Lecture 7
Main Memory

I233E OPERATING SYSTEMS

RAZVAN BEURAN

Today's Topics

Main memory

- What it is needed for
- How it works

Paging

- Mechanisms
- Hardware support

Segmentation

- Mechanisms
- Protection & sharing

Address Binding

Code generation

- Address of function
- Address of variables

Link

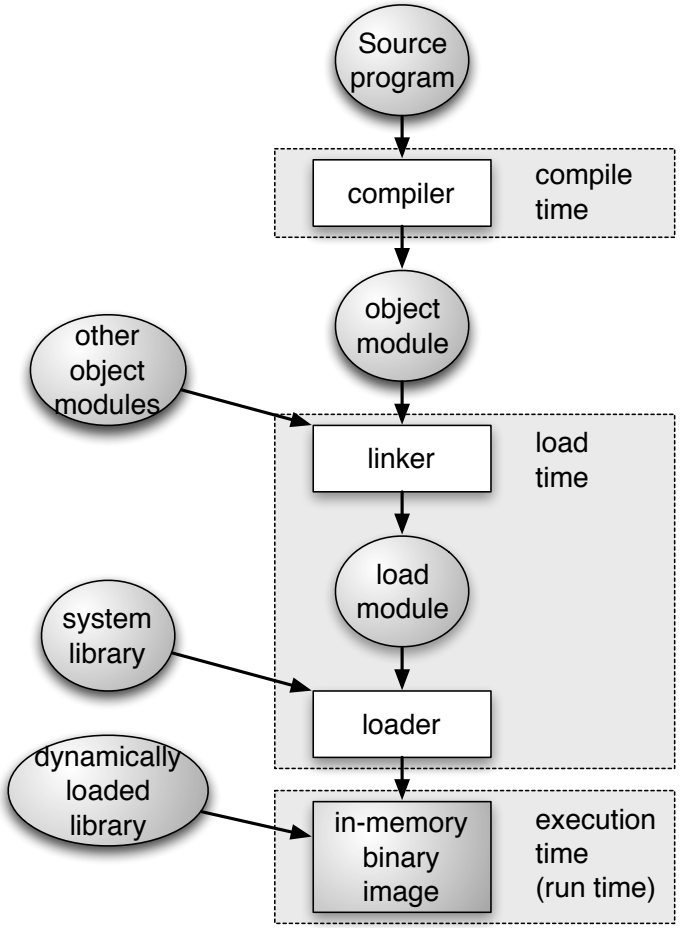
- External symbols

Load

- Decide starting position
- Handle relocatable code

Execution

- Run process



Runtime

Dynamic loading

- Routine loaded at runtime
- Loaded only when called

Dynamic linking

- A “stub” introduced into process image
- Actual code loaded at runtime
- Code shared between processes

Swapping

Mechanism

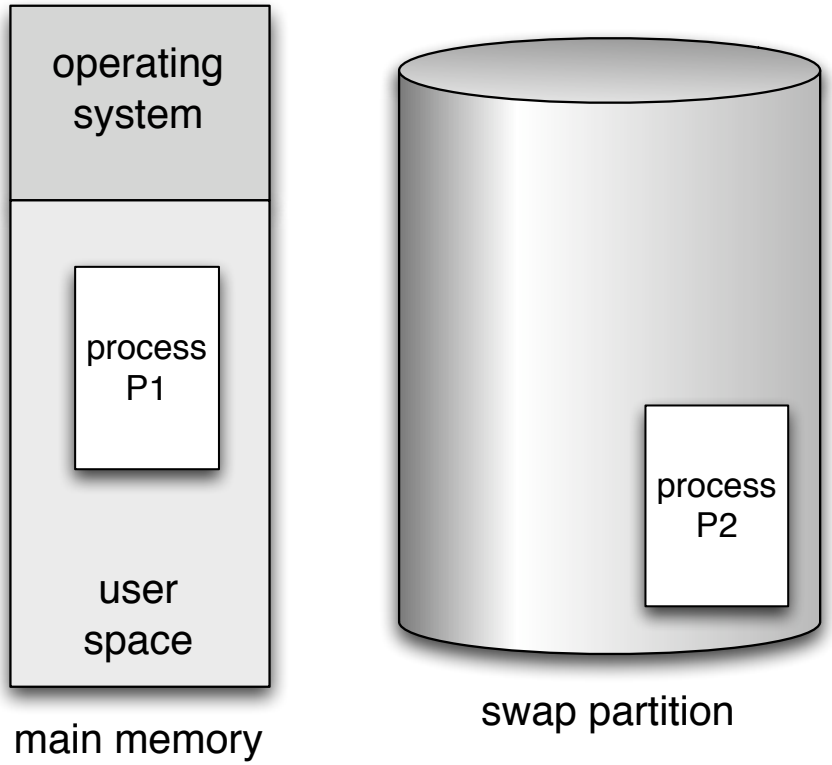
- Memory is limited
- Use disk partition

Swap out

- Unused process
- Move from memory → disk

Swap in

- Used process
- Move from disk → memory



Memory Allocation

Situation

- New process is coming
- Where to place it in memory?

First Fit

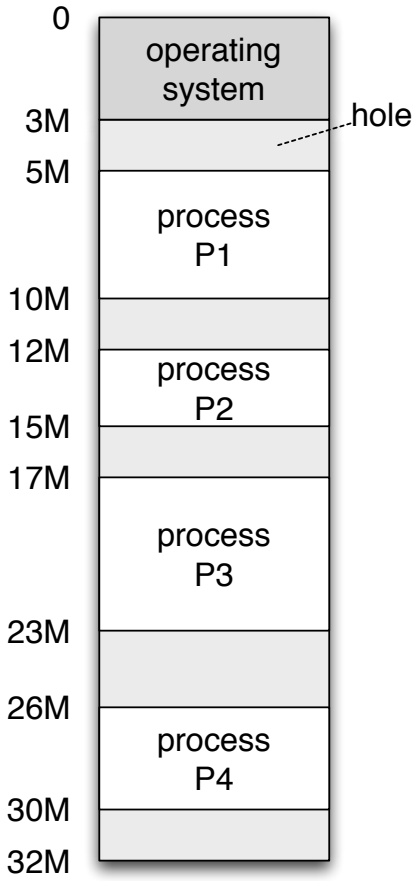
- First hole that is “big enough”

Best Fit

- Smallest possible hole

Worst Fit

- Largest available hole



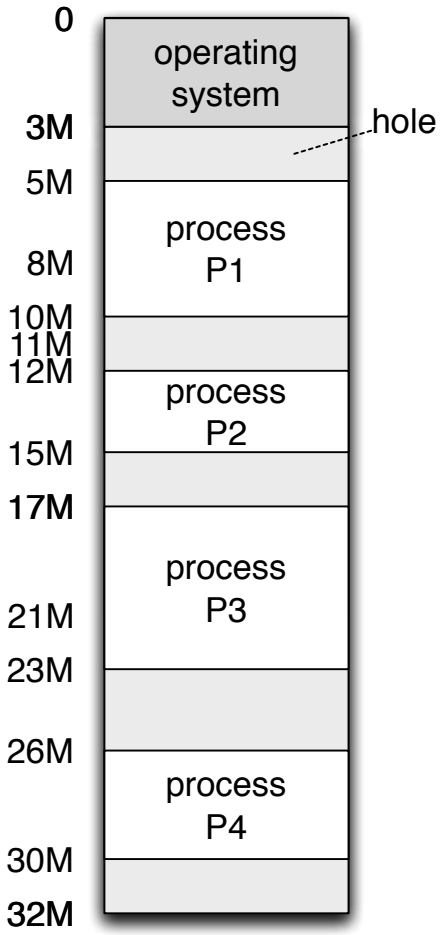
Fragmentation

External fragmentation

- Processes come and go
- They have different sizes
- Many small holes remain

Internal fragmentation

- New process comes
- A little smaller than hole
  → Useless tiny hole is created



