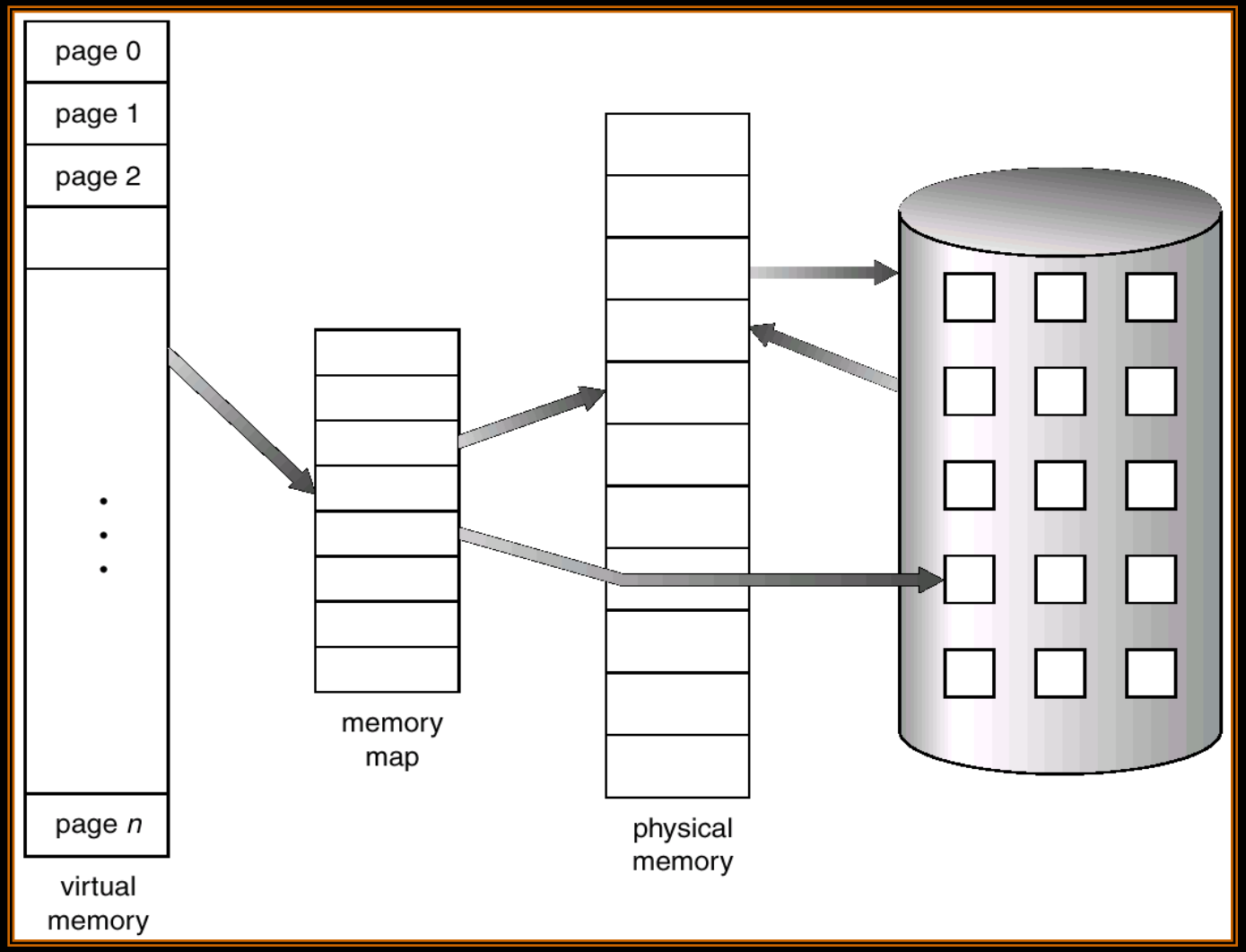
VIRTUAL MEMORY

Virtual Memory That is Larger Than Physical Memory



OS Policies

(1) Fetch Policy: It determines **when a page should be brought** into main memory

