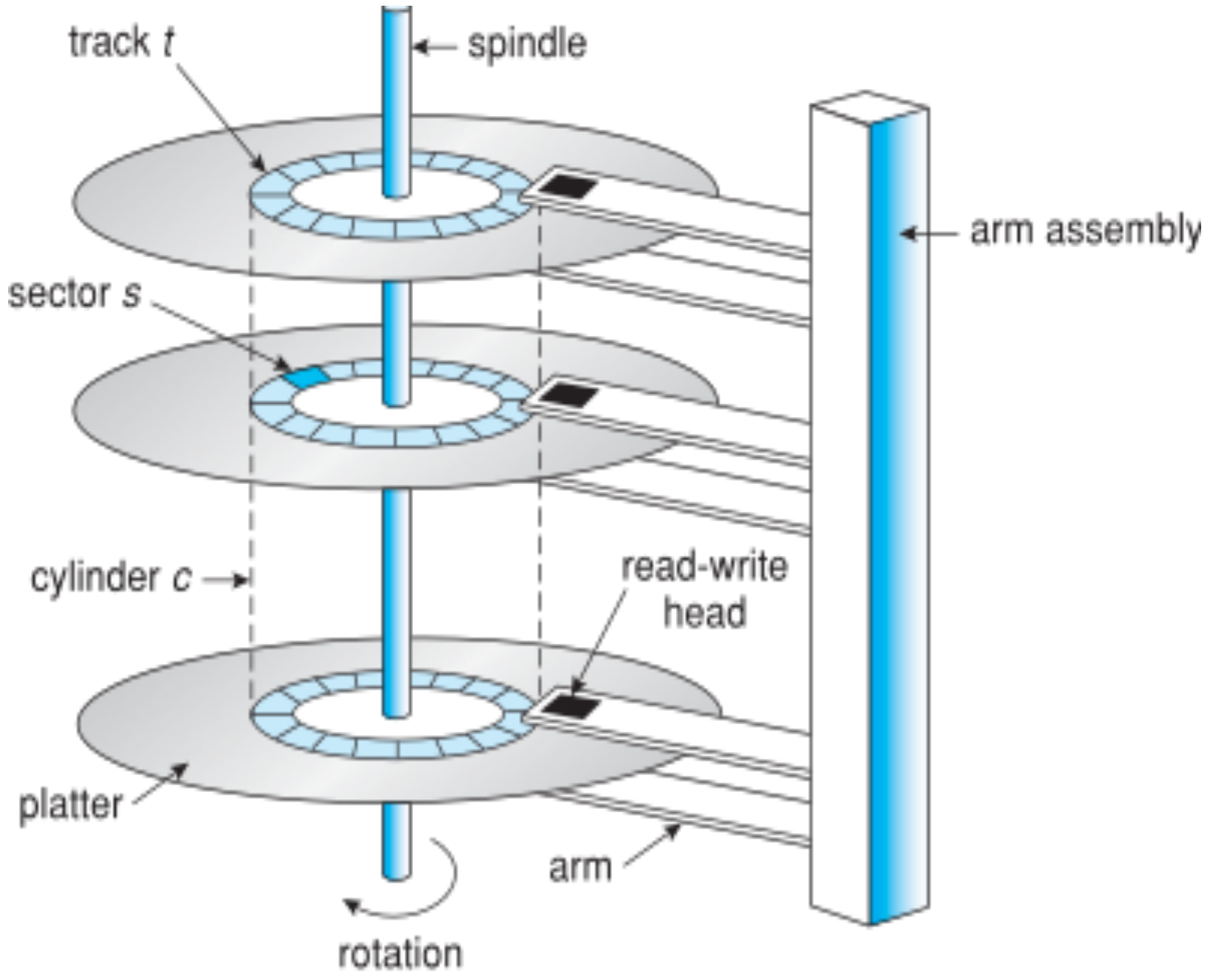
Introduction to I/O

04/13/2021

Professor Amanda Bienz

Mass Storage : Hard Disk Drives

- HDDs spin platters of magnetically-coated material under moving read-write heads
- Rotate at 60-250 times per second
- Transfer rate is rate at which data flow between drive and computer
- Random-access time is time to move disk arm to desired cylinder (seek time) and time for desired sector to rotate under disk head (rotational latency)
- Head crash : disk head makes contact with the disk surface (bad)



