CS152
Computer Architecture and Engineering
Lecture 23

I/O Systems

April 28, 1999
John Kubiatowicz (http.cs.berkeley.edu/~kubitron)

lecture slides: http://www-inst.eecs.berkeley.edu/~cs152/

The Big Picture: Where are We Now?

Today’s Topic: I/O Systems

Outline of Today’s Lecture

- Recap/Finish up with Buses
- I/O Performance Measures
- Some Queueing Theory
- Types and Characteristics of I/O Devices
- Magnetic Disks
- DMA, Multimedia, OS
- Summary

Recap: Busses

- Fundamental tool for designing and building computer systems
  - divide the problem into independent components operating against a well defined interface
  - compose the components efficiently
- Shared collection of wires
  - command, address, data
- Communication path between multiple subsystems
  - Inexpensive
  - Limited bandwidth
- Layers of a bus specification
  - mechanical, electrical, signaling, timing, transactions
Recap: A Three-Bus System

° A small number of backplane buses tap into the processor-memory bus
  • Processor-memory bus is only used for processor-memory traffic
  • I/O buses are connected to the backplane bus

° Advantage: loading on processor bus is greatly reduced
° In fact: Modern systems have multiple buses leading directly from processor

Recap: Bus Transaction

° Arbitration: Who gets the bus
° Request: What do we want to do
° Action: What happens in response

Recap: Arbitration between Multiple Masters

° Bus arbitration: choose master for next transaction
  • 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
  • Centralized, parallel arbitration
  • Distributed arbitration by self-selection: each device wanting the bus places a code indicating its identity on the bus.
  • Distributed arbitration by collision detection: Ethernet uses this.

Recap: The Daisy Chain Bus Arbitrations 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
**Recap: Centralized Parallel Arbitration**

- Used in essentially all processor-memory busses and in high-speed I/O busses

**Recap: Distributed arbitration by collision detection**

- Everyone is a master
- Any agent can begin speaking at any time:
  - Should two agents start talking at the same time, this is called a “collision”.
  - Collisions are dealt with by “backing off” for random amount of time; good behavior for low load
- All agents listen all the time:
  - They all can source / sink data at same rate
  - They discard packets that are not meant for them
- Example: Ethernet

**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, they cannot be long if they are fast
  - Variant on Synchronous: *isochronous*
    - Clock passed along with data or recovered from data
    - Used on very high-speed, point-to-point communication links
    - People are talking about 4 gigabits/sec from PC board traces

- **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

**Simple Synchronous Protocol**

- Even memory busses are more complex than this
  - Memory (slave) may take time to respond
  - It may need to control data rate
Typical Synchronous Protocol

- Slave indicates when it is prepared for data xfer
- Actual transfer goes at bus rate

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) possibly increased response time for average request

Increasing Transaction Rate on Multimaster Bus

- Overlapped arbitration
  - perform arbitration for next transaction during current transaction
- Bus parking
  - master can holds onto bus and performs multiple transactions as long as no other master makes request
- Overlapped address / data phases (prev. slide)
  - requires one of the above techniques
- Split-phase (or packet switched) bus
  - completely separate address and data phases
  - arbitrate separately for each
  - address phase yield a tag which is matched with data phase
- "All of the above" in most modern memory busses

