CS 61C
Summer 2020

Floating Point

Discussion 3: June 29, 2020

1 Pre-Check

This section is designed as a conceptual check for you to determine if you conceptually understand and have any misconceptions about this topic. Please answer true/false to the following questions, and include an explanation:

1.1 True or False. The goals of floating point are to have a large range of values, a low amount of precision, and real arithmetic results

F

1.2 True or False. The distance between floating point numbers increase as the absolute value of the numbers increase.

T

1.3 True or False. Floating Point addition is associative.

F

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.
- The *significand* or *mantissa* is akin to unsigned integers, but used to store a fraction instead of an integer.

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

1	8	23
Sign	Exponent	Mantissa/Significand/Fraction

For normalized floats:
Value $= (-1)^{Sign} * 2^{Exp+Bias} * 1.\mathbf{significand}_2$
For denormalized floats:
Value $= (-1)^{Sign} * 2^{Exp+Bias+1} * 0.\mathbf{significand}_2$

Exponent	**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.

2.1 How many zeroes can be represented using a float?

2, +-0

2.2 What is the largest finite positive value that can be stored using a single precision float?

sign = 0, exponent = 255, bias = -127, significand = 1.0, so 1 * 2 ^ 128

wrong, finite value, not infinity

2.3 What is the smallest positive value that can be stored using a single precision float?

sign = 0, exponent = 0, bias = -127, significand = 0.0000000…0001, so 1 * 2 ^ (-149)

2.4 What is the smallest positive normalized value that can be stored using a single precision float?

sign = 0, exponent = 1, bias = -127, significand = 1.0, so 1 * 2 ^ (-126)

2.5 Cover the following single-precision floating point numbers from binary to decimal or from decimal to binary. You may leave your answer as an expression.

100111.1001= 1.001111001 * 2 ^ 5

- 0x00000000 0
- 8.25 1000.01= 1.00001 * 2 ^ 3 0 10000010 00001000…
- 0x00000F00 1.9375 * 2^-138 0 00000000 000 0000 0000 1111 0000 0000
- 39.5625 0 10000100 00111100100000…
- 0xFF94BEEF NaN
- -∞ 1 11111111 00000000000000000000000

3 More Floating Point Representation

Not every number can be represented perfectly using floating point. For example, $\frac{1}{3}$ can only be approximated and thus must be rounded in any attempt to represent it. For this question, we will only look at positive numbers.

3.1 What is the next smallest number larger than 2 that can be represented completely?

3.2 What is the next smallest number larger than 4 that can be represented completely?

3.3 Define stepsize to be the distance between some value x and the smallest value larger than x that can be completely represented. What is the step size for 2? 4?

2^-22, 2^-21

3.4 Now let's see if we can generalize the stepsize for normalized numbers (we can do so for denorms as well, but we won't in this question). If we are given a normalized number that is not the largest representable normalized number with exponent value x and with significand value y, what is the stepsize at that value? Hint: There are 23 significand bits.

2^(x-150)

3.5 Now let's apply this technique. What is the largest odd number that we can represent? Part 4 should be very useful in finding this answer.

0b0000 0000 1111 1111 1111 1111 1111 1111
16777215