Chapter 8
Main Memory

8.1, 8.2, 8.3, 8.4, 8.5

Chapter 9
Virtual memory
9.1, 9.2, 9.3

https://www.akkadia.org/drepper/cpumemory.pdf

Images from Silberschatz

How does the OS manage memory?

- Allocation
- Swapping
- Hardware support

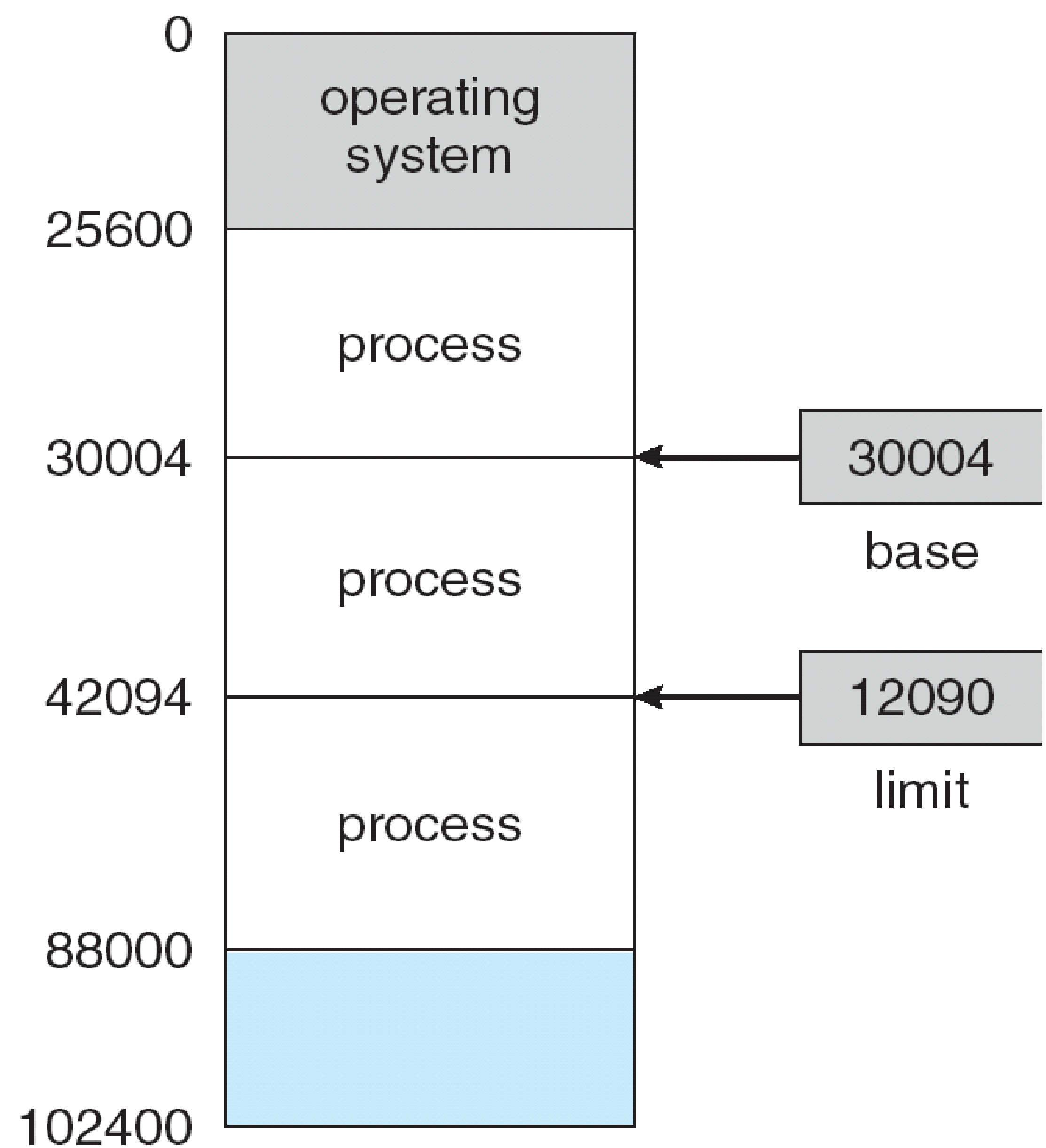
- Assume the entire process must be in memory!
  - Virtual Memory – chapter 9
  - Does not make this assumption

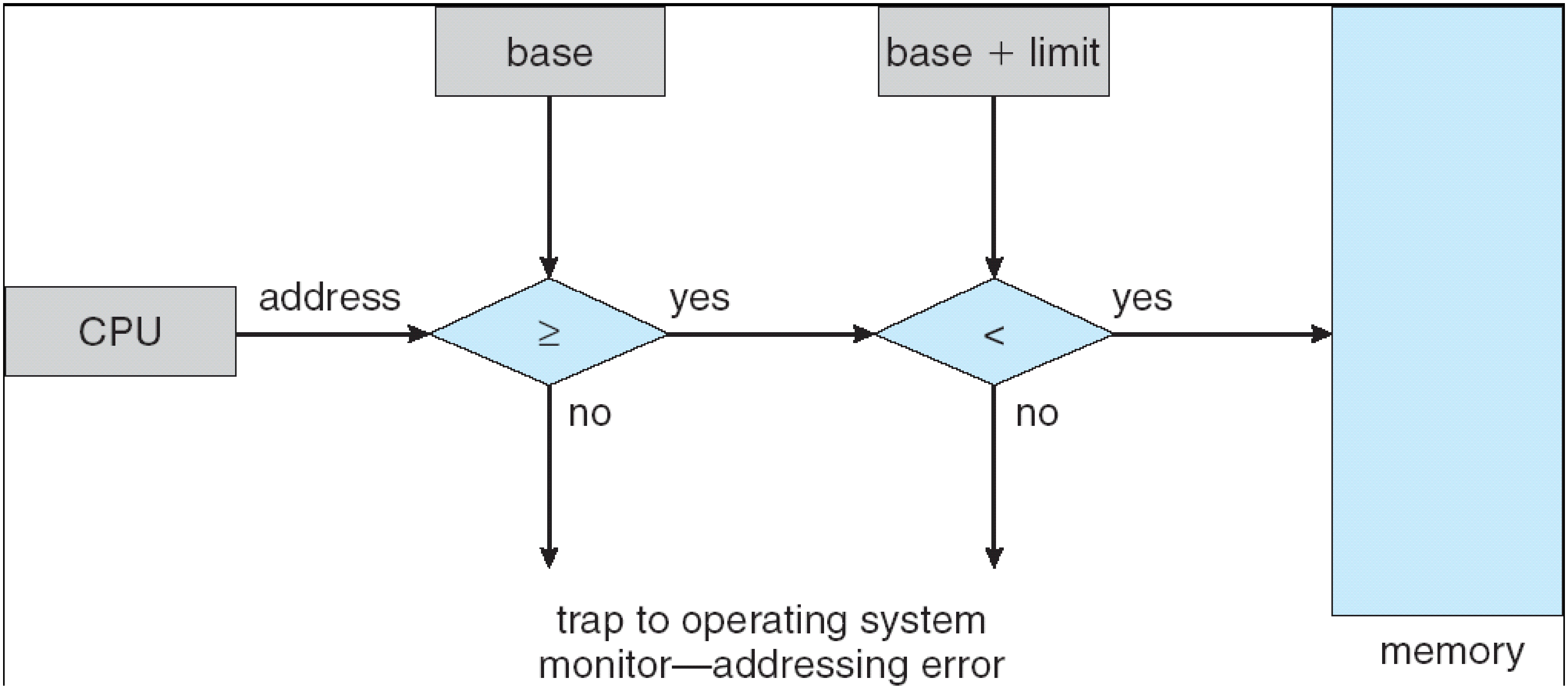
Memory Access Basics

- Register
- Cache
  - Stall
- Main Memory
- Disk
- Protection

(Basic) Mapping + Protection

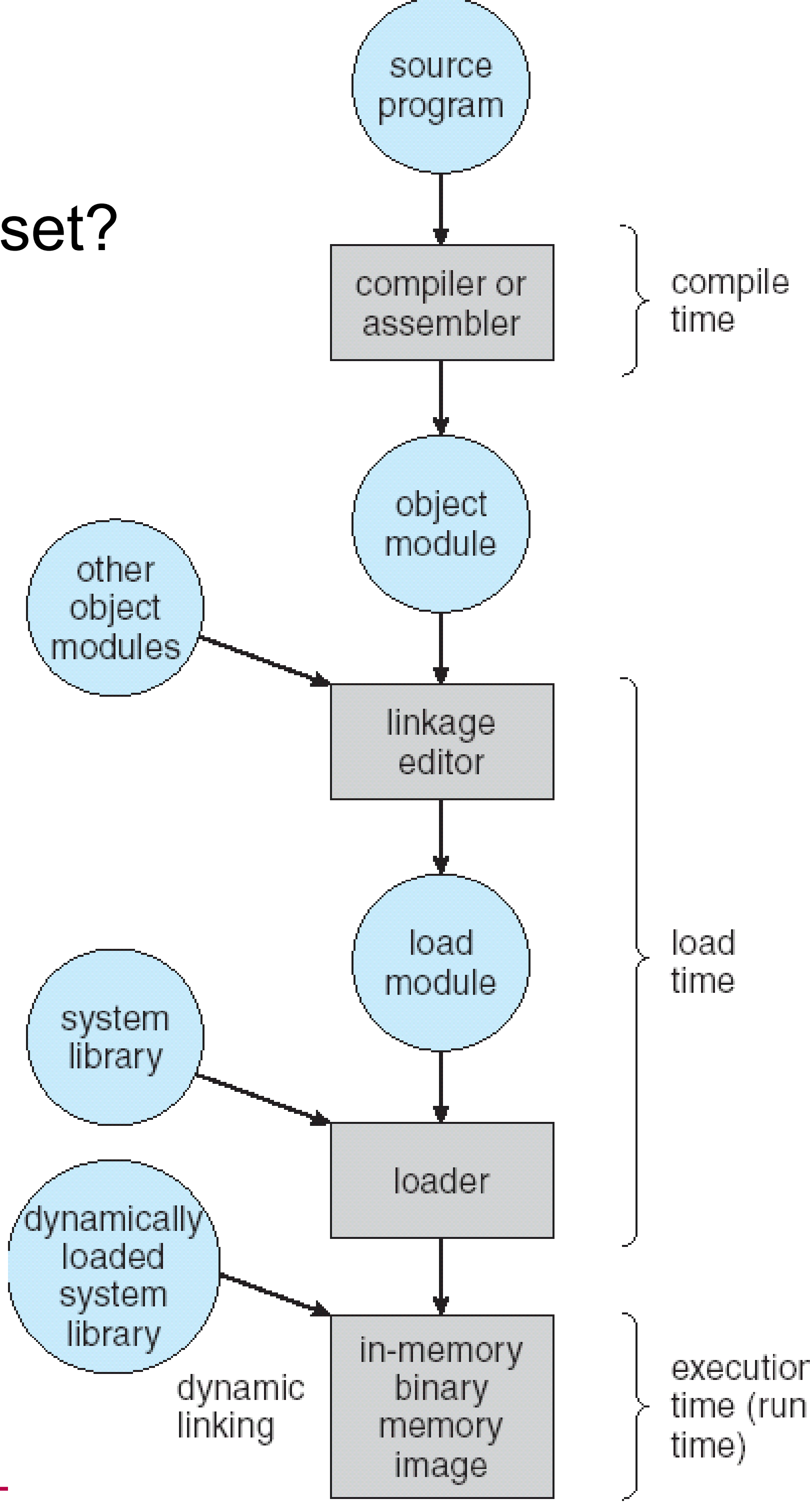
- Software
  - Thinks it can access address zero to limit
- Hardware
  - Two registers
    - Base
    - Limit
  - Privileged instructions
    - Kernel mode!
  - On error
    - Trap!




