EOPSY Lab #4

Baran Korkmaz 302809

Introduction

The aim of this exercise is to simulate and understand how Operating Systems deal with page faults and to observe the execution of the page replacement by using a memory management simulator.

Theoretical Information

Physical Memory refers to the actual RAM of the system, which usually takes the form of cards attached onto the motherboard. Also called primary memory, it is the only storage type directly accessible to the CPU and holds the instructions of programs to execute.

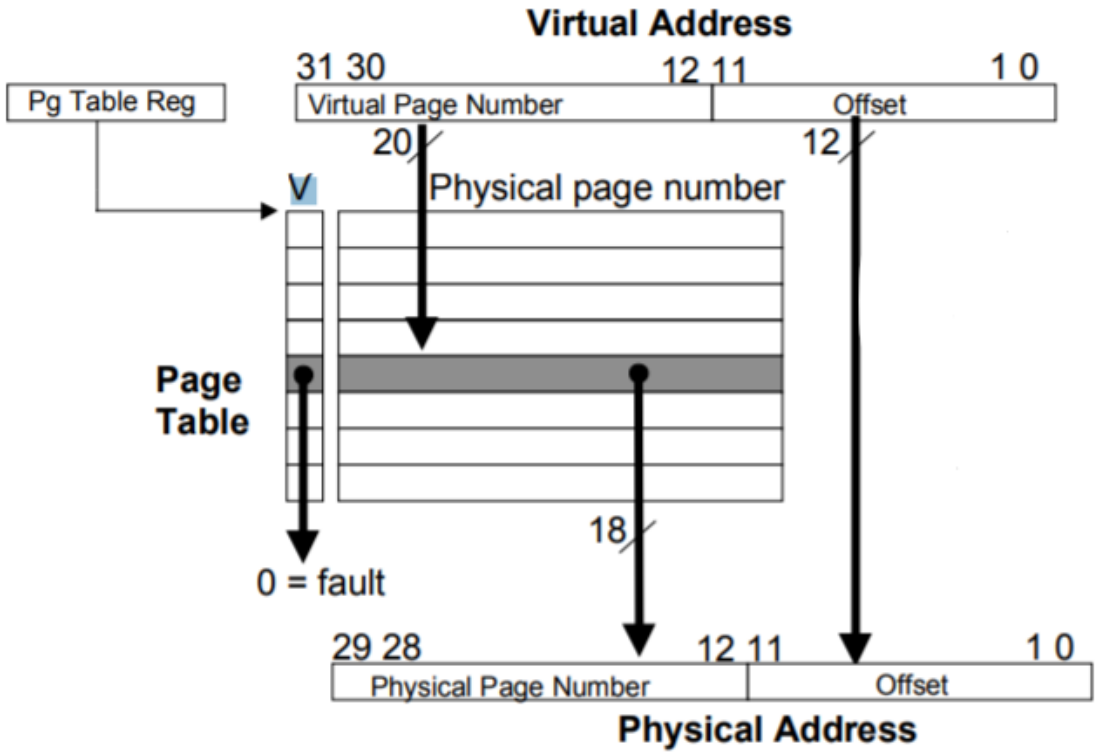
Virtual Memory is the process of mapping a logical address space numbered from 0 to the physical address space of the computer so that the RAM serves as a cache for the program's memory stored on the disk. It maps memory addresses used by a program, called **virtual addresses**, into **physical addresses** in computer memory. Virtual addresses, also called logical addresses, are program generated. In a system without virtual memory, a physical address and a logical address are the same; the virtual address is placed on the address bus directly because it refers to a physical address in the main memory. On systems with virtual memory, there is a mapping between virtual addresses and real addresses.

Virtual memory solves several critical problems:

- Simplifying RAM memory management by separating the address of memory from its physical location
- Providing protection for users by giving them their own address spaces
- Simplifying compilation and usage of libraries by allowing all programs to begin addressing memory from zero

The idea behind virtual memory is that physical memory is divided into **fixed-size pages**. Loadable modules are also divided into several **page frames**. Page frames are always the same size as the pages in memory.

Pages frames are loaded into memory only when they are needed. Adjacent page frames are not necessarily loaded into adjacent pages in memory. At a point in time during the execution of a program, only a small fraction of the total pages of a process may be loaded.



