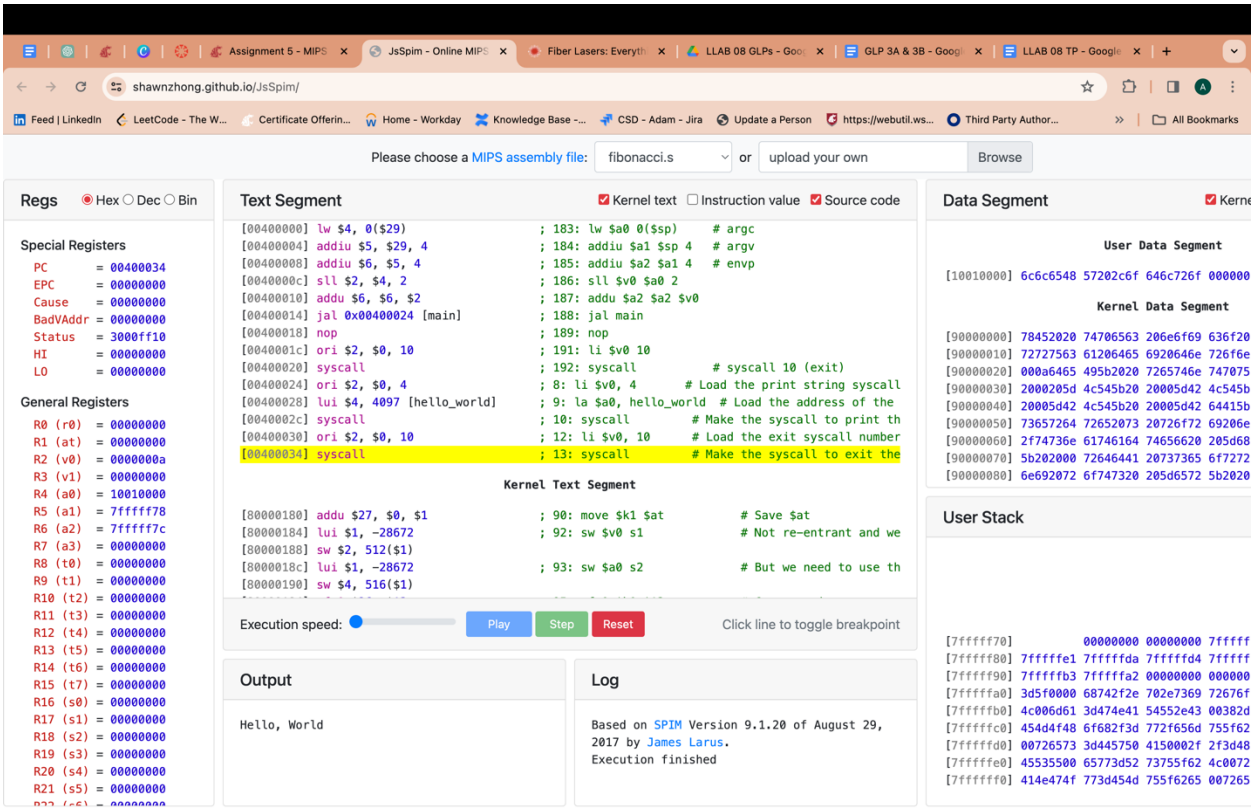
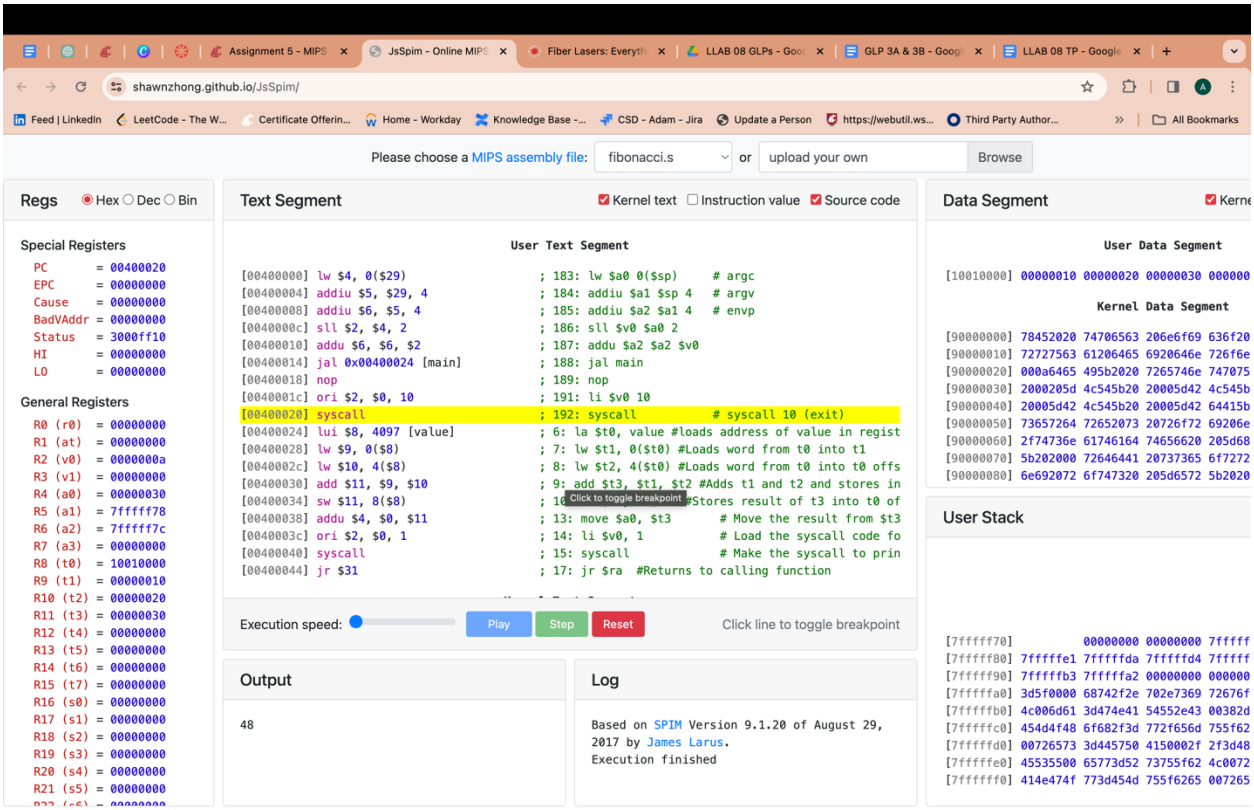
HW 5 (MIPS Assembly) Report

