NUMBER SYSTEM

* Double Dibble Method

00
01
10
11
100
101
110
111

Base → Radix → (0 to r-1)

* Decimal Number System

- In this number system, 10 digits are used to form a number. This digits are 0 to 9. As there are 10 digits the base of the decimal number is 10. The base is also called as radix.

- Number system is a positional system in which the position of a digit specifies the weighted significance attached to it.

Ex. i] 2432

$2 \times 1000 = 2 \times 10^3$
$4 \times 100 = 4 \times 10^2$
$3 \times 10 = 3 \times 10^1$
$2 \times 1 = 2 \times 10^0$

ii] 2432.55

$2 \times 1000 = 2 \times 10^3$
$4 \times 100 = 2 \times 10^2$
$3 \times 10 = 3 \times 10^1$
$2 \times 1 = 2 \times 10^0$
$9 \times 0.1 = 5 \times 10^{-1}$
$5 \times 0.01 = 5 \times 10^{-2}$

* Binary Number System

It has two digits i.e. 0 and 1. Therefore the base is 2.

Ex. i] $(101)_2$

$1 \times 2^2 = 4$
$0 \times 2^1 = 0$
$1 \times 2^0 = 1$
$(5)_{10}$

ii] $(1010110)_2$

$1 \times 2^6 = 64$
$0 \times 2^5 = 0$
$1 \times 2^4 = 16$
$0 \times 2^3 = 0$
$1 \times 2^2 = 4$
$1 \times 2^1 = 2$
$0 \times 2^0 = 0$
$(86)_{10}$

iii] $(101.10)_2$

$1 \times 2^2 = 4$
$0 \times 2^1 = 0$
$1 \times 2^0 = 1$
$1 \times 2^{-1} = 0.5$
$0 \times 2^{-2} = 0$
$(5.5)_{10}$

0 and 1 are called as bit
4 bits are called nibble
8 bits are called byte
16 bits are called 1 word
32 bits are called 1 doubleword.

1024 bytes = 1kB = 2^{10}

2KB = 2^{11}
64KB = 2^{16}
1 MB = 1024 KB = 2^{20}
1 GB = 1024MB = 2^{30}
1 TB = 1024GB = 2^{40}

* Octal Number System

It contains 8 digits i.e. 0 to 7. Every number is formed using this digits. 8 and 9 is not used in this system. Base of octal number system is 8.

0	10	20	- - - .	70
1	11	21		⋮
2	12	⋮		⋮
3	13	⋮		⋮
4	14	⋮		⋮
5	15	⋮		⋮
6	16	⋮		⋮
7	17	27		77

Eg. 1] $(357)_8$

$3 \times 8^2 = 192$

$5 \times 8^1 = 40$

$7 \times 8^0 = 7$

$(239)_{10}$

* Hexadecimal Number System

Base is 16. Total digits are also 16 i.e. 0 to 9 and A, B, C, D, E, F.

0	10	. . . . . .	F0
1	11		F1
2	12		F2
3	13		
4	14		
5	15		
6	16		
7	17		
8	18		
9	19		
A	1A		
B	1B		
C	1C		
D	1D		
E	1E		
F	1F		FF

Ex. 1] $(2A1)_{16}$

$2 \times 16^2 = 512$

$10 \times 16^1 = 160$

$1 \times 16^0 = 1$

$(673)_{10}$

Decimal	Hexademial	Octal	Binary
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	7	7	0111
8	8	10	1000
9	9	11	1001
10	A	12	1010
11	B	13	1011
12	C	14	1100
13	D	15	1101
14	E	16	1110
15	F	17	1111

3 bit number.

	7	6	5	4	3	2	1	0
unsigned →	1	0	1	0	1	1	0	1

most significant Bit MSB

least Significand Bit LSB

	7	6	5	4	3	2	1	0
signed →								

Sign Bit — MSB — $\underbrace{\quad\quad\quad}_{\text{No.}}$ — LSB

Sign Bit
-ve = 1
+ve = 0

3. Converting Decimal number to binary number

Eg. i] $(25)_{10}$

	25	
2	12	1
2	6	0
2	3	0
2	1	1
2	0	1

← LSB

← MSB

ii] $(173)_{10}$

	173	
2	86	1
2	43	0
2	21	1
2	10	1
2	5	0
2	2	1
2	1	0
2	0	1

$(1011\ 0101)_2$

iii] $(25.25)_{10}$

	25	
2	12	1
2	6	0
2	3	0
2	1	1
2	0	1

$0.25 \times 2 = 0.50$ 0

$0.50 \times 2 = 1.00$ 1

$(11001.0100)_2$

↳ Binary point

iv] $(0.765)_{10}$

$0.765 \times 2 = 1.53$ 1
$0.53 \times 2 = 1.06$ 1
$0.06 \times 2 = 0.12$ 0
$0.12 \times 2 = 0.24$ 0
$0.24 \times 2 = 0.48$ 0
$0.48 \times 2 = 0.96$ 0
$0.96 \times 2 = 1.92$ 1
$0.92 \times 2 = 1.84$ 1

$(0.1100\ 0011)_2$

2. Converting Decimal number to octal number.

i] $(80)_{10}$

	80	
8	10	0
8	1	2
8	0	1

$(120)_8$

ii] $(100)_{10}$

	100	
8	12	4
8	1	4
8	0	1

$(144)_8$

iii] $(100.88)_{10}$

$0.88 \times 8 = 7.04$ 7
$0.04 \times 8 = 0.32$ 0
$0.32 \times 8 = 2.56$ 2
$0.56 \times 8 = 4.48$ 4

$(144.7024)_8$

3. Converting Decimal number to Hexadecimal number.

i] $(100)_{10}$

	100	
16	6	4
16	0	6

$(64)_{16}$

ii] $(175)_{10}$

	175	
16	10	15 (F)
16	0	10 (A)

$(AF)_{16}$

iii] $(175.99)_{10}$

$0.99 \times 16 = 15.84$ 15 (F)
$0.84 \times 16 = 13.44$ 13 (D)
$0.44 \times 16 = 7.04$ 7
$0.04 \times 16 = 0.64$ 0
$0.64 \times 16 = 10.24$ 10 (A)
$0.24 \times 16 = 3.84$ 3

(AF. FD70A3).

4. Converting Octal Numbers to Decimal Number

e] $(88)_8$

$8 \times 8^1 + 8 \times 8^0$

$= 64 + 8$

$= (72)_8$

ex] $(127)_8 = (?)_{16}$

M-1] $1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0$

$= 64 + 16 + 7$

$= (87)_{10}$

	87	
16	5	7
16	0	5

M-2] $(127)_8$

~~sol~~ write in binary by taking 3 bit equivalent (For octant)

~~ex~~ $(001\ 010\ 111)_2$

For hexadecimal, take 4 bit equivalent

$(001\ 010\ 111)_2$

$= (57)_{16}$

2] $(756)_8 = (111\ 101\ 110)_2$

$(756)_{16} = (0111\ 0101\ 0110)_2$

3] $(8AEF)_{16} = (?)_8$

$(8AEF)_{16} = (1000\ 1010\ 1110\ 1111)_2$

$(1000\ 1010\ 1110\ 1111)_2 = (105357)_8$

4] $(7E.2A)_{16} = (?)_8$

$(7E.2A)_{16} = (0111\ 1110\ 0010\ 1010)_2$

$(01\ 111\ 110.\ 001\ 010\ 100)_2 = (176.124)_8$

Note :- For number before decimal point grouping goes from right to left, while for number after decimal point, grouping goes from left to right.

* Digital Codes

1] BCD (Binary Coded Decimal)

- It is representation of decimal number in binary form. In BCD, each individual decimal digits is converted into a 4 bit binary equivalent.

- By combining them all, the BCD code is generated.

Ex. 1] Find BCD code of decimal number 358.

$$(358)_{10} = (0011\ 0101\ 1000)_{BCD}$$

In BCD code, only the decimal digits 0-9 are used.

2] Excess Three code

- In excess three code we need to add (0011) to each 4 bit group in binary coded decimal number to get desired excess three equivalent.

- It is an extension to BCD.

Decimal	BCD	ES 3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0010	1001
7	0111	1010
8	1000	1011
9	1001	1100

Ex. 4] Find the excess three code for $(45)_{10}$

$(45)_{10}$ = (0111 1000)

4] $(89)_{10}$ = (1011 1100)

3] Gray Code

- Gray code system is a binary number system in which every successive pair of number differs in only one bit.

- Gray codes are very useful in the normal sequence of binary number generated by the hardware that may cause errors or ambiguity during the transmission from one number to another

- In finding the gray code. The X-OR operation is useful.

X - OR

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

Ex. i] $(1001)_2$

1 0 0 1

1 ~~0 1 1~~ 1 0

1 0 0 1 ← X-OR operations

MSB as it is → 1 1 0 1

(1101) gray

ii] $(0100)_2$

0 1 0 0

0 1 1 0 (0110) gray

iii] $(0111)_2$

0 1 1 1

0 1 0 0 (0100) gray

iv] $(1010)_2$

1 0 1 0

1 1 1 1 (1111) gray

Decimal	BCD	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101

Eg. i] $(1101)_{gray}$

1 1 0 1

MSB as it is → ↓ ↓ as ↓ ↓ ← x-OR operations

1 0 0 1

$(1001)_2$

$(1010)_{gray}$

1 0 1 0

1 1 0 0

$(1100)_2$.

ii] $(0101)_{gray}$

0 1 0 1

0 1 1 0 $(0110)_2$

4] ASCII Code

- It is the short form of American Standard Code for Information Interchange.

- It is the standard that standard that assigns letters, numbers and other characters an 8 bit code.

- ASCII Table is divided into 3 different sections :-

i] Non printable system code from 0 to 31.

ii] Lower ASCII between 32 to 127. It works on 7 bit ~~co~~ character tables.

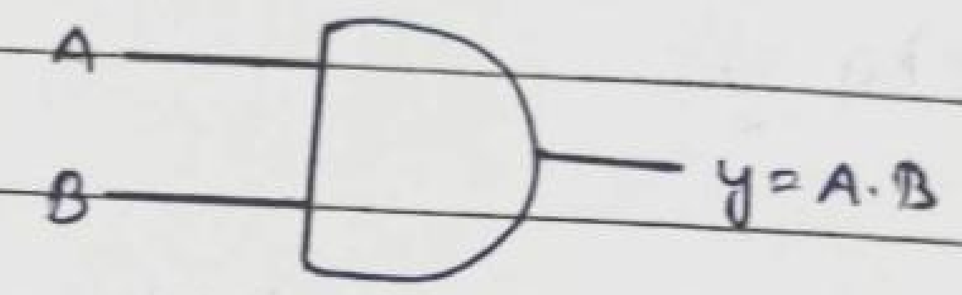
iii] Higher ASCII from 128 to 255. This portion is programmable. Characters are based on language of your operating system or program you are using.

Character	ASCII
0	48
1	49
2	50
⋮	⋮
9	57
A	65
B	66
⋮	⋮
Z	90
a	97
b	98
⋮	⋮
z	122

* LOGIC GATES

1] AND

$y = A \cdot B$



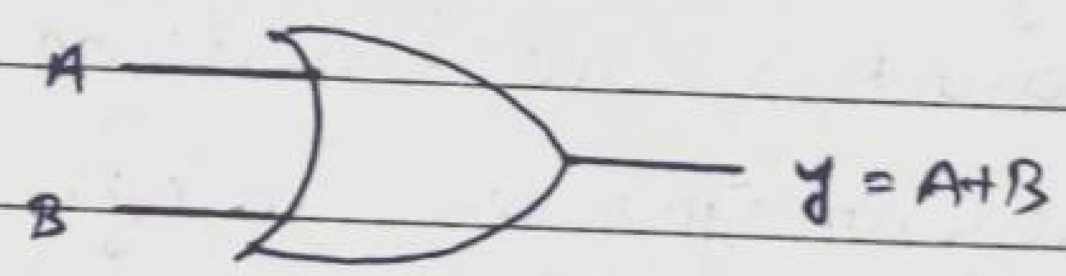
Truth Table

A	B	y
0	0	0
0	1	0
1	0	0
1	1	1

OUTPUT is high only when all INPUT is high

2] OR

$y = A + B$



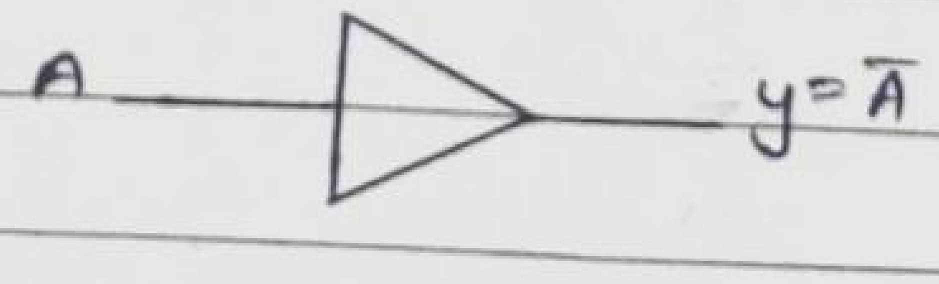
Truth Table

A	B	y
0	0	0
0	1	1
1	0	1
1	1	1

OUTPUT is low only when all INPUT is low.

3] NOT

$y = \bar{y}$



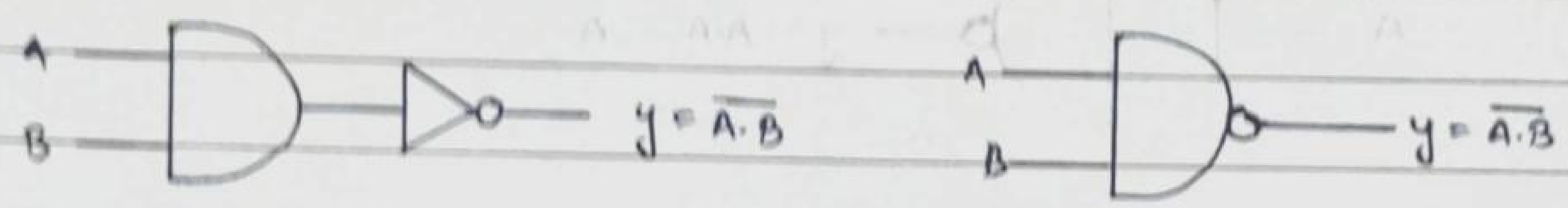
Truth Table

A	y
0	1
1	0

OUTPUT is the compliment of the INPUT

* UNIVERSAL GATES

1] NAND

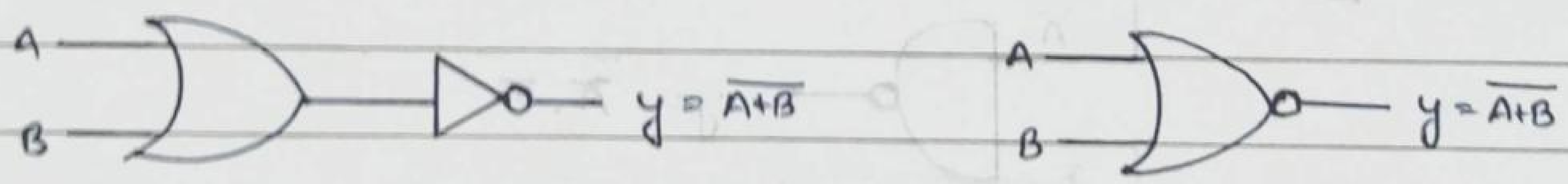


Truth Table

A	B	y
0	0	1
0	1	1
1	0	1
1	1	0

OUTPUT is low when all INPUT are high

2] NOR



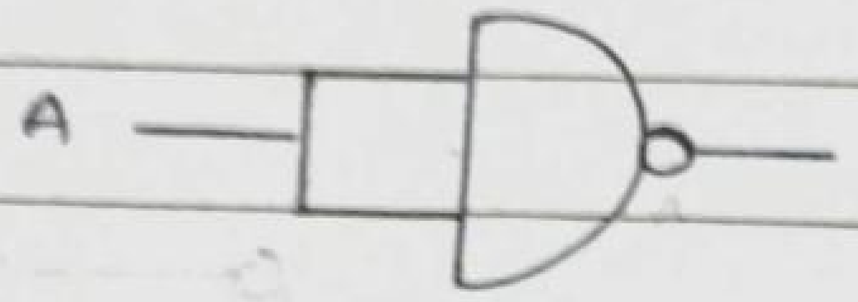
Truth Table

A	B	y
0	0	1
0	1	0
1	0	0
1	1	0

OUTPUT is high only when INPUT is low

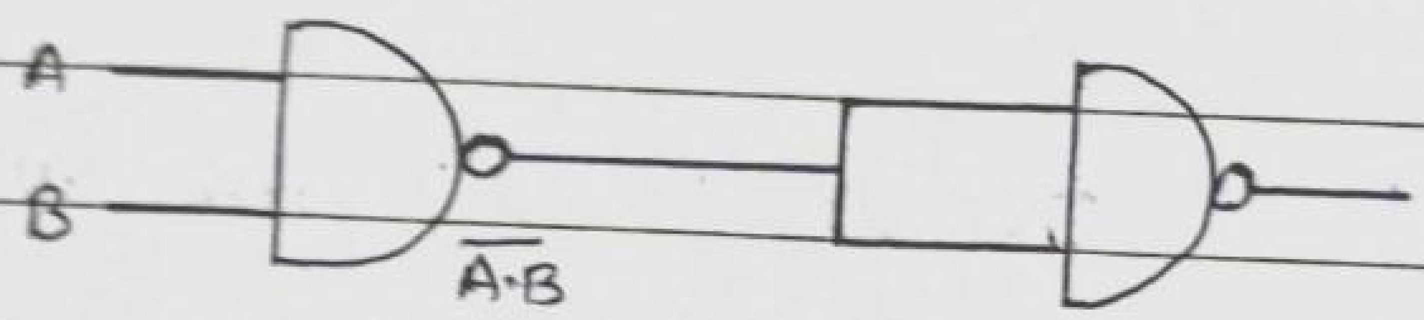
- NAND and NOR Gates called as universal building blocks because the basic gates AND, OR, NOT can be designed using NAND and NOR.

