Interfacing Peripherals

Instructor: Dmitri A. Gusev

Fall 2007

CS 502: Computers and Communications Technology

Lecture 12, October 15, 2007

Interfacing Processors and Peripherals

- I/O Design affected by many factors (expandability, resilience)
- Performance:
  - — access latency
  - — throughput
  - — connection between devices and the system
  - — the memory hierarchy
  - — the operating system
- A variety of different users (e.g., banks, supercomputers, engineers)

I/O Devices

- Very diverse devices
  - — behavior (i.e., input vs. output)
  - — partner (who is at the other end?)
  - — data rate

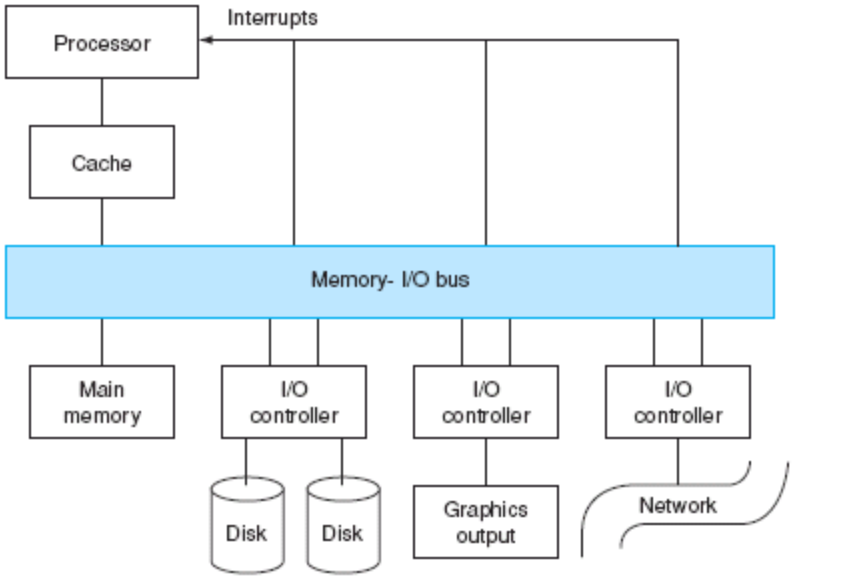
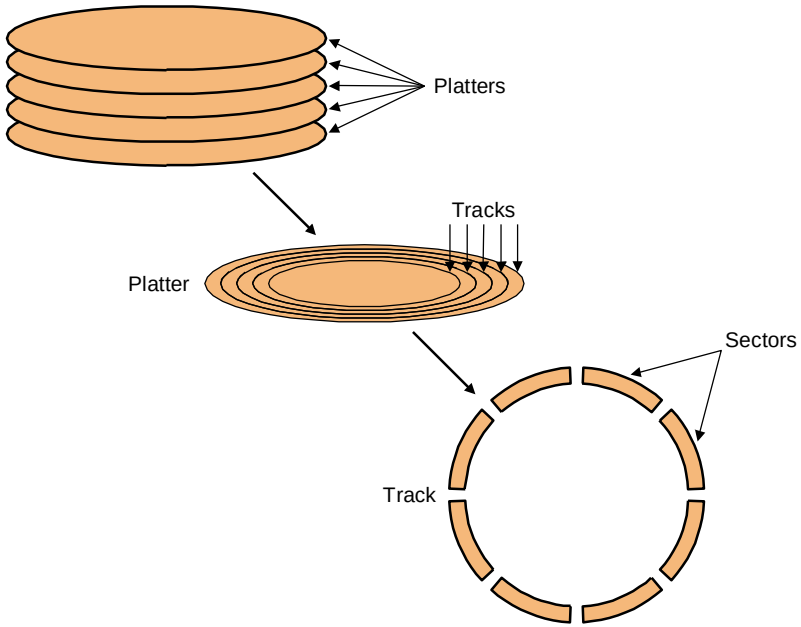


FIGURE 8.2 A typical collection of I/O devices. The connections between the I/O devices, processor, and memory are usually called *buses*. Communication among the devices and the processor use both interrupts and protocols on the bus, as we will see in this chapter. Figure 8.11 on page 585 shows the organization for a desktop PC.

