Module : 01

PAGE No.

DATE / /

COMPUTER FUNDAMENTALS

Number Systems :

- Decimal number → base :- 10 → (0-9) digits
- Binary num system → base : 2 → bits : 0 and 1
- Octal number system → base : 8 → digits : 0 to 7
  group of 3 binary bits make 1 octal digit.
- Hexadecimal num. system → base : 16 → digits : 0-9 & A-F
  where A = 10, B = 11, C = 12, ... F = 15
  4 binary bits = 1 Hex. digit

> Conversions :-

- any base to decimal
  multiply each digit by its corresponding power of base and summing the result. (start from LSB (right side))

- decimal to any base
  Divide the decimal number repeatedly by the target base and note the remainders. Remainders in reverse order will be the required conversion.

PAGE No.
DATE / /

- ANY BASE to DECIMAL

1) Binary to decimal

multiply each bit by 2^n, starting from the right.

i) $(1101)_2 = (\quad)_{10}$

→ Binary : $\underset{2^3}{1}\ \underset{2^2}{1}\ \underset{2^1}{0}\ \underset{2^0}{1}$

multiply each binary digit by 2 raised to its position and add them

$= 2^0 \times 1 + 2^1 \times 0 + 2^2 \times 1 + 2^3 \times 1$

$= 1 + 0 + 4 + 8$

$= 13$

$\therefore (1101)_2 = (13)_{10}$

2) $(101011)_2 = (\quad)_{10}$

→ Binary : $\underset{2^5}{1}\ \underset{2^4}{0}\ \underset{2^3}{1}\ \underset{2^2}{0}\ \underset{2^1}{1}\ \underset{2^0}{1}$

$= 2^0 \times 1 + 2^1 \times 1 + 2^3 \times 1 + 2^5 \times 1$

$= 1 + 2 + 8 + 32$

$= 43$

$\therefore (101011)_2 = (43)_{10}$

PAGE No.
DATE / /

3) $(11000111)_2 = (\quad)_{10}$

→ Binary : $\underset{2^7}{1}\ \underset{2^6}{1}\ \underset{2^5}{0}\ \underset{2^4}{0}\ \underset{2^3}{0}\ \underset{2^2}{1}\ \underset{2^1}{1}\ \underset{2^0}{1}$

$= 2^0 \times 1 + 2 \times 1 + 2^2 \times 1 + 2^6 \times 1 + 2^7 \times 1$

$= 1 + 2 + 4 + 64 + 128$

$= 199$

$\therefore (11000111)_2 = (199)_{10}$

4)

2) Octal to decimal

multiply each digit by 8^n, [right → left]

1) $(357)_8 = (\quad)_{10}$

→ Octal : $\underset{8^2}{3}\ \underset{8^1}{5}\ \underset{8^0}{7}$

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

$= 239$

$\therefore (357)_8 = (239)_{10}$

2) $(4725)_8 = (\quad)_{10}$

→ Octal : $\underset{8^3}{4}\ \underset{8^2}{7}\ \underset{8^1}{2}\ \underset{8^0}{5}$

$= 5 + 2\times8 + 7\times64 + 4\times512$

$= 5 + 16 + 448 + 2048$

$= 2517$

3) Hexadecimal to decimal

multiply each digit by 16^n

replace (A to F) with its decimal equivalent

1) $(ABC)_{16} = (\quad)_{10}$

→ Hexadecimal : A B C

16^2 16^1 16^0

where A = 10, B = 11, C = 12

$= 16^0\times12 + 16\times11 + 16^2\times10$

$= 12 + 176 + 2560$

$= 2748$

$\therefore (ABC)_{16} = (2748)_{10}$

2) $(4C)_{16} = (\quad)_{10}$

→ Hex : 4 C

16^1 16^0

where C = 12

$= 1\times12 + 16\times4$

$= 76$

$\therefore (4C)_{16} = (76)_{10}$

PAGE No.
DATE / /

3) $(4C8)_{16} = (\quad)_{10}$

→ Hex: 4 C 8

16^2 16^1 16^0

$= 8 + 16 \times 12 + 256 \times 4$

$= 1224$

$\therefore (4C8)_{16} = (1224)_{10}$

- DECMAL to ANY BASE

1) Decimal to binary

divide by 2 and write down remainders. (↑)

1) $(125)_{10} = (\quad)_2$

→

	Q	R
$125 \div 2$	$= 62$	1
$62 \div 2$	$= 31$	0
$31 \div 2$	$= 15$	1
$15 \div 2$	$= 7$	1
$7 \div 2$	$= 3$	1
$3 \div 2$	$= 1$	1
$1 \div 2$	$= 0$	1

↑

$\therefore (125)_{10} = (1111101)_2$

2) $(1473)_{10} = (\quad)_2$

→

	Q	R
$1473 \div 2$	$= 736$	1
$736 \div 2$	$= 368$	0
$368 \div 2$	$= 184$	0
$184 \div 2$	$= 92$	0
$92 \div 2$	$= 46$	0
$46 \div 2$	$= 23$	0
$23 \div 2$	$= 11$	1
$11 \div 2$	$= 5$	1
$5 \div 2$	$= 2$	1
$2 \div 2$	$= 1$	0
$1 \div 2$	$= 0$	1

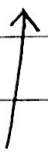
$\therefore (1473)_{10} = (10111000001)_2$

2) Decimal to octal

1) $(247)_{10} = (\quad)_8$

→

	Q	R
$247 \div 8$	$= 30$	7
$30 \div 8$	$= 3$	6
$3 \div 8$	$= 0$	3



$\therefore (247)_{10} = (367)_8$

PAGE No.
DATE / /

2) $(1473)_{10} = (\quad)_8$

→

	Q	R
$1473 \div 8 =$	184	1
$184 \div 8 =$	23	0
$23 \div 8 =$	2	7
$2 \div 8 =$	0	2

↑

$\therefore (1473)_{10} = (2701)_8$

3) **Decimal to hexadecimal**

divide the decimal num. by 16 & write down the remainder. if the remainder is 10 to 15, convert it to A to F.

