ARCOS Group

uc3m | Universidad Carlos III de Madrid

Lesson 6
Input/Output Systems

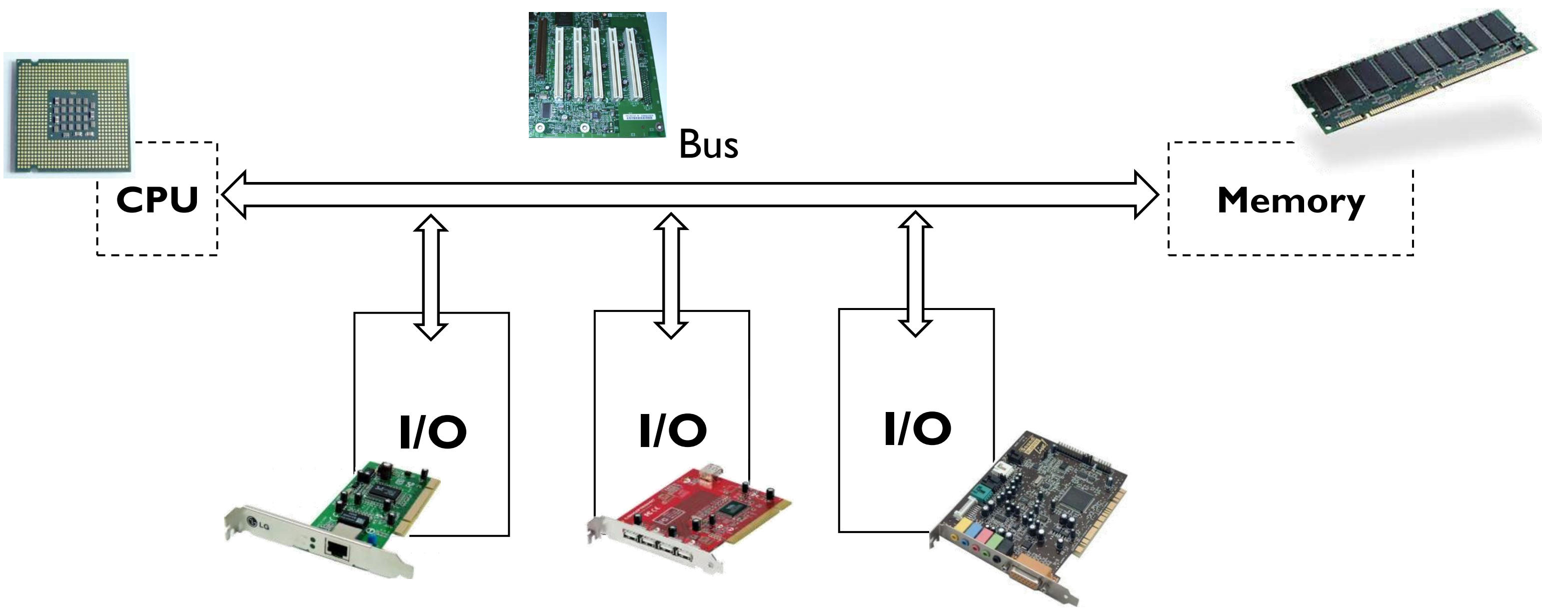
Computer Structure

Bachelor in Computer Science and Engineering

Contents

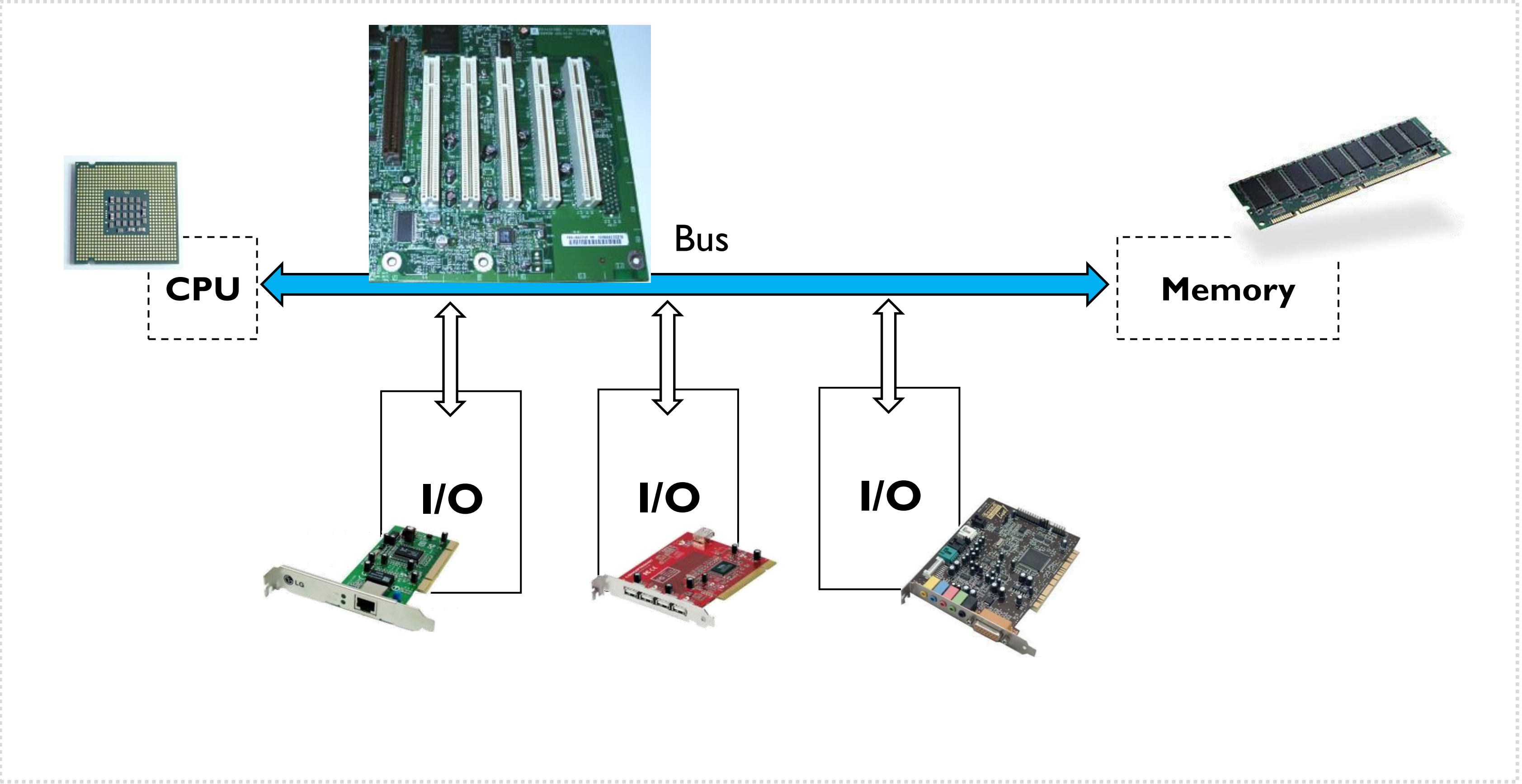
Contents

Introduction



Introduction: bus

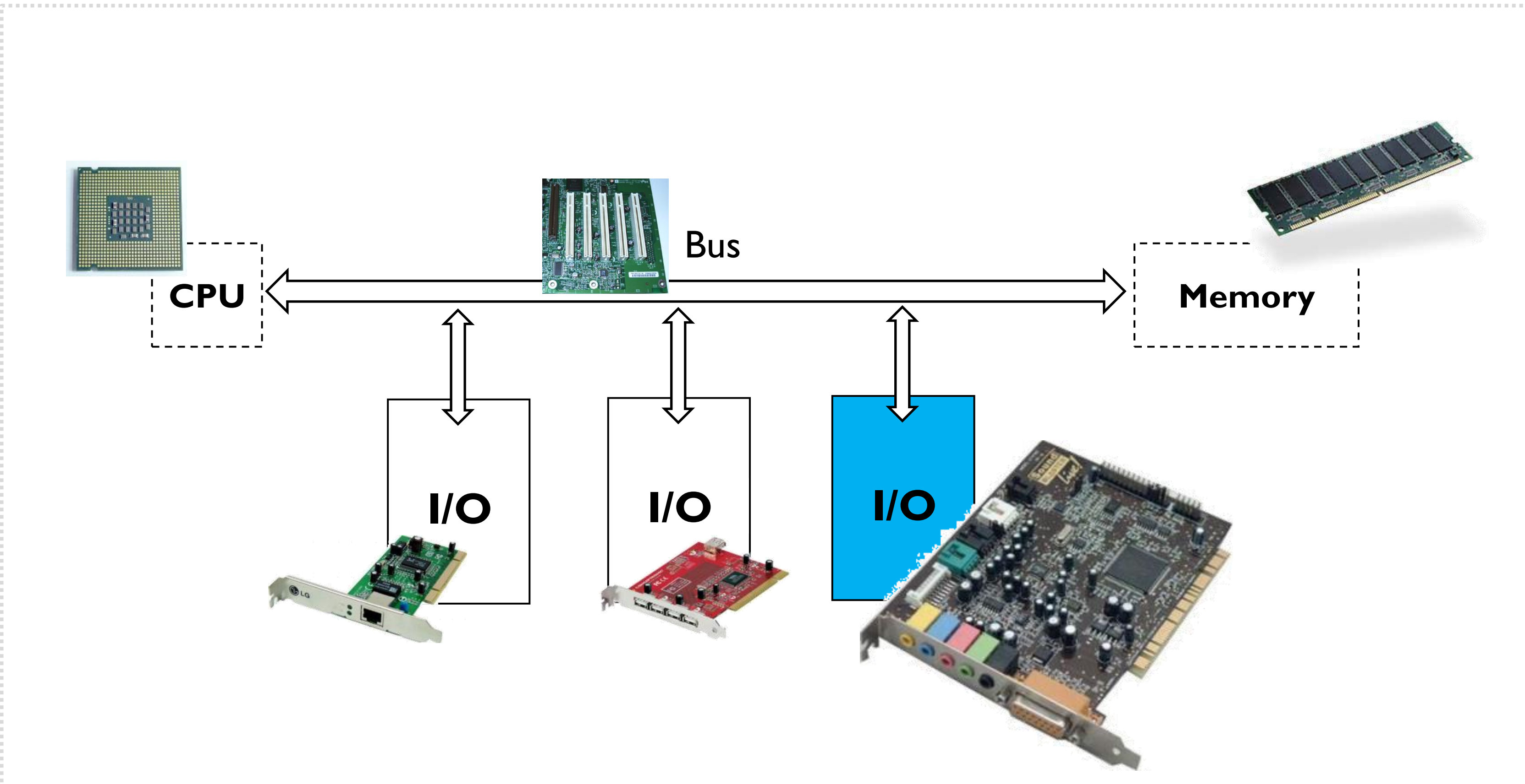
- What an interconnection bus is



Félix García Carballeira, Alejandro Calderón Mateos

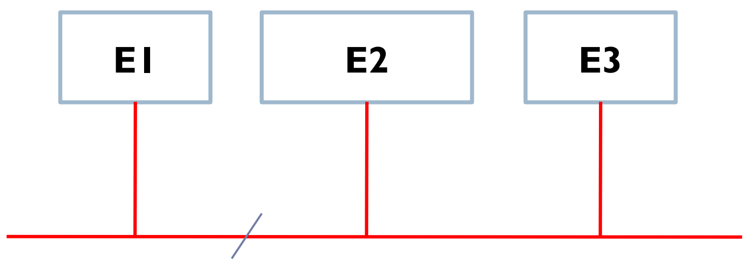
Introduction: I/O

- What a peripheral is
- What an input/output module is
- How data is accessed from peripherals



Contents

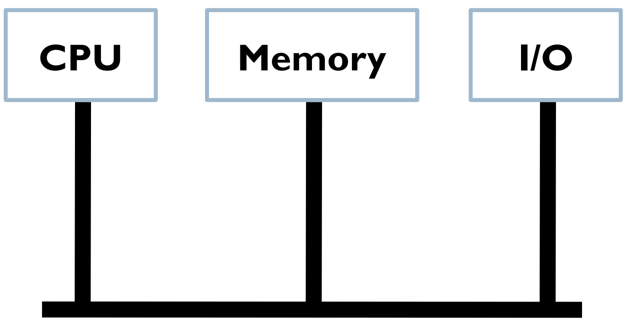
Bus



- A bus is a communication path between two or more devices.
- It consists of several bit transmission lines.
- Shared medium, univocal.

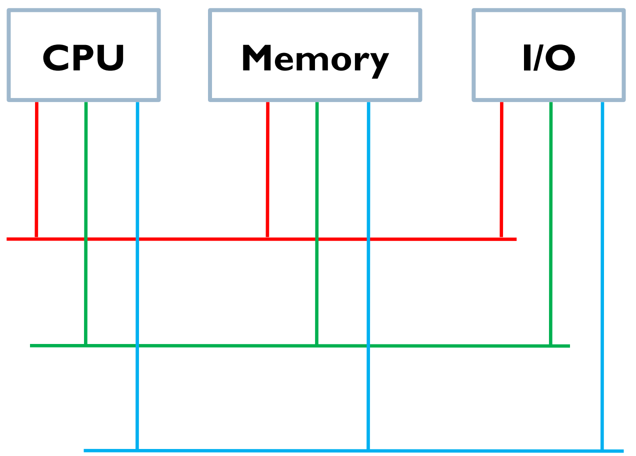
- Allows to transmit several bits between two elements connected to it

System bus



- **System bus**
  - Connects the main components of the computer
  - It represents the union of three buses:
    - Control
    - Addresses
    - Data

Buses



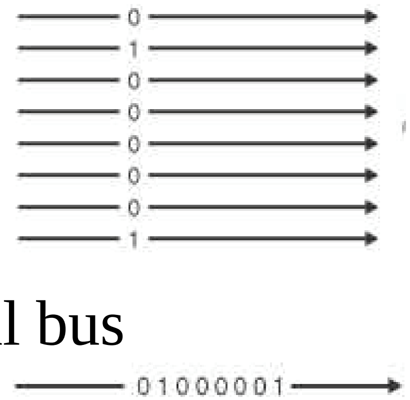
- Data bus
  - Transmits data
  - Its width and speed have a great influence on the performance
- Address bus
  - Memory addresses and I/O devices
  - Its width determines the maximum memory capacity
- Control bus
  - Control and timing signals

Characteristics of a bus

- Bus width: determines the number of bits that can be transmitted simultaneously

Parallel bus

Serial bus



- Frequency: clock frequency with which it can operate
- Transfer rate: number of bytes per clock cycle
- Bandwidth (transfer rate): transmitted bytes per second
  - Transfer rate X frequency

Exercise

- Calculate the bandwidth in MBps of a 32-bit bus with a frequency of 66 MHz

Exercise (solution)

- Calculate the bandwidth in MBps of a 32-bit bus with a frequency of 66 MHz

$$Bandwidth = \frac{32\ bits \times 66\ \ MHz}{8\ \ bits\ per\ byte} = \frac{32 \times 66 \cdot 10^6}{8} = 264 \text{ MBps}$$

Arbitration method (bus protocol)

- Determines which of the elements connected to the bus can access the bus
  - Centralized scheme: a bus controller grants the use of the bus
    - When an element wants to access the bus, it requests permission from the controller through the control lines (BUSRQ)
    - When the bus is free the controller grants the use (BUSACK)
  - Distributed scheme: each element connected to the bus includes an access control logic that allows the joint use of the bus (access protocol)

Synchronous and asynchronous buses

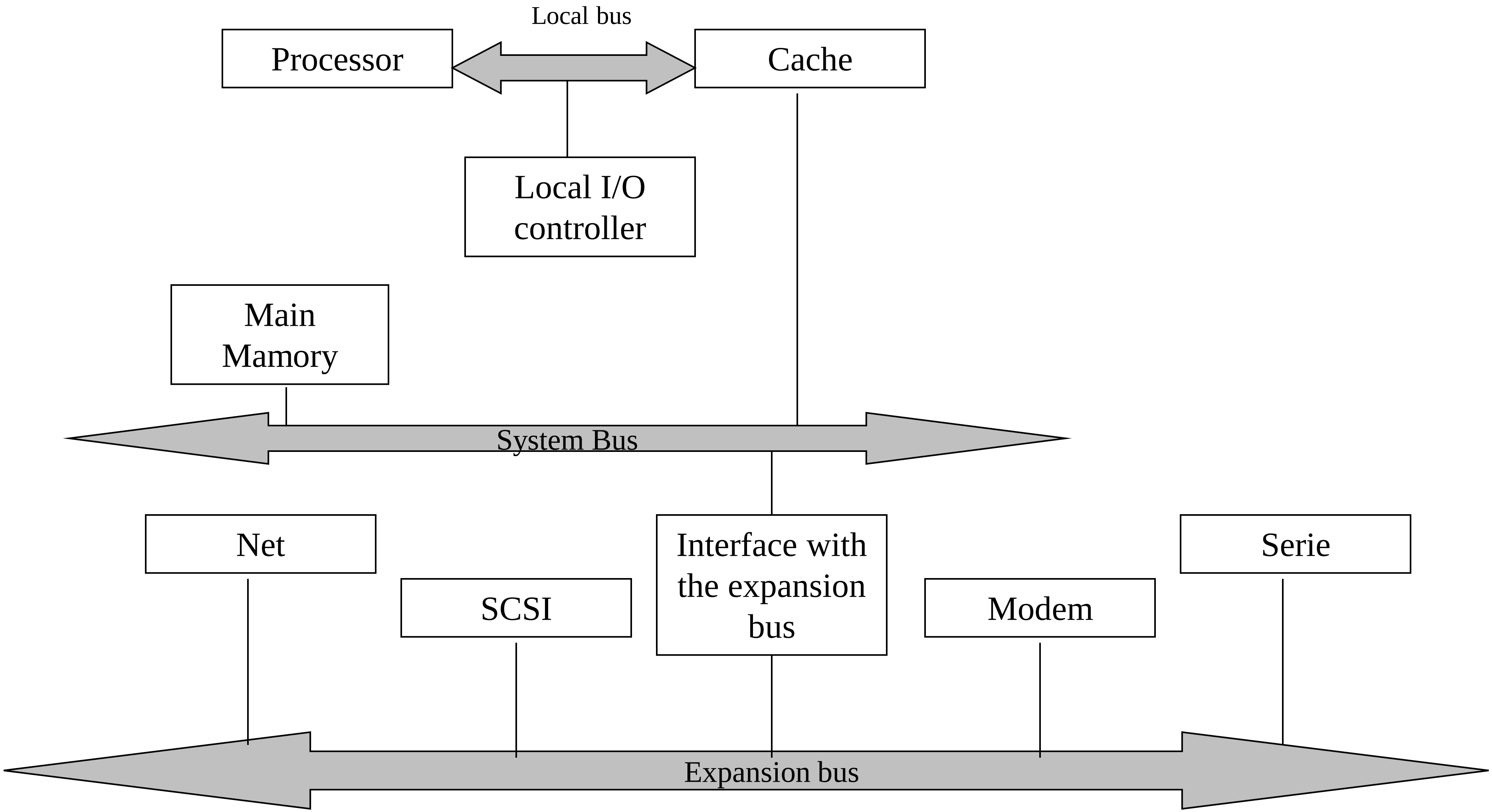
- A synchronous bus is governed by a clock signal and a communication protocol set to the operation of the clock
  - Fast
  - All devices connected to it must operate at the same clock frequency
- An asynchronous bus does not use a clock, the communication is done by sending orders through the control lines of the bus

Bus hierarchies

- Problem:
  - The more devices connected to the bus, the longer the propagation delay.
  - As the number of transfer requests increases, a bottleneck can occur.

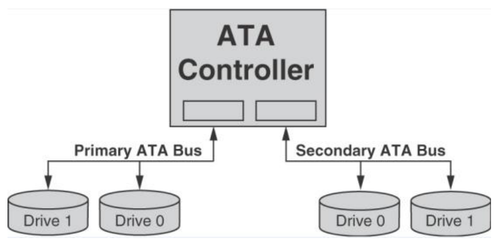
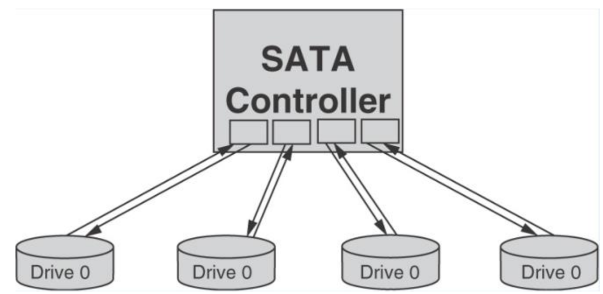
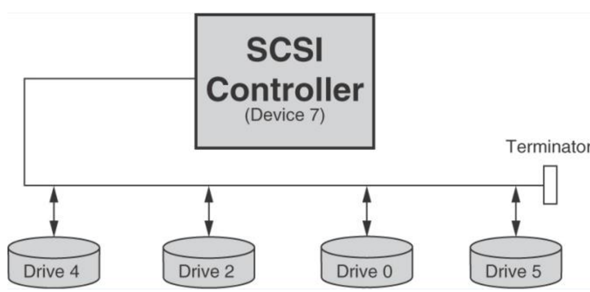
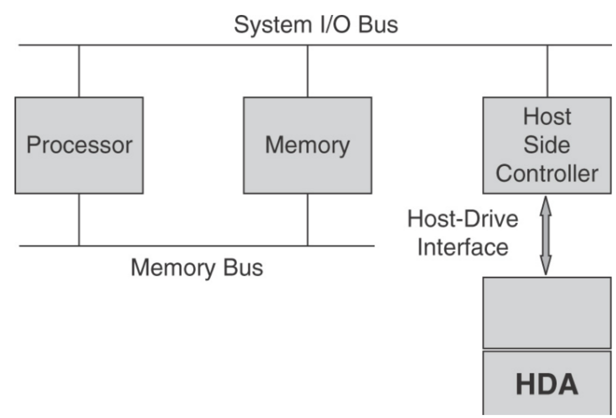
- Solutions:
  - Increase data transmission speed with wider buses.
  - Use more data buses, organized hierarchically.

Bus hierarchies
Bus diagram in a typical computer system



Disks Controllers

Memory Systems
Cache, DRAM, Disk
Bruce Jacob, Spencer Ng, David Wang
Elsevier

	~1980 … ~2010	~2010 … now
Normal PC		
High-end PC		

Curiosity:
USB family

	Transfer (per sec.)	Introduction
USB4	40 Gbps	2019
USB 3.2	20 Gbps	2017
USB 3.0	600 MB/s	2010
USB 2.0	60 MB/s	2000
USB 1.0	1.5 MB/s and 187 KB/s	1996

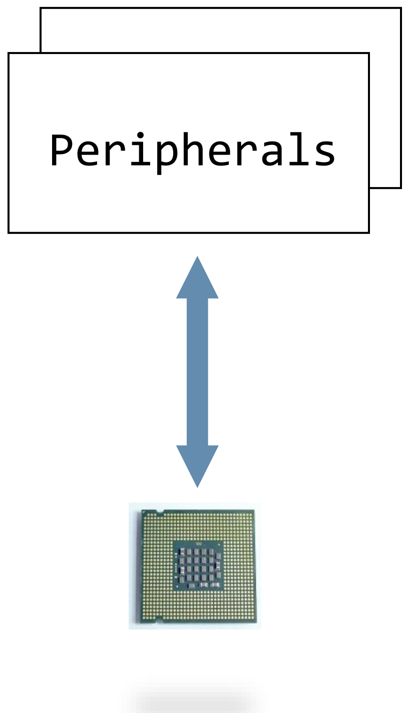
	Song / Pic	256 Flash	USB Flash	SD-Movie	USB Flash	HD-Movie
	4 MB	256 MB	1 GB	6 GB	16 GB	25 GB
USB 1.0	5.3 sec	5.7 min	22 min	2.2 hr	5.9 hr	9.3 hr
USB 2.0	0.1 sec	8.5 sec	33 sec	3.3 min	8.9 min	13.9 min
USB 3.0	0.01 sec	0.8 sec	3.3 sec	20 sec	53.3 sec	70 sec

http://www.unp.co.in/f140/comparison-of-usb-3-0-port-with-usb-2-0-and-usb-1-0-a-70063/

Contents

1. Introduction
2. Buses
   - Structure and operation
   - Bus hierarchy
3. Peripheral
   - Concept and types of peripherals
   - General structure of a peripheral
   - I/O modules
4. Case study: hard disk drive and solid-state drives
5. I/O interaction: I/O techniques

Peripheral concept



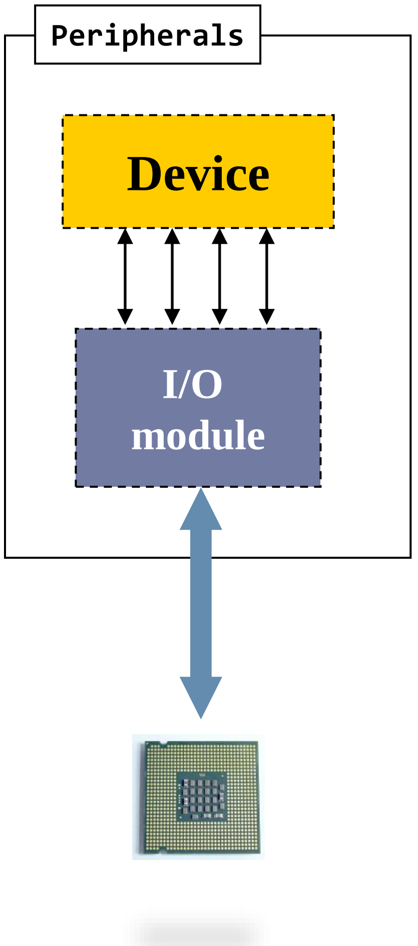
- Peripheral:
  - Any external device that connects to a processor through the input/output (I/O) modules.
  - They allow storing information or communicating the computer with the outside world.

