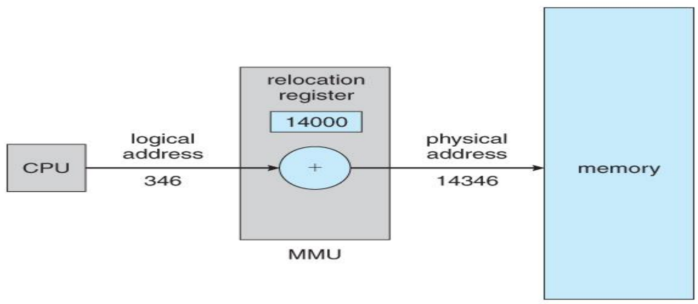
Memory Management

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.

Logical address and Physical address space



The addresses generated by a program is referred to as a **logical address or virtual address space.**
The addresses generated by a memory unit is referred to as a **physical address space.**
The runtime mapping from virtual to physical address is done by hardware device is called as the **memory management unit (MMU).**
MMU uses following mechanism to convert virtual address to physical address.

- The value in the base register or relocation register is added to every address generated by a user process, which is treated as offset at the time it is sent to memory. For example, if the base register value is 10000, then an attempt by the user to use address location 100 will be dynamically reallocated to location 10100.
- Hence, **physical address space=logical address space + contents of relocation register**
- The user program deals with virtual addresses, it never sees the real physical addresses.

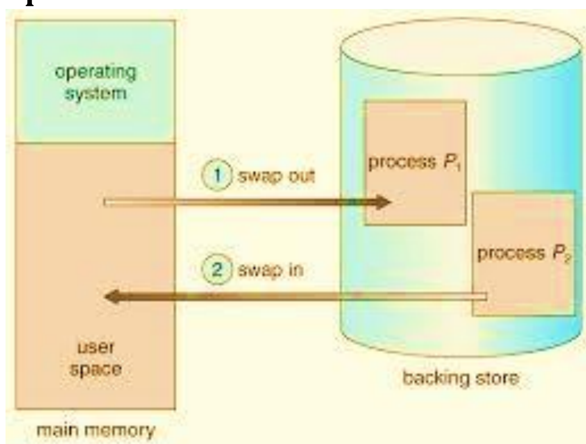
BASIS FOR COMPARISON	LOGICAL ADDRESS	PHYSICAL ADDRESS
Basic	It is the virtual address generated by CPU	The physical address is a location in a memory unit.
Address Space	Set of all logical addresses generated by CPU in reference to a program is referred as Logical Address Space.	Set of all physical addresses mapped to the corresponding logical addresses is referred as Physical Address.
Generation	The Logical Address is generated by the CPU	Physical Address is Computed by MMU

Memory Management

Swapping

Swapping is a mechanism in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes. At some later time, the system swaps back the process from the secondary storage to main memory.

Though performance is usually affected by swapping process but it helps in running multiple and big processes in parallel and that's the reason **Swapping is also known as a technique for memory compaction**.



The total time taken by swapping process includes the time it takes to move the entire process to a secondary disk and then to copy the process back to memory, as well as the time the process takes to regain main memory.

Let us assume that the user process is of size 2048KB and on a standard hard disk where swapping will take place has a data transfer rate around 1 MB per second. The actual transfer of the 1000K process to or from memory will take

2048KB / 1024KB per second
= 2 seconds
= 2000 milliseconds

Now considering in and out time, it will take complete 4000 milliseconds plus other overhead where the process competes to regain main memory.

Contiguous Memory Allocation

Main memory usually has two partitions :

Low Memory – Operating system resides in this memory.
High Memory – User processes are held in high memory.

Multiple- Fixed partition allocation

In this type of allocation, main memory is divided into a number of fixed-sized partitions where each partition should contain only one process. When a partition is free, a process is selected from the input queue and is loaded into the free partition. When the process terminates, the partition becomes available for another process.

Fixed size memory allocation is a process by which computer programs are assigned memory or space. And this method leads to dynamic storage allocation problem for which there are three types solutions available.

