**Project #1: Exploring Multiple Processes and IPC**

**Summary**

**Project purpose –**

The main purpose of this project is to understand how to create processes and communicate between these processes. Most of the times, these processes are interdependent and hence these processes need to communicate with each other. IPC (Inter-process communication) allows processes to share data between each other. IPC can be done using pipes or streams. To understand the IPC concepts, we have simulated a computer system consisting of processor, system bus and memory. Processor is a process which is responsible for reading the instructions from the memory, then executing these instructions one by one. Whereas memory is separate process which is responsible for responding to processor calls such as read and write. System bus acts as a medium of communication between these two processes. System bus implemented using the streams mainly input and output streams. Along with IPC, it is equally important to understand various components of operating systems such as registers, stacks, I/O, Interrupt handling, Procedural calls, Instruction cycle execution, Interaction between the processor and memory.

1. IPC (inter-process communication) using streams
2. Processor and Memory interaction for execution of user programs
3. Interrupt handling
4. Use of program counter, stack pointer and various other registers such as accumulator, x, y and instruction register.
5. Function calls, and returns and role of stack in calls and returns
6. Instruction cycle along with interrupts which triggers switching modes.
7. User mode and System mode, saving registers while switching modes using system stack.

**Implementation –**

I have implemented this project in Java. To simulate processor, I have created a CPU class which is responsible for performing following operations –

1. Creates a Memory process using Runtime.getRuntime().exec.
2. Fetch next instruction from memory by using program counter
3. Decode and process the instruction
4. Handle interrupt by saving registers and mode switching to system
5. Read and write to the memory using input and output stream of the memory process.

* To simulate memory, I have created a Memory class which is responsible for performing following operations –

1. Reads user program file and load it in the memory array of size 2000
2. Then it keep on checking if there is anything in its input stream which is sent by CPU. If yes, then depend upon the operation, it reads or writes into the memory.

* When CPU starts, memory process created. Memory process reads the user program file using buffered reader and loads the program into the array of integers of size 2000. If program has comments then it eliminates it and only saves the opcode which is nothing but the instruction.
* CPU initializes the program counter to 0 which is start of user memory. Then it fetches the instruction by using buffered writer stream of memory process to send the address and when Memory process sends the value then CPU read it by using buffered input stream of the process.
* There are 31 predefined instructions, one of the instruction executed by the processor at a time. Based on the instruction, processor reads or write or adds or subtracts, etc. CPU fetches the instruction and put it into IR and then using switch case the respective action executed by the CPU.
* While running the user program, two types of interrupts can occur. One is system interrupt which occurred due to the instruction and other is Timer interrupt (which is like an I/O interrupt) which occurs after specific number (timer value) of instructions. After system interrupt, CPU save registers such as PC and SC into system stack and start processing interrupt from address 1500. Whereas after timer interrupt, processing starts from address 1000. While running interrupt handler, interrupts are disabled to avoid nesting interrupts.

**Experience –**

Creating this project was really enjoyable, as I was learning a lot of things together such as IPC concepts, CPU and Memory interaction, Interrupt handling, function calls and returns. When I started developing the project, I spent major portion in understanding and developing the code for IPC between two processes. After finding the way with IPC, I was really happy and excited to develop the remaining code. The second critical portion was interrupt handling where pushing and popping the registers was involved. At the end of completion of the project, I was thorough with Op-Codes. The most interesting part was to develop a user program which will do something interesting. So there was no bounds to my imagination, after thinking a lot I decided that I want to print something interesting on the console. And after spending around 10 hours, I end up with ‘Batman’ logo on console. I was feeling accomplished after developing my ‘Batman’ program. I am really happy that this project is part of curriculum as developing this project was enriching experience in terms of knowledge about IPC and operating systems.