


Comp 310
Computer Systems and Organization

Lecture #16
Memory Management
(Memory Allocation – Part 2)

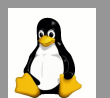
Prof. Joseph Vybihal




Announcements

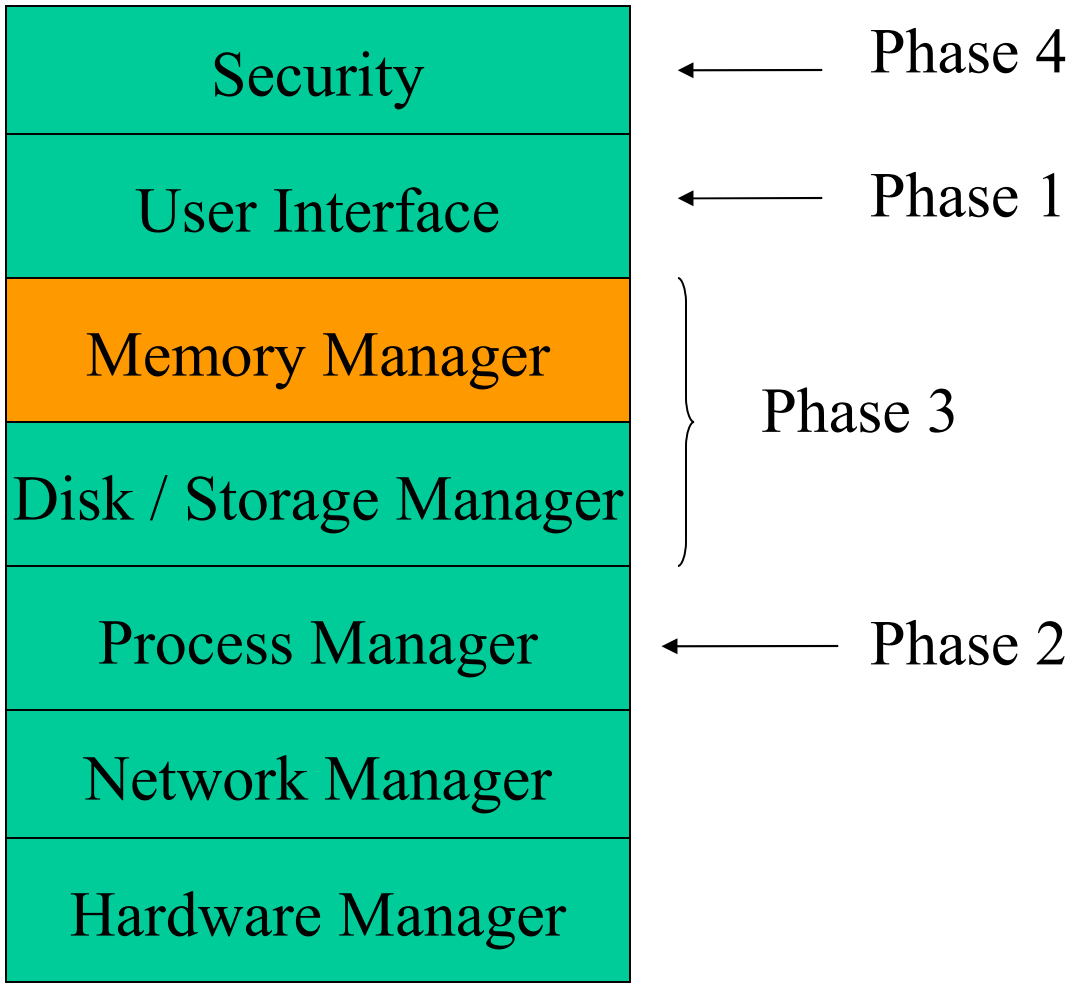
- Course evaluation:
  - Minerva
  - Dates?
  - Important to participate
- Ass#3 out tonight






Basic OS Architecture

(Course Table of Contents)

