## Part VI
### Input/Output and Interfacing

<table>
<thead>
<tr>
<th>Parts</th>
<th>Chapters</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. Background and Motivation</td>
<td>1. Combinational Digital Circuits</td>
</tr>
<tr>
<td></td>
<td>2. Digital Circuits with Memory</td>
</tr>
<tr>
<td></td>
<td>3. Computer System Technology</td>
</tr>
<tr>
<td></td>
<td>4. Computer Performance</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>II. Instruction-Set Architecture</td>
<td>5. Instructions and Addressing</td>
</tr>
<tr>
<td></td>
<td>6. Procedures and Data</td>
</tr>
<tr>
<td></td>
<td>7. Assembly Language Programs</td>
</tr>
<tr>
<td></td>
<td>8. Instruction-Set Variations</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>III. The Arithmetic/Logic Unit</td>
<td>9. Number Representation</td>
</tr>
<tr>
<td></td>
<td>10. Adders and Simple ALUs</td>
</tr>
<tr>
<td></td>
<td>11. Multipliers and Dividers</td>
</tr>
<tr>
<td></td>
<td>12. Floating-Point Arithmetic</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>IV. Data Path and Control</td>
<td>13. Instruction Execution Steps</td>
</tr>
<tr>
<td></td>
<td>14. Control Unit Synthesis</td>
</tr>
<tr>
<td></td>
<td>15. Pipelined Data Paths</td>
</tr>
<tr>
<td></td>
<td>16. Pipeline Performance Limits</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>V. Memory System Design</td>
<td>17. Main Memory Concepts</td>
</tr>
<tr>
<td></td>
<td>18. Cache Memory Organization</td>
</tr>
<tr>
<td></td>
<td>19. Mass Memory Concepts</td>
</tr>
<tr>
<td></td>
<td>20. Virtual Memory and Paging</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>VI. Input/Output and Interfacing</td>
<td>21. Input/Output Devices</td>
</tr>
<tr>
<td></td>
<td>22. Input/Output Programming</td>
</tr>
<tr>
<td></td>
<td>23. Buses, Links, and Interfacing</td>
</tr>
<tr>
<td></td>
<td>24. Context Switching and Interrupts</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>VII. Advanced Architectures</td>
<td>25. Road to Higher Performance</td>
</tr>
<tr>
<td></td>
<td>26. Vector and Array Processing</td>
</tr>
<tr>
<td></td>
<td>27. Shared-Memory Multiprocessing</td>
</tr>
<tr>
<td></td>
<td>28. Distributed Multicomputing</td>
</tr>
</tbody>
</table>

---

Mar. 2007  Computer Architecture, Input/Output and Interfacing  UCSB  BParhami  Slide 1
About This Presentation

This presentation is intended to support the use of the textbook *Computer Architecture: From Microprocessors to Supercomputers*, Oxford University Press, 2005, ISBN 0-19-515455-X. It is updated regularly by the author as part of his teaching of the upper-division course ECE 154, Introduction to Computer Architecture, at the University of California, Santa Barbara. Instructors can use these slides freely in classroom teaching and for other educational purposes. Any other use is strictly prohibited. © Behrooz Parhami

<table>
<thead>
<tr>
<th>Edition</th>
<th>Released</th>
<th>Revised</th>
<th>Revised</th>
<th>Revised</th>
<th>Revised</th>
</tr>
</thead>
</table>
VI  Input/Output and Interfacing

Effective computer design & use requires awareness of:
- I/O device types, technologies, and performance
- Interaction of I/O with memory and CPU
- Automatic data collection and device actuation

<table>
<thead>
<tr>
<th>Topics in This Part</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 21  Input/Output Devices</td>
</tr>
<tr>
<td>Chapter 22  Input/Output Programming</td>
</tr>
<tr>
<td>Chapter 23  Buses, Links, and Interfacing</td>
</tr>
<tr>
<td>Chapter 24  Context Switching and Interrupts</td>
</tr>
</tbody>
</table>
21 Input/Output Devices

