Number Systems

Classification of the Number System:-

Number System

- Weighted number system (positional) e.g. binary, octal, decimal, hexadecimal, BCD
- Non-weighted number system (non-positional) e.g. Gray code, excess-3 code, etc.

• Weighted Number System

A no. system with base (or) radix "r" contains 'r' diff. digits which are in the range of 0 to r-1, e.g.

	Base	Range
	2	0,1
	8	0 to 7
	10	0 to 9
	16	0 to F
hypothetical	4	0 to 3
hypothetical	7	0 to 6
hypothetical	12	0 to B

Conversion from Decimal to other base:
To convert decimal no. into other base, (r) system, divide the integer part & multiply the fractional part with base r repeatedly until the integer part & fractional part becomes zero.

During the division process, take the remainder value & apply division opn on quotient repeatedly until the quotient part becomes 0, later take bottom to top remainders to conclude the integer part conversion code.

During multiplication process, take the integer value & apply the multiplication on fractional part repeatedly until fractional part becomes zero or repeated to original fraction. later take top to bottom integer values to conclude the fractional part conversion code.

e.g.

$(29.625)_{10} \Rightarrow (x)_2 / (x)_4 / (x)_8$

$(29.625)_{10} = (x)_2$

$(29)_{10} = (11101)_2$

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

$0.625 \times 2 = 1.250 \rightarrow 1$

$0.250 \times 2 = 0.500 \rightarrow 0$

$0.500 \times 2 = 1.000 \rightarrow 1$

$(0.625)_{10} = (101)_2$

$(29.625)_{10} = (11101.101)_2$

$(29.625)_{10} = (x)_4$

$$\begin{array}{r|rl} 4 & 29 & \\ \hline 4 & 7 & 1 \\ \hline & 1 & 3 \end{array}$$

$\cancel{(29)_4 = 1}$

$(29)_{10} = (131)_4$

$0.625 \times 4 = 2.500 \rightarrow 2$

$0.500 \times 4 = 2.000 \rightarrow 2$

$\therefore (29.625)_{10} = (131.22)_4$

$(29.625)_{10} = (x)_8$

$$\begin{array}{r|l} 8 & 29 \\ \hline & 3 \quad 5 \end{array}$$

$(29)_{10} = (35)_8$

$0.625 \times 8 = 5.000 \rightarrow 5$

$(29.625)_{10} = (35.5)_8$

<u>Conversion from other base to decimal</u>

To convert the other base(r) system into decimal there is a need of considering the positional weights i.e.

$\sum_{i=-1}^{-n} \sum_{i=0}^{n-1} r^i * a_i$ r :- base/radix
i :- position
a :- given no.

Given no. is "N" :-

$\ldots + N_2 * r^2 + N_1 * r^1 + N_0 * r^0 + N_{-1} * r^{-1} + N_{-2} * r^{-2} \ldots$

e.g.

e.g $(35.5)_8 = (x)_{10}$

$3 \times 8^1 + 5 \times 8^0 + 5 \times \frac{1}{8}$

$= 24 + 5 + 0.625$

$= (29.625)_{10}$

$$\begin{array}{r} .625 \\ 8)\overline{50} \\ -48 \\ \hline 20 \\ -16 \\ \hline 40 \end{array}$$

Q1 What is the min. decimal equivalent of the no. 11C.0 ?

(a) 183
(b) 194
(c) 268
(d) 269

Base = (Max digit + 1)
$= (C+1)$
$= (12+1) = 13$

$12 \times 13^0 + 1 \times 13^1 + 1 \times 13^2$
$= 12 + 13 + 169 = 194$

Q2. The no. of digit 1 present in the binary representation of $3 \times 512 + 7 \times 64 + 5 \times 8 + 3$ is

(i) 8
(ii) 9
(iii) 10
(iv) 12

$(3753)_8 = (011\ 111\ 101\ 011)_2$

or

$\Rightarrow (2^1+2^0) \times 512 + (2^2+2^1+2^0) \times 64 + (2^2+2^0) \times 8 + 2^1 + 2^0$
$= 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^3 + 2^1 + 2^0$

In this any power of 2 will contain only 1 '1', e.g. $2^{10} = 1024 = (1000\,000\,000)_2$

∴ total powers of 2's :- $\boxed{9}$.

