FLOATING POINT NUMBER REPRESENTATION:

* Floating Point - A number without a fixed number of digits before and after the decimal point.

* A floating point number usually has a decimal point. Eg. 0, 3.14, 6.5 and 125.5 are floating point numbers.

* Floating point numbers are represented in scientific notation. The binary point floats to the right of the most significant 1 and an exponent is used.

$$\pm M \times B^{E}$$

* It has three parts - Mantissa, Base and Exponent

eg.

Number	Mantissa	Base	Exponent
9×10^{8}	9	10	8
110×2^{7}	110	2	7
4364.784	4364784	10	-3

IEEE 754 Floating point number representation

* A binary floating point number can be represented by
  - Single precision
  - Double Precision

* Each representation has
  - A sign for the number
  - Some significant bits
  - A Signed scale factor exponent for an implied of base 2.

Single Precision

32 bits

S	E'	M
Sign of number 0 - +ve 1 - -ve	8 bit Signed exponent in excess-127 representation	23 bits Mantissa fraction.

Value Represented $= \pm 1.M \times 2^{E'-127}$

* The basic IEEE format is 32-bit representation. The leftmost bit is the sign S for the number.

* The next 8 bits, represent signed exponent (implied base 2) and remaining 23 bits, M, the fractional part.

* Therefore, the mantissa

$$B = 1.M = 1.b_{-1}b_{-2} \dots b_{-23} \text{ has the value}$$

$$V(B) = 1 + b_{-1} \times 2^{-1} + b_{-2} \times 2^{-2} + \dots + b_{-23} \cdot 2^{-23}$$

* By convention, when binary point is placed to the right of first significant bit, the number is said to be normalized.

* Base 2 of scale factor and leading 1 of mantissa are fixed. Hence they do not appear explicitly in representation.

* Instead of actual signed exponent, E, the value stored in exponent is an unsigned integer $E' = E + 127$. This is called excess-127 format. Thus E' is in range $0 \leq E' \leq 255$ (ie) the actual exponent E is in range $-126 \leq E \leq 127$.

Double Precision:

← 64 bits →

S	E'	M
Sign	11-bit excess-1023 exponent	52-bit mantissa

Value represented $= \pm 1.M \times 2^{E' - 1023}$

* To provide more precision and range for floating point numbers, the IEEE standard specifies a double-precision format. It has increased exponent and mantissa ranges.

* The 11-bit excess-1023 exponent E' has range $-1022 \leq E \leq 1023$

* A computer must provide atleast single-precision representation to conform to the IEEE standard. Double-precision representation is optional

Example 1: Represent $(1259.125)_{10}$ in Single and Double Precision format

Step 1: Convert decimal to binary.

$(1259)_{10} \Rightarrow (10011101011)_2$

$(0.125)_{10} \Rightarrow (.001)_2$

$$
\begin{array}{l}
2\,|\,1259 \\
2\,|\,629 - 1 \\
2\,|\,314 - 1 \\
2\,|\,157 - 0 \\
2\,|\,78 - 1 \\
2\,|\,39 - 0 \\
2\,|\,19 - 1 \\
2\,|\,9 - 1 \\
2\,|\,4 - 1 \\
2\,|\,2 - 0 \\
\quad 1 - 0
\end{array}
$$

$0.125 \times 2 \Rightarrow 0.250 \Rightarrow 0$

$0.25 \times 2 \Rightarrow 0.5 \Rightarrow 0$

$0.5 \times 2 \Rightarrow 1.0 \Rightarrow 1$

$(1259.125)_{10} = (10011101011.001)_2$

Step 2: Normalize the number

Single Precision $= \pm 1.M \times 2^{E'-127}$

Double precision $= \pm 1.M \times 2^{E'-1023}$

$(10011101011.001)_2 \Rightarrow 1.0011101011001 \times 2^{10}$

Step 3: Single Precision format

$\pm 1.M \times 2^{E'-127} = 1.0011101011001 \times 2^{10}$

$E' - 127 = 10 \Rightarrow E' = 137.$

$E' = (137)_{10} \Rightarrow (10001001)_2$

$$
\begin{array}{l}
2\,|\,137 \\
2\,|\,68 - 1 \\
2\,|\,34 - 0 \\
2\,|\,17 - 0 \\
2\,|\,8 - 1 \\
2\,|\,4 - 0 \\
2\,|\,2 - 0 \\
\quad 1 - 0
\end{array}
$$

0	10001001	0011101011001 ....
1bit	8 bits	23 bits

Step 4: Double Precision format

$\pm 1.M \times 2^{E'-1023} = 1.0011101011001 \times 2^{10}$

$E' - 1023 = 10 \Rightarrow E' = 1033 \Rightarrow (10000001001)_2$

$$
\begin{array}{l}
2\,|\,1033 \\
2\,|\,516 - 1 \\
2\,|\,258 - 0 \\
2\,|\,129 - 0 \\
2\,|\,64 - 1 \\
2\,|\,32 - 0 \\
2\,|\,16 - 0 \\
2\,|\,8 - 0 \\
2\,|\,4 - 0 \\
2\,|\,2 - 0 \\
\quad 1 - 0
\end{array}
$$

0	10000001001	0011101011001 ...
1	11 bits	52 bits

exponent

1 bit	9 bits	6 bits
0	110100000	000011
Sign	Mantissa	exponent

Exponent = $(000011)_2 = (3)_{10}$

$\Rightarrow 0110.100000 \times 2^3$

Convert it into decimal.

