Interfacing processors and peripherals

- I/O Design affected by many factors (expandability, resilience)
- Performance index:
  - access latency, throughput, etc
- Factors that affect Performance:
  - connection between devices and the system
  - the memory hierarchy
  - the operating system
- A variety of different users (e.g., banks, supercomputers, engineers)
Interfacing processors and peripherals

I/O

- Important but neglected

“The difficulties in assessing and designing I/O systems have often relegated I/O to second class status”

“courses in every aspect of computing, from programming to computer architecture often ignore I/O or give it scanty coverage”

“textbooks leave the subject to near the end, making it easier for students and instructors to skip it!”
Distinction in MB

- For memory, MB = Mbytes = 2^{20} bytes
- For I/O, some people may define MB = 10^6 bytes.
  - So people quote “DOS can only support 504MB hard disc”, meaning 504 * 10^6 bytes = 528 * 2^{20} bytes.

I/O devices

- Very diverse devices
  - Behavior
    - Input
    - Output
    - Storage (can be re-read and usually rewritten): An example of non-storage but inputable and outputable devices is the network adapter.
  - Partner (who is at the other end?)
    - Human
    - Machine
  - Data rate
    - Peak rate
### I/O devices

<table>
<thead>
<tr>
<th>Device</th>
<th>Behavior</th>
<th>Partner</th>
<th>Data rate (Mbit/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard</td>
<td>input</td>
<td>human</td>
<td>0.0001</td>
</tr>
<tr>
<td>Mouse</td>
<td>input</td>
<td>human</td>
<td>0.0038</td>
</tr>
<tr>
<td>Voice input</td>
<td>input</td>
<td>human</td>
<td>0.2640</td>
</tr>
<tr>
<td>Sound input</td>
<td>input</td>
<td>machine</td>
<td>3.0000</td>
</tr>
<tr>
<td>Scanner</td>
<td>input</td>
<td>human</td>
<td>3.2000</td>
</tr>
<tr>
<td>Voice output</td>
<td>output</td>
<td>human</td>
<td>0.2640</td>
</tr>
<tr>
<td>Sound output</td>
<td>output</td>
<td>human</td>
<td>8.0000</td>
</tr>
<tr>
<td>Line printer</td>
<td>output</td>
<td>human</td>
<td>3.2000</td>
</tr>
<tr>
<td>Laser printer</td>
<td>output</td>
<td>human</td>
<td>3.2000</td>
</tr>
<tr>
<td>Graphics display</td>
<td>output</td>
<td>human</td>
<td>800.0000-8000.0000</td>
</tr>
<tr>
<td>Modem</td>
<td>input or output</td>
<td>machine</td>
<td>0.0160-0.0640</td>
</tr>
<tr>
<td>Network/LAN</td>
<td>input or output</td>
<td>machine</td>
<td>100.0000-1000.0000</td>
</tr>
<tr>
<td>Network/Wireless LAN</td>
<td>input or output</td>
<td>machine</td>
<td>11.0000-54.0000</td>
</tr>
<tr>
<td>Optical disk</td>
<td>storage</td>
<td>machine</td>
<td>80.0000</td>
</tr>
<tr>
<td>Magnetic tape</td>
<td>storage</td>
<td>machine</td>
<td>32.0000</td>
</tr>
<tr>
<td>Magnetic disk</td>
<td>storage</td>
<td>machine</td>
<td>240.0000-2560.0000</td>
</tr>
</tbody>
</table>

* Note that M = 10^6 in the above table. So for example 10 Mbit/sec = 10,000,000 bits/sec.

---

### I/O example: Disk driver

- **To access data:**
  - *seek:* position head over the proper track (3 to 14 ms. avg.)
  - *rotational latency:* wait for desired sector (.5 / RPM)
    - 0.5 rotation/3600 RPM = 8.3 ms
  - *transfer:* grab the data (one or more sectors) 30 to 80 MB/sec
  - *Control time:* The time added due to the disk controller.
*All the tracks under the heads at a given point on all surfaces are named cylinder.*

---

**I/O example: Disk driver**

- **Traditional problems of harddisk**
  - Head crash, when the power is abruptly interrupted or the harddisk drive is jarred.
  - Solution: To always (automatically) park the head at a fixed location, which is called safe-landing zone. *(auto-parking head)*
  - Another (cheaper) solution: To use software parking, which force the head to park on the innermost track.
### I/O example: Disk driver

