DEPARTMENT OF AIML
23CST202- OPERATING SYSTEMS
II YEAR IV SEM AIML-B
UNIT 3-MEMORY MANAGEMENT
TOPIC –PAGE REPLACEMENT,ALLOCATION OF FRAMES

PAGE REPLACEMENT

In an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when a new page comes in. Page replacement becomes necessary when a page fault occurs and no free page frames are in memory. in this article, we will discuss different types of page replacement algorithms.

Page Replacement Algorithms
Page replacement algorithms are techniques used in operating systems to manage memory efficiently when the physical memory is full. When a new page needs to be loaded into physical memory**,** and there is no free space, these algorithms determine which existing page to replace.

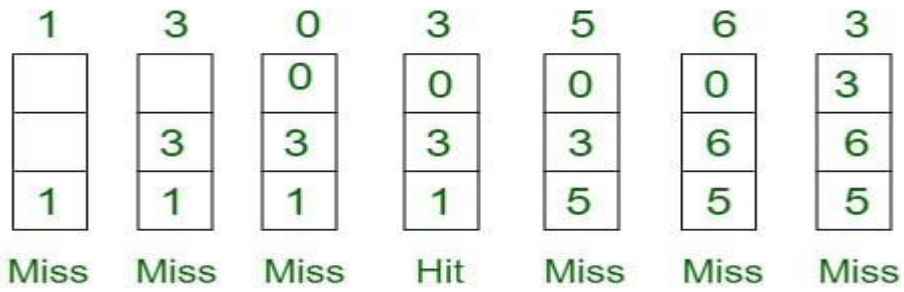
If no page frame is free, the virtual memory manager performs a page replacement operation to replace one of the pages existing in memory with the page whose reference caused the page fault. It is performed as follows: The virtual memory manager uses a page replacement algorithm to select one of the pages currently in memory for replacement, accesses the page table entry of the selected page to mark it as "not present" in memory, and initiates a page-out operation for it if the modified bit of its page table entry indicates that it is a dirty page.

Common Page Replacement Techniques
- First In First Out (FIFO)
- Optimal Page replacement
- Least Recently Used (LRU)
- Most Recently Used (MRU)

First In First Out (FIFO)
This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

Example 1: Consider page reference string 1, 3, 0, 3, 5, 6, 3 with 3-page frames. Find the number of page faults using FIFO Page Replacement Algorithm.

Page reference 1, 3, 0, 3, 5, 6, 3



Total Page Fault = 6

FIFO – Page Replacement

Initially, all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> **3 Page Faults.**
when 3 comes, it is already in memory so —> 0 **Page Faults.** Then 5 comes, it is not available in memory, so it replaces the oldest page slot i.e 1. —> 1 **Page Fault.** 6 comes, it is also not available in memory, so it replaces the oldest page slot i.e 3 —> 1 **Page Fault.** Finally, when 3 come it is not available, so it replaces 0 1-page **fault.**

Implementation of FIFO Page Replacement Algorithm

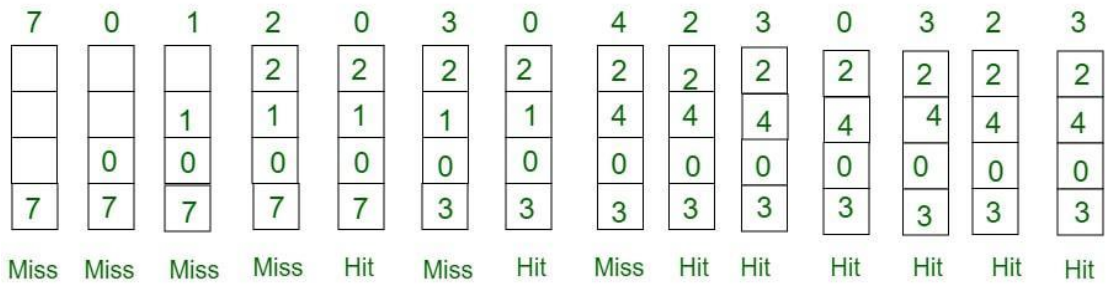
- Program for Page Replacement Algorithm (FIFO)

Optimal Page Replacement

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

Example: Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4-page frame. Find number of page fault using Optimal Page Replacement Algorithm.