First Fit

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

Best Fit

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

Worst Fit

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

Example: Given five memory partitions of 100Kb, 500Kb, 200Kb, 300Kb, 600Kb (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of 212 Kb, 417 Kb, 112 Kb, and 426 Kb (in order)? Which algorithm makes the most efficient use of memory?

First-fit:
212K is put in 500K partition
417K is put in 600K partition
112K is put in 288K partition (new partition 288K = 500K - 212K)
426K must wait

Best-fit:
212K is put in 300K partition
417K is put in 500K partition
112K is put in 200K partition
426K is put in 600K partition

Worst-fit:
212K is put in 600K partition
417K is put in 500K partition
112K is put in 388K partition
426K must wait

In this example, best-fit turns out to be the best.

Fragmentation

As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as Fragmentation.

Fragmentation is of two types:

1	**External fragmentation:** Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used.
2	**Internal fragmentation:** Memory block assigned to process is bigger. Some portion of memory is left unused, as it cannot be used by another process.

The above problem can be solved in three methods:

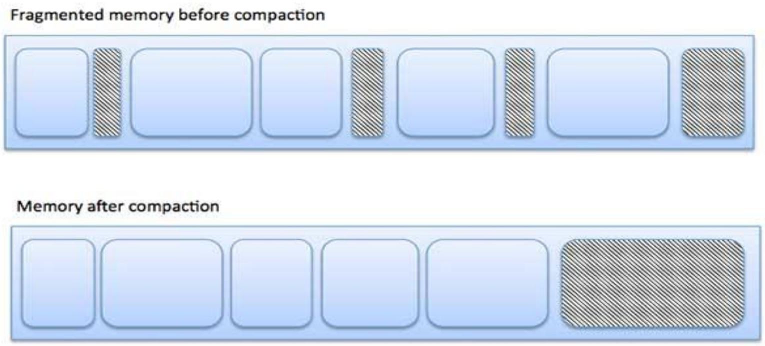
1. Compaction

2. Paging

3. Segmentation.

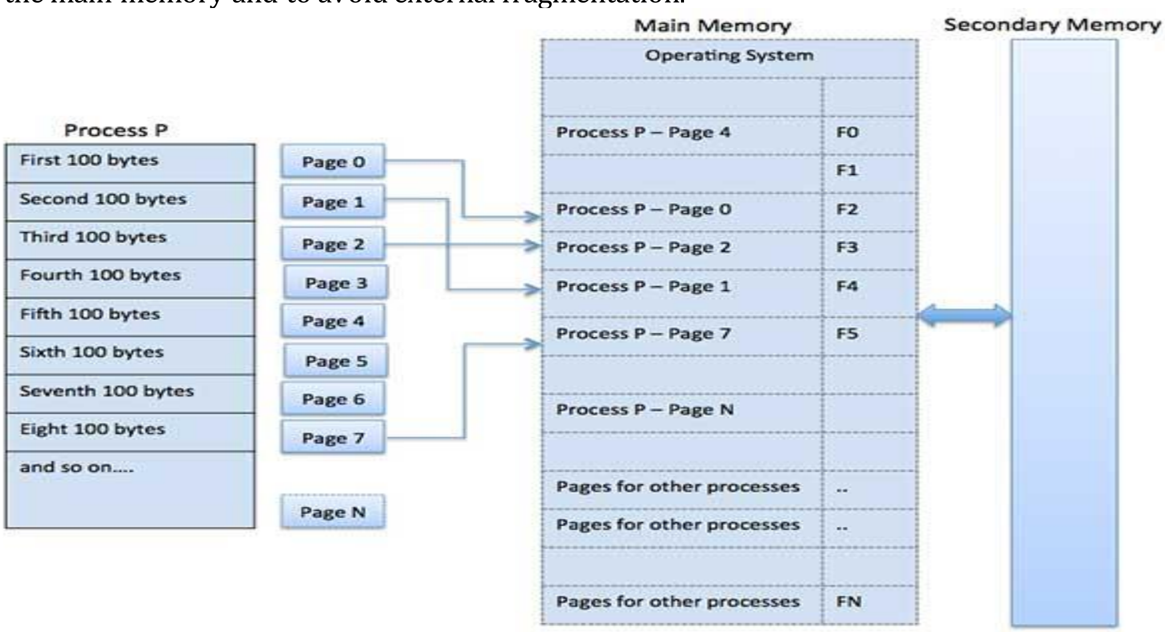
Compaction:

The following diagram shows how fragmentation can cause waste of memory and a compaction technique can be used to create more free memory out of fragmented memory.



