Storing Real Numbers

→ int
- Unsigned → Arithmetic
- Signed int → −ve & +ve number
  - → Direct
  - → 1's Complement } −0 & +0
  - → 2's Complement ← Modern Computer
    - → −ve Number
    - → Weightage Code

→ Float Numbers : The number that Consists point

Ex 3.14159 , 2.718 , 6.022×10^{-23}

float

- fixed Point → the location of Point is fixed Ex: 1.758, 37.95
- floating Point → the location of Point floats from left to right or vice-versa (opposite)

Ex: 6.022×10^{-23}, 1.75×10^{4} ← Scientific Notation

60.375×10^{-5}

(A) Fixed Point Representation

If fixed point representation of a number is

Integral Part | Fractional Part

I I I I . F F F F

So we can store minimum value 0000.0001 & Maximum Value will be 9999.9999

Three Parts

(a) Sign bit (sign field)

b) Integral Part (integer field)

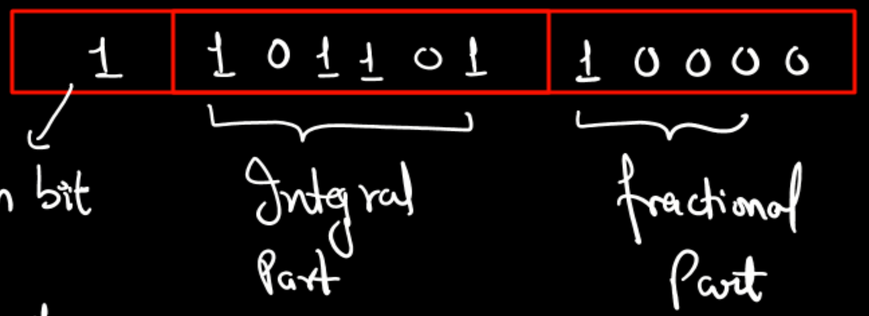
c) Fractional Part (fraction)

} Signed Number

* Bits are fixed to store fixed Point Number

- integral Part
- fractional Part
- Sign

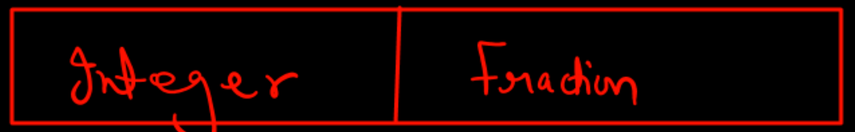
Ex: $-45.5 \rightarrow 0.5$



} 12 bits fixed Point Number

- 1 bit ← sign
- 6 bits ← integral
- 5 bits ← fractional

Unsigned Number Representation ← No Sign bit



Range in fixed Point representation

for 'k' bits :

Signed Representation : $-(2^{(k-1)} - 1)$ to $(2^{k-1} - 1)$ ← Do Not use

Unsigned Representation : $(0$ to $2^k - 1)$

Signed 2's Complement Representation :

(-2^{k-1}) to $(2^{k-1} - 1)$

(B) Floating Point Representation

$\Rightarrow (5.625)_{10} \longrightarrow (101.101)_2$

0.5625×10^{1} — Exponent

- 0.5625 — Mantissa
- 10 — Radix
- 1 — Exponent

0.101101×2^{3} — Exponent

- 0.101101 — Mantissa
- 2 — Radix
- 3 — Exponent

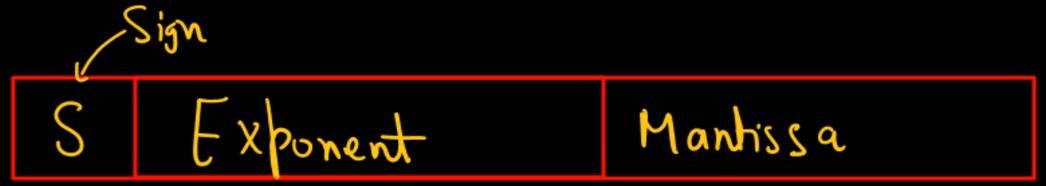
$(101.101)_2$

1.01101×2^{2}

0.101101×2^{3}

Normalizing

Representation :→



0 011 101101

$\Rightarrow (0\ 011\ 101101)_2$

Normalization : आसान तरीका

a) **Explicit Normalization** : Move the radix point to LHS of the Most Significant '1' in the Sequence

