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.

Conversion of Logical Address into Physical Address

The CPU always generates a Logical Address. But, the Physical Address is needed to access the main memory.

The Logical Address generated by the CPU has two parts:

1. Page Number(p) - It is the number of bits required to represent the pages in the Logical Address Space. It is used as an index in a page table that contains the base address of a page in the physical memory.
2. Page Offset(d) - It denotes the page size or the number of bits required to represent a word on a page. It is combined with the Page Number to get the Physical Address.

The Physical Address also consists of two parts:

1. Frame Number(f) - It is the number of bits required to represent a frame in the Physical Address Space. It is the location of the required page inside the Main Memory.
2. Frame Offset(d) - It is the page size or the number of bits required to represent a word in a frame. It is equal to the Page Offset.

Page Table

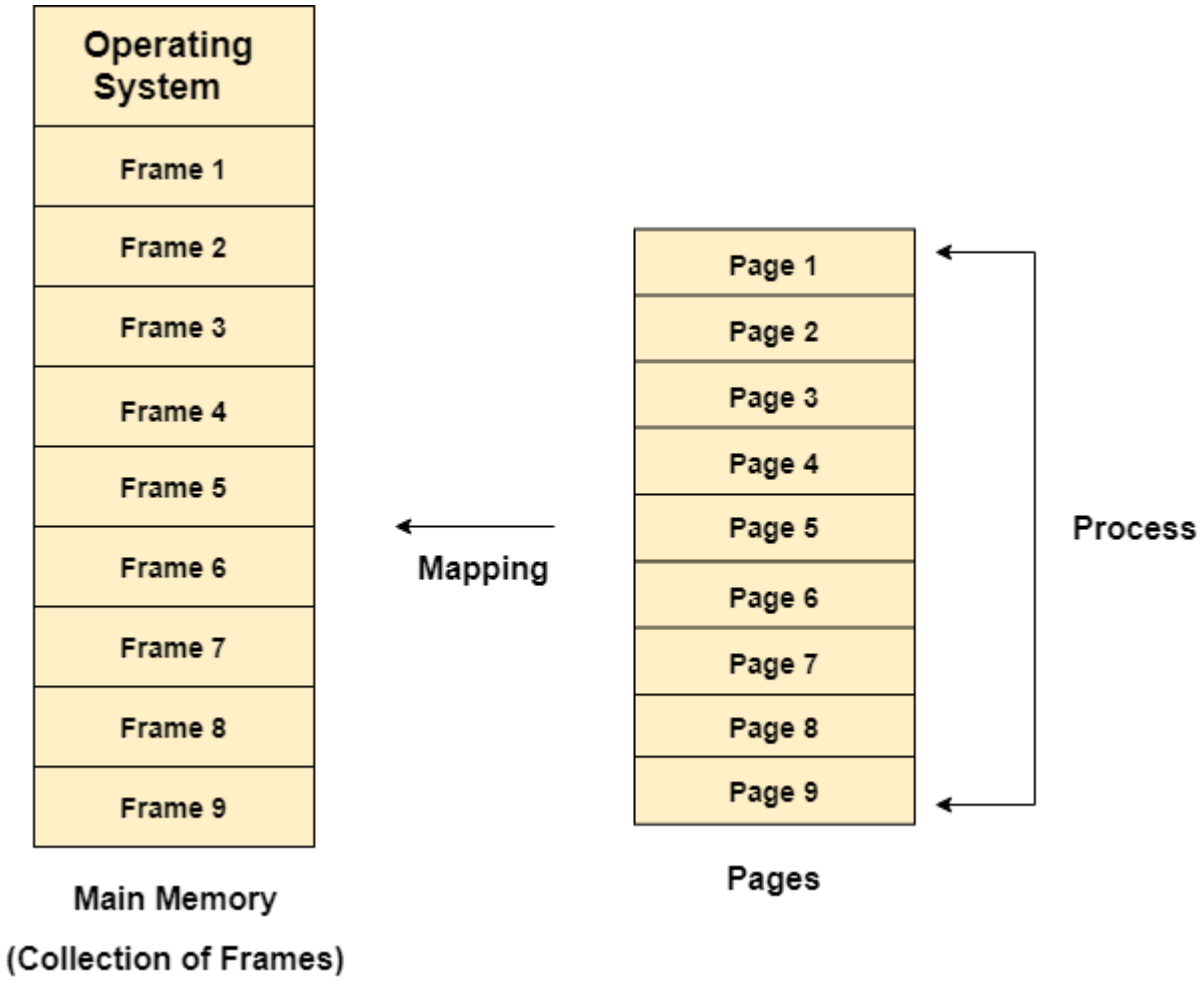
The Page Table contains the base address of each page inside the Physical Memory. It is then combined with Page Offset to get the actual address of the required data in the main memory.

The Page Number is used as the index of the Page Table which contains the base address which is the Frame Number. Page offset is then used to retrieve the required data from the main memory.