Paging

Paging is a memory management technique in which process address space is broken into blocks of the same size called **pages**. The size of the process is measured in the number of pages.
Similarly, main memory is divided into small fixed-sized blocks of (physical) memory called **frames** and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.



When a process is to be executed, its corresponding pages are loaded into any available memory frames. Suppose you have a program of 8Kb but your memory can accommodate only 5Kb at a given point in time, then the paging concept will come into picture. When a computer runs out of RAM, the operating system (OS) will move idle or unwanted pages of memory to secondary memory to free up RAM for other processes and brings them back when needed by the program.

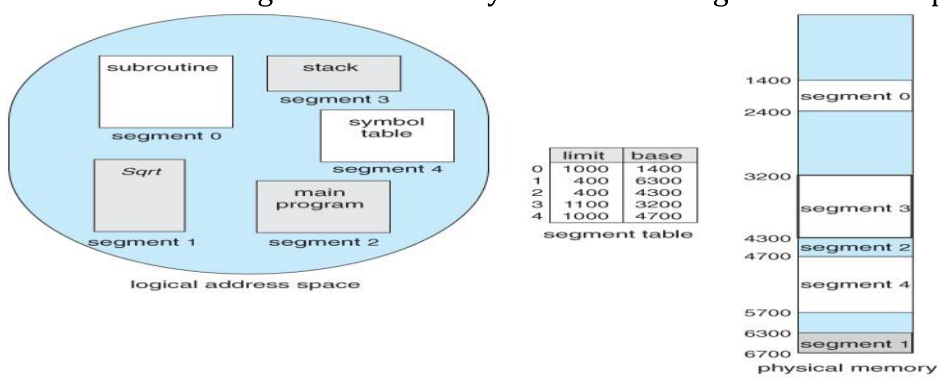
This process continues during the whole execution of the program where the OS keeps removing idle pages from the main memory and write them onto the secondary memory and bring them back when required by the program.

Advantages and Disadvantages of Paging

- Paging reduces external fragmentation, but still suffer from internal fragmentation.
- Paging is simple to implement and assumed as an efficient memory management technique.
- Due to equal size of the pages and frames, swapping becomes very easy.
- Page table requires extra memory space, so may not be good for a system having small RAM.

Segmentation

Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program.



When a process is to be executed, its corresponding segmentation are loaded into non-contiguous memory though every segment is loaded into a contiguous block of available memory.

Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.

A program segment contains the program's main function, utility functions, data structures, and so on. The operating system maintains a **segment map table** for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory. For each segment, the table stores the starting address of the segment and the length of the segment.

Advantages of memory segmentation:

1. Segmentation provides a powerful memory management mechanism.
2. It allows programmers to partition their programs into modules that operate independently of one another.
3. Segments allow two processes to easily share data.
4. Segmentation makes it possible to separate the memory areas for stack, code and data.

PAGING	SEGMENTATION
A page is of fixed block size.	A segment is of variable size.
Paging may lead to internal fragmentation.	Segmentation may lead to external fragmentation.
Does not support growing and shrinking of pages	Supports dynamic growing and shrinking of segments.
Does not support for modularity	Supports for modularity.
The user specified address is divided by CPU into a page number and offset.	The user specifies each address by two quantities a segment number and the offset (Segment limit).
The hardware decides the page size.	The segment size is specified by the user.
Paging involves a page table that contains base address of each page.	Segmentation involves the segment table that contains segment number and offset (segment length).

Advantages of Segmentation with paging

It combines the advantages of paging with the advantages of segmentation.

- Simple and can relocate things at will.
- Supports modular programming
- Segmentation allows sharing of files among users
- Combines the advantages of both paging and segmentation
- Removes the problems of both internal and external fragmentation.

Disadvantages of segmentation with paging

- Increases overheads as extra tables have to be maintained.
- Time consuming as more time is needed for address calculation to reference segments table and page table.