

Floating Point Numbers

Sriram Dayanandan

Fixed Point Representation

Any integer I can be represented in binary notation as a string B of 0s and 1s such that

$$I = 2^{n-1} B_n + 2^{n-2} B_{n-1} + \dots + 2^0 B_1$$

$$B = B_n B_{n-1} \dots B_1$$

Equation 1

E.g.

$$2_{10} = 10_2$$

$$257_{10} = 10000001_2$$

In the same way any real number R can be represented in binary notation as a string B of 0s and 1s such that

$$R = 2^{n-1} D_n + 2^{n-2} D_{n-1} + \dots + 2^0 D_1 + 2^{-1} F_1 + \dots + 2^{-m} F_m$$

$$B = D_n D_{n-1} \dots D_1 \bullet F_1 \dots F_m$$

Equation 2

E.g.

$$2.45_{10} = 10.01110001101110_2 \text{ [} m=2, n=\infty \text{]}$$

$$55.25_{10} = 110111.01_2 \text{ [} m=6, n=2 \text{]}$$

In the above equation, $D_n \dots D_1$ represents the whole part, whereas $F_1 \dots F_m$ represents the fractional part. The decimal point is between the whole part and the fractional part. Hence B expresses the number R using $m + n$ number of bits with m bits fixed for the whole part and n bits fixed for the fractional part. The above representation is known as Fixed Point representation of real numbers, because of m and n being fixed. The dynamic range of such a representation is small.

Floating Point Numbers

Sriram Dayanandan

Floating Point Representation

In the previous section we saw that the fixed point representation of real numbers is basically a simple extension to the binary representation of integers. Although simple to comprehend, such a representation is inefficient because of the fixed number of bits for the whole and fractional parts. If the number of bits allocated for the whole and fractional parts could be varied depending on the number, the dynamic range and precision can be increased. Such a representation where the number of bits for the whole and fractional part varies is known as floating point because the decimal point “floats”.

Normal Form

2.45_{10} can be expressed in exponential form as 2.45×10^0 or 0.245×10^1 , and 55.25_{10} can be expressed as 55.25×10^0 or 5.525×10^1 , 0.5525×10^2 , etc.

2.45×10^0 and 5.525×10^1 are known as the normal exponential forms.

Similarly we can express 10.01110001101110_2 in binary exponential form as $10.01110001101110_2 \times 2^0$ or $1.001110001101110_2 \times 2^1$, and 110111.01_2 can be expressed as $110111.01_2 \times 2^0$ or $1.1011101_2 \times 2^5$

Hence $1.001110001101110_2 \times 2^1$ and $1.1011101_2 \times 2^5$ are known as the normal forms.

To generalize for any base B, the normal form of a number N is

$$D.F_1F_2F_3\dots F_k \times B^{\text{int}(\log_B N)}$$

$$D \in \{1, 2, \dots, B-1\}, F_i \in \{0, 1, \dots, B-1\}$$

$$N = (D + \sum F_i B^{-i}) \times B^{\text{int}(\log_B N)}$$

Equation 3

When expressed in normal form, $\text{int}(\log_B N)$ is known as the exponent (e) and $D + \text{SUM}(F_i B^{-i})$ is known as the mantissa (m).

Floating Point Numbers

Sriram Dayanandan

$$m_B = D + \sum F_i B^{-i}$$

$$e_B = \text{int}(\log_B N)$$

Equation 4

N can be regarded as a function from $R \times I$ to R of the mantissa and the exponent as given in the equation below.

$$f_B : R \times I \rightarrow R$$

$$N = f_B(m_b, e_b)$$

Equation 5

For base 2, the normal form for N is given by

$$D.F_1F_2F_3\dots F_k \times 2^{\text{int}(\log_2 N)}$$

$$D \in \{1\}, F_i \in \{0,1\}$$

$$N = (1 + \sum F_i 2^{-i}) \times 2^{\text{int}(\log_2 N)}$$

Equation 6

The expressions for the exponent, mantissa and the function f_2 are given below.

$$m_2 = 1 + \sum F_i 2^{-i}$$

$$e_2 = \text{int}(\log_2 N)$$

$$f_2 : R \times I \rightarrow R$$

$$N = f_2(m_2, e_2)$$

Equation 7

Expressing N as a function $R \times I$ to R does not help us in terms of floating point representation on a computer as we are still on our way to describe real numbers.

