in $\bar{x}\bar{y}z + xy\bar{z}$

* By using carry bits : Let Cin be the carry into the MSB & Cout be carry out from MSB

$$Cin \oplus Cout = \begin{cases} 0 & \text{No overflow} \\ 1 & \text{Overflow} \end{cases}$$

When overflow occurs, no. of bits is increased by 1 (i.e. increasing the range).

7) Floating & fixed point representation

→ $(54.3)_{10}$ $(71.25)_8$ $(1110.001)_2$

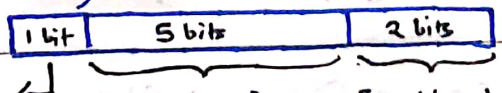
Decimal point. Octal point Binary point.

$(X_3X_2X_1 \cdot X_{-1}X_{-2})_8$

Radix Point.

→ Fixed point representation : To represent any number, bit selection is predecided (number is represented with a fixed no. of digits before & after the decimal point).

Ex : 8 bit no.



Sign bit ↵ Integer Part Fractional Part.

00000 → 0 00 → 0

01 → 0.25

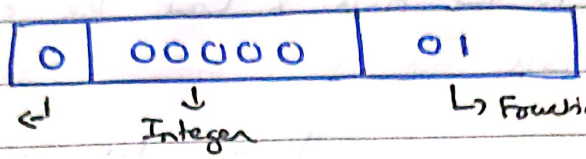
10 → 0.5

11111 → 31 11 → 0.75

Neelgagan
Date:
Page No.

Ex Represent $(0.3125)_{10}$ in the above fixed point representation.

Ans $(0.3125)_{10} = (0.0101)_2$



→ We could have easily represented 0.0101 using 5 bits but we have to take the approximate value of $(0.3125)_{10}$ {Disadvantage of FPR}

→ If we assign more bits for the integer part, then possible range increases but if we assign more bits for the fraction part, then precision increases.

→ Fixing the no. of bits for integer part & fraction part is the disadvantage of FPR (fixed point representation), therefore we move into floating point representation

→ Floating point representation

Better Precision

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

↓ Earlier, this was saved as follows:-

[101101, 3]

↓

Radix point is present after the 3rd bit from MSB

Neelgagan

Date:

Page No.

$(5.625)_{10} \rightarrow 0.5625 \times 10^{4}$ ← Exponent

Mantissa

$(101.101)_2 \rightarrow 0.101101 \times 2^{3}$ ← Exponent

Mantissa

S	Exponent	Mantissa

* Need of normalization

$(101.101)_2$ →
- $0.101101 * 2^3$
- $1.01101 * 2^2$
- $0.0101101 * 2^4$
- $101101 * 2^{-3}$

Different Representations

* Normalization : Basically are two types :—

a) Explicit normalization : Move the radix point to the LHS of most significant '1' in the bit sequence

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

Better Precision

b) Implicit normalization : Move the radix point to the RHS of most significant '1' in the bit sequence.

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

S	Exponent	Mantissa

↓

Exponent can be 'true' or '-ve'

Date:
Page No.

Exponent-

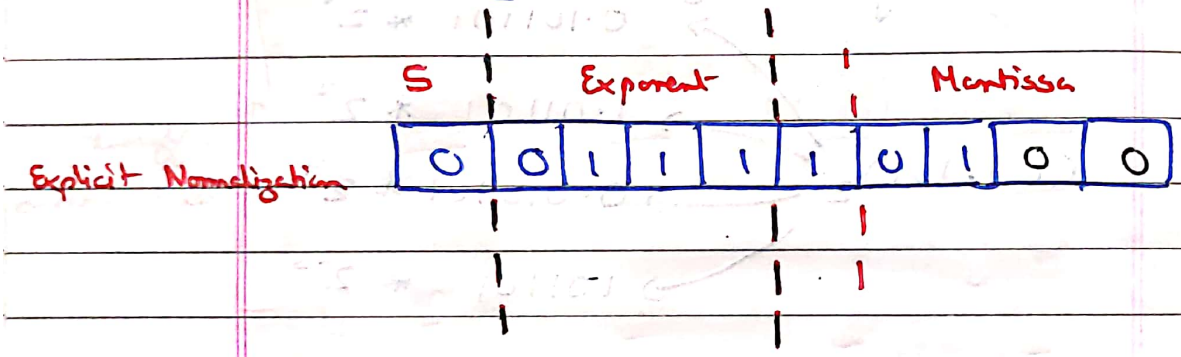
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7

↓ +8 (Biasing)