Classification of peripherals (by use)

- **Communication**:
  - Human-computer
    - (Terminal) keyboard, mouse, …
    - (Printer) plotter, scanner, …
  - Computer-computer
    - Modem, network adapter
  - Physical environment
    - (read/action) x (analogical/digital)
- **Storing**:
  - Direct access (disks, DVD, …)
  - Sequential access (tapes)

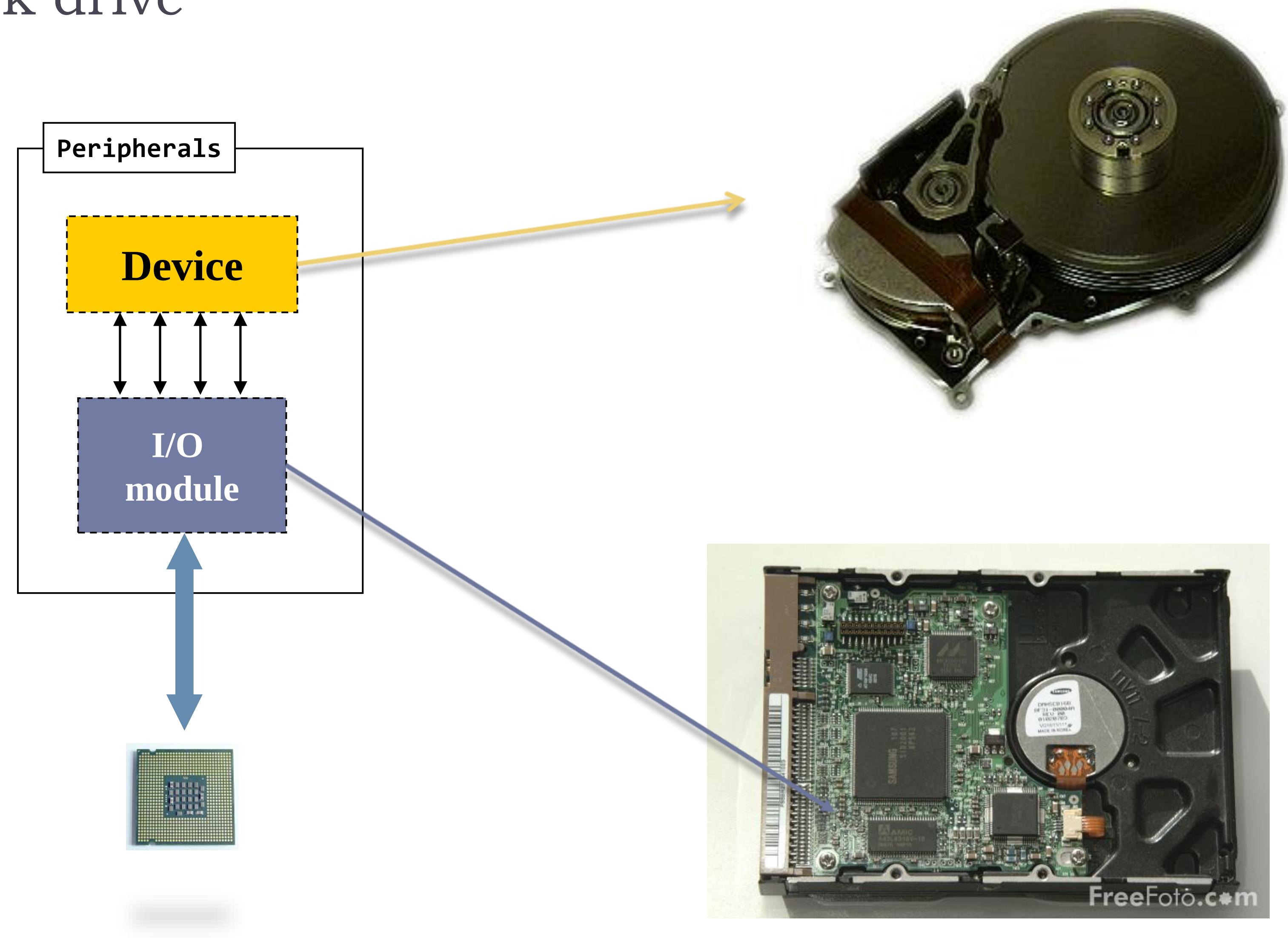
https://sites.google.com/site/ordenylimpiezapa/planeamiento-del-buen-orden-y-la-limpieza/dispositivos-de-almacenamiento

General structure of a peripheral

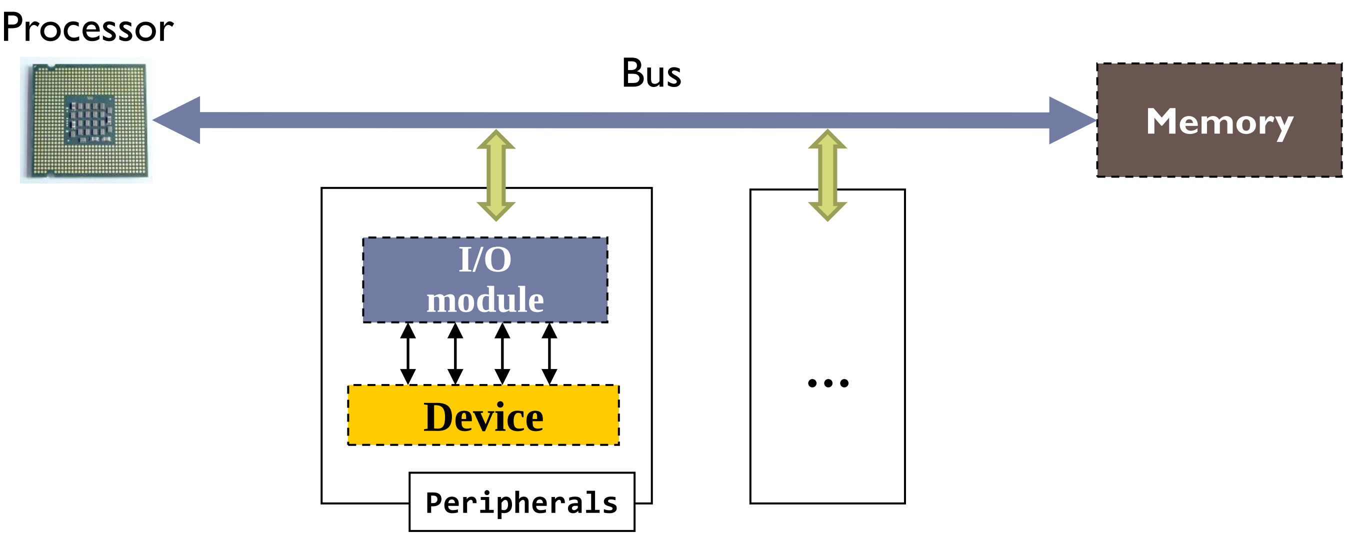


- Consisting of:
  - Device
    - Hardware that interacts with the environment
  - I/O module
    - Also called controller or I/O unit
    - Interface between the device and the processor, which hides the particularities of the processor.

Example:
Disk drive

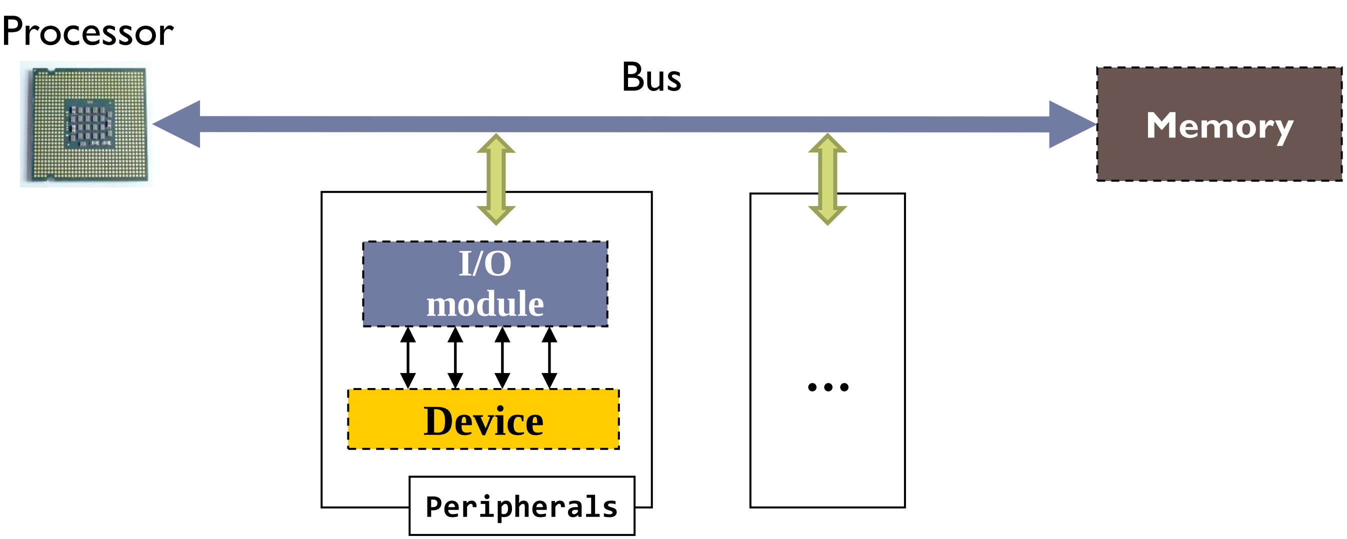


I/O module



- **I/O modules** perform the connection among the peripheral devices and the processor (or the memory)

I/O module: why there are necessary?

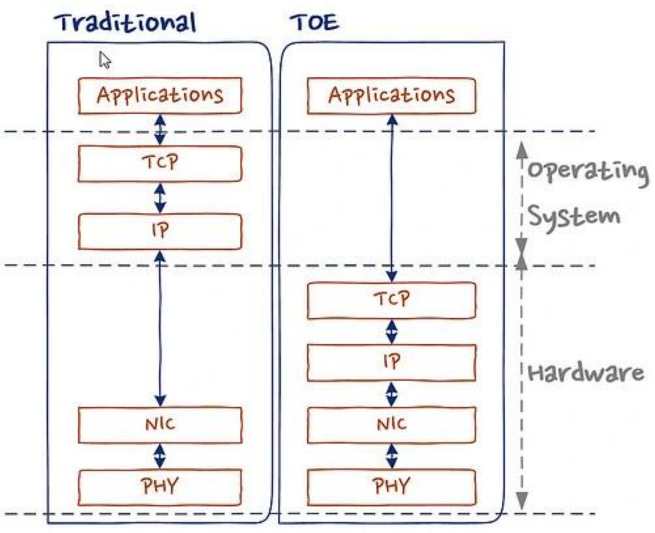


- Wide variety of peripherals.
  - **Peripherals are 'peculiar'.**
- Data transfer rate of peripherals is much slower than memory or CPU ones.
  - **Peripherals are 'very slow'.**
- Formats and word sizes of peripherals different from those of the computer to which they are connected.

I/O module: range of possible tasks

- I/O module common tasks:
  - Control and timing
  - Error detection
  - Processor/device communication
  - Data buffering
  - Etc.

- I/O module types by complexity:
  - **I/O controller** or **device driver**: simpler module, which requires the CPU to have detailed control of the device.
  - **Channel I/O** or **I/O processor**: handles most of the processing details.

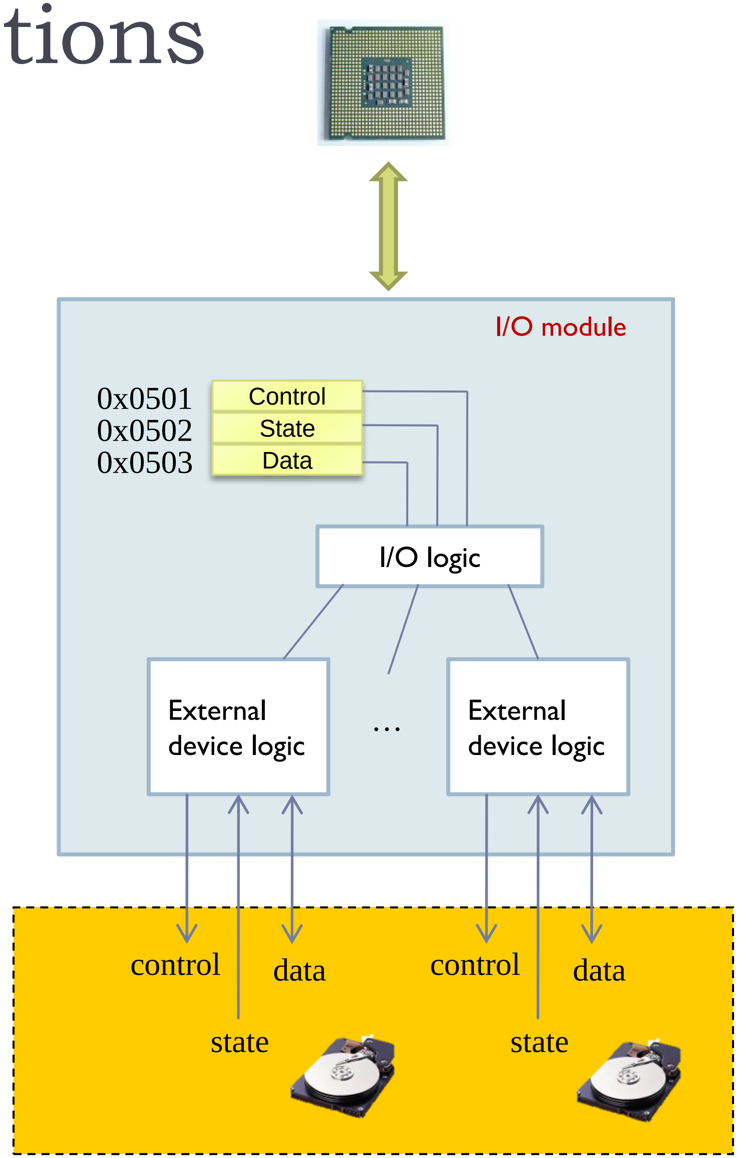


I/O controller **I/O processor**

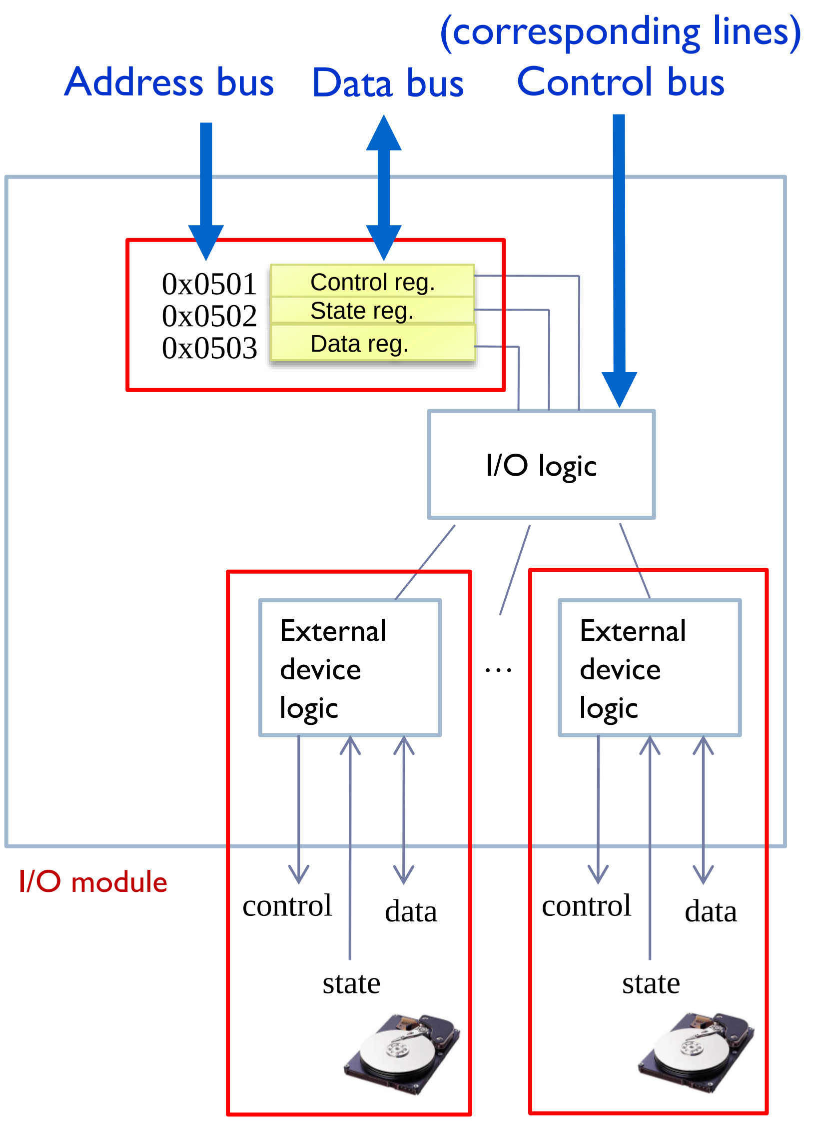
https://embeddist.wordpress.com/2015/06/02/network-performance-of-soc-with-tcpip-offload-engine-w7500-3/

I/O module: main functions

- Attending to the CPU:
  - Order decoding
  - Status information
  - Control and timing
    - E.g.: data to M.M.
- Control peripheral(s):
  - Communication with peripherals
  - Error detection
  - Temporary data storage
    - peripheral->processor

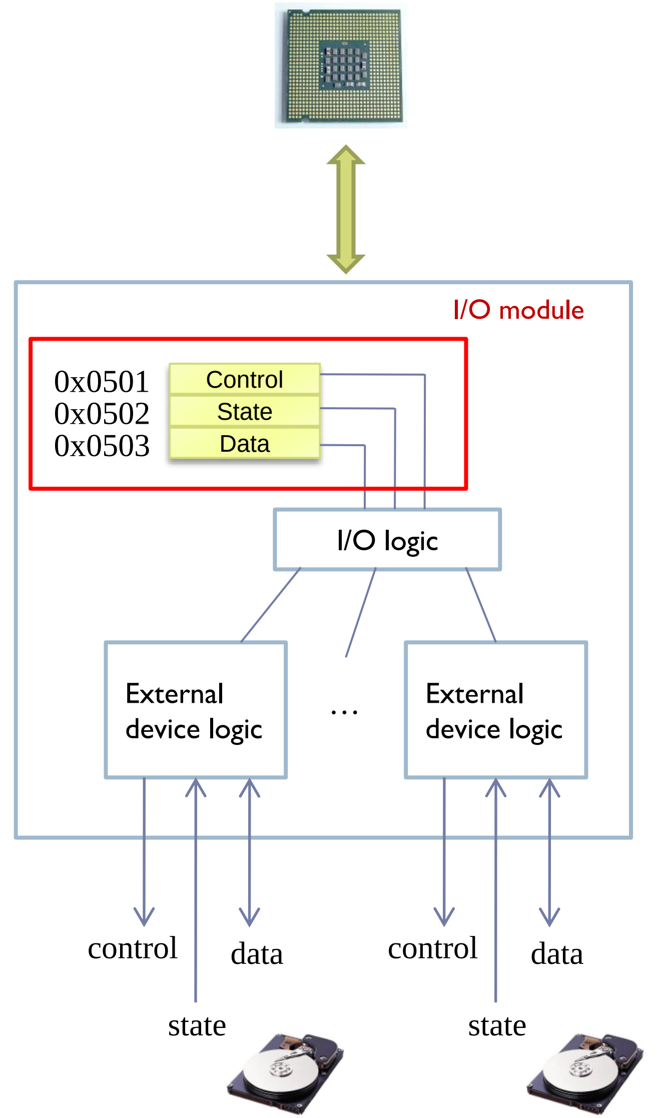


Simplified I/O module model



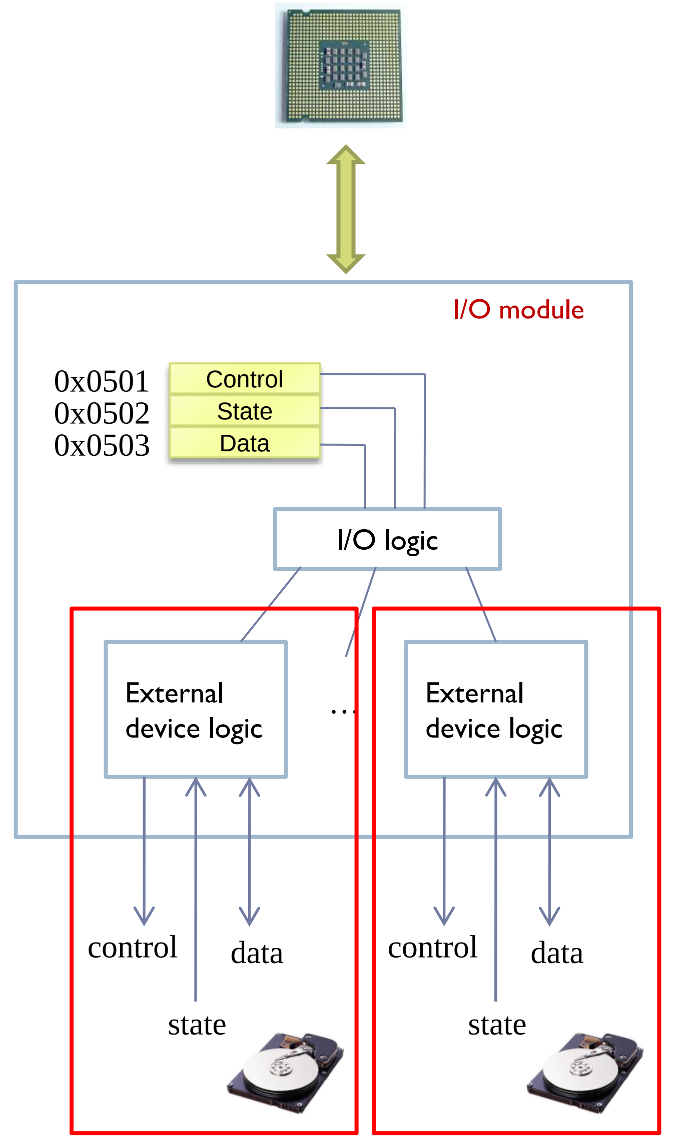
Simplified I/O module model

- Interaction between processor and I/O module through 3 registers:
  - The control register
    - Commands for the peripheral
  - The state register
    - Status of the last command
  - The data register
    - Data exchanged processor/peripheral



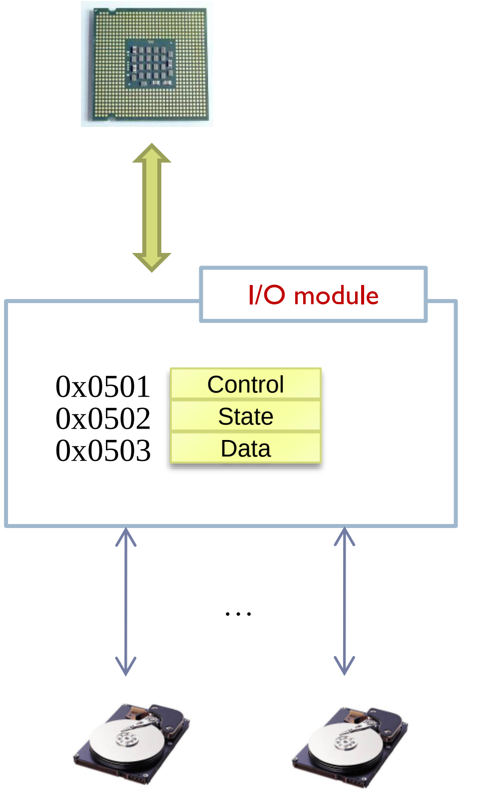
Simplified I/O module model

- Interaction between peripheral and I/O module:
  - Data signals (lines):
    for transferring information
  - State signals (lines):
    information about the device
    - Examples:
      - New data available
      - Peripheral on/off
      - Peripheral busy
      - Peripheral up and running
      - Error in last command
      - …
  - Control signals (lines):
    to control the peripheral/device
    - Examples:
      - Power on/off
      - Skip page in a printer
      - Seek in a hard disk
      - …



I/O module: main characteristics

- **Transfer unit**
  - Block
  - Character
- **Addressing**
  - Memory-mapped I/O
  - Port-mapped I/O
- **I/O techniques**
  - Programmed I/O
  - Interrupt I/O
  - DMA



Characteristics (1/3): Transfer unit

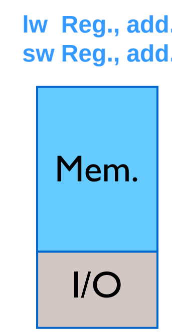
- **Block devices**:
  - Unit: block of bytes
  - Access: sequential or random
  - Operations: `read`, `write`, `seek`, …
  - Examples: "tapes" and disks

- **Character devices**:
  - Unit: chars (ASCII, Unicode, etc.)
  - Access: sequential to characters
  - Operations : `get`, `put`, ….
  - Example: terminals, printers, etc.

