Ms Vandana Sharma

BCA306

UNIT-4

MEMORY MANAGEMENT

SYLLABUS: Logical address, Physical Address, External and Internal Fragmentation, Concept of paging, Page table structure- Hierarchical Paging, hashed Page Table, Inverted Page Table.

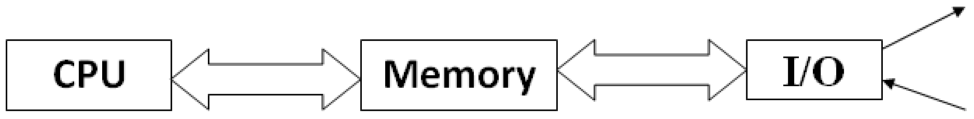
Computer Memory:-

Computer memory can be defined as a collection of some data represented in the binary format. A computer device that is capable to store any information or data temporally or permanently is called storagedevice.

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 checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

The memory management method deals with allocation of finite amount of memory to the requesting process. In ready state it is necessary that the process should have access to the certain amount of memory. Memory is a long array of bytes. Each with its own address using read and write statement the CPU and input/output device interact with the memory.

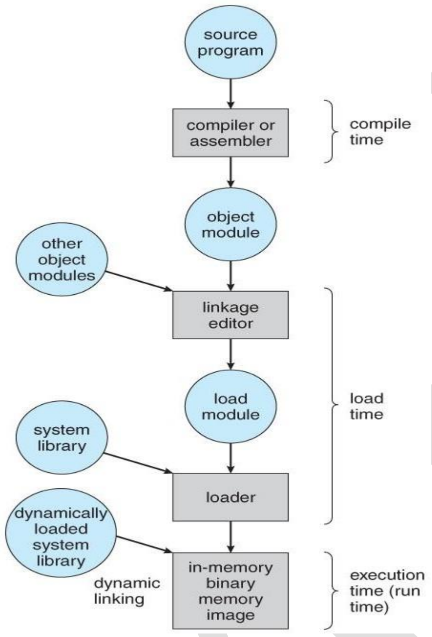


One of the main tasks of an operating system is to manage the computer’s memory. This includes many responsibilities, including. Being aware of what parts of the memory are in use and which parts are not. Allocating memory to processes when they request it and de-allocating memory when a processreleases its memory. Moving data from memory to disc, when the physical capacity becomes full, and vice versa.

Address:- It is two types

1. **Logical address:-** Logical address are generated by the CPU. These addresses are defined by CPU while writing any program generated by CPU.
2. **Physical address:-** The address of memory area where the user program actually resides is called Physical address.

Address Binding Mechanism:- Memory consist of large array of words or bytes. Each with its own address. The processor is to select one of the process from the queue and load into memory as the process is executed. It access data and program from the memory. User programs typically refer to memory addresses with symbolic names such as "i", "count", and "average Temperature". These symbolic names must be mapped or bound to physical memory addresses, which typically occurs in several stages: Diagram shows the various stages of the binding processes and the units involved in each stage:



Multistep processing of a user program

Compile Time - If it is known at compile time where a program will reside in physical memory, then absolute code can be generated by the compiler, containing actual physical addresses. However if the load address changes at some later time, then the program will have to be recompiled. DOS .COM programs use compile time binding.

Load Time - If the location at which a program will be loaded is not known at compile time, then the compiler must generate relocatable code, which references addresses relative to the start of the program. If that starting address changes, then the program must be reloaded but not recompiled.

Execution Time - If a program can be moved around in memory during the course of its execution, then binding must be delayed until execution time. This requires special hardware, and is the method implemented by most modern operating system.

Swapping:-

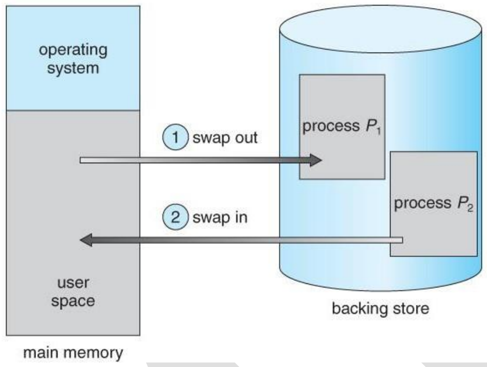
Swapping is a technique for making memory compact. It is a mechanism that is used to temporarily swap processes out of the main memory to secondary memory, and this makes more memory available for some other processes. At some later time, the system can swap back the process from the secondary memory to the main memory. Swapping does affect the performance of the system, but it helps in running multiple processes

TNU

parallels. The total time taken by the swapping of a process includes the time it takes to move the entire process to the secondary memory and then again to the main memory. It is a method of taking out the current content of memory to back store (disk) and bring the content of back store to main memory.

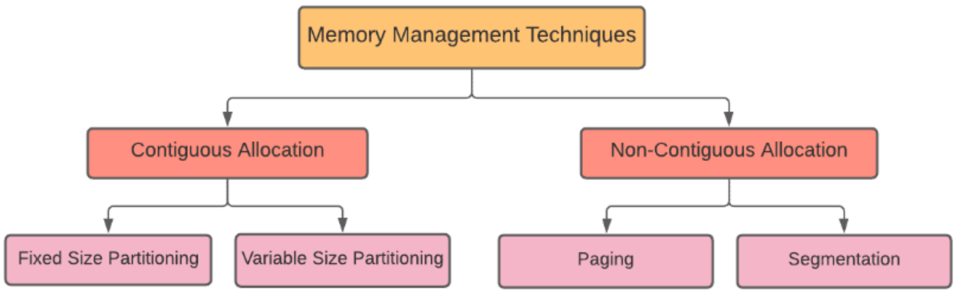
There are two operations in swapping method:-

1. Swap out(Read out):- take out to the current data from the main memory.
2. Swap in(Read in):- bring the data of new user into main memory.



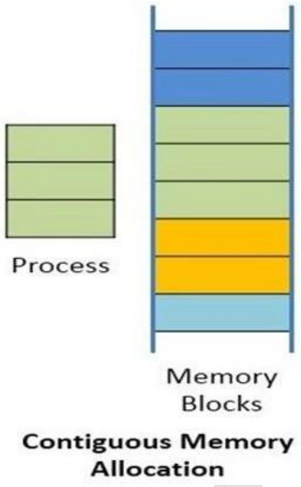
Swapping of two processes using a disk as a backing store

Memory Management Scheme:-


Memory Management Techniques
Contiguous Allocation
Non-Contiguous Allocation
Fixed Size Partitioning
Variable Size Partitioning
Paging
Segmentation

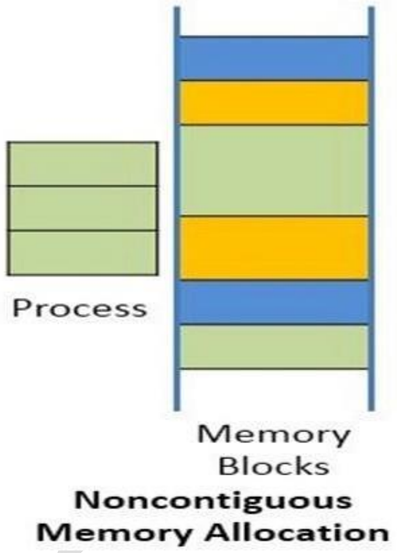
1. Contiguous Memory Allocation:-

In **contiguous memory allocation**, all the available memory space remain together in one place. It means freely available memory partitions are not scattered here and there across the whole memory space. In the **contiguous memory allocation**, both the operating system and the user must reside in the main memory. The main memory is divided into two portions one portion is for the operating and other is for theuser program. In the **contiguous memory allocation** when any user process request for the memory a single section of the contiguous memory block is given to that process according to its need. We can achieve contiguous memory allocation by dividing memory into the fixed-sized partition. A single process is allocated in that fixed sized single partition. But this will increase the degree of multiprogramming means more than one process in the main memory that bounds the number of fixed partition done in memory. Internal fragmentation increases because of the contiguous memory allocation.



