Lesson 23, 24, 25

Objectives

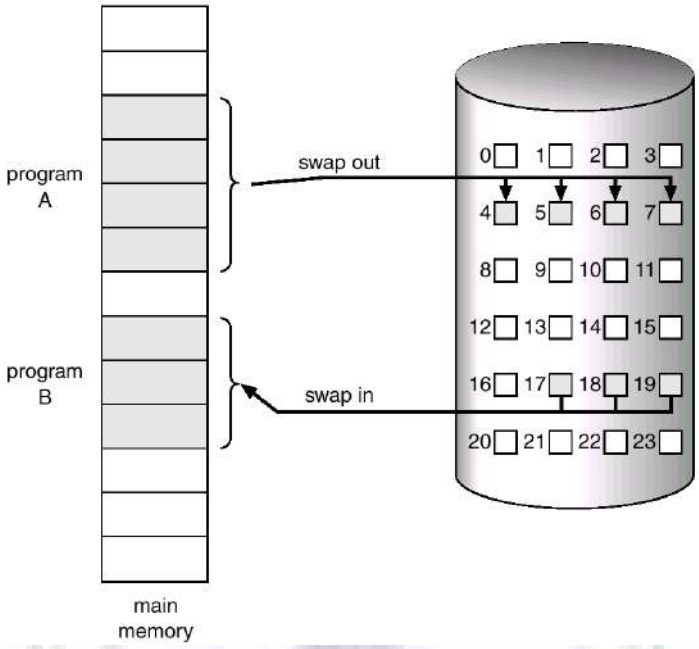
- Virtual Memory
- Demand paging
- Page fault
- Page replacement
- Thrashing

Virtual Memory

A portion of secondary storage that is used as an extension of physical memory called virtual memory. Separation of user logical memory from physical memory so that only active processes may be kept in main memory while rest may be moved to virtual memory. It is implemented via demand paging or demand segmentation or both. Its features include:

- Only part of the program needs to be in memory for execution.
- Logical address space can therefore be much larger than physical address space.
- Allows address spaces to be shared by several processes.
- Process creation becomes efficient.

It page that is no more needed may be swapped out to virtual memory from main memory and an active page may be swapped into main memory.

