Secondary Storage Management:

The operating system is responsible for the following activities related with disk management:

- Free space management.
- Storage allocation.
- Disk scheduling.

Networking:

A networked system is a collection of programs and associated protocols, which do not share memory or peripheral devices. The processors communicate each other through high speed busses or telephone lines. The set of computers are inter-collected by a communication network.

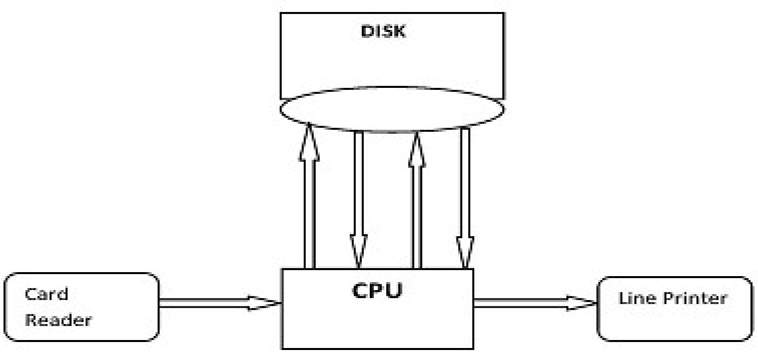
Buffering:

Buffering is another method of overlapping input/output and processing of a job. The order of execution is as follows:

- Read a set of data from input device
- After CPU starts execute the data, the input device begins to read the next set of data.
- By the time, CPU completes processing, thus CPU idle time is reduced.

Spooling:

- It is possible for the operating system to keep all the jobs on a disk rather than in a card reader.
- The cards read directly from the card reader into the disk.
- During execution, the operating system takes the input from the disk.
- When the job requests for output on a printer, the line is copied to system buffer and written on disk.
- After completion of job, output is printed.



Storage Structure:

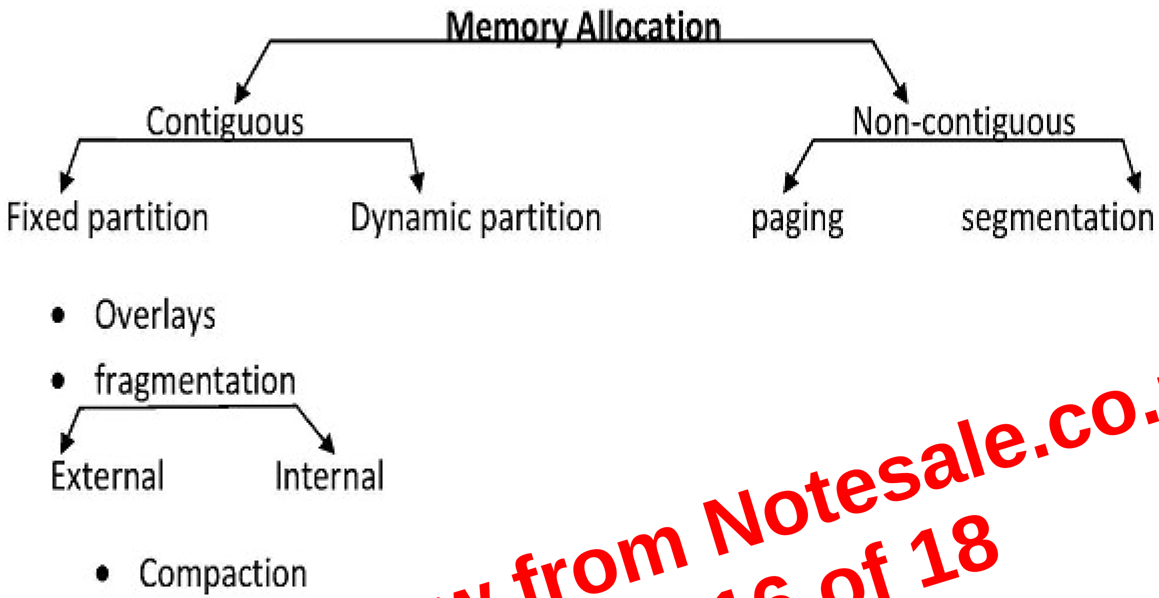
Advantages:

- Swapping can increase the degree of multi-programming

Disadvantages:

- Swapping whole partitions is expensive.
- The context switch time is high.

