UNIT – 4

MEMORY MANAGEMENT AND VIRTUAL MEMORY –

Logical versus Physical Address Space,

Swapping,

Contiguous Allocation,

Paging,

Segmentation,

Segmentation with Paging,

Demand Paging,

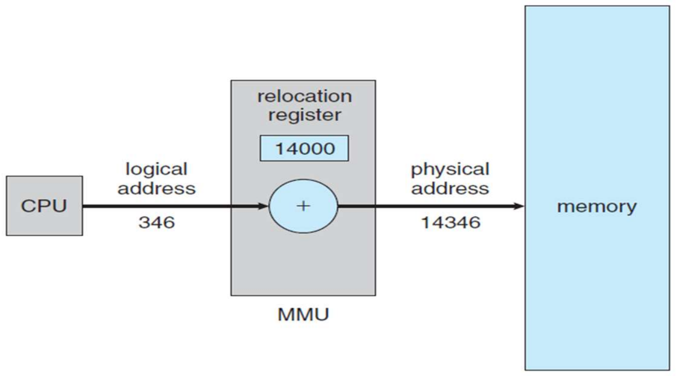
Page Replacement,

Page Replacement Algorithms.

4.1 **Logical versus Physical Address Space:**

a) Logical Address:

- A logical address in an operating system is also known as a virtual address. It represents a location in the computer's memory from the perspective of the executing process.
- These addresses are generated by the CPU during program execution. Logical addresses are used by programs to access data and instructions stored in memory.



b) Physical Address:

A physical address is the actual location in the computer's memory hardware, such as RAM or a storage device. Unlike logical addresses, which are generated by the CPU for processes, physical addresses are fixed and represent the true location of data in memory.

To continue with our previous example, when you open a document using your word processing application, the OS's memory manager translates the logical addresses generated by the CPU into physical addresses to fetch the actual content of the document from RAM. This translation is crucial because it ensures that each process can access the correct data without causing conflicts.

c) **Difference between Logical Address and Physical Address in Operating System**

S.No	Logical Address	Physical Address
1	Logical address is rendered by CPU.	Physical address is like a location that is present in the main memory.
2	It is a collection of all logical addresses rendered by the CPU.	It is a collection of all physical addresses mapped to the connected logical addresses.
3	Logical address of the program is visible to the users.	We cannot view the physical address of the program.
4	Logical address is generated by the CPU.	Physical address is computed by MMU.
5	We can easily utilise the logical address to access the physical address.	We can use the physical address indirectly.

4.2 Swapping:

Swapping in OS is one of those schemes which full fill the goal of maximum utilization of **CPU** and memory management by swapping in and swapping out processes from the main memory. Swap in removes the process from **hard drive**(secondary memory) and swap out removes the process from **RAM**(main memory).

There are several processes like P1, P2, P3, and P4 that are ready to be executed inside the ready queue, and processes P1 and P2 are very memory consuming so when the processes start executing there may be a scenario where the memory will not be available for the execution of the process P3 and P4 as there is a limited amount of memory available for process execution.

Swapping in the operating system is a memory management scheme that temporarily swaps out an idle or blocked process from the main memory to secondary memory which ensures proper memory utilization and memory availability for those processes which are ready to be executed.

The area of the secondary memory where swapped-out processes are stored is called **swap space**. The swapping method forms a temporary queue of swapped processes in the secondary memory.

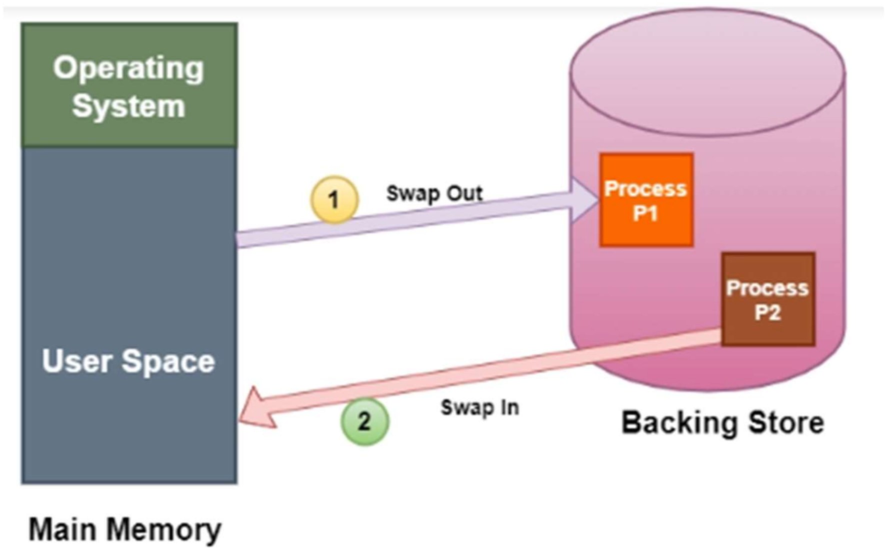
In the case of high-priority processes, the process with low priority is swapped out of the main memory and stored in swap space then the process with high priority is swapped into the main memory to be executed first.

The main goals of an operating system include **Maximum utilization of the CPU**. This means that there should be a process execution every time, the **CPU** should never stay idle and there should not be any **Process starvation** or **blocking**.

Different process management and memory management schemes are designed to fulfill such goals of an operating system.

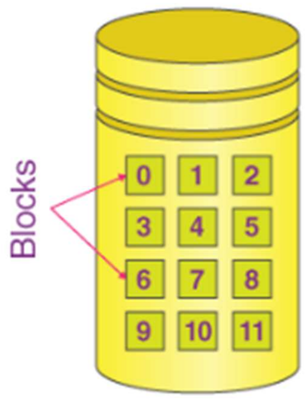
Swapping in **OS** is done to get access to data present in secondary memory and transfer it to the main memory so that it can be used by the application programs.

