

# CS370 Operating Systems

Colorado State University

Yashwant K Malaiya

Fall 2016 Lecture 2



## Slides based on

- Text by Silberschatz, Galvin, Gagne
- Various sources

# Grading breakdown

- Assignments: 30%
  - Programming & written
- Quizzes 10%
  - On-line, in-class
- Mid Term: 20% 7-Oct Fri
- Project: 15%
- Final exam: 25%

# Help me help you

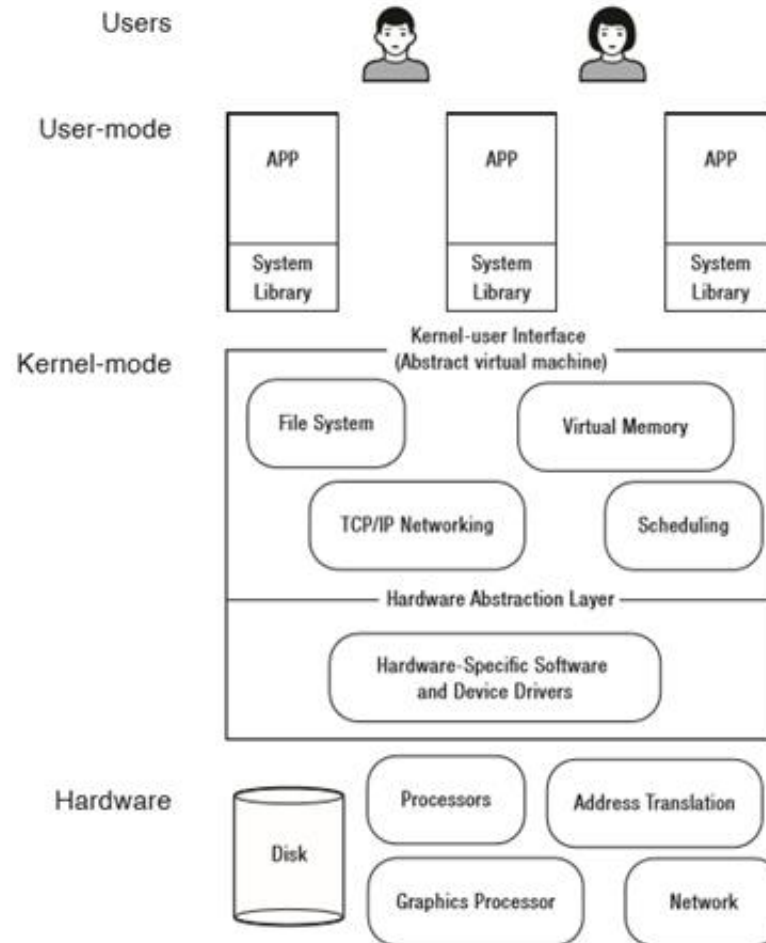
- Surveys at the end of a class: last 5 minutes
- You will provide a list of
  - 2 concepts you followed clearly
  - 2 concepts you had problems keeping up
  - Problem areas for the majority of the class will be addressed in the next class
- Electronic devices
  - Permitted only in the last row, with the pledge
  - not distract others
  - use it only for class related use
  - turn off wireless

# Help Session: C Programming

- Next 1-2 days
  - Will be recorded
  - C, pointers, dynamic memory allocation
  - Needed for upcoming programming assignment

# What is an Operating System?

# What is an Operating System?



# What is an Operating System?

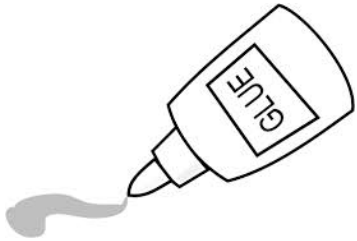


- Referee
  - Manage sharing of resources, Protection, Isolation
    - Resource allocation, isolation, communication



- Illusionist
  - Provide clean, easy to use abstractions of physical resources
    - Infinite memory, dedicated machine
    - Higher level objects: files, users, messages
    - Masking limitations, virtualization

- Glue
  - Common services
    - Storage, Window system, Networking
    - Sharing, Authorization
    - Look and feel



# Short History of Operating Systems

- One application at a time
  - Had complete control of hardware
- Batch systems
  - Keep CPU busy by having a queue of jobs
  - OS would load next job while current one runs
- Multiple users on computer at same time
  - Multiprogramming: run multiple programs at seemingly at the “same time”
- Multiple processors in the same computer



1960s  
80286  
(1984)

