

COURSE NAME : OPERATING SYSTEMS

(COURSE CODE :22516)

COURSE TEACHER : MRS.T.H.GAVHANE

UNIT NO 5 (14Marks)

MEMORY MANAGEMNET

Course Outcome: Calculate efficiency of different memory management techniques

○ UNIT OUTCOME

5a	Describe the working of specified memory management function
5b	Explain characteristics of given memory management techniques
5c	Write algorithm for the given page replacement technique.
5d	Calculate the page fault for the given page reference string

5.1 Basic Memory Management – Partitioning, Fixed and Variable, Free space management techniques Bitmap, Linked List

❖ What is 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.

❖ What is Partitioning?

Memory partitioning is the system by which the memory of a computer system is divided into sections for use by the resident programs. These memory divisions are known as partitions

Fixed Partitioning

The earliest and one of the simplest technique which can be used to load more than one processes into the main memory is Fixed partitioning or Contiguous memory allocation.

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 cons of using this technique.

1. Internal Fragmentation

If the size of the process is lesser than the total size of the partition then some size of the partition gets wasted and remains unused. This is wastage of the memory and is 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 is 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.

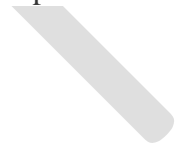
As shown in the image below, the remaining 1 MB space of each partition cannot be used as a unit to store a 4 MB process. Despite of the fact that the sufficient space is available to load the process, the process will not be loaded.

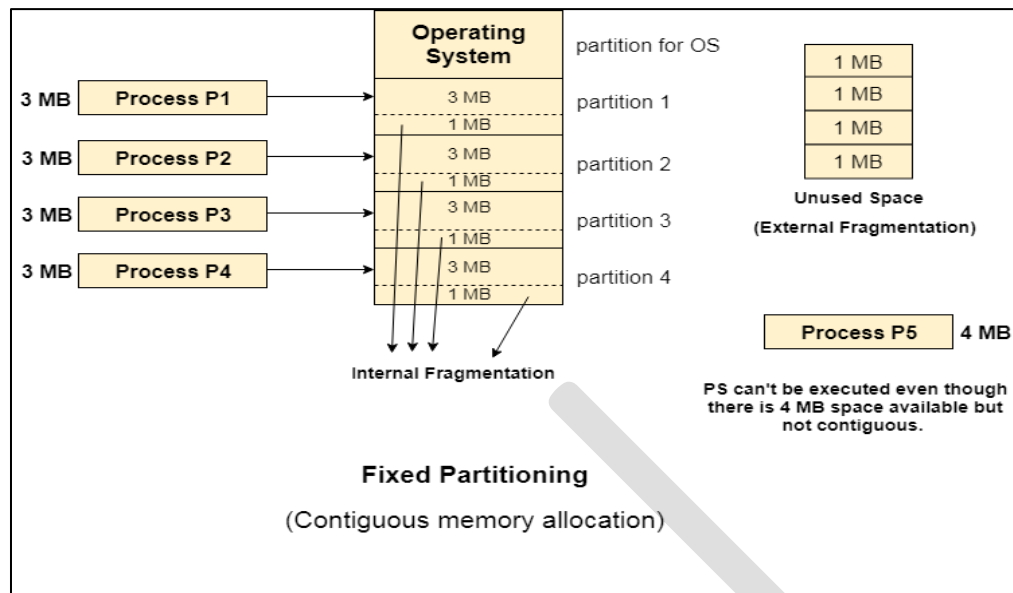
3. Limitation on the size of the process

If the process size is larger than the size of the maximum sized partition then that process cannot be loaded into the 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.

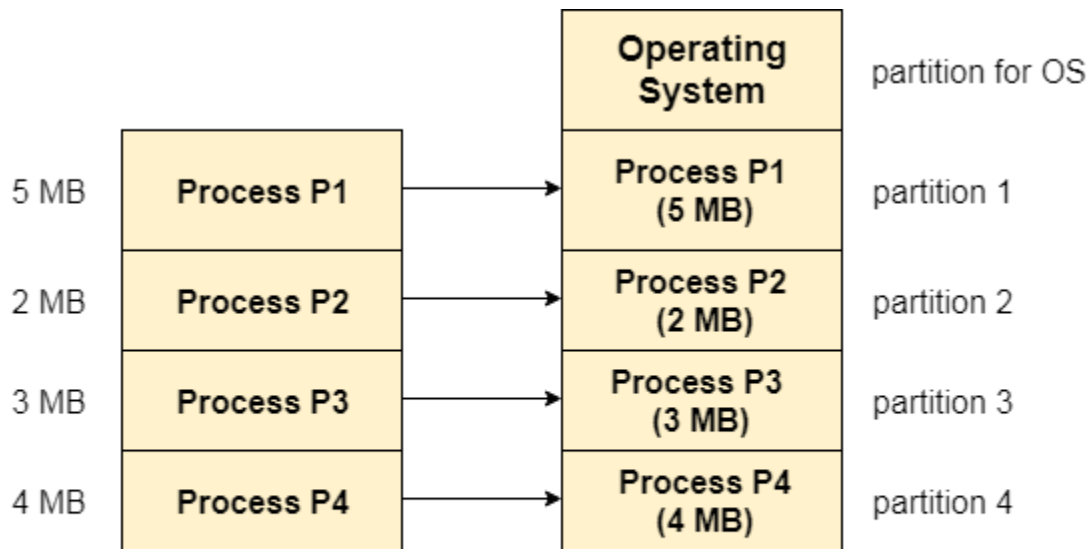




Dynamic Partitioning

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.