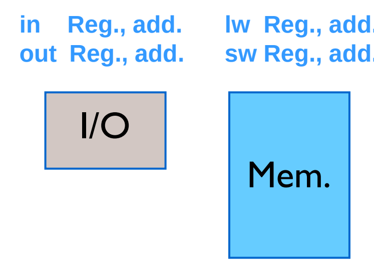
Characteristics (2/3):
I/O addressing

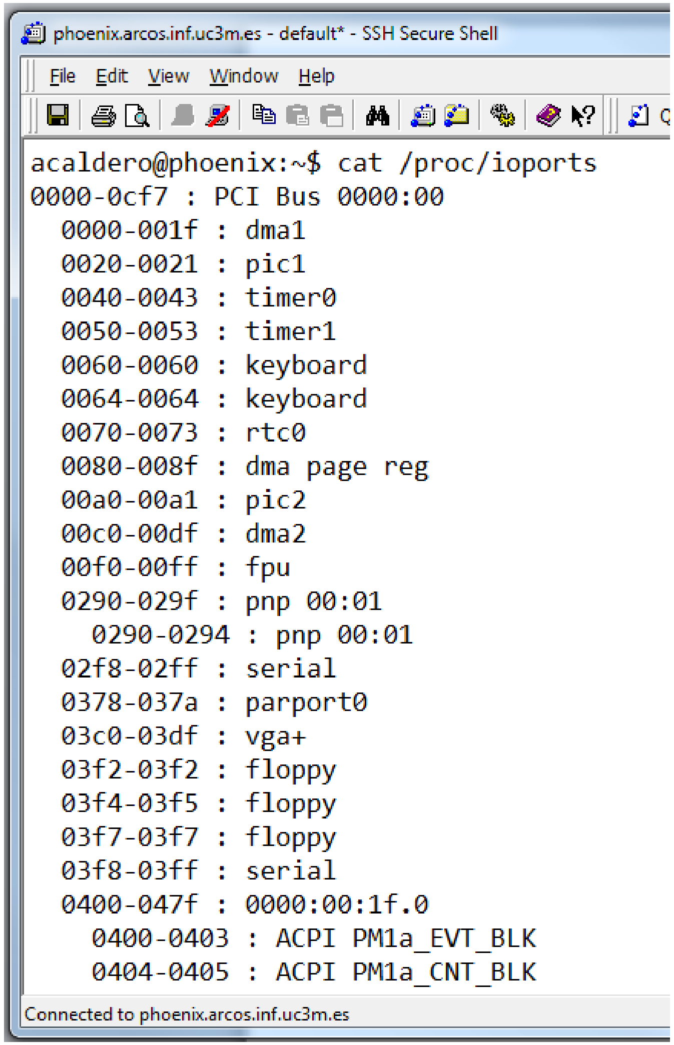
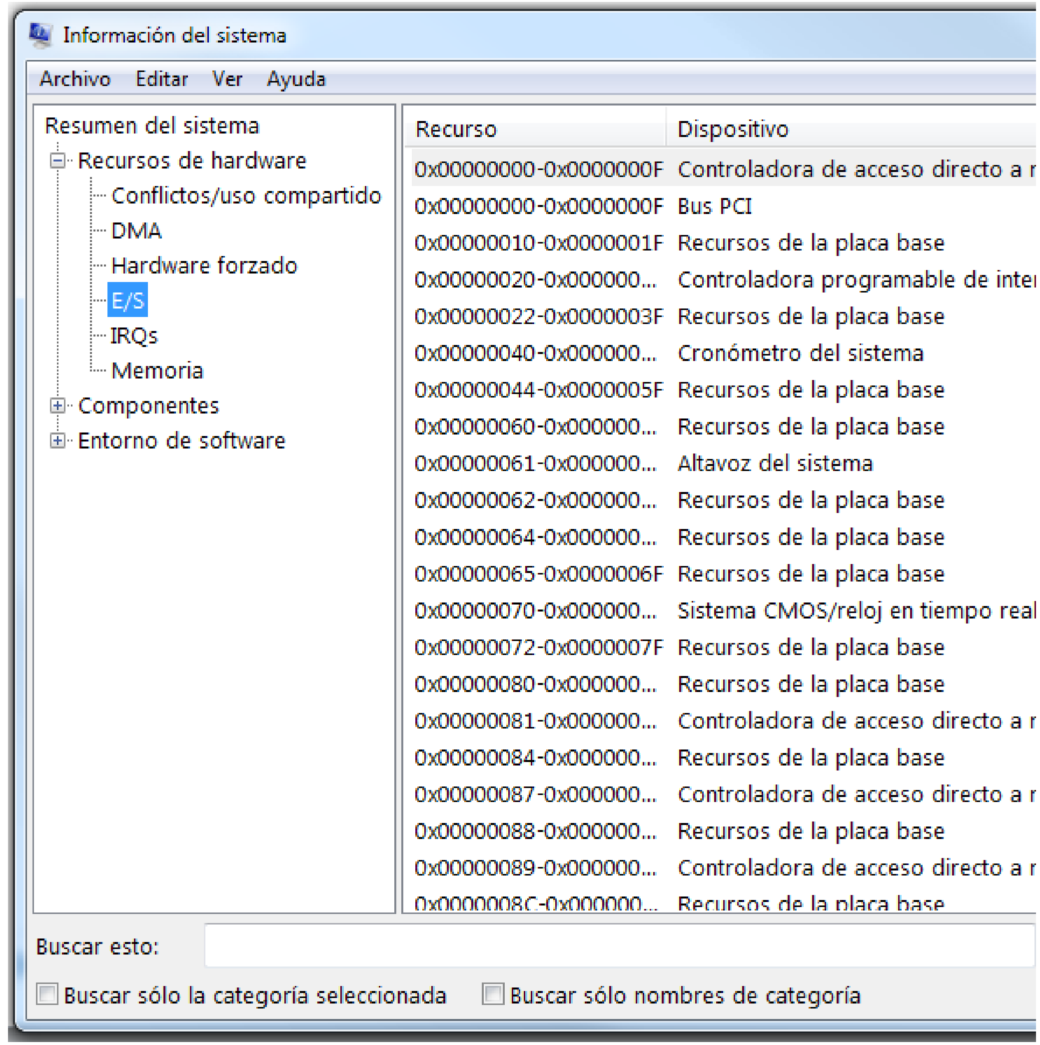
- Transfer unit
  - Block
  - Character
- Addressing
  - Memory-mapped I/O
  - Port-mapped I/O
- I/O techniques
  - Programmed I/O
  - Interrupt I/O
  - DMA

- **Memory-mapped I/O (MMIO)**
  - I/O registers are mapped in memory using a set of memory addresses for these registers.
  - Same machine instructions for memory and I/O:
    - lw $a0, label2_disk1
      - Load in the processor register "$a0" the value stored in the I/O register identified by a given address "label2_disk1"
    - sw $a0, label_disk2
      - To write an item in an I/O register from the I/O module

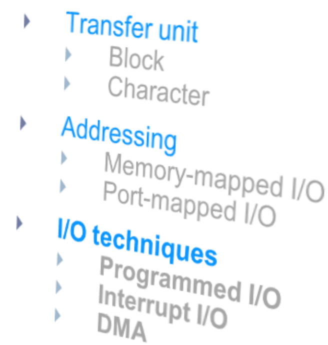


- **Port-mapped I/O (PMIO):**
  - I/O address space is isolated from memory address space.
  - Special privileged machine instructions (~ to lw/sw):
    - IN $a0, label2_disk1
      - Load in the processor register "$a0" the value stored in the I/O register identified by a given address "label2_disk1"
    - OUT $a0, label_disk2
      - To write an item in an I/O register from the I/O module

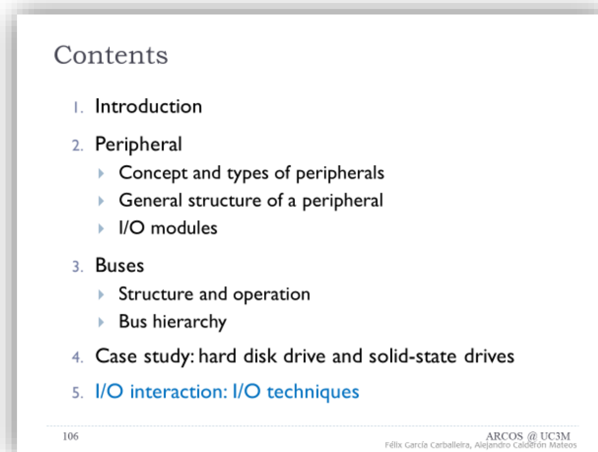


Linux	Windows
	

Characteristics (3/3):
I/O techniques



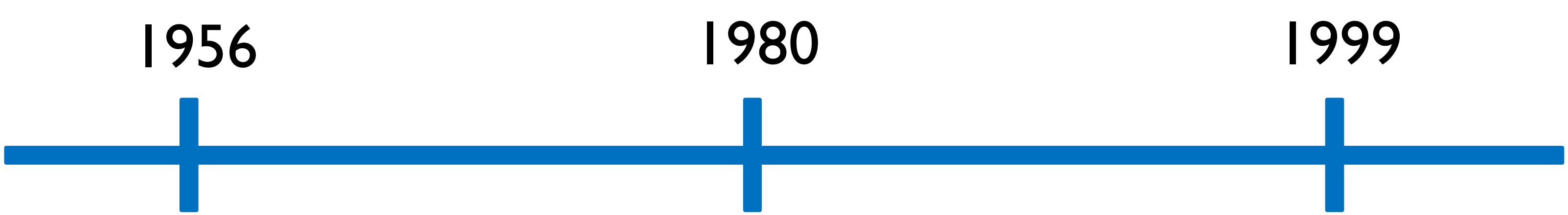
- I/O techniques:
  Processor and I/O_module interaction
  - **Programmed I/O**
  - **Interrupt I/O**
  - **DMA(Direct Memory Access) I/O**


Contents

Contents

A little bit of history…

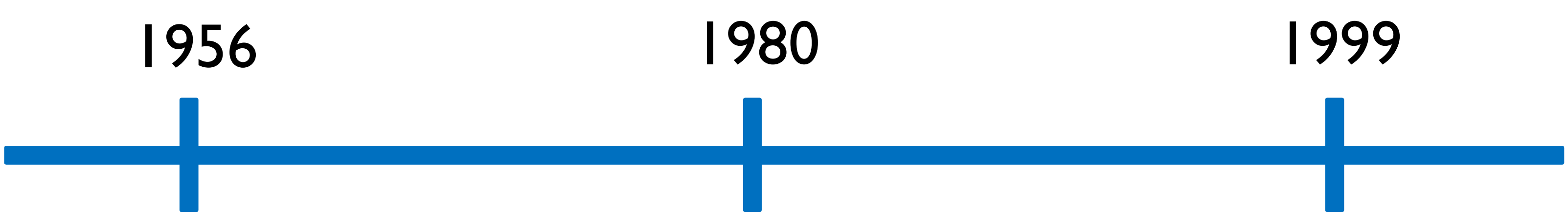
- First hard disk introduced in 1956
  - It was called IBM RAMAC 305
    - 50 aluminum disks of 61 cm (24") diameter
    - 5 MB of data
    - Spun at 3,600 revolutions per minute (RPM)
    - Had a transfer speed of 8.8 Kbps
    - 35 000$ per year rental
    - Weighed about one ton



A little bit of history…

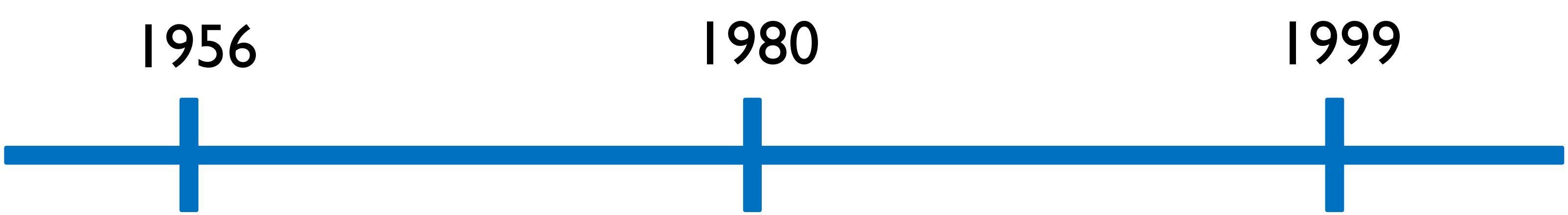

Sept 13, 1956 - IBM, today unveiled its new Ramac 305 super computer to the anxiously awaiting business, governmental and military world. The machine, with the first disk drive storage in the industry, is capable of handling electronic data faster and in greater quantities than any prior computer of its size. Seen here at the San Jose, Notre Dame Development Lab, demonstrating its capabilities for the press, is IBM employee, Wanda Little.

- IBM RAMAC 305 (1956)
  - 50 aluminum disks of 61 cm (24") diameter
  - 5 MB of data
  - 3,600 RPM
  - transfer speed: 8.8 Kbps
  - 35 000$ per year rental



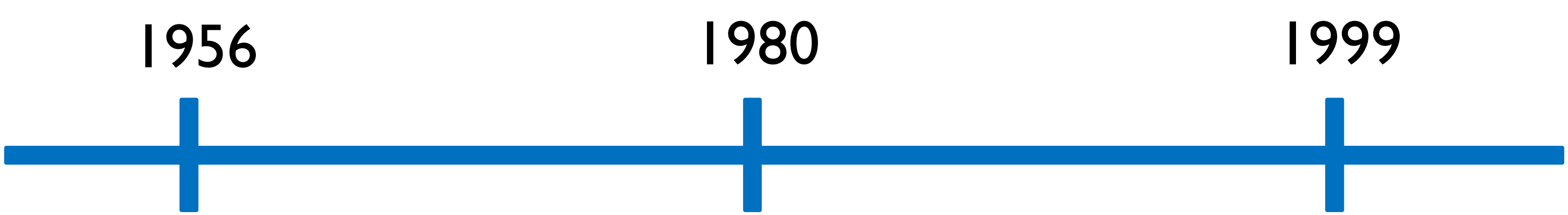
A little bit of history…

- In 1980 appeared the first 5 ¼" disk
  - 5 MB
  - ~4 500 $

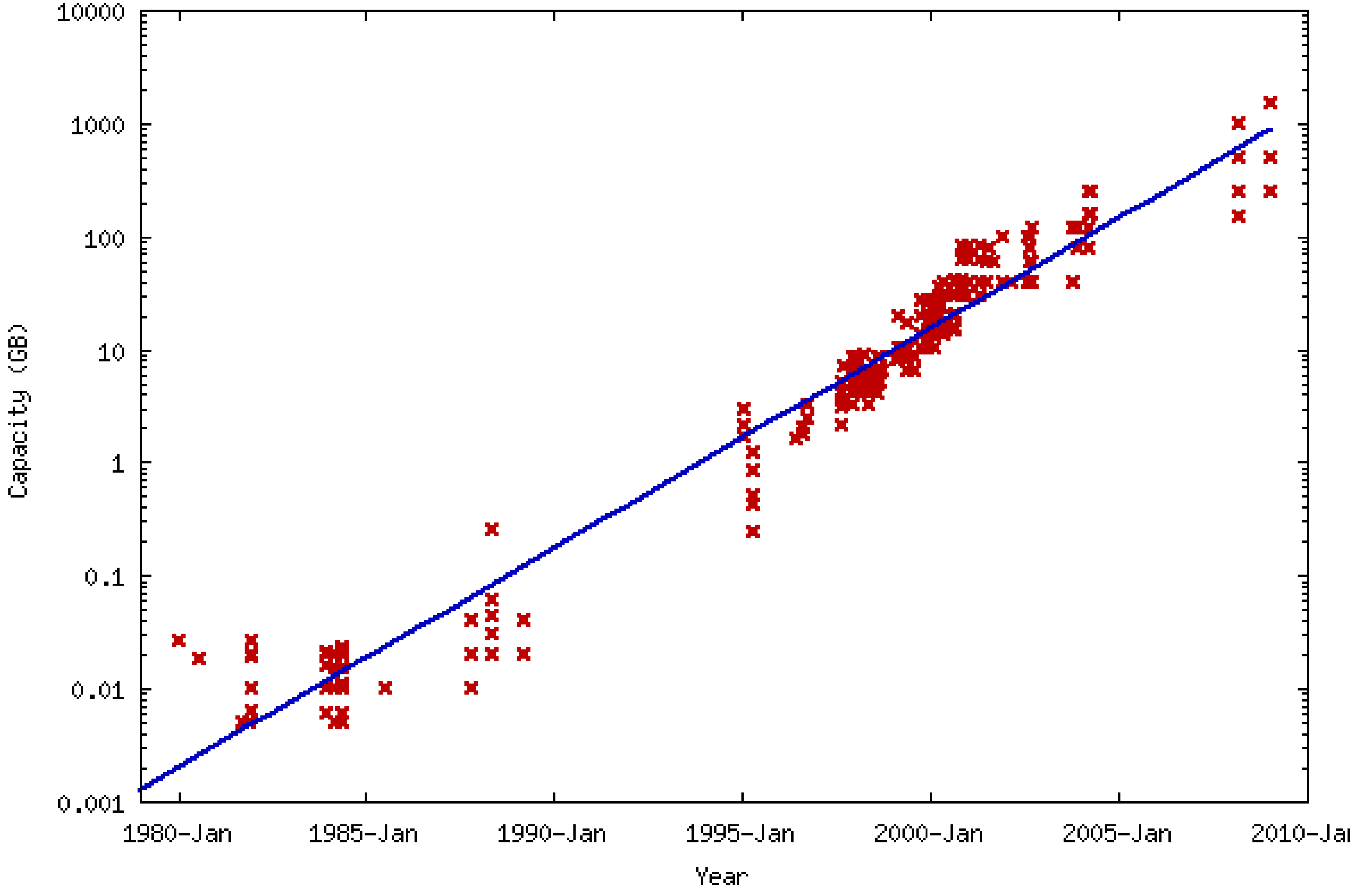


A little bit of history…

- In 1997 appeared the first disk with 15 000 RPM
- In 1999 is introduced the Microdrive
  - IBM+Hitachi, 170 MiB
  - The 2005 microdrive reach 6 GiB



Evolution…



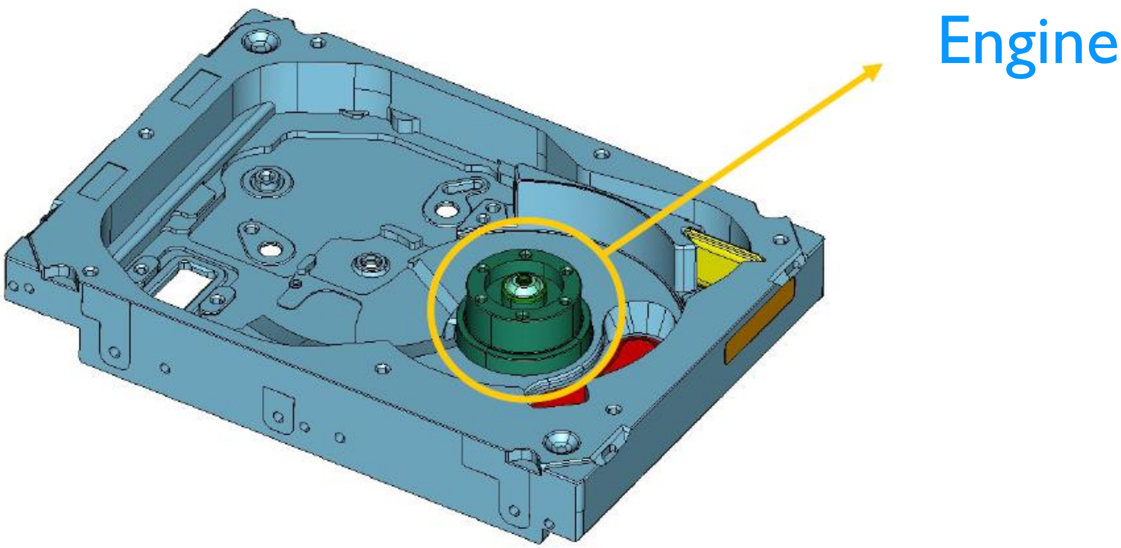
http://en.wikipedia.org/wiki/File:Hard_drive_capacity_over_time.svg

Evolution…

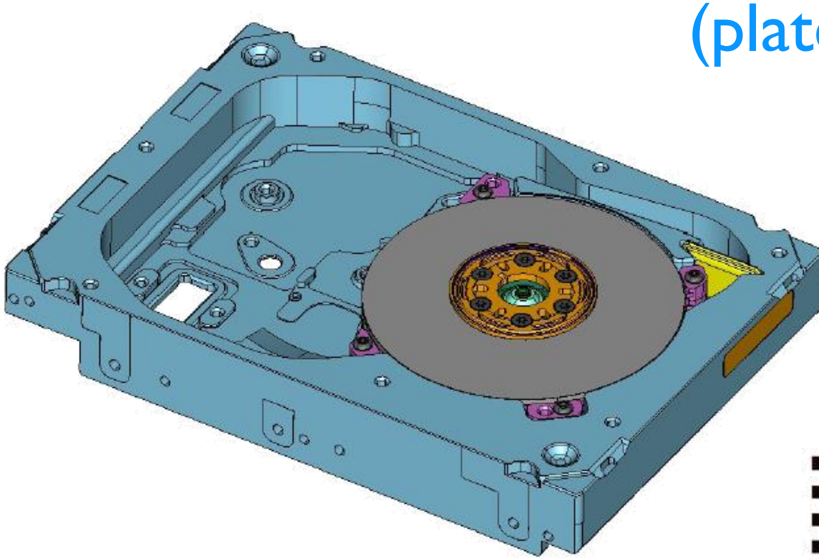
	Annual growth rate
Capacity	1.93 / year
Cost	0.60 / year
Performance	0.05 / year

https://www.pingdom.com/blog/amazing-facts-and-figures-about-the-evolution-of-hard-disk-drives/

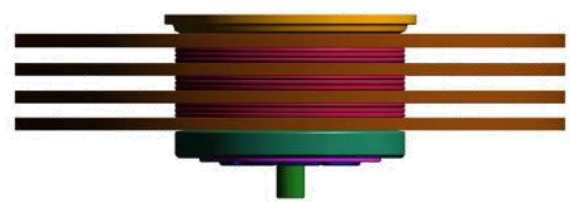
Anatomy of a hard disk drive



Anatomy of a hard disk drive

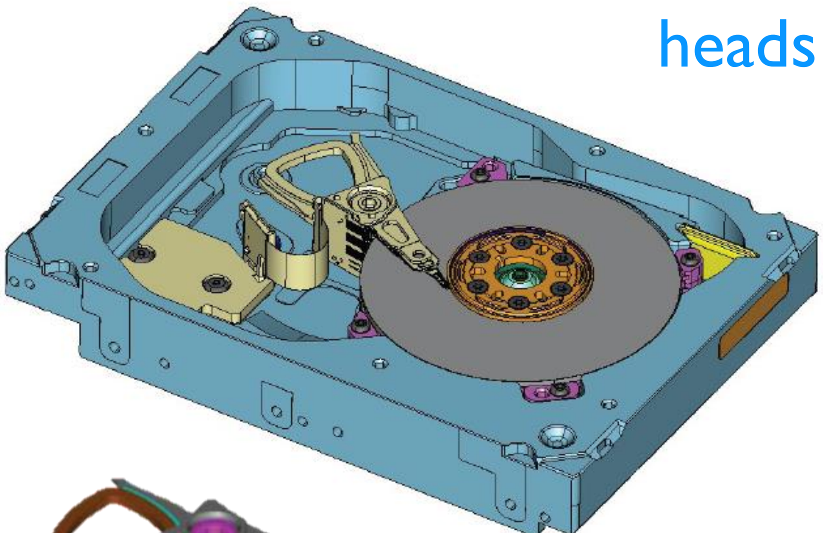


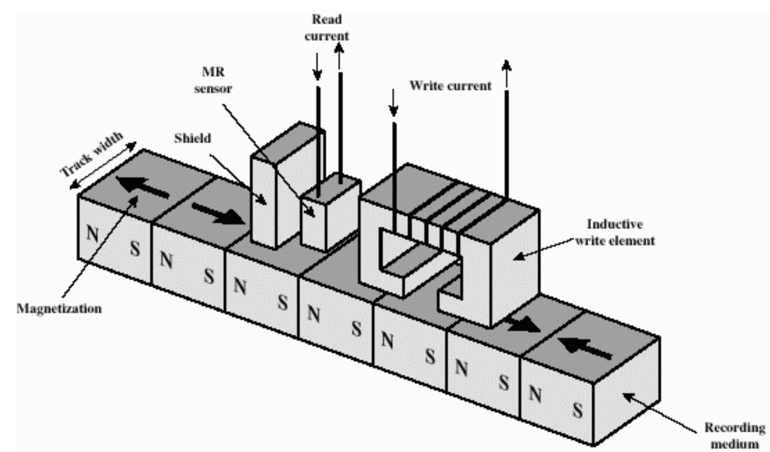
Disks
(plates)



Anatomy of a hard disk drive

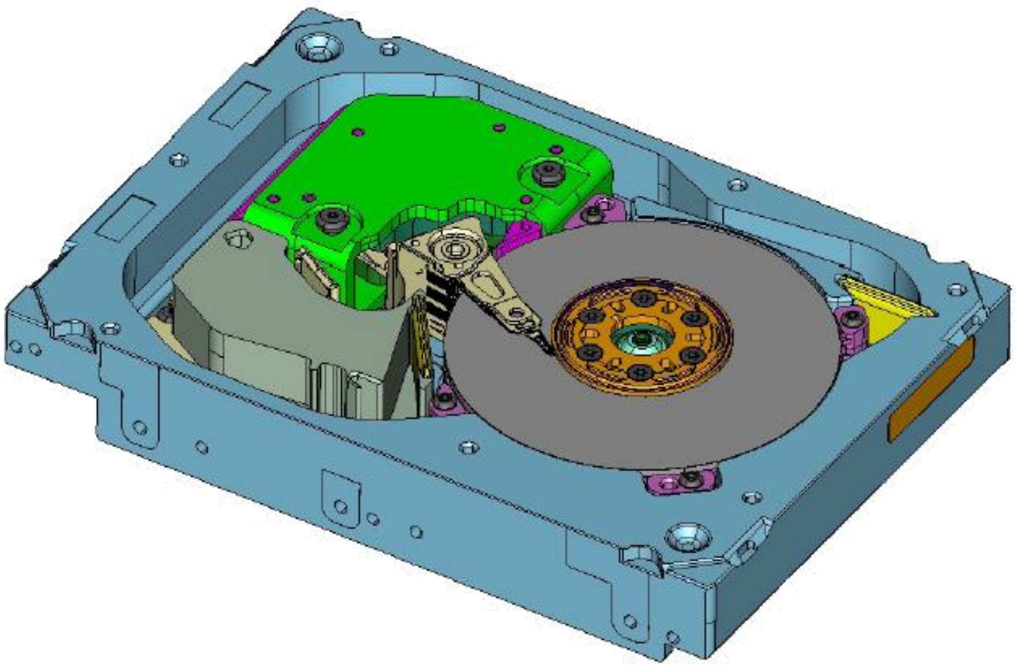
Read/write heads





https://faculty.etsu.edu/TARNOFF/ntes2150/mem_hier/mem_hier.html

Anatomy of a hard disk drive

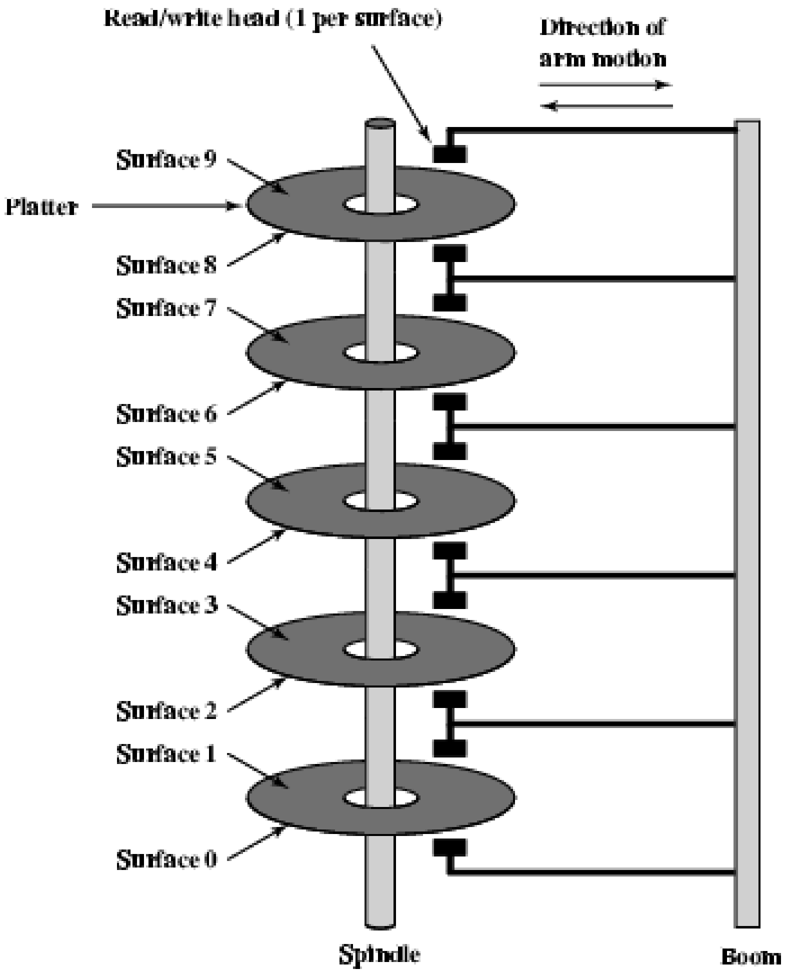


control and
mechanic module

Anatomy of a hard disk drive

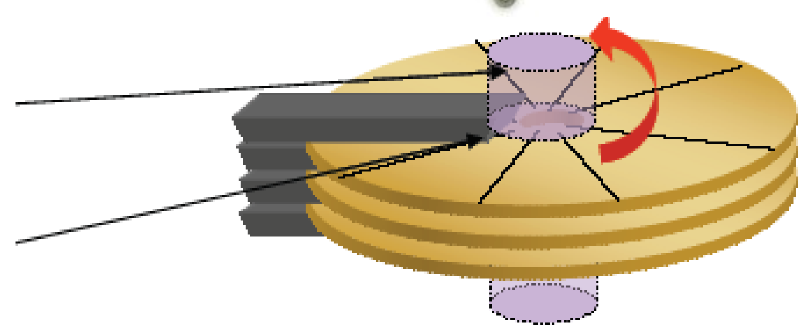
- Disk controller
  - Command scheduling
  - Error correction
  - Optimization
  - Integrity check
  - Revolutions per minute (RPM) monitoring
  - Disk cache