2. Non-contiguous memory allocation:-

In the **non-contiguous memory allocation** the available free memory space are scattered here and there and all the free memory space is not at one place. So this is time-consuming. In the **non-contiguous memory allocation**, a process will acquire the memory space but it is not at one place it is at the different locations according to the process requirement. This technique of **non-contiguous memory allocation** reduces the wastage of memory which leads to internal and external fragmentation. This utilizes all the free memory space which is created by a different process.



Difference between Contiguous and Non-contiguous Memory Allocation:

S.NO.	CONTIGUOUS MEMORY ALLOCATION	NON-CONTIGUOUS MEMORYALLOCATION
1.	Contiguous memory allocation allocatesconsecutive blocks of memory to a file/process.	Non-Contiguous memory allocation allocates separate blocks of memory to a file/process.
2.	Faster in Execution.	Slower in Execution.
3.	It is easier for the OS to control.	It is difficult for the OS to control.
4.	Overhead is minimum as not much address translations are there while executing a process.	More Overheads are there as there are moreaddress translations.
5.	Internal fragmentation occurs in Contiguousmemory allocation method.	External fragmentation occurs in Non- Contiguous memory allocation method.
6.	It includes single partition allocation andmulti-partition allocation.	It includes paging and segmentation.
7.	Wastage of memory is there.	No memory wastage is there.
8.	In contiguous memory allocation, swapped-in processes are arranged in the originallyallocated space.	In non-contiguous memory allocation, swapped-in processes can be arranged in anyplace in the memory.

1. Fixed sized partition (Static):-

In the fixed sized partition the system divides memory into fixed size partition (may or may not be of the same size) here entire partition is allowed to a process and if there is some wastage inside the partition is allocated to a process and if there is some wastage inside the partition then it is called internal fragmentation. In this technique, the main memory is divided into partitions of equal or different sizes. The operating system always resides in the first partition while the other partitions can be used to store user processes. The memory is assigned to the processes in contiguous way.

In fixed partitioning,

1. The partitions cannot overlap.
2. A process must be contiguously present in a partition for the execution.

There are various disadvantages of using this technique.

1. Internal Fragmentation

If the size of the process is lesser then the total size of the partition then some size of the partition get wasted and remain unused. This is wastage of the memory and called internal fragmentation. As shown in the image below, the 4 MB partition is used to load only 3 MB process and the remaining 1 MB got wasted.

2. External Fragmentation

The total unused space of various partitions cannot be used to load the processes even though there is space available but not in the contiguous form. As shown in the image below, the remaining 1 MB space of each partition cannot be used as a unit to store a 4 MB process. Despite of the fact that the sufficient space is available to load the process, process willnot be loaded.

3. Limitation on the size of the process

If the process size is larger than the size of maximum sized partition then that process cannot be loaded intothe memory. Therefore, a limitation can be imposed on the process size that is it cannot be larger than the size of the largest partition.

4. Degree of multiprogramming is less

By Degree of multi programming, we simply mean the maximum number of processes that can be loaded into the memory at the same time. In fixed partitioning, the degree of multiprogramming is fixed and very less due to the fact that the size of the partition cannot be varied according to the size of processes.

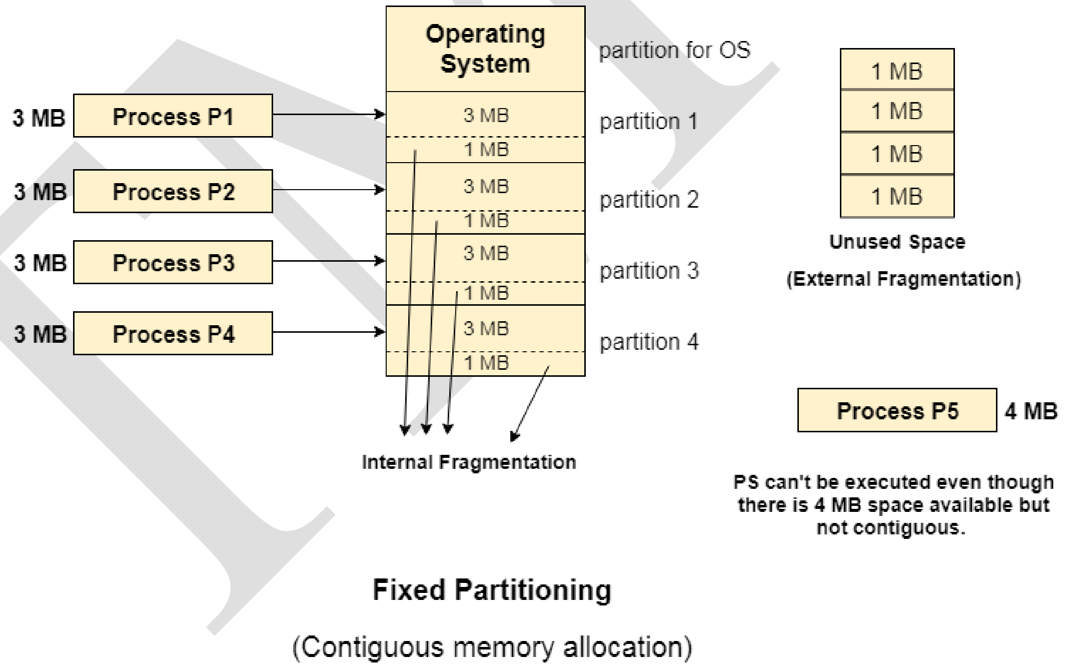
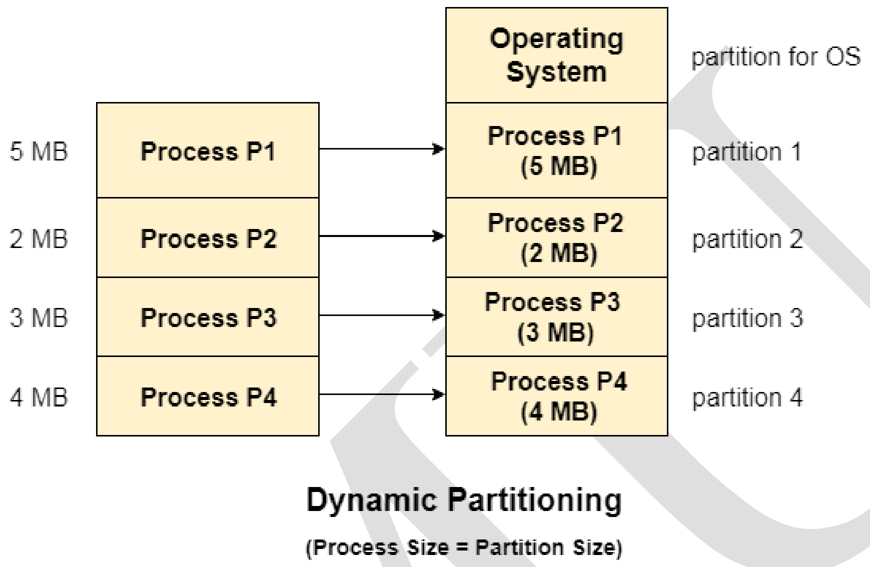