Non-Volatile Memory

- Solid-state disks, USB drives, DRAM, main storage in devices like smartphones
- Can be more reliable than HDDs
- More expensive per MB
- May have shorter life span
- Less capacity
- **Much faster**
- No moving parts, so no seek time or rotational latency

Non-Volatile Memory Devices

- Challenges:
  - Read and written in “page” increments, but cannot be overwritten in place
  - Must first be erased, and erases happen in large “block” increments
  - Can only be erased a limited number of times before worn out (can be a very large number)
  - Life space measured in drive writes per day
    - 1TB drive, rating of 5DWPD, can have 5TB per day written within warrantee period without failing

Volatile Memory

- DRAM frequently used as mass-storage device
  - Not technically secondary storage because volatile, but can have file systems and be used like very fast secondary storage
- RAM drives : raw block devices, commonly file system formatted
  - Linux /dev/ram, Mac diskutil : used to create them
- Used as high speed temporary storage
  - Programs can share bulk data quickly by reading/writing to RAM drive

Disk Scheduling

- Operating system is responsible for fast access time and disk bandwidth
- **Disk bandwidth :** the total number of bytes transferred divided by the total time between the first request for service and completion of last transfer
- Disk I/O request sources : OS, system processes, user processes
- OS maintains queue of requests, per disk or device
- Idle disk can immediately work on I/O request, but disk means work must queue
- In past, operating system responsible for queue management, but now built into storage devices and controllers

I/O Management

- Major component of operating system design and operation
  - Important aspect of computer operation
  - I/O devices vary greatly
  - Various methods to control them
  - Performance Mangement
- Device drivers encapsulate device details
  - Present uniform device-access interface to I/O subsystem

