CSE 120: Principles of Operating Systems

Lecture 8

Segmentation and Paging

October 28, 2003

Prof. Joe Pasquale

Department of Computer Science and Engineering

University of California, San Diego

Before We Begin …

Read Chapter 9 (on Paging and Segmentation)

Midterm grading

Programming assignment 2 will be available this weekend

- due in 2 weeks, Sunday Nov 16 at midnight

Structure of Process Address Space

Text: program instructions

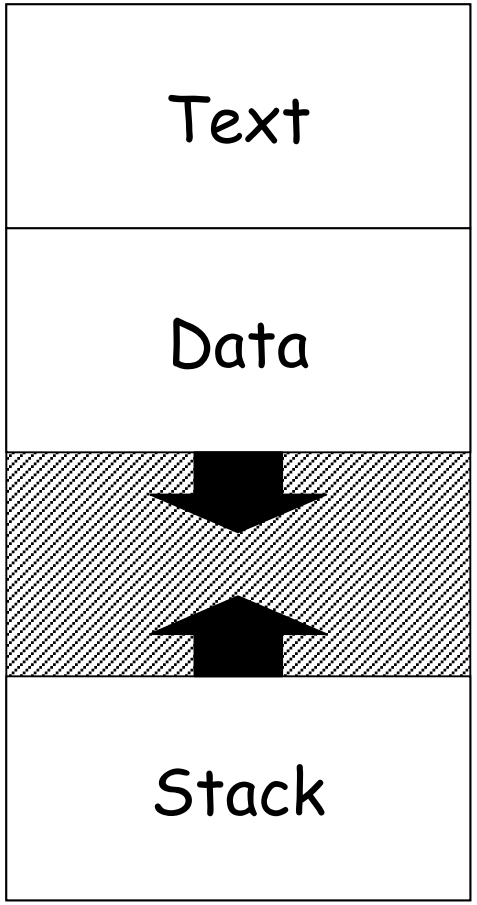
- execute-only, fixed size

Data: variables (global, heap)

- read/write, variable size
- dynamic allocation by request

Stack: activation records

- read/write, variable size
- automatic growth/shrinkage



Segmented Address Space

Address space is a set of segments

Segment: a linearly address memory

- typically contains logically-related information
- examples: program code, data, stack