Diagram showing fixed partitions

2. Variable (Dynamic) Partitioning:-

Dynamic partitioning tries to overcome the problems caused by fixed partitioning. In this technique, the partition size is not declared initially. It is declared at the time of process loading. The first partition is reserved for the operating system. The remaining space is divided into parts. The size of each partition will be equal to the size of the process. The partition size varies according to the need of the process so that the internal fragmentation can be avoided.



TMU

Advantages of Dynamic Partitioning over fixed partitioning

1. No Internal Fragmentation

Given the fact that the partitions in dynamic partitioning are created according to the need of the process, It is clear that there will not be any internal fragmentation because there will not be any unused remaining space in the partition.

2. No Limitation on the size of the process

In Fixed partitioning, the process with the size greater than the size of the largest partition could not be executed due to the lack of sufficient contiguous memory. Here, In Dynamic partitioning, the process size can't be restricted since the partition size is decided according to the process size.

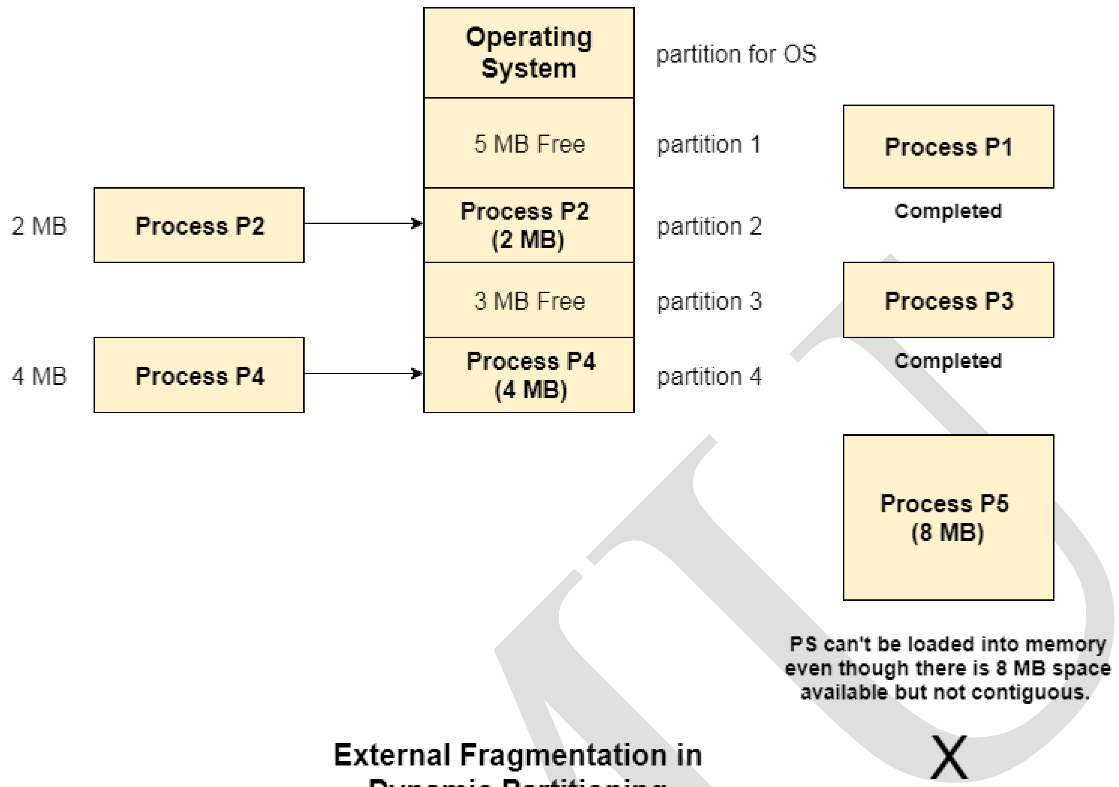
3. Degree of multiprogramming is dynamic

Due to the absence of internal fragmentation, there will not be any unused space in the partition hence more processes can be loaded in the memory at the same time.

Disadvantages of dynamic partitioning

1. External Fragmentation

Absence of internal fragmentation doesn't mean that there will not be external fragmentation. Let's consider three processes P1 (1 MB) and P2 (3 MB) and P3 (1 MB) are being loaded in the respective partitions of the main memory. After some time P1 and P3 got completed and their assigned space is freed. Now there are two unused partitions (1 MB and 1 MB) available in the main memory but they cannot be used to load a 2 MB process in the memory since they are not contiguously located. The rule says that the process must be contiguously present in the main memory to get executed. We need to change this rule to avoid external fragmentation.



External Fragmentation in Dynamic Partitioning

TMU

2. Complex Memory Allocation

In Fixed partitioning, the list of partitions is made once and will never change but in dynamic partitioning, the allocation and de allocation is very complex since the partition size will be varied every time when it is assigned to a new process. OS has to keep track of all the partitions. Due to the fact that the allocation and de allocation are done very frequently in dynamic memory allocation and the partition size will be changed at each time, it is going to be very difficult for OS to manage everything.

Algorithm use for selection of a Free area of a memory:-

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.

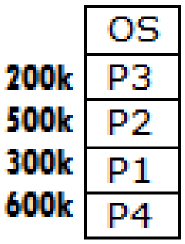
3. Worst Fit:

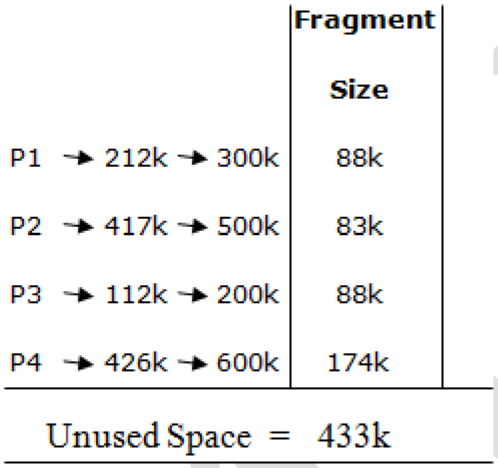
The largest hole that is big enough is allocated to program.