Q3. How many 1s are present in the binary representation of the following expression:-
$4 \times (4096) + 9 \times 256 + 7 \times 16 + 5$

$(0100\ \ 1001\ \ 0111\ \ 0101)_2$

Ans. 8

$2^2 \times 2^{12} + (2^3 + 2^0) \times 2^8 + (2^2 + 2^1 + 2^0) \times 2^4 + 2^2 + 2^0$
$= 2^{14} + 2^{11} + 2^8 + 2^6 + 2^5 + 2^4 + 2^2 + 2^0$

8

Q4. What are the values ~~of~~ R_1 & R_2 in exp.

$(235)_{R_1} = (565)_{10} = (1065)_{R_2}$

~~16 | 565~~

16 | 565
16 | 35   5
   | 2   3

$$\begin{array}{r} 35 \\ 16\,)\,565 \\ -48 \\ \hline 85 \end{array}$$

$(235)_{16}$

∴ $R_1 = 16$
$R_2 = 8$

(a) 8, 16
~~(b) 16, 8~~
(c) 6, 16
(d) 12, 8

(a) & (c) can't be the answer
∴ check for (b) & (d).

Q5. The decimal equivalent of hexadecimal no. :-

$(2A0F)_{16} = 2\times16^3 + 10\times16^2 + 15\times1$
$= 10767$

$(0010\ 1010\ 0000\ 1111.)_2$

$1+2+4+8+512+\cancel{20}$

Data Representation

Fixed point data is represented in diff. possible formats.
The classification is shown below :-

- Data Representation
  - Magnitude representation
    - unsigned representation
    - sign-magnitude representation
  - Complement representation
    - Diminished radix complement representation
    - Radix complement representation

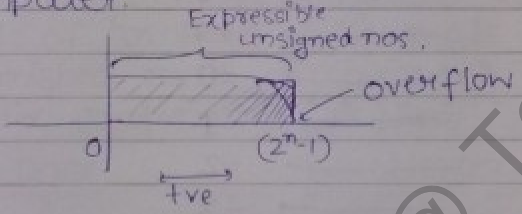
Unsigned Representation:-

In this format we can only represent the +ve nos., ∴ there is no sign bit reservation, so the complete format indicates the value.

n-bit unsigned range is $\{0 \text{ to } (2^n-1)\}$

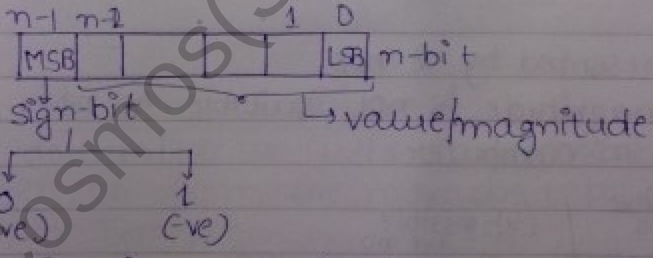
e.g. 4-bit unsigned range is 0 to $2^4-1 = 0$ to 15

In this format, there is no ambiguous values, this format is suitable to represent +ve nos. in the computer.