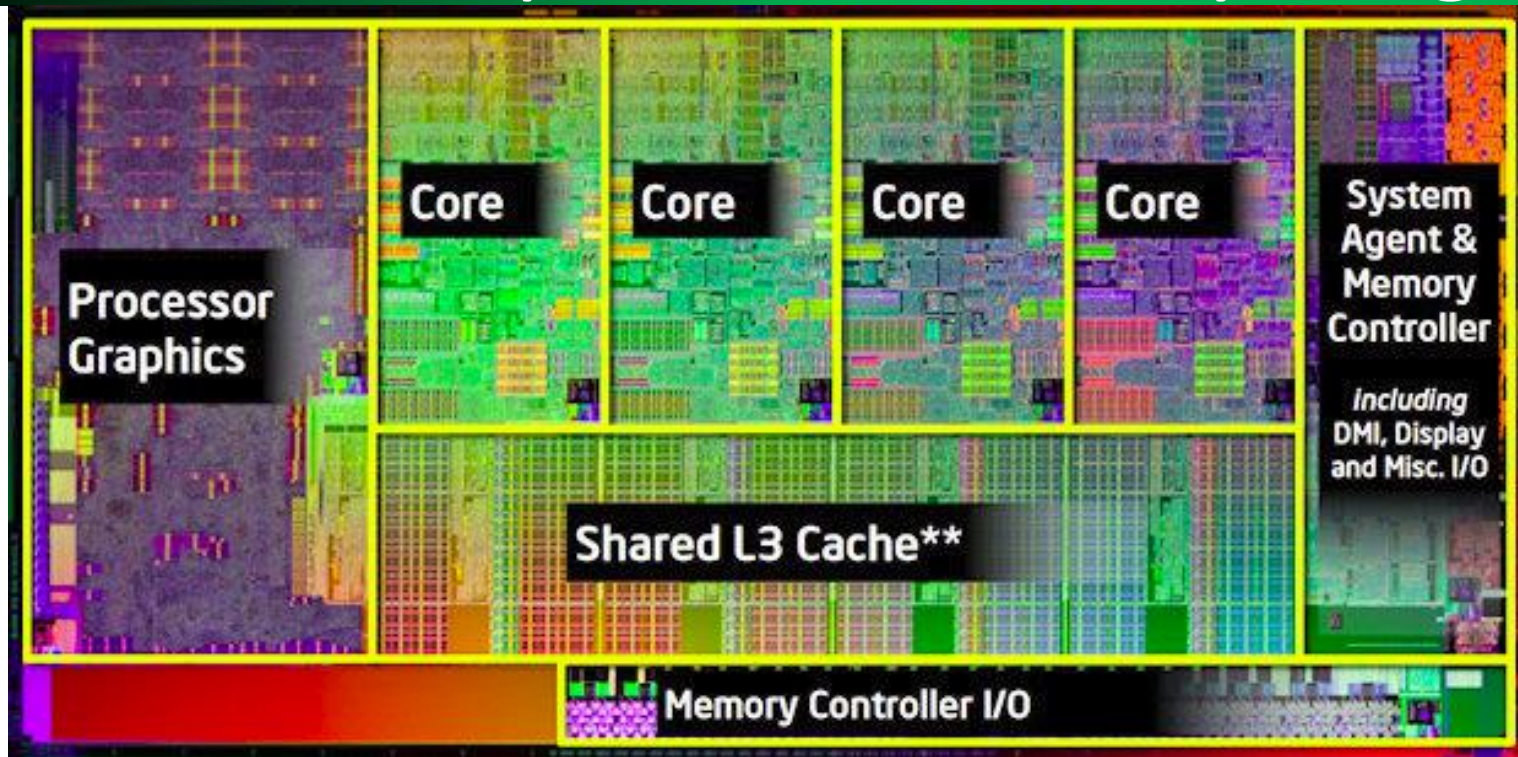


Dual  
core  
2004



# Systems

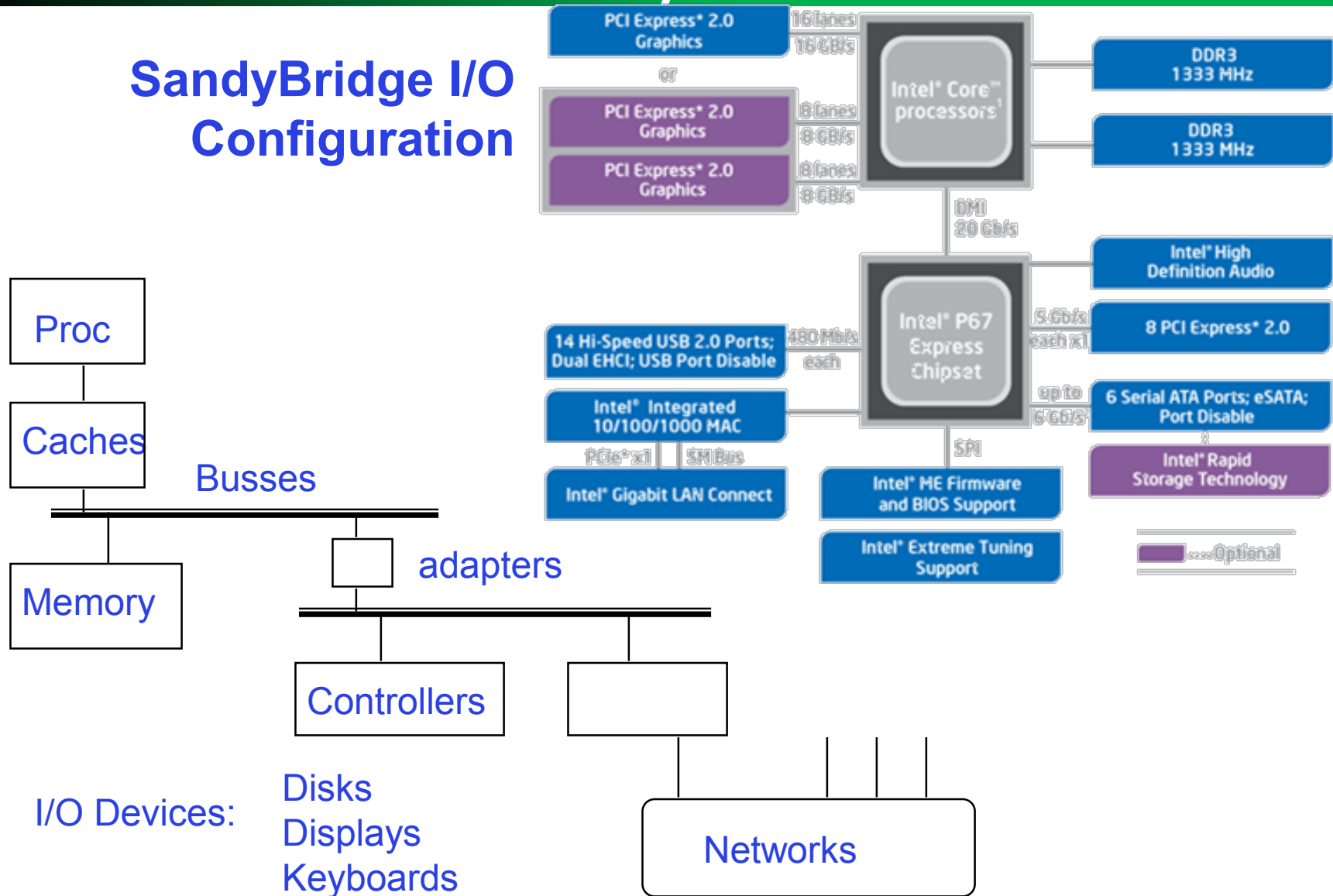
# A Modern processor: SandyBridge



- Package: LGA 1155
  - 1155 pins
  - 95W design envelope
- Cache:
  - L1: 32K Inst, 32K Data (3 clock access)
  - L2: 256K (8 clock access)
  - Shared L3: 3MB – 20MB (not out yet)
- Transistor count:
  - 504 Million (2 cores, 3MB L3)
  - 2.27 Billion (8 cores, 20MB L3)
- Note that ring bus is on high metal layers – above the Shared L3 Cache

# Functionality comes with

## SandyBridge I/O Configuration



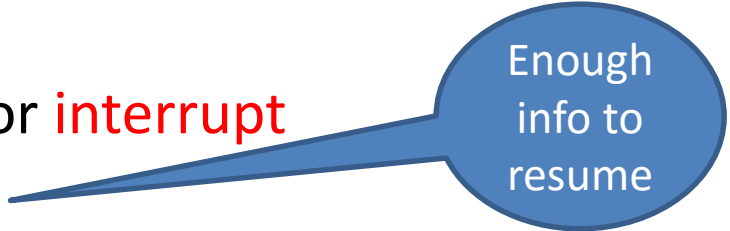
# One Processor One program View

Early processors, LC-3 is an example

- Instructions and data fetched from Main Memory using a program counter (**PC**)
- Traps and Subroutines
  - Obtaining address to branch to, and coming back
  - Using **Stack Frames** for holding
    - Prior PC, FP
    - Arguments and local variables
- Dynamic memory allocation and **heap**
- Global data

