What is Virtual Memory in OS (Operating System)?

Virtual Memory is a storage scheme that provides user an illusion of having a very big main memory. This is done by treating a part of secondary memory as the main memory.

In this scheme, User can load the bigger size processes than the available main memory by having the illusion that the memory is available to load the process.

Instead of loading one big process in the main memory, the Operating System loads the different parts of more than one process in the main memory.

By doing this, the degree of multiprogramming will be increased and therefore, the CPU utilization will also be increased.

## **How Virtual Memory Works?**

In modern word, virtual memory has become quite common these days. In this scheme, whenever some pages needs to be loaded in the main memory for the execution and the memory is not available for those many pages, then in that case, instead of stopping the pages from entering in the main memory, the OS search for the RAM area that are least used in the recent times or that are not referenced and copy that into the secondary memory to make the space for the new pages in the main memory.

Since all this procedure happens automatically, therefore it makes the computer feel like it is having the unlimited RAM.

### **Advantages of Virtual Memory**

1. The degree of Multiprogramming will be increased.
2. User can run large application with less real RAM.
3. There is no need to buy more memory RAMs.

### **Disadvantages of Virtual Memory**

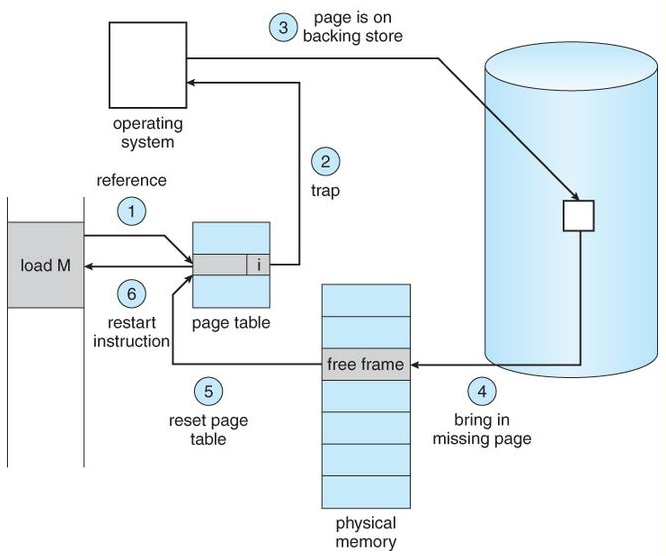
1. The system becomes slower since swapping takes time.
2. It takes more time in switching between applications.
3. The user will have the lesser hard disk space for its use.

Virtual memory is a memory management technique that allows programs to use more memory than is physically available in RAM. Page faults are an essential part of virtual memory systems.

A page fault occurs when a process cannot access a page. This can be due to a lack of access privileges or the page not being loaded into RAM.

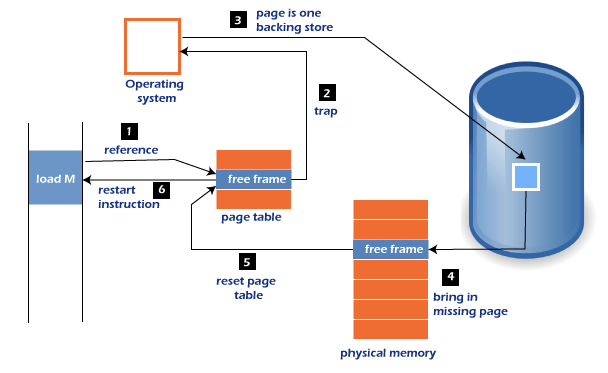
When a page fault occurs, the operating system's kernel allocates or denies RAM access to specific processes. The operating system then retrieves the required page from disk and brings it into RAM.

Page faults are common and often helpful to improve performance by raising the quantum of memory available for programs.



## **What is Page Fault in Operating System?**

Page faults dominate more like an **error**. A page fault will happen if a program tries to access a piece of memory that does not exist in physical memory (main memory). The fault specifies the operating system to trace all data into virtual memory management and then relocate it from secondary memory to its primary memory, such as a hard disk.



A page fault trap occurs if the requested page is not loaded into memory. The page fault primarily causes an exception, which is used to notify the operating system to retrieve the **"pages"** from virtual memory to continue operation. Once all of the data has been placed into physical memory, the program resumes normal operation. The Page fault process occurs in the background, and thus the user is unaware of it.

1. The computer's hardware track to the kernel and the program counter is often saved on the stack. The CPU registers hold information about the current state of instruction.
2. An assembly program is started, which saves the general registers and other volatile data to prevent the Operating system from destroying it.

## **Page Fault Handling**

A Page Fault happens when you access a page that has been marked as invalid. The paging hardware would notice that the invalid bit is set while translating the address across the page table, which will cause an operating system trap. The trap is caused primarily by the OS's failure to load the needed page into memory.

Now, let's understand the procedure of page fault handling in the OS:

1. Firstly, an internal table for this process to assess whether the reference was valid or invalid memory access.
2. If the reference becomes invalid, the system process would be terminated. Otherwise, the page will be paged in.
3. After that, the free-frame list finds the free frame in the system.
4. Now, the disk operation would be scheduled to get the required page from the disk.
5. When the I/O operation is completed, the process's page table will be updated with a new frame number, and the invalid bit will be changed. Now, it is a valid page reference.
6. If any page fault is found, restart these steps from starting.

## **Page Fault Terminology**

There are various page fault terminologies in the operating system. Some terminologies of page fault are as follows:

**1. Page Hit**

When the CPU attempts to obtain a needed page from main memory and the page exists in **main memory (RAM)**, it is referred to as a **"PAGE HIT"**.

**2. Page Miss**

If the needed page has not existed in the **main memory (RAM)**, it is known as **"PAGE MISS"**.

**3. Page Fault Time**

The time it takes to get a page from secondary memory and recover it from the main memory after loading the required page is known as **"PAGE FAULT TIME"**.

**4. Page Fault Delay**

The rate at which threads locate page faults in memory is referred to as the **"PAGE FAULT RATE"**. The page fault rate is measured per second.

**5. Hard Page Fault**

If a required page exists in the hard disk's page file, it is referred to as a **"HARD PAGE FAULT"**.

**6. Soft Page Fault**

**7. Minor Page Fault**

If a process needs data and that data exists in memory but is being allotted to another process at the same moment, it is referred to as a **"MINOR PAGE FAULT"**.

* The computer hardware traps to the kernel and program counter (PC) is saved on the stack. Current instruction state information is saved in CPU registers.
* An assembly program is started to save the general registers and other volatile information to keep the OS from destroying it.
* Operating system finds that a page fault has occurred and tries to find out which virtual page is needed. Some times hardware register contains this required information. If not, the operating system must retrieve PC, fetch instruction and find out what it was doing when the fault occurred.
* Once virtual address caused page fault is known, system checks to see if address is valid and checks if there is no protection access problem.
* If the virtual address is valid, the system checks to see if a page frame is free. If no frames are free, the page replacement algorithm is run to remove a page.
* If frame selected is dirty, page is scheduled for transfer to disk, context switch takes place, fault process is suspended and another process is made to run until disk transfer is completed.
* As soon as page frame is clean, operating system looks up disk address where needed page is, schedules disk operation to bring it in.
* When disk interrupt indicates page has arrived, page tables are updated to reflect its position, and frame marked as being in normal state.
* Faulting instruction is backed up to state it had when it began and PC is reset. Faulting is scheduled, operating system returns to routine that called it.
* Assembly Routine reloads register and other state information, returns to user space to continue execution.

