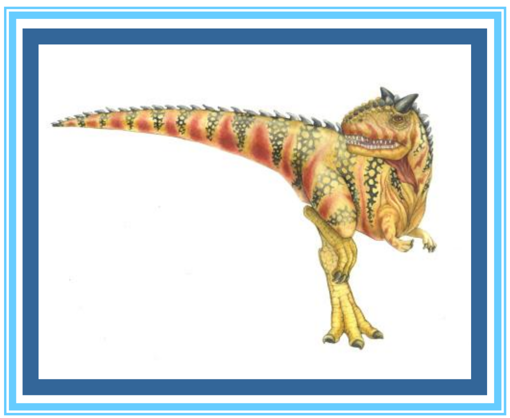
Chapter 1: Introduction

Reading: Chapter 1.1-1.6



Today's Lecture

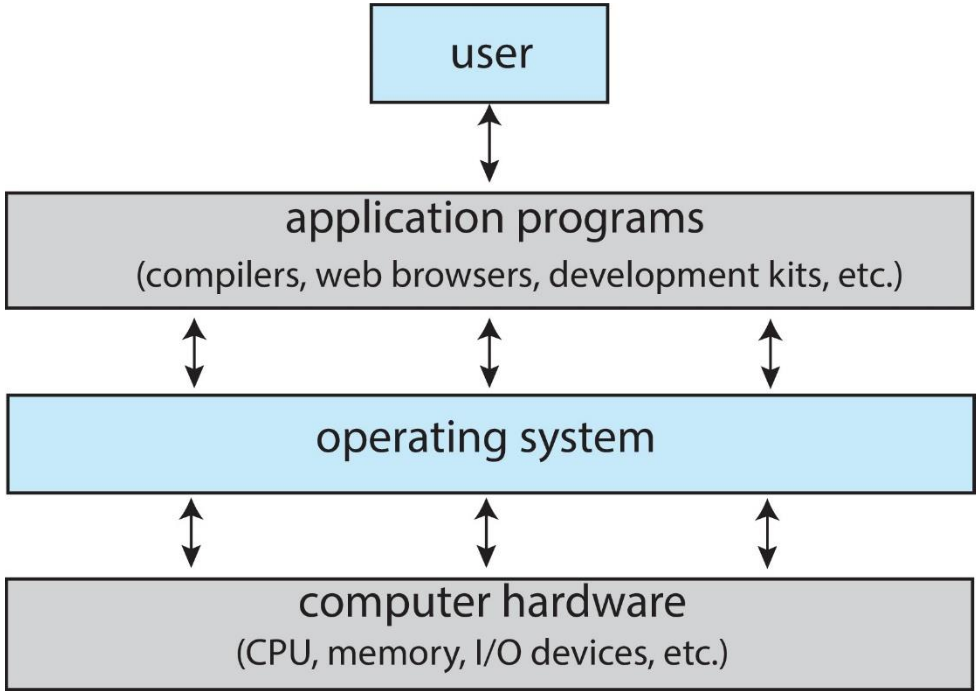
- Reading 1.1-1.3
  - What Operating Systems do
  - Computer System Organization

Operating System: What comes to your mind?

Operating System

- Software that manages computer hardware
- Acts as an intermediary between user and hardware
- Provides an environment for software applications

Components of a Computer System



What Operating Systems Do

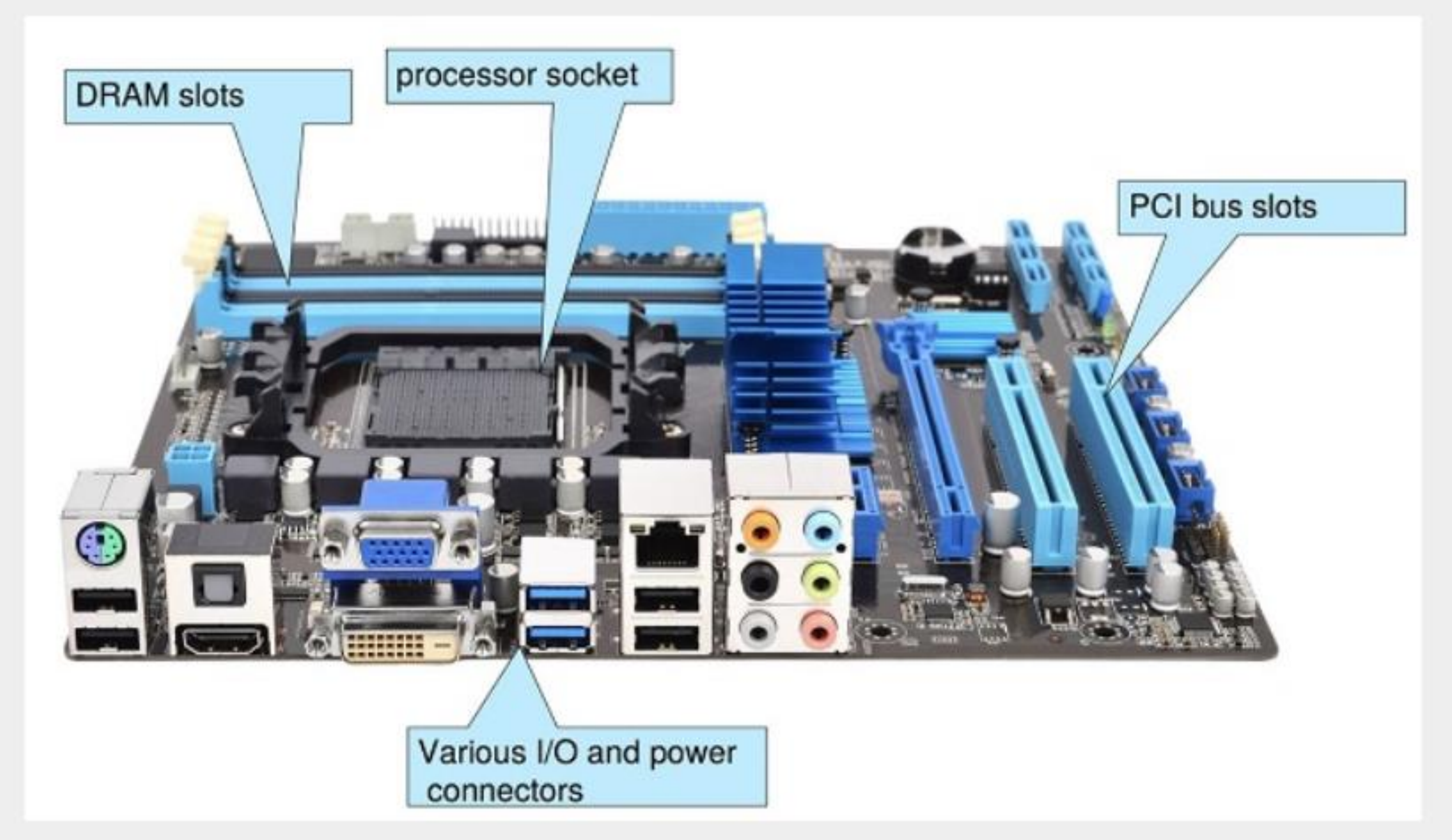
- OS is a **resource allocator**:
  - Allocates resources (e.g., CPU time, memory space, disk space, I/O devices) to specific programs *efficiently* and *fairly*
- OS is a **control program:**
  - Controls the execution of user programs to prevent errors and improper use of the computer
  - Controls I/O devices

