Paper Name : Operating System

Prepared by : Debanjali Jana

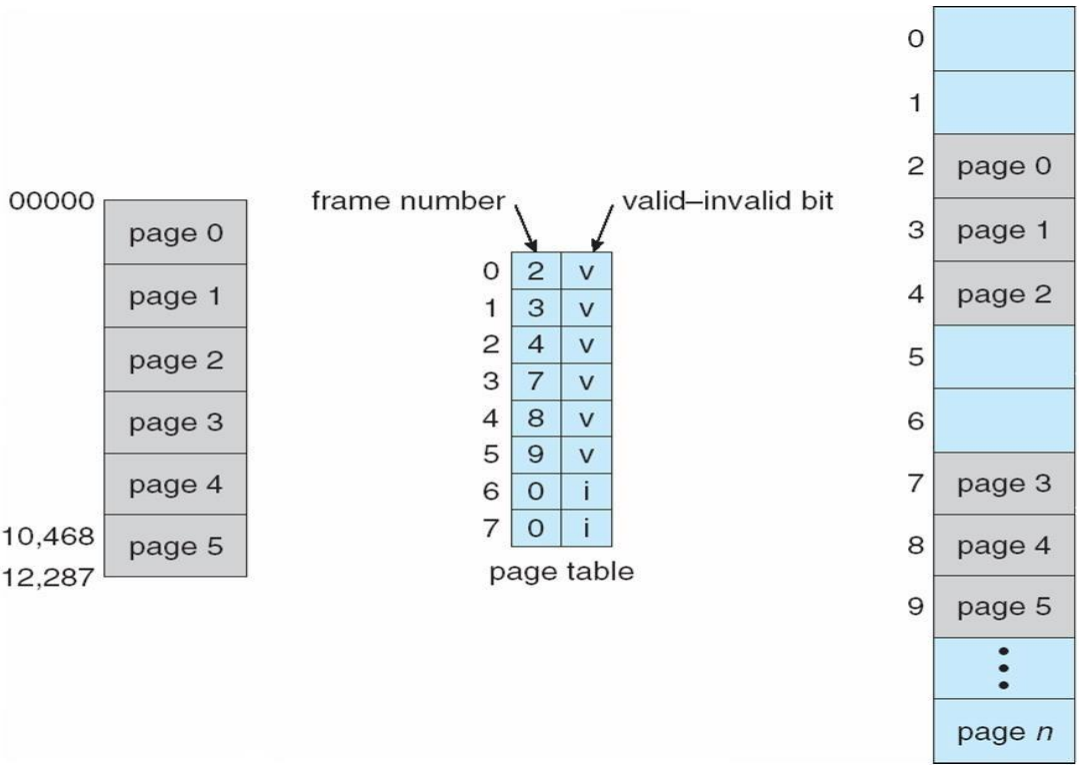
Memory Management :

Memory protection in a paged environment –

Memory protection in a paged environment is accomplished by protection bits associated with each frame to indicate if read-only or read-write access is allowed. Normally, these bits are kept in the page table.

Valid-invalid bit attached to each entry in the page table:

- "valid" indicates that the associated page is in the process' logical address space, and is thus a legal page
- "invalid" indicates that the page is not in the process' logical address space



Suppose, for example, that in a system with a 14-bit address space (0 to 16383), we have a program that should use only addresses 0 to 10468. Given a page size of 2 KB, addresses in pages 0, 1, 2, 3, 4, and 5 are mapped normally through the page table. Any attempt to generate an address in pages 6 or 7, however, will find that the valid -invalid bit is set to invalid, and the computer will trap to flee operating system (invalid page reference). Notice that this scheme has created a problem. Because the program extends only to address 10468, any reference beyond that address is illegal. However references to page 5 are classified as valid, so accesses to addresses up to 12287 are valid. Only the addresses from 12288 to 16383 are invalid.

