

EOPSY LABORATORY REPORT TASK 4

PAGE REPLACEMENT EXERCISE

Aayush gupta

309601

Contents

1. INTRODUCTION	3
A. Memory Management	3
A.1 Partitioing in Memory Management	3
A.2 Paging in Memory Management	3
A.3 Segmentation in Memory Management	3
B. Different Ways of Memory Management	3
B.1. Fixed Partitioning	3
B.2 Dynamic Partitioning	4
B.3 Simple Paging	4
B.4 Simple Segmentation	4
B.5 Virtual-Memory Paging	4
B.6 Virtual-Memory Segmentation	4
C. Page Replacement Algorithms	5
2. CONTENT OF EXPERIMENT	5

A.	Task Description	5
B.	Environment and Required Files	5
B.1	MOSS (Memory Management Simulator)	5
B.2	Configuration File	6
B.3	Commands File	7
B.4	Output File (tracefile)	7
3.	RUNNING THE SIMULATOR 1	8
A.	Configuration and Commands Files	8
A.1	Configuration File	8
A.2	Commands File	9
B.	Simulation	10
C.	Discussion	13
D.	Cocnlusion	14
E.	Output File (tracefile)	14
4.	RUNNING THE SIMULATION 2	15
A.	Configuration and Commands Files	15
A.1	Configuration File	15
A.2	Commands File	15
B.	Simulation	16
C.	Discussion	25
D.	Cocnlusion	25
E.	Output File (tracefile)	25

1. INTRODUCTION

A. Memory Management

Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution. Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

A.1 Partitioing in Memory Management

Memory partitioning is the system by which the memory of a computer system is divided into sections for use by the resident programs. These memory divisions are known as partitions. There are different ways in which memory can be partitioned: fixed, variable, and dynamic partitioning.

A.2 Paging in Memory Management

Paging 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.3 Segmentation in Memory Management

Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program.

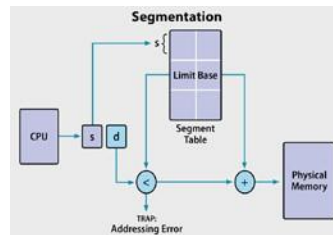


Figure 1: Segmentation Visualization

B. Different Ways of Memory Management¹

B.1. Fixed Partitioning

This partitioning approach divided into a fixed number of partitions just one process can be loaded into one partition at the same time. Strengths of this approach: easy to implement it and slandered method as a partitioning solution. Weaknesses of this approach insufficient use because of the internal fragmentation, must know the maximum number of active processes can run is fixed size of the task is limited to largest partition size, degree of multiprogramming limited by the number of partitions, memory is wasted in the partition, must translate relative address to physical address.

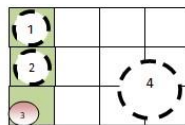


Figure 2: Fixed Partitioning Visualization

B.2 Dynamic Partitioning

Partitions are created dynamically, each process loaded into a partition is exactly have same size as the process. Strengths of this approach are, ensure more efficient use of the main memory and no internal fragmentation. Weaknesses of this approach are inefficient use of processor because of the need for compaction and external fragmentation.

Figure 3: Memory Space

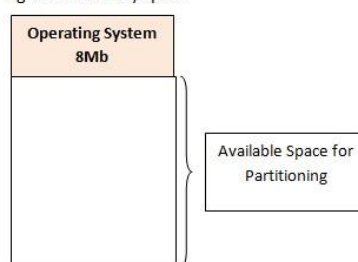


Figure 3: Dynamic Partitioning Visualization

B.3 Simple Paging

Strengths of this approach are, no need for external fragmentation, transfers between disks can be at the granularity of individual pages. Weaknesses of this approach are: maybe there is no

¹ International Journal of Scientific & Engineering Research, Volume 7, Issue 4, April-2016

correspondence between page protections. Settings and application data structures, requiring per process page tables, usually operating system need more storage for its internal data structures.

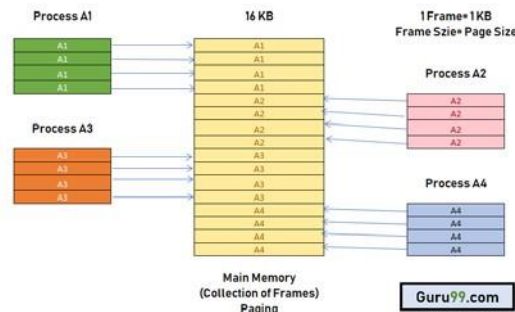


Figure 4: Paging Visualization

B.4 Simple Segmentation

Strengths of this approach is no internal fragmentation. Weaknesses of this approach are reduce the overhead compared to dynamic partitioning approach and improved the memory utilization.

B.5 Virtual-Memory Paging

Strengths of this approach are, having large virtual address space, no external fragmentation and higher degree of multi programming. Weaknesses of this approach is overhead coming from the complex of memory management.

B.6 Virtual-Memory Segmentation

Strengths of this approach are, it supports a high level of multiprogramming especially the enormous virtual address space and no internal fragmentation. Weaknesses of this approach is the overhead of complex memory management.²

C. Page Replacement Algorithms

The page replacement algorithm decides which memory page is to be replaced. The process of replacement is sometimes called swap out or write to disk. Page replacement is done when the requested page is not found in the main memory (page fault).

There are two main aspects of virtual memory, Frame allocation and Page Replacement. It is very important to have the optimal frame allocation and page replacement algorithm. Frame allocation is all about how many frames are to be allocated to the process while the page replacement is all about determining the page number which needs to be replaced in order to make space for the requested page.³

FIFO Algorithm: In this algorithm, a queue is maintained. The page which is assigned the frame first will be replaced first. In other words, the page which resides at the rare end of the queue will be replaced on the every page fault.

² International Journal of Scientific & Engineering Research, Volume 7, Issue 4, April-2016

³ <https://www.javatpoint.com/os-page-replacement-algorithms>

Optimal Page Replacement Algorithm: In this algorithm, pages are replaced which would not be used for the longest duration of time in the future. Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

Least Recently Used: In this algorithm, page will be replaced which is least recently used.⁴

2. CONTENT OF EXPERIMENT

A. Task Description

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.

