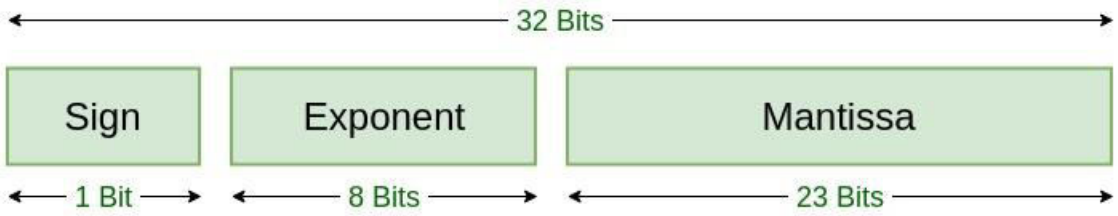
Floating Point Adder and Multiplier

IEEE Standard 754 FP Numbers

IEEE 754 has 3 basic components:

- **The Sign of Mantissa –**
  This is as simple as the name. 0 represents a positive number while 1 represents a negative number.
- **The Biased exponent –**
  The exponent field needs to represent both positive and negative exponents. A bias is added to the actual exponent in order to get the biased exponent.
- **The Normalised Mantissa –**
  The mantissa is part of a number in scientific notation or a floating-point number, consisting of its significant digits. Here we have only 2 digits, i.e. O and 1. So a normalised mantissa is one with only one 1 to the left of the decimal.

Single Precision IEEE FP Standard

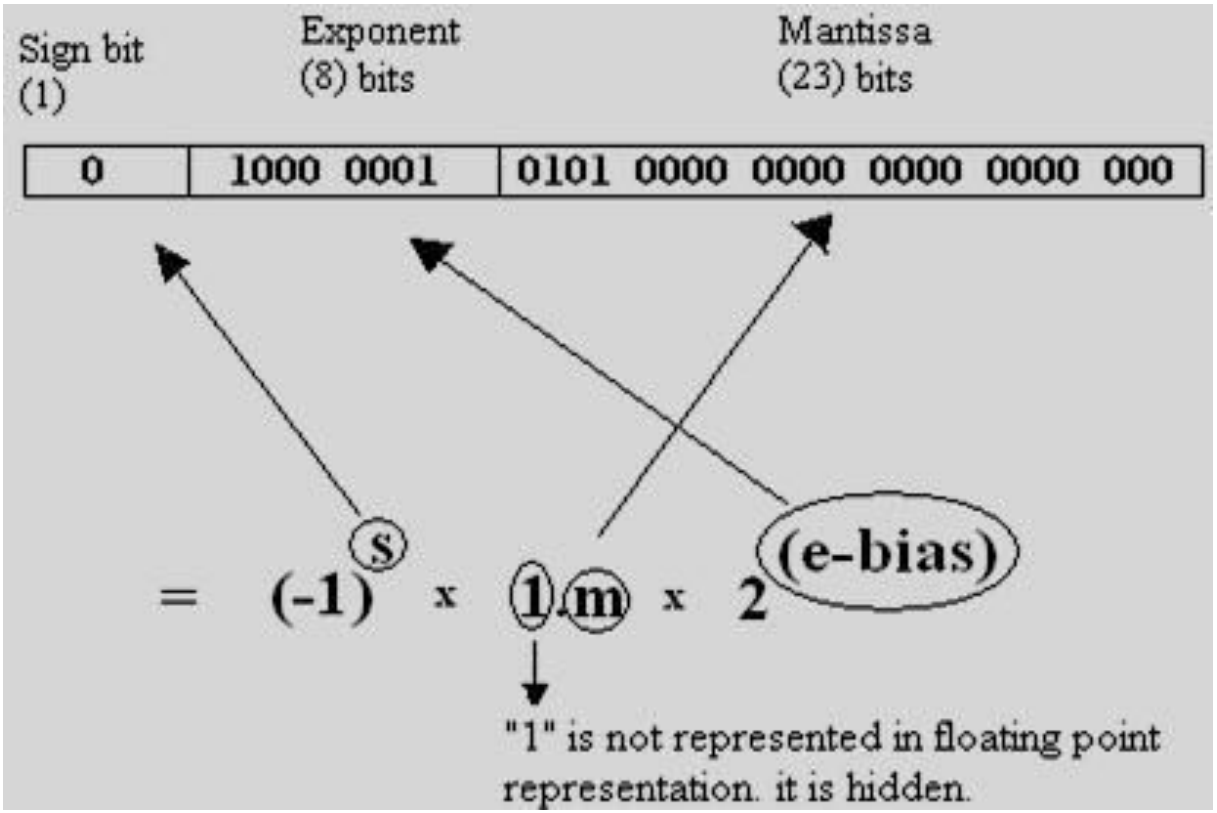


Single Precision
IEEE 754 Floating-Point Standard

$$Decimal\ Equivalent = (-1)^s * 1.m * 2^{e-bias}$$

$$bias = 2^{eb-1} - 1 \qquad bias = 2^{8-1} - 1 = 127$$

Where,
s = Sign bit
m = Mantissa
e = Exponent
eb = Number of bits in exponent



Note: "1" is hidden in the representation of IEEE 754 floating point word, since it takes up an extra bit location and it can be avoided. It is understood that we need to append the "1" to the mantissa of a floating point word for conversions and calculations.

