Unit-3 Memory Management

❖ Memory Management

- MMU (Memory Management Unit) is a part of OS that deals with main memory.
- MMU is memory manager. Its main goal is to efficiently utilize the main memory among various simultaneously executing processes.
- Requirement and Functionalities of MMU.

It perform operation like

1. **Fetch:**
   - It should determine when to move information from disk to main memory.
2. **Placement**:
   - It should determine where to put fetched information in memory (means memory is allocated for fetched information).
3. **Replacement**:
   - If memory is required but not available it needs to determine which information to remove from memory.
4. **Sharing**:
   - It needs to allow more than one process to share a piece of memory.
5. **Address Translation:**
   - It is responsible for translation logical address to physical address.
6. **Protection:**
   - It protects memory against an unauthorized request for access.
   - One process should not have access to unauthorized information of another process.

❖ Logical and physical address map

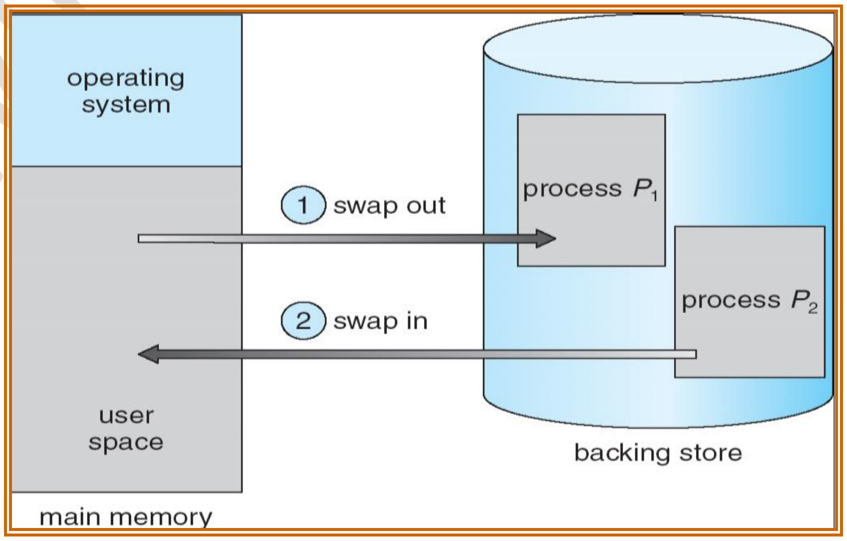
Physical address allocated to the process may not exactly match the logical addresses of process so there should be some mapping between physical address and logical address, so memory manager is required

	Logical Address	**Physical Address**
1.	The process address space can be considered as a sequential list of bytes. Each byte has an address that is used to locate it, these addresses are called **logical address**.	The entire physical memory can be considered as a sequential list of bytes. Each byte has an address that is used to locate it; these addresses are called **physical address**.
2.	It is generated by CPU, means CPU determines location of each instruction &	It is generated by Main Memory.

	data in process address space.	
3.	Logical Address Space is a set of all logical address that can be referenced by a process.	Physical Address Space is a set of physical address occupied by a process in main memory during its execution.
4.	Address in Logical Address Space is a start from 0 and goes up to some maximum value based on size of process	Physical Address Space may or may not be contiguous based upon the memory location method.
5.	Process can read and write address of its own Logical Address Space.	Process can read and write only those physical addresses that belong to its own physical address space.
6.	Logical Address is limited by address size of the processes. Ex:32 – bit processor can generate up to 2^{32} (4-GB) address.	Physical Address is limited to amount of memory installed.

❖ Swapping

- ***Swapping*** is a technique in which processes are moved between main memory and disk.
- Swapping uses some of the portion of secondary storage (disk) as a backing store; this area is called ***swap area.***
- Operation of moving process from main memory to swap area is ***swap-out.***
- Operation of moving processes from swap area to main memory is ***swap-in.***
- When a process is brought back to memory, it may be loaded at some different location rather than its original one. So It needs memory relocation,
  There is problem of external fragmentation.



❖ Memory Allocation

- Memory should be allocated to various processes and files as per requirements.
- When there is no need allocated memory should be freed.
- Memory has a limited capacity compared to disk. So memory should be allocated carefully among all these processes.
- While allocating memory two goals should be fulfilled.
    1. **High Utilization**
        - Maximum possible memory should be utilized.
        - Mo any single process of memory should be wasted.
    2. **High Concurrency**
        - Maximum processes should be in main memory.
- When more and more processes in main memory, CPU will remain busy most of the time in executing one of the processes, resulting in better throughput.
- Memory is allocated in two ways.

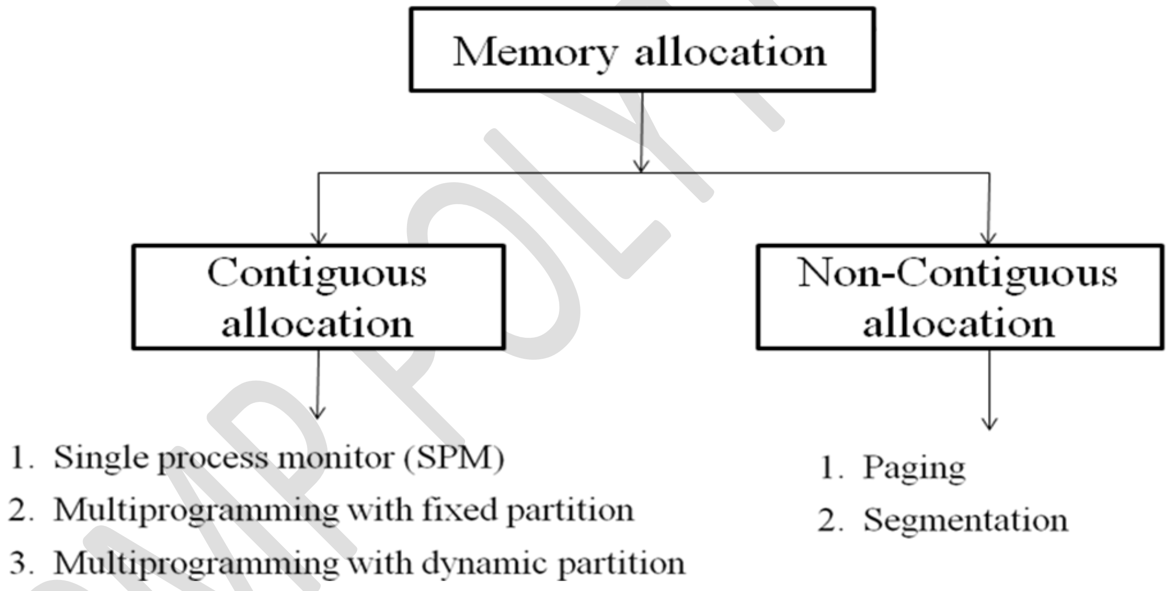

Memory allocation
Contiguous allocation
Non-Contiguous allocation
1. Single process monitor (SPM)
2. Multiprogramming with fixed partition
3. Multiprogramming with dynamic partition
1. Paging
2. Segmentation

