**Chapter 5**

**Write in short on basic memory management.**

Memory management is the functionality of an operating system which handles or manages primary memory. Memory management keeps track of each and every memory location either It is allocated to some process. It check show much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets free do run allocated and correspondingly it updates the status.

Two important terms considered while loading programs into the partition are internal fragmentation and external fragmentation.

Internal fragmentation is memory which is internal to a region but not being used i.e. a memory space wasted in a partition. If block of data is smaller than partition size then the internal fragmentation takes place.

External fragmentation occurs when memory partition is available and unused but too small for any waiting process i.e. process size is greater than the size of available partition.

**Fixed Partitioning:-**

Main memory is divided into multiple partitions of fixed size at the time of system generation. A process may be loaded into a partition of equal size or greater size. Partitions can be of equal size or unequal size.

**Equal size partitioning**:-

Main memory is divided into equal size partitions. Any process with less or equal size can be loaded in any available partition. If all partitions are full and no process is in ready or running state then the operating system can swap a process out of any partition and load in any other process.

**Disadvantage**: -Main memory Utilization is extremely inefficient.

**Unequal size partitioning**:-

Main memory is divided into multiple partitions of unequal size. Each process can be loaded into the smallest partition within which the process will fit.



**Functions of memory management**:-

* Process isolation - processes cannot interfere with other processes
* Automatic allocation and management - allocation is transparent, virtual
* Support of modular programming - modules can be of varying size
* Protection and access control - at times memory needs to be shared
* Long-term storage - a file system

**Explain static and dynamic memory partitioning method.**

Static Memory Partitioning:

Main memory is divided into multiple partitions of fixed size at the time of system generation. A process

may be loaded into a partition of equal size or greater size. Partitions can be of equal size or unequal size



**Advantages:**

 Simple to implement

 It requires minimal operating system software and processing overhead as

partitions are fixed at the time of system generation.

**Disadvantages:**

 Memory wastage

 Inefficient use of memory due to internal fragmentation.

 Maximum number of active processes is fixed.

**Dynamic (Variable) Memory partitioning:**

When a process enters in main memory, it is allocated exact size that is required by that process. So in

this method, partitions can vary in size depending on memory space required by a process entering in

main memory. Operating system maintains a table indicating which parts of memory are available and

which are occupied. When new process arrives and it needs space, system searches for available memory

space in mainmemory. If it is available then memory is allocated to the process by creating a partition in

memory. Like this depending on size of process and available memory, partitions take place in main memory.

**For example:-**Consider following table with process and memory space



Total memory size is 64 M .From this 8 M partition is occupied by operating system and remaining can be partitioned as per the size of processes.

**Fig a:** The operating system is loaded in the memory. The entire remaining space is free.

**Fig b:** Process p 1 is loaded with 20 M memory size as space is available in memory. So loading process p1 create one partition in memory.

**Fig c:** Process p 2 is loaded with 14 M memory size as space is available in memory. So loading process p 2 creates one more partition in memory with 14 M size.

**Fig d:** Process p 3 is loaded with 18 M memory size as space is available in memory. So loading process p 3 creates one more partition in memory.

**Fig e & f:** Consider Process P 2 is currently blocked. After some time process P 4 with high priority is ready for execution with memory of 8M.the existing free space is lessthan the required space. With priority scheduling, suppose P 2 is having less priority over P4 then system performs swapping of process P2 and process P 4.in this case, space occupied by process P2 is released i.e. 14 M and P 4 occupies 8 M in that free space as shown in fig f.

**Fig g:** Process P1 completes its job and then it releases its occupied space of 20 M.

**Fig h:** Process P2 can be loaded again in the memory in the free partition released by process P. but P 2 requires only 20 M, so the free space of 20 M is divided into two partitions of 14 M Occupied by P2 and 6 M free space.

Total memory size is 64M.from this 8M partition is occupied by operating system and remaining can be

Partitioned as per the size of the process

**Advantages:**

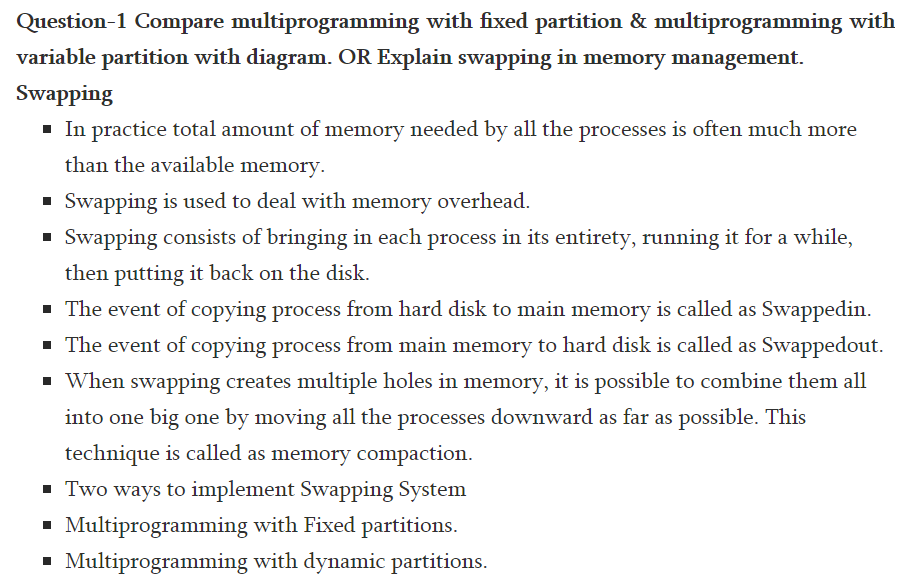
 No internal fragmentation.

