PRINCIPLES OF OPERATING SYSTEMS

LECTURE- 16
Virtual Memory- Demand paging

Introduction

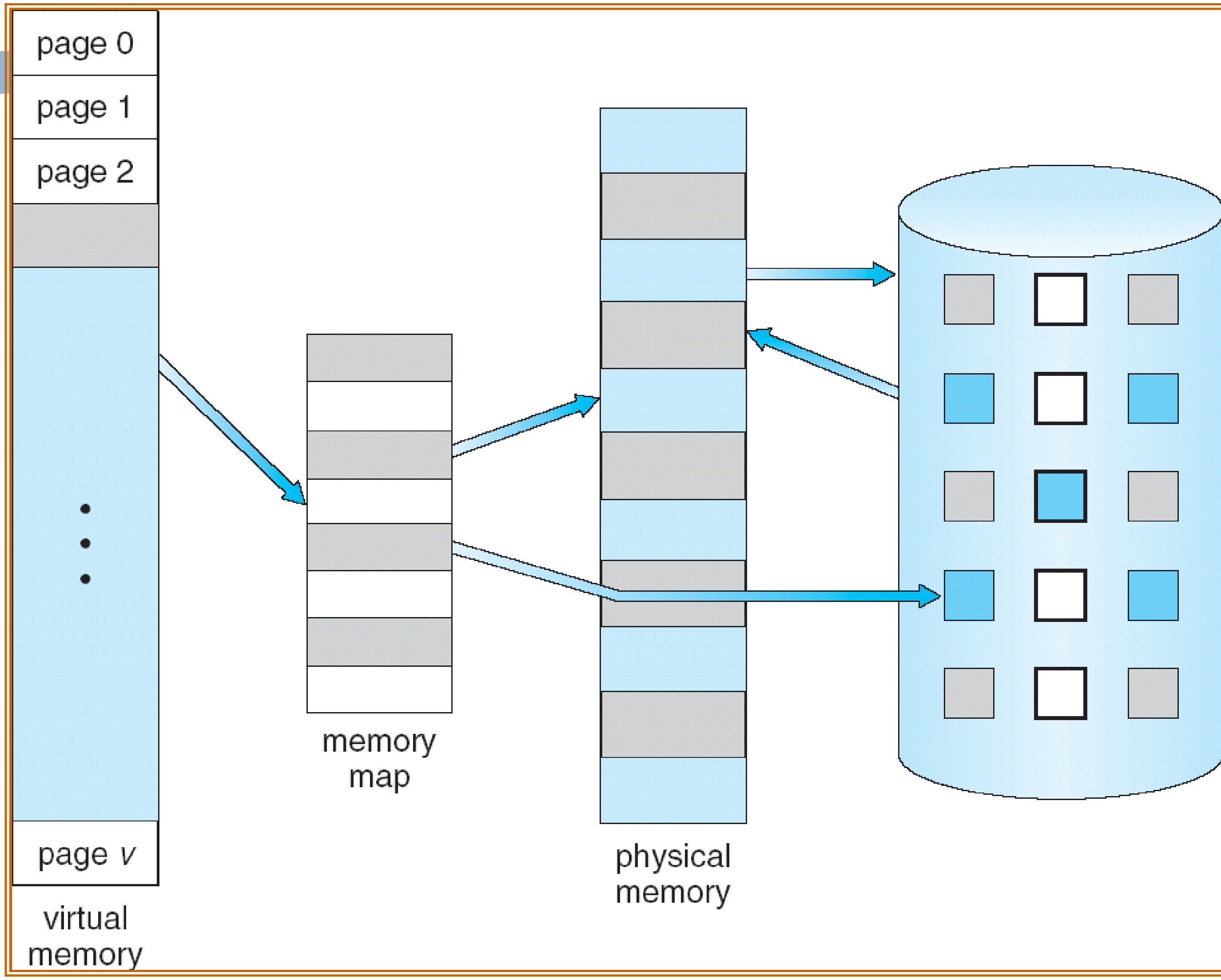
Virtual memory – separation of user logical memory from physical memory.

