# CS 61C Fall 2023

# Floating Point

Discussion 3

## 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 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.

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.

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

1.4 Floating Point addition is associative.

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

| **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 Convert the following single-precision floating point numbers from hexadecimal to decimal or from decimal to hexadecimal. You may leave your answer as an expression.

- 0x00000000
- 8.25
- 0x00000F00
- 39.5625
- 0xFF94BEEF
- -$\infty$
- 1/3

# 3 More Floating Point Representation

As we saw above, not every number can be represented perfectly using floating point. 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 What is the largest odd number that we can represent? Hint: At what power can we only represent even numbers?

# 4 Linked List

Suppose we've defined a linked list **struct** as follows. Assume `*lst` points to the first element of the list, or is `NULL` if the list is empty.

```
struct ll_node {
    int first;
    struct ll_node* rest;
}
```

4.1 Implement `prepend`, which adds one new `value` to the front of the linked list. Hint: why use `ll_node** lst` instead of `ll_node* lst`?

```
void prepend(struct ll_node** lst, int value)
```

4.2 Implement `free_ll`, which frees all the memory consumed by the linked list.

```
void free_ll(struct ll_node** lst)
```