ECE369A Computer Organization
LAB2

Objective

- Get familiar with MIPS assembly simulator, QtSpim
- To download the tool go to https://sourceforge.net/projects/spimsimulator/files/

Reading

- Recommended Reading Assignment: zyBooks, Appendix-A (7.6 and 7.9)
- Read the "QtSPIM_examples.pdf" under "Resources" folder
- Read the QtSPIM overview (next page)

Description

- There are 4 exercises (**sum of array, debug, sort, recursion**) to complete under "Resources\ece369_lab4_files" folder
    - Instructions are included in the source files
- "Resources/examples" folder includes various source codes for your review
- **Submit source files for sum of array, debug, sort and recursion (.s only) on D2L to the designated dropbox by the deadline**.
- Include the overall percent effort of each team member in each file.
- Demonstration is not required for this lab
- This lab is worth 50pts
- Penalty Conditions
    - Percent effort not reported (20% penalty)
    - Late submission (20% per day)
    - Submitting files in a folder or in compressed form (zip/tar). (25% penalty)
    - Changing the file name or extension. (25% penalty)

In the past we experimented with another MIPS simulator called **MARS.**
https://courses.missouristate.edu/KenVollmar/mars/download.htm

You should try both QtSPIM and MARS and choose the one that works best for you with its features. QtSPIM had a revision in 2020 but MARS has convenient debugging features.

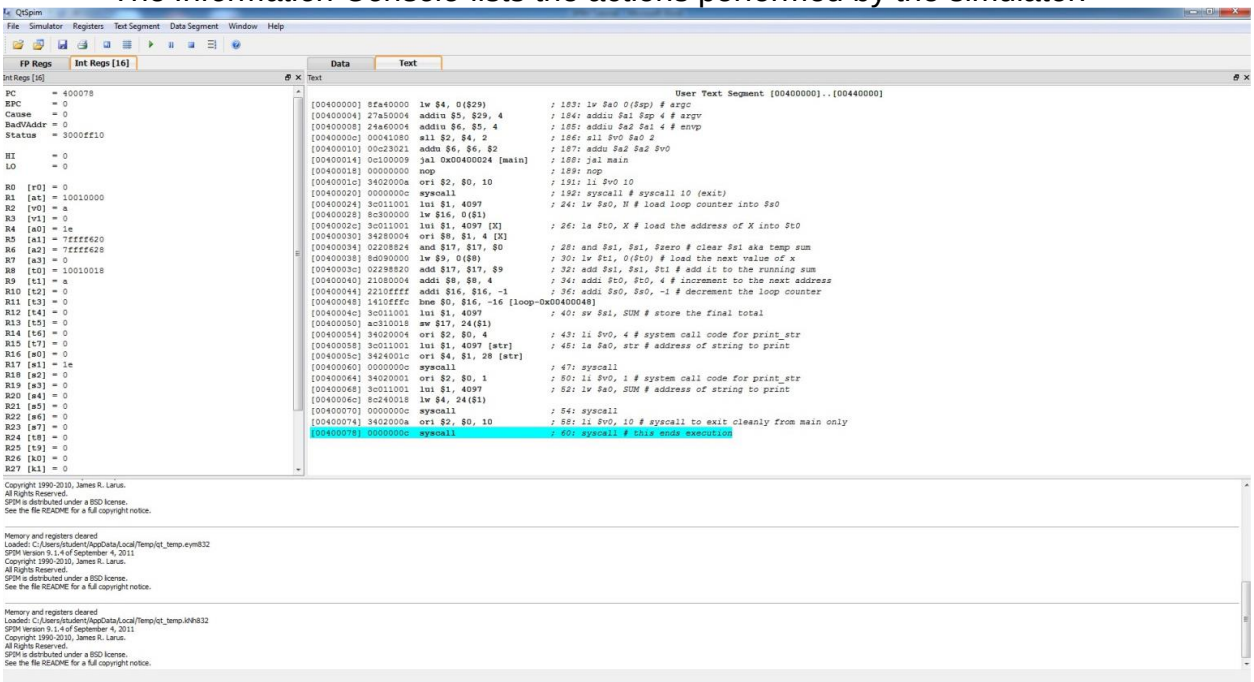
QtSpim Overview

QtSpim is software that will help you to simulate the execution of MIPS assembly programs. The most up-to-date version of the SPIM simulator, called “QtSPIM,” is maintained by James Larus, formerly of the University of Wisconsin at Madison. You may refer to this link: http://pages.cs.wisc.edu/~larus/spim.html#qtspim

To download the tool go to https://sourceforge.net/projects/spimsimulator/files/
You will see the choice of a number of downloads for Windows, Mac and Linux versions.

