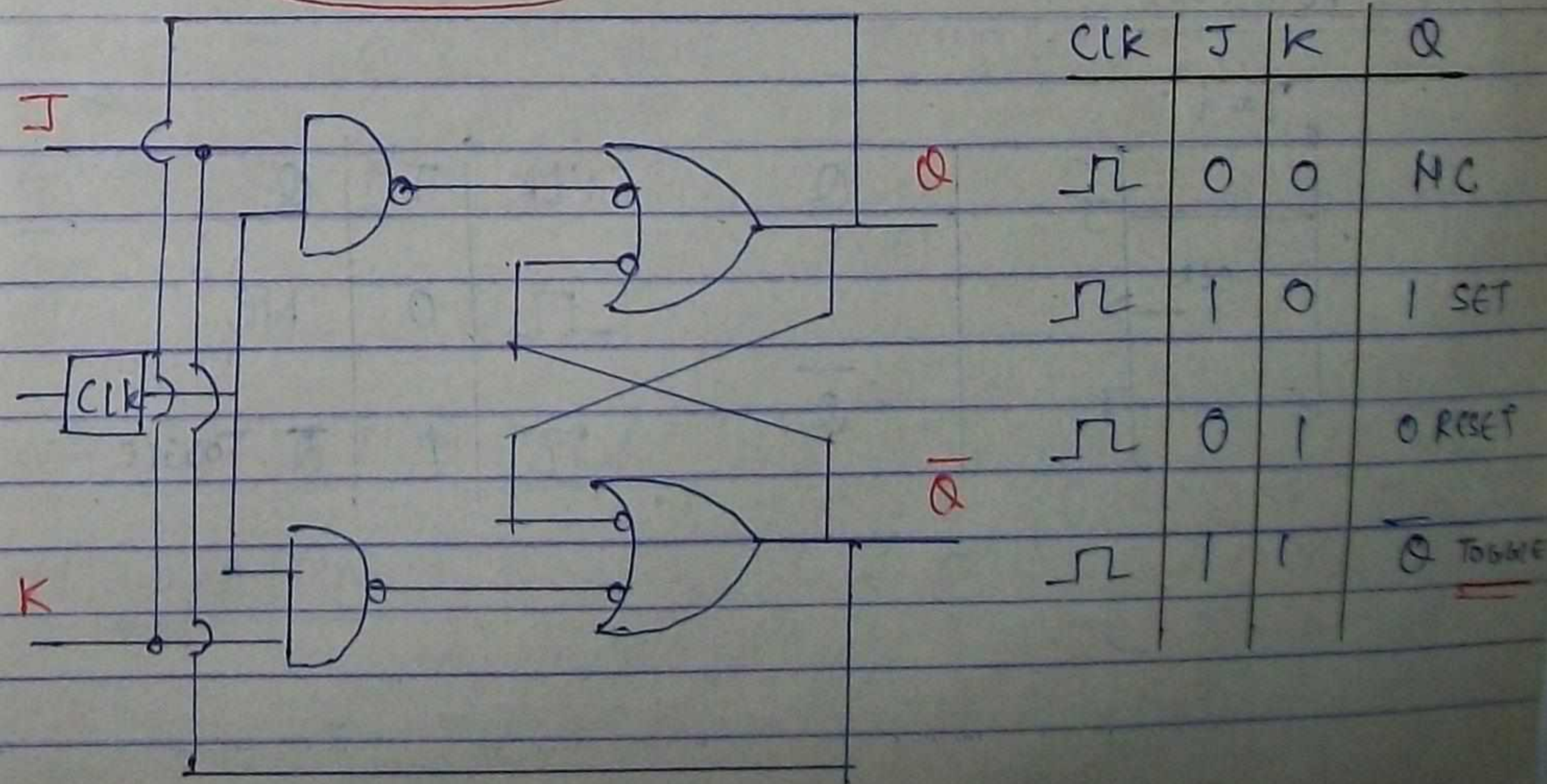
RACE CONDITION

B/w
 $Q = \bar{Q}$
 to become
 1



⇒ JK - FLIPFLOP (LATCH) -

ACTIVE HIGH



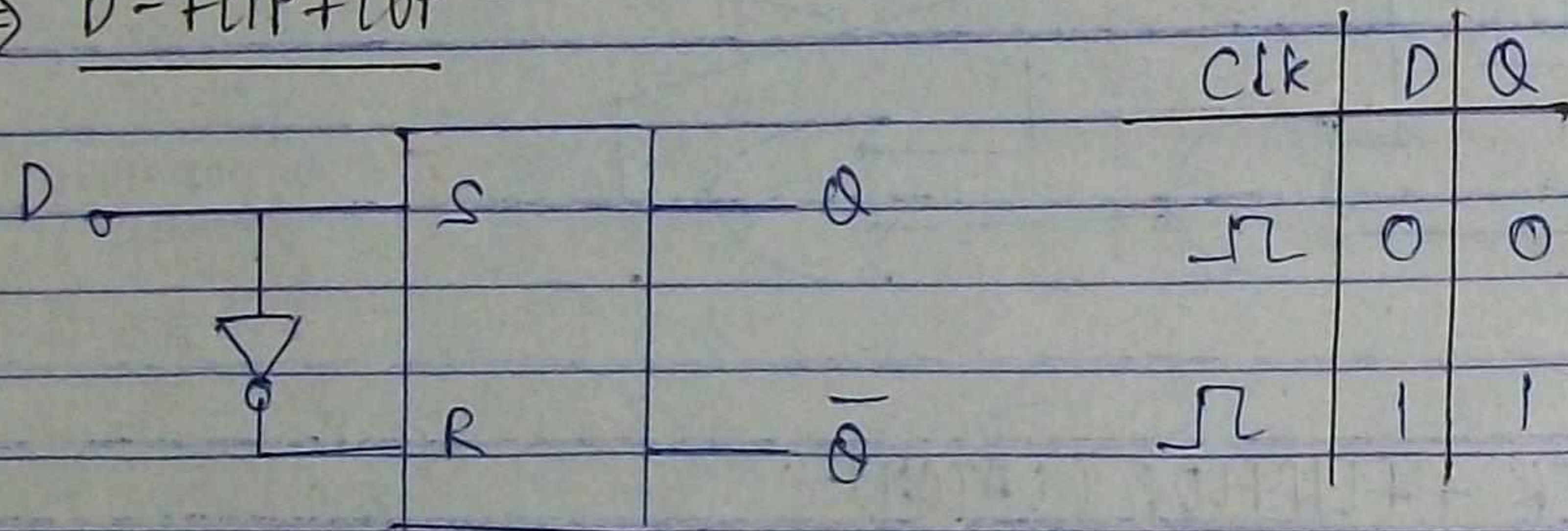
⇒ RACE AROUND CONDITION -

- Repeation of toggle is known as Race - around problem.

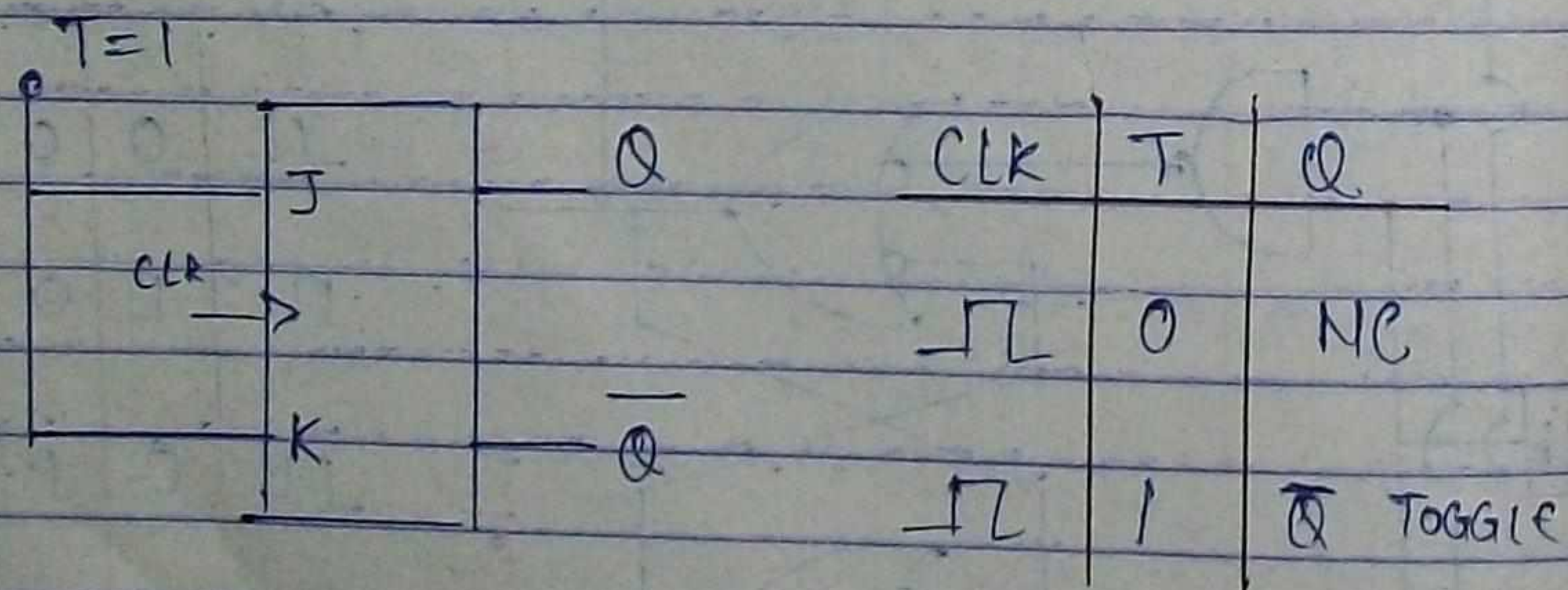
- To avoid Race around condition:

- use Edge-triggering.

⇒ D-FLIPFLOP -



⇒ T-FLIPFLOP -



Ex -

J	K	Q_n	Q_{n+1}
?	?	SET	RESET
0	1		
1	1	Ans.	X and 1

Ex

J	K	Q_n	Q_{n+1}
?	?	RESET	SET
0	1		
0	0	Ans	0 and X

Ex

S	R	Q_n	Q_{n+1}
?	?	SET	SET
1	0		
0	0	Ans.	X and 0

Ex

S	R	Q_n	Q_{n+1}
?	?	SET	RESET
0	1	Ans.	

⇒ TRANSITION TABLE -

① JK FLIPFLOP

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

② SR ff -

Q	Q'	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

③ D-ff -

Q	Q'	D
0	0	0
0	1	1
1	0	0
1	1	1

④ T-ff -

Q	Q'	T
0	0	0
0	1	1
1	0	1
1	1	0

Ques. The O/P of JK ff is 1, it changes to zero when CLK is applied then

J = ? K = ?
Soln: 0 1
 1 1 X and 1

Ques. If an SR latch has '1' on S i/p and '0' on R i/p and then S i/p $\rightarrow$ 0 then the latch will be

Soln: $\begin{matrix} S=1 \\ R=0 \end{matrix}$ SET $\rightarrow$ $\begin{matrix} S=0 \\ R=0 \end{matrix}$ NC ANS $\rightarrow$ SET

Ques. The Q=1 of JK ff and Q does not change when CLK is applied what are the i/p of J and K

Soln: $\begin{matrix} 0 & 0 \\ 1 & 0 \end{matrix}$ } ANS $\rightarrow$ X and 0

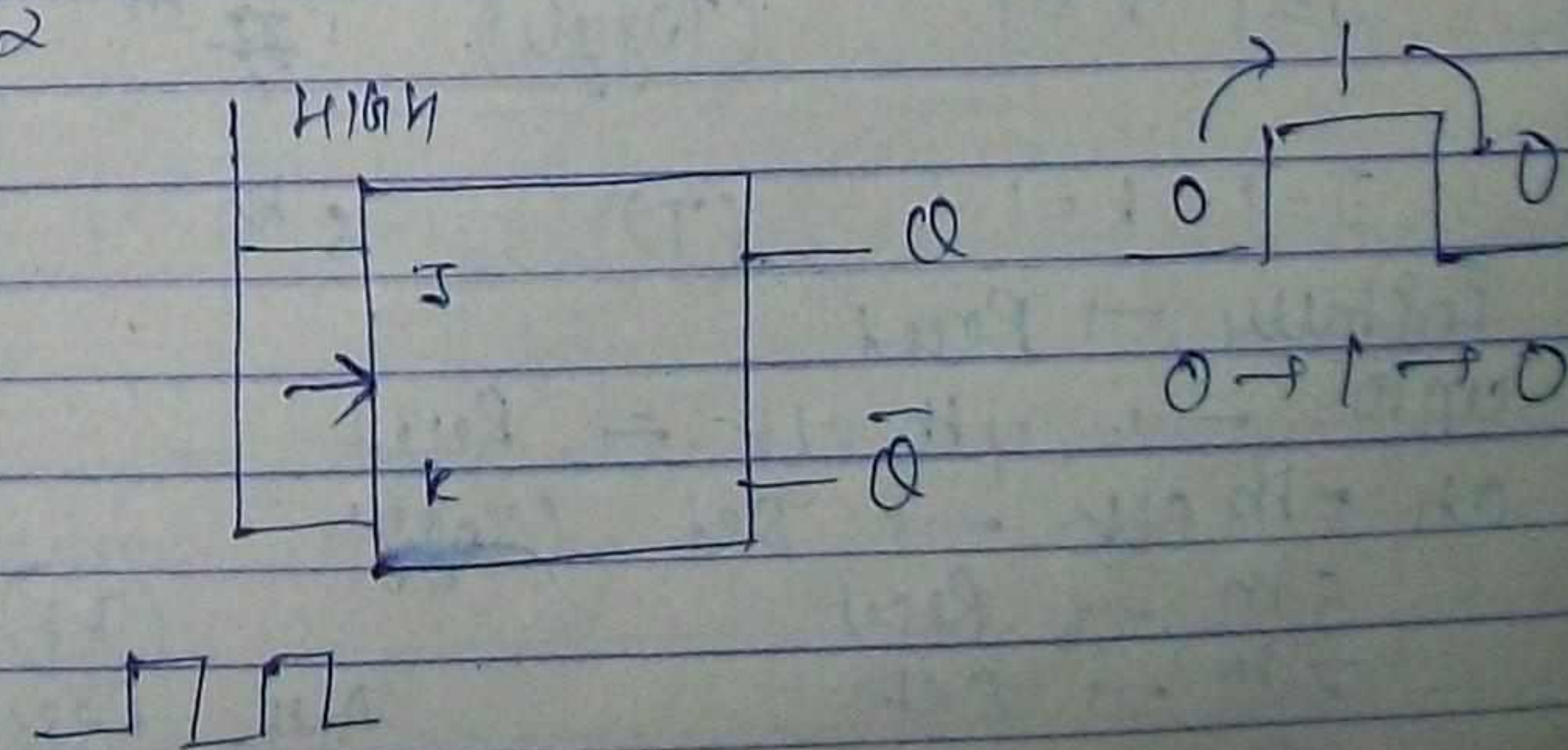
⑤ If 'x' Hz is the i/p frequency of a D-ff then O/P frequency will be $x/2$ Hz.

INPUT FREQ $\Rightarrow$ O/P FREQ
 x Hz $\frac{x}{2^n}$ Hz

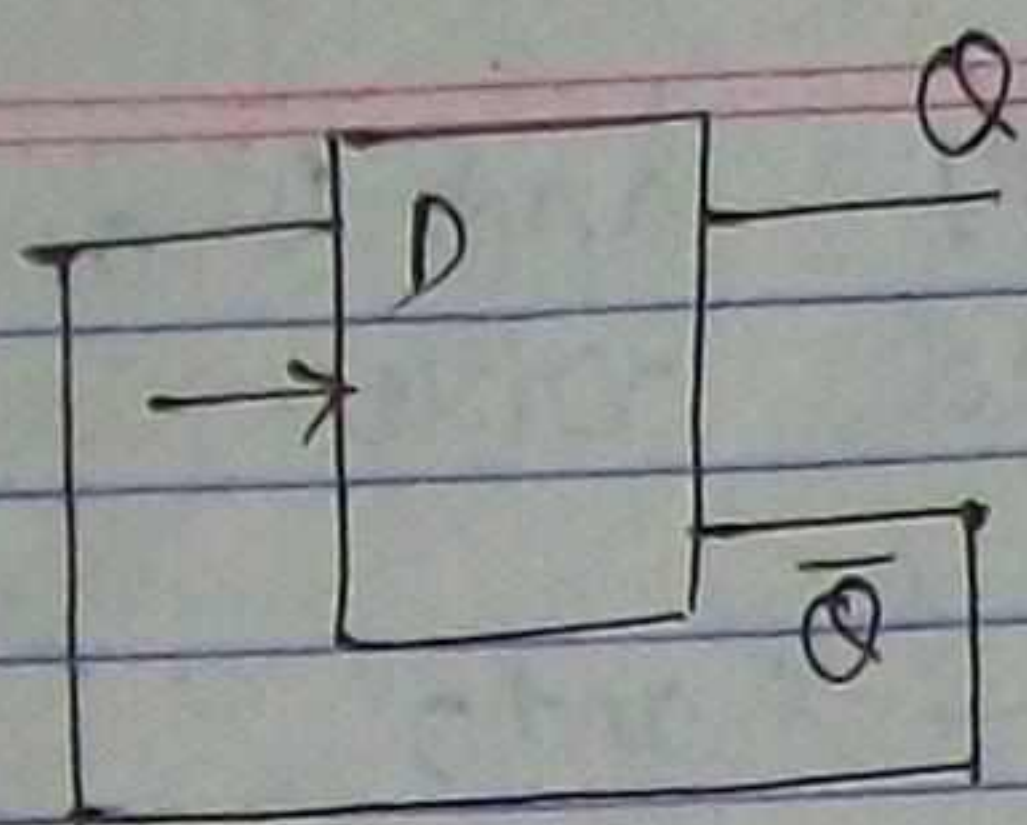
(n = no. of D flipflops.)

Ques. If J=1 and K=1 has a 10kHz CLK i/p then O/P freq?

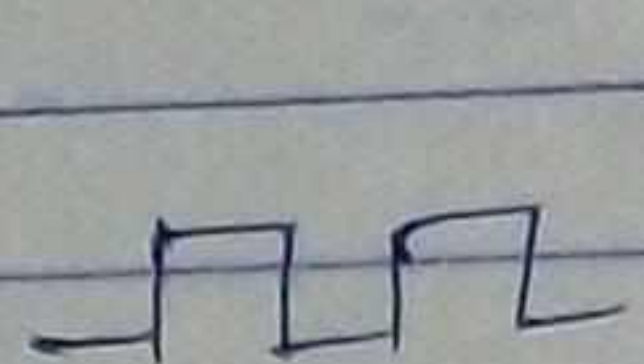
Soln: $\frac{10}{2} = 5\text{kHz}$



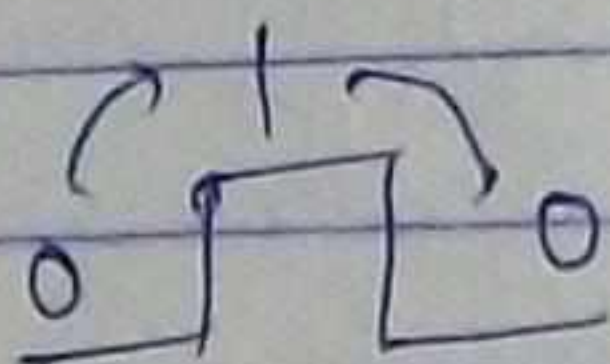
Ques.



- ① DIVIDE / 2
- ② DIVIDE / 1
- ③ DIVIDE / 4
- ④ NONE

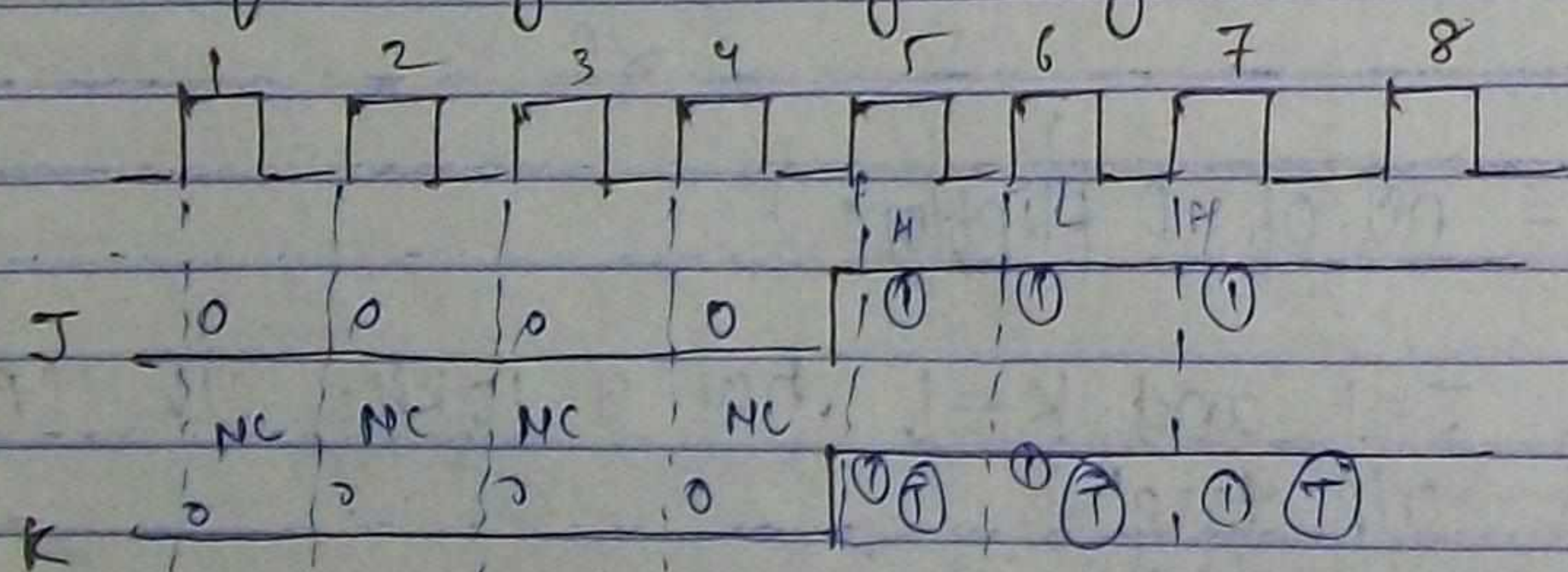


(I/P)



(O/P)

Ques. Initially J-K ff is reset, immediately after 7th clk what is the o/p for the following timing diagram



Solⁿ: J=1 K=1 (Toggle) 1st 4th

J=1 K=1 (T) 6th

Initially → Reset

up to → 4th clk → Reset

at 5th clk → Set (Toggle)

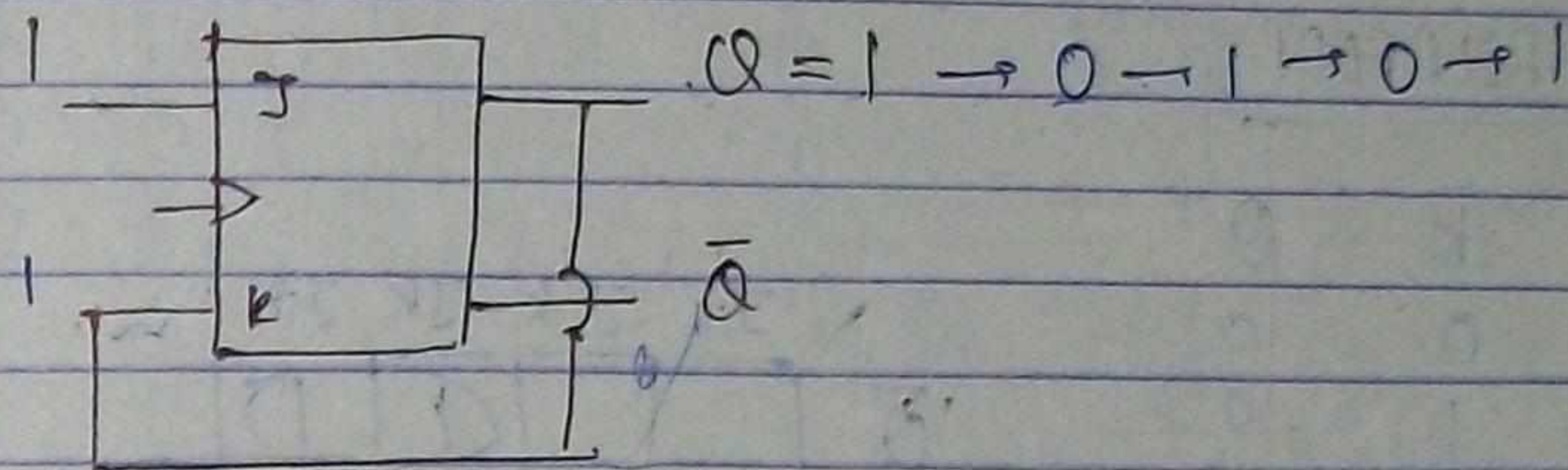
6th → Reset

7th → Set

8th → Reset

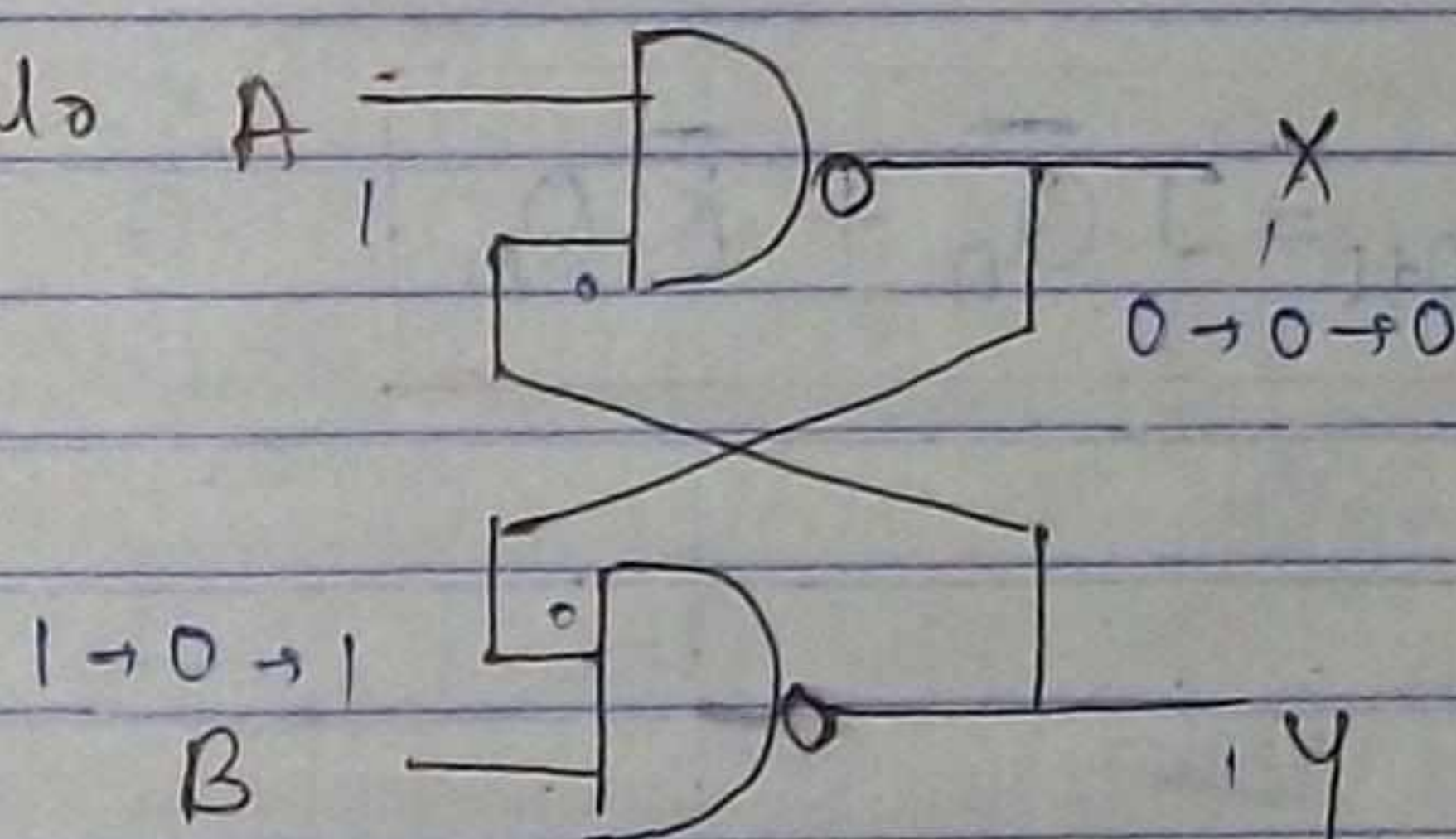
Ans. Reset-8

Ques. Initially Q=1 what are subsequent state at Q



Solⁿ: 01010101

Ques. A and B are inputs to two 2-input AND gates. Initially A=B=1 later B=101010... then X and Y=?

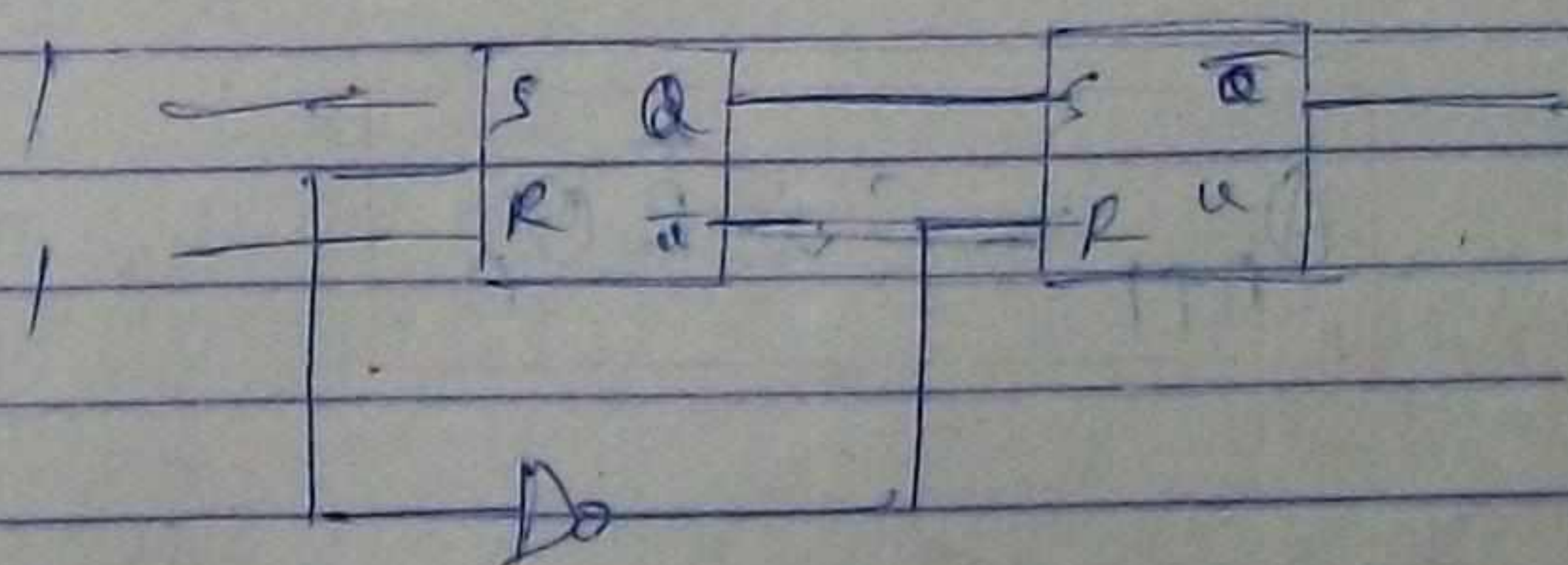


Solⁿ:

ACTIVE LOW

SR LATCH

∴ fixed at '0' and '1' repeatedly.



→ CHARACTERISTIC EQUATION OF FLIP FLOP -

① J-K FLIP FLOP -

Q	J	K	Q ⁺
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

	Jk'	J'k	JK	JK'
Q̄			1	1
Q	1			1

$$Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$$

② SR - FLIP FLOP -

Q	S	R	Q ⁺
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	X
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	X

	S'R'	S'R	SR	SR'
Q̄			X	1
Q	1		X	1

$$Q_{n+1} = S + \bar{R}Q_n$$

③ D - FLIP FLOP -

Q	D	Q ⁺
0	0	0
0	1	1
1	0	0
1	1	1

	D̄	D
Q̄		1
Q		1

$$Q_{n+1} = D$$

④ T - FLIP FLOP -

Q	T	Q ⁺
0	0	0
0	1	1
1	0	1
1	1	0

	T̄	T
Q̄	0	1
Q	1	0

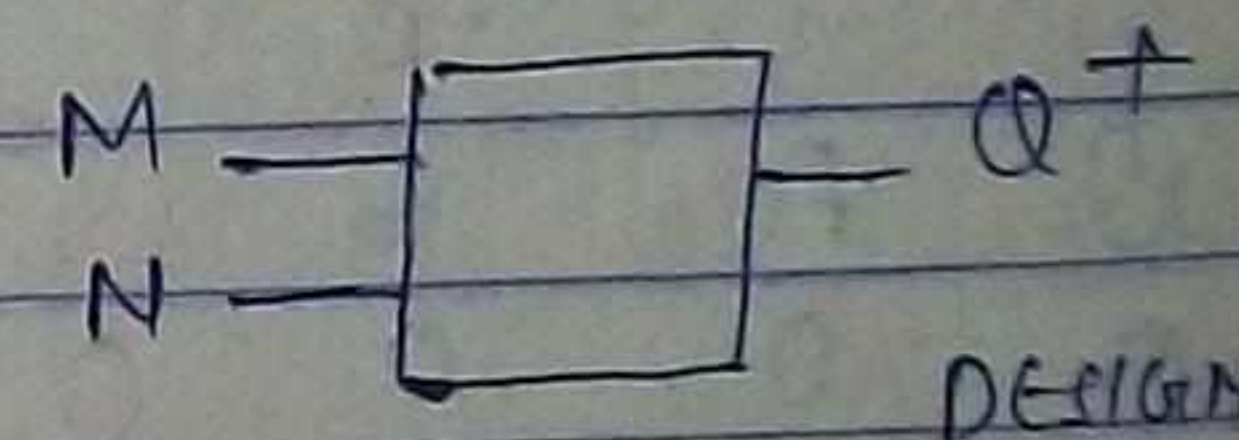
$$Q_{n+1} = T \oplus Q_n$$

$$Q_{n+1} = \bar{T}Q_n + T\bar{Q}_n$$

CIRCUIT DESIGNING - (CONVERSION OF FLIP FLOPS)

Ex =

Q	M	N	Q ⁺	J	K
0	0	0	0	0	X
0	0	1	1	1	X
0	1	0	1	1	X
0	1	1	0	0	X
1	0	0	1	X	0
1	0	1	1	X	0
1	1	0	1	X	0
1	1	1	0	X	0



Since

Q	Q ⁺	J	K
0	0	0	X
0	1	1	X
1	0	X	0
1	1	X	0

for J

for K

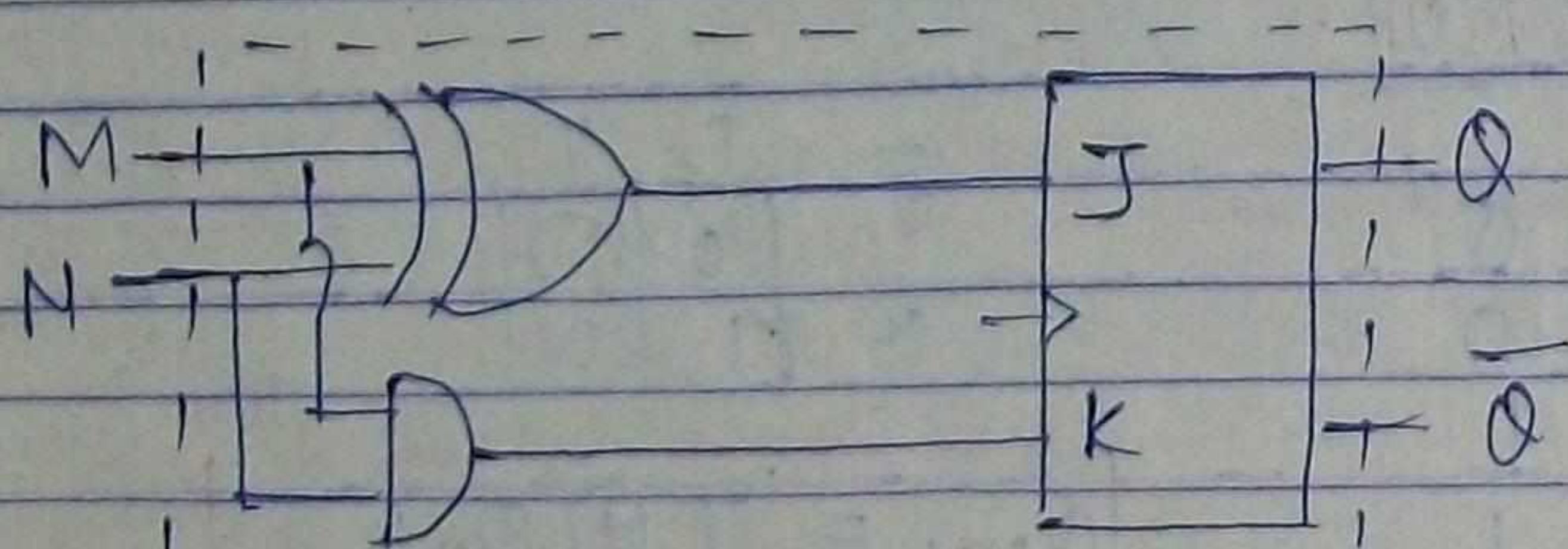
$$\begin{array}{c}
 \overline{M}\overline{N} \quad M\overline{N} \quad MN\overline{N} \\
 \begin{array}{|c|c|c|c|}
 \hline
 Q & 1 & 1 & 1 \\
 \hline
 \overline{Q} & X & X & X \\
 \hline
 \end{array}
 \end{array}$$

$$\begin{array}{c}
 \overline{M}\overline{N} \quad M\overline{N} \quad MN\overline{N} \\
 \begin{array}{|c|c|c|c|}
 \hline
 Q & X & X & X \\
 \hline
 \overline{Q} & & & 1 \\
 \hline
 \end{array}
 \end{array}$$

$$J = \overline{M}\overline{N} + MN$$

$$K = MN$$

$$J = M \oplus N$$



⇒ CONVERSION OF JK-FF INTO SR-FF —

Ex FOR DON'T CARE CONDITIONS —

$$\begin{array}{c}
 \begin{array}{c} S \\ R \end{array} \\
 \begin{array}{|c|c|c|c|c|c|}
 \hline
 Q & A & B & Q^+ & J & K \\
 \hline
 0 & 0 & 0 & 0 & 0 & X \\
 0 & 0 & 1 & 0 & 0 & X \\
 0 & 1 & 0 & 1 & 1 & X \\
 0 & 1 & 1 & X & X & X \\
 1 & 0 & 0 & 1 & X & 0 \\
 1 & 0 & 1 & 0 & X & 1 \\
 1 & 1 & 0 & 1 & X & 0 \\
 1 & 1 & 1 & X & X & X \\
 \hline
 \end{array}
 \end{array}$$

$$\begin{array}{c}
 Q^+ JK \\
 \begin{array}{|c|c|c|c|}
 \hline
 0 & 0 & 0 & X \\
 0 & 0 & 1 & X \\
 0 & 1 & 1 & X \\
 1 & 0 & 1 & 1 \\
 1 & 1 & 1 & 0 \\
 \hline
 \end{array}
 \end{array}$$

12/10/11

$$\begin{array}{c}
 \overline{A}\overline{B} \quad \overline{A}B \quad AB \quad A\overline{B} \\
 \begin{array}{|c|c|c|c|}
 \hline
 Q^+ & & X & 1 \\
 \hline
 Q & X & X & X \\
 \hline
 \end{array}
 \end{array}$$

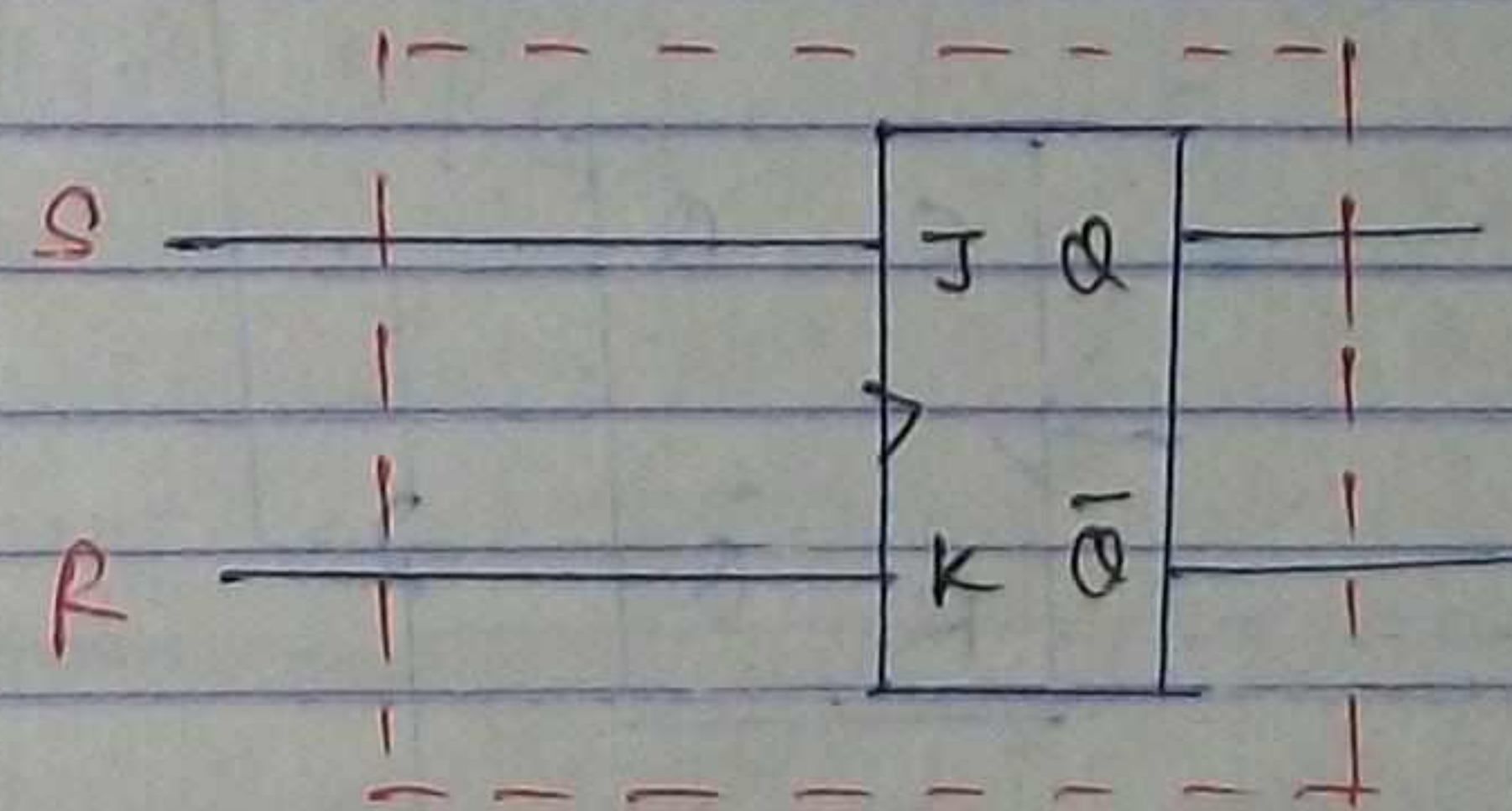
$$\begin{array}{c}
 \overline{A}\overline{B} \quad \overline{A}B \quad AB \quad A\overline{B} \\
 \begin{array}{|c|c|c|c|}
 \hline
 Q^+ & X & X & X \\
 \hline
 Q & & 1 & X \\
 \hline
 \end{array}
 \end{array}$$

$$J = A$$

$$K = B$$

$$\Rightarrow J = S$$

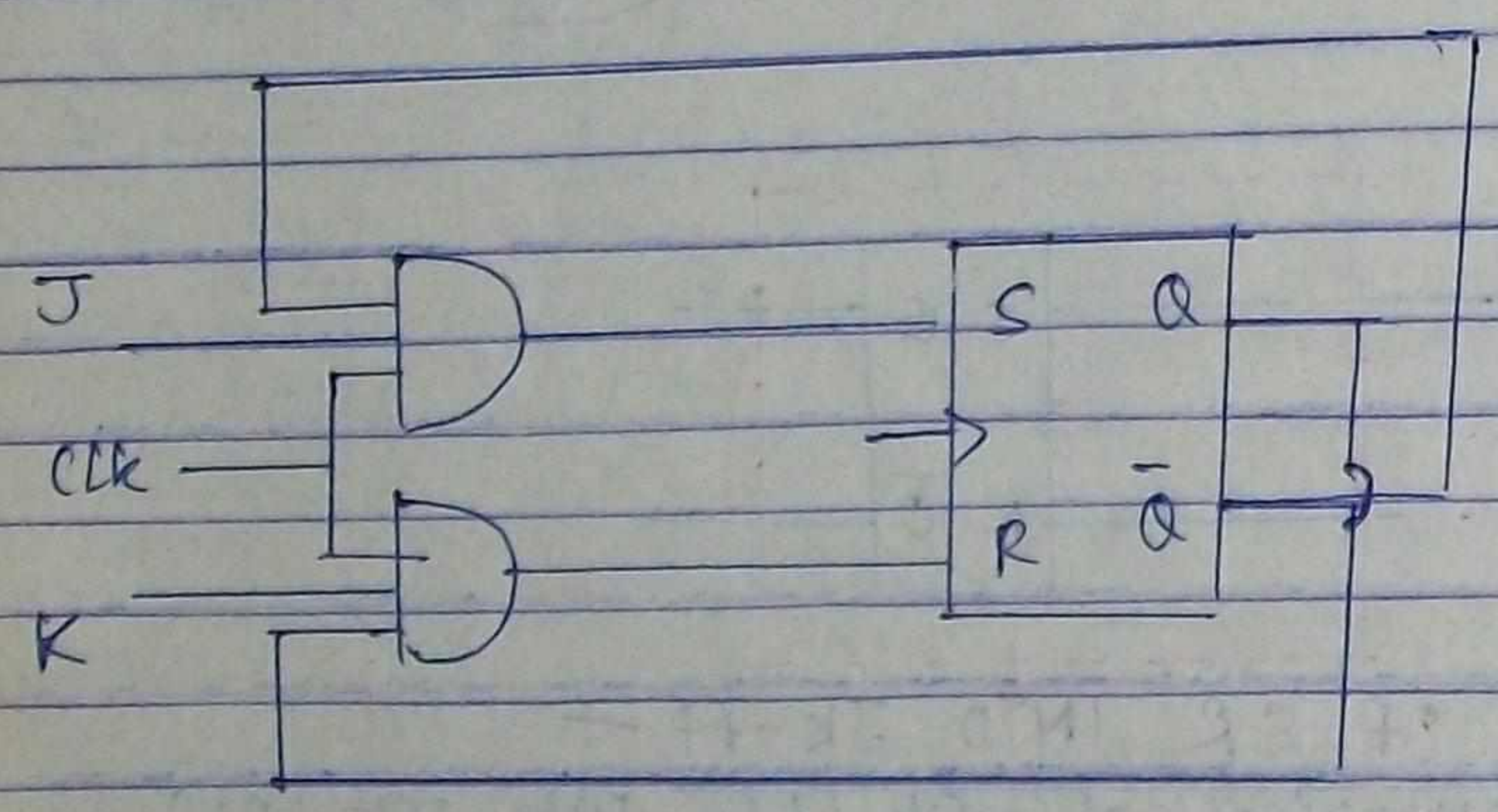
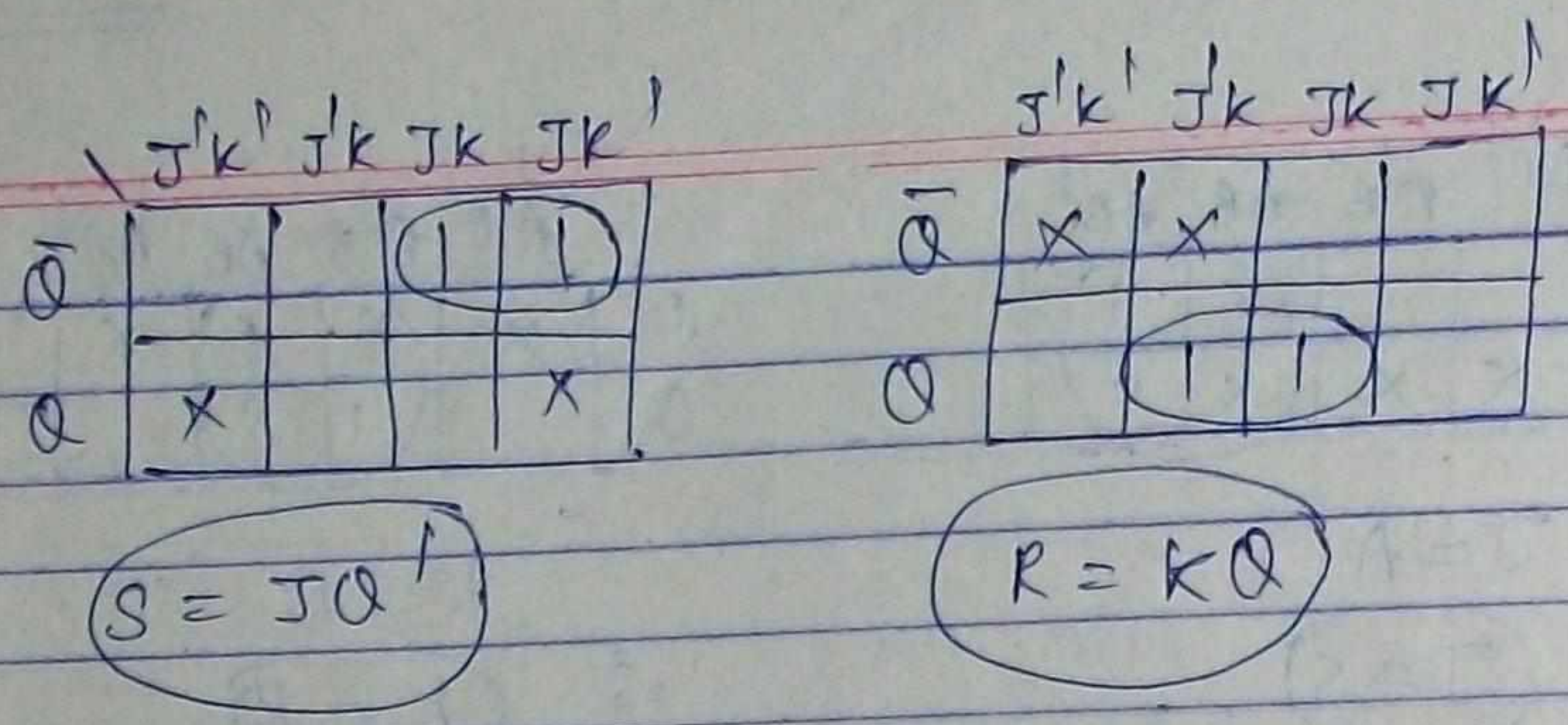
$$\Rightarrow K = R$$



⇒ CONVERSION OF SR INTO JK-FF —

Ques. In the conversion of SR-FF into JK-FF what are the I/P for S and R.

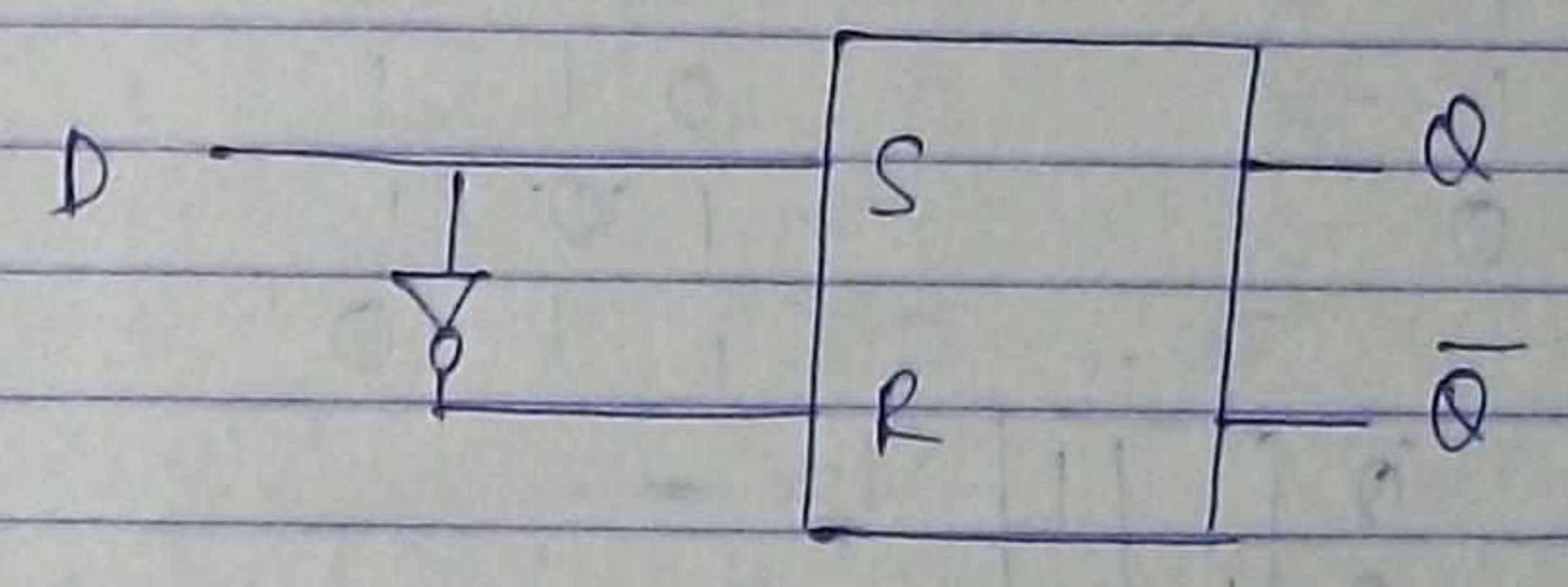
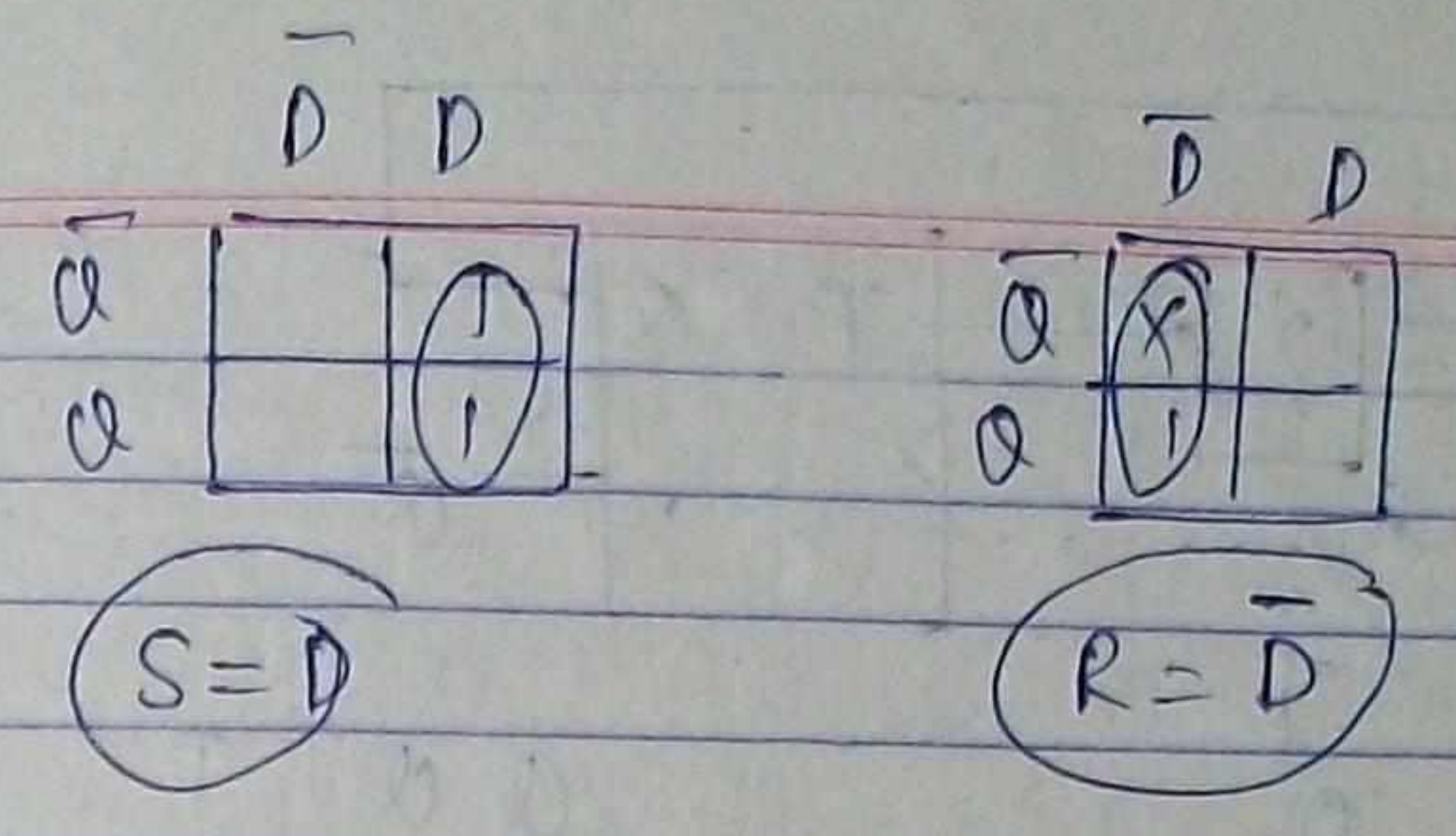
$$\begin{array}{c}
 \begin{array}{c} J \\ K \end{array} \\
 \begin{array}{|c|c|c|c|c|c|}
 \hline
 Q & A & B & Q^+ & S & R \\
 \hline
 0 & 0 & 0 & 0 & 0 & X \\
 0 & 0 & 1 & 0 & 0 & X \\
 0 & 1 & 0 & 1 & 1 & 0 \\
 0 & 1 & 1 & 1 & 1 & 0 \\
 1 & 0 & 0 & 1 & X & 0 \\
 1 & 0 & 1 & 0 & 0 & 1 \\
 1 & 1 & 0 & 1 & X & 0 \\
 1 & 1 & 1 & 0 & 0 & 1 \\
 \hline
 \end{array}
 \end{array}$$



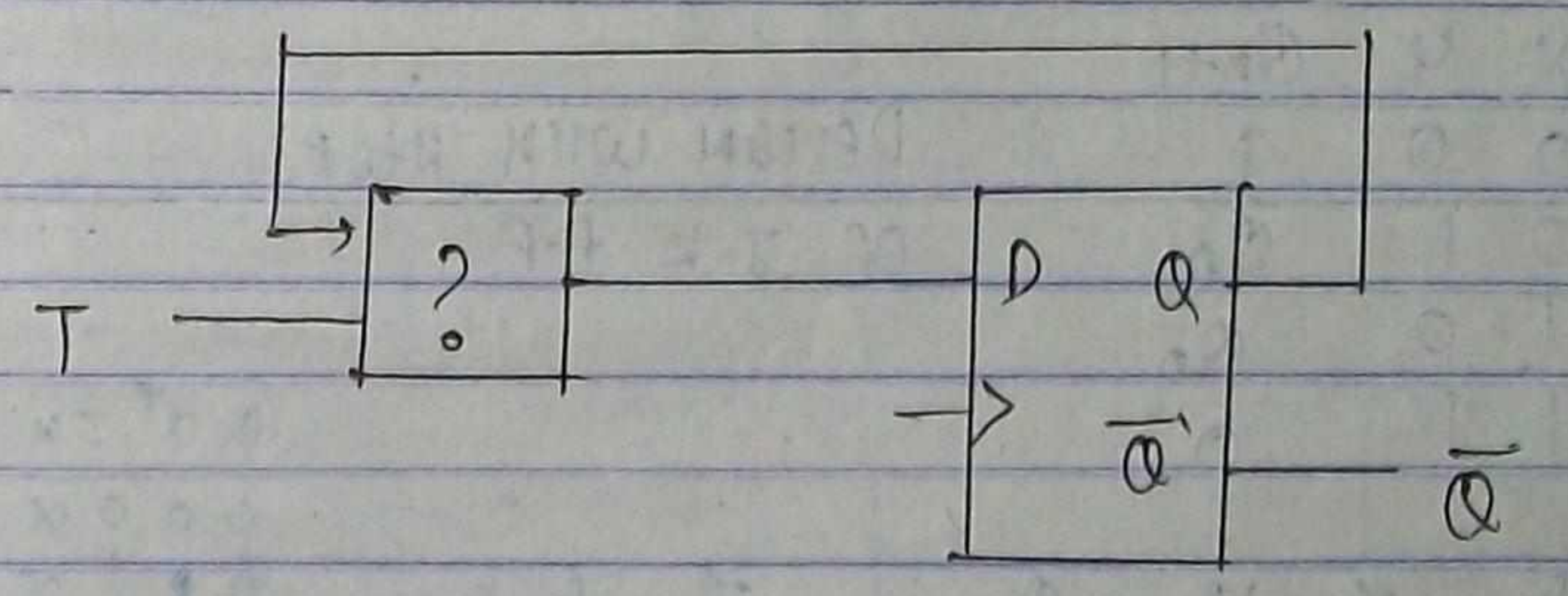
⇒ CONVERSION OF SR INTO D-FF -
 Ques. In conversion of SR FF INTO DFF $S=? R=?$

Q	D	Q^+	S	R
0	0	0	0	x
0	1	1	1	0
1	0	0	0	1
1	1	1	x	0

Q	Q^+	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0



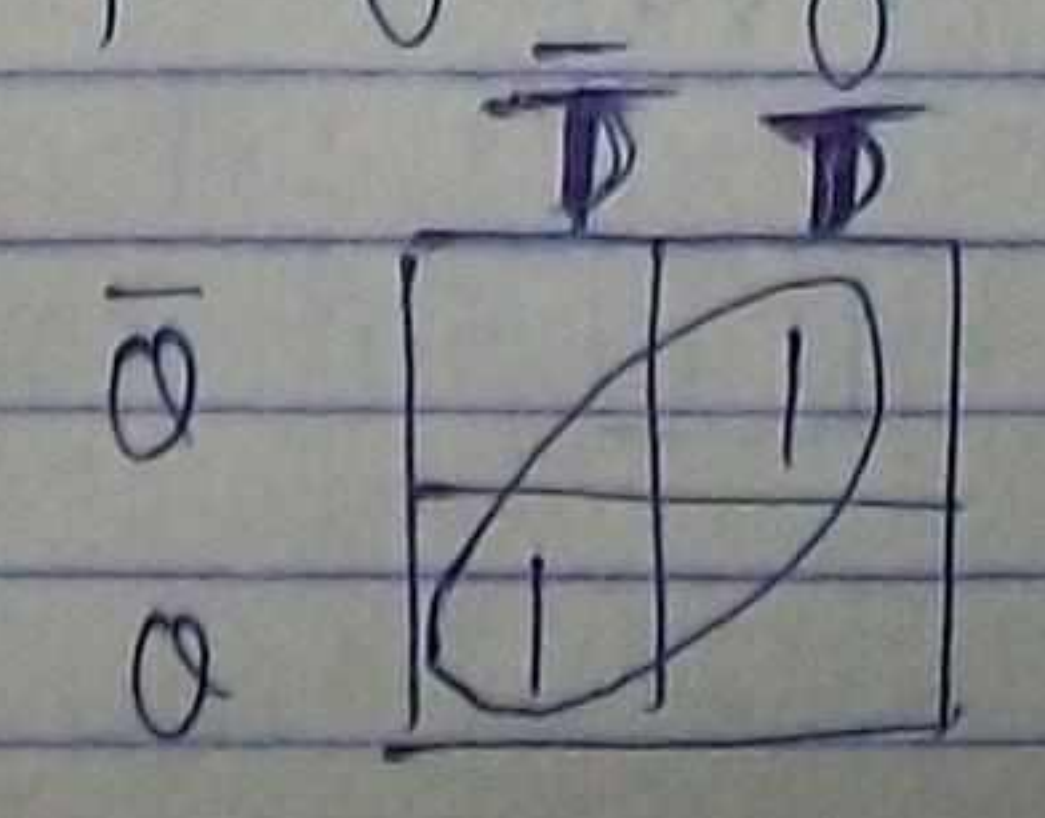
Ques.



Soln.

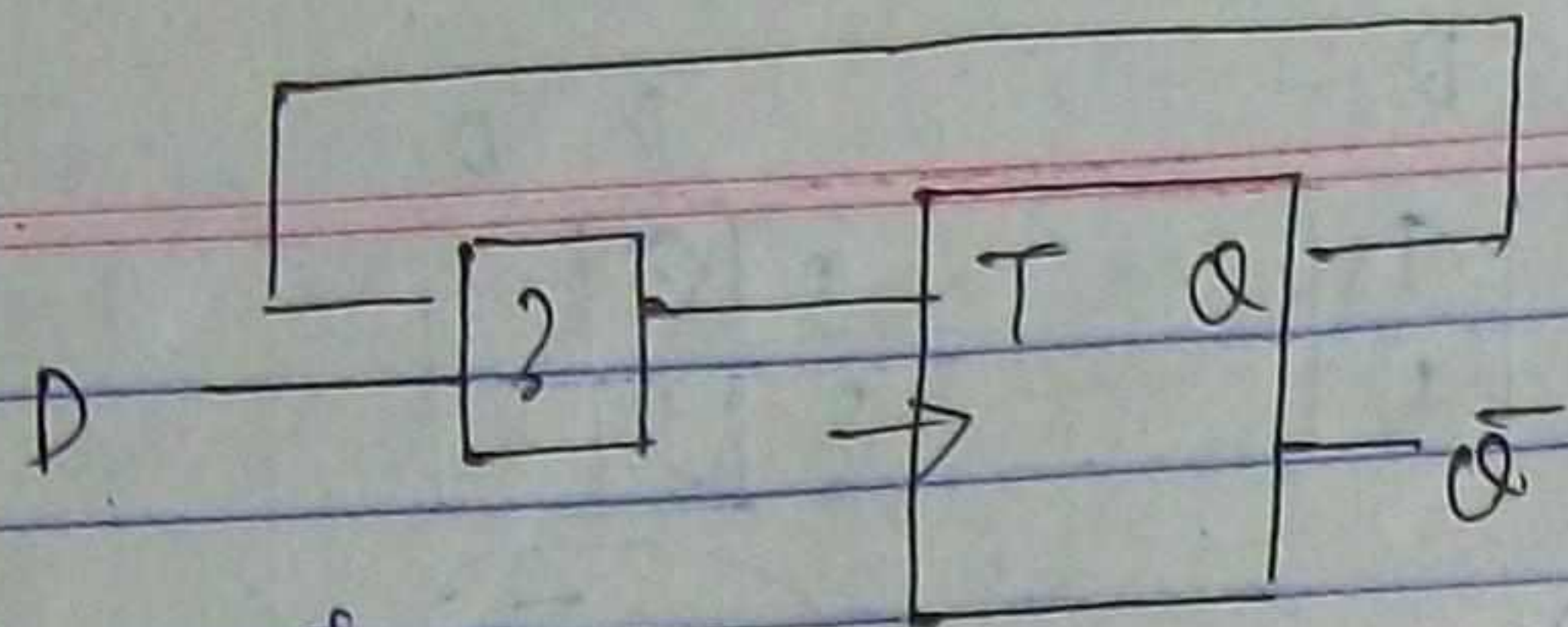
Q	T	Q^+	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1



$D = T\bar{Q} + \bar{T}Q$
 $D = T \oplus Q$

Ques



Solⁿ:

Q	D	Q ⁺	T
0	0	0	0
0	1	1	1
1	0	0	1
1	1	1	0

Q	Q ⁺	T
0	0	0
0	1	1
1	0	1
1	1	0

$$T = D \oplus Q$$

T	J
0	1
1	1

Ques

X	Y	Q _{n+1}
0	0	1
0	1	Q _n
1	0	Q _n
1	1	0

DESIGN WITH HELP
OF J-K FF

Solⁿ:

Q	X	Y	Q _{n+1}	J	K
0	0	0	1	1	X
0	0	1	1	1	X
1	1	0	1	X	0
1	1	1	0	X	1

Q	Q ⁺	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Q	X	Y	Q _{n+1}	J	K
0	0	0	1	1	X
0	0	1	1	1	X
1	1	0	1	X	0
1	1	1	0	X	1

wrong

Solⁿ:

Q	X	Y	Q _{n+1}	J	K
0	0	0	1	1	X
0	0	1	1	1	X
0	1	0	0	0	X
0	1	1	0	0	X
1	0	0	1	X	0
1	0	1	0	X	1
1	1	0	1	X	0
1	1	1	0	X	1

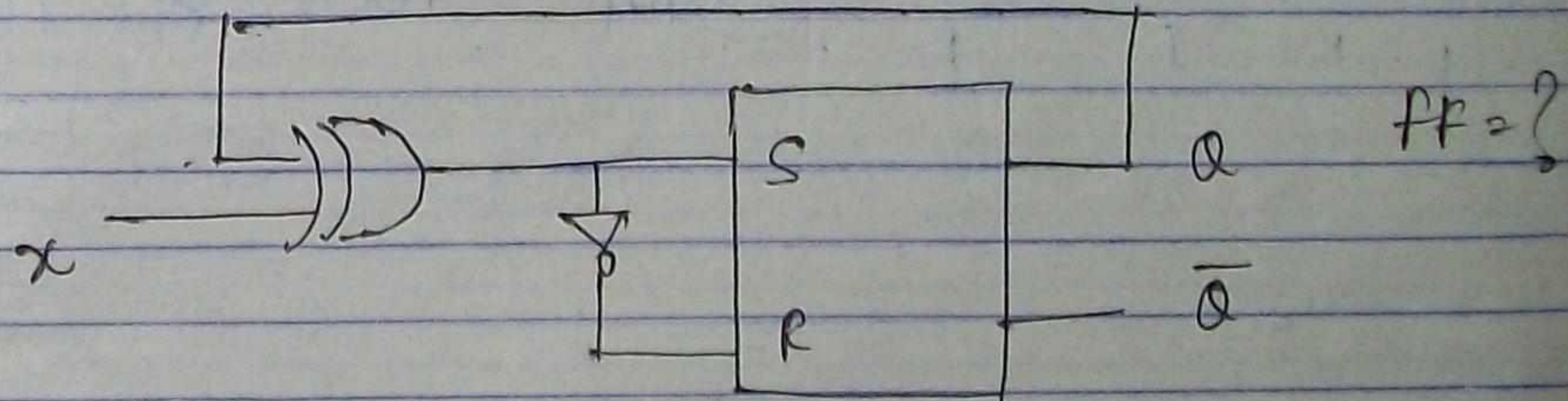
Q	X	Y	Q _{n+1}	J	K
0	0	0	1	1	X
0	0	1	1	1	X
0	1	0	0	0	X
0	1	1	0	0	X
1	0	0	1	X	0
1	0	1	0	X	1
1	1	0	1	X	0
1	1	1	0	X	1

Q	X	Y	Q _{n+1}	J	K
0	0	0	1	1	X
0	0	1	1	1	X
0	1	0	0	0	X
0	1	1	0	0	X
1	0	0	1	X	0
1	0	1	0	X	1
1	1	0	1	X	0
1	1	1	0	X	1

$$J = \bar{X}$$

$$K = Y$$

Ques



ANS.

T-FLIPFLOP

$$T = X \oplus Q$$

In the conversion of D-ff into T-ff (or) T-ff into D-ff, the gate required is EX-OR gate.

Ques. A new ff is having behaviour as described below, it has 2 i/p X AND Y when both $X=Y=1$ ff $\rightarrow$ SET else ff $\rightarrow$ RESET. If both i/p are different and are $X=0$ $Y=1$ ff complement itself otherwise it is going to retain the last state, then expression?

Solⁿ:

Q	X	Y	Q ⁺
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

	XY	X ¹ Y	XY ¹	X ¹ Y ¹
$\bar{Q}$	0	1	1	0
Q	0	0	1	1

$$Q_{n+1} = Y\bar{Q} + XQ$$

$\Rightarrow$ FLIPFLOP CONVERSION -

① BY SR FLIPFLOP -

① SR-ff TO JK-ff -

$$S = JQ^1 \quad R = K \cdot Q$$

② SR TO D -

$$S = D \quad R = \bar{D}$$

③ SR TO T -

$$S = TQ^1 \quad R = TQ$$

② BY JK FLIPFLOP -

① JK TO SR -

$$J = S \quad K = R$$

② JK TO D -

$$J = D \quad K = D^1$$

③ JK TO T $\rightarrow$

$$J = T \quad K = T$$

③ FROM D FLIPFLOP -

① D TO JK -

$$D = JQ' + K'Q$$

② D TO SR -

$$D = S + R'Q$$

③ D TO T -

$$D = TQ' + T'Q = T \oplus Q$$

④ FROM T FLIPFLOP -

① T TO JK -

$$T = JQ' + KQ$$

② T TO SR -

$$T = SQ' + RQ$$

③ T TO D -

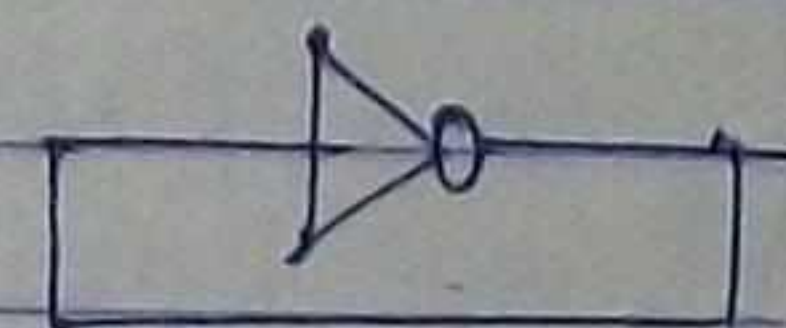
$$T = D \oplus Q \\ = D'Q + DQ'$$

When odd no of inverters are connected in a feedback it acts like a , astable multivibrator with timeperiod

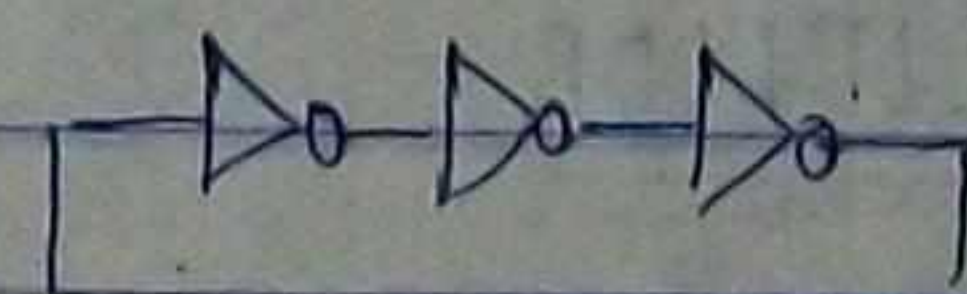
$$T = n \cdot 2T_p$$

$$f = \frac{1}{n \cdot 2T_p}$$

where, n = no. of Inverter
 T_p = prop. delay time.
 f = o/p frequency.



$$T = 2T_p$$



$$T = 3 \cdot 2T_p = 6T_p$$

NOTE -

- Repetation of toggle in the Race around problem it can be avoided by using:

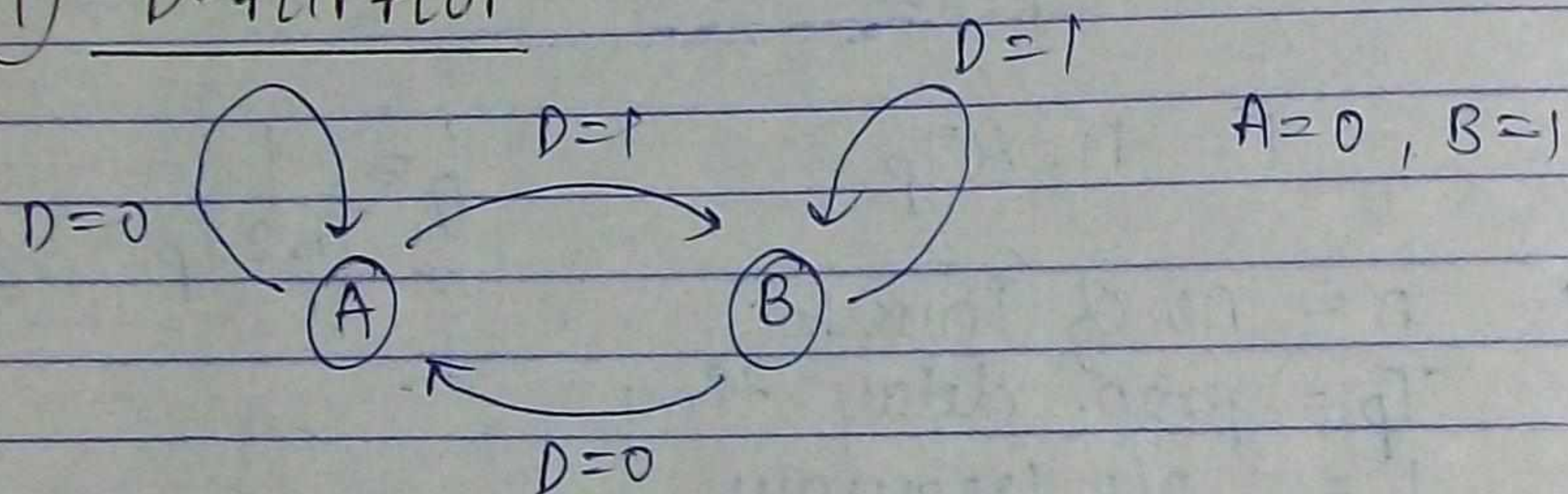
① Decreasing the pulse width.

② By using the edge triggering (with the help of integrator and differentiator circuit).

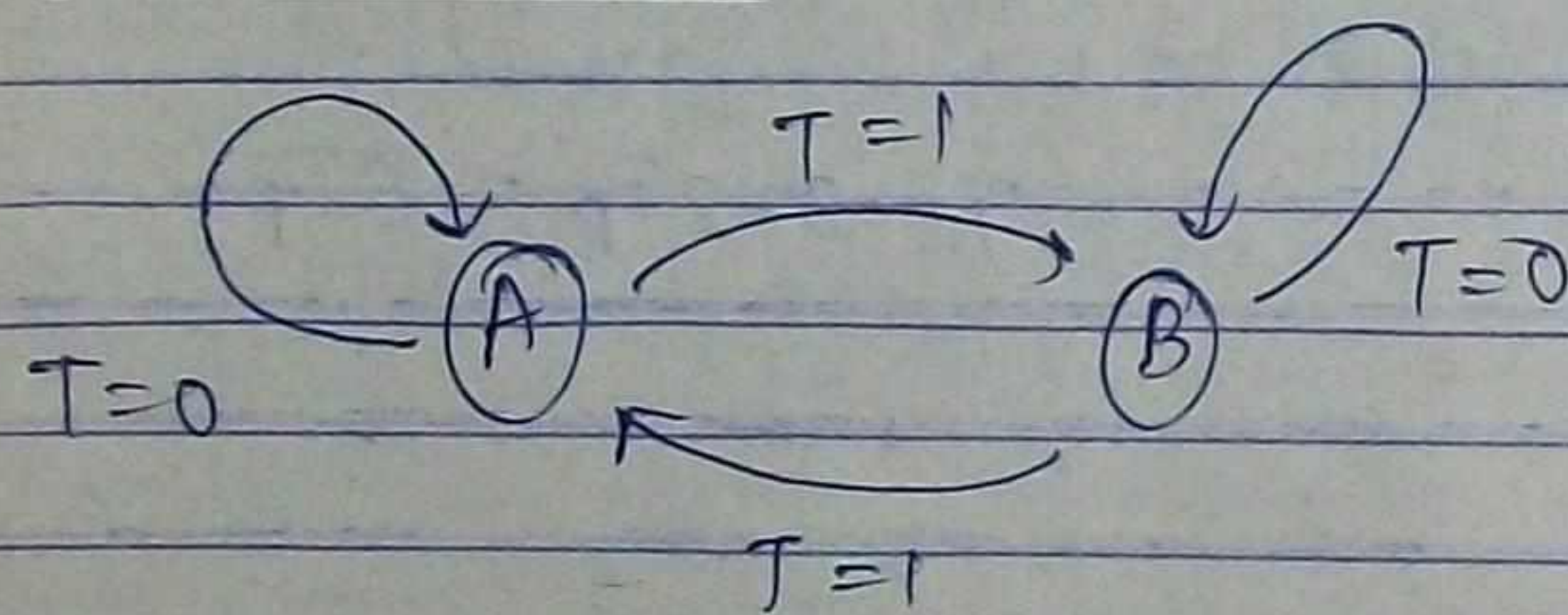
③ By increasing the propagation delay (By using master slave JK-FFs).

⇒ STATE DIAGRAM -

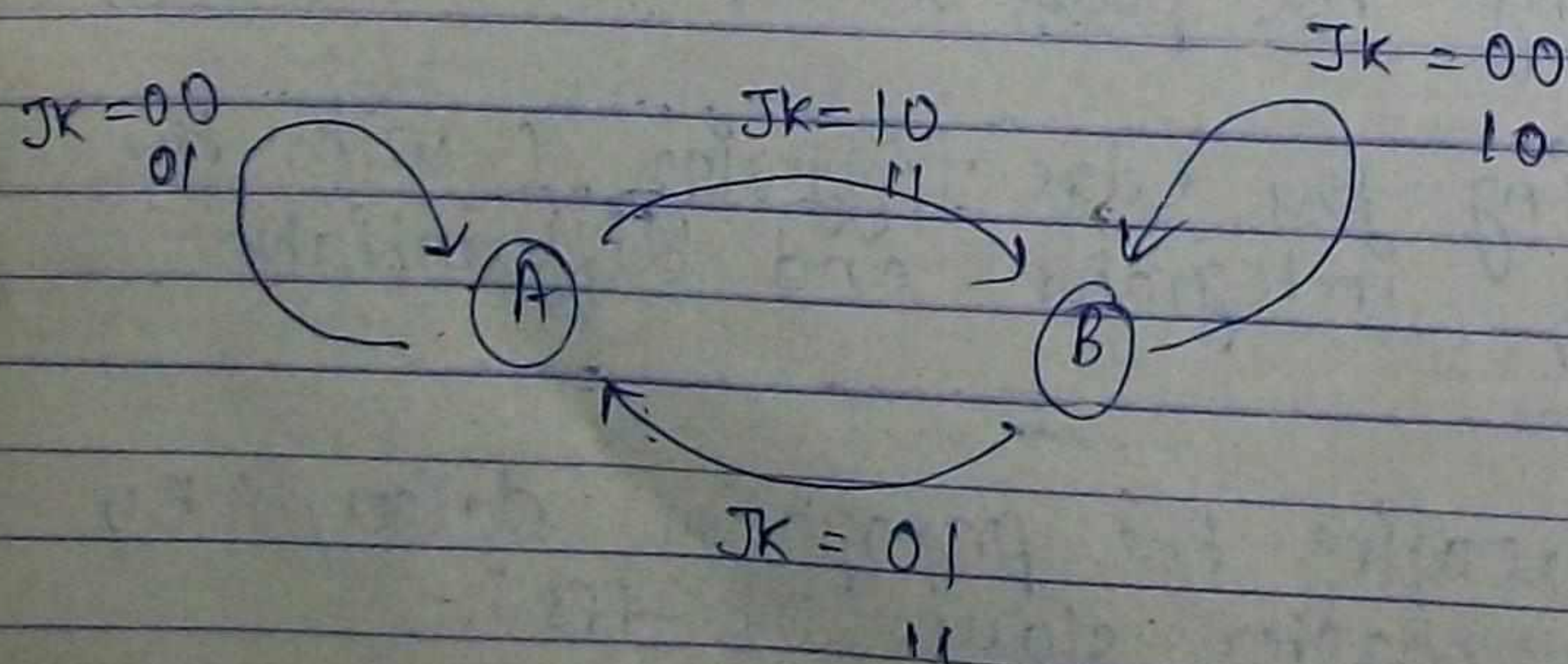
① D-FLIPFLOP -



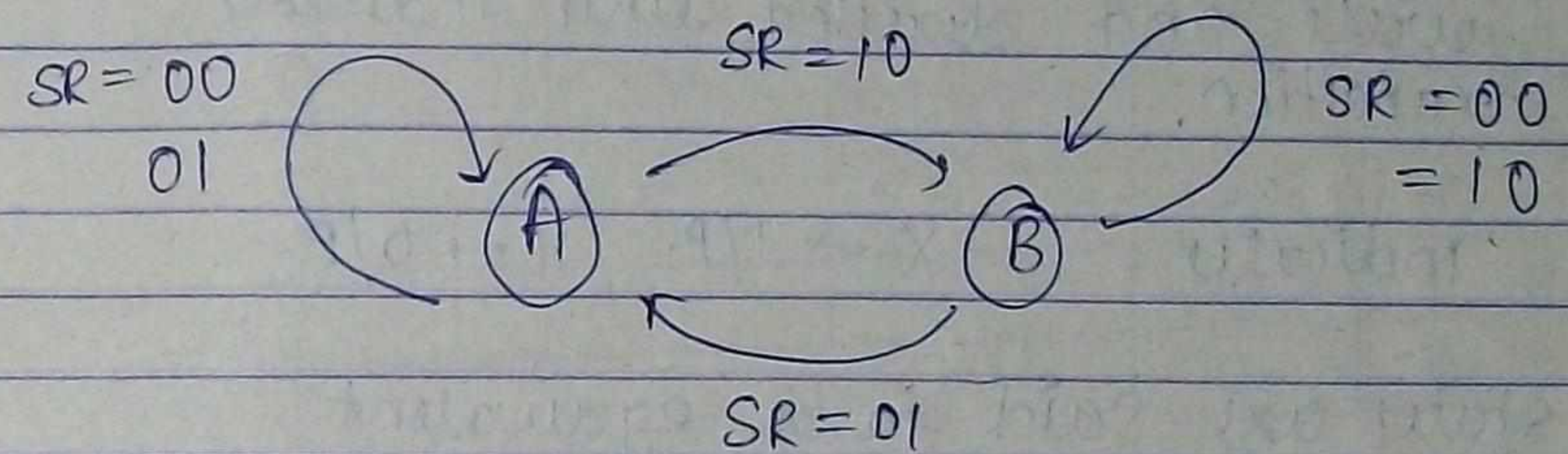
② T-FLIPFLOP -



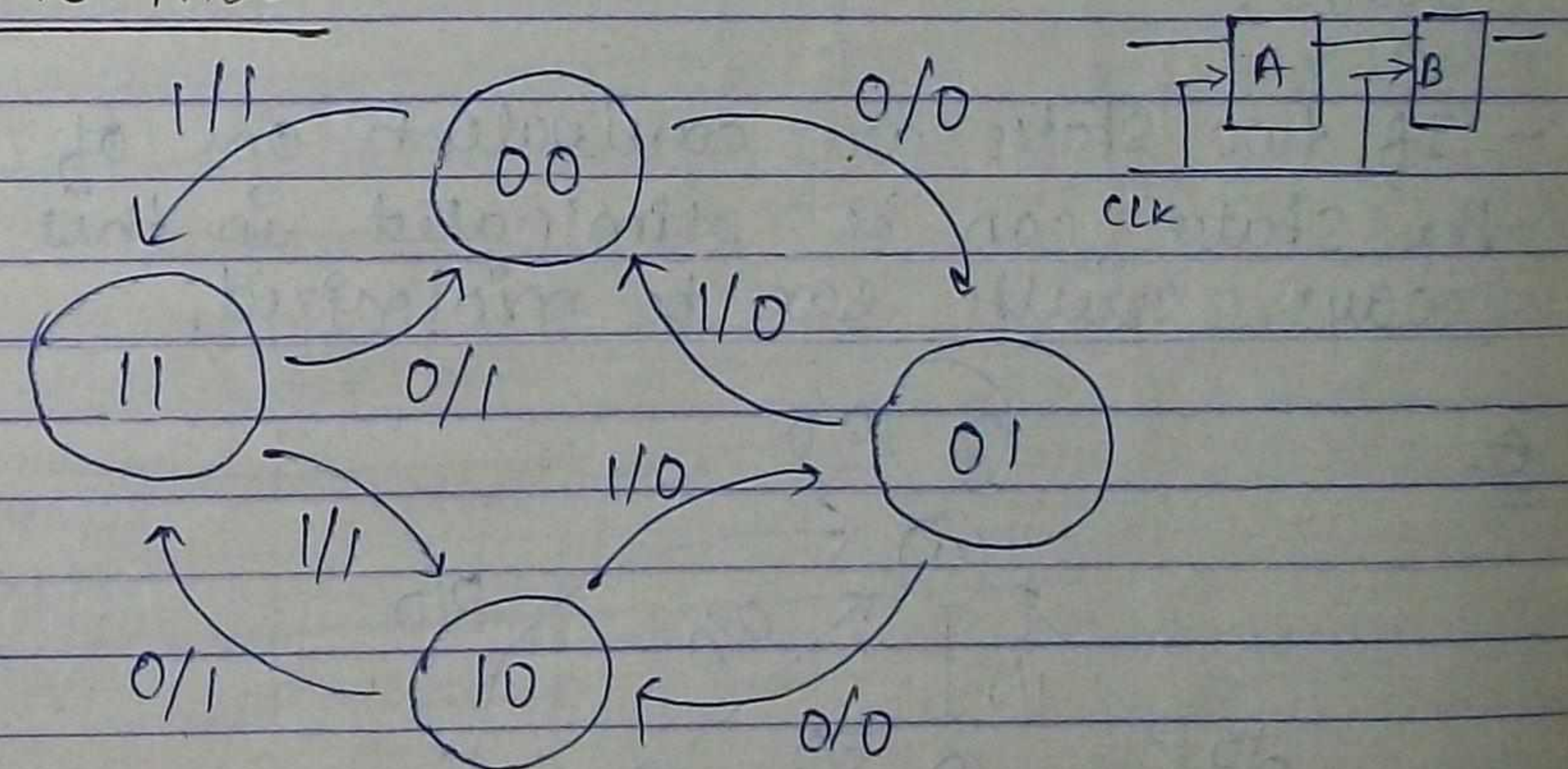
③ J.K - FLIPFLOP -



④ SR FLIPFLOP -



⇒ STATE TABLE -



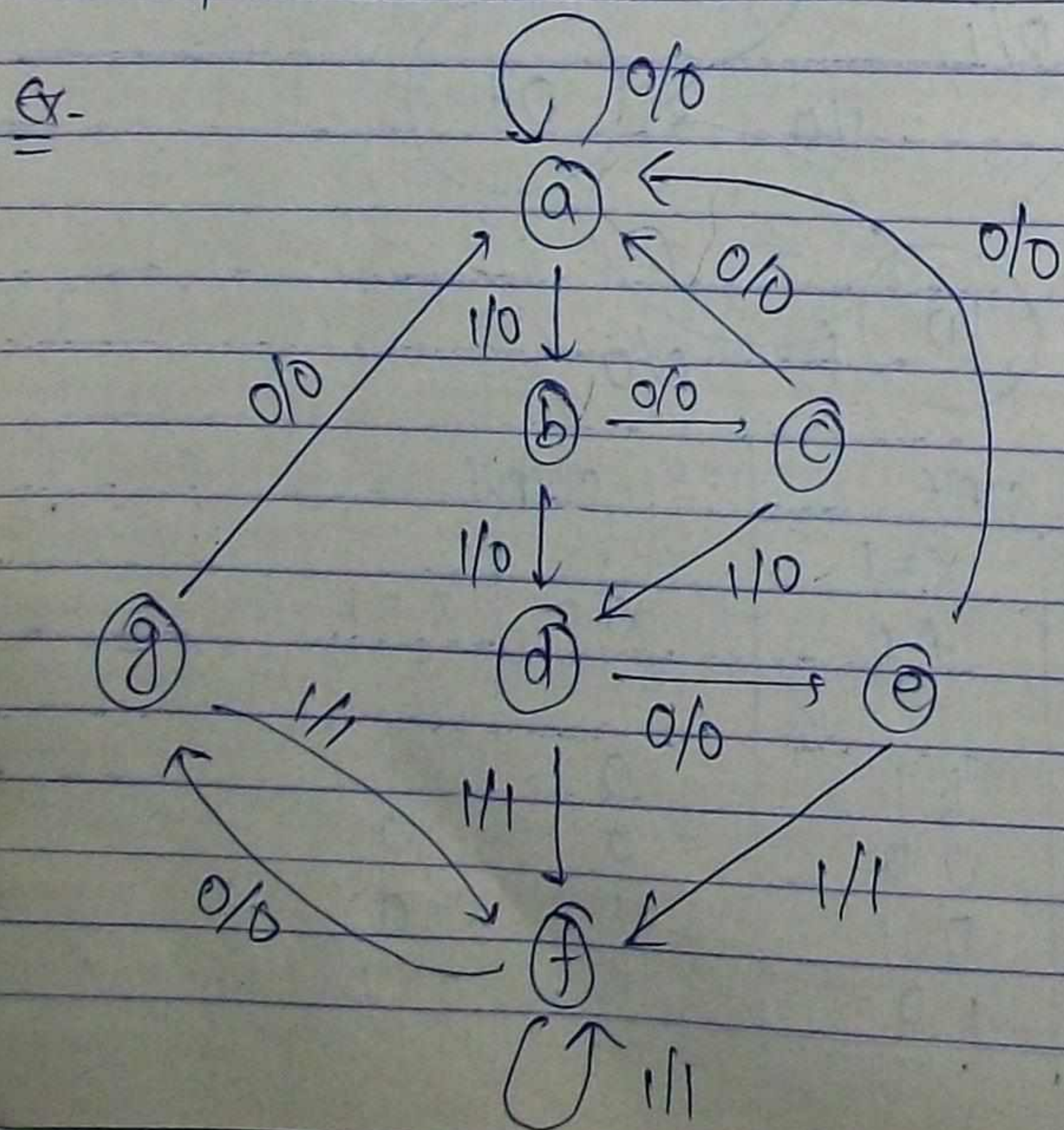
PRESENT STATE A B	NEXT STATE		OUTPUT	
	X=0 A B	X=1 A B	X=0	X=1
0 0	0 1	1 1	0	1
0 1	1 0	0 0	0	0
1 0	1 1	0 1	1	0
1 1	0 0	1 0	1	1

- state diagram is the graphical representation of state table. State is represented in a circle and directed lines indicate state transition.

- X/Y indicates $X \rightarrow I/P$, $Y \rightarrow O/P$

⊕ - Two states are said to be equivalent if their next states and O/Ps are same.

- If two states are equivalent one of the states can be eliminated in this way, circuits can be minimized.



PRESENT STATE	NEXT STATE		OUTPUT	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

$d=f$

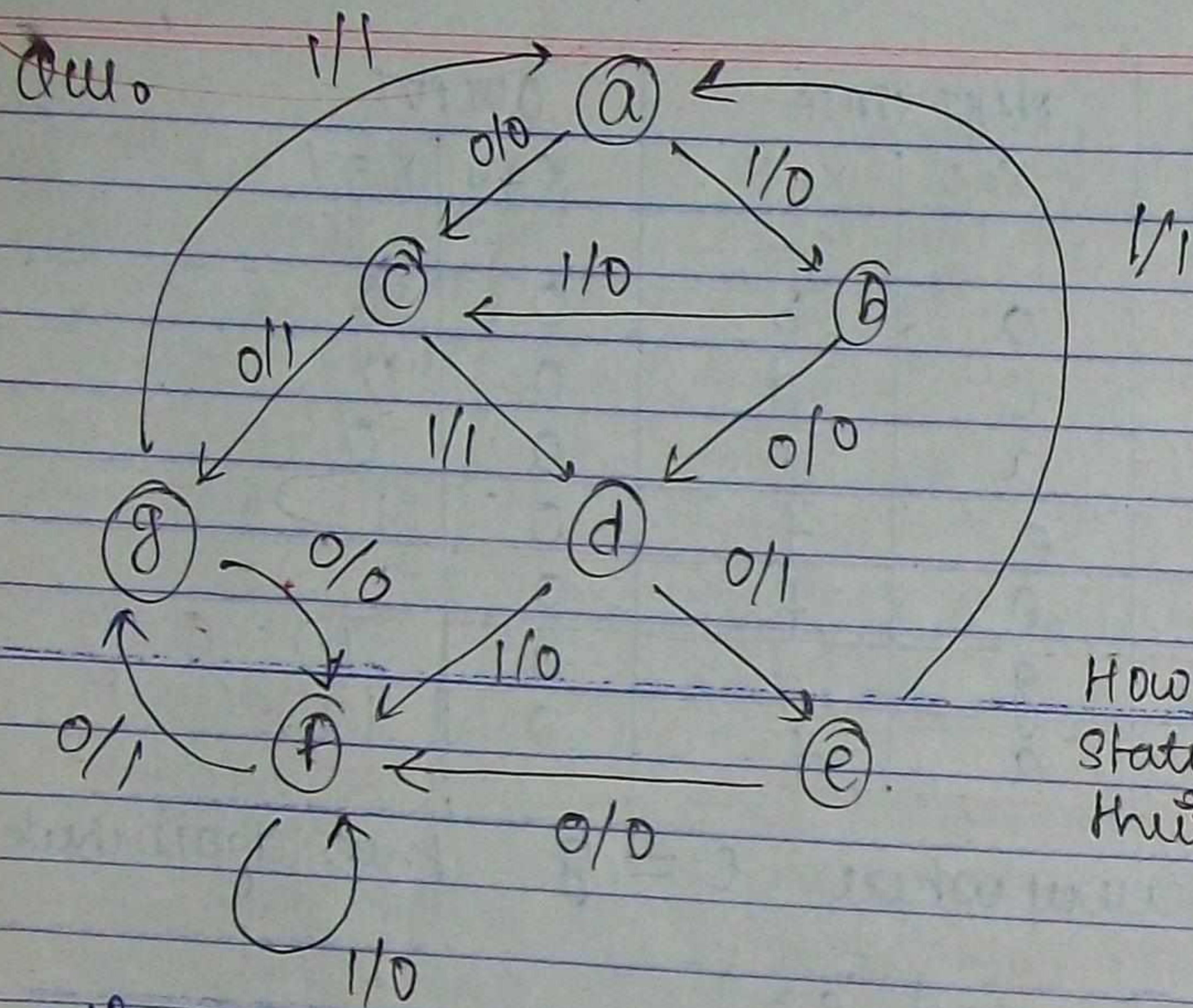
$e=g$

Now put everywhere $e=g$ then again check

so $d=f$ and $e=g$

New table

PRESENT STATE	NEXT STATE		OUTPUT	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1



How many
states are
there?

Solⁿ:

PRESENT STATE	NEXT STATE		OUTPUT	
	X=0	X=1	X=0	X=1
a	c	b	0	0
b	d	c	0	0
c	g	d	1	1
d	e	f	1	0
e	f	a	0	1
f	g	f	1	0
g	f	a	0	1

$d=f$

$e=g$

$e=g$ $d=f$

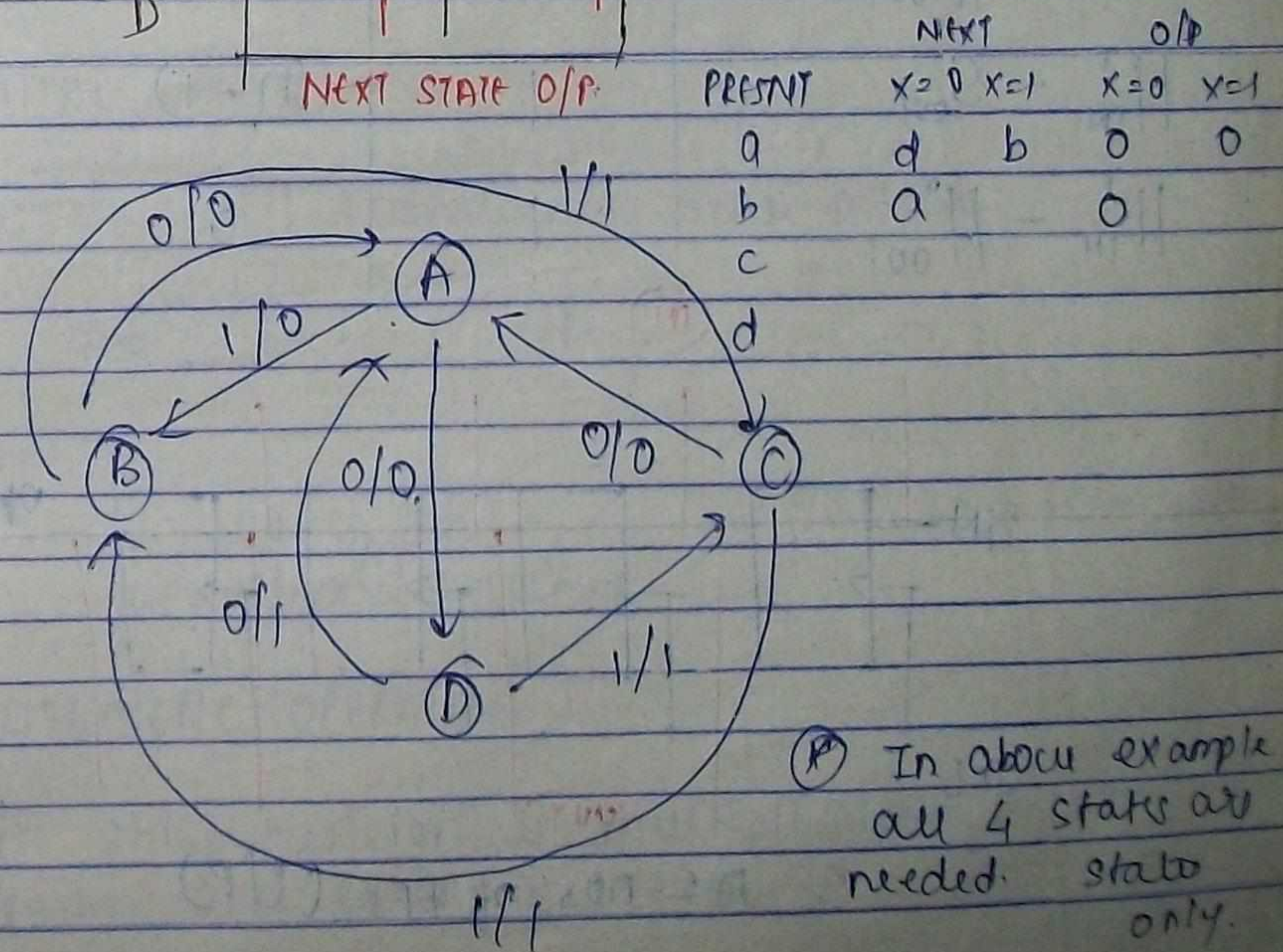
PRESENT STATE	NEXT STATE		OUTPUT	
	X=0	X=1	X=0	X=1
a	c	b	0	0
b	d	c	0	0
c	g	d	1	1
d	e	f	1	0
e	f	a	0	1

Ques.

	0	1
A	D 0	B 0
B	A 0	C 1
C	A 0	B 1
D	A 1	C 1

NEXT STATE O/P

Solⁿ:

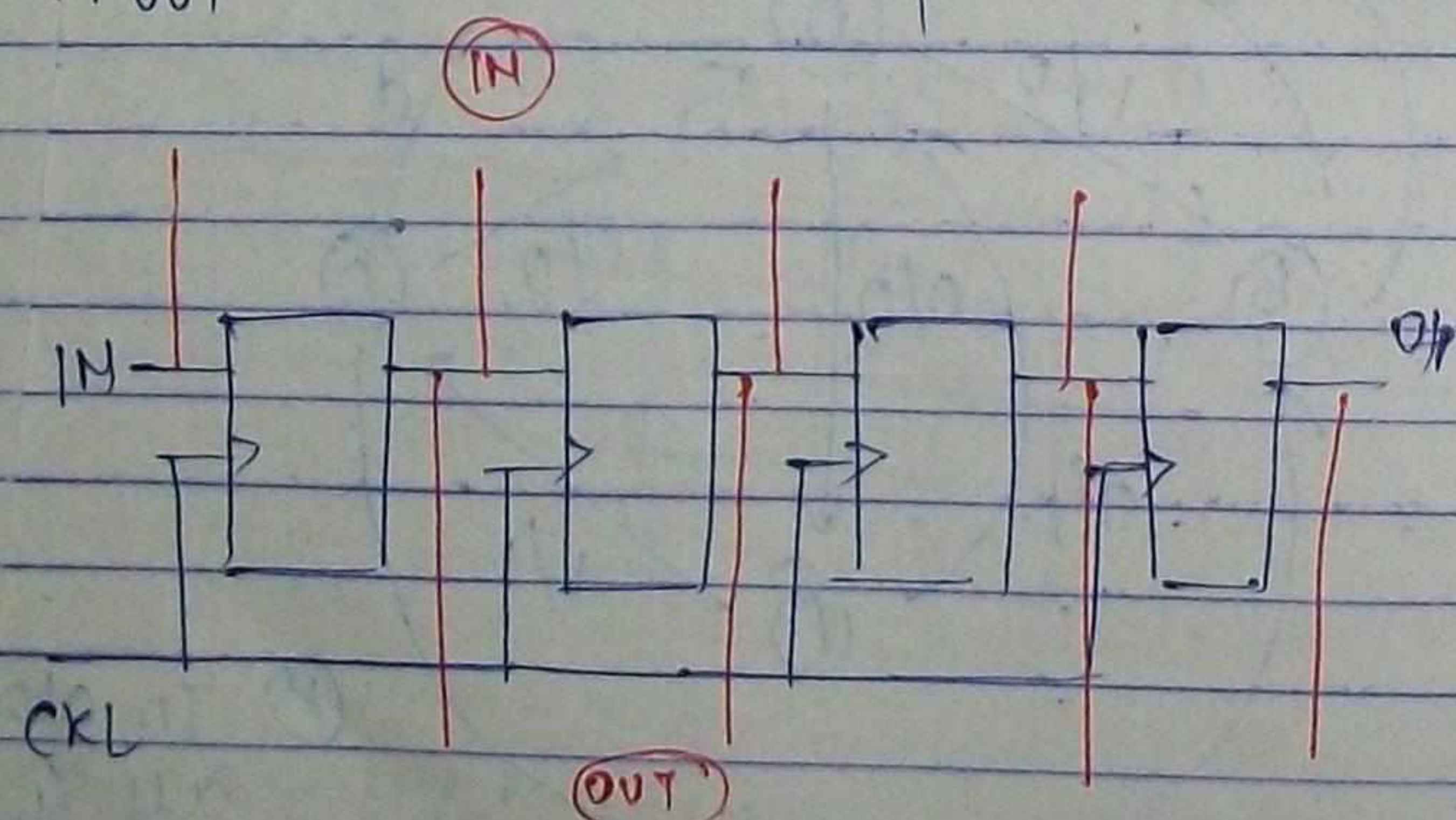


SHIFT

⇒ RESISTORS -

- when ffs are connected in cascaded manner is known as resistor
- four types of resistors:

RESISTORS TYPES	CLK REQUIRED	
	I/P	O/P
$S_{IN} - S_{OUT}$	n	$(n-1)$
$S_{IN} - \parallel_{OUT}$	n	0
$\parallel_{IN} - S_{OUT}$	1	$(n-1)$
$\parallel_{IN} - \parallel_{OUT}$	1	0



$n = \text{no. of ffs (I/P)}$

⇒ APPLICATION OF SHIFT RESISTORS -

- Shift resistors are temporary data storage element (MEMORY ELEMENT)
- (DATA CONVERSION)

SISO → convert serial data into $\parallel$ form

PISO → convert $\parallel$ data into serial form.

