University of Pittsburgh

Introduction to Operating Systems
CS 1550

Spring 2023

Sherif Khattab

ksm73@pitt.edu

(Some slides are from **Silberschatz, Galvin and Gagne ©2013)**

Announcements

- Upcoming deadlines
  - Homework 8 is due **this Friday**
  - Quiz 1 and Lab 2 due on Tuesday 2/28 at 11:59 pm
  - Project 2 is due Friday 3/17 at 11:59 pm

In an ideal world…

- The ideal world has memory that is
  - Very large
  - Very fast
  - Non-volatile (doesn't go away when power is turned off)
- The real world has memory that is:
  - Very large
  - Very fast
  - Affordable!

  ⇒Pick any two…
- Memory management goal: make the real world look as much like the ideal world as possible

Memory hierarchy

- What is the memory hierarchy?
  - Different levels of memory
  - Some are small & fast
  - Others are large & slow
- What levels are usually included?
  - Cache: small amount of fast, expensive memory
    - L1 (level 1) cache: usually on the CPU chip
    - L2 & L3 cache: off-chip, made of SRAM
  - Main memory: medium-speed, medium price memory (DRAM)
  - Disk: many gigabytes of slow, cheap, non-volatile storage
- Memory manager handles the memory hierarchy

Problem of the Day

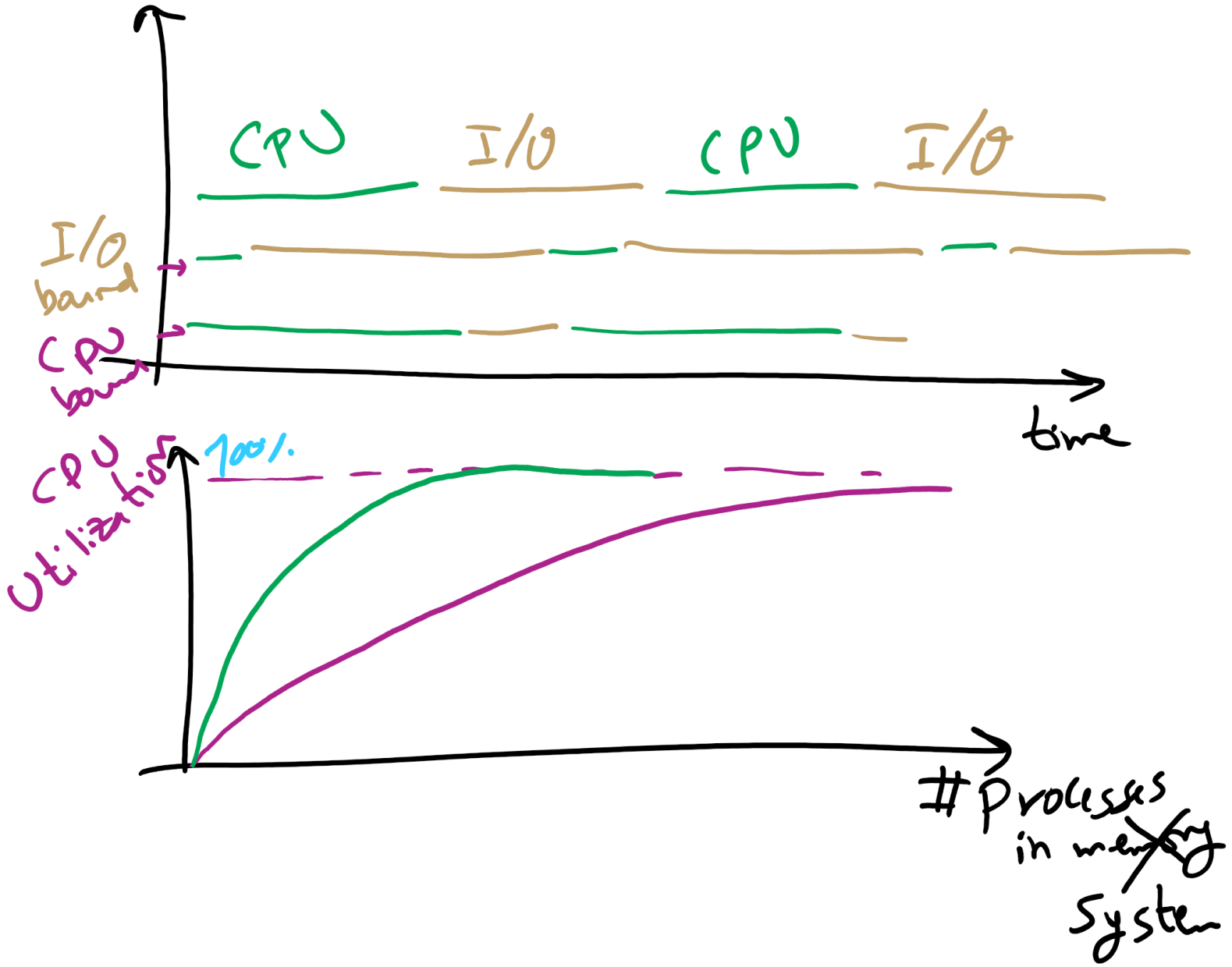
How can we share computer's memory between multiple processes?

