Input/Output

CS 342 – Operating Systems

Ibrahim Korpeoglu
Bilkent University
Department of Computer Engineering

OS and I/O

- Os is responsible to control all the I/O devices in the system
  - Issue commands
  - Catch interrupts
  - Handles errors
  - Provide a programming interface to I/O devices for users.
    - Make device independent.

I/O Hardware

- Hardware Views
  - Circuits, power supplies, motors, wires, etc.
    - This is inside of a device
  - The commands that hardware accepts, functions it carries out, errors that it reports back..
    - This is what an OS designed is interested in.

I/O Devices

- Block devices
  - Information is stored in fixed-size blocks.
  - Each block is addressable
  - Transfer from device to memory is usually done in blocks.
  - Blocks can be read/written independently
  - Example: disks.

I/O Devices

- Character Devices
  - Delivers or accepts a stream of characters.
  - Characters are not addressable and randomly accessible efficiently.
  - Example: printers, network cards, mouse, keyboard, etc.

I/O Devices

- Other type of devices
  - Clocks
    - Generates interrupts periodically.
  - Memory mapped screens
    - Accepts bytes and renders them to the CRT screen.
      - Digital to Analog signal conversion

I/O devices and transfer speeds (data rates)

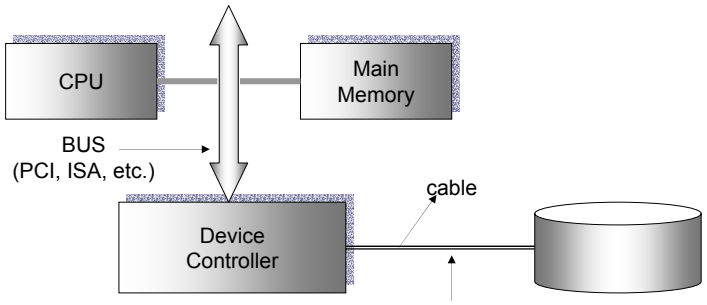
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7KB/sec
Laser Printer	100KB/sec
Scanner	400KB/sec
Ethernet	1.25 MB/sec
USB	1.5 MB/sec
Digital Camcorder	4 MB/sec
IDE Disk	5 MB/sec
40x CD-ROM	6 MB/sec

I/O devices and transfer speeds (data rates)

ISA Bus	16.7 MB/sec
PCI Bus	528 MB/sec
XGA Monitor	60 MB/sec
SCSI Disk	80 MB/sec
FireWire	50 MB/sec
SONET network	78 MB/sec
Fast Ethernet	12.5 MB/sec
Gigabit Ethernet	125 MB/sec
Ultrium Tape	320 MB/sec
Telephone Channel	8 KB/sec

I/O Device

- Consists of
  - A *mechanicel* part: device itself
  - An *electronic* part: the device controller
    - Printed circuit card that is plugged in to the slot of a computer.