Q1. For the Partition of 200k, 500k, 300k, 600k (In order) place the process of 212k, 417k, 112k, 426k(In order) now fit the value according to the Best Fit algorithm.
Solution:-

200k	500k	300k	600k

P1= 212k,P2= 417k,P3= 112k,P4= 426k



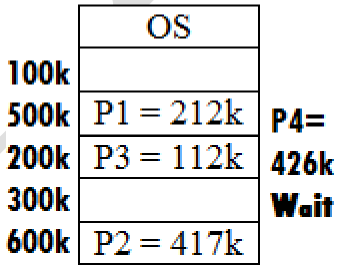


	Fragment Size
P1 → 212k → 300k	88k
P2 → 417k → 500k	83k
P3 → 112k → 200k	88k
P4 → 426k → 600k	174k
Unused Space =	433k

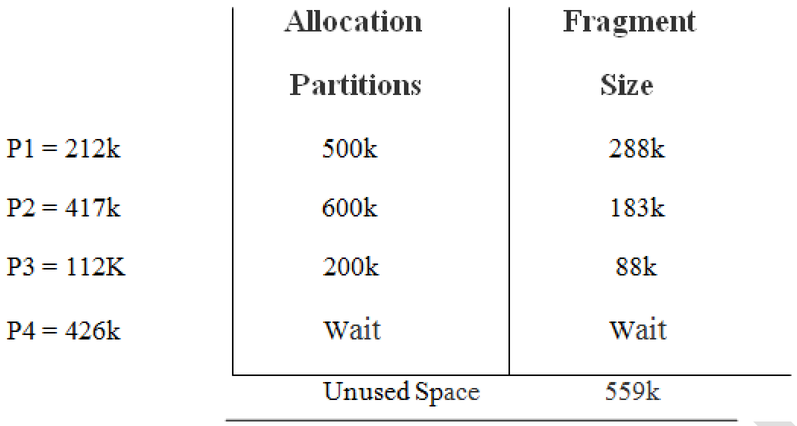
Q2. For the Partition of 100k, 500k, 200k, 300k, 600k (In order) place the process of 212k, 417k,112k, 426k (In order) now fit the value according to the First Fit algorithm.

Solution:-

212k	417k	112k	426k
P1	P2	P3	P4



TMU

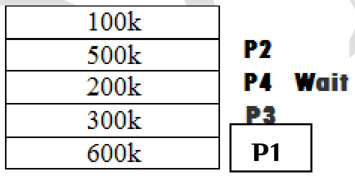


	Allocation Partitions	Fragment Size
P1 = 212k	500k	288k
P2 = 417k	600k	183k
P3 = 112K	200k	88k
P4 = 426k	Wait	Wait
	Unused Space	559k

Q3. For the Partition of 100k, 500k, 200k, 300k, 600k (In order) place the process of 212k, 417k,112k, 426k (In order) now fit the value according to the Worst Fit algorithm.
Solution:-

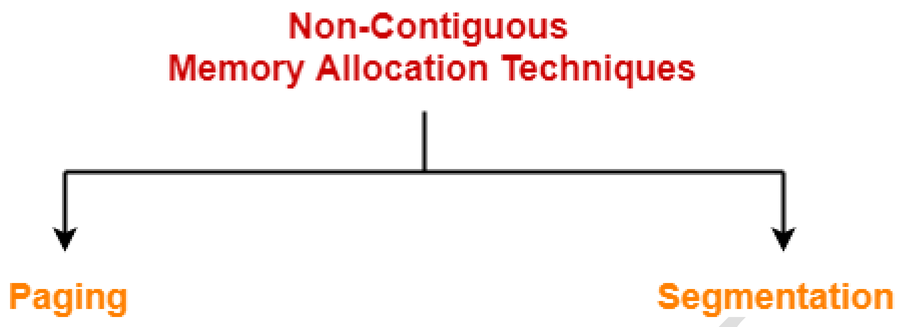
OS	100k	500k	200k	300k	600k

Worst Fit:-
212 K is put in 600 K partition. (Unused space: 600k-212k=388k)
417 K is put in 500 K partition. (Unused space: 500k-417k=83k)
112 K is put in 300 K partition. (Unused space: 300k-112k=188k)
426 K must wait.



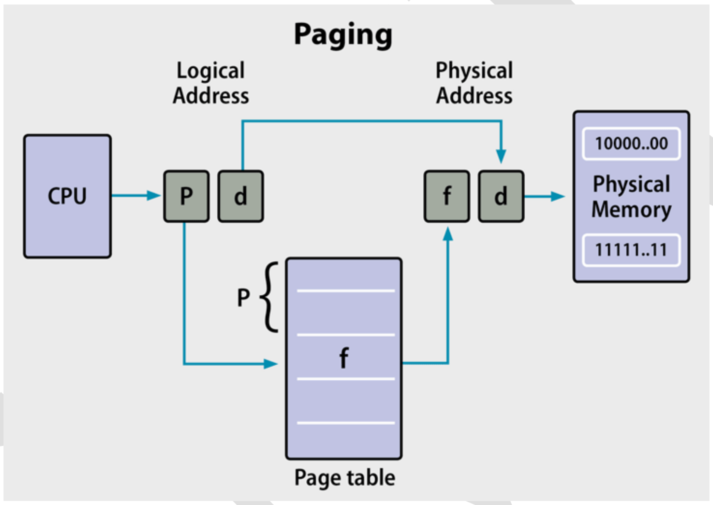
Total Unused Space= 388k+83k+188k== 659k

TMU



Paging:

The memory management function called *paging* specifies storage locations to the CPU as additional memory, called virtual memory. The CPU cannot directly access storagedisk, so the MMU emulates memory by mapping pages to frames that are in RAM. Before we launch into a more detailed explanation of pages and frames, let's define some technical terms.

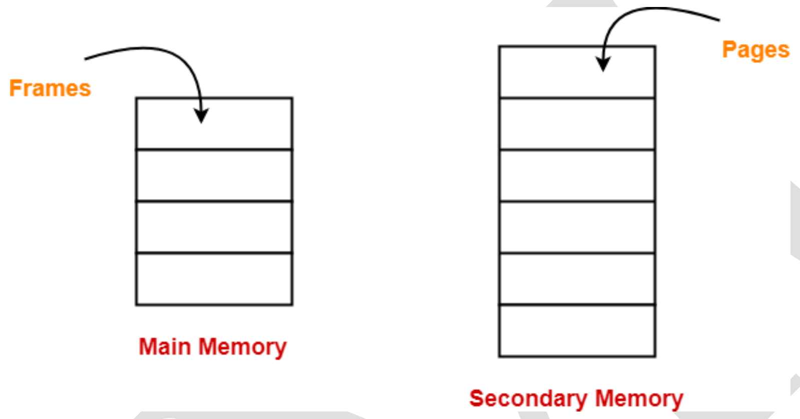


- **Page:** A fixed-length contiguous block of virtual memory residing on disk.
- **Frame:** A fixed-length contiguous block located in RAM; whose sizing is identical to pages.
- **Physical memory:** The computer's random access memory (RAM), typically contained in DIMM cards attached to the computer's motherboard.
- **Virtual memory:** Virtual memory is a portion of an HDD or SSD that is reserved to emulate RAM. The MMU serves up virtual memory from disk to the CPU to reduce the workload on physical memory.
- **Virtual address:** The CPU generates a virtual address for each active process. The MMU maps the virtual address to a physical location in RAM and passes the address to the bus. A virtual address space is the range of virtual addresses under CPU control.
- **Physical address:** The physical address is a location in RAM. The physical address space is the set of all physical addresses corresponding to the CPU's virtual addresses. A physical address space is the range of physical addresses under MMU control.

