I/O Device

Chapter 36

I/O Devices

- I/O == Input/Output

- Includes:
  - Network Interfaces
  - Graphics Cards
  - HDD/SSD (other internal storage devices)
  - USB Devices
  - Etc.

- How do these things communicate with the OS

They ride the bus!

The Hierarchical Bus Architecture

- We have a system of interconnected transfer channels
- Channels have a hierarchy supporting different transfer speeds
  - Some devices are slower than others
- Based on physics and cost
  - Physically == less latency
  - Higher speed materials == $$$

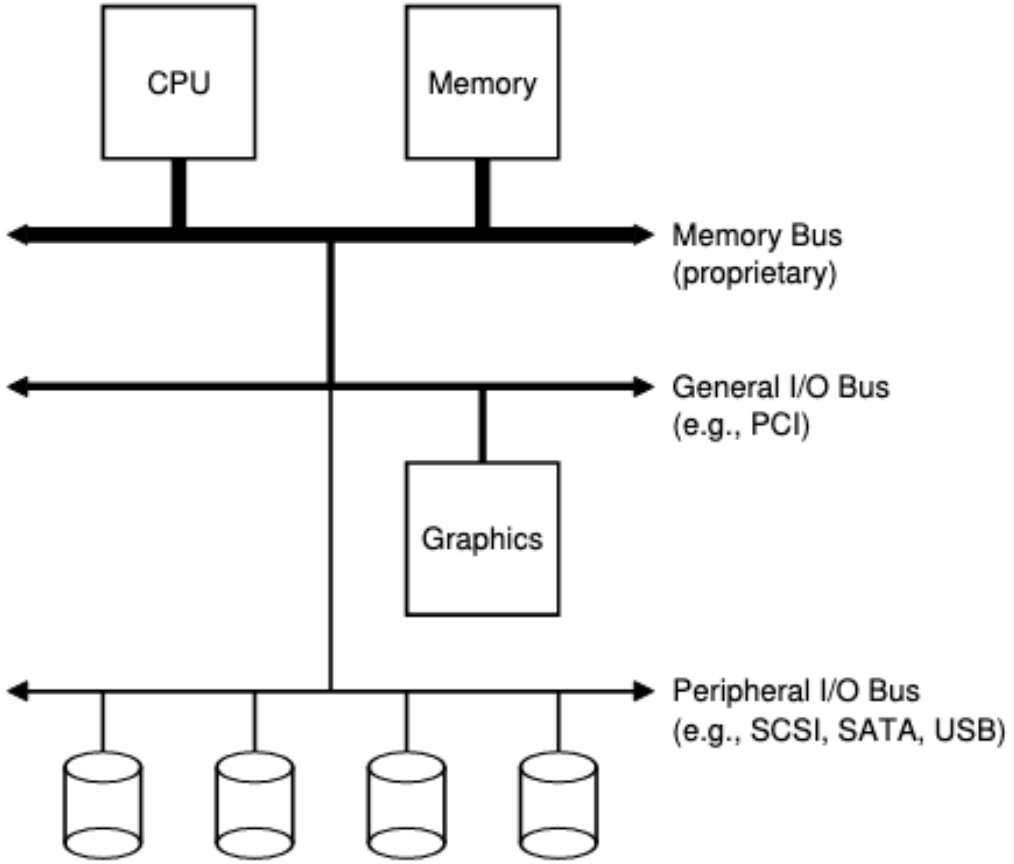


Figure 36.1: **Prototypical System Architecture**

The Hierarchical Bus Architecture

- We have a system of interconnected transfer channels
- Channels have a hierarchy supporting different transfer speeds
  - Some devices are slower than others
- Based on physics and cost
  - Physically == less latency
  - Higher speed materials == $$$

