Virtual Memory Simulation

ITRW316 Opdrag 2B

A brief discussion about concepts relating to Memory Management, methods used in memory Abstraction and virtual memory, and a description of a program designed to simulate how virtual memory works using the Least Resent Used or Not Frequently Used algorithm.

2019

Markus Madeleyn 27211452

Divan van Kruiselbergen 26378396

4/15/2019

Table of Contents

[Introduction 2](#_Toc6235927)

[Memory Management 2](#_Toc6235928)

[Virtual Memory 3](#_Toc6235929)

[Program 3](#_Toc6235930)

[Execution 4](#_Toc6235931)

[Image 1. 4](#_Toc6235932)

[Global or Local page Allocation 7](#_Toc6235933)

# Introduction

In this report we discuss a program written to simulate the process of virtual memory but before we can start we need to understand why virtual memory is used. Therefore we wil have a brief discussion on Memory Management and all the concepts in contains. We wil discover what led to the concept of virtual memory and some older methods that was used to run consecutive programs at the same time. After that the simulation program wil be discussed in detail.

# Memory Management

Main memory (RAM) is an important resource which should be carefully managed. Since Memory sizes are rapidly increasing program sizes are increasing faster which leads to programs that cant fit into RAM memory, also in computers today it is vital to be able to run two programs at the same time which rises allot of problems for operating systems as wel as hardware.

The ideal computer would be infinite in size and speed as wel as nonvolatile. However this is not yet possible and an alternative rout had to be taken which led to the idee of a Memory Hierarchy. This means a computers memory can be divided into tree main categories, first the computer has a few MB of very fast, expensive, volatile cache memory, a few GB of medium speed, medium priced volatile main memory, and a few TB of slow, cheap, nonvolatile magnetic or solid state dick memory.

Using this hierarchy we can send programs that need to execute to the fast main memory to ensure fast execution while the programs that not currently waiting to execute can be placed in the slow permanent memory. It is the job of the operating system to abstract the hierarchy into a useful model and then manage the abstraction.

# Virtual Memory

There al multiple ways of abstracting memory. One way is to do it without abstraction but this method does not work when multiple programs needs to run at the same time. Other way is to abstract memory using base and limit registers with dynamic relocation. This way does not work when the RAM needed by the processes is larger that the physical memory. Swopping can be used however it can become quite complicated and doesn’t account for bloatware. (Bos and Tananbaum 2015)

Virtual memory is an airtight way of abstracting memory and is widely used in operating systems today. Virtual memory is based on the idee of breaking programs into multiple runnable peaces called overlays. Programs has there own address space which is broken up into chunks called pages that contains a continuous range of addresses. The pages are then mapped into physical memory but not all pages as to be in physical memory al the same time. The programmed-generated addresses are called virtual addresses and vorm part of the virtual address space. When virtual memory is used the addresses is not directly send to the memory buss, it first goes to the MMU(Memory Management Unit) where it is mapped to physical memory

# Program

The program we wil now discuss is written to simulate the working of virtual memory, paging and all it contains using the Least Recently Used or the Not Frequently Used algorithm. The program contains the following classes in order to achieve simulation:

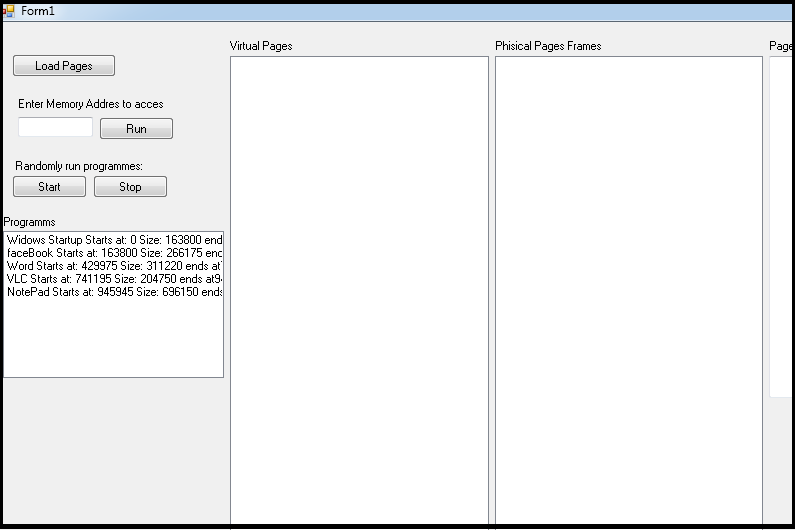
* SimProgram Class – Witch is an abstraction of a program with attributes: program name and program size. The attributes is merely for simulation and doesn’t represent the real programs.
* Page Class – Witch represent the chunks of programs witch wil be swopped in and out of main memory and hase the following attributes: page number, start adres, and an attribute that keeps count of how many times the page was use called the used attribute. This attribute is implemented in order to achieve the chosen algorithm(LRU or NFU) .
* VirtualAddressSpace Class – Witch represents the virtual address space or the permanent memory. All of the Page objects that is created when the programs is broken into page wil be stored in an object of this class. Attributes are only a list of the Page Class Objects.
* RAM Class- Witch represents the main memory from where pages wil be run. Attribute is a Array of Page class objects which can be described as page frames. RAM only contains 99page Frames in the array witch can be exchanged with any of the Page objects saved in the VirtualAddressSpace Class.

