Summary

The main purpose of this project is to explore processes and how they can communicate with each other. When using multiple processes in tandem, it is important to understand inter-process communication (IPC) so that we can avoid common mistakes such as deadlocking, mutual exclusion, life-locking, and many more. This project aims to improve us understand how a computer handles communication between the processor and main memory by building an imitation model of the system in a java or C++.

The project was implemented in java using Runtime exec to create processes and streams for communication. The CPU and memory were implemented as separate classes (named SimpleComputerProto and MemoryProxy in my case). They communicated with each other using Runtime exec to simulate processes. The CPU contained all the registers, I/O requests, and interrupts, being effectively the brain of the program. The memory did not contain any logic, only simple read/write commands, and arrays of length 2000 to simulate storage space. The memory also contains some minor code to properly process the communications through pipe sent by the CPU, but the CPU was the focus of relevant logic for this project.

Personally, I found this project to be a lesson in tedium, where the project was not necessarily difficult, but the debugging portion was extremely difficult to implement properly. There were many cases where I found it extremely difficult to find out why my program was returning incorrect outputs, and often the compilation errors were extremely unhelpful. After a few days of struggling to properly debug my program, I looked for help to better structure my program’s debugging logic and finally allow for me to debug in peace. If I learned anything from this struggle, it is that debugging is every bit as important to learn as regular programming, and that correctly structured debugging is multiple times more efficient than the blind debugging I was doing before.

On another note, I also had difficulty running my program from my personal command prompt and the school’s Linux system. At many times I was not sure why the commands needed to be formatted a certain way and at what points I needed to use absolute or relative pathnames. I ended up simply trying every combination of relative and absolute pathnames until I got it to work, but this is an issue that will make it more difficult for the grader to run my program. I am also unsure about when I need to be using the path to the .class file and when I need to be using the path to the .java file. Using run configurations in my Eclipse IDE, I know that my project is fully functional. Hopefully I can solve my problem and produce a project that can easily be run on any environment.

I think that the concept of pipes and IPC is not overly difficult to understand, what is difficult to understand is *why* the program doesn’t work and exactly what step of the program it breaks on, since the compiler doesn’t return line numbers by itself when the problem is returned from the try/catch. Perhaps this is a signal that I have been relying too heavily on the crutch that is an IDE and should work harder to produce better code with fewer errors that is easier to read and debug. That’s a good goal to set, best of luck to me.

UPDATE A FEW HOURS LATER: It works! I only went slightly insane trying to make it work, but it works! If you follow the directions I appended onto the bottom of this word doc (and in the readme.txt file), my project can easily be run in a LINUX environment! Praise the gods, (and me), but mostly me.

Instructions to Run Project on LINUX environment (also in readme.txt):

mkdir simpleComputer

// ---> Put files in simpleComputer

// Make sure that the path settings in MemoryProxy.java are correct:

// That means that memoryClassPaths should direct to one folder above the package

// as shown below

String javaRT = "java";

String memoryClassPaths = "-cp /home/011/v/vy/vyw180000/";

String memoryClassFullName = "simpleComputer/MemoryProto";

cd simpleComputer //make sure you are in simpleComputer directory

javac MemoryProxy.java

javac MemoryProto.java

cd //Now you should be one level above the simpleComputer folder(package)

javac -cp . simpleComputer/SimpleComputerProto.java

java simpleComputer/SimpleComputerProto simpleComputer/sample1.txt 30

java simpleComputer/SimpleComputerProto simpleComputer/sample2.txt 30

java simpleComputer/SimpleComputerProto simpleComputer/sample3.txt 30

java simpleComputer/SimpleComputerProto simpleComputer/sample4.txt 30

java simpleComputer/SimpleComputerProto simpleComputer/sample5.txt 30