Sign-magnitude representation:-

In this format we can represent both +ve & -ve nos., ∴ there is a need of sign-bit reservation, the general format of sign magnitude representation is:



n-bit sign magnitude range is
$\{(-2^{n-1}-1) \text{ to } (+2^{n-1}-1)\}$

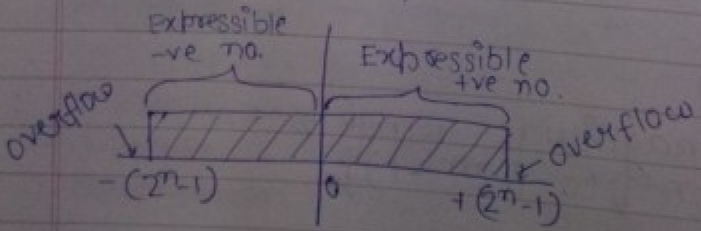
e.g.
4-bit $\Rightarrow \{-7 \text{ to } +7\}$

total nos = 15, but with 4-bit we can have 16 combinations.

Sign Magnitude Binary	Equivalent Decimal
0000	+0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

sign value

★ 0 is represented by 2 nos
Sign-magnitude is not suitable to represent nos. in computer.



0 = 0000
0 = 1000

Note: In this format 0 is associated with 2 different representations, ∴ sign magnitude format is not suitable to represent the signed nos. in the computer, so there is a need of alternative format to represent signed nos., i.e. complement representation.

Complement Representation :-

Any no. system with base 'r' ~~retares~~ contain two kinds of complement formats.
(i) Diminished radix complement (r-1's)
(ii) Radix complement (r's)

e.g.

base	complement format
2	1's, 2's
4	3's, 4's
8	7's, 8's
10	9's, 10's
12	~~A's~~, B's ~~A's~~, 12's
15	~~D's~~ E's, 15's
16	F's, 16's

(r-1)'s Complement :-

To determine the (r-1)'s complement, subtract the given no. from the max. digit possible in the given base, i.e.

(r-1's) complement of no. $N = [(r^n - 1) - N]$

r :- radix/base
N :- given no.
n :- no. of digits in the give no.

eg 9's complement of $(789)_{10}$

$[(10^3 - 1) - 789]$

$$\begin{array}{r} 999 \\ -\ 789 \\ \hline 210 \end{array}$$

e.g.

1's complement of $(101)_2$ is :-

$$\begin{array}{r} 111 \\ -101 \\ \hline 010 \\ \hline \end{array}$$

$\Rightarrow [(2^3-1)-(101)_2]$

$= [(7)_{10} - (101)_2]$

$= [(111)_{10} - (101)_2]$

$= (010)_2$

e.g. 3's complement of $(23)_4$ is :-

$$\begin{array}{r} 33 \\ -23 \\ \hline 10 \\ \hline \end{array}$$

$$\begin{array}{r|l} 4 & 15 \\ \hline & 3 \quad 3 \end{array}$$

$[(4^2-1)_{10} - (23)_4]$

$= [(15)_{10} - (23)_4]$

$= [(33)_4 - (23)_4]$

$= (10)_4$

r's Complement:-

To determine the r's complement, calculate the (r-1)'s complement & add 1 to the LSB.

★ Computer supports binary no. system, so in the complement format the data is represented in 1's complement & 2's complement.

- 1's complement format:- In this format, we can represent both +ve & -ve nos., ∴ there is a need of the sign-bit representation.
- When the MSB of the no. is 1, the sign is -ve, so take the 1's complement to ~~co~~ conclude the value.

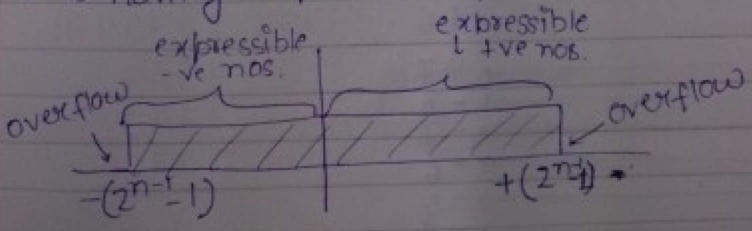
- When the MSB bit is '0', then the sign is +ve, so no need to take the complement to conclude the value
- n-bit 1's complement range is :-

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

e.g. 4 bit → {-7 to +7}

1's complement Binary	Equivalent Decimal
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 → 1's complement 0111 → 7	-7
1 0 0 1	-6
1 0 1 0	-5
1 0 1 1	-4
1 1 0 0	-3
1 1 0 1	-2
1 1 1 0	-1
1 1 1 1	-0

0 is having 2 possible representation.