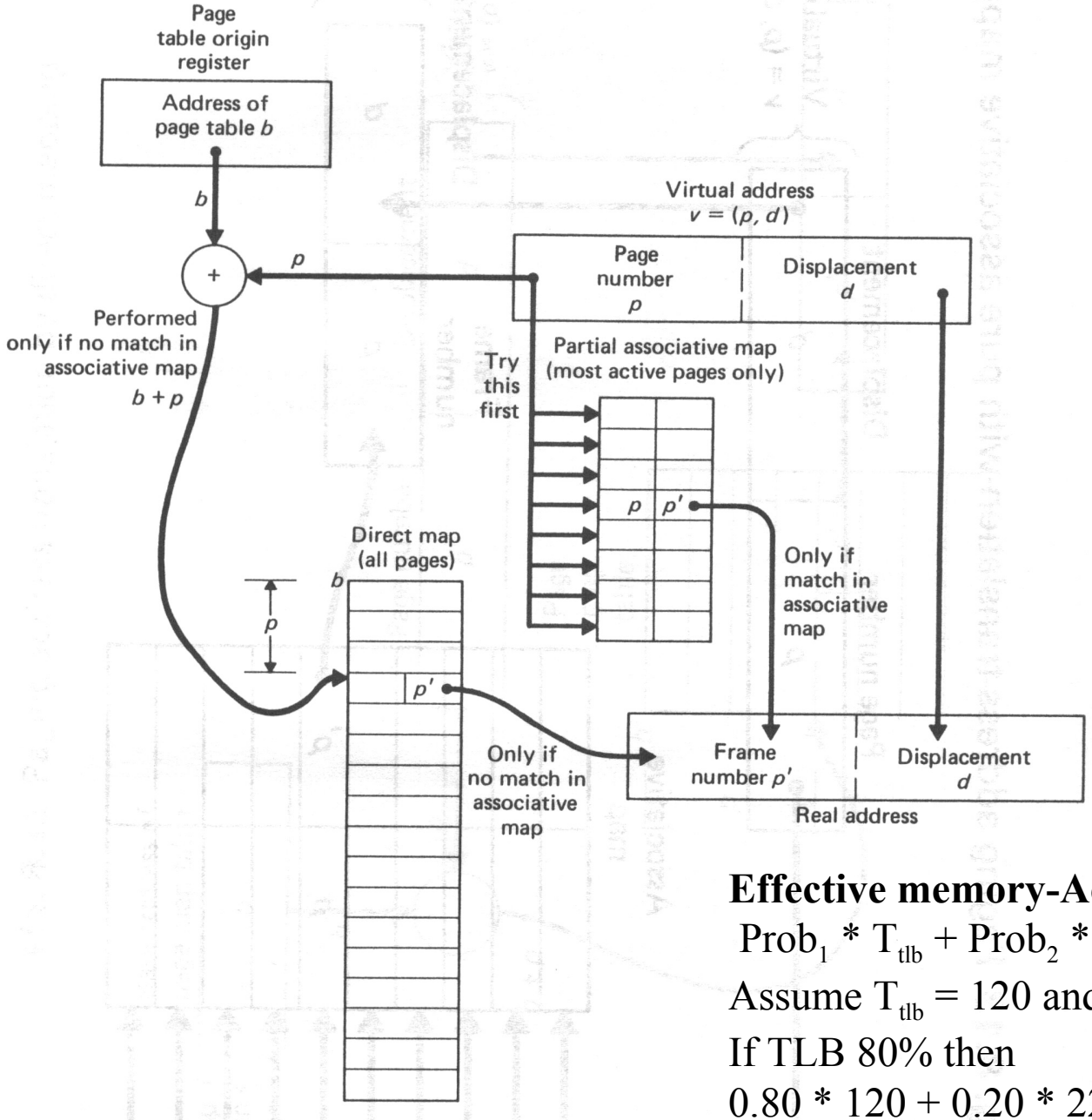




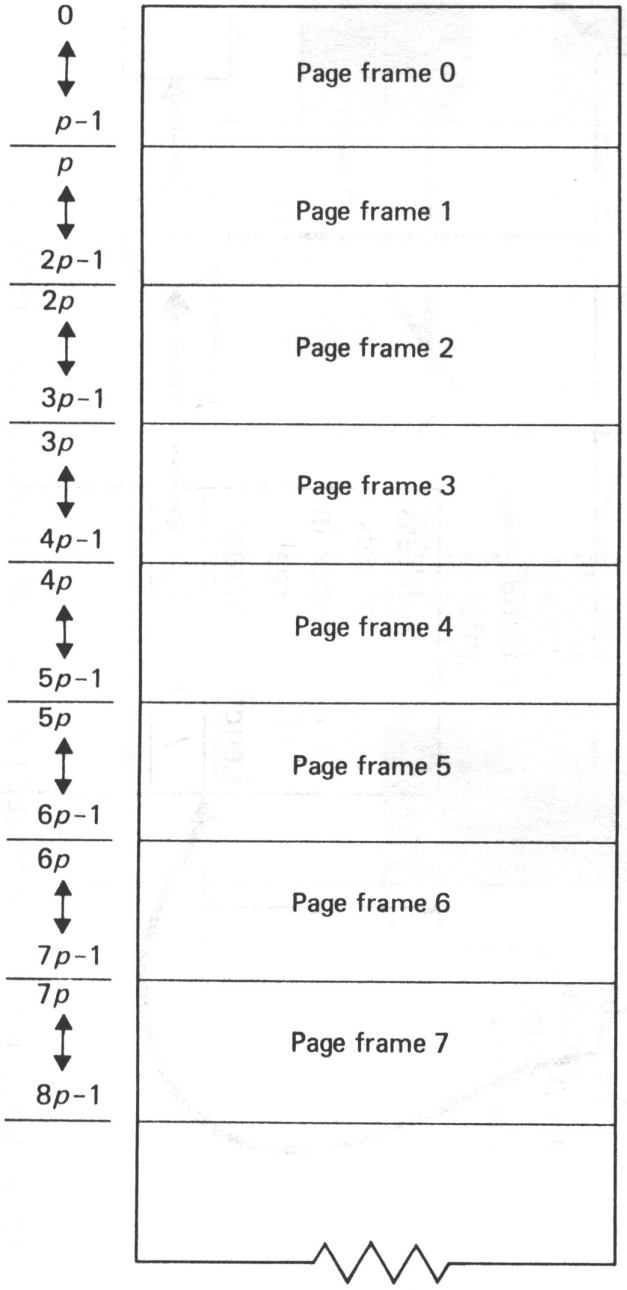

Part 1

RAM Memory Data Structures

Addressing Structures



Effective memory-Access time =
$Prob_1 * T_{tlb} + Prob_2 * T_{pt}$ =
Assume T_{tlb} = 120 and T_{pt} = 220 ns
If TLB 80% then
0.80 * 120 + 0.20 * 220 = 140 ns