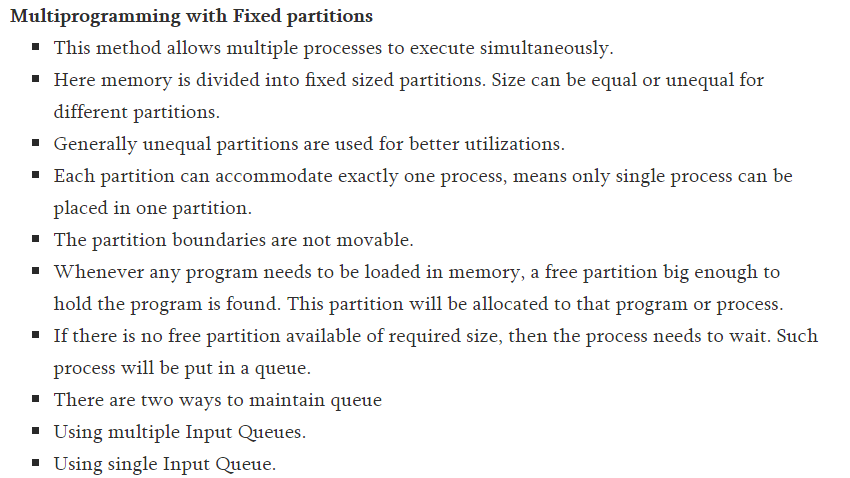
 More efficient use of main memory.

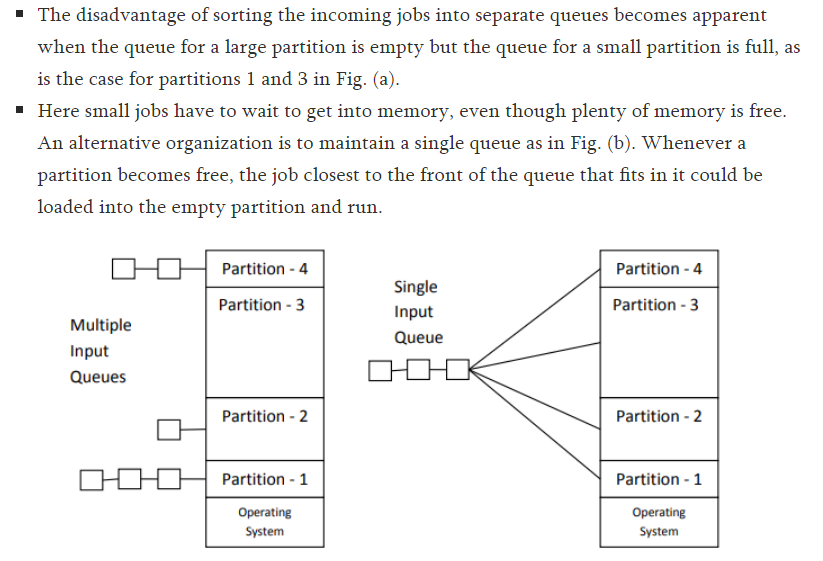
**Disadvantages:**

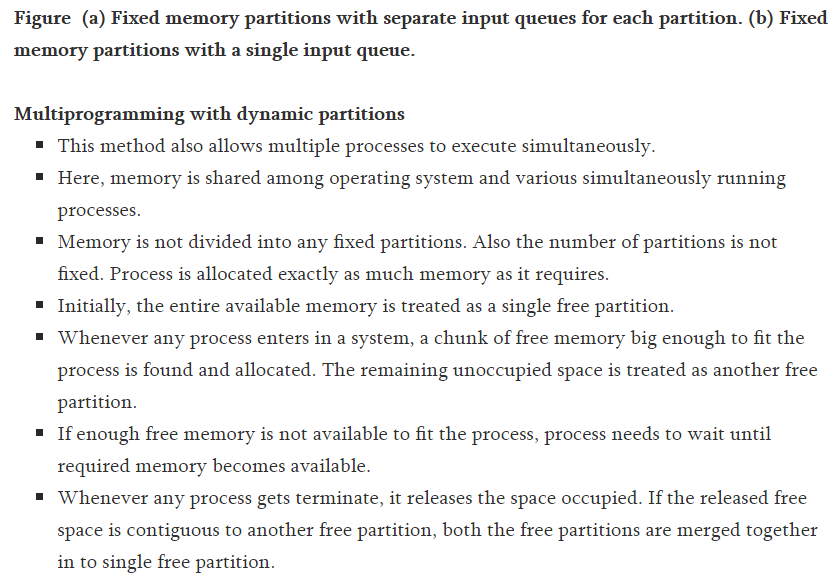
 It suffers from external fragmentation.

