Operating Systems

Chapter 4 I/O and Driver

Tien Pham Van, Dr. rer. nat.

Compiled with reference to other presentations

Concept

- Communication between an information processing system (computer) and peripherals, or another processing system
- It may refer to input/output operations for data exchange

Overview

- Two fundamental operations performed by a computing system
  - Processing of data (implement an algorithm)
  - Perform I/O (move data into and out of the system)
- Large diversity in I/O devices, capabilities and performance
  - Common Categories: storage, transmission and user-interface devices.
  - Examples: display, keyboard, network, hard disks, tape drives, CDROM ...
- OS Goal is to
  1. provide simple and consistent interface to user
  2. optimize I/O use for maximum concurrency
- Mechanisms used by OS
  - device drivers provide standard interface to I/O devices

Categories of I/O Devices

- Human readable
  - Used to communicate with the user
  - Printers
  - Video display terminals
    - Display
    - Keyboard
    - Mouse

Categories of I/O Devices

- Machine readable
  - Used to communicate with electronic equipment
  - Disk and tape drives
  - Sensors
  - Controllers
  - Actuators

Categories of I/O Devices

- Communication
  - Used to communicate with remote devices
  - Digital line drivers
  - Modems

Differences in I/O Devices

- Data rate
  - May be differences of several orders of magnitude between the data transfer rates

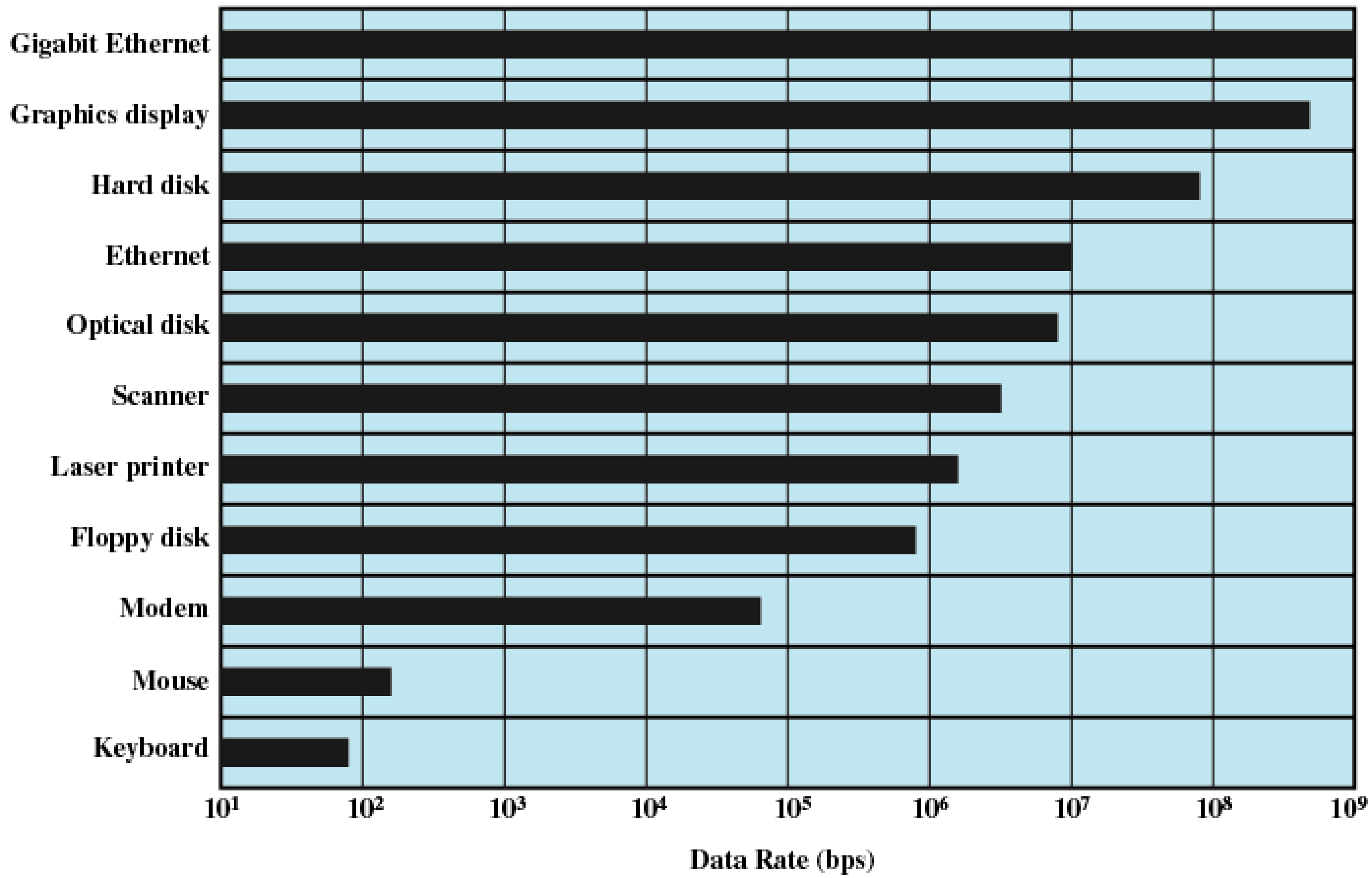


Figure 11.1 Typical I/O Device Data Rates

Embedded Networking Research Group
Email: tien.phamvan1@hust.edu.vn
School of Elec. and Telecom - Hanoi University of Science and Technology
C9-411, Dai Co Viet str. 1, HBT, Hanoi
Tel: +84-243-8693596

Differences in I/O Devices

- Application
  - Disk used to store files requires file management software
  - Disk used to store virtual memory pages needs special hardware and software to support it
  - Terminal used by system administrator may have a higher priority

Differences in I/O Devices

- Complexity of control
- Unit of transfer
  - Data may be transferred as a stream of bytes for a terminal or in larger blocks for a disk
- Data representation
  - Encoding schemes
- Error conditions
  - Devices respond to errors differently

Performing I/O

- Programmed I/O
  - Process is busy-waiting for the operation to complete
- Interrupt-driven I/O
  - I/O command is issued
  - Processor continues executing instructions
  - I/O module sends an interrupt when done