# One Processor One program View

- External devices: disk, network, screen, keyboard etc.
- Device interface: Status and data registers
- User and Supervisor modes for processor
- I/O
  - Device drivers can use polling or interrupt
  - Interrupts need *context switch*
  - I/O done in supervisor mode
  - System calls invoke device drivers



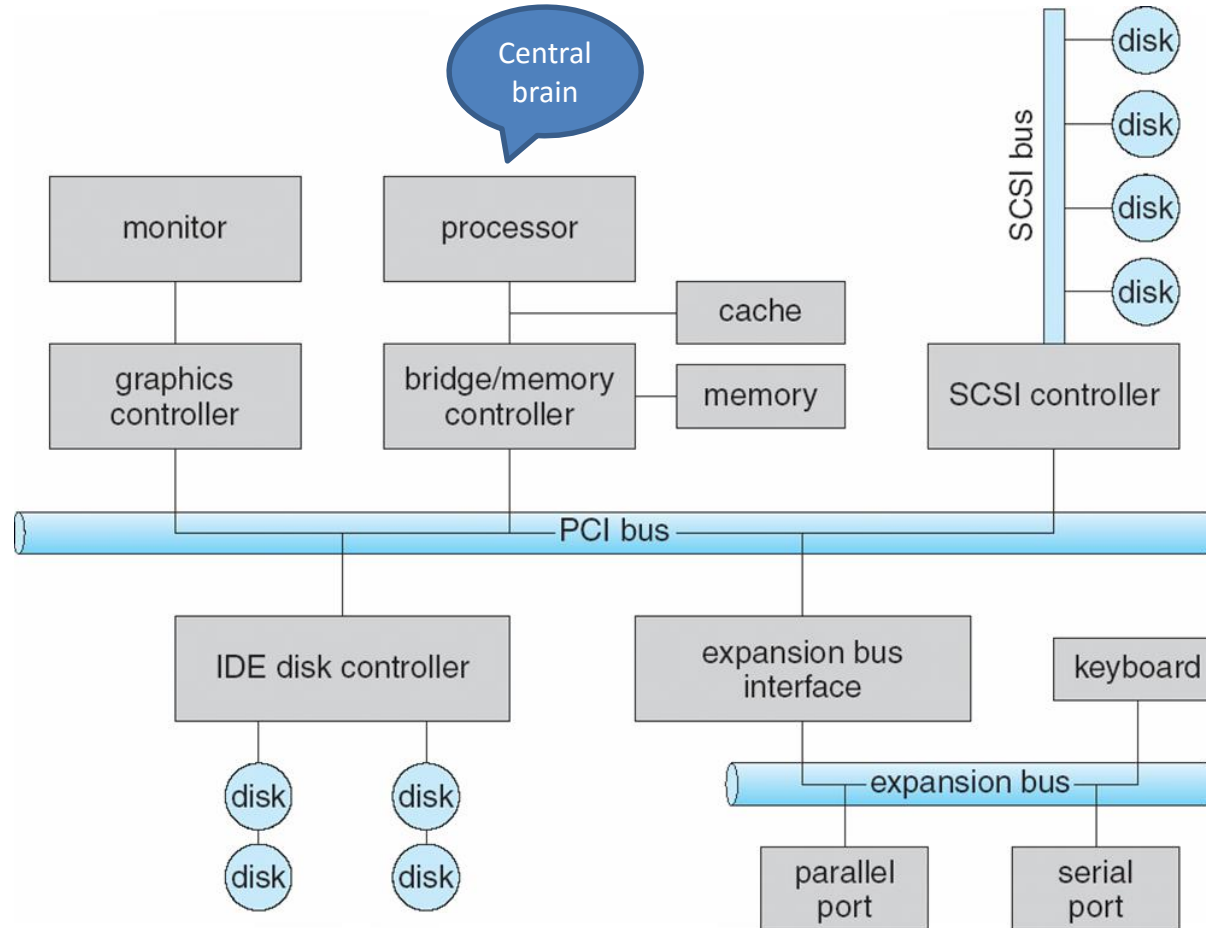
Enough  
info to  
resume

# What introductory texts don't include

- No cache
- Direct memory access (DMA) between Main Memory and Disk (or network etc)
  - Transfer by blocks at a time
- Neglecting the fact that memory access slower than register access
- Letting program run *concurrently* (Multiprogramming) or with many threads
- Multiple processors in the system (like in Multicore)

