* Binary Addition

⇒ The 4 basic rules

0 + 0 = 0 sum of 0 with carry 0

0 + 1 = 1 sum of 1 with carry 0

1 + 0 = 1 sum of 1 with carry 0

1 + 1 = 10 sum of 0 with carry 1

eg:— 1) 11 + 1 2) 111 + 11

⇒
```
  11          11
  011        111
+ 001      +  11
  100       1010
```

* Binary Subtraction

⇒ The 4 basic Rules

0 - 0 = 0

0 - 1 ⇒ 10 - 1 = 1 with borrow of 1

1 - 0 = 1

1 - 1 = 0

eg:— 1) 101 - 011

```
 0
 ¥10 1
  0 1 1
-------
  0 1 0
```


* Binary Multiplication

→ The 4 basic rules

0 × 0 = 0 0 × 1 = 0

1 × 1 = 1 1 × 0 = 0

eg:— 1) 11 × 11 2) 111 × 101

```
  11          111
  11        × 101
-----       -----
  11          111
+ 11        000
-----       111
 1001      -------
           100011
```

* Binary Division

→ Division in binary follows the same procedure as division in decimal

a) $110_2 \div 11_2$

```
   10
11|110
  -11
  ---
  000
```

b) $110_2 \div 10_2$

```
    11
10|110
  -10
  ---
   10
  -10
  ---
   00
```

3 | 6 | 6 | 0

2 | 6 | 6 | 6

(2)

3) $14 \times 9 = 126$

$$\begin{array}{r} 1110 \\ \times\ 1001 \\ \hline 1110 \\ 0000 \\ 0000 \\ 1110 \\ \hline 1111110 \end{array}$$

4) $7 \times 3 = 21$

$$\begin{array}{r} 111 \\ \times\ 11 \\ \hline 111 \\ 111 \\ \hline 10101 \end{array}$$

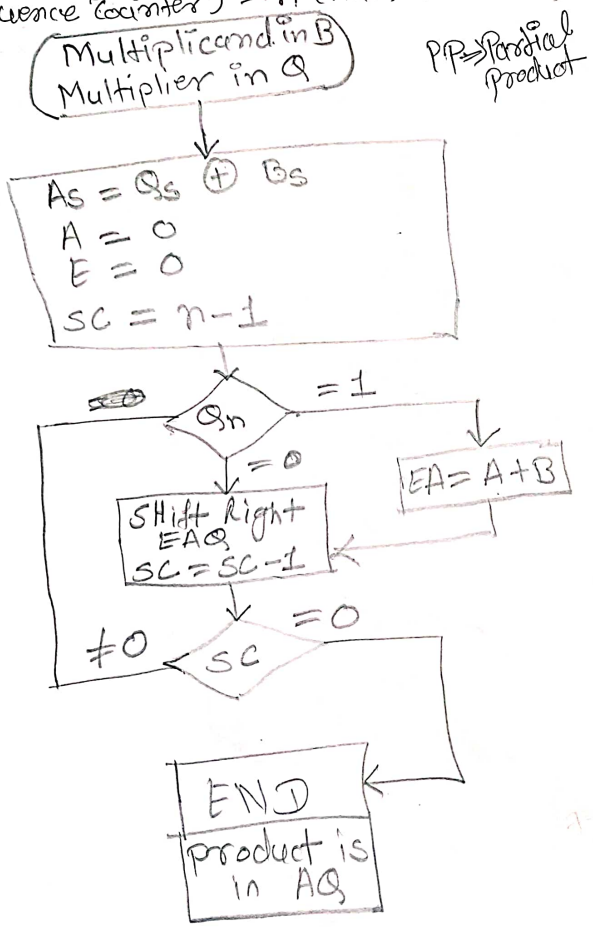
$$\begin{array}{r} 01110 \\ 10001 \overline{)\ 011110} \\ -10001 \\ \hline 01100 \\ -10001 \\ \hline 01011 \\ -10001 \\ \hline 00010 \\ -00000 \\ \hline 00101 \\ -10001 \\ \hline 00100 \\ -10001 \\ \hline 00011 \end{array}$$

A = 01110 (14)
B = 10001 (17)

