CSCI 313 Tutorial
Hsiao-Fen Fu

IEEE 754 Floating Point

IEEE 754 floating point is the most common representation today for real numbers on computers. This tutorial gives a brief overview of IEEE floating point and its representation.

What are floating point numbers?

There are several ways to represent real numbers on computers. Fixed point places a radix point somewhere in the middle of the digits, and is equivalent to using integers that represent portions of some unit. For example, one might represent 1/100ths of a unit; if you have four decimal digits, you could represent 10.82, or 00.01. Floating-point representation basically represents reals in scientific notation. Scientific notation represents numbers as a base number and an exponent. For example, 123.456 could be represented as 1.23456×10^2. In hexadecimal, the number 123.abc might be represented as $1.23abc \times 16^2$.

Floating-point solves a number of representation problems. Fixed-point has a fixed window of representation, which limits it from representing very large or very small numbers. Also, fixed-point is prone to a loss of precision when two large numbers are divided. On the other hand, floating-point employs a sort of "sliding window" of precision appropriate to the scale of the number. This allows it to represent numbers from 1,000,000,000,000 to 0.0000000000000001 with ease.

Formats

IEEE floating point number have three basic components: the sign, the exponent, and the mantissa. The mantissa is composed of the fraction and an implicit leading digit. The exponent base (2) is implicit and need not be stored.

There are two basic number formats in IEE754, single precision, and double precision. In addition there are two extended formats, which are only used as intermediate results while calculating. The following table shows the formats for single and double precision floating-point values. The number of bits for each filed are shown (bit ranges are in square brackets):

	Sign	Exponent	Fraction
Single precision	1[31]	8[30-23]	23[22-0]
Double precision	1[63]	11[62-52]	52[51-0]

The sign Bit

0 denotes a positive number; 1 denotes a negative number. Flipping the value of this bit flips the sign of the number.

The Exponent

The exponent field needs t represent both positive and negative exponents. To do this, a bias is added to the actual exponent in order to get the stored exponent. For IEEE single-precision floats, this value is 127. For double precision, the exponent field is 11 bits, and has a bias 1023.

For example, an exponent of zero in single precision means that 127 is stored in the exponent field. A stored value of 250 indicates an exponent of (200-127), or 123.

The significant

The significant, represents the precision bits of the number. It is composed of an implicit leading bit and the fraction bits.

In order to maximize the quantity of representable numbers, floating-point numbers are typically stored in normalized form. This basically puts the radix point after the first non-zero digit. An optimization is available in base two, since the only possible non-zero digit is 1. Thus we can just assume a leading digit of 1, and don't need to represent it explicitly. As a result, the mantissa has effectively 24 bits of resolution, by way of 23 fraction bits.

To sum up,

1. The sign bit is 0 for positive, 1 for negative.
2. The exponent's base is 2.
3. The exponent field contains 127 plus the true exponent for single-precision, or 1023 plus the true exponent for double precision.
4. The first bit of the mantissa (hidden bit) is typically assumed to be 1.

Special values

IEEE reserves exponent field values of all 0s and all 1s to denote special values in floating-point scheme.

Zero

Zero is not directly representable in te straight format, due to the assumption of a leading 1. Zero is a special value denoted with an exponent field of zero and a fraction field of zero.

Demoralized

If the exponent is all 0s, but the fraction is non-zero, then the value is a demoralized number, which does not have an assumed leading 1 before the binary point. Thus, for single precision, this represents a number $(-1)^s$ x 0.f x 2^{-126}, where s is the sign bit and f is the fraction.

Example

Now, let's see some examples.

1. Convert 145.84375 to binary floating point numbers in IEEE single precision.

   Step 1: convert 145.84357 to binary.
   145 = 10010011
   .84375 = .11111
   145.84375 = 10010011.11111

   Step 2: normalize and note sign
   145.84375 = $(-1)^0$ 1.001001111111 X 2^7

   Step 3: calculate excess 127 code for exponent
   e = 7 + 127 = 134 = 10000110

   Step 4: Assemble

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

2. Convert the IEEE 01000011010111010000000000000000 to decimal number.

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

   Step 1: calculate exponent to decimal
   10000110 = 134

   Step 2: rewrite IEEE to $(-1)^{\text{sign}}$ 1.significand X $2^{\text{EXP}-127}$ format
   $(-1)^0$ 1.1011101 X $2^{134-127}$

Step 3: normalize and convert to decimal

$(-1)^0$ 1.1011101 X 2^7 = + 1.1011101 X 2^7

= +11011101 = 221_{10}