I/O Example: Disk Drives



- To access data:
  - — seek: position head over the proper track (3 to 14 ms. avg.)
  - — rotational latency: wait for desired sector (.5 / RPM)
  - — transfer: grab the data (one or more sectors) 30 to 80 MB/sec

Magnetic Disks

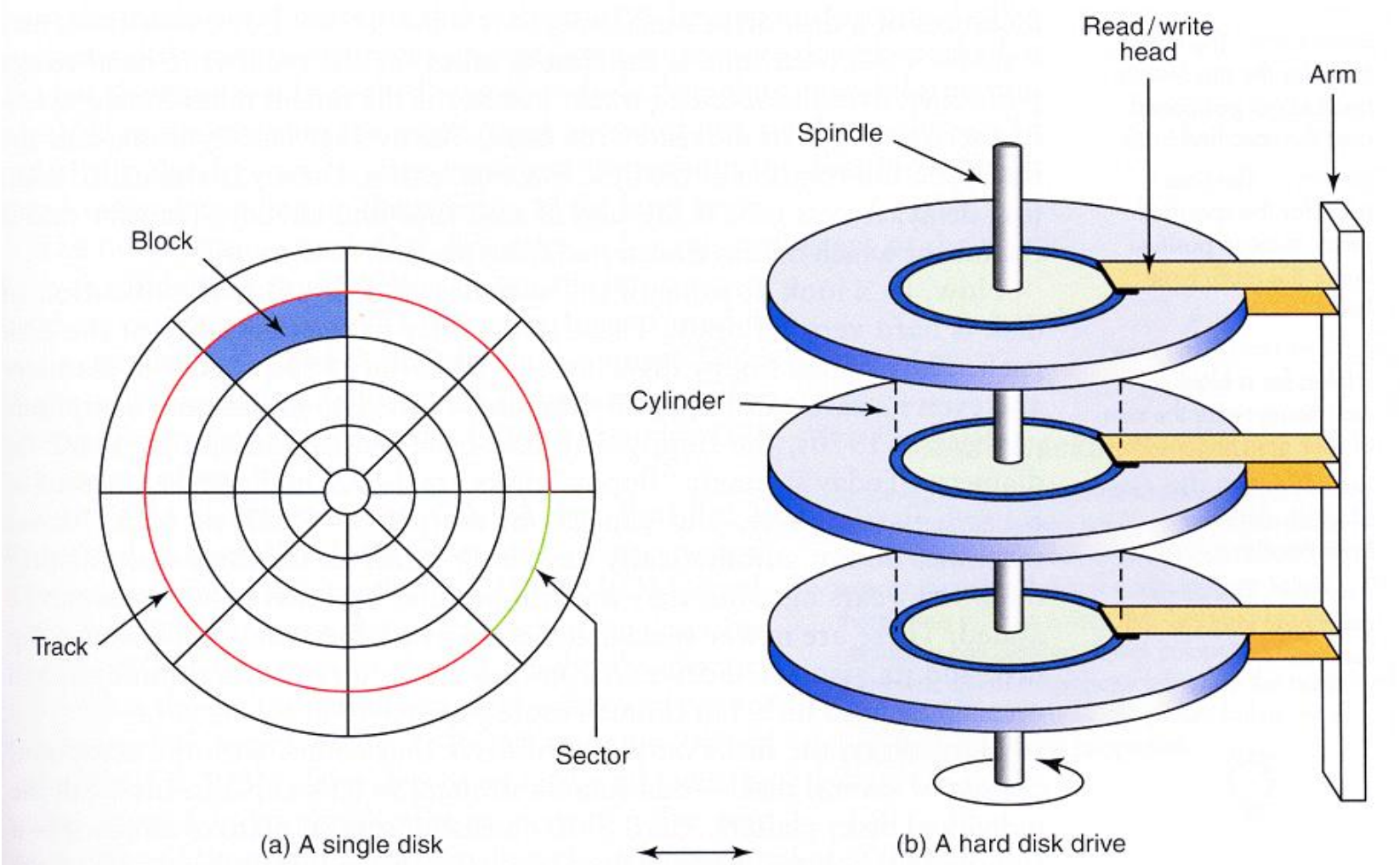


Figure 5.5 The organization of a magnetic disk

Measures of Disk Drive's Efficiency

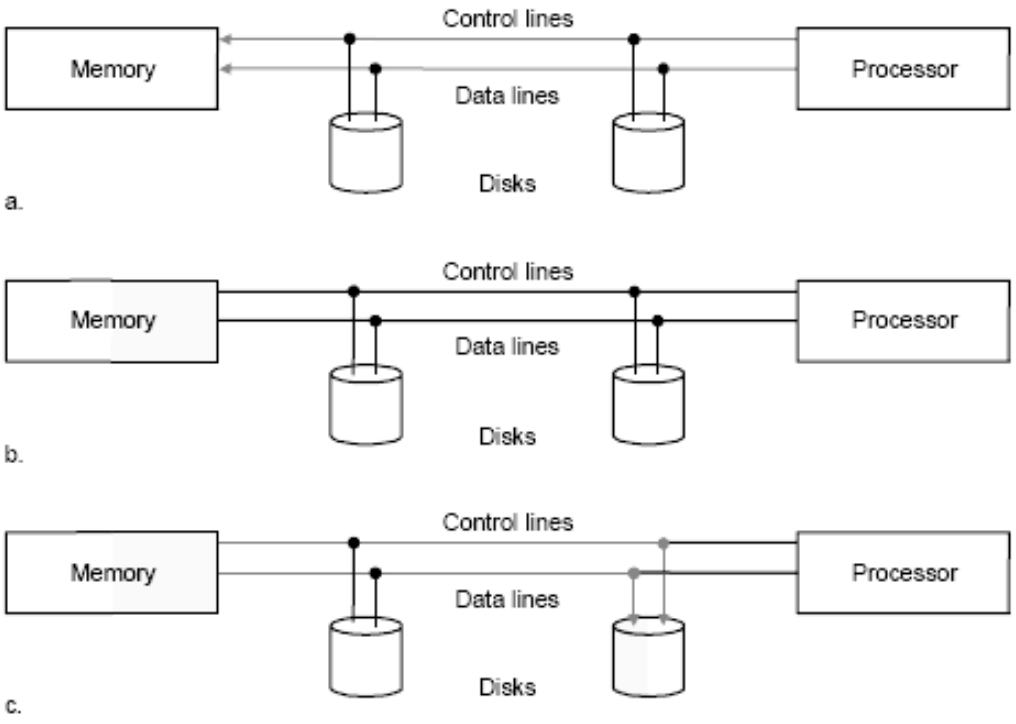
- *Seek time* is the time it takes for the read/write head to get positioned over the specified track
- *Latency* is the time it takes for the specified sector to spin to the read/write head
- *Access time* = *Seek time* + *Latency*. This is the time it takes for a block to start being read
- *Transfer rate* is the rate at which data is transferred from the disk to memory

I/O Example: Buses

- Shared communication link (one or more wires)
- Difficult design:
  - — may be bottleneck
  - — length of the bus
  - — number of devices
  - — tradeoffs (buffers for higher bandwidth increases latency)
  - — support for many different devices
  - — cost
- Bus lines
  - — Control lines
  - — Data lines (data, commands, addresses)
- Bus transactions
  - — Read (output): memory to I/O device
  - — Write (input): I/O device to memory
- Types of buses:
  - — processor-memory (short high speed, custom design)
  - — backplane (high speed, often standardized, e.g., PCI)
  - — I/O (lengthy, different devices, e.g., USB, Firewire)
- Synchronous vs. Asynchronous
  - — use a clock and a synchronous protocol, fast and small
    but every device must operate at same rate and
    clock skew requires the bus to be short
  - — don’t use a clock and instead use handshaking