FIG: Memory Allocation

1. <u>Contiguous Memory Allocation:</u>

- It is simple and old allocation method.
- It is not used in modern Operating System.
- In this, each processes occupies a block of contiguous memory location in main memory.
- Entire processes kept together in contiguous section of memory.

- When processes is brought in main memory is searched to find a chunk (group) of memory having enough size to hold a processes, once such chunk is found required memory is allocated.
- If contiguous memory space of required size is not available in main memory, this process is made to wait until contiguous space of requires size is available.
- Here logical address space is not divided into any partition.
- Physical Address space will be contiguous without any gaps.

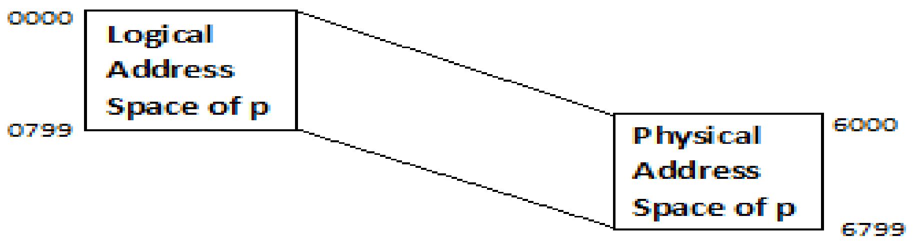


FIG: Contiguous Memory Allocation

- Advantage : - Easy to implement and understand.
- Disadvantage: - Having poor memory utilization.

2. Non - Contiguous Memory Allocation:

- This is used by modern Operating System.
- Here logical address space of processes is divided into partition and for each partition contiguous chunk of free memory is allocated.
- Physical Address space will not be contiguous now.

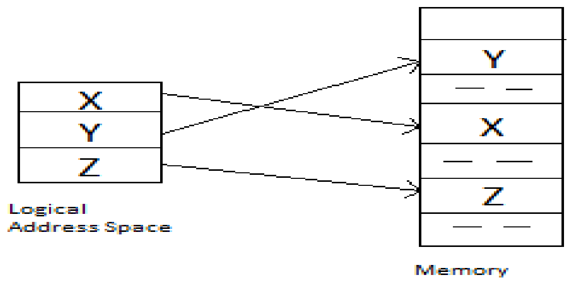


FIG: Non - Contiguous Memory Allocation

- Advantage: - Having better memory utilization.
- Disadvantage: - Complex to implement and understand.

❖ Contiguous memory allocation

- Three different ways for contiguous memory allocation:
  1. Single Process Monitor (SPM)
  2. Multiprogramming with Fixed partition.(static partition)
  3. Multiprogramming with Dynamic partition. (variable partition)

Single Process Monitor(SPM):

- It is simplest possible memory management scheme.
- Only one process is allowed to run.
- No more than one process can run simultaneously.
- Memory is shared between process and Operating System.

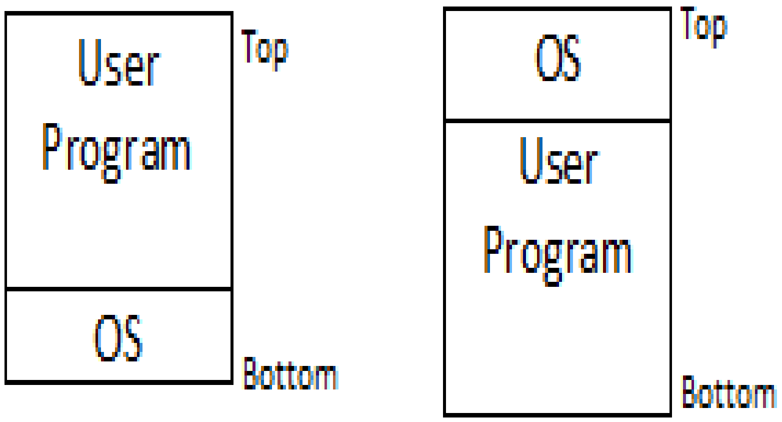


FIG: Single Process Monitor

- This method can be used in palmtop computer, Embedded System, pc running only in MS-DOS :
- Only one process can run at a time
- When user types a command the Operating System copies requested program from disk to memory and executes it.
- When process finishes, the Operating System displays a prompt character and waits for a new command.
- When Operating System loads new program in main memory, overwriting first one and execute it.
- Advantage: -
  - Simplicity.
- Disadvantage: -
  - Only one process can be executed at a time.

❖ Multiprogramming with fixed (static) partition:

- This method allows multiple processes to execute simultaneously.
- Memory is shared among Operating System and various simultaneously running processes.
- Multiprogramming increases the CPU utilization, CPU can be kept busy almost all time by keeping more than one processes simultaneously in memory.
- Memory is divided into fixed partition size can be equal or unequal for different partition. Generally unequal partitions are used for better utilization of memory.
- Each partition is accommodating exactly one process.
- Whenever program needs to be loaded in memory, a free partition big enough to hold program is found and allocated.
- If there is no free partition available of required size, which process needs to wait such process will be put in a queue.
- There are two possible ways to implement method with a queue.
    1. Using multiple input queues. – Fig (a)
    2. Using single input queue. – Fig (b)

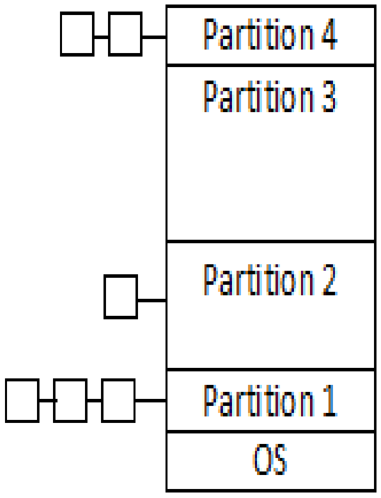


Fig (a)

