Paging in OS (Operating System)

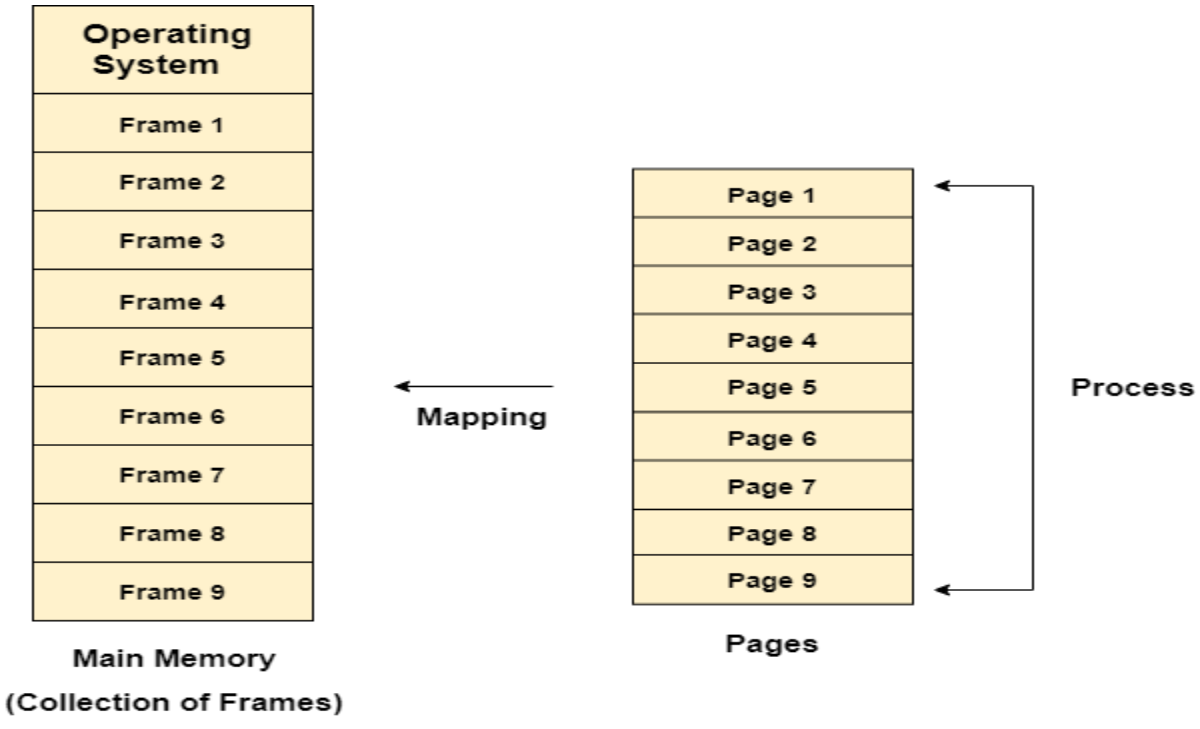
In Operating Systems, Paging is a storage mechanism used to retrieve processes from the secondary storage into the main memory in the form of pages.

The main idea behind the paging is to divide each process in the form of pages. The main memory will also be divided in the form of frames.

One page of the process is to be stored in one of the frames of the memory. The pages can be stored at the different locations of the memory but the priority is always to find the contiguous frames or holes.

Pages of the process are brought into the main memory only when they are required otherwise they reside in the secondary storage.

Different operating system defines different frame sizes. The sizes of each frame must be equal. Considering the fact that the pages are mapped to the frames in Paging, page size needs to be as same as frame size.



