EPOSY Lab 4 Report

Author: Jan Radzimiński

Index Number: 293052

---

1. Simulator Configuration

The task was to configure the simulator to map (any) 8 pages of physical memory into first 8 pages of virtual memory. Then, it should read from one address on each of 64 pages.

To set up first part of exercise I edited "*memory.conf*" file to map pages as specified in the task:

```
// 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
```

I have also modified file so that numerical values are displayed in decimal format:

```
addressradix 10
```

Rest of the variables in the file I left on the default value:

```
enable_logging true
log_file tracefile
pagesize 16384
numpages 64
```

To perform read operation on one address on all 64 pages i configured *“commands”* file - I used Microsoft Excel to generate addresses (took every 16384th - page size - value starting from 0):

```
// 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 16384
READ 32768
READ 49152
READ 65536
READ 81920
READ 98304
READ 114688
READ 131072
READ 147456
...
READ 917504
READ 933888
READ 950272
READ 966656
READ 983040
READ 999424
READ 1015808
READ 1032192
```

2. Simulation results

After compiling and running simulation it is visible that the first 8 pages were mapped as planned. Apart from that, rest of first 32 virtual pages (from 8 to 31) were mapped as well (probably as the default by simulator).

The page fault appears when the virtual page, that hasn’t been mapped to a physical page yet, is being accessed. When this happens, the page is being mapped to the first physical page in the queue (depending on used page replacement algorithm).

Since in given simulation the first 32 pages are being mapped, while the rest are not, and we are successively accessing all 64 pages, the page faults should appears starting from 32th page (since pages from 0 to 31 are mapped).
This is confirmed in simulation - when page 32 (and so on) is read, the page fault appears, and it is mapped to the physical page 0 (since its first in queue due to used algorithm - more on that later in report).
All of these results are visible in *"tracefile"* file generated by the simulation:

```
READ 0 ... okay
READ 16384 ... okay
READ 32768 ... okay
READ 49152 ... okay
...
READ 507904 ... okay
READ 524288 ... page fault
READ 540672 ... page fault
READ 557056 ... page fault
READ 573440 ... page fault
...
READ 999424 ... page fault
READ 1015808 ... page fault
READ 1032192 ... page fault
```

Address `524288` belongs to page 64, so everything went according to my predictions.

3. Used Algorithm

After going through the simulator files I found following comment in *"PageFault.java"* file:

```
 [...]
   * 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>
 [...]
```

Therefore, the algorithm used for page replacement in case of page fault is FIFO - First In First Out. In this algorithm, as the name suggests, the first page that was mapped (the "oldest") one (first-in) is being used as the replacement (first-out).

Results obtained in the simulator confirm this - after page fault appears on page 64, the physical page that was mapped at the beginning (page 0) is being mapped. After that, while next faults appear, the next pages (1, 2 and so on) are being mapped. The physical page 0 was mapped at the beginning, so the algorithm FIFO is working.

Remark: Files "memory.conf", "commands" and "tracefile" are attached to the report.