**PROJECT 1 – SUMMARY**

**OVERVIEW:**

The project will simulate a simple computer system consisting of a CPU and Memory. The CPU and Memory will be simulated by separate processes that communicate. The implementation is done in Java and can be executed on the UNIX platform.

**PURPOSE:**

The purpose of this project is to learn how multiple processes can communicate and cooperate with each other. Also, it is meant to help us learn the different concepts of an operating system such as Processor interaction with main memory, I/O processing, memory protection, system and procedure calls, role of registers and stack processing.

**IMPLEMENTATION:**

I used Java to implement the communication between the CPU and Memory. To implement this, the Java Runtime library was used to spawn the child process. In this case the CPU class spawns memory and communication between the two initiates. The I/O streams of the two processes are used to serve the purpose of pipes. The CPU starts reading instructions from the memory and processes them within a switch statement. The following functions were implemented in the project:

|  |  |  |
| --- | --- | --- |
| **FUNCTION** | **PURPOSE** | **LOCATION** |
| private static void fileNameToMemory(PrintWriter pw, InputStream is, OutputStream os, String fileName) | Send file name to the memory process | CPU.java |
| private static int readFromMemory(PrintWriter pw, InputStream is, Scanner memory\_reader, OutputStream os, int address) | Function to read and instruction from the memory | CPU.java |
| private static void writeToMemory(PrintWriter pw, InputStream is, OutputStream os, int address, int data) | Function to write data to a particular address in the memory | CPU.java |
| private static void processInstruction(int value, PrintWriter pw, InputStream is, Scanner memory\_reader, OutputStream os) | Function which processes the instruction using switch statements | CPU.java |
| private static void checkMemoryViolation(int address) | Function to check if user program is trying to access system memory | CPU.java |
| private static void interruptFromTimer(PrintWriter pw, InputStream is, Scanner memory\_reader, OutputStream os) | Function to process an interrupt caused by the timer | CPU.java |
| private static void pushValueToStack(PrintWriter pw, InputStream is, OutputStream os, int value) | Function to push a value to the stack | CPU.java |
| private static int popValueFromStack(PrintWriter pw, InputStream is, Scanner memory\_reader, OutputStream os) | Function to pop a value from the stack | CPU.java |
| public static int read(int address) | Function to read data at the given memory address | Memory.java |
| public static void write(int address, int data) | Function to write data at the given memory address | Memory.java |
| private static void readFile(File file) | Function to instantiate the memory array from the sample files | Memory.java |

**SAMPLE FILES:**

1. sample1.txt – Prints A-Z followed by 1-10

Output: ABCDEFGHIJKLMNOPQRSTUVWXYZ12345678910

1. sample2.txt – prints a smiley face  
   Output:   
    ------

/ \

/ -\* -\* \

| |

\ \\_\_\_\_/ /

\ /

------

1. sample3.txt – prints the letter A ten times, each time calling the interrupt handler which prints a count.

Output: A

0

A

1

A

2

A

3

A

4

A

5

A

5

A

6

A

7

A

8

1. myprogram.txt – Prints the first 10 Fibonacci numbers  
   Output:

0

1

1

2

3

5

8

13

21

34

**PERSONAL EXPERIENCE:**

Through this project, I learnt how the instruction cycle works (fetch and execute). I have a better understanding of IPC and interrupts. As far as implementing goes, this project required me to do a fair bit of research as I had never worked with the Java Runtime library. The hardest part of the project was setting up the communication between the two processes. Once that was done, setting up the switch statements for the instructions was tedious to do. Also debugging there instructions was quite a bit of a challenge. In the end, it was a great learning experience and has enhanced my knowledge of operating systems.