Remainder $(6)_{10}$

Multiply Operation :-

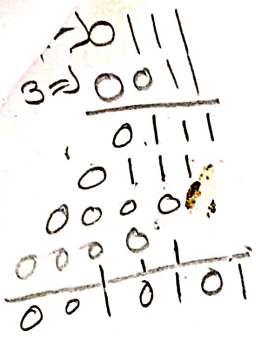
As (sign of A), Qs (sign of Q)
Bs (sign of B),
SC (Sequence Counter) = n (number of bit in Q)

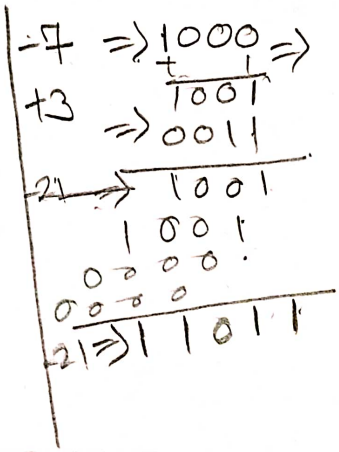


*Multiplication of Signed-Magnitude Data / Number

B = 10111 (Multiplicand)
Q = 10011 (Multi

operation	E	A	Q	SC
Initial	0	00000	10011 Q_n	101
$Q_n = 1$		A 00000		100 4
EA ← A+B	0	B+ 10111		
PP1 →				
shEAQ	0	01011	11001 Q_n	
SC ← SC-1				
$Q_n = 1$		01011		011 3
EA ← A+B	1	+10111		
PP2 → shEAQ		10000	01100 Q_n	
SC ← SC-1	0			
$Q_n = 0$				010 2
~~EA ← A+B~~	0	01000	10110 Q_n	
~~PP3~~ → shEAQ				
SC ← SC-1				
$Q_n = 0$				001 1
shEAQ	0	00100	01011 Q_n	
SC ← SC-1				
$Q_n = 1$		00100		0000
EA ← A+B	0	10111		
PP3 → shEAQ		11011		
SC ← SC-1	0	01101	10101	

