# **Summary**

One of the advantages of modern day Operating System is to create multiple processes where each process has certain instructions to execute. Some processes require other process to do some operations which will impact the former processes. Creating multiple processes also helps in running each process in different processor thereby running both at the same time. There must be a way to communicate between two processes for sharing data between them. This is done by creating a pipe and streaming data through it. The only function of the pipe is to connect two process. The two processes can send and receive requests or data through it.

In this project, the communication process between CPU and Memory is simulated. The CPU contains all the operations and the brain of the program. It has operations to carry out Interrupts and return from that state, I/O requests, storing and writing data in the registers, stacks, etc. The Memory contains blocks where data is stored. The process which will create the pipe is the IPC. The CPU and Memory communicate with each other though the IPC. The CPU sends stream of data to the pipe which has the read or write information.

The project is coded using JAVA language. The outputs were testing in UNIX college server using Putty software. Eclipse IDE was used to code the program. The program is executed by calling the CPU which is the main class. The CPU calls the memory class using Runtime.exec() operation along with an argument which contains the path of the text file of instructions. It has variables for registers namely Program Counter, Stack Pointer, X, Y, Accumulator and Instruction Register. The CPU can execute 31 operations. The operations are enclosed within a switch block which checks for the instruction. There are 15 instructions which require Opcodes. The Int instruction pushes all the registers to the stack and sets the PC to 1500. When timer is reached all the registers are pushed to the stack and sets the PC to 1000. To return from the interrupt, ret function is used which pop the registers thereby setting PC to the old value. When the process is in user mode, it cannot access memory blocks from 1000. If it tries to access, then the process is closed with an error message. There are two operations the pipe namely read and write. The read() method send read request to pipe to read data from the memory address while the write() method write data to the address.

The memory class is created by using the Runtime.exec() method and the path of the program is sent as argument. All the lines in the program are read in the memory class by using the FileReader class. The lines which does not contain valid information are ignored. If there is “.” before a number, then it is taken as JUMP to address. If only number is present, then it is taken as value. All the comments and empty lines are ignored. There are two operations the memory can perform namely read and write. The read() method reads the request from the pipe and sends back the data while the write() method, write the data to the address and does not return any acknowledgement. The memory class differentiates between read and write request by checking the format of the request. If the request has one number, then it’s a read request. If it has two numbers, then it is write request.

The sample5 program prints “UT DALLAS” is special characters. It contains instructions to load value from address to AC and print it as a character. If the value of AC becomes 0 at some point, the program exits itself.

The zip file contains the following:

1. CPU.java 🡪 Parent Class
2. Memory.java 🡪 Child Class
3. Summary
4. Sample Programs
   1. Sample1.txt
   2. Sample2.txt
   3. Sample3.txt
   4. Sample4.txt
5. Sample5.txt 🡪Created program

## **COMPILE & EXECUTE INSTRUCTIONS**

1. javac CPU.java
2. javac Memory.java
3. java CPU sample1.txt 30
4. java CPU sample2.txt 30
5. java CPU sample3.txt 30
6. java CPU sample4.txt 30
7. java CPU sample5.txt 30