OS (Memory Management)

★★Memory Management

- Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution.
- Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free.
- It decides which process will get memory at what time.
- It checks how much memory is to be allocated to processes.
- It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

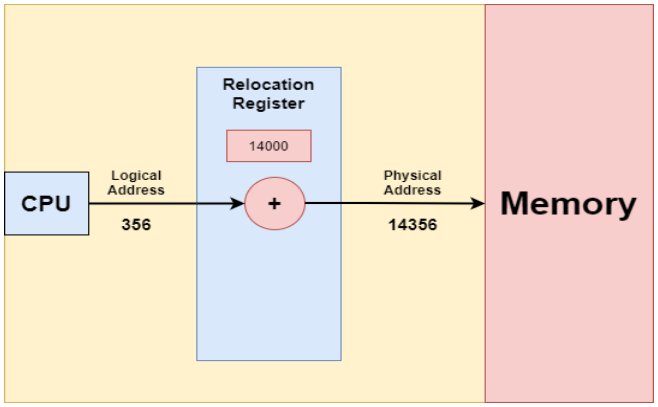
★★Logical Address and Physical Address

S.No	Logical Address	Physical Address
1.	The logical address is generated by the CPU. It is basically the address of an instruction or data used by any program.	The physical address is located in the memory unit.
2.	The logical address does not exist physically in the memory and thus termed as a Virtual address.	A physical address can be accessed by a user indirectly but not directly. The user can access/calculate the physical address with the help of a logical address.
3.	The set of all logical addresses that are generated by any program is referred to as **Logical Address Space.**	The set of all **physical addresses** corresponding to the **Logical addresses** is commonly known as **Physical Address Space.**
4.	This address is generated by the CPU.	It is computed by the Memory Management Unit(MMU).

★★Memory Management Unit (MMU) in OS

It is a hardware device that does the run-time mapping from the **virtual address** to the **physical address**. It is located within the Central Processing Unit.

Let us understand the concept of mapping with the help of a simple MMU scheme and that is a **base-register scheme**.



In the above diagram, the base register is termed the **Relocation register.** The relocation register is a special register in the CPU and is used for the mapping of logical addresses used by a program to physical addresses of the system's main memory.

The value in the relocation register is added to every address that is generated by the user process at the time when the address is sent to the memory.

☆☆MMU Example

Suppose the **base is at 14000**, then an attempt by the user to address location **0** is relocated dynamically to **14000 (0 + 14000)**; thus access to location **356** is mapped to **14356.**

*It is important to note that the **user program** never sees the real physical addresses*. The Program can create a pointer to location 356 and store it in the memory and then manipulate it after that compare it with other addresses as number 356.

User program always deals with the logical addresses. The **Memory Mapping Unit (MMU)** mainly converts the logical addresses into the physical addresses. The final location of the referenced memory address is not determined **until the reference is made.**

★★Main Memory

Main Memory refers to a physical memory that is the internal memory to the computer. The word main is used to distinguish it from external mass storage devices such as disk drives. Main memory is also known as RAM. The computer is able to change only data that is in main memory. Therefore, every program we execute and every file we access must be copied from a storage device into main memory.

Dynamic Loading: All the programs are loaded in the main memory for execution. Sometimes complete program is loaded into the memory, but sometimes a certain part or routine of the program is loaded into the main memory only when it is called by the program, this mechanism is called **Dynamic Loading**, this enhance the performance.

Dynamic Linking: Also, at times one program is dependent on some other program. In such a case, rather than loading all the dependent programs, **CPU links** the dependent programs to the main executing program when its required. This mechanism is known as **Dynamic Linking**.

★★What is swapping? Why it is needed?

Swapping is a **mechanism** in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes. At some later time, the system swaps back the process from the secondary storage to main memory.

Though performance is usually affected by swapping process but it helps in running multiple and big processes in parallel and that's the reason **Swapping is also known as a technique for memory compaction**.

