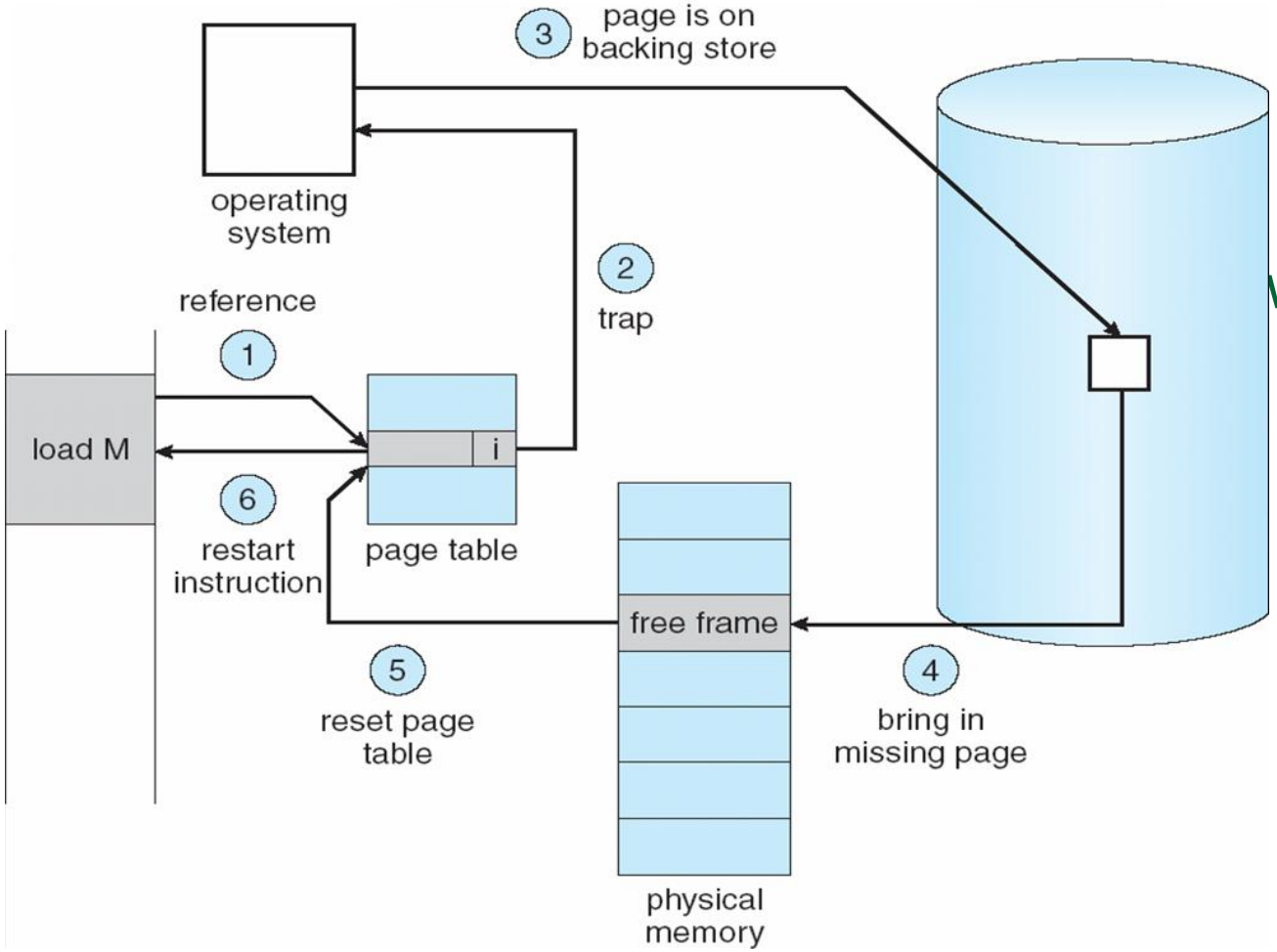