0 = 0000
0 = 1111

★ In this format '0' have 2 possible representation
∴ it is not suitable to represent the signed nos
in the computer.

2's complement

- In this format we can represent both +ve & -ve nos. So, sign bit reservation is required.
- When the MSB is '1', sign is -ve, so take the 2's complement to conclude the value.
- When the MSB is '0', then the sign is +ve, So no need to take the complement to conclude the value.

n-bit 2's complement range is:-

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

e.g. 4 bit :- {-8 to +7}

2's Complement binary	equivalent decimal
0000	+0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	→ +7
1000	→ -8
1001	→ -7
1010	→ -6
1011	→ -5
1100	→ -4
1101	→ -3
1110	→ -2
1111	→ -1

+1
1001
0110
-1
100

Note: In this format, each binary sequence have ~~user~~ its own unique meaning, so no possibility of ambiguous representation, . this format is suitable to represent the signed nos. in the computer.

Fixed Point Arithmetic

① The possible opn on fixed pt. data is
(i) addition
(ii) subtraction
(iii) multiplication
(iv) division

Addition & Subtraction :-
In the computer, Subtraction op^n is performed by using the addition unit, i.e 2's complement addition is called as subtraction :-
e.g. A-B ⇒ A+(-B) ⇒ A + 2's complement of B

Q1. Perform addition b/w the following nos.
$(ADD)_{16}$ & $(DAD)_{16}$

$$\begin{array}{r} ADD \\ +\ DAD \\ \hline 188A \end{array}$$

16 | 26
   | 1 A

Q2 Perform subtraction b/w the following
$(FACE)_{16}$ & $(CAFE)_{16}$

$$\begin{array}{r} FACE \\ CAFE \\ \hline 2FD0 \\ \hline \end{array}$$

15 + 12 - 15 ; 16 + 12 = 28

25 - 10 = 15 ; 14 - 12

Q3. Perform the XOR op^n b/w the following nos.

$(FE35)_{16}$ & $(CB15)_{16}$

FE35 1111 1110 0011 0101

CB15 XOR 1100 1011 0001 0101

0011 0101 0010 0000

3 5 2 0

ans → 3520

Q. In the signed magnitude repn the binary equivalent of decimal 22.5625 is

(a) 0, 10110.1011

(b) 0, 10110.1001

(c) 1, 10101.1001

(d) 1, 10110.1001

$(22)_{10} = (10110)_2$

$(0.5625)_{10} =$

$0.5625 \times 2 = 1.1250$

Q. Which one of the following represents $(E3)_{16}$

(a) $(1CE)_{16} + (A2)_{16}$

(b) $(1BC)_{16} - (DE)_{16}$

(c) $(2BC)_{16} - (1DE)_{16}$

(d) $(200)_{16} - (11D)_{16}$

(a) is not the answer

(b) → × 1BC − DE = E

12 − 14

28 − 14 = 14

15 − 9

(c) → × 2BC − 1DE = E

(d) 200 − 11D = E3

16 − 13

Q. $(2.3)_4 + (1.2)_4 = (Y)_4$

$Y = ?$

$$\begin{array}{r} 2.3 \\ +\ 1.2 \\ \hline 10.01 \end{array}$$

$4 \,|\, 5$ — $1\ 1\ 0$ $\quad$ $4 \,|\, 4$ — $1\ 0$

Q. 11001, 1001 & 111001 corresponds to the 2's complement representation of which one of the following sets of the no.

(a) 25, 9, & 57
(b) -6, -6 & -6
(c) -7, -7, -7
(d) -25, -9, -57

$a_2 a_1 a_0 \xrightarrow[\text{it in 8-bits}]{\text{to represent}} a_2 a_2 a_2 a_2 a_2 a_2 a_1 a_0$

in 2's complement

Multiplication

In the manual multiplication process, 2 operations are performed :-

(i) partial product generation
(ii) Summation

Based on the multiplier, we can calculate the partial products, i.e. if the multiplier bit is '1', then the partial product is multiplicand, if the multiplier bit is '0', then the partial product is 0, after calculation of the partial product, perform the summation opn. to generate the final product.

