Lab 2

Floating Point and Number Rep

Background

Goals

This lab should help you examine how computers store integer and floating point values.

Reading

- 3.4, 3.5, 3.6, 3.8, 3.10 (3rd) on floating point

Info

Recall that the single precision floating point number is stored as:

```
SEEE EEEE EIII IIII IIII IIII IIII IIII
```

where:

S is the sign bit, 0 for positive, 1 for negative E is the exponent, bias 127 I is the significant, with an implicit 1

For example, the floating point representation of 1.0 would be 0x3F800000. Verify to yourself that this is correct.

Exercises

Setup

Exercise 1: Integers

Find the shortest sequence of MIPS instructions to determine if there is a carry out from the addition of two registers, say $t3 and $t4. Place a 0 or 1

in register $t2 if the carry out is 0 or 1, respectively. (This can be done in just two instructions). Verify that your code works for the following values:

Operand	Operand	Carry out?
0x7fffffff	0x80000000	no
0xffffffff	0	no
0xffffffff	1	yes

Exercise 2: Floating Point

Find a positive floating point value x, for which x+1.0=x. Verify your result in a MIPS assembly language program, and determine the stored exponent and fraction for your x value (either on the computer or on paper).

Note: The provided MIPS program p2.s will allow you to experiment with adding floating point values. It leaves the output in $f12 and also $s0, so you can examine the hex representation of the floating point value by printing out $s0.

Exercise 3: Floating Point

Next, find the smallest positive floating point value x for which x+1.0=x. Again, determine the stored exponent and fraction for x.

Exercise 4: Floating Point Associativity

Finally, using what you have learned from the last two parts, determine a set of positive floating point numbers such that adding these numbers in a different order can yield a different value. You can do this using only three numbers. (Hint: Experiment with adding up different amounts of the x value you determined in part 3, and the value 1.0).

This shows that for three floating point numbers a, b, and c, a+b+c does not necessarily equal c+b+a.

If time permits, you should write a program to add these three values in different orders. It should be a straightforward modification of the program from part 2-3.