Operating Systems
Homework 2 - Spring 2023

Barış Ayyıldız

Data Types

Class PageTableEntry

- This class represents an entry in the page table.
- It has private member variables: dirty, present, timestamp, and pageFrameNumber.
- It has public member functions to access and modify these variables: getDirty(), getPresent(), getPageFrameNumber(), getTimestamp(), setDirty(), setPresent(), setPageFrameNumber(), setTimestamp().
- The class also has constructors (PageTableEntry() and PageTableEntry(int pageFrameNumber)) to initialize the member variables.

Class PageTable

- This class represents the page table itself.
- It has private member variables: size (size of the page table) and table (an array of PageTableEntry objects).
- It has public member functions to get and set page table entries: getPageTableEntry(int index), setPageTableEntry(int index, PageTableEntry pte).
- The class also has constructors (PageTable() and PageTable(int size)) to initialize the member variables.

Class MemoryManagement

- This class handles memory management operations.
- It has private member variables: pageTable (an instance of PageTable), virtualPageCount (number of virtual pages), pageFrameCount (number of page frames), physicalMemory (an array to represent physical memory), interval (an interval value), filename (a string representing the filename), replacementAlgorithm (an instance of the PageReplacement enum), THRESHOLD (a threshold value), and PAGE_SIZE (size of a page).
- It has public member functions to perform memory management operations: showPageTable(), forceDiskWrite(), get(int virtualPage), set(int virtualPage, int num).
- The class also has constructors (MemoryManagement() and MemoryManagement(int virtualPageCount, int pageFrameCount, std::string filename, PageReplacement replacementAlgorithm, int pageSize, int interval)) and a destructor (~MemoryManagement()) to initialize and clean up the member variables.

Enum PageReplacement

- An enum (enumeration) is a user-defined data type that consists of a set of named values.
- In the code, there is an enum PageReplacement defined.

- It defines named values for page replacement algorithms: LRU, WSClock, and SC.
- Enums provide a way to represent a set of options or choices as symbolic names, improving code readability and maintainability.

Custom Variables

- MAX_LONG: A constant representing the maximum value of a long data type.
- Number_Of_Reads, Number_Of_Writes, etc.: Variables to track the number of specific operations performed.
- pageTable: An instance of the PageTable class used in the MemoryManagement class.
- replacementAlgorithm: An instance of the PageReplacement enum representing the selected page replacement algorithm.
- virtualPageCount, pageFrameCount, physicalMemory: Variables representing counts and sizes related to memory management.
- interval, filename, THRESHOLD, PAGE_SIZE: Variables representing various parameters and settings for memory management operations.

Code Explanation

Functions

MemoryManagement

There are two memory management constructors, it takes some parameters and sets its variables with them, such as virtual page count, page framecount, filename, replacement algorithm and so on.

In the constructor, it creates a page table with the virtual page count and it creates virtual and physical memories inside the generateMemory function.

generateMemory

This function writes some random integers between 0 to 100 to the specified file name. This works like virtual memory. And also fills the physical memory with those random numbers until it gets filled.

get

This function tries to get the integer at the specified position in the virtual memory. First it calculates which virtual page entry that integer locates and then it finds the offset. If the virtual page presents in the physical memory it simply returns the integer using the offset.

But if it doesn't present in the physical memory a page miss occurs, it increments number of page misses and number of page replacements. It first checks which page replacement algorithm is used in the MemoryManagement class. If it’s LRU it finds the least recently used present page and puts the contents of the virtual page that is desired by the user in the main memory where the old page currently points. Then it sets the present bit of that old page as 0. But before it checks if the dirty bit is 1. Which means that the least recently used bit is changed before, so it writes the contents of the physical memory back to the virtual memory for that LRU page.

If the page replacement algorithm is specified by the user is WSClock it searches all the pages in the page table. If the difference between the current time and the time that page is last used is greater than a threshold value it executes the page replacement with that page.

