EOPSY TASK 4

MEMORY MANAGEMENT

LAKSMAN SIVARAM SENTHILKUMAR
317289

CONTENTS

1 INTRODUCTION

The objective of this task is to get familiarised with Memory Management Schemes, Paging, Mapping, Page Replacement Algorithms, etc.

2 MEMORY MANAGEMENT

The subdividing the memory among different processes is called memory management. Memory management is a method in the operating system to manage operations between main memory and disk during process execution. The main aim of memory management is to achieve efficient utilisation of memory.

2.1 PAGING

It is a memory management scheme that eliminates the need for contiguous allocation of physical memory. This scheme permits the physical address space of a process to be non-contiguous.

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard drive that's set up to emulate the computer's RAM. Paging technique plays an important role in implementing virtual memory.

A page is a fixed-length contiguous block of virtual memory, whereas a frame is a fixed-length block of physical memory.

2.2 MAPPING

Memory mapping is the translation between the logical(virtual) address space and the physical memory. The objectives of memory mapping are to translate from logical to physical address, to aid in memory protection, and to enable better management of memory resources.

Mapping is important to computer performance, both locally (how long it takes to execute an instruction) and globally (how long it takes to run a set of programs). In effect, each time a program presents a logical memory address and requests that the corresponding memory word be accessed, the mapping mechanism must translate that address

into an appropriate physical memory location. The simpler this translation, the lower the implementation cost and the higher the performance of the individual memory reference.

2.3 PHYSICAL AND VIRTUAL MEMORY

Physical memory refers to the RAM or the primary memory in the computer. Physical memory is a volatile memory. Therefore, it requires a continuous flow of power to retain data. However, power failures and interruptions can erase the data in the physical memory. Also, this memory is linearly addressable. In other words, the memory addresses increase in a linear manner.

Virtual memory is a logical memory. In other words, it is a memory management technique performed by the operating system. Virtual memory allows the programmer to use more memory for the programs than the available physical memory. Physical memory, which is the actual RAM, is a form of computer data storage that stores the currently executing programs. In contrast, virtual memory is a memory management technique that creates an illusion to users of larger physical memory. Thus, this is the main difference between physical and virtual memory.

2.4 PAGE FAULT

A page fault (sometimes called PF or hard fault) is an exception that the memory management unit (MMU) raises when a process accesses a memory page without proper preparations. Accessing the page requires a mapping to be added to the process's virtual address space. Besides, the actual page contents may need to be loaded from a backing store, such as a disk. The MMU detects the page fault, but the operating system's kernel handles the exception by making the required page accessible in the physical memory or denying an illegal memory access.

Valid page faults are common and necessary to increase the amount of memory available to programs in any operating system that utilises virtual memory. Page faults degrade system performance and can cause thrashing.

2.5 PAGE REPLACEMENT ALGORITHMS

2.5.1 First In First Out (FIFO)

This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced, the page in the front of the queue is selected for removal.

2.5.2 Optimal Page Replacement

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

2.5.3 Least Recently Used

In this algorithm, pages will be replaced with the one which is least recently used.

3 TASK

Create a command file that maps any 8 pages of physical memory to the first 8 pages of virtual memory, and then reads from one virtual memory address on each of the 64 virtual pages. Step through the simulator one operation at a time and see if you can predict which virtual memory addresses cause page faults. What page replacement algorithm is being used?

Locate in the sources and describe to the instructor the page replacement algorithm.

4 SIMULATIONS

4.1 MEMORY CONFIG FILE

Below you can see the contents of the memory.conf file.