Performing I/O

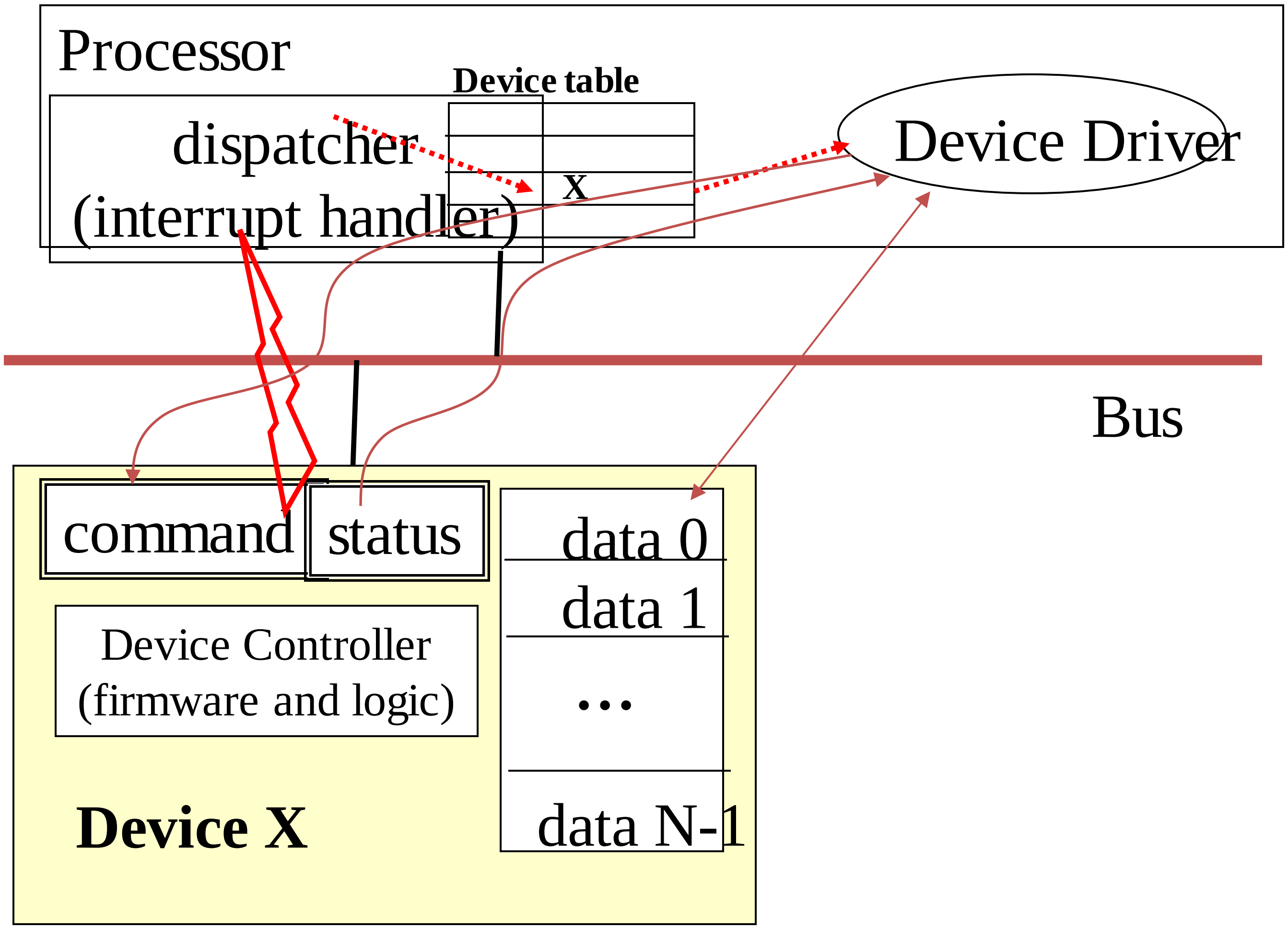
- Direct Memory Access (DMA)
  - DMA module controls exchange of data between main memory and the I/O device
  - Processor interrupted only after entire block has been transferred

Relationship Among Techniques

Table 11.1 I/O Techniques

	No Interrupts	**Use of Interrupts**
I/O-to-memory transfer through processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-memory transfer		Direct memory access (DMA)

I/O and Devices

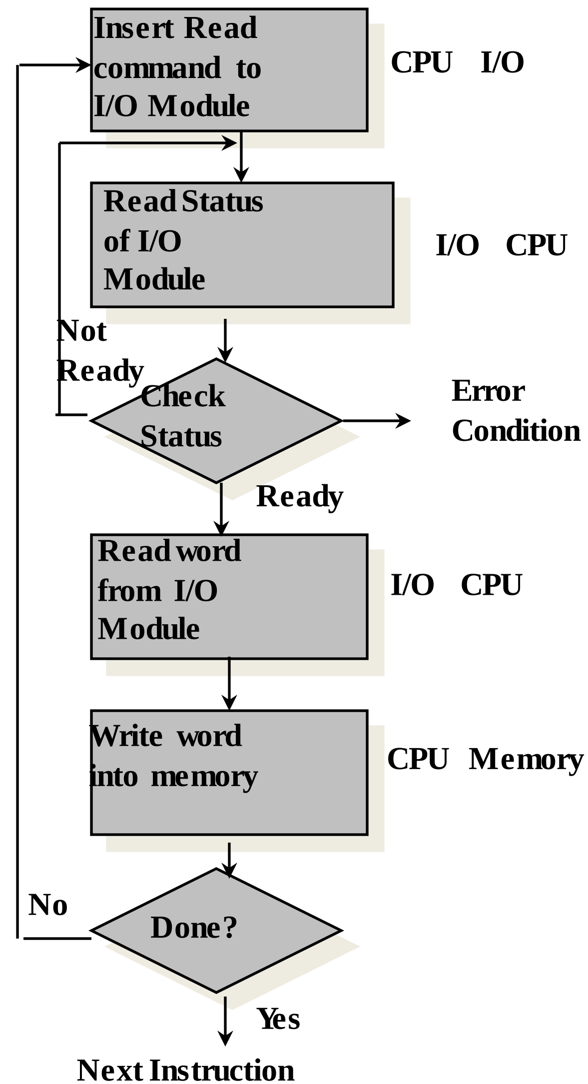


I/O Hardware

- Common concepts
  - *Port* (device-machine communication point)
  - *Bus* (daisy chain or shared communication channel)
  - *Controller* (host adapter)
- Common Techniques
  1. Direct I/O with polling, aka *Programmed I/O:* Processor does all the work. Poll for results.
  2. Interrupt Driven I/O: Device notifies CPU when I/O operation complete
  3. Memory Mapped I/O: rather than reading/writing to controller registers the device is mapped into the OS memory space. increased efficiency
  4. Direct memory access (DMA): DMA controller read and write directly to memory, freeing the CPU to do other things.CPU notified when DMA complete