Multiple plates

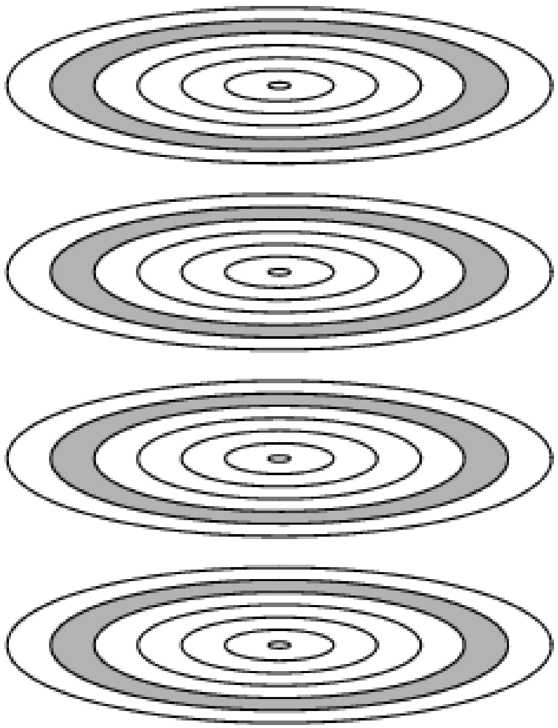


http://www.snia.org

Rotation

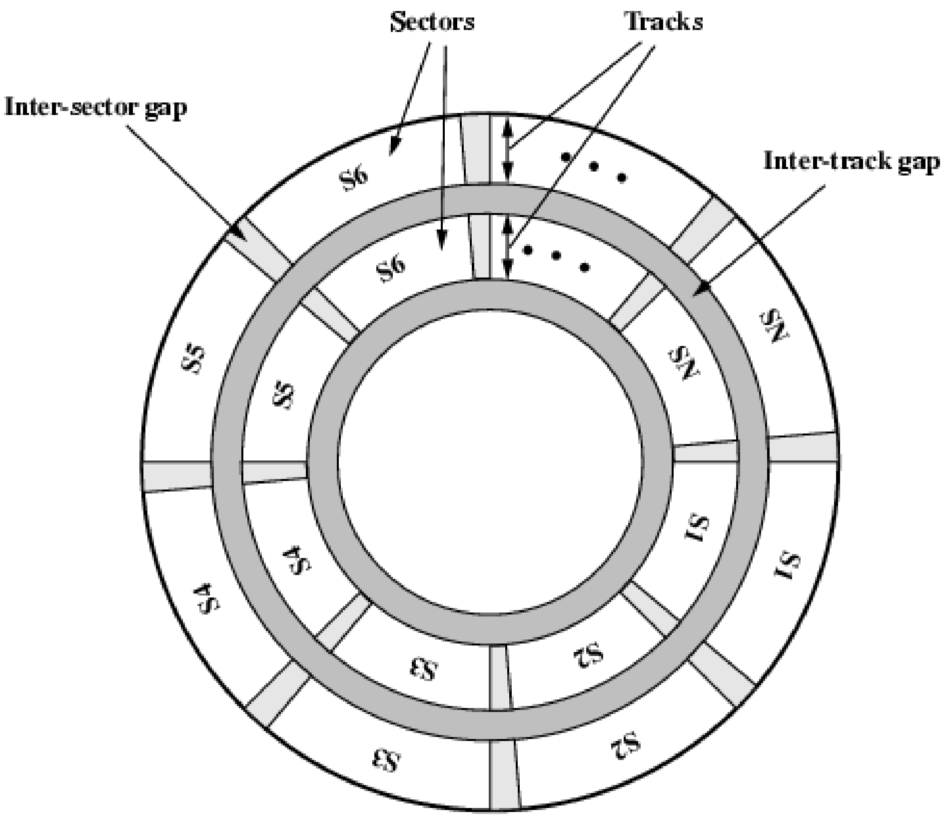


Cylinders



- Cylinder:
  information accessed by all heads in a rotation

Tracks, sectors and blocks



Track:

- Concentric ring in plate

Sector:

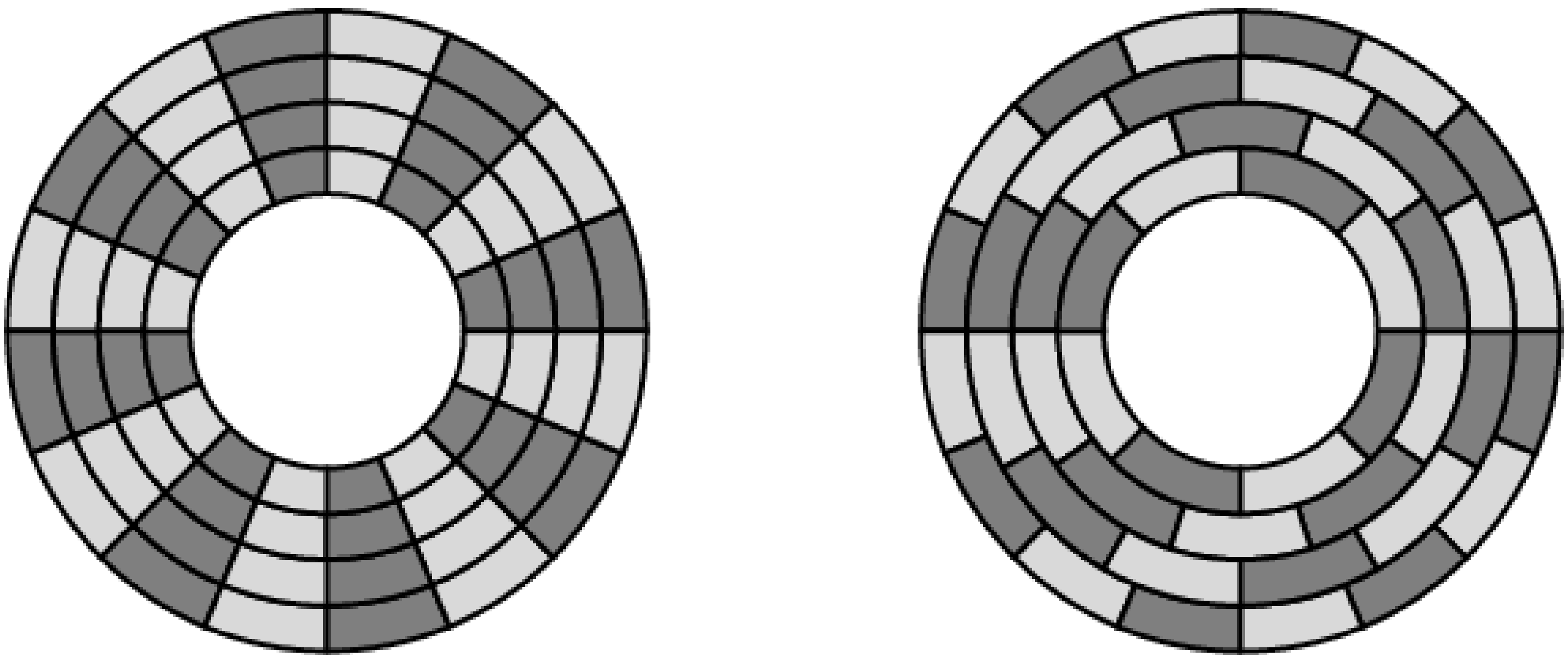
- Disk area division performed on formatting (typically 512 bytes)

Block:

- File System writes in blocks
- Sector groups

Source: Stallings, William, Computer Organization & Architecture, 6e, Prentice Hall, Upper Saddle River, NJ, Figure 6.2, p. 166.

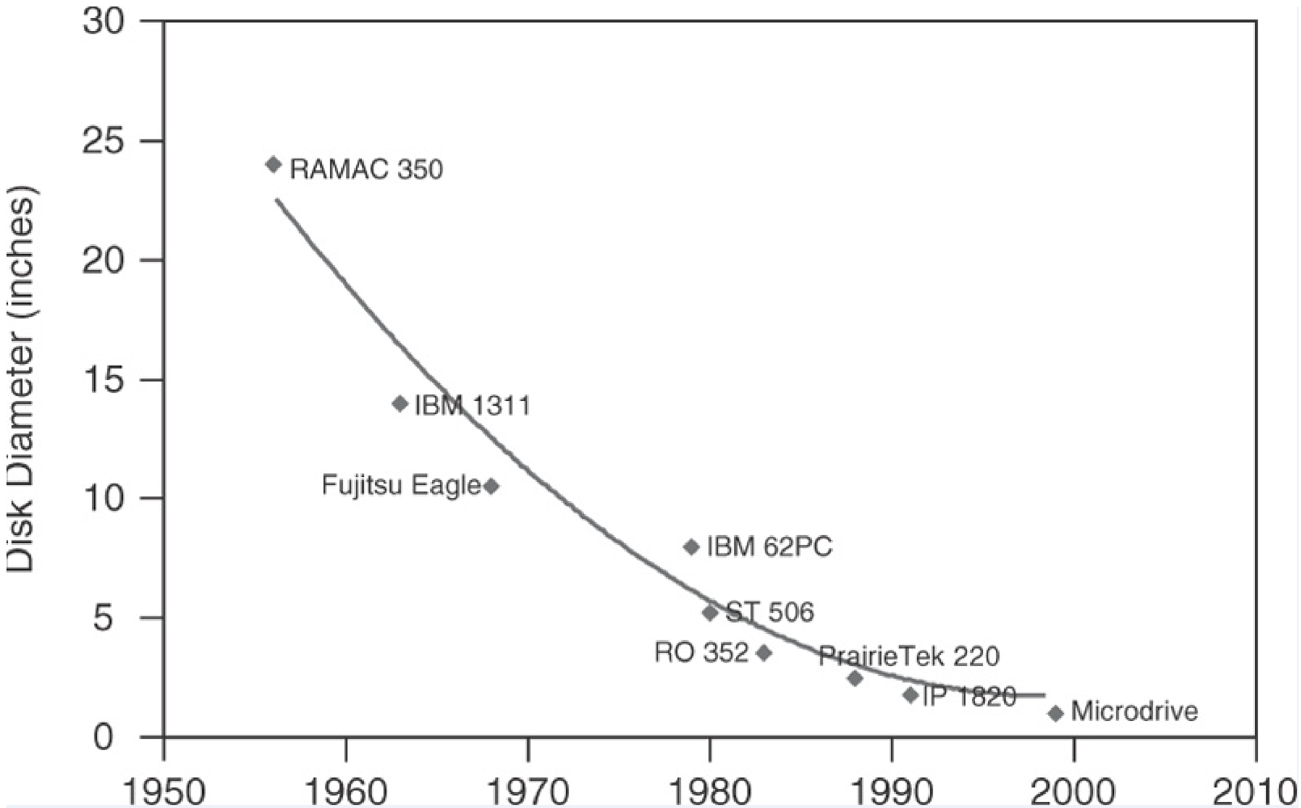
Distribution of sectors



(a) Constant angular velocity (b) Multiple zoned recording

Stallings, William, Computer Organization & Architecture, 6e, Prentice Hall, Upper Saddle River, NJ, Figure 6.3, p. 167.

Evolution of disk sizes



Memory Systems
Cache, DRAM, Disk
Bruce Jacob, Spencer Ng, David Wang
Elsevier

Capacity

- Bits per inch
  - They depend on the read/write head, the recording medium, the rotation of the disk and the speed at which the bus can accept data.
- Tracks per inch
  - They depend on the read/write head, the recording medium, the precision with which the head can be positioned and the ability of the disk to rotate in a perfect circle.

Storage capacity

- For constant angular velocity disks:
  - n_s: number of surfaces
  - p: tracks per surfaces
  - s: sectors per track
  - t_s: bytes per sector

$$Capacity = n_s \times p \times s \times t_s$$

- For mulitple zone recording:
  - z: number of zones
  - p_i: number of track per zone i
  - p_i: sectors per track in zone i

$$Capacity = n_s \times t_s \times \sum_{i=1}^{z} (p_i \times s_i)$$

Exercise

- How many bytes does a disk drive of 250 GB store?

Remainder

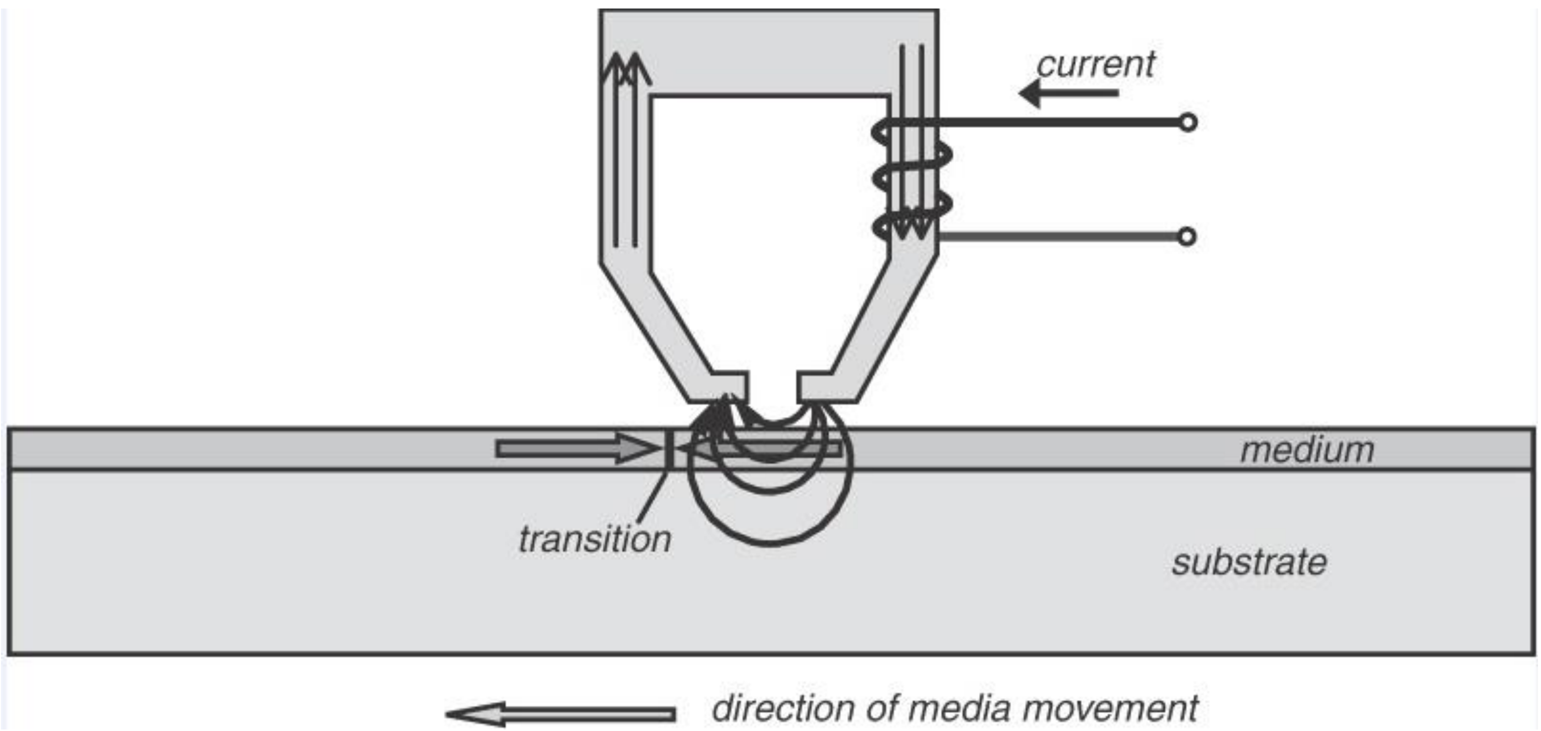
- 1 KB = 1024 bytes, but in the I.S. is 1000 bytes
- Manufactures of disk drives and telecommunications use I.S.:
  - A disk drive of 30 GB stores 30 x 109 bytes
  - A network of 1 Mbit/s transfers 106 bps.

Name	Abr	Factor	I.S.
Kilo	K	2^{10} = 1,024	10^{3} = 1,000
Mega	M	2^{20} = 1,048,576	10^{6} = 1,000,000
Giga	G	2^{30} = 1,073,741,824	10^{9} = 1,000,000,000
Tera	T	2^{40} = 1,099,511,627,776	10^{12} = 1,000,000,000,000
Peta	P	2^{50} = 1,125,899,906,842,624	10^{15} = 1,000,000,000,000,000
Exa	E	2^{60} = 1,152,921,504,606,846,976	10^{18} = 1,000,000,000,000,000,000
Zetta	Z	2^{70} = 1,180,591,620,717,411,303,424	10^{21} = 1,000,000,000,000,000,000,000
Yotta	Y	2^{80} = 1,208,925,819,614,629,174,706,176	10^{24} = 1,000,000,000,000,000,000,000,000

Recording techniques

- Over the last decade the magnetic recording has achieved 100% growth of Areal Density (AD)
- Each bit cell in a track is composed of multiple magnetic grains
- The size or the number of magnetic grains in a bit cell cannot be scaled much below a diameter of ten nanometers due to:
  - Superparamagnetic effect
  - Ambient temperature would become magnetic grains unstable
- Recording techniques:
  - Longitudinal recording: store data in a longitudinal way over a horizontal plane
  - Perpendicular recording: data are stored in vertical way, increasing the disk capacity

Read/write head



Memory Systems
Cache, DRAM, Disk
Bruce Jacob, Spencer Ng, David Wang
Elsevier

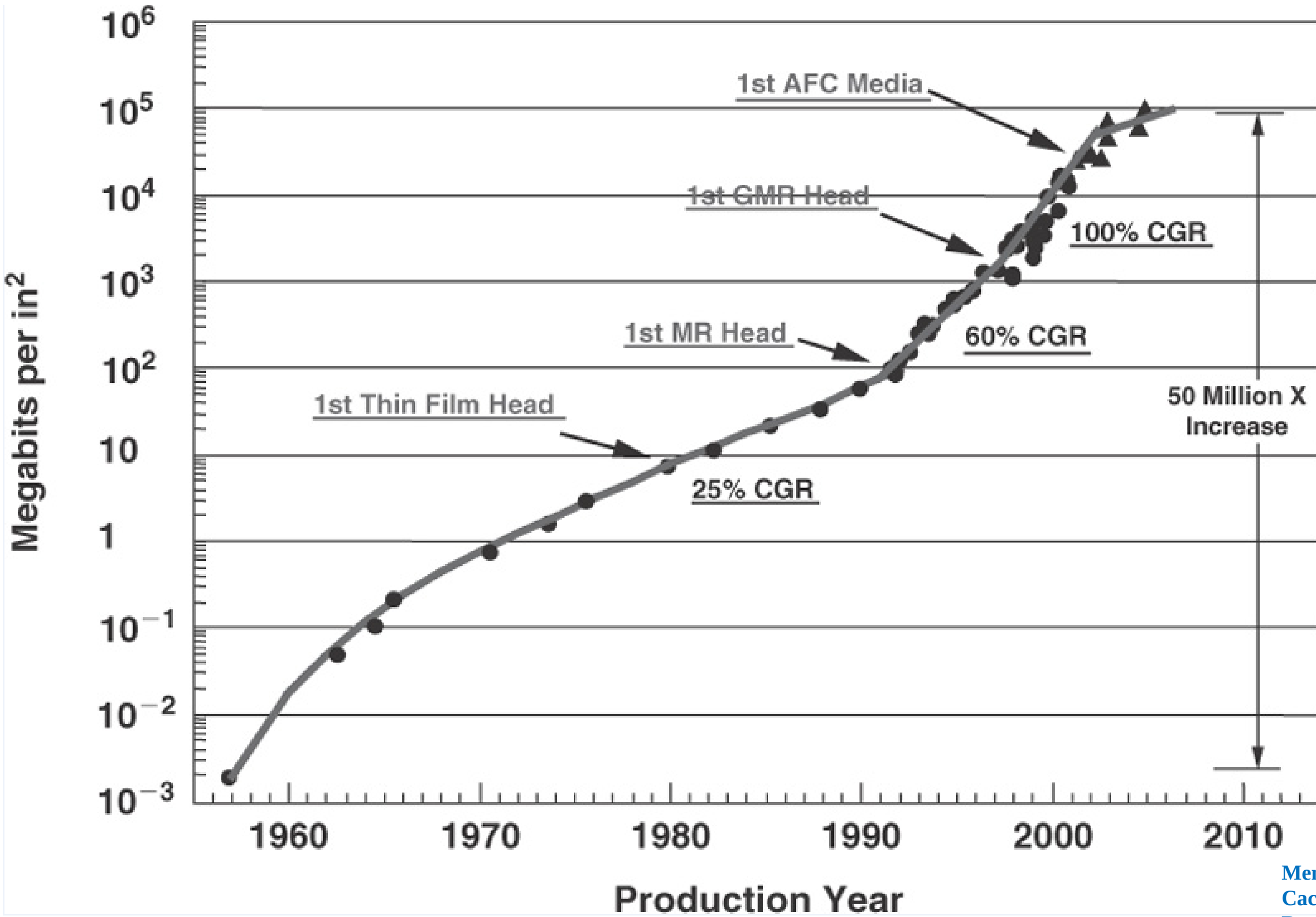
Areal density

- Improvements in disk capacity are expressed as an improvement in areal density (number of bits that can be recorded per square inch):

$$Areal\,density\,(AD) = \frac{Tracks}{Inch}\,on\,a\,disk\,surface \times \frac{Bits}{Inch}\,on\,a\,track$$

- Until 1998 the annual increase rate was 29%
- 1998-1997 the annual increase rate was 60%
- 1997-2003 the annual increase rate was 100%
- 2003-2011 the annual increase rate was 30%
- In 2011 the bigger areal density in commercial products was 400 billions of bits per square inch
- The cost per bit has improved in a factor of 1.000.000 between 1983 and 2011

Evolution of areal density



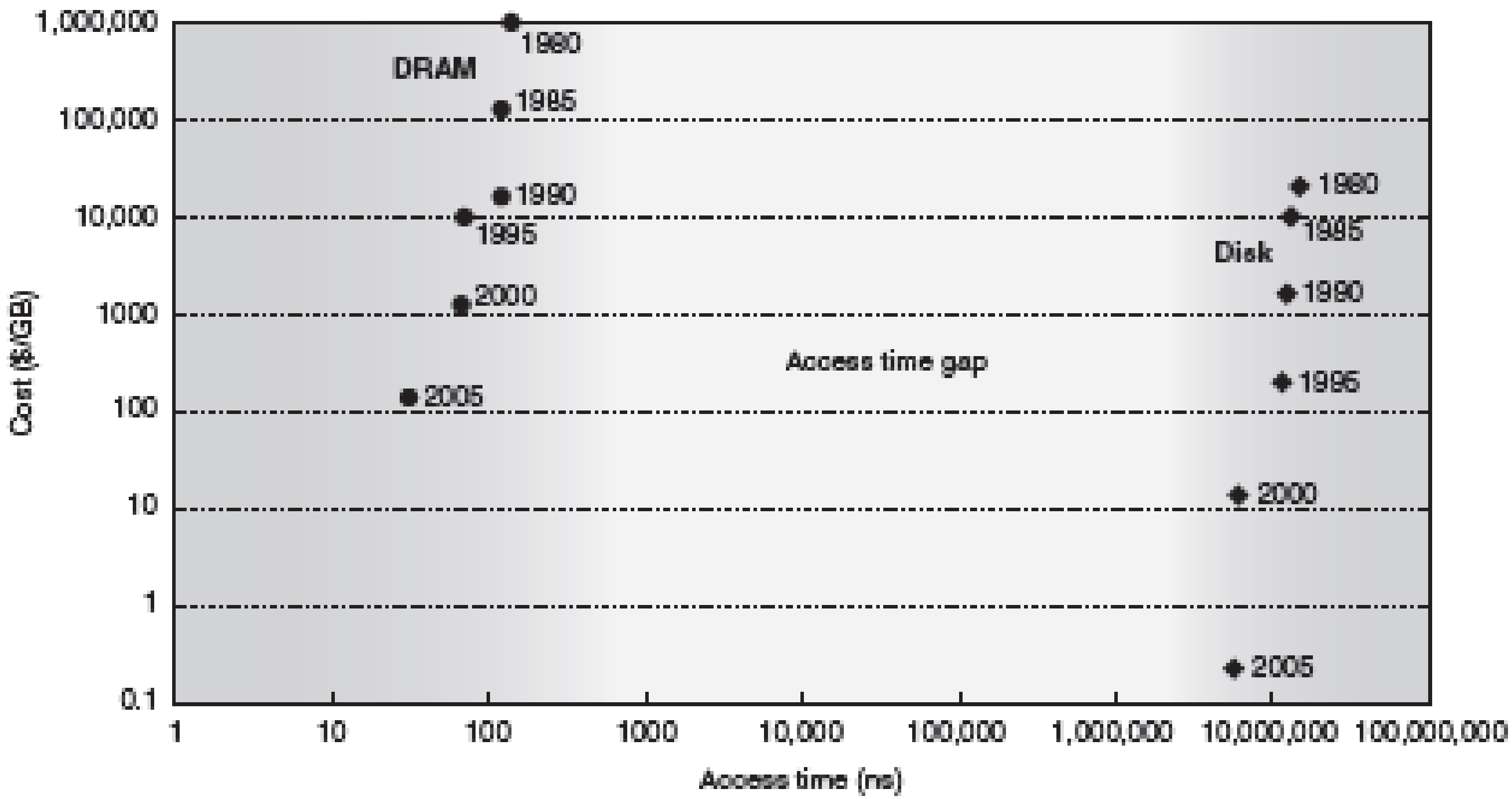
Memory Systems
Cache, DRAM, Disk
Bruce Jacob, Spencer Ng, David Wang
Elsevier

Disks and main memory

- The latency of a DRAM memory is 100.000 less than the latency of a disk
- The cost per GB in a DRAM memory is 30-150 times the cost per GB of a disk

- In 2015:
  - An 8 TB disk transfers 190 MB/s at a cost of 250 $.
  - An 8 GB DDR4 module transfers 25 GB/s and costs approx. 70$.

