## Recap: Summary of Bus Options:

<table>
<thead>
<tr>
<th>Option</th>
<th>High performance</th>
<th>Low cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus width</td>
<td>Separate address &amp; data lines</td>
<td>Multiplex address &amp; data lines</td>
</tr>
<tr>
<td>Data width</td>
<td>Wider is faster (e.g., 32 bits)</td>
<td>Narrower is cheaper (e.g., 8 bits)</td>
</tr>
<tr>
<td>Transfer size</td>
<td>Multiple words has less bus overhead</td>
<td>Single-word transfer is simpler</td>
</tr>
<tr>
<td>Bus masters</td>
<td>Multiple (requires arbitration)</td>
<td>Single master (no arbitration)</td>
</tr>
<tr>
<td>Clocking</td>
<td>Synchronous</td>
<td>Asynchronous</td>
</tr>
<tr>
<td>Protocol</td>
<td>pipelined</td>
<td>Serial</td>
</tr>
</tbody>
</table>
### I/O System Design Issues

![Diagram of I/O System Design]

- Processor
- Cache
- Memory - I/O Bus
- Main Memory
- I/O Controller
- I/O Controller
- I/O Controller
- Disk
- Disk
- Graphics
- Network
- Intermittent

### 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
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

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

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

![Flowchart of Polling: Programmed I/O]

- Is the data ready?
  - Yes
    - Read data
  - No
    - Busy wait loop

- done?
  - Yes
  - Store data
  - No

busy wait loop not an efficient way to use the CPU unless the device is very fast!

but checks for I/O completion can be dispersed among computation intensive code
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
### Interrupt Logic

- Detect and synchronize interrupt requests
  - Ignore interrupts that are disabled (masked off)
  - Rank the pending interrupt requests
  - Create interrupt microsequence address
  - Provide select signals for interrupt microsequence

![Diagram of Interrupt Logic](image)

### Program Interrupt/Exception Hardware

- Hardware interrupt services:
  - Save the PC (or PCs in a pipelined machine)
  - Inhibit the interrupt that is being handled
  - Branch to interrupt service routine
  - Options:
    - Save status, save registers, save interrupt information
    - Change status, change operating modes, get interrupt info.

- A “good thing” about interrupt:
  - Asynchronous: not associated with a particular instruction
  - Pick the most convenient place in the pipeline to handle it
Interrupt target address options:
- General: Branch to a common address for all interrupts
  Software then decode the cause and figure out what to do
- Specific: Automatically branch to different addresses based on interrupt type and/or level--vectored interrupt

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

<table>
<thead>
<tr>
<th>CPU</th>
<th>IOP</th>
<th>Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Device to/from memory transfers are controlled by the IOP directly.
- IOP looks in memory for commands.
- I/O processor (IOP)

Summary:

- Three types of buses:
  - Processor-memory buses
  - I/O buses
  - Backplane buses
- Bus arbitration schemes:
  - Daisy chain arbitration: it cannot assure fairness
  - Centralized parallel arbitration: requires a central arbiter
- 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
  - Direct memory access (DMA)
  - I/O processor (IOP)
ECE468: Objectives and Assessment

- In-depth understanding of the inner-workings of modern computers, their evolution, and trade-offs present at the hardware/software boundary.
  - Insight into fast/slow operations that are easy/hard to implement in hardware
- Experience with the design process in the context of a large complex (hardware) design.
  - Functional Spec --> Control & Datapath --> Physical implementation

The Big Picture

![Diagram showing the components of a computer system: Processor, Control, Datapath, Memory, Input, Output]