[Compaction is the process in which the *free spaces are collected together in the large memory chunk* to make some space available for processes.]

☆☆ **How to swap two processes using a disk as a back store?**

A process must be in memory to be executed. A process, however, can be **swapped** temporarily out of memory to a **backing store** and then brought back into memory for continued execution.

For example, assume a multiprogramming environment with a round-robin CPU-scheduling algorithm. When a quantum expires, the memory manager will start to swap out the process that just finished and to swap another process into the memory space that has been freed (**Figure 8.5**).

In the meantime, the CPU scheduler will allocate a time slice to some other process in memory. *When each process finishes its quantum, it will be swapped with another process*.

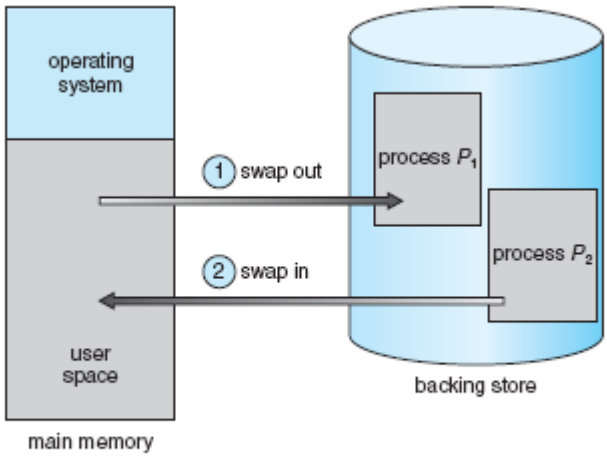


Figure 8.5 Swapping of two processes using a disk as a backing store.

This **variant** of **swapping** is sometimes called ***roll out, roll in***. Normally a process that is swapped out will be swapped back into the same memory space that it occupied previously.

★★**Memory Protection**

- Memory protection is a **phenomenon** by which we control memory access rights on a computer.
- The **main aim** of it is to prevent a process from accessing memory that has not been allocated to it.
- Hence prevents a bug within a process from affecting other processes, or the operating system itself, and instead results in a segmentation fault or storage violation exception being sent to the disturbing process, generally killing of process.

★★Contiguous Memory Allocation

- Contiguous Memory is defined as **a type of memory allocation where data is stored in a sequential and uninterrupted manner in the system's memory**.
- In contiguous memory allocation each process is contained in a single contiguous block of memory.
- Memory is divided into several **fixed size partitions**. **Each partition** contains exactly **one process**.
- When a partition is free, a process is selected from the input queue and loaded into it. The free blocks of memory are known as *holes*. The set of holes is searched to determine which hole is best to allocate.

☆☆ Memory Allocation

Memory allocation is **a process** by which computer programs are assigned memory or space. It is mainly of three types:

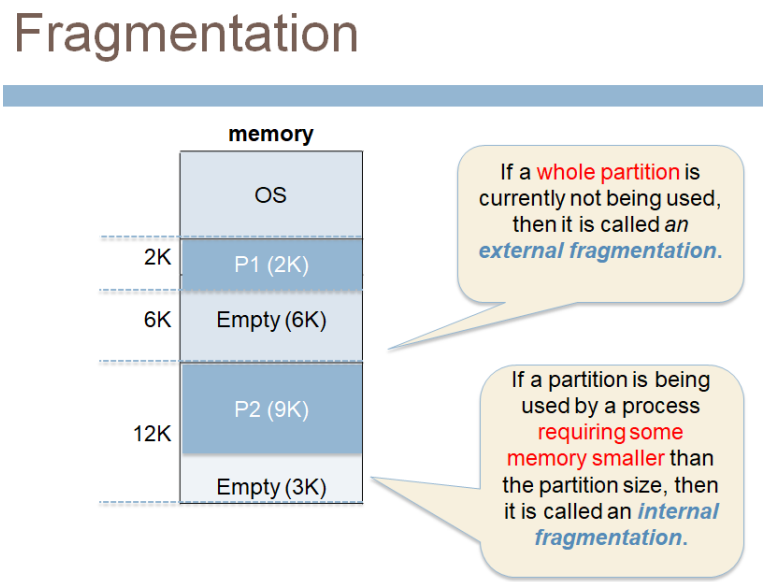
1. **First Fit:** The first hole that is big enough is allocated to program.
2. **Best Fit:** The smallest hole that is big enough is allocated to program.
3. **Worst Fit:** The largest hole that is big enough is allocated to program

