CS 61C C, Floating Point

Summer 2024 Discussion 3

1 Review: C Memory

1.1 For each part, choose one or more of the following memory segments where the data could be located: **code, static, heap, stack**.

(a) Static variables

(b) Local variables

(c) Global variables

(d) Constants (constant variables or values)

(e) Functions (i.e. Machine Instructions)

(f) Result of Dynamic Memory Allocation(`malloc` or `calloc`)

(g) String Literals

Compare these different ways of storing ”DEADBEEF”. Assume that each program is run on the same machine and architecture.

```
char arr[] = "DEADBEEF"
```

```
int arr[2];
arr[0] = 0xDEADBEEF;
arr[1] = 0x00000000;    //null terminator in hex is 0x00
```

1.2 Do these two C programs store ”DEADBEEF” in memory the same way?

You take a look at the ASCII table and translate the string ”DEADBEEF” into bytes.

```
int arr[2];
//storing "DEAD" in ascending order in arr[0]
arr[0] = 0x44454144

//storing "BEEF" in ascending order in arr[1]
arr[1] = 0x42454546
```

1.3 Does this C program store ”DEADBEEF” in memory the same way as storing it as a string?

2 Pre-Check: Floating Point

2.1 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.

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

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

2.4 Floating Point addition is associative.

3 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.

3.1 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
- -∞
- 1/3

4 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.

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

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

4.3 What is the largest odd number that we can represent? Hint: At what power can we only represent even numbers?