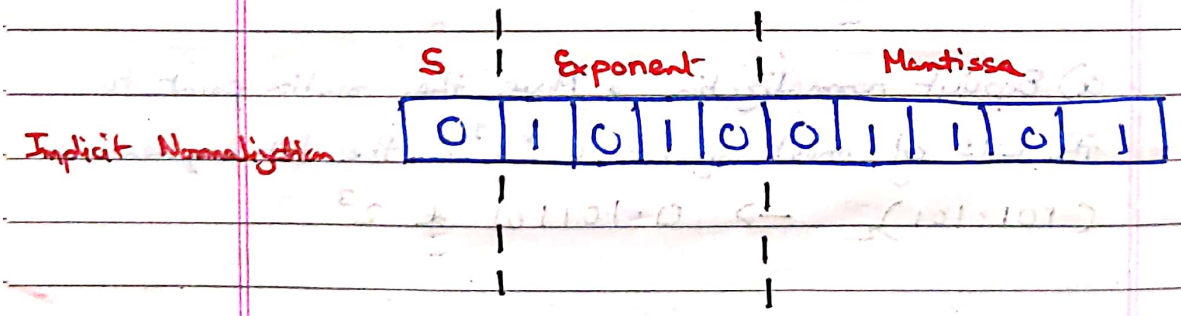
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Exponent is called excess 8

$(0.0101)_2 \rightarrow 0.101 * 2^{-1}$



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



Formula

i) Explicit Normalization : $(-1)^S * 0.M * 2^{E-Bias}$

ii) Implicit Normalization : $(-1)^S * 1.M * 2^{E-Bias}$

Neelgagan

Date:

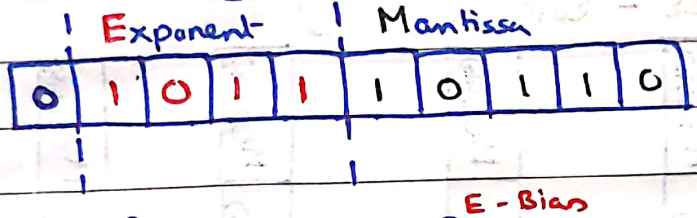
Page No.

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

Let us represent this using both explicit & implicit normalization

Explicit → Normalization

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



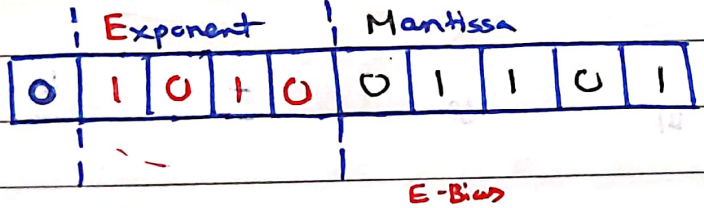
$(-1)^S * 0.M * 2^{E-Bias}$

$(101.1)_2 \rightarrow (5.5)_{10}$

More Precision

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

Implicit → Normalization



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

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

Although, we have come across explicit & implicit normalization, however the required bits to store a particular floating point number is yet to be fixed. Therefore IEEE has come across with a standard

Neelgagan

Date:

Page No.

→ IEEE 754 standard (1985)

↳ Responsible for standardization of floating point numbers.

↳ IEEE - 1985 (Anst)

IEEE - 2008

IEEE - 2019 (Revised)

IEEE 754

Name	Common Name	Significand Bits	Exponent Bits	Exponent Bias	Organization of Bits
binary16	Half Precision	11	5	15	S | E(5bits) | M(10bits)
binary32	Single Precision	24	8	127	S | E(8bits) | M(23bits)
binary64	Double Precision	53	11	1023	S | E(11bits) | M(52bits)
binary128	Quadruple Precision	113	15	16383	S | E(15bits) | M(112bits)
binary256	Octuple Precision	237	19	262143	S | E(19bits) | M(236bits)

Date:
Page No.

→ IEEE 754 - Single Precision

S(bit)	E(8 bits)	M(23 bits)	Representation
0 or 1	0000 0000 (E=0)	0000.....0 (M=0)	± ●
0 or 1	1111 1111 (E=255)	0000.....0 (M=0)	±∞
0 or 1	1 ≤ E ≤ 254	M = xxx.... x	Implicit Normalized Form
0 or 1	E = 0	M ≠ 0	Fractional Form
	E = 255	M ≠ 0	NAN

→ IEEE 754 - Double Precision

S(bit)	E(11 bits)	M(52 bits)	Representation
0 or 1	000 0000 0000 (E=0)	000....0 (M=0)	± 0
0 or 1	111 1111 1111 (E=2047)	000....0 (M=0)	± ∞
0 or 1	1 ≤ E ≤ 2046	M = xxx... x	Implicit Normalized Form
0 or 1	E = 0	M ≠ 0	Fractional Form
	E = 2047	M ≠ 0	NAN

Date:
Page No.