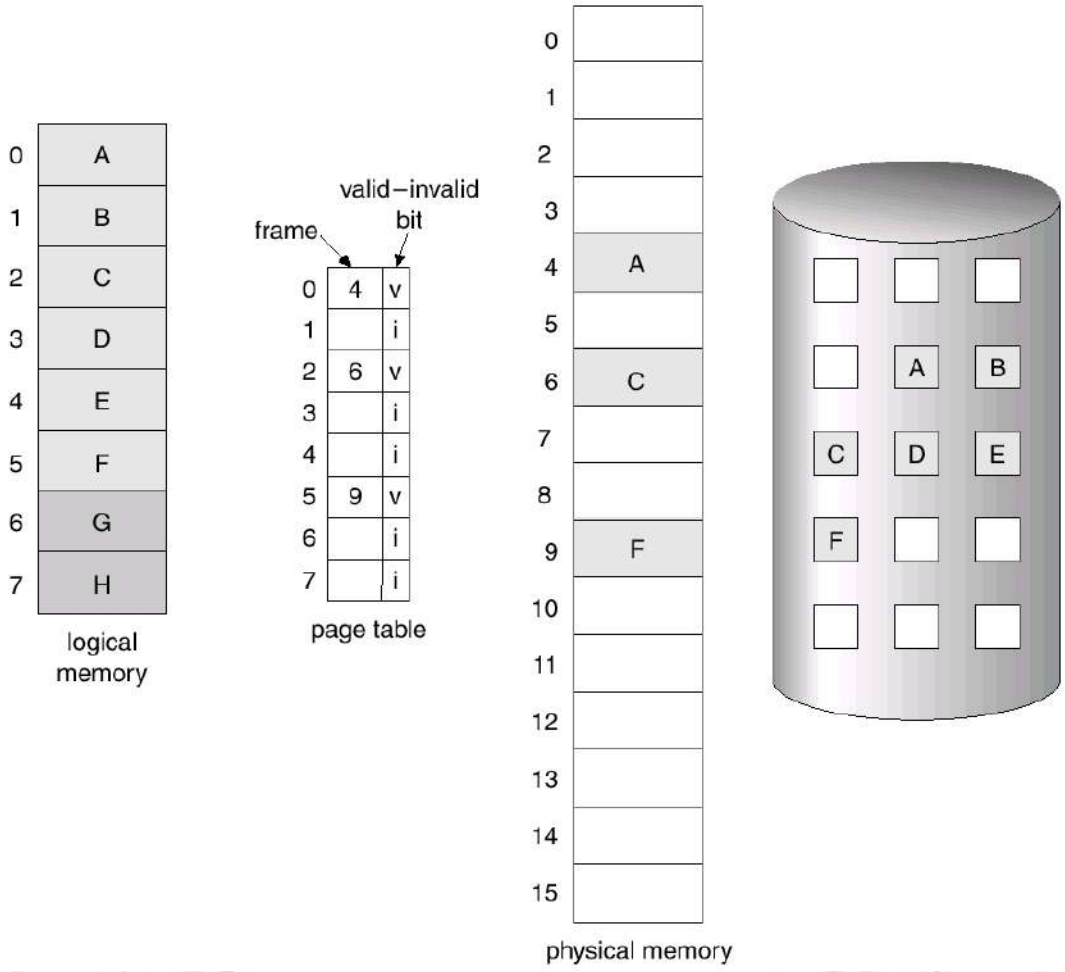


Demand Paging

Bringing the pages of a process from virtual memory to main memory on demand basis instead of bringing all the pages at once is called demand paging. Bringing a page into memory only when it is needed. If page is needed reference to, if it found use it, otherwise fetch it from virtual memory. It helps in:

- Less I/O needed
- Less memory needed
- Faster response
- More user programs

Page table contains the page references and additionally valid/invalid bit. The bit may be used for memory protection that if reference doesn't exist then it is set to 'i' otherwise it is set to 'v'. This is to avoid wrong pointers. This concept is shown in figure below.