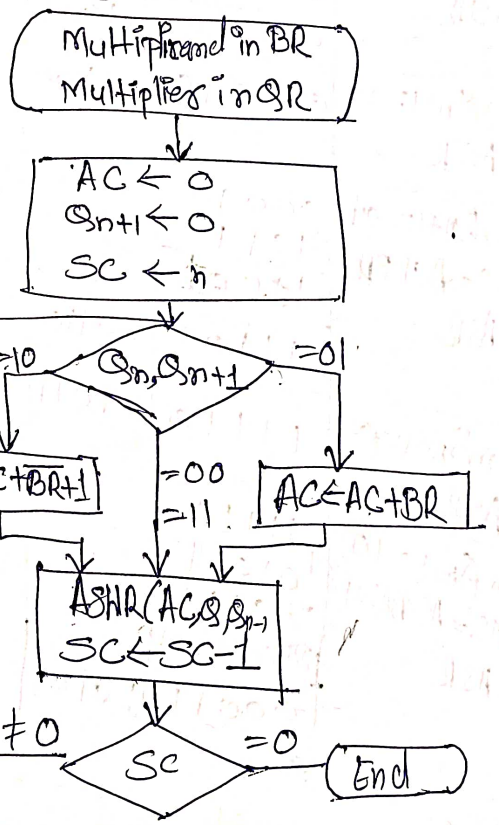


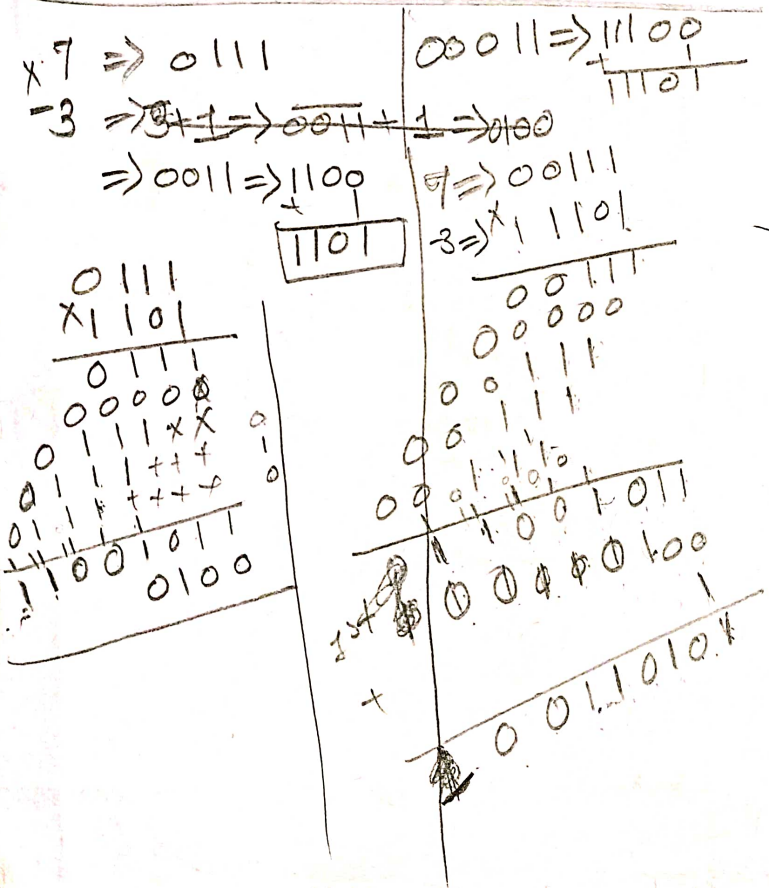


* Signed Operand Multiplication

⇒ Booth's Algorithm

- One of the most elegant and widely used schemes for twos complement multiplication was proposed by Andrew D. Booth in 1951

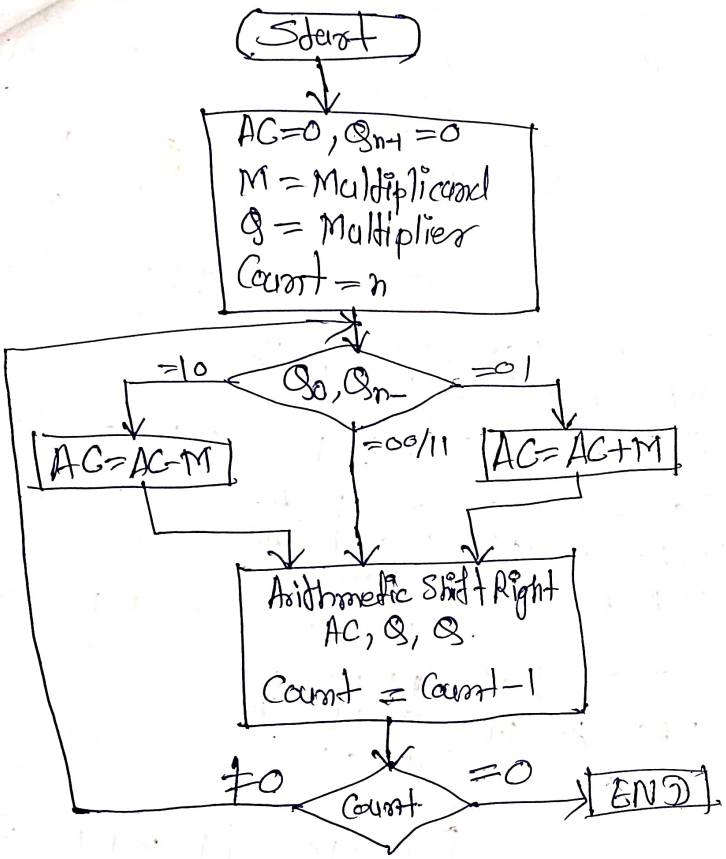




BR = 10111
QR = 10011

$\overline{BR}$ = 01000 + 1
$\overline{BR}+1$ = 01001

Operations	AC	QR	Q_{n+1}	SC
Initial	00000	10011	0	5
$Q_n, Q_{n+1}=10$	00000			5
i) $AC=AC+\overline{BR}+1$	+ 01001			
	01001			
ii) ASR	00100	11001	1	4
$Q_n Q_{n+1}=11$	00010	01100	1	3
ii) ASR				
$Q_n, Q_{n+1}=01$	00010			
i) $AC=AC+BR$	+ 10111			
	11001			
ii) ASR	11100	10110	0	2
$Q_n, Q_{n+1}=00$	11110	01011	0	1
i) ASR				
$Q_n Q_{n+1}=10$	11110			
i) $AC=AC+\overline{BR}+1$	+ 01001			
	00111			
	Discard carry			
ii) ASR	00011	10101	1	0