Operating System Definition

- An operating system consists of
  - **kernel** – core of OS, running at all times on the computer
  - **system programs** (ships with the OS, but not part of the kernel)
    - file management programs, device drivers, GUI systems, text editors, compilers, debuggers, etc.

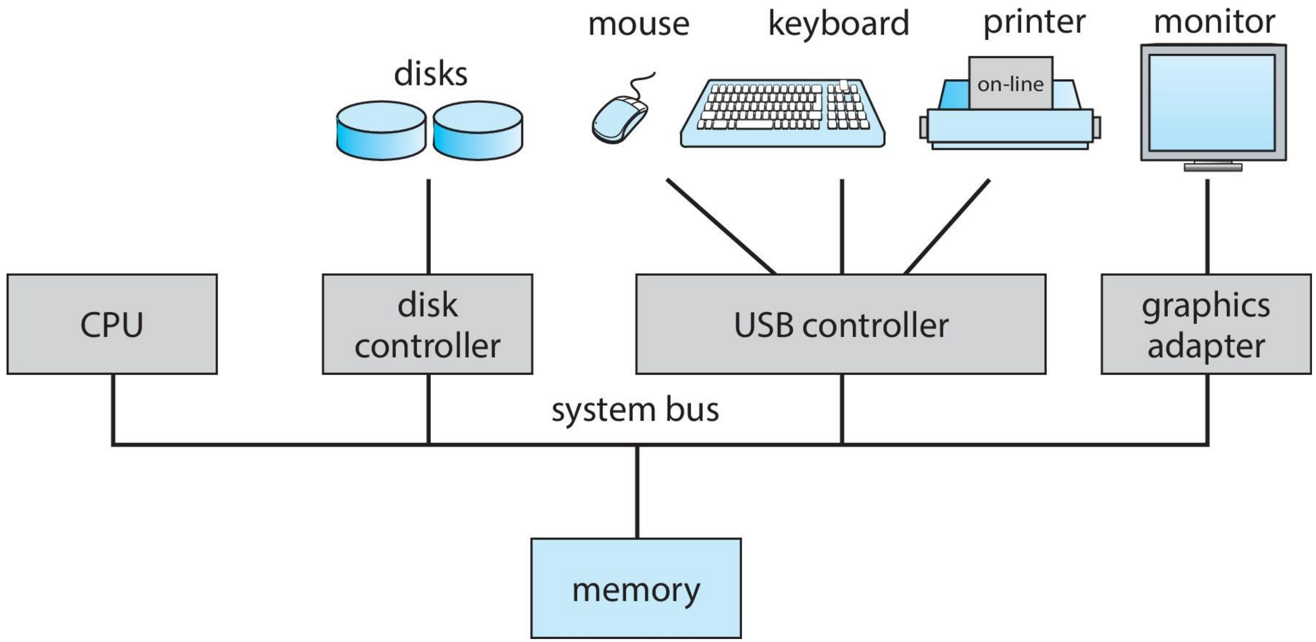
- **Application programs** are all programs not associated with the OS
  - Web browsers, word processors, database systems, video games, etc.

PC Motherboard



Computer System Organization

- One or more CPUs and device controllers connect through common **bus** providing access to shared memory
  - Each device controller is in charge of a particular device type
- CPUs and device controllers can execute in parallel, competing for memory cycles



I/O Operations

- OS has a **device driver** for each device controller
- Each device controller has a local buffer
  - Device controller moves data between its local buffer and the devices it controls
- Device driver initiates I/O on device controller
  - In case of output, CPU moves output data from main memory to local buffer of device controller
- Device controller informs CPU that it has finished its I/O operation by causing an **interrupt**
  - In case of input, CPU moves input data from local buffer of device controller to main memory

Interrupts

- Operating systems are **interrupt driven**
- An interrupt may be triggered by hardware (e.g., I/O device) or software
  - A software-generated interrupt is called a **trap** (or **exception**)
  - A **trap** is caused either by an error or a request from a user program
- Interrupt transfers control to the appropriate interrupt service routine (ISR) generally through the **interrupt vector**, which contains the addresses of all the ISRs
- Upon an interrupt, OS preserves the state of the CPU by storing registers and the program counter so that the CPU can resume the interrupted computation after servicing the interrupt