The program also contains a Form from witch you can observe the process of virtual memory and olso direct it to access a memory adres of your choice.

## Execution

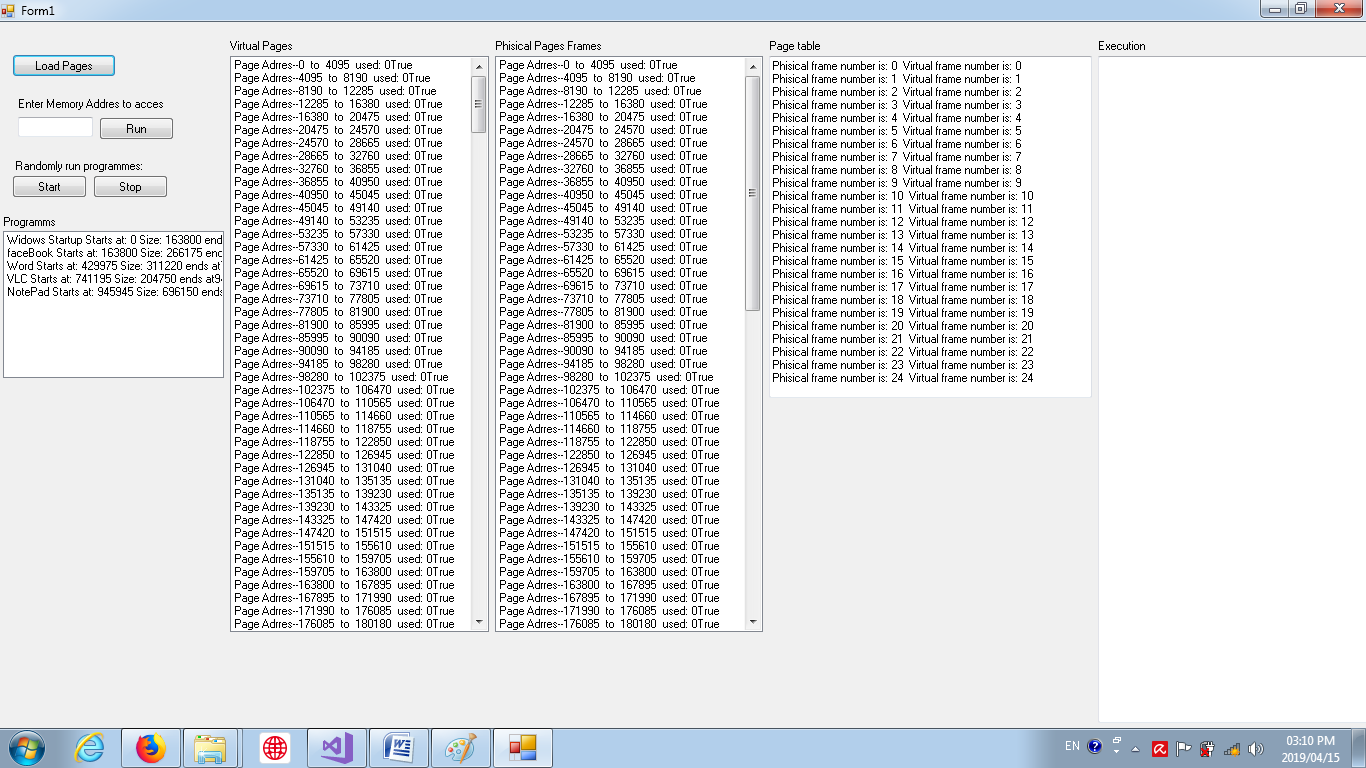
When the form is loaded it immediately creates 5 simProject class objects which acts as the 5 programs ouer simulation program wil consecutively run. The first list box is filled to display the 5 simulation programs. After the programs are loaded they are broken up into pages (Page Class objects) and loaded into the virtual address space as a list of pages. Also the RAM wil be loaded initially with the first 99 Page objects in the Virtual Address space list, and the present bid wil be set to 1 (or true in this case because of simulation purposes). The Form wil initially look as followed.