1993 MP Server Memory Bus Survey: GTL revolution

<table>
<thead>
<tr>
<th>Bus</th>
<th>MBus</th>
<th>Summit</th>
<th>Challenge</th>
<th>XDBus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Originator</td>
<td>Sun</td>
<td>HP</td>
<td>SGI</td>
<td>Sun</td>
</tr>
<tr>
<td>Clock Rate (MHz)</td>
<td>40</td>
<td>60</td>
<td>48</td>
<td>66</td>
</tr>
<tr>
<td>Address lines</td>
<td>36</td>
<td>48</td>
<td>40</td>
<td>muxed</td>
</tr>
<tr>
<td>Data lines</td>
<td>64</td>
<td>128</td>
<td>256</td>
<td>144 (parity)</td>
</tr>
<tr>
<td>Data Sizes (bits)</td>
<td>256</td>
<td>512</td>
<td>1024</td>
<td>512</td>
</tr>
<tr>
<td>Clocks/transfer</td>
<td>??</td>
<td>4</td>
<td>5</td>
<td>4?</td>
</tr>
<tr>
<td>Peak (MB/s)</td>
<td>320(80)</td>
<td>960</td>
<td>1200</td>
<td>1056</td>
</tr>
<tr>
<td>Master</td>
<td>Multi</td>
<td>Multi</td>
<td>Multi</td>
<td>Multi</td>
</tr>
<tr>
<td>Arbitration</td>
<td>Central</td>
<td>Central</td>
<td>Central</td>
<td>Central</td>
</tr>
<tr>
<td>Slots</td>
<td>16</td>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Busses/system</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Length</td>
<td>13 inches</td>
<td>12? inches</td>
<td>17 inches</td>
<td></td>
</tr>
</tbody>
</table>
### Asynchronous Write Transaction Handshake

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
<th>Read</th>
<th>Req</th>
<th>Ack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Asserts Address</td>
<td>Master Asserts Data</td>
<td></td>
<td></td>
<td>Next Address</td>
</tr>
</tbody>
</table>

- **t0**: Master has obtained control and has asserted address, direction, data
- **t1**: Waits a specified amount of time for slaves to decode target
- **t2**: Master asserts request line
- **t3**: Slave asserts ack, indicating data received
- **t4**: Master releases req
- **t5**: Slave releases ack

### Asynchronous Read Transaction

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
<th>Read</th>
<th>Req</th>
<th>Ack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Asserts Address</td>
<td>Slave Data</td>
<td></td>
<td></td>
<td>Next Address</td>
</tr>
</tbody>
</table>

- **t0**: Master has obtained control and asserts address, direction, data
- **t1**: Waits a specified amount of time for slaves to decode target
- **t2**: Master asserts request line
- **t3**: Slave asserts ack, indicating ready to transmit data
- **t4**: Master releases req, data received
- **t5**: Slave releases ack

### 1993 Backplane/IO Bus Survey

<table>
<thead>
<tr>
<th>Bus</th>
<th>SBus</th>
<th>TurboChannel</th>
<th>MicroChannel</th>
<th>PCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Originator</td>
<td>Sun</td>
<td>DEC</td>
<td>IBM</td>
<td>Intel</td>
</tr>
<tr>
<td>Clock Rate (MHz)</td>
<td>16-25</td>
<td>12.5-25</td>
<td>async</td>
<td>33</td>
</tr>
<tr>
<td>Addressing</td>
<td>Virtual</td>
<td>Physical</td>
<td>Physical</td>
<td>Physical</td>
</tr>
<tr>
<td>Data Sizes (bits)</td>
<td>8,16,32</td>
<td>8,16,24,32</td>
<td>8,16,24,32,64</td>
<td>8,16,24,32,64</td>
</tr>
<tr>
<td>Master</td>
<td>Multi</td>
<td>Single</td>
<td>Multi</td>
<td>Multi</td>
</tr>
<tr>
<td>Arbitration</td>
<td>Central</td>
<td>Central</td>
<td>Central</td>
<td>Central</td>
</tr>
<tr>
<td>32 bit read (MB/s)</td>
<td>33</td>
<td>25</td>
<td>20</td>
<td>33</td>
</tr>
<tr>
<td>Peak (MB/s)</td>
<td>89</td>
<td>84</td>
<td>75</td>
<td>111 (222)</td>
</tr>
<tr>
<td>Max Power (W)</td>
<td>16</td>
<td>26</td>
<td>13</td>
<td>25</td>
</tr>
</tbody>
</table>