B. Environment and Required Files⁵

B.1 MOSS (Memory Management Simulator)

The memory management simulator illustrates page fault behavior in a paged virtual memory system. The program reads the initial state of the page table and a sequence of virtual memory instructions and writes a trace log indicating the effect of each instruction. It includes a graphical user interface so that students can observe page replacement algorithms at work. Students may be asked to implement a particular page replacement algorithm which the instructor can test by comparing the output from the student's algorithm to that produced by a working implementation.

The program reads a command file, optionally reads a configuration file, displays a GUI window which allows you to execute the command file, and optionally writes a trace file.

The buttons:

Button	Description
run	runs the simulation to completion. Note that the simulation pauses and updates the screen between each step.
step	runs a single setup of the simulation and updates the display.
reset	initializes the simulator and starts from the beginning of the command file.
exit	exits the simulation.
page <i>n</i>	display information about this virtual page in the display area at the right.

The informational display:

⁴ <https://www.geeksforgeeks.org/page-replacement-algorithms-in-operating-systems/>

⁵ 2001, Prentice-Hall, Inc.

Field	Description
status:	RUN, STEP, or STOP. This indicates whether the current run or step is completed.
time:	number of "ns" since the start of the simulation.
instruction:	READ or WRITE. The operation last performed.
address:	the virtual memory address of the operation last performed.
page fault:	whether the last operation caused a page fault to occur.
virtual page:	the number of the virtual page being displayed in the fields below. This is the last virtual page accessed by the simulator, or the last page n button pressed.
physical page:	the physical page for this virtual page, if any. -1 indicates that no physical page is associated with this virtual page.
R:	whether this page has been read. (1=yes, 0=no)
M:	whether this page has been modified. (1=yes, 0=no)
inMemTime:	number of ns ago the physical page was allocated to this virtual page.
lastTouchTime:	number of ns ago the physical page was last modified.
low:	low virtual memory address of the virtual page.
high:	high virtual memory address of the virtual page.

B.2 Configuration File

The configuration file `memory.conf` is used to specify the the initial content of the virtual memory map (which pages of virtual memory are mapped to which pages in physical memory) and provide other configuration information, such as whether operation should be logged to a file.

The `memset` command is used to initialize each entry in the virtual page map. `memset` is followed by six integer values:

1. The virtual page # to initialize
2. The physical page # associated with this virtual page (-1 if no page assigned)
3. If the page has been read from (R) (0=no, 1=yes)
4. If the page has been modified (M) (0=no, 1=yes)
5. The amount of time the page has been in memory (in ns)
6. The last time the page has been modified (in ns)

The first two parameters define the mapping between the virtual page and a physical page, if any.

The last four parameters are values that might be used by a page replacement algorithm. For example, `memset 34 23 0 0 0 0`

specifies that virtual page 34 maps to physical page 23, and that the page has not been read or modified.

Note:

- Each physical page should be mapped to exactly one virtual page.
- The number of virtual pages is fixed at 64 (0..63).
- The number of physical pages cannot exceed 64 (0..63).
- If a virtual page is not specified by any `memset` command, it is assumed that the page is not mapped.

Keyword	Values	Description
enable_logging	true false	Whether logging of the operations should be enabled. If logging is enabled, then the program writes a one-line message for each READ or WRITE operation. By default, no logging is enabled. See also the <code>log_file</code> option.
log_file	<i>trace-file-name</i>	The name of the file to which log messages should be written. If no filename is given, then log messages are written to stdout. This option has no effect if <code>enable_logging</code> is false or not specified.
pagesize	<i>n</i> power <i>p</i>	The size of the page in bytes as a power of two. This can be given as a decimal number which is a power of two (1, 2, 4, 8, etc.) or as a power of two using the <code>power</code> keyword. The maximum page size is 67108864 or <code>power 26</code> . The default page size is <code>power 26</code> .
addressradix	<i>n</i>	The radix in which numerical values are displayed. The default radix is 2 (binary). You may prefer radix 8 (octal), 10 (decimal), or 16 (hexadecimal).

B.3 Commands File

The command file for the simulator specifies a sequence of memory instructions to be performed. Each instruction is either a memory READ or WRITE operation, and includes a virtual memory address to be read or written. Depending on whether the virtual page for the address is present in physical memory, the operation will succeed, or, if not, a page fault will occur.

There are two operations one can carry out on pages in memory: READ and WRITE. The format for each command is *operation address* or *operation random*, where *operation* is READ or WRITE, and *address* is the numeric virtual memory address, optionally preceded by one of the radix keywords bin, oct, or hex. If no radix is supplied, the number is assumed to be decimal. The keyword random will generate a random virtual memory address (for those who want to experiment quickly) rather than having to type an address.

For example, the sequence

READ bin 01010101

WRITE bin 10101010

READ random WRITE

random

causes the virtual memory manager to:

1. read from virtual memory address 85
2. write to virtual memory address 170
3. read from some random virtual memory address
4. write to some random virtual memory address

B.4 Output File (tracefile)

The output file contains a log of the operations since the simulation started (or since the last reset). It lists the command that was attempted and what happened as a result. You can review this file after executing the simulation.

The output file contains one line per operation executed. The format of each line is:

command address ... status where:

- *command* is READ or WRITE,
- *address* is a number corresponding to a virtual memory address, and
- *status* is okay or page fault.

