Thiago Ciriaco

Operating Systems Concepts

2/27/23

Professor Ozbirn

Project 1 Summary

1. Project Purpose

The purpose of this project is to use high level code to simulate, with less complexity, the interactions between a real CPU and memory. Creating this model in software makes it easier to understand how a CPU and memory work together in practice. This project models the CPU and memory as separate processes, mirroring how they are separate entities in a real computer.

1. Project Implementation

In my implementation of this project, I decided that it would be smarter to implement the CPU as the parent process and the child as the memory. This is because once the simple memory process is working, most of the debugging needs to happen in the more complicated CPU process. I created 3 C++ files: one “main.cpp” file for general code and the main function, one “processor.h” file containing a processor class and its methods, and one “memory.h” file containing the memory class.

The majority of main.cpp is setup and looping. It checks that the program arguments are in the correct format, creates the necessary pipes, forks the process, helps memory with the initial program file read, and creates the Processor() and Memory() objects.

The memory process has the initial read followed by one main loop that waits for the processor to write a request (r for read, w for write, e for ending the process). Once it receives a request, it waits for the necessary information for that request (such as an address or data) and writes back its result to the processor. In the case of reading, it returns the value at that address. In the case of writing, it returns 0 to indicate a successful write.

The processor process only has one main loop that calls only 2 functions: fetch() and execute(). These functions are written in the processor class and are meant to simulate the fetch and execute phases of a real CPU. In short, the fetch function gets the next instruction by requesting the data in the memory (based on PC), while the execute instruction selects and carries out the instruction using a huge switch statement and helper functions for each instruction.

1. Personal Experience

I began this project quite late, and while it would have been safer to start it earlier, I still really enjoyed knocking it out in a little over a weekend. At first I struggled to get the memory working because my memory array was not being instantiated correctly. However once I realized that, all of the work went into the processor class. Implementing the simpler functions was a piece of cake, but I had the most difficulty implementing the interrupts. Depending on whether it was a timer interrupt or a system call, the return would behave a little differently and sometimes skip an instruction. After about an hour of stepping through with the vscode debugger I found the issue though, and it was back to working.

For me, the most enjoyable part of the project was seeing it all come together. Having test files that test the progress of the implementation little by little encouraged me to keep going and served as checkpoints where I could backup my code just in case I broke it while trying to get the next part to work.