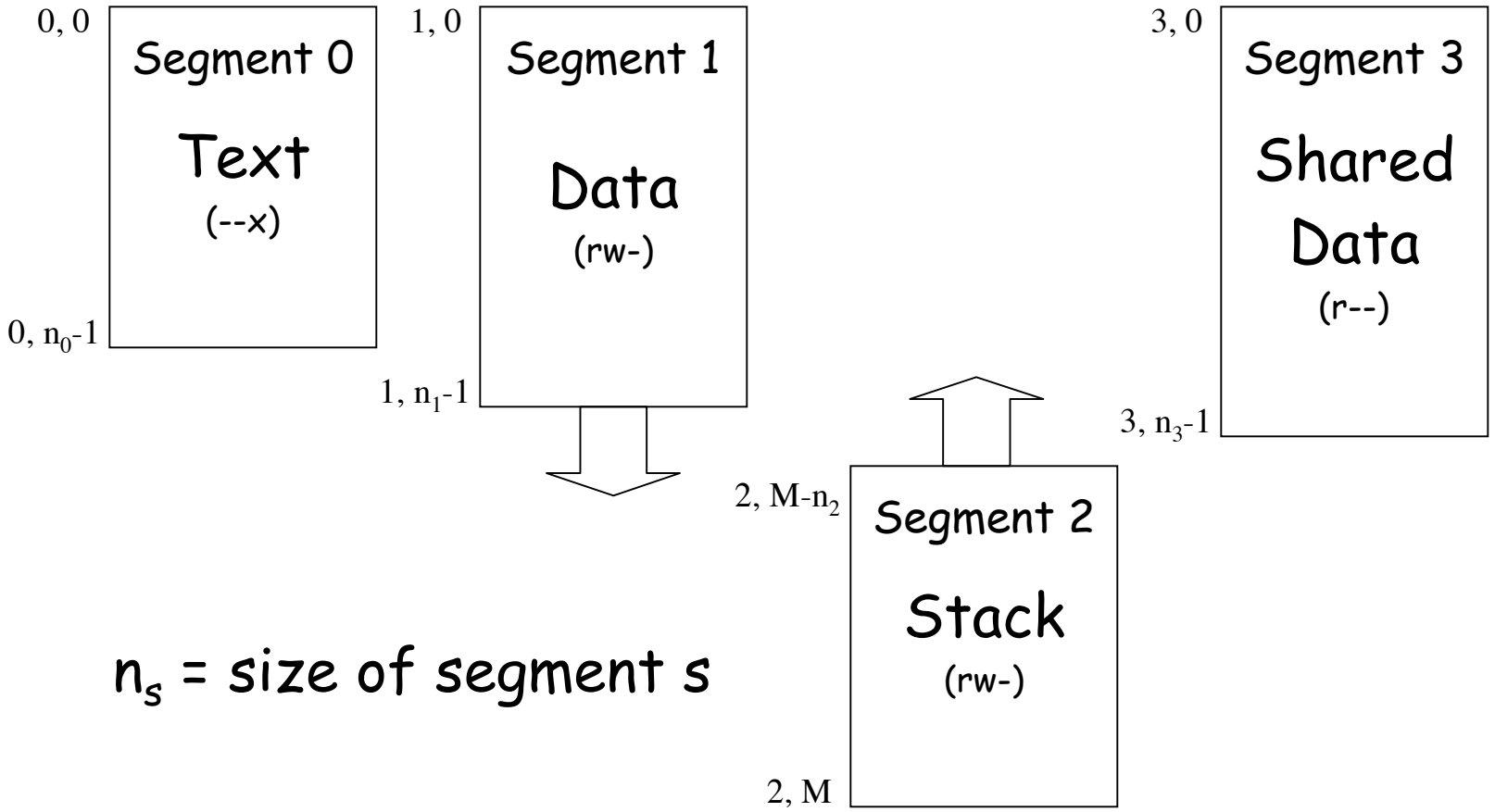
Each segment has an identifier s, and a size n

- s between 0 and S-1, S = number of segments

Logical addresses are of form (s, i)

