EOPSY Lab-4

Name: MADHANA GOPALAPURAM RAMESH RATHISH AADIRAJA
ID: 302457
Group: G

Introduction:

A process/program is generally stored in the secondary memory. When a process has to be executed it is moved from the secondary memory to physical memory(main memory). While moving, it has to be assigned to some address in the memory. In this whole part of available memory, there can be non-contiguous free spaces. Instead of allotting only contiguous memory we can use these available non-contiguous memory to store these processes. So, we can partition the whole process into pages and store them in these free spaces in physical memory called frames. Managing of this memory assignment/operation is called as paging and it is implemented by the Memory Management Unit(MMU).

In paging, each process in secondary memory is divided in the form of pages and the physical memory is also divided into frames. Each page is stored in each frame. It can be stored in different locations, but the preferred way is storing them in a contiguous way. The size of each frame must be equal. Since we map the pages to frames the page size should be the same of frame size.

Logical address is the address generated by CPU for every page and physical address is the address available on memory unit(frame). The main purpose of MMU is to convert logical address into physical address. The logical address is divided into page number and page offset. The physical address is divided into frame number and frame offset. The MMU converts page number to frame number. The offset remains same in both. To do this MMU uses a special kind of mapping which is done by page table. This page table matches the page number to the frame number in the memory.

While requesting a page in secondary memory not found in the physical memory, page fault occurs. Since physical memory is smaller than secondary memory these page faults occurs. So to fix these page faults, page replacement is done. It is about determining the page number which needs to be replaced. There are different types of page replacement algorithms.

The optimal page replacement algorithm replaces the page which will not be used for a long period of time in the future.

The least recent used page replacement algorithm replaces the page which has not been referred for a long period of time in the past.

The First In First Out page replacement algorithm replaces the first page which is assigned to the first frame. It replaces subsequent pages in a queue with subsequently assigned frames after the first frame.

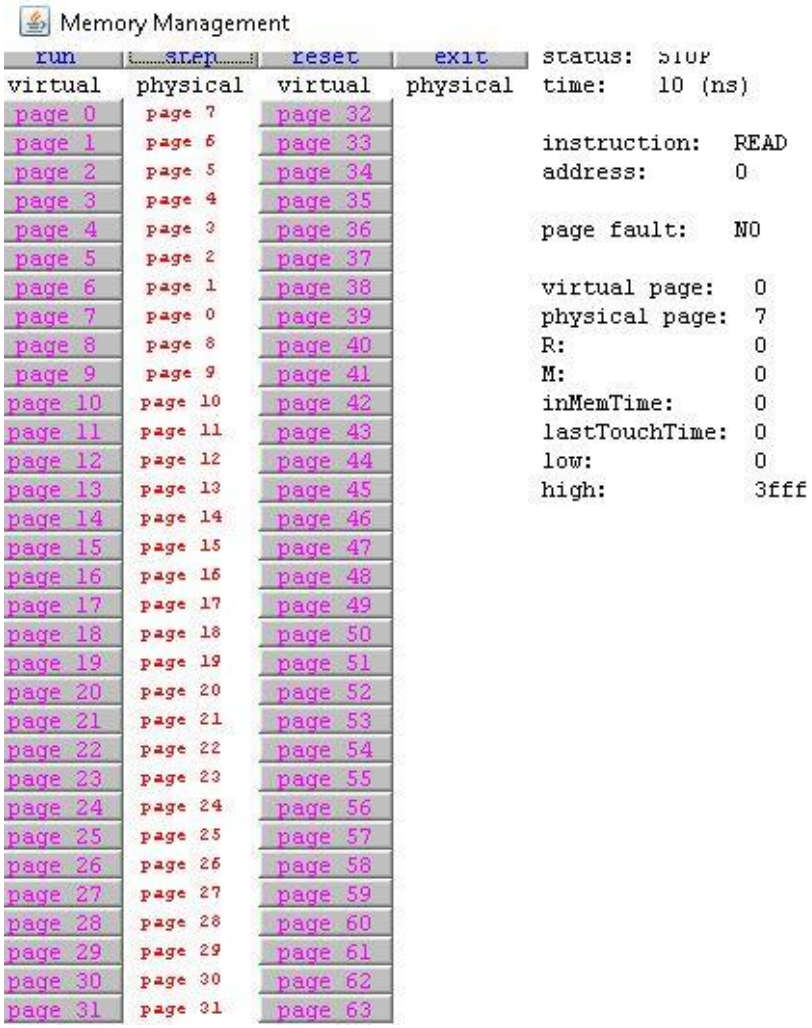
Commands:

