Belady´s Anomaly Simulator

Purpose:

Use the moss memory simulator http://www.ontko.com/moss/#memory to show how Belady´s anomaly is produced based on the examples provided in Andrew S. Tanenbaum ´s book “Modern Operating Systems”.

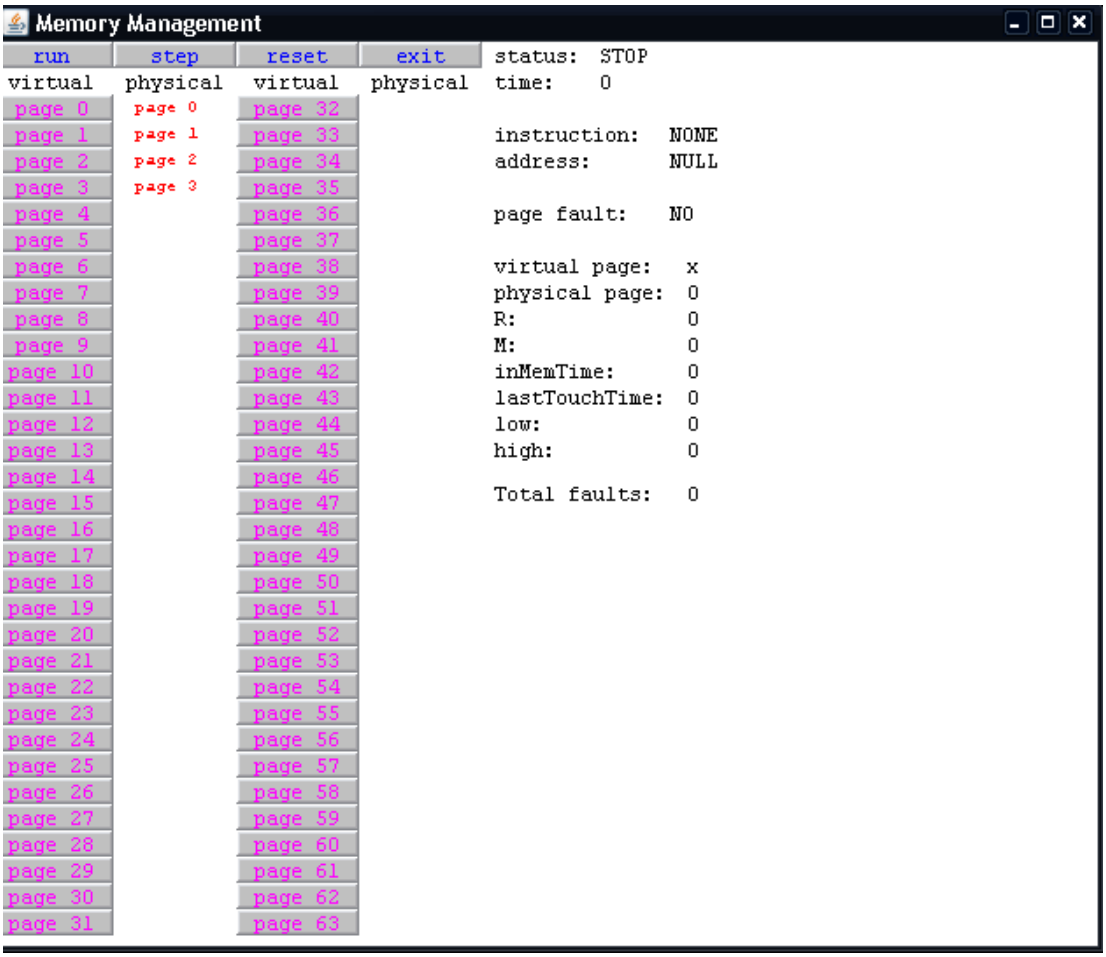
Running the simulator:

The program reads a command file, optionally reads a configuration file, displays a GUI window which allows you to execute the command file, and optionally writes a trace file.

To run the program, enter the following command line.

$ java MemoryManagement commands memory.conf

The program will display a window allowing you to run the simulator. You will notice a row of command buttons across the top, two columns of "page" buttons at the left, and an informational display at the right.



Typically you will use the step button to execute a command from the input file, examine information about any pages by clicking on a page button, and when you're done, quit the simulation using the exit button.

The buttons:

Button	Description
run	runs the simulation to completion. Note that the simulation pauses and updates the screen between each step.
step	runs a single setup of the simulation and updates the display.
reset	initializes the simulator and starts from the beginning of the command file.
exit	exits the simulation.
page *n*	display information about this virtual page in the display area at the right.

