FLOATING POINT NUMBERS :-

In a fixed point arithmetic, only small numbers can be handled. For example, in a 6-bit number, the maximum numbers represented are

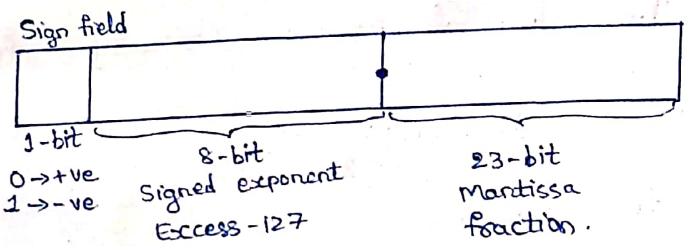
011111 → (+ ve)
111111 → (- ve)

If the value of the number is to be increased, then the no of bits is to be increased. This is impossible in computer because it is expensive.
To represent very large nos like Avogadro's number (6.024×10^{23}) or very small no like Planck's constant (6.625×10^{-34}), the floating point representation becomes essential.
The general format of floating point no is

$$\pm 1.M \times 2^{E}$$

where M is the mantissa which must be a fraction
E is the exponent can be +ve or -ve integer.

Representation of 32 bit number in floating point representation (IEEE standard) :-



The binary point is always assumed at the left most position of mantissa.
The 8-bit exponent part should represent both negative as well as positive exponent.
The highest no that can be represented by 8-bit is 256. This is divided into 2 portions. One part is used to represent positive exponent, & the other to represent negative exponent.
The last 23 bits represents mantissa. Since, binary

normalization is used, the most significant bit of the mantissa is always '1'. This bit is not represented & it is assumed to be to the immediate left of the binary point.

Hence, the 23 bits stored in the Mantissa-field actually represents the fractional part of the mantissa, i.e., the bits which are present to the right of binary point.

The range of E' $(E' = E+127)$ is $0 \le E' \le 255$.

The end values of this range, 0 & 255 are used to represent special values. Therefore, the range of E' for normal values is

$$1 \le E' \le 254.$$

This means that the actual exponent (E) is in the range

$$-126 \le E \le 127.$$

The 32-bit standard representation is called <u>Single Precision Representation</u> because, it occupies a single 32-bit word. The Scale-Factor has a range of 2^{-126} to 2^{+127}.

i) Represent $(+1259.125)_{10}$ in a single Precision format.

Converting the decimal no in binary format

$$\equiv +(01.00\,1110.1011.001)_2$$

Normalizing the number

$$\equiv +1.0011101011001 \times 2^{10}$$

From the given number

$S = 0$; $E = 10$; $E' = 137_{(10)}$; $M = 0011101011001$

$\therefore E' = 89_H = 10001001$

0	10001001	00111010110010 0.... 0

$\equiv 449D6400_{(H)}$

$16 \,|\, 1259$
$16 \,|\, 78 - B$
$4 - E$
$4EB_{(H)}$

0.125×2
$0 \leftarrow 0.250 \times 2$
$0 \leftarrow 0.5 \times 2$
$1 \leftarrow 1.0$

$16 \,|\, 137$
$8 - 9$

$(-307.1875)_{10}$

$\equiv -0001\,0011\,0011.0011_{(2)}$

$\equiv -1.00110011\,0011 \times 2^{8}$

$S = 1$; $E = 8_{(10)} = 1000_{(2)}$; $E' = 8 + 127 = 135_{(10)}$

 $= 87_{(H)} = 1000\,0111$

$M = 0011\,0011\,0011$

1	10000111	0011001100110.....0

$\equiv C3999800_{(H)}$

$16 \,|\, 307$
$16 \,|\, 19 - 3$
$1 - 3$

$133_{(H)}$

0.1875×2
$0 \leftarrow 0.375 \times 2$
$0 \leftarrow 0.75 \times 2$
$1 \leftarrow 1.5 \times 2$
$1 \leftarrow 1$

$.0011_{(2)}$

$16 \,|\, 135$
$8 - 7$

$(0.0625)_{10}$

$\equiv 0.0001_{(2)}$

$\equiv 1.000.... \times 2^{-4}$

$\therefore S = 0$; $E = -4$; $E' = 123 = 7B_{(H)} = 01111011$

$M = 00000$

0	01111011	0000 0

$\equiv 3D800000_{(H)}$

0.0625×2
$0 \leftarrow 0.125 \times 2$
$0 \leftarrow 0.25 \times 2$
$0 \leftarrow 0.5 \times 2$
$1 \leftarrow 1.0$