### High Speed I/O Bus

- **Examples**
  - graphics
  - fast networks
- **Limited number of devices**
- **Data transfer bursts at full rate**
- **DMA transfers important**
  - small controller spools stream of bytes to or from memory
- **Either side may need to squelch transfer**
  - buffers fill up
Example: PCI Read/Write Transactions

- All signals sampled on rising edge
- Centralized Parallel Arbitration
  - overlapped with previous transaction
- All transfers are (unlimited) bursts
- Address phase starts by asserting FRAME#
- Next cycle “initiator” asserts cmd and address
- Data transfers happen on when
  - IRDY# asserted by master when ready to transfer data
  - TRDY# asserted by target when ready to transfer data
  - transfer when both asserted on rising edge
- FRAME# deasserted when master intends to complete only one more data transfer

Example: PCI Read Transaction

- Turn-around cycle on any signal driven by more than one agent

Example: PCI Write Transaction

- Push bus efficiency toward 100% under common simple usage
- like RISC
- Bus Parking
  - retain bus grant for previous master until another makes request
  - granted master can start next transfer without arbitration
- Arbitrary Burst length
  - initiator and target can exert flow control with xRDY
  - target can disconnect request with STOP (abort or retry)
  - master can disconnect by deasserting FRAME
  - arbiter can disconnect by deasserting GNT
- Delayed (pended, split-phase) transactions
  - free the bus after request to slow device

PCI Optimizations
Administrivia I

- Midterm II results (the test from Hades):
  - Average: 49.3
  - Standard Dev: 17
- Problem I gave people much more trouble than expected
  - Trick on cache simulation problem was to write address in binary
    and cross out bits below cache-line boundary:
    ⇒ word size was 8 bytes, so you wanted to cross out 3 bits.
- On Average memory access time problem, solution is recursive:
  - \( \text{AMAT}_{L_1\text{-data}} = T_{L_1\text{-hit}} + \text{MissRate}_{L_1\text{-data}} \times \text{AMAT}_{L_2} \)
  - \( \text{AMAT}_{L_2} = T_{L_2\text{-hit}} + \text{MissRate}_{L_2} \times \text{AMAT}_{\text{DRAM}} \)
  - Where \( T_{L_2\text{-hit}} \) is time to fill L1 cache from L2
    we also called this \( T_{L_1\text{-penalty}} \) in class

<table>
<thead>
<tr>
<th>Component</th>
<th>Hit Time</th>
<th>Miss Rate</th>
<th>Block Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>First-Level Cache</td>
<td>1 cycle</td>
<td>5% Data</td>
<td>32 bytes</td>
</tr>
<tr>
<td>Second-Level Cache</td>
<td>10 cycles + 1 cycle/64 bits</td>
<td>3%</td>
<td>128 bytes</td>
</tr>
<tr>
<td>DRAM</td>
<td>100ns+</td>
<td>25ns/8 bytes</td>
<td>1% 16K bytes</td>
</tr>
<tr>
<td>DISK</td>
<td>50ms+</td>
<td>20ns/byte</td>
<td>0% 16K bytes</td>
</tr>
</tbody>
</table>

I/O System Design Issues

- Performance
- Expandability
- Resilience in the face of failure

![Diagram of I/O System Design Issues]

Administrivia II

- Problem II was way too hard. SORRY!!!
  - Wide variation on solutions.
    I tried to be as lenient grading this as possible.
- Problem III: not everyone understood “complete pipelining”

<table>
<thead>
<tr>
<th>IF</th>
<th>ID</th>
<th>EX1</th>
<th>MEM1</th>
<th>EX2</th>
<th>MEM2</th>
<th>EX3</th>
<th>EX4</th>
<th>WR</th>
</tr>
</thead>
</table>

