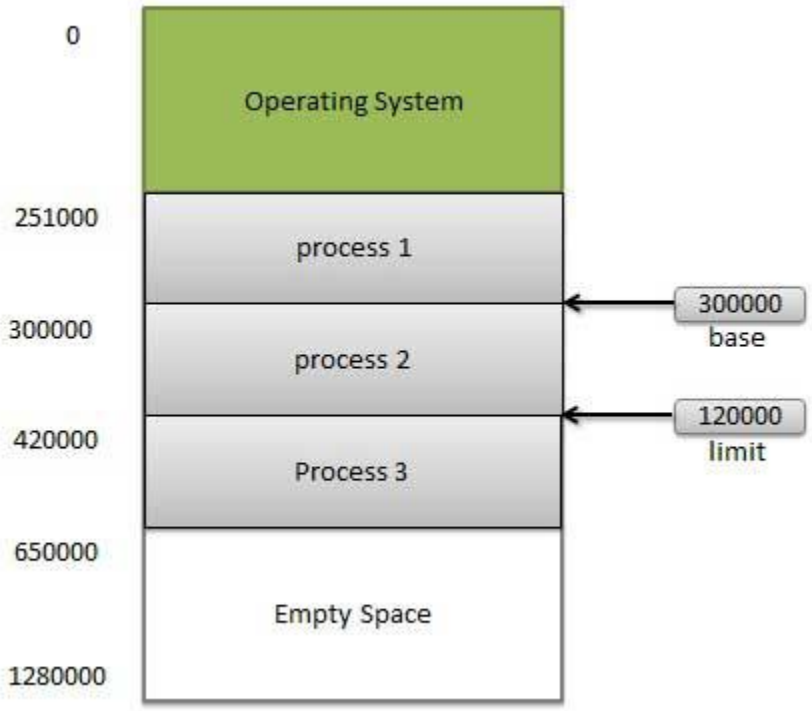
OS - Memory Management

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 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.

Memory management provides protection by using two registers, a base register and a limit register. The base register holds the smallest legal physical memory address and the limit register specifies the size of the range. For example, if the base register holds 300000 and the limit register is 1209000, then the program can legally access all addresses from 300000 through 411999.



Instructions and data to memory addresses can be done in following ways

- **Compile time** -- When it is known at compile time where the process will reside, compile time binding is used to generate the absolute code.
- **Load time** -- When it is not known at compile time where the process will reside in memory, then the compiler generates re-locatable code.
- **Execution time** -- If the process can be moved during its execution from one memory segment to another, then binding must be delayed to be done at run time

Dynamic Loading

In dynamic loading, a routine of a program is not loaded until it is called by the program. All routines are kept on disk in a re-locatable load format. The main program is loaded into memory and is executed. Other routines methods or modules are loaded on request. Dynamic loading makes better memory space utilization and unused routines are never loaded.

Dynamic Linking

Linking is the process of collecting and combining various modules of code and data into a executable file that can be loaded into memory and executed. Operating system can link system level libraries to a program. When it combines the libraries at load time, the linking is called static linking and when this linking is done at the time of execution, it is called as dynamic linking.

In static linking, libraries linked at compile time, so program code size becomes bigger whereas in dynamic linking libraries linked at execution time so program code size remains smaller.

Logical versus Physical Address Space

An address generated by the CPU is a logical address whereas address actually available on memory unit is a physical address. Logical address is also known a Virtual address.

Virtual and physical addresses are the same in compile-time and load-time address-binding schemes. Virtual and physical addresses differ in execution-time address-binding scheme.

The set of all logical addresses generated by a program is referred to as a logical address space. The set of all physical addresses corresponding to these logical addresses is referred to as a physical address space.

The run-time mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device. MMU uses following mechanism to convert virtual address to physical address.

- The value in the base 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.

- The user program deals with virtual addresses; it never sees the real physical addresses.

Swapping

Swapping is a mechanism in which a process can be swapped temporarily out of main memory to a backing store , and then brought back into memory for continued execution.