In **Demand paging**, only a set of pages of a process is loaded into the memory. This is done so that we can have more processes in the memory at the same time.

When a page that is residing in virtual memory is requested by a process for its execution, the Operating System needs to decide which page will be replaced by this requested page. This process is known as page replacement and is a vital component in virtual memory management.

A **page fault** is an interruption that occurs when a software program attempts to access a memory block not currently stored in the system's RAM. This exception tells the operating system to find the block in virtual memory so, it can be sent from a device's storage to RAM. Once all the data has been placed into physical memory, the program resumes normal operation. The Page fault process occurs in the background, and thus the user is unaware of it.

When a page fault occurs, the Operating System will have to replace one of the existing pages with the newly needed page. There are different **page replacing algorithms** exist with all aim to reduce the number of page faults.

Page Replacement Algorithms

Algorithms used to page replacement by the operating system to deal with page faults.

First in First Out Algorithm

In this algorithm, a queue of all the pages that are in the memory is currently maintained. The oldest page in memory is at the front of the queue and the most recent page is at the back of the queue.

Whenever a page fault occurs, the operating system looks at the front of the queue to get the page to be replaced by the newly requested page. It also adds this newly requested page at the end and removes the oldest page from the front of the queue.

Least Recently Used Algorithm

This algorithm keeps track of usage of pages over a period. This algorithm works based on the principle of locality of a reference which states that a program tends to access the same set of memory locations repetitively over a brief period. So, pages that have been used heavily in the past are most likely to be used heavily in the future also.

When a page fault occurs, then the page that has not been used for the longest duration of time is replaced by the newly requested page.

Last In First Out Algorithm

In this algorithm, the newest page is replaced by the requested page. Usually, this is done through a stack, where we maintain a stack of pages currently in the memory with the newest page being at the top.
When a page fault occurs, the page at the top of the stack is replaced.

Random Page Replacement

This method is choosing any random page in the memory to be replaced by the requested page.

Optimal Page Replacement

In this algorithm, the pages are replaced with the ones that will not be used for the longest duration of time in the future. In simple terms, the pages that will be referred to farthest in the future are replaced in this algorithm.

Procedure

Setup

First, we needed to map the first 8 virtual pages to any 8 physical pages. To achieve this, memory.conf file updated as:

```
memset 0 1 0 0 0 0
memset 1 3 0 0 0 0
memset 2 2 0 0 0 0
memset 3 4 0 0 0 0
memset 4 6 0 0 0 0
memset 5 0 0 0 0 0
memset 6 7 0 0 0 0
memset 7 5 0 0 0 0

enable_logging true

log_file tracefile

pagesize 128

addressradix 10

numpages 64
```

We can also observe that page size assigned as 128 Bytes and address radix as 10 so it will behave like decimal numbers.

Second, we needed to configure the simulator for reading once from 64 pages. We achieved that by editing the command file as:

```
READ    1
READ    129
READ    257
.
.
READ    7681
READ    7809
READ    7937
```

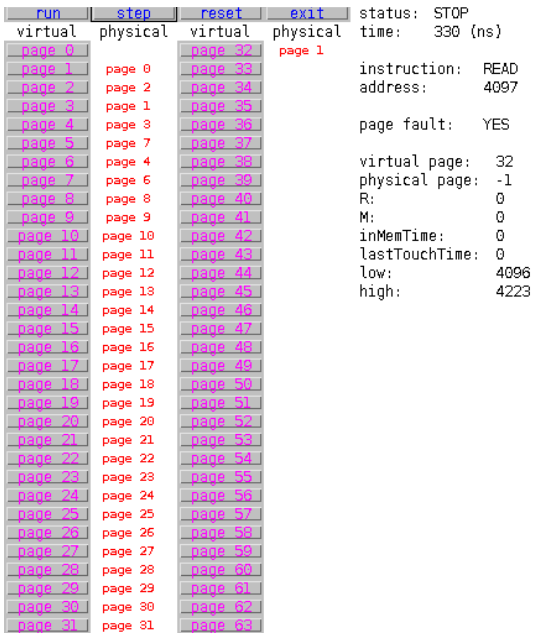
There are 64 read statements (one for each page) and numbers after each indicate the page address. We start at 1 and since every page has a 128-byte size, we shift 128 bytes to get to the next page.