3. RUNNING THE SIMULATOR 1

A. Configuration and Commands Files

A.1 Configuration File

As It was described in the Task Description part, we created the memory.conf file with 8 pages. We will observe the pages from 0 to 7 in both virtual and physical pages. Rest of the settings remain same.

```
1 // memset virt page # physical page # R (read from) M (modified) inMemTime (ns) lastTouchTime (ns)
2 memset 0 7 0 0 0 0
3 memset 1 6 0 0 0 0
4 memset 2 5 0 0 0 0
5 memset 3 4 0 0 0 0
6 memset 4 3 0 0 0 0
7 memset 5 2 0 0 0 0
8 memset 6 1 0 0 0 0
9 memset 7 0 0 0 0 0
10
11 // enable_logging 'true' or 'false'
12 // When true specify a log_file or leave blank for stdout
13 enable_logging true
14
15 // log_file <FILENAME>
16 // Where <FILENAME> is the name of the file you want output
17 // to be print to.
18 log_file tracefile
19
20 // page size, defaults to 2^14 and cannot be greater than 2^26
21 // pagesize <single page size (base 10)> or <'power' num (base 2)>
22 pagesize 16384
23
24 // addressradix sets the radix in which numerical values are displayed
25 // 2 is the default value
26 // addressradix <radix>
27 addressradix 16
28
29 // numpages sets the number of pages (physical and virtual)
30 // 64 is the default value
31 // numpages must be at least 2 and no more than 64
32 // numpages <num>
33 numpages 64
```

Figure 5: Configuration File

We did not map for all pages but we made pagesize as 64, it means that for the pages till 31, it will automatically assign physical pages.

A.2 Commands File

In hexadecimal form, for all 64 pages.


```

1 // Enter READ/WRITE commands into this file
2 // READ <OPTIONAL number type: bin/hex/oct> <virtual memory address or random>
3 // WRITE <OPTIONAL number type: bin/hex/oct> <virtual memory address or random>
4 READ hex 0000
5 READ hex 4000
6 READ hex 8000
7 READ hex c000
8 READ hex 10000
9 READ hex 14000
10 READ hex 18000
11 READ hex 1c000
12 READ hex 20000
13 READ hex 24000
14 READ hex 28000
15 READ hex 2c000
16 READ hex 30000
17 READ hex 34000
18 READ hex 38000
19 READ hex 3c000
20 READ hex 40000
21 READ hex 44000
22 READ hex 48000
23 READ hex 4c000
24 READ hex 50000
25 READ hex 54000
26 READ hex 58000
27 READ hex 5c000
28 READ hex 60000
29 READ hex 64000
30 READ hex 68000
31 READ hex 6c000
32 READ hex 70000
33 READ hex 74000
34 READ hex 78000
35 READ hex 7c000
36 READ hex 80000
37 READ hex 84000
38 READ hex 88000
39 READ hex 8c000
40 READ hex 90000
41 READ hex 94000
42 READ hex 98000
43 READ hex 9c000
44 READ hex a0000
45 READ hex a4000
46 READ hex a8000
47 READ hex ac000
48 READ hex b0000
49 READ hex b4000
50 READ hex b8000
51 READ hex bc000
52 READ hex c0000
53 READ hex c4000
54 READ hex c8000
55 READ hex cc000
56 READ hex d0000
57 READ hex d4000
58 READ hex d8000
59 READ hex dc000
60 READ hex e0000
61 READ hex e4000
62 READ hex e8000
63 READ hex ec000
64 READ hex f0000
65 READ hex f4000
66 READ hex f8000
67 READ hex fc000

```

Figure 6: Commands File

B. Simulation

Memory Management					
run	step	reset	exit	status:	STOP
virtual	physical	virtual	physical	time:	0
page 0	page 7	page 32		instruction:	NONE
page 1	page 6	page 33		address:	NULL
page 2	page 5	page 34		page fault:	NO
page 3	page 4	page 35		virtual page:	0
page 4	page 3	page 36		physical page:	7
page 5	page 2	page 37		R:	0
page 6	page 1	page 38		M:	0
page 7	page 0	page 39		inMemTime:	0
page 8	page 8	page 40		lastTouchTime:	0
page 9	page 9	page 41		low:	0
page 10	page 10	page 42		high:	3fff
page 11	page 11	page 43			
page 12	page 12	page 44			
page 13	page 13	page 45			
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	page 31		

Simulator Screen 1

At beginning everything is as we set. We can see that virtual page 0 is assigned to physical page 7 as we write in the configuration file.

Memory Management					
run	step	reset	exit	status:	STOP
virtual	physical	virtual	physical	time:	130 (ns)
page 0	page 7	page 32		instruction:	READ
page 1	page 6	page 33		address:	30000
page 2	page 5	page 34		page fault:	NO
page 3	page 4	page 35		virtual page:	12
page 4	page 3	page 36		physical page:	12
page 5	page 2	page 37		R:	0
page 6	page 1	page 38		M:	0
page 7	page 0	page 39		inMemTime:	120
page 8	page 8	page 40		lastTouchTime:	120
page 9	page 9	page 41		low:	30000
page 10	page 10	page 42		high:	33fff
page 11	page 11	page 43			
page 12	page 12	page 44			
page 13	page 13	page 45			
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			

Simulator Screen 2

Simulation still goes on because we determined pagesize as 64 and it means that 32 virtual pages will be assigned to physical address automatically.

Memory Management				status: STOP	
run step reset exit				time: 260 (ns)	
virtual	physical	virtual	physical		
page 0	page 7	page 32		instruction:	READ
page 1	page 6	page 33		address:	64000
page 2	page 5	page 34		page fault:	NO
page 3	page 4	page 35		virtual page:	25
page 4	page 3	page 36		physical page:	25
page 5	page 2	page 37		R:	0
page 6	page 1	page 38		M:	0
page 7	page 0	page 39		inMemTime:	250
page 8	page 8	page 40		lastTouchTime:	250
page 9	page 9	page 41		low:	64000
page 10	page 10	page 42		high:	67fff
page 11	page 11	page 43			
page 12	page 12	page 44			
page 13	page 13	page 45			
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	page 31		

Simulator Screen 3

Everything still looks normal. We do not have any fault and continues to read.

