CS 341: Lab 1 - Introduction to the MIPS Simulator

July 27th 2016

1 Goals

- Learn the basics of spim/xspim, a simulator for the MIPS architecture
- Write and execute some simple assembly language programs

2 Instructions

These exercises are to be done individually. While you are encouraged to discuss with your colleagues, do not cross the fine line between discussion to understand versus discussion as a short-cut to complete your lab without really understanding.

1. Create a directory called $< rollno > - < labno >$. Store all relevant files to this lab in that directory.
2. In the exercises, you will be asked various questions. Note down the answers to these in a file called $< rollno > - < labno >$.txt.
3. In some parts of the exercises, you will have to show a demo to a TA; these are marked as such. The evaluation for each lab will be in the subsequent lab, or during a time-slot agreed upon with the TAs. For this evaluation, you need to upload your code as well.
4. While submitting (on moodle), you have to create a tar.gz or zip of the entire$< rollno > - <$ $labno >$directory in which all your relevant files reside.
5. Before leaving the lab, ensure that you have signed the attendance sheet and you have uploaded your submission.
6. Things to ensure during TA evaluation of a particular lab submission:
   (a) The TA has looked at your text file with the answers to various questions
   (b) The TA has given you marks out of 10, and has entered it in the marks sheet, with his/her signature
   (c) You have counter-signed the above-mentioned marks

3 Starting QTspim

- Start the simulator by typing "qtspim&" on a terminal. A window should pop-up.
- There are four parts on the screen. Note that you can scroll within each of the four parts, using the mouse. Note that this represents a simulated MIPS machine.
- **Question [1 mark]:** In the QTspim window, identify the various parts of the screen which you can understand; also identify the parts which you cannot understand (as yet).

4 Your first MIPS assembly language program

- For a MIPS assembly program, you need to specify a "text" section, which indicates a region in memory where the instructions are stored (recall the stored program concept)
- And for spim, you need to specify a "main" label, from which execution will begin
- Cut-paste the following lines into a file using a text editor; you can name the file anything, but we will assume below that you have named it "add.s"

```
.text
main:
# Your code starts from the line below
```

- Write a program to add the integers 1 and 2; you can use the "li" or "load immediate" instruction, and any other appropriate instruction(s). Use appropriate registers for the instructions.
- Load the program in the simulator using the load button. Identify your part of the program on the screen.
- **Question [0.5 marks]:** Identify the address at which your program begins. You can use the "print" "global symbols" button for this.
- **Question [0.5 marks]:** Is "li" really a MIPS instruction? Its a pseudo-instruction. What does it get translated into?
- **Question [1 mark]:** Identify the machine code of your instructions. Identify the op-codes of the various instructions in your assembly code.
- **Demo to TA [1 mark]:** Run the program by clicking the "run" button. Observe the result of your instructions to load/add in the appropriate register(s).

5 Stepping through your program

- You can step through your program one instruction at a time. To do this, first reload "add.s" using the "reload" button.
- Now step using the "step" button.
- **Demo to TA [1 mark]:** Show how you step through the instructions.

6 Using breakpoints

- You can step through small programs, but it gets cumbersome as the program gets big.
- You can use a "breakpoint" (or several breakpoints) in the program. A "breakpoint" is a point in the code where execution will temporarily stop, allowing you to examine the status of your program until that place. You can examine register contents or memory content, as appropriate.
- **Demo to TA [1 mark]:** After reloading "add.s", set an appropriate breakpoint for your program, and demonstrate its usage to your TA. After encountering the breakpoint, step through the rest of the instructions one after another.

7 Using the MIPS instruction set reference

- You must learn to refer to and understand an instruction set reference. Look at the MIPS32 instruction set reference posted on Piazza under Resources.
- Change the "add.s" instruction to be able to load and add the following two 32-bit values: 0x10000001 and 0x20000002. You should not use any pseudo-instruction for this. You will have to find out how to load a 32-bit value onto a register. Hint: you have to use two instructions for this.
- **Demo to TA [1 mark]:** Show how you have modified "add.s" to now add two 32-bit values. Show a demo of the run on xspim.

8 Your first non-trivial assembly language program

So far you have learnt the usage of spim, and to write a barebones assembly program. Now write a program to compute the sum of a array of numbers stored in memory and put the result in register $v0. Hint-1: you will have to use the MIPS instruction reference for further relevant instructions to use. Hint-2: you may find it useful to first write the C-code and then translate it to MIPS assembly code. Assume that the following code first:

```
.data
N: .word 5
X: .word 2,4,6,8,10
SUM: .word 0

.text
main:
```

Demo to TA [3 marks]: Show how your array-sum.s program works.