Learn about input and output devices as categorized by:
- Type of data presentation or recording
- Data rate, which influences interaction with system

Topics in This Chapter

<table>
<thead>
<tr>
<th>21.1</th>
<th>Input/Output Devices and Controllers</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.2</td>
<td>Keyboard and Mouse</td>
</tr>
<tr>
<td>21.3</td>
<td>Visual Display Units</td>
</tr>
<tr>
<td>21.4</td>
<td>Hard-Copy Input/Output Devices</td>
</tr>
<tr>
<td>21.5</td>
<td>Other Input/Output Devices</td>
</tr>
<tr>
<td>21.6</td>
<td>Networking of Input/Output Devices</td>
</tr>
</tbody>
</table>
Section 21.1: Introduction

Section 21.2

Section 21.3

Section 21.4

Section 21.5: Other devices

Section 21.6: Networked I/O
# 21.1 Input/Output Devices and Controllers

Table 3.3 Some input, output, and two-way I/O devices.

<table>
<thead>
<tr>
<th>Input type</th>
<th>Prime examples</th>
<th>Other examples</th>
<th>Data rate (b/s)</th>
<th>Main uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>Keyboard, keypad</td>
<td>Music note, OCR</td>
<td>10s</td>
<td>Ubiquitous</td>
</tr>
<tr>
<td>Position</td>
<td>Mouse, touchpad</td>
<td>Stick, wheel, glove</td>
<td>100s</td>
<td>Ubiquitous</td>
</tr>
<tr>
<td>Identity</td>
<td>Barcode reader</td>
<td>Badge, fingerprint</td>
<td>100s</td>
<td>Sales, security</td>
</tr>
<tr>
<td>Sensory</td>
<td>Touch, motion, light</td>
<td>Scent, brain signal</td>
<td>100s</td>
<td>Control, security</td>
</tr>
<tr>
<td>Audio</td>
<td>Microphone</td>
<td>Phone, radio, tape</td>
<td>1000s</td>
<td>Ubiquitous</td>
</tr>
<tr>
<td>Image</td>
<td>Scanner, camera</td>
<td>Graphic tablet</td>
<td>1000s-10⁶s</td>
<td>Photos, publishing</td>
</tr>
<tr>
<td>Video</td>
<td>Camcorder, DVD</td>
<td>VCR, TV cable</td>
<td>1000s-10⁹s</td>
<td>Entertainment</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output type</th>
<th>Prime examples</th>
<th>Other examples</th>
<th>Data rate (b/s)</th>
<th>Main uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>LCD line segments</td>
<td>LED, status light</td>
<td>10s</td>
<td>Ubiquitous</td>
</tr>
<tr>
<td>Position</td>
<td>Stepper motor</td>
<td>Robotic motion</td>
<td>100s</td>
<td>Ubiquitous</td>
</tr>
<tr>
<td>Warning</td>
<td>Buzzer, bell, siren</td>
<td>Flashing light</td>
<td>A few</td>
<td>Safety, security</td>
</tr>
<tr>
<td>Sensory</td>
<td>Braille text</td>
<td>Scent, brain stimulus</td>
<td>100s</td>
<td>Personal assistance</td>
</tr>
<tr>
<td>Audio</td>
<td>Speaker, audiotape</td>
<td>Voice synthesizer</td>
<td>1000s</td>
<td>Ubiquitous</td>
</tr>
<tr>
<td>Image</td>
<td>Monitor, printer</td>
<td>Plotter, microfilm</td>
<td>1000s</td>
<td>Ubiquitous</td>
</tr>
<tr>
<td>Video</td>
<td>Monitor, TV screen</td>
<td>Film/video recorder</td>
<td>1000s-10⁹s</td>
<td>Entertainment</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Two-way I/O</th>
<th>Prime examples</th>
<th>Other examples</th>
<th>Data rate (b/s)</th>
<th>Main uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass storage</td>
<td>Hard/floppy disk</td>
<td>CD, tape, archive</td>
<td>10⁶s</td>
<td>Ubiquitous</td>
</tr>
<tr>
<td>Network</td>
<td>Modem, fax, LAN</td>
<td>Cable, DSL, ATM</td>
<td>1000s-10⁹s</td>
<td>Ubiquitous</td>
</tr>
</tbody>
</table>
Simple Organization for Input/Output

