**OS Simulation**

**CS 3410-Section 2**

**Elizabeth Herndon**

**Payton Mock**

**Zach Comstock**

**Abstract**

The project that we will be discussed in this paper is the development, implementation, and testing of an Operating System simulation. The purpose of this project is to help students understand the fundamentals of what components make up an Operating System as well as the techniques that can improve the process speed and functionality of the OS. To help us understand what an Operating System is, we will be looking at functions such as memory management, multithreading, synchronization, and multiprocessing. Our final testing should provide us with enough information that we can conclude that certain scheduling processes and the size of the pages that we use has an impact on the overall performance of our system.

**Introduction**

This project focuses on the loading of a program file that will contain a list of 30 jobs all in hex that will then go through the process of traveling through the Operating System and will include interactions with hard disk, long term scheduler, RAM, short term scheduler, and finally execution on the CPU(s). In this project we will learn how the different modules of an operating system interact with one another and how they share resources. We will also learn how to load processes to the CPU and measure the efficiency of First In First Out and Priority scheduling. In Phase 2 we will also implement the Shortest Job First scheduling method and draw conclusions. We will also gather data on the effectiveness of paging and how the number of pages and frames improves or slows down our system’s performance. We have decided as to implement Paging into our Phase 1 and will be comparing the effectiveness of paging as a whole.

**Design**

The design approach that we took when developing our OS Simulator was to break down the system into several different modules that would interact and share data with one another as the program file was processed by the system. For our design we started with a Data Flow Diagram that helped us see how the program file moves through the system and what takes place in each module. The first step was to read in the file and save it to the Hard Disk. A PCB object would be created for each job and would contain important information such as the Program Counter and Job Length that we would need to have access to throughout the OS. The Long Term Scheduler then checks RAM to make sure there is room to push the next job. The jobs are then loaded into the Ready Queue by the Short Term Scheduler. The jobs will be sorted using either First In First Out, Priority, or Shortest Job First which we added for Phase 2 of the project. From there the Dispatcher will check the CPU(s) and locate one that is idle and push the job to that CPU. The final step in the system is for the CPU to decode the job and execute the instructions. The finished job is then placed on the terminate queue. We have also designed a Stopwatch module that is responsible for gathering the metrics that we want to measure such as Average Wait time, Average Execution Time, CPU usage, and overall total time that the system ran.

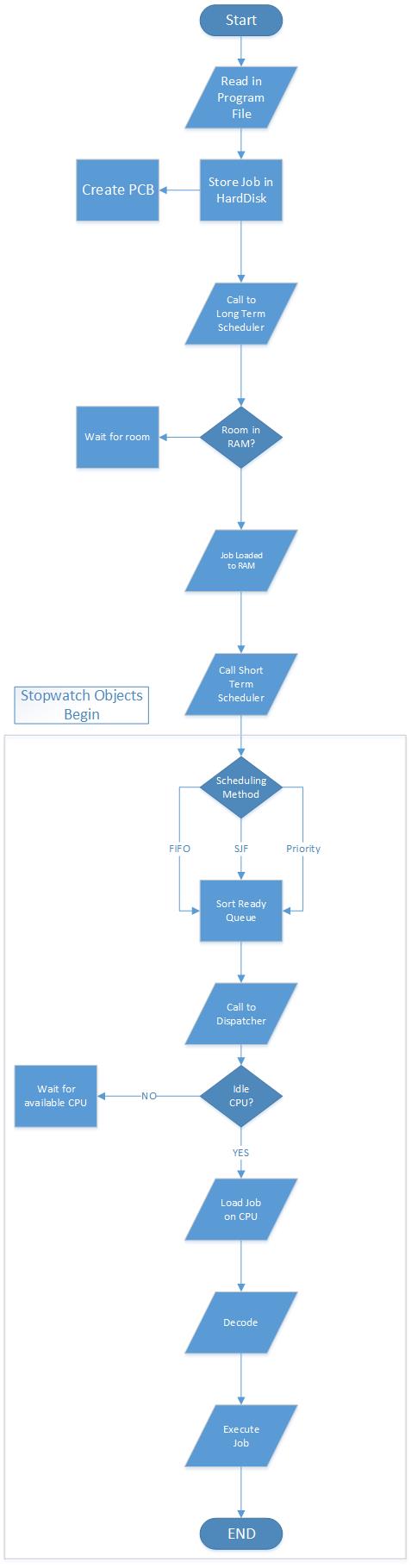


Figure 1 Operating System Flow Diagram

**Implementation**

****

Figure 2Operating System Class Diagram

**Simulation**

**Data & Results**

**Conclusion**

In conclusion, we have discovered that there are several components that make up an Operating System and that several of them impact the way the overall system runs. We have discovered that paging is a very important factor when it comes to RAM and increasing the number of pages and size of the pages can in fact enhance the performance of the system however the margin of improvement does not seem to be as large as we initially would have thought. We can also conclude that the scheduling algorithm also plays an important role in the overall speed with which the system performs but just like with paging, there doesn’t seem to be a very significant change. From our data we can conclude that there are ways to change the performance of the operating system in a way that we maximize the throughput however there exists a point where such enhancements can actually lead to decreases in the performance of the system. In closing, we have learned that there are several components that go into an operating system and that all those parts must be able to work together in order for the system to reach is maximum potential.