

Operating Systems

Computer and Operating Systems Overview

Yanyan Zhuang

Department of Computer Science

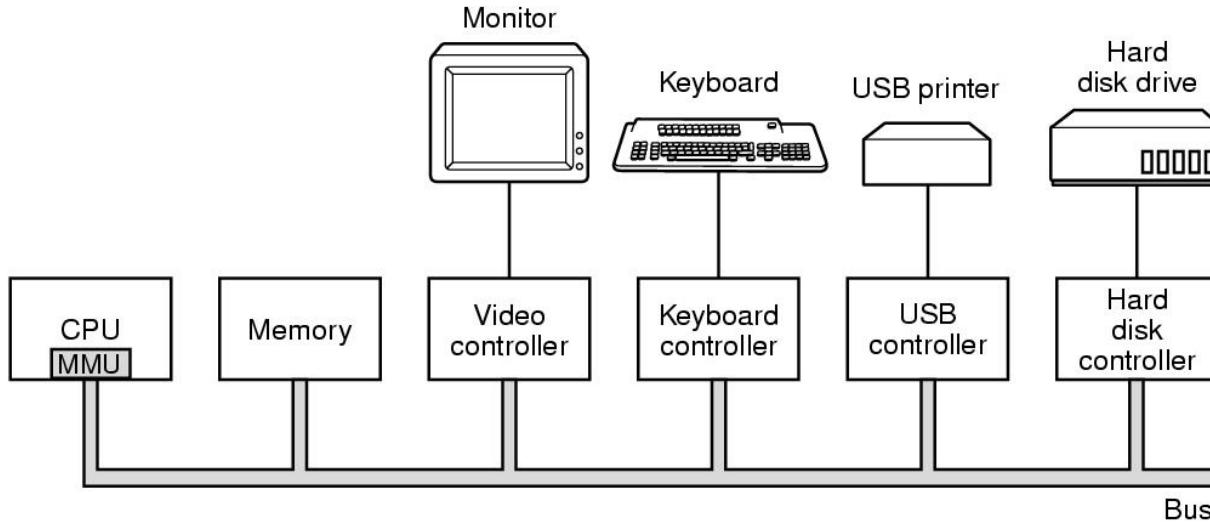
<http://www.cs.uccs.edu/~yzhuang>

Overview

- Recap of last class
 - What is an operating system?
 - Functionalities of operating systems
 - Types of operating systems
- Computer hardware review
- Operating system organization

Computer Hardware Review

- Basic components of a simple personal computer

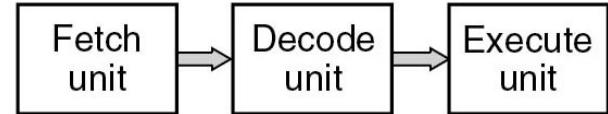


- **CPU:** data processing
- **Memory:** volatile data storage
- **Disk:** persistent data storage
- **NIC:** inter-machine communication
- **Bus:** intra-machine communication

Central Processing Unit (CPU)

- Components

- Arithmetic Logic Unit (ALU)
- Control Unit (CU)
- Fetch, decode, execute

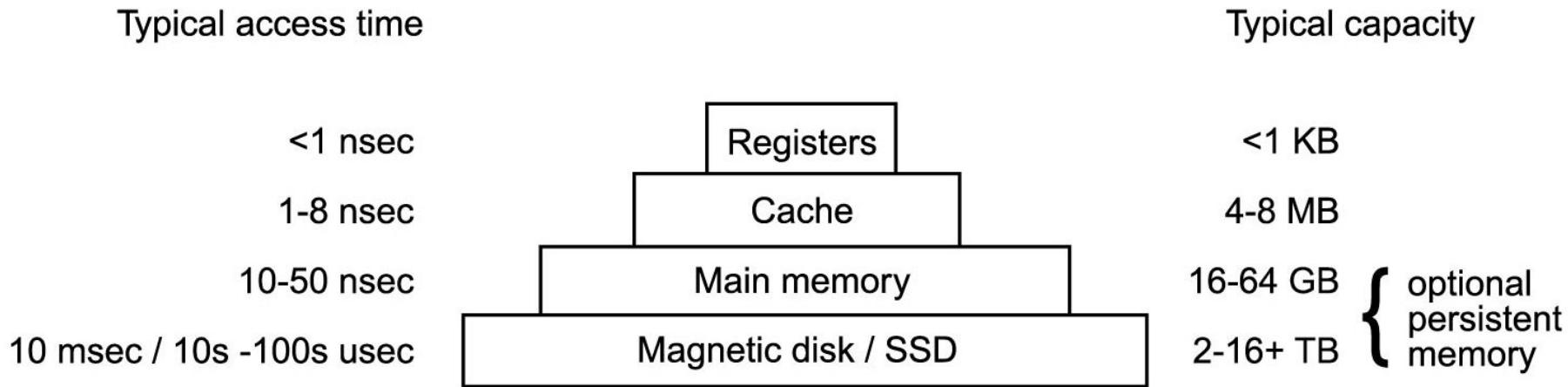


- Registers

- General-purpose registers: EAX, EBX ...
- Special-purpose registers: PC, SP, ...

