Computer Organization

UNIT-1

Basic structure of Computers :-

Computer Types, Functional Unit, *Basic operational concepts, * Bus structures.

Data Representation :-

*Data Types, *complements, Fixed point Representation, * Floating point Representation, other Binary Codes, Error Detection codes.

Computer Arithmetic :-

Addition and Subtraction, * Multiplication Algorithms, *Division Algorithms, Floating point Arithmetic operations.

UNIT-2 :-

Register Transfer Language and Micro-operations :-

Register Transfer Language, Register Transfer Bus and Memory Transfers, Arithmetic Micro-operations, logic Micro-operations, shift Micro-~~org~~ operations, Arithmetic logic shift unit.

UNIT-3 :-

Basic organisation and Design :-

Instruction Codes, Computer Register Computer Instructions, Timing and Control, Instruction cycle, Memory - Reference Instructions, Input-output and Interrupt, Design of Basic Computer, Design of Accumulator logic.

UNIT-4 :-

Central Processing Unit :-

General Register organization, Stack organization Instruction Formats, Addressing Modes, ~~Data~~ Data Transfer and Manipulation, program Control, Reduced Instruction set computer.

Micro Programmed Control :-

Control Memory, Address sequencing, Micro program Example, Design of Control.

UNIT-5 :-

The Memory System :-

Memory Hierarchy, Main Memory, Auxiliary Memory, Associative Memory, Cache

Memory, Virtual Memory.

Input-output Devices :-

peripheral Devices, Input-Output Interface, Asynchronous Data Transfer, Modes of Transfer, priority Interrupts, Direct Memory Access.

Multiprocessors :-

Introduction, characteristic of Multi processors, Interconnection ~~system~~ structure, Inter processor Arbitration.

(1) Define Computer Architecture?

(A) Computer architecture is a set of rules & methods that describe the functionality, organization and Implementation of computer systems.

The architecture of a system refers of its structure in terms of separately specified Components of that system & their inter relationships.

(2) Define Computer Organization?

(A) The computer organization is concerned with the structure and behaviour of digital computers and behaviour of a computer System as seen by the user. It acts as the interface b/w hardware & software. It deals with the components of a connection in a system.

(3) Define Complements? Discuss 1's complement, 2's complements, 9's complement & 10's Complement.

(A) The number derived by subtracting a number from a base number. Complements are used in the digital computers in order to simply if the subtraction operation and for the logical manipulations.

One's Complement:- The one's complement of a binary number is defined as the value obtained by inverting all the bits in the binary representation of the number [swapping 0's for 1's and vice-versa].

Ex:- 00110010 = 50

1's complement of 50 is 11001101

Two's Complement:-

The two's complement of a binary number is defined as the value obtained by adding 1 to one's complement.

2's complement = (1's complement) +1

Ex:- 50 = 00110010 1's comp - 11001101

2's comp - 1's comp
+1

$$\begin{array}{r} 11001101 \\ +1 \\ \hline 11001110 \end{array}$$

9's complement :- The 9's complement of a number is calculated by subtracting each digit of the number by 9.

Ex:- 9's comp of 1423

$$\begin{array}{r} 9999 \\ -1423 \\ \hline 8576 \end{array}$$

10's Complement :- The 10's complement of a numbes is calculated by subtracting each digit by 9 and then adding 1 to the result.

Ex:- 10's comp of 1423 is

$$\begin{array}{r} 9999 \\ -1423 \\ \hline 8576 \end{array}$$

$$\begin{array}{r} 8576 \\ +1 \\ \hline 8577 \end{array}$$

UNIT-1

BASIC STRUCTURE OF COMPUTERS

Computer Types :-

Digital Computer (or) a Computer is a fast calculating machine that accepts digitized coded input information process that stored information produces results as output.

* program is a set of instructions [stored] which are stored in a memory with specified address.

* we broadly classify computer types as follows:

(1) personal computer / desktop computer.

(2) portable notebook computer.

(3) work stations

(4) Enterprise system (or) Mainframes (or) Servers

(5) Super computers.

* personal computer :-

processor, storage unit, visual display unit, audio video output unit, keyboard unit etc. storage media.

* work stations :- It is also a single user

computer system, similar to personal computer however has a more powerful processor.

micro

work station Computers used for scientific or technical calculations [high end computers].

Main Frames :- [low-end computers]

Main Frame is a very large in size & is an expressive computer capable of supporting hundreds (or) even thousands of users simultaneously. Main Frame executes many programs concurrently. Generally used in applications as pay roll Computation accounting, business transactions, Engineering and Scientific computations.

* Super Computers :-

Super Computers are one of the fastest computer currently available.

very expensive & are empolyed for specialized application that require immense amount of mathematical calculations. Generally used for wheather forecasting, aircraft design, scientific simulations, (animated) graphics, fluid dynamic calculations, nuclear energy research,

electronic design and analysis of geological data.

* Difference b/w Mainframe and Super Computer and Workstation :-

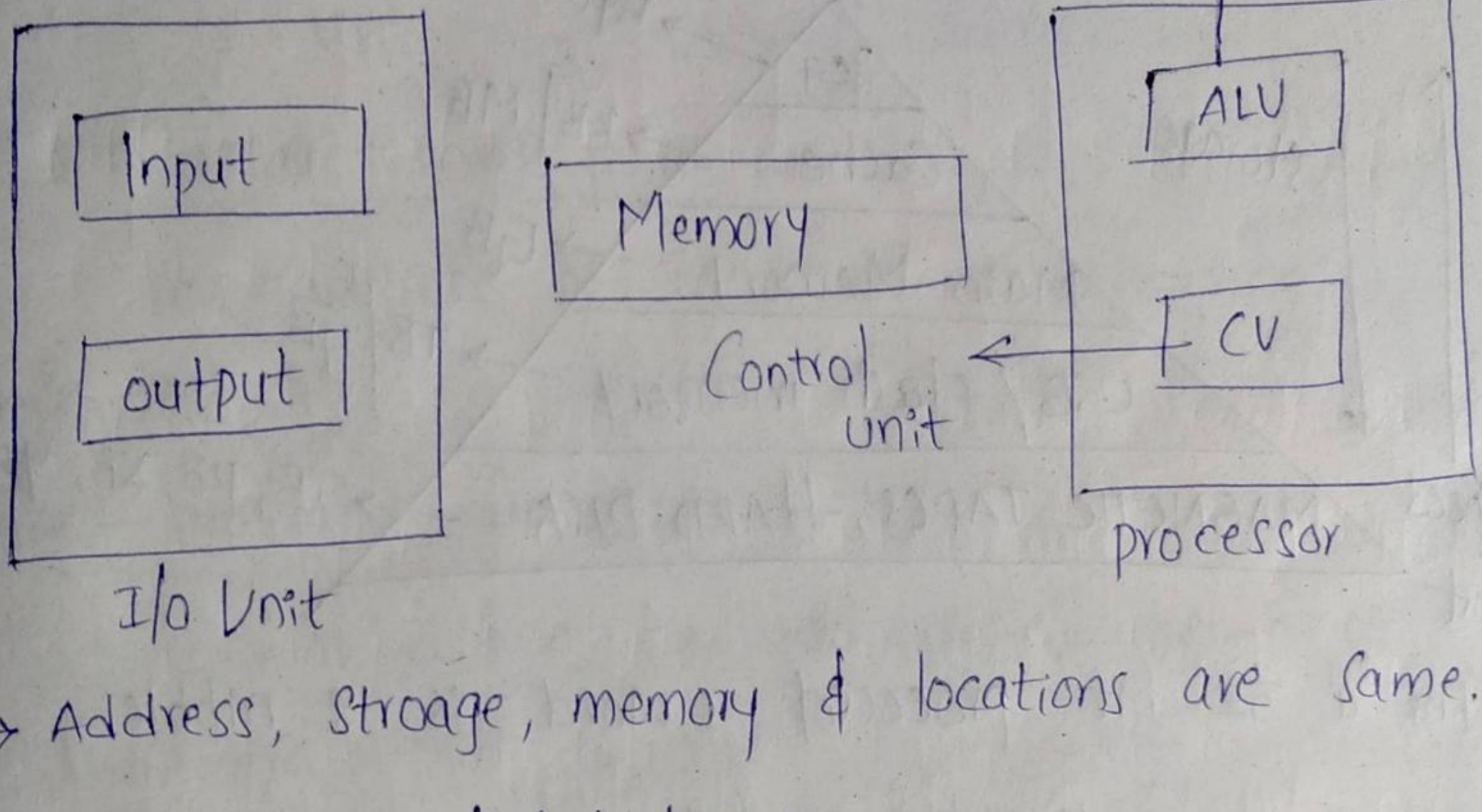
Super ~~Work~~ Computer	Mainframe Computer
1. Super Computers are used for large & complex mathematical computations.	1. Mainframe Computers are used to as a storage for large database and servel as a maximum no. of users simultaneously.
2. Super computer's speed is more than mainframe computer.	2. Mainframe Computer's Speed is comparatively less than Super Computel.
3. It Can execute billions of instructions within a second.	3. Millions of instructions are executed simultaneously.
4. largest computer	4. Smaller than Super Computer size.
5. Heavy / high cost	5. less cost than Super computer.
6. These have linux &	6. These have multiple

their operating systems.	operating systems.
7. These are mostly purpose built for one or a few specific institutional tasks.	7. These are built to handle a large variety of tasks.
8. Seymour Cray invented the this computer.	8. Invented by IBM.
9. These can have a processing Speed in the range of 100 to 900 MIPS.	9. Range of 3-4 MIPS to as high as 100 MIPS.

* Workstation Computer :-

Desktop Computer, normally more powerful than a normal pc and often dedicated to a specific task, such as an area, at a workplace, for a single worker.

Functional Units



→ Address, Stroage, memory & locations are same.