In the manual multiplication process, disadvantage is more no. of registers are needed to store the partial products & more no. of addition opn are required to generate the final product.

e.g. $\underbrace{1111}_{\text{multiplicand}} \times \underbrace{1111}_{\text{multiplier}}$

```
      101
      111111   }
     11111X    } Partial
  10 1111XX    } Product
     1111XXX   }
  ---------------
  11100001  } Final Product
```

15
×15
225

Q.1 Consider the following multiplication process
$(10W1Z)_2 * (15)_{10} = (Y0\ 1011001)_2$
what are the values of W, Y & Z

```
       10W1Z
      × 1111
   ----------
        10W1Z
       10W1Z×
      10W1Z××
     10W1Z×××
   ----------
   101011001
```

$\boxed{\begin{array}{l} Z=1 \\ W=1 \\ Y=1 \end{array}}$

$1+W=0$

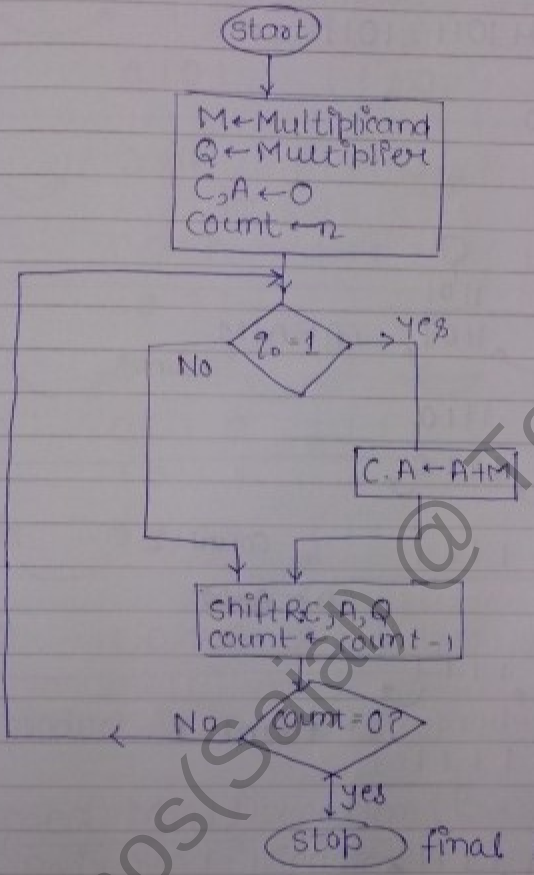
$1+W=0$

Note: To handle the disadvantages in the manual multiplication process, there is a need of accumulating addition opⁿ, this process reduces the no. of register counts & no. of arithmetic addition counts.

Unsigned Multiplication

When the multiplication opⁿ is performed b/w 2 n bit nos., then it generates o/p in the size of 2n bits.

Unsigned multiplication process is described in the following flow chart:-



final product is in A & Q reg.

Q. Perform the multiplication b/w

11 * 13

$\underbrace{1011}_{\text{multiplicand}}$ $\underbrace{1101}_{\text{multiplier}}$

M = 1011

Q = 1101

C,A = 0000

count = 4

Step 1:-

$q_0 = 1$

$\therefore$ C.A = A+M

C A = 0000+1011 = 1011

C.A = 1011

count = 3 ≠ 0

Shift

	C	A	Q	
	0	0000	1101	
$q_0=1$	0	1011	1101	CA ← A+M
				count=3
	0	0101	1110	
$q_0=0$	0	0101	1110	
	0	0010	1111	count=2
$q_0=1$	0	~~0010~~ 1101		CA ← A+M
	0	1111	1111	
				count=1
	0	0111	1111	
		1101		
$q_0=1$	1	0100	1111	
	0	1010	0111	count=0

product is in A & Q.