Multiply 7 and 3 using Booth's Algorithm

→ Register size = 4 bits AC = AC + (M)

(M) → $(7)_{10} \Rightarrow (0111)_2$ AC = AC

(Q) → $(3)_{10} \Rightarrow (0011)_2$

(−M) → 2's comp $(0111)_2 \Rightarrow (1001)_2$

1's comp ⇒ 1000

2's comp ⇒ + 1

$(1001)_2$

Operation	AC	Q	Q_{n+1}
	0000	0011	0
i) AC = AC − M	0000 (A)		
	+ 1001 (−M)		
ii) ASR AC, Q, Q_{n+1}	1001		
	1100	1001	1
$Q_0, Q_{-1} = 11$	1110	0100	1
i) ASR			
$Q_0, Q_{-1} = 01$	1110 (A)		
i) AC = AC + M	+ 0111 (M)		
	0101		
ii) ASR	Discard carry		
	0010	1010	0
$Q_0, Q_{-1} = 00$	0001	0101	0
i) ASR			
	$(7)_{10} \Rightarrow$ 0111		
	$(3)_{10} \Rightarrow$ 0011		
	$(21)_{10}$	$(00010101)_2$	

Multiply −7 and +3 using Booth's Algorithm

→ Register size = 5 bits

M → $(-7)_{10}$ → $(11001)_2$

Q → $(+3)_{10}$ → $(00011)_2$

$(-M) = (7)_{10}$ → $(00111)_2$

0 00111 +7

11000 1st

+ 1 2nd

11001

Operation	AC	Q	Q-1	
	00000	00011	0	
Q, Q-1 ⇒ 10 i) AC = AC − M ii) ASR	00000 + 00111 00111			5th
	00011	10001	1	
Q, Q-1 ⇒ 11 i) ASR	00001	11000	1	4th
Q, Q-1 ⇒ 01 i) AC = AC + M ii) ASR	00001 + 11001 11010			3rd
	11101	01100	0	
Q, Q-1 ⇒ 00 i) ASR	11110	10110	0	2nd
Q, Q-1 ⇒ 00 i) ASR	11111	01011	0	1st
−7 × 3 −21	11111 01011 + 00000 10100 ⇒ 1's 2nd 00000 10101			

(−ve) × (−ve) = + Product

Multiply −7 and −3 using Booth's Algorithm

→ Register size = 4 bits

M → $(-7)_{10}$ → 1001

−M → $(7)_{10}$ → 0111

Q → $(-3)_{10}$ → $(1101)_2$

0111 1st 1000 2nd + 1 1001

Operation	AC	Q	Q-1
	0000	1101	
Q, Q-1 → 01 i) AC = AC − M ii) A.SR	0000 A + 0111 +(−M) 0111		
	0011	1110	1
Q, Q-1 ⇒ 10 i) AC = AC + M ii) ASR	0011 A 1001 +(m) 1100		
	1110	0111	0
Q, Q-1 ⇒ 01 i) AC = AC − M ii) ASR	1110 A 0111 +(−m) 0101 Discard carry		
	0010	1011	1
Q, Q-1 ⇒ 11 i) ASR	0001	0101	1
−7 × −3 +21	1001 1101 00010101		

Datatype Representation

i) Fixed Point Number
ii) Floating Point Number

i) Fixed Point Number :-

$(+7)_{10} \Rightarrow (0111)_2$ in 4bit

$(-7)_{10} \Rightarrow$? step 1: 1st complement $0111 \Rightarrow (1000)_2$

step 2: 2nd complement $+1000$

$(-7)_{10} \Rightarrow (1001)_2$

ii) Floating Point Representation

It has three Part
a) Mantissa
b) Base
c) Exponent

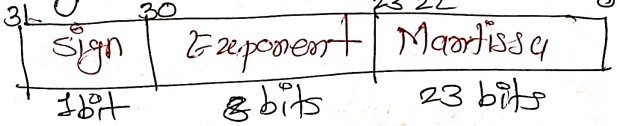
$\pm M \times B^{\pm E}$

Number	Mantissa	Base	Exponent
1) 3×10^1	3	10	6
2) 110×2^8	110	2	8
3) 6132.784	6132784	10	−3
4) 9×10^8	9	10	8
5) 43647.84	4364784	10	−3