How can we protect each process's memory partition from other processes?

How many programs is enough?

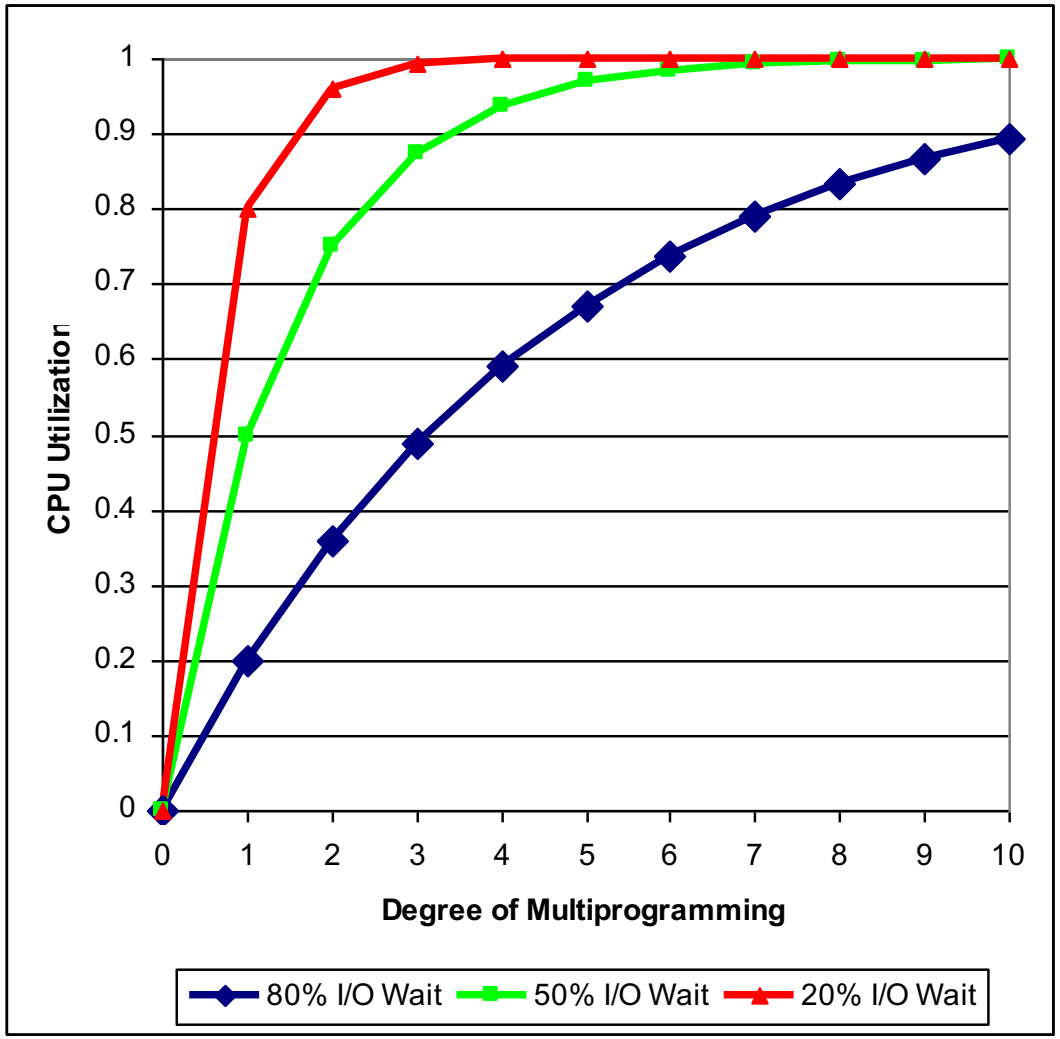
- Several memory partitions (fixed or variable size)
- Lots of processes wanting to use the CPU
- Tradeoff
  - More processes utilize the CPU better
  - Fewer processes use less memory (cheaper!)
- How many processes do we need to keep the CPU fully utilized?
  - This will help determine how much memory we need
  - Is this still relevant with memory costing $10/GB?