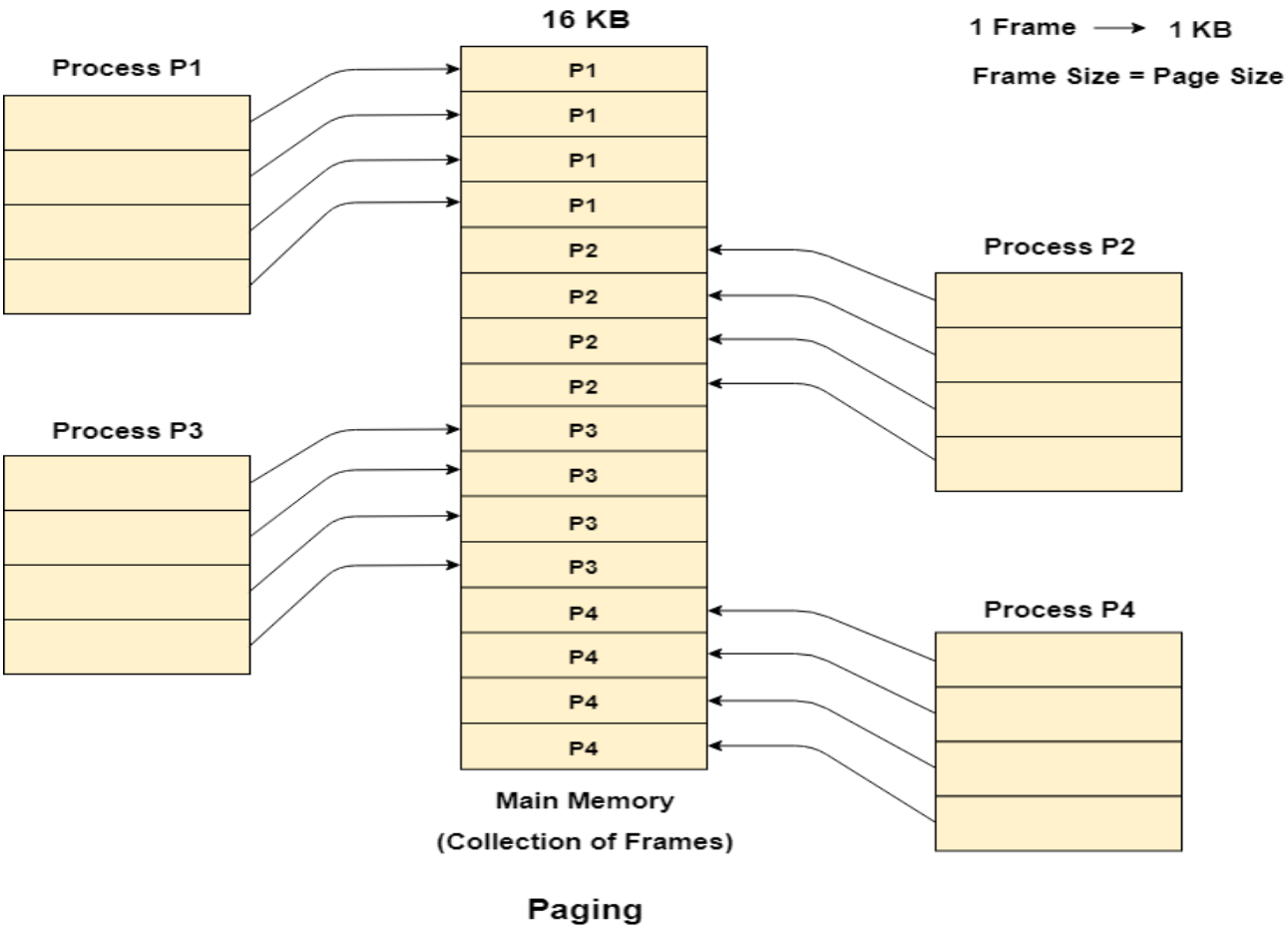
Example

Let us consider the main memory size 16 Kb and Frame size is 1 KB therefore the main memory will be divided into the collection of 16 frames of 1 KB each.

There are 4 processes in the system that is P1, P2, P3 and P4 of 4 KB each. Each process is divided into pages of 1 KB each so that one page can be stored in one frame.

Initially, all the frames are empty therefore pages of the processes will get stored in the contiguous way.

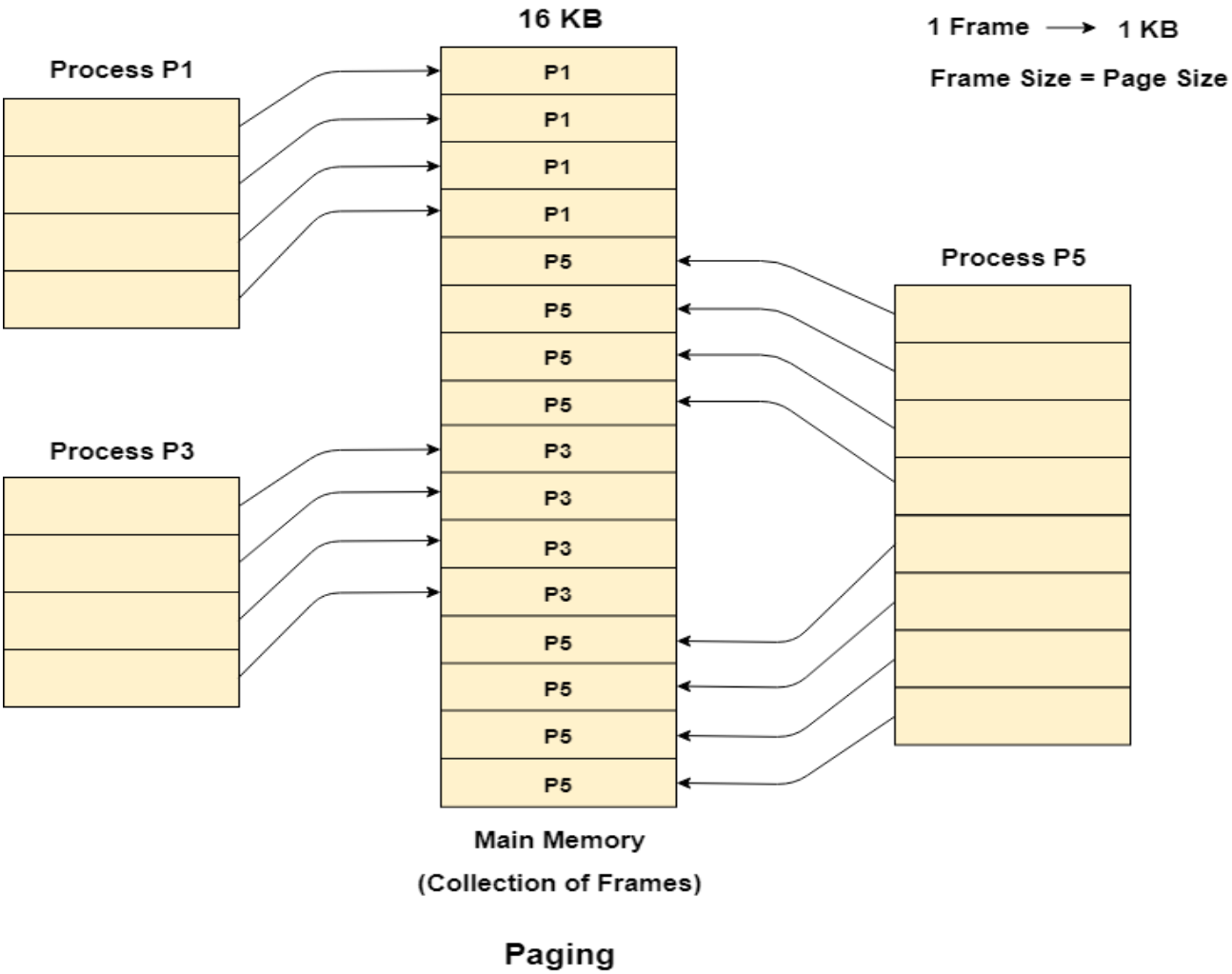
Frames, pages and the mapping between the two is shown in the image below.