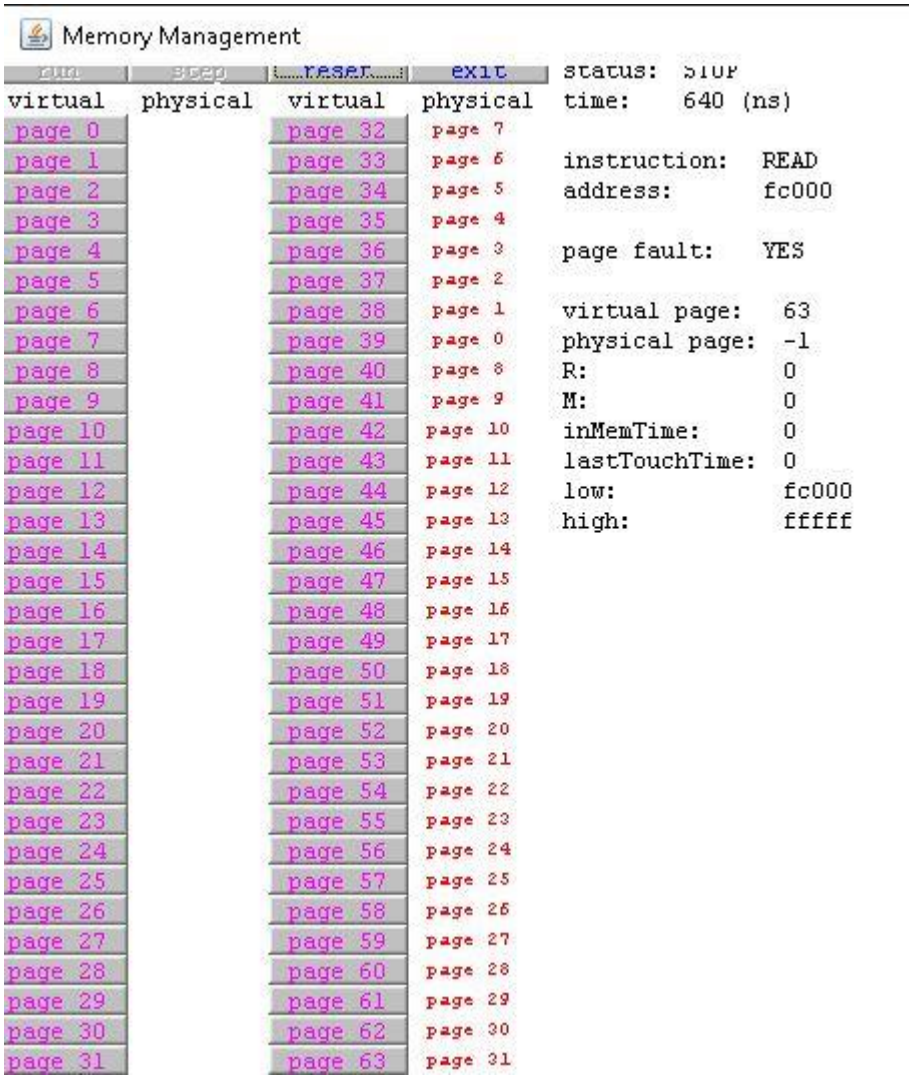
READ hex 0	READ hex 80000
READ hex 4000	READ hex 84000
READ hex 8000	READ hex 88000
READ hex C000	READ hex 8C000
READ hex 10000	READ hex 90000
READ hex 14000	READ hex 94000
READ hex 18000	READ hex 98000
READ hex 1C000	READ hex 9C000
READ hex 20000	READ hex A0000
READ hex 24000	READ hex A4000
READ hex 28000	READ hex A8000
READ hex 2C000	READ hex AC000
READ hex 30000	READ hex B0000
READ hex 34000	READ hex B4000
READ hex 38000	READ hex B8000
READ hex 3C000	READ hex BC000
READ hex 40000	READ hex C0000
READ hex 44000	READ hex C4000
READ hex 48000	READ hex C8000
READ hex 4C000	READ hex CC000
READ hex 50000	READ hex D0000
READ hex 54000	READ hex D4000
READ hex 58000	READ hex D8000
READ hex 5C000	READ hex DC000
READ hex 60000	READ hex E0000
READ hex 64000	READ hex E4000
READ hex 68000	READ hex E8000
READ hex 6C000	READ hex EC000
READ hex 70000	READ hex F0000
READ hex 74000	READ hex F4000
READ hex 78000	READ hex F8000
READ hex 7C000	READ hex FC000

