Operating Systems (INFR09047)

2019/2020 Semester 2

Secondary-storage and IO Subsystems

abarbala@inf.ed.ac.uk

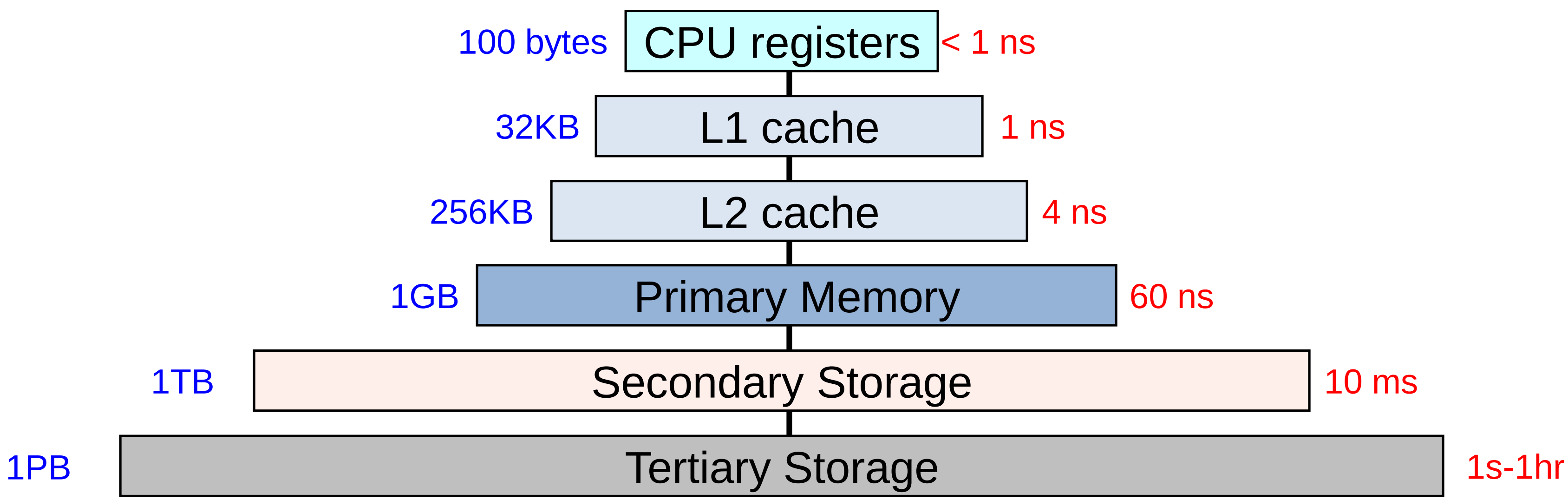
Chapter 11, Chapter 12

Secondary-storage: Overview

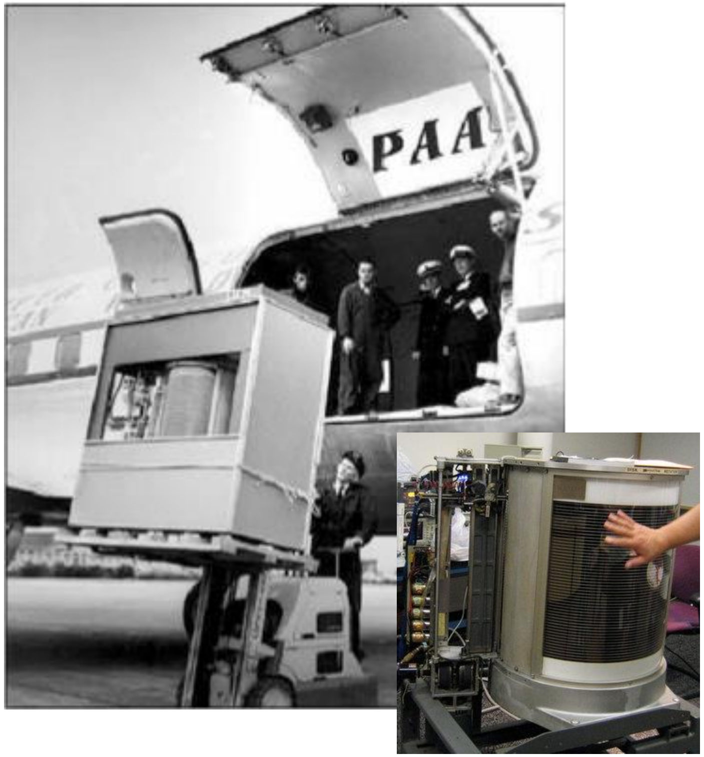
- The Memory Hierarchy
- Magnetic Disks
  - Technology
  - Performance
  - Scheduling
  - Scheduling Algorithms
- Solid-state Drives
  - Read/write
  - SSD vs HDD
- Device Management

Traditional Secondary Storage

- **Block access (vs byte access)**
  - CPU cannot access secondary storage directly
  - CPU accesses primary storage directly (e.g., move instruction)
- Characteristics
  - **Large**: 500 - 4000GB and more
  - **Cheap**: 0.035gbp/GB for hard disk drives
  - **Slow**: millisecond
  - **Persistent**: data survives power loss
  - **Fail rarely**
    - Drive dies; Mean Time Between Failure (MTBF) ~3 years
    - 100,000 drives and MTBF is 3 years, 1 “big failure” every 15 minutes!



Early Magnetic Disk Storage Systems



1956
IBM Model 350 disk storage system

5M 6-bit characters (3.75MB)
50 x 24” platters
8,800 character/sec
(part of IBM RAMAC computer)

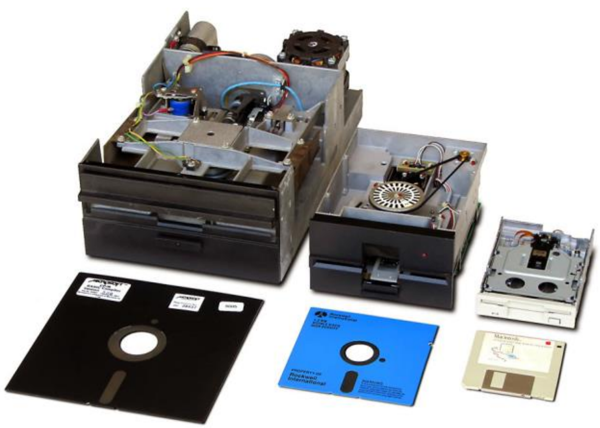
1965
IBM 2314 storage system

29.2M bytes (29.2MB)
8 x 11 platters
310,000 byte/sec

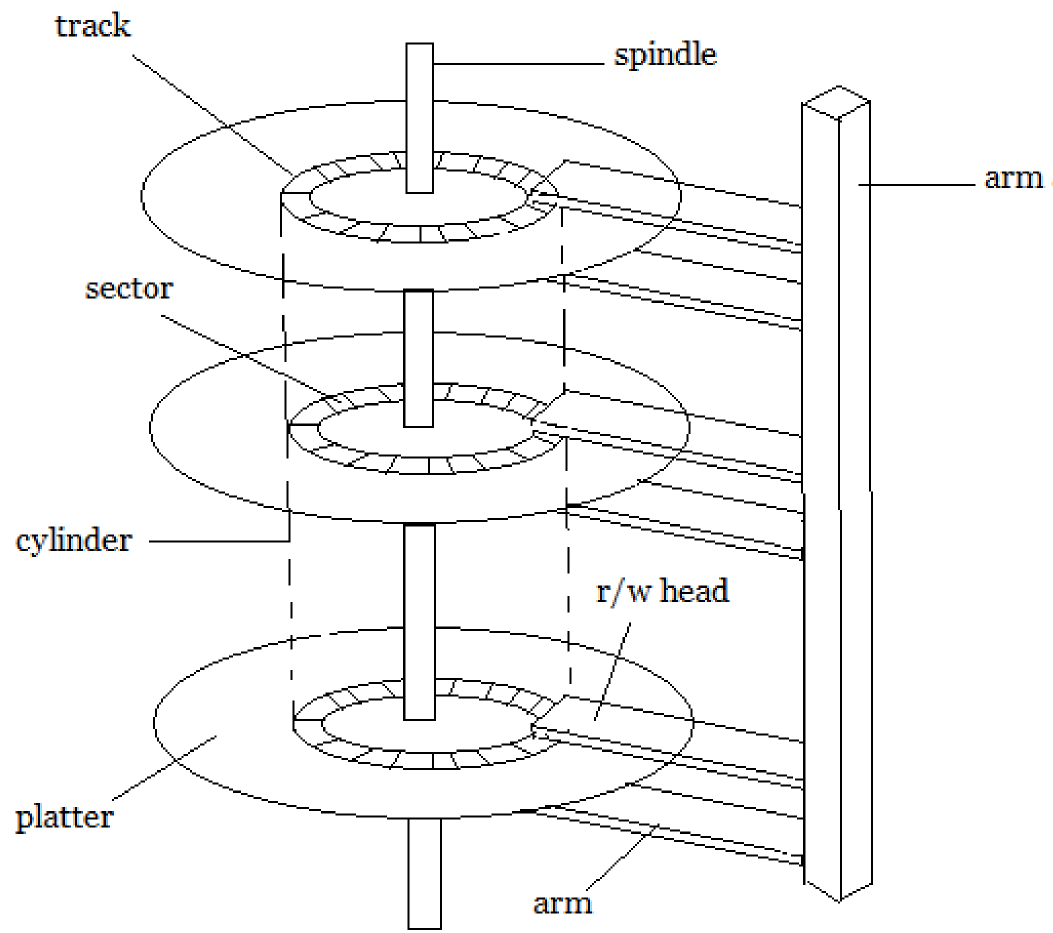
Magnetic Disks #1

Hard Disk Drive (HDD)

Floppy Disk Drive (FDD)