Floating Point Numbers

Sriram Dayanandan

Let N_m and N_e be the number of bits allocated to represent m and e respectively,

Consider function f_{2SB} as defined below.

$$f_{2SB} : S_B \times I \rightarrow R$$
$$N = f_{2SB}((F_1 \dots F_{N_m}), e_2)$$

Equation 8

Where, S_B is a set of all binary strings of length N_m and $(F_1 \dots F_{N_m}) \in S_B$ represents a string of 0s and 1s for fractional part of N with 0s appended if necessary. Here we have expressed N as a function $S_B \times I$ to R and hence have removed the domain dependency on the set of real numbers R .

Now if we express the binary string $(F_1 \dots F_{N_m})$ as an integer I_F such that

$$I_F = \sum F_i 2^{i-1}$$

Equation 9

$$f_{2i} : I \times I \rightarrow R$$
$$N = f_{2i}(I_F, e_2)$$

Equation 10

Floating Point Numbers

Sriram Dayanandan

f_{2i} seems more mathematically convenient as compared to f_{2SB} because of domain dependency on just the set of integers I .

Relationship between I_F and the mantissa m_2

Then we can express m_2 in terms as I_F and vice-versa as follows. The expressions also take the sign of the number into account.

$$m_2 = \left(\frac{I_F + 2^{Nm}}{2^{Nm}} \right) \times \text{sgn}(N)$$

$$I_F = 2^{Nm} (|m| - 1)$$

Equation 11

We can now define as f_{2i} as follows.

$$f_{2i} : I \times I \times \{-1,1\} \rightarrow R$$

$$f_{2i}(I_F, e_2, \text{sgn}) = 2^{e_2} \left(\frac{I_F + 2^{Nm}}{2^{Nm}} \right) \times \text{sgn} = \left(\frac{I_F + 2^{Nm}}{2^{Nm-e_2}} \right) \times \text{sgn}$$

Equation 12

IEEE 754 Floating point representation

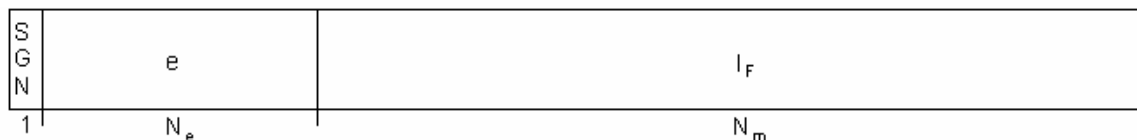


Figure 1

The IEEE 754 standard representation assigns 1 bit for the sign of the number, N_m and N_e number of bits to represent the exponent e and the integral representation of mantissa m as I_F . In order to represent negative exponents, the exponent e itself is stored in excess- $(2^{N_e-1}-1)$ format so that positive exponents

Floating Point Numbers

Sriram Dayanandan

are expressed between 2^{N_e-1} to 2^{N_e} and negative exponents between 0 to $2^{N_e-1}-2$ with $2^{N_e-1}-1$ representing the exponent 0.

$$e = e_2 + (2^{N_e-1} - 1)$$

Equation 13

The standard defines 2 mandatory formats single-precision and double-precision as shown in the table below.

Format	Total Size	N_e	N_m
Single	32 bits	8	23
Double	64 bits	11	52

Table 1

Using equations 7, 11 and 12 one can easily compute the IEEE 754 representations for any real number in the above formats and vice-versa.