1) $(199)_{10} = (\quad)_{16}$

→

	Q	R	Hex
$199 \div 16 =$	12	7	7
$12 \div 16 =$	0	12	C

↑

$\therefore (199)_{10} = (C7)_{16}$

2) $(2156)_{10} = (\quad)_{16}$

→

	Q	R	Hex
$2156 \div 16 =$	134	12	C
$134 \div 16 =$	8	6	6
$8 \div 16 =$	0	8	8

↑

$\therefore (2156)_{10} = (86C)_{16}$

PAGE No.
DATE / /

Conversions

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18
25	11001	31	19
26	11010	32	1A

DATE / /

> Binary to octal

Group 3 binary bits from right to left
If needed, add leading zeroes to make complete groups of 3
convert each 3-bit group to its octal equivalent

1) $(101011)_2 = (\ )_8$

→ Binary : 1 0 1 0 1 1
↓ ↓
Octal : 5 3

∴ $(101011)_2 = (53)_8$

2) $(1101101)_2 = (\ )_8$

→ (add leading zeros)

∴ Binary : 001 101 101
↓ ↓ ↓
Octal 1 5 5

∴ $(1101101)_2 = (155)_8$

3) $(110111000)_2 = (\ )_8$

→ Binary : 110 111 000
↓ ↓ ↓
6 7 0

∴ $(110111000)_2 = (670)_8$

> Octal to Binary

Convert each octal digit into its 3-bit binary number.
Combine all binary groups to get the final binary num.

1) $(125)_8 = (\quad)_2$

→ Octal : 1 2 5
↓ ↓ ↓
binary: 001 010 101

$\therefore (125)_8 = (001010101)_2$ or $(1010101)_2$

2) $(47)_8 = (\quad)_2$

→ Octal : 4 7
↓ ↓
binary : 100 111

$\therefore (47)_8 = (100111)_2$

3) $(155)_8 = (\quad)_2$

→ Octal : 1 5 5
↓ ↓ ↓
Binary : 001 101 101

$\therefore (155)_8 = (001101101)_2$ or $(1101101)_2$

PAGE No.
DATE / /

> Binary to Hexadecimal

Group 4 binary bits, starting from right to left
Add leading zeros if needed to complete a 4-bit group.
convert each group into its hex. equivalent.

1) $(11110101)_2 = (\quad)_{16}$

→ Binary: 1111 0101
↓ ↓
Hex: F 5

∴ $(11110101)_2 = (F5)_{16}$

2) $(1001101111)_2 = (\quad)_{16}$

→ Binary: 0010 0110 1111 (add leading zeros)
↓ ↓ ↓
Hex: 2 6 F

∴ $(1001101111)_2 = (26F)_{16}$

PAGE No.
DATE / /

> Hexadecimal to binary

Write each hex digit as a 4-bit binary num. combine all binary groups.

1) $(3A7)_{16} = (\quad)_2$

→ Hex : 3 A 7
binary : 0011 1010 0111

$\therefore (3A7)_{16} = (001110100111)_2$

2) $(FAE1)_{16} = (\quad)_2$

→ Hex : F A E 1
binary : 1111 1010 1110 0001

$\therefore (FAE1)_{16} = (1111101011100001)_2$

3) $(7B6C)_{16} = (\quad)_2$

→ Hex : 7 B 6 C
Binary : 0111 1011 0110 1100

$\therefore (7B6C)_{16} = (0111101101101100)_2$

any base → binary → anybase

PAGE No.
DATE / /

> Octal to Hexadecimal

1) Convert octal to binary ⇒ each octal digit = 3 bit binary
2) Group the binary digits in 4s from right to left (add adding zeros if needed)
3) convert each 4-bit group to Hex.

1) $(157)_8 = (\quad)_{16}$

→ Octal : 1 5 7
Binary 001 101 111

$\therefore (157)_8 = (001101111)_2$

Now, Binary : 0000 0110 1111
Hex : 0 6 F

$\therefore (157)_8 = (6F)_{16}$

2) $(726)_8 = (\quad)_{16}$

→ Octal : 7 2 6
Binary : 111 010 110

$(726)_8 = (111010110)_2$

Binary : 0001 1101 0110
Hex : 1 D 6

$\therefore (726)_8 = (1D6)_{16}$

Octal → Hex ⇒ Oct → Binary → Hex

Hex → Octal ⇒ Hex → bin → oct

PAGE No.
DATE / /

> Hexadecimal to octal

Convert hex to binary
each hex digit = 4-bit binary

Group the binary digits in 3s from right to left.
convert each 3-bit to octal.

