OS Simulator

Table of Contents

Contents

[1 Design Approach 3](#_Toc507166503)

[2 Implementation Modules 5](#_Toc507166504)

[3 Central Processing Unit 5](#_Toc507166505)

[4 CPU Module 5](#_Toc507166506)

# 

# Design Approach

At first, I gathered some information from the given information that the professor provided, and did a little bit of research as to how to design the OS simulator. I read through all of the specifications for phase 1 and finally came up with a design for the system.

Disk

Driver

MMU

Long Term Scheduler

Loader

Short Term Scheduler

CPU

Execute

Decode

Fetch

RAM

The main components that were identified in the phase 1 were the CPU, the short term scheduler, the long term scheduler, the MMU, the RAM, the Loader, the Disk, and the Driver. The main component that controls every aspect of the system and work’s as the engine of the system is the driver. From there they were other parts of the system that we needed to do which includes taking instructions from the instructions set and loads them onto the disk through the loader component. From there the disk that contains all of the information about the instructions set will be use by the long term scheduler and jobs will be loaded to the RAM. After jobs have been loaded to RAM, also known as the MMU gathers all of the jobs needed to be scheduled by the short term scheduler which then in turn utilizes the dispatcher to send jobs to the CPU. The CPU cycles through three parts of the system to fetch, decode, and execute until it has run out of instructions at which point , the OS terminates.

For the multiprocessing portion of the OS, I look more closely on the support for multiple cores, memory management among the processors, and the cache of each processor. In our design approach we took note of this and decided to implement the CPU module to handle all of these items as single processor and the using threads to handle multiple cores. Phase 2 of the project I redesign the approach to the CPU. The general architecture idea hasn’t really changed from phase 1. The Major changes came in the form of separating the Dispatcher into its own individual unit and placing it between the ShortTermScheduler and CPU modules, paging systems, and the cache.

adsad

Long Term Scheduler

RAM

MMU

Dispatcher

Fetch

Decode

Execute

CPU

Short Term Scheduler

Driver

Loader

Disk

RAM

# Implementation Modules

The hard components have been designing to be module by the an object oriented programming. These components represent real life hardware components that would be found on a regular computer. The OS components have been program using the C# language. Each module has its own way of acting to make the system work and simulate a real Operating System. Below are full descriptions of the modules that must integrate to make the system work.

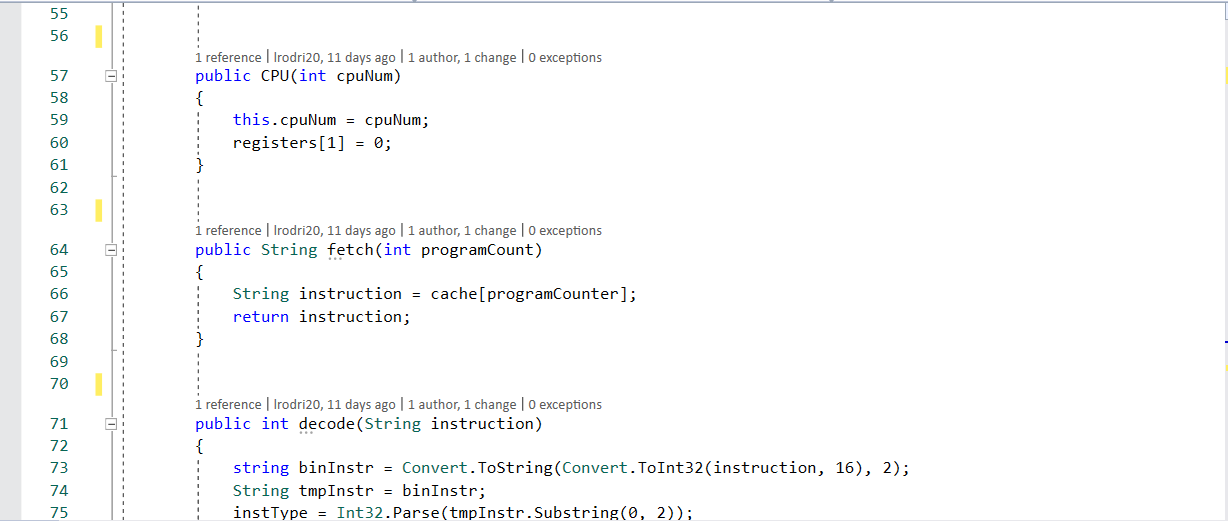
# Central Processing Unit

The Central Processing Unit has the purpose of controlling all module components in the system.