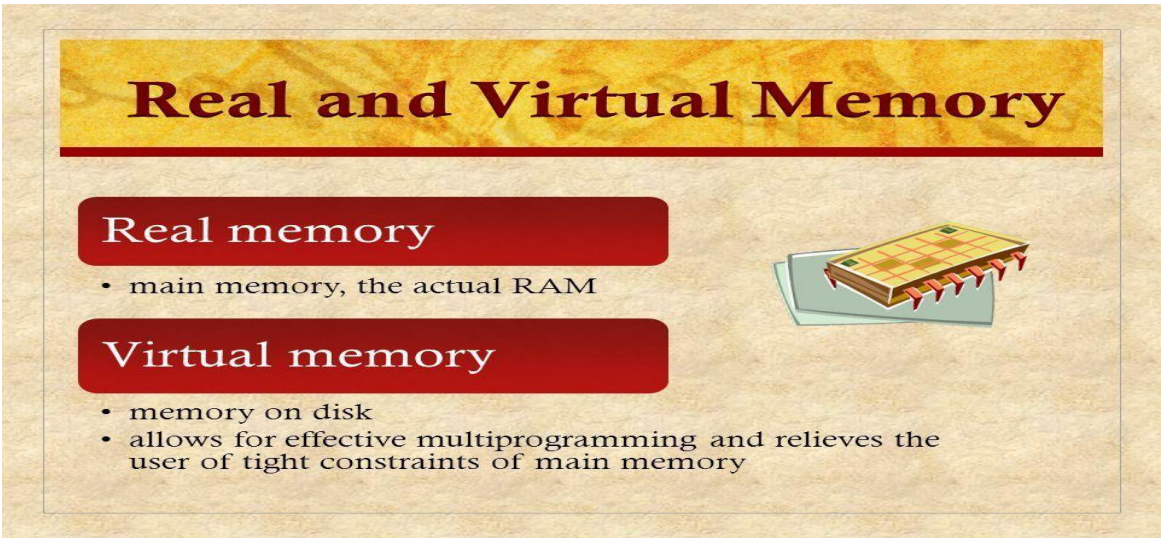
Some systems provide hardware, in the form of a **page-table length register** (**PTLR**) to indicate the size of the page table. This value is checked against every logical address to verify that the address is in the valid range for the process. Failure of this test causes an error trap to the operating system.

Virtual Memory

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of secondary memory storage space that's set up to emulate the computer's RAM. The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address.

Virtual memory – separation of user logical memory from physical memory

- Only part of the program needs to be in memory for execution
- Logical address space can therefore be much larger than physical address space
- Allows address spaces to be shared by several processes
- Allows for more efficient process creation
- More programs running concurrently
- Less I/O needed to load or swap processes



