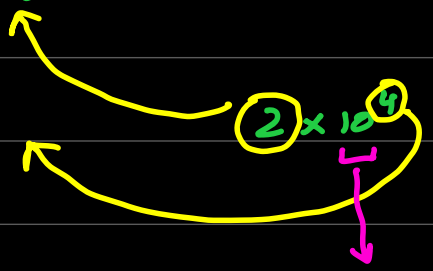
Data Representation

Representation of real numbers

↳ decimal numbers e.g: 14.525 , -14.625

Mantissa : The significant digit of FPN (Floating Point Number)

Exponent : Power



→ Since studying binary, this is of no use.

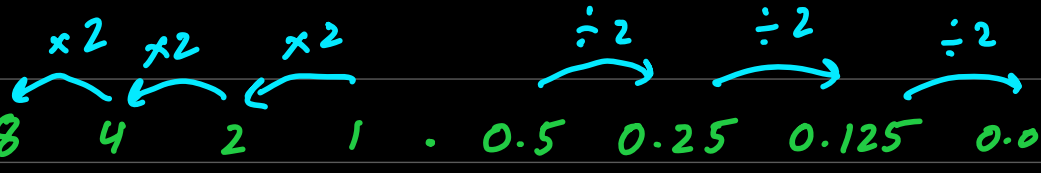
0.11110000×2^{4} → Exponent

Mantissa

↳ Since binary, base-2

Note : 128 64 32 16 8 4 2 1 . $\frac{1}{2^1}$ $\frac{1}{2^2}$ $\frac{1}{2^3}$ $\frac{1}{2^4}$ ...

OR



CASE 1 : Decimal Number to Binary Conversion

- There will be certain things that would be given in the question

- No of bits used for Mantissa — 12 bits
- No of bits used for exponent — 4 bits

↓ below question

Q- Represent +14.75 in floating-point Representation.

Step 1: Write down memorized list

128 64 32 16 8 4 2 1 . 0.5 0.25 0.125 0.0625 0.03125

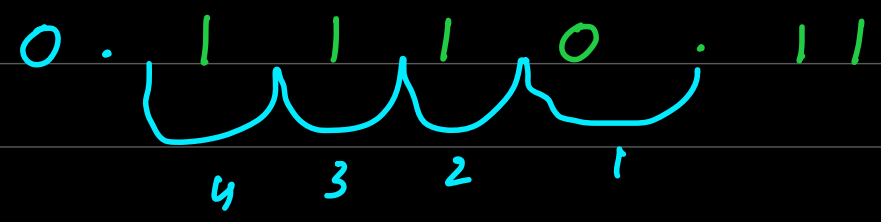
Step 2: Put "1" on numbers which will be added to fulfill question's requirement

★ Include zeroes in the middle

128 64 32 16 8 4 2 1 . 0.5 0.25 0.125 0.0625 0.03125

1 1 1 0 . 1 1 → Fixed Point Representation

Step 3 : Move the decimal place all the way to the left and write the exponent value.



⟶ $0.1110\,11 \times 2^{4}$

Step 4 : Ignore decimal and convert the exponent into binary form

4 => 8421
0100

0111011 0100

- To reduce memory consumption b/c "2" and "." is repeating

Step 5: Store in given spaces

- Always store mantissa from left side

· Always store exponent from right side (overflow, underflow)

CASE 2: Negative Decimal number to Binary Floating Point

Q- Convert -14.75 to Binary Floating Point representation

Step 1: Carry out steps 1 till 4 in CASE 1, ignoring negative sign

Step 2: Exponent would remain same and apply two's compliment method on mantissa only and store it

0111011 — Mantissa

0100 — Exponent

0111011

1000100 — One's compliment

1 - Binary Addition

1000101

Two's compliment

Binary Addition

	Sum	Carry
0 0	0	0
0 1	1	0
1 0	1	0
1 1	0	1
1 1 1	1	1

Step 3: Store the new mantissa and the same exponent using the rules.

Mantissa Exponent

- Apply one's compliment, then two's compliment

CASE 3 : Binary to Denary Conversion

Mantissa Exponent

Step 1 : Seperate Mantissa and exponent

011101100000 0100

Step 2 : Convert Exponent into denary format

8 4 2 1
0 1 0 0 Exponent = 4

Step 3: Move decimal places number of times which equal the exponent

0 1110 11000000

1 2 3 4

step 4 : Apply memorized list

8 4 2 1 . 0.5 0.25

1 1 1 0 . 11

8 + 4 + 2 . 0.5 + 0.25

+ 14 . 75 Ans

- zeroes at the start and end can be removed since they represent no values

CASE 4 : Negative Binary to Denary Conversion

- Identify +ve or -ve number by looking the left most bit

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

Step 1 : Seperate Mantissa, exponent and apply two's complement on Mantissa

1 000 101 100

0 111 010

1

0 111 011

Step 2: Place decimal, apply memorized list and at the end, add -ve sign.