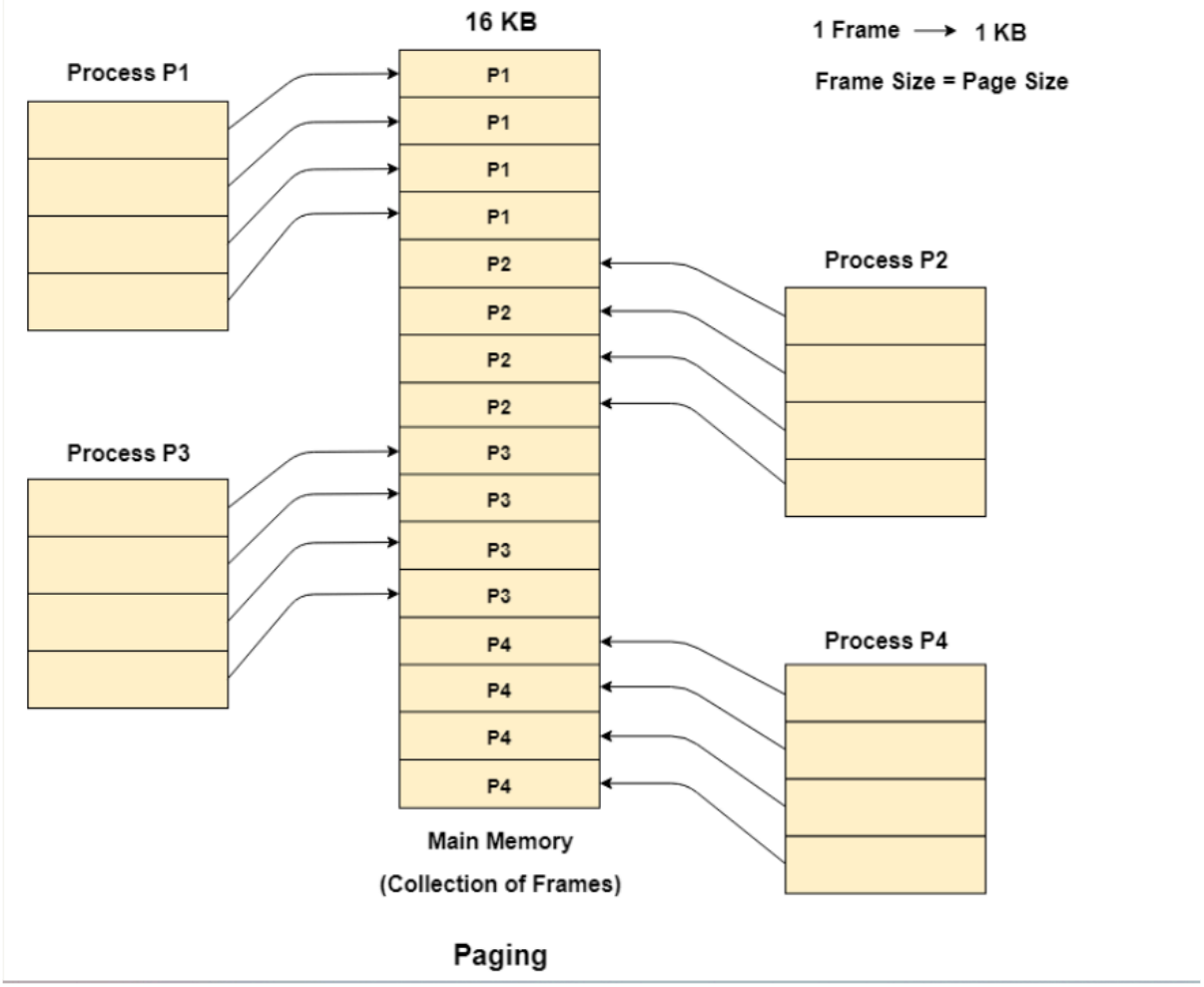
One page of the process is to be stored in one of the frames of the memory. The pages can be stored at the different locations of the memory but the priority is always to find the contiguous frames or holes.

Paging is a Memory Management technique that helps in retrieving the processes from the secondary memory in the form of pages. It eliminates the need for contagious allocation of memory to the processes. In paging, processes are divided into equal parts called pages, and main memory is also divided into equal parts and each part is called a frame.

Each page gets stored in one of the frames of the main memory whenever required. So, the size of a frame is equal to the size of a page. Pages of a process can be stored in the non-contagious locations in the main memory.


Operating System
Frame 1
Frame 2
Frame 3
Frame 4
Frame 5
Frame 6
Frame 7
Frame 8
Frame 9
Main Memory
(Collection of Frames)
Mapping
Page 1
Page 2
Page 3
Page 4
Page 5
Page 6
Page 7
Page 8
Page 9
Pages
Process

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



Paging

How Does Paging Work?

Paging enables an OS to transfer data between secondary (virtual) and primary (physical) memory. Both storage types are divided into fixed-size blocks. The blocks in primary memory are **frames**, while those in secondary storage are **pages**.

Whenever a program executes, it splits into pages, and the OS automatically stores them in secondary memory.

When the process requests memory, the OS allocates page frames from the primary memory to the process. Next, the OS moves program pages from the secondary memory to the primary memory frames.

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)

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.