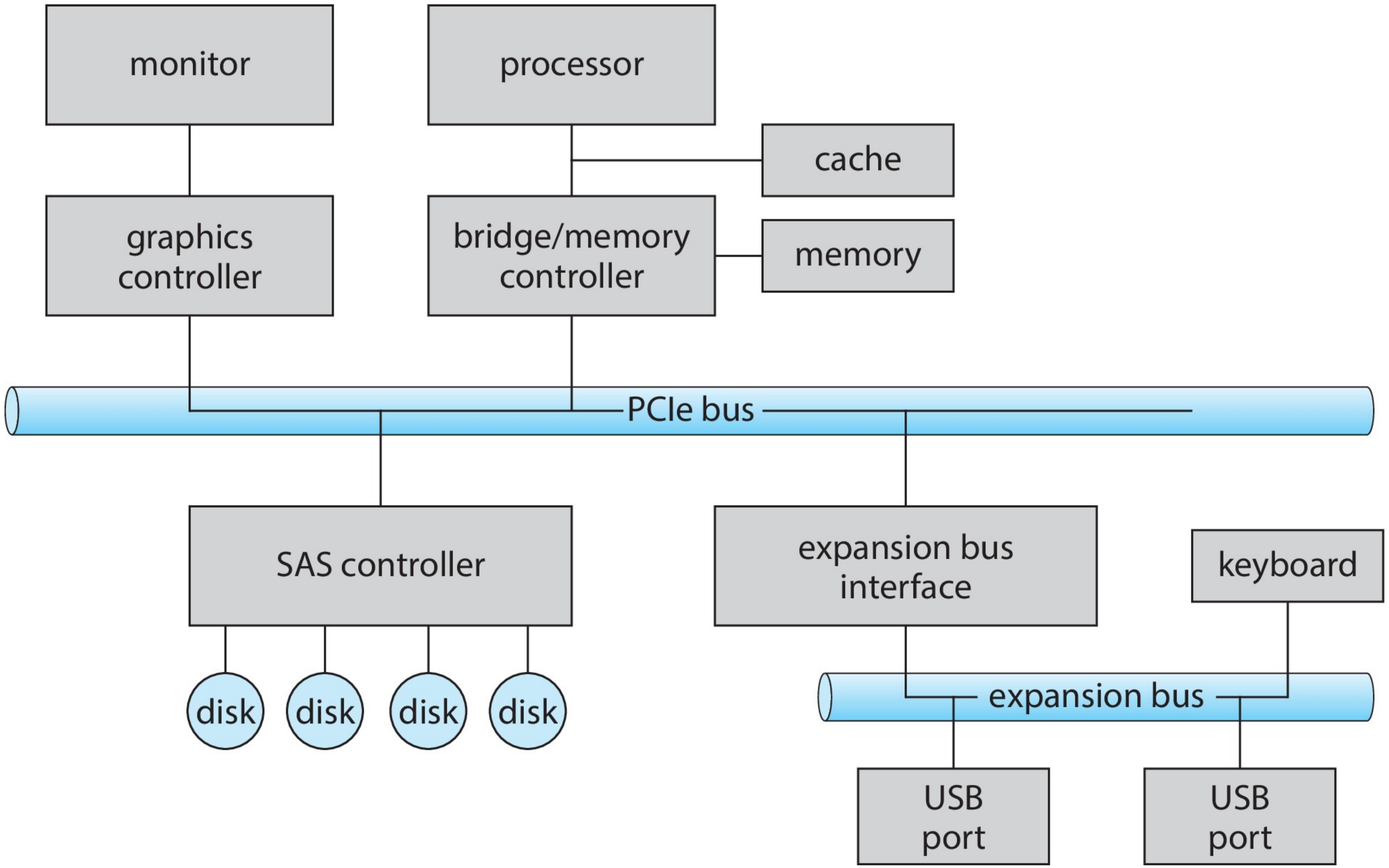
I/O Hardware

- Incredible variety of I/O devices
  - Storage
  - Transmission
  - Human-interface
- Common concepts : signals for I/O devices interface with computer
  - **Port** : connection point for device
  - **Bus :** shared direct access
    - **PCI bus :** common in PCs and servers (PCI Express)
    - **Expansion bus**: connects relatively slow devices

I/O Hardware (Cont.)

- **Controller :** (host adapter) electronics that operate port, bus, device
  - Sometimes integrated
  - Sometimes separate circuit board
  - Contains processor, private memory, bus controller, etc.

A Typical PC Bus Structure



I/O Hardware (Cont.)

- **Fibre channel (FC)** : complex controller, usually separate circuit board plugging into bus
- I/O instructions control devices
- Devices usually have registers where device driver places commands, addresses, and data to write, or read data from registers after command execution
  - Data-in register, data-out register, status register, control register

I/O Hardware (Cont.)

- Devices have addresses, used by
  - Direct I/O instructions
  - Memory-mapped I/O
    - Device data and command registers mapped to processor address space
    - Especially for large address spaces (graphics)

Device I/O Port Locations on PCs (partial)

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

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 is done
- Step 1 : busy-wait (wait for I/O from device). Reasonable if device is fast, but not if device is slow
    - CPU switches to other tasks? But cycle data could be overwritten/lost