QtSpim 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, A window will open as shown below. The window is divided into different sections:

- The *Register* tabs display the content of all registers.
- Buttons across the top are used to load and run a simulation
- 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.)
- The *Data* tab displays memory addresses and their values in the data and stack segments of the memory.
- The *Information Console* lists the actions performed by the simulator.

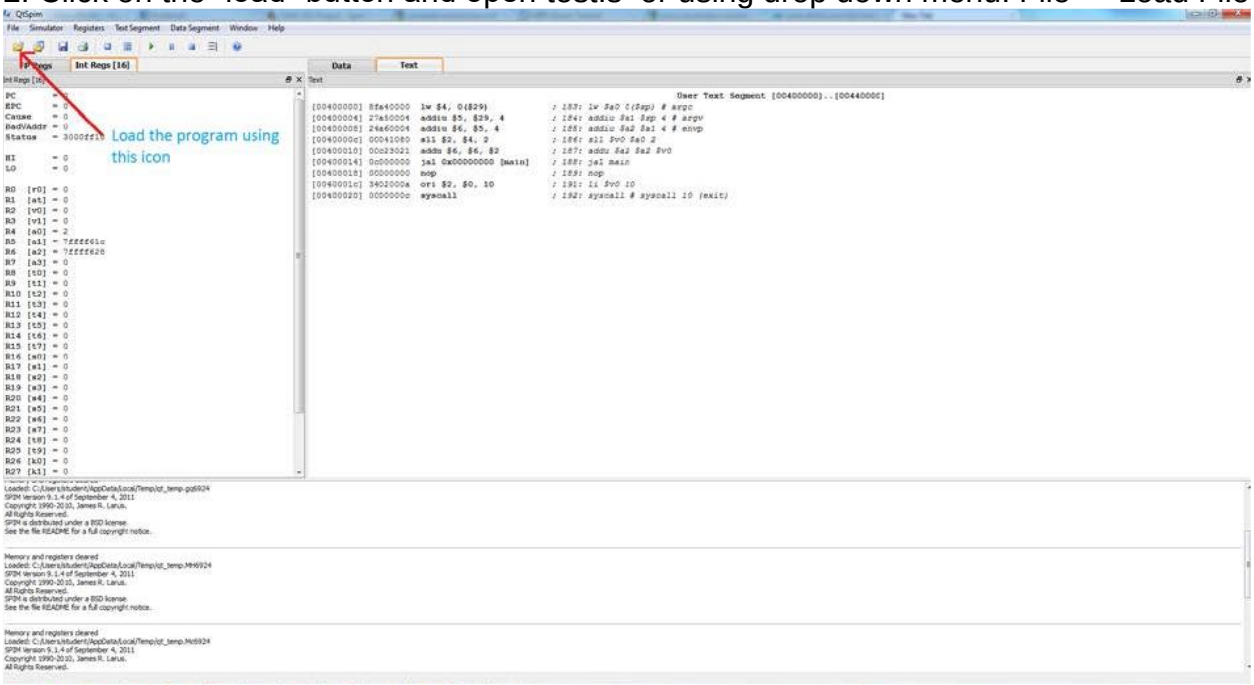


To run the program in QtSpim:

1. Use a text editor to create your program test.s and copy the following code.

```
.text
main: lw $t0,data1
lw $t1,data2
lw $t2,data3
mul $t3,$t0,$t1
mul $t3,$t3,$t2
move $a0,$t3
li $v0,1
syscall
li $v0,10
syscall
.data
data1: .word 17
data2: .word 29
data3: .word 56
```