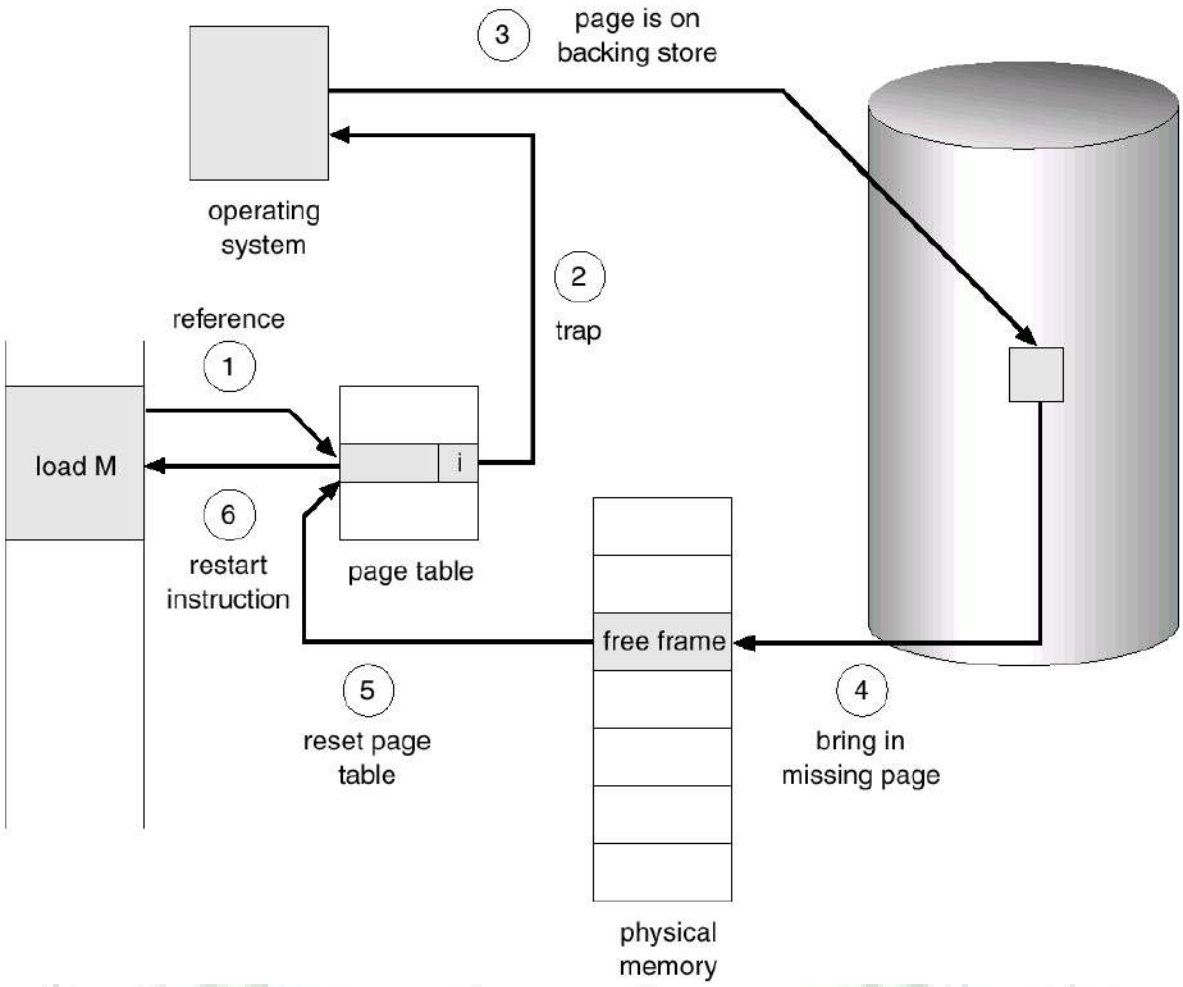


Page Fault

A situation when CPU demands a page and page is not currently in the main memory but at virtual memory. It is an interrupt and it is handled before process proceeds further. If page is found then it is called *page hit* if not found the called *page miss*. Page fault handling consists of following steps.

1. Report the trap
2. Save the state of current process
3. Find empty frame in main memory in process area if not found then replace the page
4. Fetch the page from virtual memory to main memory
5. Reset page table, also set valid bit to '1'
6. Restart the instruction (process)

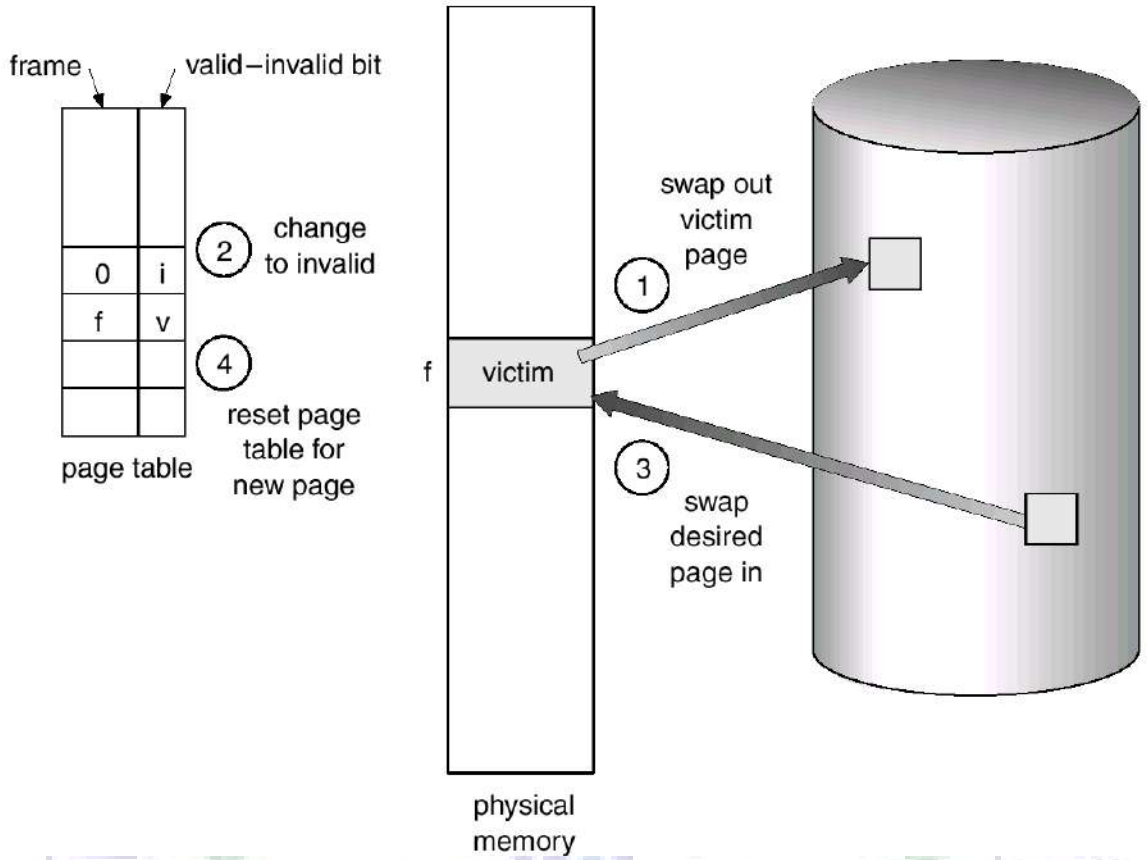
These steps are shown in figure below.



