


Comp 310
Computer Systems & Organization

Lecture #13

Memory Management

(The Basics about RAM Management)

Prof. Joseph Vybihal




Announcements

- No class Thursday




Midterm Exam Format

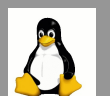
- Four questions
  - Definition questions (1 to 2)
  - Analyze and fix (0 to 2)
  - Analyze and describe (1 to 2)
  - Pseudo-code (0 to 1)
  - Actual C code (0)
- Material on Exam
  - Material including this lecture
  - Historical and current OS architectures
  - The components of a modern OS
  - Implementation of the User Interface
  - Executing multiple processes & OS run-time states
  - Inter-process communication and synchronization problems
  - Thread implementations
  - OS Scheduling problems
  - Deadlock graphs, semaphore problems and algorithms

Architectures

User Interface

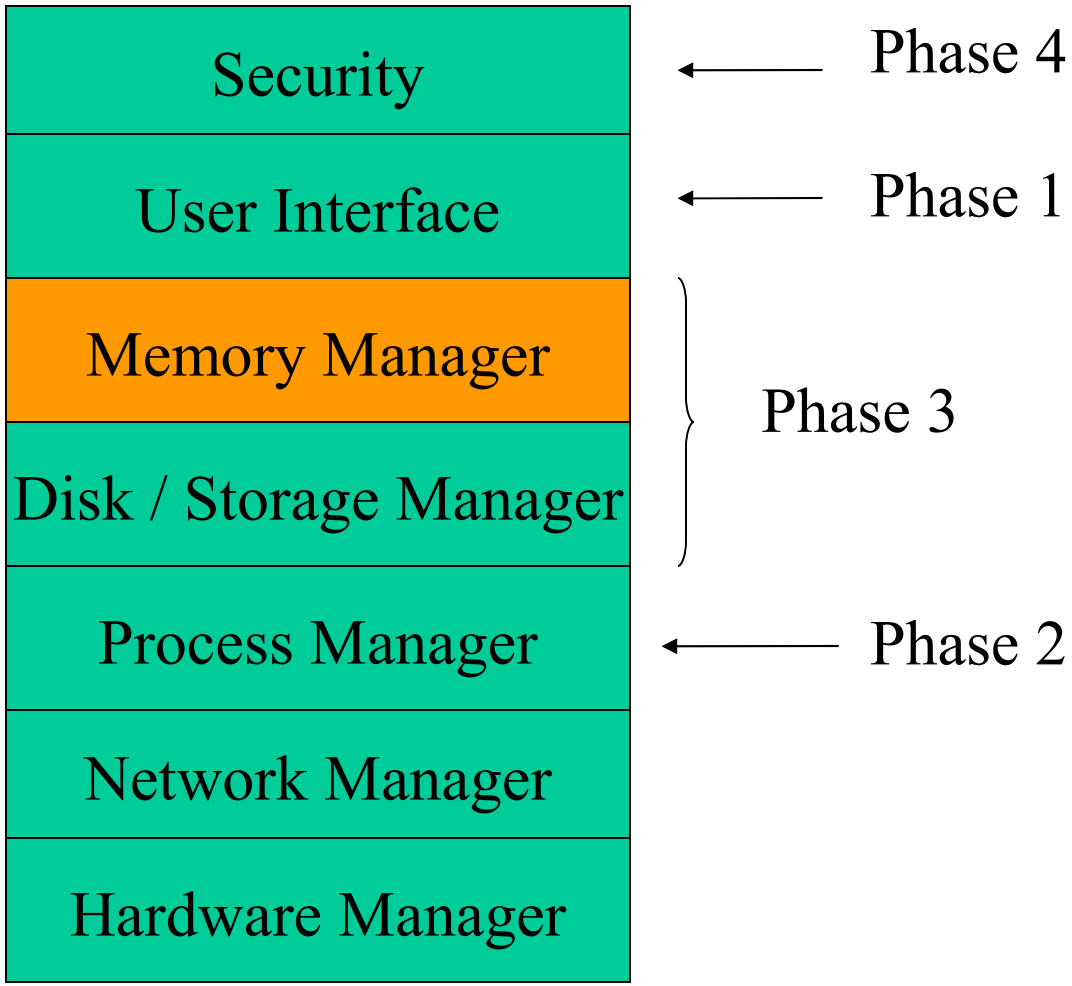
Process Management






Basic OS Architecture

(Course Table of Contents)