Memory Allocation scheme is related with Partitioning, fragmentation and other topics. These topics are described in detail in the "06. OS [Memory Management].pptx**"*

★★Fragmentation

Whenever a process is **loaded or removed** from the physical memory block, it creates a small hole in memory space which is called **fragment**. Due to fragmentation, the system fails in allocating the contiguous memory space to a process even though it have the requested amount of memory but, in a non-contiguous manner.

The fragmentation is further classified into **two categories Internal** and **External Fragmentation**.



Basis for Comparison	**Internal Fragmentation**	**External Fragmentation**
Definition	If a partition is being used by a process requiring some memory smaller than the partition size, then it is called an ***internal fragmentation*.**	If a whole partition is currently not being used, then it is called *an* ***external fragmentation*.**
Reason	It occurs when fixed sized memory blocks are allocated to the processes.	It occurs when variable size memory spaces are allocated to the processes dynamically.
Occurrence	When the memory assigned to the process is slightly larger than the memory requested by the process this creates free space in the allocated block causing internal fragmentation.	When the process is removed from the memory, it creates the free space in the memory causing external fragmentation.
Solution	The memory must be partitioned into variable sized blocks and assign the best fit block to the process.	Compaction, paging and segmentation.

★★Solution to Fragmentation (Paging in OS)

☆☆Paging

>> A solution to fragmentation problem is Paging**.**

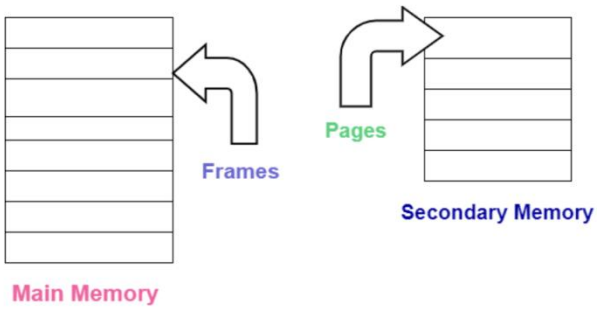
>> Paging is a memory management mechanism that allows the **physical address space** of a process to be non-contagious.

>> It is a **fixed-size partitioning scheme**. In the Paging technique, the secondary memory and main memory are divided into *equal fixed-size partitions*.

>> Paging helps to **avoid external fragmentation** and the **need for compaction**.

☆☆How Does Paging Work?

- The paging technique divides the **physical memory (main memory)** into fixed-size blocks that are known as **Frames** and also divide the **logical memory (secondary memory) into blocks of the same size** that are known as **Pages.** A frame is basically a place where a (logical) page can be (physically) placed.



- **Paging enables an OS to transfer data between secondary (virtual) and primary (physical) memory**.
  - *Whenever a program/process executes, it splits into pages, and the OS automatically stores them in secondary memory.
  - ***When the process requests memory**, the OS allocates **frames** from the primary memory to the **process.** Next, the OS moves program **pages** (of that requested Process) from the secondary memory to the primary memory frames. [i.e. transfer, as mentioned above]

    [Note: Pages of the process are only brought into the primary memory when needed. Otherwise, they stay in secondary storage.]

- The logical pages and physical page frames maintain the relationship using a structure called the **page table**. The **memory management unit (MMU)** uses the page table to translate logical addresses (virtual memory) into physical addresses (physical memory).