Figure 21.1 Input/output via a single common bus.
Figure 21.2  Input/output via intermediate and dedicated I/O buses (to be explained in Chapter 23).
21.2 Keyboard and Mouse

![Keyboard and Mouse Diagram](slide_image)
Keyboard Switches and Encoding

(a) Mechanical switch with a plunger
(b) Membrane switch
(c) Logical arrangement of keys

Figure 21.3 Two mechanical switch designs and the logical layout of a hex keypad.
Pointing Devices
How a Mouse Works

Figure 21.4  Mechanical and simple optical mice.

(a) Mechanical mouse

- Ball touching the rollers rotates them via friction

(b) Optical mouse

- Photosensor detects crossing of grid lines

Mar. 2007  
UCSB  
Computer Architecture, Input/Output and Interfacing  
B. Parhami  
Slide 12
21.3 Visual Display Units

Figure 21.5 CRT display unit and image storage in frame buffer.
How Color CRT Displays Work

Figure 21.6 The RGB color scheme of modern CRT displays.
Besides hue, saturation is used to affect the color’s appearance (high saturation at the top, low saturation at the bottom)
Flat-Panel Displays

(a) Passive display
(b) Active display

Figure 21.7  Passive and active LCD displays.
Other Display Technologies
21.4 Hard-Copy Input/Output Devices

Figure 21.8 Scanning mechanism for hard-copy input.
Character Formation by Dot Matrices

Figure 21.9  Forming the letter “D” via dot matrices of varying sizes.
Simulating Intensity Levels via Dithering

Forming five gray levels on a device that supports only black and white (e.g., ink-jet or laser printer)

Using the dithering patterns above on each of three colors forms $5 \times 5 \times 5 = 125$ different colors
Simple Dot-Matrix Printer Mechanism
Common Hard-Copy Output Devices

(a) Ink jet printing

(b) Laser printing

Figure 21.10  Ink-jet and laser printers.
How Color Printers Work

The RGB scheme of color monitors is additive: various amounts of the three primary colors are added to form a desired color.

The CMY scheme of color printers is subtractive: various amounts of the three primary colors are removed from white to form a desired color.

To produce a more satisfactory shade of black, the CMYK scheme is often used (K = black).
The CMYK Printing Process

Illusion of full color created with CMYK dots
Color Wheels

Artist’s color wheel, used for mixing paint
Subtractive color wheel, used in printing (CMYK)
Additive color wheel, used for projection

Primary colors appear at center and equally spaced around the perimeter
Secondary colors are midway between primary colors
Tertiary colors are between primary and secondary colors

Source of this and several other slides on color: http://www.devx.com/projectcool/Article/19954/0/
(see also color theory tutorial: http://graphics.kodak.com/documents/Introducing%20Color%20Theory.pdf)
21.5 Other Input/Output Devices
Sensors and Actuators

Collecting info about the environment and other conditions

- Light sensors (photocells)
- Temperature sensors (contact and noncontact types)
- Pressure sensors

Figure 21.11 Stepper motor principles of operation.
21.6 Networking of Input/Output Devices

Figure 21.12 With network-enabled peripherals, I/O is done via file transfers.
Input/Output in Control and Embedded Systems

Figure 21.13  The structure of a closed-loop computer-based control system.
22 Input/Output Programming

Like everything else, I/O is controlled by machine instructions
- I/O addressing (memory-mapped) and performance
- Scheduled vs demand-based I/O: polling vs interrupts

<table>
<thead>
<tr>
<th>Topics in This Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.1 I/O Performance and Benchmarks</td>
</tr>
<tr>
<td>22.2 Input/Output Addressing</td>
</tr>
<tr>
<td>22.3 Scheduled I/O: Polling</td>
</tr>
<tr>
<td>22.4 Demand-Based I/O: Interrupts</td>
</tr>
<tr>
<td>22.5 I/O Data Transfer and DMA</td>
</tr>
<tr>
<td>22.6 Improving I/O Performance</td>
</tr>
</tbody>
</table>
22.1   I/O Performance and Benchmarks