- Exponent = 4 2 1 / 100 = 4

- 0 1 1 1 0 11

1 2 3 4

8 4 2 1 0.5 0.25

- 1110.11

8+4+2 . 0.5 + 0.25 → 14.75 → -14.75 Ans

Normalization

- It is a technique that is used to make your data more accurate/precise

0.1 ⟶ Represents positive number

1.0 ⟶ Represents negative number

Q- How will we know/figure out that a binary representation is normalized?

- First and second bit should never be the same ⟶ Tou normalized hai

Q- How to normalize?

Q- What problems could occur if the binary representation is not normalized?

- Multiple representation of a single number.
- Precision lost
- Redundant leading zeroes in mantissa.

Q. What are the problems in floating Point Representation?

- 0.2 / 0.1 / 0.4 → These numbers can not be exactly represented And the solution for this problem is rounding which would cause rounding error.

- 0.2 has been represented by value greater than 0.2
- 0.4 has been represented by value greater than 0.4

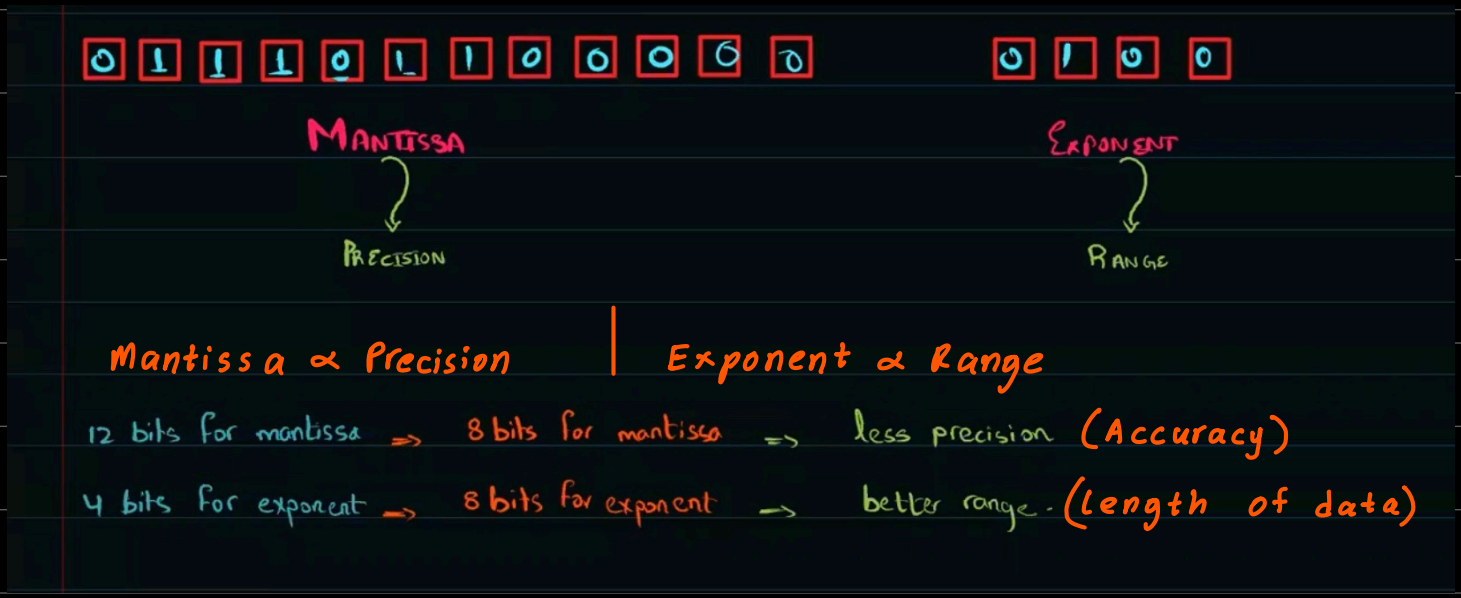
- So, after calculation with the rounded numbers the difference would increase
- The diffirence will be significant. (Compound Error)

Q- Why rounding error occurs? → Sol. increase mantissa, normalize

· Because there is no exact representation of some binary numbers.

Trade off b/w Mantissa and Exponent

· Trade off means relationship



Q- What is the trade off b/w Mantissa and exponent?

· The trade off is b/w precision and range

· If more bits are used for Mantissa that means better precision

· If more bits are used for exponent that means better range.

· More number of bits for mantissa mean less number of bits for exponent.

Largest Positive Number & Smallest Positive Number

Overflow: The result of carrying out a calculation which produces a value too large that can be stored.

Underflow: The result of carrying out a calculation which produces a value too small that can be represented.