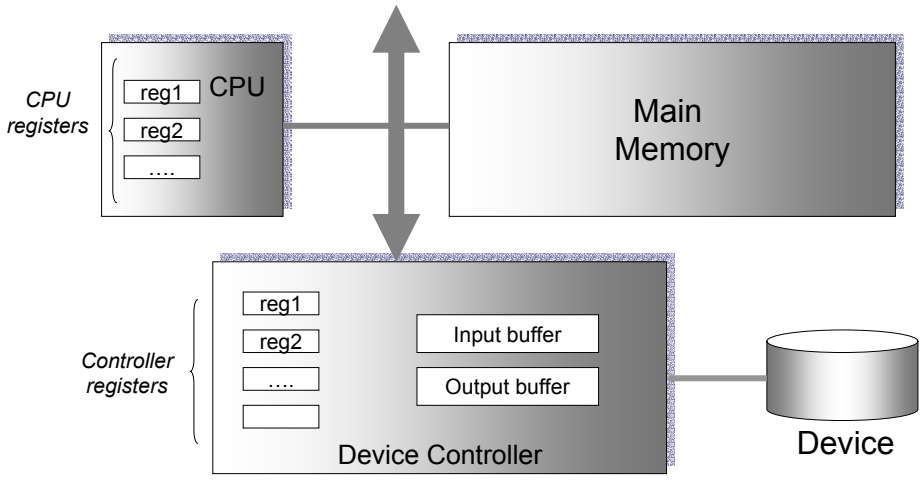
I/O Device Example: Disks

- The information coming from disks is a serial stream of bytes
  - Preamble
  - Data (4096 bits – 512 bytes) (a disk sector)
  - Error Checking Code (ECC) (checksum)
- Controller is responsible to
  - Buffer this information inside its local memory
  - Do the error checking
  - Convert this stream stream of bits into a block of bytes.
- Then the block can be copied into the main memory of the system

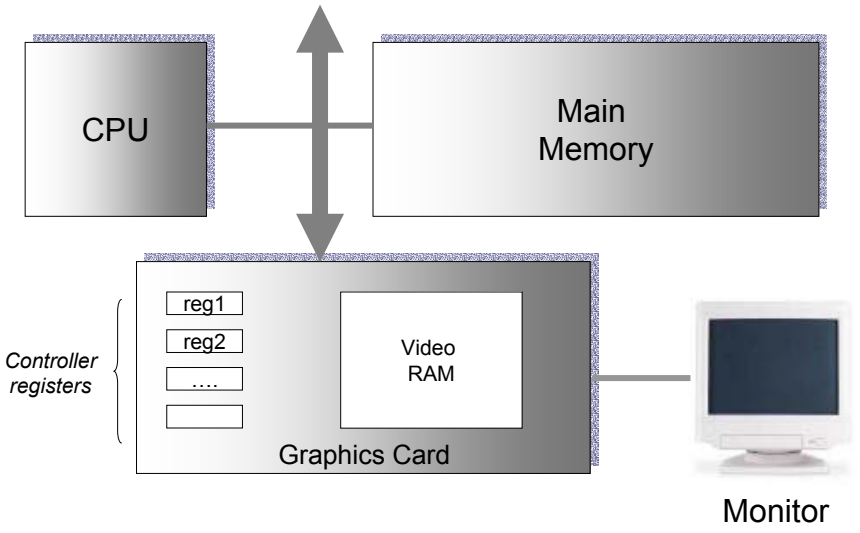
I/O Models

- Specifies how I/O device controllers are addresses and how information is transferred between controllers and main memory.
- 4 Models
  - Memory Mapped I/O
  - Separate I/O and memory space
  - Hybrid Scheme
  - Direct Memory Access (DMA)

I/O Models – Device Controller Example



I/O Models – Device Controller Model



I/O Models

- Controller Registers
  - OS can command the devices using these registers
    - It can write a command to the control registers
      - To deliver data, etc.
    - It can check the status of some of the control registers.
- Data buffer
  - OS can read and write a data buffer located on the controller.
  - Example: Video RAM

I/O Models - Addressing

- The problem is
  - how to addresses these control registers and data buffers
  - How to transfer data between controller and memory
- Two alternative methods and one hybrid method exists
  - Separate I/O port spare
  - Memory mapped I/P
  - Hybrid Scheme

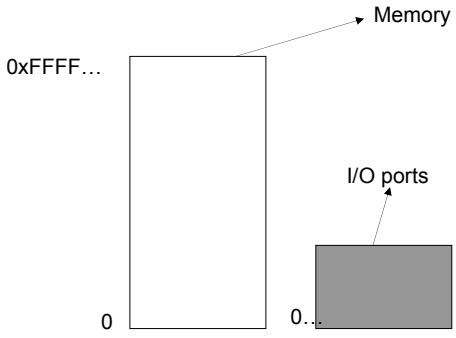
I/O Models – Addressing – Separate I/O space

- Separate I/O space
  - Each control register is assigned an I/O port number
    - An 8 or 16 bit integer
- Special I/O machine instructions are used to read and write from/into these control registers

I/O Models – Addressing – Separate I/O space

- IN REG, PORT
  - Read a byte or word from *control register* addresses as PORT into the *CPU register* REG.
- OUT PORT, REG
  - Write the content of CPU register REG into the control register addresses as PORT.
- Most early computers used the scheme

I/O Models – Addressing – Separate I/O space



IN R0, 4
MOV R0, 4
are completely different.

