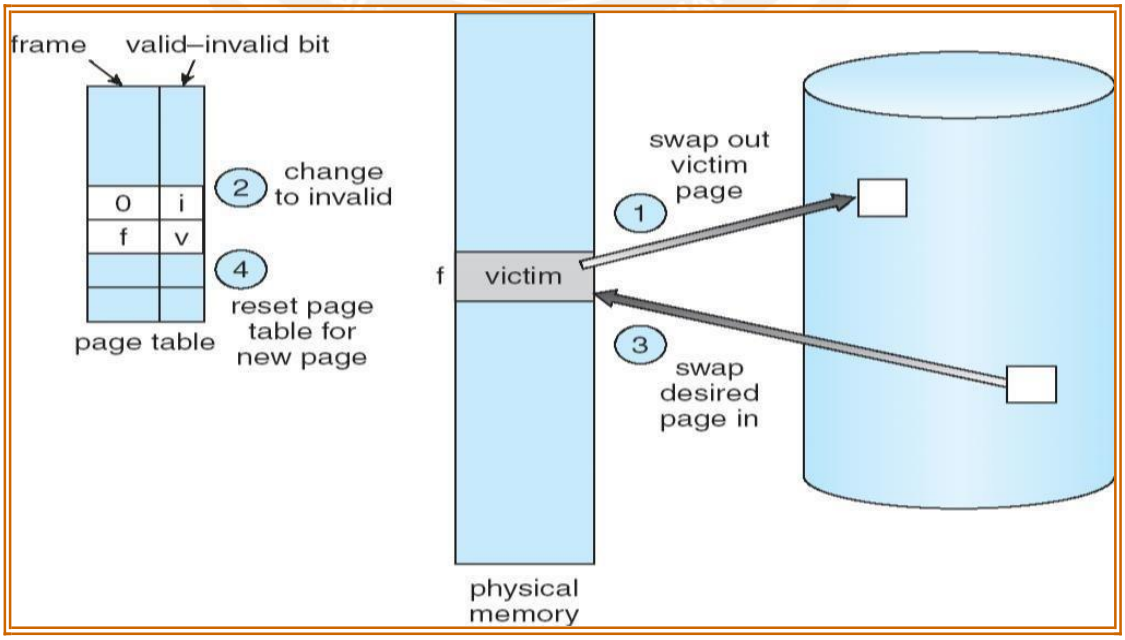
PAGE REPLACEMENT

- If no frames are free, we could find one that is not currently being used & free it.
- We can free a frame by writing its contents to swap space & changing the page table to indicate that the page is no longer in memory.
- Then we can use that freed frame to hold the page for which the process faulted.

Basic Page Replacement

1. Find the location of the desired page on disk
2. Find a free frame
    - If there is a free frame , then use it.
    - If there is no free frame, use a page replacement algorithm to select a **victim** frame
    - Write the victim page to the disk, change the page & frame tables accordingly.
3. Read the desired page into the (new) free frame. Update the page and frame tables.
4. Restart the process



Note:

If no frames are free, two page transfers are required & this situation effectively doubles the page- fault service time.

Modify (dirty) bit:

It indicates that any word or byte in the page is modified.

When we select a page for replacement, we examine its modify bit.