Paging

Paging Overview

Idea

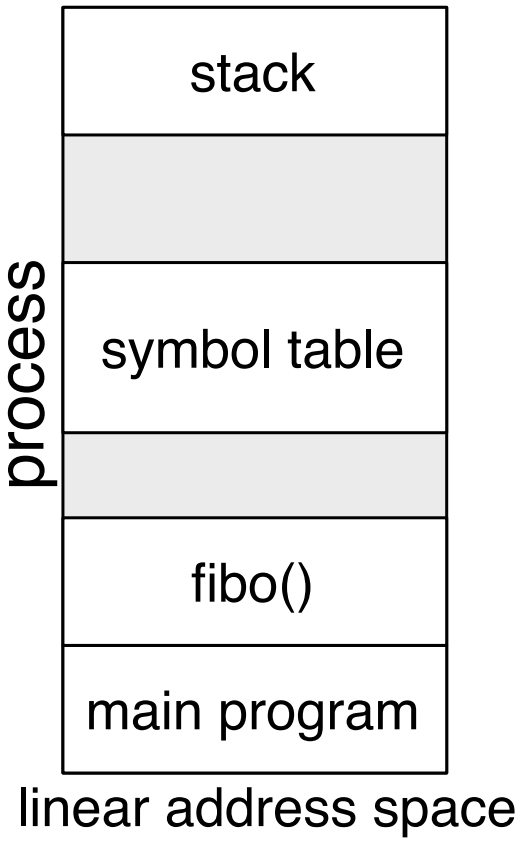
- Linear address space

Characteristics

- Code and data mixed
- Use fixed-size frames
- Transparent for programming

Goal

- Large linear virtual address space
- Protection between processes

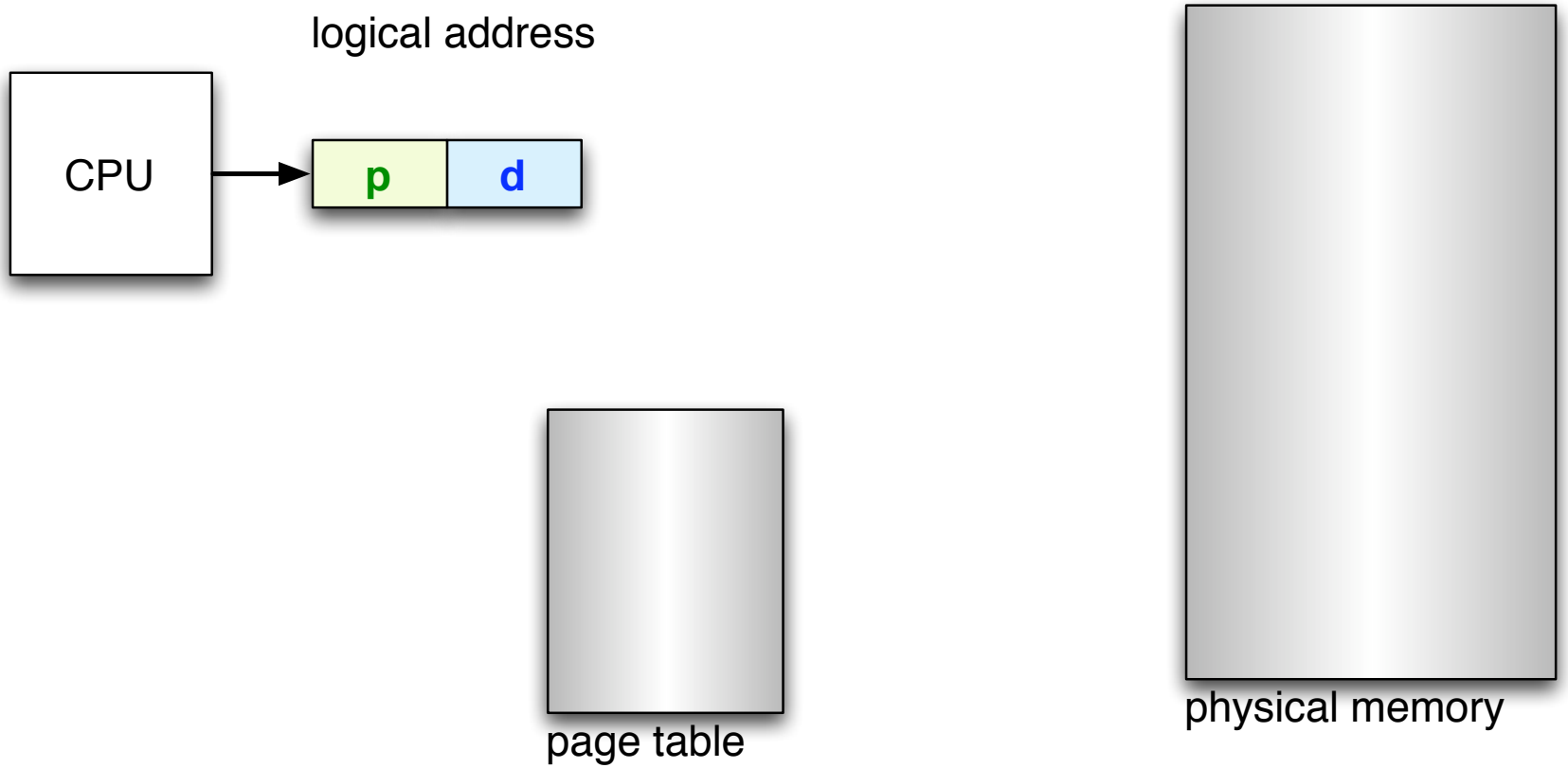


Paging Mechanisms

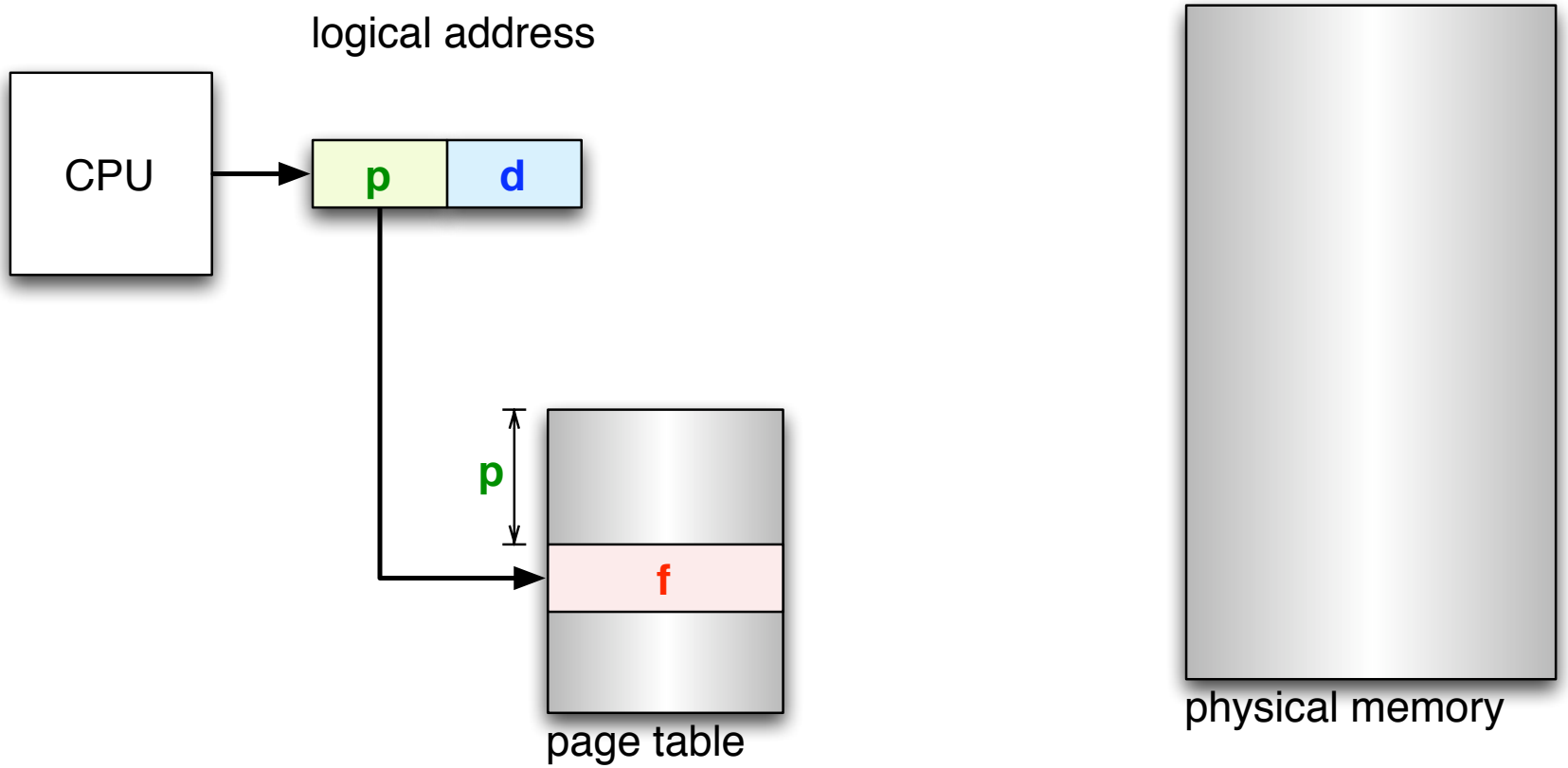
Paging Mechanisms (2)

p = *page number*
d = *memory offset*

Paging Mechanisms (3)