- offset i within segment s, i must be less than n

Ex. of Segmented Address Space



n_s = size of segment s

Address Translation for Segments

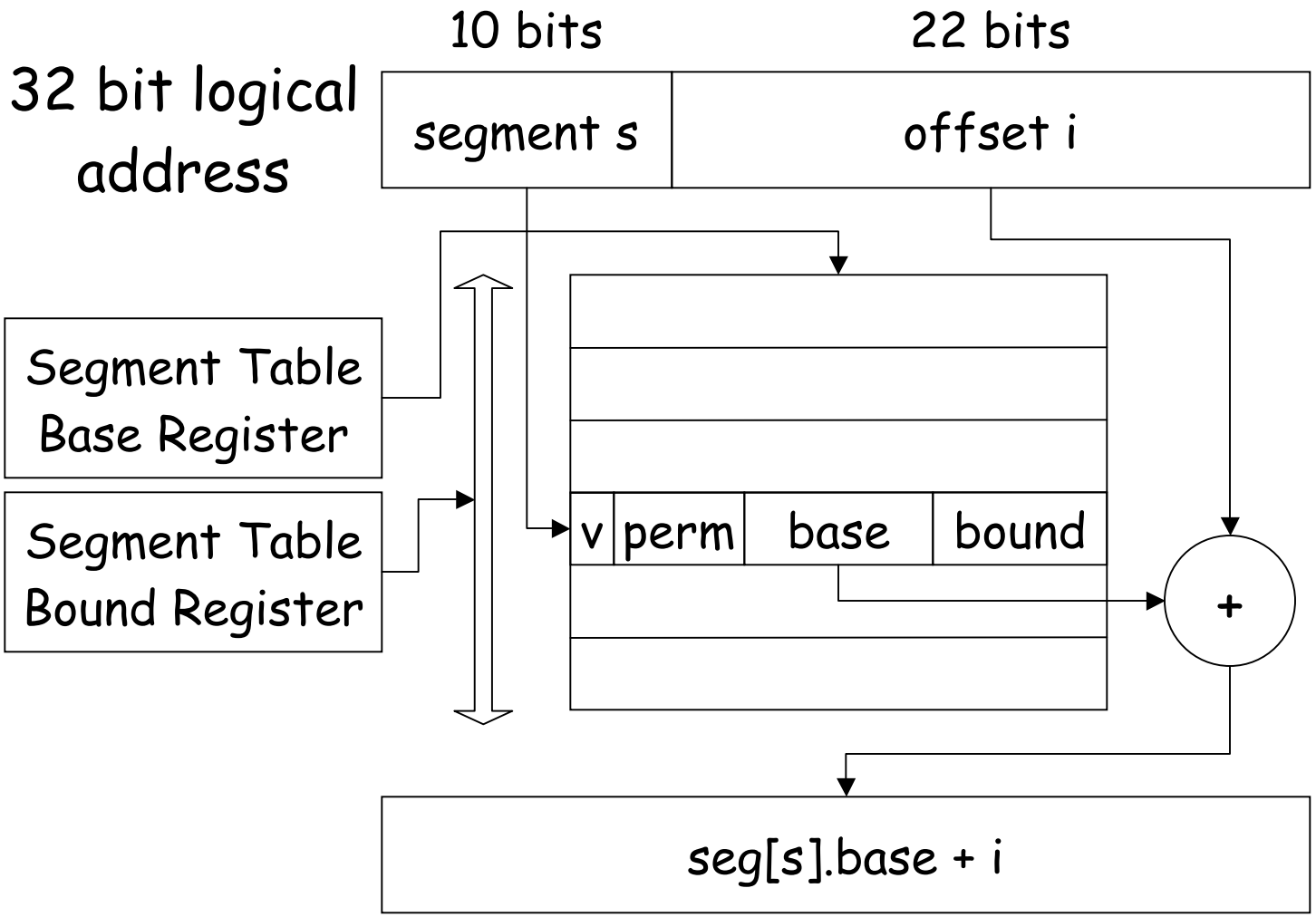
Segment table contains, for each segment s

- base, bound, permissions, (+ valid bit)

Logical to physical address translation

- check if operation is permitted
- check if i < s.bound
- physical address = s.base + i

Example of Address Translation



Advantages of Segmentation

Each segment can be

- located independently
- separately protected
- grow independently

Segments can be shared between processes

Problems with Segmentation

Variable allocation

Difficult to find holes in physical memory

Must use one of non-trivial placement algorithm

- first fit, next fit, best fit, worst fit

External fragmentation

Paged Address Space

Address space is linear sequence of pages

Page

- physical unit of information
- fixed size

Physical memory is linear sequence of frames

- a page fits exactly into a frame

Addressing

Each page is identified by a page number 0 to N-1

- N = number of pages in address space
- N * pagesize = size of address space

Logical addresses are of form (p, i)

