Operating System Design

Booting the OS
Processes (intro)
OS types

Neda Nasiriani

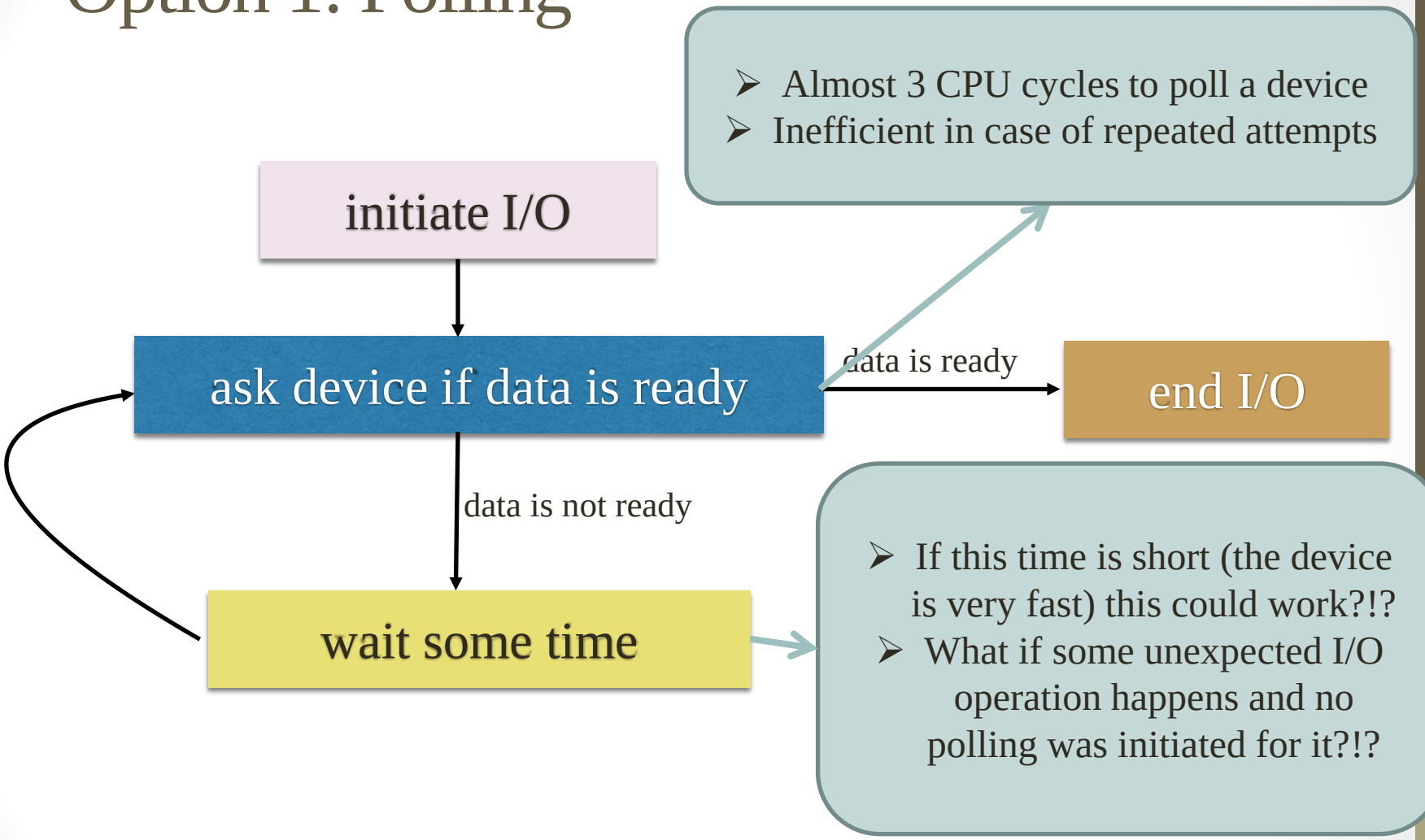
Fall 2018

Quiz 1 Review

I/O Devices Review

Two Mechanisms for Performing I/O operations?