Paging

Let us consider that, P2 and P4 are moved to waiting state after some time. Now, 8 frames become empty and therefore other pages can be loaded in that empty place. The process P5 of size 8 KB (8 pages) is waiting inside the ready queue.

Given the fact that, we have 8 noncontiguous frames available in the memory and paging provides the flexibility of storing the process at the different places. Therefore, we can load the pages of process P5 in the place of P2 and P4.



Paging

Memory Management Unit

The purpose of Memory Management Unit (MMU) is to convert the logical address into the physical address. The logical address is the address generated by the CPU for every page while the physical address is the actual address of the frame where each page will be stored.

When a page is to be accessed by the CPU by using the logical address, the operating system needs to obtain the physical address to access that page physically.

The logical address has two parts.

1. Page Number
2. Offset

Memory management unit of OS needs to convert the page number to the frame number.

Example

Considering the above image, let's say that the CPU demands 10th word of 4th page of process P3. Since the page number 4 of process P1 gets stored at frame number 9 therefore the 10th word of 9th frame will be returned as the physical address.

Physical Address Space

Physical address space in a system can be defined as the size of the main memory. It is really important to compare the process size with the physical address space. The process size must be less than the physical address space. Each location in the main memory is given an address known as physical address.

The physical address has two parts.

1. Frame Number
2. Offset

Logical Address Space

Logical address space can be defined as the size of the process. The size of the process should be less enough so that it can reside in the main memory. Each word in the process has an address known as logical address.

The logical address has two parts.

1. Page Number
2. Offset

Word

The Word is the smallest unit of the memory. It is the collection of bytes. Every operating system defines different word sizes after analyzing the n-bit address that is inputted to the decoder and the 2 ^ n memory locations that are produced from the decoder.

Page Table in OS

Page Table is a data structure used by the virtual memory system to store the mapping between logical addresses and physical addresses.

Logical addresses are generated by the CPU for the pages of the processes therefore they are generally used by the processes.

Physical addresses are the actual frame address of the memory. They are generally used by the hardware or more specifically by RAM subsystems.

Physical Address Space = M words
Logical Address Space = L words
Page Size = P words

Physical Address = $\log_2 M$ = m bits
Logical Address = $\log_2 L$ = l bits
page offset = $\log_2 P$ = p bits

The CPU always accesses the processes through their logical addresses. However, the main memory recognizes physical address only.

In this situation, a unit named as Memory Management Unit comes into the picture. It converts the page number of the logical address to the frame number of the physical address. The offset remains same in both the addresses.

To perform this task, Memory Management unit needs a special kind of mapping which is done by page table. The page table stores all the Frame numbers corresponding to the page numbers of the page table.

In other words, the page table maps the page number to its actual location (frame number) in the memory.

Mapping from page table to main memory

In operating systems, there is always a requirement of mapping from logical address to the physical address. However, this process involves various steps which are defined as follows.

1. Generation of logical address

CPU generates logical address for each page of the process. This contains two parts: page number and offset.

2. Scaling

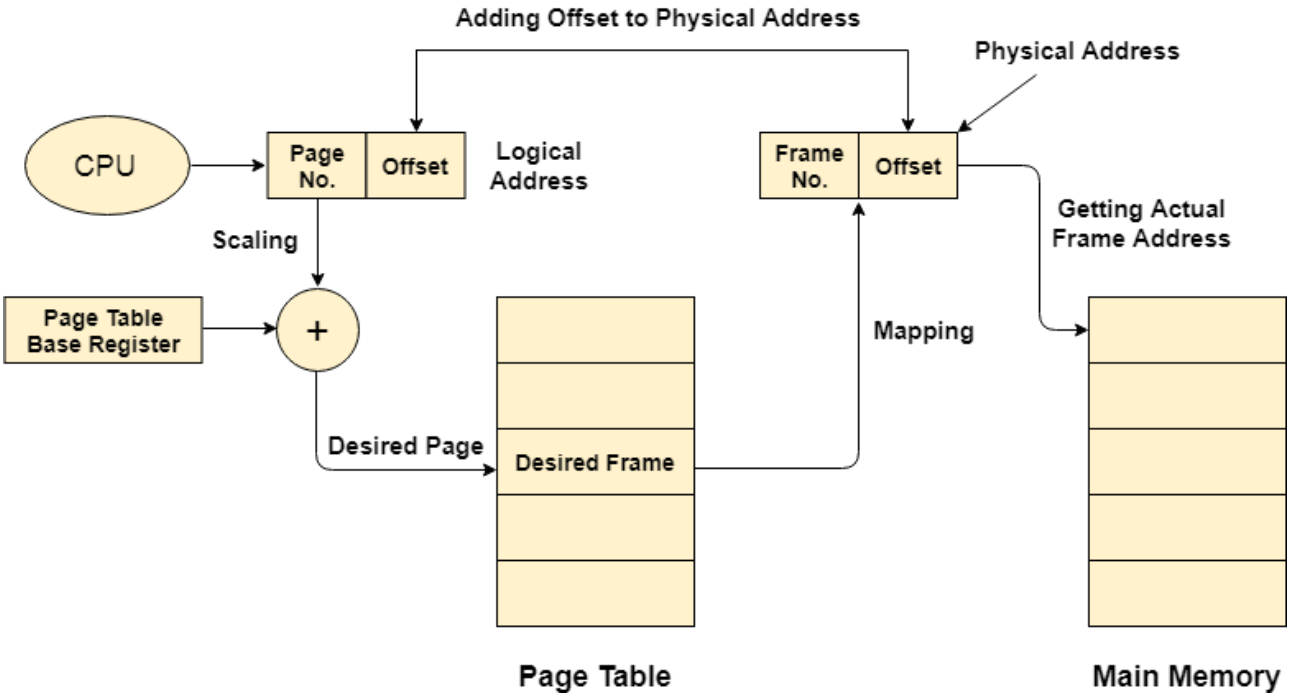
To determine the actual page number of the process, CPU stores the page table base in a special register. Each time the address is generated, the value of the page table base is added to the page number to get the actual location of the page entry in the table. This process is called scaling.