1] Implementation of NOT using NAND



$y = \overline{A \cdot A} = \overline{A}$

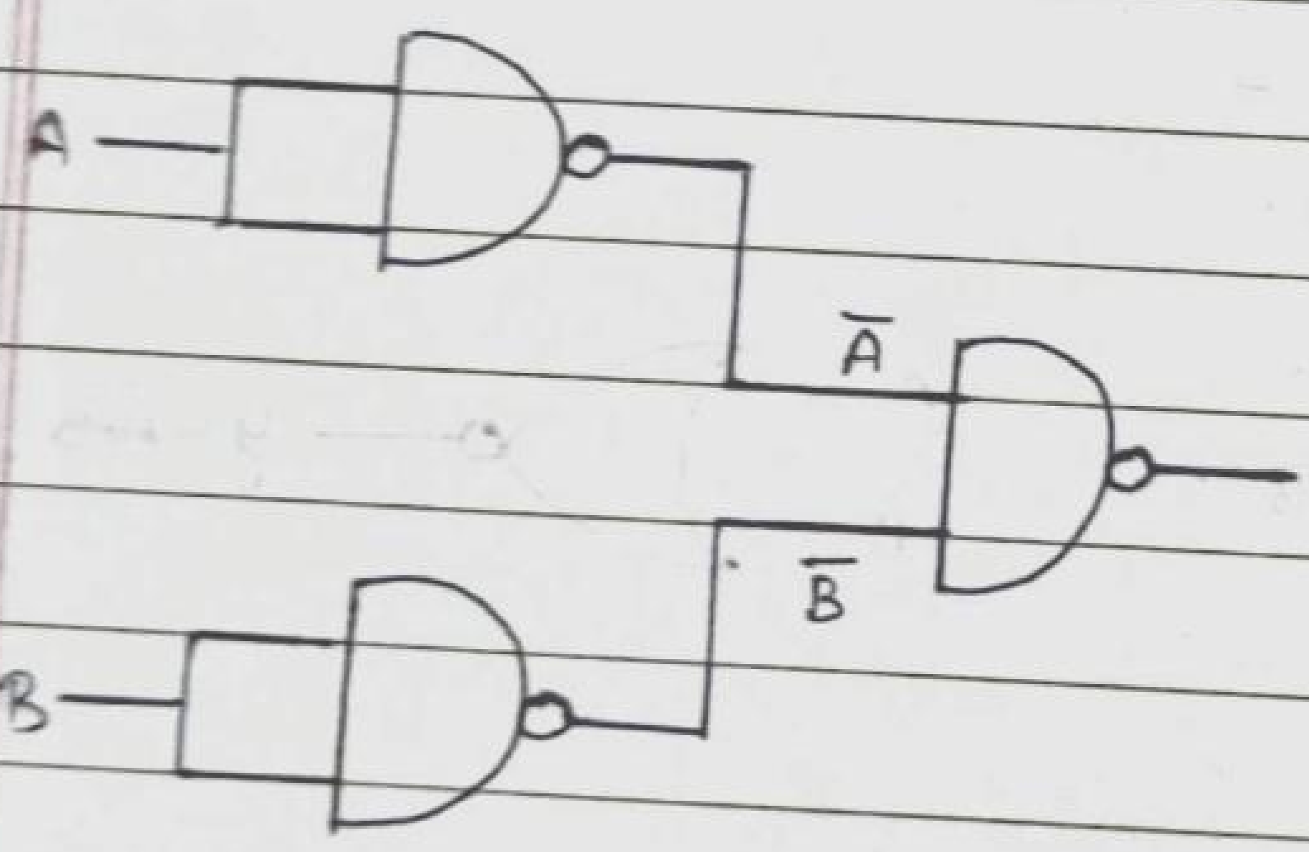
2] Implementation of AND using NAND



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

$= A \cdot B$

3] Implementation of OR using NAND

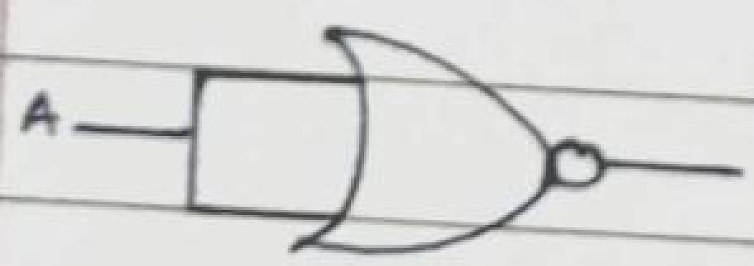


$y = \overline{\overline{A} \cdot \overline{B}}$

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

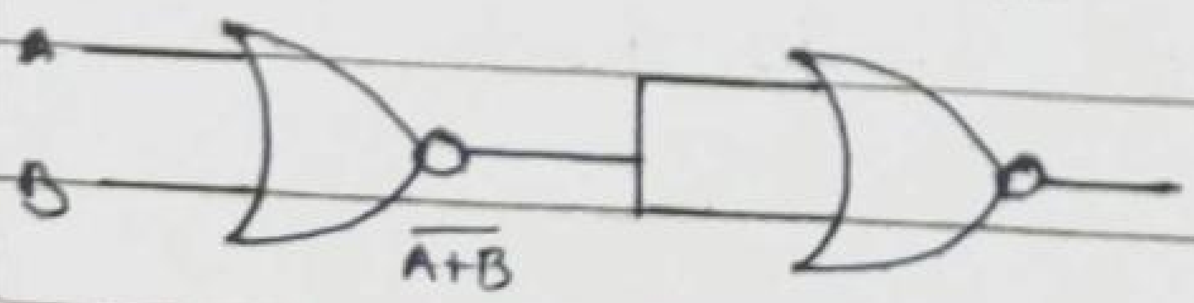
$= A + B$

4] Implementation of NOT using NOR



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

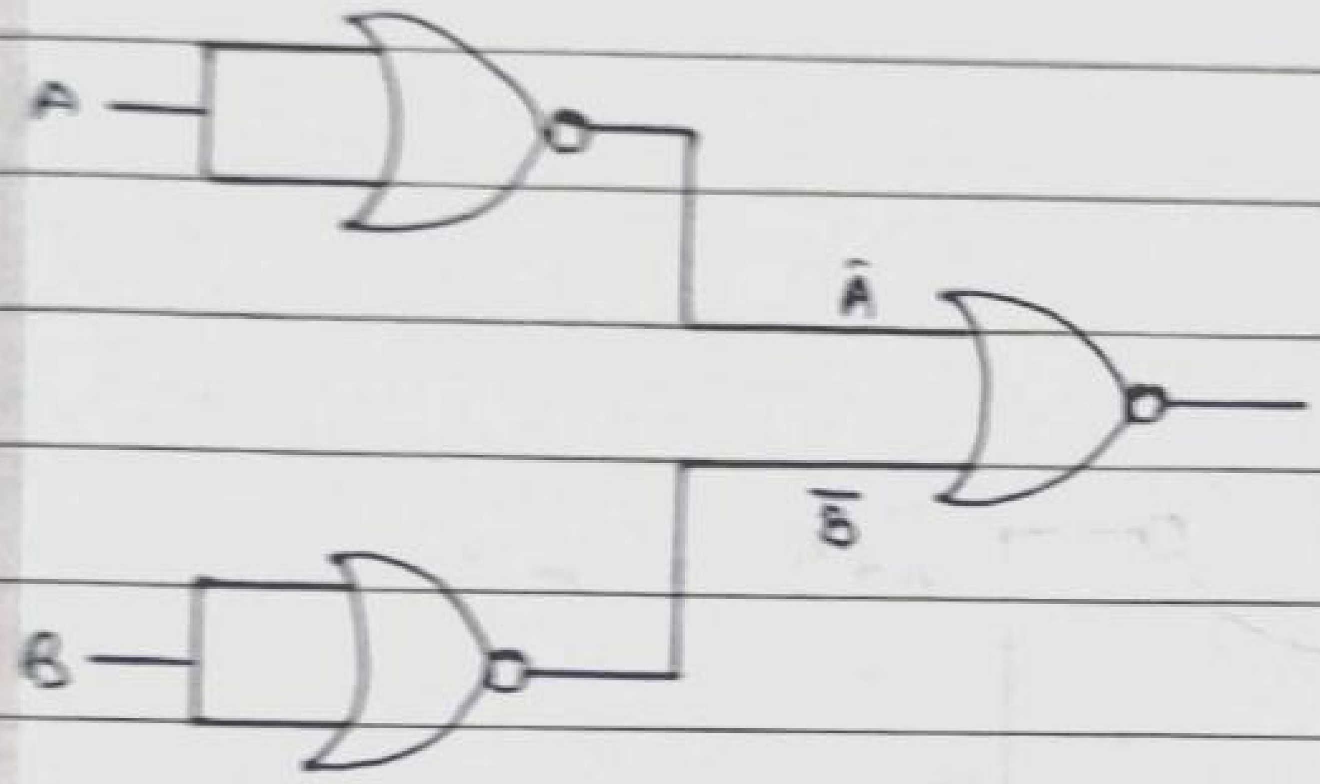
5] Implementation of OR using NOR



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

$= A + B$

6] Implementation of AND using NOR

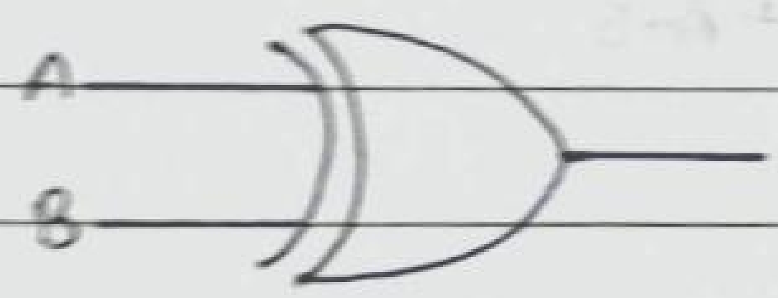


$y = \overline{\bar{A}+\bar{B}}$

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

$= A \cdot B$

* Ex-OR Gate



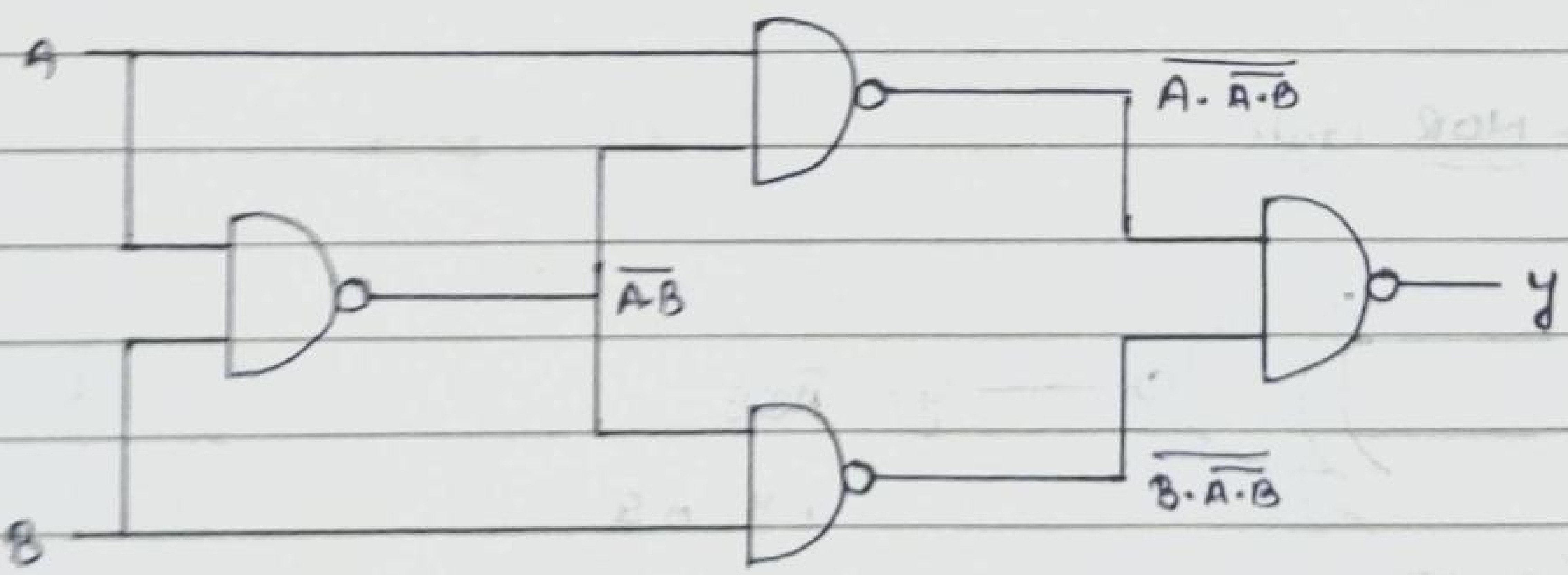
$y = A \oplus B$

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

Truth Table

A	B	y
0	0	0
0	1	1
1	0	1
1	1	0

1] Ex-OR using NAND



$y = \overline{(\overline{A \cdot \overline{A \cdot B}}) \cdot (\overline{B \cdot \overline{A \cdot B}})}$

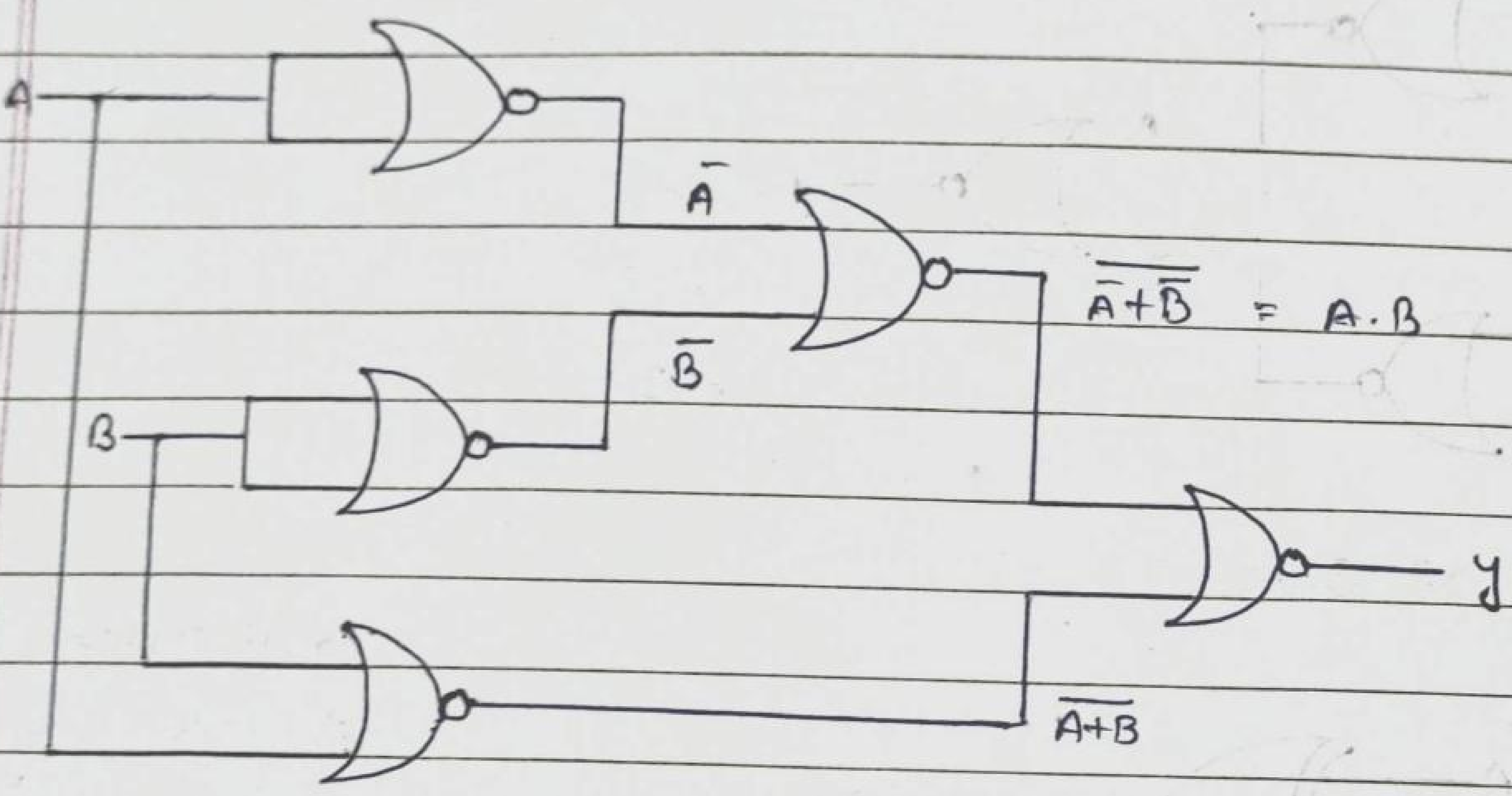
$= (\overline{\overline{A \cdot \overline{A \cdot B}}}) + (\overline{\overline{B \cdot \overline{A \cdot B}}})$