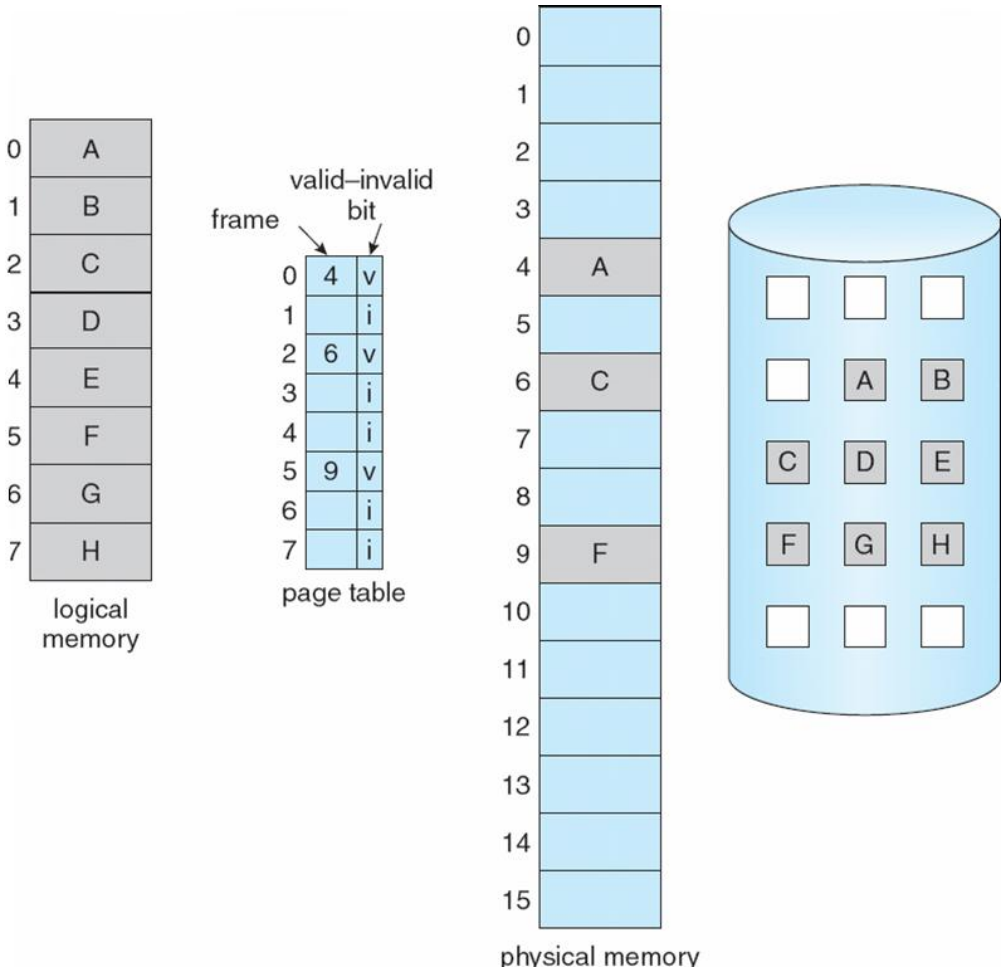
Contents