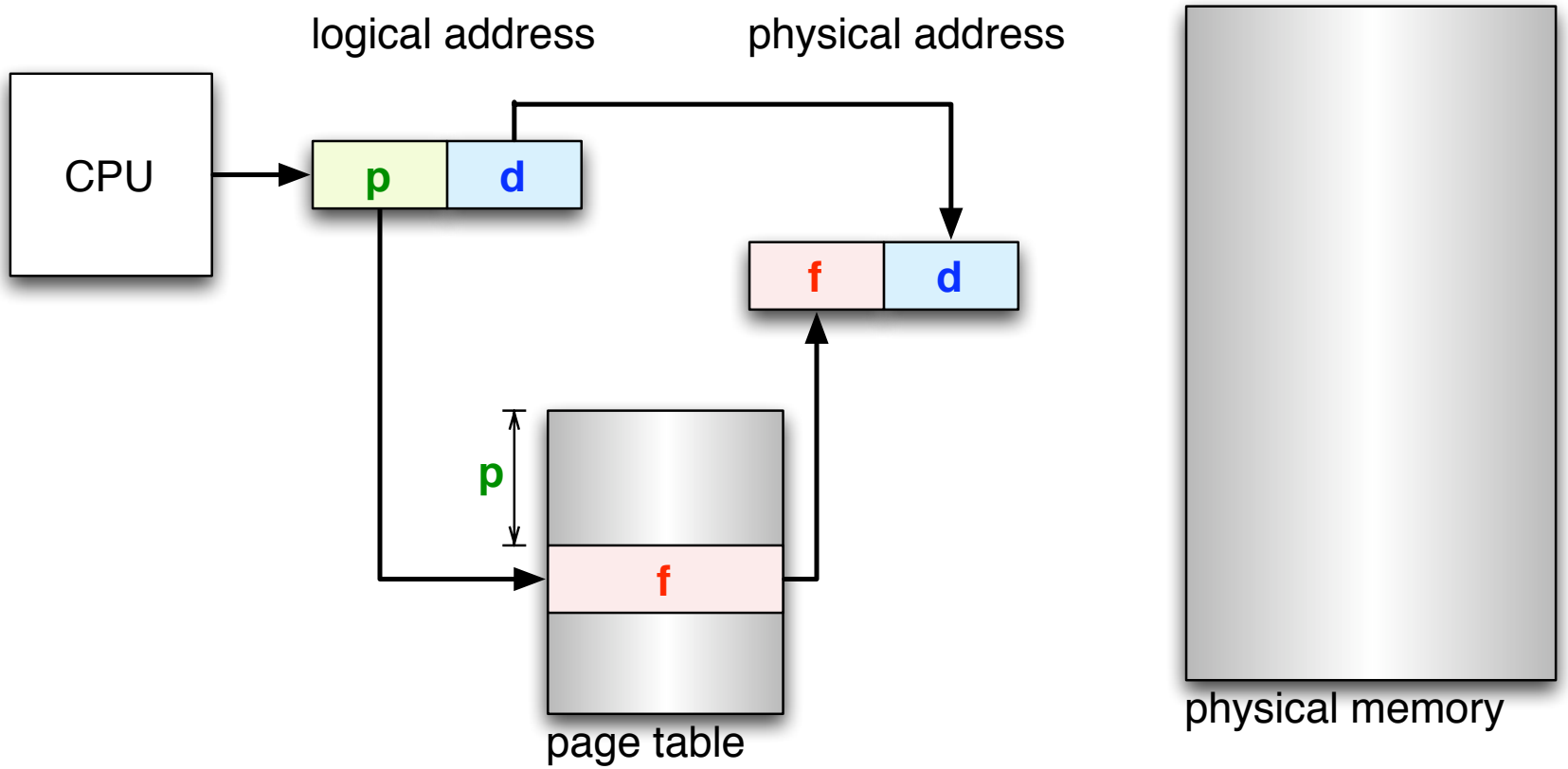


Paging Mechanisms (4)

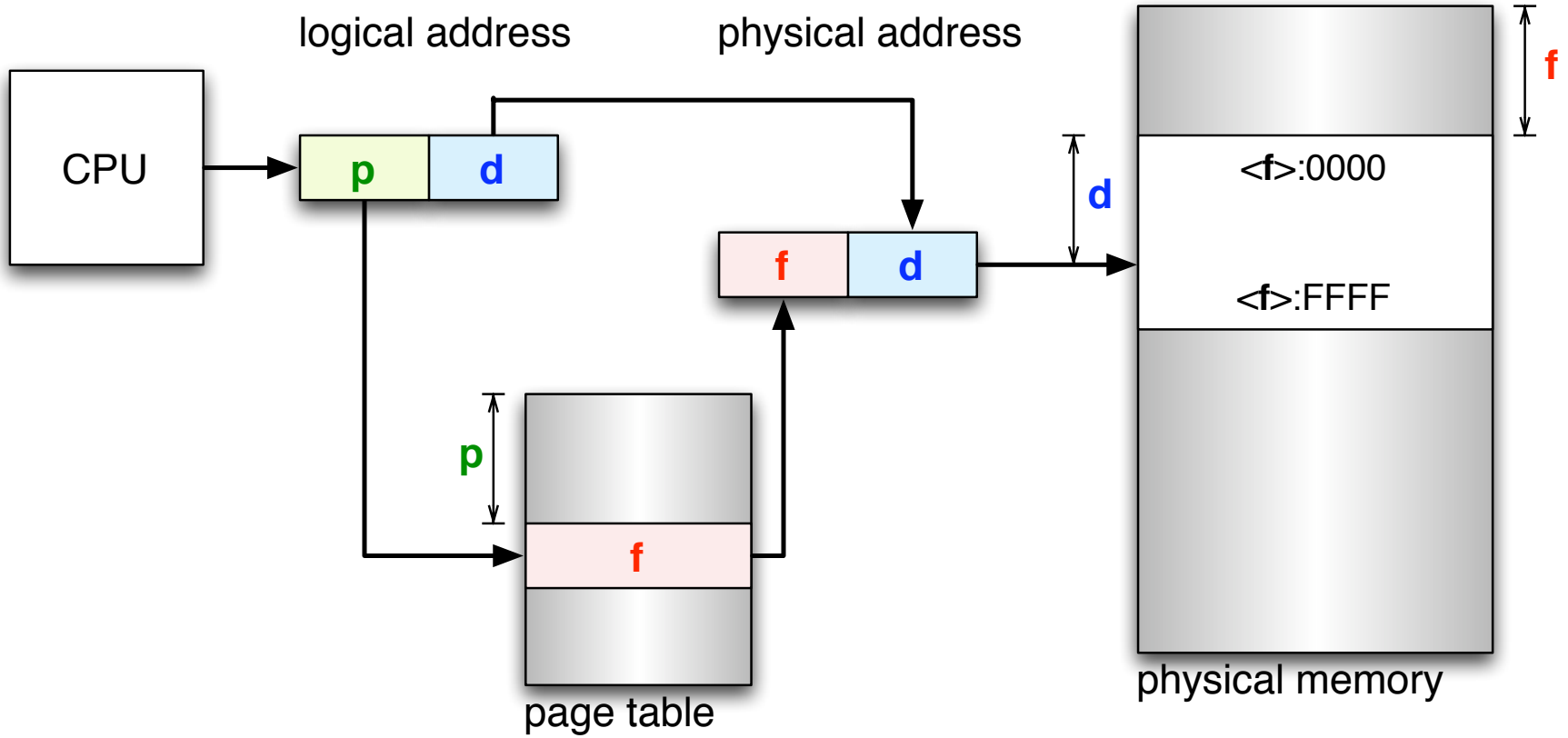


f = *frame number*

Paging Mechanisms (5)



Paging Mechanisms (6)



Paging Example

Process

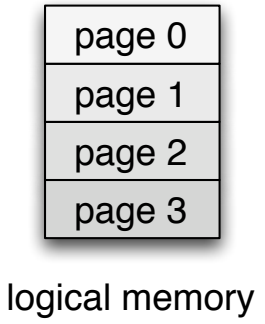
- Use logical memory
- Total of 4 pages
- Contiguous addresses

Page table

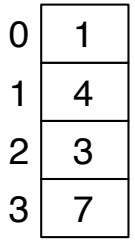
- Maps page to location

Physical memory

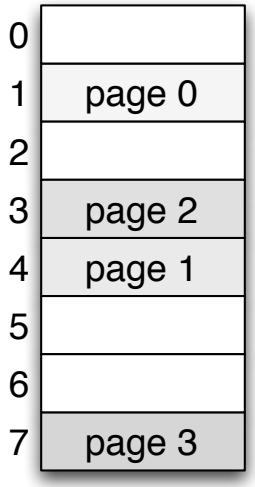
- Contains allocated pages
- Stored at various locations



