

EOPSY

Lab 4: Memory Management

Julia Czmur 300168

Introduction

Memory management is the functionality of the operating system which handles the processes allocation in memory.

The processes are moved back and forth between the parts of the memory. When it happens, the free memory space is divided further and further into little pieces. At some point these free pieces can be too small and remain not used at all – this problem is called **fragmentation**.

A solution to fragmentation can be **paging**, in which process address space is divided into equally-sized blocks called pages. The pages which belong to a specific process are loaded into available memory frames. To reiterate, frames are used to divide the physical memory, and pages are used to divide the processes' virtual address space.

Paging mechanism is important in implementing virtual memory and it is responsible for:

- mapping virtual addresses onto physical addresses (locating the referenced page and frame used by that page),
- sending pages from external memory to operating memory and sending back pages which are not required anymore.

The goal of this task was to configure the Memory Management simulator to map any 8 pages of physical memory to the first 8 pages of virtual memory, and then read from one virtual memory address on each of the 64 virtual pages.

memory.conf

```
// memset virt page # physical page # R (read from) M (modified) inMemTime (ns) lastTouchTime (ns)
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' or 'false'
// When true specify a log_file or leave blank for stdout
enable_logging true

// log_file <FILENAME>
// Where <FILENAME> is the name of the file you want output
// to be print to.
log_file tracefile

// page size, defaults to 2^14 and cannot be greater than 2^26
// pagesize <single page size (base 10)> or <'power' num (base 2)>
pagesize 16384

// addressradix sets the radix in which numerical values are displayed
// 2 is the default value
// addressradix <radix>
addressradix 10

// numpages sets the number of pages (physical and virtual)
// 64 is the default value
// numpages must be at least 2 and no more than 64
// numpages <num>
numpages 64
```

I edited the `memory.conf` file so that only 8 pages are mapped accordingly to our task's specifications (`memset` commands at the top which map respective pages between physical and virtual memory).

I also changed the address radix to 10, so that the numerical values are displayed in decimal.

The rest was left unchanged.

Then, in the `commands` file, I specified that the simulator should execute the `READ` command 64 times to addresses being the multiples of 16384, in order to read from one virtual memory address on each of the 64 virtual pages.

Part of the **commands** file:

```
READ 0
READ 16384          // 64 addresses being the multiples of 16384
READ 32768
READ 49152
READ 65536
READ 81920
...
READ 999424
READ 1015808
READ 1032192
```

Comments about the simulation results

After stepping through the simulator one operation at a time, it became clear that the first 8 pages were mapped correctly as was specified in the `memory.conf` file. Moreover, the pages 8-31 were also mapped correctly (probably by default by the simulator).

A **page fault** occurs when a virtual page which has not been mapped to a physical page yet is being referenced.

It turned out that the address 524288 caused the first page fault. It occurred at the 32nd virtual page and the simulator showed it was mapped to physical page -1 (so it was not mapped to any physical page). From that moment, each further address caused page faults up until 64th page, so the last one.

When a page fault happens, this page is being mapped to the first physical page in queue (depending on which page replacement algorithm is used).

Part of `tracefile` file:

```
READ 0 ... okay
READ 16384 ... okay
READ 32768 ... okay
READ 49152 ... okay
...
READ 507904 ... okay
READ 524288 ... page fault
READ 540672 ... page fault
...
READ 1032192 ... page fault
```

Information about the page replacement algorithm could be found in `PageFault.java` file (it's highlighted on the screenshot below).

```

/* It is in this file, specifically the replacePage function that will
   be called by MemoryManagement when there is a page fault. The
   users of this program should rewrite PageFault to implement the
   page replacement algorithm.
*/

// This PageFault file is an example of the FIFO Page Replacement
// Algorithm as described in the Memory Management section.

import java.util.*;

public class PageFault {

    /**
     * The page replacement algorithm for the memory management simulator.
     * 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 candidate
     * 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>
    */

```

The page replacement algorithm used here is **FIFO (First-In First-Out)**. So, the first page mapped (first in) is used as replacement in the first place (first out).

run	step	reset	exit	status: STOP
virtual	physical	virtual	physical	time: 320 (ns)
page 0	page 0	page 32		
page 1	page 1	page 33		
page 2	page 2	page 34		instruction: READ
page 3	page 3	page 35		address: 507904
page 4	page 4	page 36		
page 5	page 5	page 37		page fault: NO
page 6	page 6	page 38		
page 7	page 7	page 39		virtual page: 31
page 8	page 8	page 40		physical page: 31
page 9	page 9	page 41		R: 0
page 10	page 10	page 42		M: 0
page 11	page 11	page 43		inMemTime: 310
page 12	page 12	page 44		lastTouchTime: 310
page 13	page 13	page 45		low: 507904
page 14	page 14	page 46		high: 524287
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		

Last mapped page (31st)

run	step	reset	exit	status: STOP
virtual	physical	virtual	physical	time: 330 (ns)
page 0		page 32	page 0	
page 1	page 1	page 33		
page 2	page 2	page 34		instruction: READ
page 3	page 3	page 35		address: 524288
page 4	page 4	page 36		
page 5	page 5	page 37		page fault: YES
page 6	page 6	page 38		
page 7	page 7	page 39		virtual page: 32
page 8	page 8	page 40		physical page: -1
page 9	page 9	page 41		R: 0
page 10	page 10	page 42		M: 0
page 11	page 11	page 43		inMemTime: 0
page 12	page 12	page 44		lastTouchTime: 0
page 13	page 13	page 45		low: 524288
page 14	page 14	page 46		high: 540671
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		

First page fault

run	step	reset	exit	status: STOP	
virtual	physical	virtual	physical	time: 340 (ns)	
page 0		page 32	page 0		
page 1		page 33	page 1	instruction: READ	
page 2	page 2	page 34		address: 540672	
page 3	page 3	page 35			
page 4	page 4	page 36		page fault: YES	
page 5	page 5	page 37			
page 6	page 6	page 38		virtual page: 33	
page 7	page 7	page 39		physical page: -1	
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: 540672	
page 13	page 13	page 45		high: 557055	
page 14	page 14	page 46			

run	step	reset	exit	status: STOP	
virtual	physical	virtual	physical	time: 370 (ns)	
page 0		page 32	page 0		
page 1		page 33	page 1	instruction: READ	
page 2		page 34	page 2	address: 589824	
page 3		page 35	page 3		
page 4		page 36	page 4	page fault: YES	
page 5	page 5	page 37			
page 6	page 6	page 38		virtual page: 36	
page 7	page 7	page 39		physical page: -1	
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: 589824	
page 13	page 13	page 45		high: 606207	
page 14	page 14	page 46			

The screenshots above show the simulator step by step around the important point of the first page fault. It is visible that when the first page fault occurs at the 32nd page, page number 0 (which was first in) is first taken as replacement. After that, consecutive pages are taken by the replacement algorithm – at 33rd page, page number 1 is taken (which is the next “first in” after page 0) and so on.