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

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

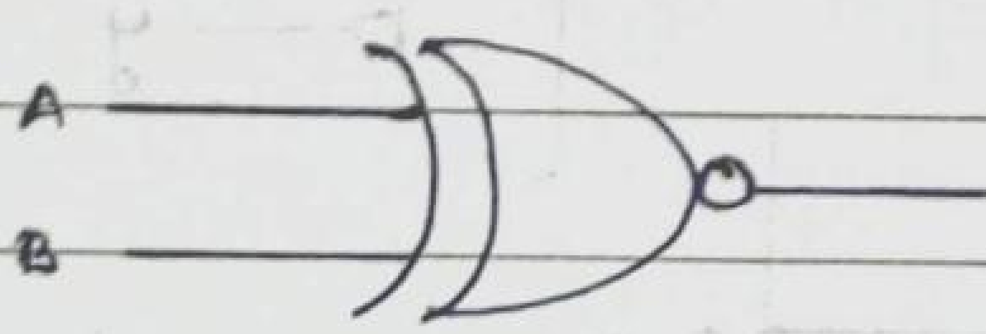
$= A \cdot \bar{B} + \bar{A} \cdot B$

2] EX-OR using NOR



$$y = \overline{(A\cdot B) + (\overline{A+B})}$$
$$= \overline{(A\cdot B)}\cdot(\overline{\overline{A+B}})$$
$$= (\bar{A}+\bar{B})\cdot(A+B)$$
$$= \bar{A}\cdot\bar{A} + \bar{A}\cdot B + \bar{B}\cdot A + \bar{B}\cdot B$$
$$= \bar{A}\cdot B + \bar{B}\cdot A$$

* EX-NOR Gate

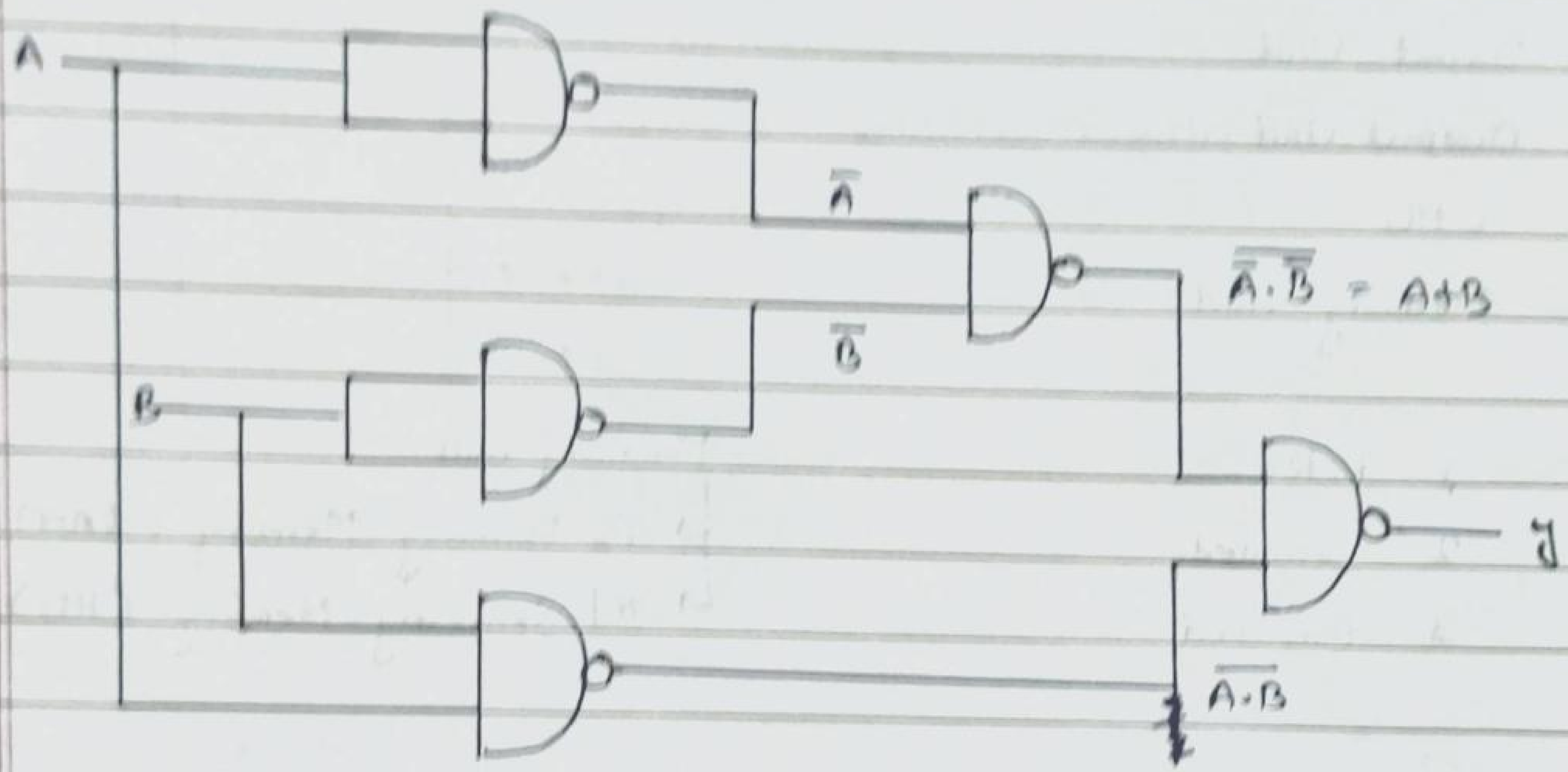


$$y = \overline{A\oplus B}$$
$$= \bar{A}\cdot B + A\cdot B$$

Truth Table

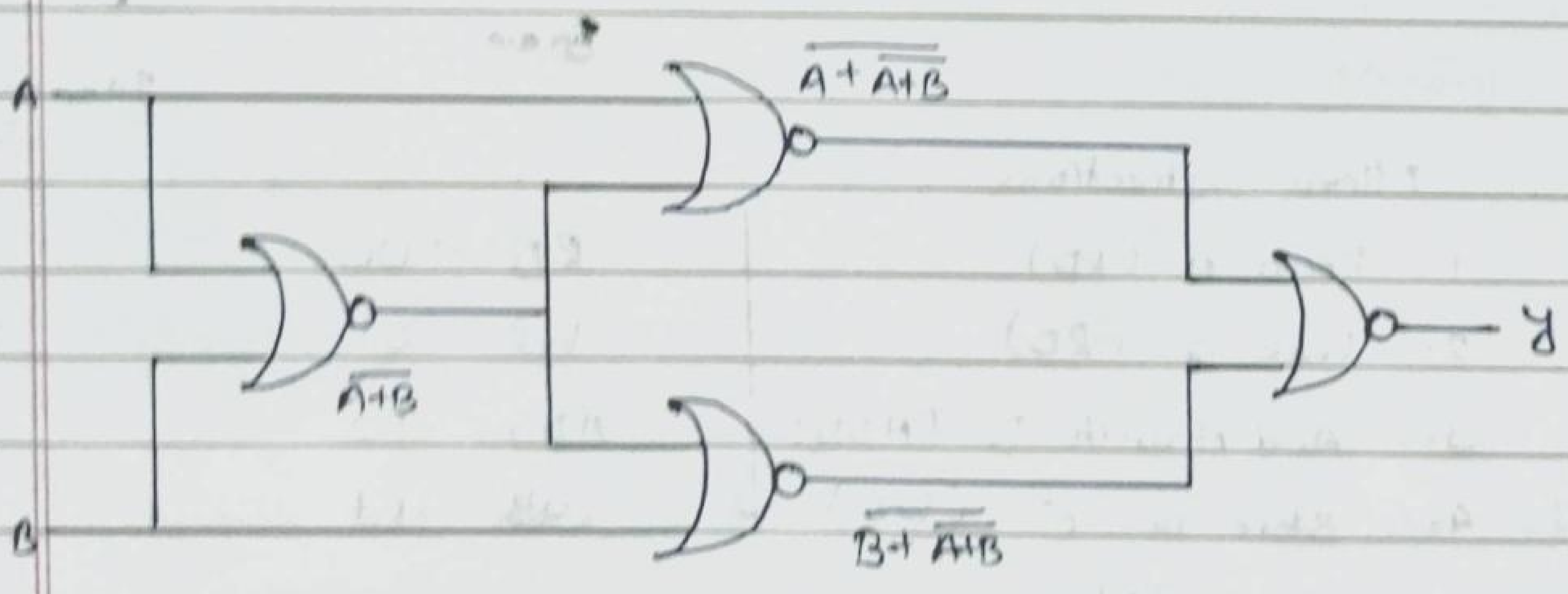
A	B	y
0	0	1
0	1	0
1	0	0
1	1	1

1] EX-NOR using NAND



$$y = \overline{(A+B)\cdot(\overline{A\cdot B})}$$
$$= \overline{(\overline{\overline{A}\cdot\overline{B}})\cdot(\overline{A\cdot B})}$$
$$= (\overline{\overline{\overline{A}\cdot\overline{B}}}) + (\overline{\overline{A\cdot B}})$$
$$y = \overline{A}\cdot\overline{B} + A\cdot B$$

2] EX-NOR using NOR



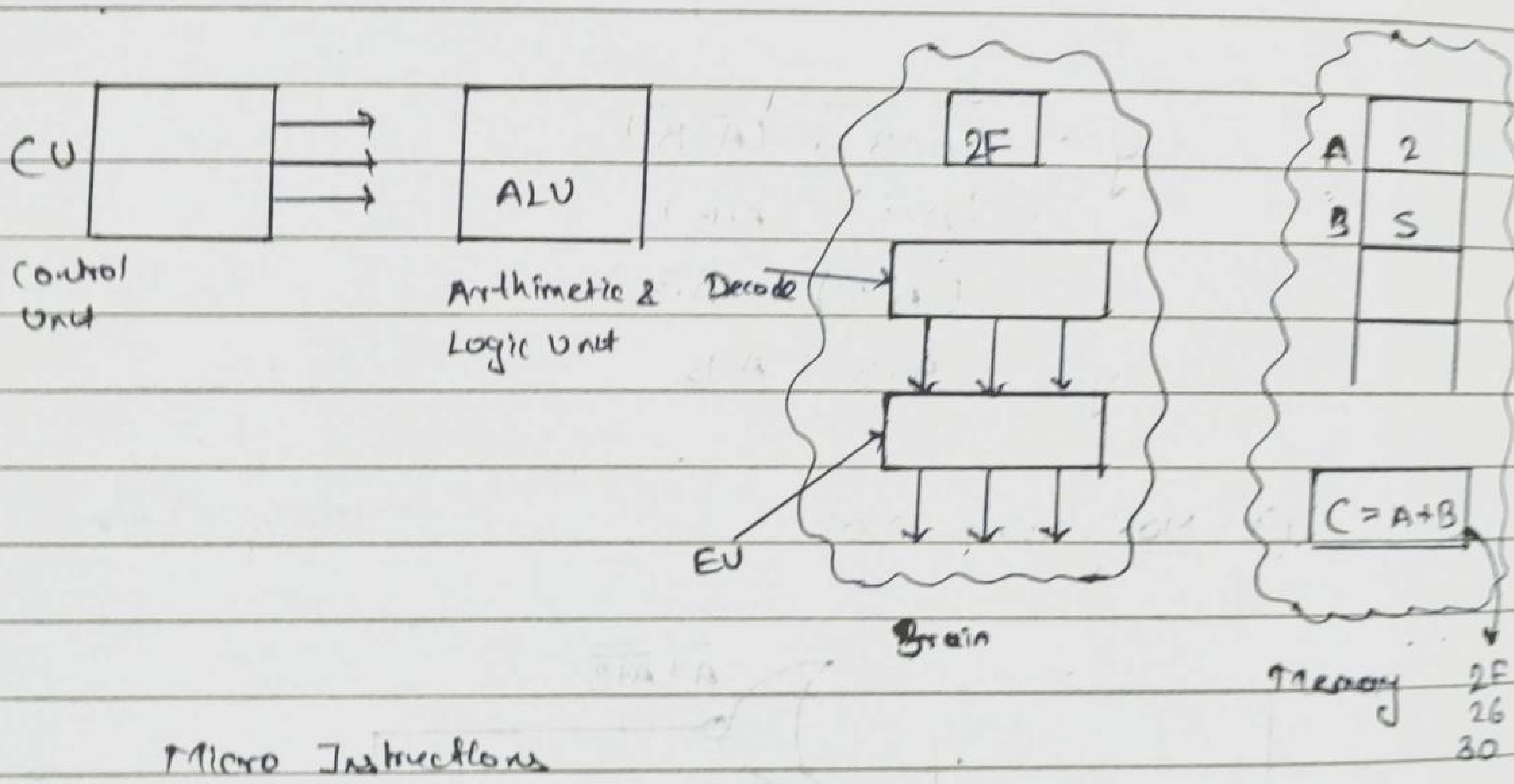
$$y = \overline{(\overline{A+\overline{A+B}}) + (\overline{B+\overline{A+B}})}$$
$$= (A+\overline{A+B})\cdot(B+\overline{A+B})$$
$$= (A+\overline{A}\cdot\overline{B})\cdot(B+\overline{A}\cdot\overline{B})$$
$$= A\cdot B + A\cdot\overline{A}\cdot\overline{B} + \overline{A}\cdot\overline{B}\cdot B + \overline{A}\cdot\overline{B}\cdot\overline{A}\cdot\overline{B}$$
$$= A\cdot B + \overline{A}\cdot\overline{B}$$

* Basic Units of Computer System

1] Input Unit
2] Output Unit
3] CPU
4] Memory unit

1 Fetch
2 Decoded
3 Executed

Memory unit
↳ i] Primary Memory (RAM)
↳ ii] Secondary Memory (HD)

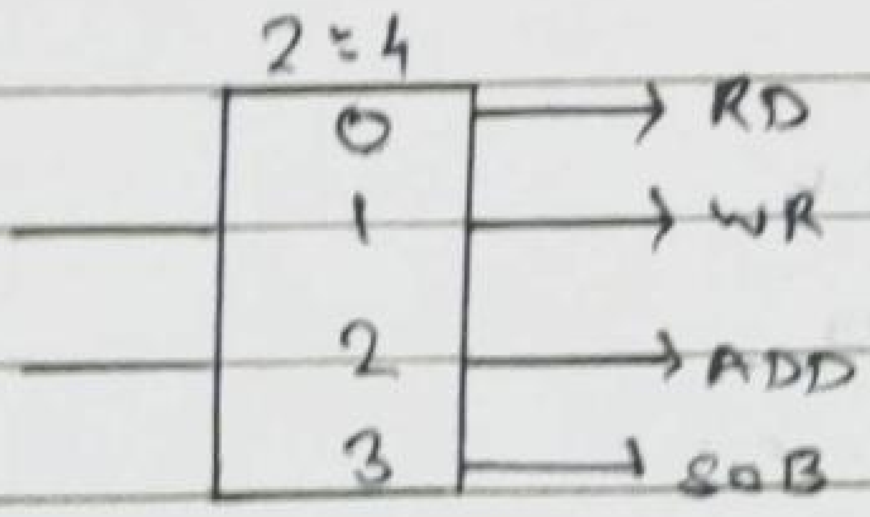


Micro Instructions

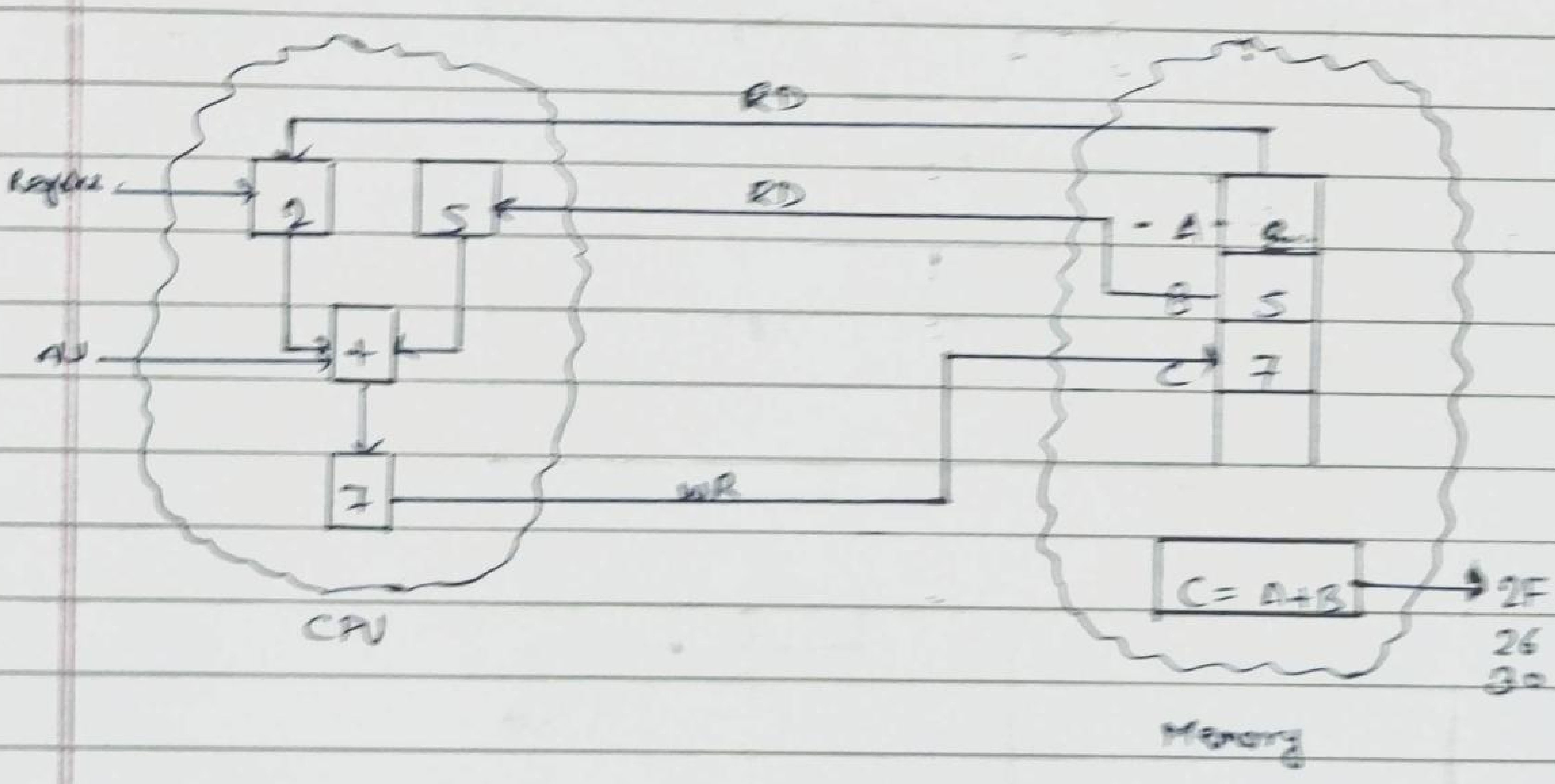
1. Read A (RD)
2. Read B (RD)
3. Add A with B (ADD)
4. Store in C (WR)