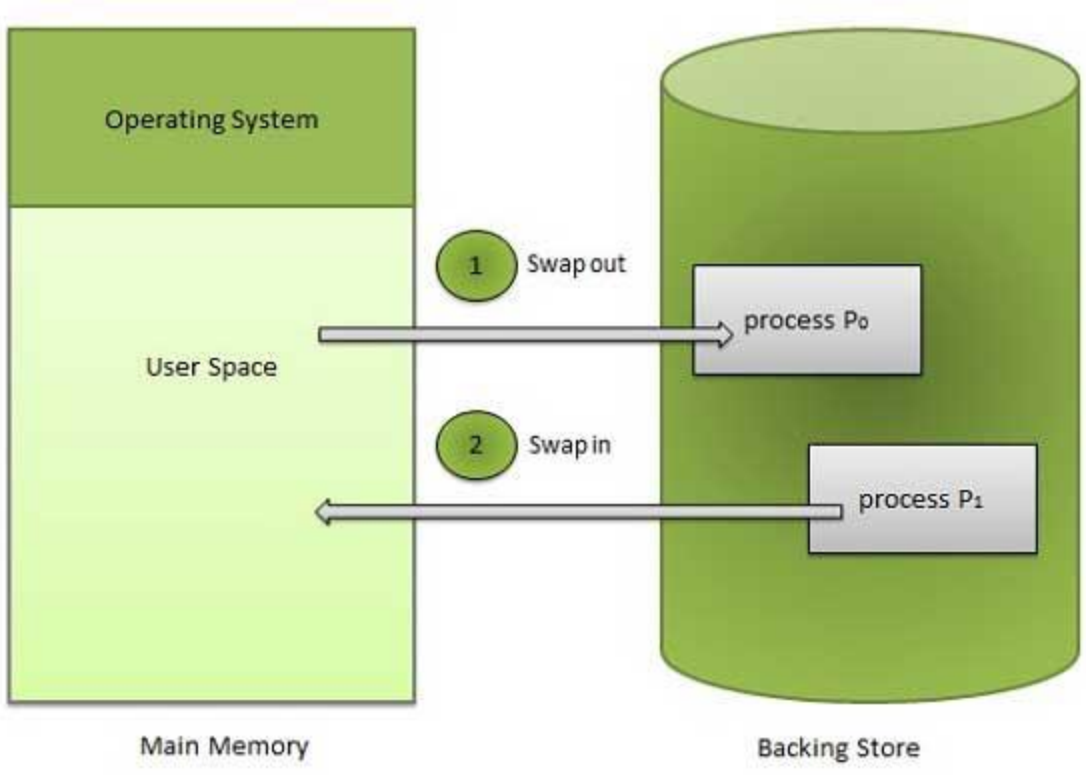
Backing store is a usually a hard disk drive or any other secondary storage which fast in access and large enough to accommodate copies of all memory images for all users. It must be capable of providing direct access to these memory images.

Major time consuming part of swapping is transfer time. Total transfer time is directly proportional to the amount of memory swapped. Let us assume that the user process is of size 100KB and the backing store is a standard hard disk with transfer rate of 1 MB per second. The actual transfer of the 100K process to or from memory will take

100KB / 1000KB per second

= 1/10 second

= 100 milliseconds



Memory Allocation

Main memory usually has two partitions

- **Low Memory** -- Operating system resides in this memory.
- **High Memory** -- User processes then held in high memory.

Operating system uses the following memory allocation mechanism.

S.N.	Memory Allocation	Description
1	**Single-partition allocation**	In this type of allocation, relocation-register scheme is used to protect user processes from each other, and from changing operating-system code and data. Relocation register contains value of smallest physical address whereas limit register contains range of logical addresses. Each logical address must be less than the limit register.
2	**Multiple-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.

Fragmentation

As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes can not 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

S.N.	Fragmentation	Description
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 can not be used.
2	**Internal fragmentation**	Memory block assigned to process is bigger. Some portion of memory is left unused as it can not be used by another process.

External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic.

