COMPUTER SCIENCE

Computer Organization and Architecture

Floating Point Representation

Floating Point Representation

Lecture_01

Vijay Agarwal sir

01 Floating Point Representation

① Introduction of COA

② Instr format & AM.

Number System

Floating Point Representation

Magnitude → Unsigned (only +ve Number)

Magnitude → Signed (+ve & -ve Number)

Signed → 0 [+ve]

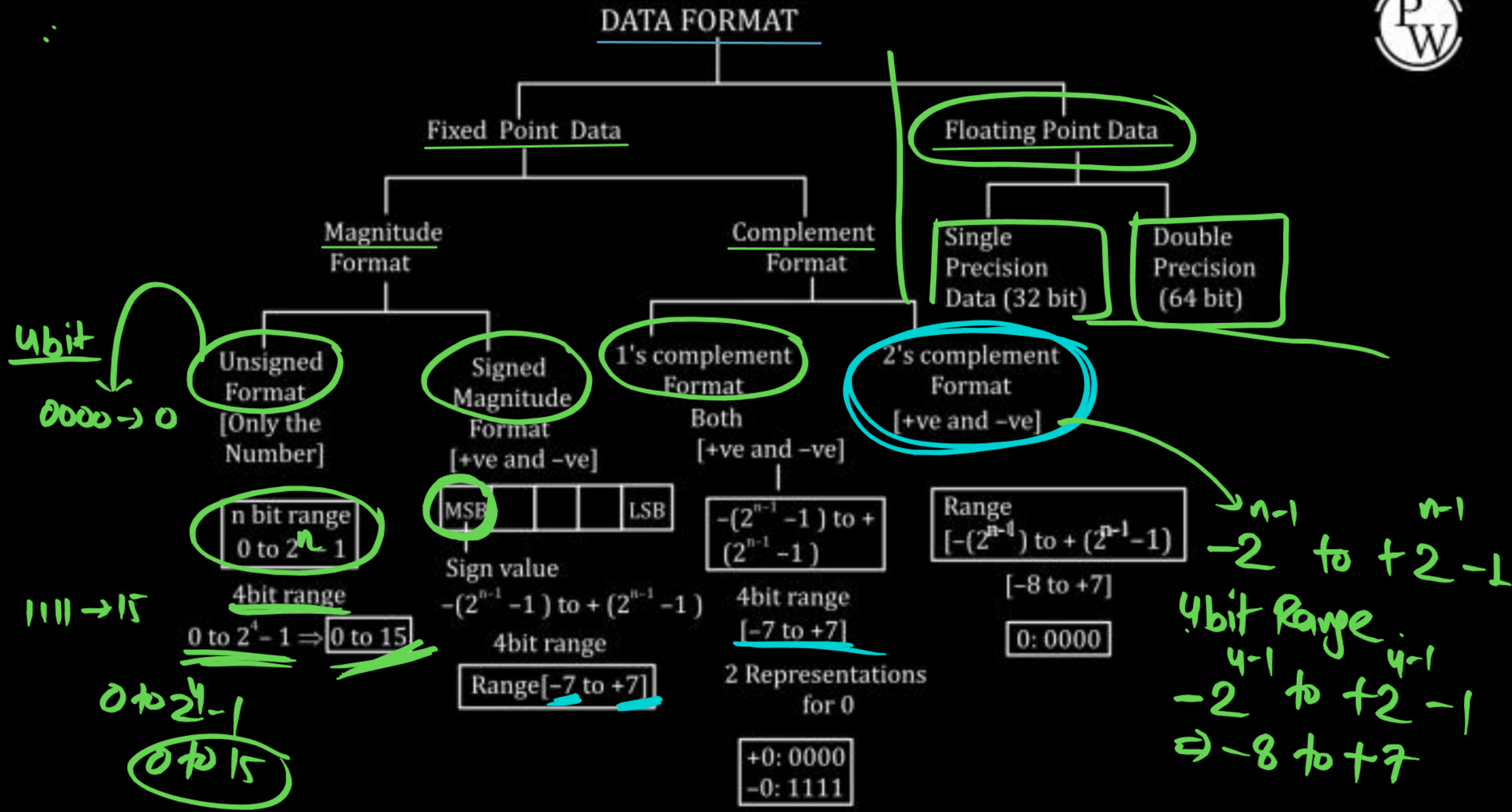
Signed → 1 [-ve]