Mathematical Operations

Addition/Subtraction

Let R_1, R_2, R_3 be three real numbers such that $R_3=R_1\pm R_2$ and

$$R_1 = f_{2i}(I_{F1}, e_{21}, s_1), R_2 = f_{2i}(I_{F2}, e_{22}, s_2), R_3 = f_{2i}(I_{F3}, e_{23}, s_3)$$

Equation 14

We wish to express I_{F3} in terms of I_{F1} and I_{F2} .

Let c be an integer such that

$$c = e_{22} - e_{21}$$

Equation 15

Then we get the following equation.

Floating Point Numbers

Sriram Dayanandan

$$R_1 = \left(\frac{I_{F1} + 2^{Nm}}{2^{Nm-e_{21}}}\right) \times s_1, R_2 = \left(\frac{I_{F2} + 2^{Nm}}{2^{Nm-(e_{21}+c)}}\right) \times s_2 = 2^{-c} \left(\frac{I_{F2} + 2^{Nm}}{2^{Nm-e_{21}}}\right) \times s_2$$

Equation 16

Hence by adding/subtracting R1 and R2 we get

$$\left(\frac{I_{F3} + 2^{Nm}}{2^{Nm-e_{23}}}\right) = \left(\frac{[I_{F1} + 2^{Nm}] \times s_1 \pm \left[\left(\frac{I_{F2} + 2^{Nm}}{2^c}\right)\right] \times s_2 - 2^{Nm} + 2^{Nm}}{2^{Nm-e_{21}}}\right)$$

$$I_{F3} = [I_{F1} + 2^{Nm}] \times s_1 \pm \left[\left(\frac{I_{F2} + 2^{Nm}}{2^c}\right)\right] \times s_2 - 2^{Nm}, e_{23} = e_{21}$$

Equation 17

We have now expressed I_{F3} in terms of I_{F1} and I_{F2} . Although mathematically correct, equation 17 does not take into account the fact that the addition/subtraction is being executed by a digital computer on two N_m bit numbers. Their addition/subtraction could cause an overflow/underflow which would yield incorrect results. Hence we would need to express I_{F3} in terms of I_{F1} and I_{F2} in a digital friendly way.

We know that I_{F1} and I_{F2} are N_m bit unsigned integers. Looking at equation 17, addition by 2^{Nm} can cause an overflow giving a N_m+1 bit result. Moreover the addition of the first two potentially N_m+1 bit terms themselves could cause a N_m+2 bit result. Moreover since we would also like to consider the sign we would need an additional bit for the sign. Hence we would need to express both I_{F1} and I_{F2} as N_m+3 bit signed integers with the $(N_m+2)^{th}$ bit representing the sign. Negative numbers will be stored in N_m+3 bit 2s complement form.

Now $N_m=23$ for single precision and 52 for double precision format numbers. This means we would need a 26-bit and a 55-bit adder unit respectively. Hence we shall use a 64-bit adder unit for our operations. Even though we are using a 64-bit adder unit, we would still be representing I_{F1} and I_{F2} as N_m+3 bit signed integers with the $(N_m+2)^{th}$ bit representing the sign.

Floating Point Numbers

Sriram Dayanandan

We define the function *compl* which computes the complement of a n-bit integer *i* and the function *neg* which computes the n-bit 2s complement negation of an integer *I* as follows.

$$\text{compl} : I \times I \rightarrow I$$

$$\text{compl}(i, n) = \text{xor}(i, 2^n - 1)$$

$$\text{neg} : I \times I \rightarrow I$$

$$\text{neg}(i, n) = \text{compl}(i, n) \oplus 1 \text{ (bit - wise addition)}$$

Equation 18

We also define a function *inbit* which computes the n bit signed representation of integer *i* as follows.

$$\text{inbit} : I \times I \times \{-1, 1\} \rightarrow I$$

$$\text{inbit}(i, n, s) = i \text{ if } s \geq 0 \text{ and } \text{neg}(i, n) \text{ if } s \leq -1$$

Equation 19

Procedure 1 below shows the procedure for addition/subtraction and obtains values for I_{F3} , e_{23} and s_3 on a digital machine.