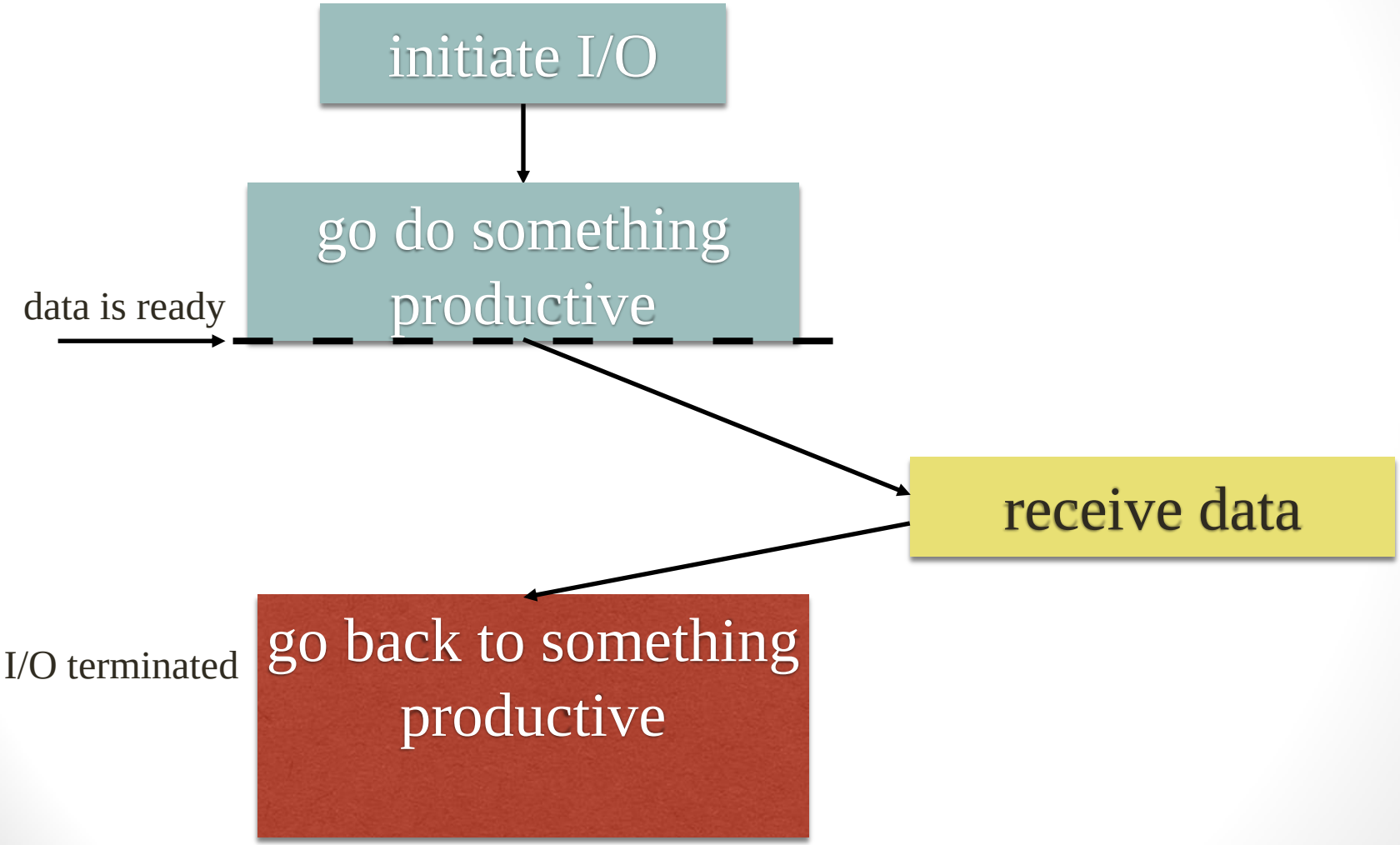
Option 1: Polling



The Simpsons: https://www.youtube.com/watch?v=18AzodTPG5U

Based on slides from Luiz F. Perrone

Option 2: Interrupt

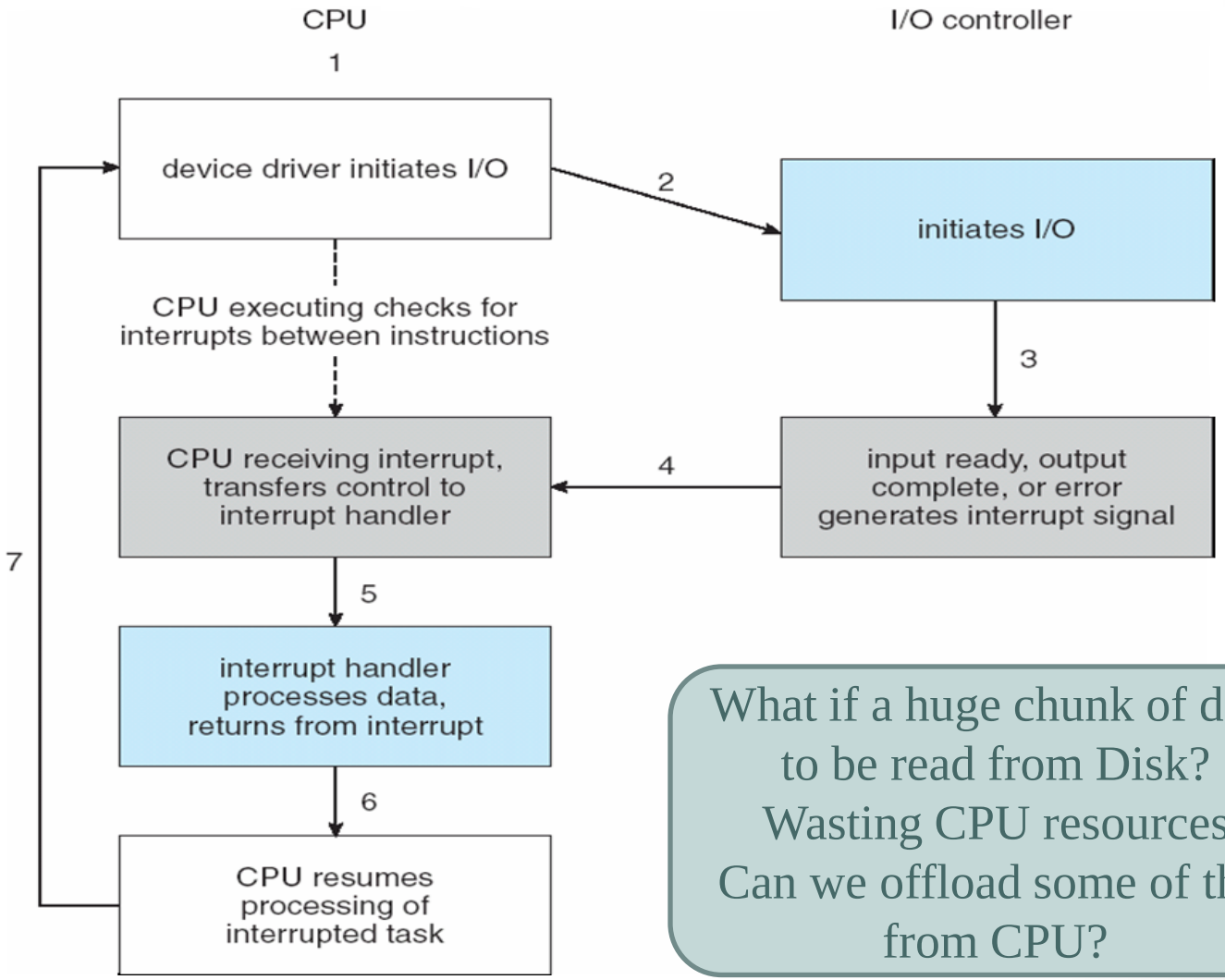


Based on slides from Luiz F. Perrone

Interrupts

- CPU **Interrupt-request line** triggered by I/O device
  - Checked by processor after each instruction
- **Interrupt handler** receives interrupts
  - **Maskable** to ignore or delay some interrupts
- **Interrupt vector** to dispatch interrupt to correct handler
  - Context switch at start and end
  - Based on priority
  - Some **nonmaskable**
  - Interrupt chaining if more than one device at same interrupt number
- Remember: **Traps** are software interrupts