- Serial data - temporal code
- parallel data - spacial code

- (TIME DELAY) -

SISO → will introduce a time delay

$$\Delta t = N \cdot T = N \cdot \frac{1}{f_c}$$

where

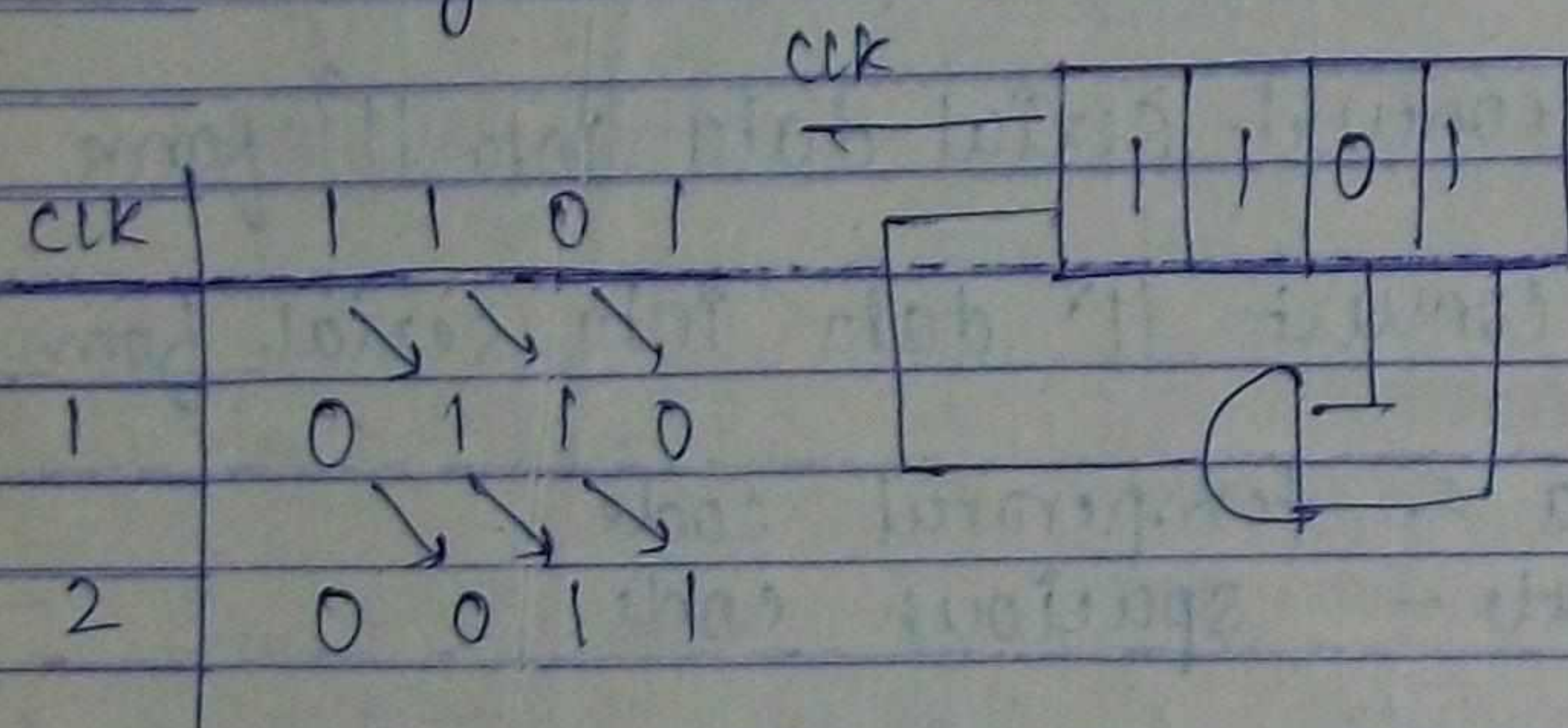
$N = \text{no. of ffs}$, $T = \text{clock free time period}$
 $f_c = \text{clock frequency}$

- (ARITHMETIC OPERATIONS)

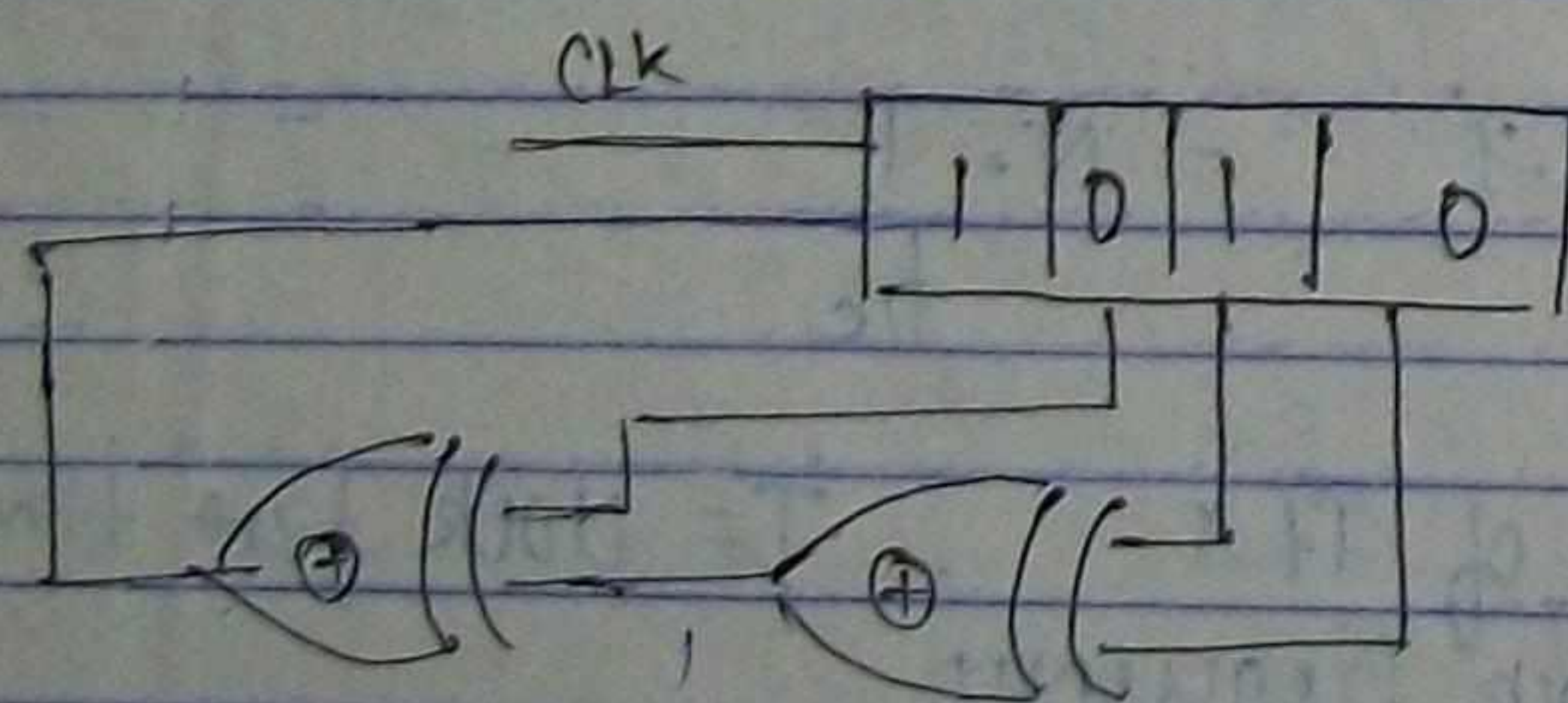
- Left shift resistor is multiplication by 2
- circuit and error is zero.

- Right shift register is divided by 2
 count with error is 0.5 for odd no.

- **RING COUNTERS** -
 Shift register can be converted
 as ring counters.

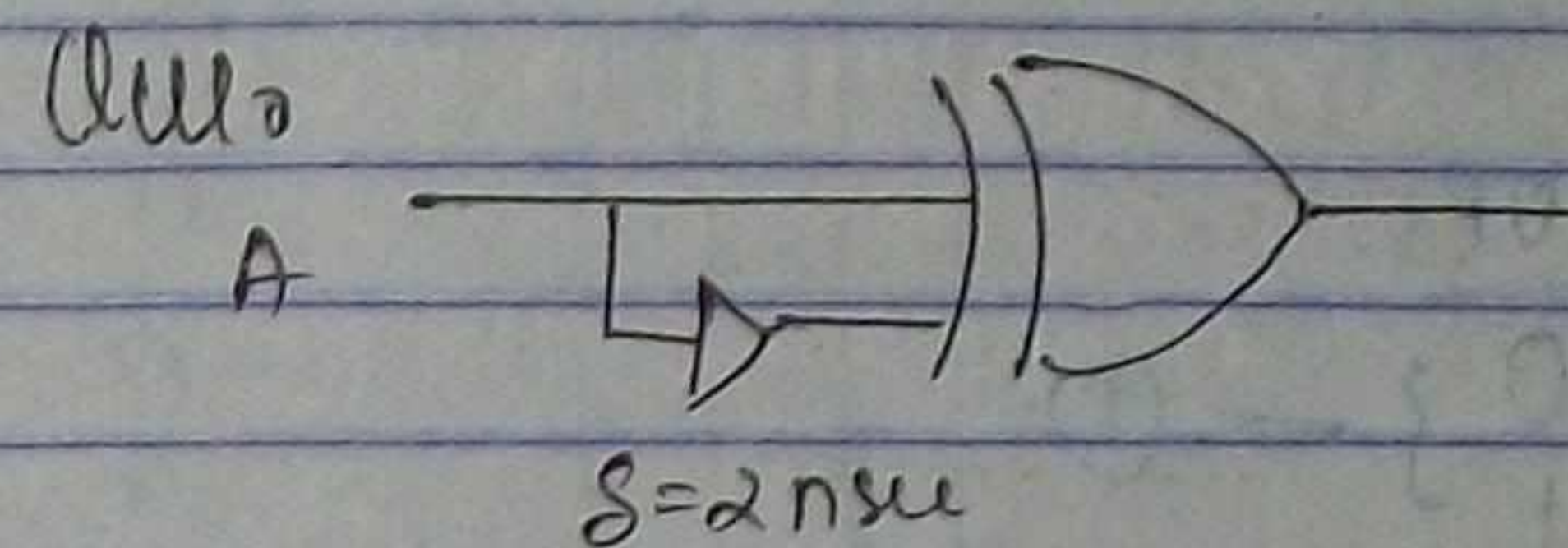


Ques. After how many clk register will get
 the same state

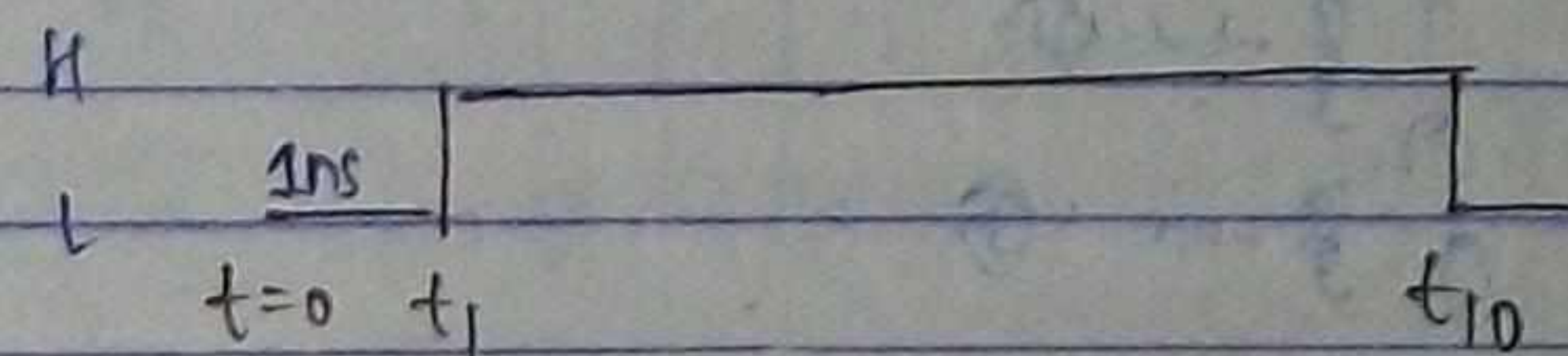


CLK	1	0	1	0
1	1	1	0	1
2	0	1	1	0
3	0	0	1	1

4	0	0	0	1
5	1	0	0	0
6	0	1	0	0
7	1	0	1	0

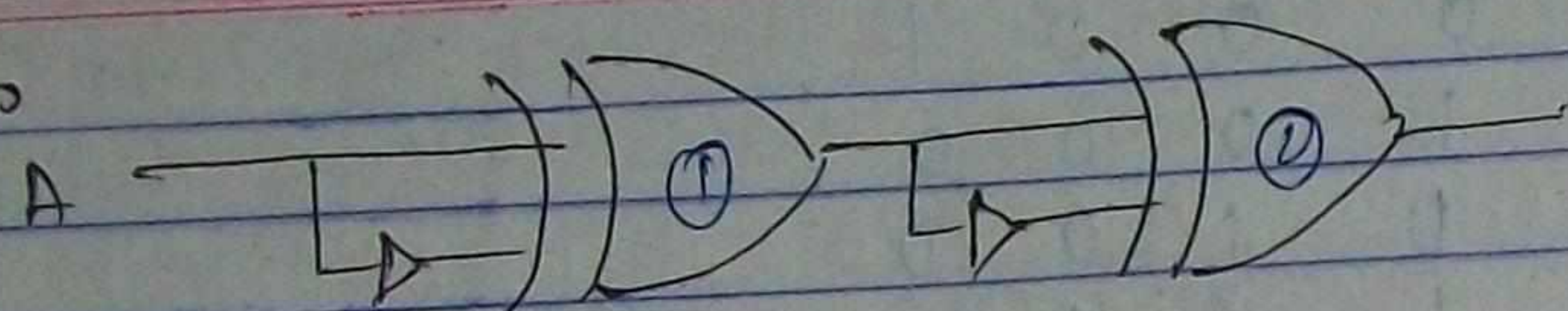


How many
 transition take
 place b/w $t_0 - t_{10}$



t	A	$S = 2 \text{ ns}$	$Y = 0/1$
t_0	0	0	0
t_1	1	0	1
t_2	1	0	1
t_3	1	1	0
t_4	1	1	0
t_5	1	1	0
t_6	1	1	0
t_7	1	1	0
t_8	1	1	0
t_9	1	1	0
t_{10}	1	1	0

Ques



Solⁿ

XOR ₁	S ₂ = 4	XOR ₂
0	0	0
1	0	1
1	0	1
0	0	0
0	0	0
0	1	1
0	0	0
0	0	0
0	0	0
0	0	0

4 change

$$\text{Total} = 2 + 4 = 6$$

Ques. If $x=1$

$$[x + z\{\bar{y} + (\bar{z} + x\bar{y})\}] \cdot \{\bar{x} + \bar{z}(x+y)\} = 1$$

Solⁿ

$$0 + \bar{z}(1+y) = 1$$

$$\bar{z} = 1$$

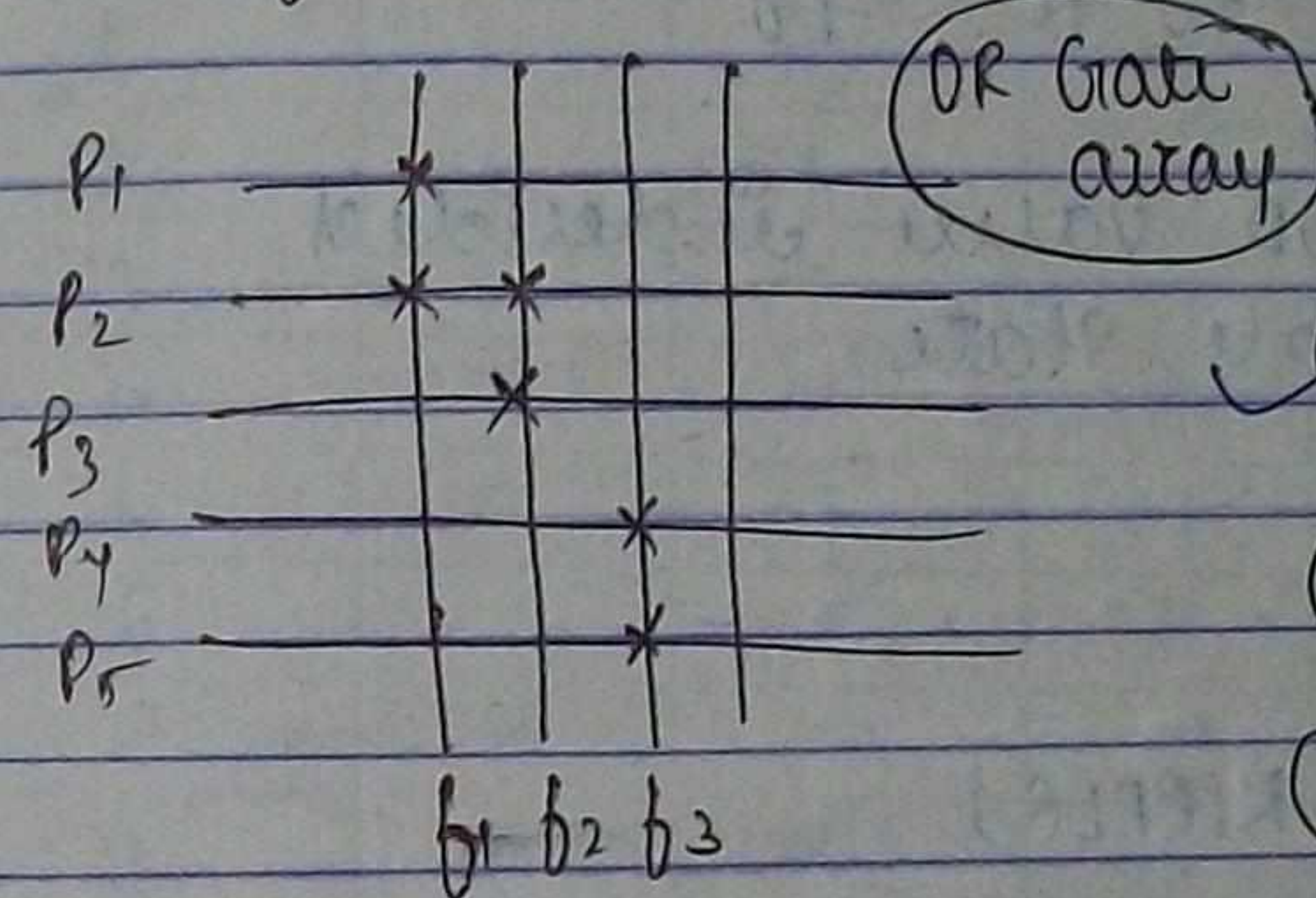
$$z = 0$$

$$\text{Ques. } b_1 = a\bar{b}c + \bar{a}b\bar{c} + bc$$

$$b_2 = a\bar{b}c + ab + \bar{a}\bar{b}\bar{c}$$

$$b_3 = \bar{a}\bar{b}\bar{c} + abc + \bar{a}c$$

$$P_1, P_2, P_3, P_4, P_5 = ?$$



$$\text{① } \bar{a}b, ac, b\bar{c}, bc, \bar{a}\bar{b}$$

$$\text{② } a\bar{b}, bc, ac, \bar{a}\bar{b}, bc$$

$$\text{③ } ac, \bar{a}b, b\bar{c}, \bar{a}\bar{b}, bc$$

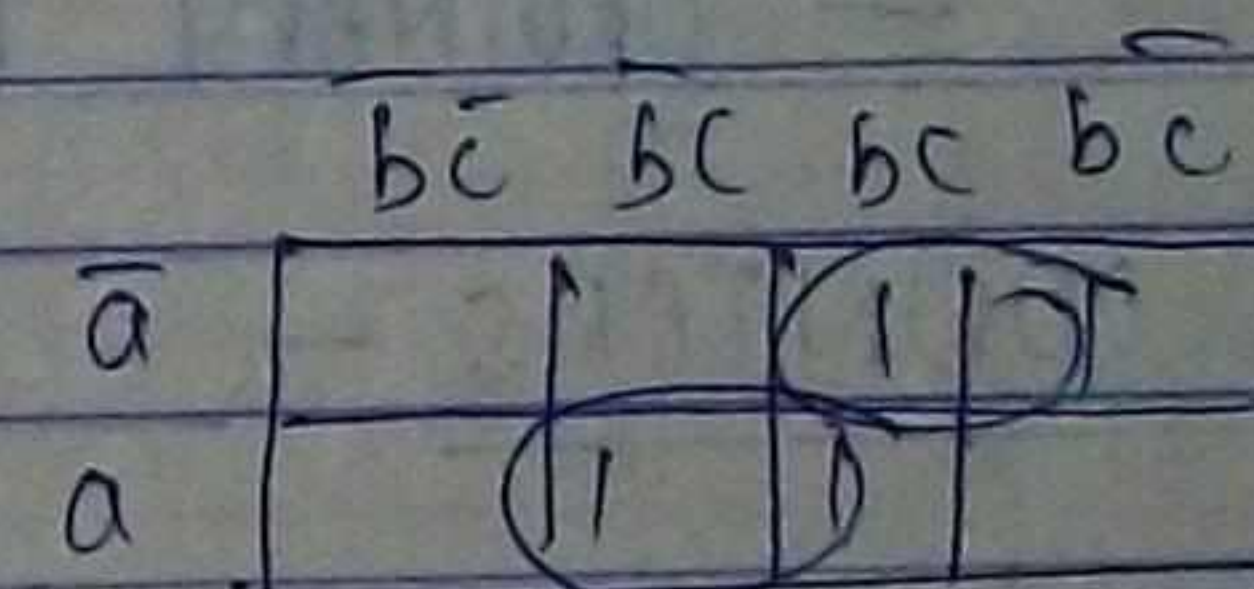
$$\text{④ None.}$$

Solⁿ

$$b_1 = P_1 + P_2$$

$$b_2 = P_2 + P_3$$

$$b_3 = P_4 + P_5$$



$$b_1 = ac + \bar{a}b$$

$$b_2 = ac + b\bar{c}$$

$$P_2 = ac$$

$$P_1 = \bar{a}b$$

$$P_3 = b\bar{c}$$

13/10/11

⇒ COUNTERS -

- It is the unit which counts number of clock pulses applied to it. fo.

- for any counter MOD value is decided by the no. of possible state.

- Types :

① SYNCHRONOUS

② ASYNCHRONOUS (RIPPLE)

③ RING COUNTERS

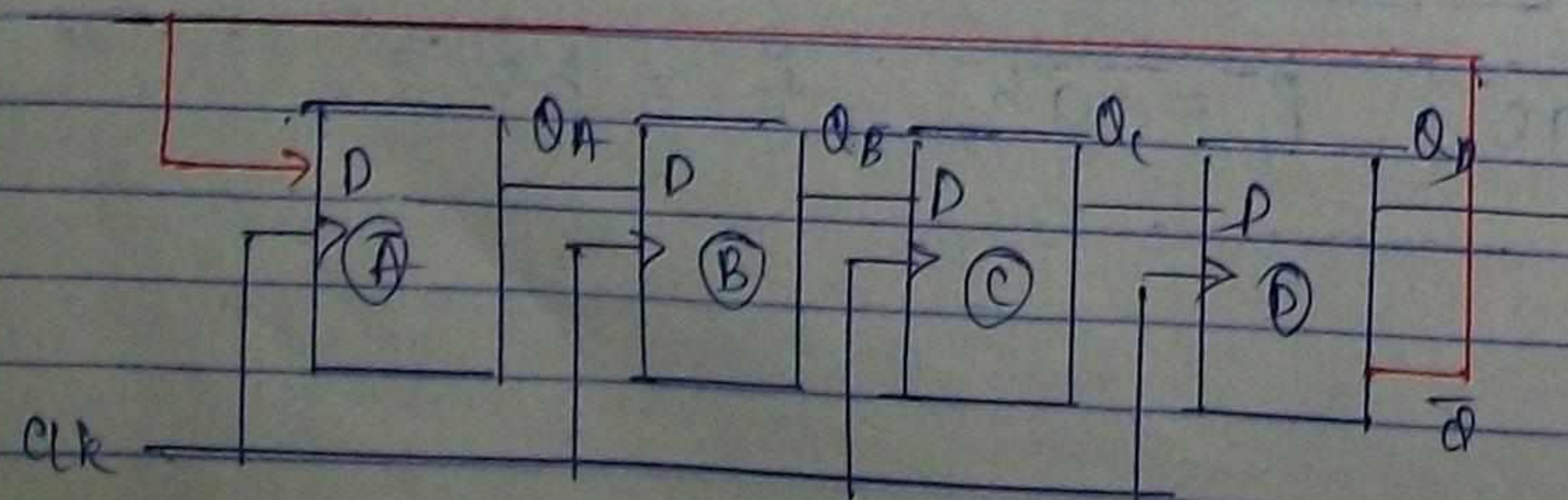
— JOHNSON'S RING COUNTER (TWISTED)

— ORDINARY RING COUNTER

⇒ RING COUNTERS -

① JOHNSON'S RING COUNTER (OR) TWISTED RING COUNTER -

- The feedback is given with $\bar{Q}$ of final flip flop



CLK	Q _A	Q _B	Q _C	Q _D
0 th	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0

MOD-8

NO. OF POSSIBLE STATE (MOD) = $2n$

= NO. of clock pulse required

$n = \text{no. of clock pulse required}$

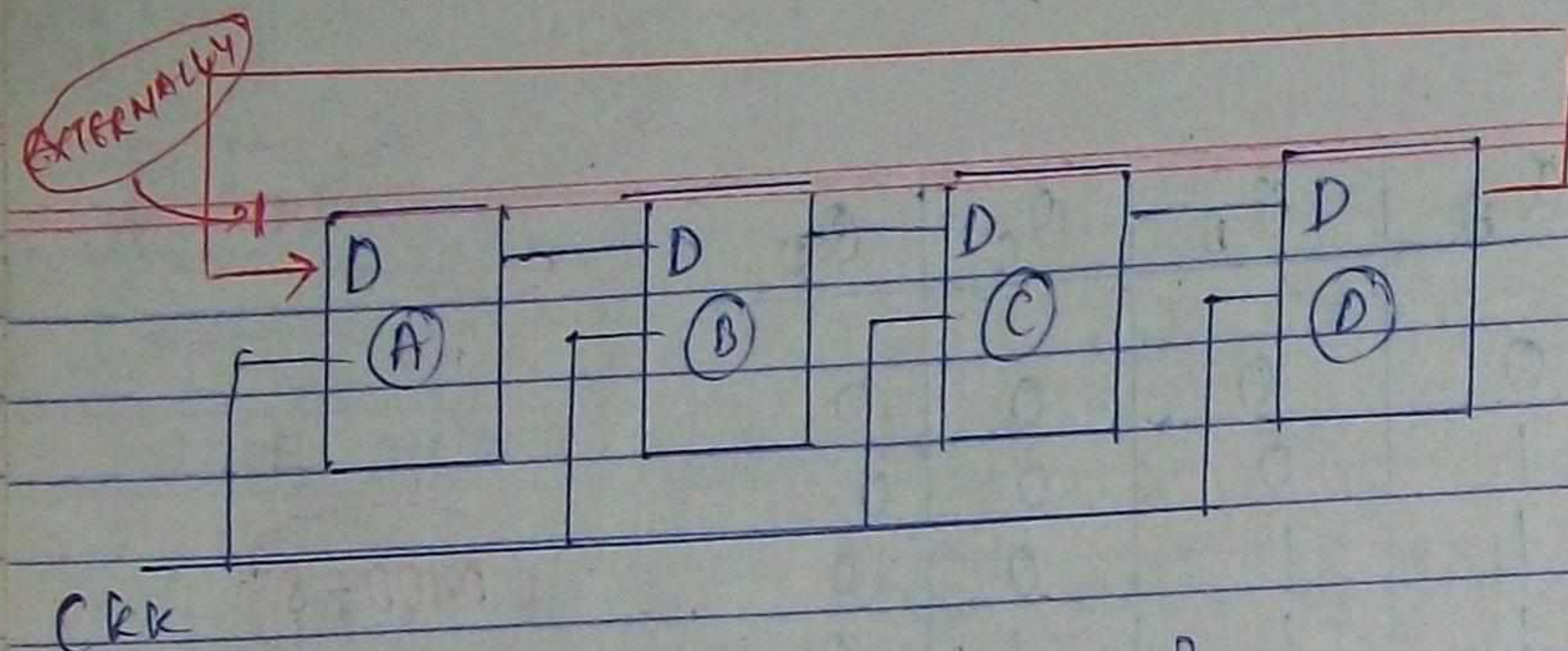
② ORDINARY RING COUNTERS -

- In ordinary ring counter feedback is given with uncomplementary output of the final ff.

NO. OF POSSIBLE STATE = n

= no. of clk required

$n = \text{no. of clk pulse}$



CLK	Q _A	Q _B	Q _C	Q _D
0 th	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	0	0	0	0

also with
J-k ff
we will
make JK
Ring counter

⇒ ASYNCHRONOUS (OR) RIPPLE COUNTER -

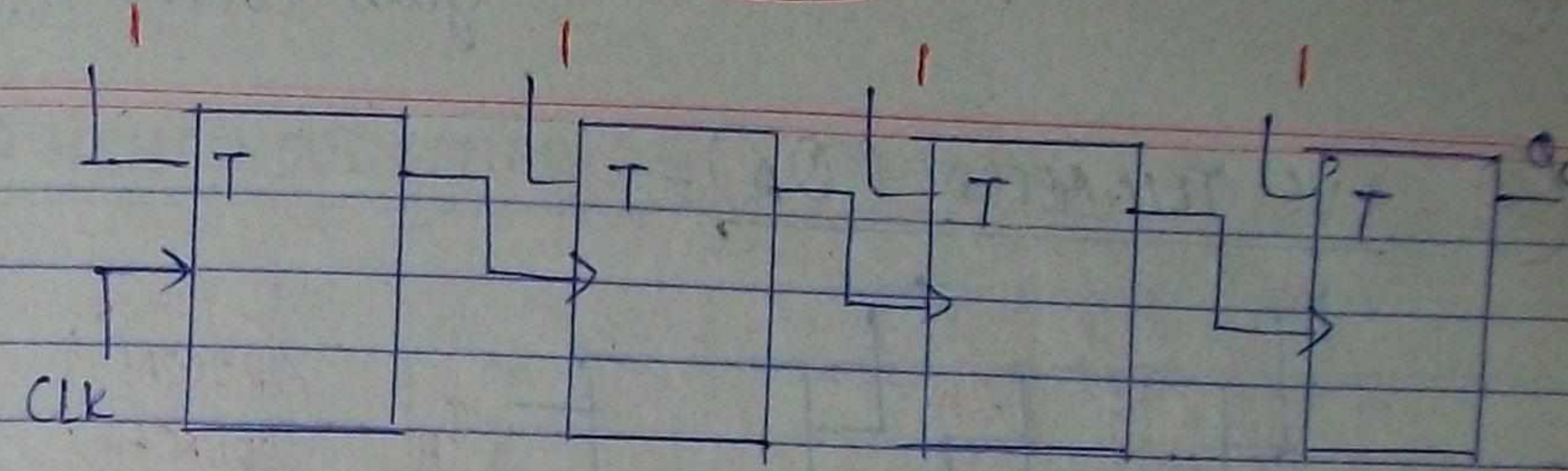
- In asynchronous counter flip-flops are triggered at different time (unsimultaneously or asynchronously).

$$\text{NO. OF POSSIBLE STATES} = 2^n$$

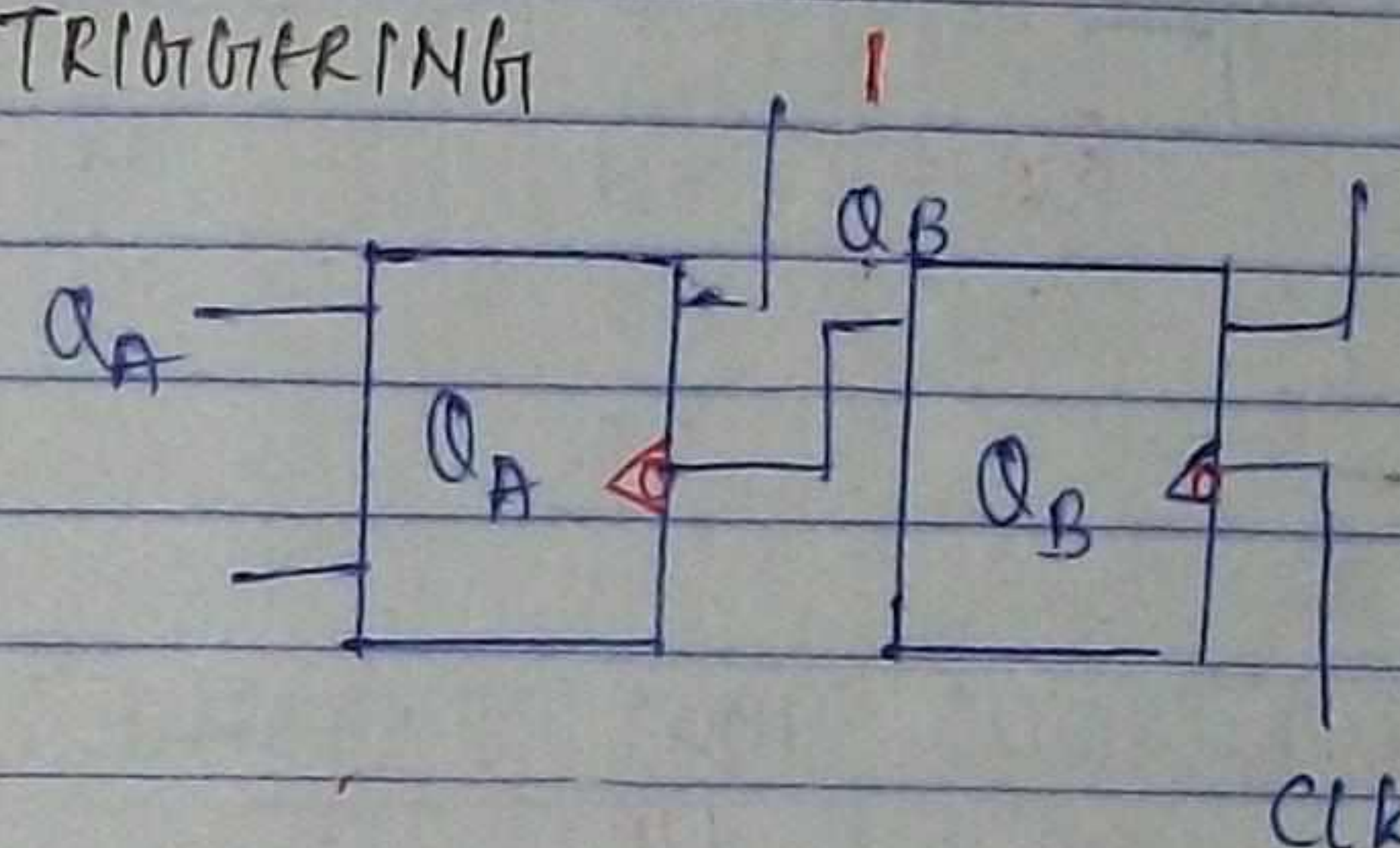
→ CLK PULSE

$$\text{DECIMAL EQUIVALENT OF CLK PULSE} = 2^n - 1$$

EXTERNALLY

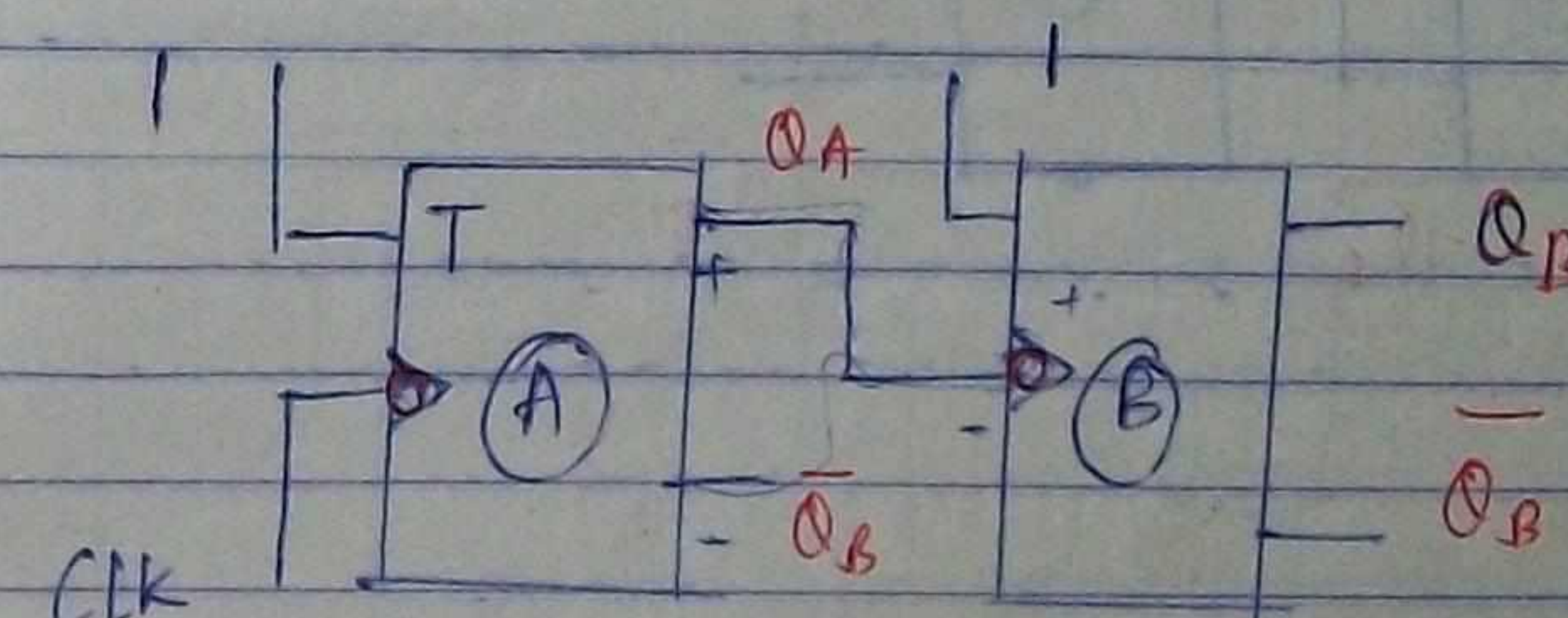


NEGATIVE
TRIGGERING



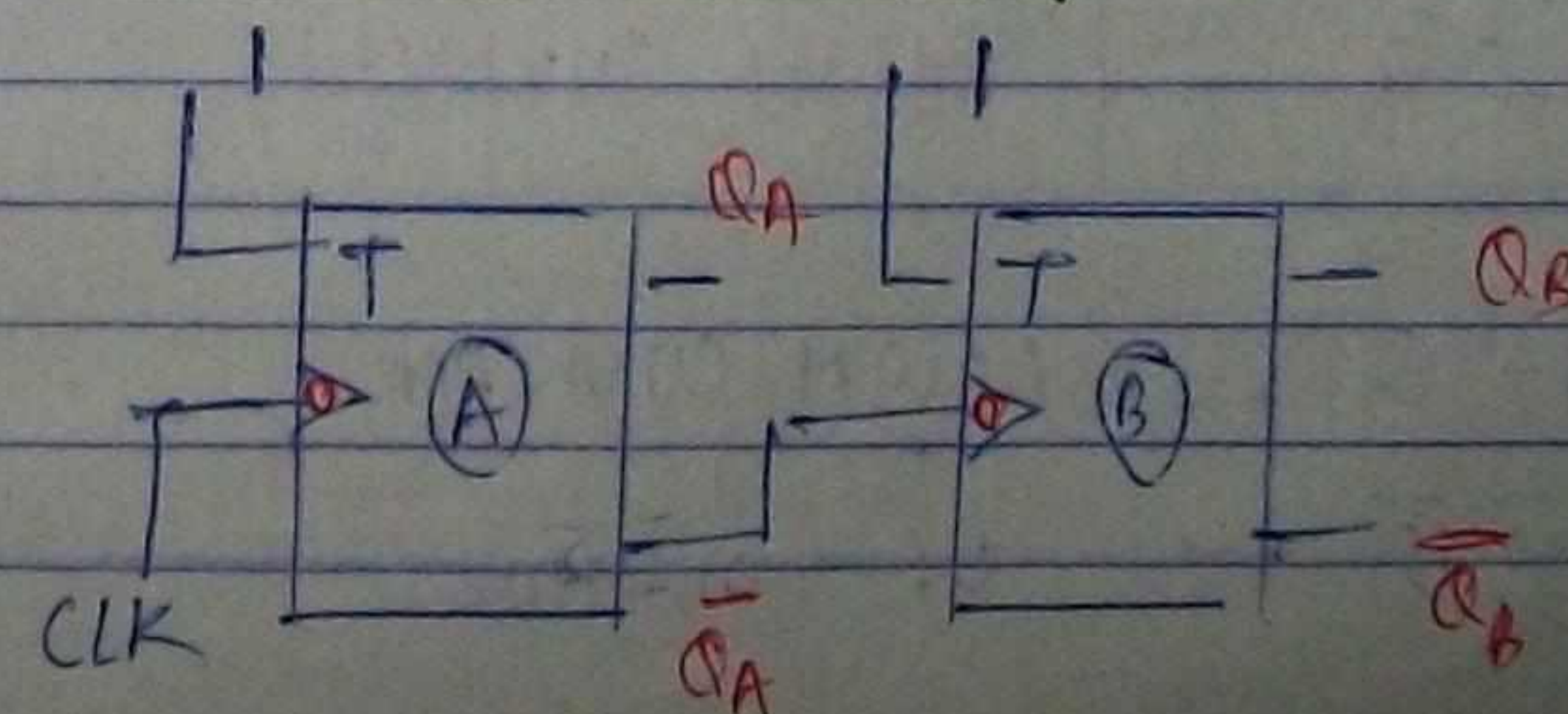
CLK	Q _A	Q _B
0	0	0
1	0	1
2	1	0
3	1	1

① -VE TRIGGERED (Q_A) -



UP
COUNTER

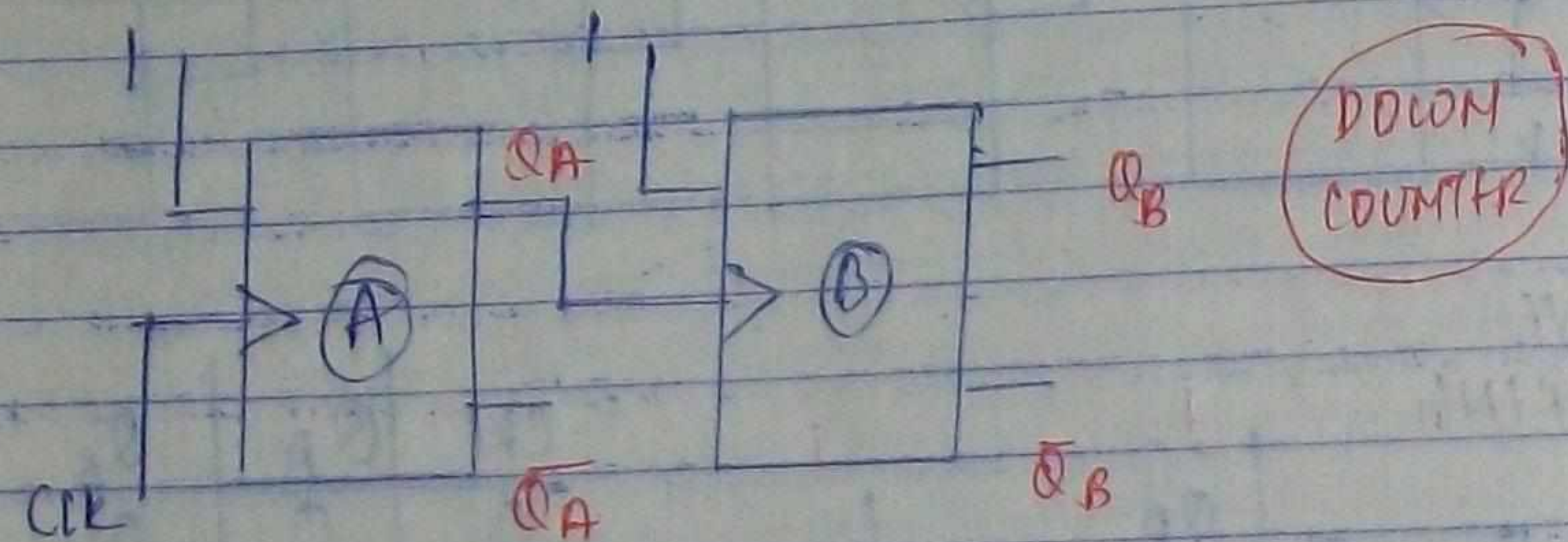
② -VE TRIGGERED (Q_A) -



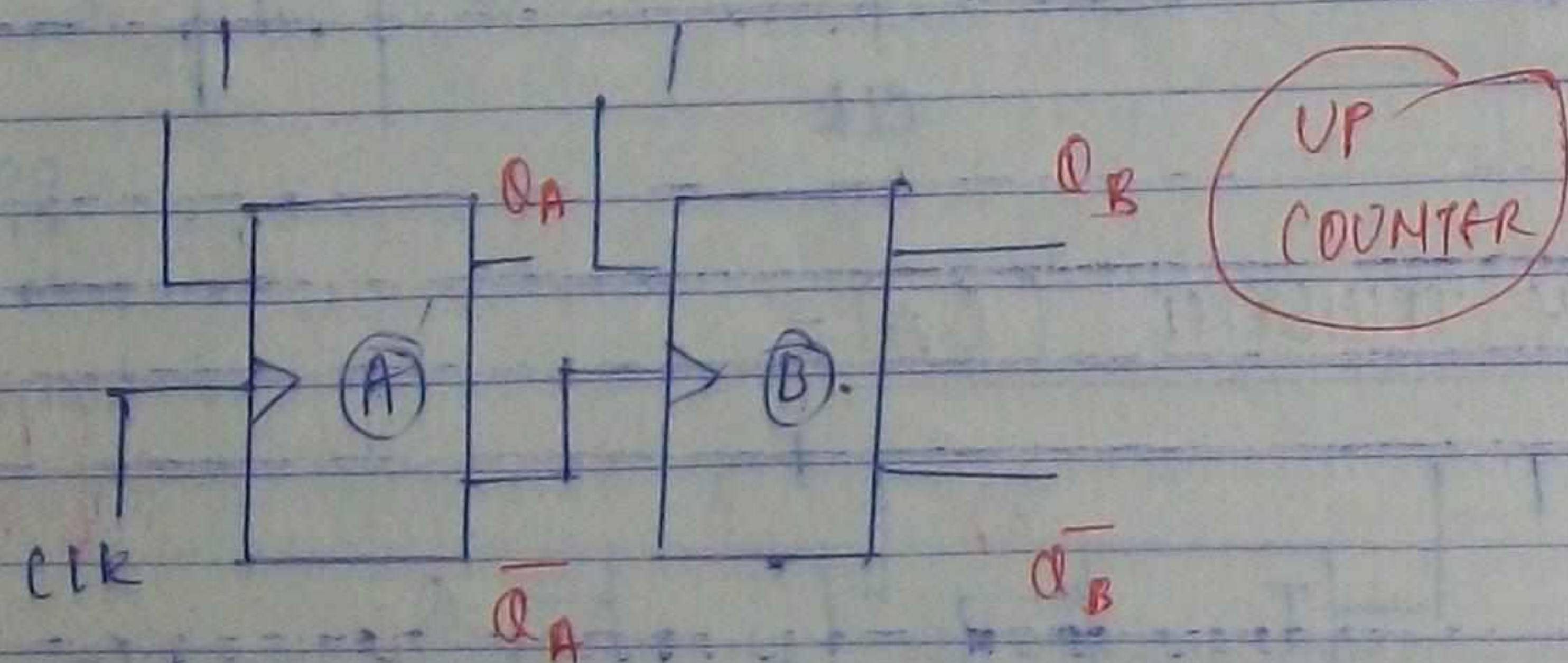
DOWN
COUNTER

⊛ valid for counters without any other gates connected.

③ +VE TRIGGERED (Q_A) -



④ +VE TRIGGERED ($\bar{Q}_A$) -

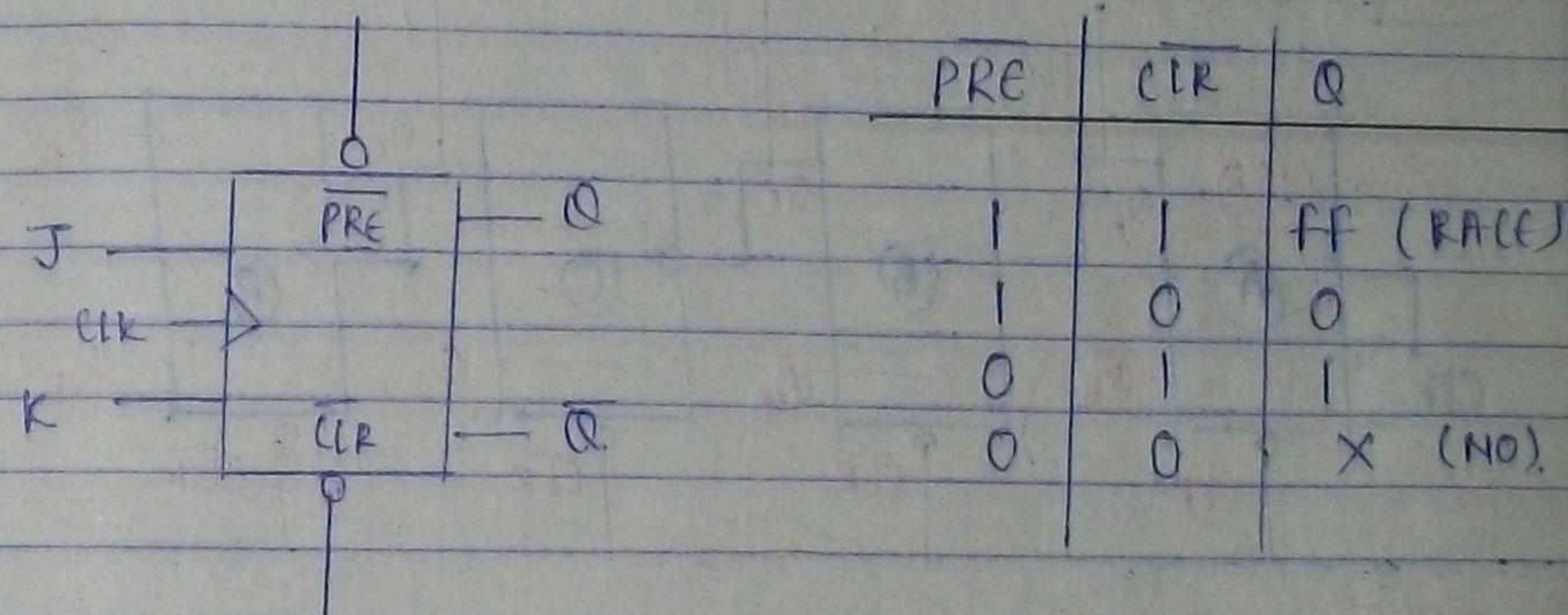


SHORT TRICK -

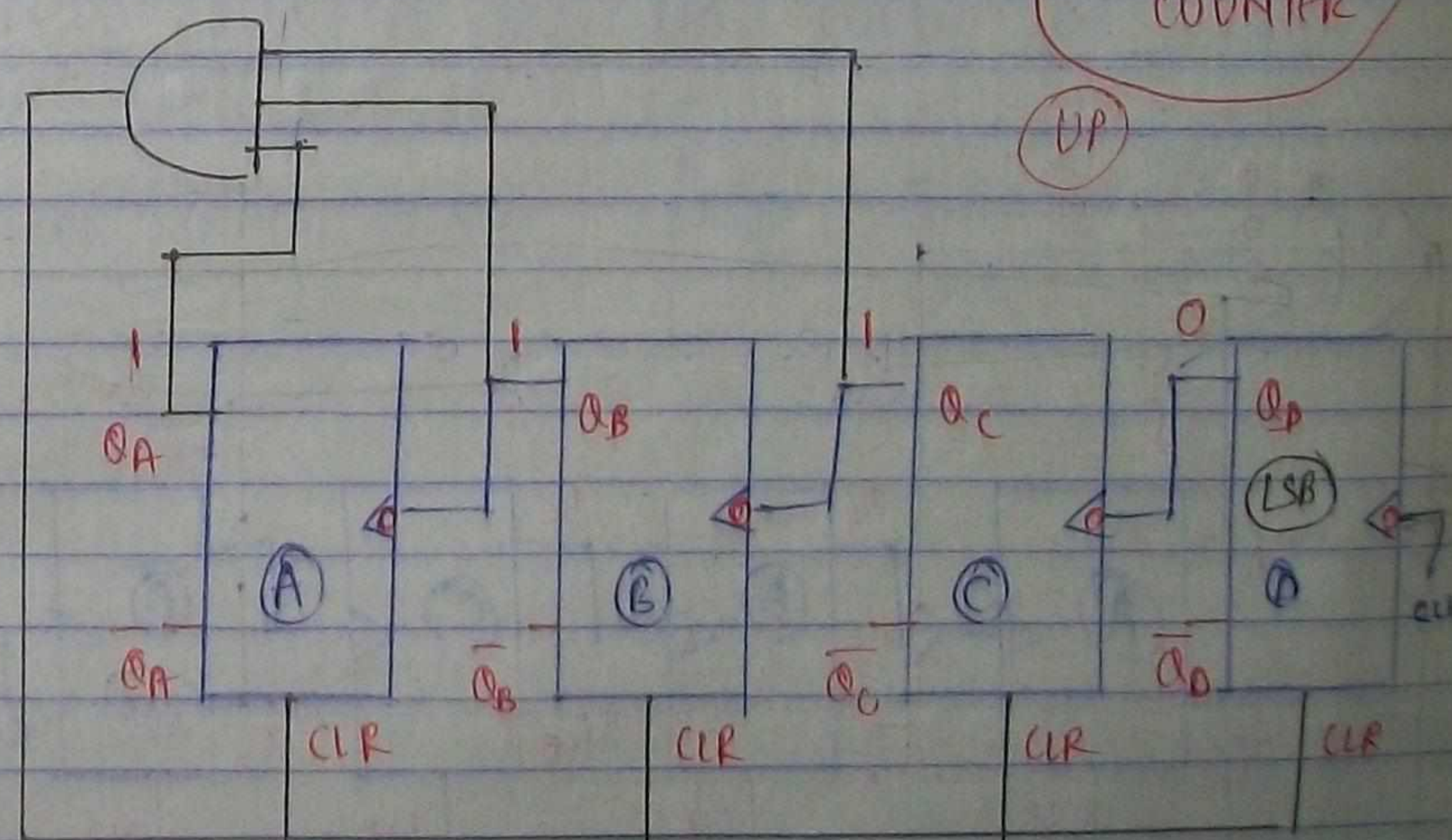
(-VE) TRIGGER $\begin{cases} Q \rightarrow \text{UP COUNTER} \\ \bar{Q} \rightarrow \text{DOWN COUNTER} \end{cases}$

(+VE) TRIGGER $\begin{cases} Q \rightarrow \text{DOWN COUNTER} \\ \bar{Q} \rightarrow \text{UP COUNTER} \end{cases}$

$\Rightarrow$ ASYNCHRONOUS INPUTS -

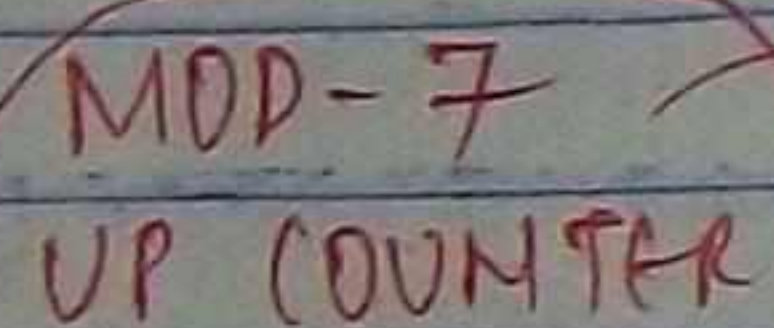


$\Rightarrow$ DIFFERENT MOD COUNTER -



1110 = 14
 $\downarrow$ LSB

⊛ FIRST I/P CLK FF IS ALWAYS LSB



MOD - 13
COUNTER

1 1 0 1 0010 0_p
L TAKE O/P
USB FROM Q

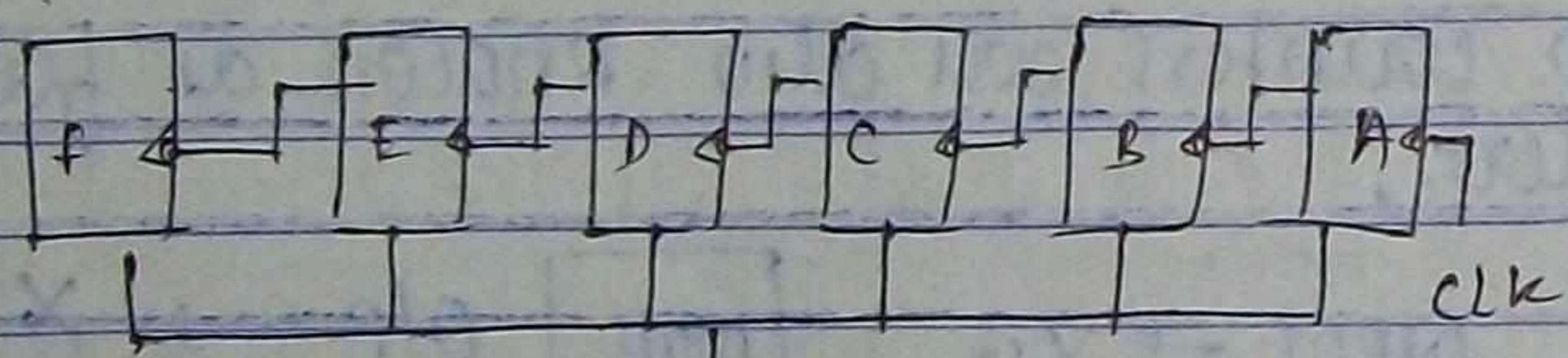
MOD
M X N
COUNTER

Que. A mod 3 counter is followed by mod 4 and mod 5 counter if i/p freq is 12 MHz then o/p freq.

Solⁿ: MOD 3 → MOD 4 → MOD 5
 = MOD 3 × 4 × 5
 = MOD 60

$$\text{o/p freq} = \frac{12 \times 10^6}{60} = 200 \text{ kHz}$$

Que.



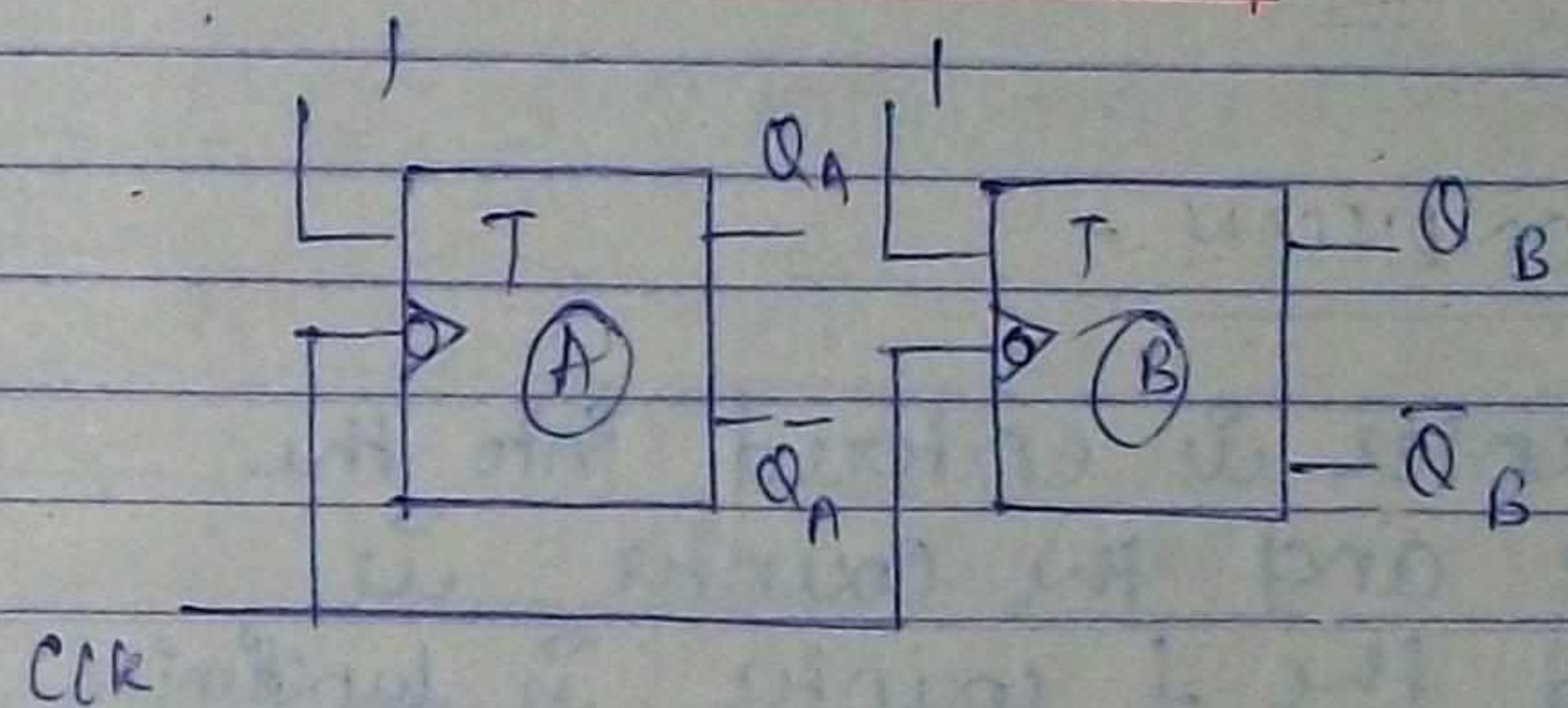
which ffs are connected to map gate to get o/p freq of 1 kHz if i/p frequency is 60 kHz

Ans. MOD 60 — 111100
 Ans → C D E F

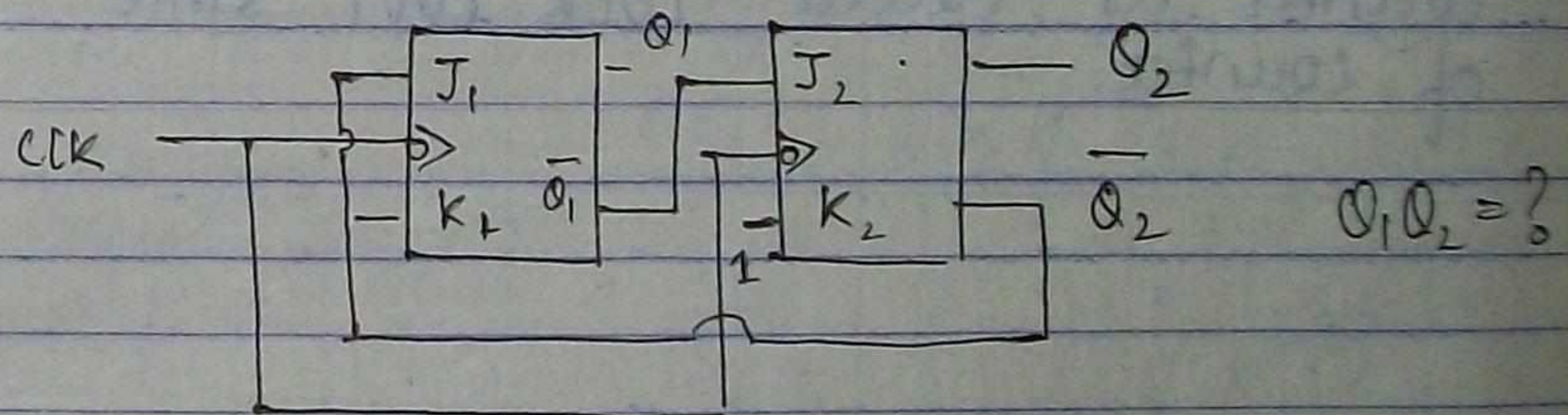
⇒ SYNCHRONOUS COUNTER -

- All the FF's of this particular counter are triggered simultaneously.

NO. OF POSSIBLE STATES = 2^n also clock pulse required



Que.



Solⁿ:

CLK	$\bar{Q}_2$	$\bar{Q}_1$	$\bar{Q}_0$	1	Q_1	Q_2
0					0	0
1	1	1	1	1	1	1
2	0	0	0	1	1	0
3	1	1	1	1	0	0

MOD-3 COUNTER

NOTE -

- Worst case delay in asynchronous counter is $= n t_p$

n = no. of ff's

- In case of synchronous counter delay is $= t_p$

⇒ LOCK-OUT CONDITION -

- When a counter is entered into the unused state and the counter is function and the counter is functioning only in unused state then the counter is called LOCK OUT STATE of counter.

⇒ DESIGNING OF SEQUENCE GENERATOR - (SYNCHRONOUS - COUNTER)

S - STATE DIAGRAM

N - NEXT TABLE

E - EXCITATION TABLE

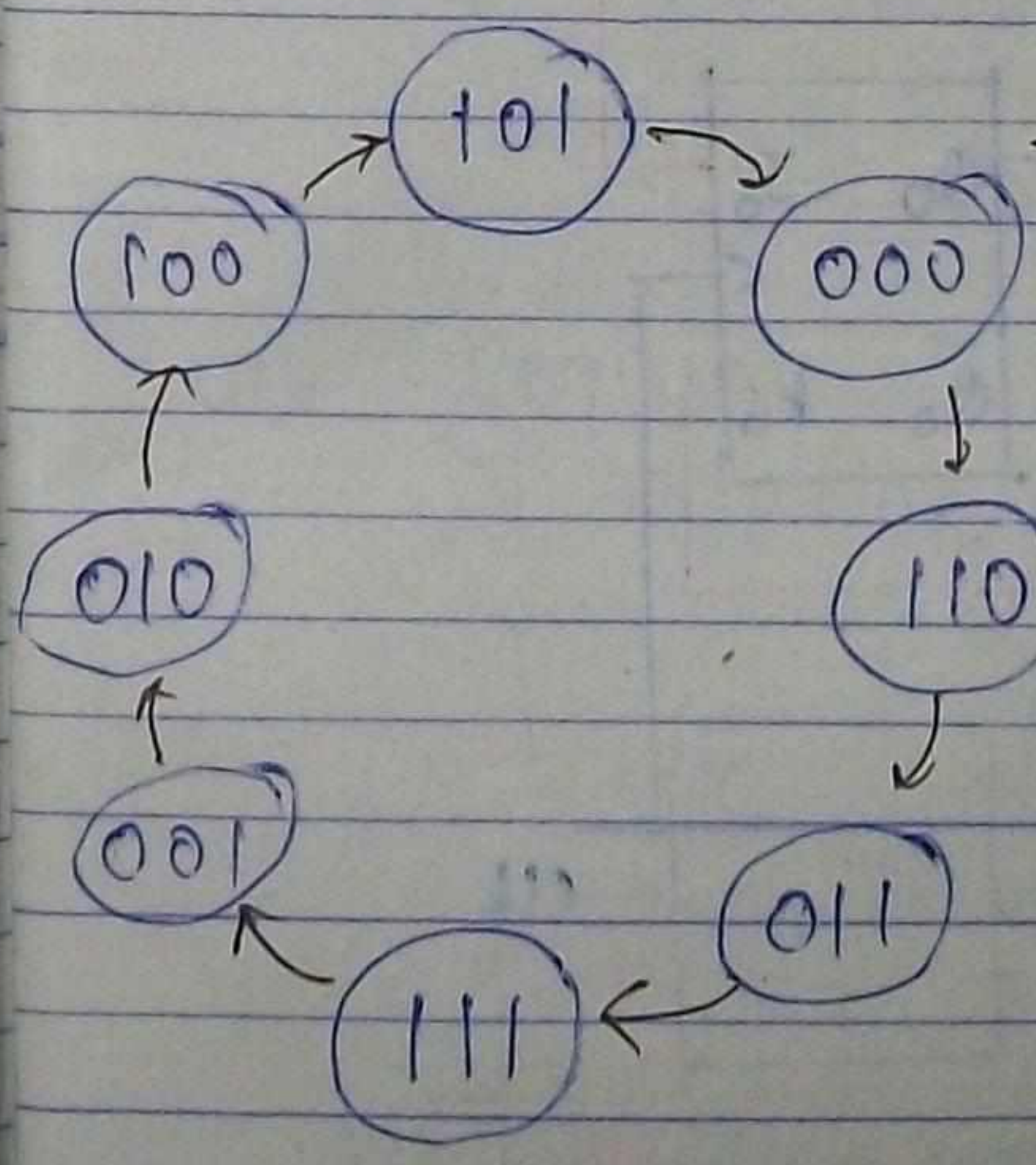
K - K-MAP

B - BOOLEAN EQUATION

D - DESIGN

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

	PRESENT STATE Q_2, Q_1, Q_0	NEXT STATE Q_2^+, Q_1^+, Q_0^+	J_0, K_0	J_1, K_1	J_2, K_2
5	101	000	X 1	0 X	X 1
0	000	110	0 X	1 X	1 X
6	110	011	1 X	X 0	X 1
3	011	111	X 0	X 0	1 X
7	111	001	X 0	X 1	X 1
1	001	010	X 1	1 X	0 X
2	010	100	0 X	X 1	0 X
4	100	101	1 X	0 X	X 0



$$\begin{array}{c} \bar{Q}_2 \\ Q_2 \end{array} \begin{array}{c} 0,0,0,0,0,0,0,0 \\ 1,0,0,0,0,0,0,0 \end{array}$$

	X	X	
1	X	X	1

$$J_0 = Q_2$$

$$\begin{array}{c} \bar{Q}_2 \\ Q_2 \end{array} \begin{array}{c} 0,0,0,0,0,0,0,0 \\ 0,0,0,0,0,0,0,0 \end{array}$$

1	X	X	1
0	0	0	0

$$J_1 = \bar{Q}_2$$

$$\begin{array}{c} \bar{Q}_2 \\ Q_2 \end{array} \begin{array}{c} 0,0,0,0,0,0,0,0 \\ 0,0,0,0,0,0,0,0 \end{array}$$

1	0	1	1
X	X	X	X

$$J_2 = Q_1 + \bar{Q}_2 \bar{Q}_0$$

$$\begin{array}{c} \bar{Q}_2 \\ Q_2 \end{array} \begin{array}{c} 0,0,0,0,0,0,0,0 \\ 0,0,0,0,0,0,0,0 \end{array}$$

X	1		X
X	1		X

$$K_0 = \bar{Q}_1$$

$$\begin{array}{c} \bar{Q}_2 \\ Q_2 \end{array} \begin{array}{c} 0,0,0,0,0,0,0,0 \\ 0,0,0,0,0,0,0,0 \end{array}$$

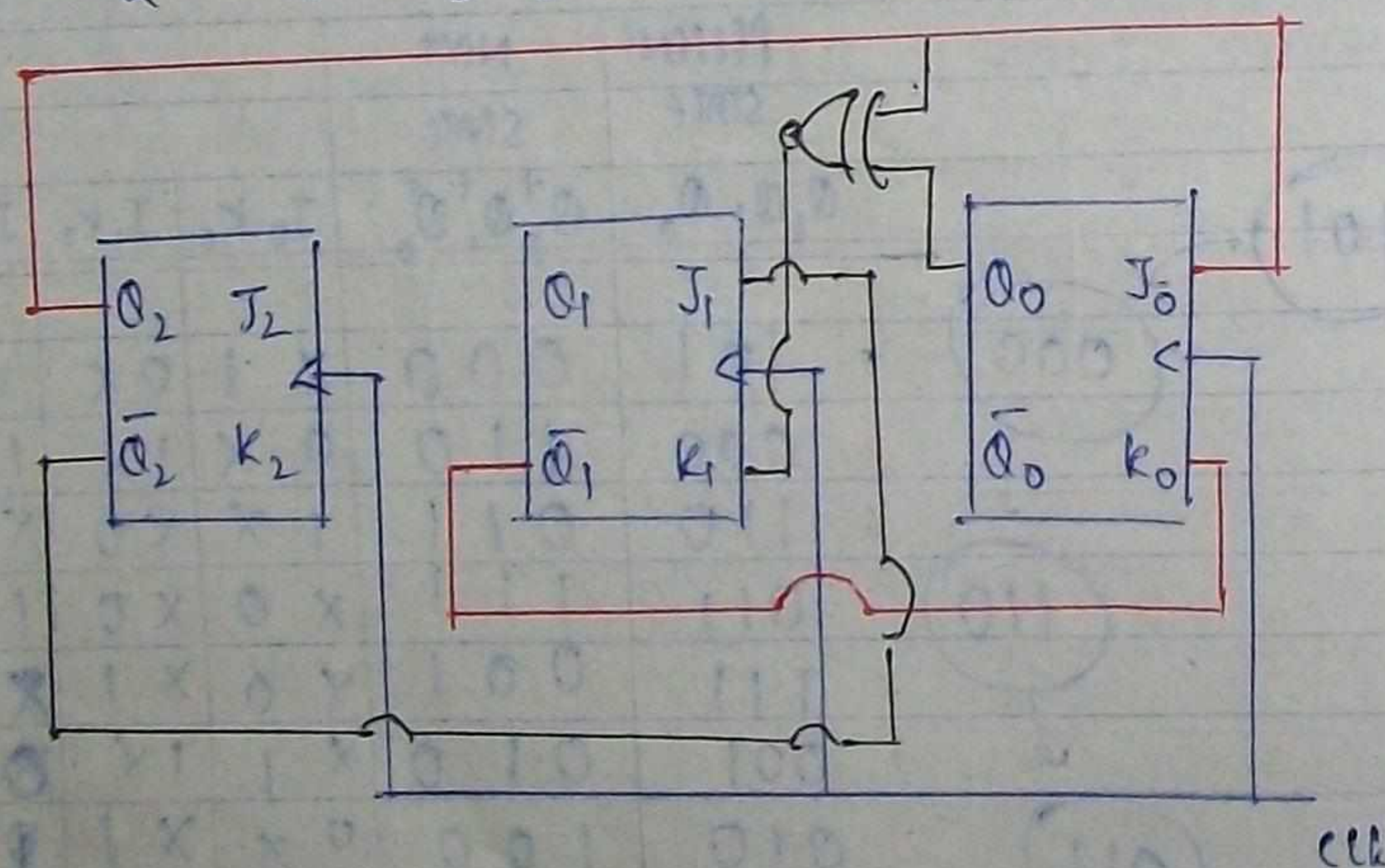
X	X	0	1
X	X	1	0

$$K_1 = Q_0 \oplus Q_2$$

$$\begin{array}{c} \bar{Q}_2 \\ Q_2 \end{array} \begin{array}{c} 0,0,0,0,0,0,0,0 \\ 0,0,0,0,0,0,0,0 \end{array}$$

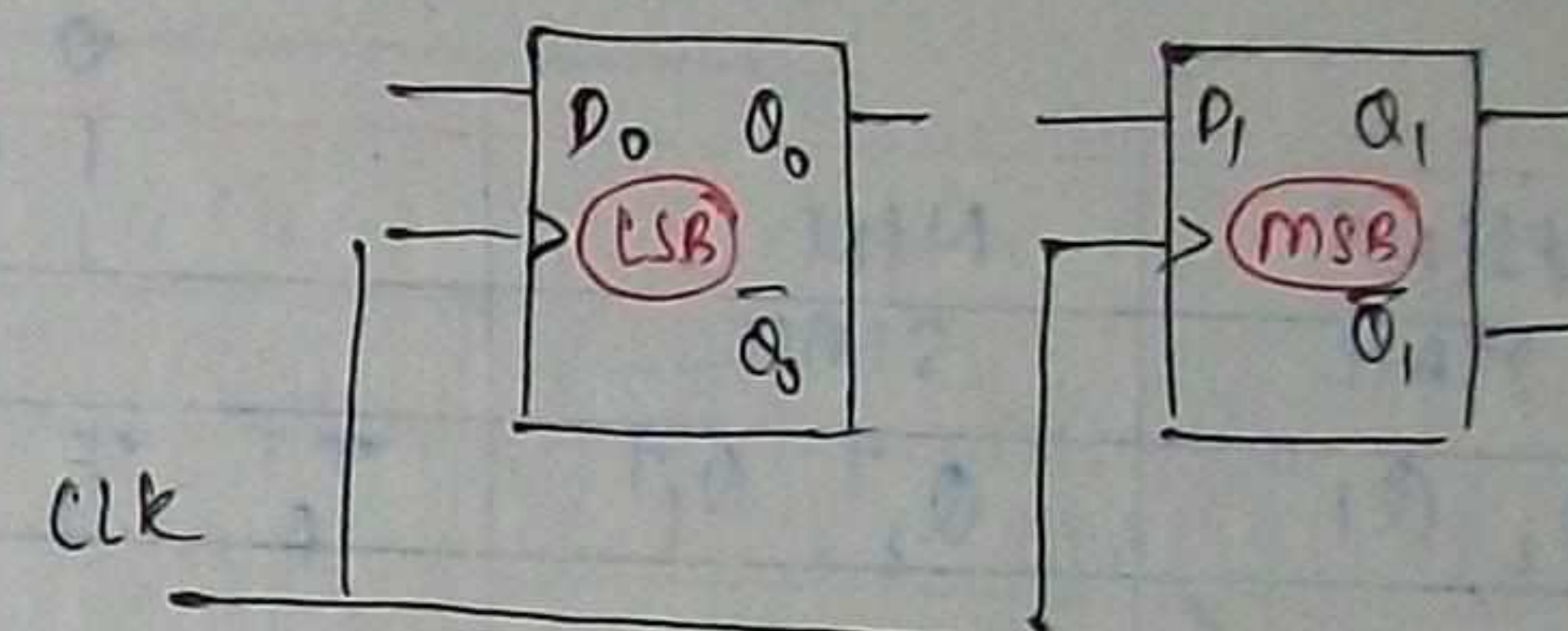
X	X	X	X
0	1	1	1

$$K_2 = Q_0 + Q_2 Q_1$$



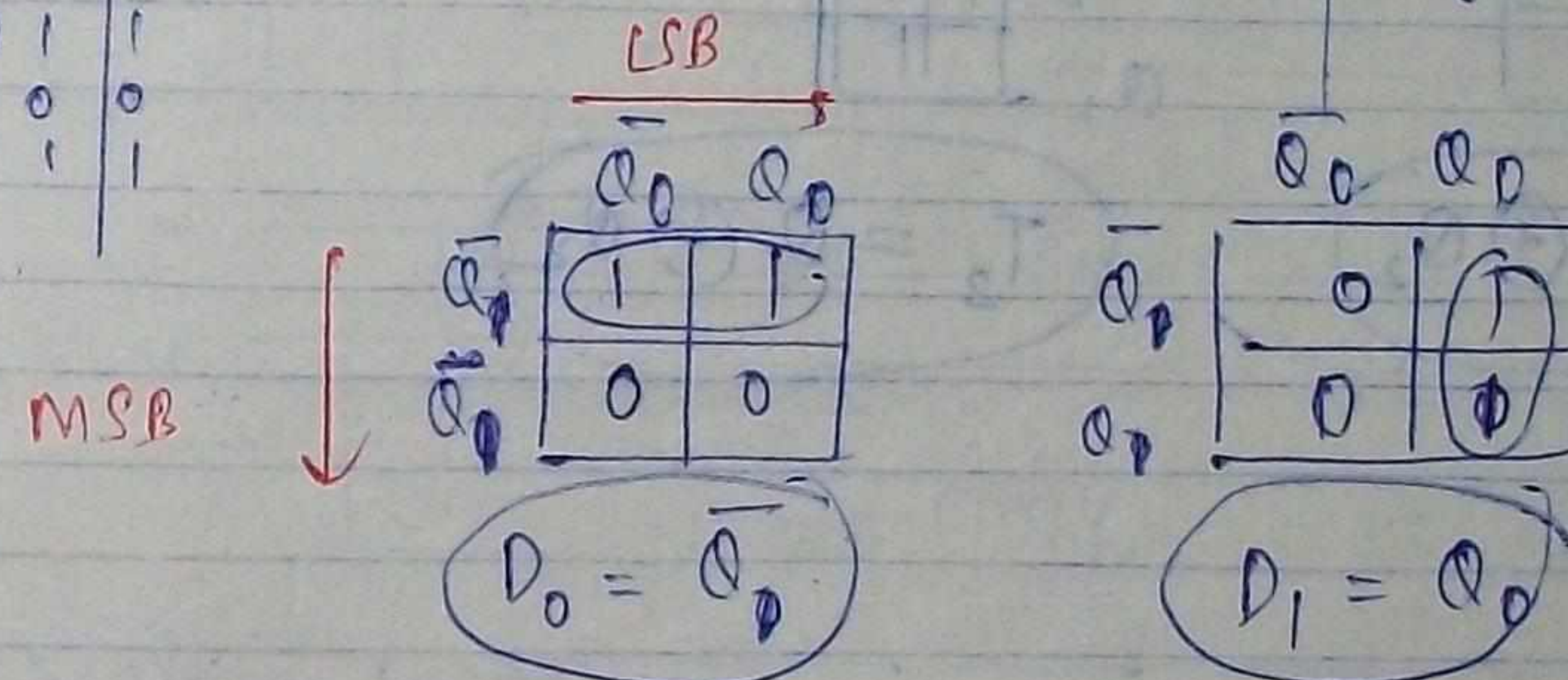
Que. $Q_1 Q_0 \rightarrow 00 \rightarrow 01 \rightarrow 11 \rightarrow 10 \rightarrow 00 \dots$

D_0 and D_1 are respectively

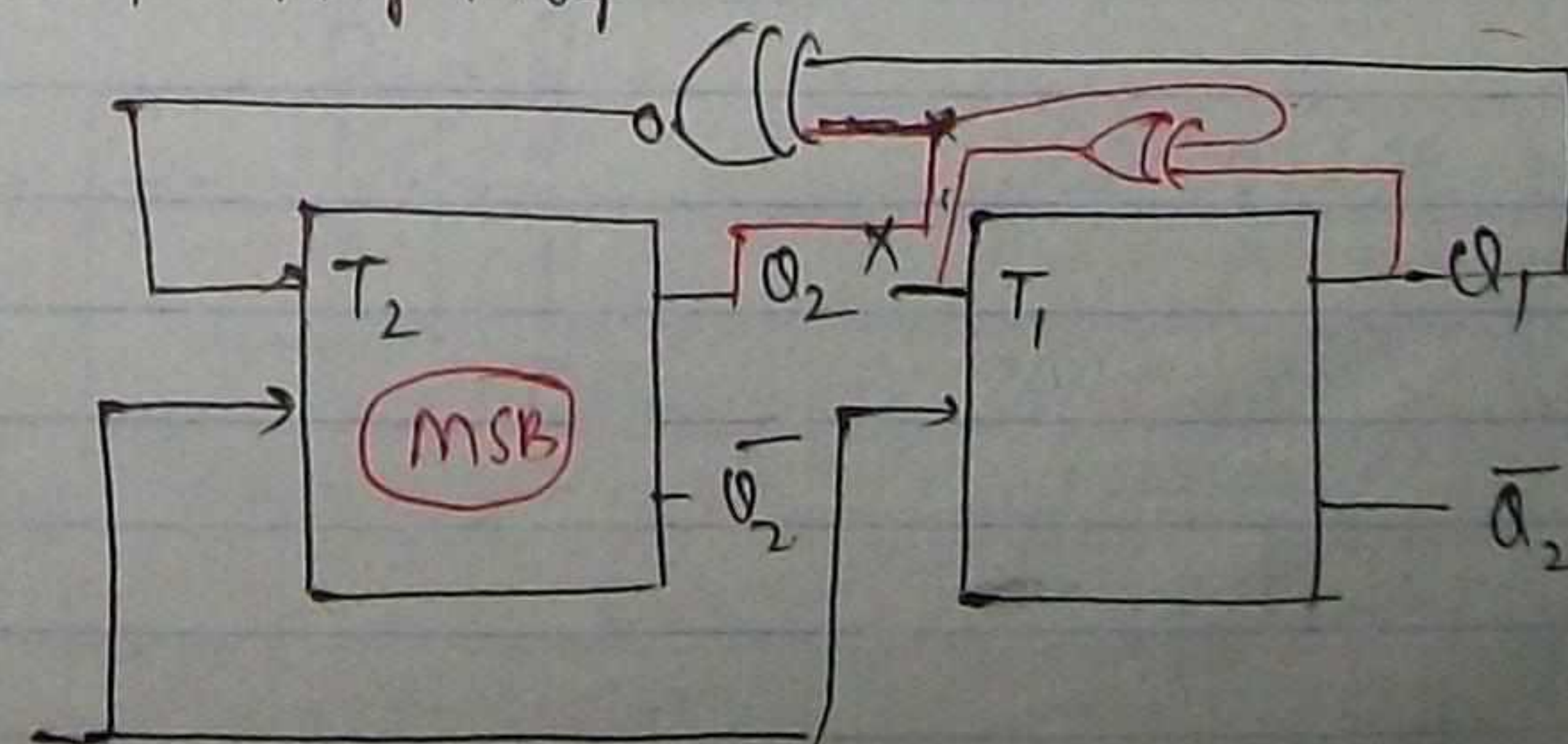


Solⁿ:

PRESENT STATE		NEXT STATE			
Q_1	Q_0	Q_1^+	Q_0^+	D_0	D_1
0	0	0	1	1	0
0	1	1	1	1	1
1	1	1	0	0	1
1	0	0	0	0	0

$$\begin{array}{c} 0010 \\ 0010 \\ 0110 \\ 1010 \\ 1110 \end{array}$$


Que. Sequence 0-2-3-1-0 Design using T-Flip Flop



- X = 0
- 1) Q_2
 - 2) $Q_1 + Q_2$
 - 3) $Q_0 + Q_2$
 - 4) $Q_1 + Q_2$

PRESENT
STATE

Q_2, Q_1

NEXT
STATE

Q_2^+, Q_1^+

T_1, T_2

0	0	0	1	0	0	1
2	1	0	1	1	1	0
3	1	1	0	1	0	1
1	0	1	0	0	1	0

$\bar{Q}_2$	Q_1
1	1

$$T_1 = Q_1 \oplus Q_2$$

Q_2	Q_1
1	1

$$T_2 = Q_1 \odot Q_2$$

COMBINATIONAL CIRCUITS

14/10/11

NO. OF
INPUT

COMBINATIONAL
CIRCUIT

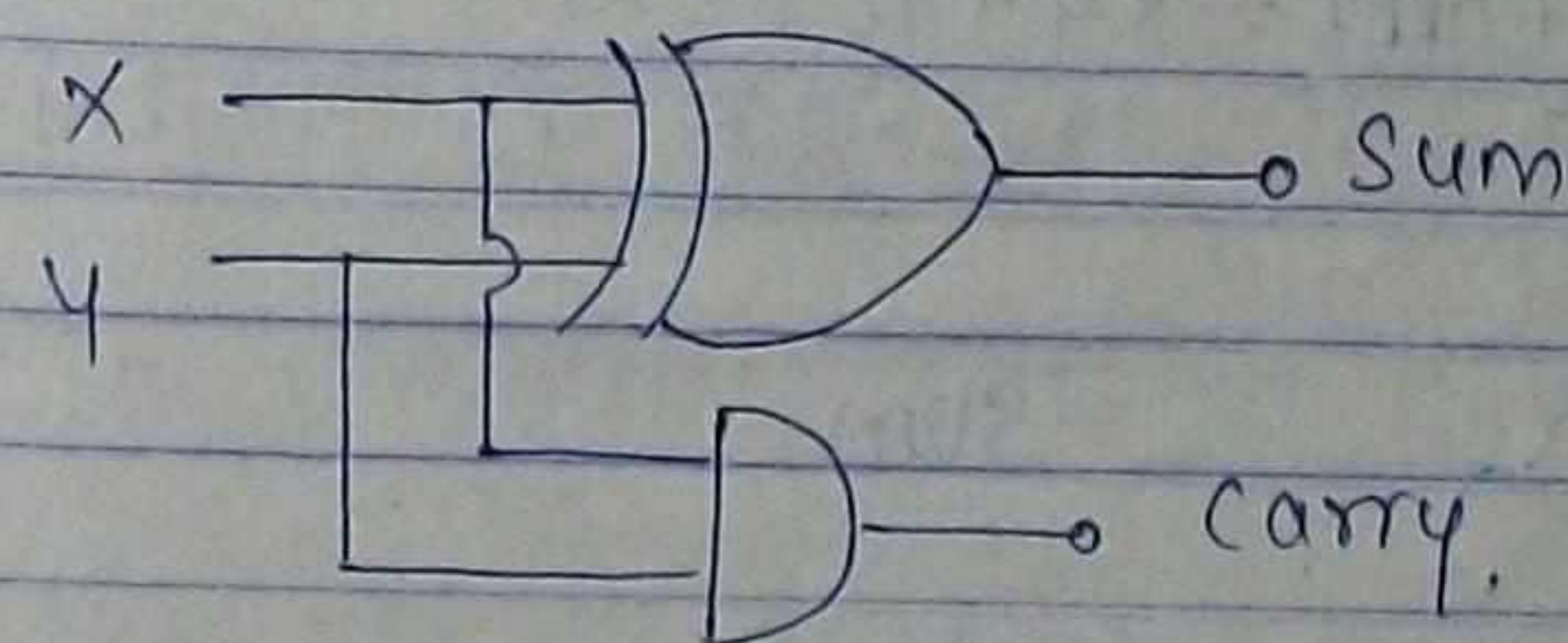
NO. OF
OUTPUT

⇒ HALF ADDER -

X	Y	CARRY	SUM
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = X'Y + XY' = X \oplus Y$$

$$CARRY = X \cdot Y$$

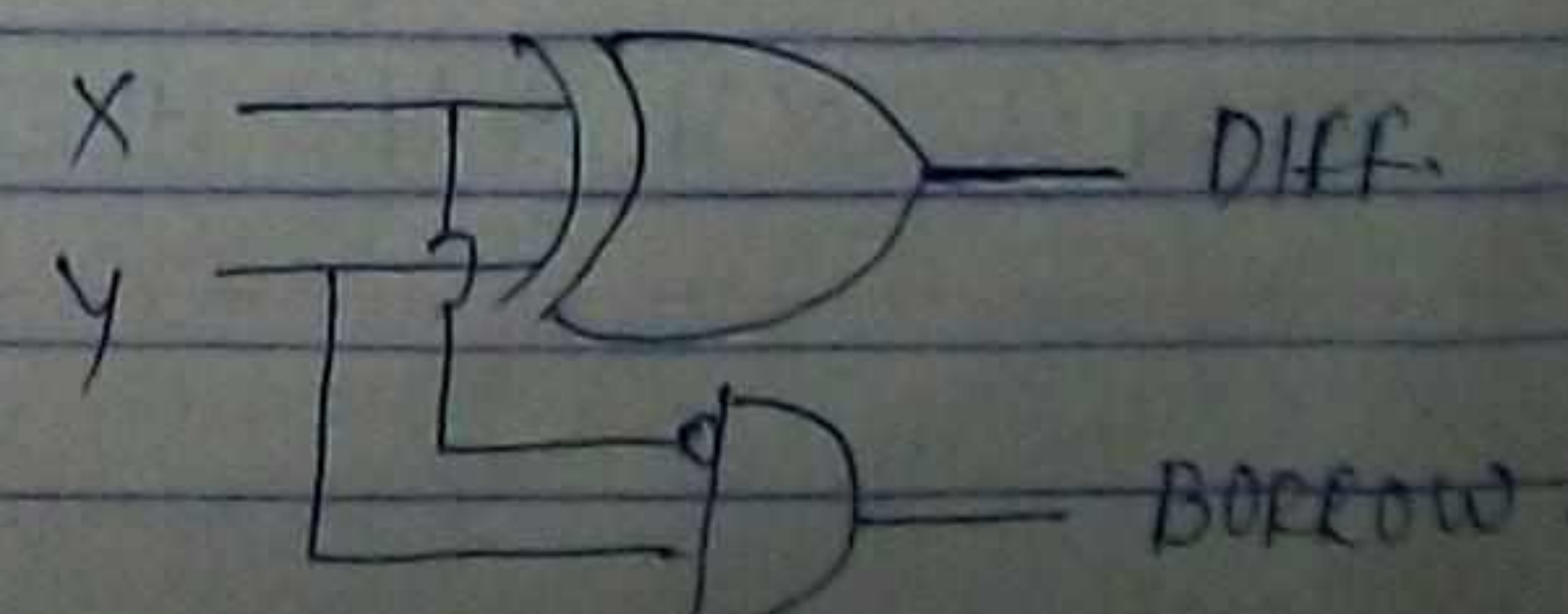


⇒ HALF SUBTRACTOR -

X	Y	BORROW	DIFF
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

$$DIFF = X \oplus Y$$

$$BORROW = X'Y$$



⇒ FULL ADDER -

X	Y	Z	CARRY	SUM
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$\text{SUM} = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$$

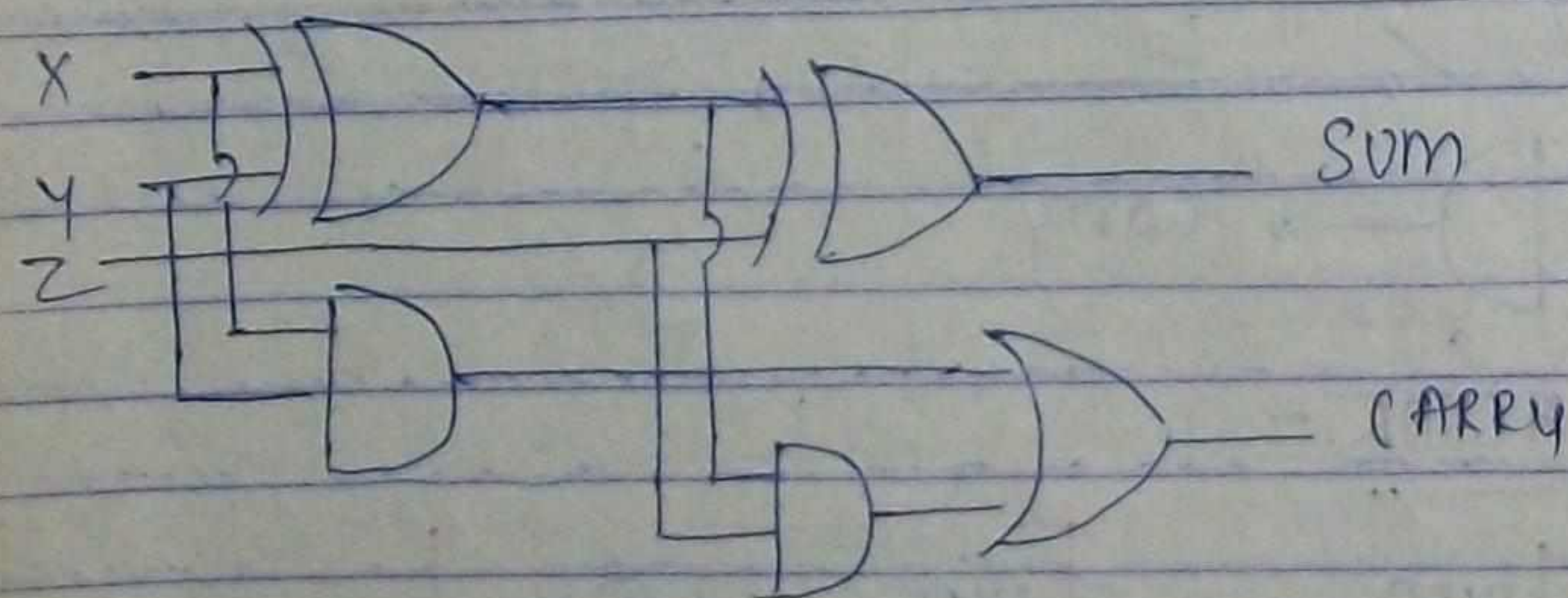
	$\bar{y}\bar{z}$	$\bar{y}z$	$y\bar{z}$	yz
$\bar{x}$		1		1
x	1		1	

$$\text{SUM} = x \oplus y \oplus z$$

$$\text{CARRY} = \bar{x}yz + x\bar{y}z + xy\bar{z} + xyz$$

	$\bar{y}\bar{z}$	$\bar{y}z$	$y\bar{z}$	yz
$\bar{x}$			1	
x		1	1	1

$$\text{CARRY} = xy + yz + zx$$



$$\text{SUM} = x \oplus y \oplus z$$

$$\text{CARRY} = xy + yz + zx$$

⇒ FULL SUBTRACTOR -

X	Y	Z	BORR.	DIFF.
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$\text{DIFF} = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$$

$$\text{DIFF} = x \oplus y \oplus z$$

$$\text{BORROW} = \bar{x}yz + \bar{x}y\bar{z} + \bar{x}y\bar{z} + xyz$$

	$\bar{y}\bar{z}$	$\bar{y}z$	$y\bar{z}$	yz
$\bar{x}$		1	1	1
x			1	

$$\text{BORROW} = x'y + x'z + yz$$

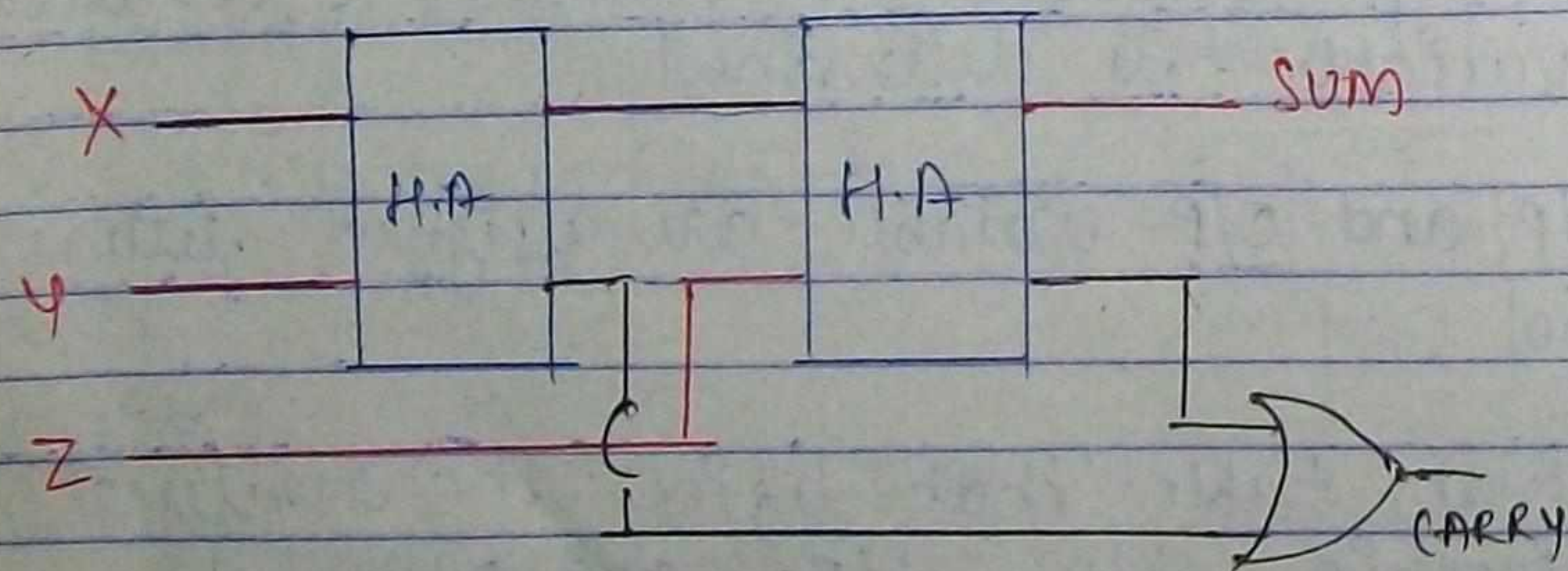
⇒ DESIGNING PROCEDURE OF COMBⁿ CKT -

- ① The problem is stated.
- ② The no. of available I/P variable & required O/P variable is determined.
- ③ The I/P and O/P variable are assigned letter symbol.
- ④ The Truth table that defines the required relationship b/w I/Ps and O/Ps is derived.
- ⑤ The simplified boolean funⁿ for each O/P is obtained.

⑥ The logic diagram is drawn.

⇒ PRECAUTIONS -

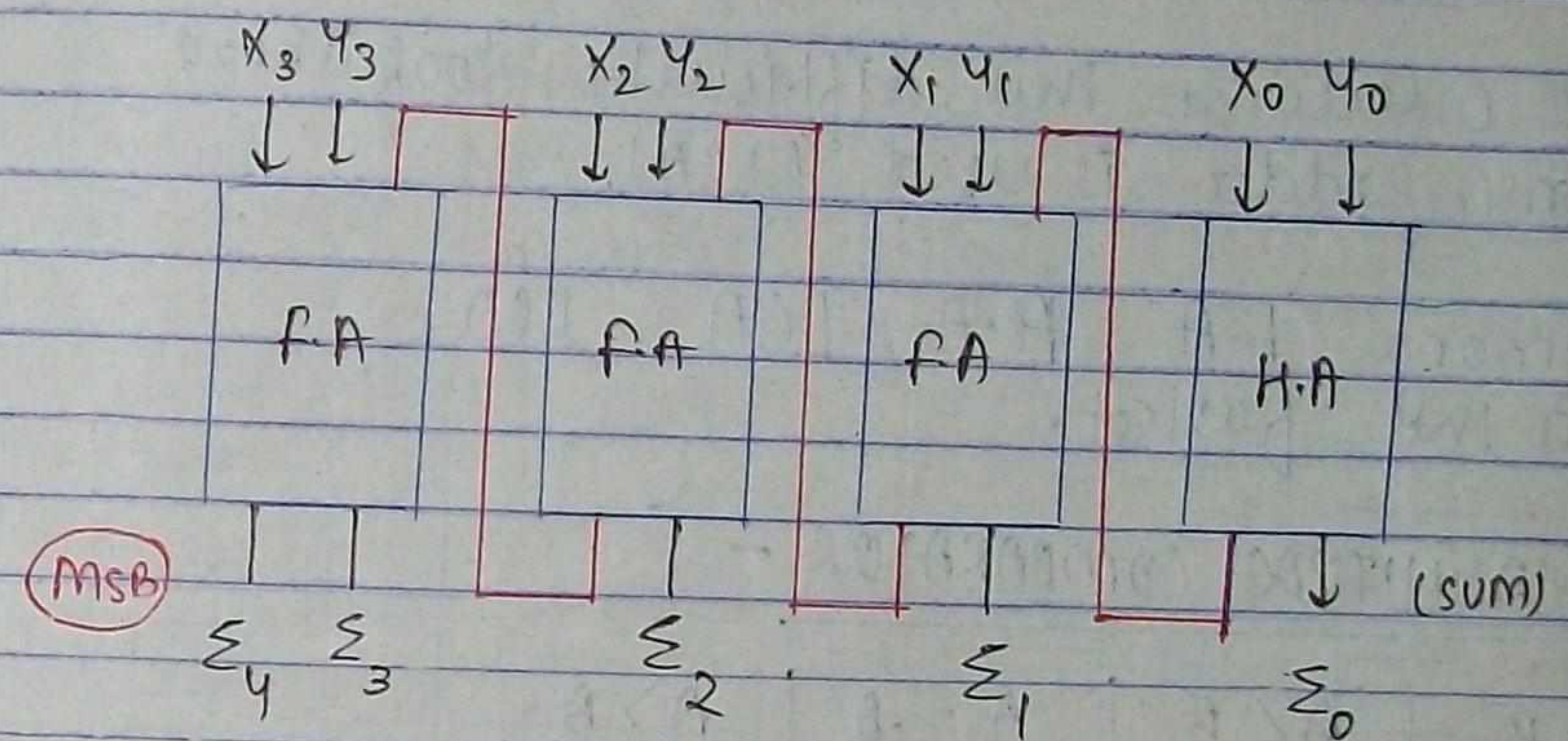
- ① Min. no. of logic gates.
- ② Min. no. of I/Ps to a gate.
- ③ Min. propagation time of the signal through the circuit.
- ④ Min. no. of interconnections.
- ⑤ Limitation of driving capability of each gate.



[FULL ADDER]

⇒ PARALLEL BINARY ADDER -

$$\begin{array}{r} X \rightarrow X_3 X_2 X_1 X_0 \\ Y \rightarrow Y_3 Y_2 Y_1 Y_0 \\ \hline \Sigma_3 \Sigma_2 \Sigma_1 \Sigma_0 \end{array}$$



⑦ To add 'n' No. of column (bits) 'n-1' F.A and 1 H.A required.

Ex - $\begin{array}{r} 1101 \\ 1111 \\ \hline \end{array}$ for the following example we required 2 - H.A and 2 - F.A.

POSSIBILITIES -

1. n - F.A
2. (n-1) F.A + 1 H.A
3. (n-1) { 2 F.A + 1 OR GATE } + 1 H.A

$$\begin{array}{r} X \rightarrow X_3 X_2 X_1 X_0 \\ Y \rightarrow Y_3 Y_2 Y_1 Y_0 \end{array}$$

$$= (2n-1) \text{ H.A } + (n-1) \text{ OR GATE}$$

- If two poi byte data retrieval from the top of stack it goes down word with numerically increasing order of its address

- During the execution of CALL subroutine, address of next instruction will store at the top of stack

- Stack work on the principle of LIFO (last I/P first O/P)

① PUSH Rp

Rp = BC $\neq$ B
DE $\neq$ D
HL $\neq$ H
PSW

(PUS = 1 Byte)

Operation - content of Rp will store at the top of stack.

ex - LXI D 2589 H
LXI SP 5325 H
PUSH D

→ [D] = 25H, [E] = 89H

SP = 5325H

D	E
25	89

89	SP = 5325
25	↑ -2
	SP = 5320

- after the execution of PUSH instruction content of SP will decrease by 2.

add. mode - Register add. mode
m.c - SCW

- Its is data transfer instruction so status of flag will not affect

- There is no conditional PUSH statement (instruction)

② POP Rp

Rp = BC $\neq$ B
DE $\neq$ D
HL $\neq$ H
PSW

(WUS = 1 BYTE)

Operation -

Content two byte data will retrieval from the top of stack and store in Rp

ex - LXI SP 2019 H

POP B

B/C	69	← 2019H
	53	↓ +2
	20	← SP new pop

Add mode → Reg. add. mode

M. Cycle → FRR

Note - after execution of POP instruction content of SP will increase by 2.

- This is data transfer instruction so status of flag will not affect.

- But in case of PSW status of flag may affect indirectly.

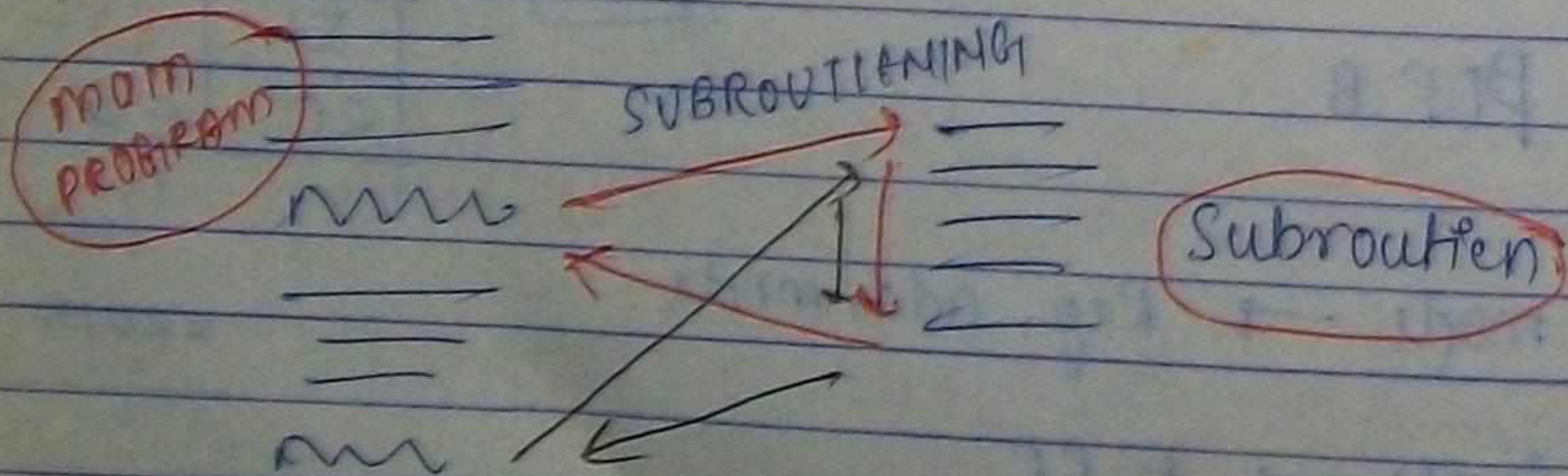
Ex
 main prog
 }
 S=0, Z=0, AC=0
 P=1, CY=1

POP PSW

S Z X AC X P X CF
 [F] = 00 1 1 0 0 0
 [A] [F]
 38H 12H
 SP=2069

⇒ SUBROUTINE -

- It is set of instruction that written separately from main program regarding the task that occur in main program frequently.



- In 8085 there are two instructions from subroutines.

① CALL ② RET (return)

① CALL INSTRUCTION -

(A) unconditional call.