Page Replacement

What will happen if there is no frame free to bring the new page of a process that is just demanded?

Find the page that is currently may not be needed, by applying some algorithm (we shall study subsequently) and replace it with the new page and then restart the process. In this way same page may be brought in the memory again and again.



Now whether the demand paging is feasible, we can see it by it performance analysis.

Assume the page fault rate is *p*; where 0<*p*<*1*; if p=0 then there is no page fault and if p=1. Then the effective access time can be found as;

EAT= (1-*p*) x (memory access time) + *p* x (page fault overhead + swapping overhead + restart overhead)

Example;

Memory access time = 1 microsecond

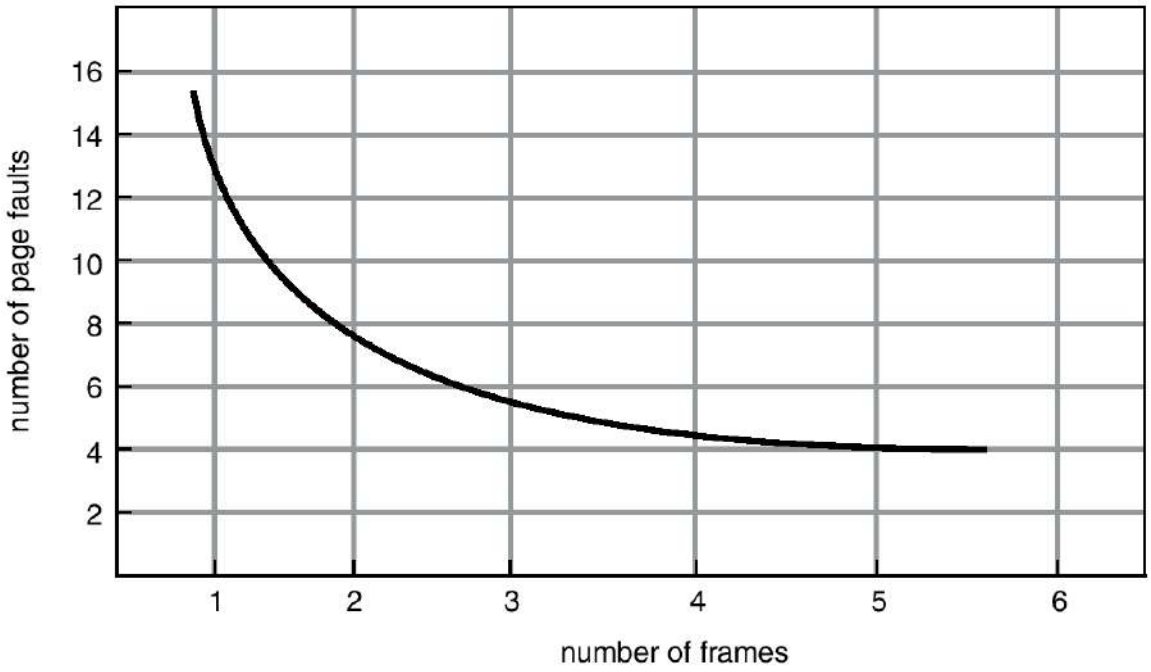
- 50% of the time the page that is being replaced has been modified and therefore needs to be swapped out.
- Swap Page Time = 10 msec = 10,000 msec
- EAT = (1 – p) x 1 + p (15000) = 1 + 15000P (in msec)

Page Replacement Algorithms

Criteria:

- Want lowest page-fault rate.