1) $(2F3)_{16} = (\ )_8$

→ Hex : 2 F 3
Binary: 0010 1111 0011

∴ $(2F3)_{16} = (0010\ 1111\ 0011)_2$

Binary : 001 011 110 011
Octal : 1 3 6 3

∴ $(2F3)_{16} = (1363)_8$

2) $(A9B)_{16} = (\ )_8$

→ Hex : A 9 B
Binary: 1010 1001 1011

∴ $(A9B)_{16} = (101010011011)_2$

Binary : 101 010 011 011
Octal : 5 2 3 3

∴ $(A9B)_{16} = (5233)_8$

PAGE No.
DATE / /

Binary addition :-

A	B	A + B	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1
1	1 +1	1	1

Ex: 1) 10101 + 11001 = 101110

→

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

(2)

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

⇒ 11011

Binary subtraction :-

A	B	A - B	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

1)

```
  0  10
  1 + 0  0
- 1  0  1  0
-----------
  0  0  1  0
```

2)

```
  1  10 10
  1  0 + 0
-   0  0  1  1
-------------
    0  1  1  1
```

(3)

```
      0  10
  1  1 + 0
-   1  1  0  1
-------------
    0  0  0  1
```

Binary num. Representation

1) Unsigned numbers
- Represents only +ve integers
- All bits are used to represent the value (magnitude)
- No sign bit is used.

Range : 0 to $(2^n - 1)$ For n bits [0 to 15] For 4-bit

Ex : 1111 = 15 , 0001 = 1

2) Signed - Binary Repres. (Sign-magnitude)
- used to represent +ve & -ve numbers.
- The MSB (Most Significant Bit) is the sign bit:
  - 0 → +ve
  - 1 → -ve
- The remaining bits represent the magnitude (value).

Ex :

Binary	Sign	Magnitude	decimal
0101	0	101	+5
1101	1	101	-5

Range : $-(2^{n-1} - 1)$ to $(2^{n-1} - 1)$ [-7 to +7] For 4 bit

a) 1's complement
b) 2's complement

PAGE No.
DATE / /

a) 1's Complement

- Negative num. are represented by inverting all bits of the +ve binary num.
- MSB acts as sign bit.
- Has two zeros : +0 & -0

Flip all bits (0 → 1, 1 → 0)

Range : $-(2^{n-1}-1)$ to $(2^{n-1}-1)$ .. For n-bits

-7 to +7 for 4-bits

Ex : +5 = 0101

-5 = 1010 (in 1's comp.)

+0 = 0000

-0 = 1111 → (1's comp. of 0000)

Hence, 1's comp. has two representations for zero.

b) 2's complement

- Negative num. are represented by :
  - Taking the 1's complement, and
  - Add 1 to the result.

Range : $-(2^{n-1})$ to $(2^{n-1}-1)$ for n-bits

For 4-bits -8 to +7

Ex : Represent -7 in 2's comp. (4-bit)

→ +7 = 0111

1's complement → 1000

Add 1 → 1000 + 1 = 1001

∴ −7 = 1001 (in 2's comp.)

2) Represent −5 in 2's complement (4-bit)

→ +5 = 0101

1's comp → 1010

add 1 → 1010
+ 1
1011

∴ −5 = 1011 (in 2's comp.)

- Single zero representation in 2's complement

+0 = 0000

1's comp → 1111

add 1 → 1111
+ 1
10000 (5-bits)

discard the leftmost carry in 4-bit representation.

∴ −0 = 0000, same as +0

* Shortcut for 2's comp.

1) Start from LSB (right)

2) Copy all bits as they are untill u copy the 1st 1.