Let us understand paging in detail:

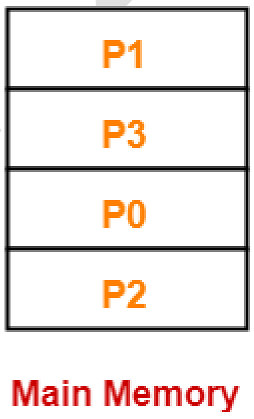
A solution to fragmentation problem is Paging. Paging is a memory management mechanism that allows the physical address space of a process to be non-contagious. Here physical memory is divided into blocks of equal size called **Pages**. The pages belonging to a certain process are loaded into available memory frames.

- Paging is a fixed size partitioning scheme.
- In paging, secondary memory and main memory are divided into equal fixed size partitions.The partitions of secondary memory are called as **pages**.
- The partitions of main memory are called as **frames**.
- Each process is divided into parts where size of each part is same as page size.The size of the last part may be less than the page size.
- The pages of process are stored in the frames of main memory depending upon their availability.



Example-

- Consider a process is divided into 4 pages P_0, P_1, P_2 and P_3.
- Depending upon the availability, these pages may be stored in the main memory frames in a non-contiguous fashion as shown-



TNJ

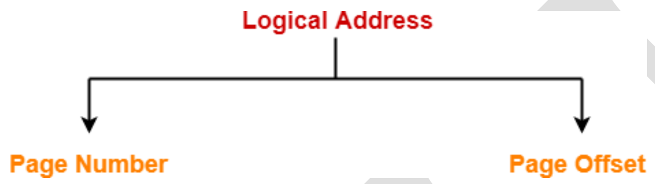
Translating Logical Address into Physical Address-

CPU always generates a logical address. A physical address is needed to access the main memory. Following steps are followed to translate logical address into physical address-

Step-01:

CPU generates a logical address consisting of two parts-

1. Page Number
2. Page Offset



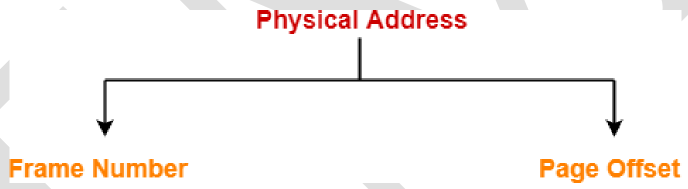
Page Number specifies the specific page of the process from which CPU wants to read the data.
Page Offset specifies the specific word on the page that CPU wants to read.

Step-02:

For the page number generated by the CPU, Page Table provides the corresponding frame number (base address of the frame) where that page isstored in the main memory.

Step-03:

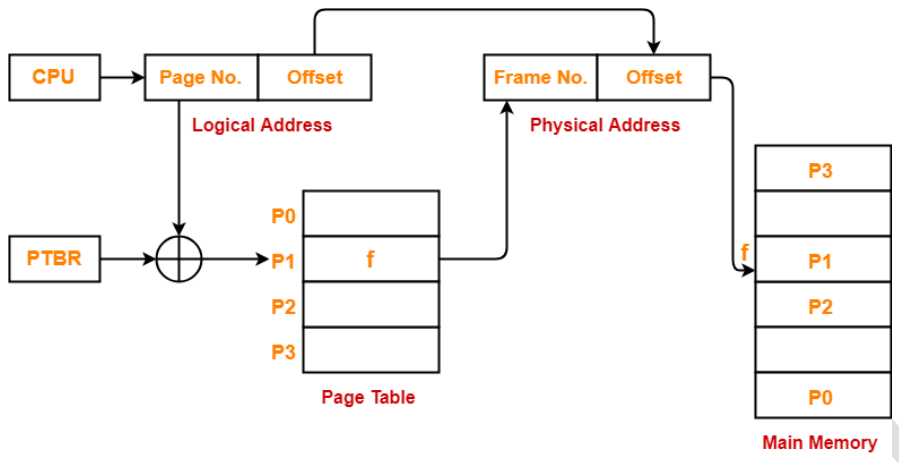
The frame number combined with the page offset forms the required physical address.



Frame number specifies the specific frame where the required page is stored.Page Offset specifies the specific word that has to be read from that page.

Diagram-

The following diagram illustrates the above steps of translating logical address into physical address-



Translating Logical Address into Physical Address

Advantages-

The advantages of paging are-

- It allows storing parts of a single process in a non-contiguous fashion.It solves the problem of external fragmentation.

Disadvantages-

The disadvantages of paging are-

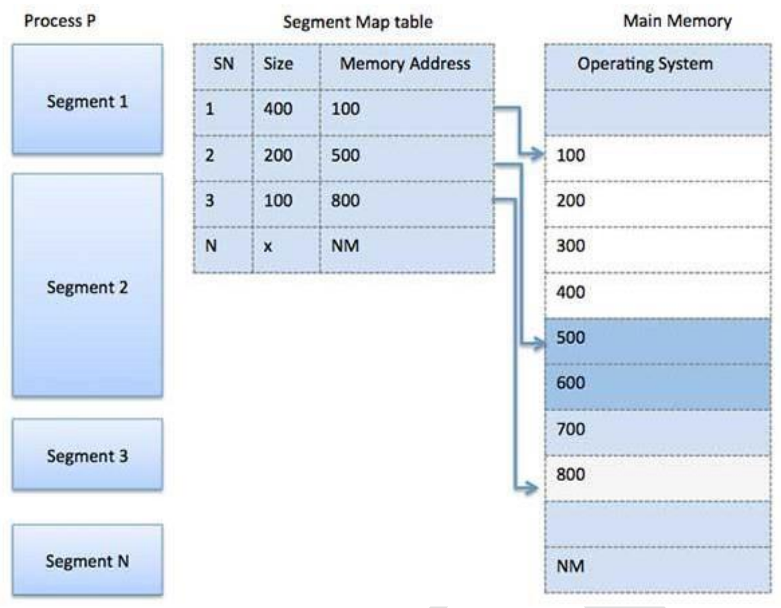
- It suffers from internal fragmentation.
- There is an overhead of maintaining a page table for each process.
- The times taken to fetch the instruction increases since now two memory accesses are required.

Segmentation

Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program. When a process is to be executed, its corresponding segmentation are loaded into non-contiguous memorythough every segment is loaded into a contiguous block of available memory. Segmentation memory management works very similar to paging but here segments are of variable-lengthwhere as in paging pages are of fixed size.

A program segment contains the program's main function, utility functions, data structures, and so on. The operating system maintains a **segment map table** for every process and a list of free memory blocks alongwith segment numbers, their size and corresponding memory locations in main memory. For each segment, the table stores the starting address of the segment and the length of the segment. A reference to a memory location includes a value that identifies a segment and an offset.