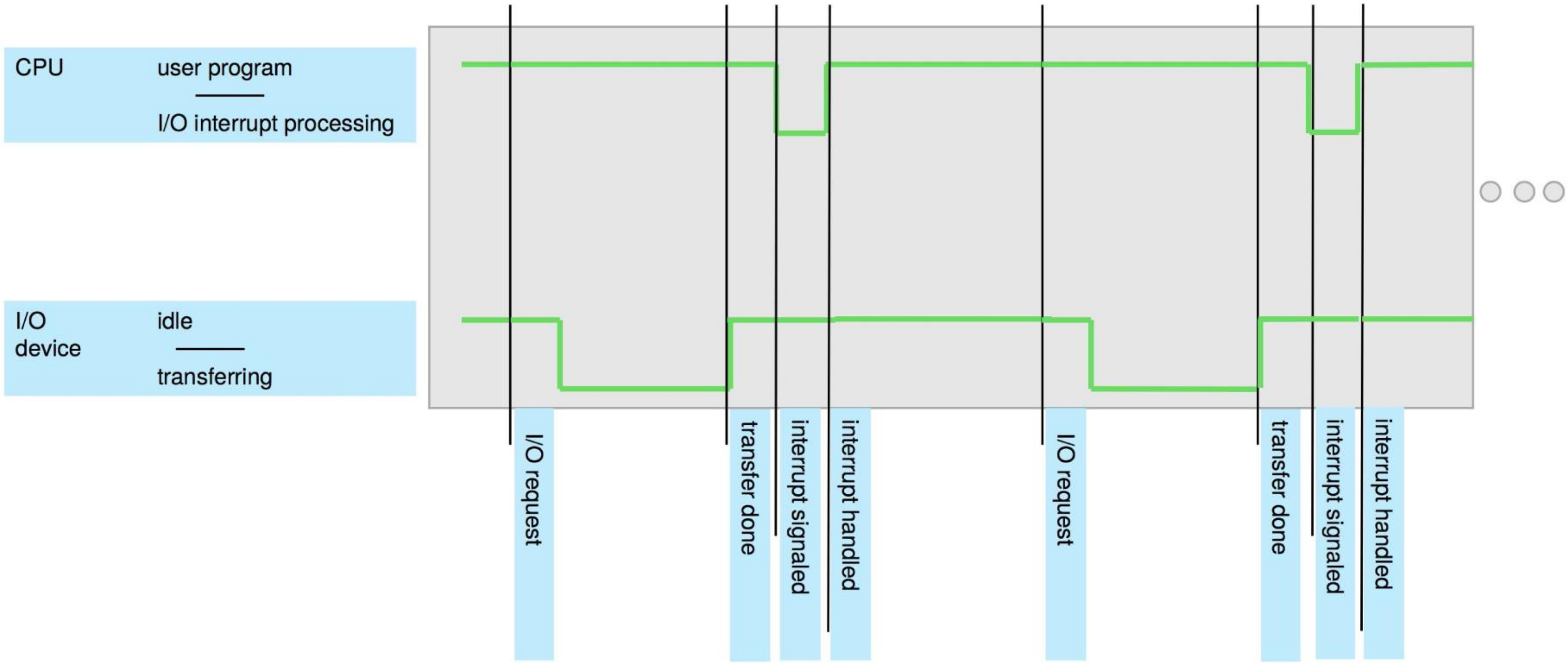
- Review Animation 1.2: Interrupt driver I/O cycle

Interrupt Vector Table

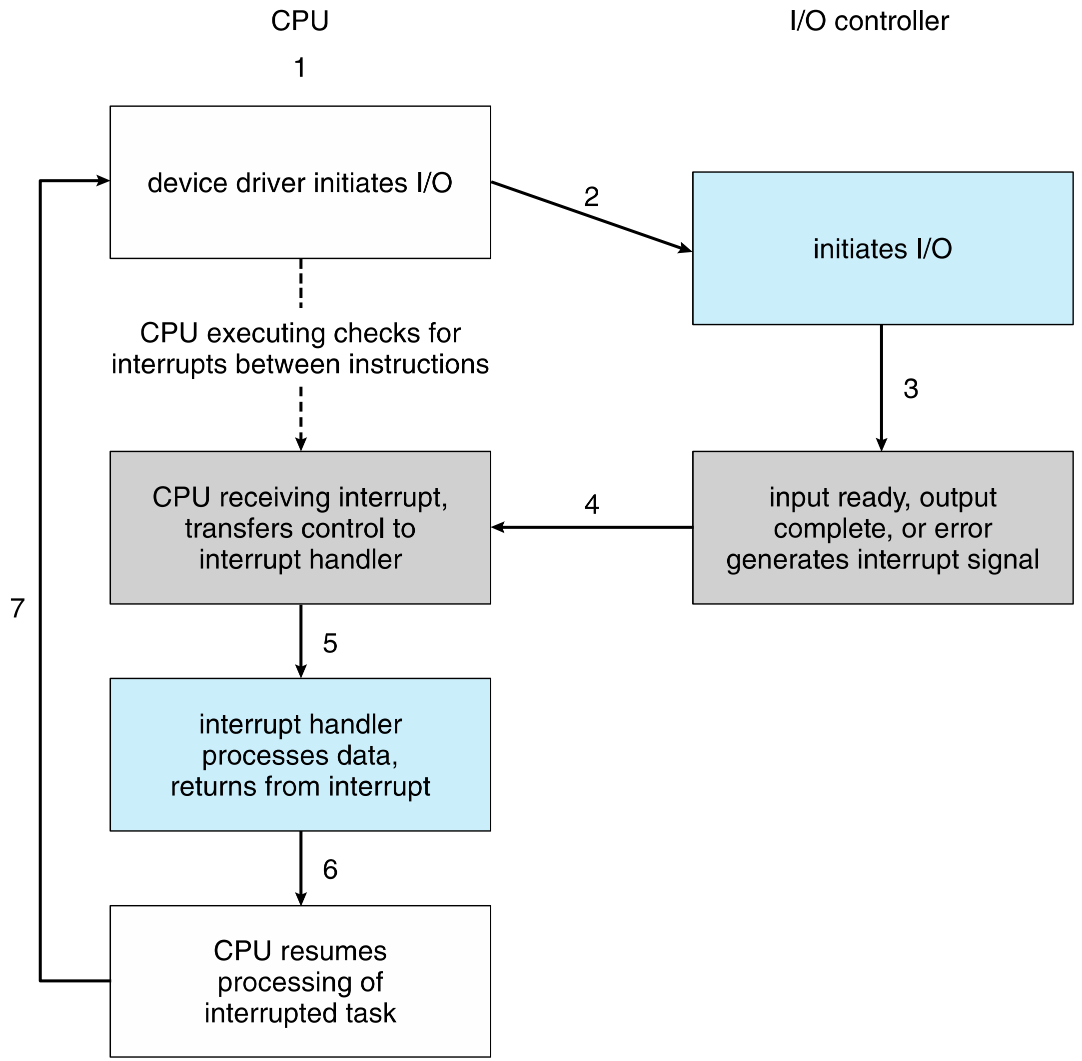
vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Figure 1.5 Intel processor event-vector table.