RD = 00
WR = 01
ADD = 10
SOB = 11

Decoder :-



CPU
- → Registers
  - → Accumulator
  - → Current Instruction Register / Instruction Register (IR)
  - → Program Counter (PC)
  - → MAR
  - → MDR
- → ALU
- → CU



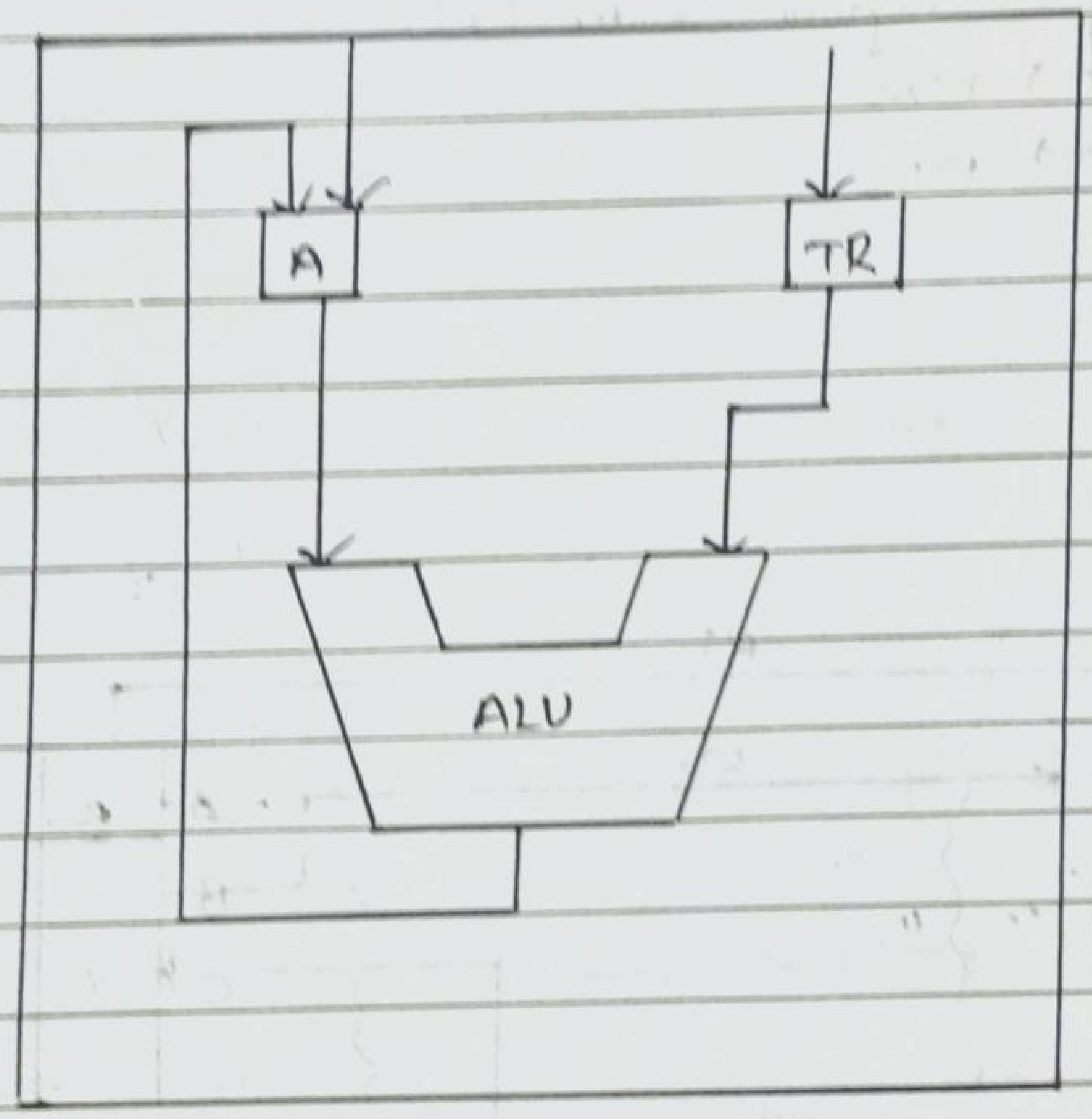
* Registers

Registers are the small memories inside the CPU which stores the data temporary before & after computation.

Register → 1] Accumulator (A)
2] Temp Register

1] Accumulator

It is the register which stores the first operand as well as immediate result after ALU operation.



Mondal

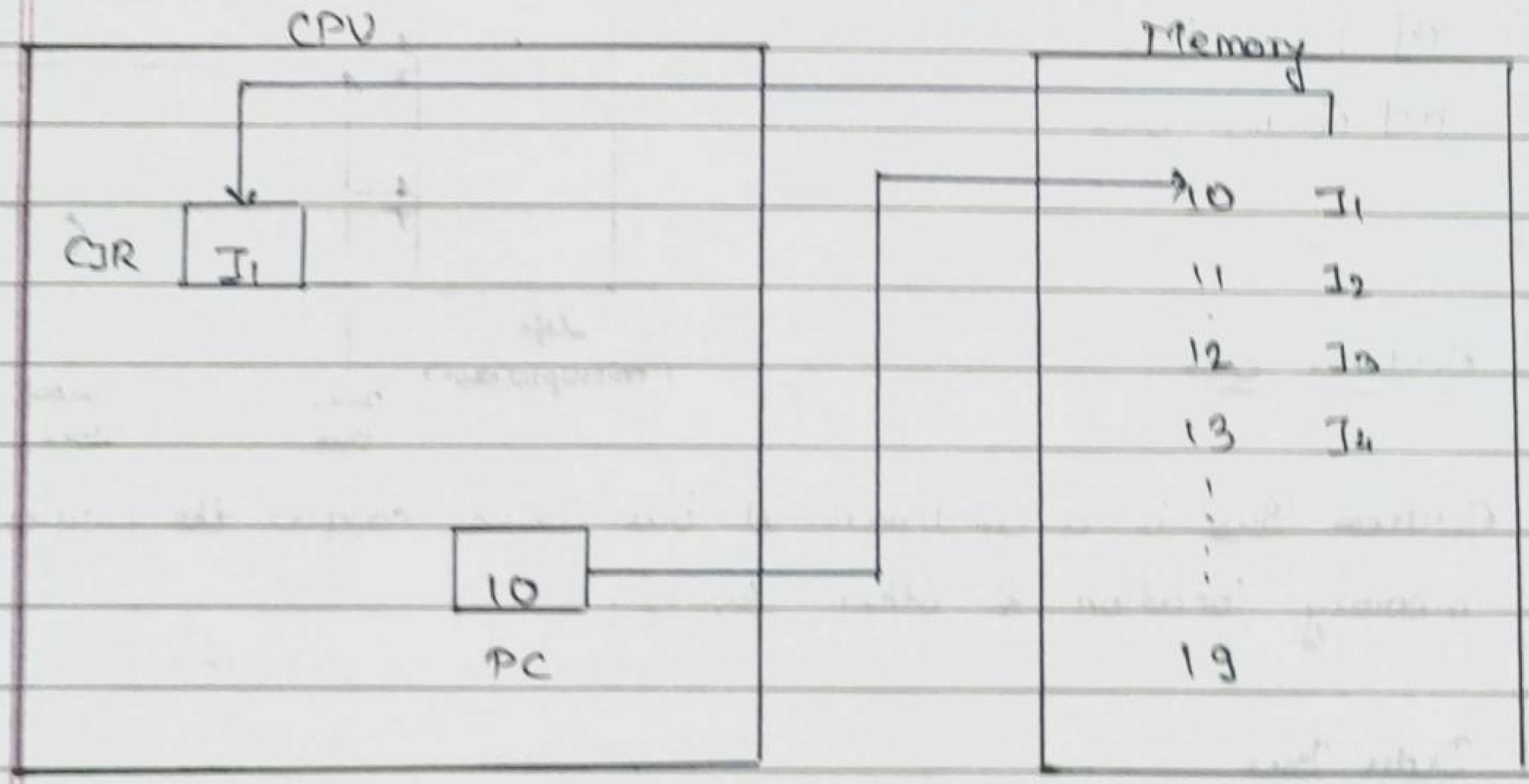
2] Program Counter (PC)

- Program Counter keeps the address of next instruction to be fetched from the memory.

- After execution of current instruction PC automatically incremented to point to the next to instruction to be executed.

- It is also called as instruction pointer

3] Current Instructor Register / Instructor Register (IR)



IR holds the current instruction to be executed.

4] Memory Address Register (MAR)

This register holds the address of memory location from where the instruction or data to be accessed from the memory.

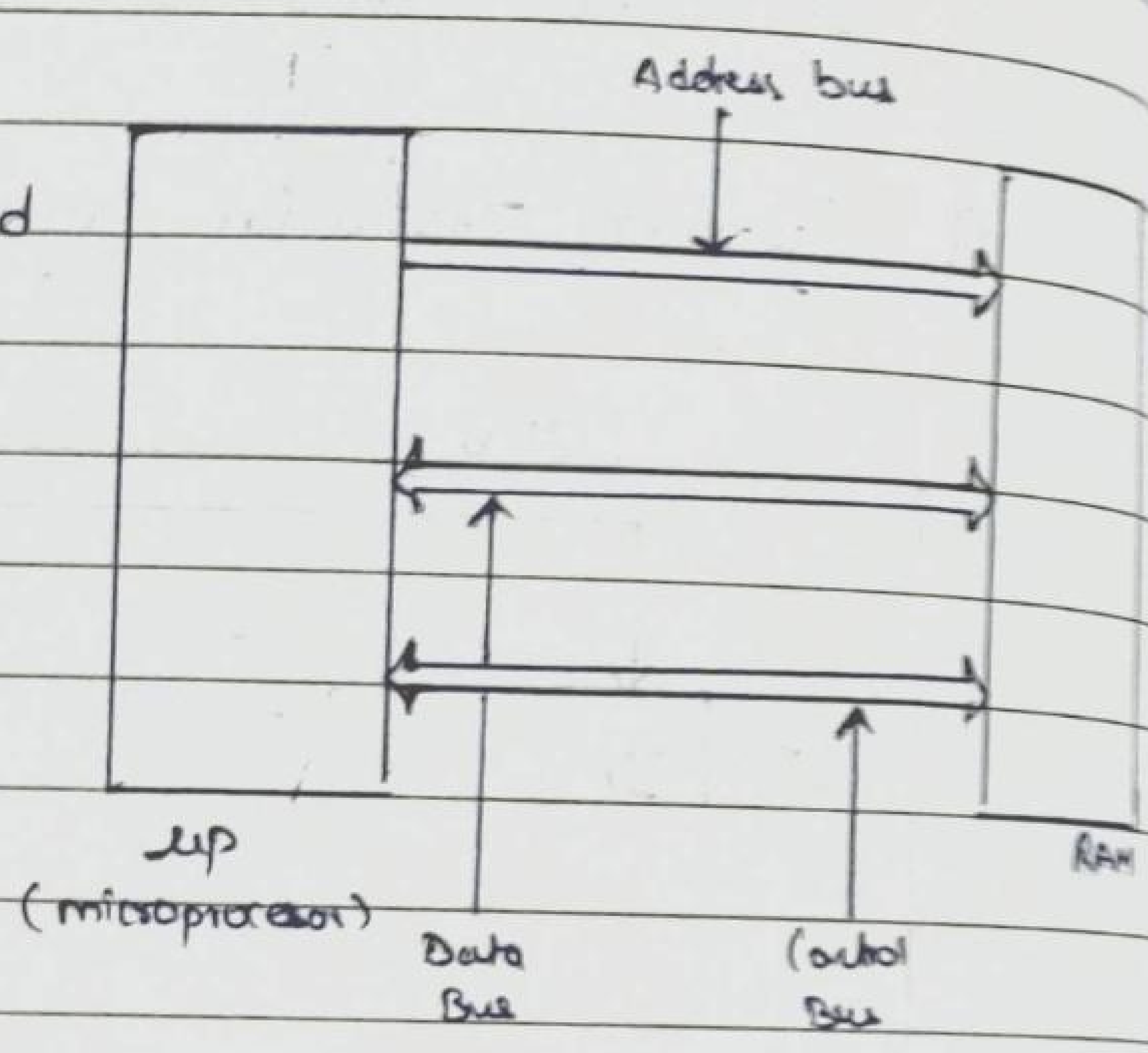
5] Memory Data Register (MDR)

It holds the data that is being transferred to or from the memory.

* Busses

Bus is a group of wire which carries electric signals from one device to another device.

3 types of Busses are identified
i] Address bus
ii] Data bus
iii] Control bus



1] Address Bus

Address Bus is a unidirectional bus which carries the address for memory location & other devices.

2] Data Bus

It is a bidirectional bus which carries the data transferred to and from up and other devices.

3] Control Bus

It is a bidirectional bus which carries control signals. Control signals are transferred in both the directions.
The examples of control signals are read, write, ready, etc.

MODULE-2
PAGE No. 27
DATE 10/08/2022

* DATA REPRESENTATION & ARITHMETIC ALGORITHMS *

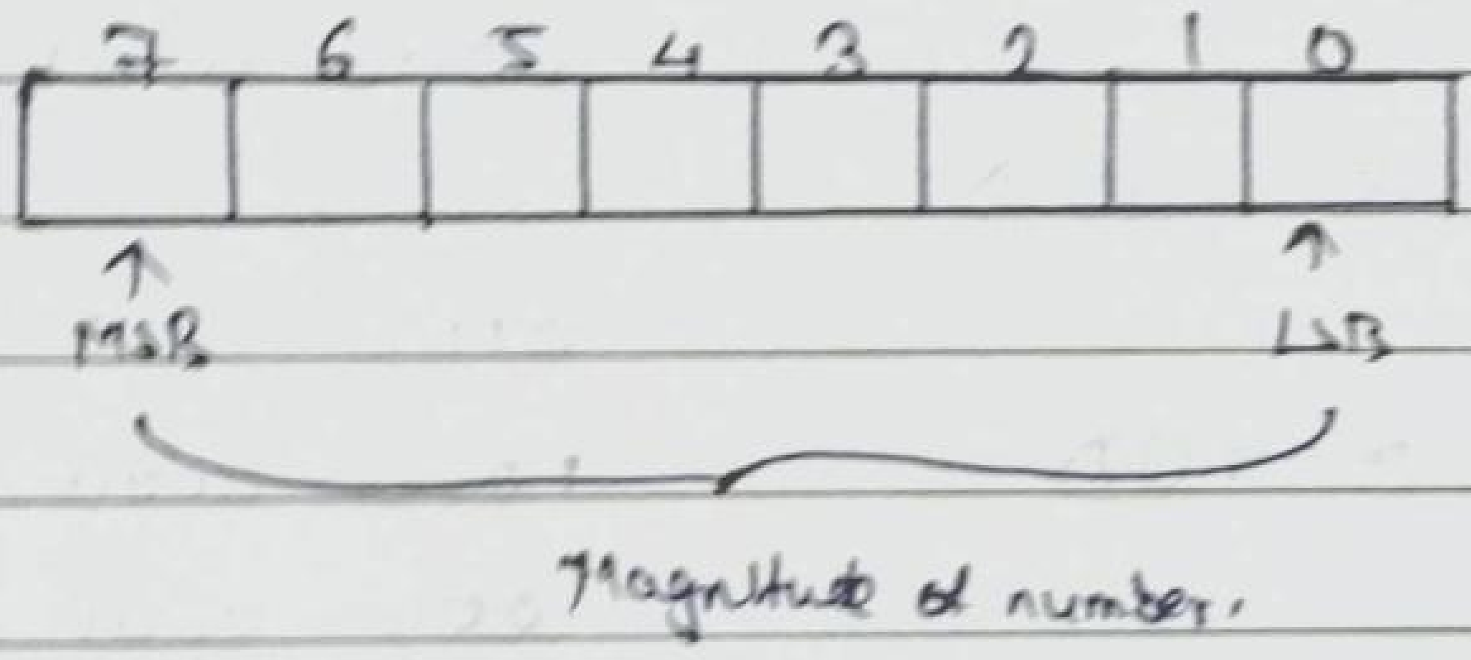
* Binary Integer Representation

i] Unsigned Integers
- can represent 0 and +ve numbers

ii] Signed Integers
- can represent 0, +ve & -ve numbers.
  a] Signed Magnitude Representation.
  b] 1's Complement Representation
  c] 2's complement Representation

1] Unsigned Integers

Ex. 8 bit unsigned binary integers



Ex. 0100 0001 = 65D
$\hookrightarrow 1\times2^6$ $\hookrightarrow 1\times2^0$

Ex. 1100 0000
1×2^7 $\hookrightarrow 1\times2^6$ = 192D

Ex. 0000 0000
$\hookrightarrow$ 0D

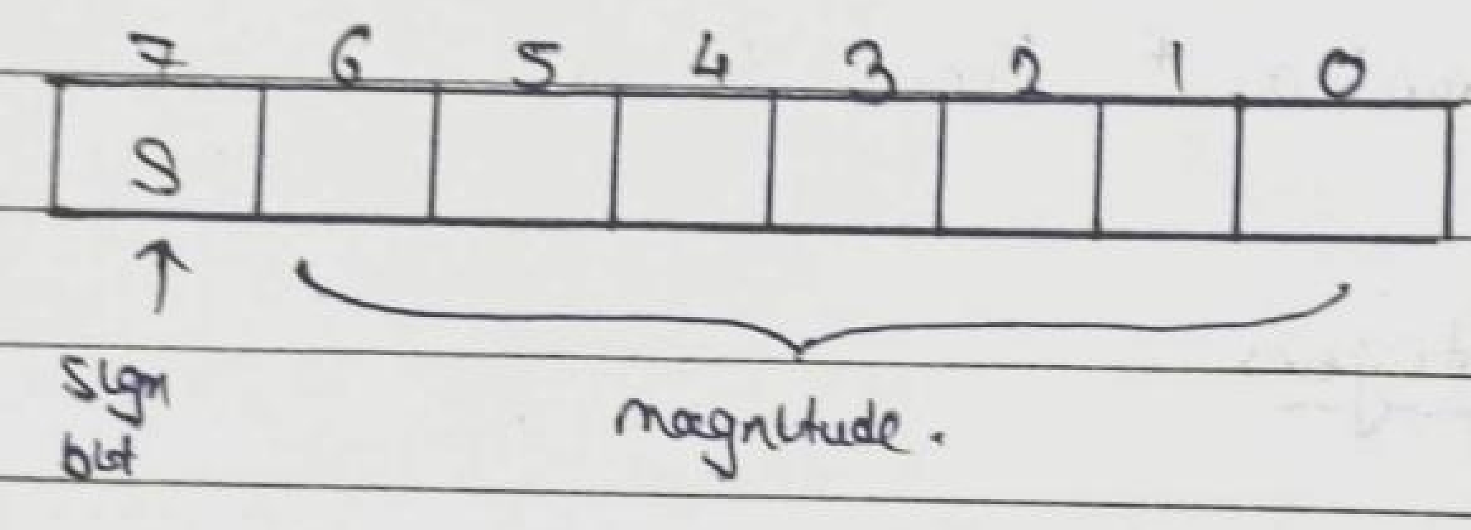
2] Signed Integers

In signed integer representation most significant bit (MSB) is reserved as a sign bit.