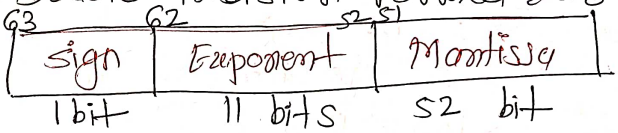
(Institute of Electrical and Electronic Engineers)

IEEE754 Floating Point Number Representation.

a) Single Precision Format :- 32 bit



b) Double Precision Format :- 64bit



Eg:- Represent $(259.125)_{10}$ in single and double precision format.

step 1:- Convert decimal Number to Binary Number

$(259)_{10} = (100000011)_2$

$(0.125)_{10} \Rightarrow 0.125 \times 2 = 0.250 \times 2 = 0.500 \times 2 = 1.000$

$(259.125)_{10} = (100000011.001)_2$

step 2:- Normalize the Number

$(1.M)_2 E-127 \Rightarrow$ Single Precision

$(1.M)_2 E-1023 \Rightarrow$ Double Precision

$\Rightarrow 1001110101 1.001$

$\Rightarrow \underbrace{1.00111010111001}_{N} \times 2^{10}$

Step 3 Single Precision Format

$(1.N)_2 E-127$

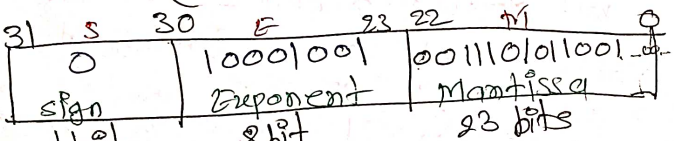
$1.00111010111001 \times 2^{10}$

$E-127=10$

$E=10+127$

$\boxed{E=137}$

$E=(10001001)_2$



Step 4 Double Precision Format

$(1.N)_2 E-1023$

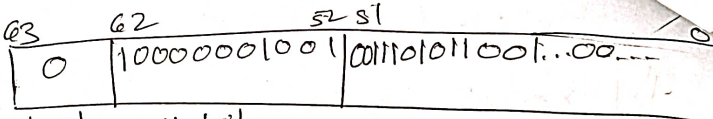
$1.00111010111001 \times 2^{10}$

$E-1023=10$

$E=1023+10$

$E=1033$

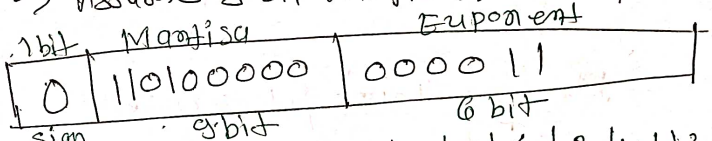
$E=(10000001001)_2$



eg- Covert the following floating point binary number into decimal.

