UNIT III

MEMORY MANAGEMENT: BACKGROUND

In general, to run a program, it must be brought into memory.

Input queue – collection of processes on the disk that are waiting to be brought into memory to run the program.

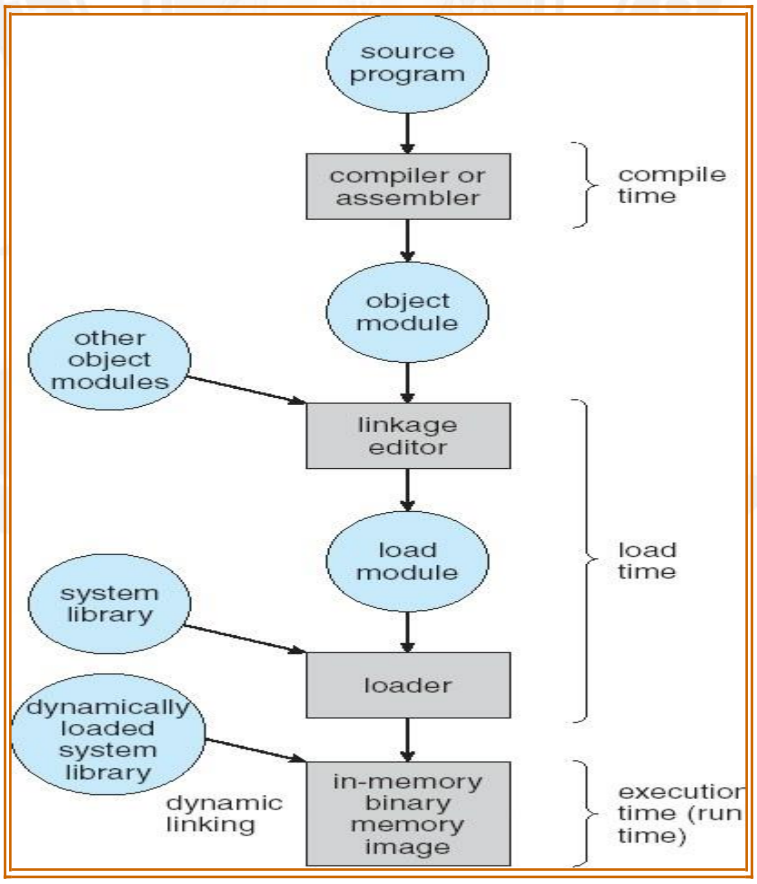
User programs go through several steps before being run

Address binding: Mapping of instructions and data from one address to another address in memory.

Three different stages of binding:

1. **Compile time**: Must generate absolute code if memory location is known in prior.
2. **Load time**: Must generate relocatable code if memory location is not known at compile time
3. **Execution time**: Need hardware support for address maps (e.g., base and limit registers).

Multistep Processing of a User Program



Logical vs. Physical Address Space

- **Logical address** – generated by the CPU; also referred to as **"virtual address"**
- **Physical address** – address seen by the memory unit.
- Logical and physical addresses are the **same** in –compile-time and load-time address-binding schemes
- Logical (virtual) and physical addresses **differ** in –execution-time address- binding scheme

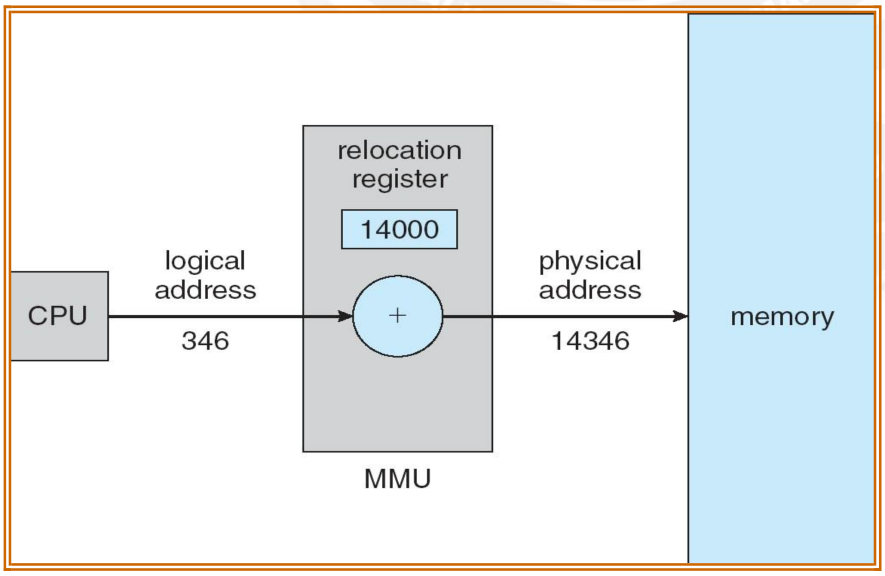
Memory-Management Unit (MMU)

It is a hardware device that maps virtual / Logical address to physical address

- In this scheme, the relocation register's value is added to Logical address generated by a user process.
- The user program deals with *logical* addresses; it never sees the *real* physical addresses
- Logical address range: 0 to max
- Physical address range: R+0 to R+max, where R—value in relocation register

Note: relocation register is a base register.