Example 22.1: The I/O wall

An industrial control application spent 90% of its time on CPU operations when it was originally developed in the early 1980s. Since then, the CPU component has been upgraded every 5 years, but the I/O components have remained the same. Assuming that CPU performance improved tenfold with each upgrade, derive the fraction of time spent on I/O over the life of the system.

Solution

Apply Amdahl’s law with 90% of the task speeded up by factors of 10, 100, 1000, and 10000 over a 20-year period. In the course of these upgrades the running time has been reduced from the original 1 to $0.1 + 0.9/10 = 0.19$, $0.109$, $0.1009$, and $0.10009$, making the fraction of time spent on input/output 52.6, 91.7, 99.1, and 99.9%, respectively. The last couple of CPU upgrades did not really help.
Types of Input/Output Benchmark

**Supercomputer I/O benchmarks**

- Reading large volumes of input data
- Writing many snapshots for checkpointing
- Saving a relatively small set of results
- I/O data throughput, in MB/s, is important

**Transaction processing I/O benchmarks**

- Huge database, but each transaction fairly small
- A handful (2-10) of disk accesses per transaction
- I/O rate (disk accesses per second) is important

**File system I/O benchmarks**

- File creation, directory management, indexing, . . .
- Benchmarks are usually domain-specific
22.2 Input/Output Addressing

Memory location (hex address)

<table>
<thead>
<tr>
<th>Hex Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xfffff0000</td>
<td>Keyboard control</td>
</tr>
<tr>
<td>0xfffff0004</td>
<td>Keyboard data</td>
</tr>
<tr>
<td>0xfffff0008</td>
<td>Display control</td>
</tr>
<tr>
<td>0xfffff000c</td>
<td>Display data</td>
</tr>
</tbody>
</table>

32-bit device registers

Figure 22.1 Control and data registers for keyboard and display unit in MiniMIPS.
Hardware for I/O Addressing

Figure 22.2  Addressing logic for an I/O device controller.
Data Input from Keyboard

Example 22.2

Write a sequence of MiniMIPS assembly language instructions to make the program wait until the keyboard has a symbol to transmit and then read the symbol into register $v0.

Solution

The program must continually examine the keyboard control register, ending its “busy wait” when the R bit has been asserted.

```
lui $t0,0xffff   # put 0xffff0000 in $t0
idle: lw $t1,0($t0)   # get keyboard’s control word
andi $t1,$t1,0x0001   # isolate the LSB (R bit)
beq $t1,$zero,idle   # if not ready (R = 0), wait
lw $v0,4($t0)   # retrieve data from keyboard
```

This type of input is appropriate only if the computer is waiting for a critical input and cannot continue in the absence of such input.
Data Output to Display Unit

Example 22.3

Write a sequence of MiniMIPS assembly language instructions to make the program wait until the display unit is ready to accept a new symbol and then write the symbol from $a0 to the display unit.

Solution

The program must continually examine the display unit’s control register, ending its “busy wait” when the R bit has been asserted.

```
lui $t0,0xffff  # put 0xffff0000 in $t0
idle: lw  $t1,8($t0)  # get display’s control word
andi $t1,$t1,0x0001  # isolate the LSB (R bit)
beq $t1,$zero,idle  # if not ready (R = 0), wait
sw  $a0,12($t0)   # supply data to display unit
```

This type of output is appropriate only if we can afford to have the CPU dedicated to data transmission to the display unit.
22.3 Scheduled I/O: Polling

Examples 22.4, 22.5, 22.6

What fraction of a 1 GHz CPU’s time is spent polling the following devices if each polling action takes 800 clock cycles?