If the page replacement algorithm is second change, it finds the oldest page in the page table. If that page entry’s dirty bit is 1, it writes its content in the physical memory to virtual memory and continues to find the second oldest page. If a selected page’s dirty bit is 0 it makes the page replacement.

set

Set function is very similar to the get function. It executes page replacement algorithms if it’s necessary. Only thing that is different is that it doesn’t return anything; it just sets the specified memory location.

mapMemoryToFile

This functions writes the contents of the memory to the file for a virtual page and a page frame

mapFileToMemory

The opposite of mapMemoryToFile, it writes the contents of the file to the memory

linarSearch

It does linear search between start and end with the target value target.

binarySearch

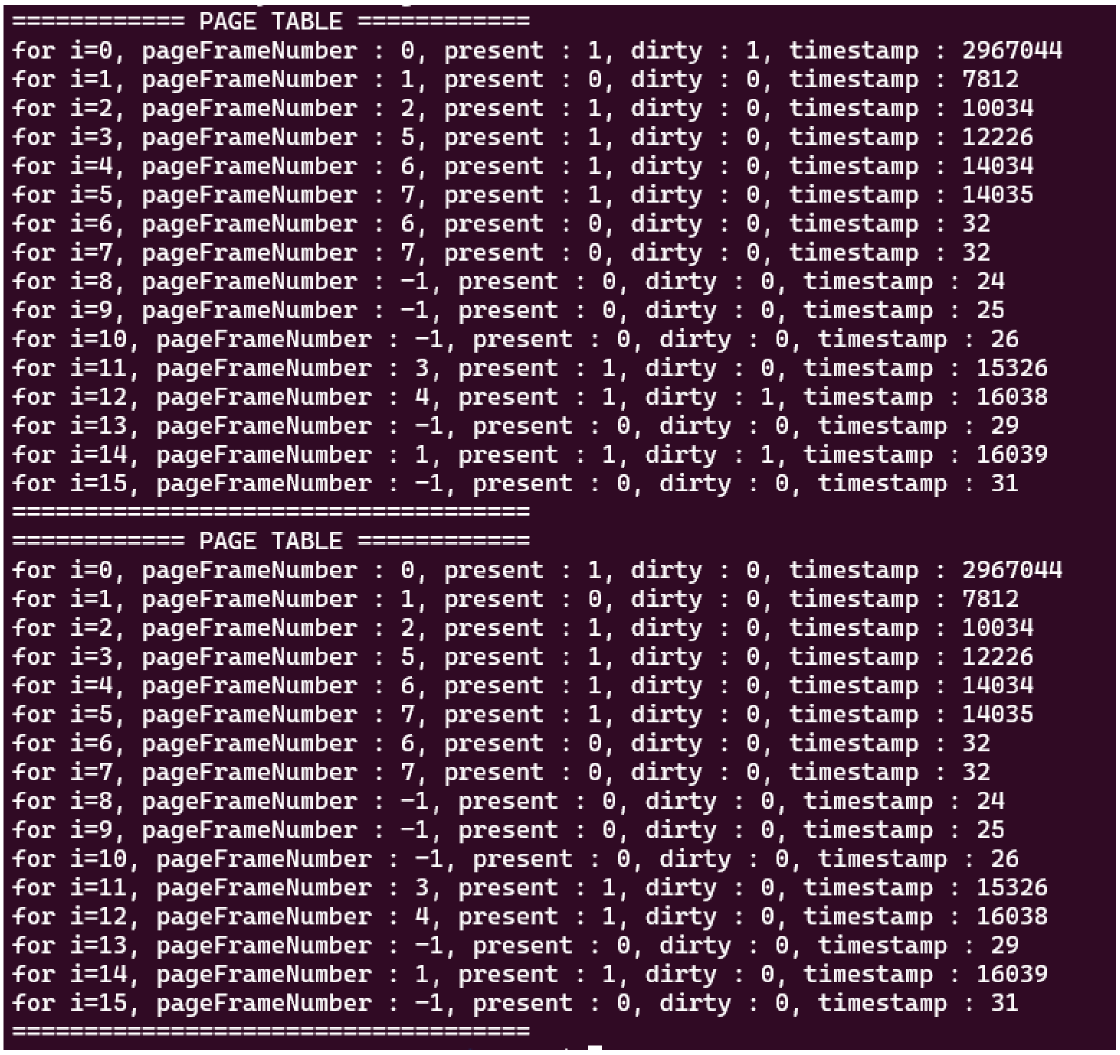
It does linear search between start and end with the target value target. In order to do that it first sorts the array.

matrixMultiplication

It takes startOne, xOne, yOne, startTwo, xTwo, yTwo and save integers as parameters. startOne is the starting point of the first matrix. xOne and yOne is the shape of the first matrix and the other parameters are for the second matrix.

part1.cpp

In this part, we create a memory management object with the virtual page count 16 and with 8 page frame count and the page size is 4kb. It makes 2 matrix multiplication, 1 array summation and calls the search algorithms. In the end it prints the page table.