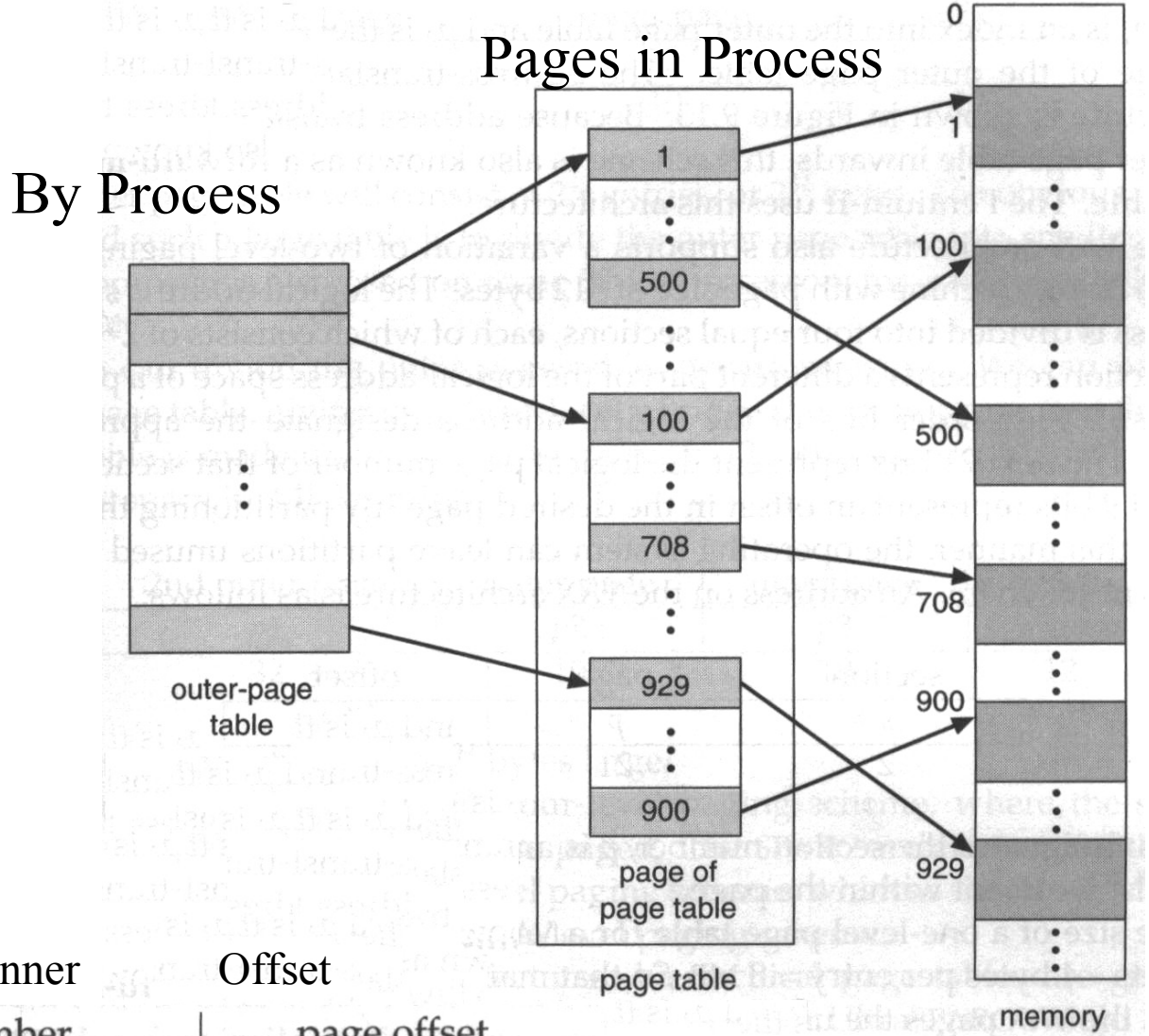
Data Structures

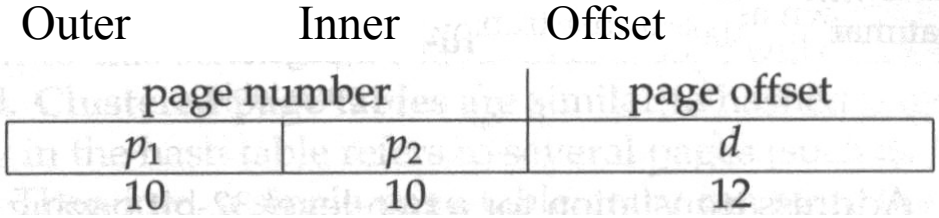
Page frame number	Page frame size	Range of real storage addresses	Valid/Used
0	p	$0 \rightarrow p-1$	1
1	p	$p \rightarrow 2p-1$	0
2	p	$2p \rightarrow 3p-1$	0
3	p	$3p \rightarrow 4p-1$	1
4	p	$4p \rightarrow 5p-1$	1
5	p	$5p \rightarrow 6p-1$	
6	p	$6p \rightarrow 7p-1$	
7	p	$7p \rightarrow 8p-1$	
⋮			

Note:

- To save space boolean stored as bit patterns

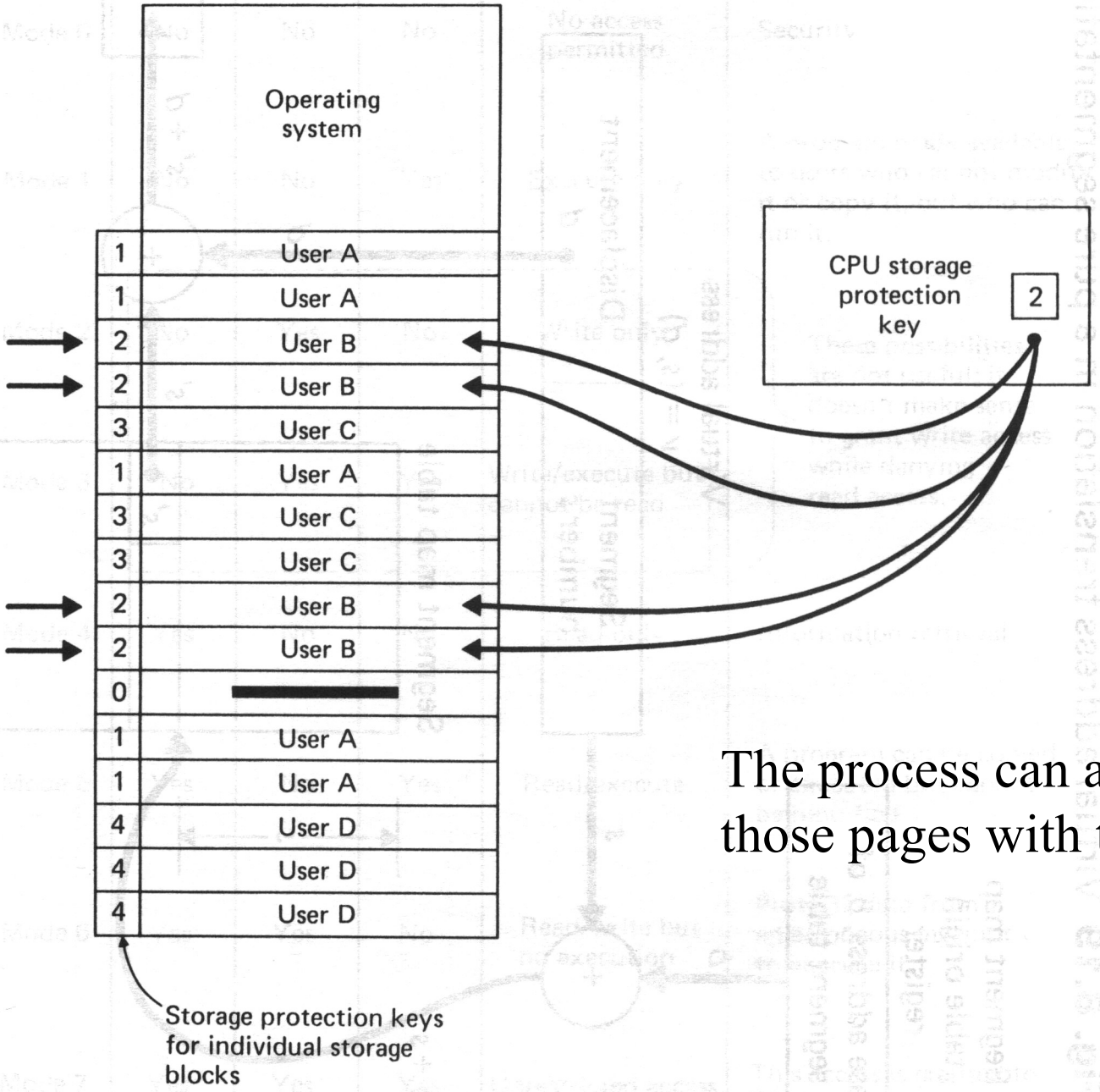
Paging Data Structures





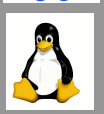



Memory Protection



The process can access only those pages with the same key






Page Table

Index	PID	Valid	Key	Addresses	Used

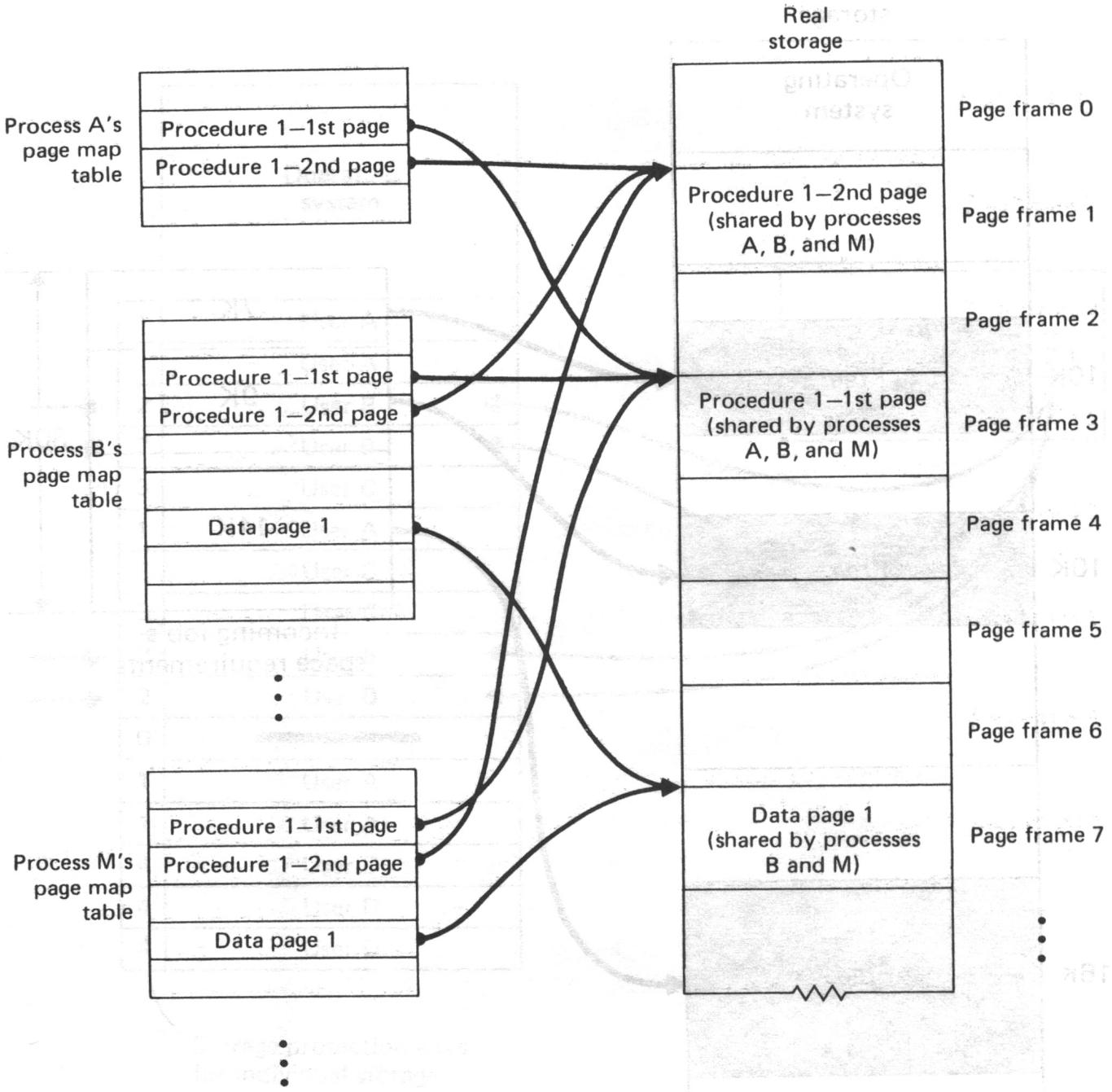
- Index is not stored, it is the index of the array cell
- PID is the owner process number (key can be this number)
- Valid indicates if that page is in memory or on disk (swap in/out)
- Key is the memory protection ID
- Address is the real address (or next level of table)
  - RAM
  - Secondary storage
- Used indicates if that page is assigned to anyone