Disks and main memory

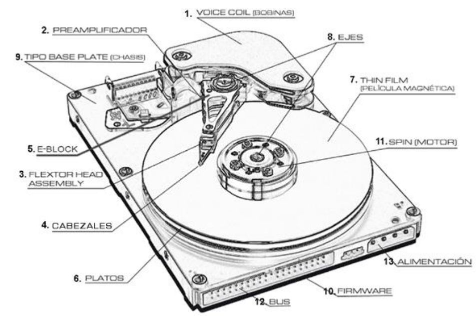


Patterson & Hennesy

Addressing

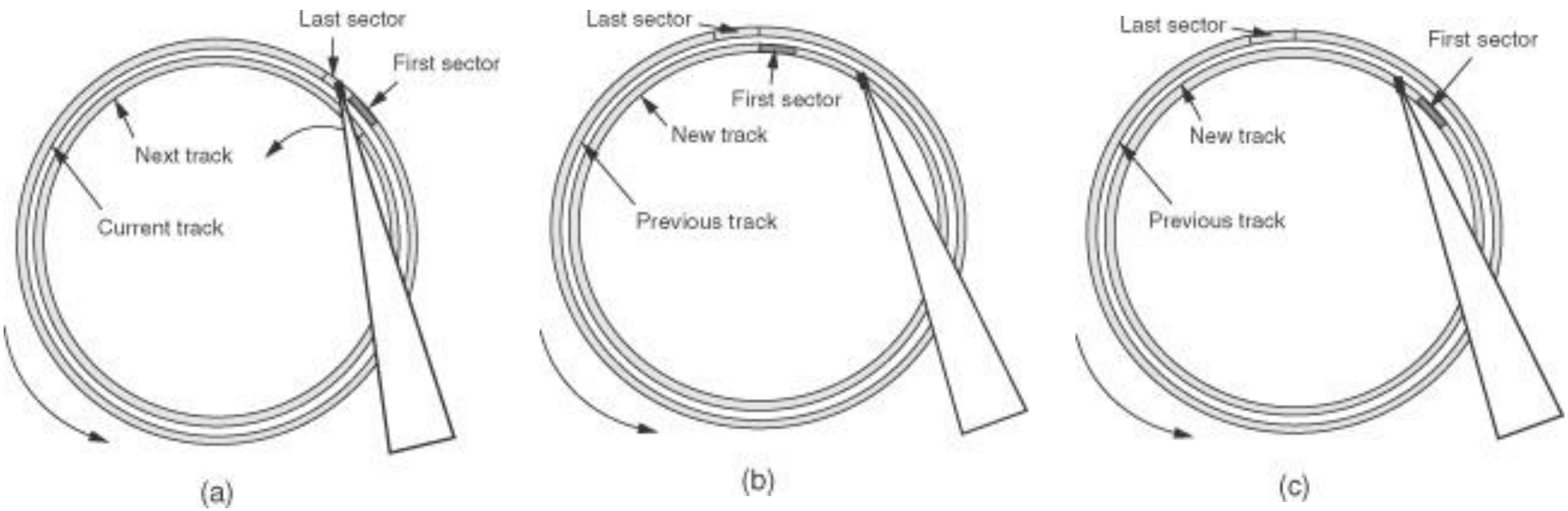
- Types of addressing:
  - Physical addressing: cylinder-track-sector. A sector is determined by these three values.
  - Logical blocks addressing (LBN)
    - Each sector has a logical number and the mapping is done by the disk
- Current disk controllers do the mapping between LBN and physical addresses

Access time



- $T_{access} = T_{seek} + T_{latency} + T_{transfer}$
- **Seek time** (T_{seek}): time to move the head from the current cylinder to the target cylinder
- **Rotational latency** ($T_{latency}$): time waiting for the rotation of the disk to bring the required sector under the read-write head
  - $T_{latency}$ = Half turn/lap time of a track
- **Data transfer time** ($T_{transfer}$): time required to traverse a sector and transfer the data from it.
  - $T_{transfer}$ = Amount of data / data transfer rate

Seek and rotation



Memory Systems
Cache, DRAM, Disk
Bruce Jacob, Spencer Ng, David Wang
Elsevier

Seek time, rotational latency and…

- Seek time
  - When the areal density increase the capacity of cylinders increase too:
    - The probability of reducing the number of seeks is increased
    - Increase the probability that the next data to request are in the same cylinder, reducing in this way the number of seeks
- Rotational latency
  - Rotational latency is generally calculated as half the time it takes the disk to do one revolution:

$$T_{rotate} = \frac{1}{2} \times \frac{60 \times 10^3}{RPM}$$

  - Zero-latency access
    - Transfer the data as soon as the head is on the desired track to a buffer where the blocks are then reordered.

… and data transfer time

- Data transfer time can be calculated as:

$$T_{transfer} = \frac{N_{request}}{N_{track}} \times \frac{60}{RPM}$$

- Where N_{track} denotes the number of sectors on a track, and $N_{request}$ the data length of a request measured in sectors.

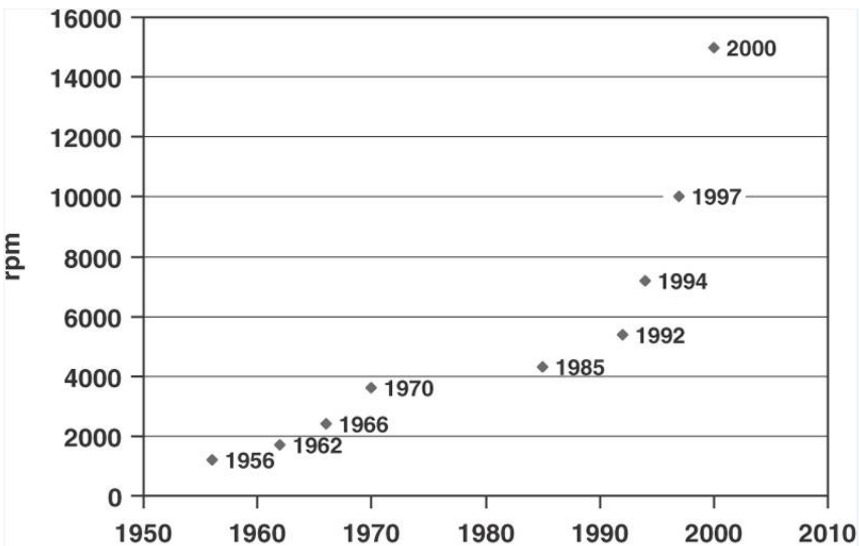
- The ratio of the sectors of the outmost zone to that of the innermost zone ranges from 1.43 to 1.58.
- Two elements:
  - External data rate to measure the transfer rate between memory and disk cache
  - Transfer rate between disk cache and disk storage media

Rotational latency and rotational speed

Rotational Speed (RPM)	Rotational Latency (ms)
5400	5.6
7200	4.2
10000	3.0
12000	2.5
15000	2.0

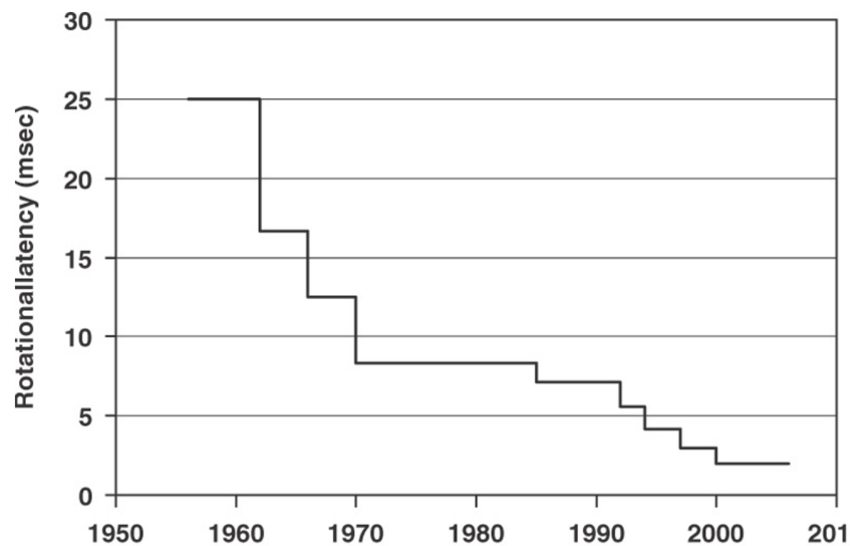
Evolution rotational latency and RPM

RPM



Memory Systems
Cache, DRAM, Disk
Bruce Jacob, Spencer Ng, David Wang
Elsevier

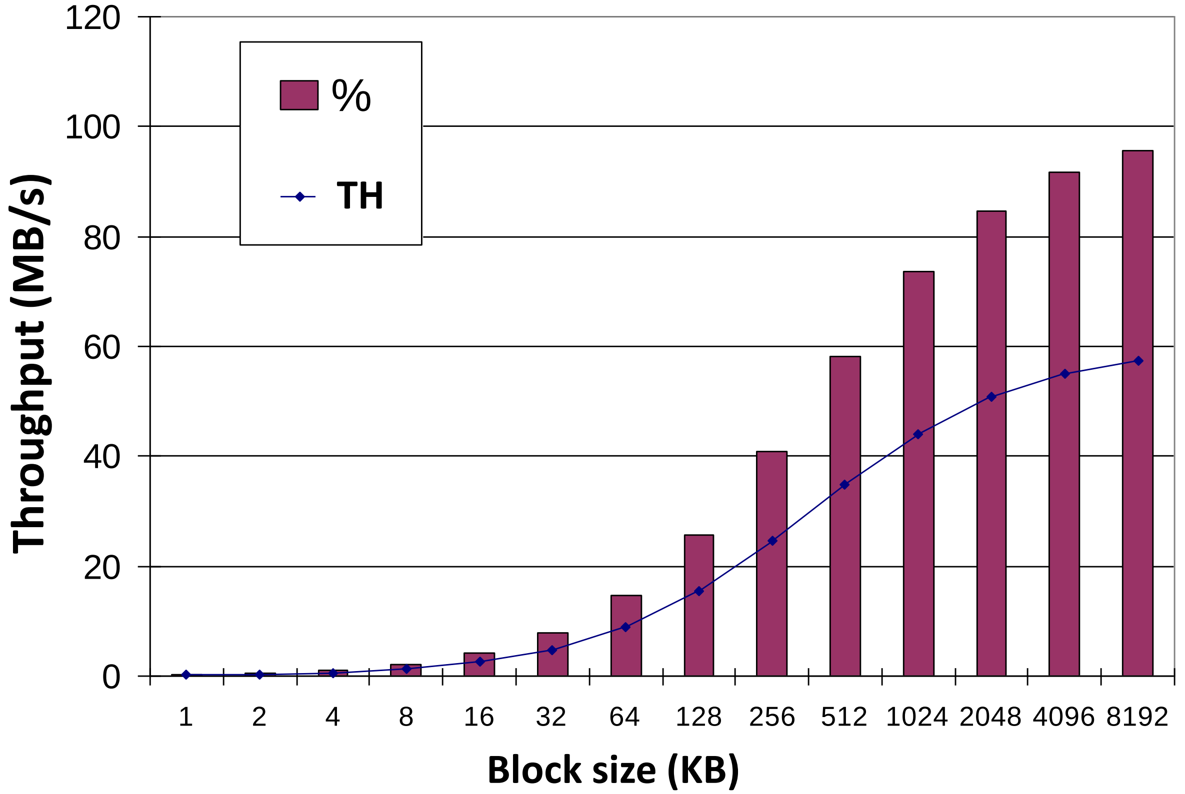
Rotational latency



Memory Systems
Cache, DRAM, Disk
Bruce Jacob, Spencer Ng, David Wang
Elsevier

Effect of the request size

- Effect of the request size (ta=6 ms y TH = 60MB/s)



Exercise

- Consider a disk with:
  - Rotational speed: 7200 rpm
  - Disk platters: 5, with 2 surfaces per plate
  - Number of tracks per plate: 30000
  - Sectors per plate: 600
  - Seek time: 1 ms per each 100 tracks
- If the disk head is in track 0 and the data requested are stored in track 600. Compute:
  - Capacity of the disk
  - Rotational latency
  - Transfer time needed to transfer a sector
  - Access time for a sector in track 600 (seek time)

Exercise (solution)

- Capacity:
  - 5 plates * 2 sides/plates * 30.000 tracks/side * 600 sector/track * 512 bytes/sector = 85,8 GB
- Rotational latency:
  - Lr = Half turn/lap time of a track
  - 7.200 rotation/minute -> 120 rotation/second -> 0,0083 seconds/rotation -> 4.2 miliseconds (half rotation)
- Sector transfer time:
  - 600 sectors per track and 1 track is read in 8,3 miliseconds
  - 8,3 / 600 -> 0.014 miliseconds
- Seek time:
  - Every 100 tracks 1 ms, and it has to seek to the track 600
  - 600 / 100 = 6 miliseconds

Disk controller

- Circuits and components to control the disk:
  - Storage interface
  - Disk sequencer
  - Error correction code(ECC)
  - Servo motor
  - Microprocessor
  - Buffer controller
  - Disk cache

Disk cache

- Exploit the locality
  - Usually does not exhibit temporal locality due to operating system data caches.
  - Typically implements read-ahead (prefetch) to improve spatial locality
- Reduce physical access to disk
  - Reduce the heat dissipation
  - Increase the performance

Disk cache

- Systems designers generally believe that the size of a cache should be at least 0.1 to 0.3 % of the disk.
- Disk caches are typically divided into independent segments corresponding to sequential data streams
- Replacement algorithms
  - Random Replacement
  - Least Frequently Used (LFU)
  - Least Recently Used (LRU)

Disk scheduler

- Disk drives maintain a queue with pending requests
  - Disk schedulers are designed to minimize the access time by reordering or rearranging pending request in the queue to reduce the seek time and rotational latency
  - E.g.: block requested: 1, 9, 2, 10 => in queue: 1, 2, 9, 10
- Scheduling algorithms:
  - First Come First Served (FCFS)
  - Shortest Seek Time First (SSTF)
  - SCAN
  - C-SCAN
  - LOOK

Other elements

- Disk sequencer: manages the data transfer between storage interface and data buffer
- ECC: responsible for adding ECC codes to the user data and also checking and correcting errors
- Servo control: detects the current position of the disk head and controls track following and seeking
- Microprocessor: controls the general disk behavior
- Buffer controller: provides arbitration and raw signal control of the buffer memory

Exercise

- A disk drive has a rotational speed of 7200 rpm and a constant areal density of 604 sectors per track. The average access time in 4ms
  - Compute the access time to a sector

Exercise

- Be a hard disk with an average seek time of 4 ms, a rotation speed of 15 000 rpm and 512-byte sectors with 500 sectors per track. We need to read a file consisting of 2 500 sectors with a total of 1.22 MB.
  Estimate the time necessary to read this file in two scenarios:
  - The file is stored sequentially, i.e. the file occupies the sectors of 5 adjacent tracks.
  - The sectors of the file are randomly distributed on the disk.

Reliability

- MTTF: mean time to failure
- MTTR: mean time to repair
- Availability is defined as:

$$availability = \frac{MTTF}{MTTF + MTTR}$$

- What does a reliability of 99% mean?
  - In 365 consecutive days device works 99*365/100 = 361.3 days
  - It is out of service 3.65 days
- Failures in disk drives produce the 20-55% of the failures in the storage systems.

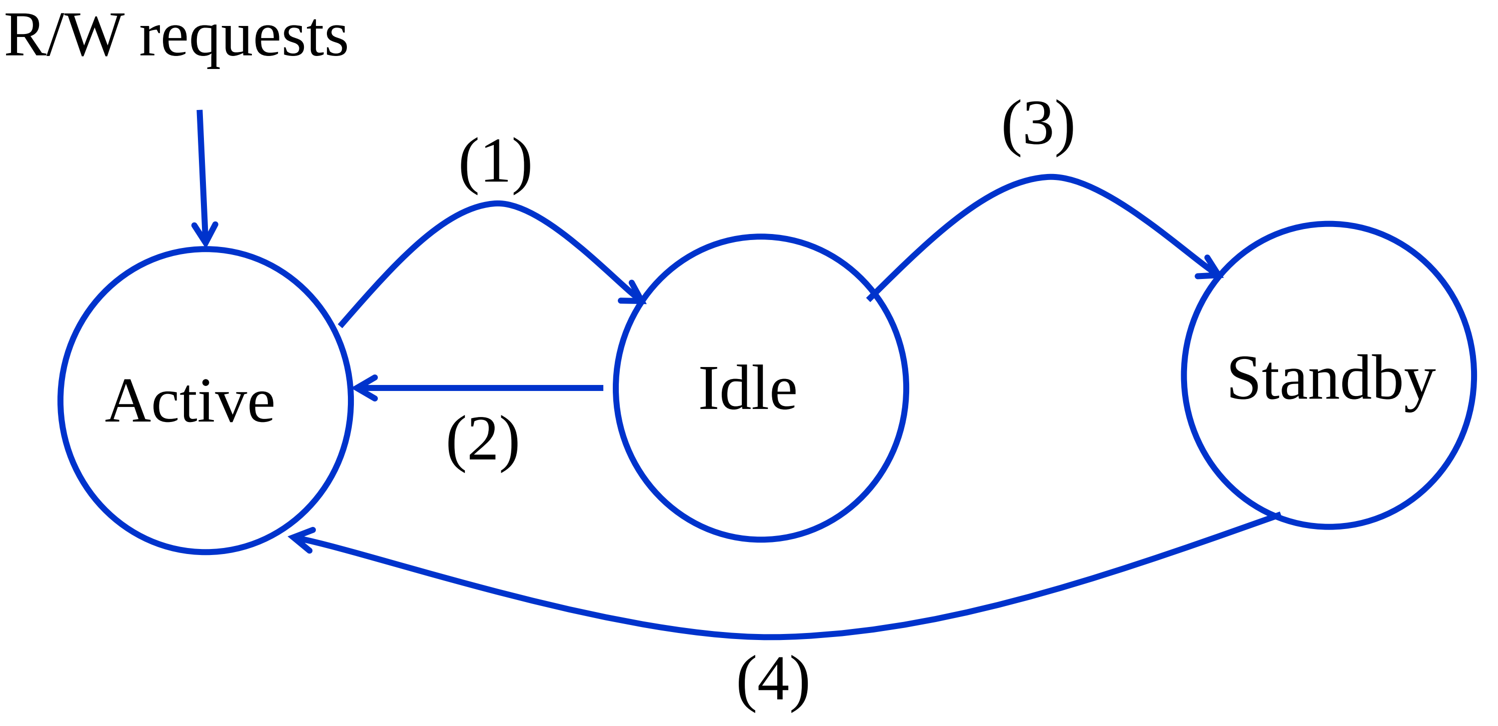
Energy consumption

- The energy consumption in a typical ATA disk drive of 2011 is:
  - 9 w when is idle
  - 11 w when is reading or writing
  - 13 w in a seek operation
- Power consumed by a disk:

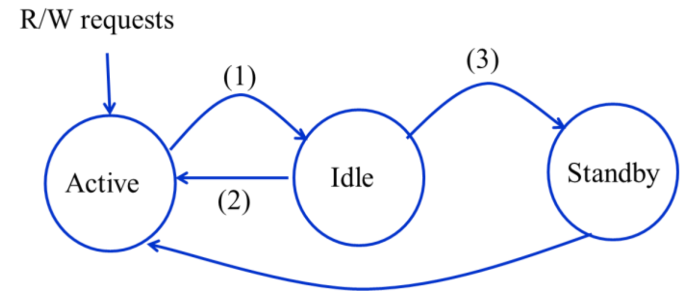
$$Power = N_{platter} \times D_{platter}^{4.6} \times RPM^{2.8}$$

- Where $N_{platter}$ is the number of disk patters y $D_{platter}$ the diameter for the platters
- Temperature is often the most important factor which affects the reliability of disk drives
  - Every 10° increase over 21° decreases the reliability by 50%

Power state transition of disk drives



Power state transitions



1. There is no pending request.
   - the disk drive is transferred to the idle state (where)
   - the disk platters are still spinning but the electronics may be partially unpowered
2. Disk drive receives a request.
3. To conserve energy
   - the disk drive is transferred to the standby state (where)
   - the disk stops spinning, and the head is moved off the disk
4. To perform requests after entering the standby state, the disk drive must be transferred back from the standby state to the active state by spinning up

Energy conservation methods

- Based on timeout strategies. Once a disk drive is idle for a specific period of time, the disk drive is spun down to save energy
- Dynamic prediction. Based on the behaviors of application
- Stochastic mechanisms.
- Application-aware power management
  - Applications inform over the access pattern (in the source code or with complier-driven methods)

Impacts of power state transitions
problems spinning down disks

- Increased consumption,
  when the platens have to rotate again.
- Reduces the reliability of the discs.
  Manufacturers usually indicate the number of start/stop cycles a disk can withstand. Above this value the probability of failure increases by 50%.
- Power saving methods are usually applied to portable devices and are not applied to servers because of the intensive data loads.

Contents

1. Introduction
2. Buses
   - Structure and operation
   - Bus hierarchy
3. Peripheral
   - Concept and types of peripherals
   - General structure of a peripheral
   - I/O modules
4. Case study: hard disk drive and solid-state drives
5. I/O interaction: I/O techniques

Solid State Drive (SSD)

- Semiconductor-based block storage device that acts as a disk drive
  - Based on Flash memories
    - Non-volatile storage
  - Based on DDR memories
    - Requires batteries and disk backup for non-volatile storage

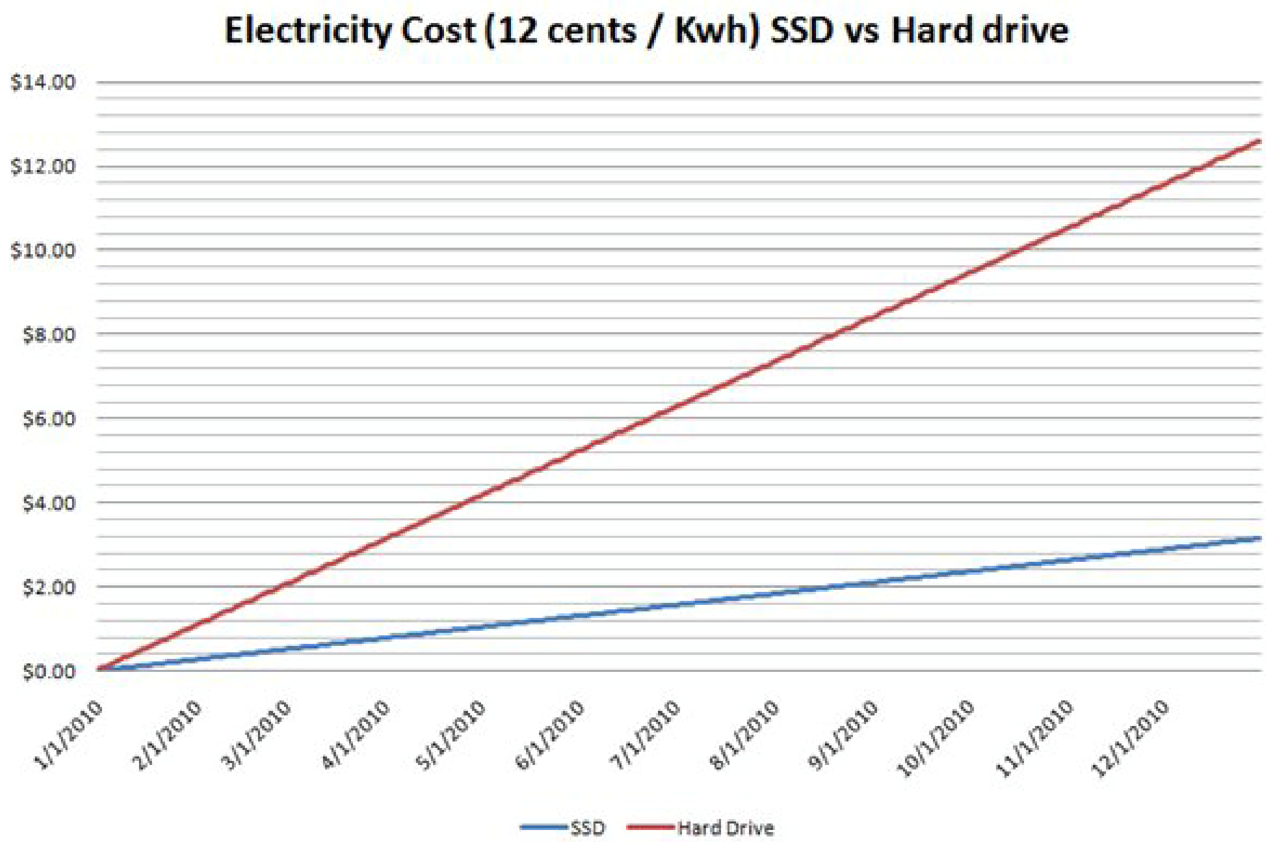
HDD
with moving parts

SDD
Without moving parts

SDD vs HDD

	SDD	HDD
Access time	0.1 ms	5-8 ms
I/O operations/sec	6000 io/s	400 io/s
consumption	2-5 watts	6-15 watts

SDD vs HDD: energy consumption



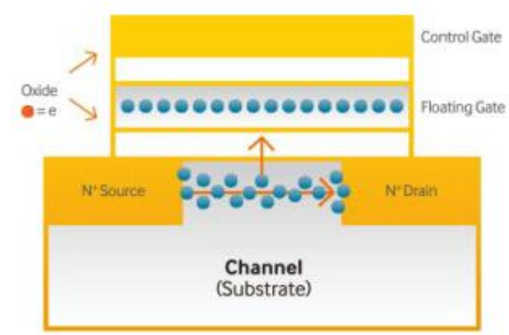
Flash Memories

- Non-volatile memories that can be deleted and recorded electrically
- Types:

Flash NOR	NAND Flash
Based on NOR gates	Based on NAND gates
Allows byte level access	Cannot access individual bytes
Good for high-speed random access	Reads and writes at high speed in sequential mode at block level
Used in BIOS memory (boot function)	Higher density and cheaper. Used in SSD
Faster reading operations	More durable, less expensive, denser, faster write/erase operations

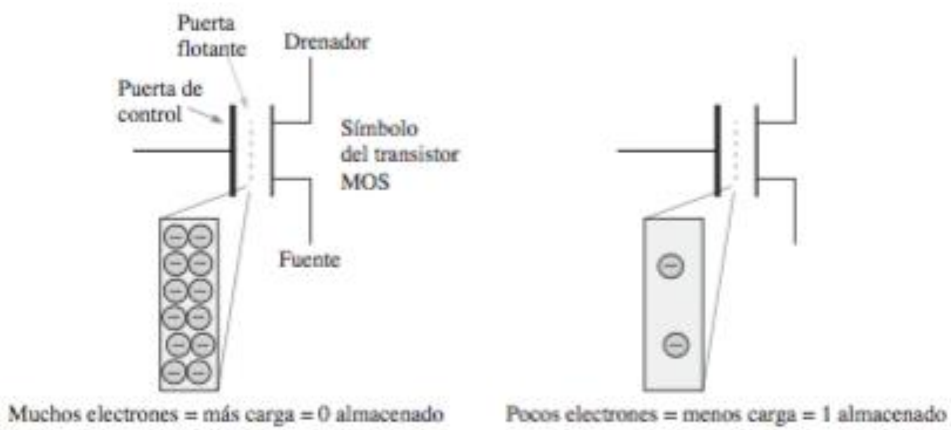
Memory cells

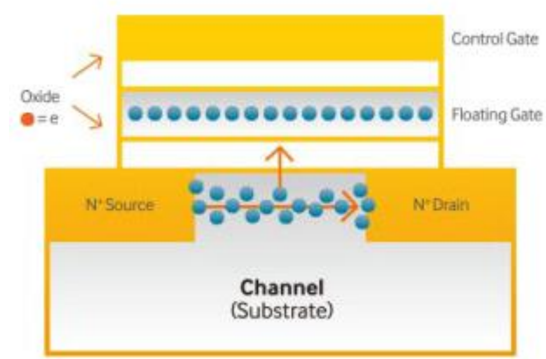
- Each storage cell consists of a floating gate MOS transistor
- There are two gates insulated by a layer of rust
  - Control gate
  - Floating gate
- The electrons flow freely between the two gates
- The floating gate is electrically insulated, trapping the electrons



Memory cells

- The data bit is stored as a charge or no charge in the floating gate, depending on whether you want to store a 0 or a 1.