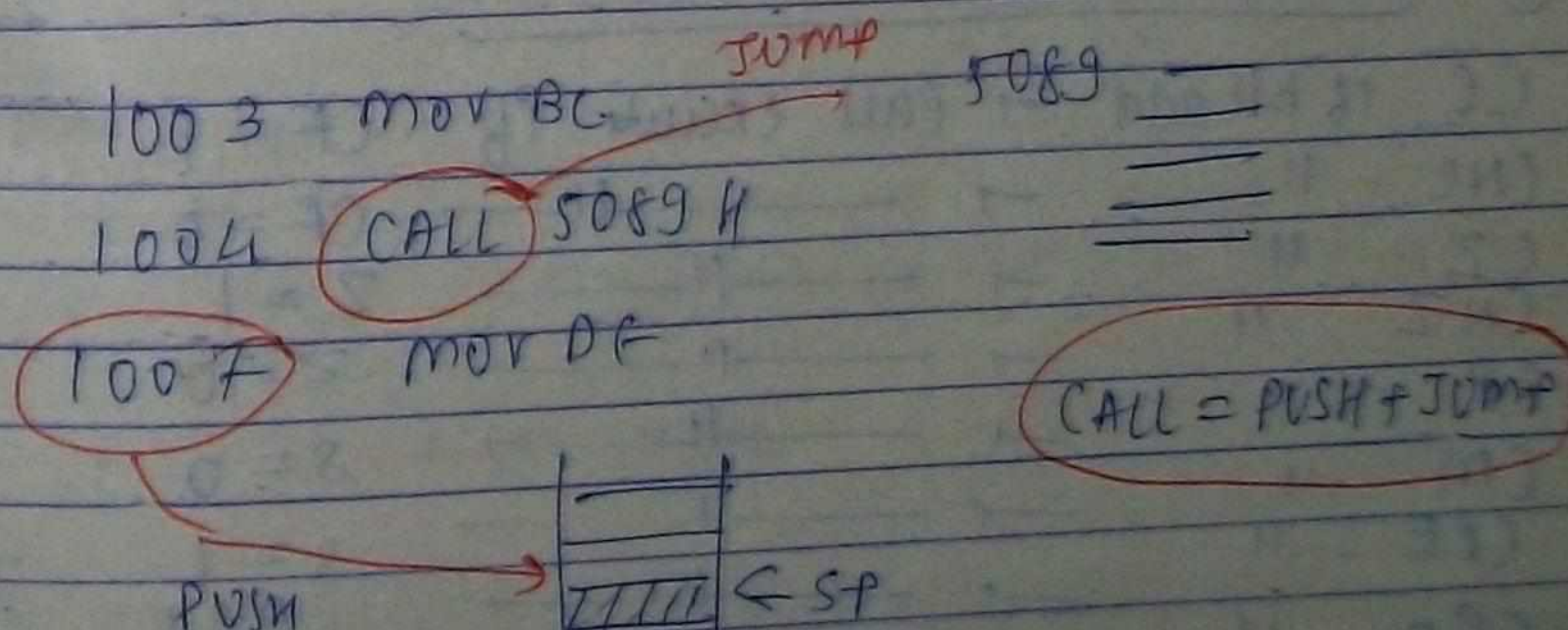
(B) conditional call.

(A) UNCONDITIONAL CALL -

CALL 16 bit add

PSW = 3 BYTE

operation - when this instruction will execute then execution/control will transfer to 16 bit vector location that given in the instruction but before transfer address of next instruction will store at the top of stack.



After execution of call instruction content of SP will decrease by two.

add. mode $\rightarrow$ Immediate add.
 mc $\rightarrow$ SRRWW (18T)

NOTE - If it is data transfer so instruction so status of flag will not affect.

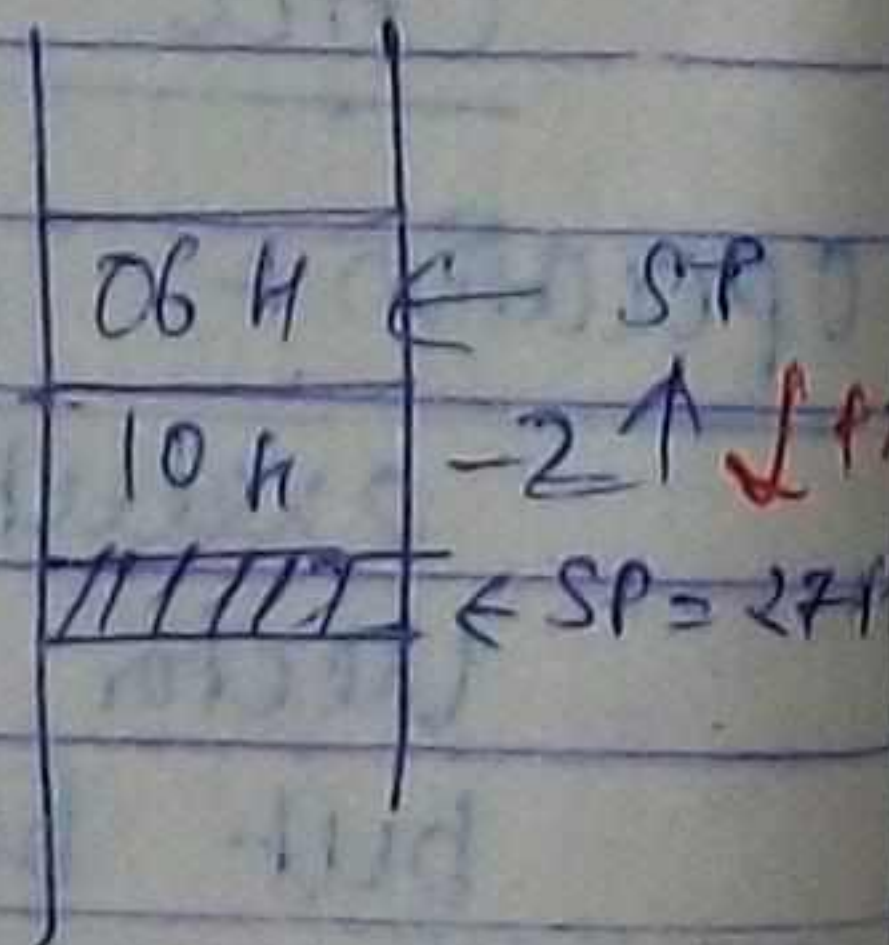
Ques 1000 : LXI SP 27FFH
 1003 CALL 1006
 1006 POP HL

after execution of POP instruction content of SP and HL pairs are

Solⁿs LXI SP 27FF

SP = 27FF

HL = 1006



(B) CONDITIONAL CALL INSTRUCTION -

CC 16 bit add $\rightarrow$ call execute if CF = 1

CNC " $\rightarrow$ " " CF = 0

CZ " $\rightarrow$ " " Z = 1

CNZ " $\rightarrow$ " " Z = 0

CP " $\rightarrow$ " " S = 0

CM " $\rightarrow$ " " S = 1

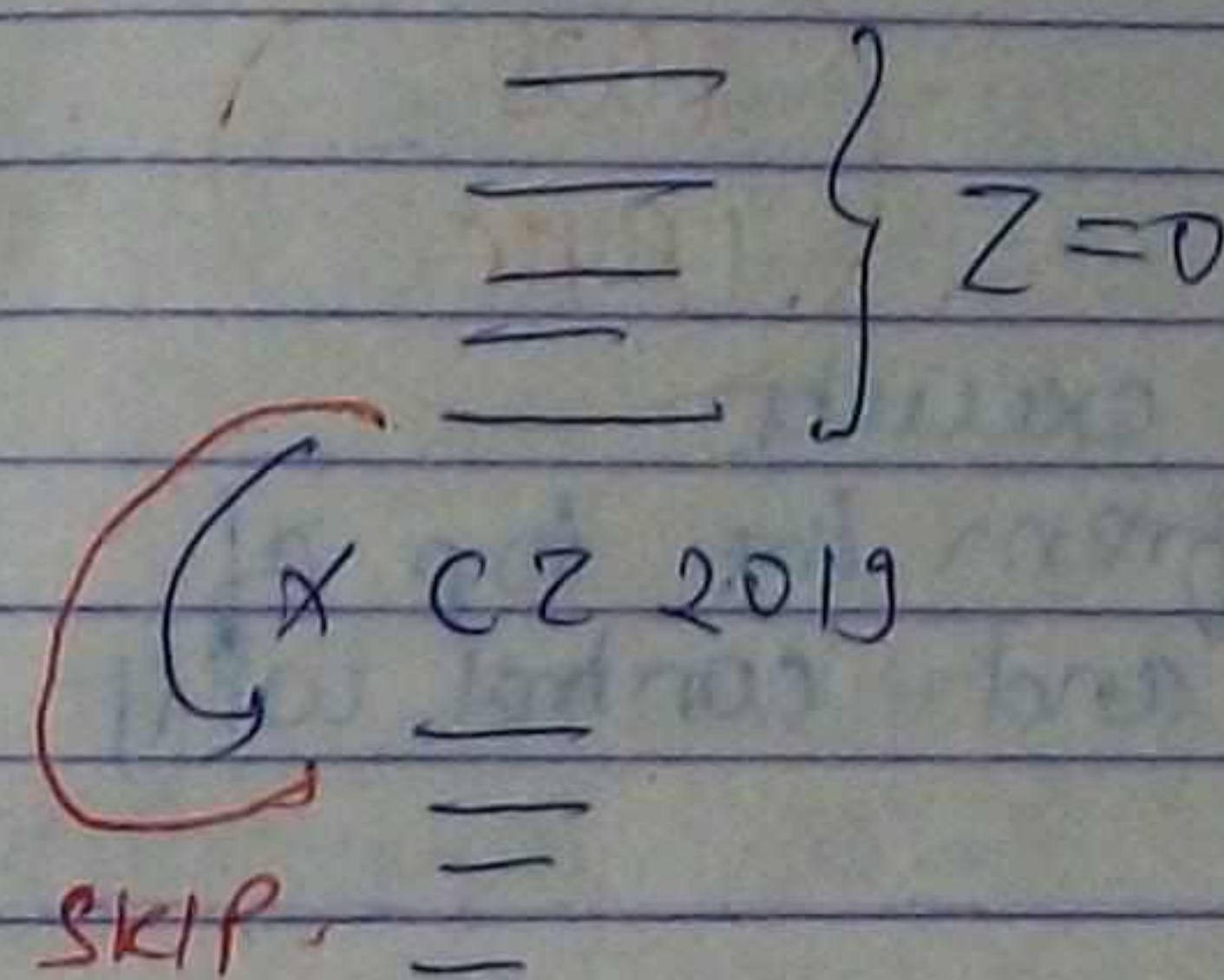
CPE " $\rightarrow$ " " P = 1

CPO " $\rightarrow$ " " P = 0

Operation -

18T = 3 B/H

If condition satisfy call instruction will execute otherwise skip it.



Add. mode $\rightarrow$ Immediate add.

M.C $\rightarrow$ Satisfy = SRRWW
 Not Satisfy = SR

NOTE -

- Call instruction is data transfer instruction so status of flag will not affect.

- Call instruction (satisfy case or unconditional) is the largest instruction of 8085 = 18T

$\Rightarrow$ RET INST -

- It is last instruction of every subroutine.

- It is of two type

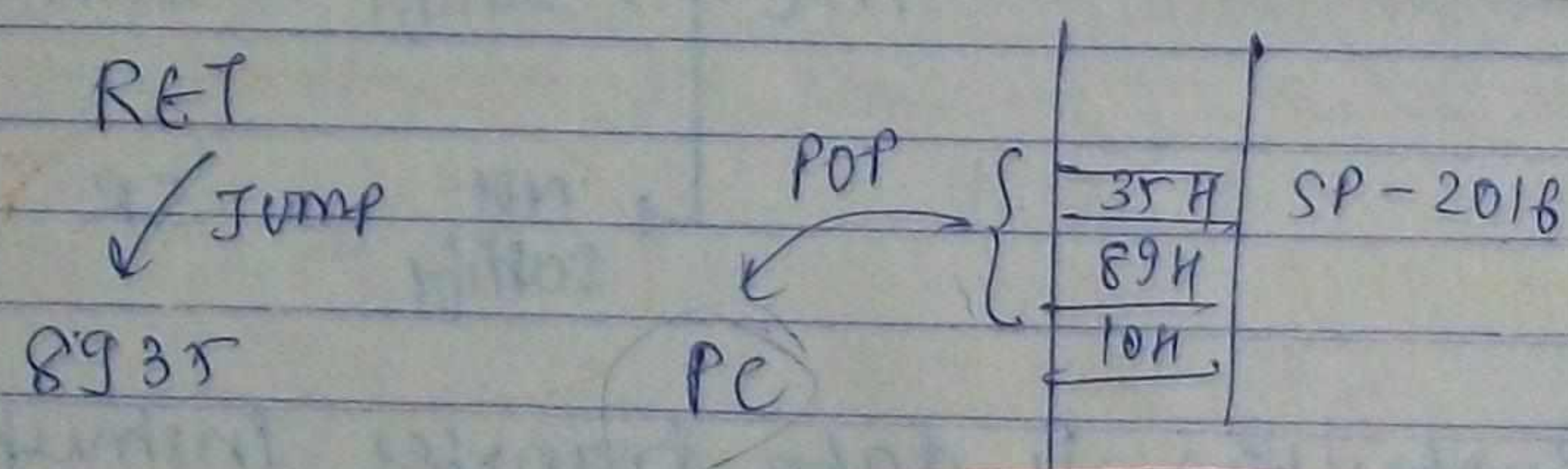
- ① unconditional return.
- ② conditional return.

① unconditional RET inst.

RET no operand

operation -

When this instruction executes two byte data retrieved from the top of stack (and store in PC) and control will jump on that.



RET = POP + JUMP

After the execution of RET instruction, content of SP will increase by two.

Add. mode
 Indirect Reg add. (✓)
 Implicit add.

machine cycle - 3RR

Ex - LXI SP 2719H
 CALL 3000H

(X41) → 3000H LXI H 1019H
 → LXI D 2015H
 → PUSH D
 → PUSH H
 → LXI SP 3C42H
 after execution of Return
 content of SP = ?
 POP PSW + 1
 POP H + 2
 POP D + 2
 RET + 2

sd 3 SHORT TRICK

3C42
 + 8
 3C4A

2nd method

[HL] = 1019

[DE] = 2015

SP = 3C42H

[AP] → 3C42H
 → SP+2
 HL { → SP+4
 DE { → SP+6
 PC { → SP+8

[P] ← SP-8
 [A] ← SP-6
 19 ← SP-4
 10 ← SP-2
 15
 20
 4
 X
 2719H
 SP

3C42
 8
 3C4AH

main prog

Ques

1002: MOV Bc

1003: CALL 5023H

(1006) MOV DE

SUBROUTINE

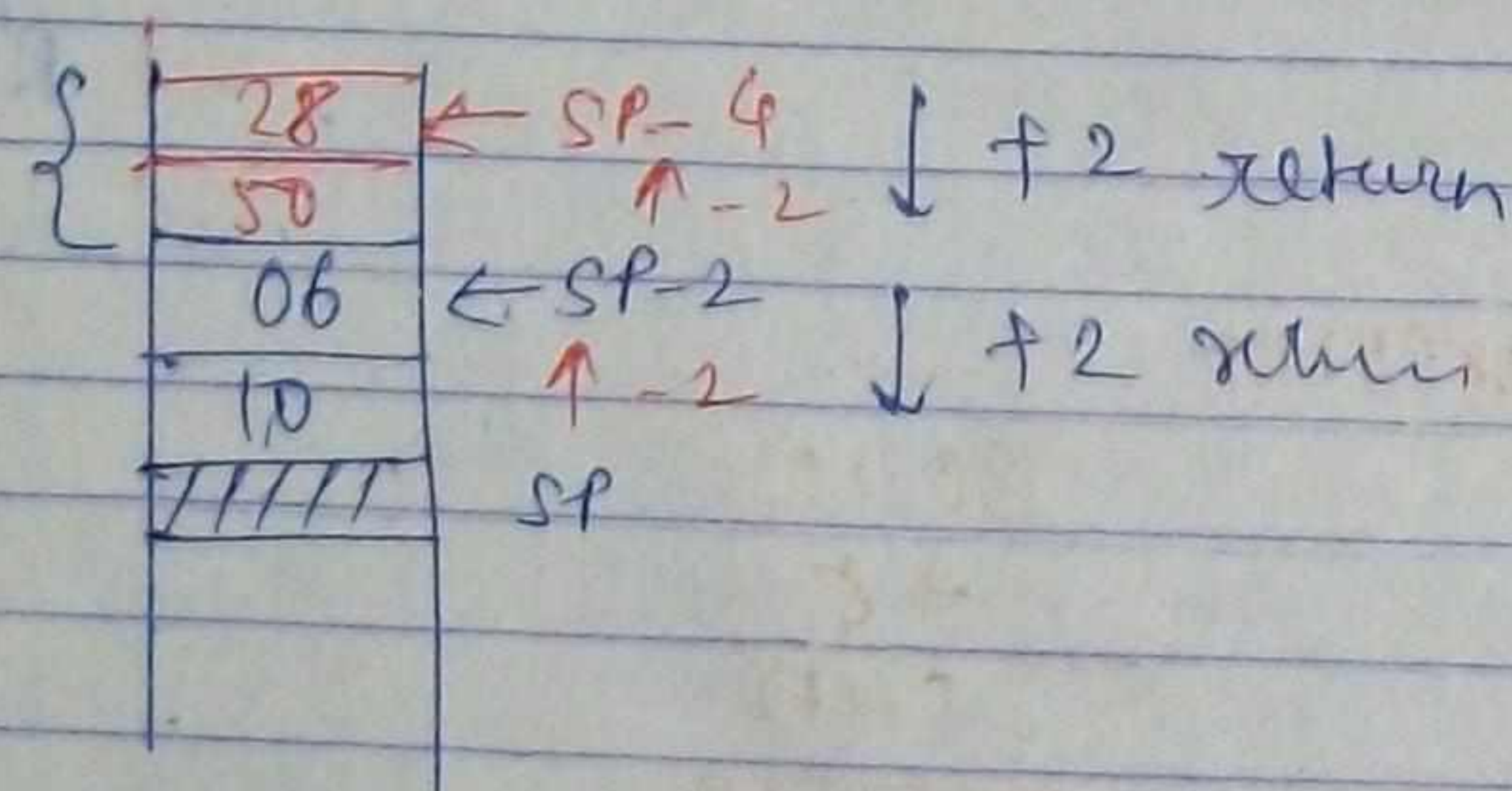
5023 MYI A, 00H

5025 CALL 5028

(5028) INRA

RET

After execution of subroutine when control return in main program. value of Accumulator



jump

[A] → 00H

[A] = 01H

Return

PC = 5028

Return

[A] = 02H

PC = 1006

∴ [A] = 02H

② CONDITIONAL RET

RC	NO operand	→	RET will execute if CF = 1
RNC	"	→	if CF = 0
RZ	"	→	Z = 1
PNZ	"	→	Z = 0
RP	"	→	S = 0
PM	"	→	S = 1
RPE	"	→	P = 1
RPO	"	→	P = 0

Operation-

If condition satisfy then RET instruction will execute otherwise skip it.

1 WORD
1 BYTE

Add. mode → Indirect Reg. add mod. ✓

→ implicit add mod.

Machine cycle → Satisfy → SRR
Not satisfy → S/F

⇒ RESTART INST

① RST N nooperand -

operation -

when this instruction will execute, control will transfer at a specific vector location of memory page no. 00H.

$N = 0, 1, 2, 3, 4, 5, 6, 7$

NOTE

- Restart instruction is used when execution trap in interrupts.

- It is just like one byte CALL INST.

- It is a software interrupt.

INSTRUCTION

RST N

VECTOR LOCATION

00XX

RST 0	→	00 00 H
RST 1	→	00 08 H
RST 2	→	00 10 H
RST 3	→	00 18 H
RST 4	→	00 20 H
RST 5	→	00 28 H
RST 6	→	00 30 H
RST 7	→	00 38 H

NOTE - XX is the hexadecimal conversion of (N x 8)

$$0 \times 8 = 00 H$$

$$1 \times 8 = 08 H$$

$$2 \times 8 = 16 = 10 H$$

Add. mode → Implicated add. mode

M.C → S

- This is data transfer instruction so status of flag will not affect.

⇒ SOME ADVANCE ARITHMATIC INSTRUCTION

① ADCR

1WS
= 1 BYTE

$R = A, B, C, D, E, H, L \& M$

Operation -

when this instruction will execute content of R will added in content of accumulator with status of carry flag & result will store in Accumulator

$$[A] \leftarrow [A] + [R] + CF_{\text{flag}}$$

Ex [A] = 10H
[B] = 3H
CF = 1
ADC B

Sol
[A] = 00010000
[B] = 00111000
CF = 1

[A] 01001100
S Z AC P CY (according to address)
0 0 0 1 0

add. mode $\begin{cases} R \neq m & \text{Register add. mode} \\ R = m & \text{Indirect add. mode} \end{cases}$

M.C $\begin{cases} R \neq m \rightarrow F \\ R = m \rightarrow FR \end{cases}$

② ACI 8 bit data -

Operation -

8 bit data will get added in [A] with status of carry flag and result will store in [A]

$[A] \leftarrow [A] + 8 \text{ bit data} + CF_{LSB}$

Add mode $\rightarrow$ Immediate add. mode
MC $\rightarrow$ FR

1WS = 2 Byte

③ SBB R -

1WS = 1 Byte

R = A B C D E H L R M

Operation -

$[A] \leftarrow [A] - [R] - CF_{LSB}$ (M.D)

add mode $\begin{cases} R \neq m & \text{register add mode} \rightarrow F \\ R = m & \text{Indirect add. mode} \rightarrow FR \end{cases}$

④ SBI 8 bit data -

1WS = 2 Byte

Operation -

$[A] \leftarrow [A] - 8 \text{ bit data} - CF_{LSB}$

add. mode $\rightarrow$ Immediate add mode. } M.C = FR

Note -

All above four instructions are arithmetic instructions so status of all flag will get affect.

⇒ INSTRUCTION RELATED TO HL PAIR -

① LHLD 16 bit add.

1WS = 3 BYTE

Load HL pair direct

Operation - when this instruction will execute 8 bit data available at memory location that is given in instruction will load in L and data available at next memory location (add+1) will store in H.

$L \leftarrow [16 \text{ bit add}]$

$H \leftarrow [16 \text{ bit add} + 1]$

Ex LHLD 2015

$[L] \leftarrow [2015]$

$[H] \leftarrow [2015 + 1]$

$[L] = 1CH$

$[H] = 25H$

add. mode - direct add. mode
Machine code - FRRRR

1CH 2015

25H

② SHLD 16 bit add.

1WS = 3 BYTE

Store HL pair direct

Operation - Content of L will store at 16 bit memory location that given in instruction and content of H will store at next memory location (16 bit add + 1)

$[L] \rightarrow [16 \text{ bit add}]$

$[H] \rightarrow [16 \text{ bit add} + 1]$

Add mode - direct add
M.C → FRRWW

Ex 1001 SHLD

1001 SHLD

1002 15

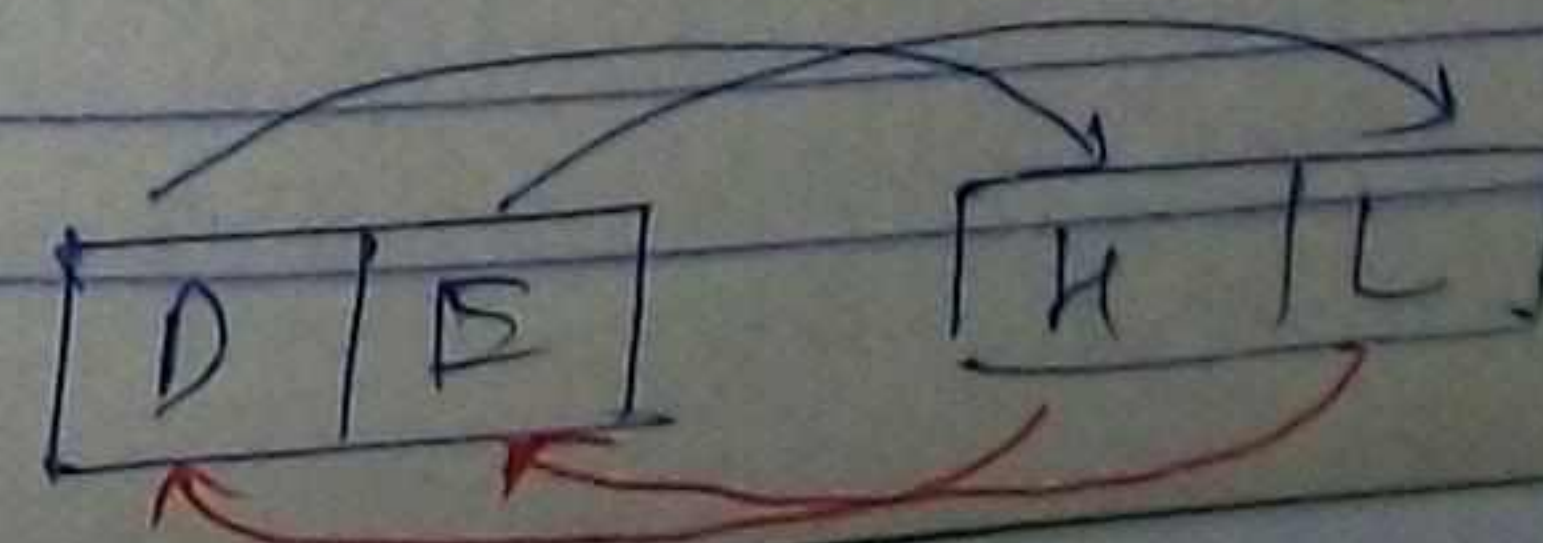
1003 20

15 2015
20 2016

③ XCHG no operand

1WS = 1 Byte

Operation - Content of HL pair will exchange from content of DE pair.



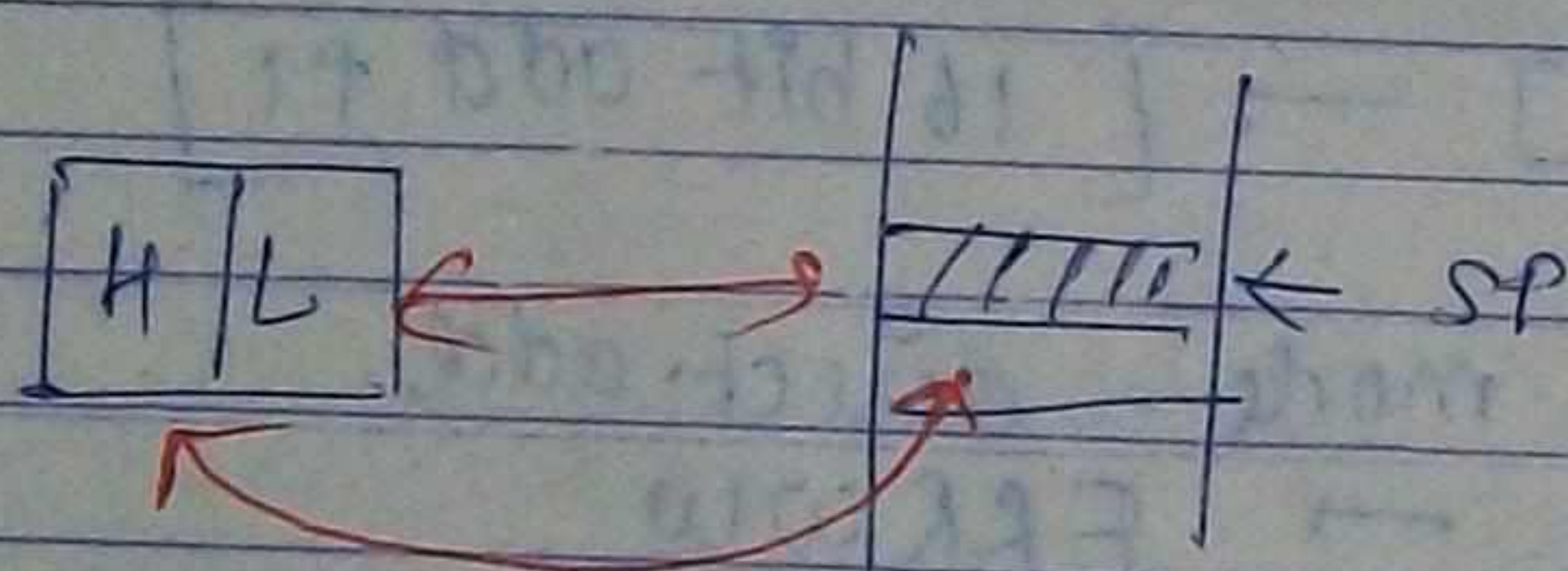
add. mode — Implicit add. mode ✓
 Register add. mode.

M.C. → F

④ XTHL no operand —

operation —

content of HL pair will exchange from the top of stack.



NOTE —

for the operation intermediate storage one memory location.

XTHL = PUSH + POP

M. cycle → FWWRR / FRRWW

Add. mode — Implicit add. ✓
 — Register add
 — Indirect add

⑤ SPHL no operand —

1WS = 1 Byte

operation — content of HL pair will copy in SP

$[SP] \leftarrow [HL]$

Note — It is indirect method to initialize a stack in main memory.

Add. mode — Implicit add mode ✓
 Register add. mode
 M.C. → S

⑥ PCHL no operand —

1WS = 1 Byte

operation —

content of HL pair will copy in PC.

$[PC] \leftarrow [HL]$

Ques. 1003: LXI H 2015
 1006 MOV B C
 1007 MVI A 92H
 1009 PCHL

after execution of PCHL next instruction will fetch from

PCHL $[PC] \leftarrow [HL]$
 $[PC] = 2015H$

add. mode — Implicit add mode
 — Register add. mode

M. Cycle $\rightarrow$ 5

NOTE —

All above instructions are data transfer instructions so status of flag will not affect

$\Rightarrow$ SOME ADVANCE INSTRUCTION —

① STC no. operand —

operation —

After the execution of this instruction carry flag get set regardless of previous status.

CF = 1

Add. mode — Implicit add. mode
 M.C $\rightarrow$ F

② CMC no operand —

operation — when this instruction will execute ~~stat~~ carry flag get complemented

CF $\leftarrow$ $\overline{\text{CF}}$

NOTE — STC & CMC will affect only 1 flag i.e. carry flag.

add. mode — Implicit add. mode
 M.C $\rightarrow$ F

③ DAA (decimal adjustment of accumulator)

DAA no operand

1WS = 1 BYTE

operation —

when this instruction will execute then content of accumulation adjust in BCD format by assuming earlier operation was BCD addition.

— If lower nibble of [A] is greater than 1001 (9) then 0110 (6) will get added in it.

— If lower nibble of [A] is equal or less than 1001 but AC flag is set, then 0110 will get added in it.

— If upper nibble of [A] is greater than 1001, 0110 will get added in it.

— If upper nibble of [A] is equal or less than 1001 but carry flag is set then 0110 will get added in it.

— Otherwise cannot add.

add. mode — implicit add. mode
mc — F

NOTE — DAA instruction affect status of all flags.

Ex —
6010H : LXI H 8A79H

MOV A, L

ADD H

DAA

MOV H, A

PCHL

after the execution of PCHL instruction next instruction will fetch from,

- ① 6019H
- ② 0379H
- ③ 6979H
- ④ None of them.

Solⁿ:

[H] = 8A

[L] = 79

[A] = 79

[A] = 79 + 8A

DAA

1000 1010

0111 1001

0111 1001

1000 1010

0000 0011

0110 0110

0110 1001

AC = 1

CF = 1

S Z AC P CF

due to A = 0 0 1 1 1

due to DAA = 0 0 0 1 0

[A] = 69H

[H] = 69H

PC ← 6979H

Ques. MVI B 28H

loop : NOP

DCRB

JNZ loop

HIT

If T = 0.5T then find total time elapsed (required for execution).

Solⁿ: (28)₁₆ = (40)₁₀

B ← 28H

40 time

(Z = 0)

satisfy,

39

satisfy

not satisfy.

Total T = 7T + 39 × 18T + 1 × 15T + 4T
= 728T

Ques. MVI B, 38H 7T operating freq = 2 MHz

loop2 MVI C, FFH 7T total

loop1. DCR C 4T time elapsed = ?

JNZ loop1 10T/7T ~~2~~ = 0

DCR B 4T

JNZ loop2 10T/7T

Solⁿ: $(38)_{16} = (58)_{10}$

$(FF)_{16} = (255)_{10}$

loop1 $\rightarrow 255 + 255 \times 8$

total T-state req. for completely execution of inner loop

$$= 255(14T) + 1(11T)$$

$$= 3567T$$

total T state = $7T + 55[7T + 3567T + 4T + 10T]$
for program

$$1[7T + 3567T + 4T + 7T]$$

$$= 20093T$$

~~(20093T)~~

	IO/ $\overline{m}$	$\overline{RD}$	$\overline{WR}$	S_1	S_0
FETCH m-cycle (F/S)	0	0	1	1	1
Memory m. read cycle (R)	0	0	1	1	0
Memory write (W) m. cycle	0	1	0	0	1
I/P read m. cycle (I)	1	0	1	1	0
O/P write (O) m. cycle	1	1	0	0	1

$S_1 S_0$

1 1 = opcode fetch

1 0 = Read operation

0 1 = write operation

if no logic on $S_1 S_0$ = Bus idle m. cycle

Ques

LXI H 2041H FRR = 10T ✓
MVI A, D6H FR = 7T ✓
CMP M FR = 7T ✓
MOV B, A F = 4T ✓
IN 20H FRI = 10T ✓
SUB B F = 4T ✓
OUT 52H PRO = 10T ✓
INX H (S) = 6T ✓
MOV MA FW = 7T ✓
HLT F = 4T ✓

- ① How many times MP execute memory read m. cycle.
- ② How many times MP perform memory read operation or data read from memory.
- ③ How many times MP perform read operation or how many times read signal will go active low.
- ④ Total T-states in program
- ⑤ Total time consume if $1T = \frac{1}{3} \mu\text{sec}$.

Soln: ① 6 times
② 16 times (including opcodes)
③ 17 times
④ 69 T
⑤ 23 μsec .

RSTART - Software Interrupt

⇒ INTERRUPTS -

In 8085, five hardware interrupts are available

TRAP → non-maskable
RST 7.5
RST 6.5
RST 5.5
INTR } maskable

On the basis of different characteristics interrupt can be classified in different group.

① MASKABLE & NON-MASKABLE -

- Interrupt that can make disable = maskable interrupt

- Interrupt that cannot make disable = non-maskable interrupt

NOTE -

- Trap is also known as RST 4.5

- To control interrupt process in 8085 there is a interrupt enable pins by set or reset we can control interrupt process.

- To set or reset of interrupt enable pin in 8085 two instructions are defined



① EI no operand

1WS = 1 BYTE

Operation -

When this instruction will execute
Interrupt enable FF will get set and
Interrupt processor will enable.

add. mode - Implicit add. mode,
M. cycle - F.

② DI no. operand

1WS = 1 BYTE

Operation -

When this instruction will execute,
Interrupt enable FFs will get reset and
Interrupt processor will be disabled.

Add. mode - Implicit add. mode,
M. cycle - F.

NOTE - EI and DI is the machine control
instruction so status of flag will not
affect.

- Trap is independent of EI and DI

- Masking concept valid only ~~when~~ in the
interrupt enable (EI) processor condition.

② VECTORED & NON VECTORED INTERRUPT

- If a interrupt is acknowledge & control transfer
(occurs)
at fixed vector location then it is known as
vectored interrupt.

- If vector location is not defined then it is
known as non vectored interrupt.

NOTE

- Vector location to non vector interrupt provide
externally.

TRAP

RST 7.5

RST 6.5

RST 5.5

INTR

Vector
Interrupt

→ non vector interrupt

- In case of vector interrupt control unit transfer
at specific memory location of memory page
no. 00H

Vectored interrupt

vector location

TRAP

RST 7.5

RST 6.5

RST 5.5

00 2 4 H

00 3 C H

00 3 4 H

00 2 C H

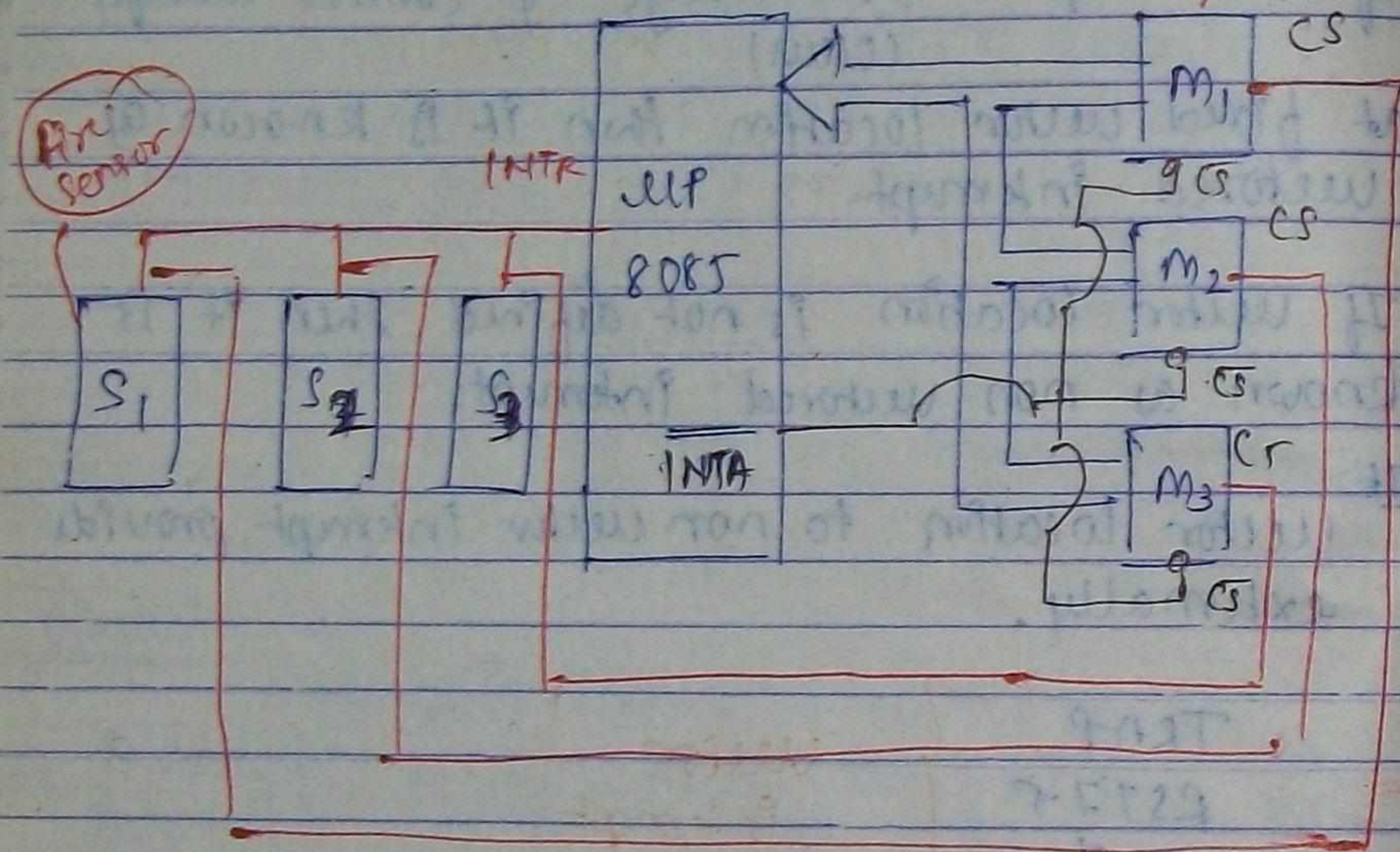
$4 \times 8 = 32 = 24H$

$7 \times 8 = 56 = 3CH$

$6 \times 8 = 48 = 30H$

$5 \times 8 = 40 = 28H$

vector location for Non vectored Interrupt -



$\left. \begin{matrix} S_1 \\ S_2 \\ S_3 \end{matrix} \right\} \text{Sensor}$
 $\left. \begin{matrix} M_1 \\ M_2 \\ M_3 \end{matrix} \right\} \text{memory}$

③ TRIGGERING INTERRUPT -

both level and edge $\leftarrow$ TRAP } edge trigger
 RST 7.5
 RST 6.5 }
 RST 5.5 } level trigger
 INTR

Step regarding for sensor

- trap is with both level & edge trigger

④ PRIORITY -

TRAP
 RST 7.5
 RST 6.5
 RST 5.5
 INTR
 HIGHEST
 LOWEST

Trap has highest
INTR has lowest

⑤ In the externally initiated signal HOLD pin pin has the highest priority

NOTE -

- when an interrupt is acknowledge then following steps will execute automatically.

- ① Execution of current instruction will complete first
- ② address of next instruction will store at the top of stack
- ③ DI instruction will execute automatically
- ④ Execution will transfer at interrupt service routine

NOTE - Programmer should write last instruction of interrupt service routine **EI**.

For
- minimum time duration of interrupt should acknowledge for definite execution is **18T**
= 17.5T (book)

⇒ **SIM (SET INTERRUPT MASK)** -

- This interrupt is used for
① for the masking of interrupt
② Reset of RST 7.5
③ Serial transfer of data through SOD pin

- SIM instruction executes on the basis of content of accumulator.

SIM no operand

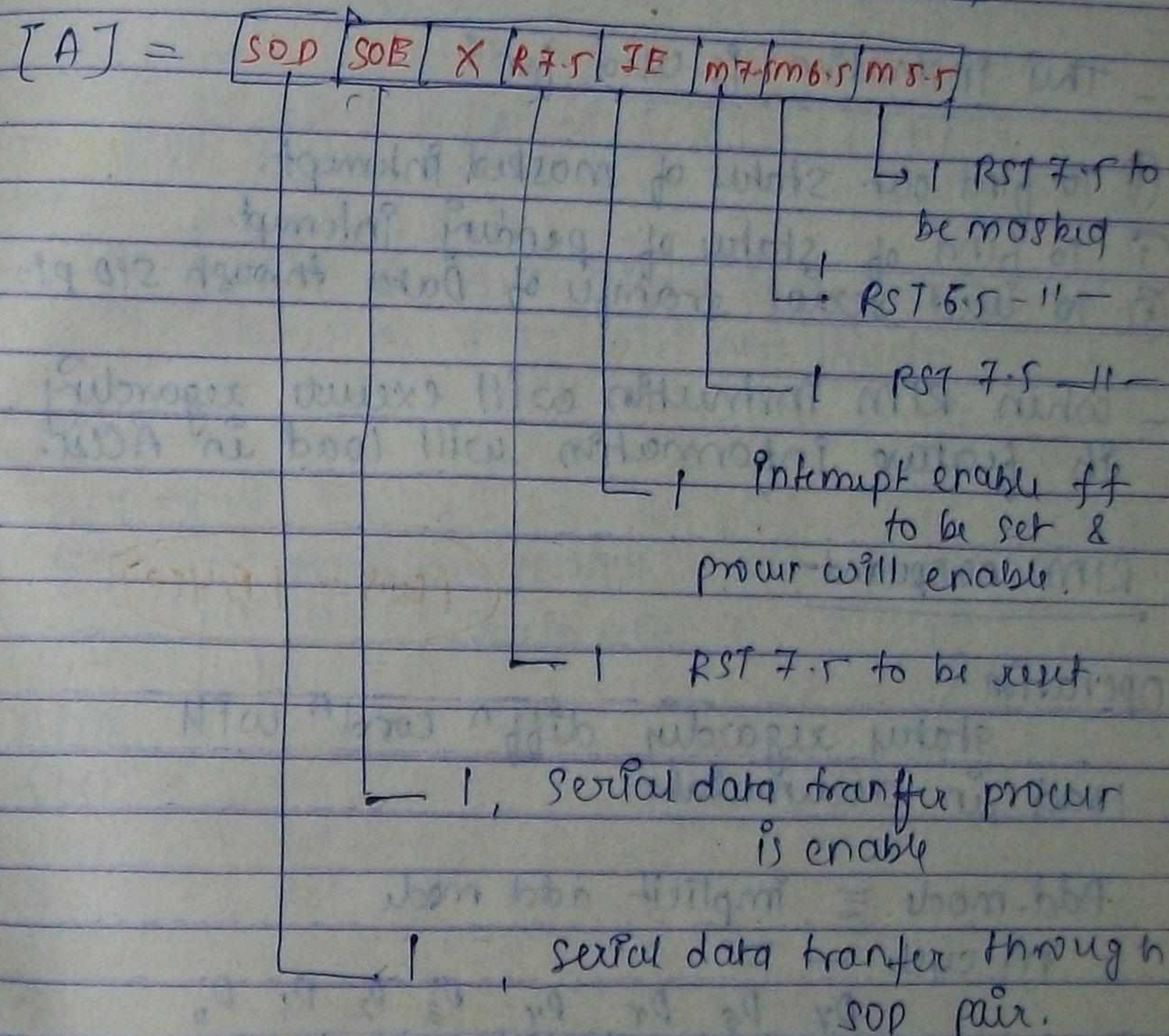
Operation -

executes on the basis of content of accumulator & perform desired task

Add mode → implicit add mode
M.C → F

(1 WS = 1 Byte)

$[A] = [D_7 | D_6 | D_5 | D_4 | D_3 | D_2 | D_1 | D_0]$



Ex -

MVI A, 0BH

SIM

$[A] = [0 | 0 | 0 | 0 | 1 | 0 | 1 | 1]$

RIM (READ INTERRUPT MASK) -

- This instruction is used for 3

- ① To find out status of masked interrupt.
- ② To find of status of pending interrupt.
- ③ To send serial transfer of data through SID pin.

- when RIM instruction will execute regarding its feature information will load in ACCU.

RIM no operand-

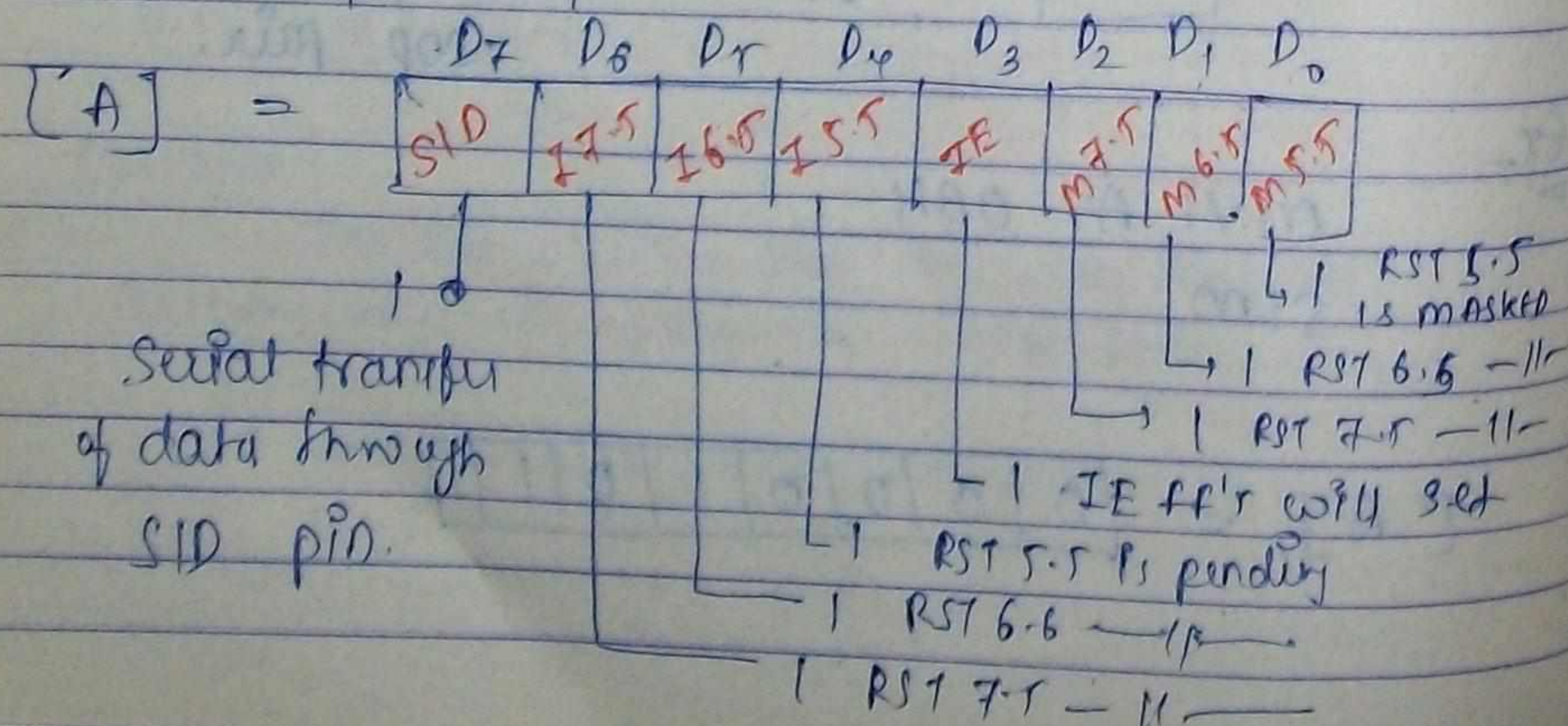
(CWS = 1 BYTE)

Operation -

status regarding diffⁿ condⁿ will load in accumulator

Add. mode = implicit add mode

m cycle - F



Status of Interrupt?

Ex -

Seven segment display

RIM

OUT 05H

[A] =

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	1	1	1	1

NOTE -

- RIM & SIM is the machine control instruction. So status of flag will not affect.

- ROP

HLT

EI

DI

RIM

SIM

machine control instruction.

4/12/19

Ques. EI about set of instructions
RIM performing which of
ANI 08H the task?
SIM

Solⁿs

- ① Send bit on O/P
- ② Accept bit at I/P
- ③ Accept RST 7.5 as interrupt
- ④ Reset RST 7.5 interrupt

Interrupt enable ff will set

RIM [A] = D₇ D₆ D₅ D₄ D₃ D₂ D₁ D₀

08H = 0 0 0 0 1 0 0 0

SIM =

RST 7.5 }
RST 6.5 } all are
RST 5.5 } marked

INTERFACING -

It is of two types:

- memory interfacing
- I/P - O/P interfacing

memory mapped I/O interfacing
I/O mapped I/O interfacing or
peripheral mapped I/O interfacing

GENERAL STEPS OF INTERFACING -

- ① Command signal connect directly to respective pin.

$\overline{RD} \rightarrow \overline{RD}$

$\overline{WR} \rightarrow \overline{WR}$

- ② Data pins connected in parallel directly

(LSB)_{device} $\longleftrightarrow$ (LSB)_{up}
:
:
(MSB)_{device} $\longleftrightarrow$ (MSB)_{up} } Data bus

- ③ Address of Interface device connect directly as

(LSB)_{device} $\longleftrightarrow$ (LSB)_{up}
:
:
(MSB)_{device} $\longleftrightarrow$ (MSB)_{up} } add. bus

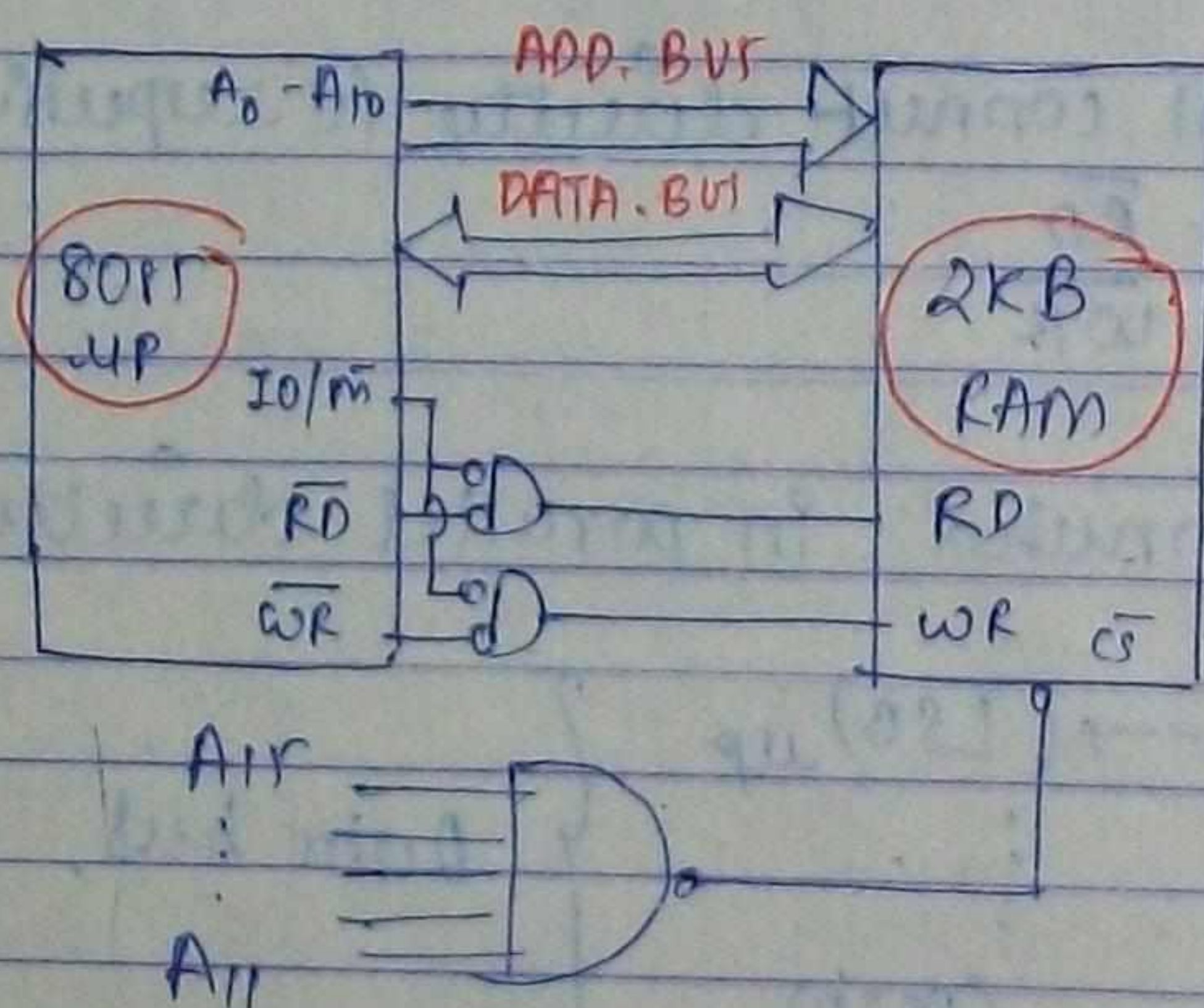
- Rest of bus of μp is used for development of chip selection logic.

- chip select logic can be define by two ways:

① By use of logic gates or buffer circuit.

② By use of decoder circuit.

⇒ MEMORY INTERFACING -



NOTE -

First find out add. line & data line in memory.

$$2^n \times d$$

$n \equiv$ add. line
 $d =$ data line

$$2^{10} \times 2^4 = 2^{14}$$

7000H to 7FFFH
2 0000H to 0FFFH

Ex $2K \times 8$
 $2^{11} \times 8$

$n=11$
 $d=8$

A ₁₀	A ₉	A ₈	A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀
0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	1

0000H
7FFFH

find memory add. range interfaced.

A ₁₅	A ₁₄	A ₁₃	A ₁₂	A ₁₁	A ₁₀	A ₉	A ₈	A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

MIN ADD: 0000 0000 0000 0000 = F800H
MAX ADD: 1111 1111 1111 1111 = FFFFH

Ques. 4KB RAM interfaced with 8085 μp and CS = $\bar{A}_{15} A_{14} A_{13}$ then fundamental add. interfaced.

Ans. 7000H - 7FFFH or 6000H - 6FFFH

4KB RAM = $2^{12} \times 8$ add. bus
000H - FFFH

Now

A ₁₅	A ₁₄	A ₁₃	A ₁₂	000H	7000H
0	1	1	1	FFF	7FFFH

$A_{15} A_{14} A_{13} A_{12} A_{11} A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0$

$0 \ 1 \ 1 \ X \ A_{11} A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0$

min add.

$0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 = 6000H$

max. add.

$0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 = 6FFFH$

min add.

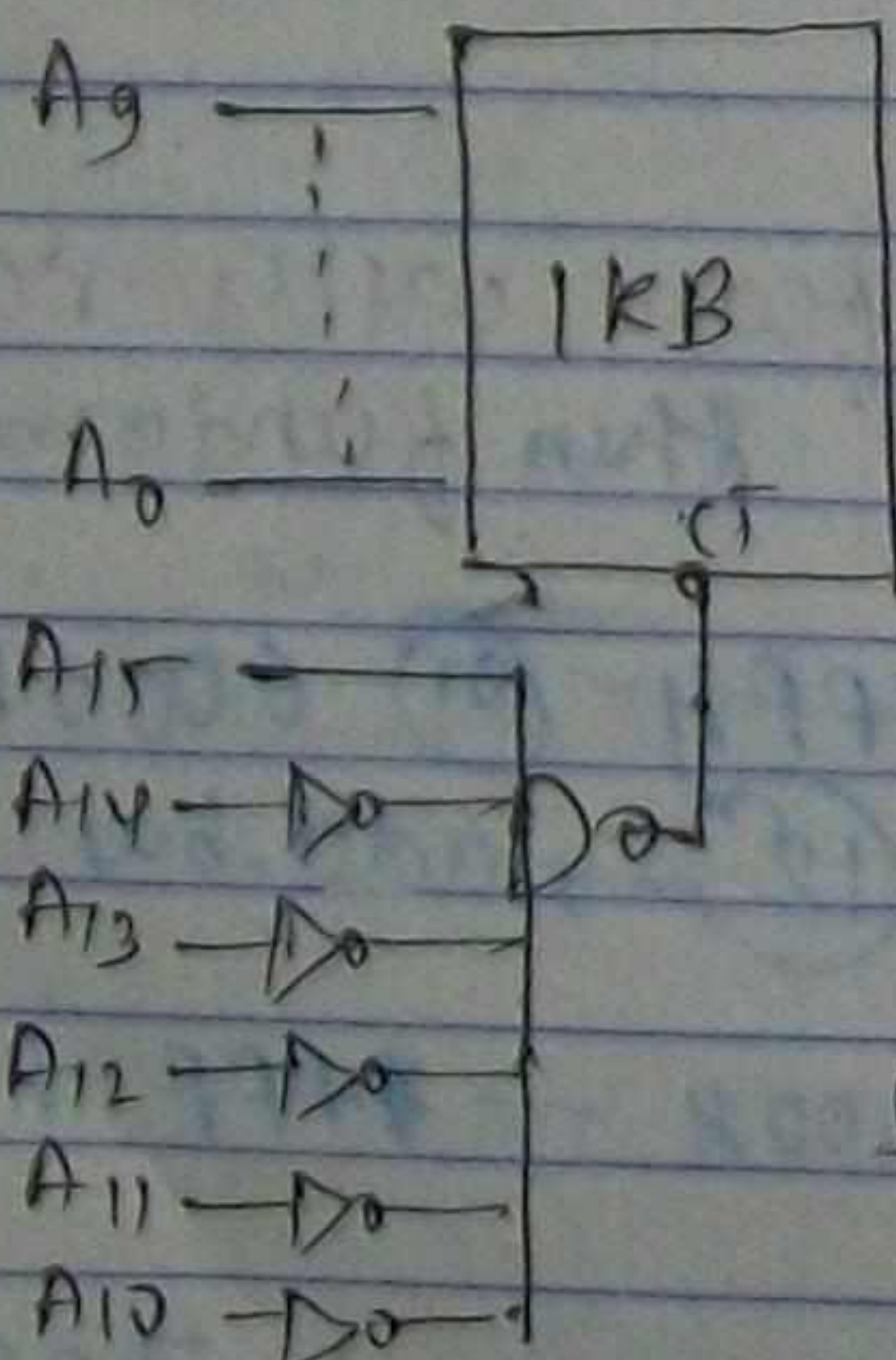
$0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 = 7000H$

max add.

$0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 = 7FFFH$

$$\text{TOTAL memory RANGE} = \frac{\text{LAST memory ADD.}}{\text{memory}} - \frac{\text{1st memory ADD.}}{\text{memory}} + 1$$

Ques.



Consider a memory chip with 1024 Bytes Storage connect to a 8085 up add. bus (or any other up having add 16 bits) as shown in fig.

What is the range of memory add.

① $0000H - D300H$

$1KB = 2^{10}$

② $1000H - 13FFH$

$n=10$

③ $8000H - 83FFH$

④ $9000H - FFFFH$

Solⁿ 1

$A_{15} A_{14} A_{13} A_{12} A_{11} A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0$

$1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0$

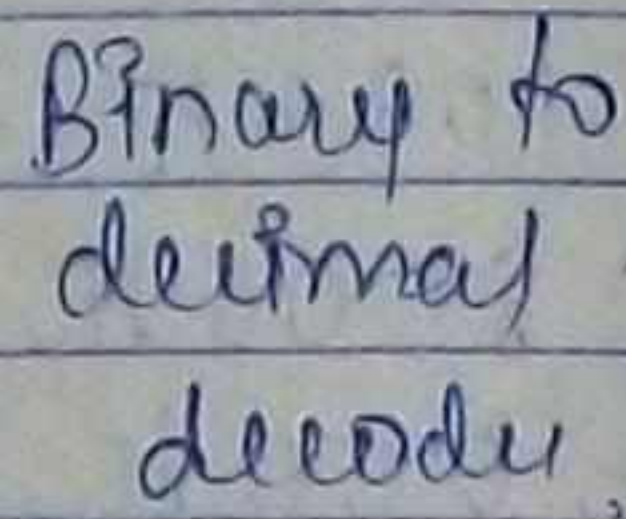
min add

$1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 = 8000H$

max. add.

$1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 = 83FFH$

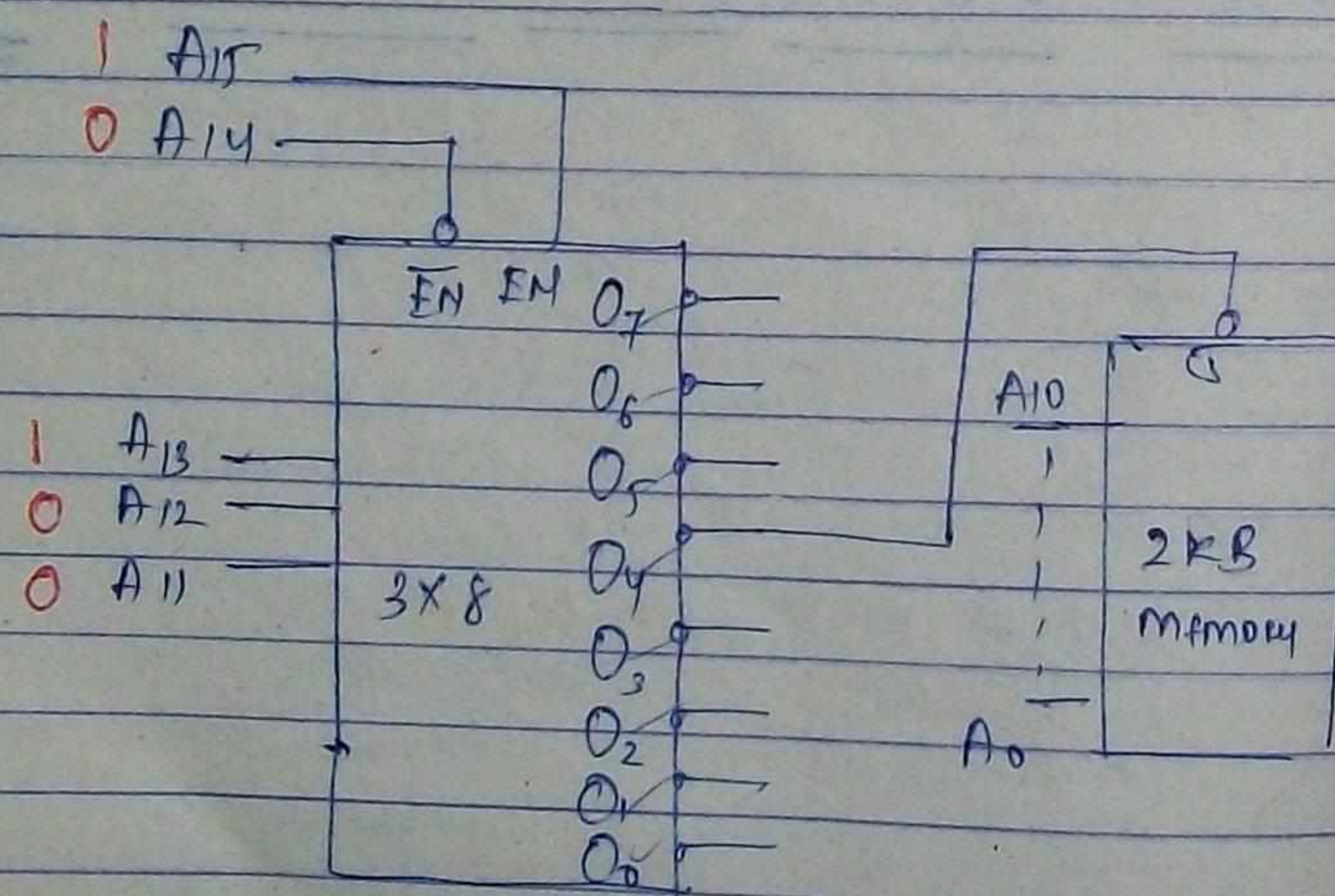
DECODER :-



MSB LSB

AB	D_3	D_2	A_1	D_0
00	0	0	0	1
01	0	0	1	0
10	0	1	0	0
11	1	0	0	0

Ex -



Find memory add. rang if 2 KB memory partitioned with 8085 μ p.

$A_{15} A_{14} A_{13} A_{12} A_{11} A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0$
 $A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0$
 $1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 = A000H$
 $1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 = A7ffH$

(msb) delect = High order add. line

(LSB) decoder = lower order add. line.

CASE (2) : If decoder function is given.

(O/P) $\rightarrow$ value of $\rightarrow$ connected
diode diode address
function line.

① If $f(ABC) \rightarrow (4)_{10} = f(100)$

↓ ↓

msb lsb

① If $f(CBA) \rightarrow (4) = f(100)$

⇒ I/O INTERFACING -

It is of two types:

- ① memory mapped I/O interfacing
- ② I/O mapped I/O interfacing or peripheral mapping I/O interfacing.

COMPARISON -

CHARACTERISTIC	MEMORY MAPPED I/O INTERFACING	PERIPHERAL MAPPED I/O INTERFACING
① COMMAND SIGNAL	MEMRD / MEMWR	IO RD / IO WR
② INSTRUCTIONS	All memory related instructions are valid.	IN & OUT only 2 instructions are valid.
③ EXECUTION	Arithmetic & logic operation directly possible & data transfer b/w any memory register & I/O	data can be transferred from / to accumulator only.
④ NO. OF DEVICE INTERFACED	64KB memory shared b/w I/O and system memory.	max 256 I/P & 256 O/P device interfaced
⑤ HARDWARE REQUIRE	more	less

⑥ EXECUTION SPEED	SLOWER	FASTER
⑦ APPLICATION	Smaller system	larger system

⇒ SOME IMPORTANT PERIPHERALS -

8255 = programmable peripheral interface (PPI)

8279 = keyboard & display interface

8259 = interrupt controller

8237 = DMA controller

8257 = DMA

8155 = programmable I/O port & timer.

8254 / 8253 = programmable interval timer.

⇒ SOME IMPORTANT DIGITAL IC -

- ✓ 74182 = Look ahead carry generator
- 74180 = 8 bit parity generator & checker
- 74121 = monostable multivibrator

7493 = 14 bit binary counter

✓ 7477 = Seven segment display

7490 = decade counter

✓ 7400 = Quad 2 I/P NAND gate

✓ 7402 = Quad 2 I/P NOR gate

7408 = Quad 2 I/P AND gate

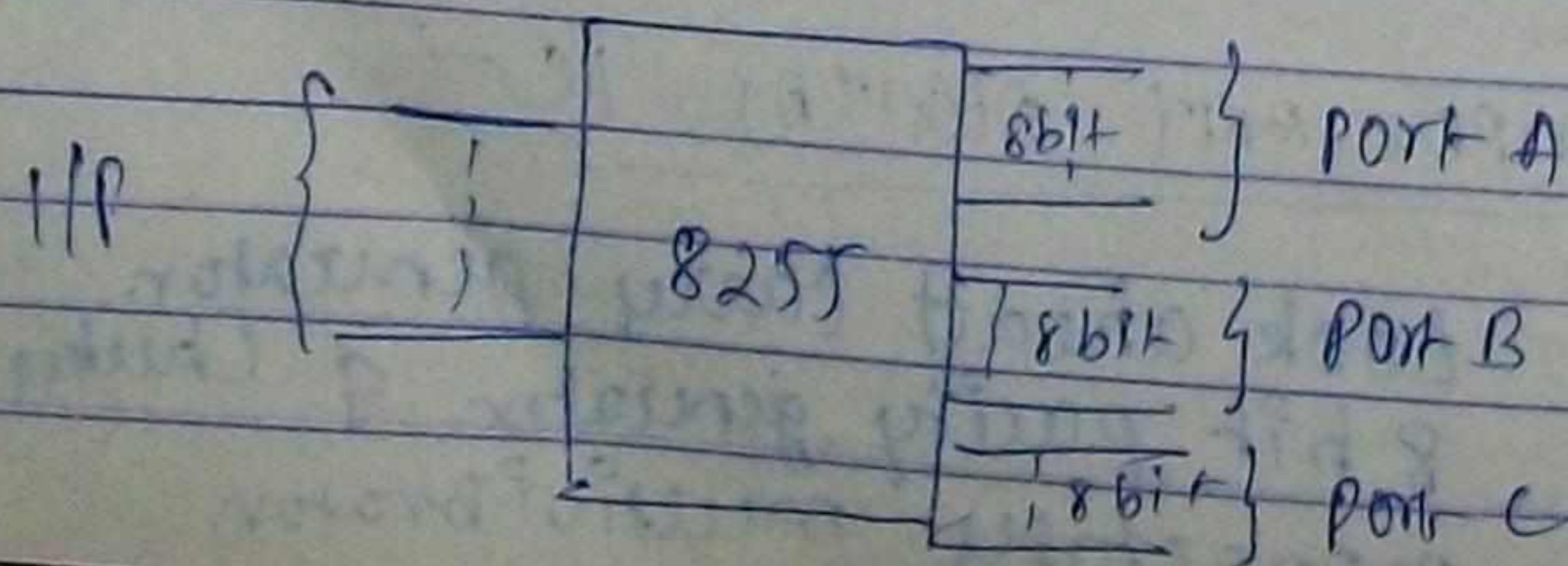
7432 = 14 OR gate

7486 = 14 X-OR gate

⇒ 8255 -

- It is 40 pin IC

- It has three port each have port add. of 8 bit as 'PORT A', 'PORT B', 'PORT C'.



- Each port can work as I/P - O/P port individually

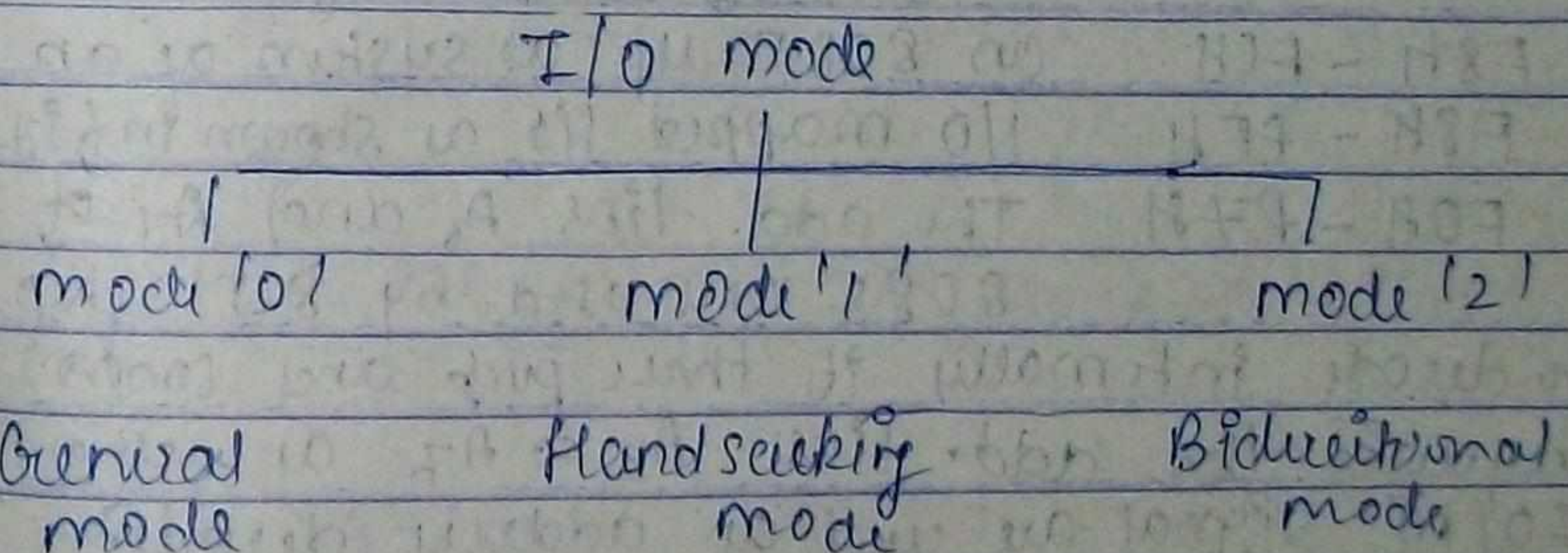
- Port C can also work in two port having port add. of four bit (lower nibble & upper nibble) as port 'C', upper and port 'C' lower.

- Port 'C' upper and port 'C' lower generate control signal for port 'A' & port 'B' separately.

- 8255 can work in two mode.

① Bit set/reset mode (BSR mode)

② I/O mode.

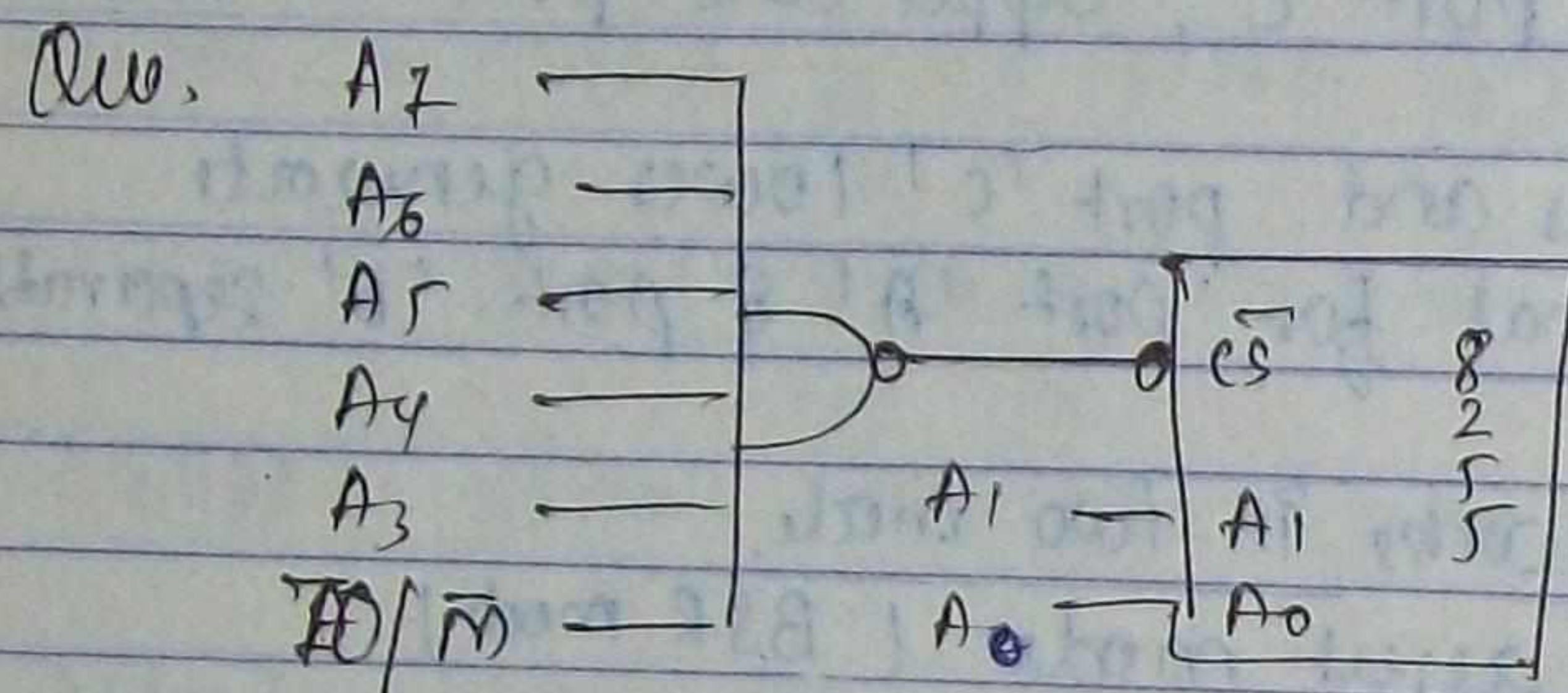


① MODE '0' GENERAL MODE -
In this mode port 'A', port 'B' and port 'C' can work as I/O port separately.

② MODE '1' HANDSHAKING MODE -
Port 'A' and port 'B' work in handshaking mode and port 'C' upper & port 'C' lower will generate control signal for port A & port B separately.

③ MODE '2' BIDIRECTIONAL MODE

port 'A' work is bidirectional mode only and port 'C' upper generate control sign for it.



① F8H - FBH

② F8H - FCH

③ F8H - FFH

④ F0H - F7H

An 8255 chip interfaced to an 8085 up as system as an I/O mapped I/O as shown in fig. The add. line A_8 and A_1 of 8085 are used by 8255 chip to decode internally its three ports and control required the add. line A_3 to A_7 as well as $IO/\bar{M}$ signal are used as address decoding. The range of address decoding. The range of add. for which 8255 get selected.

Solⁿ

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
1	1	1	1	1	0	1	1	F8H
1	1	1	1	1	0	0	0	
1	1	1	1	1	0	1	1	FBH
1	1	1	1	1	0	0	0	FCH
1	1	1	1	1	1	1	1	FFH

taken as don't care

Ques write 8085 ALP to transfer 10 byte data that store in continuous memory range 2001 to 200A

to 3001 to 300A

Solⁿ $[[2001]] \rightarrow [[3001]]$
 $[[200A]] \rightarrow [[300A]]$

```

- LXI B 2001H ; [BC] ← 2001
  LXI D 3001H ; [DE] ← 3001
  MVI H 0AH ; [H] ← 0AH
loop: LDAX B ; [A] ← [BC]
      STAX D ; [A] → [DE]
      INX B ; [BC] ← [BC] + 1
      INX D ; [DE] ← [DE] + 1
      DCR H ; [H] ← [H] - 1
      JNZ loop ; In cond'n satisfy set then jump
      HLT ; otherwise loop will terminate
  
```

stop execution

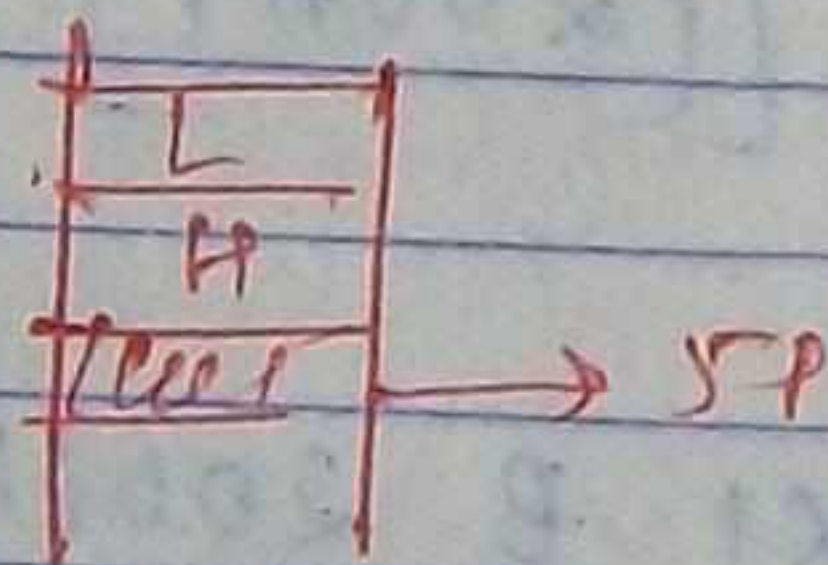
Ques. write an 8085 ALP to add a 16 bit no. in location 5000H (high byte) and 5001 (low byte) with another 16 bit no. stored in 5002 (high byte) and 5003 (low byte) and final result store in Bc.

Soln:

```

LDA 5000H ; [A] ← 5000
MOV H, A ; [HL] ← [A] ← 5000H (data)
LDA 5001H
MOV L, A
LDA 5002H
MOV B, A ; [B] ←
LDA 5003H
MOV C, A
DAD B
LXI SP, 3000H
PUSH H
POP B
HLT

```



02

```

XGHB
MOV B, D
MOV C, E

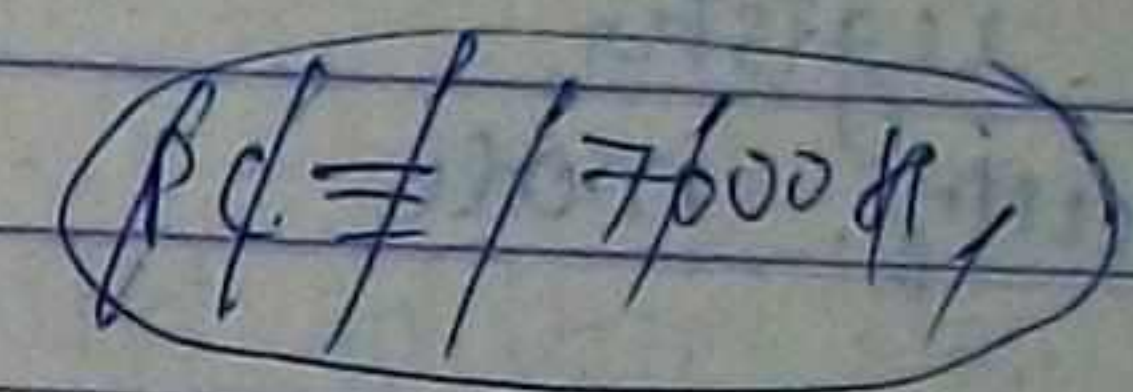
```

Q. ORG 7000H ; PC = 7000

```

BEGIN: LXI H, 7000H
      ADD H, MOV AL
      JP END
      RSTO
      END: PCHL
          HLT

```



Explain function of program

```

ORG 7000H ; PC = 7000
LXI H, 7000H ; [H] = 70H
              [L] = 00H

```

```

MOV AL, 00H ; [A] = 00H
ADD H ; 
      0000 0000
      0011 0000
      0111 0000

```

S = 0

8086-1 = 8 MHz
8086-2 = 10 MHz

INTRODUCTION OF 8086 -

- Comparison b/w 8085 & 8086
- Internal architecture
- memory segmentation & physical add.
- flag register
- Addressing mode

① COMPARISON B/W 8085 / 8086 -

<u>8085 μP</u>	<u>8086 μP</u>
- Based upon NMOS tech.	- Based upon HMOS tech.
- It is 40 pin IC.	- It is 40 pin IC.
- $V_{cc} = +5V$	- $V_{cc} = +5V$
- Operating freq = 3 MHz	- Operating freq = 5 MHz
- add. bus = 16	- Add. bus = 20
- Max. memory that can interface = 64 KB	- Max. memory that can be interface = 1 MB
- Size of ALU or bit of μ P = 8 bit	- Size of ALU or bit of μ P = 16 bit
- flag register = 8 bit	- flag register = 16 bit
- No. of flag = 5	- No. of flag = 9

INTERNAL ARCHITECTURE -

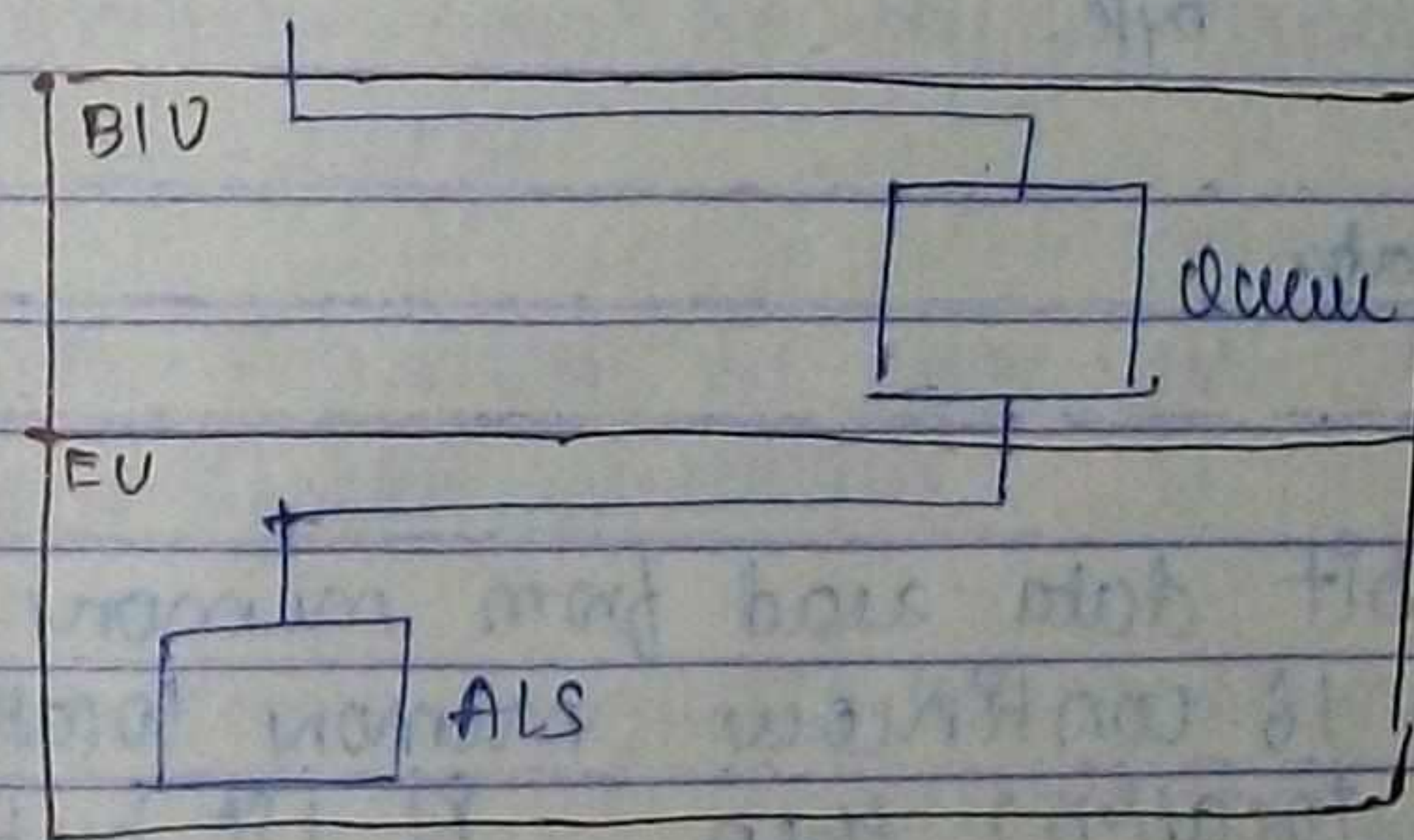
Internal architecture divide in two part

- ① Bus interface unit (BIU)
- ② Execution unit (EU)

BIU -

All type of data and address flow on bus managed by BIU.

- In BIU there is a queue of six location that used for temporary storage of instruction that fetch during the execution of one instruction.



- Queue works on the principle of FIFO principle
FIFO

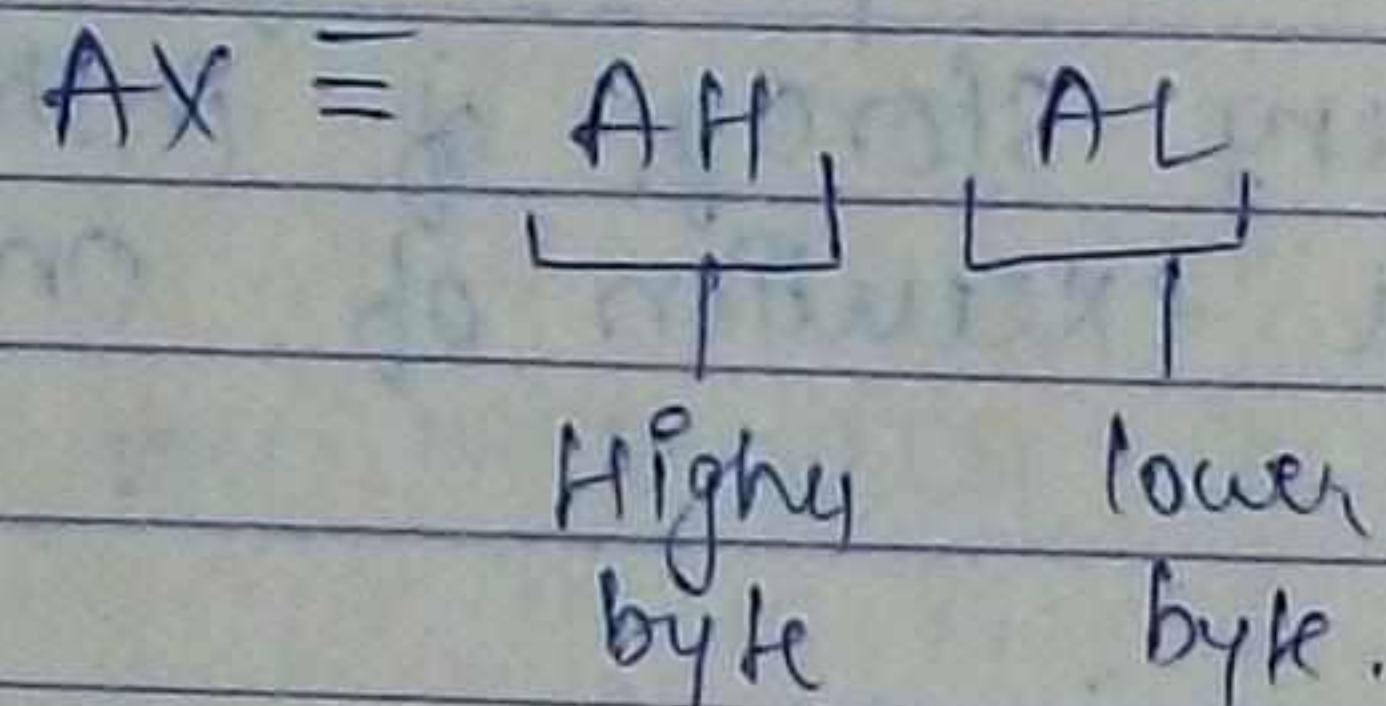
PIPELINING -

During the execution of one instruction next instructions fetched from memory is known as pipelining.

EXECUTION UNIT (EU) -

- All type of data execution is done in EU
- ALU ~~not~~ present in EU

Accumulator = AX of 16 bit Register.



AL = Accumulator

- In 8086, if 16 bit data read from memory, that stored at 16 continuous memory location (8 bit at one location) then, if 1st byte is at even add. 16 bit data will read in one machine cycle.

If 1st byte at odd address then two machine cycle required to read 16 bit data but processor 16 bit data in one machine cycle.

- But in case of 8088 if 16 bit data read from memory that store at

two continuous memory location always required two machine cycle. It means it can read only 8 bit data in one machine cycle but processor 16 bit data in one machine cycle.

It is externally 8 bit and internally 16 bit up

⇒ MEMORY SEGMENTATION -

- 1 MB memory divided in four segments of continuous memory range of 64 KB. and each segment can be define anywhere inside the 1 MB range.
- Each segment is used for storage of definite type of information.

Code segment = 64 KB continuous memory range = storage of instruction set.

Stack segment = 64 KB continuous memory range = storage of temporary information during the execution of main program.

Data segment } = 64 KB continuous memory range = data storage.
Extra segment }

- Regarding each segment a 16 bit register that hold the higher 16 bit of bottom add. of each segment.

Code seg. = CS (16 bit reg.)
 Stack seg. = SS "
 Data seg. = DS "
 Extra seg. = ES "

- Bottom add. ~~define~~ of each segment define in such a way that lower 16 bit should be zero.

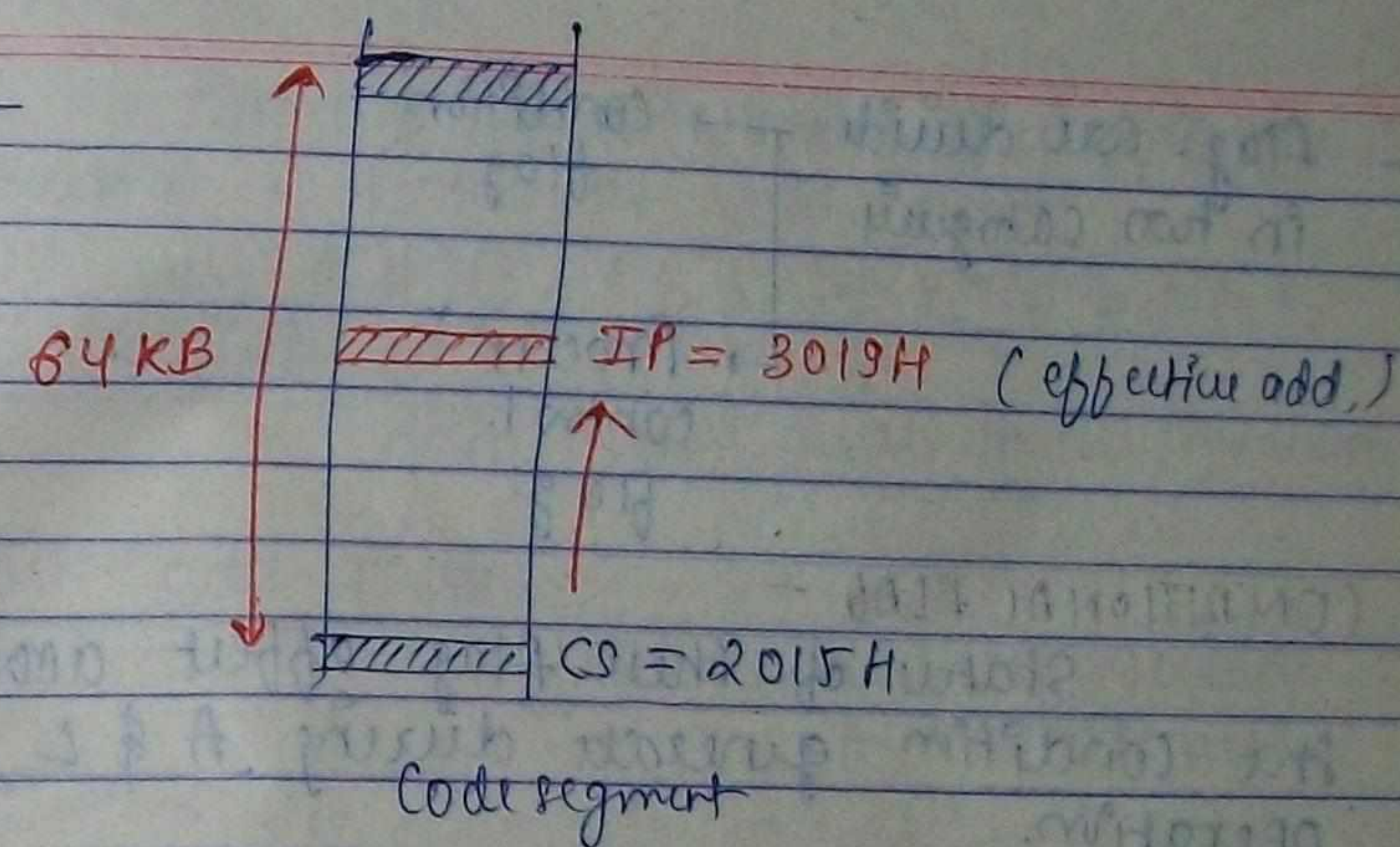
- Regarding each segment another 16 bit register that hold add. in range of 64 KB (000H - FFFFH) is known as effective add.

Code seg. = IP (Instruction pointer) 16 bit reg.
 Stack seg. = SP (Stack pointer) 16 bit reg.
 Data seg. = SI (Source Index) 16 bit reg.
 Extra seg. = DI (Destination Index) 16 bit reg.

Physical add.

(Memory add.
 memory location add.
 effective memory add.)

Ex -



- In 8086 IP is similar as PC of 8085.

CS : IP

Offset add.

physical add CS = 2015 ~~hard wired~~ zero
 IP 3019

Physical add 23169H

⇒ FLAG REGISTERS -

- All five flag of 8085 is exactly common in 8086.
- Size of flag register is 16 bit.
- Total no. of flags = 9.

- flag. are divide in two category
 - Condition flag.
 - Process control flag

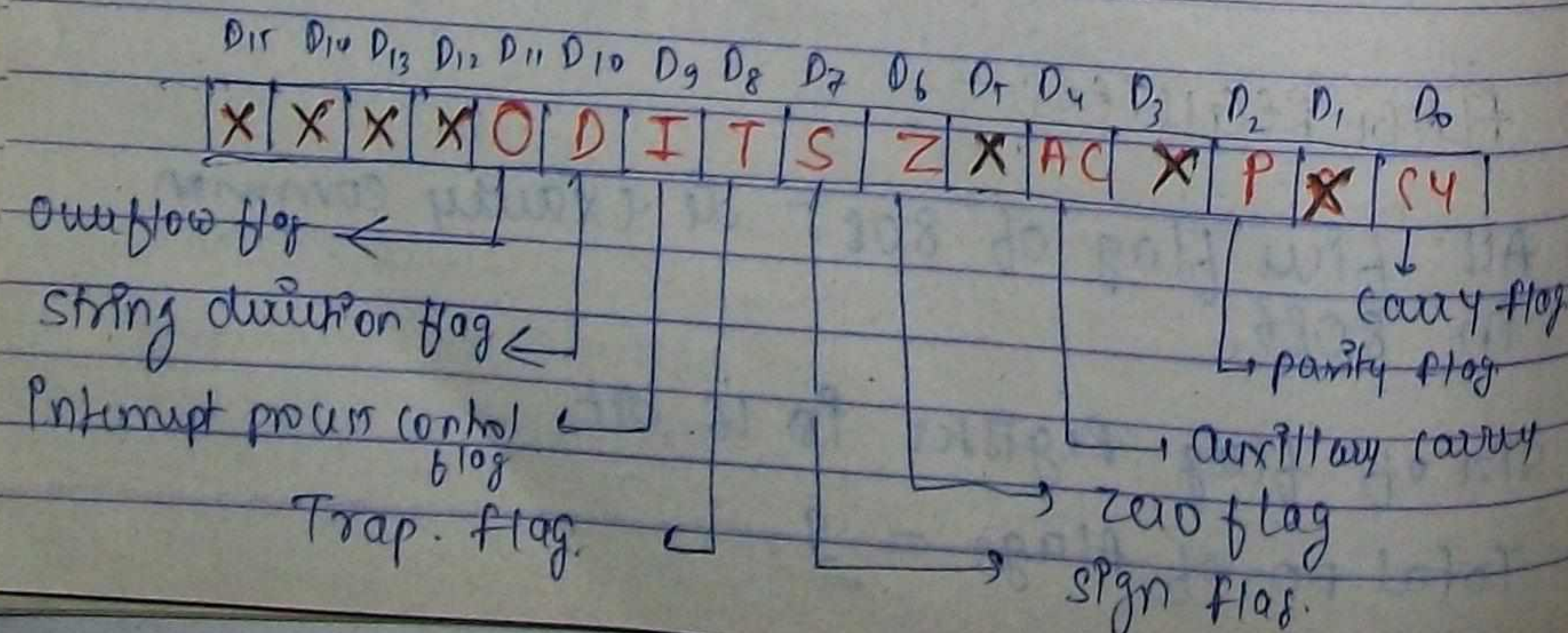
CONDITIONAL FLAG -

Status of these flag affect according the condition generated during A & L operation.

- carry flag (CY)
- Parity flag (P)
- auxiliary carry (AC)
- zero flag (Z)
- sign flag (S)
- overflow flag (O).

PROCESS CONTROL FLAG -

Status of these flag used for process controlling.



⇒ ADDRESSING MODE -

Form of effective add. given in the instruction is known as add. mode.

① Immediate add. mode -
data itself given in the instruction.

② Direct add. mode -
Effective add. itself given in the instruction.

③ Indirect add. mode -
E.A. data given in the form of content of register.

④ Relative register add. mode -

NOTE RELATIVE $\equiv$ 8 or 16 bit displacement
no. provided externally.

$$EA = [BX] + \text{8 or 16 bit displacement no.}$$

⑤ Base Index add. mode -
given as operand.

⑥ Base Index add. mode -

$$EA = [BX] + \begin{matrix} SI \\ DI \end{matrix}$$

given as operand.

⑥ Relative base indir. add. mode -

$$EA = [BX] + \left[\begin{matrix} I \\ DI \end{matrix} \right] + 8 \text{ or } 16 \text{ bit displacement}$$

gives an operand.

Number 2417102

(number on left) - position
place, color, number, etc.

(number on right) - position
place, color, number, etc.

A 16-bit value is stored in the register. The value is stored in the register in the form of a 16-bit value. The value is stored in the register in the form of a 16-bit value.

Ex	Base	Range
2	8	0, 1, 2, ... 7

The value is stored in the register. The value is stored in the register in the form of a 16-bit value. The value is stored in the register in the form of a 16-bit value.

The value is stored in the register. The value is stored in the register in the form of a 16-bit value. The value is stored in the register in the form of a 16-bit value.

NUMBER SYSTEMS

⇒ CLASSIFICATIONS OF NO. SYSTEM -

- POSITIONAL (weighted NO. system)
binary, Octal, decimal, Hexad., BCD
- NON POSITIONAL (non-weighted NO. system)
Gray, excess-3

— A NO. system whose 'base' or 'radix' is 'r' that will contain all different digit, they are in the range 0 to (r-1).

Ex	Base	range
	2	0, 1
	8	0, 1, 2, ..., 7

⇒ CONVERSION -

① DECIMAL TO OTHER BASE -

To convert in other base 'r' divide the integer part and multiply the fractional part with 'r' repeatedly until the integer and fractional will become zero.

During division operation take the remainder & follow bottom to top approach to conclude the value.

During multiplⁿ take the integer part and follow the top to bottom approach to conclude the value.

Ex - ① $(25.625)_{10} \rightarrow (x)_2, (x)_4, (x)_6, (x)_8$

2	25	1
2	14	0
2	7	1
2	3	1
1	1	

$$.625 \times 2 = 1.250 \rightarrow 1$$

$$.250 \times 2 = 0.500 \rightarrow 0$$

$$.500 \times 2 = 1.000 \rightarrow 1$$

$$.000 \times 2 = 0.000 \rightarrow 0$$

$$= (11101.1010)_2$$

4	25	1
4	7	3
	1	

$$.625 \times 4 = 2.500 \rightarrow 2$$

$$.500 \times 4 = 2.000 \rightarrow 2$$

$$.000 \times 4 = 0$$

$$(131.220)_4$$

② OTHER BASE TO DECIMAL -

$$(N)_r = (X)_{10}$$

$$\dots N_3 \cdot r^3 + N_2 \cdot r^2 + N_1 \cdot r^1 + N_0 \cdot r^0 + N_{-1} \cdot r^{-1} + N_{-2} \cdot r^{-2} \dots$$

$$\text{Ex } (22E.0)_{15} = ()_{10}$$

when base is not given then $\text{base} = \text{highest value} + 1$

$$= \text{base} = E + 1 = 15$$

⇒ DATA REPRESENTATION -

Data can be represented as:

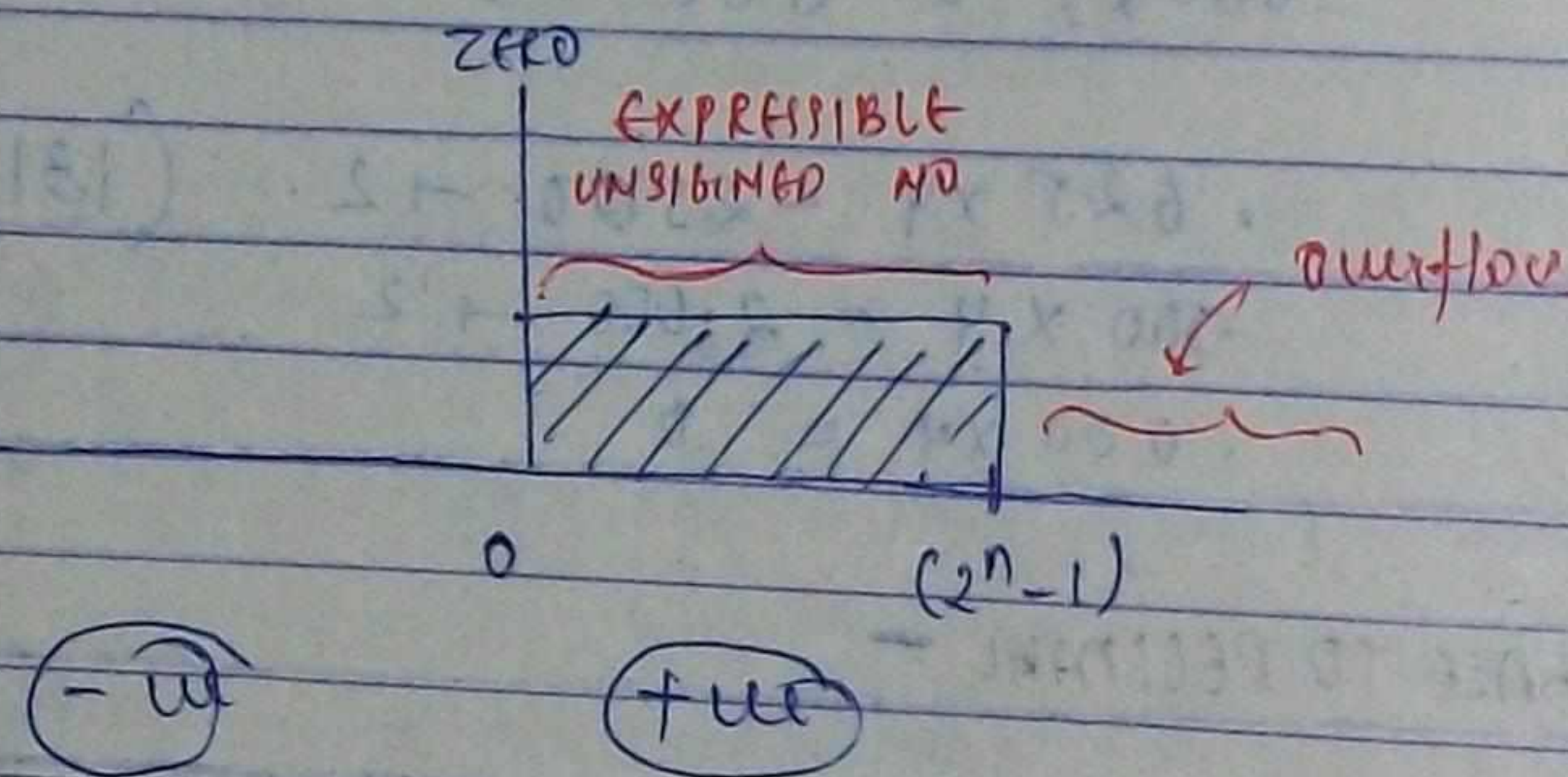
1. magnitude representation
2. Complement representation

[unsigned
signed magnitude
Direct radix complement
radix complement

① UNSIGNED REPRESENTATION -

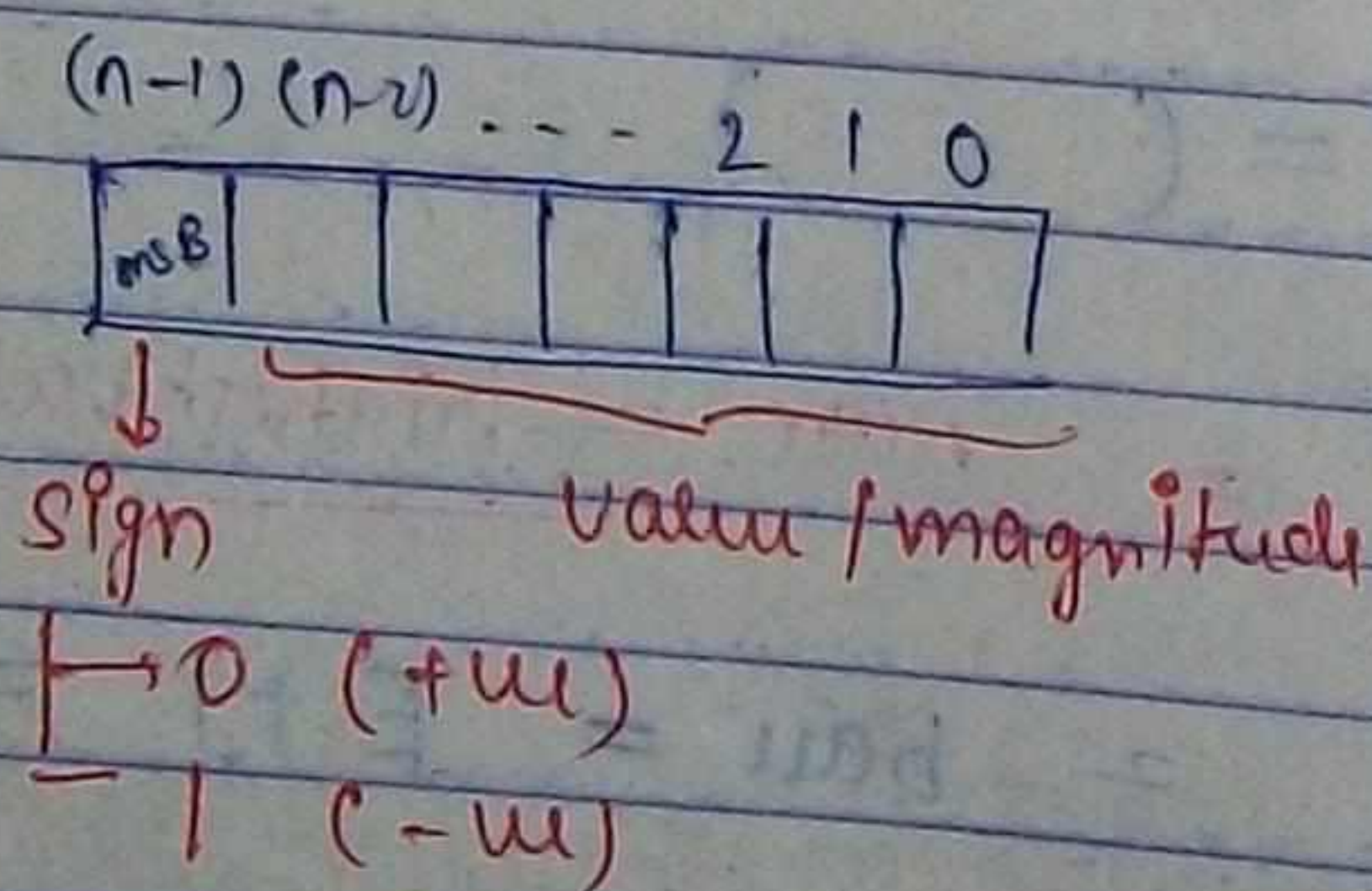
- Represents only +ve no., therefore no need of sign bit in the binary sequence.