Why do we need more processes?



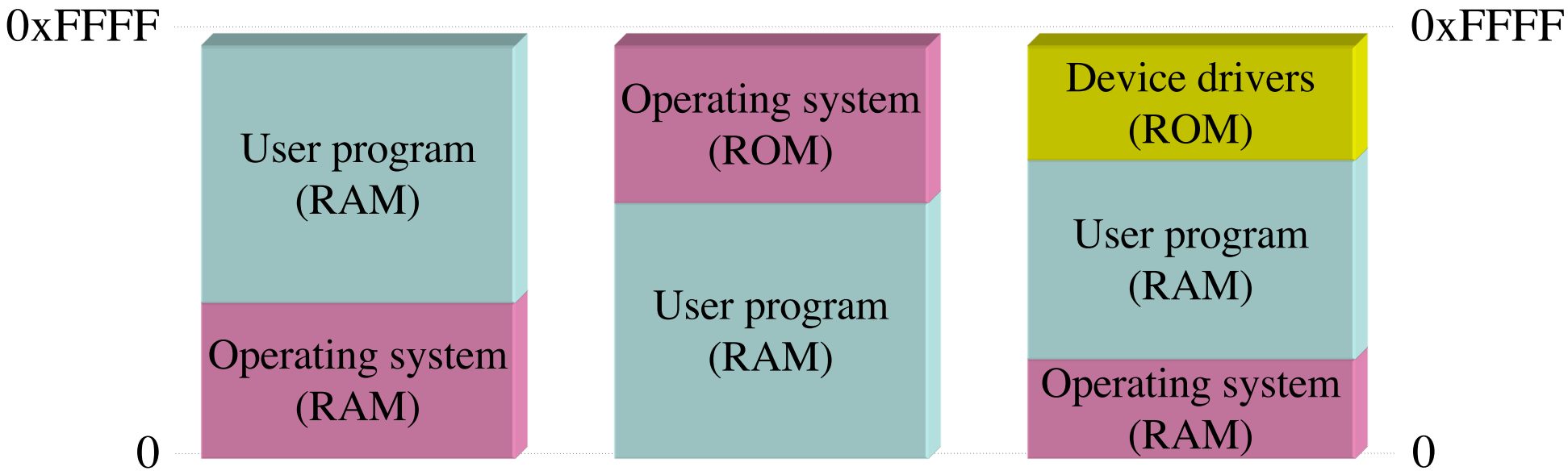
Modeling multiprogramming

- More I/O wait means less processor utilization
  - At 20% I/O wait, 3–4 processes fully utilize CPU
  - At 80% I/O wait, even 10 processes aren't enough
- This means that the OS should have more processes if they're I/O bound
- More processes => memory management & protection more important!