### Image 1.



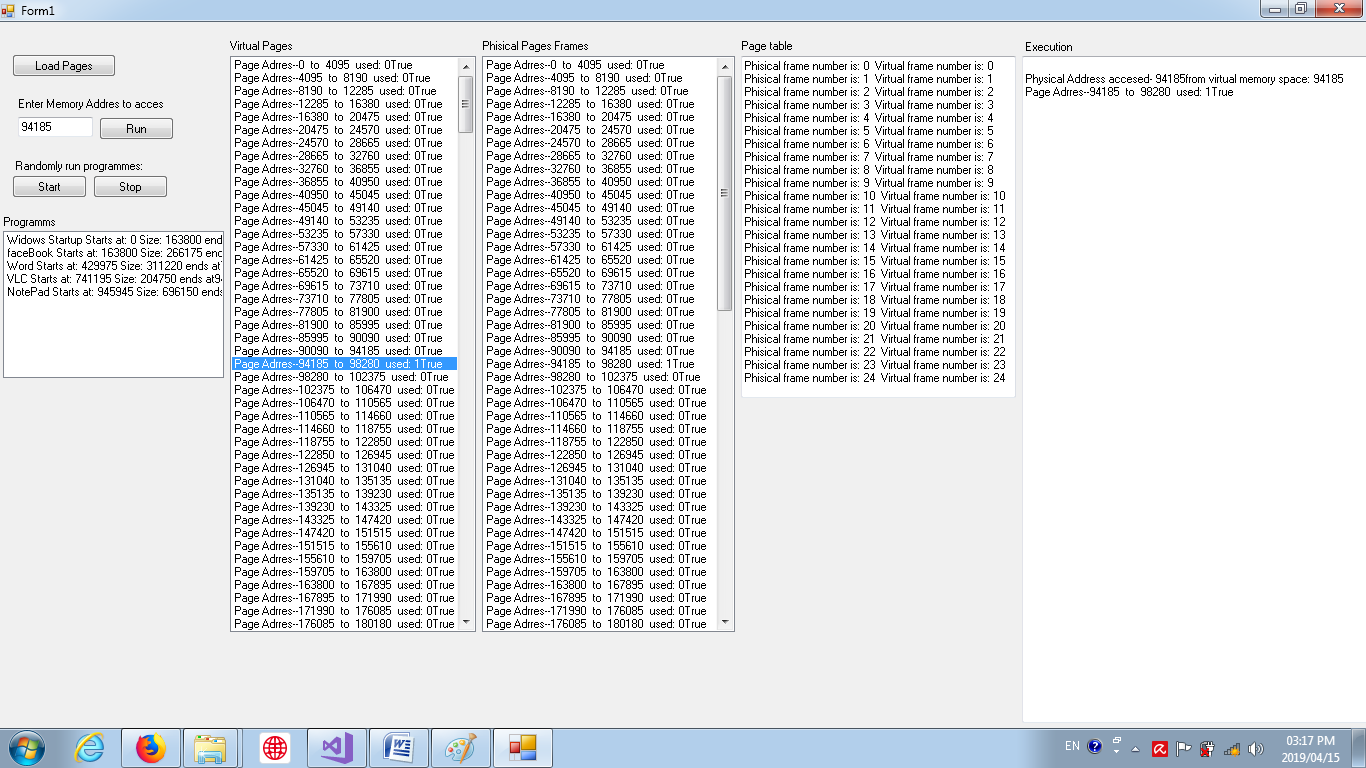
When the load pages button is pressed the Virtual pages and physical page frames list boxes as wel as the page table textbox wil be displayed as followed.

