Homework 1

Integer and Floating Point Number Representations

Integer

In the Data Lab, you are using the x86 processor in 32 bit mode. Part of what this means is that the "natural" width of an integer is 32 bits---you are employing 32 bit two's complement integers in your code. The same processor (as well as processors developed by other vendors) can also run in 64 bit mode, where the natural width is 64 bits. For this section of the homework, assume such a 64 bit two's complement format.

Problem 1

Suppose you have a 1 GHz processor and you can execute three 64 bit integer additions (or subtractions) every cycle. How long will the following loop run?

```
long long int i;  /* 64 bit integer */
long long int s;
for (i = 0 ; i >= 0; i++) {
     s += i;
}
```

Problem 2

A Full Adder (FA) logic block takes three input bits and provides two output bits. The input bits are summed to produce a two bit output. Write a table showing what each combination of input bits produces at the output. Draw a picture of how you might connect FAs together to create a (slow) 64 bit adder. If you're interested in how a *fast* adder works, do a google search for the term "carry lookahead adder".

Problem 3

Some instruction sets, including x86, 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. How many unique numbers can be represented in a 64 bit BCD quantity? Why might one use BCD?

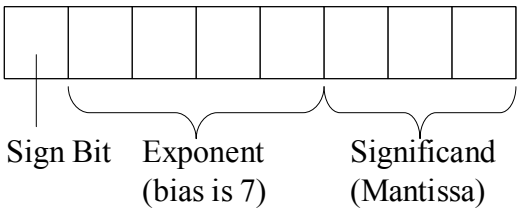
Problem 4

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?

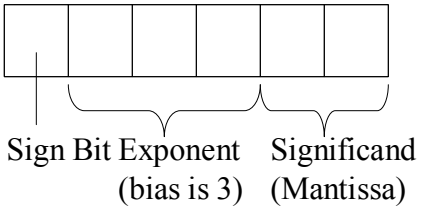
Floating Point

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