The informational display:

Field	Description
status:	RUN, STEP, or STOP. This indicates whether the current run or step is completed.
time:	number of "ns" since the start of the simulation.
instruction:	READ or WRITE. The operation last performed.
address:	the virtual memory address of the operation last performed.
page fault:	whether the last operation caused a page fault to occur.
virtual page:	the number of the virtual page being displayed in the fields below. This is the last virtual page accessed by the simulator, or the last page *n* button pressed.
physical page:	the physical page for this virtual page, if any. -1 indicates that no physical page is associated with this virtual page.
R:	whether this page has been read. (1=yes, 0=no)
M:	whether this page has been modified. (1=yes, 0=no)
inMemTime:	number of ns ago the physical page was allocated to this virtual page.
lastTouchTime:	number of ns ago the physical page was last modified.
low:	low virtual memory address of the virtual page.
high:	high virtual memory address of the virtual page.
Total faults:	The total amount of page faults

The Command File

The command file for the simulator specifies a sequence of memory instructions to be performed. Each instruction is either a memory READ or WRITE operation, and includes a

virtual memory address to be read or written. Depending on whether the virtual page for the address is present in physical memory, the operation will succeed, or, if not, a page fault will occur.

Operations on Virtual Memory

There are two operations one can carry out on pages in memory: READ and WRITE.

The format for each command is

operation address

or
operation random
where *operation* is READ or WRITE, and *address* is the numeric virtual memory address, optionally preceeded by one of the radix keywords bin, oct, or hex. If no radix is supplied, the number is assumed to be decimal. The keyword random will generate a random virtual memory address (for those who want to experiment quickly) rather than having to type an address.

The Configuration File

The configuration file memory.conf is used to specify the the initial content of the virtual memory map (which pages of virtual memory are mapped to which pages in physical memory) and provide other configuration information, such as whether operation should be logged to a file.

In the memory.conf we define the number of pages frames to be used. For this problem (Belady´s Anomaly), we use four pages frames, as showed in fig. 1.0, this give us a total of 10 pages faults. To see Belady´s Anomaly you need to change the number of page frames to three, then you should look that the total page fault is 9, which is lower than if we use four page frames. This can be made by changing to -1 the third column.

Sample Configuration File (four page frames)

// 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 -1 0 0 0 0
memset 5 -1 0 0 0 0
memset 6 -1 0 0 0 0
memset 7 -1 0 0 0 0
memset 8 -1 0 0 0 0

```
memset 9 -1 0 0 0 0
memset 10 -1 0 0 0 0
memset 11 -1 0 0 0 0
memset 12 -1 0 0 0 0
memset 13 -1 0 0 0 0
memset 14 -1 0 0 0 0
memset 15 -1 0 0 0 0
memset 16 -1 0 0 0 0
memset 17 -1 0 0 0 0
memset 18 -1 0 0 0 0
memset 19 -1 0 0 0 0
memset 20 -1 0 0 0 0
memset 21 -1 0 0 0 0
memset 22 -1 0 0 0 0
memset 23 -1 0 0 0 0
memset 24 -1 0 0 0 0
memset 25 -1 0 0 0 0
memset 26 -1 0 0 0 0
memset 27 -1 0 0 0 0
 memset 28 -1 0 0 0 0
memset 29 -1 0 0 0 0
memset 30 -1 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
```

The Output File

The output file contains a log of the operations since the simulation started (or since the last reset). It lists the command that was attempted and what happened as a result. You can review this file after executing the simulation.

The output file contains one line per operation executed. The format of each line is:

command address ... status

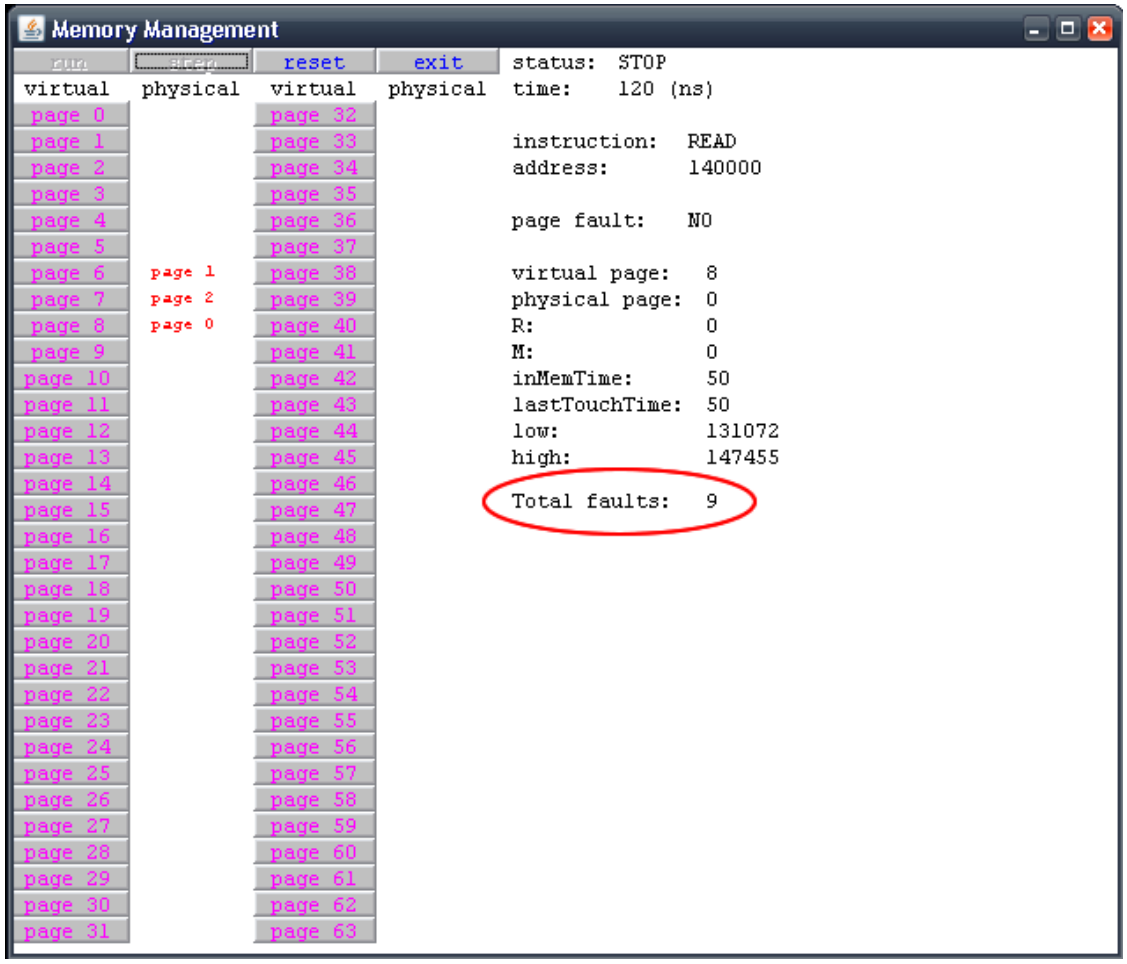
where:

- *command* is READ or WRITE,
- *address* is a number corresponding to a virtual memory address, and
- *status* is okay or page fault.

Sample Output (three page frames

The output "tracefile" looks something like this:

READ 70000 ... page fault
READ 90000 ... page fault
READ 100000 ... page fault
READ 120000 ... page fault
READ 70000 ... page fault
READ 90000 ... page fault
READ 140000 ... page fault
READ 70000 ... okay
READ 90000 ... okay
READ 100000 ... page fault
READ 120000 ... page fault
READ 140000 ... okay


Memory Management
run
step
reset
exit
status: STOP
virtual
physical
virtual
physical
time: 120 (ns)
page 0
page 1
page 2
page 3
page 4
page 5
page 6
page 7
page 8
page 9
page 10
page 11
page 12
page 13
page 14
page 15
page 16
page 17
page 18
page 19
page 20
page 21
page 22
page 23
page 24
page 25
page 26
page 27
page 28
page 29
page 30
page 31
page 32
page 33
page 34
page 35
page 36
page 37
page 38
page 39
page 40
page 41
page 42
page 43
page 44
page 45
page 46
page 47
page 48
page 49
page 50
page 51
page 52
page 53
page 54
page 55
page 56
page 57
page 58
page 59
page 60
page 61
page 62
page 63
page 1
page 2
page 0
instruction: READ
address: 140000
page fault: NO
virtual page: 8
physical page: 0
R: 0
M: 0
inMemTime: 50
lastTouchTime: 50
low: 131072
high: 147455
Total faults: 9