Computer Structure

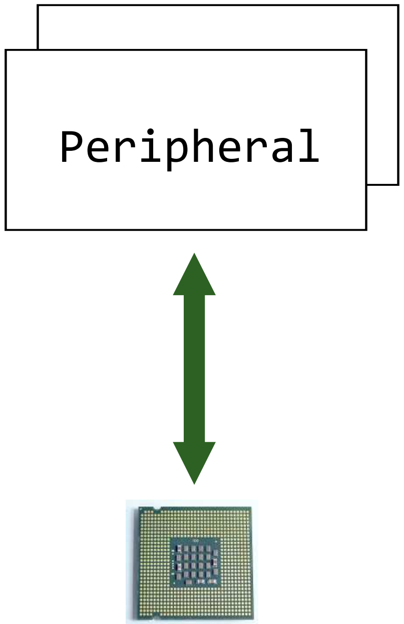
Unit 6. Input/Output Systems

Departamento de Informática
Grupo de Arquitectura de Computadores, Comunicaciones y Sistemas
UNIVERSIDAD CARLOS III DE MADRID

Félix García Carballeira, Alejandro Calderón Mateos, José Daniel García Sánchez

Contents

Peripheral concept

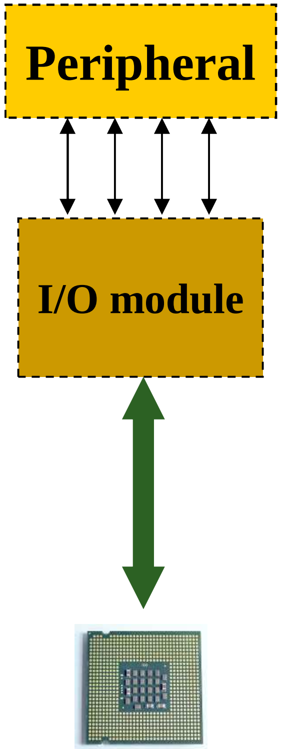


- Peripheral:
  - Device attached to a CPU connected by using input/output modules
  - Used to store information or for the communication with the computer

Classification of peripherals

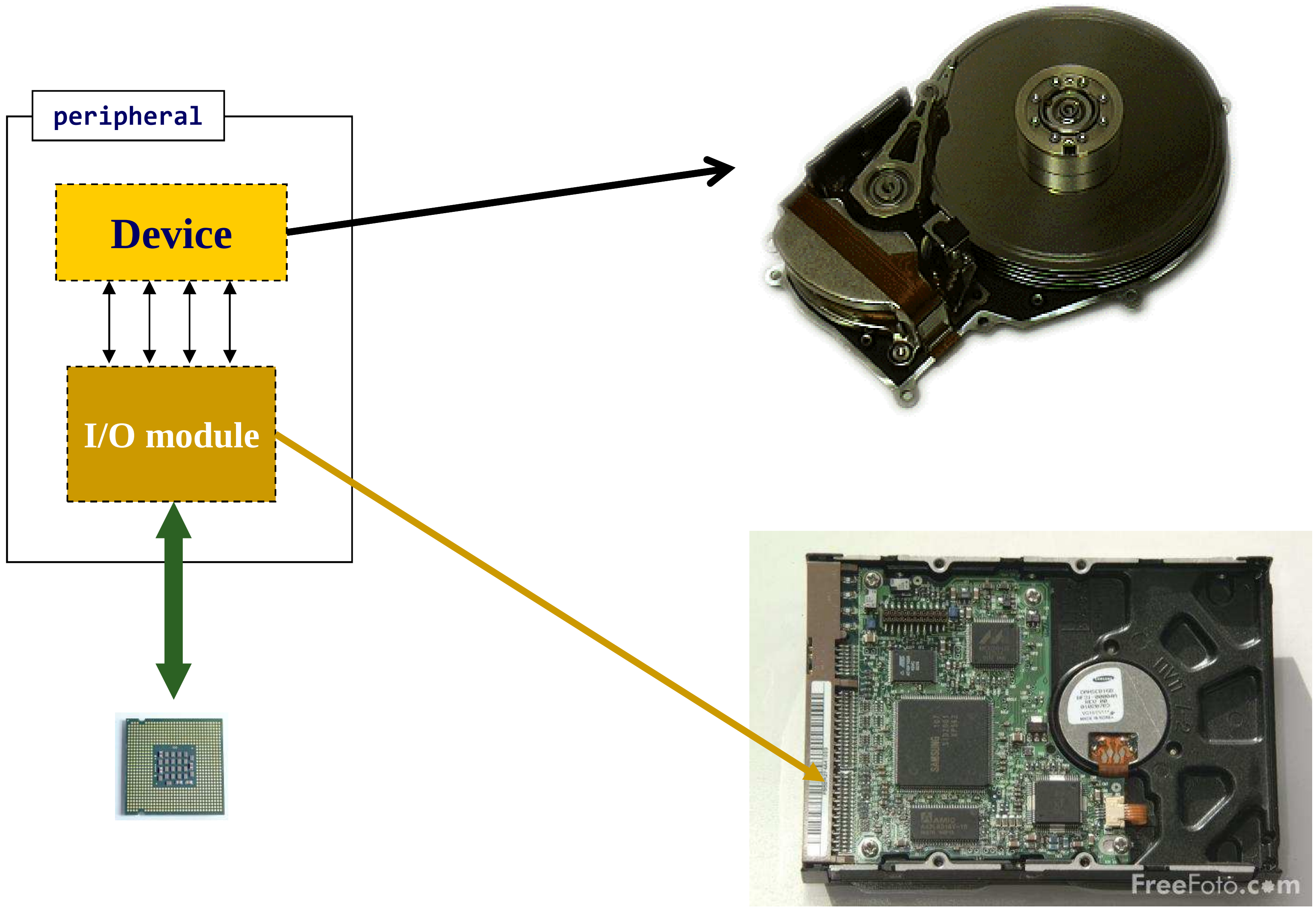
- **Communication**:
  - Human-computer
    - (Terminal) keyboard, mouse, …
    - (Printer) plotter, scanner, …
  - Computer-computer
    - Modem, network adapter
  - Physical environment
    - (read/action) x (analogical/digital)
- **Storing**:
  - Direct access (disks, DVD, …)
  - Sequential access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

Peripheral and I/O units



- Peripheral
  - Device attached to the computer
- I/O module or I/O unit
  - Also called controller
  - Interface between the processor and the devices, that hides the devices characteristics

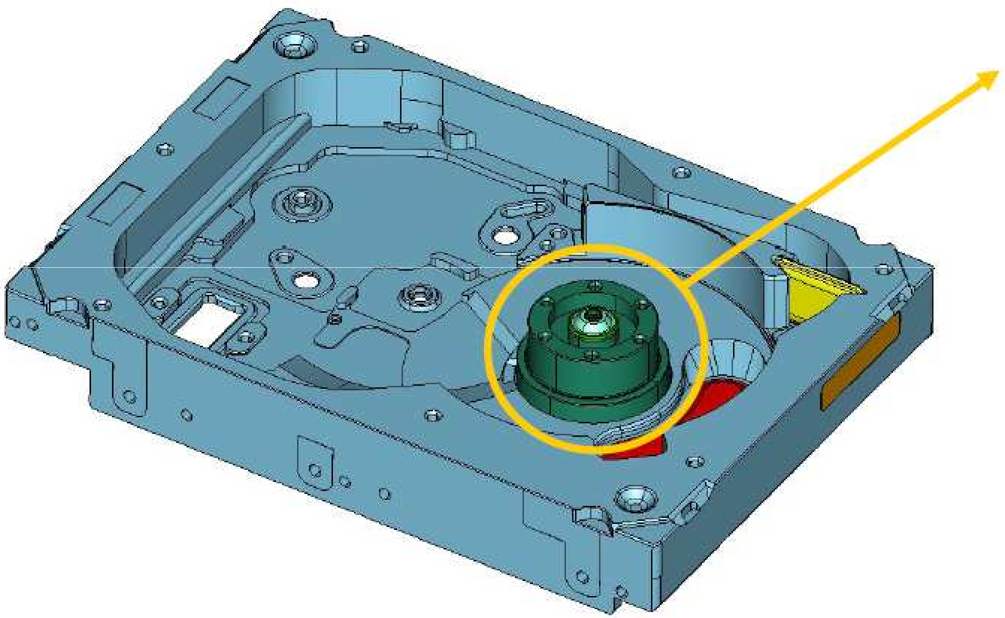
Example: Disk drive



History

- First disk drive was built in 1956
  - Installed on a IBM RAMAC 305
  - 50 disks of 24”
  - 5 MB of data
  - Hire per year: 35000 $
- In 1980 appeared the first 5 ¼” disk
  - 5 MB
  - 10000 %
- In 1997 appeared the first disk with 15000 RPM

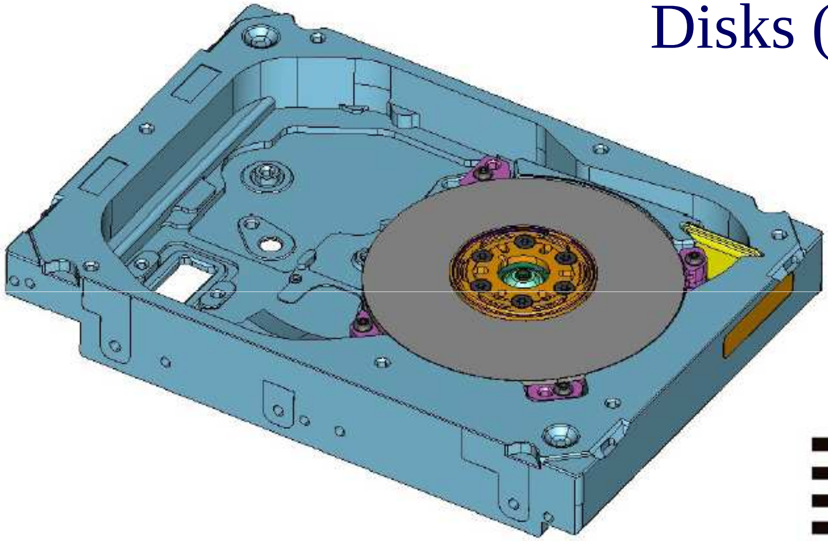
Disk anatomy



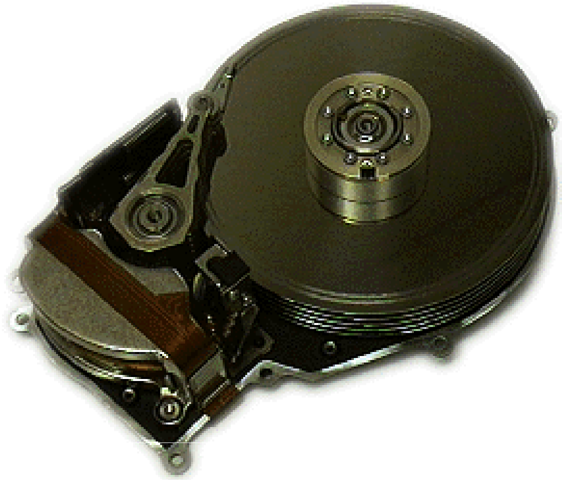
Engine

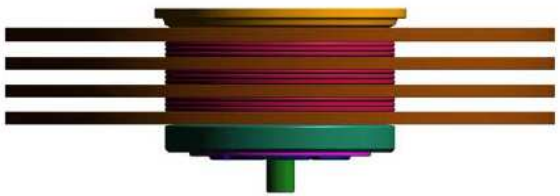
www.snia.org

Disk anatomy



Disks (plates)

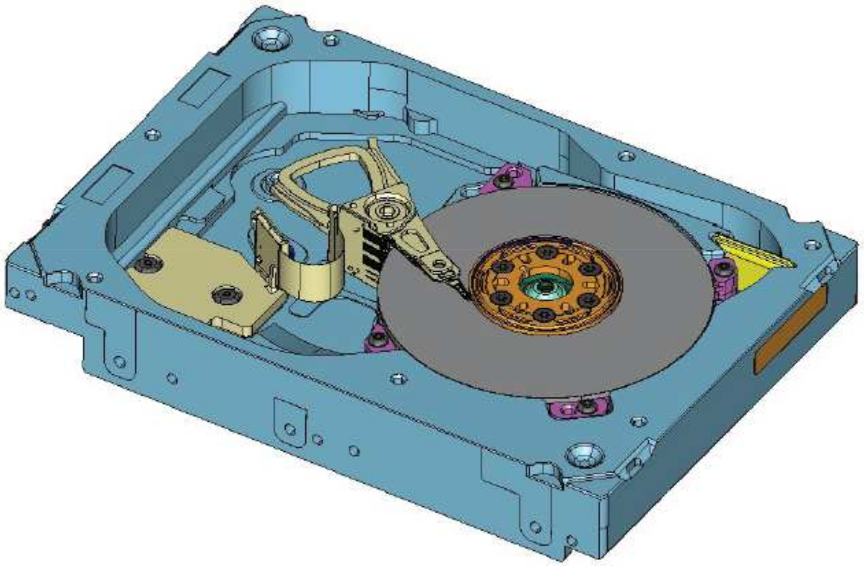


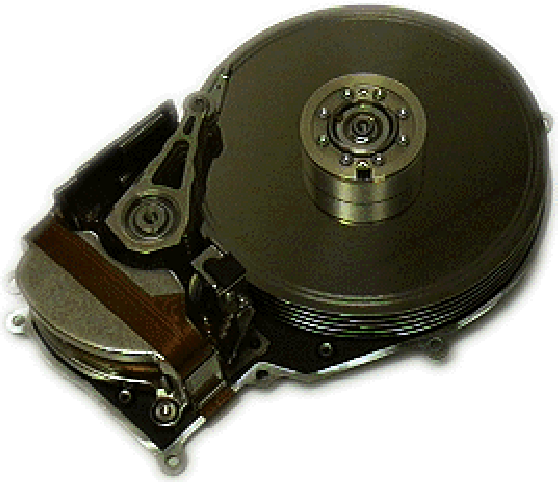


www.snia.org

Disk anatomy

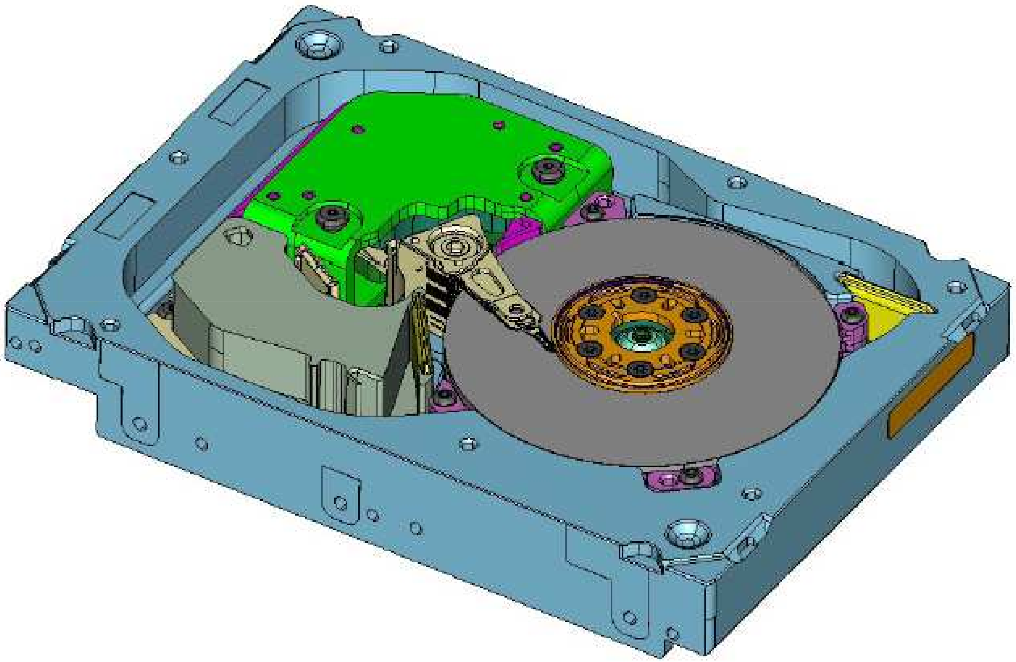
Read/write heads





www.snia.org

Disk anatomy



control and mechanic

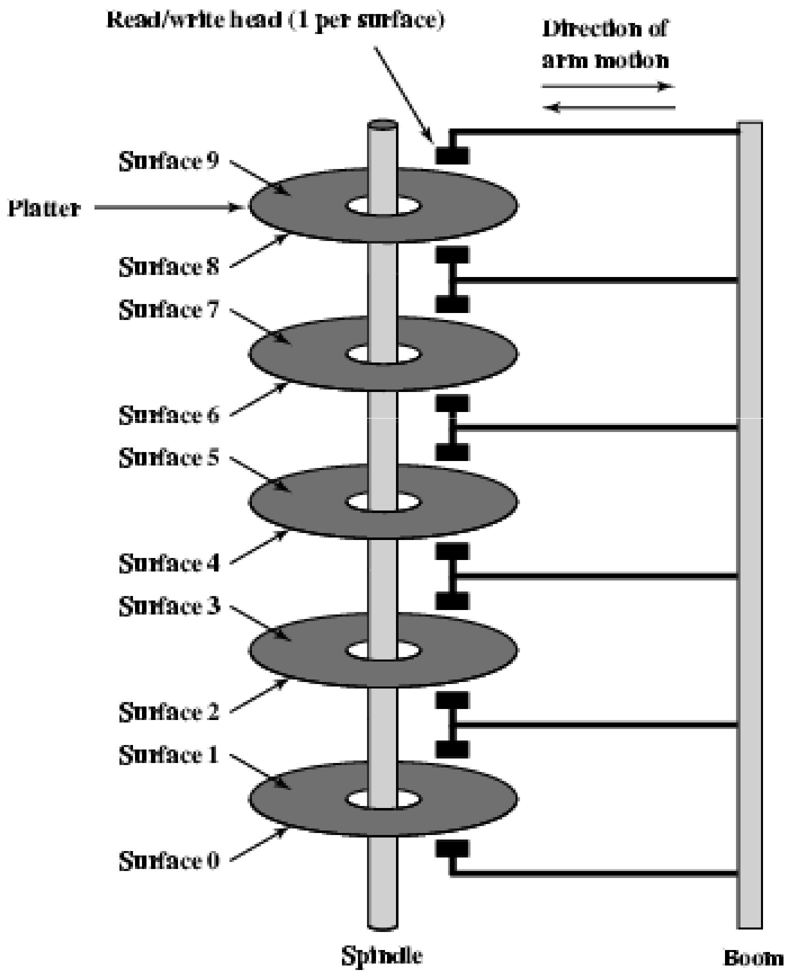
www.snia.org

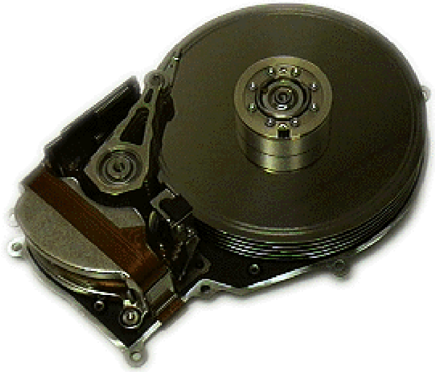
Disk anatomy

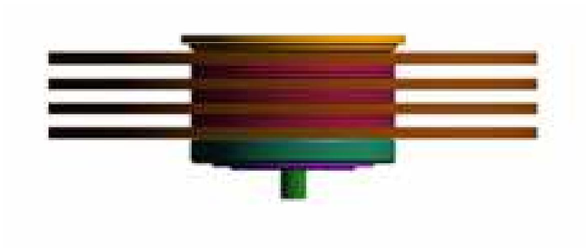
www.snia.org

- Disk controller
  - Scheduling commands
  - Error correction
  - Optimization
  - Check integrity
  - RPM control
  - Disk cache

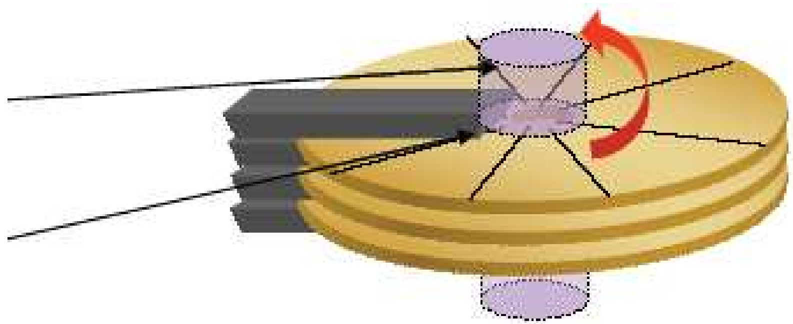
Multiple plates





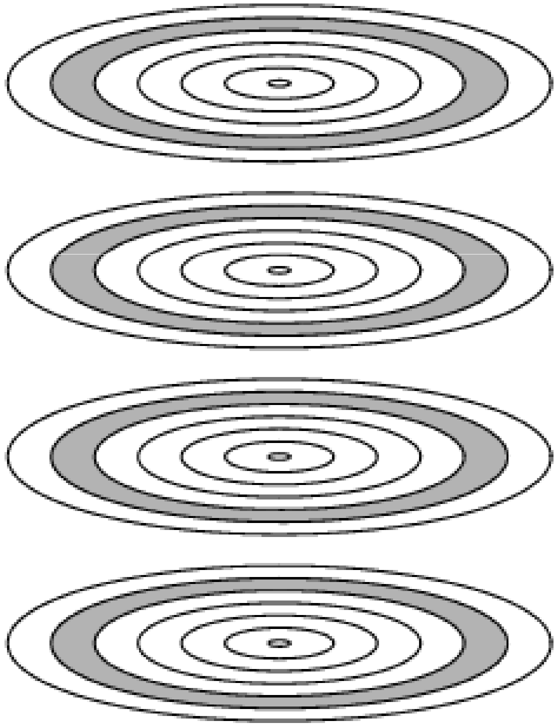


rotation



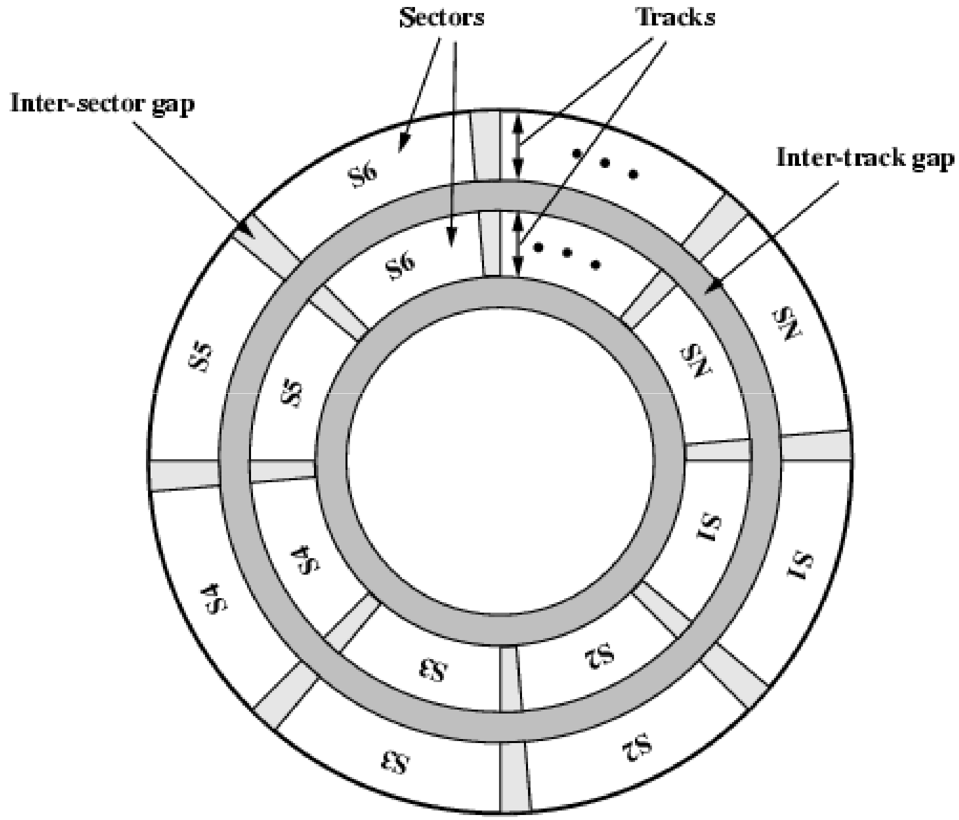
Cylinders

- Data accessed by all disk heads

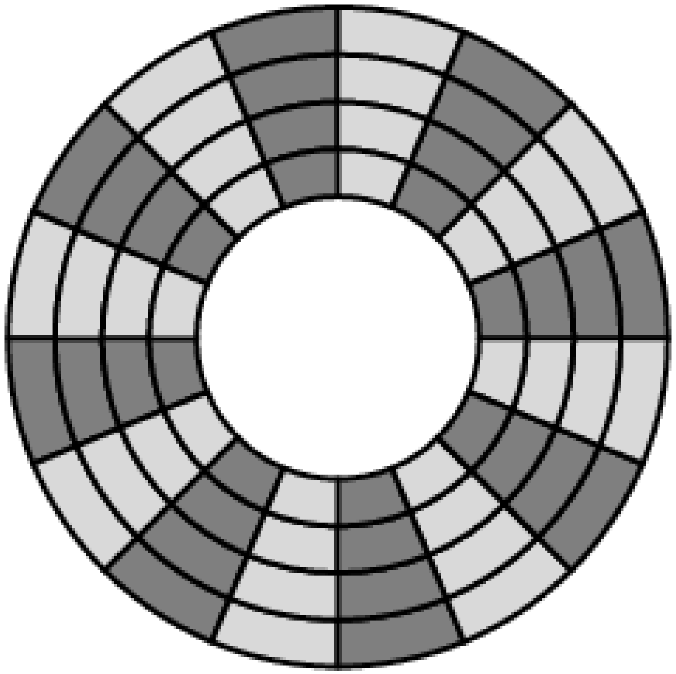


Tracks and sectors

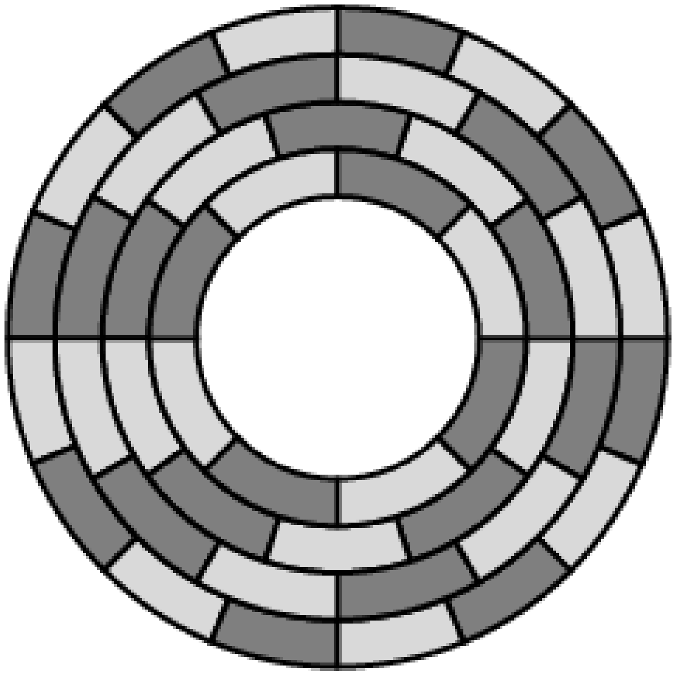
- Tracks
  - Concentric rings
- Sector
  - Division of the disk surface made when the disk is formatted (512 bytes)
- Blocks
  - File System writes in blocks
  - Sector groups