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

enable_logging true
log_file tracefile
pagesize 16384
numpages 16
```

As we were supposed to map 8 physical pages of memory to 8 virtual pages of memory. I've mapped the first 8 pages of physical memory to the first 8 pages of virtual memory. Page size has been left intact. Address radix has been changed into decimal so that we can see the addresses easily without having to do the conversion from other radixes. Since we map 8 physical pages to 8 virtual pages, I've changed numpages to be 16 so that we are not using any other pages in the beginning.

4.2 COMMANDS FILE

Below you can see the commands that the simulation will execute.

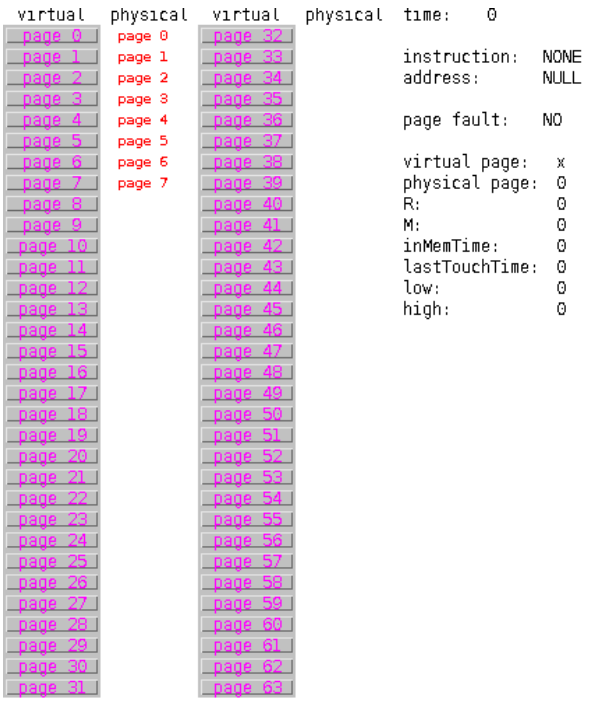
```
READ 114688
READ 131072
READ 245760
READ 49151
READ 1542
READ 16384
READ bin 100
READ 20
READ 19
WRITE 147456
```

WRITE 148456
WRITE hex CC32
READ bin 100000000000000
READ bin 100000000000000
WRITE bin 110000000000001
WRITE random
READ 131072
READ 131073
READ 200000

4.3 RUNNING THE SIMULATOR

PAGE SIZE = 3fff(hex), 16834(dec)
Since our memory blocks are of size 16384, when calculating which address maps to which page, we need to think in multiples of 16384. For example the first page will have addresses between 0-16383 and the second page will have the addresses between 16384-32767. You can also verify this by clicking on virtual pages on the simulation. In Figure 1 you can actually verify that the addresses are between 0 and 16383 by looking at the low and high values on the right side of the pages.

Below you can see the outputs of the simulator



virtual	physical	virtual	physical	time: 0
page 0	page 0	page 32		
page 1	page 1	page 33		instruction: NONE
page 2	page 2	page 34		address: NULL
page 3	page 3	page 35		
page 4	page 4	page 36		page fault: NO
page 5	page 5	page 37		
page 6	page 6	page 38		virtual page: 0
page 7	page 7	page 39		physical page: 0
page 8		page 40		R: 0
page 9		page 41		M: 0
page 10		page 42		inMemTime: 0
page 11		page 43		lastTouchTime: 0
page 12		page 44		low: 0
page 13		page 45		high: 16383
page 14		page 46		
page 15		page 47		
page 16		page 48		
page 17		page 49		
page 18		page 50		
page 19		page 51		
page 20		page 52		
page 21		page 53		
page 22		page 54		
page 23		page 55		
page 24		page 56		
page 25		page 57		
page 26		page 58		
page 27		page 59		
page 28		page 60		
page 29		page 61		
page 30		page 62		
page 31		page 63		

virtual	physical	virtual	physical	time: 0
page 0	page 0	page 32		
page 1	page 1	page 33		instruction: NONE
page 2	page 2	page 34		address: NULL
page 3	page 3	page 35		
page 4	page 4	page 36		page fault: NO
page 5	page 5	page 37		
page 6	page 6	page 38		virtual page: 8
page 7	page 7	page 39		physical page: -1
page 8		page 40		R: 0
page 9		page 41		M: 0
page 10		page 42		inMemTime: 0
page 11		page 43		lastTouchTime: 0
page 12		page 44		low: 131072
page 13		page 45		high: 147455
page 14		page 46		
page 15		page 47		
page 16		page 48		
page 17		page 49		
page 18		page 50		
page 19		page 51		
page 20		page 52		
page 21		page 53		
page 22		page 54		
page 23		page 55		
page 24		page 56		
page 25		page 57		
page 26		page 58		
page 27		page 59		
page 28		page 60		
page 29		page 61		
page 30		page 62		
page 31		page 63		

4.4 TRACEFILE

Below you can see the contents of the trace file after we've run through all the steps in our commands file

READ 114688 ... okay
READ 131072 ... page fault
READ 245760 ... page fault
READ 49151 ... okay
READ 1542 ... page fault
READ 16384 ... page fault
READ 4 ... okay
READ 20 ... okay
READ 19 ... okay
WRITE 147456 ... page fault
WRITE 148456 ... okay
WRITE 52274 ... page fault
READ 16384 ... okay
READ 16384 ... okay
WRITE 24577 ... okay
WRITE 5845 ... okay
READ 131072 ... okay
READ 131073 ... okay
READ 200000 ... page fault

4.5 OBSERVATION

Below you can find the explanation of all the commands in the commands file and why there was a page fault if there was any while executing a command.

READ 114688 ... okay
Since the address is in the virtual page 7 (114688/16384 = 7) and since the virtual page 7 is mapped we only get an okay result.

READ 131072 ... page fault
The address 131072 is in the virtual page 8 and we do not have that mapped to any physical page, hence, we get a page fault and the simulation maps the virtual page 8 to physical page 0.

READ 245760 ... page fault
The address 245760 is in the virtual page 15 and we do not have that mapped to any physical page, hence, we get a page fault and the simulation maps the virtual page 15 to physical page 1 which was the first in the queue. This also gives us a hint about which algorithm the simulation uses. It is FIFO (First In First Out).

READ 49151 ... okay
The address 49151 is in the virtual page 2 and that page is mapped to physical page 2 so we have no problem reading from that address.

READ 1542 ... page fault
Since the address 1542 is in virtual page 0 and we've actually re-mapped our physical page 0 from virtual page 0 to virtual page 8 above. We get a page fault trying to read from virtual page 0. Hence the algorithm again maps virtual page 0 to the next physical page in the queue which is physical page 2.

READ 16384 ... page fault
The address 16384 is actually in virtual page 1 but since the virtual page 1 is now not mapped to any physical page, we again get a page fault. After that the algorithm maps physical page 3 to virtual page 1.

READ 4 ... okay - READ 20 ... okay - READ 19 ... okay
Since the three commands above all read from the virtual page 0 and that page is actually mapped to physical page 2 we can easily read it without having any page fault. And basically the rest of the commands follow the same algorithm and that concludes the conclusions about this task.

From the file "PageDefault.java", we are able to analyse the code of method replacePage. It states that the "oldest" physical page that is mapped to a virtual page is then removed and assigned to the virtual page that hasn't yet been mapped. This algorithm is called First-In, First-Out, FIFO for short. It works like a queue – the first thing that comes towards the FIFO, will be the first one to leave.

replacePage from PageDefault.java

