Cameron Meyer
CS 4348.001

Project #1 Summary

In Project #1, I was tasked with simulating a simple computer system--an assignment which would help reinforce class concepts such as processes, pipes, system calls, and memory handling. To implement this myself, I had to break down a computer system's key functionality and understand the relationship between the CPU and memory. This project was effective in helping me learn about memory management and syncing up processes to communicate effectively, while also toggling between the user program and system code as necessary. The time spent on this activity was a hands-on exercise in some computer architecture and low-level concepts within an operating system.

I implemented this project in C++, so I ended up using a Unix fork to create my processes and I used Unix pipes to allow the processes to communicate. Since my program needed the ability to read file input into memory, I parsed each line of input for a period or integer value, which I could use to access specific memory addresses or run commands, respectively. Commands values read into memory are then run through a switch statement to determine which instructions to run. My program continuously increments the program counter register to access each value in memory and properly execute the command found at each address. Implementing such a feature also required memory protection, in which my program was not allowed to access out of bounds memory addresses or access system code memory if the system was not in the kernel state. The kernel state was handled in its own function, in which my program could trigger timer interrupts and system calls as necessary.

When beginning this assignment, I definitely struggled for a while to properly set up my development environment. It took me several hours to slowly sort out what I needed to do, and more guidance on this front would definitely be invaluable in future projects. After failing to get a working environment in Windows, I forayed into installing Ubuntu on my machine and messing around before ultimately figuring out that I could SSH into the cs1 server directly from VS Code. I did all my development on the UTD Linux server and used the command line to save my progress into a git repository.

Throughout development, I faced several bugs, though my most common issues involved infinite loops and invalid memory accesses. To solve these general issues, I created plenty of debug statements and spent lots of time tracing my code and output. I also made use of the ' | less' pipe in the terminal so that I could easily traverse my output when dealing with infinite loops. My infinite loops were usually caused by some value getting incremented at the wrong time, and it required me to be very careful about when my program would move on to new instructions. The bright side of dealing with infinite loops is it also helped me locate invalid memory accesses in my code, which made it much easier to develop memory protection and throw errors when appropriate. Additionally, I struggled for a while to get my push and pop commands working properly. Even now, I still don't completely know what was wrong with them to begin with, but once I eventually moved push and pop to their own functions outside of the switch statement, everything finally clicked. Now, both the push and pop functions are called directly from the switch statement and they are also called from my function that manages my

program’s kernel mode. Through plenty of trial and error and rigorous testing, I was eventually able to end up with a final product that produces the project’s expected output, and I also more deeply understand the low-level mechanisms of CPU and memory communication in a computer system.