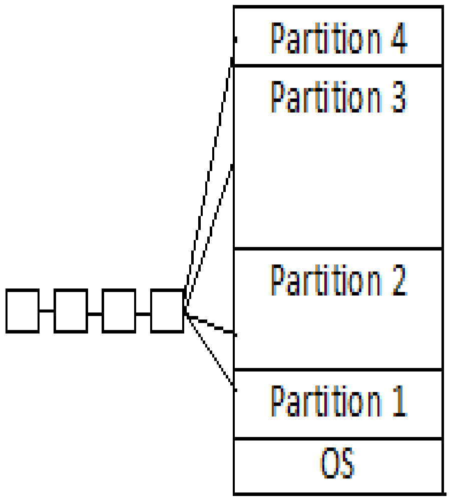


Fig (b)

1. Using Multiple Input Queues.

- For each partition separate queue is maintained.
- Whenever any process comes, it is put in a queue for smallest partition large enough to hold it.
- When partition becomes free a process from the front end of the queue is allocated that partition.
- Disadvantage: - If queue for a large partition is empty, but for small partition is full, small process needs to wait to get memory.
- Here free memory is available but it cannot be used for small process.

2. Using Single Input Queue.

- Only single queue is maintained for all partition.
- Whenever any partition become free, process from queue is selected, this can be hold by that partition.
- Process can be selected in two ways
- Just select process which is near to front of the queue.
  - Disadvantage: large partition will be wasted for small process.
- Search entire queue and select largest process which can fit in that partition.
  - Disadvantage: Starvation possible for small process.

Advantage of fixed partition method:

1. Implement is simple.
2. processing overhead are low.

Disadvantage of fixed partition method:

1. Degree of multiprogramming is fixed here, the maximum number of process can be executed simultaneously can't be changed.
2. Partitions are of fixed size, so any space in a partition not used by a process is wasted, this is called **Internal Fragmentation**.

❖ Multiprogramming with dynamic partition:

- This method allows multiple processes to execute simultaneously.
- Here memory is shared among Operating System and various simultaneously running processes.
- Here **memory is not divided into any fixed size partition. And number of partition is not fixed, process is allocated exactly as much memory as it requires.**
- Initially entire available memory is treated as a single free partition.
- Whenever any process enters in a system a chunk of free memory big enough to fit the process is found and allocated.

- If enough free memory is not available to fit the process, process needs to wait until required memory becomes variable.
- Whenever any process gets terminate, it release the space occupied. If the released space is contiguous to another free partitions are clubbed together into a single free partition.

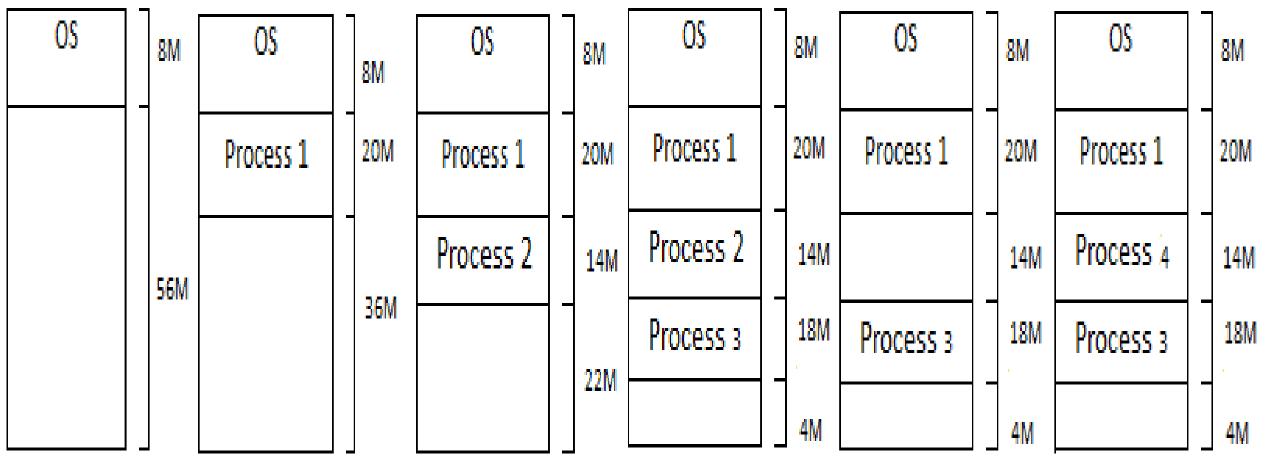


FIG: Multiprogramming with dynamic partition

- Advantage:
  1. Better Utilization of memory
  2. **No internal fragmentation** because process gets only that much memory of its size.
  3. Degree of multiprogramming (means no. of processes running at same time in memory) is not fixed here.
- Disadvantage:

1. Memory is allocated when process enters in the system and released when it terminates. These operations externally result in small holes in the memory. These holes will be small that no any process can be loaded in it, but total size of all holes may be big enough to hold any process. But memory is allocated contiguous here holes can't be used this type of memory wastage is called **External Fragmentation**

