CS 252: Computer Organization

Pre-Project

(see class website for due date)

Don't miss this handy YouTube video:

- Intro to the Unix Command Line (using Docker)
  `https://youtu.be/Yt7D1JQKmdI`

Purpose

This project exists simply to make sure that everybody figures out how to use the tools we'll be using for projects this semester. You'll write a tiny bit of code, but mostly what you'll be doing is downloading code that I've provided, and running it to test your environment.

Special Rule - Shared Work

For this project only, you may work with as many people as you want! The point of this is not to "test" you in any way, but simply to make sure that you know how to compile and run the programs you will be writing this semester.

There is only one requirement: **you may only turn in results that you generated.** You can have anyone help you, as much as you want - but in the end, you have to compile and run the code yourself, and turn in that result.

Task 1 - Write a Simple Java Program

A few students may have experience with C, but not with Java. That's fine, we won't be expecting anything very hard! Watch the videos linked at the top of this document; if you need more help, come to Office Hours. Or, you could just follow a simple Java tutorial from the web.

Some of the Simulation projects will require that you write and test some Java code. We expect that all students who are taking 252 have done some Java programming before, so for this step, you must write a simple program in Java. (Make sure to test it!) Your program must:

- Print out a welcome message to the user
- Read a single string (one line, or one word) from the keyboard
- Print out something which includes what was typed

Turn in **exactly one** `.java` file to GradeScope (name it whatever you'd like).

Task 2 - Compile Your Choice of C Programs

In this class, we don't expect that you have written C before. We'll be using C for a couple of the Simulation projects, but **don't worry** - we'll only be using a very tiny subset of the language, and I'll show you everything you need to know.

In the preProject zipfile, I've posted several fun programs from the International Obfuscated C Code Contest. These programs are quite silly - they are **intentionally designed** to be as hard to read as possible. So you are free to look inside them, but don't expect to be able to understand them - even **I** find them pretty unreadable!

You will need to compile one of these programs and run it.

- If you have used Visual Studio, you can set it up to compile C programs. (Other IDEs should work, too.)
- If you are on Mac, you should be able to compile using the Terminal (Google for details).
- If you are on Windows, you should be able to use Windows Subsystem for Linux to compile (Google for details).
- On any platform, you should be able to run Docker - and then compile on the command line there. If you want to try this, watch the video that I linked to, on the top of this spec.

Once you have compiled and run the program, save the output from the program, and turn it in to GradeScope. You should name the file `<program>.txt` (replace `<program>` with the name of the program you chose). You only have to do this for one of the programs - although you're allowed to do it for more than one if you enjoy it.

Task 3 - Run the MARS Simluator

One of the files in the zipfile is `Mars4.5la.jar`, which is the MARS simulator, which we'll be using for assembly projects. It's a Java application; you can probably run it simply by double-clicking on it. Run it, and take a screenshot of the window that pops up; make sure to include that screenshot when you turn in all your files to GradeScope. You can name the file anything you like, but it must be one of the following file types: `PNG, GIF, JPEG, BMP, PDF`. (For the sake of the TAs' sanity, we will not accept other file types.)

(spec continues on next page)

Task 4 - Run the Grading Script

Open up the Docker container (see the video to learn how), and `cd` into the directory where the preProject files have been mapped.[1]

Run the grading script `grade_preProject`. For the rest of the projects, you will be using grading scripts like this to check to see if your code works correctly. Of course, in this case, we don't have code to auto-grade; instead, I've provided a trivial little assembly program (`asm_preProj.s`) and a testcase for it (`test_01.s, test_01.out`). If your environment is working correctly, the grading script will report that you passed the testcase. Save the output from the grading script to a text file, and turn it in with everything else, when you turn in things to GradeScope. Again, name it whatever you want (so long as it's obvious what it is).

Turning In Your Answers

You must turn in your answers to GradeScope. Remember, while you can turn in things as many times as you want, you must **turn in all of the files at the same time;** we will grade the most recent files you upload. But if you only upload a few files at a time, we will only see (and grade) the ones in the last upload group.

Once you've uploaded your files to GradeScope, you're done. Good job!

[1] If you want to use some other UNIX environment, such as WSL, Mac, etc., you are welcome to try. But I think that the Docker container is the easiest way, and it's the one that I will officially support as the "right" way.