$\Rightarrow 1\,0110\,000000000011$

$\Rightarrow$ Assume 1 bit mantissa & 6 bit Exponent



32	16	8	4	2	1	$\Rightarrow 1+2=3$
0	0	0	0	1	1	

$\Rightarrow$

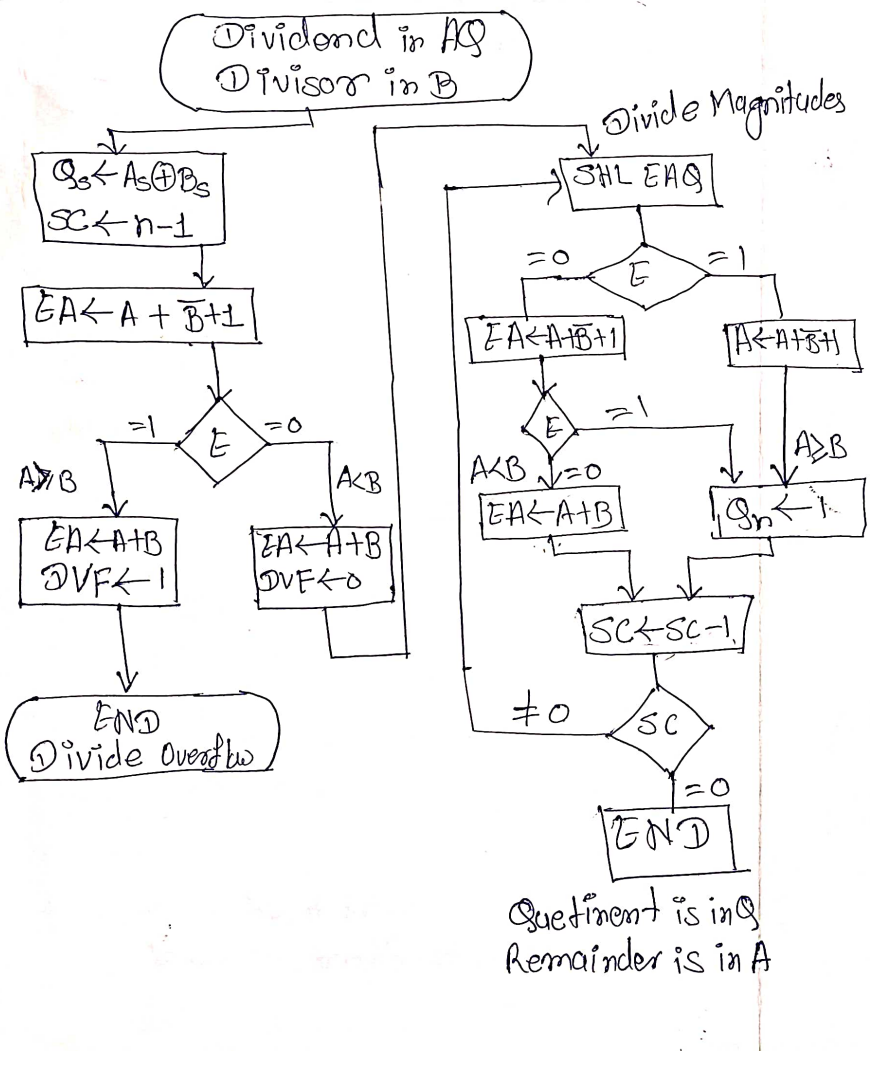
$\Rightarrow 2^3\ 2^2\ 2^1\ 2^0 \quad 2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}\ 2^{-5}\ 2^{-6}$

$0\ 1\ 1\ 0 . 1\ 0\ 0\ 0\ 0\ 0 \times 2^3$

4	2	1	0.5
1	1	0	1

$\Rightarrow (6.5)_{10}$

Division :—



A=01110 B=10001 $\overline{B}$=01110 ⇒ $\overline{B}$+1 ⇒ 01111

E	A	Q	sc	Operation
0	01110	00000	5	SHL
0	11100 + A	00000		SHL A+$\overline{B}$+1
	01111 $\overline{B}$+1			
1	01011	00001		SHE
0	10110 A	00010	4	SHL A+$\overline{B}$+1
	01111 $\overline{B}$+1			
1	00101	00011		
0	01010 A	00110	3	SHL A+B+1
	01111 $\overline{B}$+1			A+B
	11001 A			
0	10001 B			
1	01010		2	
0	10100 A	01100		SHL A+$\overline{B}$+1
	01111			
1	00011	01101	1	
0	00110	11010		SHL A+$\overline{B}$+1 A+B
	01111			
0	10101			
	10001			
	00110		0	

Quetient

final Remainder

Representations of Binary Floating Point Number.

$(5.625)_{10} \rightarrow (101.101)_2$

$(101101, 3)$

Radix point is present after the 3rd bit from MSb.

$(5.625)_{10} \rightarrow 0.5625 \times 10^1$

mantissa, base, Exponent

$(101.101)_2 \rightarrow 0.101101 \times 2^3$

S	Exponent	Mantissa

Need of Normalization :—

$(101.101)_2 \rightarrow 0.101101 \times 2^3$

$(101.101)_2 \rightarrow 1.01101 \times 2^2$

$(101.101)_2 \rightarrow 0.0101101 \times 2^4$

$(101.101)_2 \rightarrow 101101 \times 2^{-3}$

So many different Representation
So we need Normalization form

Normalization :—

- Two types

1) Explicit Normalization :—

- Move the radix point to the LHS of the most significant '1' in the bit sequence

$(101.101)_2 \rightarrow 0.101101 \times 2^3$

2) Implicit Normalization :—

- Move the radix point to the RHS of the most significant '1' in the bit sequence.