3. Generation of physical Address

The frame number of the desired page is determined by its entry in the page table. A physical address is generated which also contains two parts : frame number and offset. The Offset will be similar to the offset of the logical address therefore it will be copied from the logical address.

4. Getting Actual Frame Number

The frame number and the offset from the physical address is mapped to the main memory in order to get the actual word address.



Virtual Memory in OS

Virtual Memory is a storage scheme that provides user an illusion of having a very big main memory. This is done by treating a part of secondary memory as the main memory. In paging, when some pages of a process are loaded into main memory and some pages are left on secondary memory then secondary memory holding the remaining pages is known as Virtual memory as OS treat this part of secondary memory as main memory.

In this scheme, User can load the bigger size processes than the available main memory by having the illusion that the memory is available to load the process.
Instead of loading one big process in the main memory, the Operating System loads the different parts of more than one process in the main memory.
By doing this, the degree of multiprogramming will be increased and therefore, the CPU utilization will also be increased.

Advantages of Virtual Memory

1. The degree of Multiprogramming will be increased.
2. User can run large application with less real RAM.
3. There is no need to buy more memory RAMs.

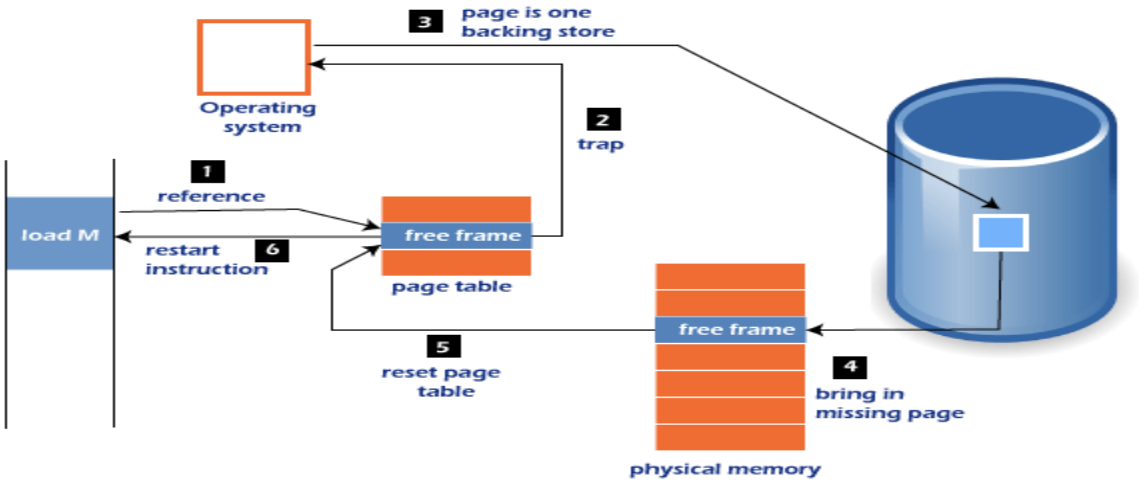
Disadvantages of Virtual Memory

1. The system becomes slower since swapping takes time.
2. It takes more time in switching between applications.
3. The user will have the lesser hard disk space for its use.

Page Fault in OS

When a process references some data, the page containing the data is searched in RAM. If found in RAM, it is called **page hit.** If the referenced page is not found in RAM (Page Table), it is called a **page miss.** When a **page miss** occurs, a signal issued to OS known as **page fault.** Page fault is a type of error that occurs when a program tries to access data that is not currently in the main memory or random-access memory (RAM). When this happens, the operating system (OS) tries to retrieve the required data from the hard disk (Virtual Memory) and this process is known as a page fault.

The fault specifies the operating system to trace all data into virtual memory and then relocate it from virtual memory i.e. secondary memory such as a hard disk, to the primary memory RAM.



Page Replacement Algorithms

When a page fault occurs, the OS will load the missing page from Virtual Memory to main memory. The page will be placed in a free Frame. But if there is no frame available for the page to be load, the OS will move page(s) from main memory to virtual memory so that frame become available for the page to be loaded. This process is called page replacement. There are three types of Page Replacement Algorithms. They are:

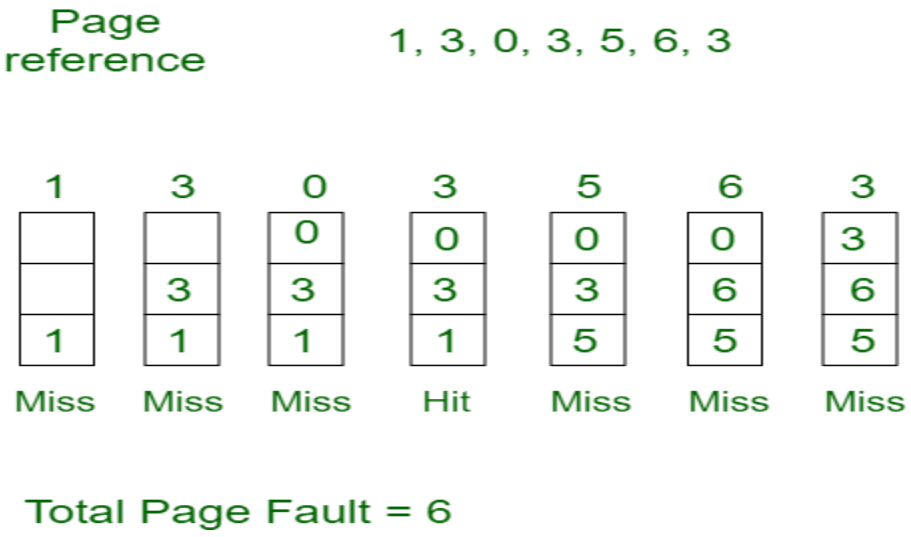
- First In First Out Page Replacement Algorithm
- Least Recently Used (LRU) Page Replacement Algorithm
- Most Recently Used (MRU) Page Replacement Algorithm

First in First out Page Replacement Algorithm

This is the first basic algorithm of Page Replacement Algorithms. When Page Fault occurs and there is no free Frame this problem arises, then the First In First Out Page Replacement Algorithm comes into picture.

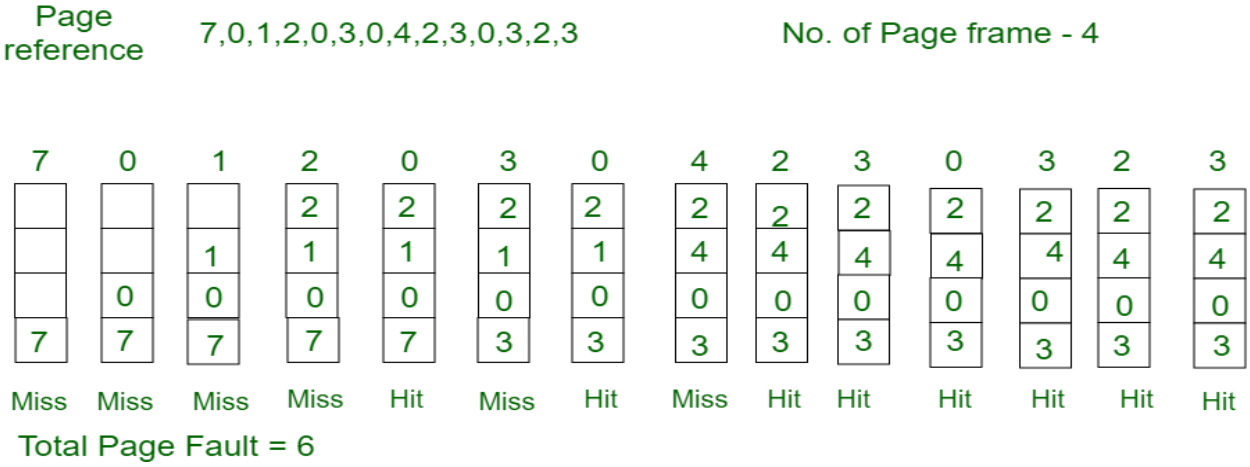
The First In First Out (FIFO) Page Replacement Algorithm removes the Page in the frame which is allotted long back (First Loaded Page). This means the useless page which is in the frame for a longer time is removed and the new page which is in the ready queue and is ready to occupy the frame is allowed by the First In First Out Page Replacement.

Example: Consider page reference string 1, 3, 0, 3, 5, 6, 3 with 3 page frames. Find the number of page faults.



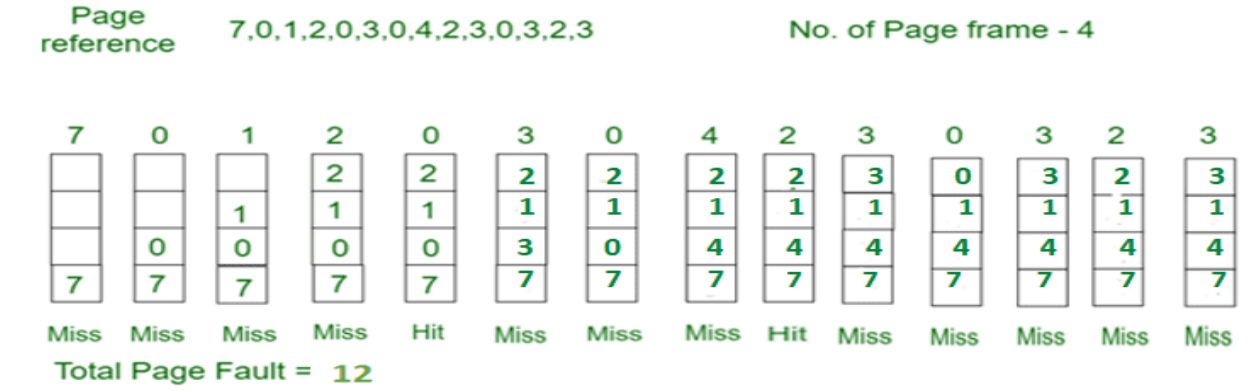
Least Recently Used (LRU) Page Replacement Algorithm

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.



