* Virtual Memory

- A technique that gives an illusion of having more memory than what physical exists in RAM.

- It treats part of the hard-disk (swap space) as an extension of RAM.

Example:- Let's suppose a laptop has 4 GB RAM, but we can still open 8 GB programs because of virtual memory which take extra 4 GB from hard disk temporarily.

- **How Virtual Memory Works ?**

Virtual memory divides programs into pages (fixed-sized blocks). RAM is also divided into frames (same as page size). Pages of the program are mapped into frames using page table.

This connection ensures that only the needed parts of the program are in RAM.

- **Demand Paging**

Instead of loading the entire program, the OS loads only the pages that are needed right now; if a required page is not in RAM, then OS fetches it from disk.

Demand paging makes OS efficient use of virtual memory.

- **Valid- Invalid Bit**

To track which pages are in RAM, the OS uses a valid- invalid bit.

Valid (v) → page is in RAM.
Invalid (o) → page is on disk.

- **Page Fault**

If CPU tries to access a page with an invalid-bit; a page fault occurs.

Steps to handle the page fault:-

1. Check if memory access is legal. (Within the program)
2. If yes, find a free frame in RAM.
3. Load the page from disk into RAM frame.
4. Upade the page table (mark as valid).
5. Resume program execution.

- **Pure Demand Paging**

The best use of Demand paging, when a program starts, no pages are loaded into RAM. Only when the program tries to access a page → a page fault occurs → OS load the page from disk into RAM.

- **Locality of Reference**

It means programs tend to access the same data repeatedly (temporal) or nearby data (Spatial), and the OS uses this behavior to keep the right pages in RAM for efficiency.

Two main types are :-

① **Temporal locality** :- If a program access a piece of data or instruction, it is likely to access it again soon. So, OS keeps it in RAM.

② **Spatial locality** :- If a program access one memory location, it is likely to access nearby locations too. So, OS prefetches nearby memory pages into RAM.

- Why locality of reference ?
  Demand paging relied on it. If programs follow locality then,
  ① Page faults are fewer.
  ② Performance is fast.

If programs don't follow locality, page fault increases leads to "thrashing."

- **Page-Replacement Algorithms**
  When RAM is full and a new page needs to be loaded, OS must choose a page to remove from RAM for further execution. Algorithms mostly used :- FIFO, LRU & Optimal.

- **Thrashing** :-
  If a program needs too many pages and they don't fit in RAM, then page fault happen continuously. CPU spends more time in swapping pages rather than doing actual work.

"Thrashing is down side of the Demand paging."

- **Advantages of Virtual Memory**

1. Programs larger than RAM can also run.
2. Multiple programs run simultaneously.
3. Users get smooth multitasking experience.

Disadvantages of Virtual Memory

1. Disk access is slower → reduce performance
2. Too many pages faults → thrashing.

* Page Replacement Algorithm

1. FIFO (First-In First-Out) :-

- Replace the oldest page.
- Easy to implement, but performance can be poor as it may remove a frequently used page.

Belady's Anomaly :- Increasing frames may increases page faults.
Normally, we would expect that more frames will decrease the page faults but in FIFO sometimes the opposite happens that we called "Belady's Anomaly"

2. Optimal (OPT)

- Replace the page that is not in used for the longest time in future.
- Produces minimum page fault.

- Not practical to use as requires future knowledge of reference string.

③ LRU (least Recently used)

: Remove the least used page (page which are not in used for longest in past)

- Implementation :-

ⓐ Counter method (Time stamp Approach)
Each page stores last used time → replace the smallest.

ⓑ Stack method (Linked list Approach)
Maintain a stack of page numbers
- Top → most recently used.
- Bottom → least recently used.

When a page is needed :-
① If it's already in the stack → remove it and place it at the top.

② If it's not in the stack :→
- Push it from disk and place it on top.
- If stack is full → remove the bottom element, and placed the new page on top.

- LRU always gives the page directly, but maintaining the stack needs extra overhead.

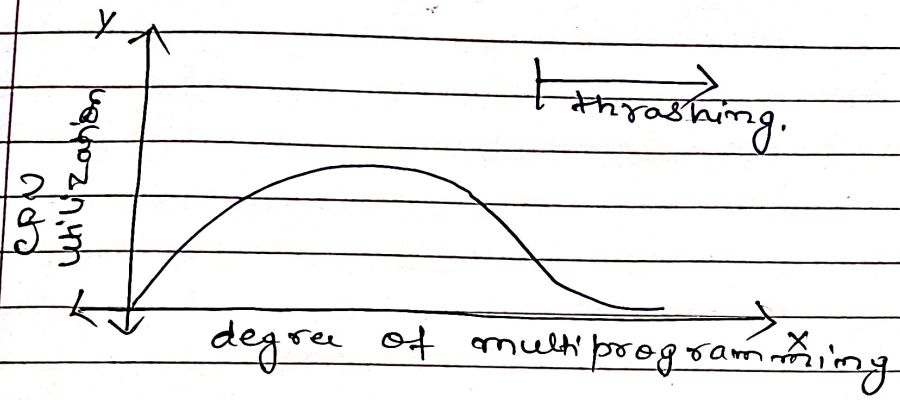
④ Counting - Based Algorithm
Keep a counter of references for each page:

Page No.:
Date:

(a) LFU (Least Frequently Used)
Replace page with lowest reference count.
Ideal for use in practice, but may keep old pages forever.

(b) MFU (Most Frequently Used)
Replace the most frequently used page.
Rarely used in practice.

* How to Handle thrashing



- Two main method to handle it:-

(1) Working Set Model
- Every process requires a set of pages, that we called "Working Set", if we provide enough frames to fit the Working Set then page fault won't happen.

(2) Page Fault Frequency (PFF)
- Thrashing shows high page fault freq.
- so, OS monitors page fault:-
  (A) If page faults are high → OS gives more frames.
  (B) If page faults are low, → OS take away few frames and give it to the necessary processes.

—x— (18/09/2025)