* **Page Table:** A Page Table is the **data structur**e used by a virtual memory system in a computer operating system to store the **mapping** between *virtual address* and *physical addresses.*

Virtual address is also known as Logical address and is generated by the CPU, while Physical address is the address that actually exists on memory.

☆☆Advantages of Paging

- Paging mainly allows to storage of parts of a single process in a **non-contiguous fashion**.
- With the help of **Paging, the problem** of external fragmentation is solved.
- Paging is one of the simplest algorithms for memory management.

☆☆Disadvantages of Paging

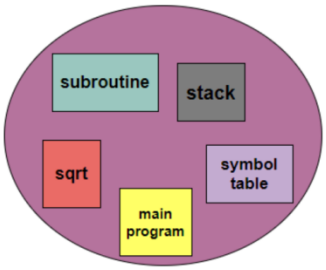
- In Paging, sometimes the page table consumes more memory.
- Internal fragmentation is caused by this technique.
- There is an increase in time taken to fetch the instruction since now two memory accesses are required.

★★Segmentation

- Segmentation is another memory management scheme that supports the user-view of memory.
- Basically, a process is divided into segments. **Like paging**, **segmentation** divides or segments the **logical memory (secondary memory)**. But there is a difference and that is while the **paging** divides the memory into a **fixed size** and on the other hand, segmentation divides the **memory into variable segments.**
- A Program is basically a collection of segments. And a **segment** is a logical unit such as:
  - main program
  - procedure
  - function/method
  - object
  - local variable and global variables.
  - symbol table

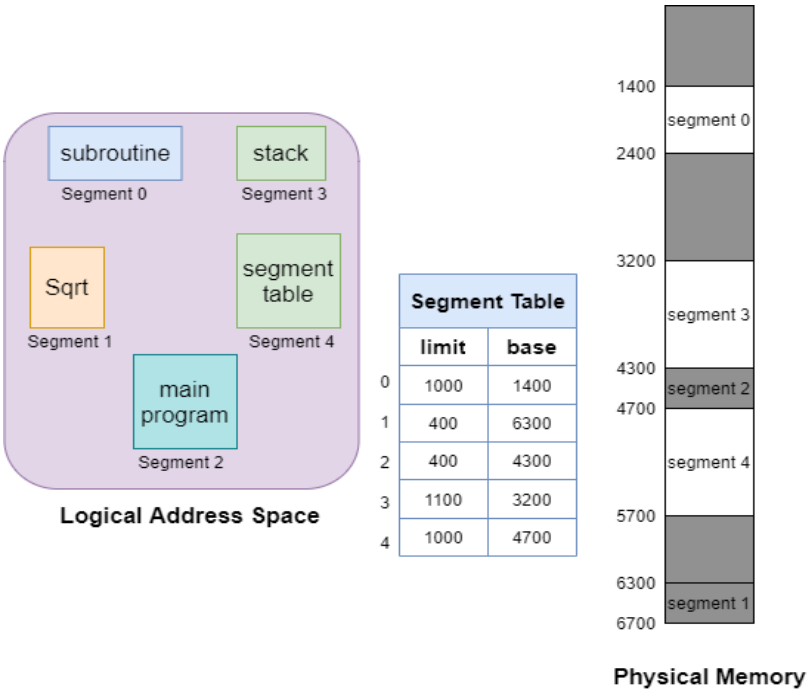
- common block
- stack
- arrays

Given below **figure** shows the user's view of segmentation (i.e. user-view of memory):



☆☆Example of Segmentation

- Given below is the example of the segmentation, There are five segments numbered from 0 to 4.
- These segments (from **logical/secondary memory**) will be stored in Physical memory as shown.
- There is a separate entry for each segment in the segment table which contains the beginning entry address of the segment in the physical memory (denoted as the **base**) and also contains the length of the segment(denoted as **limit**).