- **Pre Paging**
- **Demand Paging**

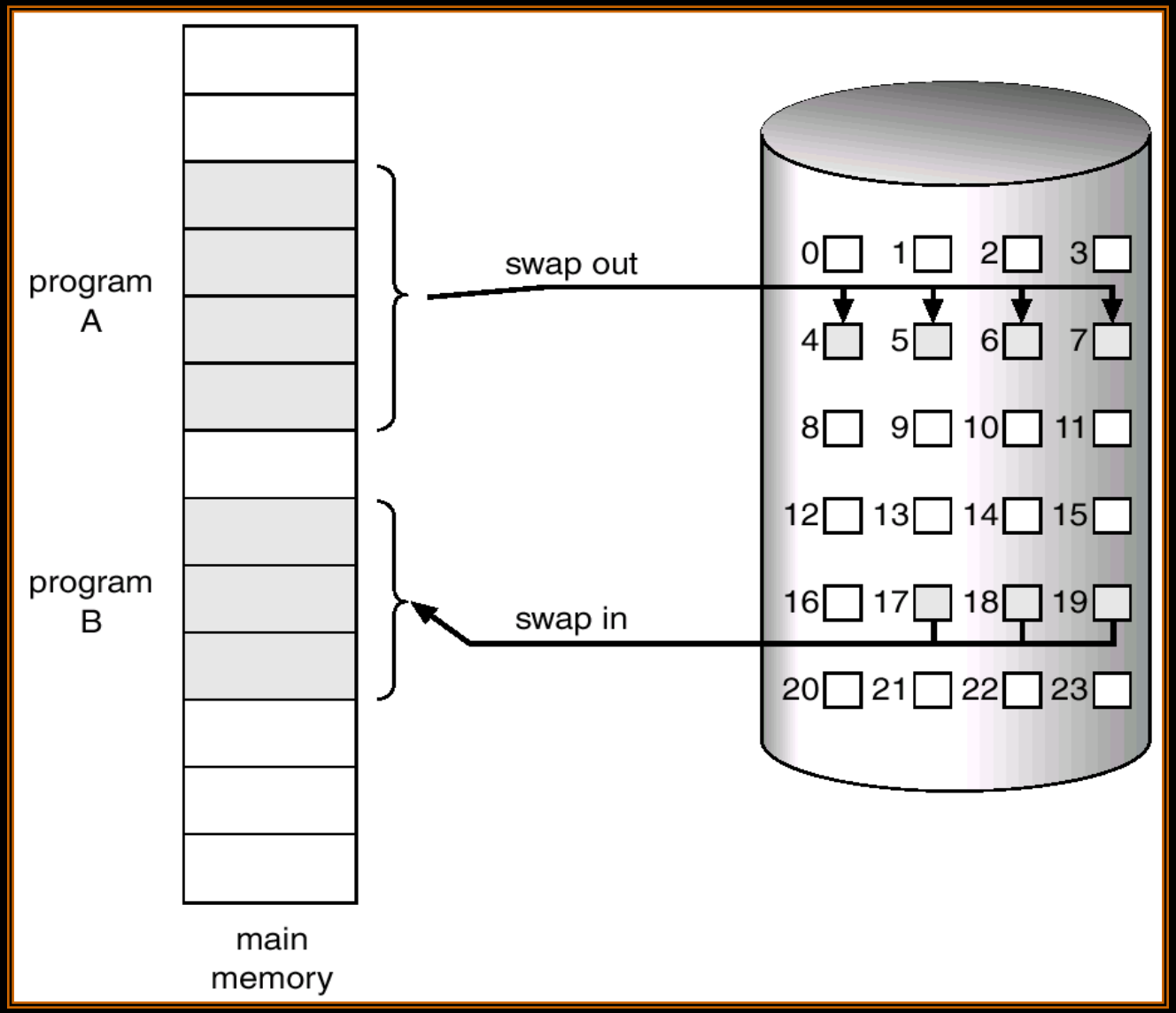
(2) Placement Policy: It determines **,where in real memory a process piece** is to be **reside (First Fit, Best Fit, Worst Fit)**

(3) Replacement Policy: It deals with **selection of page** in **memory** to be **replaced**, when a page must be brought in

Demand Paging

- Bring a page into memory only when it is needed.
  - Less memory needed
  - More users
- If there is a reference to a page, then Check
  - invalid reference ⇒ abort
  - not-in-memory (page fault) ⇒ bring to memory

