Paging with Example

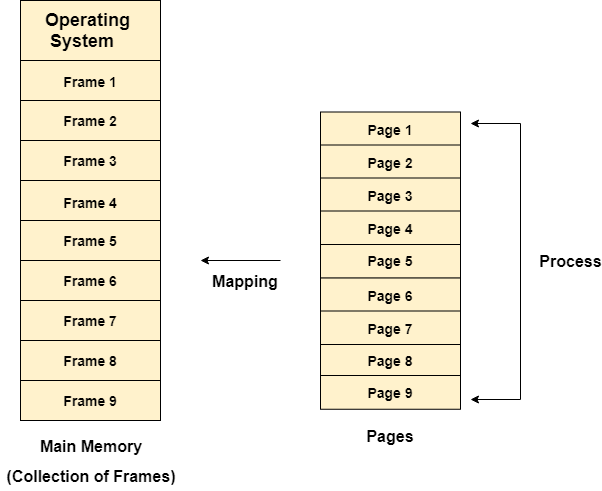
In Operating Systems, Paging is a storage mechanism used to retrieve processes from the secondary storage into the main memory in the form of pages.

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.



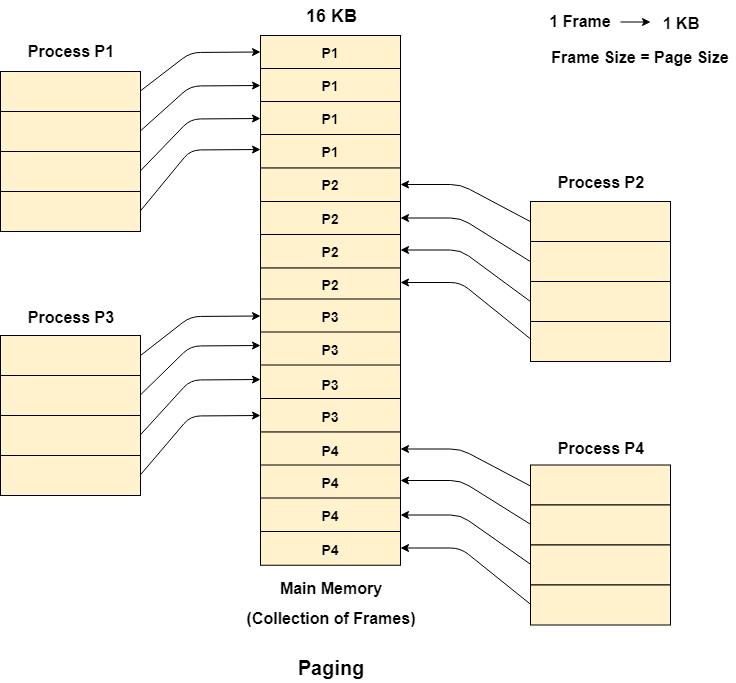
Example

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.

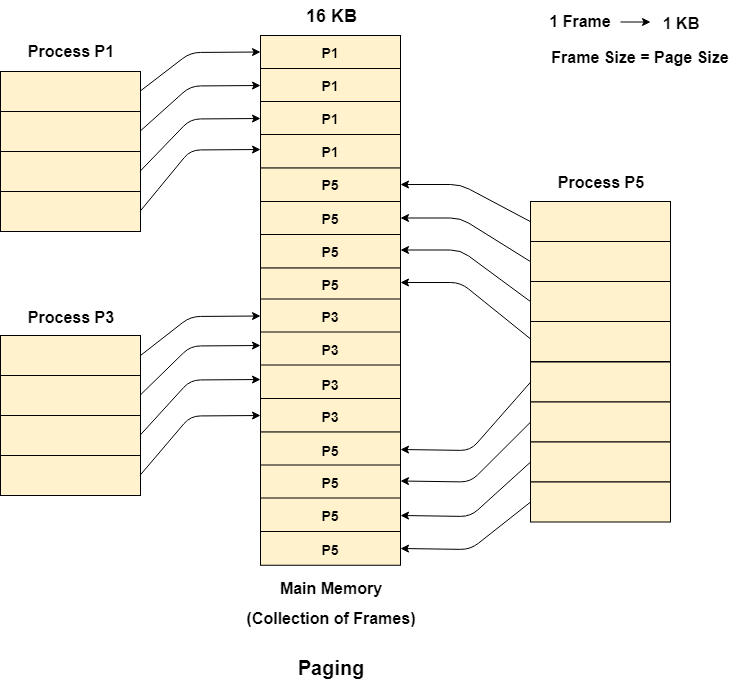
Initially, all the frames are empty therefore pages of the processes will get stored in the contiguous way.