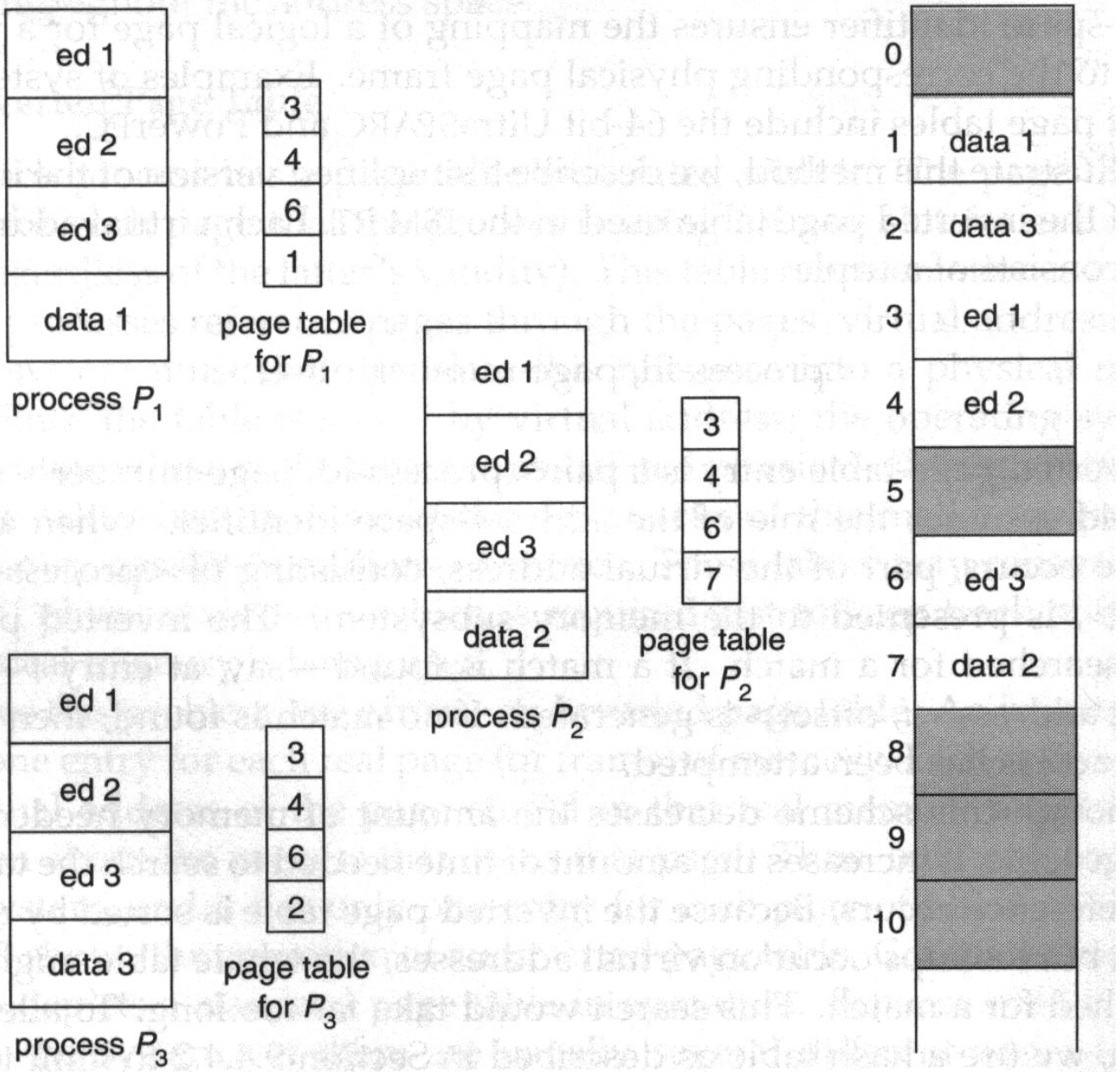



Shared Pages

- Paging systems are able, as is, to assign the same page to more than one process
- *Reentrant code* or *shared data* are pages designated as sharable.
  - *Editors*
  - *Compilers*
  - *Shells*

Shared Pages




ed 1
ed 2
ed 3
data 1
process P_1
3
4
6
1
page table for P_1
ed 1
ed 2
ed 3
data 2
process P_2
3
4
6
7
page table for P_2
ed 1
ed 2
ed 3
data 3
process P_3
3
4
6
2
page table for P_3
0
1 data 1
2 data 3
3 ed 1
4 ed 2
5
6 ed 3
7 data 2
8
9
10




Part 2

Segmentation




Segmentation

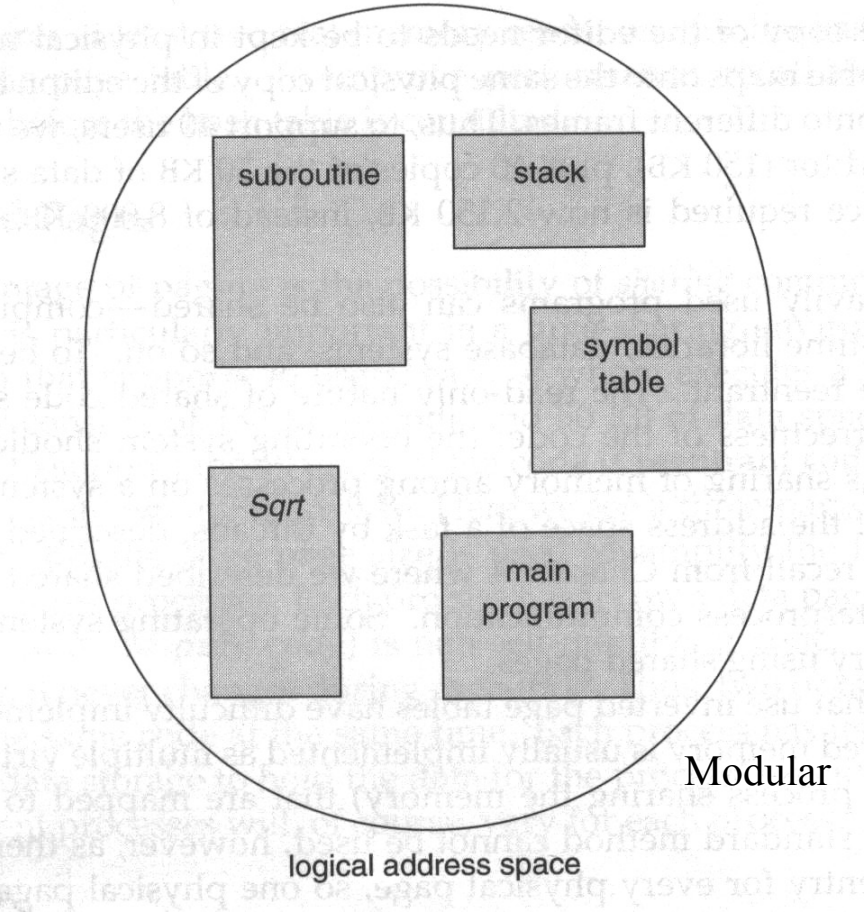
- We would like the user's view of memory to be contiguous for development purposes.
- We would like the OS to have the freedom or reorganizing memory without having to be concerned with how the programmers constructed the code and data.