Distribution of sectors

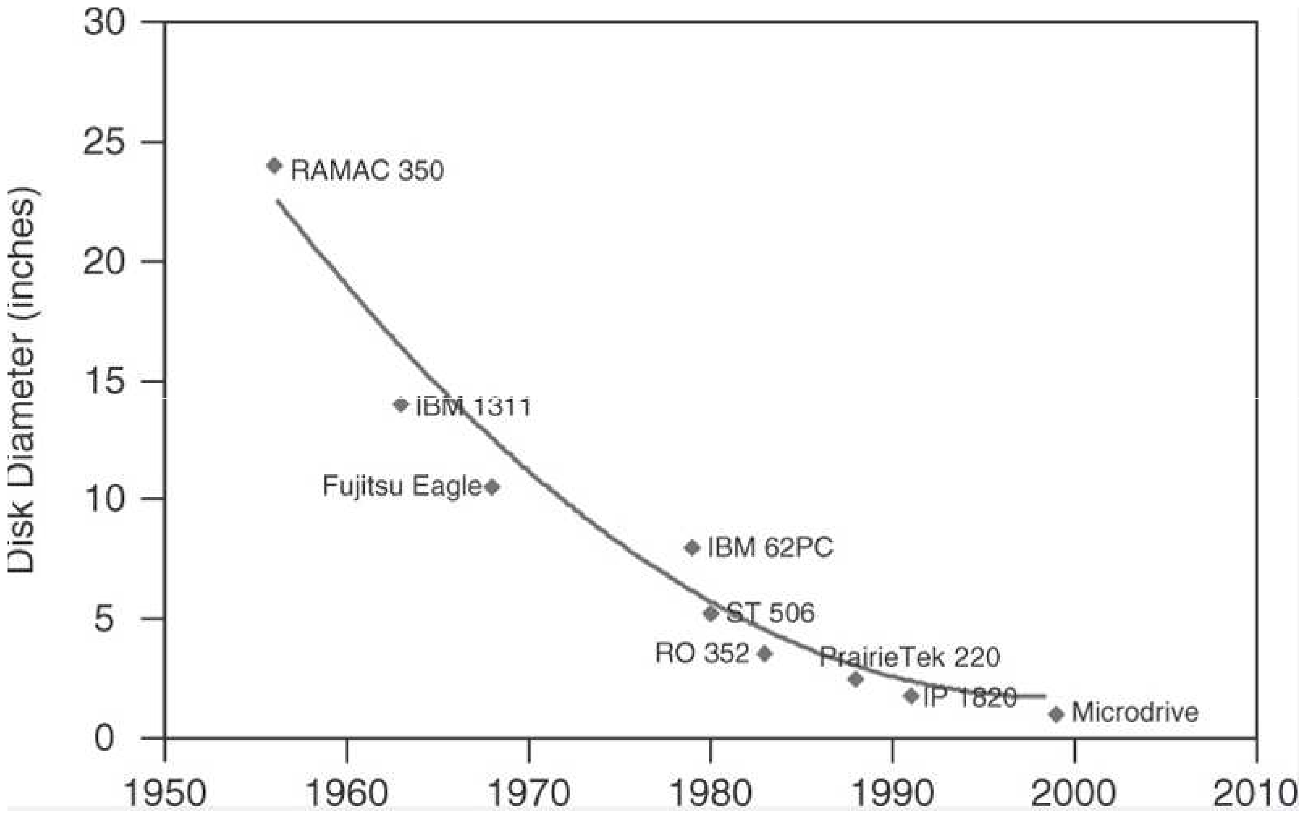


(a) Constant angular velocity



(b) Multiple zoned recording

Evolution of disk sizes



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

Capacity

- Bits per inch
  - Depend on read/write head, physical medium, disk rotation and bus velocity
- Tracks per inch
  - Depend on read/write head, physical medium, head precision, disk precision

Storage capacity

- For constant angular velocity disks:
  - n_s: number of surfaces
  - p: trakcs per surface
  - s: sectos per track
  - t_s: bytes per sector

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

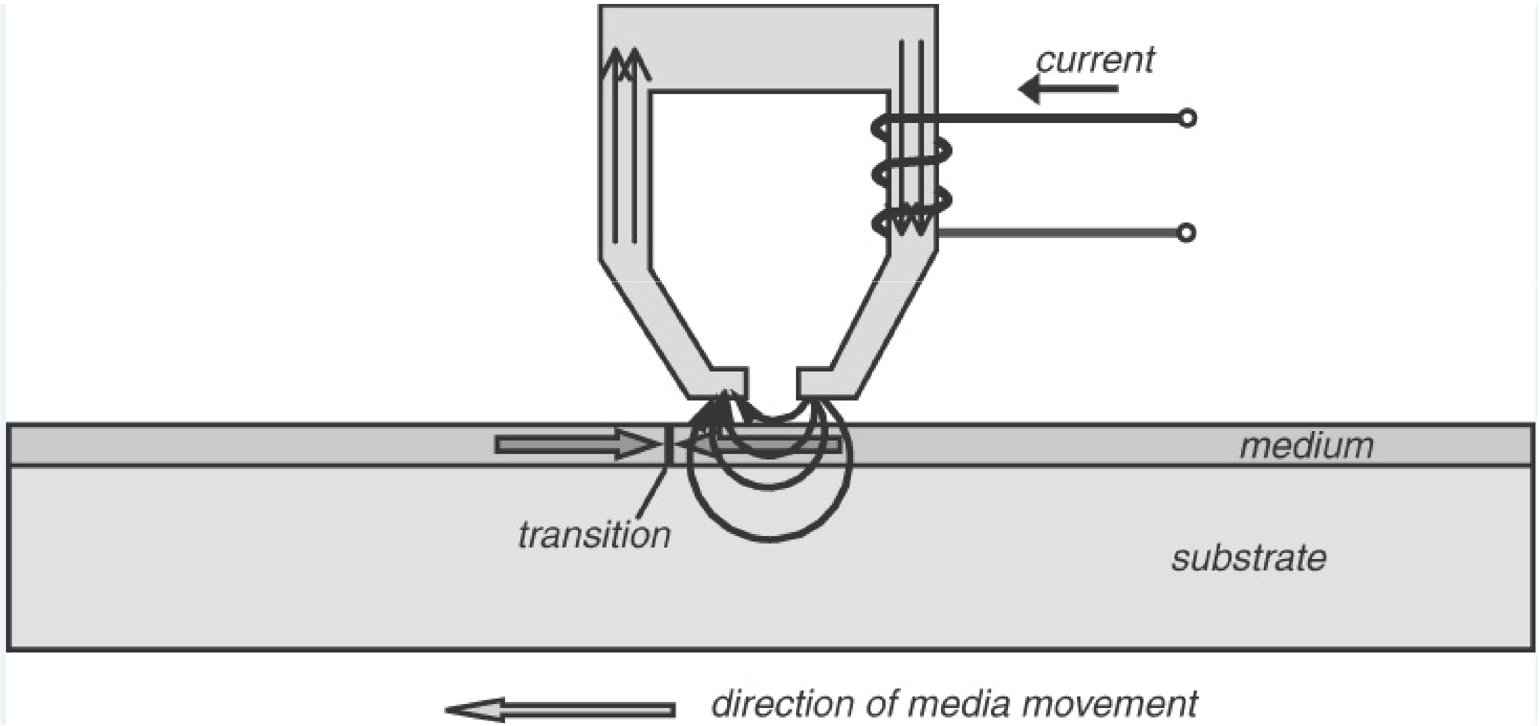
- For multple zone recording:
  - z: number of zones
  - p_i: number of tracks per zone I
  - s_i: sectors per track in zone i

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

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

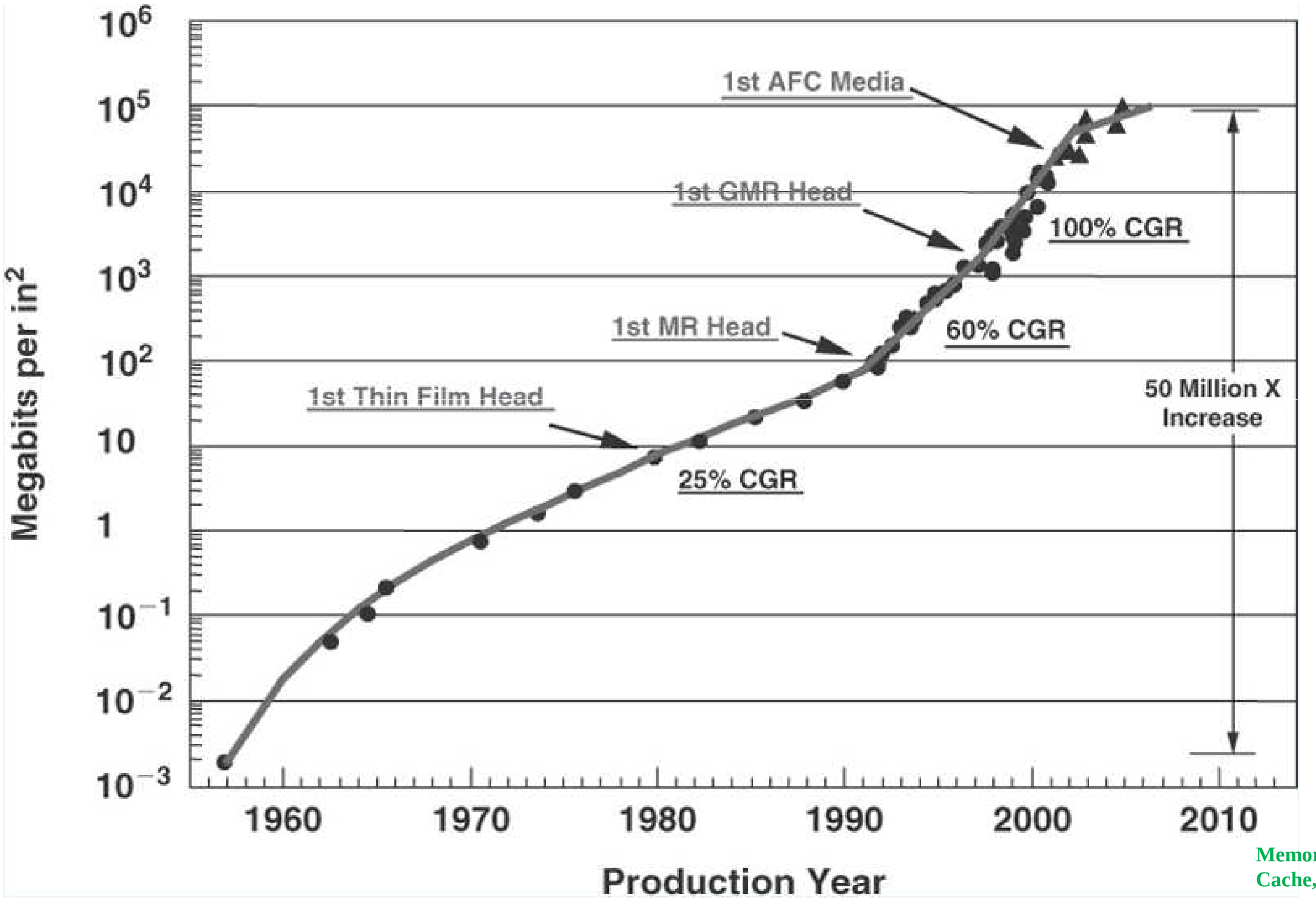
Areal density

- Number of bits that can be recorded per square inch (AD):

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

- 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 inc
- The cost per bit has improved in a factor of 1.000.000 between 1983 and 2011

Evolution of areal density

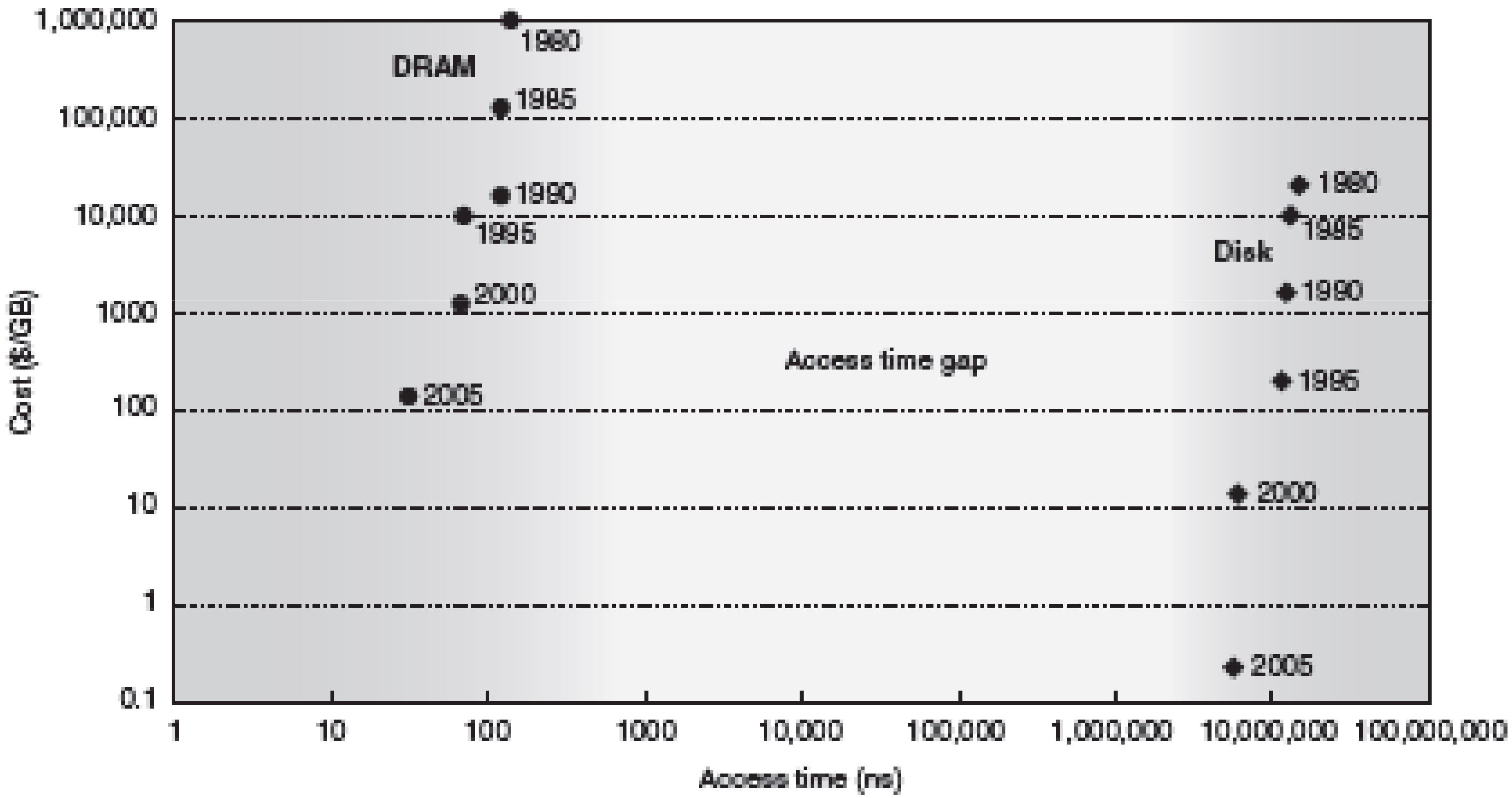


Memory Systems
Cache, DRAM, Disk
Bruce Jacob, Spencer Ng, David Wang
Elsevier

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 2011:
  - A disk transfers 200 MB/s in a disk of 600 GB and costs 400 $
  - A 4GB DRAM memory transfers 16.000 MB/s and costs 200$

Disks and main memory



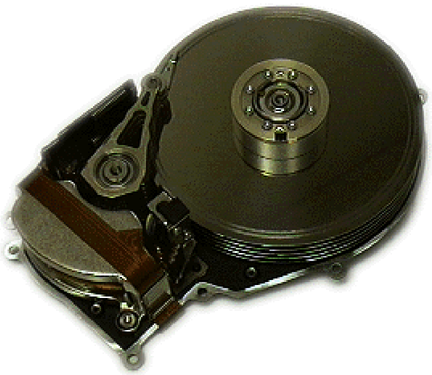
Patterson y Hennesy

Addressing

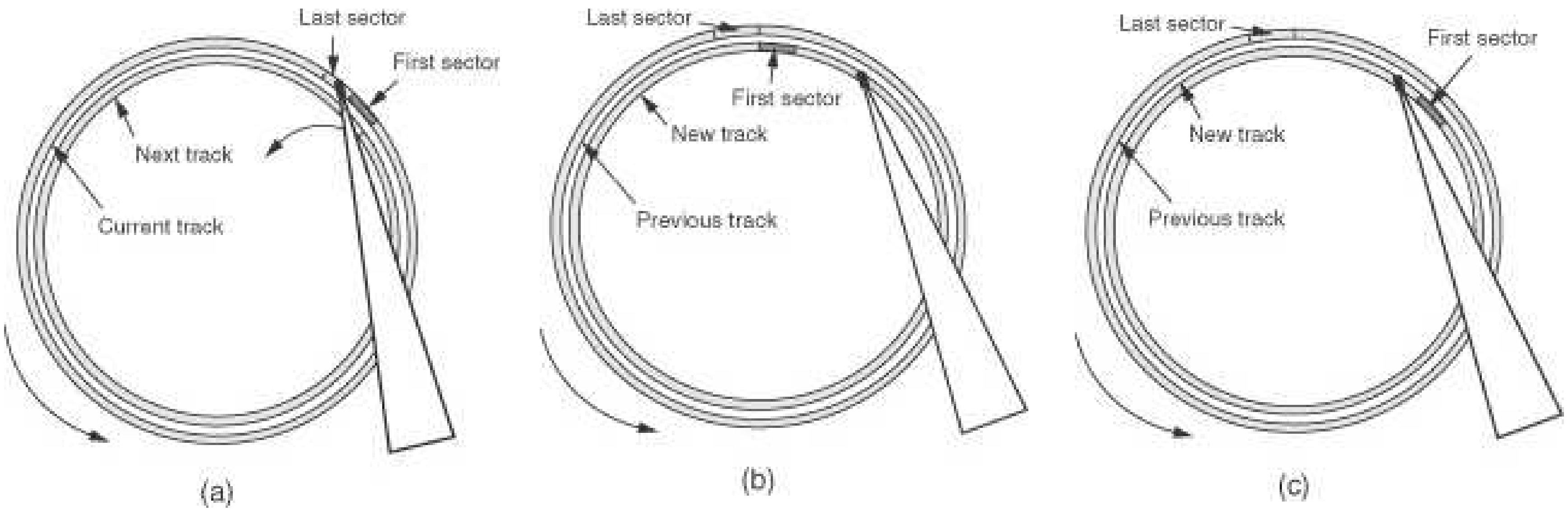
- Types of addressing:
  - Physical addressing: cylinder-track-sector.
  - Logical blocks addressing (LBN)
    - Each sector has a logical number and the mapping is done by the disk
- Current disk controllers do the mapping between LBN and physical addresses

Access time

- Seek time (T_{seek}): time to move the head from the current cylinder to the target cylinder
- Rotational latency(T_{rotate}): delay waiting for the rotation of the disk to bring the required disk sector under the read-write head
  - Average rotational latency is calculated as half time it takes the disk to do one revolution
- Data transfer time ($T_{transfer}$): amount of data divided by the data transfer rate
- Access time
  - $T_{access} = T_{seek} + T_{rotate} + T_{transfer}$



Seek and rotation



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

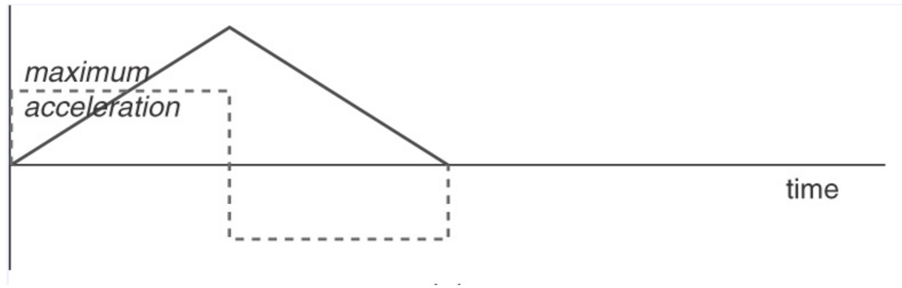
Seek time

- Elements to include:
  - Head acceleration time (T_{acc})
  - Coast period (T_c) for long distances
  - Deceleration time(T_{dec})
  - Head Settling time (T_s).
  - The average seek time is generally taken to be the average time needed to seek between two random tracks on the disk, called $D_{average}$