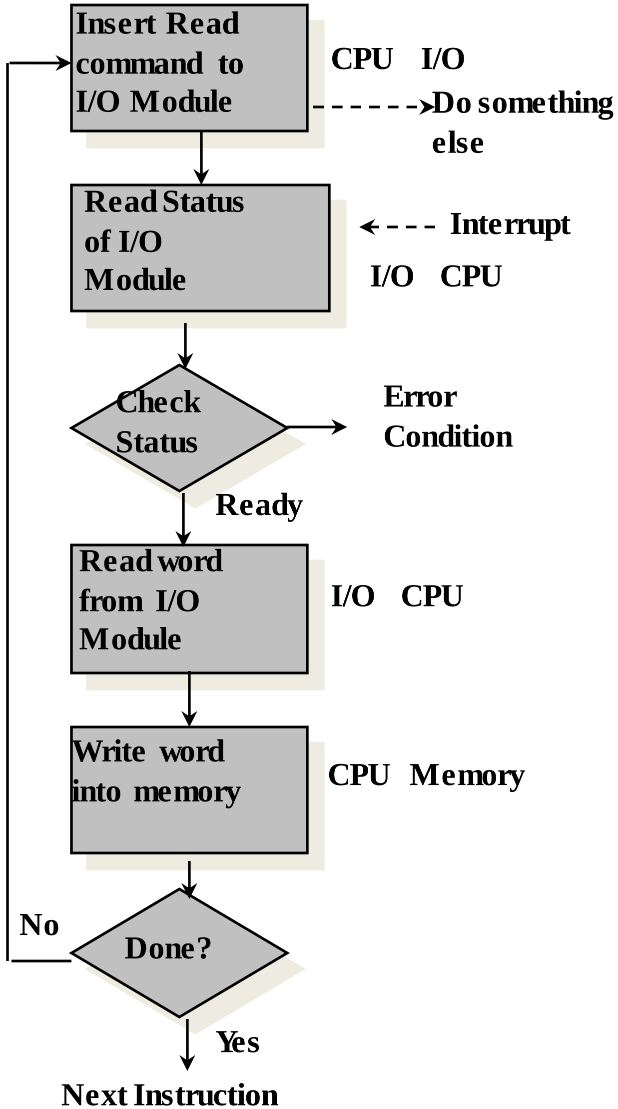
Programmed I/O

- *Busy-wait* cycle to wait for I/O from device
- Poll at select times: periodic, entering/leaving kernel etc.
  - Determines state of device: `command-ready`, `busy`, `error`
- Processor transfer data to/from device.
- Read/write directly to status and command registers
- Processor polls device for status
- Consumes a lot of processor time because every word read or written passes through the processor



Interrupt-Driven I/O

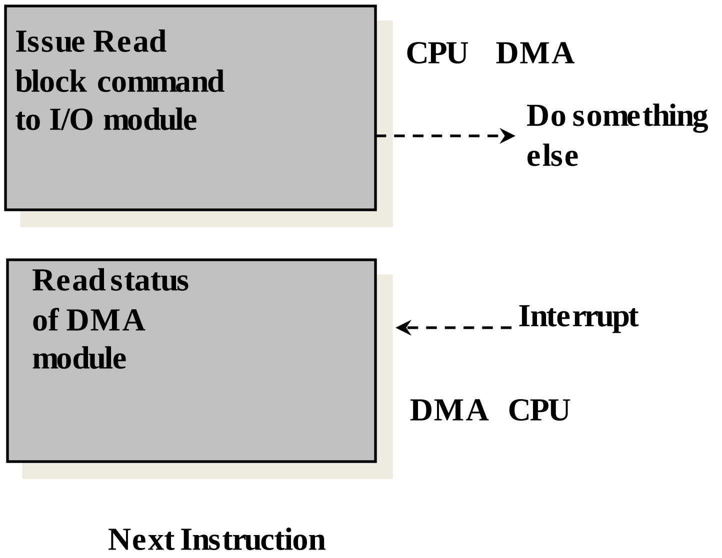
- Similar to direct I/O but processor not required to poll device.
- Interrupt asserted to notify processor of a change in status
- *Interrupt handler* receives interrupts
- *Maskable* to ignore or delay some interrupts
- *Interrupt vector* to dispatch interrupt to correct handler
  – Based on priority
  – Some unmaskable



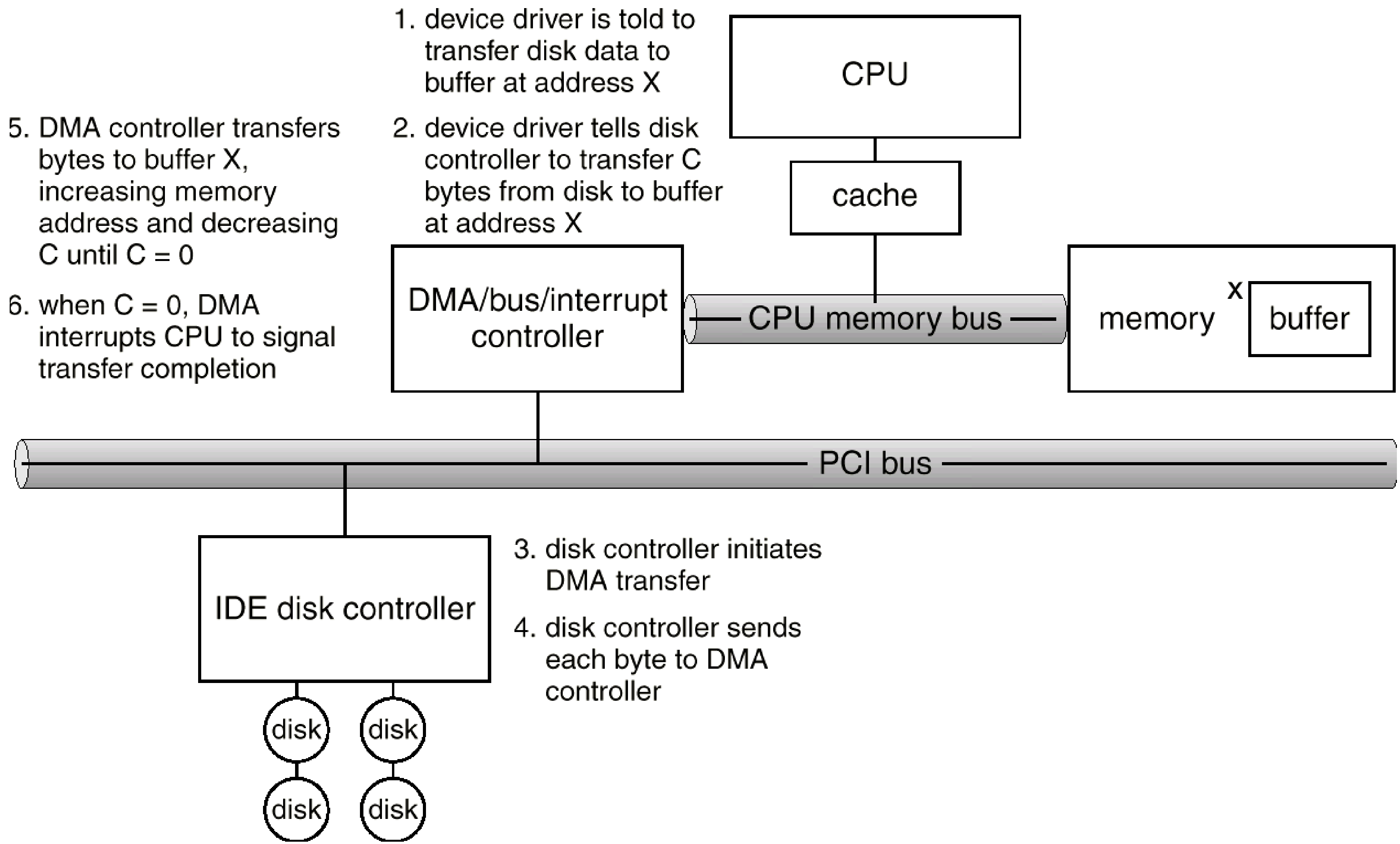
Interrupt mechanism also used for exceptions