User's View of Memory

View 1

My code and data contiguous

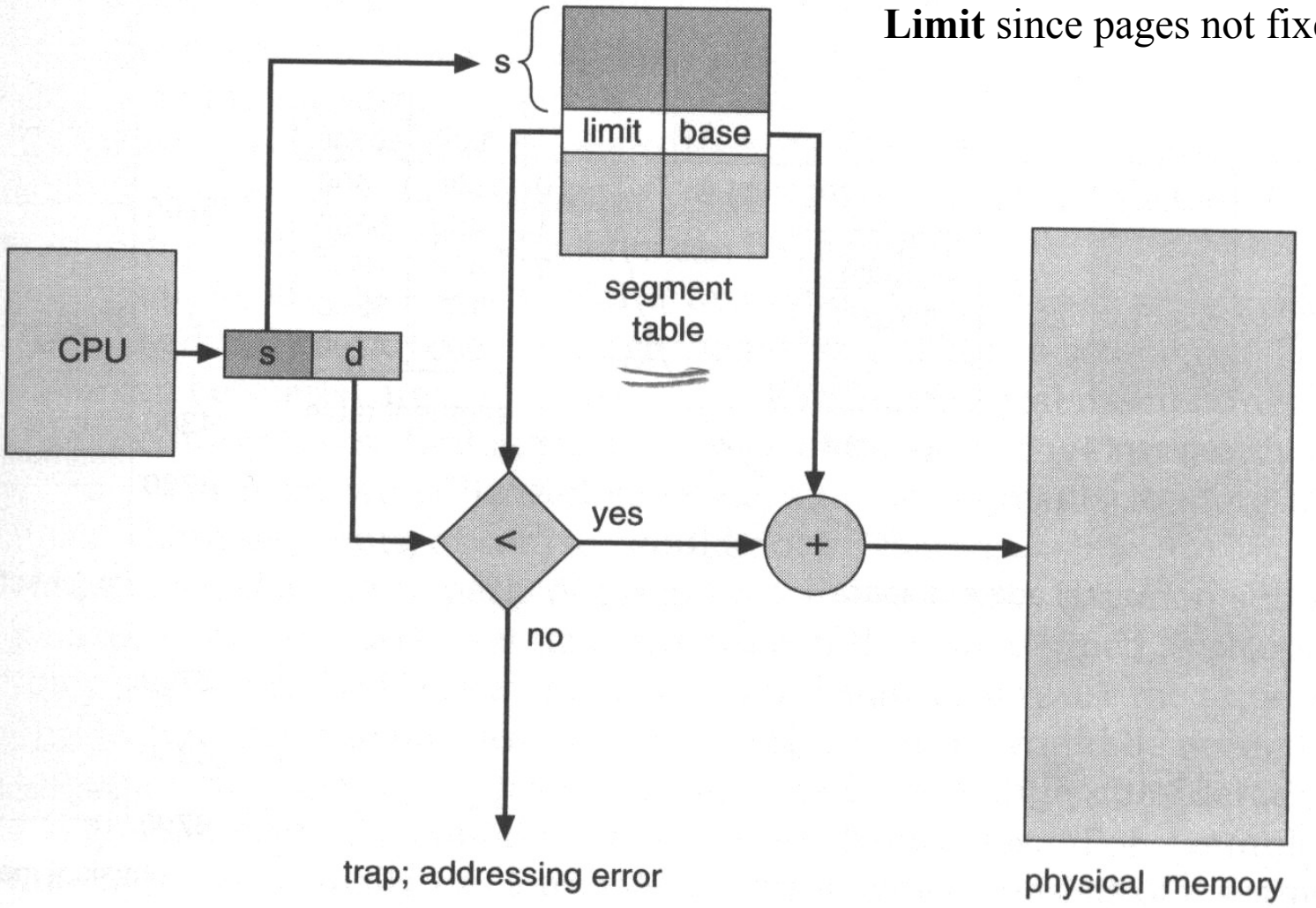
View 2



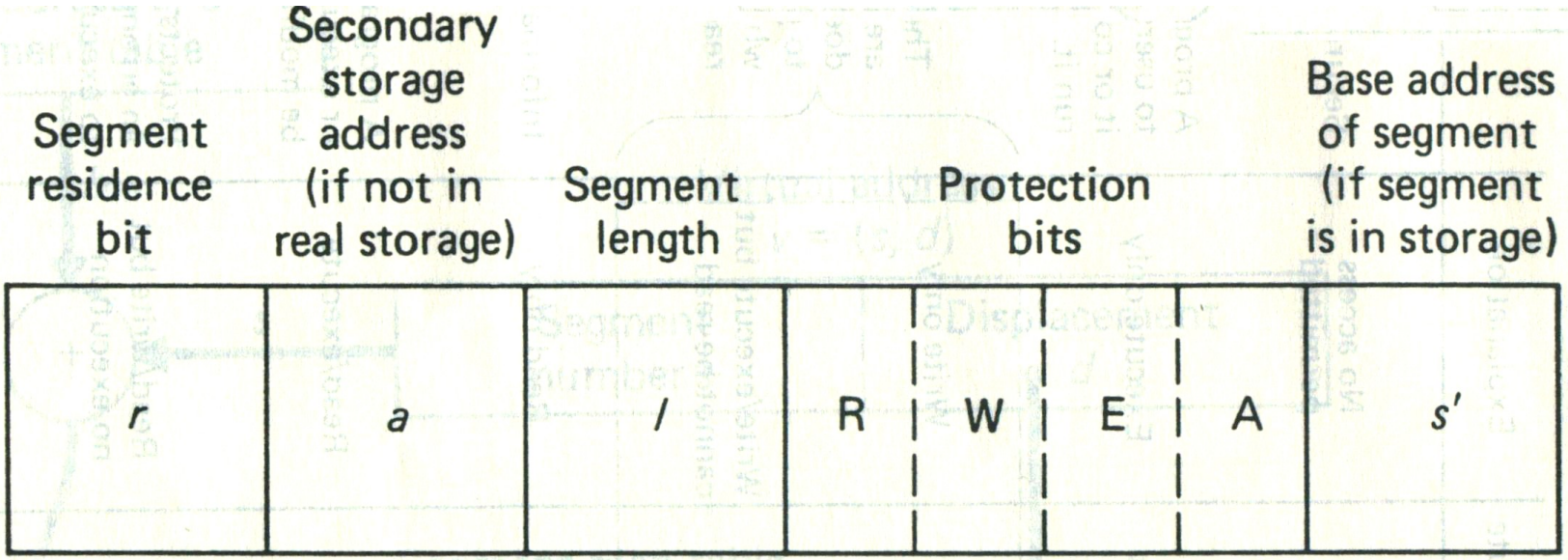
Pages are therefore not fixed in size.

