Input/Output

Fall 2016

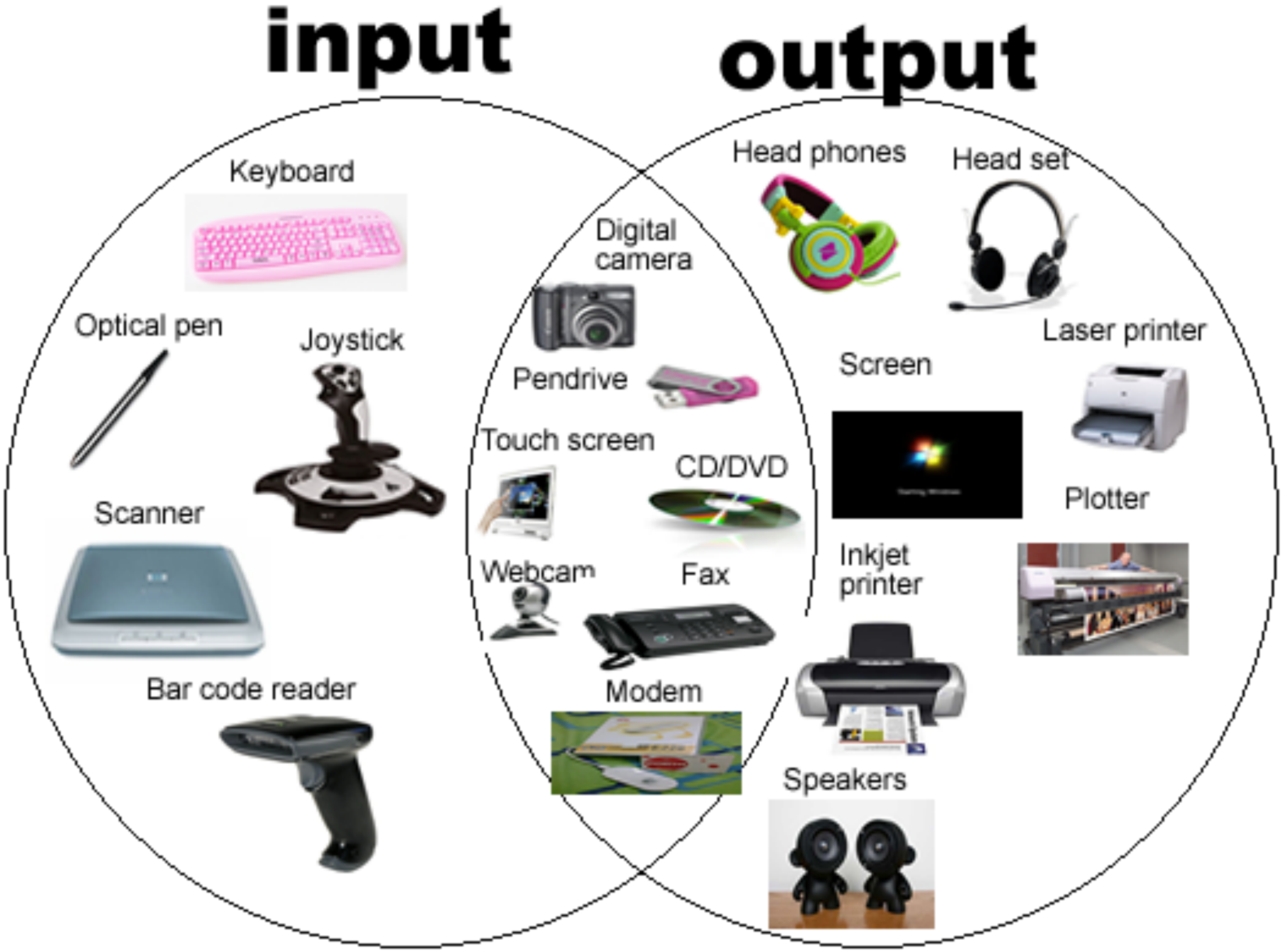
Reading

- Chapter 5 (Sec 5.1), “Modern Operating Systems, Fourth Ed.”, Andrew S. Tanenbaum