Administrivia III

- Important: Design for Test
  - You should be testing from the very start of your design
  - Consider adding special monitor modules at various points in design ⇒ I have asked you to label trace output from these modules with the current clock cycle #
  - The time to understand how components of your design should work is while you are designing!
- Pending schedule:
  - Friday 4/30: Update on design
  - Today 4/26 and Wednesday 4/28
  - Monday 5/3: no class
  - Wednesday 5/5: Multiprocesing
  - Monday 5/10 Last class (wrap up, evaluations, etc)
  - Tuesday 5/11 Oral reports: 10-12am and 2-4pm
    - Signup sheet will be on my office door today or tomorrow
  - Tuesday 5/11 by 5pm: final project reports due.
  - Friday 5/14 grades should be posted.
### I/O Device Examples

<table>
<thead>
<tr>
<th>Device</th>
<th>Behavior</th>
<th>Partner</th>
<th>Data Rate (KB/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard</td>
<td>Input</td>
<td>Human</td>
<td>0.01</td>
</tr>
<tr>
<td>Mouse</td>
<td>Input</td>
<td>Human</td>
<td>0.02</td>
</tr>
<tr>
<td>Line Printer</td>
<td>Output</td>
<td>Human</td>
<td>1.00</td>
</tr>
<tr>
<td>Floppy disk</td>
<td>Storage</td>
<td>Machine</td>
<td>50.00</td>
</tr>
<tr>
<td>Laser Printer</td>
<td>Output</td>
<td>Human</td>
<td>100.00</td>
</tr>
<tr>
<td>Optical Disk</td>
<td>Storage</td>
<td>Machine</td>
<td>500.00</td>
</tr>
<tr>
<td>Magnetic Disk</td>
<td>Storage</td>
<td>Machine</td>
<td>5,000.00</td>
</tr>
<tr>
<td>Network-LAN</td>
<td>Input or Output</td>
<td>Machine</td>
<td>20 – 1,000.00</td>
</tr>
<tr>
<td>Graphics Display</td>
<td>Output</td>
<td>Human</td>
<td>30,000.00</td>
</tr>
</tbody>
</table>

### I/O System Performance

* I/O System performance depends on many aspects of the system ("limited by weakest link in the chain"):
  - The CPU
  - The memory system:
    - Internal and external caches
    - Main Memory
  - The underlying interconnection (buses)
  - The I/O controller
  - The I/O device
  - The speed of the I/O software (Operating System)
  - The efficiency of the software’s use of the I/O devices

* Two common performance metrics:
  - Throughput: I/O bandwidth
  - Response time: Latency

### Simple Producer-Server Model

- **Throughput:**
  - The number of tasks completed by the server in unit time
  - In order to get the highest possible throughput:
    - The server should never be idle
    - The queue should never be empty

- **Response time:**
  - Begins when a task is placed in the queue
  - Ends when it is completed by the server
  - In order to minimize the response time:
    - The queue should be empty
    - The server will be idle

### Throughput versus Respond Time

- **Throughput:**
  - The number of tasks completed by the server in unit time
  - In order to get the highest possible throughput:
    - The server should never be idle
    - The queue should never be empty

- **Response time:**
  - Begins when a task is placed in the queue
  - Ends when it is completed by the server
  - In order to minimize the response time:
    - The queue should be empty
    - The server will be idle
Throughput Enhancement

- In general throughput can be improved by:
  - Throwing more hardware at the problem
  - Reduces load-related latency
- Response time is much harder to reduce:
  - Ultimately it is limited by the speed of light (but we’re far from it)

I/O Benchmarks for Magnetic Disks

- Supercomputer application:
  - Large-scale scientific problems ⇒ large files
  - One large read and many small writes to snapshot computation
    - **Data Rate**: MB/second between memory and disk
- Transaction processing:
  - Examples: Airline reservations systems and bank ATMs
  - Small changes to large shared software
  - **I/O Rate**: No. disk accesses / second given upper limit for latency
- File system:
  - Measurements of UNIX file systems in an engineering environment:
    - 80% of accesses are to files less than 10 KB
    - 90% of all file accesses are to data with sequential addresses on the disk
    - 67% of the accesses are reads, 27% writes, 6% read-write
  - **I/O Rate & Latency**: No. disk accesses / second and response time