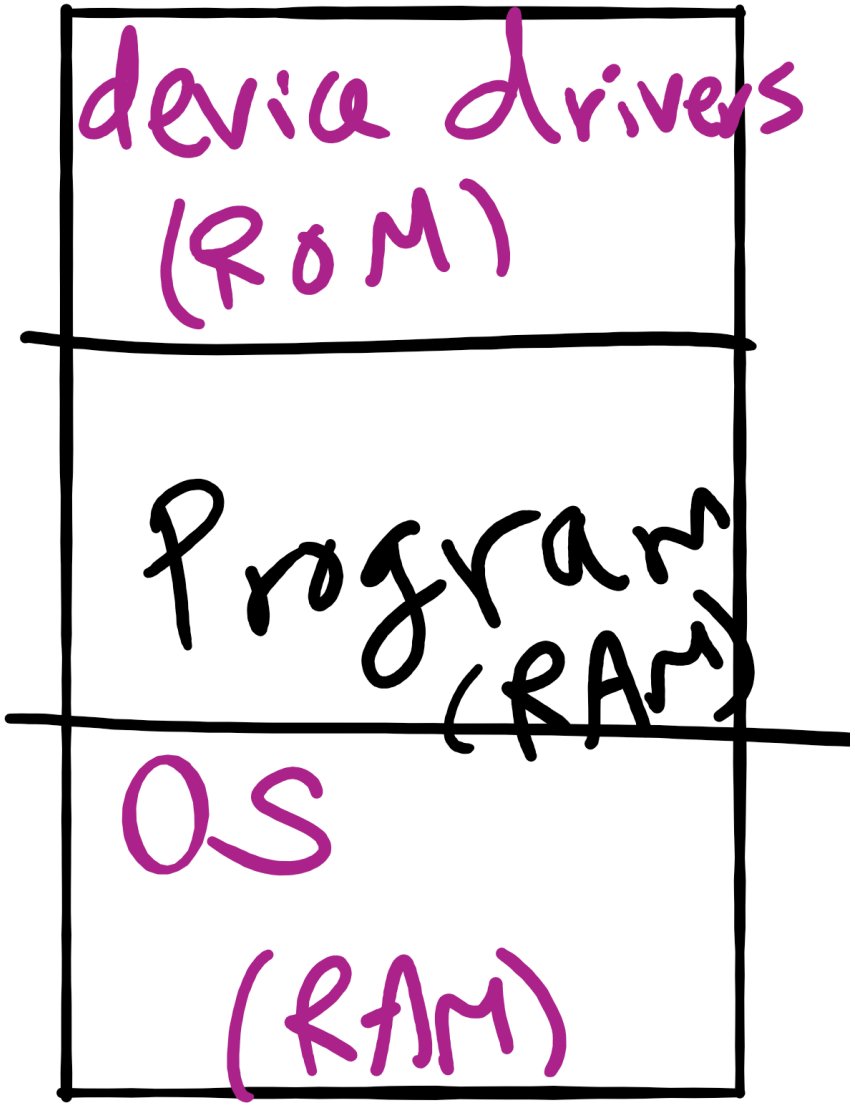


Basic memory management

- Components include
  - Operating system (perhaps with device drivers)
  - Single process
- Goal: lay these out in memory
  - Memory protection may not be an issue (only one program)
  - Flexibility may still be useful (allow OS changes, etc.)
- No swapping or paging

