Topics: **Virtual Memory**
Demand Paging
Page Replacement algorithms

Virtual Memory

Two characteristics of paging and segmentation are the keys to a breakthrough in memory management:
- all memory references within a process are logical addresses that are dynamically translated into physical addresses at run time. A process may be swapped in and out of main memory such that occupies different regions of main memory at different times.

- a process may be broken up into a number of pieces (pages or segments) and these pieces need not to be contiguously located in main memory during the execution.

If these two characteristics are present, then it is not necessary that all the pages or all the segments of a process be in main memory during execution.
Virtual Memory is a technique that allows the execution of processes that may not be completely in memory.

Benefits:
- more processes may be maintained in main memory; this will bring an increase in CPU utilization and throughput.
- it is possible for a process to be larger than all the main memory. A program would no longer be constrained by the amount of physical memory that is available.
- less I/O would be needed to load or swap each user program into memory.

The 'Virtual memory' concept allows large virtual memory to be provided for programmers when only a smaller physical memory is available.

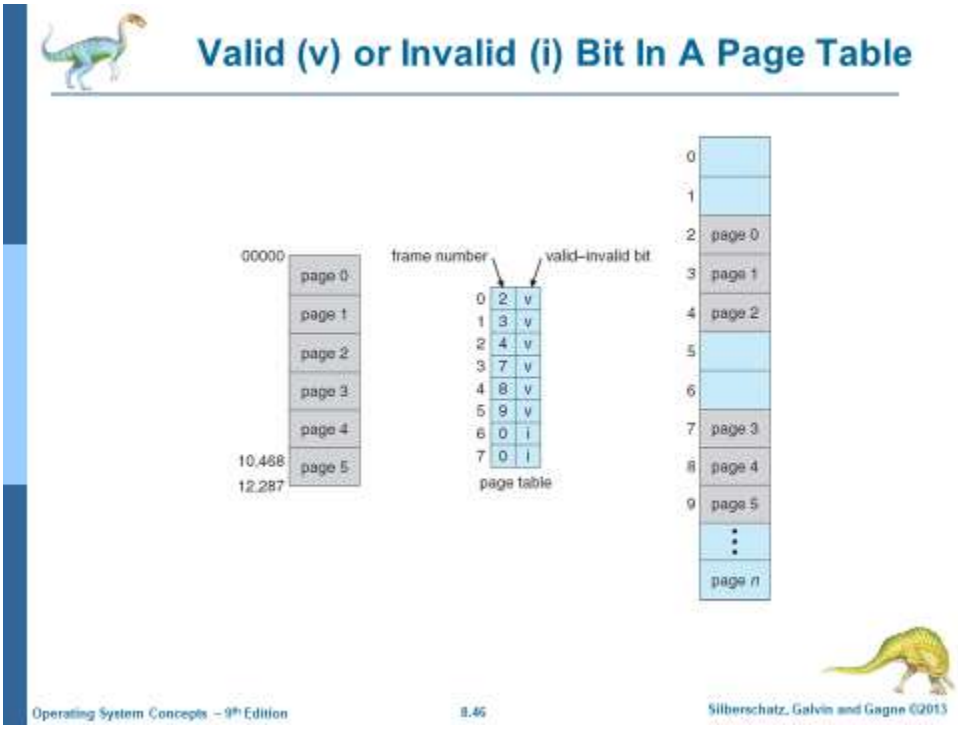
Virtual memory is implemented by:
- demand paging
- demand segmentation

Demand Paging

Demand Paging never swaps a page into memory unless that page will be needed (is demanded). Instead of swapping-in a whole process, the pager brings only those necessary pages into memory. If a page was brought in main memory, then the page is 'memory resident'.

valid - invalid bit:
'v' the associated page is both legal and in memory. (1)
'i' the page is either not valid or is valid but not in main memory. (0)

The page table entry might contain either the frame number, or the address of the page on the disk.



Access to a page marked invalid causes a 'page-fault' trap. Most of the time this trap is the result of the operating system's failure to bring the desired page into memory.

In case of a 'page fault' the O.S. reads the desired page into memory, into a free frame, and restarts the process from the interrupt point, as the page had always been in memory.

SERVING A PAGE FAULT – a specific case
(A SIMILAR PROBLEM WILL BE IN THE EXAM)

CPU generates logical address:

get to page table → check V/I bit → in this case is 0 → trap to the OS.
OS invoked, OS check cause of interrupt → page fault

OS MUST SERVE THE PAGE FAULT - must bring the demanded page in to the main memory.

1. check if there exist a free frame:
   A: YES
   B: NO

A: YES free frame

B: NO free frame
either there is no free frame in the entire memory OR they're free frame but the process is limited to a fixed number of frames and they are all taken.

If the process is limited to a fixed number of frames – use Local allocation
If the process can use other frames than its own frame – use Global allocation

Page Replacement
There exists the possibility that there are no 'free' frames in main memory.
The operating system has several actions at this point:
- terminate the process (not the best choice)
- swap out a process, freeing all its frames, and reducing the level of multiprogramming.
- page replacement: if no frame is free, we find one that is not currently being used and free it. The freed frame can now be used to hold the page for which the process faulted. All the tables has to be updated accordingly.

Allocation of Frames

Number of Frames: The minimum number of frames is decided by the instruction set architecture. There must be enough frames to hold all the different pages that any different instruction can reference. The worst-case occurs in the architecture that allows multiple levels of indirection. To overcome this case, a limit on the levels of indirection must be considered. In the worst case the entire virtual memory must be in physical memory.

The maximum number of frames is defined by the amount of available physical memory.

Allocation Algorithms

Global Versus Local Allocation
Global Replacement: allows a process to select a replacement frame from the set of all frames, even if that frame is currently allocated to some other process.
(the number of frame allocated to a specific process may change)
The process cannot control its own page-fault rate.

Local Replacement: each process selects from only its own set of allocated frames.
Local replacement might slow down a process by not making available to it other, less used pages of memory.

In general Global Replacement results in greater throughput and represents the common used method.

Further considerations:

modify (dirty) bit:
0 if the page has not been modified.
1 if the page has been modified. (whenever any byte or word in the page is written into)

A possibility for deciding what frame will be replaced:
When we select a page for replacement, we examine its 'modify bit'.
If the bit is set, then that page has been modified since it was read in from the disk. In this case, we must write back the page to the disk. If the copy of the page on the disk has not been overwritten, we can avoid writing the memory page to the disk.
This will shorten the page-fault service time.

REPLACEMENT ALGORITHMS

Optimal Algorithm
Replace the page that will not be used for the longest period of time.

FIFO Algorithm
When a page must be replaced, the oldest page in memory is chosen.

Least Recently Used (LRU)
Replace the page that has not been used for the longest period of time.
- the LRU algorithm relies on the existence of *locality* in the page frame stream. Since programs are usually written as loops, then it is likely that the loop body will fit within some small number of frames. If a page has been referenced recently, it is likely to be referenced again soon.
- the LRU does nearly as well as the optimal policy.
- the major problem is *how* to implement the LRU algorithm.

Enhanced Second-Chance Algorithm (used in Macintosh virtual-memory management scheme).
We consider both the reference bit and the modify bit. There are 4 classes:
(0,0) - neither recently used, nor modified
(0,1) - not recently used but modified
(1,0) - recently used but clean
(1,1) recently used and modified
Replace the page that is in the lowest class.