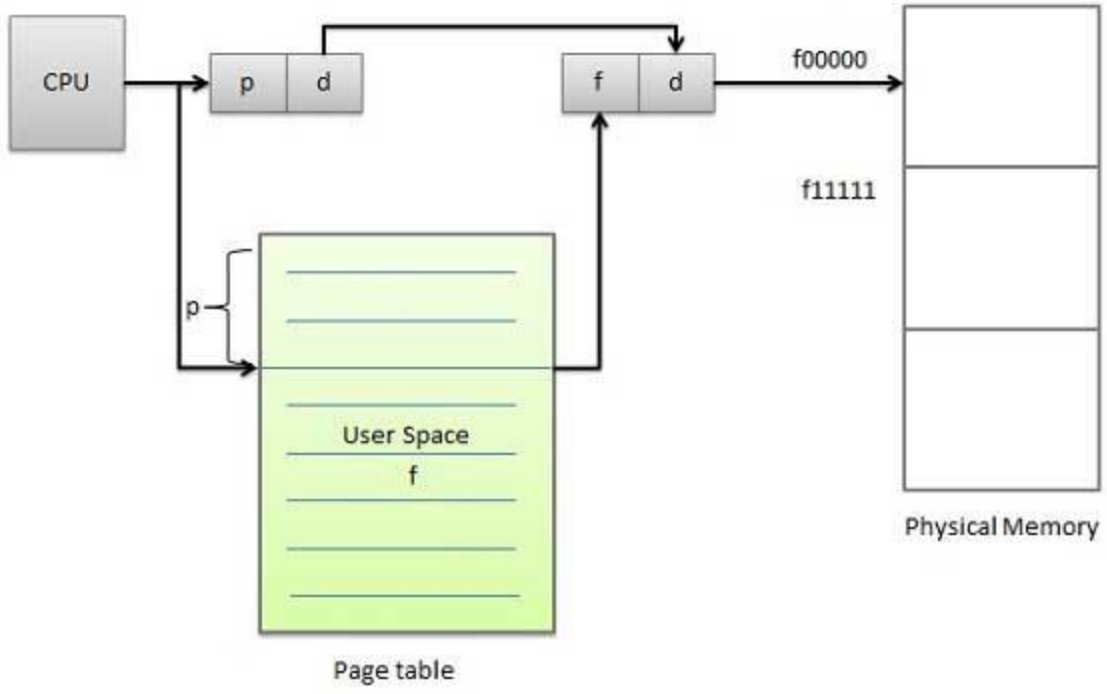
Paging

External fragmentation is avoided by using paging technique. Paging is a technique in which physical memory is broken into blocks of the same size called pages (size is power of 2, between 512 bytes and 8192 bytes). When a process is to be executed, it's corresponding pages are loaded into any available memory frames.

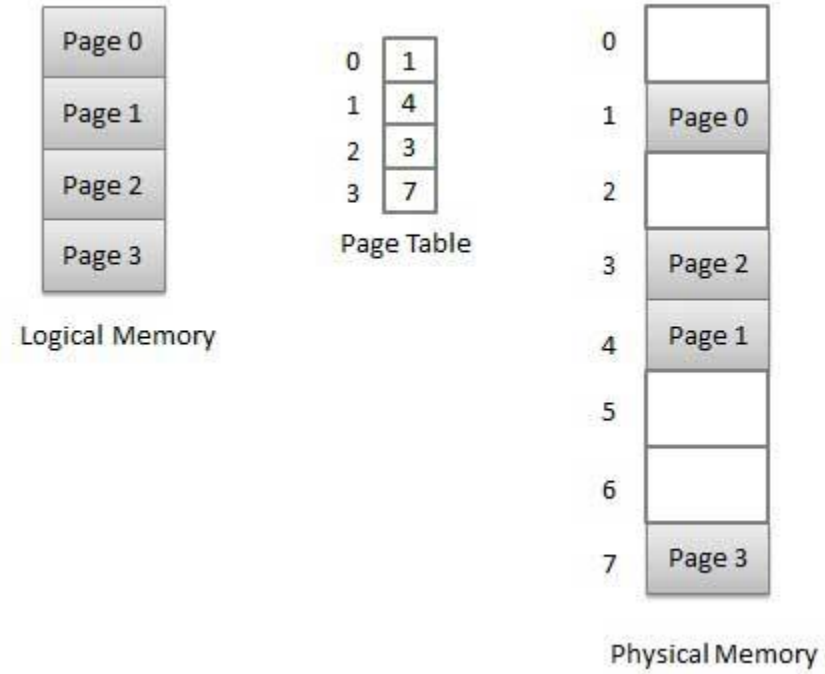
Logical address space of a process can be non-contiguous and a process is allocated physical memory whenever the free memory frame is available. Operating system keeps track of all free frames. Operating system needs n free frames to run a program of size n pages.

Address generated by CPU is divided into

- **Page number (p)** -- page number is used as an index into a page table which contains base address of each page in physical memory.
- **Page offset (d)** -- page offset is combined with base address to define the physical memory address.