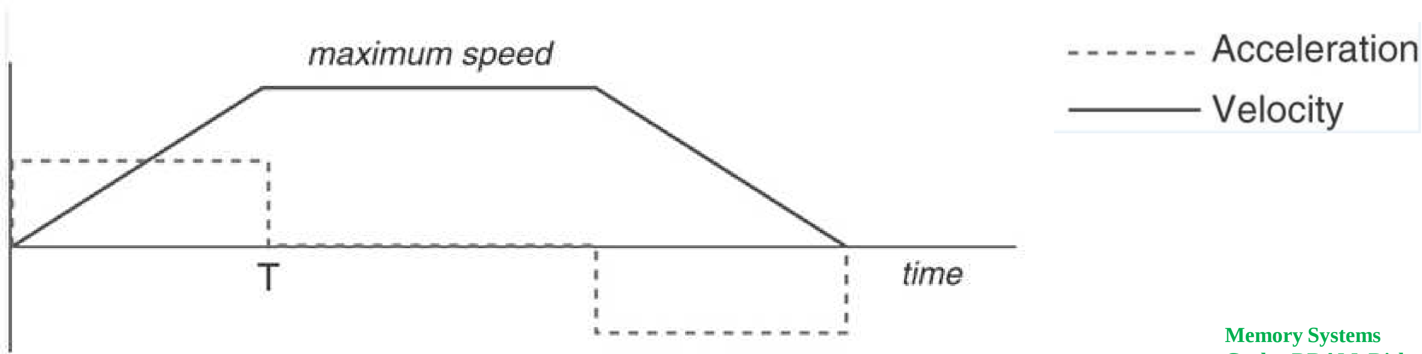
$$D_{average} = \frac{1}{2} \times a_{acc} \times t_{acc}^2 + a_{dec} \times t_{dec}^2$$

Seek time

- Seek time for near tracks



- Seek time for long tracks



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

Seek time

- When the acceleration time is equal to the deceleration time:

$$t_{acc} = t_{dec} = \sqrt{\frac{D_{average}}{a}}$$

$$t_{seek} = 2 \times \sqrt{\frac{D_{average}}{a}} + t_s$$

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

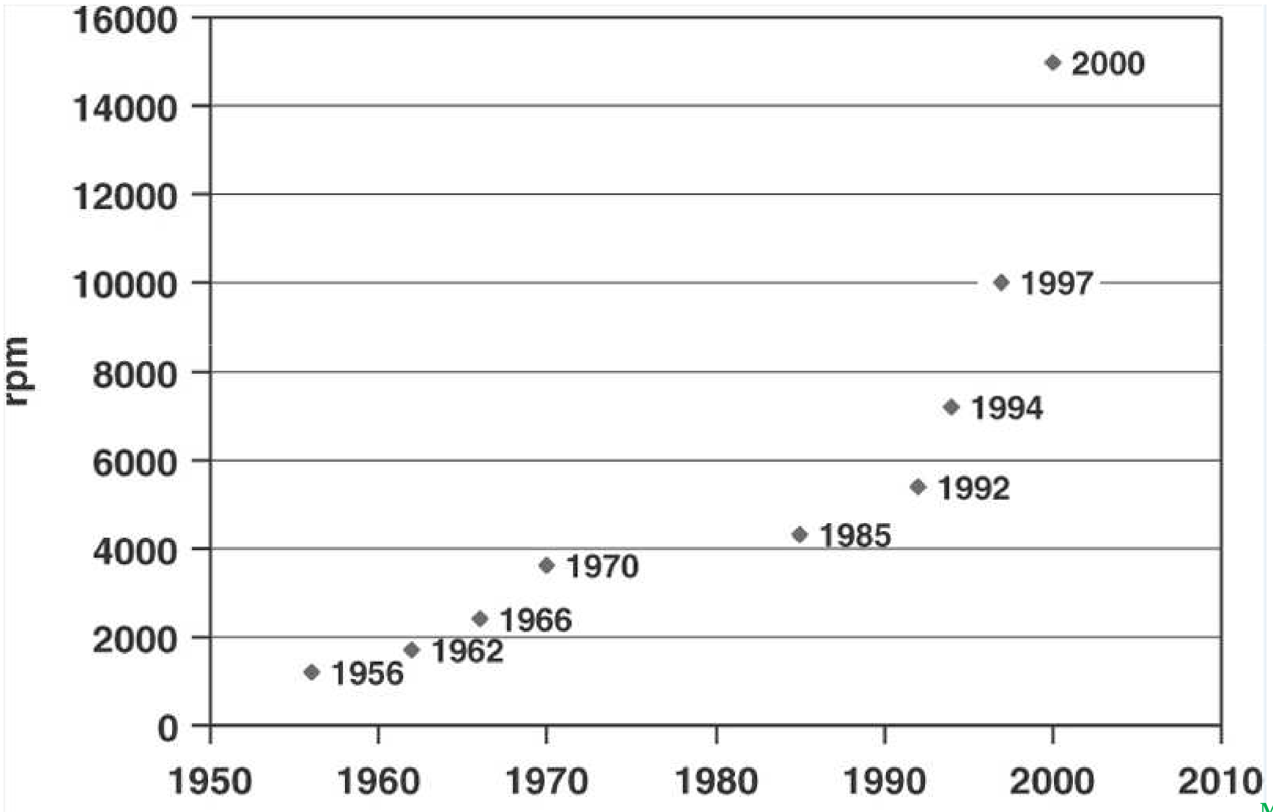
$$T_{rotate} = \frac{1}{2} \times \frac{60 \times 10^3}{RPM}$$

- *Zero-latency access*
  - New feature of modern disk drives, can start transferring data when the disk head is positioned above any of the sectors in a request
  - If multiple contiguous sectors are required to be read, the disk head can read the sectors from the media into its buffer in any order

Rotational speed and rotational latency

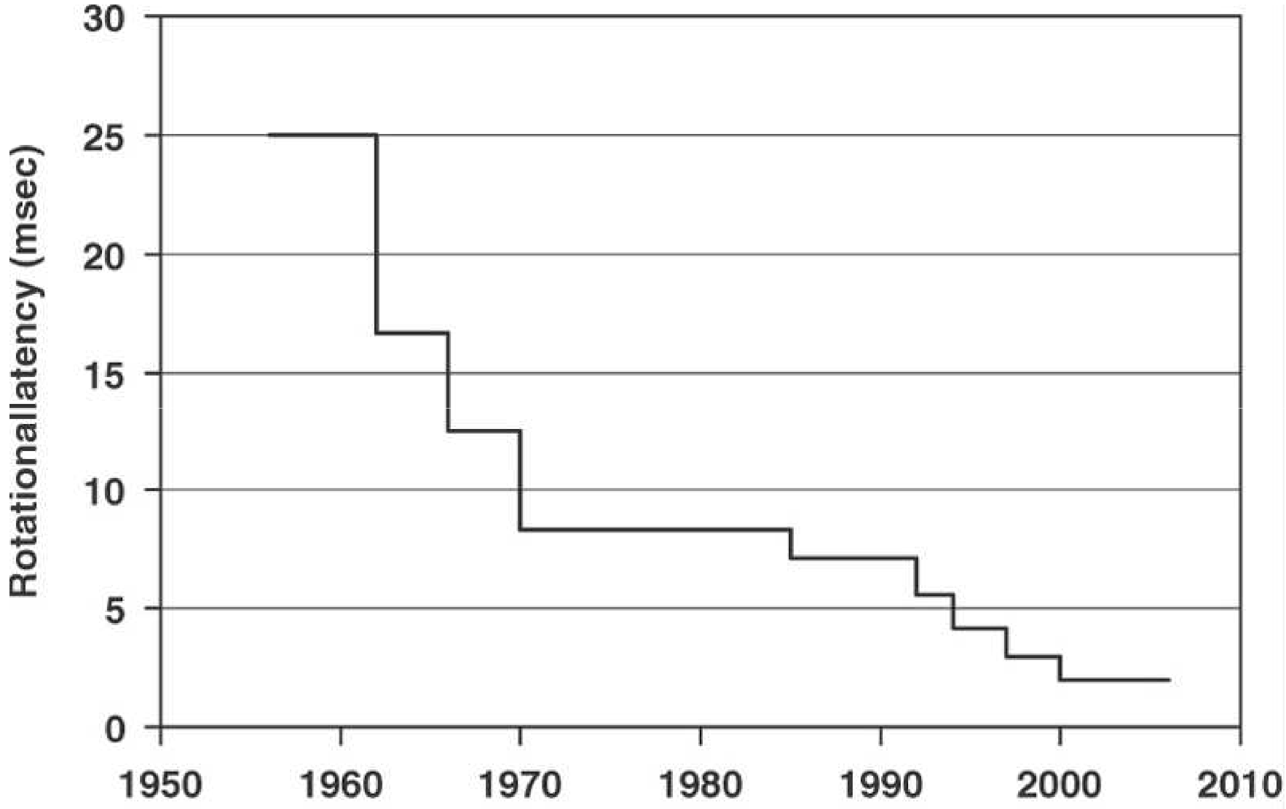
Rotational Speed	Rotational Latency (in ms)
5400	5.6
7200	4.2
10000	3.0
12000	2.5
15000	2.0

Evolution of RPM



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

Evolution of rotational latency



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

Data Transfer time

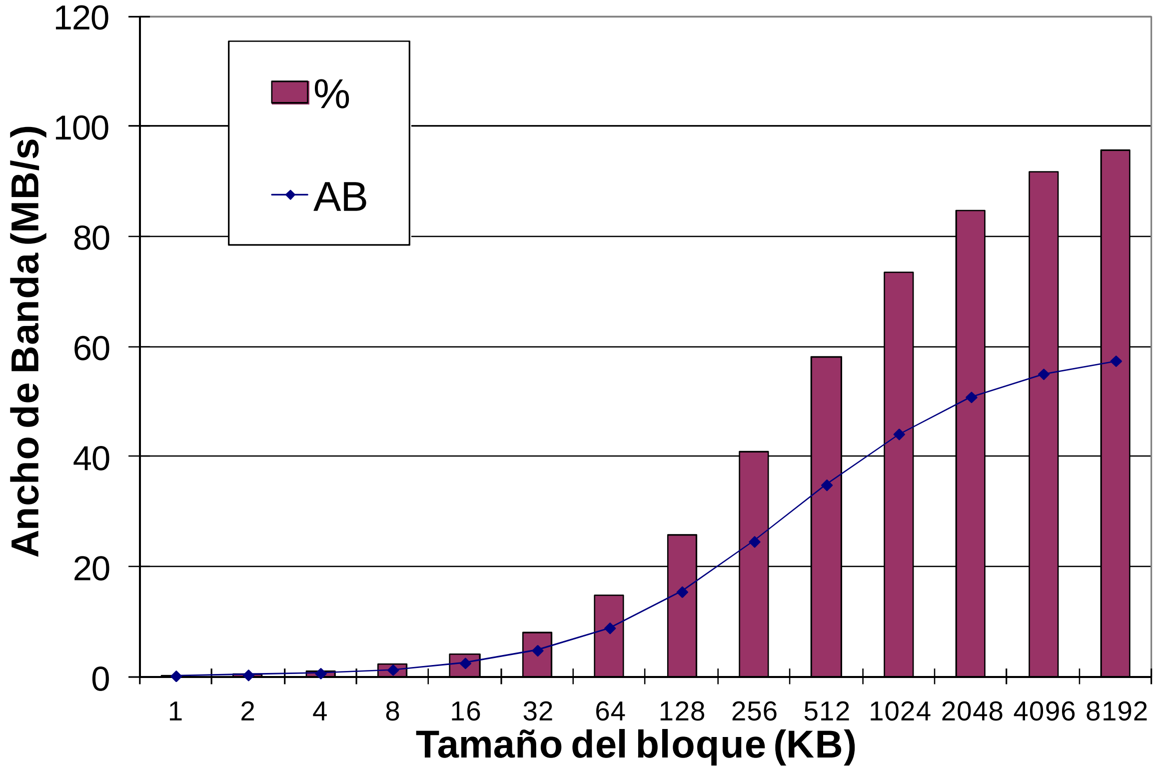
- Two elements:
  - External data rate to measure the transfer rate between memory and disk cache
  - Transfer rate between disk cache and disk storage media
- Data transfer time can be calculated as:

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

- 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.

Effect of the request size

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



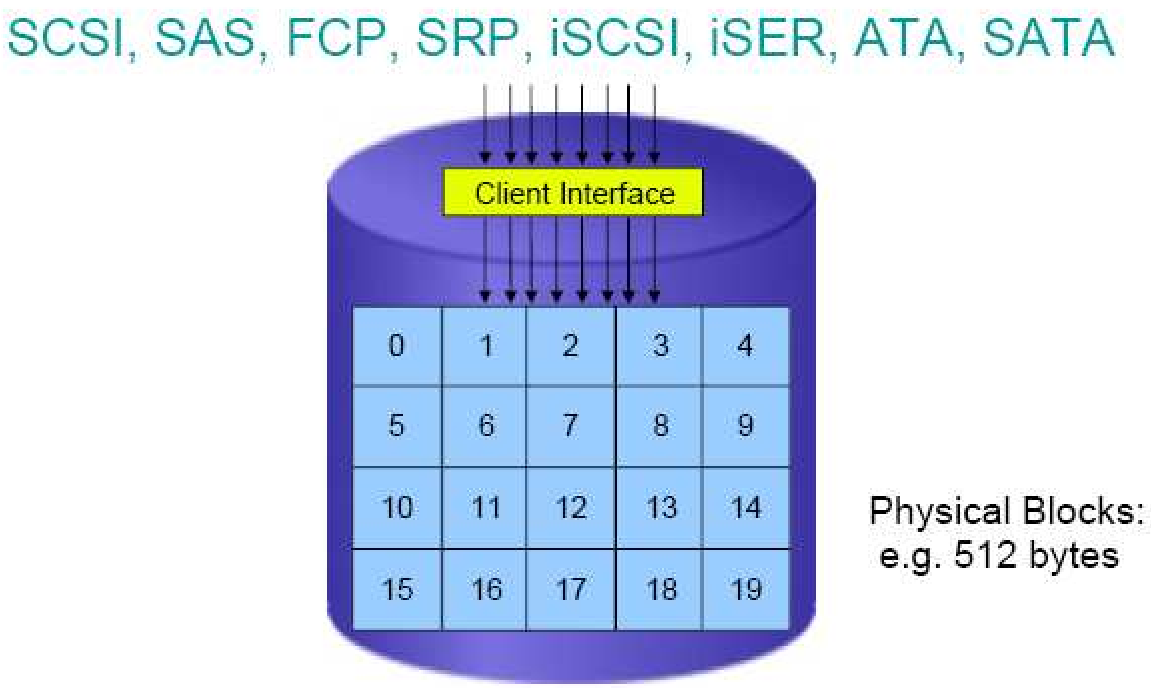
Disk controller

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

Storage interface

- Offers a standard protocol for the disk drives to communicate with its client.
  - IDE, SCSI, FC, SATA
- Exposes storage capacity as a linear array of fixed-size blocks to file systems
- Data access is specified by a LBN and a data block length
  - The disk controller is responsible for translating the LBN to physical addresses (Cylinder/Head/Sector)
- There are other types of devices: *Object-based Storage Devices* (OSD)

Disk access


SCSI, SAS, FCP, SRP, iSCSI, iSER, ATA, SATA
Client Interface
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
Physical Blocks:
e.g. 512 bytes

Disk cache

- Exploit the locality
- Reduce physical access to disk
  - Reduce the heat dissipation
  - Increase the performance
- Disk cache have almost no temporal locality because the host memory (OS) is larger than the disk cache
- Implements perfectingg

Disk cache

- Replacement algorithms
  - Random Replacement
  - Least Frequently Used (LFU)
  - Least Reccently Used (LRU)

- Systems designers generally believe that the size of a cache should be at least 0.1 to 0.3 % of the backing store.

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

- 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 appending ECC symbols 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 disk behavior
- Buffer controller: provides arbitration and raw signal control to the bank of buffer memory

Exercise

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

Remainder

Name	Abr	Factor	IS
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

- 1 KB = 1024 bytes, but in IS is 1000 bytes
- Manufactures of disk drives and telecomunications use IS:
  - A disk drive of 30 GB stores 30 x 10^{9} bytes
  - A network of 1 Mbit/s transfers 10^{6} bps.

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

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

- A disk drive has an average accesss time of 4 ms, 15000 rpm and 500 sectors per track. Let be a file of 2500 sectors with a size of 1.22 MB. Compute the time needed to read the file in the following scenarios:
  - The file is stored in sequential way, and occupies 5 consecutive tracks.
  - The sectors of the file are randomly stored in the disk

Reliability

- MTTF: mean time to failure
- MTTR: mean time to restore
- Reliability is defined as:

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

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

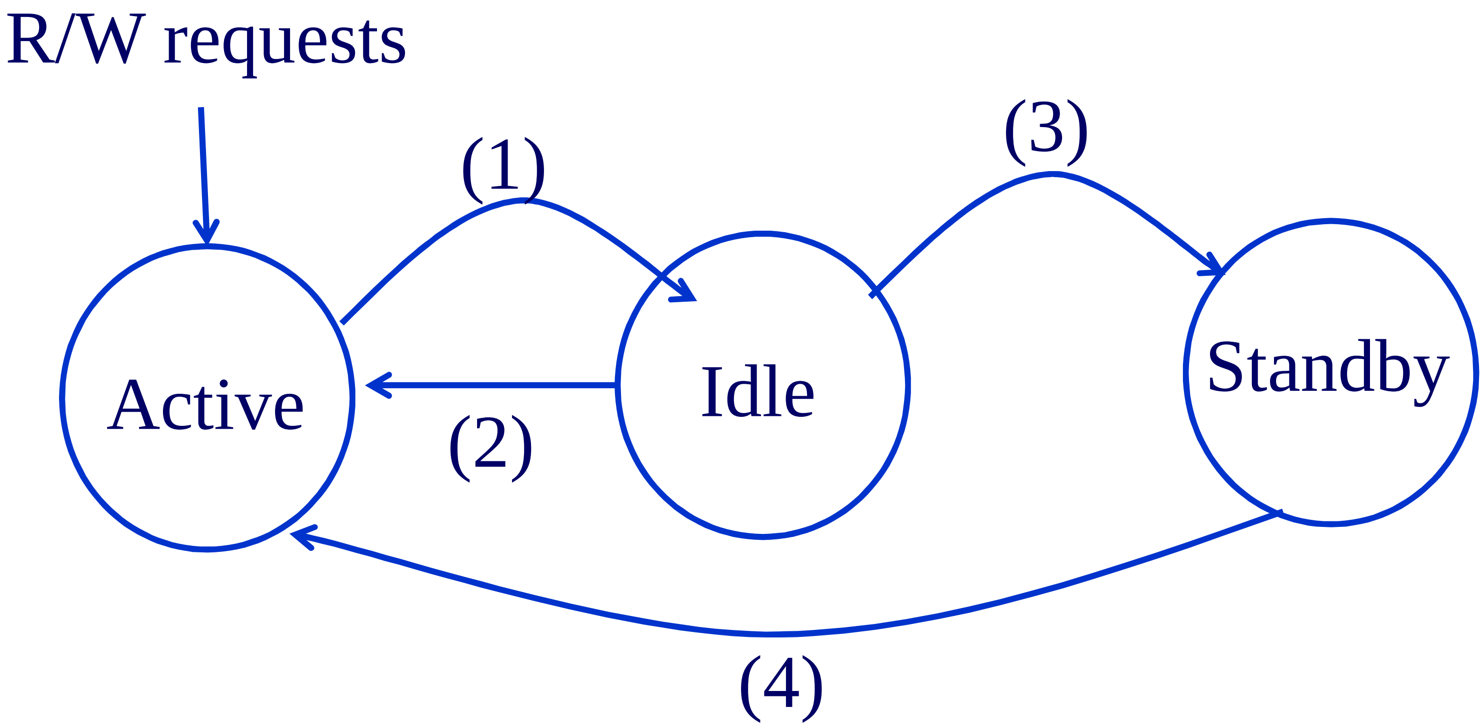
Energy consumption

- 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 succeeding 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 can be spun down 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 spinningg 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

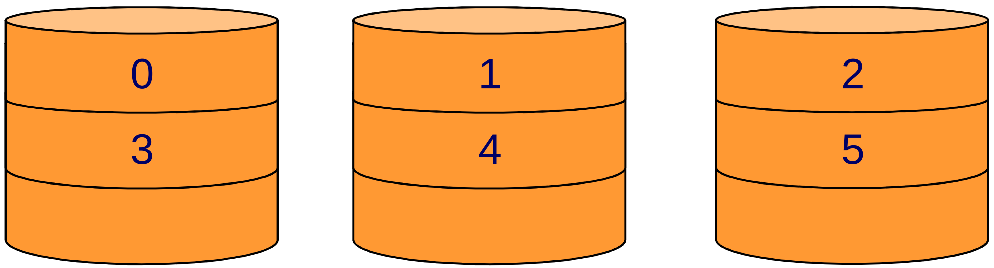
- The power state transition can incur a significant energy cost and time penalty as the disk patters have to spun up to full speed.
- Reduce the reliability. Disk drives manufacturers provide a duty-cycle rating which is the number of times the disk platters can be spun down before the chances of failures increase to more than 40% on drive spin up.
- The methods cannot be applied toe server disk drives, since the spinning down and spinning up time of the server disk drives are much longer than that of the desktop and laptop

RAID disk

- *Redundant Array of Inexpensive Disks*
- Set of disk drives that works as an unit
- Distributed data across several disks
- Store redundant information

Parallelism in disks

- *Disk stripping*
  - Data are divided in blocks and each block is written in a different disk
- RAID 0
- Increase the performance
  - Distributes the I/O load through several disks and channels



Advantages of parallelism in disks

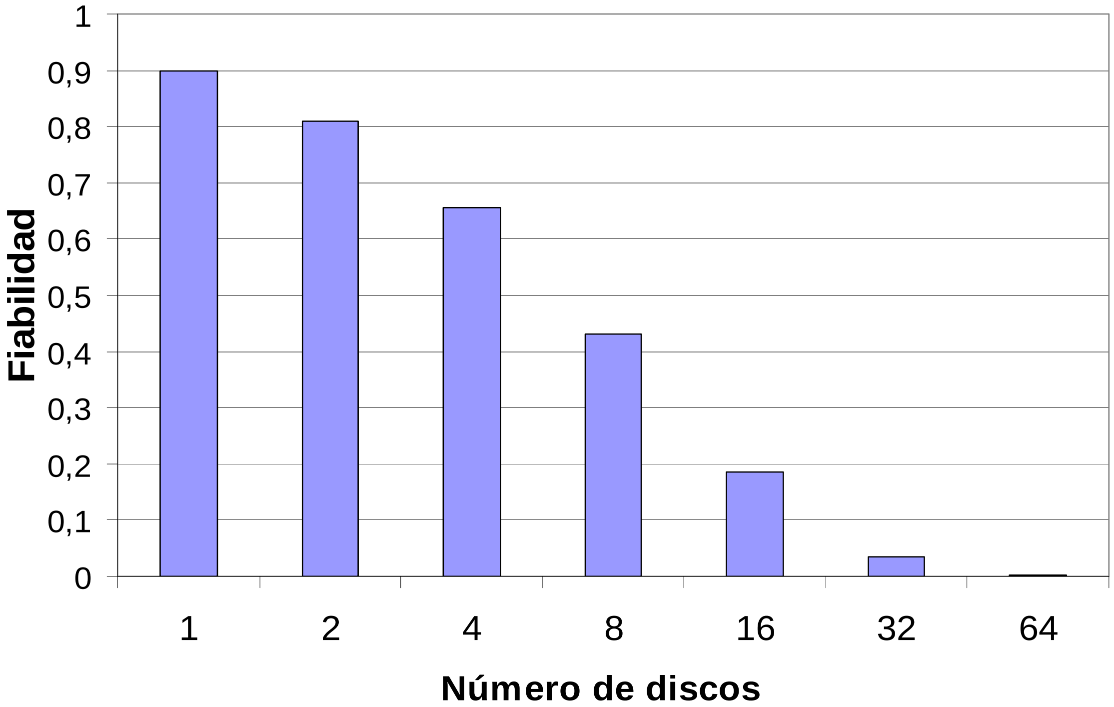
- **Large requests**
  - Parallelism inside the request
  - The request can be satisfied by several disks
  - Reduce the access time
- **Short requests**
  - Parallelism among requests
  - Several requests can be made at the same time
  - Seeks in parallel
  - Transfers in parallel