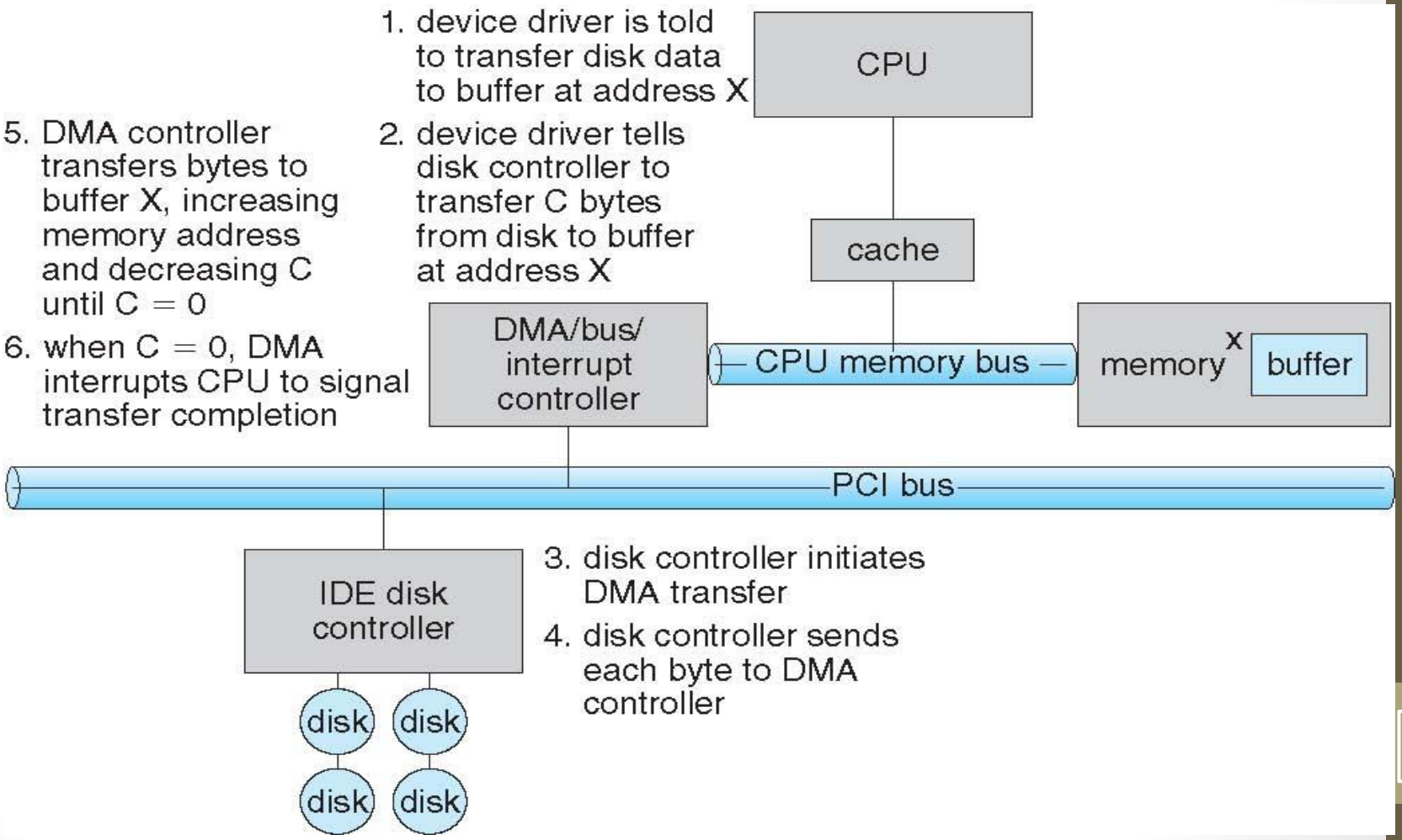
Interrupt Driven I/O Cycle



What if a huge chunk of data to be read from Disk?
Wasting CPU resources
Can we offload some of this from CPU?

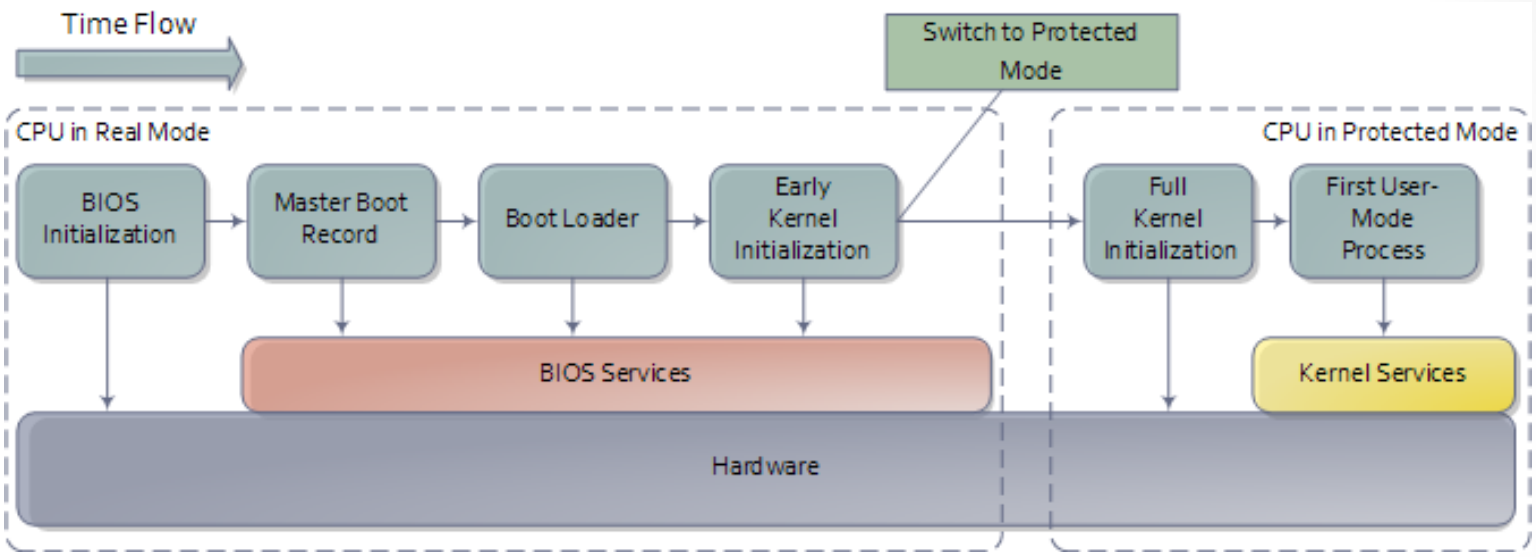
Direct Memory Access?