Dynamic Partitioning

(Process Size = Partition Size)

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

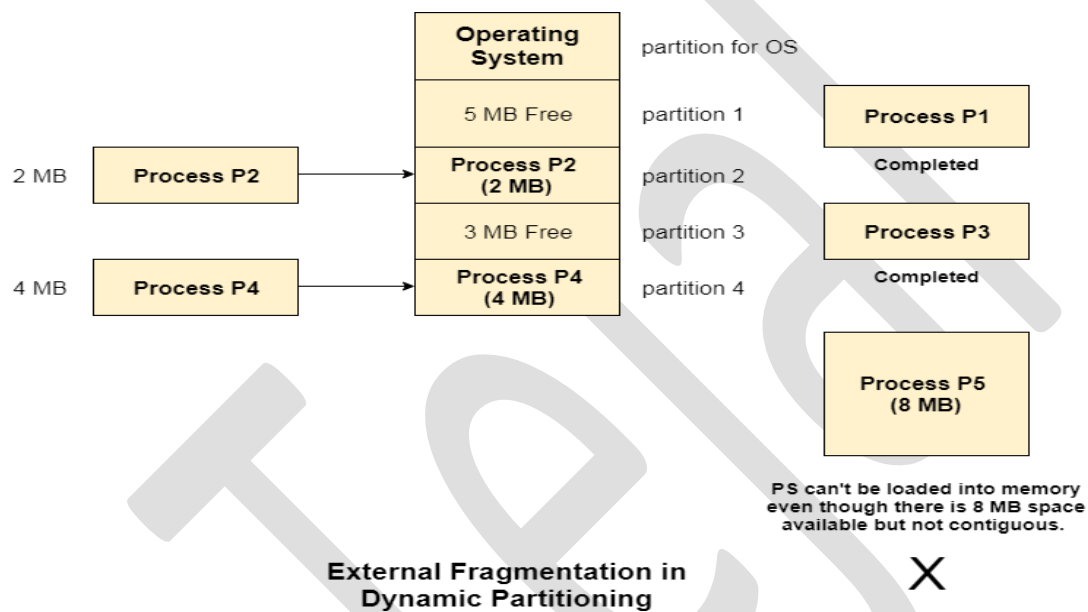
External Fragmentation

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.



Complex Memory Allocation

In Fixed partitioning, the list of partitions is made once and will never change but in dynamic partitioning, the allocation and deallocation 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 deallocation 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.

❖ Algorithms for Partitioning :

There are various algorithms which are implemented by the Operating System in order to find out the holes in the linked list and allocate them to the processes.

The explanation about each of the algorithm is given below.

1. First Fit Algorithm

First Fit algorithm scans the linked list and whenever it finds the first big enough hole to store a process, it stops scanning and load the process into that hole. This procedure produces two partitions. Out of them, one partition will be a hole while the other partition will store the process.

First Fit algorithm maintains the linked list according to the increasing order of starting index. This is the simplest to implement among all the algorithms and produces bigger holes as compare to the other algorithms.

2. Next Fit Algorithm

Next Fit algorithm is similar to First Fit algorithm except the fact that, Next fit scans the linked list from the node where it previously allocated a hole.

Next fit doesn't scan the whole list, it starts scanning the list from the next node. The idea behind the next fit is the fact that the list has been scanned once therefore the probability of finding the hole is larger in the remaining part of the list.

Experiments over the algorithm have shown that the next fit is not better then the first fit. So it is not being used these days in most of the cases.

3. Best Fit Algorithm

The Best Fit algorithm tries to find out the smallest hole possible in the list that can accommodate the size requirement of the process.

Using Best Fit has some disadvantages.

1. 1. It is slower because it scans the entire list every time and tries to find out the smallest hole which can satisfy the requirement the process.
2. Due to the fact that the difference between the whole size and the process size is very small, the holes produced will be as small as it cannot be used to load any process and therefore it remains useless. Despite of the fact that the name of the algorithm is best fit, It is not the best algorithm among all.
- 3.

4. Worst Fit Algorithm

The worst fit algorithm scans the entire list every time and tries to find out the biggest hole in the list which can fulfill the requirement of the process.

Despite of the fact that this algorithm produces the larger holes to load the other processes, this is not the better approach due to the fact that it is slower because it searches the entire list every time again and again.

5. Quick Fit Algorithm

The quick fit algorithm suggests maintaining the different lists of frequently used sizes. Although, it is not practically suggestible because the procedure takes so much time to create the different lists and then expending the holes to load a process.

The first fit algorithm is **the best algorithm** among all because

1. It takes lesser time compare to the other algorithms.
2. It produces bigger holes that can be used to load other processes later on.
3. It is easiest to implement.

❖ Free space management techniques : Bitmap, Linked List

A file system is responsible to allocate the free blocks to the file therefore it has to keep track of all the free blocks present in the disk. There are mainly two approaches by using which, the free blocks in the disk are managed.

1. Bit Vector

In this approach, the free space list is implemented as a bit map vector. It contains the number of bits where each bit represents each block.

If the block is empty then the bit is 1 otherwise it is 0. Initially all the blocks are empty therefore each bit in the bit map vector contains 1.

As the space allocation proceeds, the file system starts allocating blocks to the files and setting the respective bit to 0.

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

Assume the following are free. Rest are allocated:

2,3,4,5,9,10,13

Blocks →	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Bits →	0	0	1	1	1	1	0	0	0	1	1	0	0	1

Advantages:

- This technique is relatively simple.
- This technique is very efficient to find the free space on the disk.