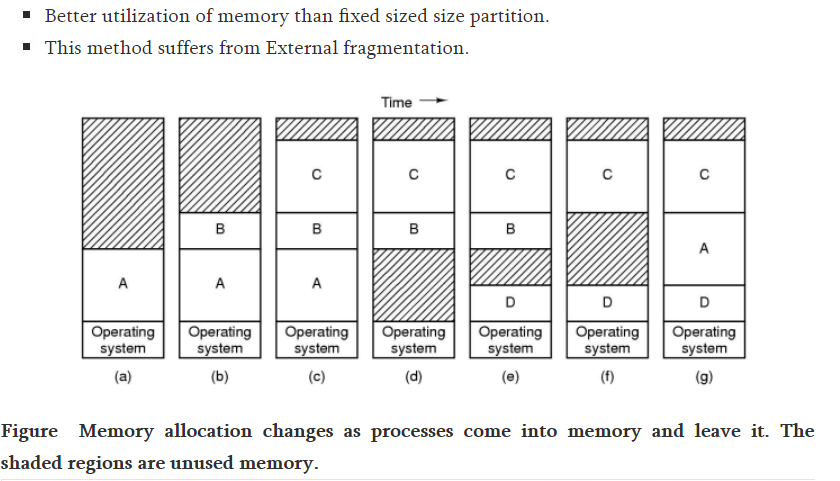
 It needs compaction











**Explain Bit map free-space management technique.**

Frequently, the free-space list is implemented as a bit map or bit vector. Each block is represented by a 1 bit. If the block is free, the bit is 0; if the block is allocated, the bit is

1. For example, consider a disk where blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, and 27 are free, and the rest of the blocks are allocated. The free-space bit map would be:

11000011000000111001111110001111…

The main advantage of this approach is that it is relatively simple and efficient to find n consecutive free blocks on the disk. Unfortunately, bit vectors are inefficient unless the entire vector is kept in memory for most accesses. Keeping it main memory is possible for smaller disks such as on microcomputers, but not for larger ones.

**Explain “Bitmap” method in free space management technique. (Bit Vector)**

The free space management techniques Bitmap is also referred as Bit vector.

**Bit vector:**

The free space list is implemented as a Bit Map or Bit Vector. Each block is represented by one bit. If the block is free, the bit is „1‟; if the block is allocated the bit is „0‟.

Example considers a disk where blocks 2,3,4,5,8,9,10,11,12,13,17,18,25,26 and 27 are free the remaining blocks are allocated then the free space bit map would be:

001111001111110001100000011…

The main advantage of this approach is that it is relatively simple and efficient to find the first free blocks or n consecutive free blocks on the disk.

**Explain concept of Virtual Memory with Diagram**

1. Virtual memory is the separation of user logical memory from physical memory. This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available.
2. Virtual memory makes the task of programming much easier, because the programmer no longer needs to worry about the amount of physical memory available for execution of program.
3. It is the process of increasing the apparent size of a computer's RAM by using a section of the hard disk storage as an extension of RAM.
4. As computers have RAM of capacity 64 or 128 MB to be used by the CPU resources which is not sufficient to run all applications that are used by most users in their expected way and all at once.



Example:

* Consider, an e-mail program, a web browser and a word processor is loaded into RAM simultaneously; the 64 MB space is not enough to store all these programs.
* Without a virtual memory, a message “You cannot load any more applications. Please close an application to load a new one.” would be displayed. By using a virtual memory, a computer can look for empty areas of RAM which is not being used currently and copies them on to the hard disk device.
* Thus RAM is freed to load new applications. Actually it is done automatically, the user do not even know that it is happening, and the user feels like RAM has unlimited space even though the RAM capacity is 32 MB.
* It is a process of increasing computer’s RAM by using a section of the hard disk storage as an extension of RAM.

**What is the concept of paging?**

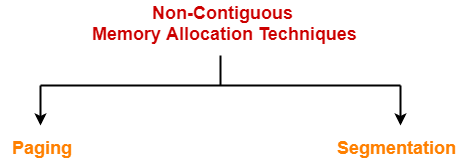
Paging refers to the transfer of memory pages from physical memory to disk and vice versa. Virtual memory uses a technique called demand paging for its implementation. Logical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available.