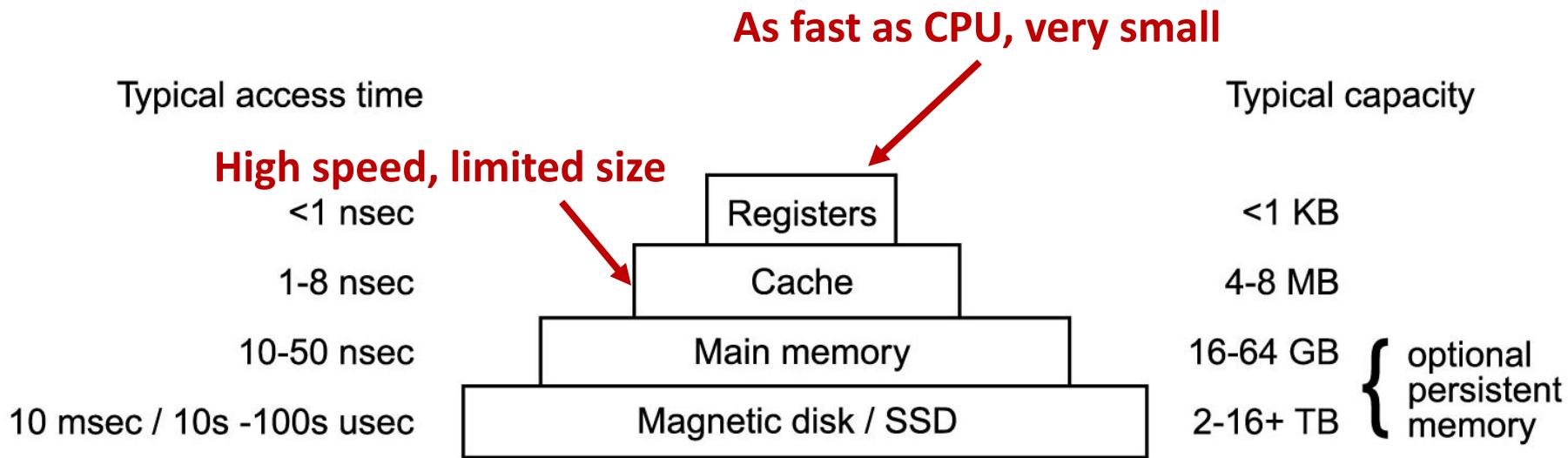
IF	ID	EX	MEM	WB				
i	IF	ID	EX	MEM	WB			
	t	IF	ID	EX	MEM	WB		
		IF	ID	EX	MEM	WB		
			IF	ID	EX	MEM	WB	

Memory



A typical memory hierarchy. The numbers are very rough approximations.

Memory



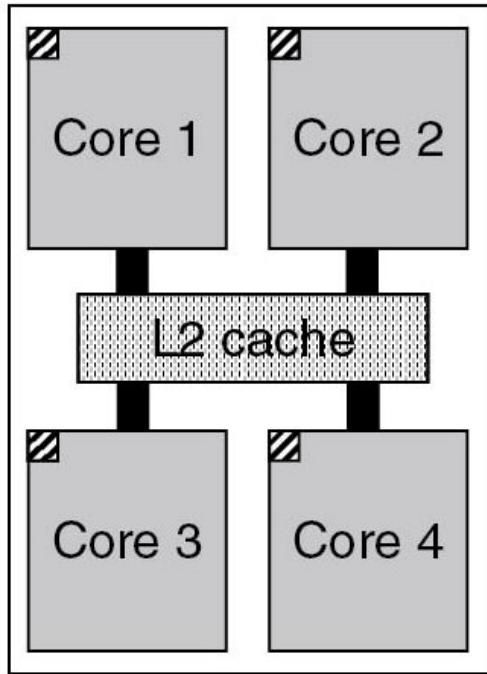
A typical memory hierarchy. The numbers are very rough approximations.

Cache

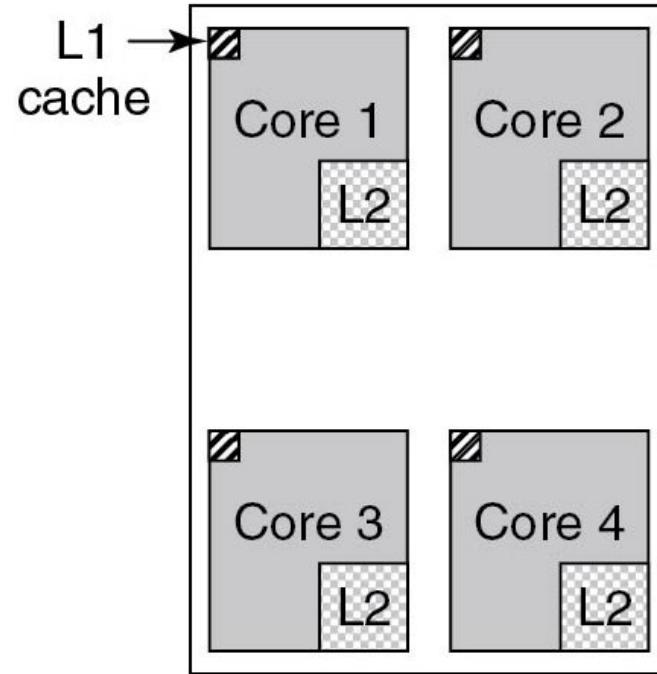
- Different types of cache
 - L1 cache: inside CPU and feeds decoded instructions into CPU's execution engine (no delay)
 - L2 cache: inside or outside CPU, holds megabytes of recently used memory words (1-2 clock cycles)