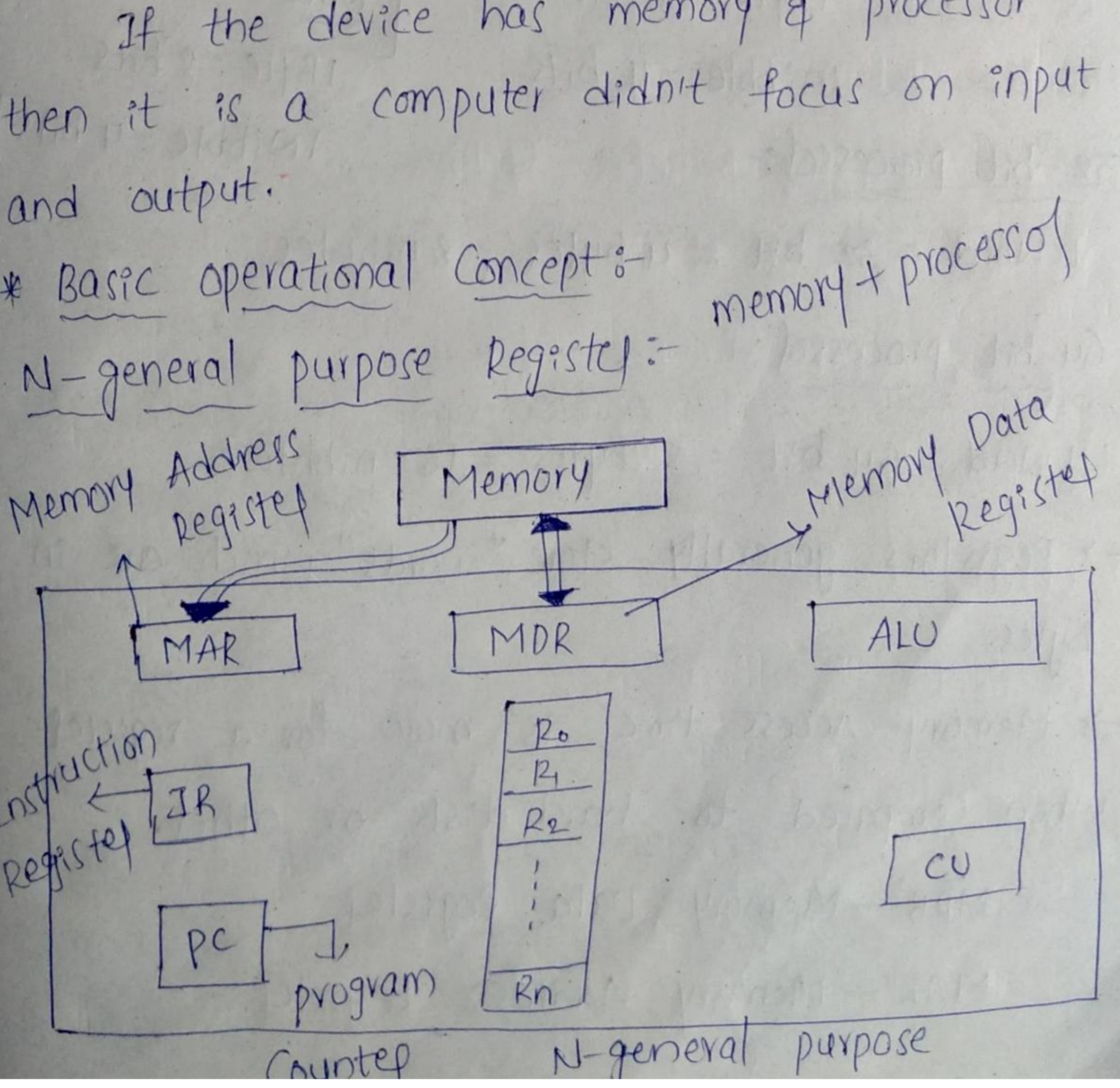
* Von-heumon Architecture :-

If the device has memory & processor then it is a computer didn't focus on input and output.

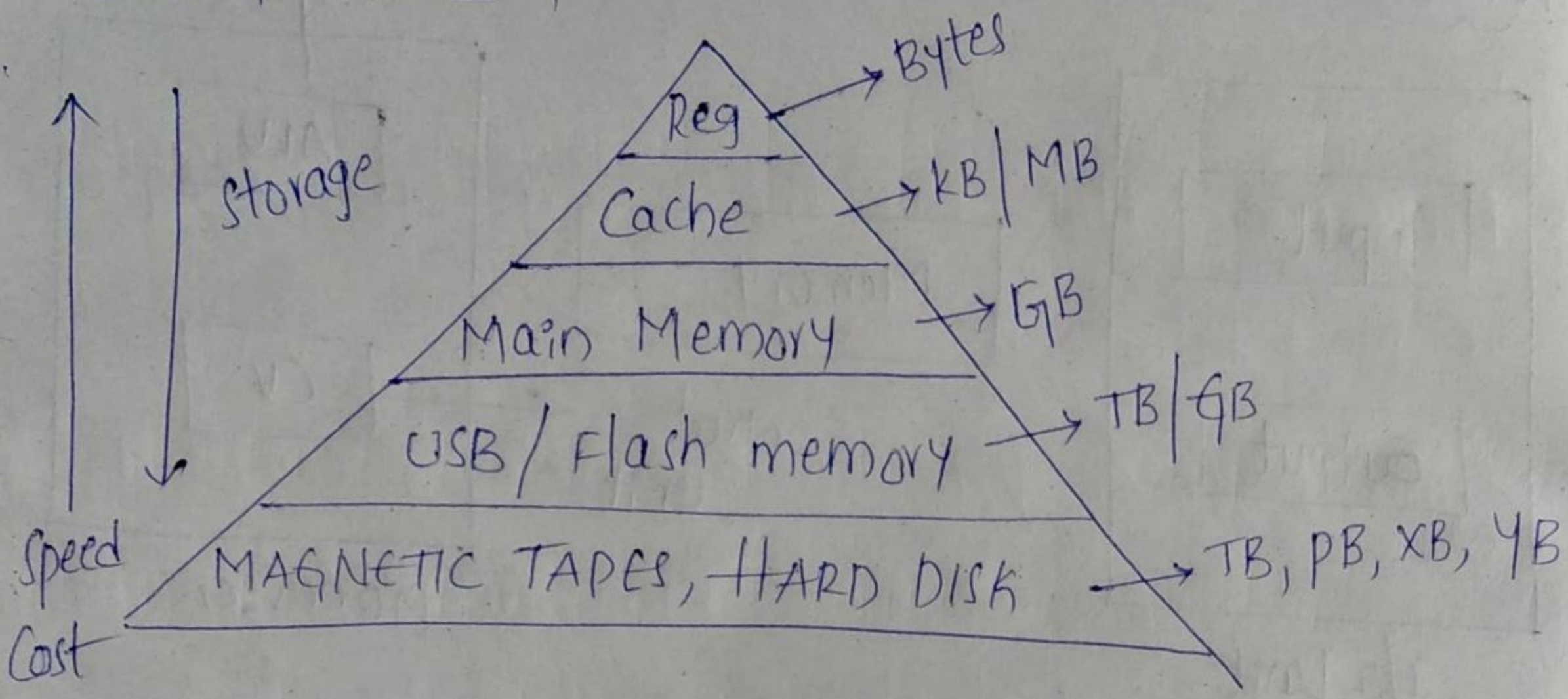
* Basic operational Concept :-

memory + processor

N-general purpose Registey :-



* Memory Hirarchy :-



For a 32-bit processor 1 word is 8 nibble.

1 Word = 32 bit

1 byte = 8 bits

1 nibble = 4 bits

[Note :-
1 BYte = 8 bits
1 Nibble = 4 bits]

32-bit processor :-

1 word = 32 bit = 4 bytes = 8 nibbles

64-bit processor :-

1 word = 64 bit = 8 bytes = 16 nibbles.

→ Registors generally store "words" which are in bytes.

→ Memory Access time is more for a registel when Compared to hard disk or othel.

MDR - Memory Data Registel

MAR - Memory Address Registel.

* MDR is used for storing memory.
* MAR is used for storing Address.
* IR - Instruction Register :-

The address of the current executing instruction.

* PC - program Counter :-

It will define the address of the next instruction which is yet to execute.

(i)

```
ADD R0, R1
```

(ii)

```
mov LOC A, R0
ADD R0, R1
```

(iii)

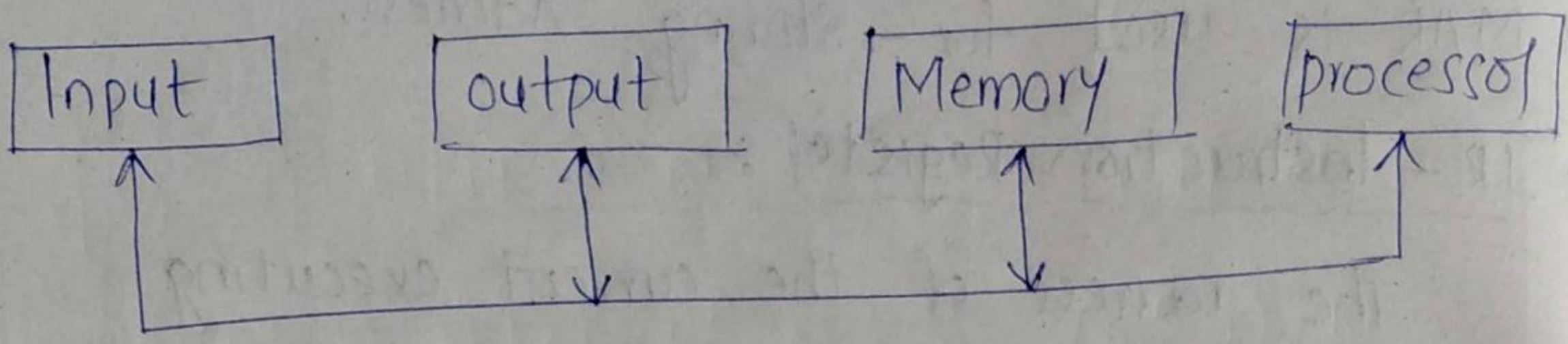
```
mov LOCA, R0
mov LOC B, R1
ADD R1, R0
```

(i) Content of R_0 & R_1 are added, the result is stored in R_1.

(ii) Content in the location A is copied to R_0 and the content of R_0 & R_1 are added and the result is stored in R_1.

(iii) Content in the location A is copied to the R_0 & the content in the location B is copied to the R_1 and the contents of both R_0 and R_1 are added and result is added to the R_0.

Bus :-



Single Bus :- Transfer of data in only one way.

Mutti Bus :- Transfer of data in multiple ways.

* Data Representation :-

Data Types :- Numbers, Alphabets, Special symbols, characters.

Number system :-

(1) $r=2$, Binary Number system

(2) $r=10$, Decimal Number system

(3) $r=8$, Octal Number system

(4) $r=16$, Hexa Decimal Number System.