- for n bit unsigned range is 0 to $2^n - 1$.



② SIGNED MAGNITUDE REPRESENTATION -

- Represent both +ve and -ve no. format

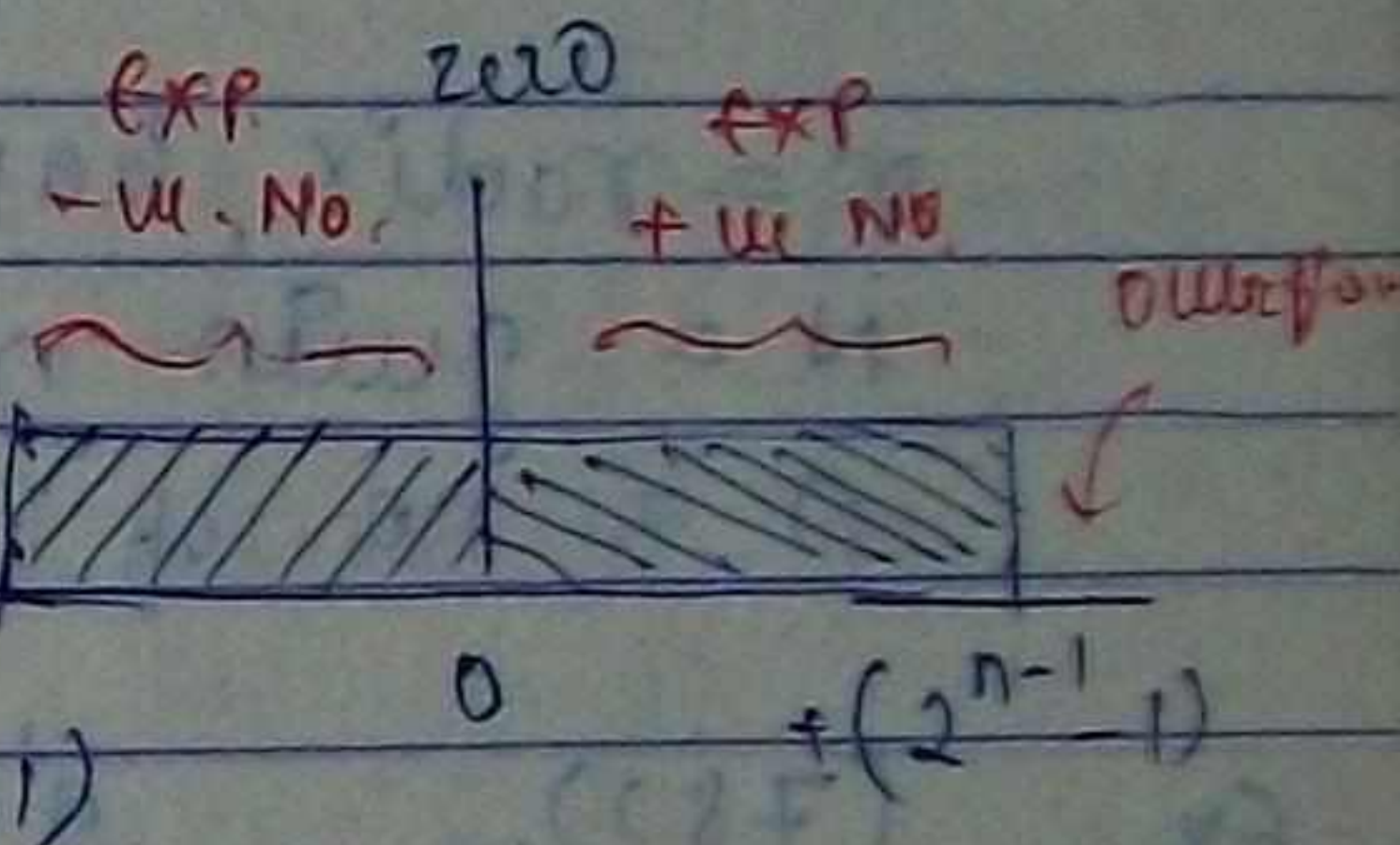


- for n bit signed mag range is $-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$

BINARY SEQ

Decimal EQUIVALENT

0000	+0
0001	+1
0010	+2
⋮	⋮ to +7
⋮	then 0, -1, ...
1101	-5
1110	-6
1111	-7



not used in 11p
since 0 is associated with two patterns - 0000 and 1000 - more CF will be required.
ambiguity.

⇒ COMPLEMENT REPRESENTATION -

1. Diminished radix $(r-1)$'s complement
2. radix (r) 's complement

Base

Complement

2	1's	3's
	2's	4's
8	7's	9's
	8's	12's
10	9's	15's
	10's	16's
16	15's	
	16's	

① (r-1)'s complement -

To determine (r-1)'s complⁿ Subtract the given no. from the max. possible no in the given base.

$$(r-1)'s = [(r^n - 1) - N]$$

r = radix, base

N = given no.

n = no. of digit

Ex $(782)_{10}$ 9's complement = ?

$$= 999 - 782$$

Ex $(32)_4$ 3's com

$$= [(4^2 - 1) - 32]$$

$$= 33 - 32 = 1$$

② r's complement -

To determine 'r's compl. first write the (r-1)'s complement then add 1 to the LSB.

* The processor is able to recognise the binary sequence so to represent the data using the complement format take the base 2 no. system.

- Base 2 no. system consist of two different format to represent the data i.e 1's and 2's complⁿ format

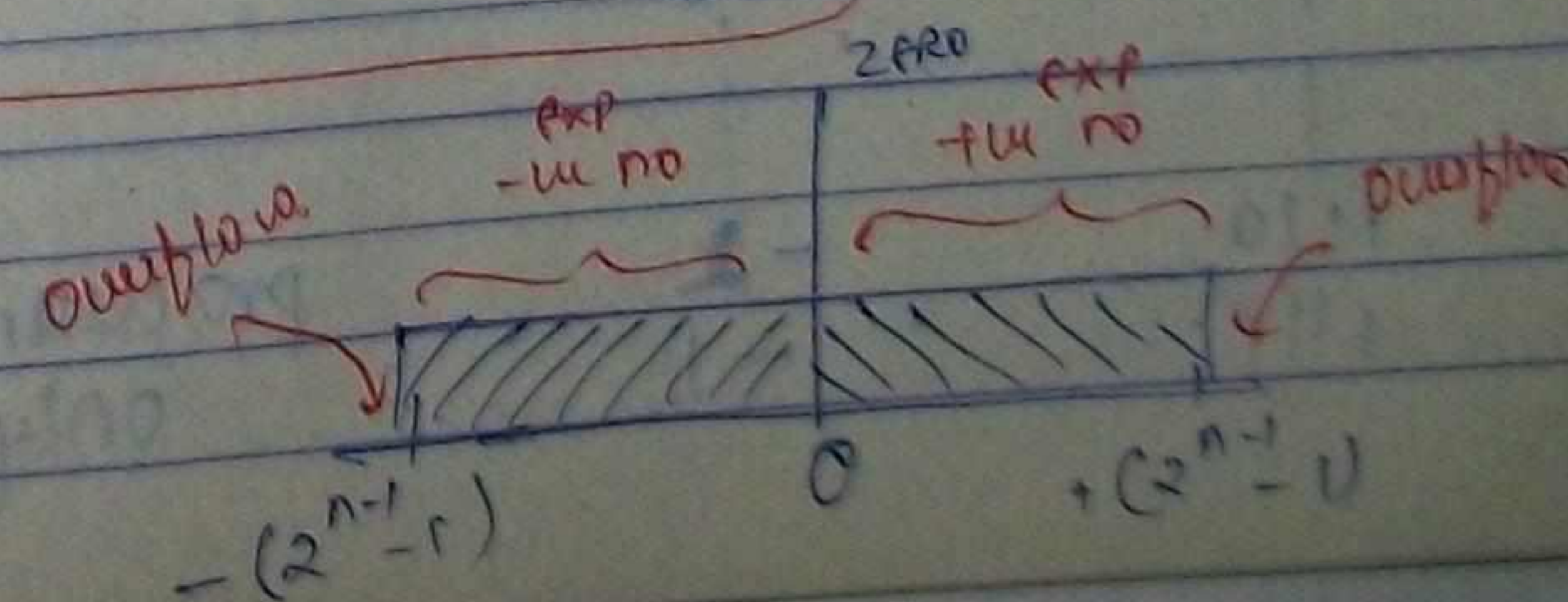
① 1's COMPLEMENT REPRESENTATION -

- Represent both +ve & -ve no.

- when sign bit = 0, no need to take the complement to conclude the value, when sign bit = 1, there is a need of taking of complement to conclude the value.

range for n-bit: $-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$

Binary	equivalent	
0000	+0 (0)	ambiguity
0001	+1	
1		
0111	+7	
1000	-7 (complement)	
1001	-6	
1		
1110	-1	
1111	-0 (0)	



- In this format also '0' is associated with two different pattern therefore it is not suitable to represent the signed no.

② 2's COMPLEMENT REPRESENTATION -

- Represent both +ve & -ve no.
- When sign bit = 1, there is a need of taking complement to conclude the value, otherwise no need of taking complement.

n bit range. $-(2^{n-1})$ to $(2^{n-1}-1)$

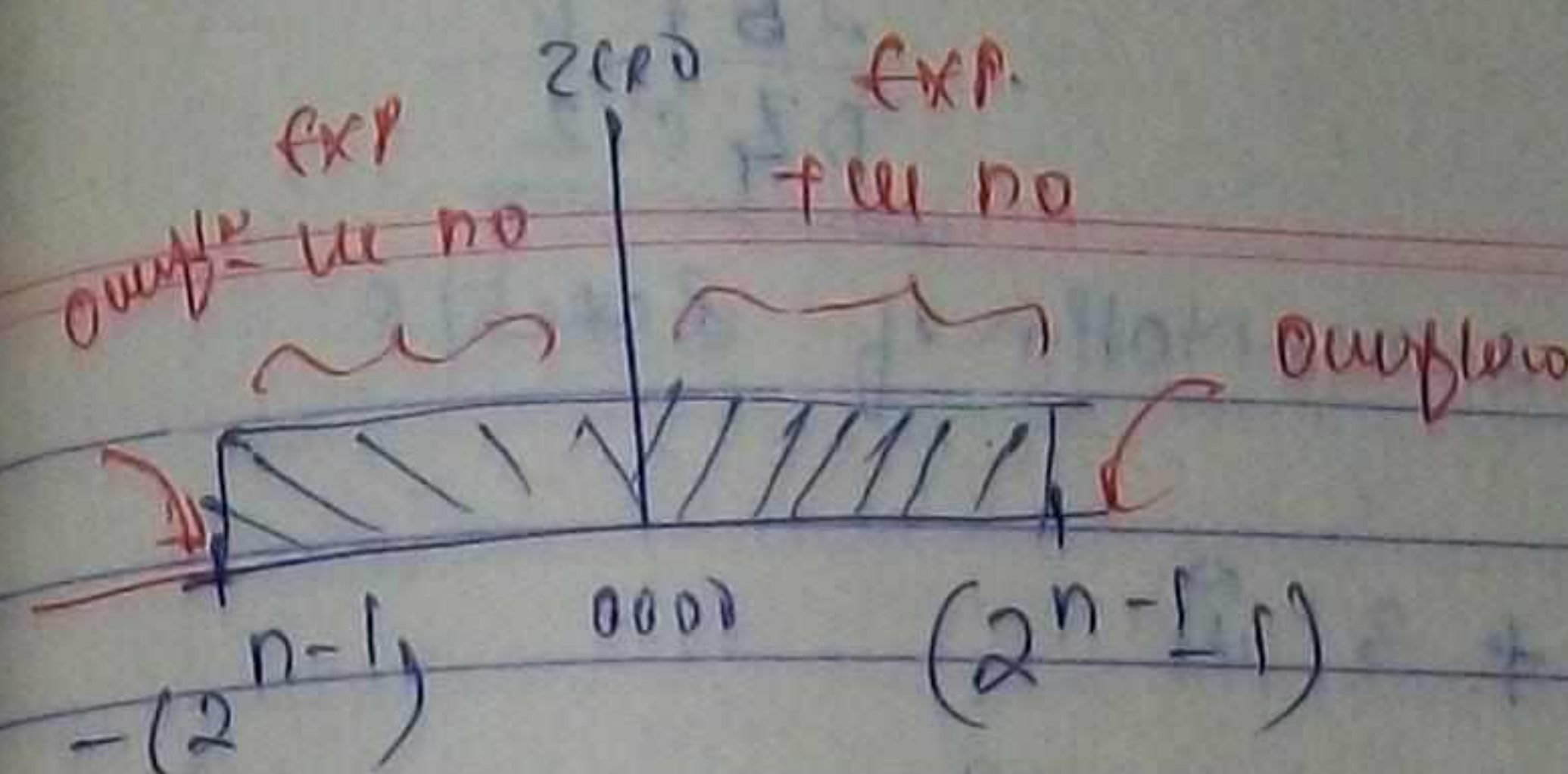
eg 4 bit $\Rightarrow -(+2^{4-1}-1)$ to $+(2^{4-1}-1)$

$\Rightarrow -8$ to $+7$

Binary eq.	Decimal
0000	+0
0001	+1
...	...
0111	+7
1000	-8
...	...
1110	-2
1111	-1

1000	1001
↓	↓
0111	0110
+1	+1
1000	1111 = 7

processor understand only 2's complement



- The above format consist of unique meaning each binary sequence (no ambiguity)
 2's complement representation is used to represent the signed no.

Q. Given in binary 2's complⁿ as (10011) decimal eqⁿ = ?

Ans. -13
 = binary equivalent of

Q. The binary eq of signed mag form of no 22.5625

Ans. 0 10110.1001

Q. min decimal eqⁿ of no. (11C)0 is

$$12 \times 13^0 + 1 \times 13^1 + 13^2 \times 1$$

$$12 + 13 + 169 = 194$$

Q. (FE35)₁₆ XOR (CB15)₁₆

(3520)₁₆

0011
 0001
 0010

ffff
28fd
D402

Ques. No. of 1's in binary representation of 3642

$$3 \times 512 + 7 \times 64 + 5 \times 8 + 3 \times 1$$

$$(2+1) \times 2^9 + (4+2+1) \times 2^6 + (4+1) \times 2^3 + 2+1$$

$$2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^3 + 2^1 + 2^0$$

$$\textcircled{1} \quad \textcircled{2} \quad \textcircled{3} \quad \textcircled{4} \quad \textcircled{5} \quad \textcircled{6} \quad \textcircled{7} \quad \textcircled{8} + \textcircled{9}$$

$$= \text{no of 1's} = 9 \quad (2 \equiv 10)$$

Ques. how many 1's are present in binary rep. of

$$4 \times 4096 + 9 \times 256 + 7 \times 16 + 5$$

$$2^2 \times 2^{12} + (2^3+1) 2^8 + (2^2+2+1) 2^4 + 2^2+1$$

$$\textcircled{1} \quad \textcircled{2} \quad \textcircled{3} \quad \textcircled{4} \quad \textcircled{5} \quad \textcircled{6} \quad \textcircled{7} \quad \textcircled{8}$$

$$= 8 \text{ (1's)}$$

Ques. range of signed decimal no

$$-(2^{n-1}-1) \text{ to } +(2^{n-1}-1)$$

$$\# n=6$$

$$-(2^5-1) \text{ to } +(2^5-1)$$

$$-31 \text{ to } +31$$

① operating freq of 8085 is 3MHz but 5MHz is also available (not so famous)

② 8085 is the improved version of 8080

All instruction set of 8080 is exactly common in 8085

Instruction Set of 8080 + RIM + SIM = Inst set of 8085

upper compatibility.

speed of processor $\propto$ freq $\propto$ bit of up

INTA $\rightarrow$ acknowledge.

Simplex mode of tx.

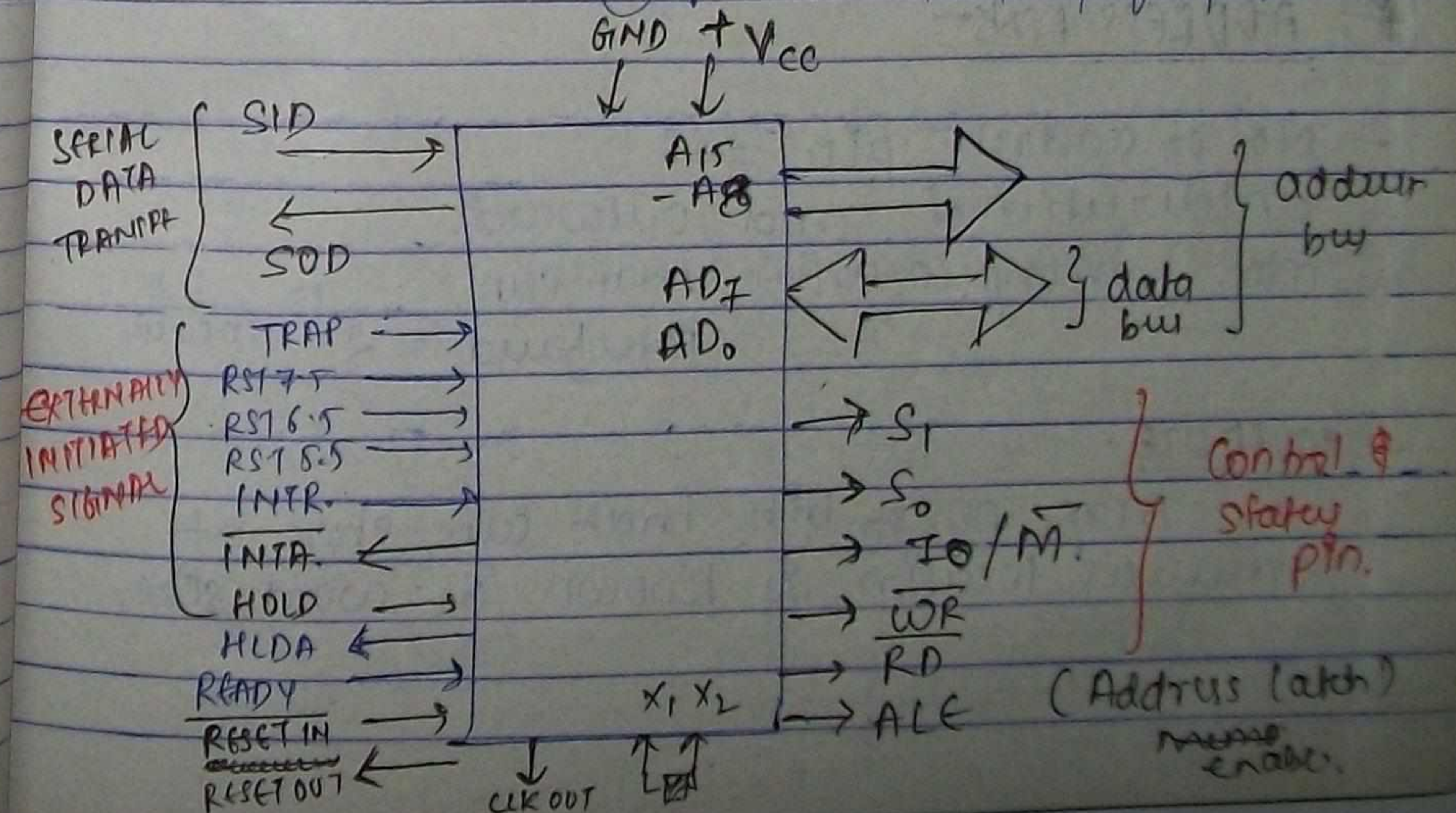
ARCHITECTURE OF 8085 -

- External arch.
- Internal arch.

EXTERNAL ARCHITECTURE -

- It is 40 pin IC
- Pin 20 $\rightarrow$ gnd. pin.
- No chip enable pin available.
- All 40 pins are divide in 6 group as per their function.

- ① Address pin.
- ② Data pin
- ③ Control & status pin
- ④ Externally Initiated signal pin
- ⑤ Serial transfer data pin
- ⑥ power supply & freq pin.



- In 8085 No. of active low pins are: 5
 $\overline{IO/\overline{M}}$, $\overline{WR}$, $\overline{RD}$, $\overline{INTA}$, $\overline{RESET}$ IN

8085 $\rightarrow$ 80th century
 8-bit
 5 - operating voltage

8085 - first complete processor.

- In 8085 No. of pins outward direction -
27

- No. of input pin = 21

ADDRESS PINS

- No. of address pin = 16
- unidirectional and outward.
- max. memory location that can interfaced = 2^{16} location

WORD SIZE

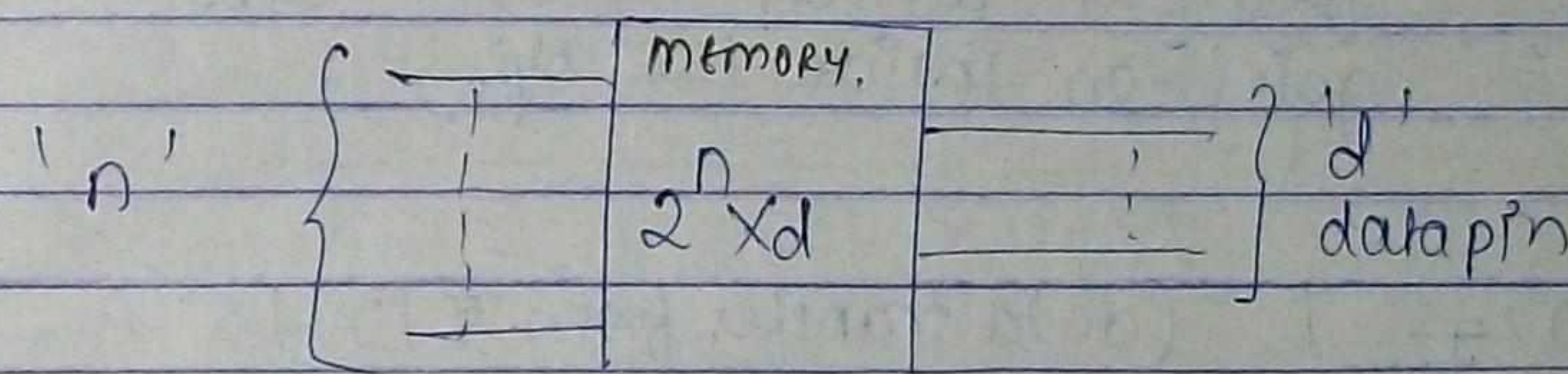
max. no. of bits that can store at particular location is known as word size.

2^{10} - Kilo
 2^{20} - Mega
 2^{30} - Giga

DATA PINS

- No. of data pin = 8
- Bidirectional

- lower 8 pin of add. pins used as data pin.



- In 8085 max. memory that can interface
 $= 2^{16} \times 8$ bits
 $= 2^{16}$ bytes
 $= 2^6$ KB
 $= 64$ KB
 $= 65536$ Bytes

Ques $2^{16} \times 8$ Ram, max. no. of add hardware pins

$$= \underbrace{16}_{\text{add}} + \underbrace{8}_{\text{data}} + V_{CC} + V_{GND} + \underbrace{(\overline{WR})}_{\text{not present in } 2^{16} \times 8 \text{ ROM}} + \overline{RD} + \overline{EN}$$

$= 29$ pins.

CONTROL & STATUS PIN -

direction is mixed direction.

$\left. \begin{matrix} S_1 \\ S_0 \end{matrix} \right\} =$ By measuring logic on that we can find out which machine cycle going on inside the μp .

$IO/\bar{M} = 1$ (data transfer from I/O)

$\rightarrow 0$ (delta transfer from memory)

$\overline{WR} = 0$ (write operation perform)

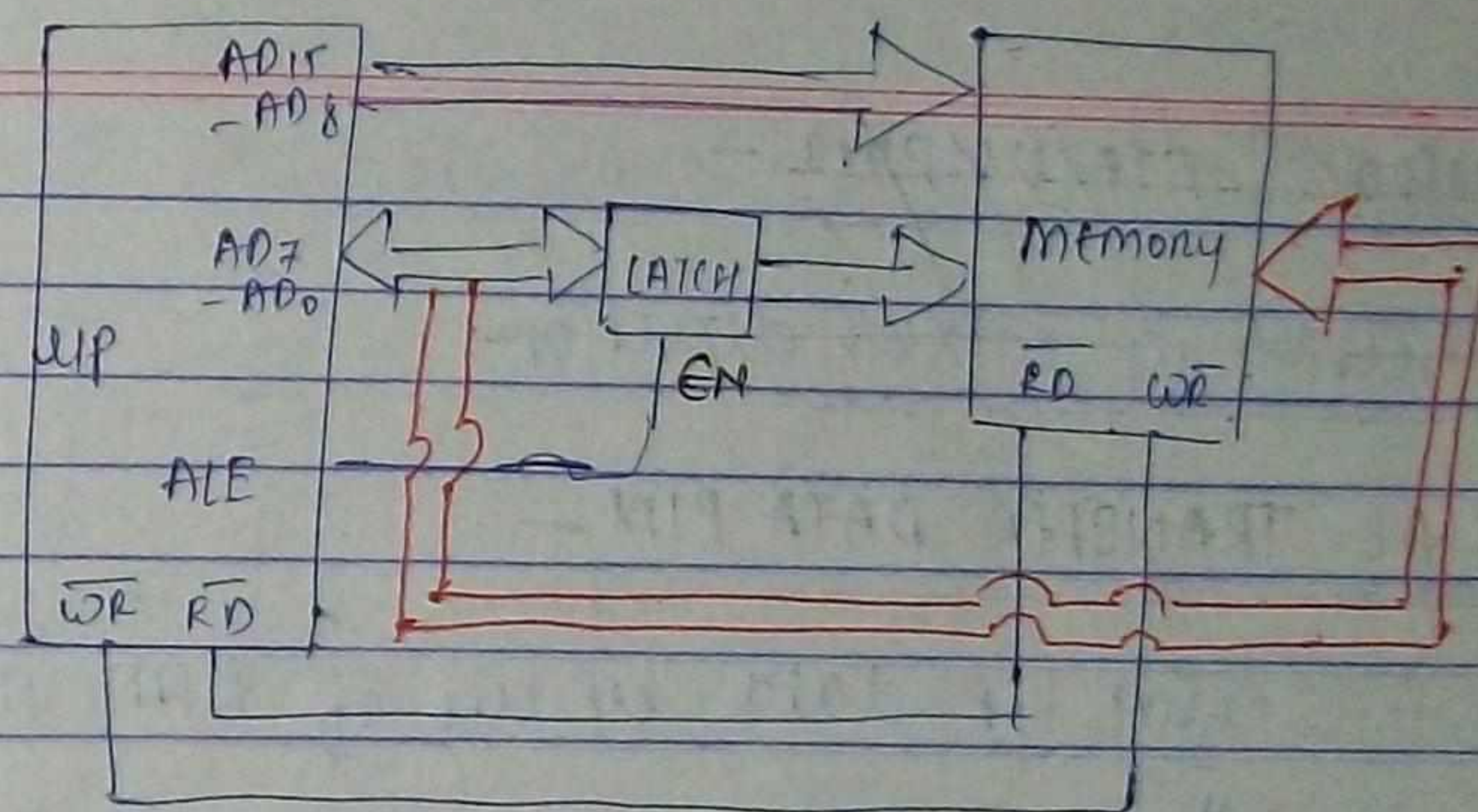
$\overline{RD} = 0$ (read operation perform)

$$\left. \begin{array}{l} \overline{WR} = 0 \\ \overline{RD} = 0 \end{array} \right\} \begin{array}{l} \text{cannot} \\ \text{perform} \\ \text{simultaneously} \end{array}$$
$$\left. \begin{array}{l} \overline{COR} = 1 \\ \overline{RD} = 1 \end{array} \right\} \text{positive}$$

ALE (address latch enable) -

ALE $\begin{cases} 1 & AD_7 - AD_0 \text{ (address)} \\ 0 & AD_7 - AD_0 \text{ (data)} \end{cases}$

~~L0 AD₇ - AD₀ (data)~~



~~A sequ~~
NOTE -

NOTE -

- Every sequential circuit may acts as latch if it is level triggered or followed its characteristic eqn at definite duration.

— Every eqⁿ. unit may act as flip-flop if it is edge triggered or followed by chara. eqⁿ at definite instant only.

— By use of Alt pin multiplexing / Demultiplexing
B/w lower 8 pins of address pins and
data pins are performed
(time division — multiplexing / demultiplexing)

(Time division — multiplexing / demultiplexing)

CONTROL & STATUS PINS -

~~Direction is mixed definition:~~

SERIAL TRANSFER DATA PIN -

SID - serial I/P data by use of RIM instruction

SOD = Serial O/P data by use of SIM instruction

RIM - read interrupt mask

SIM - set interrupt mask

POWER SUPPLY & FRQ. PIN -

$V_{CC} = +5V$

freq = 3 MHz

NOTE - To get 3 MHz clock, a 6 MHz crystal oscillator is connected b/w X_1 & X_2 and to get 3 MHz a internally divide by two circuit is used.

CLKOUT - By use of this pin, clock signal given to other interfaced peripheral for better synchronization.

EXTERNALLY INITIATED SIGNAL PINS -

Restart
software interrupt

TRAP

RST 7.5

RST 6.5

RST 5.5

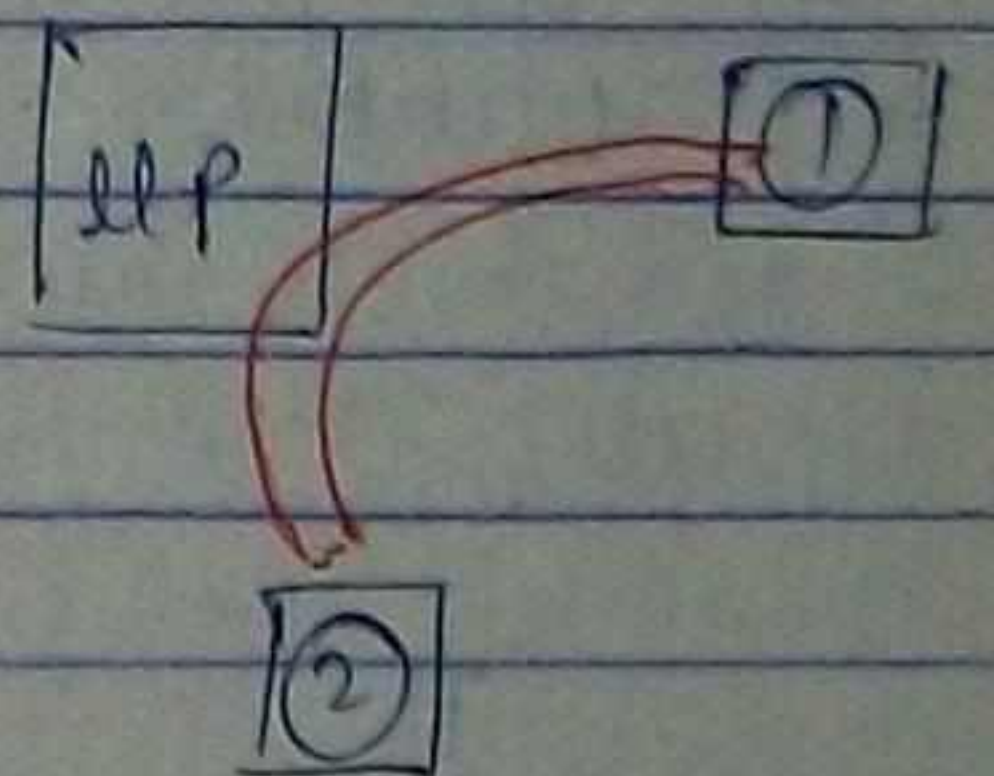
INTR

} five h/w interrupt are present in 8085

HOLD -

By use of hold pin DMA (direct memory access) like peripheral make request to up to relinquish its data pin

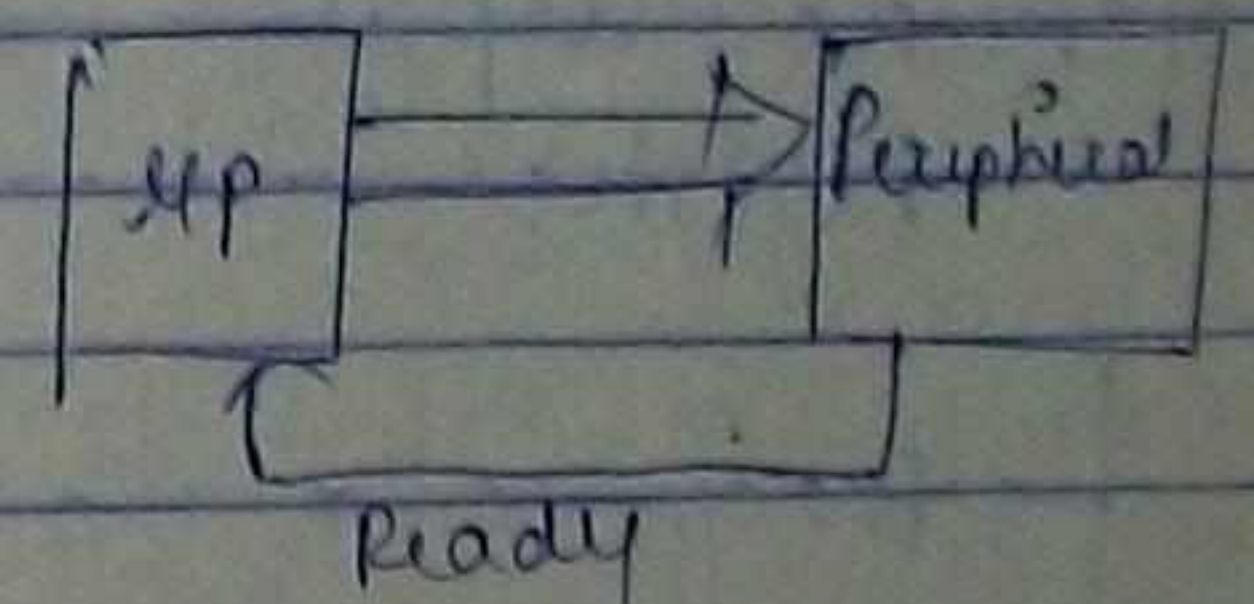
leave



slower

READY -

By use of ready pin slower peripheral make interfaced with 8085 up.



RESET IN - By use of this pin Reset command given to 8085 up for reset.

NOTE - as soon as 8085 receive reset command it generates reset out command for other interfaced peripheral.

INTERNAL ARCHITECTURE -

① ALU (arithmetic & logic unit)

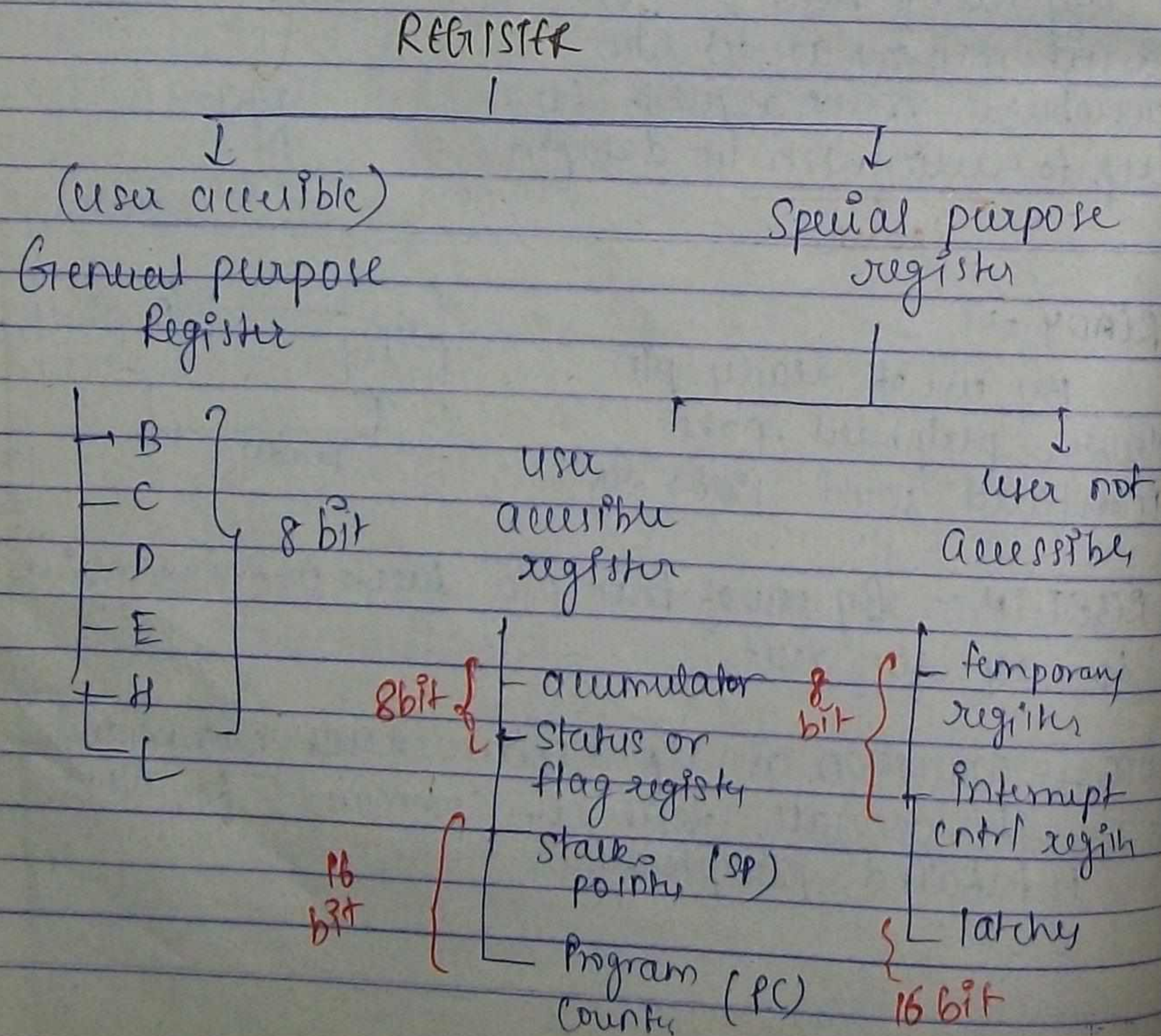
- accumulator
- Register
- Register array

② Timing & control circuit

③ Interrupt control circuit

④ Temporary Register

⑤ Latch



all computers are 2's complement.

① Sign flag (S) -

If the final result of arithmetic & logic operation D_7 bit or MSB

$$\begin{aligned} D_7 = 1 & \quad S = 1 \text{ (set)} \\ = 0 & \quad S = 0 \text{ (Reset)} \end{aligned}$$

② Zero Flag (Z) -

If in final result of A & L operation all bits are zero then
(all bits = 0) $Z = 1$ Set
otherwise $Z = \text{Reset} = 0$

③ Parity Flag (P) -

If in the final result of A & L operation no. of 1's are even then

Ex -

$$\begin{aligned} (000000) \quad P &= (\text{set}) = 1 & \text{(even 1's)} \\ P &= 0 & \text{(odd 1's)} \end{aligned}$$

- 8085 up is based upon odd parity system

$$\left[D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 \right] P$$

$$\begin{array}{r} 812 \\ 491 \\ \hline 321 \end{array}$$

④ Carry Flag (CF) -

If in the A & L operation carry is generated & discarded from MSB or D_7 bit then carry flag will set

$$\begin{aligned} C_F &= 1 & \text{(carry generated at MSB)} \\ &= 0 \end{aligned}$$

NOTE - In subtraction operation C.F. works as borrow flag.

In sub. operation borrow is taken at MSB then borrow flag will get set otherwise reset.

$$C_F = 1 \quad \text{if borrow at } D_7$$

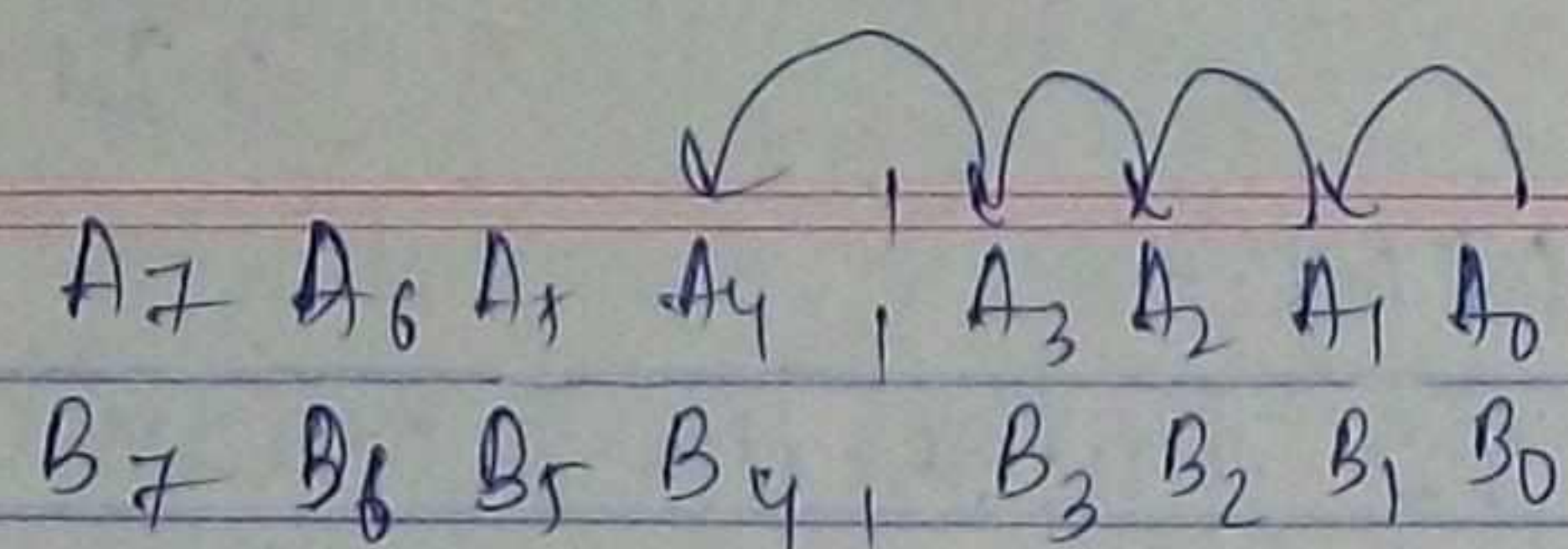
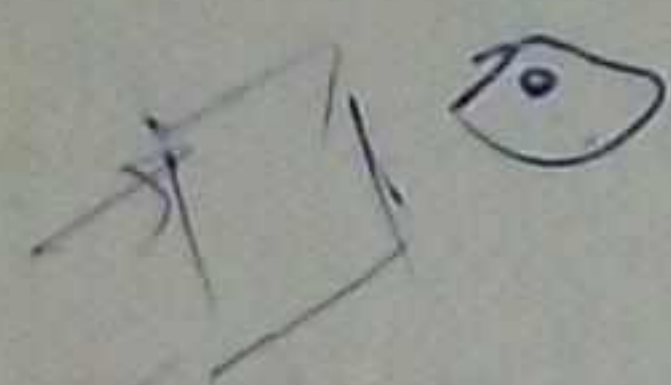
Ex

$$\begin{array}{r} 00101100 \\ 00111100 \\ \hline 11110000 \end{array}$$

⑤ Auxiliary Carry Flag (AC) -

In A & L operation carry passes from $D_3 \rightarrow D_4$ or lower nibble to upper nibble then AC flag will get set

$$\text{(AC = 1)} \quad \text{otherwise AC = 0}$$



NOTE -

All four flag can divide in two category

① that affect according the final result of A & L

Ex - S, Z, P

② that depend on cond.ⁿ generate during the execution of A & L operation

Ex - C, AC

③ Programmer can use status of four flag in programming

P, Z, S, C

but status of AC flag not used by programmer in programming.

④ Status of AC flag used internally for BCD adjustment of content of Accumulator during the execution of DAA instruction (decimal adjustment)

2/12/12

STACK POINTER (SP) -

- It is 16-bit special type of register.
- Stack pointer will always hold the address of top of stack that define in main memory.

PROGRAM COUNTER -

- It is 16-bit special type of register.
- Program counter will always store the address of next instruction to be fetch.
- If 16 bit no. stored in memory then lower order byte always store at lower order address and higher order byte store in higher order address.

NOTE -

If 16 bit no store in required pair then only special type of register can couple

BC
DE
HL

PROGRAM STATUS WORD - (PSW)

It is 16 bit user defined register
PSW = A F
↓
accum status required.

INSTRUCTION -

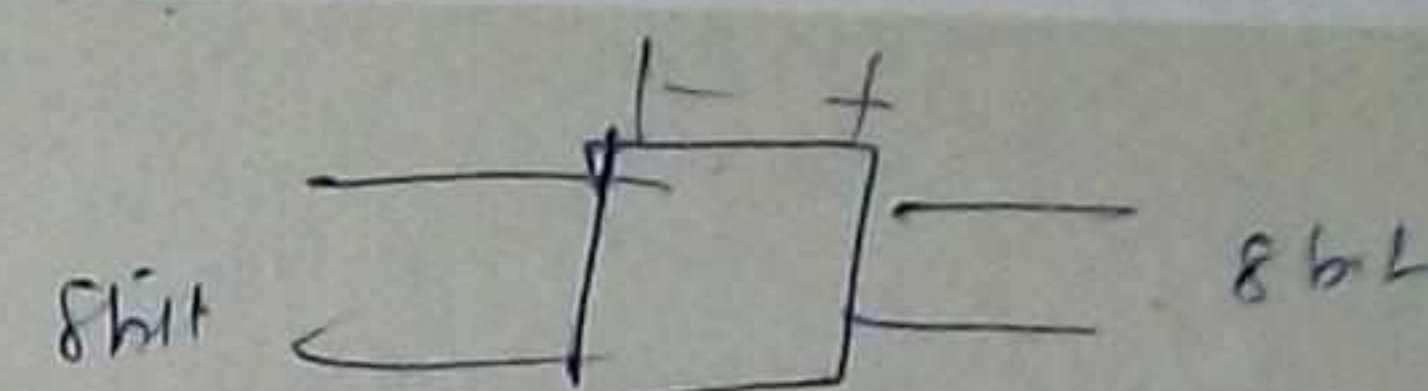
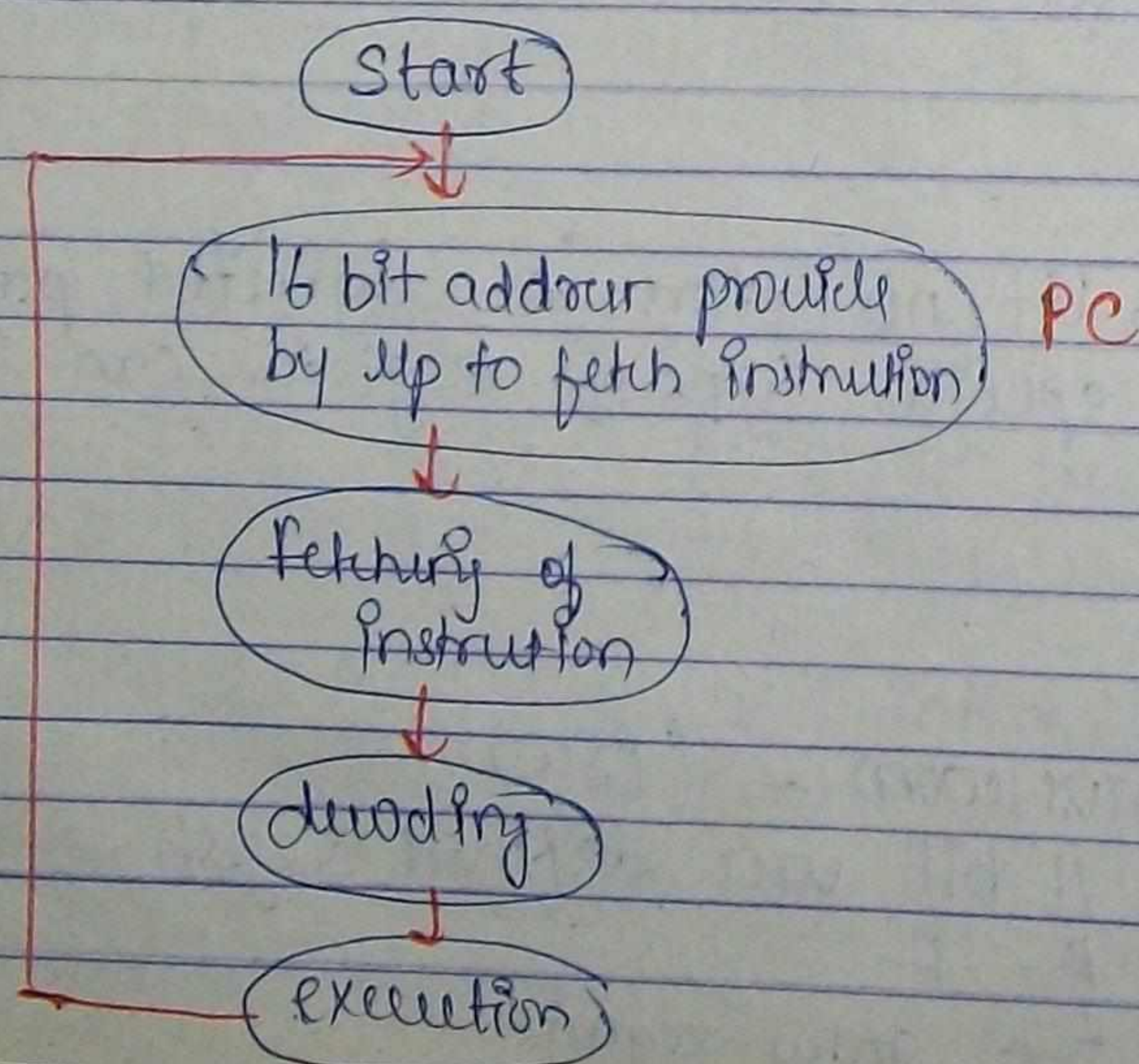
It is command that gives to step to perform a desire task.

FORMAT OF INSTRUCTION -

Table: opcode operand; description

↓ address memory location	↓ Type of Task to be perform (operational code)	↓ on which data.	↓ for user only.
------------------------------------	--	------------------------	------------------------

FLOW CHART OF EXECUTION OF PROGRAM -



- In 8085 I/p port add. are of 8 bit & o/p. port add. are of 8 bit so max. no. of I/p devices that can be connected is $2^8 = 256$ and o/p devices = $2^8 = 256$ so max. no. of I/p - o/p devices are $256 + 256 = 512$

INSTRUCTION WORD SIZE (IWS) -

memory location required to feed a instruction in memory is known as IWS.

- On the basis of IWS, instruction are of three type:

- ① 1 Byte instruction.
- ② 2 Byte instruction.
- ③ 3 Byte instruction.

① 1 BYTE INSTRUCTION -

If in the instruction, Register, Register pair or no. operand then

$$IWS = 1 \text{ Byte}$$

Ex
MOV BC
LDAX B
ADD C

① 2 BYTE INSTRUCTION -

If in the instruction, 8 bit no. is given either as add. or data then
IWS = 2 Byte

Ex MVI B 29H
IN 82H

② 3 BYTE INSTRUCTION -

If in the instruction, 16 bit no. is given either as add. or data then

IWS = 3 Byte

~~Ex~~ Ex LXI B 2015H
= LDA 1015H

* Size of machine code is of 8 bit so max. no. of command / instruction can be define is 256

MOV BC } different
MOV DE } instructions

* Total opcodes define in 8085 = 74

* By use of 74 opcode we can define 256 instruction max.

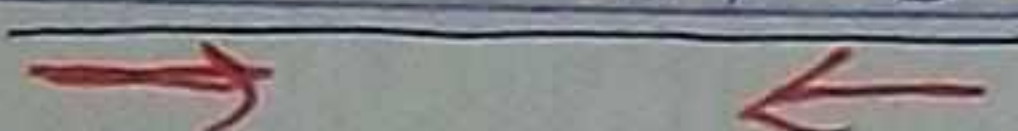
- But only 246 instructions are defined.

Instruction	- name
	- format
	- IWS
	- operation
	- Status of flag
	- addressing mode
	- machine cycle required
	- Total time

PROGRAMMING

① 8-BIT DATA TRANSFER INSTRUCTION -

① MOV R_d, R_s -



R_d = destination register.
 R_s = source register

$R_d = \{ A, B, C, D, E, H, L \} \neq M$
 $R_s = \{ A, B, C, D, E, H, L \} \neq M$

Ex - MOV BC
 MOV DE

IWS = 1 Byte

Operation -

When this instruction will execute content of R_s will pass in R_d .

$[R_d] \leftarrow [R_s]$ copy of data of R_s will pass

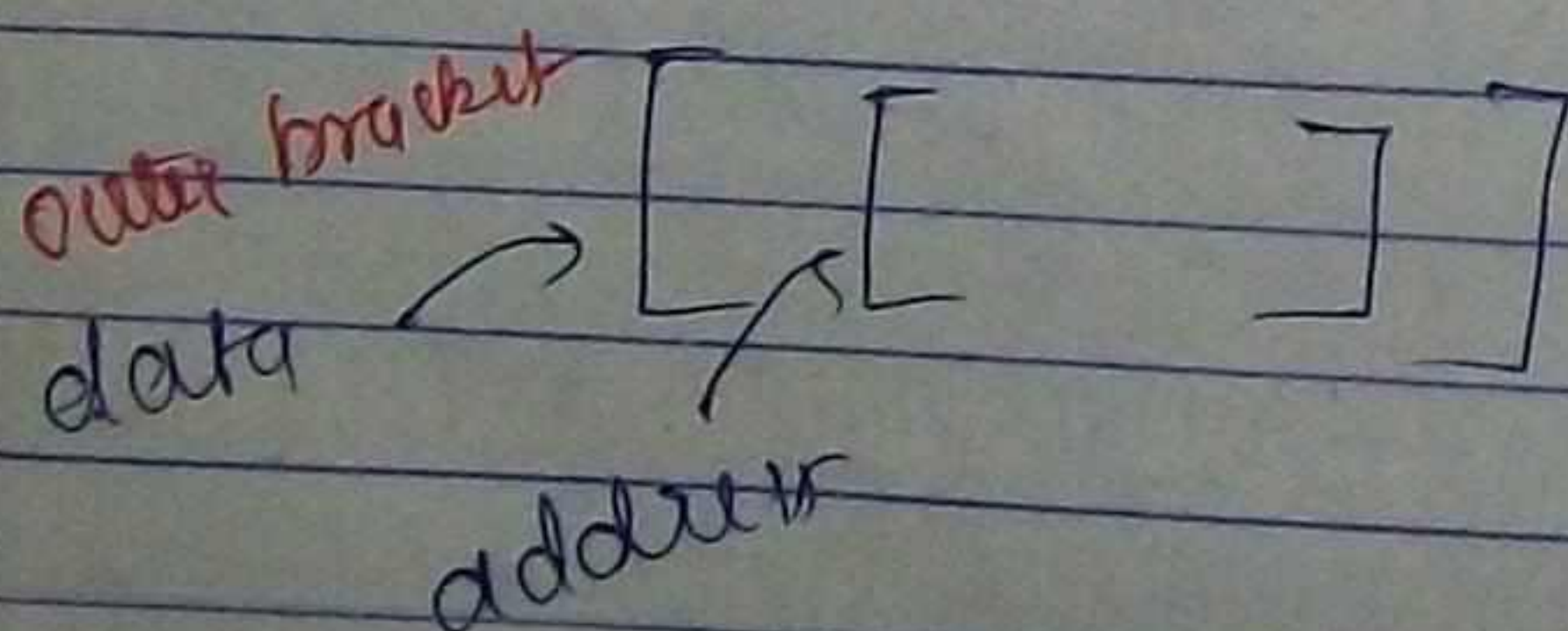
MOV B, D

$[B] = 15H$

$[D] = 15H$

$[B] = 29H$

$[D] = 15H$



M - 8 bit undefined register in main memory

opcode has last character $\rightarrow$ I $\rightarrow$ given no. always has as data
 other than I $\rightarrow$ otherwise address

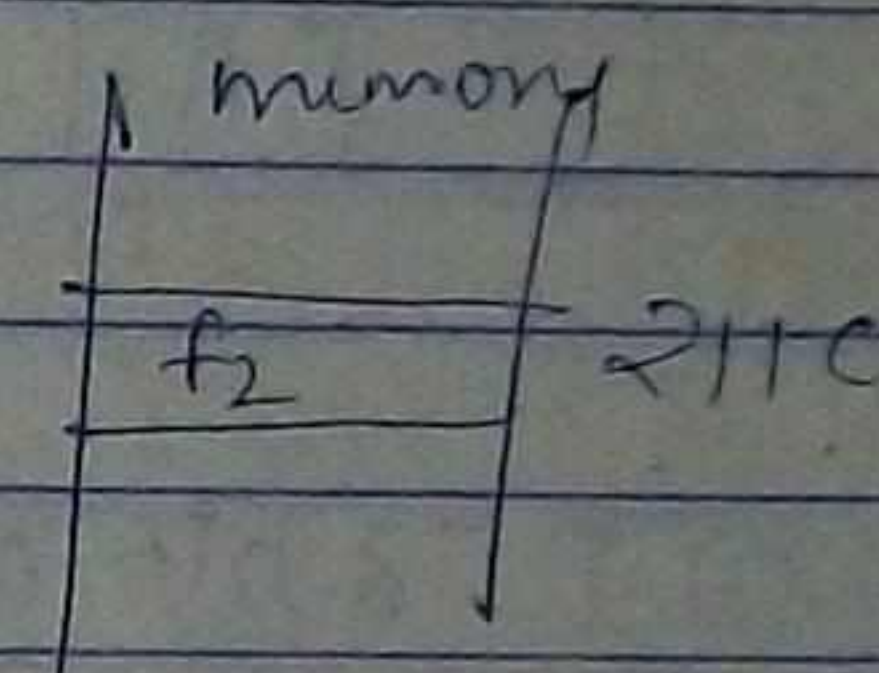
and its address is the content of HL pair.

Ex $[H] = 21H$

$[L] = 1CH$

MOV D, M

$[D] = f_2$



② MVI R, 8 bit data -

$R = A, B, C, D, E, H, L \neq M$

Operation -

8 bit data given in register instruction will pass in R.

$[R] \leftarrow 8 \text{ bit data}$

IWS = 2 Byte

Ex MOV D 21H

$[D] = 21H$

③ IN 8 bit port add. -

IWS = 2 Byte

Operation -

When this instruction will execute 8 bit data available at the given port add. in the instruction will load/store in Accumulator.

Ex - IN 20H

$[A] \leftarrow [20]$

④ OUT 8 bit address -

operation -

when this instruction will execute content of accumulator will available at 8 bit port address.

IPS = 2 Byte

MYI A 10H

10110011 Port add
B 3

OUT B3H

Seven segment display

000011000000

All instruction are data transfer instruction so status of flag will not affect.

⑤ MACHINE CONTROL INSTRUCTION -

① NOP (no operand) -

operation - when this instruction will execute it will not perform any task.

IPS = 1 Byte

② This instruction is used for creation of delay.

② HLT - no operand -

IPS = 1 Byte

operation -

when this instruction will execute further increment of PC will stop.

NOTE - HLT instruction is the last instruction of every program & after execution of HLT program execution will terminate.

Ex 1000: MOV B, C

1001: MOV D, E

1002: MYI A 05H

1004: HLT

NOP and HLT is the machine control instruction so status of flag will not affect.

ADDRESSING MODE -

- Form of address of data given in the instruction is known as addressing mode.

① Register addressing mode -

if address of data given in the form of register $\equiv$ Reg. add. mod.

Ex - MOV B, C
MOV D, E

① IMMEDIATE ADD. MODE -

If data is itself given in the instruction then $\equiv$ Immediate add. mode

Ex MOV B 12H

NOTE - If opcode has last character I then it is immediate add. mode but not vice versa.

② DIRECT ADD. MODE -

If add. of data given in the instruction then $\equiv$ direct add. mode.

Ex IN 20H

④ REGISTER INDIRECT / INDIRECT REGISTER / INDIRECT ADDⁿ MODE

If add. of data given in the form of content of accumulator then register pair or address of data given directly then $\equiv$ indirect add. mode.

Ex MOV D M

⑤ IMPLICIT ADD. MODE -

If add. of data not required in the instruction, or add. of data define in opcode $\equiv$ implicit add. mode

Ex - NOP
CMA
HLT

③ 8 BIT ARITHMETIC INSTRUCTION -

① ADD R -

R = A, B, C, D, E, H, L & M

IWS = 1 Byte

operation -

when this instruction will execute content of R will get added in content of Accum. & result will store in A

$$[A] \leftarrow [A] + [R]$$

Ex - $[A] = 20H = 0010\ 0000$

$[C] = 59H = 0101\ 1001$

ADD C $[A] = [0111\ 1001]$

If $R \neq A$ content not change
 $R = A$ content change

Ex ADD A

add. mode $R \neq m$ = Register addⁿ mode
 $R = m$ = Indirect mode.

② ADI 8 bit data

IWS = 2 Byte

Operation -

8 bit data given in the instruction will get added in accumulator & result will store in Acc.

Add. mode $\rightarrow$ Immediate add. mode.

③ SUB R -

$R = A, B, C, D, E \text{ \& } M$
 H, L

Operation -

Content of R will get subtracted from content of accumulator and result will store in accumulator

IWS = 1 Byte

$$[A] \leftarrow [A] - [R]$$

$R \neq m$ = Register addⁿ mode.
 $R = m$ = Indirect addⁿ mode.

Clear work 1 line instruction that make content of Acc. 00H regardless of previous status.

- ① SUBA, ② MVI A 00H ③ ANI 00H ④ XRAA

④ SUI 8 bit data -

IWS = 2 Byte

Operation -

8 bit data given will subtract from accuml. and result will store in accuml.

$$[A] \leftarrow [A] - 8 \text{ bit data.}$$

address $\rightarrow$ Immediate add. mode.

⑤ INR R -

$R = A, B, C, D, E, H, L, \text{ \& } M$

Operation -

Content of R will increase by 1 and result will store in R

$$[R] \leftarrow [R] + 1_{LSB}$$

Ex - MVI D 25H $[D] = 00100101$
 INR D $[D] = 00100110$

$R \neq m$ Register add. mode

$R = m$ Indirect Reg. add mode.

③ DCR R -

R = A, B, C, D, E, H, L & M

operation

Content of R will decrease by 1 and result will store in R.

$$[R] \leftarrow [R] - 1_{LSB}$$

NOTE -

- ADD, ADI, SUB, SVI will affect status of all flag.
- INR & DCR will affect only status of four flag. S, Z, P, AC.
- INR & DCR will not affect status of carry flag.

Ex - MVI FFH
INR D

[D] $\overbrace{11111111}^{8\text{ bits}}$
 + 1
 10000000

S	Z	AC	P	CY
0	1	1	1	1

Parity
Status

⇒ INSTRUCTION CYCLE, MACHINE CYCLE AND T-STATE.

INSTRUCTION CYCLE -

Total time required for the execution of one instruction is known as instruction cycle.

- In one instruction cycle, one or more than one machine cycle can exist.

MACHINE CYCLE -

During the execution of instruction different type of operations are performed that is known as machine cycle.

In 8085 six type of machine cycles are present.

- ① opcode fetch^{machine} cycle (T/s)
- ② machine code fetch machine cycle

Time required by execution to fetch (read) machine code or opcode from memory is known as opcode machine fetch cycle.

MVI D
machine cycle

NOP
machine cycle

IF = 4T
IS = 6T
fetch store

processor can read or execute 8 bit data at a time.

TRISTATE -

$$T = \frac{1}{f_{\text{operating freq}}} \text{ sec.}$$

⇒ Machine code regarding inst. required 6 T states are.

CALL & RETURN
RESTART
INX & DCX
SPHL & PCHL
PUSH

(CRIST)

NOTE - If it is nothing but read operation from memory.

- Opcode fetch machine cycle is the first machine cycle of every instruction.

② MEMORY READ MACHINE CYCLE (R) -

Total time required by execution to read 8 bit data from memory.

$$R = 3T$$

③ MEMORY WRITE MACHINE CYCLE (W) -

Total time required by execution to store 8 bit data in memory.

$$W = 3T$$

④ INPUT READ MACHINE CYCLE (I) -

Total time required by execution to read 8 bit data from I/P port.

$$I = 3T$$

⑤ O/P WRITE MACHINE CYCLE (O) -

Total time required by execution to make available 8 bit data at O/P port.

$$O = 3T$$

⑥ BUS ^{IDLE} MACHINE CYCLE (IB) -

Time duration during the bus of μp will be in idle condition, at the time of execution of some instruction is known as bus idle machine cycle.

$$IB = 3T$$

Ex DAD instruction

* INSTRUCTION	MACHINE CYCLE	T-STATE
MOV B, C	F	4T
MOV D, M	FR	7T
MOV M, A	FW	7T
MVI B, 29H	FR	7T
MVT M, 10H	FRW	10T
INR B	F	4T
INR M	FRW	10T
ADD C	F	4T
ADI 29H	FR	7T
SUB M	FR	7T
IN 82H	FRI	10T
OUT 12H	FRO	10T
NOP	F	4T

because machine cycle.

HLT F 4T/5T

⇒ 8 BIT LOGICAL INSTRUCTION -

(A) AND OPERATION -

(1) ANA R -

(PWS = 1 Byte)

R = A, B, C, D, E, H, L & M

Operation -

[R] will get AND operation with [A] and bit by bit and result will store in [A].

Ex [A] = 29H

A = 0010 1001

[B] = 10H

B = 0001 0000

0000 0000

ANA B

R ≠ M - Register addressing mode

R = M - Indirect addressing mode

(only fetch)
T = 4T

↓
(both Fetch & Read)
FR T = 7T

ANA A → result remains same.

② ANI 8 bit data -

IWS = 2 Byte

Operation -

8 bit data get AND operation with [A] and result will store in [A]

add. mode $\rightarrow$ Immediate

③ OR OPERATION -

① ORA R -

$R = A, B, C, D, E, H, L, M$

Operation -

Content of R will get OR operation with content A. All bit by bit and result will store in A

add. mode $R \neq M$ Register add. $\equiv F$
 $R = M$ Immediate R $\equiv FR$

② ORI 8 bit -

IWS = 2 Bytes

Operation -

8 bit data get or'ed with Acc and stored in Acc

Add mode $\rightarrow$ Immediate $\equiv FR$

★
 ③ EXOR OPERATION -

① XRA R -

$R \neq M \rightarrow$ Register $\equiv F$
 $R = M \rightarrow$ Immediate $\equiv FR$

Operation - content of

$[R] \leftarrow [A] \text{ EXOR } [R]$
 bit by bit

Ex $[A] = 56H$
 $[D] = 25H$

XRAD

$A = 01010110$
 $D = 00100101$
 $\oplus 1100111$

XRA A $\rightarrow$ result zero

② XRI 8 bit data

IWS = 2 Byte

$[A] \leftarrow [A] \text{ EXOR } [8 \text{ bit data}]$
 bit by bit

Add. mode $\rightarrow$ Immediate

$\equiv FR$

≠ CMA NO OPERAND.

IWS = 1 BYTE

Operation —

$[A] \leftarrow [\bar{A}]$ bit by bit.

Ex $[A] = 1FH \rightarrow 00011111$

CMA $[A] \rightarrow 1110000$

Implicit add. mod. Math's = F

FLAG —

	S	Z	Ac	P	CY
ANA	✓	✓	1	✓	0
ANI	✓	✓	1	✓	0
ORA	✓	✓	0	✓	0
ORI	✓	✓	0	✓	0
XRA	✓	✓	0	✓	0
XRI	✓	✓	0	✓	0
CMA	X	X	X	X	X

✓ = As per result

X = not affect

0 = Reset

1 = Set

NOTE —

- CMA will not affect any flag
- AND, OR, EX-OR operation always reset carry flag.
- AND operation always set the auxiliary carry flag
- OR and EXOR operation always reset the auxiliary carry flag
- All other flag for OR, AND, EX-OR as per usual.

Ques, FF00: MVI A 23H 2 byte
 FF02 MVI B 32H 2 byte
 FF04 XRA B 1 Byte
 FF05 ADI 88H 2 Byte
 FF07 HLT. FF08

after execution content of program counter, PSW = ?, B = ?

Solⁿ: PC = FF08

If X = 0

PSW = AF = 00010001 = 84
 $\begin{matrix} S & Z & X & AC & X & P & X & CY \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{matrix}$
 $\begin{matrix} A = 00100011 \\ B = 00110010 \\ [A] = 00010001 \\ 88H = 10001000 \end{matrix}$
 $\begin{matrix} 00010001 \\ 10001000 \\ \hline 10011001 \end{matrix}$

Ques. MVI A, 2AH
 ADD A
 ORI AFH
 INR A
 CMA
 HLT

after execution of program status of flag.

Solⁿ:

	S	Z	X	AC	X	P	X	CF
	0	0		1		0		0
	1	0		0		1		0
	0	1		1		1		0
	0	1		1		1		0

[A] —	0	0	1	0	1	0	1	0
	0	0	1	0	1	0	1	0
ADDA —	0	1	0	1	0	1	0	0
	0	1	0	1	0	1	0	0
ORI AFH →	1	0	1	0	1	1	1	1
	1	0	1	0	1	1	1	1

INR A + 1

	0	0	0	0	0	0	0	0
--	---	---	---	---	---	---	---	---

CMA

	1	1	1	1	1	1	1	1
--	---	---	---	---	---	---	---	---

[A]

Before.

Ques. After arithmetic operation status of flag register = BBH
 So after the after arith. operation may content of Acc may be,

BBH = S Z X AC X P X CF

75H 0 1 1 1 0 1 0 1

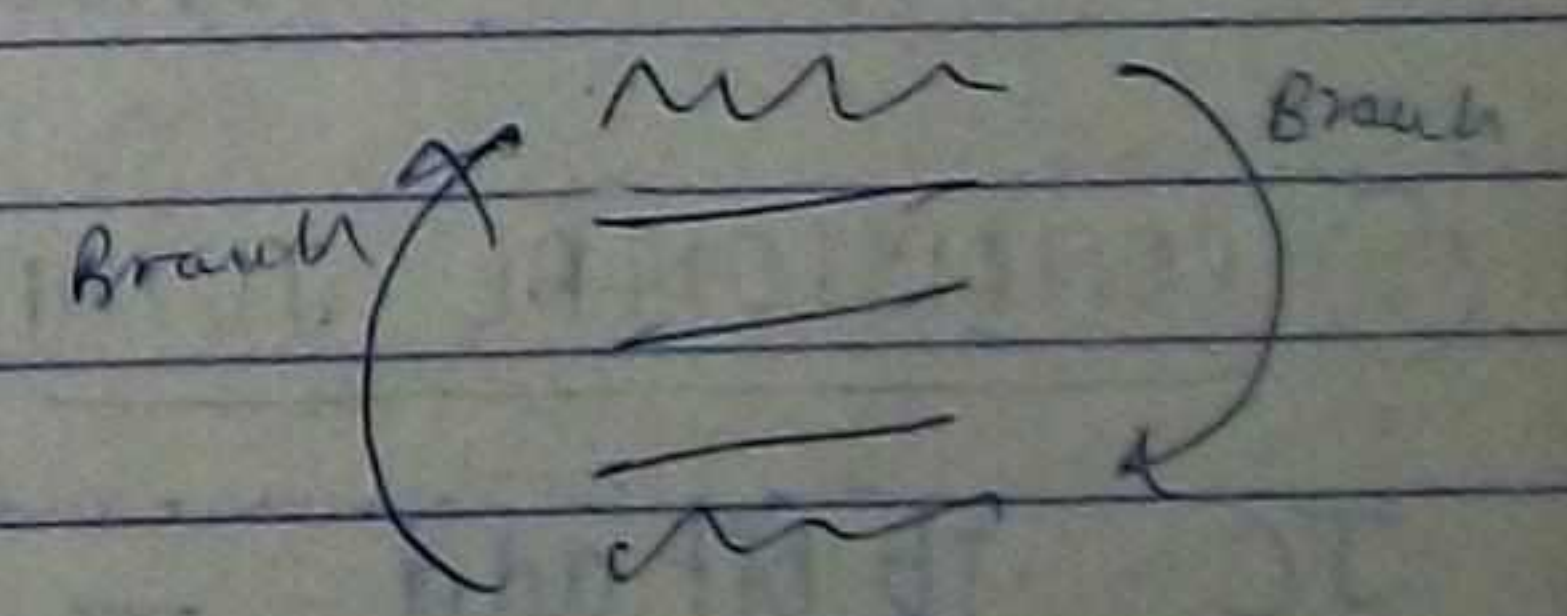
65H 0 1 1 0 0 1 0 1

DBH 1 1 0 1 1 0 1 1

✓ B67 1 0 1 1 0 1 1 0 odd parity

⇒ BRANCH OPERATION —

— loop is the special case of branch operation



— In 8085 three instructions are available for branch operation

- ① JUMP INST
- ② CALL INST
- ③ RESTART INST

3/12/11

* JUMP INSTRUCTION -

It is of two type

- ① unconditional jump
- ② conditional jump

① UNCONDITIONAL JUMP -

JMP 16 bit add

operation -

execution / control will jump at 16 bit vector location given in the instruction.

machine cycle

⇒ FRR

PER

1006 Jump

1007 13H

addressing → Immediate add. mode,

1008 20H

② CONDITIONAL JUMP -

JC	16 bit add	→	executes Jump	if	Carry = 1
JNC	"	→	"	"	Carry = 0
JZ	"	→	"	"	Zero = 1
JNZ	"	→	"	"	Zero = 0
JP	"	→	"	"	Sign = 0 (+ve)
JM	"	→	"	"	Sign = 1 (-ve)

JPE 16 bit → executes Jump if P = 1
JPO " → " " " P = 0

operation -

If condition satisfy then jump operation will execute otherwise skip it

1WS = 3 BYTE

Not satisfy.

carry

1001

1002

1003

JP 2015

CONDITIONAL JUMP

Satisfy M.C → FRR

not satisfy M.C → FR*

NOTE -

It is data transfer instruction, so status of flag will not affect

⇒ 16 bit data transfer instruction -

① LXI Rp 16 bit data Rp = register pair

Rp = BC } = B
 DE } = D
 HL } = H
 SP

IWS = 3 Bytes

operation -

16 bit data will store in Register pair

Add. mode → immediate add.

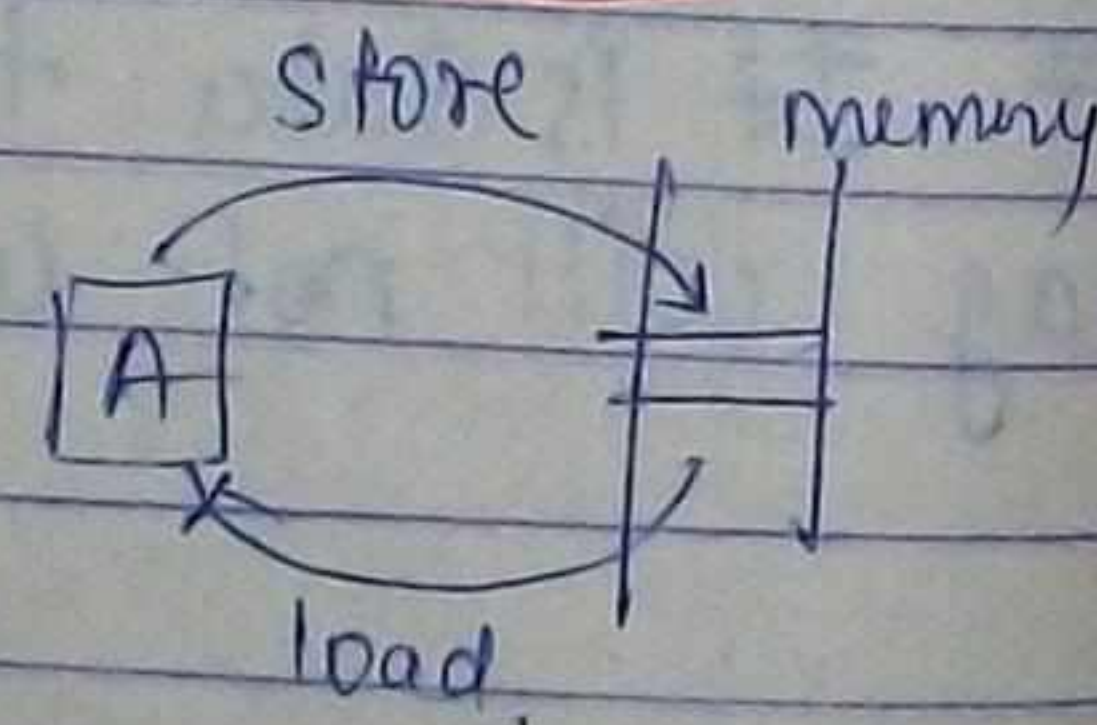
Machine cycle → FRR

② LDA 16 bit add. -

operation -

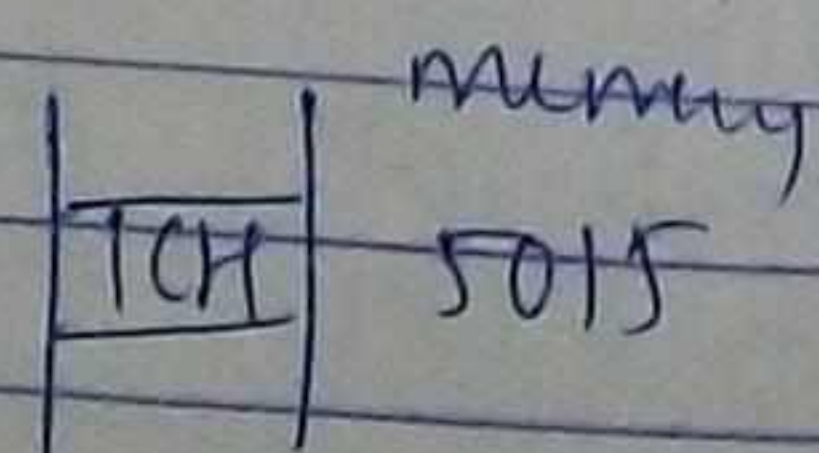
8 bit data available at the 16 bit add. given in the instruction will load in accumulator.

IWS = 3 Byte



$[A] \leftarrow [16 \text{ bit add.}]$

eg LDA 5015 H
[A] = 1CH



Machine cycle → FRRR

add. mode → direct add. mode

③ STA 16 bit -

operation -

Content of accumulator will store at 16 bit memory location given in the address

$[A] \Rightarrow [16 \text{ bit add.}]$

Add. mode → direct add.

Machine cycle → FRRW

④ LDAX Rp

operation -

Content of Rp will load in accumulator

$[A] \leftarrow [Rp]$

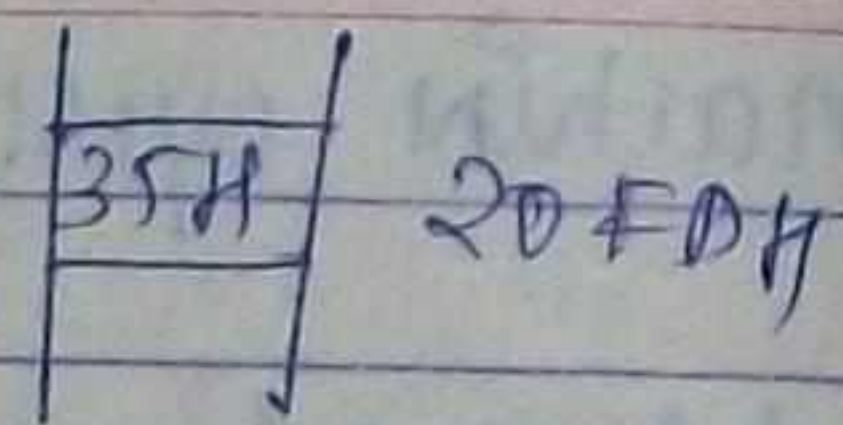
- 8 bit data available at 16 bit memory address that is content of Rp will load in Accumulator

F 1001 LDA
 R 1002 15 H
 R 1003 50 H
 R 1004

IWS = 3 Byte

IWS = 1 Byte

Ex. LXID 20FDH
LDAX D
[A] = 35



$$[A] \leftarrow [DE]$$

$$[20FD]$$

load address

Add. mode - Indirect register add.

Machine cycle - FR

⑤ STAX R_p -

operation -

content of accum. will store at the memory location i.e. content of R_p

$$[A] \rightarrow [R_p]$$

NOTE -

for LDAX and STAX only

$R_p = BC$
 $= DE$ } only.

HL $\rightarrow$ used for M

NOTE -

all above instruction are data transfer instruction
so status of flag will not affect.

$\Rightarrow$ 16 bit arithmetic instruction

① INX R_p -

1WS = 1 Byte

$R_p \rightarrow \begin{matrix} BC \\ DE \\ HL \end{matrix}$

operation - content of R_p will increase by 1
and result will store in R_p

$$[R_p] \leftarrow [R_p] + 1$$

Add. mode $\rightarrow$ Register add.
M.C $\rightarrow$ S (6T)

② DCX R_p -

1WS = 1 Byte

$R_p \rightarrow BC, DE, HL$

operation -

$$[R_p] \leftarrow [R_p] - 1$$

Add. mode $\rightarrow$ Register add.
M.C $\rightarrow$ S.

NOTE - INX and DCX are arithmetic operation but they will not affect status of any flag

③ DAD Rp -

Rp = BC, DE, HL

1WS = 1 Byte

operation -

$$[HL] \leftarrow [HL] + [Rp]$$

add. mode → register add. mode.

M. cycle → F B B

3T 6T → delay no instruction flow in BUS.

NOTE -

DAD instruction affect only one flag that is carry flag.

HL = 20 15H
DE = 15 13H

★ ★

⇒ 8 BIT LOGICAL ROTATIONAL INSTRUCTION -

- These instruction execute on basis of content of accumulator.

- when these type of instruction execute content of accumulator shift by one bit either left or right as per instruction.

A → 8 bit
PC → 16 bit
SP → 16 bit

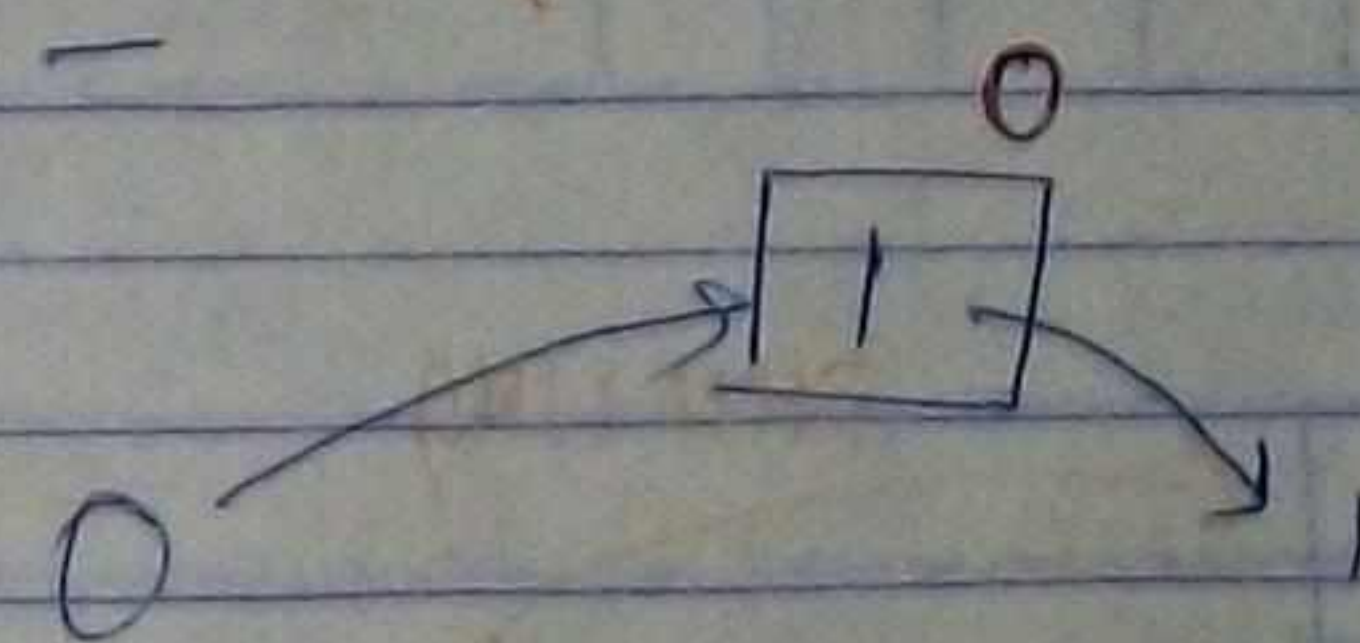
① RLC no operand → Content of Acc. rotate Left by 1 bit without carry

② RAL " → " with carry

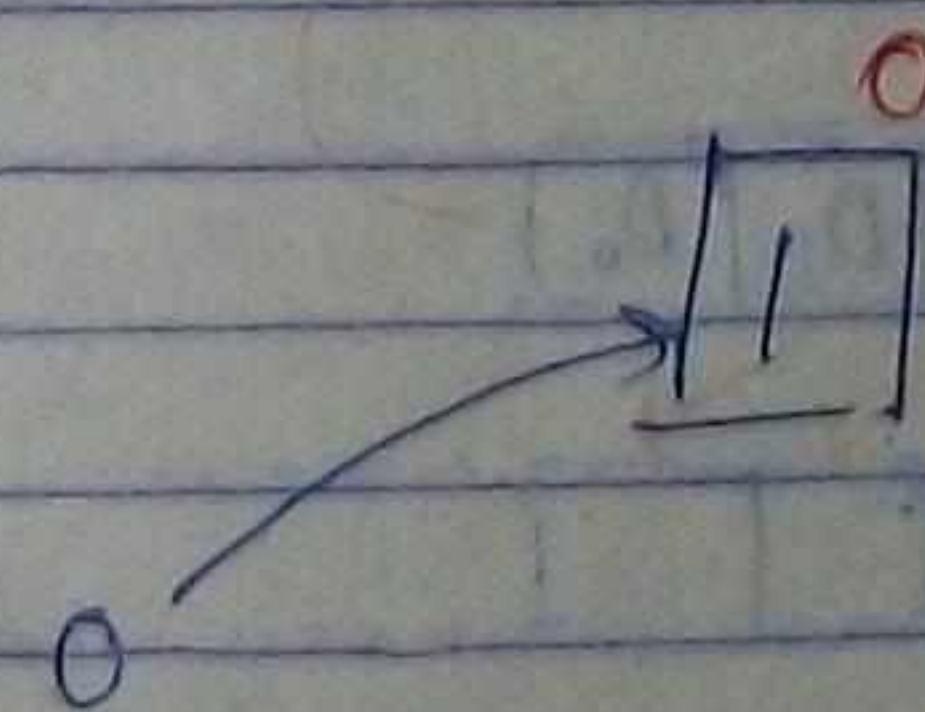
③ RRC " → content of Acc. rotate Right by 1 bit without carry

④ RAR " → " with carry

NOTE -



SHIFTING OR SWAPPING



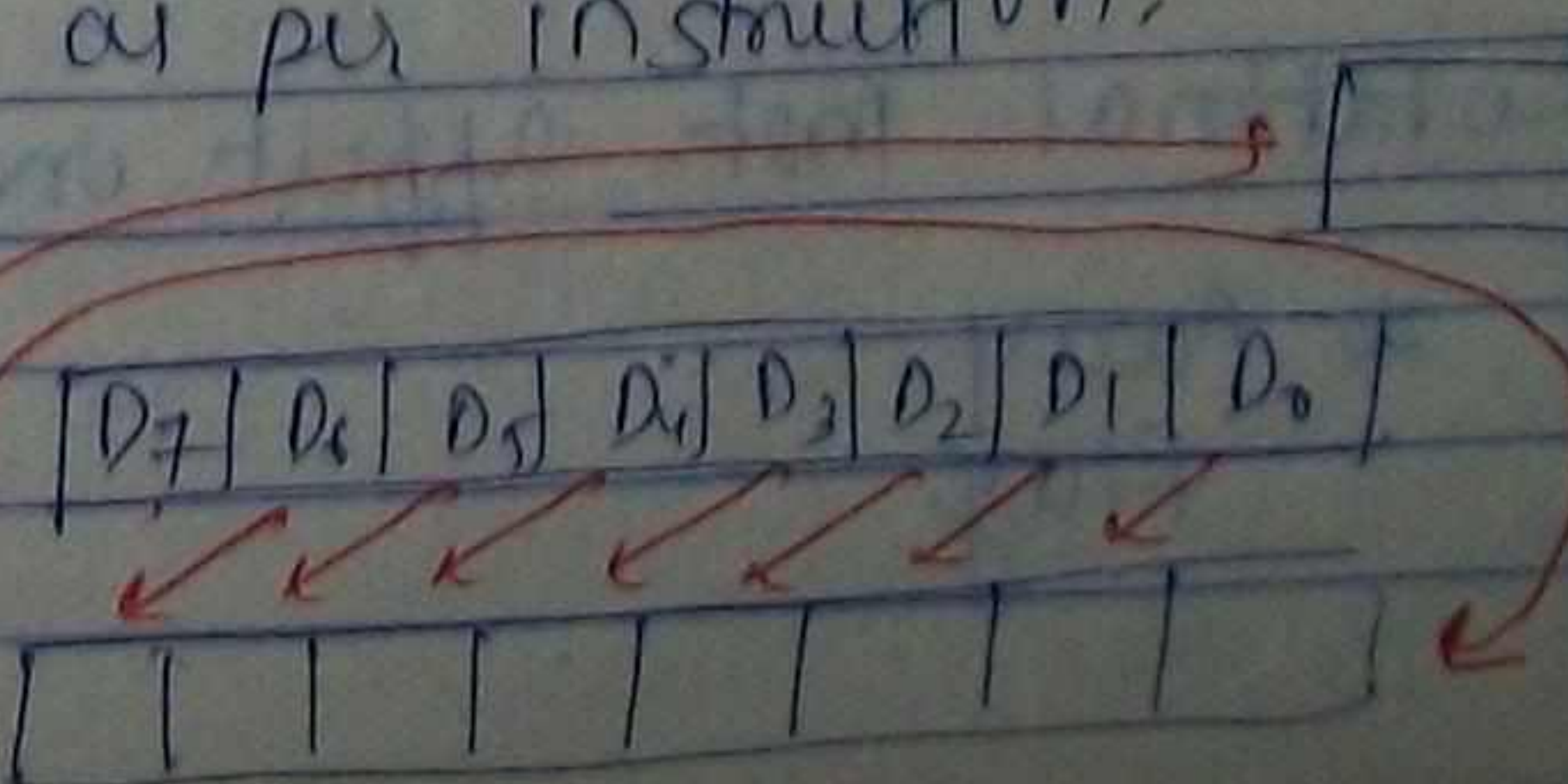
OVERWRITE

1WS = 1 Byte

operation -

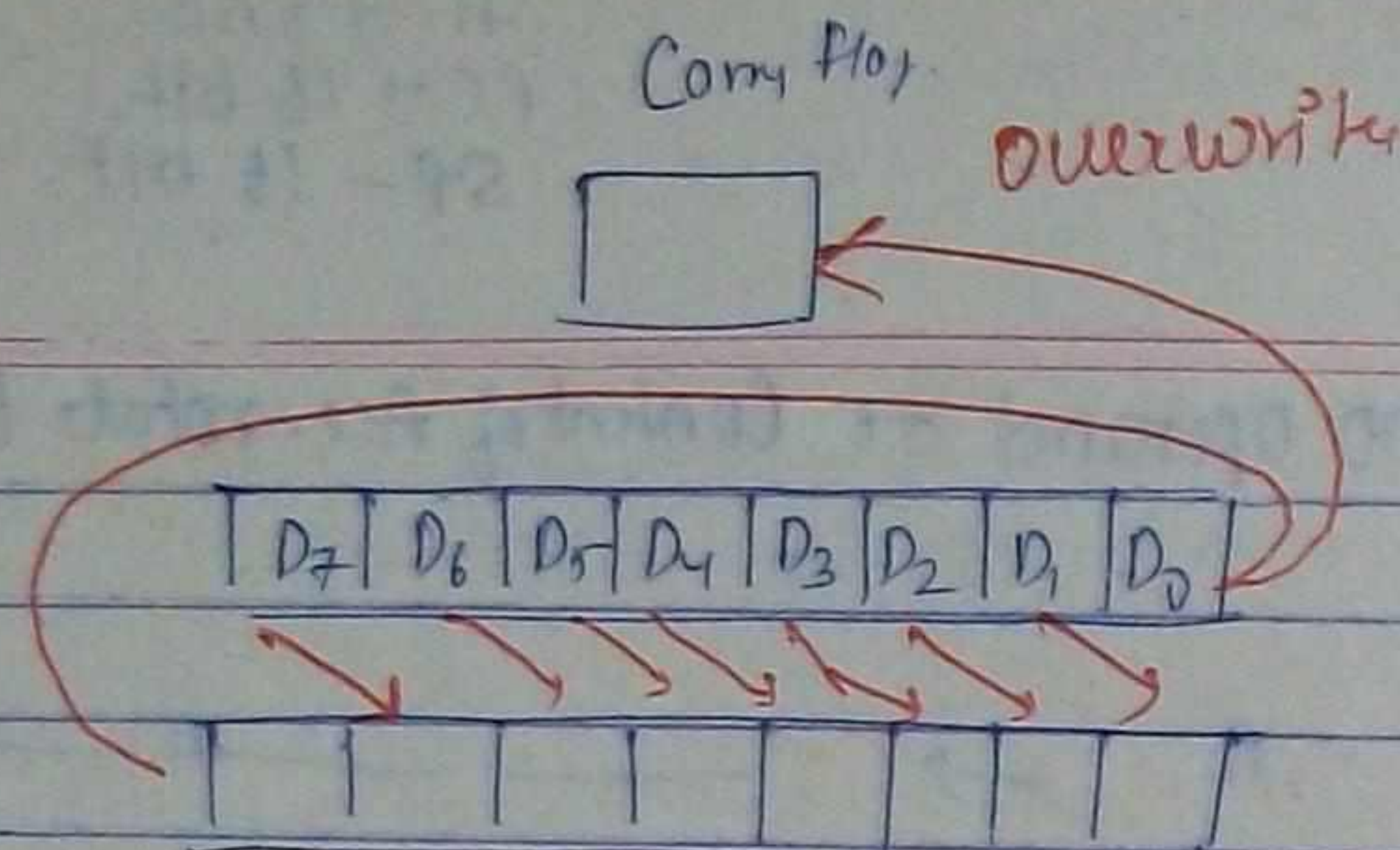
when these instruction will execute content of accumulator will rotate left or right by one bit as per instruction.

① RLC -

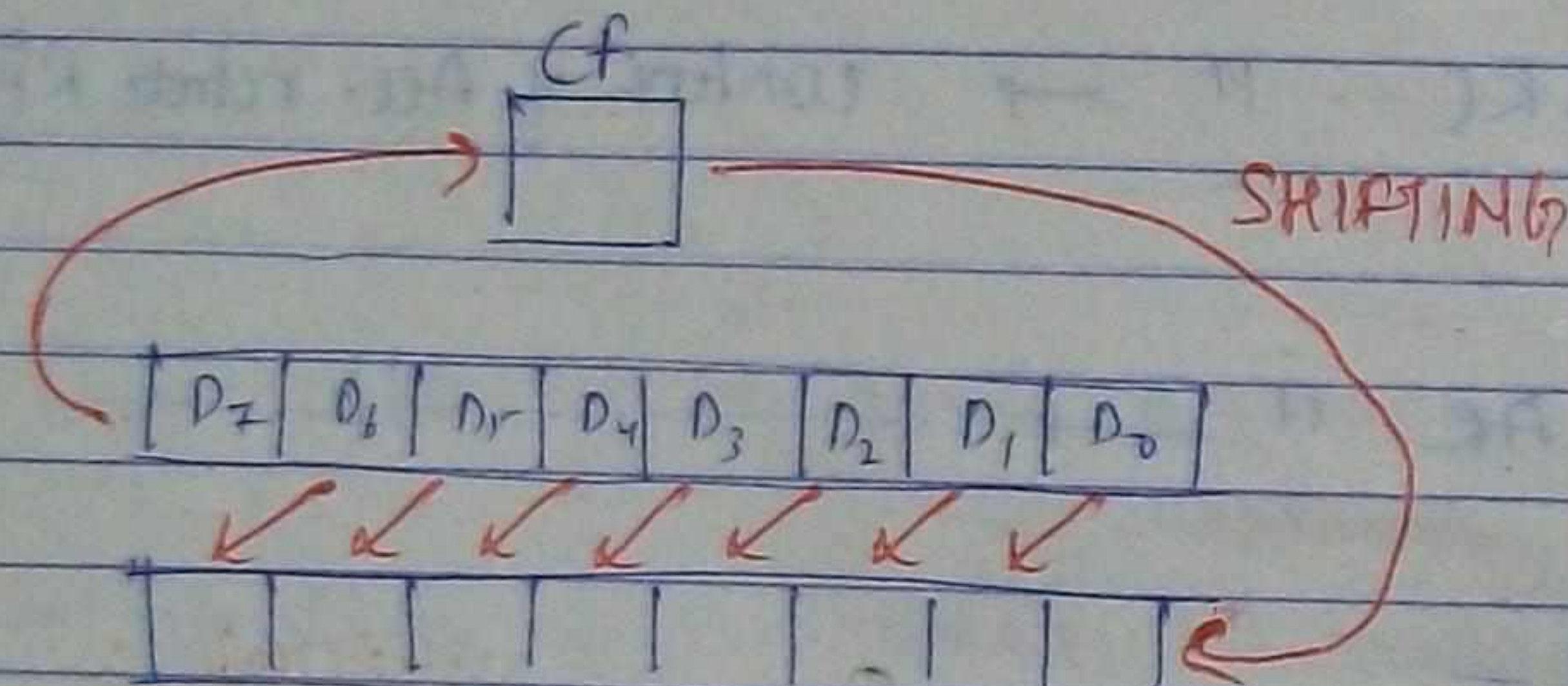


overwrite

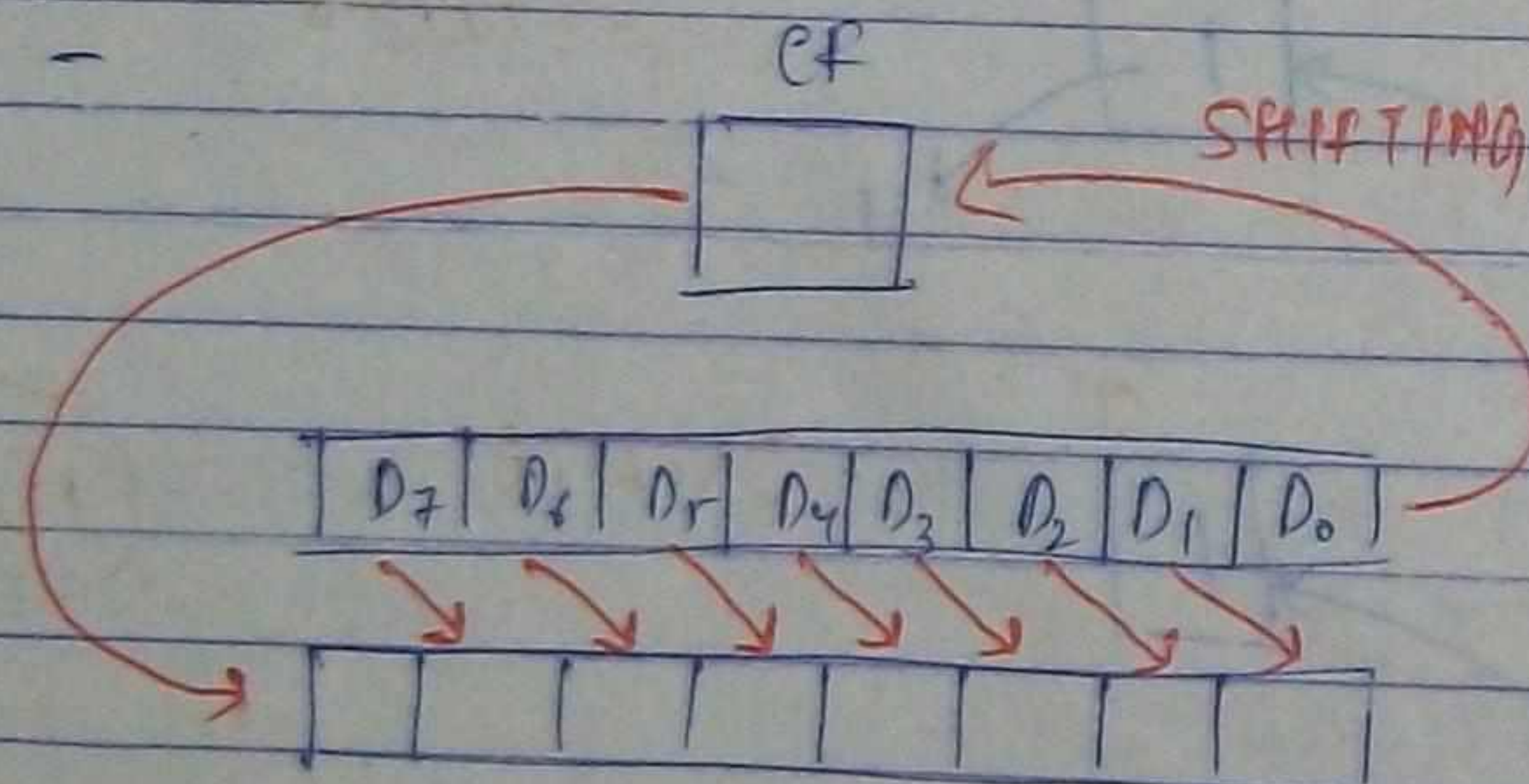
② RRC -



③ RAL -



④ RAR -



add. mode → implicit add.
Maths cycle → F

NOTE -

- logical Rotational inst. affect only one flag = carry flag.

Ques. MVI A FOH
ORA A
loop: INR A
JNC loop:
HLT.

flaggy method same address

Solⁿ: [A] = FO = 11110000

[A] = FO = 11110000, CF = 0

+1
INRA [A] = 11110001, CF = 0

+1
JNC [A] = 11110010, CF = 0

11111111
+1
00000000, CF = 0

Infinite loop.

Ques. 1000H MVI A A1H
1002 LXI H 1007H
1005 SUB M (1007)
1006 OUT 05H
HLT

after the execution of above program display at o/p port

① A1H

② 9CH

③ Can't determine

o/p port add
= 5EH

Solⁿ: mvi [A] ← A1H
LXI [H] ← 10H
[L] ← 07H
SUB M [A] = 1010 0001
[M] = 0000 0901

data at
(007H)

(05H)

[A] = $\frac{1001}{9} \frac{1100}{C}$

Ans. LXI H, 2000H
LDA 2002H
XRA M
MOV E, A
MVI D, 20H
LDAX D
OUT 01H
HLT

memory ad.	content
2000H	00H
2001H	01H
2002H	02H
2003H	03H

after execution of this program display
at o/p port.

Solⁿ: H = 20H
L = 00H

[A] = 02H

[A] = 0000 0010

[M] = 0000 0000

[A] = 0000 0010 H

[A] = 02H

[E] = 02H [A] = 02H

[D] = 20H [A] ← [DE]

[A] = 02H

o/p port add = 02H

Ans. MVI A, 89H
ORA A
RLC
RAL
ADI 2CH
RAR
HLT

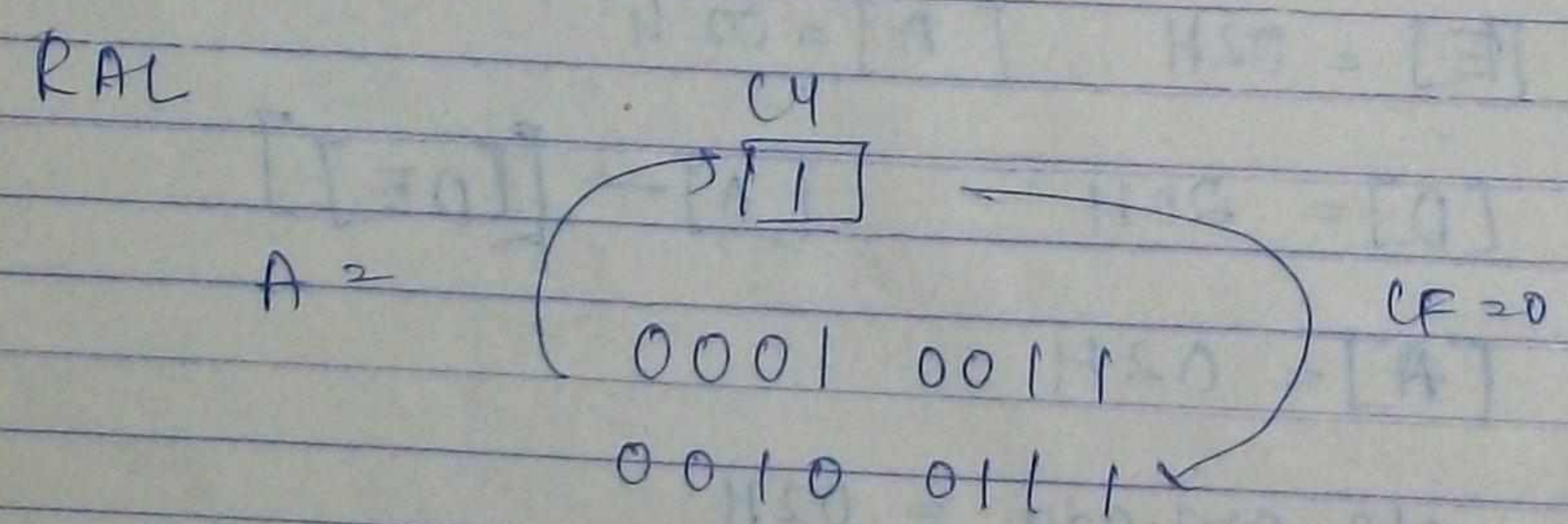
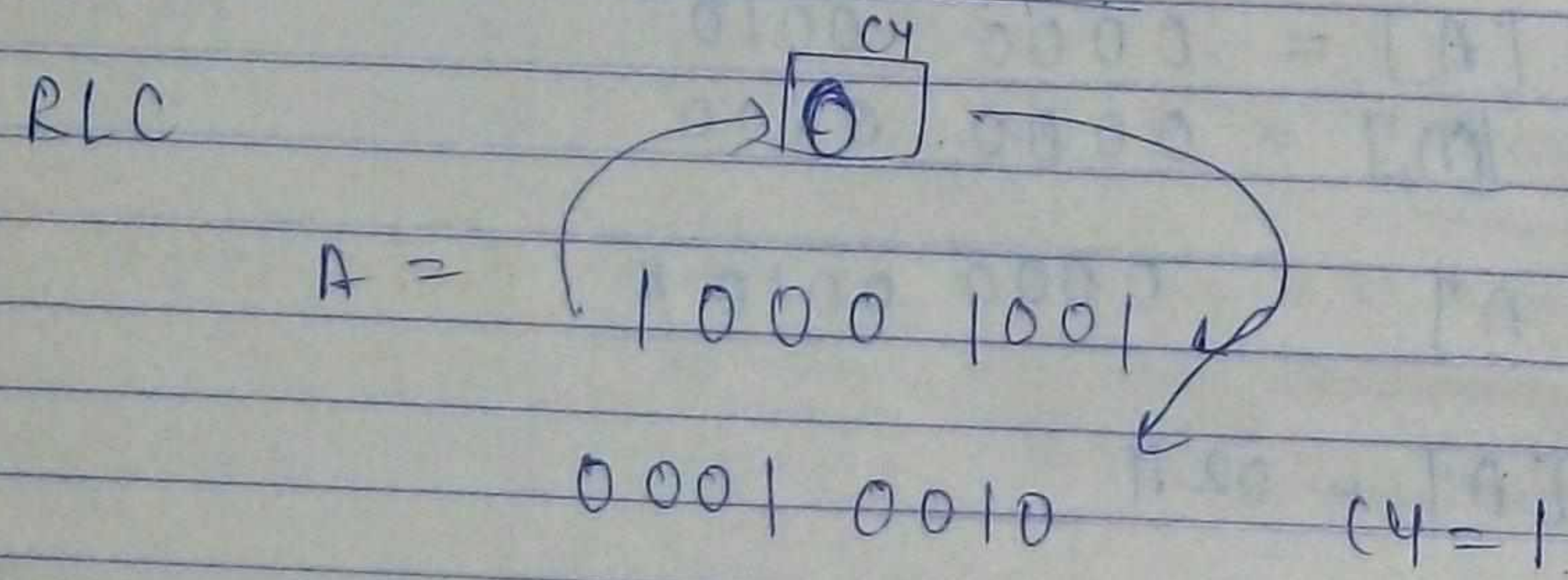
After the execution of above prog. content
of accum & CF + status



$$A+B=4$$

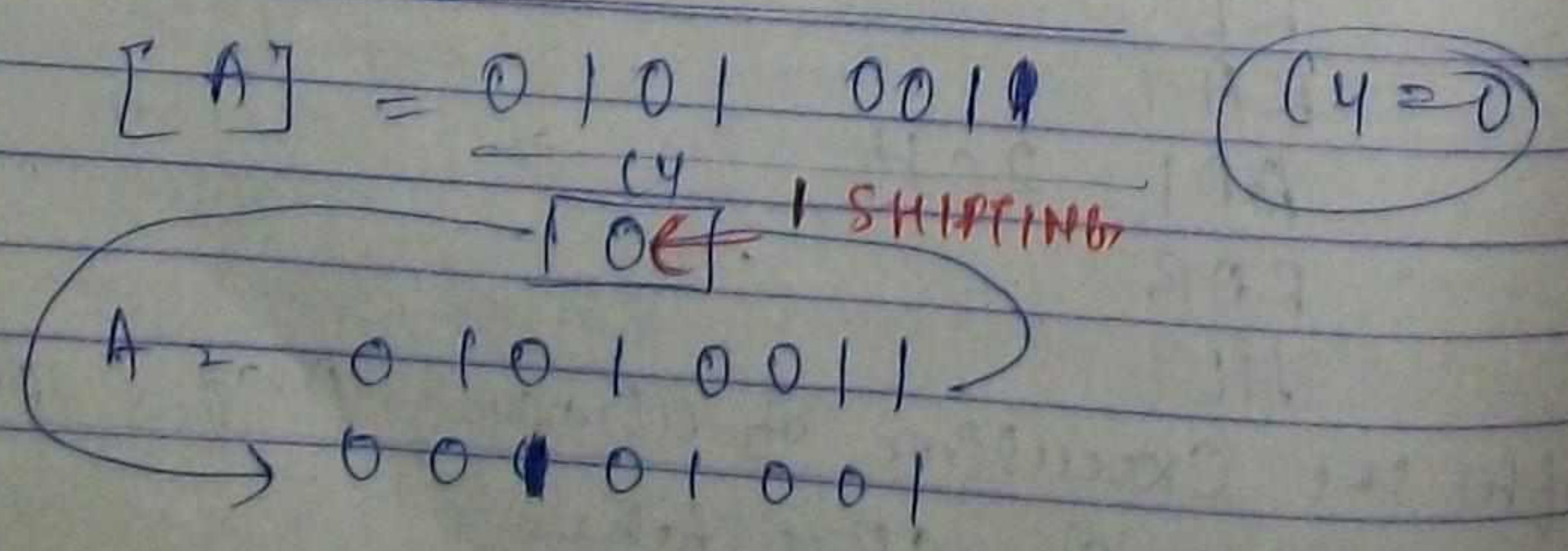
Soln: [A] = 89H

$$\begin{array}{r} [A] = 1000 \ 1001 \\ = 1000 \ 1001 \\ \hline 1000 \ 1001 \end{array} \quad CF=0$$



ADI 2CH

$$\begin{array}{r} [A] = 0010 \ 0111 \\ 2CH = 0010 \ 1100 \end{array}$$



$$[A] = 0010 \ 1001$$

$$[A] = 29H, \quad CY=1$$

⇒ LOGICAL COMPARE INSTRUCTION

① CMP R

IPS = 1 Byte

R = A, B, C, D, E, H, L & M.

operation -

When this instruction will execute content of R will compare with content of accumulator & status of flag will affect accordingly.