Why 2's Complement:

In Signed Magnitude & 1's Complement 0 has Redundant Representation

+0

-0


DATA FORMAT
Fixed Point Data
Floating Point Data
Magnitude Format
Complement Format
Single Precision Data (32 bit)
Double Precision (64 bit)
Unsigned Format [Only the Number]
Signed Magnitude Format [+ve and –ve]
1's complement Format
Both [+ve and –ve]
2's complement Format [+ve and –ve]
4bit
0000 → 0
n bit range 0 to 2^n – 1
4bit range
0 to 2^4 – 1 ⇒ 0 to 15
1111 → 15
0 to 2^4 – 1
0 to 15
MSB
LSB
Sign value
–(2^(n–1) –1) to + (2^(n–1) –1)
4bit range
Range[–7 to +7]
–(2^(n–1) –1) to + (2^(n–1) –1)
4bit range
[–7 to +7]
2 Representations for 0
+0: 0000
–0: 1111
Range [–(2^(n–1)) to + (2^(n–1)–1)]
[–8 to +7]
0: 0000
–2^(n–1) to +2^(n–1) –1
4bit Range
–2^(4–1) to +2^(4–1) –1
⇒ –8 to +7

DE | DLD.

1's Complement

1's → 0

0 → 1

1011

⇓

1's Complement : 0100

Flags

Carry
Parity
Sign
Zero
Auxiliary
Overflow

Conditional Flag

2's Complement

1's Complement + 1.

1's Complement

1000 ⇒ 0111 = -7

1001 ⇒ 0110 = -6

1010 ⇒ 0101 = -5

1011 ⇒ 0100 = -4

1111 ⇒ 0000 = -0

2's Compl.

0111 +1 1000 ⇒ -8

0110 +1 0111 ⇒ -7

0101 +1 0110 = -6

0100 +1 0101 ⇒ -5

0000 +1 0001 ⇒ -1

4 bit Binary	Unsigned Data	Signed Magn. Data	1's Complement Data	2's Complement Data
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	-7	-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

Redundant Representations for '0'

Note

2's Complement are used in the Computer System to Represent the Negitive Number.

'n' bit 2's Complement Range $= -\left(2^{n-1}\right)$ to $+\left(2^{n-1}-1\right)$

Eg 4 bit 2's Complement Range $= -\left(2^{4-1}\right)$ to $+\left(2^{4-1}-1\right) \Rightarrow$ Ans -8 to $+7$

Floating-Point Representation

Principles

- With a fixed-point notation it is possible to represent a range of positive and negative integers centered on or near 0.
- By assuming a fixed binary or radix point, this format allows the representation of numbers with a fractional component as well
- Limitations:
  - Very large numbers cannot be represented nor can very small fractions
  - The fractional part of the quotient in a division of two large numbers could be lost

Why floating point Representation : ?

Very - Very large Number $\Rightarrow$ [Nearly ∞] 9764.12345.....

Very - Very Small Number $\Rightarrow$ [0] 00.....-12

16 bit Number $\Rightarrow -\left(2^{16-1}\right)$ to $+\left(2^{16-1}-1\right)$

$\Rightarrow -2^{15}$ to $+2^{15}-1$

$\Rightarrow$ **-32k to +32k -1**

If we want

Assume 51,000 (51K) Not Possible with 16 bit Data

Floating-Point Representation

16 bit fixed point data format then

Range $= -2^{16-1}$ to $+ (2^{16-1} - 1)$

$\Rightarrow$ $-(2^{15})$ to $+ (2^{15} - 1)$

If we want to store 61,000 then we cannot store

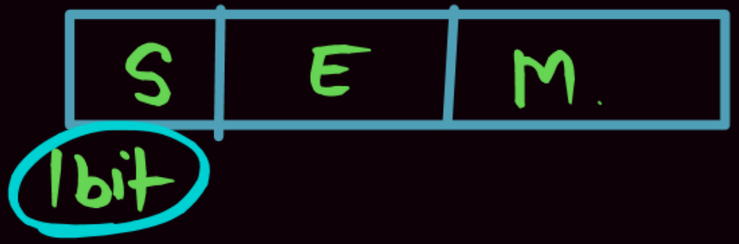
Because range [–32k to + 32 k –1]

So floating point representation is to represent very large data and very small fraction and consume less memory

Floating point used to represent

- $+ 8564100000000000 \ldots [\Rightarrow \infty]$
- $+ 0.000000000007892 \Rightarrow [\Rightarrow 0]$

FLOATING POINT Rep.