$16 \,|\, 123$
$7 - B$

$(+1.725)_{10}$

$\doteq 1.1011100110011001100 \times 2^{0}$

$S = 0$; $E = 0$; $E' = 127 = 7F = 01111111$

$M = 1011100110011001100$

0	01111111	1011100110011001100 1100

$= 3FDCCCCC_{(H)}$

$16 \,|\, 127$
$7 - F$

0.725×2
$1 \leftarrow 1.450 \times 2$
$0 \leftarrow 0.9 \times 2$
$1 \leftarrow 1.8 \times 2$
$1 \leftarrow 1.6 \times 2$
$1 \leftarrow 1.2 \times 2$
$0 \leftarrow 0.4 \times 2$
$0 \leftarrow 0.8 \times 2$
$1 \leftarrow 1.6 \times 2$
$1 \leftarrow 1.2 \times 2$
$0 \leftarrow 0.4 \times 2$
$0 \leftarrow 0.8$

* To provide more precision & range for floating point numbers, the IEEE standard also specifies a DOUBLE PRECISION Format. The Double Precision format has an increased exponent & mantissa range.

←— 64 bit —→

S	E'	M
0 → +ve 1 → -ve	11-bit exponent Excess 1023	52-bit Mantissa

$2^{11} \rightarrow 2048$
0 to 2047

The range of E' is $0 \le E' \le 2047$. The end values of this range, 0 & 2047 are used to represent special values. Therefore, the range of E' for normal values is $\underline{1 \le E' \le 2046}$. This means that the actual exponent $(E = E' - 1023)$ is in the range $\underline{-1022 \le E \le 1023}$.

i) Represent +1259.125 in double precision format

$\equiv +(010011101011.001)_2$

$\equiv +1.0011101011001 \times 2^{10}$

$\therefore S = 0$; $M = 00111010110010 0 \ldots\ldots 0$

$E = 10 \Rightarrow E' = 10 + 1023 = 1033_{(10)}$

$= 409_{(H)} = 10000001001_{(2)}$

16 | 1033
16 | 64 - 9
4 - 0

0	100 0000 1001	0011 1010 1100 1000 0 . . . 0

$\equiv 4093AC8000000000_{(H)}$

ii) +45 in single precision & double precision format

$\equiv 00101101 \times 2^0 \equiv 1.01101 \times 2^5$

16 | 45
2 - D
= 00101101

$\therefore S = 0$; $M = 01101$; $E = 5$

in single precision ; $E' = E + 127 = 132 = 84_{(H)}$
$= 10000100$

16 | 132
8 - 4

in double precision ; $E' = E + 1023 = 1028 = 404_{(H)}$
$= 10000000100$

16 | 1028
16 | 64 - 4
4 - 0

i)	0	10000100	011010 0	≡ 42340000
ii)	0	10000000100	011010 0	≡ 4046800000000000

(26)

The Hexadecimal value of π is 3.243F6A8885A308D3.......
work out the IEEE standard representation π in Single & double precision format.

π ≡ 0011.001001000011111101101010100010001000010110100011
0000,10001101
0011
...

≡ 1.1001001000011111101101010100010001000010110100011 0000,1000 11010011...
× 2^1

a) Single precision format

S = 0 ; E = 1 ; M = 10010010000111111011010

E' = 1+127 = 128 = $80_{(H)}$ = 1000 0000

0	10000000	10010010000111111011010

≡ 40490FDA

b) Double precision format :

S = 0 ; E = 1 ; M = 1001001000011111101101010100010001000010110100011
000

E' = 1024 = $400_{(H)}$

0	10000000000	1001001000011111101101010100010001000010110100011000

≡ 400921FB54442D18

-25.125 } S.P.F & D.P.F
-0.08125

≡ 00011001.001 ≡ 1.1001001 × 2^4 ; E' = 131 = $83_{(H)}$
E' = 1027 = $403_{(H)}$

1	10000011	100100100..........0	≡ C1C90000
1	10000000011	100100100..........0	≡ C039200000000000

19

0.125 × 2
0 ← 0.25 × 2
0 ← 0.5 × 2
1

-0.08125

≡ -0.00010100110011 0011..........