Most Recently Used (LRU) Page Replacement Algorithm

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

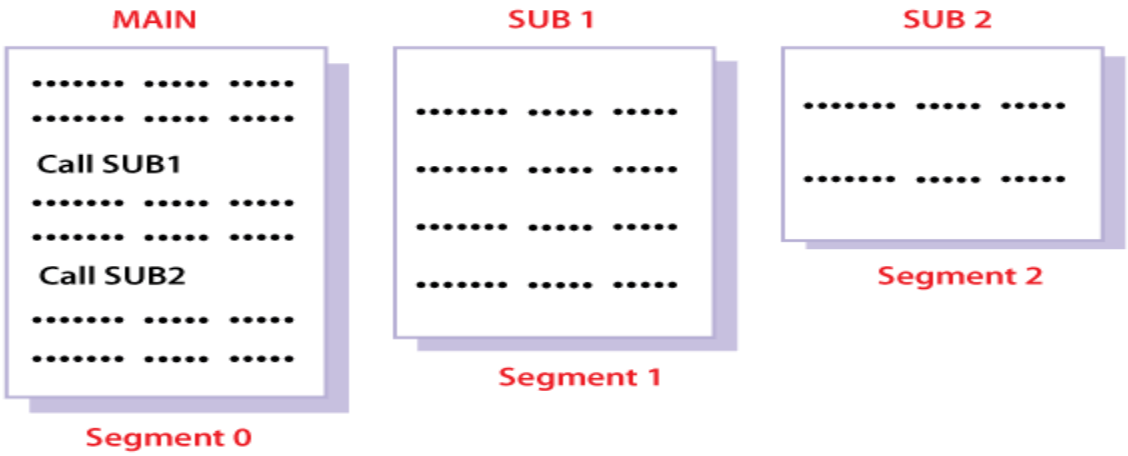


Optimal Page replacement Algorithm

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future. Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests.

Segmentation in OS

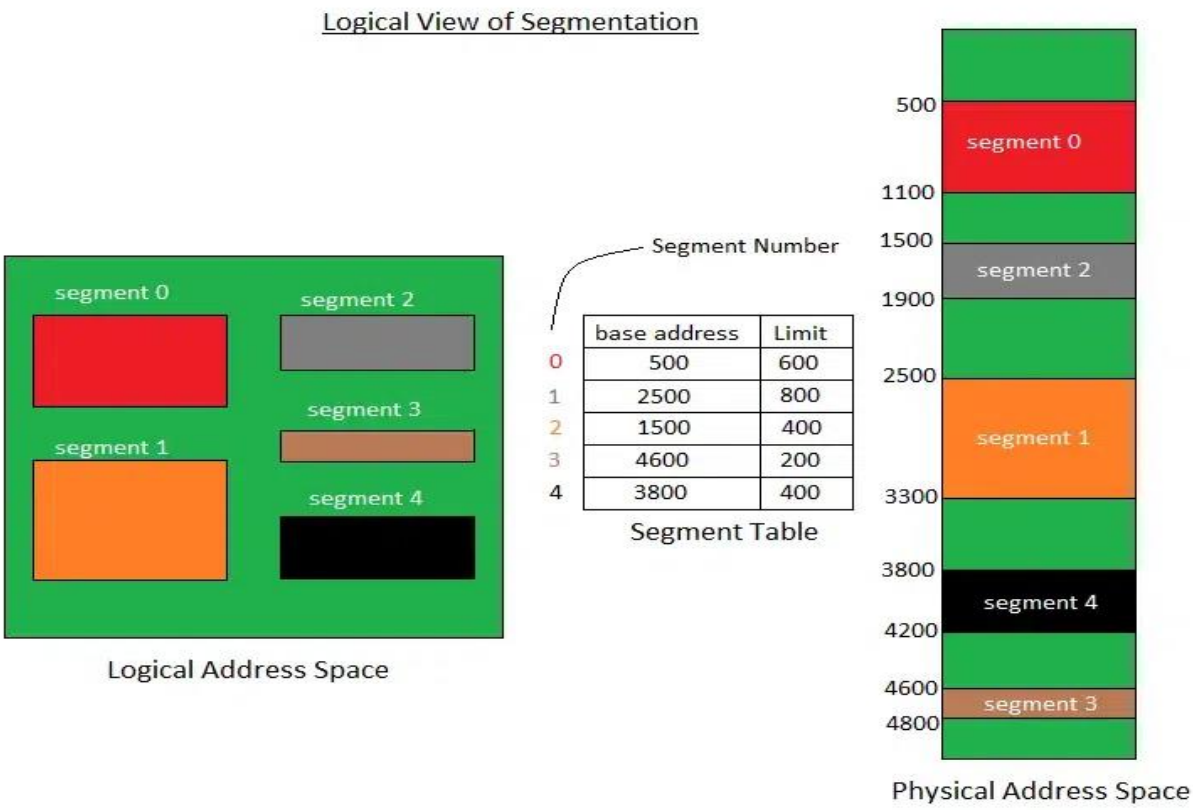
In Operating Systems, Segmentation is a memory management technique in which the memory is divided into the variable size parts. Each part is known as a segment which can be allocated to a process. Segmentation gives the user's view of the process which paging does not provide. Here the user's view is mapped to physical memory. It is better to have segmentation which divides the process into the segments. Each segment contains the same type of functions such as the main function can be included in one segment and the library functions can be included in the other segment.



The details about each segment are stored in a table called a segment table. Segment table is stored in one (or many) of the segments.