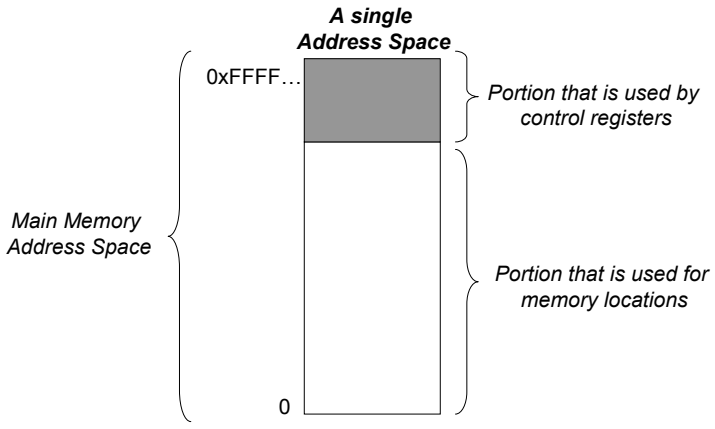
4 in the first instruction refers to main memory location whose address is 4

4 in the second instruction refers to control register whose address is 4.

I/O Models – Addressing – Memory Mapped I/O

- All control registers are mapped into the main memory address space!
  - This means control register addresses are part of main memory address space.
  - Each control register is assigned a unique memory address, to which no physical memory is assigned.
  - Usually the addresses are on top of the address space.

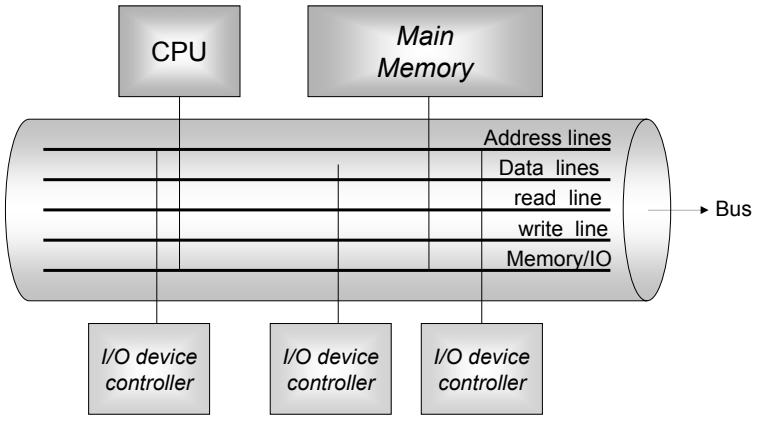
I/O Models – Addressing – Memory Mapped I/O



I/O Models – Addressing – Hybrid Scheme

- Data buffers use memory mapped addresses
- Control registers use addresses from a separate address space.
  - Intel uses this scheme
    - 0-64K – I/O port address space
    - 640K- 1M – reserved main memory address space part that is used by data buffers

How data is transferred



How data is transferred – Separate I/O space

- Assume we will do a read!
- CPU puts the address on the bus
- CPU activates the read line
- CPU asserts a signal line that says whether we will read from *memory* or *I/O device*
- If we will read from memory
  - Address lines contains a memory address from the memory address space
  - Main *Memory responds*.
- If we will read from I/O device
  - Address lines contain an address from I/O port address space.
  - Respective *device controller responds*

How data is transferred – Memory Mapped I/0

- Assume we will do a read!
- CPU puts the address on the bus
- CPU activates the read line
- If we will read from memory
  - Address lines contain an address from part of address space for main memory
  - Main *Memory responds*.
- If we will read from I/O device
  - Address lines contain an address from part of address space for device controllers.
  - Respective *device controller responds*