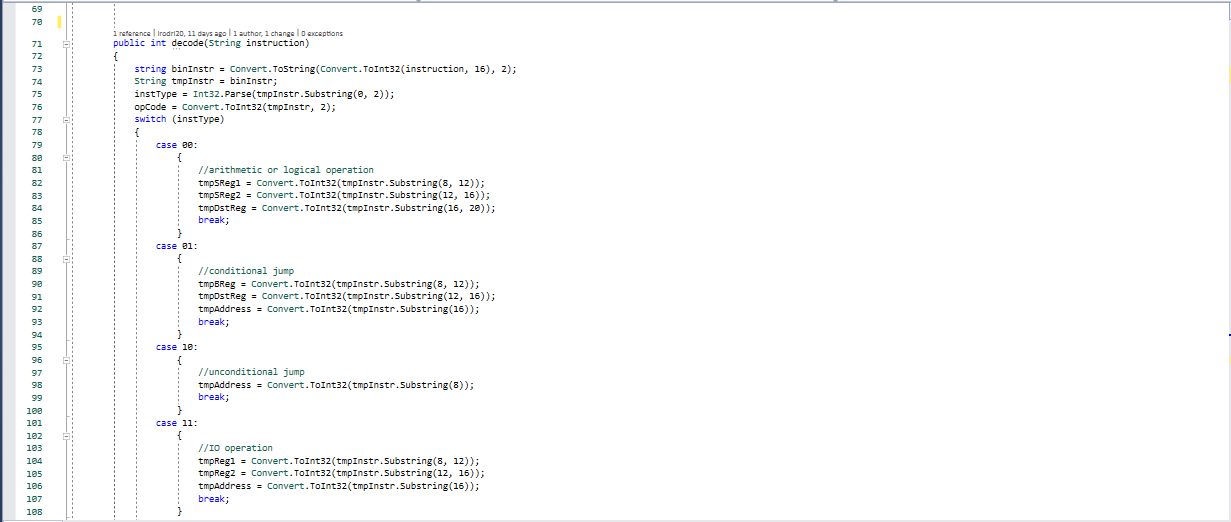
# CPU Module

Phase 1 contains the CPU that has several key methods that are integrate in the OS. The Fetch works with the disk and gathers what is the required information for the next instruction and increase the process counter value by 1 every single time, by retrieving the next instruction to be decoded upon the next fetch call. The decoder takes this call,instruction, and converts the hexadecimal instruction set into 32 bit binary sequence, separates the instructions into the correct types, and stores the organizes the proper information for the execution method. The execute method has a switch loop which switches to perform an action depending on what is has decoded from the decoded instruction set. When the executer finishes the job than the program will increase the counter by one. The CPU method named Effective Address handles all of the address translation. In phase 2 Dr.Bobbie require that we have multiple CPUs running at once, we accomplish this making a tread pool and executing every cpu at the same time. The OS simulator is capable of N-CPU and has been tested to with last about 12 CPUs since the memory of the RAM is not that big.