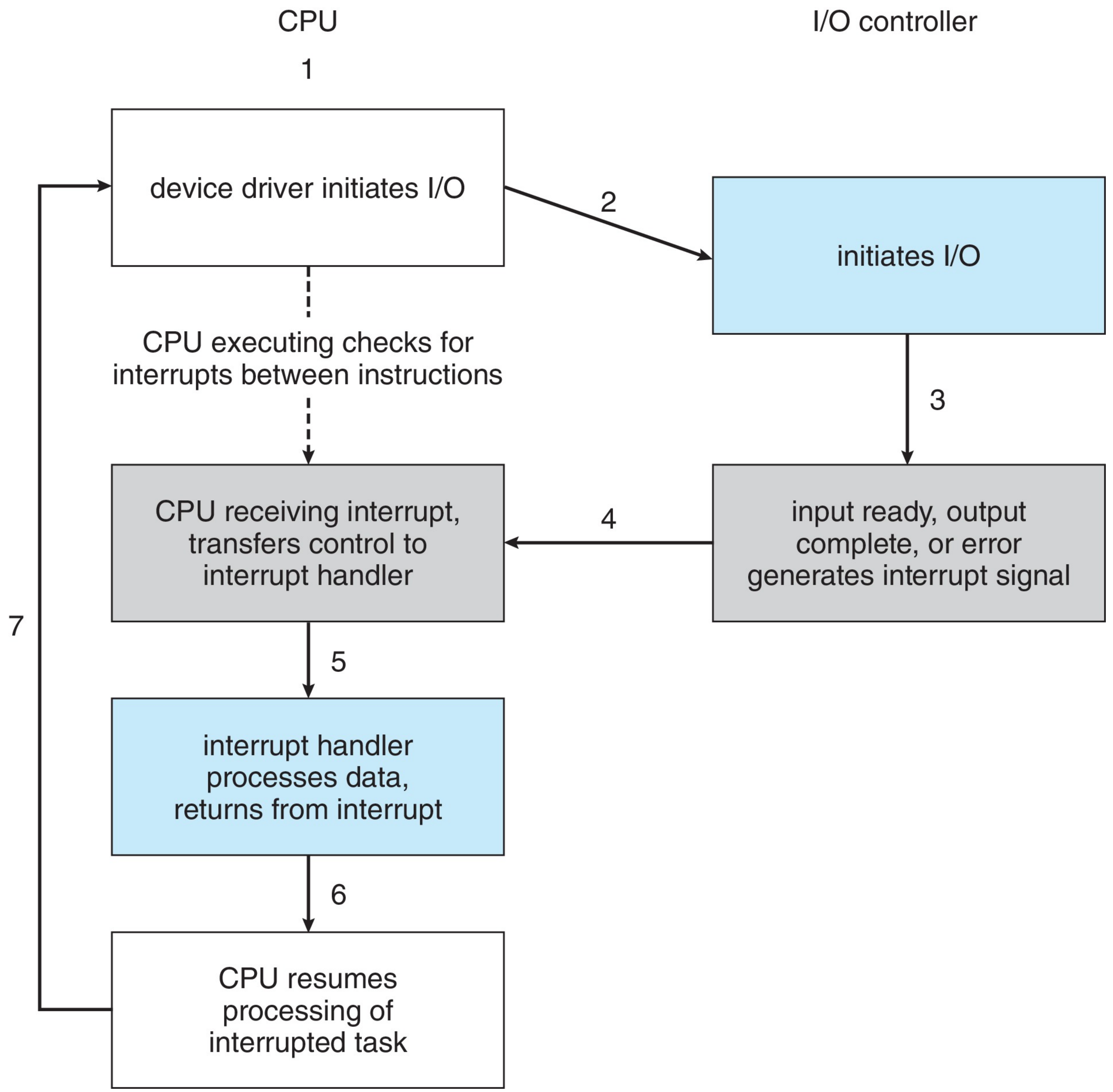
Interrupts

- Polling can happen in 3 instruction cycles
  - Read status, logical-and to extract status bit, branch if not zero
  - Host to be more efficient if non-zero infrequently?
- **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

Interrupts (Cont.)

- Interrupt vector to dispatch interrupt to correct handler
  - Context switch at start and end
  - Based on priority
  - Some non-maskable
  - 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
- System call executes via **trap** to trigger kernel to execute request
- Multi-CPU systems can process interrupts concurrently
  - If operating system designed to handle it
- Used for time-sensitive processing, frequent, must be fast

Latency

- Stressing interrupt management because even single-user systems manage hundreds of interrupts per second and servers hundreds of thousands

- For example, a quite macOS desktop generated 23,000 interrupts over 10 seconds

```
Fri Nov 25 13:55:59                                 0:00:10
                         SCHEDULER      INTERRUPTS
---------------------------------------------
total_samples                   13           22998

delays <  10 usecs              12           16243
delays <  20 usecs               1            5312
delays <  30 usecs               0             473
delays <  40 usecs               0             590
delays <  50 usecs               0              61
delays <  60 usecs               0             317
delays <  70 usecs               0               2
delays <  80 usecs               0               0
delays <  90 usecs               0               0
delays < 100 usecs               0               0
total  < 100 usecs              13           22998
```

