**First assignment – Shark machine Operating System**

Neta Shiff

Nova Southeastern University

CSIS 3810: Operating Systems Concepts

Professor Gregory Simco

October 19, 2022

**Table of Contents**

Title Page 1

Table of Contents 2

Introduction 3

Purpose 3

Deliverables and Expectations 4

Solution Approach Overview 4

Methodology 5

Goals 5

Strategies 6

Constraints and Technical Considerations 6

Reliability and Validity 6

Activity leading to Design 4

Design 3

Scope/Overview 6

Data Design 6

System Architecture Design 6

Object and Class Module Design 6

User Interface Design 6

Test Provisions 6

System’s Output 7

Neta Shiff

CSIS 3810

Prof. Gregory Simco

19 October 2022

Design Programming project Assignment 1 – Shark machine Operating System

**Introduction:**

1. Purpose:

The purpose of this project is to demonstrate a simulation of a multi-tasking Shark Machine operating system where array is prime memory design and develop a high-level program using JAVA

1. Deliverables and Expectations

* **Language** – the expectation is to have a functioning language according to the assignment where each element represents a machine word.
* **The operations-** The system should be able to execute the following operations: add, sub, fetch, load, store, branch, conditional branch
* **Memory**- the system will have a 1024 string array which will contain the prime memory and the registers which will be used in the program.
* **PCB Process Control Block**- Every job will have its own PCB which will be connected in a link list. The PCB will contain the data of the registers.
* **Program scheduler and holder**- The system will have link list which will allow to create PCB objects based on the files we received, and organize them in order.
* **Round Robin algorithm** – The program will use round robin algorithm to change between the jobs according to which one taking too much time.
* **Multi-tasking capability**- the system should be able to get several numbers of files and activate all of them using round robin algorithm according to the order on the name of the file.
* **Switch cases-** the system use switch case to decode the string to the function which will be used.

1. General Solution Approach Overview

The Shark Machine operating system is a High-level program in JAVA that simulate the operation of a basic machine system. The system is built as two classes a main multi-tasking class with the process and a PCB class for the specific job. The project uses an array for the memory, and a link list for getting the files and organizing them according to the order and to execute the job using the round-robin protocol program. Each object of PCB in the link list will have its own registers, filename, time, and other variables.

The files will import from the directory, and their order will be according to the numbers on the filename. For example, for the file name “text\_32.txt” the number will be 32. The project uses a linkList to load the files and organize the order as mentioned; while loading the files to the linkList as PBC, the program will go over the file, read each line and place it in the memory starting from place 512 of the array. Each line of code will be considered as one unit of time. After a certain amount of time, in my code, it counts to seven, the program will use a round-robin algorithm and the LinkedList to move a job which had seven lines executed, to the last placement in the list. By synchronizing the registers between the two classes, the system should be able to create the Shark Machine language, run the system, store the data, and execute the text files we entered.

**Methodology:**

1. Goals:

The main goals of the Shark Machine Operating System are to learn the concept of multi-tasking and better understand the project's framework. Those main goals will be accomplished in smaller steps on the way. This program began with the goal of creating a single operation to work. Therefore, the first minor goal was to develop the primary array for the memory. After this was accomplished, the goal was to read a single file into the memory array. Then, the next step was to create different operations such as add, sub, and more. After achieving a single process to work, the multi-tasking process began. The first goal was to create the PCB class and its functions, constructors, and variables. The second goal was to create the load method, which included reading the files and entering the lines into the memory array, creating an instance of PCB, and entering the object into a linkList. The third goal is to create a execute function that will go over the memory and call the proper functions; after that, the step was to create a round-robin algorithm.

1. Strategies:

Different strategies were used to achieve the goals of the project. After multiple tries, which failed to multitask with a queue or to find a way to organize the linkList in order, the most appropriate data structures for writing and coding this specific code were found. For each one of the goals mentioned in the previous section, there are multiple ways of approach. The first minor goal is developing the primary array for the memory was written in the assignment instruction to use an array of 1024; the reason it is a string array is the use of string for the description of the operations such as "ADD." The second goal of reading a single file into the memory array. BufferedReader can read each line separately; this way is easier to enter the memory line by a line using a while loop and a pointer which starts from the second half of the memory. This way, the next part, which involves the function, already has the base for the load function. The third step, creating operators, the classes, and the explanation in the instruction, helps. While creating the PCB class, the goals and the use of the class were clear. Therefore, we needed a timer, file name, and all of the registers. The functions were created based on need. The load method was the most complicated part for me. The problem which accrued at the moment is entering the files into the ArrayList using "ArrayList.add(new PCB(time, place, filename)" the problem is that in some of the cases, the array list will contain just the last instance, which added for each one of the places. It caused the memory not to go over all of the files. I started the function using a for loop, which goes over all the files in the directory, making sure they are text file; then it create an instance of PCB object and adds it to the linkList. After the addition, we use the file reading developed previously. After reading the files, I could find the number that organized the PCB jobs by order. The last goal was to have an easy "line"; the execution priority is essential; therefore, utilizing the round-robin helped to accomplish it efficiently.

1. Constraints and Technical Considerations:

The strategies used require some constraints and technical considerations. The main constraint we have in this program is the memory array size. The size is 1024, and this divides into two parts. Half is for the data we will operate, and the other 512 indexes are for the operators. The array size restricts the number of lines of code that will operate to 511, including yield operations. Therefore, it won't be possible to load more than 64 instructions.

1. Reliability and Validity:

I am excited to see the project executed. After changing certain from an ArrayList for only Linklist use, I can say that the operating system is working successfully. The design is reliable. The program is able to get several files from the folder located, load them to the link list, organize them by the number after the underline and execute them. The system can use the different operations implemented, use registers, read the data from memory properly, and writes it into the data. At first, The Shark Machine Operating System works well except for some instances.