logical memory

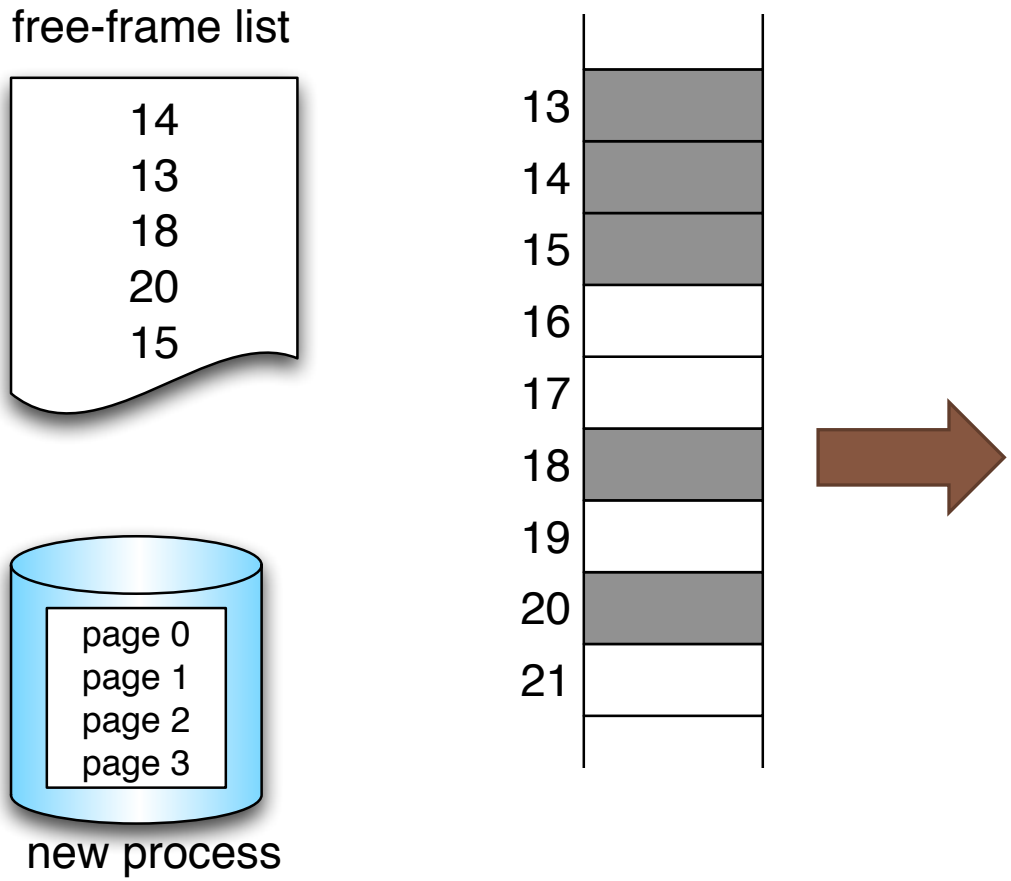


page table



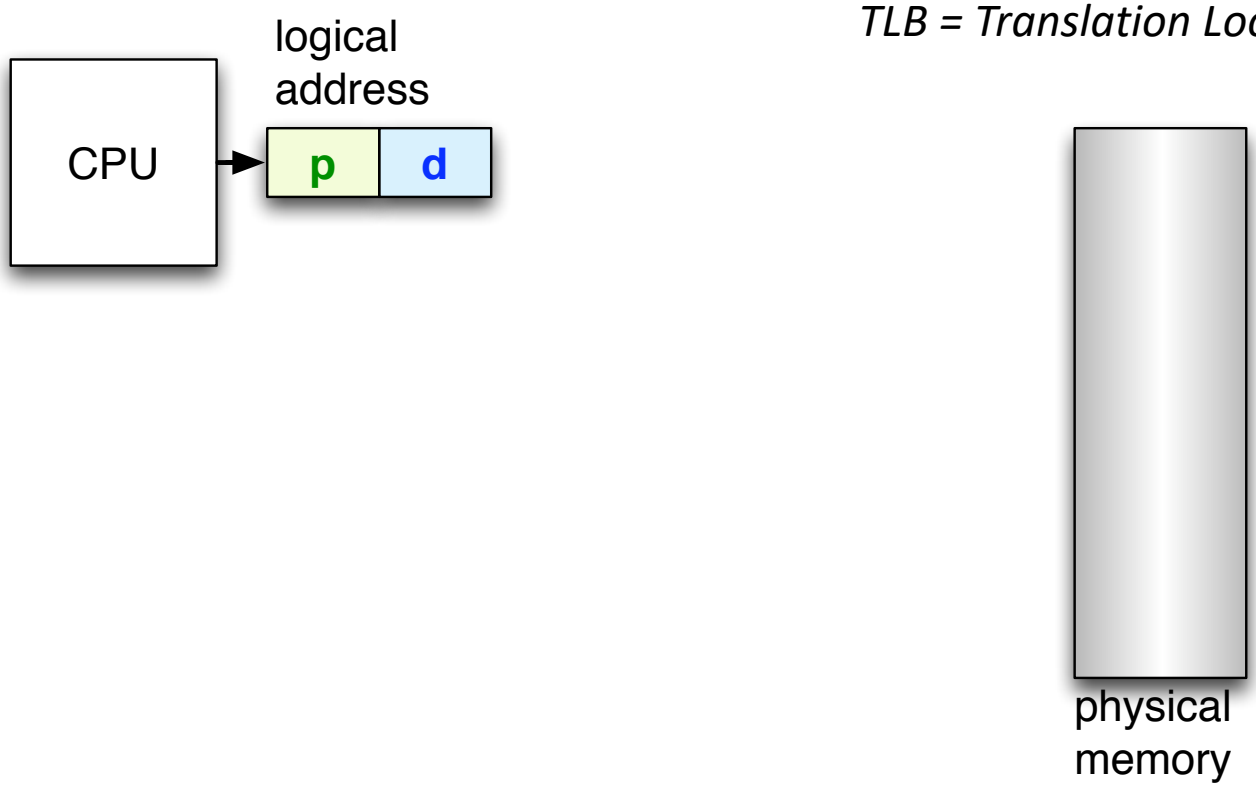
physical memory

Frame Allocation Example

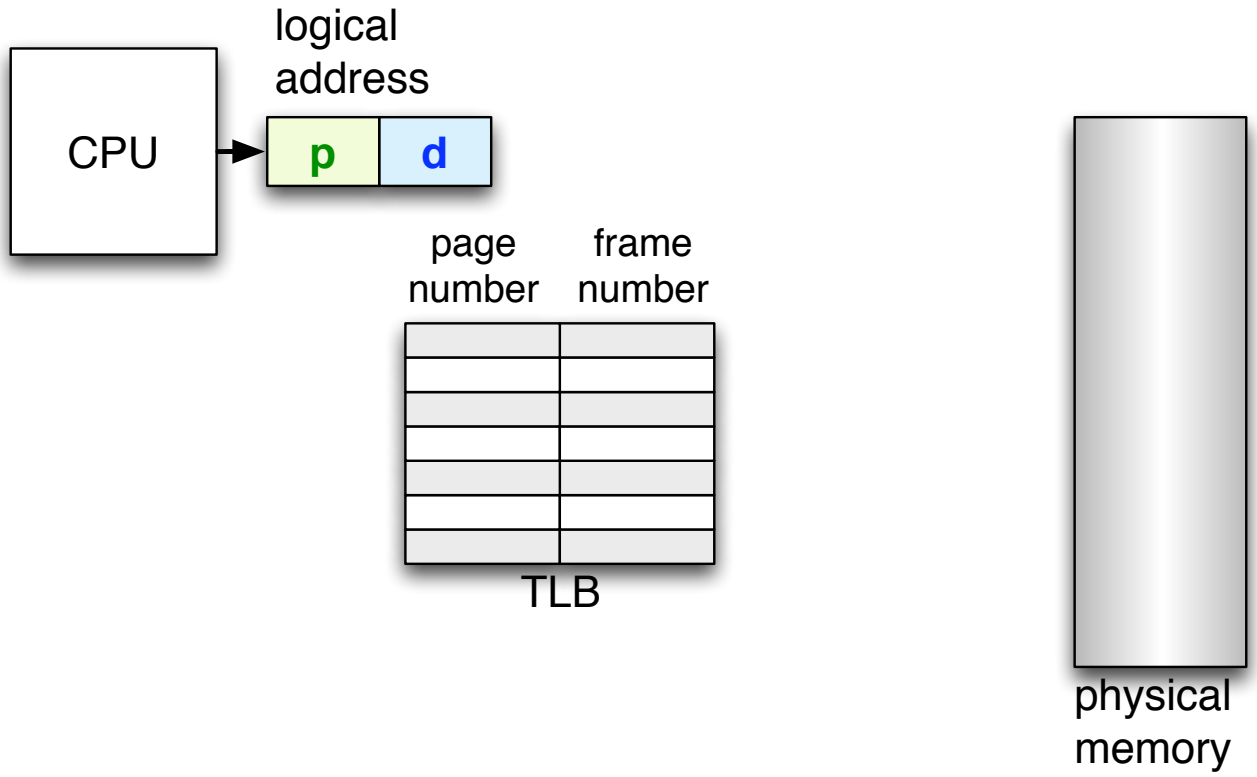


Hardware Support: TLB

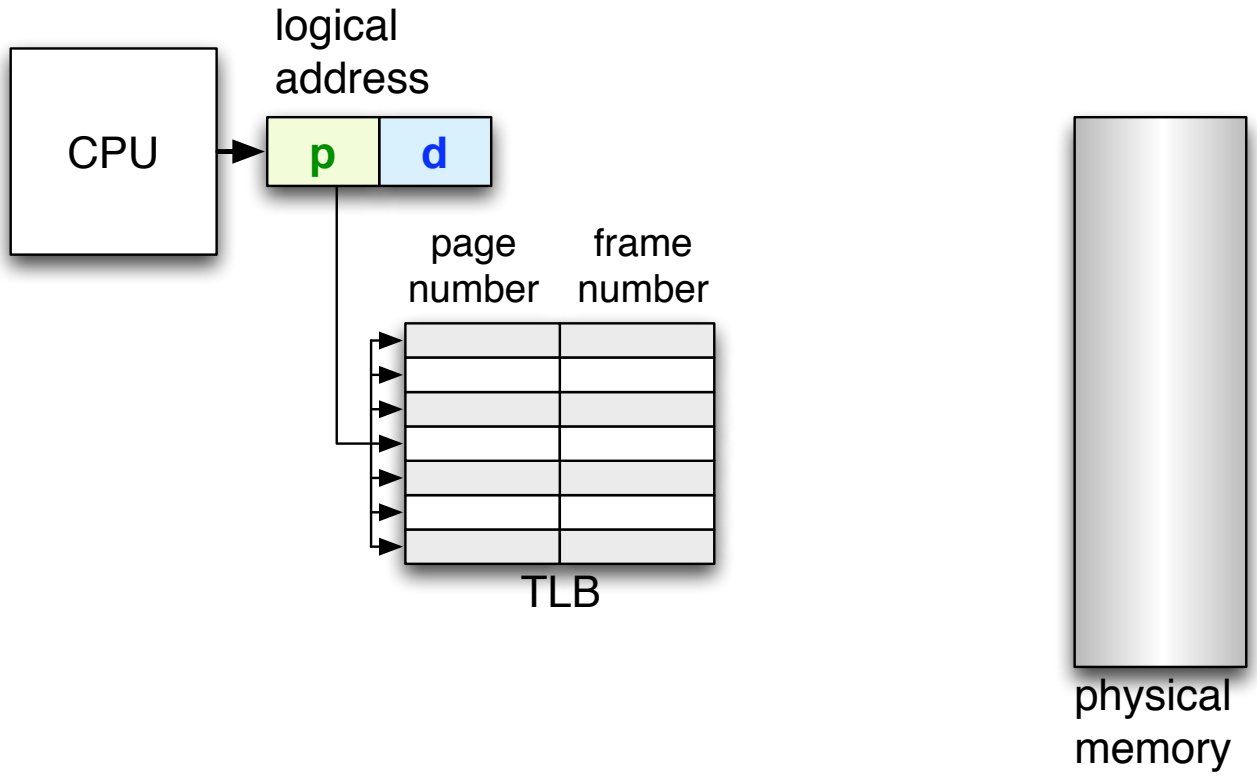