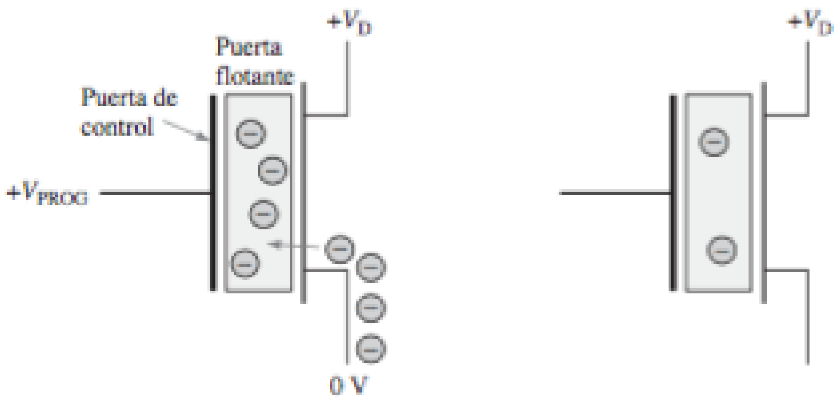


Fundamenros de Sistemas Digitales
Thomas L. Floyd

Basic operations of a flash memory cell

- Programming
  - Initially all cells are in state 1, because the charge is removed
- Read operation
- Delete operation

Programming a Flash memory cell



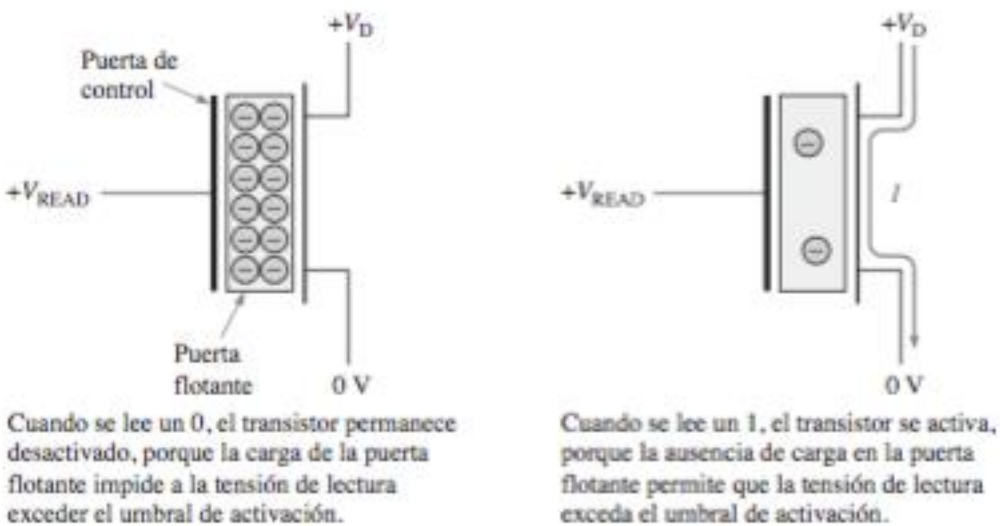
Fundamenros de Sistemas Digitales
Thomas L. Floyd

- To store a 0, a sufficiently positive voltage is applied to the control gate with respect to the source, to add charge to the floating gate during programming (attracts electrons)
- In the writing process, electrons are added to those gates that should store a 0 and not added to those that should store a 1.
- Only "0" is written

- To store a 1, no charge is added, leaving the cell in the deleted state

Reading a Flash memory cell

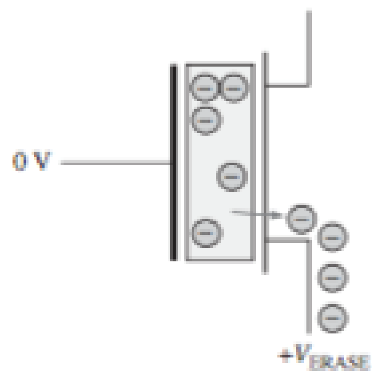
- Positive voltage is applied to the control gate



Fundamenros de Sistemas Digitales
Thomas L. Floyd

Delete

- During the delete operation, the charge is removed from all memory cells. A voltage is applied in the opposite direction to remove the electrons



Para borrar una célula, se aplica a la fuente una tensión suficientemente positiva con respecto a la puerta de control, con el fin de extraer la carga de la puerta flotante durante la operación de borrado.

Fundamenros de Sistemas Digitales
Thomas L. Floyd

NAND Flash memory types

- Single-level cell flash (SLC)
  - Store 1 bit per cell
  - 50000 - 100000 writings per cell
  - Used primarily in military and industrial applications
- Multi-level cell flash (MLC)
  - Store several bits per cell, depending on the number of electrons stored in the cell
  - They offer more capacity but less duration (they wear more)
  - < 10000 writings per cell
  - Used in consumer electronics
  - Lower cost
  - Half the performance of SLCs

Wear leveling

- Problem:
  - A NAND flash memory can only write a certain number of times in each block (or cell)
  - When the limit is exceeded, the cell wears out (its oxide layer) and no longer stores electrons properly
- Solution: Wear Leveling
  - A process used by an SSD controller to maximize the life of the flash memory
  - This technique levels out the wear and tear on all blocks by distributing the data writing across all blocks
    - When a block is to be modified, it is written in a new one

Structure of a Nand Flash

- NAND flashes are divided into 128 KB blocks that are subdivided into 2KB pages
- The deletion is done from a complete block
- Programming and reading a page

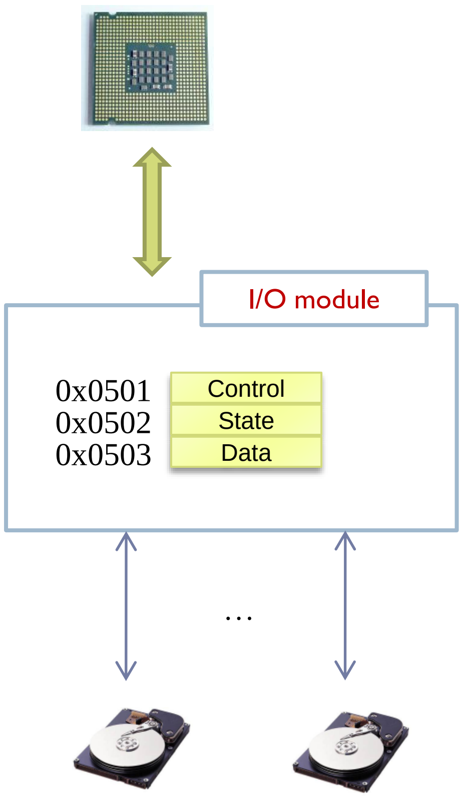
Contents

1. Introduction
2. Buses
   - Structure and operation
   - Bus hierarchy
3. Peripheral
   - Concept and types of peripherals
   - General structure of a peripheral
   - I/O modules
4. Case study: hard disk drive and solid-state drives
5. I/O interaction: I/O techniques

Félix García Carballeira, Alejandro Calderón Mateos

I/O module: main characteristics

- **Transfer unit**
  - Block
  - Character
- **Addressing**
  - Memory-mapped I/O
  - Port-mapped I/O
- **I/O techniques**
  - Programmed I/O
  - Interrupt I/O
  - DMA



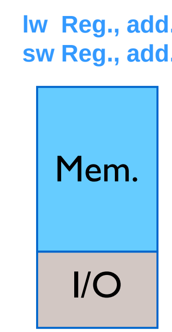
Characteristics (1/3):
Transfer unit

- **Block devices**:
  - Unit: block of bytes
  - Access: sequential or random
  - Operations: `read`, `write`, `seek`, ...
  - Examples: "tapes" and disks

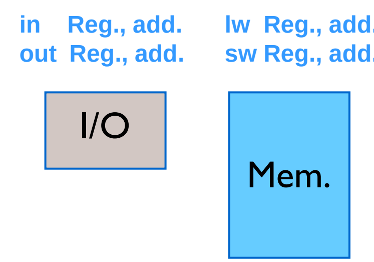
- **Character devices**:
  - Unit: chars (ASCII, Unicode, etc.)
  - Access: sequential to characters
  - Operations : `get`, `put`, ...
  - Example: terminals, printers, etc.

Characteristics (2/3):
I/O addressing

- **Memory-mapped I/O (MMIO)**
  - I/O registers are mapped in memory using a set of memory addresses for these registers.
  - Same machine instructions for memory and I/O:
    - lw $a0, label2_disk1
      - Load in the processor register "$a0" the value stored in the I/O register identified by a given address "label2_disk1"
    - sw $a0, label_disk2
      - To write an item in an I/O register from the I/O module



- **Port-mapped I/O (PMIO):**
  - I/O address space is isolated from memory address space.
  - Special privileged machine instructions (~ to lw/sw):
    - IN $a0, label2_disk1
      - Load in the processor register "$a0" the value stored in the I/O register identified by a given address "label2_disk1"
    - OUT $a0, label_disk2
      - To write an item in an I/O register from the I/O module



Characteristics (3/3)

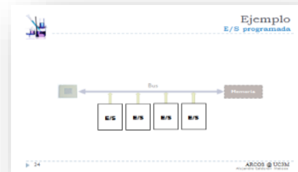
I/O techniques: interaction CPU - I/O module

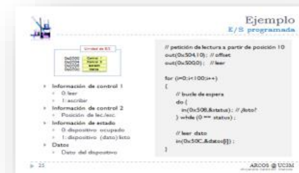
- **Programmed I/O**
  - CPU does all I/O: busy wait → transfer
  - *'polling'*
- **Interrupt I/O**
  - CPU does not wait, only transfer data
- **DMA (Direct Memory Access) I/O**
  - CPU neither wait, nor transfer, it is notified at the end

Characteristics (3/3)

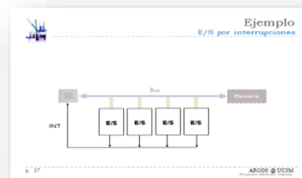
I/O techniques: interaction CPU - I/O module

- **Programmed I/O**

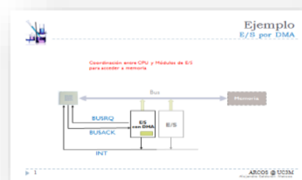


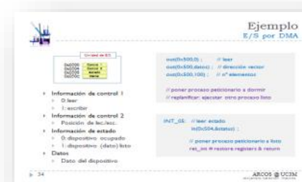


- **Interrupt I/O**

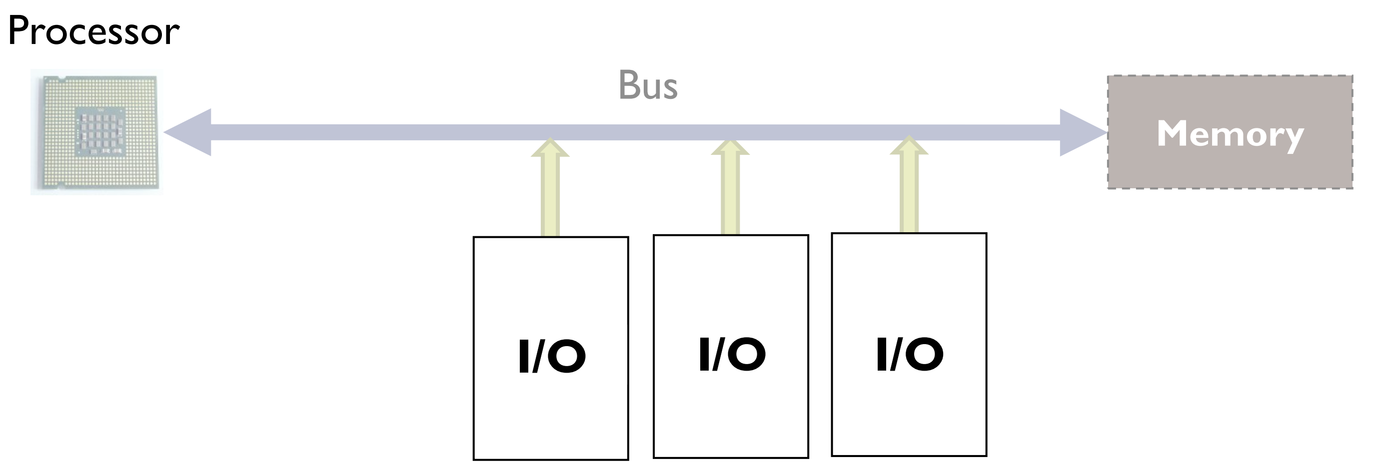


- **DMA I/O**

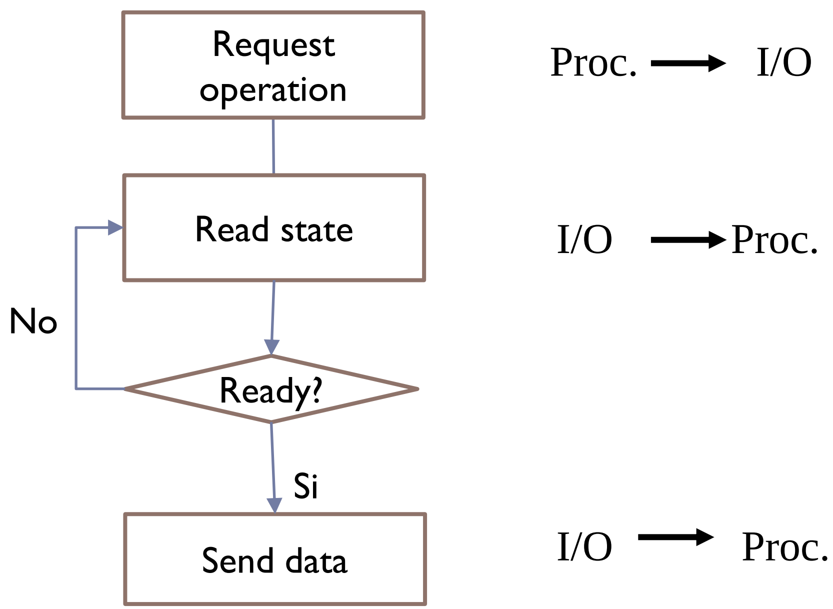




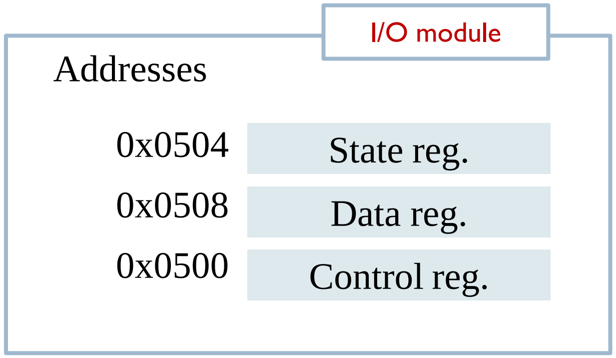
Programmed I/O



Interaction via programmed I/O



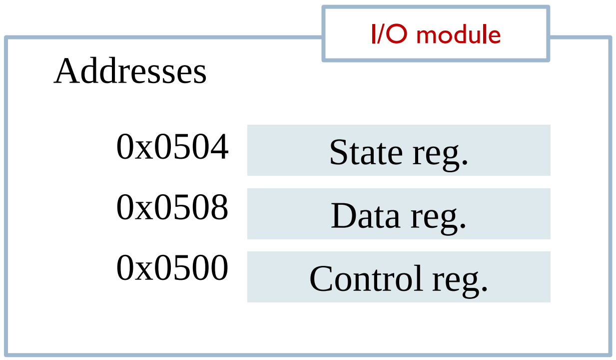
Example



- Control information:
  - 0: read
  - 1: write
- State information:
  - 0: device not ready
  - 1: device (data) ready
- Memory-mapped I/O:
  - lw and sw MIPS instructions

ARCOS @ UC3M
Félix García Carballeira, Alejandro Calderón Mateos

Example



- Control information:
  - 0: read
  - 1: write
- State information:
  - 0: device not ready
  - 1: device (data) ready
- Memory-mapped I/O:
  - lw and sw MIPS instructions

- How to read a data (word)?

1. Send the command
```
li $t0, 0
sw $t0, 0x0500
```
2. Read state
```
bucle: lw $t0, 0x0504
```
3. Check state
```
beqz $t0, bucle
```
4. Read the data (word)
```
lw $t0, 0x0508
```

Example

I/O module

Addresses

0x0504	State reg.
0x0508	Data reg.
0x0500	Control reg.

- Control information:
  - 0: read
  - 1: write
- State information:
  - 0: device not ready
  - 1: device (data) ready
- Memory-mapped I/O:
  - lw and sw MIPS instructions

- How to write a data (word)?

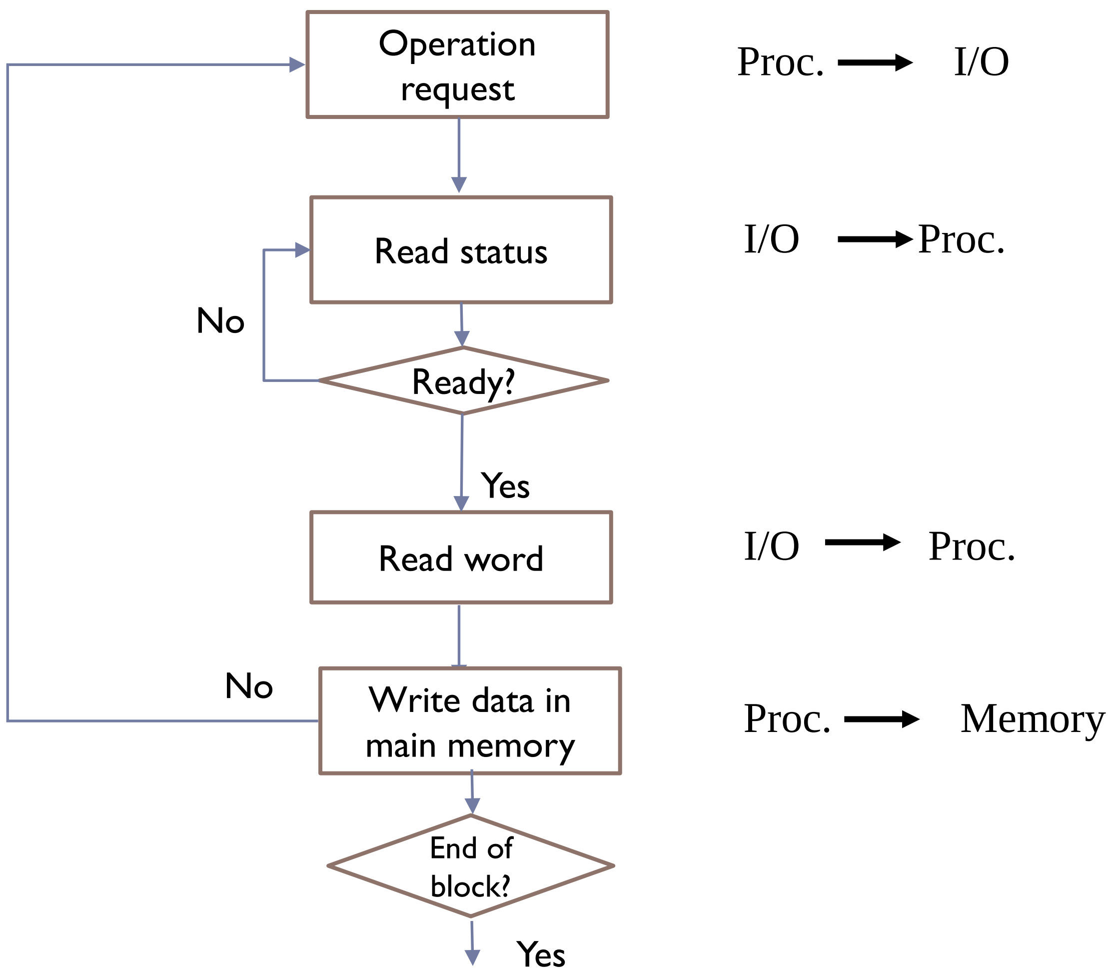
1. Send the data (word)
```
li $t0, 123
sw $t0, 0x0508
```

2. Send the command
```
li $t0, 1
sw $t0, 0x0500
```

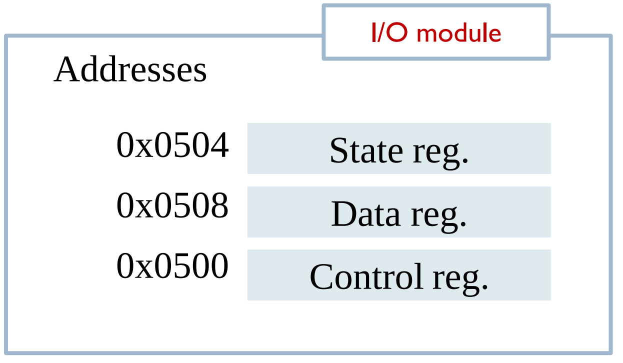
3. Read state
```
bucle: lw $t0, 0x0504
```

4. Check state
```
beqz $t0, bucle
```

Reading a data block



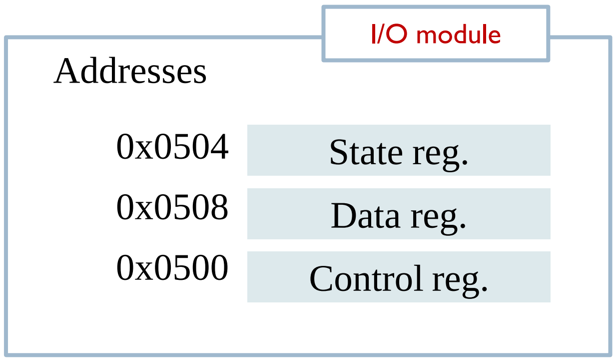
Exercise



- Control information:
  - 0: read
  - 1: write
- State information:
  - 0: device not ready
  - 1: device (data) ready
- Memory-mapped I/O:
  - lw and sw MIPS instructions

Code an assembler program that reads 100 integers using the described I/O module, and stores them in the main memory at address given by the 'data1' label.

Exercise (solution)



- Control information:
  - 0: read
  - 1: write
- State information:
  - 0: device not ready
  - 1: device (data) ready
- Memory-mapped I/O:
  - lw and sw MIPS instructions

```
.data
      data1:  .space  400
.text
.globl main
main:          li $t3 0
      loop1:   li $t0 0
               sw $t0 0x500
      loop2:   lw $t1 0x504
               beqz $t1 loop2
               lw $t2 0x508
               sw $t2 data1($t3)
               add $t3 $t3 4
               bne $t3 400 loop1
```

Exercise (solution)

I/O module

Addresses	
0x0504	State reg.
0x0508	Data reg.
0x0500	Control reg.

- Control information:
  - 0: read
  - 1: write
- State information:
  - 0: device not ready
  - 1: device (data) ready
- Memory-mapped I/O:
  - lw and sw MIPS instructions

```
.data
    data1:  .space  400
.text
.globl main
main:       li $t3 0
    loop1:  li $t0 0
            sw $t0 0x500
    loop2:  lw $t1 0x504
            beqz $t1 loop2
            lw $t2 0x508
            sw $t2 data1($t3)
            add $t3 $t3 4
            bne $t3 400 loop1
```

Synchronization loop

Exercise (solution)

I/O module

Addresses

0x0504	State reg.
0x0508	Data reg.
0x0500	Control reg.