2. Scattered (spread) free memory partition can be combined using compaction or De-Fragmentation.

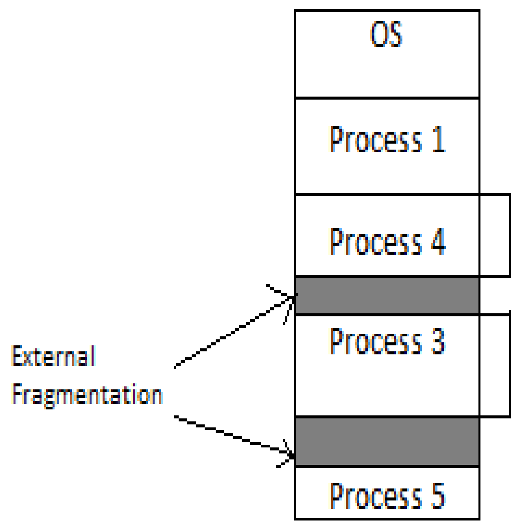
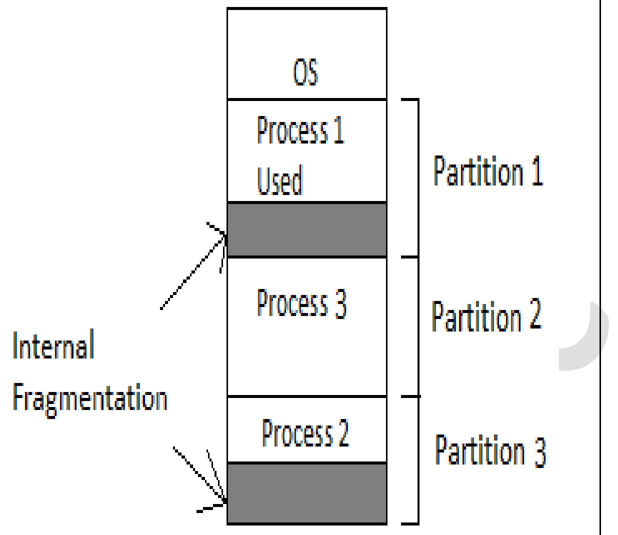
❖ Fragmentation (wastage of memory):

- Memory is allocated when process enters in the system and released when it terminates.
- We can't utilize (100%) full memory due to some problems (fragmentation).
- **Fragmentation refers to unused memory that cannot be allocated to any process.** Means there is free memory available but it can't be used.
- Two kind of fragmentations:
  1. External Fragmentation

2. Internal Fragmentation

❖ Internal and External Fragmentation and compaction

	External Fragmentation	Internal Fragmentation
1.	It refers to wastage of memory between partitions.	It refers to wastage of free memory with in partitions.
2.	It caused by scattered non – contiguous free space.	It caused by the differences between the size of partition and size of process loaded.
3.	External Fragmentation is a severe problem in a contiguous memory allocation method with dynamic partitioning.	Internal Fragmentation is a severe problem in a contiguous memory allocation method with fixed partition.
4.	Memory allocation & de-allocation operations eventually result in small holes in memory. These holes will be so small that no any processes can be loaded in it, but total size of all holes may be big enough to hold process.	When memory de-allocates whole partition may be free, so it can be allocated other process. So there will not any hole in partition. operations eventually result in
5.	As memory is allocated contiguously, here the holes can't be used. This type of memory wastage is called **External Fragmentation**.	Partitions are of fixed size. So any space in a partition not used by a process is wasted this is **Internal Fragmentation**.
6.	Solution: Compaction or de-fragmentation of memory.	Solution: To use dynamic partitioning where process is allocated exactly as much memory as possible.

7.	 OS Process 1 Process 4 Process 3 Process 5 External Fragmentation	 OS Process 1 Used Process 3 Process 2 Partition 1 Partition 2 Partition 3 Internal Fragmentation

❖ Memory relocation and protection mechanism

1. Memory Relocation:

- A process can loaded in any partition in main memory.
- Address in Logical Address Space and Physical Address Space is not a same here.
- Logical Address Space specifies the location of instructions and data within process address space.
- Physical Address Space specifies actual location in main memory.
- Logical Address is required to actually fetch instruction. So whenever there is a reference to any Logical Address it should be converted to physical address this problem is called memory relocation.
- EX: suppose process is loaded at location 1000 in main memory & there is need to fetch instruction located at location 5 in Logical Address Space.
- So Logical Address (5) should be converted to actual Physical Address (1000 + 5) = 1005.

2. Memory Protection:

- Multiprogramming allows running more than one process run simultaneously.
- So memory is shared among processes as well as Operating System.
- All concurrent processes will be in the main memory at the same time, these processes should not have accessed to unauthorized of other processes. Process should read and write data belonging to that process only. This problem is called memory protection.

Solution to both the problem:-

- Two register are used. These registers are called **limit register** and **base register**.
- Limit register are used to store size of process
- Base register are used to store starting location of process in main memory.

- Whenever any process is loaded in main memory, its staring location and size are stored in these two registers are respectively.
- CPU generates logical address start from 0 and goes upto size of process.
- Thus in contiguous memory allocation, the problems of memory relocation and protection can be overcome by maintaining two registers limit register and base register