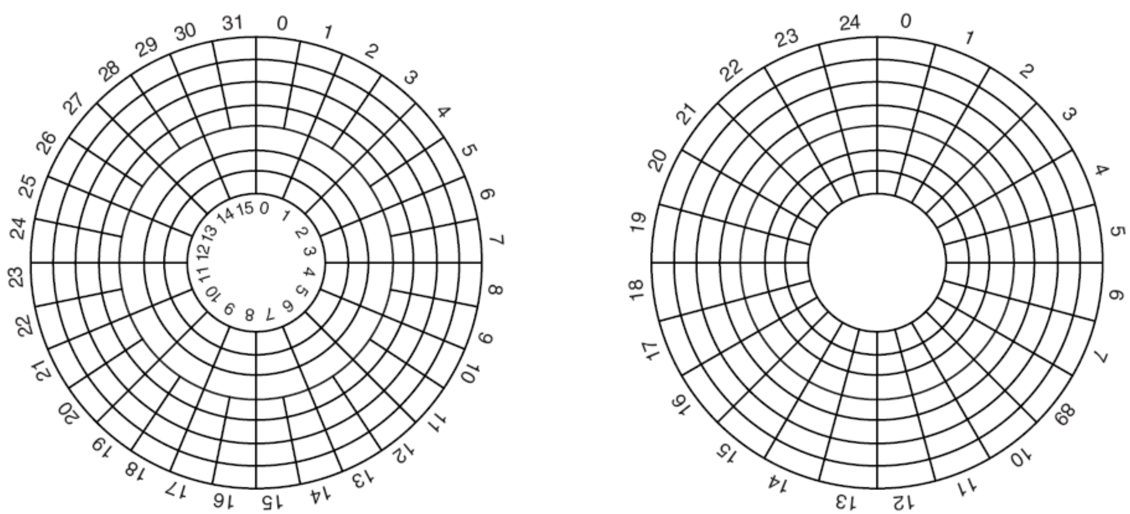
(single arm/head)



https://www.studytonight.com/operating-system/images/secondary-storage-1.png

Magnetic Disk #2

- Read/write errors, bad blocks, missed seeks, etc.
- Physical Geometry vs Addressing
  - **Previously** geometry used for addressing: **Cylinder, head, sector**
  - **Now** independent: **Logical Block Address (LBA)**
    - Mapped onto the sectors of the disk sequentially



(**left**) Physical geometry of a disk with two zones. (**right**) A possible virtual geometry (addressing) for this disk

Example: Seagate Barracuda 3.5” Disk Drive

- **35gbp** cost (March 2020)
- **1Terabyte** of storage (1000 GB)
- 4 platters, 8 disk heads
- 63 sectors (512 bytes) per track
- 16,383 cylinders (tracks)
- 7200 rpm
- up to 300 MB/second transfer (SATA)
- 9 ms avg. seek, 4.5 ms avg. rotational latency
- 1 ms track-to-track seek
- 64 MB cache

Disk Performance

- Depends on …
- **Seek time:** moving the disk arm to the correct cylinder
  - Depends on how fast disk arm can move
    - Not diminishing quickly due to physics
- **Rotation (latency):** waiting for the sector to rotate under head
  - Depends on rotation rate of disk
    - Rates are slowly increasing
- **Transfer time:** transferring data from surface to disk controller
  - Depends on density of bytes on disk
    - Increasing, relatively quickly

- When the OS uses the disk, **tries to minimize** all such costs
  - Specifically, seeks and rotation

Performance

- OS may increase file block size
  - **Reduce seeking**
- OS may seek to co-locate “related” items
  - **Reduce seeking**
    - Blocks of the same file
    - Data and metadata for a file
- OS may keep data or metadata in memory to reduce physical disk access
  - **Avoid** slow disk **accesses**
    - But wasting valuable physical memory
- OS may fetch blocks into memory before requested
  - **Hide** slow disk **accesses**

Performance via Disk Scheduling

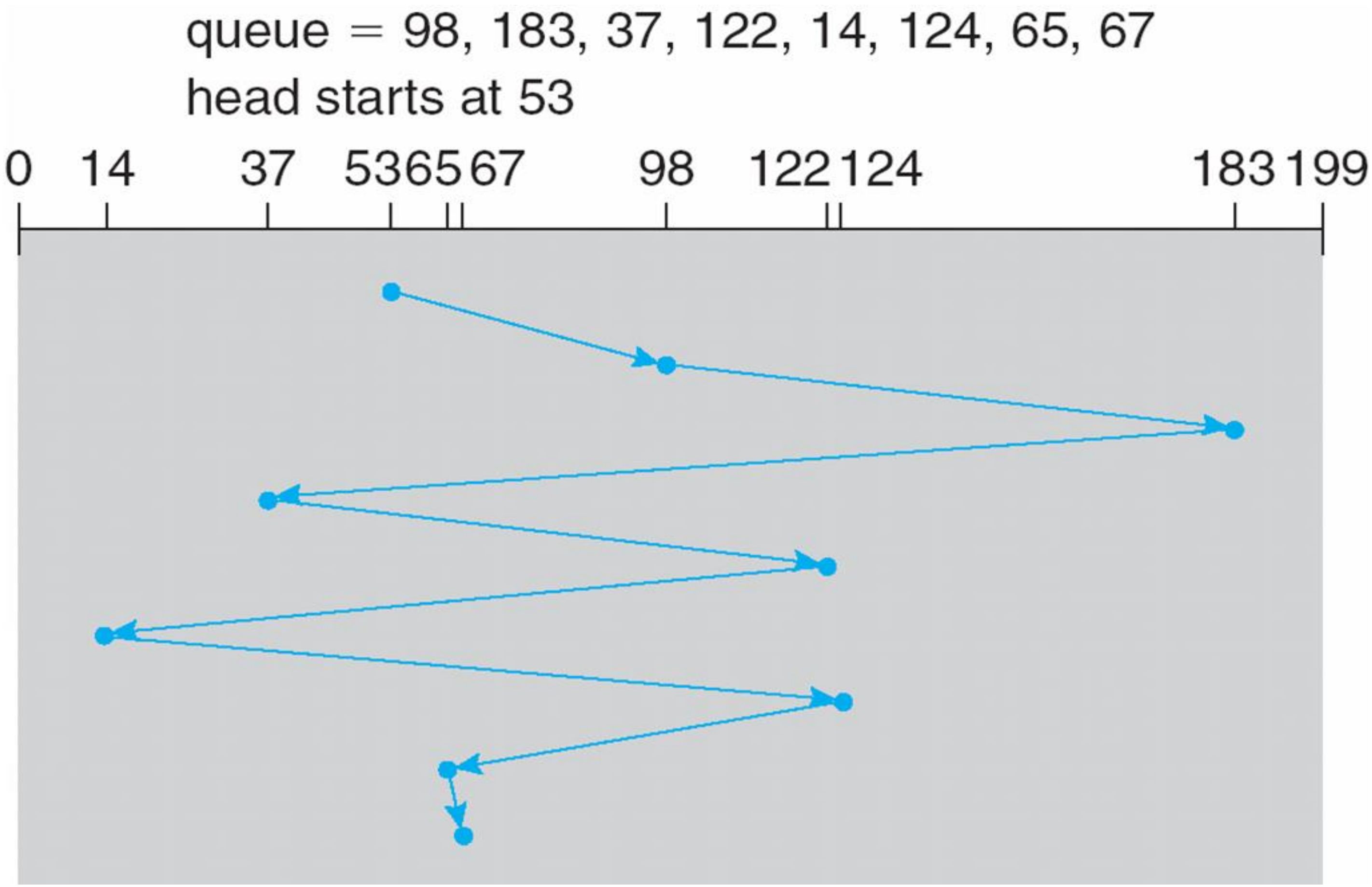
- Applications request data accesses to the OS
  - OS maintains **request queues**
  - OS generates **transfer commands** to/from the disk(s)
    - Imply seeks, waits for rotations, data transfers

- How to **reduce** applications' **waiting time**?
  - OS modifies **order of disk requests queued waiting** for the disk
    - Based on cylinder #
  - Fairness, timeliness, etc.

- Multiple disk scheduling algorithms
  - FCFS (first come first served, no scheduling)
  - SSTF (shortest seek time first)
  - SCAN (elevator algorithm)
  - C-SCAN (typewriter)

FCFS

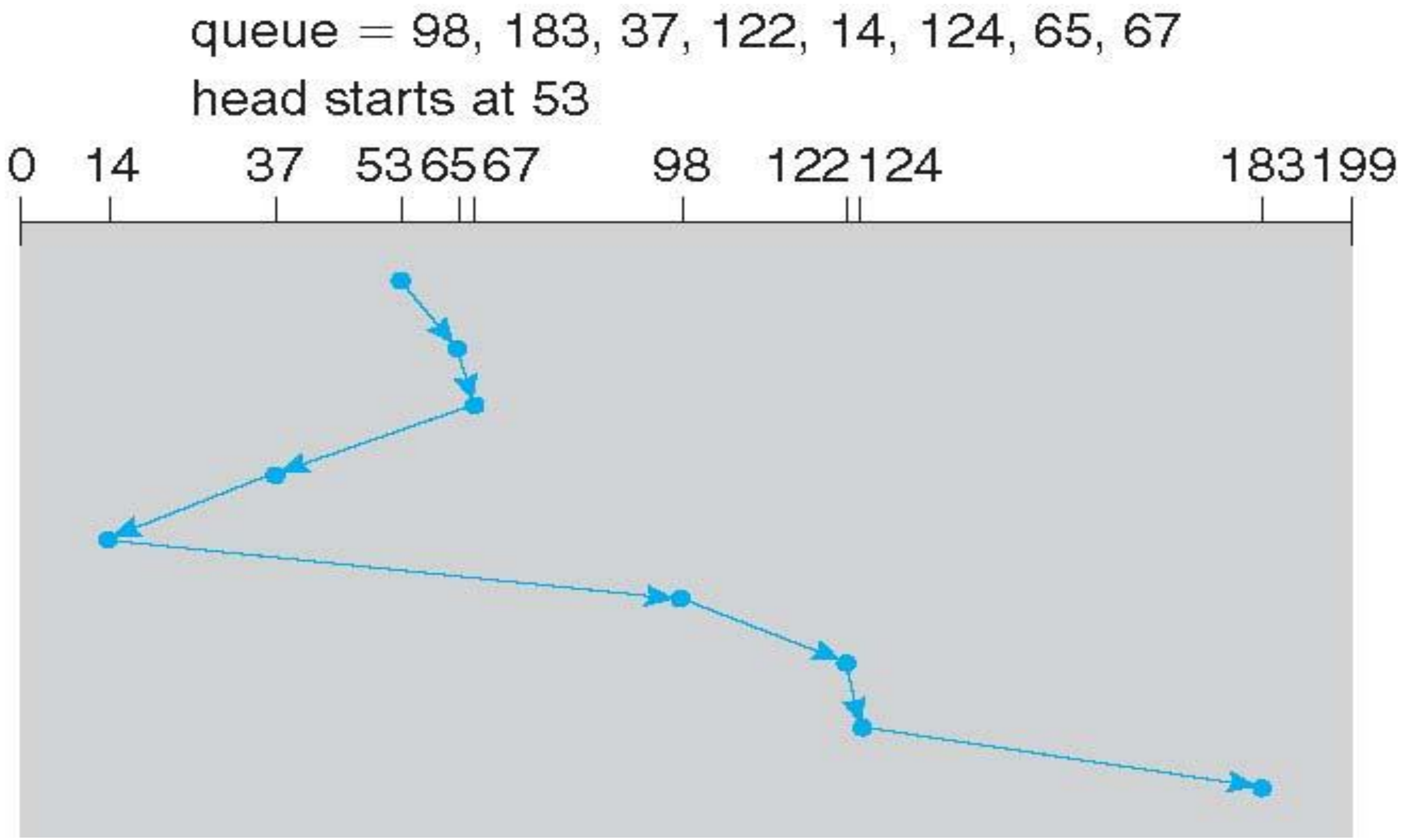
- First come first served



- Reasonable when load is low
- Long waiting time for long request queues

SSTF

- Shortest seek time first



- Minimize arm movement (seek time), maximize request rate
- Unfairly favors middle (clustered) blocks