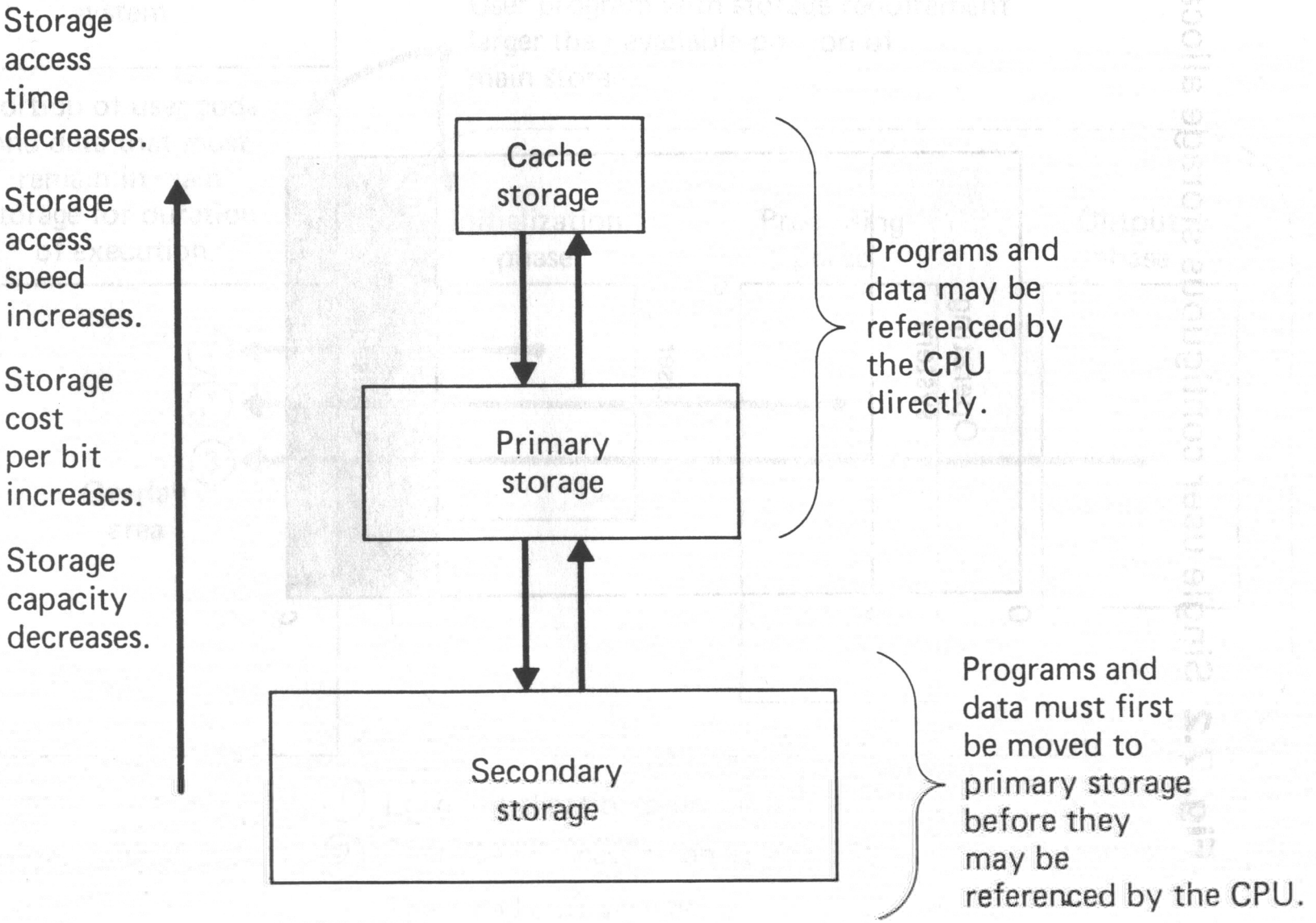





Part 1

The Programmer's Point of View

Memory Hierarchy

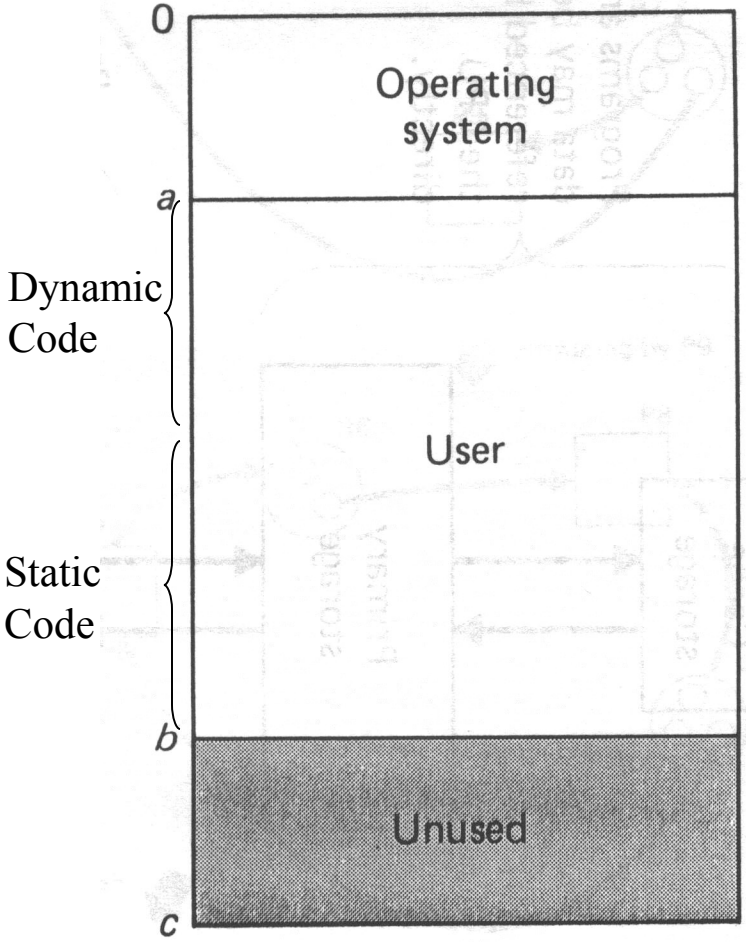





Relative Time to Get Data From Storage

Type of Storage	Access Speed	Relative Slower
Register	2 clock cycles (cs)	
L1 (on CPU) cache	4 cs	2x
L2 (separate chip) cache (SRAM)	6 – 20 cs	3 – 10x
Memory (DRAM)	50 cs	25x
Magnetic/Hard Disk	2,000 cs	1,000x
Optical Disk	6,000 cs	3,000x

Private Run-Time Environment



Sharable OS components
- File access
- Network access
- Inter-process communication