**Non-Contiguous Memory Allocation-**

* Non-contiguous memory allocation is a memory allocation technique.
* It allows to store parts of a single process in a non-contiguous fashion.
* Thus, different parts of the same process can be stored at different places in the main memory.

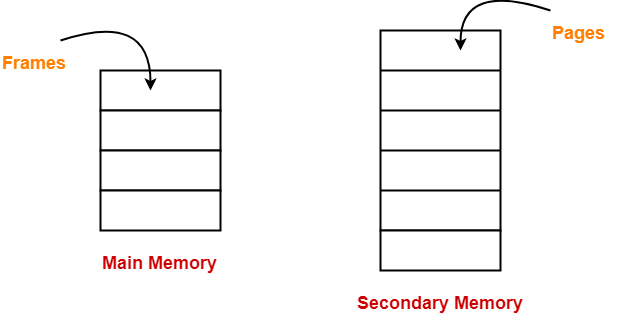
**Techniques-**

 There are two popular techniques used for non-contiguous memory allocation-



**Paging-**

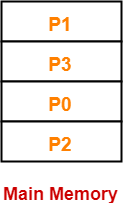
* Paging is a fixed size partitioning scheme.
* In paging, secondary memory and main memory are divided into equal fixed size partitions.
* The partitions of secondary memory are called as **pages**.
* The partitions of main memory are called as **frames**.



* 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 P0, P1, P2 and P3.
* Depending upon the availability, these pages may be stored in the main memory frames in a non-contiguous fashion as shown-



**Translating Logical Address into Physical Address-**

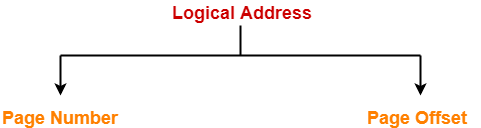
* CPU always generates a logical address.
* A physical address is needed to access the main memory.

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.

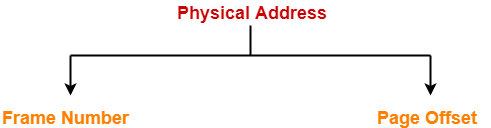
**Step-02:**

 For the page number generated by the CPU,

* [**Page Table**](https://www.gatevidyalay.com/page-table-paging-in-operating-system/) provides the corresponding frame number (base address of the frame) where that page is stored in the main memory.

**Step-03:**

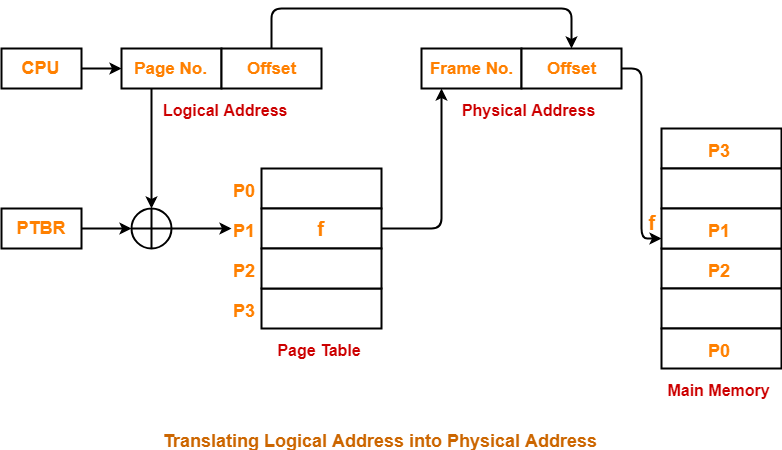
 The frame number combined with the page offset forms the required physical address.



* 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-



**Segmentation**

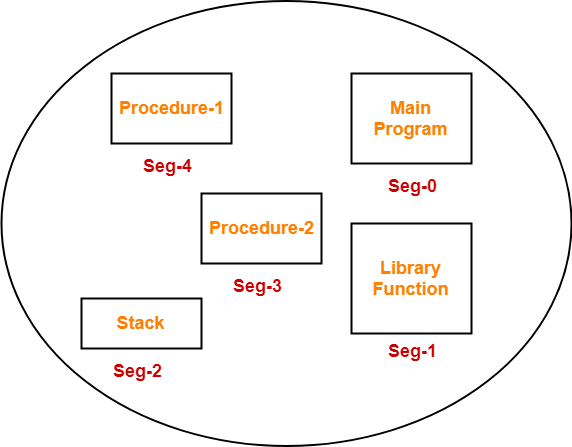
**Characteristics-**

 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 depend on the length of modules.
* The partitions of secondary memory are called as **segments**.