Page reference 7,0,1,2,0,3,0,4,2,3,0,3,2,3 No. of Page frame - 4



Total Page Fault = 6

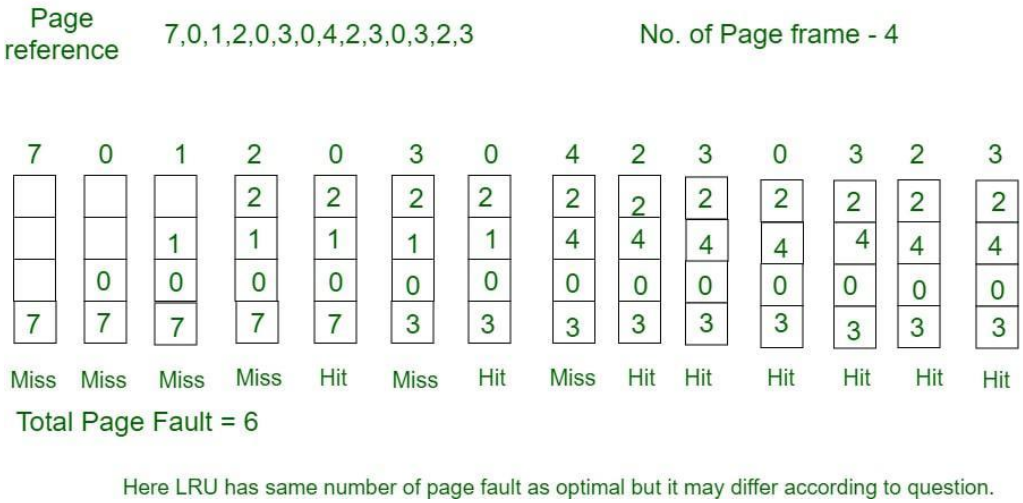
Optimal Page Replacement

Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**
0 is already there so —> **0 Page fault.** when 3 came it will take the place of 7 because it is not used for the longest duration of time in the future—> **1 Page fault.** 0 is already there so —> **0 Page fault.** 4 will takes place of 1 —> **1 Page Fault.**

Now for the further page reference string —> 0 **Page fault because** they are already available in the memory. Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

Least Recently Used
In this algorithm, page will be replaced which is least recently used.

Example Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4-page frames. Find number of page faults using LRU Page Replacement Algorithm.



Here LRU has same number of page fault as optimal but it may differ according to question.

Least Recently Used – Page Replacement

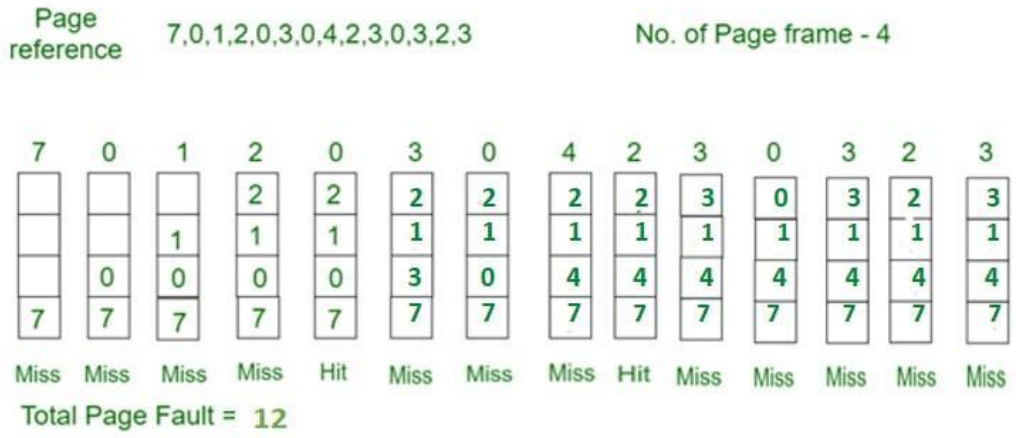
Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**
0 is already there so —> **0 Page fault.** when 3 came it will take the place of 7 because it is least recently used —> **1 Page fault**
0 is already in memory so —> **0 Page fault**.
4 will takes place of 1 —> **1 Page Fault**
Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