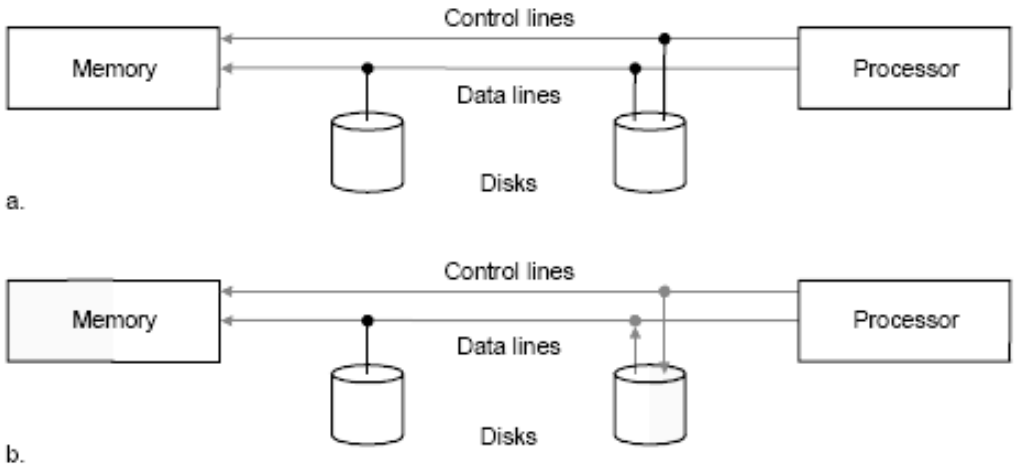
Bus transactions: output

- Request a read on control lines and supply data at the data lines
- Memory access
- Memory signals data availability and transfer them on the data line to I/O

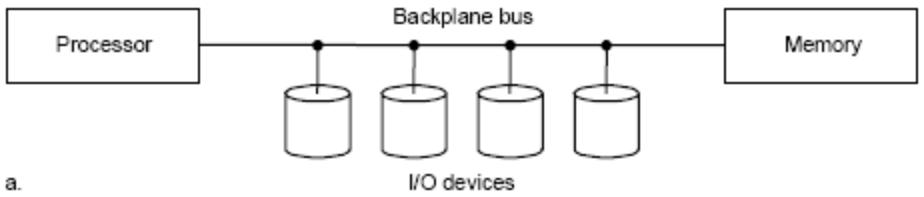


Bus transactions: input

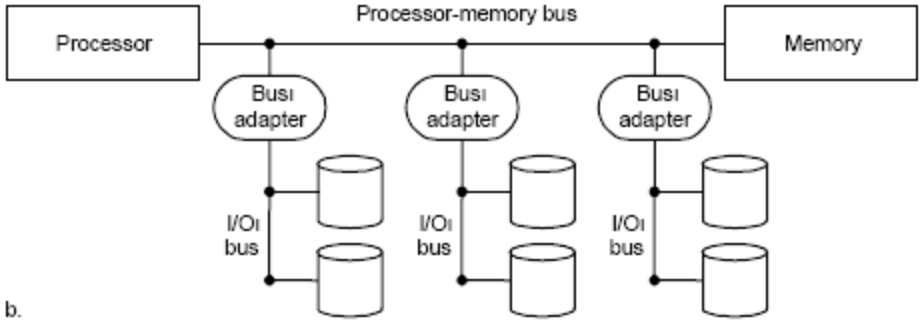
- Write request for memory on the control lines and address on the data lines
- Memory signal the I/O device that is ready and transfers starts



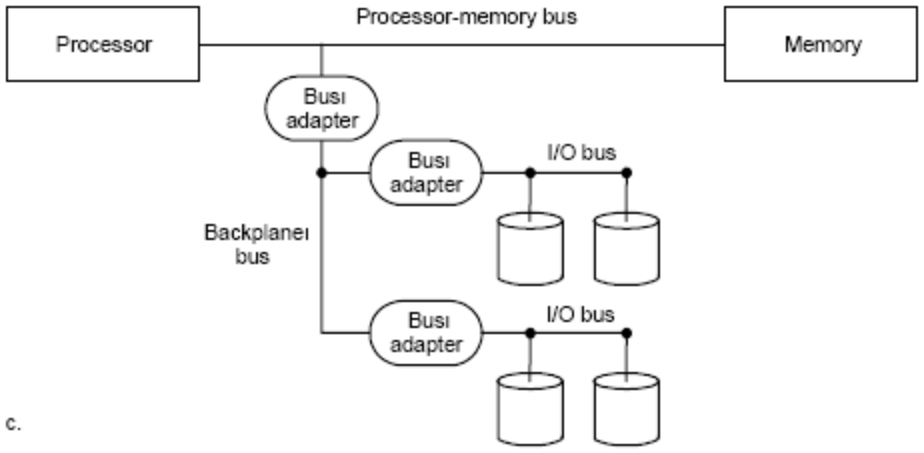
Types of buses



a.



