# CprE 381 - Computer Architecture and Assembly Level Programming, Spring 2017 Lab-4

Last week, we looked into assembly programs and saw some very basic instructions like divide and shift arithmetic. In this lab, you will write your own MIPS assembly programs and run them on SPIM.

**[For Windows machines on Coover]** To run Qtspim, go to [Start > QtSpim] and click QtSpim or search for QtSpim.

**Part I. SPIM:**QtSpim is a software that will help you to simulate the execution of MIPS assembly programs. It does a context and syntax check while loading an assembly program. In addition, it adds in necessary overhead instructions as needed, and updates register and memory content as each instruction is executed.  
  
When you open QtSpim, Two windows will open: main window and console. The main window is divided into different sections:

**1.** The **Register** tabs display the content of all registers.   
**2.** Buttons across the top are used to load and run a simulation   
**3.** The **Text** tab displays the MIPS instructions loaded into memory to be executed. (From left-to-right, the memory address of an instruction, the contents of the address in hex, the actual MIPS instructions – where register numbers are used, the MIPS assembly that you wrote, and any comments you made in your code are displayed.)   
**4.** The Data tab displays memory addresses and their values in the data and stack segments of the memory.   
**5.** The Information Console lists the actions performed by the simulator and some error/warning messages, if any.  
**6.** The separate console shows the output string from a program, if any.

**To run an assembly program in QtSpim:**   
**1.** Use a text editor to create your assembly program **xyz.s**, for example**.**   
**2.** Click on the “Load File” button (or [File > Load File]) and open **xyz.s**   
**3.** You can then run the program by simply pressing the “Run/Continue” (play) button – all instructions will be executed, and the final contents of memory and the register file will be reflected in the QtSpim window. For debugging, you can single step through the program using the “Single Step” button (or [Simulator > Single Step] or F10 key) to execute one instruction at a time. You can also use breakpoints for debugging purposes.   
**4.** **Load** and **single step** through the program **lab4a.s** given to you in the zip file. Familiarize yourself with SPIM, refer to the URL below to learn more about the simulator. *http://spimsimulator.sourceforge.net/further.html*  
**5.** To restart a program select [Simulator > Clear Registers].  
**6.** Use [File > Reinitialize and Load File] when you load a new program.

**Note:** when you reach the last instruction and try to execute the program further, you may get an error. Do not panic. This happens because the simulator doesn’t know what to execute next. If you want to re-run the program, click on “Clear Registers” and then click on “Run/Continue”.

**Part II. MIPS Assembly:**

**1.** Open the program **lab4a.s** in SPIM, and single step through it. When the program terminates, take a screenshot of the main window (for both [Int Regs] and [Text] tabs; the registers involved in the program must be visible clearly in your screenshot) and the console. Check the final value in $v0 and explain why. Add detailed comments to the assembly file to explain what the code does. Submit the commented assembly file as well. (see below for SPIM system call codes)

**2.** Open the program **lab4b.s** in SPIM, and single step through it. When the program terminates, take a screen-shot of the main window (for both [Int Regs] and [Text] tabs; the registers involved in the program must be visible clearly in your screenshot) and the console. Also, take a screenshot of [Data] tab, and locate the data that has been written by the program (i.e., **sw $t0, y**). Add detailed comments to the assembly file to explain what the code does. Submit the commented assembly file as well.  
 **3.** Write an Assembly program that uses a procedure call to compute the function f = (a0 + a1) \* (a2 + a3), using the registers **$a0, . . ., $a3** for the arguments and returning the result into **$v0**. Refer to the example given on page 98-99 of your textbook, store your result into $**s0** first, before finally putting it into $**v0**. Submit your code and a screen-shot of your output. Also investigate the **User Stack,** report out the value at the top of the stack before the procedure call is made, during the call, and after the procedure has completed.

**Part III. Machine Code:**  
**1.** Let us see what machine code looks like after the compiler has generated the binary file. Use the same code you wrote in Part II.3. The second column from the left in the text section of QtSpim gives you the encodings. Refer to appendix A.10 of your textbook for all the instruction encodings. Locate the procedure in the code, verify them with the encodings from the book and submit a screenshot of the procedure.

**Submission**All submissions are through blackboard. If you are submitting multiple files, give the files relevant names and put all files into a single zip file for uploading.

**Appendix. SPIM System Call Code**

