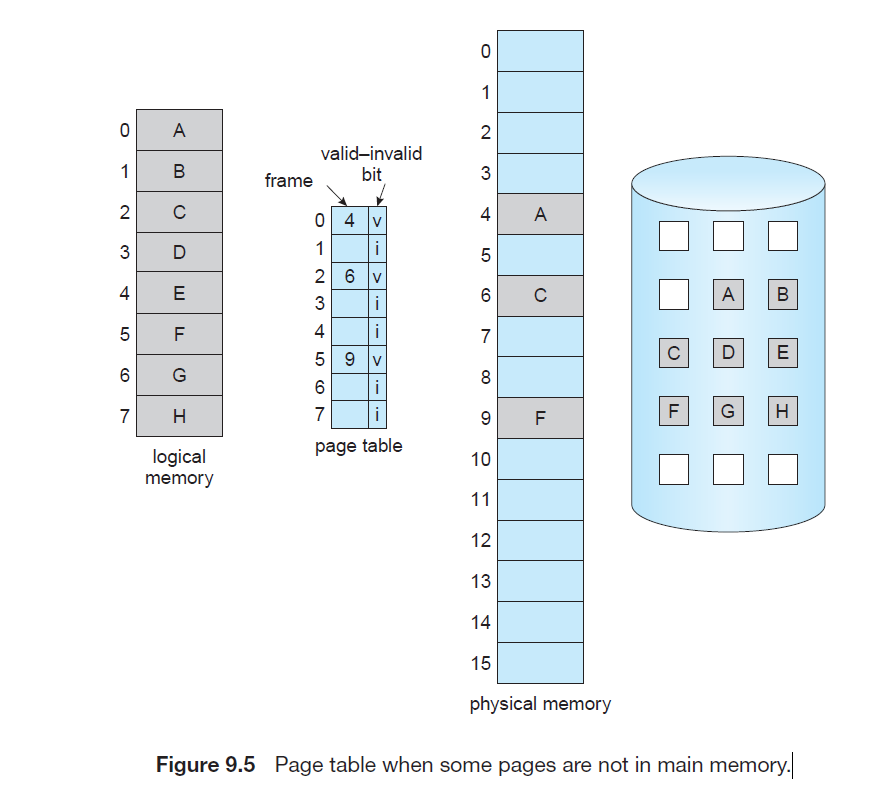
**DEMAND PAGING**

Consider how an executable program might be loaded from disk into memory. One option is to load the entire program in physical memory at program execution time. However, a problem with this approach is that we may not initially ***need*** the entire program in memory.

An alternative strategy is to load pages only as they are needed. This technique is known as **demand paging** and is commonly used in virtual memory systems. With demand-paged virtual memory, pages are loaded only when they are demanded by the CPU during program execution. Hence, it is also named as lazy swapper because the swapping of pages is done only when required by the CPU.

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.

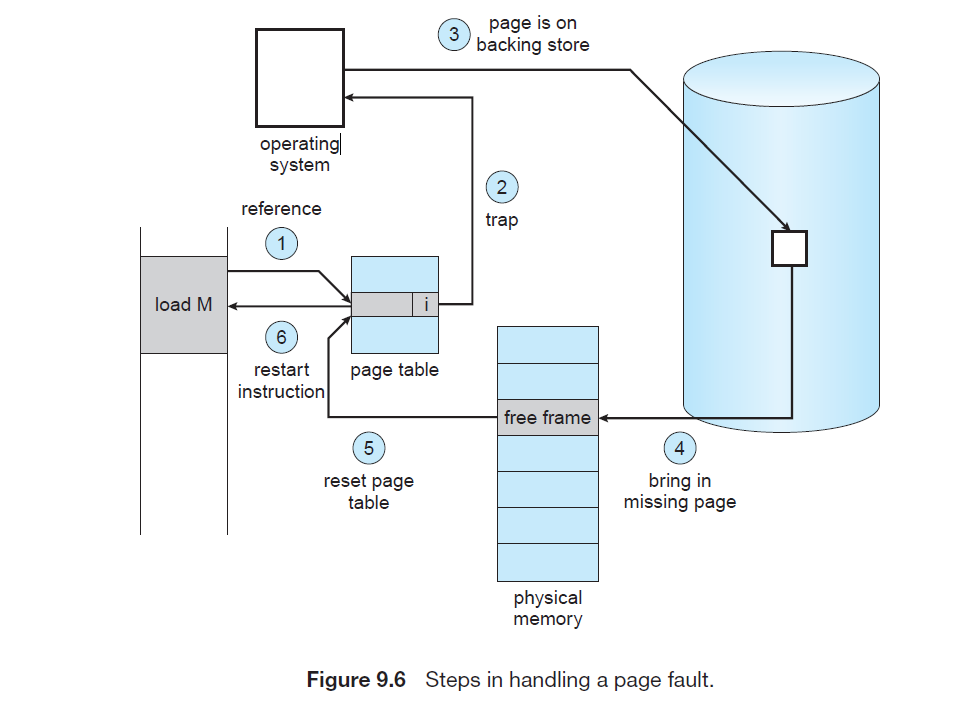
With this scheme, we need some form of hardware support to distinguish between the pages that are in memory and the pages that are on the disk. The valid–invalid bit scheme can be used for this purpose. This time, however, when this bit is set to “valid,” the associated page is both legal and in memory. If the bit is set to “invalid,” the page either is not exist (that is, not in the logical address space of the process) or is valid but is currently on the disk.



**Page Fault:**

When the page referenced by the CPU is not found in the main memory then the situation is termed as Page Fault. Whenever any page fault occurs, then the required page has to be fetched from the secondary memory into the main memory.

**Demand Paging working:**

****

whenever a page fault occurs these steps are followed by the operating system and the required page is brought into memory.

1. Now, if the CPU wants to access page of a process P, first it will search the page in the page table.
2. As the page table does not contain this page so it will be a **trap**or **page fault**. As soon as the trap is generated and context switching happens and the control goes to the operating system.
3. The OS system will put the process in a waiting/ blocked state. The OS system will now search that page in the backing store or secondary memory.
4. The OS will then read the page from the backing store and load it to the main memory.
5. Next, the OS system will update the page table entry accordingly.
6. Finally, the control is taken back from the OS and the execution of the process is resumed.

**Page Fault Service time:** So whenever a page fault occurs all the above steps(2–6) are performed. This time taken to service the page fault is called the Page fault service time.

**Pure Demand Paging:**

In some cases, we can start executing a process with ***no*** pages in main memory, pages in such cases are only loaded when are demanded by the process by generating page faults. It is then referred to as **pure demand paging**: never bring a page into memory until it is required.