[EOPSY] Laboratory task 4 : Memory Management

- Rajagopal Thirugnanasambandam(309263)

===

Instruction :

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.

===

Introduction :

Memory Management Unit:

Memory management is a feature of an operating system that controls or manages primary memory and transfers processes between main memory and disc during execution. Memory management keeps track of all memory locations, whether they are allocated to a process or are free. It determines how much memory will be allotted to processes. It determines which processes will have access to memory at what moment. It keeps track of when memory is released or unallocated and changes the state accordingly. In most cases, a process is saved in secondary memory. While a process is running, it is transferred from secondary memory to physical memory (main memory). It must be allocated to a memory address when travelling. There may be non-contiguous vacant spots in this entire section of accessible memory. Instead of allocating only contiguous memory, we can store these processes in available non-contiguous memory.

As a result, we may divide the entire process into pages and store them in these vacant regions in physical memory known as frames. The Memory Management Unit manages this memory assignment/operation, which is known as paging (MMU). Each process in secondary memory is separated into pages in paging, and physical memory is likewise divided into frames. Each frame contains a copy of each page. It can be kept in many locations, but it is preferable to keep them in a continuous manner. Each frame must have the same size. Because we are mapping the pages to frames, the page size should be the same as the frame size.

A page fault happens when a page in secondary memory is requested that is not found in physical memory. Page faults arise when physical memory is less than secondary memory. Page replacement is used to correct certain page defects. It is necessary to determine the page number that must be changed. Page replacement algorithms are classified into several categories. The page replacement algorithm used to replace the long time unused page in future.

The first page given to the first frame is replaced via the First In First Out page replacement procedure. After the initial frame, it replaces succeeding pages in a queue with later allocated frames.

==

Commands:

// 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 hex 0

READ hex 4000

READ hex 8000

READ hex c000

READ hex 10000

READ hex 14000

READ hex 18000

READ hex 1c000

READ hex 20000

READ hex 24000

READ hex 28000

READ hex 2c000

READ hex 30000

READ hex 34000

READ hex 38000

READ hex 3C000

READ hex 40000

READ hex 44000

READ hex 48000

READ hex 4C000

READ hex 50000

READ hex 54000

READ hex 58000

READ hex 5C000

READ hex 60000

READ hex 64000

READ hex 68000

READ hex 6C000

READ hex 70000

READ hex 74000

READ hex 78000

READ hex 7C000

READ hex 80000

READ hex 84000

READ hex 88000

READ hex 8C000

READ hex 90000

READ hex 94000

READ hex 98000

READ hex 9C000

READ hex A0000

READ hex A4000

READ hex A8000

READ hex AC000

READ hex B0000

READ hex B4000

READ hex B8000

READ hex BC000

READ hex C0000

READ hex C4000

READ hex C8000

READ hex CC000

READ hex D0000

READ hex D4000

READ hex D8000

READ hex DC000

READ hex E0000

READ hex E4000

READ hex E8000

READ hex EC000

READ hex F0000

READ hex F4000

READ hex F8000

READ hex FC000

==

Memory Configuration:

// memset virt page # physical page # R (read from) M (modified) inMemTime (ns) lastTouchTime (ns)

