Project - 1
Summary

This project is having two different modules **Memory** and **CPU** which are running in parallel. CPU is responsible for the all the execution inside the processor and Memory is responsible for the handling of memory inside the processor. The approach I used to handle this situation is mentioned below.

Purpose of the Project:

This project was specifically designed for the understanding of processes and Inter-Process Communication. After completion of this project I have understand the working of processes, Creation of the child processes and Inter-Process Communication between two Processes.

Implementation of Project:

Language and Compiler Specifications:

The language I am using is **C++**. I am using **g++** compiler to compile my code. The functions I am using for process creation and Inter Process Communication are **fork()** and **pipe()** respectively.

Class Specifications:

In C++ I have used two classes. One for the **Memory** and Another for the **CPU**. Memory class is consist of basic memory operations **Memory Write** and **Memory Read**. CPU tasks are divided in two major parts.
Decode: which decodes instructions and read the all memory location related to the instruction.
Execute: this part will execute the instruction.

Approach:
Very first step toward any problem would be the analysis of the problem, so I did the same. First few hours I spent to analyze the problem. And as an outcome of that I designed the class structure mentioned above. This part took around **3 hours**.
Second step which I did was the being familiar with the fork() and pipe() commands. I did a simple program which fork a process and done **IPC** between them. This part I spent around **2 Hours.**

Now the next part was to develop the code. I started with the memory. I implemented memory class with two of its functions. After testing the memory part perfectly, I also developed and tested the CPU part. So I have two classes or modules working with together and producing output. This implementation took around **5 hours.**

Lastly I started to implement the final step, integrating the whole code. I divided this part in two steps.

1. Integrate the whole code as a single process and test the output of code
2. Create two processes and implement a pipe between them

In first part First I write a function named **init()** in memory to read from the sample.txt file and initialize the memory. And also called the memory functions directly inside the CPU. After the testing of this code I moved to the step 2 where I just need to add pipe for two different processes. This part take around **12 hours.** At the end of this stage I tested all the sample files uploaded on the e-learning.

Lastly I implemented the final step. Created two processes and using pipe made an IPC between them. The memory read and write part in CPU in above section replaced with the Read and Write of pipe. The protocol is as below.

To read memory send '1' followed by the address and to write into memory send '2' followed by the address and the data. (**Note: all this communication done here is in string. E.g. if I want to read memory 123. The string which is to be sent is "1123"**) .This part took around **3 hours**.

The last part was to implement the sample5.txt file. I have created the **Fibonacci series** for the first 12 numbers. It will print this series. This part took around **2 Hours.**

0 1 1 2 3 5 8 13 21 34 55 89