DMA: How it works?



Booting the OS

Booting up the OS

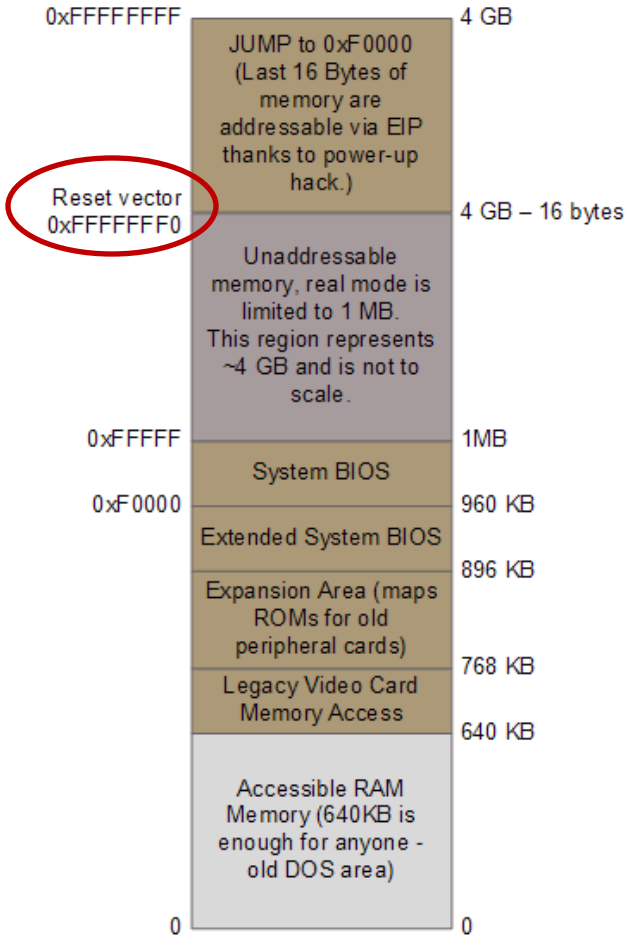


Source: http://duartes.org/gustavo/blog/post/how-computers-boot-up/

Booting Steps

1) Motherboard is powered up and loads its firmware
2) One of the CPUs are dynamically chosen to be the Boot Strap Processor (BSP)
    1) BSP runs the BIOS and kernel initialization code
    2) CPU is in real mode (Read what it is on your own)
3) Instruction Pointer (Program Counter) register holds the memory address for the reset vector
4) Reset vector holds the JUMP address to System BIOS address (0XF0000).
5) Note that the BIOS is stored on Flash disk which is mapped to BIOS address in memory (We will see more on this later)

Memory layout



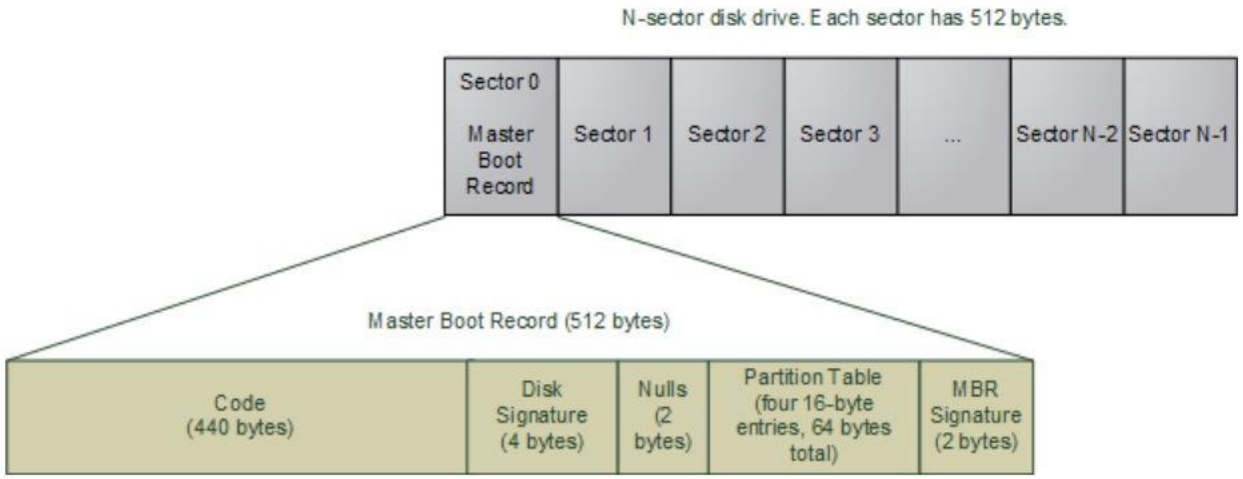
Source: http://duartes.org/gustavo/blog/post/how-computers-boot-up/

Booting Steps

6) CPU starts executing BIOS code
   - Initializes HW in the machine
   - BIOS starts Power On Self Test (POST)
     - POST tests various components in the system, e.g., video card
     - POST takes care of initialization, e.g., I/O ports, memory range, interrupts
   - BIOS also build data tables that describe the devices in computer system
7) Now the BIOS wants to boot up an Operating System
   - Boot device can be chosen by the user (CD-ROM, hard drive,…)
   - The order to look for boot device can be configured by the user