Multi-Core Processors with Cache

- Multiple CPUs



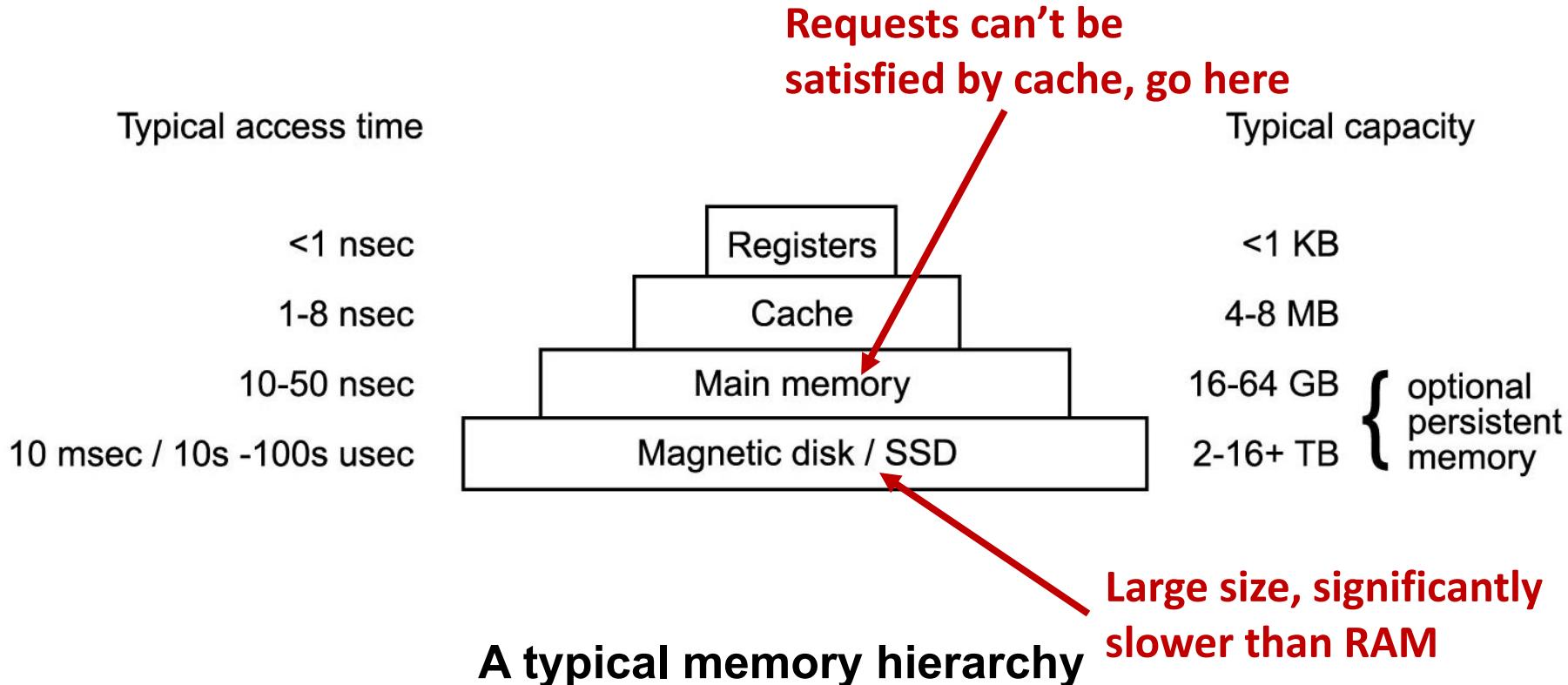
(a)



(b)

(a) A quad-core chip with a shared L2 cache. (b) A quad-core chip with separate L2 caches.

Memory

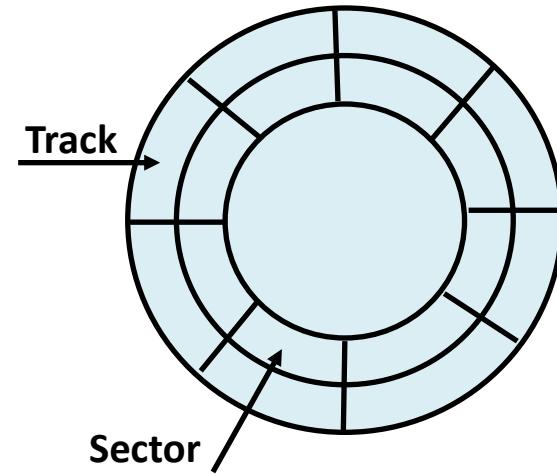
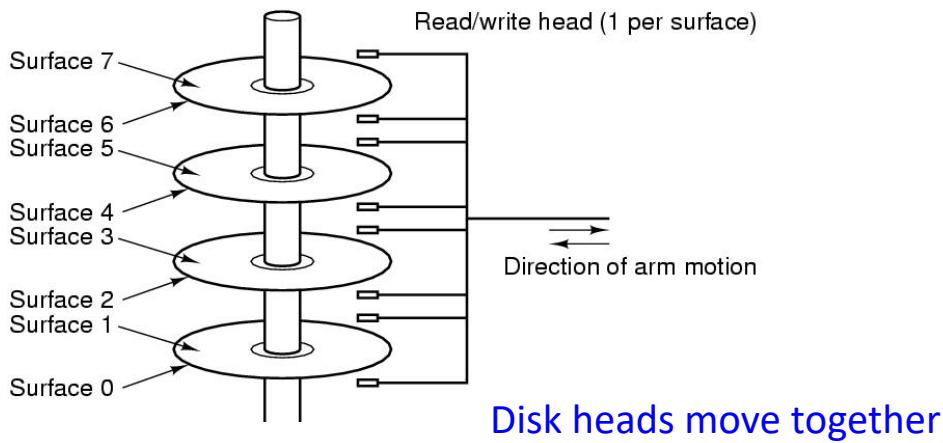


Memory Management

- Multiprogramming
 - Accommodate multiple processes in main memory
 - How to *protect* the programs from one another and the kernel from them all?
 - Processes should not reference memory locations in another process without permission
 - What if all processes' memory requirement exceeds the physical memory?

Virtual memory address \longleftrightarrow Physical memory address

Hard (Magnetic) Disks



- A stack of platters, a surface with a magnetic coating
- Disk head: each side of a platter has separate disk head
- Each surface is divided into tracks and sectors
 - 500 to 2,000 tracks per surface
 - 32 to 128 sectors per track

Magnetic Disk Characteristics

- Read/write data is a **three-stage process**:
 - Seek time: position the arm over the proper track
 - Rotational latency: wait for the desired sector to rotate under the read/write head
 - seek time is the longest
 - Transfer time: transfer a block of bits (sector) under the read-write head
- Average seek time as reported by the industry:
 - Typically in the range of 8 ms to 15 ms
- Due to locality of disk reference
 - Actual average seek time may only be 25% to 33% of the advertised number

Solid State Drives (SSD)