Disadvantages:

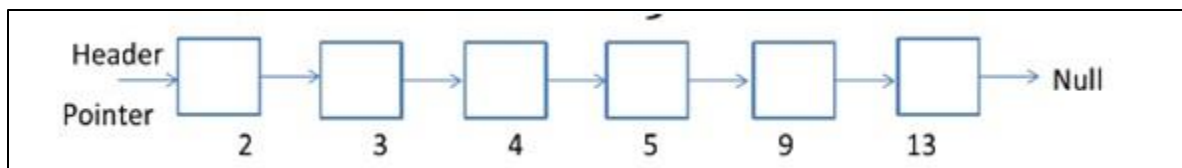
- This technique requires a special hardware support to find the first 1 in a word it is not 0.
- This technique is not useful for the larger disks.

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 bitmap would be: 00111100111110001100000011100000

2. Linked List

It is another approach for free space management. This approach suggests linking together all the free blocks and keeping a pointer in the cache which points to the first free block.

Therefore, all the free blocks on the disks will be linked together with a pointer. Whenever a block gets allocated, its previous free block will be linked to its next free block.



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.

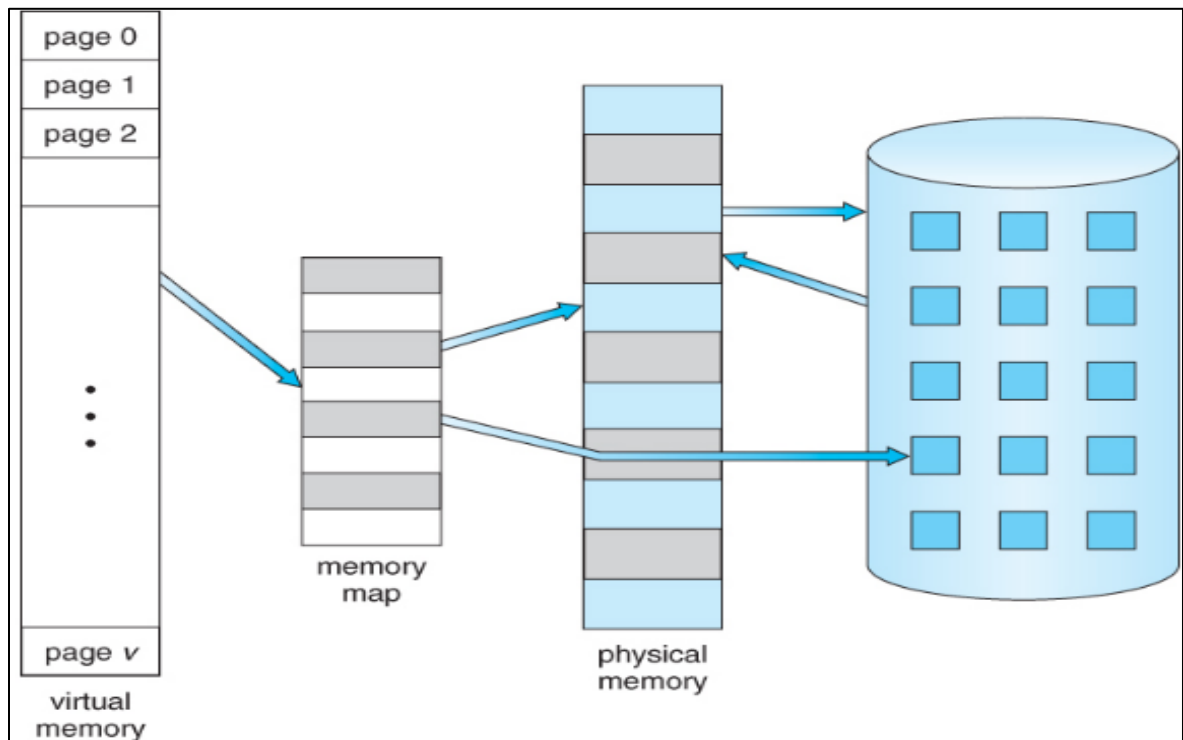
Disadvantages:

- 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.

5.2 Virtual Memory – Introduction to Paging, Segmentation, Fragmentation and Page Fault

❖ What is Virtual Memory?

- Virtual Memory is a storage scheme that provides user an illusion of having a very big main memory. This is done by treating a part of secondary memory as the main memory.
- In this scheme, User can load the bigger size processes than the available main memory by having the illusion that the memory is available to load the process.
- Instead of loading one big process in the main memory, the Operating System loads the different parts of more than one process in the main memory.
- By doing this, the degree of multiprogramming will be increased and therefore, the CPU utilization will also be increased.



Advantages of Virtual Memory

1. The degree of Multiprogramming will be increased.
2. User can run large application with less real RAM.
3. There is no need to buy more memory RAMs.

Disadvantages of Virtual Memory

1. The system becomes slower since swapping takes time.
2. It takes more time in switching between applications.
3. The user will have the lesser hard disk space for its use.

❖ Paging :

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

The main idea behind the paging is to divide each process in the form of pages. The main memory will also be divided in the form of frames.

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

Pages of the process are brought into the main memory only when they are required otherwise they reside in the secondary storage.

Different operating system defines different frame sizes. The sizes of each frame must be equal. Considering the fact that the pages are mapped to the frames in Paging, page size needs to be as same as frame size.