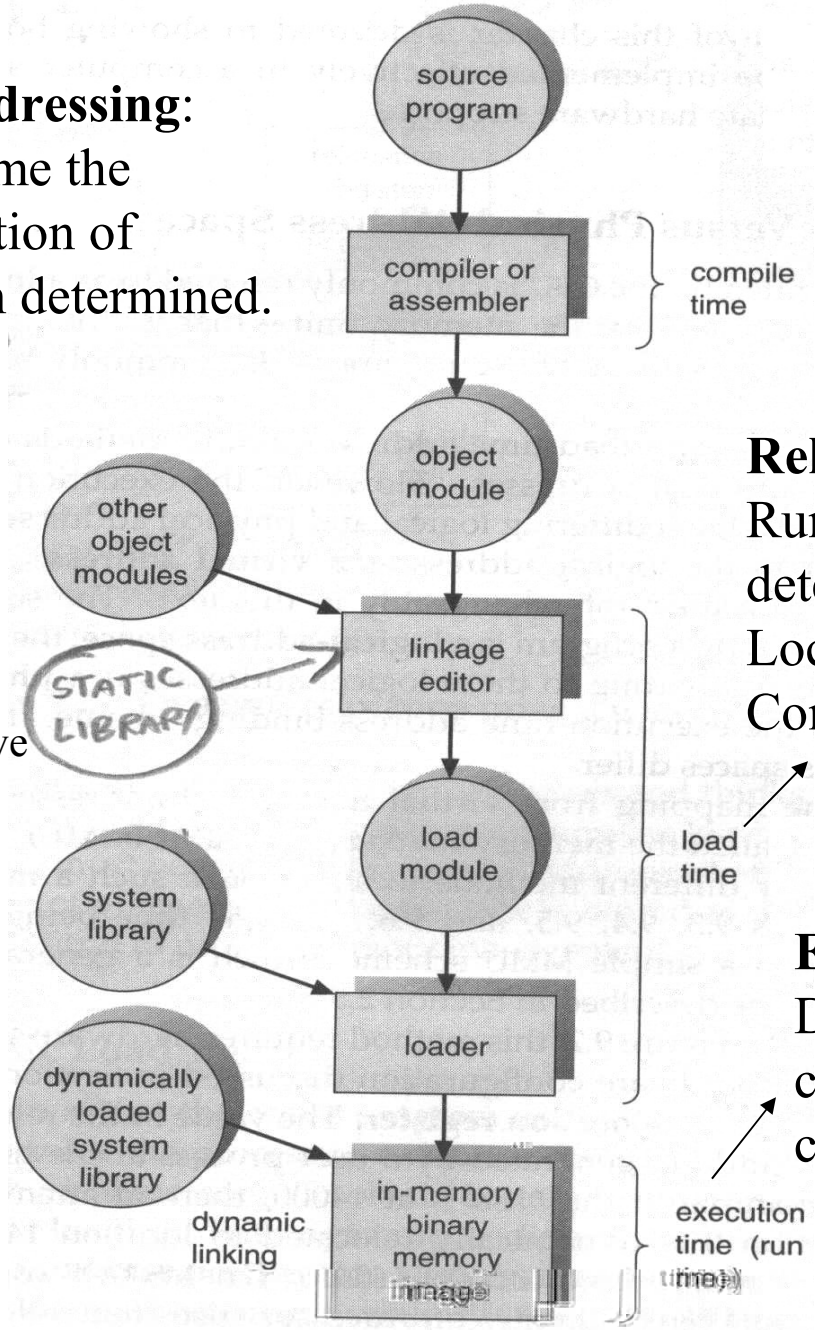
Must learn OS function calls.

"No one can tamper with user's code"

Space assigned by OS

Memory Addressing

Absolute Addressing:
At compile time the run-time location of code has been determined.



Becomes part of code on disk. Each program would have it.

DLLs, only loaded at run-time by OS. Shareable.

Relocatable Addressing:
Run-time location of code determined when launched.
Loc = LOAD + Offset
Compiled using offsets.

Execution Addressing:
During execution the code can be relocated (moving code at run-time)