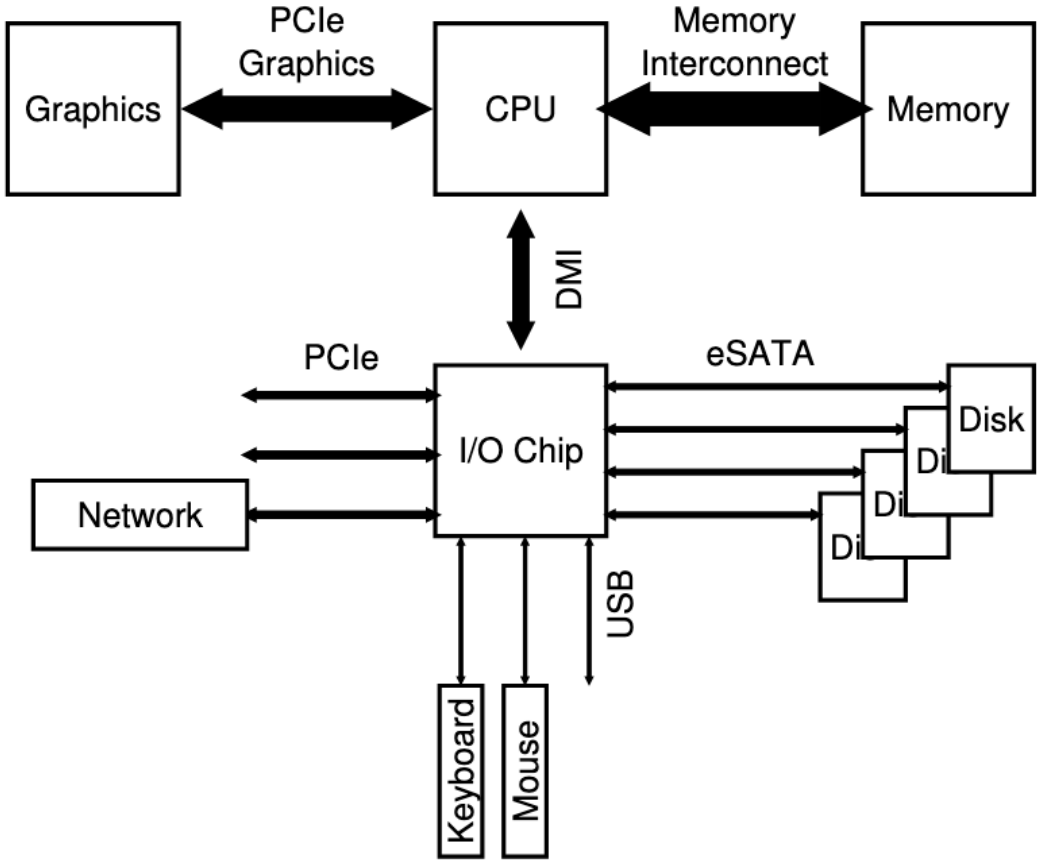


Figure 36.2: **Modern System Architecture**

Programmed I/O (PIO)

- The CPU directly tells a device to do something
  - The device uses **firmware** to perform its function

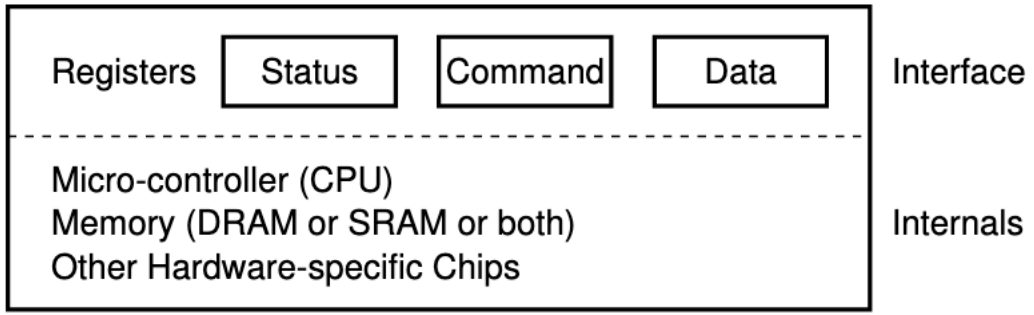


Figure 36.3: **A Canonical Device**

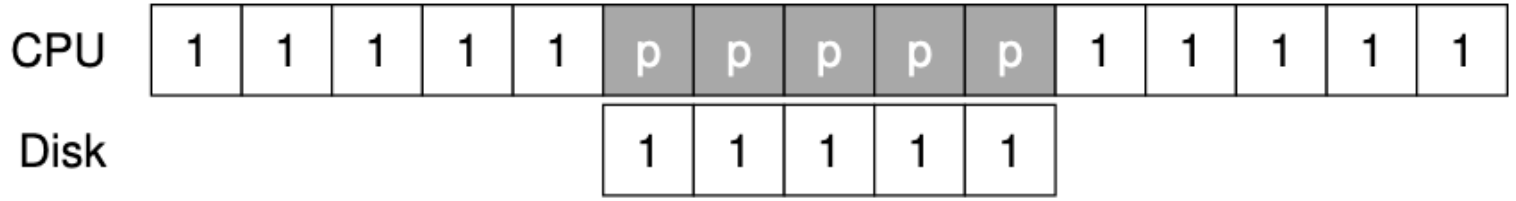
- The CPU:
  - gets the status of the device
  - puts data in the ”Data” register
  - puts the operations in the ”Command” register

```
While (STATUS == BUSY)
    ; // wait until device is not busy
Write data to DATA register
Write command to COMMAND register
    (starts the device and executes the command)
While (STATUS == BUSY)
    ; // wait until device is done with your request
```