- 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

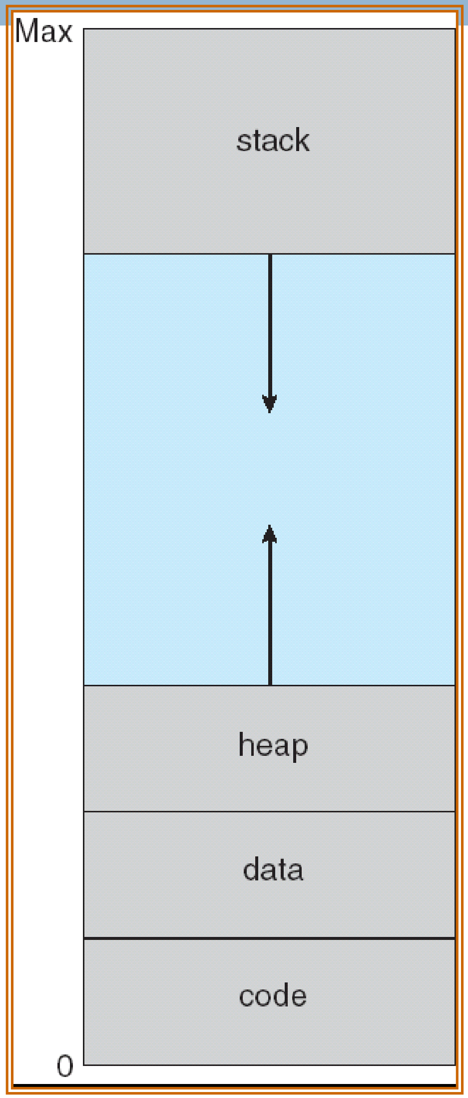
Introduction

- Virtual memory can be implemented via:
  - Demand paging
  - Demand segmentation

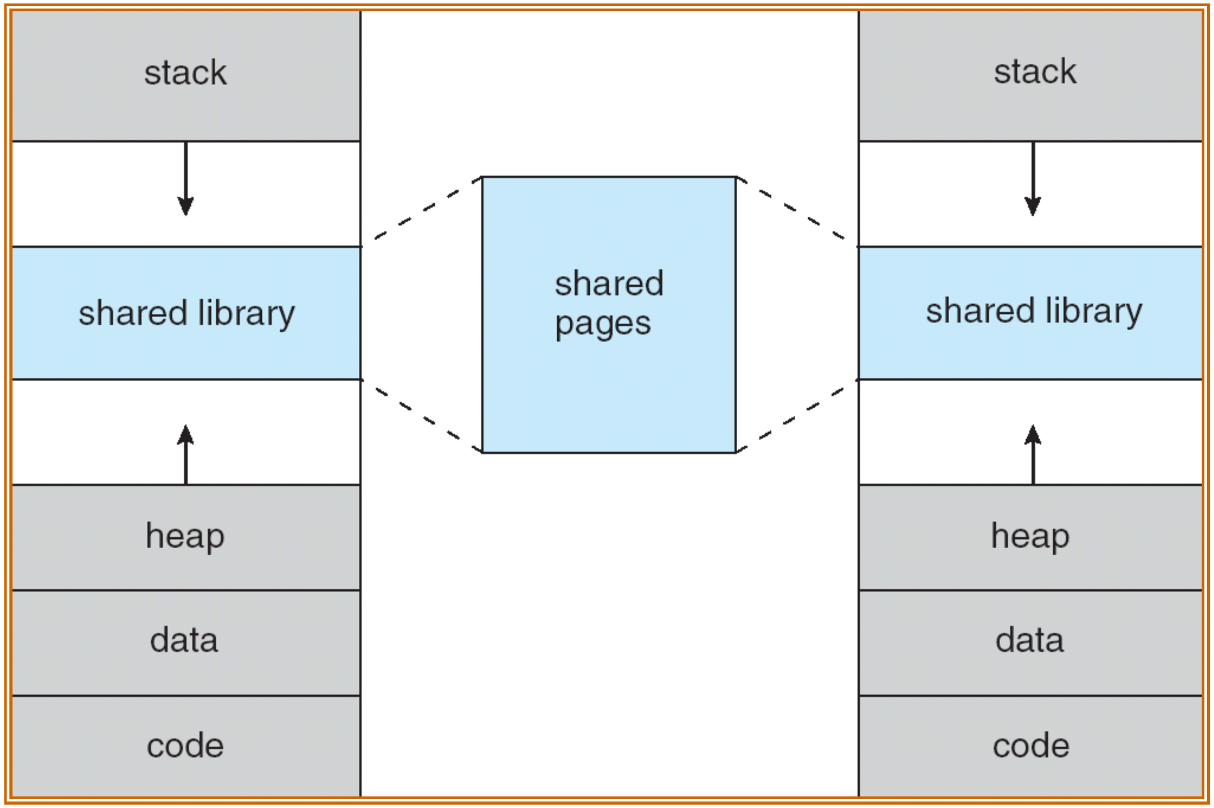
Virtual Memory That is Larger Than Physical Memory