Transfer of a Paged Memory to Contiguous Disk Space



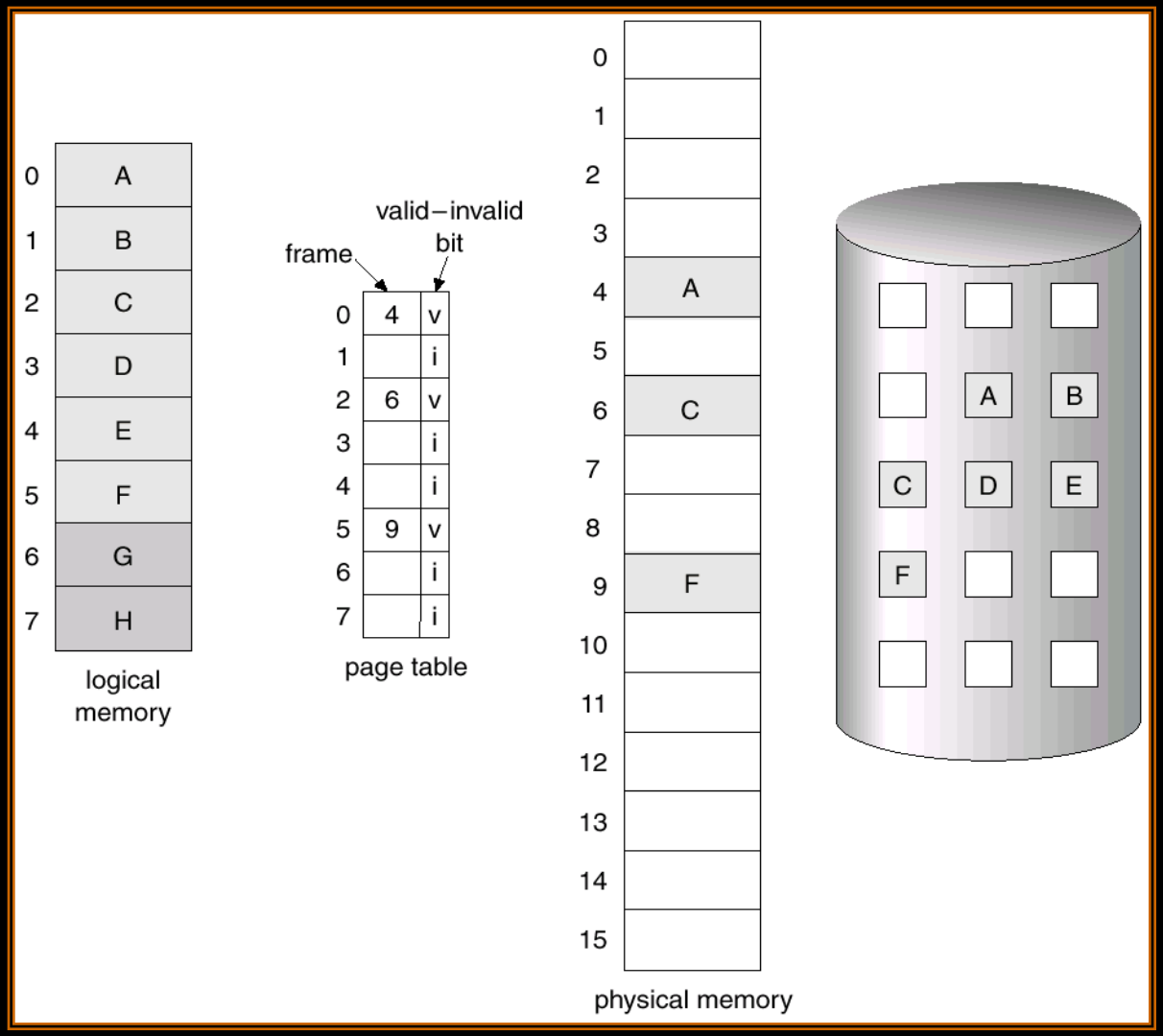
Valid-Invalid Bit

- With each page table entry, a valid–invalid bit is associated (1 ⇒ in-memory, 0 ⇒ not-in-memory)

- Initially valid–invalid bit is set to 0 on all entries.

- During address translation, if valid–invalid bit in page table entry is 0 ⇒ page fault.

Page Table When Some Pages Are Not in Main Memory



Handling a Page Fault

❑ OS handles the page fault as follows:

1. Check an internal table (kept with PCB) to determine whether the reference was a valid or invalid memory access.

2. If the reference was invalid, TRAP the process to OS.

If it was valid, but the page need to be brought in.