- Keyboard must be interrogated at least 10 times per second
- Floppy sends data 4 bytes at a time at a rate of 50 KB/s
- Hard drive sends data 4 bytes at a time at a rate of 3 MB/s

Solution

For keyboard, divide the number of cycles needed for 10 interrogations by the total number of cycles available in 1 second:

\[
\frac{10 \times 800}{10^9} \approx 0.001\%
\]

The floppy disk must be interrogated 50K/4 = 12.5K times per sec

\[
\frac{12.5K \times 800}{10^9} \approx 1\%
\]

The hard disk must be interrogated 3M/4 = 750K times per sec

\[
\frac{750K \times 800}{10^9} \approx 60\%
\]
22.4 Demand-Based I/O: Interrupts

Example 22.7

Consider the disk in Example 22.6 (transferring 4 B chunks of data at 3 MB/s when active). Assume that the disk is active 5% of the time. The overhead of interrupting the CPU and performing the transfer is 1200 clock cycles. What fraction of a 1 GHz CPU’s time is spent attending to the hard disk drive?

Solution

When active, the hard disk produces 750K interrupts per second

\[0.05 \times (750K \times 1200)/10^9 \approx 4.5\%\] (compare with 60% for polling)

Note that even though the overhead of interrupting the CPU is higher than that of polling, because the disk is usually idle, demand-based I/O leads to better performance.
Interrupt Handling

Upon detecting an interrupt signal, provided the particular interrupt or interrupt class is not masked, the CPU acknowledges the interrupt (so that the device can deassert its request signal) and begins executing an interrupt service routine.

1. Save the CPU state and call the interrupt service routine.
2. Disable all interrupts.
3. Save minimal information about the interrupt on the stack.
4. Enable interrupts (or at least higher priority ones).
5. Identify cause of interrupt and attend to the underlying request.
6. Restore CPU state to what existed before the last interrupt.
7. Return from interrupt service routine.

The capability to handle nested interrupts is important in dealing with multiple high-speed I/O devices.
22.5 I/O Data Transfer and DMA

Figure 22.3 DMA controller shares the system or memory bus with the CPU.
DMA Operation

(a) DMA transfer in one continuous burst

(b) DMA transfer in several shorter bursts

Figure 22.4 DMA operation and the associated transfers of bus control.
22.6 Improving I/O Performance

Example 22.9: Effective I/O bandwidth from disk

Consider a hard disk drive with 512 B sectors, average access latency of 10 ms, and peak throughput of 10 MB/s. Plot the variation of the effective I/O bandwidth as the unit of data transfer (block) varies in size from 1 sector (0.5 KB) to 1024 sectors (500 KB).

Solution

![Graph showing effective I/O bandwidth vs. block size]

Figure 22.5
Distributed Input/Output

Figure 22.6 Example configuration for the Infiniband distributed I/O.
23 Buses, Links, and Interfacing

Shared links or buses are common in modern computers:
- Fewer wires and pins, greater flexibility & expandability
- Require dealing with arbitration and synchronization

<table>
<thead>
<tr>
<th>Topics in This Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>23.1 Intra- and Intersystem Links</td>
</tr>
<tr>
<td>23.2 Buses and Their Appeal</td>
</tr>
<tr>
<td>23.3 Bus Communication Protocols</td>
</tr>
<tr>
<td>23.4 Bus Arbitration and Performance</td>
</tr>
<tr>
<td>23.5 Basics of Interfacing</td>
</tr>
<tr>
<td>23.6 Interfacing Standards</td>
</tr>
</tbody>
</table>
### 23.1 Intra- and Intersystem Links

Trench with via

1. Etched and insulated
2. Coated with copper
3. Excess copper removed

(a) Cross section of layers

(b) 3D view of wires on multiple metal layers

---

**Figure 23.1** Multiple metal layers provide intrasystem connectivity on microchips or printed-circuit boards.
Multiple Metal Layers on a Chip or PC Board

Active elements and their connectors

Modern chips have 8-9 metal layers

Upper layers carry longer wires as well as those that need more power

Cross section of metal layers
Intersystem Links

(a) RS-232  
(b) Ethernet  
(c) ATM

Figure 23.2  Example intersystem connectivity schemes.