Magnetic Disk

- Purpose:
  - Long term, nonvolatile storage
  - Large, inexpensive, and slow
  - Lowest level in the memory hierarchy
- Two major types:
  - Floppy disk
  - Hard disk
- Both types of disks:
  - Rely on a rotating platter coated with a magnetic surface
  - Use a moveable read/write head to access the disk
- Advantages of hard disks over floppy disks:
  - Platters are more rigid (metal or glass) so they can be larger
  - Higher density because it can be controlled more precisely
  - Higher data rate because it spins faster
  - Can incorporate more than one platter

Organization of a Hard Magnetic Disk

- Typical numbers (depending on the disk size):
  - 500 to 2,000 tracks per surface
  - 32 to 128 sectors per track
    - A sector is the smallest unit that can be read or written
- Traditionally all tracks have the same number of sectors:
  - Constant bit density: record more sectors on the outer tracks
- Recently relaxed: constant bit size, speed varies with track location
**Magnetic Disk Characteristic**

- **Cylinder**: all the tracks under the head at a given point on all surfaces
- **Read/write data is a three-stage process:**
  - **Seek time**: position the arm over the proper track
  - **Rotational latency**: wait for the desired sector to rotate under the read/write head
  - **Transfer time**: transfer a block of bits (sector) under the read-write head
- **Average seek time as reported by the industry:**
  - Typically in the range of 8 ms to 12 ms
  - (Sum of the time for all possible seek) / (total # of possible seeks)
- **Due to locality of disk reference, actual average seek time may:**
  - Only be 25% to 33% of the advertised number

**Typical Numbers of a Magnetic Disk**

- **Rotational Latency**:
  - Most disks rotate at 3,600 to 7,200 RPM
  - Approximately 16 ms to 8 ms per revolution, respectively
  - An average latency to the desired information is halfway around the disk:
    - 8 ms at 3600 RPM, 4 ms at 7200 RPM
- **Transfer Time is a function of**:
  - Transfer size (usually a sector): 1 KB / sector
  - Rotation speed: 3600 RPM to 7200 RPM
  - Recording density: bits per inch on a track
  - Diameter typical diameter ranges from 2.5 to 5.25 in
  - Typical values: 2 to 12 MB per second

**Disk I/O Performance**

- **Disk Access Time** = Seek time + Rotational Latency + Transfer time + Controller Time + Queueing Delay

**Example**

- 512 byte sector, rotate at 5400 RPM, advertised seeks is 12 ms, transfer rate is 4 BM/sec, controller overhead is 1 ms, queue idle so no service time

**Disk Access Time** = Seek time + Rotational Latency + Transfer time + Controller Time + Queueing Delay

- Disk Access Time = 12 ms + 0.5 / 5400 RPM + 0.5 KB / 4 MB/s + 1 ms + 0 ms
- Disk Access Time = 12 ms + 0.5 / 90 RPS + 0.125 / 1024 s + 1 ms + 0 ms
- Disk Access Time = 12 ms + 5.5 ms + 0.1 ms + 1 ms + 0 ms
- Disk Access Time = 18.6 ms

- If real seeks are 1/3 advertised seeks, then its 10.6 ms, with rotation delay at 50% of the time!
### Magnetic Disk Examples