base
base + limit
CPU
address
≥
yes
<
yes
no
no
trap to operating system
monitor—addressing error
memory

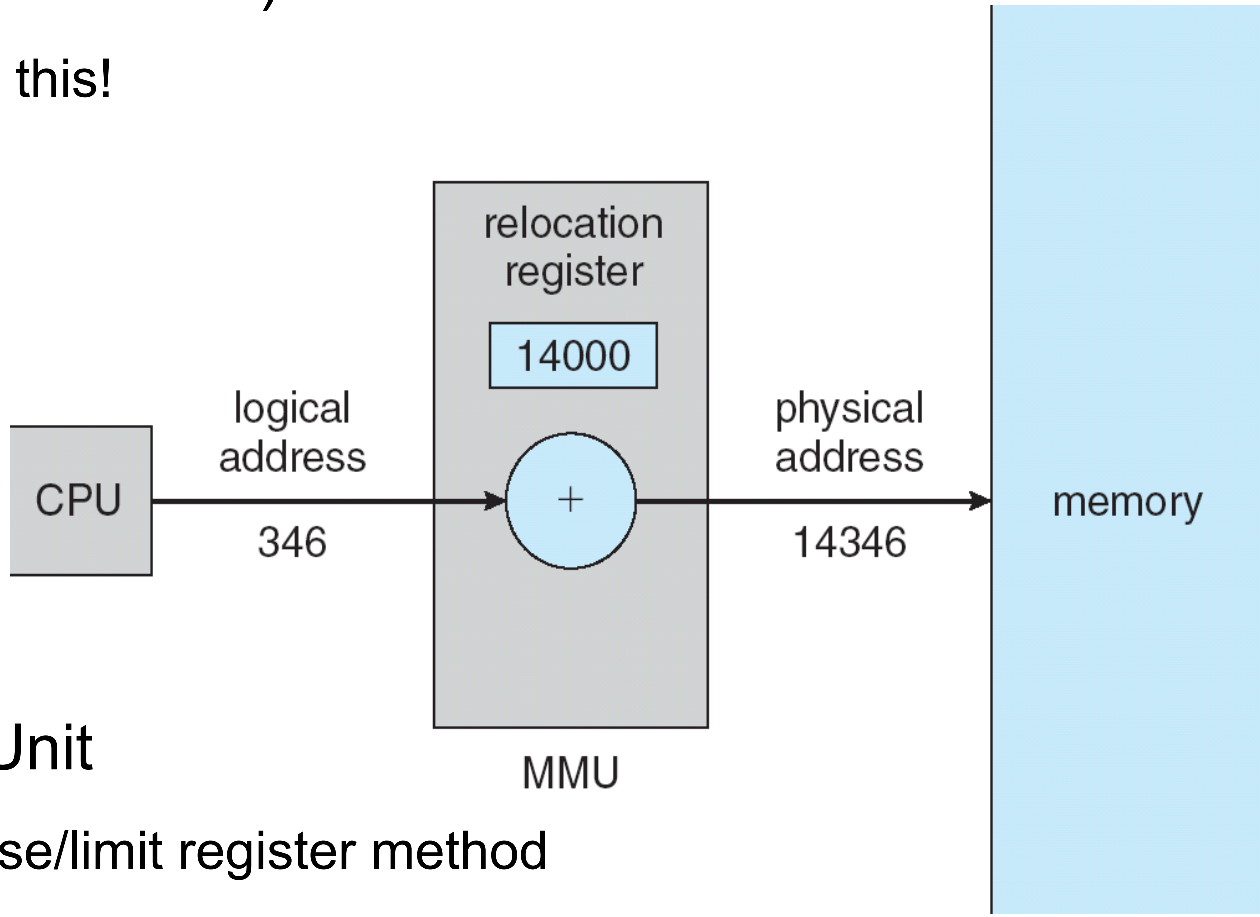
Address Bind Time

- When are addresses in the executable set?
  - Compile time
    - Must always be in the same location
  - Load time
    - Can be loaded anywhere
  - Execution time
    - Can be *moved* during execution!



Logical vs Physical Addresses

- Logical Address (Virtual Address)
  - Software only ever sees this!
- Physical Address



- Memory Management Unit
  - Generalization of the base/limit register method
  - Relocation register

Dynamic Linking

- Linking at execution time
- Static linking
- stub
- Shared libraries
  - .dll or .so

Swapping

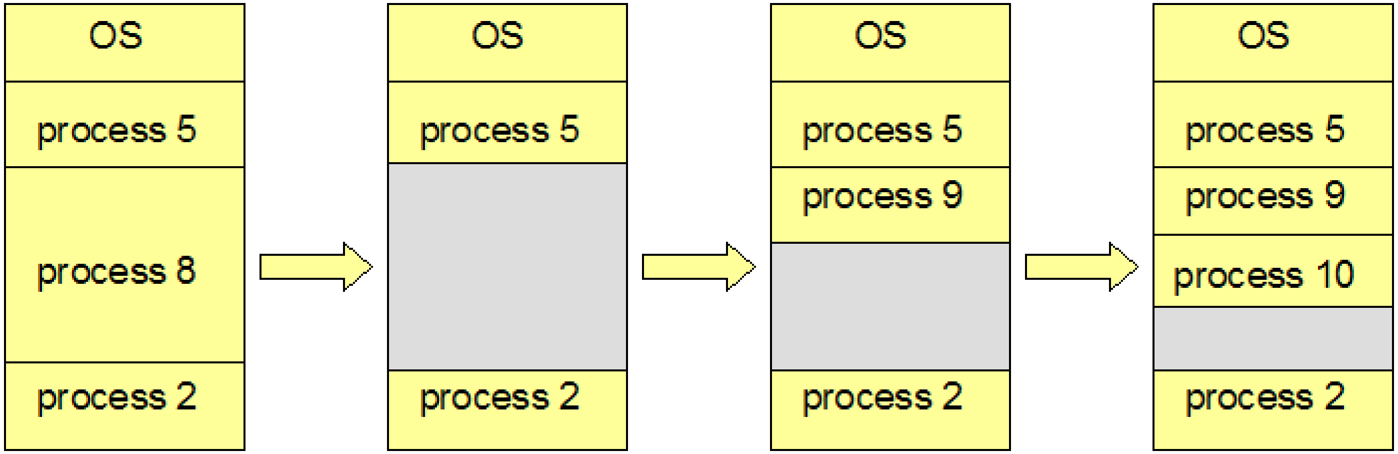
- Not all processes fit in physical memory
  - Chapter 9: not all of a *single process* will fit into physical memory
- Physical memory **<==>** Backing store
- Swap back into memory
  - Same location
  - Different location
- Context Switch Time
  - Size * Transfer rate
  - How does this affect time slices?

Contiguous Memory Allocation

- Two Partitions
  - OS
  - User Processes

Allocation of Memory

- Allocate part of User Space partition to each process
- Hole (technical term)
- First Fit
- Best Fit
- Worst Fit



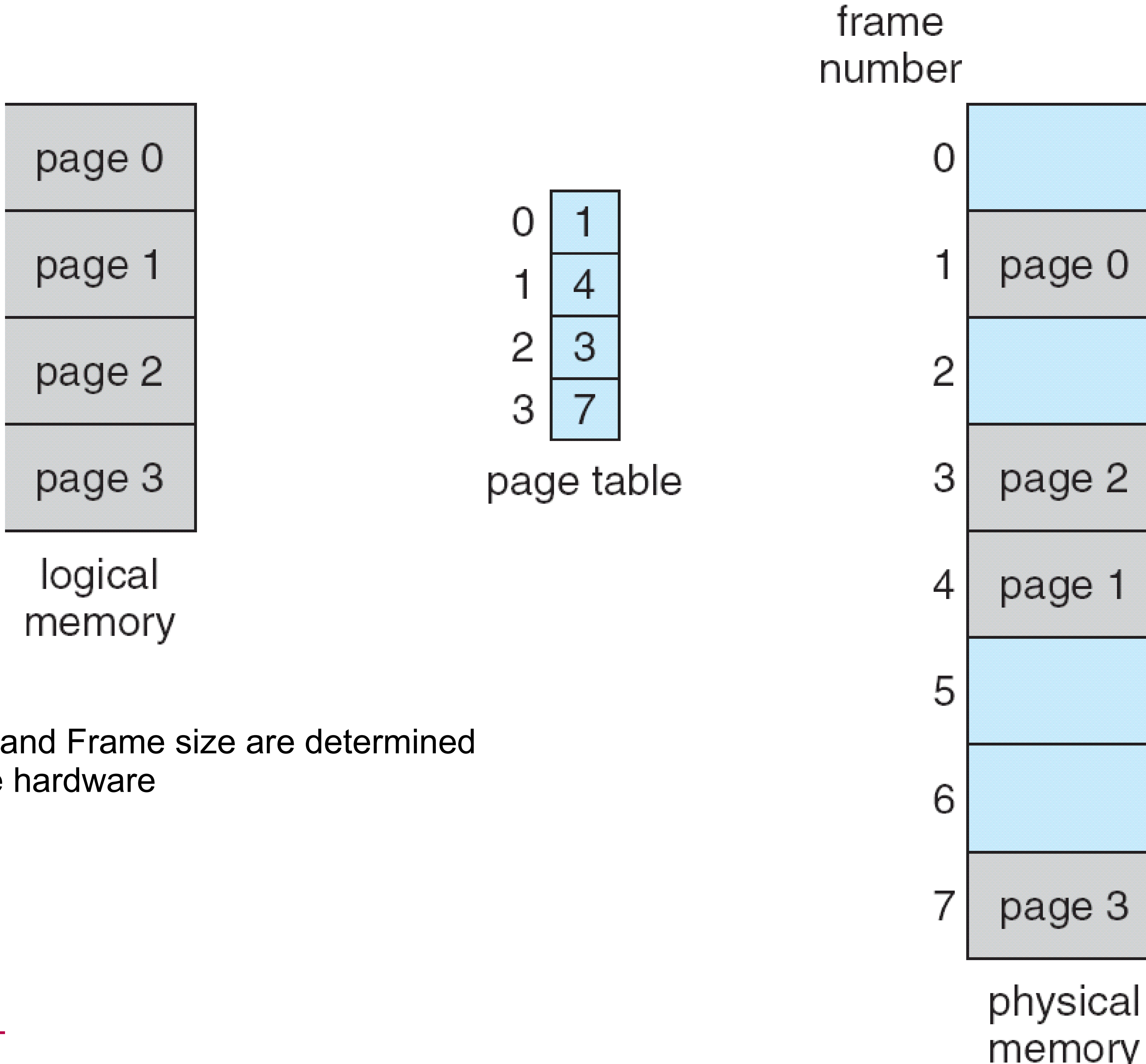
- Best Fit/First Fit found (experimentally) to be better than Worst Fit in terms of time and memory utilization
- What happens if 5 & 2 terminate?

Fragmentation

- External
- Internal
- Compaction
- 50% Rule

Paging!