Task 1:



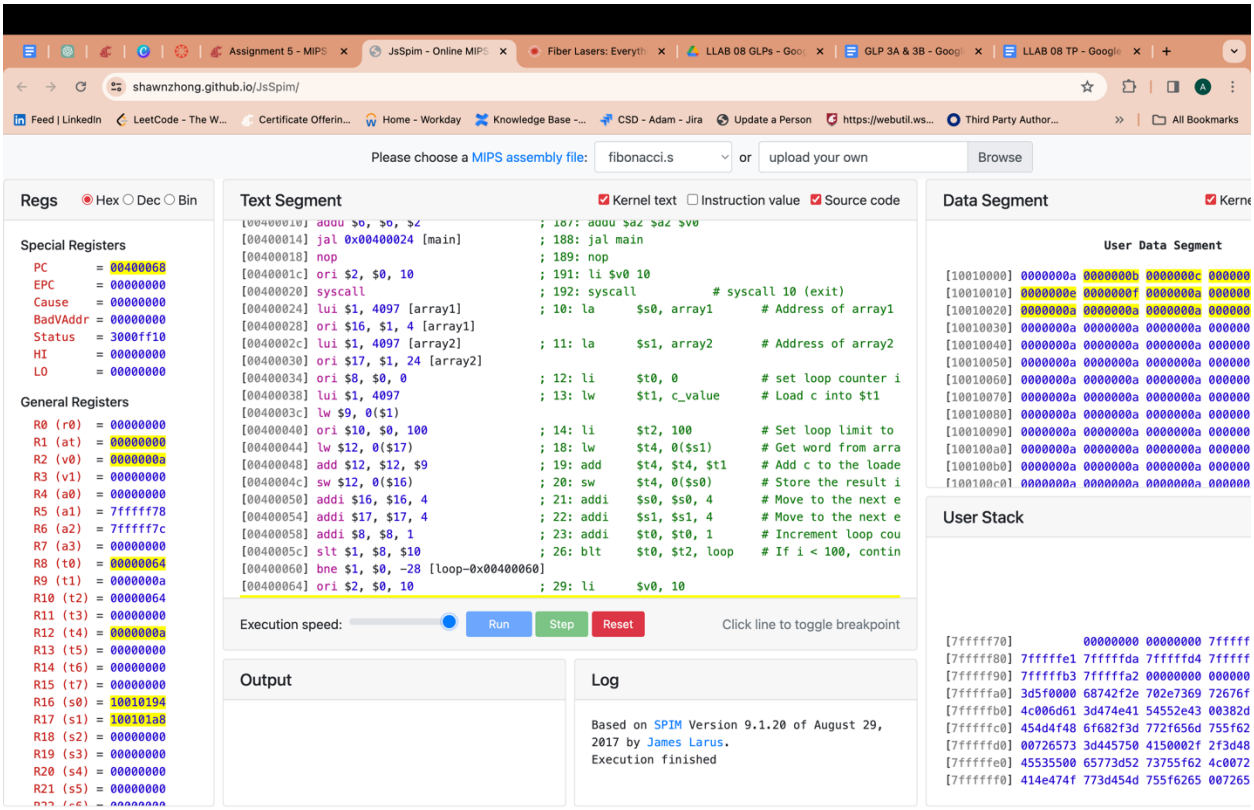
For task 1 I downloaded a QtSpim version compatible with Mac OSX. This version did not work particularly well so I decided to use the online simulator. For task 2 I made a practice “Hello World” in MIPS. The results of the successful operation of the JSPIM simulator and my “Hello World” program can be seen in the screenshot above as well as my .asm file.