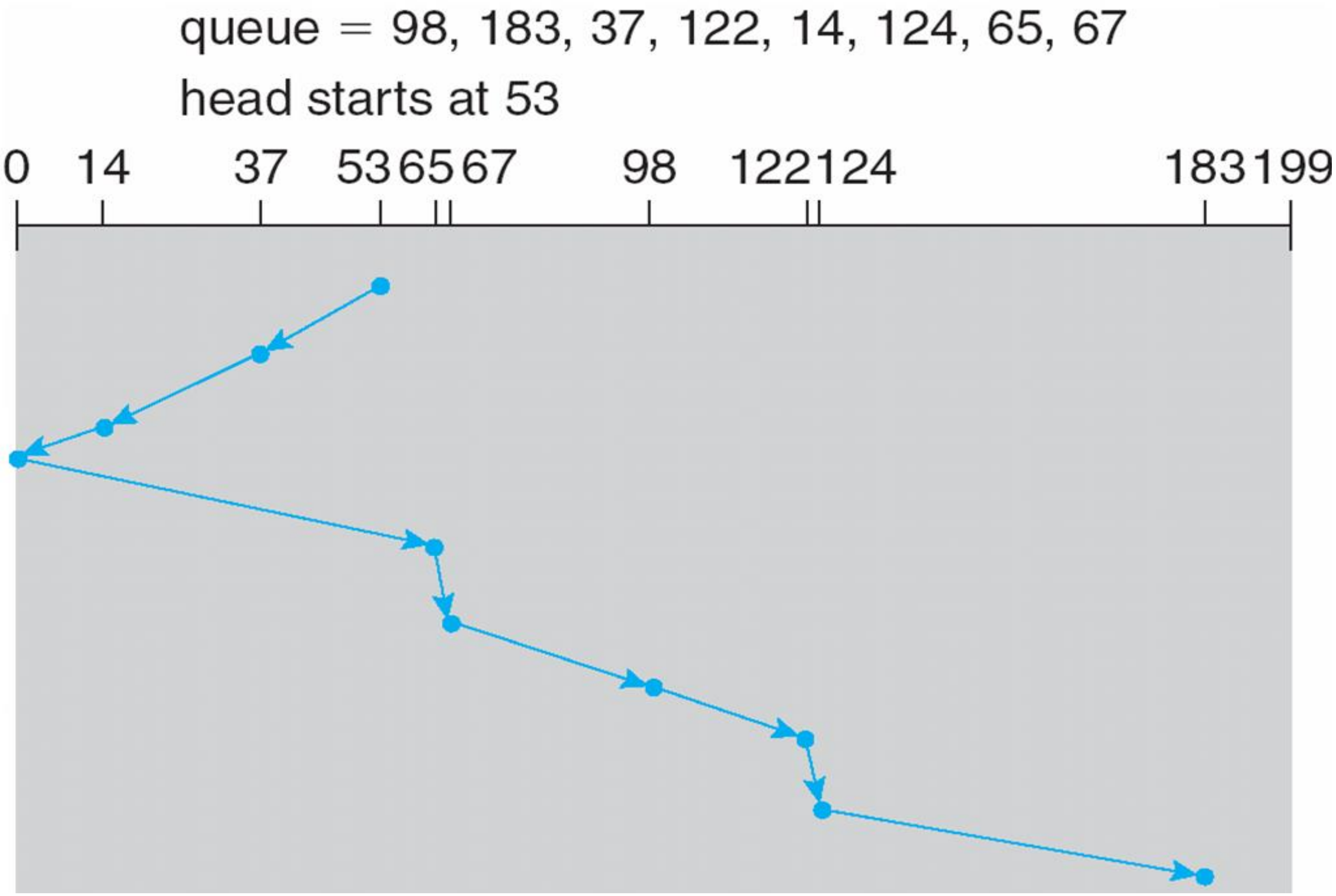
SCAN #1

- Disk arm **starts at one end** of the disk
  - Moves **toward the other end**
- Servicing requests until it gets to the other end of the disk
  - Where the head movement is reversed, and **servicing continues**

- **SCAN algorithm** called the **elevator algorithm**
  - https://www.popularmechanics.com/technology/infrastructure/a20986/the-hidden-science-of-elevators/

- Note
  - If requests are uniformly dense
  - largest density at other end of disk
  - and those wait the longest

SCAN #2



- Skews wait times non-uniformly

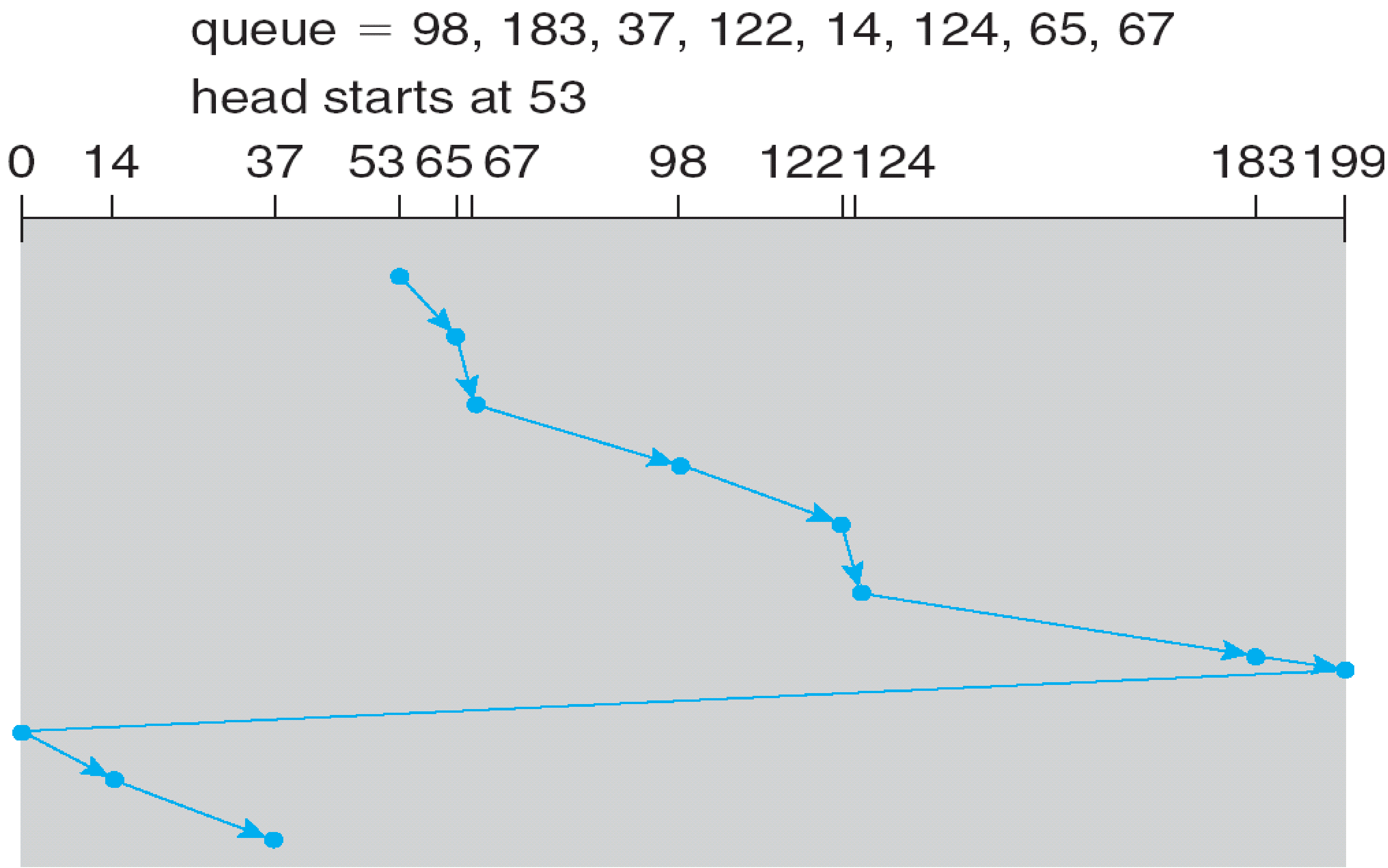
C-SCAN #1

- Provides a **more uniform wait time** than SCAN

- Head moves from **one end** of the disk **to the other**
  - Servicing requests as it goes
- When it reaches the **other end**
  - Immediately returns to the **beginning of the disk**
  - **Without servicing** any requests on the return trip