### Image 2:



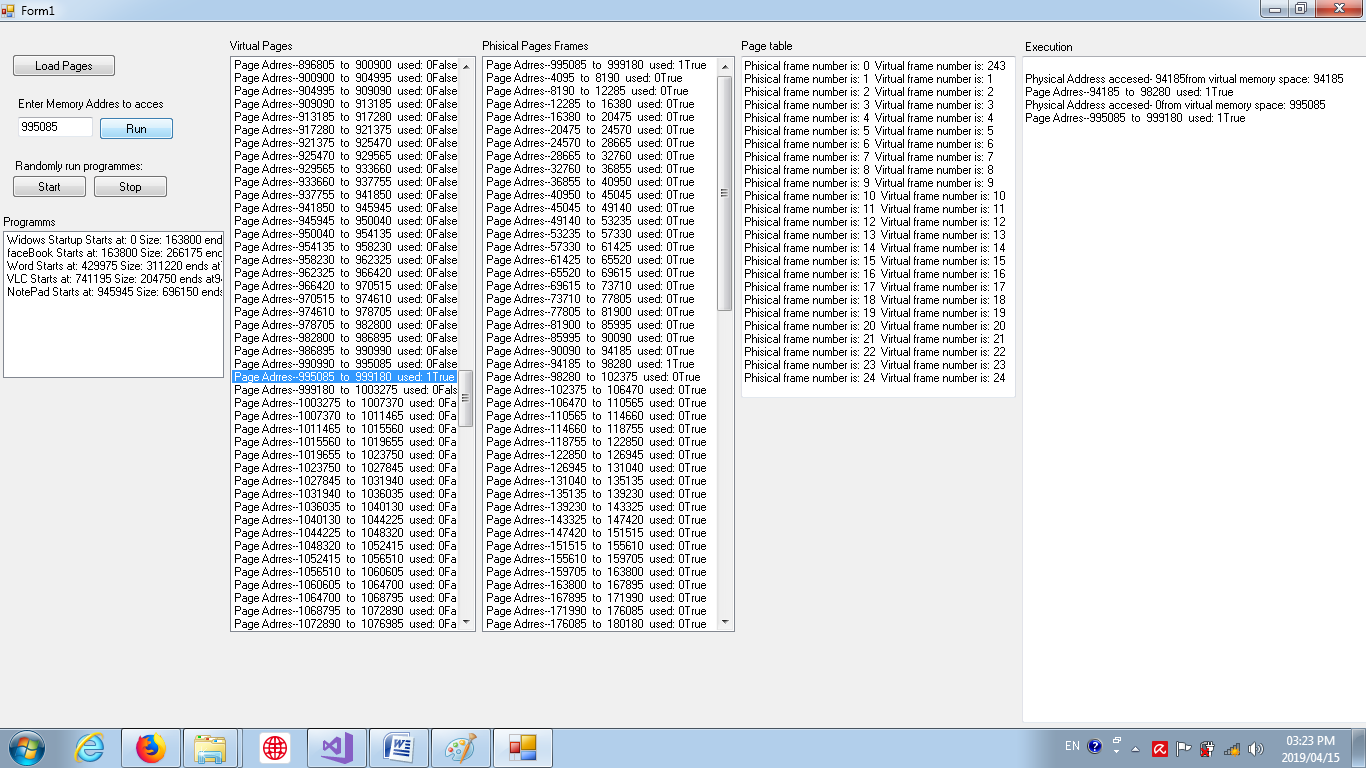
The user can now either enter a specific memory address he wants to access or select one from the list boxes. The program wil first check to se if the page is already in main memory by checking the present bit. If so the mapping is done and process is executed and its used attribute of the page is increased by one. If the present bid indicates that the page is not in physical memory the swop method wil initiate. The swop method iterate through the pages in the physical memory looking for pages with the lowest used attribute. The page with the lowest used attribute is then replaced by the page that is requested. Clicking on the start button selects random memory addresses and runs them by a timer as wil happens when clicking on the run button and clicking stop wil stop the process so that the execution can be observed.

The following screen shots displays the working of the simulation program with brief description of what happened.

