Homework 1

Integer and Floating Point Number Representations

Integer

Problem 1

Suppose you have a 3 GHz x64 core and you can execute three integer operations (additions or subtractions) every cycle. How long will the following loop run?

```
int i;  /* 32 bit integer */
int s;
for (i = -1 ; i < 0; i--) {
      s += i;
}
```

Problem 2

What logical operator is equivalent to multiplying two single-bit integers together? How would you multiply two two-bit integers together using logic and addition? How would you generalize to multiplying two n-bit integers?

Problem 3

Some instruction sets, including x64, provide an integer representation in addition to two's complement. This representation is called Binary Coded Decimal (BCD). In BCD, a decimal digit (0,1,2,3,4,5,6,7,8,9) is encoded into a group of 4 bits using 0000 through 1011. How many unique numbers can be represented in a 64 bit BCD quantity? Why might one use BCD to represent prices like $10.99?

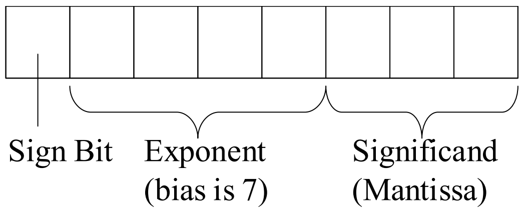
Problem 4

Many instruction sets have instructions where when you multiply two k bit numbers the result is stored as two k bit numbers. Why? Similarly, many have instructions where if you divide two k bit numbers, the result is stored as two k bit numbers. Why?

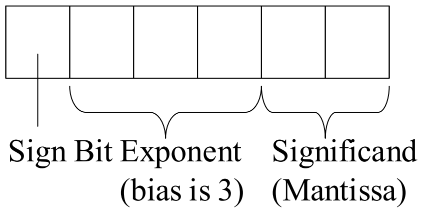
Floating Point

Consider the following two small floating point formats based on the IEEE standard:

- Little Format



- Tiny Format



Except for the sizes of these formats, the rules are those of the IEEE standard.

Problem 1

For both formats, determine the following values (in decimal)

1. Largest positive finite number
2. Positive normalized number closest to zero
3. Largest negative denormalized number
4. Negative denormalized number closest to zero

Problem 2

Encode the following values in the 8 bit Little Format: ¾, -13/16, 44, and –104, show each in binary and hexadecimal.

Problem 3

Determine the values corresponding to the following Little Format bit patterns. The leftmost bit is the most significant

1. 10101011
2. 01111000
3. 10110101
4. 01011111
5. 11000101

Problem 4

Convert the following 8 bit Little Format numbers into 6 bit Tiny Format numbers. Overflow should yield +/- infinity, underflow should yield +/- 0.0, and rounding should follow the "round-to-nearest-even" tie-breaking rule.

1. 00010010
2. 11101011

3. 10100011
4. 11001110
5. 00110101