EOPSY Task 4 by Fernando Santana Falcón

Files are attached as justification.

1. 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?
In this question I have edited two files ***memory.conf*** and ***Commands***:

I have modified ***memory.conf*** file for map 8 pages of physical memory to the first 8 pages of virtual memory from virtual page number 0 to number 7. The pages from 8 to 63 have been mapped automatically. I continue using the default number of pages 64 however I changed the pagesize value to 64 so command addresses interval will be 64. The log_file is ***tracefile*** where we can find the simulation results. I have changed addressradix to 10 changing the radix to decimal.

I have introduced in the ***Commands*** file the READ commands with the memory addresses in decimal.

After compile and run the *make* command we can observe the Memory Management window and the ***tracefile*** with simulation results. As expected, we have only 32 physical pages from 0 to 31.
We can observe in the memory management window the 31st virtual page has high at 2047 and the 32nd at 2111. We can observe that starting from page 32 the physical page is assigned as -1, so the tracefile shows it as page fault address between 2111 and 4032. This occurs because there is no more space in physical memory.
The algorithm used is FIFO (First In First Out) Page Replacement which works replacing the oldest page with the first not assigned page waiting. The algorithm searches through the current memory checking the time. This is not the most efficient algorithm because it does not make a difference between pages frequently used.

memory content:

```
// 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 64

// 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
```

tracerfile content:

```
READ 63 ... okay
READ 127 ... okay
READ 191 ... okay
READ 255 ... okay
READ 319 ... okay
READ 383 ... okay
READ 447 ... okay
READ 511 ... okay
READ 639 ... okay
READ 703 ... okay
READ 767 ... okay
READ 831 ... okay
READ 895 ... okay
READ 959 ... okay
READ 1023 ... okay
READ 1087 ... okay
READ 1151 ... okay
READ 1215 ... okay
READ 1279 ... okay
READ 1343 ... okay
READ 1407 ... okay
READ 1471 ... okay
READ 1535 ... okay
READ 1599 ... okay
READ 1663 ... okay
READ 1727 ... okay
READ 1791 ... okay
READ 1855 ... okay
READ 1919 ... okay
READ 1983 ... okay
```

READ 2047 ... okay
READ 2111 ... page fault
READ 2175 ... page fault
READ 2239 ... page fault
READ 2303 ... page fault
READ 2367 ... page fault
READ 2431 ... page fault
READ 2495 ... page fault
READ 2559 ... page fault
READ 2623 ... page fault
READ 2687 ... page fault
READ 2751 ... page fault
READ 2815 ... page fault
READ 2879 ... page fault
READ 2943 ... page fault
READ 3007 ... page fault
READ 3071 ... page fault
READ 3135 ... page fault
READ 3199 ... page fault
READ 3263 ... page fault
READ 3327 ... page fault
READ 3391 ... page fault
READ 3455 ... page fault
READ 3519 ... page fault
READ 3583 ... page fault
READ 3647 ... page fault
READ 3711 ... page fault
READ 3775 ... page fault
READ 3839 ... page fault
READ 3903 ... page fault
READ 3967 ... page fault
READ 4031 ... page fault
READ 4032 ... page fault

2. Locate in the sources and describe to the instructor the page replacement algorithm.
As I can see observe in ***PageFault.java*** file, there is a FIFO Page Replacement algorithm. This algorithm will be called by Memory Management when there is a page fault. The algorithm is implemented with a while loop to look for through the current memory contents for a candidate replacement page, attending to First In First Out function. The algorithm is used to find the proper page making sure that virtual page number is not exceeded.