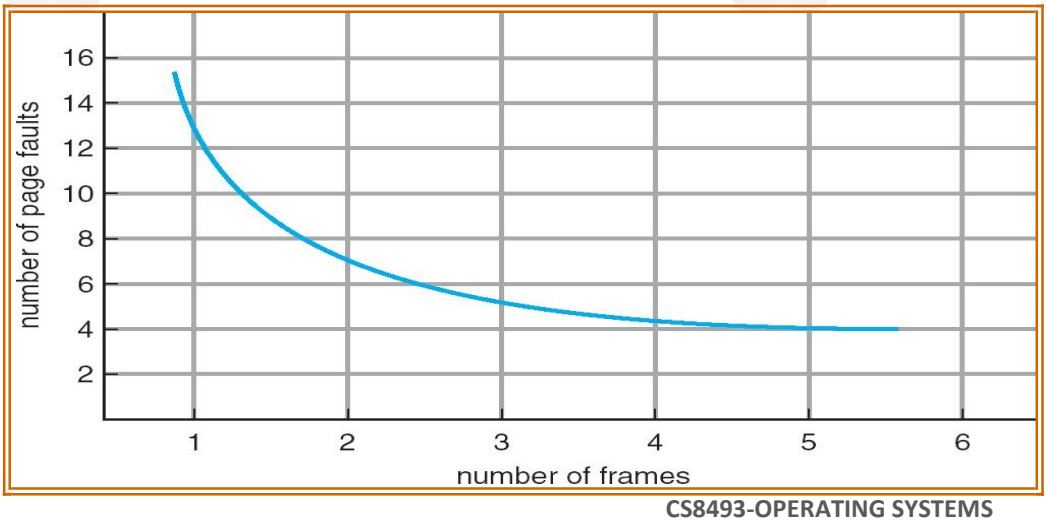
- If the bit is set, we know that the page has been modified & in this case we must write that page to the disk.
- If the bit is not set, then if the copy of the page on the disk has not been overwritten, then we can avoid writing the memory page on the disk as it is already there.

Page Replacement Algorithms

1. FIFO Page Replacement
2. Optimal Page Replacement
3. LRU Page Replacement
4. LRU Approximation Page Replacement
5. Counting-Based Page Replacement

We evaluate an algorithm by running it on a particular string of memory references & computing the number of page faults. The string of memory reference is called a "reference string" The algorithm that provides less number of page faults is termed to be a good one.

As the number of available frames increases , the number of page faults decreases. This is shown in the following graph:



(a) FIFO page replacement algorithm

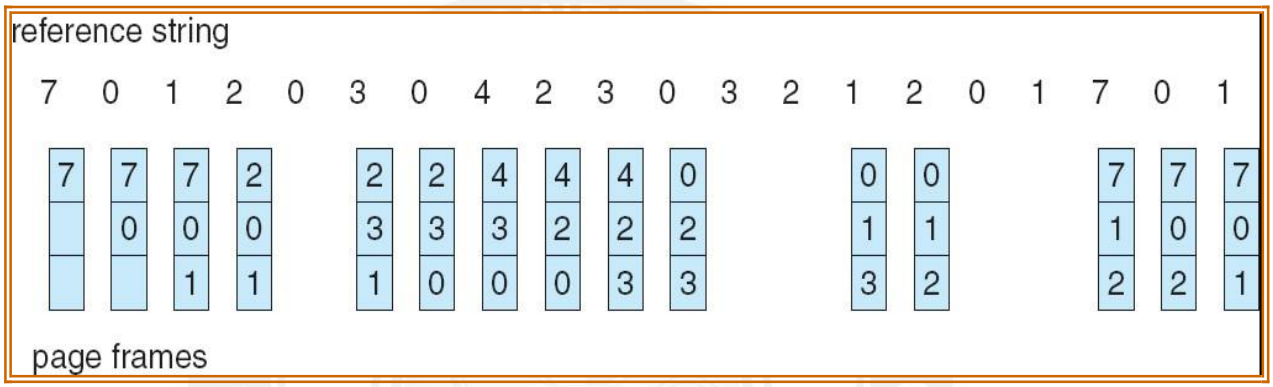
Replace the oldest page.

This algorithm associates with each page , the time when that page was brought in.

Example:

Reference string: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

No.of available frames = 3 (3 pages can be in memory at a time per process)



No. of page faults = 15

Drawback:

FIFO page replacement algorithm's performance is not always good.