Ex 1 Consider a 32 bit register which stores the Floating Point numbers in IEEE single precision format,

i) Find the decimal value of the following 32 bits

0	1000 0011	1100 0

ii) Find the decimal value of the following 32 bits

0	1111 1111	1100 01

Sol i) Biased exponent : 131 (1000 0011)

Bias (Weight) : 127

Value = $(-1)^S * 1.M * 2^{E-Bias}$

$= (-1)^0 * 1.11 * 2^{131-127} = (111\,00)_2 = (28)_{10}$

ii) Biased exponent : 255 & $M \neq 0$

Represent NAN

Ex 2 Consider a 64 bit register which stores the Floating point numbers in IEEE double precision format,

i) Find the decimal value of the following 64 bits :-

1	100 0000 0011	1101100.... 0

ii) Find the decimal value of the following 64 bits :—

0	111 1111 1111	0000 0

Date:
Page No.

Sol i) Biased Exponent : 1027 (100 0000 0011)

Bias (Weight) = 1023

True exponent = 1027 - 1023 = 4

Value : $(-1)^S * 1.M * 2^{E-Bias}$

$(-1)^1 * 1.11011 * 2^4 = (-11101.1)_2$

$\updownarrow$

$(-29.5)_{10}$

ii) Biased exponent : 2047 & M=0 (S=0) $\rightarrow +\infty$

Ex3 Represent $(116)_{10}$ in IEEE 754 single & double precision format.

Sol $(116)_{10} \rightarrow (111\ 0100)_2$

Implicit Normalized form : $1.110100 * 2^6$

→ Single precision : Biased exponent = 6 + 127 = 133

0	1000 0101	110100 ... 0

→ Double precision : Biased exponent = 6 + 123 = 1029

0	100 0000 0101	110100 ... 0

Ex4 Determine the range of (+ve) floating point numbers represented using fractional form of IEEE 754 single precision format.

Sol Fractional form : $(-1)^S * 0.M * 2^{-126}$

Smallest fraction : $(-1)^0 * 0.000\ldots01 * 2^{-126} = 2^{-23} * 2^{-126} = 2^{-149}$

2^{-23}

Neelgagan

Date:

Page No.

Largest fraction : $(-1)^0 * 0.111\ldots11 * 2^{-126}$

2^{-1} 2^{-2} 2^{-3} 2^{-22} 2^{-23}

$= \left(\frac{1}{2} + \frac{1}{2^2} \cdots \frac{1}{2^{23}}\right) * 2^{-126}$

$= \left[\frac{\frac{1}{2}\left\{\left(\frac{1}{2}\right)^{23} - 1\right\}}{\frac{1}{2} - 1}\right] * 2^{-126} = (1 - 2^{-23}) * 2^{-126}$

Range : $2^{-149} \longleftrightarrow (1 - 2^{-23}) * 2^{-126}$

Ex5 Add the following numbers in 8-bit register using signed 2's complement notation.

a) 50 & -5 b) 45 & -65 c) 75 & 85

Cout = Cin : No overflow

Carry out — Sign Bit 0 — 1

Carry in — Sign Bit 1 — 0

Overflow

Underflow

Sol a) 2's com : 50 — 0 0 1 1 0 0 1 0

-5 — 1 1 1 1 1 1 0 1 1

1 0 0 1 0 1 1 0 1

Answer : 45

Carry in = Carry out : No underflow/overflow

Answer is positive (Neglect 1)

Date:
Page No.

b) 45 : 0 0 1 0 1 1 0 1 (carries: 1 1 1 1 1 1)

-65 : + 1 0 1 1 1 1 1 1

1 1 1 0 1 1 0 0

0 0 0 1 0 0 1 1

Carry in = Carry out = 0 : No overflow / underflow

0 0 0 1 0 1 0 0 → 20

No Carry : Take 2's complement of final answer

Answer is negative (-20)

c) 75 : 0 1 0 0 1 0 1 1 (carries: 1 1 1 1 1 1 1)

85 : 0 1 0 1 0 1 0 1

1 0 1 0 0 0 0 0

Carry out ≠ Carry in : Overflow

Carry out = 0, Carry in = 1

Normalization of floating point numbers : A floating point number is said to be normalized if the most significant position of the mantissa is a non-zero.

$(37.25)_{10} = (100101.01)_2 = 1.0010101 * 2^5$

$(7.625)_{10} = (111.101)_2 = 1.11101 * 2^2$

$(0.3125)_{10} = (0.0101)_2 = 1.01 * 2^{-2}$