// memset virt page # physical page # R (read from) M (modified) inMemTime (ns) lastTouchTime (ns)
memset 0 7 0 0 0 0
memset 1 6 0 0 0 0
memset 2 5 0 0 0 0
memset 3 4 0 0 0 0
memset 4 3 0 0 0 0
memset 5 2 0 0 0 0
memset 6 1 0 0 0 0
memset 7 0 0 0 0 0

Results:

READ 0 ... okay	READ 80000 ... page fault
READ 4000 ... okay	READ 84000 ... page fault
READ 8000 ... okay	READ 88000 ... page fault
READ c000 ... okay	READ 8c000 ... page fault
READ 10000 ... okay	READ 90000 ... page fault
READ 14000 ... okay	READ 94000 ... page fault
READ 18000 ... okay	READ 98000 ... page fault
READ 1c000 ... okay	READ 9c000 ... page fault
READ 20000 ... okay	READ a0000 ... page fault
READ 24000 ... okay	READ a4000 ... page fault
READ 28000 ... okay	READ a8000 ... page fault
READ 2c000 ... okay	READ ac000 ... page fault
READ 30000 ... okay	READ b0000 ... page fault
READ 34000 ... okay	READ b4000 ... page fault
READ 38000 ... okay	READ b8000 ... page fault
READ 3c000 ... okay	READ bc000 ... page fault
READ 40000 ... okay	READ c0000 ... page fault
READ 44000 ... okay	READ c4000 ... page fault
READ 48000 ... okay	READ c8000 ... page fault
READ 4c000 ... okay	READ cc000 ... page fault
READ 50000 ... okay	READ d0000 ... page fault
READ 54000 ... okay	READ d4000 ... page fault
READ 58000 ... okay	READ d8000 ... page fault
READ 5c000 ... okay	READ dc000 ... page fault
READ 60000 ... okay	READ e0000 ... page fault
READ 64000 ... okay	READ e4000 ... page fault
READ 68000 ... okay	READ e8000 ... page fault
READ 6c000 ... okay	READ ec000 ... page fault
READ 70000 ... okay	READ f0000 ... page fault
READ 74000 ... okay	READ f4000 ... page fault
READ 78000 ... okay	READ f8000 ... page fault
READ 7c000 ... okay	READ fc000 ... page fault


Memory Management
run
step
reset
exit
status: stop
virtual
physical
virtual
physical
time: 10 (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 7
page 6
page 5
page 4
page 3
page 2
page 1
page 0
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
instruction: READ
address: 0
page fault: NO
virtual page: 0
physical page: 7
R: 0
M: 0
inMemTime: 0
lastTouchTime: 0
low: 0
high: 3fff


Memory Management
run
step
reset
exit
status: stop
virtual
physical
virtual
physical
time: 640 (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 7
page 6
page 5
page 4
page 3
page 2
page 1
page 0
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
instruction: READ
address: fc000
page fault: YES
virtual page: 63
physical page: -1
R: 0
M: 0
inMemTime: 0
lastTouchTime: 0
low: fc000
high: fffff

Observations:

The commands are written as shown above. Each of the 64 virtual pages are read by the simulator. The first eight pages of virtual memory is mapped to eight different pages of physical memory.

As we can see from above images when it reads from each of the virtual page from 0-31, it is mapped to the physical frame 0-31 and no page fault occurs. The page fault occurred only when the program attempts to access a block of virtual memory of a virtual page that is not mapped or stored in the physical memory. From the virtual page 32 none of the virtual pages are mapped to any physical frames. So when the program reads from the virtual memory 32 it should map it to any frame in the physical memory. But we can see that all of the available physical frames are occupied. So to handle this situation, page replacement algorithm is used. In this simulation First In First Out algorithm is used. We can observe that the virtual page 32 is mapped to physical frame 7, which is the first used/mapped physical frame. So that frame no longer matches to the virtual page 0. It is replaced by the virtual page 32. The same happens to all subsequent virtual pages from 32.

Also there cannot be two virtual pages mapped to same physical frame. When the second virtual page maps to the already mapped physical frame, the previously mapped virtual page is no longer mapped to that physical frame. The latest one is mapped to that physical frame and the old virtual page is mapped to another available physical frame.