TMU

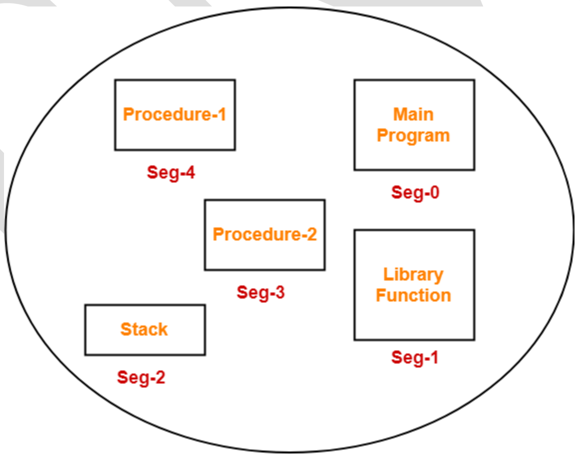


Like Paging, Segmentation is another non-contiguous memory allocation technique. In segmentation, process is not divided blindly into fixed size pages.Rather, the process is divided into modules for better visualization.

Segmentation is a variable size partitioning scheme. In segmentation, secondary memory and main memory are divided into partitions of unequal size.The size of partitions depends on the length of modules. The partitions of secondary memory are called as **segments**.

Example-

Consider a program is divided into 5 segments as-

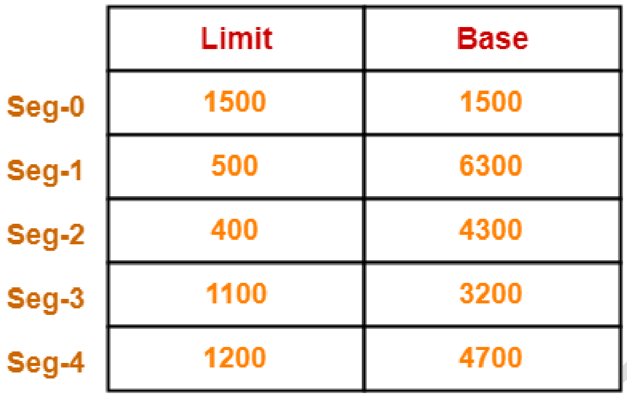


Segment Table-

Segment table is a table that stores the information about each segment of the process.It has two columns. First column stores the size or length of the segment. Second column stores the base address or starting address of the segment in the main memory.Segment table is stored as a separate segment in the main memory. Segment table base register (STBR) stores the base address of the segment table.

For the above illustration, consider the segment table is-

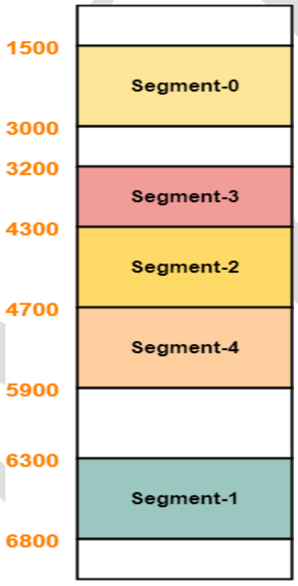
TMU



	Limit	Base
Seg-0	1500	1500
Seg-1	500	6300
Seg-2	400	4300
Seg-3	1100	3200
Seg-4	1200	4700

Segment Table

Here, Limit indicates the length or size of the segment. Base indicates the base address or starting address of the segment in the main memory. In accordance to the above segment table, the segments are stored in the main memory as-



Main Memory

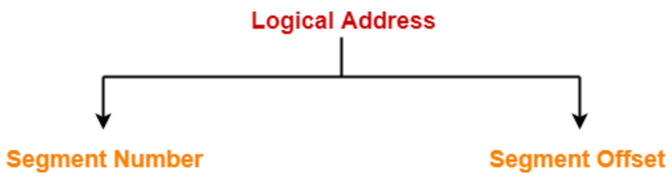
Translating Logical Address into Physical Address-

CPU always generates a logical address. A physical address is needed to access the main memory. Following steps are followed to translate logical address into physical address

Step-01:

CPU generates a logical address consisting of two parts-

1. Segment Number
2. Segment Offset



TNU

Segment Number specifies the specific segment of the process from which CPU wants to read the data.Segment Offset specifies the specific word in the segment that CPU wants to read.

Step-02:

For the generated segment number, corresponding entry is located in the segment table.Then, segment offset is compared with the limit (size) of the segment. Now, two cases are possible-

Case-01: Segment Offset >= Limit

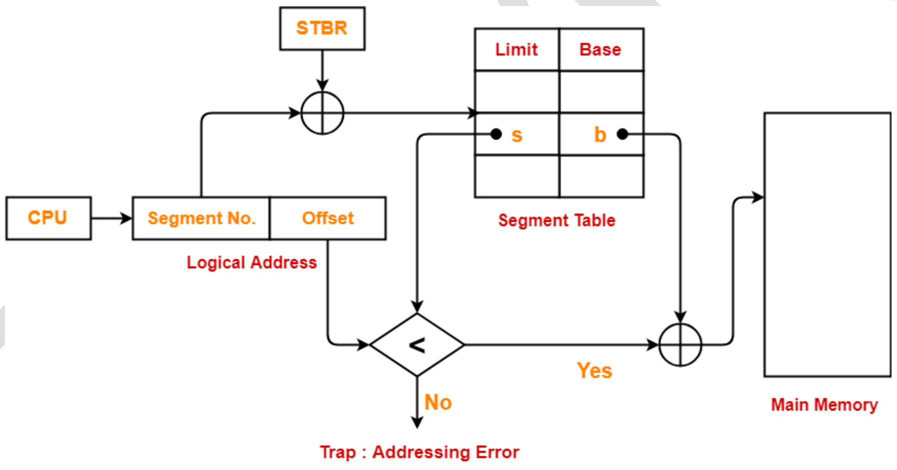
If segment offset is found to be greater than or equal to the limit, a trap is generated.

Case-02: Segment Offset < Limit

If segment offset is found to be smaller than the limit, then request is treated as a valid request.The segment offset must always lie in the range [0, limit-1], Then, segment offset is added with the base address of the segment. The result obtained after addition is the address of the memory location storing the required word.

Diagram-

The following diagram illustrates the above steps of translating logical address into physical address-



Translating Logical Address into Physical Address

Advantages-

The advantages of segmentation are-

- It allows dividing the program into modules which provides better visualization.Segment table consumes less space as compared to **Page Table** in paging.
- It solves the problem of internal fragmentation.

Disadvantages-

The Disadvantages of segmentation are-

- There is an overhead of maintaining a segment table for each process.
- The time taken to fetch the instruction increases since now two memory accesses is required.Segments of

unequal size are not suited for swapping.
- It suffers from external fragmentation as the free space gets broken down into smaller pieces with the processes being loaded and removed from the main memory.

Paging VS Segmentation

S.no	Paging	Segmentation
1	Non-Contiguous memory allocation	Non-contiguous memory allocation
2	Paging divides program into fixed sizepages.	Segmentation divides program into variablesize segments.
3	OS is responsible	Compiler is responsible.
4	Paging is faster than segmentation	Segmentation is slower than paging
5	Paging is closer to Operating System	Segmentation is closer to User
6	It suffers from internal fragmentation	It suffers from external fragmentation
7	There is no external fragmentation	There is no external fragmentation
8	Logical address is divided into pagenumber and page offset	Logical address is divided into segmentnumber and segment offset
9	Page table is used to maintain the page information.	Segment Table maintains the segmentinformation
10	Page table entry has the frame number andsome flag bits to represent details about pages.	Segment table entry has the base address of the segment and some protection bits for thesegments.

