Homework 2

Student Name	Muhammed
Student Surname	Oguz
Student Number	1801042634

Content

Understanding The Basics

First of all, I should learn how to virtual memory works. Thanks to the OS lecture and my final preperation, I already familiar with it and I already familir with page replacement algorithms in advance.

Also, I use some online content to understand some context:

LRU: https://youtu.be/u23ROrlSK_g

FIFO: https://youtu.be/16kaPQtYo28

Compiling and Running

Since I use WSL (Windows Subsystem for Linux) as my main development enviroment, I had to figure out how to do this with it.

After some research, with downloading those given apps, Enviroment was ready to go.

```
sudo apt-get install grub-common
sudo apt-get install libisoburn-dev
sudo apt-get install xorriso
sudo apt-get install grub-pc-bin
```

But unfortunately, using `make run` command to execute and running the OS is not possible with this setup.

For avoiding this, I changed my makefile a little bit to work on.

I have to compile and convert to ISO file and manually select this file from VM.

Modified makefile part

```
muo: mykernel.iso
  mv mykernel.iso /mnt/c/Users/Muhammed/Desktop/testIso
```

This command, first creates ISO file and moves to my desktop to select easily from my VM.

I commented this part when delivered my homework.

What does this OS do?

I watched most of the videos of given YouTube series.

This OS mainly have a very basic driver, GUI and very basic multitasking functionality.

Also has some keyboard and mouse interrupts functionality,.

Those things could be helpful while implementing this multi threading functionality.

How Virtual Paging and Pace Replacement Works

In order to achieve special address space for every process, we should separete our addresses to handle correctly for proceudres.

For achiving this, there is a concept that called virtual memory and there are some algorithms for implementing this.

Every page replacement algorithm tries to achieve to `Optimal Algorithm` but it is impossible.

So, Operating System experts tries to implement every page replacement algorithm and calculates which one is near to optimal algorithm.

This is very special case in Computer Science.

FIFO

How FIFO Works

Fifo is very simple. It removes first page when page is full and goes with the next one.

How I Implemented

I simple keep a counter. I replaced page with keeping this counter as index. In every run it will be incremented and when it reaches the border, It moduled to starting position.

Second Chance

How Second Chance Works

Second chance works like fifo but with checking R and M bit. If R and M bit both zero, it replaces it immediatelly. If every R bit is 1, It works like normal FIFO

How I implemented

I use the same metodoloy, It checks every page till find a R bit with zero. If it reaches to the end It works like normal FIFO

LRU

How LRU Works

It uses the access system. It cheecks the refarence time. If a page has the latest refarence it removes.

How I Implemented

I keeped a accessed information on my struct. Since this homework is a simulation. I defined access information with some operations and get the least value when it times to replacement.

Implementing The Homework

There are 2 main parts and each one has 3 little parts.

- Implementing Page Replacement Algorithms
  - FIFO
  - Second Chance
  - LRU
- Implementing Sorts Algorithms
  - Bubble
  - Insertion
  - Quick

General Implementation

WYOOS (Write your own operating system) lecturer gives the fundamental code to start to the homework. Also, I use my HW1 part for this homework.

`pagereplacement.h`

This is the essantial part of my homework. It has necessary classes and definition.

PAGE COUNT Constant

```
{
#define PAGE_COUNT 10
```

Initially I have 10 page count.

Page Struct

```
typedef struct
{
  int data;
  int lastAccessed;
  int rBit;
  int mBit;
} Page;
```

It bassically, keeps a integer data. Has a lastAccessed information (used from LRU) and R and M bit.

R bits used from Second Chance. Also M bits used from Second Chance but since it is a simple simulation, M bits has don't meaningfull existance.

FIFO Class

```
class FIFO
{
private:
  Page pageTable[PAGE_COUNT];
  int current = 0;
  int getFirstIndex();

public:
  FIFO();
  // get a sort function as parameter and inputArr
  void run(int *inputArr, int inputSize, int (*sort)(int *, int), char *msg);
  int pageHitCount = 0;
  int pageMissCount = 0;
};
```

This class has a number of pages as defined above.