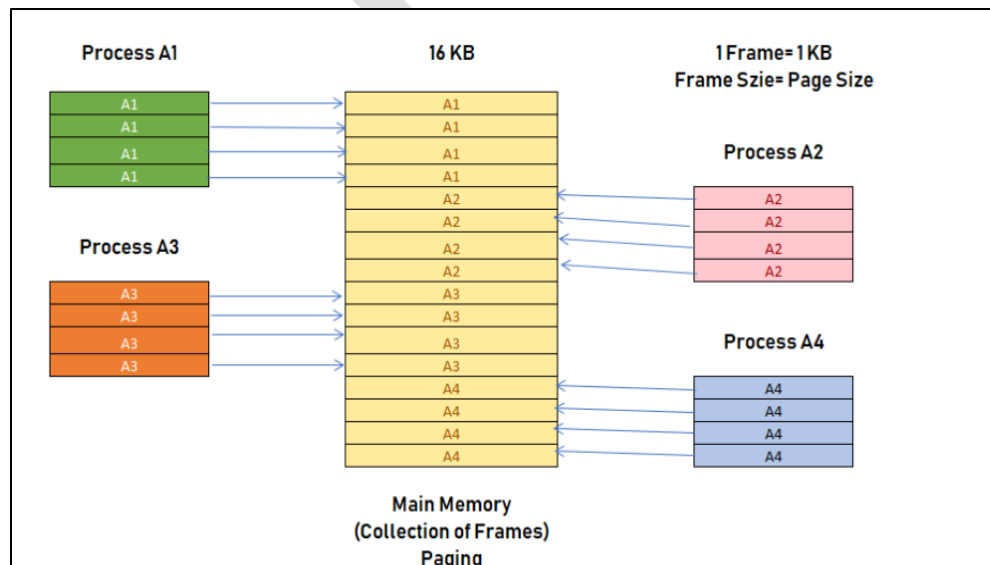
Example

Let us consider the main memory size 16 Kb and Frame size is 1 KB therefore the main memory will be divided into the collection of 16 frames of 1 KB each.

There are 4 processes in the system that is P1, P2, P3 and P4 of 4 KB each. Each process is divided into pages of 1 KB each so that one page can be stored in one frame.

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

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



Segmentation :

In Operating Systems, Segmentation is a memory management technique in which the memory is divided into the variable size parts. Each part is known as a segment which can be allocated to a process.

The details about each segment are stored in a table called a segment table. Segment table is stored in one (or many) of the segments.

Segment table contains mainly two information about segment:

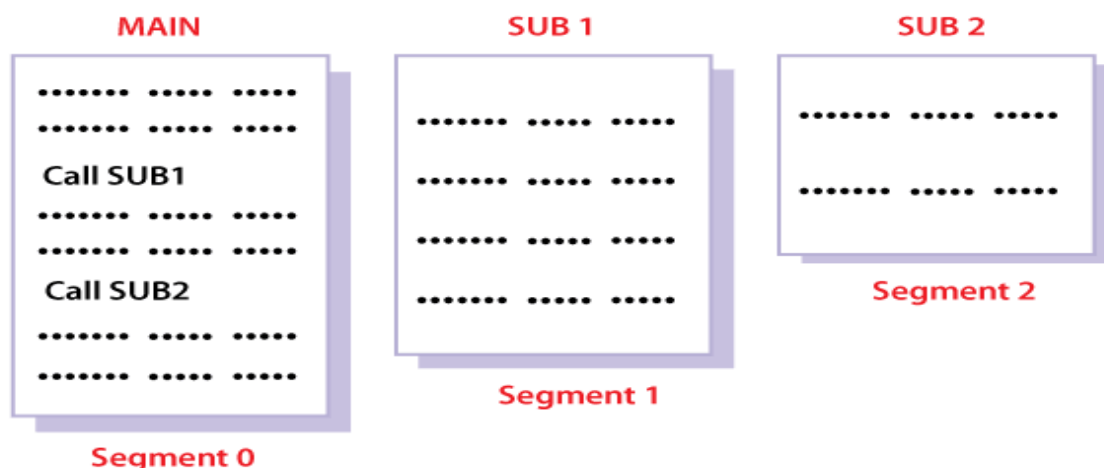
1. Base: It is the base address of the segment
2. Limit: It is the length of the segment.

Why Segmentation is required?

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

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

It is better to have segmentation which divides the process into the segments. Each segment contains the same type of functions such as the main function can be included in one segment and the library functions can be included in the other segment.



Translation of Logical address into physical address by segment table

CPU generates a logical address which contains two parts:

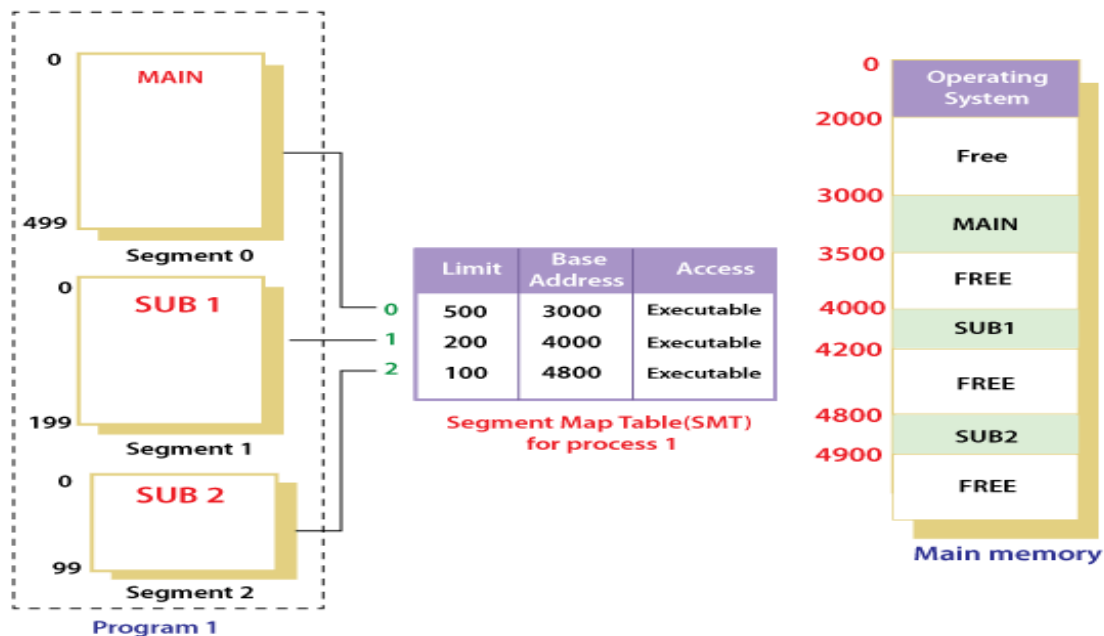
1. Segment Number
2. Offset

For Example:

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

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

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



With the help of segment map tables and hardware assistance, the operating system can easily translate a logical address into physical address on execution of a program.

The **Segment number** is mapped to the segment table. The limit of the respective segment is compared with the offset. If the offset is less than the limit then the address is valid otherwise it throws an error as the address is invalid.

In the case of valid addresses, the base address of the segment is added to the offset to get the physical address of the actual word in the main memory.

The above figure shows how address translation is done in case of segmentation.

Advantages of Segmentation :

1. No internal fragmentation
2. Average Segment Size is larger than the actual page size.
3. Less overhead
4. It is easier to relocate segments than entire address space.
5. The segment table is of lesser size as compared to the page table in paging.

Disadvantages :

1. It can have external fragmentation.
2. it is difficult to allocate contiguous memory to variable sized partition.
3. Costly memory management algorithms.

Fragmentation :

Fragmentation is an unwanted problem in the operating system in which the processes are loaded and unloaded from memory, and free memory space is fragmented. Processes can't be assigned to memory blocks due to their small size, and the memory blocks stay unused.

Contiguous memory allocation allocates space to processes whenever the processes enter **RAM**. These **RAM** spaces are divided either by fixed partitioning or by dynamic partitioning. As the process is loaded and unloaded from memory, these areas are fragmented into small pieces of memory that cannot be allocated to coming processes.

In this article, you will learn about fragmentation and its types.

What is Fragmentation?

Fragmentation is an unwanted problem in the operating system in which the processes are loaded and unloaded from memory, and free memory space is fragmented. Processes can't be assigned to memory blocks due to their small size, and the memory blocks stay unused. It is also necessary to understand that as programs are loaded and deleted from memory, they generate free space or a hole in the memory. These small blocks cannot be allotted to new arriving processes, resulting in inefficient memory use.

The conditions of fragmentation depend on the memory allocation system. As the process is loaded and unloaded from memory, these areas are fragmented into small pieces of memory that cannot be allocated to incoming processes. It is called **fragmentation**.

Causes of Fragmentation

User processes are loaded and unloaded from the main memory, and processes are kept in memory blocks in the main memory. Many spaces remain after process loading and swapping that another process cannot load due to their size. Main memory is available, but its space is insufficient to load another process because of the dynamical allocation of main memory processes.

Types of Fragmentation

There are mainly two types of fragmentation in the operating system. These are as follows:

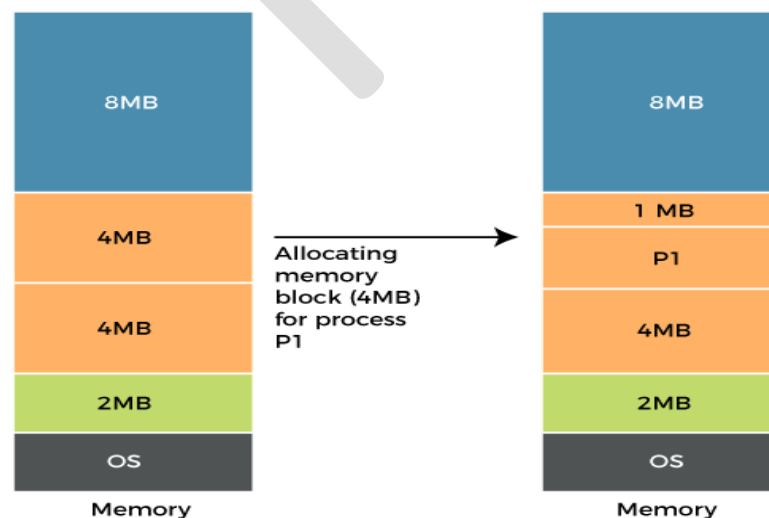
1. **Internal Fragmentation**
2. **External Fragmentation**

Internal Fragmentation

When a process is allocated to a memory block, and if the process is smaller than the amount of memory requested, a free space is created in the given memory block. Due to this, the free space of the memory block is unused, which causes **internal** fragmentation.

For Example:

Assume that memory allocation in RAM is done using fixed partitioning (i.e., memory blocks of fixed sizes). **2MB**, **4MB**, **4MB**, and **8MB** are the available sizes. The Operating System uses a part of this RAM.



Let's suppose a process **P1** with a size of **3MB** arrives and is given a memory block of **4MB**. As a result, the **1MB** of free space in this block is unused and cannot be used to allocate memory to another process. It is known as **internal fragmentation**.