- offset i within page p
- i is less than page size

Address Translation for Pages

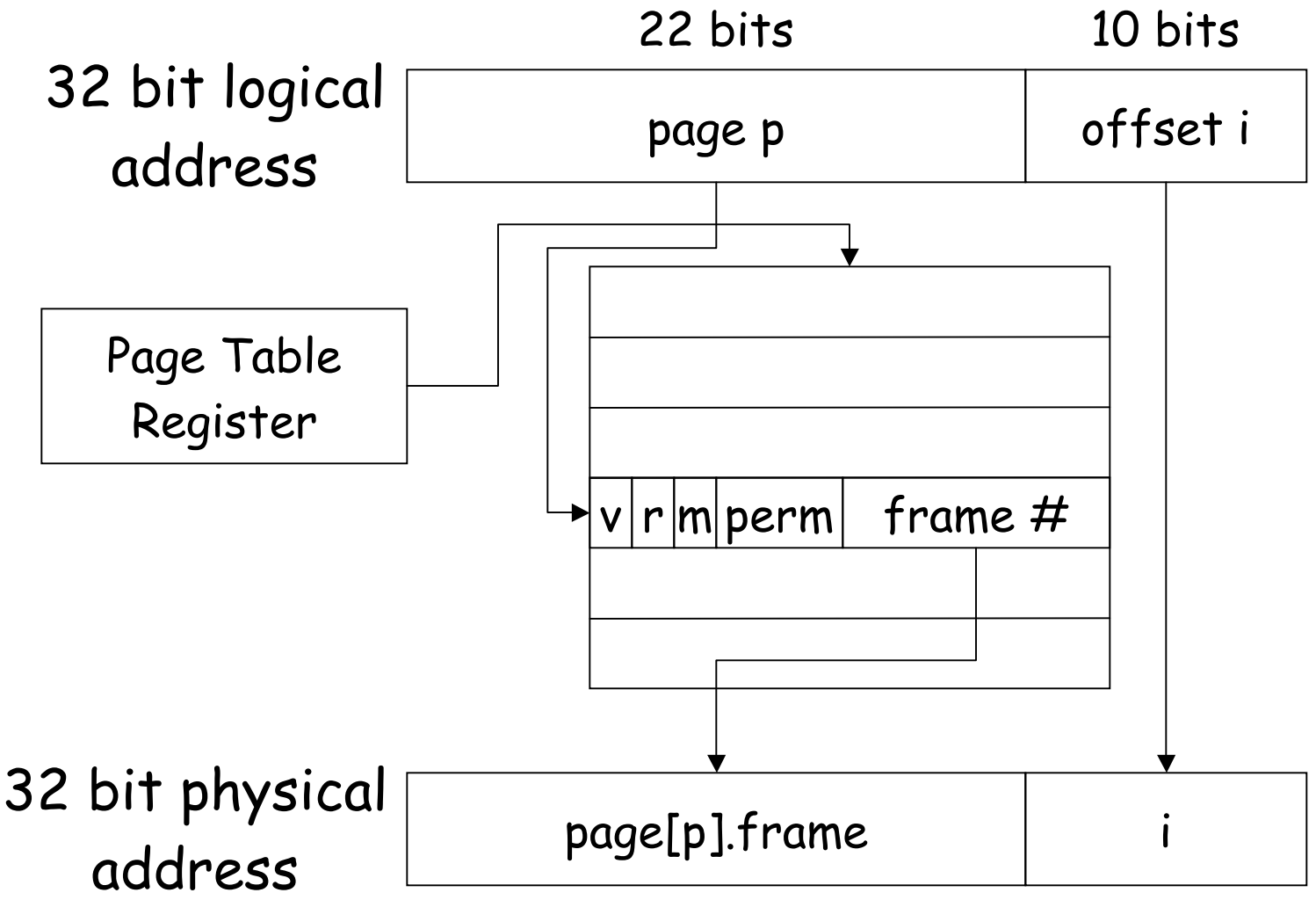
Page table contains, for each page p

- frame number that corresponds to p
- other: perms, valid bit, reference bit, modified bit