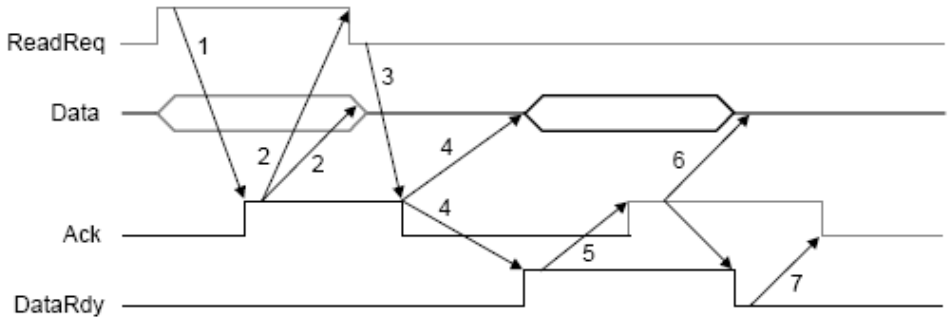
b.



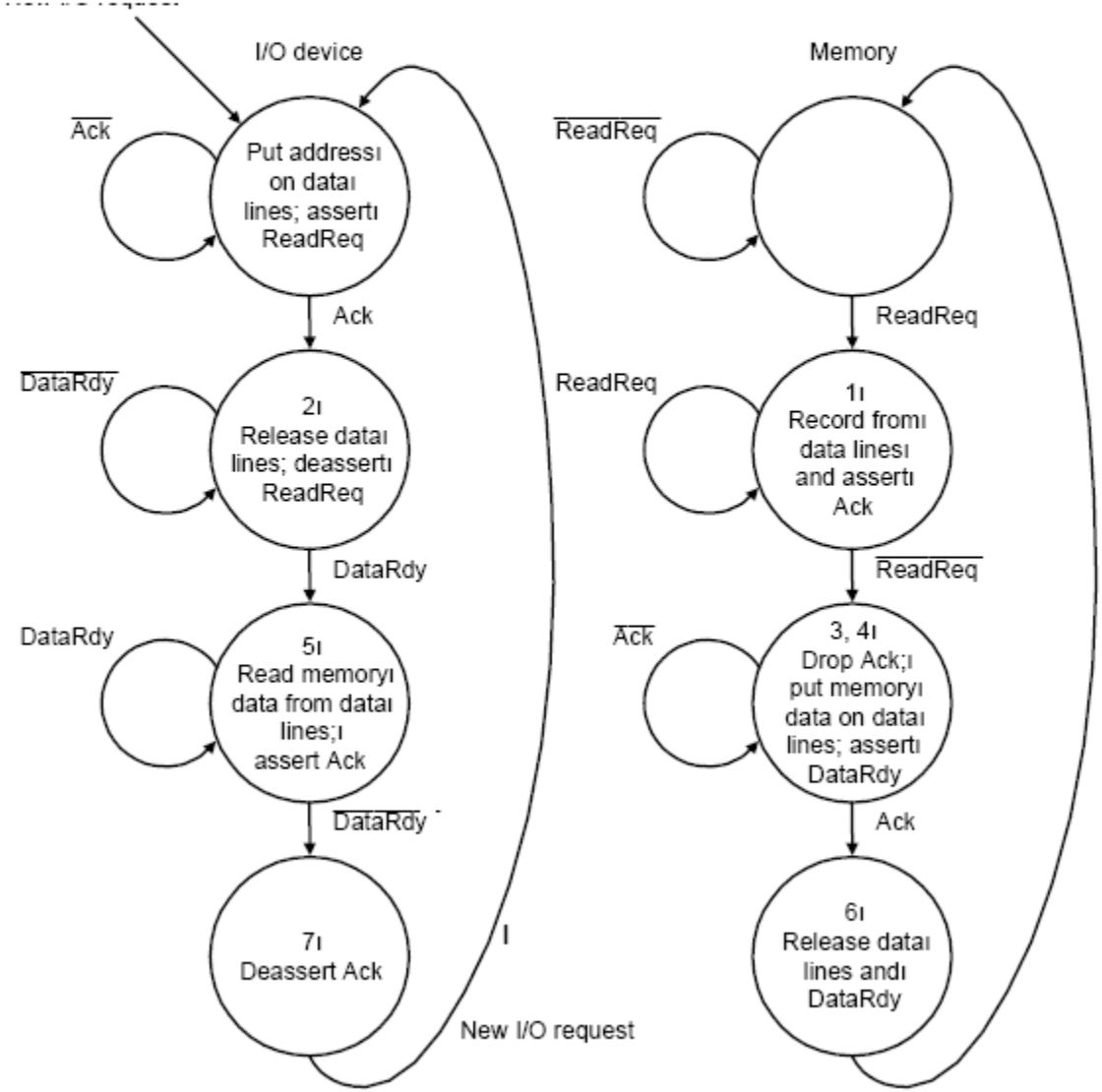
c.