If MSB = 0 ; integer is +ve.

If MSB = 1 ; integer is -ve.

a] Signed Magnitude Representation



Ex. 0100 0001 = +65 D

Ex. 1100 0001 = -65 D

Ex. 0000 0000 = +0D

Ex. 1000 0000 = -0D

(Range)

0000 0000	+0
0000 0001	+1
⋮	
0111 1111	+127
1000 0000	-0
1000 0001	-1
⋮	
1111 1111	-127

(-127 ... -0 +0 ... +127)

- Drawback of signed magnitude representation is there are two representation for the number '0', which ~~cou~~ could lead to inefficiency and confusion. and also +ve and -ve numbers to be processed seperately.

b] 1s complement representation

The 1s complement of a number is formed by changing all 1 to 0 and all 0 to 1.

Ex. 0110 = +6
1s 1001 = -6

Ex. 01001 = +9
1s 10110 = -9

11110 = -1

10110 = -9

00110 = +6
11001 = -6
sign bit

(Range)

0000 0000 = +0
1000 0000 = -127
1111 1111 = -0

0000 0000 +0
0000 0001 +1
⋮
0111 1111 +127
1000 0000 -127
1000 0001 -126
⋮
1111 1110 -1
1111 1111 -0

(-127 --- -0 +0 --- +127)

c] 2s complement representation

2s complement of a number is obtained by taking 1s complement of a number and adding 1 to the LSB.

2s complement = 1s complement + 1

Ex. 2s complement 6 is 0110

$$\begin{array}{r} 0110 \\ 1001 \\ + \quad 1 \\ \hline 1010 \end{array}$$ → 2s complement.

Ex. 0100 0001 → 65D

$$\begin{array}{r} +1001\ 1110 \\ 1 \\ \hline 1011\ 1111 \end{array}$$ → 2s complement

$$\begin{array}{r} 1000\ 0001 \\ \\ +\ 0111\ 1110 \\ 1 \\ \hline 0111\ 1111 \end{array}$$ ⇒ −127D

Ex. 1111 1111 ⇒ −1D

$$\begin{array}{r} 0000\ 0000 \\ 1 \\ \hline 0000\ 0001 \end{array}$$

Ex. 1000 0000

$$\begin{array}{r} +\ 0111\ 1111 \\ \hline 1000\ 0000 \end{array}$$ →

- (Range)

0000	0000	+0
0000	0001	+1
⋮		
0111	1111	+127
1000	0000	−128
1000	0001	−127
⋮		
1111	1111	−1

(−128 — 0 — 127)

ASSIGNMENT NO. 1

1] What are the ranges of 8, 16, 32, 64 bit Integer in unsigned and signed representation.

2] ~~Use the val~~ Give the value of 88, 0, 127, 255, 1 in 8 bit unsigned representation.

3] Give the value of +88, -88, -1, 0, +1, -128, +127 in 8 bit 1s complement & 2s complement representation.

* Binary Arithematic

1] Binary addition

Rules:-

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

- Perform the addition of following numbers:-

1]
```
+  1 0 1 0 1
   1 1 0 0 1
 -----------
 1 0 1 1 1 0
```

2) Sub Binary Subtraction

Rules :-

A	B	Differ	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Ex.

$$\begin{array}{r} 1011 \\ -\ 0110 \\ \hline 0101 \end{array} \qquad \begin{array}{r} 0110 \\ 1011 \\ \hline 11011 \end{array}$$

THURSDAY

MODULE-2

PAGE No. 33

DATE 11/08/2022

DATA REPRESENTATION & ARITHMETIC ALGORITHMS

3] Binary Multiplication

Rules:-

A	B	Multiplication
0	0	0
0	1	0
1	0	0
1	1	1

Ex.

```
            1 1 1 0
          * 1 0 1 1
  -----------------
            1 1 1 0
  +       1 1 1 0 x
  +     0 0 0 0 x x
  +   1 1 1 0 x x x
  -----------------
    1 0 0 1 1 0 1 0
```

(1110) * (1011) = 1001 1010

If we multiply two 4-bit number, result is 8-bit number.

4] Binary Division

Ex. 4]

```
              1101
       1001 | 1110101
            - 1001↓
              01011
            -  1001
              00100
            -     0
               1001
            -  1001
               0000
```

If divisor is 4 bit, Quotient & Remainder is 4-bit, & Dividend is 8-bit & Division is called 4-bit division.

ii] ~~15 ÷ 3~~ 15/3

```
       101
11 ) 1111
   - 11↓↓
     0011
       11
     ----
       00
```

iii] 15/2

```
       111
10 ) 1111
   - 10
     011
   - 10
     011
   - 10
     ---
      01
```

5] 1s Complement Arithmetic

Ex. i] -7 + 8

(00111) (01000)

11000

```
        + 01000
          11000
  carry 1 00000
          ↳    1
        -------
          00001  ⇒ final answer
```

If carry is generated ans is +ve, add to the rest answer.

+1 = +(0001)

ii] 7 + (-8)

(00111) (01000) 1s → 10111

$$
\begin{array}{r}
+\ 00111 \\
10111 \\
\hline
11110
\end{array}
$$

00001 ⇒ Final answer

Since carry is not generated take its 1s complement of the answer.

(-1) = -(0001)

iii] 13 + (-9)

~~X~~

(1101) + -(1001)

1s: 01101 → 10010

01001 1s → 10110

iii] 13 + (-9)

(1101) + (-1001)

1001 1s → 0110

$$
\begin{array}{r}
+\ 1101 \\
0110 \\
\hline
\boxed{1}\,0011 \\
1 \\
\hline
0100
\end{array}
$$

0100 → final answer (4)

iv] (−1101) + (1001)
(−13) + (9)

$1s \hookrightarrow$ 1101
0010

+ 0010
1001

1011
$1s \hookrightarrow$ 0100

Final Answer − (0100)

v] 100 + (−11)

11 → 011 } 1s
100

size of number should be same.

+ 100
100

1] 000
1

001

vi] (−100) + (−101)

0100 } 1s
0011

0101 } 1s
1010

```
 + 1011
   1010
 -------
 1]0101
+     1
 -------
1s ( 0110
     1001   ⇒ Final Answer
```

vii] (-1101) + (-1001)

```
1s ( 01101            01001 ) 1s
     10010            10110

  + 10010
    10110
  --------
  1]01000
 +       1
  --------
    01001
1s (
    10110   ⇒ Final Answer
   ↳(-10110)
```

6] 2s complement Arithmetic

Ex. i] (1101) + (-1001)

```
            1001 ) 2s
            0111

 + 1101
   0111
 -------
 1]0100   ⇒ +(0100) ⇒ final answer.
```

If carry is generated, ignore it answer is rest of the number

ii] (-1101) + (1001)

```
2s ( 0011      + 0011
                 1001
                ------
                 1100
             2s ( 0100  ⇒ final answer
                 -(0100)
```

If no carry is generated, take 2s complement of answer and -ve sign.

ex] (−100) + (−101)

0100) 1s
+ 1011
1
1100 (2s)

0101) 1s
+ 1010
1
1011 (2s)

+ 1100
1011
$\boxed{1}$0111) 2s
1001

Ignore carry

−(1001) ⇒ final answer:

Scanned by TapScanner

* Hexadecimal Arithematic

$$\begin{array}{r} +\ 7A \\ BA \\ \hline +134 \end{array} \qquad \begin{array}{r} 0111\ 1010 \\ 1011\ 1010 \\ \hline 10011\ 0100 \end{array}$$

$(+134)_{16}$

$$\begin{array}{r} C0 \\ -\ 7A \\ \hline +46 \end{array} \qquad \begin{array}{r} 1100\ \ 0000 \\ 1000\ \ 0110 \\ \hline 1\,|\,0100\ \ 0110 \end{array} \qquad \begin{array}{ll} 0111\ 1010 \xrightarrow{2s} & (7A) \\ 1000\ 0110 & (-7A) \end{array}$$

ignore the carry and result is +ve.

$$\begin{array}{r} -\ 7A \\ C0 \\ \hline (-46) \end{array} \qquad \begin{array}{r} 0111\ \ 1010 \\ 0100\ \ 0000 \\ \hline 1011\ \ 1010 \end{array} \qquad \begin{array}{ll} 1100\ \ 0000 \xrightarrow{2s} & (C0) \\ 0100\ \ 0000 & (-C0) \end{array}$$

No carry generated ∴ ans is -ve thus take 2s complement

0100 0110

* BCD Arithematic (Binary Coded Decimal)

1] BCD Addition

2 + 2 = 4

$$\begin{array}{r} 0010 \\ 0010 \\ \hline 0100 \end{array}$$

6 + 7

$$\begin{array}{r} 0110 \\ 0111 \\ \hline 1101 \\ 0110 \\ \hline 10011 \end{array} \quad \Big\} \text{ add } 6$$

(13)

+ 24	+ 0010	0100	
25	0010	0101	
49	0100	1001	
	4	9	

+ 26	+ 0010	0110	
28	0010	1000	
54	0100	1110	+ 4E
	+ 0000	0110	06
	0101	0100	

+ 62	+ 0110	0010	
82	1000	0010	
144	1110	0100	+ E4
	+ 0110	0000	60
	1 0100	0100	144

+ 28	+ 0010	1000	
29	0010	1001	
57	0101	0001 ← because here it is a carry we have to add 6.	
	+ 0000	0110	
	0101	0111	

+ 99	+ 1001	1001	
88	1000	1000	
187	1 0010	0001	
	0110	0110	
	1 1000	0111	

2) BCD Subtraction

```
 41        + 0100  0001        0001  0110 } is (16)
-16          1110  1001        1110  1001      (-16)
 25        1 0010  1010
             └──→     1
             0010  1011
           , 0000  1010   ( Add A which is -6 )   0110 } (10)
             0010  0101                           1010 } 2s (-6)
                  ignore

-41          0100  0001
 12          1110  1101
 29        1 0010  1110
             └──→     1
             0010  1111
           + 0000  1010   (If digit is > 9 then only add A)
             0010  1001

 12          0001  0010
-41          1011  1110
-29          1101  0000   } 1s²
           + 0010  1111
             0000  1010   (add A)
             0010  1001
                  ignore
```

$$\begin{array}{r} + 341 \\ 252 \\ \hline 593 \end{array} \qquad \begin{array}{rrrr} + & 0011 & 0100 & 0001 \\ & 0010 & 0101 & 0010 \\ \hline & 0101 & 1001 & 0011 \end{array}$$

$$\begin{array}{r} - 341 \\ 252 \\ \hline 089 \end{array} \qquad \begin{array}{rrrr} + & 0011 & 0100 & 0001 \\ & 1101 & 1010 & 1101 \\ \hline & 1\,0000 & 1110 & 1110 \\ & & & 1 \\ \hline & 0000 & 1110 & 1111 \\ + & 0000 & 1010 & 1010 \\ \hline & & 1000 & 1001 \end{array}$$

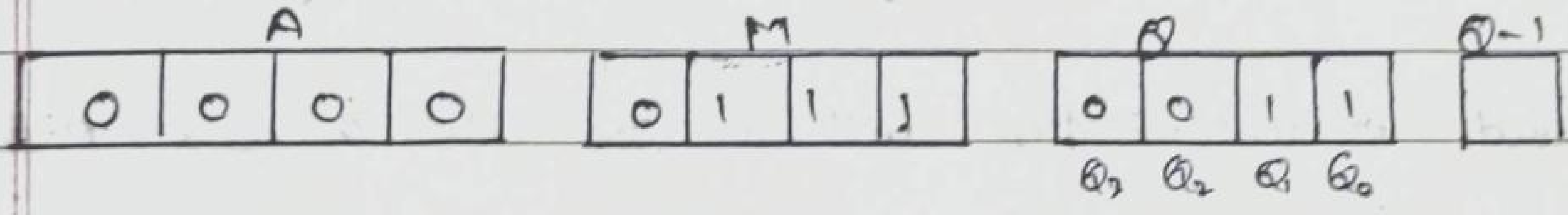
(add A)

* BOOTH's Algorithm :-

- Booth's algorithm is used for signed as well as unsigned multiplication.
- It uses 3 registers.
  - M → Multiplicand
  - Q → Multiplier
  - A → Accumulator

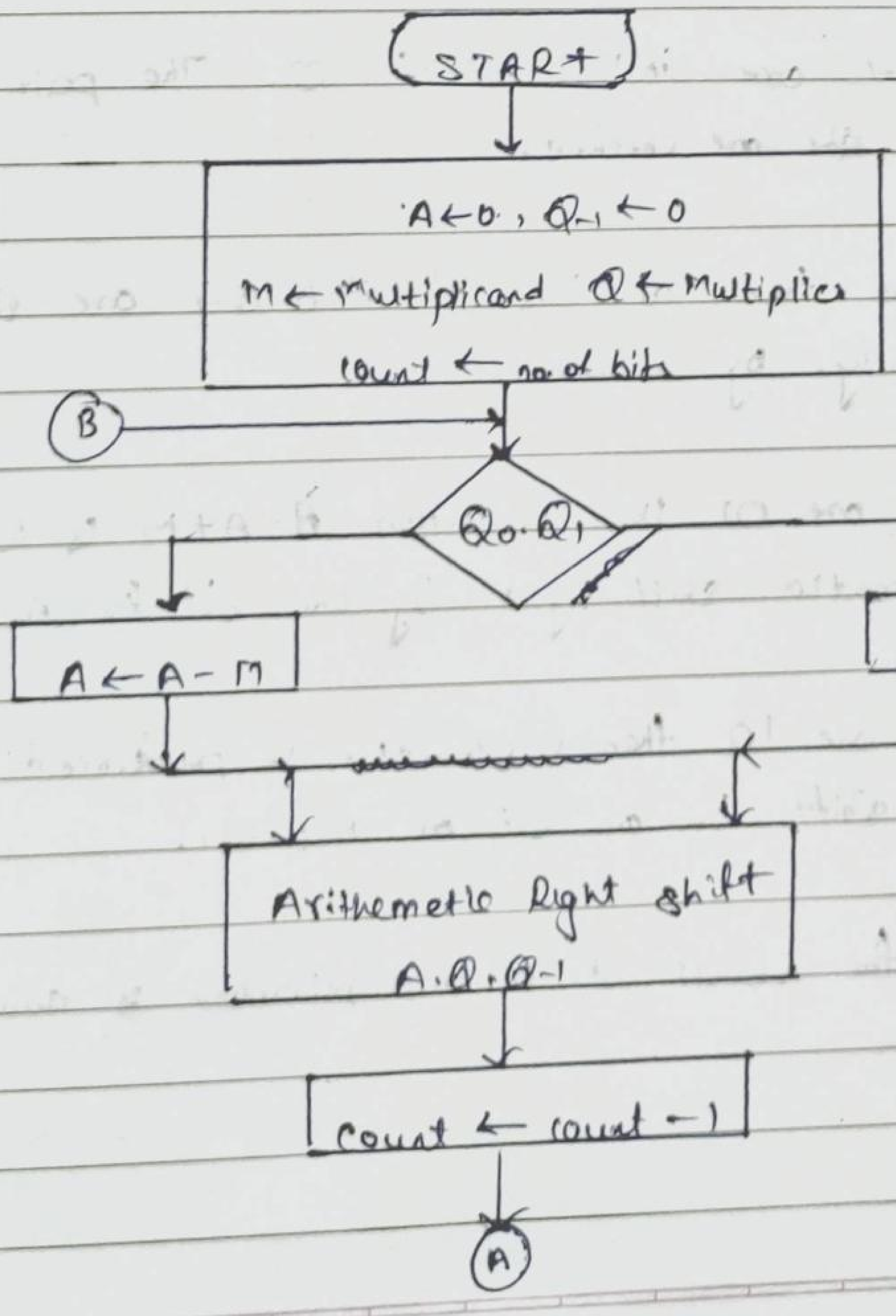
All the three registers are of same size. and accumulator (A) is initialized to 0.