3) After that invert all remaining bits

+5 = 1 0 1 → copy this as it is after that invert all

−5 = 0 1 1

10100 → 01100

↑ after this invert

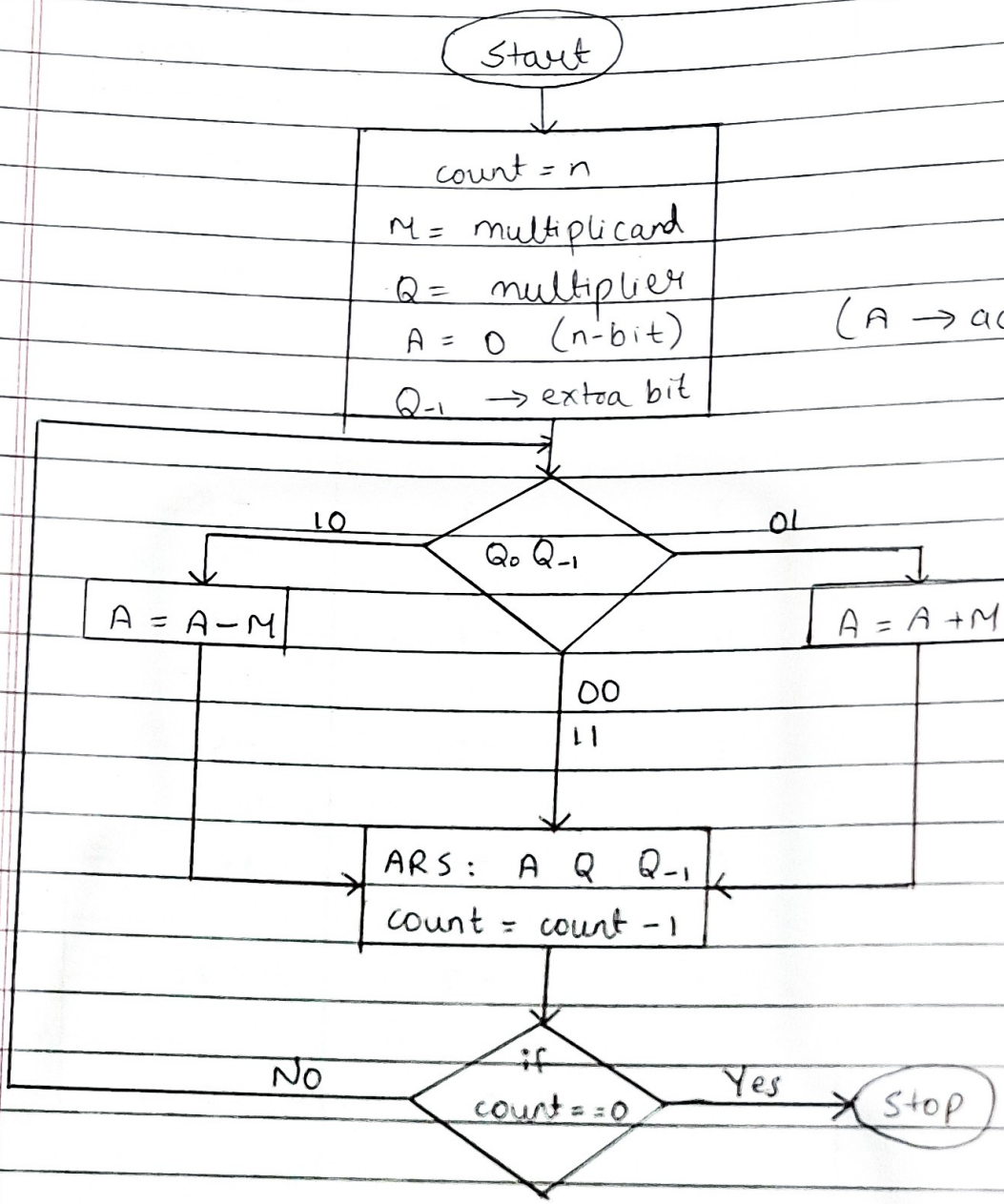
PAGE No.
DATE / /

4-bit binary representations :-

Binary	Unsigned	Signed	1's comp.	2's comp.
0000	0	+0	+0	+0
0001	1	+1	+1	+1
0010	2	+2	+2	+2
0011	3	+3	+3	+3
0100	4	+4	+4	+4
0101	5	+5	+5	+5
0110	6	+6	+6	+6
0111	7	+7	+7	+7
1000	8	-0	-0	-8
1001	9	-1	-6	-7
1010	10	-2	-5	-6
1011	11	-3	-4	-5
1100	12	-4	-3	-4
1101	13	-5	-2	-3
1110	14	-6	-1	-2
1111	15	-7	-0	-1

ARS → arithmetic right shift

Booth's Algorithm (multiplication algo)



1) Solve : 5 × 3 by booth's multiplication

→ M = 5 → 0101 } 4-bit
Q = 3 → 0011 }
A = 0000
Q_{-1} = 0 (extra bit)
count = 4 (bcoz we are using 4-bit nums)
-M = -5 → 1011

count	A	Q	Q-1	Operation
4	0000	0011	0	A = A - M
				0000 + 1011
	1011	0011	0	
	1101	1001	1	ARS
3	1110	1100	1	~~A = A - M~~ ARS
2	0011	1100	1	A = A + M = 1110 +
	0001	1110	0	ARS
1	0000	1111	0	ARS

$AQ = (00001111)_2 = (15)_{10}$

2) Solve : $5 \times (-3)$ using booth's algo.

→ $M \rightarrow 5 \equiv 0101$, $-M = -5 \equiv 1011$

~~$Q \rightarrow -3 \equiv 0011$, $-Q = 1101$~~

$Q \rightarrow -3 = 1101$, as [$+3 = 0011$, $-3 = 1101$]

count	~~Q~~ A	Q	Q-1	Operation
4	0000	1101	0	
	1011	1101	0	A = A - M
	1101	1110	1	ARS

3	0010	1110	1	A = A + M
	0001	0111	0	ARS
2	1100	0111	0	A = A - M
	1110	0011	1	ARS
1	1111	0001	1	ARS

$AQ = (11110001)_2 = (-15)$

3) Solve : $(-5) \times 3$

→ $M \rightarrow -5 \equiv 1011$, $-M \rightarrow 5 = 0101$

$Q \rightarrow 3 = 0011$

count	A	Q	Q_{-1}	Operation
4	0000	0011	0	
	0101	0011	0	A = A - M
	0010	1001	1	ARS
3	0001	0100	1	ARS
2	1100	0100	1	A = A + M
	1110	0010	0	ARS
1	1111	0001	0	ARS