$\therefore$ product is

	C	A	Q	
	0	0000	1101	
$q_0 = 1$	0	1011	1101	CA ← A+M
	0	0101	1110	count = 3
$q_0 = 0$	0	0101	1110	
	0	0010	1111	count = 2
		1011		
$q_0 = 1$	0	1101	1111	
	0	0110	1111	count = 1
		1011		
$q_0 = 1$	1	0001	1111	count = 0
	0	1000	1111	

∴ product is in A & Q, ∴ product is $(10001111)_2 = (143)_{10}$

Signed Multiplication (Booth's Algorithm)
(Bit-Pair Multiplication)

(i) In the signed multiplication process, arithmetic shift op^n are required to maintain same sign before & after op^n.

(ii) In this process diff op^n are performed by considering the multiplier bit pairs.

(iii) The bit pair op^n are listed below:

Bit Pair		Operation
00	$\longrightarrow$	ASR (Arithmetic Shift right)
01	$\longrightarrow$	add, ASR
10	$\longrightarrow$	sub, ASR
11	$\longrightarrow$	ASR

(iv) Bit pairs are formulated based on the multiplier in this process $q_{-1} = 0$ to pair with q_0 & q_{-1}, i.e.

multiplier bit pairs:-

q_0	$q_{-1}(0)$
q_1	q_0
q_2	q_1
$\vdots$	$\vdots$
q_{n-2}	q_{n-3}
q_{n-1}	q_{n-2}

Q. Consider Booth's multiplication process, if multiplier is 1011 0111 1011 0101, how many arithmetic op^n are required during the multiplication?

Ans. 1011 0111 1011 0101 0

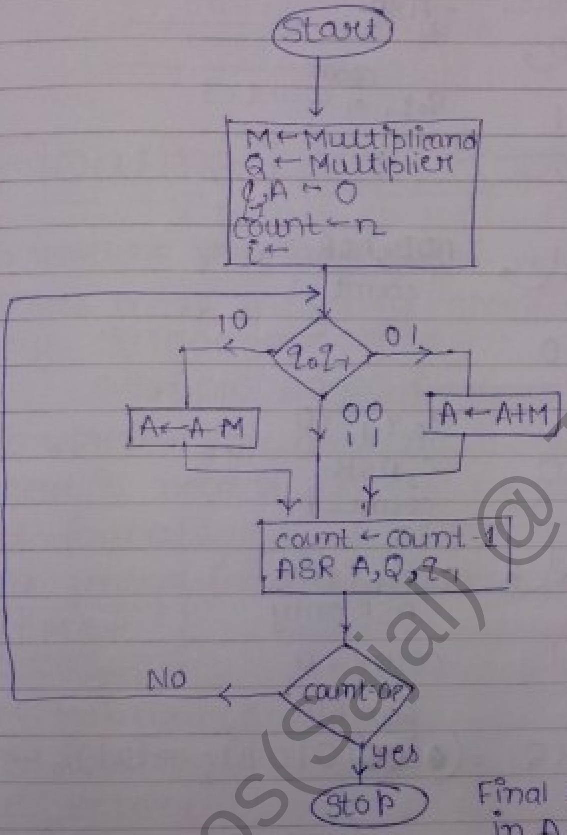
q_{15} ... $q_1 q_0\ q_{-1}$

Bit pairs

1 0 ✓
0 1 ✓
1 0 ✓
0 1 ✓
1 0 ✓ $\longrightarrow$ 11
1 1
0 1 ✓
1 0 ✓
1 1
1 1
1 1
0 1 ✓
1 0 ✓
1 1
0 1 ✓
1 0 ✓

Arithmetic op^n can be add or sub, which can happen in bit pairs sequence as 01, 10.