TLB = Translation Lookaside Buffer



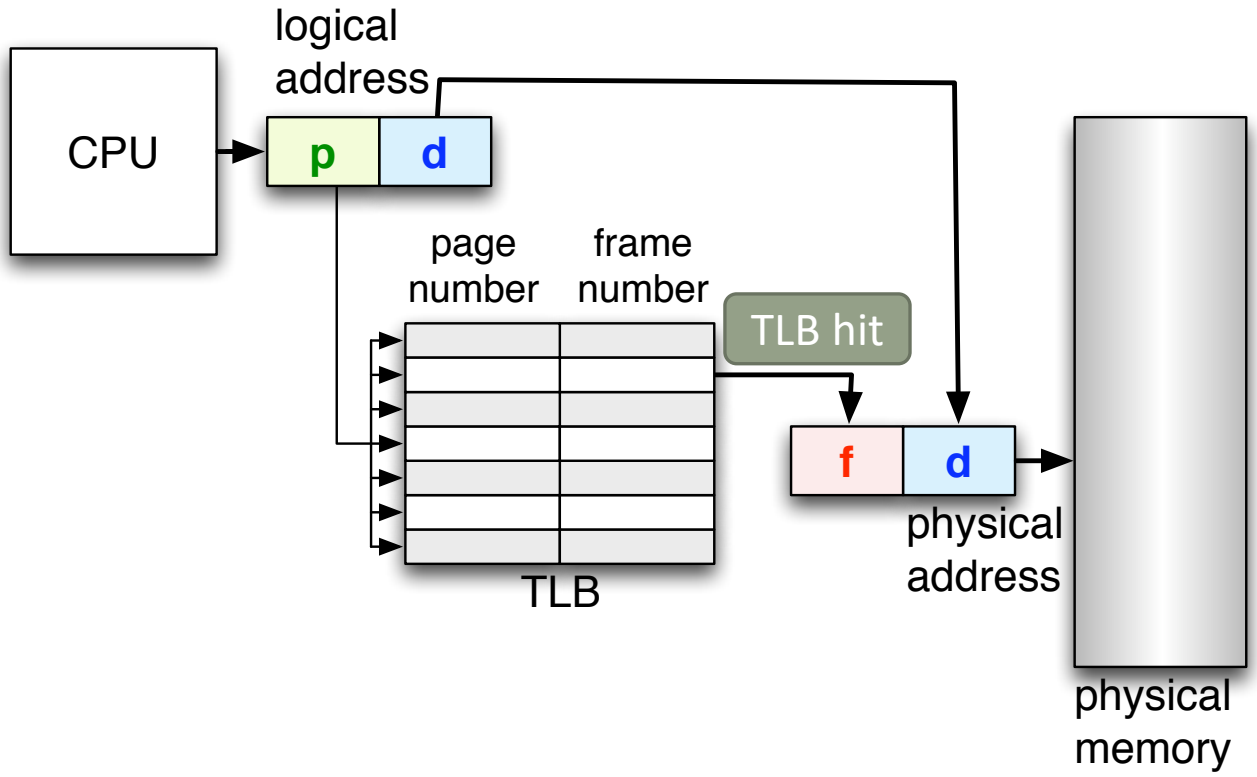
Hardware Support: TLB (2)



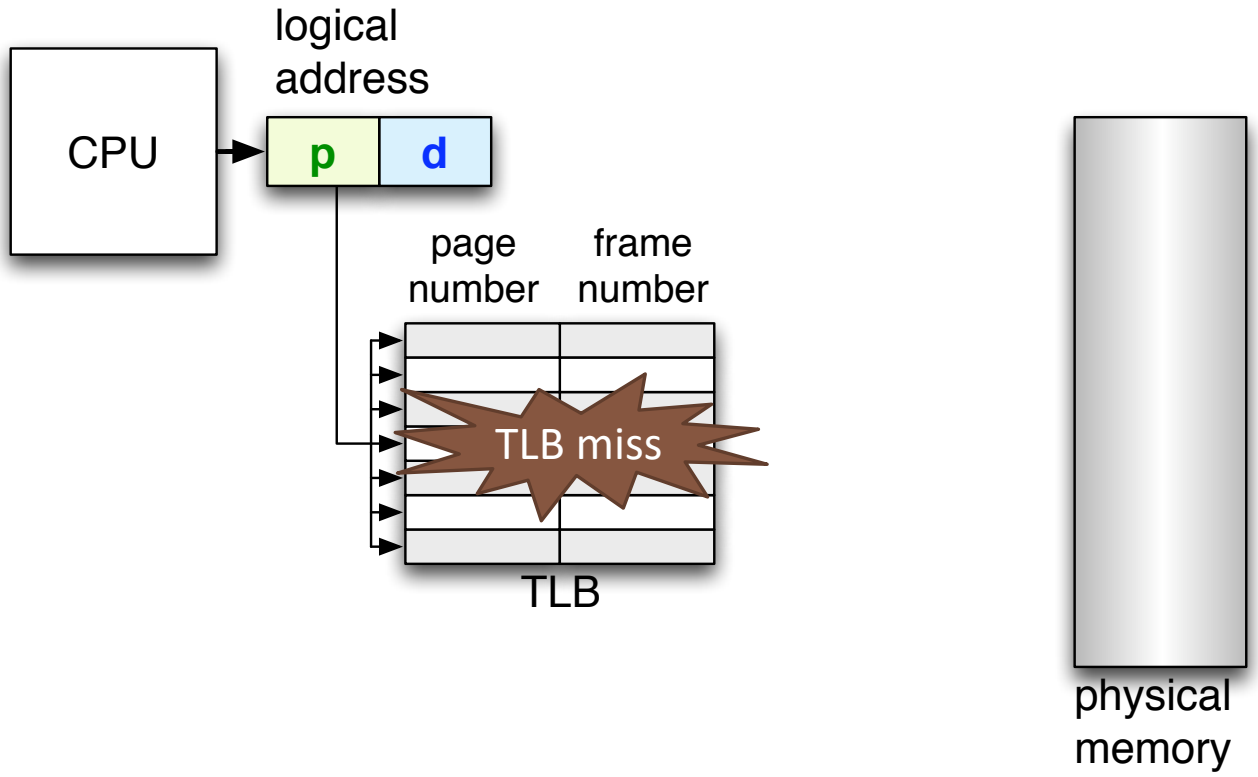
Hardware Support: TLB (3)



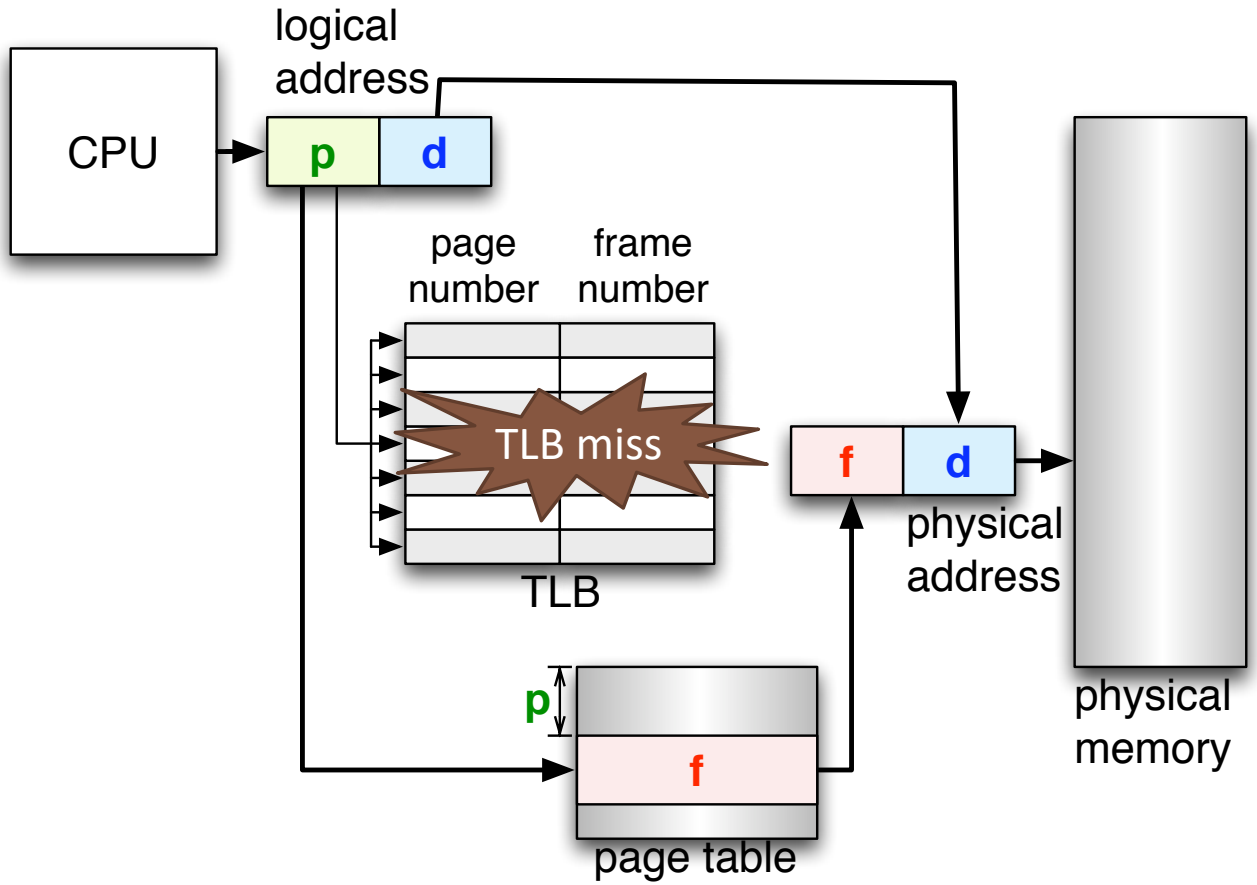
Hardware Support: TLB (4)