Interrupt Timeline



Interrupt-driven I/O Cycle



Multiprocessor system

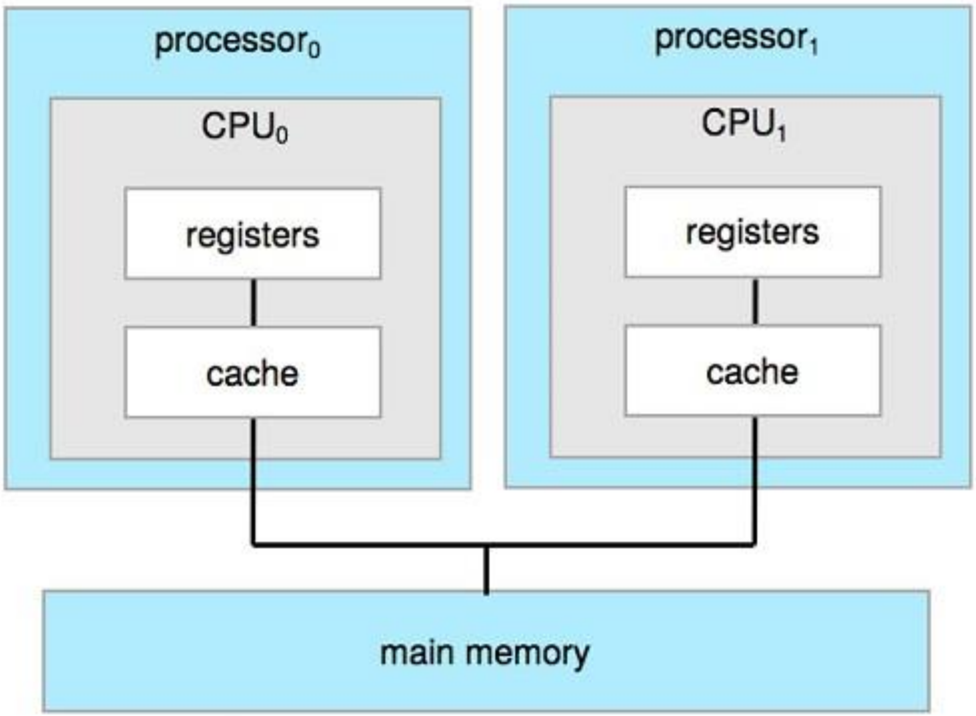


Figure 1.8 Symmetric multiprocessing architecture.

Multicore system

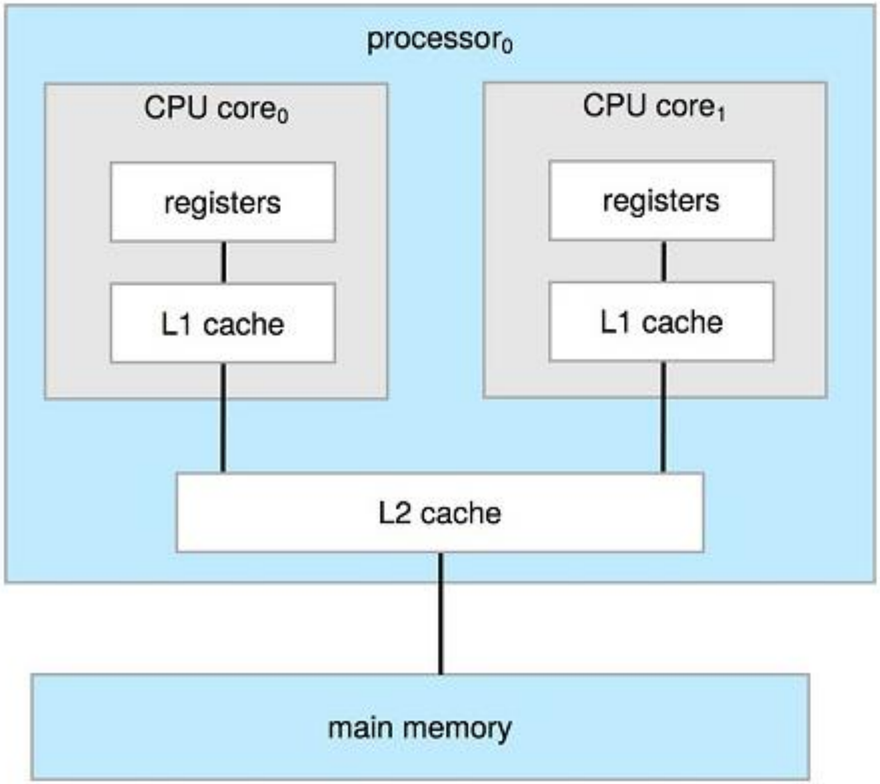
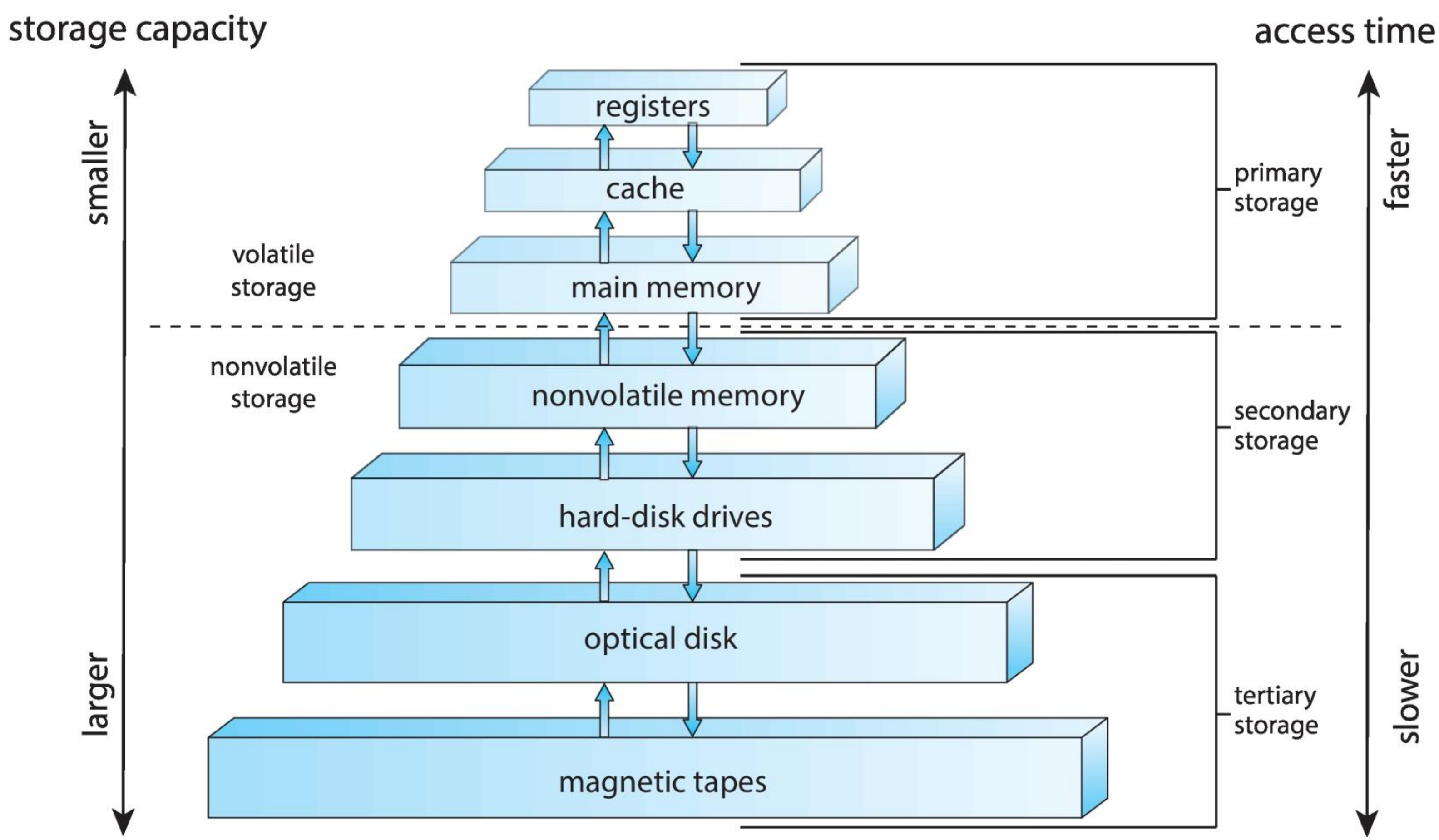