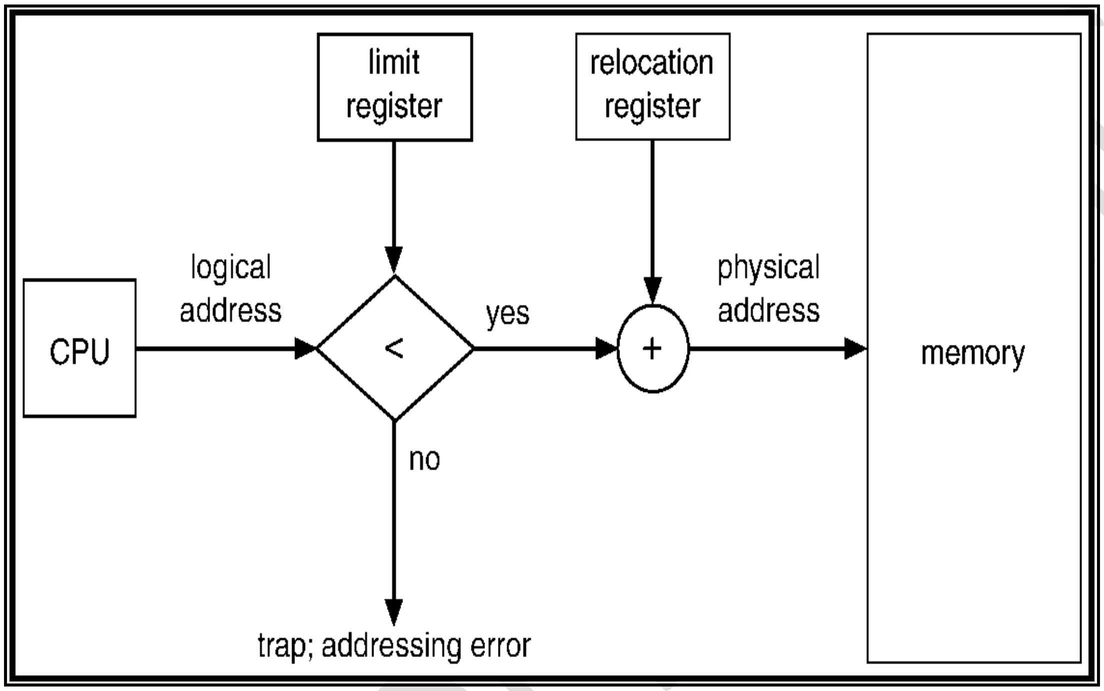


FIG: Memory Relocation & Protection

❖ Allocation techniques – First Fit, Best Fit and Worst Fit

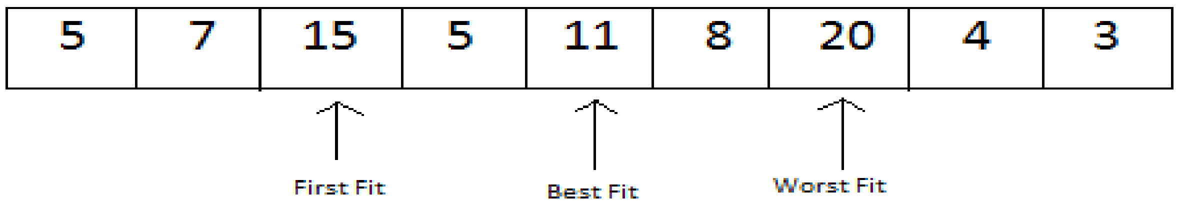
Strategies to select partition:-

- Whenever any process enters in a system required free memory is allocated for that process.
- All free memory, also called holes, is searched to find out chunk of required free memory.
- Four strategies or algorithm are to do this

1. **First Fit:**
   - Search start from starting location of memory.
   - First available hole, which is large enough to hold the processes, is selected for allocation.
   - Search time is small here.
   - Memory loss is higher as very large hole may be selected for small process.
2. **Next Fit:**
   - This is a minor variation over first fit.
   - Searches start from where previous search ended.

- This is avoiding repeating search of memory which has been already allocated; such partition will be mostly at front end of the main memory.

3. **Best Fit:**
   - Entire memory is searched here.
   - The smallest hole, which is large enough to hold the processes, is selected for allocation.
   - Search time is high, as it searches entire memory.
   - Memory loss is less, more sensitive to external fragmentation, as it leaves tiny holes into which no process can fit.
4. **Worst Fit:**
   - Entire memory is searched here.
   - The largest hole, which is large enough to hold the processes, is selected for allocation.
   - It is not sensitive to external fragmentation.
   - This algorithm can be used only with dynamic partition.



❖ Non Contiguous Memory allocation

There are two types of non-contiguous allocation techniques:

1. **Paging**
2. **Segmentation**

❖ Paging Technique

- Logical address space of a process is divided into blocks of fixed size, called **pages.**
- Physical memory is divided into blocks fixed size; called **frames** (page frames).
- The page and frame will be of the same size. This size will be power of 2.It varies between 512 bytes to a few MB.
- **Whenever a process is to be executed, its pages are moved from secondary storage (disk) ,to the available frames in physical memory.**
- Memory allocation refers to,
  - The task of finding free frames in memory,
  - allocating them to pages, and
  - Keeping track of which frame belongs to which page.

- **Page table**
  - Operating system maintains a table, called **page table** for each process.
  - It is indexed by a **page number.**
  - Page table contains information about frame no. in which pages are stored.

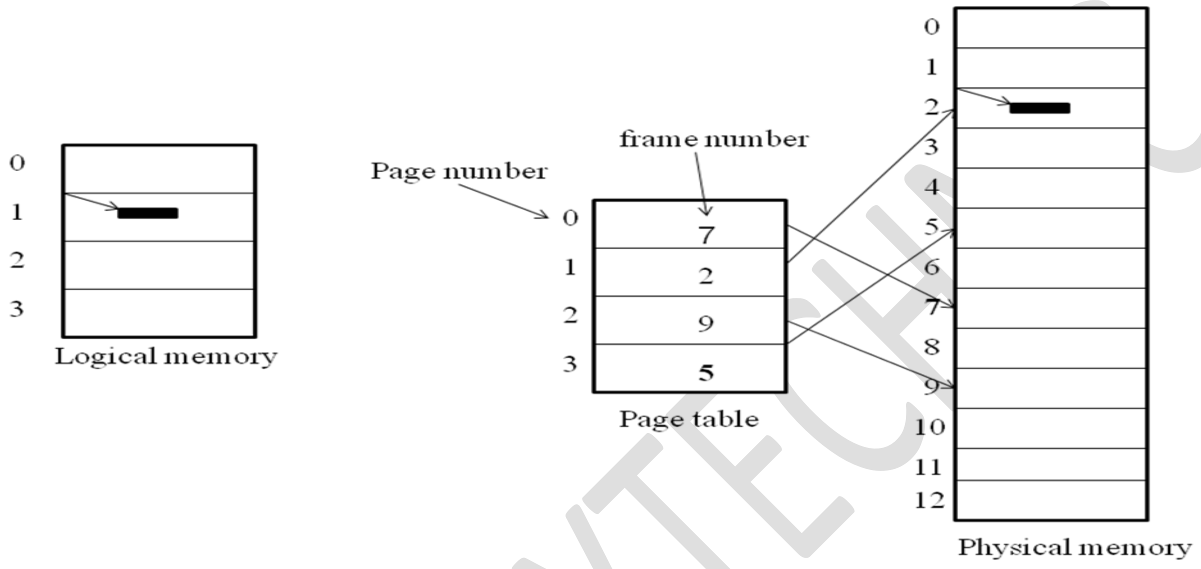


Fig -1: Basic concept of paging

Here, the logical address space of a process, also called logical memory, has been divided into four pages. Physical memory contains total of 11 frames. Also, a page table is given which shows the mapping between pages and frames.

Implementation of Paging technique

- Here, logical address is divided into two parts:
  - **page number,** which gives the number of a page
  - **An offset,** which gives the actual location within a page.

Logical address (L) :

Page no(p)	**offset(d)**

- **Page table is used to implement paging.**
- When process is to be executed, pages **(**logical memory) **are moved to free frames (**main or physical memory).
- During the process execution ,a Logical address (L) is generated by CPU during process execution , to access instruction or data from page table
- Page number **(p)** is used to search any page from page table and if that page is searched then related frame number **(f)** is obtained from page table.
- Frame no. indicates actual frame on physical memory in which page is stored.