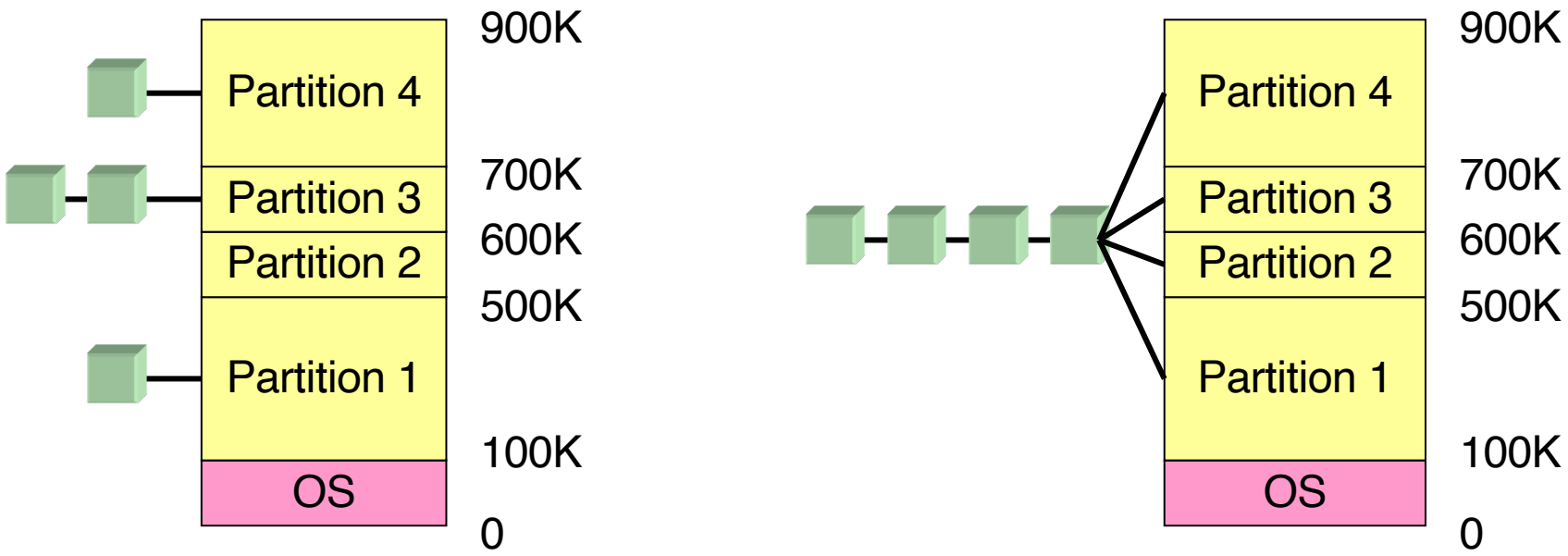


Memory Management for Embedded Systems



Fixed partitions: multiple programs

- Fixed memory partitions
  - Divide memory into fixed spaces
  - Assign a process to a space when it's free
- Mechanisms
  - Separate input queues for each partition
  - Single input queue: better ability to optimize CPU usage



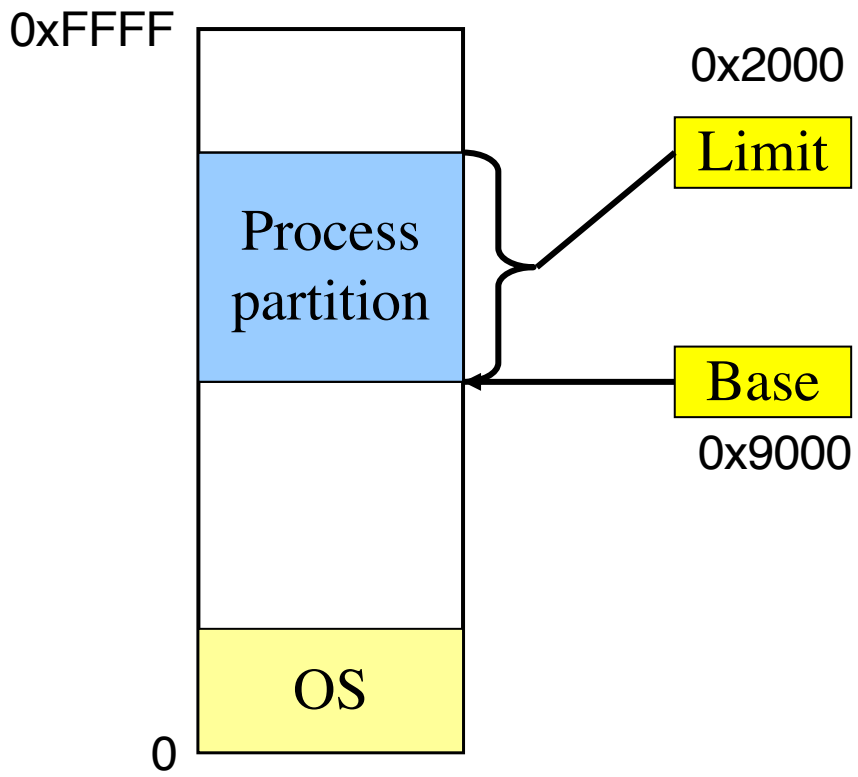
Problem of the Day

How can we share computer's memory between multiple processes?

How can we protect each process's memory partition from other processes?