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

Conversion of Logical Address into Physical Address

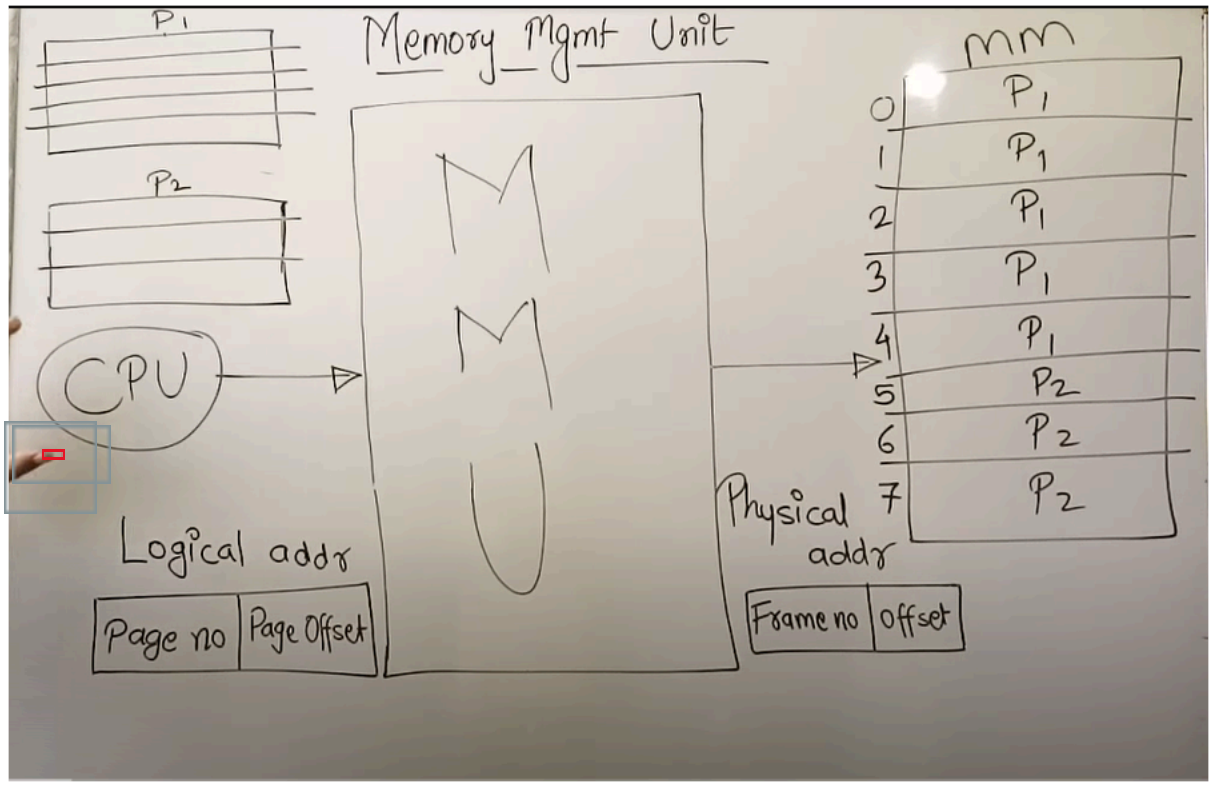
The CPU always generates a Logical Address. But, the Physical Address is needed to access the main memory.

The Logical Address generated by the CPU has two parts:

1. Page Number(p) - It is the number of bits required to represent the pages in the Logical Address Space. It is used as an index in a page table that contains the base address of a page in the physical memory.
2. Page Offset(d) - It denotes the page size or the number of bits required to represent a word on a page. It is combined with the Page Number to get the Physical Address.

The Physical Address also consists of two parts:

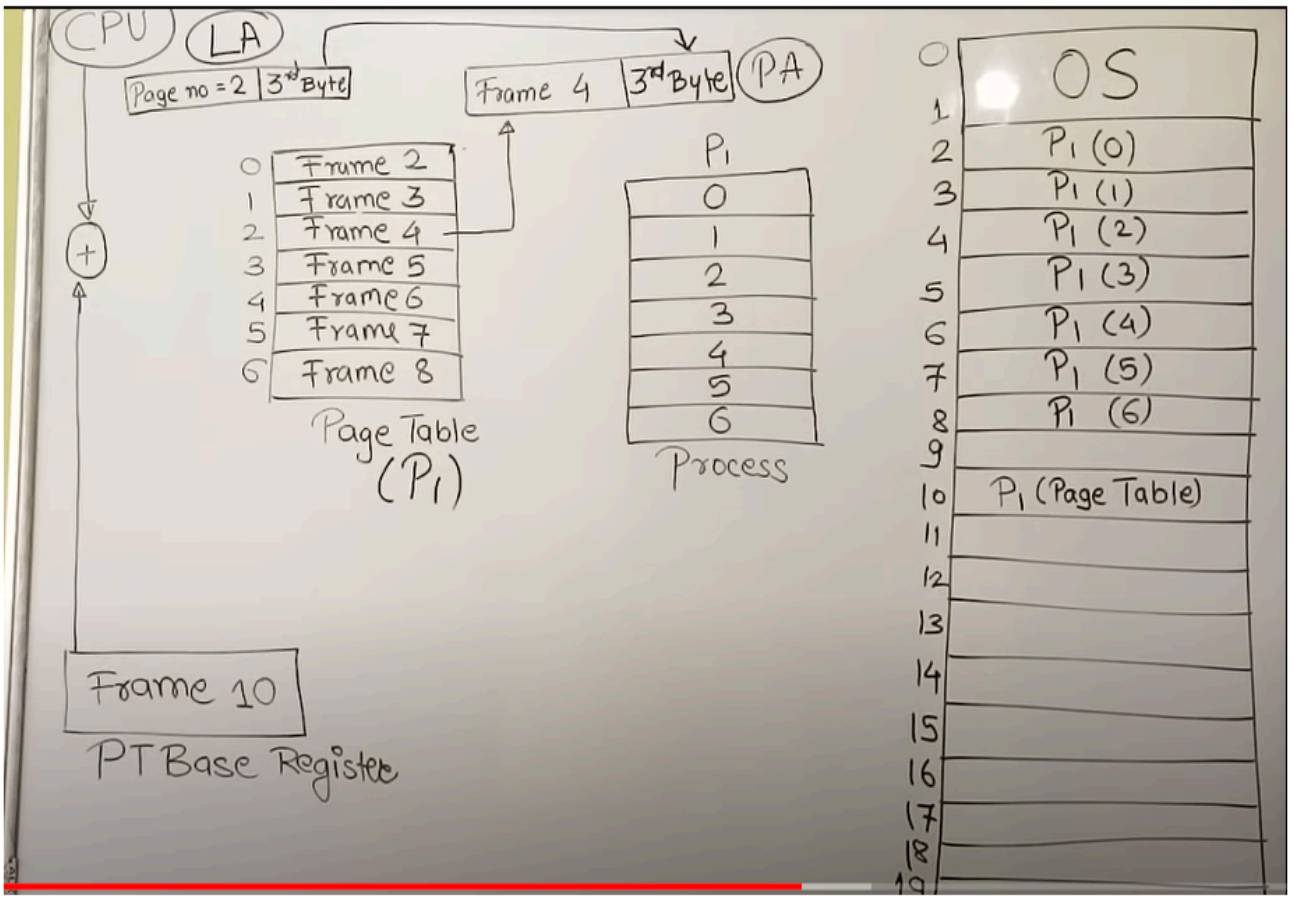
1. Frame Number(f) - It is the number of bits required to represent a frame in the Physical Address Space. It is the location of the required page inside the Main Memory.
2. Frame Offset(d) - It is the page size or the number of bits required to represent a word in a frame. It is equal to the Page Offset.