* $(FED)_{16} = (\quad)_{10}$

$(15\ 14\ 13)_{16} = (\quad)_{10}$

$15 \times 16^2 + 14 \times 16^1 + 13 \times 16^0$

$= 15 \times 256 + 14 \times 16 + 13$

$= 3,840 + 224 + 13 = (4077)_{10}$

* $(4077)_{10} = (\quad)_2$

$= (111111011101)_2$

* $(111111011101)_2 = (\quad)_8$

$(\underline{111}\ \underline{111}\ \underline{101}\ \underline{101})_2$

$(7755)_8$

* $(A05H1)_{16} - (\quad)_{10}$

B

$(10\ 0\ 5\ 11\ 1)_{16}$

$10\times16^4 + 0\times16^3 + 5\times16^2 + 11\times16^1 + 1\times16^0$

$655360 + 0 + 1280 + 176 + 1$

$= 6,56,817$

$(656817)_{10}$

* $(656817)_{10} = (\quad)_2$

$(10100000011110110001)_2$

$(\underline{10}\ \underline{100}\ \underline{000}\ \underline{011}\ \underline{110}\ \underline{110}\ \underline{001})_2$

$(2403661)_8$

2 | 4077
2 | 2038 - 1
2 | 1019 - 0
2 | 509 - 1
2 | 254 - 1
2 | 127 - 0
2 | 63 - 1
2 | 31 - 1
2 | 15 - 1
2 | 7 - 1
2 | 3 - 1
1 - 1

2 | 656817
2 | 328408 - 1
2 | 164204 - 0
2 | 82102 - 0
2 | 41051 - 0
2 | 20525 - 1
2 | 10262 - 1
2 | 5131 - 0
2 | 2566 - 1
2 | 1283 - 1
2 | 641 - 1
2 | 320 - 1
2 | 160 - 0
2 | 80 - 0
2 | 40 - 0
20 - 0

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

* Complements :-

r's	(r-1)'s
$r : [(r^n - 1) - N] + 1 = r^n - N$	$(r^n - 1) - N$
$r = 10 : [(10^n - 1) - N] + 1 = 10^n - N$	$(10^n - 1) - N$
$r = 2 : [(2^n - 1) - N] + 1 = 2^n - N$	$(2^n - 1) - N$
2's complement	1's complement
10's Complement	9's complement

N = Number in number system

n = no. of digits in N

r = radix (or) Base

* $(3426079 78)_{10}$ n=9, r=10

$[(10^9 - 1) - 342607978] + 1 = 10^9 - 342607978$

$[999999999 - 342607978] + 1 = 1000000000 - 342607978$

$657392022 = 657392022$

9's Complement :-

$(10^9 - 1) - N$

$999999999 - 342607978$

$= 657392021 + 1 = 657392022$

* Subtraction of unsigned numbel :-

The direct method of subtraction uses the borrow's concepts. The subtraction of 'n' digit unsigned numbel (M-N) N ≠ 0 in base are can be done as follows:-

* → Add the minend 'm' to the r's complement of the subtracted n. This performs

$$(m + r^n - N) = M - N + r^n$$

→ If M >= N the sum will produce an end carry r^n which is discarded and what is left in result M-N.

→ If M < N the sum doesn't produce an end carry is equal to $r^n - (N-M)$ which is the r's complement of (N-M).

For example:-

72532 - 13250

The 10's complement of 13250 is

99999 - 13250 = 86749 + 1 = 86750

M = 72532

10's complement N = 86750

```
  1
 72532
 86750
------
159282 ✓
```

discard end Carry $10^5 = 100000$

Answer - 59282

For eg:- M<N i.e, 13250 - 72532
M N

Produces negative 59282

M = 18250

10's comp N = 27468

Sum = 40718

There is no end carry. Answer is negative 59282 = 10's Complement of 40718.

99999
40718
59281
+1
59282

* Binary Numbers:-

X = 1010100 & Y = 1000011

X = 1010100

2's Complement → Y = 0111101

X-Y

Sum = 10010001

Discard Carry 2^x = -10000000

0010001

* Fixed point Representation:-

There are two ways; we are going to represent the position if the binary point in a registel.

(i) Fixed position.

(ii) Floating point Representation.

* Fixed point method assumes that the binary point is always fixed in one position.

The two positions most widely used as

(1) A binary point in one extreme left of the register to make the stored numbel a fraction.

(2) A binary point in the extreme right of the register to make the stored numbel as an integels.

Integel Representation:- +ve = 0, -ve = 1 } sum bit

when an integel binary number is +ve the sign is represented by 0 and the

magnitude by a +ve binary number when an integer binary number when an integer binary number is negative the sign is represented by '1' but the rest of the number may be represented is one of three possible ways.

(1) Signed magnitude Representation.

(2) Signed - 1's Complement

(3) Signed - 2's Complement

The signed magnitude representation of a negative number consists of the magnitude and a negative sign. Remaining 2, we are going to perform either 1's or 2's Complement.

For example:-

+14 = 00001110 — 8 bit Register

left most bits are zero is 8 bit registel.

*14 can be represent in 3 ways.

→ The signed magnitude representation of -14 is obtained from +14 by complementing only the sign bit [-14 = 10001110] → sign

magnitude.

→ The signed 1's complement Representation of -14 is obtained by Complementing are all the bits of +14, including the sign bit

[-14 = 11110001]

→ The signed 2's complement is obtained by taking the 2's complement of the positive number, including its sign bit.

[-14 = 11110001 + 1 = 11110010]

* Arithmetic Addition:-

→ The addition of 2 numbers is the signed magnitude system borrows the roles.

→ If the signs are the same, we add the 2 magnitudes are give the sum the Common sign.

→ If the signs are different, we subtract the smaller magnitude from the larger and give the result the sign of the larger.

+6 = 00000110

+13 = 00001101

+19 = 00010011

+6 = 00000110

-13 = 11110011

-7 = 11111001

-6 = 11111010

+13 = 00001101

+7 = 00000111

-6 = 11111010

-13 = 11110011

-19 = 11101011

In each and every case, we are going to perform only addition.

If there is any negative value, the total result will be the 2's complement value.

* Arithmetic Subtraction:-

Subtraction of two signed binary numbers when negative numbers are in 2's complement form is very simple.

→ Take the 2's complement of the subtraction (concluding the sign bit) and it to the minuend (including the sign bit). A carry out of the sign bit

position is discarded.

→ A subtraction operation can be changed to an addition operation is the sign of the subtracted is changed.

This can be demonstrated through the relationship.

$$(\pm AB) - (+B) = (\pm A) + (-B)$$

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

But changing a positive number to a negative number is easily done by taking its 2's complement the reverse is also true.

* Floating point Representation :-

Signed + mantisa$_x^s$ = Floating point representation

$[m \times r^e]$

where m = mantisa$_x^s$

e = exponent

r = radix (or) Base.

Ex:- Decimal numbel :- +6132.789 (r=10)

Floating point representation :- 0.6132789×10^4

$[m \times r^e)$

Fraction	exponent
+0.6132789	+04
(m)	(e)

Ex:-2

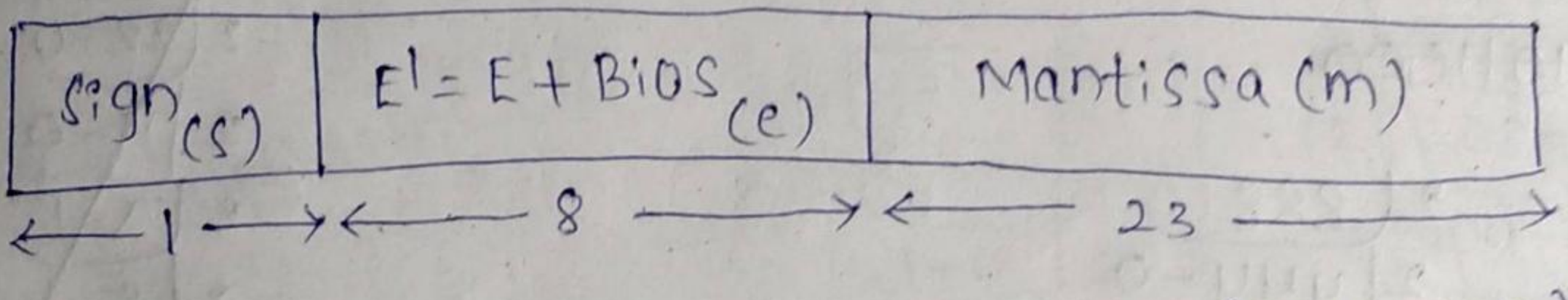
Binary numbel :- +10011.11 (r=2)

Floating point Representation :- $+0.1001111 \times 2^5$

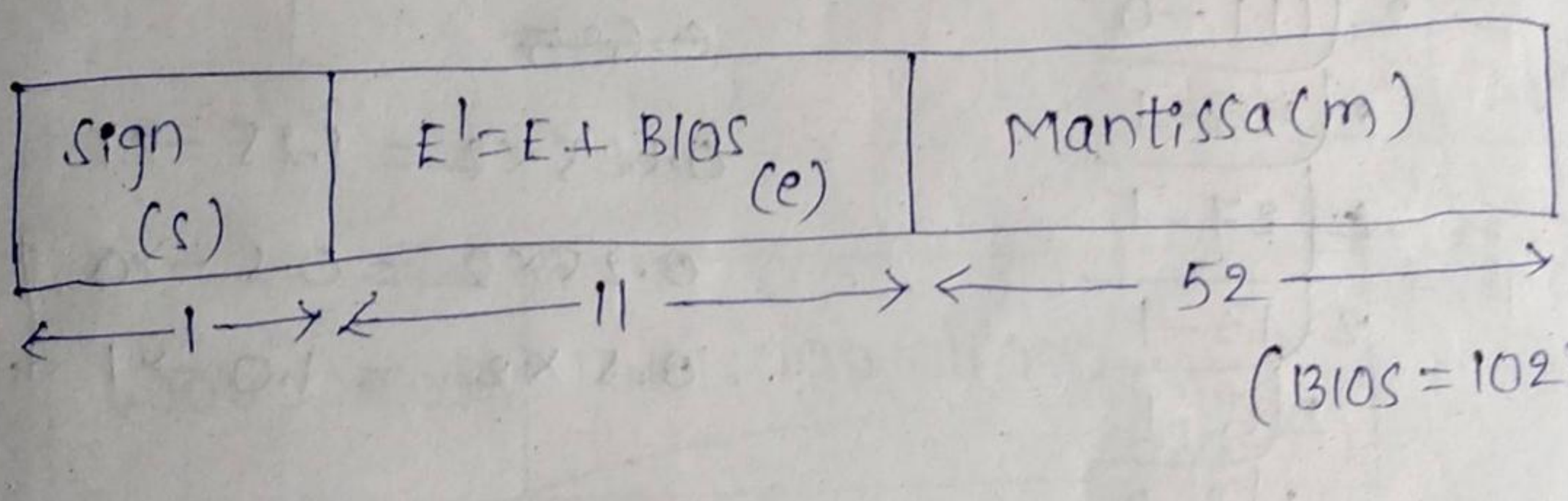
Fraction	exponent
+0.1001111	+05
(m)	(e)

* Two ways :-

1. 32 bit Floating point Representation (single precision)
2. 64 bit Floating point Representation (Double precision).