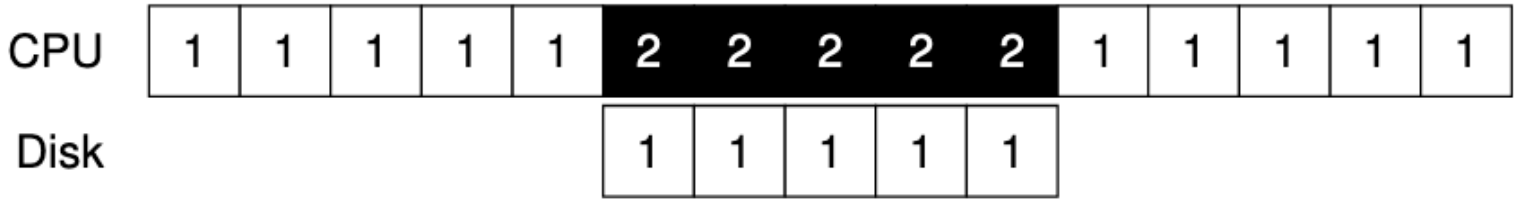
- Polling (like busy waiting) means the CPU can’t serve requests

Interrupts

- Polling isn't ideal for slower devices



- What if we let the CPU do other stuff, and then use interrupts to notify it when the job is done?



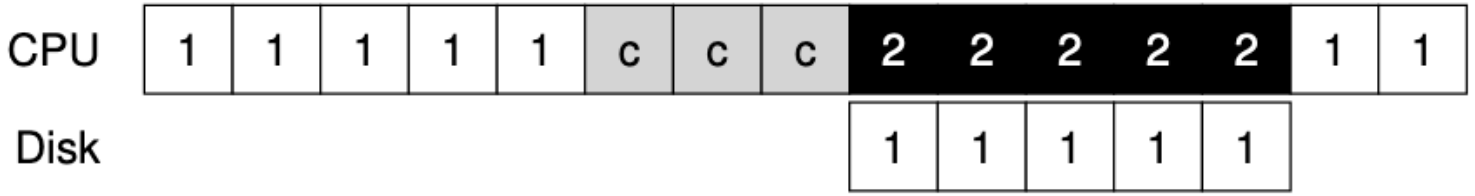
- Better overall use of resources!

So always use interrupts?

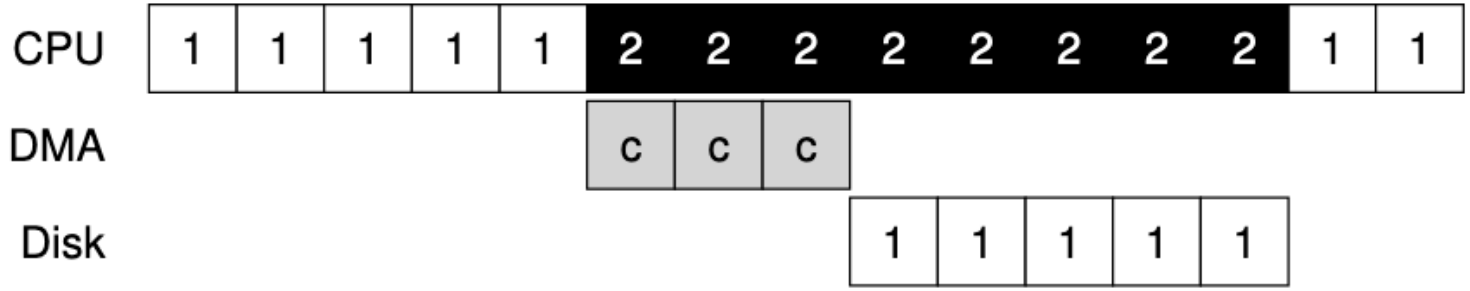
- Nope, interrupts generate overhead
- A fast device generating a bunch of interrupts can slow the system down as it needs to perform context switching to service the interrupt
- In some scenarios like networking, it's possible to flood a system with so many requests that it becomes livelocked
  - (D)DOS Attack
- We may be able to mitigate this a bit buy using coalescing to group of requests with a single interrupt
  - Lower interrupt overhead, but more latency added to the request