CPU Fetch

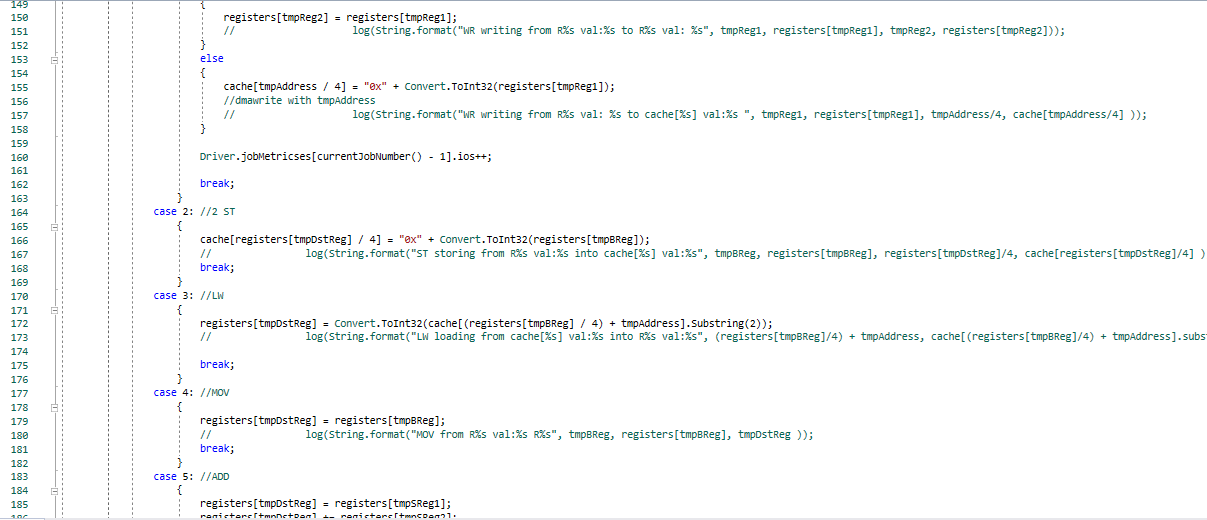


CPU decode



CPU Execute

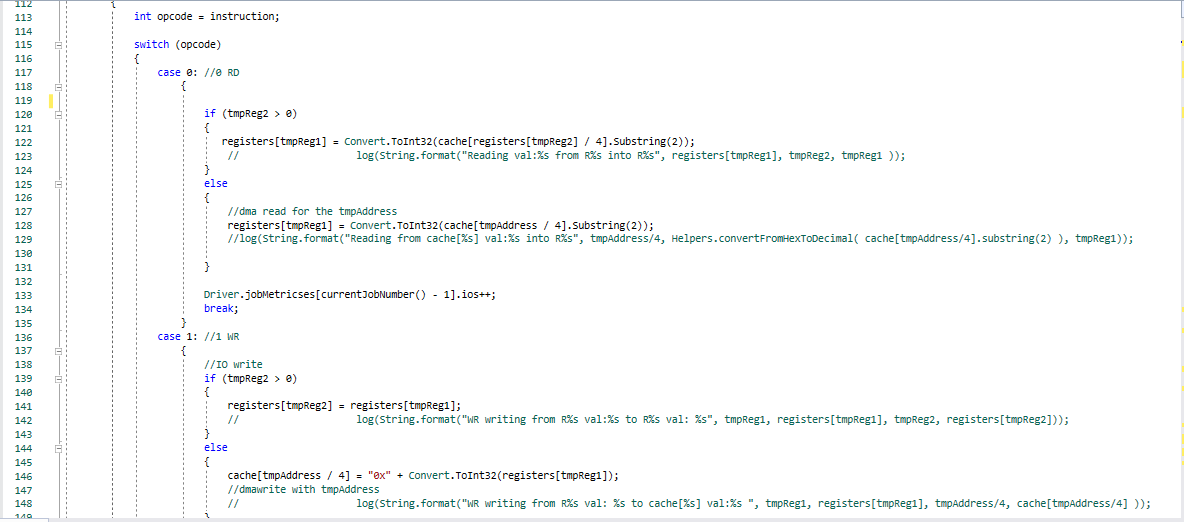




# DMA Hardware

This device receives all of the instructions that are in I/O reading or writing and make a call to the module that is not there and would be able to handle the reading and writing from different simulated I/O devices white the Driver handles the “compute only” functionalities. This process leaves the CPU from being executed with different process and it frees the time for the process by transferring the files and leaving the more computational time instructions that would be handled for a much faster time.

DMA Read/Write



# Scheduling

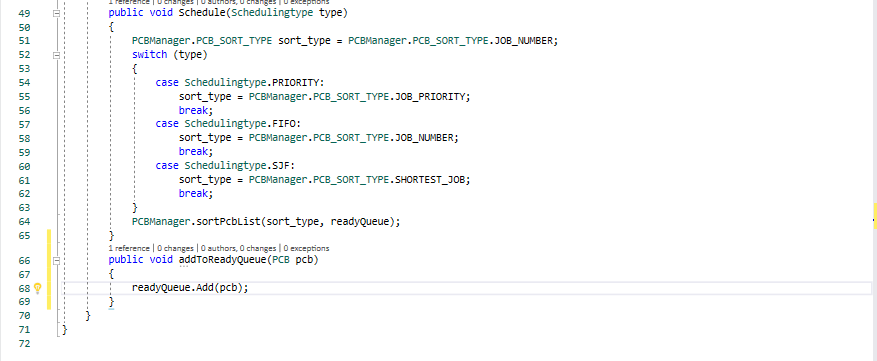
# Long Term Scheduler component

The long term scheduler loads all of the jobs in the ram to be ready to execute in the ready queue. The main part of the long term scheduler is to connect with PCBManager for data and sends everything off to be written in the RAM. When jobs have been completed in the RAM and removed. The RAM checks wheter they are more jobs in the disk to be completed, the Long Term Scheduler will send more job from the RAM as necessary.