Physical address (P):

frame no(f)	**offset(d)**

Following figure explains paging implementation

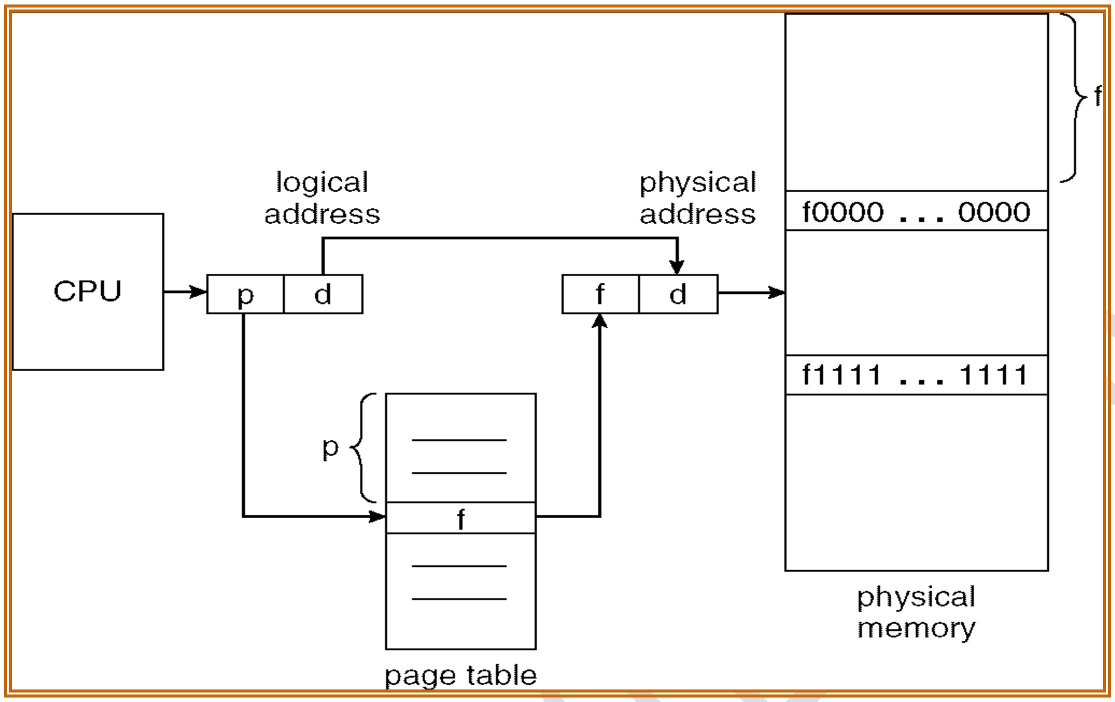


Fig-2: Address translation in paging

Address translation:

1) Page size and frame size = 2^n bytes.
2) **Logical address (L) :**
   a. **Page no (p) : L / 2^n**
   b. **offset (d) : L % 2^n**
3) **physical address (P) : f * 2^n + d**

Example : (No need to give example in GTU exam if not asked specifically)

If page size (or frame size) is 4 byte (**2^2** byte).The logical address generated by the CPU is 11.then calculate physical address.

Ans : page size=frame size =**2^2** byte

Logical address (L) :

1) page no (p) = L / **2^2** = 11 / 4 = 2
2) offset (d) = L % **$2^{2=}$** 11 % 4 = 3

from fig-1 ,we can get frame no 9 for page no. 2

so now **physical address (P) : f *2^{n+d}**

$^=$ 9 * 4 + 3 = **39**

Advantage of paging :

1. Allocation and de-allocation of process is very fast .only list of free frames need to be maintained .No any page algorithm is needed.
2. Swap-in and swap-out operations are very fast.(because page and frame size are same ,data moves easily)

3. Here, available memory is divided into fixed sized block and no wastage of memory between partitions, so there is no External fragmentation.

Disadvantage of paging:

1. Additional memory reference is required to read information from page table .Two memory access required one for page table and second for instruction or data.
2. If process is of large size then page table of that process also have large size. It may be too large to keep such page table in memory.
3. Here, it may be possible that space within partitions (page) remain unused by process because of differ in page size and part of process loaded in that page . So paging suffers from internal fragmentation.

❖ Segmentation

- Logical address is collection of code, data and stack. **Code** can be main function, user define function library function**. Data** can be local and global variable, array, symbol table or other data structure.
- Logical address space of a process is divided into blocks of varying size, called ***segment.*** Each segment contains a logical unit of a process.
- Logical unit can be main function, user define function, procedure, stack, array, symbol table.
- Each segment can be considered as a independent address space.
- Segment consists of linear sequence of address starting from '0' to some maximum limit. All segments are of varying lengths and length depends upon the size of a logical unit
- All segments are given unique numbers to identify them.
- When process is to be executed its segments are moved from secondary storage to main memory.
- OS maintains **segment table** for each process. **Segment table** contains two information about each process.
    1. **Size of segment**
    2. **Offset** , which gives actual location within segment

Implementation of segment:

- Segment table is used to implement segmentation .Each entry of the segment table has a segment **base** and segment **limit.**
    - A segment **base** contains the starting physical address where the segment resides in memory.
    - The segment **limit** specifies length of segment.