Dynamic relocation using relocation register



Dynamic Loading

- Through this, the routine is not loaded until it is called.

- Better memory-space utilization; unused routine is never loaded
- Useful when large amounts of code are needed to handle infrequently occurring cases
- No special support from the operating system is required implemented through program design

Dynamic Linking

- Linking postponed until execution time & is particularly useful for libraries
- Small piece of code called stub, used to locate the appropriate memory-resident library routine or function.
- Stub replaces itself with the address of the routine, and executes the routine
- Operating system needed to check if routine is in processes' memory address

Shared libraries: Programs linked before the new library was installed will continue using the older library

Overlays:

- Enable a process larger than the amount of memory allocated to it.
- At a given time, the needed instructions & data are to be kept within a memory.

Swapping

A process can be swapped temporarily out of memory to a backing store (SWAP OUT) and then brought back into memory for continued execution (SWAP IN).

Backing store

Fast disk large enough to accommodate copies of all memory images for all users & it must provide direct access to these memory images

Roll out, roll in

Swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed

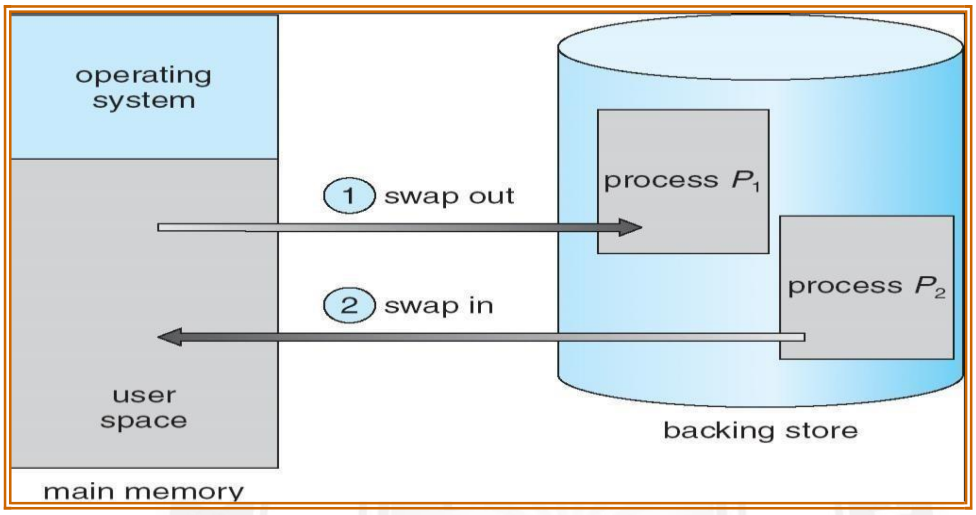
Transfer time :

- Major part of swap time is transfer time
- Total transfer time is directly proportional to the amount of memory swapped.

Example: Let us assume the user process is of size 1MB & the backing store is a standard hard disk with a transfer rate of 5MBPS.

Transfer time = 1000KB/5000KB per second

= 1/5 sec = 200ms



(i) Memory Protection:

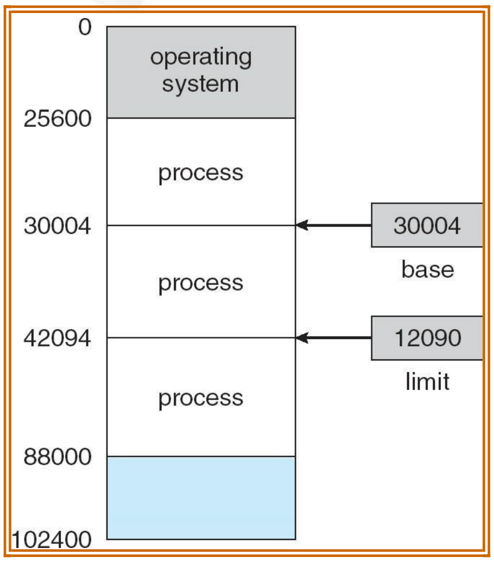
Memory protection is needed for

a) Protecting the OS from user process.