It has a public function that callled run. It takes array to input and its input size. It also takes a sort function as desired in PDF.

SecondChance Class

```
class SecondChance
{
private:
  Page pageTable[PAGE_COUNT];
  int current = 0;
  int getFirstIndex();
  int getFirstRMZeroIndex();

public:
  SecondChance();
  // get a sort function as parameter and inputArr
  void run(int *inputArr, int inputSize, int (*sort)(int *, int), char *msg);
  int pageHitCount = 0;
  int pageMissCount = 0;
};
```

It has a function called `getFirstRMZeroIndex` that finds first page with R and M values are zero and goes other one. If it can not found any like this, calls `getFirstIndex` function since it behaves like FIFO if it is necessary.

It has a public function that callled run. It takes array to input and its input size. It also takes a sort function as desired in PDF.

LRU Class

```
class LRU
{
private:
  Page pageTable[PAGE_COUNT];
  int current = 0;
  int getLastAccessedIndex();

public:
  LRU();
  // get a sort function as parameter and inputArr
  void run(int *inputArr, int inputSize, int (*sort)(int *, int), char *msg);
  int pageHitCount = 0;
  int pageMissCount = 0;
};
```

It has a `getLastAccessedIndex` function. It uses lastAccessed information in PAGE structure. Choes to lowest one and replaces with it.

It has a public function that callled run. It takes array to input and its input size. It also takes a sort function as desired in PDF.

PageReplacement Class

```
class PageReplacement
{
public:
  PageReplacement();
  FIFO fifo;
  SecondChance secondChance;
  LRU lru;
};
```

It has this tree classes objects.

`kernel.cpp`

I modified this part.

I add sort functions and modified keyboard event handler to take input from user.

I also keep my random array in `kernel.cpp file`

SORT FUNCTIONS

```
int bubbleSort(int arr[], int n)···

int insertionSort(int arr[], int n)···

int partition(int arr[], int low, int high, int *sortTime)      You, 1
int quickSortHelper(int arr[], int low, int high)···

int quickSort(int arr[], int n)···
```

> 💡 Notice that, every sort function takes an array and size. and returns an integer. This return value is the time information that I found a solution to include time information.

Using KeyboardEventHandler

This one is very simple since I only call run functions with desired sort functions and messages.