Floating Point Numbers

Sriram Dayanandan

$$in_1 = inbit(I_{F1} + 2^{Nm}, 2^{Nm+3}, s_1)$$

$$in_2 = inbit\left(\frac{I_{F2} + 2^{Nm}}{2^c}, 2^{Nm+3}, s_2\right)$$

$t_1 = in_1 \oplus in_2$ for addition and $in_1 \oplus neg(in_2, 2^{Nm+3})$ for subtraction

$s_3 = 1$ if $and(t_1, 2^{Nm+2}) = 0$ and -1 otherwise

$t_2 = t_1$ if $s_3 = 1$ and $neg(t_1, 2^{Nm+3})$ otherwise

$$t_3 = t_2 \oplus neg(2^{Nm}, 2^{Nm+3})$$

$I_{F3} = and(t_3, 2^{Nm} - 1)$ if $and(t_3, 2^{Nm}) = 0$

$$= \frac{and(t_3, 2^{Nm} - 1)}{2} \text{ if } and(t_3, 2^{Nm+1}) = 0$$

$= and(t_3, 2^{Nm} - 1) \times 2$ otherwise

$e_{23} = e_{21}$ if $and(t_3, 2^{Nm}) = 0$

$= e_{21} + 1$ if $and(t_3, 2^{Nm+1}) = 0$

$= e_{21} - 1$ otherwise

Procedure 1

Similarly we can define the equations and procedures for multiplication and division as follows.

Floating Point Numbers

Sriram Dayanandan

Multiplication

$$\begin{aligned}\left(\frac{I_{F3} + 2^{Nm}}{2^{Nm-e_{23}}}\right) &= \left(\frac{I_{F1} + 2^{Nm}}{2^{Nm-e_{21}}}\right) \times s_1 \times \left(\frac{I_{F2} + 2^{Nm}}{2^{Nm-e_{22}}}\right) \times s_2 \\ &= \frac{(I_{F1} + 2^{Nm}) \times (I_{F2} + 2^{Nm})}{2^{Nm-(e_{21}+e_{22})}} \times s_1 s_2 \\ &= \frac{\left(\frac{I_{F1}I_{F2}}{2^{Nm}} + I_{F1} + I_{F2}\right) + 2^{Nm}}{2^{Nm-(e_{21}+e_{22})}} \times s_1 s_2 \\ \therefore I_{F3} &= \left(\frac{I_{F1}I_{F2}}{2^{Nm}} + I_{F1} + I_{F2}\right), e_{23} = e_{21} + e_{22}, s_3 = s_1 s_2\end{aligned}$$

Equation 20

Division

$$\begin{aligned}\left(\frac{I_{F3} + 2^{Nm}}{2^{Nm-e_{23}}}\right) &= \frac{\left(\frac{I_{F1} + 2^{Nm}}{2^{Nm-e_{21}}}\right) \times s_1}{\left(\frac{I_{F2} + 2^{Nm}}{2^{Nm-e_{22}}}\right) \times s_2} \\ &= \frac{\left(\frac{I_{F1} + 2^{Nm}}{I_{F2} + 2^{Nm}} - 2^{Nm}\right) + 2^{Nm}}{2^{Nm-(e_{21}-e_{22})}} \times \frac{s_1}{s_2} \\ \therefore I_{F3} &= \left(\frac{I_{F1} + 2^{Nm}}{I_{F2} + 2^{Nm}} - 2^{Nm}\right), e_{23} = e_{21} - e_{22}, s_3 = \frac{s_1}{s_2}\end{aligned}$$

Equation 21