Figure 1.9 A dual-core design with two cores on the same chip.

Storage-Device Hierarchy



Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - Typically **volatile,** i.e., loses contents when power is removed
  - Any programs must be first loaded into main memory to run
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
  - **Mechanical: Hard Disk Drives** (**HDD**)
  - **Electrical: nonvolatile memory (NVM)** devices– faster than hard disks
    - Various technologies: flash memory, solid-state drive (SSD)

A short quiz

- Interrupts are triggered only by hardware
  - True
  - False

OS Operations

- OS is **interrupt driven**
  - Hardware interrupt generated by one of the devices
  - Software interrupt (**trap** or **exception**):
    - Error (division by zero, invalid memory access)
    - Request for operating system service – **system call**

Today's Lecture

- Reading 1.4-1.6
    - Operating-System Operations
    - Resource Management
    - Security and Protection

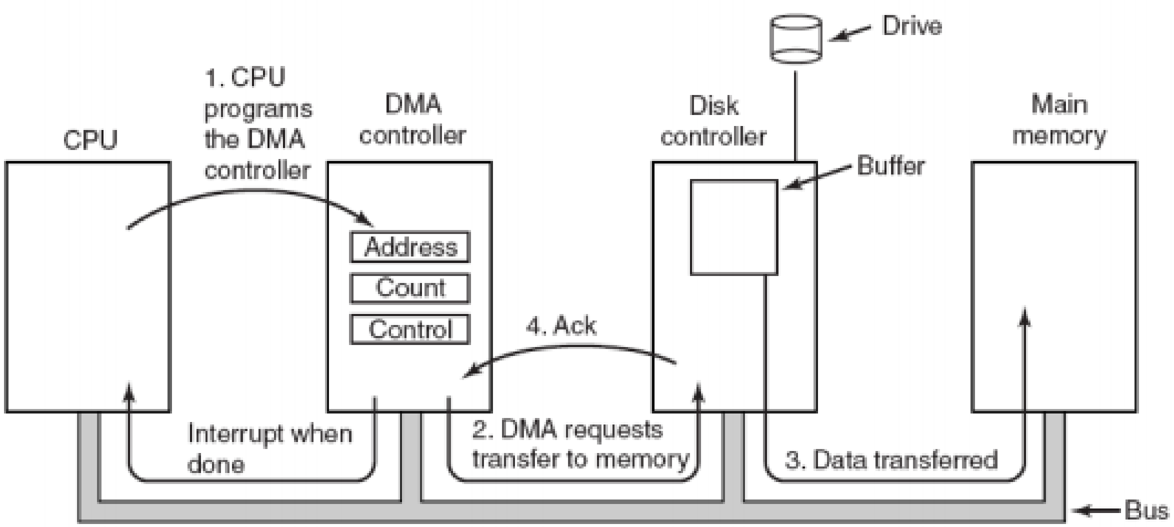
I/O Structure

- Drawbacks of interrupt-driven I/O
  - Transfer data one byte at a time
  - CPU involved in data transfer between memory and device controller
  - Incurs high overhead when used for bulk data movement (e.g., disk I/O)
- **Direct Memory Access (DMA)** – DMA controller tells the device controller to transfer a block of data directly between its buffer storage and memory without CPU intervention
  - CPU can do other work while data transfer is in progress
  - DMA controller interrupts the CPU when entire block is transferred