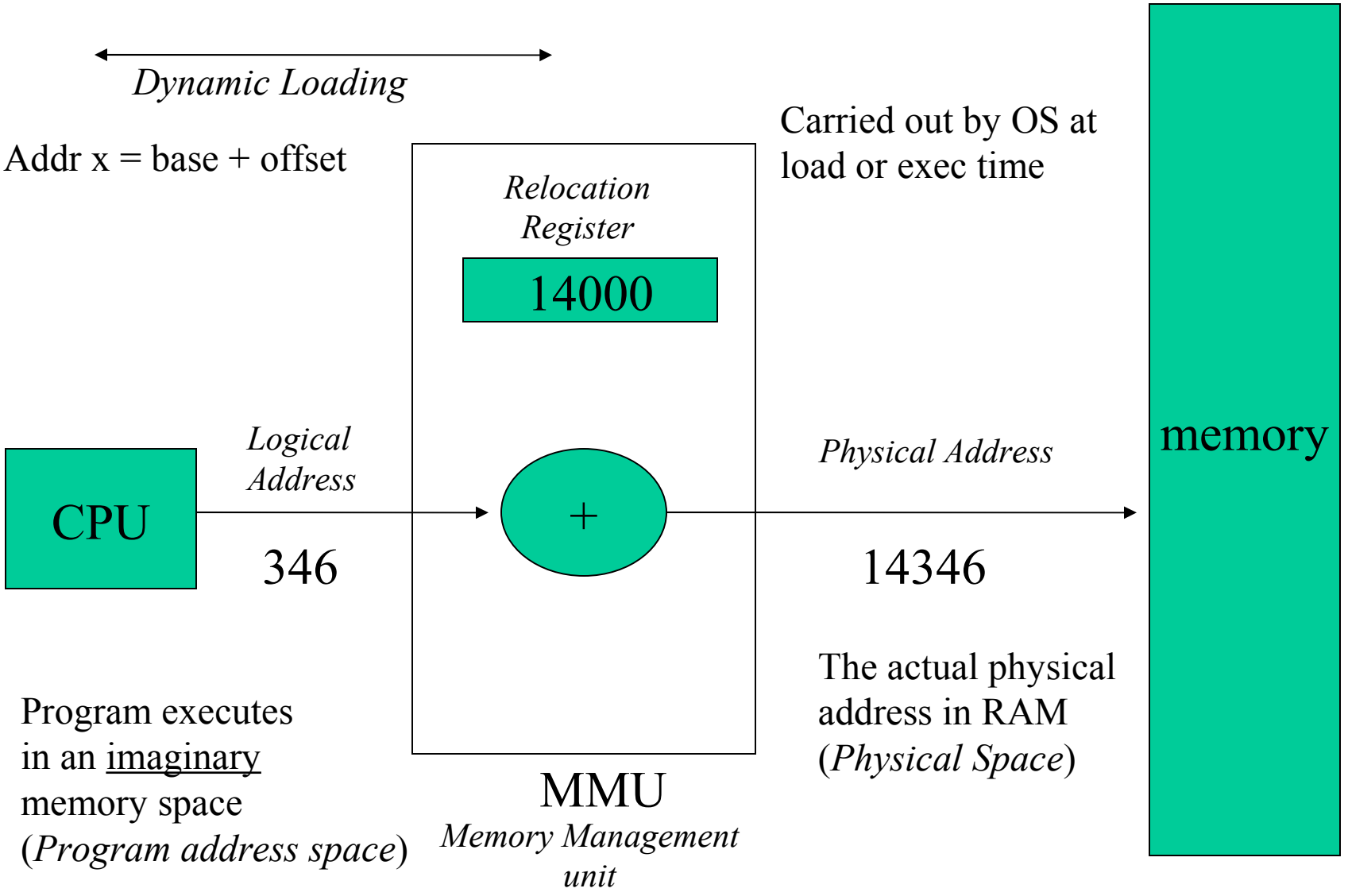



Part 2

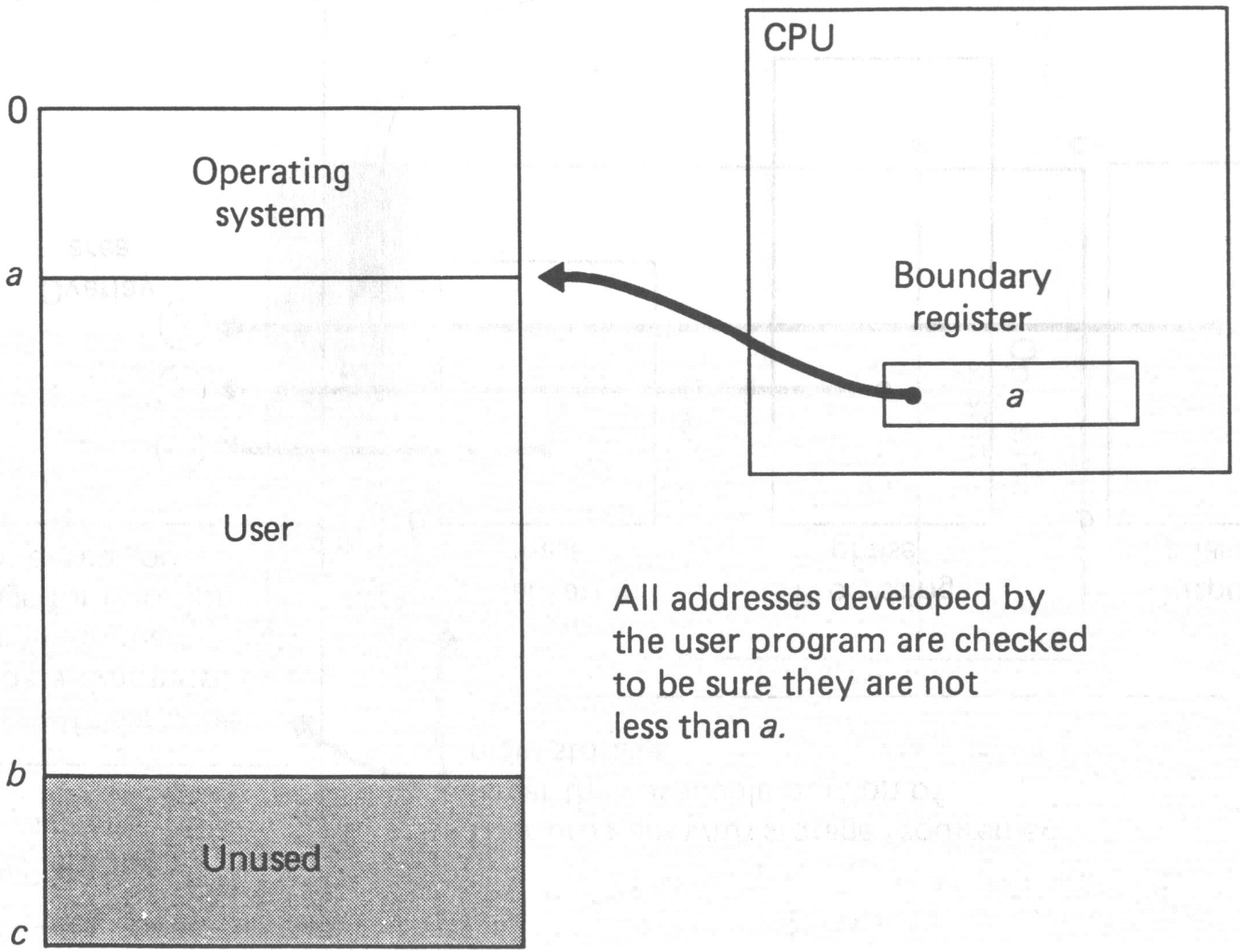
Supporting Memory Hardware

(If not provided then OS cannot do it)