≡ -0.000101001100110011 × 2^{-4} ; E' = 123 = $7B_{(H)}$
≡ -1.0100110011 0011..... E' = 1019 = $3FB_{(H)}$

1	01111011	0100110011 0011 0011.....	≡ BDA66666
1	01111111011	0100110011 0011 0011.....	≡ BFB4CCCCCCCCCCCC

0.08125 × 2
0 ← 0.1625 × 2
0 ← 0.325 × 2
0 ← 0.65 × 2
1 ← 1.3 × 2
0 ← 0.6 × 2
1 ← 1.2 × 2
0 ← 0.4 × 2
0 ← 0.8 × 2
1 ← 1.6 × 2
1 ← 1.2 × 2
0 ← 0.4 × 2

Scanned with CamScanner

(0 42047) Excus 102

* SPECIAL VALUES:-

The end values 0 & 255 of the excess-127 E' are used to represent special values.

1> when $E' = 0$ & $M = 0$, the value exact 0 is represented

2> when $E' = 0$ & $M \neq 0$, a denormal number is represented. Their value is $\pm 0.M \times 2^{-126}$ i.e., their nos are smaller than smallest normal number.

3> When $E' = 255$ & $M = 0$, the value infinity is represented, where infinity is the result of dividing a normal number by 0.

4> When $E' = 255$ & $M \neq 0$, the value represented is called "NOT A NUMBER" (NAN). NAN is the result of invalid operation such as $\frac{0}{0}$, $\sqrt{-1}$.

* EXCEPTIONS:-

According to IEEE standards, the processor sets the Exception flags if Underflow, Overflow, $\div$de by zero, Inexact or Invalid condition occurs during the program Execution.

> UNDER FLOW: In a single precision, if the number requires an exponent less than (−126) or in a double precision, if the number requires an exponent less than (−1022) to represent its normalized form, UNDER FLOW occurs.

> OVER FLOW: In a single precision, if the number requires an exponent greater than (+127) or in a double precision, if the number requires an exponent greater than (+1023) to represent its normalized form OVER FLOW occurs.

> DIVIDE BY ZERO: Divide by zero exception occurs when any number is divided by zero.

> INEXACT: Inexact is the name for a result that requires rounding in order to be represented in one of the normal format.

> INVALID: An Invalid exception occurs when an operation such as $\frac{0}{0}$ or $\sqrt{-1}$ are attempted.

When exception occurs, the result are set to special values. Systems & User-defined routines are used to handle such exceptions.

CHOPPING-OFF (OR) TRUNCATING:

If an 8-bit number is to be reduced to a 3-bit number, simply chop-off the high order bits as follows:

Suppose the binary number is (0.10101001). This can be reduced to 3-bit by neglecting all the bits after the third, so that the approximation is (0.101).

This process is called TRUNCATING or CHOPPING-OFF.

ROUNDING:

Instead of truncating, if the closed available value is chosen, it is called ROUNDING. In the above case, the rounded value is (0.11). In computer, after any arithmetic operation, it is always required to perform either Rounding or Truncating to get a representable value.

ARITHMETIC OPERATION USING FLOATING POINT NUMBER:

ADDITION & SUBTRACTION:

Choose the number with smaller exponent & shift its mantissa right number of times equal to difference in exponent.

Set the exponent of the result equal to larger exponent.

Perform Addition / Subtraction on the Mantissa & determine the sign of the result.

Normalize the resulting value.

Check for over-flow / Under-flow & take appropriate action.