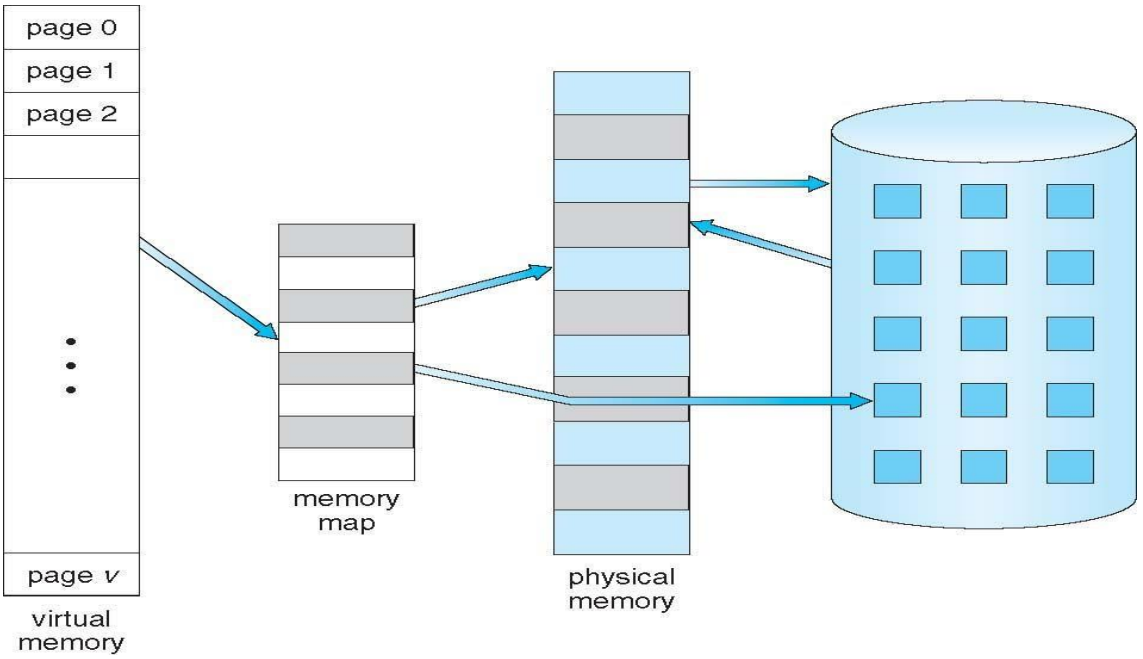


Figure: Virtual memory that is larger than physical memory

Virtual memory can be implemented via:

- Demand paging
- Demand segmentation

Demand Paging

Demand paging is an application of virtual memory. In a system that uses demand paging, the operating system copies a disk page into physical memory only if an attempt is made to access it (i.e., if a page fault occurs). It follows that a process begins execution with none of its pages in physical memory, and many page faults will occur until most of a process's working set of pages is located in physical memory. This is an example of lazy loading techniques. The page table indicates if the page is on disk or memory using a valid bit. Once a page is brought from disk into memory, the OS updates the page table and the valid bit.

- In this technique a page is brought into memory for its execution only when it is demanded
- It is a combination of paging and swapping

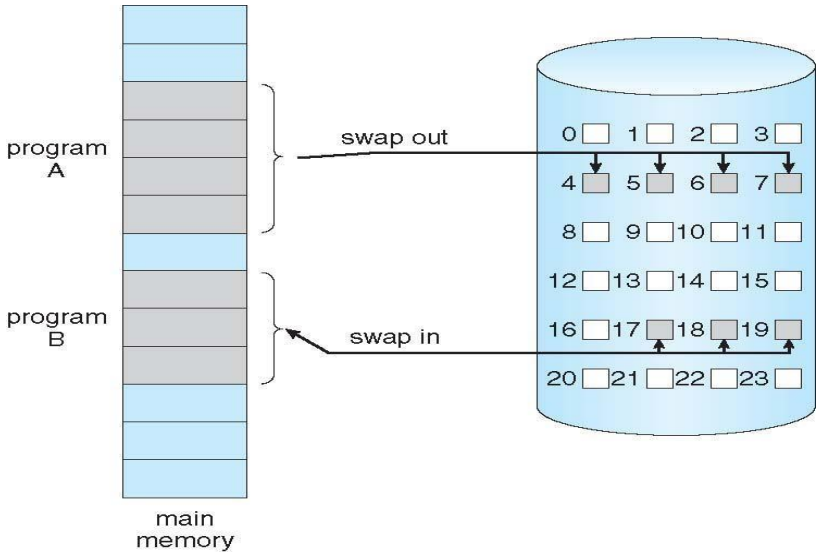


Figure: Demand Paging

Here in the above diagram, all the pages are loaded into backing store (hard disk). By the mechanism of swapping when the main memory requests the page Only then it is loaded from hard disk As main memory is small in size and cannot handle large programs only few pages are loaded into main memory after completing its execution it is swapped out simply and new process is then swapped in.