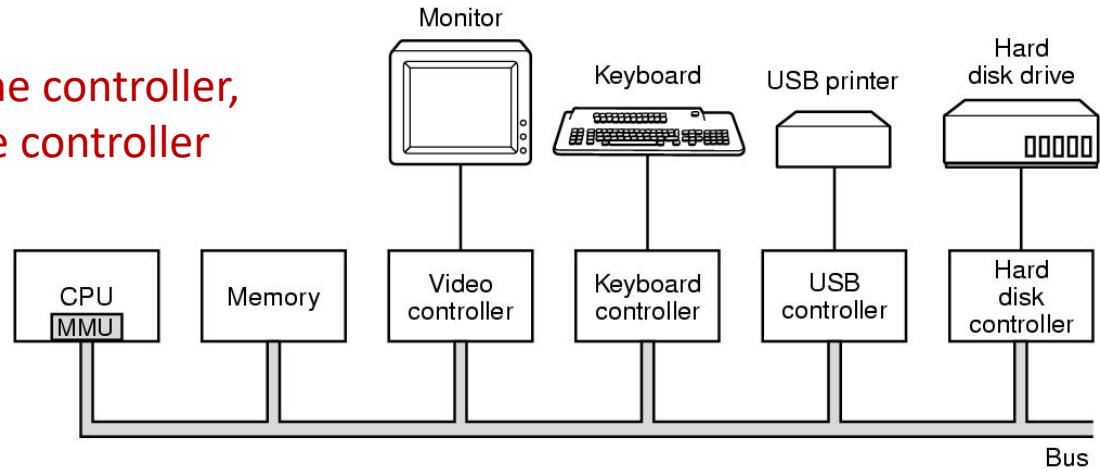
- No moving parts, data in electronic (flash) memory
- Much faster than magnetic disks



I/O Devices

- Device controller
 - Chips that control device, provide a simple interface to OS
 - Accepts commands from OS, and carries them out
- Device driver
 - The software that talks to a controller, giving it commands and accepting responses (one for each type of device controller)

Actual device is hidden behind the controller,
all that OS sees is interface to the controller



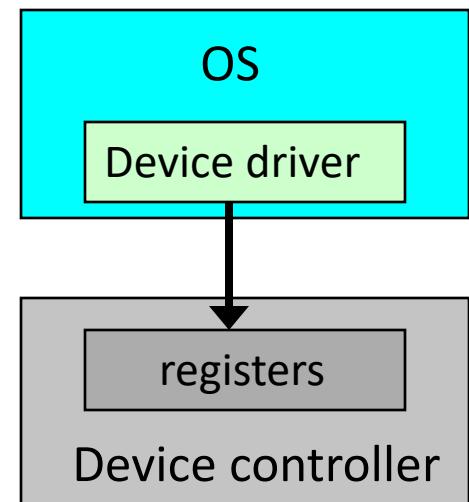
I/O Devices

- Device controller
 - Chips that control device, provide a simple interface to OS
 - Accepts commands from OS, and carries them out
- Device driver
 - The software that talks to a controller, giving it commands and accepting responses (one for each type of device controller)

Device driver communicate with controller via registers:

Every controller has a small number of registers:

A disk controller might have registers for specifying disk address, direction (r/w), etc.



Interactions between OS and I/O Devices

- The OS gives commands to the I/O devices
- The I/O device **notifies** the OS when the I/O device has completed an operation or has encountered an error
- Data is transferred between memory and an I/O device

How I/O Devices Notify the OS?

- Hardware device events
 - Hardware devices produce events at times and in patterns not known in advance
 - ▶ Keyboard presses, incoming network packets, mouse movements
- How to make information generated by these events available to running applications?
 - Three methods: polling, interrupt, DMA

How I/O Devices Notify the OS?

- Polling (busy waiting)
 - The I/O device put information in a status register
 - The OS periodically check the status register

Simple design: OS periodically checks device

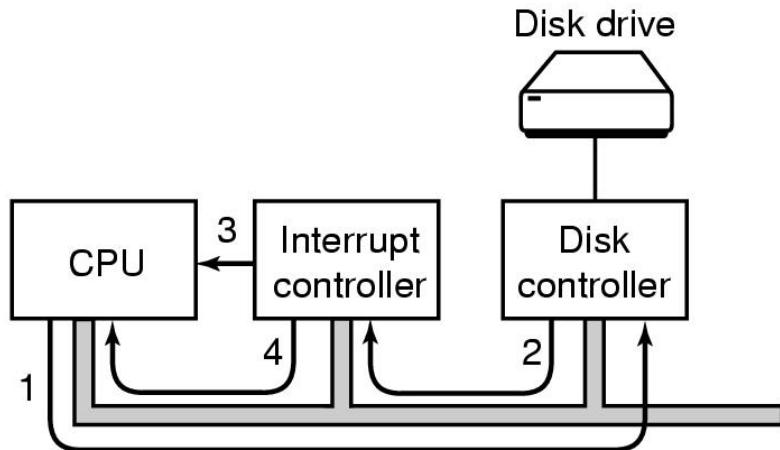
Disadvantage:

- (1) Most time device has no input data – waste CPU time
- (2) Latency – device input must wait for polling interval