Virtual-address Space



Shared Library Using Virtual Memory



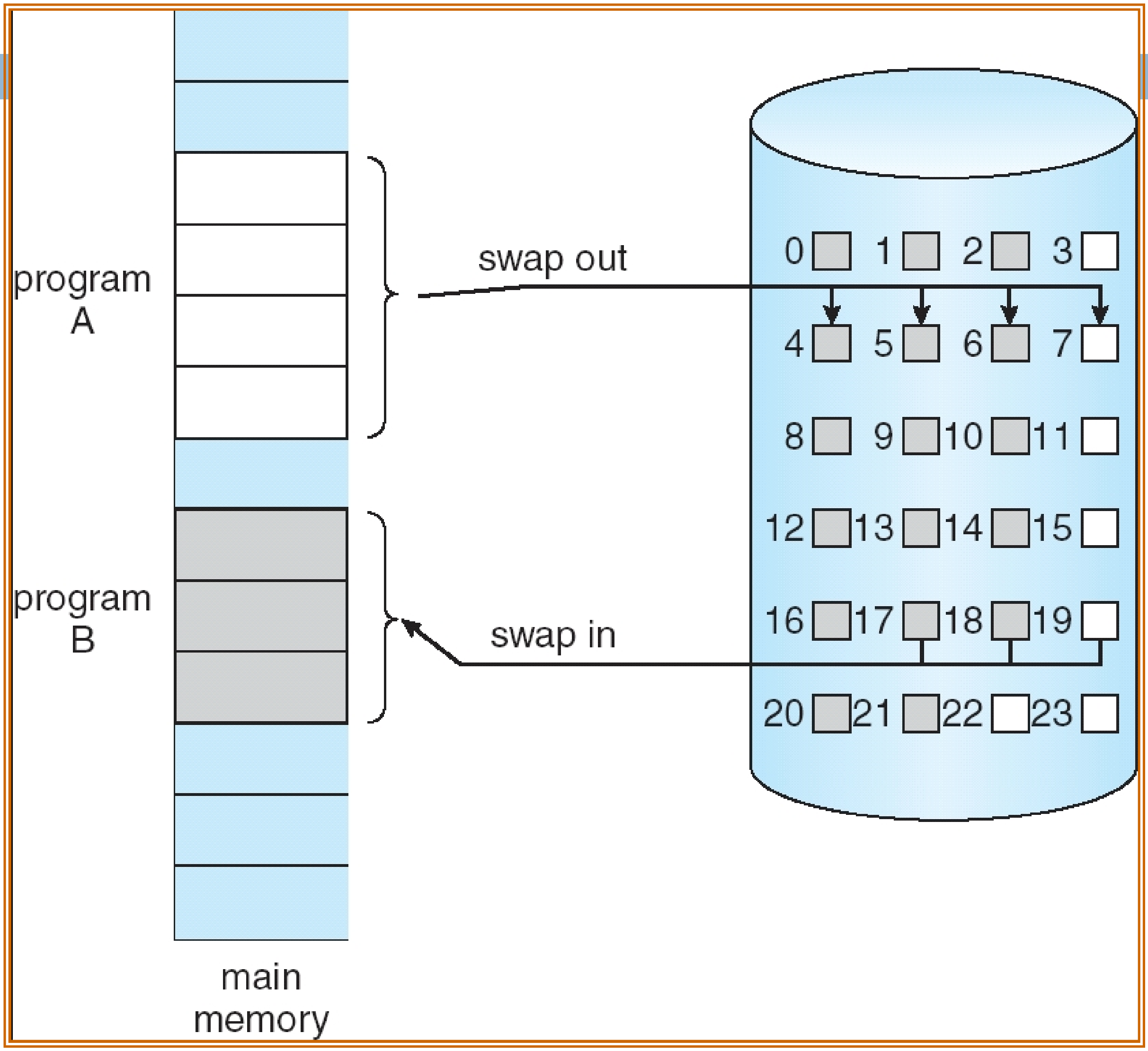
Demand Paging

- Bring a page into memory only when it is needed
  - Less I/O needed
  - Less memory needed
  - Faster response
  - More users

Demand Paging

- Page is needed $\Rightarrow$ reference to it
  - invalid reference $\Rightarrow$ abort
  - not-in-memory $\Rightarrow$ bring to memory
- **Lazy swapper** – never swaps a page into memory unless page will be needed
  - Swapper that deals with pages is a **pager**

Transfer of a Paged Memory to Contiguous Disk Space



Valid-Invalid Bit

- With each page table entry a valid–invalid bit is associated
  (**v** ⇒ in-memory, **i** ⇒ not-in-memory)