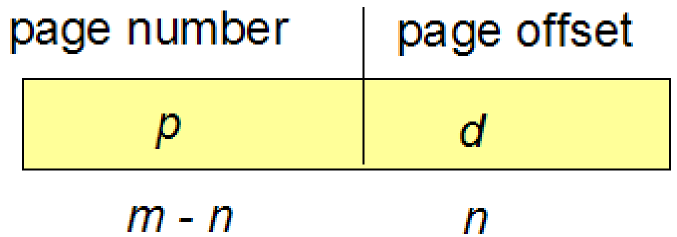
- Noncontiguous memory allocation
- Frame
  - Physical memory
- Page
  - Logical memory
  - Allocate an entire page at a time
- Page table
- Internal Fragmentation



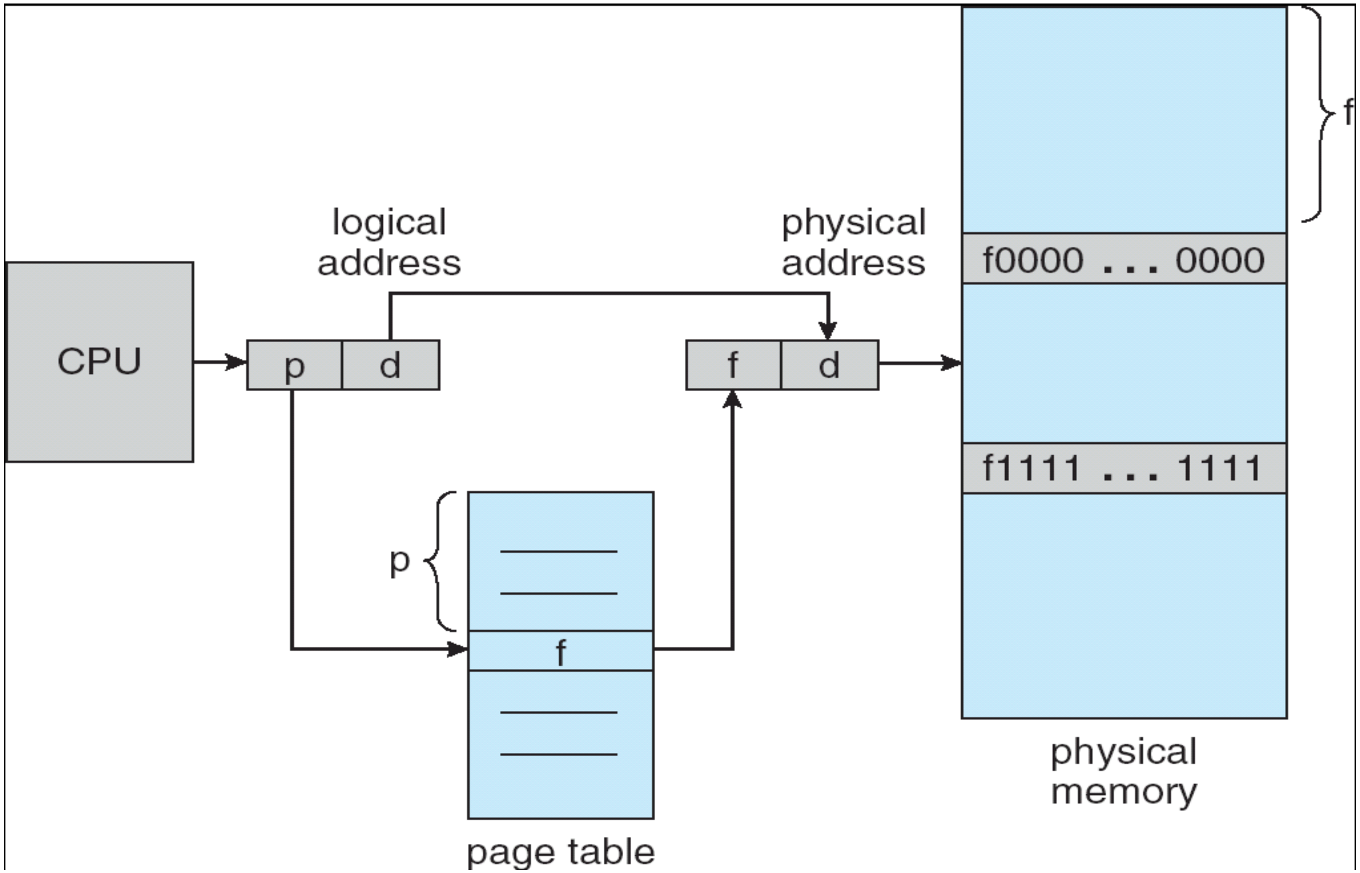
Page and Frame size are determined by the hardware

Address Translation

- Logical Address to Page Number + Offset

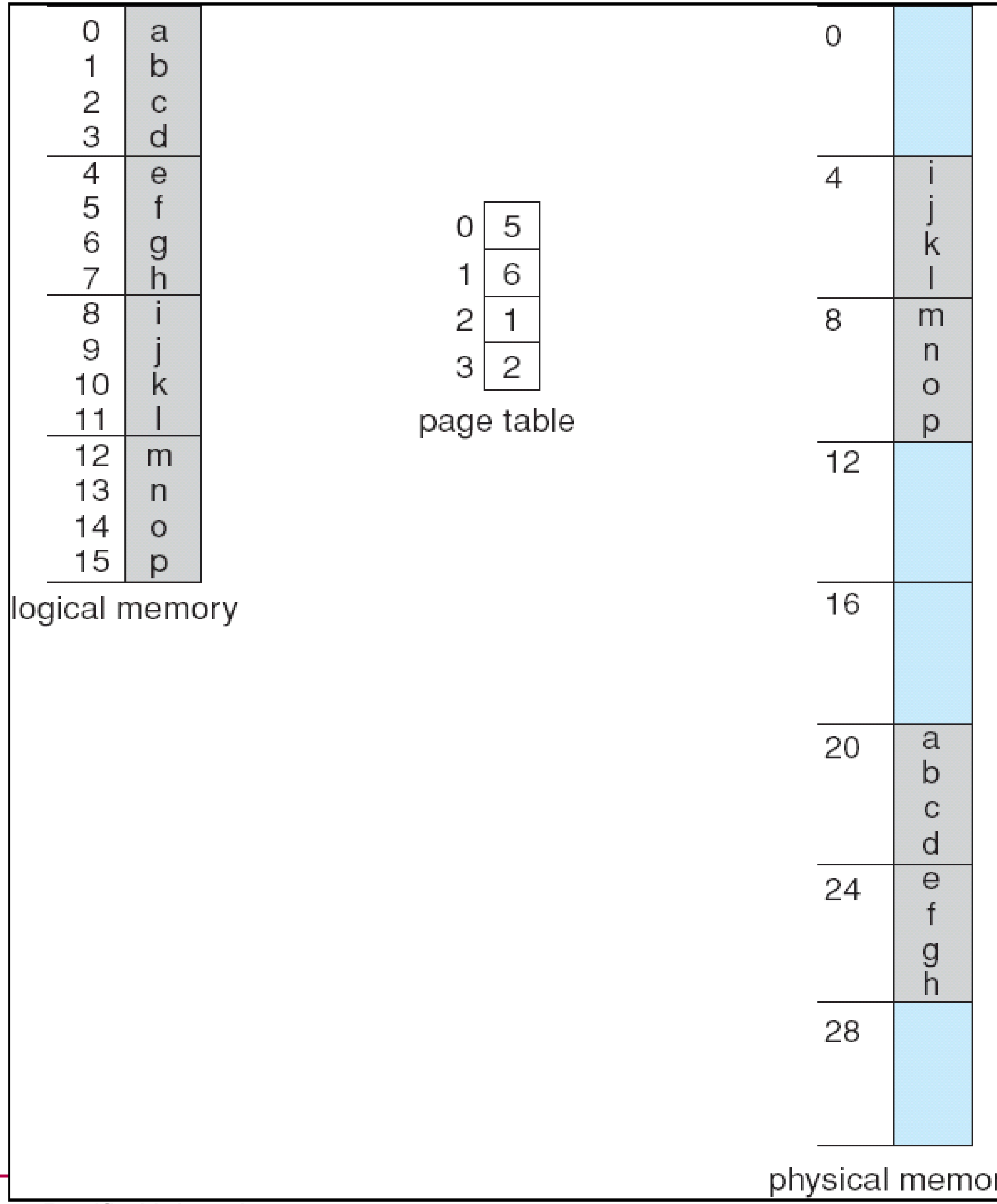


  - Logical address space 2^m and page size 2^n



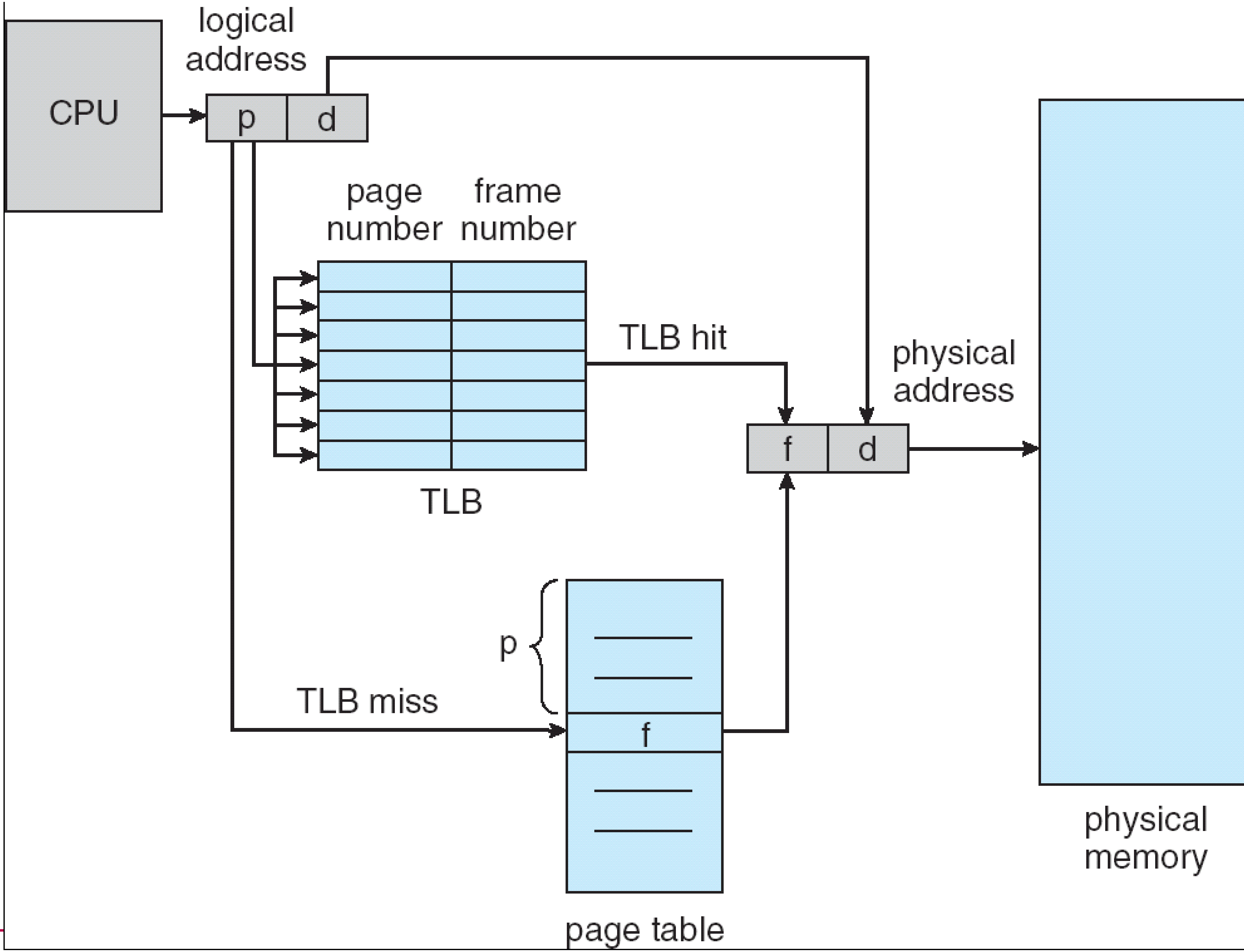
- 32 byte memory
- 4 byte pages
- No guarantee of ordering
- What happens

```
char *pChar = 0x7;
pChar ++;
print pChar;
```