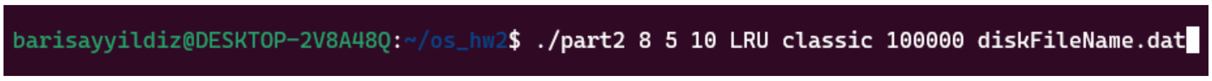


```
============ PAGE TABLE ============
for i=0, pageFrameNumber : 0, present : 1, dirty : 1, timestamp : 2967044
for i=1, pageFrameNumber : 1, present : 0, dirty : 0, timestamp : 7812
for i=2, pageFrameNumber : 2, present : 1, dirty : 0, timestamp : 10034
for i=3, pageFrameNumber : 5, present : 1, dirty : 0, timestamp : 12226
for i=4, pageFrameNumber : 6, present : 1, dirty : 0, timestamp : 14034
for i=5, pageFrameNumber : 7, present : 1, dirty : 0, timestamp : 14035
for i=6, pageFrameNumber : 6, present : 0, dirty : 0, timestamp : 32
for i=7, pageFrameNumber : 7, present : 0, dirty : 0, timestamp : 32
for i=8, pageFrameNumber : -1, present : 0, dirty : 0, timestamp : 24
for i=9, pageFrameNumber : -1, present : 0, dirty : 0, timestamp : 25
for i=10, pageFrameNumber : -1, present : 0, dirty : 0, timestamp : 26
for i=11, pageFrameNumber : 3, present : 1, dirty : 0, timestamp : 15326
for i=12, pageFrameNumber : 4, present : 1, dirty : 1, timestamp : 16038
for i=13, pageFrameNumber : -1, present : 0, dirty : 0, timestamp : 29
for i=14, pageFrameNumber : 1, present : 1, dirty : 1, timestamp : 16039
for i=15, pageFrameNumber : -1, present : 0, dirty : 0, timestamp : 31
====================================
============ PAGE TABLE ============
for i=0, pageFrameNumber : 0, present : 1, dirty : 0, timestamp : 2967044
for i=1, pageFrameNumber : 1, present : 0, dirty : 0, timestamp : 7812
for i=2, pageFrameNumber : 2, present : 1, dirty : 0, timestamp : 10034
for i=3, pageFrameNumber : 5, present : 1, dirty : 0, timestamp : 12226
for i=4, pageFrameNumber : 6, present : 1, dirty : 0, timestamp : 14034
for i=5, pageFrameNumber : 7, present : 1, dirty : 0, timestamp : 14035
for i=6, pageFrameNumber : 6, present : 0, dirty : 0, timestamp : 32
for i=7, pageFrameNumber : 7, present : 0, dirty : 0, timestamp : 32
for i=8, pageFrameNumber : -1, present : 0, dirty : 0, timestamp : 24
for i=9, pageFrameNumber : -1, present : 0, dirty : 0, timestamp : 25
for i=10, pageFrameNumber : -1, present : 0, dirty : 0, timestamp : 26
for i=11, pageFrameNumber : 3, present : 1, dirty : 0, timestamp : 15326
for i=12, pageFrameNumber : 4, present : 1, dirty : 0, timestamp : 16038
for i=13, pageFrameNumber : -1, present : 0, dirty : 0, timestamp : 29
for i=14, pageFrameNumber : 1, present : 1, dirty : 0, timestamp : 16039
for i=15, pageFrameNumber : -1, present : 0, dirty : 0, timestamp : 31
====================================
```

First it printed the page table then it forced the memory management to write all the dirty pages back to the virtual memory. That’s why when it printed the page table for the second time we had no dirty bits.