Synchronous and Asynchronous buses

- Synchronous buses: Using a clock synchronized communication protocol
  - All devices use a single clock
  - Cannot be long
- Asynchronous buses: handshaking protocol
  - Reading a word from memory:
    - Read request (*ReadReq*)
    - Data are ready to read (*DataRdy*)
    - Acknowledge the ReadReq and DataRdy signals of the other party (*Ack*)



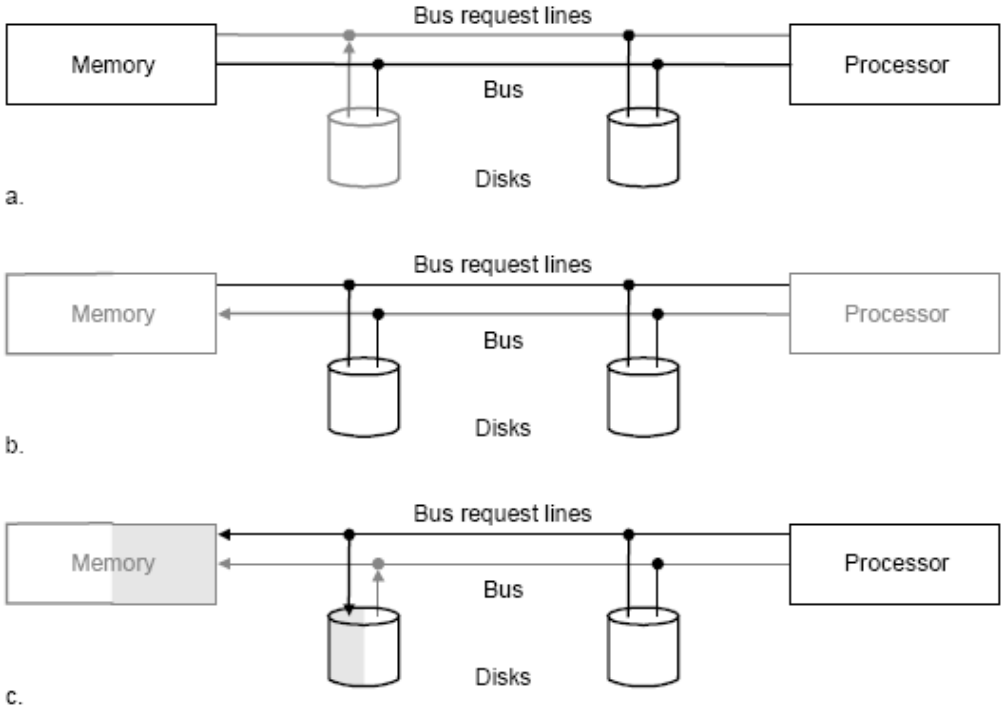
Implementing handshaking



Increasing the bus bandwidth

- Increasing the width of the data bus lines
- Using separate address and data lines
- Transferring multiple words (blocks)

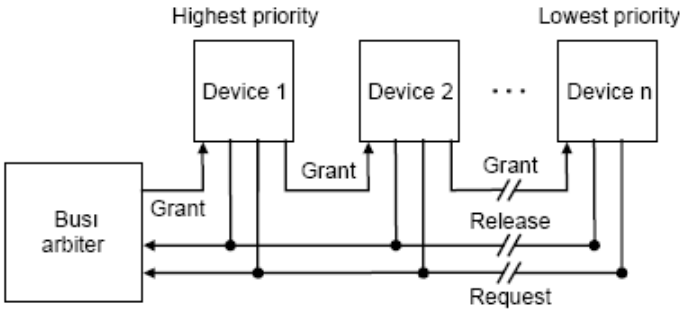
Bus access: master and slave



- I/O device generates a request to the processor to use the bus
- The processor responds and generates the corresponding bus control signals
- The processor notifies the device and the device places the address on the bus