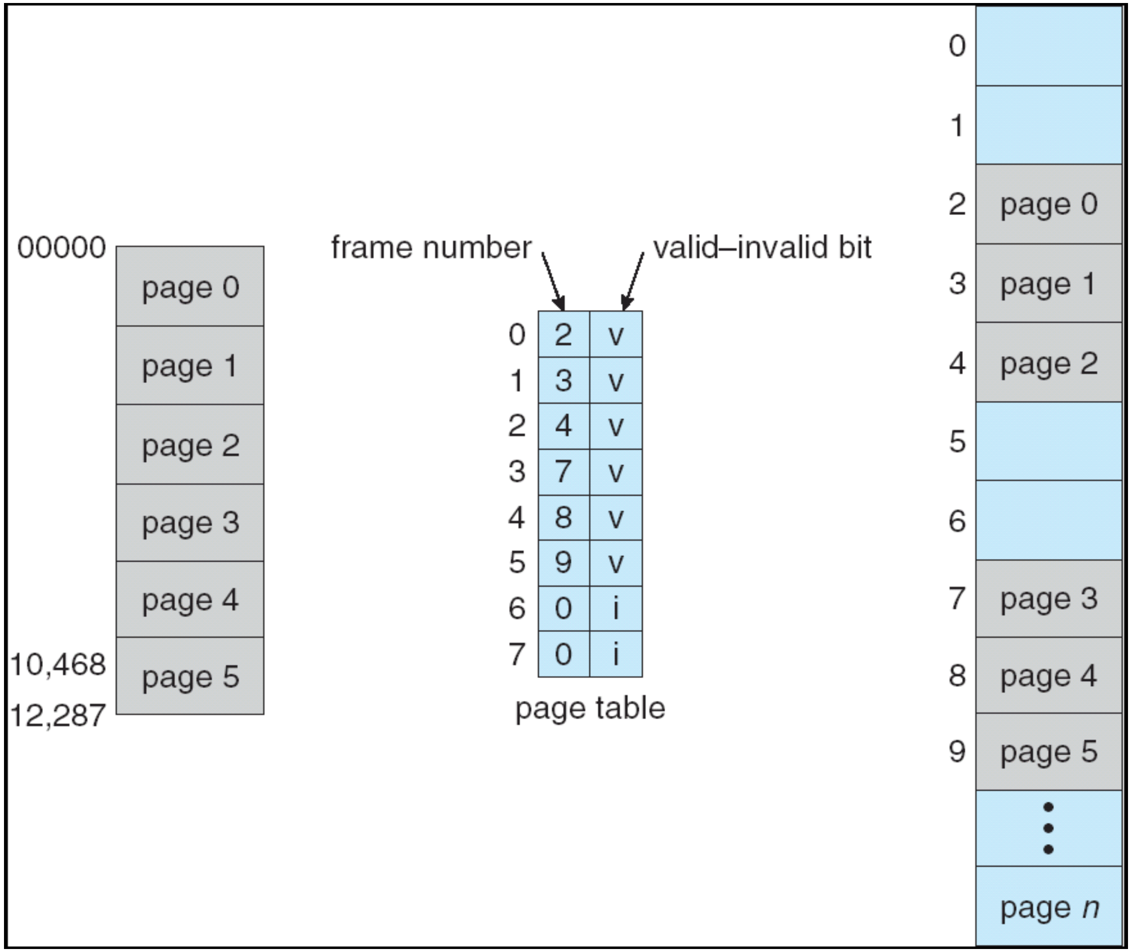
Page Table

- Pages are not always reloaded to the same frame
  - ??
- Contains base address of each page in physical memory
  - Per process (usually)
  - Which frame is it in
  - In main memory
- Hardware (not per process)
  - Page table base register (PTBR)
  - Page table length register (PRLR)
  - Translation look-aside buffers (TLBs)
    - Address space identifiers (ASIDs)
    - protection


logical address
CPU
p
d
page number
frame number
TLB hit
physical address
f
d
TLB
p
TLB miss
f
physical memory
page table

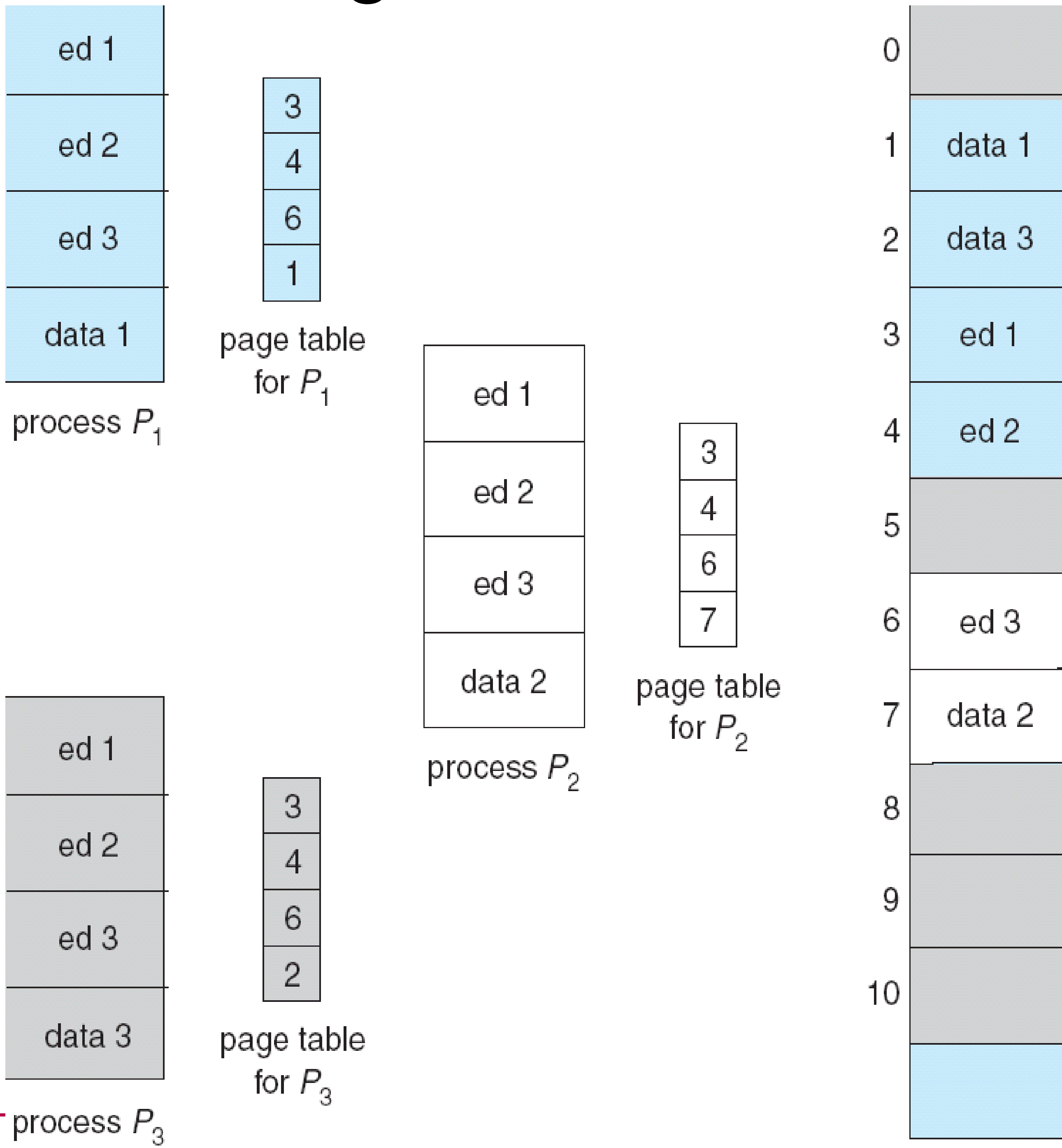
Protection

- Add valid/invalid bit to each page table entry
- ASIDs in TLBs denote which process owns each frame



Shared Pages

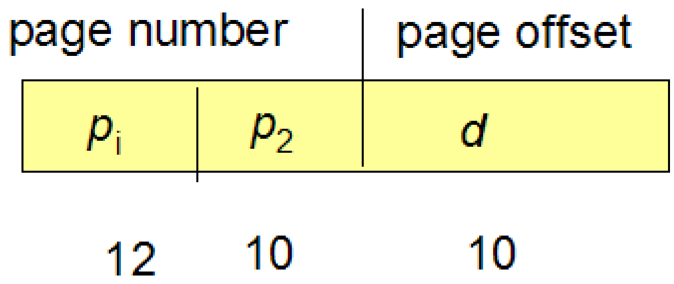
- .dll / .so
  - Share read only code pages
- Shm
  - Shared read/write data pages