Implementation of LRU Page Replacement Algorithm

- Program for Least Recently Used (LRU) Page Replacement algorithm

Most Recently Used (MRU)
In this algorithm, page will be replaced which has been used recently. Belady's anomaly can occur in this algorithm.

Example 4: Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4-page frames. Find number of page faults using MRU Page Replacement Algorithm.



Most Recently Used – Page Replacement

Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**

0 is already their so–> **0 page fault**

when 3 comes it will take place of 0 because it is most recently used —> **1 Page fault**

when 0 comes it will take place of 3 —> **1 Page fault**

when 4 comes it will take place of 0 —> **1 Page fault**

2 is already in memory so —> **0 Page fault**

when 3 comes it will take place of 2 —> **1 Page fault**

when 0 comes it will take place of 3 —> **1 Page fault**

when 3 comes it will take place of 0 —> **1 Page fault**

when 2 comes it will take place of 3 —> **1 Page fault**

when 3 comes it will take place of 2 —> **1 Page fault**

ALLOCATION OF FRAMES

An important aspect of operating systems, virtual memory is implemented using demand paging. Demand paging necessitates the development of a page-replacement algorithm and a **frame allocation algorithm**. Frame allocation algorithms are used if you have multiple processes; it helps decide how many frames to allocate to each process.

There are various constraints to the strategies for the allocation of frames:

- You cannot allocate more than the total number of available frames.

- At least a minimum number of frames should be allocated to each process. This constraint is supported by two reasons. The first reason is, as less number of frames are allocated, there is an increase in the page fault ratio, decreasing the performance of the execution of the process. Secondly, there should be enough frames to hold all the different pages that any single instruction can reference.

Frame allocation algorithms –

The two algorithms commonly used to allocate frames to a process are:

1. **Equal allocation:** In a system with x frames and y processes, each process gets equal number of frames, i.e. x/y. For instance, if the system has 48 frames and 9 processes, each process will get 5 frames. The three frames which are not allocated to any process can be used as a free-frame buffer pool.
- **Disadvantage:** In systems with processes of varying sizes, it does not make much sense to give each process equal frames. Allocation of a large number of frames to a small process will eventually lead to the wastage of a large number of allocated unused frames.

2. **Proportional allocation:** Frames are allocated to each process according to the process size.
   For a process pi of size si, the number of allocated frames is **ai = (si/S)*m**, where S is the sum of the sizes of all the processes and m is the number of frames in the system. For instance, in a system with 62 frames, if there is a process of 10KB and another process of 127KB, then the first process will be allocated (10/137)*62 = 4 frames and the other process will get (127/137)*62 = 57 frames.
- **Advantage:** All the processes share the available frames according to their needs, rather than equally.

Global vs Local Allocation –

The number of frames allocated to a process can also dynamically change depending on whether you have used **global replacement** or **local replacement** for replacing pages in case of a page fault.

1. **Local replacement:** When a process needs a page which is not in the memory, it can bring in the new page and allocate it a frame from its own set of allocated frames only.
- **Advantage:** The pages in memory for a particular process and the page fault ratio is affected by the paging behavior of only that process.
- **Disadvantage:** A low priority process may hinder a high priority process by not making its frames available to the high priority process.

2. **Global replacement:** When a process needs a page which is not in the memory, it can bring in the new page and allocate it a frame from the set of all frames, even if that frame is currently allocated to some other process; that is, one process can take a frame from another.
- **Advantage:** Does not hinder the performance of processes and hence results in greater system throughput.
- **Disadvantage:** The page fault ratio of a process can not be solely controlled by the process itself. The pages in memory for a process depends on the paging behavior of other processes as well.