04.05.2021, Warsaw
Wojciech Marosek
ID: 295818

EOPSY | Lab 4

Task:

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.

Basic description:

Replacement algorithm - is using to decide which page needs to be replaced when new page comes in

What page replacement algorithm is being used?

We can mention page replacement algorithm like:

- **First In First Out (FIFO)** – the simples page replacement algorithm, operating system keeps track of all pages in the memory queue, the oldest page in front of queue. If some pages need to be replaced, the first is selected for removal
- **Optimal Page replacement**- are replaced which would not be used for the longest duration of time in the future. This algorithm is perfect, however is not possible to use in practice, because the operating system doesn't know future requests.
- **Second-Chance page**- A simple modification to FIFO that avoids the problem of throwing out a heavily used page is to inspect the R bit of the oldest page. If it is 0, the page is both old and unused, so it is replaced immediately. If the R bit is 1, the bit is cleared, the page is put onto the end of the list of pages, and its load time is updated as though it had just arrived in memory. Then the search continues.
- **Least Recently Used**- page will be replaced which is least recently used.

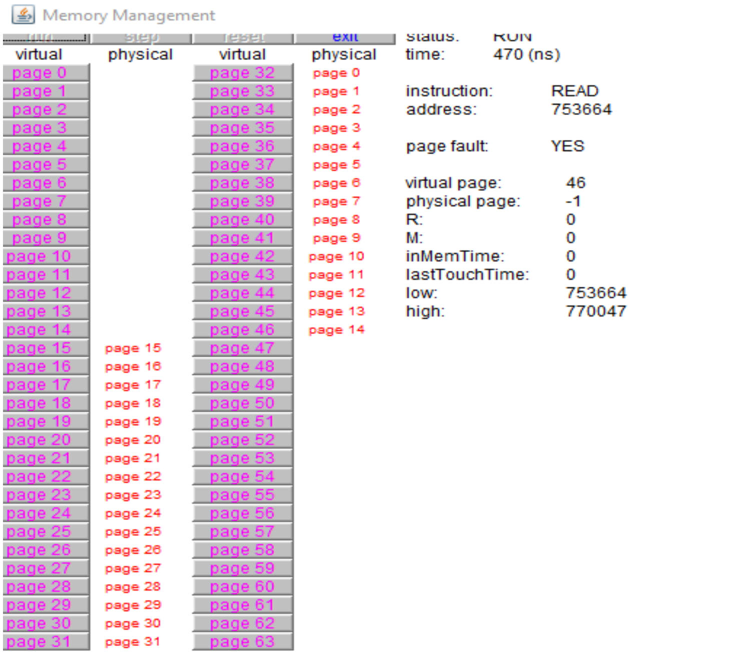
More algorithms:

Algorithm	Comment
Optimal	Not implementable, but useful as a benchmark
NRU (Not Recently Used)	Very crude approximation of LRU
FIFO (First-In, First-Out)	Might throw out important pages
Second chance	Big improvement over FIFO
Clock	Realistic
LRU (Least Recently Used)	Excellent, but difficult to implement exactly
NFU (Not Frequently Used)	Fairly crude approximation to LRU
Aging	Efficient algorithm that approximates LRU well
Working set	Somewhat expensive to implement
WSClock	Good efficient algorithm

In documentation, we can find below information:

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 candidate 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.

Example of FIFO replacement algorithm:



Can you predict which virtual memory addresses cause page faults?

Page Fault is raise, when memory page is mapped into a virtual address, but is not loaded to physical memory. The page fault should appear on 32^{nd} page, because that for virtual pages (32 – 64), there are no physical pages available, so we need to replace and page fault is raised 32 times

Configuration:

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.

Therefore, we maps our physical pages to virtual pages in this way:

Memory.conf

```
// 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 16384

// 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
```

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
```

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 -  524288 / 16384 = 32
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
```