Memory Management				status: STOP	
run step reset exit				time: 330 (ns)	
virtual	physical	virtual	physical		
page 0		page 32	page 7	instruction:	READ
page 1	page 6	page 33		address:	80000
page 2	page 5	page 34		page fault:	YES
page 3	page 4	page 35		virtual page:	32
page 4	page 3	page 36		physical page:	-1
page 5	page 2	page 37		R:	0
page 6	page 1	page 38		M:	0
page 7	page 0	page 39		inMemTime:	0
page 8	page 8	page 40		lastTouchTime:	0
page 9	page 9	page 41		low:	80000
page 10	page 10	page 42		high:	83fff
page 11	page 11	page 43			
page 12	page 12	page 44			
page 13	page 13	page 45			
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	page 31		

Simulator Screen 4

Now we are out of mapped pages. We got an page fault here. It displayed physical page:-1, it means that page 32 is not mapped to any physical page. It started to search from virtual page 0 and saw the physical page 7 at the beging so it assigned it to there. It is because of FIFO algorithm.

Memory Management				status: STOP	
run	step	reset	exit	time: 490 (ns)	
virtual	physical	virtual	physical		
page 0		page 32	page 7		
page 1		page 33	page 6	instruction: READ	
page 2		page 34	page 5	address: c0000	
page 3		page 35	page 4		
page 4		page 36	page 3	page fault: YES	
page 5		page 37	page 2		
page 6		page 38	page 1	virtual page: 48	
page 7		page 39	page 0	physical page: -1	
page 8		page 40	page 8	R: 0	
page 9		page 41	page 9	M: 0	
page 10		page 42	page 10	inMemTime: 0	
page 11		page 43	page 11	lastTouchTime: 0	
page 12		page 44	page 12	low: c0000	
page 13		page 45	page 13	high: c3fff	
page 14		page 46	page 14		
page 15		page 47	page 15		
page 16		page 48	page 16		
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	page 31		

Simulator Screen 5

It still gives page fault and taking the physical page numbers from the mapped ones.

Memory Management				status: STOP	
run	step	reset	exit	time: 620 (ns)	
virtual	physical	virtual	physical		
page 0		page 32	page 7		
page 1		page 33	page 6	instruction: READ	
page 2		page 34	page 5	address: f4000	
page 3		page 35	page 4		
page 4		page 36	page 3	page fault: YES	
page 5		page 37	page 2		
page 6		page 38	page 1	virtual page: 61	
page 7		page 39	page 0	physical page: -1	
page 8		page 40	page 8	R: 0	
page 9		page 41	page 9	M: 0	
page 10		page 42	page 10	inMemTime: 0	
page 11		page 43	page 11	lastTouchTime: 0	
page 12		page 44	page 12	low: f4000	
page 13		page 45	page 13	high: f7fff	
page 14		page 46	page 14		
page 15		page 47	page 15		
page 16		page 48	page 16		
page 17		page 49	page 17		
page 18		page 50	page 18		
page 19		page 51	page 19		
page 20		page 52	page 20		
page 21		page 53	page 21		
page 22		page 54	page 22		
page 23		page 55	page 23		
page 24		page 56	page 24		
page 25		page 57	page 25		
page 26		page 58	page 26		
page 27		page 59	page 27		
page 28		page 60	page 28		
page 29		page 61	page 29		
page 30	page 30	page 62			
page 31	page 31	page 63	page 31		

Simulator Screen 6

It still gives page fault and taking the physical page numbers from the mapped ones.

Memory Management				status: STOP	
Step				time: 640 (ns)	
virtual	physical	virtual	physical		
page 0		page 32	page 7	instruction:	READ
page 1		page 33	page 6	address:	fc000
page 2		page 34	page 5	page fault:	YES
page 3		page 35	page 4	virtual page:	63
page 4		page 36	page 3	physical page:	-1
page 5		page 37	page 2	R:	0
page 6		page 38	page 1	M:	0
page 7		page 39	page 0	inMemTime:	0
page 8		page 40	page 8	lastTouchTime:	0
page 9		page 41	page 9	low:	fc000
page 10		page 42	page 10	high:	fffff
page 11		page 43	page 11		
page 12		page 44	page 12		
page 13		page 45	page 13		
page 14		page 46	page 14		
page 15		page 47	page 15		
page 16		page 48	page 16		
page 17		page 49	page 17		
page 18		page 50	page 18		
page 19		page 51	page 19		
page 20		page 52	page 20		
page 21		page 53	page 21		
page 22		page 54	page 22		
page 23		page 55	page 23		
page 24		page 56	page 24		
page 25		page 57	page 25		
page 26		page 58	page 26		
page 27		page 59	page 27		
page 28		page 60	page 28		
page 29		page 61	page 29		
page 30		page 62	page 30		
page 31		page 63	page 31		

Simulator Screen 7

It finished and we saw that it gave page fault for virtual pages between 32 and 63.

C. Discussion

When we check the java code we can see that FIFO algorithm runs for the page faults. It was directly considering the order of the mapping process. First mapped page was replaced for the first page fault.

FIFO algorithm did not care about the pages its own order. However, with Optimal Page replacement algorithm, we could making this replacing process depend on the duration of the last use.

Optimal Page Replacement: The idea is simple, for every reference we do following :⁵

1. If referred page is already present, increment hit count.
2. If not present, find if a page that is never referenced in future. If such a page exists, replace this page with new page. If no such page exists, find a page that is referenced farthest in future. Replace this page with new page.

⁵ <https://www.geeksforgeeks.org/optimal-page-replacement-algorithm/>

D. Conclusion

We had chance to observe page replacement process with the FIFO algorithm. We saw that only mapped physical pages can be used by virtual pages. If there is not enough physical pages mapped, existing physical pages are replaced and virtual pages continue to their process. After the exercise we understood better about the page replacement process.

We also observed the some status of pages. We learned that if virtual memory did not mapped over physical memory we get page fault. In real world examples, we know that physical memory is much smaller than virtual memory, and it is obvious that we will get page faults. With this task we had an idea about how to handle with such problems.

