# Lab 2a (SPIM Tutorial)
# EC413 – Computer Organization

One of the keys to being a good programmer is having knowledge of, and skill with, (not to mention using) programming environments.

The purpose of lab 2a is to introduce you to the SPIM (MIPS backwards) simulation and debugging environment. In particular, you will use single step and breakpoints, view memory locations, and view and modify registers.

We recommend that you use qtSPIM. Unfortunately it is not installable on the 305/307 computers, but you should be able to install it on your own laptops. Please see the TAs if you have questions. An alternative is XSPIM, which is installed on those machines. It is a bit more primitive, but definitely usable.

1. Read "Getting Started With SPIM."

2. Load the program. In the sub window labeled "Text" you should see your code (level 3), along with the equivalents in level 2 (bare MIPS assembly language) and level 1 (MIPS machine language). SPIM actually does not start execution with your program, but rather with some set-up code. You should find your program code following a "syscall" instruction.

3. Begin by clicking on run. Note and write down the output. Simply loading and running the program should have worked fine this time, but in general your programs will need to be debugged. In that case, simply clicking on run will just give you garbage, and worse, little way to figure out what went wrong.

4. Reload the program. This time, single step until you reach the instruction corresponding to 'main.' The last debugger instruction executed should be 'jal main' which transfers control to your program.

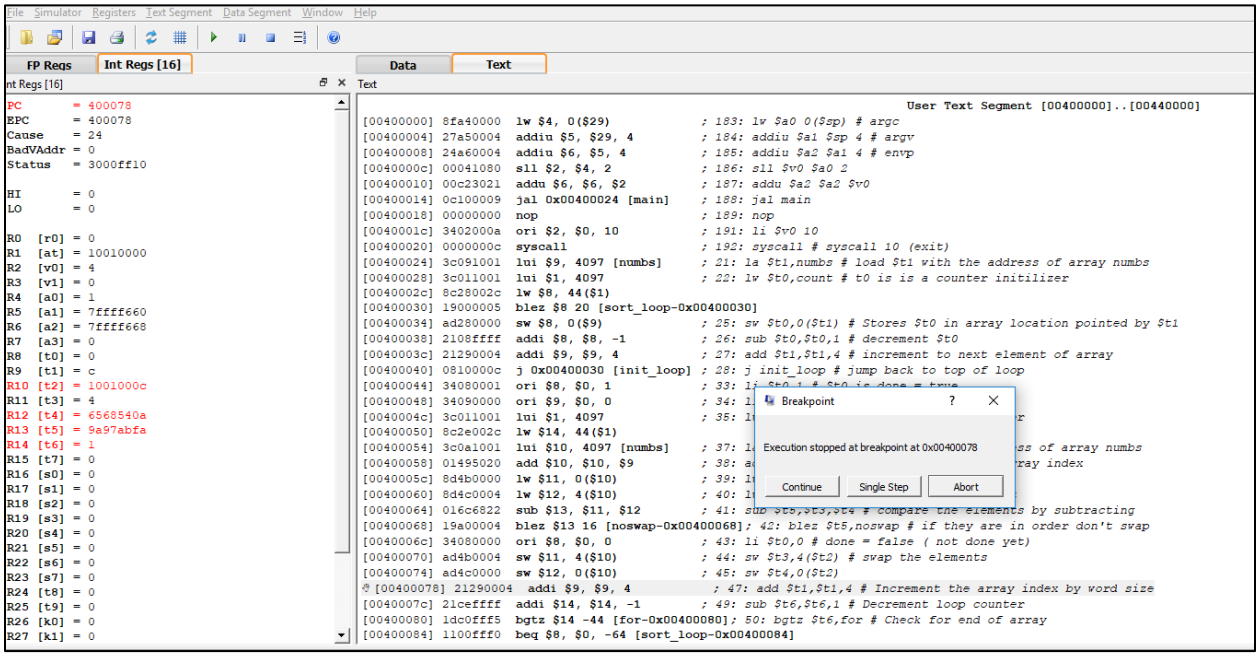
5. In class, for convenience, we assume that the data starts at memory location 0. Where does SPIM put it? What is the address of variable "msg"? Where is the variable "count"?

6. Continue to single step through the program until you reach the instruction corresponding to the label 'sort loop'. What's happening? Observe the values being loaded into the array in the data segment as you step through the program. What are the new array values?

7. Insert a breakpoint at the instruction corresponding to the 'noswap' label.

8. Continue execution using the run command (repeatedly) until the program ends. Write down the contents of the array each time the breakpoint is encountered.

Note: When the execution reaches a breakpoint, QtSpim informs the user with a pop-up window as shown in the figure below. It has been observed by the TAs that QtSpim does not display the complete current state of registers and memory until you close the pop-up window by clicking abort, continue, or single step. If you are required to write down and/or modify the register/memory contents at the breakpoint, we recommend that you click abort to close the pop-up window in order to view the correct execution state.




9. Reload the program. This time, insert a breakpoint at “sw $t4,0($t2)” and run the program.

10. At the breakpoint, change the value of register $t4 to 0 and the value of the second element of the array to 7.

11. Repeat step 8.

12. Modify the program so that it sorts 8 numbers instead of 4. Reload and run it. What are the array values this time?

**Hand in the answers to the questions and a description of your program changes in the last step.**