To illustrate this, consider the following example:

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

- If No.of available frames -= 3 then the no.of page faults =9
- If No.of available frames =4 then the no.of page faults =10
- Here the no. of page faults increases when the no.of frames increases .This is called as **Belady's Anomaly.**

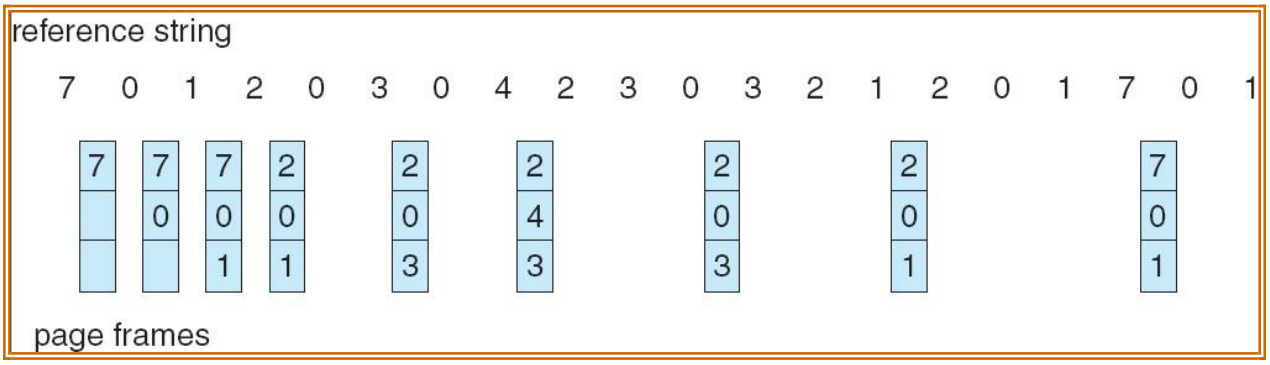
(b) Optimal page replacement algorithm

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

Example:

Reference string: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

No.of available frames = 3



No. of page faults = 9

Drawback:

It is difficult to implement as it requires future knowledge of the reference string.

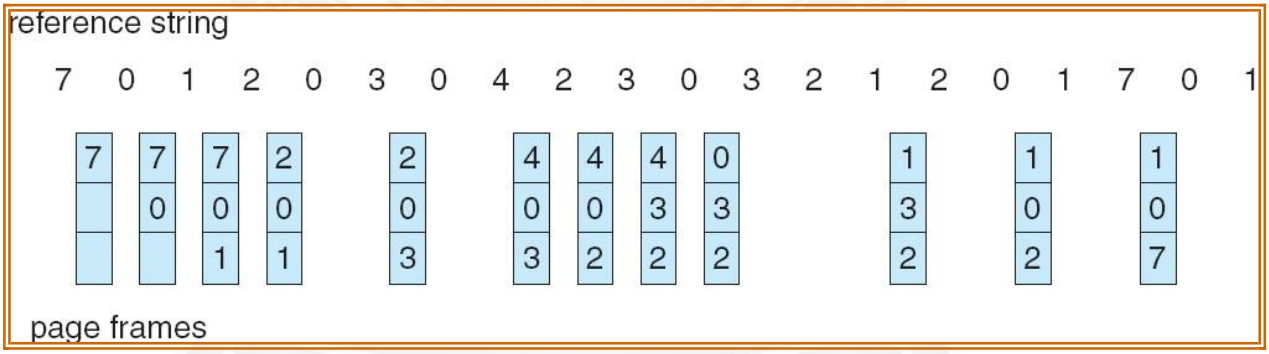
(c) LRU(Least Recently Used) page replacement algorithm

Replace the page that has not been used for the longest period of time.