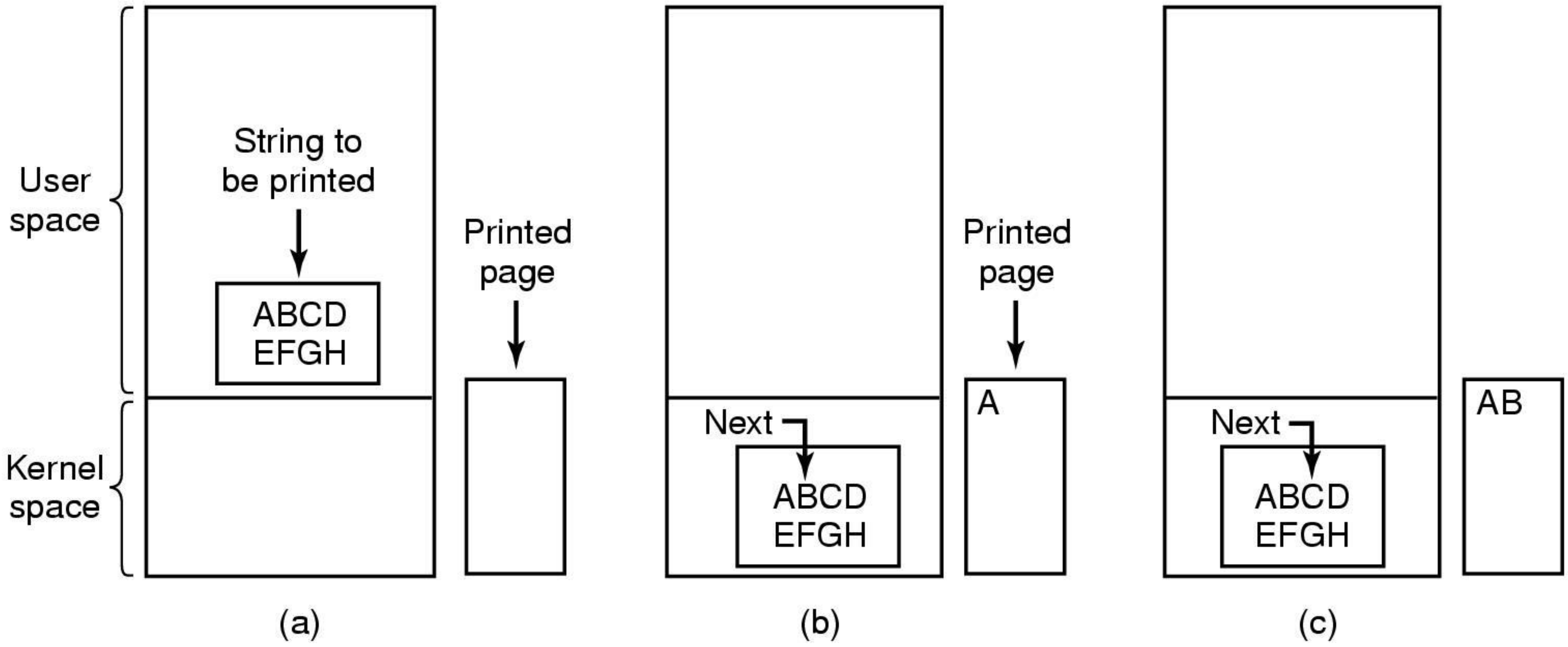
Direct Memory Access (DMA)

- I/O exchanges to memory
  - Processor grants I/O module authority to read from or write to memory
  - Relieves the processor from the task
  - Processor is free to do other things
- An interrupt is sent when the task is complete
- The processor is only involved at the beginning and end of the transfer
- Used to avoid programmed I/O for large data movement
- Requires DMA controller



Six steps to perform DMA transfer





Steps in printing a string

Programmed I/O (2)

```
copy_from_user(buffer, p, count);              /* p is the kernel bufer */
for (i = 0; i < count; i++) {                   /* loop on every character */
    while (*printer_status_reg != READY) ;      /* loop until ready */
    *printer_data_register = p[i];              /* output one character */
}
return_to_user( );
```

Writing a string to the printer using programmed I/O

Interrupt-Driven I/O

```
copy_from_user(buffer, p, count);
enable_interrupts( );
while (*printer_status_reg != READY) ;
*printer_data_register = p[0];
scheduler( );
```

(a)

```
if (count == 0) {
     unblock_user( );
} else {
     *printer_data_register = p[i];
     count = count – 1;
     i = i + 1;
}
acknowledge_interrupt( );
return_from_interrupt( );
```

(b)

- Writing a string to the printer using interrupt-driven I/O
  - Code executed when print system call is made
  - Interrupt service procedure

I/O Using DMA

```
copy_from_user(buffer, p, count);
set_up_DMA_controller( );
scheduler( );
```

