CS231, DLDCA Lab, Lab 04

Goals

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

Instructions

1. These exercises are to be done individually.
2. 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.
3. Create a directory called <rollno>-<labno>. Store all relevant files to this lab in that directory.
   a. In the exercises, you will be asked various questions. Note down the answers to these in a file called “answers.txt”.
   b. 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. Before leaving the lab, ensure the following:
   a. You have marked attendance on SAFE
   b. You have uploaded your submission on BodhiTree, and downloaded and checked if the submissions is right
5. 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
6. You have to use the MIPS conventions, unless mentioned otherwise.

Starting qtspim

- Start qtspim 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).

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 xspim using the load button. Identify your part of the program on the screen.
- **Question [0.5 marks]:** Identify the address at which your program (first line of main) begins.
- **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 (higher order 6 bits of the machine code) 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).

Stepping through your program

- You can step through your program one instruction at a time. To do this, first reload add.s using after re-initialize simulator.
- Now step using the step button.

Demo to TA [1 mark]: Show how you step through the instructions.

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. (You can set a breakpoint by right-clicking on the instruction). After encountering the breakpoint, step through the rest of the instructions one after another.

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 available on bodhitree.
- Change the add.s instruction to be able to load and add the following two 32-bit values: 0x20000001 and 0x10000002. 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 qtspim.

Your first non-trivial assembly language program

- So far you have learnt the usage of qtspim, and to write a barebones assembly program. Now write a program to check, given two integers, if one is a multiple of the other.
- Call your program check-multiple.s. You can assume that the first integer is in \$s0 and the second integer is in \$s1. The first two lines of your code should initialize \$s0 and \$s1. The rest of the code should work irrespective of the values of \$s0 and \$s1 (you can assume they are both positive non-zero integers). Your code should work for both s0 > s1 as well as s0 <= s1. Finally, your code should set \$s2 to 1 if one integer is a multipleof the other, 0 otherwise.
- *Hint-1: you will have to use the MIPS instruction reference for further relevant instructions to use, or simply subtract the smaller int from the larger in an appropriate while loop.*
- *Hint-2: you may find it useful to first write the C or Java-code and then translate it blindly to MIPS assembly code.*

Demo to TA and submit [3 marks]: Show how your check-multiple.s program works and submit your check-multiple.s file.