★ Signed multiplication process is represented in the below flow chart.



Final product in A & Q register

-7 * -3

0111 1001 * 1101 ← 2's complement format

1000 Multiplicand Multiplier

0011

1100 Bit pairs

q_0	1	q_{-1}	0	⟶ −, ASR
q_1	0	q_0	1	⟶ +, ASR
q_2	1	q_1	0	⟶ −, ASR
q_3	1	q_2	1	⟶ ASR

0111
1000
+ 1
1001
+ 1
1010
0101

M = 1001 ; count = 4

A	Q	q_{-1}	
0000	1101	0	$q_0 q_{-1} = 10$
1001			$-$, ASR
0111	1101	0	($A = A - M$)
0011	1110	1	$q_0 q_{-1} = 01$ count = 3
1001			
1100	1110	1	ADD, ASR
1110	0111	0	count = 2
1001			$q_0 q_{-1} = 10$
0101	0111	0	$-$, ASR
0010	1011	1	count = 1
			$q_0 q_{-1} = 11$
0001	0101	1	ASR only, count = 0

$\therefore$ product in A & Q $= (00010101)_2 = (21)_{10}$ (sign bit)

Division

In the division process, dividend must be expressed in 2n bits & divisor in n bits, in the division opn the dividend is scanned from MSB to LSB. In each bit scan, the value is compared with the divisor, if it is less than divisor, then keep 0 in the quotient & scan the next bit, if the value is greater than or equal to divisor, then keep 1 in the quotient & subtract the divisor from the dividend. Continue the process until all the bits of the dividend is completed.

e.g: $7 \div 3$

```
      00000010
0011 ) 00000111
      -   0011
      ---------
          00001
      ---------
```

Floating point Representation

1. To represent very large nos & very small fractions takes more memory space.
   e.g 98700000000000
   ~~0.987~~ 0.000000000000987
2. To represent the above nos in the limited memory space, there is need of special format (known as floating pt format).
3. The general form of floating pt format is
   $\boxed{\pm M * B^{\pm e}}$
   $\pm$: sign
   M : Mantisa/significand
   B : Base/radix
   e : exponent
4. To store very large nos. & very small fractions with limited mem space, convert the nos. into floating pt format by alligning the decimal digits.

- $987000000000000 * 10^{0}$
  after 14 shift right op^n, the no. is :-
  $\boxed{9.87 * 10^{14}}$
- $0.00000000000987 * 10^{0}$
  after 11 shift left op^n, the no. will be .
  $\boxed{9.87 * 10^{-11}}$

After converting the no. into floating pt format, we can store the no. into the memory by using the floating point representation (layout).
Floating pt. layout contains 3 fields:-
(i) sign (mantissa sign & not exponent sign)
(ii) biased exponent
(iii) normalised mantissa
the hypothetical floating pt layout is (an example)

20 bit

Sign	Biased exponent	Mantissa
1 bit	6 bit	13 bit

- sign: this field indicates the sign of the floating point no., when it is 1, then the sign is -ve, otherwise '+ve'.
- biased exponent: it is equal to actual exponent + bias. Biased exponent differentiates the +ve & -ve exponent values.

Bias is the max. possible +ve exponent. The range of exponents are depending on the size of the biased exponent field in the floating pt format, ∴ Bias value also depends upon the biased exponent field size i.e.

Biased Exponent field size	Range of exponents	Bias
6 bit	-2^{6-1} to $+2^{6-1}-1$ = -32 to +31	+31
10 bits	-2^{10-1} to $+2^{10-1}-1$ ~~= -1024 to +1023~~ = -512 to +511	+511
n-bit	-2^{n-1} to $+2^{n-1}-1$	$+2^{n-1}-1$

BE = Actual exponent + Bias

- Bias ⇒ max. possible +ve exponent

if BE > Bias (+ve exponent)

BE ≤ Bias (-ve exponent)