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



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 non contiguous 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.



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 and Logical Address Space

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.

Physical Address Space = Size of the Main Memory  
  
If, physical address space = 64 KB = 2 ^ 6 KB = 2 ^ 6 X 2 ^ 10 Bytes = 2 ^ 16 bytes  
  
Let us consider,  
word size = 8 Bytes = 2 ^ 3 Bytes  
  
Hence,  
Physical address space (in words) = (2 ^ 16) / (2 ^ 3) = 2 ^ 13 Words  
  
Therefore,  
Physical Address = 13 bits  
  
In General,  
If, Physical Address Space = N Words  
  
then, Physical Address = log2 N

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.

**Let's say,**

Logical Address Space = 128 MB = (2 ^ 7 X 2 ^ 20) Bytes = 2 ^ 27 Bytes  
Word size = 4 Bytes = 2 ^ 2 Bytes  
  
Logical Address Space (in words) = (2 ^ 27) / (2 ^ 2) = 2 ^ 25 Words  
Logical Address = 25 Bits  
  
In general,  
If, logical address space = L words  
Then, Logical Address = Log2L bits

What is a 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.

[**next →**](https://www.javatpoint.com/os-mapping-from-page-table-to-main-memory)[**← prev**](https://www.javatpoint.com/os-physical-and-logical-address-space)

Page Table

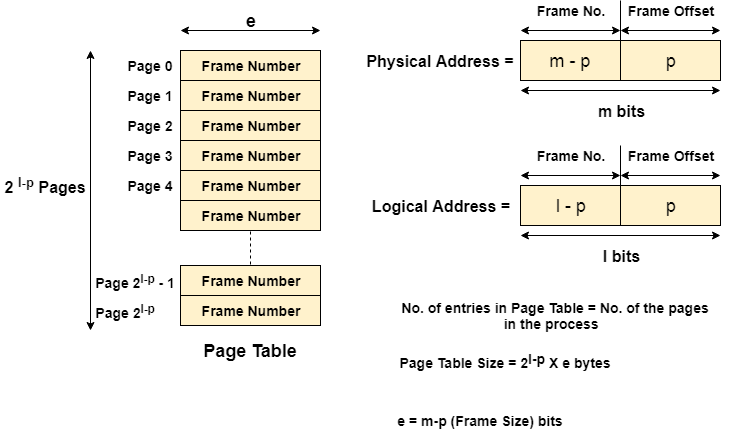
Page Table is a data structure used by the virtual memory system to store the mapping between logical addresses and physical addresses.

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.

The image given below considers,

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



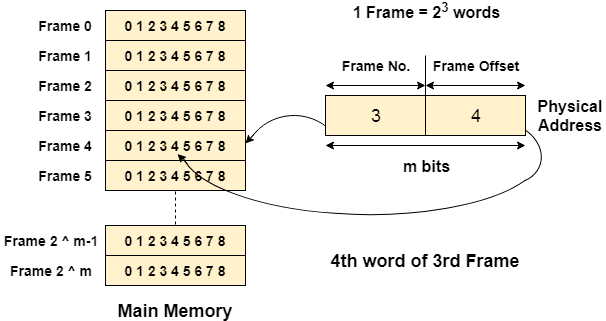
The CPU always accesses the processes through their logical addresses. However, the main memory recognizes physical address only.

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.

In the image given below shows, how the required word of the frame is accessed with the help of offset.