Simulation

The initial state of the simulator is shown below. It is expected that the first 8 pages of virtual memory are mapped to 8 pages as configured before.

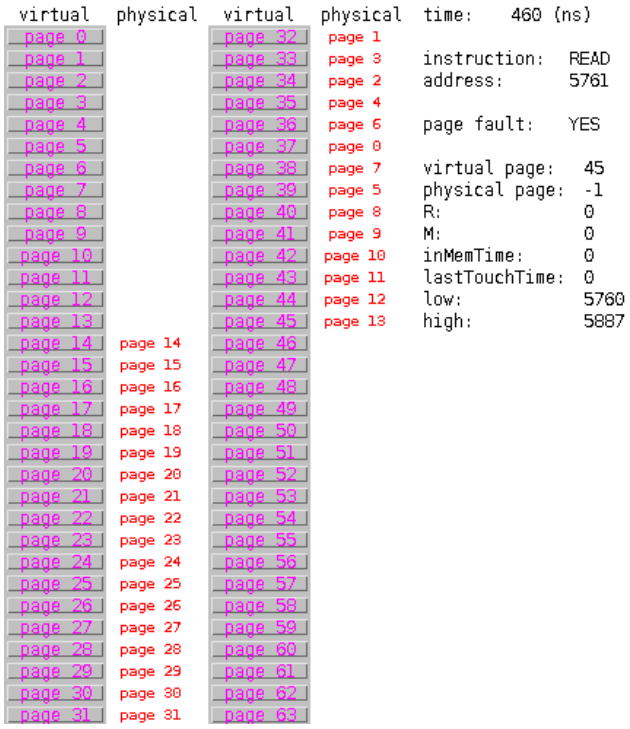
After stepping through one by one in the first 31 pages, there was no page fault observed. This is because the program has not tried to access a memory block not stored in the physical memory.

We can see from the below figure at the 33rd virtual page we have a flag for page fault.

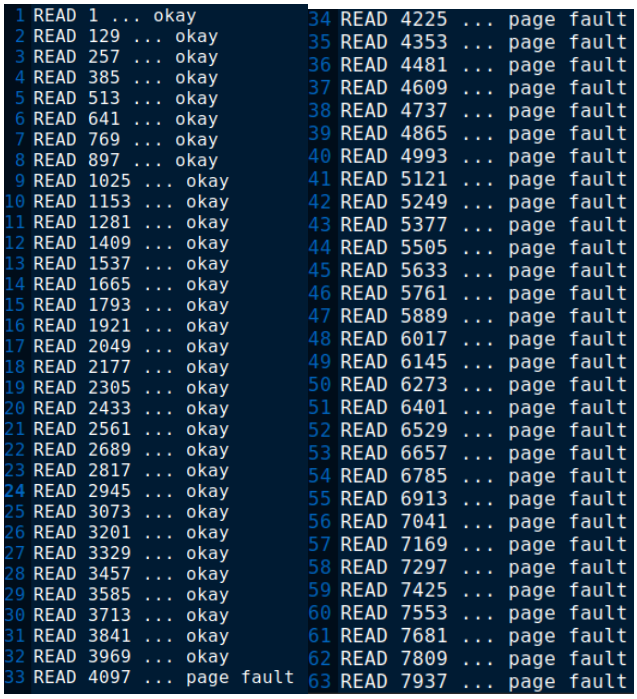


At this point, the simulator replaced the first page in the queue to newly requested page. This gives us a clue that this simulator might be using First In First Out algorithm.

Once we continue with the simulation, it is clear that, this simulator indeed uses the First In First Out algorithm since it continues to replace it from the first page from the queue.



The same outcome can be observed from the trace file:



Conclusions

- The First In First Out algorithm was used in this simulation. It is a simple algorithm that is not optimal since it does not make a difference between pages frequently used or not used at all like other algorithms such as Least Recently Used or Optimal Page Replacement.
- This simulator creates an equal number of physical and virtual memory pages and since we configured it to have 64 total pages, it created 32 virtual and 32 physical memory pages.
- Since it creates 32 virtual memory pages, but we needed to read from 64, page faults started occurring on the 33rd page. The simulator then used the replacement algorithm to find a block in virtual memory and map it to physical memory.