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>IBM 3090</th>
<th>IBM UltraStar</th>
<th>Integral 1820</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk diameter (inches)</td>
<td>10.88</td>
<td>3.50</td>
<td>1.80</td>
</tr>
<tr>
<td>Formatted data capacity (MB)</td>
<td>22,700</td>
<td>4,300</td>
<td>21</td>
</tr>
<tr>
<td>MTTF (hours)</td>
<td>50,000</td>
<td>1,000,000</td>
<td>100,000</td>
</tr>
<tr>
<td>Number of arms/box</td>
<td>12</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Rotation speed (RPM)</td>
<td>3,600</td>
<td>7,200</td>
<td>3,800</td>
</tr>
<tr>
<td>Transfer rate (MB/sec)</td>
<td>4.2</td>
<td>9-12</td>
<td>1.9</td>
</tr>
<tr>
<td>Power/box (watts)</td>
<td>2,900</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>MB/watt</td>
<td>8</td>
<td>102</td>
<td>10.5</td>
</tr>
<tr>
<td>Volume (cubic feet)</td>
<td>97</td>
<td>0.13</td>
<td>0.02</td>
</tr>
<tr>
<td>MB/cubic feet</td>
<td>234</td>
<td>33000</td>
<td>1050</td>
</tr>
</tbody>
</table>

### Reliability and Availability

- Two terms that are often confused:
  - Reliability: Is anything broken?
  - Availability: Is the system still available to the user?

- Availability can be improved by adding hardware:
  - Example: adding ECC on memory

- Reliability can only be improved by:
  - Bettering environmental conditions
  - Building more reliable components
  - Building with fewer components
    - Improve availability may come at the cost of lower reliability

### Disk Arrays

- A new organization of disk storage:
  - Arrays of small and inexpensive disks
  - Increase potential throughput by having many disk drives:
    - Data is spread over multiple disk
    - Multiple accesses are made to several disks

- Reliability is lower than a single disk:
  - But availability can be improved by adding redundant disks (RAID):
    - Lost information can be reconstructed from redundant information
  - MTTR: mean time to repair is in the order of hours
  - MTTF: mean time to failure of disks is tens of years

### Optical Compact Disks

- Disadvantage:
  - It is primarily read-only media

- Advantages of Optical Compact Disk:
  - It is removable
  - It is inexpensive to manufacture
  - Have the potential to compete with new tape technologies for archival storage
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

I/O Device Notifying 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
  • I/O Interrupt:
    - Whenever an I/O device needs attention from the processor, it interrupts the processor from what it is currently doing.
  • Polling:
    - The I/O device put information in a status register
    - The OS periodically check the status register

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

Example: Device Interrupt

° 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)
**Disable Network Intr**

```asm
subi $r4,$r1,#4
slli $r4,$r4,#2
lw  $r2,0($r4)
lw  $r3,4($r4)
add $r2,$r2,$r3
sw  8($r4),$r2
lw  $r1,12($zero)
beq $r1,no_mess
lw  $r1,20($r0)
lw  $r2,0($r1)
addi $r3,$r0,#5
sw  0($r1),$r3
```

**Polling Point (check device register)**

```
Polling Point
```

**Clear Network Intr**

```
```

---

**Polling is faster/slower than Interrupts**

- Polling is faster than interrupts because
  - Compiler knows which registers in use at polling point. Hence, do not need to save and restore registers (or not as many).
  - Other interrupt overhead avoided (pipeline flush, trap priorities, etc).

- Polling is slower than interrupts because
  - Overhead of polling instructions is incurred regardless of whether or not handler is run. This could add to inner-loop delay.
  - Device may have to wait for service for a long time.

- When to use one or the other?
  - Multi-axis tradeoff
    - Frequent/regular events good for polling, as long as device can be controlled at user level.
    - Interrupts good for infrequent/irregular events
    - Interrupts good for ensuring regular/predictable service of events.

---

**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.
Delegating I/O Responsibility from the CPU: IOP

- Issues instruction to IOP
- IOP looks in memory for commands
- Device to/from memory transfers are controlled by the IOP directly.
- IOP steals memory cycles.