[**next →**](https://www.javatpoint.com/os-page-table-entry)[**← prev**](https://www.javatpoint.com/os-page-table)

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

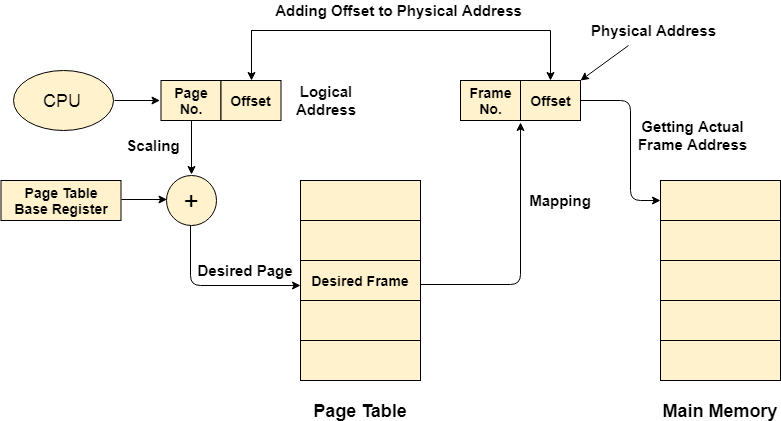
To determine the actual page number of the process, CPU stores the page table base in a special register. Each time the address is generated, the value of the page table base is added to the page number to get the actual location of the page entry in the table. This process is called scaling.

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.



Ne

# Virtual Memory

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 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.

## How Virtual Memory Works?

In modern word, virtual memory has become quite common these days. In this scheme, whenever some pages needs to be loaded in the main memory for the execution and the memory is not available for those many pages, then in that case, instead of stopping the pages from entering in the main memory, the OS search for the RAM area that are least used in the recent times or that are not referenced and copy that into the secondary memory to make the space for the new pages in the main memory.

Since all this procedure happens automatically, therefore it makes the computer feel like it is having the unlimited RAM.

## Demand Paging

Demand Paging is a popular method of virtual memory management. In demand paging, the pages of a process which are least used, get stored in the secondary memory.

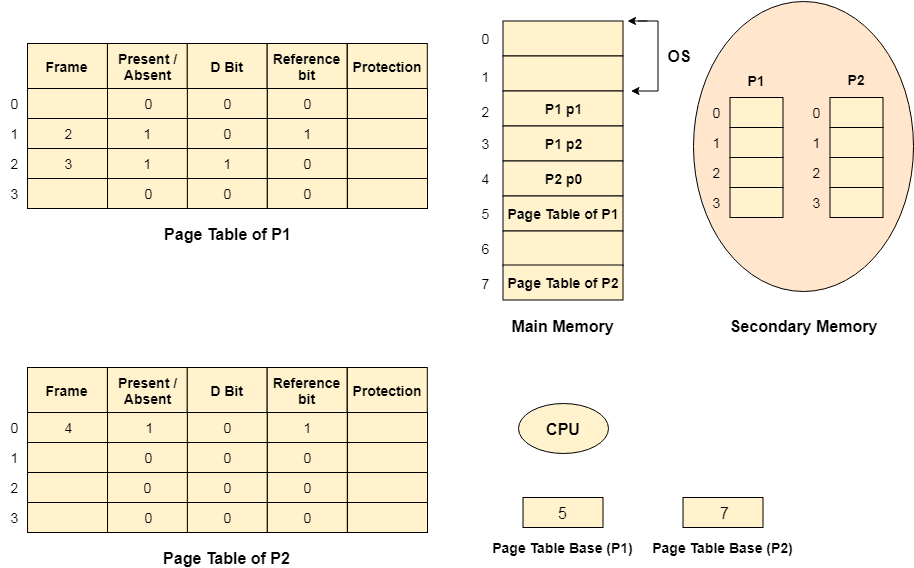
A page is copied to the main memory when its demand is made or page fault occurs. There are various page replacement algorithms which are used to determine the pages which will be replaced. We will discuss each one of them later in detail.

## Snapshot of a virtual memory management system

Let us assume 2 processes, P1 and P2, contains 4 pages each. Each page size is 1 KB. The main memory contains 8 frame of 1 KB each. The OS resides in the first two partitions. In the third partition, 1st page of P1 is stored and the other frames are also shown as filled with the different pages of processes in the main memory.

The page tables of both the pages are 1 KB size each and therefore they can be fit in one frame each. The page tables of both the processes contain various information that is also shown in the image.

The CPU contains a register which contains the base address of page table that is 5 in the case of P1 and 7 in the case of P2. This page table base address will be added to the page number of the Logical address when it comes to accessing the actual corresponding entry.