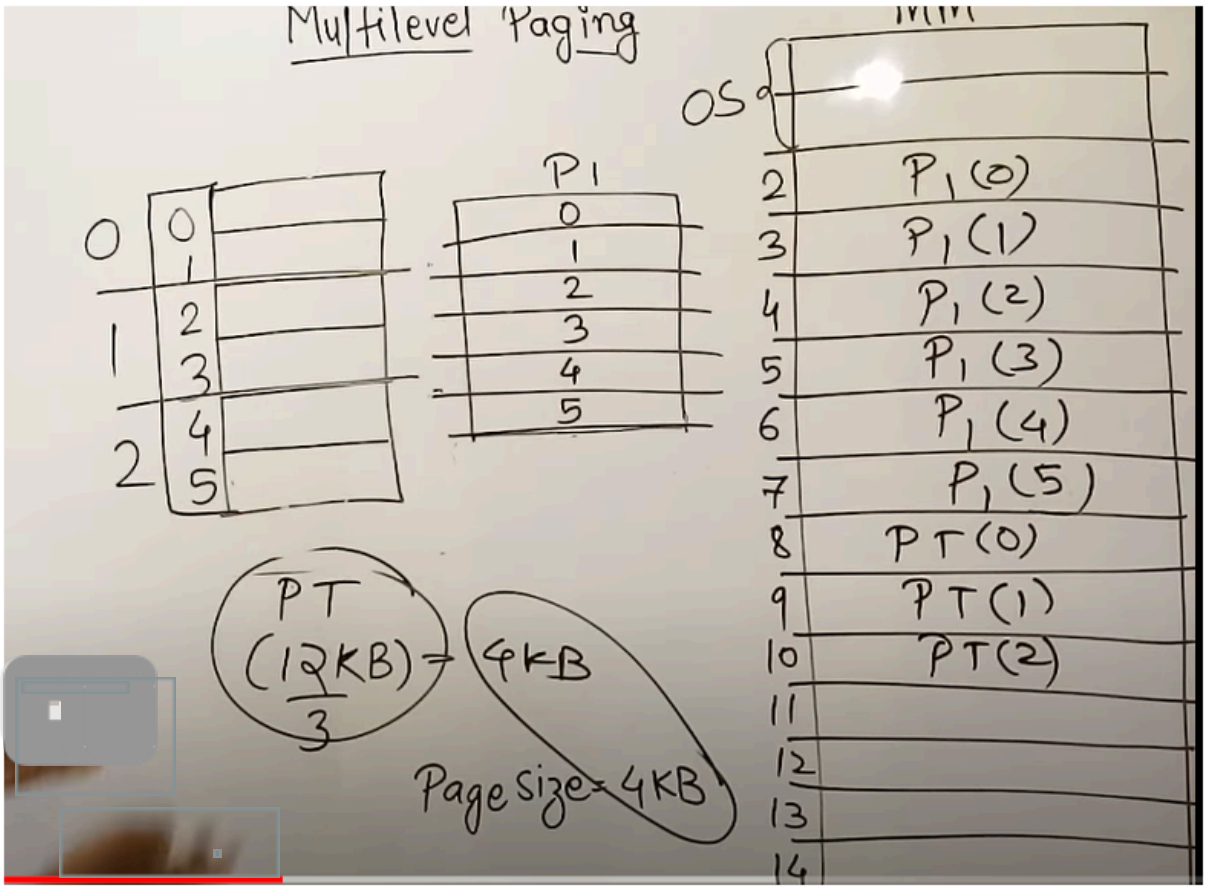
Page Table

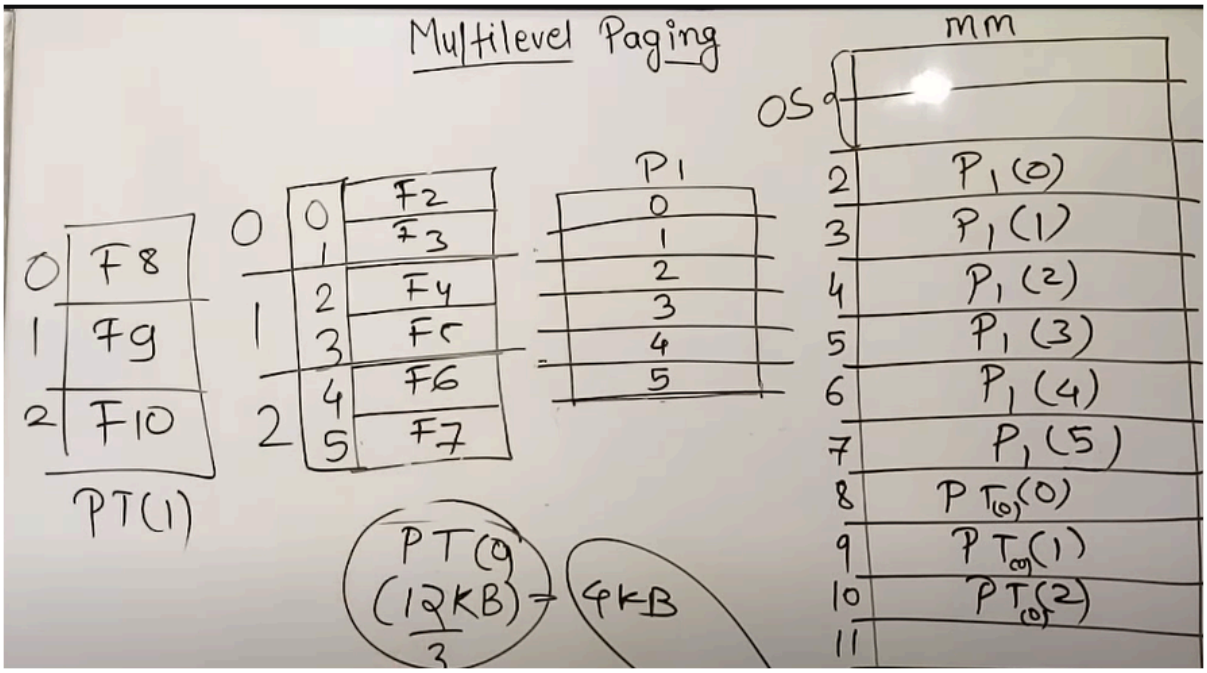
The Page Table contains the base address of each page inside the Physical Memory. It is then combined with Page Offset to get the actual address of the required data in the main memory.

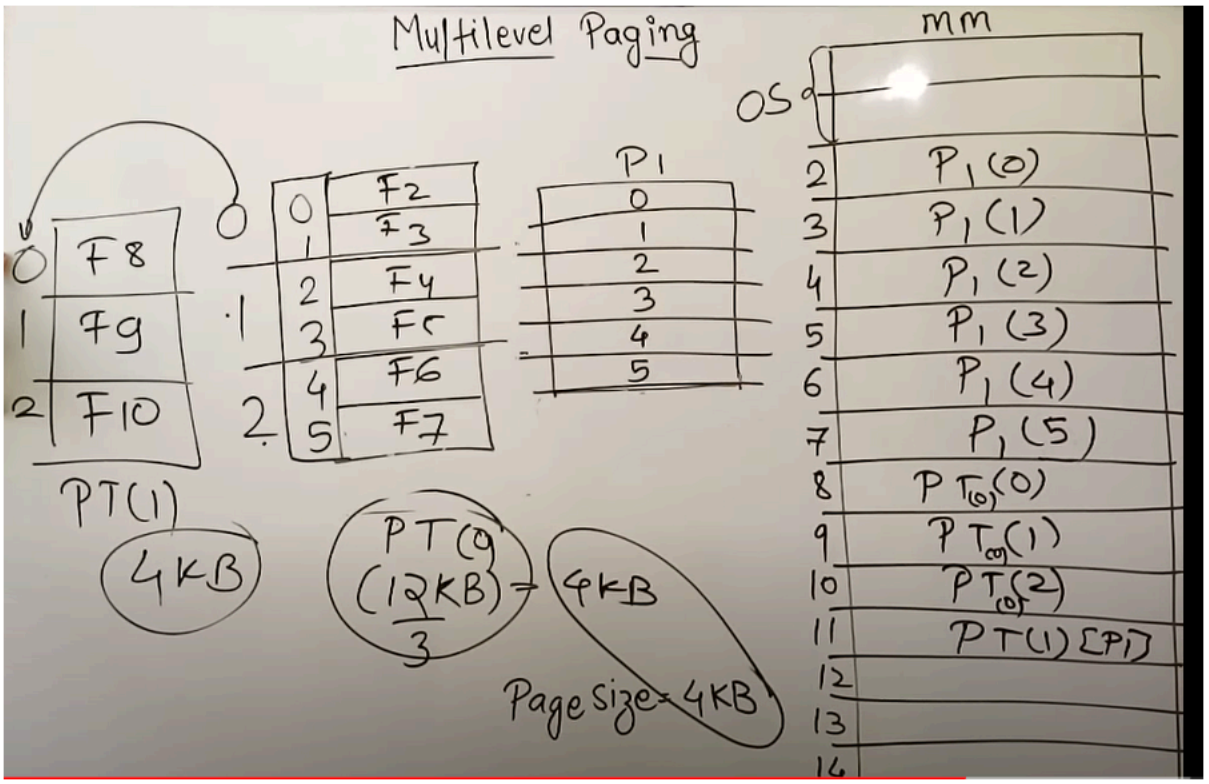
The Page Number is used as the index of the Page Table which contains the base address which is the Frame Number. Page offset is then used to retrieve the required data from the main memory.


CPU
LA
Page no = 2
3rd Byte
Frame 4
3rd Byte
PA
0 Frame 2
1 Frame 3
2 Frame 4
3 Frame 5
4 Frame 6
5 Frame 7
6 Frame 8
Page Table
(P1)
P1
0
1
2
3
4
5
6
Process
+
Frame 10
PT Base Register
0 OS
1
2 P1 (0)
3 P1 (1)
4 P1 (2)
5 P1 (3)
6 P1 (4)
7 P1 (5)
8 P1 (6)
9
10 P1 (Page Table)
11
12
13
14
15
16
17
18
19

Main Memory


Multilevel Paging
MM
OS
P1
0
1
2
3
4
5
P1(0)
P1(1)
P1(2)
P1(3)
P1(4)
P1(5)
PT(0)
PT(1)
PT(2)
PT (12KB) / 3
4KB
Page size = 4KB


Multilevel Paging
MM
OS
P1
0
1
2
3
4
5
P1(0)
P1(1)
P1(2)
P1(3)
P1(4)
P1(5)
PT0(0)
PT0(1)
PT0(2)
F2
F3
F4
F5
F6
F7
F8
F9
F10
PT(1)
PT(0) (12KB) / 3
4KB



Advantages of Paging

- It is one of the easiest Memory Management Algorithms.
- Paging helps in storing a process at non-contagious locations in the main memory.
- Paging removes the problem of External Fragmentation.

