Chapter 2
Floating-Point Numbers

Latest update: 29/08/2024

Table of Contents

I. Definitions....2
1. Signs, Mantissas and Exponents....2
2. Normalization of the Mantissa in its Binary Form....3
II. IEEE Standard for Floating-Point Arithmetic....4
1. Introduction....4
2. Representation of Normalized Floating-Point Numbers....4
3. Representation of the Number 0....6
4. Representation of Infinity....6
5. Representation of NaNs....6
6. Representation of Denormalized Floating-Point Numbers....7
7. The Standard in a Nutshell....7
III. Conversions....8
1. Converting Decimal Numbers into their Normalized IEEE Representations....8
2. Converting Normalized IEEE Representations into their Decimal Forms....8
3. Other Conversions....9

I. Definitions

1. Signs, Mantissas and Exponents

Representations of numbers can take various forms. One common way of representing them is scientific notation.

For instance, we can express the number 987.65 in many forms using scientific notation:

987.65
$0.0000098765 \times 10^{8}$
0.0098765×10^{5}
9.8765×10^{2}
98765.0×10^{-2}
$9876500000.0 \times 10^{-7}$

There are only two differences between all of these forms:
- **The power-of-ten exponent**.
- **The position of the point** (this is why we can say that the point is floating).

We find similar forms for negative numbers:

− 987.65
$-0.0000098765 \times 10^{8}$
-0.0098765×10^{5}
-9.8765×10^{2}
-98765.0×10^{-2}
$-9876500000.0 \times 10^{-7}$

We can also express any binary number in scientific notation. The power of ten has to be replaced by a power of two. Take for instance the following positive or negative binary number:

± 110.01
$\pm 0.0000011001 \times 2^{8}$
$\pm 0.0011001 \times 2^{5}$
$\pm 1.1001 \times 2^{2}$
$\pm 11001.0 \times 2^{-2}$
$\pm 1100100000.0 \times 2^{-7}$

We can conclude that any binary number can be expressed in the following form:

$$(-1)^s \times m \times 2^e$$

- s represents the sign of the number. It can be either 0 or 1. Obviously, if $s = 0$, the number is positive, and if $s = 1$, the number is negative.
- m is called the ‘mantissa’ (also ‘significand’) and is an integer or a fraction.
- e is called the ‘exponent’ and is an integer.

Take for instance the following number: $-0.0011001_2 \times 2^5$

- $s = 1$
- $m = 0.0011001_2$
- $e = 5$

2. Normalization of the Mantissa in its Binary Form

In its binary form, a normalized mantissa always begins with a ‘1’ followed by a point ‘.’. This ‘1’ is called the ‘leading 1’. For example, in the table below, only one mantissa is normalized:

110.01	
0.0000011001×2^8	
0.0011001×2^5	
$\mathbf{1.1001} \times 2^2$	← This mantissa is normalized because it begins with ‘**1.**’
11001.0×2^{-2}	
$1100100000.0 \times 2^{-7}$	

Therefore, there is only one way to represent a binary number in scientific notation with a normalized mantissa.

It is noteworthy that **the number 0 cannot be expressed with a normalized mantissa**, because the latter is greater than or equal to 1 and a power of two is greater than 0 (their product is never null).

II. IEEE Standard for Floating-Point Arithmetic

1. Introduction

The Institute of Electrical and Electronics Engineers (IEEE) has defined a floating-point standard, which is known as **‘the IEEE 754 floating-point standard’**. This standard was established in 1985. A lot of different formats were once used to encode floating-point numbers, but today, the IEEE standard dominates the computer industry. It is commonly found in both hardware and software.

The IEEE standard defines three binary fields and an exponent bias.

S	E	M

- *S* is the ‘sign field’. It is used to encode the **sign** of a floating-point number.
- *E* is the ‘exponent field’. It is used to encode the **exponent** of a floating-point number.
- *M* is the ‘mantissa field’. It is used to encode the **mantissa** of a floating-point number.

The **exponent bias** (or **bias**) is a special value used to encode the exponent.

The size of the fields and the value of the bias are relative to the precision defined by the standard. Actually, the standard defines several levels of precision. The table below gives the size of the fields and the value of the bias according to some levels of precision.

	S (bits)	E (bits)	M (bits)	Total (bits)	Bias
Half Precision	1	5	10	16	15
Single Precision	1	8	23	32	127
Double Precision	1	11	52	64	1,023
Quadruple Precision	1	15	112	128	16,383

The two most commonly used levels of precision are the **‘single precision’** and **‘double precision’**.

2. Representation of Normalized Floating-Point Numbers

For a reason discussed later, the minimum value of *E* (all of its bits are 0s) and the maximum value of *E* (all of its bits are 1s) are reserved for special values. Therefore, the lines of reasoning in this part are not valid for these two extreme values of *E*.

S	E	M
Φ	≠ 00..0 ≠ 11..1	ΦΦ.....Φ

Φ is either 0 or 1

We have seen that any binary number could be expressed in the following form:

$$(-1)^s \times m \times 2^e$$

The IEEE standard uses this form to encode the sign (s), the exponent (e), and the mantissa (m). These encoded values are then stored in the sign field (S), the exponent field (E) and the mantissa field (M).

The relations between the values (s, e, m) and their respective fields (S, E, M) are:

- $s = S$
- $e = E$ – bias
- $m = (1.M)_2$

From these expressions, a normalized number can be expressed in the following form:

$$(-1)^S \times (1.M)_2 \times 2^{E - bias}$$

- **The *S* field contains the value of *s***. Therefore, if $S = 0$, the number is positive, and if $S = 1$, the number is negative.