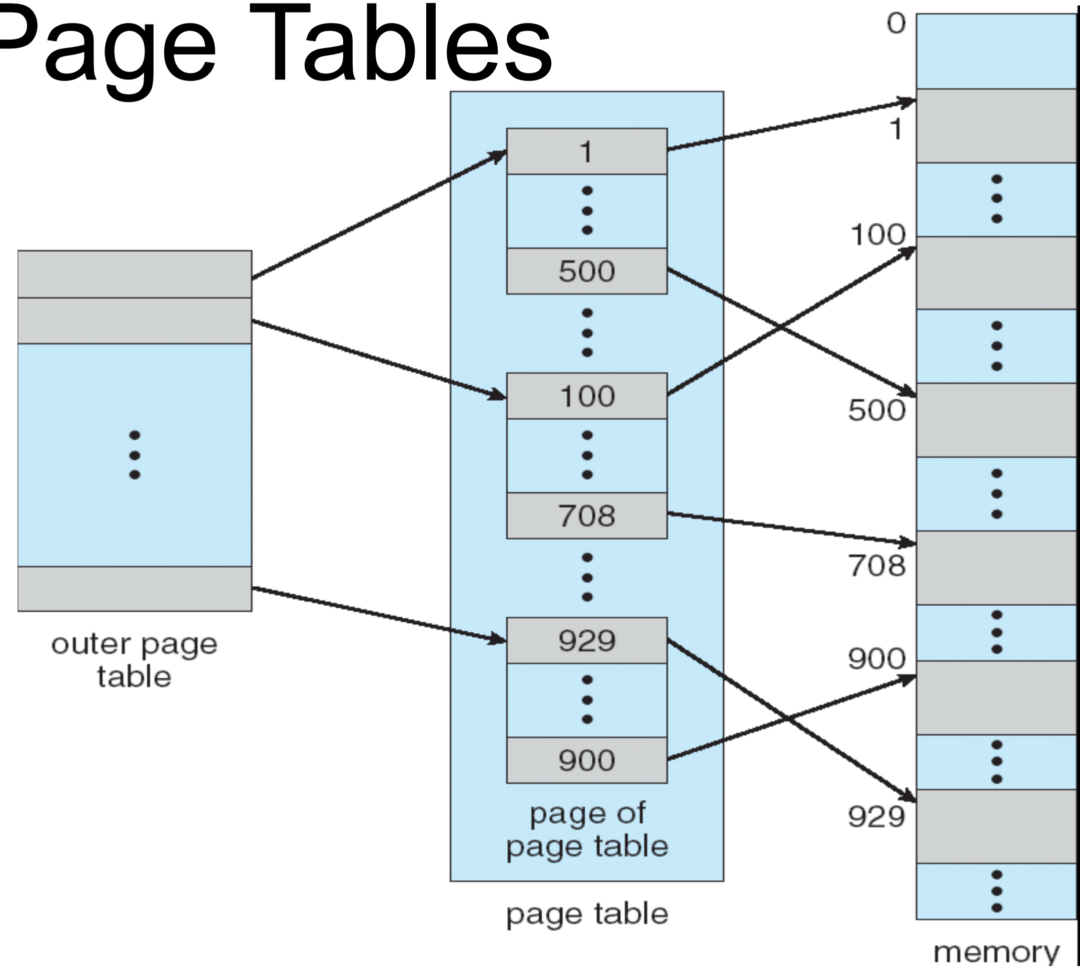
Problems with page tables

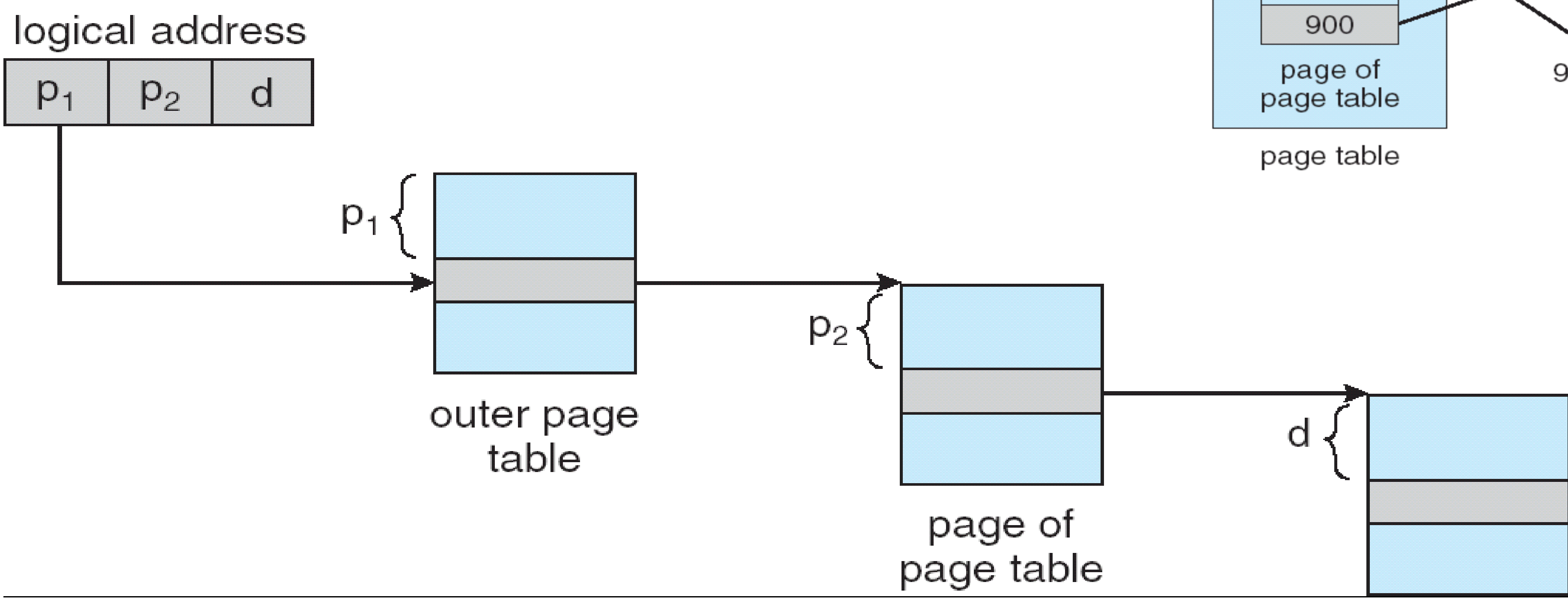
- What do you think?

Multilevel Page Tables



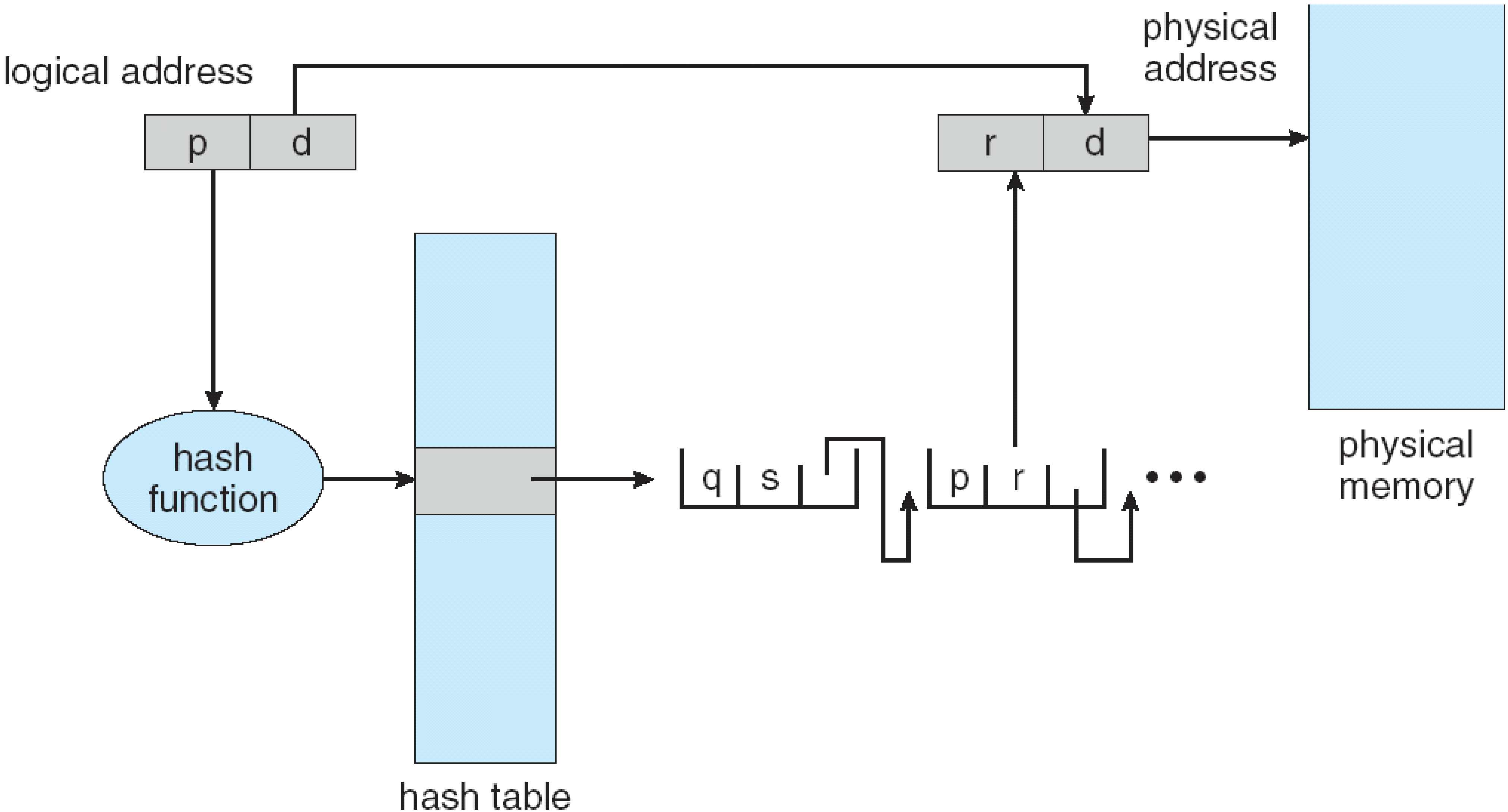
- Page the page table
- Forward mapped page table





Hashed Page Tables

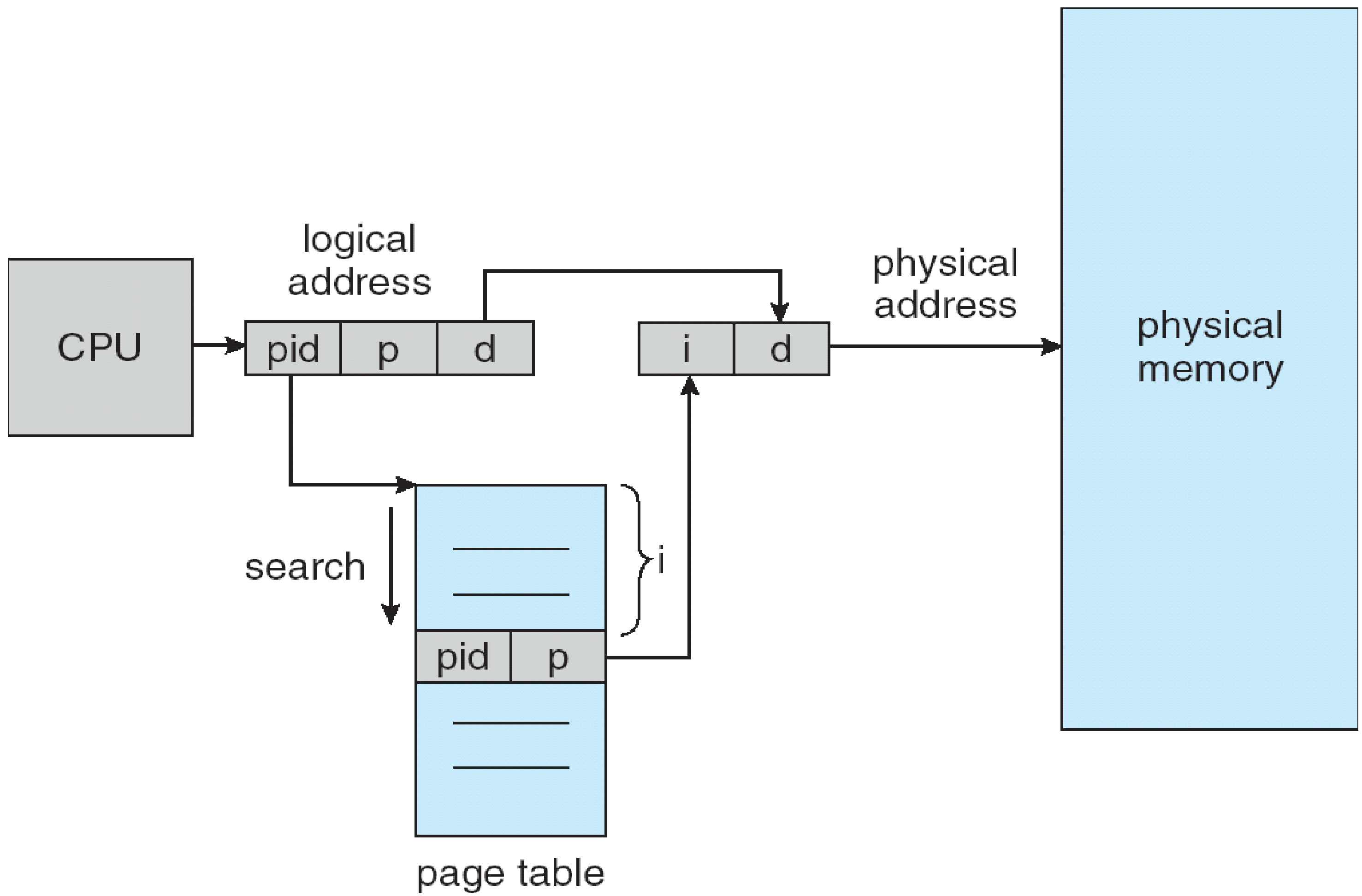
- Address spaced > 32 bits
- Use Virtual address to hash into the table



Inverted Page Table

- One entry per ***frame*** in physical memory
- One page table for the entire system
- Track pid in the table
- Problem?

- Solution?