Outline

Principles of I/O Hardware

- **I/O Devices**
- Device Controllers
- Addressing Control Registers
- Direct Memory Access
- Interrupts



input

output

Keyboard

Optical pen

Joystick

Scanner

Bar code reader

Digital camera

Pendrive

Touch screen

CD/DVD

Webcam

Fax

Modem

Head phones

Head set

Laser printer

Screen

Plotter

Inkjet printer

Speakers

http://www.bankersguru.org/

Input/Output

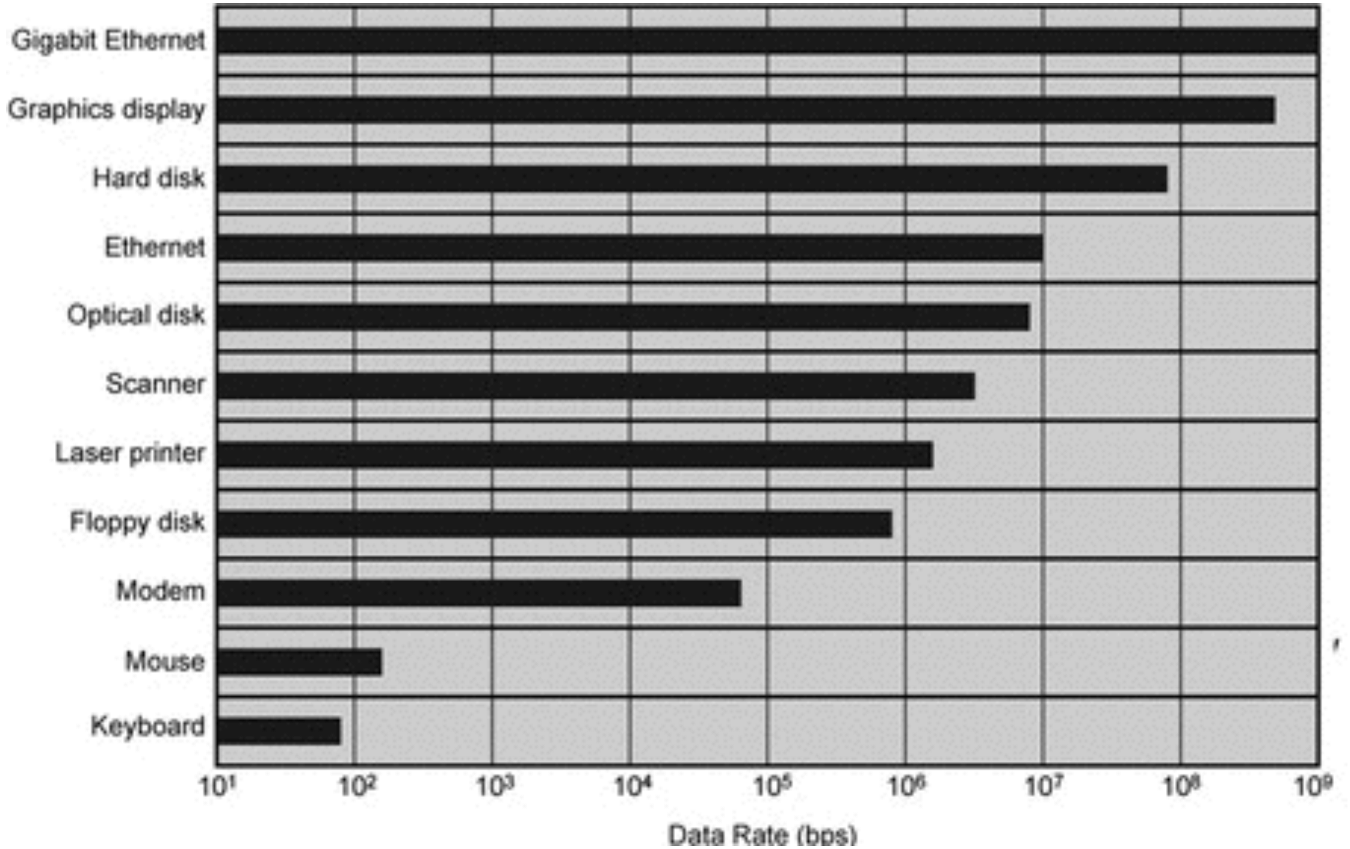
- Operating system abstracts I/O devices
- Operating system provides an interface between the I/O devices and the rest of the system
- Objective: move data from/to the I/O device
- Main operations:
  - commands to the I/O device,
  - catch interrupts, and
  - handle errors

I/O Devices Categories

- Block Devices
  - Stores information in fixed-size blocks
  - Each block has its own address
  - All transfers are in units of one or more entire (consecutive) blocks
  - Example: disk
- Character Devices
  - Delivers or accepts a stream of characters
  - Not addressable or have any seek operation
  - Example: Printers, network interfaces, mouse, display

Data Rates of I/O Devices

- Objective: perform well over many orders of magnitude in data rates.