Dynamic Address Relocation

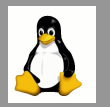


Memory Protection

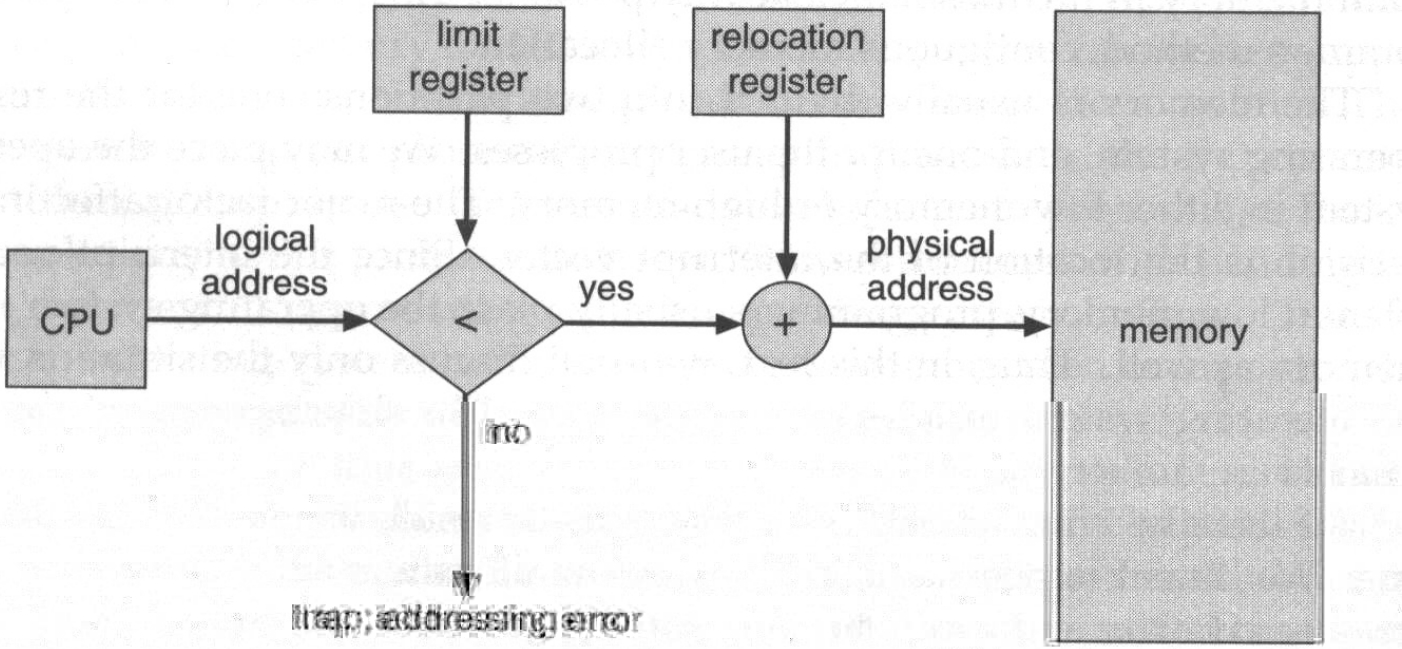


Protecting the OS from a user process…

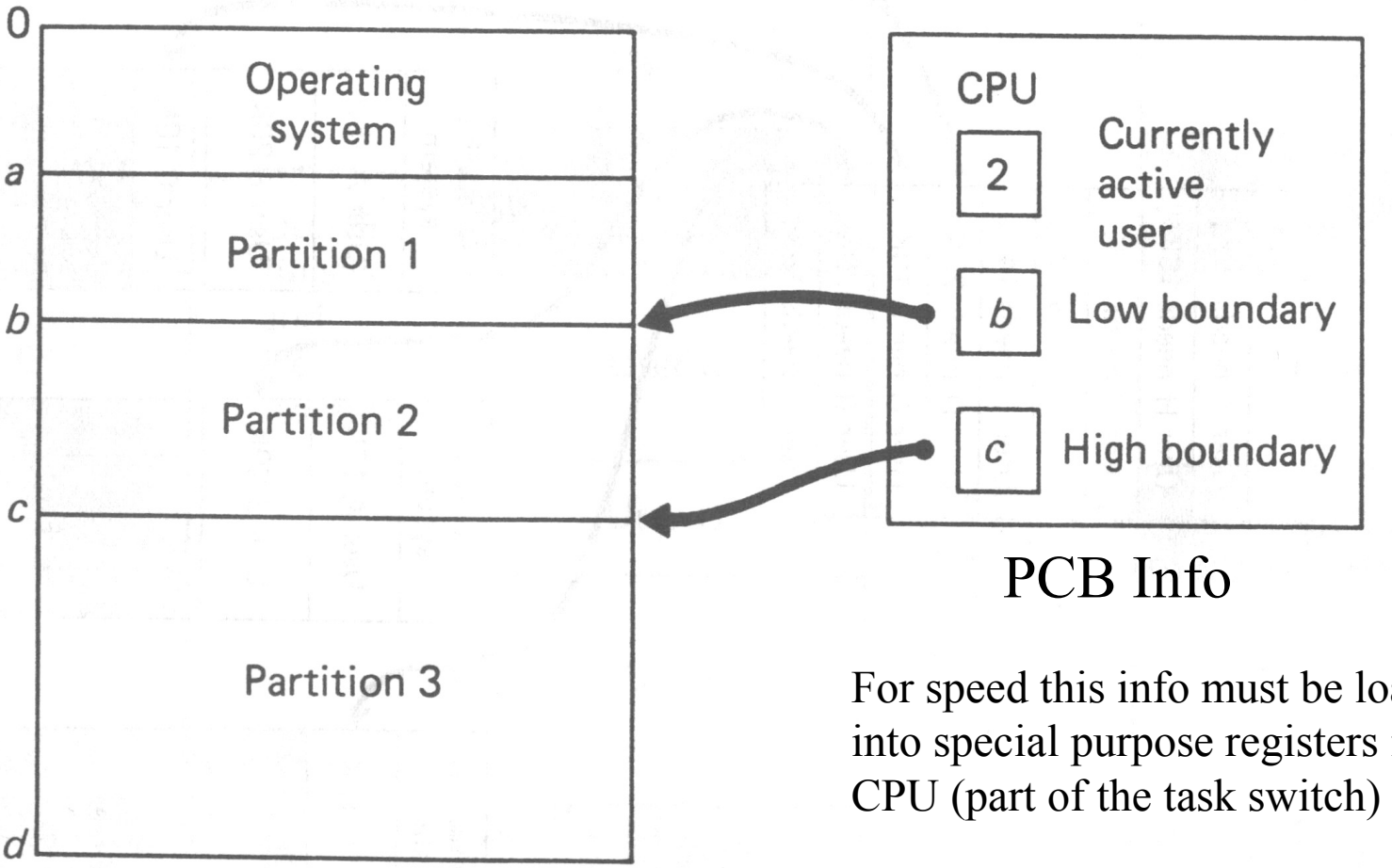





A hardware implementation



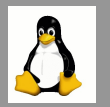
Protection Between Processes



