VIRTUAL MEMORY

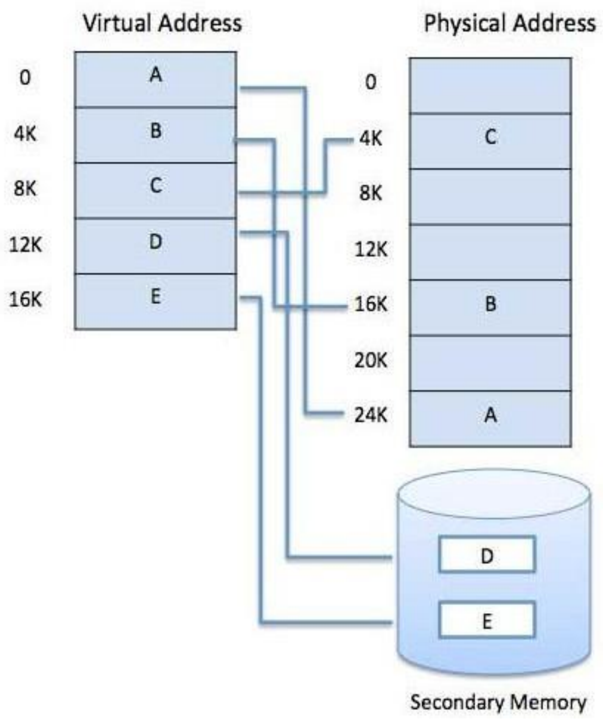
Virtual Memory is a space where large programs can store themselves in form of pages while their execution and only the required pages or portions of processes are loaded into the main memory. This technique is useful as large virtual memory is provided for user programs when a very small physical memory is there.

In real scenarios, most processes never need all their pages at once, for following reasons:

- Error handling code is not needed unless that specific error occurs, some of which are quite rare.
- Arrays are often over-sized for worst-case scenarios, and only a small fraction of the arrays are actually used in practice.
- Certain features of certain programs are rarely used.

Virtual Memory:

1. In the most of the computer system, the physical main memory is not as large as address space of the processor.
2. Suppose user tries to run a program.
3. If the program run by the user does not completely fit into the main memory then the parts of its currently being executed are stored in main memory and remaining portion is stored in secondary storage device such as HDD.
4. When a new part of program is to be brought into main memory for execution and if the memory is full, it must replace another part which is already is in main memory.
5. As this secondary memory is not actually part of system memory, so for CPU, secondary memory is considered as Virtual Memory.
6. Virtual memory is a memory management technique that is implemented using both hardware and software.
7. It maps memory addresses used by a program, called virtual addresses, into physical addresses in computer memory.



Benefits of having Virtual Memory:

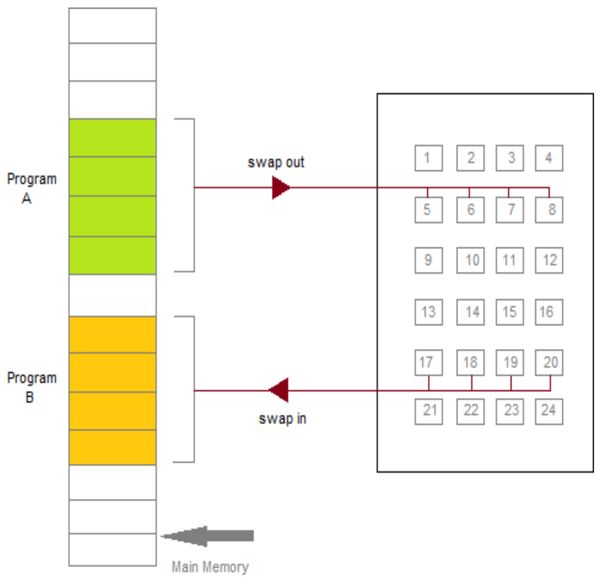
1. Large programs can be written, as virtual space available is huge compared to physical memory.
2. Less I/O required, leads to faster and easy swapping of processes.
3. More physical memory available, as programs are stored on virtual memory, so they occupy very less space on actual physical memory.