Image 3:

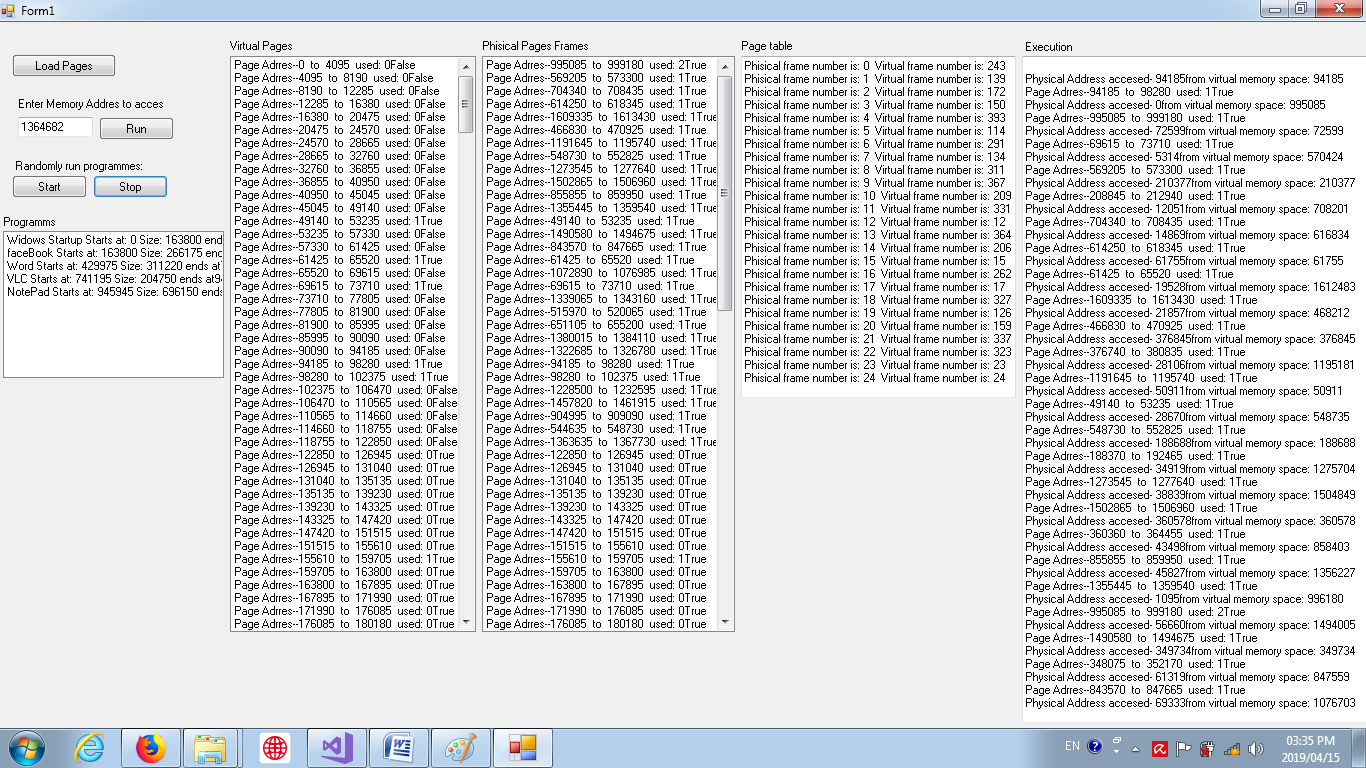
The selected items in the Virtual pages list boxes’ memory address is placed in textbox and ran. The execution textbox displays what memory is accessed from virtual and physical memory. As you can see the page frames dit not change as the memory adres was already present from the initial load of the form.

### Image 4:



As you can see i have scrolled down in the virtual pages list box to select a address that is not contained in the physical memory list box. The execution text box shows which memory is accessed in both physical and virtual spaces. In the virtual pages list box one can see that the used variable has increased and the present variable changed to 1 of true in the case of the simulation. One can also observe the new page loaded into the first page frame in the physical list box.

### Image 5:



In this image i have let the program run choosing random memories to access for 24 times. As you can see the page frames list box hase 29 pages whose used variable is not 0 yet the program has executed 30 time. If you observe the first item in the list box you wil see that page hase been used twice, hence 30 time of execution. This fully confirm that the program executes correctly and the virtual memory simulation is successful.

# Global or Local page Allocation

In local page allocations the algorithm wil use the used variable and to se witch page frame among pages related to the same process or program is leased used. It wil try to allocate a page frame to a page that is related to the process or program of the page the page frame currently possesses. Global allocation only revers to the used variable an relocates the page frame that contains the page least frequently used. Global allocation works beter when the working set size varies allot over time.

# Bibliography

Bos, Herbert, and Andrew S Tananbaum. *Modern Operating Systems.* London: Pearson Education Limited, 2015.