- Also known as **typewriter** algorithm

C-SCAN #2



- Uniform wait times

Selecting a Disk-Scheduling Algorithm

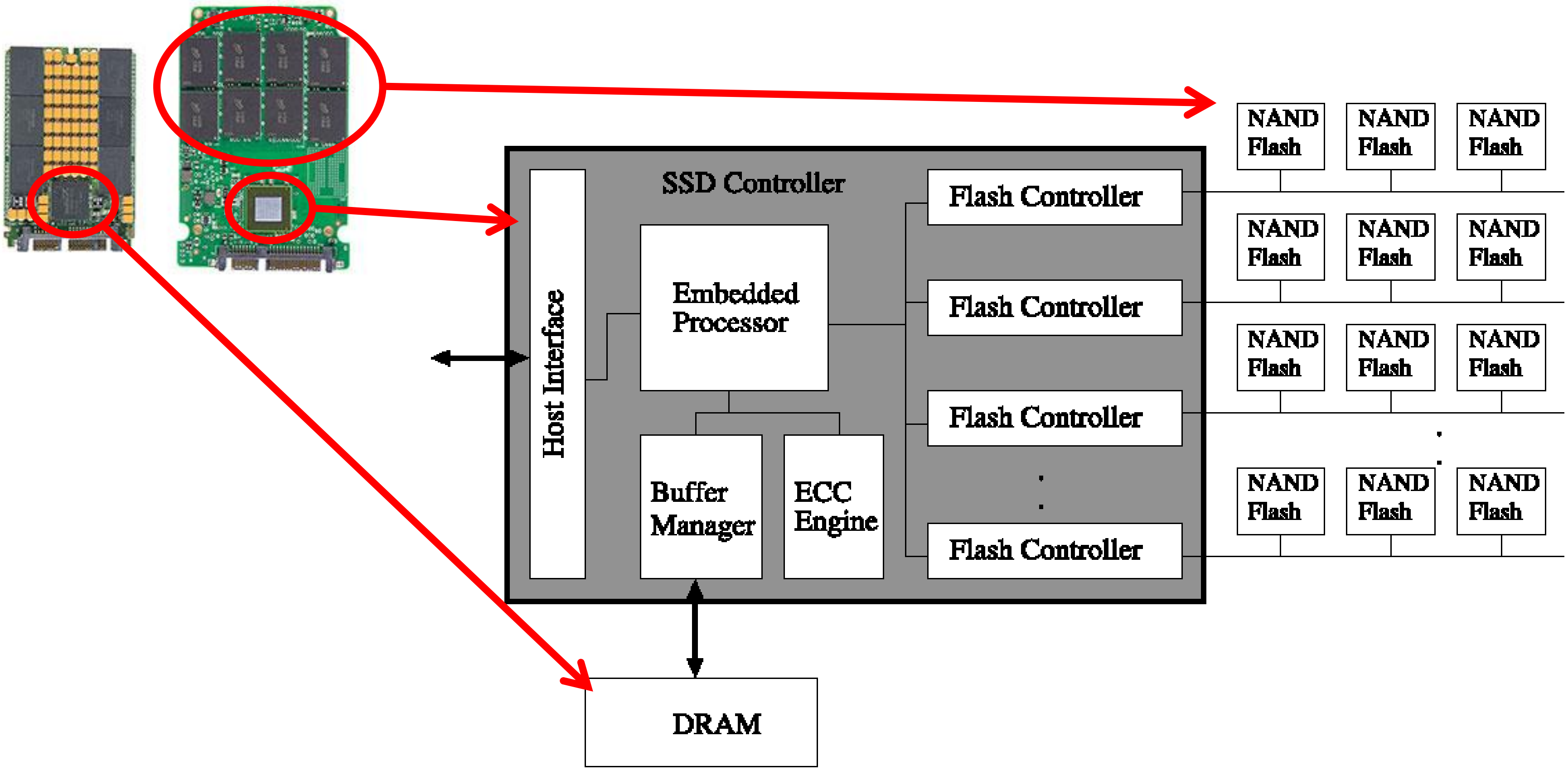
- When there is **one request** all algorithms behave like FCFS
- SCAN and C-SCAN perform better for systems with **heavy load** on the disk (less starvation)

- Performance depends on the **number and types of requests**
- Requests for disk service can be **influenced by**
  - File-allocation method
  - Metadata layout

- OS disk-scheduling algorithm
  - **Module** of the OS, ease replacement
- **Linux**
  - **Deadline:** variation of C-SCAN with two queues
  - **NOOP:** variation of FCFS
  - **CFQ:** uses the concept of timeslices

Solid-State Drives #1

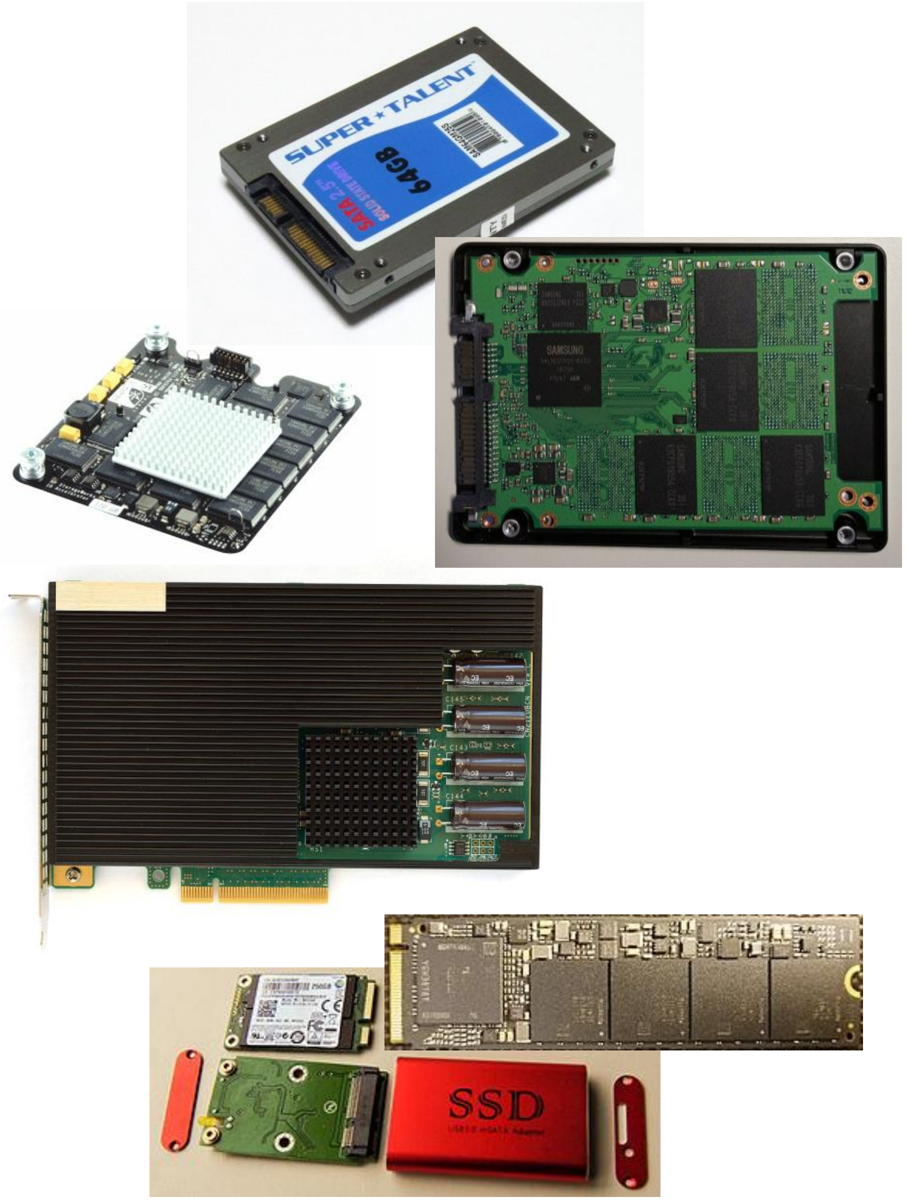
Flash Disks



Solid-State Drives #2

- Different technologies
  - NOR
  - **NAND**
  - **3D XPoint**
  - Memristor
  - …

- Multiple interfaces
  - USB
  - SATA, mSATA
  - NVMe (M.2, PCIe)
  - …



SSD Performance: Reads

- Reads
  - Unit **of read is a *page***, typically 4kB
- COTS SSD handles
  - **~100k reads/s**
- **10-100us** latency
  - 50-1000x better than magnetic disks
- **60-600 MB/s** read throughput
  - 1-10x better than magnetic disks

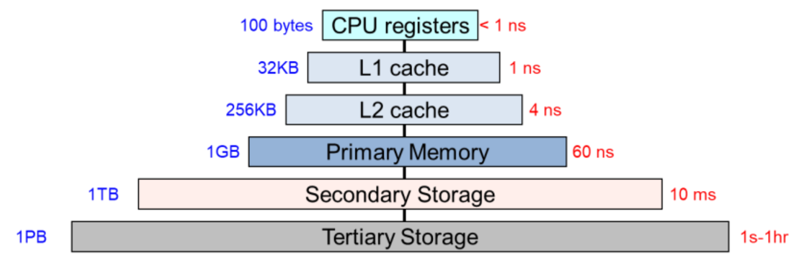
SSD Performance: Writes

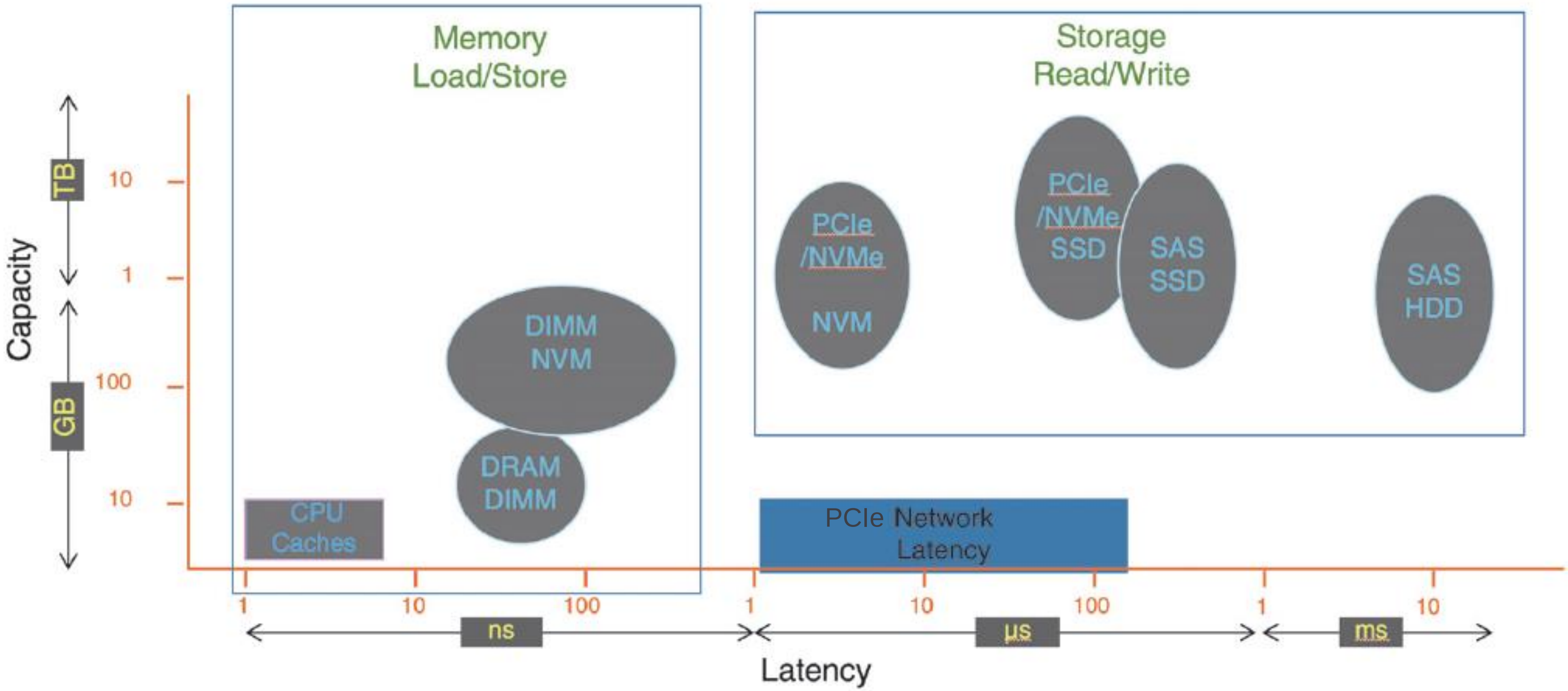
- Writes
  - Unit of **write is a page**
    - Lower writes/s than reads/s
    - Higher write latency than read latency
    - Lower throughput than read
- Flash media must be ***erased* before it can be written**
  - **Unit of erase is a block**, typically 64-256 pages
    - Takes ~1ms to erase a block
    - Can only be erased a certain number of times before unusable
      - Typically 10,000 – 1,000,000 times
- To extend lifetime require **Flash Translation Layer** (FTL)
  - Implemented in firmware
  - Wear leveling