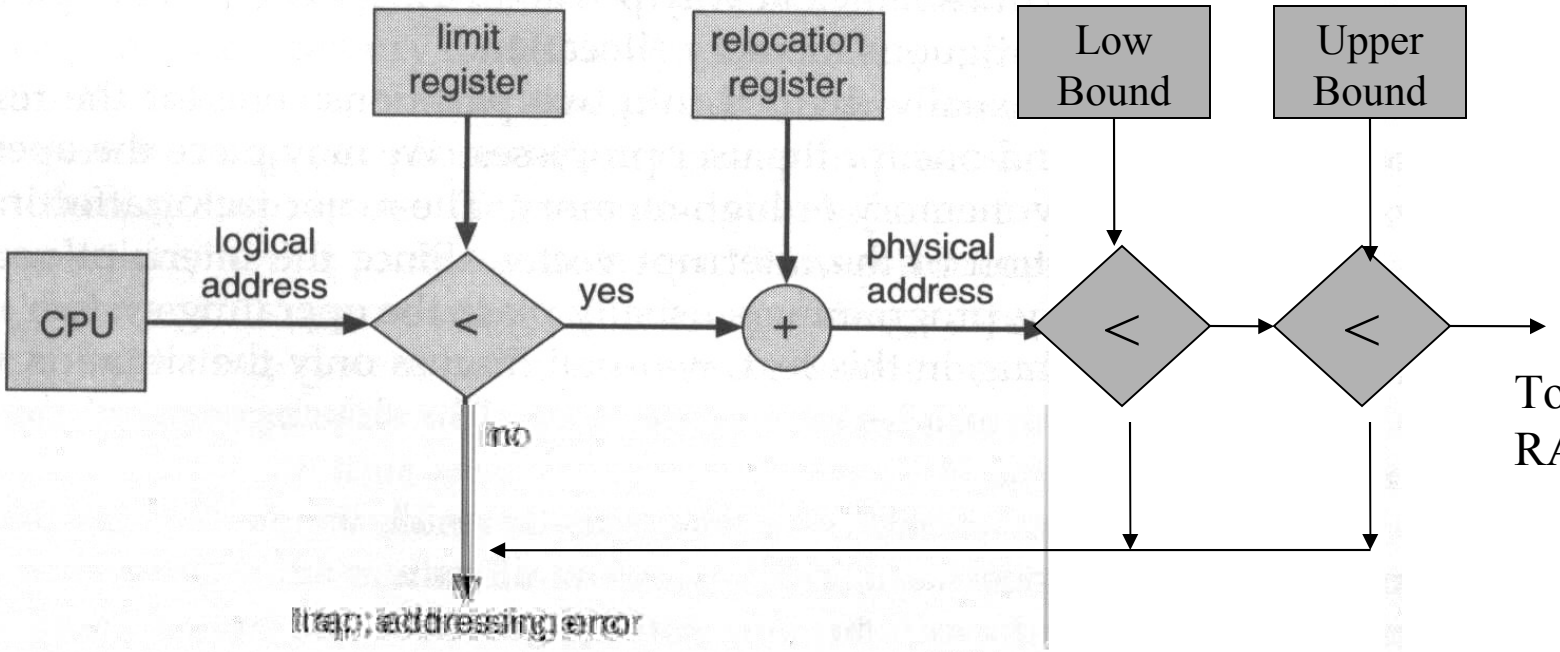
PCB Info

For speed this info must be loaded into special purpose registers in the CPU (part of the task switch)






Delays Processing (2 cs * 4 per instruction minimum)






Part 3

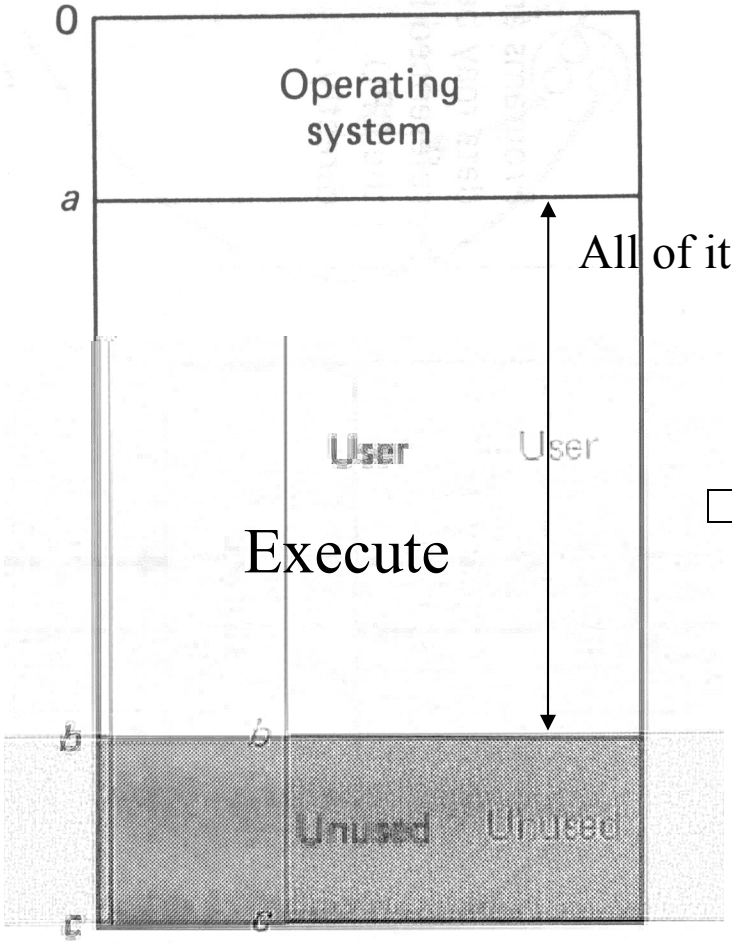
Introductory Memory Organization Techniques




Fixed Memory

No Organization (min)