$AQ = (11110001)_2 = (-15)_{10}$

PAGE No.
DATE / /

4) $(-5) \times (-3)$

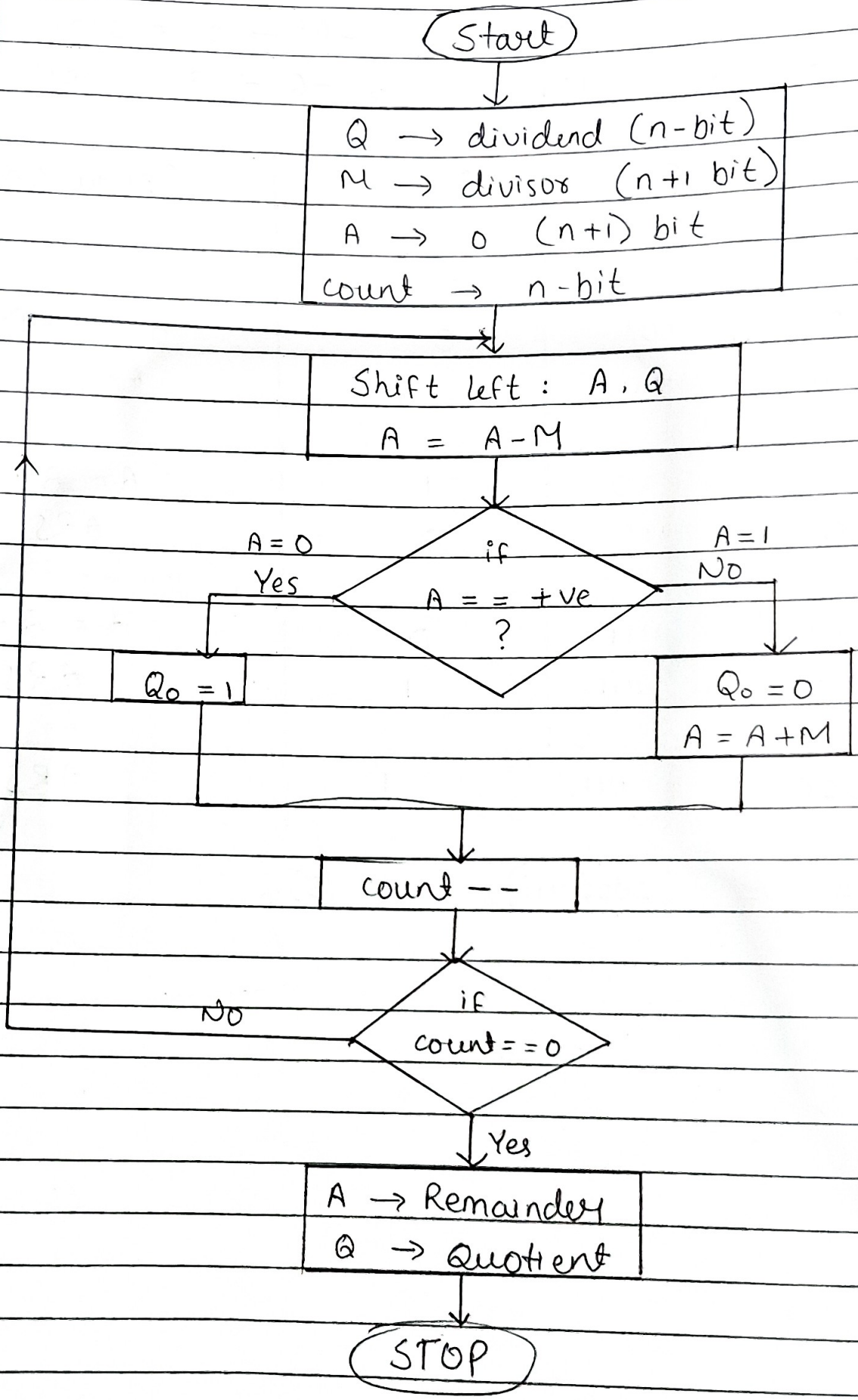
→ $M \rightarrow -5 = 1011$, $-M \rightarrow 5 \equiv 0101$

$Q \rightarrow -3 = 1101$, $-Q \rightarrow 3 = 0011$

Count	A	Q	Q-1	Operation
4	0000	1101	0	
	0101	1101	0	A = A - M
	0010	1110	1	ARS
3	1101	1110	1	A = A + M
	1110	1111	0	ARS
2	0011	1111	0	A = A - M
	0001	1111	1	ARS
1	0000	1111	1	ARS

$\therefore AQ = (00001111)_2 = (15)_{10}$

Restoring division Algorithm



PAGE No.
DATE / /

1) Solve : $6 \div 4$

→ $Q \rightarrow 6 = 0110$ (n-bit)

$M \rightarrow 4 = 00100$, $-M \rightarrow -4 = 11100$ [5-bit]

$A = 00000$ [(n+1) bit]

count = n = 4

count	A	Q	Operation
4	00000	0110	
	00000	110□	Shift left
	11100	1100	A = A - M
	00000	1100	A = A + M
3	00001	100□	Shift left
	11101	1000	A = A - M
	00001	1000	A = A + M
2	00011	000□	Shift left
	11111	0000	A = A - M
	00011	0000	A = A + M
1	00110	000□	Shift left
	00010	0001	A = A - M
	↓ remainder	↓ quotient	

∴ Quotient $Q = 0001 = (1)_{10}$

Rem. $A = 00010 = (2)_{10}$

2) $7 \div 2$

→ $Q \rightarrow 7 = 0111$ (4 bit), $A = 00000$ (5-bit)
$M \rightarrow 2 = 00010$ (5 bit), $-M = 11110$

count	A	Q	operation
4	00000	0111	
	00000	111 □	Shift left
	11110	111 0	A = A - M
	00000	1110	A = A + M
3	00001	110 □	Shift left
	11111	111 0	A = A - M
	00001	1110	A = A + M
2	00011	110 □	Shift left
	00001	100 1	A = A - M
1	00011	001 □	Shift left
	00001	0011	A = A - M
	↓	↓	
	rem.	Q	