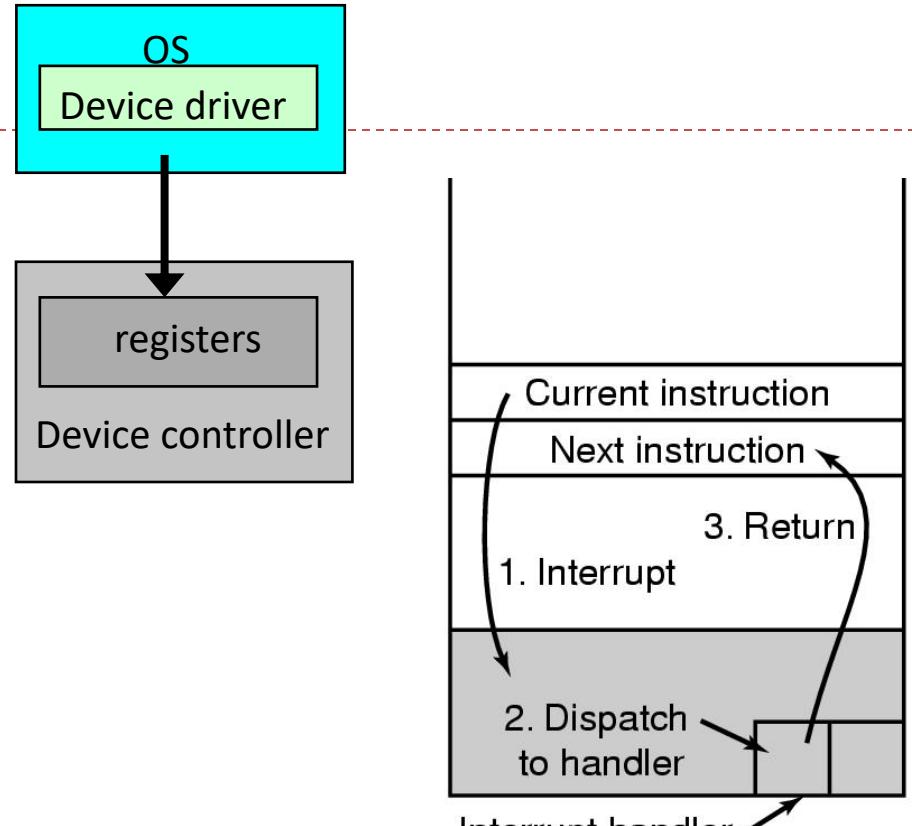
How I/O Devices Notify the OS?

- Polling (busy waiting)
 - The I/O device put information in a status register
 - The OS periodically check the status register
- Interrupt
 - Whenever an I/O device needs attention from the processor, it interrupts the processor from what it is currently doing
 - HW signals the OS when events occur
 - OS switches away from running process, handle the event
 - More responsive than polling, but complex implementation

I/O Interrupt



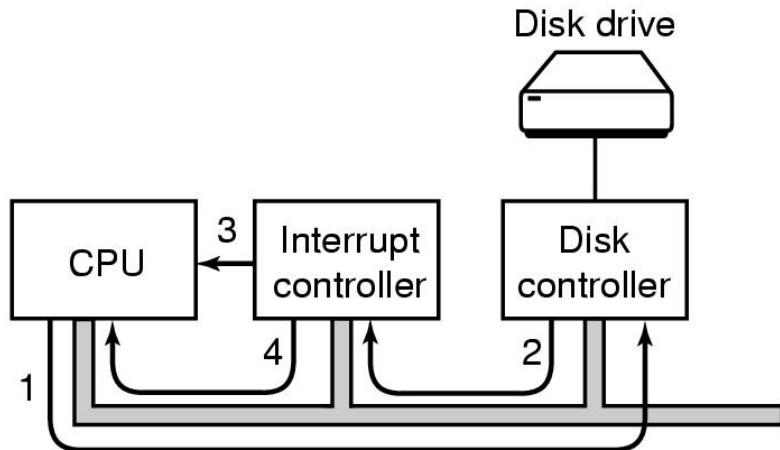
(a)



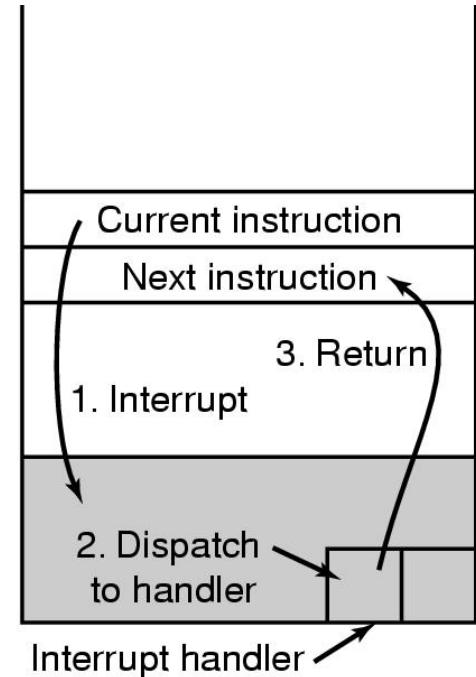
(b)

1. CPU writes cmds in to device registers
2. The device signals interrupt controller
3. Interrupt controller informs CPU
4. CPU accepts the interrupt and triggers the service routine

I/O Interrupt



(a)



(b)

1. Save current instruction info, switch to kernel mode
2. Find interrupt handler (service routine) for device
3. Interrupt handler finishes: return to user program

Interrupts

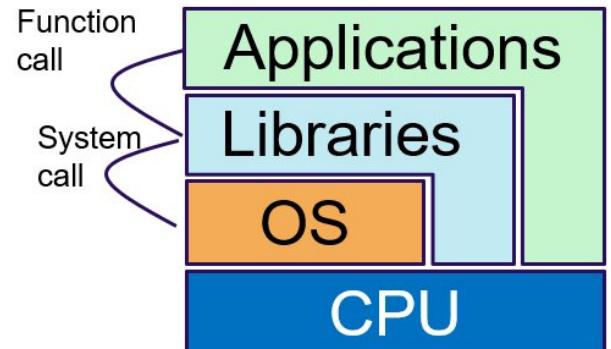
- I/O Interrupts
 - Complex
 - ▶ A suspension of a process caused by an external event
 - ▶ Performed in a way that the process can be resumed
 - Improves processing efficiency (compare to polling)
- Types of interrupts
 - I/O, Timer

How I/O Devices Notify the OS?

