Exploring Multiple Processes and IPC

**Project Purpose**During February of this year, I had the absolute pleasure of working on a coding project that involved emulating the inner workings of a computer. Through this simulation, we were to further our understanding of how CPU and Memory cooperate and communicate with each other. The CPU consists of registers and implemented instructions that allow it to read and write to the memory. The project also let us understand different concepts inside of a computer such as stacking processing, system calls, and interrupts. Overall, the purpose of the project was for us to be more knowledgeable on the communication and technicalities it takes for the CPU and Memory to function together and in the end, served a great purpose of learning.  
  
  
**Implementation**  
  
The implementation of the project was difficult at the start. At first, I wanted to go with my gut and choose C because of the already built-in mechanics, but in the end, I chose coding in Java as I am the most well-versed in the language. I have two big classes inside of my main Project1 class called Mem and CPU (respectively). The main inside of my Project1 class creates an instance of the CPU which then generates all of its registers (AC - accumulator, PC - program counter, SP - stack pointer, IR - instruction register, X and Y). Then, the runtime exec method creates the new process. “Scanner memIn = new Scanner(Mem.getInputStream()” gets the input stream and “PrintWriter memOut = new PrintWriter(Mem.getOutputStream()” gets the output streams that communicate between the processes. The Mem class gives the actual memory of the emulation 2000 entries (0-999 for the user program, and 1000-1999 for the system code). Altogether, we are able to write to my Mem class while going through the CPU to iterate through instructions from the user through the use of a switch case. The CPU also handles the vital component of my program which is known as the timer. The timer, which interrupts the process after a given number of instructions utilizes the program's instruction counter to track said number of instructions.   
  
  
**Personal Experience**  
  
My personal experience throughout the course of the month while coding this project was a stressful one. Balancing the workload it took to complete the said project while also working part-time and focusing on other exams and academic work was definitely a challenge (to say the least). Through it all, however, I believe that my understanding of communication between CPU and Memory has come to an all-time high. This project has also massively helped me in expanding my understanding of piping and interrupts and I feel as if I learned a lot more than I knew before. Personally, I would say that the hardest part of the project was getting my memory properly working and coordinating with the CPU. I spent a good couple of days on that process alone. One of the aspects that I enjoyed the most about the project was the amount of research I had to do to understand all aspects of the creation of processes in Java. During the first couple of weeks when I was still trying to get my wheels rolling, I was absorbing information like a sponge. Overall, this project was a fantastic learning experience for me, and I feel as if I gained a very strong project in my portfolio. The pure amount of information that I learned is exhilarating to reflect on.