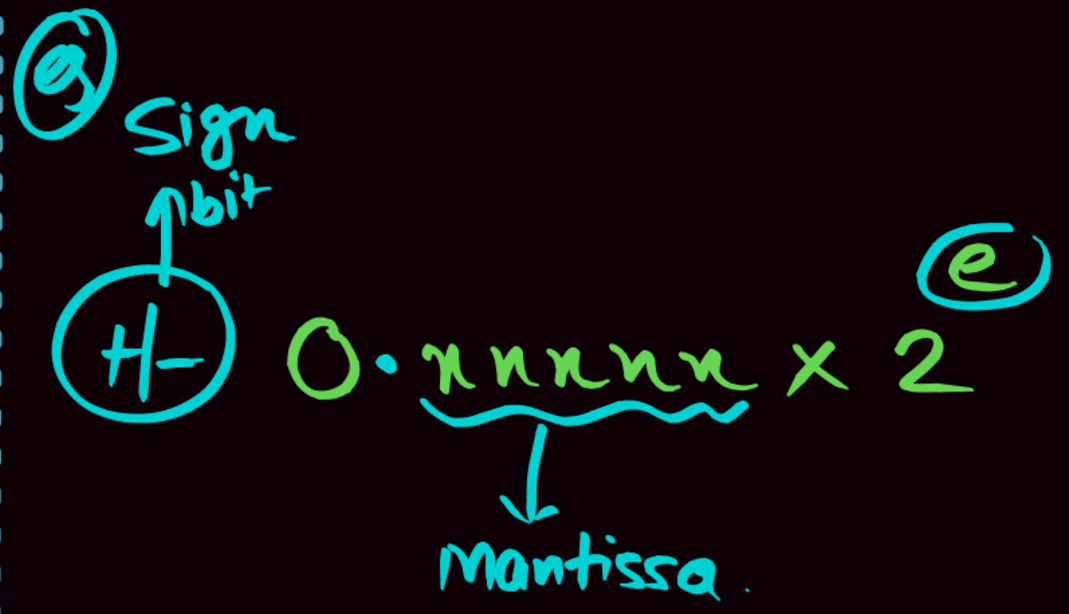


S[sign] → 0 [+ve]

S[sign] → 1 [-ve]

E/BE : Exponent / Biased Exponent

M : Mantissa.