Data Transfer

- PIO requires the CPU be directly involved in the copying of data
  - Not exactly an efficient use of the CPU's time



- Add Direct Memory Access (DMA) hardware to help serve the request
  - OS tells the DMA where the data is, how much to copy, and where to put it
  - DMA issues an interrupt when the job is done



Interacting with I/O Devices

- Explicit I/O Instructions
  - **Privileged** instructions indicating the register holding the data and the port to identify the device
- Memory-mapped I/O
  - Device registers are exposed as memory addresses where data can be loaded/stored like any other memory address
  - The hardware routes the load/store to the device directly
  - Requires fewer specific instructions in instruction set
- Both are still used today and are relatively equivalent

Coordinating with the OS

- Devices have **firmware** software built-in to make them function

- The OS uses a **device driver** software to provide an abstraction for the OS to interact with the device
  - E.g. Nvidia drivers do not work with AMD video cards

- Hierarchical Abstraction
  - API interfaces at the top (exposed to users)
  - Device drivers to perform operations near the bottom

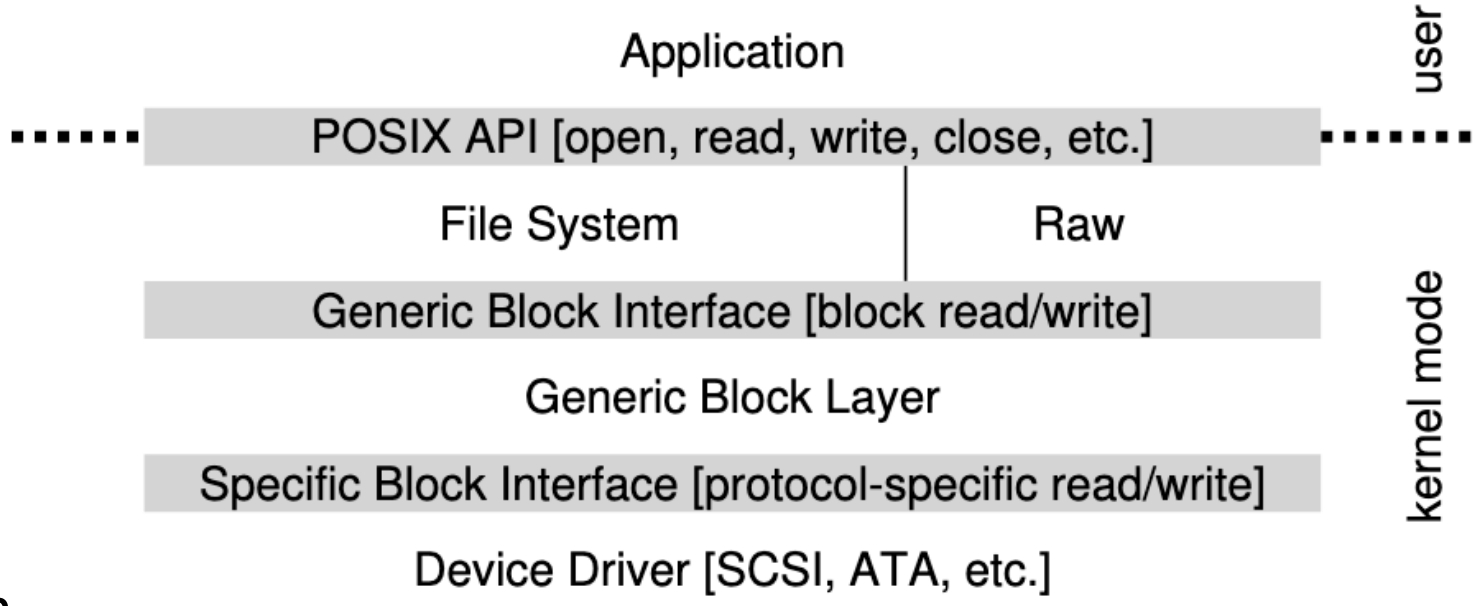


Figure 36.4: **The File System Stack**

Next Time

- We talk about Hard Disk Drives
  - Primarily the spinning metal platter variety
- Calculating disk performance
- Disk Scheduling