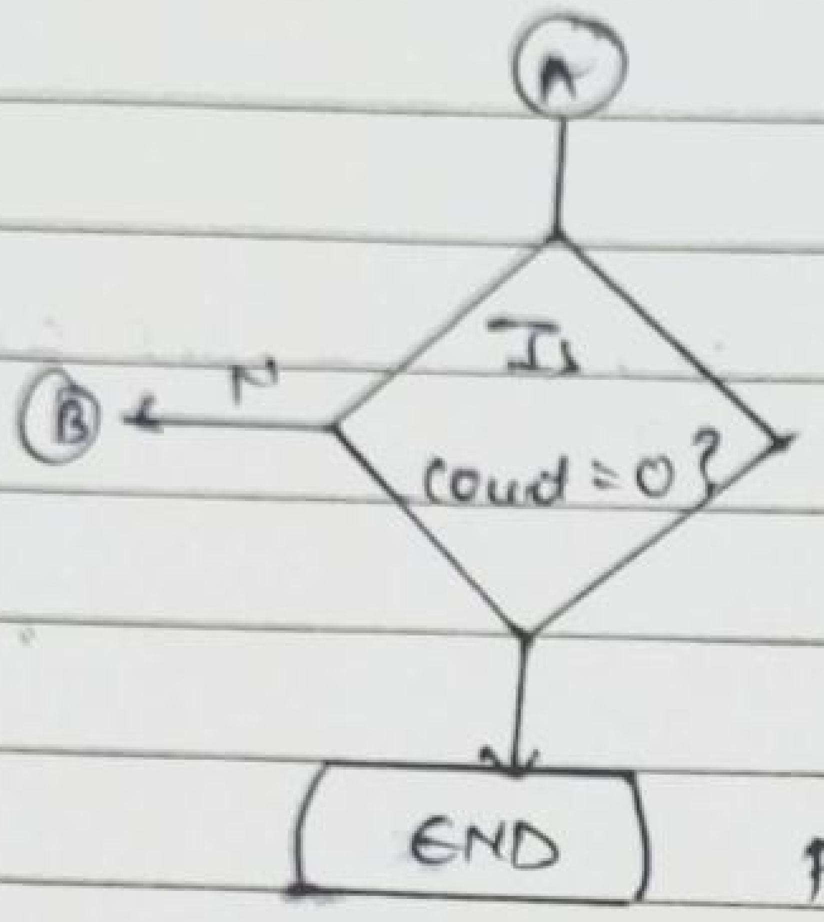
7 23
0111
0011



00
11

10 01





Product in A.Q.

- A one bit register placed logically to the right of the least significant bit Q_0 of register Q and designated as Q_{-1}.
- The result of multiplication will be available in pair A:Q register.
- Register A and Q_{-1} are initialized to 0. The pair of both two bits Q_0 and Q_{-1} are checked.
- If both are same then all bits of A.Q.Q_{-1} are shifted right arithmetically by one bit.
- If Q_0 and Q_{-1} are 01 then addition A = A + M is to be done and arithmetic shift right by one bit for A.Q.Q_{-1}.
- If Q_0 and Q_{-1} are 10 then subtraction is performed A = A - M and arithmetic shift by one bit for A.Q.Q_{-1}.
- After completion the result of of multiplication is available in pair A:Q register.

1] 7 × 6

A = 0000 ; Q = 0110 ; M = 0111 ; −M = 1001

Q_{-1} = 0

Count	A	Q	Q_{-1}	M	−M	Action
4	0000	0110	0	0111	1001	Initially
	0000	0011	0			ARS AQQ_{-1}
3	0000	0011	0			Q_0Q_{-1} = 10
	1001					A ← A − M
	1001	0011	0			
	1100	1001	1			ARS AQQ_{-1}
2	1100	1001	1			Q_0Q_{-1} = 11
	1110	0100	1			ARS AQQ_{-1}
1	1110	0100	1			Q_0Q_{-1} = 01
	1 0111					A ← A + M
	0101	0100	1			
	0010	1010	0			ARS AQQ_{-1}
0						

Product is available in AQ Register $(00101010)_2$ = $(42)_{10}$

2] 9x8

A = 0000;

Q = 1000;

M = 1001;

-M = 0111;

Q_{-1} = 0

Count	A	Q	Q_{-1}	M	-M	Actions
4	0000	1000	0	1001	0111	Initially
						Q_0Q_{-1} = 00
	0000	0100	0			ARS AQQ_{-1}
3	0000	0100	0			Q_0Q_{-1} = 00
	0000	0010	0			ARS AQQ_{-1}
2	0000	0010	0			Q_0Q_{-1} = 00
	0000	0001	0			ARS = AQQ_{-1}
1	0000	0001	0			Q_0Q_{-1} = 10
	0111					A ← A-M
	0111	0001	0			ARS - AQQ_{-1}
	0011	1000	1			

2] 9 × 8

A = 00000

Q = 01000

Q_{-1} = 0

M = 01001

−M = 10111

Count	A	Q	Q_{-1}	M	−M	
5	00000	01000	0	01001	10111	Initially
						Q_0Q_{-1} = 00
	00000	00100	0			ARS AQQ_{-1}
						Q_0Q_{-1} = 00
4	00000	00100	0			ARS AQQ_{-1}
	00000	00010	0			Q_0Q_{-1} = 00
3	00000	00010	0			ARS AQQ_{-1}
	00000	00001	0			Q_0Q_{-1} = 10
2	00000	00001	0			A ← A − M
	10111					ARS AQQ_{-1}
	10111	00001	0			
	11011	10000	1			Q_0Q_{-1} = 01
1	11011	10000	1			A ← A + M
	01001					ARS AQQ_{-1}
	~~00101~~	~~01000~~	~~0~~			
	00100	10000	1			
	$\boxed{00010}$	$\boxed{01000}$	0			
0						

$(00010\ 01000)_2 = (72)_{10}$

3] -7×6

~~Count~~ A = 0000

Q = 0110

~~0000~~ M = 1001 (-7)

-M = 0111 (7)

Count

Count	A	Q	Q_{-1}	M	-M	Action
4	0000	0110	0	1001	0111	Initialize
						Q_0Q_{-1} = 00
	0000	0011	0			ARS AQQ_{-1}
3	0000	0011	0			Q_0Q_{-1} = 10
	0111					A ← A-M
	0111	0011	0			ARS AQQ_{-1}
	0011	1001	1			
2	0011	1001	1			Q_0Q_{-1} = 11
	0001	1100	1			ARS AQQ_{-1}
1	0001	1100	1			Q_0Q_{-1} = 01
	+ 1001					A ← A+M
	1010	1100	1			ARS AQQ_{-1}
	1101	0110	0			
0	1101	0110	0			

1101 0110 ~~0~~

↑ MSB is 1 ∴ no is -ve.

$(1101\ 0110)_{2's}$

0010 1010

↓

$\boxed{-42}$.

Q) 7 × -6

A = 0000

Q = 1010 (-6)

M = 0111

-M = 1001

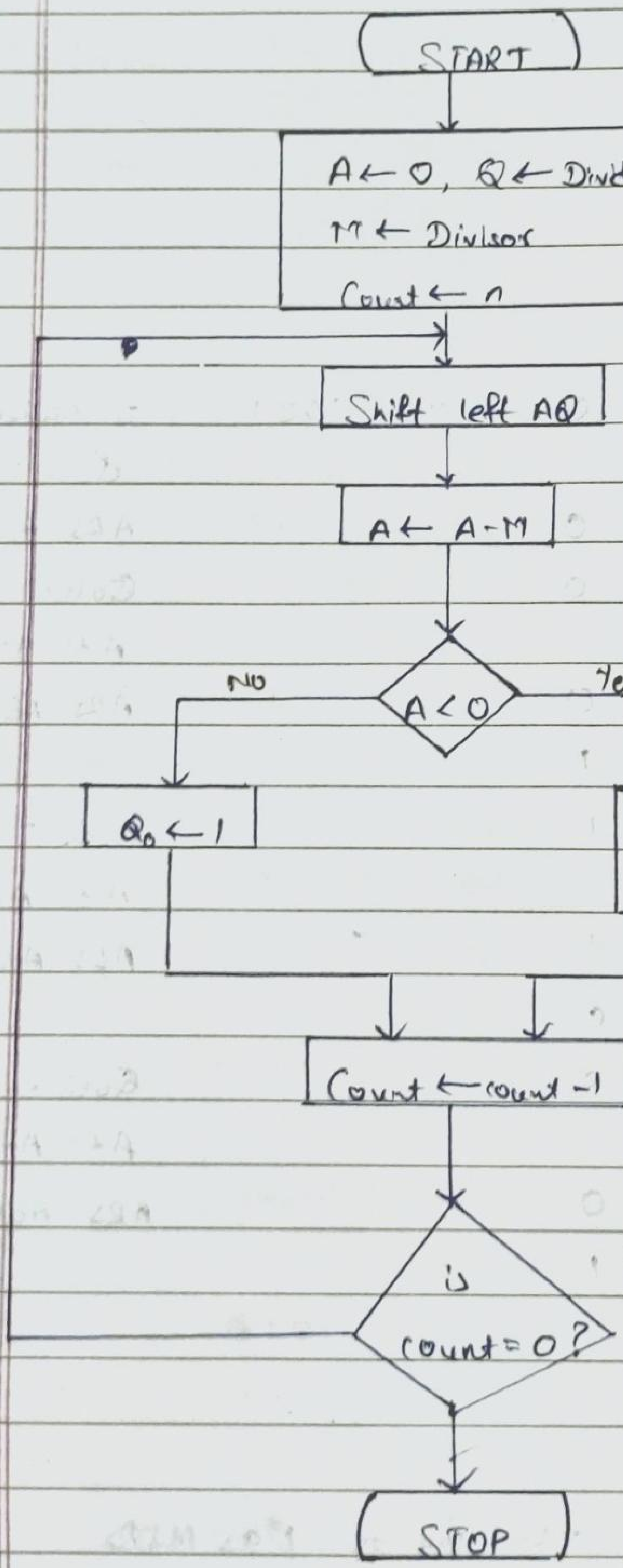
Count	A	Q	Q_{-1}	M	-M	
4	0000	1010	0	0111	1001	Initialise
						Q_0Q_{-1} = 00
	0000	0101	0			ARS AQQ_{-1}
3	+ 0000	0101	0			Q_0Q_{-1} = 10
	1001					A ← A-M
	1001	0101	0			ARS AQQ_{-1}
	1100	1010	1			
2	+ 1100	1010	1			Q_0Q_{-1} = 01
	0111					A ← A+M
	0011	1010	1			ARS AQQ_{-1}
	0001	1101	0			
1	+ 0001	1101	0			Q_0Q_{-1} = 10
	1001					A ← A-M
	1010	1101	0			ARS AQQ_{-1}
	(1101	0110)	1			

$(1101\ 0110)_{2s}$ -ve due to 1 as MSD

0010 1010

= -42.

* Restoring Division



1] $13 \div 3$, $Q = 4$, $R = 1$

$A \leftarrow 0000$; $Q \leftarrow 1101$; $M \leftarrow 0011$; $-M \leftarrow 1101$, $n \leftarrow 4$

Count	A	Q	M	-M	Actions
4	0000	1101	0011	1101	Initially
	+ 0001	101_			SHL AQ
	1101				$A \leftarrow A - M$
	1110	1010			$A < 0$; $Q_0 \leftarrow 0$
	+ 0011				$A \leftarrow A + M$
	0001	1010			
3	0001	1010			SHL AQ
	+ 0011	010_			$A \leftarrow A - M$
	1101				
	0000	0101			$A \geq 0$; $Q_0 \leftarrow 1$
2	0000	0101			
	+ 0000	101_			SHL AQ
	1101				$A \leftarrow A - M$
	1101	1010			$A < 0$; $Q_0 \leftarrow 0$
	+ 0011				$A \leftarrow A + M$
	0000	1010			
1	0000	1010			
	+ 0001	010_			SHL AQ
	1101				$A \leftarrow A - M$
	1110	0100			$A < 0$; $Q_0 \leftarrow 0$
	+ 0011				$A \leftarrow A + M$
	0001	0100			
0	0001	0100			
	(R)	(Q)			

$R = 1$, $Q = 4$

27] 10 ÷ 2 , Q = 5 , R = 0

A ← 0000 ; Q ← 1010 ; M ← 0010 ; −M ← 1110 ; n ← 4

Count	A	Q	M	−M	Actions
4	0000	1010	0010	1110	Initialize.
	+ 0001	010_			SHL AQ
	1110				A ← A−M
	1111	0100			A < 0 ; Q0 ← 0
	+ 0010				A ← A+M
	0001	0100			
3	0001	0100			
	+ 0010	100_			SHL AQ
	1110				A ← A−M
					~~A ≥ 0 ; Q0 ← 1~~
	0000	1001			A ≥ 0 ; Q0 ← 1
2	0000	1001			
	+ 0001	001_			SHL AQ
	1110				A ← A−M
	1111	0010			A < 0 ; Q0 ← 0
	+ 0010				A ← A+M
	0001	0010			
1	0001	0010			
	+ 0010	010_			SHL AQ
	1110				A ← A−M
	0000	0101			A ≥ 0 ; Q0 ← 1
0	0000	0101			
	(R)	(Q)			

R = 0 , Q = 5 .

3] 23 ÷ 4 , Q = 5 , R = 3

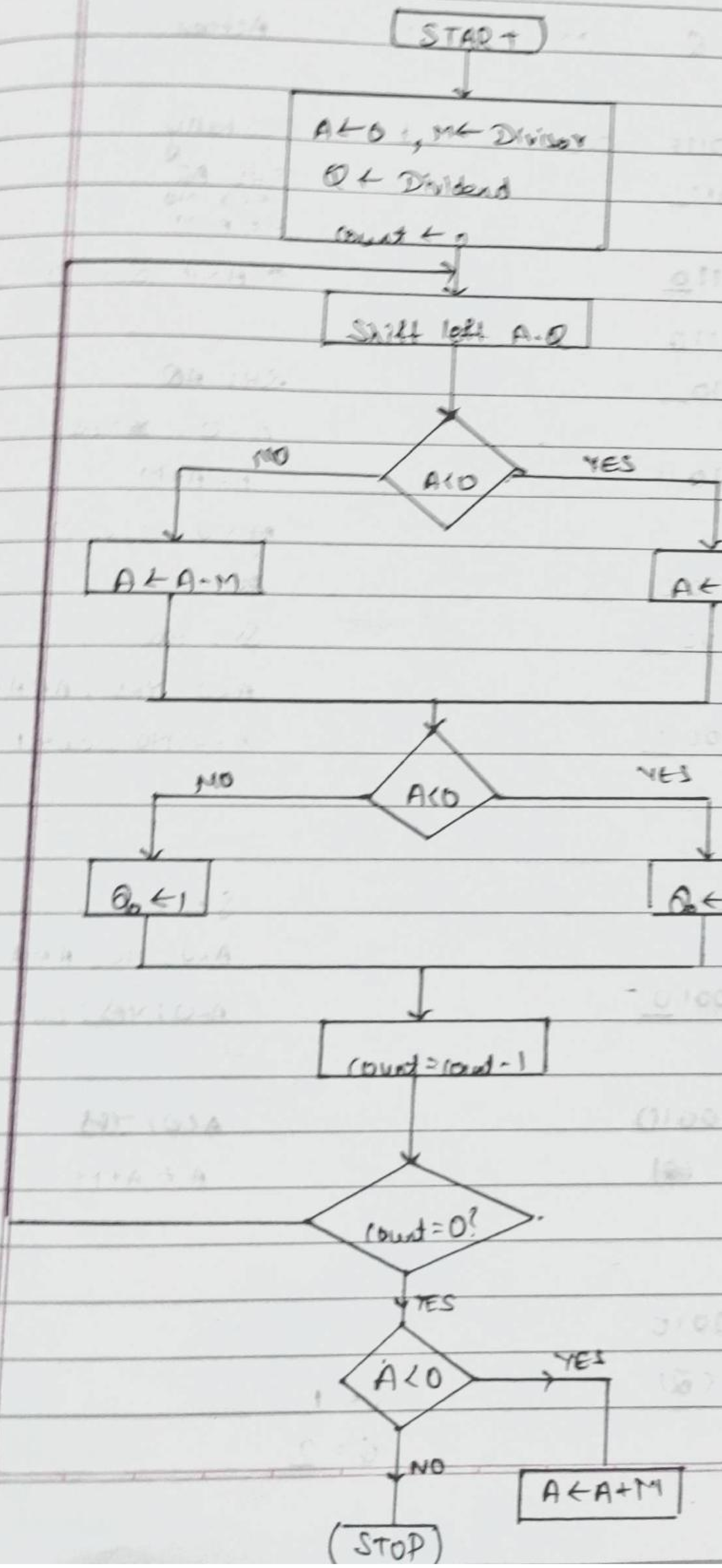
A ← 00000 ; Q ← 10111 ; M ← 00100 ; −M ← 11100 ; n ← 5

Count	A	Q	M	−M	Actions
5	00000	10111	00100	11100	Initialize
	+ 00001	0111_			SHL AQ
	11100				A ← A − M
	11101	0111<u>0</u>			A < 0 ; Q_0 ← 0
	+ 00100				A ← A + M
	00001	01110			
4	00001	01110			
	+ 00010	1110_			SHL AQ
	11100				A ← A − M
	11110	1110<u>0</u>			A < 0 ; Q_0 ← 0
	+ 00100				A ← A + M
	00010	11100			
3	00010	11100			
	+ 00101	1100_			SHL AQ
	11100				A ← A − M
	00001	1100<u>1</u>			A ≥ 0 ; Q_0 ← 1
2	00001	11001			
	+ 00011	1001_			SHL AQ
	11100				A ← A − M
	11111	1001<u>0</u>			A < 0 ; Q_0 ← 0
	+ 00100				A ← A + M
	00011	10010			
1	00011	10010			
	+ 00111	0010_			SHL AQ
	11100				A ← A − M
	00011	0010<u>1</u>			A ≥ 0 ; Q_0 ← 1
0	00011 (R)	00101 (Q)			

4] $9 \div 8$, $Q = 1$, $R = 1$ (If both are 4 bits take 5 bits.)