How a Modern Computer Works

- Review Animation 1.3

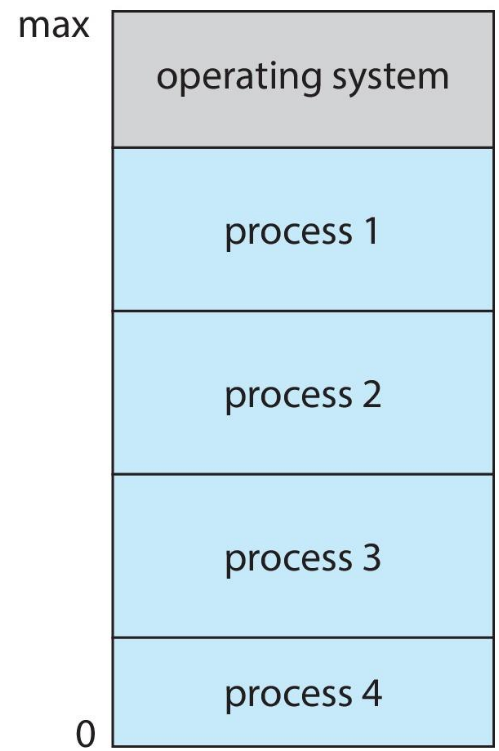
DMA



Operation of a DMA transfer

Multiprogramming

- **Multiprogramming** needed for efficiency
  - Single program cannot keep CPU and I/O devices busy at all times
  - OS keeps several **processes** in memory simultaneously so CPU always has a process to execute
    - A program in execution is called a **process**
  - One process is selected and run via **CPU scheduling**
  - When a process has to wait (e.g., for I/O completion), OS switches to another process



Memory layout

Exercise

Consider two programs:

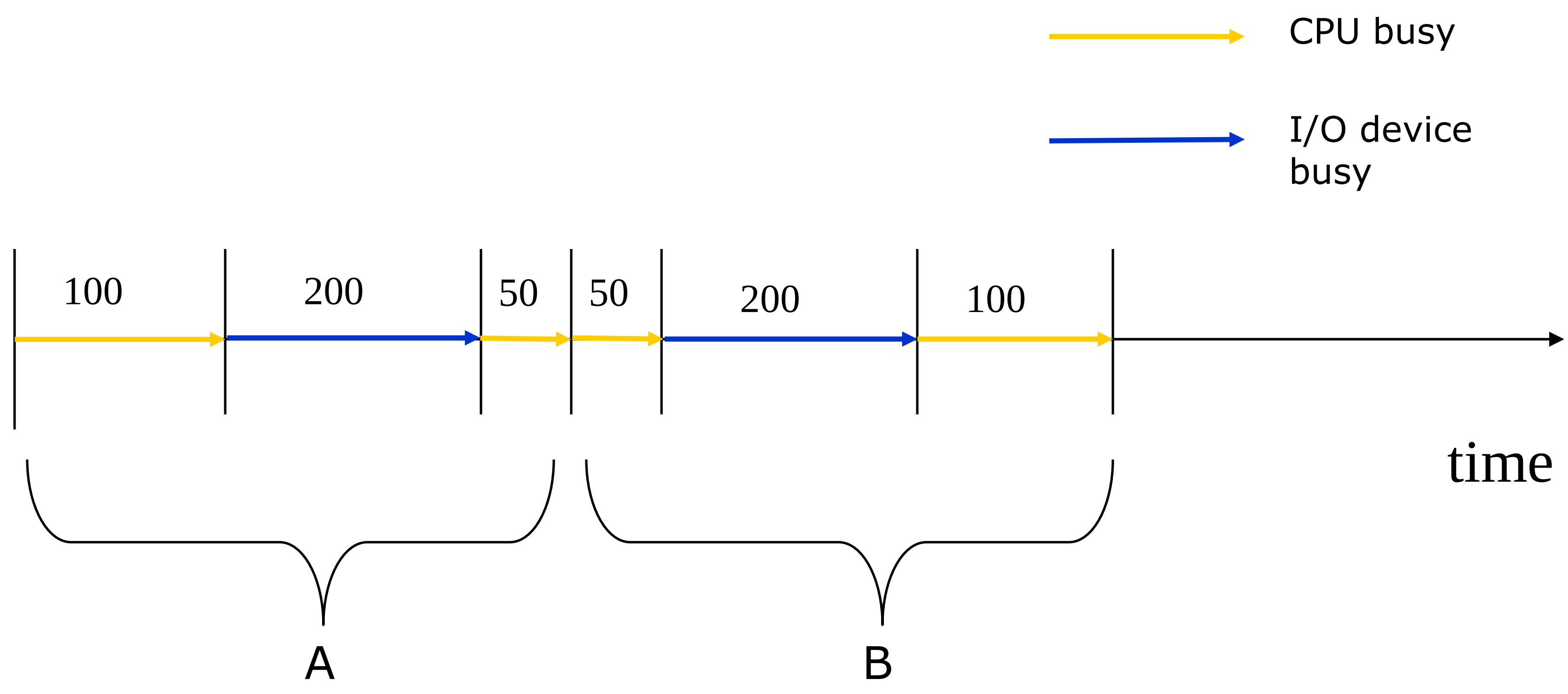
- Program A: computation (100ms), then I/O on device 1 (200ms), then computation (50ms)
- Program B: computation (50ms), then I/O on device 2 (200ms), then computation(100ms)

How long it will take to complete the two programs when

(1) Monoprogramming (i.e., only one program is kept in memory) is used?