Advantages of Memory Mapped I/O

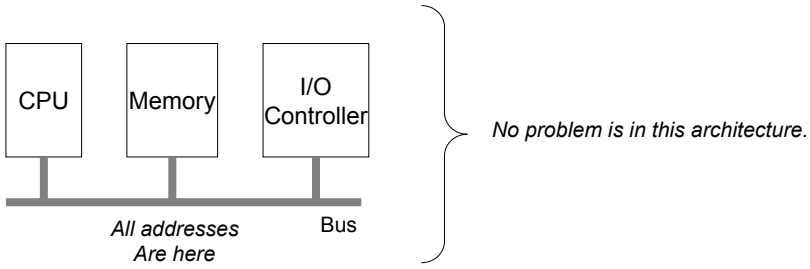
- No special instructions are needed to read and write from/to control registers
  - Control registers can be a variable in device driver C code.
  - Normal assignment statements will move data between control registers and main memory
  - Device deriver can be implemented purely as a C code.
- No special protection mechanism is necessary to protect control registers from direct access of users
  - Control registers can be part of kernel address space
- Every instruction that references memory can also reference control registers.
  - Example: TEST instruction (test for zero)

Disadvantages of Memory Mapped I/O

- Caching
  - If device controller register values are caches, this can be problematic.
    - We don't want to read cached soft copy, but we want to read the control register content.
    - Solution: Disable caching for some virtual pages that correspond to control registers.
- All *main memory modules* and all *I/O device controllers* must examine all memory references (address).

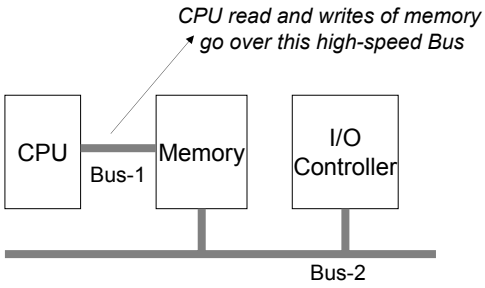
Disadvantages of Memory Mapped I/O

- This can be easy or difficult depending on the architecture of the system



Disadvantages of Memory Mapped I/O

- This can be easy or difficult depending on the architecture of the system

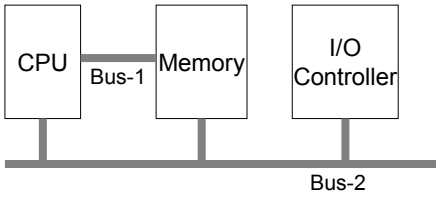


Problem: How can I/O controllers see the addresses that are used for control registers, since all addresses are main memory addresses.

Disadvantages of Memory Mapped I/O

- Solution 1

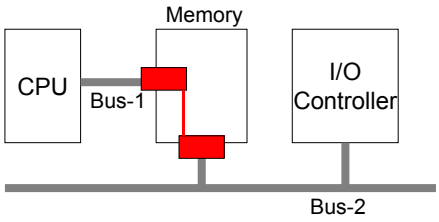
CPU sends an address to main memory over the memory bus. If main memory does not respond (since address range is for I/O), the CPU will put the address on the other buses.



Disadvantages of Memory Mapped I/O

- Solution 2

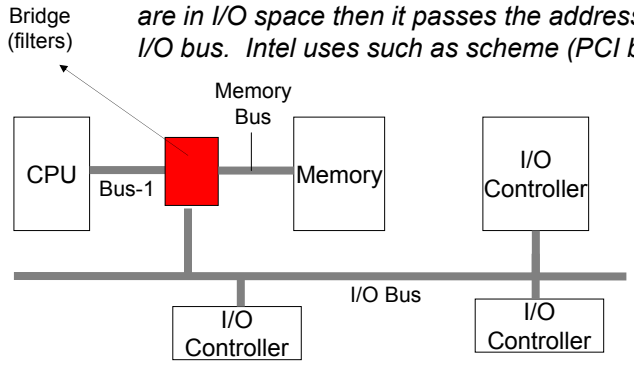
A snooping device on the memory bus can be used to pass all addresses in I/O range to the I/O bus.



Disadvantages of Memory Mapped I/O

- Solution 3

A bridge filters all memory addresses. If addresses are in main memory address range, then it passes the addresses to the memory bus; if addresses are in I/O space then it passes the addresses to the I/O bus. Intel uses such as scheme (PCI bridge)



DMA – Direct Memory Access

- How to exchange data between CPU-memory and device controllers?
  - 1. CPU can request data from device controllers one word (or byte) at a time.
    - CPU is involved in the transfer.
    - Waste of CPU time
  - 2. DMA chip handles the transfer.
    - CPU just initiates the DMA
    - DMA does the rest.
      - COU is not involved during the transfer.