Memory Allocation:



Â

Â

Virtual Memory

- **Demand paging**
- **Page fault**
- **Page replacement algorithm**

Thrashing:
Virtual memory is a technique that allows the execution of process, which may not be completely fit in memory. One major advantage of this technique is that programs can be larger than the physical memory.

Overlays and dynamic loading also can solve this type of problems, but they generally require special logic and extra work by the programmer. But in the case of virtual memory, all those overhead can be eliminated.

Virtual memory is commonly implemented by demand paging. It can also be implemented by using demand segmentation.

Demand Paging:

The demand paging system is similar to a paging with swapping process resides on secondary memory. When we want to execute a process, we swap it into memory.

Rather than swapping the entire process at a time into the memory, we swap one particular page, which is required for execution. For swapping the pages we use lazy swapper. A lazy swapper never swaps a page into the memory unless that page will be needed.

When a process is to be swapped in, the pager (a swapper manipulates entire process at a time but a pager is connected with the individual pages of the process) decide which pages will be used before the process is swapped out again. Instead of swapping full pages at a time, the pager brings only those necessary pages into memory. Thus, it swaps only required pages, which is required for execution.

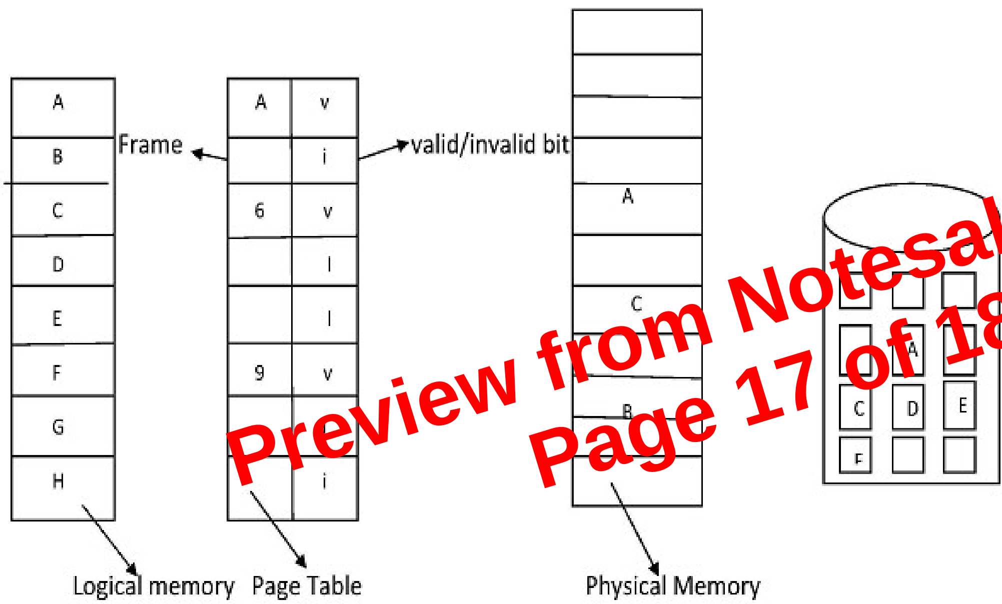


Fig: Page Table when some pages are not in Main Memory.

Page Fault & steps to handle page fault:

When one process is trying to access one particular page, but the page is not present in the memory at that time, page fault occurs. So accessing a page marked as invalid causes a page fault trap.

This trap is the result of operating system failure to bring the desired page into memory.

The procedure for handling this page fault is described as follows.

Step1: We check an internal table for this process, to determine whether the reference was valid or invalid Memory access.

Step2: If the reference was invalid, we terminate the process. If it is valid, the page is not yet in the memory. We have to bring that page into the memory.

Step3: We have to find a free frame.

Step4: Now we perform one disk operation to read the desired page into the newly allocated frame.

Step5: When the disk read is completed, we modify the internal table and the page table to indicate that the page is now in memory.

Step6: We restart the instruction that was interrupted to the operating system. Now the process can access the page and continue the execution.