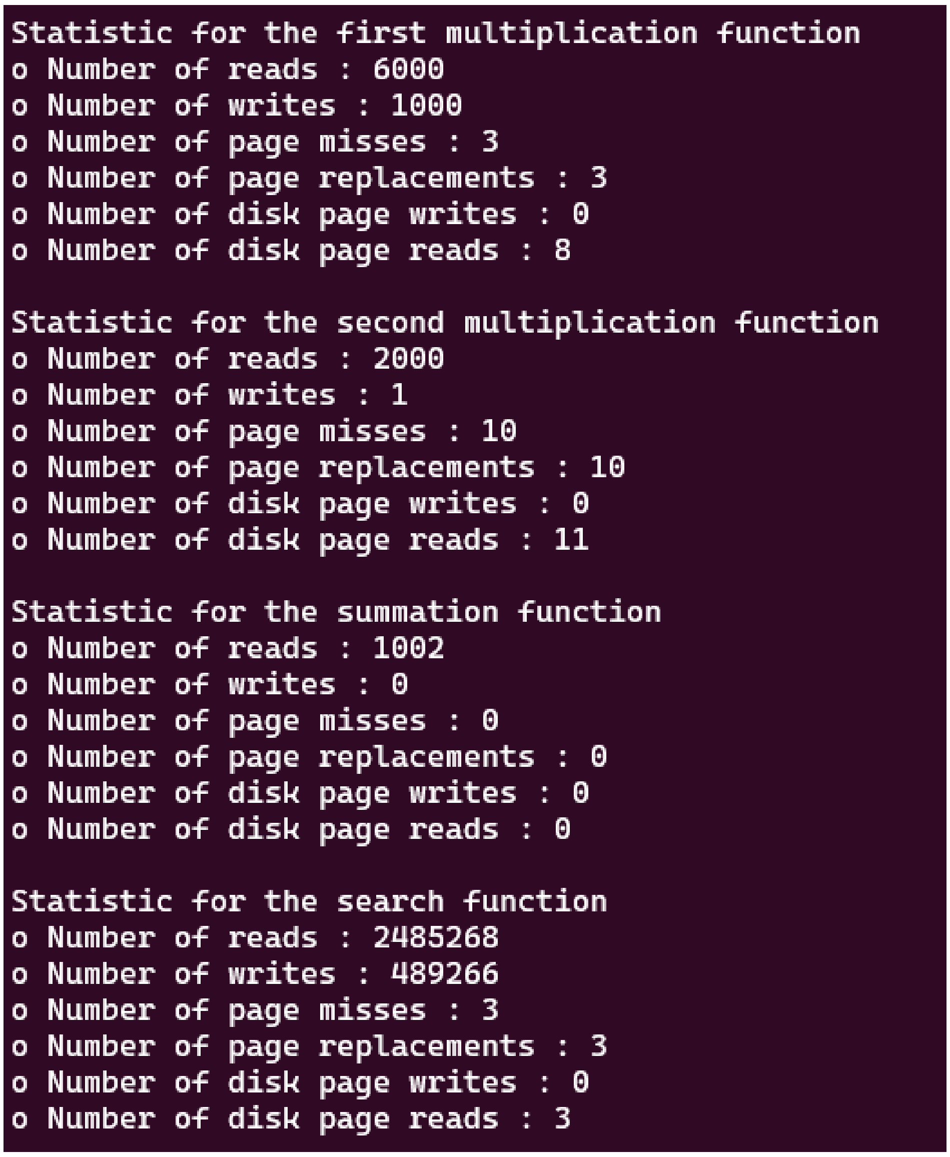
part2.cpp

In this part, we call the matrix multiplication function for different parts of the memory and print the number of reads, number of writes, number of page misses, number of page replacements, number of disk page writes and number of disk page reads. In the homework we were told to use a frame size of 4096 integers, 32 physical frames, 1024 virtual frames. But creation of that type of memory takes 5-10 minutes each time. So in this part of the code I reduced those numbers.



```
barisayyildiz@DESKTOP-2V8A48Q:~/os_hw2$ ./part2 8 5 10 LRU classic 100000 diskFileName.dat
```

I ran part2 like this. It creates a frame size with 2^8 integers, 2^5 physical frames, 2^10 virtual frames, uses the classic version of page table, prints the page table in 100000 memory accesses and uses diskFileName.dat as the virtual memory.

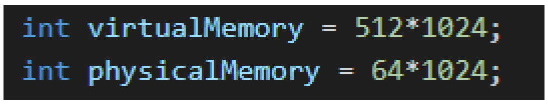


```
Statistic for the first multiplication function
o Number of reads : 6000
o Number of writes : 1000
o Number of page misses : 3
o Number of page replacements : 3
o Number of disk page writes : 0
o Number of disk page reads : 8

Statistic for the second multiplication function
o Number of reads : 2000
o Number of writes : 1
o Number of page misses : 10
o Number of page replacements : 10
o Number of disk page writes : 0
o Number of disk page reads : 11

Statistic for the summation function
o Number of reads : 1002
o Number of writes : 0
o Number of page misses : 0
o Number of page replacements : 0
o Number of disk page writes : 0
o Number of disk page reads : 0

Statistic for the search function
o Number of reads : 2485268
o Number of writes : 489266
o Number of page misses : 3
o Number of page replacements : 3
o Number of disk page writes : 0
o Number of disk page reads : 3
```