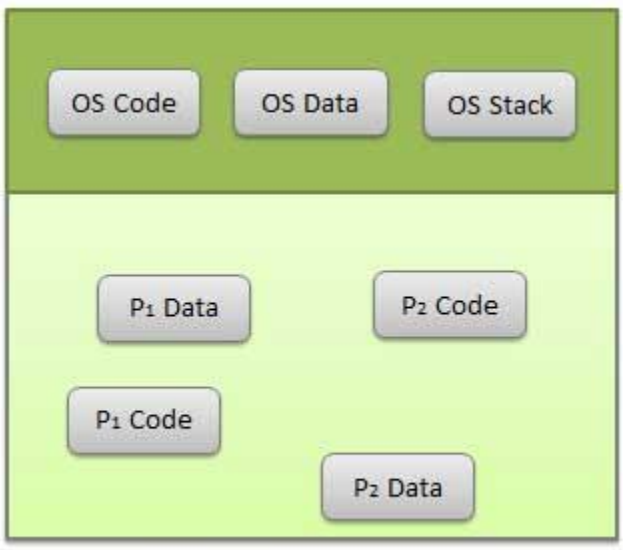
Following figure show the paging table architecture.



Segmentation

Segmentation is a technique to break memory into logical pieces where each piece represents a group of related information. For example ,data segments or code segment for each process, data segment for operating system and so on. Segmentation can be implemented using or without using paging.

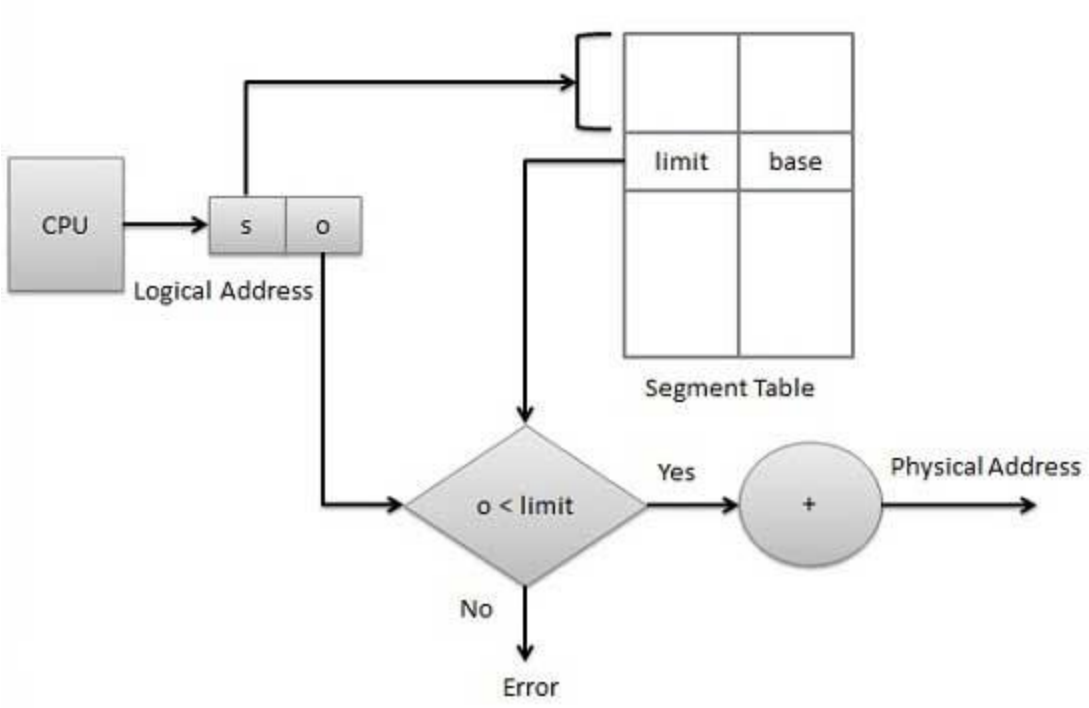
Unlike paging, segment are having varying sizes and thus eliminates internal fragmentation. External fragmentation still exists but to lesser extent.



Logical Address Space

Address generated by CPU is divided into

- **Segment number (s)** -- segment number is used as an index into a segment table which contains base address of each segment in physical memory and a limit of segment.
- **Segment offset (o)** -- segment offset is first checked against limit and then is combined with base address to define the physical memory address.



Source:

http://www.tutorialspoint.com/operating_system/os_memory_management.htm