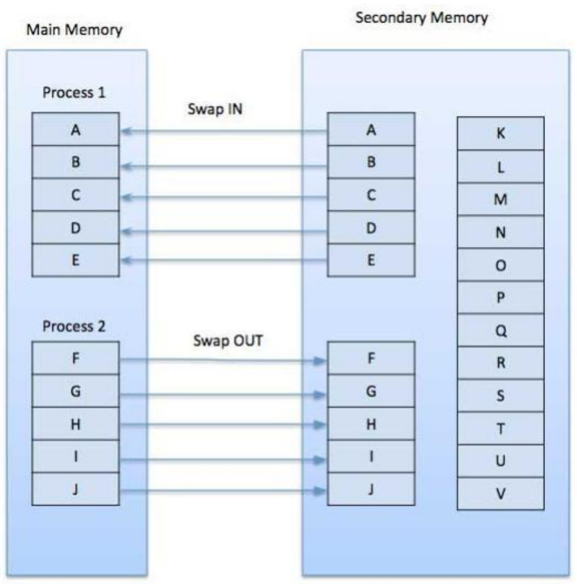
Disadvantages of Virtual Memory:

1. Applications run slower if the system is using virtual memory.
2. It takes more time to switch between applications.
3. Less hard drive space for your use.
4. It reduces system stability.

DEMAND PAGING

The basic idea behind demand paging is that when a process is swapped in, its pages are not swapped in all at once. Rather they are swapped in only when the process needs them(On demand). This is termed as lazy swapper, although a pager is a more accurate term.





Initially only those pages are loaded which will be required the process immediately.

The pages that are not moved into the memory are marked as invalid in the page table. For an invalid entry the rest of the table is empty. In case of pages that are loaded in the memory, they are marked as valid along with the information about where to find the swapped out page.

When the process requires any of the pages that is not loaded into the memory, a page fault trap is triggered and following steps are followed,

1. The memory address which is requested by the process is first checked, to verify the request made by the process.

2. If it found to be invalid, the process is terminated.
3. In case the request by the process is valid, a free frame is located, possibly from a free-frame list, where the required page will be moved.
4. A new operation is scheduled to move the necessary page from disk to the specified memory location. (This will usually block the process on an I/O wait, allowing some other process to use the CPU in the meantime.)
5. When the I/O operation is complete, the process's page table is updated with the new frame number, and the invalid bit is changed to valid.
6. The instruction that caused the page fault must now be restarted from the beginning.

There are cases when no pages are loaded into the memory initially; pages are only loaded when demanded by the process by generating page faults. This is called **Pure Demand Paging**.

The only major issue with Demand Paging is, after a new page is loaded, the process starts execution from the beginning. It is not a big issue for small programs, but for larger programs it affects performance drastically.

Advantages

Following are the advantages of Demand Paging –

- Large virtual memory large virtual memory.
- More if More efficient use of memory efficient use of memory.
- There is no limit on degree of multiprogramming. There is no limit on degree of multiprogramming.

Disadvantages

- Number of tables and the amount of processor overhead for handling page interrupts are greater than in the case of the simple paged management techniques.
- It suffers from internal fragmentation.
- There is an overhead of maintaining a page table for each process.
- The time taken to fetch the instruction increases since now two memory accesses is required.

PAGE REPLACEMENT

As studied in Demand Paging, only certain pages of a process are loaded initially into the memory. This allows us to get more number of processes into the memory at the same time. But what happens when a process requests for more pages and no free memory is available to bring them in. Following steps can be taken to deal with this problem:

1. Put the process in the wait queue, until any other process finishes its execution thereby freeing frames.
2. Or, remove some other process completely from the memory to free frames.
3. Or, find some pages that are not being used right now, move them to the disk to get free frames. This technique is called **Page replacement** and is most commonly used. We have some great algorithms to carry on page replacement efficiently.
4. Find the location of the page requested by ongoing process on the disk.
5. Find a free frame. If there is a free frame, use it. If there is no free frame, use a page-replacement algorithm to select any existing frame to be replaced, such frame is known as **victim frame**.
6. Write the victim frame to disk. Change all related page tables to indicate that this page is no longer in memory.
7. Move the required page and store it in the frame. Adjust all related page and frame tables to indicate the change.
8. Restart the process that was waiting for this page.