Segment 2 is 400 bytes long and begins at location 4300. Thus in this case a reference to byte 53 of segment 2 is mapped onto the location 4300 (4300+53=4353). A reference to segment 3, byte 85 is mapped to 3200(the base of segment 3)+852=4052.

☆☆**Advantages of Segmentation**

- In the Segmentation technique, the segment table is mainly used to keep the record of segments. Also, the segment table occupies less space as compared to the paging table.
- There is no Internal Fragmentation.
- Segments are of variable size.

☆☆**Disadvantages of Segmentation**

- Maintaining a segment table for each process leads to **overhead**
- This technique is **expensive.**
- Segments are of unequal size in segmentation and thus are not suitable for swapping.
- This technique leads to external fragmentation as the free space gets broken down into smaller pieces along with the processes being loaded and removed from the main memory then this will result in a lot of memory waste.

★★**Segmentation with Paging (Segmented paging)**

Both paging and segmentation have their advantages and disadvantages, it is better to combine these two schemes to improve on each. The combined scheme is known as **Segmented paging** or **'Page the Elements'**.

- **Segmented paging** is a way to manage computer memory that breaks it into small chunks called **segments**, and within each **segment**, there are fixed-sized **pages**. It's like dividing a big book into chapters, and each chapter into pages, to make it easier to read and manage.

- Each **virtual address (va)** in this method consists of a segment number (s), a page number (p), and an offset within that page (d). i.e. $va = (s, p, d)p$

- The segment number indexes into the ***segment table***, which returns the page table's base address for that segment. The page number is an index into the **page table**, each item of which represents a page frame. The **physical address** is obtained by adding the PFN (page frame number) and the **offset**.

  As a result, addressing may be defined by the function: $va = (s, p, d)p$

★★Virtual Memory

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called **virtual memory** and it is a section of a hard disk that's set up to emulate the computer's RAM.

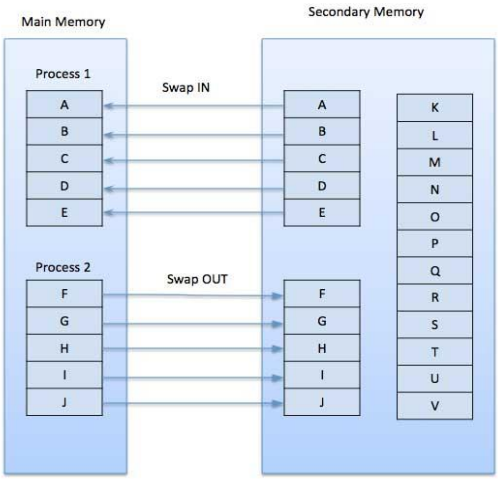
The main visible **advantage** of this scheme is that programs can be larger than physical memory. Virtual memory **serves two purposes**. First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address.

Virtual memory is commonly implemented by **demand paging**. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.

☆☆Demand Paging

A **demand paging system** is quite similar to a ***paging system with swapping*** where processes reside in secondary memory and pages are loaded only on demand, not in advance.

When a **context switch** occurs, the operating system does not copy any of the old programs pages out to the disk **or** any of the new programs pages into the main memory; **Instead**, it just begins executing the new program after loading the first page and fetches(swap in>> **Fig**) that programs pages as they are referenced.



While executing a program, if the program references a page which is not available in the main memory because it was swapped out a little ago, the processor treats this invalid memory reference as a **page fault** and transfers control from the program to the operating system to demand the page back into the memory.

Advantages

- Large virtual memory.
- More efficient use of memory.
- There is no limit on degree of multiprogramming.

Disadvantages

- Number of tables and the amount of processor overhead for handling page interrupts are greater than in the case of the simple paged management techniques.

☆☆ Related Terms

Page fault: We know every program is divided into **some pages/frames**. A **page fault** occurs when a program attempts to access data or code i.e. pages in its address space but is not currently located in the physical memory (main memory, RAM).