**Example-**

 Consider a program is divided into 5 segments as-

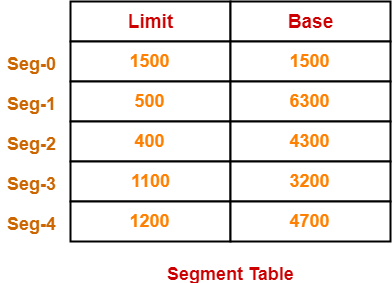


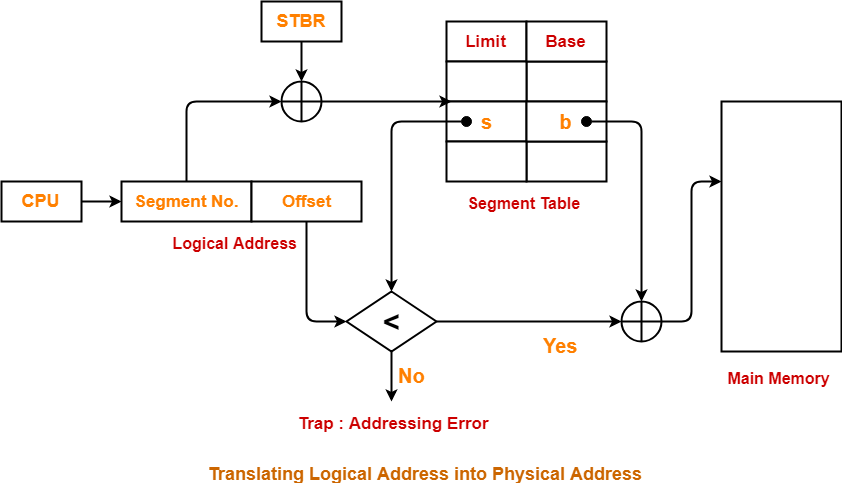
**Segment Table-**

 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-





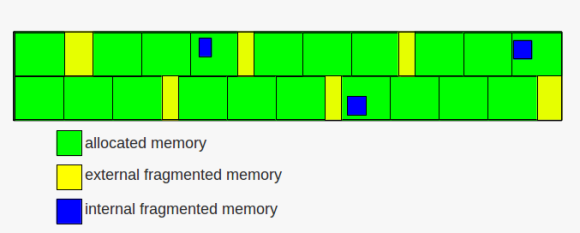
**Difference between Segmentation and Paging (Any 6 points)**

|  |  |  |
| --- | --- | --- |
| **No.** | **Paging** | **Segmentation** |
| 1 | Non-Contiguous memory allocation | Non-contiguous memory allocation |
| 2 | Paging divides program into fixed size pages. | Segmentation divides program into variable size 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 page number and page offset | Logical address is divided into segment number and segment offset |
| 9 | Page table is used to maintain the page information. | Segment Table maintains the segment information |
| 10 | Page table entry has the frame number and some flag bits to represent details about pages. | Segment table entry has the base address of the segment and some protection bits for the segments. |

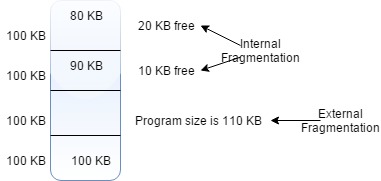
## What is Internal Fragmentation?

It somehow relates to fixed size partitioning. The system allocates memory to various programs and processes by dividing them into small blocks as required by the program. However, more memory is allocated sometimes than is needed by the process, which eventually results in excess memory going waste or left unused.

For example, memory can only be allocated to programs in blocks divisible by 4, 8, or 16. When a process requests 24 bytes, it usually gets a block of 32 bytes, the excess 8 bytes is left unused. Thus unused memory resides within a specific allocated location and it is so small that a new process cannot be allocated to it, resulting in waste. This waste is termed as internal fragmentation. Probably the only way to remove this type of fragmentation is by dynamic memory allocation.

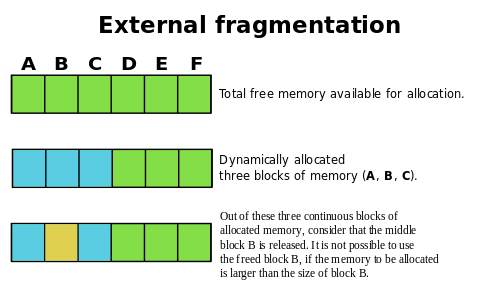


When a program is allocated to a memory block, if that program is lesser than this memory block and remaining space goes wasted, this situation is called internal fragmentation. Generally, internal fragmentation memory partition is static or fixed.