Problem: Reliability

Serial system

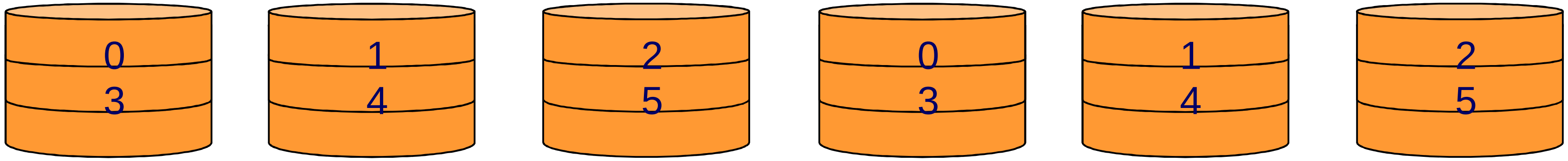
Reliaibity

$$R = \prod R_i$$



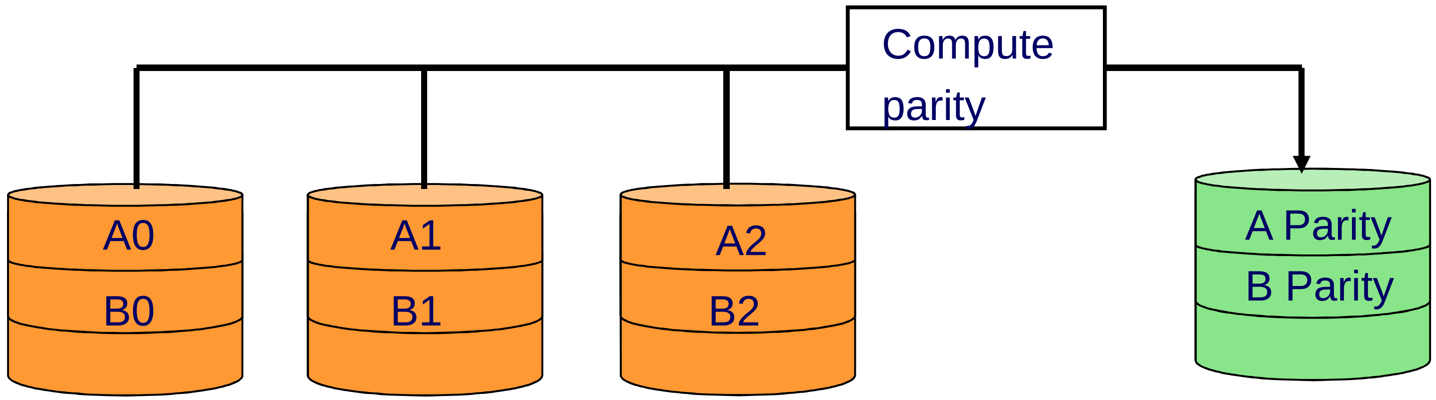
Any failure is supoorted

RAID 1



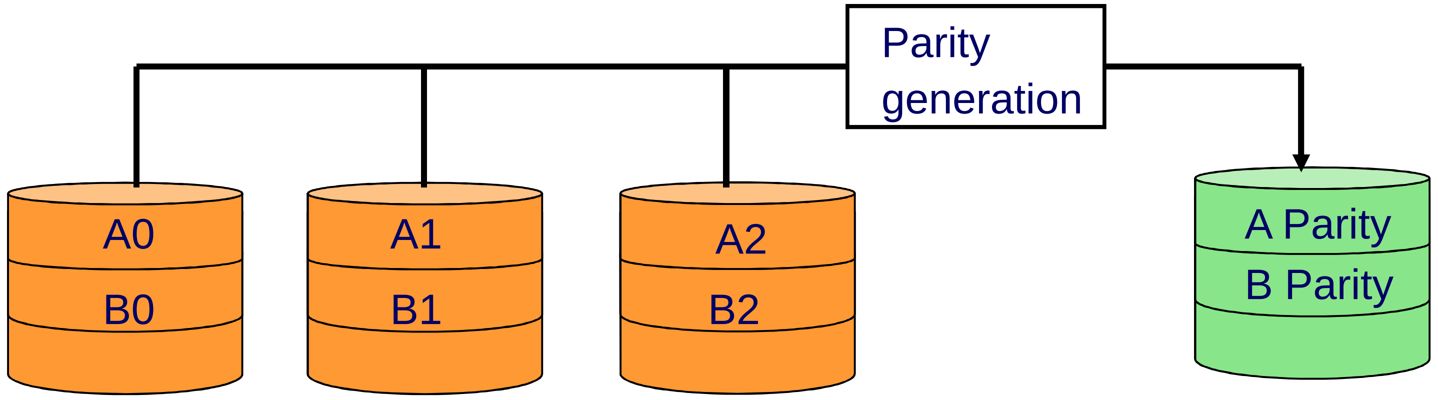
- 2 replicated RAID0
- One read operation and two write operations
  - A write operation on each block
  - A read operation using one block
- Redundancy: 100%
- Parallelism limited
- Easy recovery

RAID 3



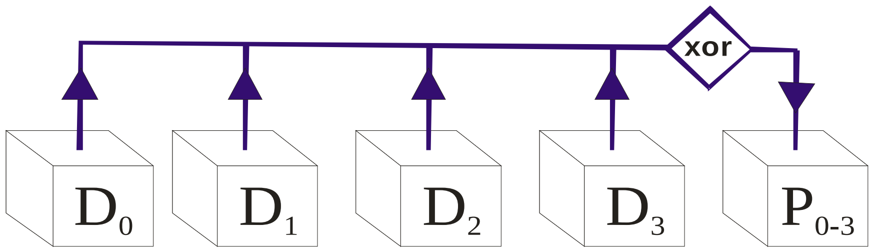
- A data block is divided in bytes and theses bytes are written in all disks
- Parity computed in write operations
- Reduce the transfer time for reads and writes
  - Appropriate for large requests

RAID 4

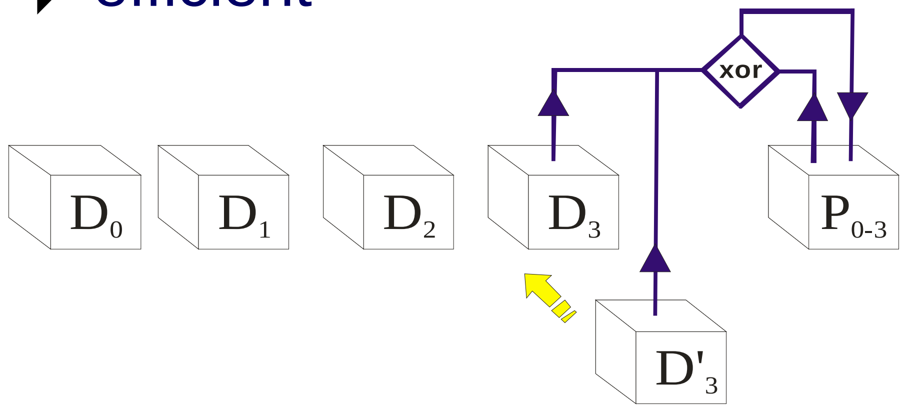


- Each block is stored in a different disk
- Transfer time for reads similar to use only one disk
- No parallelism in write operations
  - All write operations use the parity disk
- Concurrent read operations

Parity generation

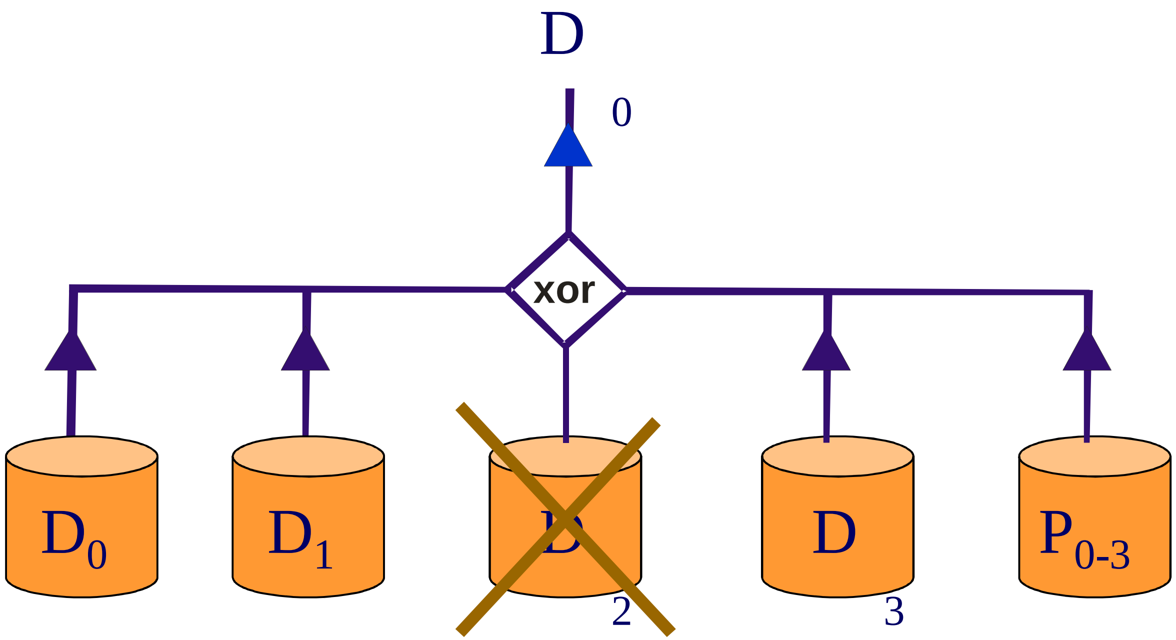


Whole stripe ⇨ efficient

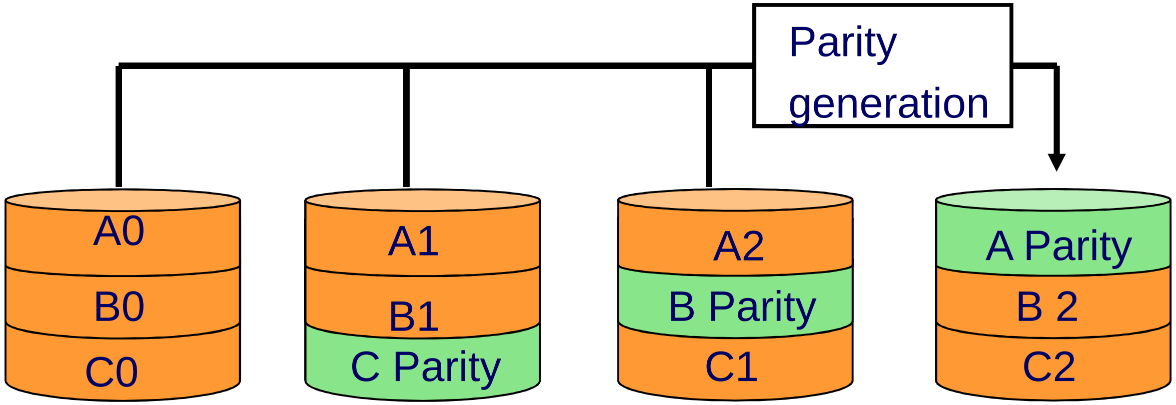


Short write operations ⇨ performance problems

Data rebuilding



RAID 5



- Parity blocks are distributed among all disk drives
- Reducing the contention over the parity disk
- Concurrent reads and writes operations
  - Appropriate for high I/O throughput

Reliability calculation

- If P is the probability of failure, the reliability R is:

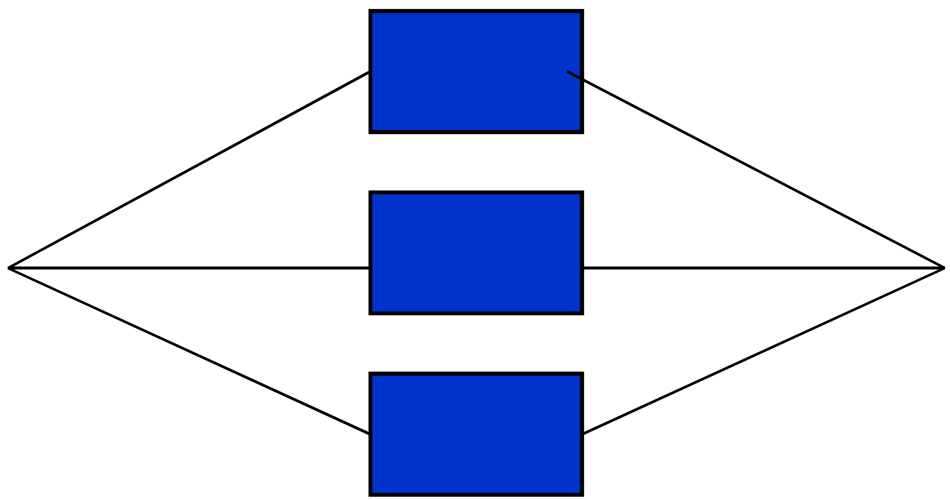
$$R = 1-P$$

- Reliability of a serial system
- Reliability of a parallel system
- Reliability of a *k-out-of-n system*

Reliability of a serial system

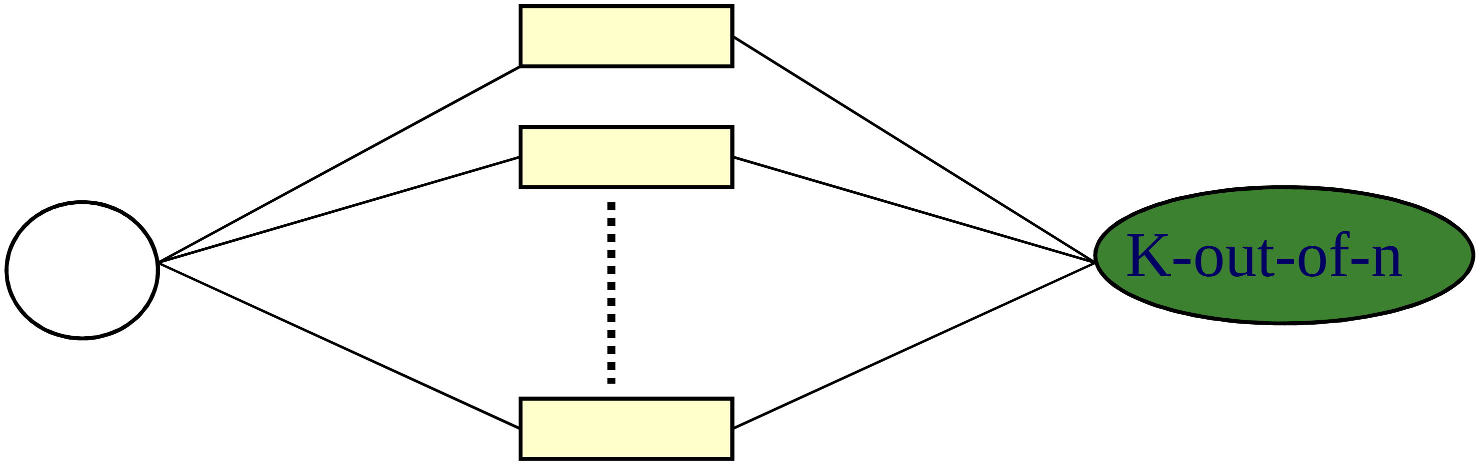
$$R = \prod R_i$$

Reliability of a parallel system



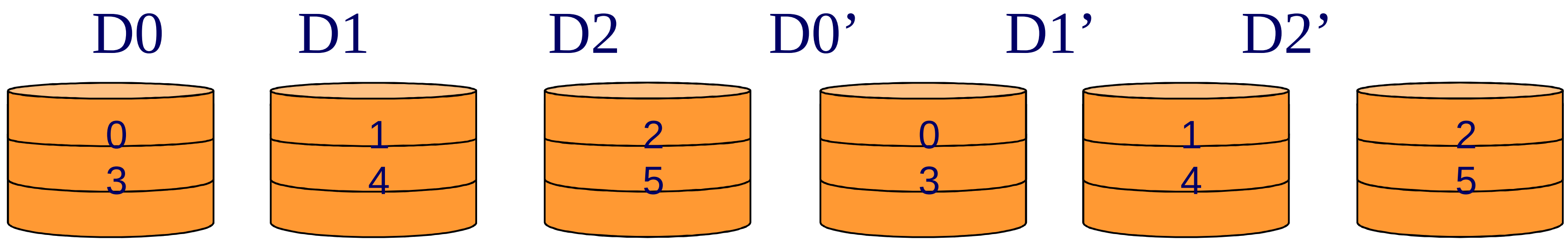
$$R = 1 - \prod (1 - R_i)$$

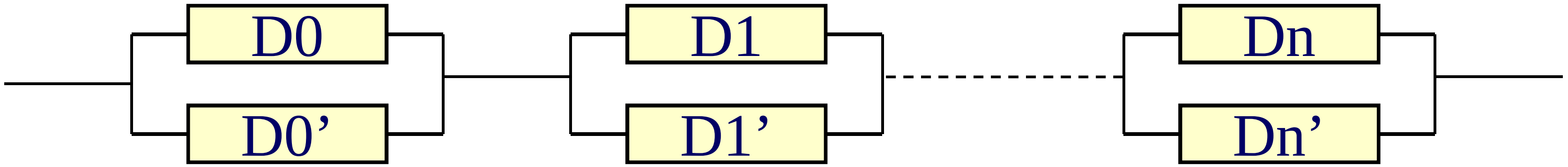
Reliability of a *k-out-of-n system*



$$R(k,n)=\sum_{r=k}^{r=n}\binom{n}{r}R^{r}(1-R)^{n-r}$$

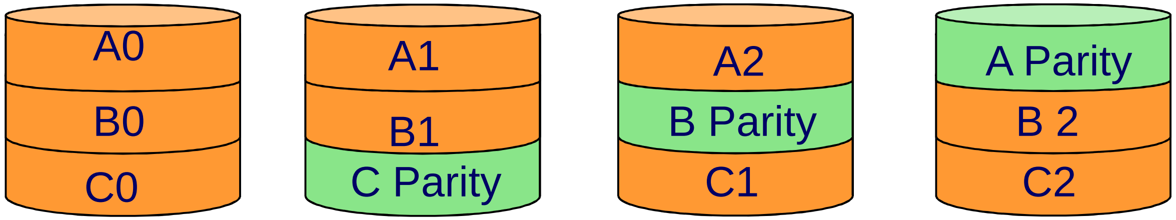
Reliability of a RAID1





$$R = \prod_{i=1}^{i=\frac{n}{2}} (1-(1-R)(1-R)) = (2R - R^2)^{\frac{n}{2}}$$

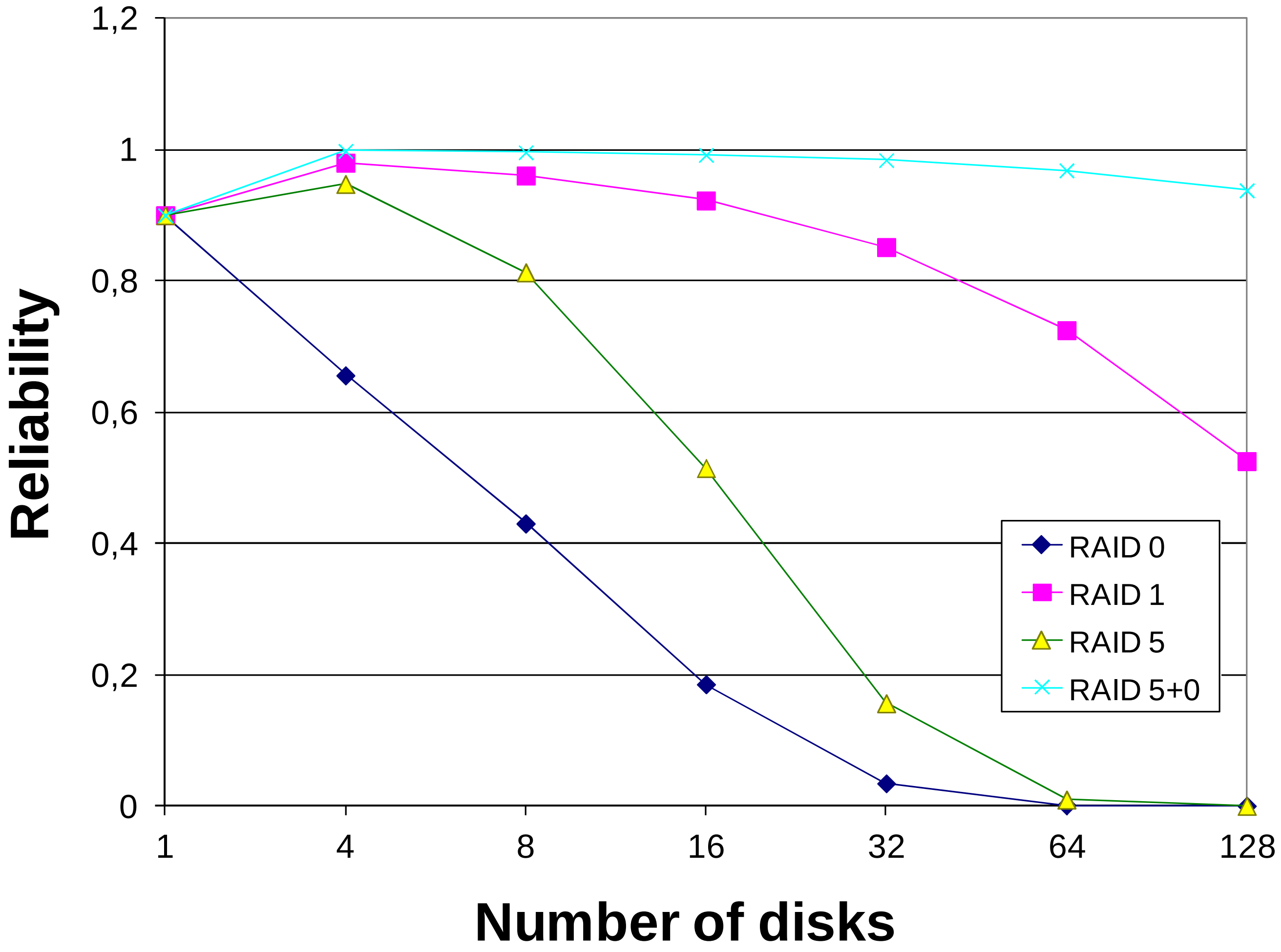
Reliability of a RAID5



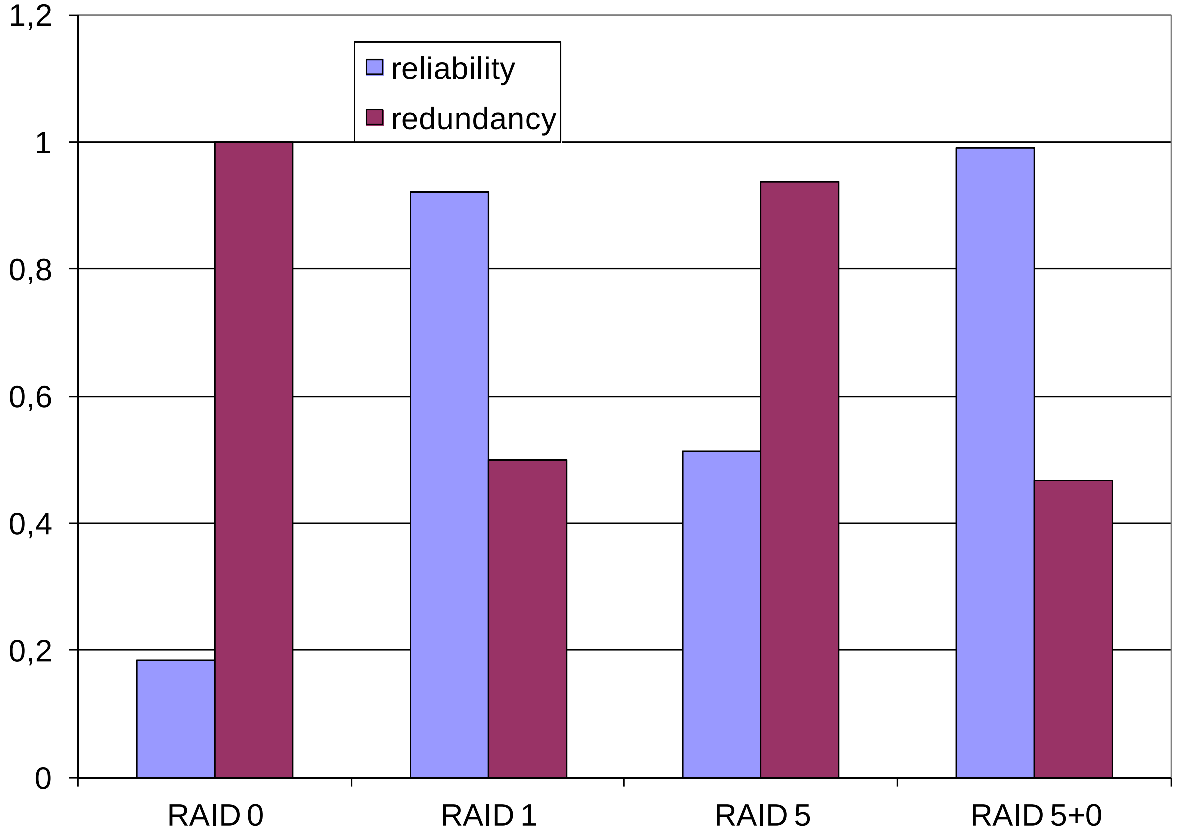
- For n disks is a *(n-1)-out-of-n* model

$$R(n-1,n) = \sum_{r=n-1}^{r=n} \binom{n}{r} R^r (1-R)^{n-r} = nR^{n-1} + R^n (1-n)$$