- Initially valid–invalid bit is set to **i** on all entries

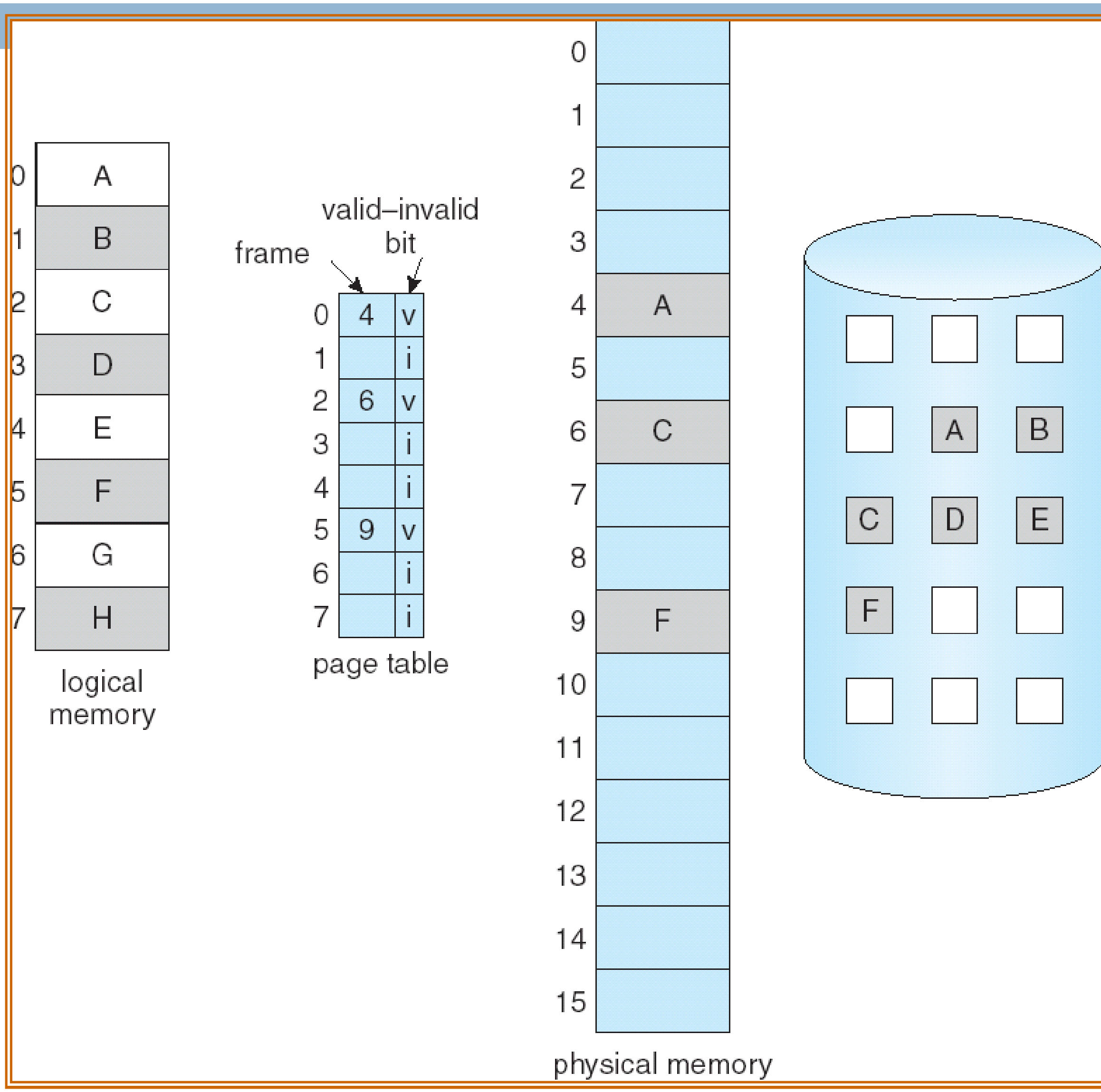
- During address translation, if valid–invalid bit in page table entry is **i** ⇒ page fault

....

Frame #	valid-invalid bit
	v
	v
	v
	v
	i
	i
	i
	i

page table

Page Table When Some Pages Are Not in Main Memory



Page Fault

If there is a reference to a page, and the referenced page is not in memory, but the page is a valid page in the process's virtual memory, then it is a **page fault**.