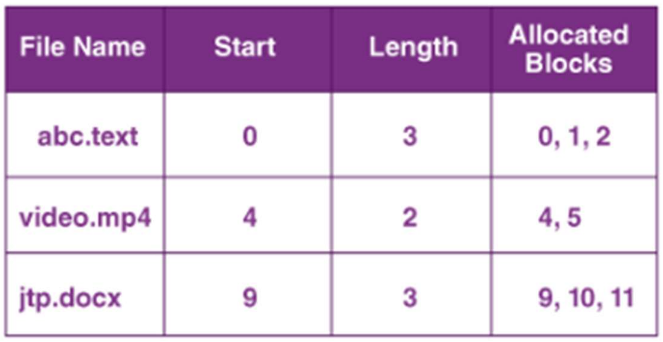
It can affect the performance of the system but it helps in running more than one process by managing the memory. Therefore swapping in os is also known as the **memory compaction technique**.



4.3 Contiguous Memory Allocation

Contiguous memory allocation refers to a memory management technique in which whenever there occurs a request by a user process for the memory, one of the sections of the contiguous memory block would be given to that process, in accordance with its requirement.





File Name	Start	Length	Allocated Blocks
abc.text	0	3	0, 1, 2
video.mp4	4	2	4, 5
jtp.docx	9	3	9, 10, 11

Directory

Contiguous Allocation

Types of Partitions
Contiguous memory allocation can be achieved when we divide the memory into the following types of partitions:
1. Fixed-Sized Partitions
Another name for this is static partitioning. In this case, the system gets divided into multiple fixed-sized partitions. In this type of scheme, every partition may consist of exactly one process. This very process limits the extent at which multiprogramming would occur, since the total number of partitions decides the total number of processes. Read more on fixed-sized partitions here.

2. Variable-Sized Partitions
Dynamic partitioning is another name for this. The scheme allocation in this type of partition is done dynamically. Here, the size of every partition isn't declared initially. Only once we know the process size, will we know the size of the partitions. But in this case, the size of the process and the partition is equal; thus, it helps in preventing internal fragmentation.
On the other hand, when a process is smaller than its partition, some size of the partition gets wasted (internal fragmentation). It occurs in static partitioning, and dynamic partitioning solves this issue. Read more on dynamic partitions here.

• First fit. Allocate the first hole that is big enough. Searching can start either at the beginning of the set of holes or at the location where the previous first-fit search ended. We can stop searching as soon as we find a free hole -that is large enough.•

Best fit. Allocate the smallest hole that is big enough. We must search the entire list, unless the list is ordered by size. This strategy produces the smallest leftover hole.

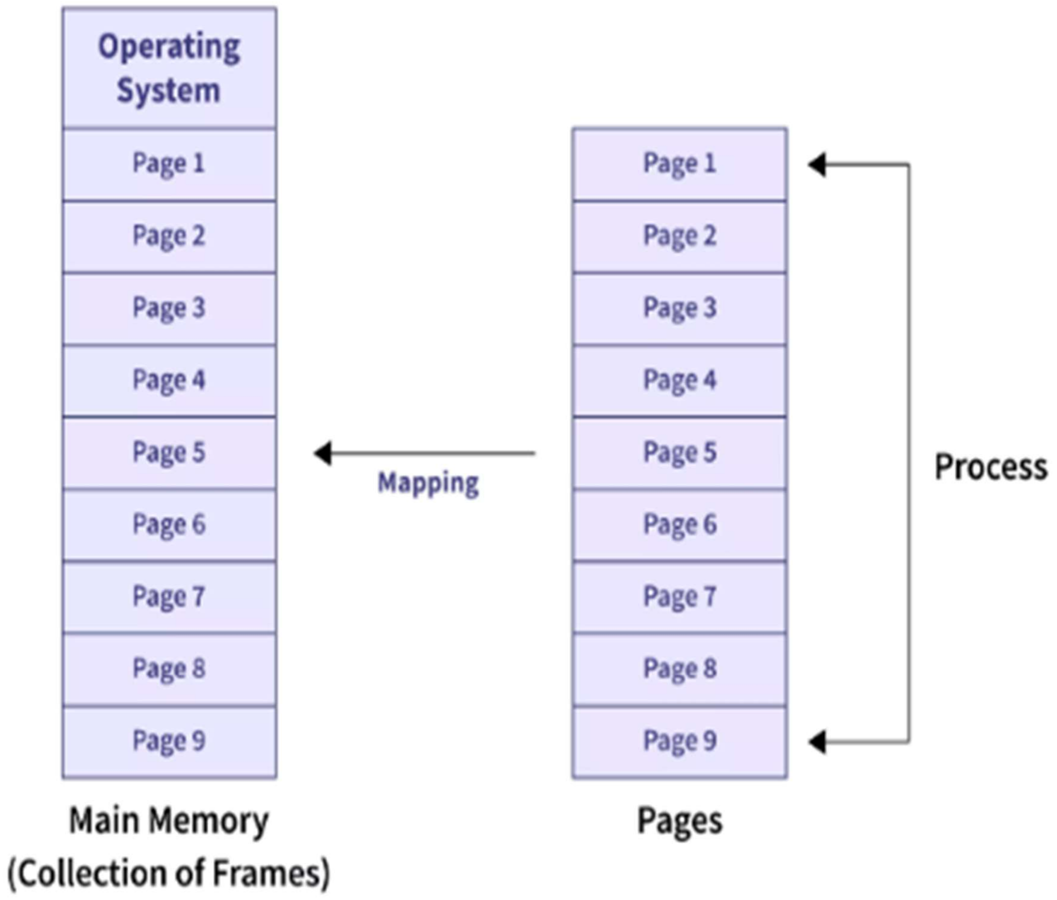
• Worst fit. Allocate the largest hole. Again, we must search the entire list, unless it is sorted by size. This strategy produces the largest leftover hole, which may be more useful than the smaller leftover hole from a best-fit approach.

4.4 Paging:

Paging is a technique that divides memory into fixed-sized blocks. The main memory is divided into blocks known as Frames and the logical memory is divided into blocks known as Pages. Paging requires extra time for the address conversion, so we use a special hardware cache memory known as TLB.

This concept of Paging in OS includes dividing each process in the form of pages of equal size and also, the main memory is divided in the form of frames of fixed size. Now, each page of the process when retrieved into the main memory, is stored in one frame of the memory, and hence, it is also important to have the pages and frames of equal size for mapping and maximum utilization of the memory.