Disadvantages of Paging

- It may cause Internal Fragmentation.
- More Memory is consumed by the Page Tables.
- It faces a longer memory lookup time.

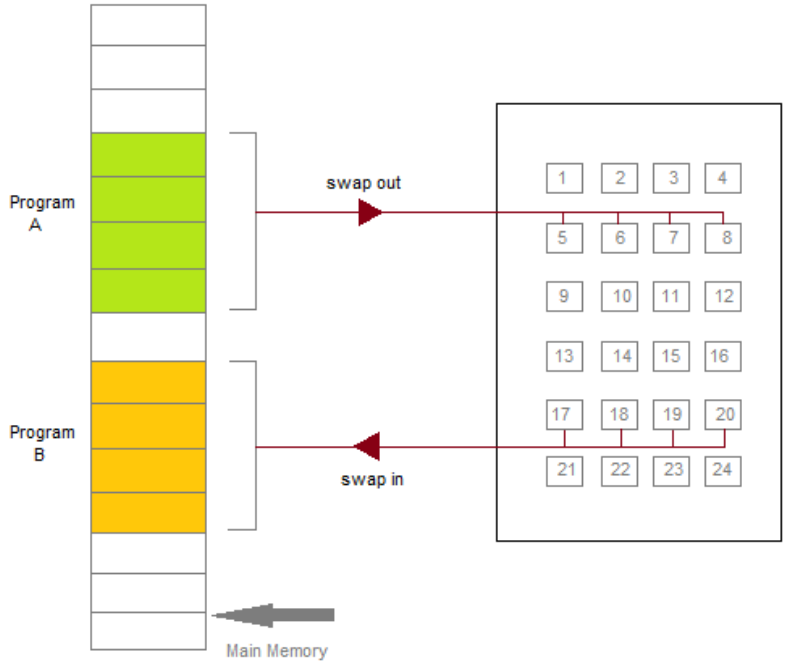
What is Demand Paging in OS (Operating System)?

Demand paging is a memory management technique where the operating system loads only the required pages into memory when they are needed. the entire program is not loaded into memory at once. Instead, only those parts of the program that are required at a particular point in time are loaded into memory. This approach is very efficient in terms of memory usage since it does not require the entire program to be loaded into memory at once. However, there is a small delay in accessing pages that have not been loaded into memory, which can lead to slower performance.

According to the concept of Virtual Memory, in order to execute some process, only a part of the process needs to be present in the main memory which means that only a few pages will only be present in the main memory at any time.

However, deciding, which pages need to be kept in the main memory and which need to be kept in the secondary memory, is going to be difficult because we cannot say in advance that a process will require a particular page at particular time.

Therefore, to overcome this problem, there is a concept called Demand Paging is introduced. It suggests keeping all pages of the frames in the secondary memory until they are required. In other words, it says that do not load any page in the main memory until it is required.



The Benefits of Demand Paging in OS for Operating Systems

Demand paging is a memory management technique used by modern operating systems to **manage physical memory** more efficiently. It allows the operating system to load only the required parts of a program into memory at the time of execution, rather than loading the

entire program into memory at once. This helps to reduce the memory requirements of the system, thereby improving its overall performance and responsiveness.

One of the main benefits of demand paging is that it allows the operating system to use memory more efficiently. Rather than allocating memory for a program's entire data and code segments, demand paging allows the system to allocate memory only when needed, freeing up memory that can be used for other processes.

Another benefit of demand paging is that it allows multiple processes to run simultaneously on a single system without having to worry about running out of memory. The operating system can simply swap out the pages that are not being used by a process and bring in the pages that are required for the currently running process, allowing multiple processes to run simultaneously without running out of memory.

Demand paging also helps to reduce the time it takes to load a program into memory. Instead of loading the entire program into memory at once, the operating system can load only the required pages on demand, reducing the amount of time it takes to load the program and making the system more responsive.

Page Replacement Algorithms in Operating Systems

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 there are no free page frames in memory. However, another page fault would arise if the replaced page is referenced again. Hence it is important to replace a page that is not likely to be referenced in the immediate future. y. 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.
Page Fault: A page fault happens when a running program accesses a memory page that is mapped into the virtual address space but not loaded in physical memory. Since actual physical memory is much smaller than virtual memory, page faults happen. In case of a page fault, Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.

When the program tries to access a page that is not currently in memory, a page fault is triggered, and the operating system is responsible for handling the fault.

There are various page fault terminologies in the operating system. Some terminologies of page fault are as follows

1. Page Hit

When the CPU attempts to obtain a needed page from main memory and the page exists in **main memory (RAM)**, it is referred to as a **"PAGE HIT"**.

2. Page Miss

If the needed page has not existed in the **main memory (RAM)**, it is known as **"PAGE MISS"**.

Page Replacement Algorithms:
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 of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

- This algorithm is implemented by keeping the track of all the pages in the queue.
- As new pages are requested and are swapped in, they are added to the tail of a queue and the page which is at the head becomes the victim.
- This is not an effective way of page replacement but it can be used for small systems.