```
/**
   * The page replacement algorithm for the memory management sumulator.
   * This method gets called whenever a page needs to be replaced.
   * <p>
   * The page replacement algorithm included with the simulator is
   * FIFO (first-in first-out).  A while or for loop should be used
   * to search through the current memory contents for a canidate
   * replacement page.  In the case of FIFO the while loop is used
   * to find the proper page while making sure that virtPageNum is
   * not exceeded.
   * <pre>
   *   Page page = ( Page ) mem.elementAt( oldestPage )
   * </pre>
   * This line brings the contents of the Page at oldestPage (a
   * specified integer) from the mem vector into the page object.
   * Next recall the contents of the target page, replacePageNum.
   * Set the physical memory address of the page to be added equal
   * to the page to be removed.
   * <pre>
   *   controlPanel.removePhysicalPage( oldestPage )
   * </pre>
   * Once a page is removed from memory it must also be reflected
   * graphically.  This line does so by removing the physical page
   * at the oldestPage value.  The page which will be added into
   * memory must also be displayed through the addPhysicalPage
   * function call.  One must also remember to reset the values of
   * the page which has just been removed from memory.
   *
   * @param mem is the vector which contains the contents of the pages
   *   in memory being simulated.  mem should be searched to find the
   *   proper page to remove, and modified to reflect any changes.
   * @param virtPageNum is the number of virtual pages in the
   *   simulator (set in Kernel.java).
   * @param replacePageNum is the requested page which caused the
   *   page fault.
   * @param controlPanel represents the graphical element of the
   *   simulator, and allows one to modify the current display.
    */
  public static void replacePage ( Vector mem , int virtPageNum , int replacePageNum ,
ControlPanel controlPanel )
  {
    int count = 0;
    int oldestPage = -1;
    int oldestTime = 0;
    int firstPage = -1;
    int map_count = 0;
    boolean mapped = false;