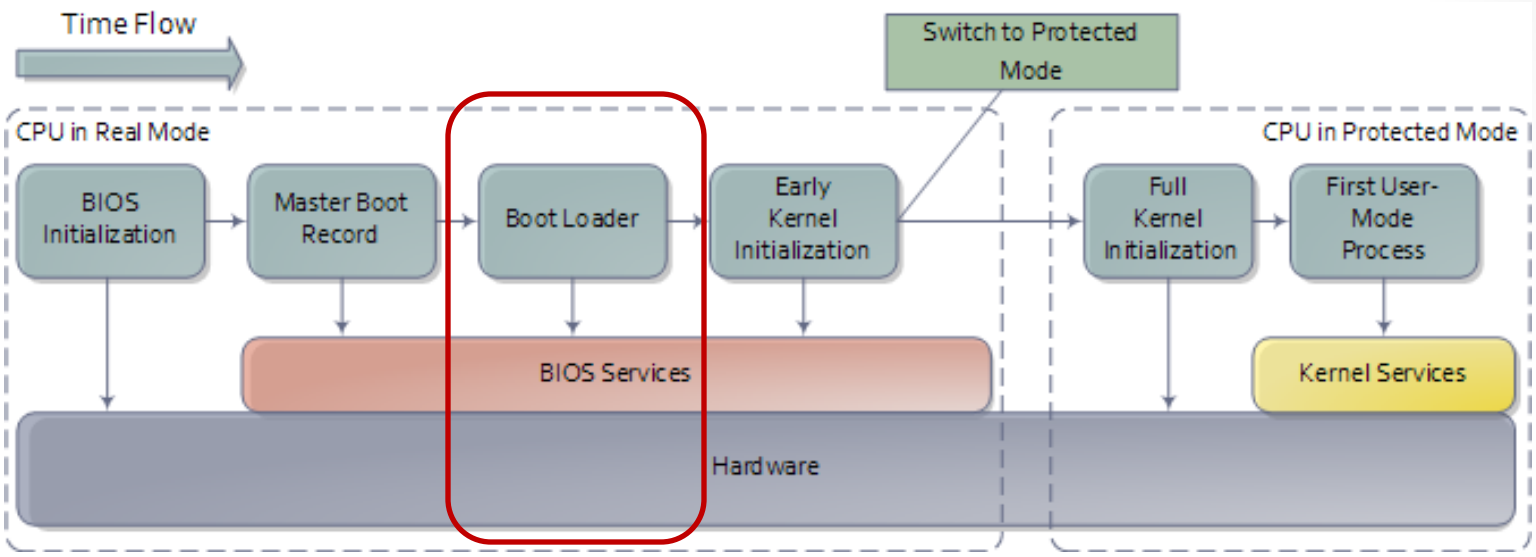
Booting Steps

6) CPU starts executing BIOS code
   - Initializes HW in the machine
   - BIOS starts Power On Self Test (POST)
     - POST tests various components in the system, e.g., video card
     - POST takes care of initialization, e.g., I/O ports, memory range, interrupts
   - BIOS also build data tables that describe the devices in computer system
7) Now the BIOS wants to boot up an Operating System
   - Boot device can be chosen by the user (CD-ROM, hard drive,…)
   - The order to look for boot device can be configured by the user
8) BIOS reads the first 512-byte sector of the hard disk (which is master boot record (MBR))
   - MBR contains: (i) tiny OS-specific bootstrapping program, (ii) partition table for the disk
   - MBR is loaded to memory for execution
     - The code part of MBR could be windows loader, or linux GRUB,…
     - The disk partition table is a standard table with specific format, so different OSs can be installed on the same machine



Check out this link for an example of MBR
https://www.bydavy.com/2012/01/lets-decrypt-a-master-boot-record/

Booting up the OS

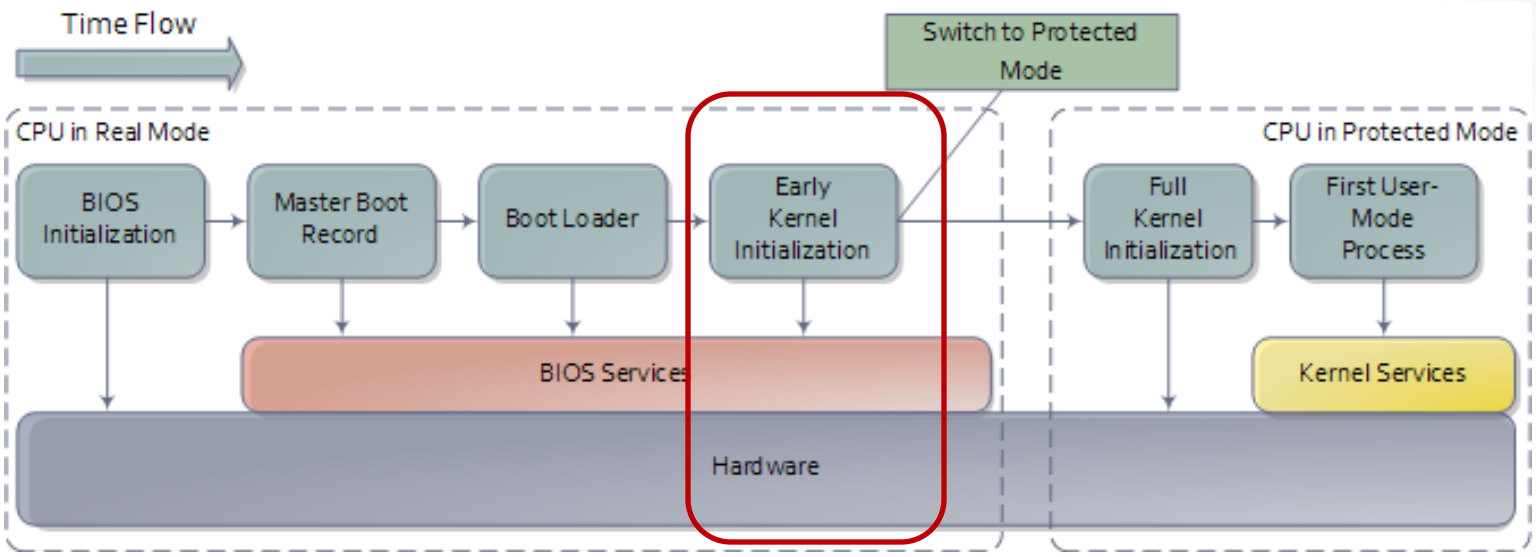


Source: http://duartes.org/gustavo/blog/post/how-computers-boot-up/

Booting Steps

9) Boot loader either chooses the partition by itself, which is the case for windows loader, or prompt the user to choose from the available partitions in case of Linux GRUB
10) The first sector of the partition is called **boot sector**,
11) Now it is time to load the Operating System from the file system (**boot partition**)
    - Windows server 2003: some of the kernel start-up code is separated from the kernel image and needs to be executed first
    - Linux: loads a file named “vmlinux-2.6….” containing the kernel into memory and jumping to kernel bootstrap code

Booting up the OS



- BIOS is firmware (flash memory). Power on self tests (POST) check if machine is in shape to run.
- Every disk has an MBR, which contains a bootstrap program and a partition table. Each partition has a boot sector with the boot loader.
- How does the machine know the address of the first instruction to run???