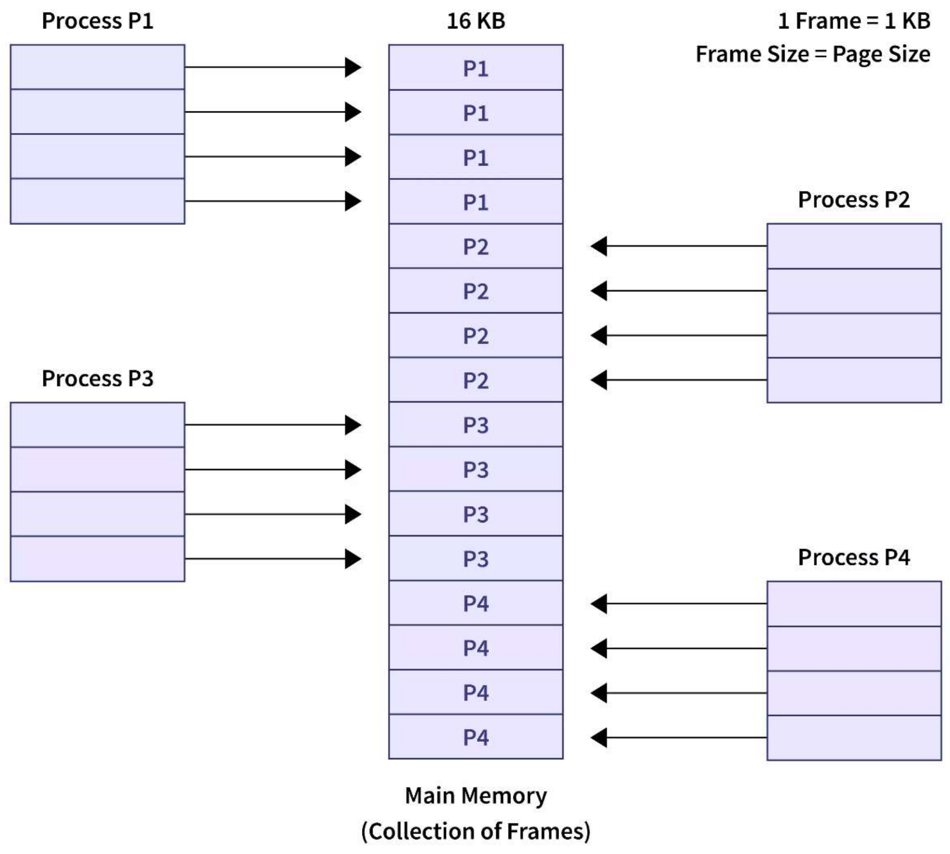
Its main advantage is that the pages can be stored at different locations of the memory and not necessarily in a contiguous manner, though priority is always set to firstly find the contiguous frames for allocating the pages.



If a process has n pages in the secondary memory, then there must be n frames available in the main memory for mapping.

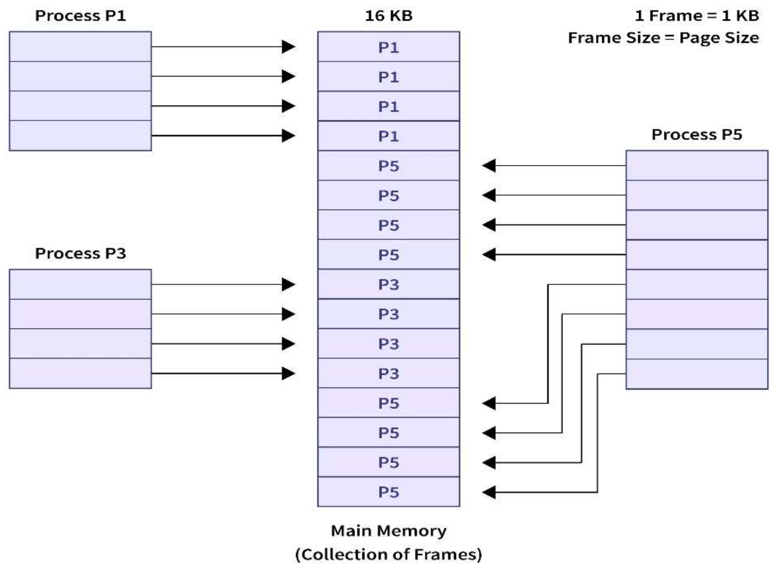
Example to understand Paging in OS

CASE-1 (Contiguous Allocation of Pages)



As we can see in the above image, we have main memory divided into 16 frames of the size of 1KB each. Also, there are 4 processes available in the secondary (local) memory: P1, P2, P3, and P4 of a size of 4KB each. Clearly, each process needs to be further subdivided into pages of size of 1KB each, so that one page can be easily mapped to one frame of the main memory. This divides each process into 4 pages and the total for 4 processes gives 16 pages of 1KB each. Initially, all the frames were empty and therefore, pages will be allocated here in a contiguous manner.

CASE-2 (Non-Contiguous Allocation of Pages)



Let us assume that in Case-1, processes P2 and P4 are moved to the waiting state after some time and leave behind the empty space of 8 frames. **In Case-2, we have another process P5 of size 8KB (8 pages) waiting inside the ready queue to be allocated.** We know that with Paging, we can store the pages at different locations of the memory, and here, we have8non-contiguous frames available. Therefore, we can easily load the 8 pages of the process P5 in the place of P2 and P4 which we can observe in the above image.

Memory Management Unit

MMU is a computer hardware component that responsible for all memory and caching operations which are associated with the CPU. MMU usually exist in CP but in some cases it operates on separate integrated circuit (IC) chip. Here's an overview of how the MMU, specifically in the context of paging, functions in an operating system:

- **Virtual Memory Translation:** The MMU translates virtual addresses generated by a process into physical addresses in RAM (Random Access Memory). This translation allows processes to run as if they have access to a large, continuous block of memory, even if the physical memory is fragmented or limited.
- **Paging:** Paging is a memory management scheme where physical memory is divided into fixed-size blocks called "frames," and logical memory is divided into equally sized blocks called "pages." The MMU maps pages to frames, enabling efficient allocation and management of memory.
- **Page Tables:** The MMU uses page tables to maintain the mapping between virtual pages and physical frames. Each process has its own page table, which keeps track of the virtual-to-physical address translations.
- **Page Faults:** When a process accesses a page that is not in physical memory, a page fault occurs. The MMU triggers an interrupt, and the operating system must load the required page from secondary storage (usually a hard disk) into a free frame in RAM.
- **Page Replacement:** If all frames in physical memory are in use, the operating system must choose a page to evict. This process, known as page replacement, is often managed using algorithms like the Least Recently Used (LRU) or the Second Chance algorithm.