#### Format of Hard disk

<table>
<thead>
<tr>
<th>Type</th>
<th>Cylinder</th>
<th>Head</th>
<th>Sector</th>
<th>Land</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>306</td>
<td>4</td>
<td>17</td>
<td>305</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>615</td>
<td>4</td>
<td>17</td>
<td>615</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>615</td>
<td>6</td>
<td>17</td>
<td>615</td>
<td>32</td>
</tr>
<tr>
<td>4</td>
<td>940</td>
<td>8</td>
<td>17</td>
<td>940</td>
<td>65</td>
</tr>
<tr>
<td>36</td>
<td>1024</td>
<td>14</td>
<td>17</td>
<td>1023</td>
<td>124</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
306 \times 4 \times 17 \times 512 &= 10404K = 10.16M = 10.64 \text{ Mega} \\
615 \times 4 \times 17 \times 512 &= 20910K = 20.42M = 21.14 \text{ Mega} \\
615 \times 6 \times 17 \times 512 &= 31265K = 30.63M = 32.12 \text{ Mega} \\
940 \times 8 \times 17 \times 512 &= 63,924K = 62.42M = 65.45 \text{ Mega} \\
1024 \times 14 \times 17 \times 512 &= 119M = 124.78 \text{ Mega}
\end{align*}
\]

#### Some notes on hard driver

- Now each track is about 60-200 sectors.
- Originally, a sector consists of 512 bytes and is the smallest access unit for hard driver; but later due to the technology advance, sectors are subdivided to the next-generation-minimum-access-units, blocks. Each block is 512 bytes in size.
  - Logical Block Access (LBA): Disk drive is addressed by blocks.
- Originally, every track contain the same number of sectors. But now outer tracks have more sectors than inner tracks. (Zone Bit Recording Technique or Multiple Zone Recording Technique)
  - Note that in such case, the head moves faster on the outer tracks!
I/O example: Disk driver

- Some notes on hard driver
  - Some hard drive also equips with cache.
  - So the “advertised” transfer rate may be the cache access rate!

Example.

Please calculate the average time to read a 512-byte sector for a disk with 6ms average seek time, 10,000 RPM, 50MB/sec transfer rate and 0.2ms controller overhead.

Answer:

Average seek time + average rotational delay + transfer time + controller overhead

= 6 msec + 0.5/(10000/60) sec + 0.5 KB/ (50MB/sec) + 0.2 msec

= 6ms + 3ms + 0.01ms + 0.2ms = 9.2ms (~ 0.0546MB/sec)
Sample characteristics of disks

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Seagate ST373453</th>
<th>Seagate ST3200822</th>
<th>Seagate ST94811A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk diameter (inches)</td>
<td>3.50</td>
<td>3.50</td>
<td>2.50</td>
</tr>
<tr>
<td>Formatted data capacity (GB)</td>
<td>73.4</td>
<td>200</td>
<td>40.0</td>
</tr>
<tr>
<td>Number of disk surfaces (heads)</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Rotation speed (RPM)</td>
<td>15,000</td>
<td>7200</td>
<td>5400</td>
</tr>
<tr>
<td>Internal disk cache size (MB)</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>External interface, bandwidth (MB/sec)</td>
<td>Ultra320 SCSI, 320</td>
<td>Serial ATA, 150</td>
<td>ATA, 1000</td>
</tr>
<tr>
<td>Sustained transfer rate (MB/sec)</td>
<td>57-86</td>
<td>32-58</td>
<td>34</td>
</tr>
<tr>
<td>Minimum seek (read/write)(ms)</td>
<td>0.2/0.4</td>
<td>1.0/1.2</td>
<td>1.5/2.0</td>
</tr>
<tr>
<td>Average seek (read/write)(ms)</td>
<td>3.6/3.9</td>
<td>8.5/0.5</td>
<td>12.0/14.0</td>
</tr>
<tr>
<td>Mean time to failure (MTTF) hours</td>
<td>1,200,000@25°C</td>
<td>600,000@25°C</td>
<td>330,0000@25°C</td>
</tr>
<tr>
<td>Warranty (years)</td>
<td>5</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>Nonrecoverable read errors per bits read</td>
<td>&lt;1 per 10¹⁰</td>
<td>&lt;1 per 10⁹</td>
<td>&lt;1 per 10⁹</td>
</tr>
<tr>
<td>Price in 2004, $/GB</td>
<td>$5</td>
<td>$0.5</td>
<td>$2.5</td>
</tr>
</tbody>
</table>

Buses: Connecting I/O devices to processor and memory

- In concept, all the peripherals share the (address and data) bus with the motherboard.