Types of Page Table

The data structure used by the virtual memory system in the operating system to store the mapping between physical and logical addresses is commonly known as ***Page Table***. The logical address generated by the CPU is translated into the physical address with the help of the page table. Thus page table mainly provides the corresponding frame number (base address of the frame) where that page is stored in the main memory. Some of the characteristics of the Page Table are as follows:

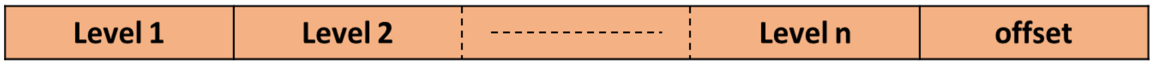
- It is stored in the main memory.
- The number of entries in the page table is equal to the Number of Pages in which the process is divided.
- Page table base register (PTBR) is basically used to hold the base address for the page table of the current process.
- Each process has its own independent page table.

Here are some of the common techniques that are used for structuring the Page table, such as:

1. **Hierarchical Paging or Multilevel Paging**
2. **Hashed Page Tables**
3. **Inverted Page Tables**

1. Hierarchical Paging

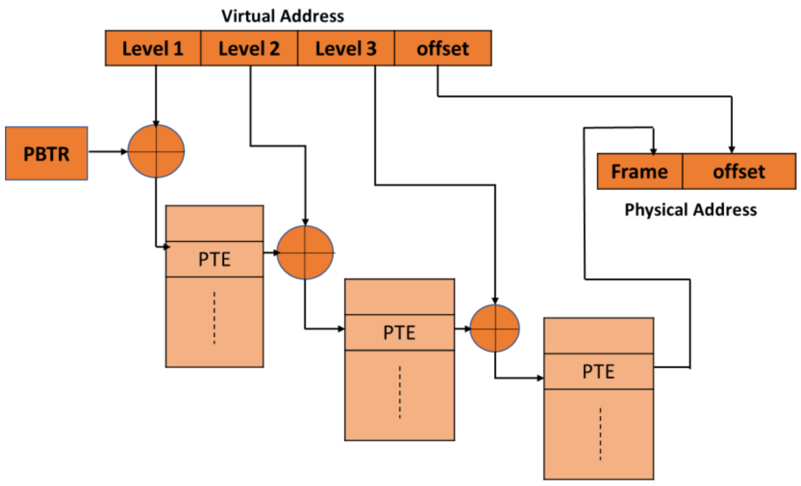
Multilevel paging is a hierarchical technique consisting of two or more layers of page tables. Level 1 page table entries are pointers to a level 2 page table, while level 2 page table entries are pointers to a level 3 page table, and so on. The actual frame information is stored in the entries of the final level page table. Level 1 has a single-page table whose address is contained in PTBR (Page Table Base Register).



Virtual Address

Working of Hierarchical paging

The page table that is larger than the frame size is divided into several parts. Except for the last component, the size of each part is the same as the size of the frame. The page table pages are then stored in various frames of main memory. Another page table is maintained to keep track of the frames that store the pages of the divided page table. As a consequence, the page table hierarchy is created. Multilevel paging is carried out until the level is reached, where the complete page table can be stored in a single frame. In multilevel paging, regardless of the levels of paging, all page tables are kept in the main memory. As a result, obtaining the physical address of a page frame necessitates more than one memory access. Each level requires one access. Except for the last level page table item, each page table entry provides the base address of the subsequent level page table.



3 Level Paging System

2. Hashed Page Table

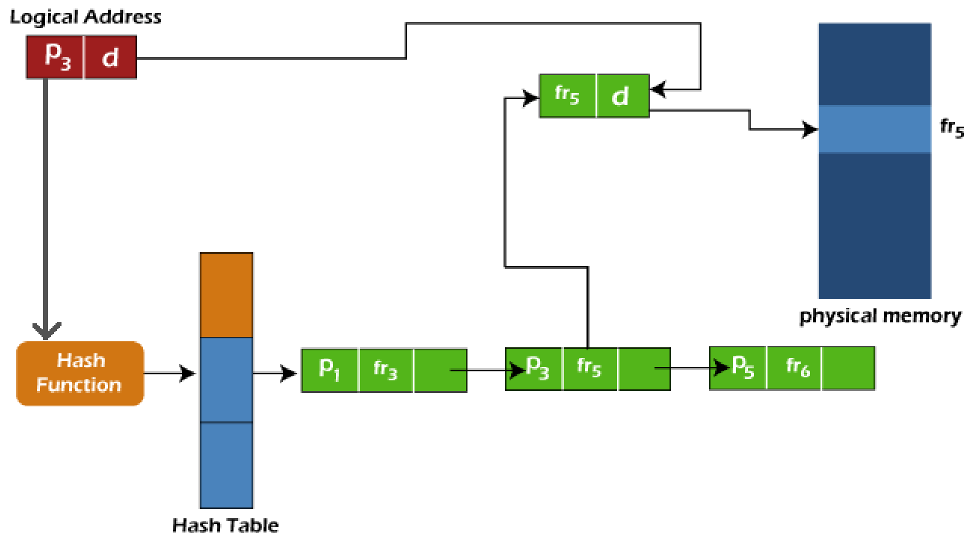
Hashed page tables are a technique for structuring page tables in memory. In a hashed page table, the virtual addresses are hashed into the hash table. Each element in the table comprises a linked list of elements to avoid collisions. The hash value used is the virtual page number, i.e., all the bits that are not part of the page offset. Each element in the hash table has the virtual page number, the value of the mapped page, and a pointer to the next element. For each element in the hash table, there are three fields available,

1. The virtual Page Number (which is the hash value).
2. The value of the mapped page frame.
3. A pointer to the next element in the linked list.

The virtual page number is matched against the first field, i.e., the virtual address, and if a match is found, the corresponding mapped address in the second field is used to form the desired memory address. If a match is not found, the linked list is traversed using the next pointer until a match is found. Though we can structure the large page table using the multilevel page table, it would consist of several levels that increase the page table's complexity.

Working of Hashed Page Table

We would understand the working of the hashed page table with the help of an example. The CPU generates a logical address for the page it needs. Now, this logical address needs to be mapped to the physical address. This logical address has two entries, i.e., a page number (P_3) and an offset, as shown below.



- The page number from the logical address is directed to the hash function.
- The hash function produces a hash value corresponding to the page number.
- This hash value directs to an entry in the hash table.
- As we have studied earlier, each entry in the hash table has a link list. Here the page number is compared with the first element's first entry. If a match is found, then the second entry is checked.

In this example, the logical address includes page number P_3 which does not match the first element of the link list as it includes page number P_1. So we will move ahead and check the next element; now, this element has a page number entry, i.e., P3, so further, we will check the frame entry of the element, which is fr_5. We will append the offset provided in the logical address to this frame number to reach the page's physical address. So, this is how the hashed page table works to map the logical address to the physical address.