e : exponent

$E = e + bias$

OR

$BE = AE + bias$

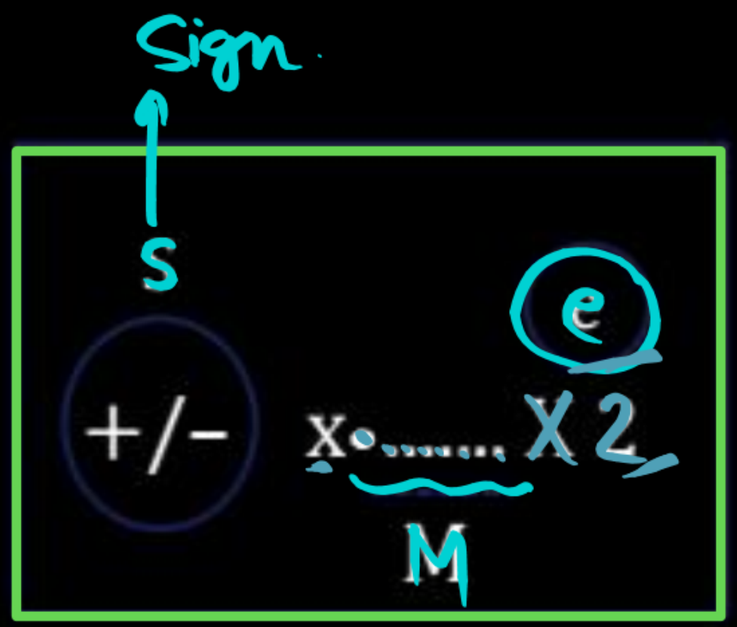
Floating-Point Representation

S	E	M

S: sign bit → 0 +ve, 1 –ve

E: exponent

M: Mantissa

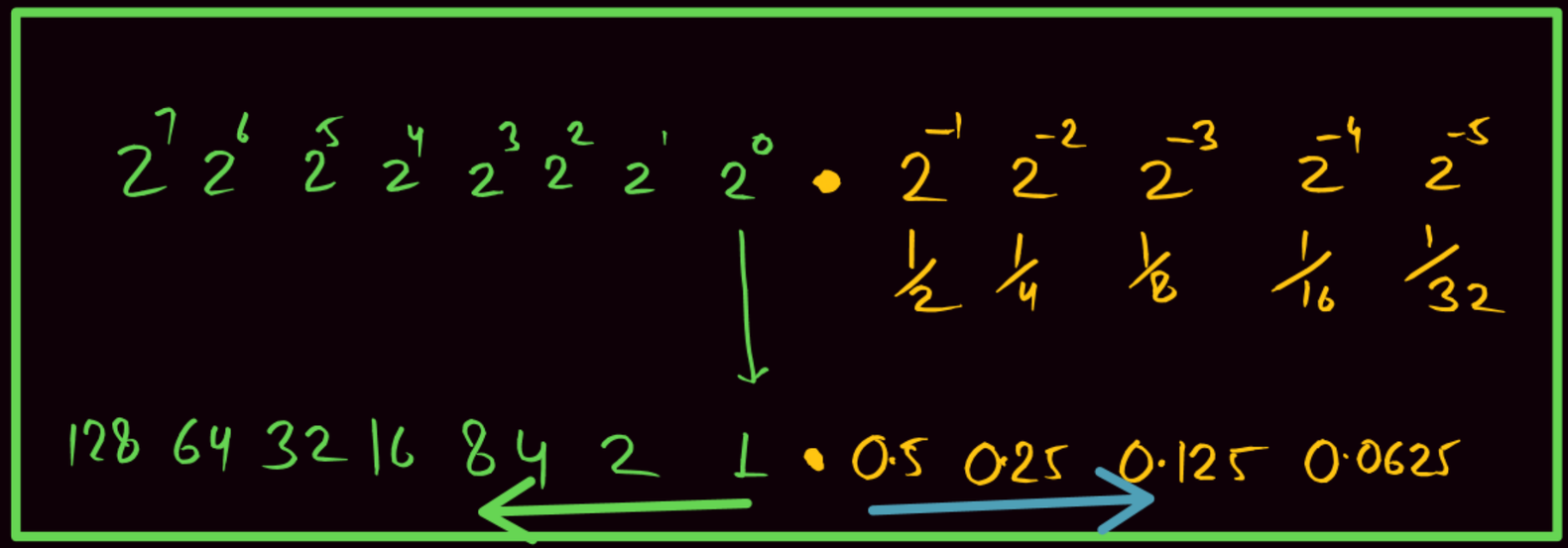


$\pm$ ..... $X2^e$

6.5 in Binary $\Rightarrow$ 110.1

Q.1 6.5

110.1



$\pm x.\underbrace{\text{.....}}_{\text{Mantissa}} \times 2^{e}$

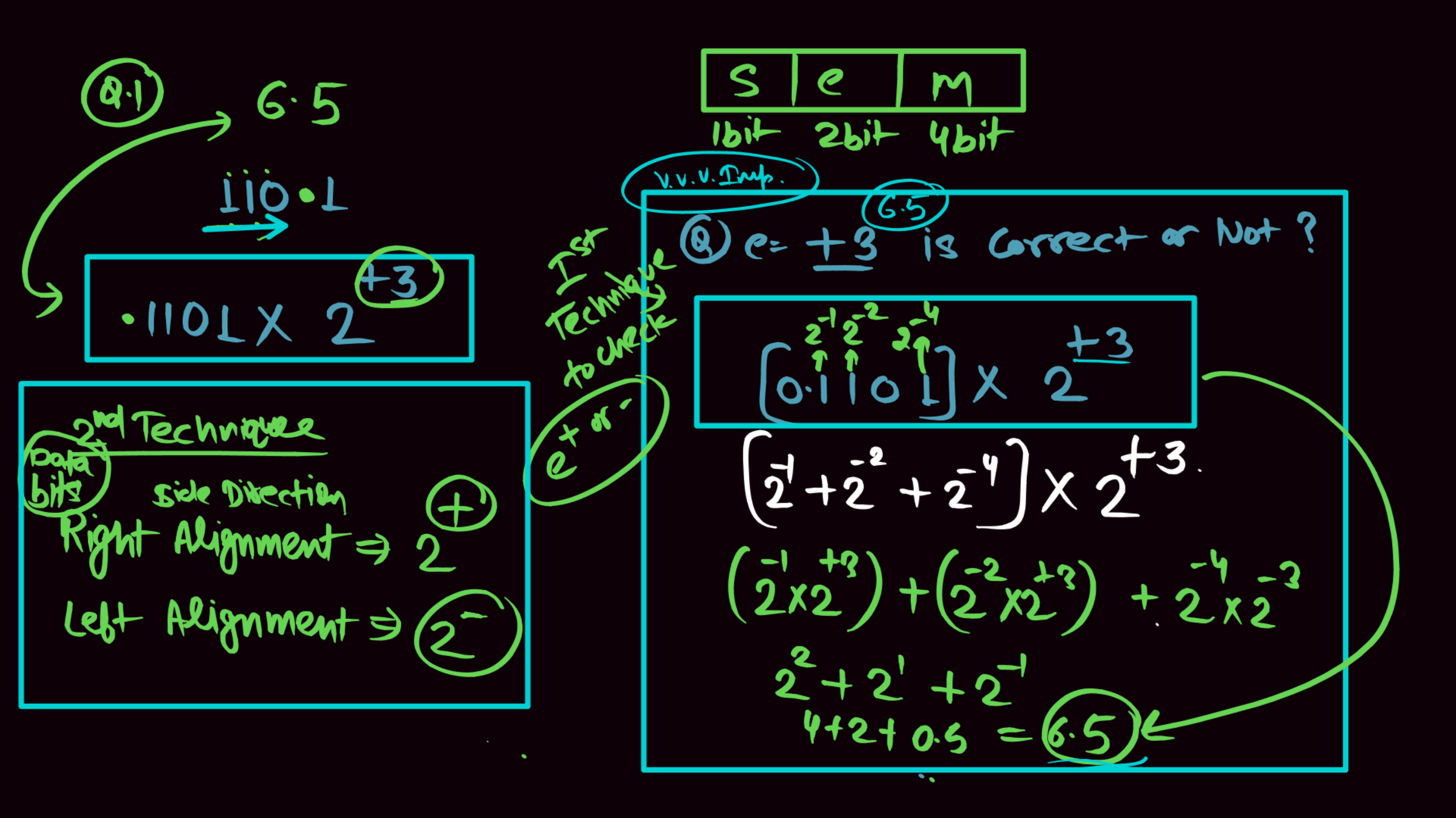
(6)

- 4bit 0110
- 5bit 00110
- 6bit 000110

] X

6

110


Q.1
6.5
110.1
.1101 X 2 +3
2nd Technique
Data bits
Side Direction
Right Alignment ⇒ 2 (+)
Left Alignment ⇒ 2 (-)
e + or -
1st Technique to check
S | e | m
1bit 2bit 4bit
V.V.V.Imp.
Q e = ±3 is Correct or Not ?
6.5
[0.1101] X 2 ±3
2^-1 2^-2 2^-4
[2^-1 + 2^-2 + 2^-4] X 2 +3
(2^-1 X 2^+3) + (2^-2 X 2^+3) + 2^-4 X 2^-3
2^2 + 2^1 + 2^-1
4+2+0.5 = 6.5

Q.1 → 6.5

110.1

$+.1101 \times 2^{+3}$

S = 0 (+ve)

e = 3 [11]

M = 1101

S	e (2bit)	m (4bit)
0	11	1101

S	e	m
1bit	2bit	4bit

V.V.V. Imp.