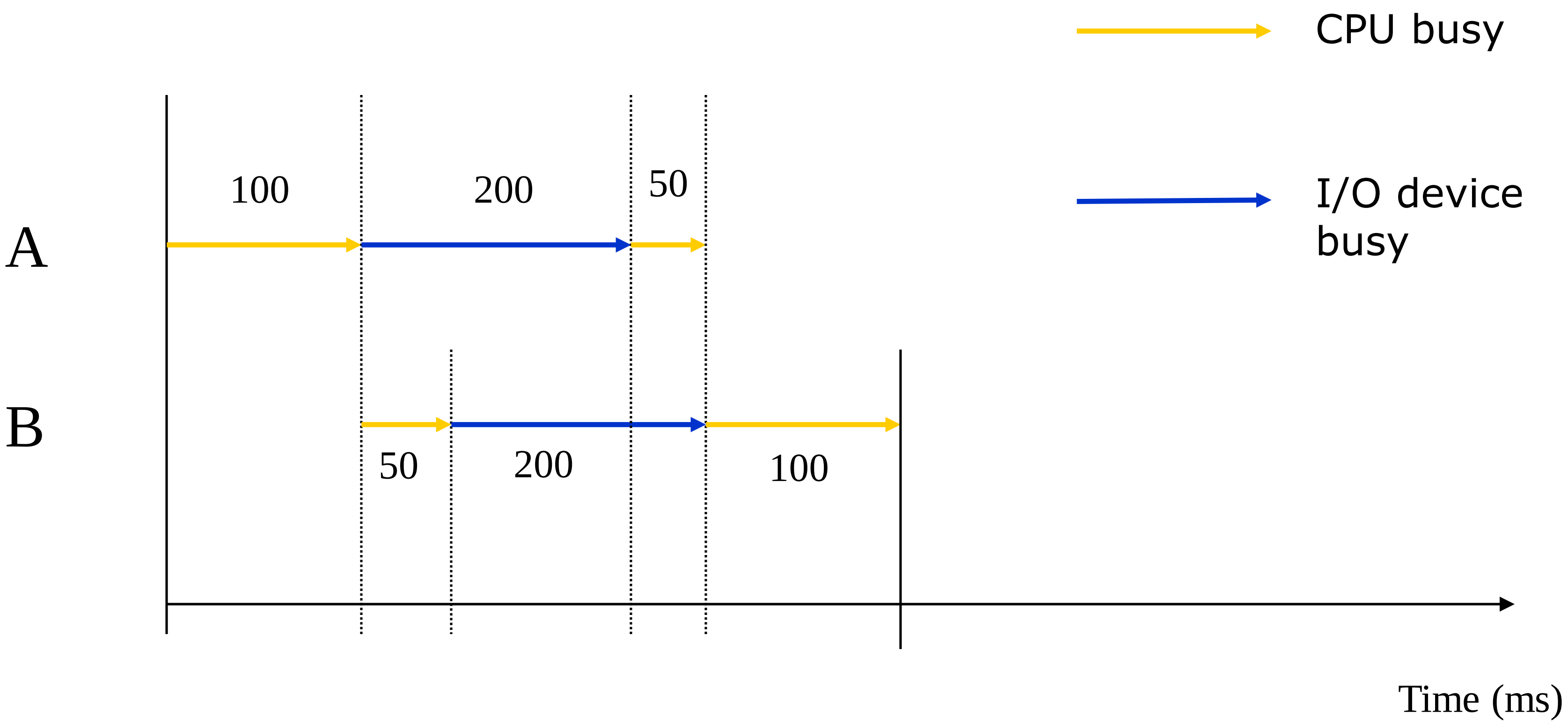
(2) Multiprogramming is used?

Monoprogramming



Total time = 700ms

Multiprogramming



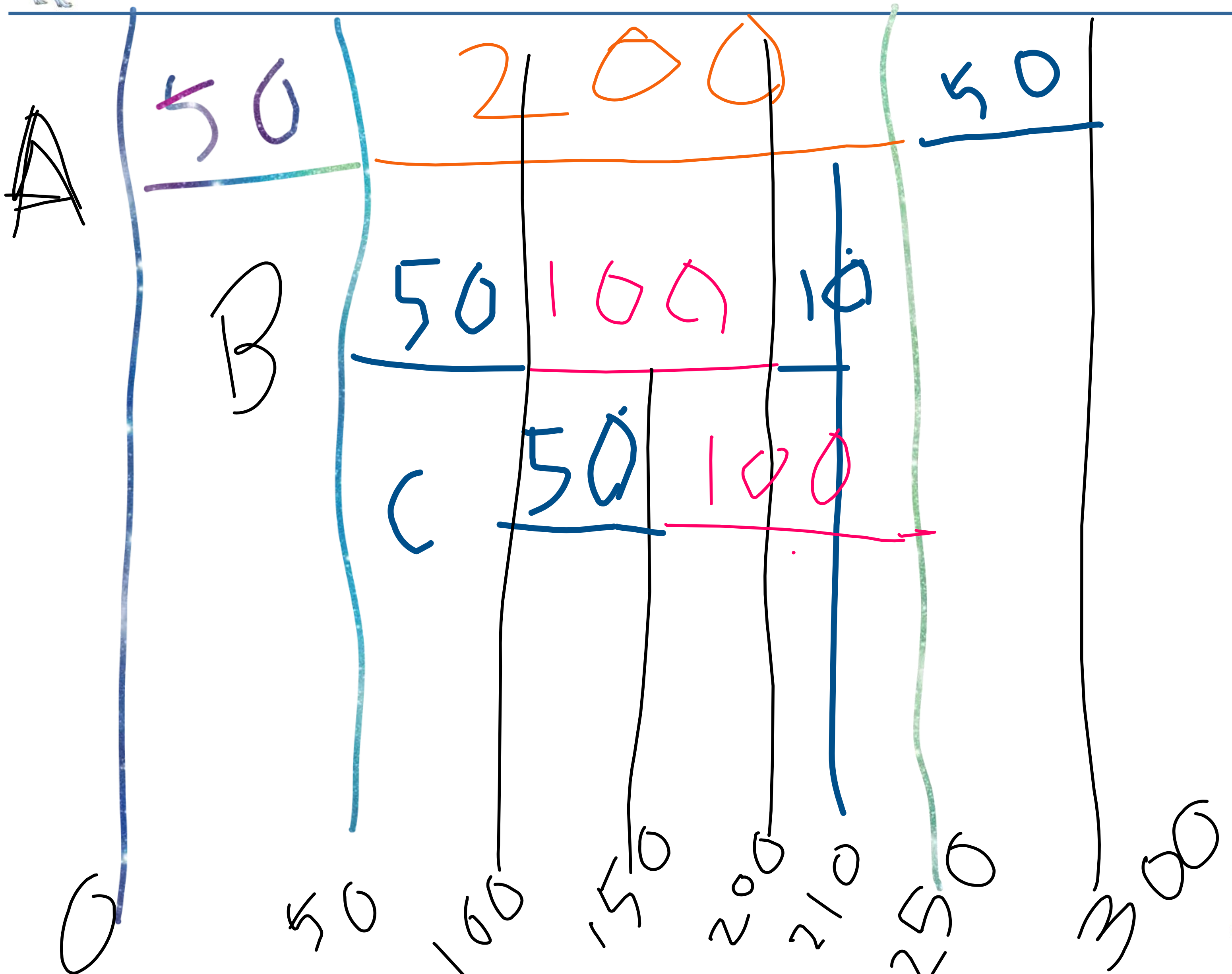
Total time = 450ms

Your turn (10 mins)