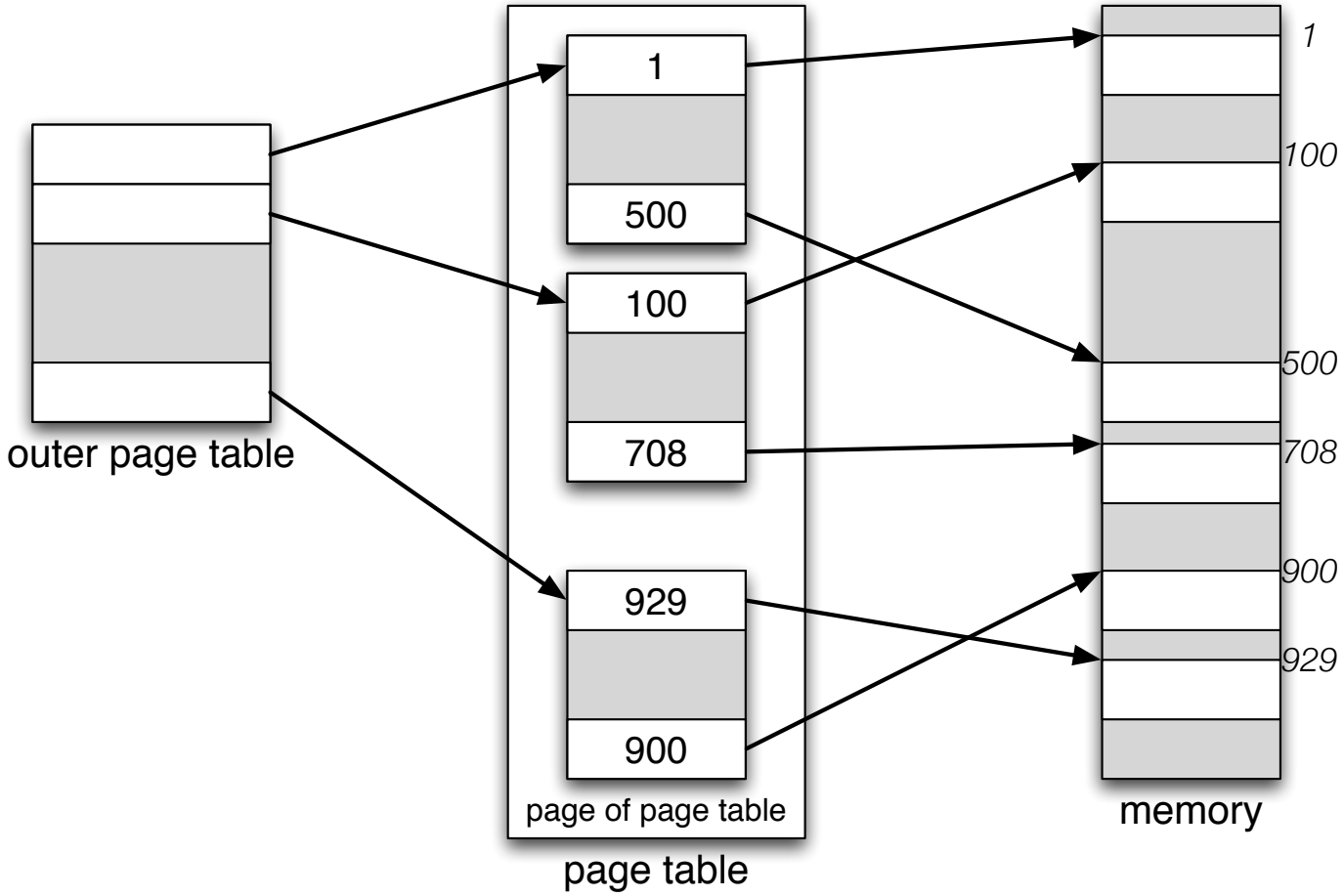
Hardware Support: TLB (5)



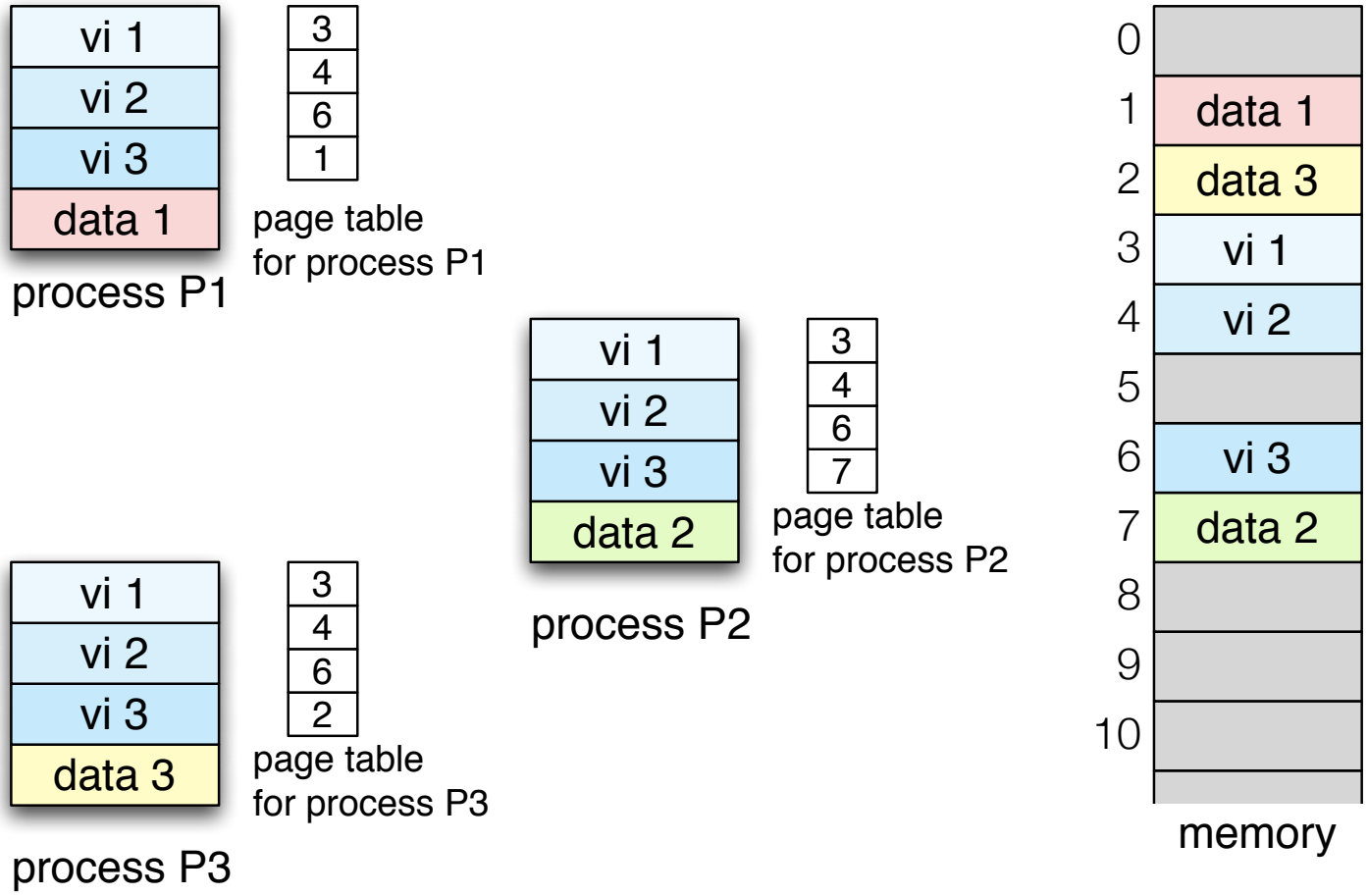
Hardware Support: TLB (6)



Two-Level Page Table



Shared Pages



Segmentation

Segmentation Overview

Idea

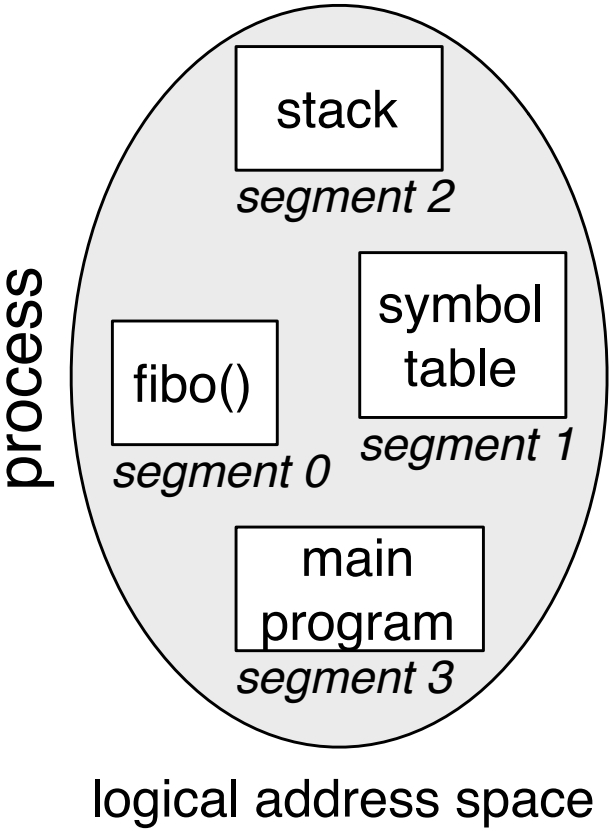
- Logical address space

Characteristics

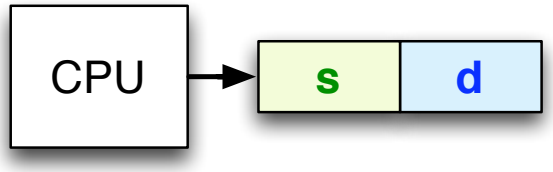
- Code and data separated
- Dynamic-size data OK
- Program must do management