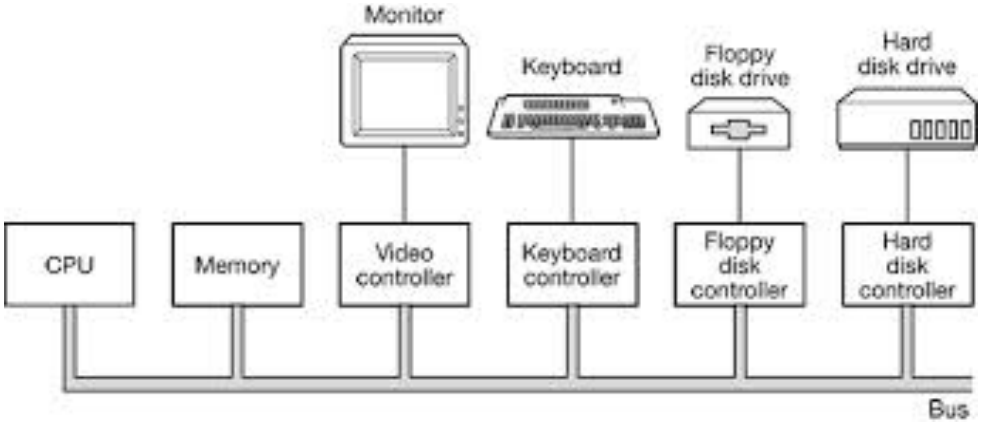
Outline

Principles of I/O Hardware

- I/O Devices
- **Device Controllers**
- Addressing Control Registers
- Direct Memory Access
- Interrupts

I/O Unit Components

- Mechanical component
  - this is the device
- Electronic component (controller or adapter)
  - In PC, a chip on the parentboard or a printed circuit card that can be inserted into a (PCIe) expansion slot
  - Can handle several identical devices



Interface Between the Controller and the Device

- The interface between the controller and the device is often a very low-level one
- Example1: bits streamed from a disk are assembled into a block and any errors are checked and corrected by the controller and copied to memory
- Example 2: LCD display controller reads bytes containing the characters to be displayed from memory and generates the signals to modify the polarization of the backlight for the corresponding pixels in order to write them on screen

Interface Between the Controller and the OS

- Each controller has few registers used to communicate with the processor

how?

- OS controls the device by writing/reading into/from control registers
  - OS writing to the registers: deliver data, accept data, switch itself on or off
  - OS reading registers: learn about device state
- There are data buffers in the device that can be read/written by the OS
  - OS write to the video RAM of the display

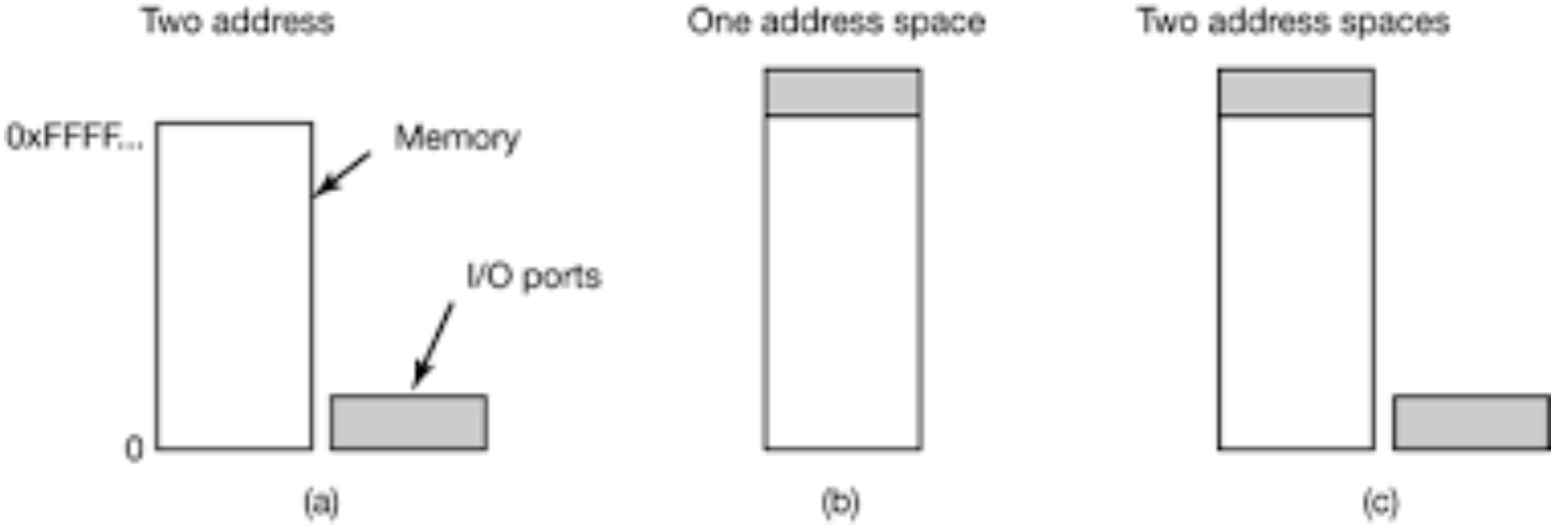
Outline

Principles of I/O Hardware

- I/O Devices
- Device Controllers
- **Addressing Control Registers**
- Direct Memory Access
- Interrupts