(BIOS = 127)



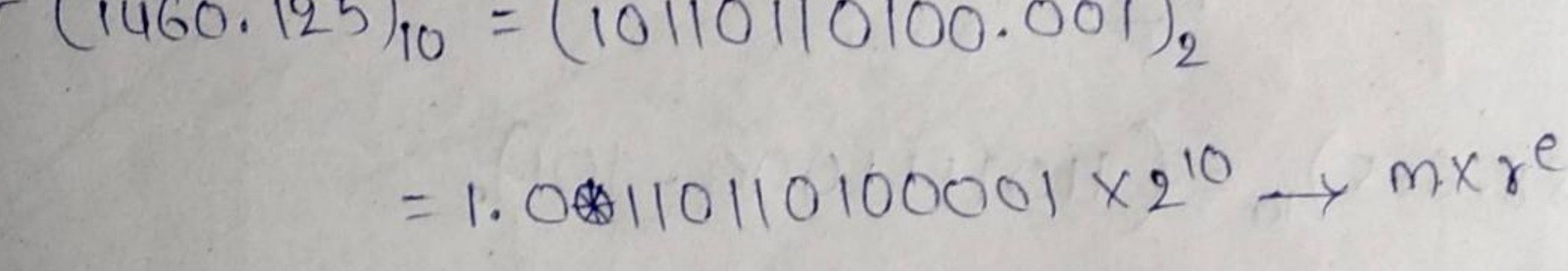
(BIOS = 1023)

Eg :- $(1460.125)_{10} = (10110110100.001)_2$

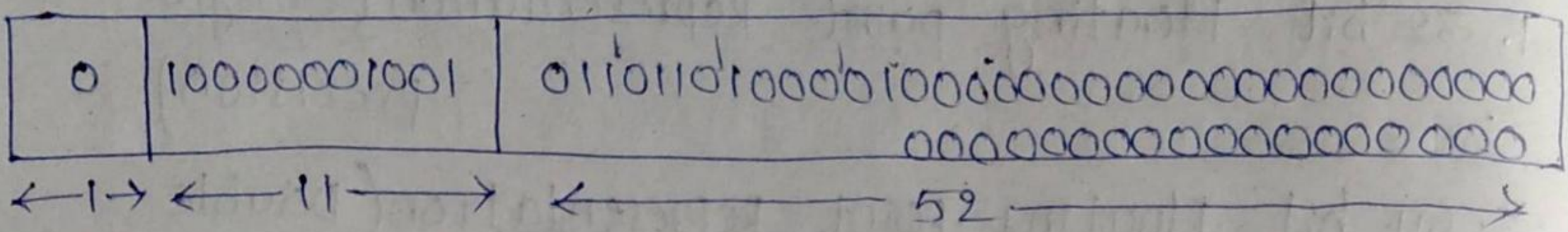
$= 1.0110110100001 \times 2^{10} \rightarrow m \times r^e$

$E' = E + BIOS$

$= 10 + 127 = 137$



$E^1 = E + BIOS = 10 + 1023 = 1033$



* $(888.625)_{10}$

~~$(1011000$~~

2 | 888
2 | 444 - 0
2 | 222 - 0
2 | 111 - 0
2 | 55 - 1
2 | 27 - 1
2 | 13 - 1
2 | 6 - 1
2 | 3 - 0
1 - 1 ↑