Figure 23.3  RS-232 serial interface 9-pin connector.

DTR: data terminal ready  
Transmit data  
Receive data  
Signal ground  
CTS: clear to send  
RTS: request to send  
DSR: data set ready
Figure 23.4  Commonly used communication media for intersystem connections.
Comparing Intersystem Links

Table 23.1  Summary of three interconnection schemes.

<table>
<thead>
<tr>
<th>Interconnection properties</th>
<th>RS-232</th>
<th>Ethernet</th>
<th>ATM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum segment length (m)</td>
<td>10s</td>
<td>100s</td>
<td>1000s</td>
</tr>
<tr>
<td>Maximum network span (m)</td>
<td>10s</td>
<td>100s</td>
<td>Unlimited</td>
</tr>
<tr>
<td>Bit rate (Mb/s)</td>
<td>Up to 0.02</td>
<td>10/100/1000</td>
<td>155-2500</td>
</tr>
<tr>
<td>Unit of transmission (B)</td>
<td>1</td>
<td>100s</td>
<td>53</td>
</tr>
<tr>
<td>Typical end-to-end latency (ms)</td>
<td>&lt; 1</td>
<td>10s-100s</td>
<td>100s</td>
</tr>
<tr>
<td>Typical application domain</td>
<td>Input/Output</td>
<td>LAN</td>
<td>Backbone</td>
</tr>
<tr>
<td>Transceiver complexity or cost</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>
23.2 Buses and Their Appeal

Point-to-point connections between $n$ units require $n(n - 1)$ channels, or $n(n - 1)/2$ bidirectional links; that is, $O(n^2)$ links.

Bus connectivity requires only one input and one output port per unit, or $O(n)$ links in all.
Bus Components and Types

Figure 23.5 The three sets of lines found in a bus.

A typical computer may use a dozen or so different buses:

1. Legacy Buses: PC bus, ISA, RS-232, parallel port
2. Standard buses: PCI, SCSI, USB, Ethernet
3. Proprietary buses: for specific devices and max performance
23.3 Bus Communication Protocols

Figure 23.6 Synchronous bus with fixed-latency devices.

Figure 23.7 Handshaking on an asynchronous bus for an input operation (e.g., reading from memory).
Example Bus Operation

**Figure 23.8** I/O read operation via PCI bus.
23.4 Bus Arbitration and Performance

Figure 23.9 General structure of a centralized bus arbiter.
Daisy Chaining

Figure 23.9  Daisy chaining allows a small centralized arbiter to service a large number of devices that use a shared resource.
23.5 Basics of Interfacing

Figure 23.11 Wind vane supplying an output voltage in the range 0-5 V depending on wind direction.
### 23.6 Interfacing Standards

**Table 23.2** Summary of four standard interface buses.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>PCI</th>
<th>SCSI</th>
<th>FireWire</th>
<th>USB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of bus</td>
<td>Backplane</td>
<td>Parallel I/O</td>
<td>Serial I/O</td>
<td>Serial I/O</td>
</tr>
<tr>
<td>Standard designation</td>
<td>PCI</td>
<td>ANSI X3.131</td>
<td>IEEE 1394</td>
<td>USB 2.0</td>
</tr>
<tr>
<td>Typical application domain</td>
<td>System</td>
<td>Fast I/O</td>
<td>Fast I/O</td>
<td>Low-cost I/O</td>
</tr>
<tr>
<td>Bus width (data bits)</td>
<td>32-64</td>
<td>8-32</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Peak bandwidth (MB/s)</td>
<td>133-512</td>
<td>5-40</td>
<td>12.5-50</td>
<td>0.2-15</td>
</tr>
<tr>
<td>Maximum number of devices</td>
<td>1024*</td>
<td>7-31#</td>
<td>63</td>
<td>127$</td>
</tr>
<tr>
<td>Maximum span (m)</td>
<td>&lt; 1</td>
<td>3-25</td>
<td>4.5-72$</td>
<td>5-30$</td>
</tr>
<tr>
<td>Arbitration method</td>
<td>Centralized</td>
<td>Self-select</td>
<td>Distributed</td>
<td>Daisy chain</td>
</tr>
<tr>
<td>Transceiver complexity or cost</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
<td>Low</td>
</tr>
</tbody>
</table>

**Notes:**  
* 32 per bus segment;  
# One less than bus width;  
$ With hubs (repeaters)
Standard Connectors

Figure 23.12  USB connectors and connectivity structure.

