CS323 Operating Systems
Input/Output I

Yuanyuan Zhou
Lecture 18
3/3/2003

Content of this lecture

- Administrative announcements
- I/O basic concepts
- Interrupts
- DMA
- Memory Mapped I/O
- Summary

Administrative

- MP2
- Midterm1
  - Conflict exam: 3/11, 5-6pm

Review

- protection hardware
- example systems
- fragmentation
- paging policies
- random
- optimal
- FIFO

Overview

- Basic I/O hardware
  - ports, buses, devices and controllers
- I/O Software
  - Interrupt Handlers, Device Driver, Device-Independent Software, User-Space I/O Software
- Real I/O devices
  - Disks, Character-Oriented Terminals, Graphical User Interfaces, Network Terminals,
- Power Management

Devices

- Devices
  - Storage devices (disk, tapes)
  - Transmission devices (network card, modem)
  - Human interface devices (screen, keyboard, mouse)
  - Specialized device (joystick)

Device-Computer/Device-Device Communication

- Physically: via signals over a cable or through air
- Logically: via a connection point - port (e.G., Serial port)
- Multiple devices are connected via a bus
  - A common set of wires and a rigidly defined protocol that specifies a set of messages that can be sent on the wires

Device Controller

- I/O units typically consist of a mechanical component and an electronic component. The electronic component is called the device controller or adapter. Example is a circuit board. The mechanical component is the device itself.
- Interface between controller and device is a very low level interface.
- Example:
  - Disk's controller converts the serial bit stream, coming off the drive into a block of bytes, and performs error correction. The block of bytes is first assembled in a buffer inside the controller. After its checksum has been verified, the error-free block is copied to main memory.
  - Built-in controllers

I/O Controller

- Disk controller - implements the disk side of the protocol that does: bad error mapping, prefetching, buffering, caching
- Controller has registers for data and control
- CPU and controllers communicate via I/O instructions and registers
- Memory-mapped I/O

I/O Ports

- 4 registers - status, control, data-in, data-out
  - **Status** - states whether the current command is completed, byte is available, device has an error, etc
  - **Control** - host determines to start a command or change the mode of a device
  - **Data-in** - host reads to get input
  - **Data-out** - host writes to send output
- Size of registers - 1 to 4 bytes

Host-controller interface: Interrupts

- CPU hardware has the interrupt report line that the CPU senses after executing every instruction
  - device raises an interrupt
  - CPU catches the interrupt and saves the state (e.g., Instruction pointer)
  - CPU dispatches the interrupt handler
  - interrupt handler determines cause, services the device and clears the interrupt
- Why interrupts?
- Real life analogy?

Support for Interrupts

- Need the ability to defer interrupt handling during critical processing
- CPUs have two interrupt request lines
  - non-maskable interrupt (reserved for unrecoverable memory errors)
  - maskable interrupt (can be turned off by cpus before the execution of critical instructions
- Need efficient way to dispatch the proper device
  - Interrupt comes with an address (offset in interrupt vector) that selects a specific interrupt handling
- Need multilevel interrupts - interrupt priority level

Interrupt Handler

- At boot time, OS probes the hardware buses to determine what devices are present and installs corresponding interrupt handlers into the interrupt vector
- During I/O interrupt, controller signals that device is ready

Other Types of Interrupts

- Interrupt mechanisms are used to handle wide variety of exceptions:
  - Division by zero, wrong address
  - Virtual memory paging
  - System calls (software interrupts, trap)
  - Multi-threaded systems

Direct Memory Access (DMA)

- Programmed I/O (PIO)
  - use CPU to watch status bits and feed data into a controller register 1 byte at a time - EXPENSIVE for large transfers
- Direct memory access (DMA)
  - use a special purpose processor, called a DMA controller

Discussion

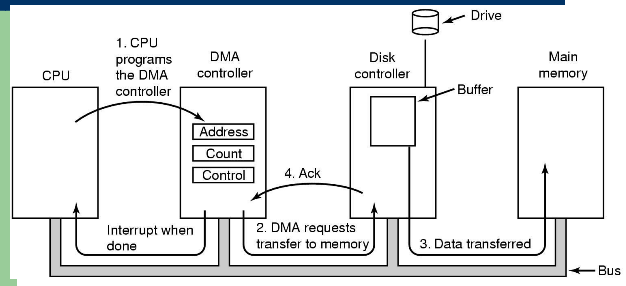
- Why use DMA? The benefits of DMA?

DMA-CPU Protocol

- Host writes a DMA command block into memory
  - pointer to source, pointer to destination, count of bytes to be transferred
- CPU writes the address of this command block to the DMA controller and goes on with other work
- DMA controller proceeds to operate the memory bus directly without help of main CPU

17 3/4/2003 CS 323 - Operating Systems, Yuanyuan Zhou

Direct Memory Access (DMA)



Operation of a DMA transfer

18 3/4/2003 CS 323 - Operating Systems, Yuanyuan Zhou

DMA Issues

- Handshaking between DMA controller and the device controller
- Cycle stealing
  - DMA controller takes away CPU cycles when it uses CPU memory bus, hence blocks the CPU from accessing the memory
- In general DMA controller improves the total system performance

19 3/4/2003 CS 323 - Operating Systems, Yuanyuan Zhou

Device Classification

- Character-stream or block devices
- Sequential or random access devices
- Synchronous or asynchronous devices
- Sharable or dedicated devices
- Speed of operation
- WO, RO, RW devices

20 3/4/2003 CS 323 - Operating Systems, Yuanyuan Zhou

Block Devices

- Block device - transfers blocks of data (e.G., Disk device)
- Commands: read, write, seek (if random access device)
- Memory-mapped files access can be layered on top of block-device drivers

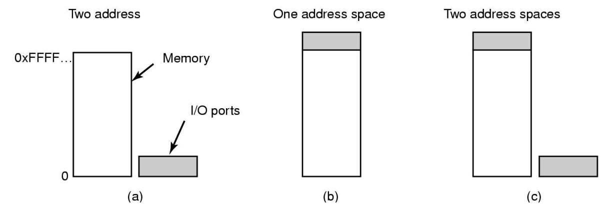
Character Devices

- Character device - transfers byte by byte (e.G., Keyboard)
- Commands: get, put one character
- Libraries can built on top of this device which provide line-at-a time access with buffering and editing services, etc
- Question: can block device be implemented using character device?

Network Devices

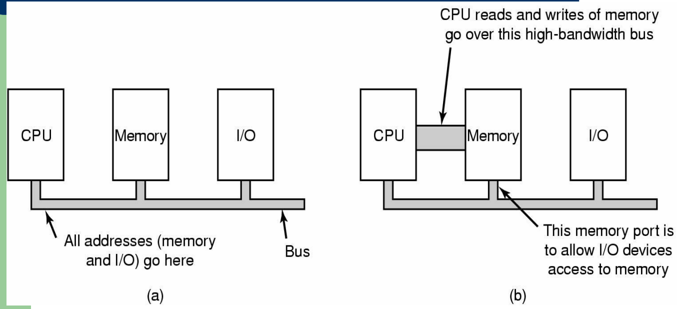
- Network socket interface
- Commands; create socket, connect local socket to remote address, select information from a socket
- Other interfaces: half-duplex pipes, full-duplex fifos, message queues, sockets (UNIX)

Memory-Mapped I/O (1)



- (a) Separate I/O and memory space
- (b) Memory-mapped I/O
- © Hybrid

Memory-Mapped I/O (2)



(a) A single-bus architecture
(b) A dual-bus memory architecture

Reminder

- MP2
- Midterm1