**SUMMARY**

Project Purpose

The purpose of this project was to simulate a computing system. We attempted to simulate a processor and memory within a computer system.

How the Project was implemented?

We created two processes:

1. Processor(parent)
2. Memory(child)

These two processes run at the same time thus simulating the concept of multiprocessing. These two processes were created by forking Memory process from the Processor process. The *fork()* method was used to create parent and child process.

Communication between the two processes was carried out with the help of two pipes. These pipes were created by the *pipe()* method. The two pipes used were:

1. toMemory: This pipe was used to send data by the Processor to the Memory
2. toProcessor: This pipe was used to send data by the Memory to the Processor.

A single pipe could not be used as there were concurrency issues. Ex- Consider that we kept a single bi-directional pipe. Sometimes, the Processor might send addresses on the pipe. Now, the memory is supposed to send address contents in response. However, before the memory could respond Processor might read the data on the pipe thinking that it is response sent by the Memory. But, this is the same that Processor wrote. This leads to an ambiguity and consequent fatal error. Hence, I had to resort to using two pipes.

I also tried to keep the code as modular as possible. These are the various steps I took to ensure it:

1. If the program needs to exit from any point in execution, it has to exit via the *exitParent()* function which sends an exit instruction to Memory process and then waits for it to end before exiting the Processor.
2. I defined a separate function for pipe communication. Every time, the processor wants to send data or receive data from the pipe, it calls the function to do it. All the *write()* and *read()* calls for the pipe are within this method.

My personal experience in doing the project

We implemented a rather small instruction set and did not have many features of common processors like PSW, interrupts etc. Hence, implementing this project turned out to be an easy job. Apart from taking care of the Call, Return and Jump instructions, the other instructions were fairly easy to simulate.

I used a single pipe in the beginning and faced the aforementioned sync issue in communication. But, I resolved that by using two pipes.

The most serious problems I faced were in writing the sample program. While trying to write a sample code, I found that the instruction set is very limited. Because of a missing compare instruction and missing PSW register, I could not implement a basic division on this processor. Also, basic multiplication required transferring the sum obtained at every step to memory and then retrieving it back. Also, decrement counter instruction was implemented only in Register X. However, to check whether this value reached zero or not we had to transfer counter to Accumulator and the use the JumpIfEqual instruction. This proved to be very tedious for the programmer. I felt that our project scope was limited. We should have included more instructions in our instruction set.

However, I understand that supporting many instructions was not the most essential purpose of our project. Our intention was to implement a multiprocessing simulation of a computer system and understand how the communication between two processes behaves. I think this project enabled me to understand both these concepts very well.