2. Click on the “load” button and open test.s or using drop down menu: File -> Load File

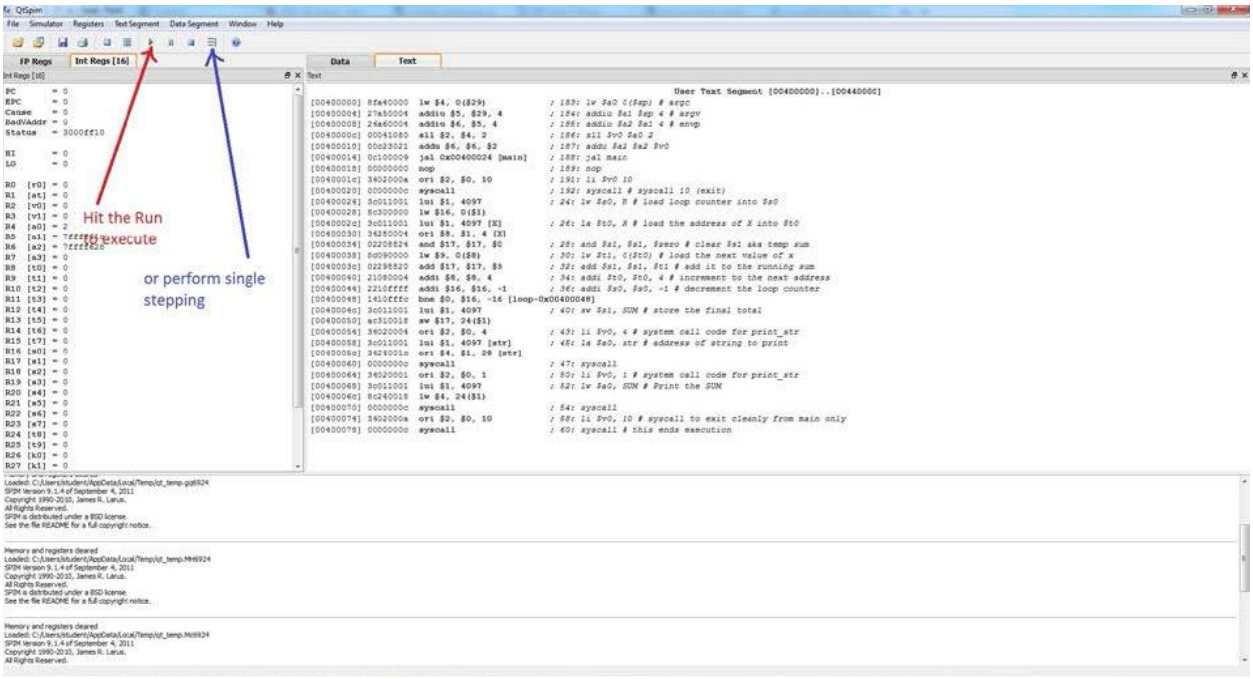


3. You can then run the program by simply pressing the “run” (play) button – all instructions will be executed, and the final contents of memory and the register file will be reflected in the QtSpim window.

Using drop down menu:

Simulator -> Run/Continue or F5 shortcut key

Simulator -> Single Step or F10 shortcut key



Debugging

Suppose your program does not do what you expect. What can you do? QtSpim has two features that help debug your program. The first, and perhaps the most useful, is single-stepping, which allows you to run your program an instruction at a time. The single stepping icon can be found in the toolbar. Every time you do single stepping, QtSpim will execute one instruction and update its display, so that you can see what the instruction changed in the registers or memory.

What do you do if your program runs for a long time before the bug arises? You could single-step until you get to the bug, but that can take a long time. A better alternative is to use a *breakpoint* , which tells QtSpim to stop your program immediately before it executes a particular instruction. When QtSpim is about to execute the instruction where there is a breakpoint, it asks for continue, single stepping or abort. To set breakpoint: In the Text tab, move your mouse over the instruction you want to set a breakpoint at, right click and choose “Set Breakpoint”.

Single-stepping and setting breakpoints will probably help you find a bug in your program quickly. How do you fix it? Go back to the editor that you used to create your program and change it. Click on the Reinitialize simulator tab in the tools bar and load the source file again. (or File -> Reinitialize and Load File)

Other Tips

- You can access all of the commands via the “File” and “Simulator” menus as well.
- When examining register or memory data, you can view the data in binary, hex, or decimal
  format. Just use the “Register” pull down menu to select.

- Kernel Text and Kernel Data may not be necessary to be viewed all the times, you can unselect them by unselecting “Kernel Text” in the “Text Segment” pull down menu and unselecting “Kernel Data” in the “Data Segment” pull down menu.
- To view memory data, simply click on the Data tab.
- By right clicking on a register file value or memory address value, you can change its contents
  dynamically.