(a)

```
acknowledge_interrupt( );
unblock_user( );
return_from_interrupt( );
```

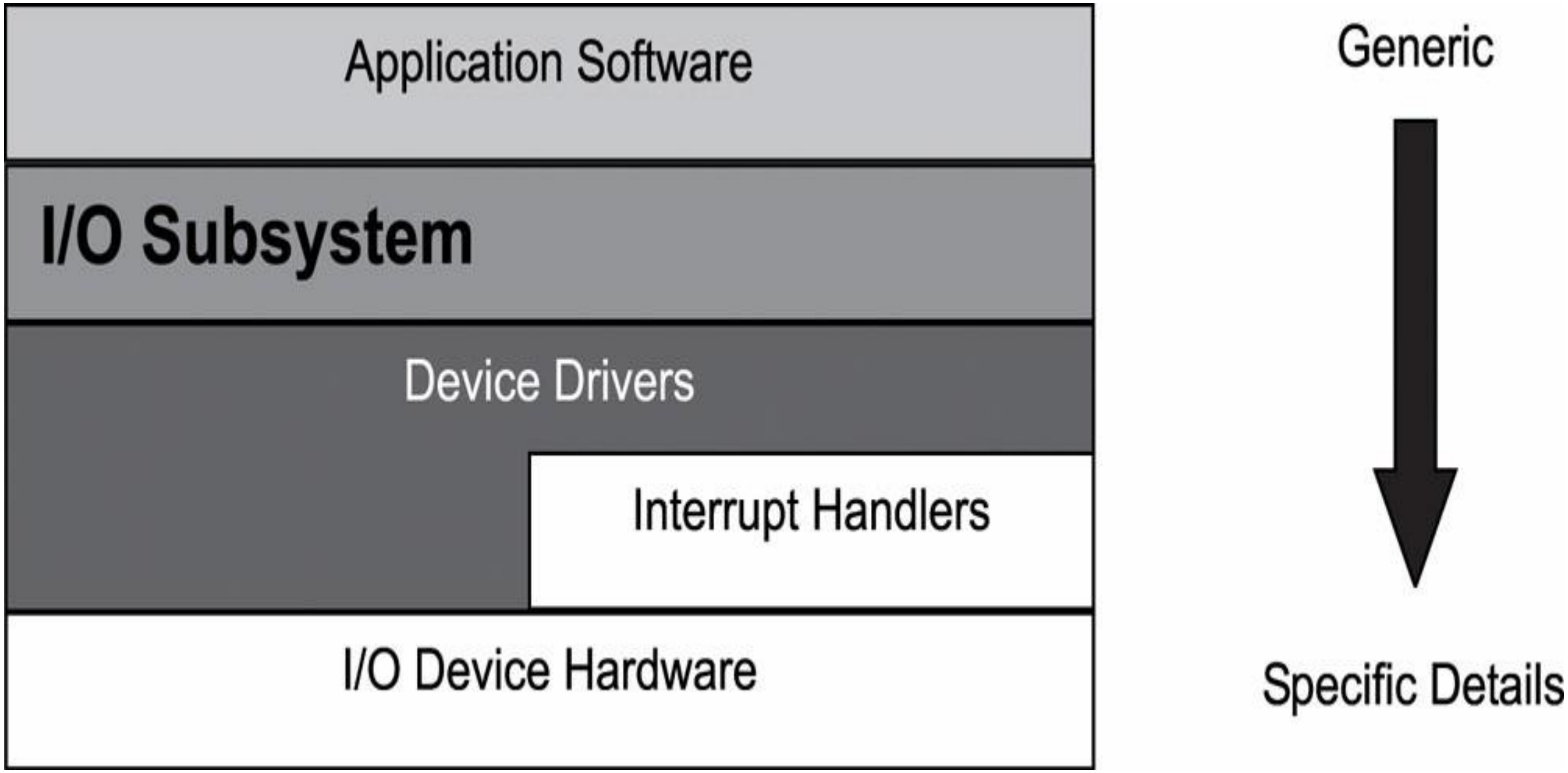
(b)

- Printing a string using DMA
  - code executed when the print system call is made
  - interrupt service procedure

Introduction to I/O

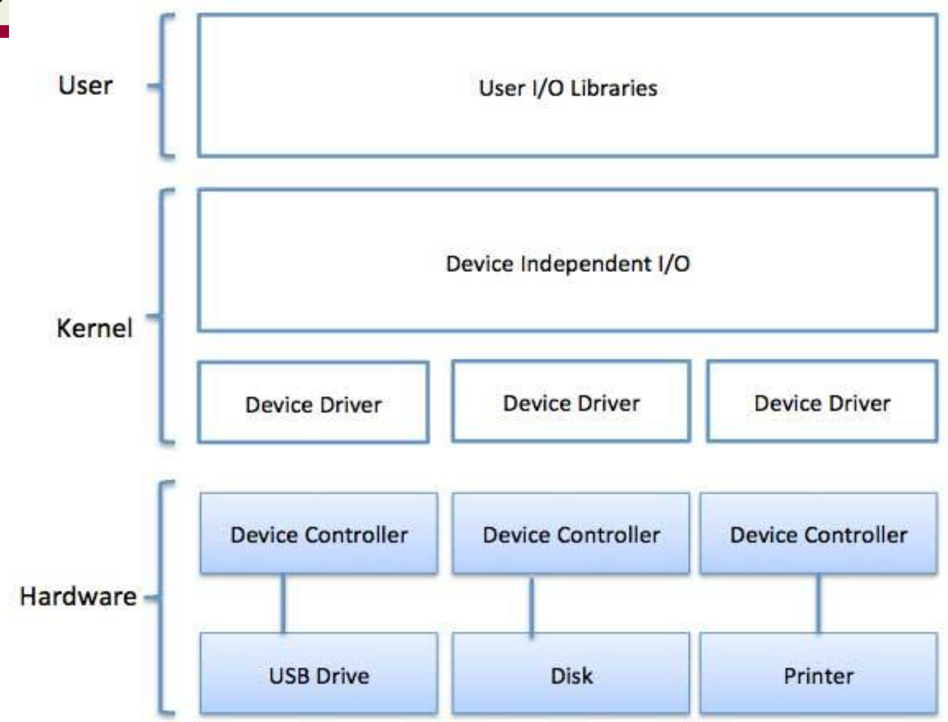
- https://www.youtube.com/watch?v=DYGrqNBWymw&ab_channel=GnanaTeja