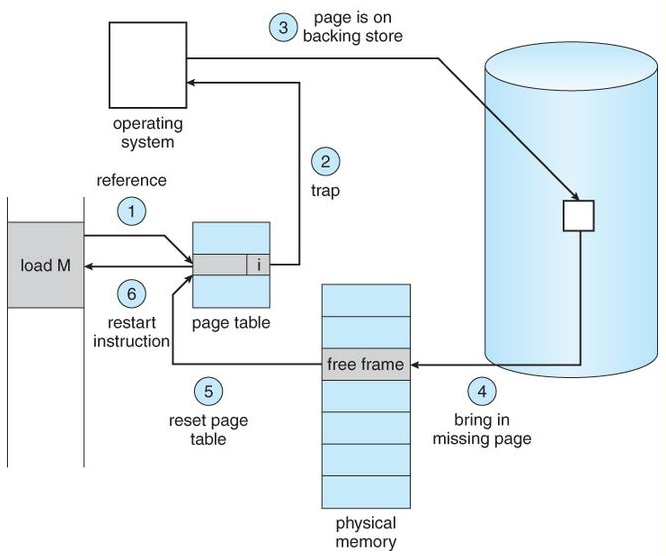


## What is External Fragmentation?

The main memory forms holes between portions of allocated memory that are too small to hold any process. It’s the downside of storage allocation algorithms, when contiguous blocks of unused spaces cannot serve a new request because the spaces are too small for large memory application needs. In simple terms, the non-contiguous blocks create holes in the memory resulting in unused storage that are outside the allocated regions, meaning it cannot be used along with the main memory for larger memory tasks. They end up being isolated and cannot be totally eliminated from the memory space. This is called external fragmentation. It can be removed by compaction which shuffles contents of the memory to place all free memory together.



**Page Fault**

A page fault is occurs when a **program attempts to access a block of memory** that is not stored in the physical memory, or RAM. The fault notifies the OS that it must locate the data in Virtual Memory, then transfer it from the storage device, such as an**HDD or SSD**, to the system **RAM**.  
  


Page faults are common and are part of the normal way computers handle**virtual memory**. In programming terms, a page **fault generates an exception**, which notifies the operating system that it must retrieve the memory blocks or "**pages**" from virtual memory in order for the program to continue. Once the **DATA is moved into physical memory**, the program continues as normal. This process takes place in the background and usually goes unnoticed by the user.

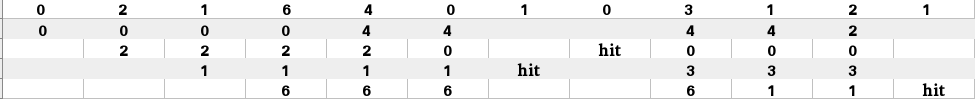
**First In First Out (FIFO) page replacement algorithm –**  
This is the simplest page replacement algorithm. In this algorithm, operating system keeps track of all pages in the memory in a queue, oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

**Example -1.** Consider page reference string 1, 3, 0, 3, 5, 6 and 3 page slots.

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.**  
Finally 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>1 **Page Fault.**

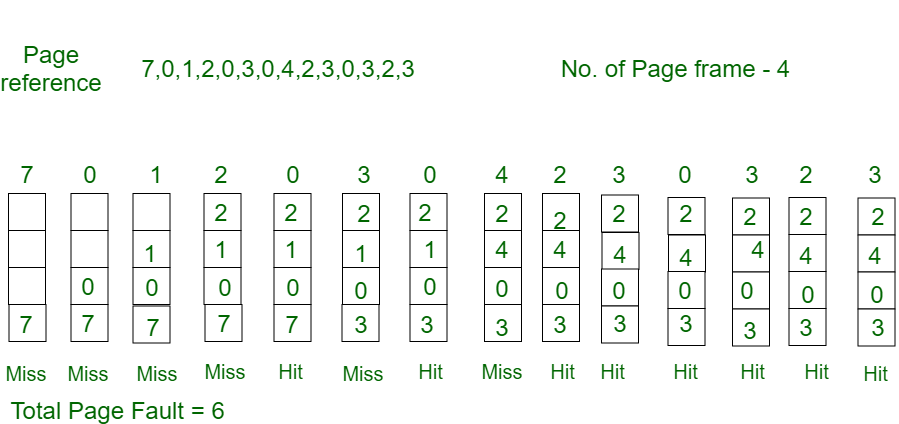
So total page faults = **5**.

**Example -2.** Consider the following reference string: 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1.

Using FIFO page replacement algorithm –  


So, total number of page faults = 9.

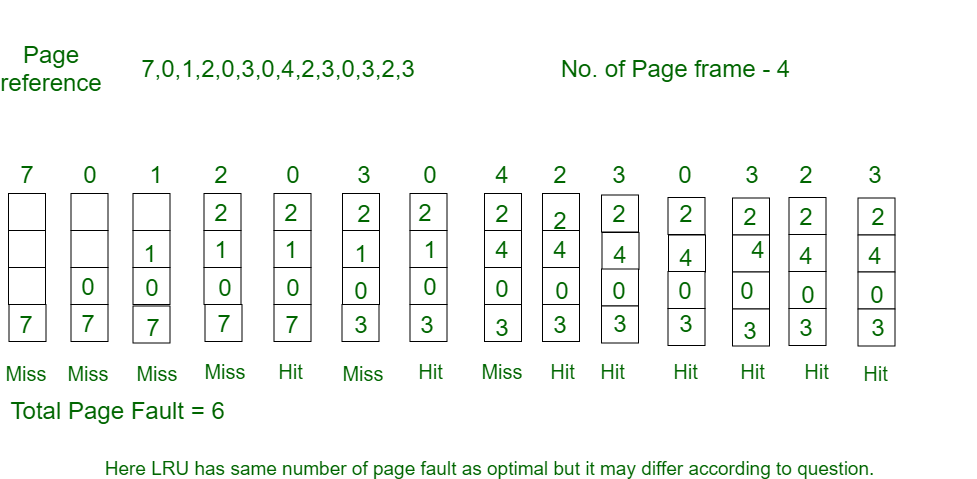
* **Optimal Page replacement –**  
  In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