Based on slides from Luiz F. Perrone

Source: http://duartes.org/gustavo/blog/post/how-computers-boot-up/

Processes

Processes

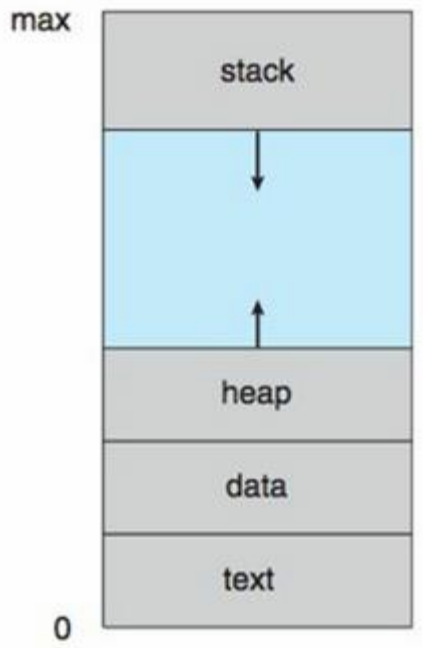
- What is a process?
  - Informally: a program in execution
- What are the main components of a process?
  - Text section
    - The code
  - Stack
    - Local variables
    - Function parameters
    - …
  - Heap
    - Dynamically allocated memory
  - Data Section
    - Global variables
  - What else?

Processes (continued)

- Assume processes A and B are running in a system
  - The CPU decides to switch from process A to process B
  - What information will the CPU need to resume process A later?
    - Program Counter
    - Value of registers
- SO, a process is associated with the following components
  - Text section
  - Data section
  - Heap
  - Stack
  - Program Counter
  - Value of Registers

Processes (continued)

- Assume processes A and B are running in a system
  - The CPU decides to switch from process A to process B
  - What information will the CPU need to res
    - Program Counter
    - Value of registers
- SO, a process is associated with the follov
  - Text section
  - Data section
  - Heap
  - Stack
  - Program Counter
  - Value of Registers
- The process in memory looks like this

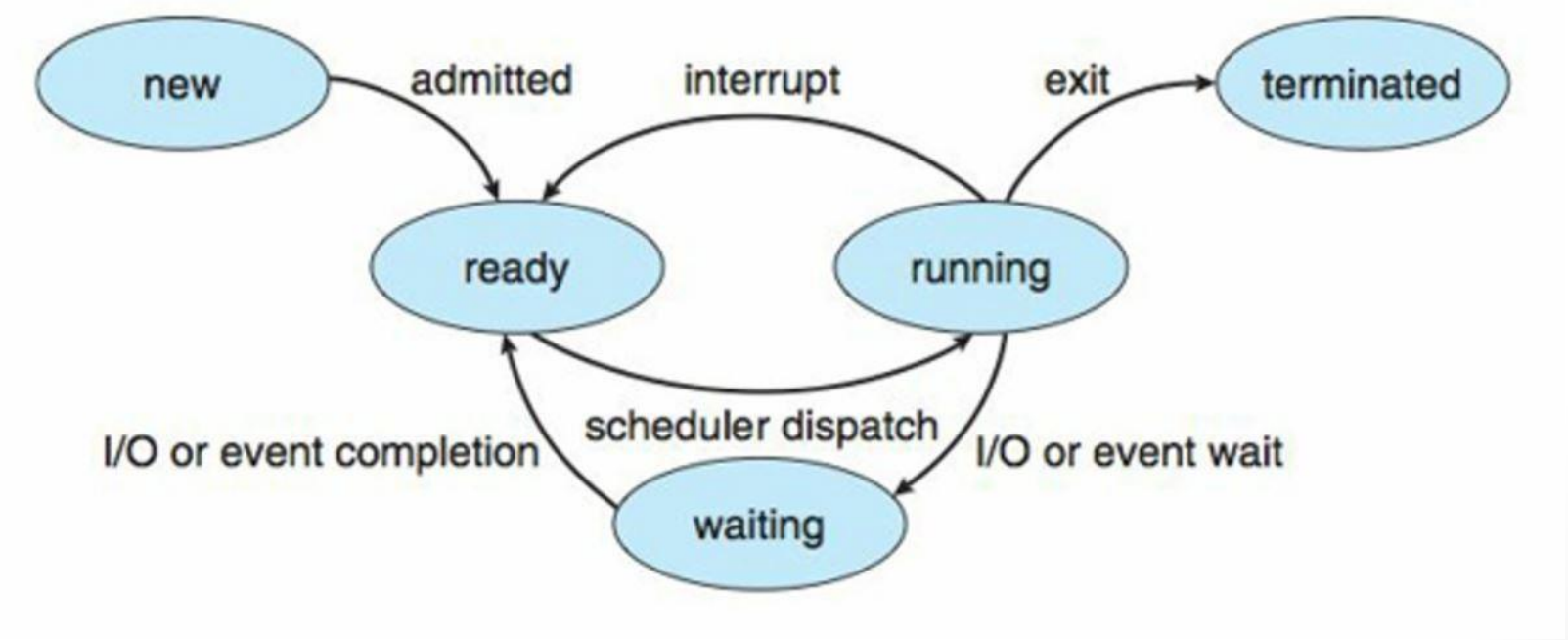


Processes States

As a process executes in the system its state changes

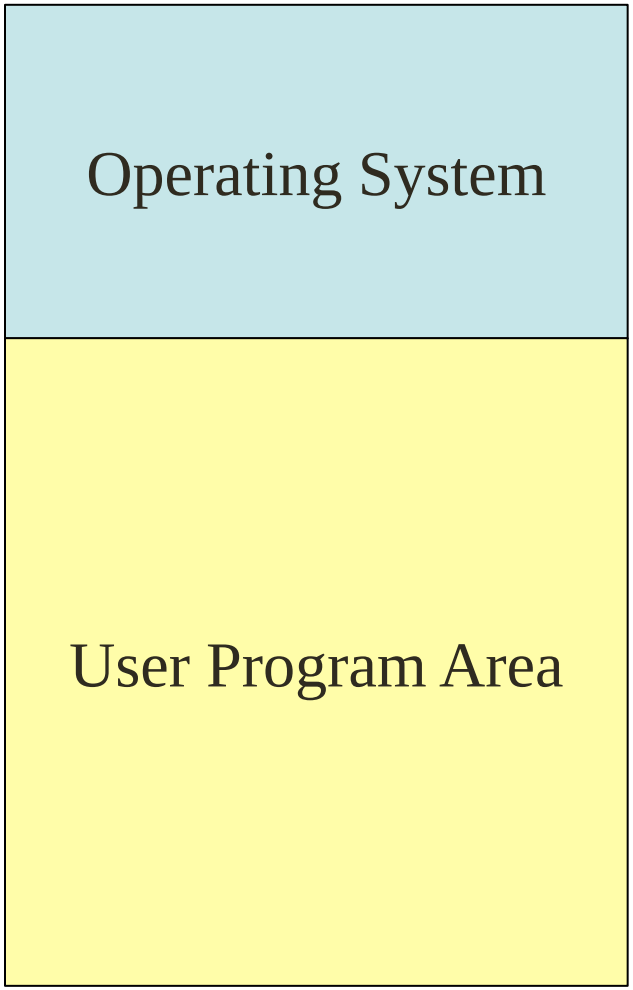
- new: The process is being created.
- running: Instructions are being executed.
- waiting: The process is waiting for some event to occur.
- ready: The process is waiting to be assigned to a processor.
- terminated: The process has finished execution.