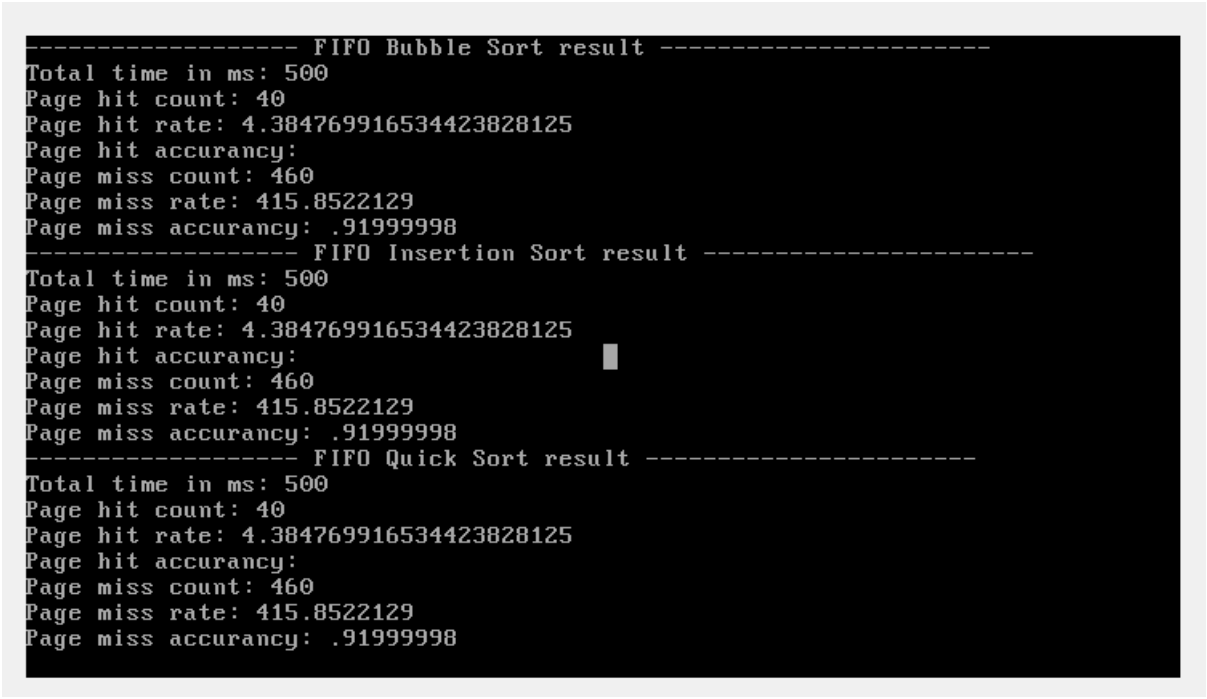
```
class PrintKeyboardEventHandler : public KeyboardEventHandler
{
public:
  void OnKeyDown(char c)
  {
    char *inputBuffer = " ";
    inputBuffer[0] = c;

    if (inputBuffer[0] == '1')
    {
      printf("\nFIFO Results\n");
      page.fifo.run(randValues, 1000, bubbleSort, "\nBubble Sort result");
      page.fifo.run(randValues, 1000, insertionSort, "\nInsertion Sort result");
      page.fifo.run(randValues, 1000, quickSort, "\nQuick Sort result");
    }
    else if (inputBuffer[0] == '2')
    {
      printf("\nSecond Chance\n");
      page.secondChance.run(randValues, 1000, bubbleSort, "\nBubble Sort result");
      page.secondChance.run(randValues, 1000, insertionSort, "\nInsertion Sort result")
      page.secondChance.run(randValues, 1000, quickSort, "\nQuick Sort result");
    }
    else if (inputBuffer[0] == '3')
    {
      printf("\nLRU\n");
      page.lru.run(randValues, 1000, bubbleSort, "\nBubble Sort result");
      page.lru.run(randValues, 1000, insertionSort, "\nInsertion Sort result");
      page.lru.run(randValues, 1000, quickSort, "\nQuick Sort result");
    }
  }
};
```

Running and Results

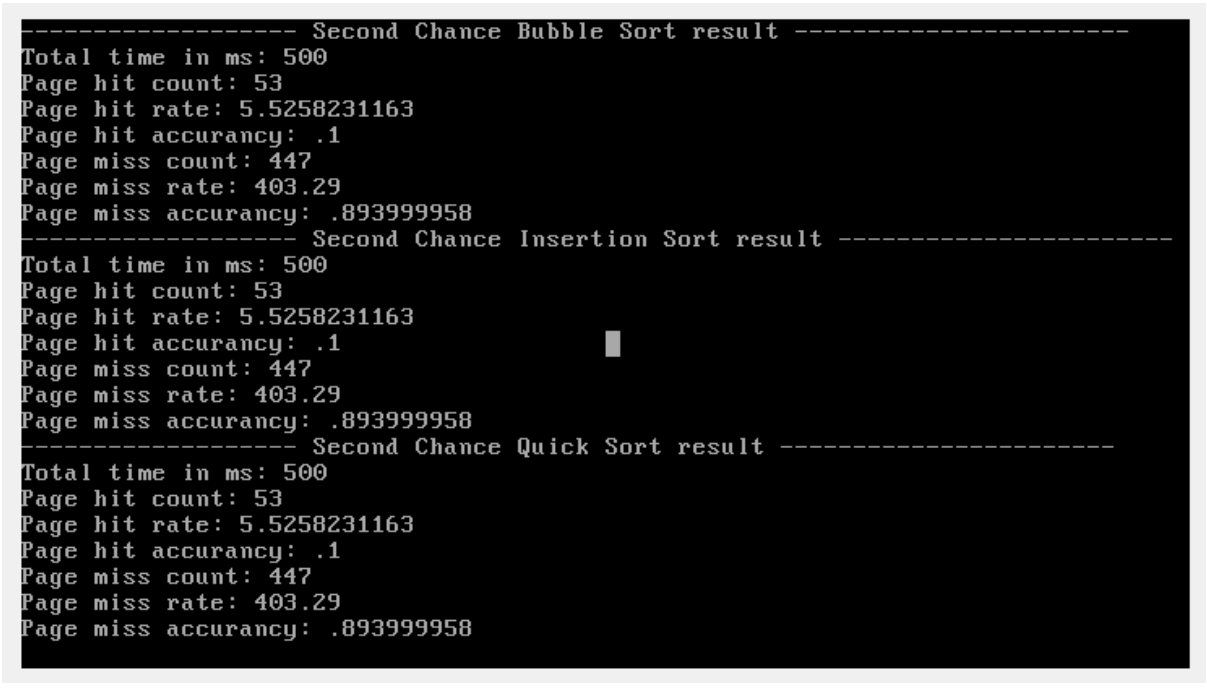
Please be aware of, I tried to show accurancy and some additional information. But sometimes this information leads wrong results unexpectedly. Since There is no easy way to print doubles on screen on this operating system.