Communication between the Controller and the OS

- I/O port space
- Memory-mapped I/O



I/O Port Space

- I/O port space: set of all I/O ports
- Each control register is assigned an I/O port number
- I/O port space is protected. Only OS can access it:
  - Eg: IN REG,PORT
  - Eg: OUT PORT,REG
- Address spaces for memory and I/O are different. These two 4s have different meanings:
  - IN R0,4 (reads content of I/O port 4)
  - MOV R0,4 (reads content of memory address 4)

Memory-Mapped I/O

- Map all the control registers into the memory space
- Each control register is assigned a unique memory address
- In most systems, the assigned addresses are at or near the top of the address space

Hybrid Between I/O Port Space and Memory-Mapped I/O

- Memory-mapped I/O for data buffers
- separate I/O ports for the control registers
- x86 uses this architecture

How Does it Work?

- When the processor wants to read a word from memory or I/O port
  - It puts the address on the bus address line
  - It asserts a control signal on a bus control line
  - It asserts a signal on another control line to tell if it is memory space or I/O space
  - In case on memory mapped I/O, the range of addresses each is using is compared

Pros and Cons of Memory Mapped I/O

- Pros:
  - The device drivers can be completely written in C
  - Easier protection mechanism
  - Use same instructions
- Cons:
  - Memory uses a cache and this does not work with I/O devices (solution: disabling caching)
  - Each memory reference has to be checked to whether it is for I/O control registers or not

Outline

Principles of I/O Hardware

- I/O Devices
- Device Controllers
- Addressing Control Registers
- **Direct Memory Access**
- Interrupts

Direct Memory Access (DMA)

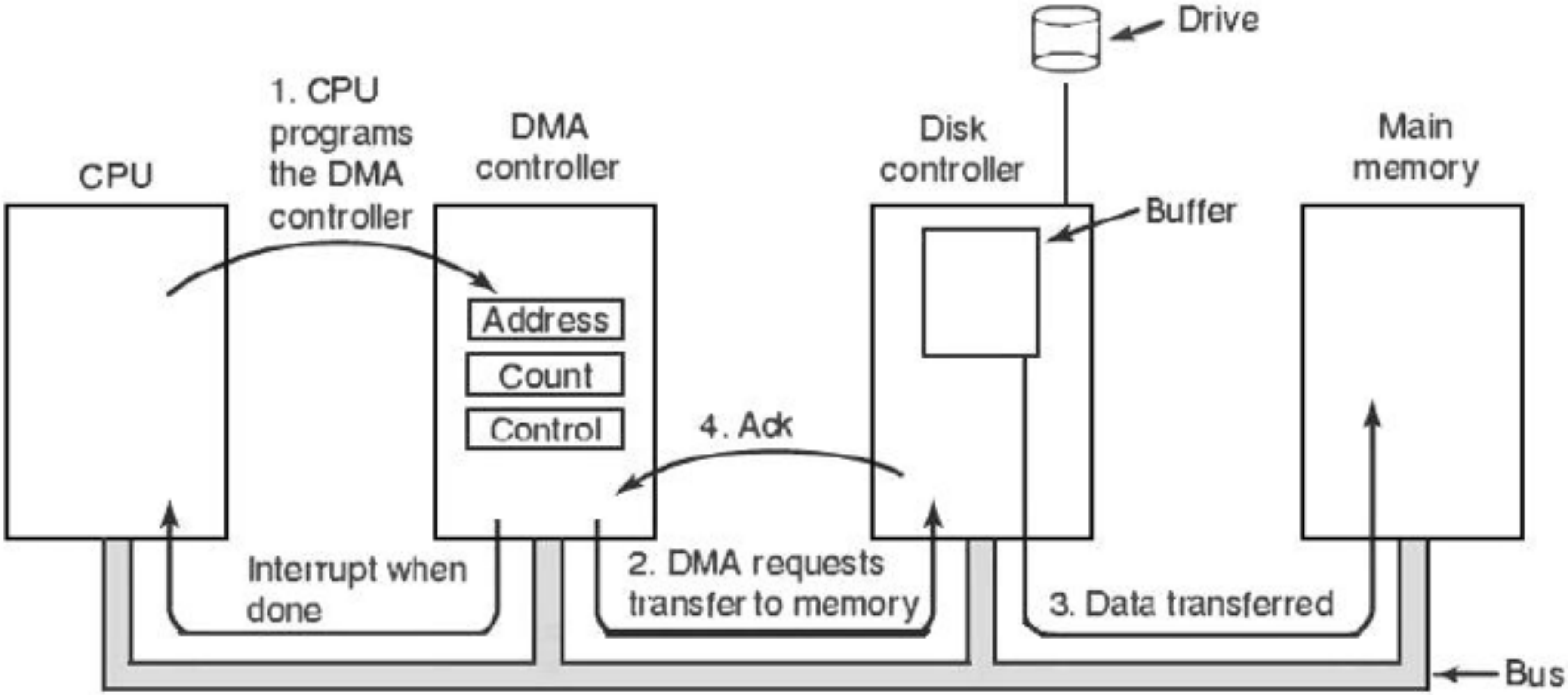
Goal: Accessing controllers for exchanging data