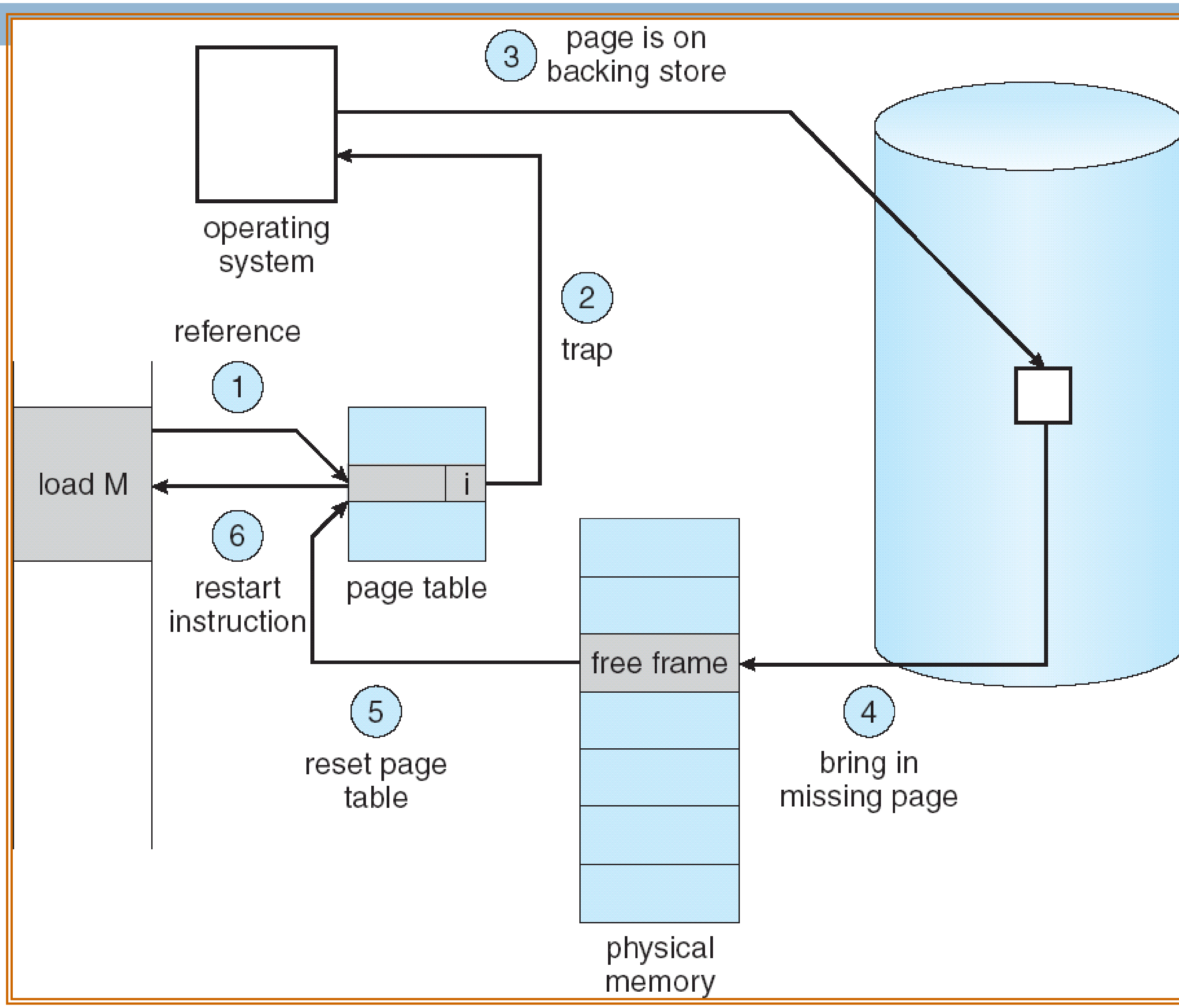
First reference to that page will trap to operating system **page fault.**

Page Fault

1. Operating system looks at another table (may be captured in PCB) to decide:
   - Invalid reference $\Rightarrow$ abort
   - Just not in memory
2. If (there is a free frame)
   1. Get empty frame

3. Swap page into frame
4. Reset tables
5. Set validation bit = **v**
6. Restart the instruction that caused the page fault

Steps in Handling a Page Fault



Performance of Demand Paging

- Page Fault Rate $0 \leq p \leq 1.0$
  - if $p = 0$ ➔ no page faults
  - if $p = 1$ ➔ every reference is a fault

- Effective Access Time (EAT)

$$
\begin{aligned}
\text{EAT} = (1 - p) \text{ x memory access} \\
+\, p \text{ x } (\quad & \text{1. page fault overhead} \\
& + \text{2. swap page out} \\
& + \text{3. swap page in} \\
& + \text{4. restart overhead})
\end{aligned}
$$

Demand Paging Example

- Memory access time = 200 nanoseconds

- Average page-fault service time = 8 milliseconds

- EAT = (1 – p) x 200 + p (8 milliseconds)
  = (1 – p) x 200 + p x 8,000,000
  = 200 + p x 7,999,800

- If one access out of 1,000 causes a page fault, then
  EAT = 8.2 microseconds.
  This is a slowdown by a factor of 40!!