2 | 88
2 | 44 - 0
2 | 22 - 0
2 | 11 - 0
2 | 5 - 1
2 | 2 - 1
1 - 0 ↑

~~0.625~~

$0.625 \times 2 = 1.25 \rightarrow 1$

$0.25 \times 2 = 0.5 \rightarrow 0$ ↓

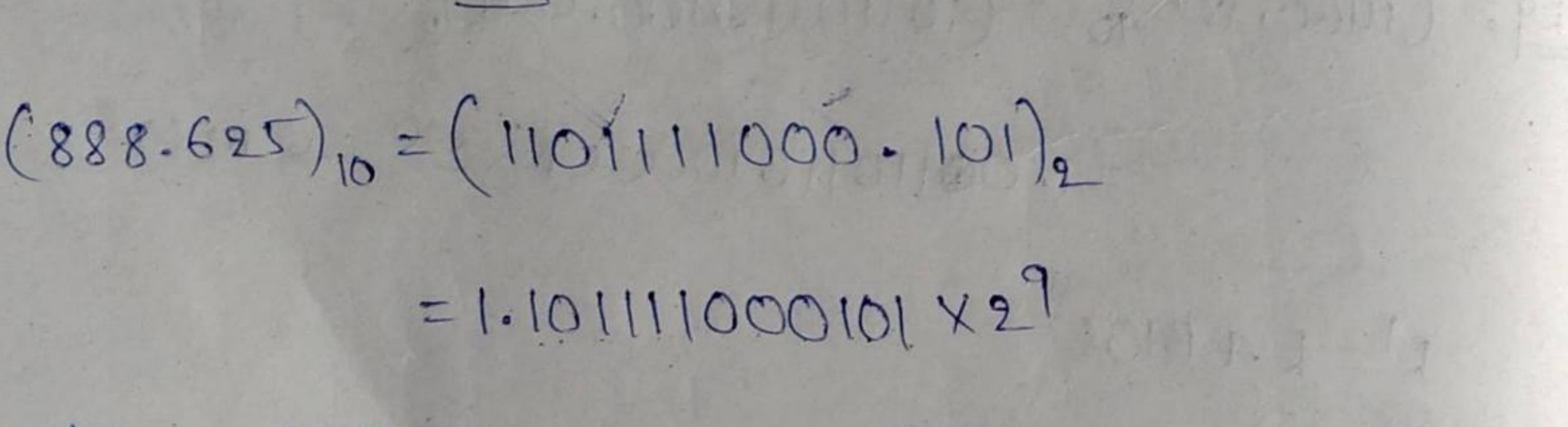
$0.5 \times 2 = 1.0 \rightarrow 1$ ↓

$(888.625)_{10} = (1101111000.101)_2$

$= 1.101111000101 \times 2^9$

$E^1 = E + BIOS$

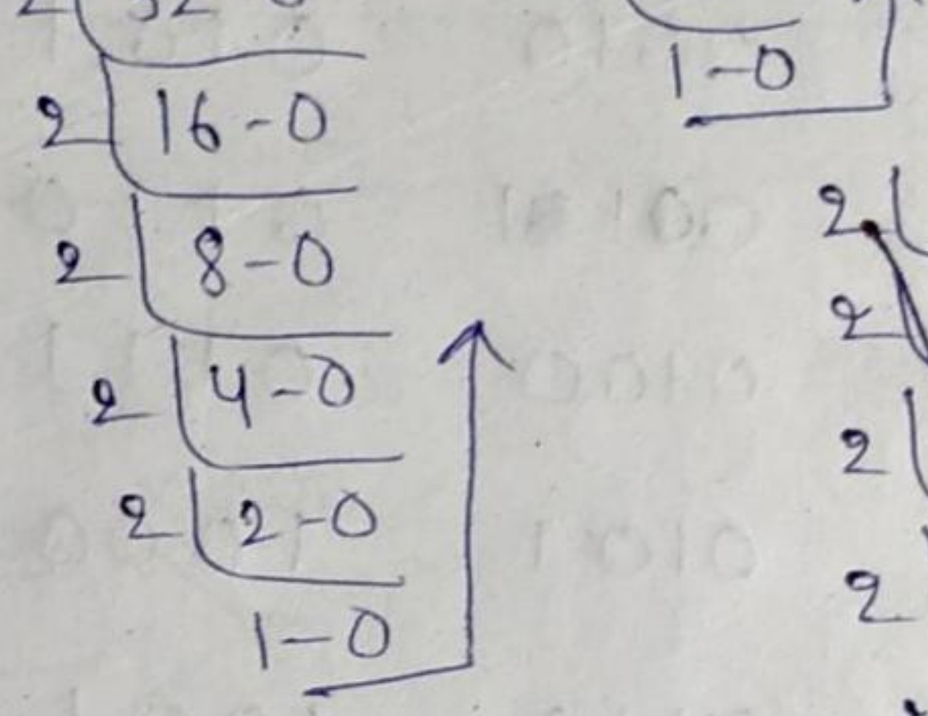
$= 9 + 127 = 136$

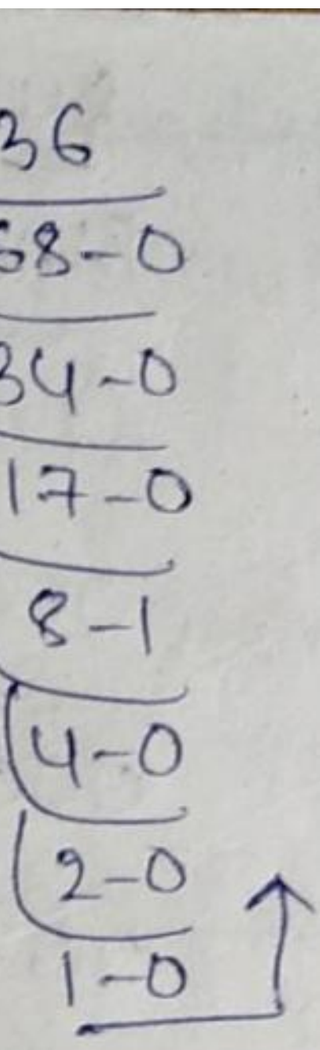


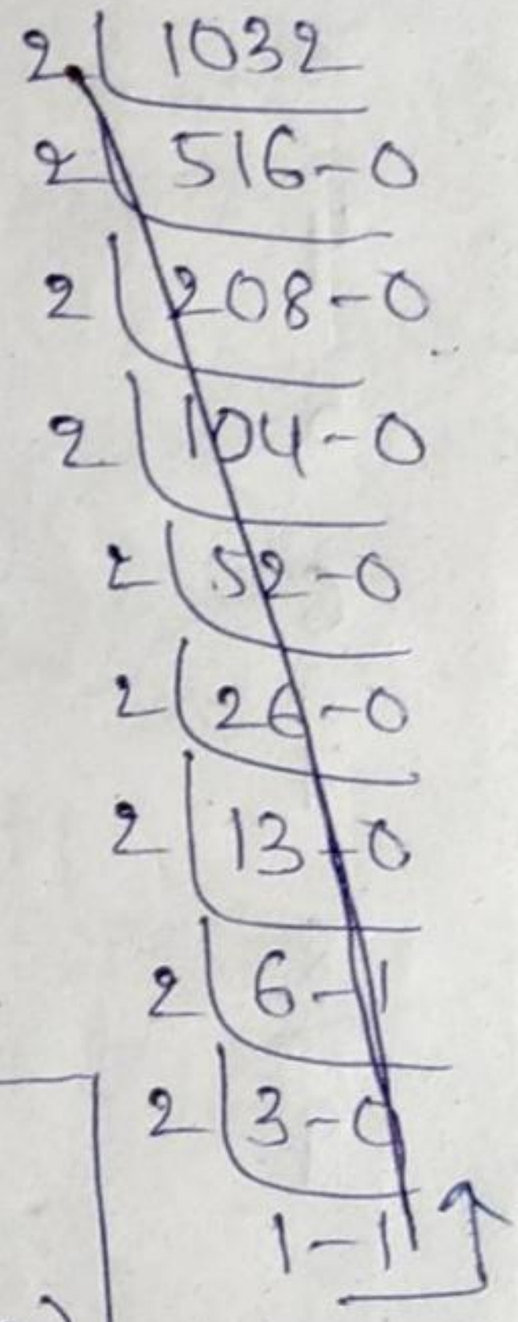
$E^1 = E + BIOS$

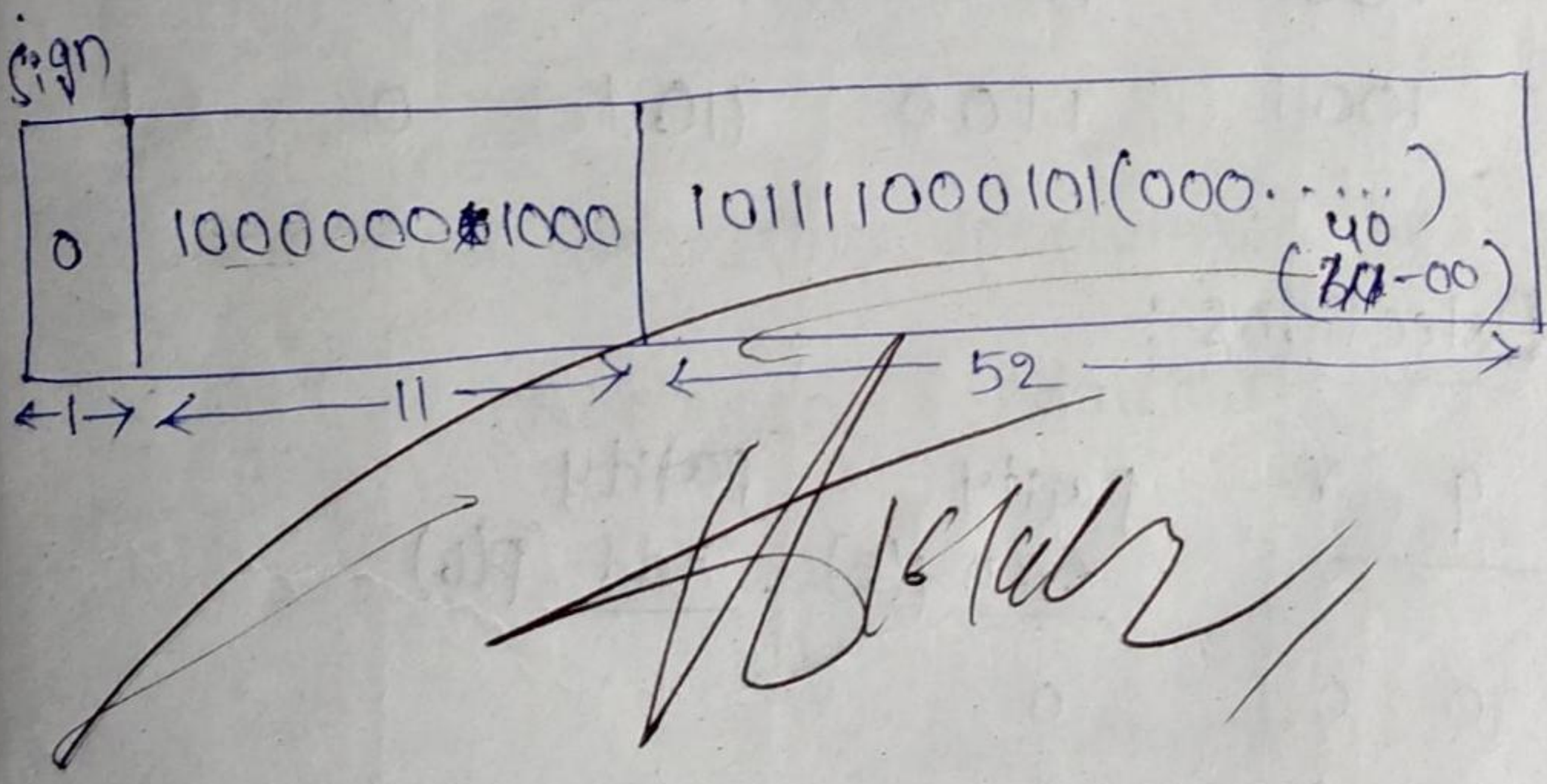
$= 9 + 1023$

$= 1032$









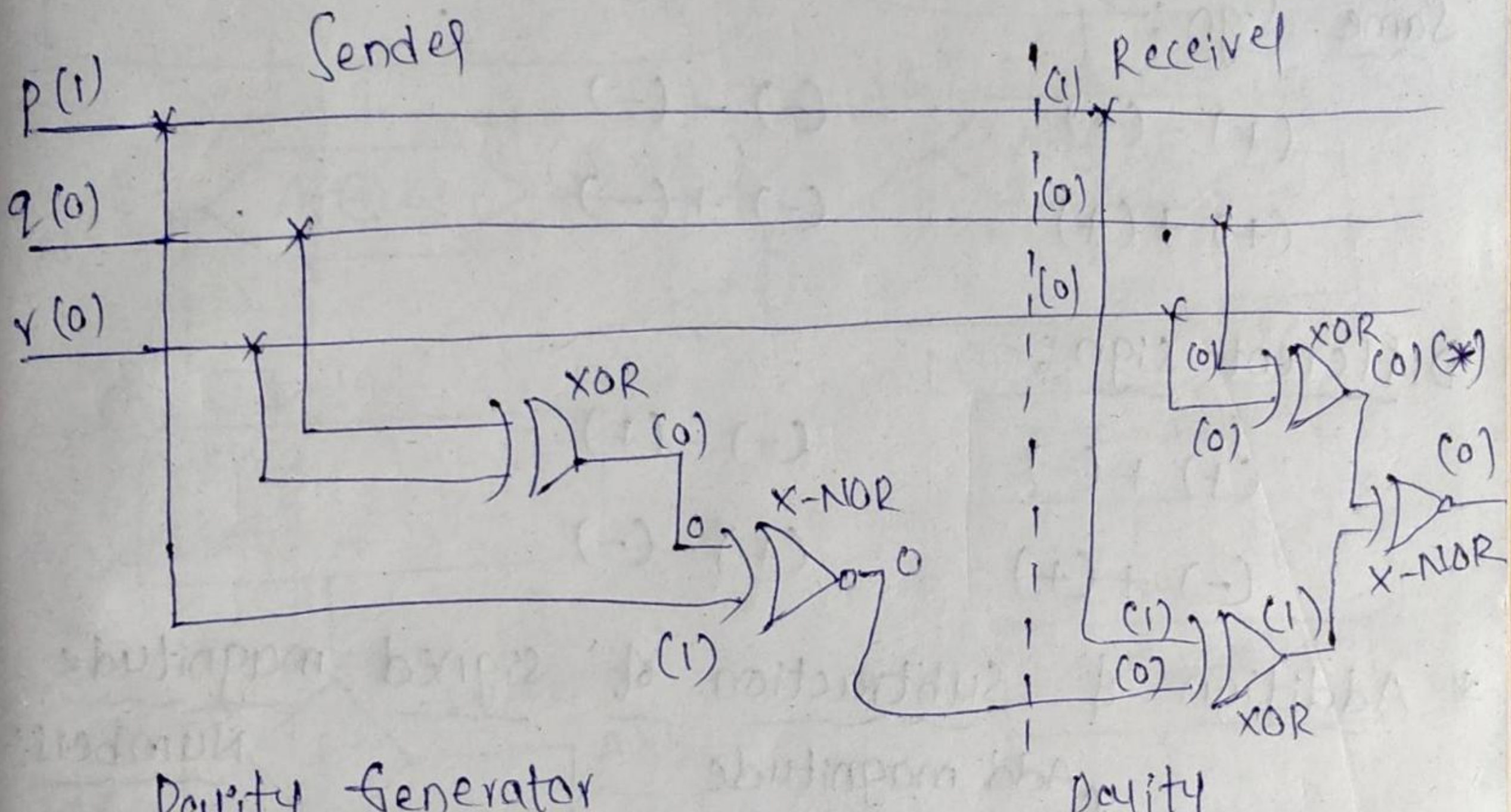
* Decimal	8421 Binary	Excess-3	Gray code	Parity Even	pality odd
0	0000	0011	0000	0	1
1	0001	0100	0001	1	0
2	0010	0101	0011	1	0
3	0011	0110	0010	0	1
4	0100	0111	0110	1	0
5	0101	1000	0111	0	1
6	0110	1001	0101	0	1
7	0111	1010	0100	1	0
8	1000	1011	1100	1	0
9	1001	1100	1101	0	1

* Error Detections :-

P	q	r	pality even P(E)	pality odd P(O)
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

* Odd parity Circuit :- (1) parity ~~sender~~ (Sender) generator

(2) parity checker (Receiver).



* Subtraction operation performs by

Complement + Addition.

$$A - B = A + (\underbrace{\overline{B} + 1}_{\downarrow})$$

2's complement of B.

* Rule 1 :- Same sign → perform add as addition operation, sub as subtraction operation.

* Rule 2 :- Different sign → perform sub for Add, Add for sub.

Same sign :-

(+) - (+) (-) - (-)

(+) + (+) (-) + (-)

Different sign :-

(+) + (-) (-) - (+)

(-) + (+) (+) - (-)

* Addition & Subtraction of signed magnitude Numbers:-

↑ Add magnitude

S.NO	operation	↑	Subtraction Magnitude		
			(i) A>B	(ii) A<B	(iii) A=B
1.	(+A)+(+B)	+ (A+B)	x	x	x
2.	(+A)+(-B)	x	+(A-B)	-(B-A)	+(A-B)
3.	(-A)+(+B)	x	-(A-B)	+(B-A)	+(A-B)
4.	(-A)+(-B)	$\overline{(A+B)}$	x	x	x
5.	(+A)-(+B)	x	+(A-B)	-(B-A)	+(A-B)
6.	(+A)-(-B)	+ (A+B)	x	x	x
7.	(-A)-(+B)	$\overline{(A+B)}$	x	x	x
8.	(-A)-(-B)	x	-(A-B)	+(B-A)	+(A-B).

