EOPSY Lab 4 - Memory Management

Author: Kacper Kamieniarz (293065)

The Configuration File

First step was to setup the memory.conf file to specify the the initial content of the virtual memory map, specifically to map any 8 pages of physical memory to the first 8 pages of virtual memory. I did it in the following way:

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

Apart from that I changed the radix in which numerical values are displayed to 10, so that it is in decimal. I left the rest of the configuration information unchanged.

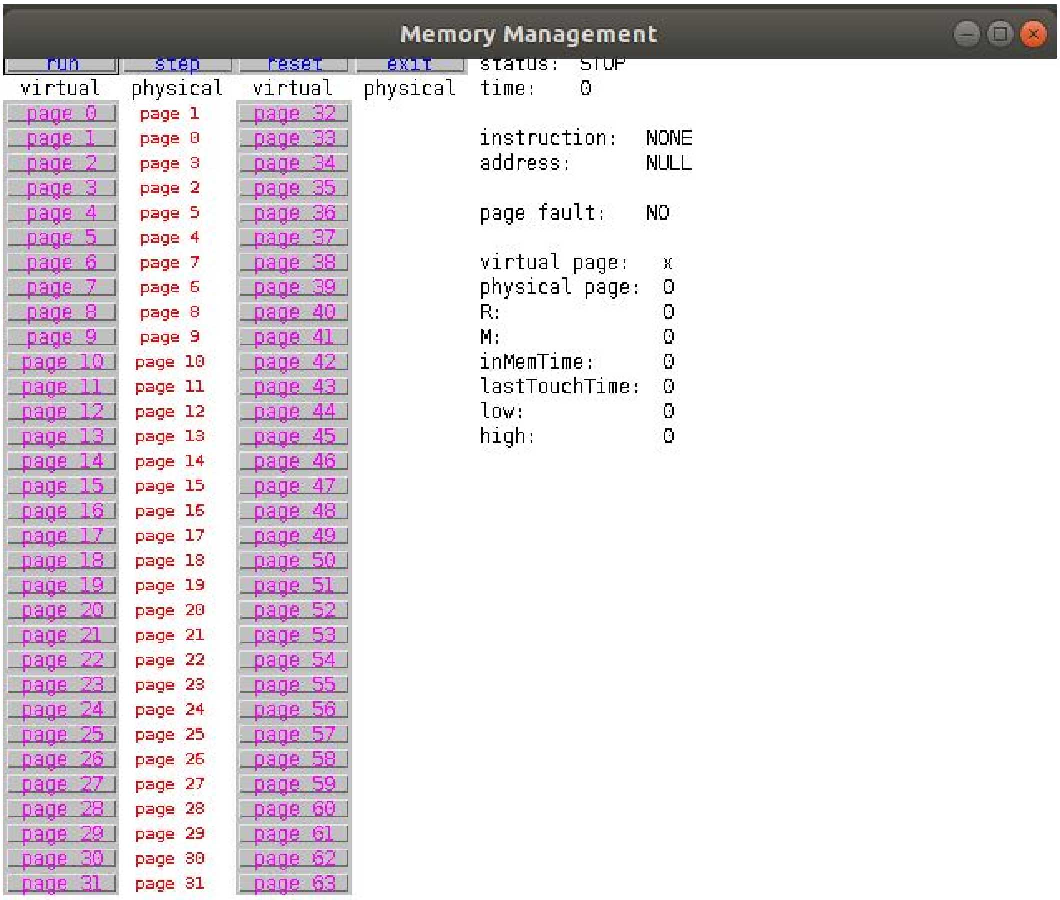
The Command File

Next step was to setup the command file to specify a sequence of memory instructions to be performed, specifically to read from one virtual memory address on each of the 64 virtual pages. As the pagesize attribute is set to 16384 in the configuration file as default, I decided to keep it that way. Therefore to read from one address on each of the 64 pages, I decided to read from every first address on each of the pages. So starting from 0 I defined 64 READ operations on each multiple of 16384 in the following way.

```
READ 0
READ 16384
READ 32768
.
.
.
READ 999424
READ 1015808
READ 1032192
```

Simulation initial state

After launching the simulator, we see the initial state of the memory pages mapping. Virtual pages 0-7, are mapped to the physical pages that we specified in the Configuration file.