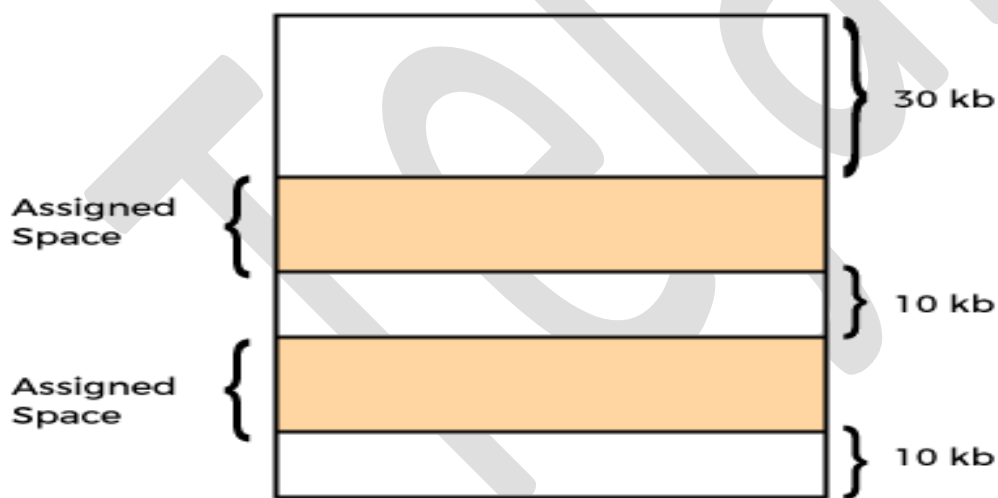
How to avoid internal fragmentation?

The problem of internal fragmentation may arise due to the fixed sizes of the memory blocks. It may be solved by assigning space to the process via dynamic partitioning. Dynamic partitioning allocates only the amount of space requested by the process. As a result, there is no internal fragmentation.

External Fragmentation

External fragmentation happens when a dynamic memory allocation method allocates some memory but leaves a small amount of memory unusable. The quantity of available memory is substantially reduced if there is too much external fragmentation. There is enough memory space to complete a request, but it is not contiguous. It's known as **external** fragmentation.

For Example:



Process 05 needs 45kb memory space

Let's take the example of external fragmentation. In the above diagram, you can see that there is sufficient space (**50 KB**) to run a process (**05**) (**need 45KB**), but the memory is not contiguous. You can use compaction, paging, and segmentation to use the free space to execute a process.

How to remove external fragmentation?

This problem occurs when you allocate RAM to processes continuously. It is done in paging and segmentation, where memory is allocated to processes non-contiguously. As a result, if you remove this condition, external fragmentation may be decreased.

Compaction is another method for removing external fragmentation. External fragmentation may be decreased when dynamic partitioning is used for memory allocation by combining all free memory into a single large block. The larger memory block is used to allocate space based on the requirements of the new processes. This method is also known as defragmentation.

Advantages and disadvantages of fragmentation

There are various advantages and disadvantages of fragmentation. Some of them are as follows:

Advantages

There are various advantages of fragmentation. Some of them are as follows:

Fast Data Writes

Data write in a system that supports data fragmentation may be faster than reorganizing data storage to enable contiguous data writes.

Fewer Failures

If there is insufficient sequential space in a system that does not support fragmentation, the write will fail.

Storage Optimization

A fragmented system might potentially make better use of a storage device by utilizing every available storage block.

Disadvantages

There are various disadvantages of fragmentation. Some of them are as follows:

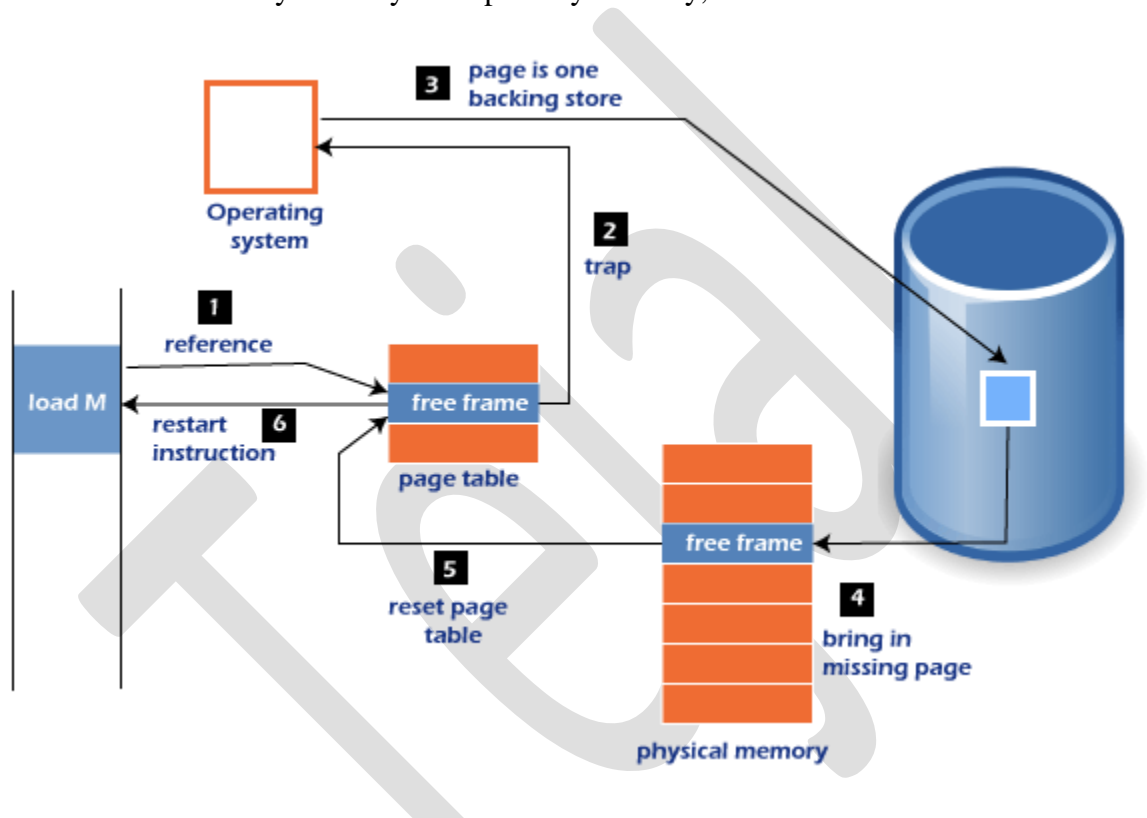
Need for regular defragmentation

A more fragmented storage device's performance will degrade with time, necessitating the requirement for time-consuming defragmentation operations.

Page Fault :

What is Page Fault in Operating System?

Page faults dominate more like an **error**. A page fault will happen if a program tries to access a piece of memory that does not exist in physical memory (main memory). The fault specifies the operating system to trace all data into virtual memory management and then relocate it from secondary memory to its primary memory, such as a hard disk.



A page fault trap occurs if the requested page is not loaded into memory. The page fault primarily causes an exception, which is used to notify the operating system to retrieve the "**pages**" from virtual memory to continue operation. Once all of the data has been placed into physical memory, the program resumes normal operation. The Page fault process occurs in the background, and thus the user is unaware of it.

1. The computer's hardware track to the kernel and the program counter is often saved on the stack. The CPU registers hold information about the current state of instruction.
2. An assembly program is started, which saves the general registers and other volatile data to prevent the Operating system from destroying it.

Page Fault Handling

A Page Fault happens when you access a page that has been marked as invalid. The paging hardware would notice that the invalid bit is set while translating the address across the page table, which will cause an operating system trap. The trap is caused primarily by the OS's failure to load the needed page into memory.

Now, let's understand the procedure of page fault handling in the OS:

1. Firstly, an internal table for this process to assess whether the reference was valid or invalid memory access.
2. If the reference becomes invalid, the system process would be terminated. Otherwise, the page will be paged in.
3. After that, the free-frame list finds the free frame in the system.
4. Now, the disk operation would be scheduled to get the required page from the disk.
5. When the I/O operation is completed, the process's page table will be updated with a new frame number, and the invalid bit will be changed. Now, it is a valid page reference.
6. If any page fault is found, restart these steps from starting.

Page Fault Terminology

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

1. Page Hit

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

2. Page Miss

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

3. Page Fault Time

The time it takes to get a page from secondary memory and recover it from the main memory after loading the required page is known as **"PAGE FAULT TIME"**.

4. Page Fault Delay

The rate at which threads locate page faults in memory is referred to as the "**PAGE FAULT RATE**". The page fault rate is measured per second.

5. Hard Page Fault

If a required page exists in the hard disk's page file, it is referred to as a "**HARD PAGE FAULT**".

6. Soft Page Fault

If a required page is not located on the hard disk but is found somewhere else in memory, it is referred to as a "**SOFT PAGE FAULT**".

7. Minor Page Fault

If a process needs data and that data exists in memory but is being allotted to another process at the same moment, it is referred to as a "**MINOR PAGE FAULT**".

5.3 Page Replacement algorithm- FIFO,LRU and Optimal

The **page replacement algorithm** decides which memory page is to be replaced. The process of replacement is sometimes called swap out or write to disk. Page replacement is done when the requested page is not found in the main memory (page fault).

Types of Page Replacement Algorithms :

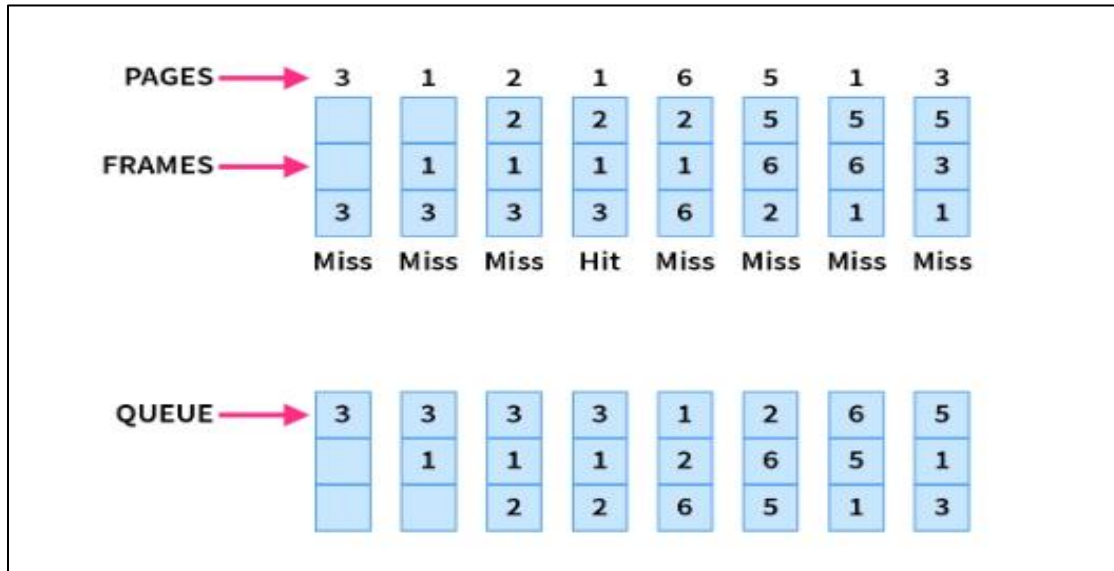
- 1. FIFO(first in first out)**
- 2. Optimal Page replacement**
- 3. LRU(least recently used)**

1. FIFO (First In First Out) :

FIFO algorithm is the simplest of all the page replacement algorithms. In this, we maintain a queue of all the pages that are in the memory currently. The oldest page in the memory is at the front-end of the queue and the most recent page is at the back or rear-end of the queue.