∴ remainder = $(00001)_2 = (1)_{10}$
quotient = $(0011)_2 = (3)_{10}$

PAGE No.
DATE / /

3) $11 \div 5$

→ $Q = 11 \rightarrow 1011$

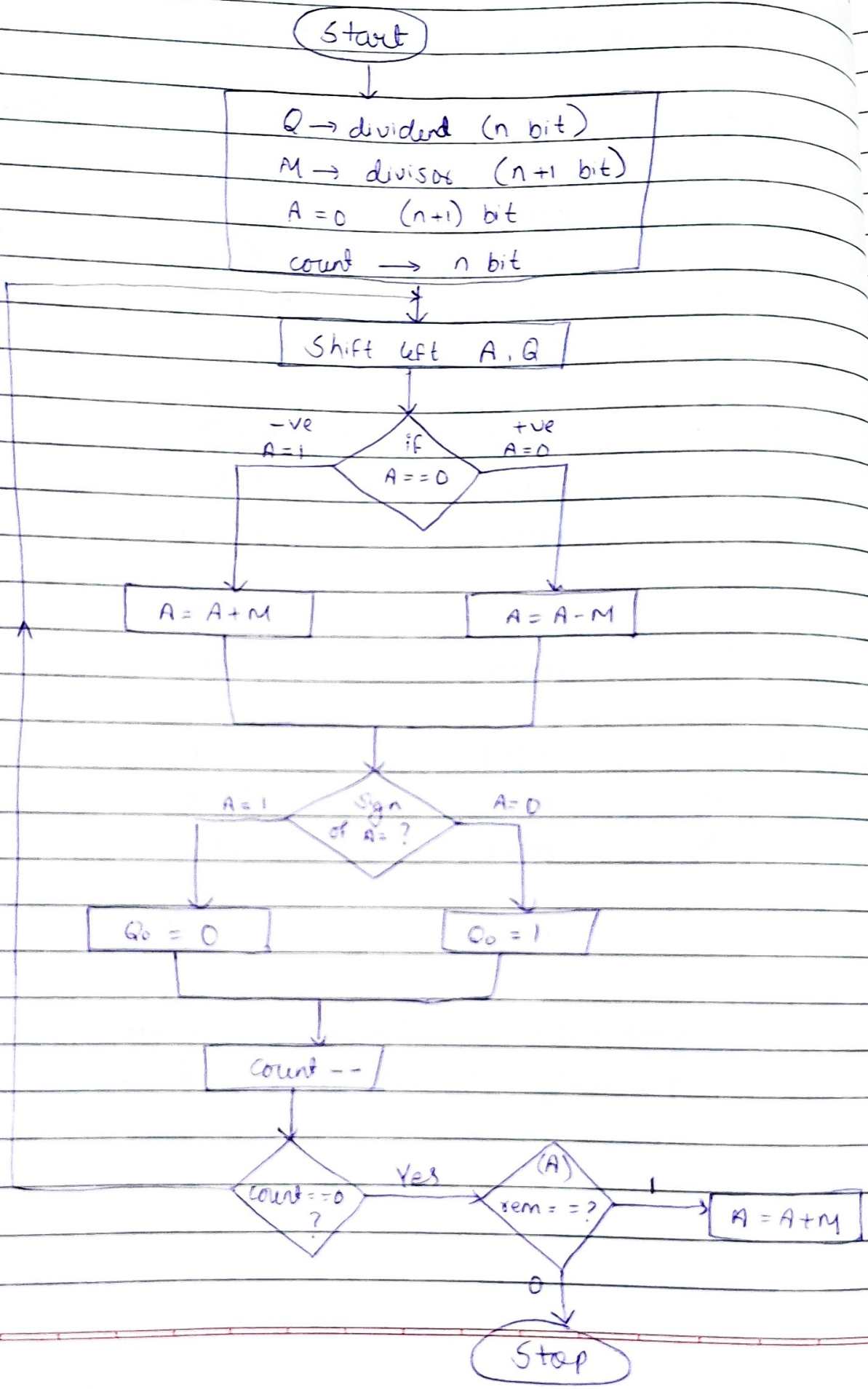
$M = 5 \rightarrow 00101 \Rightarrow -M = 11011$

count	A	Q	Operation
4	00000	1011	
	00001	011 □	SL
	11100	0110	A = A - M
	00001	0110	A + = M
3	00010	110 □	SL
	11101	1100	A = A - M
	00010	1100	A = A + M
2	00101	100 □	SL
	00000	1001	A = A - M
1	00001	001 □	SL
	11100	0010	A = A - M
	00001	0010	A = A + M
	↓	↓	
	rem.	quot.	

∴ remainder = $(00001)_2 = (1)_{10}$

quotient = $(0010)_2 = (2)_{10}$

> Non- restoring



PAGE No.
DATE / /

1) 8 ÷ 3 by non-restoring - - -

→ Q → 8 = 1000

M → 3 = 00011 , −M → 11101

A = 00000

count	A	Q	Operation
4	00000	1000	
	[0]0001	000[ ]	SL
	[1]1110	000[0]	A = A−M
3	11100	000[ ]	SL
	[1]1111	0000	A = A+M
2	11110	000[ ]	SL
	00001	000[1]	A = A+M
1	00010	001[ ]	SL
	[1]1111	0010	A = A−M
0	00010	0010	A = A+M
	↓	↓	
	rem	Quot.	