- **The *E* field contains the biased exponent**. This field needs to represent both positive and negative exponents. To do so, the exponent is biased; that is to say a bias is added to it. The biased exponent is then stored into the field ($E = e$ + bias). For example, in the single precision, the value of the 8-bit biased exponent (E) is between 1 and 254 (the values 0 and 255 are reserved for special encodings), which means that the value of the exponent (e) is between −126 and +127.

- **The *M* field contains the fractional part of the normalized mantissa**. Therefore, to encode a number in this format, **the mantissa has to be previously normalized**. Thanks to this normalization, the ‘leading 1’ does not have to be stored in this field.

As we have already said, the number 0 cannot be encoded with a normalized mantissa. So, this special value needs to be encoded in a different way. Actually, the number 0 is not the sole special value to be encoded, which is why the minimum and maximum values of E are reserved.

3. Representation of the Number 0

The representation of the number 0 is denoted when both *E* and *M* are 0s.

S	E	M
Φ	00.....0	00.....0

Φ is either 0 or 1

Note that since *S* is either 0 or 1, '−0' and '+0' can be considered as distinct values if needed.

4. Representation of Infinity

The representations of '+∞' and '−∞' are denoted when all the bits of *E* are 1s and *M* is 0. The *S* field represents the sign.

S	E	M
Φ	11.....1	00.....0

Φ is either 0 or 1

5. Representation of NaNs

A 'Not a Number' (NaN) represents an indeterminate or invalid value (e.g. division by zero) as well as any entity that is not a number.

The NaNs are denoted when all the bits of *E* are 1s and *M* is not 0.

S	E	M
Φ	11.....1	≠ 00.....0

Φ is either 0 or 1

6. Representation of Denormalized Floating-Point Numbers

The IEEE standard allows encoding numbers with a denormalized mantissa when *E* is 0 and *M* is not 0.

S	E	M
Φ	00...0	≠ 00.....0

Φ is either 0 or 1

The relations between the values (*s*, *e*, *m*) and their respective fields (*S*, *E*, *M*) are:

- $s = S$
- $e = 1 - bias$
- $m = (0.M)_2$

It is noteworthy that the exponent (*e*) is fixed. From these expressions, a denormalized number can be expressed in the following form:

$$(-1)^S \times (0.M)_2 \times 2^{1-bias}$$

In terms of absolute values, the largest denormalized number is smaller than the smallest normalized number, which explains why they are often called ‘subnormal numbers’. In other words, they fill the gap between zero and the smallest normalized number.

7. The Standard in a Nutshell

E	M	Correspondence	Comment
00.....0	00.....0	± 0	Zero
00.....0	≠ 00.....0	$(-1)^S \times (0.M)_2 \times 2^{1-bias}$	Denormalized numbers
≠ 00....0 ≠ 11....1	ΦΦ.....Φ	$(-1)^S \times (1.M)_2 \times 2^{E-bias}$	Normalized numbers
11.....1	00.....0	± ∞	Infinity
11.....1	≠ 00.....0	NaN	Not a Number

Φ is either 0 or 1

III. Conversions

1. Converting Decimal Numbers into their Normalized IEEE Representations

To perform the conversion, you can apply the following five-step method:
1. Determine the value of *S* according to the sign of the decimal number.
2. Convert the absolute value of the decimal number into its binary form (it can be a fraction).
3. Normalize the binary form. From this normalization, you can easily deduce *M* and *e*.
4. Work out *E* according to *e* and the bias: $E = e + bias$.
5. Write down the final result in its binary form.

Example:
Convert the number −145.625 into its binary single-precision floating-point representation.
1. **S = 1**
2. $|-145.625| = 145.625 = 10010001.101_2$
   $0.625 \times 2 = \mathbf{1}.25$
   $0.25 \times 2 = \mathbf{0}.5$
   $0.5 \times 2 = \mathbf{1}$
3. $145.625 = (1.0010001101)_2 \times 2^7$
   M = 00100011010…0₂ and e = 7
4. E = e + bias = 7 + 127 = 6 + 128
   E = 1000 0110₂
5. **−145.625 → 1 10000110 00100011010000000000000**

2. Converting Normalized IEEE Representations into their Decimal Forms

To perform the conversion, you can apply the following five-step method:
1. Determine the sign of the number according to the value of *S*.
2. Work out *e* according to *E* and the bias: $e = E - bias$.
3. Determine the value of *m* in its binary form: $m = (1.M)_2$.
4. Write down the result in the following form: $\pm m \times 2^e$.
5. Simplify if necessary by (re)moving the point and convert into decimal.

Example:
Convert the following double-precision floating-point number into its decimal representation:
$2401\ 8000\ 0000\ 0000_{16} = 0010\ 0100\ 0000\ 0001\ 1000\ 0000\ldots\ldots0_2$
1. S = 0 → **positive**
2. $e = E - bias = 010\ 0100\ 0000_2 - 1023 = 576 - 1023$ **= –447**
3. **m** $= (1.M)_2 =$ **$(1.00011)_2$**
4. $+m \times 2^e = (1.00011)_2 \times 2^{-447}$
5. $= (100011)_2 \times 2^{-452}$
   $=$ **35×2^{-452}**

3. Other Conversions

Encoding zeros, infinities and NaNs is very simple; there is no particular difficulty in converting them.

To convert denormalized numbers, you can use similar methods used for normalized numbers. The main differences are that $e = 1 - bias$ and $m = (0.M)_2$. Actually, converting denormalized numbers is even easier because their exponents are fixed and their mantissas do not have to be normalized.