NOTE -

Compare inst. is like SUB(A-R) and status of flag will affect according result of SUB(A-R), but result will discard, content of A and R will not change.

	CY (Borrow)	Z
$[A] > [R]$	0	0
$[A] < [R]$	1	0
$[A] \equiv [R]$	0	1

Other flag affect according result of SUB (A-B)

Add. mode

$R \neq M$ - Register add. mode

$M-C \rightarrow F$

$R = M$

Indirect add. mode

$M-C \rightarrow FR$

Ex - $[A] = 37H$

$[B] = 25H$

$[A] - [B]$

Method

$[A] = 00110111$

$[B] = 00100101$

00010010

$CY = 0$

$S = 0$

$Z = 0$

$P = 1$

$AC = ?$

Method

$A - B = A + (-B)$

$= A + (2's \text{ complement of } B)$

$[A] = 00110111$

$[B] = 11011011$

00000010

$CY = 1$

$S = 0$

$Z = 0$

$P = 1$

$AC = 1$

Method

Ex - $[A] = 25H$

$[B] = 37H$

$[A] - [B] =$

00100101

00110111

11101110

$CY = 1$

$S = 1$

$Z = 0$

$P = 1$

$AC = ?$

Method

$[A] + [2's \text{ complement of } B]$

$[A] = 00100101$

$[B] = 11001001$

11101110

$CY = 0$

$S = 1$

$Z = 0$

$P = 1$

$AC = 0$

Ques. $[A] = 29H$

$[B] = 13H$

CMP B

Ans

$[A] > [B]$

$CY = 0$

$Z = 0$

$[A] = 00101001$

$S = 0$

$AC = ?$

$[B] = 00010011$

$P = 0$

00010110

for AC

$$\begin{array}{r}
 [A] = 00101001 \\
 2^5 [B] = 11101101 \\
 \hline
 00010110
 \end{array}$$

$$CY \leftarrow \overline{CY} = 0$$

$S=0, P=0, Z=0, AC=1$

② CPI 8 bit data -

operation -

8 bit data will compare with
content of accumulator and status of flag affect
accordingly

	CY	Z
$[A] > 8 \text{ bit data}$	0	0
$[A] < 8 \text{ bit data}$	1	0
$[A] \equiv 8 \text{ bit data}$	0	1

remain flag affect according the result
of $(A - 8 \text{ bit data})$

add. mode — Immediate add. mode
mc — FR

NOTE -

CMP & CPI is logical instruction so
status of all flag will affected.

⇒ STACK -

- It is group of continuous memory location that
define in main memory, used for temporary
storage of information during the execution of
main program.

- Stack initialized in main memory location by
LXI SP 16 bit data.

- At a time two byte data can store or retrieve
at / from top of stack.

- In 8085 two instructions are available for data
store or retrieve at / from top of stack that
is PUSH & POP.

PUSH = data store

POP = data retrieve.

- If two bytes data store at the top of stack
it grows upward with numerically
decreasing of its address.

LXI SP 2063

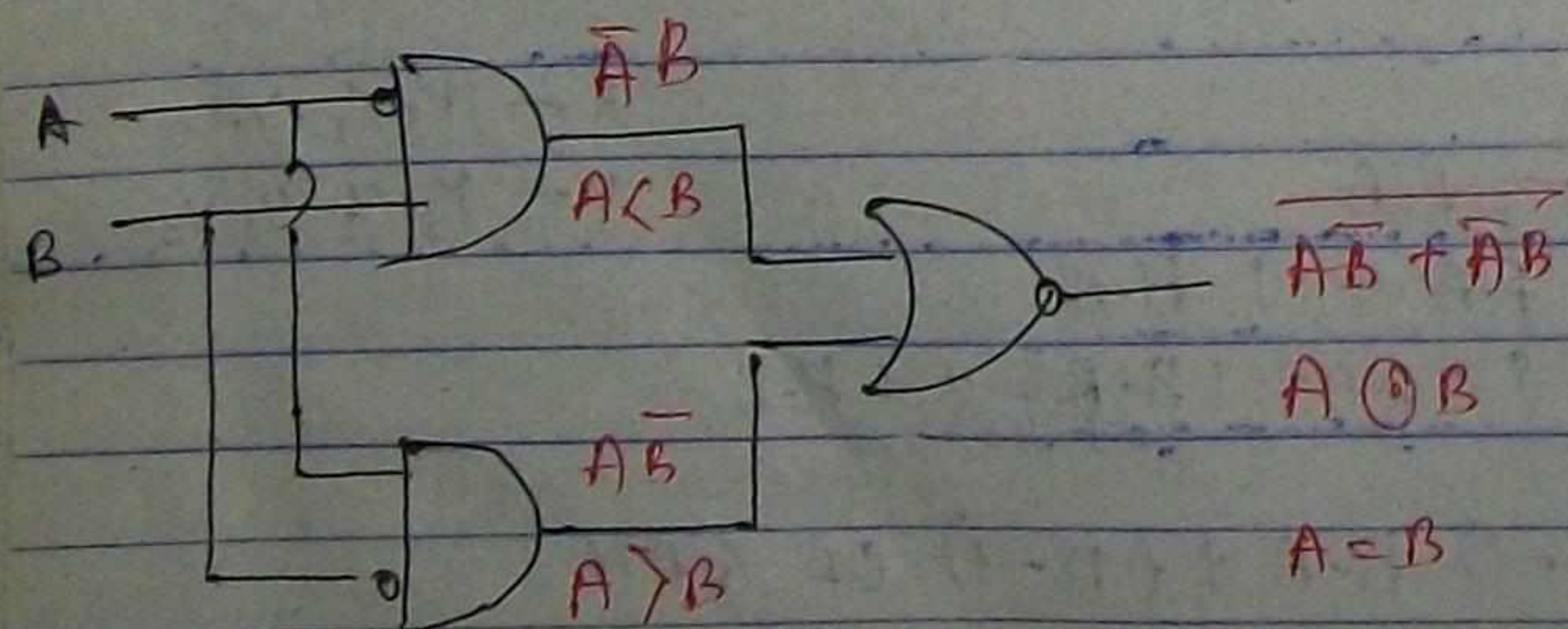
29	← SP New position
69	
00	← SP = 2063

DISADVANTAGE -

- Parallel binary adder is having carry propagation delay, this is also known as ripple carry adder.
- To overcome this difficulty look ahead carry adder is used (LCA).
- Among F.A, P.A, LCA, LCA is the fastest.

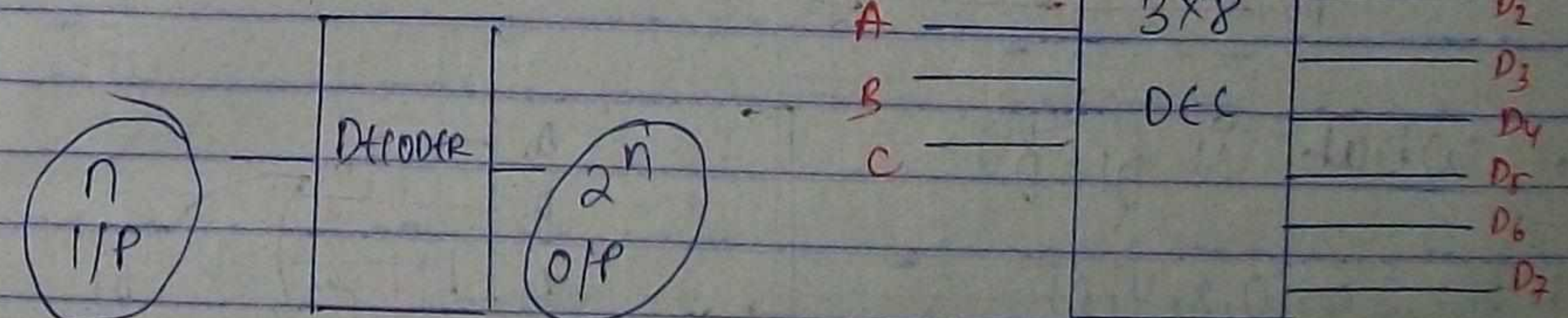
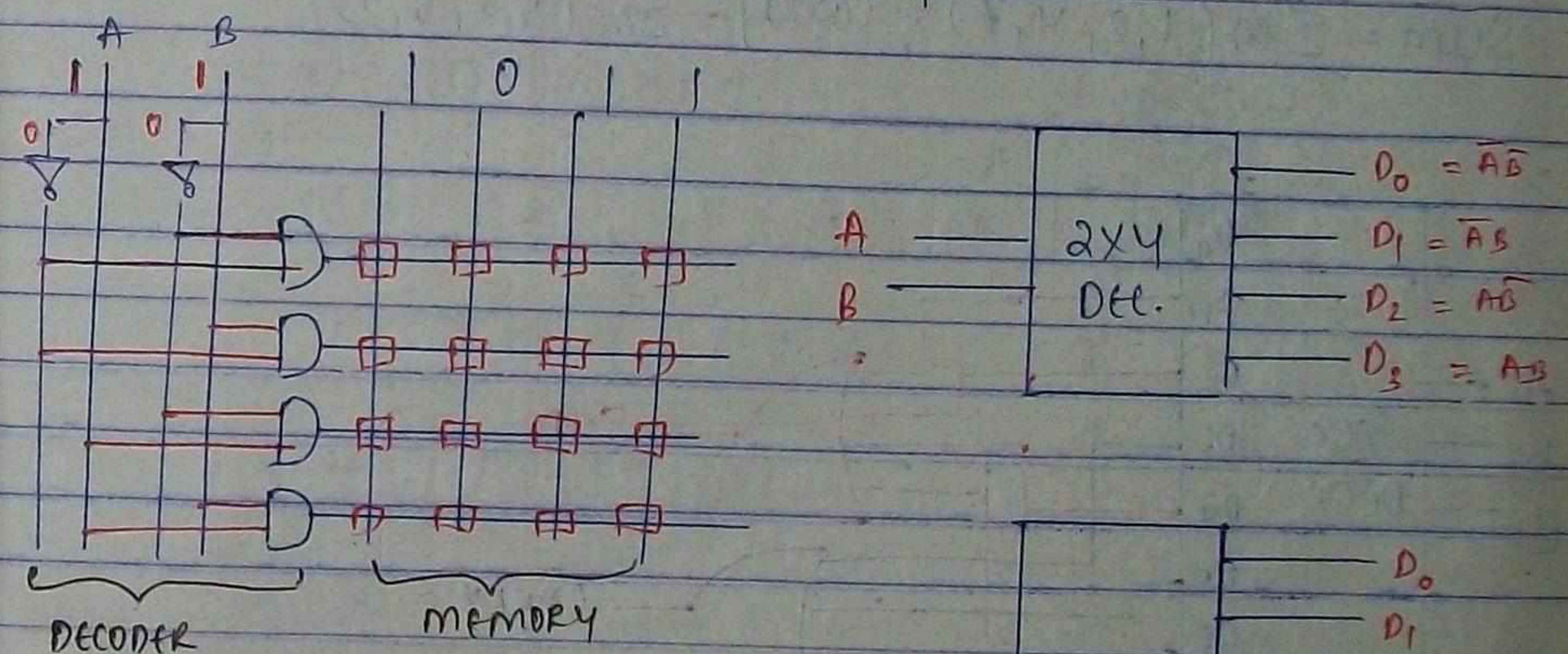
MAGNITUDE COMPARATOR -

A	B	$A < B$	$A = B$	$A > B$
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0
		$\bar{A}B$	$A \odot B$	$A\bar{B}$



DECODER -

- A Decoder have many I/P and many O/P lines.
- It is a combinational circuit that convert binary information from n-I/P line to max 2^n O/P lines. (unique)

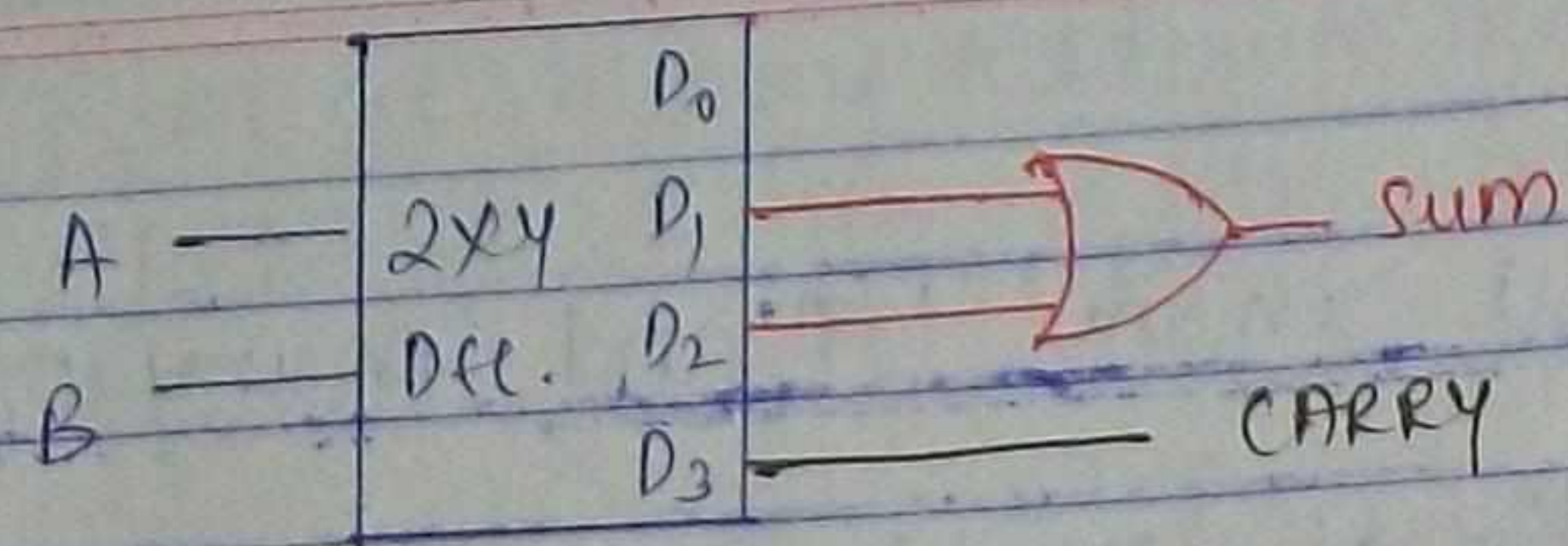


HALF ADDER -

$$\text{Sum} = \sum m(1, 2)$$

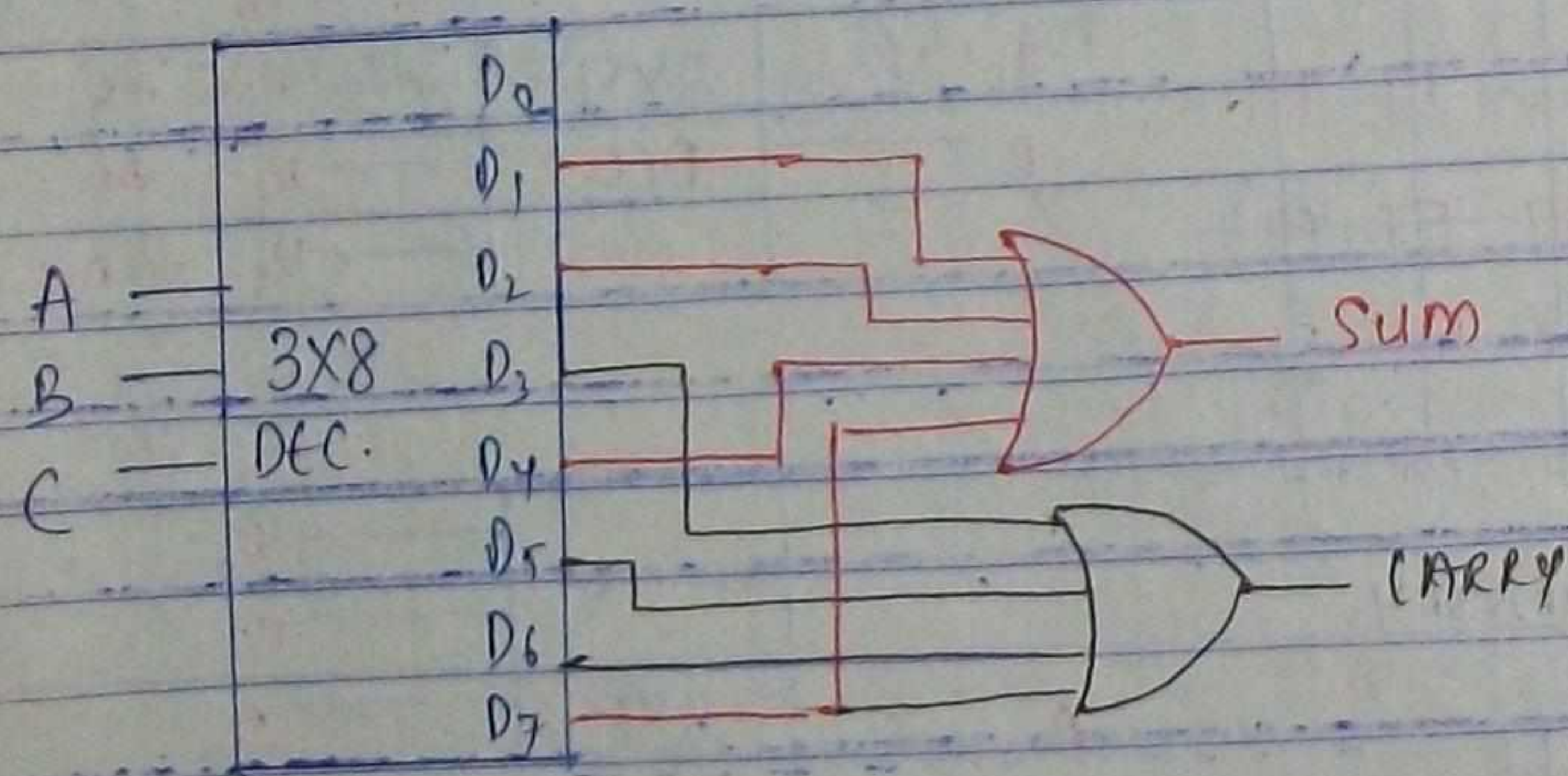
$$\text{Carry} = \sum m(3)$$

A	B	C	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



⇒ FULL ADDER —

$$\text{Sum} = \sum m(1, 2, 4, 7), \text{ carry} = \sum m(3, 5, 6, 7)$$

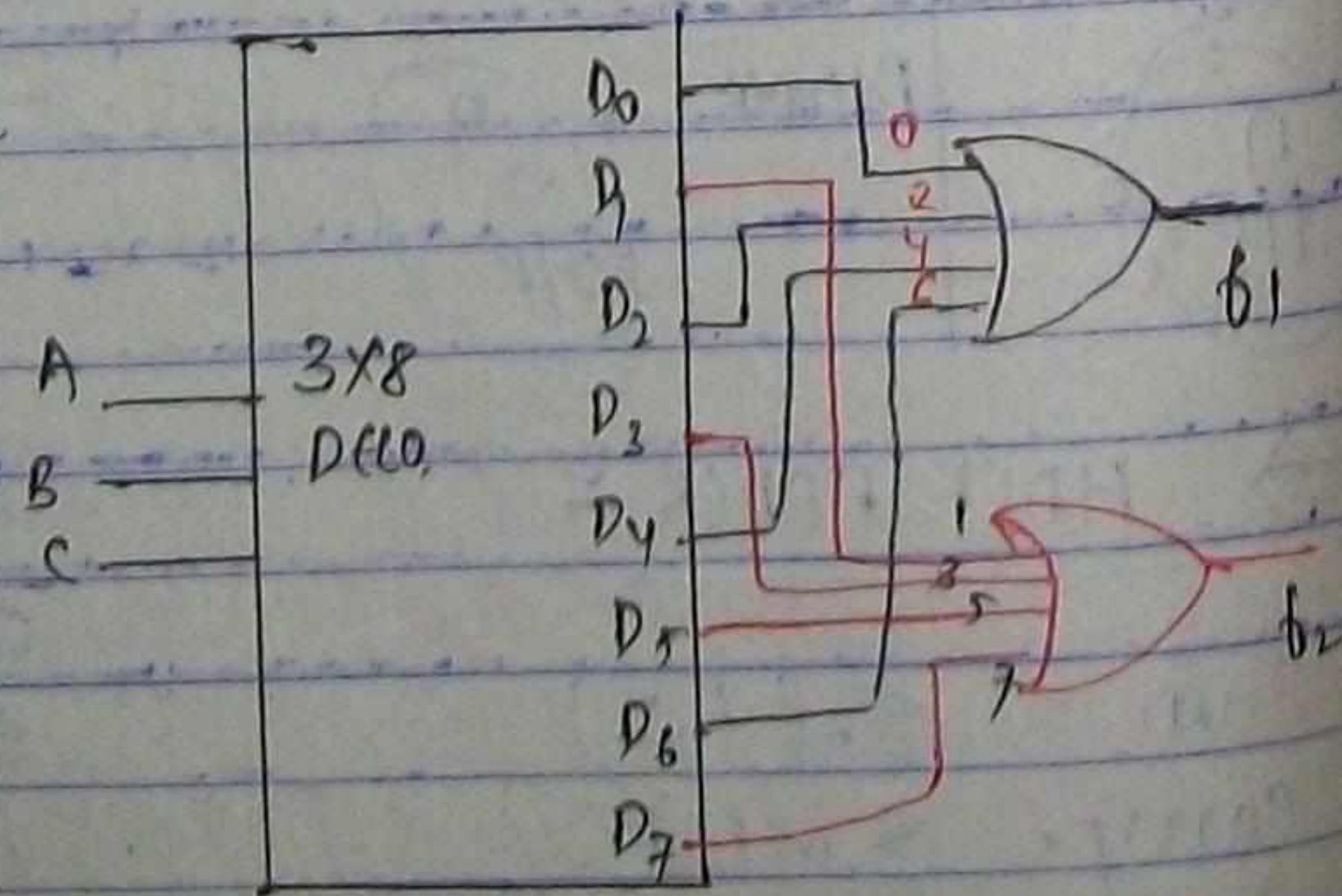
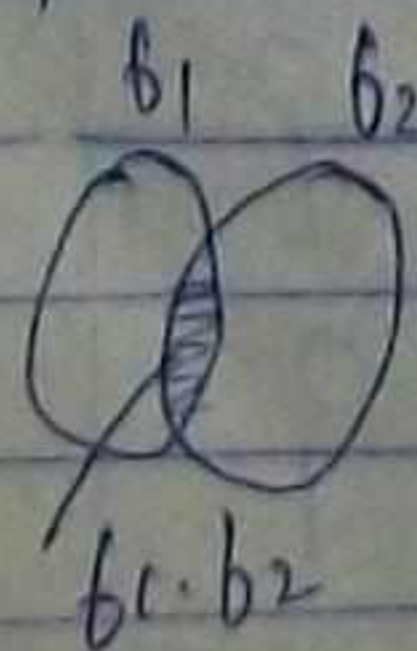


Ques. what is $b_1 \cdot b_2$

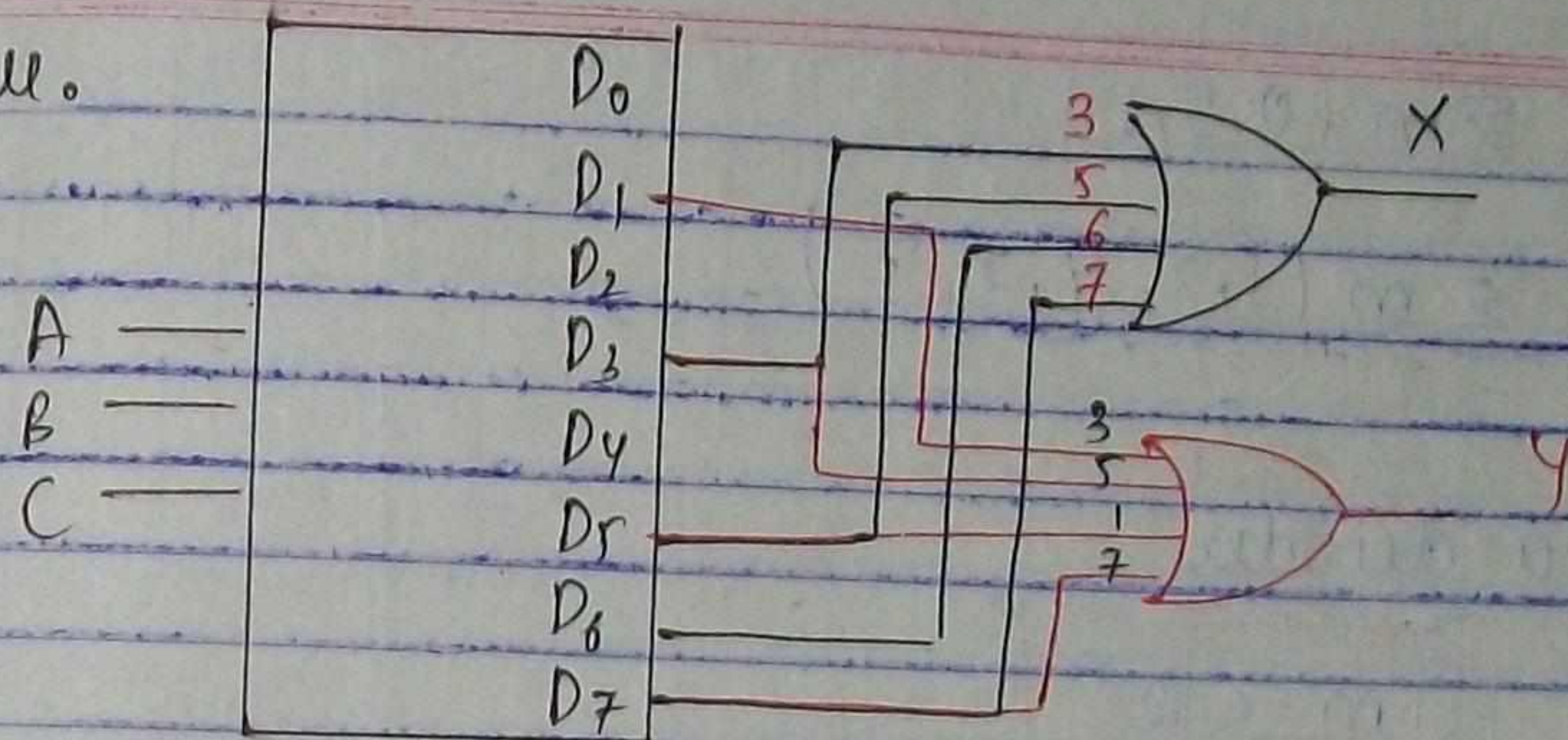
Solⁿ: $b_1 = \sum m(0, 2, 4, 6)$

$$b_2 = \sum m(0, 3, 5, 7)$$

$$b_1 \cdot b_2 = 0$$



Ques.



what is X and Y,

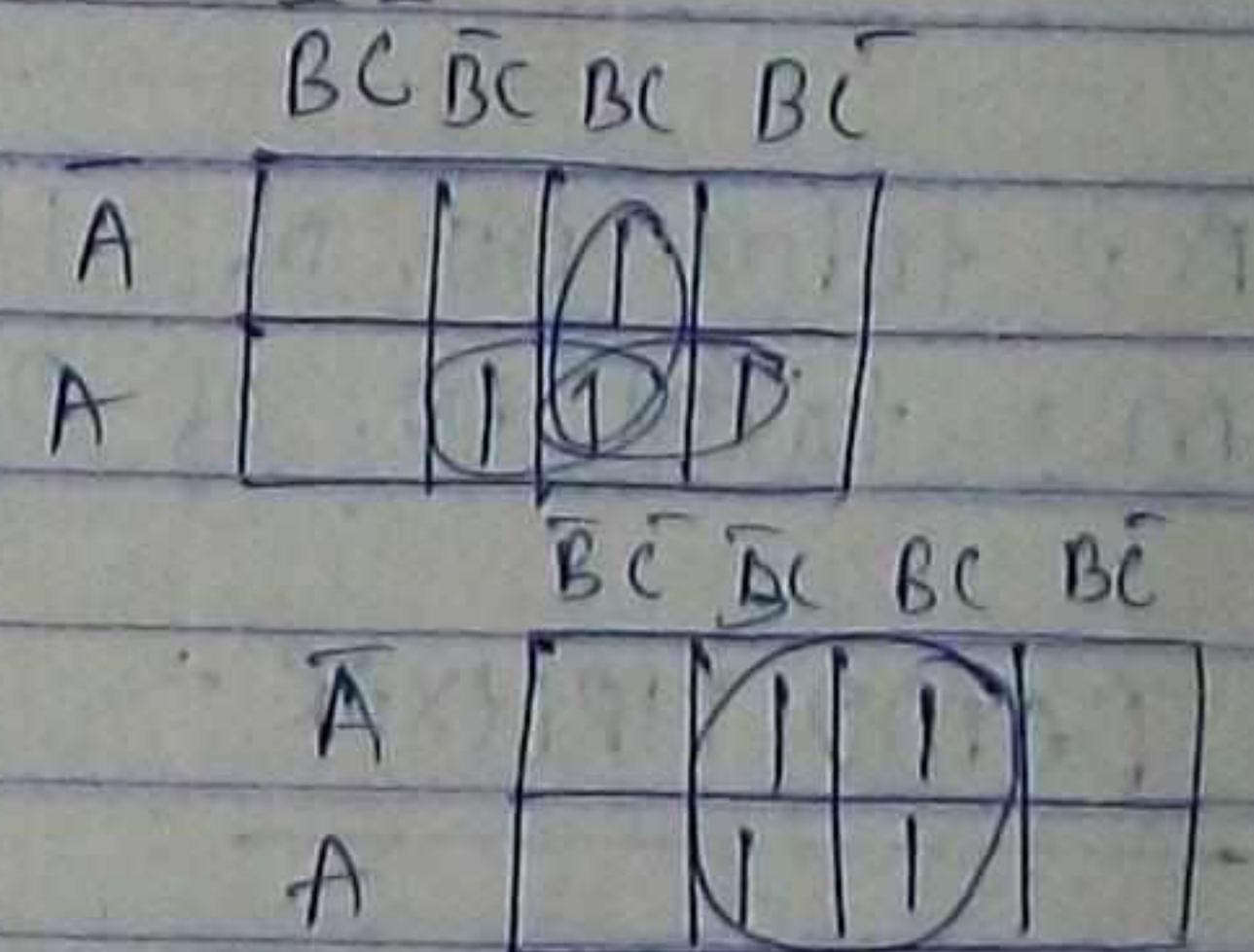
Solⁿ:

$$X = \sum m(3, 5, 6, 7)$$

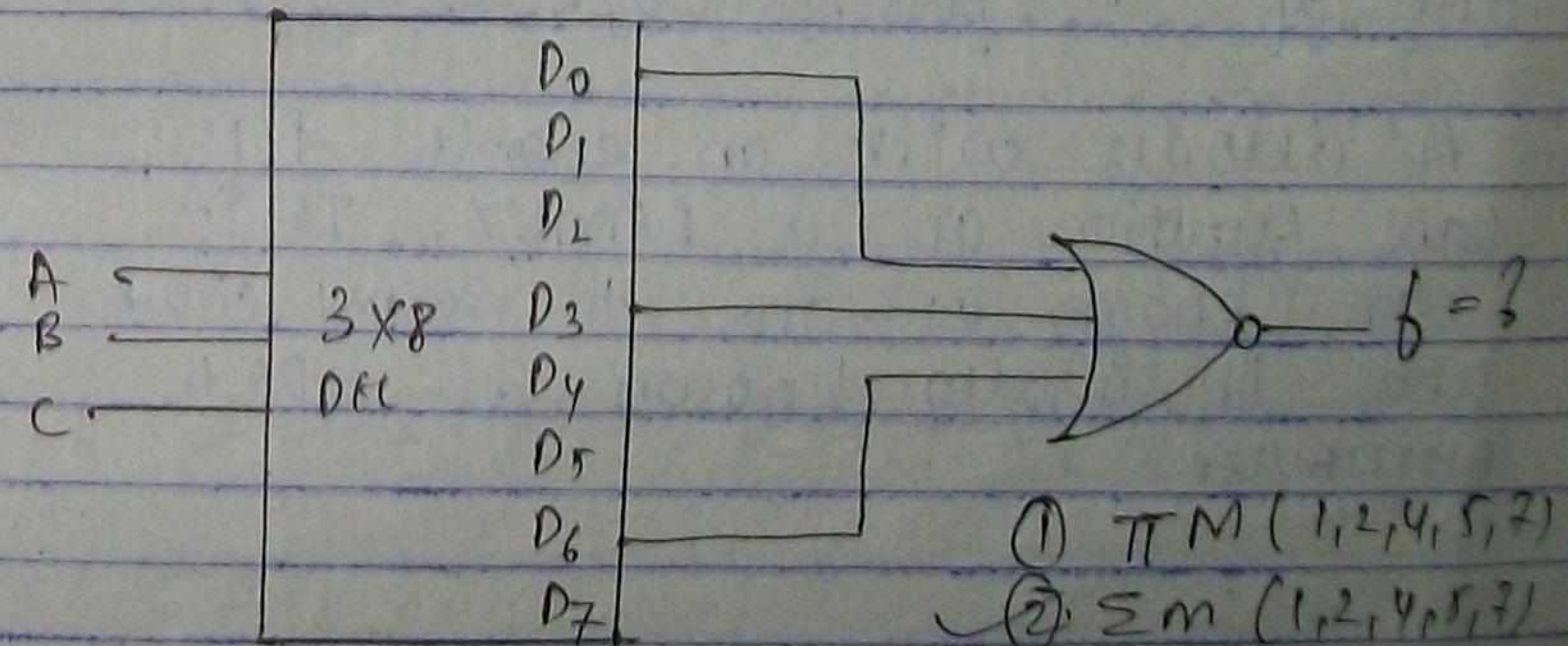
$$Y = \sum m(3, 5, 6, 7)$$

$$X = AB + BC + CA$$

$$Y = \underline{\underline{C}}$$



Ques.



- ① $\pi M(1, 2, 4, 5, 7)$
- ② $\sum m(1, 2, 4, 5, 7)$
- ③ $\sum m(0, 3, 6)$
- ④ NONE

$$\left\{ \begin{aligned} \Sigma m(1,2) &= \Sigma m(0,3) \\ &= \Pi m(1,2) \end{aligned} \right\}$$

$$f = \Sigma m(0,3,6)$$

$$= \Sigma m(1,2,4,5,7)$$

for a decoder

$$m \leq 2^n$$

n = total no. of I/P lines.

m = total no. of O/P lines.

⇒ Demultiplexer -

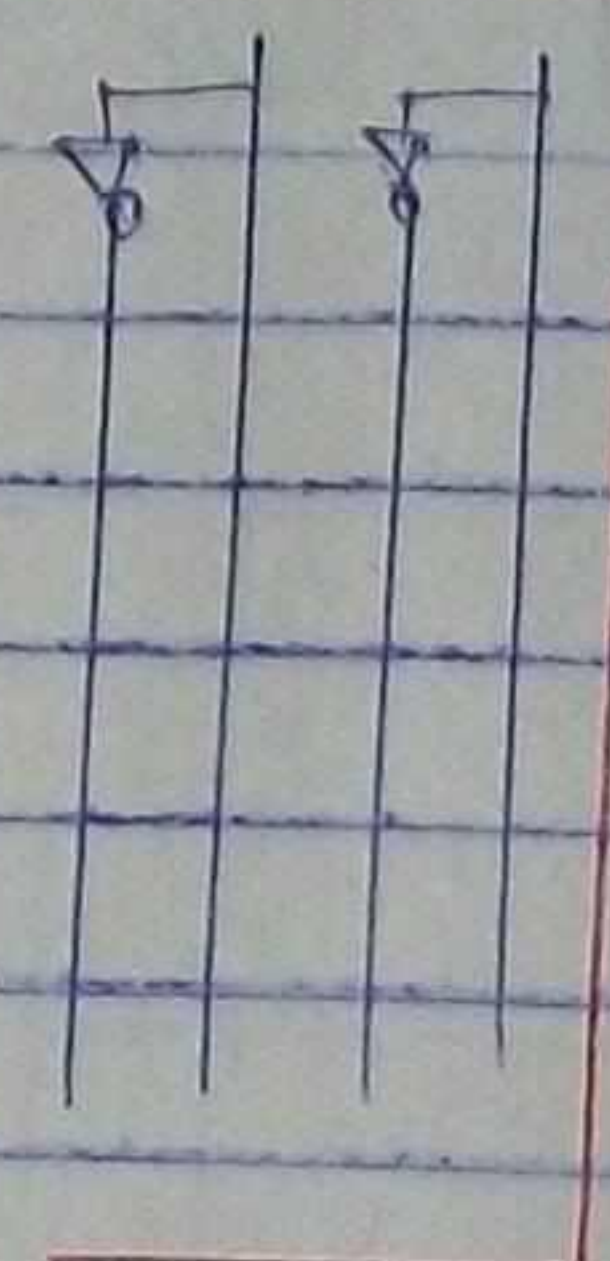
- Demux. is a combinational circuit that receives information on a single line and transmits this on one of 2^n possible O/P lines.

- A decoder with an enable I/P can function as a Demux. It is also known as one to many circuit and it is also known as DATA DISTRIBUTER.

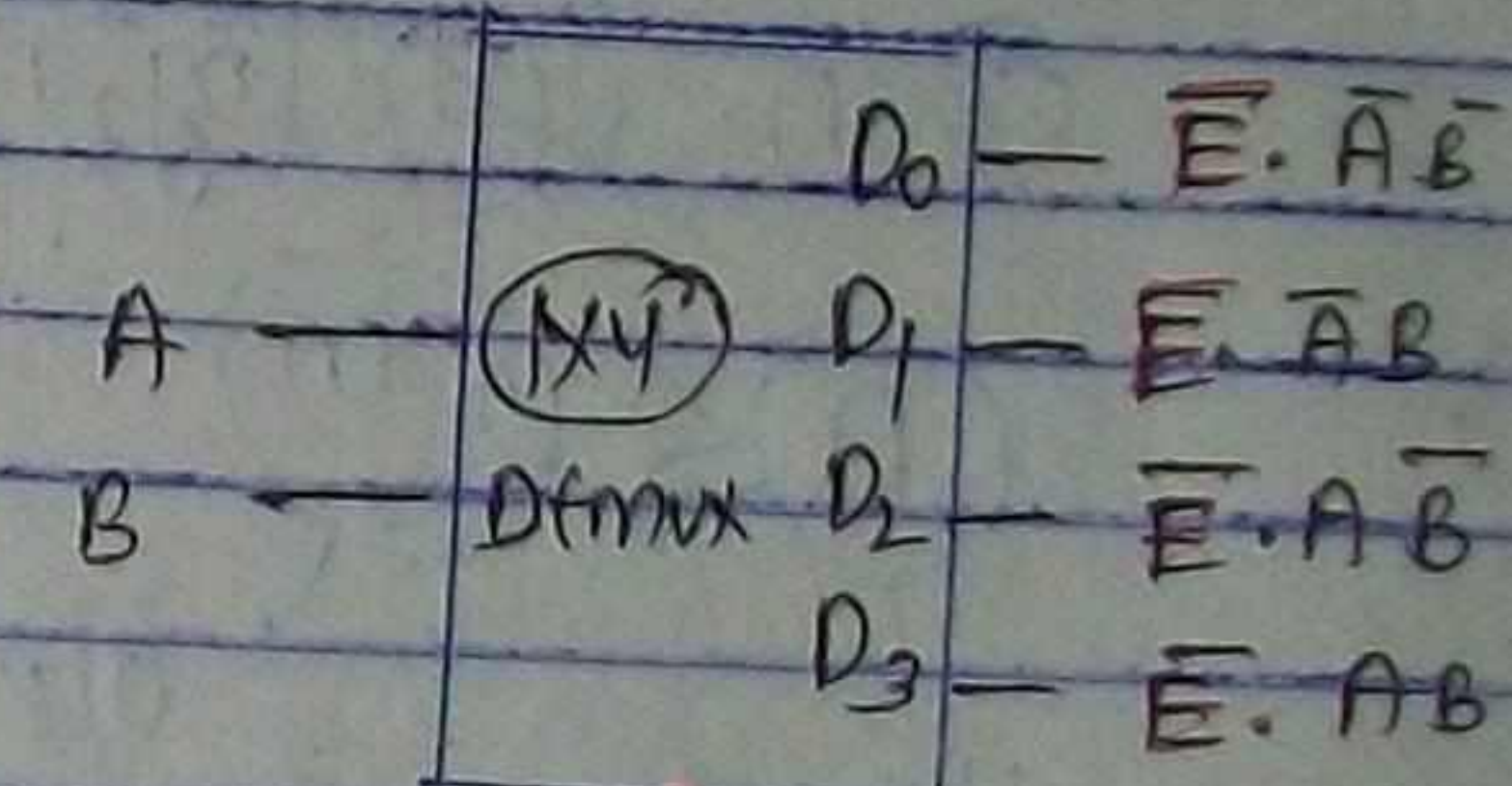
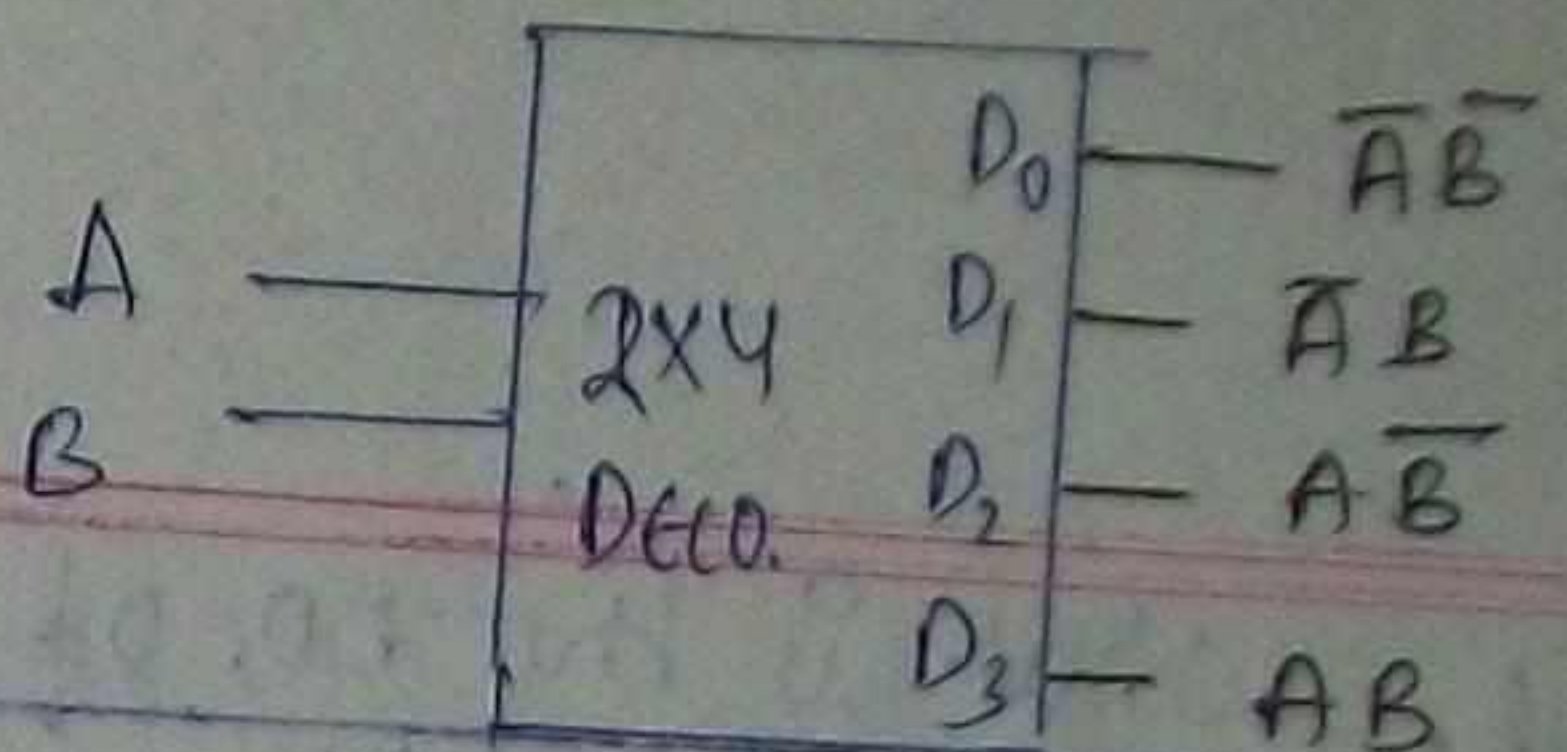
DEMUX

MEMORY

A B

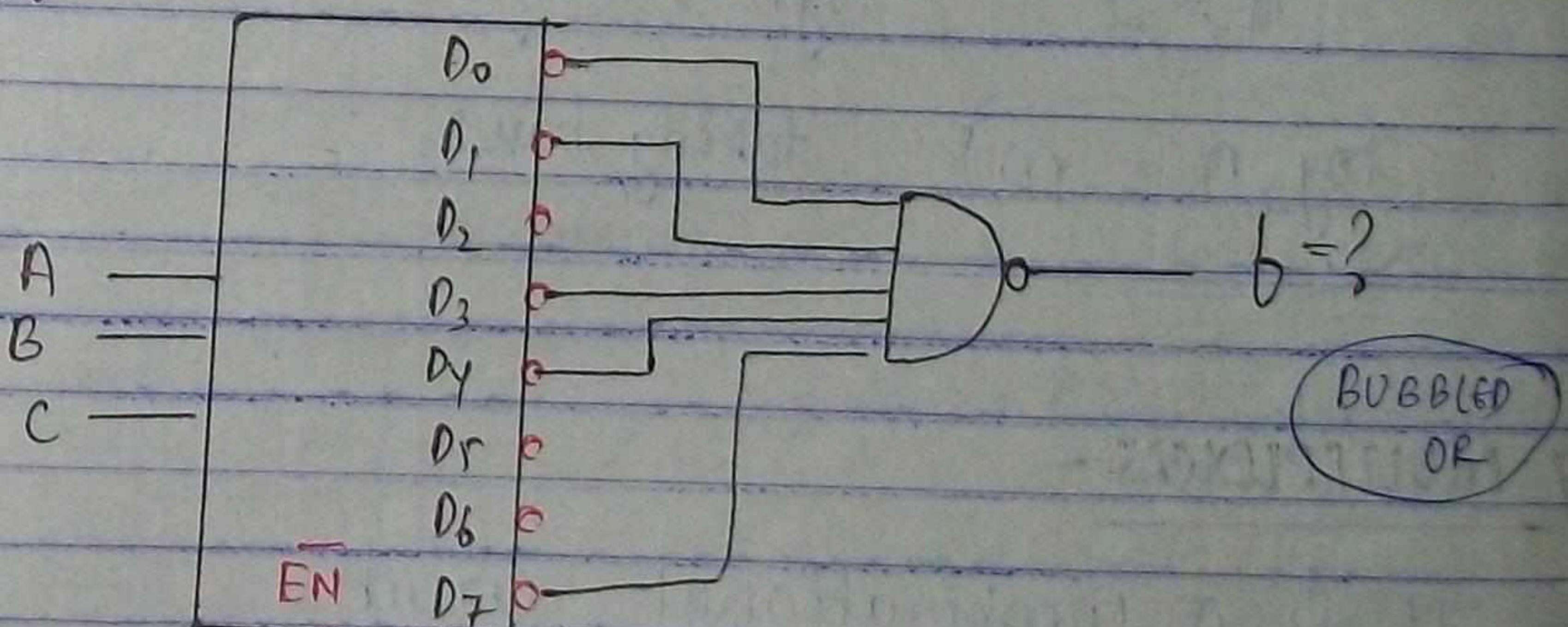


DATA LINE



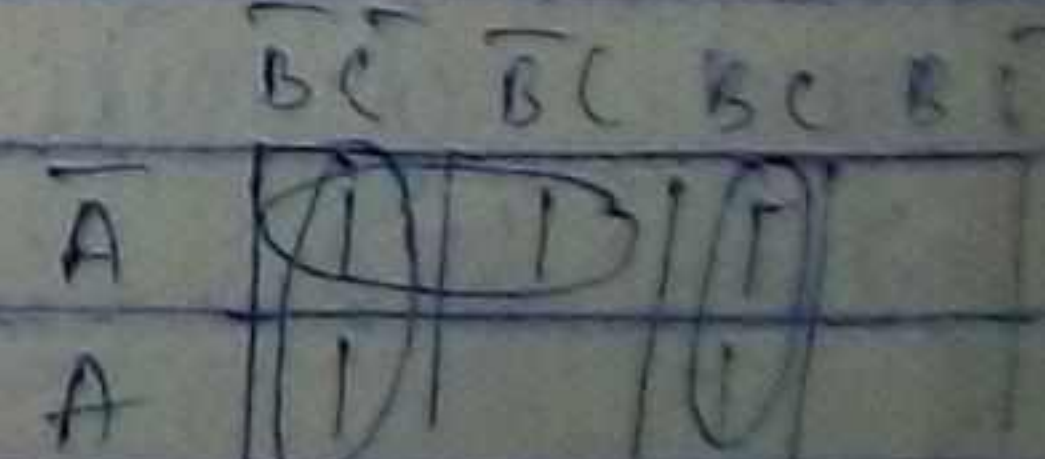
DATA LINE

Ques



Soln: $f = \overline{EN} \times \Sigma m(0,1,3,4,7)$

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



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

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

Ques. what is the no. of select line required in a single I/P and n O/P demux.

- ① 2 ② n ③ $\log_2 n$ ④ 3

Solⁿ:

$$n < 2m$$

HP line

O/P line

7 12 2 3

$$\log n = m \log 2$$

$$\log_2 n = m$$

taking base 2

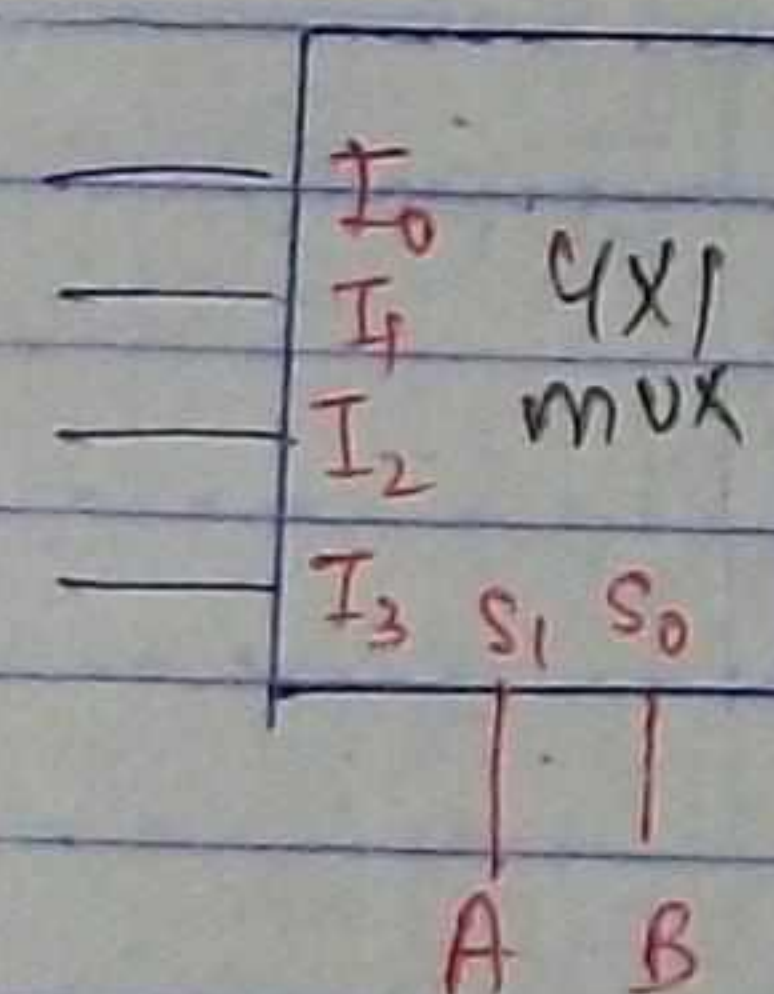
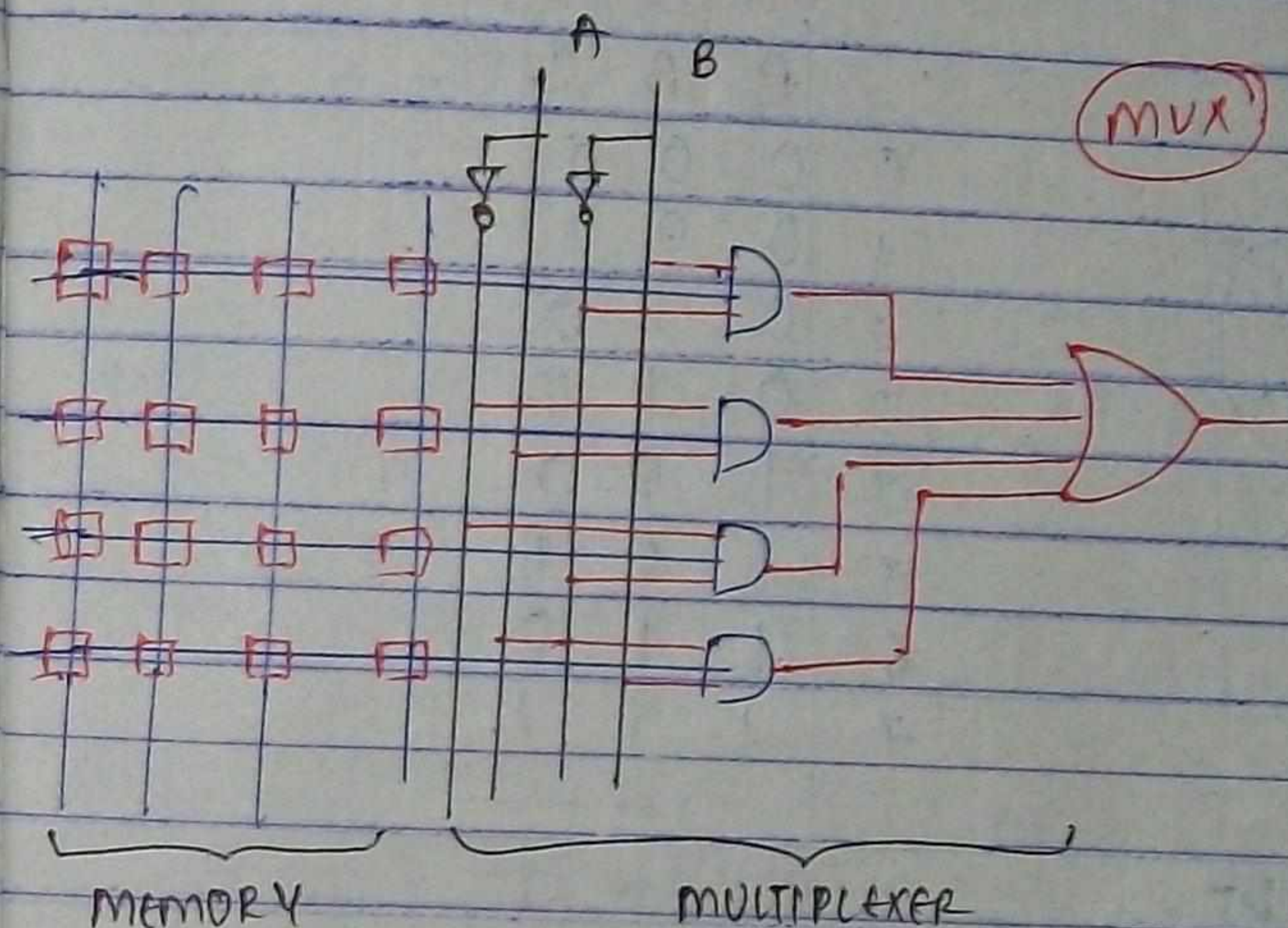
⇒ MULTIPLEXERS -

- It is a combinational circuit that select binary information from one of many I/P lines and directs it to a single O/P line.

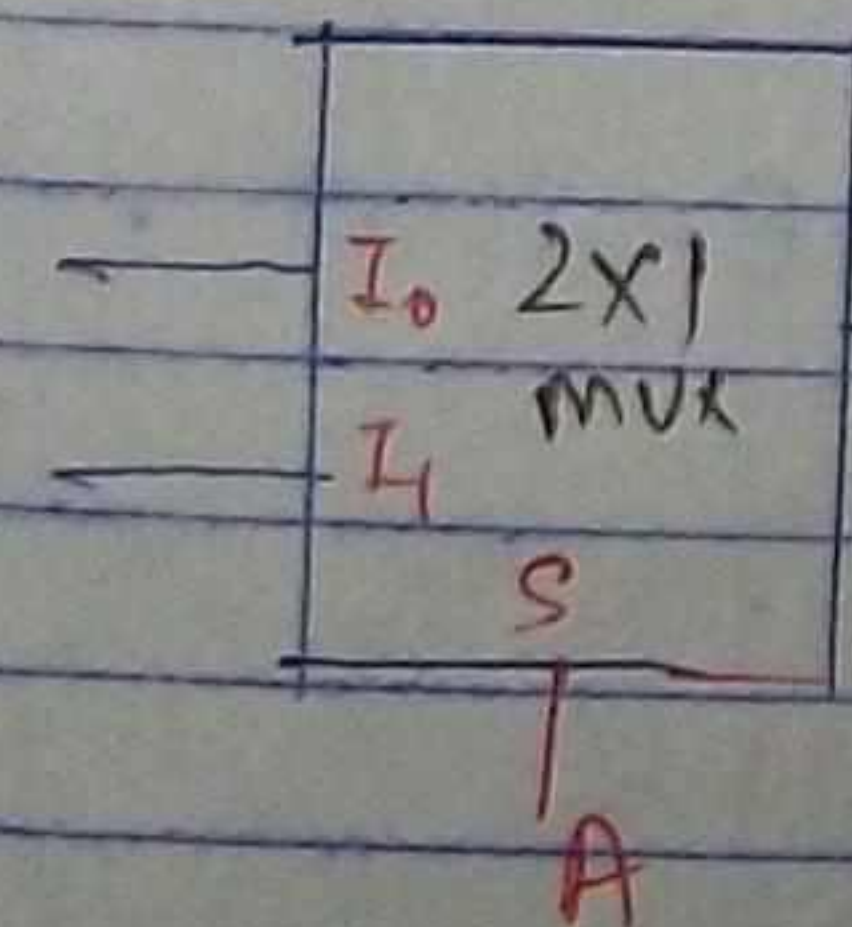
— multiplexers can also be called as

1. DATA SELECTOR
2. MANY TO ONE CIRCUIT
3. UNIVERSAL LOGIC CIRCUIT

40 PARALLEL TO SERIAL CONVERTER
50 WAVE FORM GENERATOR



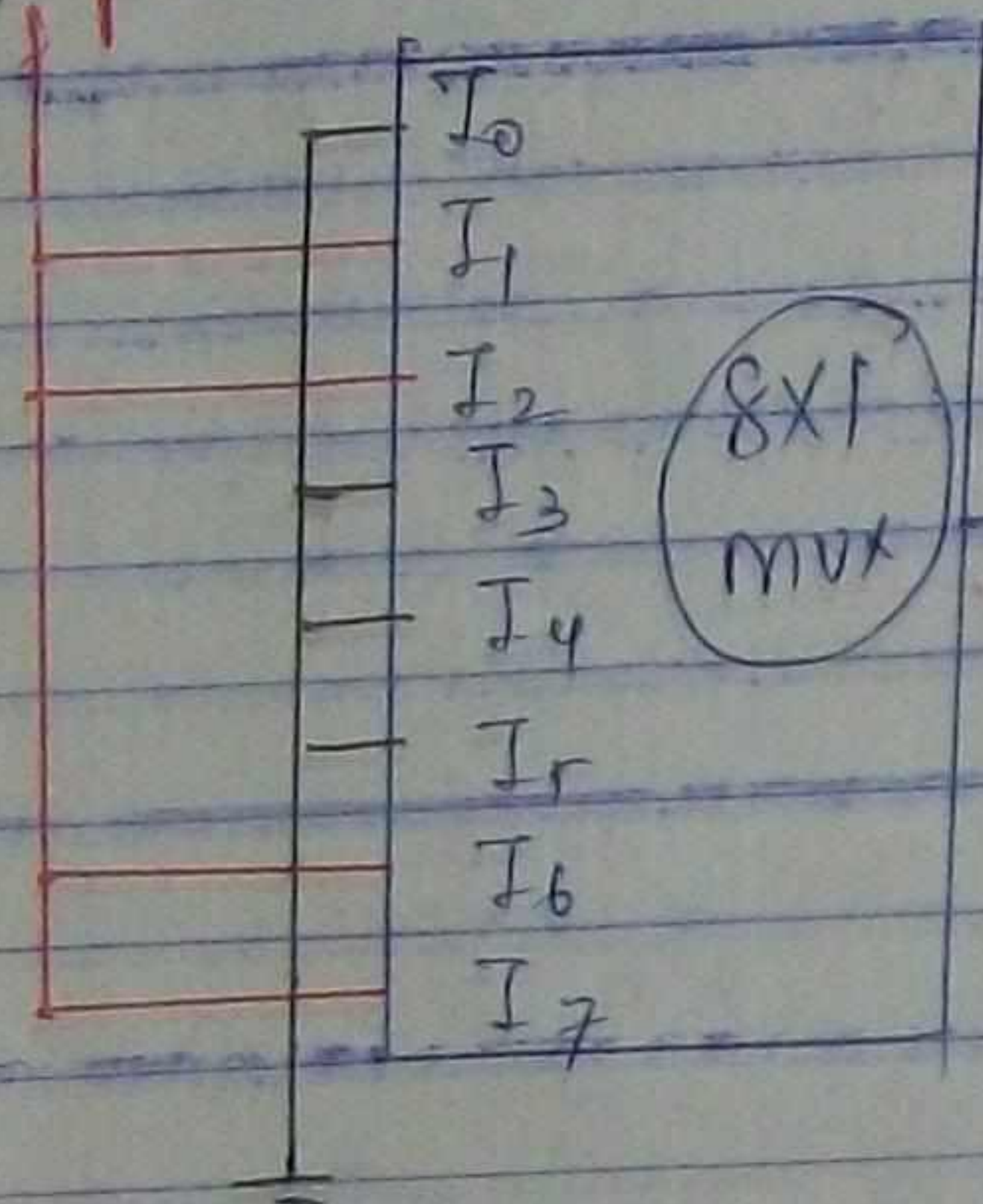
$$f = \overline{A}\overline{B}(I_0) + \overline{A}B(I_1) + A\overline{B}(I_2) + AB(I_3)$$



$$I = \bar{A}I_0 + AI_1$$

Ex - $f(ABC) = \sum (1, 2, 6, 7)$

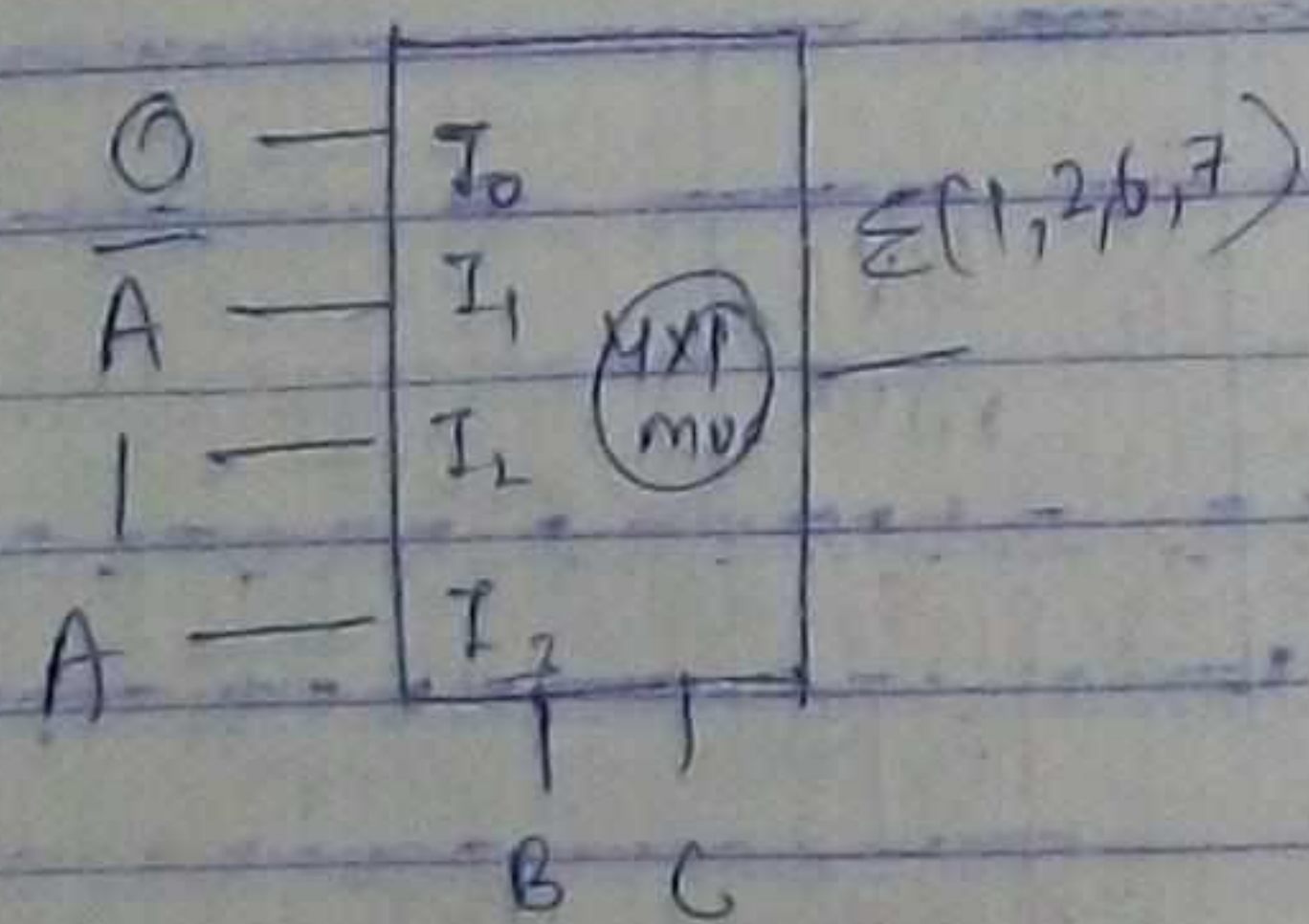
① DIRECT METHOD -



	msb		lsb
	A	B	C
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

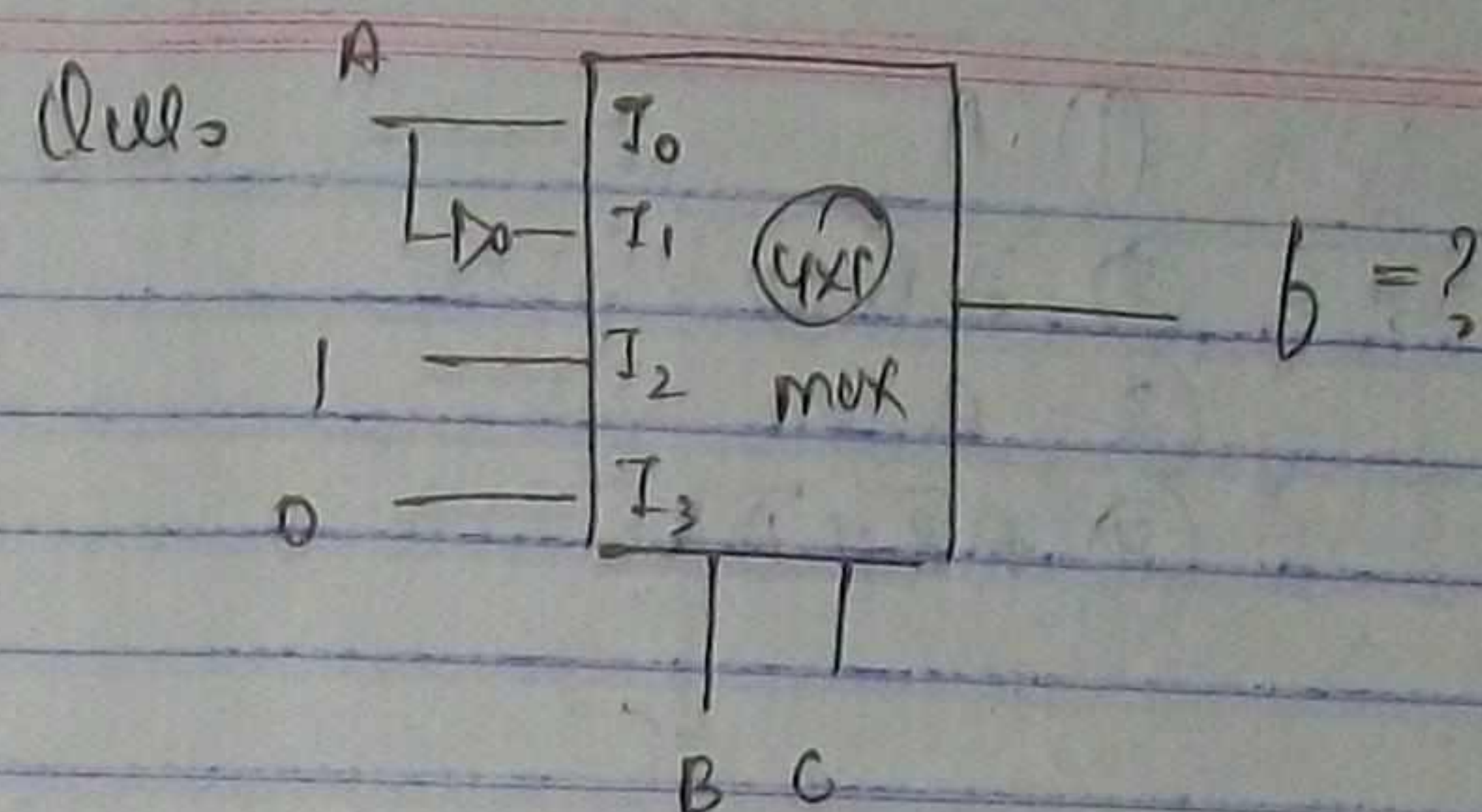
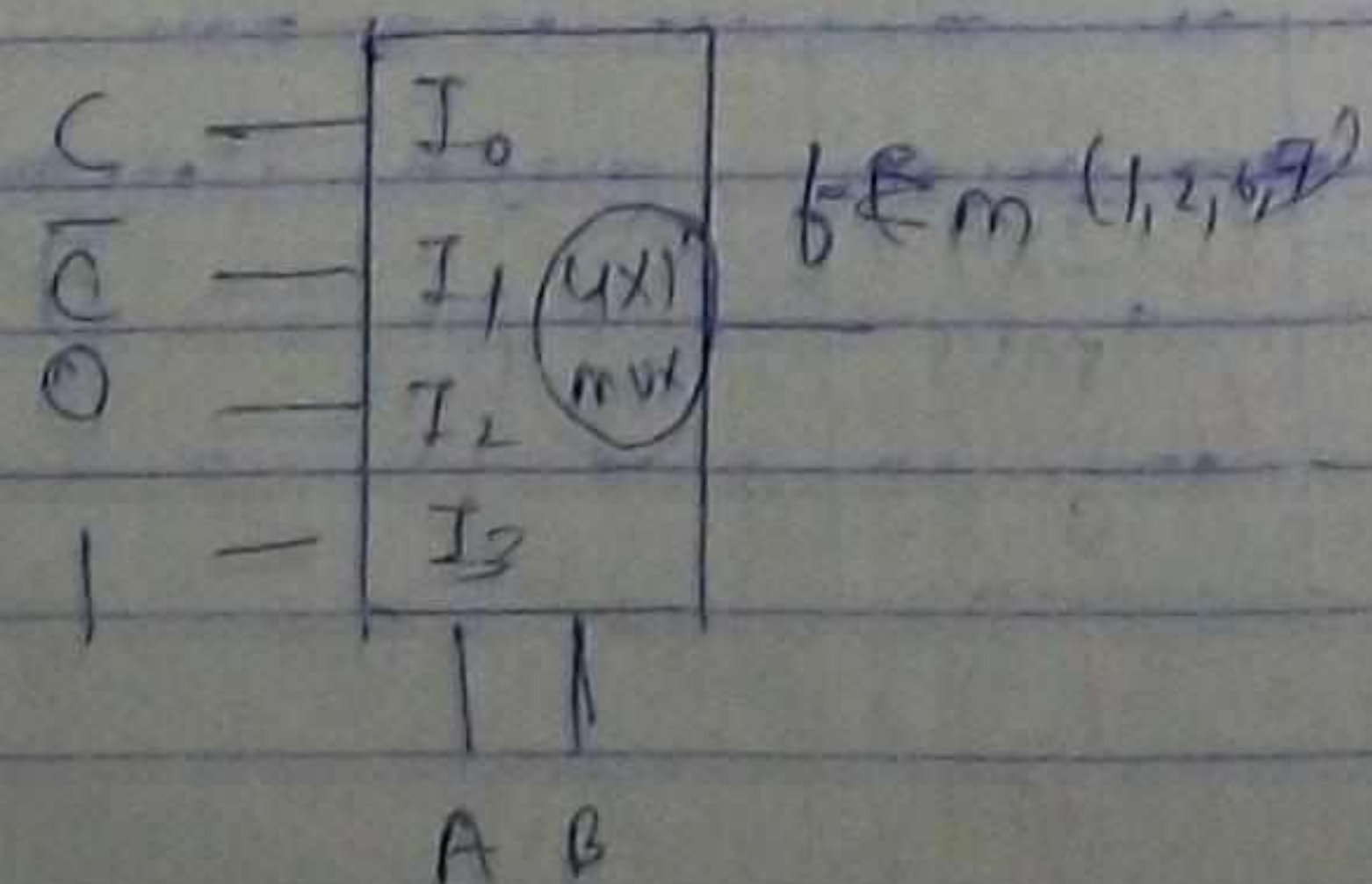
② MSB METHOD -

	I ₀	I ₁	I ₂	I ₃
$\bar{A}$	0	1	2	3
A	4	5	6	7
	0	$\bar{A}$	1	A



③ LSB METHOD -

	I ₀	I ₁	I ₂	I ₃
$\bar{C}$	0	2	4	6
C	1	3	5	7
	$\bar{C}$	0	1	

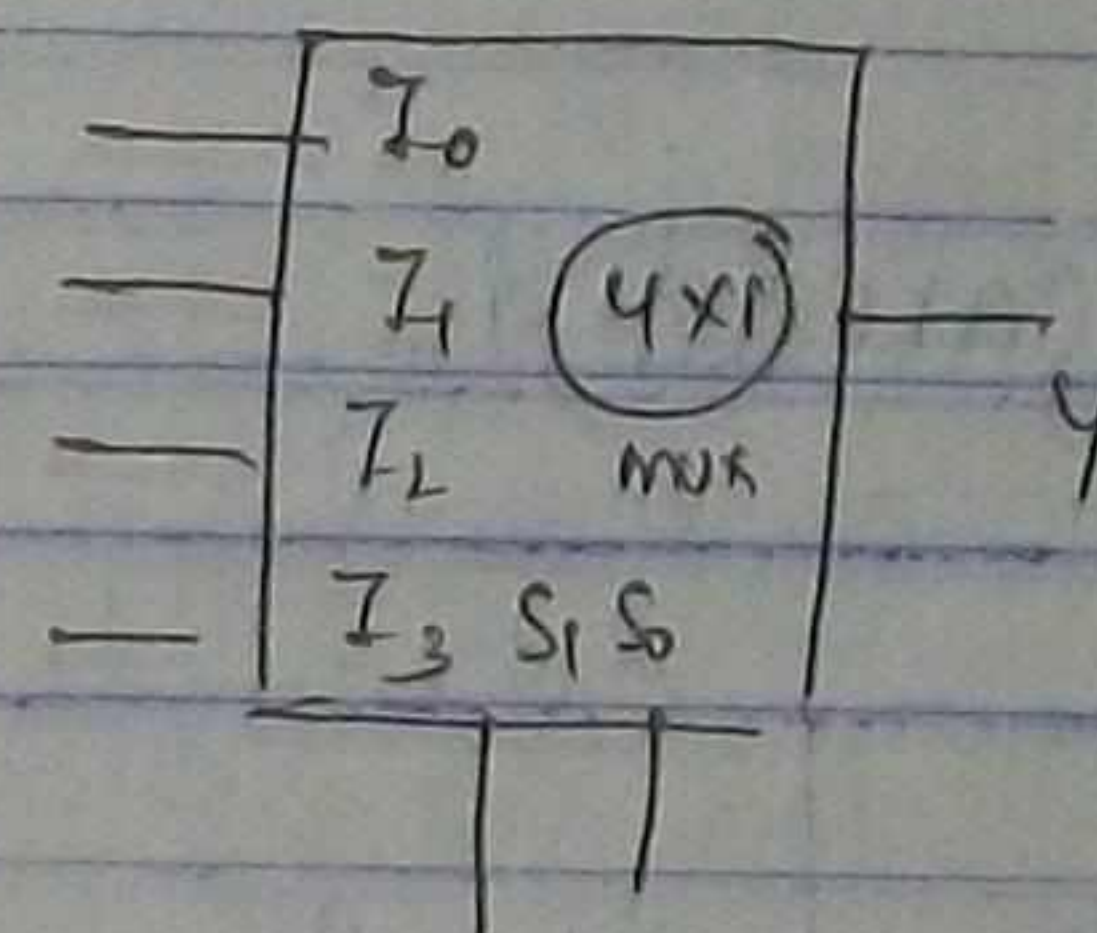


$f = \sum m(1, 2, 4, 6)$

soln:

	I ₀	I ₁	I ₂	I ₃
$\bar{A}$	0	1	2	3
A	4	5	6	7
	A	$\bar{A}$	1	0

Ques



$(I_0 I_1 I_2 I_3 S_1 S_0) = ?$

if

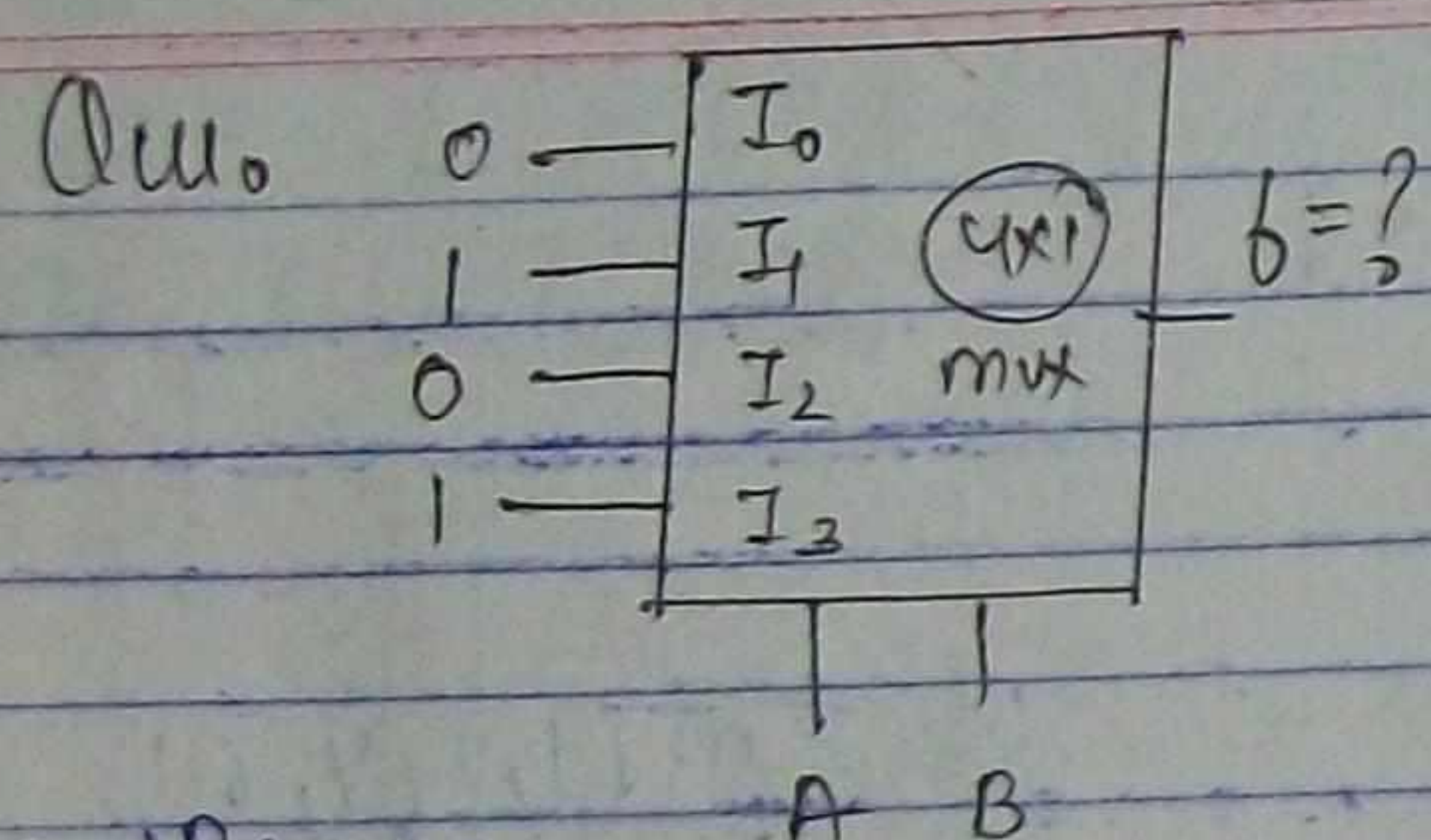
$f(ABC) = \prod m(0, 2, 4, 7)$

soln:

$f = \prod m(0, 2, 4, 7)$
 $= \sum m(1, 3, 5, 6)$

$\sum (0, 1, A, \bar{A}, B, C)$

	I ₀	I ₁	I ₂	I ₃
$\bar{A}$	0	1	2	3
A	4	5	6	7
	0	1	A	A



- ① A
- ② $\bar{A}B$
- ③ B
- ④ $A\bar{B} + \bar{A}B$

Soln:

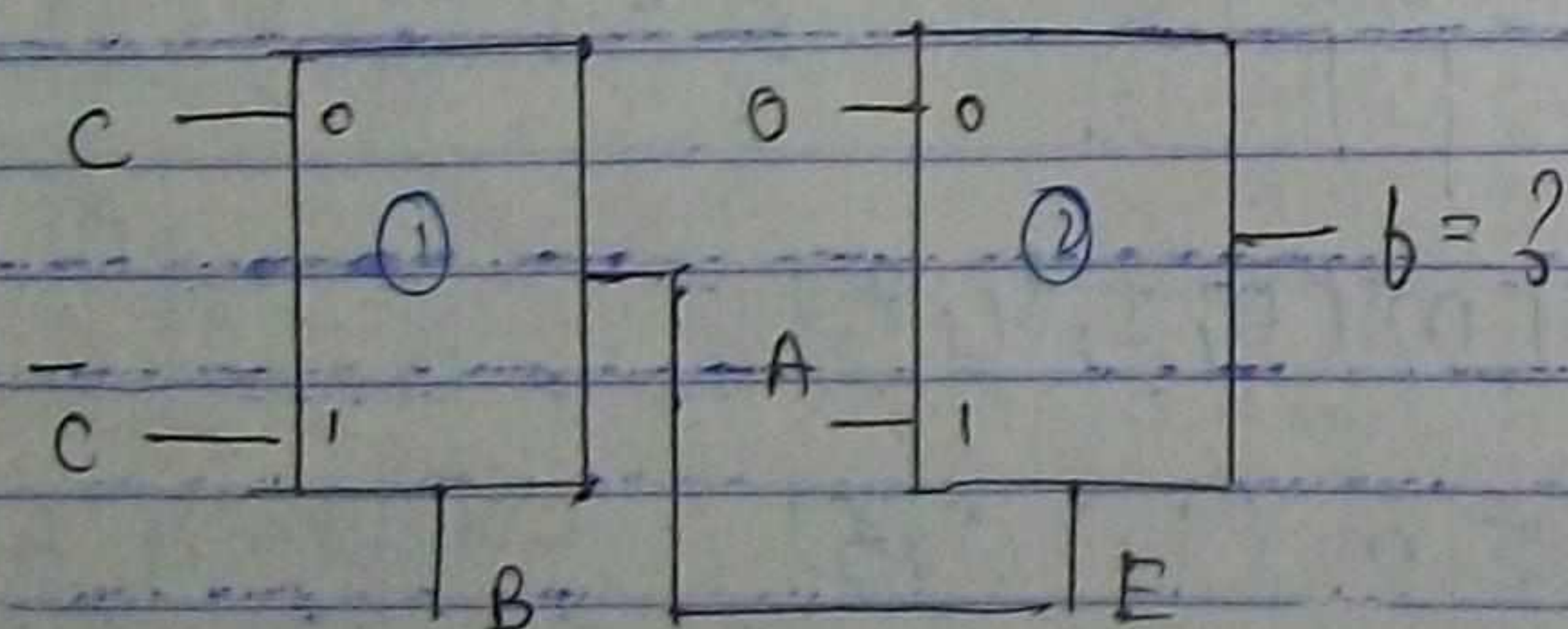
$$f = \bar{A}B + AB$$

$$= B$$

Sinu $f = \bar{A}\bar{B}(I_0) + \bar{A}B(I_1) + AB(I_2) + AB(I_3)$

⑧ Multiplexers can be cascaded.

Ques.



① $\bar{A}BC + ABC$

② $ABC + \bar{A}\bar{B}\bar{C}$

③ $\bar{A}BC + \bar{A}\bar{B}\bar{C}$

④ $\bar{A}\bar{B}C + \bar{A}B\bar{C}$

Soln: from ①

$$E = \bar{B}(I_0) + B(I_1)$$

$$= \bar{B}(1) + B(0) = \bar{B}$$

from ②

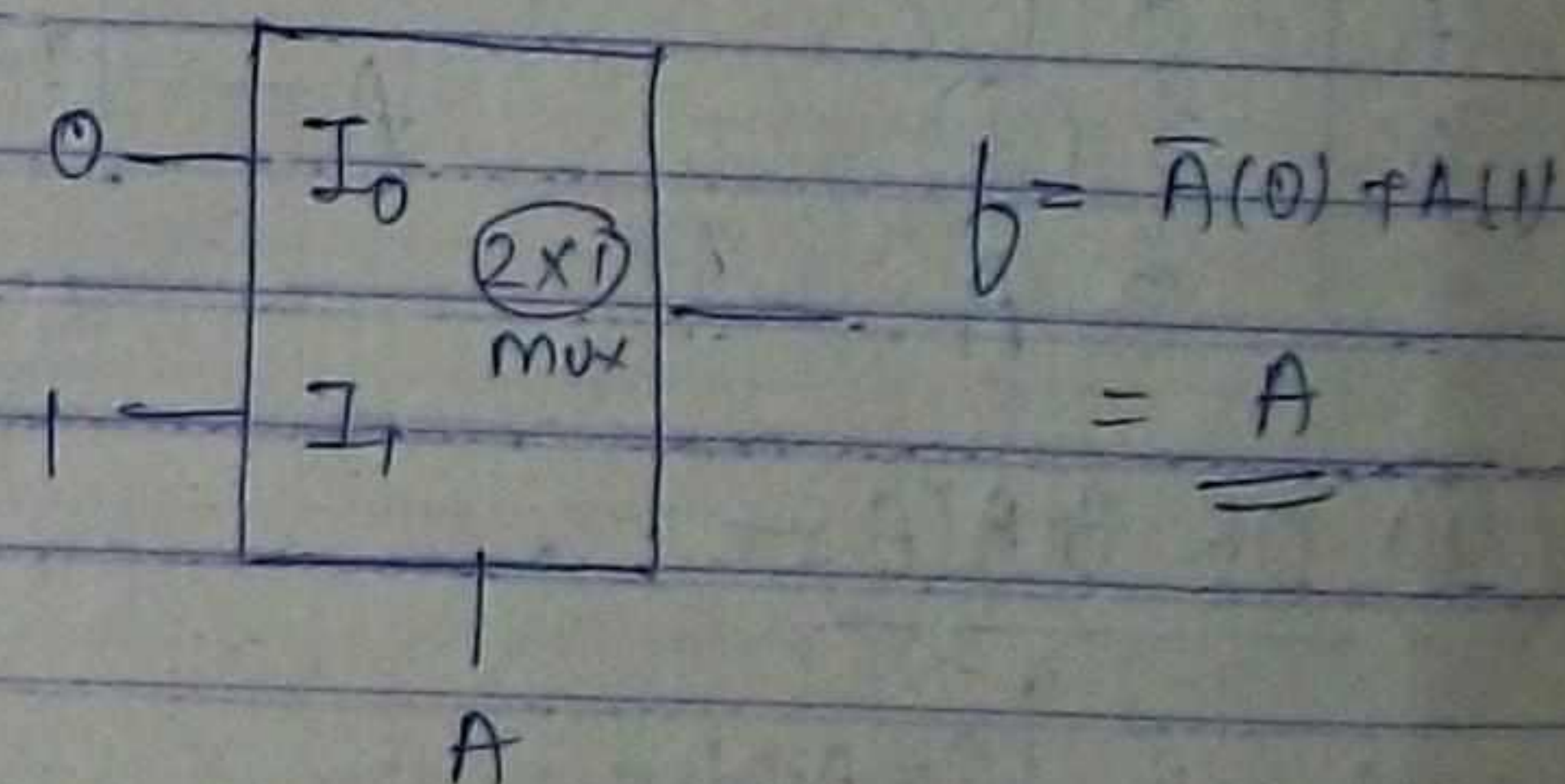
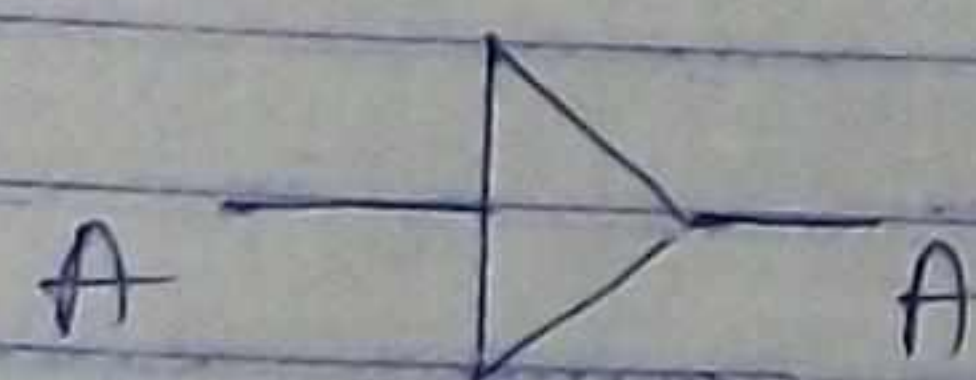
$$f = \bar{E}(I_0) + E(I_1)$$

$$= 0 + \bar{B}(A)$$

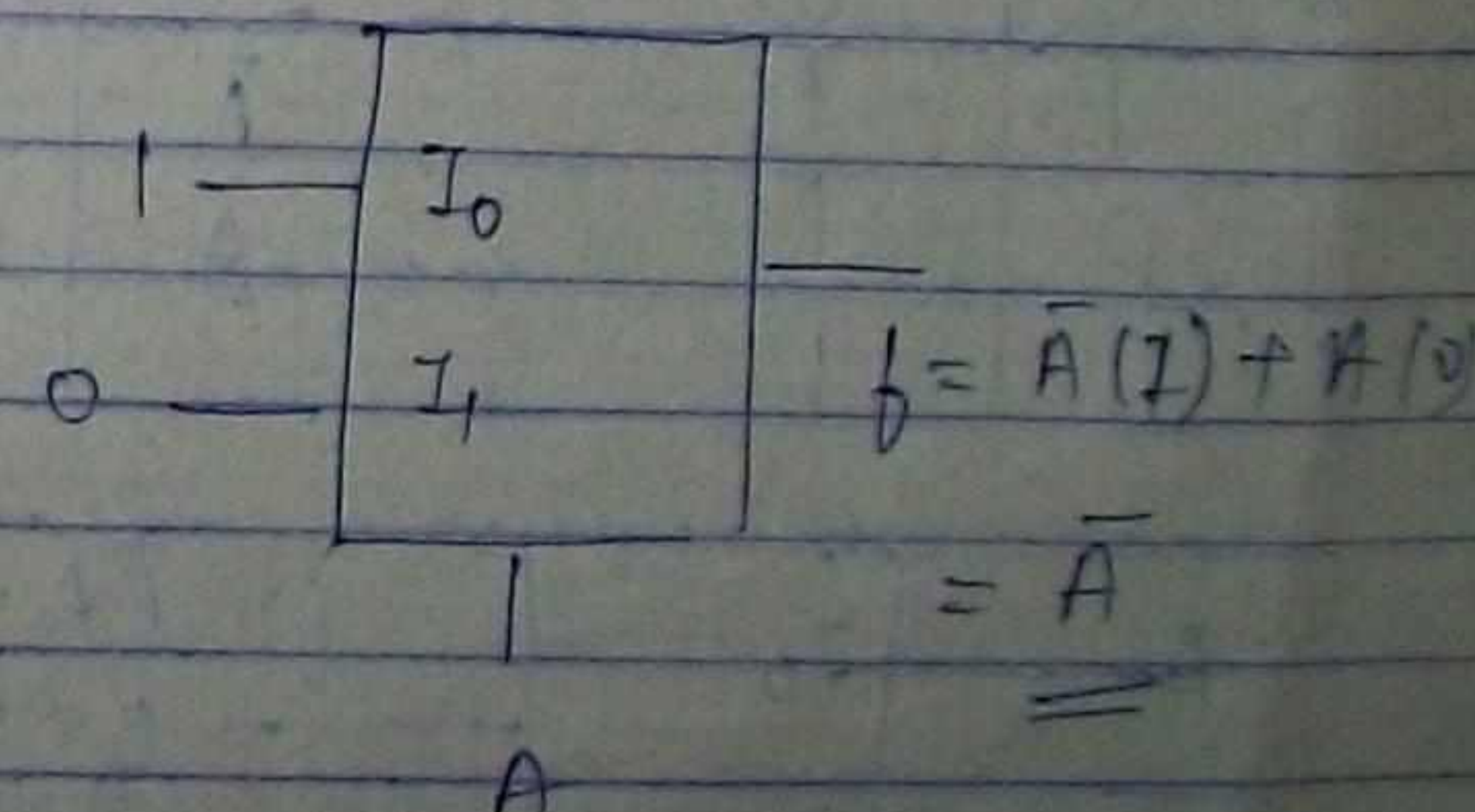
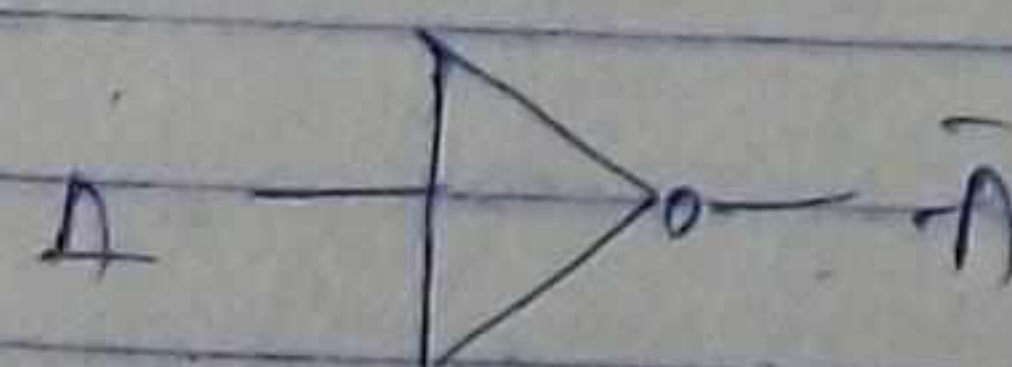
$$= \bar{A}B + A\bar{B}$$

⇒ MUX AS UNIVERSAL LOGIC CIRCUIT -

① BUFFER -



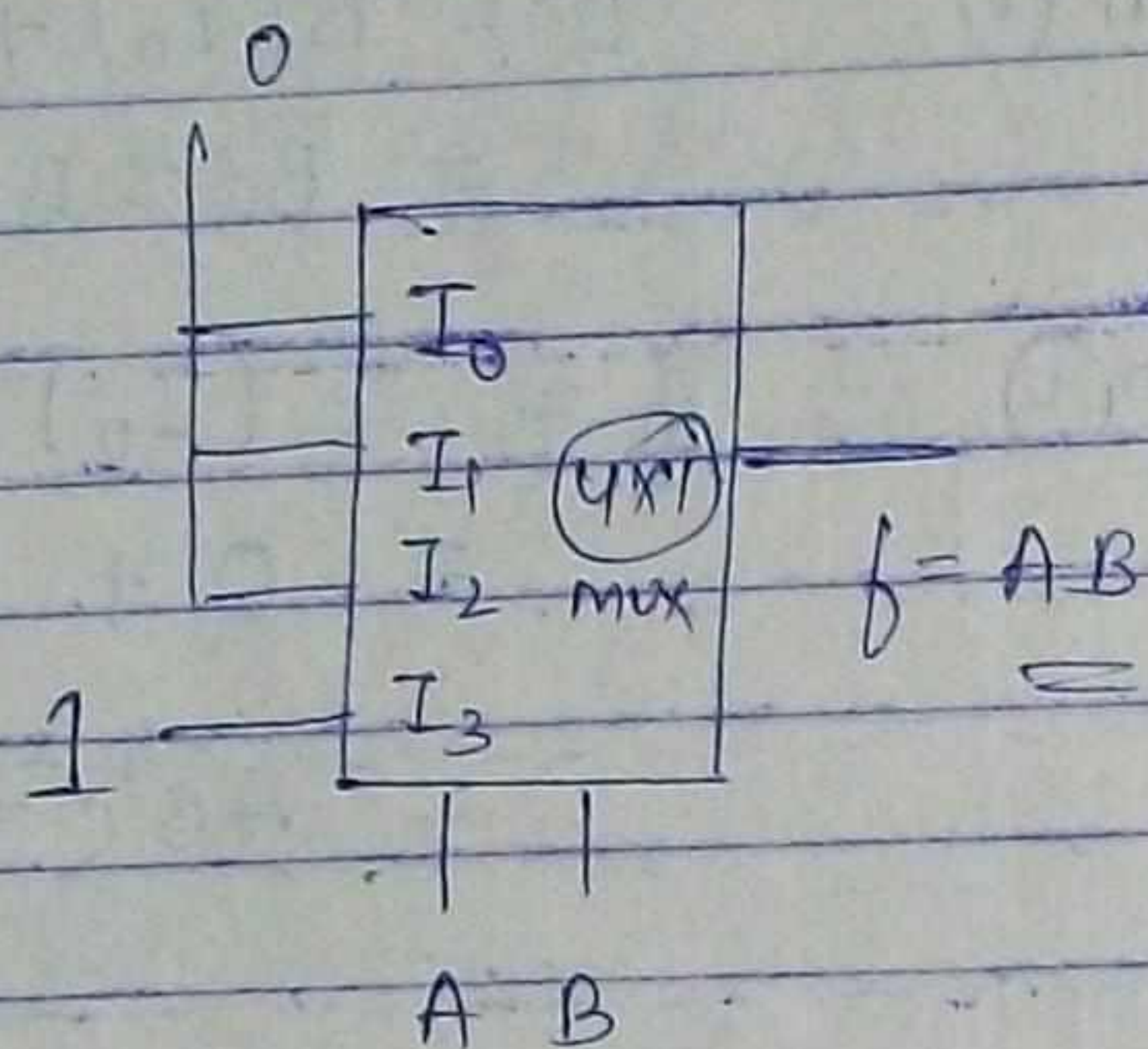
② NOT GATE -



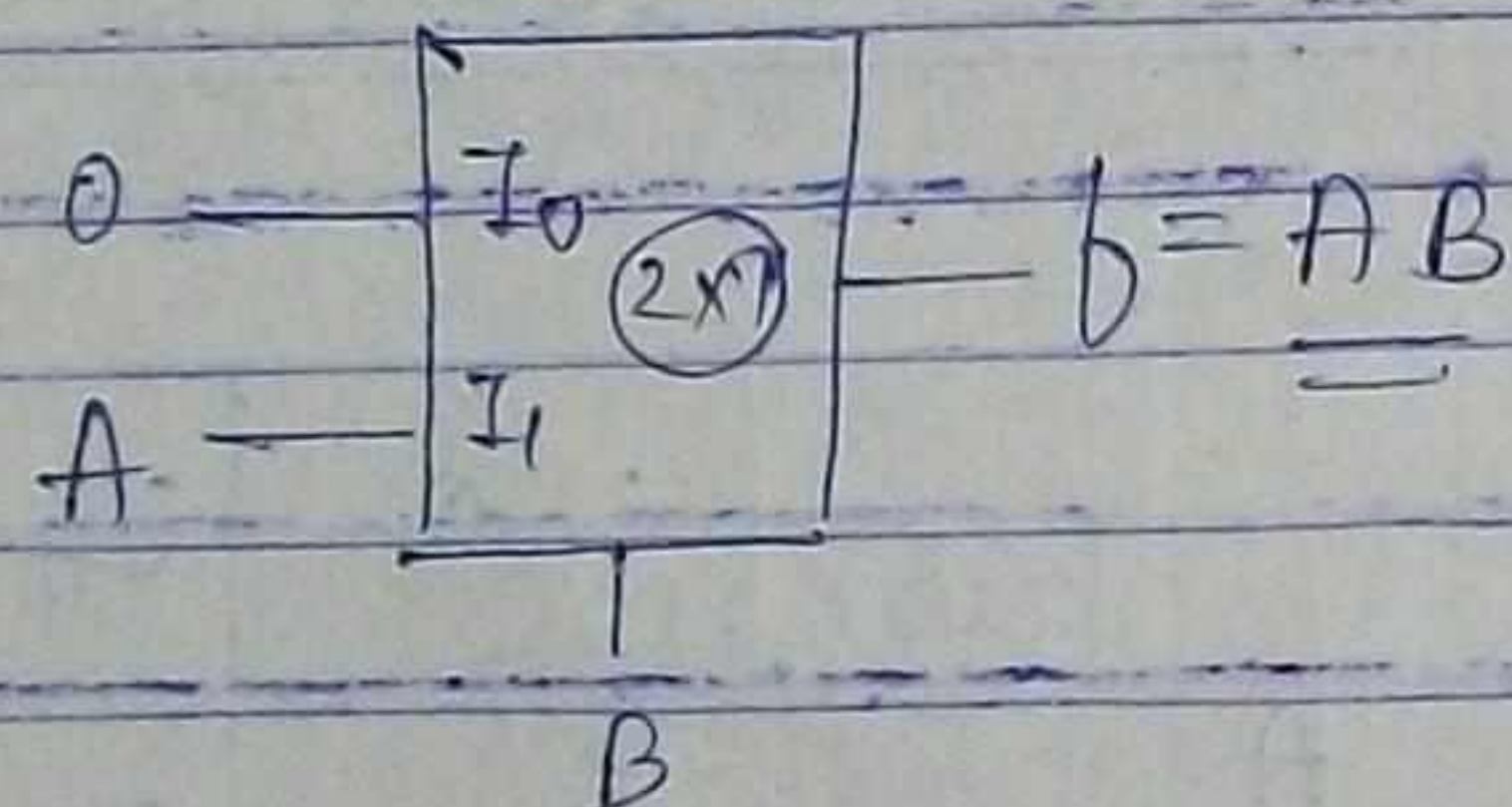
③ AND GATE -

A	B	$f = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

$$f = \sum m(3)$$



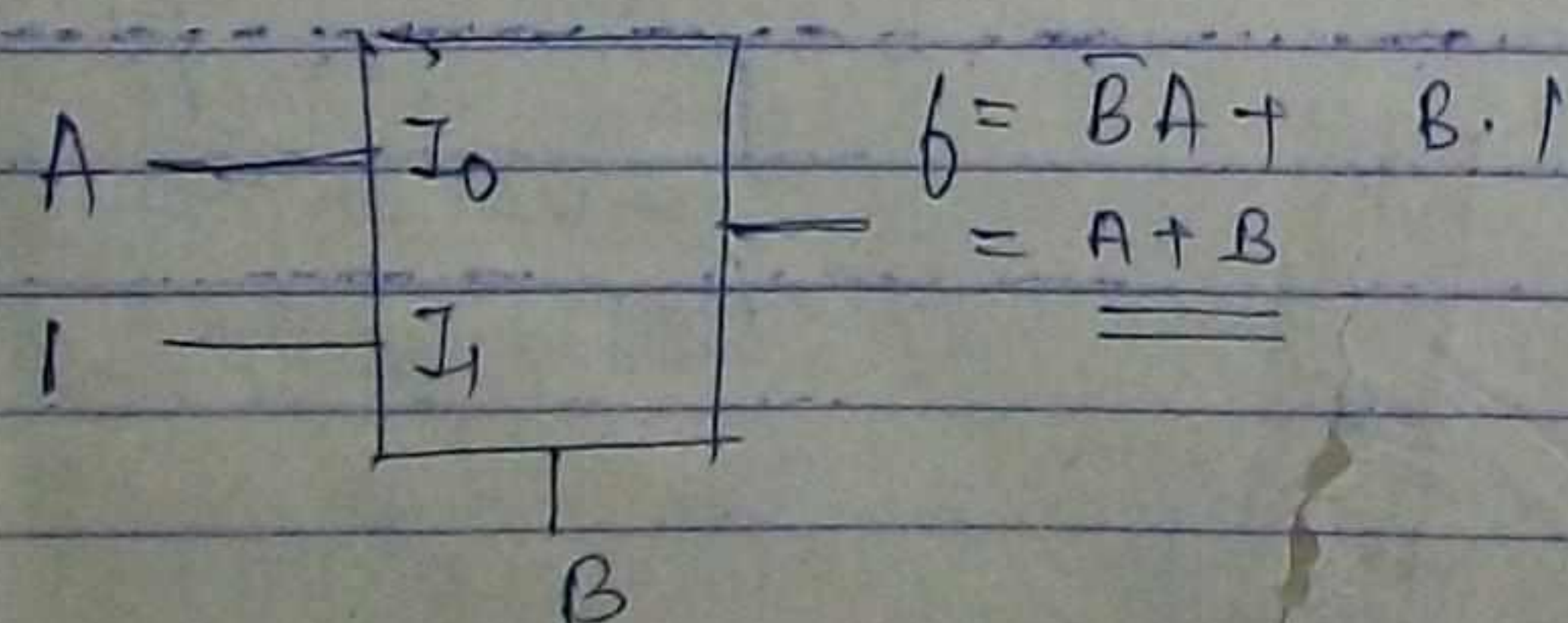
A	I0	I1
0	0	1
1	2	3
0	A	



④ OR GATE -

A	B	$f = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

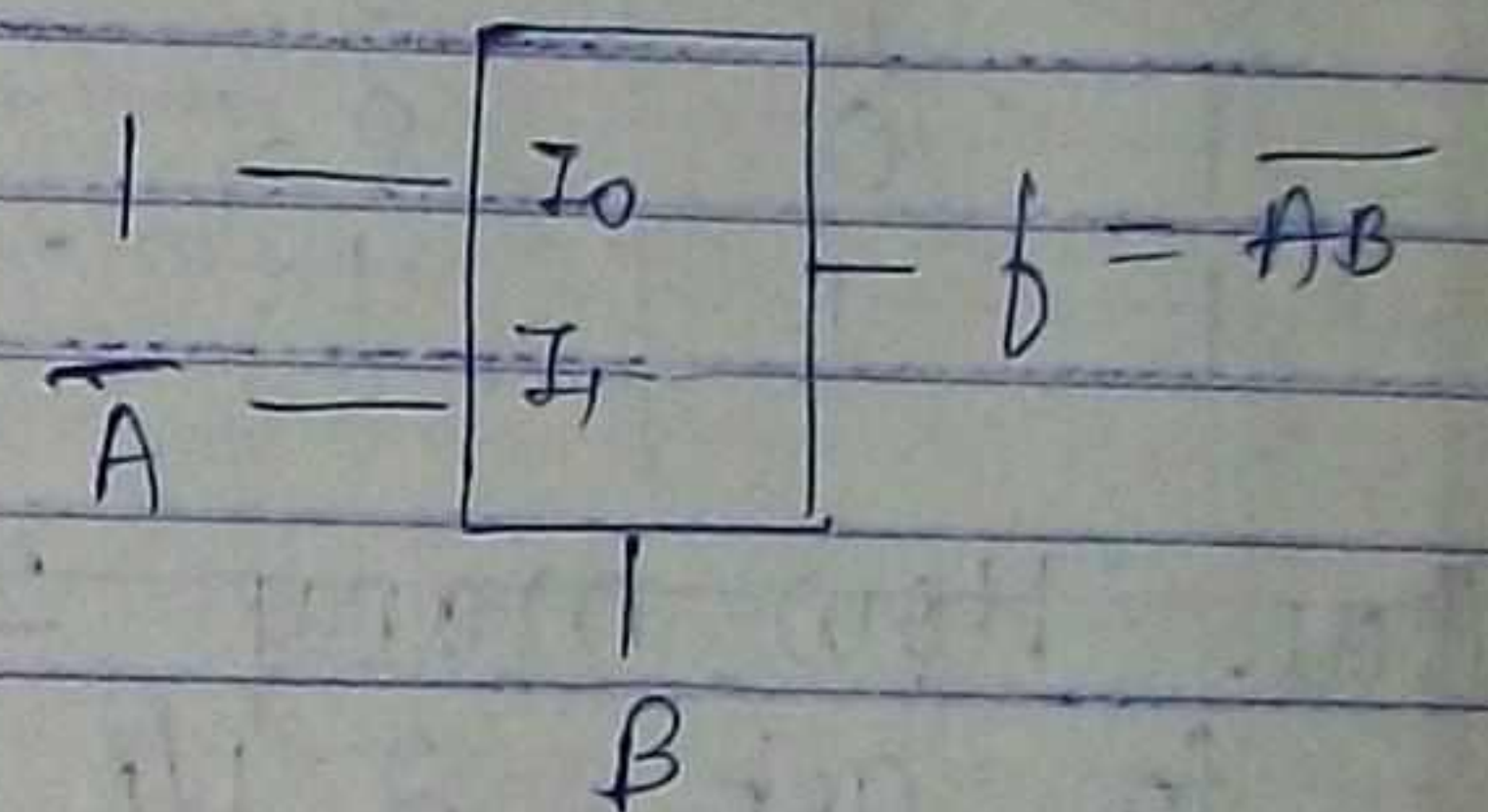
A	I0	I1
0	0	1
1	2	3
0	A	



⑤ NAND GATE -

A	B	$y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

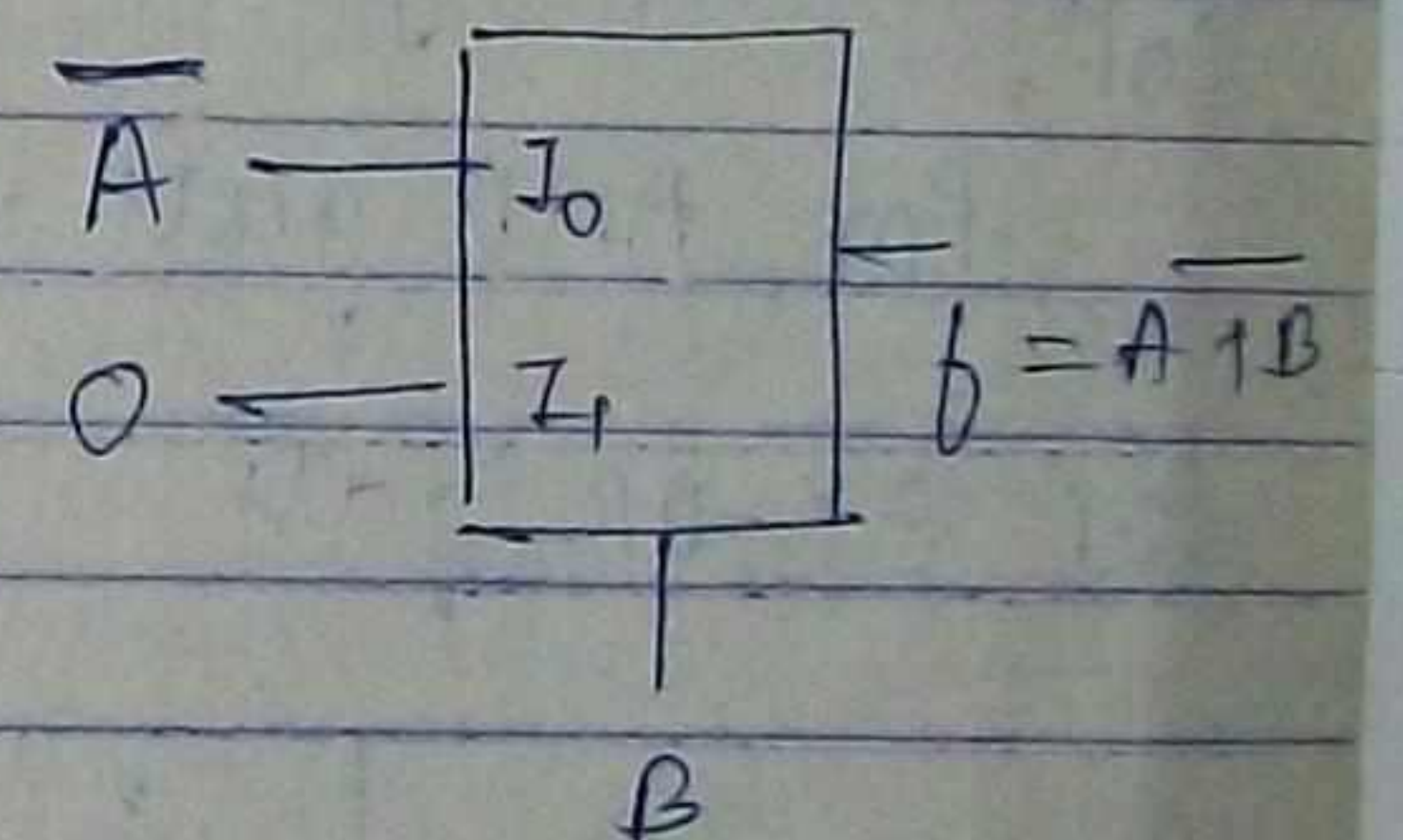
A	I0	I1
0	0	1
1	2	3
0	A	



⑥ NOR GATE -

A	B	$y = \overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

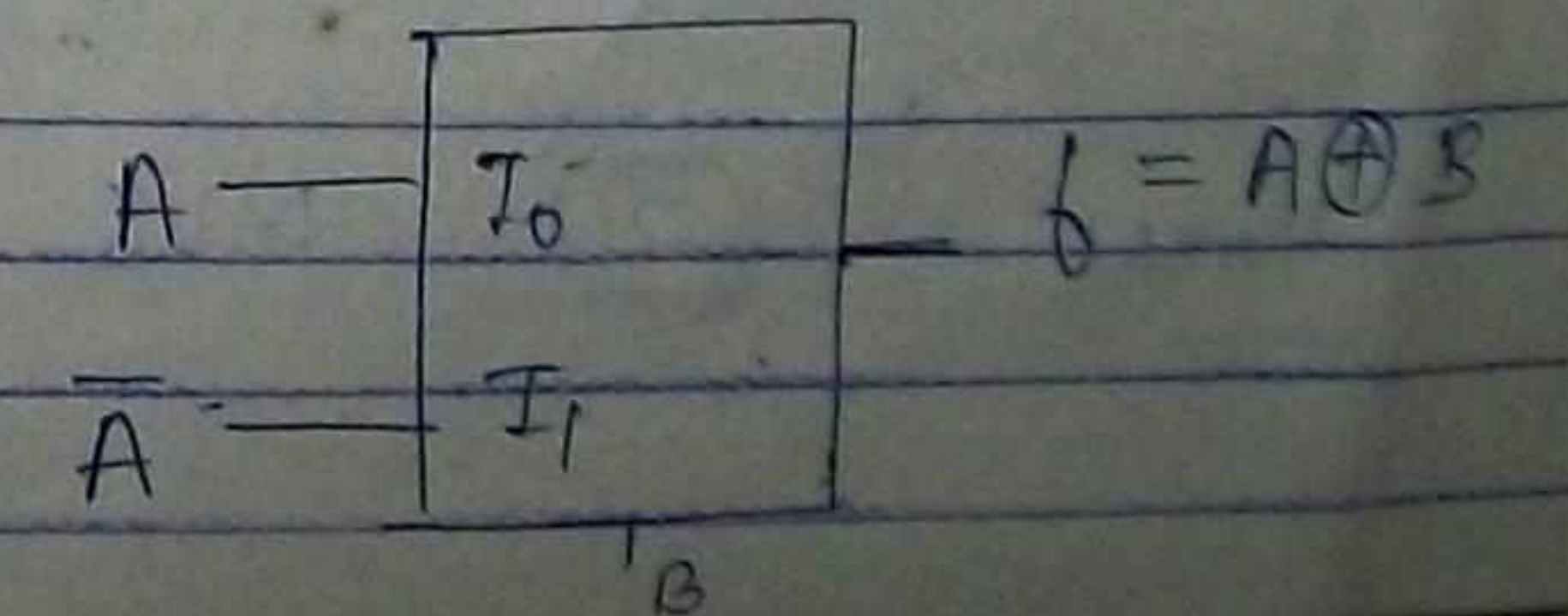
A	I0	I1
0	0	1
1	2	3
0	A	



⑦ EX-OR GATE

A	B	$y = \overline{A \cdot B} + A \cdot \overline{B}$
0	0	0
0	1	1
1	0	1
1	1	0

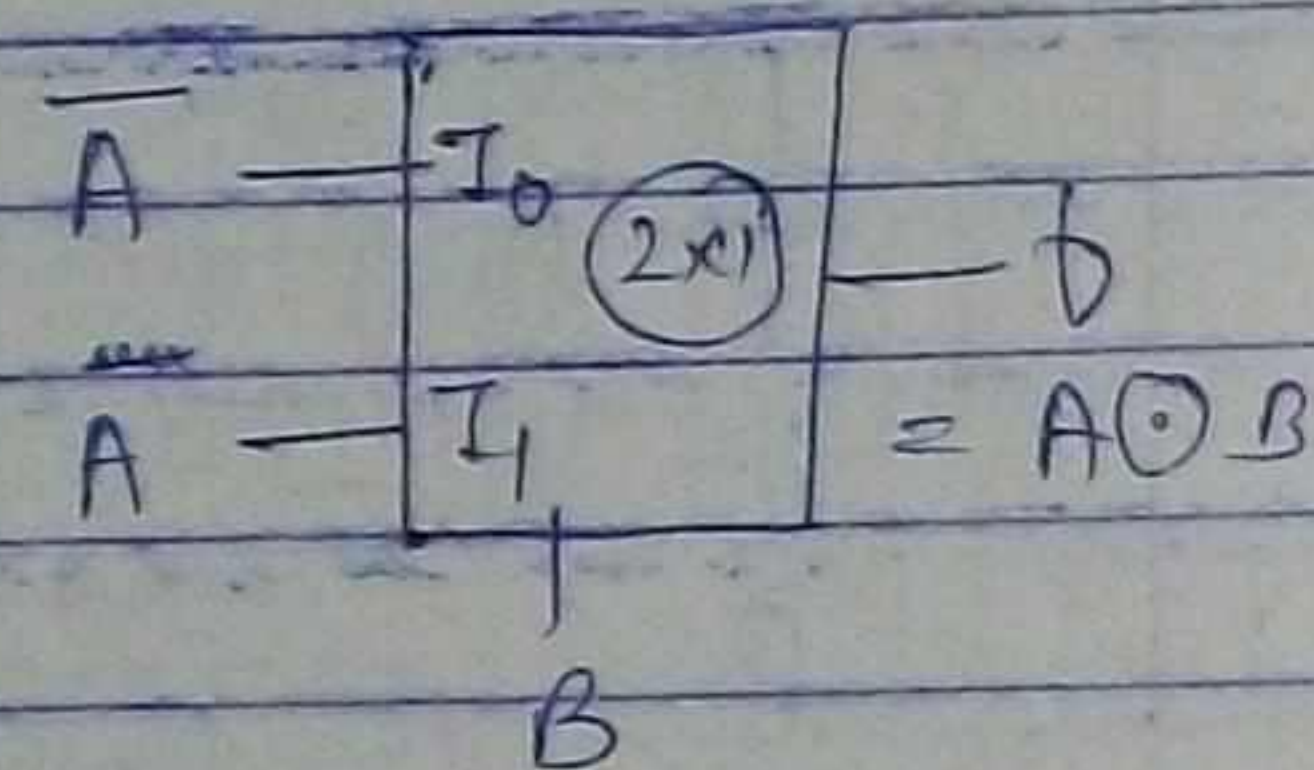
A	I0	I1
0	0	1
1	2	3
0	A	



⑧ EX-NOR —

A	B	$Y = A \odot B$
0	0	1
0	1	0
1	0	0
1	1	1

	I_0	I_1
$\bar{A}$	1	0
A	0	1
$\bar{A}$	A	A



Ques. How many 2x1 mux is Required to get 2 i/p AND gate, 2 i/p EX-OR gate.