remainder = $(00010)_2 = (2)_{10}$

Quotient = $(0010)_2 = (2)_{10}$ //

PAGE No.
DATE / /

2) 9 ÷ 5 , by non-restoring ---

→ Q → 9 → 1001

M = 5 → 00101 , −M = 11011

count	A	Q	Operation
4	00000	1001	
	00001	001□	SL
	11100	0010	A = A − M
3	11000	010□	SL
	11101	010[0]	A = A + M
2	11010	100□	SL
	11111	1000	A = A + M
1	11111	000□	SL
	00100	0001	A = A + M

remainder = $(00100)_2 = (4)_{10}$

quotient = $(0001)_2 = (1)_{10}$

3) −(8 ÷ 3)

→ Q → 8 → 1000

M → 3 → 00011 , −M = 11101

A = 00000

PAGE No.
DATE / /

count	A	Q	operation
4	00000	1000	
	00001	000□	SL
	11110	000[0]	A = A - M
3	11100	000□	SL
	11101	0000	A = A + M.
2	11110	000□	SL
	00001	0001	A = A + M
1	00010	001□	SL
	11111	0010	A = A - M
0	00010	0010	A = A + M

rem = $(00010)_2$ = $(2)_{10}$

quotient → take 2's complement

∴ Q = 1110

DATE / /

Logic gates

> Basic gates

1) AND gate

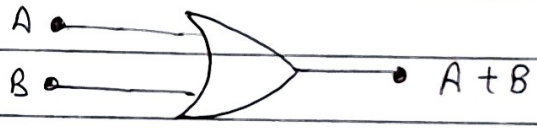
o/p is high only if all inputs are high

Boolean eq^ : A·B

A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

2) OR gate

o/p is high ~~on~~ if atleast one i/p is high



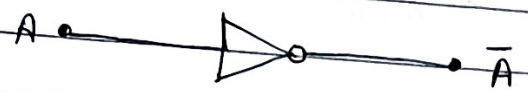
Boolean eq^ : A + B

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

PAGE No.
DATE / /

3) NOT gate

it inverts the binary i/p
i.e o/p is the opposite of the i/p.



Boolean eq^ : $\bar{A}$

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

> Universal gates

1) NAND gate

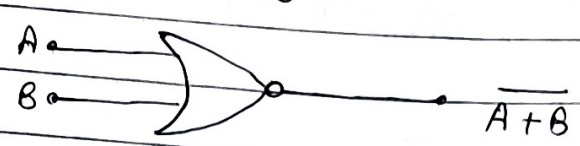
o/p is ~~#~~ low only when all inputs are high.

Boolean eq^ : $\overline{A \cdot B}$

A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

2) NOR gate

o/p is high only when all inputs are low

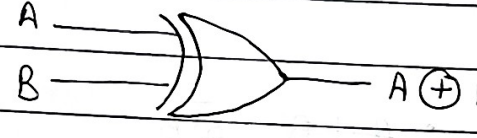


Boolean eq^n : $\overline{A+B}$

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

> EX-OR gate (Exclusive OR) / XOR

o/p is high only when inputs are different.



Boolean eq^n : $A \oplus B = A\bar{B} + \bar{A}B$

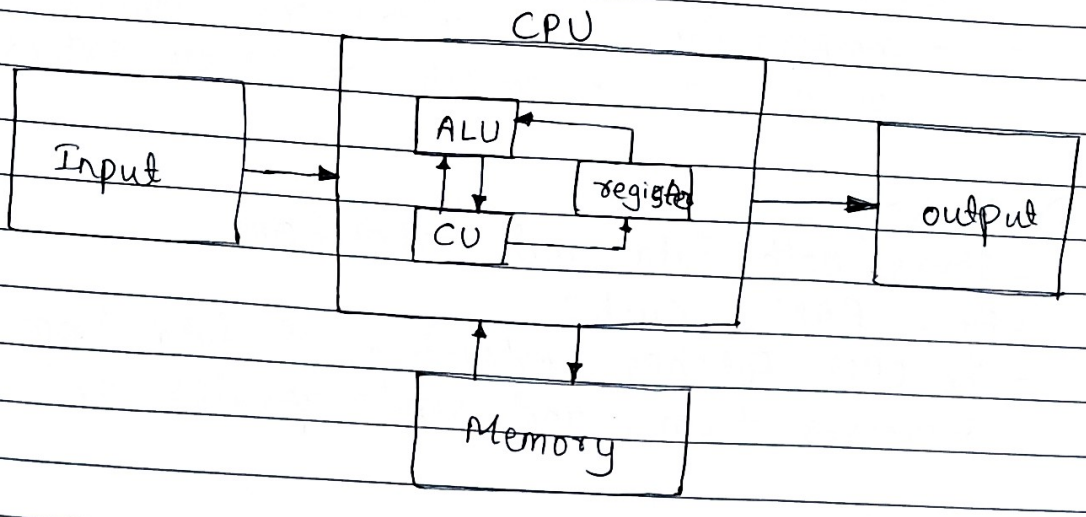
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

if $A = B \Rightarrow$ o/p low

if $A \neq B \Rightarrow$ o/p high

PAGE No.
DATE / /

Von Neuman Architecture



1) Input
- The i/p unit provides data & instructions to the computer system.
- Ex: keyboard, mouse, scanner.
- data goes from input to CPU

2) CPU (central Processing Unit)
The CPU is the brain of the computer. It is divided into different components as shown:
- ALU (arithmetic logic unit)
  - Performs arithmetic operations (addition, subtraction, multiplication, division) and logical operations (comparision, AND, OR, NOT).
  - It is connected with control unit & registers
- CU (control unit)
  - directs the flow of data & instructions b/w i/p, o/p, memory, and ALU.
  - Acts like the "traffic manager" of the CPU.
  - It fetches instructions from memory, decodes them, and tells the ALU / register what to do.