Actual or illusion: user is the only process executing on system



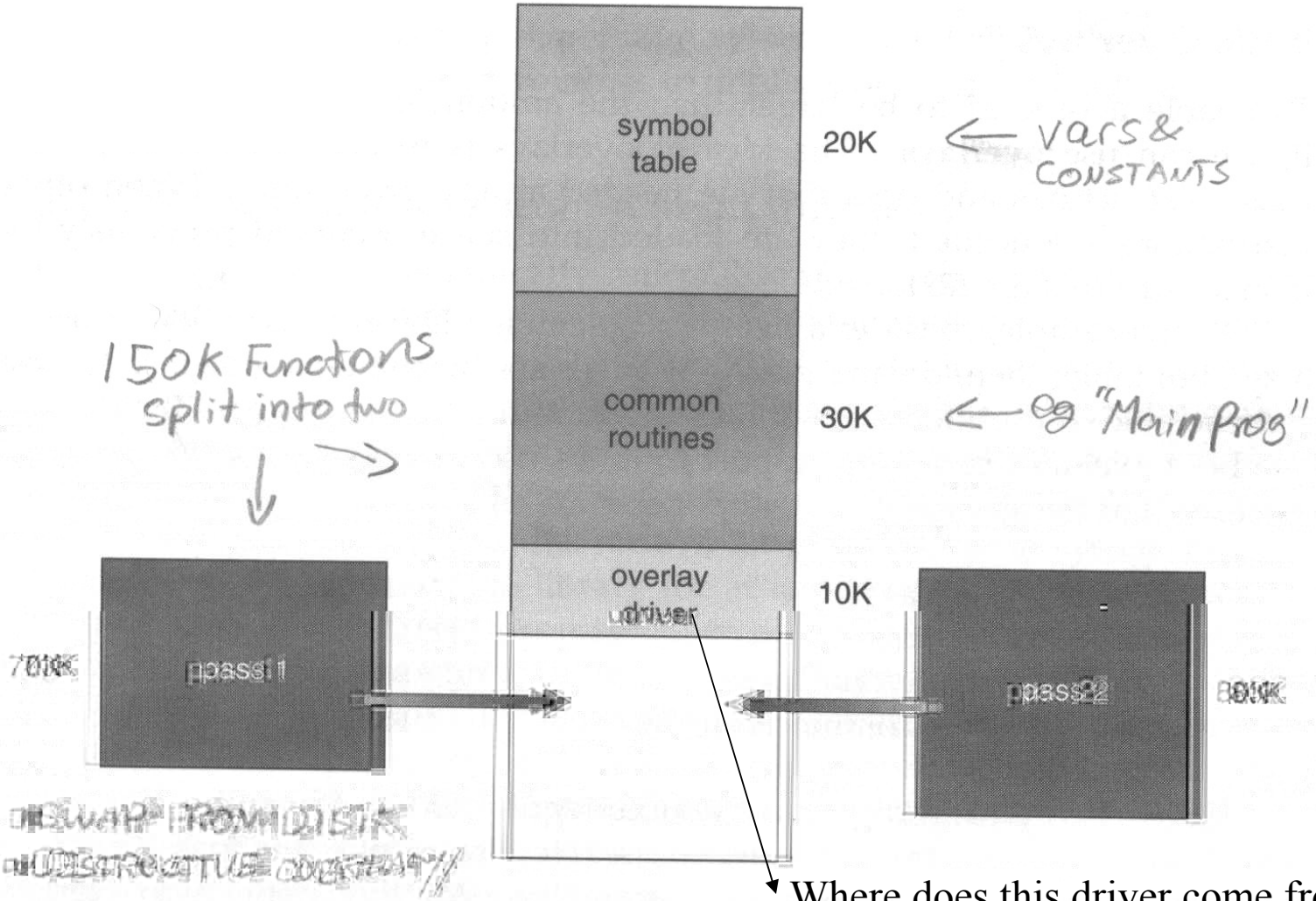
All of it fits in memory, fixed size.