$(101.101)_2 \rightarrow 1.01101 \times 2^2$

Biasing :—

S	Exponent	Mantissa

$(101.101)_2 \rightarrow 0.101101 \times 2^3$

$(101.101)_2 \rightarrow 101101 \times 2^{-3}$

-8 (+8)	-7 (+8)	-6 (+8)	-5 (+8)	-4 (+8)	-3 (+8)	-2 (+8)	-1 (+8)	0 (+8)	1 (+8)	2 (+8)	3 (+8)	4 (+8)	5 (+8)	6 (+8)	7 (+8)
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Exponent Excess 8

Signed Number Representations :-

-2^3	2^2	2^1	2^0	2_5
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	-8
1	0	0	1	-7
1	0	1	0	-6
1	0	1	1	-5
1	1	0	0	-4
1	1	0	1	-3
1	1	1	0	-2
1	1	1	1	-1

bias => single = 127
Double = 1023

$E' = E + bias$

$(1460.125)_{10}$
$\Rightarrow (10110110100.001)_2$
$\Rightarrow$ Normalization
$1.0110110100001 \times 10^{10}$

$S = 0$
$E = 10$
$m = 01101101000001$
$E' = 10 + 127 = 137$
binary $= 10001001$

S	E	
0	10001001	01101101000001...0

32 bit Single

S	E	
0	10000001001	01101101000001....0

64 bit Double

$S = 0, E = 10, E' = 10 + 1028 = 1038 = (10000001001)$

$(0.01)_2 \rightarrow 0.101 \times 2^{-1}$

$(101.101)_2 \rightarrow 1.01101 \times 2^2$

S	Exponent	Mantissa
0	0 1 1 1 1	0 1 0 1 0

$(101.101)_2$

0	1	0	1	0	0	1	1	0	1

Formula :—

1) Explicit Normalization

$(-1)^S \times 0.M \times 2^{E-Bias}$

2) Implicit Normalization

$(-1)^S \times 1.M \times 2^{E-Bias}$

$2^{8-1} - 1 \Rightarrow 2^7 - 1$

$\Rightarrow 128 - 1$

$\Rightarrow 127$

2

* Explicit Vs. Implicit Normalization

$(5.625)_{10} \rightarrow (101.101)_2$

S	E (4bit)	M (5 bit)

10 bit

$(101.101)_2 \rightarrow 0.10110 \times 2^3$

$(101.101)_2 \rightarrow 1.01101 \times 2^2$

S	E				M				
0	1	0	1	1	1	0	1	1	0

S = 0

E = 3 + 8 = 11 $\Rightarrow (1011)_2$

M = 10110

our M is 5 bit so removing last 0

$= (-1)^S \times 0.M \times 2^{E-Bias}$

$= (-1)^0 \times 0.10110 \times 2^{11-8}$

$= (1011)_2$

$= (5.5)_{10}$

2	5	1
2	2	0
2	1	1

$0.5 \times 2 = 1$

S	E				M				
0	1	0	1	0	0	1	1	0	1

S = 0

E = 2 + 8 = 10 $\Rightarrow (1010)_2$

M = 01101

$= (-1)^S \times 1.M \times 2^{E-Bias}$

$= (-1)^0 \times 1.01101 \times 2^{10-8}$

$= (101.101)_2$

$= (5.625)_{10}$

2	5	1
2	2	0
2	1	1

$0.625 \times 2 = 1.25 = 1 + 0.25$

$0.25 \times 2 = 0.5 = 0 + 0.5$

$0.5 \times 2 = 1 = 1 + 0.0$

Binary Arithmetic

1) Binary Addition

Rules:-

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 10$ carry bit

eg:- $1010 \Rightarrow 2^3 \times 1 + 2^2 \times 0 + 1 \times 2^1 + 2^0 \times 0$

$0111 \Rightarrow 8 + 0 + 2 + 0 \Rightarrow 10$

$1001 \Rightarrow 8 + 0 + 0 + 1 \Rightarrow 9$

$16 + 0 + 0 + 0 + 1 = 17$

2) Binary Subtraction

Rules:-

$0 - 0 = 0$

$0 - 1 \Rightarrow 10 - 1 = 1$

$1 - 0 = 1$

$1 - 1 = 0$

eg:- $1010 - 0111 \Rightarrow 1010 \Rightarrow 0111 \Rightarrow 0011$

$0010 \Rightarrow 0111 \Rightarrow 0011$