    while ( ! (mapped) || count != virtPageNum ) {
      Page page = ( Page ) mem.elementAt( count );
      if ( page.physical != -1 ) {
        if (firstPage == -1) {
          firstPage = count;
        }
```

```
        if (page.inMemTime > oldestTime) {
          oldestTime = page.inMemTime;
          oldestPage = count;
          mapped = true;
        }
      }
      count++;
      if ( count == virtPageNum ) {
        mapped = true;
      }
    }
    if (oldestPage == -1) {
      oldestPage = firstPage;
    }
    Page page = ( Page ) mem.elementAt( oldestPage );
    Page nextpage = ( Page ) mem.elementAt( replacePageNum );
    controlPanel.removePhysicalPage( oldestPage );
    nextpage.physical = page.physical;
    controlPanel.addPhysicalPage( nextpage.physical , replacePageNum   );
    page.inMemTime = 0;
    page.lastTouchTime = 0;
    page.R = 0;
    page.M = 0;
    page.physical = -1;
  }
```

4.5 ADDITIONAL CASE

In this case we explore a bug/flaw in the simulator. When we try to map a virtual page to a physical page in a non-linear arrangement, the simulator maps incorrect physical page addresses to the virtual pages.

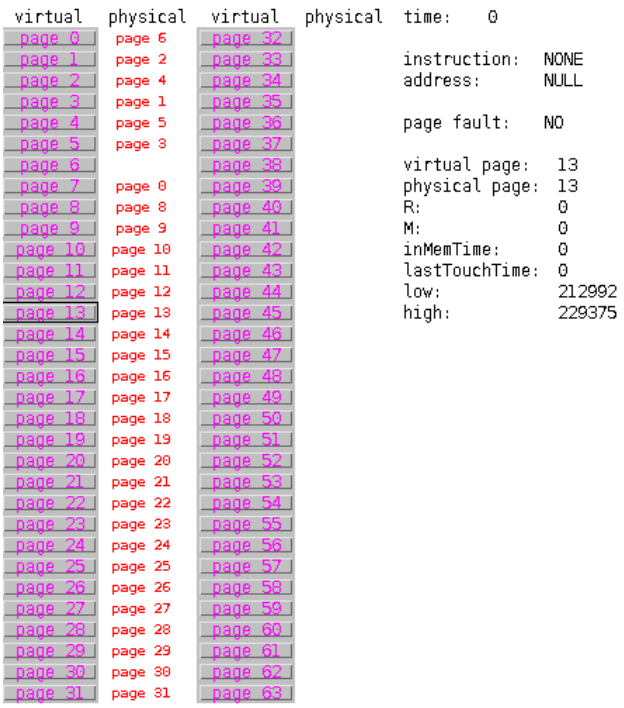
MEMORY CONFIGURATION FILE

```
memset 0 7 0 0 0 0
memset 1 3 0 0 0 0
memset 2 1 0 0 0 0
memset 3 5 0 0 0 0
memset 4 2 0 0 0 0
memset 5 4 0 0 0 0
memset 6 0 0 0 0 0
memset 7 13 0 0 0 0