**Example-2:**Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, with 4 page frame. Find number of page fault.  


Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —>**4 Page faults**  
0 is already there so —> **0 Page fault.**  
when 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—>**1 Page fault.**  
0 is already there so —>**0 Page fault.**.  
4 will takes place of 1 —>**1 Page Fault.**  
Now for the further page reference string —>**0 Page fault** because they are already available in the memory.

Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

**Least Recently Used – (LRU)**  
In this algorithm page will be replaced which is least recently used.

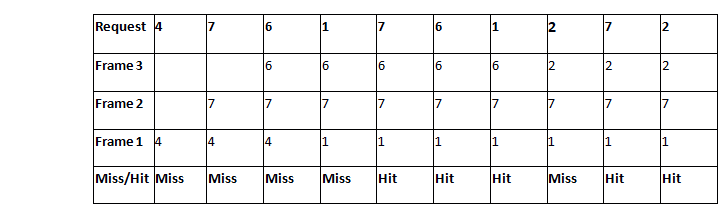
**Example-3**Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 with 4 page frames.Find number of page faults.  


Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —>**4 Page faults**  
0 is already their so —> **0 Page fault.**  
when 3 came it will take the place of 7 because it is least recently used —>**1 Page fault**  
0 is already in memory so —>**0 Page fault**.  
4 will takes place of 1 —>**1 Page Fault**  
Now for the further page reference string —>**0 Page fault** because they are already available in the memory.

Q. 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:

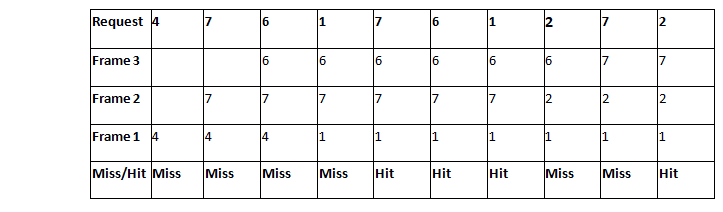
1. Optimal Page Replacement Algorithm
2. FIFO Page Replacement Algorithm
3. LRU Page Replacement Algorithm

## Optimal Page Replacement Algorithm

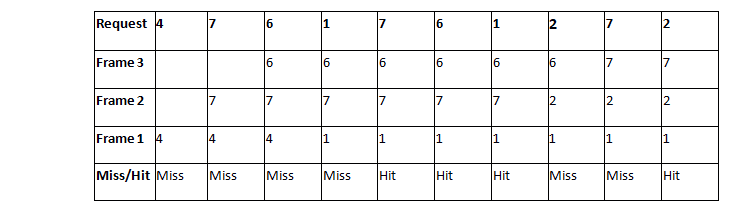
**Number of Page Faults in Optimal Page Replacement Algorithm = 5**

## LRU Page Replacement Algorithm

**Number of Page Faults in LRU = 6**

## FIFO Page Replacement Algorithm

**Number of Page Faults in FIFO = 6**

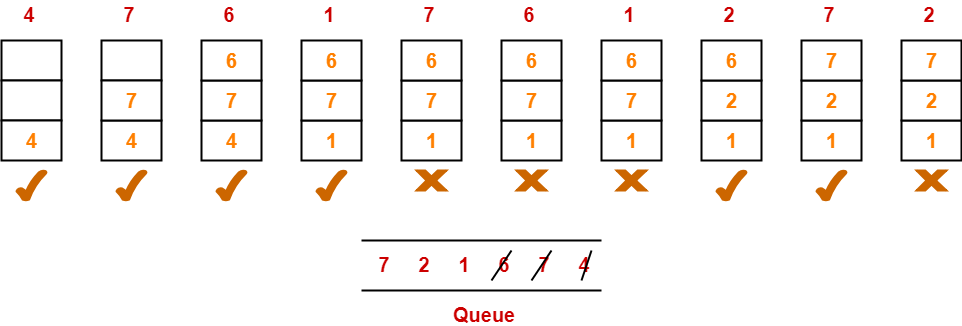
A system uses 3 page frames for storing process pages in main memory. It uses the First in First out (FIFO) page replacement policy. Assume that all the page frames are initially empty. What is the total number of page faults  that will occur while processing the page reference string given below-

4 , 7, 6, 1, 7, 6, 1, 2, 7, 2

Also calculate the hit ratio and miss ratio.