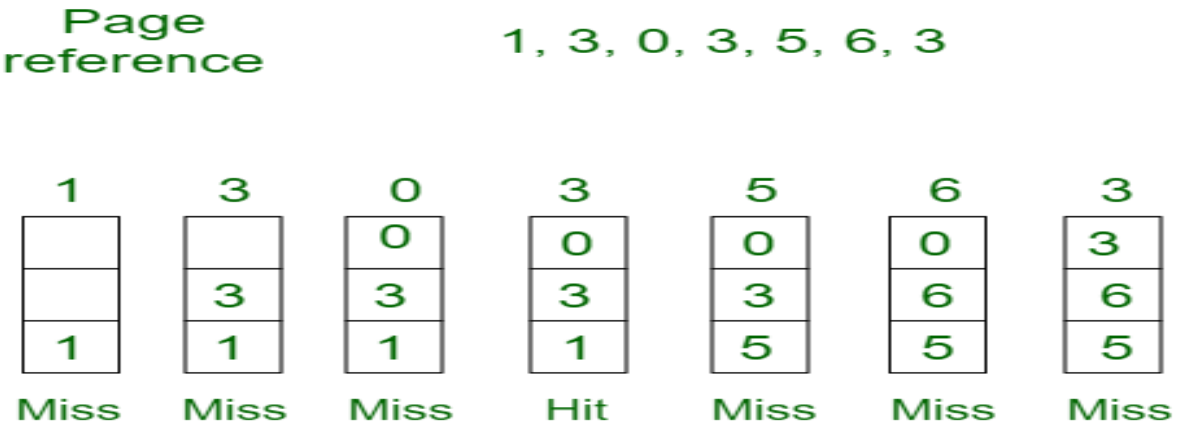
Advantages

- This algorithm is simple and easy to use.
- FIFO does not cause more overhead.

Disadvantages

- This algorithm does not make the use of the frequency of **last used time rather** it just replaces the Oldest Page.
- There is an increase in **page faults** as page frames increases.
- The performance of this algorithm is the worst.

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



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.**

Q2. Consider a reference string: 4, 7, 6, 1, 7, 6, 1, 2, 7, 2. the number of frames in the memory is 3. Find out the number of page faults respective to:

Request	**4**	**7**	**6**	**1**	**7**	**6**	**1**	**2**	**7**	**2**
Frame 3			6	6	6	6	6	6	7	7
Frame 2		7	7	7	7	7	7	2	2	2
Frame 1	4	4	4	1	1	1	1	1	1	1
Miss/Hit	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Miss	Miss	Hit

Number of Page Faults in FIFO = 6

Request	4	7	6	1	7	6	1	2	7	2
Frame 3			6	6	6	6	6	2	2	2
Frame 2		7	7	7	7	7	7	7	7	7
Frame 1	4	4	4	1	1	1	1	1	1	1
Miss/Hit	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Miss	Hit	Hit

2.LRU Page Replacement Algorithm in OS

This algorithm stands for "Least recent used" and this algorithm helps the Operating system to search those pages that are used over a short duration of time frame.

- The page that has not been used for the longest time in the main memory will be selected for replacement.
- This algorithm is easy to implement.
- This algorithm makes use of the counter along with the even-page.

Advantages of LRU

- It is an efficient technique.
- With this algorithm, it becomes easy to identify the faulty pages that are not needed for a long time.
- It helps in Full analysis.

Disadvantages of LRU

- It is expensive and has more complexity.
- There is a need for an additional data structure.

Example:

Example-1: 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.

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

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

Example-2: c

		6	6		6	6	6	6	7	7	
	7	7	7		7	7	7	2	2	2	
4	4	4	1		1	1	1	1	1	1	
f	f	f	f		h	h	h	f	f	h	

Total page fault =6

3.Optimal Page Replacement Algorithm

This algorithm mainly replaces the page that will not be used for the longest time in the future. The practical implementation of this algorithm is not possible.

- Practical implementation is not possible because we cannot predict in advance those pages that will not be used for the longest time in the future.
- This algorithm leads to less number of page faults and thus is the best-known algorithm

Also, this algorithm can be used to measure the performance of other algorithms.

Advantages of OPR

- This algorithm is easy to use.
- This algorithm provides excellent efficiency and is less complex.
- For the best result, the implementation of data structures is very easy

Disadvantages of OPR

- In this algorithm future awareness of the program is needed.
- Practical Implementation is not possible because the operating system is unable to track the future request

Example-2: Consider the page reference string 4,7, 6, 1, 7, 6, 1, 2, 7, 2 with 3-page frames. Find number of page faults.

		6	6	6	6	6	2	2	2			
	7	7	7	7	7	7	7	7	7			
4	4	4	1	1	1	1	1	1	1			
f	f	f	f	h	H	h	f	h	h			

Segmentation in OS (Operating System)

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.

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:

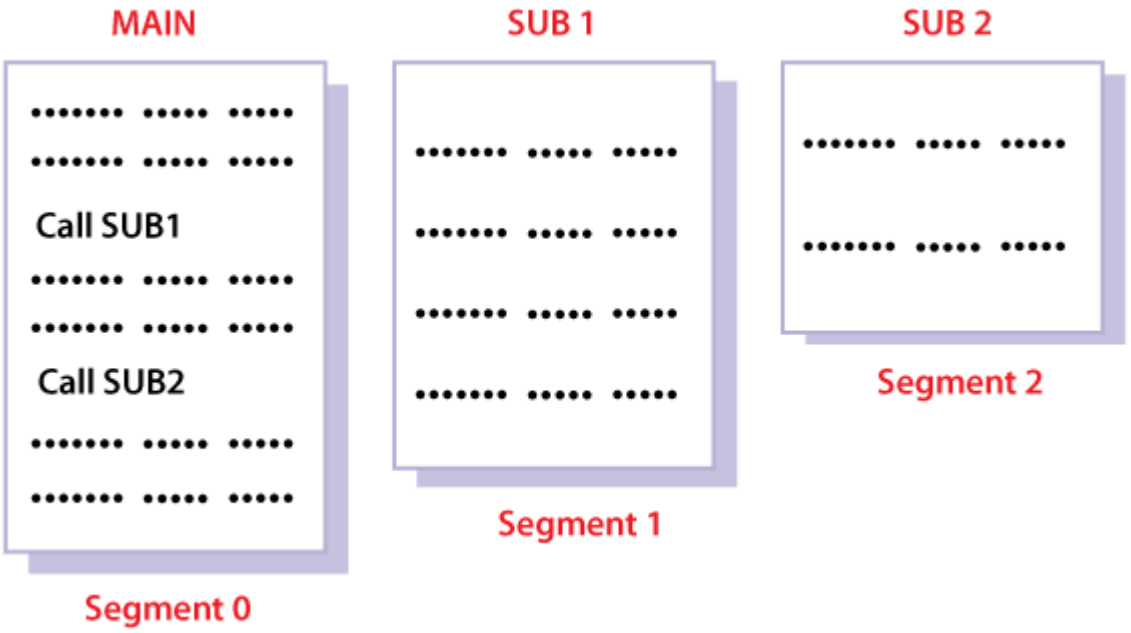
1. Base: It is the base address of the segment
2. Limit: It is the length of the segment.

Why Segmentation is required?

Till now, we were using Paging as our main memory management technique. Paging is more close to the Operating system rather than the User. It divides all the processes into the form of pages regardless of the fact that a process can have some relative parts of functions which need to be loaded in the same page.

Operating system doesn't care about the User's view of the process. It may divide the same function into different pages and those pages may or may not be loaded at the same time into the memory. It decreases the efficiency of the system.

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.



Translation of Logical address into physical address by segment table

CPU generates a logical address which contains two parts:

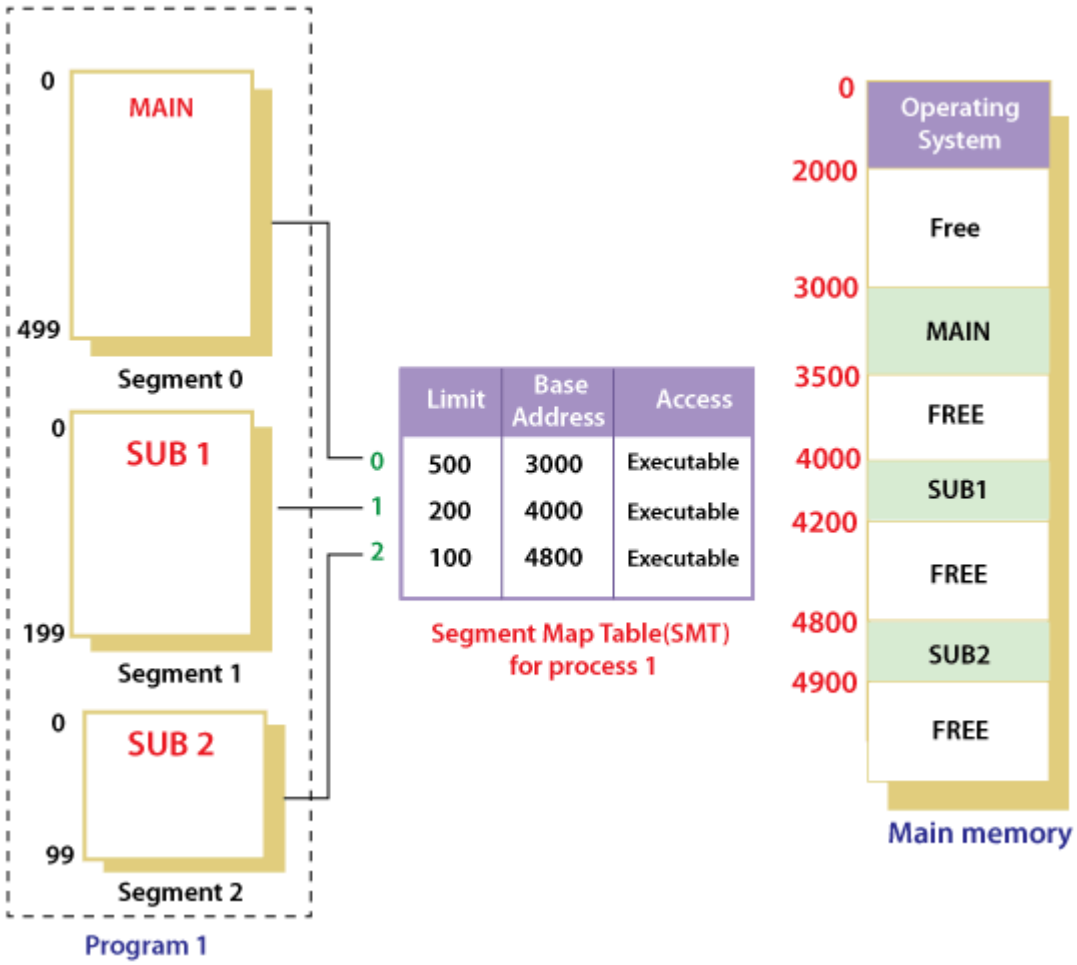
1. Segment Number
2. Offset

For Example:

Suppose a 16 bit address is used with 4 bits for the segment number and 12 bits for the segment offset so the maximum segment size is 4096 and the maximum number of segments that can be refereed is 16.

When a program is loaded into memory, the segmentation system tries to locate space that is large enough to hold the first segment of the process, space information is obtained from the free list maintained by memory manager. Then it tries to locate space for other segments. Once adequate space is located for all the segments, it loads them into their respective areas.

The operating system also generates a segment map table for each program.



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.