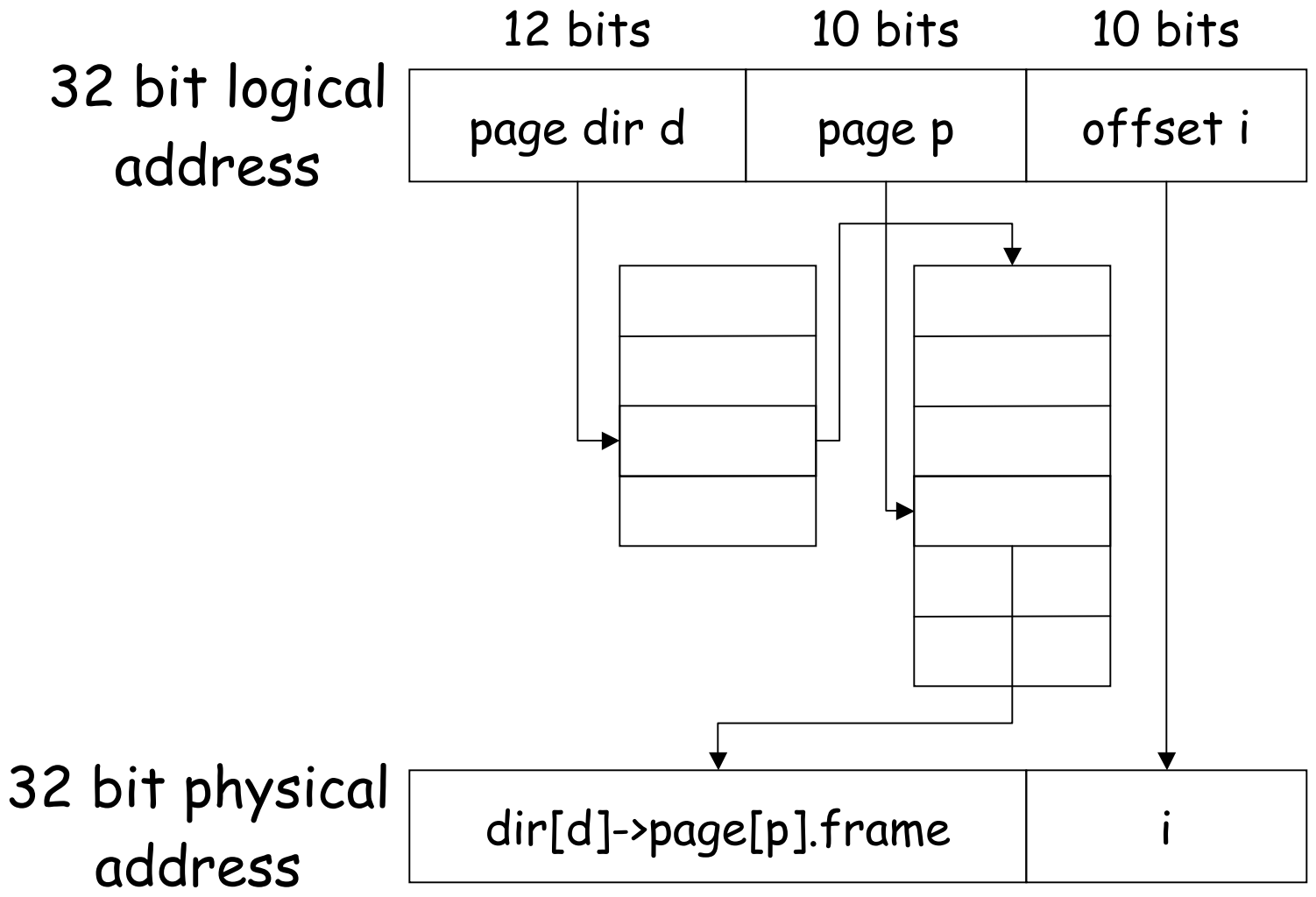
Logical address (p, i) to physical address translation

- check if operation is permitted
- physical address = p.frame + i

Example of Address Translation



Multi-Level Page Tables



Segmentation vs. Paging

Segment is good logical unit of information

- sharing, protection

Page is good physical unit of information

- simple memory management

Best of both

- segmentation on top of paging

Combining Segmentation and Paging

Logical memory is composed of segments

- each segment is composed of pages

Segment table

- per process, in memory pointed to by register
- entries map seg # to page table base
- shared segment: entry points to shared page table

Page tables (like before)