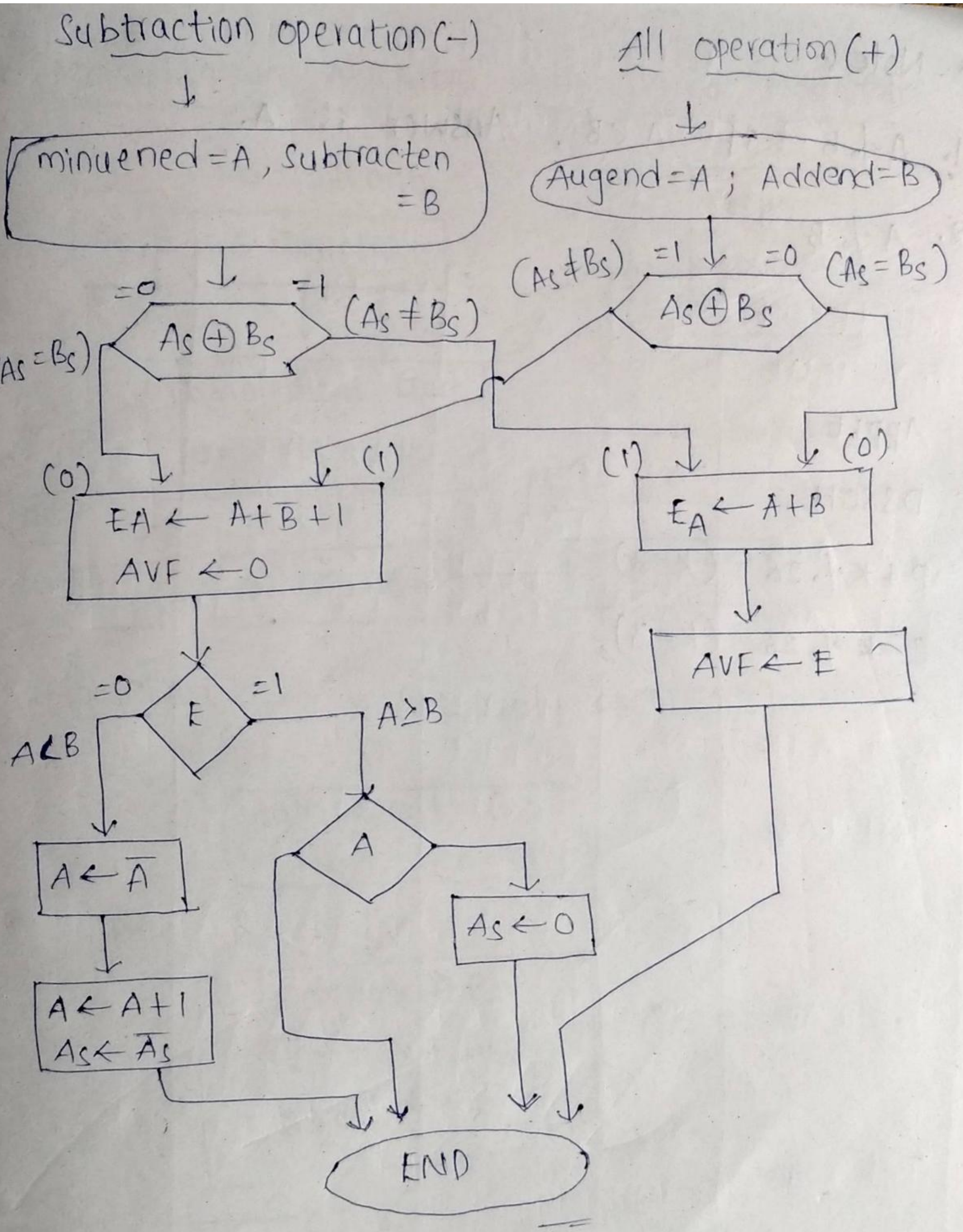


Fig:- Flowchart for add and subtract operations.

* Note :-

1. A %. B if A < B, Answer is A.

2. A %. B

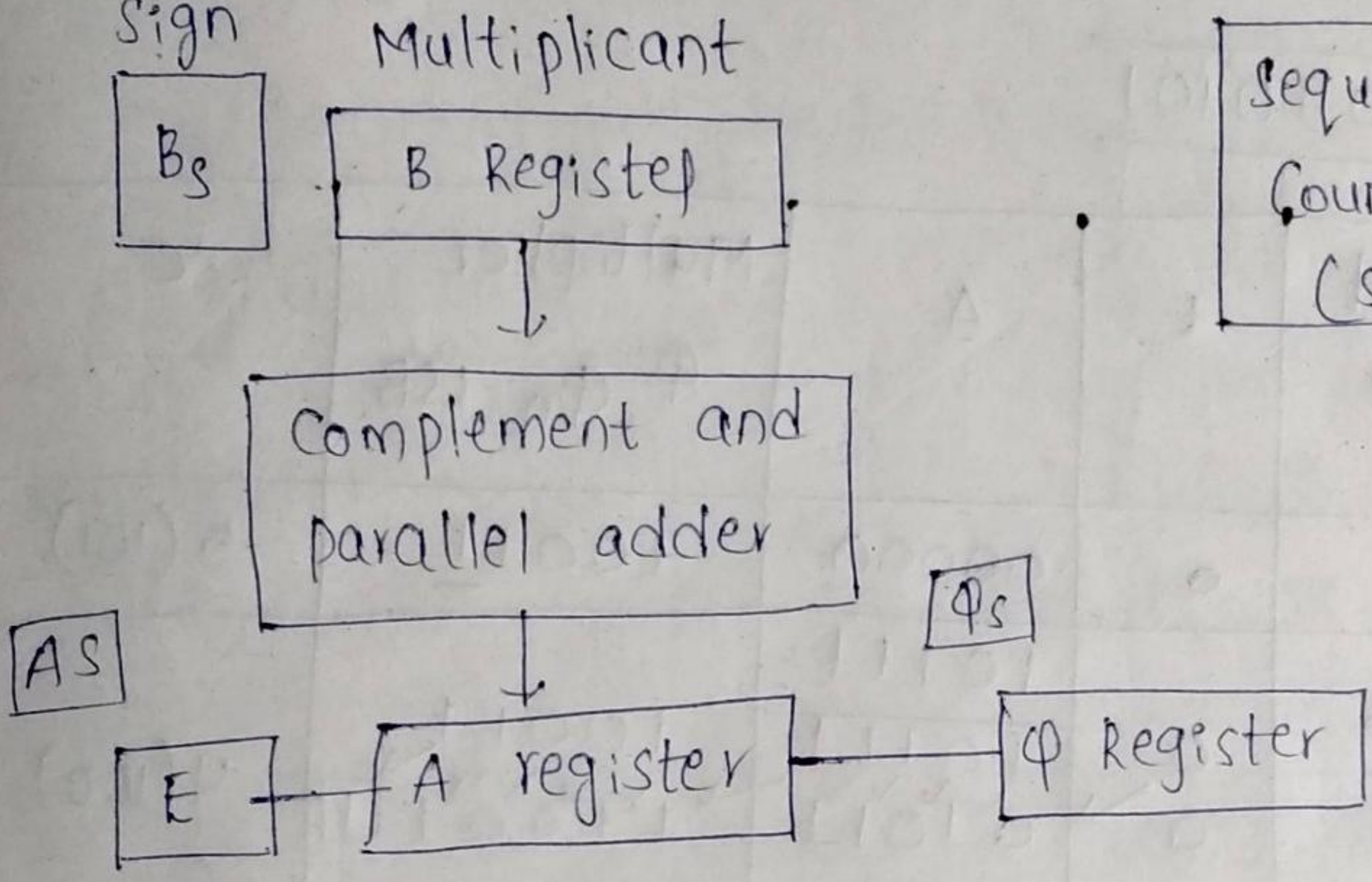
APPLE

DSSOH

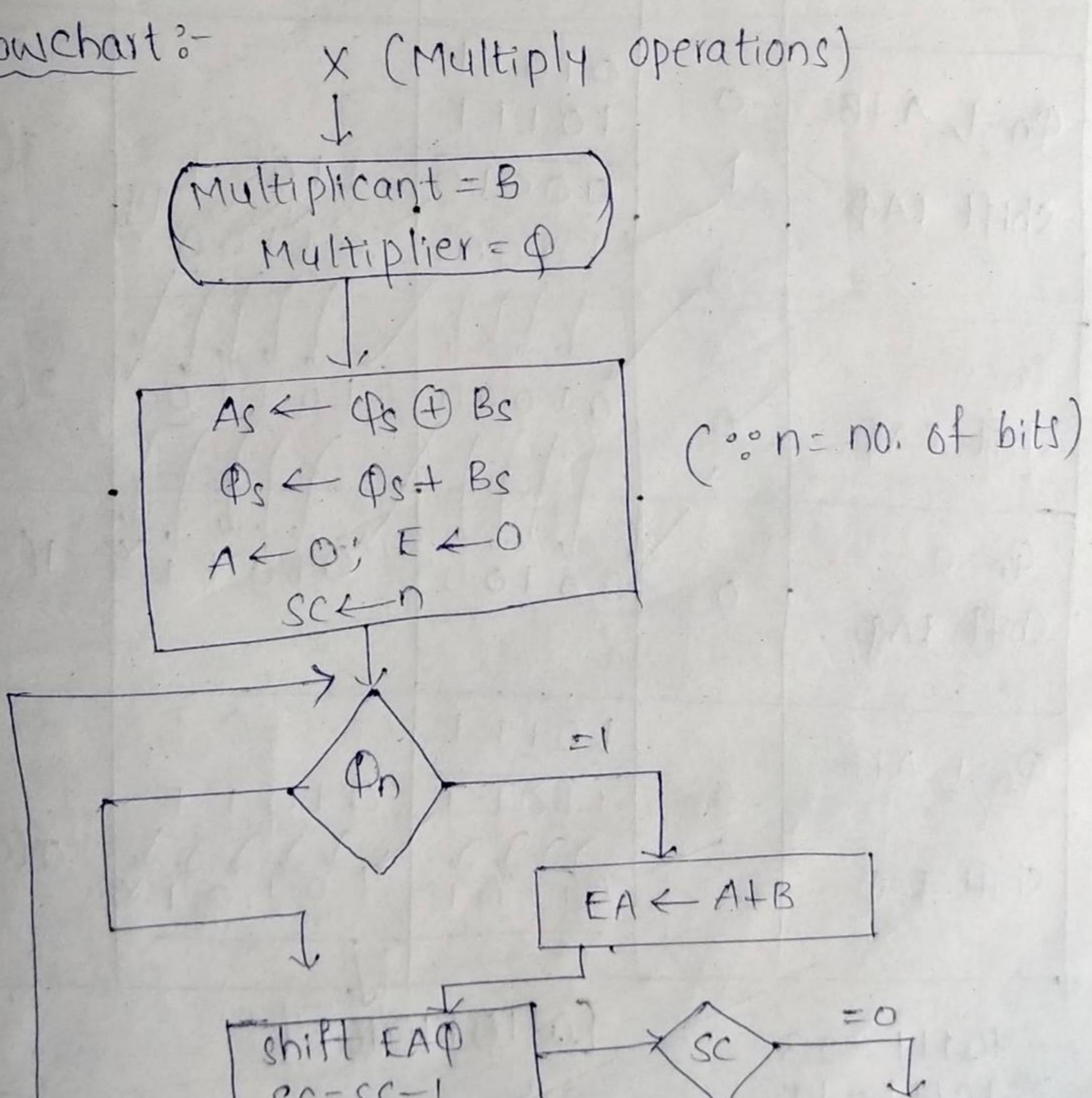
P + K %. 26 (K = 3)

C - K %. 26 (K = 3)