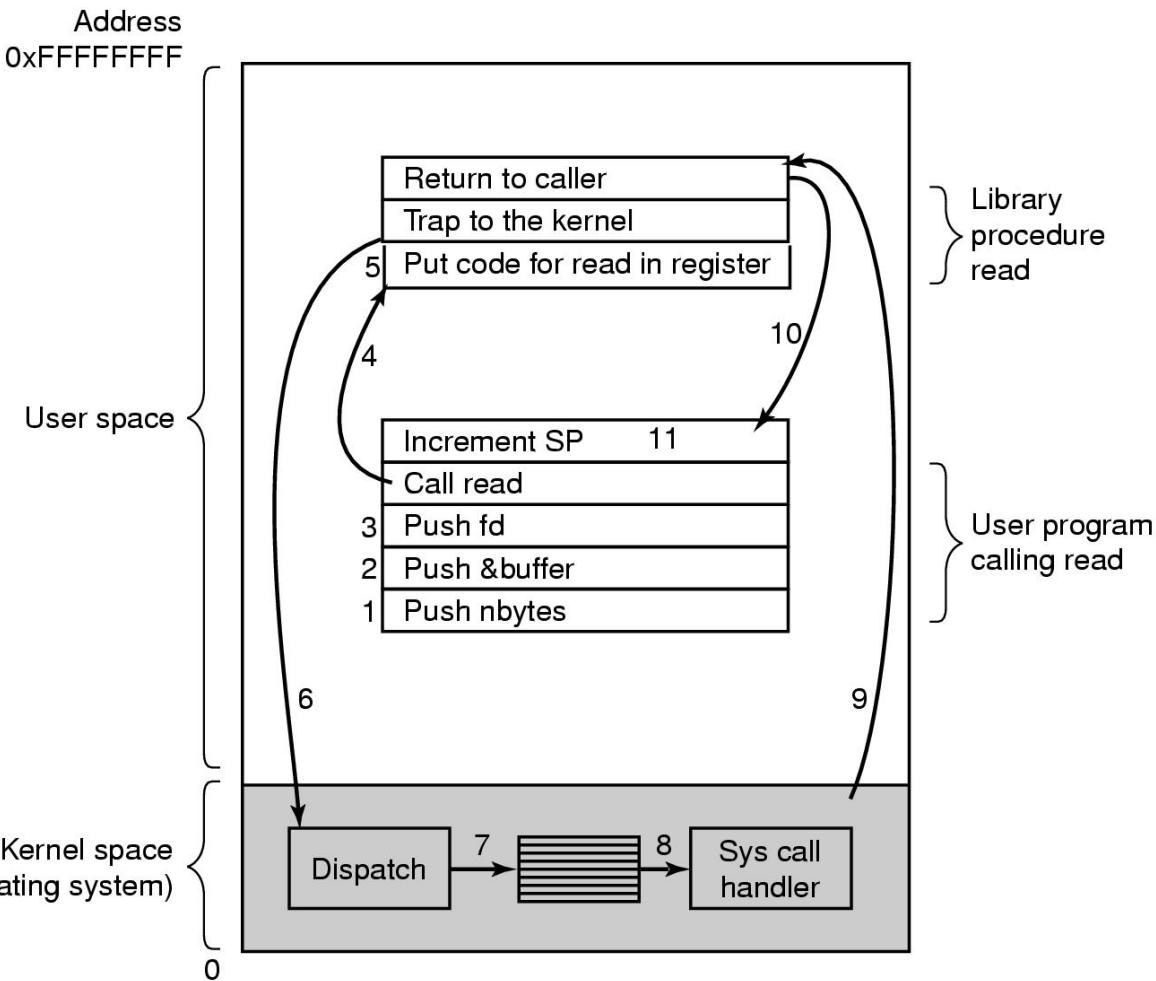
- Polling
 - The I/O device put information in a status register
 - The OS periodically check the status register
- Interrupt
 - Whenever an I/O device needs attention from the processor, it interrupts the processor from what it is currently doing
- DMA
 - Direct memory access: delegate I/O responsibility from CPU (more in Chapter 5)

System Calls

- (Lec1): transition between user/kernel mode
 - Hardware interrupt – HW device requests OS services
 - System calls (aka trap) – user program requests OS services
 - ▶ The interface an OS offers to apps to request services (file, network)
 - ▶ A user program that needs service execute a **trap** instruction to transfer control to OS kernel
 - ▶ OS inspects parameters, carries out system call, returns control to the instruction following the system call
 - Exception – error handling