4.5 Segmentation:

Segmentation divides processes into smaller subparts known as modules. The divided segments need not be placed in contiguous memory. Since there is no contiguous memory allocation, internal fragmentation does not take place. The length of the segments of the program and memory is decided by the purpose of the segment in the user program. We can say that logical address space or the main memory is a collection of segments. Hence, segmentation was introduced in which the code is divided into modules so that related code can be combined in one single block.

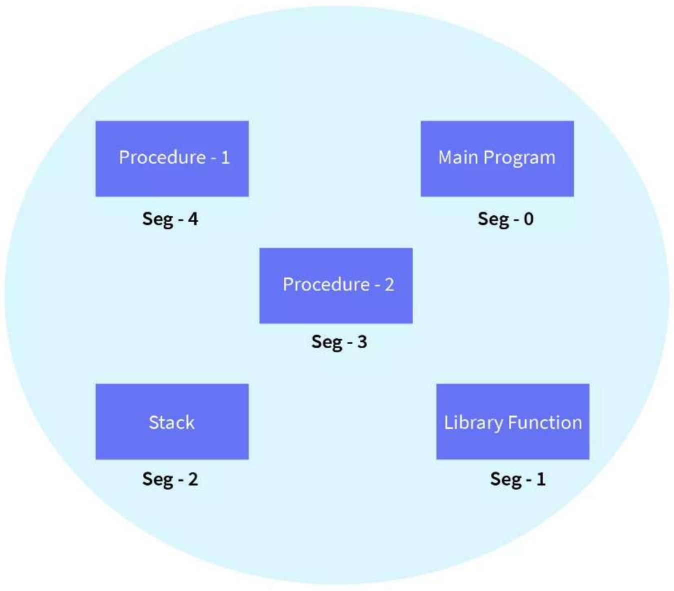
a) Types of Segmentation

Segmentation can be divided into two types:

i) **Virtual Memory Segmentation:** Virtual Memory Segmentation divides the processes into n number of segments. All the segments are not divided at a time. Virtual Memory Segmentation may or may not take place at the run time of a program.

ii) **Simple Segmentation:** Simple Segmentation also divides the processes into n number of segments but the segmentation is done all together at once. Simple segmentation takes place at the run time of a program. Simple segmentation may scatter the segments into the memory such that one segment of the process can be at a different location than the other (in a noncontinuous manner).

- In the case of the paging technique, a function or piece of code is divided into pages without considering that the relative parts of code can also get divided.
- Hence, for the process in execution, the CPU must load more than one page into the frames so that the complete related code is there for execution.
- Paging took more pages for a process to be loaded into the main memory.
- Other memory management techniques have also an important drawback - the actual view of physical memory is separated from the user's view of physical memory.
- Segmentation helps in overcoming the problem by dividing the user's program into segments according to the specific need.

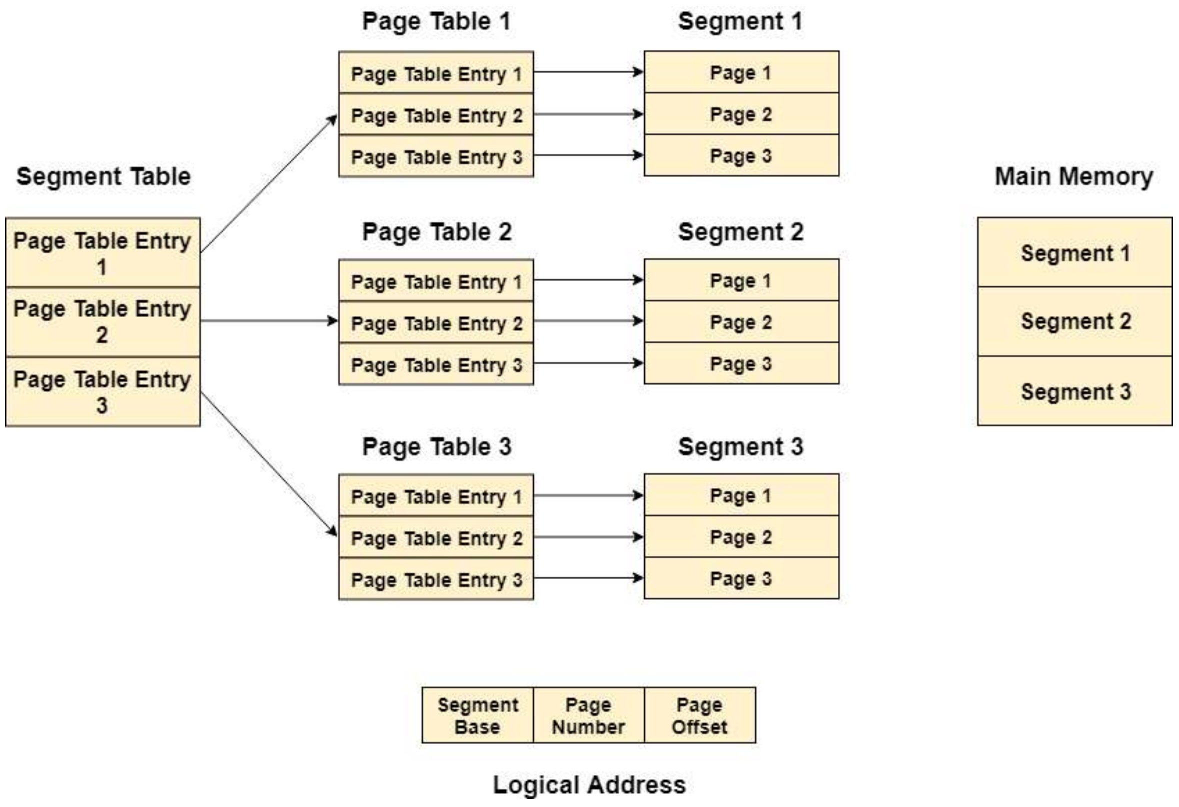


Advantages of Segmentation in OS:

- No internal fragmentation is there in segmentation.
- Segment Table is used to store the records of the segments. The segment table itself consumes small memory as compared to a page table in paging.
- Segmentation provides better CPU utilization as an entire module is loaded at once.
- Segmentation is near to the user's view of physical memory. Segmentation allows users to partition the user programs into modules. These modules are nothing but the independent codes of the current process.