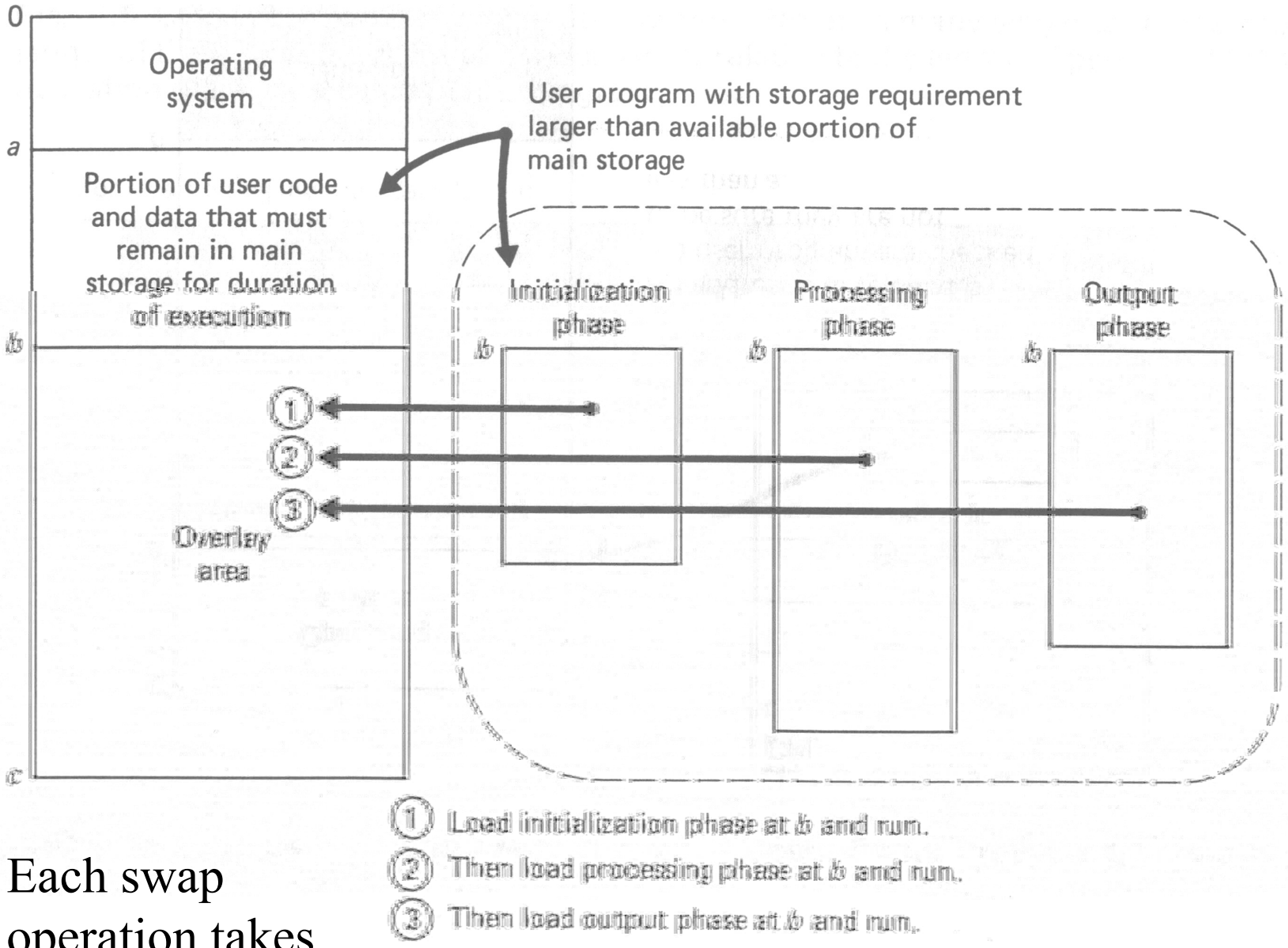
Load

Execute

Terminate

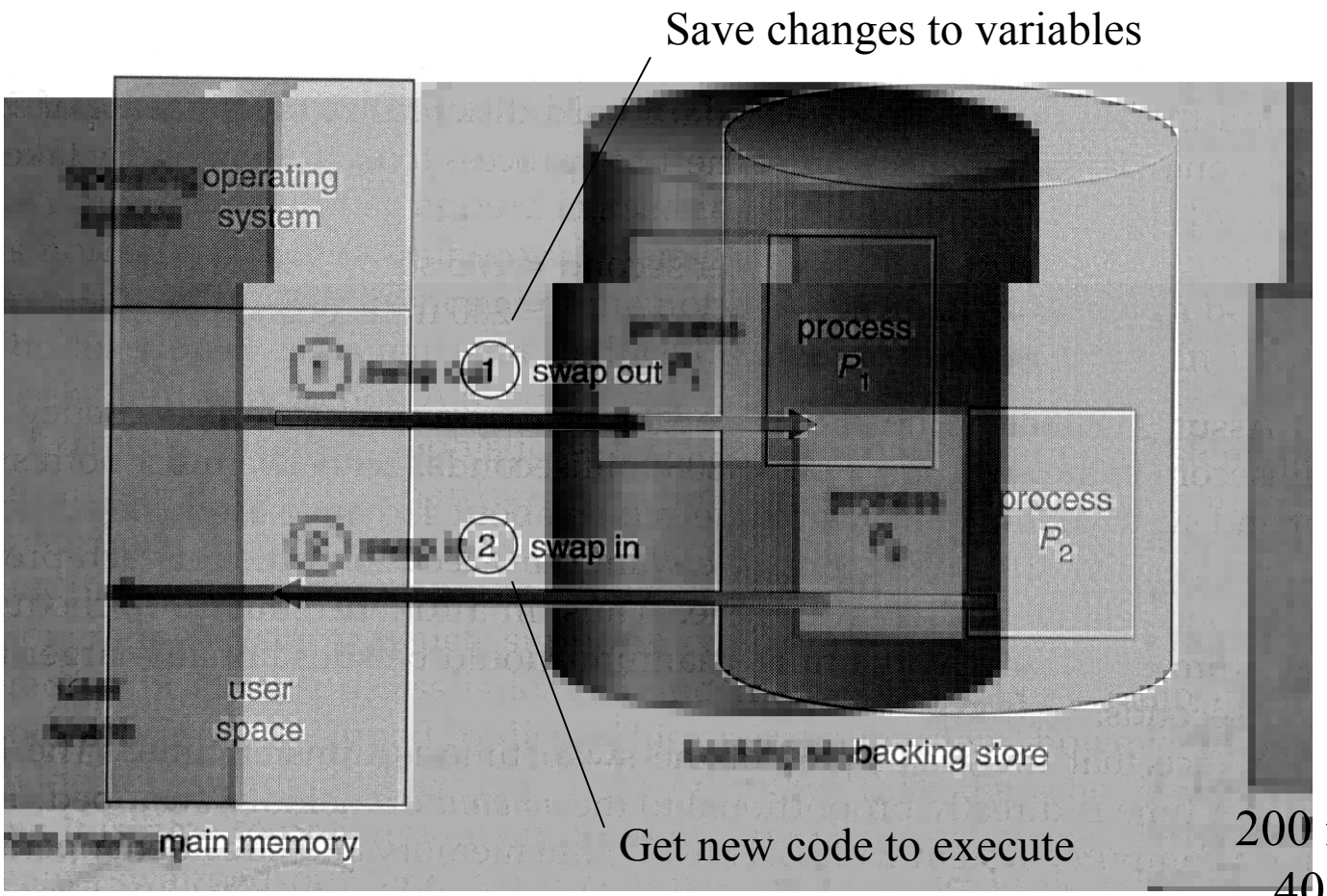
Overlays


symbol table
20K
vars & CONSTANTS
150K Functions split into two
common routines
30K
eg "Main Prog"
overlay driver
10K
pass1
pass2
Where does this driver come from?
• Programmer, Compiler, or O/S



Each swap operation takes time

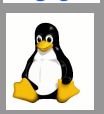
Swapping Processes



200 ms * 2 = 400 ms ~ ½ sec

Swap time = bytes / bytes per sec
= 1000 KB / 5000 KB = 200 milliseconds (for 1 swap)

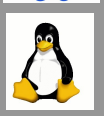





Questions

- When should an OS use swapping?
  - When out of memory
  - When program is bigger than memory
- Is it worth doing?
  - How expensive is a swap?
- Rhetorical questions maybe...
  - How should swapping be implemented?
  - How should it be managed on disk?






Swapping Issues

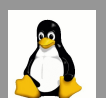
- Swap all or part of process?
  - All is expensive
  - If some, then which part?
    - What if the part we swapped out is the next one we need?
- Is it waiting for I/O?
  - Synchronous transfer
    - Easy to know about
  - Asynchronous transfer
    - It happens as a background activity by device




Part 3

At Home






Things to try out

1. If you have an advanced compiler like Visual C++ or C++ Builder…
   - Load a large program you have written
   - Go to the compilation options section and play with the values and run your program after each change. Try the following:
     - Change stack & heap sizes
     - Turn on/off page swapping
     - Force entire program in RAM
2. Go to the OS Control Panel and make changes to the memory manager (be careful – try this on a machine that is not too important – like your Dad's…)!