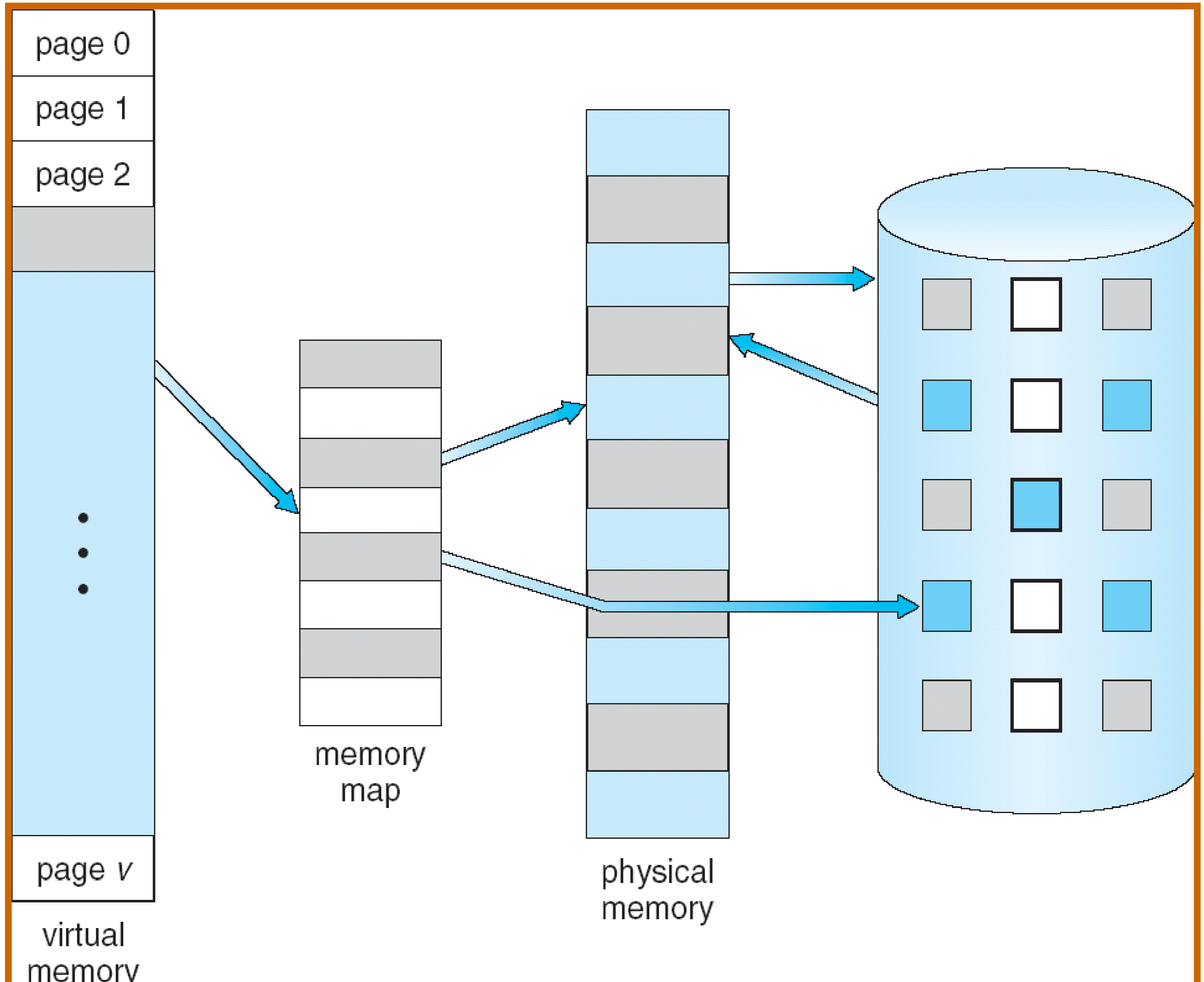


Chapter 9
Virtual Memory

Images from Silberschatz

Virtual Memory

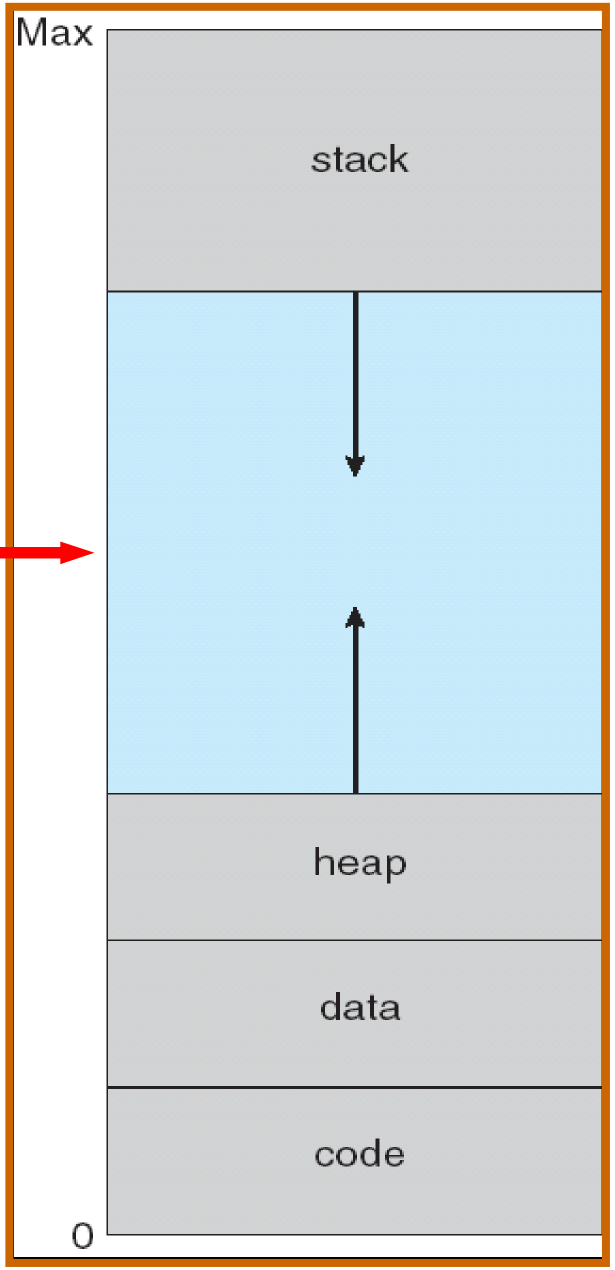
- Processes do not need to be completely in memory to execute
  - data
  - code
  - data set can be larger than physical memory
- Demand Paging



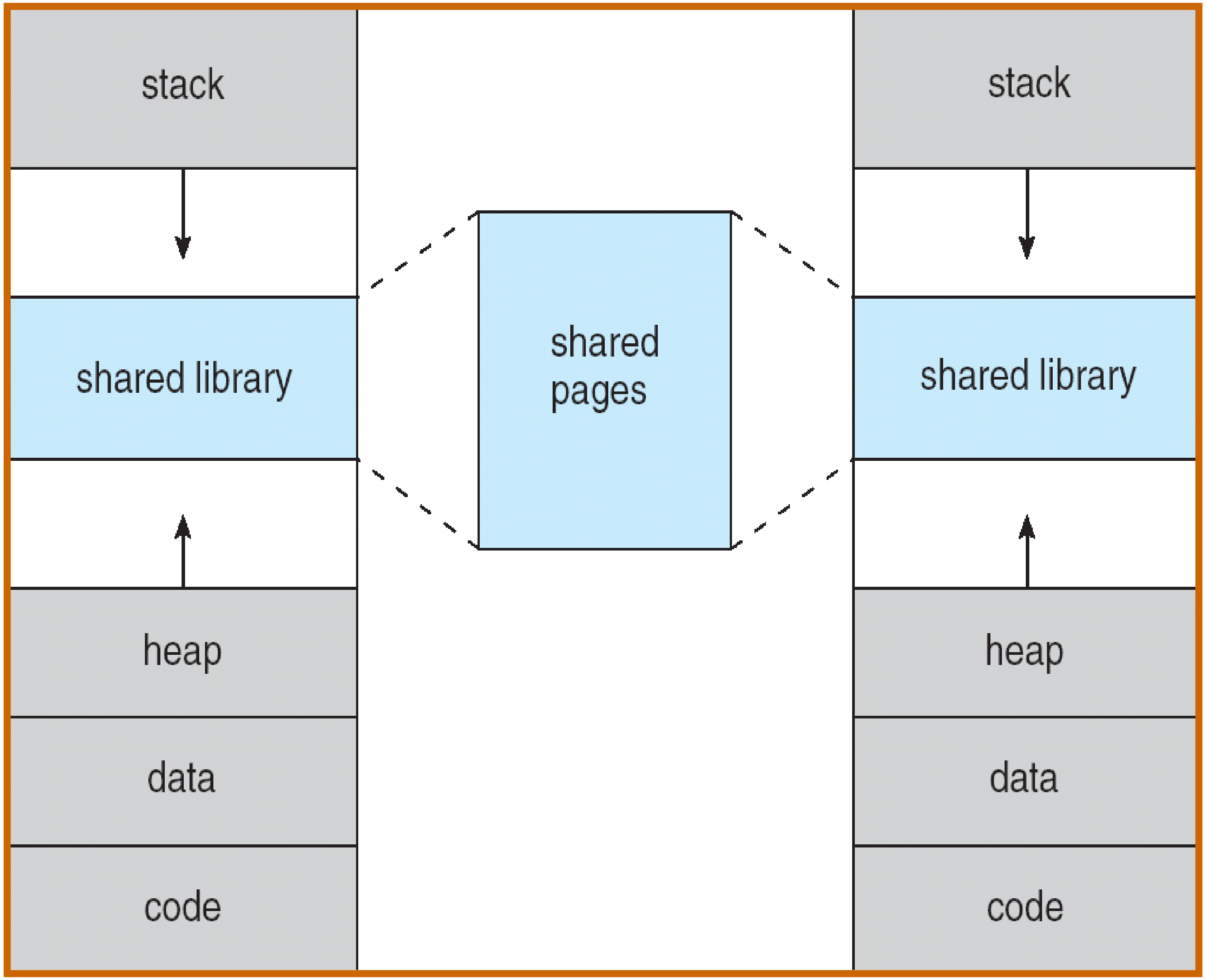
Process View

- Big Virtual Memory space
- Only allocated needed pages
- Empty pages are ignored