Task 2:



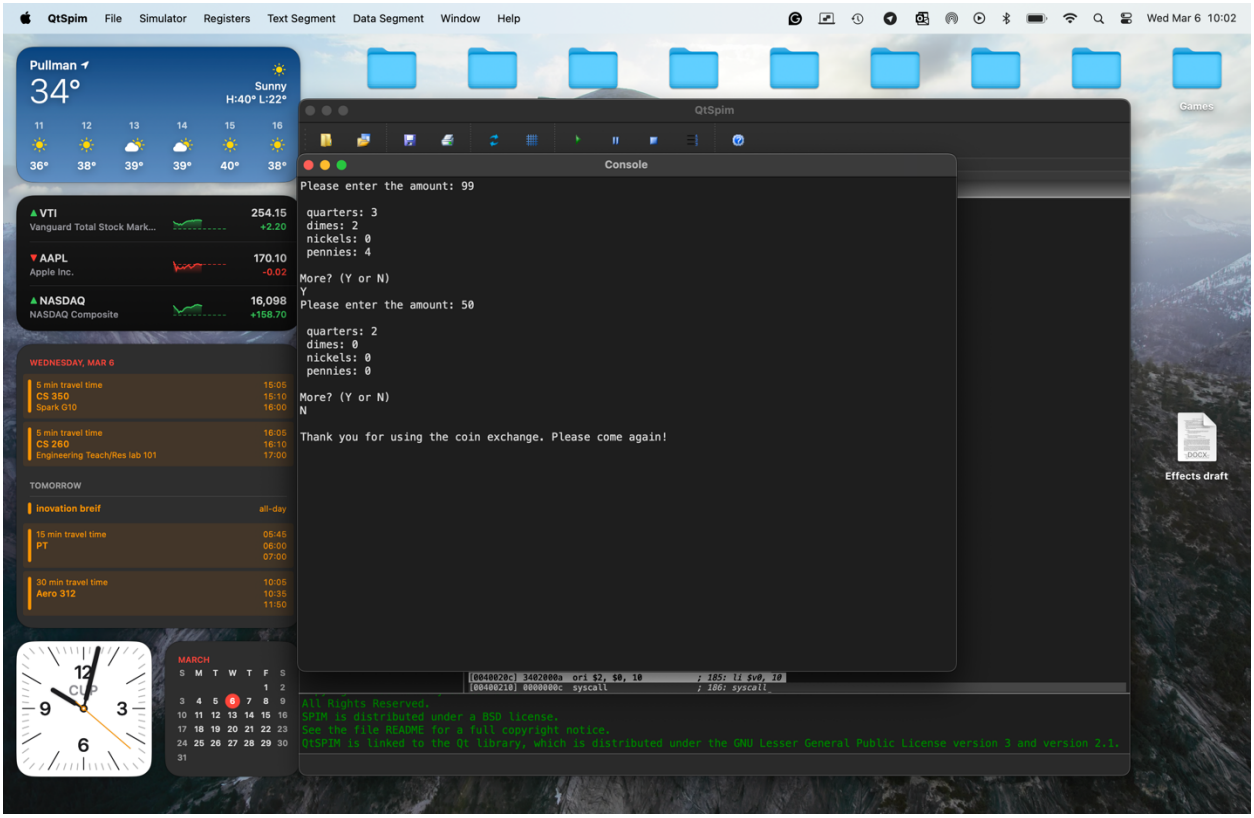
For task two I utilized MIPS code to add two numbers in memory and store the result back into memory. I added extra MIPS instructions to print the result from memory to the screen to check my work.

Task 3:



For task 3 I wrote an assembly program to add a constant to every element of an array. I translated the for loop provided into MIPS with the base array address of array A in $s0, the base address of array B in $s1, I associated with $t0, and C associated with $t1. I chose a random constant value for c of 10. Results can be seen operating successfully in the JSPIM screenshot above as well as in my included .asm file.

Task 4:



For task four we had to translate the Python code for a very simple teller into MIPS assembly code. I did this by referencing the Python design and directly translating parts into labels which then fit together nicely. The results of input and correct output as well as testing that the upper bound of 99 works can be seen in the screenshot above.

Experience:

My experience with QTSPIM and JsSpim (I used both) was very mixed. While JsSpim worked very well, it did not display text next to input when getting user input making debugging annoying and difficult in certain circumstances. Whereas QTSPIM worked very well for debugging and execution of my code, it kept crashing as the Mac OSX version was not very stable. Overall, it was an interesting yet frustrating experience as I wish there were better simulation tools at my disposal.

From this assignment though I have learned about the intricacies of assembly code and how programming languages work at a lower level. I especially appreciated the work spent on translating Python code and appreciated example code from task 2. I also gained a better perspective on how code works closer to the processor and how instruction sets interact with processor registers.