6.5

Q) e = ±3 is Correct or Not ?

$[0.1101] \times 2^{\pm 3}$ (2^{-1} 2^{-2} 2^{-4})

$(2^{-1} + 2^{-2} + 2^{-4}) \times 2^{+3}$

$(2^{-1} \times 2^{+3}) + (2^{-2} \times 2^{+3}) + 2^{-4} \times 2^{-3}$

$2^2 + 2^1 + 2^{-1}$

$4 + 2 + 0.5 = 6.5$

$2^4\ 2^3\ 2^2\ 2^1\ 2^0 \cdot 2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}$

$0.1101 \times 2^{+3} \Rightarrow 6.5$

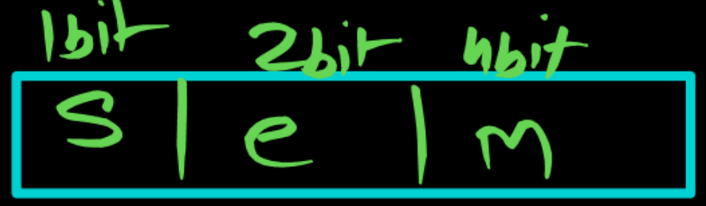
$\left[2^{-1} + 2^{-2} + 2^{-4}\right] \times 2^{+3}$

$2^{+2} + 2^{+1} + 2^{-1}$

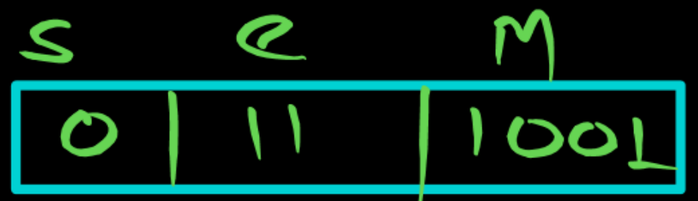
$4 + 2 + \frac{1}{2}$

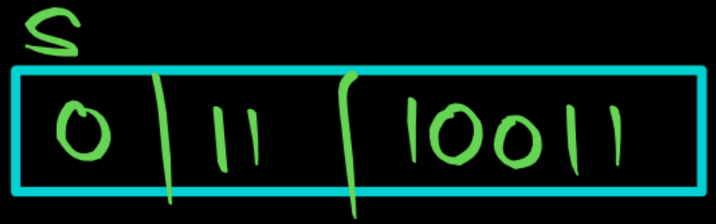
$\boxed{6.5}$

Q.2 +(4.5)



Q.3 +4.75





Q. 1 $+6.5$

$6.5 = (110.1)_2$

$\underset{S}{0}.\underset{M}{1101} \times \underset{2^e}{2^3}$

$S = 0\ (+)$

$M = 1101$

$e = 3 = (11)_2$

S(1bit)	e(2bit)	M(4bit)
0	11	1101

Very. Imp

$6.5 = 110.1$

$= .1101 \times 2^3$

$= [.2^{-1} + 2^{-2} + 2^{-4}] \times 2^3$

$= [2^2 + 2^1 + 2^{-1}]$

$= 6.5$

Right Alignment = 2^{+ve} ⇒ +ve

Left Alignment = 2^{-ve} ⇒ −ve

e

Q. 2 $+4.5$

100.1

0.1001×2^3

S =0 (+ve)

M = 1001

e = 3 [11]

S	e	M
0	11	1101

Q. 3

$+4.75$

100.11

$.10011 \times 2^3$

$S = 0$

M: 10011

$e = 3 \Rightarrow (11)_2$

S	e	M
0	11	10011

NOTE:

Mantissa alignment process is used to adjust the decimal point; in this process right alignment increments the exponent and left alignment decrements the exponent.

