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.

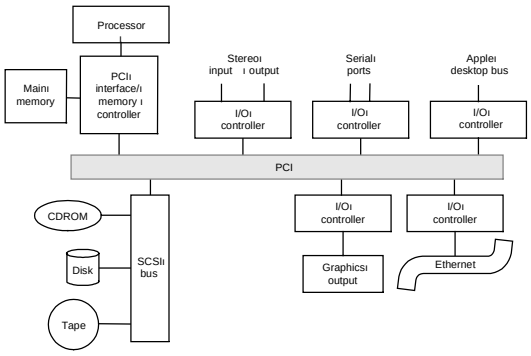
DMA and the memory system

- 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.*



Apple Macintosh 7200 I/O system