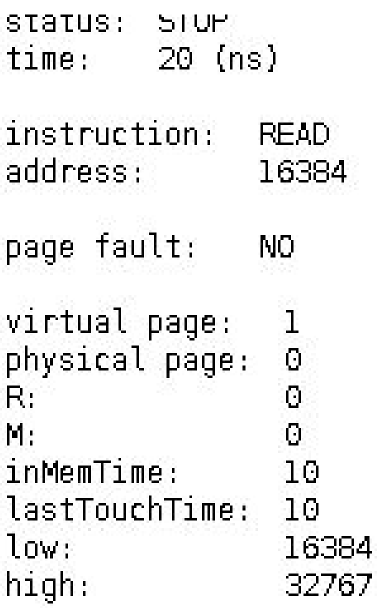
virtual	physical	virtual	physical
page 0	page 1	page 32	
page 1	page 0	page 33	
page 2	page 3	page 34	
page 3	page 2	page 35	
page 4	page 5	page 36	
page 5	page 4	page 37	
page 6	page 7	page 38	
page 7	page 6	page 39	
page 8	page 8	page 40	
page 9	page 9	page 41	
page 10	page 10	page 42	
page 11	page 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	

```
status:  STOP
time:    0

instruction:  NONE
address:      NULL

page fault:   NO

virtual page:   x
physical page:  0
R:              0
M:              0
inMemTime:      0
lastTouchTime:  0
low:            0
high:           0
```

What’s interesting is that the virtual pages 8-31 are mapped to physical pages with the same number. Pages 32-63 have no mapping at this point.