Example

85.125

85 = 1010101

0.125 = 001

85.125 = 1010101.001

= 1.010101001 x 2^6 (Normalised)

Sign = 0(1 bit)

Single precision biased exponent, 127+6=133

133 = 10000101(8 bits)

Mantissa = 01010100100000000000000(23 bits)

= 0 10000101 01010100100000000000000(32 bits)

IEEE 754 Floating Point Addition

X3 = X1 + X2 ie., X3 = $((-1)^{s1} \times 1.M1 \times 2^{E1}) +/- ((-1)^{s2} \times 1.M2 \times 2^{E2})$

X1 and X2 can be added iff E1==E2

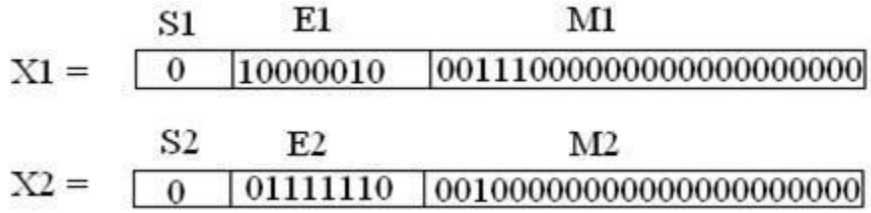
Algorithm:

1. Assume(|X1|>|X2|) if not swap X1 and X2
2. Set E3=E1 and S3=S1
3. Calculate D = E1-E2
4. Left Shift M2 by D (Denormalize X2)
5. Compute the sum/difference of the mantissas depending on the sign bit S1 and S2.
   - If signs of X1 and X2 are equal (S1 == S2) then add the mantissas
   - If signs of X1 and X2 are not equal (S1 != S2) then subtract the mantissas (2's compliment addition)
6. Normalize the resultant mantissa (M3) if needed. (1.M3 format) and the initial exponent result E3=E1 needs to be adjusted according to the normalization of mantissa.

X1 = 9.75

X2= 0.5625

Equivalent floating point binary words are



1) Abs (A) > Abs (B)? Yes.
2) Result of Initial exponent E3 = E1 = 10000010 = $130_{(10)}$
3) E1 - E2 = (10000010 - 01111110) => (130-126)=4
4) Shift the mantissa M2 by (E1-E2) so that the exponents are same for both numbers.

$$
\begin{aligned}
\mathbf{1.}\boldsymbol{M2} &= 1.00100000000000000000000 \\
&= 00001.00100000000000000000000 \\
&= 0.00010010000000000000000 \quad \text{(Aligned mantissa)}
\end{aligned}
$$

5) Sign bits of both are equal? Yes. Add the mantissa's

$$
\begin{array}{ll}
 & 1.00111000000000000000000 \ \text{(1.M1)} \\
+ & 0.00010010000000000000000 \ \text{(aligned M2)} \\
\hline
 & 1.01001010000000000000000 \ \mathbf{1.}\boldsymbol{M3}
\end{array}
$$

6) Normalization needed? No, (if Normalization was required for M3 then the initial exponent result E3=E1 should be adjusted accordingly)

7) Result

	S3	E3	M3
X3 =	0	10000010	01001010000000000000000

IEEE 754 Floating Point Multiplication

Result X3 = X1 * X2

$= (-1)^{s1} (M1 \times 2^{E1}) * (-1)^{s2} (M2 \times 2^{E2})$

Algorithm:

1. Check if one/both operands = 0 or infinity. Set the result to 0 or inf. i.e. exponents = all "0" or all "1".
2. S1, the signed bit of the multiplicand is XOR'd with the multiplier signed bit of S2. The result is put into the resultant sign bit.
3. The mantissa of the Multiplier (M1) and multiplicand (M2) are multiplied and the result is placed in the resultant field of the mantissa (truncate/round the result for 24 bits). M3=M1 * M2
4. The exponents of the Multiplier (E1) and the multiplicand (E2) bits are added and the bias value is subtracted from the added result. The subtracted result is put in the exponential field of the result block. E3=E1+E2-bias
5. Normalize the sum, either shifting right and incrementing the exponent or shifting left and decrementing the exponent.

Example

- Let's consider two decimal numbers
  X1 = 125.125 (base 10)
  X2 = 12.0625 (base 10)
  X3= X1 * X2 = 1509.3203125
- Equivalent floating point binary words are

	S1	E1	M1
X1 =	0	10000101	11110100100000000000000

	S2	E2	M2
X2 =	0	10000010	10000010000000000000000

2. Find the sign bit by xor-ing sign bit of A and B
   i.e. Sign bit = > (0 xor 0) => 0
3. Multiply the mantissa values including the "hidden one". The Resultant product of the 24 bits mantissas (M1 and M2) is 48bits (2 bits are to the left of binary point)

$$M3 = 1.M1 * 1.M2 = (10).111100101010100100000000000000000000000000000000$$

$$M3 = 1.011110010101010010000000000000000000000000000000 \times 2^{1}$$

(Normalized binary)

Hidden "1"

4. If M3 (48) = "1" then left shift the binary point and add "1" to the exponent else don't add anything. This normalizes the mantissa. Truncate the result to 24 bits. Add the exponent "1" to the final exponent value.

   Find exponent of the result. = E1 + E2 -bias + (normalized exponent from step 2) = $(10000101)_2 + (10000010)_2$ - bias + 1 = 133 + 130 - 127 + 1 = 137.
5. Normalised Value.

X3 =

S3	E3	M3
0	10001001	01111001010101001000000

Note:

For Addition Use Recursive Doubling Adder

For Shifting use barrel shifter

For Multiplier use Wallace tree multiplier