CS61C Spring 2026

Precheck: C, Floating Point, IEC

Discussion 3

1 Discussion Pre-Check

1.1 True or False: 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 True or False: 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 True or False: 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 True or False: 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.

$\boxed{1.6}$ Convert the following numbers into the quantity of bytes each term represents (you may leave your answer in terms of powers of 2). (See precheck section on IEC Prefixes for assistance)

a) 4 KiB

One KiB is 2^{10} and $4 = 2^2$ so 4 KiB = $2^2 \times 2^{10} = 2^{12}$ bytes

b) 2 MiB

One MiB is 2^{20} and $2 = 2^1$ so 2 MiB = $2 \times 2^{20} = 2^{21}$ bytes

c) 8 Kib

Notice how the unit is Kib (Kibibits) and not KiB (Kibibytes). One Kib is 2^(10) bits so 8 Kib = $2^3 \times 2^{10} = 2^{13}$ bits. Because there are $8 = 2^3$ bits in one byte, we divide our answer to get $8 \text{ Kib} = 2^{10}$ bytes

(Note that $8 \text{ Kib} = 1 \text{ KiB}$)

d) 24 GiB

We can factor $24 = 4 * 6 = 2^2 * 2 * 3 = 2^3 * 3$. One GiB $= 2^{30}$ so we can write 24 GiB = $2^3 * 3 \times 2^{30} = 3 \times 2^{33}$ bytes (alternatively, just 24×2^{30} bytes).

e) 19 TiB

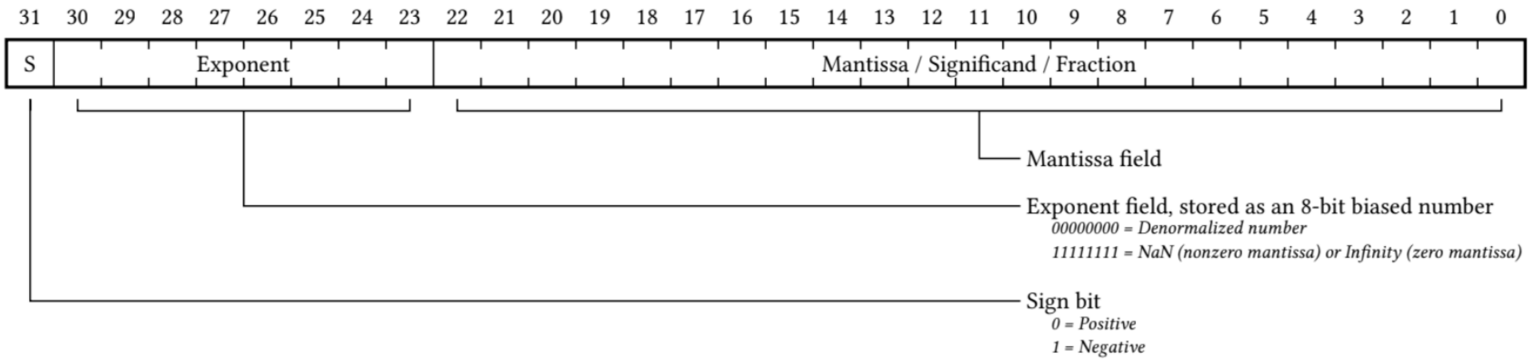
Note that 19 cannot be factored or easily representable in powers of 2. Following the same process as above, we can simplify to $19 \text{ TiB} = 19 \times 2^{40}$ bytes ≈ 19 trillion bytes

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.


31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
S
Exponent
Mantissa / Significand / Fraction
Mantissa field
Exponent field, stored as an 8-bit biased number
00000000 = Denormalized number
11111111 = NaN (nonzero mantissa) or Infinity (zero mantissa)
Sign bit
0 = Positive
1 = Negative

For normalized floats:

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

For denormalized floats:

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

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.

3 IEC Prefixes and Symbols

IEC Prefix multipliers are a set of standard units used to represent powers of 2 and are often used in discussion about caches and memory. The Base-2 (bi: "bee") IEC standard prefixes represent binary quantities officially up to exbi ("exbee"). Their comparison to SI units are shown below:

Prefix (Abbr)	**SI Size**
Kilo (k)	$10^3 = 1,000$
Mega (M)	$10^6 = 1,000,000$
Giga (G)	$10^9 = 1,000,000,000$
Tera (T)	$10^{12} = 1,000,000,000,000$
Peta (P)	$10^{15} = 1,000,000,000,000,000$
Exa (E)	$10^{18} = 1,000,000,000,000,000,000$
Zetta (Z)	$10^{21} = 1,000,000,000,000,000,000,000$
Yotta (Y)	$10^{24} = 1,000,000,000,000,000,000,000,000$

IEC (Abbr)	**IEC Factor**
Kibi (Ki)	$2^{10} = 1,024$
Mebi (Mi)	$2^{20} = 1,048,576$
Gibi (Gi)	$2^{30} = 1,073,741,824$
Tebi (Ti)	$2^{40} = 1,099,511,627,776$
Pebi (Pi)	$2^{50} = 1,125,899,906,842,624$
Exbi (Ei)	$2^{60} = 1,152,921,504,606,846,976$
Zebi (Zi)	$2^{70} = 1,180,591,620,717,411,303,424$
Yobi (Yi)	$2^{80} = 1,208,925,819,614,629,174,706,176$