Segmentation Addressing

Limit since pages not fixed size



Segment Table Entry

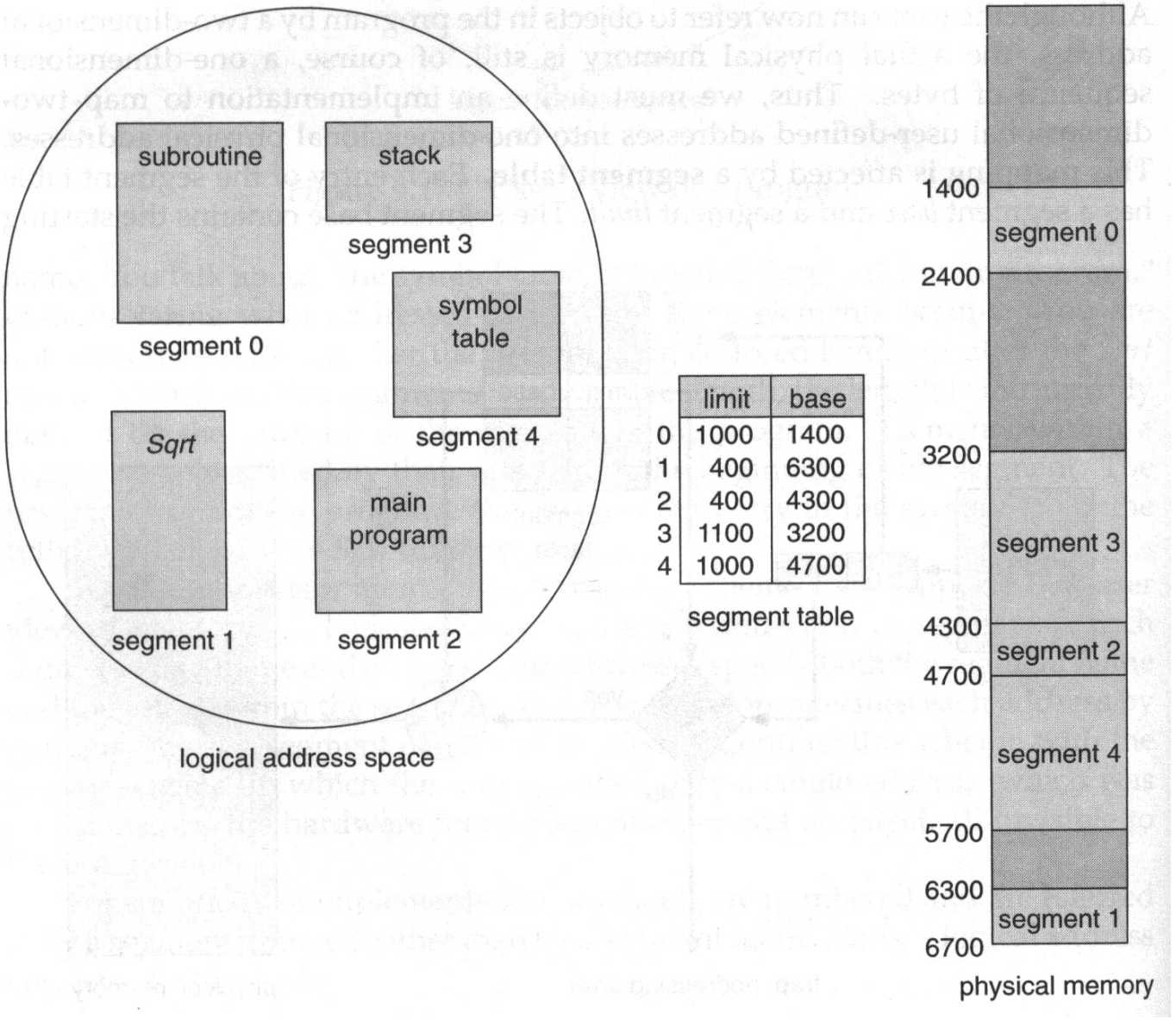


$r = 0$ if segment is not in primary storage
$r = 1$ if segment is in primary storage

Protection bits: (1—yes, 0—no)