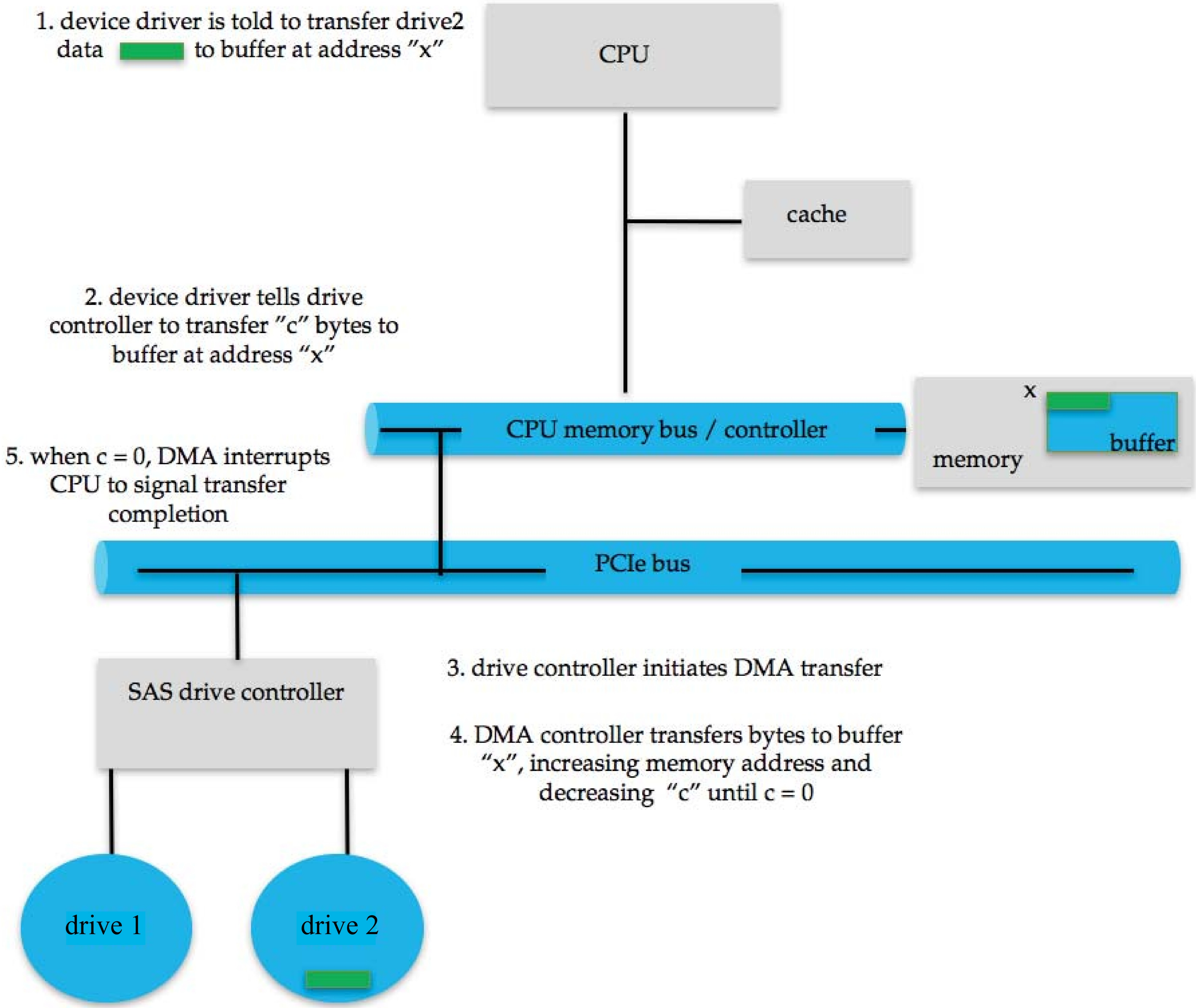
Intel Pentium Processor Event-Vector Table

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

Direct Memory Access

- Used to avoid programmed I/O (one byte at a time) for large data movement
- 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)
  - When done, interrupt to signal completion