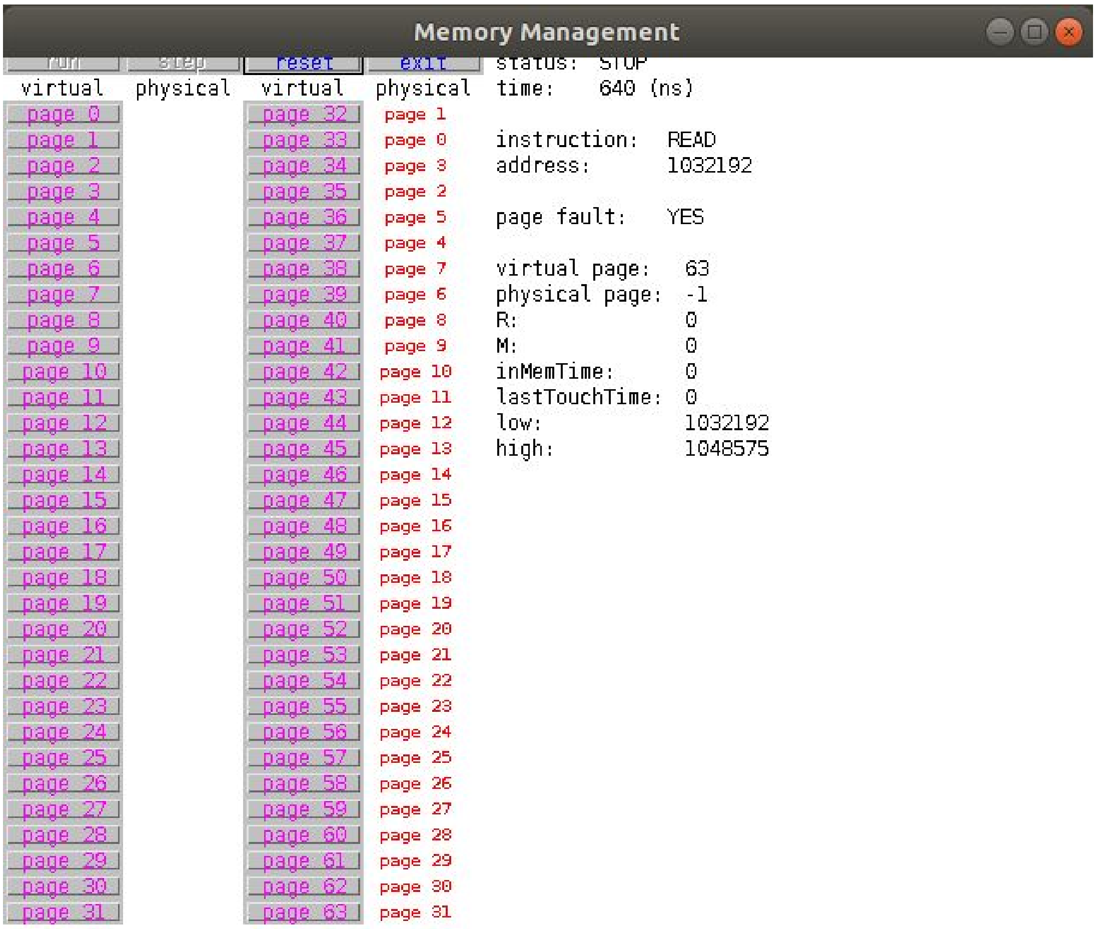
Simulation

After stepping through the pages 0-31 one operation at a time, we can see that at each of the steps there is no page fault present and time of simulations increases 10 ns after every step.



```
status:  STOP
time:    20 (ns)

instruction:  READ
address:      16384

page fault:   NO

virtual page:   1
physical page:  0
R:              0
M:              0
inMemTime:      10
lastTouchTime:  10
low:            16384
high:           32767
```

At the virtual page 32 first page fault appeared. This was easy to predict as this virtual page did not have any mapping to a physical page. **Page fault** occurs whenever there has been a reference to a page which is currently absent in the main memory, so in our case to a virtual page which does not have a mapping to a physical page. After the failed attempt to read virtual page 32, the simulator maps physical page 1 to it.



```
virtual  physical  virtual  physical   status:  STOP
                                       time:    640 (ns)
page 0             page 32  page 1
page 1             page 33  page 0     instruction:  READ
page 2             page 34  page 3     address:      1032192
page 3             page 35  page 2
page 4             page 36  page 5     page fault:   YES
page 5             page 37  page 4
page 6             page 38  page 7     virtual page:   63
page 7             page 39  page 6     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:            1032192
page 13            page 45  page 13    high:           1048575
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
```

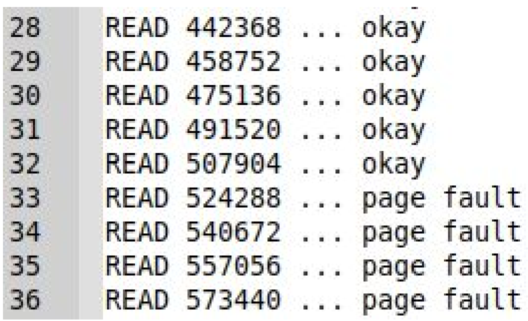
After the simulation has finished, we can observe that each of the virtual pages 32-63 has been mapped to a physical page in the same order as pages 0-31. This indicates that First In - First Out page replacement algorithm was used.

Page Replacement Algorithm

The algorithm used in the simulation is First In - First Out. In this replacement algorithm, all the pages in the memory are kept in a queue. As the name suggests, those pages that were least recently used, are at the top of the queue, so for instance after the failed attempt to read memory from virtual page 32, the physical page 1 is assigned which was previously assigned to the virtual page 0 (as a first one to be assigned in the simulation). So the physical page was the first in and now when the system needs to assign a physical page to a virtual page it is the first one out.

The trace file

The tracefile contains a log of the operations since the simulation started together with the statuses. Just as a confirmation of the simulation process we can see that the first 32 READ operations were successful, and all of the remaining operations statuses indicate page fault.



```
28    READ 442368 ... okay
29    READ 458752 ... okay
30    READ 475136 ... okay
31    READ 491520 ... okay
32    READ 507904 ... okay
33    READ 524288 ... page fault
34    READ 540672 ... page fault
35    READ 557056 ... page fault
36    READ 573440 ... page fault
```