Empty Until Needed



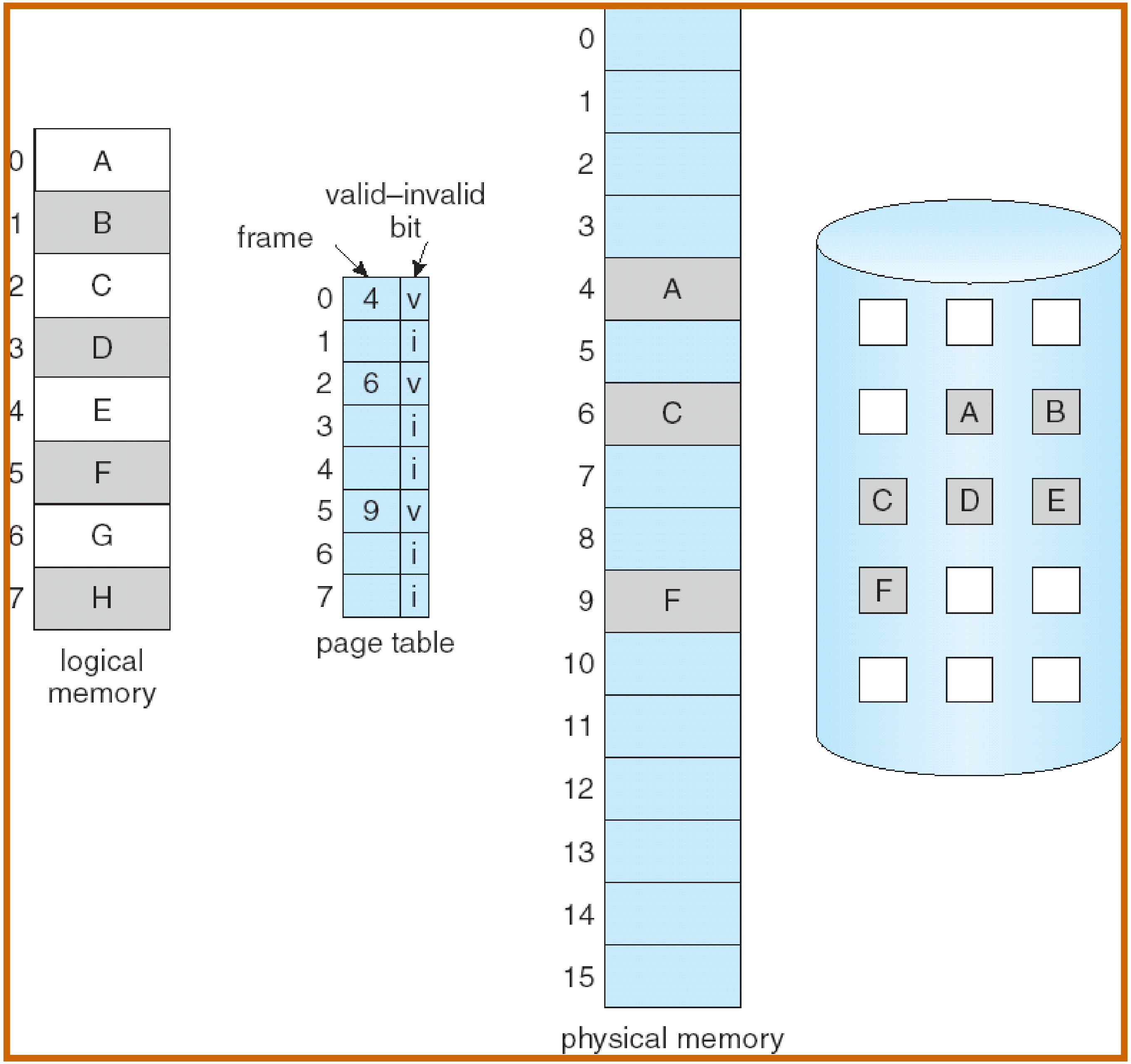
Sharing Memory



Demand Paging

- Load pages as they are needed
  - lazy swapping (pager)
  - less I/O (up front)
  - less memory used at once
  - faster response
  - more processes fit into memory
  - mark pages as in memory or not (similar to valid/invalid)

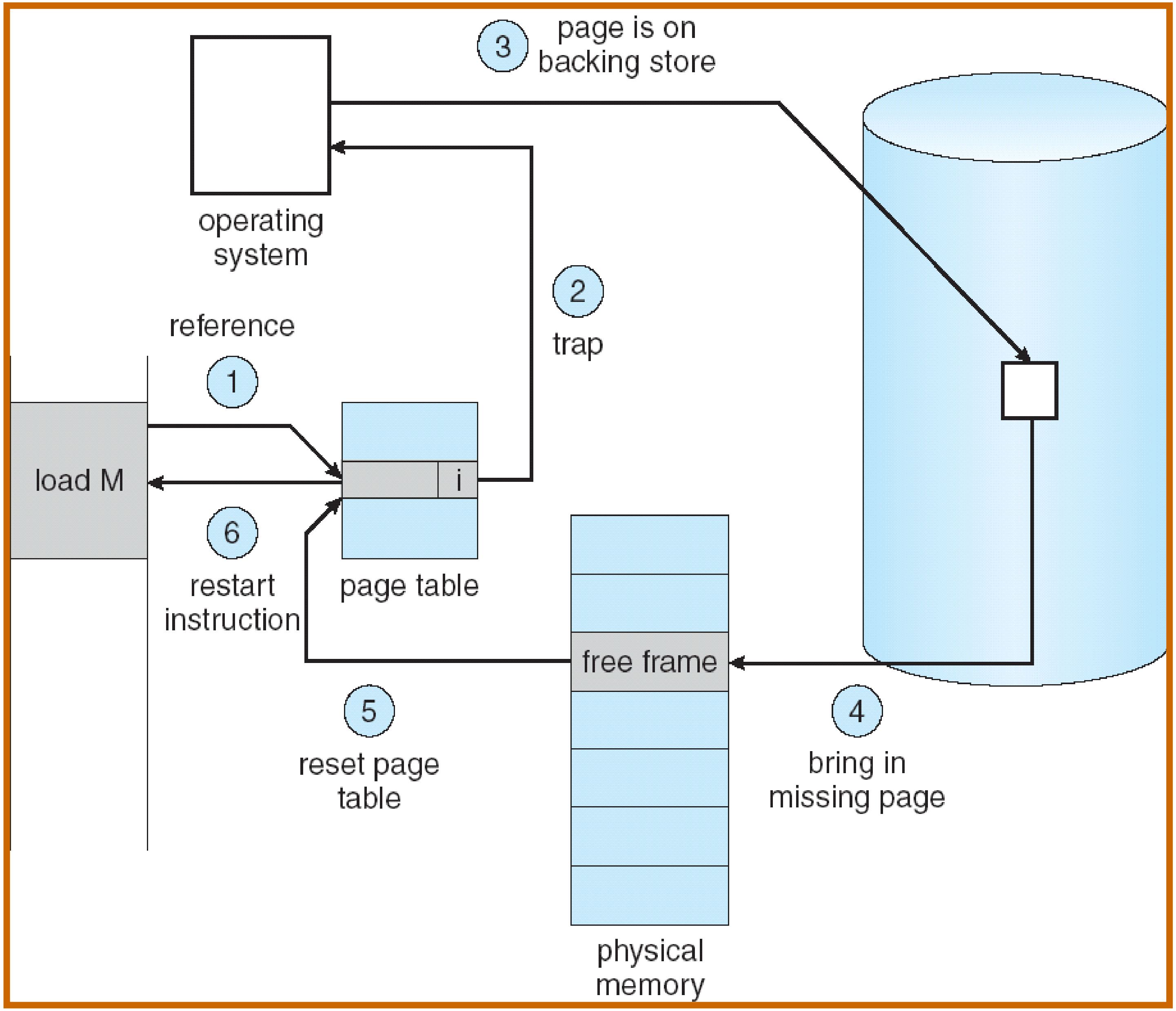
New Page Table



Hardware Support

- Accessing an out-of-memory page causes a page fault trap
- OS handles this and brings the page into memory
- Also must check for invalid address
- Pure Demand Paging
  - Locality of reference
- Page fault may occur anywhere in an instruction
  - may backup and rerun something

Page Fault!



3
page is on
backing store
operating
system
2
trap
reference
1
load M
i
6
restart
instruction
page table
free frame
5
reset page
table
4
bring in
missing page
physical
memory

Copy-on-Write

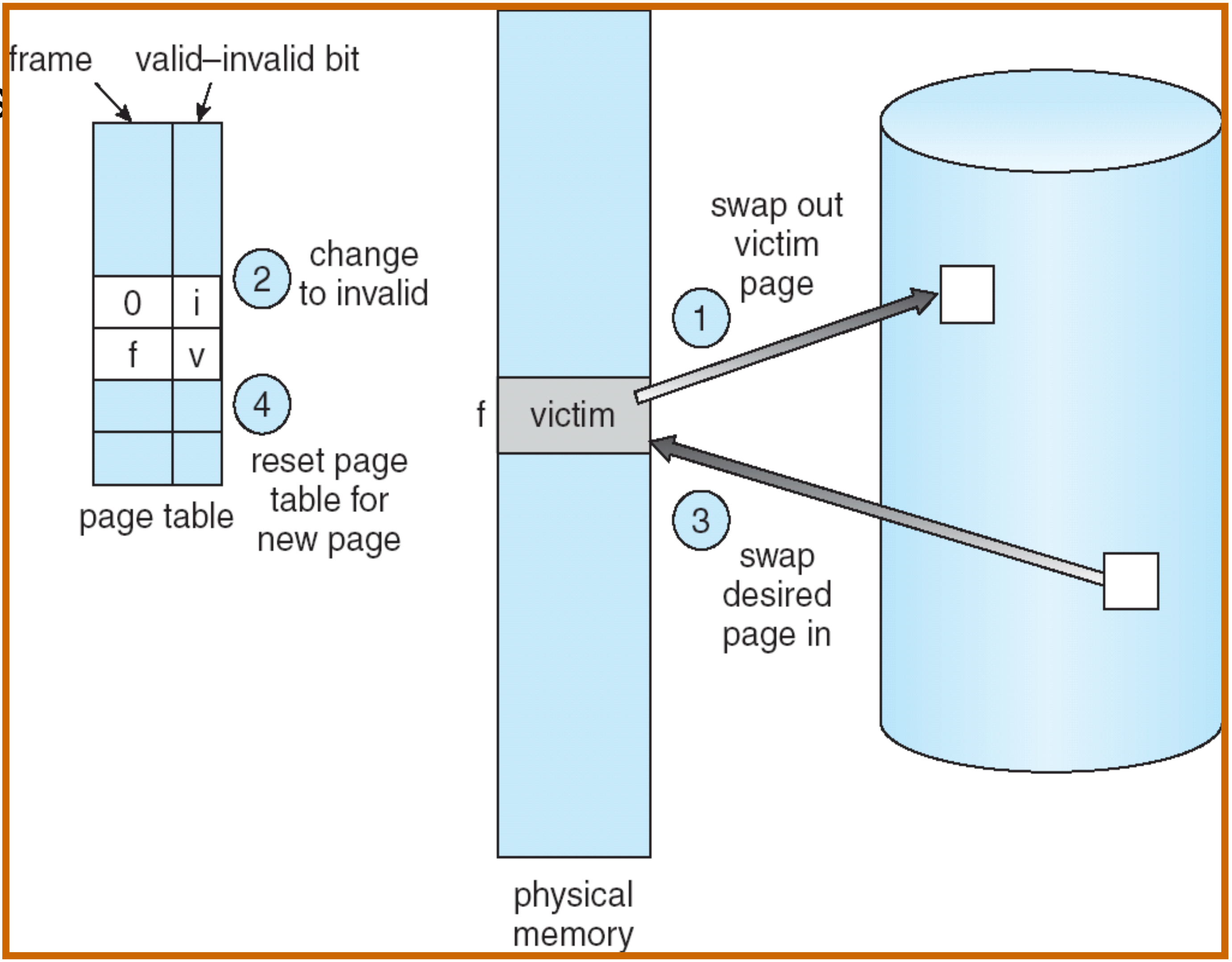
- When do processes share pages?

- Only copy (create a new page) when one process writes to a shared page
  - faster

- vfork()/exec()

Page Replacement

- Remove page from physical memory to make room
  - swap out a process/frame
- Two I/O operations
  - out then in
  - time consuming
  - page may still be on disk
  - dirty bit!