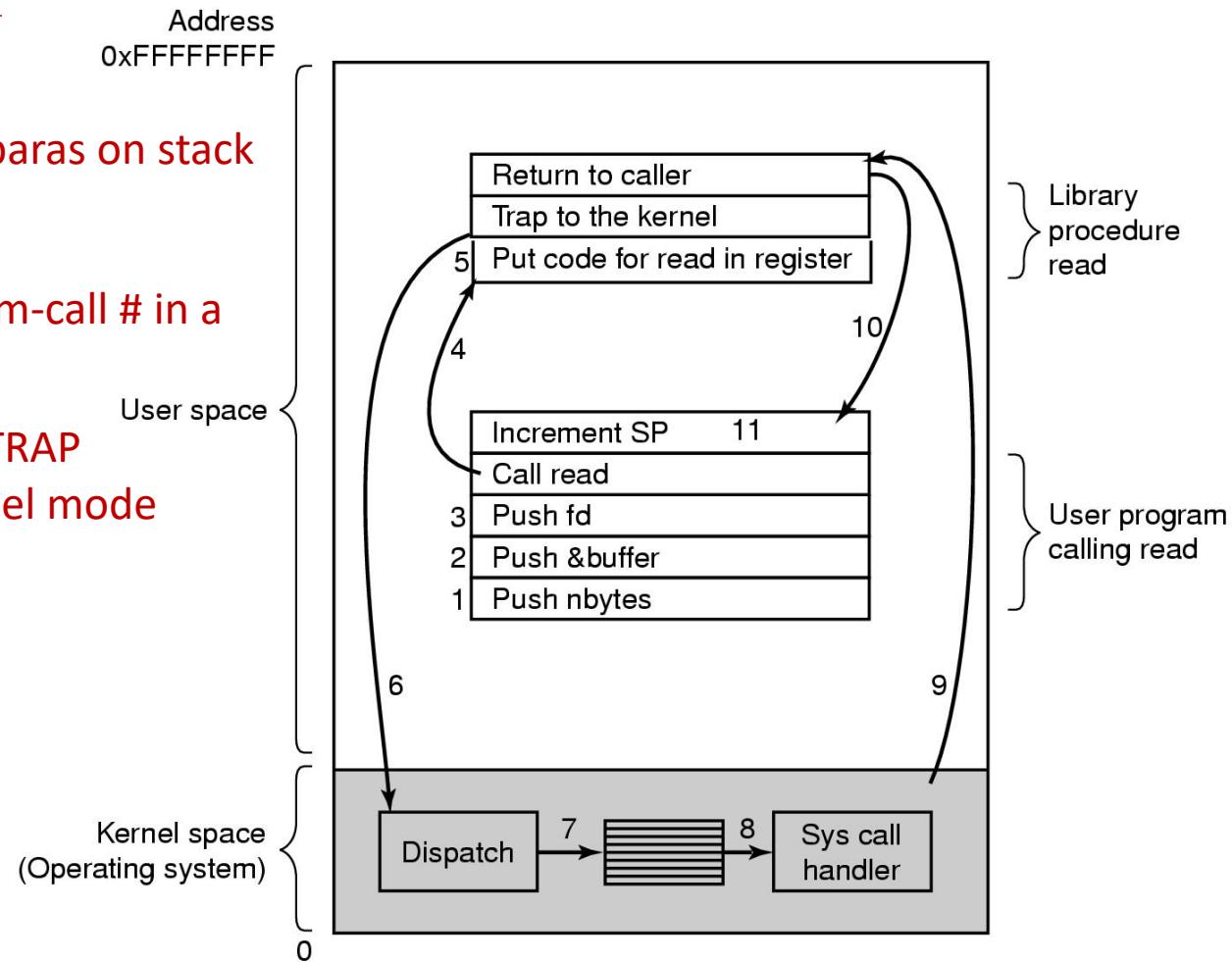
System Calls



There are 11 steps in making the system call `read(fd, buffer, nbytes)`

System Calls

- 1 – 3: calling program pushes paras on stack
- 4: calls the library procedure
- 5: library procedure puts system-call # in a register
- 6: library procedure executes TRAP instruction: user mode → kernel mode



There are 11 steps in making the system call `read(fd, buffer, nbytes)`

System Calls

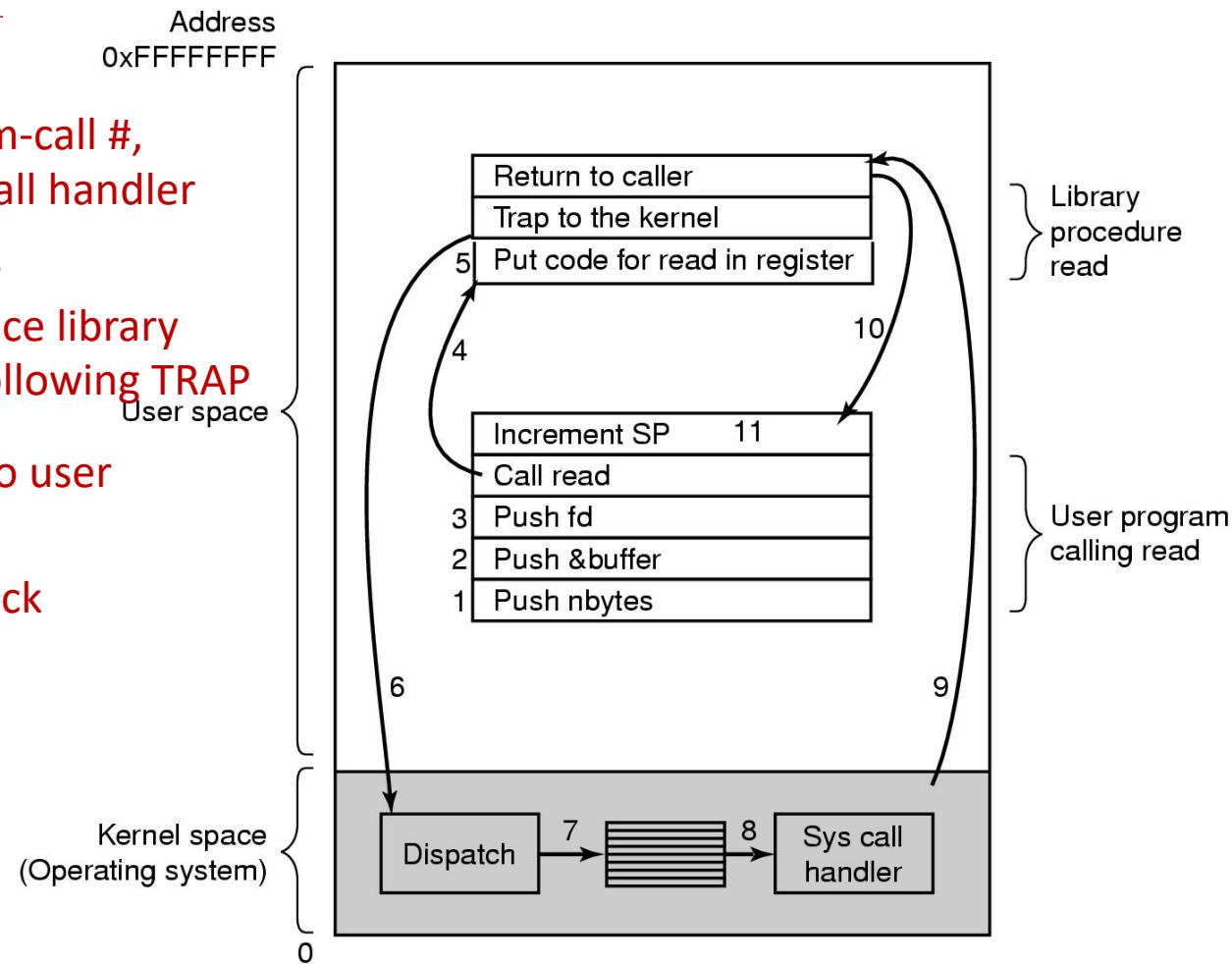
7: kernel code examines system-call #,
dispatches to correct system-call handler