3. Find a free frame.

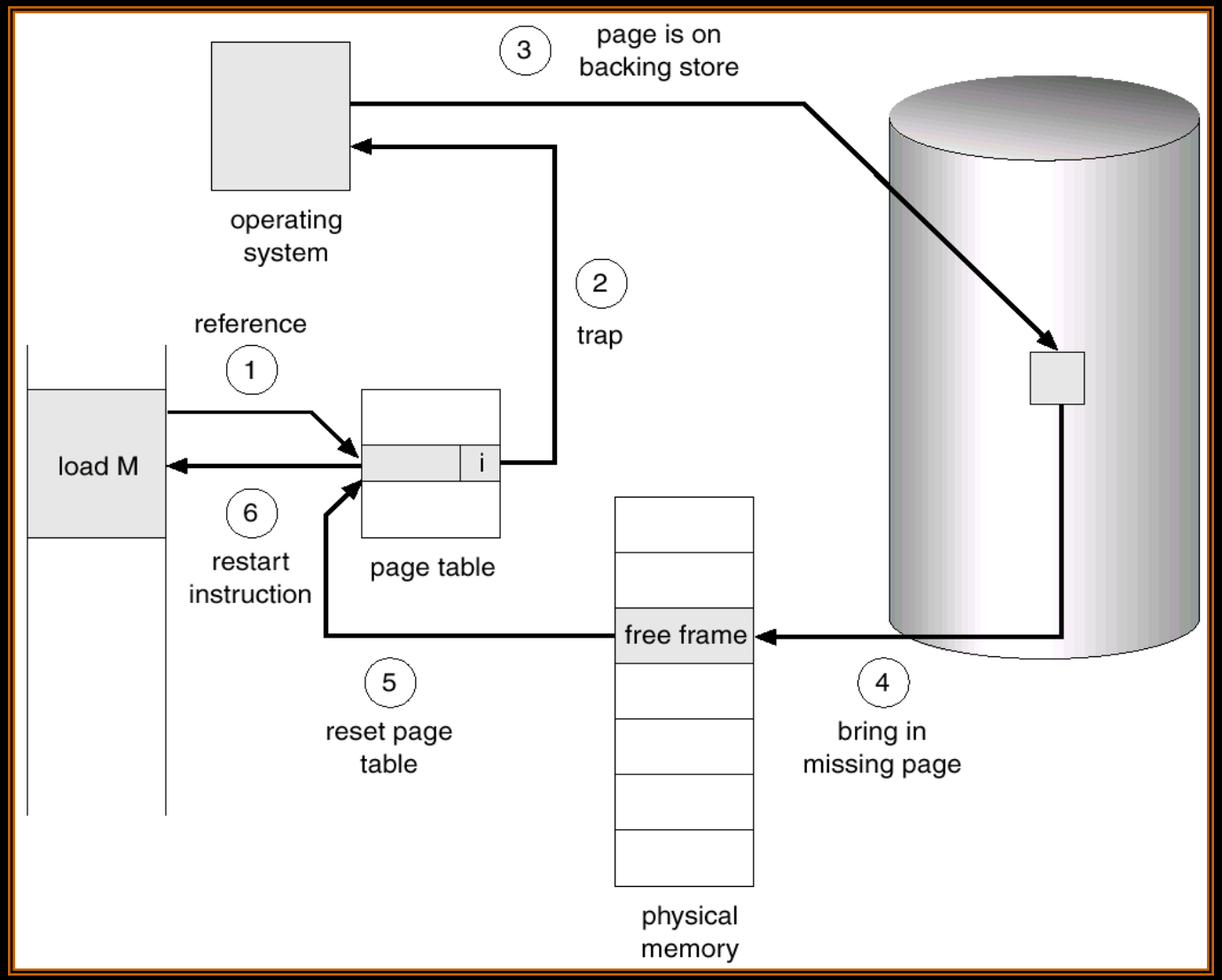
Handling a Page Fault

4. Schedule a disk operation to read the desired page into the newly allocated frame.

5. When the disk read is complete, modify the internal table in the PCB and the page table to indicate that the page is now in memory.