- Memory Management requirements
- Memory Partitioning: Fixed and Variable Partitioning,
- Allocation Strategies (First Fit, Best Fit, and Worst Fit), Fragmentation, Swapping.
- Virtual Memory: Concepts, Segmentation, Paging, Address Translation,
- Page Replacement Policies (FIFO, LRU, Optimal, Other Strategies),
- Thrashing.



What happens when there is no free frame?

Page Table When Some Pages Are Not in Main Memory



Page Fault

- If there is a reference to a page, first reference to that page will trap to operating system:

  page fault

1. Operating system looks at another table to decide:
   - Invalid reference $\Rightarrow$ abort
   - Just not in memory
2. 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

What happens if there is no free frame?

- Page replacement – find some page in memory, but not really in use, swap it out
  - Different algorithms, different performance
  - Want an algorithm which will result in minimum number of page faults
- Same page may be brought into memory several times

Page Replacement

- Prevent over-allocation of memory by modifying page-fault service routine to include page replacement

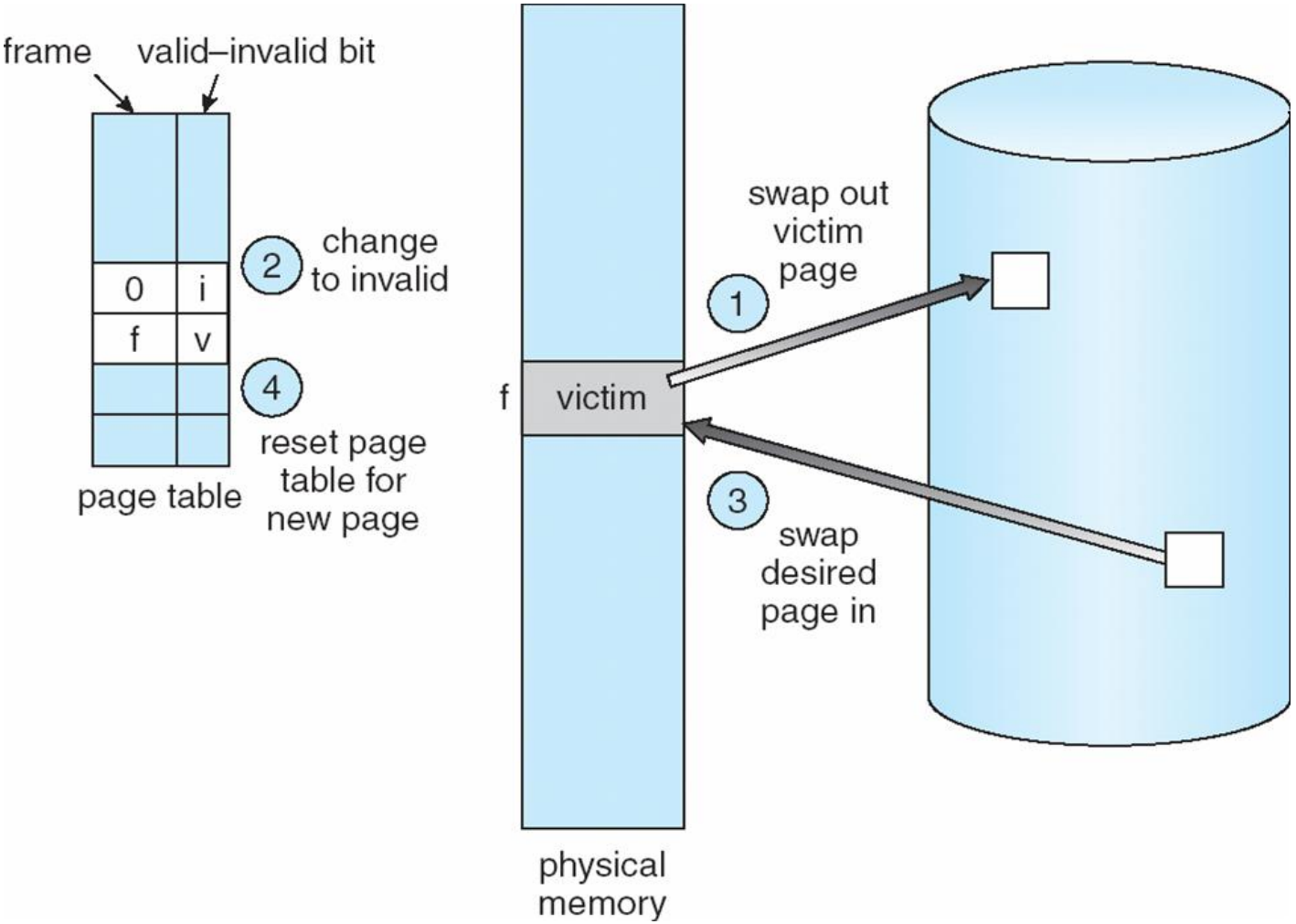
- Use **modify (dirty) bit** to reduce overhead of page transfers – only modified pages are written to disk