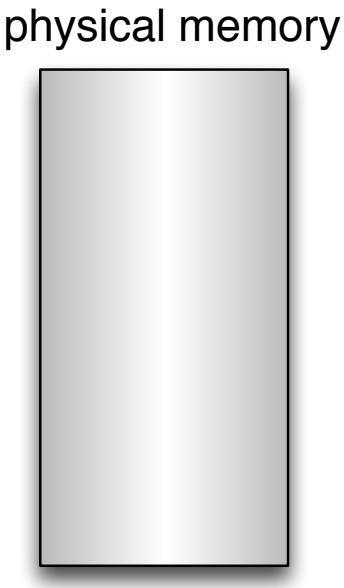
Goal

- Improved sharing
- Improved protection
- Improved structuring



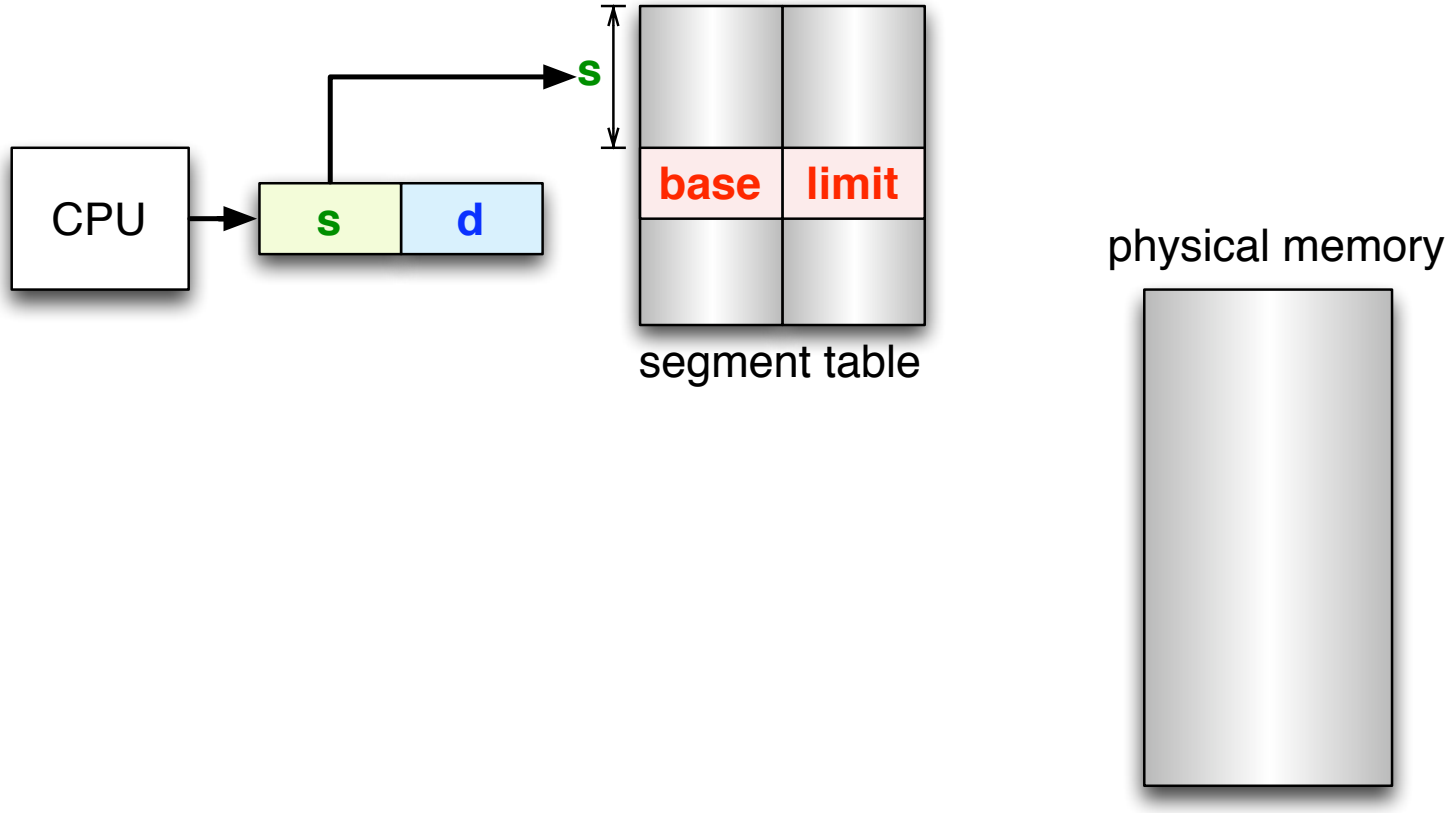
Segmentation Mechanisms



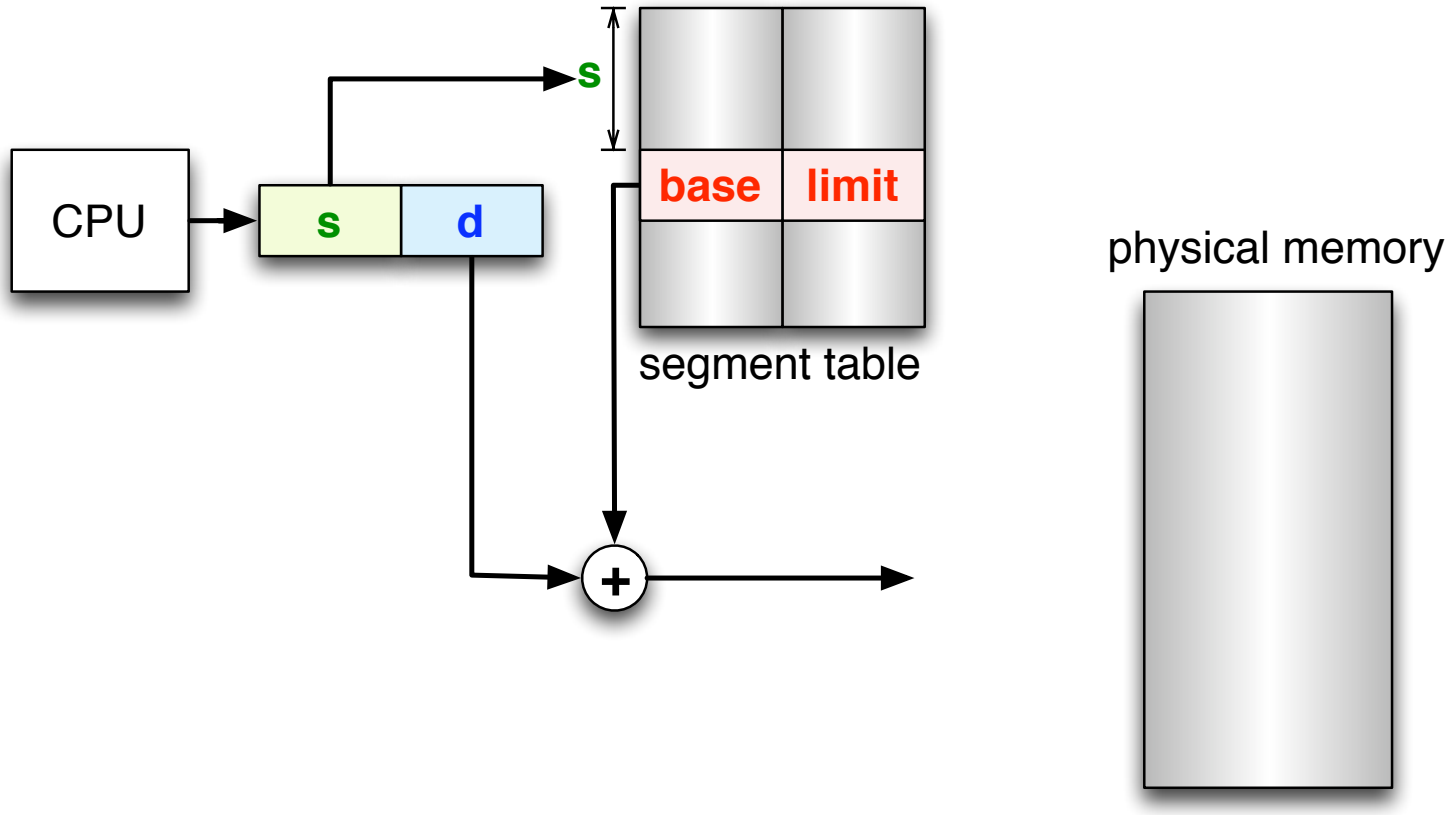


s = *segment number*
d = *memory offset*

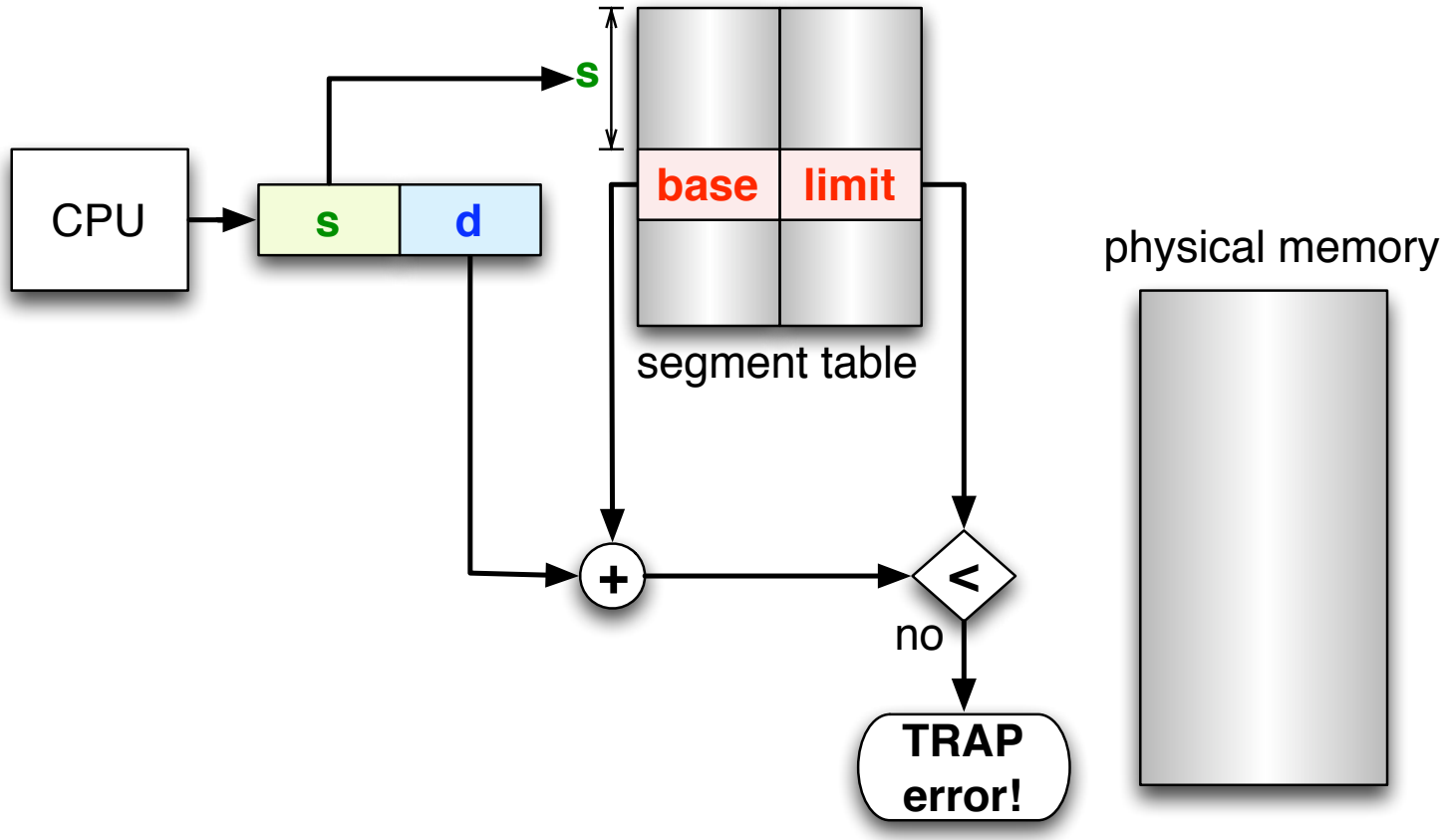
Segmentation Mechanisms (2)