$0\times2^3 + \cancel{0}1\times2^2 + 1\times2^1 + 0\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 0\times2^{-3} \ldots$

$4 + 2 + 0.5 = (6.5)_{10}$.

(ii) 0101010000000010 – 9 bits mantissa & 6 bits exponent

Sign	Mantissa	exponent
0	101010000	000010
1 bit	9 bits	6 bits

Exponent = $(000010)_2 = (2)_{10}$

$\Rightarrow 010.1010000 \times 2^2$

Convert it to decimal.

$0\times2^2 + 1\times2^1 + 0\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} \ldots$

$2 + 0.5 + 0.125 = (2.625)_{10}$

FLOATING POINT OPERATIONS:

ADDITION OF FLOATING POINT NUMBERS:

Steps

~~1. Choose the number with the smaller exponent~~

1. Compare the exponent of both the operands.
2. (i) If it is equal add the two operands (mantissa). If it is not equal then increase the smaller exponent
   (ii) Shift the smaller number to the right until its exponent would match the larger exponent

2021-11-11 08:43

3. Set the exponent of the result equal to the larger exponent.
4. Perform addition on the mantissa and determine sign of result.
5. Normalize the result, if necessary

Eg: 9.75 + 18.5625

$(9.75)_{10} = (?\ )_2$

$0.75 \times 2 = 1.5 \Rightarrow 1$

$0.5 \times 2 = 1.0 \Rightarrow 1$

$2\,|\,9$
$2\,|\,4 - 1$
$2\,|\,2 - 0$
$1 - 0$

$(9)_{10} = (1001)_2$

$(0.75)_2 = (0.11)_2$

$(9.75)_{10} = (01001.11)_2 \Rightarrow 1.00111 \times 2^3$.

$(18.5625)_{10} = (?\ )_2$

$0.5625 \times 2 = 1.125 \Rightarrow 1$

$0.125 \times 2 = 0.25 \Rightarrow 0$

$0.25 \times 2 = 0.5 \Rightarrow 0$

$0.5 \times 2 = 1.0 \Rightarrow 1$

$2\,|\,18$
$2\,|\,9 - 0$
$2\,|\,4 - 1$
$2\,|\,2 - 0$
$1 - 0$

$(18)_{10} = (10010)_2$

$(0.5625)_{10} = (.1001)_2$

$(18.5625)_{10} = (010010.1001)_2 \Rightarrow 1.00101001 \times 2^4$

$(9.75)_{10} = 1.00111 \times 2^3 \Rightarrow 0.100111 \times 2^4$

Add the Mantissa:

$$
\begin{array}{r}
0.10011100 \\
1.00101001 \\
\hline
1.11000101
\end{array}
$$

$\boxed{\text{Result: } 1.11000101 \times 2^4.}$

SUBTRACTION: Same procedure as addition.

Eg: 9.75 − 0.5625.

$(9.75)_{10} = (01001.11)_2 \Rightarrow 1.00111 \times 2^3$

$(0.5625)_{10} = (0.1001)_2 \Rightarrow 1.001 \times 2^{-1} \Rightarrow 1.001 \times 2^{-1} \times 2^4$

$\Rightarrow 0.0001001 \times 2^3$

Subtraction:

$$
\begin{array}{r}
1.0011\overset{0}{1}\overset{1}{10}\ 10 \\
0.0001001 \\
\hline
1.0010011
\end{array}
$$

$\boxed{\text{Result: } 1.0010011 \times 2^3}$

2021-11-11 08:43

MULTIPLICATION OF FLOATING POINT NUMBER:

* Multiplication of a pair of floating point number $X = m_x \times 2^a$ and $Y = m_y \times 2^b$ is represented as

$$X * Y = (m_x * m_y) \cdot 2^{a+b}$$

Algorithm:

1. Compute the exponent of the product by adding the exponent of operands.
2. Multiply the mantissas
3. Normalize and round the final product.

Example: Multiply $X = 1.000 \times 2^{-2}$ and $Y = -1.010 \times 2^{-1}$

1. Add the exponents: $(-2) + (-1) = -3$
2. Multiply the mantissas

```
      1.000 0 X 1.010
     ----------------
        0 0 0 0
      1 0 0 0 -
    0 0 0 0 - -
  1 0 0 0 - - -
  ---------------
  1.0 1 0 0 0 0 0
```

Hence, $1.000 \times -1.010 = -1.010000$

3. After rounding the product is -1.0100×2^{-3}

DIVISION OF FLOATING POINT NUMBER

* Division of pair of floating point numbers $X = m_x \times 2^a$ and $Y = m_y \times 2^b$ is represented as $X / Y = (m_x / m_y) \times 2^{a-b}$

Algorithm:

1. Compute the exponent of the result by Subtracting the exponent
2. Divide the mantissa and decide sign.
3. Normalize and round the result.

eg. Divide $X = 1.0000 \times 2^{-2}$ and $Y = -1.0100 \times 2^{-1}$

1. Sub. exponent: $(-2) - (-1) = -1$
2. Divide mantissa: $(1.0000) \div (-1.0100) \Rightarrow -0.1100$
3. Result is -0.1100×2^{-1}

2021-11-11 08:44