![Diagram of buses](image-url)
Example: Pentium 4 system organization

Advantages of bus organization

- **Versatility:**
  - New devices can be added easily
  - Peripherals can be moved between computer systems that use the same bus standard

- **Low Cost:**
  - A single set of wires is shared in multiple ways

- **Manage complexity by partitioning the design**
Disadvantages of bus organization

- It creates a communication bottleneck
  - The bandwidth of that bus can limit the maximum I/O throughput
    - E.g., an I/O device may be able to provide 40MB/sec data transfer rate, but the attached bus is only capable of transceiving in 8MB/sec.

- The maximum bus speed is largely limited by:
  - The length of the bus
  - The number of devices on the bus
  - The need to support a range of devices with:
    - Widely varying latencies
    - Widely varying data transfer rates

Buses: Connecting I/O devices to processor and memory

- Summary on potential difficulties in bus design:
  - may be bottleneck
  - length of the bus
  - number of devices
  - tradeoffs (buffers for higher bandwidth increases latency)
  - support for devices with varying characteristics
  - cost
Buses: Connecting I/O devices to processor and memory

- Control lines:
  - Signal requests and acknowledgments
  - Indicate what type of information is on the data lines
- Data lines carry information between the source and the destination:
  - Data and Addresses
  - Complex commands

Buses: Connecting I/O devices to processor and memory

- Terminologies
  - **Memory read**: CPU reads data in from memory (DRAM)
  - **Memory write**: CPU writes data out to memory (DRAM)
  - **I/O read or I/O input**: CPU reads data in from I/O devices (possibly to memory or CPU registers)
  - **I/O write or I/O output**: CPU writes data out to I/O devices (possibly from memory or CPU registers)
Types of buses

- Processor-Memory Bus (design specific)
  - Short and high speed
  - Only need to match the memory system
  - Connects directly to the processor
  - Optimized for cache block transfers

- I/O Bus (industry standard, e.g., SCSI)
  - Usually is lengthy and slower
  - Need to match a wide range of I/O devices
  - Connect to the processor-memory bus or backplane bus

- Backplane Bus (standard or proprietary)
  - Backplane: an interconnection structure within the chassis (backplane)
  - Allow processors, memory, and I/O devices to coexist
  - Cost advantage: one bus for all components

A computer system with single bus:
Backplane bus

- A single bus (the backplane bus) is used for:
  - Processor to memory communication
  - Communication between I/O devices and memory

- Advantages: Simple and low cost
- Disadvantages: slow (the processor-memory transfer may be capable of a much faster rate than processor-I/O transfer) and the bus can become a major bottleneck
- Example: IBM PC-AT
A two-bus system

- **I/O buses tap into the processor-memory bus via bus adaptors:**
  - Processor-memory bus: mainly for processor-memory traffic
  - I/O buses: provide expansion slots for I/O devices
- **Example: Apple Macintosh-II**
  - NuBus: Processor, memory, and a few selected I/O devices
  - SCCI Bus: the rest of the I/O devices

A three-bus system

- **A small number of backplane buses tap into the processor-memory bus**
  - Processor-memory bus is used for processor memory traffic
  - I/O buses are connected to the backplane bus
- **Advantage: loading on the processor bus is greatly reduced**
Synchronous and asynchronous bus

- **Synchronous Bus:**
  - Includes a clock in the control lines
  - A fixed protocol for communication that is relative to the clock
  - Advantage: involves very little logic and can run very fast
  - Disadvantages:
    - Every device on the bus must run at the same clock rate
    - To avoid clock skew, the connection length cannot be long if they are fast

- **Asynchronous Bus:**
  - It is not clocked
  - It can accommodate a wide range of devices
  - It can be lengthened without worrying about clock skew
  - It requires a handshaking protocol (sometimes, being implemented as a FSM)

Clock skew: The supposed synchronous clocks have difference in clock arrivals.

- **No clock skew**

Before the next clocks come in.

- Clock arrives at time $t$
- Component delay $t_d$
- $A_n \rightarrow A_{n+1}$
- $A_{n+1} \rightarrow A_{n+2}$

After the next clocks come in.

- Clock arrives at time $t$
- Component delay $t_d$
- $A_n \rightarrow A_{n+1}$
- $A_{n+1} \rightarrow A_{n+2}$
Clock skew: The supposed synchronous clocks have difference in clock arrivals.

- With clock skew

Before the next clocks come in.

- Clock arrives at time $t$

After the next clocks come in.