SSD vs HDD

- **Capacity** (March 2020)
  - Flash SSD costs at min 1gbp/GB
    - 1TB drive costs around 100gbp (cheap models)
  - 1TB hard drive costs around 35gbp
- **Energy**
  - SSD is typically more energy efficient than a hard drive
    - 1-2 watts to power an SSD
    - ~10 watts to power a hard disk drive
- **Physical resistance**
  - SSD has no moving parts
  - Hard disk drive cannot work correctly if subject to physical acceleration

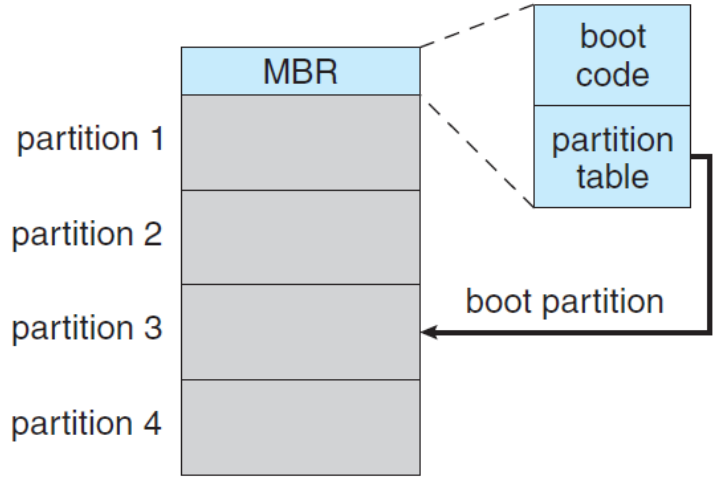
New and Old **Secondary Storage** vs **Primary Storage**





Storage Device Management

- **Storing the data** on the device is not enough
  - Need metadata
- Before storing the data, device **needs to be initialized**
  - Low-level formatting
  - Volume creation (lvm2)
  - Logical formatting (file system)



- **Booting**
  1. Firmware, or BIOS
  2. Reads code in MBR
  3. MBR also contain partition table
  4. Code in MBR reads boot sector of the selected partition
  5. Pass control to code in selected partition

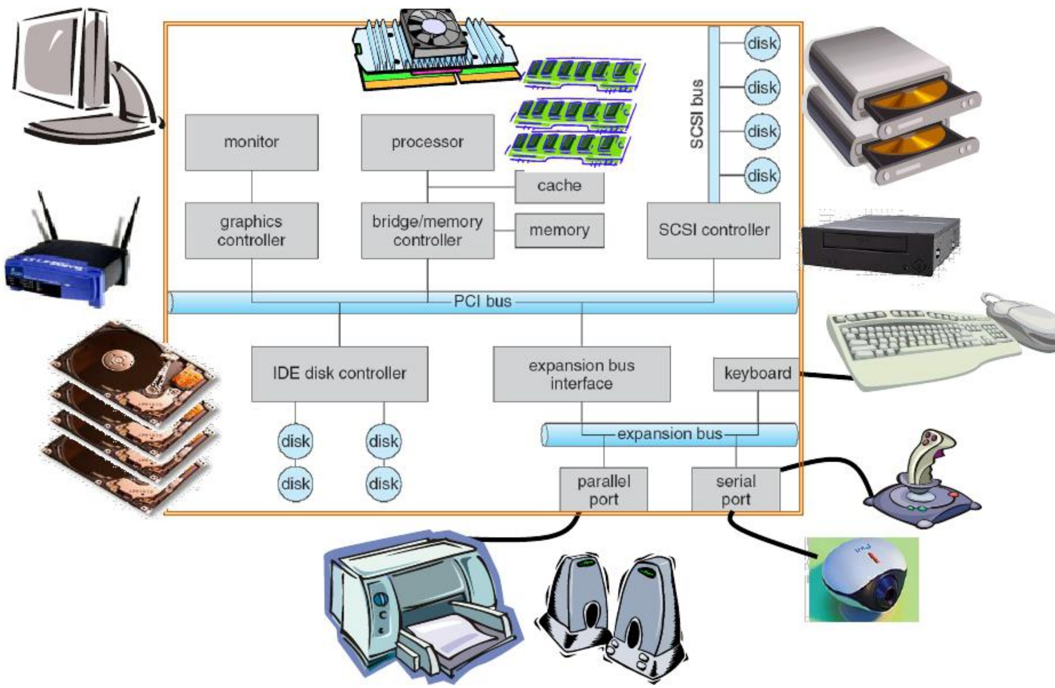
IO Subsystems: Overview

- Computers do **IO and compute**
- OS manages and controls IO for applications
  - Common interfaces to IO devices
  - IO Services

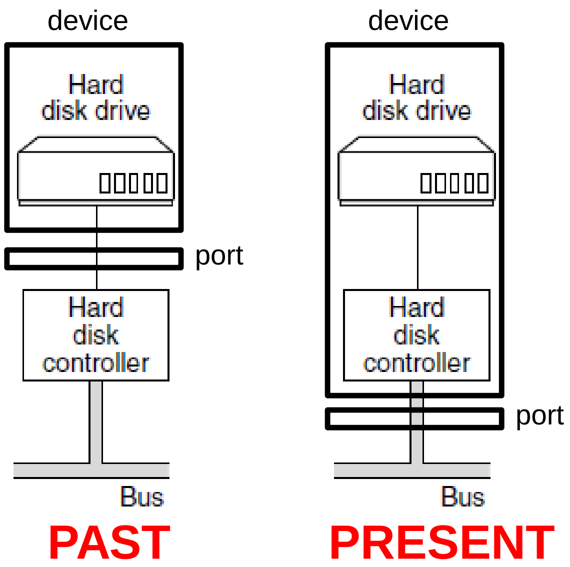
- IO Hardware
- CPU to device communication
  - PIO
  - DMA
- Device Drivers
- IO Subsystem
- An IO Syscall Example

Devices

- Storage devices
  - Disk
  - Tape
- Transmission devices
  - Network connections
  - Bluetooth
- Human-interface devices
  - Screen
  - Keyboard
  - Mouse
  - Audio in
  - Audio out
- Specialized devices
  - E.g., to control a machine/equipment (aircraft)



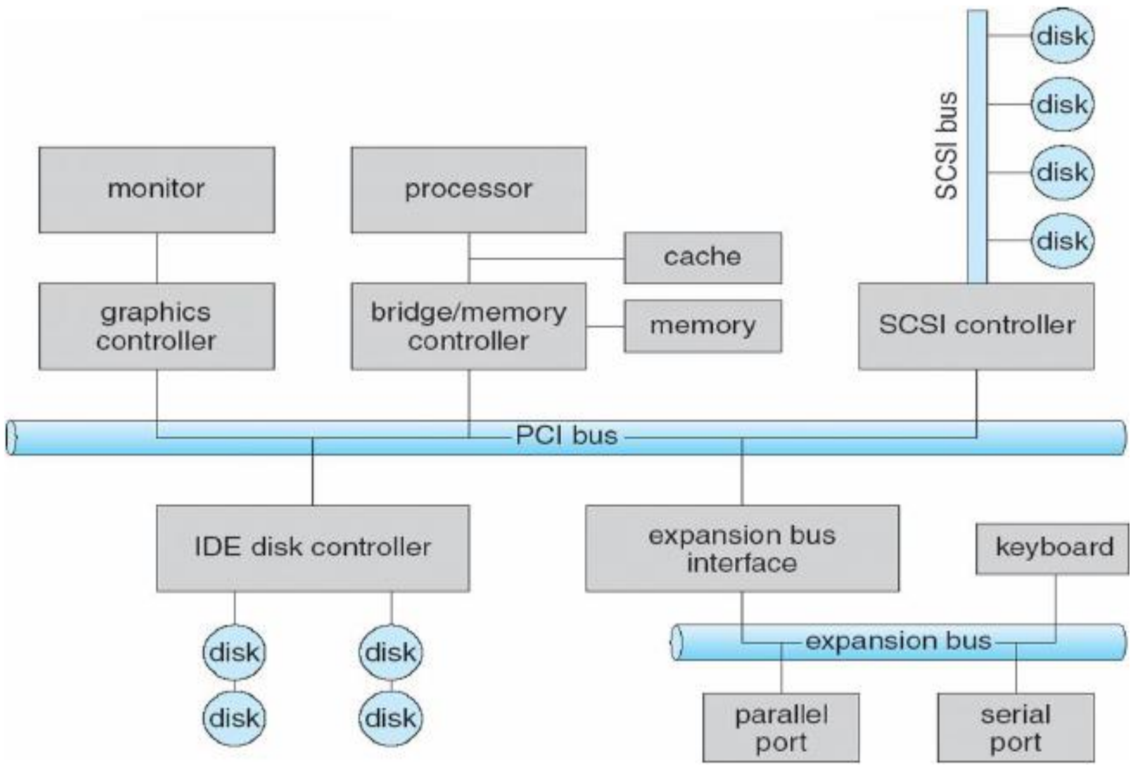
I/O Hardware #1



- Variety of I/O devices
- Common concepts
  - **Port**
    - Connection point for device (e.g., USB, parallel, serial, Ethernet)
  - **Bus**
    - **Peripheral** buses (e.g., PCI/PCIe)
    - **Expansion bus** connects relatively slow devices
  - **Device**
  - **Controller** (**host adapter**)
    - Electronics that operate port, bus, device
      - Sometimes integrated
      - Sometimes separate circuit board (host adapter)
    - Contains processor, microcode, private memory, bus controller, etc.