Solⁿ:

for AND gate — 1 mux (A, B)

for EX-OR gate (A, A, B)

So 2 mux

Ques. To implement half adder how many 2x1 mux is required.

Solⁿ:

EX-OR — 2 mux

AND — 1 mux

TOTAL ③ — mux

SHORT TRICK —

2x1 mux

NOT — 1

AND — 1

OR — 1

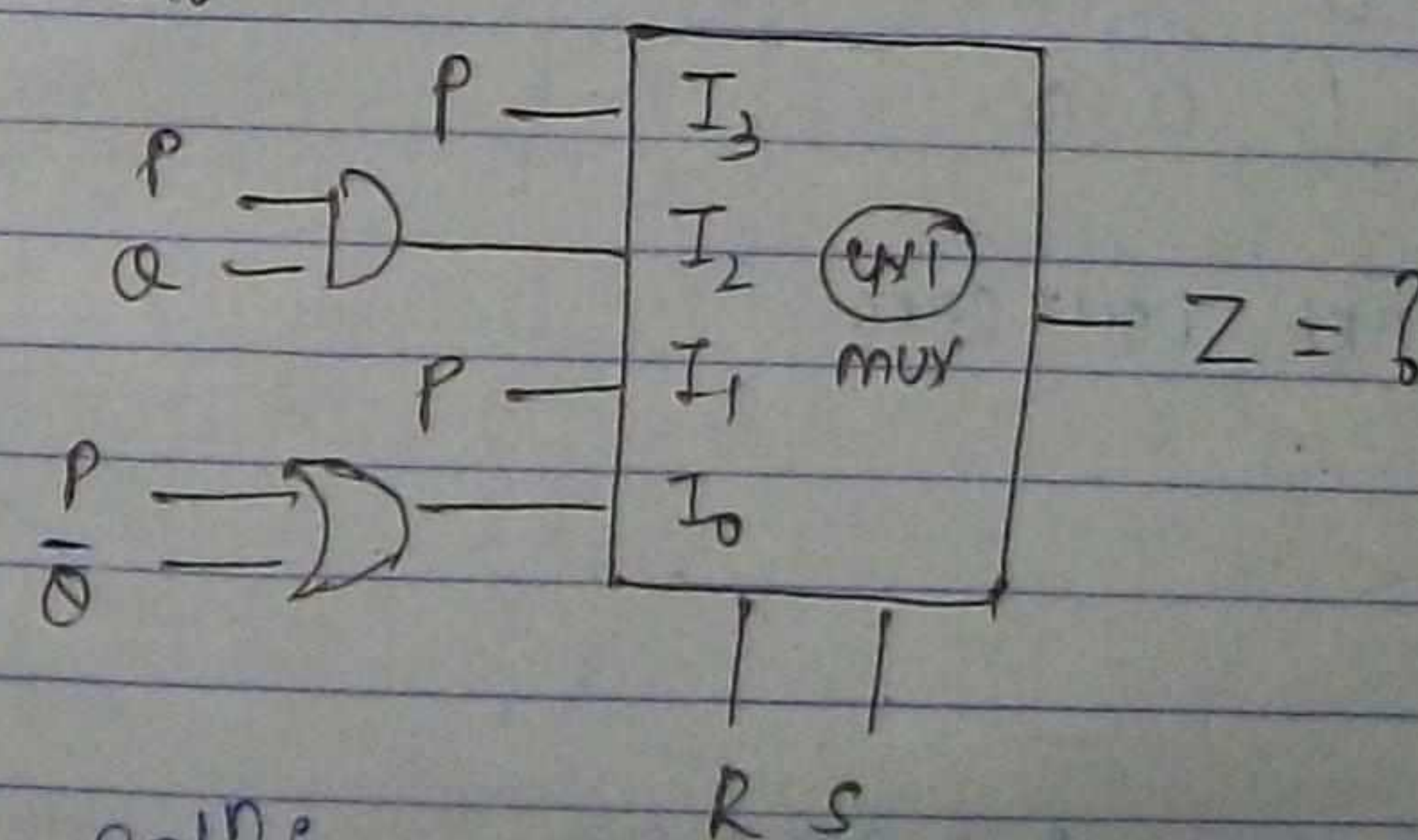
EX-OR — 2

EX-NOR — 2

NAND — 2

NOR — 2

Ques.



Solⁿ:

	$\bar{R}\bar{S}$	$\bar{R}S$	RS	$R\bar{S}$
$\bar{P}\bar{Q}$	1			
$\bar{P}Q$	0			
$P\bar{Q}$	1	1	1	1
PQ	1	1	1	1

$$\begin{aligned}
 Z &= \bar{R}\bar{S}(P+\bar{Q}) + \bar{R}S(P) + R\bar{S}(PQ) + RS(P) \\
 &= \bar{R}\bar{S}P + \bar{R}\bar{S}\bar{Q} + \bar{R}SP + R\bar{S}PQ + RS P \\
 &= P\bar{Q} + PS + \bar{Q}\bar{R}\bar{S}
 \end{aligned}$$

NUMBER SYSTEM

BCD
BINARY
EXCESS-3
GRAY

⇒ CODE CONVERTER -

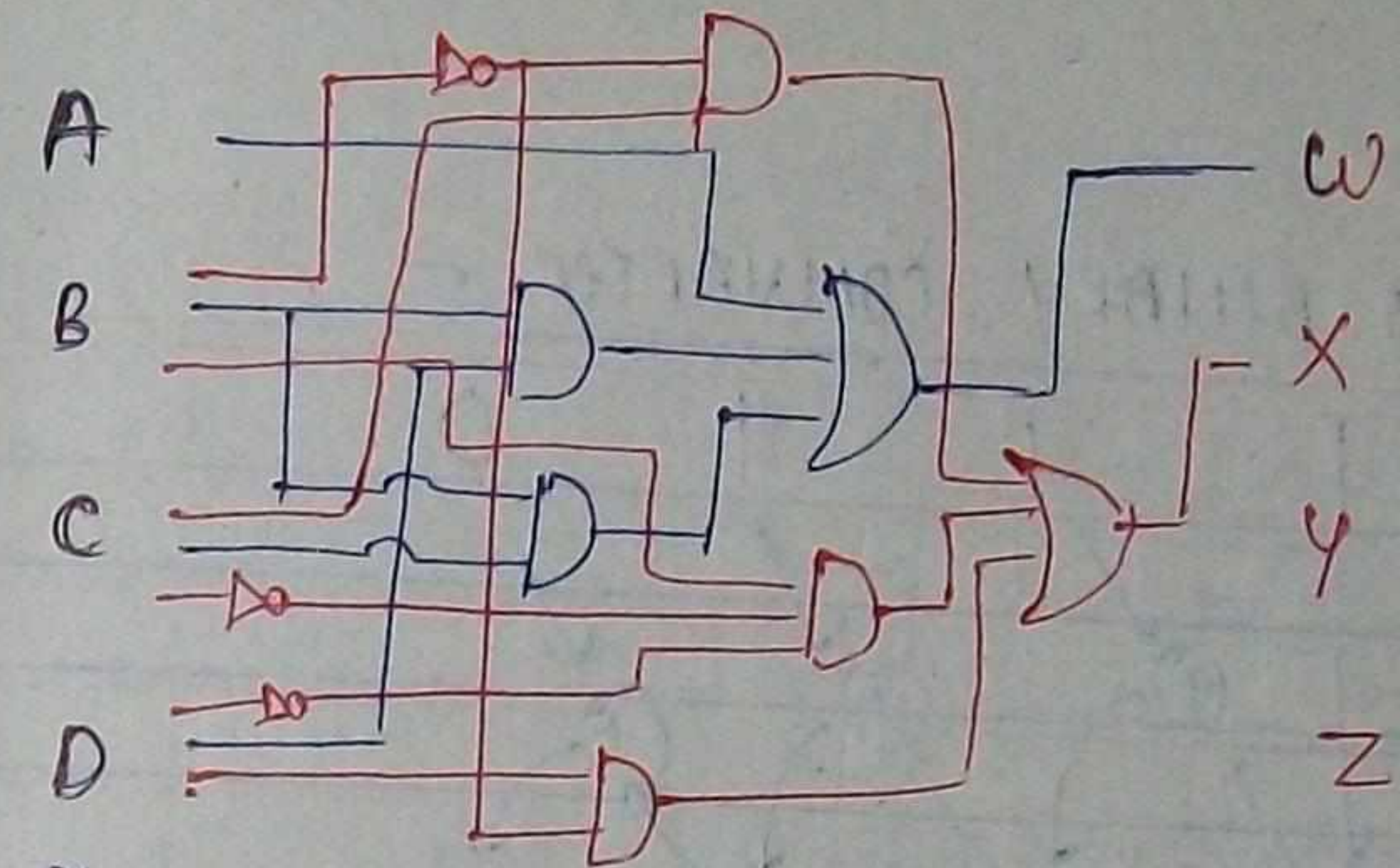
① BCD → EXCESS-3 CONVERTER -

BCD (I/P)				EXCESS-3 (O/P)			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

10-15 → TAKE DONT CARE
INVALID BCD CODE

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				
$\bar{A}B$		1	1	1
AB	X	X	X	X
$A\bar{B}$	1	1	X	X

$$W = A + BD + BC$$



Similarly,

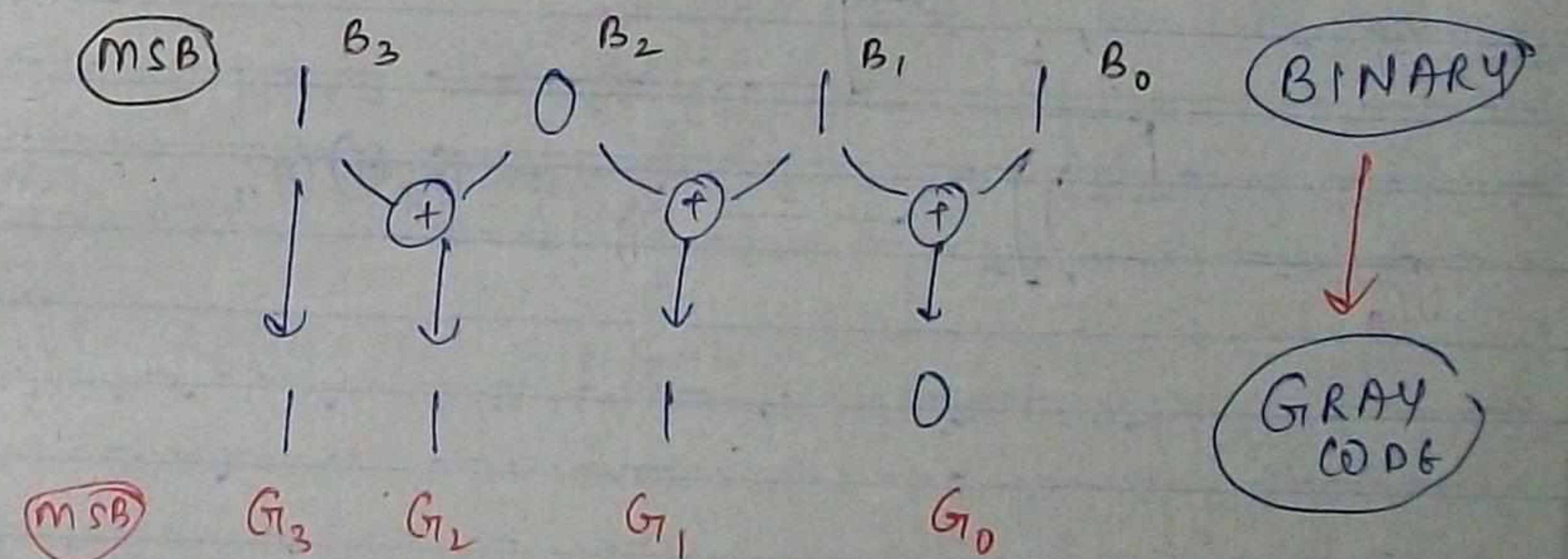
$$W = A + BC + BD$$

$$X = \bar{B}C + \bar{B}D + B\bar{C}\bar{D}$$

$$Y = CD + \bar{C}\bar{D}$$

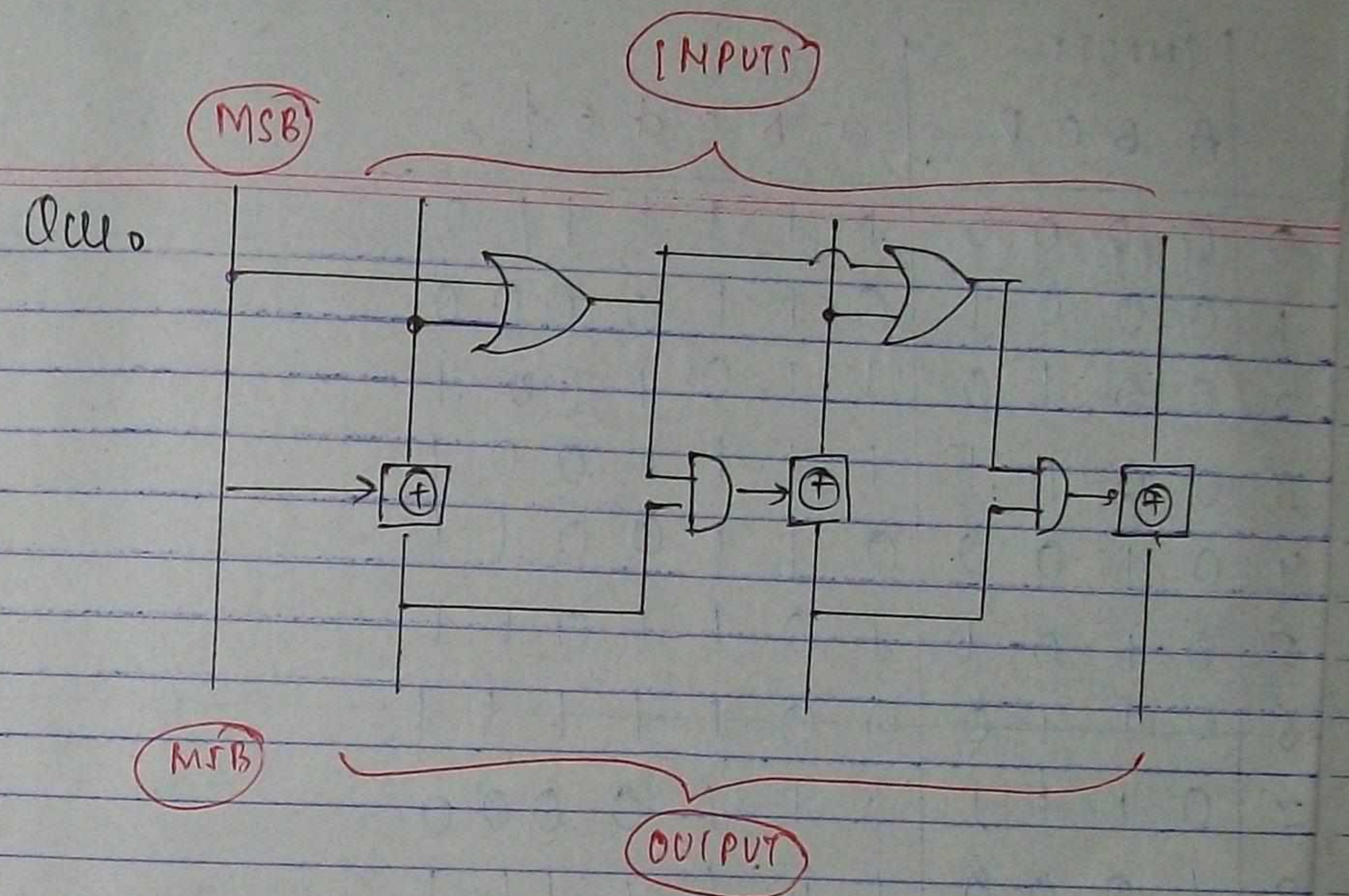
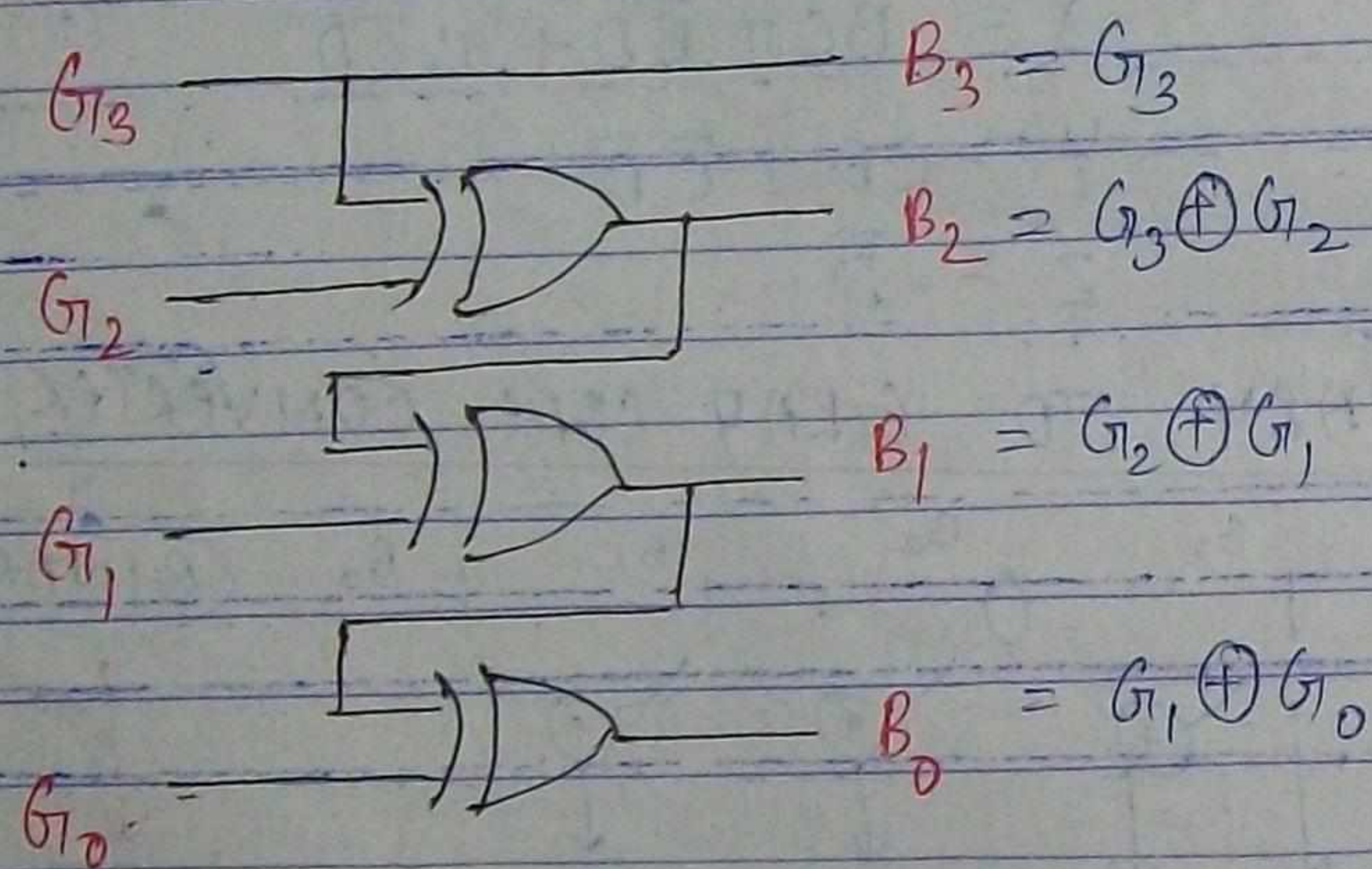
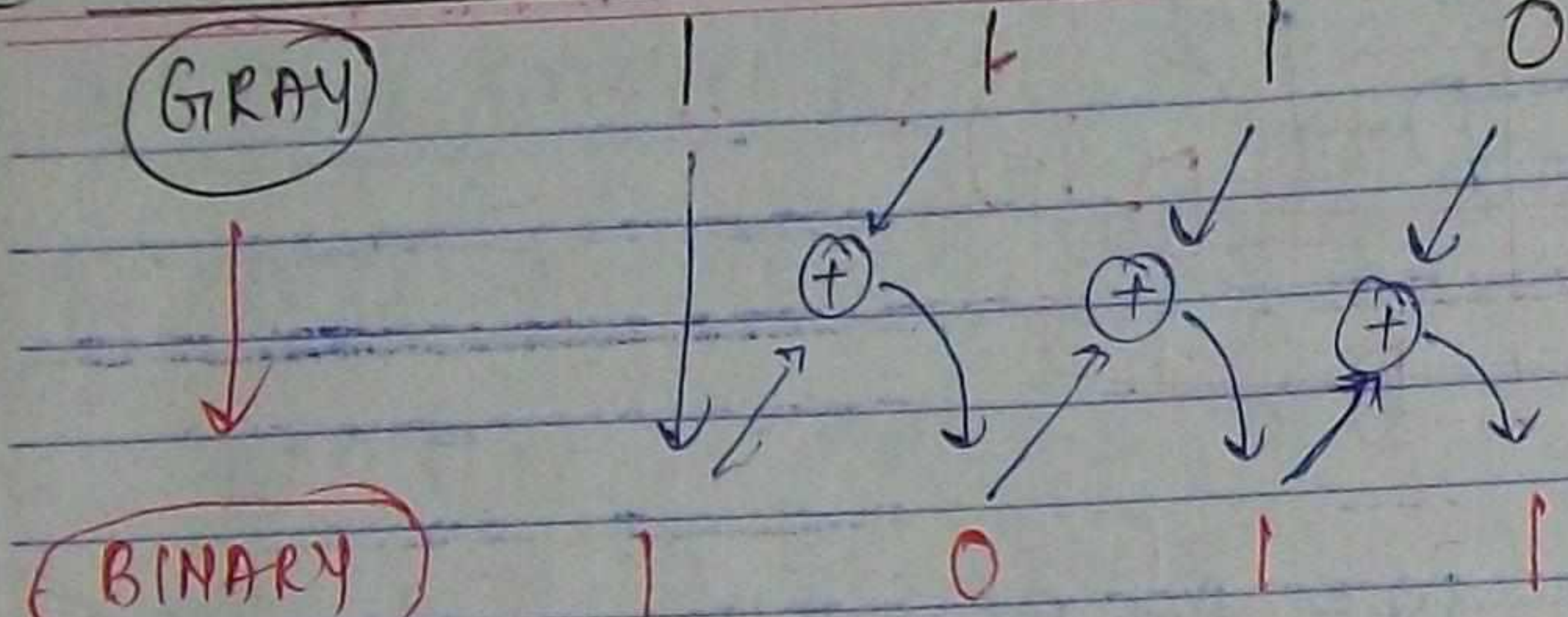
$$Z = \bar{D}$$

② BINARY TO GRAY CODE CONVERTER -



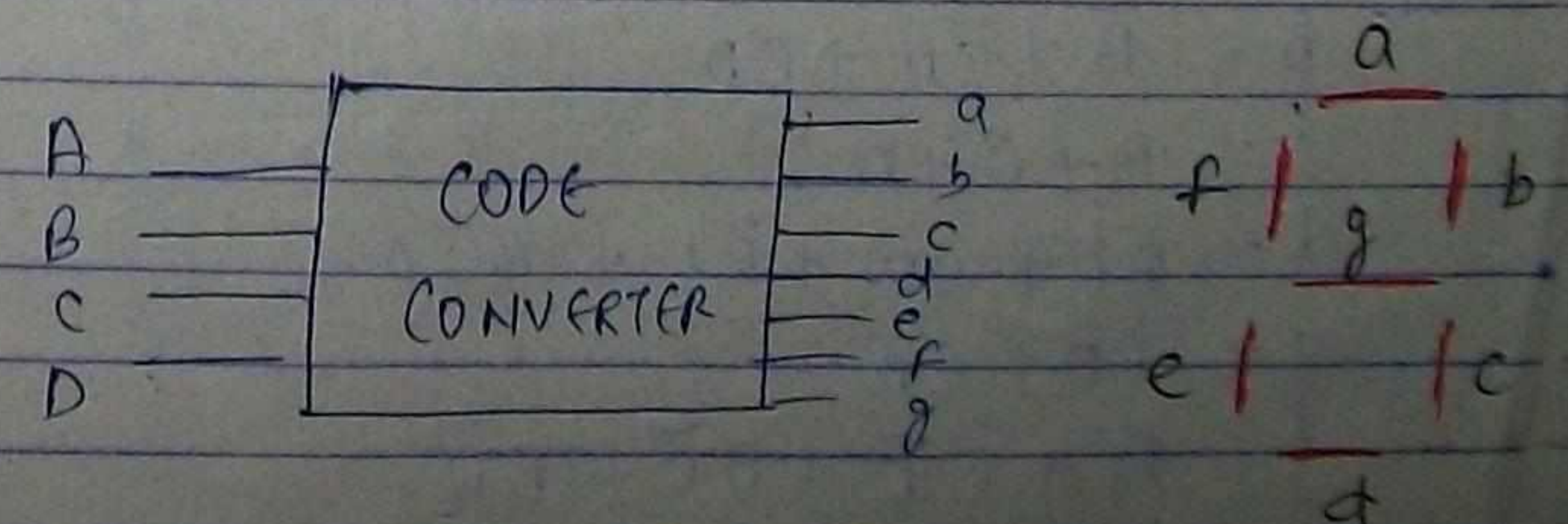
$$\begin{aligned} B_3 & \rightarrow G_3 = B_3 \\ B_2 & \rightarrow G_2 = B_3 \oplus B_2 \\ B_1 & \rightarrow G_1 = B_2 \oplus B_1 \\ B_0 & \rightarrow G_0 = B_1 \oplus B_0 \end{aligned}$$

③ GRAY CODE TO BINARY CONVERTER -



- ① BCD TO BINARY
- ② BINARY TO EXCESS-3
- ③ EX-3 TO GRAY
- ④ GRAY TO BINARY

⇒ BCD TO 7 SEGMENT CONVERTER -



INPUTS											
	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	0	1
$\bar{A}B$	0	0	0	1
$A\bar{B}$	X	X	X	X
AB	1	0	X	X

for e $e = C\bar{D} + \bar{B}\bar{D}$

$a = \bar{B}\bar{D} + BD + CD + A$
 $b = \bar{B} + \bar{C}\bar{D} + CD$
 $c = B + \bar{C} + D$
 $d = \bar{B}\bar{D} + C\bar{D} + \bar{B}C + B\bar{C}D$
 $e = \bar{B}\bar{D} + C\bar{D}$
 $f = A + \bar{C}\bar{D} + \bar{B}\bar{C} + B\bar{D}$
 $g = A + B\bar{C} + \bar{B}C + C\bar{D}$

Ques. Design circuit

P_2	P_1	O/P
0	0	0
0	1	2
1	0	5
1	1	E

Solⁿ:

P_2	P_1	a	b	c	d	e	f	g
0	0	1	1	1	1	1	1	0
0	1	1	1	0	1	1	0	1
1	0	1	0	1	1	0	1	1
1	1	1	0	0	1	1	1	1

$a = 1$, $b = \bar{P}_2$, $c = \bar{P}_1$
 $d = 1$, $e = P_1 + \bar{P}_2$, $f = \bar{P}_1 + P_2$, $g = P_1 + P_2$

Ques. Previous question

- (Ques 1) (1) $g = \bar{P}_1 + P_2$ $d = c + e$ (2) $g = P_1 + P_2$ $d = c + e$
 (3) $g = \bar{P}_1 + P_2$ $e = b + c$ (4) $g = P_1 + P_2$ $e = b + c$

(Ques 2) How many min. no. of NOT, 2-input OR gates

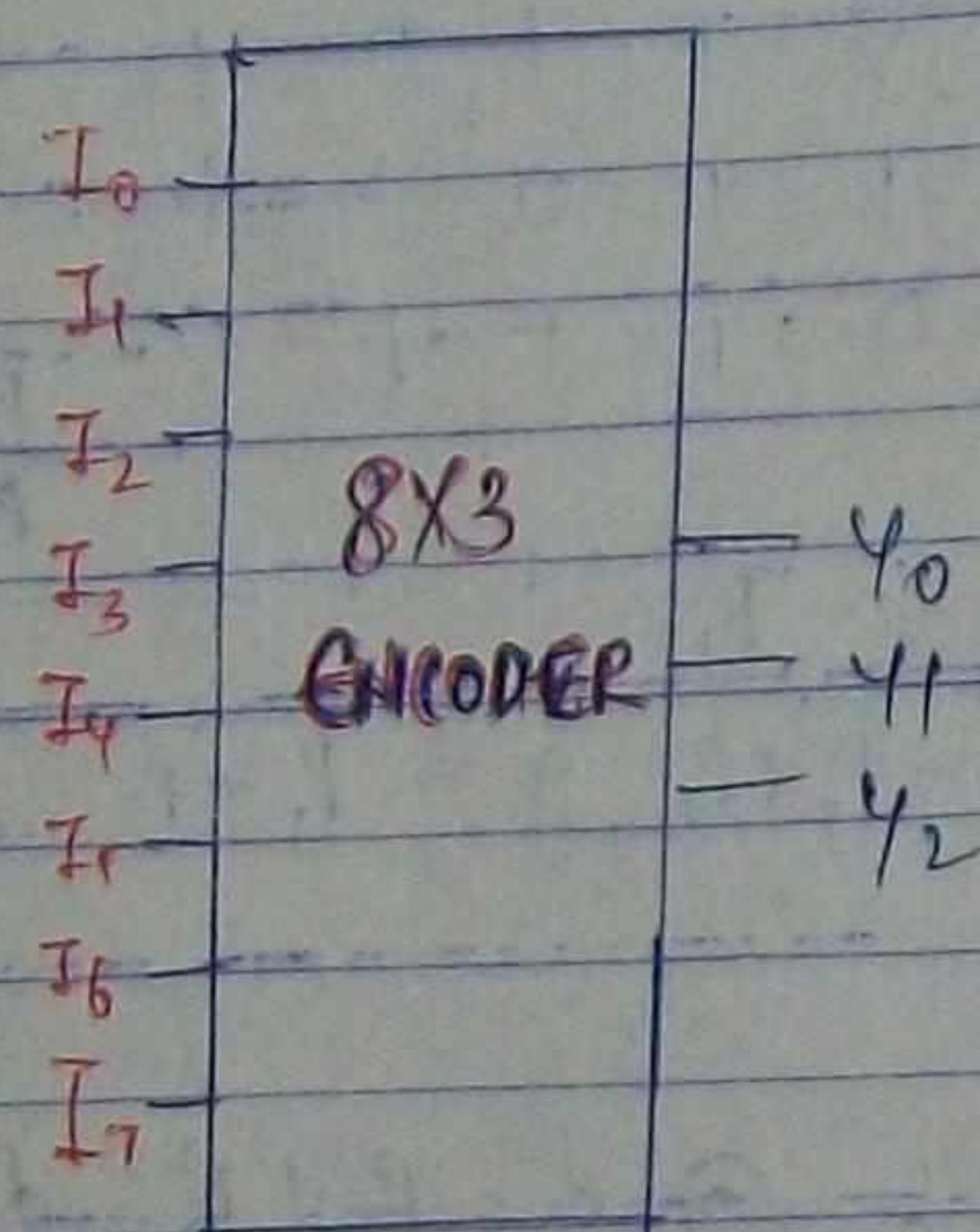
- (1) 3 NOT 4 OR (2) 2 NOT 4 OR
 (3) 1 NOT 3 OR (4) 2 NOT 3 OR

Solⁿ:

(1) $g = P_1 + P_2$
 $d = c + e = \bar{P}_1 + \bar{P}_2$
 $d = c + e = \bar{P}_1 + \bar{P}_2 + P_1$
 $= 1$
 Ans (2)

⇒ ENCODER —

① OCTAL TO BINARY ENCODER —

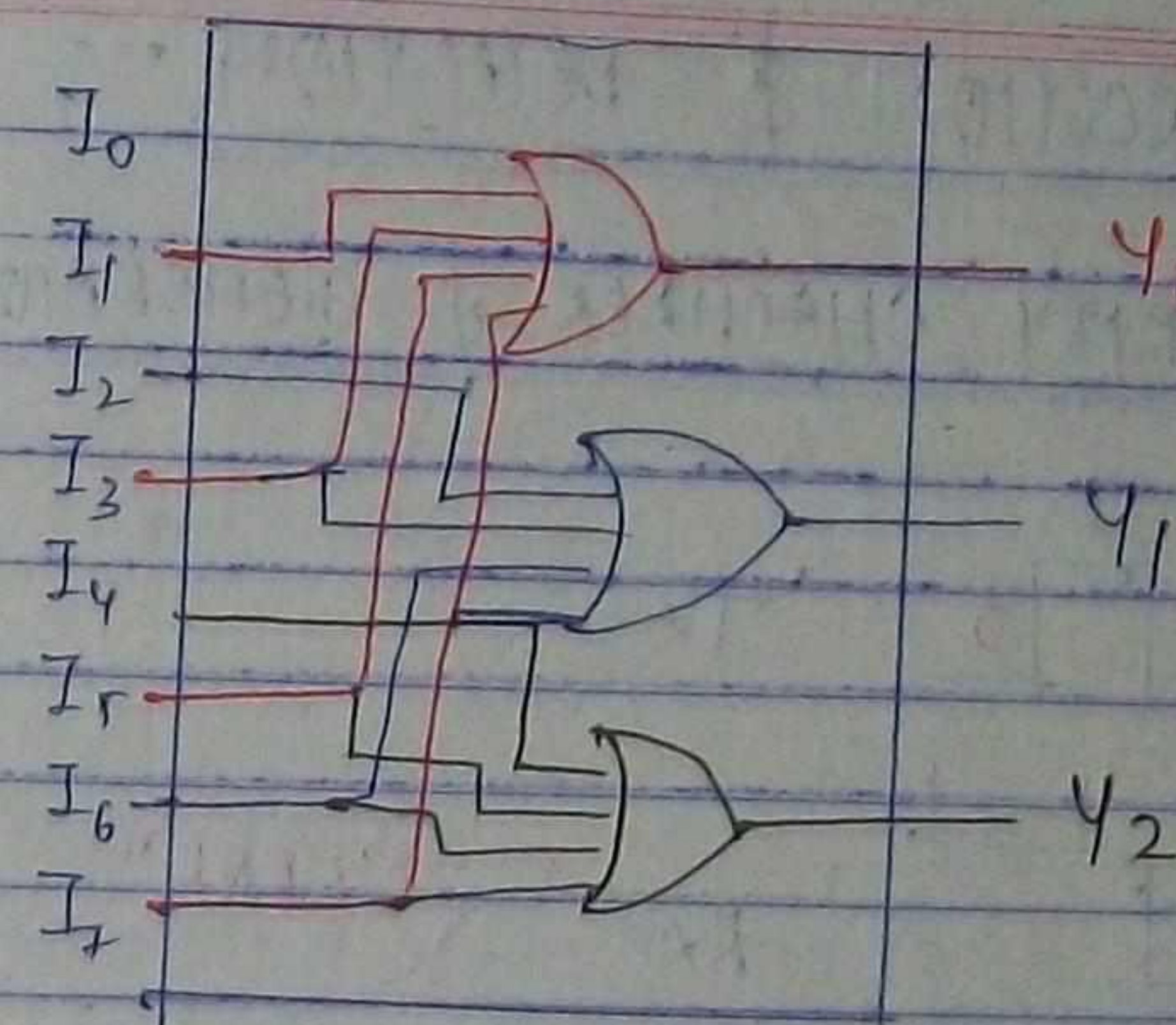


I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$$Y_0 = I_1 + I_3 + I_5 + I_7$$

$$Y_1 = I_2 + I_3 + I_6 + I_7$$

$$Y_2 = I_4 + I_5 + I_6 + I_7$$



⊗ In a multiplexers AND, OR gates are preferable.

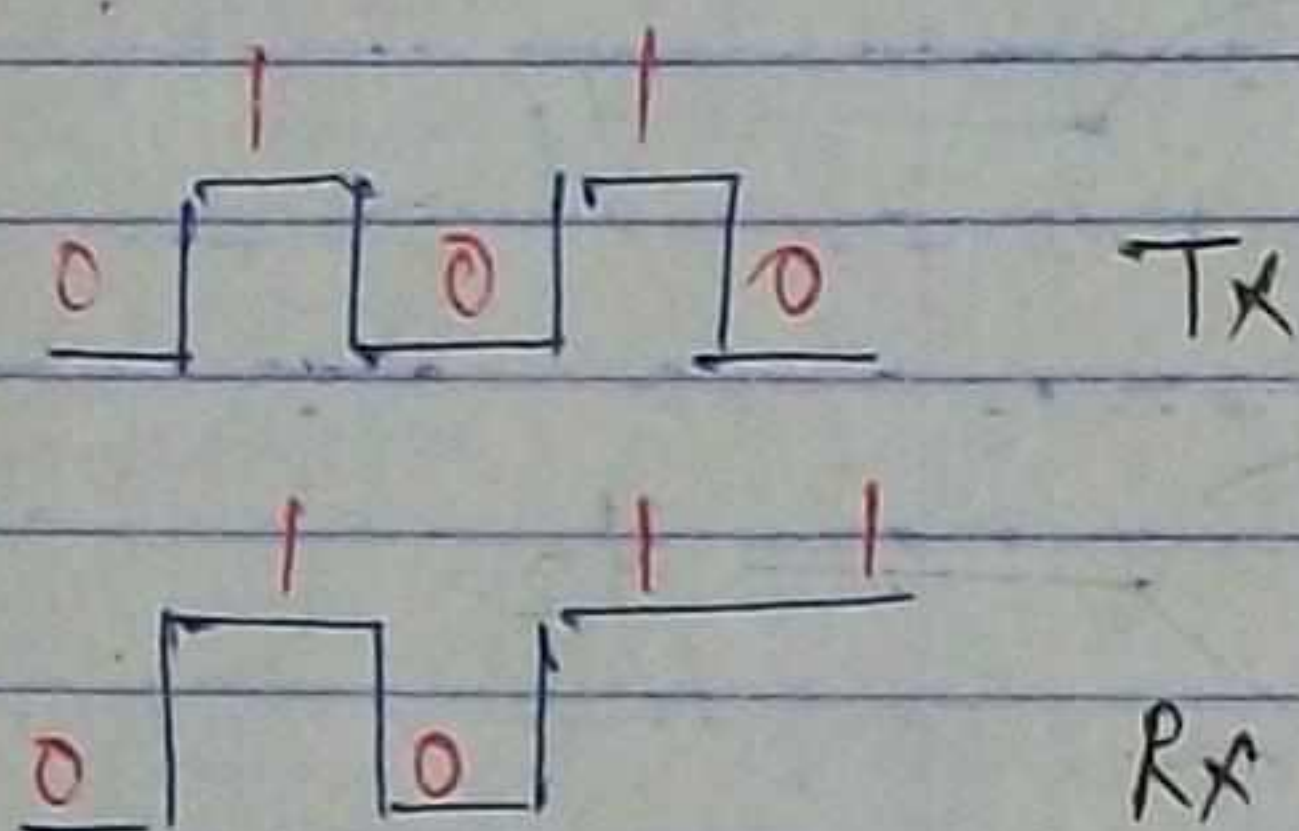
⊗ In demultiplexers (Decoder) — AND gates

⊗ In Encoders — OR gates

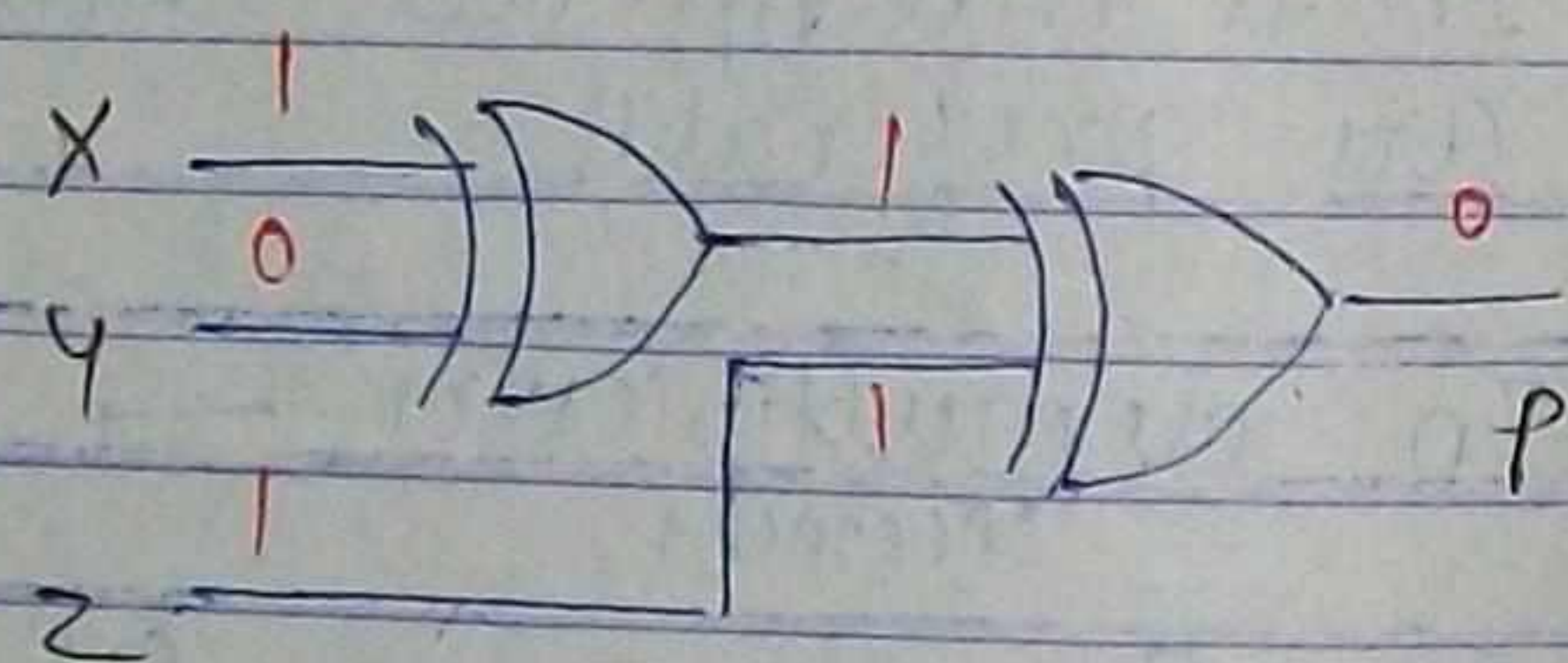
15/10/11

⇒ ERROR CORRECTION & DETECTION -

⇒ EVEN PARITY CHECKER & GENERATOR -



X	Y	Z	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

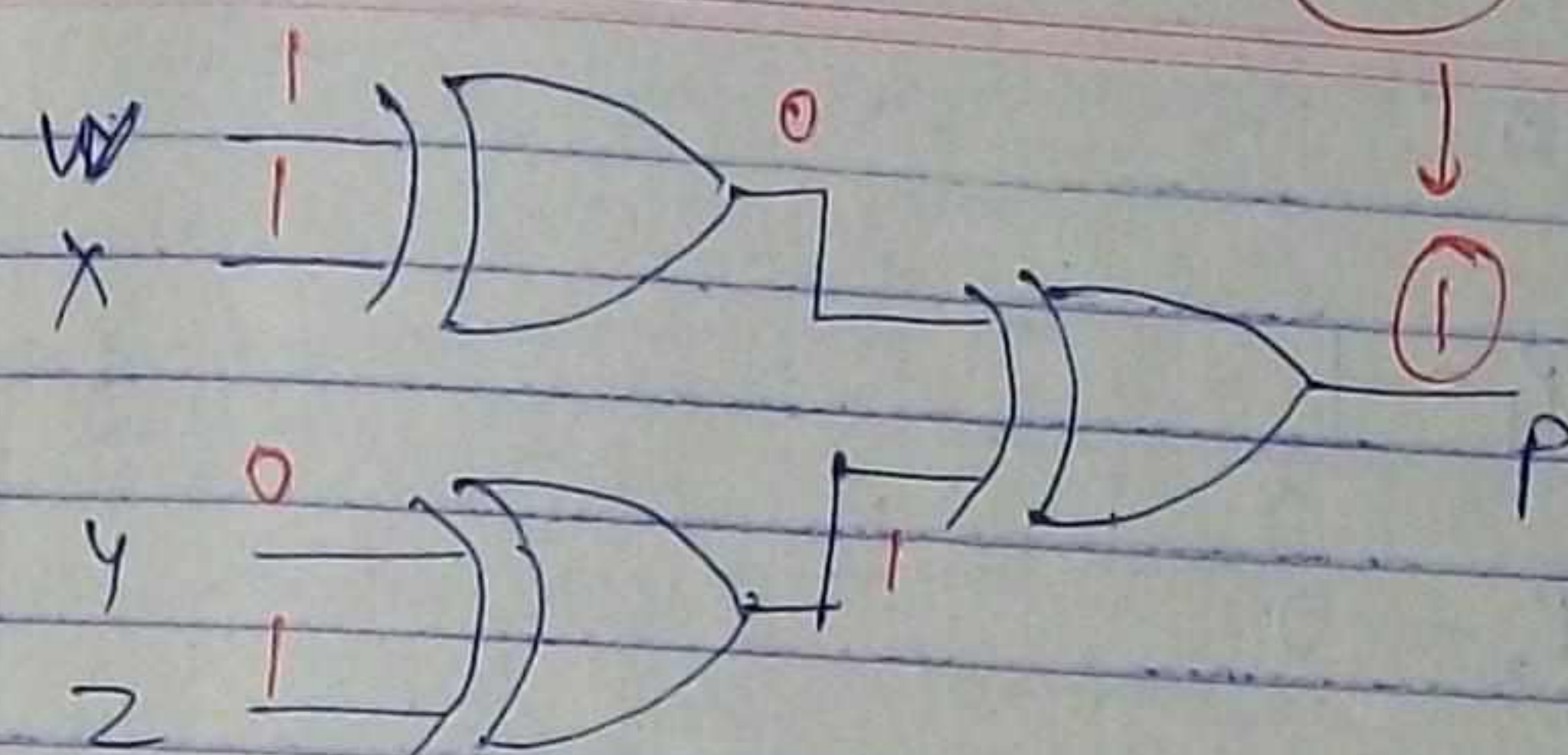
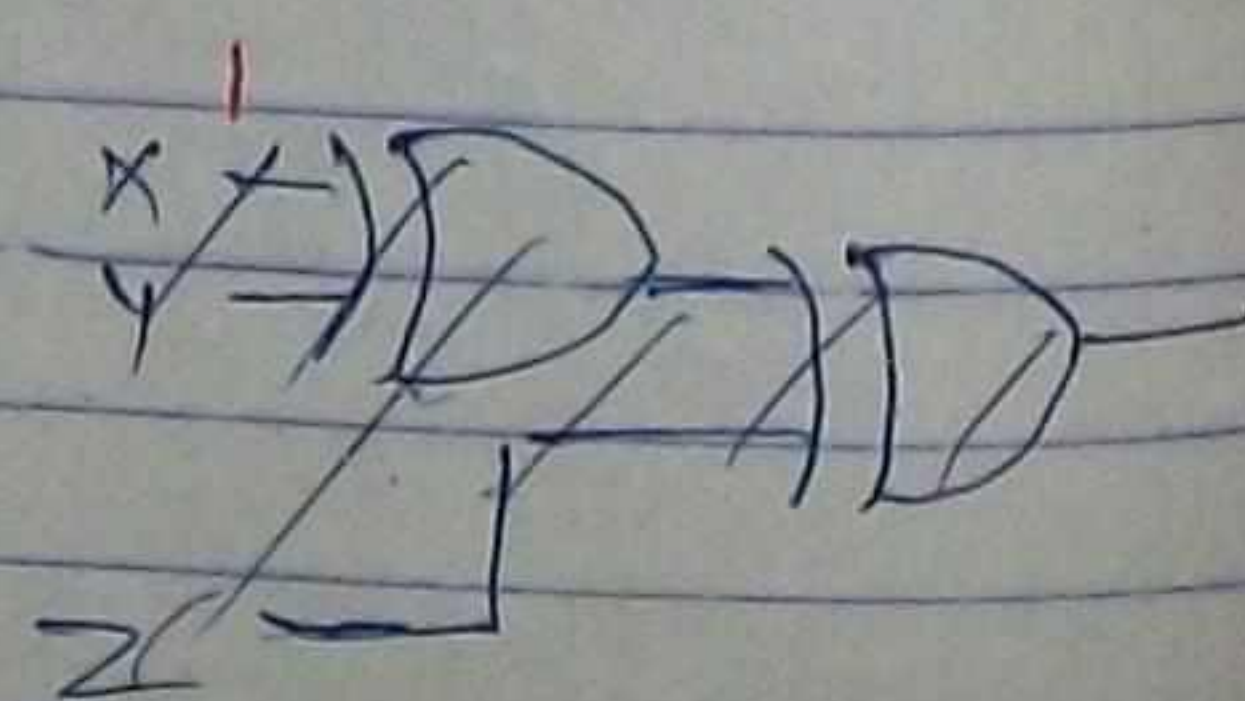


EVEN PARITY GENERATOR

for checking

Rx - P = 1010 → no error

P = 1101 → **ERROR**



Odd no. of 1
ERROR

EVEN PARITY CHECKER

⇒ HAMMING CODE -
① FOR TRANSMISSION -
DATA : 1 0 1 1

Even Parity

P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_4	P_2	P_1
							X	Y	Z
0	1	1	0	0	1	1	0	0	0
							1	0	0
							2	0	1
							3	0	1
							4	1	0
							5	1	0
							6	1	0
							7	1	1

$P_1 \rightarrow (1, 3, 5, 7)$ (P_1 101) even parity $P_1 = 0$

$P_2 \rightarrow (2, 3, 6, 7)$ (P_2 111) even parity $P_2 = 1$

$P_4 \rightarrow (4, 5, 6, 7)$ (P_4 011) even parity $P_4 = 0$

Tx. data is : 0 1 1 0 0 1 1

Ex - DATA - 1101

$P_1 \rightarrow (P_1 111) \quad P_1 = 1$

$P_2 \rightarrow (P_2 101) \quad P_2 = 0$

$P_4 \rightarrow (P_4 101) \quad P_4 = 0$

$P_1 \quad P_2 \quad D_3 \quad P_4 \quad D_5 \quad D_6 \quad D_7$

1 0 1 0 1 0 1 ← ~~1~~

② RECEIVING -

Rx → 1101101
DATA

$P_1 \quad P_2 \quad D_3 \quad P_4 \quad D_5 \quad D_6 \quad D_7$
1 1 0 1 1 0 1

Even
parity

$P_1 \rightarrow (1, 3, 5, 7) (1011) \text{ error } C=1$

$P_2 \rightarrow (2, 3, 6, 7) (1001) \text{ No error } B=0$

$P_4 \rightarrow (4, 5, 6, 7) (1101) \text{ error } A=1$

∴ A B C

1 0 1

= 5th bit have error

Correct sequence

= 1101001

Ex - Rx DATA - 1001001

$P_1 \rightarrow (1001) \quad C=0$

$P_2 \rightarrow (0001) \quad B=1$

$P_4 \rightarrow (1001) \quad A=0$

A B C

0 1 0

2nd
bit

$P_1 \quad P_2 \quad D_3 \quad P_4 \quad D_5 \quad D_6 \quad D_7$

1 0 0 1 0 0 1

Correct sequence

1101001

⊗ for 12 Bit - (Hamming code)

$P_1 \quad P_2 \quad D_3 \quad P_4 \quad D_5 \quad D_6 \quad D_7 \quad P_8 \quad D_9 \quad D_{10} \quad D_{11} \quad D_{12}$

⊗ for 15 Bit -

$P_1 \quad P_3 \quad D_3 \quad P_4 \quad D_5 \quad D_6 \quad D_7 \quad P_8 \quad D_9 \quad D_{10} \quad D_{11} \quad D_{12} \quad D_{13} \quad D_{14} \quad D_{15}$

⇒ GRAY CODE -

Gray code is a unit distance code

$\begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{matrix}$ } 1 unit distance - (gray code)
 } 2 distance
 } 1 unit distance - (gray code)

0 0 0 0

0 0 0 1

0 0 1 1

0 0 1 0

0 1 1 0

0 1 1 1

0 1 0 1

0 1 0 0

1 1 0 0

1 1 0 1

1 1 1 1

1 1 1 0

1 0 1 0

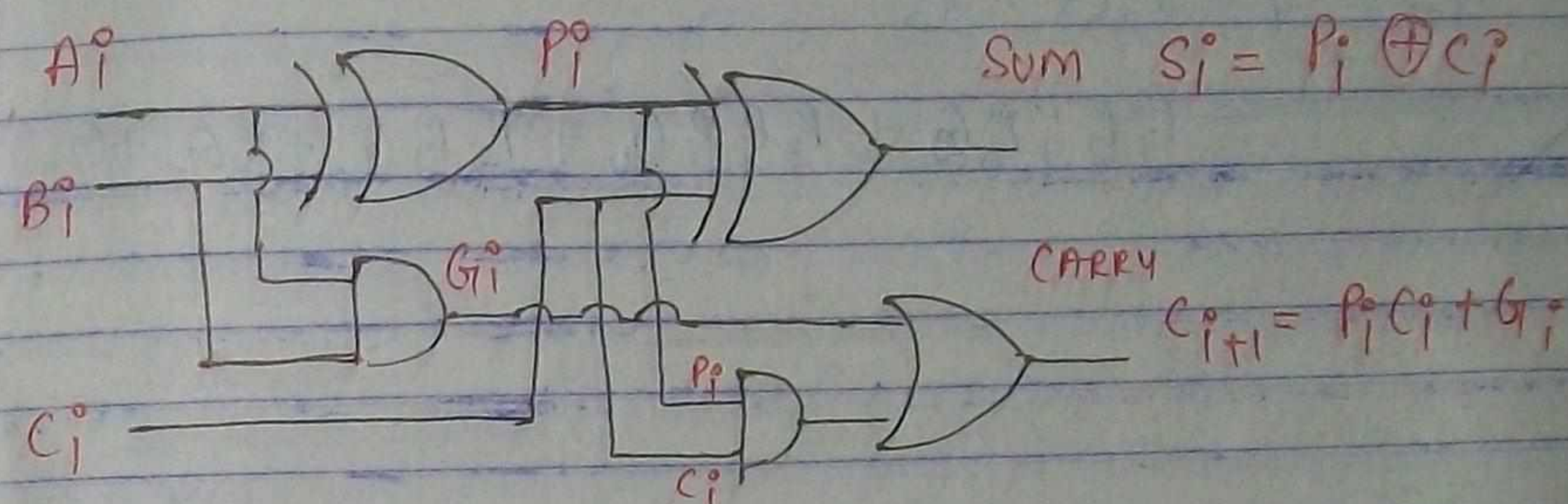
1 0 1 1

1 0 0 1

1 0 0 0

Gray code

⇒ LOOK AHEAD CARRY GENERATOR -



G_i = Carry generator term
 P_i = Carry propagation term,

$P_0 = A_0 \oplus B_0$	$P_0 = A_0 \oplus B_0$	$G_0 = A_0 B_0$	$S_0 = P_0 \oplus C_0$
$G_1 = A_1 B_1$	$P_1 = A_1 \oplus B_1$	$G_1 = A_1 B_1$	$S_1 = P_1 \oplus C_1$
$S_1 = P_1 \oplus C_1$	$P_2 = A_2 \oplus B_2$	$G_2 = A_2 B_2$	$S_2 = P_2 \oplus C_2$
$C_{i+1} = P_i C_i + G_i$	$P_3 = A_3 \oplus B_3$	$G_3 = A_3 B_3$	$S_3 = P_3 \oplus C_3$

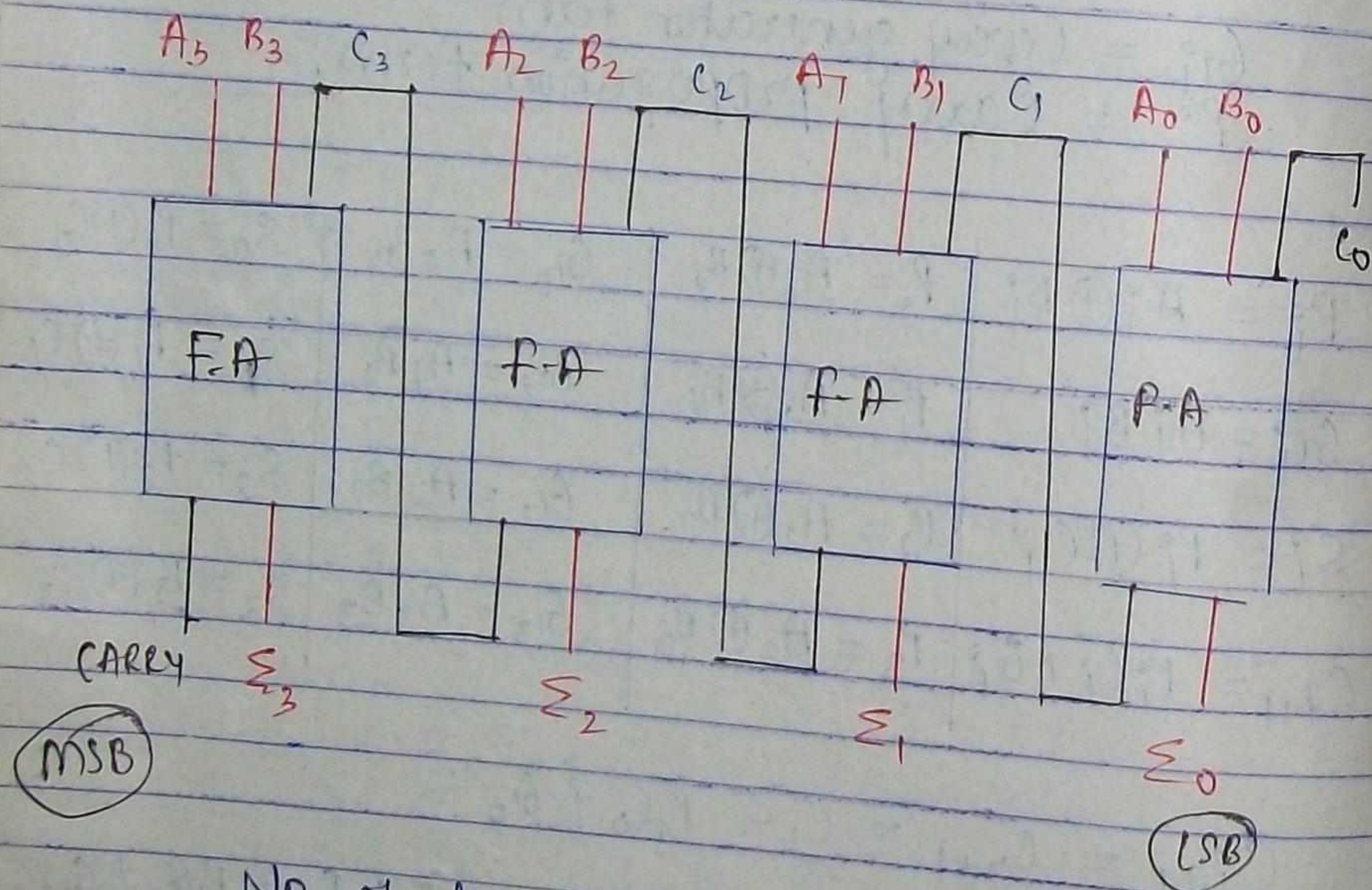
$$C_{i+1} = C_{0+1} = C_1 = P_0 C_0 + G_0$$

$$= C_{1+1} = C_2 = P_1 C_1 + G_1 = P_1 P_0 C_0 + P_1 G_0 + G_1$$

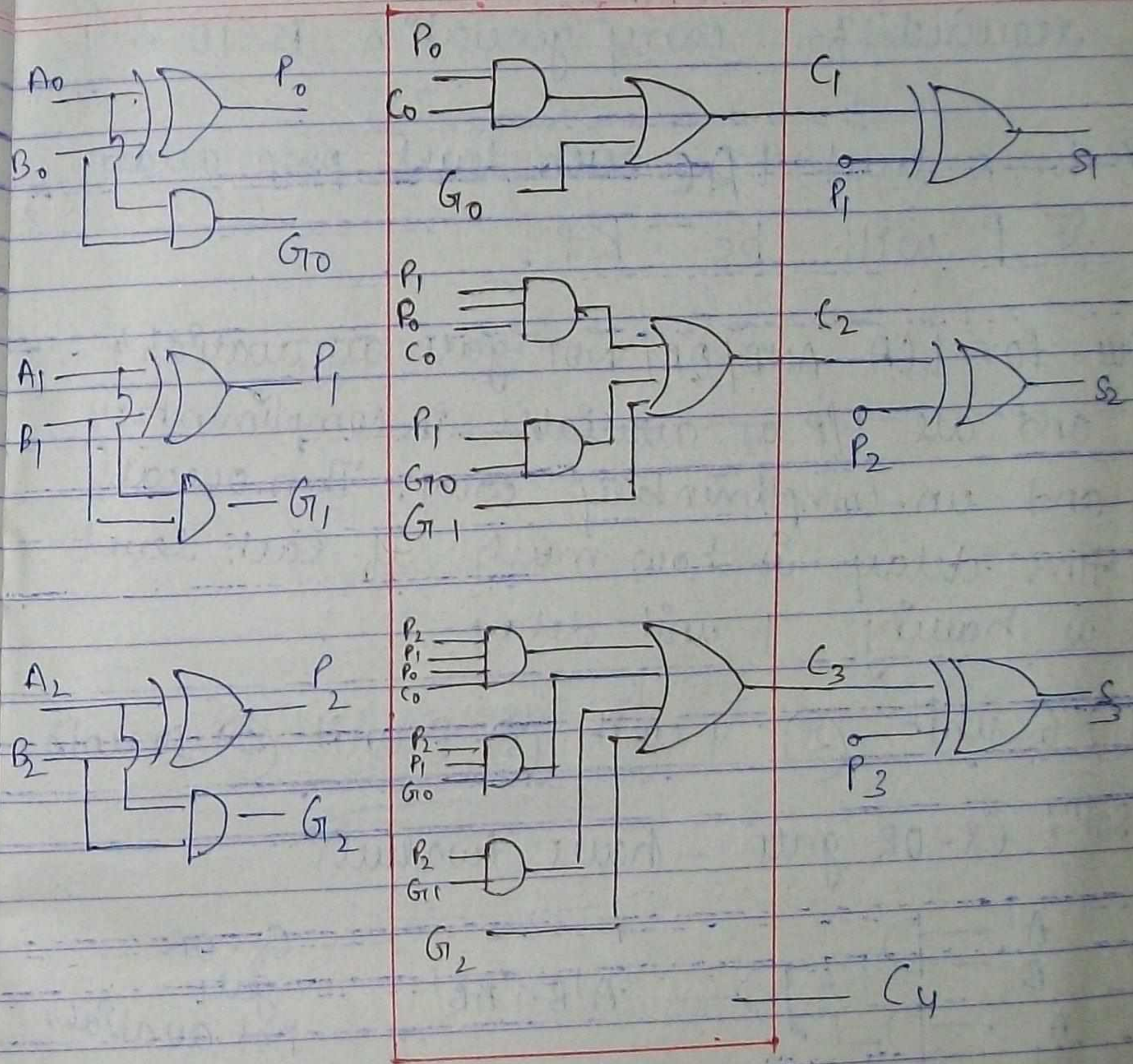
$$= C_{2+1} = C_3 = P_2 C_2 + G_2 = P_2 P_1 P_0 C_0 + P_2 P_1 G_0 + P_2 G_1 + G_2$$

$$= C_{3+1} = C_4 = P_3 C_3 + G_3$$

$$= P_3 P_2 P_1 P_0 G_0 + P_3 P_2 P_1 G_1 + P_3 P_2 G_2 + P_3 G_3 + G_4$$



No. of levels are more due to which there is a large propagation delay in 4-bit parallel adder.



⊛ No. of AND gate required for Look ahead carry generation.

Only carry generation OR gate $\rightarrow n$
AND gate $\rightarrow \frac{n(n+1)}{2}$

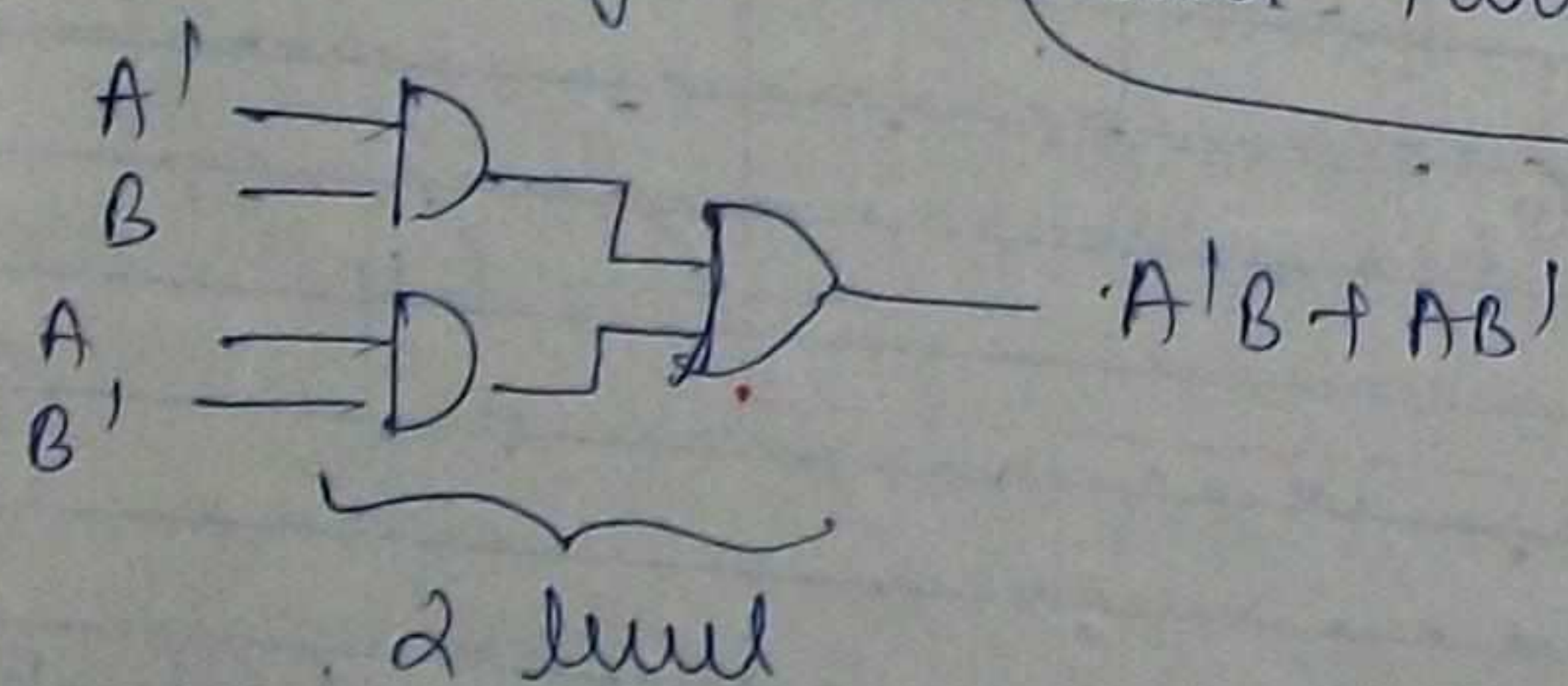
④ for 4-bit LCAO. of AND and OR gate required for carry generation is 10 & 4

⑤ for k-level LAC each level prop delay is p will be kp.

Ques. for LCA AND/OR/NOT gate are available and all I/P as available in complimentary and un-complimentary way. Then overall time delay is how much if each level is having 1 unit delay.

① 6 unit ② 4 unit ③ 8 unit ④ 10 unit

Solⁿ: Ex-OR gate - have two level



Ex-OR gate not available

∴ from previous diagram
Overall delay = 2 + 2 + 2 = 6 unit

⇒ COMPLEMENT'S METHOD -

COMPLEMENT	BIN	DEC.	OCTAL.	HEX.
x^{th}	2's	10's	8's	16's
$(x-1)^{\text{th}}$	1's	9's	7's	15's

Ex - 9th complement of $(7)_{10}$

$$\begin{array}{r} 9 \\ 7_{10} \\ \hline 2 \end{array}$$

Ex - 10th complement of $(7)_{10}$

$$\begin{array}{r} 9 \\ 7_{10} \\ \hline 2 \\ + 1 \\ \hline 3 \end{array}$$

Ex 10th complement of $(17.2)_{10}$

$$\begin{array}{r} 99.9 \\ 17.2 \\ \hline 82.7 \\ + 1 \\ \hline 82.8 \end{array}$$

Ex - 9th complement of

$$\begin{array}{r} 99.9 \\ - 75.3 \\ \hline 24.6 \end{array}$$

(75.3)₁₀

Ex - 2's complement of

① 110100

↓ 2's
001100

③ 1000

↓ 2's
1000

② 00101
↓ 2's
11011

Ex 2st complement of 110100

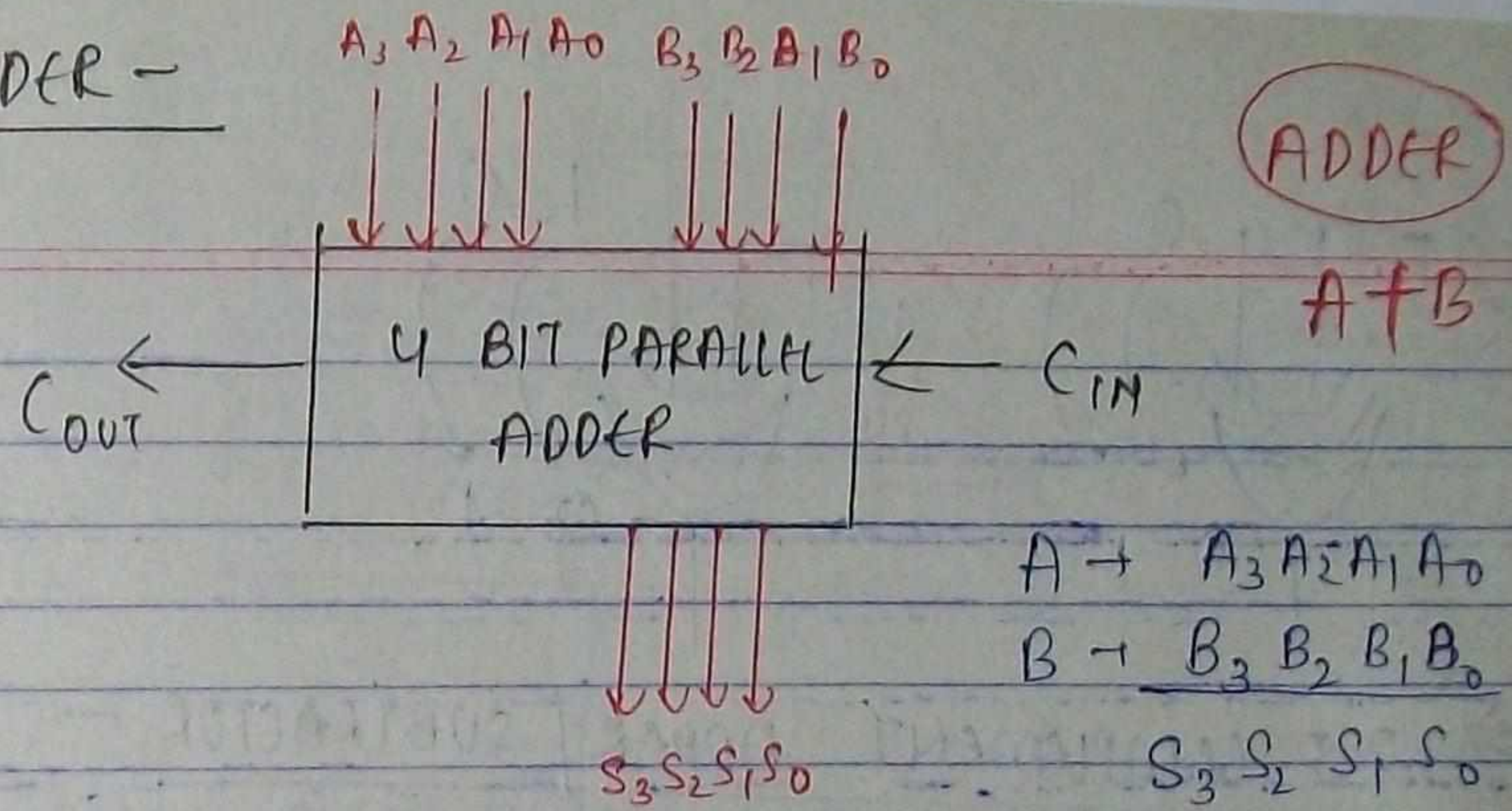
111111
- 110100

001011
+ 1

001100 2's complement

SHORT TRICK

⇒ ADDER -

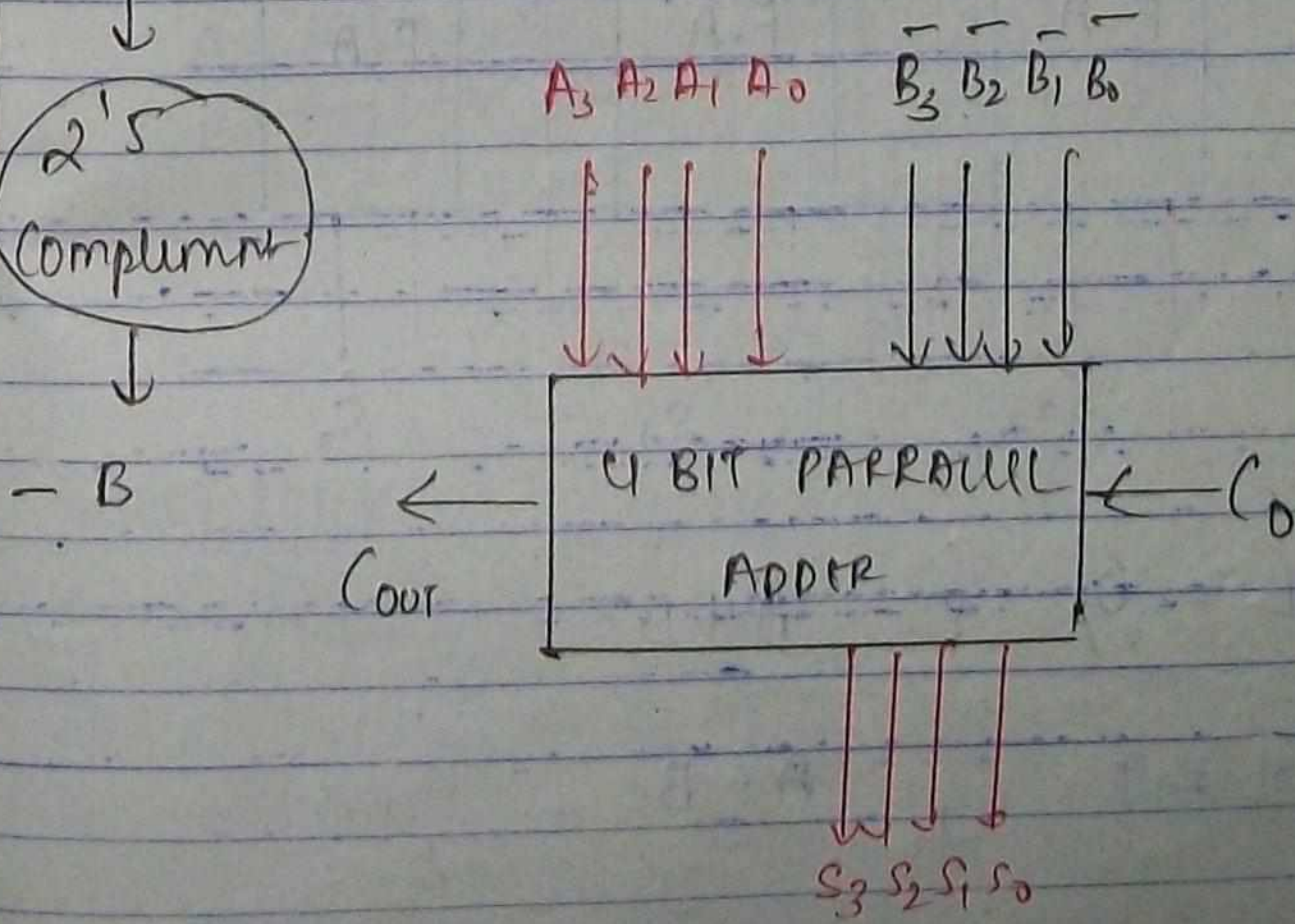


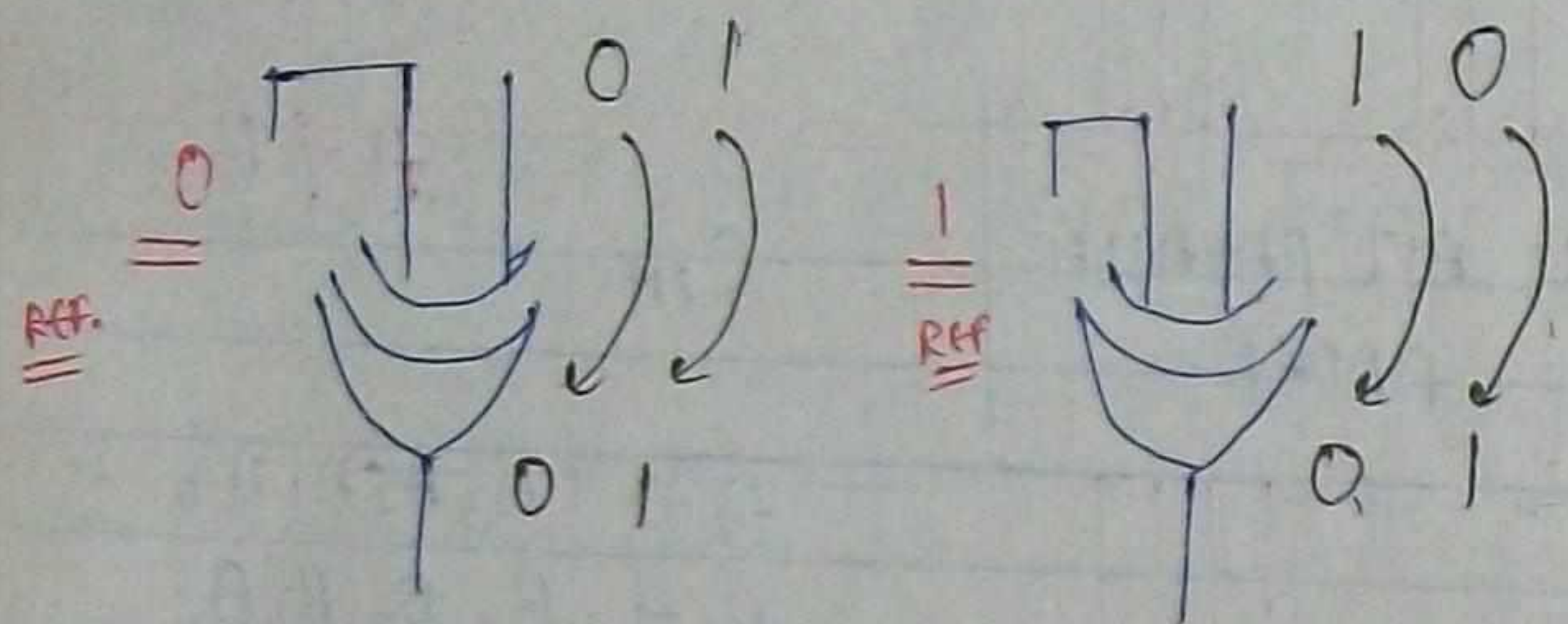
⇒ SUBTRACTOR -

$$= A - B$$

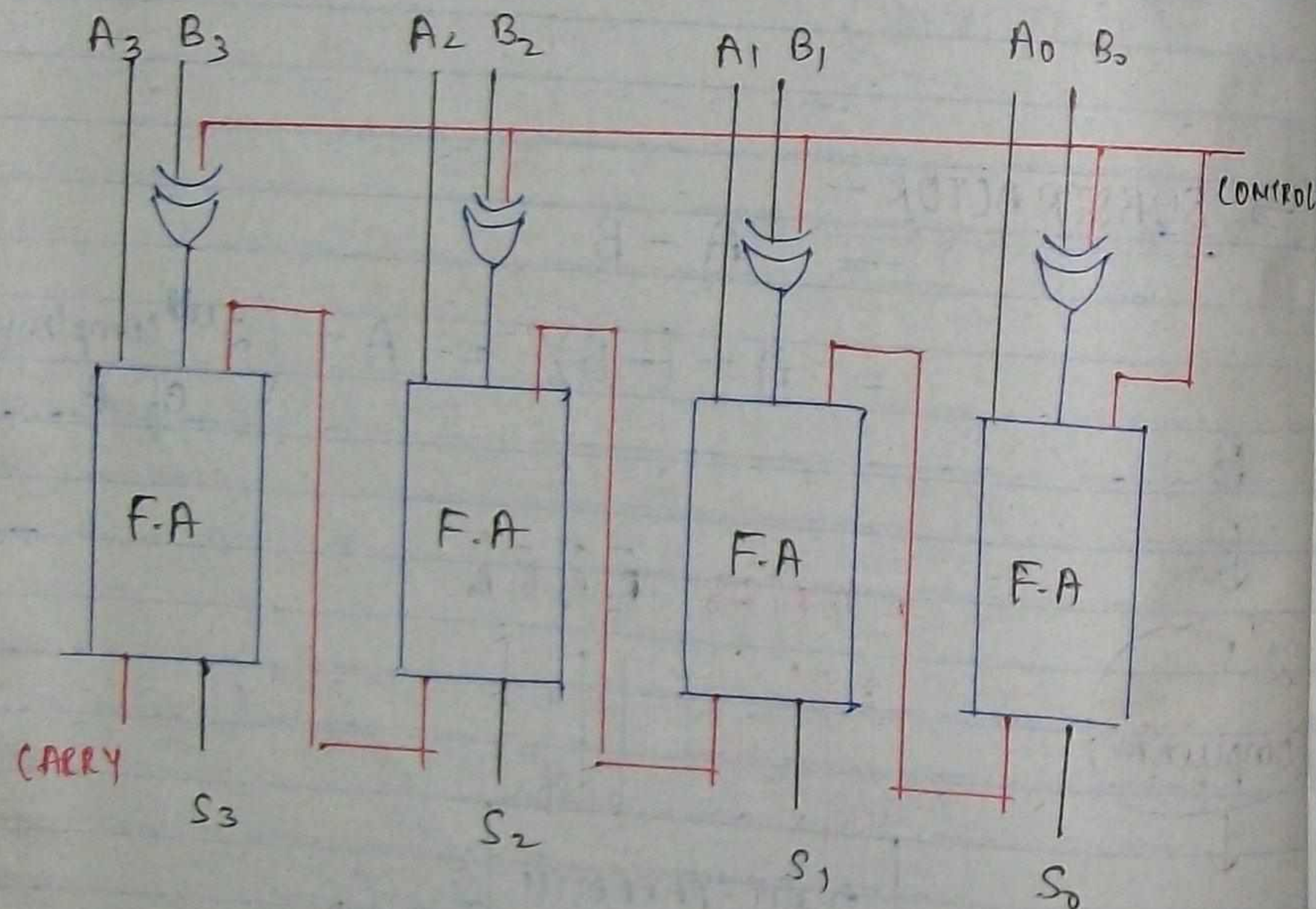
$$= A + (-B) = A + (\text{2nd complement of } B)$$

B
↓
2's Complement
↓
-B





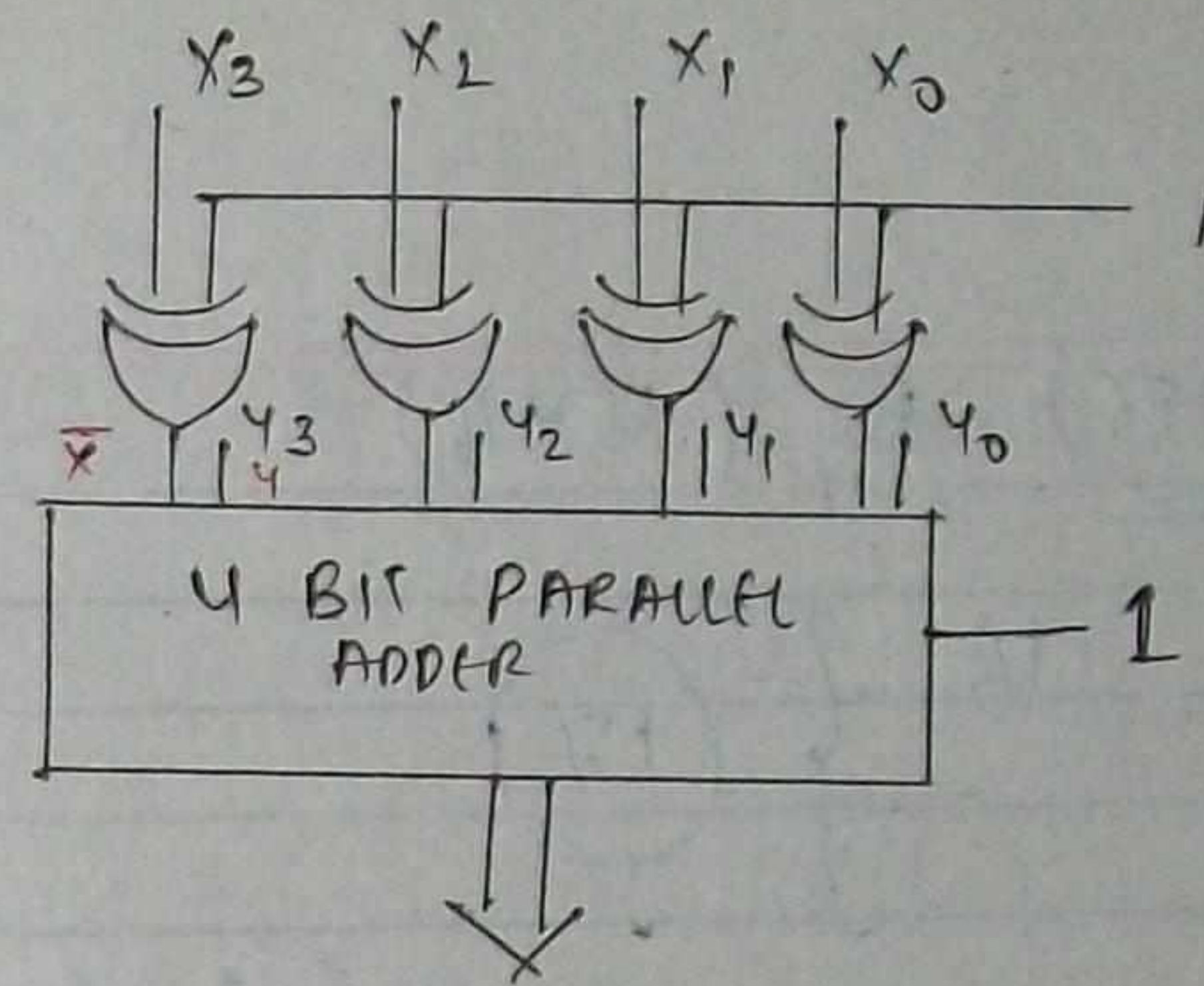
⇒ 2's COMPLEMENT ADDER / SUBTRACTOR —



CONTROL = 0 , $A + B$

CONTROL = 1 , $A - B$

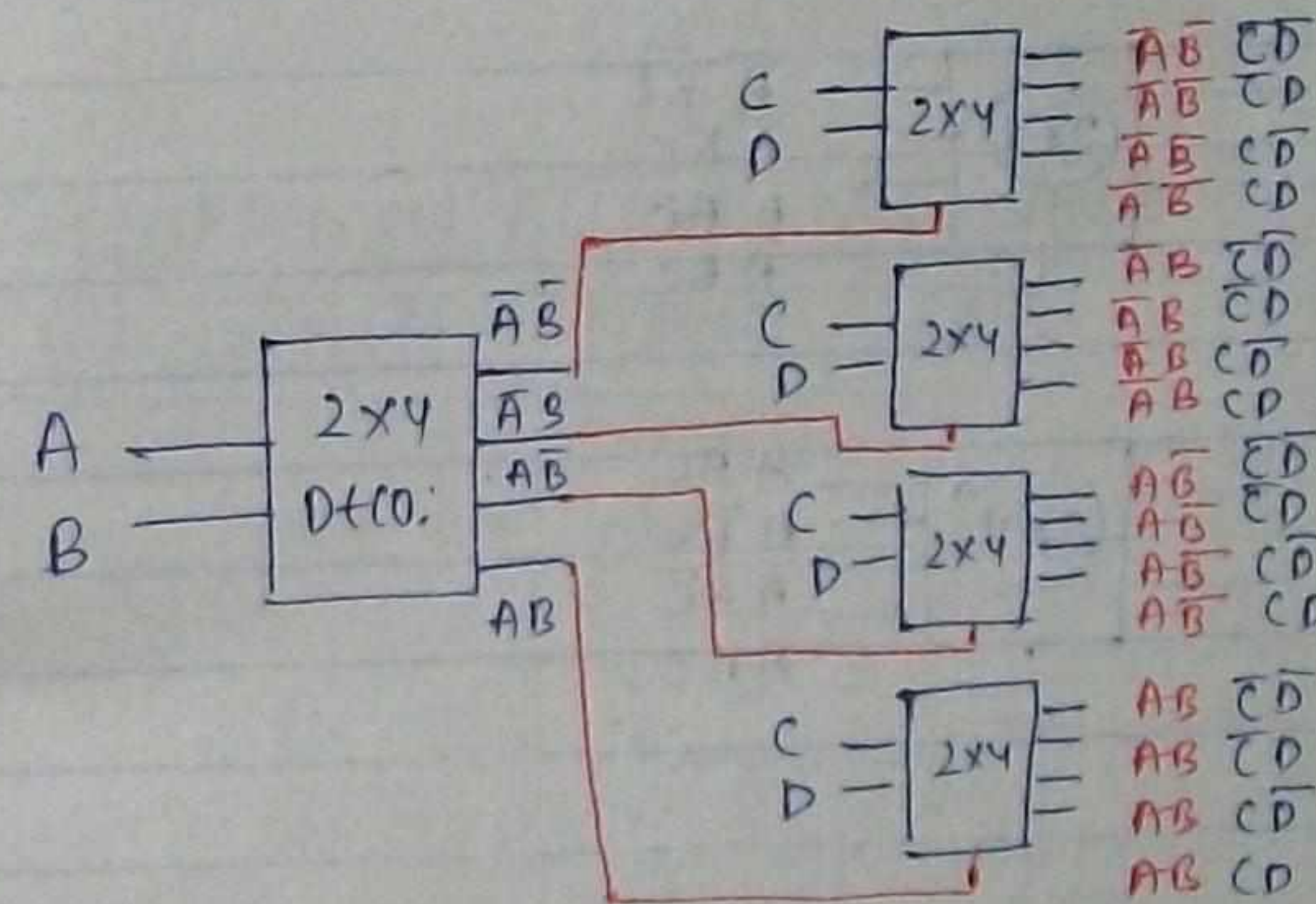
Qw.



- ① $X + Y$
- ② $Y - X$
- ③ $X - Y$
- ④ $X + 1$

⇒ IMPLEMENTATION OF HIGHER ORDER DECODER & MUX'S WITH LOWER ORDERS —

Ex - (4×16) DECODER BY (2×4) DECODER —



SHORT TRICK - $(2 \times 16) \leftarrow (2 \times 4)$

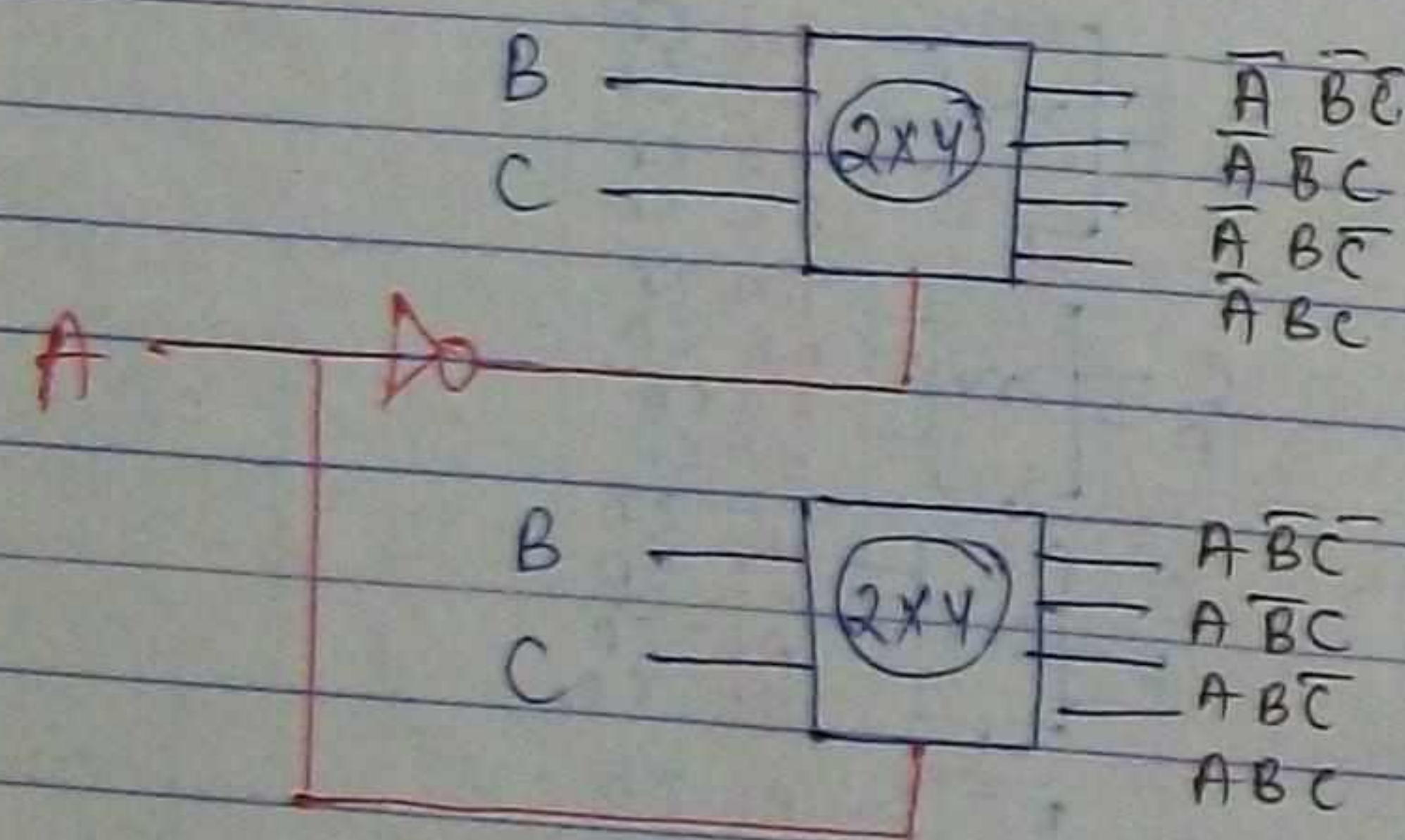
$$\left. \begin{array}{l} \frac{16}{4} = 4 \\ \frac{4}{4} = 1 \end{array} \right\} = \text{⑤ - decoder}$$

Ex (64×256) By (4×16)

$$\left. \begin{array}{l} \frac{256}{16} = 16 \\ \frac{16}{16} = 1 \end{array} \right\} (17)$$

Ques. Design the (3×8) Decoder by using (2×4) decoder.