Address Translation

Logical address: segment #, page #, and offset

Index seg # into seg table: get base of page table

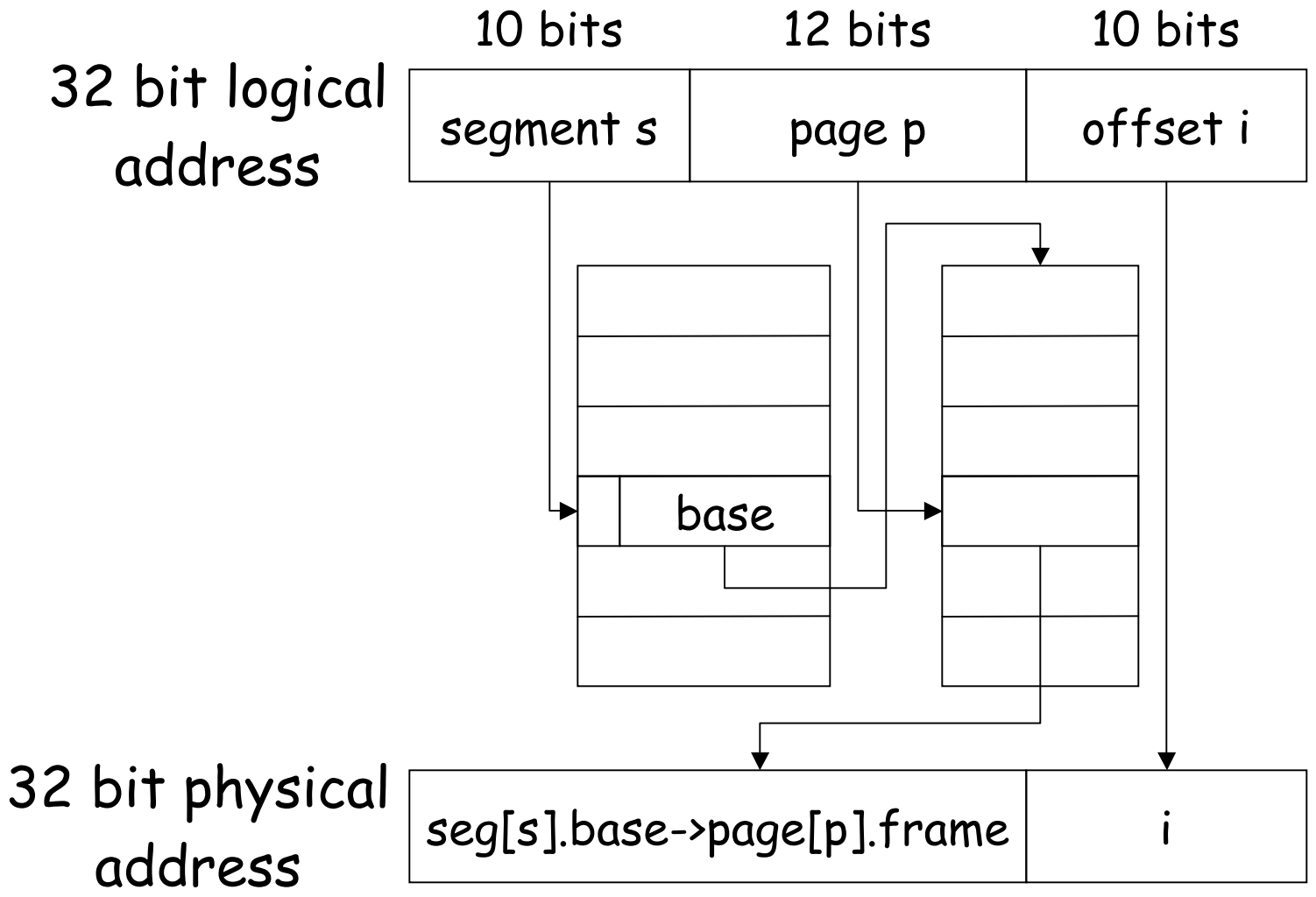
Check bounds (number of pages using page #)

- may get a segmentation violation

Use page # to index into page table, get frame #

Concatenate frame # and offset to get physical address

Example of Address Translation



Cost of Translation

Each page table lookup costs a memory reference

- for each reference, additional references required
- slows machine down by factor of 2 or more

Take advantage of locality of reference

- most references are to a small number of pages
- keep translations of these in high-speed memory

Problem: we don't know which pages until referenced

Translation Lookaside Buffer (TLB)

Fast associative memory keeps most recent translations

logical page	page frame

Determine whether non-offset part of LA is in TLB

- yes: get corresponding frame num for phys addr
- no: wait for normal memory translation (parallel)

Translation Cost with TLB

Cost is determined by

- speed of memory: ~ 100 nsec
- speed of TLB: ~ 20 nsec
- hit ratio: fraction of refs satisfied by TLB, ~95%

Speed with no address translation: 100 nsec

Speed with address translation

- TLB miss: 200 nsec (100% slowdown)
- TLB hit: 120 nsec (20% slowdown)
- average: 120 x .95 + 200 x .05 = 124 nsec

TLB Design Issues

The larger the TLB

- the higher the hit rate
- the slower the response
- the greater the expense

TLB has a major effect on performance!

- must be flushed on context switches
- alternative: tagging entries with PIDs

MIPS: has only a TLB, no page tables!

- devote more chip space to TLB