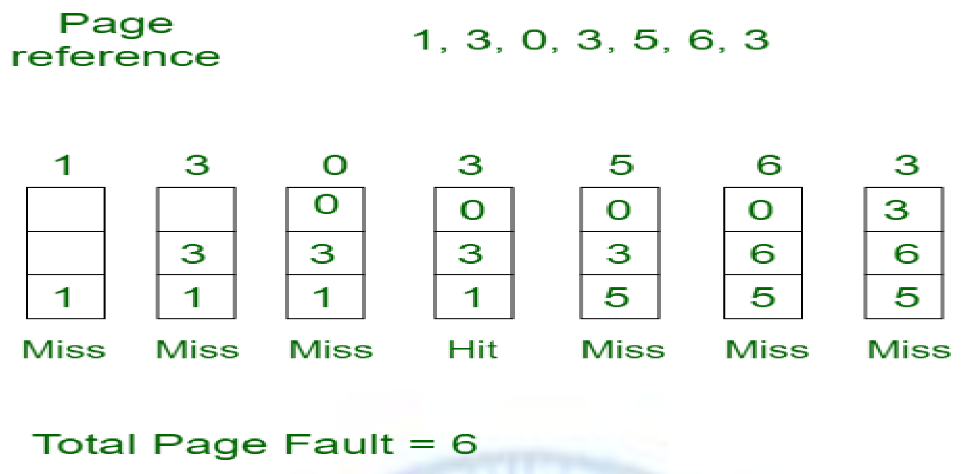
In an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when new page comes in.

Page Fault – A page fault happens when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory. Since actual physical memory is much smaller than virtual memory, page faults happen. In case of page fault, Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.

Page Replacement Algorithms:

- **First In First Out (FIFO) –**
  This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

  Example-1 Consider page reference string 1, 3, 0, 3, 5, 6 with 3 page frames. Find number of page faults.



Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> **3 Page Faults.** When 3 come, it is already in memory so —> **0 Page Faults.**

Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1. —>**1 Page Fault.**
6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>**1 Page Fault.**

Finally when 3 come it is not avilable so it replaces 0 **1 page faults**

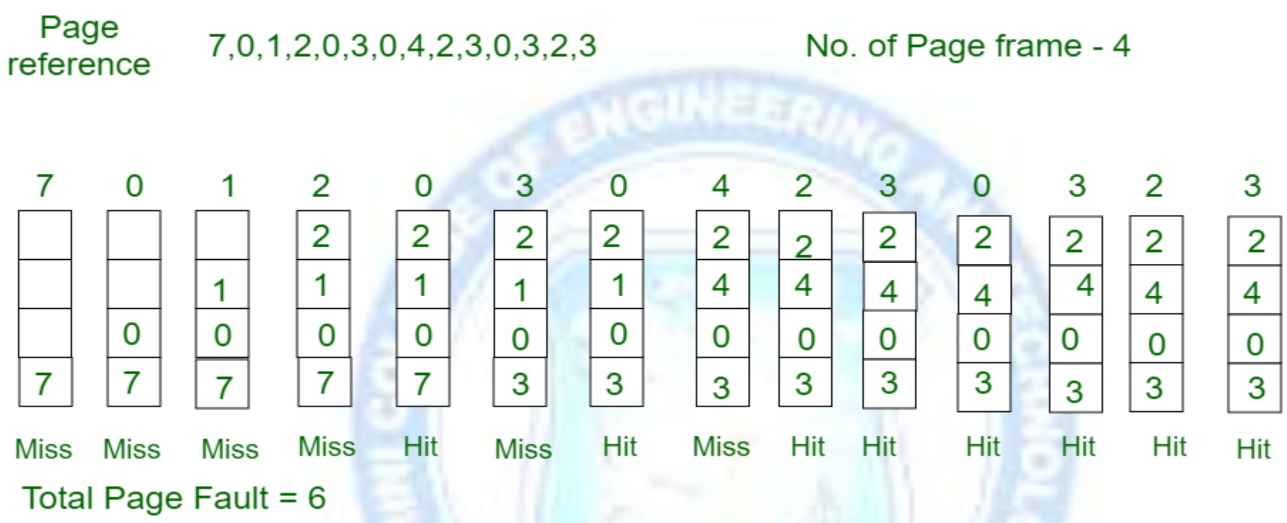
Belady's anomaly – Belady's anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm. For example, if we consider reference string 3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4 and 3 slots, we get 9 total page faults, but if we increase slots to 4, we get 10 page faults.