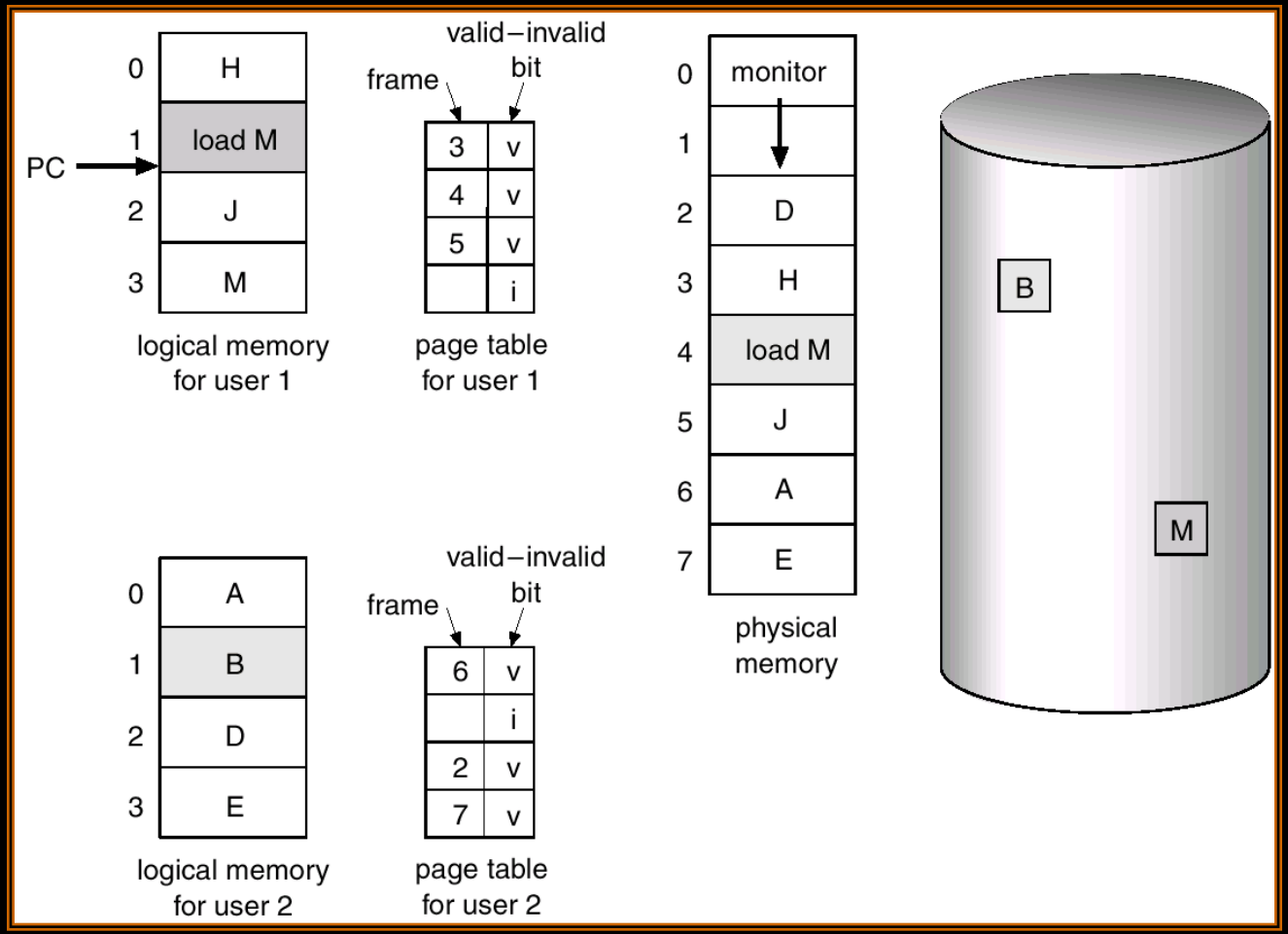
6. Restart the instruction that was interrupted by the page fault trap. The process can now access the page

Steps in Handling a Page Fault



What happens if there is no free frame?

Page replacement – find some **page** in memory, but **not really in use**, swap it out.



Basic Page Replacement

1. Find a free frame:
   - If there is a free frame, use it.

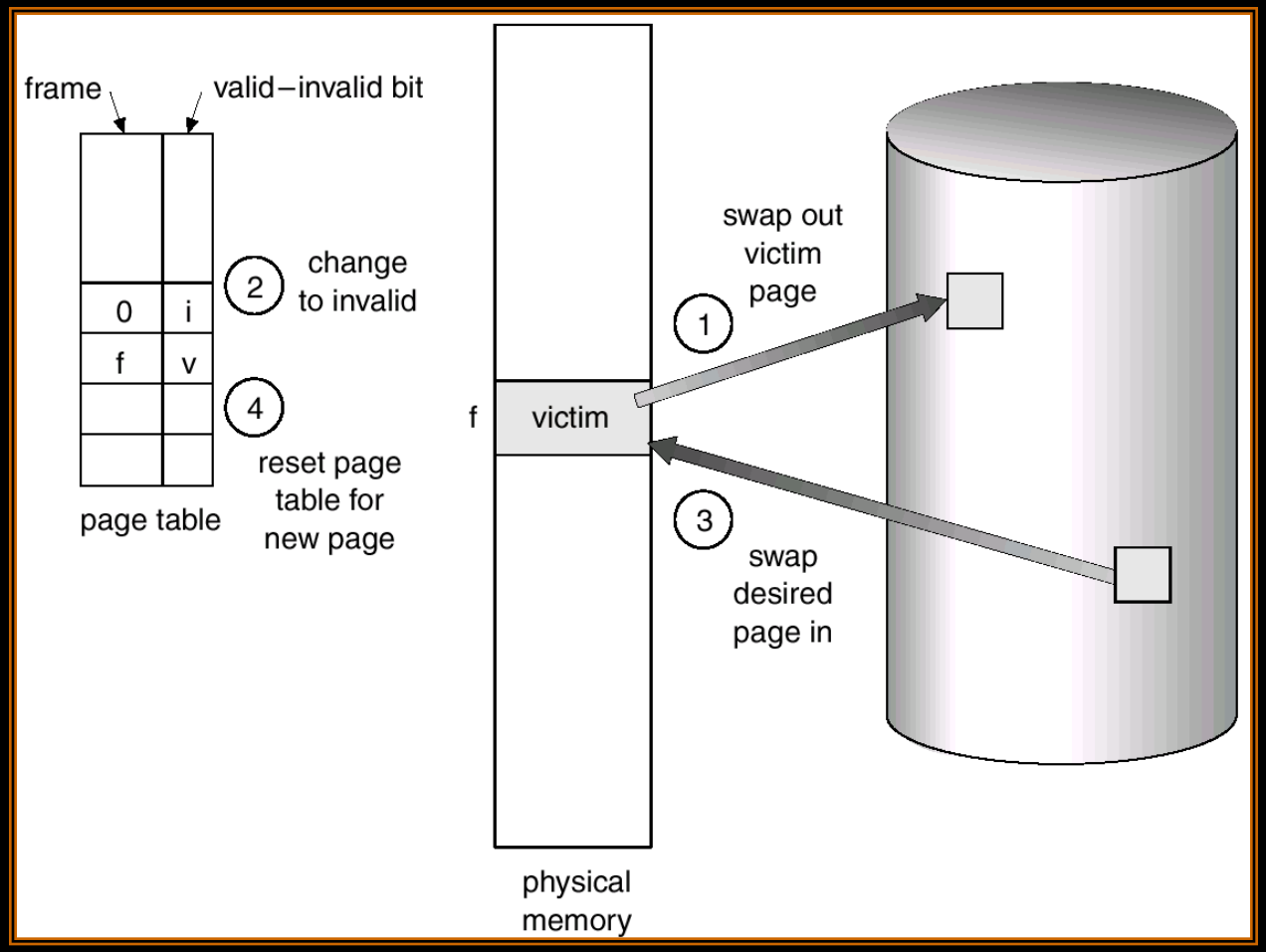
- If there is no free frame, use a page replacement algorithm to select a *victim* frame.