2^{+shift} power(+) = Right alignment ⇒ Increment the exponent

2^{-shift} power (-) = Left alignment ⇒ Decrease the exponent

$1.01 \times 2^{-8} \times (2^{+5})$

1.01×2^{-3}

Right Alignment

6.5

110.1

$\Rightarrow .1101 \times 2^{3}$ (2^{+3})

$\Rightarrow [.2^{-1} + 2^{-2} + 2^{-4}] \times 2^{3}$

$\Rightarrow 2^{2} + 2^{1} + 2^{-1}$

$\Rightarrow 4 + 2 + 0.5$

$\Rightarrow 6.5$ Ans

Left Alignment

Data: $0.0000000101 \times 2^{+5}$

$[1.01 \times 2^{+5-8}]$

$+1.01 \times 2^{-3}$

(Align to use upto 8 times)

Q) $+0.00101$

Left Alignment

$+0.101 \times 2^{-2}$

1bit	4bit	5 bit
S	e	m

S = 0 [+ve]

M = 101

e = -2

e = -ve → No Provision to Represent [tells] exponent is Negitive.

Here exponent is Negitive but Sign bit [0] Bcz Number is +ve.

So How to Deal this Negitive exponent.

Q.1 Why Biasing is Required ?

Q.2 How bias value Decide/Select ?

Q) $+0.00101$

Left Alignment

$+0.101 \times 2^{-2}$

1bit	4bit	5bit
S	e	m

1bit	4bit	M (5bit)
0	1110	10100

Ans

$S = 0$ [+ve]

$M = 101$

$e = -2$

Here e is -ve

So taking 2's Complement

$0010 \Rightarrow$ 1's $1101 \Rightarrow$ 2's 1110

Need of E ?

$E = e + bias$

WHy biasing Required ?

Note if e = -ve then taking 2's complement.

$$2\text{'s Complement} = -\left(2^{n-1}\right) \text{ to } +\left(2^{n-1}-1\right)$$

$$4\text{bit } 2\text{'s Complement} = -2^{4-1} \text{ to } +2^{4-1}-1$$

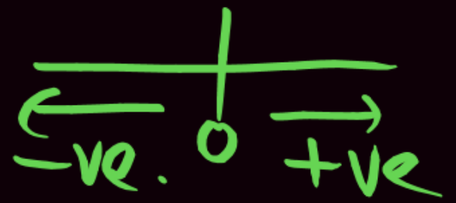
5 bit then

$$-2^4 \text{ to } +2^4-1$$

$$\Rightarrow (-16 \text{ to } +15)$$

$$\Rightarrow -2^3 \text{ to } +2^3-1$$

$$= \boxed{-8 \text{ to } +7}$$



if we take 2's Complement = 5 bit ⇒ −16 to +15

So Solution

Note

Convert the Number into '0' (or) Any +ve Number.

So biasing Will be Used.

Q. 4 $+0.00101$

0.101×2^{-2}

$M = 101$

$E = -2$

$S = 0$

$E = -2 = (1110)_2$ 2's complement

So taking 2's complement

S	E(4bit)	M(5 bit)
0	1110	10100
	E	M

E=14

e=-2 → 1110

e=14 ⇒ 1110

Biasing: is method in which we convert the negative number into the positive number [exponent].

Biasing = to Convert -ve Number (Any Number) into 0 (or) Any +ve Number.

Q2 How bias is select ?

$bias = 2^{k-1}$ $E = k$ bit

Sol' If exponent is k bit then 2's Complem = -2^{k-1} to $+2^{k-1}-1$

eg 4 bit Range = -2^{4-1} to $+2^{4-1}-1 \Rightarrow$ -8 to $+7$

Note In order to Convert ALL Number into +ve (Positive Number) then take the Most (Highest) Negitive Number & Add as bias.