# System I/O

# System I/O (Chap 13)





# I/O Hardware (Cont.)

- I/O Devices usually have registers where device driver places commands, addresses, and data
  - Data-in register, data-out register, status register, control register
  - Typically 1-4 bytes, or FIFO buffer
- Devices have addresses, used by
  - Direct I/O instructions
  - Memory-mapped I/O
    - Device data and command registers mapped to processor address space

# Polling vs Interrupt

- Polling: handled by software
- Interrupts: polling is done by hardware

# Polling

## ■ For each byte of I/O

1. Read **busy bit** from status register until 0
2. Host sets read or write bit and if write copies data into data-out register
3. Host sets command-ready bit
4. Controller sets busy bit, executes transfer
5. Controller clears busy bit, error bit, command-ready bit when transfer done

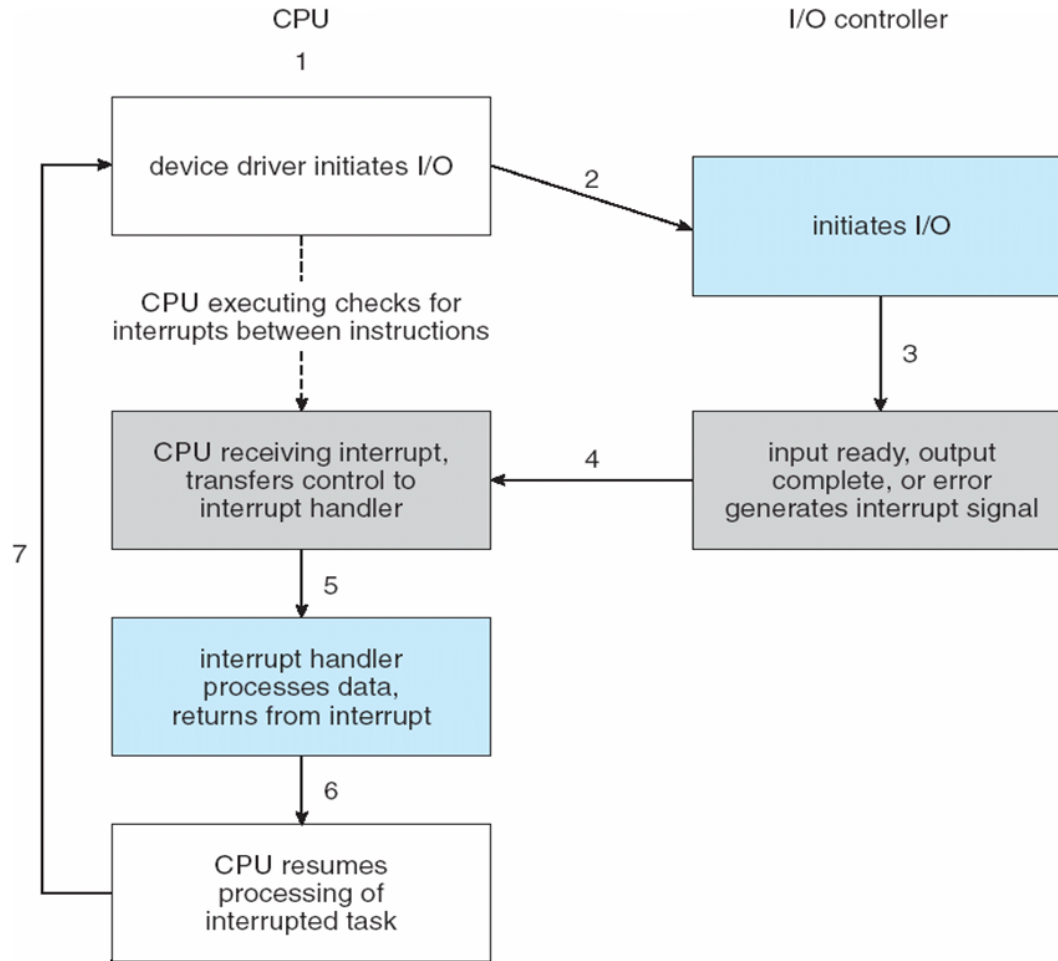
## ■ Step 1 is **busy-wait** cycle to wait for I/O from device

- Reasonable if device is fast
  - But inefficient if device slow
  - CPU switches to other tasks?
    - ▶ But if miss a cycle data overwritten / lost

# Interrupts

- Polling is slow
- Interrupts used in practice
- 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