1. Activity leading to the Design:

**Design:**

1. Scope/Overview:
2. Data Design:
3. System Architecture Design:
4. Object and Class Module Design:
5. User Interface Design:-

In this project we didn’t create a user interface. The program run using the terminal. The user can chose if he want to see the final memory and the final registers, before they are deleted.

1. Test Provisions

Systems’ Output:

Welcome to OS machine:

System initializing

reading files in the folder:

another directory - not reading

another directory - not reading

another directory - not reading

another directory - not reading

another directory - not reading

another directory - not reading

another directory - not reading

The files list is:

test\_1.txt, ar\_2.txt, test\_3.txt, test\_10.txt, Te\_11.txt, test\_50.txt, test\_55.txt, test\_60.txt}

running the system

test\_1.txtThe program had 7 operations already it is YIELD

ar\_2.txtThe program had 7 operations already it is YIELD

test\_3.txtThe program had 7 operations already it is YIELD

test\_10.txtThe program had 7 operations already it is YIELD

Te\_11.txtThe program had 7 operations already it is YIELD

test\_50.txtThe program had 7 operations already it is YIELD

The file test\_55.txt finish all of his operations

test\_60.txtThe program had 7 operations already it is YIELD

test\_1.txtThe program had 7 operations already it is YIELD

The file ar\_2.txt finish all of his operations

The file test\_3.txt finish all of his operations

test\_10.txtThe program had 7 operations already it is YIELD

Te\_11.txtThe program had 7 operations already it is YIELD

The file test\_50.txt finish all of his operations

The file test\_60.txt finish all of his operations

The file test\_1.txt finish all of his operations

The file test\_10.txt finish all of his operations

The file Te\_11.txt finish all of his operations

Total Time: 95

The memory is:

CELL 0: 100

CELL 1: 150

CELL 10: 40

CELL 11: 17

CELL 12: 1

CELL 13: 16

CELL 14: 24

CELL 50: 20

CELL 51: 10

CELL 52: 1

CELL 53: 10

CELL 54: 6

CELL 55: 30

CELL 56: 50

CELL 100: 8

CELL 101: 11

CELL 102: 10

CELL 103: 13

CELL 150: 169

CELL 151: 50

CELL 200: 30

CELL 201: 40

CELL 202: -20

CELL 203: 50

CELL 204: 150

CELL 400: 2

CELL 401: 200

CELL 509: 40

CELL 510: 10

CELL 511: 40

CELL 512: LDI 2

CELL 513: STR 400

CELL 514: LDI 202

CELL 515: SUB 400

CELL 516: STR 401

CELL 517: CBR 544

CELL 518: BRH 538

CELL 519: LDI 400

CELL 520: STR 402

CELL 521: LDI 500

CELL 522: STR 403

CELL 523: YLD

CELL 524: LDI 30

CELL 525: STR 200

CELL 526: LDI 40

CELL 527: STR 201

CELL 528: LDI 50

CELL 529: STR 202

CELL 530: LDI 0

CELL 531: ADD 200

CELL 532: ADD 201

CELL 533: ADD 202

CELL 534: STR 203

CELL 535: LDI 100

CELL 536: ADD 203

CELL 537: STR 204

CELL 538: YLD

CELL 539: LDI 20

CELL 540: STR 50

CELL 541: LDI 10

CELL 542: STR 51

CELL 543: LDI 1

CELL 544: STR 52

CELL 545: LDI 10

CELL 546: STR 53

CELL 547: LDI 6

CELL 548: STR 54

CELL 549: LDI 30

CELL 550: STR 55

CELL 551: LDI 50

CELL 552: STR 56

CELL 553: LDI 8

CELL 554: STR 100

CELL 555: YLD

CELL 556: LDI 100

CELL 557: STR 150

CELL 558: LDI 50

CELL 559: STR 151

CELL 560: LDA 200

CELL 561: SUB 151

CELL 562: STR 202

CELL 563: YLD

CELL 564: LDI 40

CELL 565: STR 511

CELL 566: LDI 50

CELL 567: SUB 511

CELL 568: STR 510

CELL 569: ADD 200

CELL 570: STR 509

CELL 571: YLD

CELL 572: LDI 100

CELL 573: STR 0

CELL 574: LDI 50

CELL 575: ADD 0

CELL 576: STR 1

CELL 577: YLD

CELL 578: LDI 40

CELL 579: STR 10

CELL 580: LDI 17

CELL 581: STR 11

CELL 582: LDI 1

CELL 583: STR 12

CELL 584: LDA 11

CELL 585: SUB 12

CELL 586: STR 13

CELL 587: LDA 10

CELL 588: SUB 13

CELL 589: STR 14

CELL 590: YLD

CELL 591: LDI 11

CELL 592: STR 101

CELL 593: LDI 10

CELL 594: STR 102

CELL 595: LDI 13

CELL 596: STR 103

CELL 597: LDI 0

CELL 598: ADD 50

CELL 599: ADD 51

CELL 600: ADD 52

CELL 601: ADD 53

CELL 602: ADD 54

CELL 603: ADD 55

CELL 604: ADD 56

CELL 605: ADD 100

CELL 606: ADD 101

CELL 607: ADD 102

CELL 608: ADD 103

CELL 609: STR 150

CELL 610: YLD

CELL 611: END

The registers are:

the ACC is :169

The PSIAR is: 610

The SAR is: 150

The SDR: 169

The TMPR: 150

The CSIAR is: 0

The final total time is: 95

exiting the system

finish