Swapping: Whenever a page fault happens, the operating system will try to **fetch** that page from secondary memory and try to swap it with one of the pages in RAM. This process is called **swapping**.

☆☆Performance of Demand Paging

The performance of demand paging is often measured in terms of the ***effective access time***. ***Effective access time*** is the amount of time it takes to access memory, if the cost of page faults are amortized over all memory accesses. In some sense it is an average or expected access time.

$$\textit{Effective access time, } ea = (1 - p) * ma + p * pft$$

Where, ea = effective access time, ma = physical memory (core) access time
pft = page fault time, p = probability of a page fault occurring
(1-p) = the probability of accessing memory in an available frame

Example: If we take an average page-fault service time of 25 milliseconds and a memory-access time of 100 nanoseconds, then the effective access time in nanoseconds is

Effective access time, ea = (1 - p) * (100) + p * (25 milliseconds)

= (1 - p) * 100 + p * 25,000,000

= 100 + 24,999,900 * p.

Amortized: to reduce a cost by paying small regular amounts

★★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 (RAM), and there is no free space, these algorithms determine which existing page to replace.

☆☆Basic Scheme - Page Replacement Algorithm in OS

Page Replacement technique uses the following approach. If there is no free frame, then we will find the one that is not currently being used and then free it. A-frame can be freed by writing its content to swap space and then change the page table in order to indicate that the page is no longer in the memory.

1. First of all, find the location of the desired **page** on the disk (from secondary memory).
2. Find a free Frame (in Physical Memory):
   a) If there is a free frame, then use it.
   b) If there is no free frame then **make use of the page-replacement algorithm** in order to find the one that is not currently being used and then **free it**.
3. After that read the desired page into the newly freed frame and then change the page and frame tables.
4. Restart the process.

☆☆Common Page Replacement Techniques

1. First In First Out (FIFO)
2. Optimal Page replacement (OPR)
3. Least Recently Used (LRU)
4. Most Recently Used (MRU)

(1) 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/head of the queue.
- Oldest page in main memory is the one which will be selected for replacement.

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.

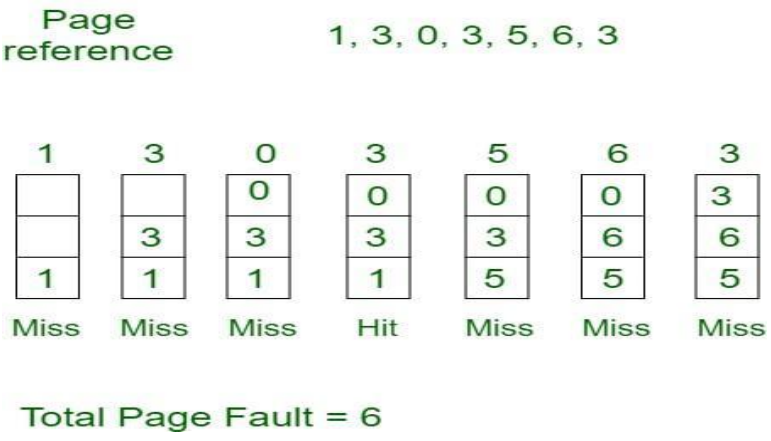


Figure: Example >> FIFO – Page Replacement

- Initially, all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> **3 Page Faults.**
  [as no pages found i.e. empty so page faults counts]
- When 3 come, 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.**

This is not an effective way of page replacement but it can be used for small systems.

[Hit and Miss: *When a page is found in the main memory, then it is known as hit. Hit radio is defined as the ratio of total number of hits to the sum of total number of hit and misses. In miss, when a page is not in main memory, it is found in secondary memory.]*

(2) Optimal Page Replacement (OPR)

- An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms.
- In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

Example 2: 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.