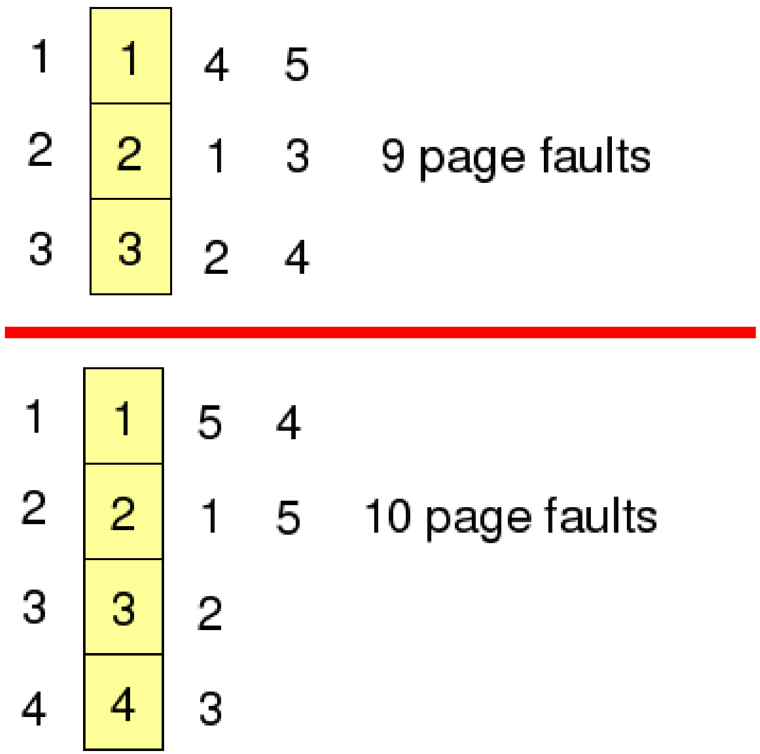
Algorithms

- Goal: Few page faults
- Frame Allocation

- Page Replacement

FIFO

- First In, First Out
- Ref String: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



- Belady's Anomaly:
  - more frames, more faults

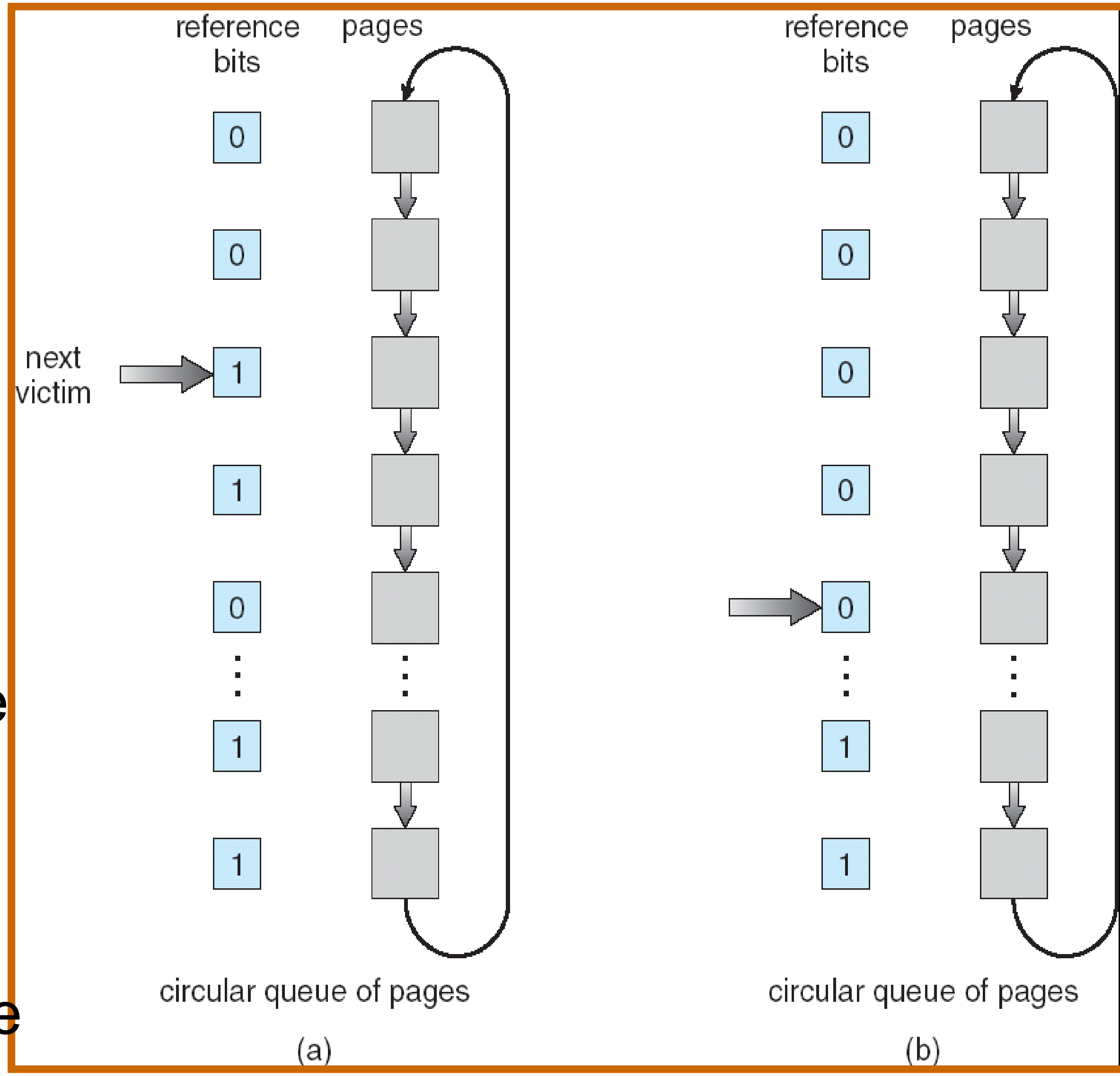
Optimal Replacement Algo

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

- Problems with this?

Approximate Optimal

- LRU
- LRU-Approximate
  - reference bit
  - may be also FIFO (second chance)
- LRU-Additional-Reference-Bits
  - many (8?) bits
- Enhanced Second Chance
  - referenced, modified bits



Counting Algorithms

- Count references per page
  - rarely used in real world
- Least Frequently Used
- Most Frequently Used

Global vs Local

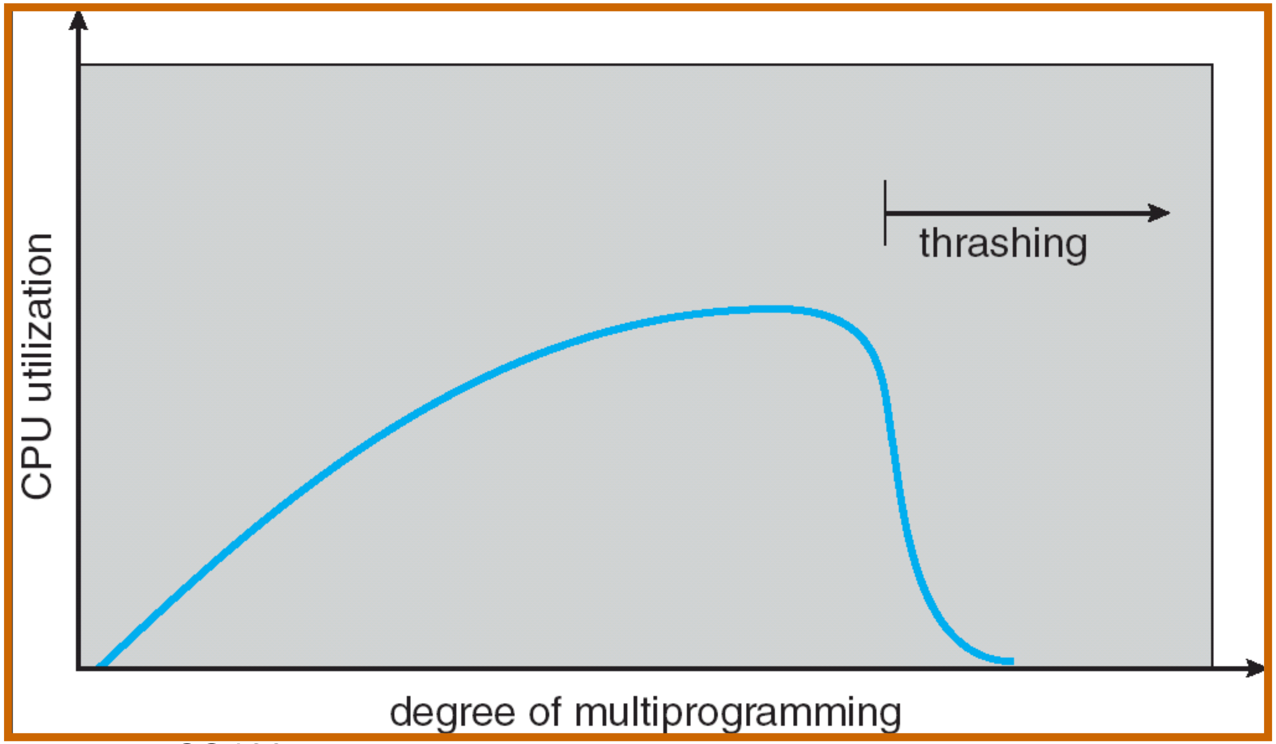
- Global replacement

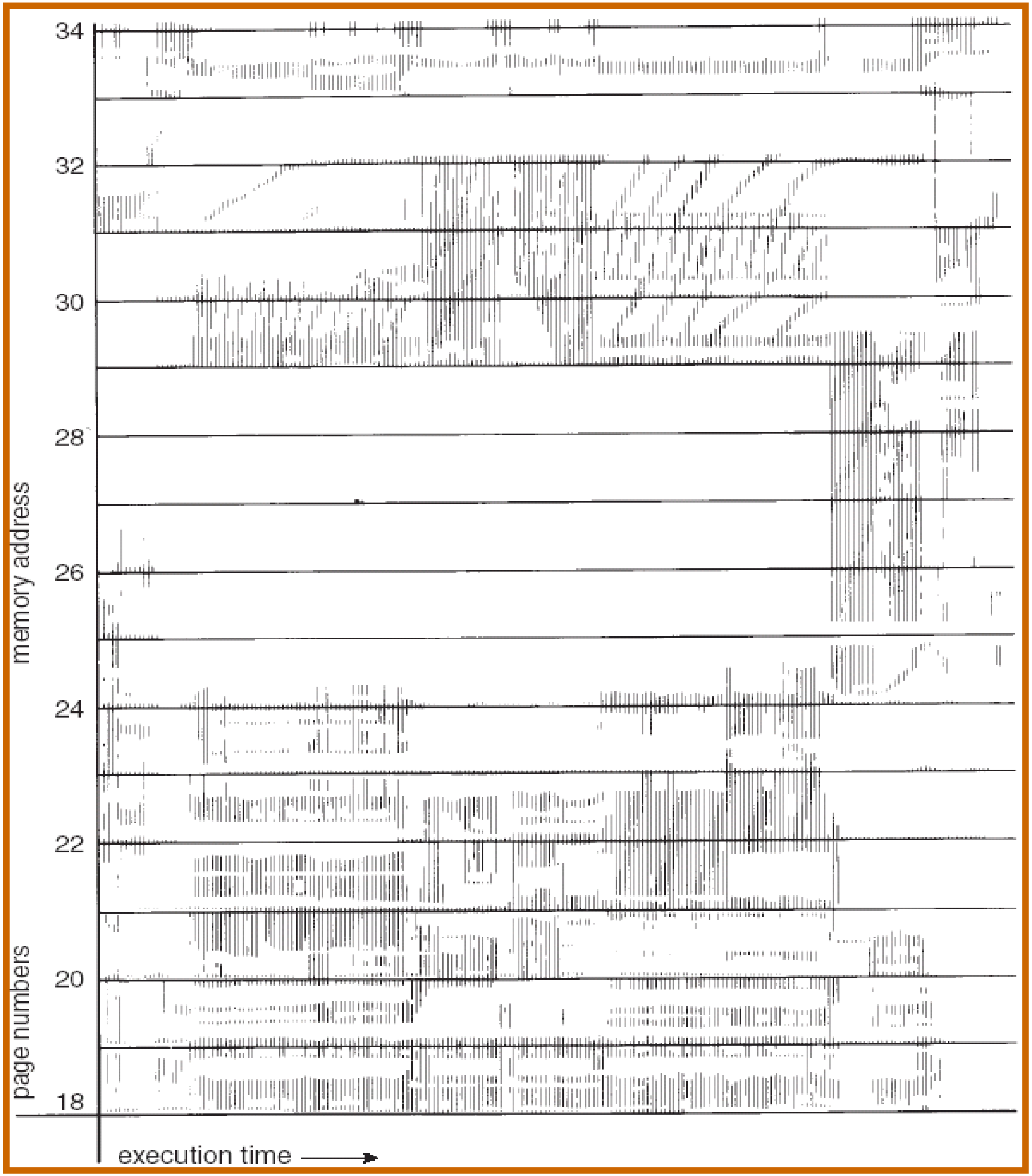
- Local replacement

Thrashing

- Furiously swapping pages in and out
- Problems?

- CPU utilization is low, so OS adds more processes
  - more frames are used
- Poor data layout in your application




34
32
30
28
26
24
22
20
18
memory address
page numbers
execution time