# Short Term Scheduler component

The short term scheduler has the responsibility has the responsibilities of scheduling the jobs in the desire scheduling methods:First In First Out(FIFO), Priority Scheduling, or Shortest Job First(SJF). The Scheduling method changes between the calls from the dispatcher calls the jobs from the RAM to CPU. The component loads the jobs from the ready queue for the dispatcher. Short Term Scheduler calls the dispatcher according to the choice, then loads the CPU.



# Dispatcher

The Dispatcher component has the main responsibility of assign and loading jobs in the CPU. Since phase 2 required us to load the jobs into multiple CPU cores, the dispatcher was modified. The dispatcher is responsible for setting jobs that have been mark as completed and mark them as terminated state, load the correct jobs in the idle CPU’s , and send a signal to the CPU and let the module know that all jobs have been completed. The dispatcher runs the PCB module and loads the necessary jobs onto the CPU while gathering the metric data for each job.

Dispatcher Terminate Jobs



Dispatcher Completed Jobs

# 

# Memory System

Below are the components that make up the filing system for the operating system. These components are to contain the data that runs in the operating system.

# Disk Hardware

Disk Hardware component in this software system is a virtual model disk that contains all the information. Disk contains a string of array to contain all the information in the hard drive.



# Page Component

The page component contains all of the pages in the system for paging. It contains the sizes of the pages, the information of the pages, and functions to write to the page.



# RAM Component

The virtual component representing the RAM where the processes are loaded into. The RAM has an array of strings that are represented in 4 bit words which has similar structure of the Disk Module. The RAM module will interact with the MMU such that no other similar modules interact directly with it.



# MMU component

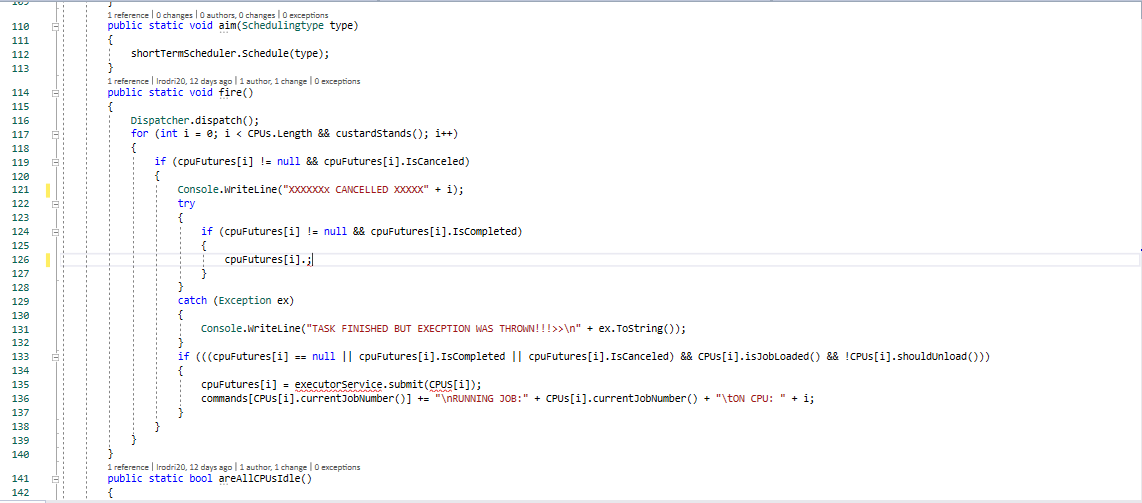
MMU stands the memory management unit. The RAM and other modules communicate with component, and this component makes it easier for the components to communicate.

MMU read/write



# Driver Component

Driver component has the procedures. This modules calls all of the modules to function correctly. The first thing that the module does is it calls the scheduler , which can then turn the dispatcher. The driver loops until no more instructions remain in the ready queue. The compute only in the run function.



# Loader Component

The purpose of the loader component is to load the text file containing all of the instructions and loads the data in the disk. The loader passes the data into the PCB manager with the information for the processes, and insert the instructions in the RAM.

Loader Start()