Responsibilities of the Operating System

- The operating system acts as the interface between:
  - The I/O hardware and the program that requests I/O
- Three characteristics of the I/O systems:
  - The I/O system is shared by multiple programs using the processor
  - I/O systems often use interrupts (external generated exceptions) to communicate information about I/O operations.
    - Interrupts must be handled by the OS because they cause a transfer to supervisor mode
  - The low-level control of an I/O device is complex:
    - Managing a set of concurrent events
    - The requirements for correct device control are very detailed

Operating System Requirements

- Provide protection to shared I/O resources
  - Guarantees that a user’s program can only access the portions of an I/O device to which the user has rights
- Provides abstraction for accessing devices:
  - Supply routines that handle low-level device operation
- Handles the interrupts generated by I/O devices
- Provide equitable access to the shared I/O resources
  - All user programs must have equal access to the I/O resources
- Schedule accesses in order to enhance system throughput

OS and I/O Systems Communication Requirements

- The Operating System must be able to prevent:
  - The user program from communicating with the I/O device directly
- If user programs could perform I/O directly:
  - Protection to the shared I/O resources could not be provided
- Three types of communication are required:
  - The OS must be able to give commands to the I/O devices
  - The I/O device must be able to notify the OS when the I/O device has completed an operation or has encountered an error
  - Data must be transferred between memory and an I/O device
### Multimedia Bandwidth Requirements

° **High Quality Video**
  - Digital Data = (30 frames / second) (640 x 480 pels) (24-bit color / pel) = 221 Mbps  
  
° **Reduced Quality Video**
  - Digital Data = (15 frames / second) (320 x 240 pels) (16-bit color / pel) = 18 Mbps  

° **High Quality Audio**
  - Digital Data = (44,100 audio samples / sec) (16-bit audio samples)  
  - (2 audio channels for stereo) = 1.4 Mbps  

° **Reduced Quality Audio**
  - Digital Data = (11,050 audio samples / sec) (8-bit audio samples) (1 audio channel for monaural) = 0.1 Mbps  

° compression changes the whole story!

### Multimedia and Latency

° **How sensitive is your eye / ear to variations in audio / video rate?**

° **How can you ensure constant rate of delivery?**

° **Jitter (latency) bounds vs constant bit rate transfer**

° **Synchronizing audio and video streams**
  - you can tolerate 15-20 ms early to 30-40 ms late

### P1394 High-Speed Serial Bus (firewire)

° a digital interface – there is no need to convert digital data into analog and tolerate a loss of data integrity,

° physically small - the thin serial cable can replace larger and more expensive interfaces,

° easy to use - no need for terminators, device IDs, or elaborate setup,

° hot pluggable - users can add or remove 1394 devices with the bus active,

° inexpensive - priced for consumer products,

° scalable architecture - may mix 100, 200, and 400 Mbps devices on a bus,

° flexible topology - support of daisy chaining and branching for true peer-to-peer communication,

° fast - even multimedia data can be guaranteed its bandwidth for just-in-time delivery, and

° non-proprietary

° mixed asynchronous and isochronous traffic

### Firewire Operations

° Fixed frame is divided into preallocated CBR slots + best effort asynchronous slot

° Each slot has packet containing “ID” command and data

° Example: digital video camera can expect to send one 64 byte packet every 125 µs
  - 80 * 1024 * 64 = 5MB/s
Summary:

° I/O performance limited by weakest link in chain between OS and device
° Queueing theory is important
  • 100% utilization means very large latency
  • Remember, for M/G/1 queue:
    - queue size goes as $u/(1-u)$
    - latency goes as $T_{ser} = u/(1-u)$
° Disk I/O Benchmarks: I/O rate vs. Data rate vs. latency
° Three Components of Disk Access Time:
  • Seek Time: advertised to be 8 to 12 ms. May be lower in real life.
  • Rotational Latency: 4.1 ms at 7200 RPM and 8.3 ms at 3600 RPM
  • Transfer Time: 2 to 12 MB per second
° I/O device notifying the operating system:
  • Polling: it can waste a lot of processor time
  • I/O interrupt: similar to exception except it is asynchronous
° Delegating I/O responsibility from the CPU: DMA, or even IOP
° wide range of devices
  • multimedia and high speed networking poise important challenges