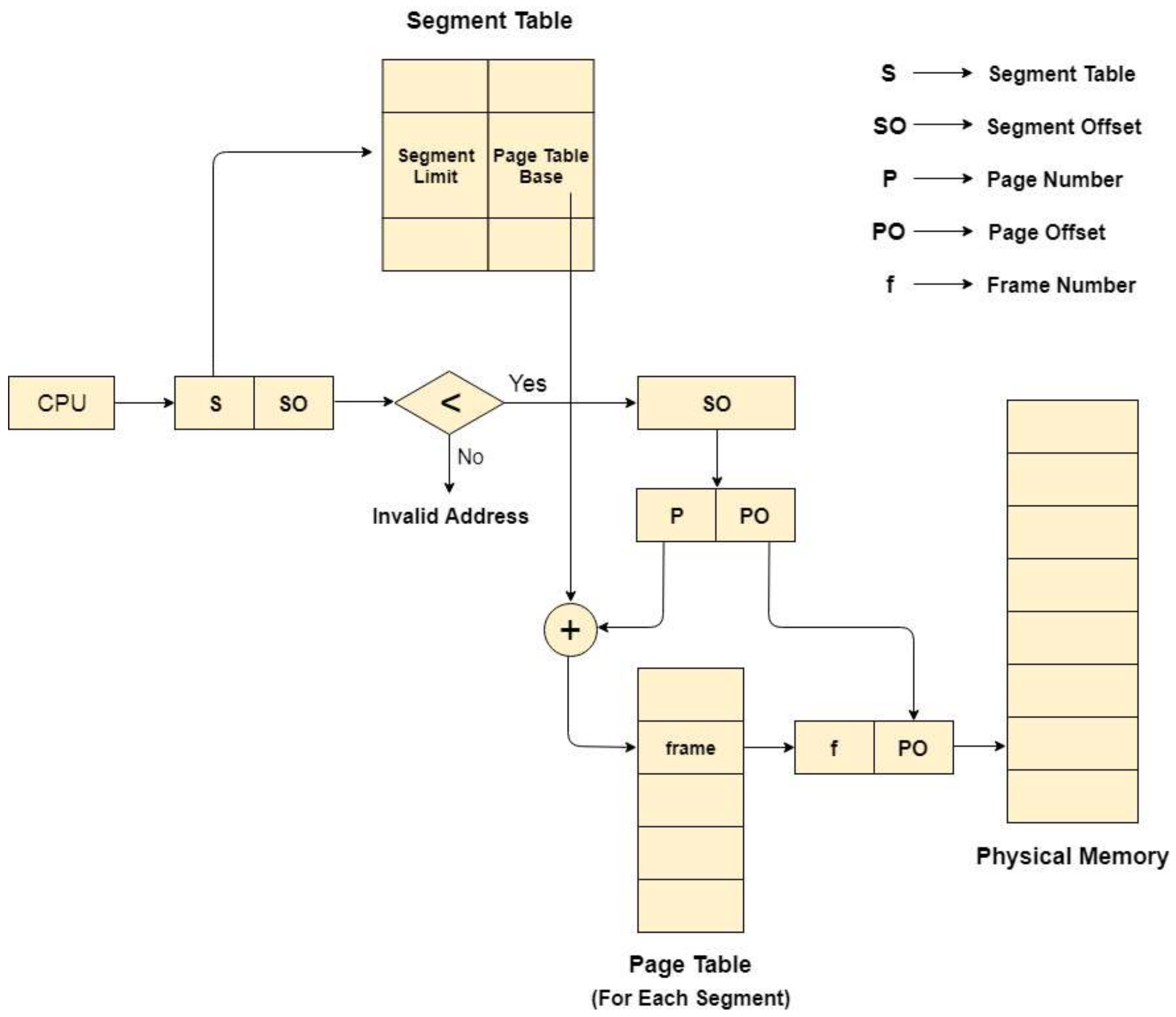
4.6 Segmentation with paging:

- A memory management strategy that combines segmentation with paging is called segmented paging.
- Pure segmentation is not very popular and not being used in many of the operating systems.
- Segmentation can be combined with Paging to get the best features out of both the techniques.
- In Segmented Paging, the main memory is divided into variable size segments which are further divided into fixed size pages.
- Pages are smaller than segments.
- Each Segment has a page table which means every program has multiple page tables.
- The logical address is represented as Segment Number (base address), Page number and page offset.
    - Segment Number → It points to the appropriate Segment Number.
    - Page Number → It Points to the exact page within the segment.
- **Page Offset** → Used as an offset within the page frame
- Each Page table contains the various information about every page of the segment.
- The Segment Table contains the information about every segment.
- Each segment table entry points to a page table entry and every page table entry is mapped to one of the page.