- Clock arrives at time $t + \Delta t$, where $\Delta t > t_d$

Component delay $t_d$

Handshaking example for asynchronous transfer
Increasing the bus bandwidth

- Separate versus multiplexed address and data lines:
  - Address and data can be transmitted in one bus cycle if separate address and data lines are available
  - Cost: (a) more bus lines, (b) increased complexity

- Data bus width:
  - By increasing the width of the data bus, transfers of multiple words require fewer bus cycles
  - Example: SPARCstation 20’s memory bus is 128 bit wide
  - Cost: more bus lines

- Block transfers:
  - Allow the bus to transfer multiple words in back-to-back bus cycles
  - Only one address needs to be sent at the beginning
  - The bus is not released until the last word is transferred
  - Cost: (a) increased complexity
    (b) decreased response time for request

Arbitration: Obtaining access to the bus

- One of the most important issues in bus design:
  - How is the bus reserved by a device that wishes to use it?

- Chaos is avoided by a master-slave arrangement:
  - Only the bus master can control access to the bus:
    - It initiates and controls all bus requests
  - A slave responds to read and write requests

- The simplest system – single bus master system:
  - Processor is the only bus master
  - All bus requests must be controlled by the processor
  - Major drawback: the processor is involved in every transaction
Multiple potential bus masters: The need for arbitration

- **Bus arbitration scheme:**
  - A bus master wanting to use the bus asserts the bus request
  - A bus master cannot use the bus until its request is granted
  - A bus master must signal to the arbiter after finish using the bus

- **Bus arbitration schemes usually try to balance two factors:**
  - **Bus priority:** the highest priority device should be serviced first
  - **Fairness:** Even the lowest priority device should never be completely locked out from the bus

Bus arbitration schemes can be divided into four broad classes:

- **Daisy chain arbitration**: single device with all request lines.
- **Centralized, parallel arbitration (e.g., PCI)**: see the next-next slide
- **Distributed arbitration by self-selection**: each device wanting the bus places a code indicating its identity on the bus, and determining the highest priority request by a pre-specified rule.
  - Example. NuBus in Apple Macintosh IIs.
- **Distributed arbitration by collision detection (e.g., Ethernet)**
### Daisy chain arbitration scheme

- **Advantage:** Simple
- **Disadvantages:**
  - Cannot assure fairness:
    - A low-priority device may be locked out indefinitely
  - The use of the daisy chain grant signal also limits the bus speed

---

### Centralized, parallel arbitration scheme

- **Disadvantage:** The bus arbiter may become the bottleneck.
- **Example.** Used in essentially all processor-memory busses and in high-speed I/O busses, such as PCI.
Giving commands to I/O devices

- Two methods are used to address the device:
  - Special I/O instructions
  - Memory-mapped I/O
- Special I/O instructions specify:
  - Both the device number and the command word
    - Device number: the processor communicates this via a set of wires normally included as part of the I/O bus
    - Command word: this is usually send on the bus’s data lines
- Memory-mapped I/O:
  - Portions of the address space are assigned to I/O device
  - Read and writes to those addresses are interpreted as commands to the I/O devices
  - User programs are prevented from issuing I/O operations directly:
    - The I/O address space is protected by the address translation

Example of I/O from INTEL viewpoint

- Direct or isolated I/O versus Memory-mapped I/O
**I/O map on PC**

- **(Isolated) I/O map in PC**
  - **00H~FFH**: Reserved for motherboard
  - **100H~3FFH**: Reserved for computer (peripheral) system
  - **400H~FFFFH**: Non-hardware-decoded area or I/O expansion area

- **Separate Assembly language for I/O access and memory access**
  - **IN 03F0H**;
  - **MOV AX, [03F0H]**;

---

**How I/O device notify the OS?**

- **The OS needs to know when:**
  - the I/O device has completed an operation
  - the I/O operation has encountered an error

- **This can be accomplished in two different ways:**
  - **Polling:**
    - The I/O device put information in a status register
    - The OS periodically check the status register
  - **I/O Interrupt**: (usually, hardware interrupt)
    - Whenever an I/O device needs attention from the processor, it interrupts the processor from what it is currently doing.
How I/O device notify the OS?

- **Polling** will waste a certain ratio of processor time.