Here are the results

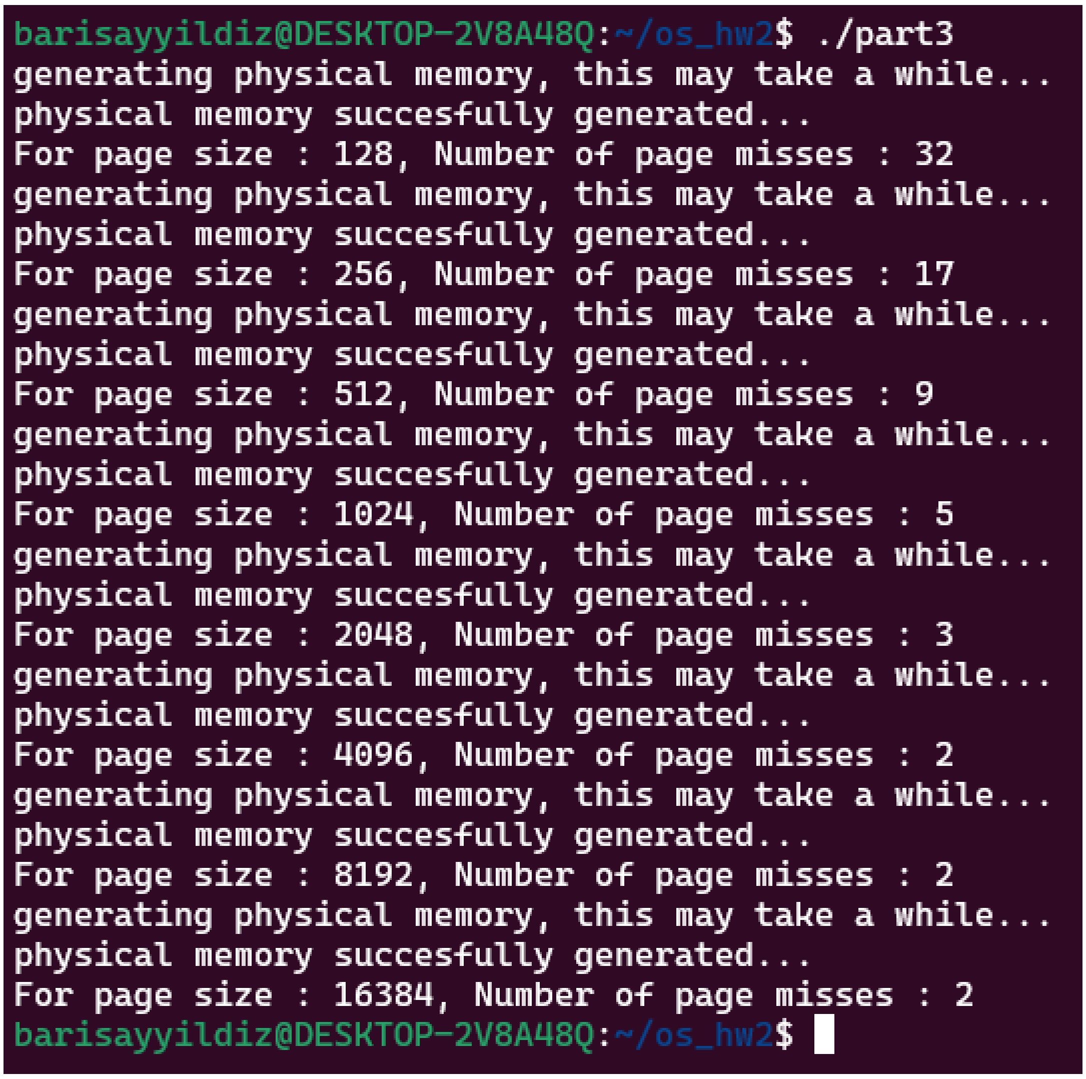
part3.cpp

It creates a memory management unit with the virtual memory 512kb, physical memory with 64kb, which means we can store 128kb of integers in the virtual and 16kb of integers in the physical memory.

