Kevin Roa

Professor Ozbirn

CS 4348.501

03/06/2022

Project Summary

The overall purpose of this assignment was to simulate the CPU and memory of a simple computer. The CPU had to implement a special instruction set and communicate with the memory. The main goal was to understand how multiple processes could communicate with each other in an orderly fashion. Additionally, it was to understand the following concepts: instruction behavior, registers, stack processing, procedure calls, system calls, interrupt handling, memory protection, I/O, and virtualization/emulation. By completing the project, students should have touched on each of those topics to some extent and gained a better understanding overall.

My project was implemented using C++. I created a main.cpp which initializes the program and loads the various other files. I created new classes for both the CPU and Memory and put them in their own header files. I decided to only use header files because it was simpler to write/build. In the main.cpp, I forked new processes for the CPU and Memory respectively and also, created 2 Unix pipes to facilitate communication between the processes.

In the CPU class, I created a runtime loop which loops until the program receives an instruction to terminate (or if it runs into an error). The loop reads the value at the address of the program counter and sets it to the instruction register. It then enters a large switch statement which calls the relevant instruction method. Upon entering the method, it performs what the specific instruction entails and then returns back to the loop. Throughout the duration of the loop, a timer object counts every instruction. When a timer interrupt occurs, instead of reading the program counter, it immediately calls the interrupt instruction and performs the relevant tasks. Additionally, the CPU has two methods to read/write to the memory process. Various instruction utilize these methods to modify the memory stack.

In the memory class, it first reads the file passed in as a parameter and parses it. The parsed data gets written to the memory array (stack) and thereafter, the process loops until it is terminated. In the memory runtime loop, it waits to read data from its pipe. Once data is read, it figures out what type of action it should perform and reads in the necessary parameters. In the case of a READ action, the memory will also write data back into the pipe for the CPU process to read.

Overall, I thought that the project was fun, and I learned a lot. I required me to review previous assignments to figure out how to handle certain things such as processes and pipes. While on the other hand, it also made me read the relevant sections in the book to learn about topics I had never covered before. My only gripe is that some things were not as clear as I thought they should have been. Obviously, I understood everything in the end, however, it was not without its trial and error. As for actually writing the program, there were some points where I had no clue what was wrong and my initial process for debugging was not great. (Using breakpoints and stepping through the program) It caused me to create a small system for creating debug files where I could quickly understand what was going wrong. (Funnily enough, the biggest problem I had was from a small 1 line typo which was quite frustrating) Other than that, the project went by smoothly.