CS61C Fall 2025

Precheck: C, Floating Point

Discussion 3

1 Discussion Pre-Check

1.1 The idea of floating point is to use the ability to move the radix (decimal) point wherever to represent a large range of real numbers as exact as possible.

True. Floating point:

- Provides support for a wide range of values. (Both very small and very large)
- Helps programmers deal with errors in real arithmetic because floating point can represent `+∞`, `-∞`, `NaN` (Not a Number)
- Keeps high precision. Recall that precision is a count of the number of bits in a computer word used to represent a value. IEEE 754 allocates a majority of bits for the significand, allowing for the use of a combination of negative powers of two to represent fractions.

1.2 Floating Point and Two's Complement can represent the same total amount of numbers (any reals, integer, etc.) given the same number of bits.

False. Floating Point can represent infinities as well as NaNs, so the total amount of representable numbers is lower than Two's Complement, where every bit combination maps to a unique integer value.

1.3 The distance between floating point numbers increases as the absolute value of the numbers increase.

True. The uneven spacing is due to the exponent representation of floating point numbers. There are a fixed number of bits in the significand. In IEEE 32-bit storage there are 23 bits for the significand, which means the LSB represents 2^{-23} times 2 to the exponent. For example, if the exponent is zero (after allowing for the offset) the difference between two neighboring floats will be 2^{-23}. If the exponent is 8, the difference between two neighboring floats will be 2^{-15} because the mantissa is multiplied by 2^8. Limited precision makes binary floating-point numbers discontinuous; there are gaps between them.

1.4 Floating Point addition is associative.

False. Because of rounding errors, you can find Big and Small numbers such that: `(Small + Big) + Big != Small + (Big + Big)`

FP approximates results because it only has 23 bits for the significand.

1.5 Why does normalized scientific notation always start with a 1 in base-2?

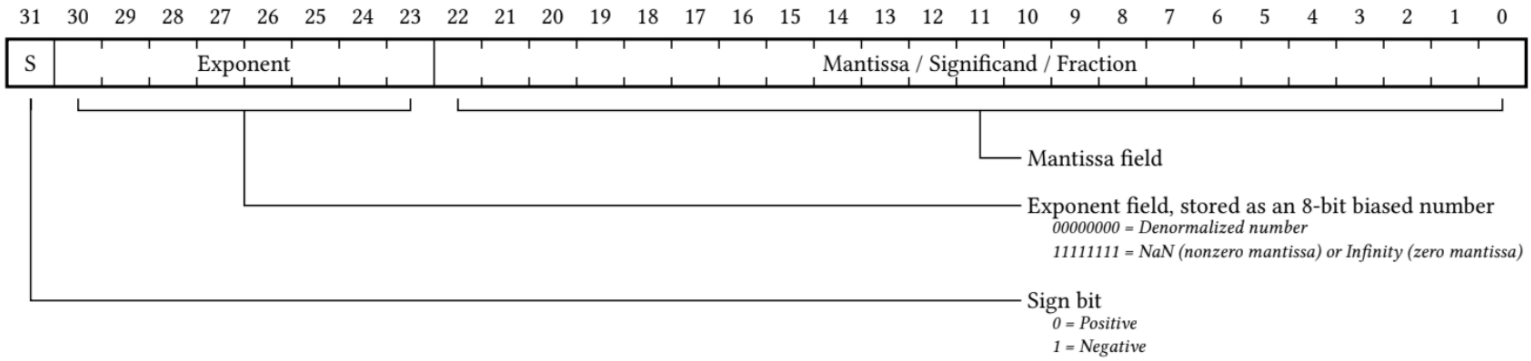
A non-zero digit is required prior to the radix in scientific notation, and since the only non-zero digit in base-2 is 1, the normalized value will always start with a 1.

2 Floating Point

The IEEE 754 standard defines a binary representation for floating point values using three fields.

- The *sign* determines the sign of the number (0 for positive, 1 for negative).
- The *exponent* is in **biased notation**. For instance, the bias is −127, which comes from -($2^{\{8-1\}}$ − 1) for single-precision floating point numbers. For double-precision floating point numbers, the bias is −1023
- The *significand* (or *mantissa*) is akin to unsigned integers but used to store a fraction instead of an integer and refers to the bits to the right of the leading "`1`" when normalized. For example, the significand of `1.010011` is `010011`.

The table below shows the bit breakdown for the single-precision (32-bit) representation. The leftmost bit is the MSB, and the rightmost bit is the LSB.



For normalized floats:

$$\textbf{Value} = (-1)^{\text{Sign}} \times 2^{\text{Exp+Bias}} \times 1.\text{Significand}_2)$$

For denormalized floats:

$$\textbf{Value} = (-1)^{\text{Sign}} \times 2^{\text{Exp+Bias+1}} \times 0.\text{Significand}_2)$$

Exponent (Pre-bias)	Significand	Meaning
0	Anything	Denorm
1-254	Anything	Normal
255	0	± Infinity
255	Nonzero	NaN

Note that in the above table, our exponent has values from `0` to `255`. When translating between binary and decimal floating point values, we must remember that there is a bias for the exponent.