Segment table contains mainly two information about segment:

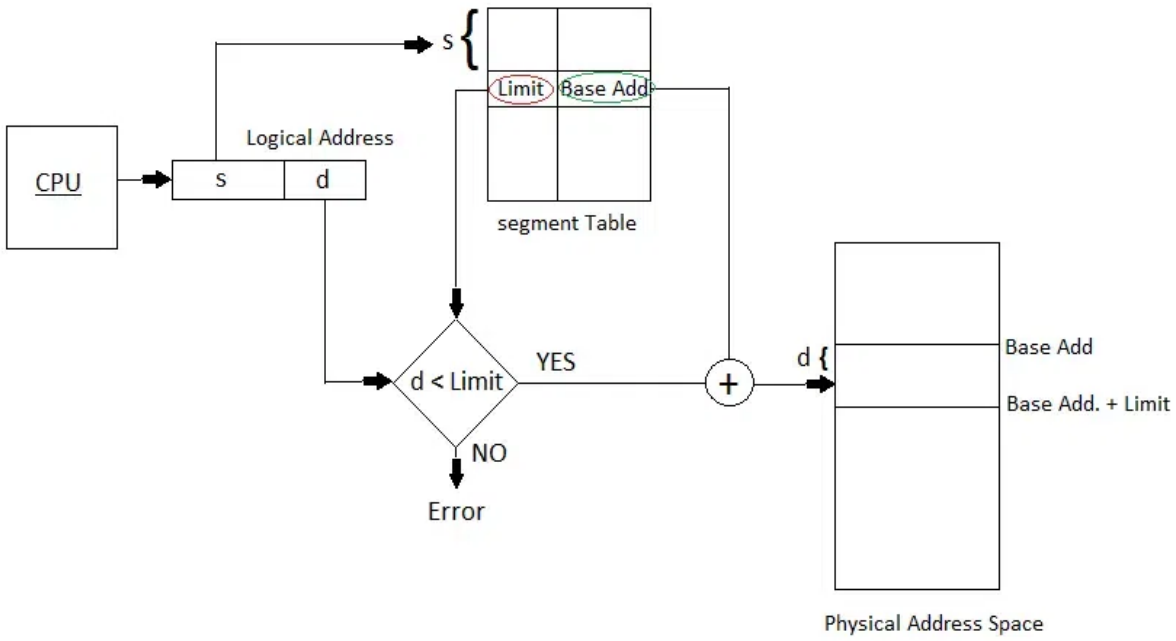
1. Base: It is the base address of the segment
2. Limit: It is the length of the segment.



With the help of segment map tables and hardware assistance, the operating system can easily translate a logical address into physical address on execution of a program.

The **Segment number** is mapped to the segment table. The limit of the respective segment is compared with the offset. If the offset is less than the limit then the address is valid otherwise it throws an error as the address is invalid.

In the case of valid addresses, the base address of the segment is added to the offset to get the physical address of the actual word in the main memory.



The above figure shows how address translation is done in case of segmentation.

Advantages of Segmentation

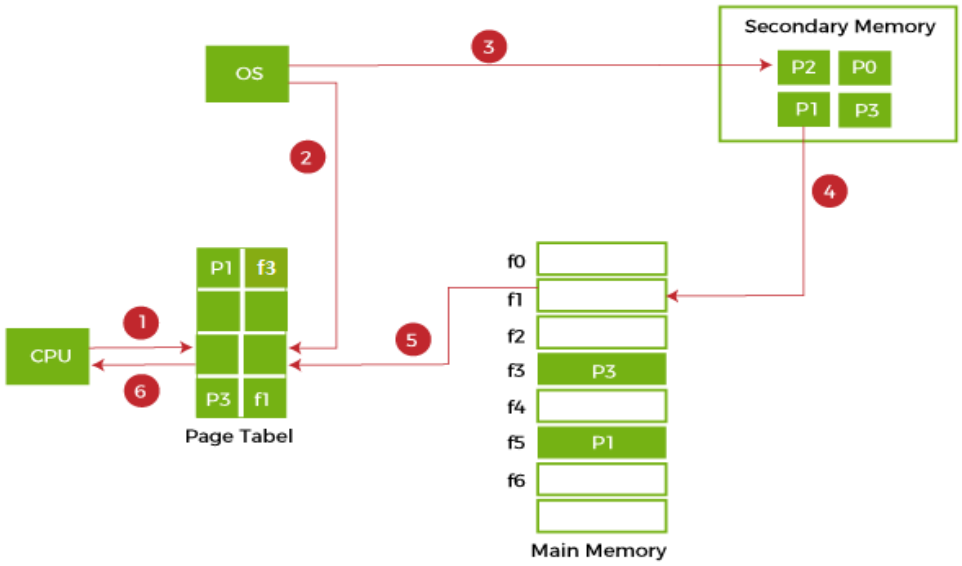
1. No internal fragmentation
2. Average Segment Size is larger than the actual page size.
3. Less overhead
4. It is easier to relocate segments than entire address space.
5. The segment table is of lesser size as compared to the page table in paging.

Disadvantages

1. It can have external fragmentation.
2. it is difficult to allocate contiguous memory to variable sized partition.
3. Costly memory management algorithms.

Demand Paging in OS

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. In demand paging, the operating system loads only the necessary pages of a program into memory at runtime, instead of loading the entire program into memory at the start. A page fault occurred when the program needed to access a page that is not currently in memory. The operating system then loads the required pages from the disk into memory and updates the page tables accordingly. This process is transparent to the running program and it continues to run as if the page had always been in memory.