  ![Process Flow Diagram](image)

  - polling, then processing peripheral’s request
  - pure polling

- **Interrupts**

  ![Interrupt Flow Diagram](image)

Polling: Programmed I/O

- **Advantage:**
  - Simple: the processor is totally in control and does all the work

- **Disadvantage:**
  - Polling overhead can consume a lot of CPU time
Interrupt driven data transfer

- **Advantage:**
  - User program progress is only halted during actual transfer
- **Disadvantage,** special hardware is needed to:
  - Cause an interrupt (I/O device)
  - Detect an interrupt (processor)
  - Save the proper states to resume after the interrupt (processor)

---

I/O interrupt

- An I/O interrupt is just like the exceptions except:
  - An I/O interrupt is asynchronous
  - Further information needs to be conveyed
- An I/O interrupt is asynchronous with respect to instruction execution:
  - I/O interrupt is not associated with any instruction
  - I/O interrupt does not prevent any instruction from completion
    - You can pick your own convenient point to take an interrupt
- I/O interrupt is more complicated than exception:
  - Needs to convey the identity of the device generating the interrupt
  - Interrupt requests can have different urgencies:
    - Interrupt request needs to be prioritized
Delegating I/O responsibility from the CPU: DMA

- **Direct Memory Access (DMA):**
  - External to the CPU
  - Act as a maser on the bus
  - Transfer blocks of data to or from memory without CPU intervention

CPU sends a starting address, direction, and length count to DMAC. Then issues "start".

DMAC provides handshake signals for Peripheral Controller, and Memory Addresses and handshake signals for Memory.

---

Basic DMA operations

- **without DMA**
  - CPU to disk
  - CPU to memory

- **with DMA**
  - CPU to disk
  - CPU to memory
Basic DMA operations

- DMA Controller

CPU

Cache

Generate control and address signals

DMA Controller

data

disk

memory

BUS

Problems introduced due to DMA

- Address translation in virtual memory
  - Virtual address versus physical address
- Stale data problem or coherency problem
  - If some of the locations into which the DMA writes are in the cache, the processor will receive the old value when it does a read.
  - If the cache is write-back, the DMA may read a value directly from memory when a newer value is in the cache.
- Solution
  - Cache flushing (either by OS or by hardware): Invalidate the cache for an I/O read, and force write-back for an I/O write.
I/O Bus Standards

- Today we have two dominant bus standards:

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Firewire (1.394)</th>
<th>USB 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus type</td>
<td>I/O</td>
<td>I/O</td>
</tr>
<tr>
<td>Basic data bus width (signals)</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Coding</td>
<td>adjustable</td>
<td>adjustable</td>
</tr>
<tr>
<td>Theoretical peak bandwidth</td>
<td>50 MB/sec (Firewire 400) or 100 MB/sec (Firewire 800)</td>
<td>480 MB/sec (high speed), 48 MB/sec (full speed), 12 MB/sec (low speed)</td>
</tr>
<tr>
<td>Hot-pluggable</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Maximum number of devices</td>
<td>63</td>
<td>127</td>
</tr>
<tr>
<td>Maximum bus length (coaxial wire)</td>
<td>4.5 meters</td>
<td>5 meters</td>
</tr>
<tr>
<td>Standard name</td>
<td>IEEE 1394, 1394</td>
<td>USB Implementers Forum</td>
</tr>
</tbody>
</table>

FIGURE 8.9 Key characteristics of two dominant I/O bus standards.

Other important issues

- Performance Analysis techniques:
  - queuing theory
  - simulation
  - analysis, i.e., find the weakest link (see "I/O System Design")

- Many new developments
Pentium 4

I/O Options

- Serial ATA (150 MB/sec)
- Parallel ATA (100 MB/sec)
- Parallel ATA (20 MB/sec)
- PCI bus (132 MB/sec)
- AGP 8X (2.1 GB/sec)
- Serial ATA (150 MB/sec)
- USB 2.0 (480 Mbps)

Fallacies and Pitfalls

- Fallacy: the rated mean time to failure of disks is 1,200,000 hours, so disks practically never fail.
  - Statistics does not apply to single event.
  - Failed disks = (1000 drives * 5 years/drive) / 1200000 hours/failure = 36, i.e., 3.6% of the disk drives would fail within 5 years.

- Fallacy: A 100 MB/sec bus can transfer 100 MB/sec.

- Pitfall: Moving functions from the CPU to the I/O processor, expecting to improve performance without analysis.
  - Can so-called intelligent I/O improve system performance?
Suggestive exercises

- 8.1, 8.3, 8.4, 8.5, 8.13, 8.14, 8.15, 8.16, 8.17, 8.18, 8.23, 8.29, 8.33, 8.39, 8.46