Program 2 Instruction

Overview
In program 1, you have done a assembler for MIPS Assembly Language, which you take an MIPS code input and convert it to its corresponding machine language (binary) code. This program is called a MIPS Simulator, in which you will take a machine code input and simulate executing the code in high level language. The input file of this program is the same as the output file of program 1, and should actually run it. For example, if an adder MIPS code is given to you, your first program should translate it into a binary code file. If you input that into your second program, it should prompt the user to enter two integers and output the correct answer.

Logic
The logic behind this program is fairly easy. You read in the file line by line, just like the first program, and you do not even need to tokenize the string! You check what type of instruction this line is like you did before, but with binary code as input. Then you execute the correct instruction.

The challenge behind this project is really just how you keep and manage registers. Since you do not have registers in high level language, you will need to declare variables to store them. You will need to memorize which registers your variables are representing and manage the data in it when instructions are executed.

To summarize the logic:
1. You read in a single line of machine code.
2. You check what kind of instruction it is.
3. You check which instruction it is, and which registers it uses.
4. You find the corresponding registers (the ones you declared since you cannot see the real register).
5. You execute the code.

Unlike last time, I will not limit what you write by saying you have to turn in which files. There are a lot of confusions caused by doing that, and many students did not follow it anyways. Therefore, you decide how you want to write it.

Things to mind
1. In this program, you will need to support an additional instruction names “syscall”. This instruction is used to do system services, for example, I/O is done by this. More details can be found here: http://courses.missouristate.edu/kenvollmar/mars/help/syscallhelp.html
2. Similar to the first program, your program 2 SHOULD BE ABLE TO RUN ON LINUX SYSTEM. Please test your program on a linux environment before you turn in.

3. A discussion section will be set up on blackboard, on which you can post your questions. I will be checking it daily, and answer questions. If you find someone else is having problems you know how to solve, please help me out by answering.

Grading

Instruction classification - 10%
Register Simulation - 20%
Execution - 50%
Coding style/Comments - 10%
Report - 10%

NO GITHUB CODES. If you are caught copy and pasting from github, you will get a zero on the program. Changing names of functions and variables, and switching order of functions are not going to work :)