$A \leftarrow 0000$; $Q \leftarrow$

* Non Restoring Division



A : Remainder

Q : Quotient

1] 7 ÷ 3

A ← 0000 ; Q = 0111 ; M = 0011 ; −M = 1100 ; n = 4

Count	A	Q	M	−M	Action
4	0000	0111	0011	1100	Initially
	0000	111_			SHL AQ
	+ 1100				A<0 NO
					A ← A−M
	1100	111$\underline{0}$			A<0 ; Q_0 ← 0
3	1100	1110			
	1001	110_			SHL AQ
	+ 0011				A<0 ; YES
	1100	110$\underline{0}$			A ← A+M
					A<0 ; Q_0 ← 0
2	1100	1100			
	1101	100_			SHL AQ
	+ 0011				A<0 YES ; A ← A+M
	0000	100$\underline{1}$			A<0 ; NO ; Q_0 ← 1
1	0000	1001			
	0001	001_			SHL AQ
	+ 1101				A<0 ; NO ; A ← A−M
	1110	001$\underline{0}$			A<0 ; YES ; Q_0 ← 0
0	1110	0010			A<0 ; YES
					A ← A+M
	1110				
	+ 0011				
	0001	0010			
	(R)	(Q)			

R = 1

Q = 2

2] 8 ÷ 8

A ← 00000 ; Q = 01000 ; M = 01000 ; ~~-M = 00111, -M = 11~~ -M = 11000

Count	A	Q	M	-M	Action
5	00000	01000	01000	11000	Initially

* Floating point number representation :-

IEEE 754 Format

i] 32-bit (Single Precision)
ii] 64-bit (Double Precision)

1] Single precision format (32 bit)

31	30 – 23	22 – 0
1 bit	8 bits	23 bits
sign bit	bias exponent	Mantissa

Bias Exponent :- Exponent + 127 ← $2^7 - 1 = 127$

= Exponent + $(7F)_{16}$

2] Double precision format (64 bits)

63	62 – 52	51 – 0
1 bit	11 bits	52 bits
sign bit	bias exponent	Mantissa

Bias Exponent :- Exponent + 1023 ← $2^{10} - 1 = 1023$

= Exponent + $(3FF)_{16}$

Q. Represent the following numbers into single precision format

1] $(10.25)_{10}$

→ ~~$(10.25)_{10}$~~ → $(1010.0010\ 0101)_2$ ~~~~

~~1.025×10^1~~ →

$0.25 \times 2 = 0.5$

$0.5 \times 2 = 1.0$

$(10.25)_{10} \rightarrow (1010.0100)_2$

1010.0100

1.0100100×2^3

$1.M \times 2^E$

$E = 3$

$BE = 3 + 127$
$= (130)_{10}$

$BE = 3 + 7F$
$= (82)_{16}$
↓

$BE = (1000\ 0010)_2$

$M = 0100100$

Sign in +ve

$\underline{0}\ \underline{1000}\ \underline{0010}\ \underline{01001000}\ldots$

$(41240000)_{16}$

2] $(10.25)_{10}$

$0.25 \times 2 = 0.5$

$0.5 \times 2 = 1.0$

$(10.25)_{10} \to (1010.0100)_2$

1.010010×2^3

$1.M \times 2^E$

$E = 3$

$BE = 3 + (3FF)_{16}$

$= (402)_{16}$

$BE = \underbrace{(0100\ 0000\ 0010}_{11 \text{ bits only.}})_2$

$B.E = (100\ 0000\ 0010)_2$

$M = 010010$

Sign is +ve.

$\underline{0}\ \underline{100}\ \underline{0000}\ \underline{0010}\ \underline{0100}\ \underline{1000}$ - - -

$(4024800000000000)_{16}$

PAGE No.
DATE / /

3] $(-8.08)_{10}$

→ $8 = (01000)_2$

$0.08 = (000101)_2$

$0.08 \times 2 = 0.16$
$0.16 \times 2 = 0.32$
$0.32 \times 2 = 0.64$
$0.64 \times 2 = 1.28$
$0.28 \times 2 = 0.56$
$0.56 \times 2 = 1.12$

$(-8.08) = -(1000.000101)$

$1.M \times 2^E$

$M = 1.000000101$, $E = 3$

$BE = 7F + 3$
$= (82)_{16}$

$BE = 1000\ 0010$

IEEE 754 = 1100 0001 0000000010100 ....

$= (C1010000)_{16}$

4] (-35.35)

→ $(35)_{10} = (100011)_2$

$0.35 \times 2 = 0.70$
$0.7 \times 2 = 1.40$
$0.4 \times 2 = 0.8$
$0.8 \times 2 = 1.6$
$0.6 \times 2 = 1.2$
$0.2 \times 2 = 0.4$

$0.35 = (010110)_2$

$(-35.35)_{10} = (100011.010110)_2$

1.00011010110×2^5

$M = 00011010110$, $E = 5$

Scanned by TapScanner

$BE = 7F + 5$
$= (84)_{16}$

$= (1000 \quad 0100)_2$

IEEE754 = (1100 0010 0000 1101 0110)

$= (C20D6000)_{16}$

Q. Convert the following IEEE 754 32-bit number into its equivalent decimal number.

1] 1 0111 1111 1000 0000 = (BF(00000)

→ $\underline{1}$ $\underline{0111 \ 1111}$ $\underline{1000 \ 0000}$
Signed BE M

$BE = 7F$
$E = 7F - 7F = 0$
$M = 1000 \quad 0000$
$1.M \times 2^E$
$-(1.1000\ 0000) \times 2^0$
$= -(1.10000000)_2 \Rightarrow (-1.5)_{10}$

2] C2430000

→ $\underline{1}\underline{100 \ 0010 \ 0}\underline{100 \ 0011 \ 0000} \ldots$
BE M

$BE = 84$
$E = 84 - 7F = 5$
$M = 1000 \ 0110 \ 000.$

$-(1.1000\ 0110) \times 2^{*}$

$= -(110000.11000)$

$= -(48.75)$

Sign	Binary Exponent	Mantissa	Number
0	0	0	+0
1	0	0	-0
0	FF	0	$+\infty$
1	FF	0	$-\infty$
x	0	Non-zero	Denormalized number
x	FF	Non-zero	Not a number (NaN)

BOOLEAN ALGEBRA

* Boolean Laws

1] AND laws

$A \cdot 0 = 0$

$A \cdot A = A$

$A \cdot 1 = A$

$A \cdot \bar{A} = 0$

2] OR laws

$A + 0 = A$

$A + 1 = 1$

$A + A = A$

$A + \bar{A} = 1$

3] Commutative laws

$A+B = B+A$

$A \cdot B = B \cdot A$

4] Associative law

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

$A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C)$

5] Distributive law

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

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

6] De Morgan's law

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

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

7] Other laws

$A + AB = A$

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

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

$AB + AC = A \cdot (B+C)$

$\bar{\bar{A}} = A$

PAGE No.
DATE / /

A	B	A+B	LHS $\overline{A+B}$	$\bar{A}$	$\bar{B}$	RHS $\bar{A}\cdot\bar{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

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

A	B	A·B	LHS $\overline{A\cdot B}$	$\bar{A}$	$\bar{B}$	RHS $\bar{A}+\bar{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

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

vii] $y = A + \bar{A}\cdot B + A\cdot\bar{B}$
$= A + B + A\bar{B}$
$= A + B + A$
$= A + A + B$
$= A + B$

viii] $y = (X+Y)\cdot(X+\bar{Y})\cdot(\bar{X}+Z)$
$= ((X\cdot X) + X\cdot\bar{Y} + Y\cdot X + Y\cdot\bar{Y})\cdot(\bar{X}+Z)$
$= (X + X\cdot\bar{Y} + X\cdot Y + 0)\cdot(\bar{X}+Z)$
$= (X + X(Y+\bar{Y}))\cdot(\bar{X}+Z)$
$= (X + X(1))\cdot(\bar{X}+Z)$
$= (X+X)(\bar{X}+Z)$
$= X(\bar{X}+Z)$
$= X\cdot\bar{X} + XZ$
$= XZ$

iii] $y = \overline{(\overline{A}B + \overline{A} + AB)}$

$= (\overline{\overline{A}\cdot B}) \cdot (\overline{\overline{A}}) \cdot (\overline{AB})$

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

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

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

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

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

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

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

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

$= A \cdot \overline{B}$

$= \overline{\overline{A}(B+1) + AB}$

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

$= \overline{\overline{A}} \cdot \overline{AB}$

$= A \cdot \overline{AB}$

$= A \,[\overline{A} + \overline{B}]$

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

$= A\overline{B}$

iv] $y = \overline{A + \overline{BC}} \cdot (A\overline{B} + \overline{ABC})$

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

$= (\overline{A} \cdot BC)(\cancel{\overline{B}} + (A + \overline{A}) \cdot \overline{B} + \overline{C})$

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

$= \cancel{(\overline{A} \cdot B \cdot C) \cdot (\overline{B \cdot C})}$

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

$=$

$= \overline{A} \cdot BC\,[A\overline{B} + \overline{A} + \overline{B} + \overline{C}]$

$= \overline{A} \cdot BC\,[A\overline{B} + \overline{B} + \overline{A} + \overline{C}]$

$= \overline{A} \cdot BC\,[\overline{B}(A+1) + \overline{A} + \overline{C}]$

$= \overline{A} \cdot BC\,[\overline{A} + \overline{B} + \overline{C}]$

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

$= \overline{A}BC$

v] $y = C(\overline{ABC} + A\overline{B}C)$

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

$= C(\overline{A} + \overline{B} + \overline{C} + A\overline{B} \cdot C) \rightarrow$ $\cancel{Y}$ $\overline{A} \cdot C + \overline{B} \cdot C + C \cdot A \cdot \overline{B}$

$= C(\cancel{C\overline{A}} + \overline{C} + \overline{B}(1 + AC))$

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

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

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

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

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

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

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

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

$= \cancel{C(A + A)}$

$= (\overline{A} + (1 + A) \cdot \overline{B}) \cdot C$

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

Scanned by TapScanner

ii] $y = \bar{A}\cdot\bar{B}\cdot\bar{C}\cdot\bar{D} + \bar{A}\cdot\bar{B}\cdot C\cdot\bar{D} + \bar{A}\cdot B\cdot\bar{C}\cdot\bar{D} + \bar{A}\cdot B\cdot C\cdot\bar{D} + A\cdot\bar{B}\cdot\bar{C}\cdot\bar{D}$

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

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

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

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

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

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

$= \overline{AB}\cdot\bar{C}\cdot\bar{D} + \bar{A}\cdot\bar{D}\cdot C$

$= (\overline{AB}\cdot\bar{C} + \bar{A}\cdot C)\cdot\bar{D}$

$= (\overline{AB} + \bar{C} + \bar{A}\cdot C)\cdot\bar{D}$

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

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

$= \overline{A\cdot B\cdot C}\cdot\bar{D}$

$=$.

$y = \bar{A}\cdot\bar{B}\cdot\bar{C}\cdot\bar{D} + \bar{A}\cdot\bar{B}\cdot C\cdot\bar{D} + \bar{A}\cdot B\cdot\bar{C}\cdot\bar{D} + \bar{A}\cdot B\cdot C\cdot\bar{D} + A\cdot\bar{B}\cdot\bar{C}\cdot\bar{D}$

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

$= \bar{A}\cdot\bar{B}\cdot\bar{D} + \bar{A}\cdot B\cdot\bar{D} + A\cdot\bar{B}\cdot\bar{C}\cdot\bar{D}$

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

$= \bar{A}\cdot\bar{D} + A\cdot\bar{B}\cdot\bar{C}\cdot\bar{D}$

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

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

$= \bar{D}\cdot\bar{A} + \bar{D}\cdot\bar{B}\cdot\bar{C}$

Q. $(1011011 \cdot 00101101)_2$

→ ~~10110~~ $\underline{101}\,\underline{1011}$

$(5B \cdot 2C)_{16}$

Q. 15.4 to IEEE 754

→ $0.4 \times 2 = 0.8$
$0.8 \times 2 = 1.6$
$0.6 \times 2 = 1.2$
$0.2 \times 2 = 0.4$
$0.4 \times 2 = 0.8$

$(15.4)_{10} \rightarrow (1111.0110)_2$

1111.010
1.1110110×2^3
$1.M \times 2^E$

$E = 3$
$BE = 3 + 7F$
$= (82)_{16}$

$BE = (1000\ 0010)_2$
$M = 1110110$

$\underline{0}\ \underline{1000}\ \underline{0010}\ \underline{1110}\ \underline{1100}$ .....

$= (41760000)_{16}$

Scanned by TapScanner

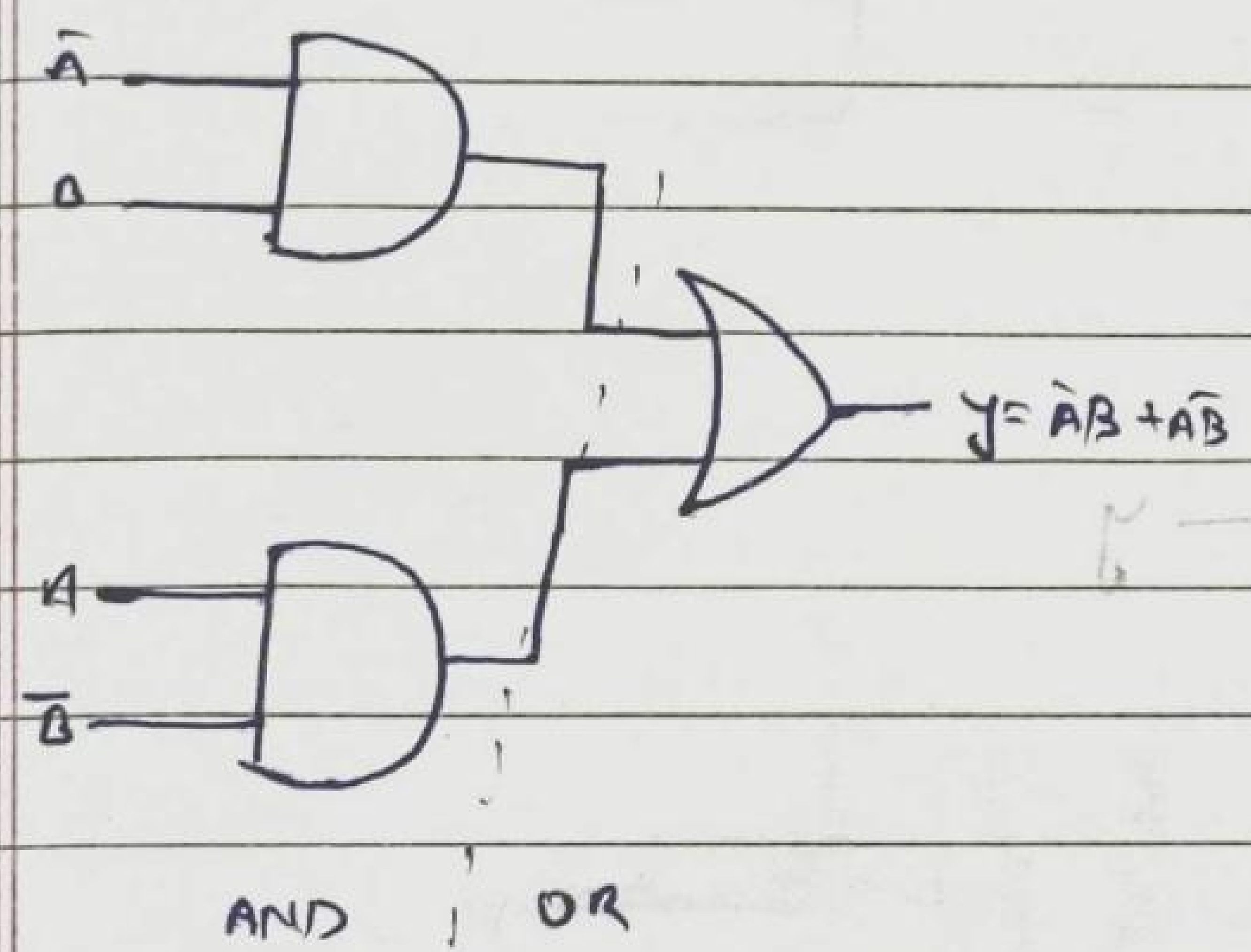
* Standard form of Boolean expression.