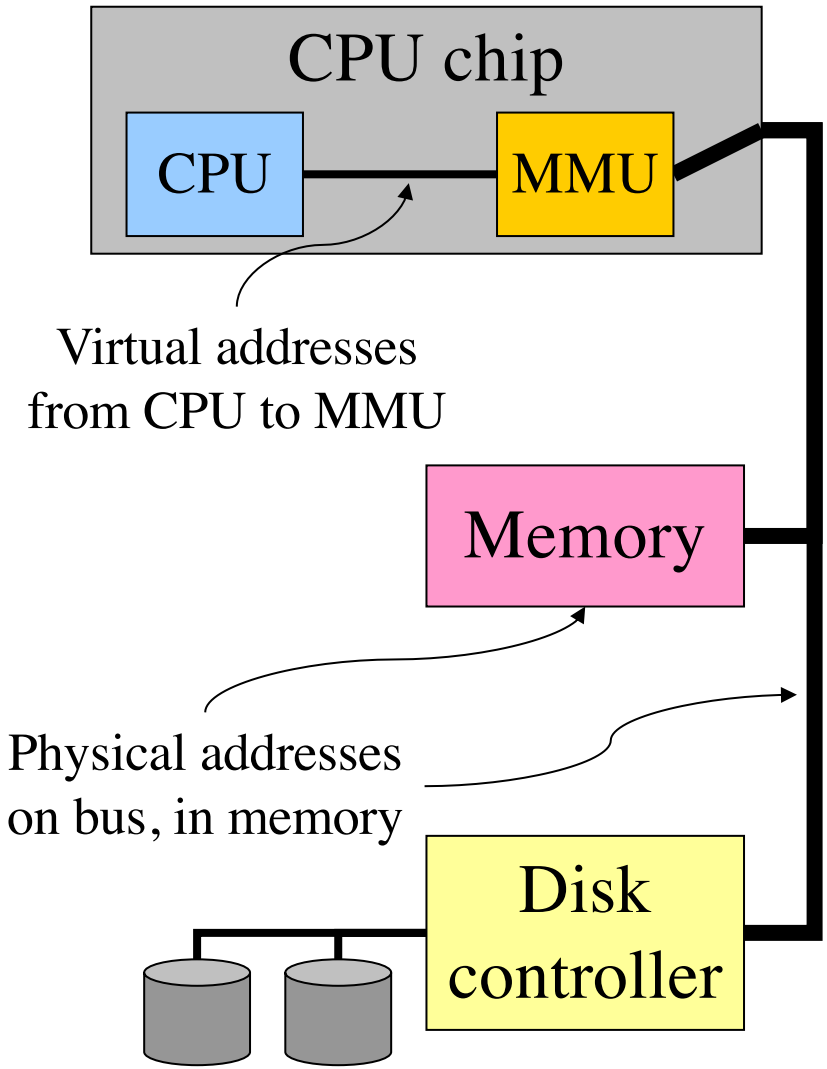
Base and limit registers

- Special CPU registers: base & limit
  - Access to the registers limited to kernel (privileged) mode
  - Registers contain
    - Base: start of the process's memory partition
    - Limit: length of the process's memory partition
- Address generation
  - Physical address: location in actual memory
  - Logical address: location from the process's point of view
  - Physical address = base + logical address
  - Logical address larger than limit => error