- Pin 1: +5V DC
- Pin 2: Data –
- Pin 3: Data +
- Pin 4: Ground

Max cable length: 5m

Figure 23.13  IEEE 1394 (FireWire) connector. The same connector is used at both ends.

- Pin 1: 8-40V DC, 1.5 A
- Pin 2: Ground
- Pin 3: Twisted pair B –
- Pin 4: Twisted pair B +
- Pin 5: Twisted pair A –
- Pin 6: Twisted pair A +
- Shell: Outer shield
24  Context Switching and Interrupts

OS initiates I/O transfers and awaits notification via interrupts
• When an interrupt is detected, the CPU switches context
• Context switch can also be used between users/threads

<table>
<thead>
<tr>
<th>Topics in This Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>24.1  System Calls for I/O</td>
</tr>
<tr>
<td>24.2  Interrupts, Exceptions, and Traps</td>
</tr>
<tr>
<td>24.3  Simple Interrupt Handling</td>
</tr>
<tr>
<td>24.4  Nested Interrupts</td>
</tr>
<tr>
<td>24.5  Types of Context Switching</td>
</tr>
<tr>
<td>24.6  Threads and Multithreading</td>
</tr>
</tbody>
</table>
24.1 System Calls for I/O

Why the user must be isolated from details of I/O operations

Protection: User must be barred from accessing some disk areas
Convenience: No need to learn details of each device’s operation
Efficiency: Most users incapable of finding the best I/O scheme

I/O abstraction: grouping of I/O devices into a small number of generic types so as to make the I/O device-independent

Character stream I/O: get(●), put(●) – e.g., keyboard, printer
Block I/O: seek(●), read(●), write(●) – e.g., disk
Network Sockets: create socket, connect, send/receive packet
Clocks or timers: set up timer (get notified via an interrupt)
24.2 Interrupts, Exceptions, and Traps

<table>
<thead>
<tr>
<th>Interrupt</th>
<th>Both general term for any diversion and the I/O type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exception</td>
<td>Caused by an illegal operation (often unpredictable)</td>
</tr>
<tr>
<td>Trap</td>
<td>AKA “software interrupt” (preplanned and not rare)</td>
</tr>
</tbody>
</table>

- **Interrupt**: Both general term for any diversion and the I/O type
- **Exception**: Caused by an illegal operation (often unpredictable)
- **Trap**: AKA “software interrupt” (preplanned and not rare)

**Figure 24.1** The notions of interrupts and nested interrupts.
24.3 Simple Interrupt Handling

Acknowledge the interrupt by asserting the IntAck signal
Notify the CPU’s next-address logic that an interrupt is pending
Set the interrupt mask so that no new interrupt is accepted

Figure 24.2 Simple interrupt logic for the single-cycle MicroMIPS.
Interrupt Timing

Figure 24.3    Timing of interrupt request and acknowledge signals.
Next-Address Logic with Interrupts Added

Figure 24.4 Part of the next-address logic for single-cycle MicroMIPS, with an interrupt capability added (compare with the lower left part of Figure 13.4).
24.4 Nested Interrupts

Figure 24.6 Example of nested interrupts.
24.5 Types of Context Switching

(a) Human multitasking
(b) Computer multitasking

Figure 24.7  Multitasking in humans and computers.
24.6 Threads and Multithreading

(a) Task graph of a program

(b) Thread structure of a task

Figure 24.8 A program divided into tasks (subcomputations) or threads.
Multithreaded Processors

Figure 24.9 Instructions from multiple threads as they make their way through a processor’s execution pipeline.