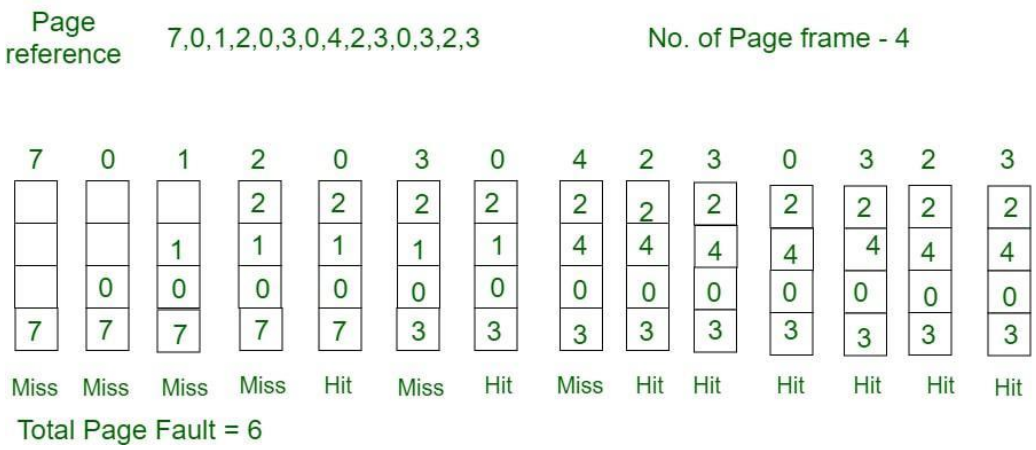


Figure: Example >> Optimal Page Replacement

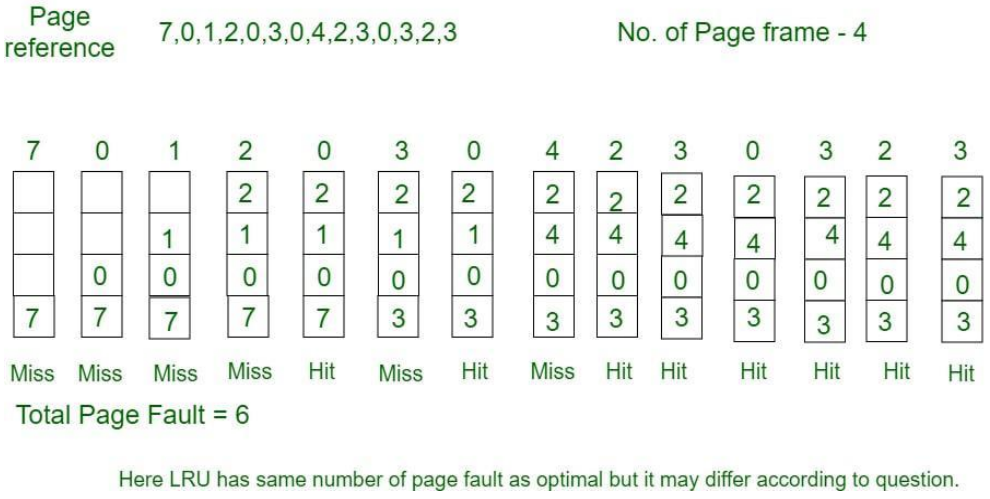
- Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**
- 0 is already there so —> **0 Page fault.**
- when 3 came it will take the place of 7 because it is not used for the longest duration of time in the future—> **1 Page fault.**
- 0 is already there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

(3) Least Recently Used (LRU)

- In this algorithm, page will be replaced which is least recently used (or, page which has not been used for the longest time).

Example 3: 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.

Figure: Example >> 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.

**Due to the same example, here OPR and LRU produce same result. It may differ according to question.*

(4) Most Recently Used (MRU)

- In this algorithm, page will be replaced which has been used recently.

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.