Example:

Reference string: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

No.of available frames = 3



No. of page faults = 12

LRU page replacement can be implemented using

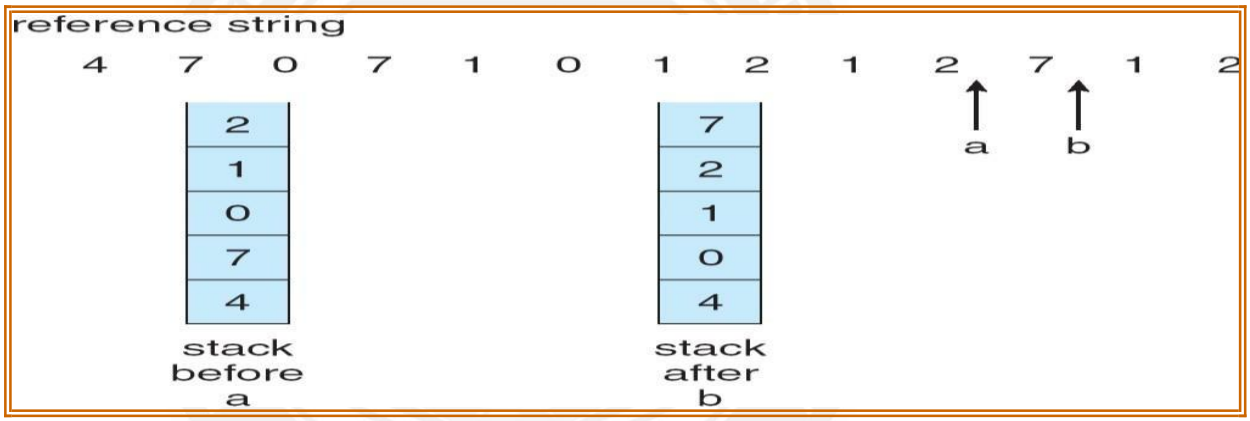
1. **Counters**
   - Every page table entry has a time-of-use field and a clock or counter is associated with the CPU.
   - The counter or clock is incremented for every memory reference.
   - Each time a page is referenced , copy the counter into the time-of-use field.

- When a page needs to be replaced, replace the page with the smallest counter value.

2. Stack

- Keep a stack of page numbers
- Whenever a page is referenced, remove the page from the stack and put it on top of the stack.
- When a page needs to be replaced, replace the page that is at the bottom of the stack.(LRU page)

Use of A Stack to Record The Most Recent Page References



(d) LRU Approximation Page Replacement

Reference bit

- With each page associate a reference bit, initially set to 0
- When page is referenced, the bit is set to 1

- When a page needs to be replaced, replace the page whose reference bit is 0
- The order of use is not known , but we know which pages were used and which were not used.

(i) Additional Reference Bits Algorithm

- Keep an 8-bit byte for each page in a table in memory.
- At regular intervals , a timer interrupt transfers control to OS.
- The OS shifts reference bit for each page into higher- order bit shifting the other bits right 1 bit and discarding the lower-order bit.

Example:

