Project Purpose

The purpose of this project was to teach us about how a simple operating system works and how operating systems use multiprocessing. We created our own virtual operating system that ran its Memory and CPU on separate processes.

Implementation

I implemented my project in c++ and put all of the functionality in one file. Using one file made compiling and running the program easy but led to some spaghetti code in the CPU section of the program for the more complex instructions. If I was to create another virtual OS I would write separate CPU and Memory classes so I could simplify the code.

My implementation of the OS starts by reading the entire user program into memory, before the process split. This kept the CPU from needing to wait on memory while reading the input. After the split, memory blocks itself waiting to read information from the CPU through a pipe. When the CPU is ready for an instruction it writes to memory asking to read from a specific address, it then waits for memory to send that instruction. It then executes the instruction it gets, doing read or write calls when necessary. The CPU uses the program counter to track where in the user program it is, it reads instructions into the instruction register, the accumulator is used for modifying data, the stack pointer points to the end of the current stack, and x and y are temporary variables used for modifying and storing data as well. The timer counts up before every instruction and causes an interrupt when it reaches the threshold designated by the input. Memory has two components, the user memory and system memory. They can each hold 1000 integers and each have a stack that grows down from the end of it. At first I implemented my stacks where SP pointed ahead of the value, but sample2 assumed that SP pointed directly at the value so I had to slightly change my implementation of the stack. For the protected system memory I just used a boolean called kernelMode to check if a user was trying to access system memory. If the address being accessed was in system memory and the OS wasn’t in kernel mode, then it exited with an error.

Personal Experience

My first hour or two with the project was frustrating. The scope of the project left me confused and overwhelmed at first, but after a couple hours of planning and continuously rereading the word document the project seemed much more manageable. It took almost two hours to fill in the main structure of the code, and then 3-4 times that working through bugs. I ran into a lot of problems while bug fixing: I created the pipes after splitting processes, silly mistake by me; I misunderstood what the Loadspx instruction was doing, I luckily realized my mistake when someone else was talking about a completely different problem they had with that instruction; I went back and forth many times on decrementing PC after certain jumps and interrupts, lots of trial and error trying to find which ones did and didn’t need a -1. Despite all the struggles with bugs, I was surprised to find that I enjoyed this project quite a lot. My code was long and messy, but relatively simple to write once I figured out what I was doing. Getting more familiar with the command line was very nice. I also really enjoyed creating sample 5; trying to write a program with a limited language was a fun challenge and pretty refreshing after hours bug fixing. Overall, I had a lot more fun with this project than I thought I would and I feel like a learned a lot from it.