memset 0 6 0 0 0 0

memset 1 7 0 0 0 0

memset 2 5 0 0 0 0

memset 3 2 0 0 0 0

memset 4 0 0 0 0 0

memset 5 4 0 0 0 0

memset 6 1 0 0 0 0

memset 7 3 0 0 0 0

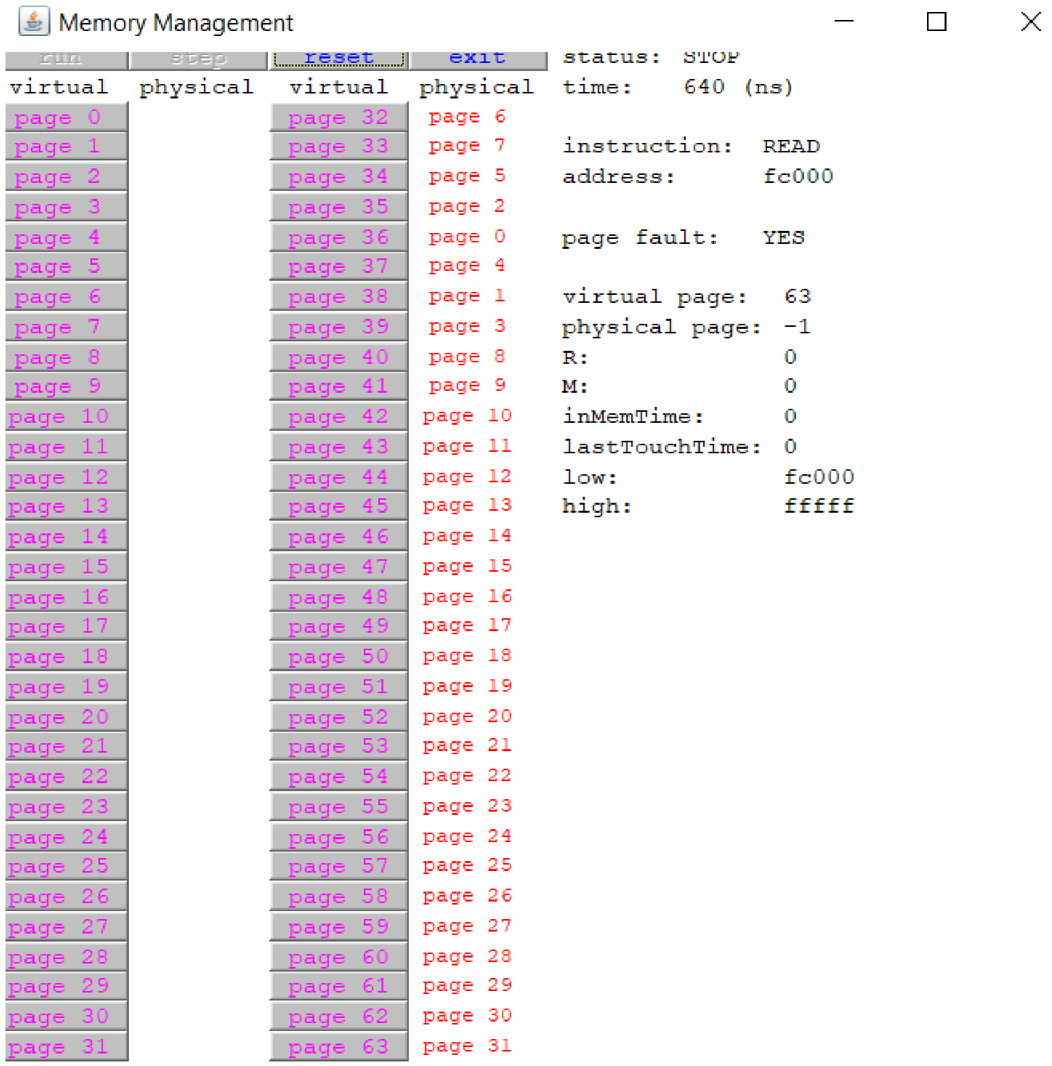
==

Result:

READ 0 ... okay
READ 4000 ... okay
READ 8000 ... okay
READ c000 ... okay
READ 10000 ... okay
READ 14000 ... okay
READ 18000 ... okay
READ 1c000 ... okay
READ 20000 ... okay
READ 24000 ... okay
READ 28000 ... okay
READ 2c000 ... okay
READ 30000 ... okay
READ 34000 ... okay
READ 38000 ... okay
READ 3c000 ... okay
READ 40000 ... okay
READ 44000 ... okay
READ 48000 ... okay
READ 4c000 ... okay
READ 50000 ... okay
READ 54000 ... okay
READ 58000 ... okay
READ 5c000 ... okay
READ 60000 ... okay
READ 64000 ... okay
READ 68000 ... okay
READ 6c000 ... okay
READ 70000 ... okay
READ 74000 ... okay
READ 78000 ... okay
READ 7c000 ... okay
READ 80000 ... page fault
READ 84000 ... page fault
READ 88000 ... page fault
READ 8c000 ... page fault
READ 90000 ... page fault
READ 94000 ... page fault

READ 98000 ... page fault
READ 9c000 ... page fault
READ a0000 ... page fault
READ a4000 ... page fault
READ a8000 ... page fault
READ ac000 ... page fault
READ b0000 ... page fault
READ b4000 ... page fault
READ b8000 ... page fault
READ bc000 ... page fault
READ c0000 ... page fault
READ c4000 ... page fault
READ c8000 ... page fault
READ cc000 ... page fault
READ d0000 ... page fault
READ d4000 ... page fault
READ d8000 ... page fault
READ dc000 ... page fault
READ e0000 ... page fault
READ e4000 ... page fault
READ e8000 ... page fault
READ ec000 ... page fault
READ f0000 ... page fault
READ f4000 ... page fault
READ f8000 ... page fault
READ fc000 ... page fault


Memory Management
run
step
reset
exit
status: STOP
virtual
physical
virtual
physical
time: 0
instruction: NONE
address: NULL
page fault: NO
virtual page: 0
physical page: 6
R: 0
M: 0
inMemTime: 0
lastTouchTime: 0
low: 0
high: 3fff
page 0 page 4 page 32
page 1 page 6 page 33
page 2 page 3 page 34
page 3 page 7 page 35
page 4 page 5 page 36
page 5 page 2 page 37
page 6 page 0 page 38
page 7 page 1 page 39
page 8 page 8 page 40
page 9 page 9 page 41
page 10 page 10 page 42
page 11 page 11 page 43
page 12 page 12 page 44
page 13 page 13 page 45
page 14 page 14 page 46
page 15 page 15 page 47
page 16 page 16 page 48
page 17 page 17 page 49
page 18 page 18 page 50
page 19 page 19 page 51
page 20 page 20 page 52
page 21 page 21 page 53
page 22 page 22 page 54
page 23 page 23 page 55
page 24 page 24 page 56
page 25 page 25 page 57
page 26 page 26 page 58
page 27 page 27 page 59
page 28 page 28 page 60
page 29 page 29 page 61
page 30 page 30 page 62
page 31 page 31 page 63



==

Result:

The commands are written in the format described above. The simulator reads each of the 64 simulated pages. The first eight pages of virtual memory are assigned to eight separate physical memory pages. The page fault occurred only when the software attempted to access a virtual memory block of a virtual page that was not mapped or saved in physical memory. None of the virtual pages from virtual page 32 are mapped to any physical frames. As a result, whenever the application reads from virtual memory 32, it should transfer it to any frame in physical memory. However, we can see that all of the available physical frames are taken. As a result, the page replacement mechanism is utilized to deal with this problem. The First In, First Out (FIFO) method is employed in this simulation.

The virtual page 32 is mapped to physical frame 7, which is the first used/mapped physical frame. As a result, that frame no longer corresponds to virtual page 0. It has been replaced with virtual page 32. The same holds true for every future virtual pages starting with 32.

==
==