FIFO Results



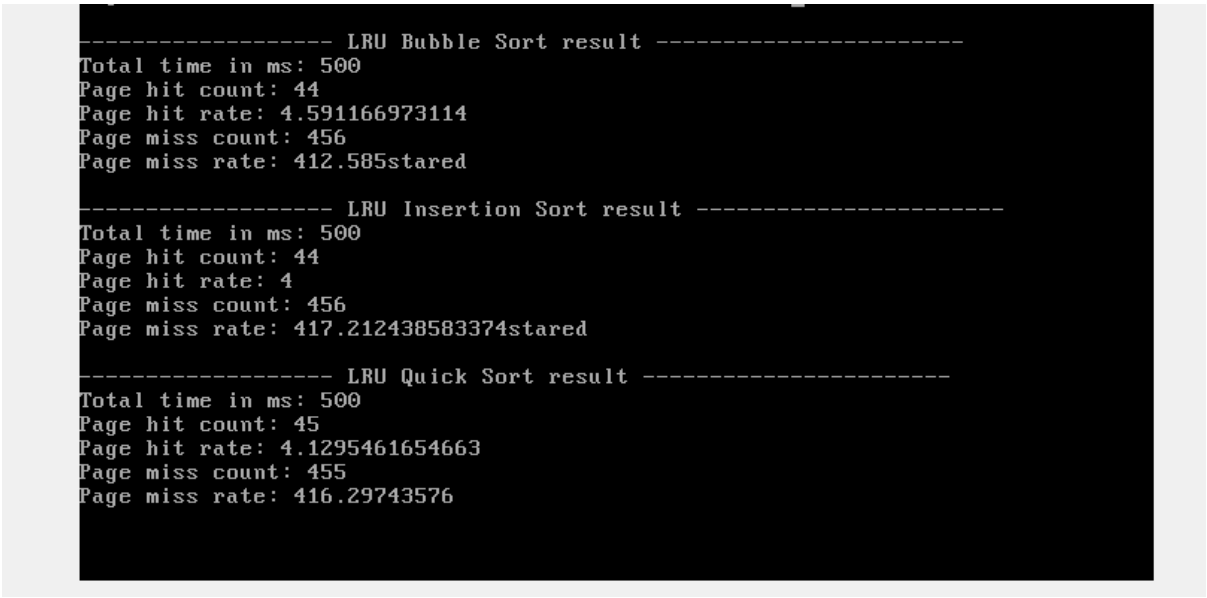
As shown above, It produces different results with different sort functions.

Second Chance



As shown above, It produces different results with different sort functions.

LRU



As shown above, It produces different results with different sort functions.

Problems That I Encountered

Implementing LRU was a bit thoug. Sınce I had to figure out how can it hold and behave like there is a referance time to on a page. I handled it eventually.