Reliability



Reliability/redundancy



16 disks

Solid state disks

- Storage devices that use semiconductors to data storage
  - Based on flash memories
    - Non volatile memories
  - Based on DDR memories
    - Require batteries and disk backup to provide non volatile

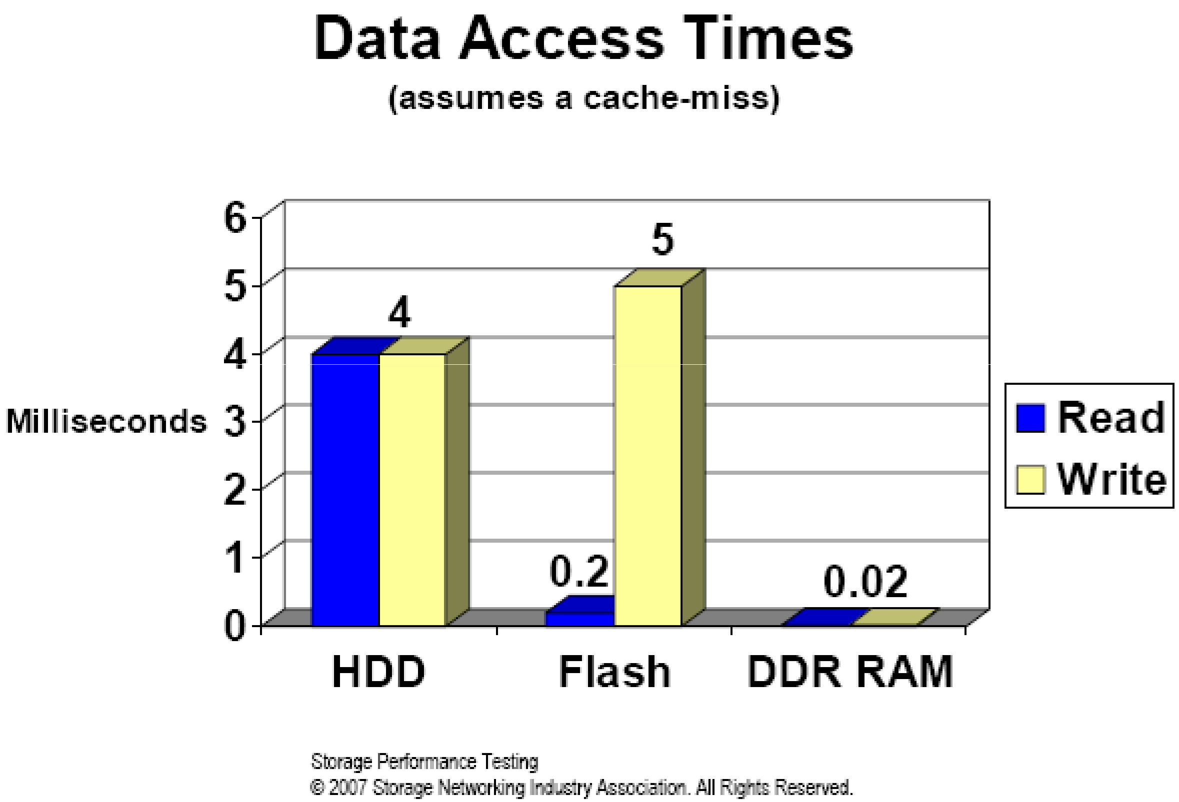
Flash memory

- Non volatile semiconductor memory
- types:
  - NOR flash
    - Based on NOR gates
    - Byte accessible
    - Faster than NAND flash
  - NAND flash
    - Based on NAND gates
    - Less expensive and much more popular than NOR flash
    - Accessed using blocks

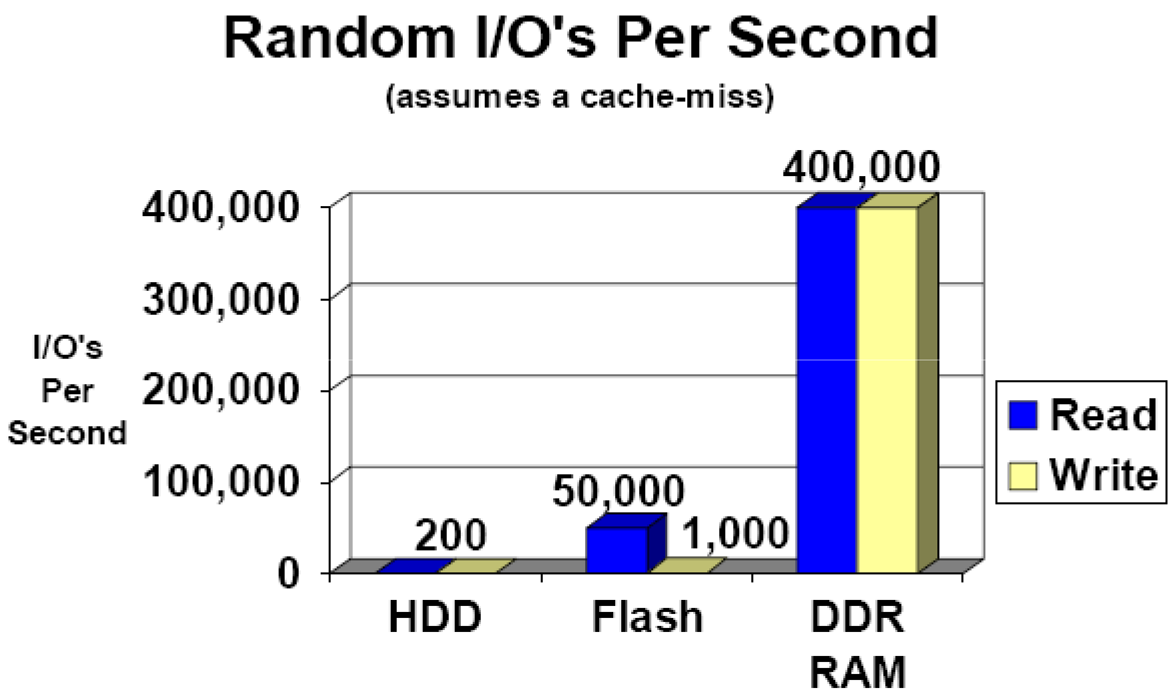
Write operation in Flash

- A NAND flash is composed of a fixed number of blocks, where each block consists of a number of pages, and each page has a fixed-size main data area
- Data on a NAND flash memory is read or written in a unit of one page, and the erasing is performed in a unit of one block
- A block is erased put “1” in all bits
- The write operation only can write “0”

Access time

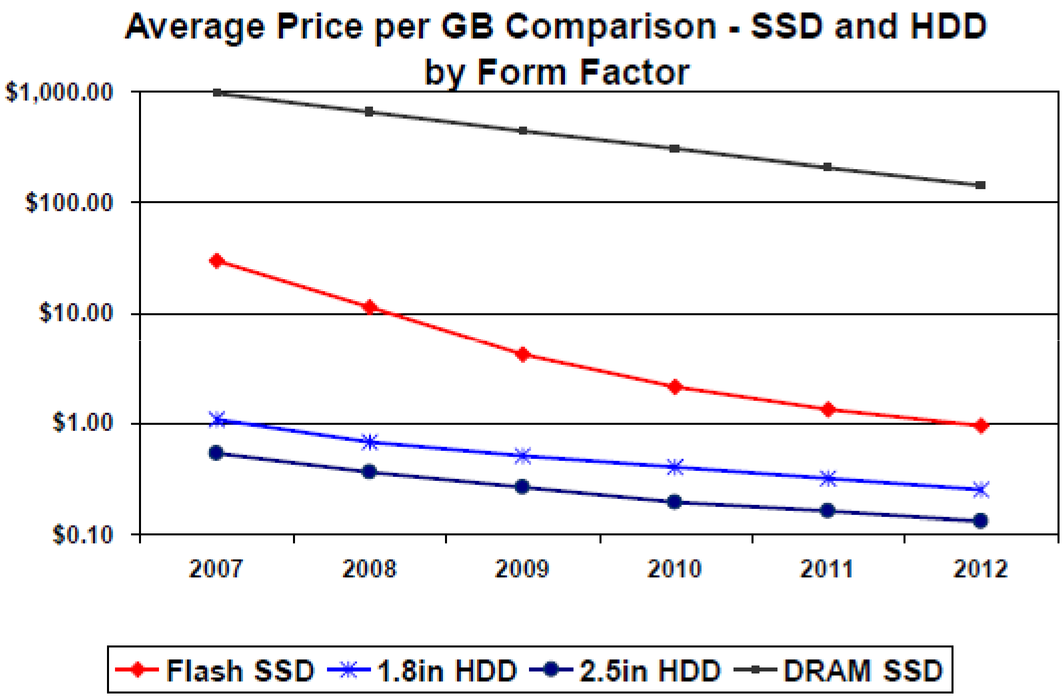

Data Access Times
(assumes a cache-miss)
Milliseconds
6
5
4
3
2
1
0
4
5
0.2
0.02
HDD
Flash
DDR RAM
Read
Write
Storage Performance Testing
© 2007 Storage Networking Industry Association. All Rights Reserved.

I/O operations



Storage Performance Testing

Prices


Average Price per GB Comparison - SSD and HDD
by Form Factor
$1,000.00
$100.00
$10.00
$1.00
$0.10
2007
2008
2009
2010
2011
2012
Flash SSD
1.8in HDD
2.5in HDD
DRAM SSD

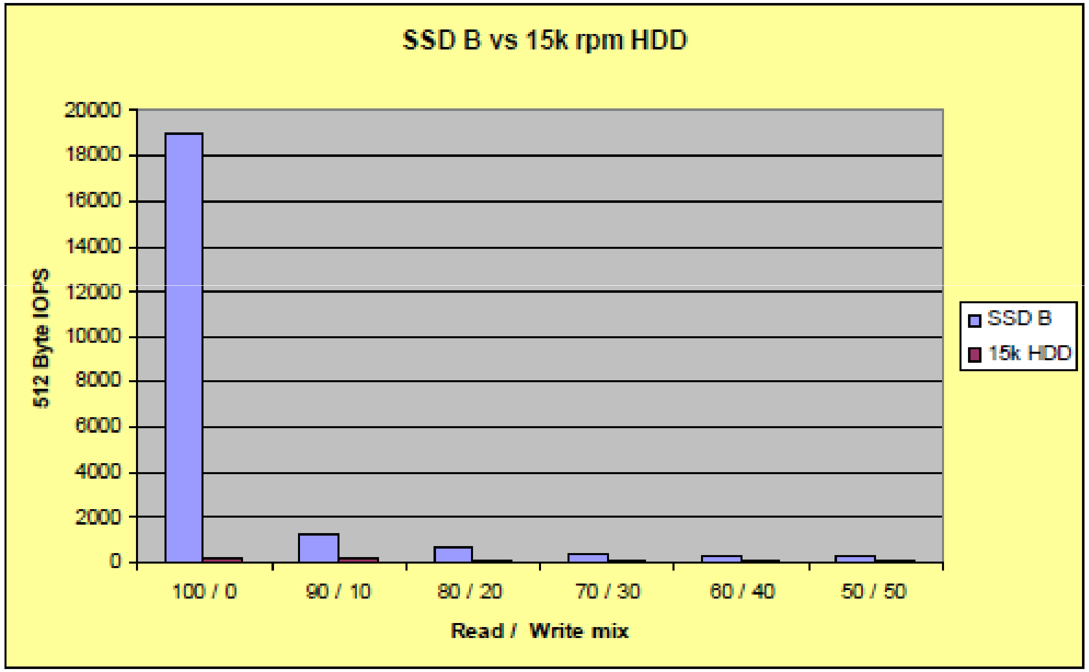
Performance

SSD Random 512 byte IOPs Performance		
	Read	Write
SSD A	45000	16000
SSD B	19000	130
SSD C	7000	15
SSD D	6300	926
15K rpm HDD*	185	170
7.2K rpm HDD*	79	73
5.4K rpm HDD*	60	57

* calculated from data sheet seek time

Performance

- Mixtures loads

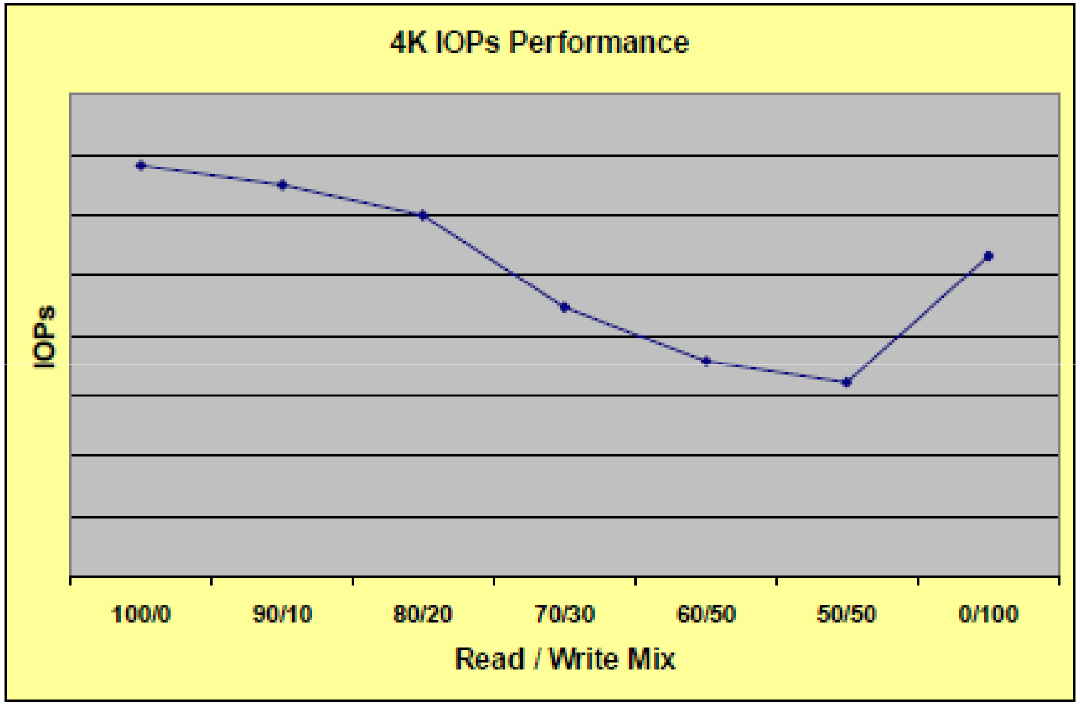


Solid State Storage in the Enterprise

Performance

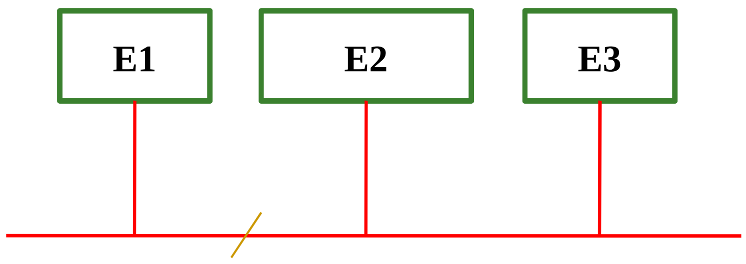
SSD Sequential Performance MB/sec		
	Read	Write
SSD A	220	115
SSD B	130	120
SSD C	57	38
SSD D	100	80
15K rpm HDD	171	171
7.2K rpm HDD	105	105
5.4K rpm HDD	61	61

Performance



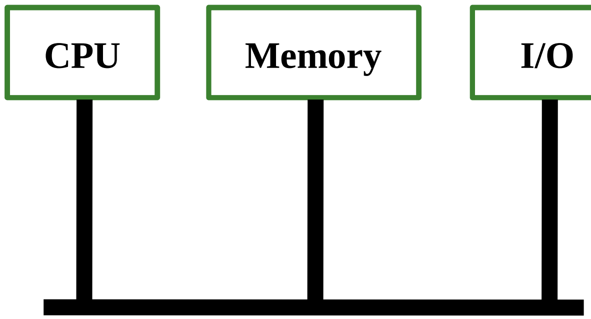
Solid State Storage in the Enterprise

Bus



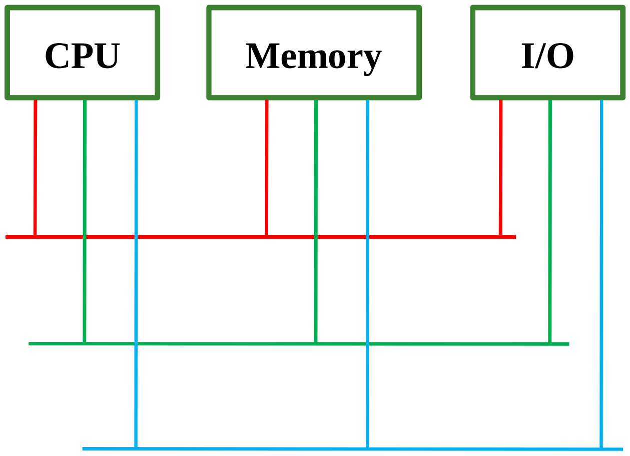
- A bus allows the communication among two or more devices
- Uses several lines to transmit bits
- The bus is shared
- Types:
  - Serial
  - Parallel

System bus



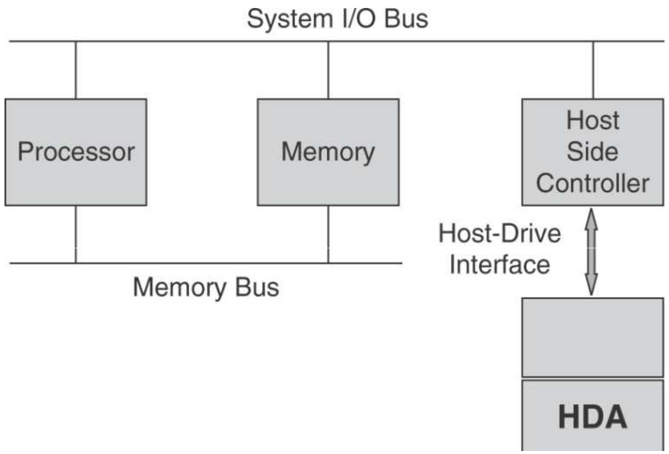
- **System bus**
  - Connects the main components in the computer
  - Is the union of three buses:
    - Control
    - Address
    - Data

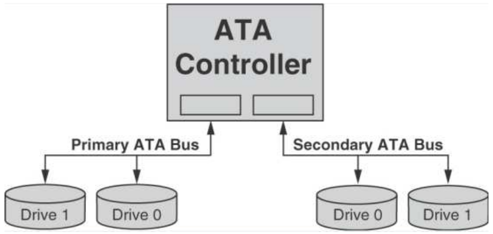
Buses

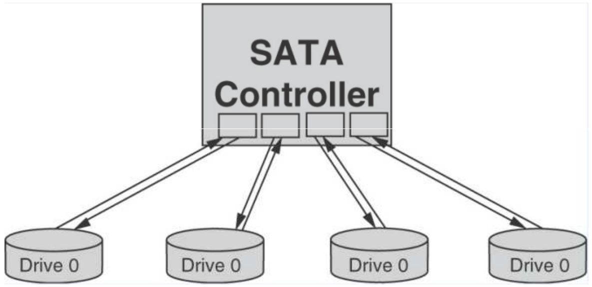


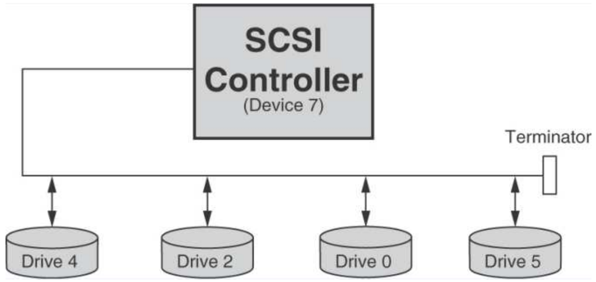
- Data bus
- Address bus
  - Memory and I/O addresses
- Bus de control
  - Control signals and temporization