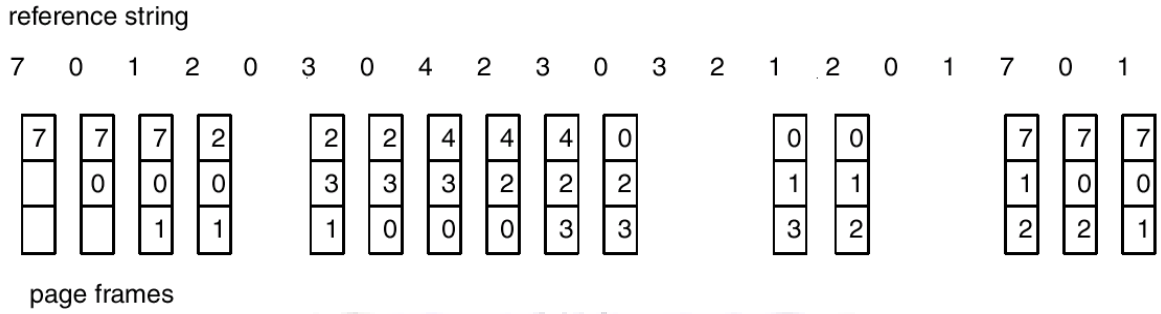
- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string.
- More number of frames cause less number of page faults



First in first out (FIFO)

Replace the page that is brought first in the memory.

Example-1



There are 15 page faults.

Example-2

Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

3 frames (3 pages can be in memory at a time per process)

1	1	4	5	
2	2	1	3	9 page faults
3	3	2	4	

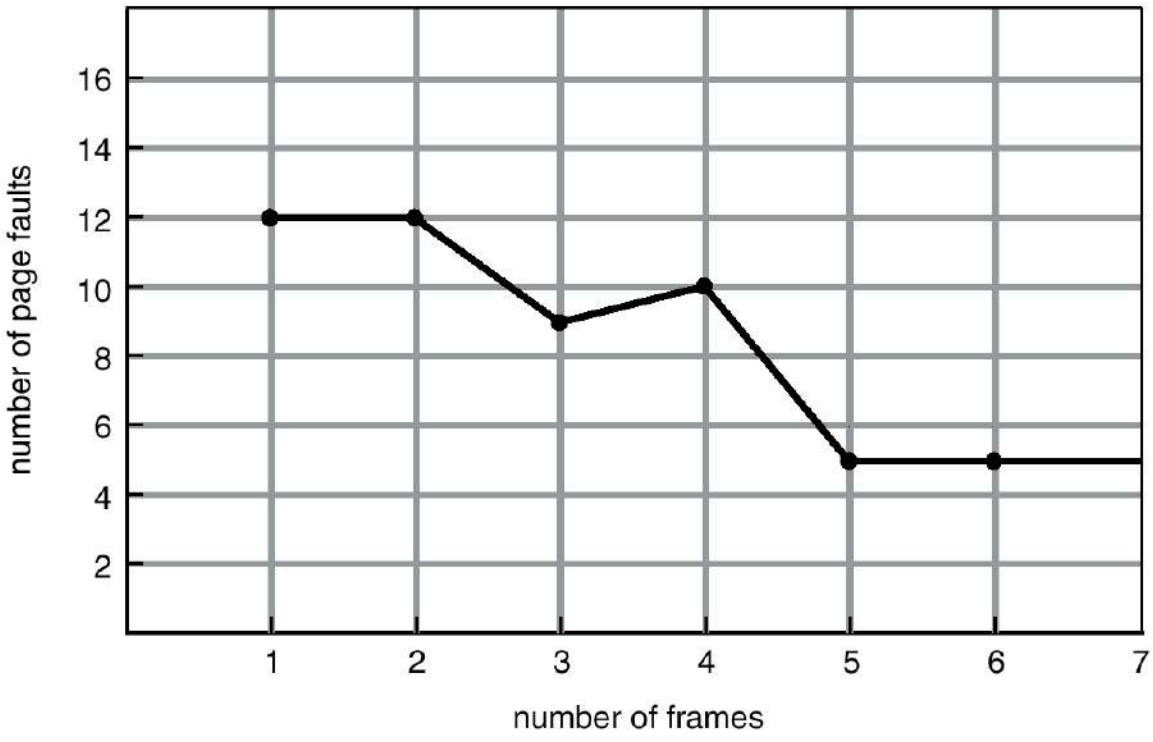
4 frames

1	1	5	4	
2	2	1	5	10 page faults
3	3	2		
4	4	3		

FIFO Replacement – Belady's Anomaly

- more frames ⇒ less page faults

Illustration of Belady's anomaly is given below.