Solⁿ: $\frac{8}{4} = (2)$



(*) $(3 \times 8) \leftarrow (2 \times 4)$

odd

hence NOT gate is used.

also

$$3 - 2 = (1)$$

one NOT gate

Ques $(5 \times 32) \leftarrow (4 \times 16)$ decoder

Solⁿ:

$$\frac{32}{16} = (2) \text{ decoder}$$

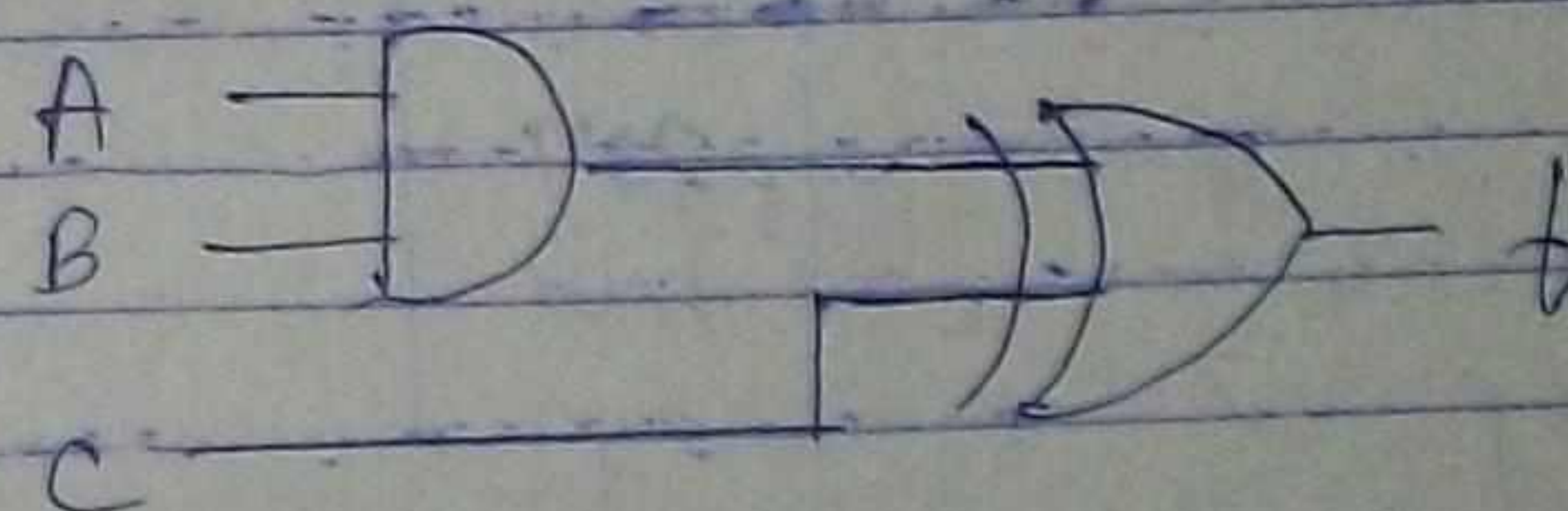
$$5 - 4 = (1) \text{ NOT gate}$$

(*) If the To be implemented gate is odd no. then NOT gate is needed.

Ex $5 - 4 = (1) \text{ NOT gate}$

$\Rightarrow$ IMPLEMENTATION OF AND / OR EXPRESSION BY USING NAND GATE -

Ex - $f = AB + C$



PROCEDURE -

① AND $\rightarrow$ NAND

② OR $\rightarrow$ Bubbled OR (NAND)

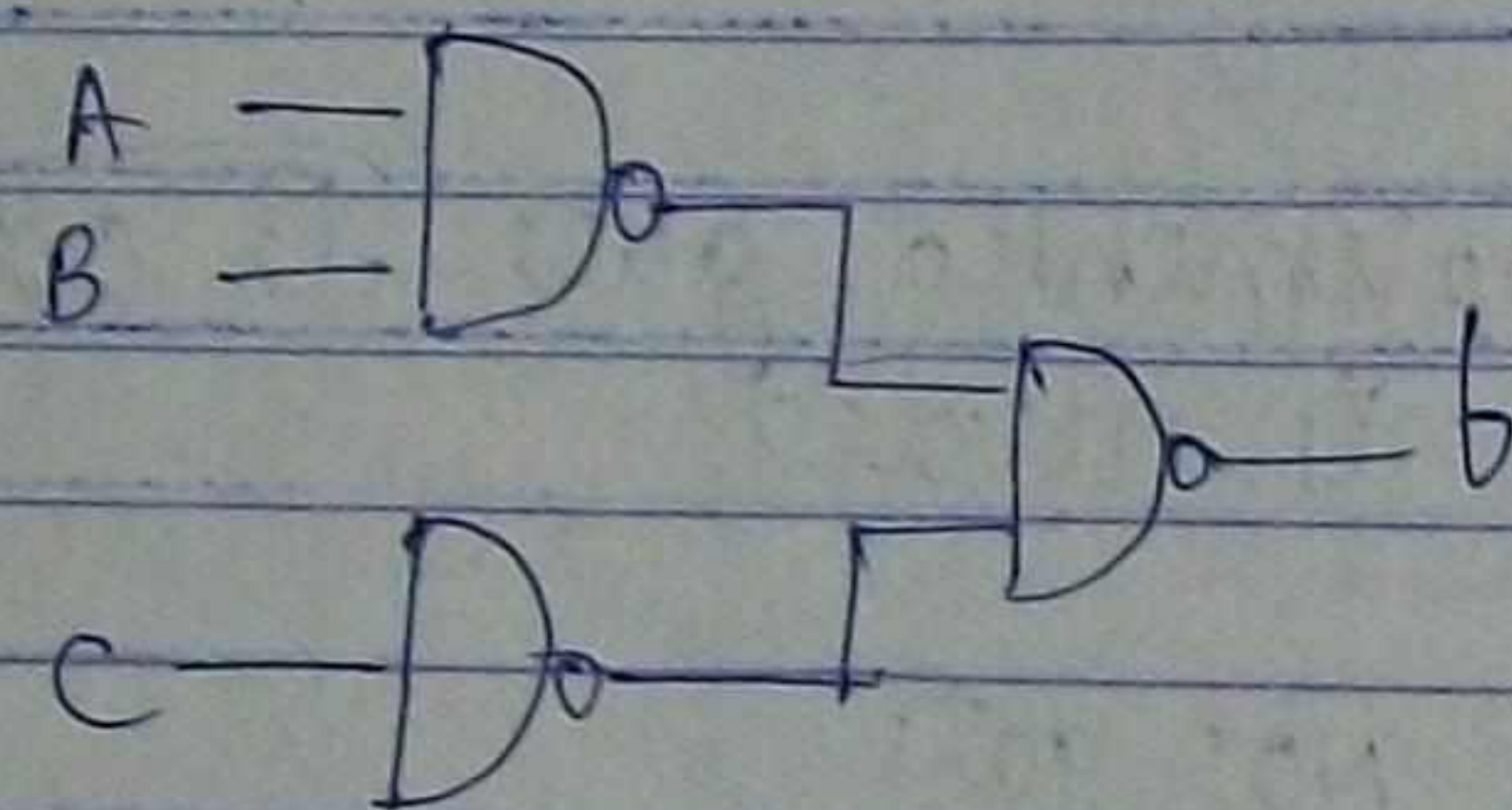
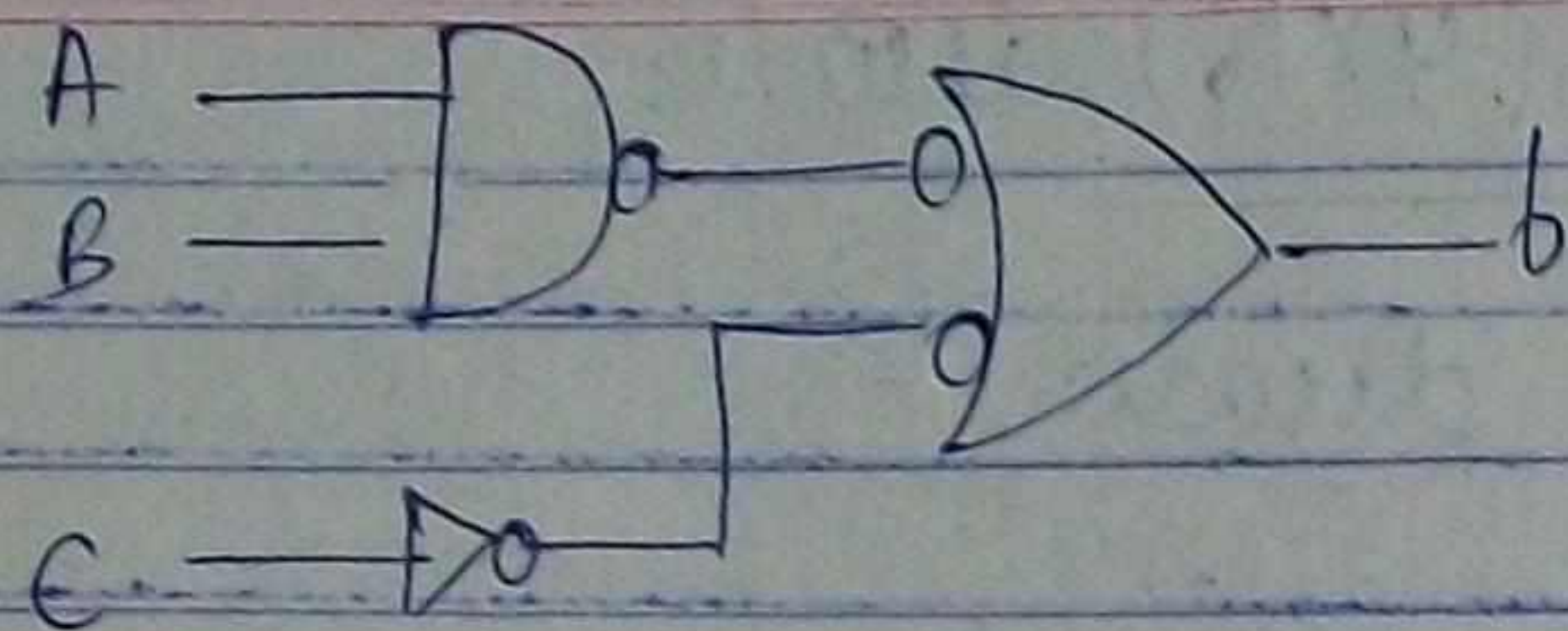
③ NOT $\rightarrow$ must be balanced using NAND

(NOT)

OR $\rightarrow$ BUBBLED NOT

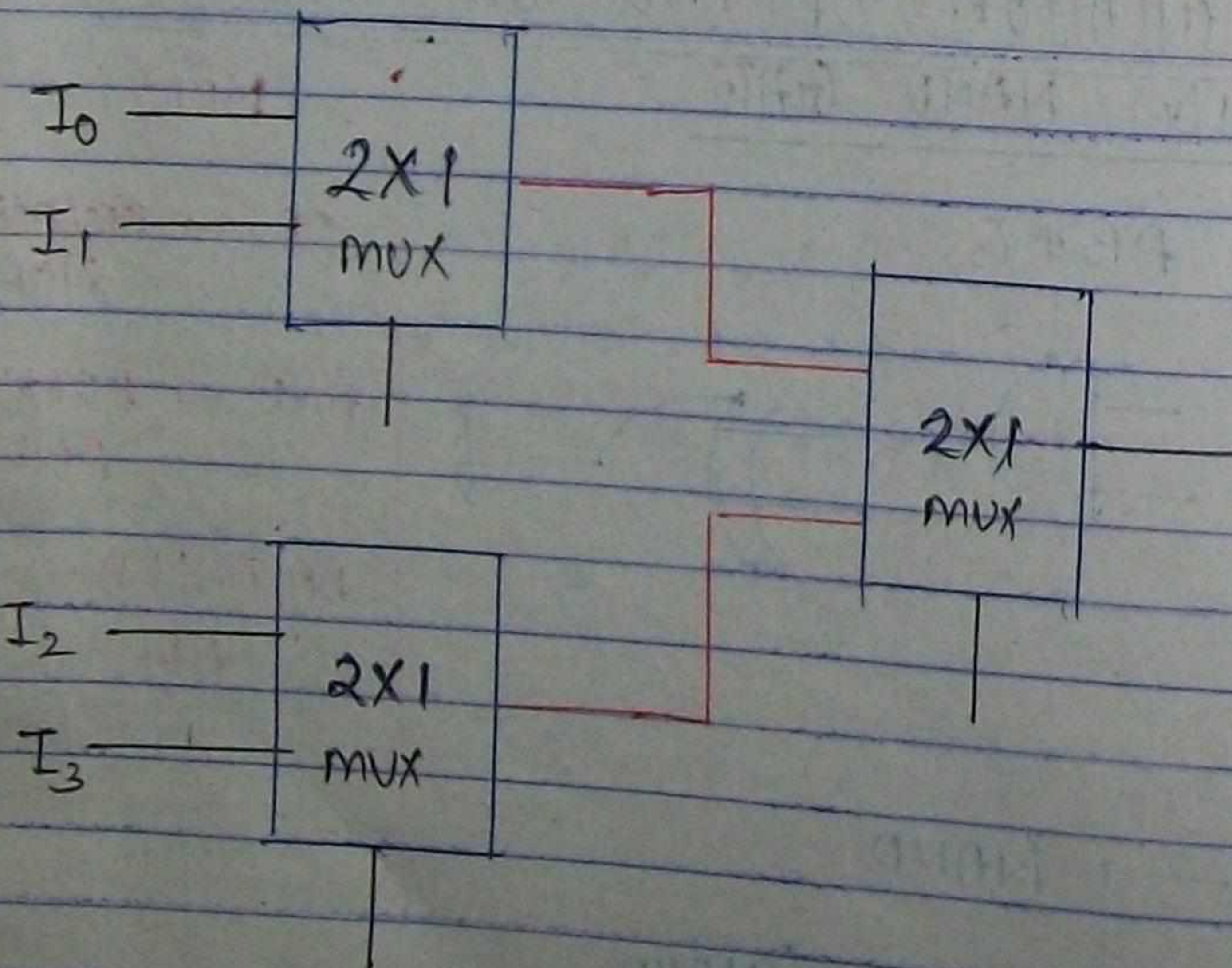
AND $\rightarrow$ BUBBLED AND

BALANCED NOT



3 NAND
GATE

Que. Design (4x1) mux by (2x1) mux.
Solⁿs



LOGIC FAMILIES

⇒ CLASSIFICATION OF DIGITAL IC'S -

① SSI - SMALL SCALE INTEGRATION
(less than 12 logic gates)

No. of component → upto 99

② MSI - (MEDIUM SCALE INTEGRATION)

logic gate - 12 to 99

No. of component - 100 to 999

③ LSI - (LARGE SCALE INTEGRATION)

logic gate - 1000 to 999

No. of component - 1000 to 9999

④ VLSI - (VERY LARGE SCALE INTEGRATION)

logic gate - about 1000

No. of component - about 10,000.

⇒ CHARACTERISTIC PARAMETERS OF LOGIC FAMILIES -

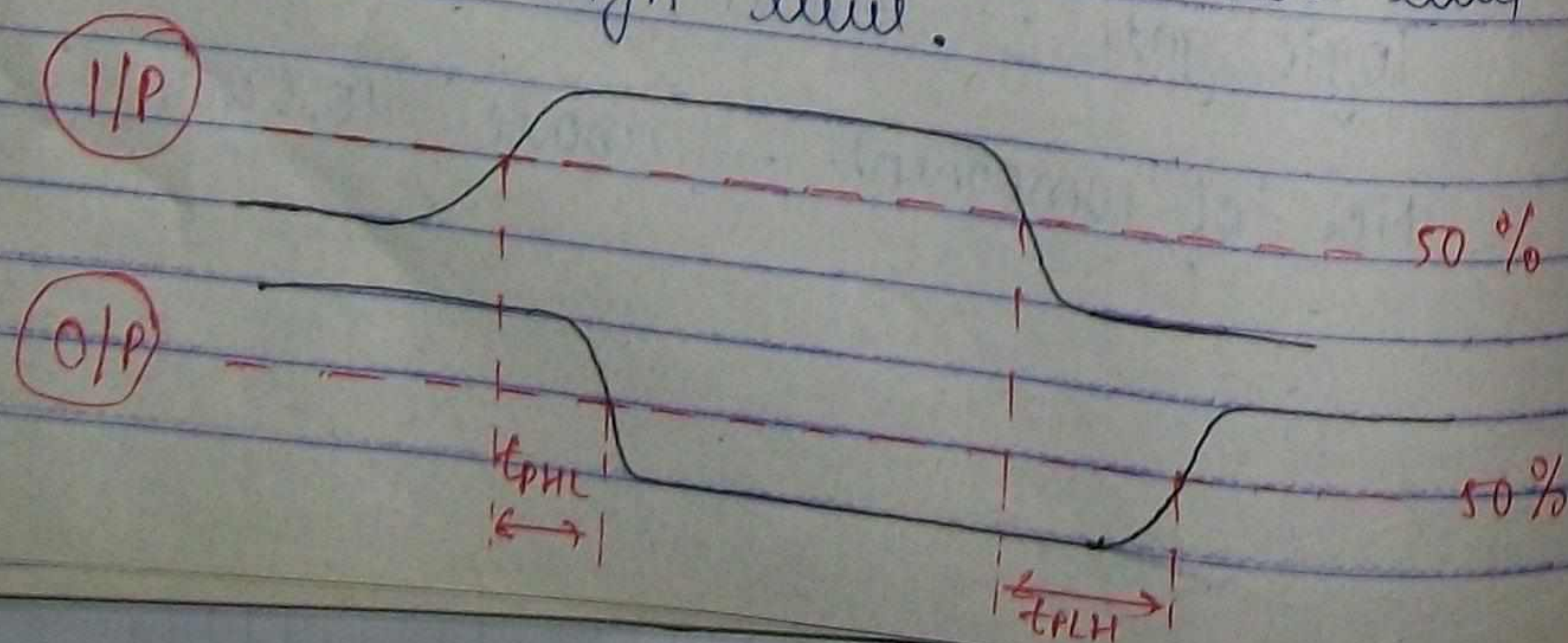
① PROPAGATION DELAY (t_{pd}) -

- It is the time delay b/w the application of I/P and occurrence of O/P.

$$t_{pd} = \frac{t_{PHL} + t_{PLH}}{2}$$

t_{PHL} - Time b/w a specified reference point on the I/P pulse and a corresponding reference point on the resulting O/P pulse when the O/P changing from the high level to low level.

t_{PLH} - Time B/w a specified reference point on the I/P pulse and a corresponding reference point on the resulting pulse when the O/P changing from the low level to the high level.



② unit of Prop. delay → nanosecond

$t_{PHL} > t_{PLH}$ IN FET

DUE TO LARGE CAPACITANCE FORMED

$t_{PHL} < t_{PLH}$ IN BJT

DUE TO REVERSE RECOVERY TIME

② POWER DISSIPATION (P.d) -

- It is the amount of power dissipated in the IC.

$$P.d = V_{cc} \cdot I_{cc}$$

$$I_{cc} = \frac{I_{ccL} + I_{ccH}}{2}$$

I_{ccL} - current value when all I/Ps are low.

I_{ccH} - current value when all I/Ps are high.

unit of P.d → milliwatts.

③ FIGURE OF MERIT (FOM, SPP) -

- It is the product of propagation delay and power dissipation.

- also known as speed power product.

unit → picoJoules

Ques. A gate have $P.d = 5 \text{ ns}$, $I_{ccH} = 1 \text{ mA}$
 $I_{ccL} = 2.5 \text{ mA}$ with a DC voltage = 5 V .
 then find SPP.

Soln:

$$\begin{aligned} \text{SPP} &= P.d \times V_{cc} \cdot I_{cc} \\ &= 5 \times 10^{-9} \times 5 \times \left(\frac{1+2.5}{2} \right) \\ &= 43.75 \text{ pJ} \end{aligned}$$

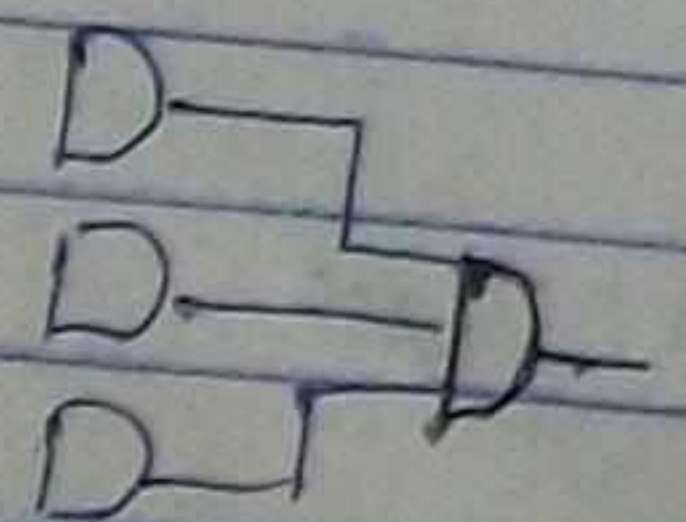
* for the best operation of IC's figure of merit should be much smaller as possible.

④ FAN-IN —

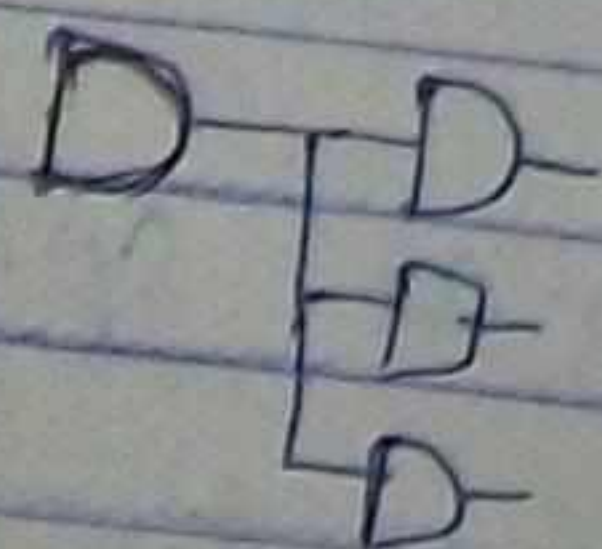
— The no. of I/P that a logic gate can handle.

⑤ FAN-OUT —

— It is the max. no. of similar logic gate that can be driven by a logic gate O/P without affecting the logic gate performance.

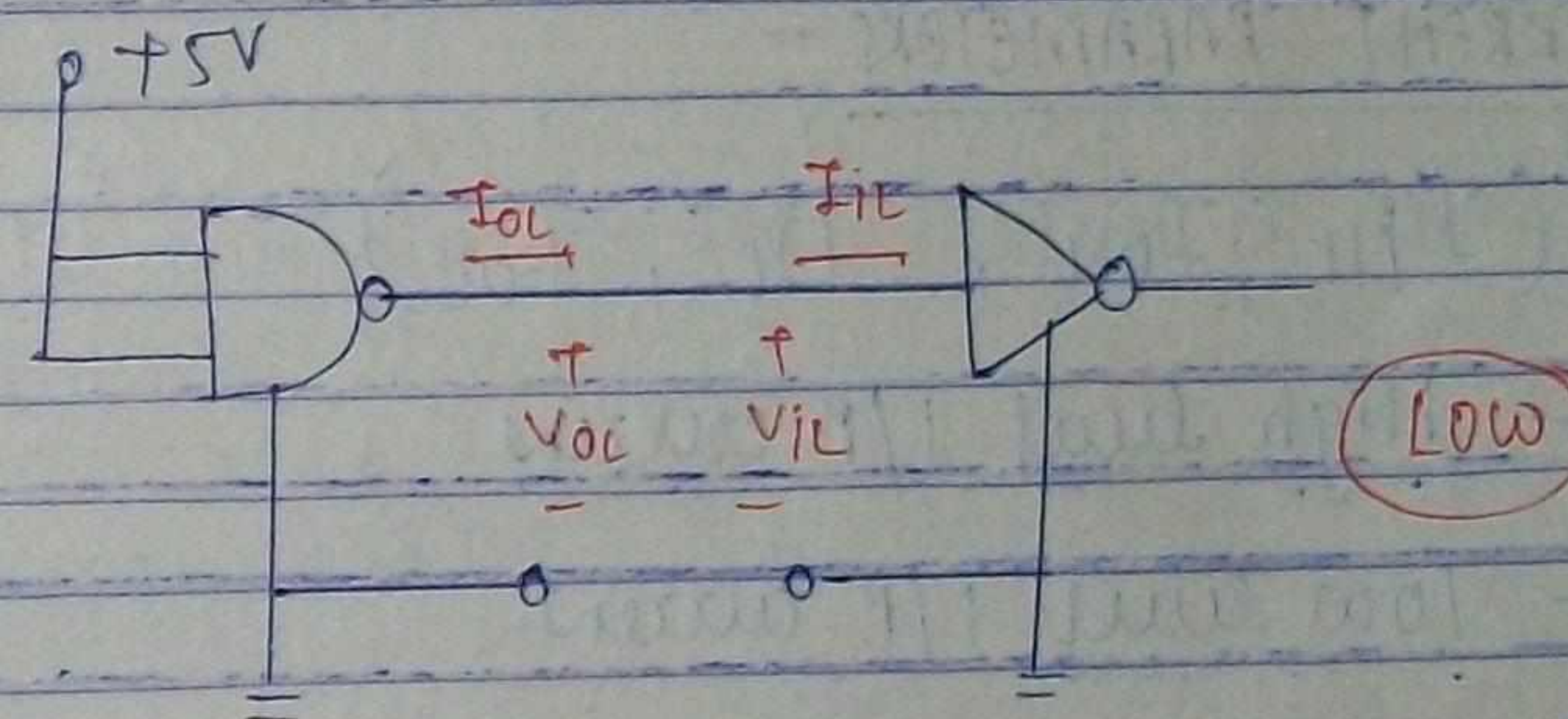
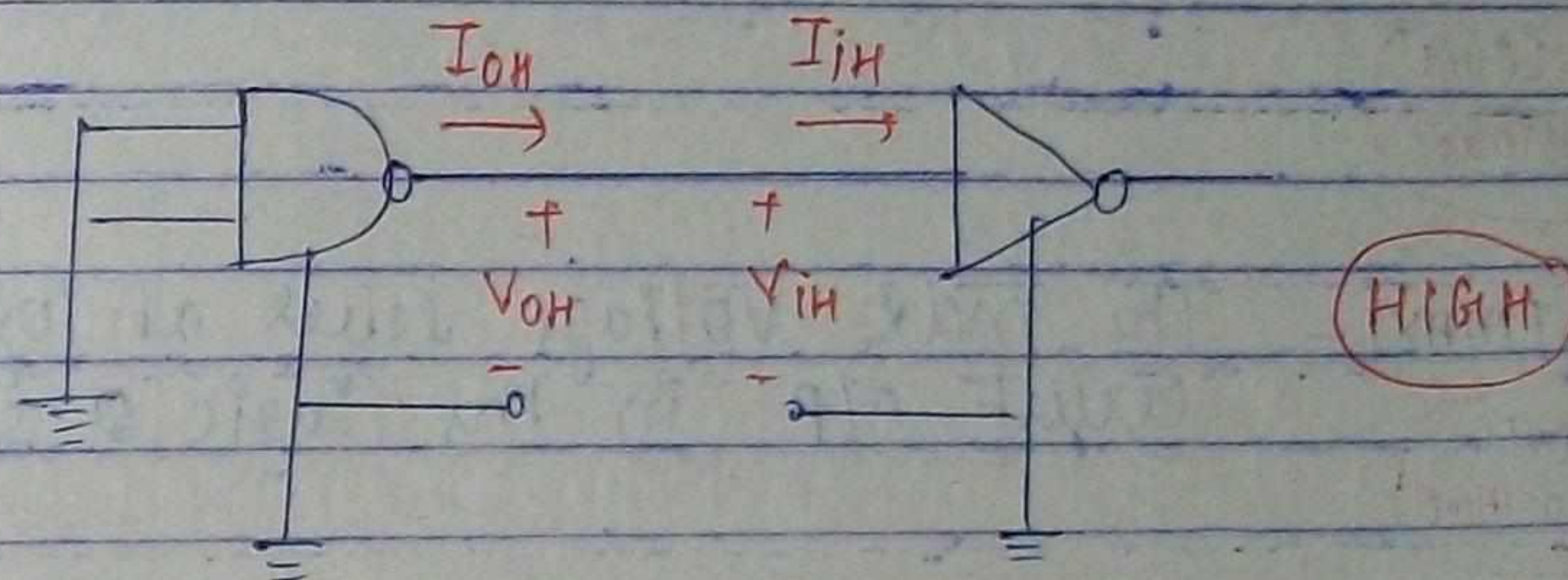


FAN-IN



FAN-OUT

⑥ VOLTAGE PARAMETERS —



$V_{IH}(\text{min})$ — The min. voltage level required for a logic '1' at an I/P. Any voltage below this level will not be accepted as a HIGH by the logic circuit.

HIGH LEVEL I/P VOLTAGE

$V_{IL}(\text{max})$ — The max. voltage level required for a logic '0' at I/P. Any voltage above this level will not be accepted as low by the logic circuit.

LOW LEVEL I/P VOLTAGE

$V_{OH(min)}$ - The min. voltage level at the o/p in the logic 1 state.

HIGH LEVEL
O/P VOLTAGE

$V_{OL(max)}$ - The max. voltage level at logic output o/p in the logic 0 state.

LOW LEVEL
O/P VOLTAGE

⑦ CURRENT PARAMETERS -

$(I_{IH}, I_{OH}, I_{IL}, I_{OL})$

I_{IH} - High level i/p current

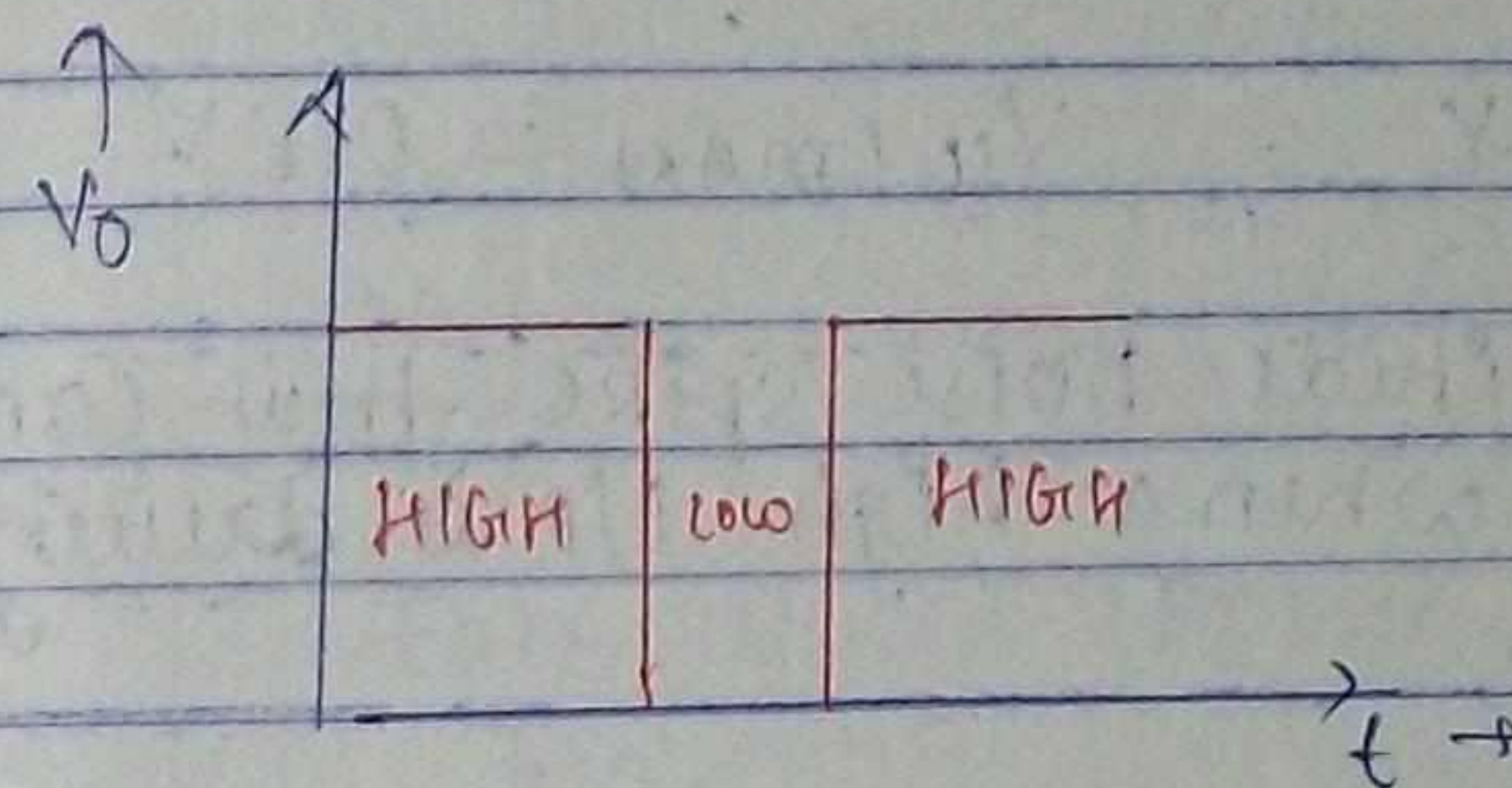
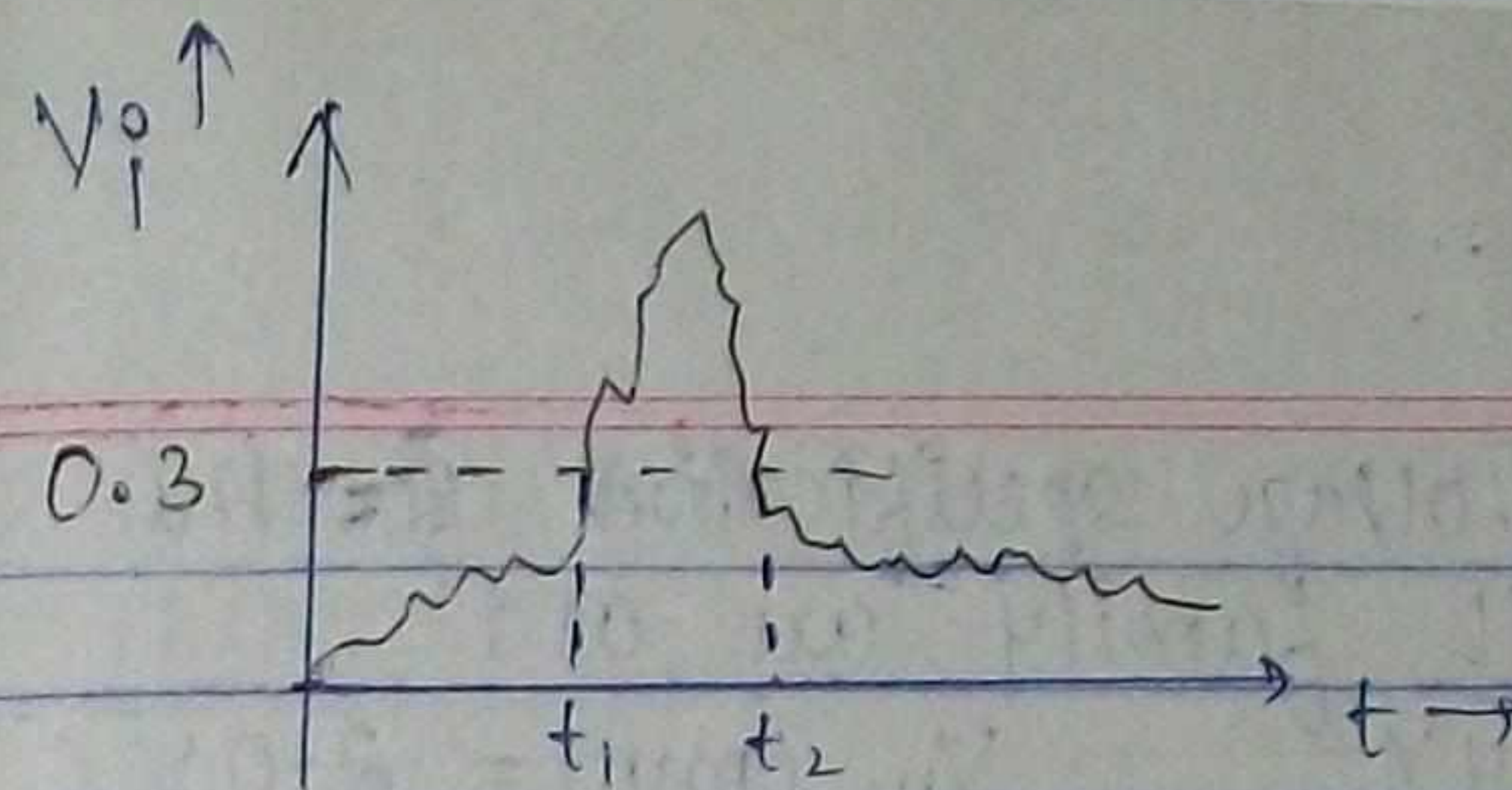
I_{IL} - Low level i/p current

I_{OH} - High level o/p current

I_{OL} - Low level o/p current

⑧ NOISE IMMUNITY

- It is the circuit ability to tolerate noise without causing any unwanted change in the o/p voltage.



- Noise margin is the measure of noise immunity.

$$V_{OH} > V_{IH} > V_{IL} > V_{OL}$$

HIGH STATE NM $(V_{NH}) = V_{OH(min)} - V_{IH(min)}$

LOW STATE NM $(V_{NL}) = V_{IL(max)} - V_{OL(max)}$

OVERALL NM $NM = [V_{NH} \cdot V_{NL}]_{min}$

Ques. The I/P-O/P voltage specification for the standard TTL family are as:

$$V_{OH(\min)} = 2.4V \quad V_{IH(\min)} = 2.0V$$

$$V_{OL(\max)} = 0.4V \quad V_{IL(\max)} = 0.8V$$

(1) The max. amplitude noise spike that can be tolerated when a high I/P is driving an I/P is —.

(A) 0.4V (B) 0.8V (C) 0.2V (D) 0.6V

(2) The max. amplⁿ noise spike that can be tolerated when a low O/P is driving an I/P is —.

(A) 0.4V (B) 0.8V (C) 0.2V (D) 0.6V

Solⁿ: $V_{NH} = 2.4 - 2.0 = 0.4$

$$V_{NL} = 0.8 - 0.4 = 0.4$$

★ $\Rightarrow$ FINDING THE FAN-OUT WITH THE HELP OF CURRENT PARAMETER —

FAN OUT (NO. OF UNIT) LOADS	=	$\frac{I_{OH}}{I_{IH}}$	=	$\frac{I_{OL}}{I_{IL}}$
-----------------------------------	---	-------------------------	---	-------------------------

— The fan-out is chosen the smaller of these two.

Ques. The current rating are —

$$I_{OL(\max)} = 8mA \quad I_{OH(\max)} = -0.4mA$$

$$I_{IL(\max)} = -0.1mA \quad I_{IH(\max)} = 20\mu A$$

then find the fan-out?

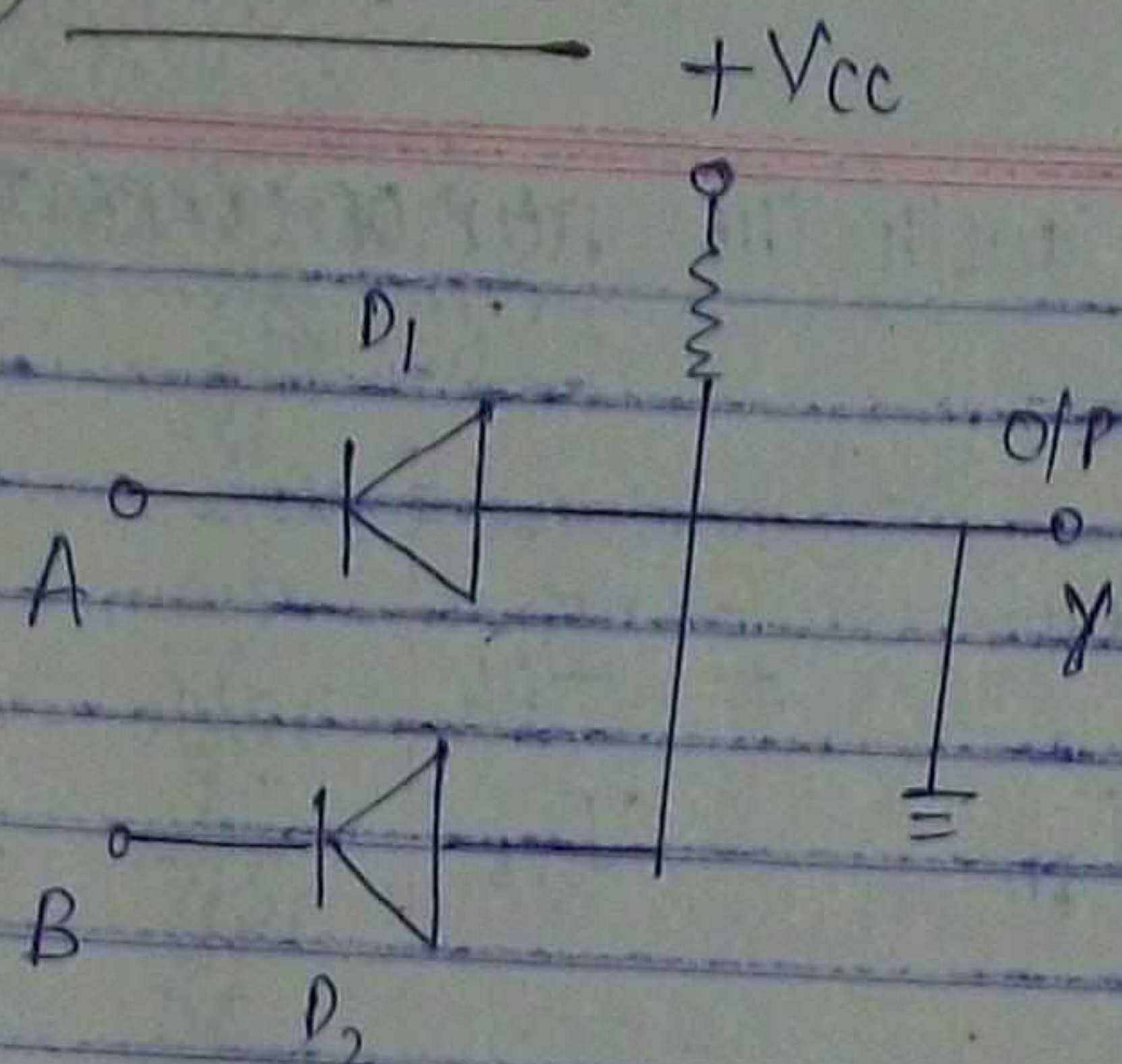
Solⁿ: fan out = $\frac{-0.4}{+20 \times 10^{-6}} = \frac{8}{-0.1}$

↓

= 20

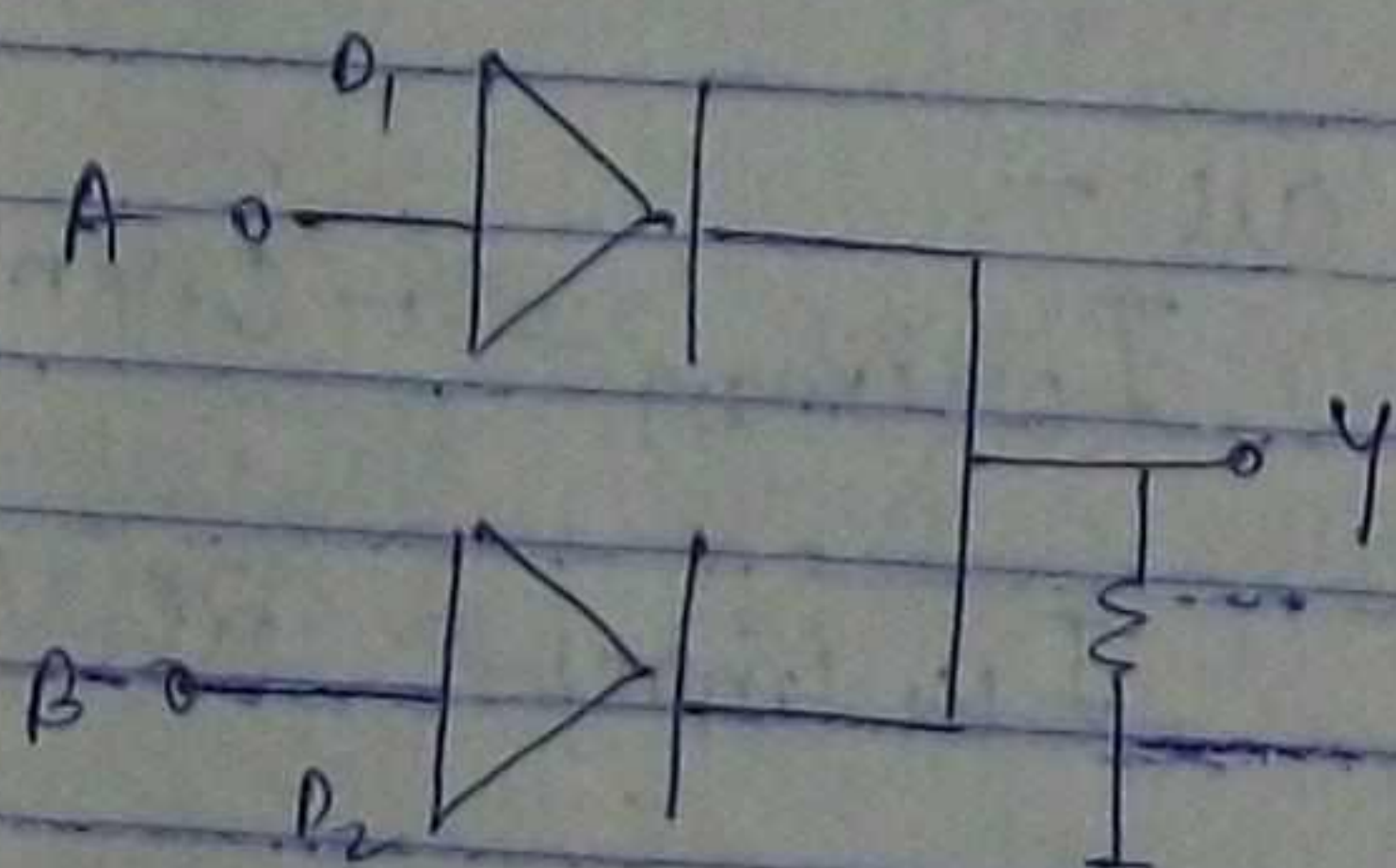
$\Rightarrow$ LOGIC GATES BY USING SIMPLE WITH DIODES —

① AND GATE -



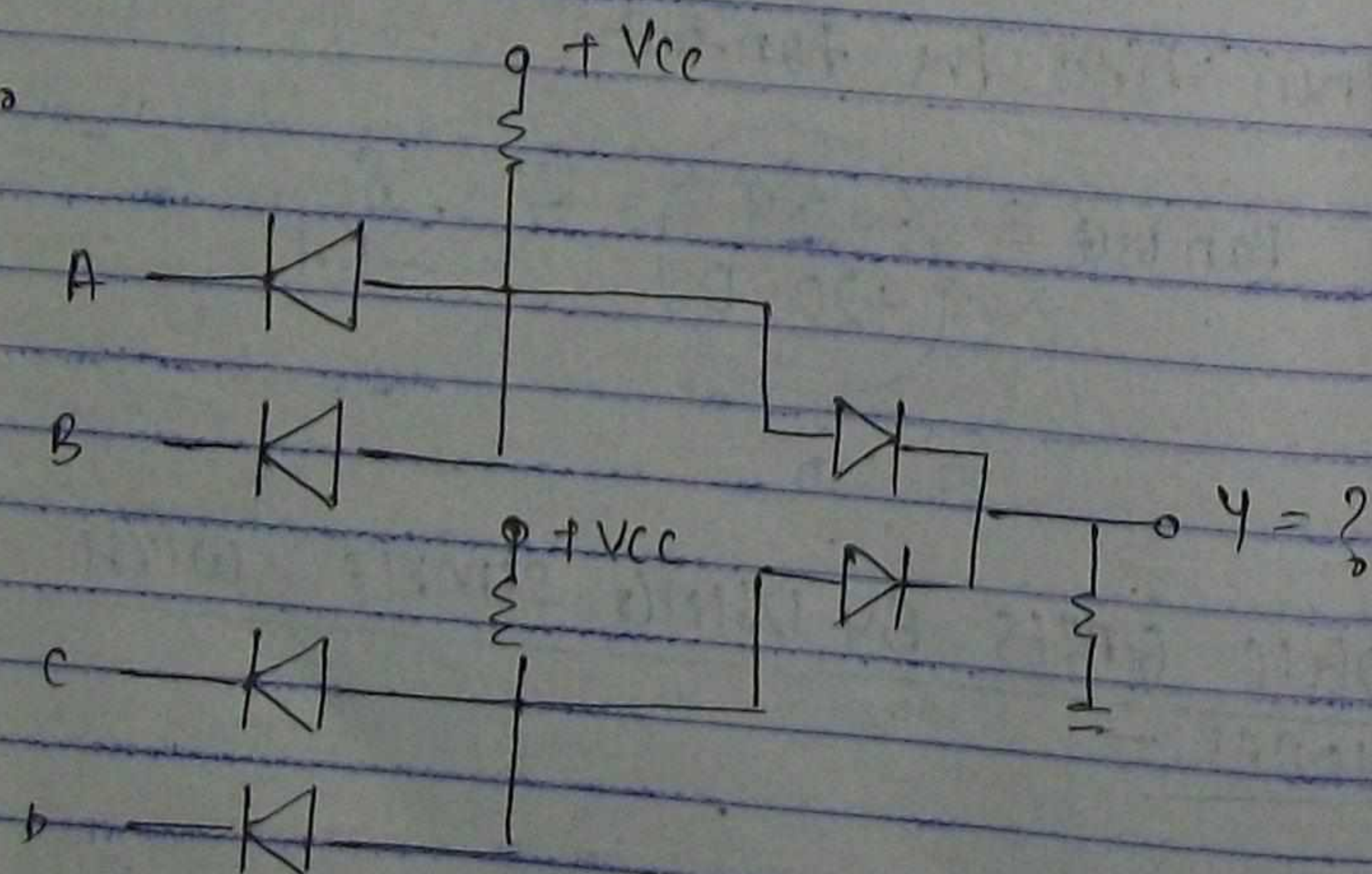
A	B	D ₁	D ₂	Y
0	0	ON	ON	0
0	1	ON	OFF	0
1	0	OFF	ON	0
1	1	OFF	OFF	1

② OR GATE -



A	B	D ₁	D ₂	Y
0	0	OFF	OFF	0
0	1	OFF	ON	1
1	0	ON	OFF	1
1	1	ON	ON	1

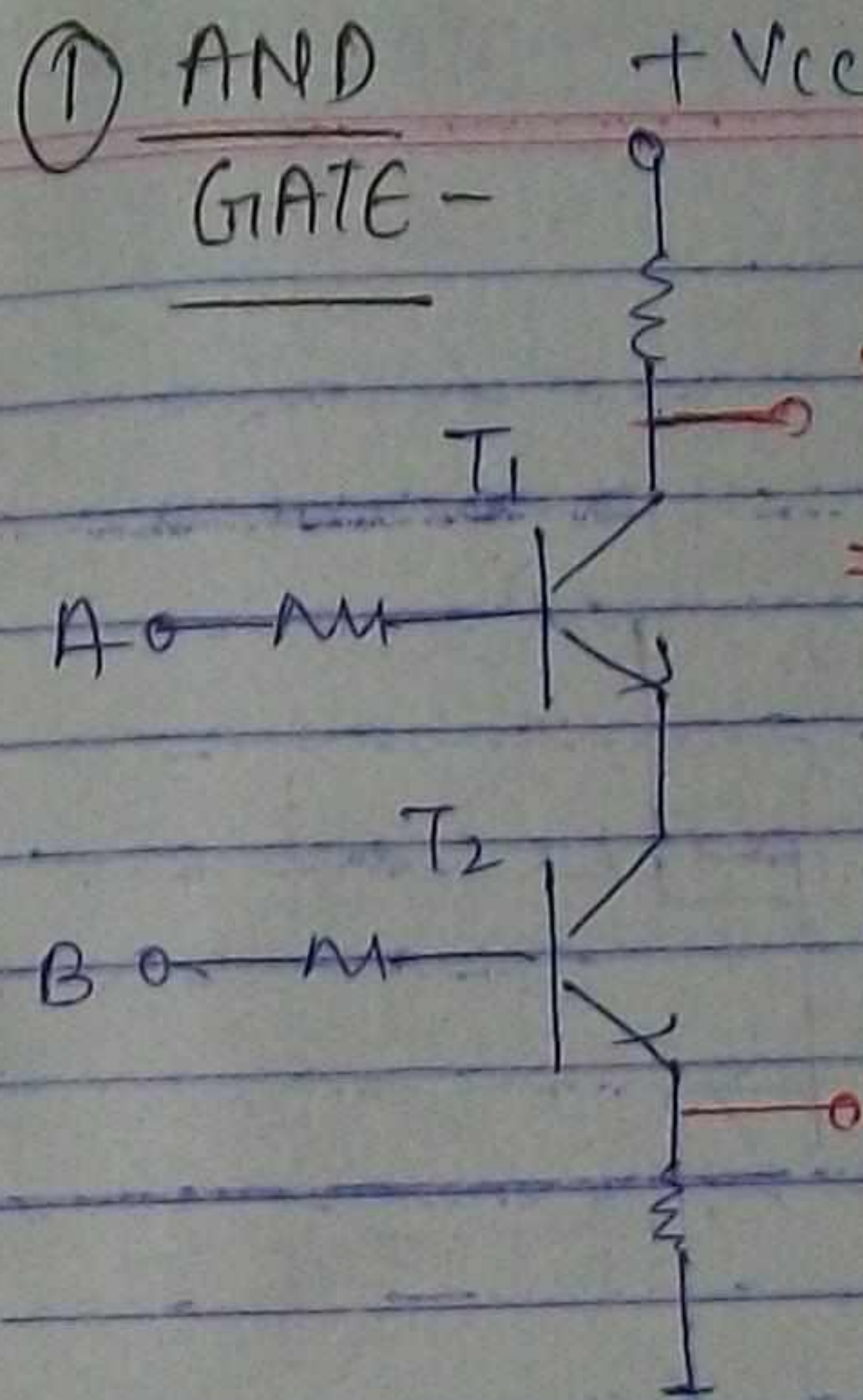
Ques,



ANS $Y = AB + CD$

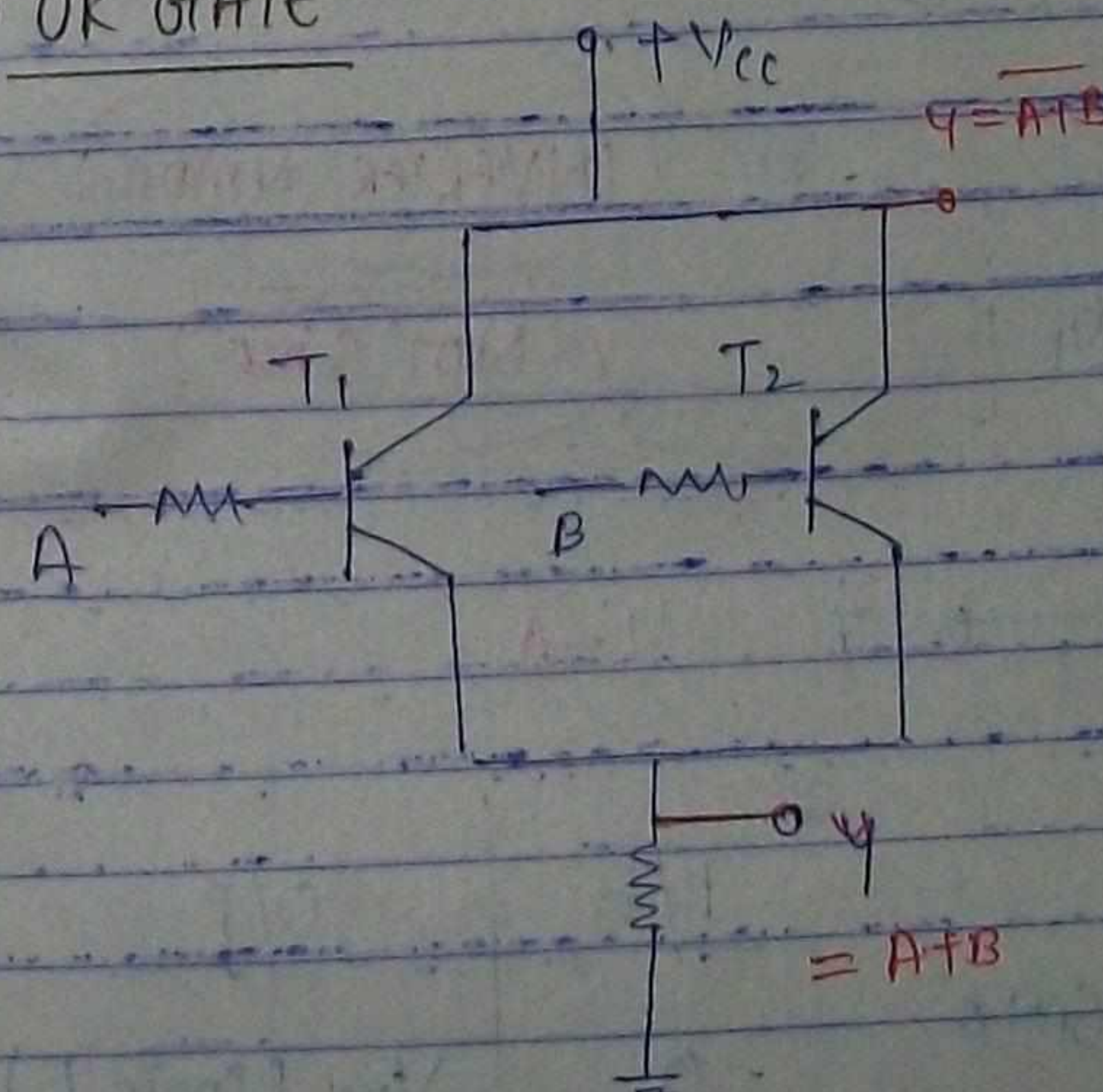
⇒ BY TRANSISTOR -

① AND GATE -



A	B	T ₁	T ₂	Y ₁	Y ₂
0	0	OFF	OFF	0	1
0	1	OFF	ON	0	1
1	0	ON	OFF	0	1
1	1	ON	ON	1	0

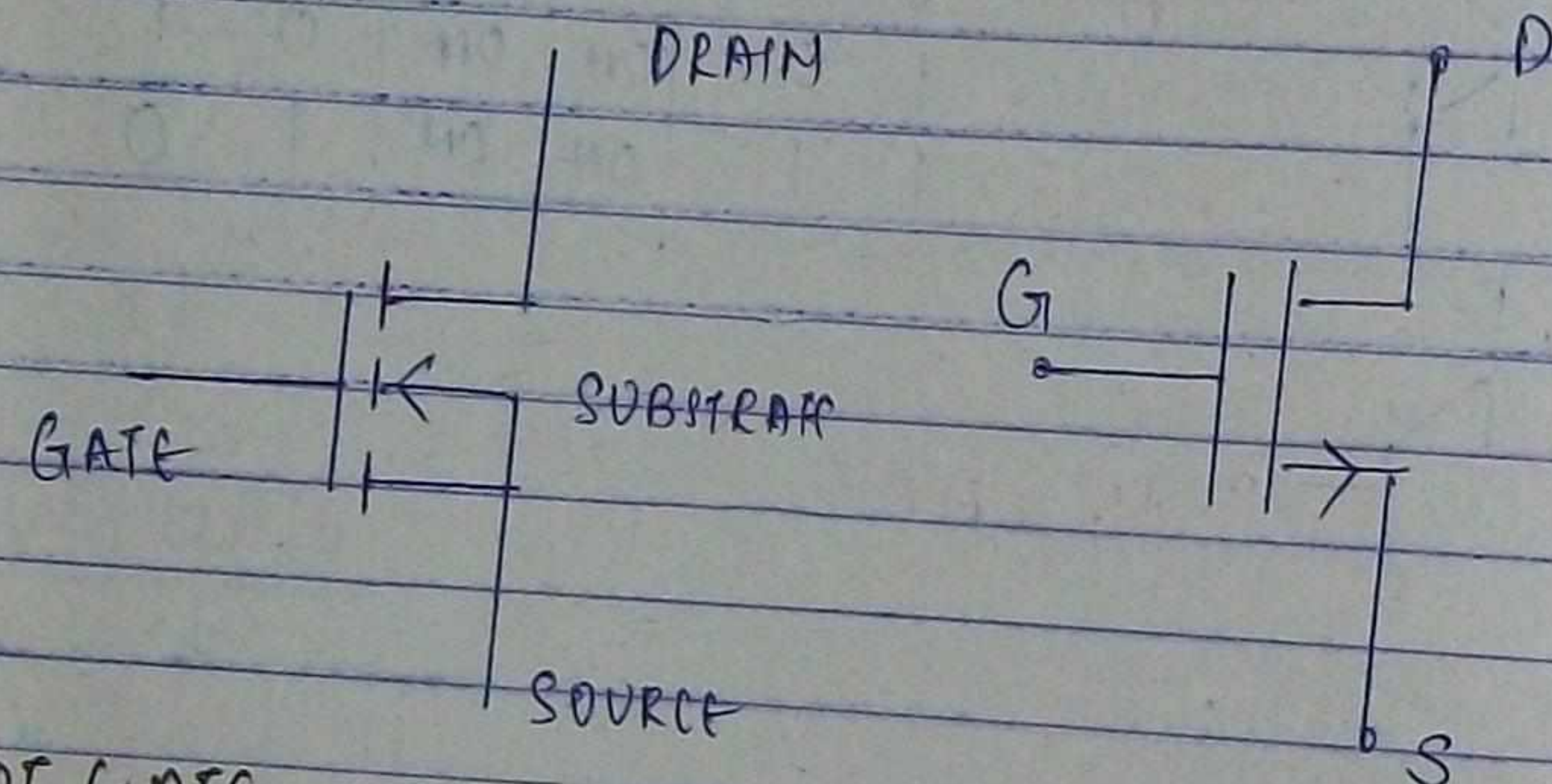
② OR GATE -



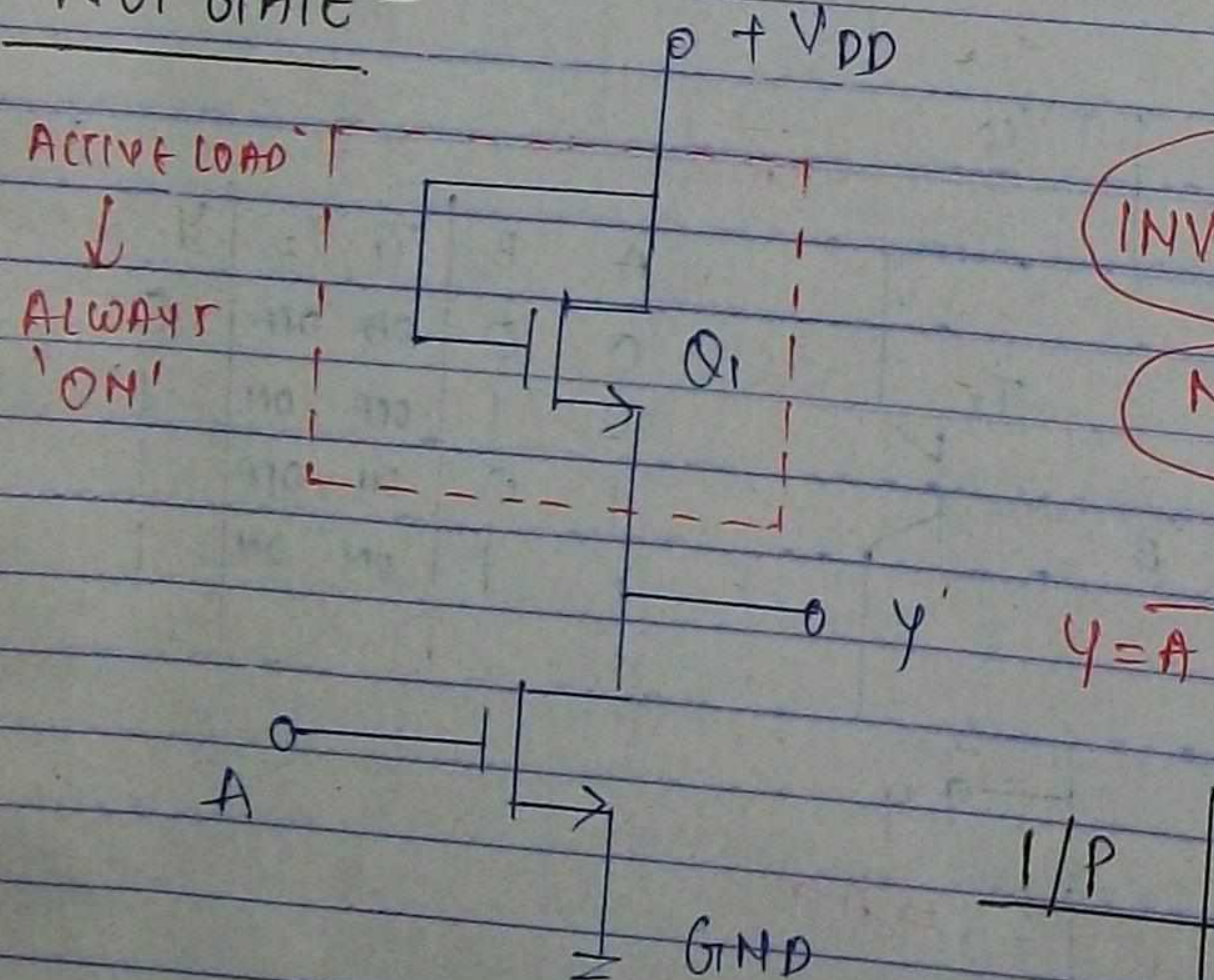
A	B	T ₁	T ₂	Y
0	0	OFF	OFF	0
0	1	OFF	ON	1
1	0	ON	OFF	1
1	1	ON	ON	1

⇒ MOS LOGIC FAMILIES -

① N-CHANNEL MOS -



NOT GATE -

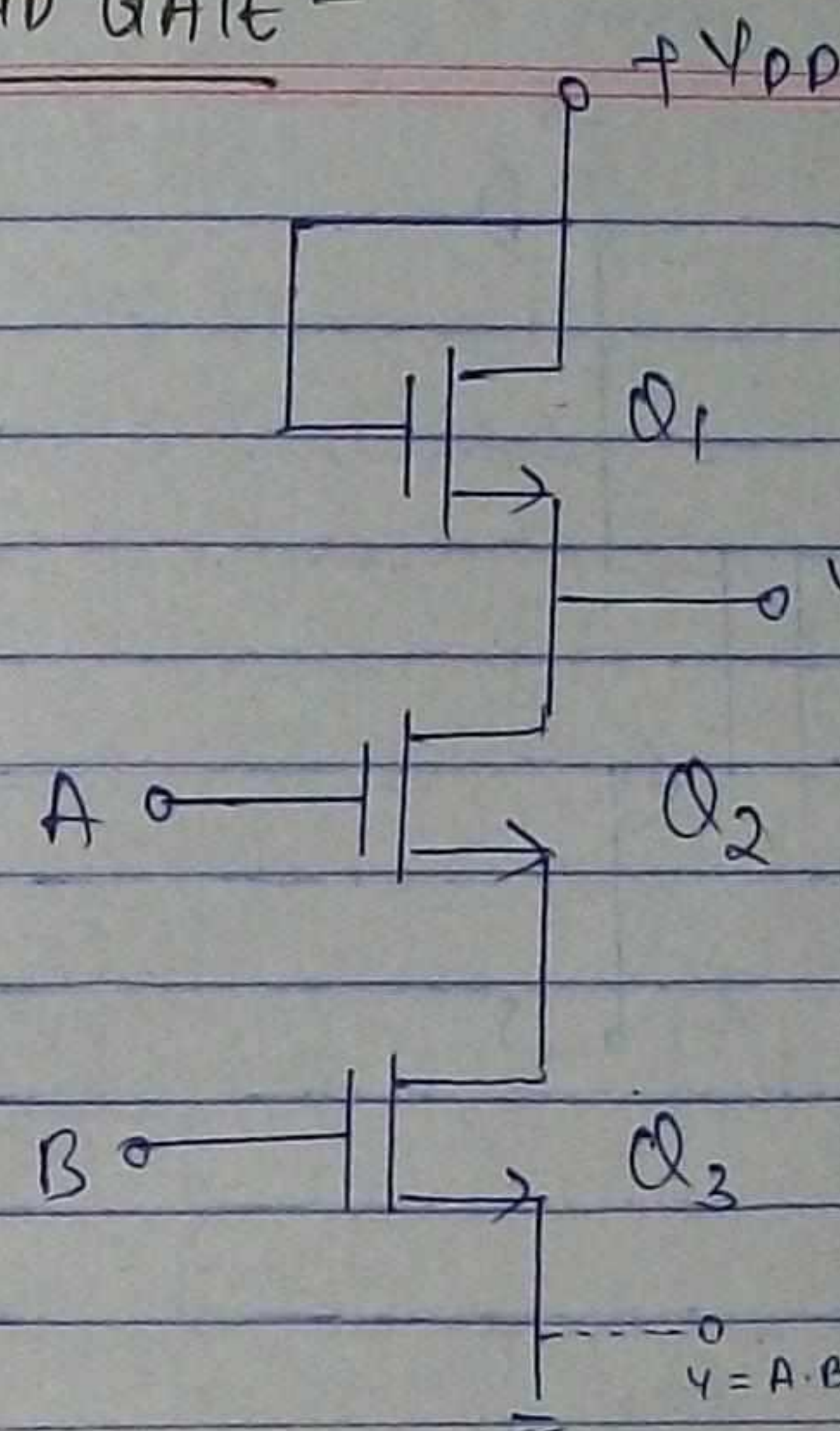


INVERTER NMOS

NOT GATE

I/P	O/P
0	V_{DD} (logic 1)
V_{DD}	GND (logic 0)

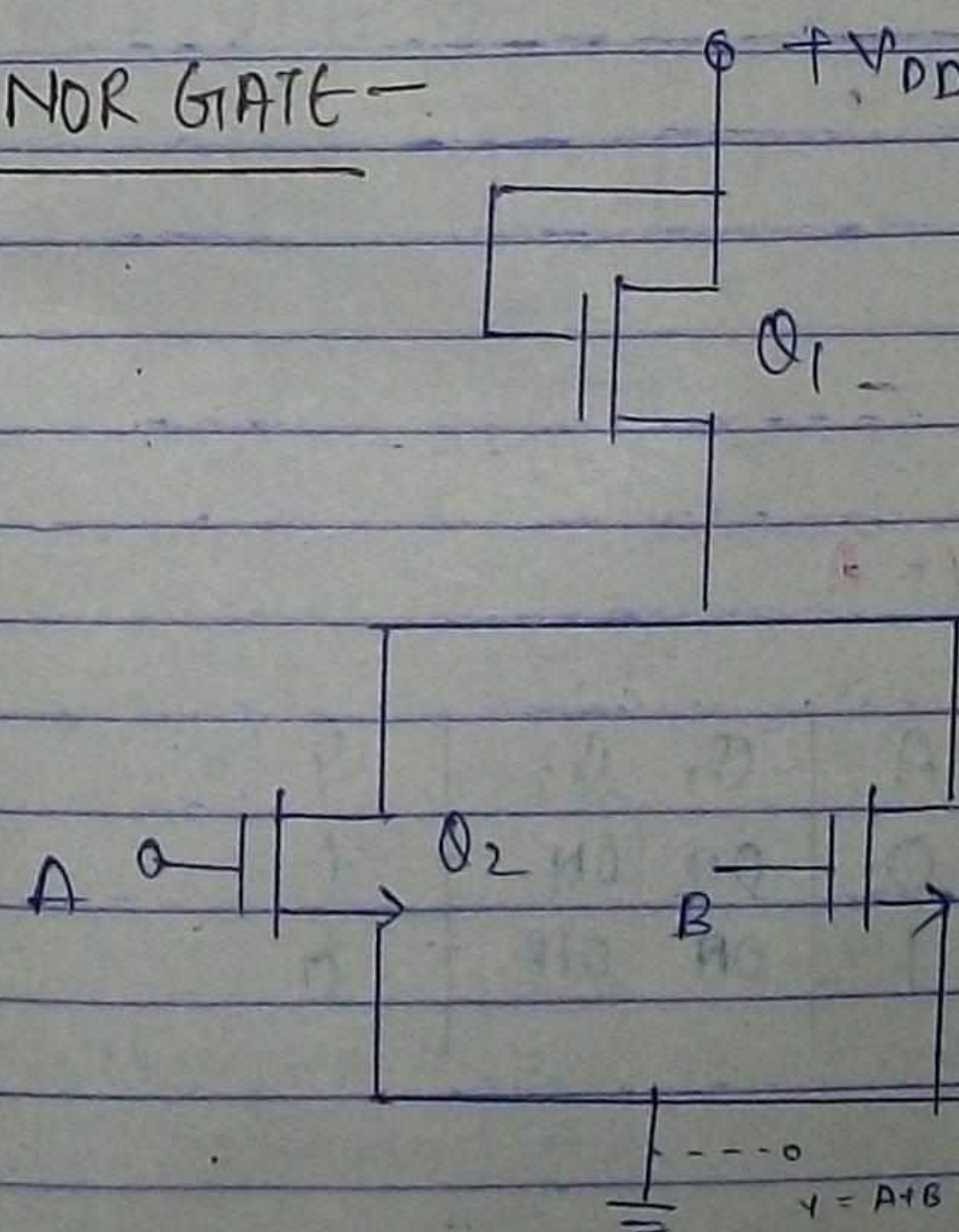
NAND GATE -



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

A	B	Q ₁	Q ₂	Q ₃	Y	Logic
0	0	ON	OFF	OFF	V_{DD}	1
0	1	ON	OFF	ON	V_{DD}	1
1	0	ON	ON	OFF	V_{DD}	1
1	1	ON	ON	ON	0	0

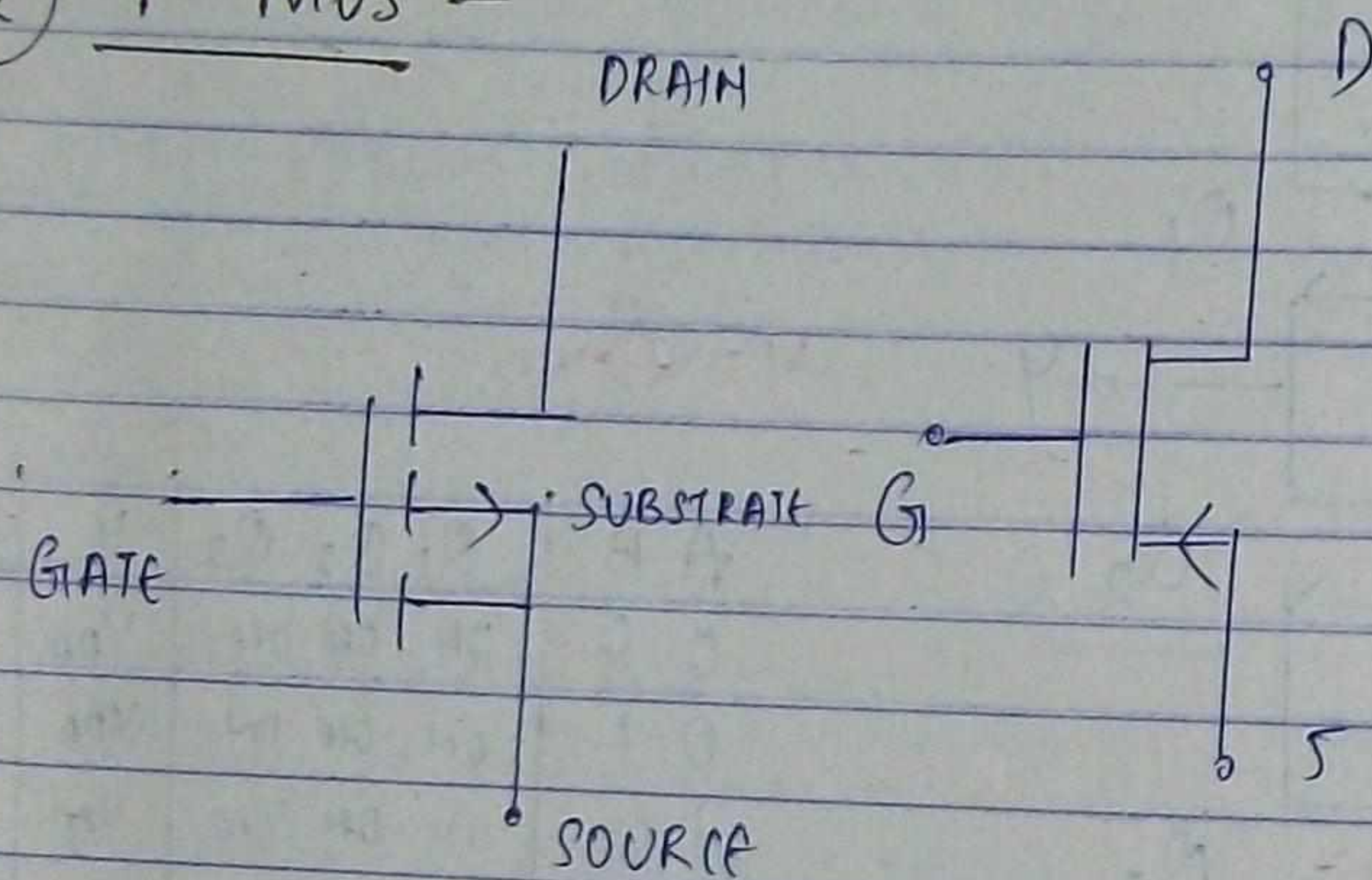
NOR GATE -



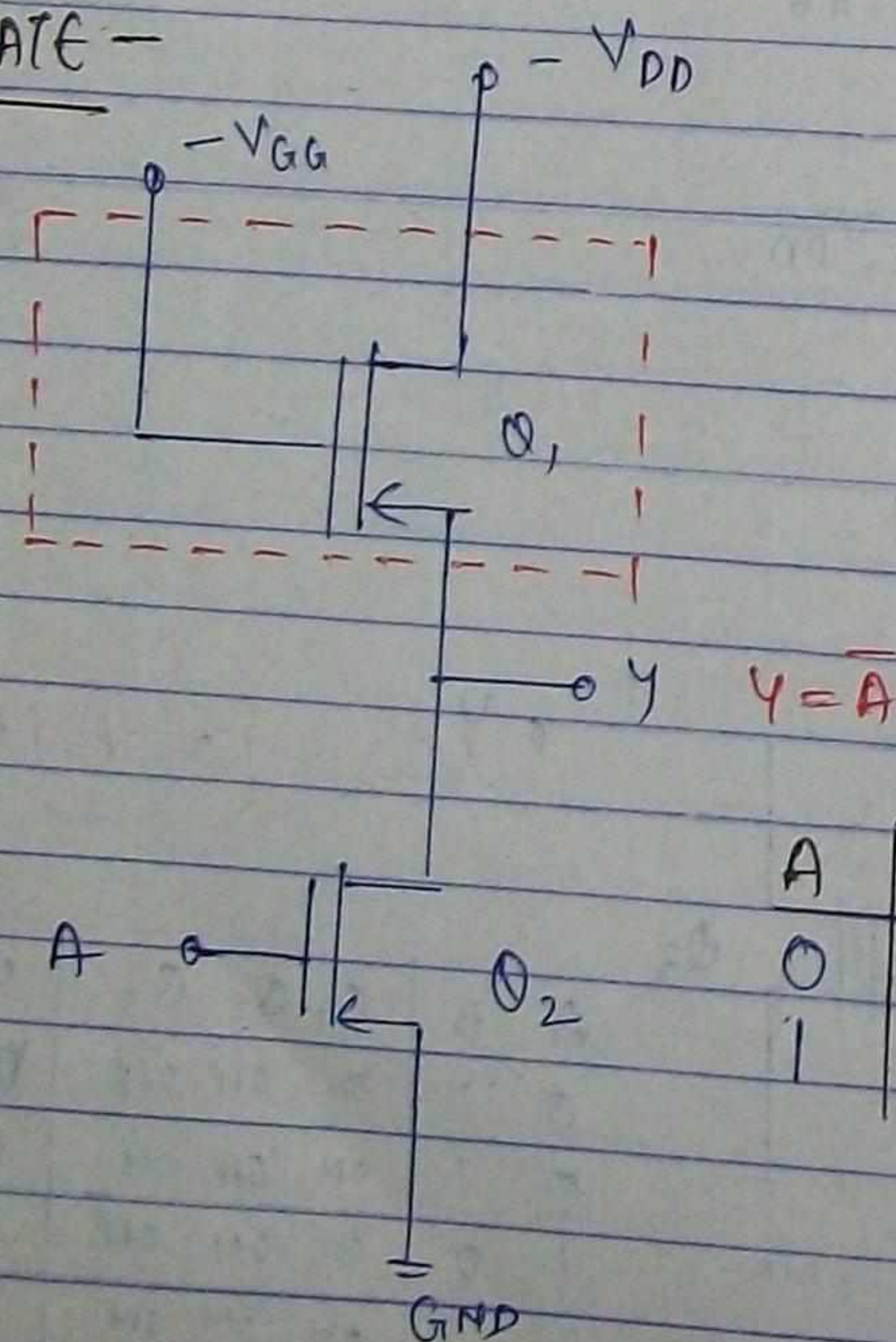
$$Y = \overline{A + B}$$

A	B	Q ₁	Q ₂	Q ₃	Y	Logic
0	0	ON	OFF	OFF	V_{DD}	1
0	1	ON	OFF	ON	0	0
1	0	ON	ON	OFF	0	0
1	1	ON	ON	ON	0	0

② P-MOS -

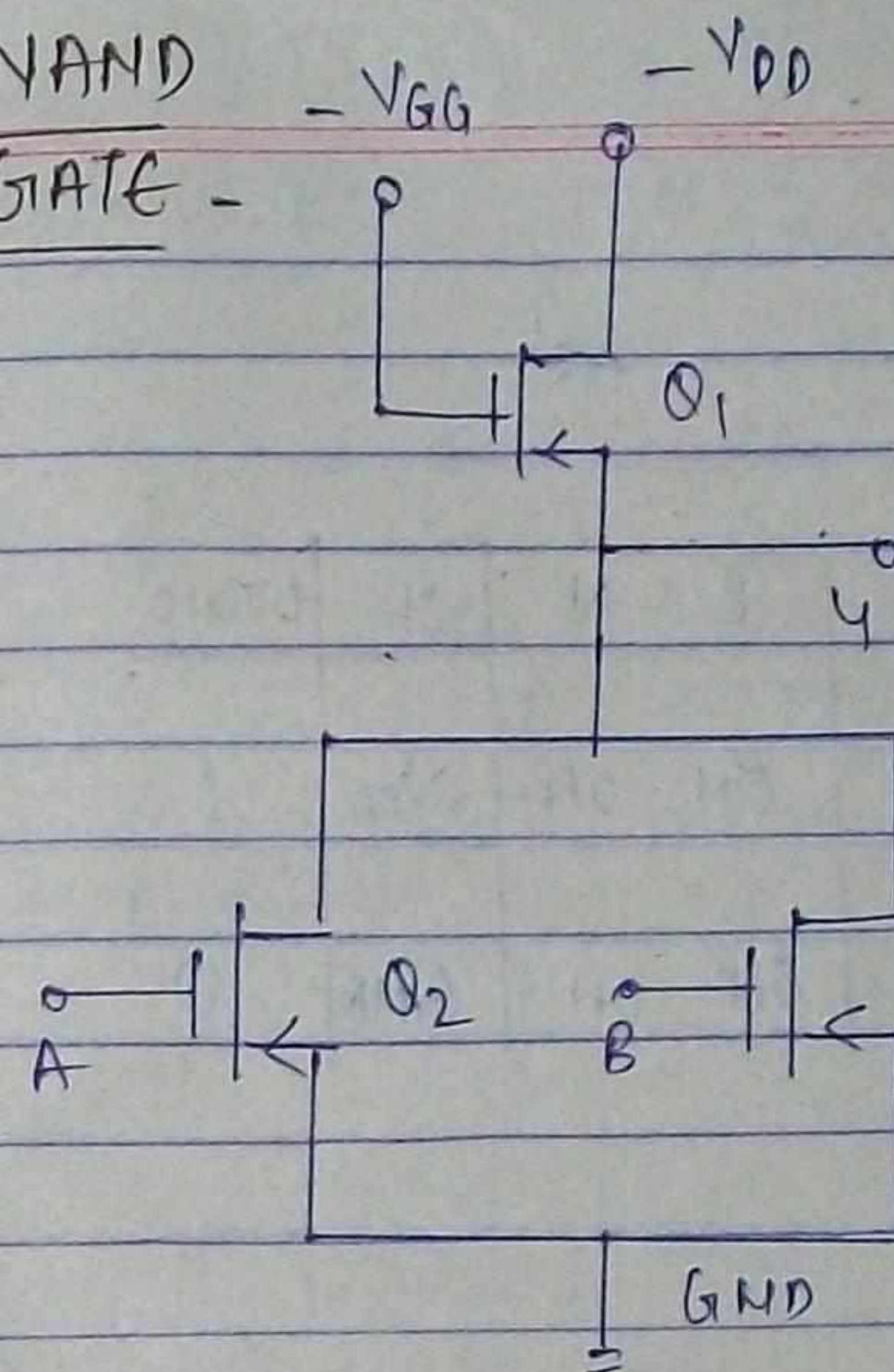


NOT GATE -



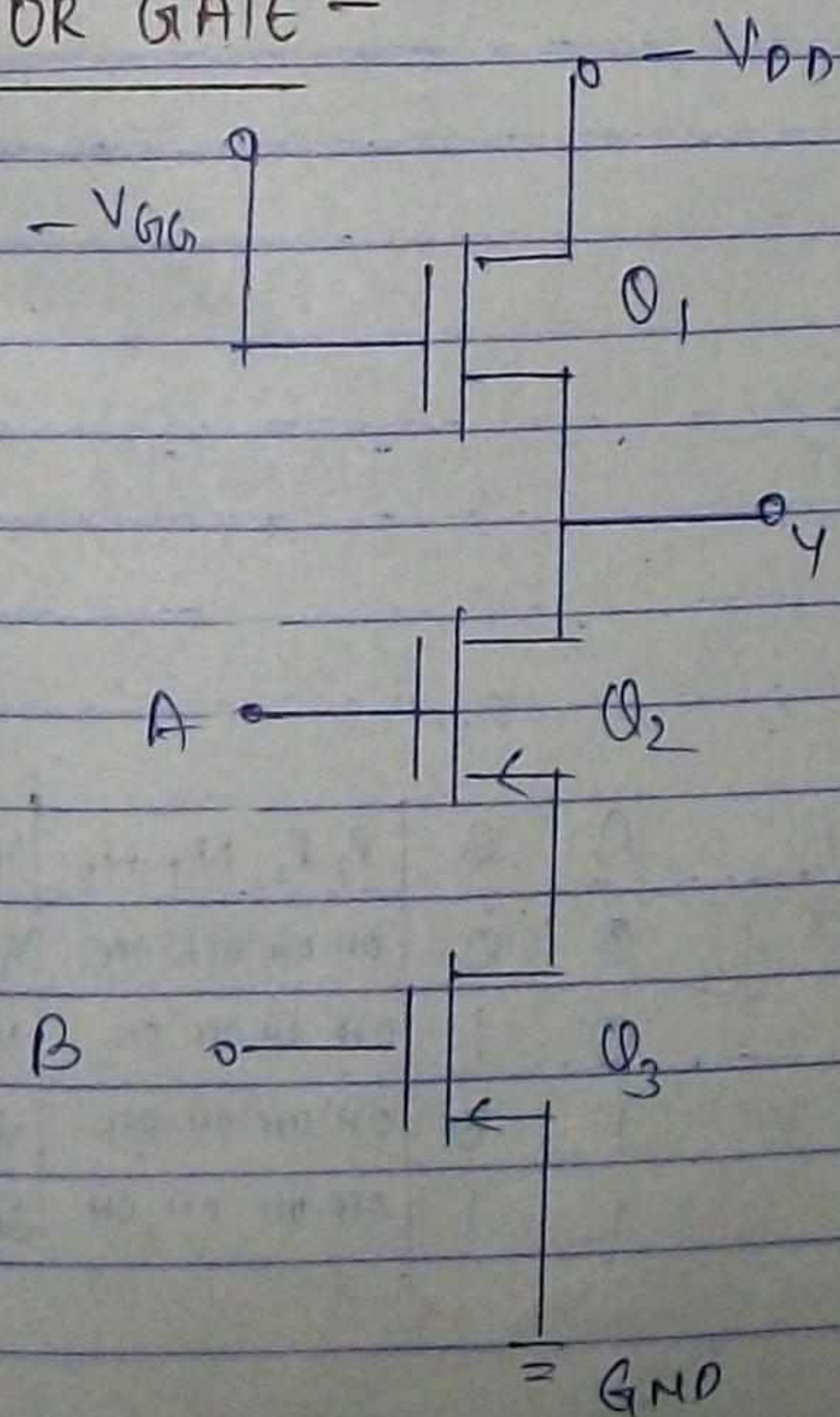
A	Q ₁	Q ₂	Y
0	ON	ON	1
1	ON	OFF	0

NAND GATE -



A	B	Q ₁	Q ₂	Q ₃	Y	LOGIC
0	0	ON	ON	ON	GND	1
0	1	ON	ON	OFF	GND	1
1	0	ON	OFF	ON	GND	1
1	1	ON	OFF	OFF	-VDD	0

NOR GATE -

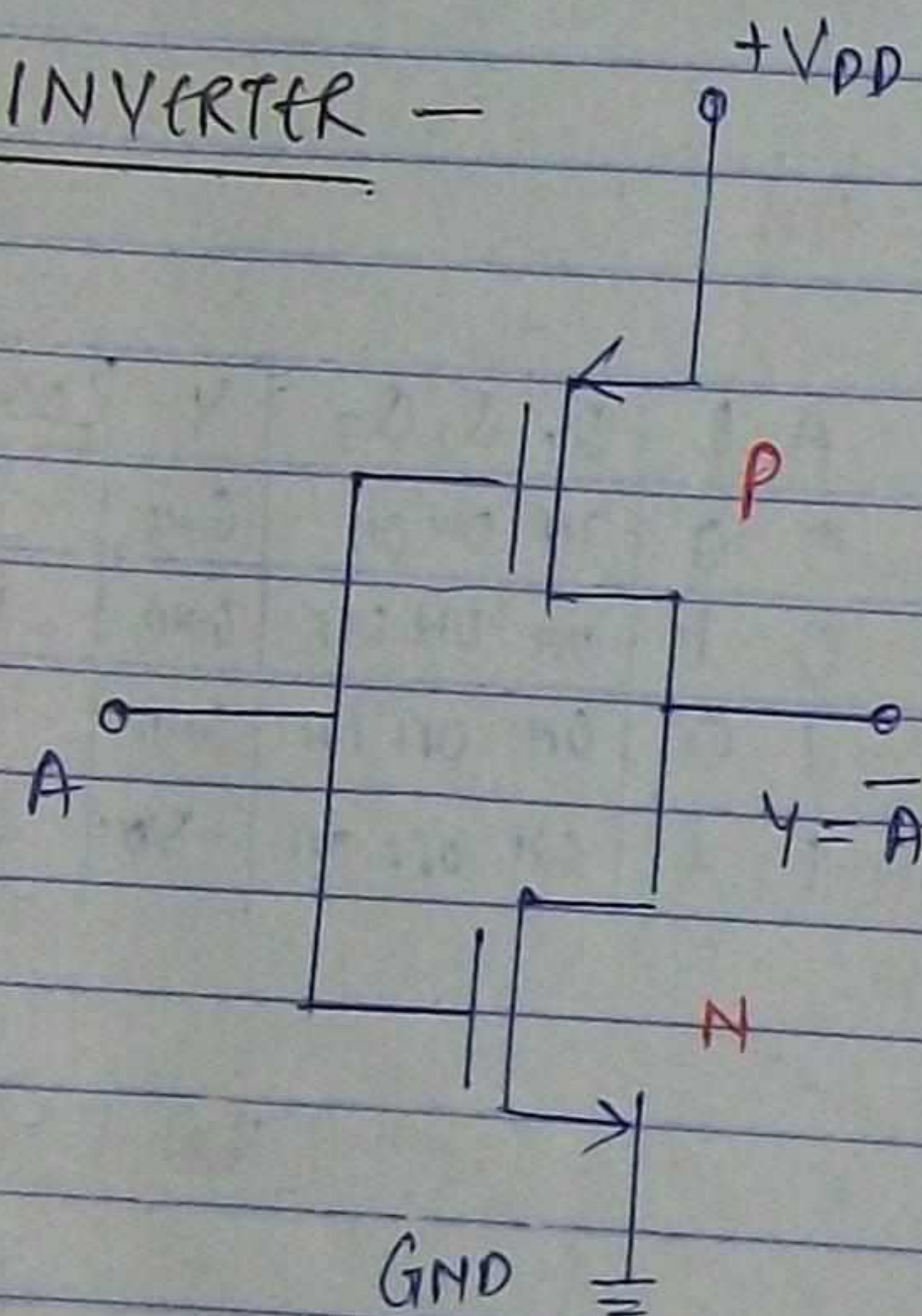


A	B	Q ₁	Q ₂	Q ₃	Y	LOGIC
0	0	ON	ON	ON	GND	1
0	1	ON	ON	OFF	-VDD	0
1	0	ON	OFF	ON	-VDD	0
1	1	ON	OFF	OFF	-VDD	0

GND - logic 1
-VDD - logic 0

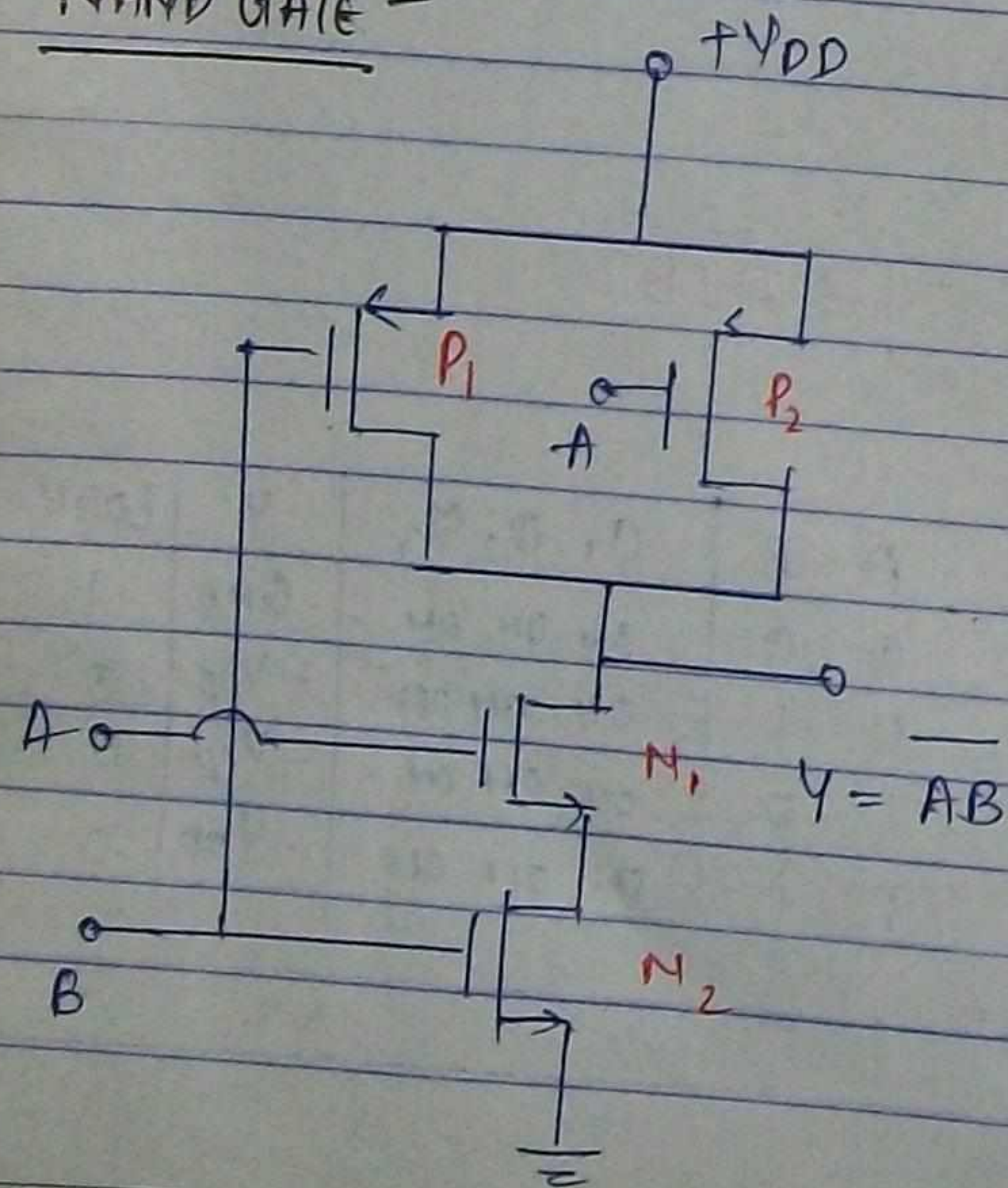
③ CMOS -

INVERTER -



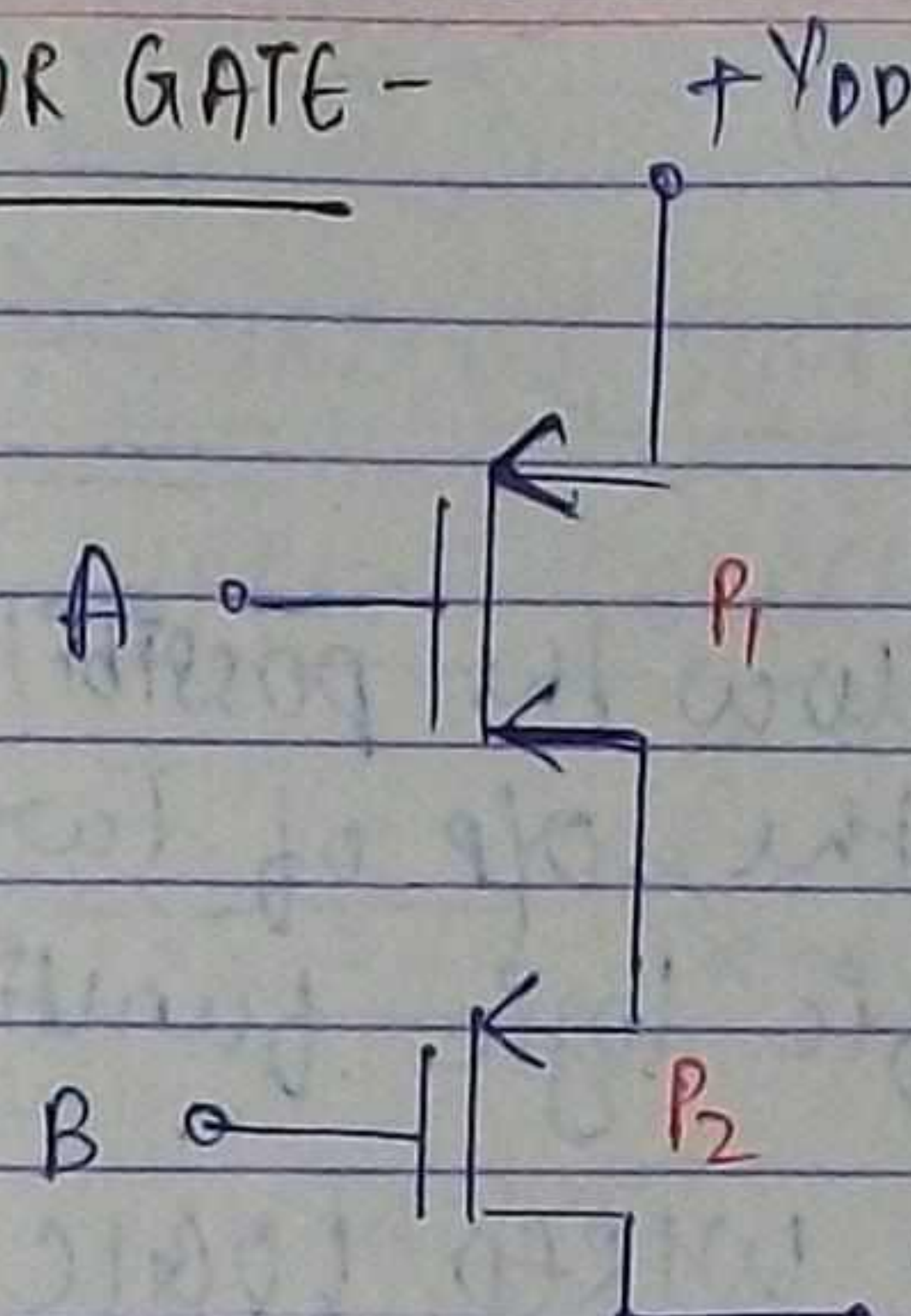
A	P	N	Y	LOGIC
0	ON	OFF	V_{DD}	1
1	OFF	ON	GND	0

NAND GATE -



A	B	P ₁ P ₂	N ₁ N ₂	Y	LOGIC
0	0	ON ON	OFF OFF	V_{DD}	1
0	1	OFF ON	OFF ON	V_{DD}	1
1	0	ON OFF	ON OFF	V_{DD}	1
1	1	OFF OFF	ON ON	GND	0

NOR GATE -



A	B	P ₁ P ₂	N ₁ N ₂	Y	LOGIC
0	0	ON ON	OFF OFF	V_{DD}	1
0	1	ON OFF	OFF ON	GND	0
1	0	OFF ON	ON OFF	GND	0
1	1	OFF OFF	ON ON	GND	0

SHORT TRICK -

COLUMN WISE - MULTIPLICATION

ROW WISE - ADDITION

TAKEN UPPER - COMPLEMENT

TAKEN LOWER - NON COMPLEMENT

NMOS

CMOS

OUTPUT

COLUMNWISE - ADDITION

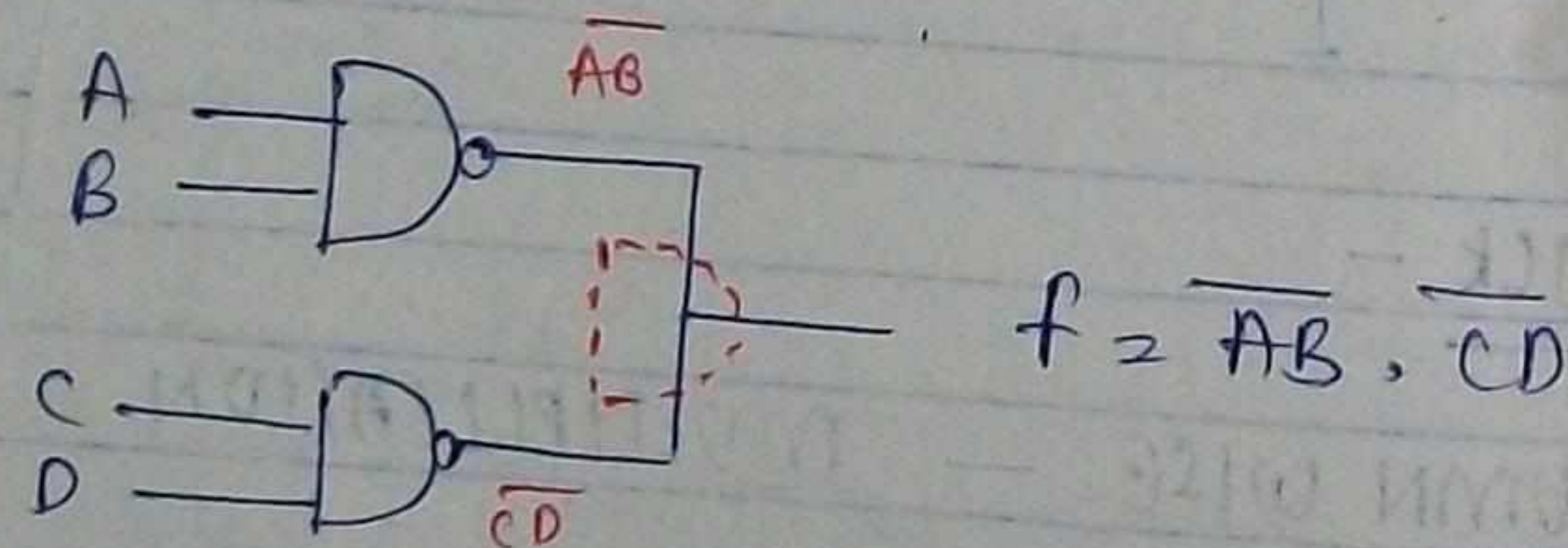
ROW WISE - MULTIPLICATION

PMOS

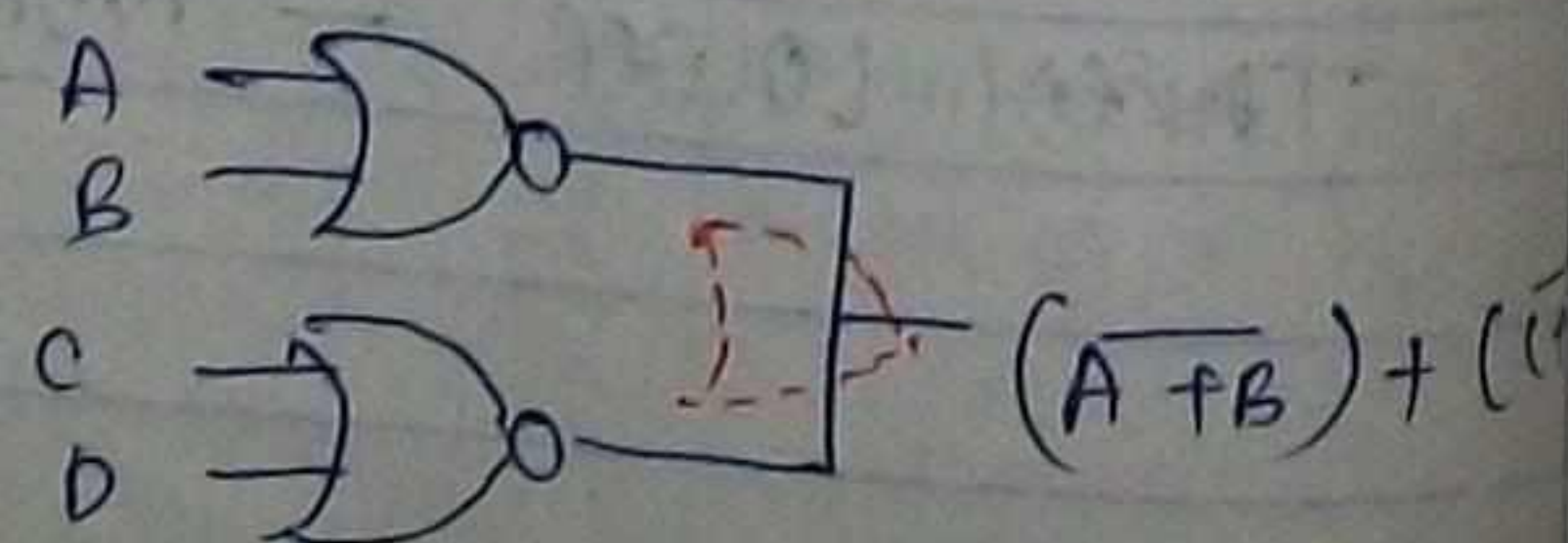
⇒ WIRED LOGIC -

Some NAND, NOR gates allow the possibility of wired connections b/w the O/P of two gates to provide a specific logic function. This type of logic is called WIRED LOGIC.