Disk controllers



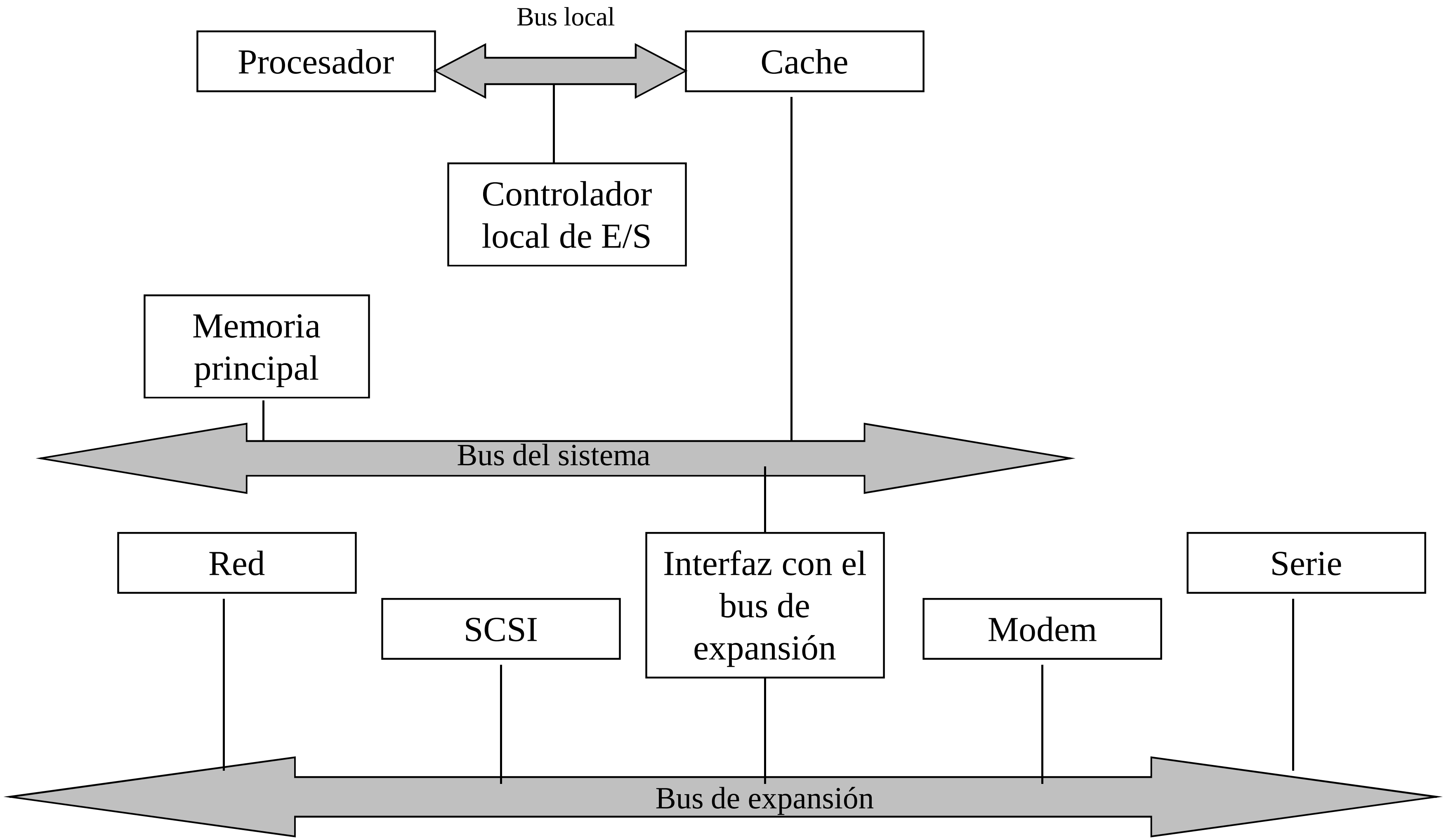




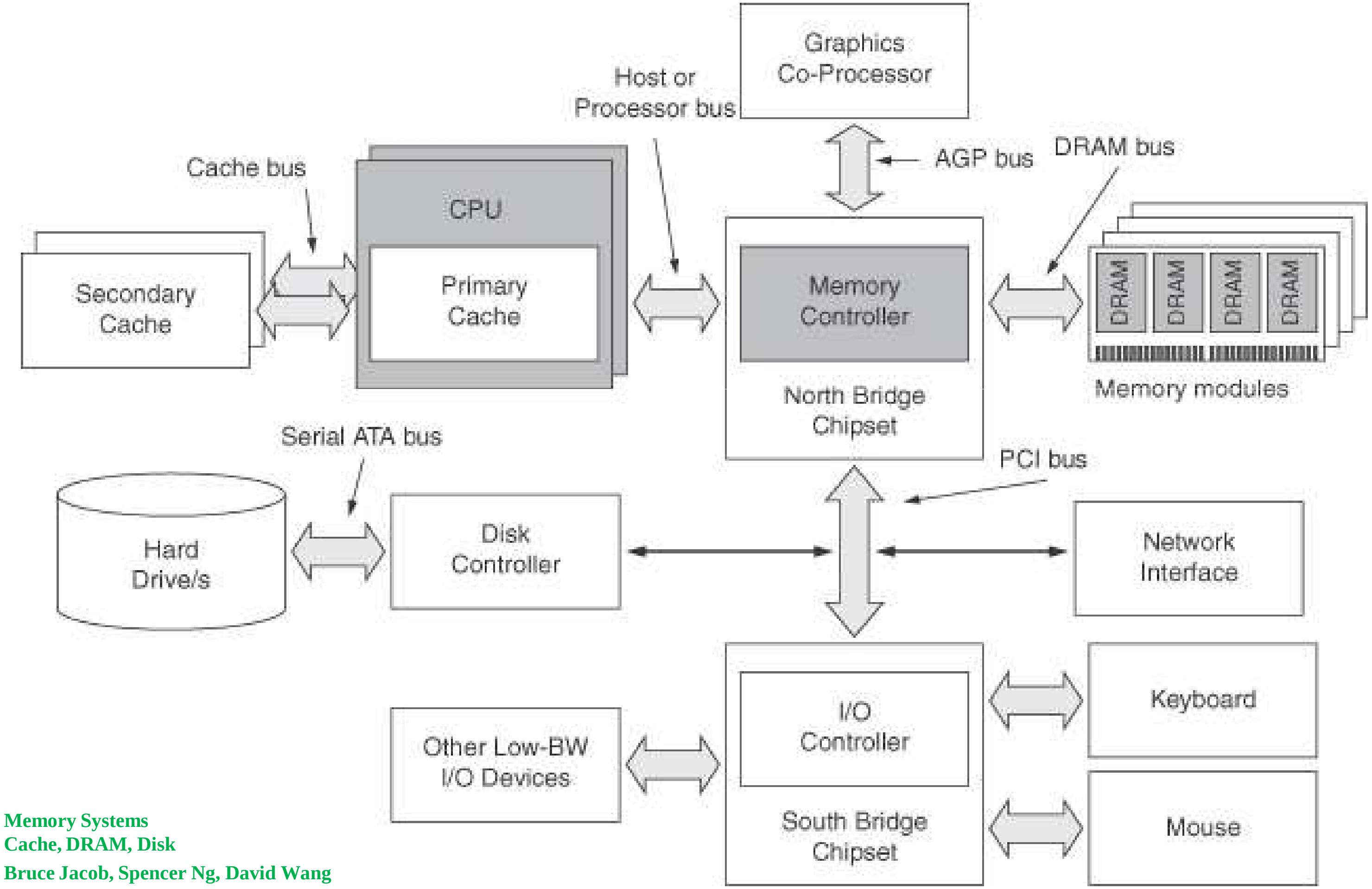


Memory Systems
Cache, DRAM, Disk
Bruce Jacob, Spencer Ng, David Wang
Elsevier

Bus hierarchy



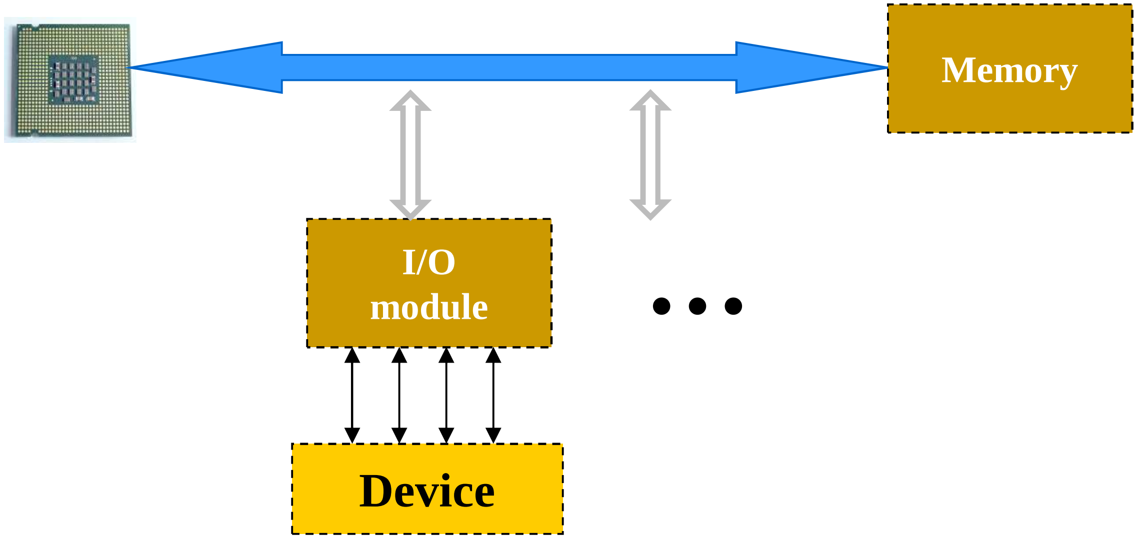
Example of configuration



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

I/O module

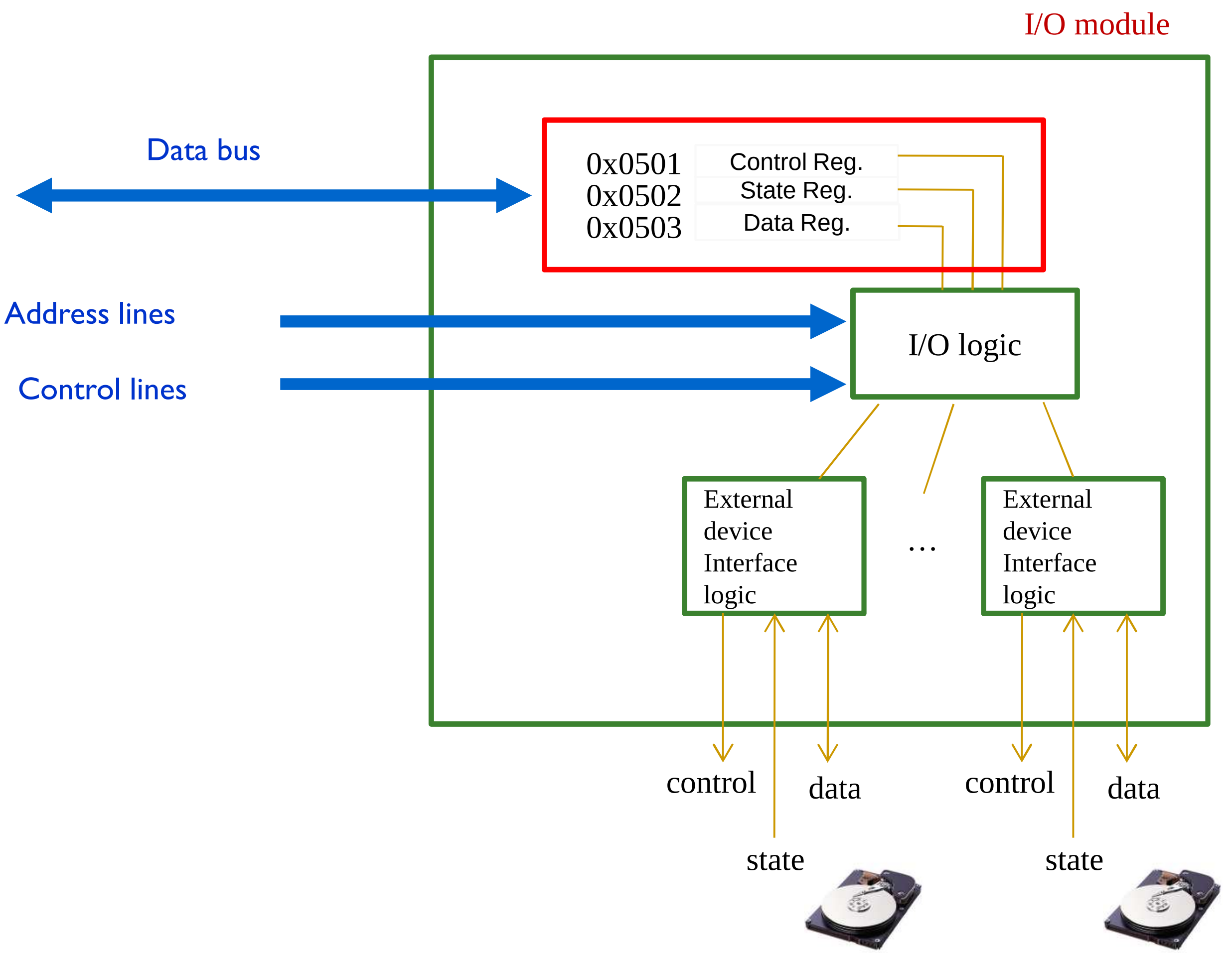
- I/O modules perform the connection among the peripheral devices and the processor or the memory



Functions of I/O modules

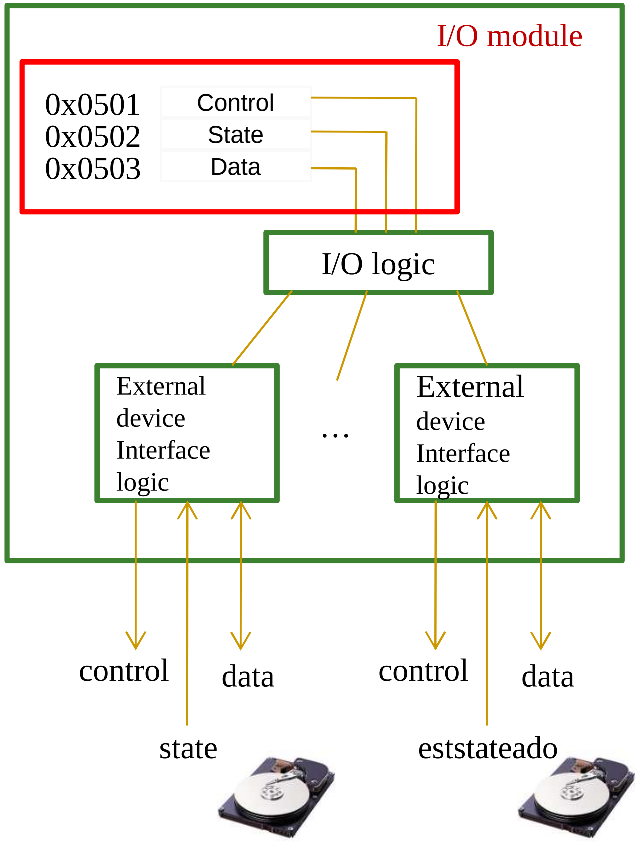
- Control and timing
- Processor communication
- Device communication
- Data buffering
- Error detection

Block diagram of an I/O module



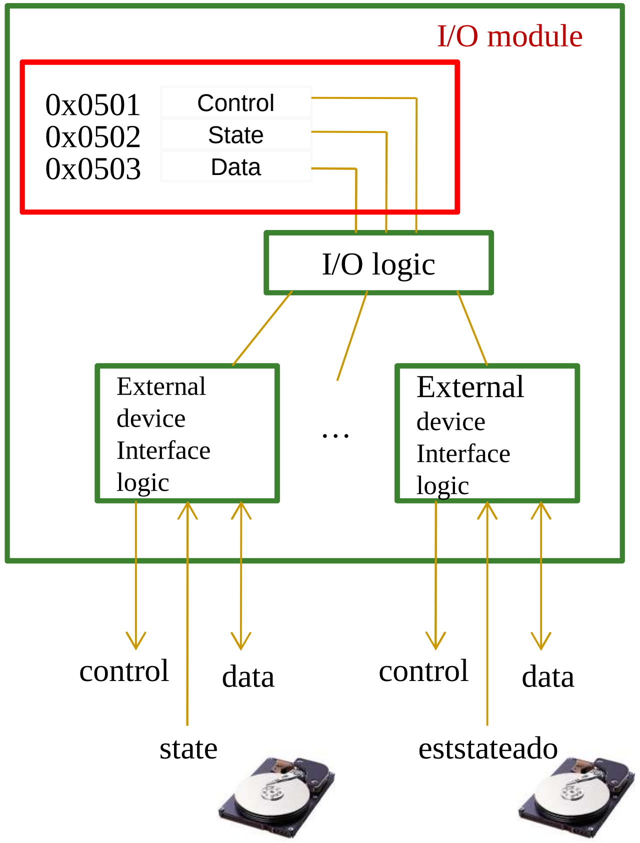
Block diagram of an I/O module

- Interaction between the processor and the I/O module:
  - ❑ Control register
    - Commands for the device
  - ❑ State register
    - Information about the state of the device
  - ❑ Data register
    - Data exchanged CPU/Device



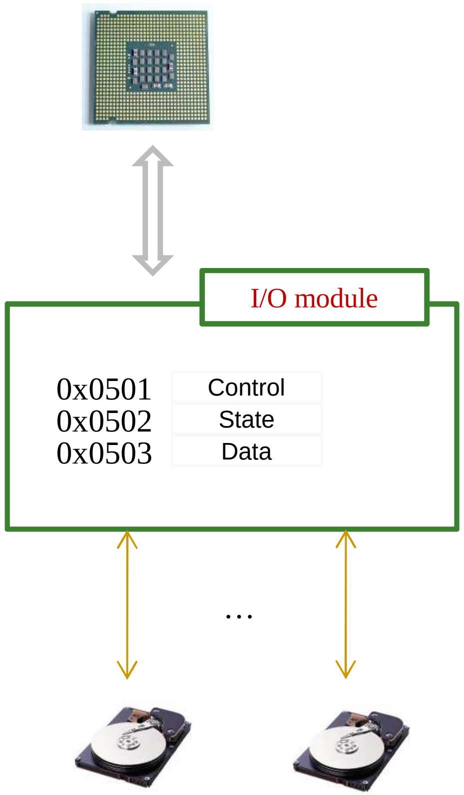
Block diagram of an I/O module

- Data lines: for transferring information
- State lines: information about the device
  - New data available
  - Device on/off
  - Device busy
  - Device working or not
  - Error
  - …
- Control lines: to control the device
  - On/off
  - Read/write
  - Seek operation in a disk drive



I/O module

- Main features:
  - Transfer unit
  - Addressing
  - Interaction computer-controller



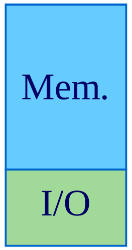
Transfer unit

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

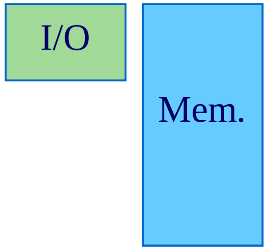
- **Character devices**:
  - Unit: chars (ASCII, Unicode, etc.)
  - Access: sequential
  - Operations : `get`, `put`, ….
  - Example: terminals, printers, etc.

I/O addressing

- **Memory mapped I/O**
  - Address space for I/O and memory is shared. I/O registers are mapped in memory using a set of memory addresses for these registers. Use the same machine instructions that the used for memory
  - Ej: sw $a0 etiqueta_discoA



- **Isolated I/O**
  - The address space for I/O is isolated from the address space used for the memory. It uses special machine instructions to access the I/O registers
  - Ej: out $a0 0x105A



Addressing

Linux

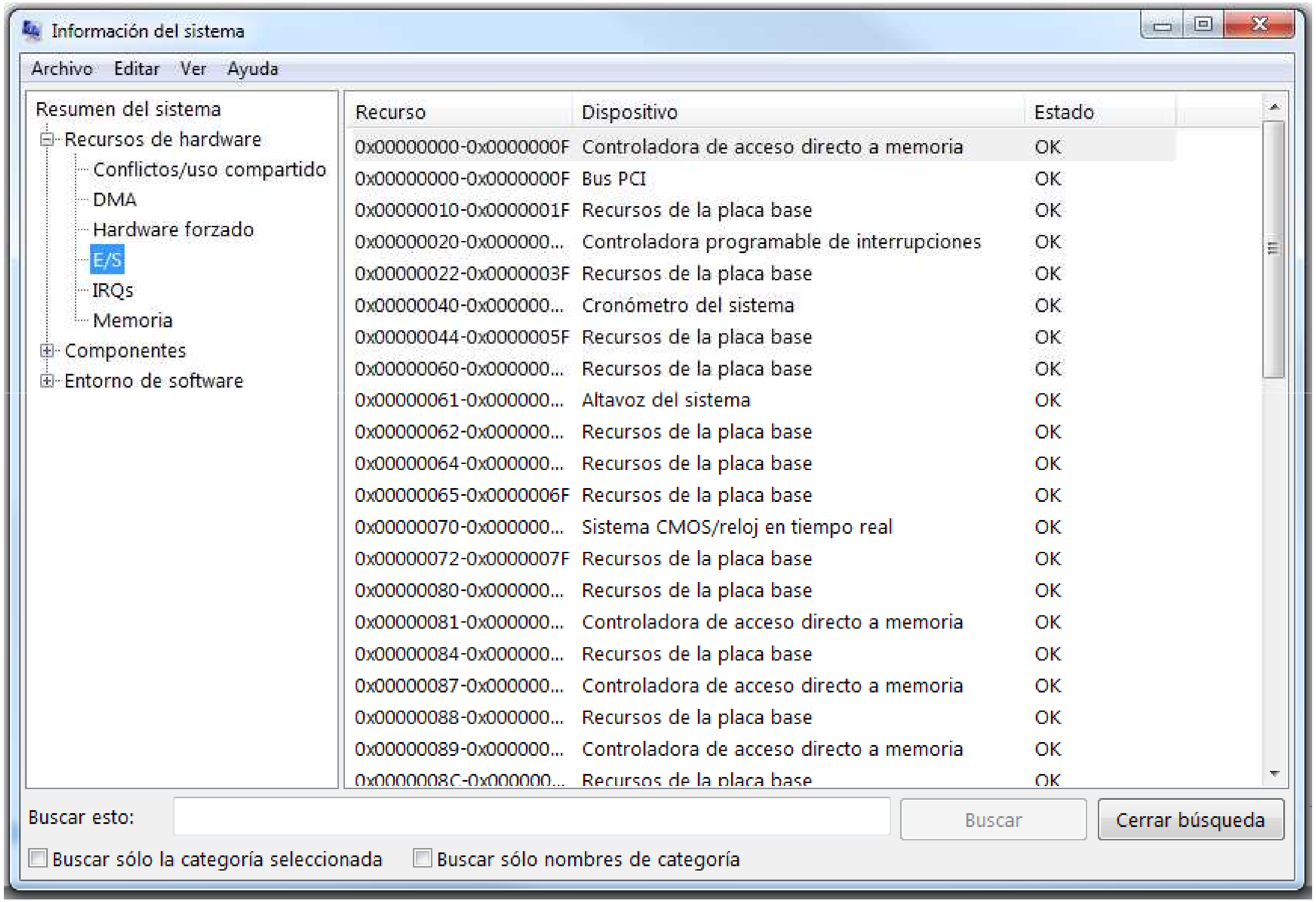
phoenix.arcos.inf.uc3m.es - default* - SSH Secure Shell

```
acaldero@phoenix:~$ 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
  0070-0073 : rtc0
  0080-008f : dma page reg
  00a0-00a1 : pic2
  00c0-00df : dma2
  00f0-00ff : fpu
  0290-029f : pnp 00:01
    0290-0294 : pnp 00:01
  02f8-02ff : serial
  0378-037a : parport0
  03c0-03df : vga+
  03f2-03f2 : floppy
  03f4-03f5 : floppy
  03f7-03f7 : floppy
  03f8-03ff : serial
  0400-047f : 0000:00:1f.0
    0400-0403 : ACPI PM1a_EVT_BLK
    0404-0405 : ACPI PM1a_CNT_BLK
```

Connected to phoenix.arcos.inf.uc3m.es

Addressing

Windows



I/O techniques

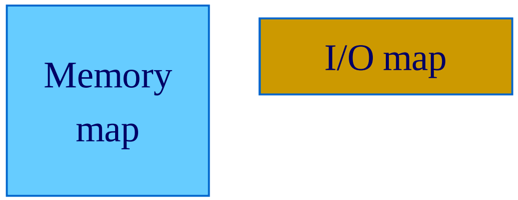
- Programmed I/O
- Interrupts
- DMA, *Direct memory access*

Programmed I/O

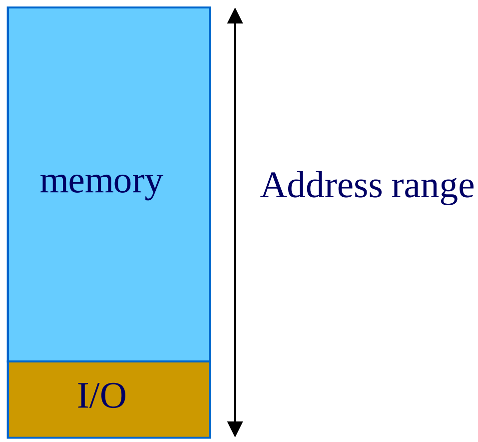
- The transfers between the processor (memory) and the I/O module is controlled by the processor that executes I/O machine instructions
- I/O instructions:
  - Special instructions (similar to lw and sw)
  - Privileged instructions
- Example of hypothetical I/O instructions
  - IN Reg, address
    - Load in the processor register Reg the item stored in the I/O register with a given address
  - OUT Reg, address
    - To write an item in an I/O register

I/O map

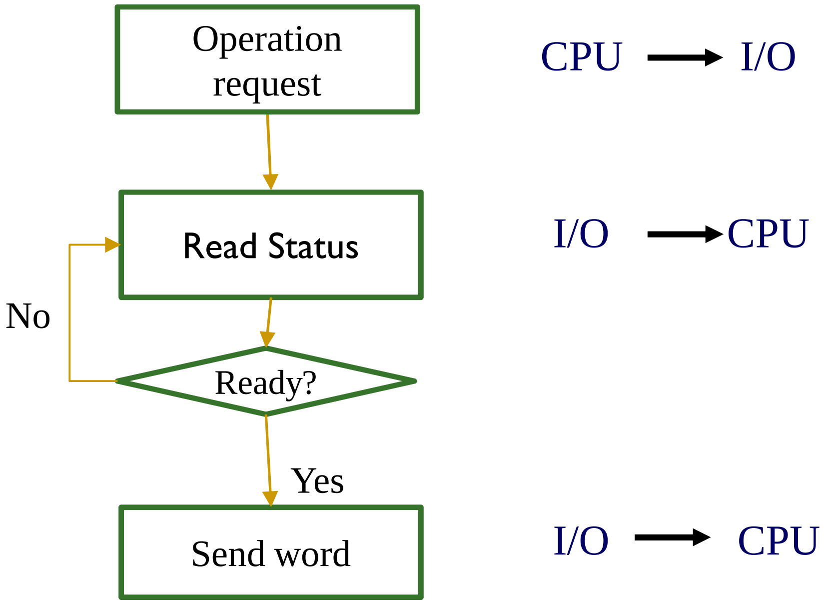
- Space address for I/O
  - With p bits, 2^p possible addresses
- types:
  - Isolated I/O map
    - Include I/O instructions



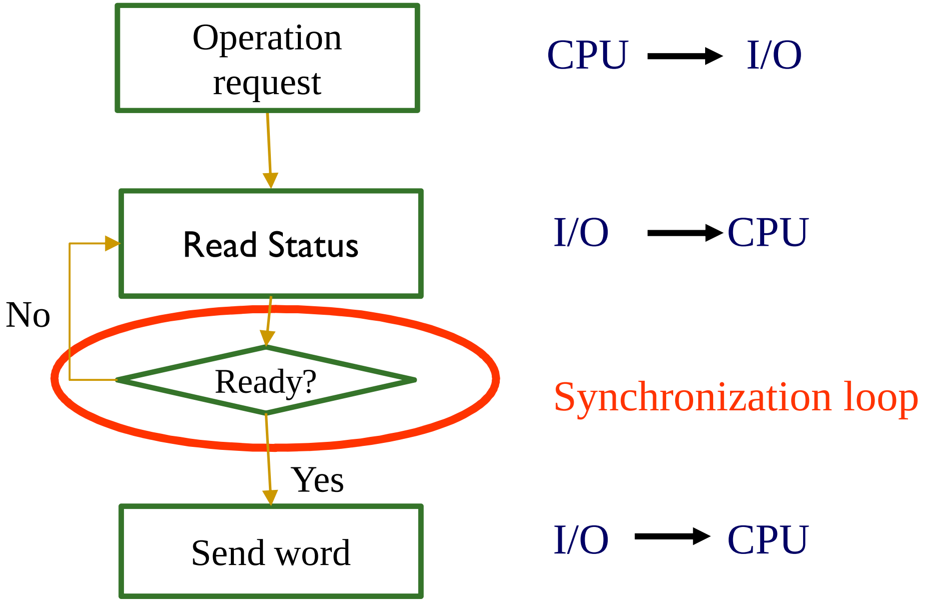
  - Shared map
    - Same machine instructions for I/O and memory



Programmed I/O



Programmed I/O



Example

I/O module

	address
State Reg.	1000
Data Rreg.	1004
Control Reg.	1008

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

Example

I/O module

	address
State Reg.	1000
Data Rreg.	1004
Control Reg.	1008

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

- Instructions needed to write “1” in the register 1004 (data reg.)?