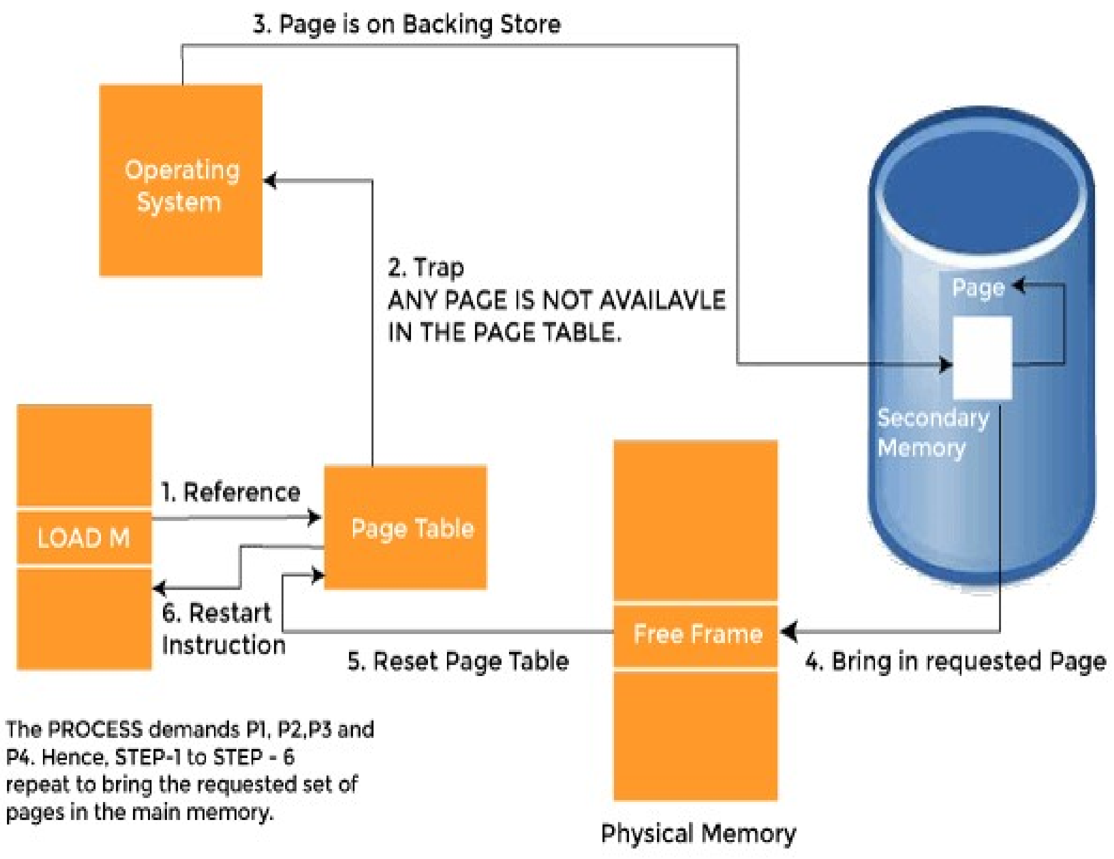
Translation of logical address to physical address

- The CPU generates a logical address which is divided into two parts: Segment Number and Segment Offset.
- The Segment Offset must be less than the segment limit. Offset is further divided into Page number and Page Offset.
- To map the exact page number in the page table, the page number is added into the page table base.
- The actual frame number with the page offset is mapped to the main memory to get the desired word in the page of the certain segment of the process.



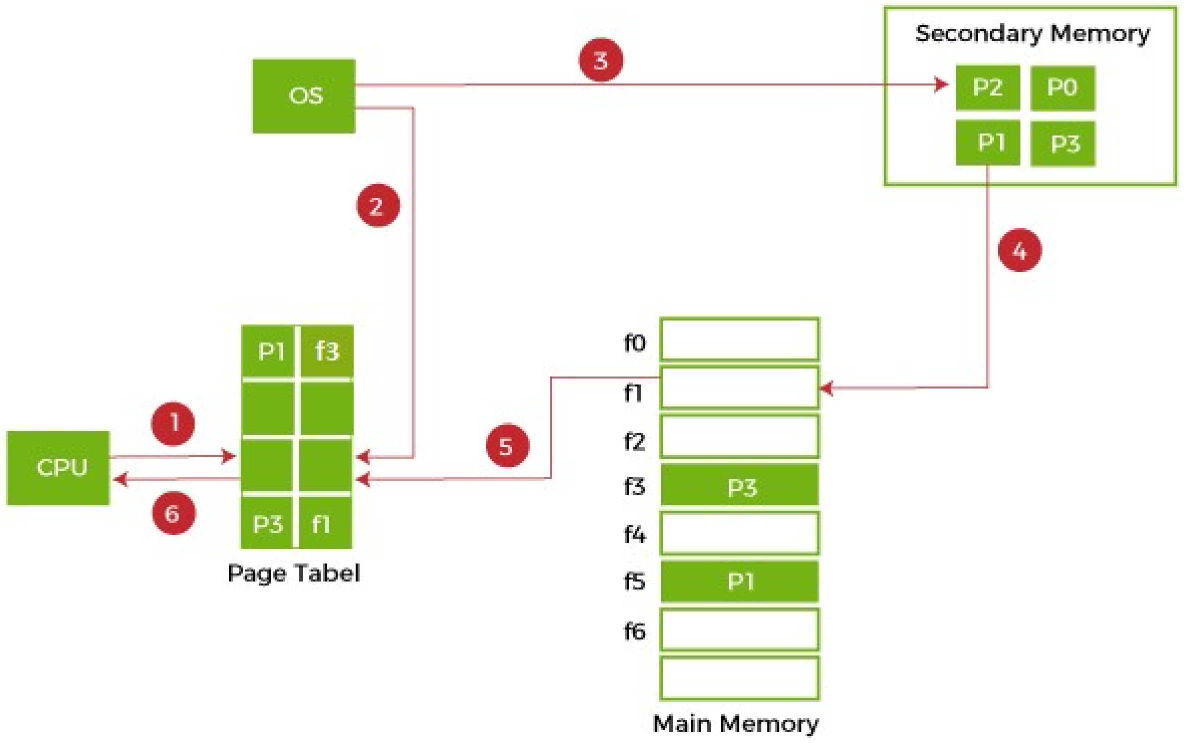
4.7 Demand Paging:

- Demand paging is a technique used in virtual memory systems where the pages are brought in the main memory only when required or demanded by the CPU.
- Hence, it is also called lazy swapper because the swapping of pages is done only when the CPU requires it.
- Virtual memory is commonly implemented in demand paging.
- In demand paging, the pager brings only those necessary pages into the memory instead of swapping in a whole process.
- Thus, demand paging avoids reading into memory pages that will not be used anyways, decreasing the swap time and the amount of physical memory needed.
- The demand paging system depends on the page table implementation because the page table helps map logical memory to physical memory.
- Bitwise operators are implemented in the page table to indicate that pages are ok or not (valid or invalid).
- All valid pages exist in primary memory, and invalid pages exist in secondary memory.
- Now all processes go-ahead to access all pages, and then the following things will be happened, such as:
    - Attempt to currently access page.
    - If the page is ok (Valid), then all processing instructions work as normal.
    - If anyone's page is found invalid, then a page fault issue arises.
    - Now memory reference is determined whether a valid reference exists on the auxiliary memory or not. If not exist, then the process is terminated, otherwise needed pages are paged in.
    - Now disk operations are implemented to fetch the required page into primary memory.