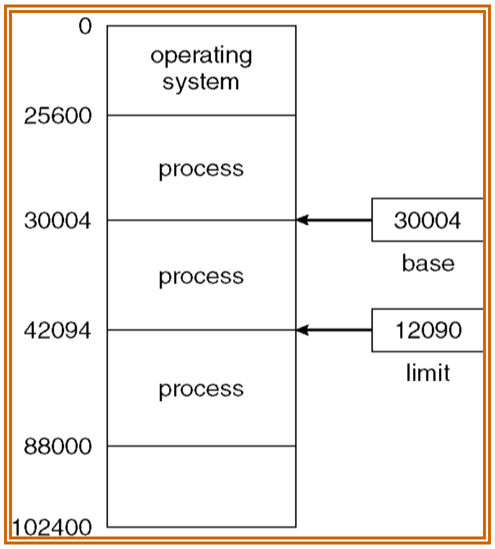
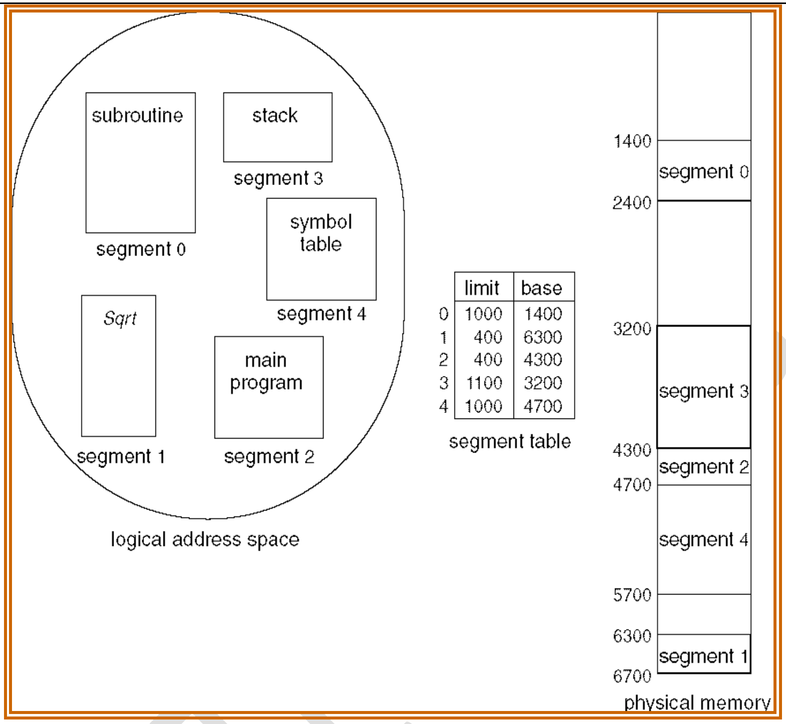
	

Fig-3 (a) :base and limit of segment	**Fig-3 (b)** : basic concept of paging

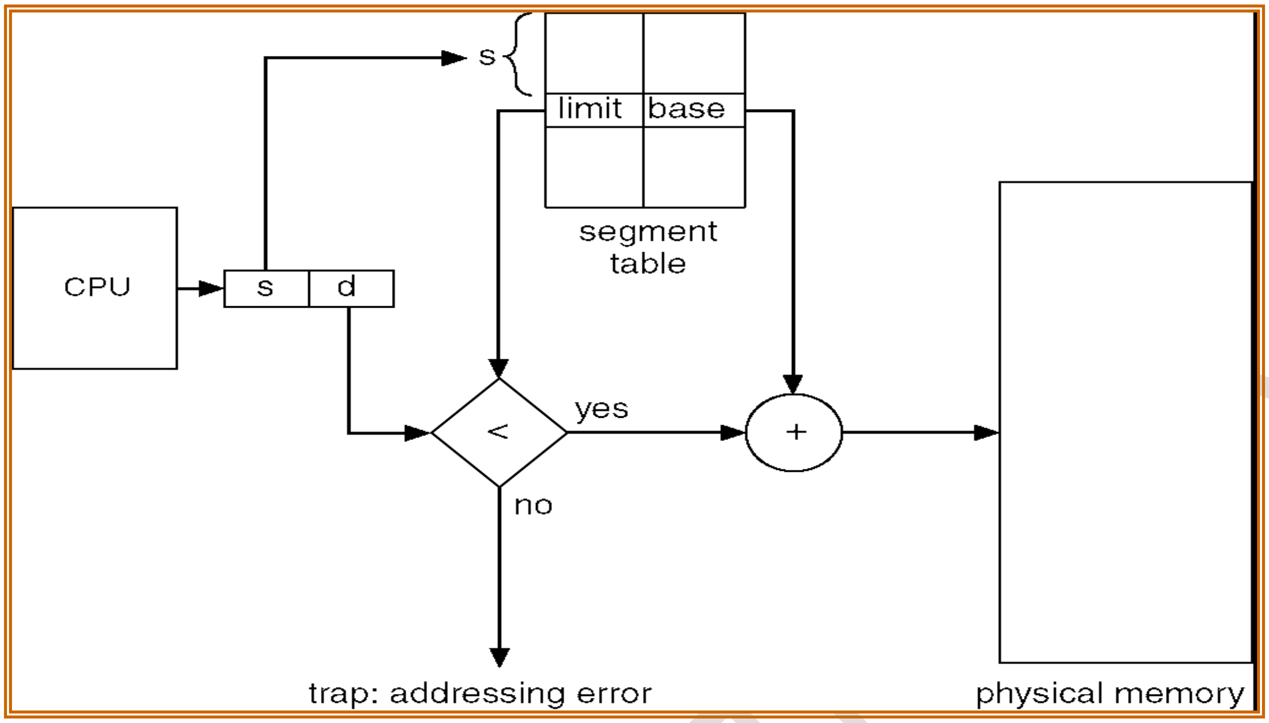
- A pair of **base** and **limit** registers define the logical address space and stores segment base and segment limit.

Logical address:

- During the process execution ,a CPU generates a logical address (L) to access instruction or data from a particular location.
- Logical address is divided into two parts as shown in figure:

Segment no.(S)	Offset(d)

- The segment number is used as index into segment table. The offset (d) of the logical address must be between 0 and the segment limit.
- If offset is within range of limit register value than offset is added to the segment base to produce the physical address.
- If offset is not within limit than illegal address error will be generated.



Example : (refer Fig-3 (b))

- Suppose CPU has generates some logical address, and is given form of(segment no , offset)-
  L=(2,125)
- To get physical address related to logical address, segment table will be searched to get entry of segment no 2.
  - For segment 2 ,**base is 4300** and **limit is 400,**
  - **Offset compared with limit value (125 < 400)**

So, offset is valid

Physical address (P) = offset + base

= 125 + 4300=4425

Example 2

- Suppose CPU has generates some logical address, and is given form of(segment no , offset)-
  L=(2,410)
- To get physical address related to logical address, segment table will be searched to get entry of segment no 2.
  - For segment 2 ,**base is 4300** and **limit is 410,**
  - **Offset compared with limit value (125 < 410)**

So, offset is not valid. so physical address can not be generated.

Advantage:

- All segments are independent from each other, so segments can grow and shrink without affecting other segments.
- If procedure in one segment is modified and recompiled, no other segments need to be changed or recompiled.

- Sharing of procedure and data between various processes is simple.
- Different segment of a single process can be given different kind of protection. One segment can be read only ,while another can be writable.
- No internal fragmentation because segments are allocated exactly as much memory as required.

Disadvantage:

- It is still expensive to allocate contiguous free memory to segments.
- External fragmentation is possible, which requires memory de-fragmentation or compaction.