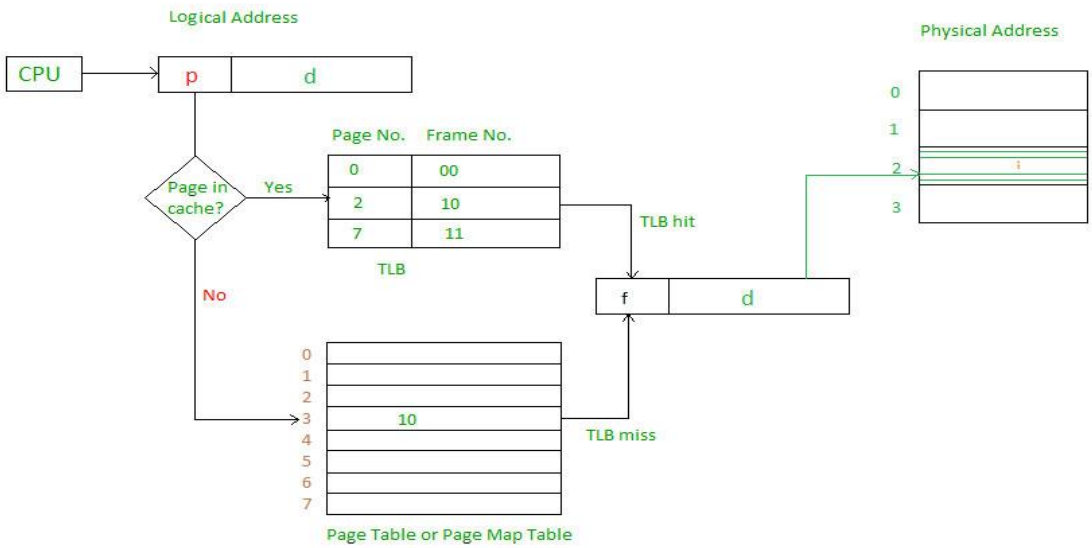


Translation look aside buffer (TLB)

A Translation Lookaside Buffer can be defined as a memory cache which can be used to reduce the time taken to access the page table again and again.

It is a memory cache which is closer to the CPU and the time taken by CPU to access TLB is lesser then that taken to access main memory.

In other words, we can say that TLB is faster and smaller than the main memory but cheaper and bigger than the register.



Steps in TLB hit

1. CPU generates a virtual (logical) address.
2. It is checked in TLB (present).
3. The corresponding frame number is retrieved, which now tells where the main memory page lies.

Steps in TLB miss

1. CPU generates a virtual (logical) address.
2. It is checked in TLB (not present).
3. Now the page number is matched to the page table residing in the main memory (assuming the page table contains all PTE).
4. The corresponding frame number is retrieved, which now tells where the main memory page lies.
5. The TLB is updated with new PTE (if space is not there, one of the replacement techniques comes into the picture i.e. either FIFO, LRU or MFU etc.

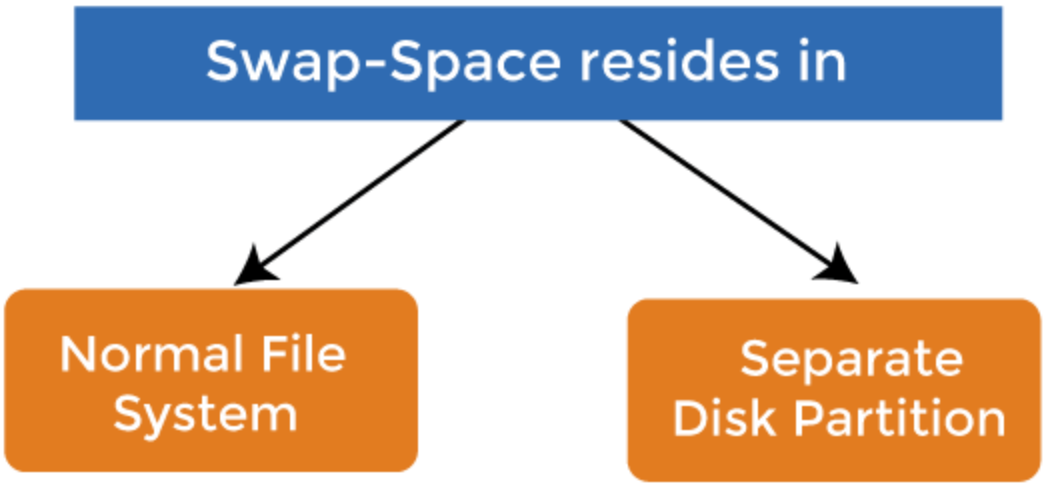
Swap-Space Management in Operating System

A computer has a sufficient amount of physical memory, but we need more, so we swap some memory on disk most of the time. ***Swap space*** is a space on a hard disk that is a substitute for physical memory. It is used as virtual memory, which contains process memory images. Whenever our computer runs short of physical memory, it uses its virtual memory and stores information in memory on a disk.

Virtual memory is a combination of RAM and disk space that running processes can use. **Swap space** is the **portion of virtual memory** on the hard disk, used when RAM is full.

This interchange of data between virtual memory and real memory is called ***swapping*** and space on disk as swap space. Swap space helps the computer's operating system pretend that it has more RAM than it actually has. It is also called a ***swap file***.

Swap-Space Management is a memory management technique used in multi-programming to increase the number of processes sharing the CPU. Swap-space management is another low-level task of the operating system. Virtual memory uses disk space as an extension of main memory. It is a technique of removing a process from the main memory and storing it into secondary memory, and then bringing it back into the main memory for continued execution. This action of moving a process out from main memory to secondary memory is called **Swap Out** and the action of moving a process out from secondary memory to main memory is called **Swap In**.



Thrashing is a condition or a situation when the system is spending a major portion of its time servicing the page faults i.e. **swapping**, but the actual processing done is very negligible. Thrashing in OS is a phenomenon that occurs in operating systems when a system spends a significant amount of time in paging rather than in executing basic application instructions. It is characterized by the constant swapping of data between the main and virtual memory, significantly reducing system performance.

Causes of thrashing:

1. High degree of multiprogramming.
2. Lack of frames.
3. Page replacement policy.