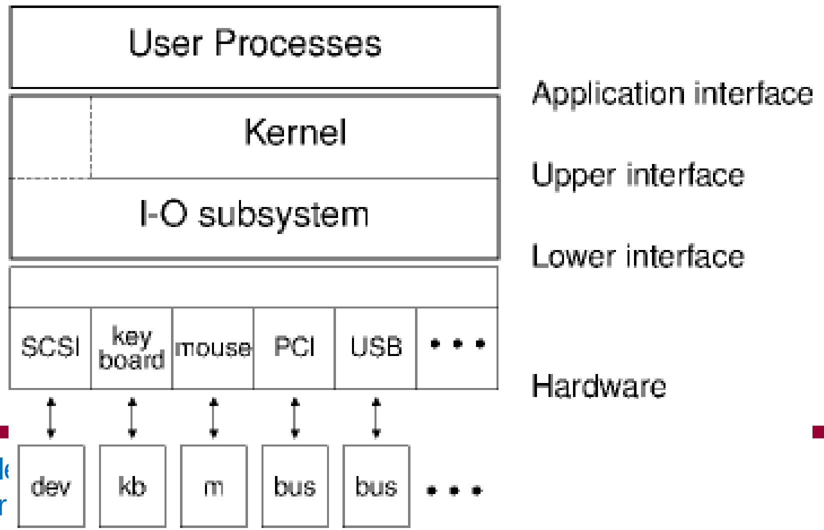
I/O Subsystem and the Layered Design



IO Subsystem

- Each I/O device driver can provide a driver-specific set of I/O API to applications
- Purpose of IO subsystem in embedded systems
  - To hide device-specific information from the kernel and from application developer
  - To provide a uniform access method to the peripheral IO devices





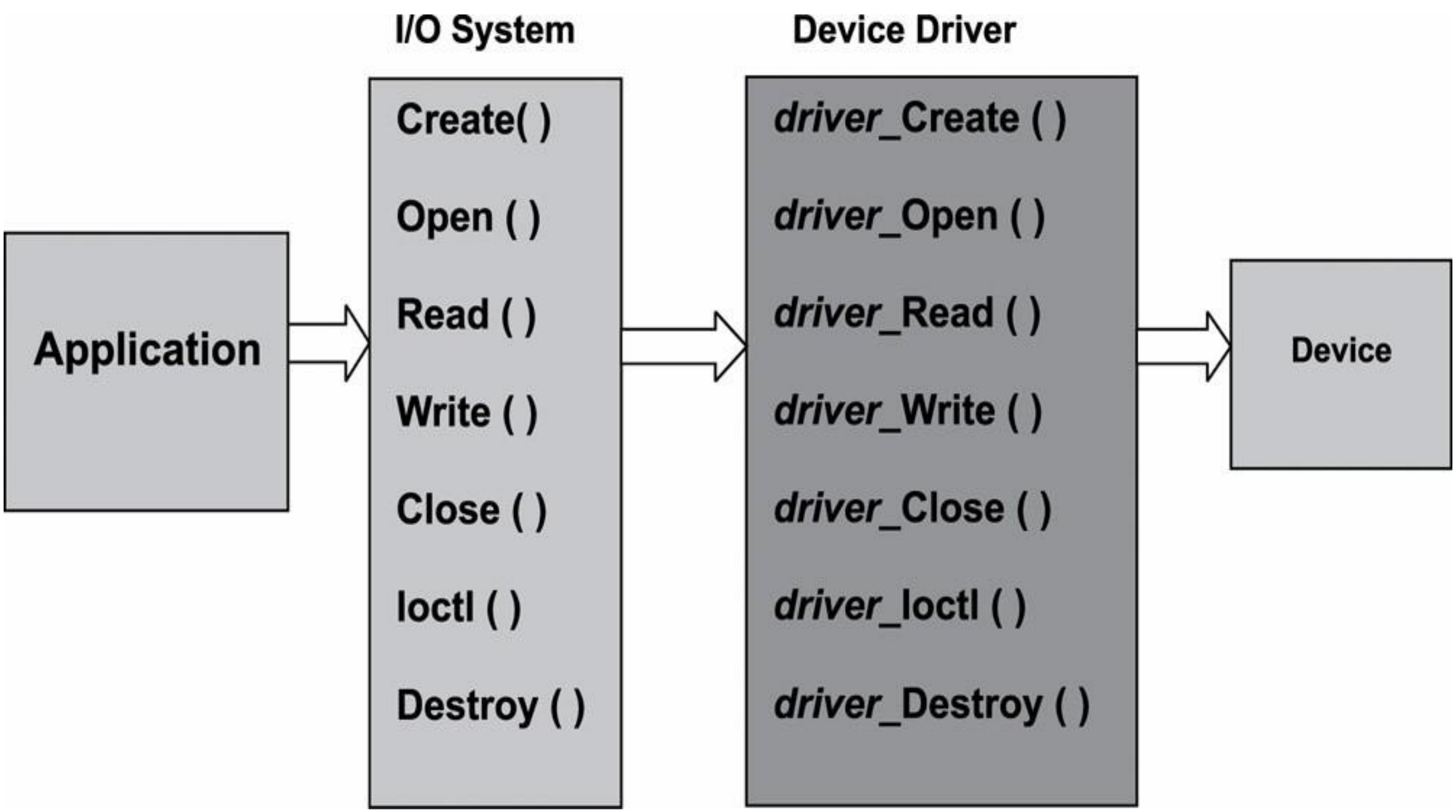
IO Subsystem

- IO subsystem defines *a standard set of functions* for IO operations
  - To hide device peculiarities from applications
- All IO device drivers conform to and support this function set
  - To provide uniform IO to applications across a wide spectrum of IO devices of varying types