- Page replacement completes separation between logical memory and physical memory – large virtual memory can be provided on a smaller physical memory

Basic Page Replacement

1) Find the location of the desired page on disk
2) 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
3) Bring the desired page into the (newly) free frame; update the page and frame tables
4) Restart the process

Page Replacement



Page-Replacement Algorithms

- Want lowest page-fault rate.
- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string.
- In all our examples, the reference string is

A B C A B D A D B C B

Basic Replacement Algorithms

- First-in, first-out (FIFO)
  - Treats page frames allocated to a process as a circular buffer
  - Pages are removed in round-robin style
  - Simplest replacement policy to implement
  - Page that has been in memory the longest is replaced
  - These pages may be needed again very soon
  - How would you implement FIFO strategy?
    - Keep a queue and do round robin

Example: FIFO

- Suppose we have 3 page frames, 4 virtual pages, and following reference stream:
  - A B C A B D A D B C B
- Consider FIFO Page replacement:

Ref: / Page:	A	B	C	A	B	D	A	D	B	C	B
1	A					D				C	
2		B					A				
3			C						B		

  - FIFO: 7 faults.
  - When referencing D, replacing A is bad choice, since need A again right away

Basic Replacement Algorithms

- Optimal policy
  - Selects for replacement that page for which the time to the next reference is the longest
  - Impossible to have perfect knowledge of future events

Example: Optimal(MIN)

- Suppose we have the same reference stream:
  - A B C A B D A D B C B
- Consider Optimal Page replacement:

Ref: / Page:	A	B	C	A	B	D	A	D	B	C	B
1	A									C	
2		B									
3			C			D					

  - MIN: 5 faults
  - Where will D be brought in? Look for page not referenced farthest in future.

Basic Replacement Algorithms

- Least Recently Used (LRU)
  - Replaces the page that has not been referenced for the longest time
  - By the principle of locality, this should be the page least likely to be referenced in the near future
  - Each page could be tagged with the time of last reference. This would require a great deal of overhead.

Example: LRU

- Suppose we have the same reference stream:
  - A B C A B D A D B C B
- Consider LRU Page replacement:

Ref: / Page:	A	B	C	A	B	D	A	D	B	C	B
1	A									C	
2		B									
3			C			D					

  - LRU: 5 faults
- What will LRU do?
  - Same decisions as MIN here, but won't always be true!

LRU Algorithm (Cont.)