# Interrupt-Driven I/O Cycle



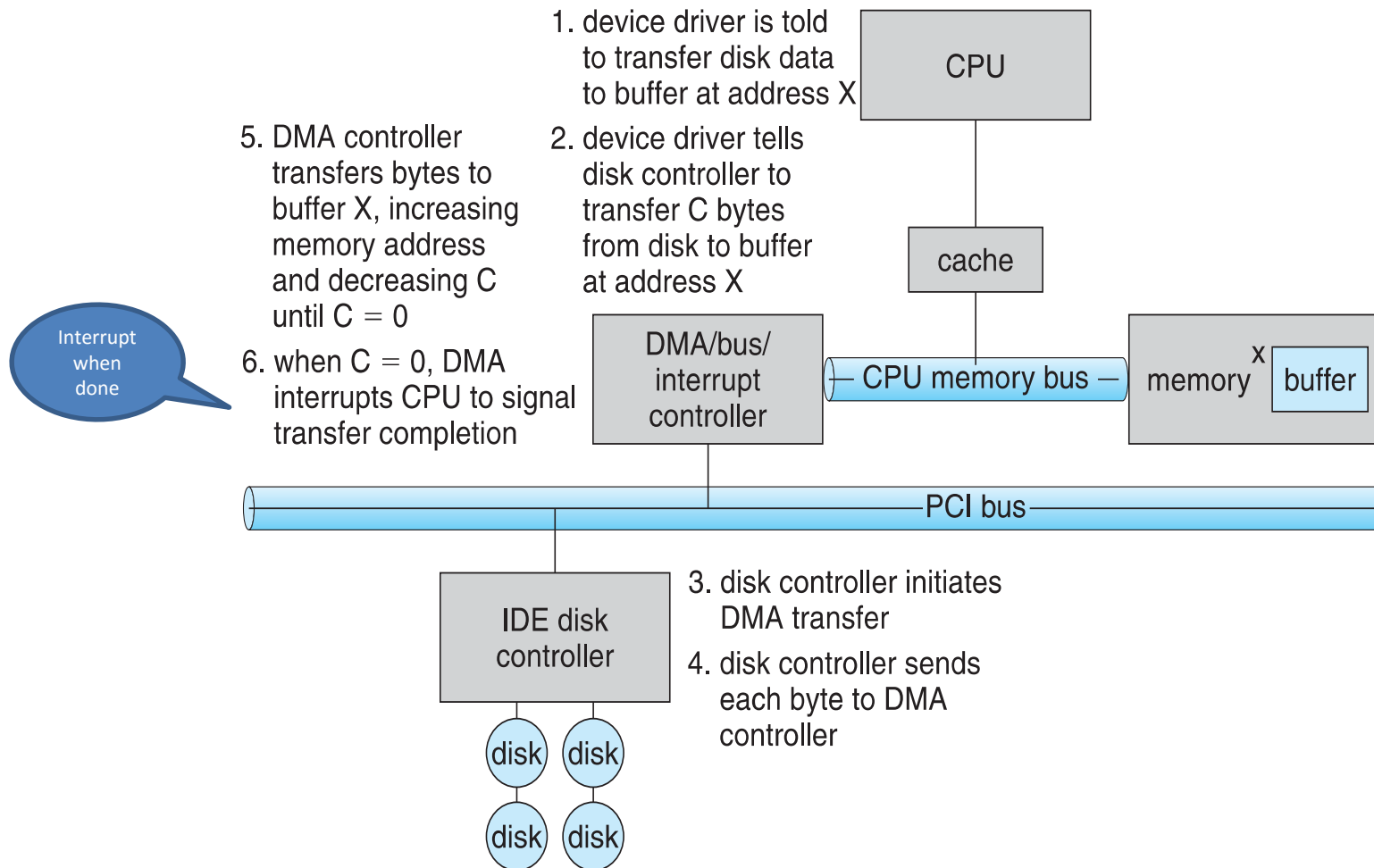
# Interrupts (Cont.)

- Interrupt mechanism also used for **exceptions**
  - Terminate process, crash system due to hardware error
  - Page fault executes when memory access error
  - OS causes switch to another process
  - System call executes via **trap** to trigger kernel to execute request

# Direct Memory Access

- for movement of a block of data
  - To/from disk, network etc.
- Requires **DMA** controller
- Bypasses CPU to transfer data directly between I/O device and memory
- OS writes DMA command block into memory
  - Source and destination addresses
  - Read or write mode
  - Count of bytes
  - Writes location of command block to DMA controller
  - Bus mastering of DMA controller – grabs bus from CPU
    - **Cycle stealing** from CPU but still much more efficient
  - When done, interrupts to signal completion