enable_logging true
log_file tracefile
pagesize 16384
numpages 64
```

virtual	physical	virtual	physical	time: 0
page 0	page 6	page 32		
page 1	page 2	page 33		instruction: NONE
page 2	page 4	page 34		address: NULL
page 3	page 1	page 35		
page 4	page 5	page 36		page fault: NO
page 5	page 3	page 37		
page 6		page 38		virtual page: 6
page 7	page 0	page 39		physical page: 0
page 8	page 8	page 40		R: 0
page 9	page 9	page 41		M: 0
page 10	page 10	page 42		inMemTime: 0
page 11	page 11	page 43		lastTouchTime: 0
page 12	page 12	page 44		low: 98304
page 13	page 13	page 45		high: 114687
page 14	page 14	page 46		
page 15	page 15	page 47		
page 16	page 16	page 48		
page 17	page 17	page 49		
page 18	page 18	page 50		
page 19	page 19	page 51		
page 20	page 20	page 52		
page 21	page 21	page 53		
page 22	page 22	page 54		
page 23	page 23	page 55		
page 24	page 24	page 56		
page 25	page 25	page 57		
page 26	page 26	page 58		
page 27	page 27	page 59		
page 28	page 28	page 60		
page 29	page 29	page 61		
page 30	page 30	page 62		
page 31	page 31	page 63		

virtual	physical	virtual	physical	time: 0
page 0	page 6	page 32		
page 1	page 2	page 33		instruction: NONE
page 2	page 4	page 34		address: NULL
page 3	page 1	page 35		
page 4	page 5	page 36		page fault: NO
page 5	page 3	page 37		
page 6		page 38		virtual page: 7
page 7	page 0	page 39		physical page: 13
page 8	page 8	page 40		R: 0
page 9	page 9	page 41		M: 0
page 10	page 10	page 42		inMemTime: 0
page 11	page 11	page 43		lastTouchTime: 0
page 12	page 12	page 44		low: 114688
page 13	page 13	page 45		high: 131071
page 14	page 14	page 46		
page 15	page 15	page 47		
page 16	page 16	page 48		
page 17	page 17	page 49		
page 18	page 18	page 50		
page 19	page 19	page 51		
page 20	page 20	page 52		
page 21	page 21	page 53		
page 22	page 22	page 54		
page 23	page 23	page 55		
page 24	page 24	page 56		
page 25	page 25	page 57		
page 26	page 26	page 58		
page 27	page 27	page 59		
page 28	page 28	page 60		
page 29	page 29	page 61		
page 30	page 30	page 62		
page 31	page 31	page 63		



TRACEFILE

```
READ 114688 ... okay
READ 131072 ... okay
READ 245760 ... okay
READ 49151 ... okay
READ 1542 ... okay
READ 16384 ... okay
READ 4 ... okay
READ 20 ... okay
READ 19 ... okay
WRITE 147456 ... okay
WRITE 148456 ... okay
WRITE 52274 ... okay
READ 16384 ... okay
READ 16384 ... okay
WRITE 24577 ... okay
WRITE 659307 ... page fault
READ 131072 ... okay
READ 131073 ... okay
READ 200000 ... okay
```

OBSERVATION
Here, when we try to map virtual page 7 to physical address 13, the simulator maps virtual page 7 to physical page 13 and also maps virtual page 13 to physical page 13, which is incorrect behaviour of the simulator. Also, we can notice that the physical pages we specified in the memory configuration file for other virtual pages are decreased by 1. From the tracefile, we can see that we only get one page fault this time, because we enabled all 64 pages of memory.

5 CONCLUSION

As I can see in the replacePage method, there is a FIFO Page Replacement algorithm. This algorithm will be called by Memory Management when there is a page fault. The algorithm is implemented with a while loop to look through the current memory contents for a candidate replacement page, attending to First In First Out function. The algorithm is used to find the proper page making sure that virtual page number is not exceeded.

Advantages of FIFO Page Replacement

- It is simple and easy to understand & implement.
- It is efficiently used for small systems
- It does not cause more overheads

Disadvantages of FIFO Page Replacement

- The process effectiveness is low.
- When we increase the number of frames while using FIFO, we are giving more memory to processes. So, page faults should decrease, but here the page faults are increasing. This problem is called Belady's Anomaly.
- Every frame needs to be taken into account.
- It uses an additional data structure.

6 REFERENCES

[1]https://www.ia.pw.edu.pl/~tkruk/edu/eopsy/lab/task4.gtz

[2]https://tldp.org/LDP/tlk/mm/memory.html

[3]http://faculty.salina.k-state.edu/tim/ossg/Memory/virt_mem/page_replace.html

---