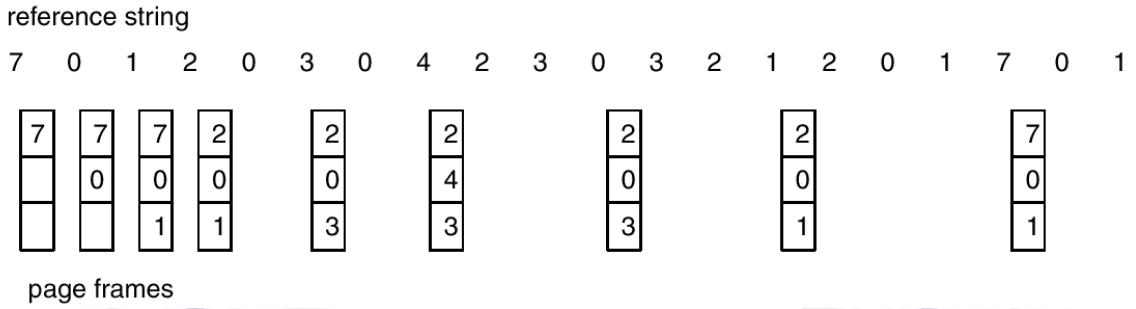
Global and Local Replacement

Global replacement – process selects a replacement frame from the set of all frames; one process can take a frame from another.

Local replacement – each process selects from only its own set of allocated frames.

Optimal Algorithm

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

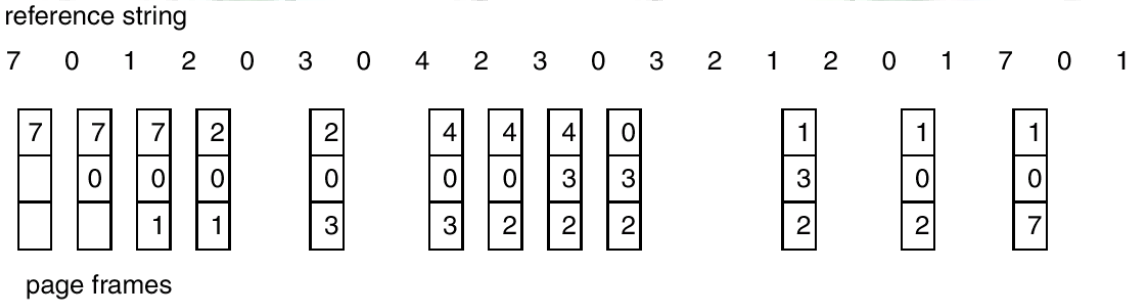


There are 9 page faults.

Least Recently Used (LRU) Algorithm

Counter implementation: Counter determines that how much a page is recent.

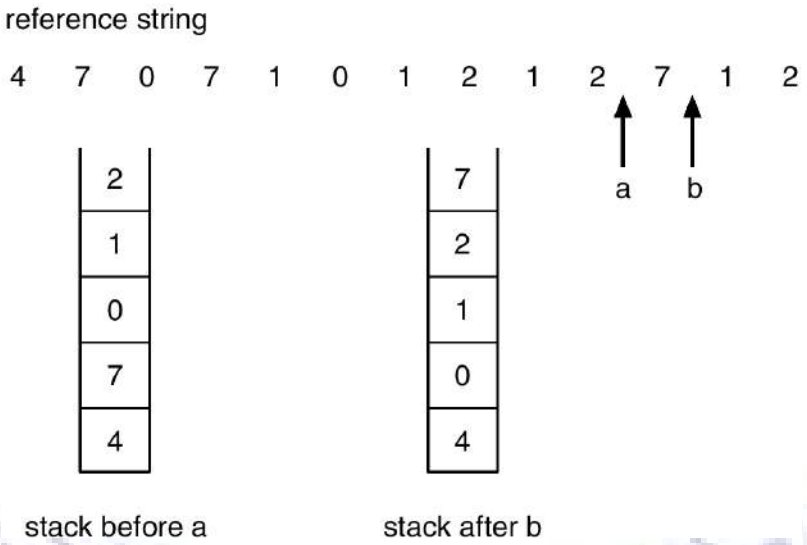
- Every page entry has a counter; every time page is referenced through this entry, copy the clock into the counter.
- When a page needs to be changed, look at the counters to determine which are to change.



It exhibits 12 number of page faults.

Stack implementation – keep a stack of page numbers in a double link form:

- Page referenced:
- Move it to the top
- Requires at the most 6 pointers to be changed (number of distinct pages)
- No search for replacement



Thrashing

If a process does not have "enough" pages, the page-fault rate is very high. This leads to:

- Low CPU utilization.
- Operating system thinks that it needs to increase the degree of multiprogramming.
- Another process added to the system.
- Thrashing ≡ a process is busy swapping pages in and out.