E. Output File (tracefile)

1	READ 0 ... okay	33	READ 80000 ... page fault
2	READ 4000 ... okay	34	READ 84000 ... page fault
3	READ 8000 ... okay	35	READ 88000 ... page fault
4	READ c000 ... okay	36	READ 8c000 ... page fault
5	READ 10000 ... okay	37	READ 90000 ... page fault
6	READ 14000 ... okay	38	READ 94000 ... page fault
7	READ 18000 ... okay	39	READ 98000 ... page fault
8	READ 1c000 ... okay	40	READ 9c000 ... page fault
9	READ 20000 ... okay	41	READ a0000 ... page fault
10	READ 24000 ... okay	42	READ a4000 ... page fault
11	READ 28000 ... okay	43	READ a8000 ... page fault
12	READ 2c000 ... okay	44	READ ac000 ... page fault
13	READ 30000 ... okay	45	READ b0000 ... page fault
14	READ 34000 ... okay	46	READ b4000 ... page fault
15	READ 38000 ... okay	47	READ b8000 ... page fault
16	READ 3c000 ... okay	48	READ bc000 ... page fault
17	READ 40000 ... okay	49	READ c0000 ... page fault
18	READ 44000 ... okay	50	READ c4000 ... page fault
19	READ 48000 ... okay	51	READ c8000 ... page fault
20	READ 4c000 ... okay	52	READ cc000 ... page fault
21	READ 50000 ... okay	53	READ d0000 ... page fault
22	READ 54000 ... okay	54	READ d4000 ... page fault
23	READ 58000 ... okay	55	READ d8000 ... page fault
24	READ 5c000 ... okay	56	READ dc000 ... page fault
25	READ 60000 ... okay	57	READ e0000 ... page fault
26	READ 64000 ... okay	58	READ e4000 ... page fault
27	READ 68000 ... okay	59	READ e8000 ... page fault
28	READ 6c000 ... okay	60	READ ec000 ... page fault
29	READ 70000 ... okay	61	READ f0000 ... page fault
30	READ 74000 ... okay	62	READ f4000 ... page fault
31	READ 78000 ... okay	63	READ f8000 ... page fault
32	READ 7c000 ... okay	64	READ fc000 ... page fault

Figure 7: Output File

4. RUNNING THE SIMULATION 2

A. Configuration and Commands Files

A.1 Configuration File

As It was described in the Task Description part, we created the memory.conf file with 8 pages. We will observe the pages from 0 to 7 in both virtual and physical pages. We set addressradix to 10 to be able to observe the values in decimal, we could also observe in hexadecimal and 8 bit representation. Pagesize set to 16 because we only have 16 pages in total, we will not have mess on the simulator screen. Rest of the settings remain same.

```

1 // memset virt page # physical page # R (read from) M (modified) inMemTime (ns) lastTouchTime (ns)
2 memset 0 0 0 0 0 0
3 memset 1 1 0 0 0 0
4 memset 2 2 0 0 0 0
5 memset 3 3 0 0 0 0
6 memset 4 4 0 0 0 0
7 memset 5 5 0 0 0 0
8 memset 6 6 0 0 0 0
9 memset 7 7 0 0 0 0
10
11
12 // enable_logging 'true' or 'false'
13 // When true specify a log_file or leave blank for stdout
14 enable_logging true
15
16 // log_file <FILENAME>
17 // Where <FILENAME> is the name of the file you want output
18 // to be print to.
19 log_file tracefile
20
21 // page size, defaults to 2^14 and cannot be greater than 2^26
22 // pagesize <single page size (base 10)> or <'power' num (base 2)>
23 pagesize 16384
24
25 // addressradix sets the radix in which numerical values are displayed
26 // 2 is the default value
27 // addressradix <radix>
28 addressradix 10
29
30 // numpages sets the number of pages (physical and virtual)
31 // 64 is the default value
32 // numpages must be at least 2 and no more than 64
33 // numpages <num>
34 numpages 16

```

Figure 8: Configuration File For The Task

A.2 Commands File

We also made changes in the commands file. To have a better understanding, we will assign address numbers for the half of the operations and for the rest we will use random to observe ‘page fault’. It may not have fault also but we wish for the random numbers to go beyond the mapped pages.

```

// Enter READ/WRITE commands into this file
// READ <OPTIONAL number type: bin/hex/oct> <virtual memory address or random>
// WRITE <OPTIONAL number type: bin/hex/oct> <virtual memory address or random>
READ 0
READ 1
READ 2
READ 3
READ random
READ random
READ random
READ random
WRITE 0
WRITE 1
WRITE 2
WRITE 3
WRITE random
WRITE random
WRITE random
WRITE random

```

Figure 9: Commands File For The Task

B. Simulation

Simulation will map the pages first. We know that each pagesize will be 16384. We can conclude that addresses from 0 to 131071 will be mapped. We can see the low and high address values for the page 0 from the screenshot 1.