Bus arbitration schemes

•**Daisy chain arbitration:** A single bus grant line is run through the devices from highest priority to lowest. A higher priority device intercepts the bus grant signal, not allowing a lower priority device to see it.
•**Centralized, parallel arbitration:** Using multiple request lines managed by a centralized arbiter.
•**Distributed arbitration by self-selection:** Multiple request lines, but the device requesting the bus access determine who will be granted access. Each bus requesting device places its identity code on the bus.
•**Distributed arbitration by collision detection:** Each device independently requests the bus. Multiple simultaneous requests result in a collision, which is detected and solved.



I/O Bus Standards

- Today we have two dominant bus standards:

Characteristic	Firewire (1394)	USB 2.0
Bus type	I/O	I/O
Basic data bus width (signals)	4	2
Clocking	asynchronous	asynchronous
Theoretical peak bandwidth	50 MB/sec (Firewire 400) or 100 MB/sec (Firewire 800)	0.2 MB/sec (low speed), 1.5 MB/sec (full speed), or 60 MB/sec (high speed)
Hot plugable	yes	yes
Maximum number of devices	63	127
Maximum bus length (copper wire)	4.5 meters	5 meters
Standard name	IEEE 1394, 1394b	USE Implementors Forum

FIGURE 8.9 Key characteristics of two dominant I/O bus standards.

Interfacing I/O: still open questions

- How is a program I/O request transformed into device commands and communicated to the device?
- How is data actually transferred to or from memory?
- Why is the operating system involved?
    - The I/O devices are shared by multiple programs managed by the operating system
    - I/O devices use interrupts which are handled by the operating system
- What is the role of the operating system?

The role of the OS

- The OS guarantees that each program accesses only the portion of the I/O device to which it has rights.
- The OS provides abstractions (high-level functions) for handling low-level I/O device operations.
- The OS handles the interrupts generated by I/O devices
- The OS provides equitable access to the shared I/O devices.

In order to perform these functions the OS must be able to communicate to the I/O devices and to prevent the user programs from communicating directly to the I/O devices. That is:

- The OS must be able to give commands to I/O devices
- The I/O devices must be able to notify the OS for some events.
- Data must be transferred between memory and I/O devices.

Commands to I/O devices

• *Memory mapped I/O*. Using a special address space assigned to each I/O device - command words, data buffers, status registers. The memory system ignores these addresses.

• *Special I/O instructions*. Processor instructions specifying the device number and command word. These instructions can be executed only in supervisor (OS) mode.

Communicating with the processor

• *Polling.* Periodically checking the status words of the I/O devices. Polling depends on the type of the device. Some devices initiate operations independently (mouse), others - under the control of the OS (disk).

• *Interrupt-driven I/O.* The device notifies the processor that it has completed some operation or needs attention by causing an interrupt.

- The I/O interrupt is asynchronous with respect to the instruction execution (in contrast to page faults or arithmetic exceptions). Check for I/O interrupt only when starting a new instruction.
- In addition, information for the device causing the interrupt and its priority must be conveyed to the OS.

Transferring data between I/O device and memory

- Data transfer using the processor
  - *Data transfer based on polling*. For example, reading the mouse position.
  - *Interrupt-driven data transfer*. When the OS recognizes an I/O interrupt it reads the status word and reads or writes data.

- Direct memory access (DMA). Transferring data between the I/O and memory without involving the processor - *DMA controller*.
  - The processor supplies to the DMA controller the device number, the type of the operation, the memory address and the number of bytes to transfer.
  - DMA starts the operation and arbitrates the bus while transferring the data.
  - When the transfer is complete the DMA controller interrupts the processor.

DMA and the memory system

- DMA and virtual memory (physical or virtual pages?). Crossing page boundaries when physical pages are used is a problem.
  - Solution 1: using virtual pages.
  - Solution 2: chained transfer (breaking DMA transfer into a series of transfers each one within a single physical page).