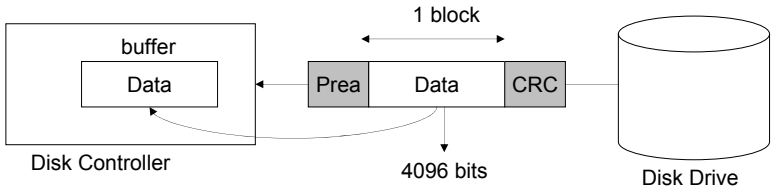
DMA – Direct Memory Access

- DMA controller
  - DMA controller is usually located on the motherboard.
  - It can access the system bus independent of CPU.
  - Contains several registers that can be read and written by the CPU
    - Memory address register
    - Byte count register
    - I/O port to use
    - Direction of Transfer
    - Transfer Unit (byte or word)
    - Burst size (Number of bytes to transfer in one burst)

Control information

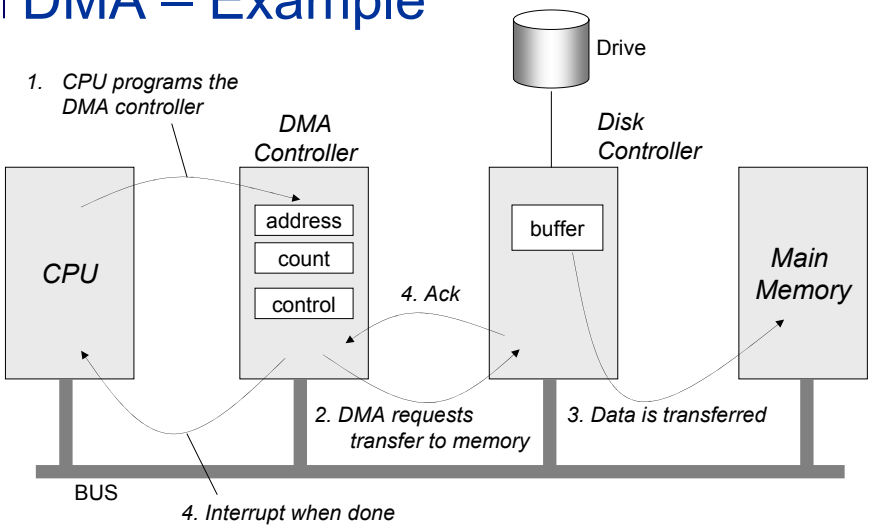
DMA – Example

- How to we read from hard disk.
  - Reads from hard disk occur in blocks (512 bytes).



Disk Controller reads 1 block of data from hard disk, stores it in its buffer, does the error check and if block is error free, it can be transferred to Main memory.

DMA – Example



DMA – Use of the Bus

- Many buses can operate in one of two modes
  - Cycle stealing
  - Burst mode
- In cycle stealing
  - Controller transfer one word to the memory when it seizes Bus cycles from CPU
- In burst mode
  - Controller transfer more than one word (in burst) from controller buffer to memory one after an other. During this time is has the Bus.
    - Acquiring the Bus takes time. This method decreases the affect of the overhead.

DMA – Buffer work in DMA?

- There are two ways of moving words from IO controller to main memory
  - 1. move the work directly from IO controller to memory
  - 2. Move the word first to DMA controller from IO controller. Then move the word from DMA controller to main memory
    - Requires more cycles.
    - But is more flexible: can manage device-to-device copies.

Why I/O Controller Buffering?

- We need to buffer a disk block content in disk controller for two reasons:
  - 1. We must compute the checksum over the block and decide if block has errors or not.
  - 2. Transfer of one block from disk to disk controller happens in a continues manner.
    - But words can not be transferred from disk controller to main memory at the same speed or continuously
    - The BUS is shared and the disk controller can not have the bus for the duration of one block transfer from disk controller to main memory.
    - Therefore we need buffer at disk controller.