- Write the victim page to the disk; update the page and frame tables accordingly.

2. Read the desired page into the (newly) free frame.

Update the page and frame tables.

3. Restart the user process.



Page Replacement Algorithms

➢ The process of how to select a victim page taken care by replacement algorithm

- FIFO
- Optimal Page Replacement
- Least Recently Used
- Clock Page Replacement

1. FIFO

Replace a page which is the oldest page of all the pages of main memory

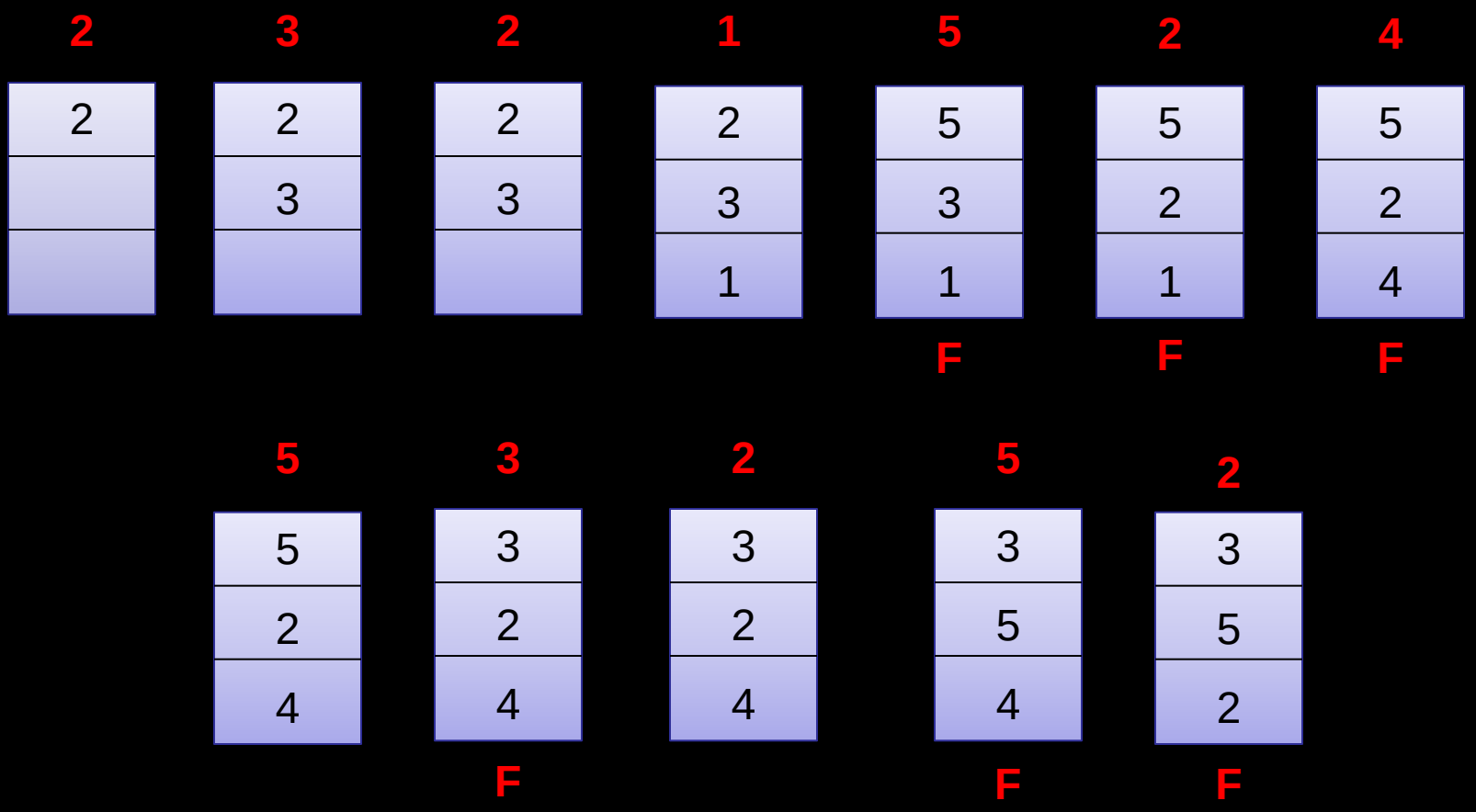
(or)