* Multiplication Algorithm with Signed magnitude



* Flowchart :-



* Eg:- 23 = 10111 (multiplicant) = B

19 = 10011 (Multiplier) = Φ

11011 0101

Multiplicant (B) – 10111	E	A	Multiplier Φ $Φ_n$ = LSB	sc
	0	00000	10011	5(101)
$Φ_n$=1, A+B shift EAΦ	0 0	10111 10111 → 01011	10011 → 11001 [1]	4(100)
$Φ_n$=1, A+B shift EAΦ	0 1 0	01011 10111 00010 → 10001 → 01000	11001 → 01100 [1] → 10110 [x]	3(011)
$Φ_n$=0 shift EAΦ	0	01000 → 00100	10110 → 01011 [x]	2(010)
$Φ_n$=0 shift EAΦ	0	00100 → 00010	01011 → 00101 [x]	1(001)
$Φ_n$=1, A+B shift EAΦ	0	10111 11011 → 01101	00101 → 10010 → 01001 [x]	0(000)

10111 = 23

× 10011 = 19

110110101 → Answer

[011011010 1]

* 13 * 17

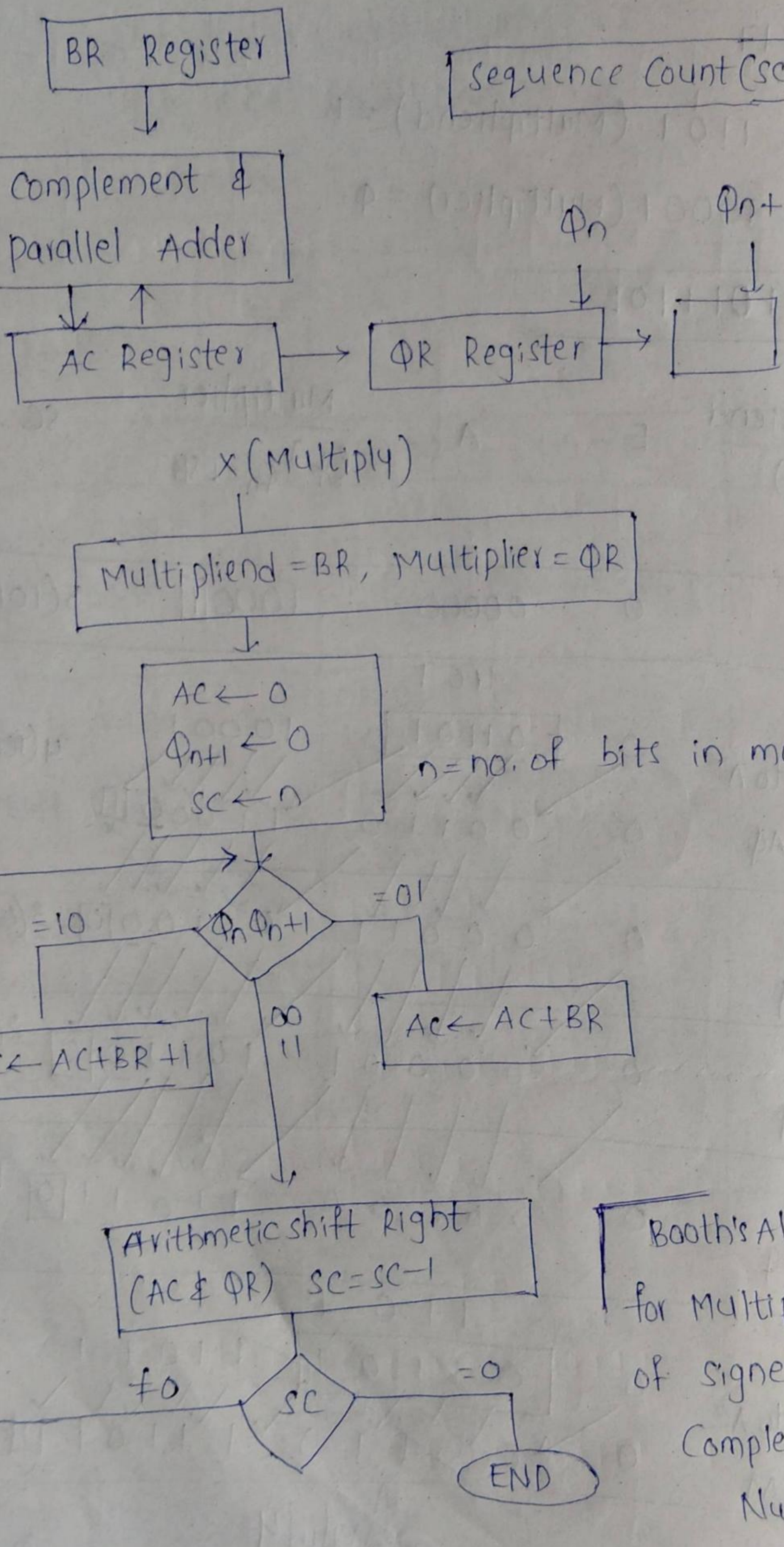
13 = 1101 (Multipliend) = B

17 = 10001 (Multiplier) = Φ

1101 1101

	Multipliend (B)	E	A	Multiplier (Φ) $Φ_n$ = LSB	SC
	1101				
		0	00000	10001	5(101)
			1101		
$Φ_n$=1, add "B to A" shift EAΦ		0	01101	10001	4(100)
		0	00110	11000	
$Φ_n$=0 shift EAΦ		0	00011	01100	3(011)
$Φ_n$=0 shift EAΦ		0	00001	10110	2(010)
$Φ_n$=0 shift EAΦ		0	00000	11011	1(001)
			1101		
$Φ_n$=1 add "B to A" shift EAΦ		0	01101	11011	0 (000)
		0	00110	11011	

Ans(11)4



Booth's Algorithm for Multiplication of Signed 2's Complement Numbers.

* −9 = BR = 10111 (2's signed Complement) — Multiplend

−13 = QR = 10011 (2's signed Complement)

+117 (n=5) — Multiplier.

$Q_n Q_{n+1}$	Multiplend BR = 10111 $\overline{BR}$ = 01000 $\overline{BR}+1$ = 01001	AC	Multiplier QR	Q_{n+1}	SC
		00000	10011	0	101 (5)
10	Subtract BR	01001			
		01001	10011	0	
	ASR (AC & QR)	0 01001	1 1001	1	100 (4)
11	ASR (AC & QR)	0 00100	11100	1	011 (3)
01	Add BR	10111			
		11001			
	ASR (AC & QR)	11100	10111	0	010 (2)
00	ASR (AC & QR)	11110	01011	1	001 (1)
10	Subtract BR	01101			
		00111			
	ASR (AC & QR)	00011	10101	1	000 (0)

* −11 = BR (Multiplicant) = 10101

−13 = QR (Multiplier) = 10011

10001111 (n=5)

Multiplicant

BR = 10101

$\overline{BR}$ = 01010

$\overline{BR}+1$ = 01011

Q_nQ_{n+1}		AC	Multiplier QR	Q_{n+1}	SC
			10011		(n=5)
		00000	10011	0	5(101)
10	Subtract BR	01011			
		01011	10011		
	ASR (AC & QR)	↓ 00101	11001	1	4(100)
11	ASR (AC & QR)	↓ 00010	11100	1	3(010)
01	AC ← AC+BR	10101			
		10111	11100		
	ASR (AC & QR)	↓ 11011	11110	0	2(010)
00	ASR (AC & QR)	↓ 11101	11111	0	1(001)
10	Subtract BR	01011			
	ASR (AC & QR)	01000	11111		
		00100	01111	1	0(000)

* Note :-

1. Multiplication Algorithm works for signed and unsigned numbers.

2. Whele Booth's Algorithm works for 2's complement for signed magnitude numbers.

* Division Algorithm :-

Divisior

$B = (10001)_2$

$B = (17)_{10}$

Dividend

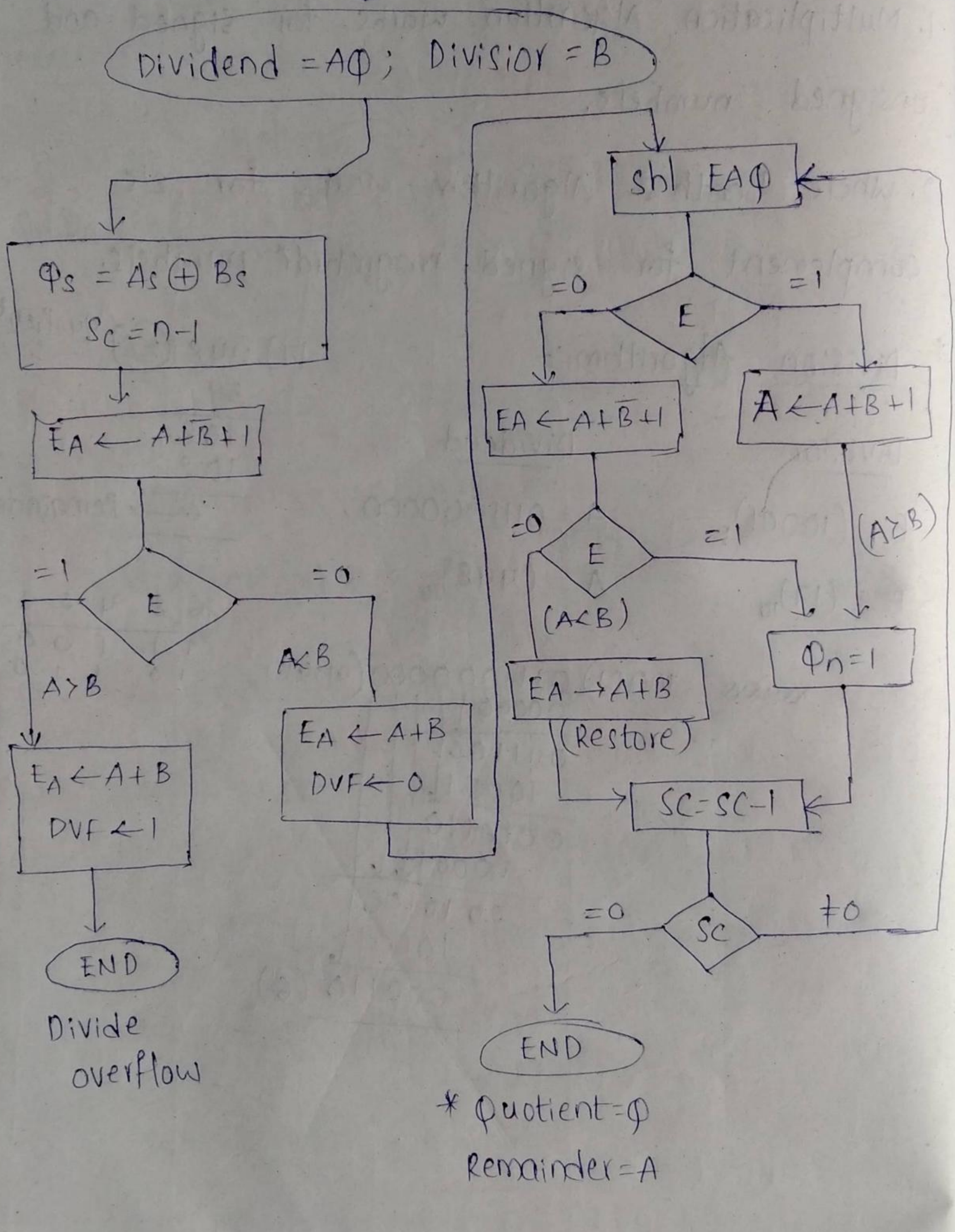
$A = 0111000000$

$A = (448)_{10}$

17) 448 ((26)) quotient
34
108
102
6 → Remainder

16	8	4	2	1
1	1	1	0	0
1	0	1	1	0

```
~~10000~~   10001) 0111000000 (011010
              00000 ↓
              -----
               01110 ↓
               10001 ↓
               -----
               001010 ↓
                10001 ↓
                -----
                000110 (6)
                -----
```