Basic concepts of demand paging:

When a process is to be swapped in, the pager guesses which pages will be used before the process is swapped out again. Instead of swapping in a whole process, the pager brings only those pages into memory. Thus, it avoids reading into memory pages that will not be used anyway, decreasing the swap time and the amount of physical memory needed. With this scheme, we need some form of hardware support to distinguish between the pages that are in memory and the pages that are on the disk. The valid–invalid bit scheme can be used for this purpose. This time, however, when this bit is set to "valid," the associated page is both legal and in memory. If the bit is set to "invalid," the page either is not valid (that is, not in the logical address space of the process) or is valid but is currently on the disk. The page-table entry for a page that is brought into memory is set as usual, but the page-table entry for a page that is not currently in memory is either simply marked invalid or contains the address of the page on disk. Marking a page invalid will have no effect if the process never attempts to access that page.

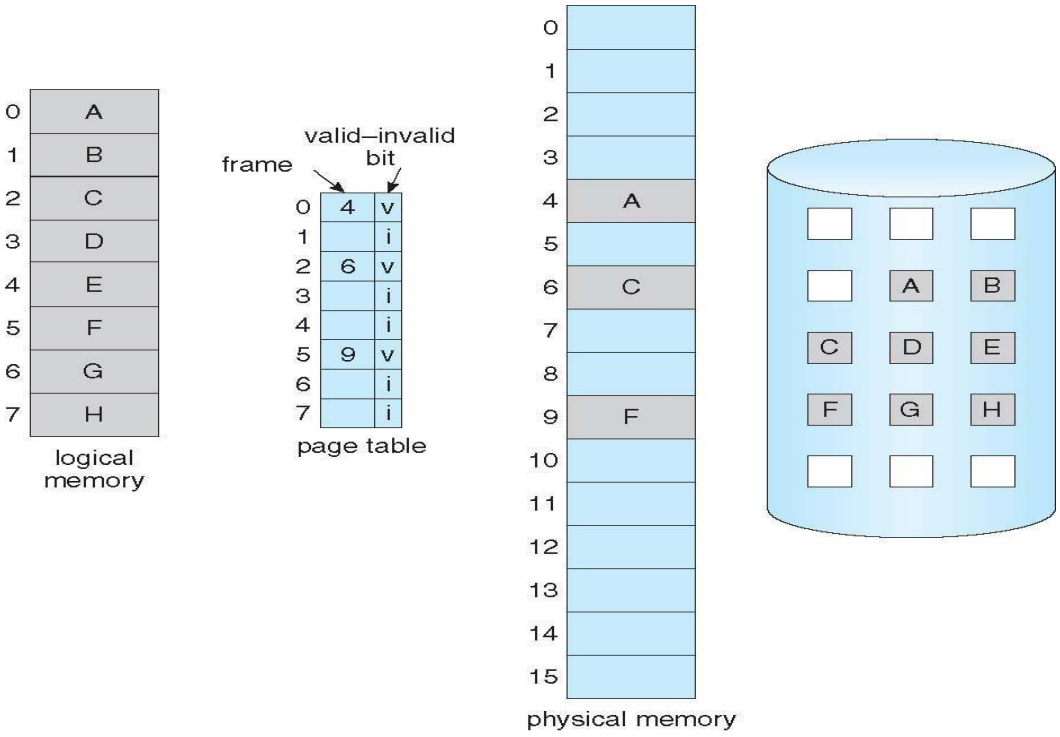


Figure: Page table when some pages are not in main memory

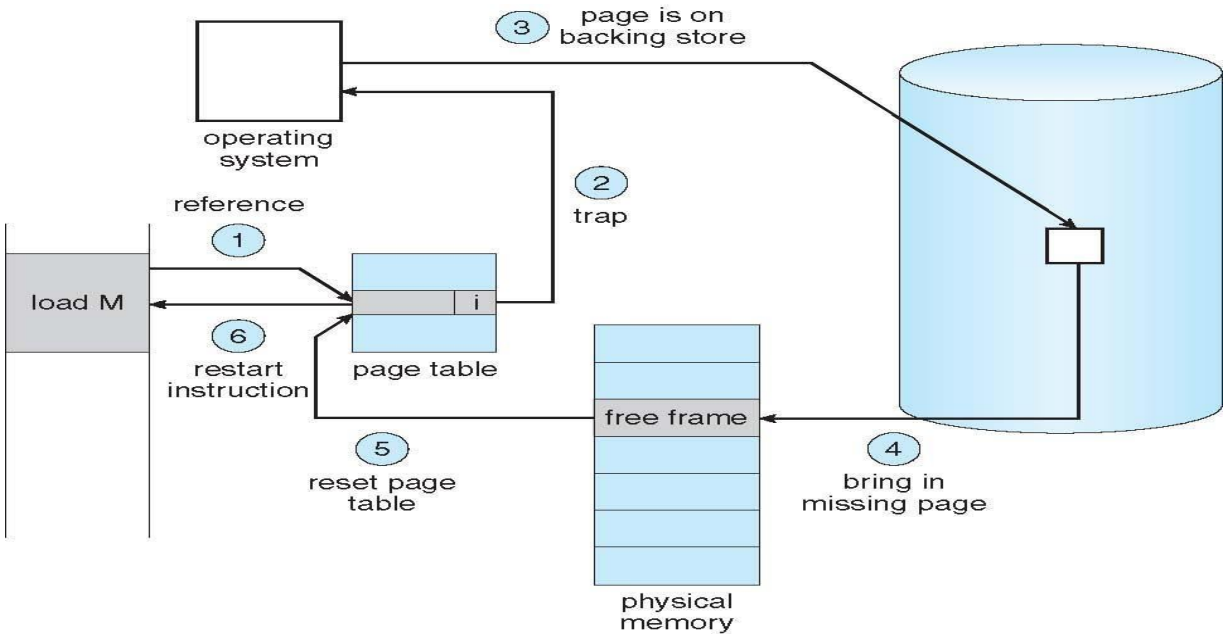


Figure: Steps in handling page fault

But what happens if the process tries to access a page that was not brought into memory? Access to a page marked invalid causes a **page fault**. The paging hardware, in translating the address through the page table, will notice that the invalid bit is set, causing a trap to the operating system. This trap is the result of the operating system's failure to bring the desired page into memory. The procedure for handling this page fault is :

1. We check an internal table (usually kept with the process control block) for this process to determine whether the reference was a valid or an invalid memory access.
2. If the reference was invalid, we terminate the process. If it was valid, but we have not yet brought in that page, we now page it in.
3. We find a free frame (by taking one from the free-frame list, for example).