R—read access
W—write access
E—execute access
A—append access

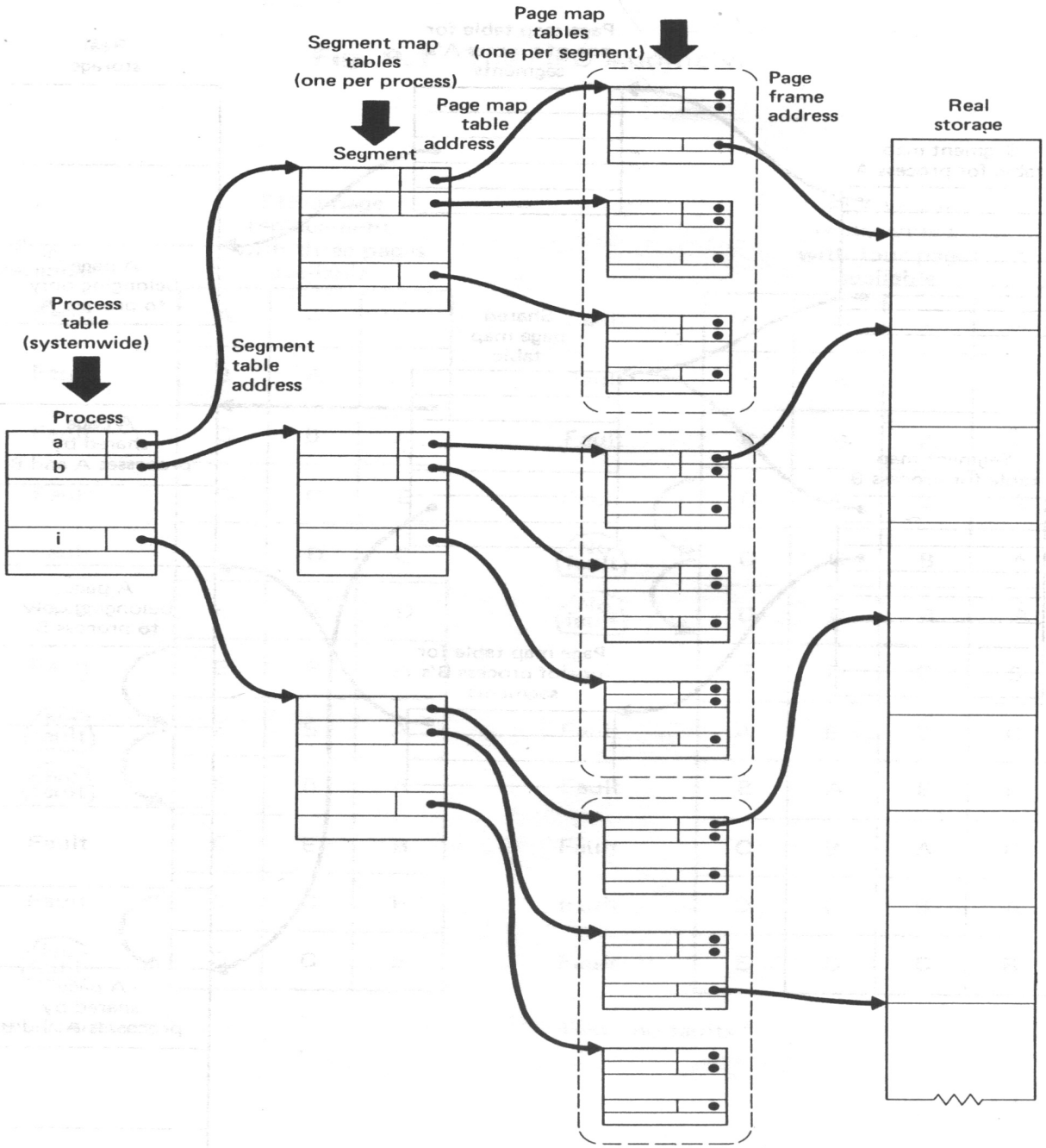
Example Segmentation





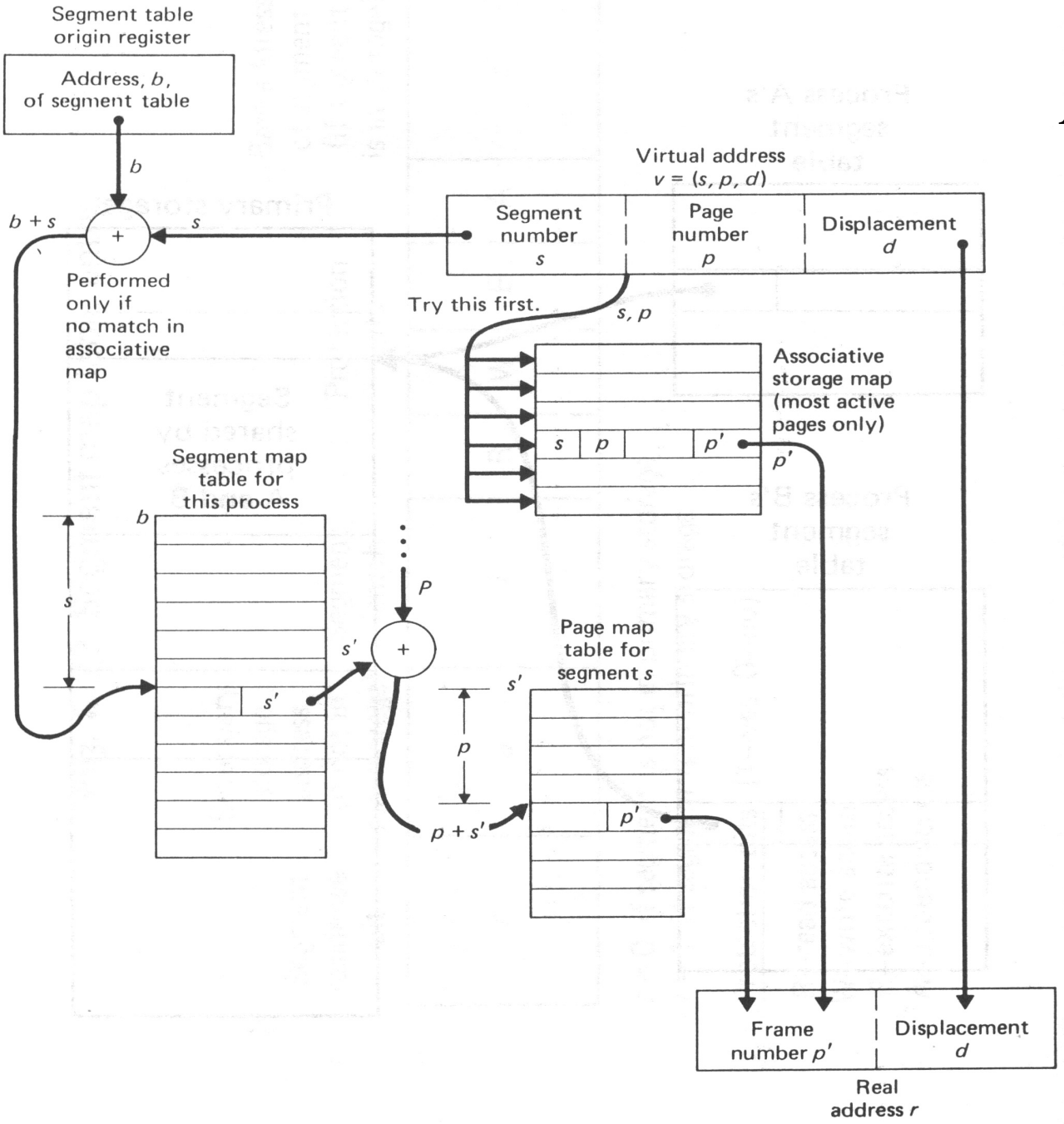

Segmentation Implementation

- Segmentation is good for the user but slows the OS
- Paging is good for the OS but may impose requirements on the user
- Compromise:
  - Implement segmentation by using paging

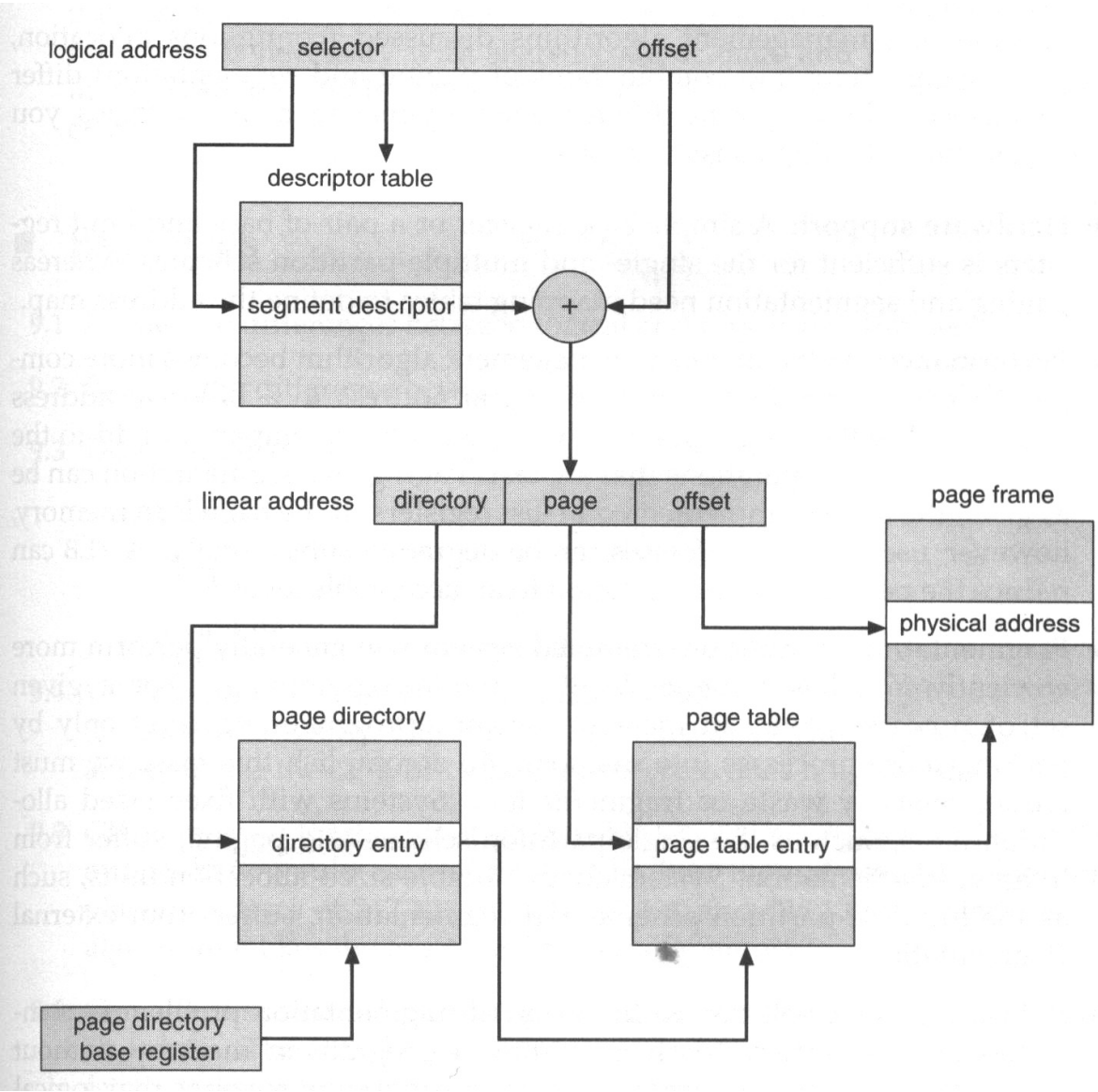


Structure Layout

Addressing Layout



80x86 Case Study (80386, 80586)



Directory = process
(Segmentation)

Paging




Part 3

At Home




Things to try out

1. Does your OS use
   - Paging?
   - Segmentation?
   - Mixed?

   Try to find this out either from your OS documentation or the internet.