Divisior

B = 10001

$\overline{B}+1$ = 01111

Dividend = AQ

(A = 01110 ; Q = 00000)

SC = 5

(No. of bits in Q)

	E	A	Q	SC
		~~00~~		
		01110	00000	5(101)
Shl EAQ :- →	0	11100	0000 0	
Add A+$\overline{B}$+1	0	01111		
when E=1	1	01011		4(100)
let Q_n=1	1	01011	00001	
Shl EAQ →	0	10110	0001 0	
Add A+$\overline{B}$+1	1	00101		3(011)
when E=1			00011	
let Q_n=1	1			
Shl EAQ →	0	01010	0011 0	
Add A+$\overline{B}$+1 →	0	10001		2(010)
when E=0,				
here Q_n=0		01010	00110	
Add A+B (Restore)	1			
Shift l EAQ →	0	10100	0110 0	
Add A+$\overline{B}$+1 →	1	00111		1(001)
when E=1, Q_n=1			0110 1	
Shl EAQ →	0	00110	110 0	

Shl EAQ → 0 00110 11010 [0] 0(000)

Add A+B̄+1 → 01111

0 10001

when E=0, 10001

here Q_n=0 00110 11010

Add A+B(Restore) [1] 00110 11010

[00110] [11010]

---

* 444 ÷ 17

17) 444(26
34
104
102
2

Divisor Dividend

B = $(17)_{10}$ A = 0110111100

B = $(10001)_2$ A = $(444)_{10}$

10001) 0110111100(011001
00000
01101
10001
01010
10001
00011100
10001
01011

16	8	4	2	1
0	1	1	0	1
1	1	0	0	1
			1	
1	0	1	0	0
		1	1	1
	1	1	1	0
1	1	1	0	0

$B = 10001$

$\bar{B} = 01110$

$\bar{B}+1 = 01111$

Divided = $A\phi$

$A = 01101$

$\phi = 11100$

```
                    01101
10001 ) 0110111100 ( 011010
          00000
          -----
           11011
           10001
           -----
           01010
            10001
            -----
            00100 10
             10001
             -----
             00010 0
              00000
              -----
               10
```

	E	A	ϕ	SC
		01101	11100	5(101)
Shl EAϕ	0	11011	1100 [0]	
Add $A+\bar{B}+1$	[1]	01111		
		01010		
When E=1				4(100)
Let $\phi_n=1$		01010	11001	
Shl EAϕ	0	10101	1001 [0]	
Add $A+\bar{B}+1$	[1]	01111		
		00100		
when E=1				3(011)
Let $\phi_n=1$		00100	10011	
Shl EAϕ	0	01001	0011 [0]	
Add $\bar{A}+\bar{B}+1$	[0]	11000	00110	
when E=0		10001		
here $\phi_n=0$		01001	00110	2(010)
Add $A+B$ (Restore)	[1]			

1 01001 00110 2(010)

Shl EAQ 0 10010 0110 0

Add A+B̄+1 1 01111 / 00001

when E=1,

Let Qn=1 00001 01101 1(001)

Shl EAQ 0 00010 1101 0

Add A+B̄+1 0 01111 / 10001

when E=0, 10001

Qn=0 1 00010 11010 0(000).

Restore A+B 00010 11010

(Restore)

---

* Floating point Representation:-

(1) $(1259.125)_{10}$ → Single precision

(2) $(263.3)_{10}$ → Double precision.

Ans).

(1) $(1259.125)_{10} = (10011101011.001)_2$

$= (1.0011101011001) \times 2^{10}$

$\Rightarrow m \times r^e$

$0.125 \times 2 = 0.250 = 0.$

$0.25 \times 2 = *.0.5 = 0$

$0.5 \times 2 = 1 = 1$

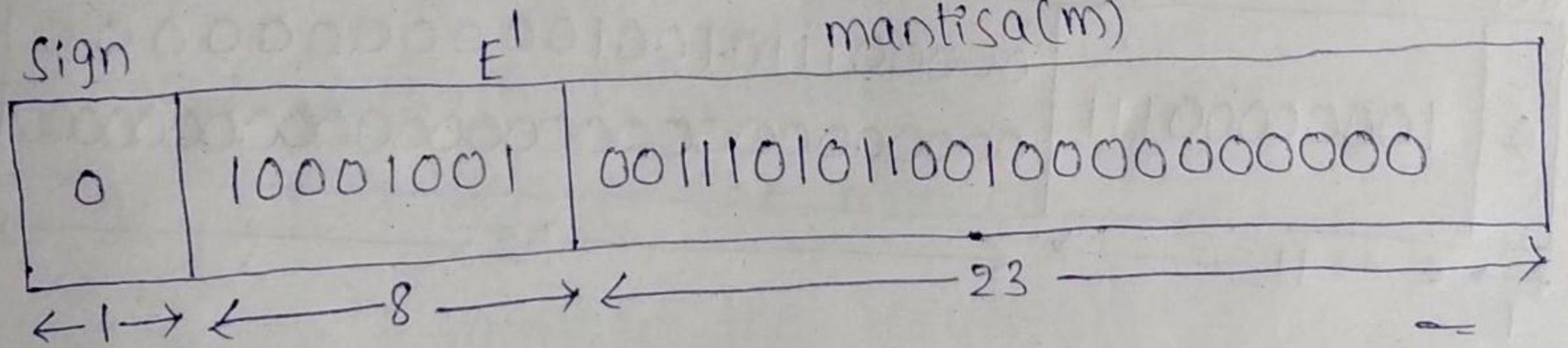
2 | 1259
2 | 629 - 1
2 | 314 - 1
2 | 157 - 0
2 | 78 - 1
2 | 39 - 0
2 | 19 - 1
2 | 9 - 1
2 | 4 - 1
2 | 2 - 0
1 - 0

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

$E^1 = E + BIOS$

$= 10 + 127$

$= 137$



(2) $(263.3)_{10} = (100000111.01001)_2$

2 | 263
2 | 131 - 1
2 | 65 - 1
2 | 32 - 1
2 | 16 - 0
2 | 8 - 0
2 | 4 - 0
2 | 2 - 0
1 - 0

$0.3 \times 2 = 0.6 \rightarrow 0$

$0.6 \times 2 = 1.2 \rightarrow 1$

$0.2 \times 2 = 0.4 \rightarrow 0$

$0.4 \times 2 = 0.8 \rightarrow 0$

$0.8 \times 2 = 1.6 \rightarrow 1$ ✓

$0.6 \times 2 = 1.2 \rightarrow 1$

$\Rightarrow (1.00000111101001) \times 2^8$

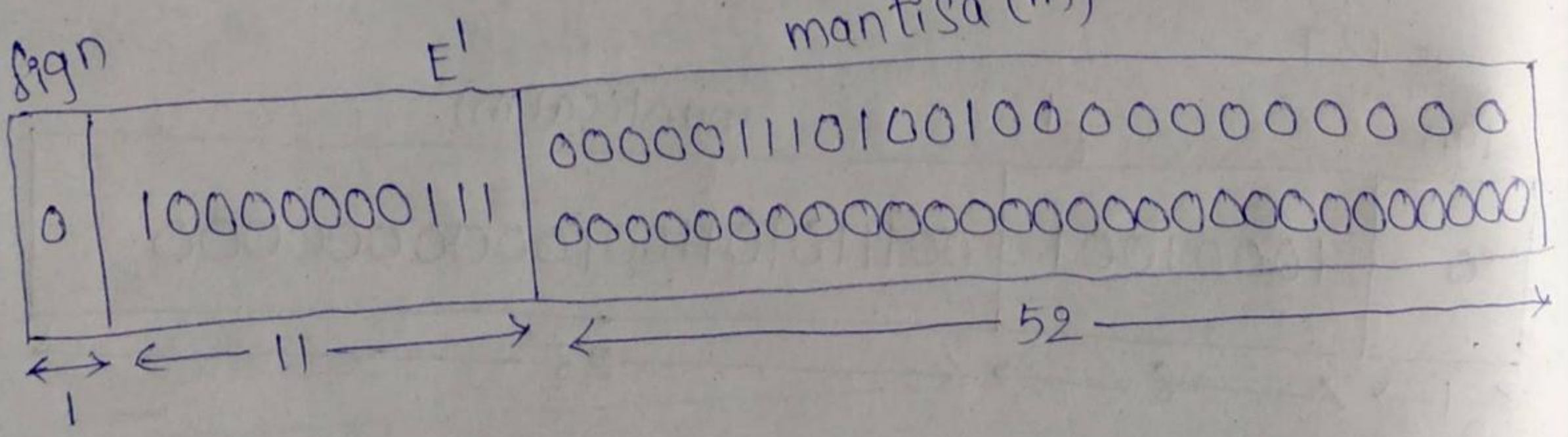
$\Rightarrow m \times r^e$

$E^1 = E + 1023$

$= 8 + 1023$

$= 1031$

2	1031
2	515 - 1
2	257 - 1
2	128 - 1
2	64 - 0
2	32 - 0
2	16 - 0
2	8 - 0
2	4 - 0
2	2 - 0
	1 - 0 ↑



* Floating Point Arithmetics :-

(1) Addition :-

Steps :-

Step 1 :- check for zeroes

Step 2 :- Align the mantisa.

Step 3 :- Add the mantisa.

Step 4 :- Normalize the result

(2) Subtraction :-

Step 1 :- check for zeroes.

Step 2 :- Align the mantisa.

Step 3 :- Subtract the mantisa.

Step 4 :- Normalize the result

(3) Multiplication :-

Step 1 :- check for zeroes

Step 2 :- Add the exponents.

Step 3 :- Multiply the mantisa.

Step 4 :- Normalize the result.

(4) Division :-

Step 1 :- check for zeroes.

Step 2 :- Initialize the registers & evaluate

the sign.

step 3 :- Align the dividend.

step 4 :- Subtract the exponents.

step 5 :- Divide the mantisas.

step 6 :- Normalize the result.

* $A = (0.5372400 \times 10^{2})$

$B = (0.1580000 \times 10^{-1})$