Example

I/O module

	address
State Reg.	1000
Data Rreg.	1004
Control Reg.	1008

```
li $t0, 1
sw $t0, 1004
```

- Write “1” in register with address 1004 (data)

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

Example

I/O module

	address
State Reg.	1000
Data Rreg.	1004
Control Reg.	1008

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

Operations to read a word

?

Example

I/O module

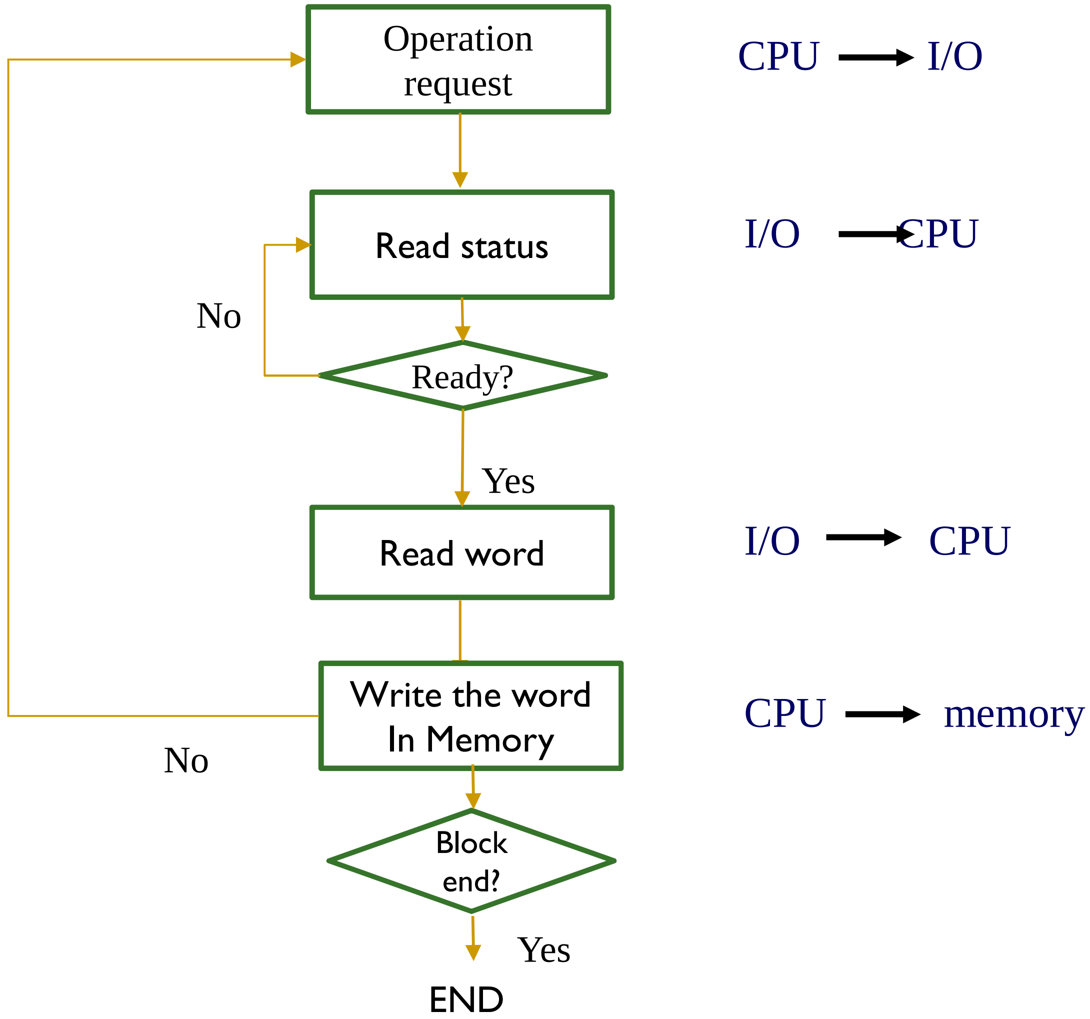
	address
State Reg.	1000
Data Rreg.	1004
Control Reg.	1008

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

Operations to read a word

1. Send the command
```
li $t0, 0
sw $t0, 1008
```
2. Read status
```
bucle: lw $t0, 1000
```
3. Check status
```
beqz $t0, bucle
```
4. Read the word
```
lw $t0, 1004
```

Reading a data block



Example

I/O module

	address
State Reg.	1000
Data Rreg.	1004
Control Reg.	1008

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

Program to read 100 words and sore these data in memory

?

Example

I/O module

	address
State Reg.	1000
Data Rreg.	1004
Control Reg.	1008

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

Program to read 100 word and storing these data in memory

```
.data
    dat:  .space  400
.text
.globl main
main:      lii    $t3,  0
  bucle1:  li     $t0, 0
           sw     $t0, 1008
  bucle2:  lw     $t1, 1000
           beqz $t1, bucle2
           lw     $t2,1004
           sw     $t2 ,dat($t3)
           add i  $t3, $t3, 4
           bne    $t3, 100, bucle1
```

Example

I/O module

	address
State Reg.	1000
Data Rreg.	1004
Control Reg.	1008

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

Program to read 100 word and store these data in memory

```
.data
    dat:  .space  400
.text
.globl main
main:       lii     $t3,  0
  bucle1:   li      $t0, 0
            sw      $t0, 1008
  bucle2:   lw      $t1, 1000      } Synchronization
            beqz $t1, bucle2       } loop
            lw      $t2,1004
            sw      $t2 ,dat($t3)
            add i   $t3, $t3, 4
            bne     $t3, 100, bucle1
```

Example

I/O module

	address
State Reg.	1000
Data Rreg.	1004
Control Reg.	1008

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

Program to read 100 word and store these data in memory

```
.data
    dat:  .space  400
.text
.globl main
main:       lii     $t3,  0
  bucle1:   li      $t0, 0
            sw      $t0, 1008
  bucle2:   lw      $t1, 1000
            beqz $t1, bucle2
            lw      $t2,1004
            sw      $t2 ,dat($t3)
            add i   $t3, $t3, 4
            bne     $t3, 100, bucle1
```

Synchronization loop

Transfer loop

Exercise

- The processor must wait until the data byte is available
  - Processor cycles wasted
- Example:
  - If a processor executes 200 MIPS and the waiting time is 5 ms
    - How many instructions must be executed in the synchronization loop?

Solution

- Synchronization loop:
  - In average 5 ms
  - 200 MIPS are executed
  - $I_{bs} = 200*10^6 * 5*10^{-3} = 10^6$
- Transfer loop:
  - 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%)
  - CPU does not do utile work

```
.data
    dat:  .space  400
.text
.globl main
main:       lii    $t3,  0
  bucle1:   li     $t0, 0
            sw     $t0, 1008
  bucle2:   lw     $t1, 1000
            beqz $t1, bucle2
            lw     $t2,1004
            sw     $t2 ,dat($t3)
            add i  $t3, $t3, 4
            bne    $t3, 100, bucle1
```

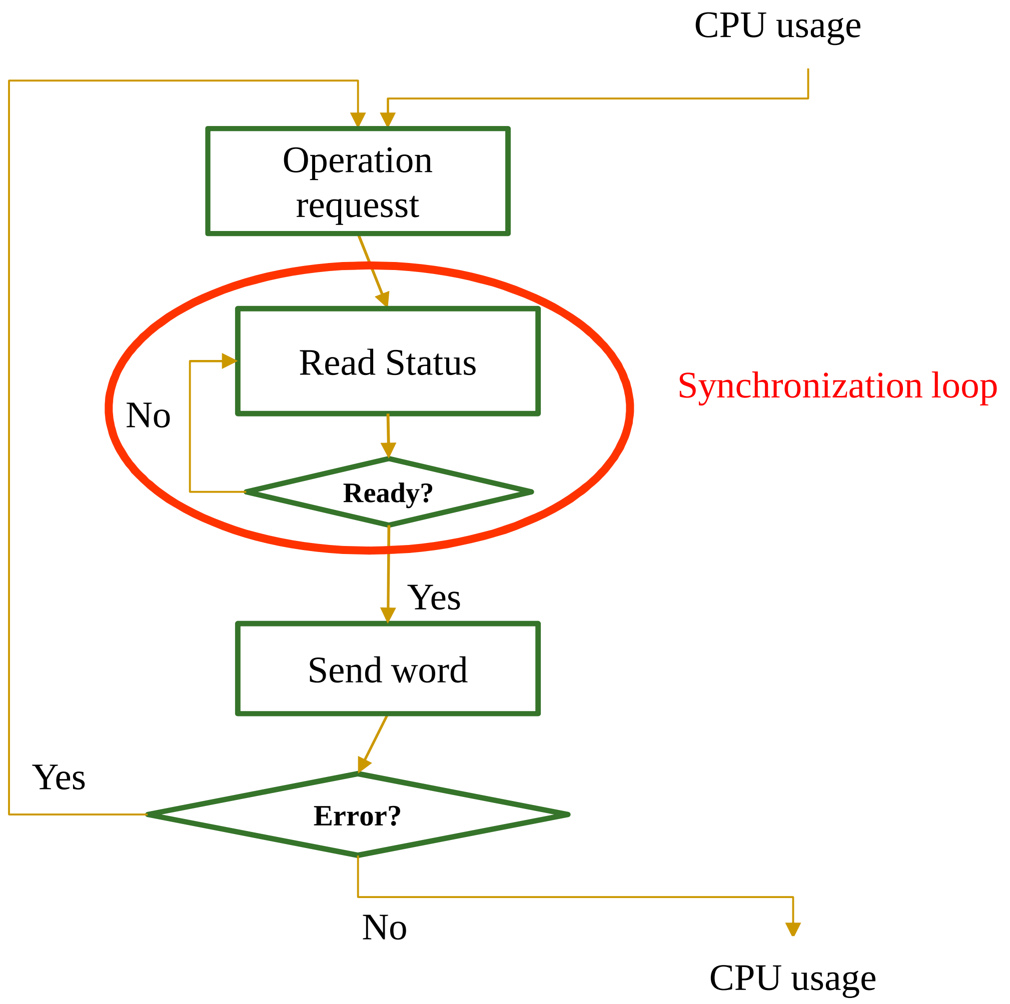
Synchronization loop

Transfer loop

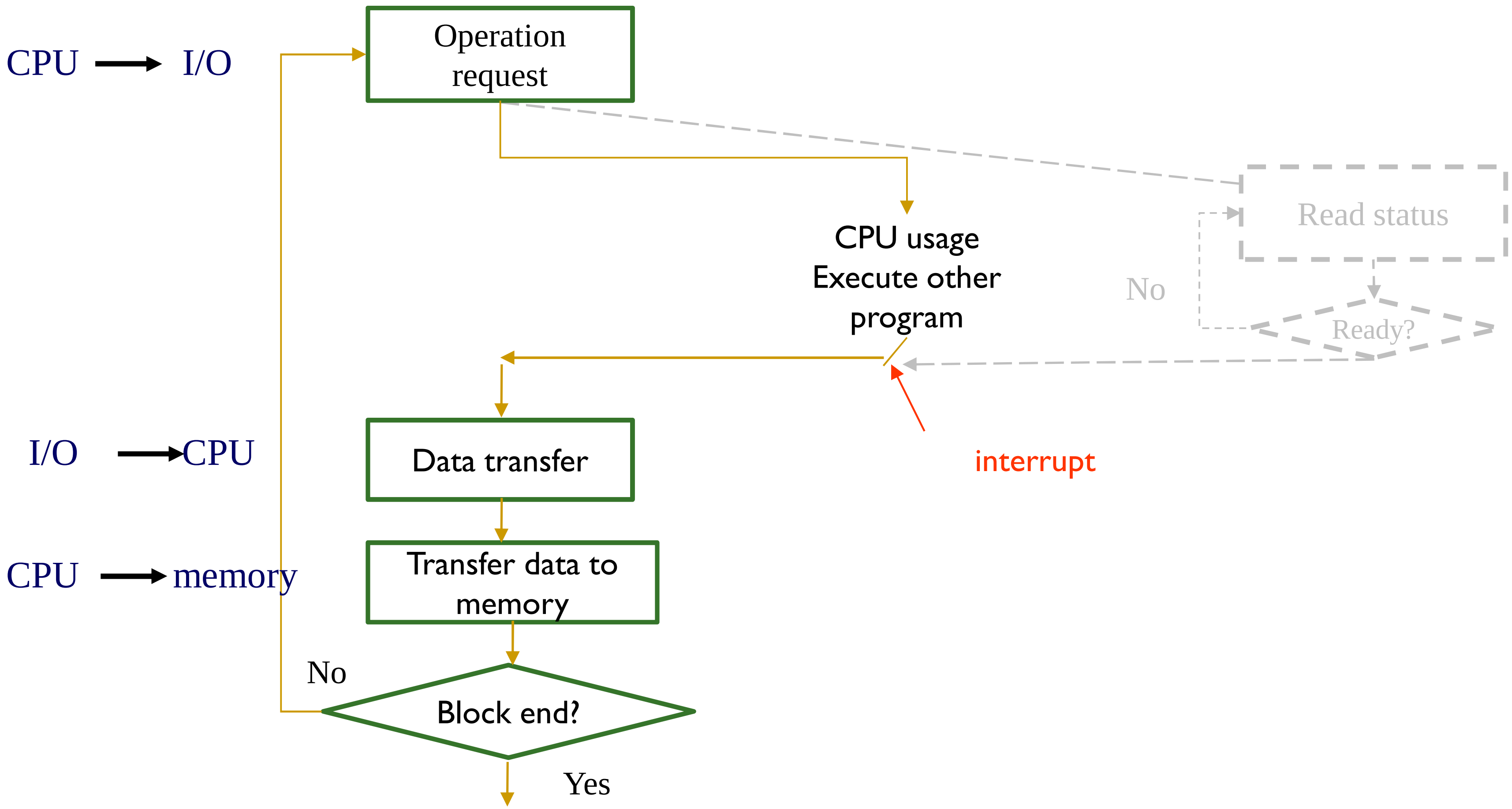
Exercise

- Let be a processor of 500 MHz. If the average number of clock cycles needed to perform an instruction is 25,
  - What is the average number of instructions that this computer can execute?

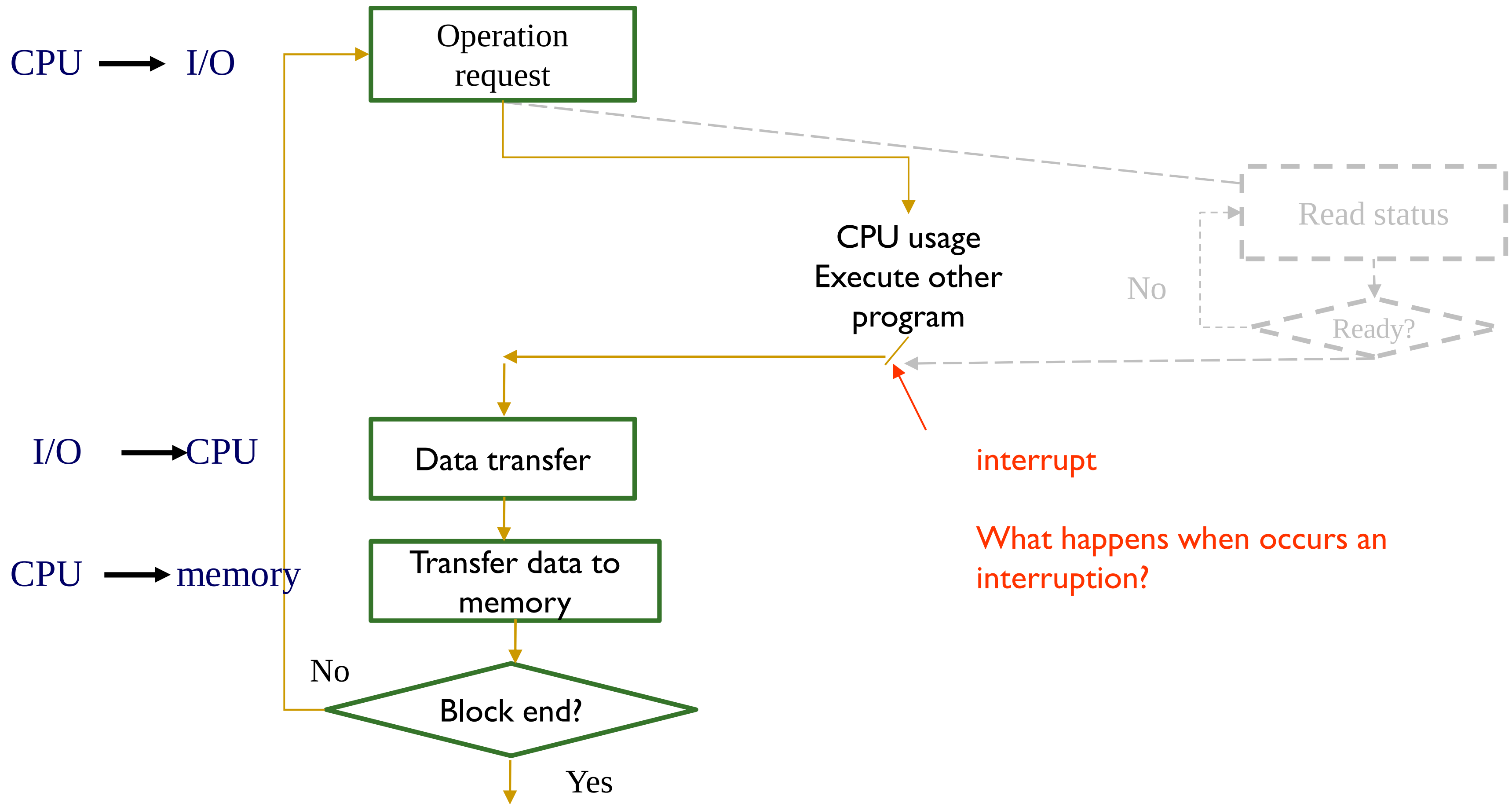
Programmed I/O



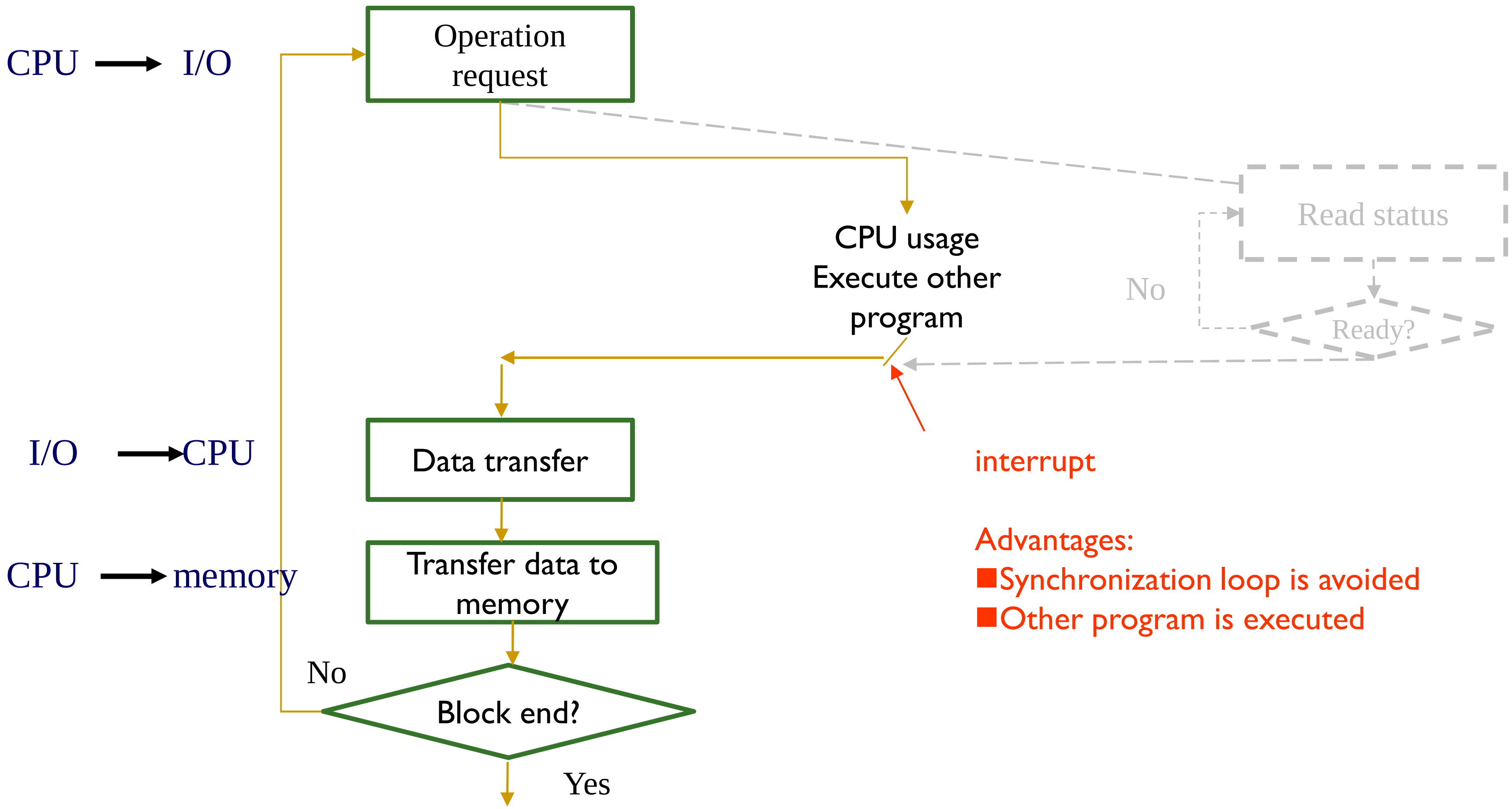
Interrupt driven I/O



Interrupt driven I/O



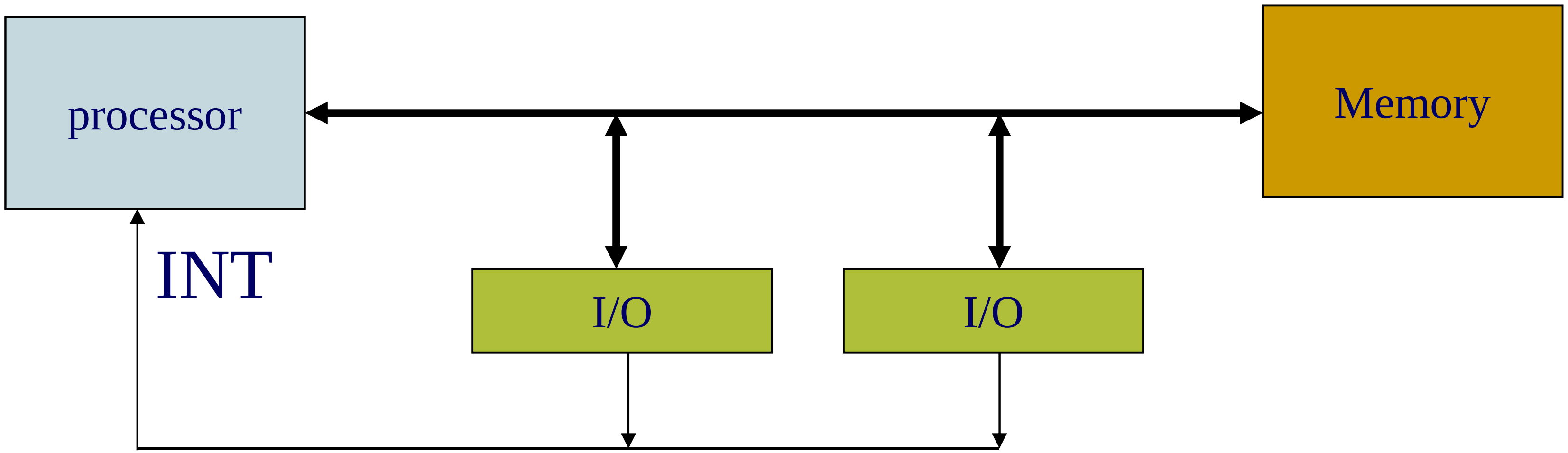
Interrupt driven I/O



Advantages:

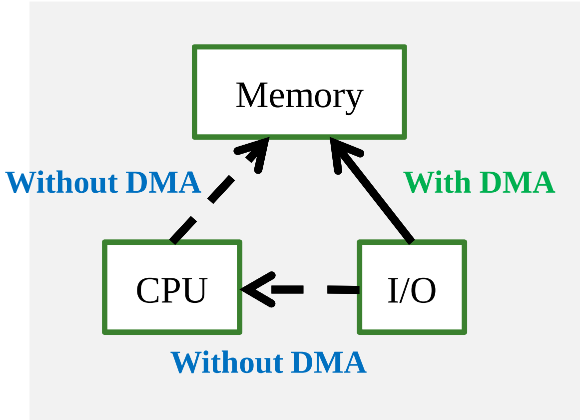
- Synchronization loop is avoided
- Other program is executed

Interrupt driven I/O

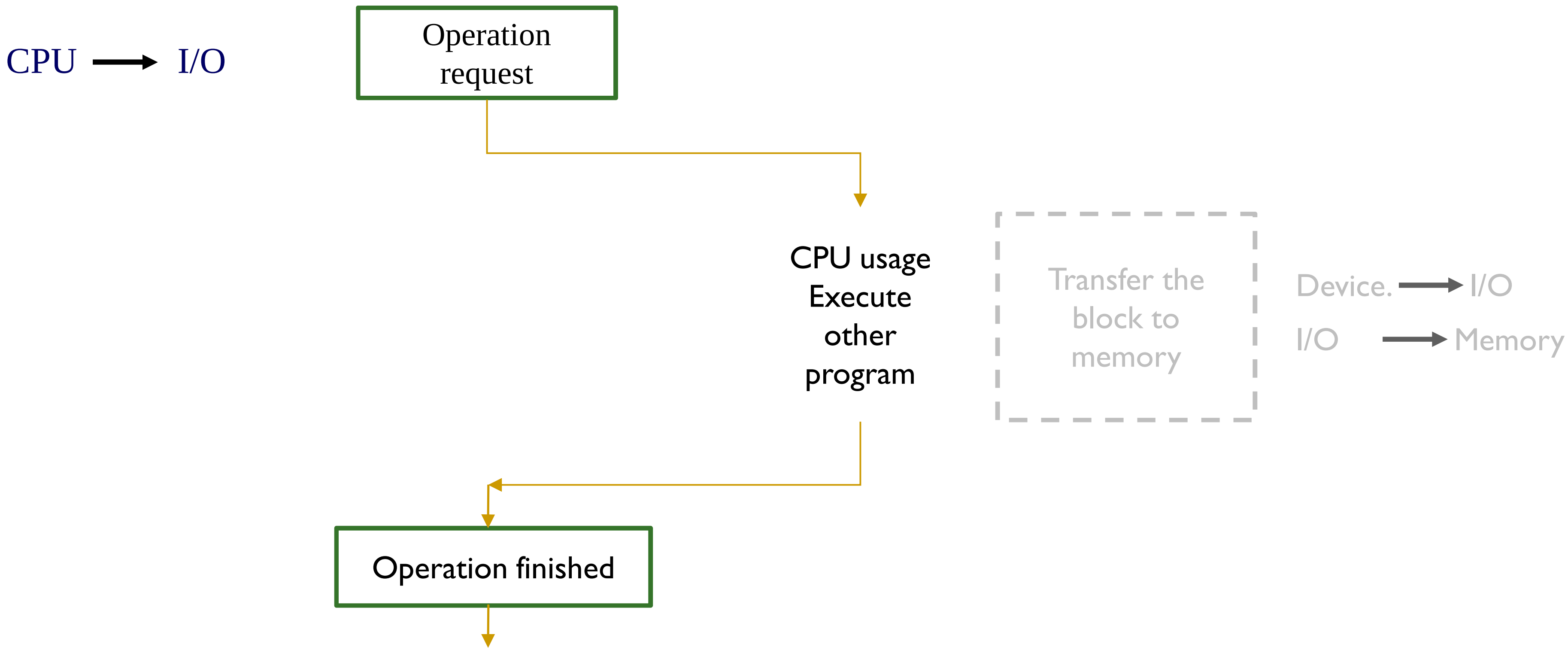


DMA, Direct memory access

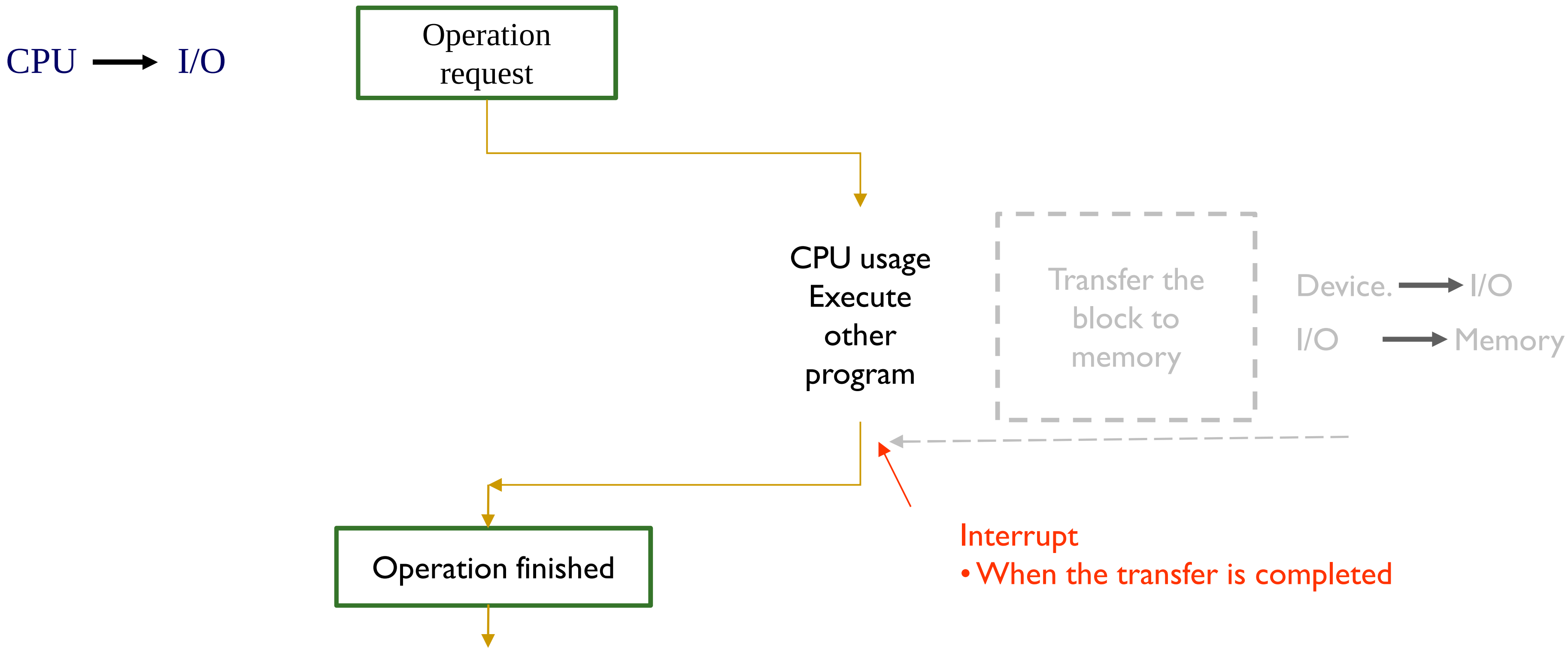
- The processor does not carry out the transfer between the I/O module and the memory
  - With interrupts the synchronization loop is avoided, but the transfer is carry out by the processor
  - For a block with N bytes, N interrupts are needed
- Using DMA, the whole transfer is carried out by the I/O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



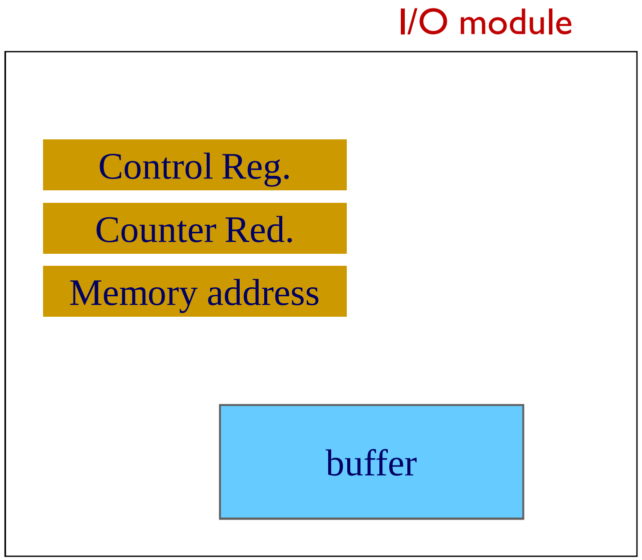
Transfer a block using DMA



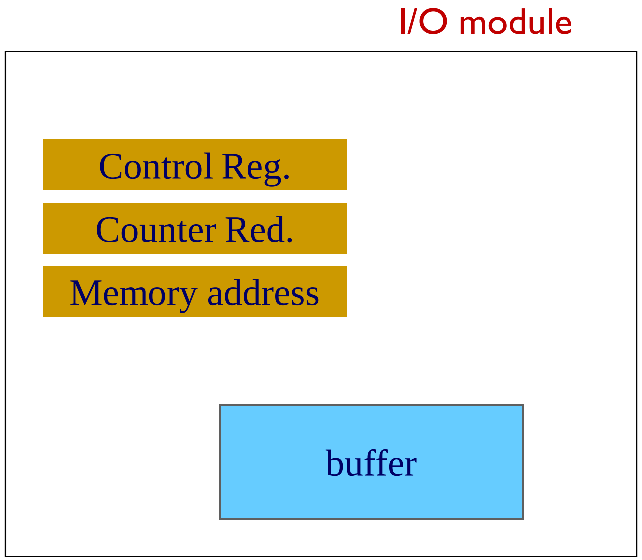
Transfer a block using DMA



Typical DMA block diagram

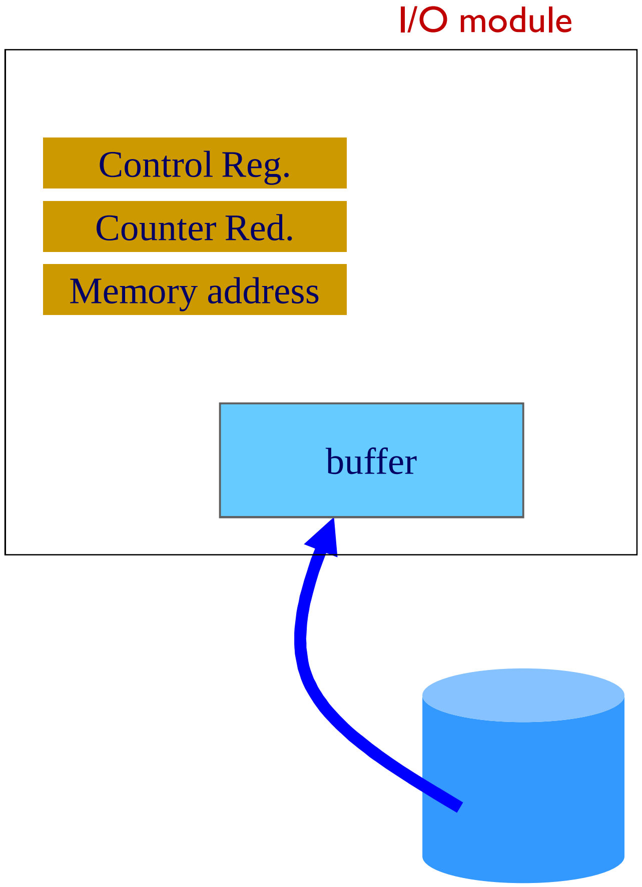


Data transfer with DMA



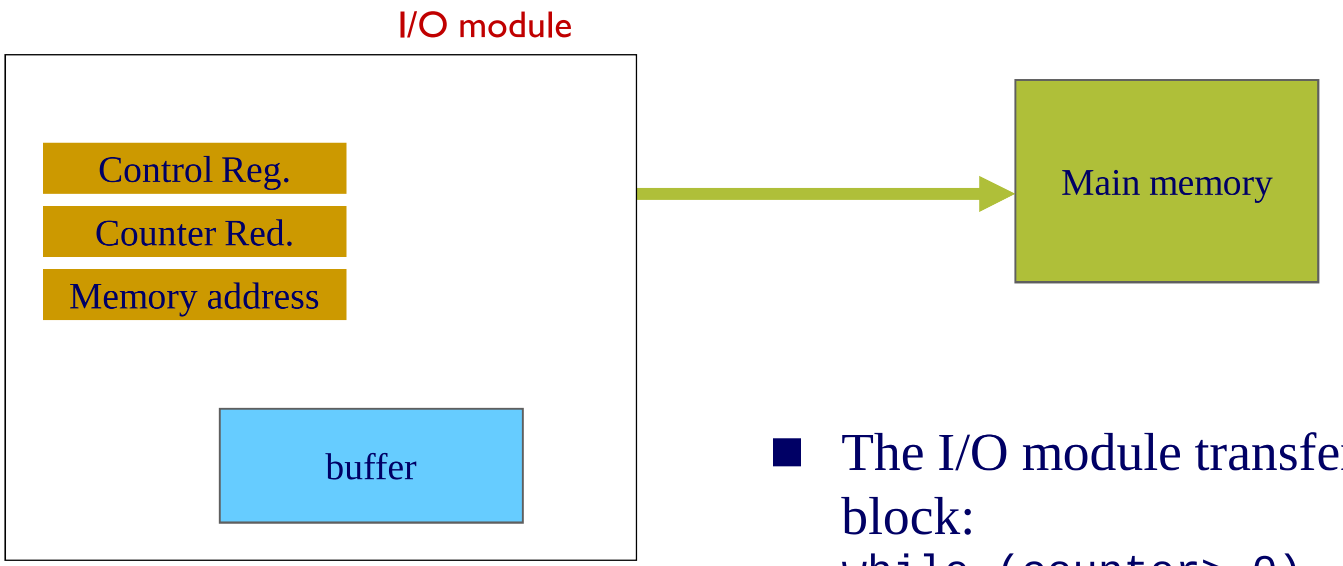
- The processor writes in I/O registers (using I/O instructions)
  - Operation (control reg.)
    - Read, write,
  - The number of bytes to transfer (counter reg.)
  - Memory address where
    - Data are stored (write in device)
    - Store the data (reading from device)

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)

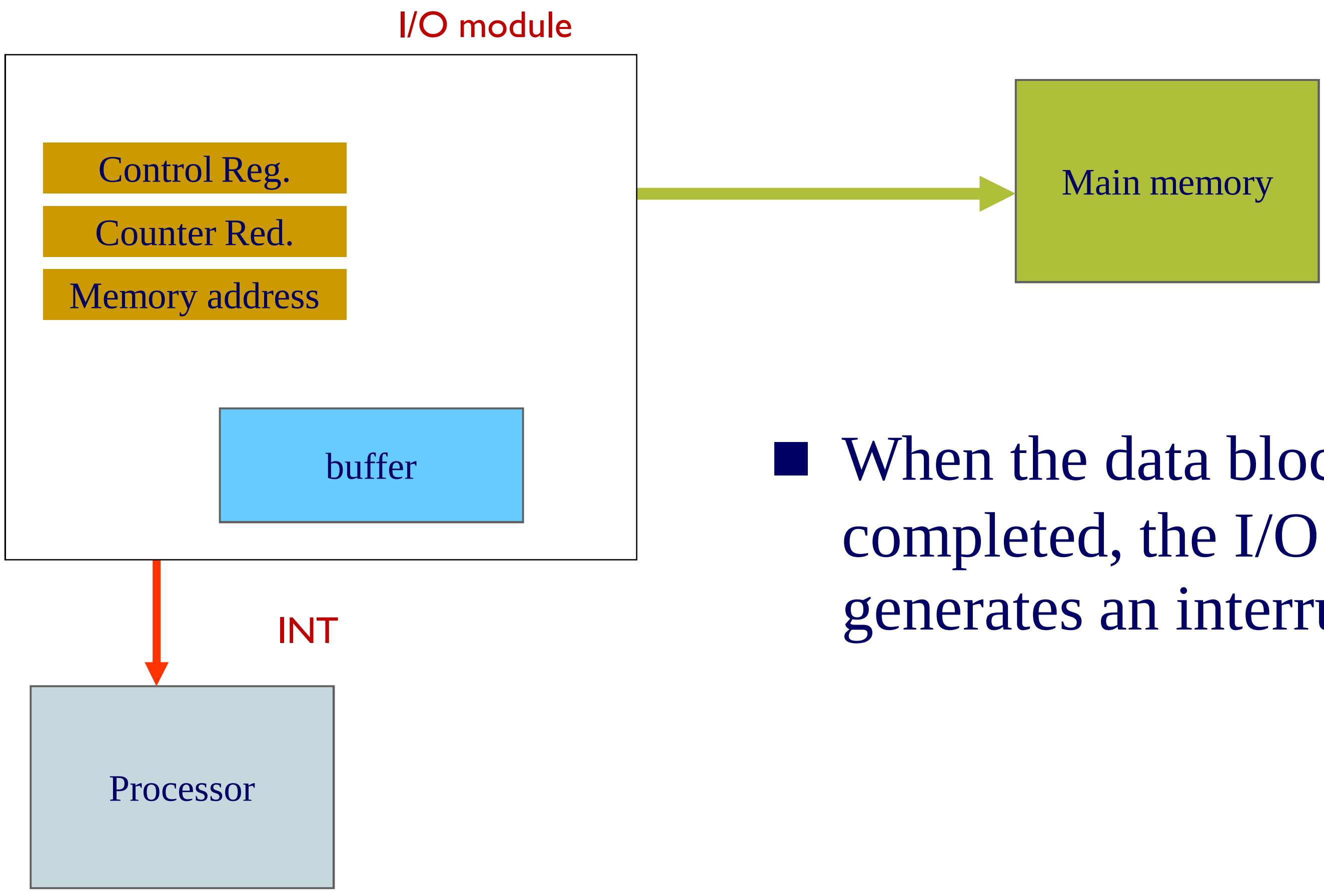
Data transfer with DMA



- The I/O module transfers the data block:

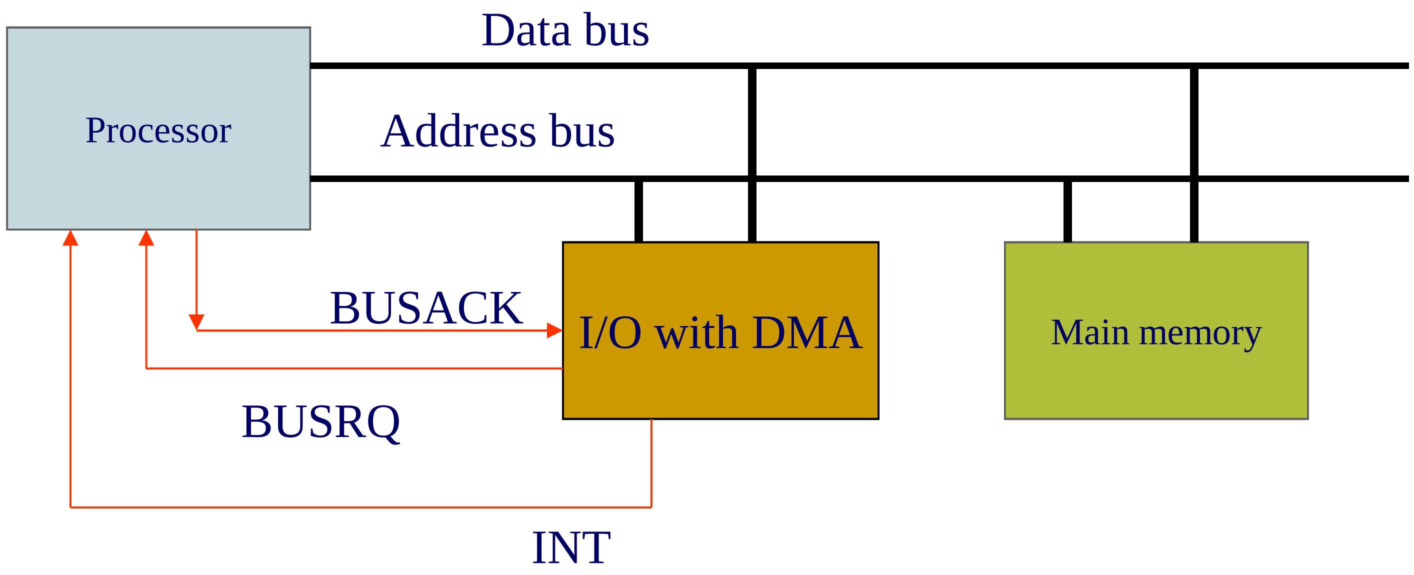
```
while (counter> 0)
{
   Byte (word) ——→ memory addr.
   memory addr ++;
    counter --;
}
```

Data transfer with DMA



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

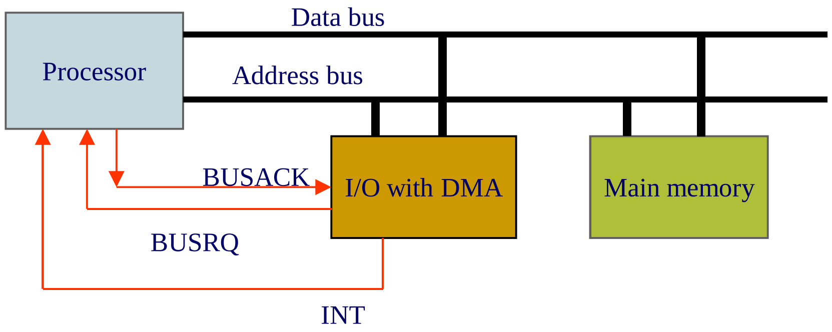
Connection among DMA devices and memory



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

Connection among DMA devices and memory

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.

The importance of controllers

- Linux kernel statistics (2007-2008):
  - 9,2 millions or code lines
  - 10% of increase every year:
    - On average, every day:
      - 4.500 code lines are added,
      - 1.800 code lines are erased
      - 1.500 code lines are modified
  - Most of the code belongs to drivers:
    - 55% of source code are code for device controllers (drivers)
    - Software of the operating system needed to control the behavior of the associated device