Let $A = M_x 2^{E_x}$ and $B = M_y 2^{E_y}$

Then $Z = A \pm B$ [Make exponents equal]

Add single precision floating point numbers A & B where $A = 4490000$ & $B = 42A00000$

I step:

S E' M

$A \equiv 0 \vdots 10001001 \vdots 00100000000000000000000$

$B \equiv 0 \vdots 10000101 \vdots 01000000000000000000000$

$\therefore$ A & B are positive

$E'_A = 89_{(H)} = 137_{10} \Rightarrow E_A = E'_A - 127 = \underline{10}$

$M_A = 001000\ldots$

$E'_B = 85_{(H)} = 133_{10} \Rightarrow E_B = E'_B - 127 = \underline{6}$

$M_B = 01000\ldots$

$\therefore A \equiv 1.001 \times 2^{10}$

$B \equiv 1.010 \times 2^{6}$

~~Adjusting value of exponent of B is 10~~

$\therefore$ B has smaller exponent with difference

$\therefore$ Its mantissa is shifted right by 4-bits as shown

Shifted $M_B = 000101000\ldots$

II step:

Addition of Mantissa

$M_A \equiv 00100000\ldots$

$+ M_B \equiv 00010100\ldots$

Result $\equiv 00110100\ldots$

$\therefore A + B \equiv 0 \vdots 10001001 \vdots 00110100000000000000000$

$\equiv \underline{449A0000}$

$= 1.0011001 \times 2^{10}$

Subtraction of Mantissa

$M_A = 00100000\ldots$

$- M_B = 00010100\ldots$

Result $= 00001100$

$A - B = 0 \vdots 10001001 \vdots 000011000\ldots = 44860000$

* FLOATING POINT MULTIPLICATION :-

Let $X = M_x \times 2^{E_x}$ & $Y = M_y \times 2^{E_y}$ be the 2 floating point numbers. Then $Z = X * Y$

$$= (M_x * M_y) \times 2^{(E_x + E_y)}$$

i) Add the exponent & subtract the bias (127 in case single precision number & 1023 in case of double precision number)

ii) Multiply the mantissa & determine the sign of the result.

iii) Normalize the result.

iv) Check for overflow / underflow & take the appropriate action.

Ex: Perform the multiplication of

A = 36100000

B = D4100000

A ≡ 0 01101100 00100000000 0000 0000 0000

B ≡ 1 10101000 00100000000 0000 00000000

A ; S = 0 ; $E' = 6C = 108_{(10)}$; $E = E' - 127$

$= 108 - 127 = -19$

$\therefore A \equiv +1.001 \times 2^{-19} \equiv \cancel{+1001 \times 2^{-22}}$

B ; S = 1 ; $E' = A8 = 168_{(10)}$; $E = E' - 127$

$= 168 - 127 = 41$

$\therefore B = -1.001 \times 2^{41} = \cancel{-1001 \times 2^{38}}$

$\cancel{1001} * \cancel{-1001} \equiv * \cancel{0111} = \cancel{1001} * \cancel{2-1}$

~~000 0 111~~

~~1001 0~~

~~Result = 1001111~~ $\xrightarrow{2's}$ ~~0110001~~

$\therefore$ ~~Result~~ $\equiv 1001111 \times 2^{38-22}$

$\equiv \cancel{1001111 \times 2^{16}} = \cancel{1.001111 \times 2^{22}}$

~~1001~~ 01110 ~~+100-1~~ 2 ~~-1~~

~~22+127~~

Result $= (-1.001 \times 1.001) \times 2^{41-19}$

$= -1.010001 \times 2^{22}$

1 | 10010101 | 01000100000......

$\equiv$ CAA20000

$E = 22$

$E' = 149$

$16 \overline{)149}$
9-5

* FLOATING POINT DIVISION:-

Let $X = M_x \times 2^{E_x}$ & $Y = M_y \times 2^{E_y}$

$$Z = \frac{X}{Y} = \frac{M_x}{M_y} \cdot 2^{E_x - E_y}$$

i) Subtract the exponent. Then the bias being vanished, add the bias (127 or 1023). Without the addition of bias, it will be zero-exponent.

ii) Divide the mantissa & determine the sign of the result.

iii) Normalize the resulting value if necessary.

iv) Check for overflow / underflow & take suitable action.

Ex: Divide A = 36100000 by B = D4100000

$A \equiv 1.001 \times 2^{-19}$

$B \equiv -1.001 \times 2^{41}$

Result $= \frac{1.001}{-1.001} \times 2^{-19-41} = -1.0000 \times 2^{-60}$

$E = -60$; $E' = 127 - 60 = 67$

$16 \overline{)67}$
4-3

$\therefore$ Result $\equiv$ 1 | 01000011 | 00000000...

= A1800000

* MEMORY LOCATION & ADDRESSES:

Refer Text