What is the base and limit registers ?

- **Base register** : from what address in the memory the process start
- **Limit register** : the length of the process in the memory , that equals to end address minus start address

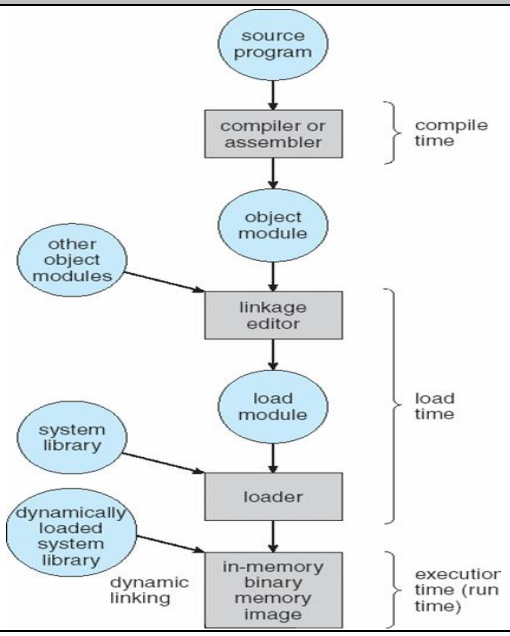
What is the address binding of instructions and data to memory addresses ?

Define the instruction location address

What the at three different stages that the Address binding of instructions and data to memory addresses can happen in ?

- **Compile time:** If memory location known a priori, absolute code can be generated; must recompile code if starting location changes
- **Load time:** Must generate reloadable code if memory location is not known at compile time
- **Execution time:** Binding delayed until run time if the process can be moved during its execution from one memory segment to another. Need hardware support for address maps (e.g., base and limit registers)

Explain in figure Multistep Processing of a User Program ?



What is the logical address and physical address and different between them ?

- **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

What is memory management unit MMU ?

- Hardware device that maps virtual to physical address
- In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory
- The user program deals with *logical* addresses; it never sees the *real* physical addresses

What is dynamic loaded ?

- 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 (switch in java)

What is dynamic linking ?

- Linking postponed until execution time
- Small piece of code, *stub*, used to locate the appropriate memory-resident library routine
- 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
- Dynamic linking is particularly useful for libraries
- System also known as **shared libraries**

What is swapping ?

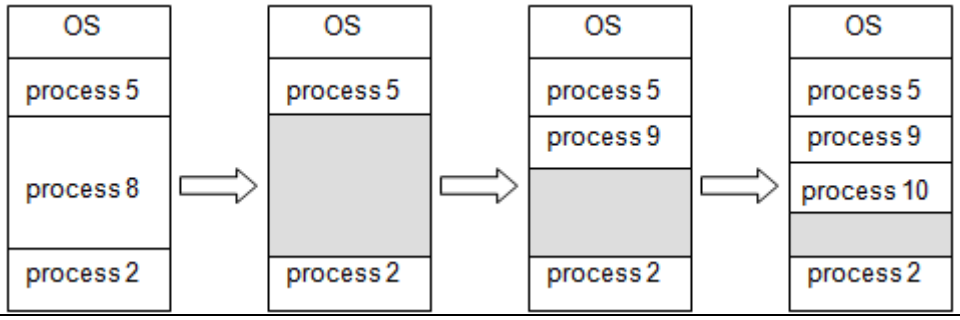
- A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution
- **Backing store** : fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images
- **Roll out, roll in** : swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed
- Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped

What the parts of RAM ?

- **Resident operating system** : usually held in low memory or high memory depend on interrupt vector place (together)
- **User processes :** in the other side

Explain the contiguous allocation ?

- **Hole** : block of available memory; holes of various size are scattered throughout memory
- When a process arrives, it is allocated memory from a hole large enough to accommodate it
- Operating system maintains information about:
  a) allocated partitions
  b) free partitions (hole)



What is the types of satisfy a request of size *n* from a list of free holes?

- **First-fit**: Allocate the *first* hole that is big enough **(No Overhead)**
- **Best-fit**: Allocate the *smallest* hole that is big enough; must search entire list, unless ordered by size **(Overhead)**
  - Produces the smallest leftover hole **(storage utilization)**
- **Worst-fit**: Allocate the *largest* hole; must also search entire list
  - Produces the largest leftover hole **(may also allocated later)**

❖ **First-fit and best-fit better than worst-fit in terms of speed and storage utilization**

What is the types of fragmentation ?

- **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous
- **Internal Fragmentation** – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used

What is the solution of external fragmentation ?

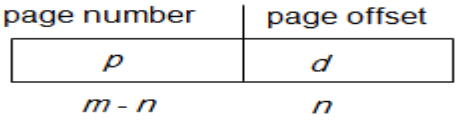
- by **compaction**
  - Shuffle memory contents to place all free memory together in one large block
  - Compaction is possible *only* if relocation is dynamic, and is done at execution time
- **Disadvantages** :
  - Overhead
  - I/O problem

Explain the paging ?

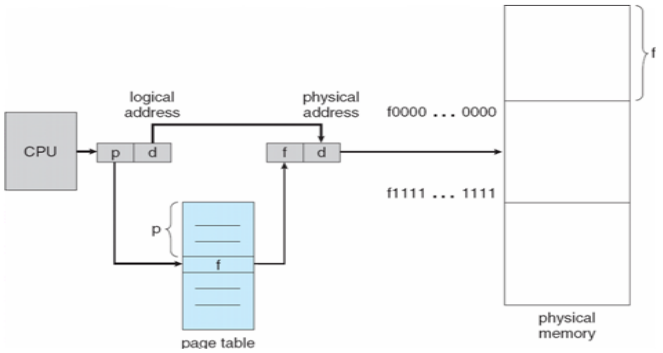
- Logical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available
- Divide physical memory into fixed-sized blocks called **frames** (size is power of 2, between 512 bytes and 8,192 bytes)
- Divide logical memory into blocks of same size called **pages**
- Keep track of all free frames
- To run a program of size ***n*** pages, need to find ***n*** free frames and load program
- Set up a page table to translate logical to physical addresses
- **Disadvantage** : Internal fragmentation

How to translate between Logical to physical in paging ?

- Address generated by CPU is divided into:
  - **Page number (*p*)** : used as an index into a *page table* which contains base address of each page in physical memory
    - **Page offset (d)** : combined with base address to define the physical memory address that is sent to the memory unit



For given logical address space 2^m *and page size* 2^n



How to translate from logical address to physical address ?

Physical address = (frame number * page size) + order

What is segmentation ?

Memory-management scheme that supports user view of memory

Explain segmentation architecture ?

- Logical address consists of a two tuple:
- <segment-number, offset>,
- **Segment table** : maps two-dimensional physical addresses; each table entry has:
- **base** : contains the starting physical address where the segments reside in memory
- **limit** : specifies the length of the segment
- **Segment-table base register (STBR)** points to the segment table's location in memory
- **Segment-table length register (STLR)** indicates number of segments used by a program;
- segment number ***s*** is legal if ***s*** < **STLR**