3. Page is on Backing Store
Operating System
2. Trap
ANY PAGE IS NOT AVAILAVLE IN THE PAGE TABLE.
Page
Secondary Memory
LOAD M
1. Reference
Page Table
6. Restart Instruction
5. Reset Page Table
Free Frame
4. Bring in requested Page
The PROCESS demands P1, P2,P3 and P4. Hence, STEP-1 to STEP - 6 repeat to bring the requested set of pages in the main memory.
Physical Memory

Examples of Demand Paging

Suppose we have to execute a process P having four pages P0, P1, P2, and P3. Currently, in the page table, we have pages P1 and P3.



- If the CPU wants to access page P2 of a process P, it will first search the page in the page table.
- As the page table does not contain this page so it will be a trap or page fault. The control goes to the operating system as soon as the trap is generated and context switching happens.
- The OS system will put the process in a waiting/ blocked state. The OS system will now search that page in the backing store or secondary memory.
- The OS will then read the page from the backing store and load it to the main memory.
- Next, the OS system will update the page table entry accordingly.
- Finally, the control is taken back from the OS, and the execution of the process is resumed.
- Hence, the operating system follows these steps whenever a page fault occurs and the required page is brought into memory.

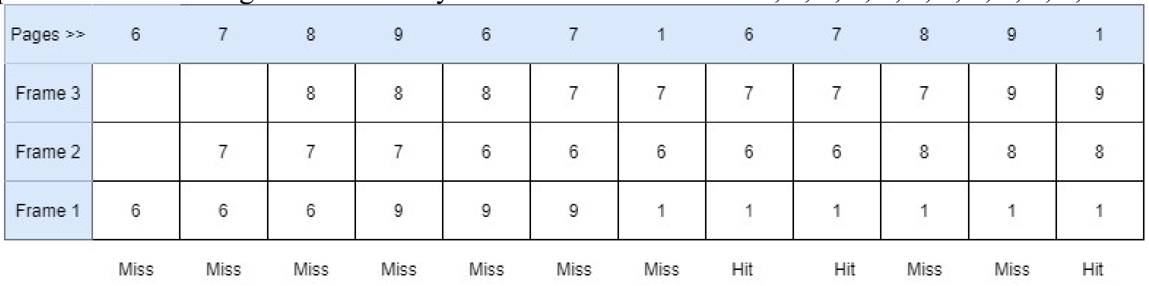
4.8 Page Replacement algorithm:

- Page Replacement Algorithm is used when a page fault occurs.
- Page Fault means the page referenced by the CPU is not present in the main memory.
- When the CPU generates the reference of a page, if there is any vacant frame available in the main memory then the page is loaded in that vacant frame.
- In another case, if there is no vacant frame available in the main memory, it is required to replace one of the pages in the main memory with the page referenced by the CPU.
- Page Replacement Algorithm is used to decide which page will be replaced to allocate memory to the current referenced page.
- Different Page Replacement Algorithms suggest different ways to decide which page is to be replaced.
- The main objective of these algorithms is to reduce the number of page faults.

a) First In First Out:

- This algorithm is similar to the operations of the queue.
- All the pages are stored in the queue in the order they are allocated frames in the main memory.
- The one which is allocated first stays in the front of the queue.
- The one which is allocated the memory first is replaced first.
- The one which is at the front of the queue is removed at the time of replacement.

Example: Consider the Pages referenced by the CPU in the order are 6, 7, 8, 9, 6, 7, 1, 6, 7, 8, 9, 1

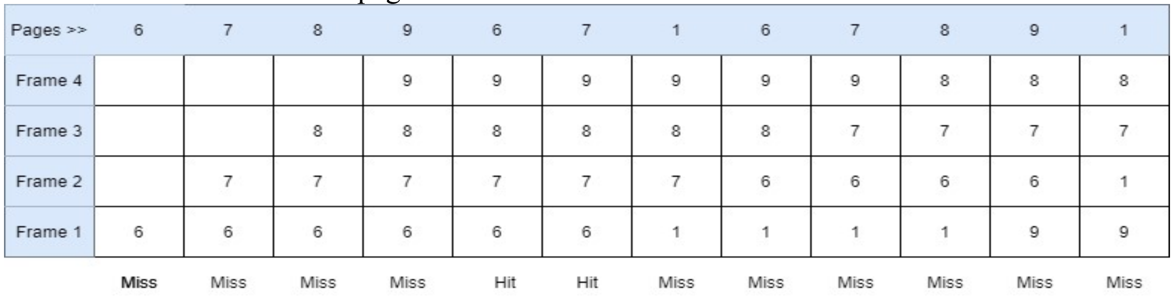


Pages >>	6	7	8	9	6	7	1	6	7	8	9	1
Frame 3			8	8	8	7	7	7	7	7	9	9
Frame 2		7	7	7	6	6	6	6	6	8	8	8
Frame 1	6	6	6	9	9	9	1	1	1	1	1	1
	Miss	Miss	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Hit

- As in the above figure shown, Let there are 3 frames in the memory.
- 6, 7, 8 are allocated to the vacant slots as they are not in memory.
- When 9 comes page fault occurs, it replaces 6 which is the oldest in memory or front element of the queue.
- Then 6 comes (Page Fault), it replaces 7 which is the oldest page in memory now.
- Similarly, 7 replaces 8, 1 replaces 9.
- Then 6 comes which is already in memory (Page Hit).
- Then 7 comes (Page Hit).
- Then 8 replaces 6, 9 replaces 7. Then 1 comes (Page Hit).
- Number of Page Faults = 9

"While using the First In First Out algorithm, the number of page faults increases by increasing the number of frames. This phenomenon is called Belady's Anomaly".

Let's take the same above order of pages with 4 frames.