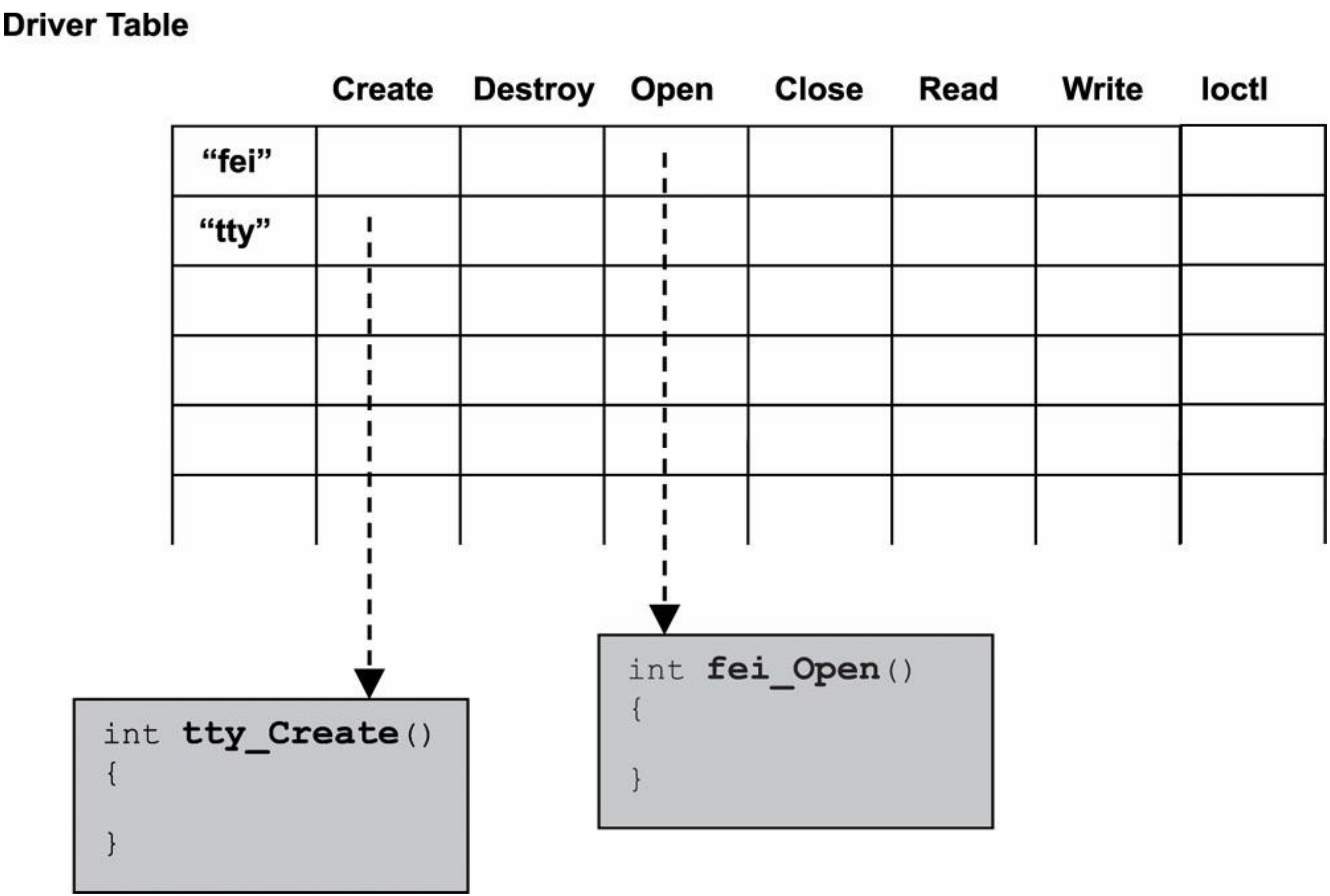
IO Functions

Function	Description
Create	Creates a virtual instance of an I/O device
Destroy	Deletes a virtual instance of an I/O device
Open	Prepares an I/O device for use.
Close	Communicates to the device that its services are no longer required, which typically initiates device-specific cleanup operations.
Read	Reads data from an I/O device
Write	Writes data into an I/O device
Ioctl	Issues control commands to the I/O device (I/O control)

IO Function Mapping



Uniform IO Driver Table



Device tree

- Devicetree is a data structure describing the hardware components of a particular computer so that the kernel can use and manage those components, including the CPUs, memory, buses and peripherals
- It is a tree of named nodes and properties. Nodes contain properties and child nodes, while properties are name–value pairs

Block and Character Devices

- Block devices include disk drives
  - Commands include read, write, seek
  - Raw I/O or file-system access
  - Memory-mapped file access possible
- Character devices include keyboards, mice, serial ports
  - Commands include `get, put`
  - Libraries layered on top allow line editing

Kernel I/O Subsystem

- *Scheduling*
  - Efficiency, fairness, prioritized access
- *Buffering* - store data in memory while transferring between devices
  - To cope with device speed mismatch, transfer size mismatch and to maintain “copy semantics”
- *Caching* - fast memory holding copy of data
  - Always just a copy, Key to performance
- *Spooling* - hold output for a device
  - Used when device can serve only one request at a time, i.e., Printing
- *Device reservation* - provides exclusive access to a device
  - System calls for allocation and deallocation
  - Watch out for deadlock

Error Handling

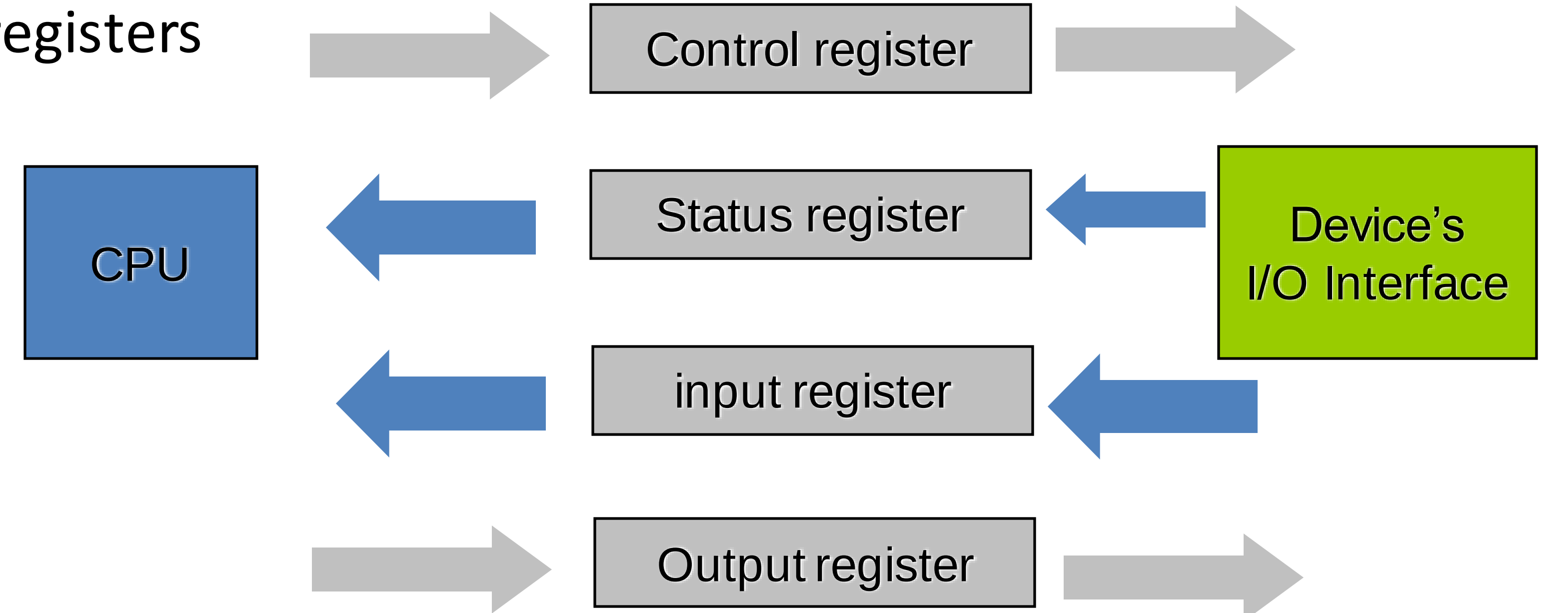
- OS can recover from disk read, device unavailable, transient write failures
- Most return an error number or code when I/O request fails
- System error logs hold problem reports