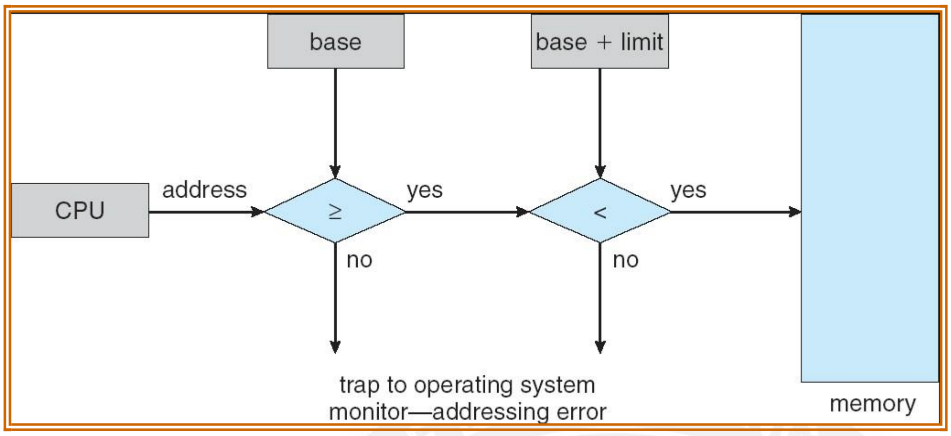
b) Protecting user processes from one another.

- The above protection is done by **"Relocation-register & Limit-register scheme** –
- Relocation register contains value of smallest physical address i.e base value.
- Limit register contains range of logical addresses – each logical address must be less than the limit register

A base and a limit register define a logical address space



HW address protection with base and limit registers



Contiguous Allocation

Each process is contained in a single contiguous section of memory.

There are two methods namely :

- Fixed – Partition Method
- Variable – Partition Method

Fixed – Partition Method :

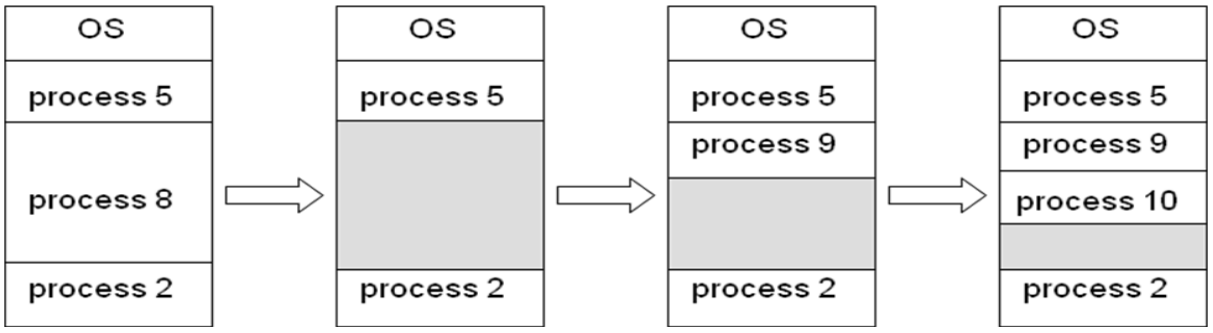
- Divide memory into fixed size partitions, where each partition has exactly one process.
- The drawback is memory space unused within a partition is wasted.

(eg.when process size < partition size)

Variable-partition method:

- Divide memory into variable size partitions, depending upon the size of the incoming process.
- When a process terminates, the partition becomes available for another process.
- As processes complete and leave they create holes in the main memory.

Hole – block of available memory; holes of various size are scattered throughout memory.



Dynamic Storage-Allocation Problem:

How to satisfy a request of size ‗n‘ from a list of free holes?

Solution:

First-fit: Allocate the *first* hole that is big enough.

Best-fit: Allocate the *smallest* hole that is big enough; must search entire list, unless ordered by size. Produces the smallest leftover hole.

Worst-fit: Allocate the *largest* hole; must also search entire list. Produces the largest leftover hole.

NOTE: First-fit and best-fit are better than worst-fit in terms of speed and storage utilization

Fragmentation:

External Fragmentation

– This takes place when enough total memory space exists to satisfy a request, but it is not contiguous i.e, storage is fragmented into a large number of small holes scattered throughout the main memory.

Internal Fragmentation

– Allocated memory may be slightly larger than requested memory.

Example: hole = 184 bytes

Process size = 182 bytes. We are left with a hole of 2 bytes.

Solutions:

1. **Coalescing :** Merge the adjacent holes together.
2. **Compaction:** Move all processes towards one end of memory, hole towards other end of memory, producing one large hole of available memory. This scheme is expensive as it can be done if relocation is dynamic and done at execution time.
3. Permit the logical address space of a process to be **non-contiguous**.

This is achieved through two memory management schemes namely **paging** and **segmentation**.