Whenever a page fault occurs, the operating system looks at the front-end of the queue to know the page to be replaced by the newly requested page. It also adds this newly requested page at the rear-end and removes the oldest page from the front-end of the queue.

Example: Consider the page reference string as 3, 1, 2, 1, 6, 5, 1, 3 with 3-page frames. Let's try to find the number of page faults:



- Initially, all of the slots are empty so page faults occur at 3,1,2.

Page faults = 3

- When page 1 comes, it is in the memory so no page fault occurs.

Page faults = 3

- When page 6 comes, it is not present and a page fault occurs. Since there are no empty slots, we **remove the front of the queue, i.e 3.**

Page faults = 4

- When page 5 comes, it is also not present and hence a page fault occurs. The front of the queue i.e **1 is removed.**

Page faults = 5

- When page 1 comes, it is not found in memory and again a page fault occurs. The front of the queue i.e **2 is removed.**

Page faults = 6

- When page 3 comes, it is again not found in memory, a page fault occurs, and **page 6 is removed** being on top of the queue

Total page faults = 7

Belady's anomaly: Generally if we increase the number of frames in the memory, the number of page faults should decrease due to obvious reasons. Belady's anomaly refers to the phenomena where increasing the number of frames in memory, increases the page faults as well.

Advantages

- Simple to understand and implement
- Does not cause more overhead

Disadvantages

- Poor performance
- Doesn't use the frequency of the last used time and just simply replaces the oldest page.
- Suffers from Belady's anomaly.

2. Optimal Page Replacement

Optimal page replacement is the best page replacement algorithm as this algorithm results in the least number of page faults. In this algorithm, the pages are replaced with the ones that will not be used for the longest duration of time in the future. In simple terms, the pages that will be referred farthest in the future are replaced in this algorithm.

Example:

Let's take the same page reference string 3, 1, 2, 1, 6, 5, 1, 3 with 3-page frames as we saw in FIFO. This also helps you understand how Optimal Page replacement works the best.

PAGES →	3	1	2	1	6	5	1	3
FRAMES →			2	2	6	5	5	5
		1	1	1	1	1	1	1
	3	3	3	3	3	3	3	3
	Miss	Miss	Miss	Hit	Miss	Miss	Hit	Hit

- Initially, since all the slots are empty, **pages 3, 1, 2** cause a page fault and take the empty slots.

Page faults = 3

- When page 1 comes, it is in the memory and no page fault occurs.

Page faults = 3

- When page 6 comes, it is not in the memory, so a page fault occurs and **2 is removed** as it is not going to be used again.

Page faults = 4

- When page 5 comes, it is also not in the memory and causes a page fault. Similar to above **6 is removed** as it is not going to be used again.

page faults = 5

- When page 1 and page 3 come, they are in the memory so no page fault occurs.

Total page faults = 5

Advantages

- Excellent efficiency
- Less complexity
- Easy to use and understand
- Simple data structures can be used to implement
- Used as the benchmark for other algorithms

Disadvantages

- More time consuming
- Difficult for error handling
- Need future awareness of the programs, which is not possible every time

3. Least Recently Used (LRU) Page Replacement Algorithm

The least recently used page replacement algorithm keeps the track of usage of pages over a period of time. This algorithm works on the basis of the **principle of locality of a reference** which states that a program has a tendency to access the same set of memory locations repetitively over a short period of time. So pages that have been used heavily in the past are most likely to be used heavily in the future also.

In this algorithm, **when a page fault occurs, then the page that has not been used for the longest duration of time is replaced by the newly requested page.**

Example: Let's see the performance of the LRU on the same reference string of 3, 1, 2, 1, 6, 5, 1, 3 with 3-page frames:

LRU page replacement								
PAGES	3	1	2	1	6	5	1	3
FRAMES			2	2	2	2	1	1
		1	1	1	1	5	5	5
	3	3	3	3	6	6	6	3
	Miss	Miss	Miss	Hit	Miss	Miss	Miss	Miss

- Initially, since all the slots are empty, **pages 3, 1, 2** cause a page fault and take the empty slots.

Page faults = 3

- When page 1 comes, it is in the memory and no page fault occurs.

Page faults = 3

- When page 6 comes, it is not in the memory, so a page fault occurs and the least recently used page **3 is removed**.

Page faults = 4

- When page 5 comes, it again causes a page fault and page **1 is removed** as it is now the least recently used page.

Page faults = 5

- When page 1 comes again, it is not in the memory and hence page **2 is removed** according to the LRU.

Page faults = 6

- When page 3 comes, the page fault occurs again and this time page **6 is removed** as the least recently used one.

Total page faults = 7

Now in the above example, the LRU causes the same page faults as the FIFO, but this may not always be the case as it will depend upon the series, the number of frames available in memory, etc. In fact, on most occasions, LRU is better than FIFO.

Advantages

- It is open for full analysis
- Doesn't suffer from Belady's anomaly
- Often more efficient than other algorithms

Disadvantages

- It requires additional data structures to be implemented
- More complex
- High hardware assistance is required