## In-class Exercises and Homework, Sept. 12

Please upload your assignment as a PDF or Word file to Blackboard. This assignment is worth 20 points as indicated.

1. (2 points) The IEEE-754 floating-point standard specifies that single-precision floating-point numbers be represented in a base-2 format with a 24-bit significand:

$$\hat{x} = \pm a_0.a_1a_2\cdots a_{23} \times 2^e. \tag{1}$$

   We say that the nonzero quantity $\hat{x}$ is *normalized* if $a_0 = 1$ and $-126 \leq e \leq 127$. We define the normalized representation of zero as $0.00\ldots0 \times 2^0$.

   (a) What is the normalized IEEE single-precision representation of the number 5.5? Express your answer in the form of Eq. (1). In other words, specify the values of $a_1, \ldots, a_{23}$ and $e$.

   (b) If we change the significand of 5.5 by 1 ulp, by how much does the value of the floating-point representation change? Express your answer as a power of 2.

2. (3 points) Let $x = 1/3$.

   (a) Find the (infinite, eventually repeating) binary representation of $x$.

   (b) Find the IEEE-754 single-precision representation $\hat{x}$ of $x$ when rounding to nearest. That is, find the closest approximation to $x$ by an expression of the form (1).

   (c) What is the absolute error due to rounding? In other words, what is $|\hat{x} - x|$?

3. (1 point) Let $\hat{x} = 1.a_1a_2\cdots a_{23} \times 2^e$, where each $a_i$ is either 0 or 1. The Fortran language has a built-in function, spacing(x), whose value is the difference between $\hat{x}$ and the next largest floating-point number. (In other words, spacing(x) equals 1 ulp.) How does the spacing depend on $e$? Justify your answer.

4. (1 point) How many possible nonnegative normalized IEEE single-precision floating-point numbers are there? Justify your answer.

5. (2 points) Consider IEEE single-precision representations of the form (1).

   (a) Is 1,000,000.0 exactly representable in IEEE single precision? If not, what is the absolute error in its representation, assuming rounding to nearest?

   (b) What is the smallest positive integer $M$ that does *not* have an exact IEEE single-precision representation? Justify your answer.

6. (2 points) True or false: If $x$ has a terminating base-2 expansion, then $x$ has a terminating base-10 expansion. State a proof using a result given in lecture or find a counterexample.

7. (4 points) The machine epsilon for IEEE single-precision numbers is $2^{-23} \approx 1.2 \times 10^{-7}$. In this respect, single-precision IEEE floating-point is roughly equivalent to 7 decimal digits. On the other hand, 7 decimal digits do not suffice to represent an IEEE single-precision floating-point number uniquely. This exercise outlines a proof.

   Consider real numbers $x$ such that $10 \leq x < 16$, i.e., the interval $[10, 16)$.

(a) The numbers in this interval that are exactly representable in 7 decimal digits are $10.00000, 10.00001, 10.00002, \ldots, 15.99999$. How many numbers are in this set?

(b) The numbers in $[10, 16)$ that are exactly representable in IEEE single precision run from

$$10_{10} = 1.01\underbrace{00\cdots00}_{21 \text{ bits}} \times 2^3 \quad \text{to} \quad \left(16 - 2^{-20}\right)_{10} = 1.11\underbrace{11\cdots11}_{21 \text{ bits}} \times 2^3.$$

How many numbers are in this set?

(c) Explain how the pigeonhole principle implies that at least two different IEEE single-precision numbers have the same 7-digit decimal representation (and why that proves the result).

(d) Explain why 8 decimal digits also don't suffice to represent IEEE single-precision numbers uniquely.

8. (5 points) The floating-point represention in Eq. (1) can be stored in 32 bits, as follows: 1 bit for the sign, 23 bits for $a_1, a_2, \ldots, a_{23}$, and 8 bits for the exponent. There is no need to store $a_0$ since it is always 1 if $x \neq 0$.

However, Eq. (1) is not the only possible way to encode a floating-point number in 32 bits. Another approach, which we'll call the $(\pm, s, \ell)$ format, uses a sign bit, a significand $s$, and an integer "level" $\ell$, as follows. Let $s$ be expressed as a binary number such that $1 \leq s < 2$, and require that $-7 \leq \ell \leq 7$. We can express real numbers greater than or equal to 1 in magnitude in the following way. If $\ell = 0$, then the real number represented by $(\pm, s, 0)$ is $\pm s$. If $\ell = 1$, then $(\pm, s, 1)$ represents $\pm 2^s$; if $\ell = 2$, then $(\pm, s, 2)$ represents $\pm 2^{2^s}$, $\ell = 3$ represents $\pm 2^{2^{2^s}}$, and so on. Numbers between 0 and 1 in magnitude are represented by negative values of $\ell$: $(\pm, s, -1)$ represents $\pm 2^{-s}$; $(\pm, s, -2)$ represents $\pm 2^{-2^s}$, etc. The $(\pm, s, \ell)$ format is just as compact as the IEEE format: $\ell$ requires 4 bits, the sign takes 1 bit, and $s$ can occupy the remaining bits of a 32-bit word.

(a) What real number is represented by $(+, 1.5, 0)$? $(-, 1.5, 1)$? $(+, 1.5, 2)$? $(-, 1.5, -1)$?

(b) What is the set of representable values for $\ell = 0, \pm 1, \pm 2, \pm 3, \pm 4$?

(c) The number $G = 10^{100}$ is called a *googol*; $10^G$ is a *googolplex*. Express the largest representable value of $(+, s, 4)$ as an approximate power of $G$.

(d) Is the largest representable value of $(+, s, 5)$ greater or less than a googolplex? Explain.

(e) Invent your own terminology as necessary to describe the largest representable values of $(+, s, 6)$ and $(+, s, 7)$.

Although overflow and underflow are unlikely to occur in the $(\pm, s, \ell)$ format, there are no simple algorithms for addition and multiplication, which is why we use formats like Eq. (1).