# Six Step Process to Perform DMA Transfer

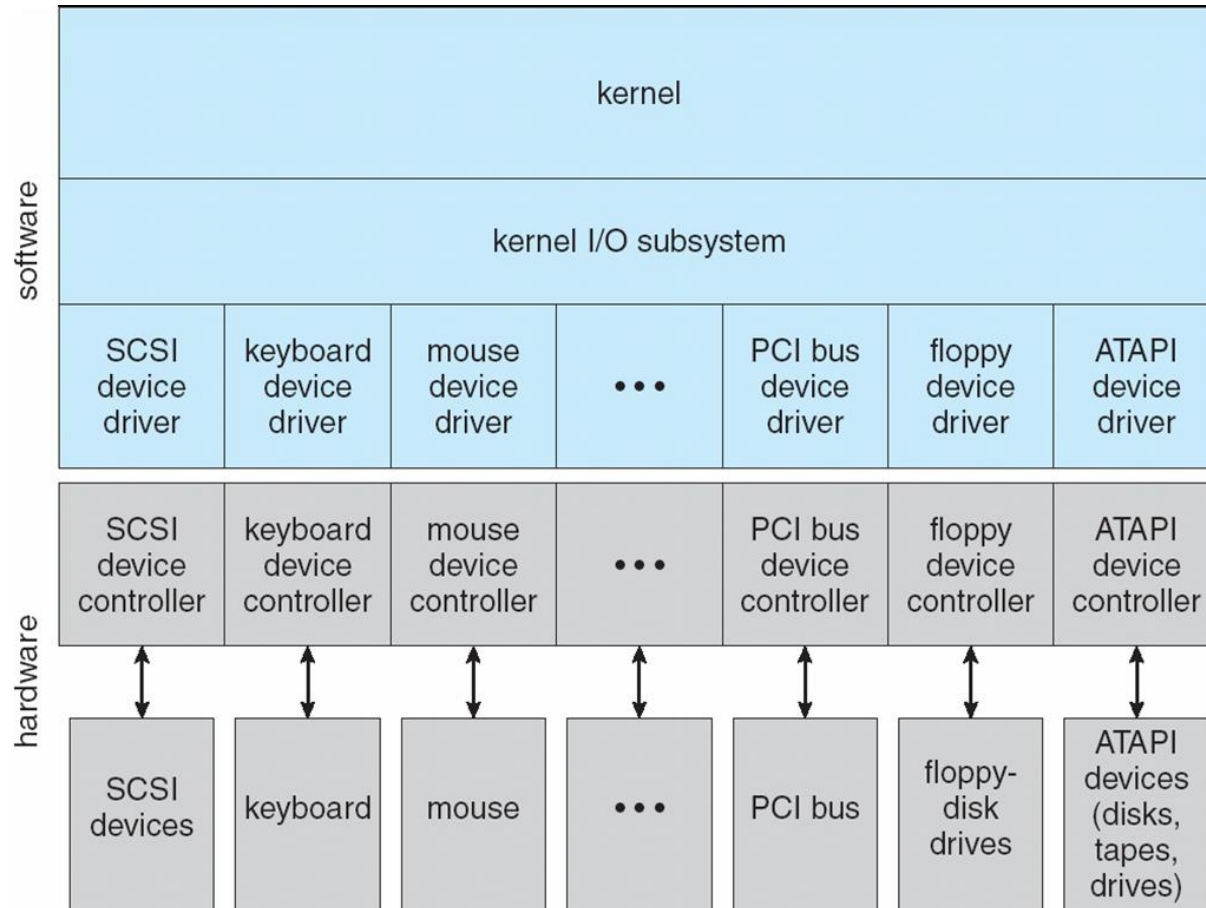




# Application I/O Interface

- I/O system calls encapsulate device behaviors in generic classes
- Device-driver layer hides differences among I/O controllers from kernel
- New devices talking already-implemented protocols need no extra work
- Each OS has its own I/O subsystem structures and device driver frameworks
- Devices vary in many dimensions
  - **Character-stream** or **block**
  - **Sequential** or **random-access**
  - **Synchronous** or **asynchronous** (or both)
  - **Sharable** or **dedicated**
  - **Speed of operation**
  - **read-write**, **read only**, or **write only**