❖ Page replacement algorithm – FIFO, LRU

There are various page replacement algorithms. Each algorithm has a different method by which the pages can be replaced.

1. **Optimal Page Replacement algorithm** → this algorithms replaces the page which will not be referred for so long in future. Although it cannot be practically implementable but it can be used as a benchmark. Other algorithms are compared to this in terms of optimality.
2. **Least recent used (LRU) page replacement algorithm** → this algorithm replaces the page which has not been referred for a long time. This algorithm is just opposite to the optimal page replacement algorithm. In this, we look at the past instead of staring at future.
3. **FIFO** → in this algorithm, a queue is maintained. The page which is assigned the frame first will be replaced first. In other words, the page which resides at the rare end of the queue will be replaced on the every page fault.

Consider a main memory with five page frames and the following sequence of page references: 3, 8, 2, 3, 9, 1, 6, 3, 8, 9, 3, 6, 2, 1, 3. which one of the following is true with respect to page replacement policies First-In-First-out (FIFO) and Least Recently Used (LRU)?

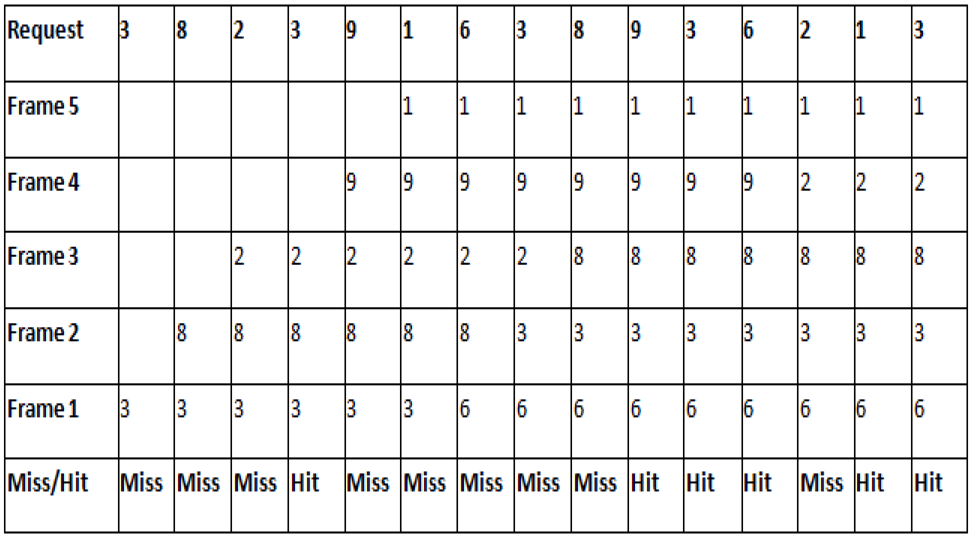
A. Both incur the same number of page faults
B. FIFO incurs 2 more page faults than LRU
C. C. LRU incurs 2 more page faults than FIFO
D. D. FIFO incurs 1 more page faults than LRU

Solution:

Number of frames = 5

FIFO

According to FIFO, the page which first comes in the memory will first goes out.



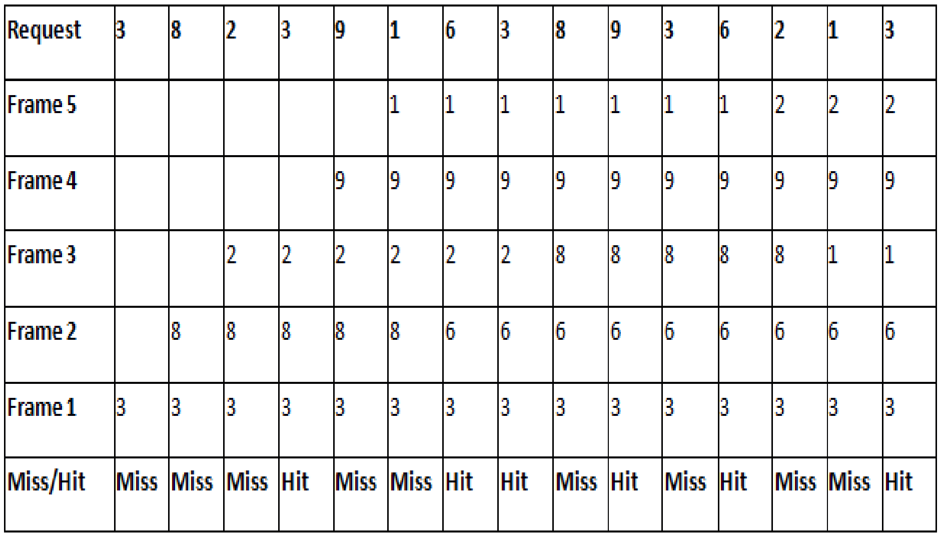
Request	3	8	2	3	9	1	6	3	8	9	3	6	2	1	3
Frame 5						1	1	1	1	1	1	1	1	1	1
Frame 4					9	9	9	9	9	9	9	9	2	2	2
Frame 3			2	2	2	2	2	2	8	8	8	8	8	8	8
Frame 2		8	8	8	8	8	8	3	3	3	3	3	3	3	3
Frame 1	3	3	3	3	3	3	6	6	6	6	6	6	6	6	6
Miss/Hit	Miss	Miss	Miss	Hit	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Miss	Hit	Hit

Number of Page Faults = 9

Number of hits = 6

LRU

According to LRU, the page which has not been requested for a long time will get replaced with the new one.



Request	3	8	2	3	9	1	6	3	8	9	3	6	2	1	3
Frame 5						1	1	1	1	1	1	1	2	2	2
Frame 4					9	9	9	9	9	9	9	9	9	9	9
Frame 3			2	2	2	2	2	2	8	8	8	8	8	1	1
Frame 2		8	8	8	8	8	6	6	6	6	6	6	6	6	6
Frame 1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Miss/Hit	Miss	Miss	Miss	Hit	Miss	Miss	Hit	Hit	Miss	Hit	Miss	Hit	Miss	Miss	Hit

Number of Page Faults = 9

Number of Hits = 6

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. FIFO Page Replacement Algorithm
2. LRU Page Replacement Algorithm

LRU Page Replacement Algorithm

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 LRU = 6

FIFO Page Replacement Algorithm

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