Replace the page that has been in the memory longest

EXAMPLE

Page address stream : 2 3 2 1 5 2 4 5 3 2 5 2

Frame Size: 3



➢The performance of the page replacement algorithm is measured through " **PAGE FAULT RATE**"

Page Fault Rate = No. of page Fault / no of bits in the address stream page

=> 6 / 12

=> 1/2

=> 0.5 %

2. Optimal Replacement

- Replace a page that will not be used for the longest period of time

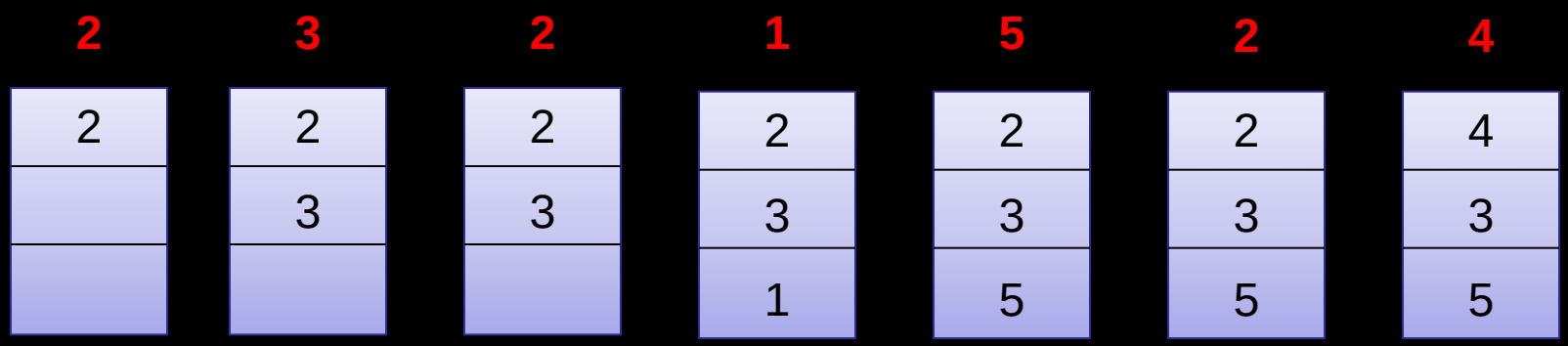
  {i.e. the time to the next reference is the longest}

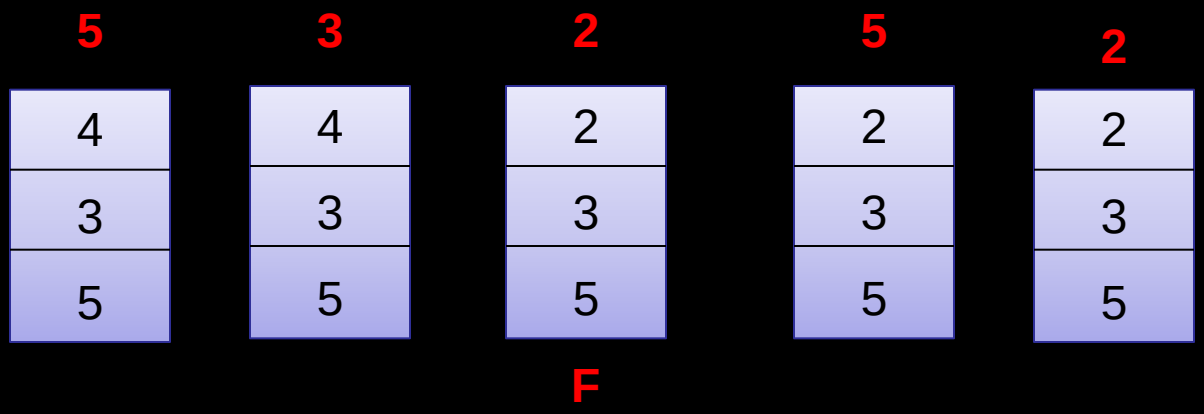
- Difficult to implement because it requires future knowledge of page streams