Process Lifecycle



OS types

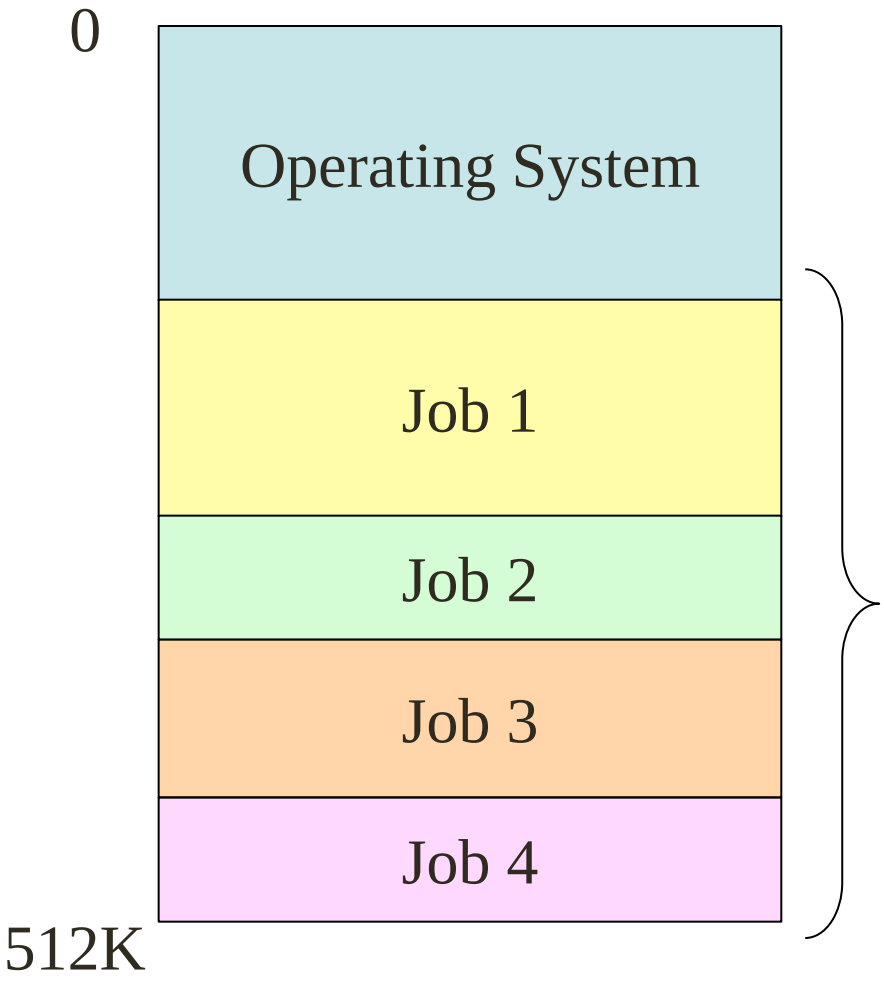
Memory Layout for a Simple Batch System



One program at a time is loaded, runs to completion, and leaves the system.

Based on slides from Luiz F. Perrone

Multiprogrammed Batch Systems



Several jobs are kept in main memory at the same time, and the CPU is multiplexed among them.

Based on slides from Luiz F. Perrone

OS Features Needed for Multiprogramming

- I/O routines supplied by the system.
- Memory management – allocate memory to each of several jobs.
- CPU scheduling – determine which job runs when.
- Control access to multiple devices.

Based on slides from Luiz F. Perrone

Time-Sharing Systems: Interactive Computing

- The CPU is multiplexed among several jobs that are kept in memory and on disk (the CPU is allocated to a job only if the job is in memory).
- On-line interaction among the user and the system is provided:
  - User can input any command at any time
- On-line system must be available for users to see the result of his/her task
- Each user has the impression that the entire resources are dedicated to his/her task
- CPU scheduling is very crucial in this type of system. We well get to this later in the course!

Based on slides from Luiz F. Perrone

FYR: Desktop Systems

- Personal computers – computer system dedicated to a single user.
- I/O devices – keyboards, mice, display screens, small printers.
- User convenience and responsiveness.
- Can adopt technology developed for larger operating system:
  - Often individuals have sole use of computer and do not need advanced CPU utilization of protection features.
- May run several different types of operating systems (Windows, MacOS, UNIX, Linux).

Multiprocessor Systems: Parallel Systems

- Systems with more than one CPU in close communication
- *Tightly coupled system* – processors share memory and a clock; communication usually takes place through the shared memory.
- Advantages of parallel system:
  - Increased *throughput*
    - Not linear increase though because of overheads of coordination between tasks and resource contention
  - Economical
    - Sharing peripherals
  - Increased reliability (in some cases)
    - If tasks are distributed properly then failure of one processor will not halt the system

