EOPSY

MEMORY MANAGEMENT

SEBASTIAN GRZELAK

General idea:

The aim of the laboratory task was to configure and run a memory management simulator. We had to edit two files "commands" and "memory.conf" by setting a proper configuration. We observed the results in a file named "traceline" and on a graphical simulator which allowed to watched memory mapping in real time.

The task was to map any 8 pages of physical memory to the first 8 pages of virtual memory. In real world physical memory addresses are not mapped in the same addresses as in virtual memory it means that such a simulation gives as more realistic scenario. Then the case was to read each of the 64 virutal pages from one vifrual memory address and try to find out which virtual memory addresses will cause page faults. Additional question was, which kind of replacement algorithm was used in this simulation.

Files configuration:

memory.conf

```
memset 0 1 0 0 0 0
memset 1 5 0 0 0 0
memset 2 8 0 0 0 0
memset 3 14 0 0 0 0
memset 4 17 0 0 0 0
memset 5 23 0 0 0 0
memset 6 26 0 0 0 0
memset 7 27 0 0 0 0

enable_logging true
log_file ../../tracefile
pagesize 16384
addressradix 10
numpages 64
```

Parameters description:

Keyword	Description
memset	peforms mapping between virtual page and physical page
enable_logging	‘true’ or ‘false’ turn on / off logs
log_file	path to log file and name
pagesize	Page size, default 2^14 and cannot be greater than 2^26
addressradix	Sets the radix in which numerical values are displayed
numpages	Sets number of pages (physical and virtual)

In my case the **pagesize** is **16384** and this is important information for proper configuration ”**commands**” file.

In “**commands”** file we set addresses of each virtual page that we will be read. Because the size of page is **16384** address of each page will be (**16384** *i) where i = 0,1,2...63.

commands

```
READ 0
READ 16384
READ 32768
READ 49152
READ 65536
READ 81920
READ 98304
READ 114688
READ 131072
READ 147456
READ 163840
READ 180224
READ 196608
READ 212992
READ 229376
READ 245760
READ 262144
READ 278528
READ 294912
READ 311296
READ 327680
READ 344064
READ 360448
READ 376832
READ 393216
READ 409600
READ 425984
READ 442368
READ 458752
READ 475136
READ 491520
READ 507904
READ 524288
READ 540672
READ 557056
READ 573440
READ 589824
READ 606208
READ 622592
READ 638976
READ 655360
READ 671744
READ 688128
READ 704512
READ 720896
READ 737280
READ 753664
READ 770048
READ 786432
READ 802816
READ 819200
READ 835584
READ 851968
READ 868352
READ 884736
READ 901120
READ 917504
READ 933888
READ 950272
READ 966656
READ 983040
READ 999424
READ 1015808
READ 1032192
```

My mapping:

Virutal Page	PhysicalPage
0	1
1	5
2	8
3	14
4	17
5	23
6	26
7	27

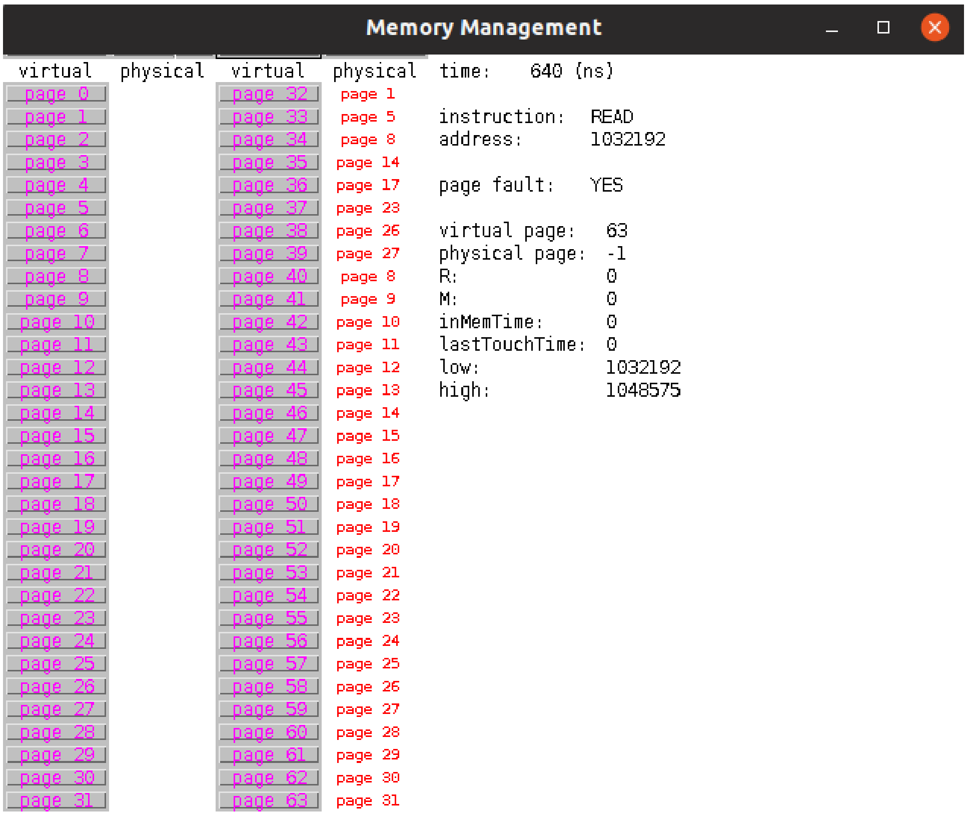
Because of the number of physical pages in simulation was 32 and the number of virtual memory pages was set by me to 64 the page fault can be observed. In case when the number of virtual memory pages and physical pages is equal the page fault would not be observe.

