Systems I
Lab 0
Simple assembly programming
This assignment is due Thursday, Sep-10, 11:59 pm.

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 *Venus Simulator*, which you can access in your web browser at:

`https://venus.cs61c.org`

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://marks.page/riscv/`

Hold in mind that **we will return to RISC-V assembly later in the semester**. At this moment, we want to see some bare basics about how software is written and how it behaves at this lowest level. We will explore it more deeply in a couple of months.

2 A pre-written program or two

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

`https://bit.ly/COSC-175-2627F-lab0-source`

This link should download a file named `lab-0.zip`. You should extract the contents of this zip file.[1]

Upload the first program into the simulator: Open the simulator in your browser. On top, if it is not already selected, click on the `Venus` tab. Below, you will see a basic terminal and shell prompt. At this prompt, end the following command:

```
[user@venus] /# upload
```

The simulator will present you a file dialog. Choose the `numbers.s` file from the extracted zip file.

[1] On *macOS*, double-click the file to extract its contents. On Windows, right-click the file and select *Extract all…* to do the same.

Open the uploaded code: In Venus, enter the following command:

```
[user@venus] /# edit numbers.s
```

That will bring you to the `Editor` tab, where you will find the `numbers.s` code. You can look through it and try to follow the comments, which describe what the code does at each step.

Run the program: *We will walk through this part together during lab*, first examining the source code, and then seeing its translation into machine code steps as we step through the execution of the program.

Another program: Now let's consider `countdown.s`, another program from the zip file that counts down (and prints!) integers from 5 to 1. Repeating the steps above, *upload* this code into the Venus simulator, open it in the *Editor*, and then run it in the *Simulator*. Be sure that you understand what is happening at each step of the program; ask questions about anything that is mysterious!

3 Your assignment

There is one last assembly file, `doubling.s`, in the collection that you downloaded as a zip archive. Upload this source file into the Venus simulator, and then open it in the editor.

Notice that the Java code shown in the comments describes a loop, but that the assembly code below is incomplete—the body of the loop is missing. **Your task** is to write that loop body, doubling the value in register `a1` (a.k.a., the variable `x`) until, somehow, the value becomes negative.

Which also poses **a question for you to try to answer**: How, if we start with a non-negative initial value for `a1/x` and then double it with each iteration, does the value become *negative*? What is happening? At what value does it become negative, and why?

4 How to submit your work

Download your code: When you have written code within the `doubling.s` file, go back to the `Venus` tab terminal. Then, download it, like so:[2]

```
[user@venus] /# download doubling.s
```

Copy your completed `doubling.s` to the `lab-0` folder within your Google Drive folder for this class.

[2]Indeed, you might want to do this step periodically, downloading a backup of the work-in-progress as a guard against your session with the Venus simulator disconnecting/crashing/getting lost.