- A very simple way of Page replacement is FIFO (First in First Out)
- As new pages are requested and are swapped in, they are added to tail of a queue and the page which is at the head becomes the victim.
- Its not an effective way of page replacement but can be used for small systems.

- **Optimal Page replacement –**

  In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

  Example-2: Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, with 4 page frame. Find number of page fault.



Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**

0 is already there so —> **0 Page fault.**

when 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—>**1 Page fault.**

0 is already there so —> **0 Page fault.**

4 will takes place of 1 —> **1 Page Fault.**

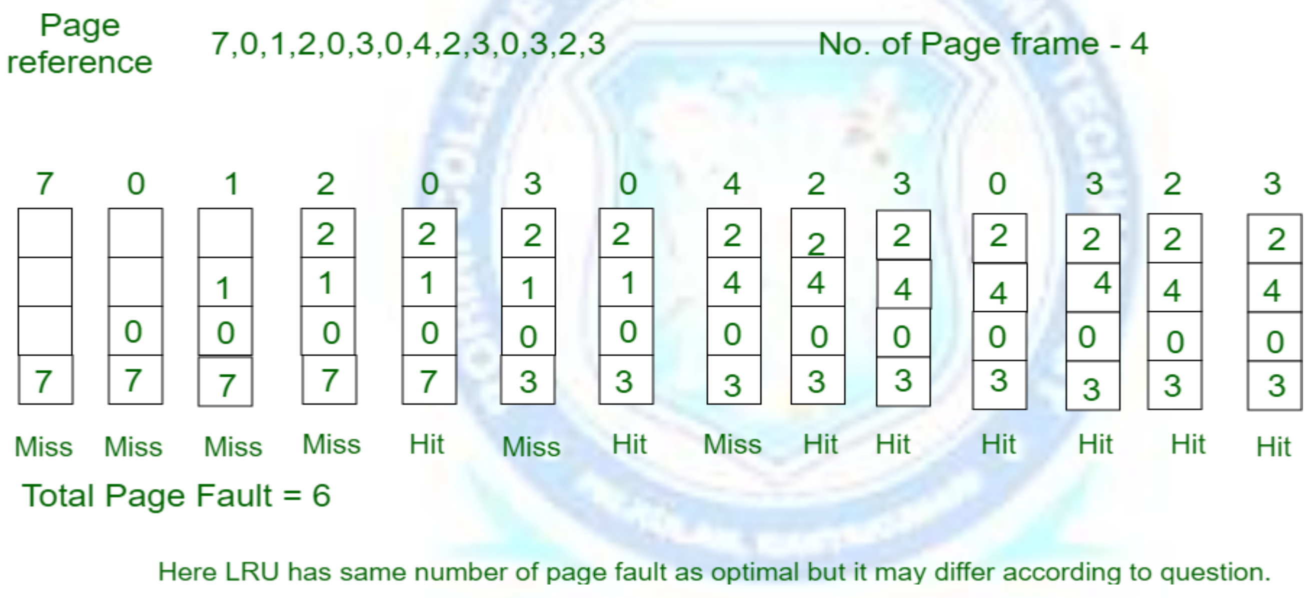
Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

- **Least Recently Used –**

  In this algorithm page will be replaced which is least recently used.

Example-3 Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 with 4 page frames. Find number of page faults.



Page reference
7,0,1,2,0,3,0,4,2,3,0,3,2,3
No. of Page frame - 4
7 0 1 2 0 3 0 4 2 3 0 3 2 3
Miss Miss Miss Miss Hit Miss Hit Miss Hit Hit Hit Hit Hit Hit
Total Page Fault = 6
Here LRU has same number of page fault as optimal but it may differ according to question.

Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**

0 is already their so —> **0 Page fault.**

when 3 came it will take the place of 7 because it is least recently used —>**1 Page fault**

0 is already in memory so —> **0 Page fault**. 4 will takes place of 1 —>

1 Page Fault

Now for the further page reference string —> **0 Page fault** because they are already available in the memory.