Predict page faults:

First page fault occurs when want to map virutal memory address to physical memory address but there is no more avaliable space in physical memory. It means that some of memoery address have to be swap to our external memory to make space for currently need it memory.

tracefile
READ 0 ... okay
READ 16384 ... okay
READ 32768 ... okay
READ 49152 ... okay
READ 65536 ... okay
READ 81920 ... okay
READ 98304 ... okay
READ 114688 ... okay
READ 131072 ... okay
READ 147456 ... okay
READ 163840 ... okay
READ 180224 ... okay
READ 196608 ... okay
READ 212992 ... okay
READ 229376 ... okay
READ 245760 ... okay
READ 262144 ... okay
READ 278528 ... okay
READ 294912 ... okay
READ 311296 ... okay
READ 327680 ... okay
READ 344064 ... okay
READ 360448 ... okay
READ 376832 ... okay
READ 393216 ... okay
READ 409600 ... okay
READ 425984 ... okay
READ 442368 ... okay
READ 458752 ... okay
READ 475136 ... okay
READ 491520 ... okay
READ 507904 ... okay
READ 524288 ... page fault
READ 540672 ... page fault
READ 557056 ... page fault
READ 573440 ... page fault
READ 589824 ... page fault
READ 606208 ... page fault
READ 622592 ... page fault
READ 638976 ... page fault
READ 655360 ... page fault
READ 671744 ... page fault
READ 688128 ... page fault
READ 704512 ... page fault
READ 720896 ... page fault
READ 737280 ... page fault
READ 753664 ... page fault
READ 770048 ... page fault
READ 786432 ... page fault
READ 802816 ... page fault
READ 819200 ... page fault
READ 835584 ... page fault
READ 851968 ... page fault
READ 868352 ... page fault
READ 884736 ... page fault
READ 901120 ... page fault
READ 917504 ... page fault
READ 933888 ... page fault
READ 950272 ... page fault
READ 966656 ... page fault
READ 983040 ... page fault
READ 999424 ... page fault
READ 1015808 ... page fault
READ 1032192 ... page fault

Finished simulation:



Memory Management

virtual	physical	virtual	physical
page 0		page 32	page 1
page 1		page 33	page 5
page 2		page 34	page 8
page 3		page 35	page 14
page 4		page 36	page 17
page 5		page 37	page 23
page 6		page 38	page 26
page 7		page 39	page 27
page 8		page 40	page 8
page 9		page 41	page 9
page 10		page 42	page 10
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

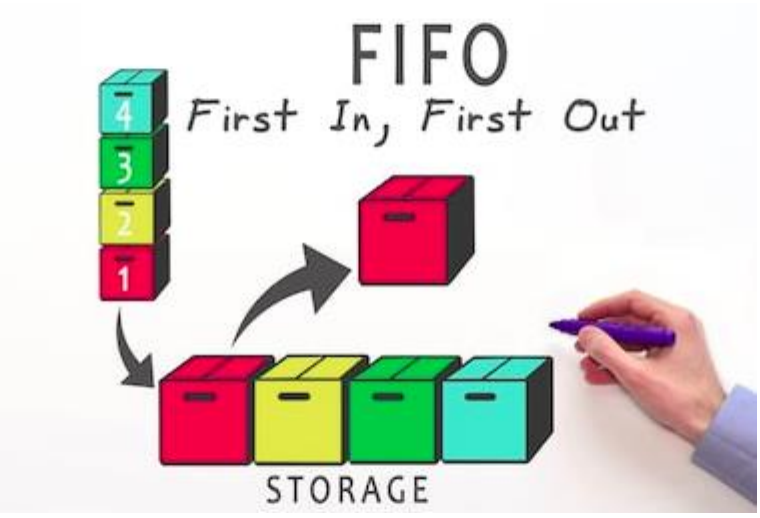
```
time:     640 (ns)

instruction:   READ
address:       1032192

page fault:    YES

virtual page:    63
physical page:   -1
R:               0
M:               0
inMemTime:       0
lastTouchTime:   0
low:             1032192
high:            1048575
```

In this simulation was used FIFO algorithm which represents below image:



As the picture shows the FIFO algorithm work in such a way that the first added element to the queue is also the first element which is remove when it is necessary.

FIFO algorithm is not very efficient because it does not make a difference between pages frequently. Much better algorithm might be e.g. LRU or priority queue.