**Project #1 Summary**

The purpose of this project was to simulate a simple computer system consisting of a CPU and Memory. The CPU and Memory was simulated by separate processes that communicate with each other via pipes. Both processes were created from a single program that used a forking method to create both a parent (CPU) and child (Memory).

The project objective was to obtain a better understanding on the use of pipes and how the communication between a CPU and Memory interact in a real computer system, as well as understand the low level functionality of a primitive Operating System, which include: communication and cooperation between multiple processes, basics of processor-memory interaction, simple processor instruction behavior, functionality and purpose of system registers and user registers, simple stack processing, procedure calls, system calls, basic interrupt handling, exposure to memory protection, and simple input and output through I/O ports.

The team divided the program into the two main parts: CPU and Memory. The program structure was quickly decided by all parties that the CPU and Memory should be separate objects, which are then initialized and joined by a third script. Ashley Tharp and Brian Quackenbush designed the initial makeup of Memory, whereas Tyler Reed designed the initial makeup of CPU. Ashley and Brian wrote and debugged the Memory class then assisted Tyler in completing and debugging the CPU class. After each class was written to the minimum specifications, they were combined and tested. The joining and debugging process was performed using the peer programming method. Ashley and Tyler debugged Memory and CPU while Brian went through each and helped them both on errors. Once the program had been compiled, it was immediately tested by the sample problems given. In short, Ashley wrote the Memory class, Tyler wrote CPU class, and Brian assisted both with design and debugging and testing of each part of code.

The code was implemented in order of need. Memory was first implemented to ensure that the pipes were connected correctly and that it read and wrote through the pipe correctly. CPU was first implemented to communicate with Memory, and was given little functionality to ensure that the program compiled. Once the two objects were stable enough to communicate properly, functionality was added to both, as consistently tested with the various sample programs. This style implementation of consisted of multiple debug sessions where leaks and small mishandlings were found and fixed by all three member of the team.

The overall team experience in completing this project exposed them to working on a more complex computer system than they had written before, and included an understanding that a problem is more easily found by a new set of eyes then by the direct person who wrote that particular line of code, leading peer programming to be a successful development process. The team each gained personal insight into everyone’s writing styles, and lead to the recognition that each style had pros and cons, but when combined with slight changes developed into easy reading for everyone.