3. Inverted Page Table

The concept of normal paging says that every process maintains its own page table, which includes the entries of all the pages belonging to the process. The large process may have a page table with millions of entries. Such a page table consumes a large amount of memory. Consider we have six processes in execution. So, six processes will have some or the other of their page in the main memory, which would compel their page tables also to be in the main memory consuming a lot of space. This is the drawback of the paging concept.

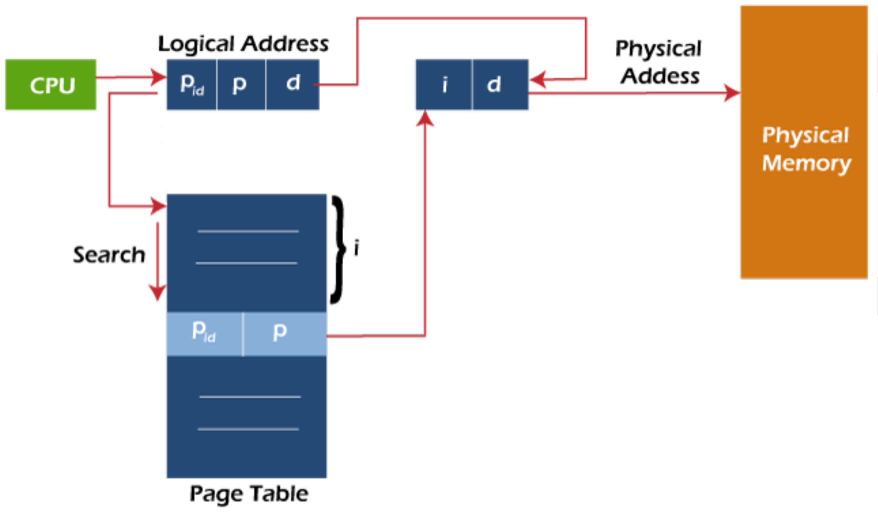
The inverted page table is the solution to this wastage of memory. The concept of an inverted page table consists of a one-page table entry for every frame of the main memory. So the number of page table entries in the Inverted Page Table reduces to the number of frames in physical memory. A single page table represents the paging information of all the processes.

The overhead of storing an individual page table for every process gets eliminated through the inverted page table. Only a fixed portion of memory is required to store the paging information of all the processes together. This technique is called inverted paging, as the indexing is done with respect to the frame number instead of the logical page number. Each entry in the page table contains the following fields.

- **Page number:** It specifies the page number range of the logical address.
- **Process id:** An inverted page table contains the address space information of all the processes in execution. Since two different processes can have a similar set of virtual addresses, it becomes necessary to store each process's process ID to identify its address space uniquely in the Inverted Page Table. This is done by using the combination of Pid and Page Number. So this Process Id acts as an address space identifier and ensures that a virtual page for a particular process is mapped correctly to the corresponding physical frame.

Working of Inverted Page Table

The CPU generates the logical address for the page it needs to access. The logical address consists of three entries process id, page number, and the offset, as shown below.



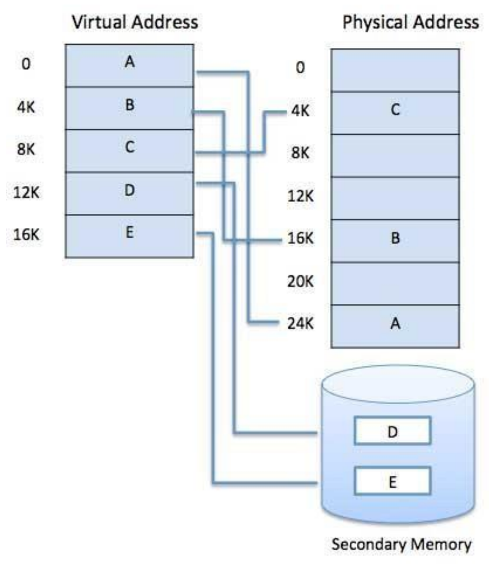
The process id identifies the process of which the page has been demanded, the page number indicates which page of the process has been asked for, and the offset value indicates the displacement required. The match of process id and associated page number is searched in the page table and says if the search is found at the i^{th} entry of page table, then i and offset together generate the physical address for the requested page. This is how the logical address is mapped to a physical address using the inverted page table.

Though the inverted page table reduces the wastage of memory but increases the search time, this is because the entries in an inverted page table are sorted based on physical address. In contrast, the lookup is performed using a logical address. It sometimes happens that the entire table is searched to find the match.

So these are the three techniques that can be used to structure a page table that helps the operating system map the logical address of the page required by the CPU to its physical address.

Virtual Memory:-

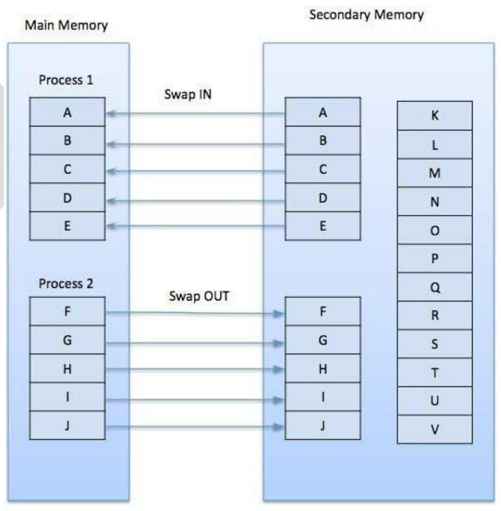
A computer can address more memory than the amount physically installed on the system. This extra memory is actually called **virtual memory** and it is a section of a hard disk that's set up to emulate the computer's RAM. The main visible advantage of this scheme is that programs can be larger than 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. A memory management unit, or MMU, is built into the hardware. The MMU's job is to translate virtual addresses into physical addresses. Virtual memory is commonly implemented by demand paging. A basic example is given below −



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 program’s pages out to the disk or any of the new program’s pages into the main memory Instead, it just begins executing the new program after loading the first page and fetches that program’s 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

Following are the advantages of Demand Paging –

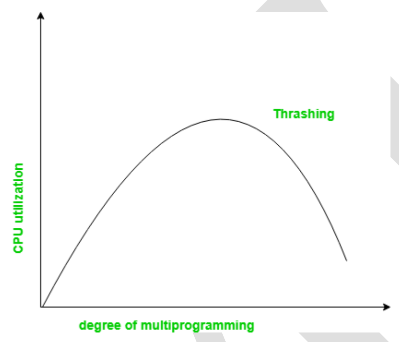
- Large virtual memory.
- More efficient use of memory.
- There is no limit on degree of multiprogramming.

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.

Thrashing:-

Thrashing is a condition or a situation when the system is spending a major portion of its timein servicing the page faults, but the actual processing done is very negligible.



The basic concept involved is that if a process is allocated too few frames, then there will be too many and too frequent page faults. As a result, no useful work would be done by the CPU and the CPU utilization would fall drastically. The long-term scheduler would then try to improve the CPU utilization by loading some more processes into the memory thereby increasing the degree of multiprogramming. This would result in a further decrease in the CPU utilization triggering a chained reaction of higher page faults followed by an increase inthe degree of multiprogramming, called Thrashing.

TMU