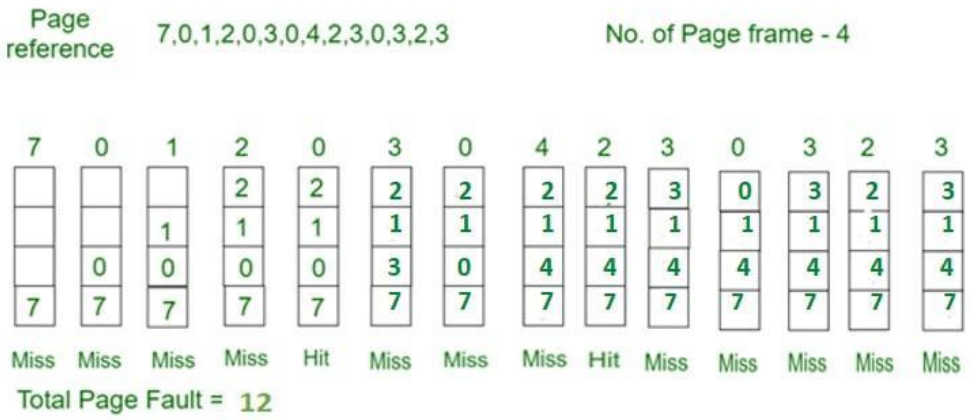


Figure: Example >> Most Recently Used – Page Replacement

- Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**
- 0 is already there 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 in OS

The main memory of the operating system is divided into various frames. The process is stored in these frames, and once the process is saved as a frame, the CPU may run it.

As a result, the operating system must set aside enough frames for each process. There are mainly **five ways of frame allocation** algorithms in the OS. These are as follows:

(1) Equal Frame Allocation: In equal frame allocation, the frames are assigned equally among the processes in the OS. For example, if the system has 30 frames and 7 processes, each process will get 4 frames. The 2 frames that are not assigned to any system process may be used as a free-frame buffer pool in the system.

(2) Proportional Frame Allocation: The proportional frame allocation technique **assigns frames** based on the size needed for execution and the total number of frames in memory.

The allocated frames for a process **pi** of size **si** are **ai = (si/S)*m**, in which **S** represents the total of all process sizes, and **m** represents the number of frames in the system.

(3) Priority Frame Allocation: Priority frame allocation assigns frames based on the Priority. Suppose a process has a high priority and requires more frames that many frames will be allocated to it. Following that, lesser priority processes are allocated.

(4) Global Replacement Allocation

When a process requires a frame that isn't currently in memory, it may put it in and select a frame from the all frames sets, even if another process is already utilizing that frame. In other words, one process may take a frame from another.

(5) Local Replacement Allocation

When a process requires a frame that isn't already in memory, it can bring it in and assign it a frame from its set of allocated frames.

Global Vs. Local Replacement Allocation

The number of frames assigned to a process does not change using a local replacement strategy. On the other hand, using global replacement, a process can choose only frames granted to other processes and enhance the number of frames allocated.

★★ Thrashing

In computer science, ***thrash*** is the poor performance of a virtual memory (or paging) system when the same pages are being loaded repeatedly due to a lack of main memory to keep them in memory. Depending on the configuration and algorithm, the actual throughput of a system can degrade by multiple orders of magnitude.

In computer science, ***thrashing*** occurs when a computer's virtual memory resources are overused, leading to a constant state of paging and page faults, inhibiting most application-level processing. It causes the performance of the computer to degrade or collapse. The situation can continue indefinitely until the user closes some running applications or the active processes free up additional virtual memory resources.

To know more clearly about thrashing, first, we need to know about page fault and swapping.

- **Page fault:** We know every program is divided into some pages. A page fault occurs when a program attempts to access data or code in its address space but is not currently located in the system RAM.
- **Swapping:** Whenever a page fault happens, the operating system will try to fetch that page from secondary memory and try to swap it with one of the pages in RAM. This process is called swapping.

Thrashing is when the page fault and swapping happens very frequently at a higher rate, and then the operating system has to spend more time swapping these pages. This state in the operating system is known as thrashing. Because of thrashing, the CPU utilization is going to be reduced or negligible.