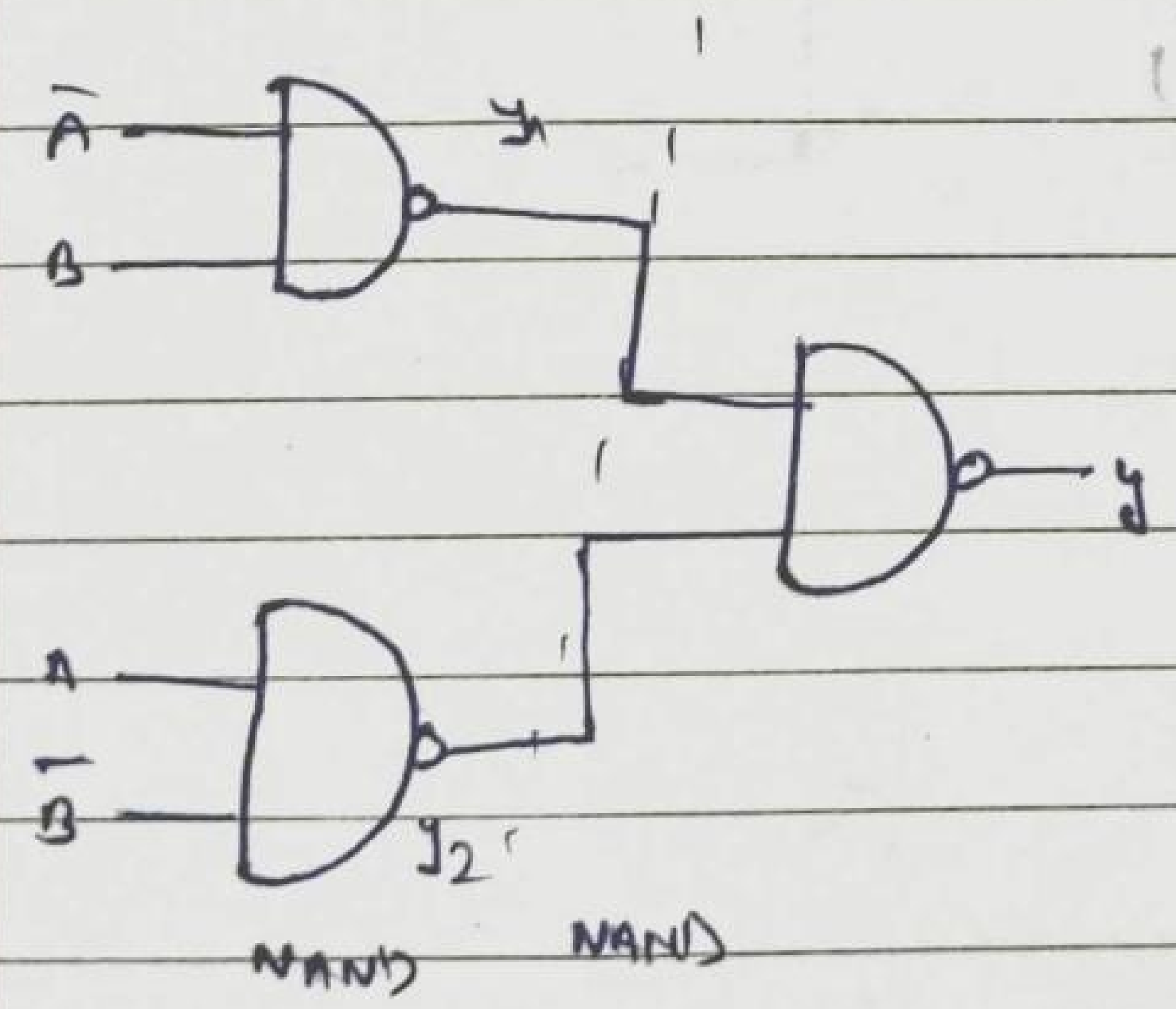
(SOP) Sum of Product

(POS) Product of Sum.

i] $y = \bar{A}.B + A.\bar{B}$

ii] $y = (\bar{A}+B).(A+\bar{B})$

i] 



$y_1 = \overline{\bar{A}.B}$; $y_2 = \overline{A.\bar{B}}$

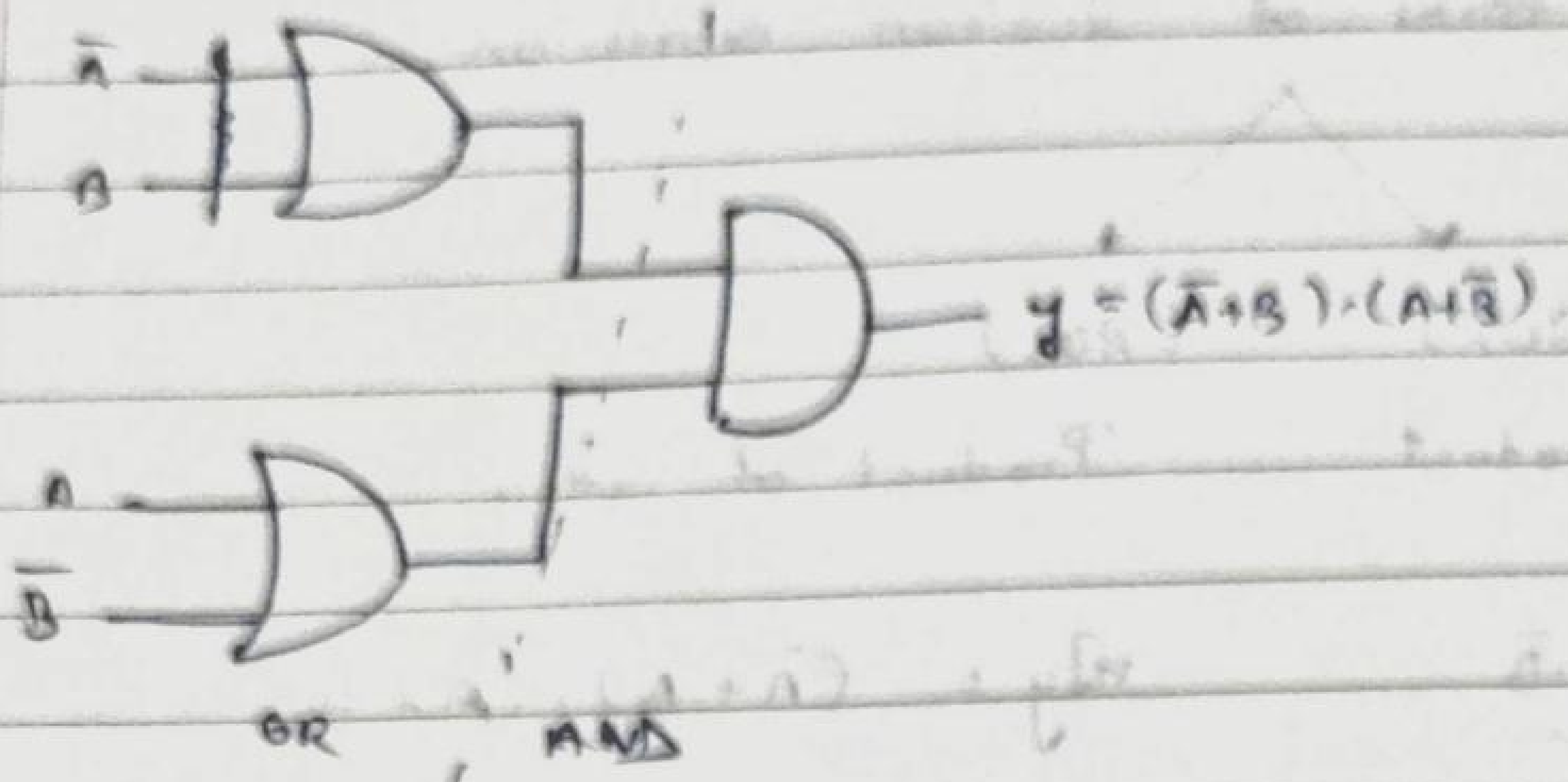
$y = \overline{y_1 . y_2}$

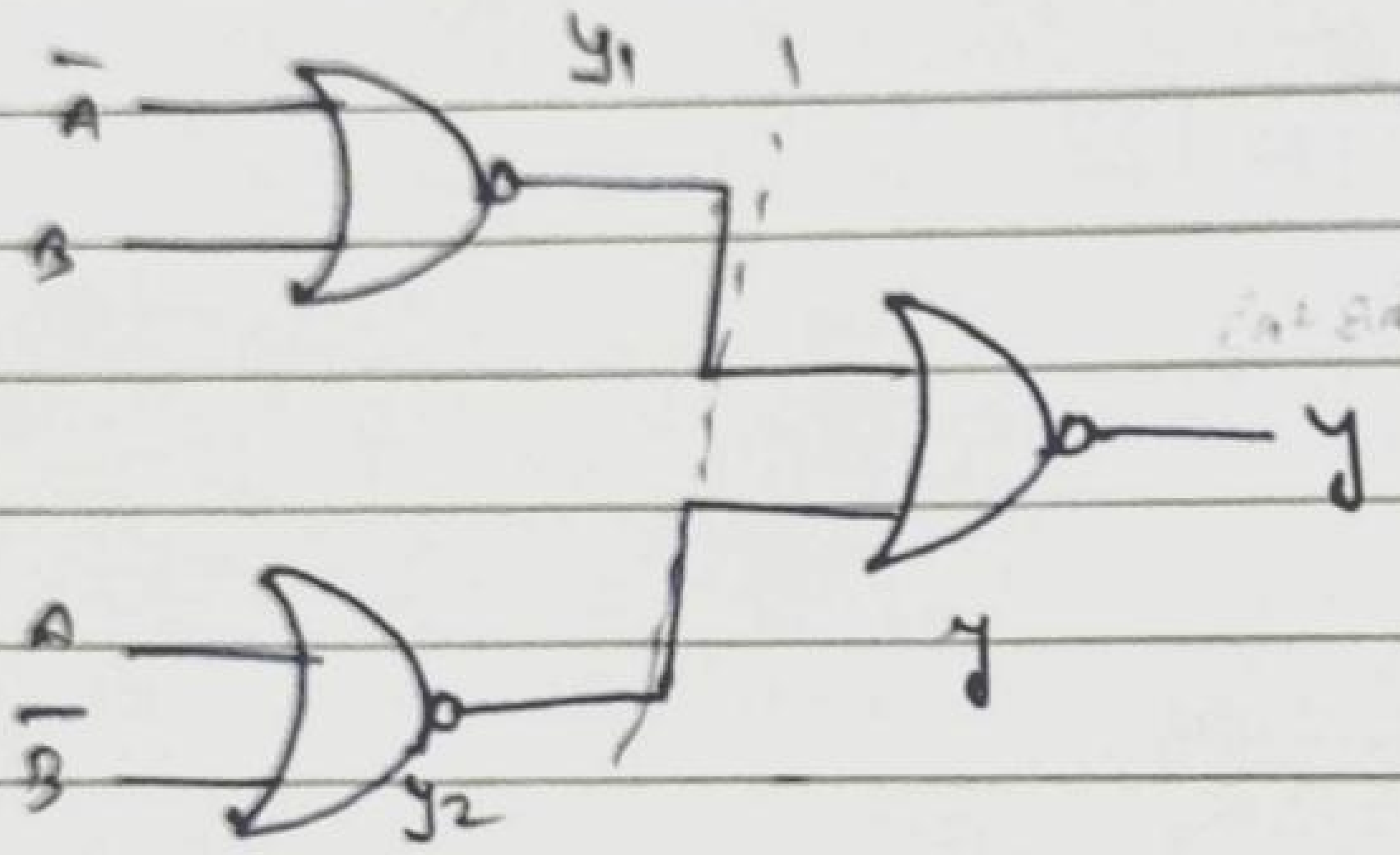
$y = \overline{(\overline{\bar{A}.B}).(\overline{A.\bar{B}})}$

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

Prove AND/OR = NAND/NAND

ii)





$y_1 = \overline{(\bar{A}+B)}$; $y_2 = \overline{(A+\bar{B})}$

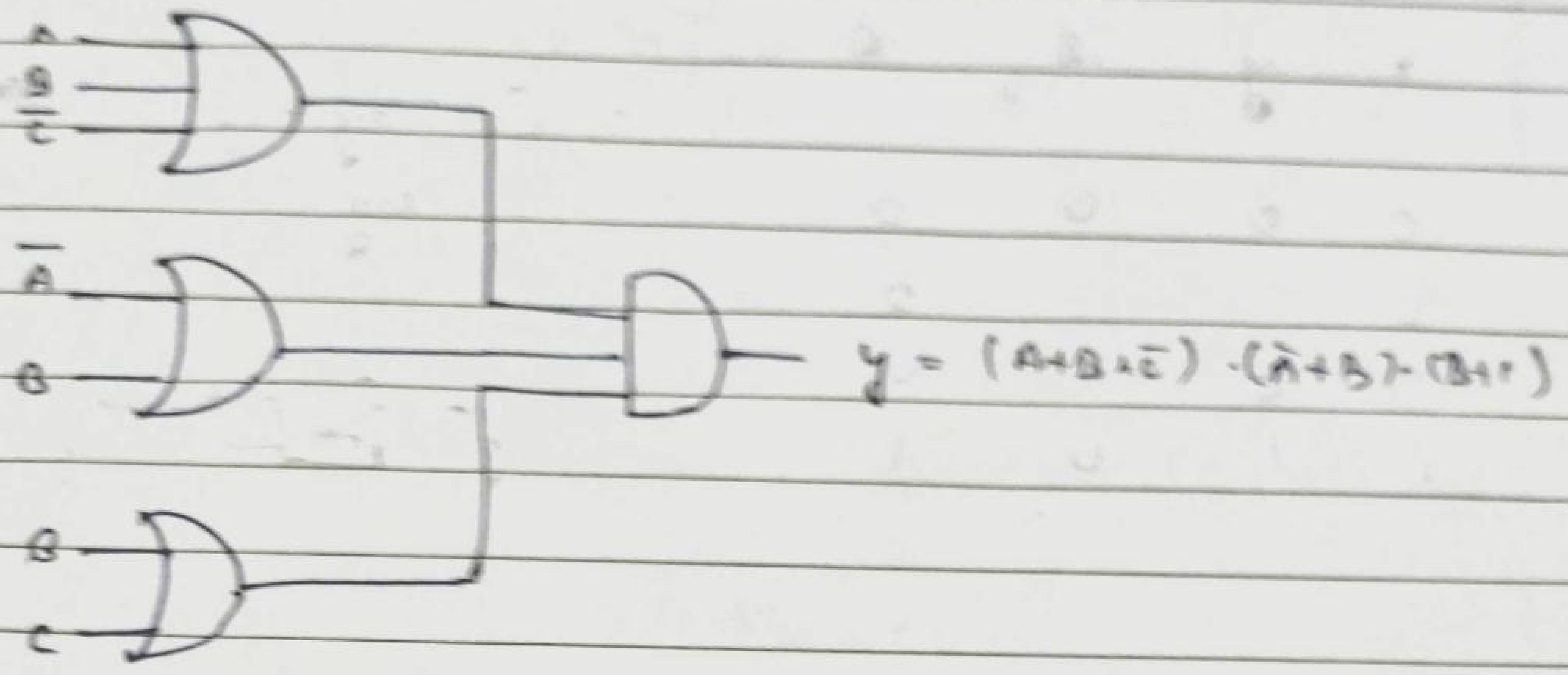
$y = \overline{y_1 + y_2}$

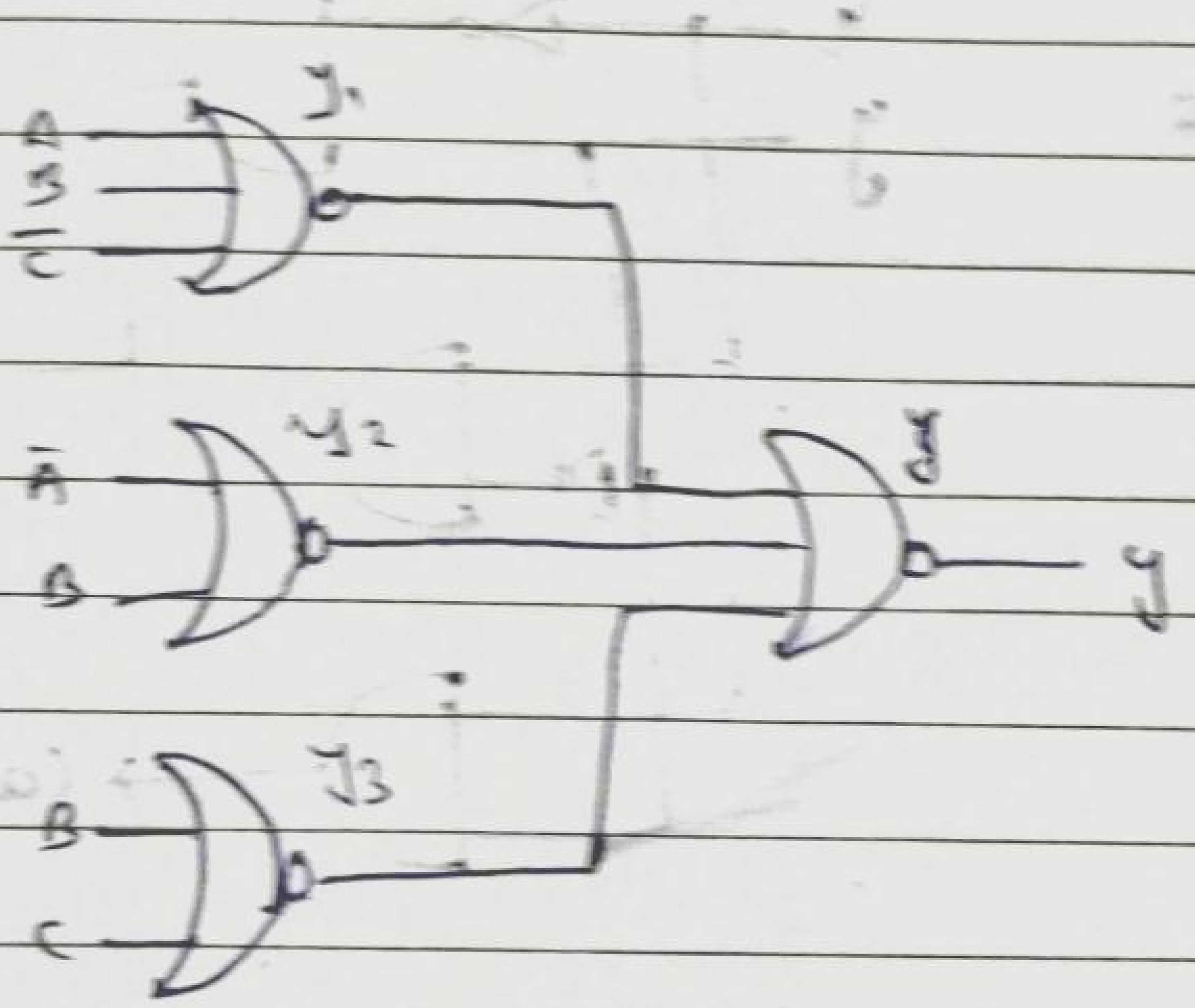
$= \overline{\overline{(\bar{A}+B)} + \overline{(A+\bar{B})}}$

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

Prove OR / AND = NOR / NOR

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





$y_1 = \overline{(A+B+\bar{C})}$; $y_2 = \overline{(\bar{A}+B)}$; $y_3 = \overline{(B+C)}$

$y = \overline{y_1+y_2+y_3}$

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

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

Prove OR/AND = NOR/NOR