# A Kernel I/O Structure



# Storage

# Storage Structure

Memory  
for short

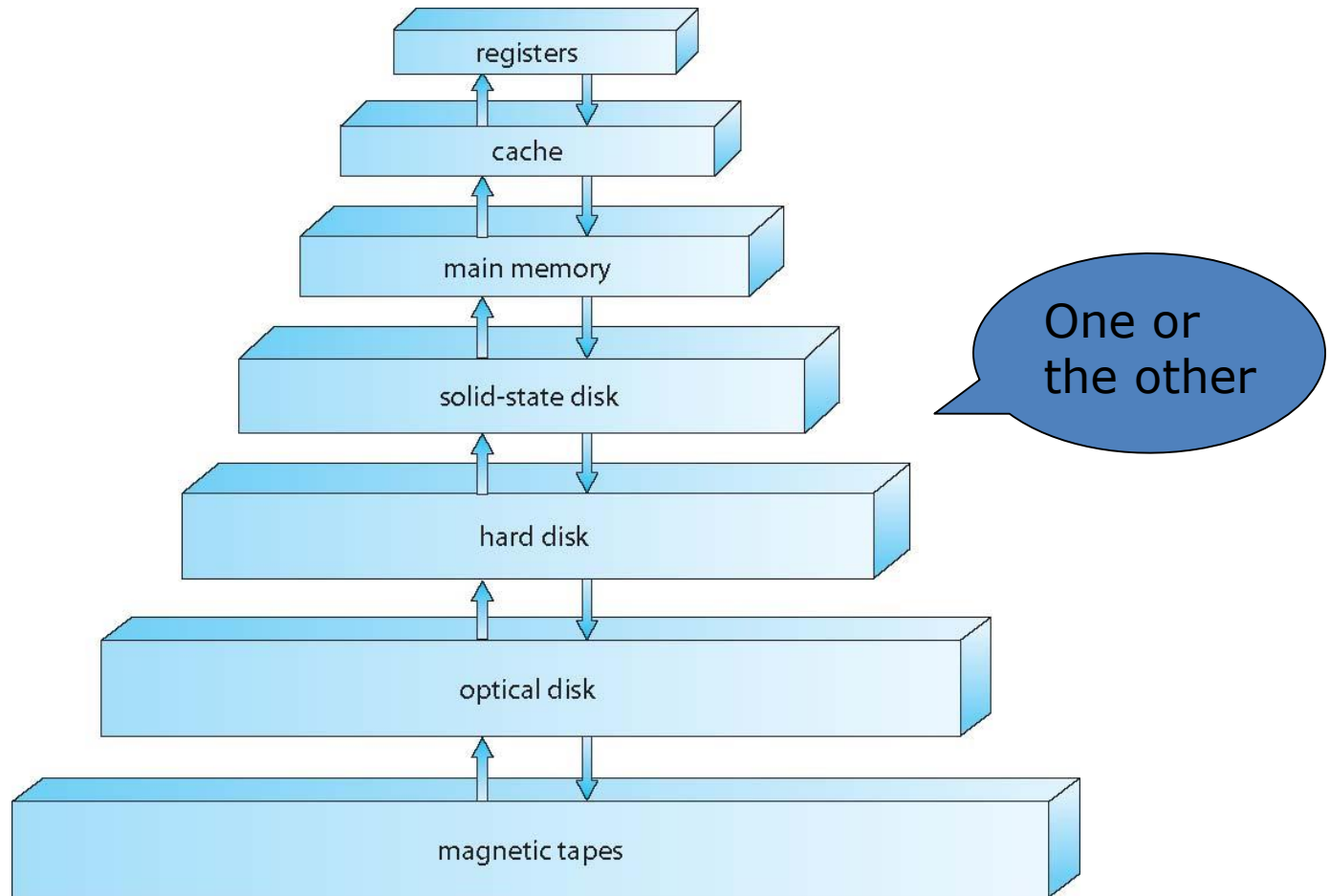
Disk  
for short

- Main memory – only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile (except for ROM)**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
  - Hard disks (HDD) – rigid platters covered with magnetic recording material
    - Disk surface divided into **tracks**, which are subdivided into **sectors**
    - The **disk controller – transfers** between the device and the processor
  - **Solid-state disks (SSD)** – faster than hard disks, lower power consumption
    - More expensive, but becoming more popular
- Tertiary/removable storage
  - External disk, thumb drives, cloud backup etc.

# Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- **Caching** – copying information into faster storage system; main memory can be viewed as a cache for secondary storage
- **Device Driver** for each device controller to manage I/O
  - Provides uniform interface between controller and kernel

# Storage-Device Hierarchy



# Performance of Various Levels 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

Movement between levels of storage hierarchy can be explicit or implicit

- Cache managed by hardware. Makes main memory appear much faster.
- Disks are several orders of magnitude slower.

# Caching

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use 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 smaller than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy



# Direct Memory Access Structure

- high-speed I/O devices
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only one interrupt is generated per block