8: the system-call handler runs

9: control returned to user-space library
procedure at the instruction following TRAP

10: library procedure returns to user
program

11: user program cleans up stack

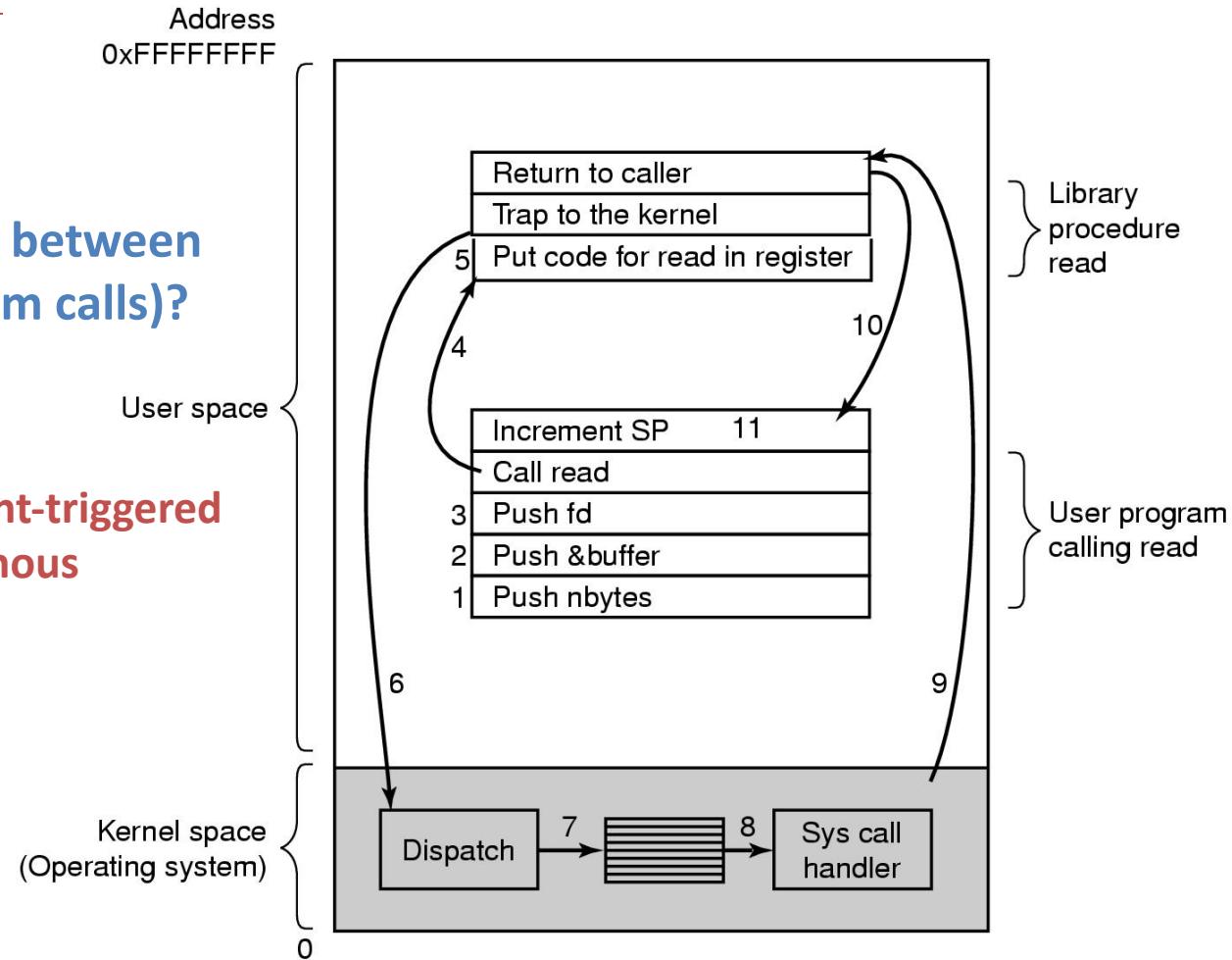


There are 11 steps in making the system call `read(fd, buffer, nbytes)`

System Calls

What is the key difference between
interrupts and traps (system calls)?

- program-triggered vs. event-triggered
- synchronous vs. asynchronous



There are 11 steps in making the system call `read(fd, buffer, nbytes)`

Operating System Components

- Process management
- Memory management
- File and storage
- Networking

Process Management (Chapter 2)

- Process: a fundamental OS concept
 - Memory address space
 - Some set of registers (program counter, stack pointer)
 - ...
- OS responsibilities for process management
 - Process creation and deletion
 - Process scheduling, suspension, and resumption
 - Inter-process communication and synchronization

Memory Management (Chapter 3)

- Memory
 - A large array of addressable words and bytes
- OS responsibilities for memory management
 - Allocate and de-allocate memory space
 - Keep memory space efficiently utilized
 - Keep track of which part of memory are used and by whom

Design goals: *transparency* and *efficiency*

File and Storage Management (Chapter 4)

- A file is a collection of data usually stored on disk with a unique name
 - Programs, data
 - Devices (UNIX & Linux)
- OS responsibilities for file management
 - Organize directories and files
 - Map files onto disk
- OS responsibilities for disk management
 - Disk space management
 - Disk scheduling



Additional Readings and Practice

- Section 1.6 and try the following Linux commands
 - ▶ who, uname, ls, cat, cp, rm, mv, cd, mkdir, touch, chmod
 - ▶ Use “man” to see the manual of above commands
- Write a C program with an output to the screen
 - ▶ strace -o trace.txt ./YOUR_PROG
 - ▶ See the system calls triggered (execve, write, ...)
 - ▶ <http://unix.stackexchange.com/questions/797/understanding-the-linux-kernel-source>



Summary

- Computer hardware
 - Time-sharing: CPU
 - Space-sharing: memory, disk
- OS components
 - Process management
 - Memory management
 - File and storage management