$(1\,0\,1\,.\,1\,0\,1)_2 \longrightarrow 0.1\,0\,1\,1\,0\,1 \times 2^3$

b) **Implicit Normalization**: Move the radix point to the RHS of the Most Significant '1' in the Sequence

$(1\,0\,1\,.\,1\,0\,1)_2 \longrightarrow 1.0\,1\,1\,0\,1 \times 2^{(2)}$ ← Exponent

Mantissa

Biasing : जितने Bits का Exponent होता है उसकी Range के सबसे बड़े Number को हमें Add करना होता है Exponent के साथ ताकि Exponent +ve हो जाए।

$(0.0011.01)_2 \rightarrow 1.01 \times 2^{(-3)}$ ← Negative

2's Complement form → Range :: (-2^{k-1}) to $(2^{k-1}-1)$

2's Complement..

4 bits = 2^4

-2^{4-1} to $2^{4-1}-1$

$= -2^3$ to 2^3-1

$= [-8$ to $+7]$

Suppose we have 4 bit Exponent

Largest Value (Ignore sign)

−8	−7	−6	−5	−4	−3	−2	−1	0	1	2	3	4	5	6	7
+8	+8	+8	+8	+8	+8	+8	+8	+8	+8	+8	+8	+8	+8	+8	+8
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

} 16 combinations

Without Bias = 1.01×2^{-3}

With Bias = $1.01 \times 2^{-3+8}$

$= 1.01 \times 2^{5}$ ← +ve

Converting the Number into actual form:

a) Explicit Normalization : $(-1)^{S} \times 0.M \times 2^{\text{Exponent} - \text{Bias}}$, $S \in \{0, 1\}$

b) Implicit Normalization : $(-1)^{S} \times 1.M \times 2^{\text{Exponent} - \text{Bias}}$

$-1^{0} = 1$

$-1^{1} = -1$

Ex $(101.101)_2 \longrightarrow$ Explicit : 0.101101×2^{3}

$\hookrightarrow 0.1011 01 \times 2^{3+8}$

$= 0.\underline{101101} \times 2^{11}$

Ex Exponent = 4bits

4 bits = 16

-2^{k-1} to $2^{k-1}-1$

-8 to 7

$\Rightarrow (0\ 1011\ 101101)_2$

$(11)_2 \rightarrow 1011$

Implicit Mode: $(10.101) \rightarrow 1.01101 \times 2^{2}$

Adding Bias = $1.01101 \times 2^{2+8}$

$= 1.01101 \times 2^{10}$ ← Based Exponent

Exponent = 4 bits → 16 comb.

2's comp = -2^{k-1} to $2^{k-1}-1$ → -8 to $+7$

$(10)_{10} \rightarrow (1010)_{2}$

Sign → 1 bit

Exponent → 4 bit

5 bits

=) total = 10 bits → 2 Byte

Suppose → Exponent → 10 bits → 000000 1010 ← Bit Extension

Mantissa → 10 bits → 01101 00000

bit Extension

IEEE Standard :

↳ Precision

→ Half Precision : (16 Bit)

↳ 1 bit → Sign

5 bits → Exponent

10 bits → Mantissa

→ Single Precision (32 bit)

↳ 1 bit → Sign

8 bit → Exponent

23 bits → Matissa

→ Double Precision (64-bit)

↳ 1 bit = Sign

11 bits = Exponent

52 bits = Mantissa

→ Quadruple Precision (128 bit)

↳ 1 bit → Sign

15 bit ⇒ Exponent

112 bit ⇒ Mantissa