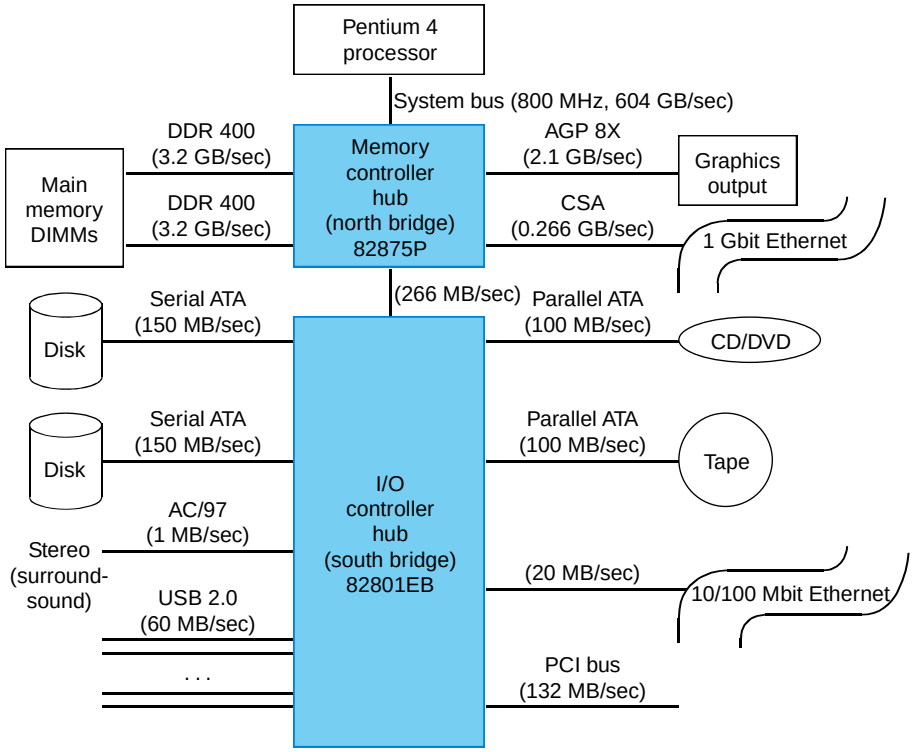
- DMA and the cache (coherency problem)
  - Route DMA through the cache (negative impact on performance).
  - Invalidate the cache for I/O read or force write-back (flashing) for I/O write.
  - Selective (partial) invalidation or flashing.

Designing an I/O system

- Taking in account *latency constraints and bandwidth constraints.*

Pentium 4

- I/O Options



	875P chip set	845GL chip set
Target segment	Performance PC	Value PC
System bus (64 bit)	800/533 MHz	400 MHz
Memory controller hub ("north bridge")		
Package size, pins	42.5 × 42.5 mm, 1005	37.5 × 37.5 mm, 760
Memory speed	DDR 400/333/266 SDRAM	DDR 266/200, PC133 SDRAM
Memory buses, widths	2 × 72	1 × 64
Number of DIMMs, DRAM Mbit support	4, 128/256/512 Mbits	2, 128/256/512 MBits
Maximum memory capacity	4 GB	2 GB
Memory error correction available?	yes	no
AGP graphics bus, speed	yes, 8X or 4X	no
Graphics controller	external	internal (Extreme Graphics)
CSA Gigabit Ethernet interface	yes	no
South bridge interface speed (8 bit)	266 MHz	266 MHz
I/O controller hub ("south bridge")		
Package size, pins	31 × 31 mm, 460	31 × 31 mm, 421
PCI bus: width, speed, masters	32-bit, 33 MHz, 6 masters	32-bit, 33 MHz, 6 masters
Ethernet MAC controller, interface	100/10 Mbit	100/10 Mbit
USB 2.0 ports, controllers	8, 4	6, 3
ATA 100 ports	2	2
Serial ATA 150 controller, ports	yes, 2	no
RAID 0 controller	yes	no
AC-97 audio controller, interface	yes	yes
I/O management	SMbus 2.0, GPIO	SMbus 2.0, GPIO

FIGURE 8.12 Two Pentium 4 I/O chip sets from Intel. The 845GL north bridge uses many fewer pins than the 875 by having just one memory bus and by omitting the AGP bus and the Gigabit Ethernet interface. Note that the serial nature of USB and Serial ATA means that two more USB ports and two more Serial ATA ports need just 39 more pins in the south bridge of the 875 versus the 845GL chip sets.