Task description.

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? Locate in the sources and describe to the instructor the page replacement algorithm.
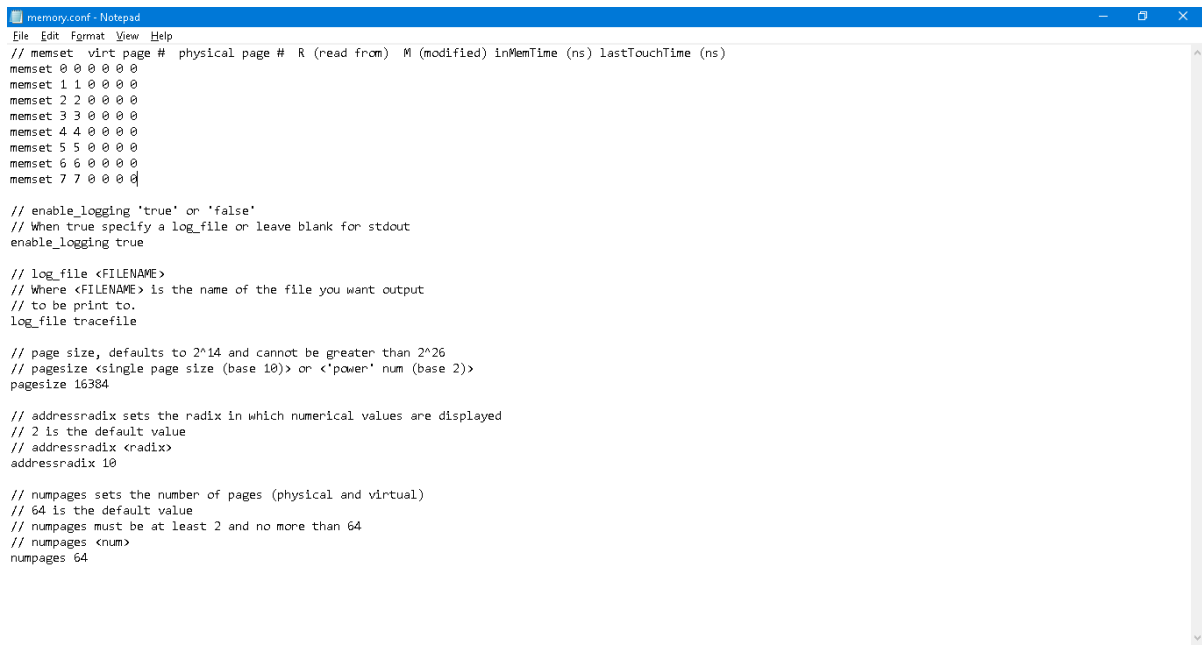
As a background, I can say, the computer memories are created registers, where each register composes cell which able to store one-bit data, address lines used for data routing which means data buses. The matter of this is to write to the memory or read from it is handled by read and write connection lines. Those register make a one-dimensional array which is used for storing data and processes. The easiest way to write them into the memory would be to organize them in a contiguous manner so right after the last bit of one program we write the 1st bit of another one. However, the processes come and go, hence when we free a memory space is left. Fortunately, the device called Memory Management Unit keeps track of it and allocates the free memory space. If we want to access to the memory which has no physical address mapped, then a page fault warning is returned. But the processes differ in sizes. This issue is partially handled by participating the memory into segments called pages. And in our task, we are dealing with the static partitioning, where the size of each partition is pre-set. The "page size" variable in the memory configuration file of our program has been set to the default – 16384 and the number of pages is set to – 64:

```
// 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 16

// 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
```

The next step was to setup the memory.conf. file to specify the initial content of the virtual memory map, specifically to map any 8 pages of physical memory to the 1st 8 pages of virtual memory.

```
memory.conf - Notepad
File  Edit  Format  View  Help
// 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
```

memset – performs mapping between virtual and physical page;

enable_logging – True or False turn on/off logs;

log_file – path to log file and name;

pagesize – page size, by default 2^14 and can not be > than 2^26;

addressradix – sets the radix in which numerical values are displaced.

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

Configuration of the Commands file

Here, was needed to setup the commands file to specify a sequence of memory instructions to be performed, for reading from one virtual memory address on each of the 64 virtual pages. As the page size attribute is set to 16384 in the configuration file as default, so I decided to keep in that way. Therefore, to read from 1 address on each of the 64 pages, I decided to read from every 1 address on each page. Starting from 0-64 I defined READ operations on each multiple of 16384 in the following way.

```
commands - Notepad                                               -    □    ×
File  Edit  Format  View  Help
// 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 884736
READ 901120
READ 917504
READ 933888
READ 950272
READ 966656
READ 983040
READ 999424
READ 1015808
```

Simulation the initial state

After running the simulator with CMD, I see the initial state of the memory pages mapping. The virtual pages 0-7 are mapped to the physical pages that we specified in the configuration file. The virtual pages 8-31 are mapped to physical pages with the same number. Pages 32-63 have no mapping.

What is interesting, at the 32 virtual page fault appeared, however it was easy to predict, because this virtual page has no mapping to a physical page. Page fault occurs in which there has been a reference to a page which does not have mapping to a physical page. After the failed attempt to read virtual page 32, the simulator maps physical page 1 to it:

After this simulation, we understood that each of the virtual pages 32-63 has been mapped to a physical page in the same order as pages 0-31. So, this indicate that First In – First Out (FIFO) algorithm was used.

FIFO algorithm is not adequate, because it does not make a difference among pages. Much better to use Least Recently Used (LRU) Page Replacement algorithm. In LRU algorithm is a Greedy algorithm, where the page to be replaced is the least recently used. The idea is based on locality of reference, the least recently used page is not likely.

Tracefile

The "tracefile" file contains a log of the operators since the simulation started together with the statuses. Just as a confirmation of the simulation process we can see that the 1st 32 READ operations were successful and all the remaining operations statuses indicate page fault:

```
tracefile - Notepad
File  Edit  Format  View  Help
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
```