(2^{k-1})

$\therefore$

if exponent k bit

eg $e = 4$ bit $\Rightarrow -2^{4-1}$ to $+2^{4-1} = -8$ to $+7$

$$bias = -8$$

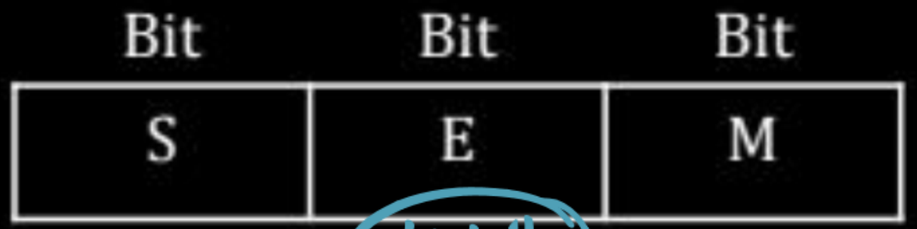
So $E = e + bias$

$BE = AE + bias$

$-8 + 8 = 0$
$-7 = 1$
$-6 = 2$
$-5 = 3$
$-4 = 4$
$-3 = 5$
$-2 = 6$
$-1 = 7$
-0
$+1$

$+7 + 8 = 15$

4 bits



S = Sign

E/BE = Exponent or

BE = bias exponent

M = Mantissa

E = e + bias

Bias = 2^{K-1} where K is exponent bits

Example

If K = 4 bits

Exponent = 4 bit then

bias = $2^{K-1} = 2^{4-1} = 8$

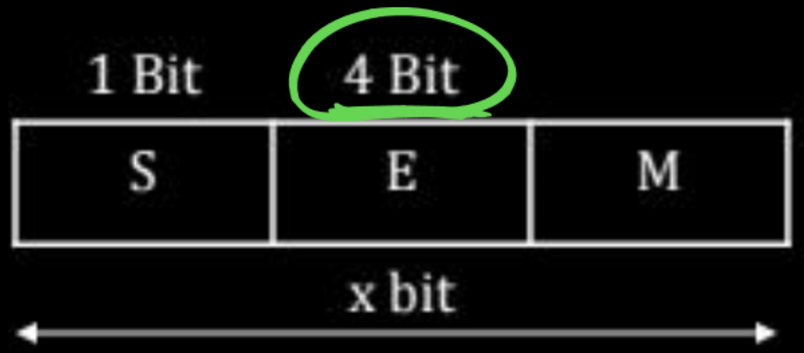
$E = e + bias$

OR

$BE = AE + bias$.

$Bias = 2^{k-1}$

k : #bits in exponent



Bias = $2^{K-1} = 2^{4-1}$

bias = 8

E = e + bias

E = e + 8

E = 4 bit

or

Excess 8 code

$2^{K-1} = 8$

$2^{K-1} = 23$

K – 1 = 3

K = 4

E = 4 bit

$2^{k-1} = 8$

$2^{k-1} = 2^3$

$k-1 = 3$

$k = 4$

if k = 4
then bias
$2^{4-1} = 8$

e [original exponent]	Stored exponent [BE] E
-8	0
-7	1
-6	2
-5	3
-4	4
-3	5
-2	6
-1	7
0	8
1	9
2	10
3	11
4	12
5	13
6	14
7	15

e = +3

+8 +8 +8 +8 +8 +8 +8

Q. From previous question

0.00101

0.101×2^{-2}

M = 101

Bias = 2^{5-1}

Bias = 16

e = −2

$E = e + bias$

$e = E - bias$

1 bit	5 bit	4 bit
S	E	M

E = 5 bit

$bias = 2^{5-1} = 16$

$E = -2 + 16$

$E = 14$

$E = e + bias$

$E = -2 + 16$

$E = 14$

$E = (01110)_2$

BE = AE + bias

−2 + 16

= 14 → 5 bits

1 bit	5 bit	4 bit
0	01110	1010

E = 14, bias = 16. M. Ans

Formula: $(-1)^S \times 0.M \times 2^e$

$(-1)^0 \times 0.101 \times 2^{E-bias}$

$0.101 \times 2^{14-16} = 0.101 \times 2^{-2}$

0.00101 Ans

$$E = e + bias$$

$$e = E - bias$$

Normalized Mantissa

Explicit Normalized

Syntax

$0.1\underline{\text{.....}} \times 2^{e}$
M

Formula to get number

[value formula]

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

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

Implicit Normalized

Syntax

$1.\underline{\text{.....}} \times 2^{e}$
M

Formula to get number

[value formula]

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

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

THANK YOU!