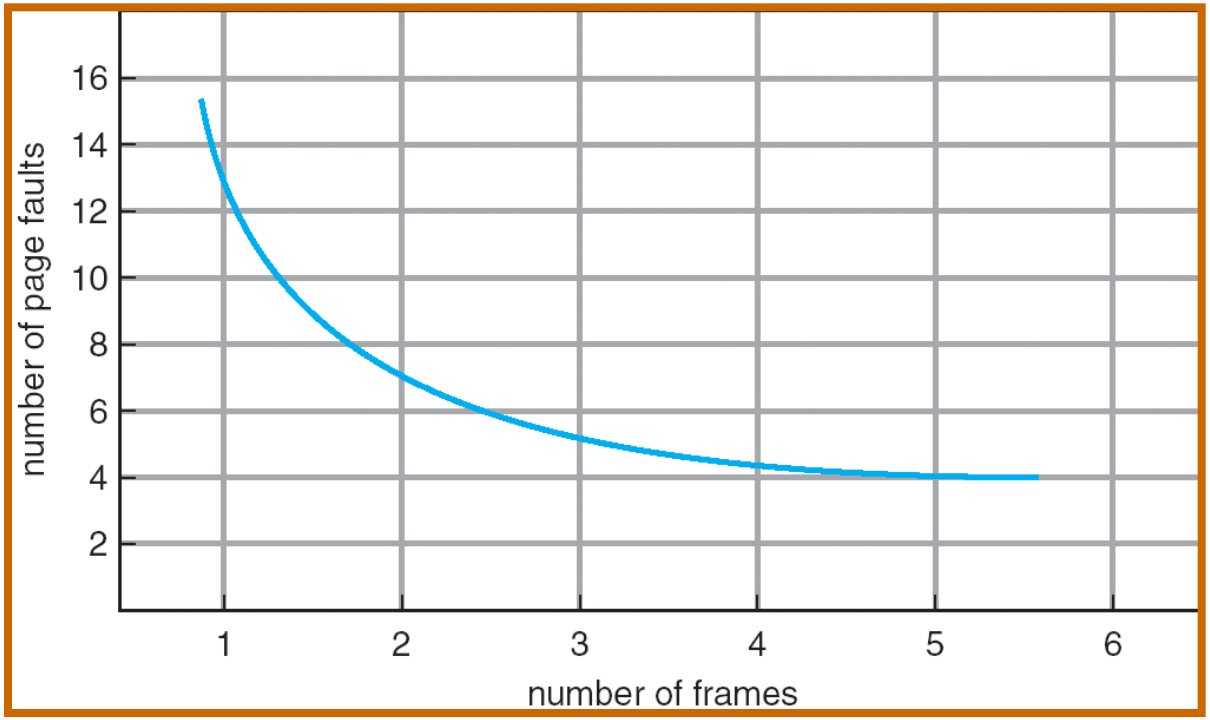
- How would you implement LRU strategy?
- Stack implementation – keep a stack of page numbers in a double link form:
  - Any time Page is referenced move it to the top
  - Replace page from the bottom of the stack
  - No search for replacement

- RANDOM:
  - Pick random page for every replacement
  - Typical solution for TLB's. Simple hardware
  - Pretty unpredictable – makes it hard to make real-time guarantees

Global vs. Local Allocation

- Global replacement – process selects a replacement frame from the set of all frames; one process can take a frame from another.
- Local replacement – each process selects from only its own set of allocated frames.

Graph of Page Faults Versus The Number of Frames



- One desirable property: When you add memory the miss rate goes down
  - Does this always happen?
  - Seems like it should, right?
- No: BeLady's anomaly
  - Certain replacement algorithms (FIFO) don't have this obvious property!

Adding Memory Doesn't Always Help Fault Rate

- Does adding memory reduce number of page faults?
  - Yes for LRU and MIN
  - Not necessarily for FIFO! (Called Belady's anomaly)

Ref: / Page:	A	B	C	D	A	B	E	A	B	C	D	E
1	A			D			E					
2		B			A					C		
3			C			B					D	

Ref: / Page:	A	B	C	D	A	B	E	A	B	C	D	E
1	A						E				D	
2		B						A				E
3			C						B			
4				D						C		

Comparison of Algorithms

	FIFO	OPT/MIN	LRU
Data structure for implementation	Queue	NA	DLL using counter or stack method
Traversal of reference string	Forward	Forward	Backward
Implementation	Easy to understand n imple.	Diff. to impl.	Approximation of OPT ,impl is possible
Performance	Not good	Mainly used for comparison study	Good
Time to be considered	Uses the time when page was brought into memory	Uses the time when a page is to be used	Approximation of OPT , Uses the time when a page is used