- CPU can request data from an I/O controller one byte at a time
  - Disadvantage: wastes CPU time
- Use DMA
  - Each device has a DMA controller (integrated with the device controller)
  - Or a single DMA controller for multiple devices
  - DMA controller has access to system bus
  - Contains several registers that can be written and read by the CPU

Read Operation — No DMA

- Disk controller reads a block bit by bit into its internal buffer
- Disk controller computes a checksum when the transfer is complete to make sure no errors has occurred
- The controller causes an interrupt and OS runs
- OS read the disk block from the controller’s buffer a byte or a word at a time and store it in memory

Operation of DMA Transfer



Read Operation — DMA

- CPU sets the DMA controller registers with information about the data to transfer and where
- CPU issues a command to the Disk controller to read the data and apply checksum
- DMA controller issues a read request over the bus to the disk controller
- Data is transferred to the memory
- Disk controller acknowledges completing transfer of the data — sent to DMA controller
- DMA interrupts the CPU
- OS starts and it does not have to copy data into memory as it is already there

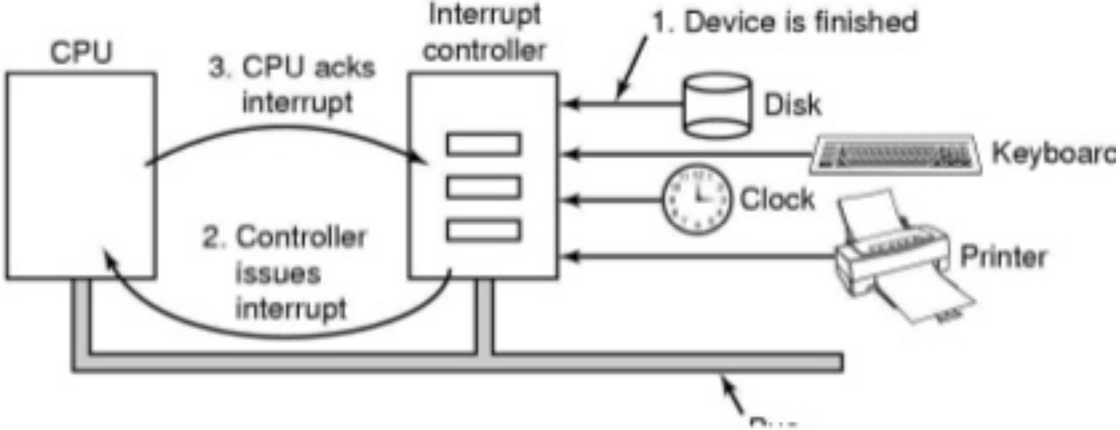
Outline

Principles of I/O Hardware

- I/O Devices
- Device Controllers
- Addressing Control Registers
- Direct Memory Access
- **Interrupts**

Interrupts

- When I/O device has finished, it causes an interrupt by asserting a signal on the bus
- Interrupt controller issues an interrupt by putting the address of the device and interrupts the CPUs
- Address is used as index to the interrupt vector to fetch the program counter
- Acknowledge the interrupt controller



Interrupt Controller

- When the interrupt controller detects an interrupt signal on the bus line, it decides what to do:
  - If no pending interrupts, then handle it directly
  - If other interrupt is in progress, or there is another higher priority interrupt, then ignore for now
- To handle the interrupt, the controller
  - puts a number on the address lines specifying which device wants attention
  - asserts a signal to interrupt the CPU

Thank You
Have Wonderful Future

😊