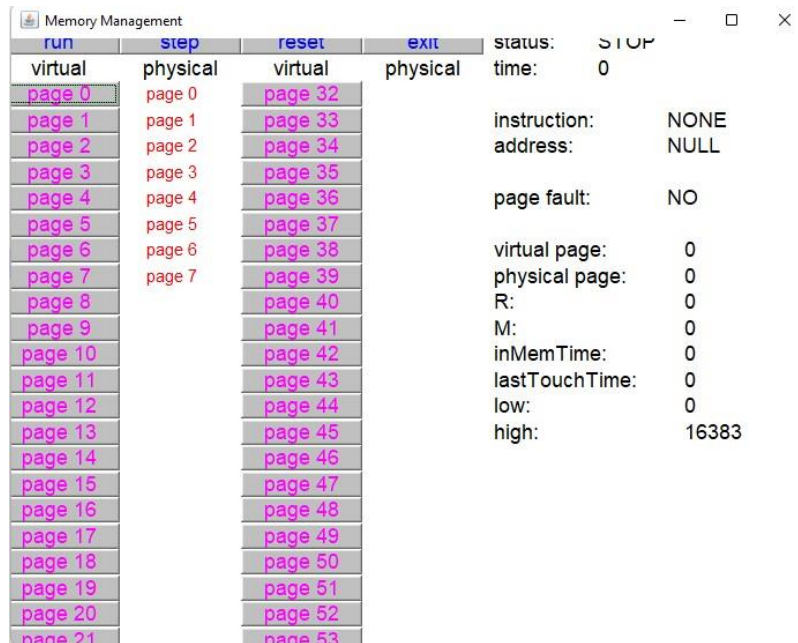
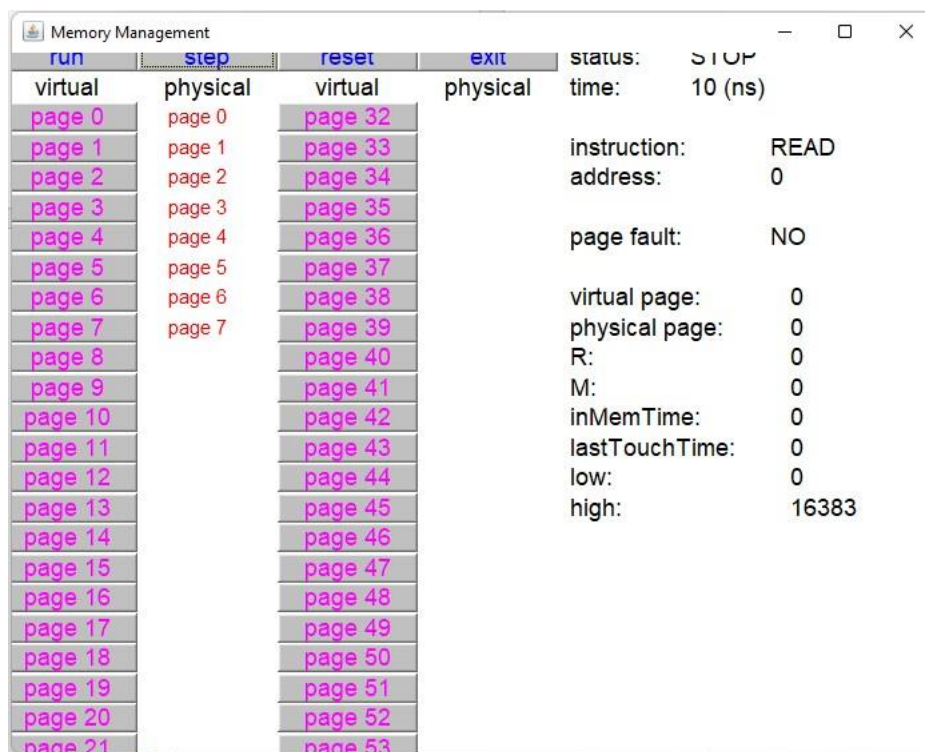
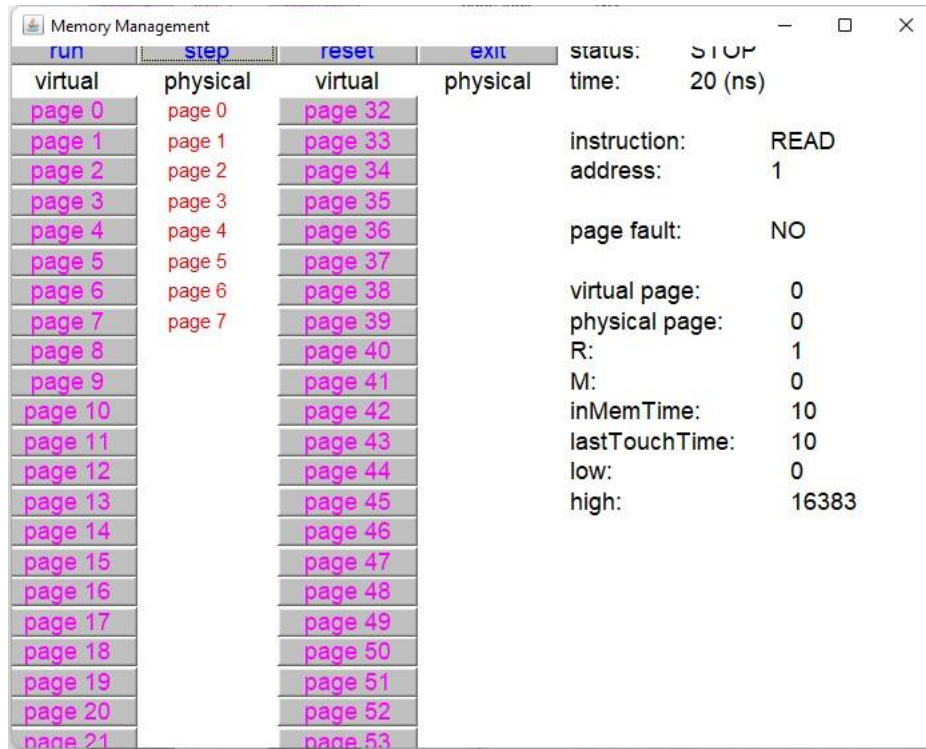