## ****Solution-****

Total number of references = 10



From here,

Total number of page faults occurred = 6

### **Calculating Hit ratio-**

Total number of page hits

= Total number of references – Total number of page misses or page faults

= 10 – 6

= 4

Thus, Hit ratio

= Total number of page hits / Total number of references

= 4 / 10

= 0.4 or 40%

### **Calculating Miss ratio-**

Total number of page misses or page faults = 6

Thus, Miss ratio

= Total number of page misses / Total number of references

= 6 / 10

= 0.6 or 60%

**Alternatively**,

Miss ratio

= 1 – Hit ratio

= 1 – 0.4

= 0.6 or 60%

## ****Problem-02:****

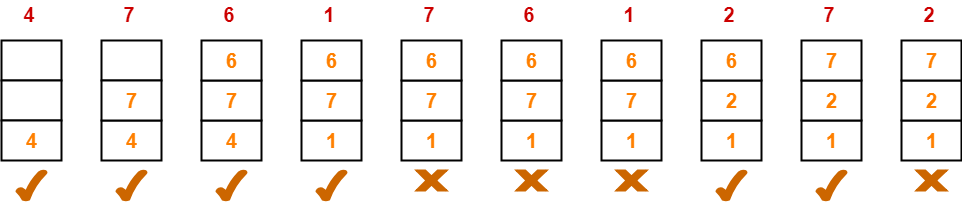
A system uses 3 page frames for storing process pages in main memory. It uses the Least Recently Used (LRU) page replacement policy. Assume that all the page frames are initially empty. What is the total number of page faults  that will occur while processing the page reference string given below-

4 , 7, 6, 1, 7, 6, 1, 2, 7, 2

Also calculate the hit ratio and miss ratio.

## ****Solution-****

Total number of references = 10



From here,

Total number of page faults occurred = 6

In the similar manner as above-

* Hit ratio = 0.4 or 40%
* Miss ratio = 0.6 or 60%

## ****Problem-03:****

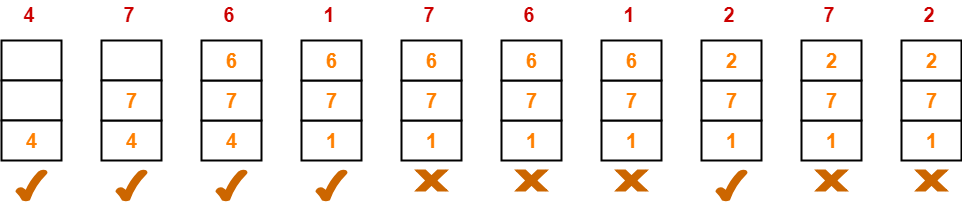
A system uses 3 page frames for storing process pages in main memory. It uses the Optimal page replacement policy. Assume that all the page frames are initially empty. What is the total number of page faults  that will occur while processing the page reference string given below-

4 , 7, 6, 1, 7, 6, 1, 2, 7, 2

Also calculate the hit ratio and miss ratio.

## ****Solution-****

Total number of references = 10



From here,

Total number of page faults occurred = 5

**Difference between bitmap and linked list free space management techniques**

Bitmap:

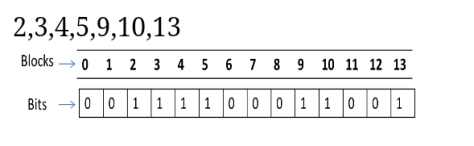
When free space is list is implemented as bit map, each block of disk is represented by a bit.

REPORT THIS AD

If the block is free, its bit is set to 1. If the block is allocated, the bit is 0.

For Example: Apple Macintosh operating system uses bit map method to allocate the disk space.

Assume the following are free. Rest are allocated:

[](https://ccsindian.files.wordpress.com/2017/03/wp-1490642922841.png)

Advantages:

  It is relatively simple.

  Efficient to find the free space on the disk.

  Fast random access allocation check

Disadvantages:

  This may require special hardware support to find the first 1 in a word if it is not 0.

  Wasteful on larger disks

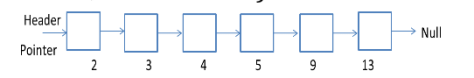
  Poor scalability

Linked List:

In this method, a linked list of all the free disk blocks is maintained.

The first free block in the list can be pointed out by a head pointer, which is kept in a special location on the disk.

Using this disk, It is not easy to search the free list.

[](https://ccsindian.files.wordpress.com/2017/03/wp-1490642980301.png)

Advantages:

  Whenever a file is to be allocated a free block, the operating system can simply allocate the first block in free space list and move the head pointer to the next free block in the list.

Disadvantages:

  Searching the free space list will be very time consuming; each block will have to be read from the disk, which is read very slowly as compared to the main memory.

  Not Efficient for faster access.