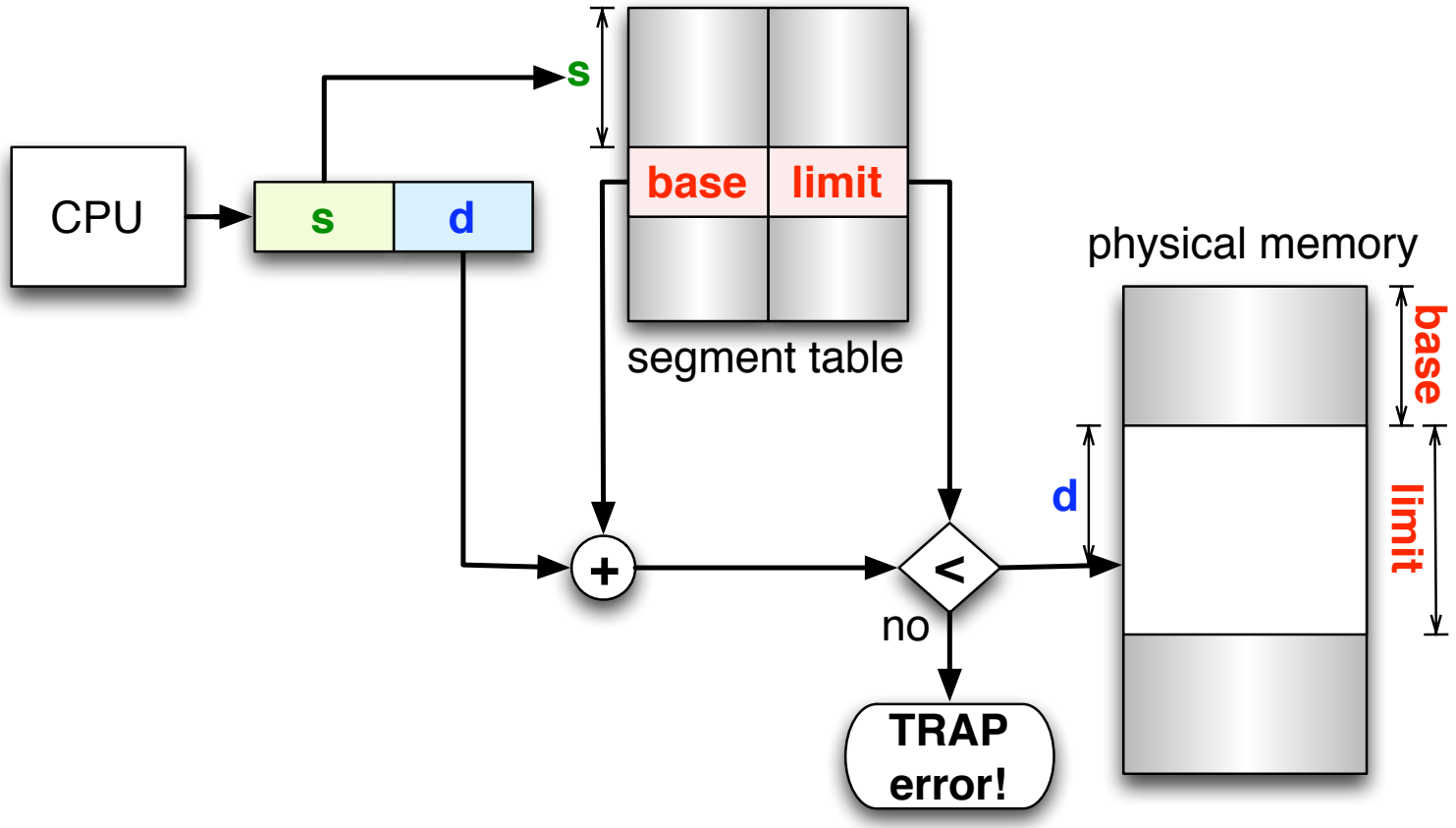
Segmentation Mechanisms (3)



Segmentation Mechanisms (4)



Segmentation Mechanisms (5)



Protection and Sharing

Protection

- Different protections possible for each segment
- E.g., read-only, execute-only, etc.

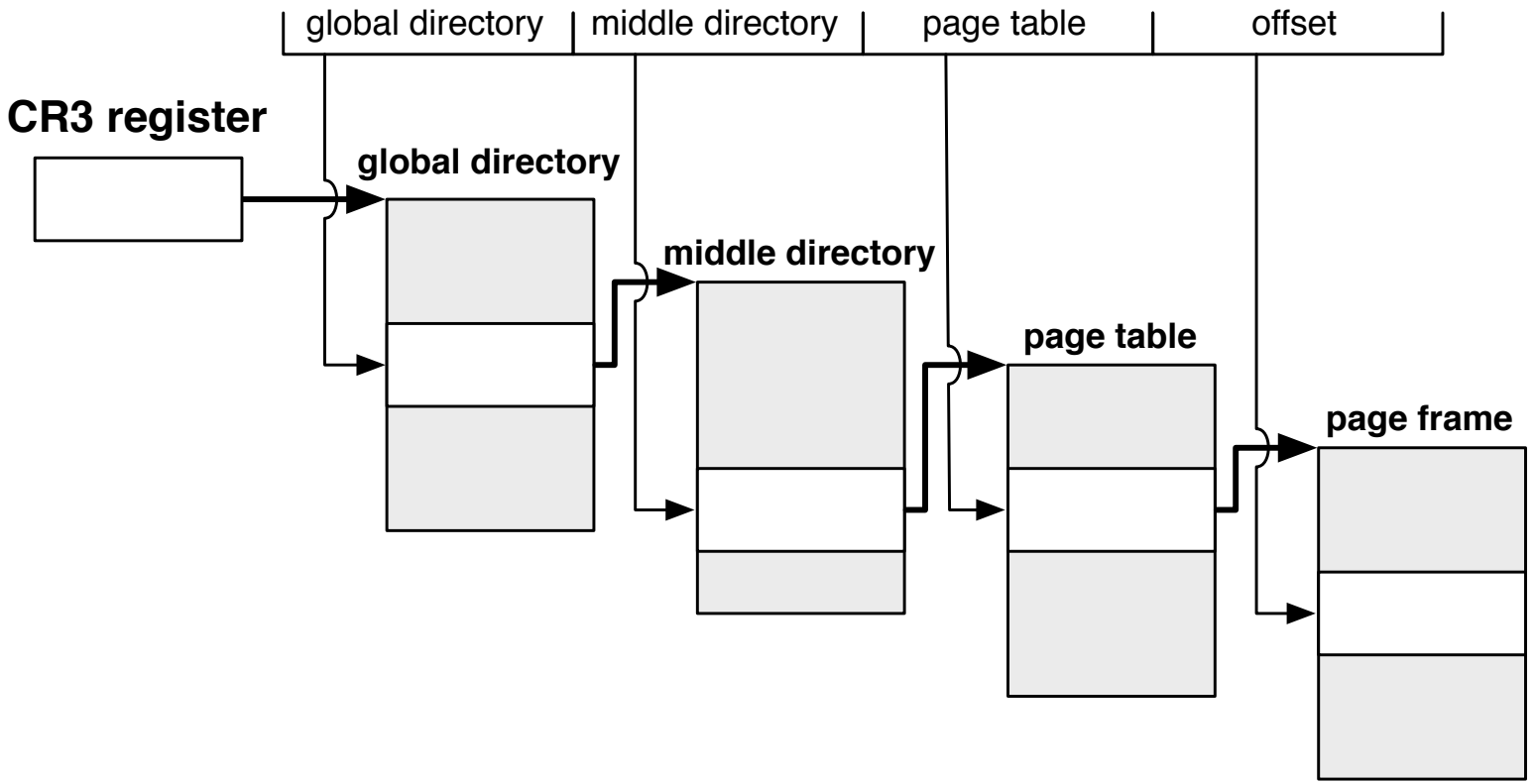
Sharing

- Protected segments can be shared
- E.g., portions of code, shared libraries

Fragmentation

- Segments have *variable* length
  → Problem with external fragmentation

Memory Management in Linux



Summary

Main memory

- Needed to store running programs
- Issues: swapping, fragmentation

Paging

- Mechanism for transparently managing memory by dividing it into several fixed-size areas

Segmentation

- Mechanism that improves sharing and protection compared to paging by using variable-size areas with known limit

Next Time

Virtual memory

Page fault

Page replacement