Floating-point Arithmetic

Reading: pp. 312-328

Floating-Point Representation

Non-scientific floating point numbers:

A non-integer can be represented as:

$$2^4 2^3 2^2 2^1 2^0 . 2^{-1} 2^{-2} 2^{-3} 2^{-4}$$

where you sum up the integer to the left of the decimal point and the fraction to the right.

So, what is -1101.0101 in decimal?

13 3/8 = 13.375 = 8 + 4 + 1 + 1/4 + 1/8 in binary is: 1101.011

What about repeating decimals? i.e. 0.3333333333

In general, the way to convert decimals into binary is the following:

1. Convert whole number part (if >1) using repeated division by two
2. Multiply the remaining fraction by the radix to get first digit right of the radix
3. Repeat step 2 until
    a. result is a whole number or
    b. a recognizable pattern emerges or
    c. you have extended the binary number to an acceptable precision

Examples:

1. What is 6.5859375 in binary using the above representation?

2. What is 1/3 = 0.3333333 in binary?

Scientific Notation:

You use floating point notation to represent very large or very small numbers.

Rewrite 53,760,000 using scientific notation:

Rewrite .00005376 using scientific notation:

The same thing can be done with binary numbers. The notation for this is:

$$\pm m \times B^{\pm e}$$

The binary number 1101.011 is the same as 1.101011×2^3

Let us now consider 16 bit floating point numbers in the following format:

Bit Number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Contents	s_e	e_4	e_3	e_2	e_1	e_0	s_m	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9

Where:

s_e: is the sign of the exponent
e_4 e_3 e_2 e_1 e_0: is the exponent
s_m: is the sign of the mantissa (significand)
m_1 m_2 m_3 m_4 m_5 m_6 m_7 m_8 m_9 is the mantissa or significand digits

With this breakdown of the 16 bits, our number is:

s_m **.** $m_1m_2m_3m_4m_5m_6m_7m_8m_9$ **x** $2^{\text{se e4e3e2e1e0}}$

The requirements that the first bit in the mantissa is nonzero and follows the radix point results in what is known as a **normalized** number. This normalization ensures that all numbers have a unique representation.

To **decode** a floating point number in the above format, consider the following example:

0	0	0	1	0	1	1	1	0	1	1	1	0	0	0	0

This pattern corresponds to $-.10111 \times 2^5$ (note: the exponent of 00101 is 5 in decimal notation). At this point we can either decode the mantissa and multiply by 2^5 or simplify things a bit by moving around the radix point.

$-.10111 \times 2^5 = -10111. \times 2^0 = -23$ in base 10

To **encode** a number in floating point notation, we simply reverse the process (remember to normalize the number):

3/8 is .011 in binary, so we have

> $.011 \times 2^{0} = .11 \times 2^{-1}$ (the right-hand term is the normalized form of the number)

and so we encode the number as:

1	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0

Examples for you:

P1. Encode -46 ½ as a 16-bit normalized floating point number using the above format.

P2. Decode the following 16-bit floating point number using the above format:

0	0	0	1	1	1	1	1	0	0	1	0	1	1	0	0

Q1: What is the range of the exponent using the above format?

Q2: What is the range of the mantissa using the above format?

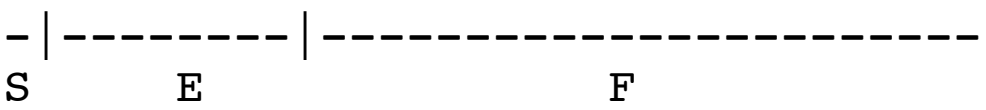
Q3: What is the largest number that can be represented by this 16-bit floating-point number?

IEEE Standard 754 Floating-point format

This standard was developed to allow portability of programs among various computers and also allow complex numerical programs. This standard has been widely adopted and therefore can be found in most programming environments.

There is a 32-bit and 64-bit representation. The principal feature of this representation features the hidden 1 since the numbers are all normalized. We will discuss this in greater detail later on.

Single-precision format



where S is the sign bit, E is a binary integer (8 bits), and F is a binary fraction (23 bits).

The value of this number is: $(-1)^S \times 2^{E-127} \times 1.F$

The representation used for the exponent is called a "biased representation." This means that a fixed value called the bias is subtracted from the binary value of E to get the actual value.

Q6: What is the bias of the IEEE 754 fp format? What are the ranges that the exponent can represent?

P3: Find what value C03E0000 represents using the IEEE 754 FP format.

For double-precison, the value of this number is: $(-1)^{S}x2^{E-1023}x1.F$

P4: Find the value C03E000000000000 represents using the IEEE 754 double-precision FP format.

It is important to note that not all bit patterns in the IEEE format are interpreted in the same way. I will discuss the single-precision examples and you can extend this to include double-precision. The following are some exceptions:

- An exponent of 0 with a fraction of 0 represents +0 or -0 depending on the sign bit.
- An exponent of all 1's with a fraction of 0 represents positive or negative infinity.
- An exponent of 0 with a nonzero fraction represents a denormalized number.
- An exponent of all 1's with a nonzero fraction represents a NaN (not a number) and is used to represent various exceptions.
- Exponents in the range of 1-254 with normalized fractions implies the resulting exponent value will be in the range of -126 to +127. Since the number is normalized, we do not need to represent the 1. This bit is implied and called the hidden 1. It is actually a way of adding one more bit of precision to the fraction.

P5: Let's go the other way and find the representation of -1.0 in IEEE 754 FP single-precision format. Express your result in hex only.