- Control information:
  - 0: read
  - 1: write
- State information:
  - 0: device not ready
  - 1: device (data) ready
- Memory-mapped I/O:
  - lw and sw MIPS instructions

```
.data
    data1: .space  400
.text
.globl main
main:         li $t3 0
   loop1:    li $t0 0
             sw $t0 0x500
   loop2:    lw $t1 0x504
             beqz $t1 loop2
             lw $t2 0x508
             sw $t2 data1($t3)
             add $t3 $t3 4
             bne $t3 400 loop1
```

Synchronization loop

Transfer loop

Exercise

- Be a computer with the capacity to execute 200 million instructions per second (200 MIPS).
- The I/O module described above is connected with an average read timeout of 5 ms.
- Calculate how many instructions are executed in the synchronization loop and in the transfer loop for the program shown.

```
.data
    data1: .space  400
.text
.globl main
main:       li $t3 0
   loop1:   li $t0 0
            sw $t0 0x500
   loop2:   lw $t1 0x504
            beqz $t1 loop2
            lw $t2 0x508
            sw $t2 data1($t3)
            add $t3 $t3 4
            bne $t3 400 loop1
```

Synchronization loop

Transfer loop

Exercise (solution)

- Bucle de sincronización:
  - In average 5 ms
  - 200 MIPS are executed
  - $I_{bs} = 200*10^6 * 5*10^{-3} = 10^6$
- Bucle de transferencia:
  - 1 (li \$t3 0) $+ 6 * 100 + 10^6$ (I_{bs})

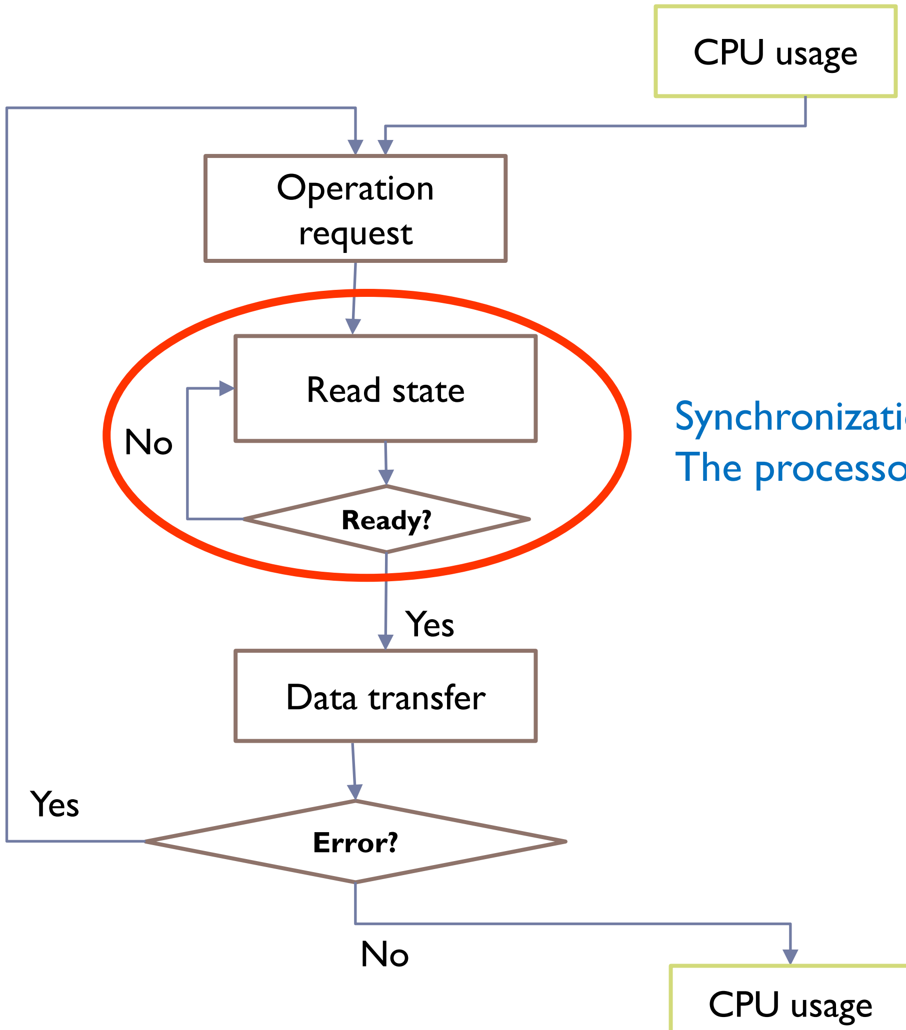
- 1,000,601 instructions are executed, and 1,000,000 are instructions executed in the synchronization loop (el 99,9%)
  - It is a waste of processor cycles
  - The CPU does not perform useful work

```
.data
    data1: .space  400
.text
.globl main
main:        li $t3 0
    loop1:   li $t0 0
             sw $t0 0x500
    loop2:   lw $t1 0x504
             beqz $t1 loop2
             lw $t2 0x508
             sw $t2 data1($t3)
             add $t3 $t3 4
             bne $t3 400 loop1
```

Synchronization loop

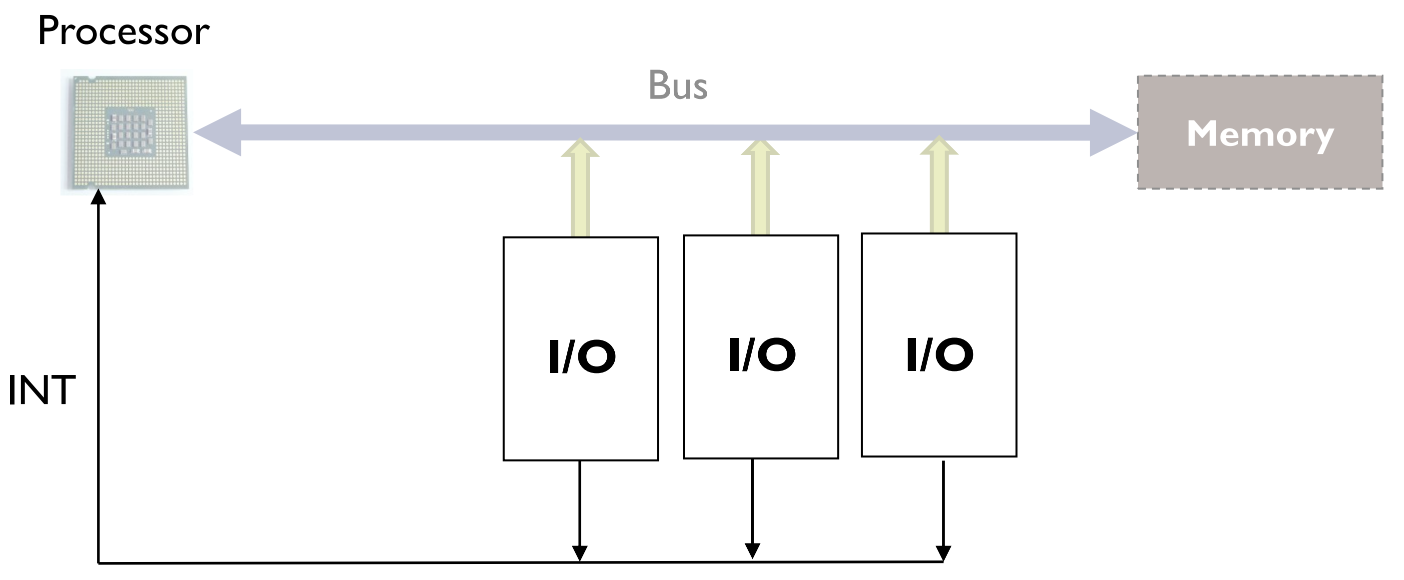
Transfer loop

Main problem of the programmed I/O

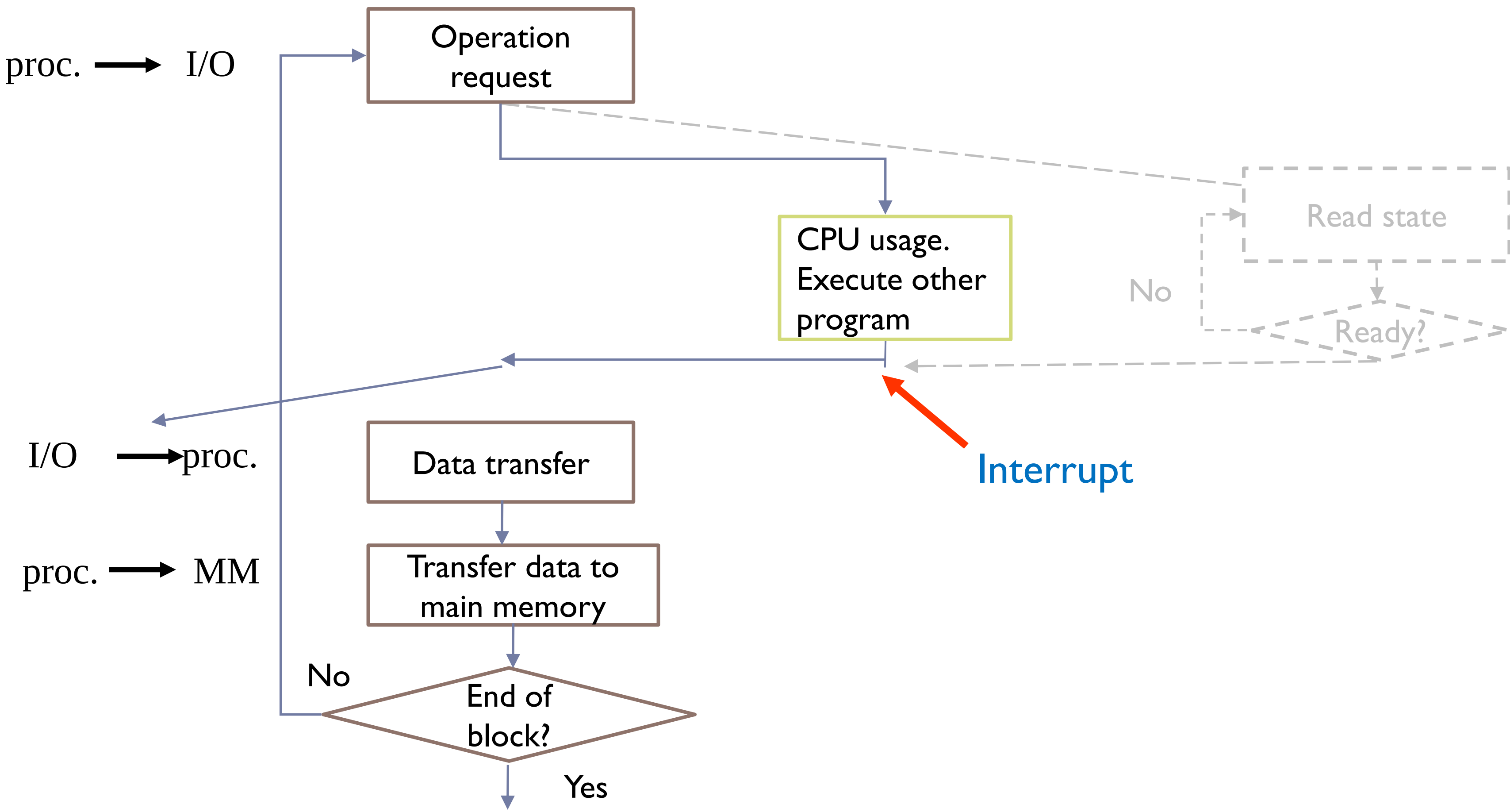


Synchronization loop
The processor does not perform useful work

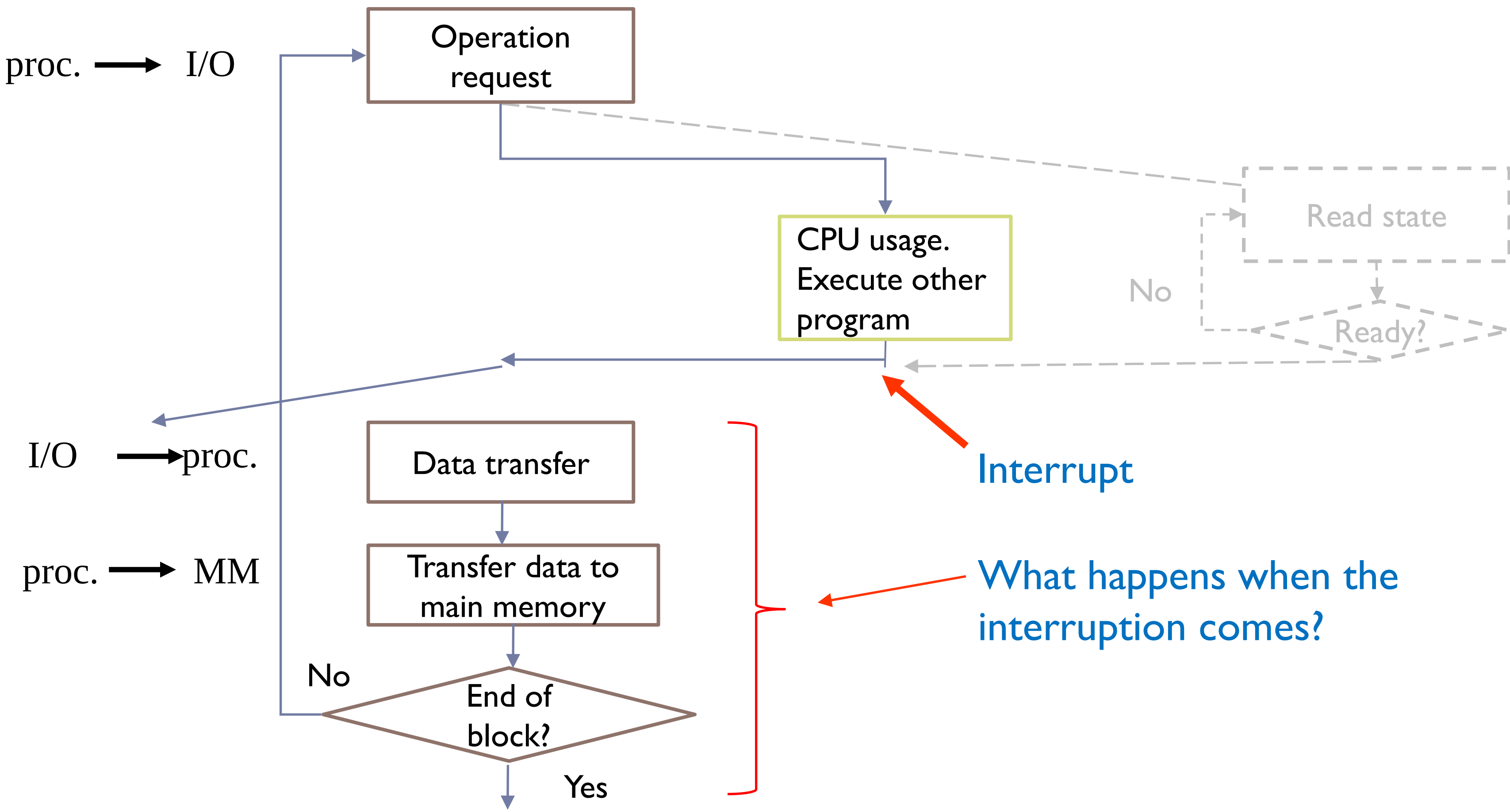
Interrupt driven I/O



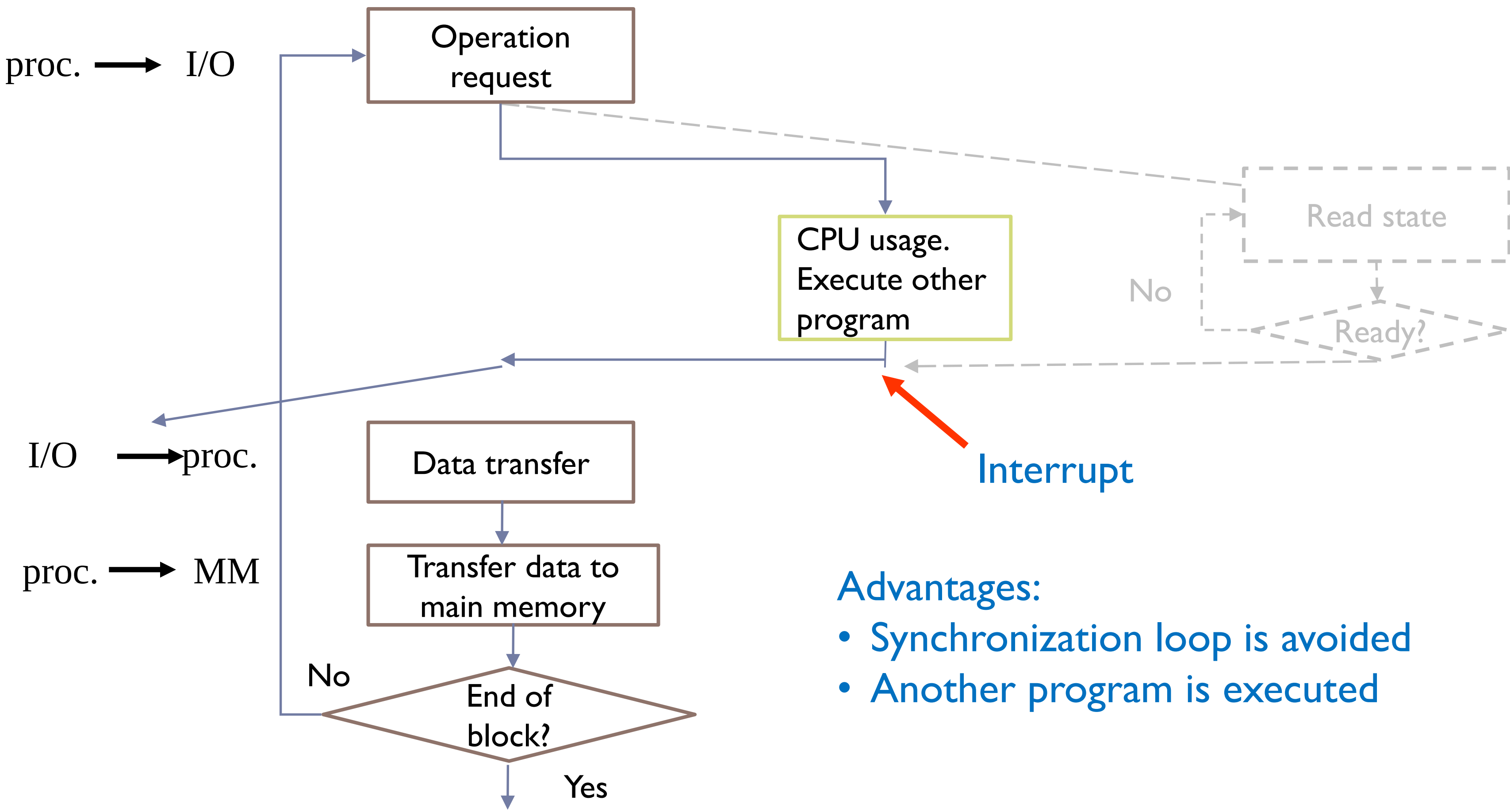
Interrupt I/O



Interrupt I/O

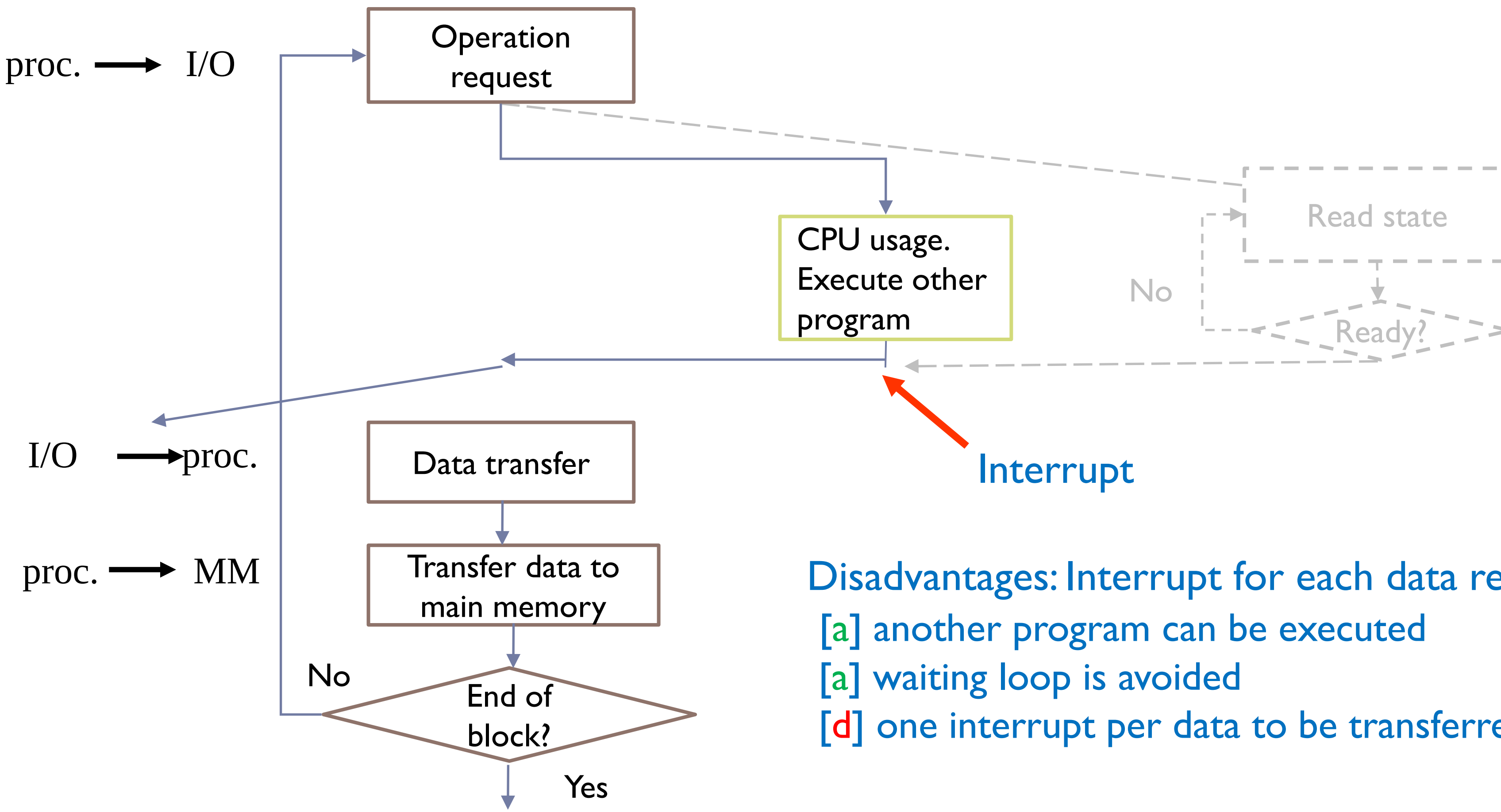


Interrupt I/O



Advantages:
- Synchronization loop is avoided
- Another program is executed

Interrupt I/O



Disadvantages: Interrupt for each data read:

[a] another program can be executed

[a] waiting loop is avoided

[d] one interrupt per data to be transferred...

Example

I/O module

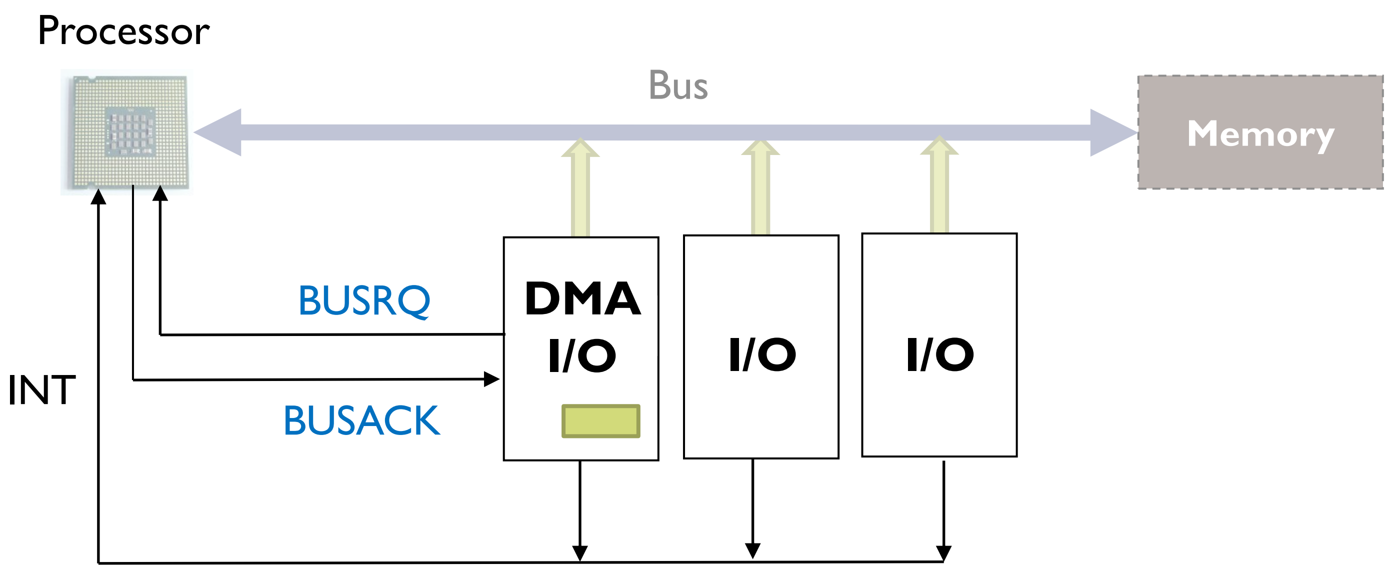
Addresses	
0x0504	State reg.
0x0508	Data reg.
0x0500	Control reg.