4. We schedule a disk operation to read the desired page into the newly allocated frame.
5. When the disk read is complete, we modify the internal table kept with the process and the page table to indicate that the page is now in memory.
6. We restart the instruction that was interrupted by the trap. The process can now access the page as though it had always been in memory.

In the extreme case, we can start executing a process with no pages in memory. When the operating system sets the instruction pointer to the first instruction of the process, which is on a non-memory-resident page, the process immediately faults for the page. After this page is brought into memory, the process continues to execute, faulting as necessary until every page that it needs is in memory. At that point it can execute with no more faults. This scheme is known as **pure demand paging** : never bring a page into memory until it is required.

Advantages of demand paging :

• Demand paging does not load the pages that are never accessed, so saves the memory for other programs and increases the degree of multiprogramming.
• There is less loading latency at the program startup.
• There is less of initial disk overhead because of fewer page reads.
• It does not need extra hardware support than what paging needs, since protection fault can be used to get page fault.
• Pages will be shared by multiple programs until they are modified by one of them, so a technique called copy on write will be used to save more resources.
• Ability to run large programs on the machine, even though it does not have sufficient memory to run the program. This method is easier for a programmer than an old manual overlays.

Disadvantages of demand paging :

• Individual programs face extra latency when they access a page for the first time. So prepaging, a method of remembering which pages a process used when it last executed and preloading a few of them, is used to improve performance.
• Programs running on low-cost, low-power embedded systems may not have a memory management unit that supports page replacement.
• Memory management with page replacement algorithms becomes slightly more complex.