I/O Hardware #2

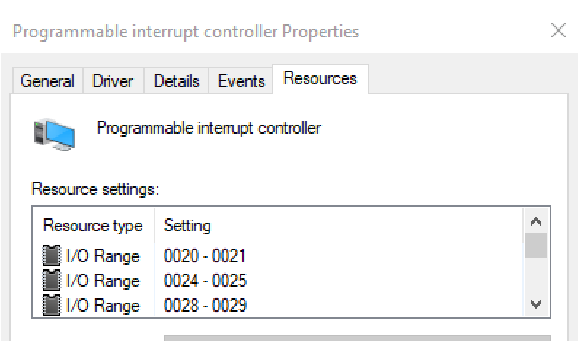
- Buses (cyan)
  - Handle the traffic between I/O devices and processor
- Examples
  - PCI/PCIe
    - Connects with high speed graphics, networking, etc.
    - Connects to low speed buses
  - SCSI
    - Used to be for fast devices with large bandwidth (disks, scanners, etc.)



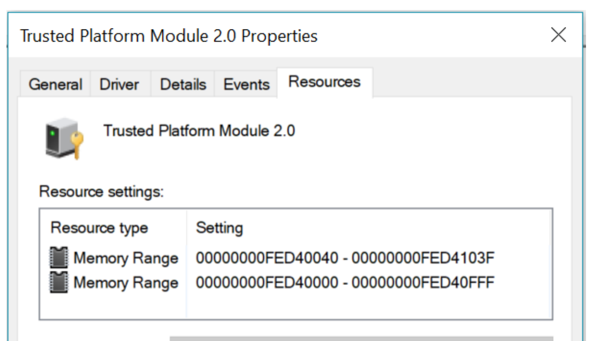
CPU to Device Communication

- Controllers have
  - **Registers** for data and control
  - **Buffers** (memory-like areas) mostly for data
- CPU **communicates** with devices by reading and writing in registers and buffers
- Communication methods
  - **IO Ports**
  - **Memory-mapped IO**
  - **Hybrid**

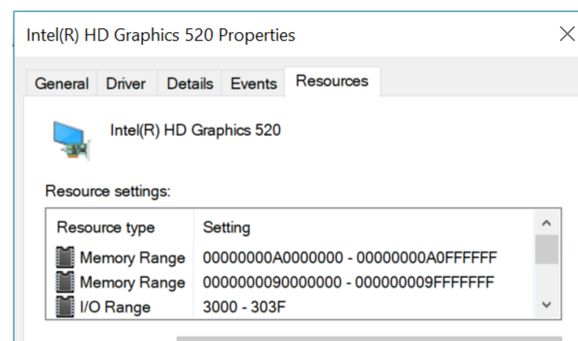
I/O Ports



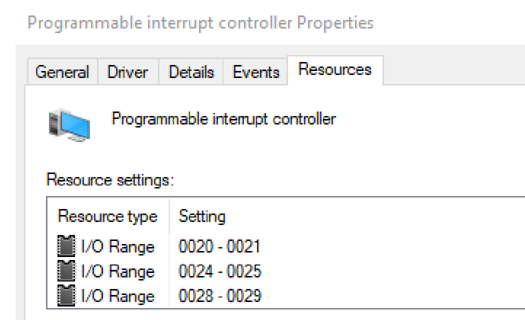
Memory Mapped I/O



Hybrid

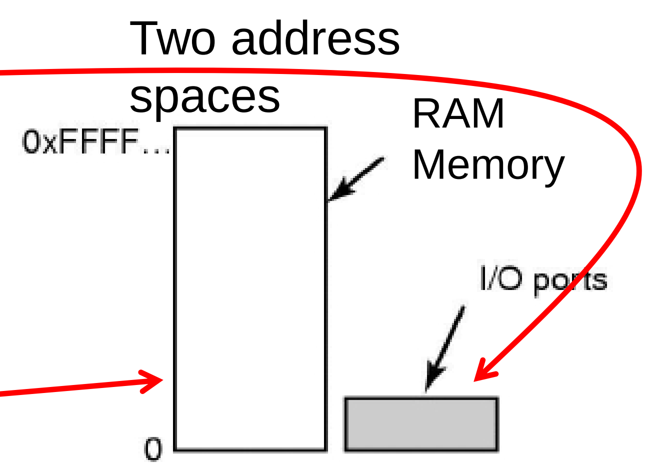


I/O Ports #1

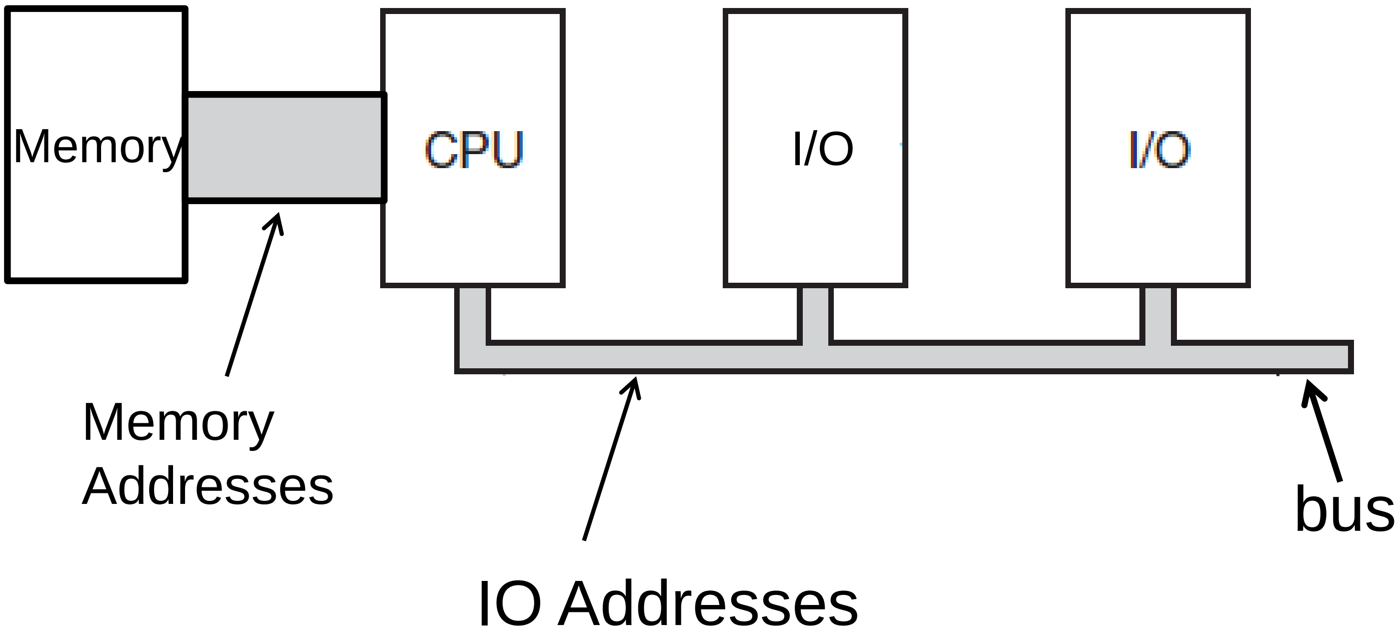


- Each control register an I/O port number
- Special instructions to access the I/O port space
  - CPU reads in from device I/O PORT to CPU register
    - IN REG, PORT
  - CPU writes to device I/O PORT from CPU register
    - OUT PORT, REG
- Instruction are privileged (OS kernel only)

- Separate **I/O port space** and **memory space**
  - I/O instructions
    - *IN R0, 4*
    - *OUT 4, R0*
  - Similar memory access instruction
    - *MOV R0, 4*
    - *MOV 4, R0*



I/O Ports #2



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

I/O Ports #3

antonio@antonio-VirtualBox: ~

File Edit View Search Terminal Help

```
antonio@antonio-VirtualBox:~$ cat /proc/ioports
0000-0000 : PCI Bus 0000:00
  0000-0000 : dma1
  0000-0000 : pic1
  0000-0000 : timer0
  0000-0000 : timer1
  0000-0000 : keyboard
  0000-0000 : keyboard
  0000-0000 : rtc_cmos
    0000-0000 : rtc0
  0000-0000 : dma page reg
  0000-0000 : pic2
  0000-0000 : dma2
  0000-0000 : fpu
  0000-0000 : 0000:00:01.1
    0000-0000 : ata_piix
  0000-0000 : 0000:00:01.1
    0000-0000 : ata_piix
  0000-0000 : 0000:00:01.1
    0000-0000 : ata_piix
  0000-0000 : vga+
  0000-0000 : 0000:00:01.1
    0000-0000 : ata_piix
0000-0000 : PCI conf1
0000-0000 : PCI Bus 0000:00
  0000-0000 : 0000:00:07.0
    0000-0000 : ACPI PM1a_EVT_BLK
    0000-0000 : ACPI PM1a_CNT_BLK
    0000-0000 : ACPI PM_TMR
    0000-0000 : ACPI GPE0_BLK
  0000-0000 : 0000:00:07.0
    0000-0000 : piix4_smbus
  0000-0000 : 0000:00:01.1
    0000-0000 : ata_piix
  0000-0000 : 0000:00:03.0
    0000-0000 : e1000
  0000-0000 : 0000:00:04.0
  0000-0000 : 0000:00:05.0
    0000-0000 : Intel 82801AA-ICH
  0000-0000 : 0000:00:05.0
    0000-0000 : Intel 82801AA-ICH
  0000-0000 : 0000:00:0d.0
    0000-0000 : ahci
  0000-0000 : 0000:00:0d.0
    0000-0000 : ahci
  0000-0000 : 0000:00:0d.0
```