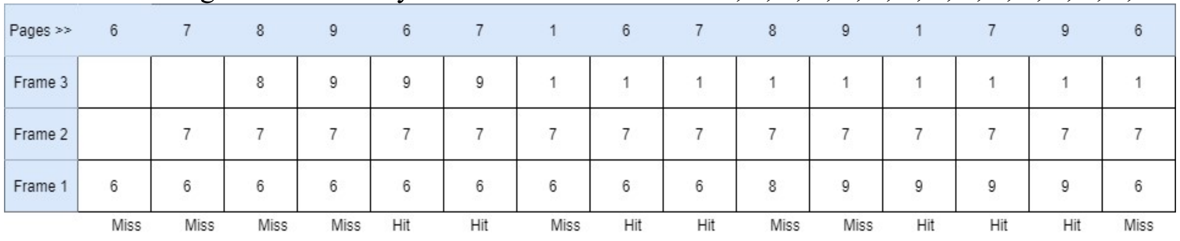


Pages >>	6	7	8	9	6	7	1	6	7	8	9	1
Frame 4				9	9	9	9	9	9	8	8	8
Frame 3			8	8	8	8	8	8	7	7	7	7
Frame 2		7	7	7	7	7	7	6	6	6	6	1
Frame 1	6	6	6	6	6	6	1	1	1	1	9	9
	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Miss	Miss	Miss	Miss

- In the above picture shown, it can be seen that the number of page faults is 10.
- There were 9 page faults with 3 frames and 10 page faults with 4 frames.
- The number of page faults increased by increasing the number of frames.

b) Optimal Page Replacement –

- In this algorithm, the page which would be used after the longest interval is replaced.
- In other words, the page which is farthest to come in the upcoming sequence is replaced.
- Example:
  Consider the Pages referenced by the CPU in the order are 6, 7, 8, 9, 6, 7, 1, 6, 7, 8, 9, 1, 7, 9, 6.

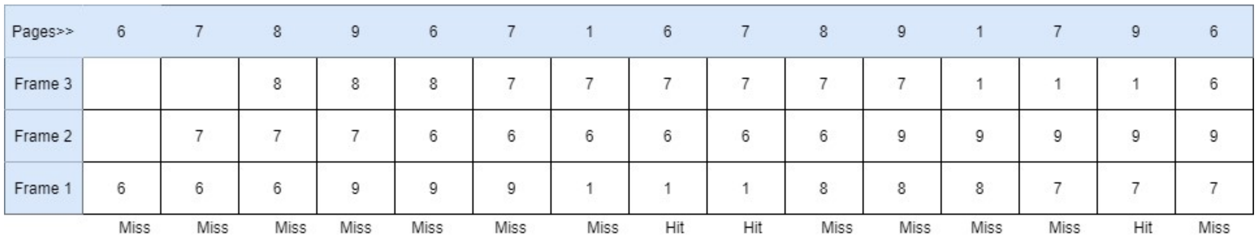


Pages >>	6	7	8	9	6	7	1	6	7	8	9	1	7	9	6
Frame 3			8	9	9	9	1	1	1	1	1	1	1	1	1
Frame 2		7	7	7	7	7	7	7	7	7	7	7	7	7	7
Frame 1	6	6	6	6	6	6	6	6	6	8	9	9	9	9	6
	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Hit	Hit	Miss	Miss	Hit	Hit	Hit	Miss

- First, all the frames are empty. 6, 7, 8 are allocated to the frames (Page Fault).
- Now, 9 comes and replaces 8 as it is the farthest in the upcoming sequence. 6 and 7 would come earlier than that so not replaced.
- Then, 6 comes which is already present (Page Hit).
- Then 7 comes (Page Hit).
- Then 1 replaces 9 similarly (Page Fault).
- Then 6 comes (Page Hit), 7 comes (Page Hit).
- Then 8 replaces 6 (Page Fault) and 9 replaces 8 (Page Fault).
- Then 1, 7, 9 come respectively which are already present in the memory.
- Then 6 replaces 9 (Page Fault), it can also replace 7 and 1 as no other page is present in the upcoming sequence.
- The number of Page Faults = 8
- This is the most optimal algorithm but is impractical because it is impossible to predict the upcoming page references.

c) Least Recently Used –

- This algorithm works on previous data.
- The page which is used the earliest is replaced or which appears the earliest in the sequence is replaced.
- Example:
- Consider the Pages referenced by the CPU in the order are 6, 7, 8, 9, 6, 7, 1, 6, 7, 8, 9, 1, 7, 9, 6

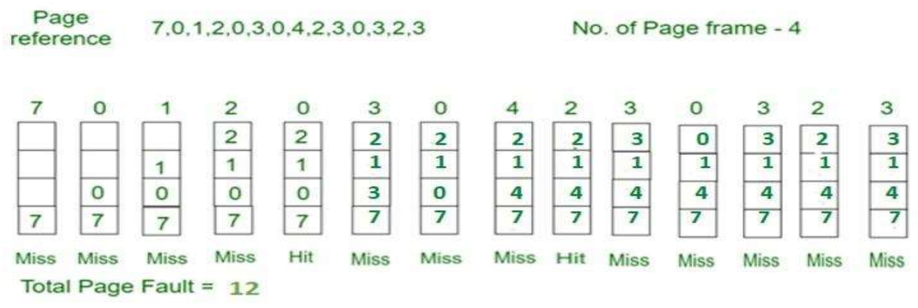


Pages>>	6	7	8	9	6	7	1	6	7	8	9	1	7	9	6
Frame 3			8	8	8	7	7	7	7	7	7	1	1	1	6
Frame 2		7	7	7	6	6	6	6	6	6	9	9	9	9	9
Frame 1	6	6	6	9	9	9	1	1	1	8	8	8	7	7	7
	Miss	Miss	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Miss	Miss	Hit	Miss

- First, all the frames are empty. 6, 7, 8 are allocated to the frames (Page Fault).
- Now, 9 comes and replaces 6 which is used the earliest (Page Fault).
- Then, 6 replaces 7, 7 replaces 8, 1 replaces 9 (Page Fault).
- Then 6 comes which is already present (Page Hit).
- Then 7 comes (Page Hit).
- Then 8 replaces 1, 9 replaces 6, 1 replaces 7, and 7 replaces 8 (Page Fault).
- Then 9 comes (Page Hit).
- Then 6 replaces 1 (Page Fault).
- The number of Page Faults = 12

d) 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.



- 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