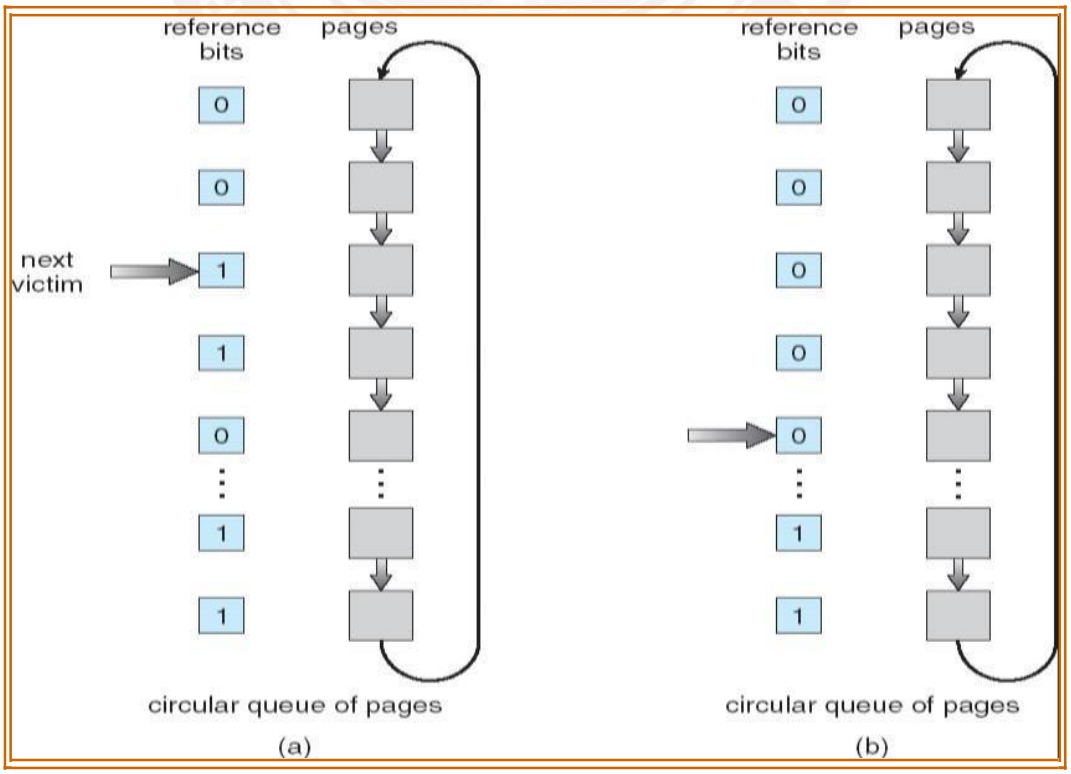
- If reference bit is 00000000 then the page has not been used for 8 time periods.
- If reference bit is 11111111 then the page has been used atleast once each time period.
- If the reference bit of page 1 is 11000100 and page 2 is 01110111 then page 2 is the LRU page.

(ii) Second Chance Algorithm

Basic algorithm is FIFO

When a page has been selected , check its reference bit.

- If 0 proceed to replace the page
- If 1 give the page a second chance and move on to the next FIFO page.
- When a page gets a second chance, its reference bit is cleared and arrival time is reset to current time.
- Hence a second chance page will not be replaced until all other pages are replaced.



(iii) Enhanced Second Chance Algorithm

Consider both reference bit and modify bit o There are four possible classes

1. (0,0) – neither recently used nor modified Best page to replace
2. (0,1) – not recently used but modified page has to be written out before replacement.
3. (1,0) - recently used but not modified page may be used again
4. (1,1) – recently used and modified page may be used again and page has to be written to disk

(e) Counting-Based Page Replacement

Keep a counter of the number of references that have been made to each page

1. **Least Frequently Used (LFU)Algorithm**: replaces page with smallest count
2. **Most Frequently Used (MFU)Algorithm**: replaces page with largest count

-It is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

Page Buffering Algorithm

These are used along with page replacement algorithms to improve their performance

Technique 1:

- A pool of free frames is kept.
- When a page fault occurs, choose a victim frame as before.
- Read the desired page into a free frame from the pool
- The victim frame is written onto the disk and then returned to the pool of free frames.

Technique 2:

- Maintain a list of modified pages.
- Whenever the paging device is idles, a modified is selected and written to disk and its modify bit is reset.

Technique 3:

- A pool of free frames is kept.
- Remember which page was in each frame.
- If frame contents are not modified then the old page can be reused directly from the free frame pool when needed

Allocation of Frames

There are two major allocation schemes

- Equal Allocation
- Proportional Allocation

Equal allocation

If there are n processes and m frames then allocate m/n frames to each process.

Example: If there are 5 processes and 100 frames, give each process 20 frames.

Proportional allocation

Allocate according to the size of process

Let s_i be the size of process i.

Let m be the total no. of frames

Then $S = \sum s_i$

$a_i = s_i / S * m$where a_i is the no.of frames allocated to process i.

Global vs. Local Replacement

Global replacement – each 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.