EXAMPLE

Page address stream : 2 3 2 1 5 2 4 5 3 2 5 2

Frame Size: 3





=> 3 / 12

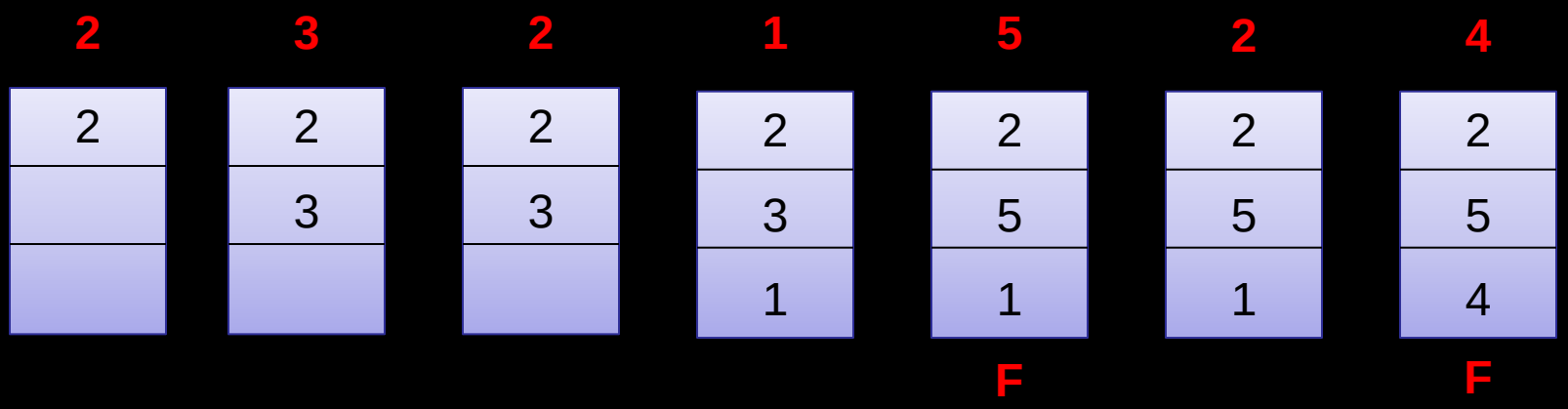
=> 0.25

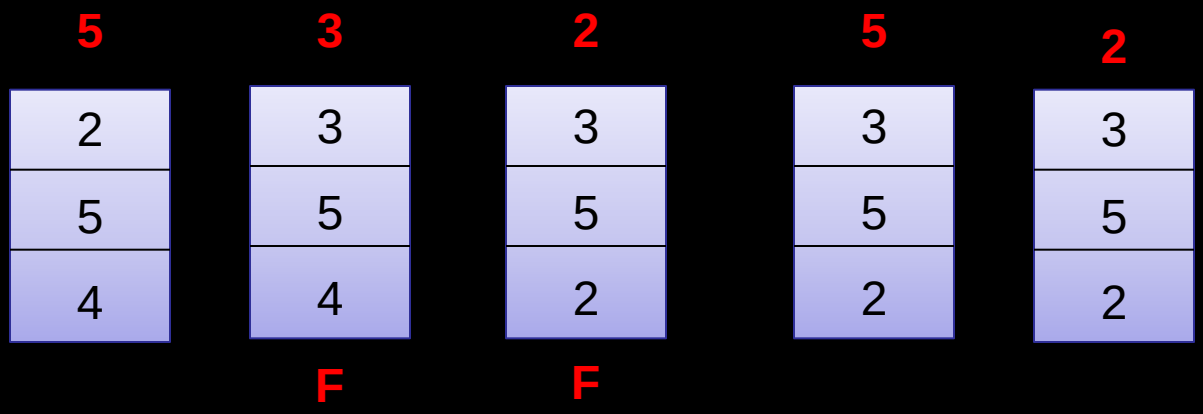
=> 25 %

3. Least Recently Used [LRU]

- **Replaces** the **page** in memory that has **not been referenced** for the **longest time**

{It **looking** **backward** in **time** rather than **forward**}





=> 4 / 12

=> 0.33

=> 33 %

THRASHING

- **Swapping out a piece of a process** just before that piece is needed

- The processor spends **most of its time swapping** pieces rather than executing user instructions

Thrashing ≡ a process is busy swapping pages in and out