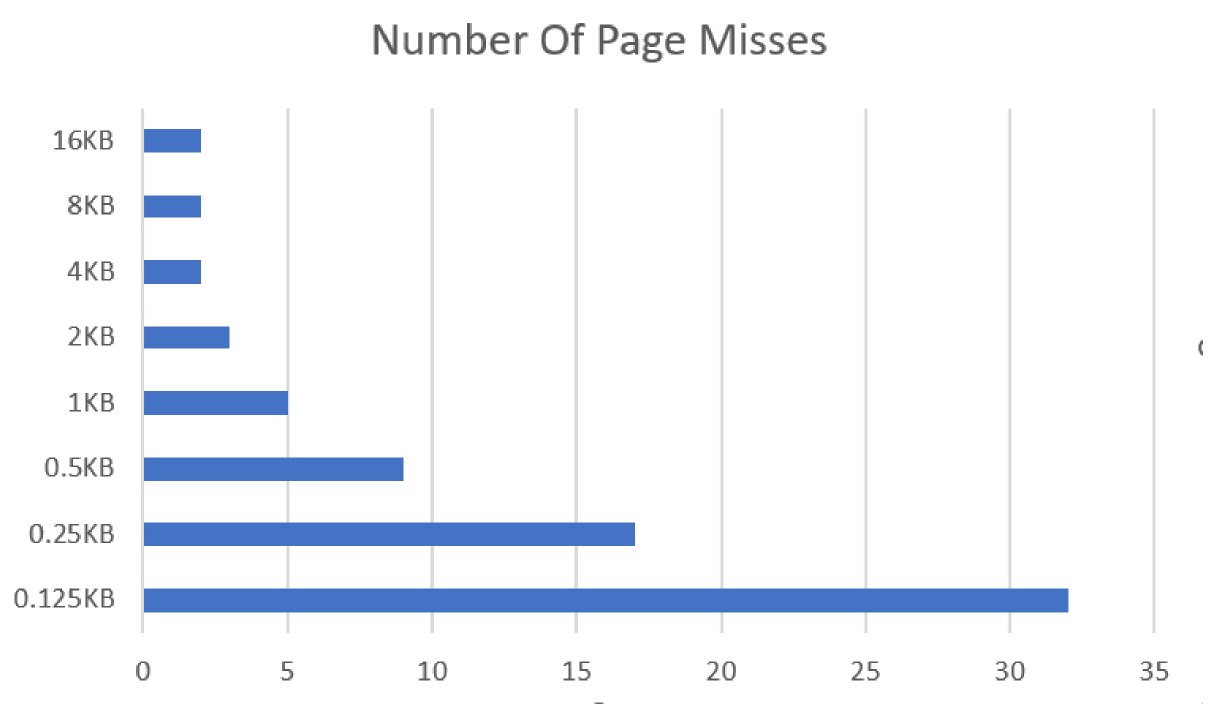


```
int virtualMemory = 512*1024;
int physicalMemory = 64*1024;
```

Then it creates different memory management units with different pages sizes. And prints how many page misses occured in each of the time.



```
barisayyildiz@DESKTOP-2V8A48Q:~/os_hw2$ ./part3
generating physical memory, this may take a while...
physical memory succesfully generated...
For page size : 128, Number of page misses : 32
generating physical memory, this may take a while...
physical memory succesfully generated...
For page size : 256, Number of page misses : 17
generating physical memory, this may take a while...
physical memory succesfully generated...
For page size : 512, Number of page misses : 9
generating physical memory, this may take a while...
physical memory succesfully generated...
For page size : 1024, Number of page misses : 5
generating physical memory, this may take a while...
physical memory succesfully generated...
For page size : 2048, Number of page misses : 3
generating physical memory, this may take a while...
physical memory succesfully generated...
For page size : 4096, Number of page misses : 2
generating physical memory, this may take a while...
physical memory succesfully generated...
For page size : 8192, Number of page misses : 2
generating physical memory, this may take a while...
physical memory succesfully generated...
For page size : 16384, Number of page misses : 2
barisayyildiz@DESKTOP-2V8A48Q:~/os_hw2$
```



How to run

make part1
make part2
make part3

    To compile part1.cpp, part2.cpp and part3.cpp

./part1
./part2 <frame size> <physical frames> <virtual frames> <page replacement algorithm> classic <interval> <file name>
./part3

    To run executable files

Make clean

    To remove the object and executable files