### 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.

Translation Look aside buffer

Drawbacks of Paging

1. Size of Page table can be very big and therefore it wastes main memory.
2. CPU will take more time to read a single word from the main memory.

How to decrease the page table size

1. The page table size can be decreased by increasing the page size but it will cause internal fragmentation and there will also be page wastage.
2. Other way is to use multilevel paging but that increases the effective access time therefore this is not a practical approach.

How to decrease the effective access time

1. CPU can use a register having the page table stored inside it so that the access time to access page table can become quite less but the register are not cheaper and they are very small in compare to the page table size therefore, this is also not a practical approach.
2. To overcome these many drawbacks in paging, we have to look for a memory that is cheaper than the register and faster than the main memory so that the time taken by the CPU to access page table again and again can be reduced and it can only focus to access the actual word.

Locality of reference

In operating systems, the concept of locality of reference states that, instead of loading the entire process in the main memory, OS can load only those number of pages in the main memory that are frequently accessed by the CPU and along with that, the OS can also load only those page table entries which are corresponding to those many pages.

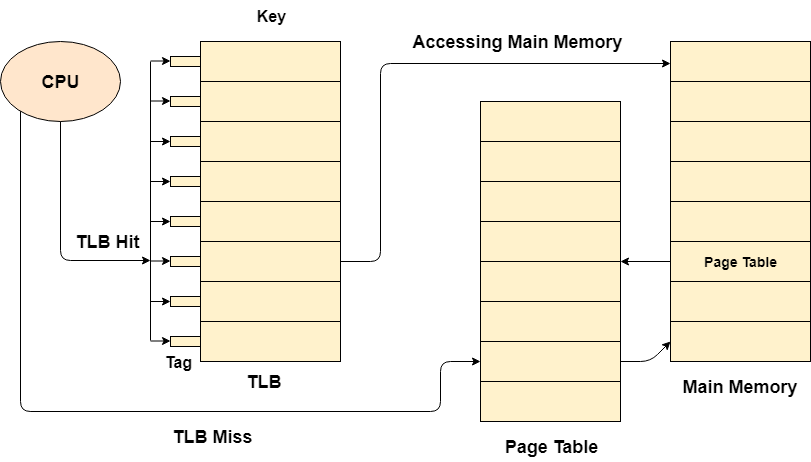
Translation look aside buffer (TLB)

A Translation look aside 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.

TLB follows the concept of locality of reference which means that it contains only the entries of those many pages that are frequently accessed by the CPU.



In translation look aside buffers, there are tags and keys with the help of which, the mapping is done.

TLB hit is a condition where the desired entry is found in translation look aside buffer. If this happens then the CPU simply access the actual location in the main memory.

However, if the entry is not found in TLB (TLB miss) then CPU has to access page table in the main memory and then access the actual frame in the main memory.

Therefore, in the case of TLB hit, the effective access time will be lesser as compare to the case of TLB miss.

If the probability of TLB hit is P% (TLB hit rate) then the probability of TLB miss (TLB miss rate) will be (1-P) %.

Therefore, the effective access time can be defined as;

1. EAT = P (t + m) + (1 - p) (t + k.m + m)

Where, p → TLB hit rate, t → time taken to access TLB, m → time taken to access main memory k = 1, if the single level paging has been implemented.

By the formula, we come to know that

1. Effective access time will be decreased if the TLB hit rate is increased.
2. Effective access time will be increased in the case of multilevel paging.

Demand Paging

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.

Whenever any page is referred for the first time in the main memory, then that page will be found in the secondary memory.

After that, it may or may not be present in the main memory depending upon the page replacement algorithm which will be covered later in this tutorial.

What is a Page Fault?

If the referred page is not present in the main memory then there will be a miss and the concept is called Page miss or page fault.

The CPU has to access the missed page from the secondary memory. If the number of page fault is very high then the effective access time of the system will become very high.

What is Thrashing?

If the number of page faults is equal to the number of referred pages or the number of page faults are so high so that the CPU remains busy in just reading the pages from the secondary memory then the effective access time will be the time taken by the CPU to read one word from the secondary memory and it will be so high. The concept is called thrashing.