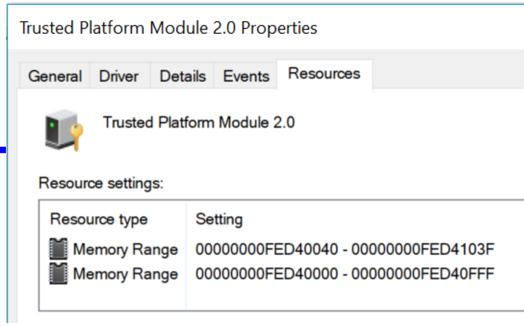
You must be root!

root@antonio-VirtualBox: /home/antonio

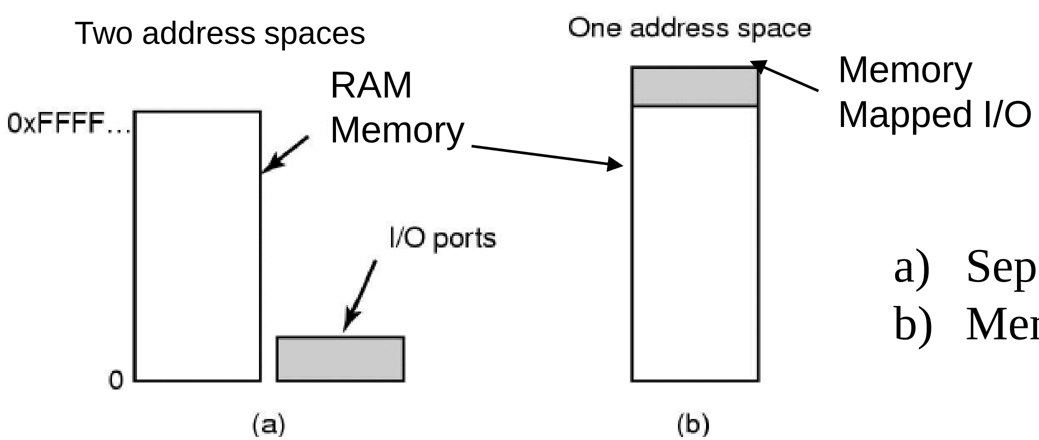
File Edit View Search Terminal Help

```
root@antonio-VirtualBox:/home/antonio# cat /proc/ioports
0000-0cf7 : PCI Bus 0000:00
  0000-001f : dma1
  0020-0021 : pic1
  0040-0043 : timer0
  0050-0053 : timer1
  0060-0060 : keyboard
  0064-0064 : keyboard
  0070-0071 : rtc_cmos
    0070-0071 : rtc0
  0080-008f : dma page reg
  00a0-00a1 : pic2
  00c0-00df : dma2
  00f0-00ff : fpu
  0170-0177 : 0000:00:01.1
    0170-0177 : ata_piix
  01f0-01f7 : 0000:00:01.1
    01f0-01f7 : ata_piix
  0376-0376 : 0000:00:01.1
    0376-0376 : ata_piix
  03c0-03df : vga+
  03f6-03f6 : 0000:00:01.1
    03f6-03f6 : ata_piix
0cf8-0cff : PCI conf1
0d00-ffff : PCI Bus 0000:00
  4000-403f : 0000:00:07.0
    4000-4003 : ACPI PM1a_EVT_BLK
    4004-4005 : ACPI PM1a_CNT_BLK
    4008-400b : ACPI PM_TMR
    4020-4021 : ACPI GPE0_BLK
  4100-410f : 0000:00:07.0
    4100-4108 : piix4_smbus
  d000-d00f : 0000:00:01.1
    d000-d00f : ata_piix
  d010-d017 : 0000:00:03.0
    d010-d017 : e1000
  d020-d03f : 0000:00:04.0
  d100-d1ff : 0000:00:05.0
    d100-d1ff : Intel 82801AA-ICH
  d200-d23f : 0000:00:05.0
    d200-d23f : Intel 82801AA-ICH
  d240-d247 : 0000:00:0d.0
    d240-d247 : ahci
  d248-d24b : 0000:00:0d.0
    d248-d24b : ahci
  d250-d257 : 0000:00:0d.0
```

Memory-mapped I/O #1

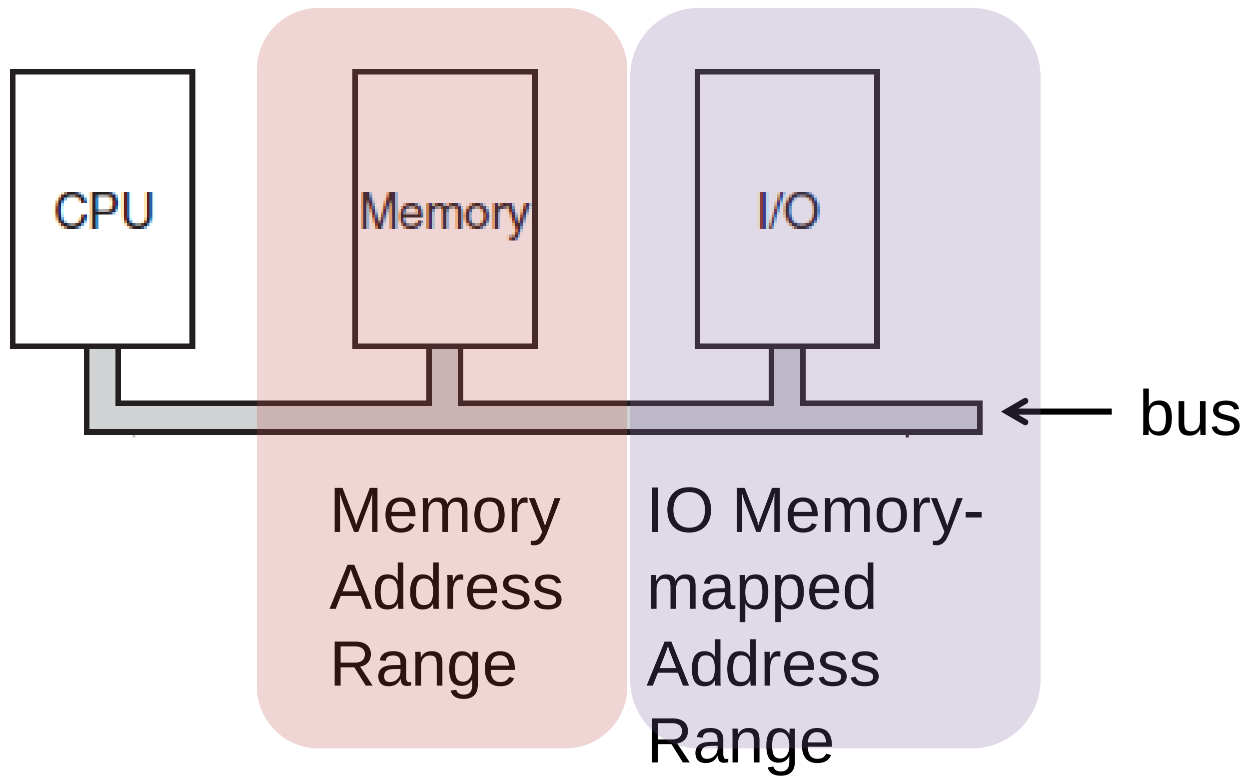


- All control registers and buffers into the memory space
- Each control register is assigned a unique memory address
  - There is no actual RAM memory for this address
- Such addresses may be at the top of the physical address space



a) Separate I/O and memory space
b) Memory-mapped I/O

Memory-mapped I/O #2



Example 0 … 32GB 127.999TB … 128TB

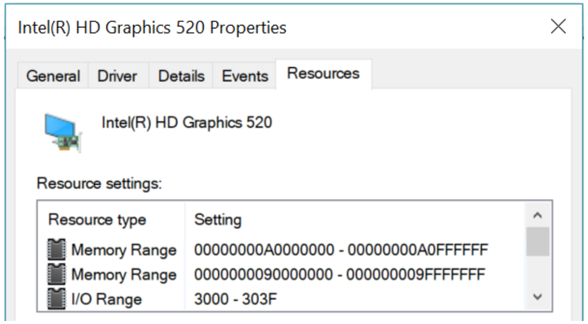
Memory-mapped I/O #3

```
antonio@antonio-VirtualBox: ~
File  Edit  View  Search  Terminal  Help
antonio@antonio-VirtualBox:~$ cat /proc/iomem
00000000-00000000 : Reserved
00000000-00000000 : System RAM
00000000-00000000 : Reserved
00000000-00000000 : PCI Bus 0000:00
00000000-00000000 : Video ROM
00000000-00000000 : Adapter ROM
00000000-00000000 : Reserved
  00000000-00000000 : System ROM
00000000-00000000 : System RAM
  00000000-00000000 : Kernel code
  00000000-00000000 : Kernel data
  00000000-00000000 : Kernel bss
00000000-00000000 : ACPI Tables
00000000-00000000 : PCI Bus 0000:00
  00000000-00000000 : 0000:00:02.0
  00000000-00000000 : 0000:00:03.0
    00000000-00000000 : e1000
  00000000-00000000 : 0000:00:04.0
    00000000-00000000 : vboxguest
  00000000-00000000 : 0000:00:04.0
  00000000-00000000 : 0000:00:06.0
    00000000-00000000 : ohci_hcd
  00000000-00000000 : 0000:00:0d.0
    00000000-00000000 : ahci
00000000-00000000 : Reserved
  00000000-00000000 : IOAPIC 0
00000000-00000000 : Local APIC
  00000000-00000000 : Reserved
00000000-00000000 : Reserved
00000000-00000000 : System RAM
antonio@antonio-VirtualBox:~$
```

