CS 590: Topics in Computer Science
Assignment 10: Floating Point and Number Representation

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

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.

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 `pfloat2.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`.

```
…
num1: .word 0x3F800000
num2: .float 1.0
result: .word 0
string: .asciiz "\n"

.text
main:
la $t0, num1
lwc1 $f2, 0($t0)
lwc1 $f4, 4($t0)

…
# Do the actual addition

add.s $f12, $f2, $f4

# Transfer the value from the floating point reg to the integer reg

swc1 $f12, 8($t0)
lw $s0, 8($t0)

# At this point, $f12 holds the sum, and $s0 holds the sum which can
# be read in hexadecimal
…
```

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`.

In Exc 2:

x => $f2 (0x3F800000 = 1.0)

0.1=> $f4 (0x3F800000 = 1.0)

Result => $f12, $s0 (0x40000000 = 2.0)

Let us look at the content of registers through video 10-3.mp4

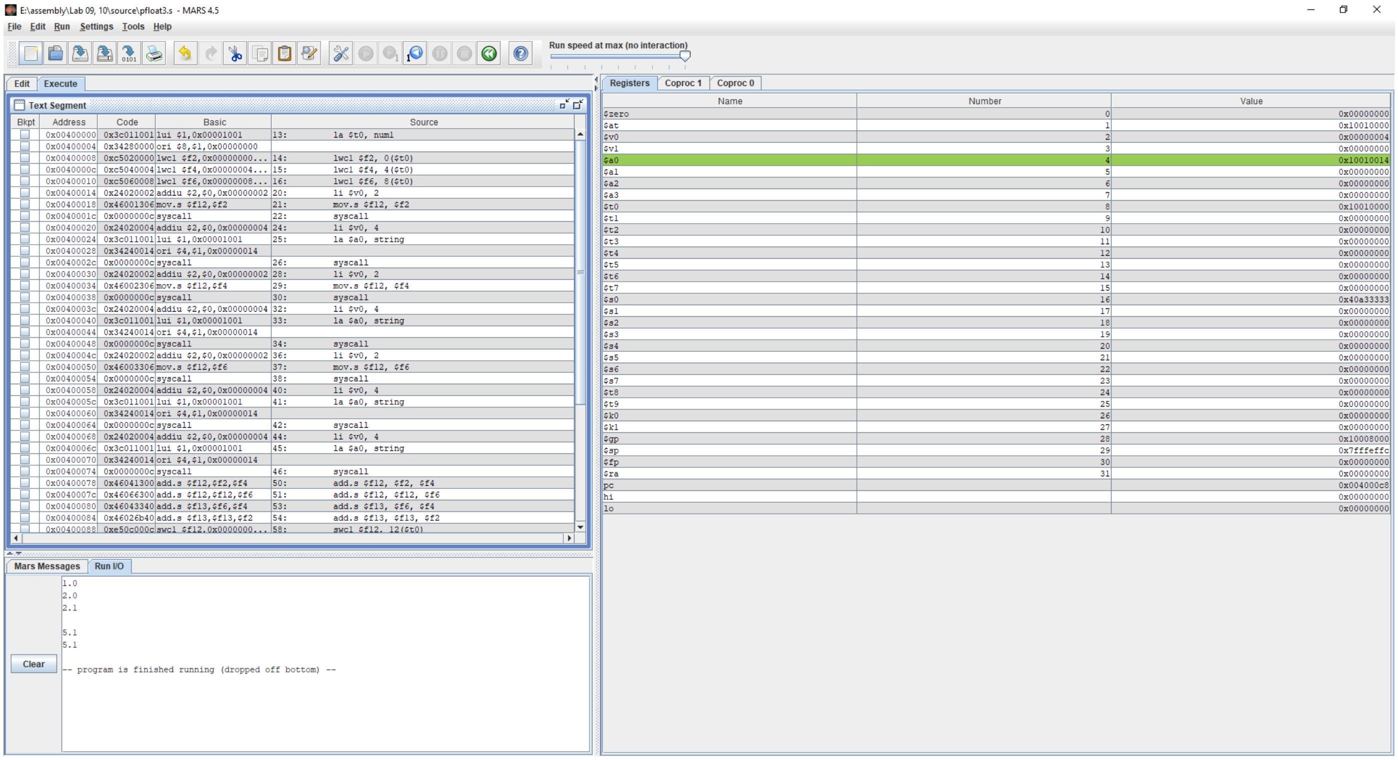
Exercise 4: Floating Point Associativity

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`.

Write a program to add these three values in different orders. It should be a

straightforward modification of the program from part 2-3.



Questions/Tasks:

1. How come that Out-of-Order execution is an indispensable performance feature? Explain.
2. Explain how Meltdown exploits the side effects of out-of-order execution.
3. Explain how come that the Meltdown attack is independent of the operating system, and it does not rely on any software vulnerabilities.
4. What's the main difference between Spectre and Meltdown exploits? Explain.