* The wired 'AND' logic can be physically implemented by connecting collector of two TTL NAND gates.



* The wired 'OR' logic can be constructed by the NOR O/P to be tied together and the logic function is given by $f = (\overline{A+B}) + (\overline{C+D})$

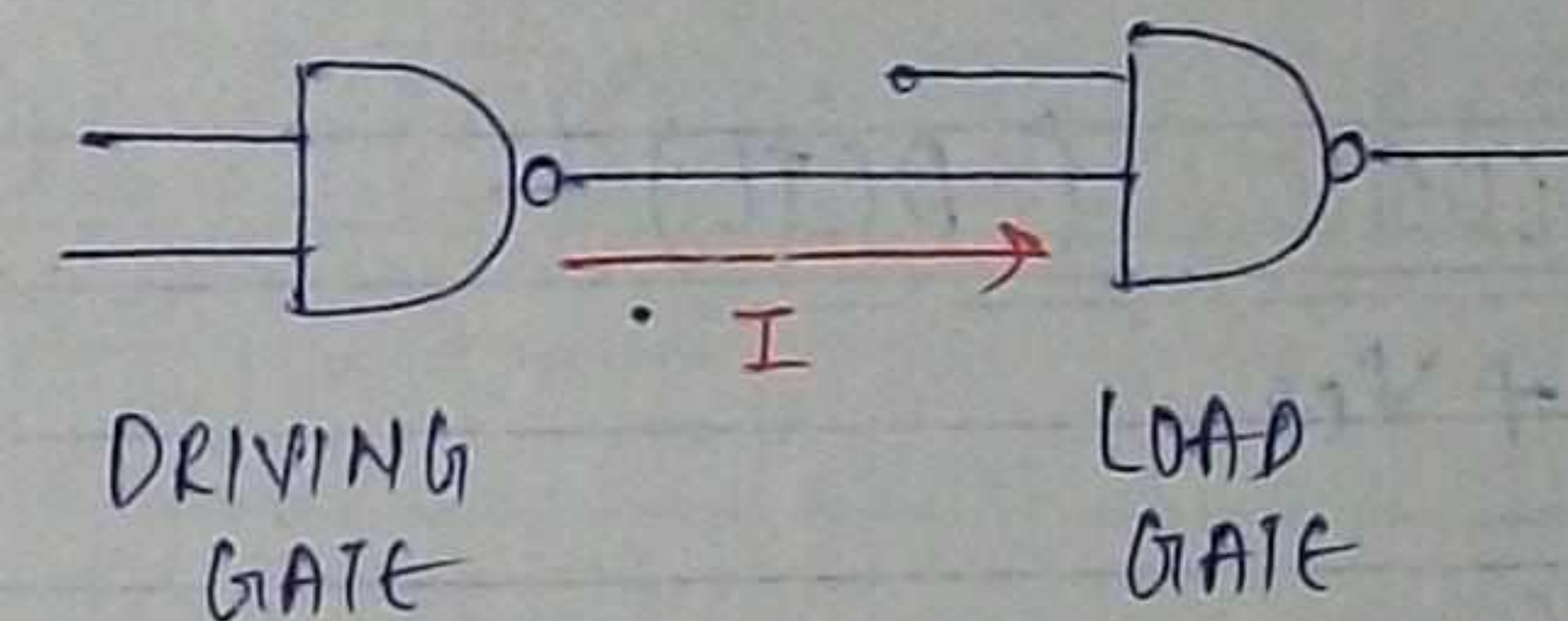


* The wired 'OR' logic can be implemented by connecting the NOR O/P of two ECL gates together.

⇒ CURRENT SOURCING & SINKING ACTIONS -

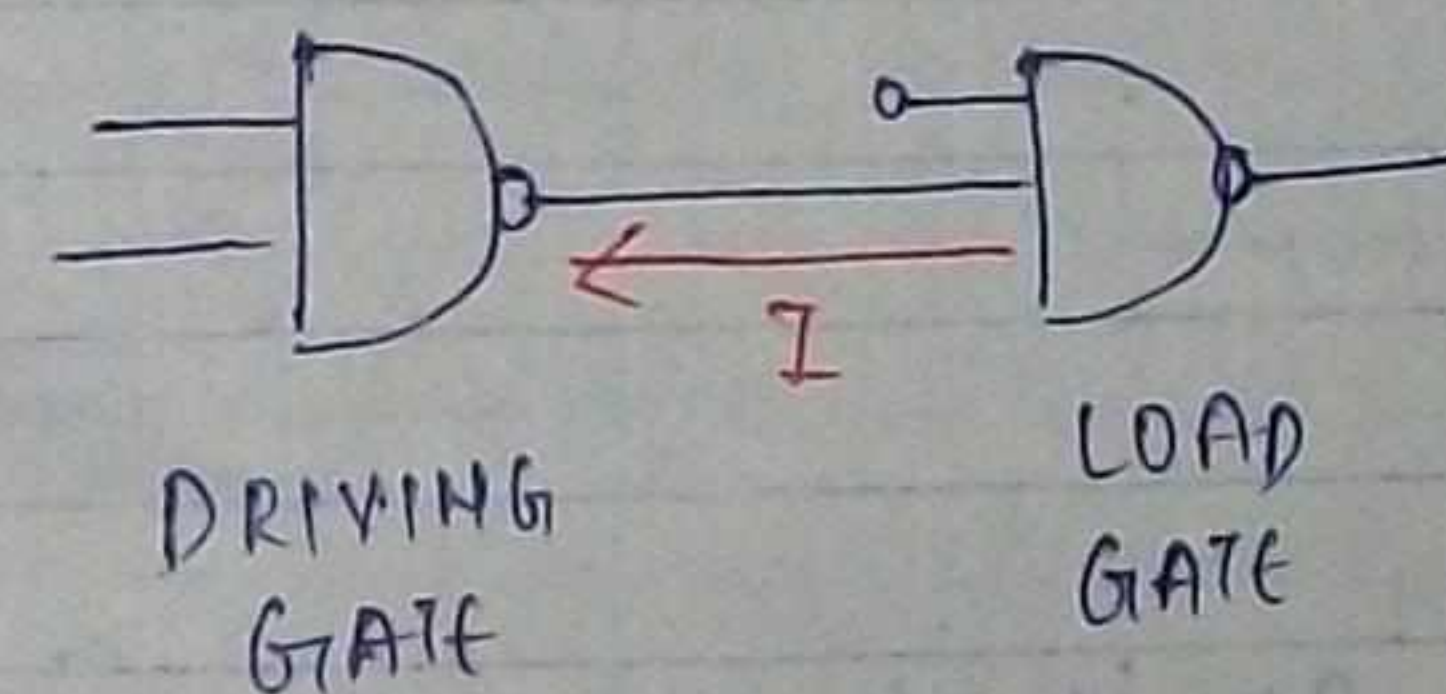
① CURRENT SOURCING -

Driving gate supplies the current. to the load is current sourcing.

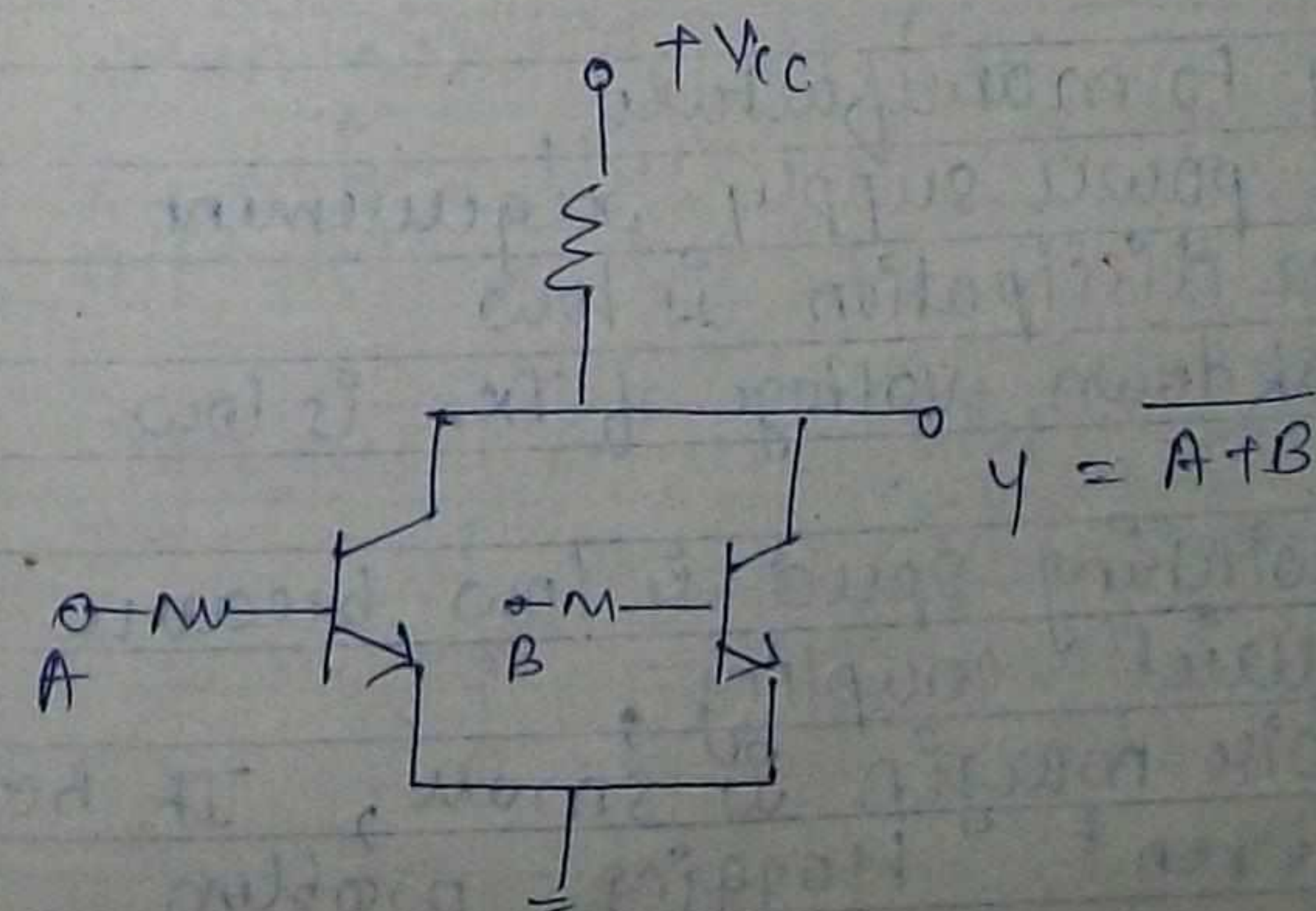


② CURRENT SINKING -

Driving gate receives the current, it is from the load gate, it is current sinking.



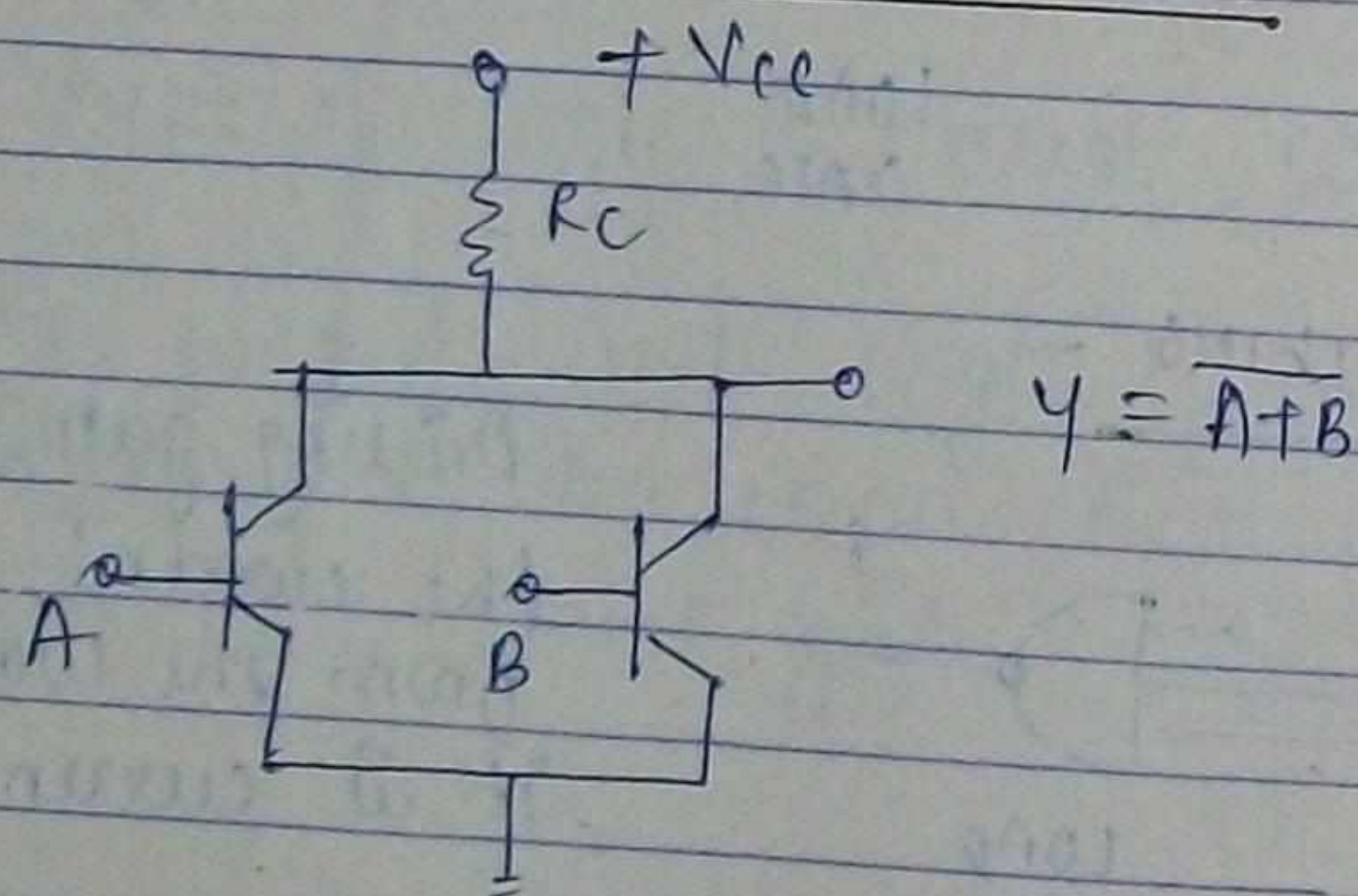
⇒ RESISTOR-TRANSISTOR LOGIC (RTL) -



ADVANTAGE - The wired AND logic is possible

DISADVANTAGE - Lower speed, less fan-out, less Noise margin.

⇒ DIRECT COUPLED TX. LOGIC (DCTL) -



- DCTL is exactly same as that of RTL circuit except 1/P resistors are removed.

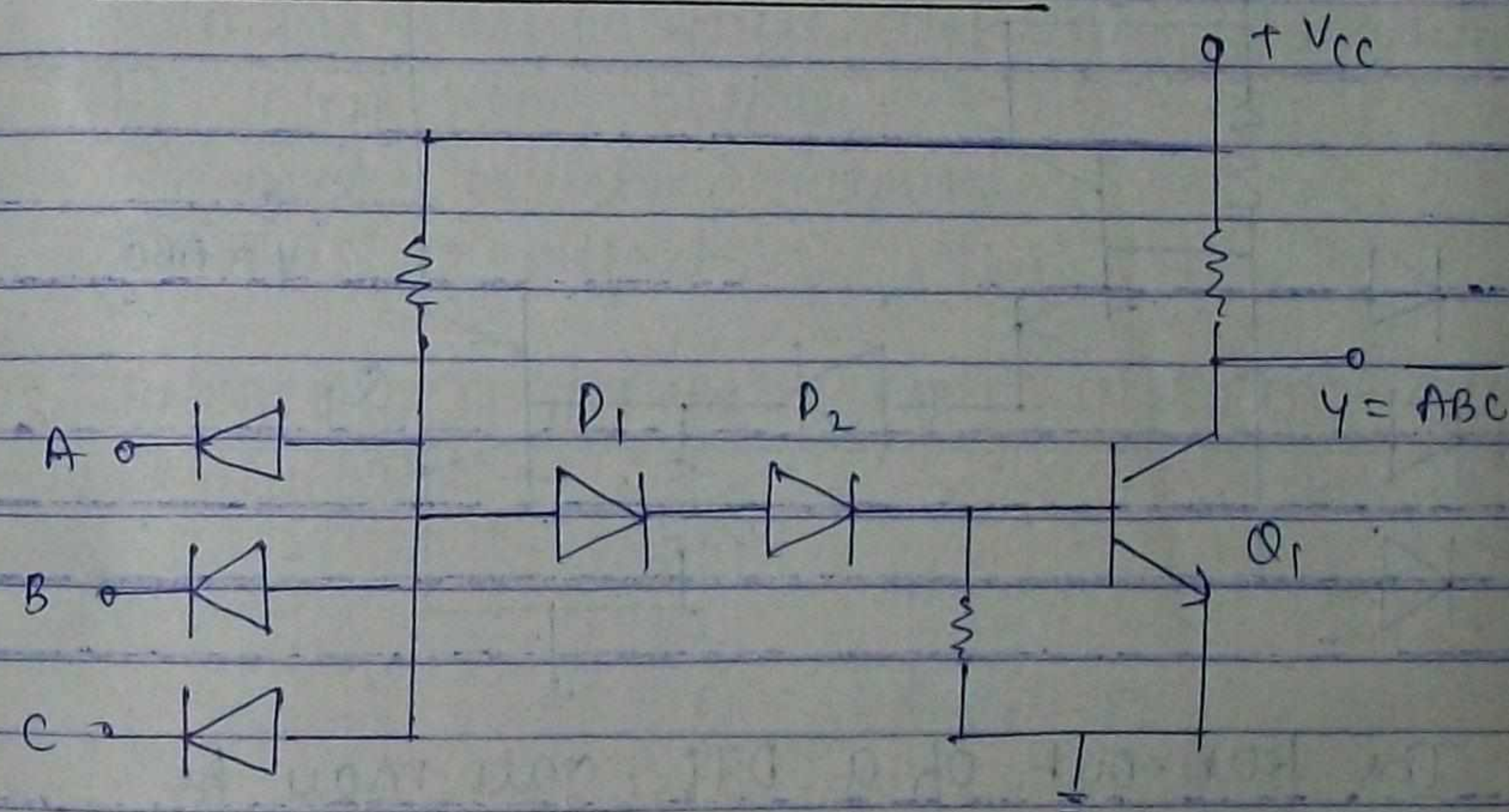
ADVANTAGES - Easy to manufacture,
low power supply requirement
power dissipation is low.
Breakdown voltage of Tx. is low

DISADVANTAGES - Switching speed is low because of direct coupling,
Noise margin is small, It has current hogging problem.

(CURRENT HOGGING)

⊛ Because of different current saturation level of Tx. Some Tx will be in the starvation (lack) of current. This situation doesn't supply the required current for some Tx.

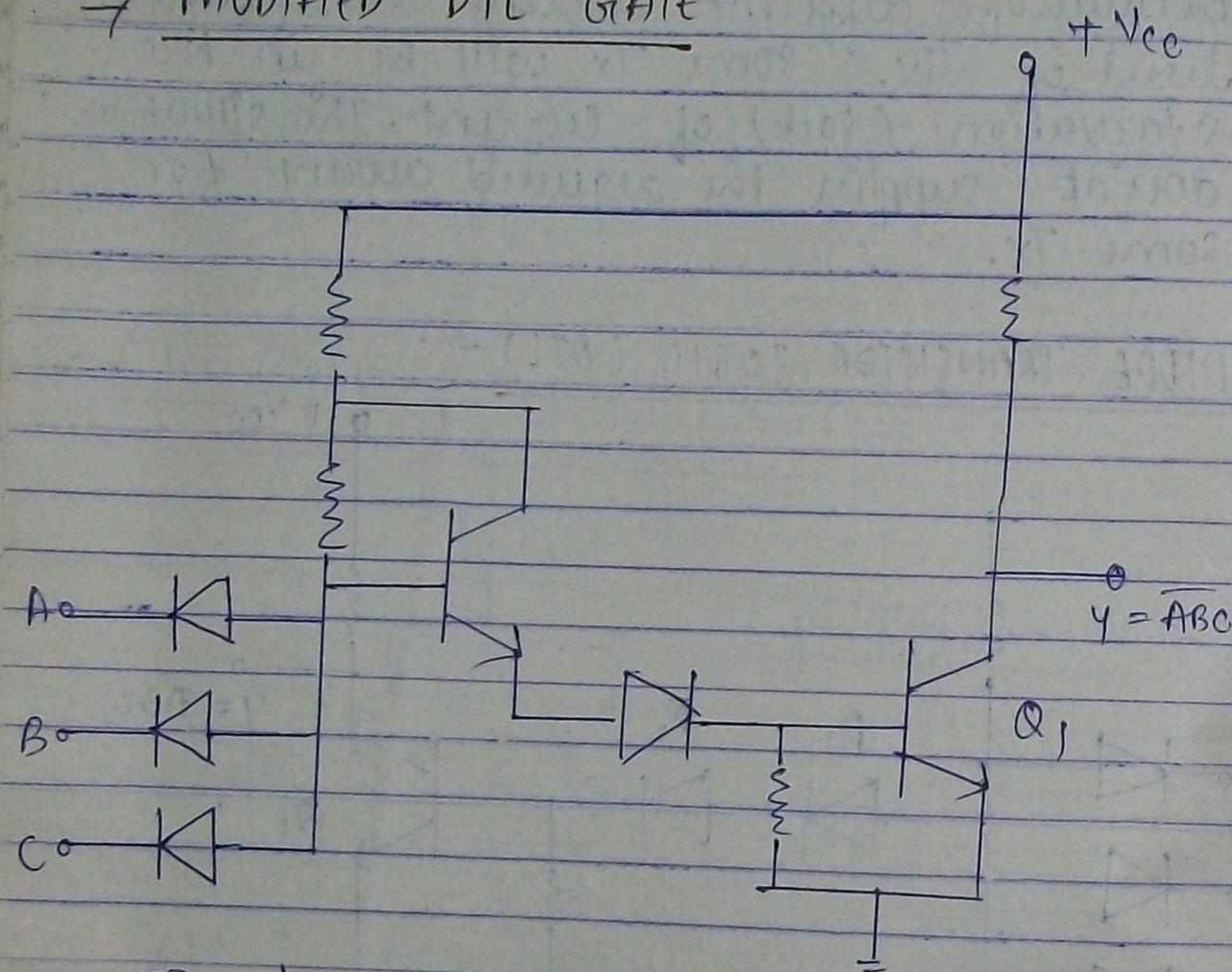
⇒ DIDDLE TRANSISTOR LOGIC (DTL) -



- When all I/Ps are low or any one of the I/P is low then Q_1 is in cutoff and the O/P voltage is high

- When all I/Ps are high, the D_1, D_2 will be ON and Q_1 is in saturation and the O/P voltage is low.

⇒ MODIFIED DTL GATE -



- The fan-out of a DTL gate may be increased by replacing one of the diode as shown in the above figure.

* High threshold logic (HTL) is a type of DTL gate only which possesses highest noise immunity.

* The improved version of DTL family which is free from the current hogging is Integrated Injection logic (IIL)

⇒ TRANSISTOR - TRANSISTOR LOGIC (TTL) -

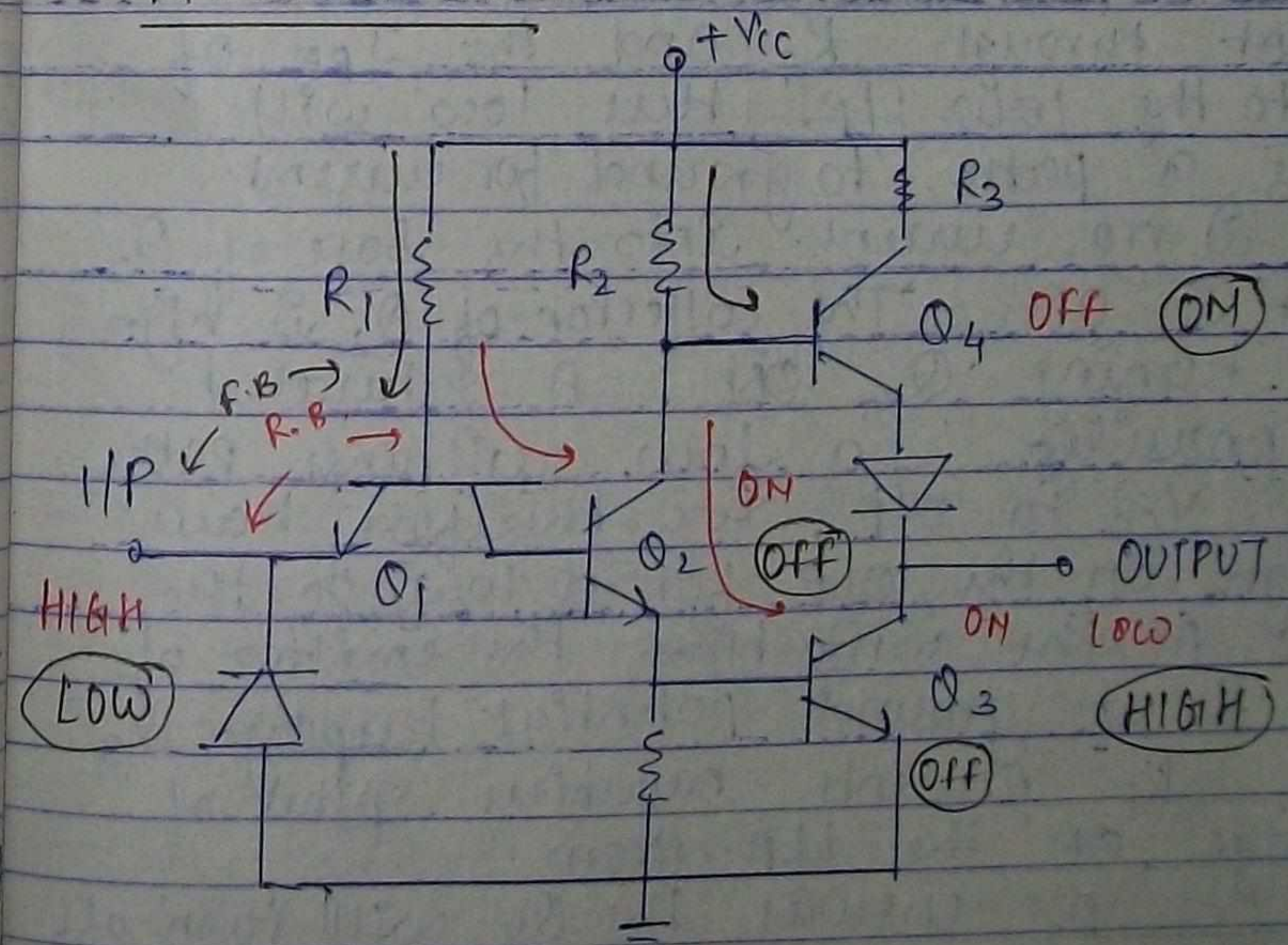
- The original basic TTL gate is a slight improvement over the DTL logic gate

- It is most widely used type.

- In TTL version three different types are there:

- ① Totem-pole output.
- ② open-collector output.
- ③ Tri-state output.

STANDARD TTL INVERTER CIRCUIT IN TOTEM-POLE ARRANGEMENT -

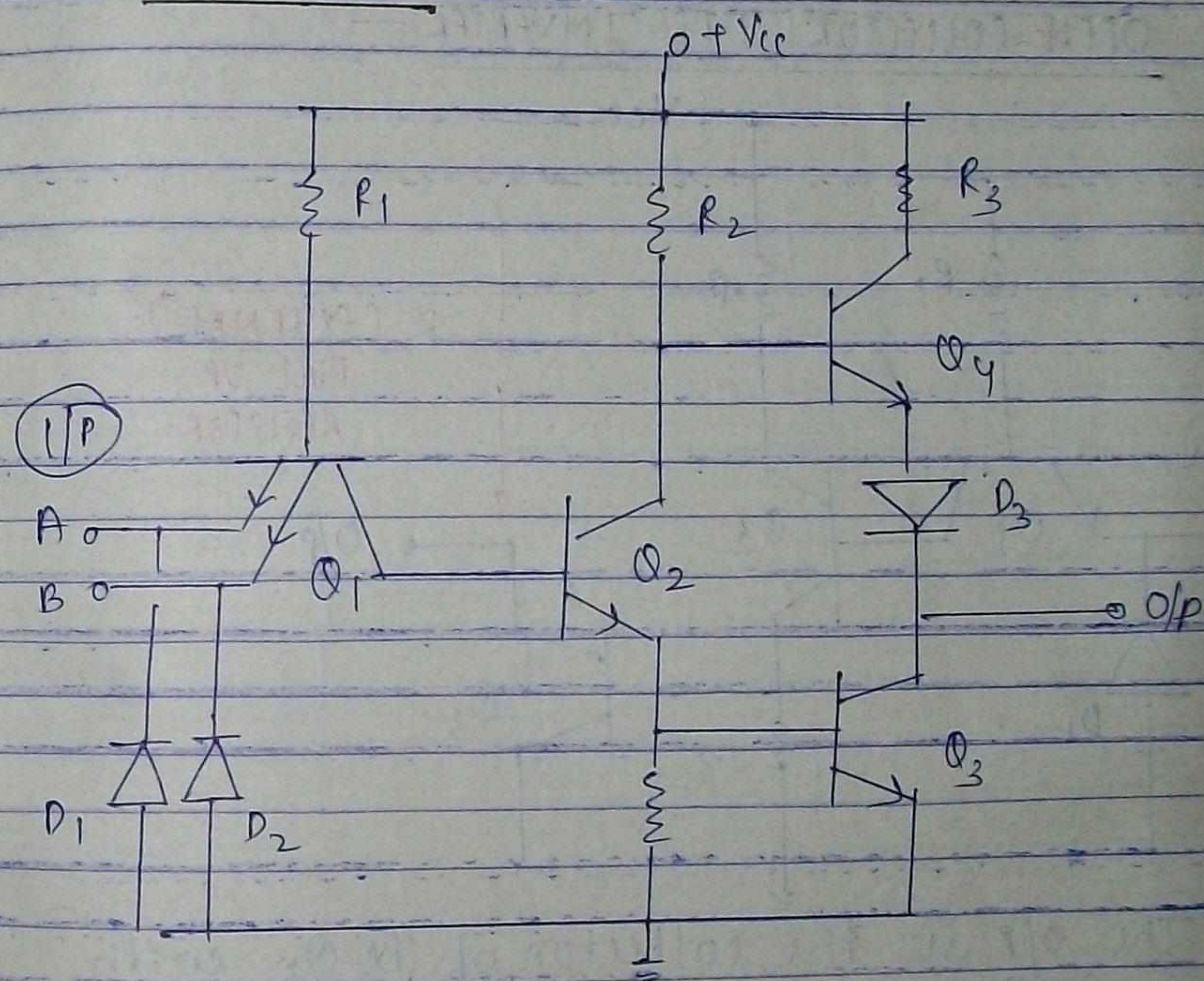


— When the I/P is High the J_{BE} of Q_1 is R.B (NPN-7x). and the J_{BC} is F.B. This condition permits current through R_1 and the J_{BC} of Q_1 and it makes Q_2 to go into saturation, as a result Q_3 is turned ON by Q_2 and its collector voltage which is the O/P is near ground potential. we therefore have a low O/P for a High I/P. At the same time the collector of Q_2 is sufficiently at low voltage level to keep Q_4 off.

— When the I/P is low the J_{BE} of Q_1 is F.B. and the J_{BC} is R.B, there is a current through R_1 and the J_{BE} of Q_1 to the low I/P. Here low will provide a path to ground for current. There is no current into the base of Q_2 so it is off. The collector of Q_2 is high. Thus turning Q_4 ON. A saturated Q_4 provides a low resistance path from V_{CC} to O/P. we therefore have a high on the O/P for a low on the I/P. At the same time the emitter of Q_2 is at ground potential keeping Q_3 off. D_1 prevents negative spikes of voltage on the I/P from D_2 ensures the Q_4 will turn off.

when Q_2 is ON.

⇒ NAND GATE - (OPEN COLLECTOR LOGIC) -

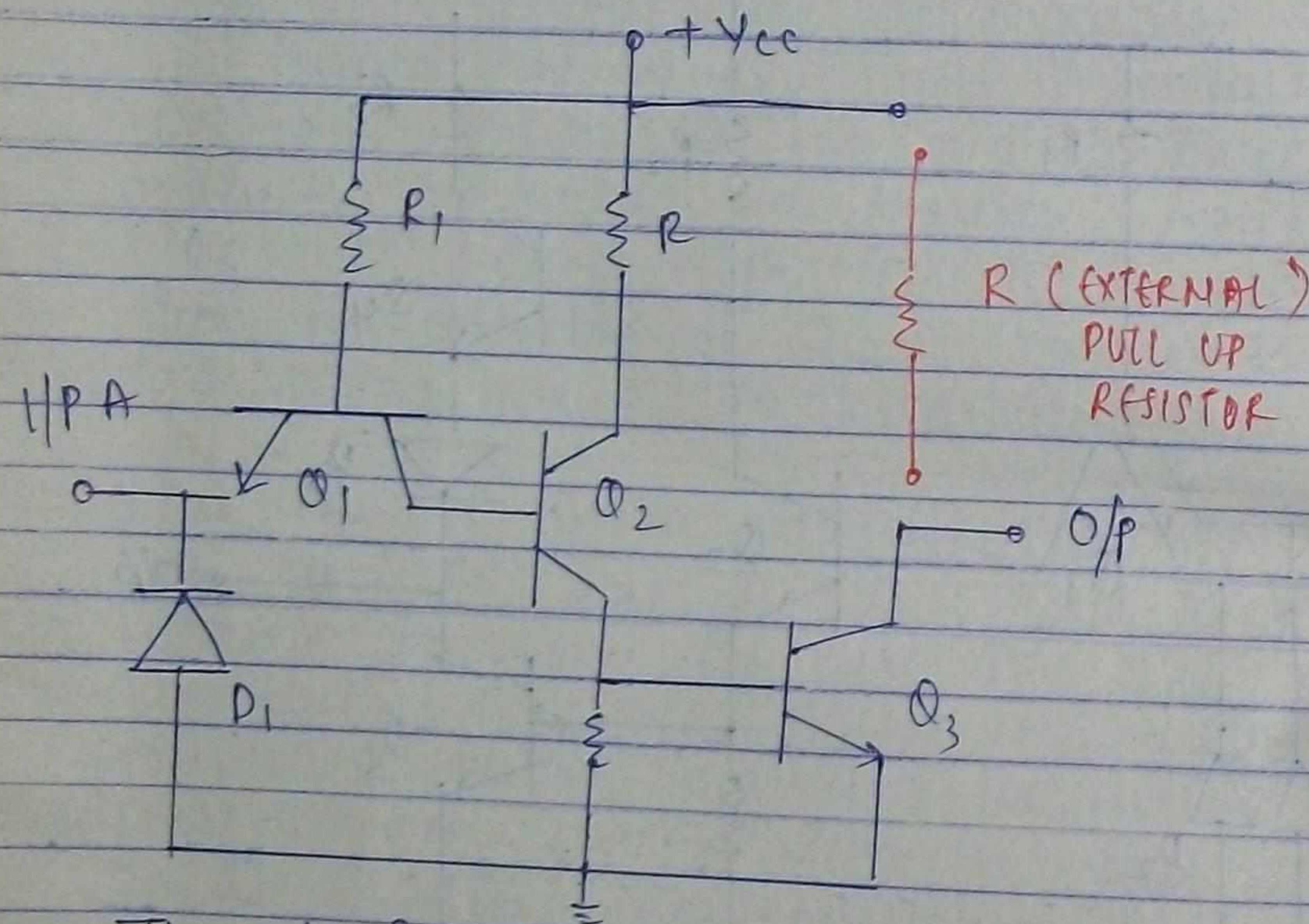


— A low on either I/P A or I/P B, F.B is the reverse-biased diode and R.B is D_3 . ($Q_1 - J_{BC}$) this action keeps Q_2 off and results in a high O/P. as the

— A High on both I/Ps R.B the both I/P diodes and F.B D_3 (Q_1 base-collector Junction)

This action turns Q_2 ON and results in a low O/P.

⇒ OPEN COLLECTOR TTL INVERTER —



— The O/P is the collector of Tx Q_3 with nothing connected to it hence the name is OPEN COLLECTOR.

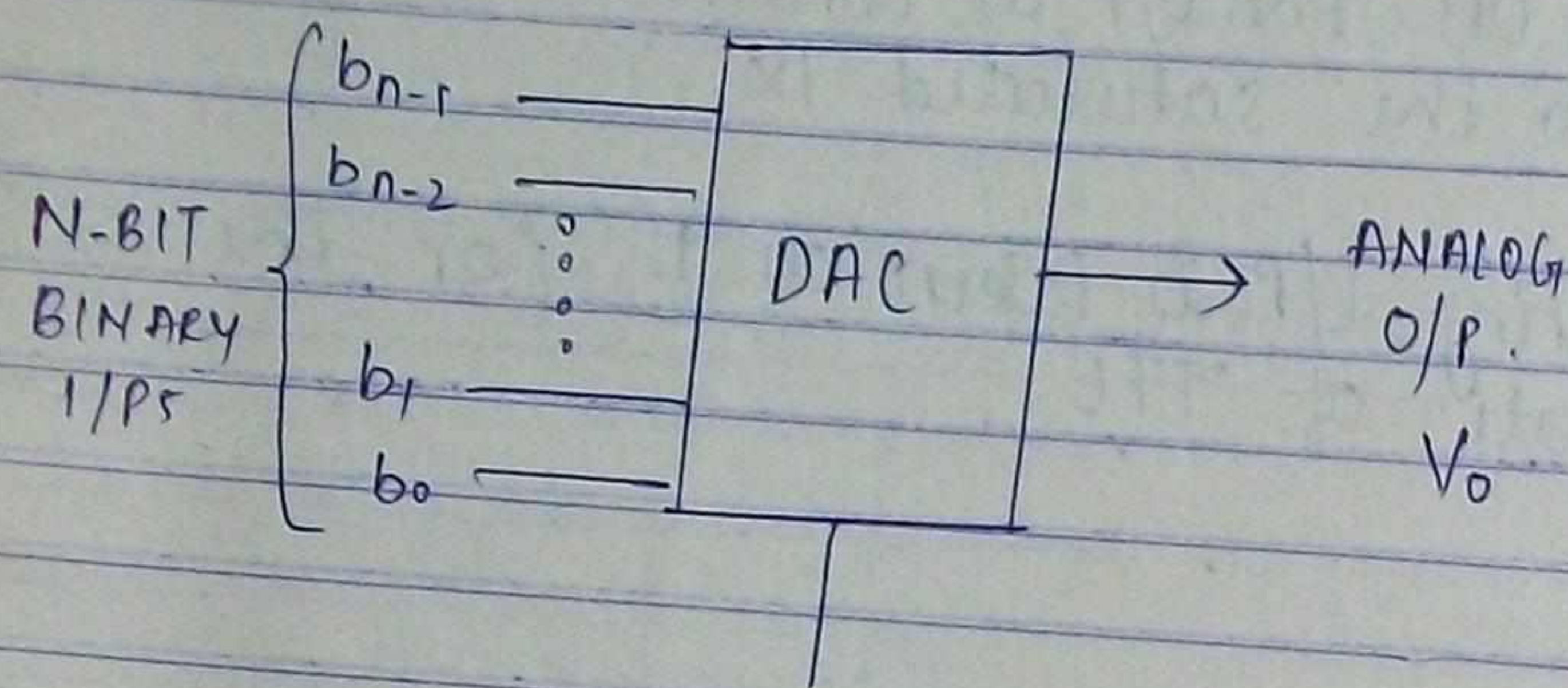
— In order to get the proper high and low logic level of the output, an external pull-up resistor is connected to V_{cc} from the collector of Q_3 . When Q_3 is OFF, O/P is pulled up to V_{cc} through external pull up resistor.

TABULAR FORM → BOOK (THEORY)

When Q_3 is ON the O/P is connected to near ground to the saturated Tx.

* The floating I/P is taken as 1 for the AND gate of TTL.

⇒ DIGITAL TO ANALOG CONVERTERS - 16/10/11



$$V_o = K \left[2^{N-1} b_{N-1} + 2^{N-2} b_{N-2} + \dots + 2^2 b_2 + 2^1 b_1 + b_0 \right]$$

Analog output = $K \times$ Digital input

where K = proportionality constant.

$b_n = 1$ if n^{th} bit of digital I/P is '1'

$b_n = 0$ if n^{th} bit of digital I/P is '0'.

Ques. What is the largest value of O/P from an 8-bit DAC that produces 1V for a digital I/P of 00110010 = 50

Solⁿ:

(largest value)

11111111 = 255 → largest analog.

$$\text{largest analog O/P} = K \times (\text{Digital I/P}) = K \times (255)$$

$$\text{also } 1V = K (50)$$

$$K = \frac{1}{50} = 20 \text{ mV}$$

$$\therefore \text{largest analog O/P} = \frac{255}{50} = 5.1 \text{ V}$$

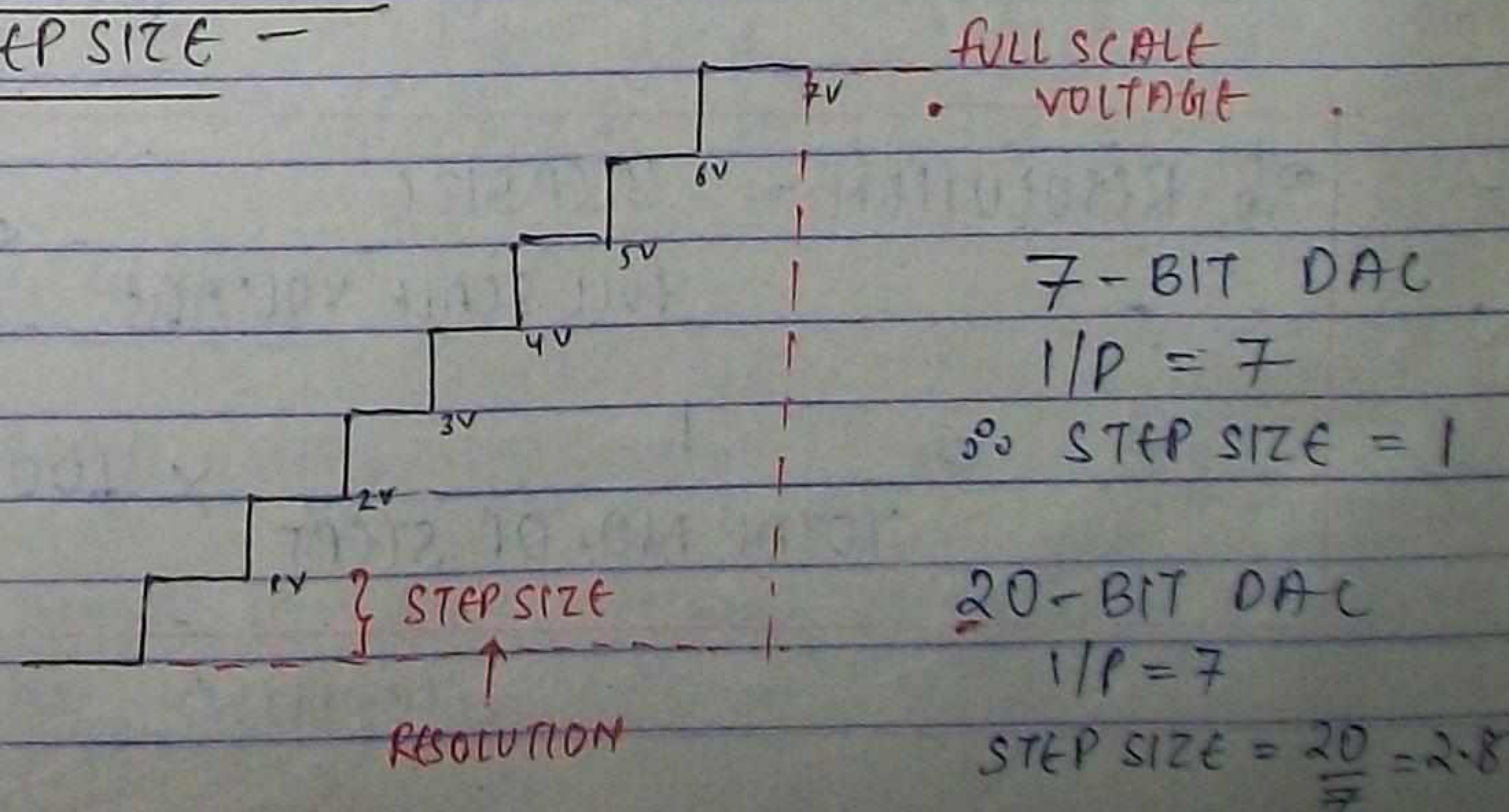
Ques. A 5-bit DAC produces $V_o = 0.1 \text{ V}$ for a digital I/P of 00001, then find the value of V_o when I/P is 11111.

Solⁿ:

$$\frac{0.1}{1} = K$$

$$\text{largest analog O/P} = 0.1 \times 31 = 3.1 \text{ V}$$

* RESOLUTION OR STEP SIZE -



⇒ SPECIFICATIONS FOR DAC -

① RESOLUTION - (STEP SIZE)

It is defined as the smallest change that can occur in the analog o/p as a result of change in the digital i/p.

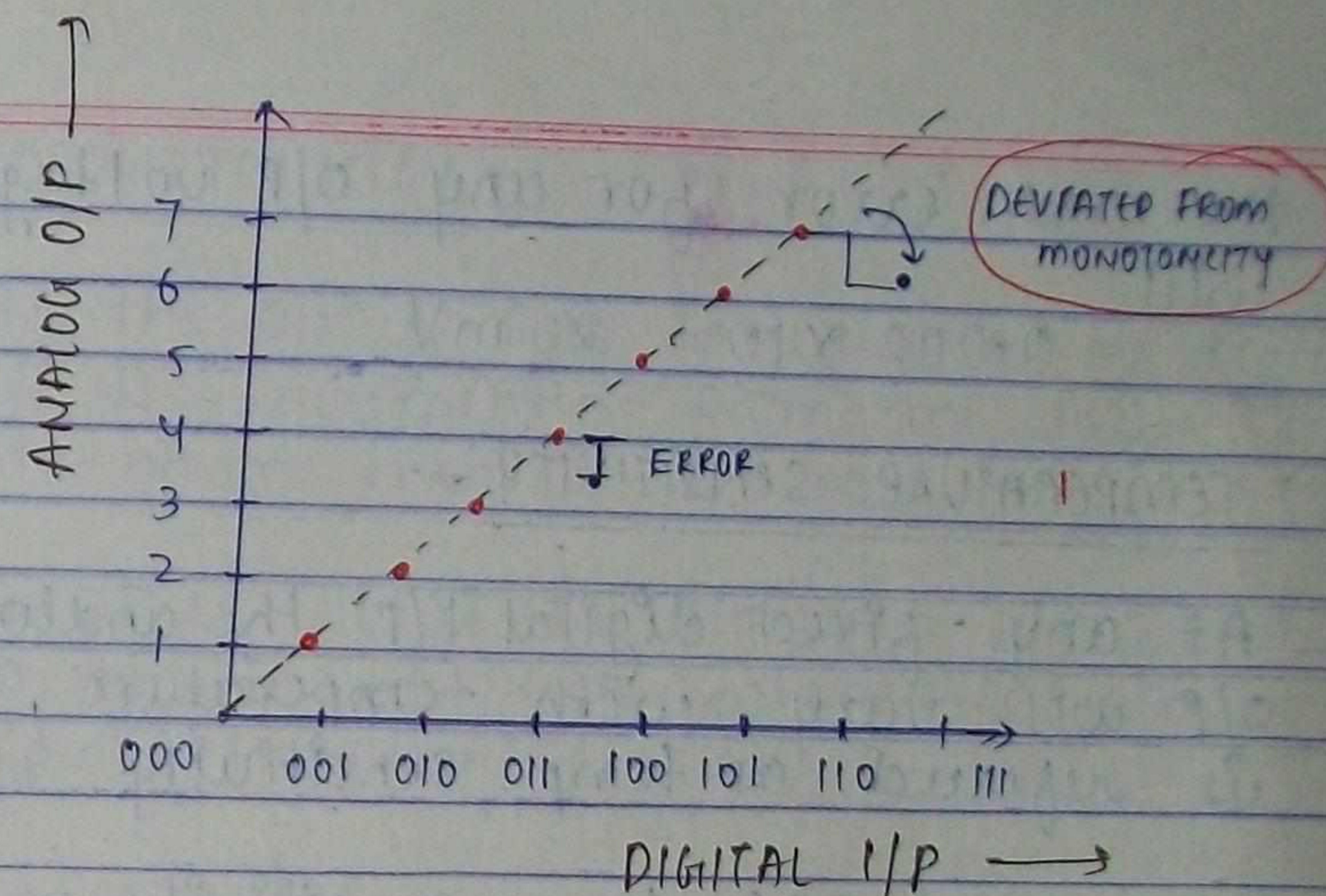
- It is also known as STEP SIZE. And is always equal to the weight of the LSB.

$$\text{STEP SIZE} = \frac{\text{FULL SCALE ANALOG VOLTAGE}}{\text{NO. OF STEPS}}$$

$$\text{STEP SIZE} = \frac{V_a}{2^n - 1} \quad \text{no. of bits}$$

V_a = full scale analog voltage

$$\begin{aligned} \% \text{ RESOLUTION} &= \frac{\text{STEP SIZE}}{\text{FULL SCALE VOLTAGE}} \times 100 \% \\ &= \frac{1}{\text{TOTAL NO. OF STEPS}} \times 100 \% \end{aligned}$$



② LINEARITY -

The linearity of a converter is a measure of the precision with which the linear i/p-o/p relationship is satisfied.

- The linearity error for a digital i/p is the difference b/w the voltage corresponding to the dot (blue) and the voltage obtained from the straight line (expected o/p). This is indicated by (e).

③ ACCURACY -

It is the difference b/w actual o/p voltage and expected o/p voltage.

- For example a DAC has 10 V full scale V_o and the accuracy is $\pm 0.2\%$ then

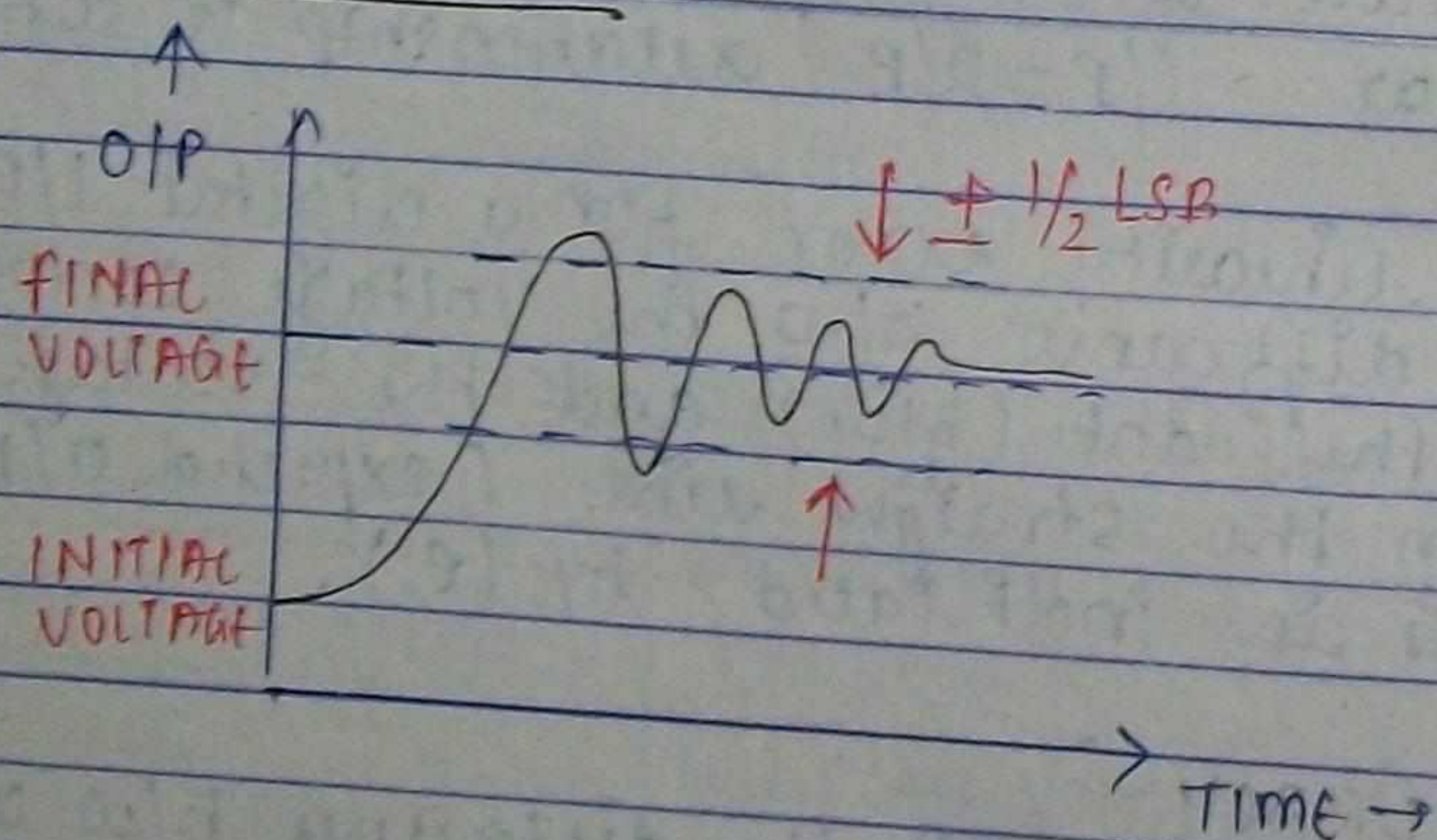
the max. error for any o/p voltage will

$$0.002 \times 10 = 20 \text{ mV}$$

④ TEMPERATURE SENSITIVITY -

- At any fixed digital I/P the analog o/p will vary with temperature, this is referred as temp. sensitivity.
- for a good DAC temp. sensitivity must be minimum.

⑤ SETTLING TIME -



- The time required for the analog o/p to settle within $\pm 1/2 \text{ LSB}$ of the final value after a change in the digital I/P.

⑥ MONOTONICITY -

- If the o/p of DAC increases as the digital I/P is incremented from one value to the other then it is monotonic.
- In other words the staircase o/p will have no downward steps when the binary I/P is incremented.

⑦ OFFSET ERROR -

- Ideally the o/p of DAC will be zero volt when all binary I/Ps are zero, but in practice there will be small o/p voltage known as offset error.

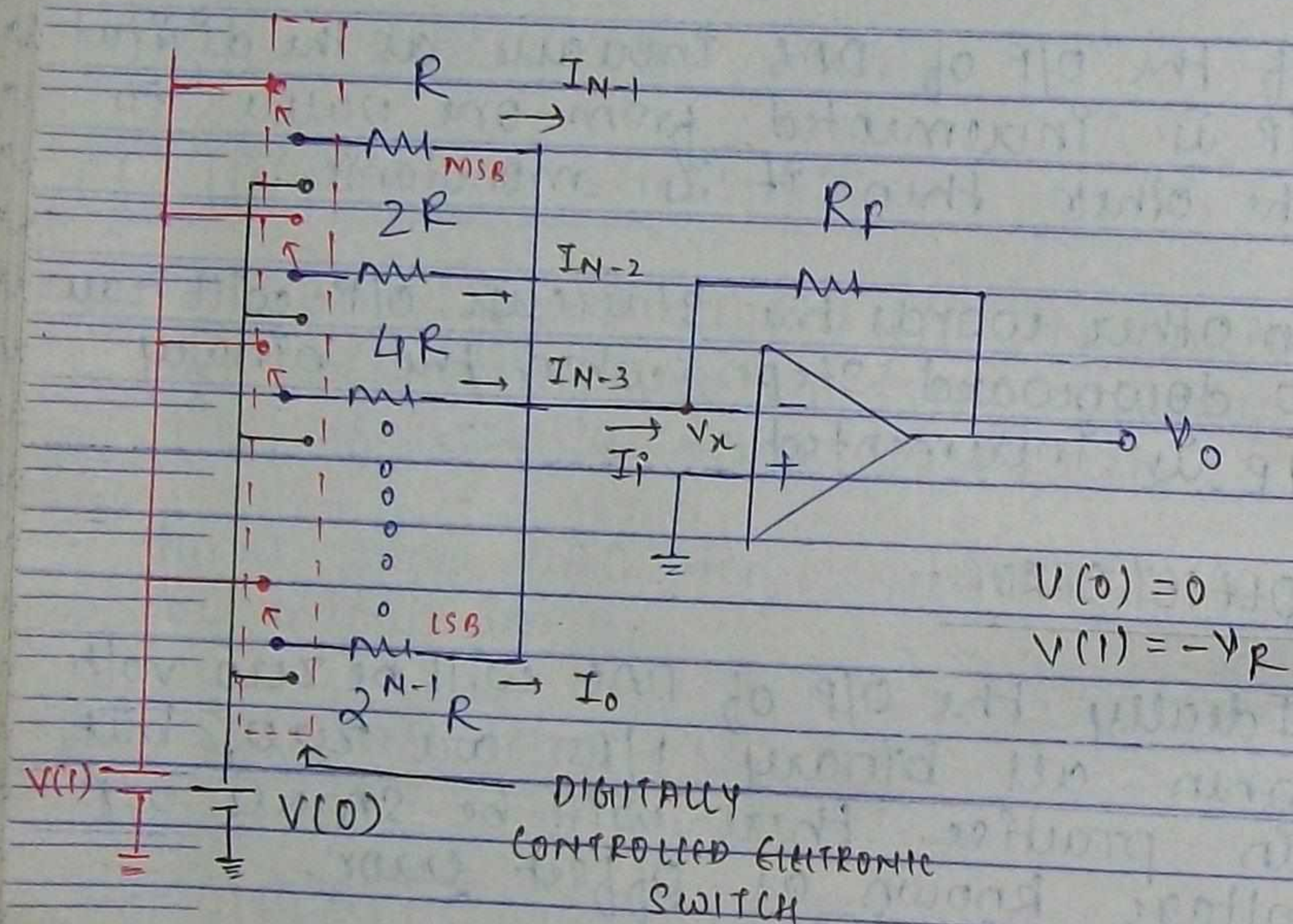
⇒ TYPES OF DACS -

① Weighted resistor DAC.

② R-2R Ladder

- Inverting type
- Non-Inverting type

① WEIGHTED RESISTOR DAC —



② OUTPUT VOLTAGE —

$$V_0 = -(-V_R) \left[\frac{R_f b_{N-1}}{R} + \frac{R_f b_{N-2}}{2^1 R} + \frac{R_f b_{N-3}}{2^2 R} + \dots + \frac{R_f b_0}{2^{N-1} R} \right]$$

$$K = \frac{R_f}{2^{N-1} R} V_R$$

$$I_i = I_{N-1} + I_{N-2} + \dots + I_2 + I_1 + I_0$$

$$I_{N-1} = \frac{V_{N-1}}{R}$$

$$I_{N-2} = \frac{V_{N-2}}{2R}$$

...

$$I_0 = \frac{V_0}{2^{N-1} R}$$

where,

$$V_n = V(1) \text{ if } b_n = 1$$

$$V_n = V(0) \text{ if } b_n = 0$$

DISADVANTAGES —

— Due to large difference in resistor value i.e. b/w MSB and LSB we cover a wide range of resistance and which track over a wide temp. range are difficult to produce.

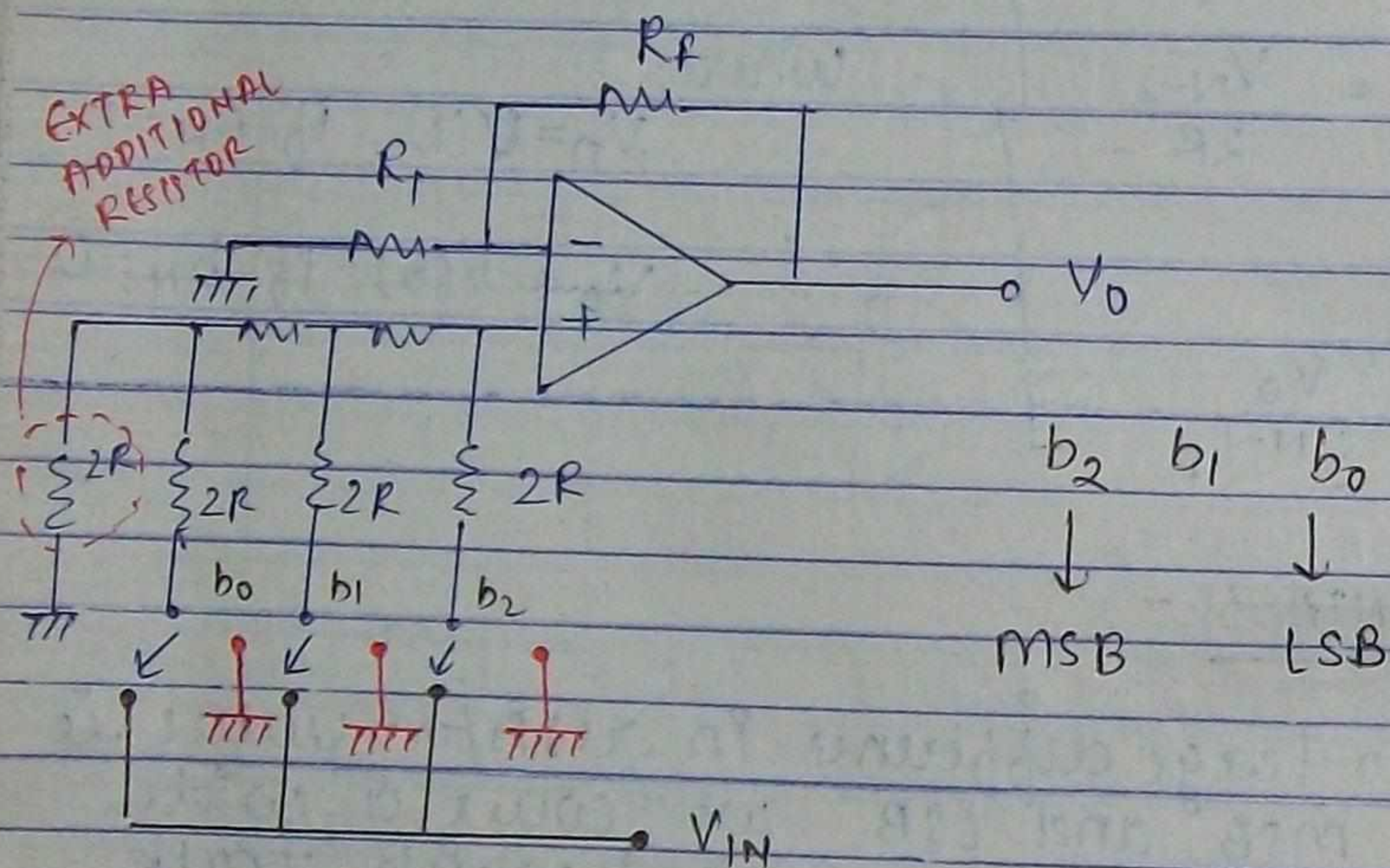
— Due to the loading effect I/P impedance will not remain constant.

— Accuracy of the system is less.

— Linearity is less.

② R-2R LADDER TYPE DAC -

(A) NON INVERTING OPAMP -



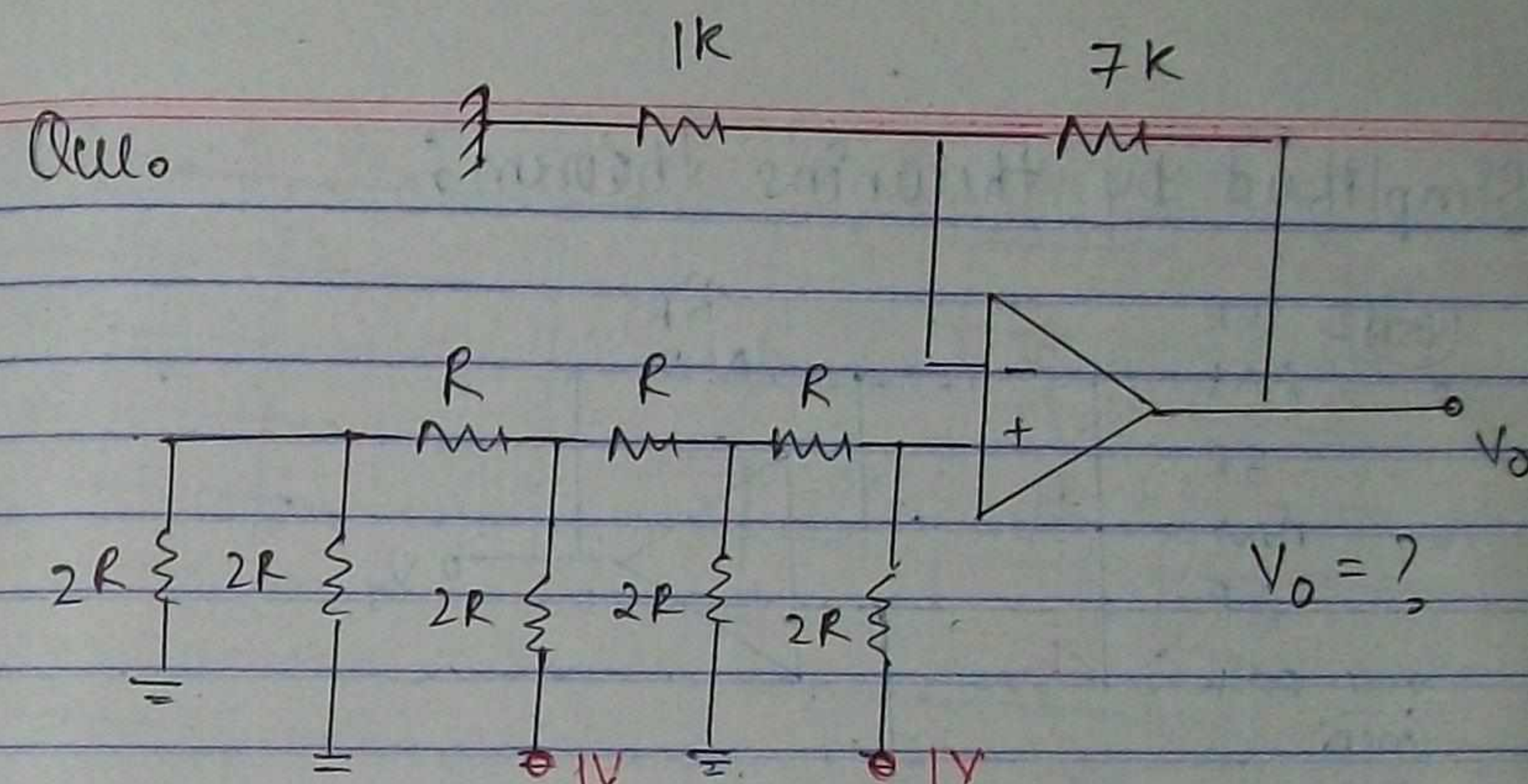
$$\text{Gain of Opamp} = \frac{V_o}{V_i} = \left[1 + \frac{R_f}{R_1} \right]$$

OUTPUT VOLTAGE -

$$V_o = \frac{V_i}{2^N} \left[\sum_{i=0}^{N-1} 2^i b_i \right] \times \left[1 + \frac{R_f}{R_1} \right]$$

$$V_o = \frac{V_i}{2^N} \times \text{Decimal equivalent of binary data} \times \text{Gain}$$

$N = \text{NO. OF BITS}$

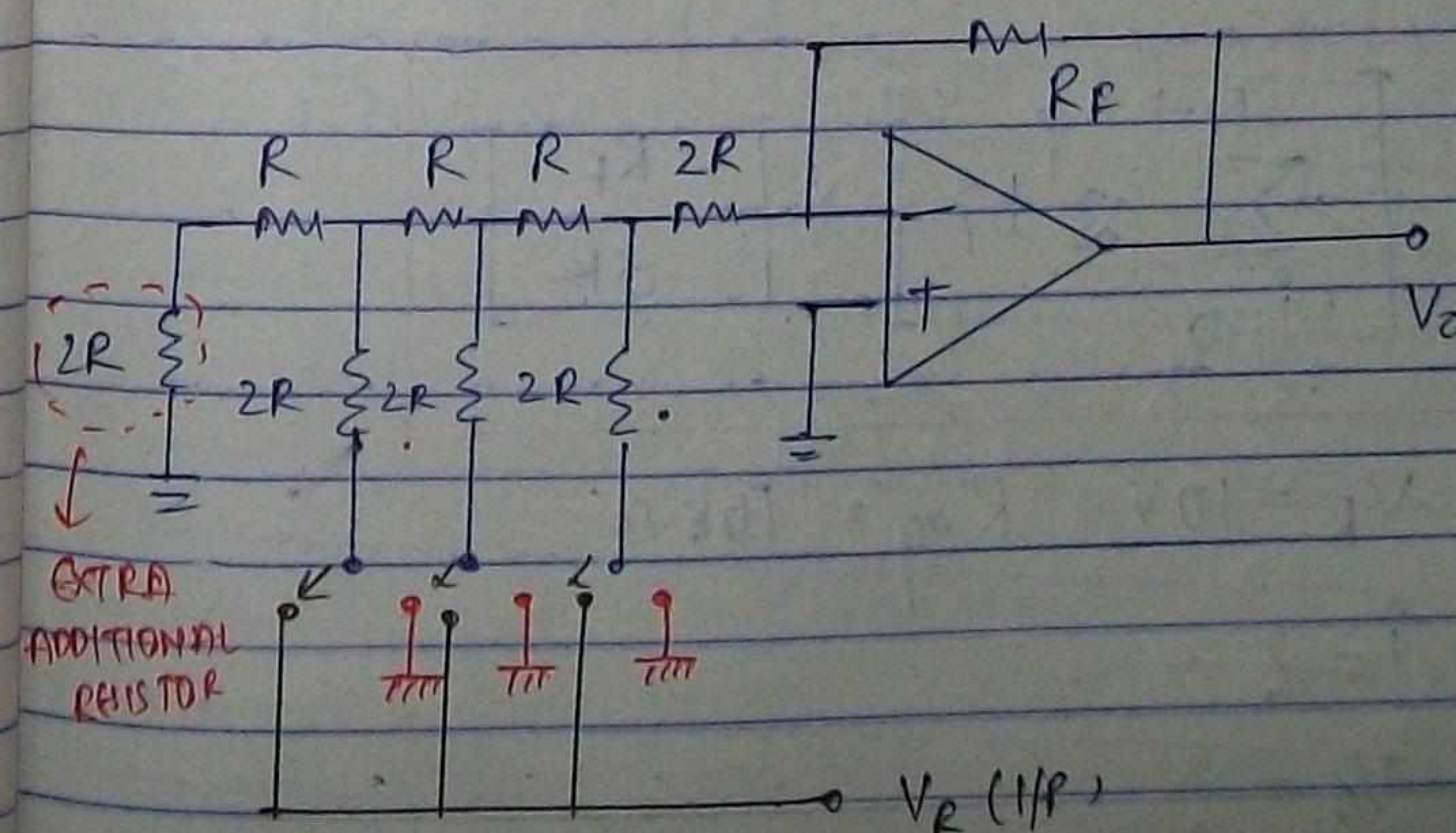


$$\text{Sol}^n: \text{Gain} = \left(1 + \frac{7}{1} \right) = 8$$

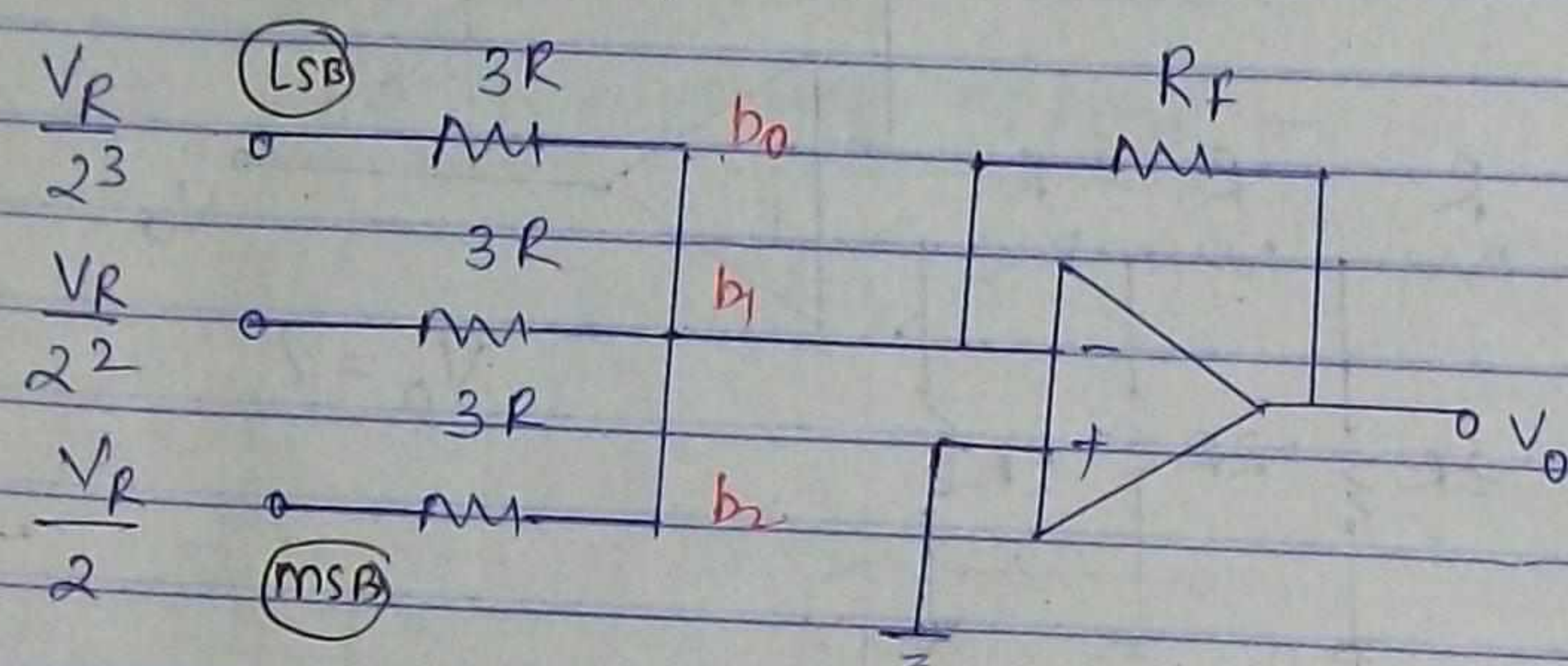
$$V_o = \frac{V_i}{2^N} \times (1010) \times 8$$

$$= \frac{1}{2^4} \times 10 \times 8 = \frac{80}{16} = \underline{\underline{5V}}$$

(B) INVERTING OPAMP -



Simplified by thevenin's theorem:



OUTPUT VOLTAGE-

$$V_0 = - \left[\frac{R_f}{3R} \cdot \frac{V_R}{2^3} b_0 + \frac{R_f}{3R} \cdot \frac{V_R}{2^2} b_1 + \frac{R_f}{3R} \cdot \frac{V_R}{2} b_2 \right]$$

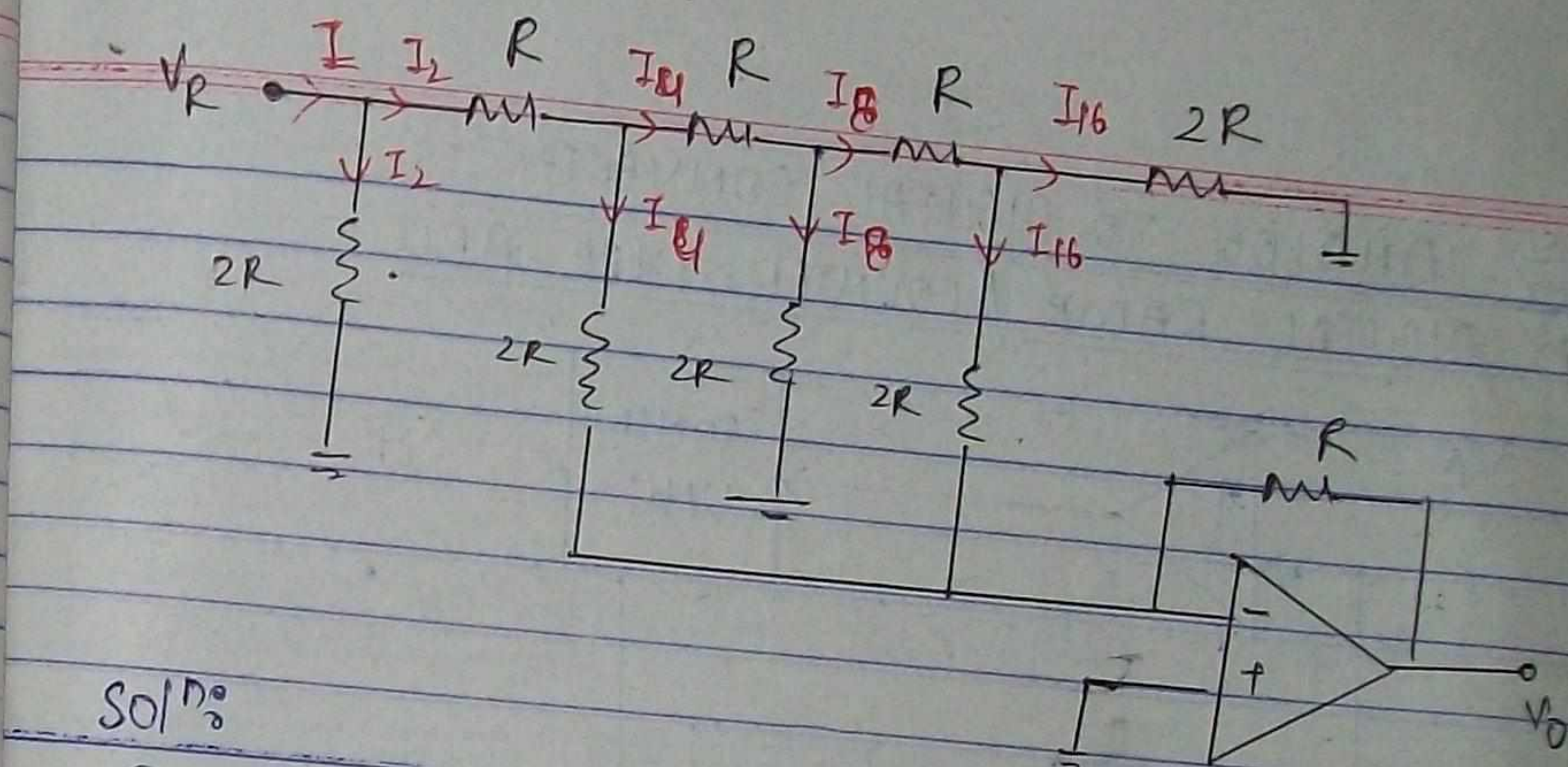
$$V_0 = \left[\frac{-R_f}{3R} \right] \left[\frac{V_R}{2^3} \right] [4b_2 + 2b_1 + b_0]$$

$$V_0 = \frac{V_R}{2^N} \left[\sum_{i=0}^{N-1} 2^i b_i \right] \times \left[\frac{-R_f}{3R} \right]$$

Ques. Given $V_R = 10V$ $R_{eq} = 10k\Omega$

① current $I = ?$

② voltage $V_0 = ?$



Solⁿ:

$$① I = \frac{V_R}{R_{eq}}$$

$$I = \frac{10}{10k} = 1mA$$

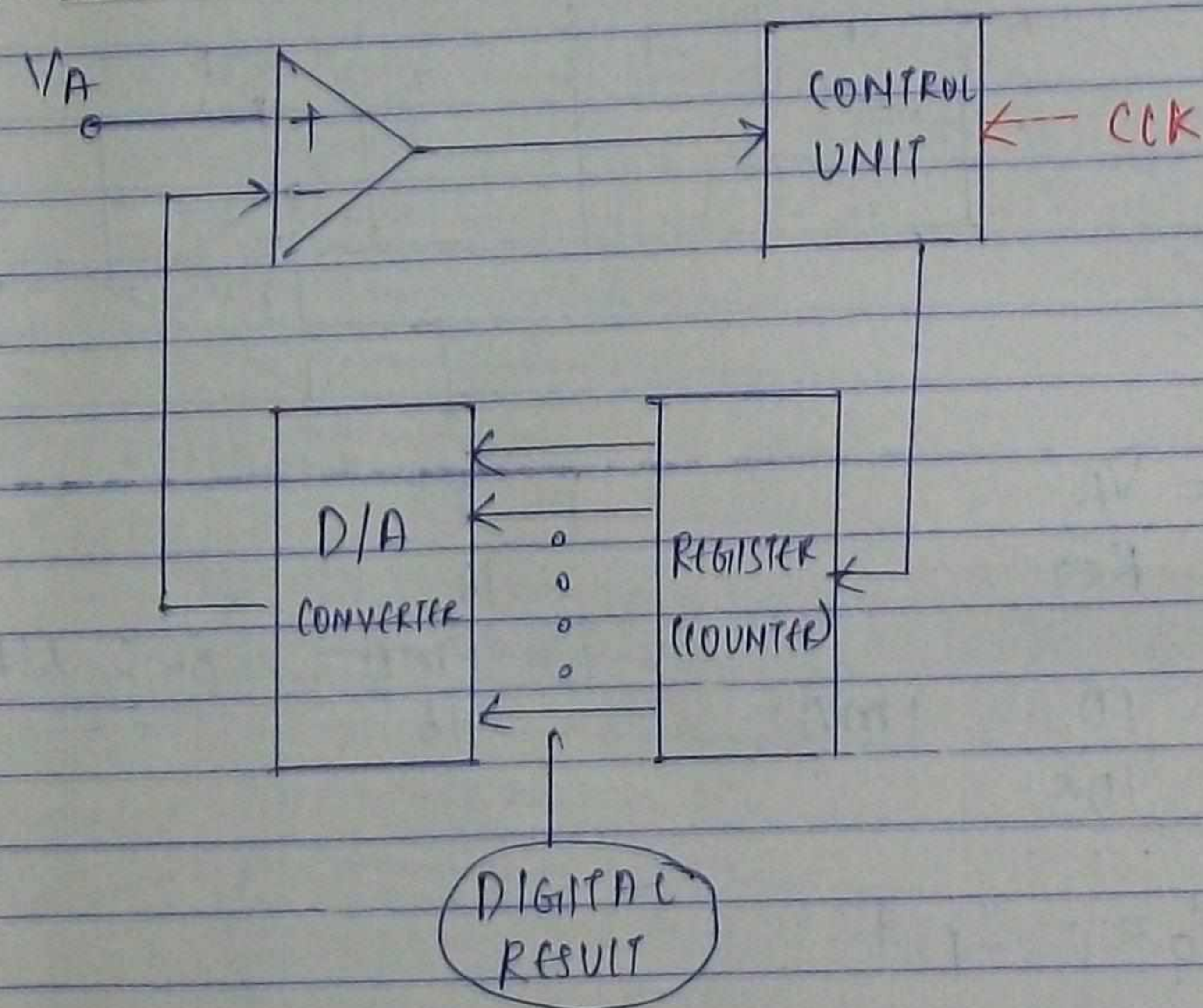
$$I = I$$

$$I = \frac{1mA}{16} = 62.5 \mu A$$

$$② V_0 = -I_f \cdot R$$

$$= - \left(\frac{I}{4} + \frac{I}{16} \right) = -3.125V$$

⇒ ANALOG TO DIGITAL CONVERTER -
 (1) DIGITAL RAMP / COUNTER TYPE ADC -



- Digital Ramp type ADC also referred as counter type ADC and the conversion time t_{AC} is given by:

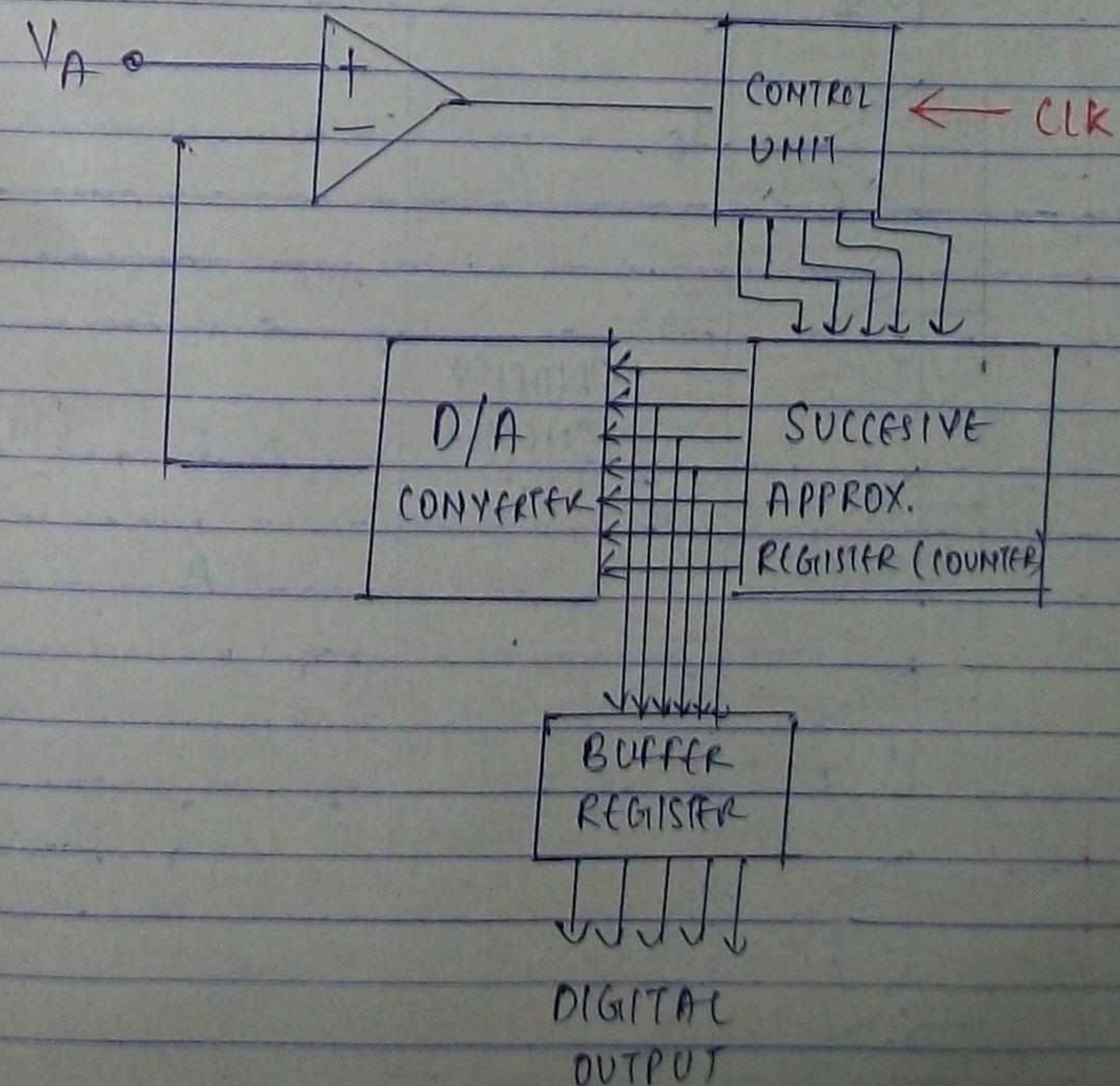
$$t_c = (2^N - 1) \text{ CLK CYCLES}$$

N = No. of bits

DISADVANTAGES -

- It has large conversion time.
- If its resolution is improved then conversion time also increases.
- It is used only for low speed application.

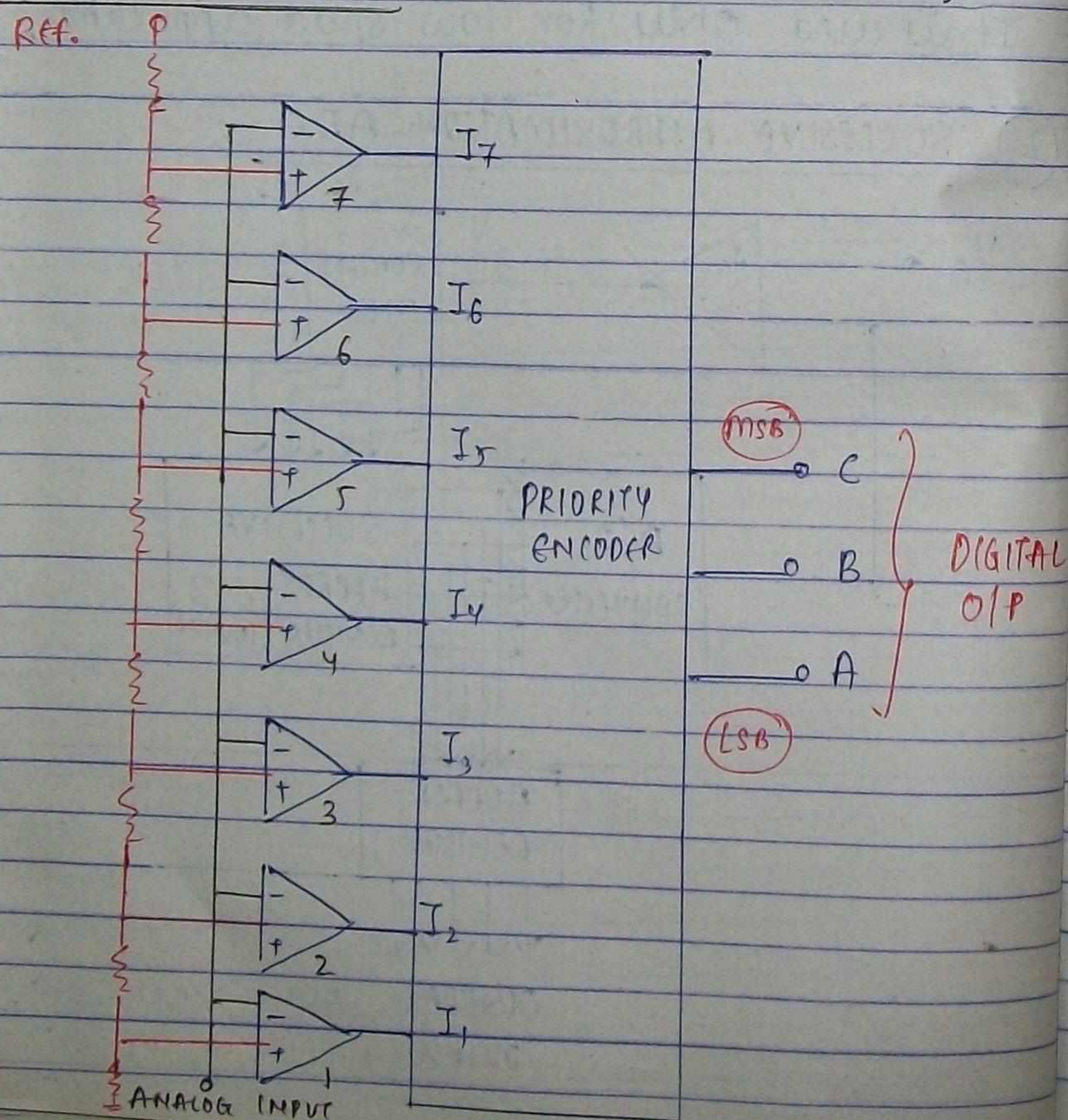
(2) SUCCESSIVE APPROXIMATION ADC -



- The processing of each bit takes 1 clk cycle so, the total conversion time for n-bit SADC (SAC) will be:

$$t_c = N \times 1 \text{ CLK CYCLES}$$

⇒ FLASH TYPE ADC - (11th COMPARATOR TYPE)



- It is also called flash type or parallel comparator type ADC. or simultaneous ADC.

- flash type ADC is the highest speed ADC.

- for n-bit converter the total no. of flash or 11th comparator equal to

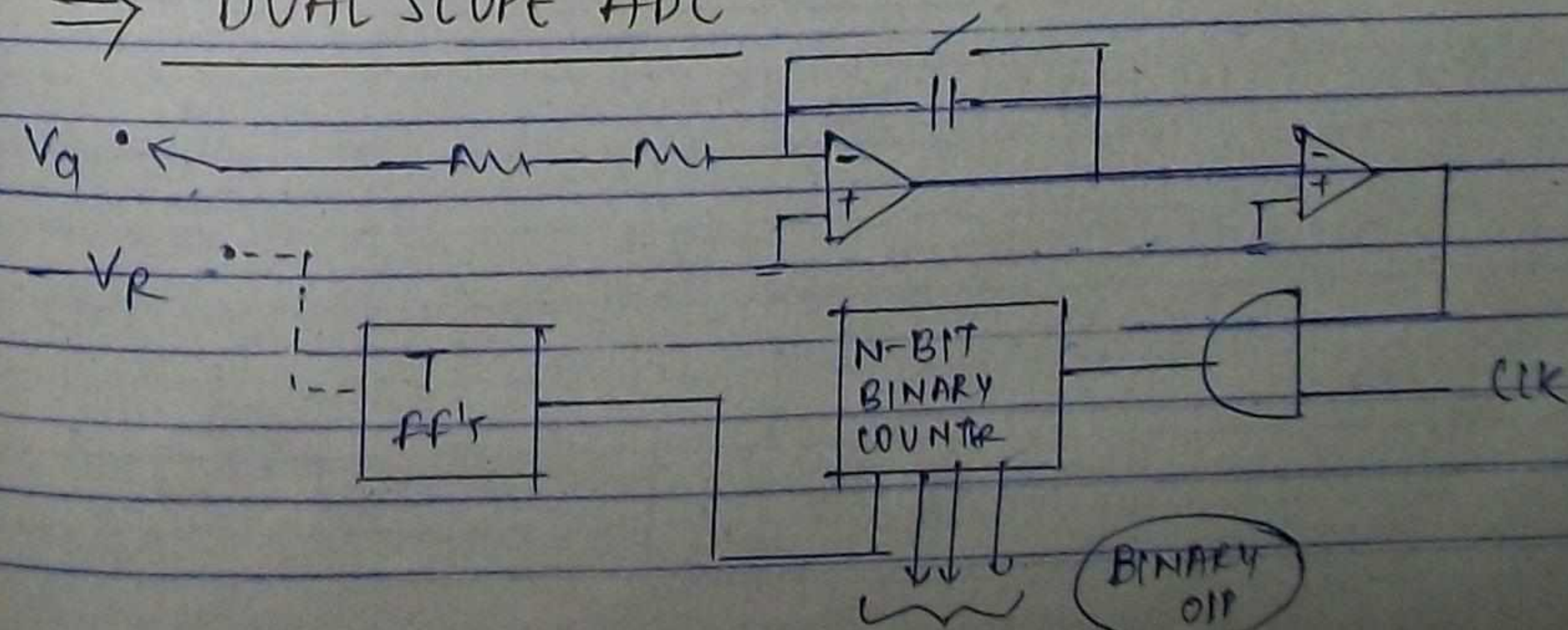
$$\text{NO. OF COMPARATOR} = 2^N - 1$$

- Conversion time: This ADC uses no clk signal. Only one clock pulse is sufficient

DISADVANTAGE -

- It requires much more circuitry.

⇒ DUAL SLOPE ADC -



Conversion time t_c :-

$$t_c = 2^{N+1} \text{ CLK CYCLES}$$

- It is the slowest ADC. ($T=RC \uparrow \downarrow$)
- It is most accurate, and used in digital voltmeters.

NUMBER SYSTEM

⇒ NUMBER REPRESENTATION -

- ① Sign magnitude method
- ② 1's complement method
- ③ 2's complement method.

① SIGN - MAGNITUDE METHOD -

51 → 110011

+51 →

0	110011
---	--------

↓
SIGN
BIT

SAME FOR
1'S AND 2'S
COMPLEMENT
METHOD

-51 →

1	110011
---	--------

SIGN
MAGNITUDE

② 1's COMPLEMENT METHOD -

-51 , 51 → 110011

1's of 51 → 001100

-51 →

1	001100
---	--------

② 2's complement -

$$51 \rightarrow 110011$$

$$2's \text{ of } 51 \rightarrow 001101$$

$$-51 \rightarrow \boxed{1 \mid 001101}$$

⇒ RANGE OF NO. -

① SIGN MAGNITUDE -

$$-(2^{N-1}-1) \text{ TO } +(2^{N-1}-1)$$

② 1's complement -

$$-(2^{N-1}-1) \text{ TO } +(2^{N-1}-1)$$

③ 2's complement -

$$-2^{N-1} \text{ TO } +(2^{N-1}-1)$$

where,

N = no. of bits.

⇒ OVER FLOW CONDITION -

$$+5 \rightarrow 0101$$

$$+4 \rightarrow 0100$$

$$1001 \rightarrow (-1)$$

$$\begin{array}{r} 1001 \\ \downarrow \end{array}$$

Overflow

- In signed arithmetic operation overflow may occur during the addition of same signed number.

⇒ METHODS TO DETECT OVERFLOW CONDⁿ -

① METHOD -

$$\begin{array}{r} \overline{x} \overline{y} z + x y z \\ 0101 \\ 0100 \\ 1001 \end{array} = \overline{0} \overline{0} 1 + 00\overline{1}$$

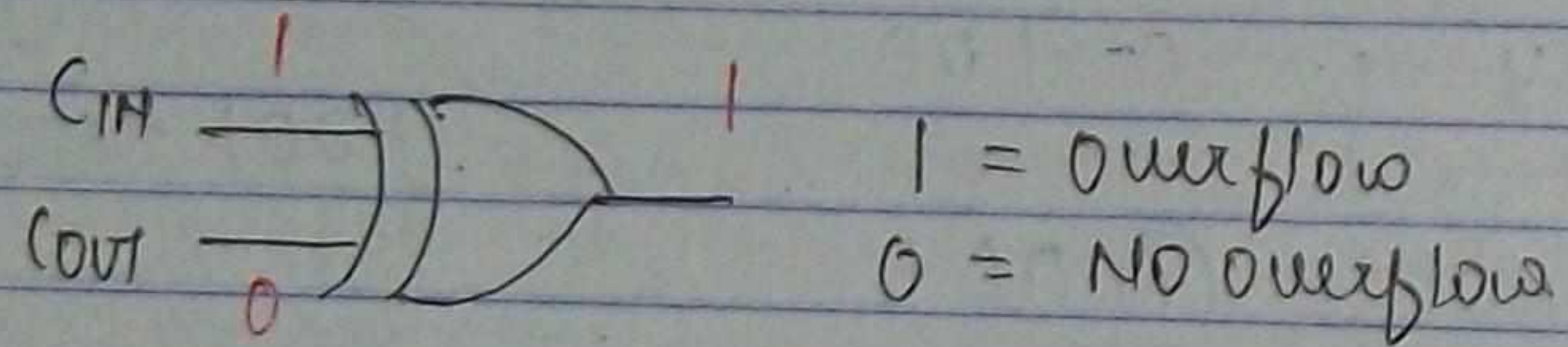
$$= 1 + 0 = 1 \rightarrow \text{overflow.}$$

Let x and y are signed bit of given no. and z is the resultant signed bit then the condⁿ for overflow is

$$\overline{x} \overline{y} z + x y z = 1 \text{ overflow}$$

$$= 0 \text{ no overflow}$$

② METHOD -



C_{IN} = carry into the MSB

C_{OUT} = carry from the MSB

~~***~~
TABULAR FORM OF
THEORY BOOK

P-164

MICROPROCESSORS

1/12/11

- Introduction
- Architecture
- Program

INTRODUCTION

DEFINITION -

It is an electronic integrated chip that has computing & decision making capability. It is an electronic chip that fetch instruction from memory execute them and provide result.

NOTE -

A μp cannot perform any task on its own

$$\mu p = \left[\begin{array}{c} \text{Hardware} \\ \mu p \end{array} + \begin{array}{c} \text{(Software)} \\ \text{Rom} \end{array} \right]$$

ROM -

System information software is stored in rom and come in picture at the time of power switched on.

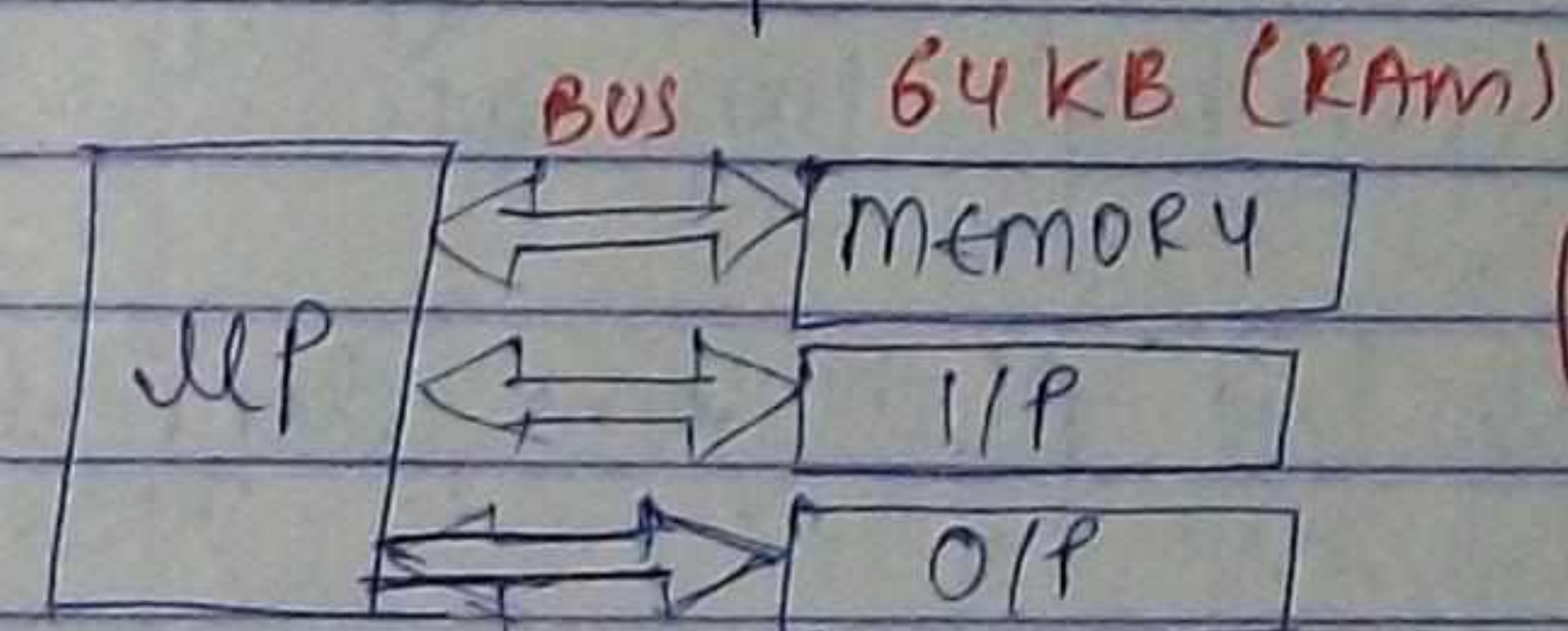
RAM - (MAIN memory or memory)

Program always feed in main memory or RAM

data $\rightarrow$ 8 bits inst
add $\rightarrow$ 16 bit

ALL COMPUTER -

If all task of CPU is performed by μp , then such type of computer is known as all computer.



$$\text{all computer} = \mu p + \text{memory} + \text{I/P - O/P device}$$

NOTE - all computer on single chip is known as microcontroller.

BUS - It is group of (parallel combination) of metal wires that is used for interfacing b/w different devices.

ASIC (APPLⁿ SPECIFIC INTEGRATION CHIP) -

Microcontroller is the example of ~~hardware~~ ASIC design.

MACHINE LANGUAGE -

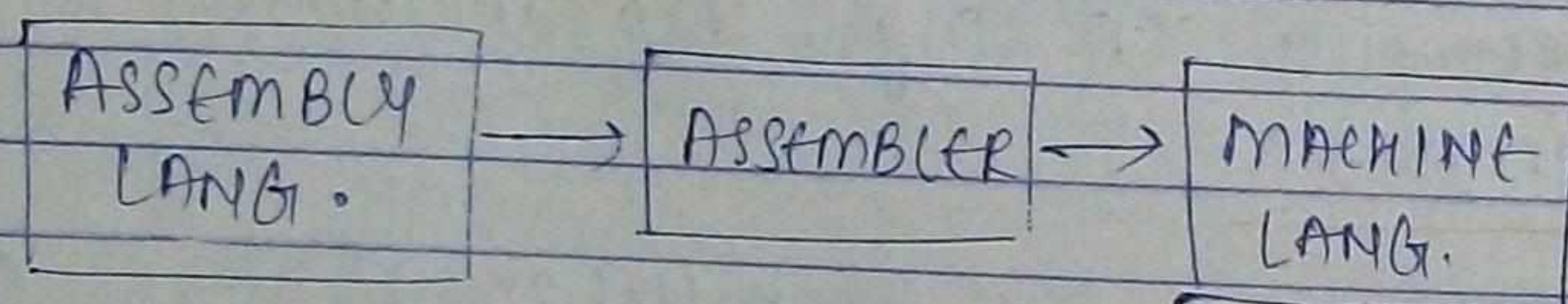
If command or instructions are written in binary format then such type of language is known as machine language.

NOTE - machine language is platform dependent
machine code - - - -

ASSEMBLY LANGUAGE-

If binary pattern is replaced by english like words that is known as 'Mnemonics', then such type of language is known as assembly language.

Assembly language is platform dependent language.



Assembler -

Assembler is nothing but a software that convert assembly lang. to machine lang.

If this task is performed manually then it is known as hand assembly.

LOW LEVEL LANG.-

All platform dependent languages are known as low level lang.

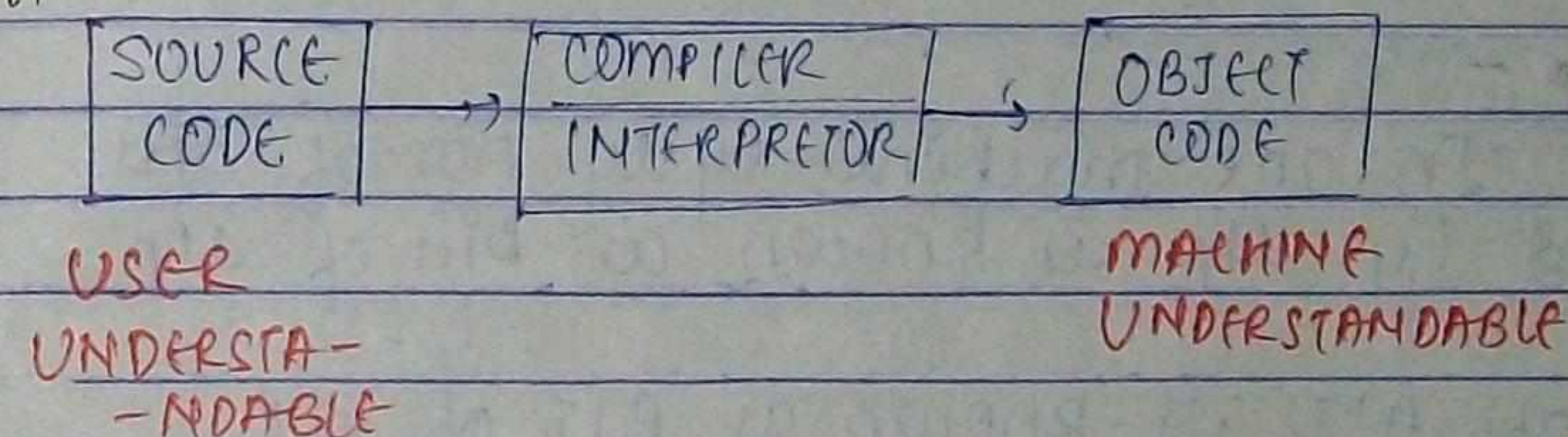
ex - assembly lang. and machine lang.

HIGH LEVEL LANG.-

All platform independent lang.

ex - C, C++, Java etc.

#



NOTE -

Compiler read whole program at once from source code produce its object code that executed by processor.
Compiler is a software.

ex - C, C++

INTERPRETOR -

It reads one instruction at once from source code produce its object code, that is executed by step before reading next instruction from source-code.

Interpretor is a software.

NOTE -

- ① 8085A μp is commonly known as 8085 μp
- ② 8085 μp based upon NMOS technology
- ③ It is 8 bit μp .

BIT OF μp -

In one machine cycle no. of bits executed by μp is known as bits of μp .

- Size of ALU is known as BIT of μp .

④ <u>μp</u>	<u>BIT OF μp</u>	<u>Tech</u>
4004	4 bit	PMOS
8008	8 bit	NMOS
8080	8 bit	NMOS
8085	8 bit	NMOS
8086	16-bit	HMOS
8088	8/16 bit	(high density channel-mos)
(Improved version of 8086)		HMOS

8088 is externally 8 bit, but internally 16 bit μp .

80186	16 bit	CMOS tech
80286	16 bit	
80386	32-bit	
80486	32-bit	
80586	64-bit	
<u>PENTIUM</u>		
Pentium Pro	64 bit	
P-II		
P-III	64 bit	
P-IV		
Dual core		
Core 2 Duo		
i3		
i5		2nd generation.
i7		