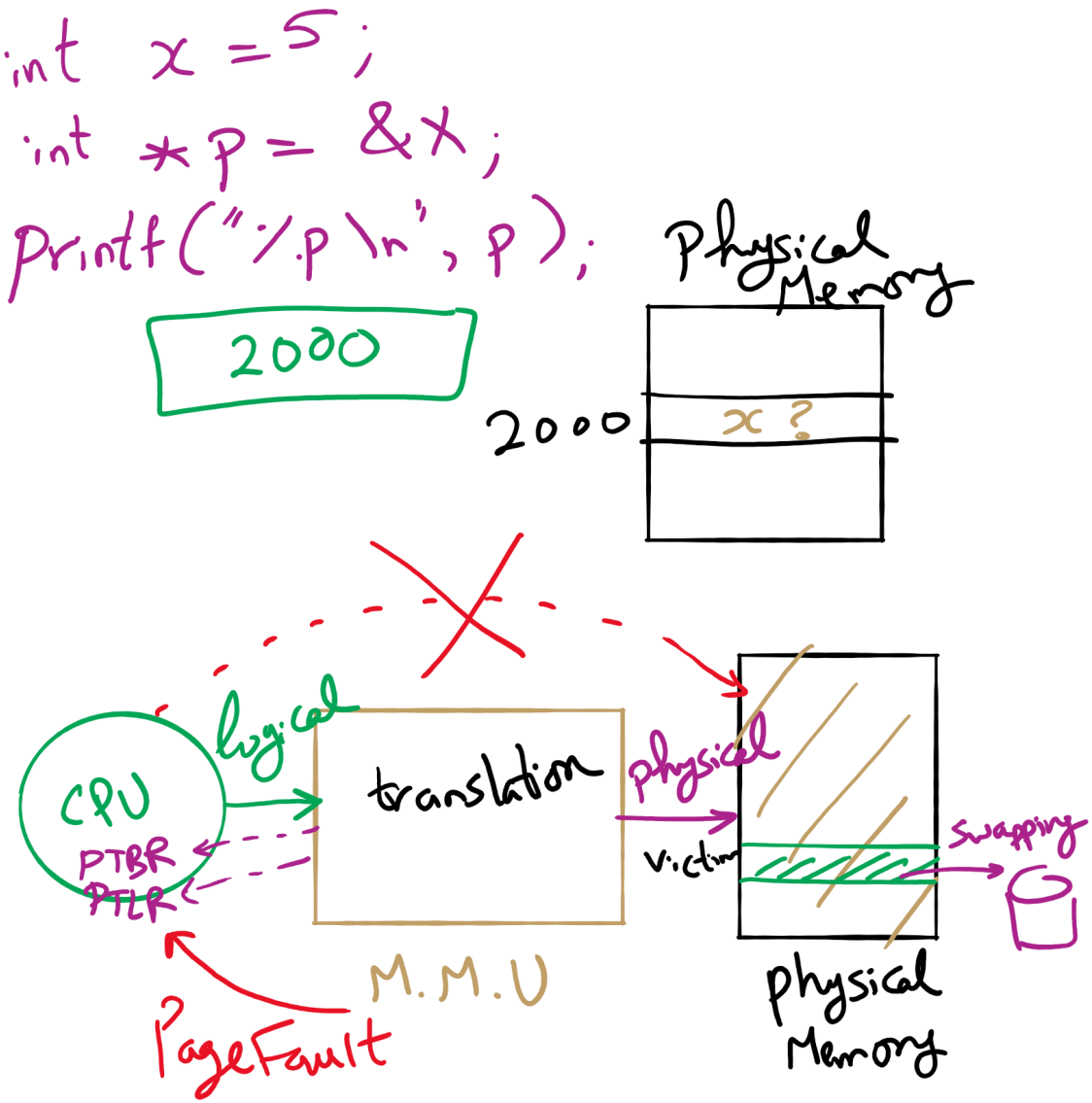
Logical address: 0x1204
Physical address:
0x1204+0x9000 = 0xa204

Virtual and physical addresses



- Program uses *virtual addresses*
  - Addresses local to the process
  - Hardware translates virtual address to *physical address*
- Translation done by the ***M**emory **M**anagement **U**nit*
  - Usually on the same chip as the CPU
  - Only physical addresses leave the CPU/MMU chip
- Physical memory indexed by physical addresses

Address Translation



Virtual memory

- Basic idea: allow the OS to hand out more memory than exists on the system
- Keep recently used stuff in physical memory
- Move less recently used stuff to disk
- Keep all of this hidden from processes
  - Processes still see an address space from 0 – max address
  - Movement of information to and from disk handled by the OS without process help
- Virtual memory (VM) especially helpful in multiprogrammed system
  - CPU schedules process B while process A waits for its memory to be retrieved from disk

Paging and page tables

- Virtual addresses mapped to physical addresses
  - Unit of mapping is called a *page*
  - All addresses in the same virtual page are in the same physical page
  - *Page table entry* (PTE) contains translation for a single page
- Table translates virtual page number to physical page number
  - Not all virtual memory has a physical page
  - Not every physical page need be used
- Example:
  - 64 KB virtual memory
  - 32 KB physical memory