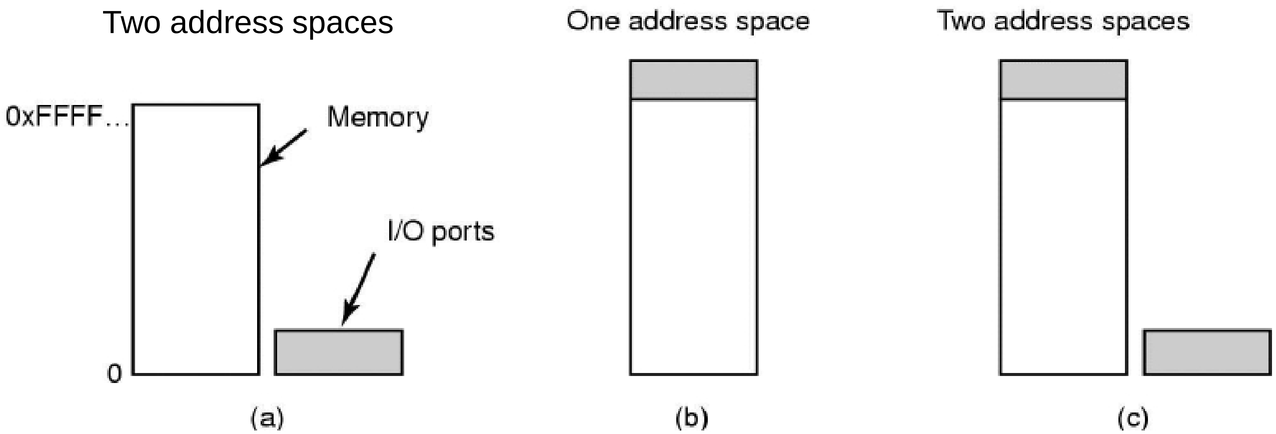
```
root@antonio-VirtualBox: /home/antonio
File  Edit  View  Search  Terminal  Help
root@antonio-VirtualBox:/home/antonio# cat /proc/iomem
00000000-00000fff : Reserved
00001000-0009fbff : System RAM
0009fc00-0009ffff : Reserved
000a0000-000bffff : PCI Bus 0000:00
000c0000-000c7fff : Video ROM
000e2000-000ef3ff : Adapter ROM
000f0000-000fffff : Reserved
  000f0000-000fffff : System ROM
00100000-dffeffff : System RAM
  20a00000-216031d0 : Kernel code
  216031d1-2206a43f : Kernel data
  222e2000-2253dfff : Kernel bss
dfff0000-dfffffff : ACPI Tables
e0000000-fdffffff : PCI Bus 0000:00
  e0000000-e1ffffff : 0000:00:02.0
  f0000000-f001ffff : 0000:00:03.0
    f0000000-f001ffff : e1000
  f0400000-f07fffff : 0000:00:04.0
    f0400000-f07fffff : vboxguest
  f0800000-f0803fff : 0000:00:04.0
  f0804000-f0804fff : 0000:00:06.0
    f0804000-f0804fff : ohci_hcd
  f0806000-f0807fff : 0000:00:0d.0
    f0806000-f0807fff : ahci
fec00000-fec00fff : Reserved
  fec00000-fec003ff : IOAPIC 0
fee00000-fee00fff : Local APIC
  fee00000-fee00fff : Reserved
fffc0000-ffffffff : Reserved
100000000-11fffffff : System RAM
root@antonio-VirtualBox:/home/antonio#
```

```
00001000-0009fbff : System RAM
00100000-dffeffff : System RAM
100000000-11fffffff : System RAM
```

Hybrid #1

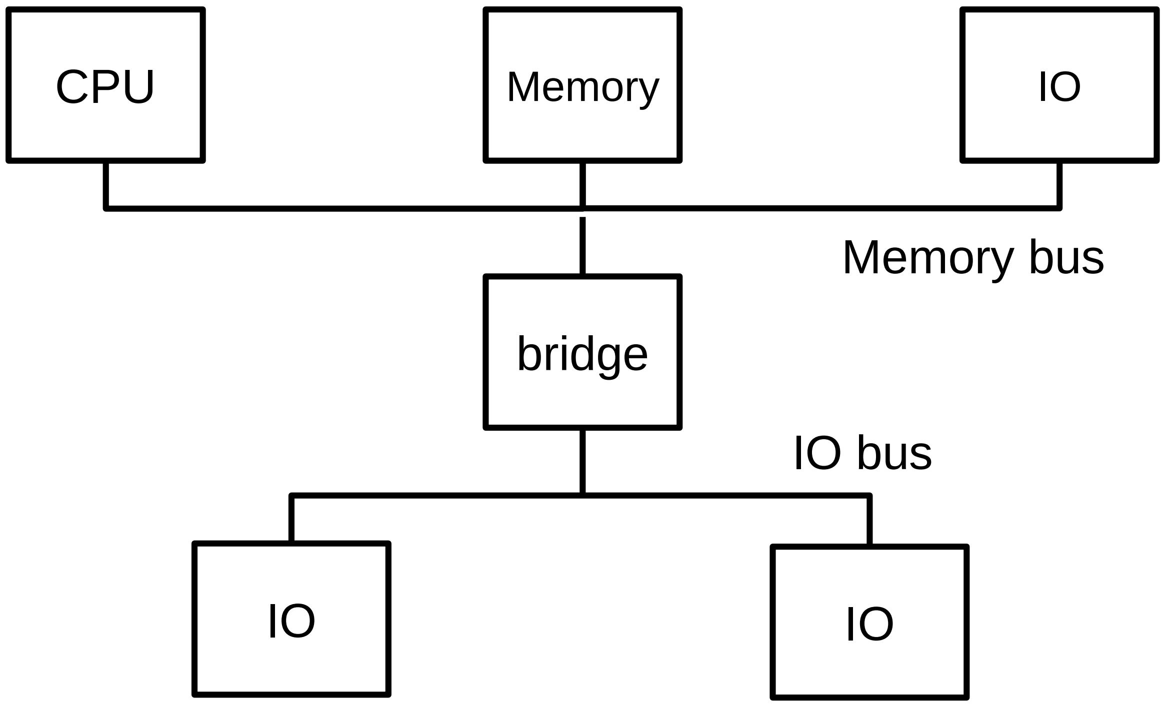


- I/O ports and memory-mapped IO
- Example
  - Memory-mapped I/O data buffers and separate I/O ports for the control registers
  - x86 CPUs, memory addresses 640K to 1M − 1 being reserved for device data buffers, in addition to I/O ports 0 to 64K − 1



a) Separate I/O and memory space
b) Memory-mapped I/O
c) Hybrid

Hybrid #2

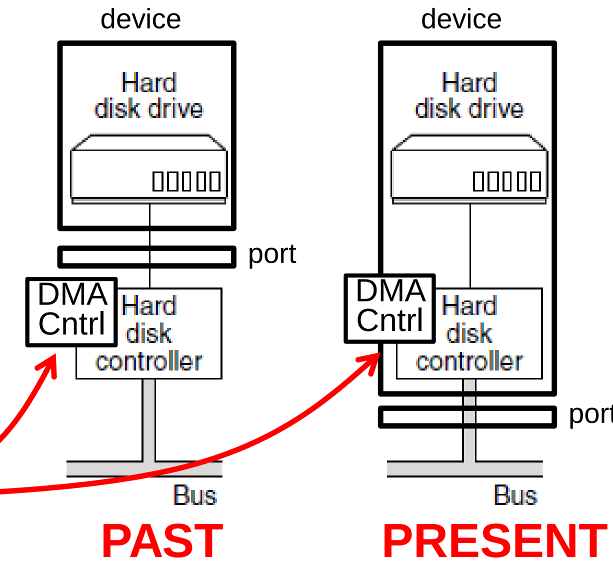


Offloaded Communication

- The CPU can request data from an I/O controller **one byte at a time**
  - Programmed IO (PIO)
    - (Previous slides)
  - This wastes CPU's time for large data transfers
    - Small data transfers are OK

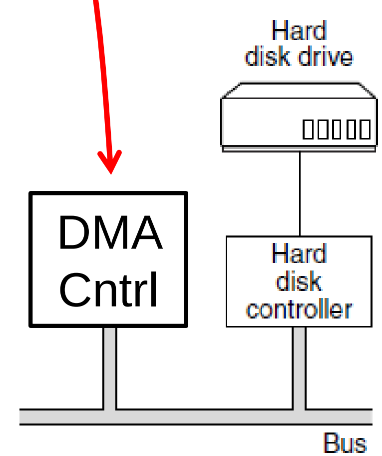
- CPU offloads data transfers
- **DMA (Direct Memory Access) controller** transfers data for the CPU
  - From/to an IO Device
  - Between IO Devices

Direct Memory Access



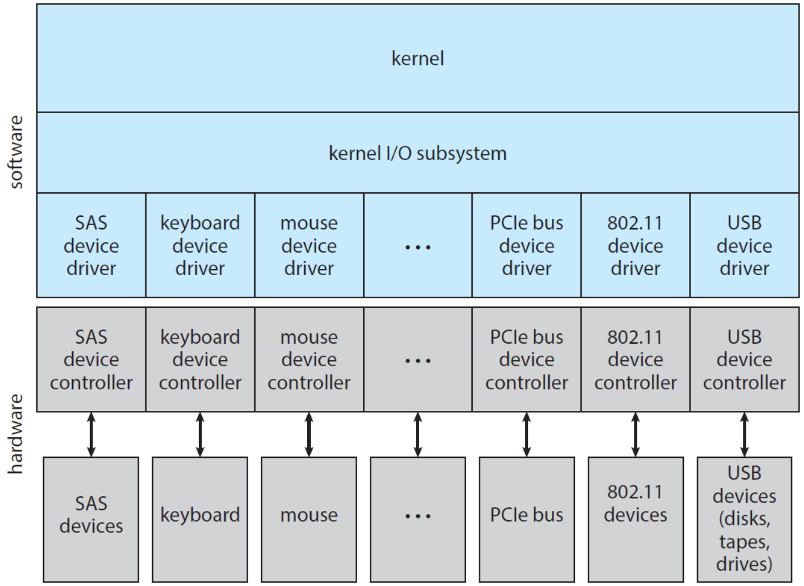
- Requires a **DMA controller**
  - On the device host controller
  - On the motherboard

- **DMA controller** contains registers to be read/written by the software
  - Memory address register
  - Byte count register
  - Control registers to
    - Direction of the transfer
    - Transfer unit
    - Byte burst size
    - …



OS Device Drivers

- Great variety of devices
  - Each device vendor/model its specs
- OS deals with IO devices in a **standard** and **uniform way**
  - Abstraction
  - Encapsulation
  - Software layering
- Use specific interface (file)
- Encapsulate the differences in devices by **device drivers classes**
  - Each OS its standards



- Example
  - An application can open a file without knowing what kind of disk it is
  - Independently of the disk technology

Characterizing IO 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

IO Services Provided by the OS

- Kernel **IO subsystem** services
  - Available to applications and to other parts of the OS

- Management of the name space for files and devices
- Access control to files and devices
- Operation control (for example, a modem cannot seek())
- File-system space allocation
- Device allocation
- Buffering, caching, and spooling
- I/O scheduling
- Device-status monitoring, error handling, and failure recovery
- Device-driver configuration and initialization
- Power management of I/O devices

Putting Everything Together: Life Cycle of an IO Request