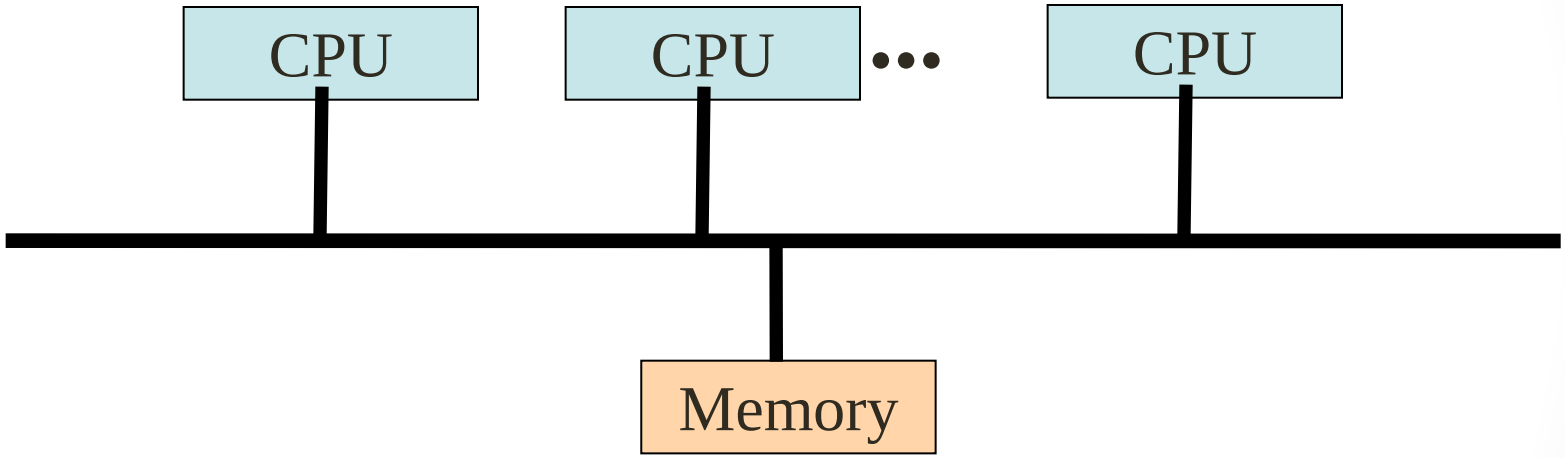
Based on slides from Luiz F. Perrone

FYR: Multiprocessor Systems: Parallel Systems (Cont.)

- *Asymmetric multiprocessing*
  - Each processor is assigned a specific task; master processor schedules and allocated work to slave processors.
  - More common in extremely large systems.
- *Symmetric multiprocessing (SMP)*
  - Each processor runs an identical copy of the operating system.
  - Many processes can run at once without performance deterioration.
  - Most modern operating systems support SMP.

Based on slides from Luiz F. Perrone

FYR: Symmetric Multiprocessing Architecture



Based on slides from Luiz F. Perrone

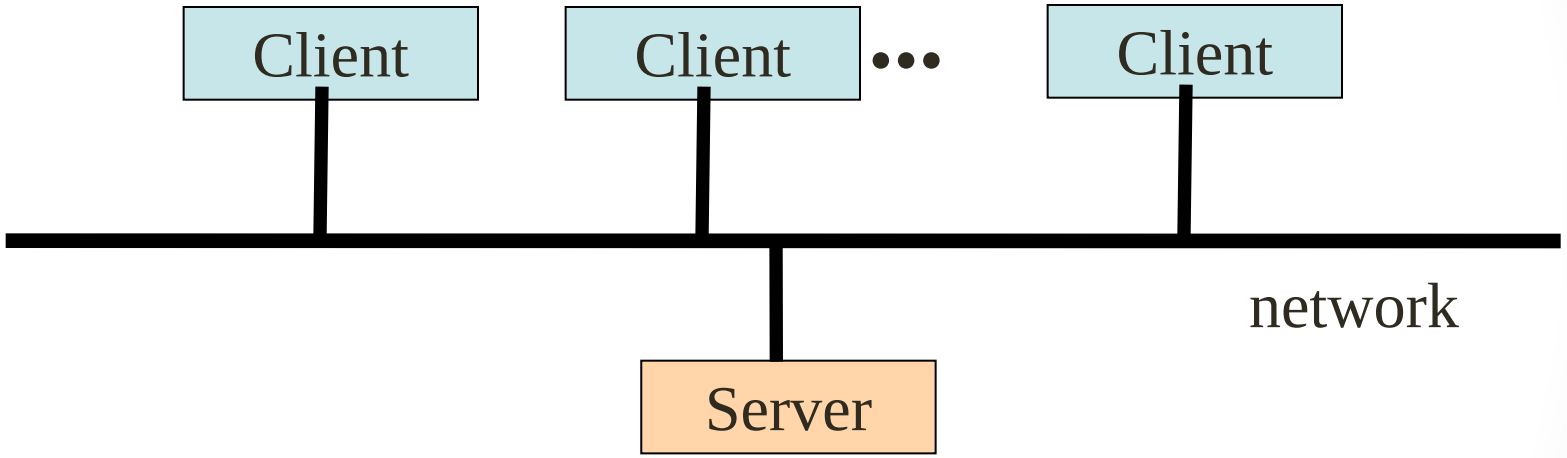
Distributed Systems

- Distribute the computation among several physical processors.
- *Loosely coupled system* – each processor has its own local memory; processors communicate with one another through various communications lines, such as high-speed buses or telephone lines.
- Advantages of distributed systems:
  - Resources Sharing,
  - Computation speed up – load sharing,
  - Reliability,
  - Communications.

Distributed Systems (cont.)

- Requires networking infrastructure.
- Local area networks (*LAN*) or Wide area networks (*WAN*).
- May be either *client-server* or *peer-to-peer* systems.

General Structure of Client-Server System



Based on slides from Luiz F. Perrone

FYR: Clustered Systems

- Clustering allows two or more systems to share storage.
- Provides high reliability.
- *Asymmetric clustering*: one server runs the application or applications while other servers standby.
- *Symmetric clustering*: all N hosts are running the application or applications.

Based on slides from Luiz F. Perrone

Real-Time Systems

- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.
- Well-defined fixed-time constraints.
- Real-Time systems may be either *hard* or *soft* real-time.

Real-Time Systems (Cont.)

- Hard real-time:
  - Secondary storage limited or absent, data stored in short term memory, or read-only memory (ROM).
  - Conflicts with time-sharing systems, not supported by general-purpose operating systems.
- Soft real-time:
  - Limited utility in industrial control of robotics.
  - Can be integrated with time-shared systems.
  - Useful in applications (multimedia, virtual reality) requiring tight response times.

Embedded Systems

- Appliances.
- Smart sensors.
- Digital control systems.
- Issues:
  - Limited memory,
  - Slower processors,
  - Small display screens (if any).

Next Session

- Continue Processes