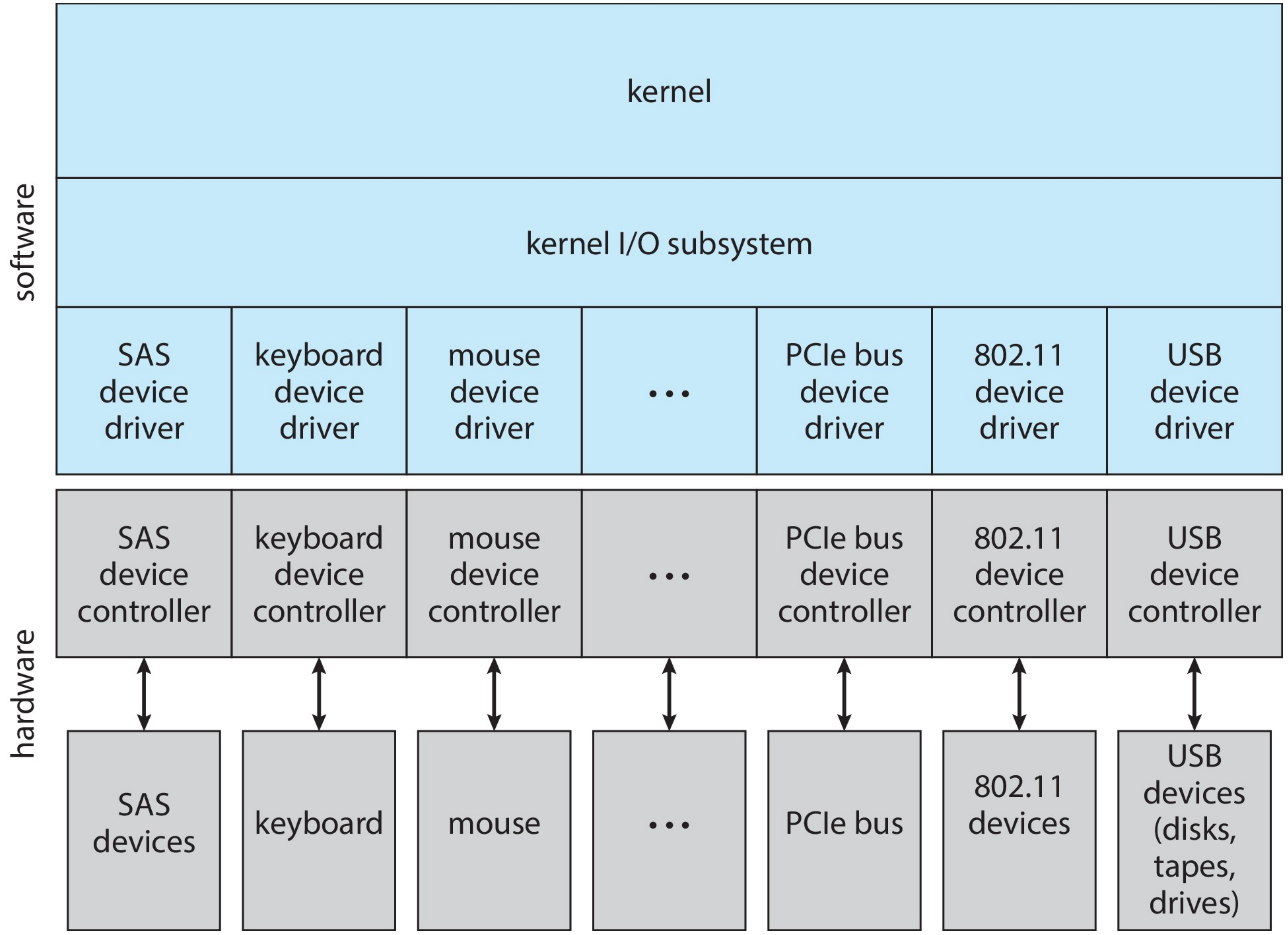
Six Step Process : DMA Transfer



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 taking 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 vs block, sequential vs random-access, <a>synchronous, sharable or dedicated, speed of operations, read-write vs read only vs write only

A Kernel I/O Structure



Characteristics of I/O Devices

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only read–write	CD-ROM graphics controller disk

Characteristics of I/O Devices (Cont.)

- Subtleties of devices handled by device drivers
- Broadly I/O devices can be grouped by the OS into:
  - Block I/O
  - Character I/O (stream)
  - Memory-mapped file access
  - Network sockets
- For direct manipulation of I/O device specific characteristics, usually an escape / back door
  - Unix ioctl() call to send arbitrary bits to a device control register and data to device data register

Block and Character Devices

- Block devices include disk drives
  - Commands include read, write, seek
  - Raw I/O, direct I/O, or file-system access
  - Memory-mapped file access possible (mapped to virtual memory and cluster brought via demand paging)
  - DMA
- Character devices include keyboards, mice, serial ports
  - Commands include get(), put()

Network Devices

- Varying enough from block and character to have own interface
- Linux, Unix, Windows, and many others include socket interface
- Approaches vary widely (pipes, FIFOs, streams, queues, mailboxes)

Nonblocking and Asynchronous I/O

- **Blocking** - process suspended until I/O completed
  - Easy to use and understand
  - Insufficient for some needs
- **Nonblocking** - I/O call returns as much as available
  - User interface, data copy (buffered I/O)
  - Implemented via multi-threading
  - Returns quickly with count of bytes read or written
  - `select()` to find if data ready then `read()` or `write()` to transfer
- **Asynchronous** - process runs while I/O executes
  - Difficult to use
  - I/O subsystem signals process when I/O completed