CS/ECE 347, Spring 1996
Recitation Assignment 7: Computer Arithmetic

Assigned: Tue., Mar. 12
Due: Mon., Mar. 18

Problem 1: Two's Complement Multiplication

There are two integer multiply instructions on a MIPS machine: `mult`, for signed arguments, and `multu` for unsigned. Both deposit the 64-bit product into two special registers `HI` and `LO`. These values can then be retrieved with assembly instructions `mfhi` and `mflo`. In compiling C code involving multiplies, the compiler always uses the `mult` instruction (whether or not the arguments are signed) followed by an `mflo`. Thus the C function:

```
int smult(int x, int y)
{
  return x*y;
}
```

gets compiled into the code:

```
smult:
        mult    $4,$5
        mflo    $2
        j       $31
```

Suppose we want to be more cautious about multiplication. That is, when the product of two signed numbers is too far in the positive direction to represent as a signed number, we would return the value `POS_INF` (defined below) instead. Similarly, when the product is too far in the negative direction, we would return `NEG_INF`.

```
#define NEG_INF (1<<31)
#define POS_INF (~NEG_INF)
/* Modified multiplication that detects overflow
  and returns limiting values when this occurs */
int lim_mult(int x, int y);
```

1. What values of the `HI` and `LO` registers would indicate that one of the forms of overflow had occurred?
2. Write an assembly code routine implementing `lim_mult`. The best way to do this is to write as much as possible in C, compile to assembly, and hand modify this assembly. If done cleverly, you only need to modify a single line!

Prolem 2: Floating Point Representation

For the following base ten numbers, give their IEEE 754 Floating-point representations. For each number, first show it in normalized binary form (i.e, as number of the form $\texttt{1.xxx}_2 \times 2^{exp_{10}}$), and then show the binary representation.

Indicate the binary representation in the boxes provided. The entries for trailing 0's in the significand may be left blank. You may find the following table useful:

Negative Powers of 2	
2^{-1}	0.5
2^{-2}	0.25
2^{-3}	0.125
2^{-4}	0.0625
2^{-5}	0.03125
2^{-6}	0.015625

1. **Number:** `0.75`

   Normalized Binary:

   Binary Representation:

<table><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>s</td><td colspan="8">exponent</td><td colspan="23">significand</td></tr></table>

2. **Number:** `−1.875e−1`

   Normalized Binary:

   Binary Representation:

<table><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>s</td><td colspan="8">exponent</td><td colspan="23">significand</td></tr></table>

3. **Number:** `−30429.0`

   Normalized:

   Binary Representation:

<table><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>s</td><td colspan="8">exponent</td><td colspan="23">significand</td></tr></table>

4. **Number:** `513.375`

   Normalized Binary:

   Binary Representation:

<table><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>s</td><td colspan="8">exponent</td><td colspan="23">significand</td></tr></table>