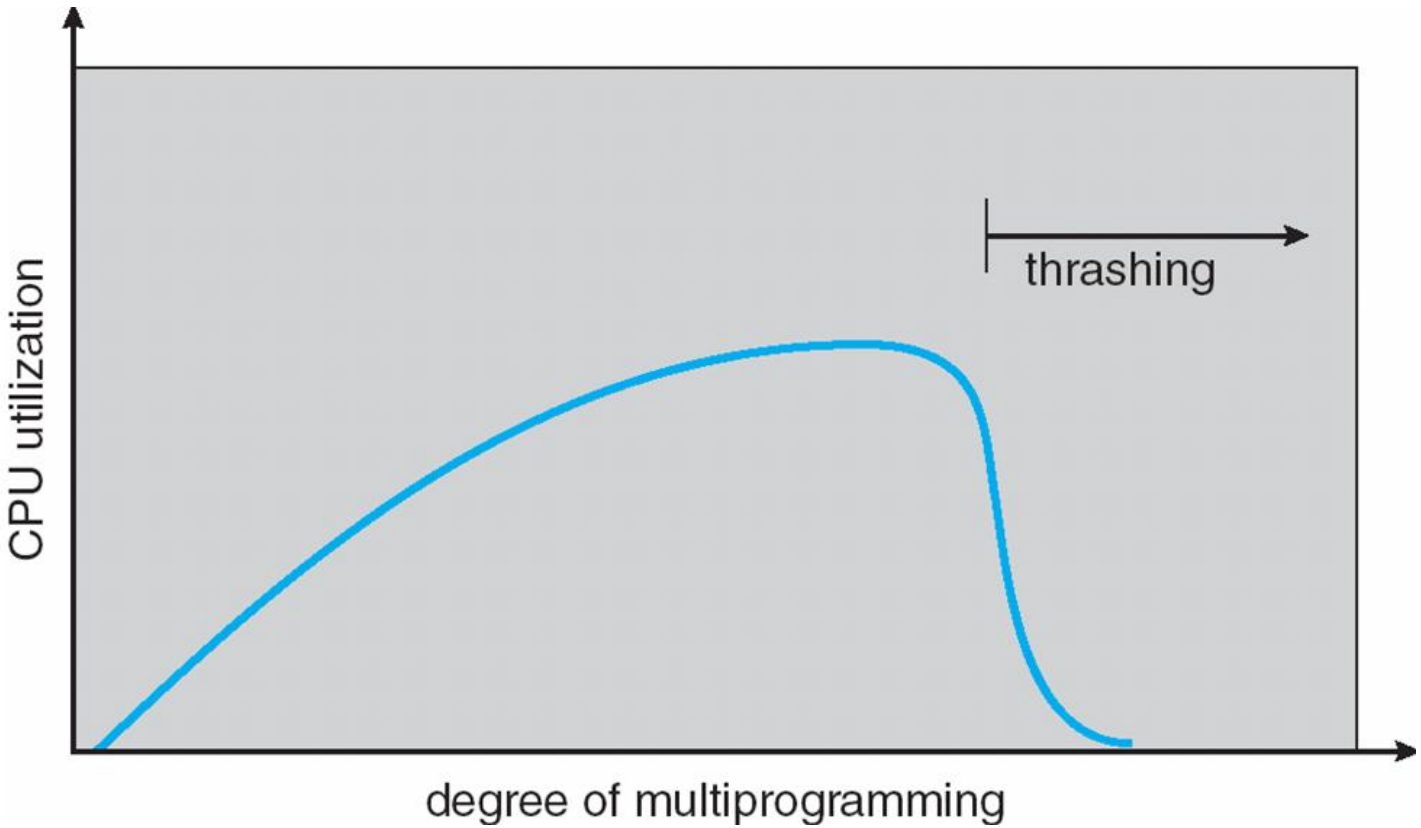
Contents

- Memory Management requirements
- Memory Partitioning: Fixed and Variable Partitioning,
- Allocation Strategies (First Fit, Best Fit, and Worst Fit), Fragmentation, Swapping.
- Virtual Memory: Concepts, Segmentation, Paging, Address Translation,
- Page Replacement Policies (FIFO, LRU, Optimal, Other Strategies),
- Thrashing.

Thrashing

- If a process does not have “enough” pages, the page-fault rate is very high. This leads to:
  - low CPU utilization
  - operating system thinks that it needs to increase the degree of multiprogramming
  - another process added to the system
- **Thrashing** ≡ a process is busy swapping pages in and out
- 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 (Cont.)



- If global page replacement algorithm is used then thrashing occurs
- To avoid this use local page replacement algorithm

End of Chapter