INTRODUCTION TO COMPUTER ARCHITECTURE

LAB 6

Simple assembly programming

1 The assembler/simulator

Assembly programming in RISC-V will show us how a general-purpose programmable circuit—a *processor*—may be used, and what it would need to be capable of doing. We will write assembly programs and run them through an *assembler/simulator* for a RISC-V processor and memory. This simulator will allow us to move step-by-step through our programs, watching the registers and memory change as we do so.

We will use the *BRISC-V* assembler/simulator, which you can access in your web browser at:

`https://ascslab.org/research/briscv/simulator/simulator.html`

You may also find useful, as we do this work, the following RISC-V assembly reference, which includes a number of details and examples about how to write the instructions and annotate your assembly code so that it can be correctly parsed:

`https://github.com/riscv/riscv-asm-manual/blob/master/riscv-asm.md`

2 A pre-written program

Get some code: The best way to understand how this simulator works is to try using it. Begin by getting some sample source code that we will load into the simulator and then step through:

`https://sfkaplan.people.amherst.edu/courses/2019/fall/COSC-163/assignments/lab-6/add-two-numbers.s`

Save this code to your own computer, and then open it in any *text editor*.[1] You will notice a sequence of instructions that are commented, and we will walk through their interpretation during lab.

Use it in the simulator: Go to the simulator page. At the upper-left of the window, immediately under the header text, *RISC-V Assembly*, there are four icons that control the simulator. Click the leftmost one, which is an *upload* icon. Find and select your `add-two-numbers.s` file.

We will then use the *step* button (third from the left of the icons) to move through the program, one line and a time, watching the registers and the main memory contents, to the right, change. Be sure you understand what you're seeing, and ask questions.

[1] Emacs, vi, Sublime Text, Visual Studio, Xcode, whatever. Even Notepad or TextEdit will do.

3 Finding the max

Now grab another assembly source code file:

`https://sfkaplan.people.amherst.edu/courses/2019/fall/COSC-163/assignments/lab-6/find-max.s`

This file contains the skeleton of an assembly program that traverses an array of values, remembering the maximum value so far. **Your task** is to write the loop that finds the maximum value in that array. Once your loop is done, the code already at the end of the program will store that array into a main memory location.

4 How to submit your work

We will one again be using the CS submission system to turn in this programming work. Specifically, you should submit your completed `find-max.s`.

This assignment is due on Oct-24, 11:59 pm.