If the page fault rate is PF %, the time taken in getting a page from the secondary memory and again restarting is S (service time) and the memory access time is ma then the effective access time can be given as;

1. EAT = PF X S + (1 - PF) X (ma)

Segmentation in Operating System

Last Updated: 16-08-2019

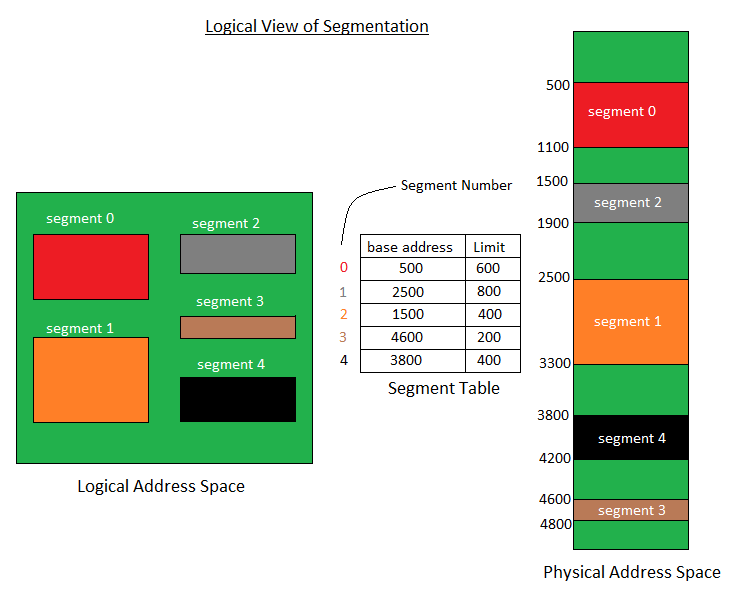
A process is divided into Segments. The chunks that a program is divided into which are not necessarily all of the same sizes are called segments. Segmentation gives user’s view of the process which paging does not give. Here the user’s view is mapped to physical memory.  
There are types of segmentation:

1. **Virtual memory segmentation –**  
   Each process is divided into a number of segments, not all of which are resident at any one point in time.
2. **Simple segmentation –**  
   Each process is divided into a number of segments, all of which are loaded into memory at run time, though not necessarily contiguously.

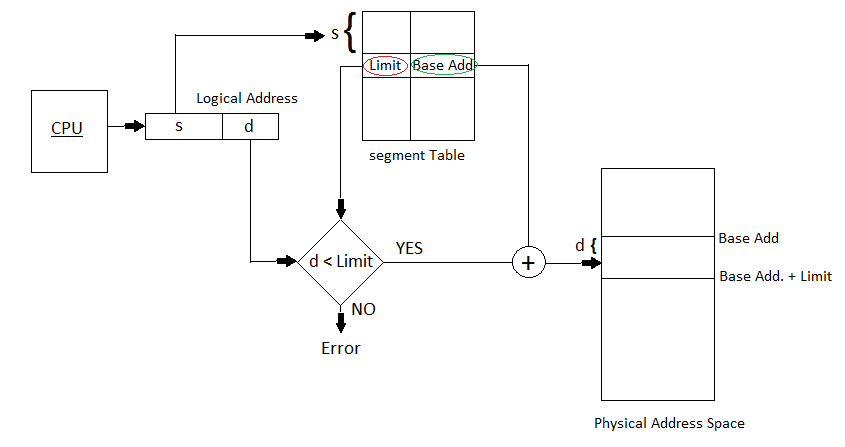
There is no simple relationship between logical addresses and physical addresses in segmentation. A table stores the information about all such segments and is called Segment Table.

**Segment Table –** It maps two-dimensional Logical address into one-dimensional Physical address. It’s each table entry has:

* **Base Address:**Itcontains the starting physical address where the segments reside in memory.
* **Limit:** It specifies the length of the segment.



Translation of Two dimensional Logical Address to one dimensional Physical Address.



Address generated by the CPU is divided into:

* **Segment number (s):** Number of bits required to represent the segment.
* **Segment offset (d):** Number of bits required to represent the size of the segment.

**Advantages of Segmentation –**

* No Internal fragmentation.
* Segment Table consumes less space in comparison to Page table in paging.

**Disadvantage of Segmentation –**

* As processes are loaded and removed from the memory, the free memory space is broken into little pieces, causing External fragmentation.