Project 1 Summary

**Purpose:**

The purpose of this project was for us to become more familiar with the inner-workings of an operating system regarding the relationship between the CPU and memory. The project provides an intuitive way to understand how memory is divided between kernel space that is reserved for OS specific needs, and user space that is accessible by any process. By using a simple array to simulate memory, and forking a process to simulate the CPU, this project provides an easy way to learn about the fetch and execute process, the use of registers, the use of stack space, and how procedure calls and interrupts are handled and affect all of these other components.

**Implementation:**

I simulated memory and CPU by forking a second process in my main function. The parent process was my “memory” and its sole job was to read and write to the memory array. I used several functions to perform the reading and writing and to check the validity of given addresses. Memory was initialized by reading from a file that was given as a command line argument. The “CPU” was contained in the child process and takes care of fetching and executing commands and handling interrupts. The CPU read a timer value from the second command line argument and counts up to that value while in user space, jumping to address 1000 when the timer expires. In order to be most consistent with a real operating system, the CPU does not have direct access to memory, it can only make a request to memory. So, I implemented two pipes that allow the CPU to make a request to memory for reading or writing. This request is formatted using an array; the zeroth index represents whether we are in user or kernel mode (0/1), the first index represents whether we are making a request to read or to write (0/1), the second index holds the address at which we are reading or writing, and the third index contains the value to be written if any. This way, all of the information that memory needs to fulfill a request is conveniently contained in one structure. Finally, a switch statement in the CPU process determines which command was fetched and makes the appropriate actions to fulfill that command.

**Personal Experience:**

I believe that this project was a wonderful way to become familiar with the basic functions of an operating system. This project challenged my knowledge of the concepts we have learned so far, but was not so complicated as to be a burden. Since completing this project I feel much more comfortable in my understanding of the fetch/execute cycle, how the stack is used, and in how procedure calls and interrupts are handled by the CPU. Additionally, this project provided me with good practice using pipes to allow processes to communicate.