- Control information:
  - 0: read
  - 1: write
- State information:
  - 0: device not ready
  - 1: device (data) ready
- Memory-mapped I/O:
  - lw/sw MIPS instructions

```
request:
    // read request
    p.counter = 0;
     p.neltos = 100;
    out(0x500, 0) ;   // request read first element
    // Voluntary context switching (V.C.S.)
```

```
INT_05:
     in(0x508, &(p.status)) ;                  // read state
    in(0x50C, &(p.data[p.counter])) ;       // read data
    if ((p.counter < p.neltos) && (p.status == OK)) {
            p.counter++ ;
            out(0x500, 0) ;  // request read next elto.
    } else  { // process.state to READY  }
    return_interrupt # restore registers & return
```

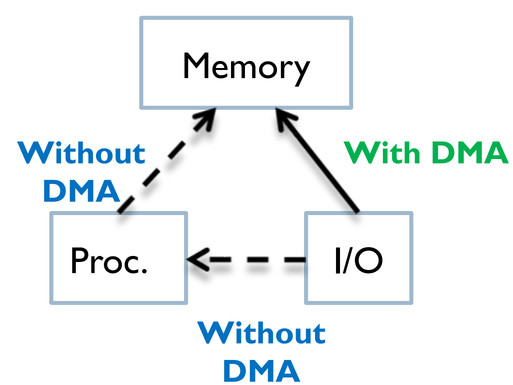
DMA I/O



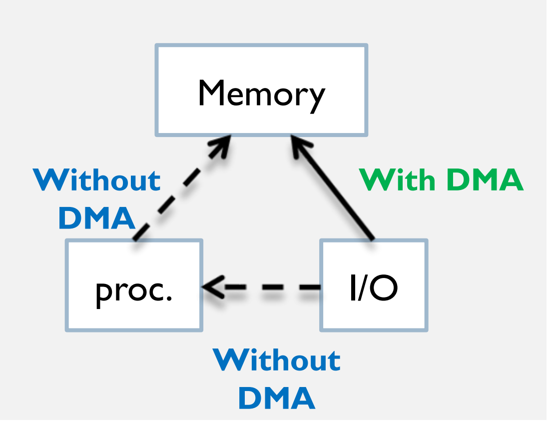
- A coordination is needed to control the access to memory from the processor and I/O modules

DMA I/O

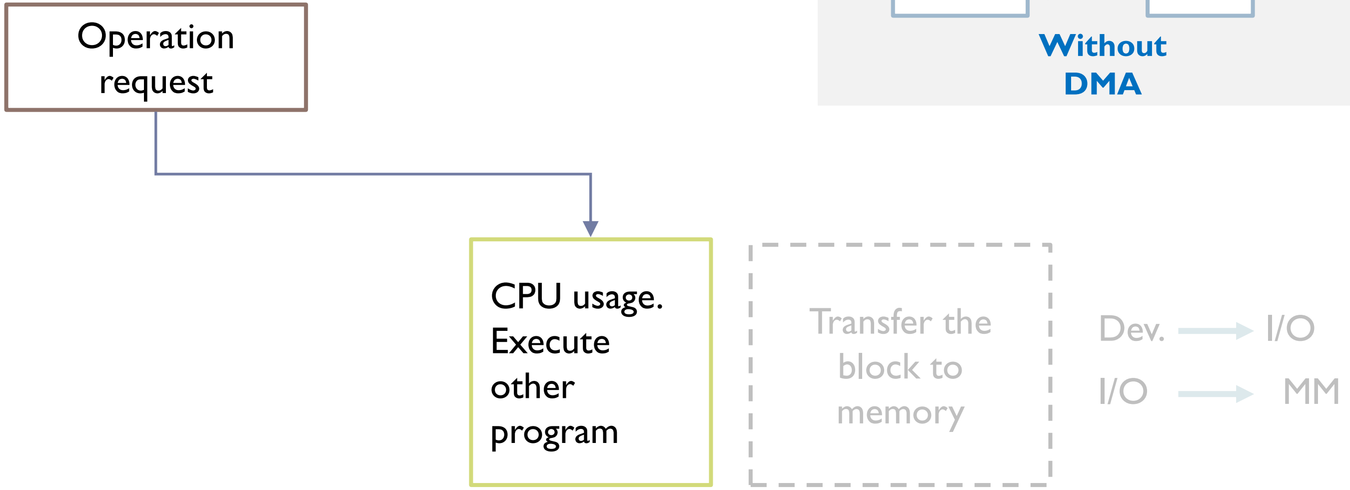
- DMA: Direct Memory Access
- CPU does not carry out the transfer between the I/O module and the memory
  - With interrupts the synchronization loop is avoided, but the transfer is carry out by CPU
  - For a block with N bytes, N interrupts are needed
- Using DMA, the whole transfer is done by the I/O module
  - Only one interrupt at the end



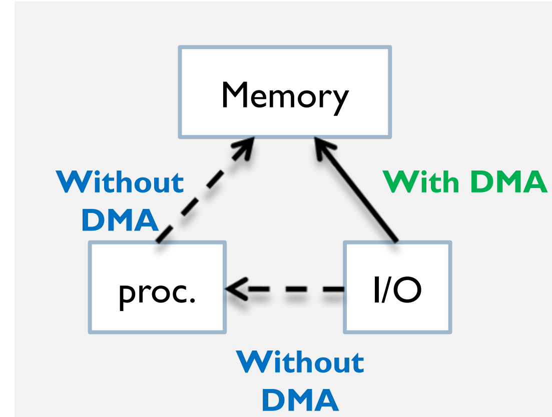
Transfer a block using DMA

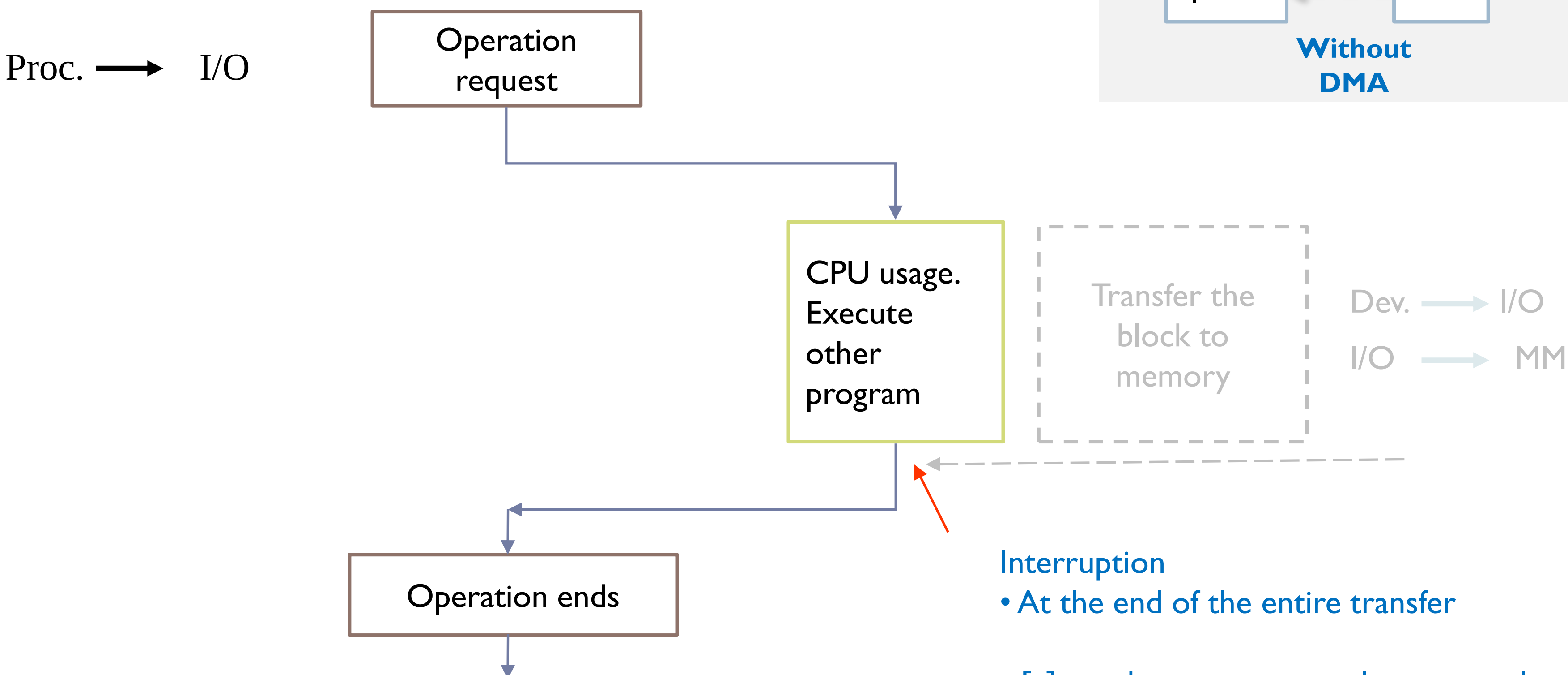


Proc. → I/O



Transfer a block using DMA

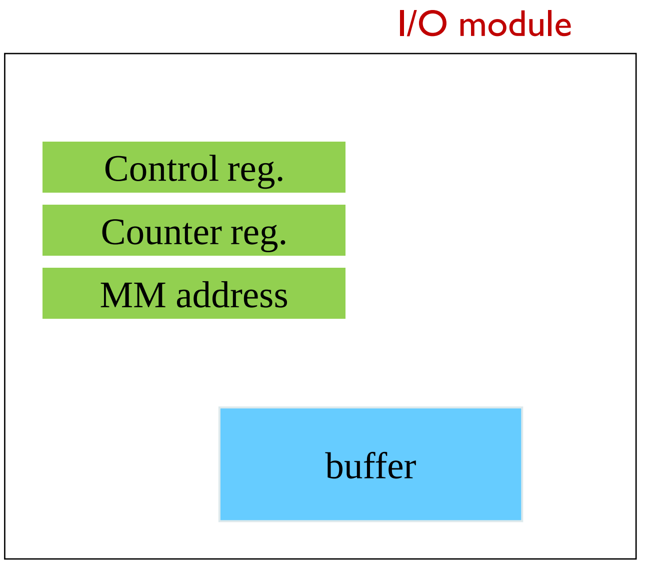




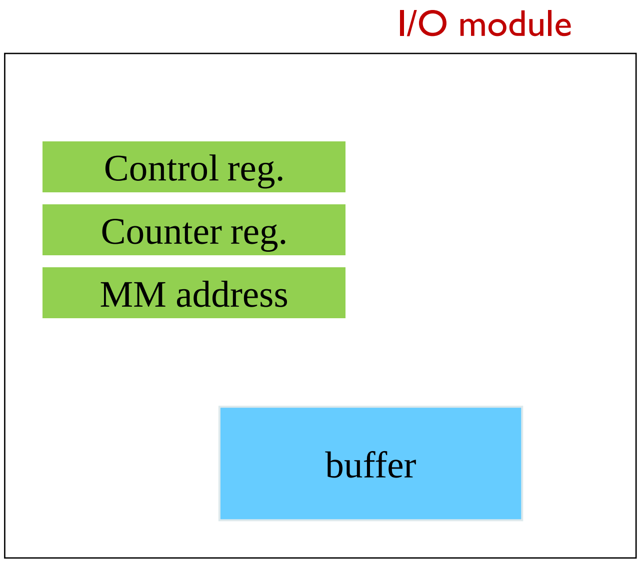
Interruption
- At the end of the entire transfer

- [v] another program can be executed
- [v] a single interrupt

Simplified structure of I/O module for DMA

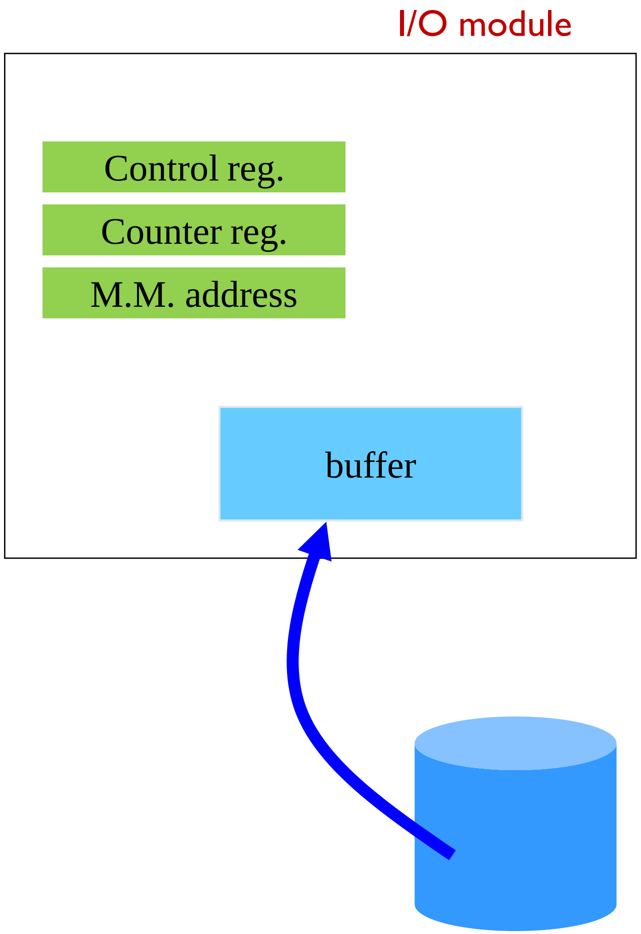


Data transfer with DMA



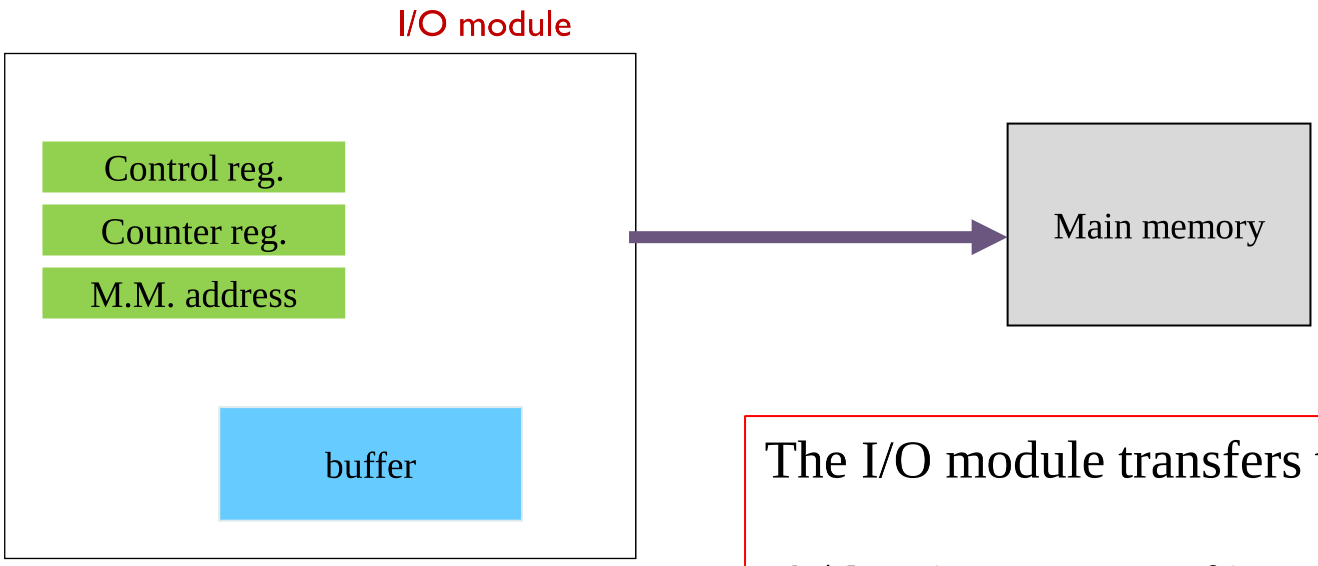
- The processor writes in I/O registers (using I/O instructions)
  - Operation (control reg.)
    - Read, write, etc.
  - The number of bytes to transfer (counter reg.)
  - Memory address where:
    - Data are stored (write in device)
    - Store the data (reading from device)

Data transfer with DMA



- I/O module transfers the data block from the device to the internal buffer inside the I/O module (in a reading operation)

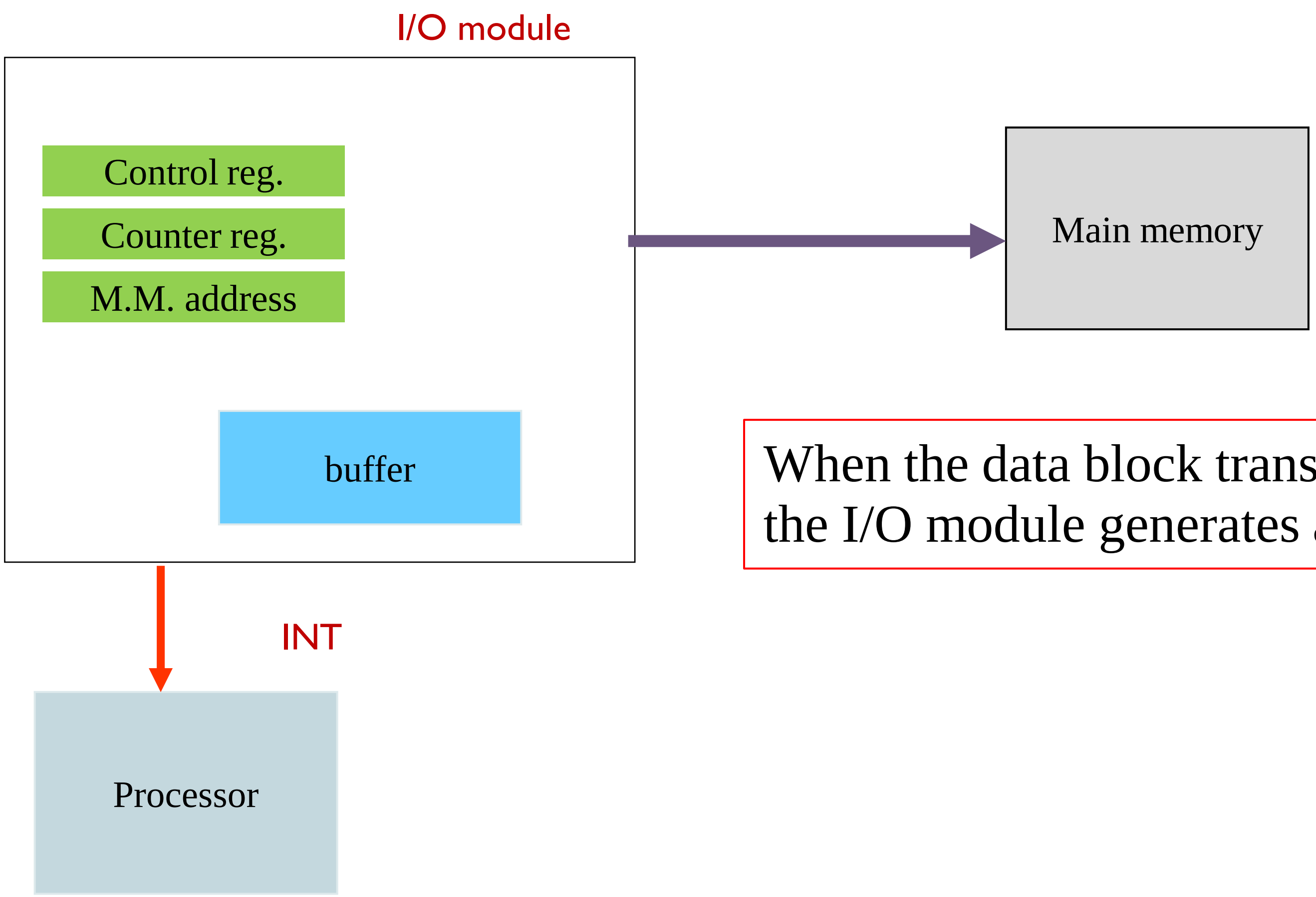
Data transfer with DMA



The I/O module transfers the data block:

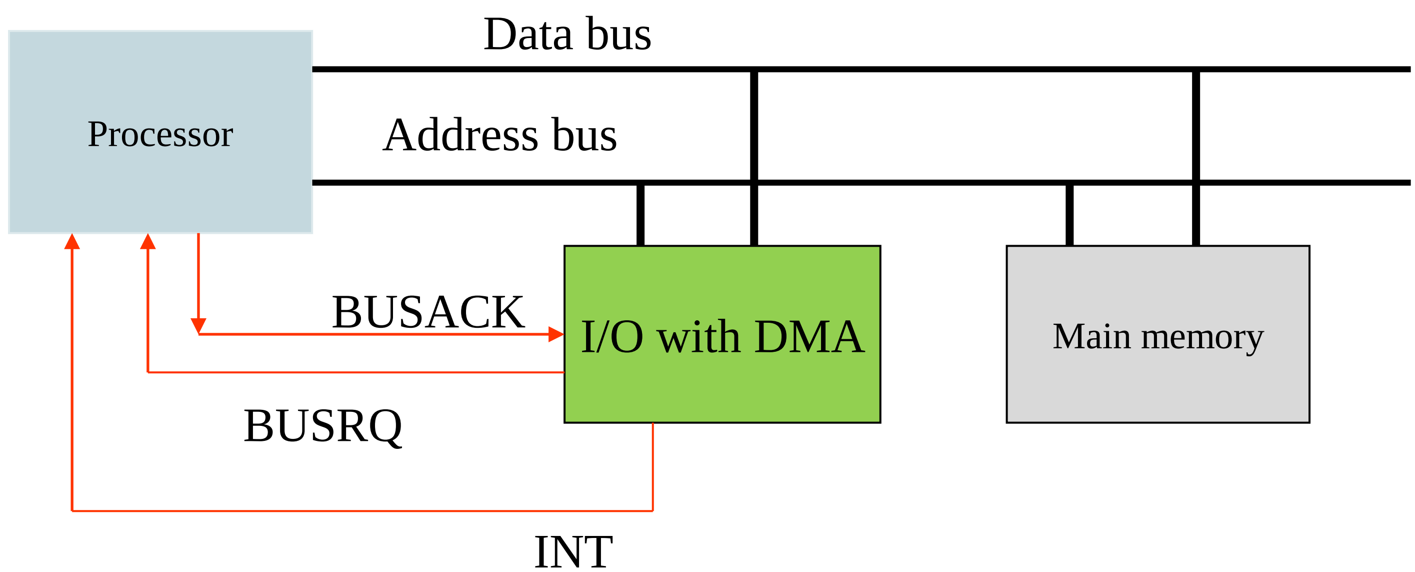
```
while (counter > 0)
{
   Byte (word) -> MP[MM_address]
   MM_address++
   counter--
}
```

Data transfer with DMA



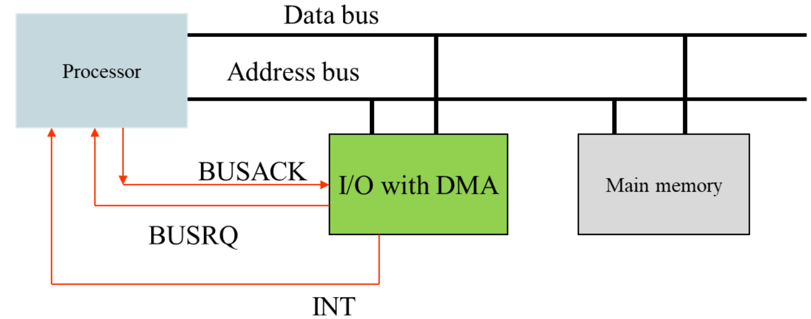
When the data block transfer is completed, the I/O module generates an interrupt

I/O module access to M.M.



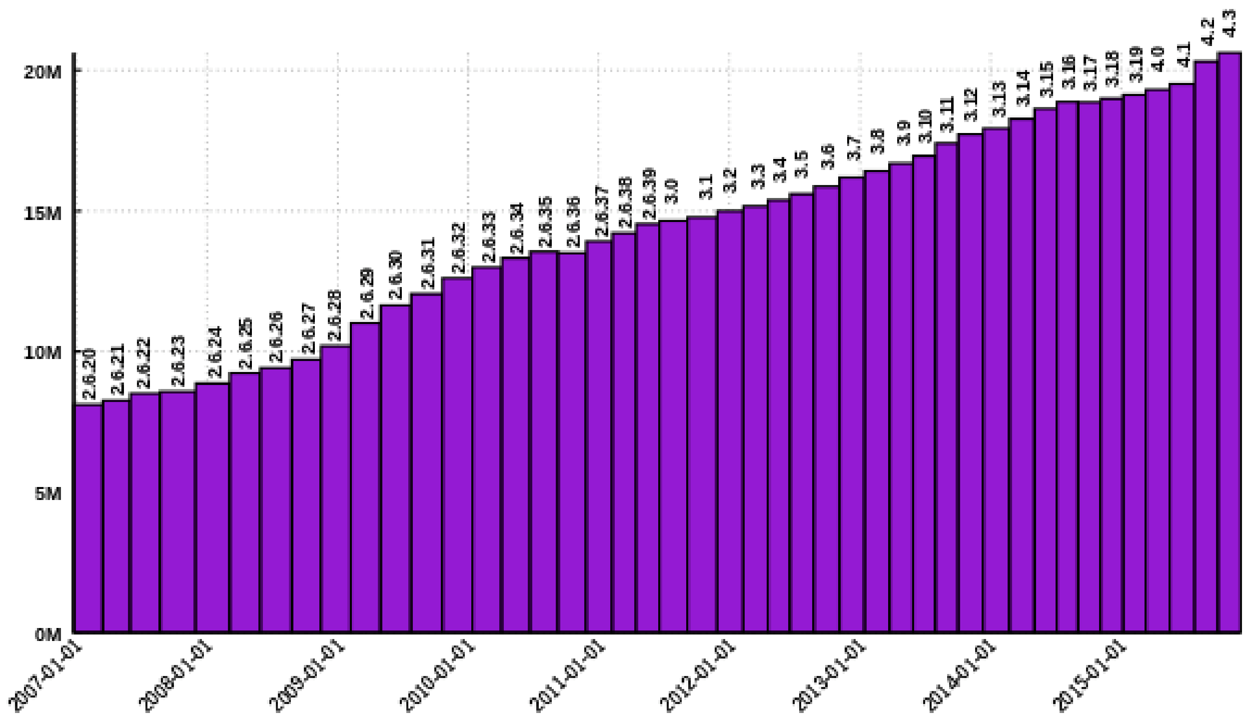
- A coordination is needed to control the access to memory from the processor and I/O modules

I/O module access to MM:
Cycle stealing



- When the I/O module is ready to transfer a word:
  - Activates BUSRQ signal to request bus access
  - At the end of each phase of an instruction, the processor checks this signal. If this signal is activated, the processor does not use the buses and activate the BUSACK signal
  - The I/O module access to memory and then deactivate BUSRQ signal
  - The processor then can use the buses
  - At the end of the data block transfer, the I/O module sends an interrupt signal to the processor.

Curiosity: the importance of drivers
Linux kernel



Lines of code of the Linux kernel

- 70% of Linux code is related to device drivers.

http://cityblogger.com/archives/2008/06/16/linux-kernel-stats

ARCOS Group

uc3m | Universidad Carlos III de Madrid

Lesson 6
Input/Output Systems

Computer Structure

Bachelor in Computer Science and Engineering