PAGE No.
DATE / /

> Registers
- Small, high-speed storage inside the CPU
- temporarily stores instructions, memory addresses, or intermediate results for faster processing.

3) Memory
- Stores both data and instructions
- Ex : RAM, Cache
- The CPU fetches instructions & data from memory, processes them, and sends results back if needed.

4) Output
- The o/p unit shows the processed data (results) to the user.
- Ex : Monitor, Printer, Speakers
- Data goes from CPU to o/p.

> Working Principle :-

1) Input gives data/instructions to CPU
2) Memory stores these data & instructions
3) Control unit (CU) fetches instructions from memory, decodes them, & co-ordinates with ALU & registers.
4) ALU performs required operations (arithmetic/logical)
5) Registers temporarily hold data & intermediate results.
6) CPU sends the final result to output unit.

PAGE No.
DATE / /

IEEE 754 Floating point representation & conversation

it is a way to represent real numbers in binary.

1) Single Precision
- > total size : 32 bits
- > Sign bit : 1 bit
- > Exponents : 8 bits
- > Mantissa (fraction) : 23 bits
- > Excess - 127

Sign	Exponent	Mantissa
1-bit	8-bit	23 bits

∵ IEEE 754 has 8 exponent bits

$2^8 = 256$ (both +ve & -ve)

$2^7 = 128$

∴ range : (0 - 127)

∴ Excess - 127

$(2^{k-1} - 1 = 2^7 - 1)$

2) Double Precision
- > total size : 64 bits
- > Sign bit : 1 bit
- > Exponent : 11 bits
- > Mantissa (Fraction) : 52 bits
- > Excess - 1023 $(2^{11-1} - 1 = 2^{10} - 1 = 1023)$

sign	Exponent	Mantissa
1-bit	11-bits	52 bits

64 bits

1) (85.125)

→ Step 1: Convert 85 (integer part) to binary

2	85	1
2	42	0
2	21	1
2	10	0
2	5	1
2	2	0
	1	1

$\therefore (85)_{10} = (1010101)_2$

Step 2: Convert 0.125 (Fractional part) to binary
Multiply by 2 repeatedly

$0.125 \times 2 = 0.25 \rightarrow 0$

$0.25 \times 2 = 0.5 \rightarrow 0$

$0.5 \times 2 = 1.0 \rightarrow 1$

$\therefore (0.125)_{10} = (0.001)_2$

$\therefore (85.125)_{10} = (1010101.001)_2$

Step 3:
Change the number in the form: $1.xx... \times 2^E$

$\therefore 1010101.001 = 1.010101001 \times 10^6$

Mantissa = 010101001
Exponent E = 6

PAGE No.
DATE / /

Step 4: Find the exponent field (8-bits)
use excess B_2 - 127 format
Encoded exponent = 6 + 127 = 133

2	133	1
2	66	0
2	33	1
2	16	0
2	8	0
2	4	0
2	2	0
	1	1

$\therefore (133)_{10} = (10000101)_2$

Step 5: Sign bit
$\because$ 85.125 is positive $\Rightarrow$ sign bit = 0

$\Rightarrow$ IEEE 754 - 32 bit (single Precision)

85.125 = | 0 | 10000101 | 01010100100000000000000 |
1-bit 8-bits 23-bits

$\longrightarrow$ For double precision
as $(85.125)_{10} = (1010101.001)_2$
also $(1010101.001)_2 = 1.01010|001 \times 10^6$

Mantissa = 01010101
Exponent E = 6

Exponent field (11-bits) $\Rightarrow$ Excess - 1023
$\therefore$ Encoded exponent = 6 + 1023 = 1029

PAGE No.
DATE / /

2	1029	1
2	514	0
2	257	0
2	128	0
2	64	0
2	32	0
2	16	0
2	8	0
2	4	0
2	2	0
	1	1

$\therefore (1029)_{10} = (10000000101)_2$

IEEE -754 : 64-bits

=>

0	10000000101	0101010010000 ..... (52 bits)

2) (74.625)

$\rightarrow (74)_{10} = (1001010)_2$

2	74	0
2	37	1
2	18	0
2	9	1
2	4	0
2	2	0
	1	1

$0.625 \times 2 = 1.250 \rightarrow 1$

$0.250 \times 2 = 0.5 \rightarrow 0$

$0.5 \times 2 = 1.0 \rightarrow 1$

PAGE No.
DATE / /

$(0.625)_{10} = (0.101)_2$

$\therefore (74.625)_{10} = (1001010.101)_2$

$= 1.001010101 \times 10^6$

$\therefore$ Mantissa = 001010101

E = 6

Encoded exponent = 6 + 127 = 133

$(133)_{10} = (10000101)_2$

→ Single Precision :

0	10000101	00101010100...0
1 bit	8 bit	23 bit

For double precision,

E = 6

M = 001010101

Excess - 1023,

Encoded-exponent = 6 + 1023 = 1029

$(1029)_{10} = (10000000101)_2$

$\therefore$ double precision :

0	10000000101	00101010100.....00
1 bit	11 bits	52 bits