Figure 10: initial Screen of Simulator



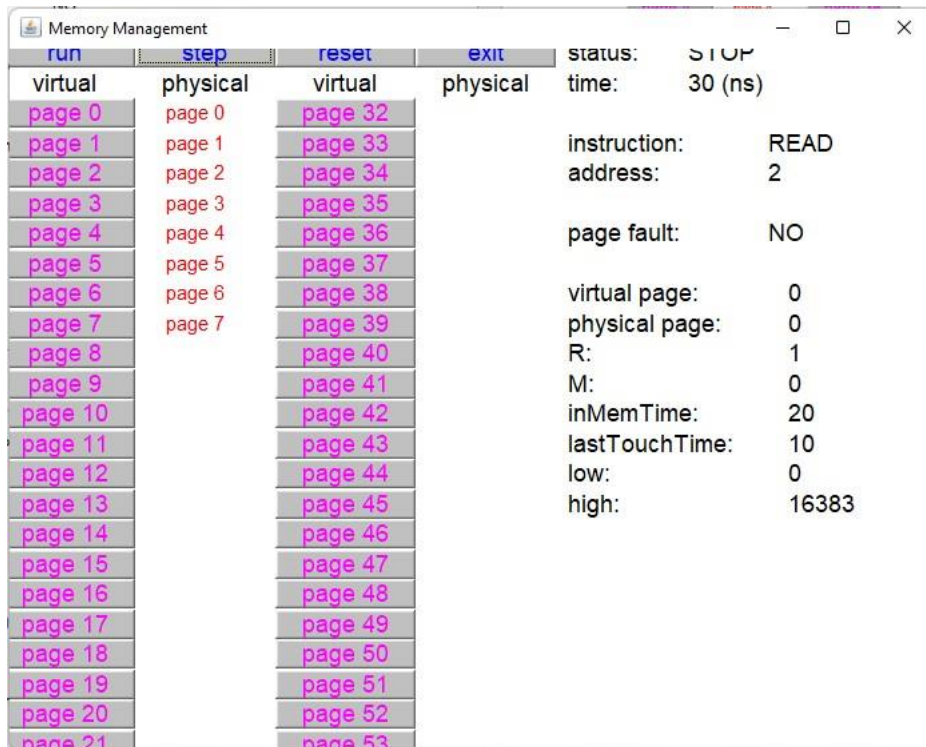
STEP 1

We can not see any page fault, It reads the 0 successfully.



STEP 2

Again we read the page 1 and it is successful. We initialize the first 4 page in command file and because of it inMemTime is increasing linearly by the time. But last touch is reset after every step.



STEP 3 Again

we read the page 2 and it is successful.

Memory Management					
run	step	reset	exit	status:	STOP
virtual	physical	virtual	physical	time:	40 (ns)
page 0	page 0	page 32		instruction:	READ
page 1	page 1	page 33		address:	3
page 2	page 2	page 34		page fault:	NO
page 3	page 3	page 35		virtual page:	0
page 4	page 4	page 36		physical page:	0
page 5	page 5	page 37		R:	1
page 6	page 6	page 38		M:	0
page 7	page 7	page 39		inMemTime:	30
page 8		page 40		lastTouchTime:	10
page 9		page 41		low:	0
page 10		page 42		high:	16383
page 11		page 43			
page 12		page 44			
page 13		page 45			
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			

STEP 4 Again

we read the page 3 and it is successful.

Memory Management					
run	step	reset	exit	status:	STOP
virtual	physical	virtual	physical	time:	50 (ns)
page 0	page 0	page 32		instruction:	READ
page 1	page 1	page 33		address:	105896
page 2	page 2	page 34		page fault:	NO
page 3	page 3	page 35		virtual page:	6
page 4	page 4	page 36		physical page:	6
page 5	page 5	page 37		R:	0
page 6	page 6	page 38		M:	0
page 7	page 7	page 39		inMemTime:	40
page 8		page 40		lastTouchTime:	40
page 9		page 41		low:	98304
page 10		page 42		high:	114687
page 11		page 43			
page 12		page 44			
page 13		page 45			
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			

STEP 5

Remember from the commands file, we set *random* and it return page 6 as result. We were lucky that also page 6 was mapped with memory.conf file and we do not have any fault here.

Memory Management				status:	STOP
run	step	reset	exit	time:	60 (ns)
virtual	physical	virtual	physical		
page 0		page 32		instruction:	READ
page 1	page 1	page 33		address:	190883
page 2	page 2	page 34		page fault:	YES
page 3	page 3	page 35		virtual page:	11
page 4	page 4	page 36		physical page:	-1
page 5	page 5	page 37		R:	0
page 6	page 6	page 38		M:	0
page 7	page 7	page 39		inMemTime:	0
page 8		page 40		lastTouchTime:	0
page 9		page 41		low:	180224
page 10		page 42		high:	196607
page 11	page 0	page 43			
page 12		page 44			
page 13		page 45			
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			

STEP 6

This time *random* returned page 11 and it gives us error because we did not map the page 11.

Memory Management				status:	STOP
run	step	reset	exit	time:	70 (ns)
virtual	physical	virtual	physical		
page 0		page 32		instruction:	READ
page 1	page 1	page 33		address:	195696
page 2	page 2	page 34		page fault:	NO
page 3	page 3	page 35		virtual page:	11
page 4	page 4	page 36		physical page:	0
page 5	page 5	page 37		R:	0
page 6	page 6	page 38		M:	0
page 7	page 7	page 39		inMemTime:	10
page 8		page 40		lastTouchTime:	10
page 9		page 41		low:	180224
page 10		page 42		high:	196607
page 11	page 0	page 43			
page 12		page 44			
page 13		page 45			
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			

STEP 7

Now simulator start from 0 and look for a physical address to assign page 11. Page 0 was free and it used page 0 as physical address.

Memory Management					
run	step	reset	exit	status:	STOP
virtual	physical	virtual	physical	time:	80 (ns)
page 0		page 32		instruction:	READ
page 1	page 1	page 33		address:	195696
page 2	page 2	page 34		page fault:	NO
page 3	page 3	page 35		virtual page:	11
page 4	page 4	page 36		physical page:	0
page 5	page 5	page 37		R:	1
page 6	page 6	page 38		M:	0
page 7	page 7	page 39		inMemTime:	20
page 8		page 40		lastTouchTime:	10
page 9		page 41		low:	180224
page 10		page 42		high:	196607
page 11	page 0	page 43			
page 12		page 44			
page 13		page 45			
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			

STEP 8

After, it was be able to read. And there is no fault. And R value which indicates the reading status become 1.