Consider three programs:

- Program A: computation (50ms), then I/O on device 1 (200ms), then computation (50ms)
- Program B: computation (50ms), then I/O on device 2 (100ms), then computation(10ms)
- Program C: computation (50ms), then I/O on device 2 (100ms)

- Monoprogramming: Total time: 610ms
- Multiprogramming: Total time: 300ms


A
50
200
50
B
50
100
10
C
50
100
0
50
100
150
200
210
250
300

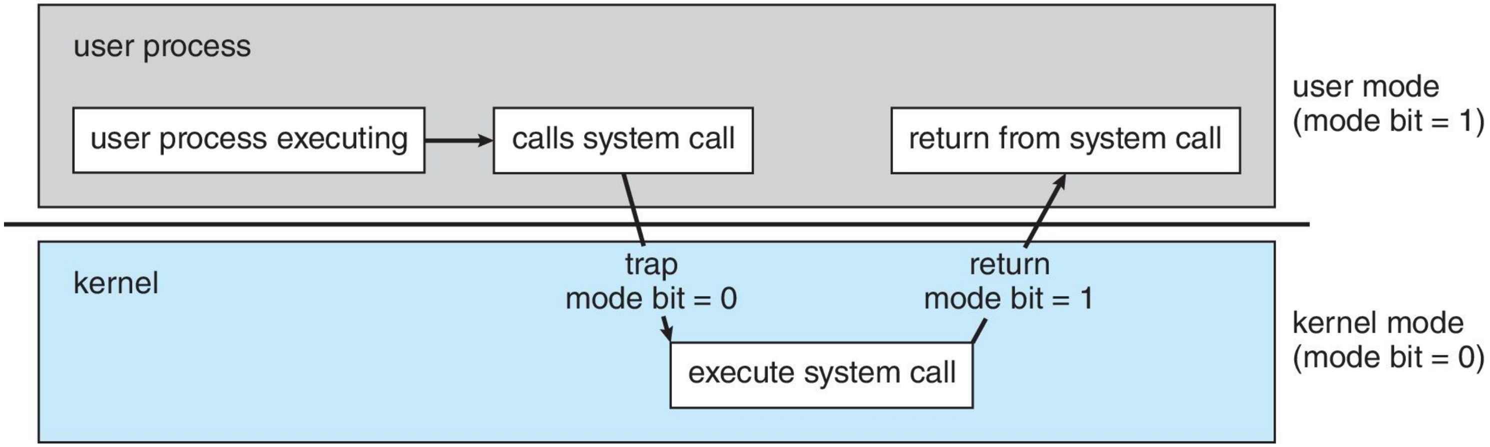
Multitasking

- A multiprogramming OS focuses on efficient use of CPU and I/O devices
- **Multitasking** (**timesharing**) is a logical extension of multiprogramming in which users can interact with each process while it is running, creating **interactive** computing
  - OS rapidly switches between users' processes to provide fast response time to users
  - Each user is given the impression that the entire computer system is dedicated to his/her use

Dual-Mode Operation

- **Dual-mode** operation allows OS to protect itself and user programs
  - Ensures that an incorrect (or malicious) program cannot cause other programs or the OS to execute incorrectly
- CPU can operate in two modes: **User mode** (1) and **kernel mode** (0)
  - **Mode bit** provided by hardware, distinguishes when system is running user code or kernel code
- Some instructions are designated as **privileged instructions**, only executable in kernel mode
  - Such instructions may cause harm to system resources (I/O devices, memory, files) and user/OS code
  - E.g., clear memory, access I/O device
- User programs execute in user mode. When a trap or interrupt occurs, OS puts CPU in kernel mode
- OS puts CPU in user mode before passing control to a user program

Transition from User to Kernel Mode



System call changes mode to kernel mode, return from call resets mode to user mode

Timer

- To prevent a process from hogging resources, a timer is set to interrupt the CPU after some time period
- A timer is implemented by the physical clock and a counter
  - OS sets the counter (privileged instruction)
  - Counter is decremented every time the clock ticks
  - When counter reaches zero, generate an interrupt
  - OS sets up timer before scheduling a process in order to regain control

Process Management

- A **process** is a program in execution; it is a unit of work in an OS
- Process needs resources to accomplish its task
  - CPU time, memory, I/O devices, files
  - Process termination requires OS to reclaim any reusable resources
- OS activities in connection with process management:
  - Creating and deleting both user and system processes
  - Suspending and resuming processes
  - Providing mechanisms for process synchronization
  - Providing mechanisms for process communication
  - Providing mechanisms for deadlock handling

Memory Management

- To execute a program:
  - All or part of the instructions must be in memory
  - All or part of the data needed by the program must be in memory
- OS keeps multiple programs in memory to improve CPU utilization and speed of computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

File-System Management

- OS provides a uniform, logical view of information storage
  - OS defines a logical storage unit – **file**
  - OS maps files onto physical media
- File-system management activities
  - Creating and deleting files and directories
  - Supporting primitives for manipulating files and directories
  - Mapping files onto secondary storage
  - Access control to determine who can access what

Secondary-Storage Management

- Secondary storage (e.g., hard disk drives, nonvolatile memory devices) used to store programs and data that do not fit in main memory
- OS activities
  - Mounting and unmounting file systems
  - Free-space management
  - Storage allocation
  - Disk scheduling
  - Disk partitioning

Cache Management

- **Caching** – information in use is copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache is smaller than storage being cached
  - Cache management an important design problem
  - Cache size and replacement policy

Characteristics of Various Types of Storage

Level	1	2	3	4	5
Name	registers	cache	main memory	solid-state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25-0.5	0.5-25	80-250	25,000-50,000	5,000,000
Bandwidth (MB/sec)	20,000-100,000	5,000-10,000	1,000-5,000	500	20-150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Protection & Security, Networking

- **Protection** – mechanism for controlling access of processes or users to resources of computer system
  - OS distinguishes between authorized and unauthorized usage of resources
- **Security** – defense of the system against external attacks
  - E.g., denial-of-service attacks, worms, viruses, identity theft, theft of service
- **Networking** – OS support network protocols (e.g., TCP/IP)