Memory Management					
run	step	reset	exit	status:	STOP
virtual	physical	virtual	physical	time:	90 (ns)
page 0	page 1	page 32		instruction:	WRITE
page 1		page 33		address:	0
page 2	page 2	page 34		page fault:	YES
page 3	page 3	page 35		virtual page:	0
page 4	page 4	page 36		physical page:	-1
page 5	page 5	page 37		R:	0
page 6	page 6	page 38		M:	0
page 7	page 7	page 39		inMemTime:	0
page 8		page 40		lastTouchTime:	0
page 9		page 41		low:	0
page 10		page 42		high:	16383
page 11	page 0	page 43			
page 12		page 44			
page 13		page 45			
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			

STEP 9

Memory Management

run	step	reset	exit	status:	STOP
virtual	physical	virtual	physical	time:	100 (ns)
page 0	page 1	page 32		instruction:	WRITE
page 1		page 33		address:	1
page 2	page 2	page 34		page fault:	NO
page 3	page 3	page 35		virtual page:	0
page 4	page 4	page 36		physical page:	1
page 5	page 5	page 37		R:	0
page 6	page 6	page 38		M:	0
page 7	page 7	page 39		inMemTime:	10
page 8		page 40		lastTouchTime:	10
page 9		page 41		low:	0
page 10		page 42		high:	16383
page 11	page 0	page 43			
page 12		page 44			
page 13		page 45			
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			

STEP 10

Memory Management

run	step	reset	exit	status:	STOP
virtual	physical	virtual	physical	time:	110 (ns)
page 0	page 1	page 32		instruction:	WRITE
page 1		page 33		address:	2
page 2	page 2	page 34		page fault:	NO
page 3	page 3	page 35		virtual page:	0
page 4	page 4	page 36		physical page:	1
page 5	page 5	page 37		R:	0
page 6	page 6	page 38		M:	1
page 7	page 7	page 39		inMemTime:	20
page 8		page 40		lastTouchTime:	10
page 9		page 41		low:	0
page 10		page 42		high:	16383
page 11	page 0	page 43			
page 12		page 44			
page 13		page 45			
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			

STEP 11

Memory Management				status: STOP	
run	step	reset	exit	time:	120 (ns)
virtual	physical	virtual	physical		
page 0	page 1	page 32		instruction:	WRITE
page 1		page 33		address:	3
page 2	page 2	page 34		page fault:	NO
page 3	page 3	page 35		virtual page:	0
page 4	page 4	page 36		physical page:	1
page 5	page 5	page 37		R:	0
page 6	page 6	page 38		M:	1
page 7	page 7	page 39		inMemTime:	30
page 8		page 40		lastTouchTime:	10
page 9		page 41		low:	0
page 10		page 42		high:	16383
page 11	page 0	page 43			
page 12		page 44			
page 13		page 45			
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			

STEP 12

Memory Management				status: STOP	
run	step	reset	exit	time:	130 (ns)
virtual	physical	virtual	physical		
page 0	page 1	page 32		instruction:	WRITE
page 1		page 33		address:	132377
page 2		page 34		page fault:	YES
page 3	page 3	page 35		virtual page:	8
page 4	page 4	page 36		physical page:	-1
page 5	page 5	page 37		R:	0
page 6	page 6	page 38		M:	0
page 7	page 7	page 39		inMemTime:	0
page 8	page 2	page 40		lastTouchTime:	0
page 9		page 41		low:	131072
page 10		page 42		high:	147455
page 11	page 0	page 43			
page 12		page 44			
page 13		page 45			
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			

STEP 13

Memory Management				status: STOP	
run	step	reset	exit	time:	140 (ns)
virtual	physical	virtual	physical		
page 0	page 1	page 32		instruction:	WRITE
page 1		page 33		address:	77261
page 2		page 34		page fault:	NO
page 3	page 3	page 35		virtual page:	4
page 4	page 4	page 36		physical page:	4
page 5	page 5	page 37		R:	0
page 6	page 6	page 38		M:	0
page 7	page 7	page 39		inMemTime:	130
page 8	page 2	page 40		lastTouchTime:	130
page 9		page 41		low:	65536
page 10		page 42		high:	81919
page 11	page 0	page 43			
page 12		page 44			
page 13		page 45			
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			

STEP 14

Memory Management				status: STOP	
run	step	reset	exit	time:	150 (ns)
virtual	physical	virtual	physical		
page 0	page 1	page 32		instruction:	WRITE
page 1		page 33		address:	77261
page 2		page 34		page fault:	NO
page 3	page 3	page 35		virtual page:	4
page 4	page 4	page 36		physical page:	4
page 5	page 5	page 37		R:	0
page 6	page 6	page 38		M:	1
page 7	page 7	page 39		inMemTime:	140
page 8	page 2	page 40		lastTouchTime:	10
page 9		page 41		low:	65536
page 10		page 42		high:	81919
page 11	page 0	page 43			
page 12		page 44			
page 13		page 45			
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			

STEP 15

Memory Management				status: STOP	
				time: 160 (ns)	
virtual	physical	virtual	physical	instruction: WRITE	
page 0	page 1	page 32		address: 205731	
page 1		page 33		page fault: YES	
page 2		page 34		virtual page: 12	
page 3		page 35		physical page: -1	
page 4	page 4	page 36		R: 0	
page 5	page 5	page 37		M: 0	
page 6	page 6	page 38		inMemTime: 0	
page 7	page 7	page 39		lastTouchTime: 0	
page 8	page 2	page 40		low: 196608	
page 9		page 41		high: 212991	
page 10		page 42			
page 11	page 0	page 43			
page 12	page 3	page 44			
page 13		page 45			
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			

STEP 16

C. Discussion

FIFO algorithm acts so basic and directly takes the first physical address as new address during page fault.

We can say the same arguments as previous simulation.

D. Cocnclusion

We had a more clear view over the behaviour of the simulator. There was more clear explanation for the FIFO algorithm.

E. Output File (tracefile)

```
READ 0 ... okay
READ 1 ... okay
READ 2 ... okay
READ 3 ... okay
READ 105896 ... okay
READ 190883 ... page fault
READ 195696 ... okay
READ 195696 ... okay
WRITE 0 ... page fault
WRITE 1 ... okay
WRITE 2 ... okay
WRITE 3 ... okay
WRITE 132377 ... page fault
WRITE 77261 ... okay
WRITE 77261 ... okay
WRITE 205731 ... page fault
```

Figure 11: Output File (tracefile)

5. COMMENT

The simulator which exist in '[/ Operating Systems \(EOPSY\) - course homepage / \(pw.edu.pl\)](http://Operating Systems (EOPSY) - course homepage / (pw.edu.pl))' has problems about displaying the virtual and physical pages at the beginning. Working version can be found at '[Moss | Memory Management Simulator | User Guide \(ontko.com\)](http://Moss | Memory Management Simulator | User Guide (ontko.com))'.