Improving Performance

- *I/O a major factor in system performance*
  - Demands CPU to execute device driver, kernel I/O code
  - Context switches due to interrupts
  - Data copying
  - Network traffic especially stressful
- Improving Performance
  - Reduce number of *context switches*
  - Reduce *data copying*
  - Reduce *interrupts* by using large transfers, smart controllers, polling
  - Use DMA
  - Balance CPU, memory, bus, and I/O performance for highest throughput

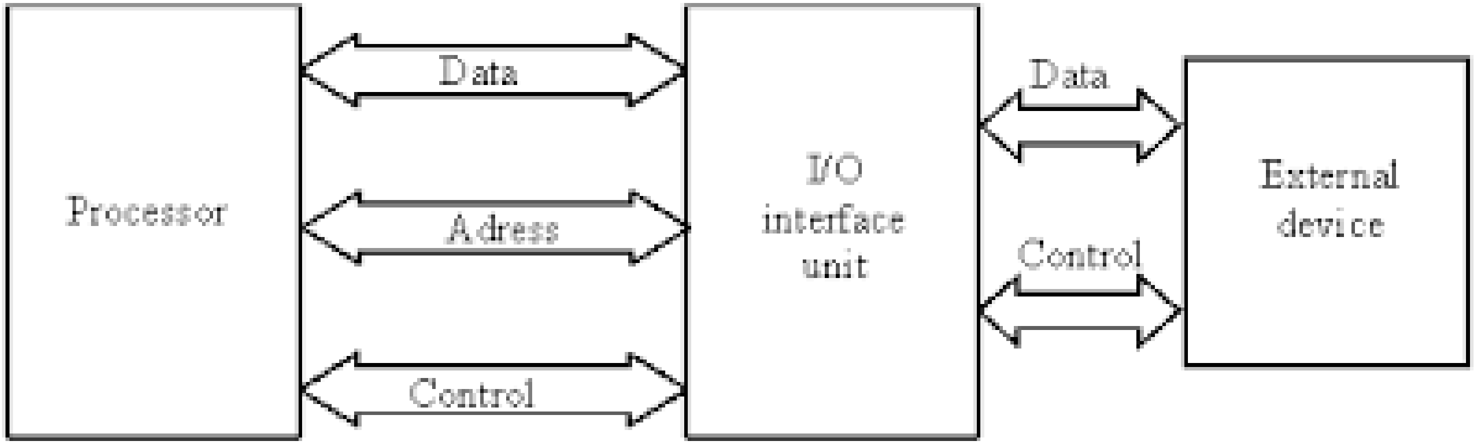
I/O and registers

- The I/O ports of each device are actually structured into a set of specialized registers
- CPU write commands to control registers
- CPU reads a value that represents the internal state of the device from status register
- CPU fetch data from a device by reading input register
- CPU push data to device by writing bytes into output registers



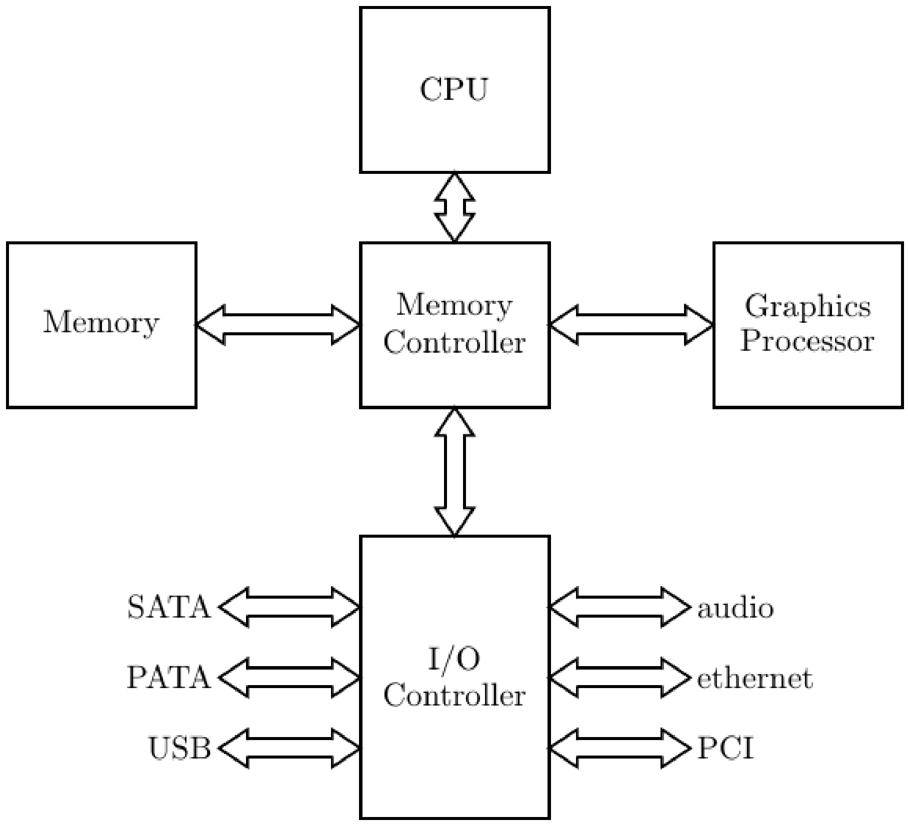
What is an I/O interface?

- An I/O interface is a hardware circuit inserted between a group of I/O ports and the corresponding device controller.
- It acts as an interpreter that translates the values in the I/O ports into commands and data for the device.



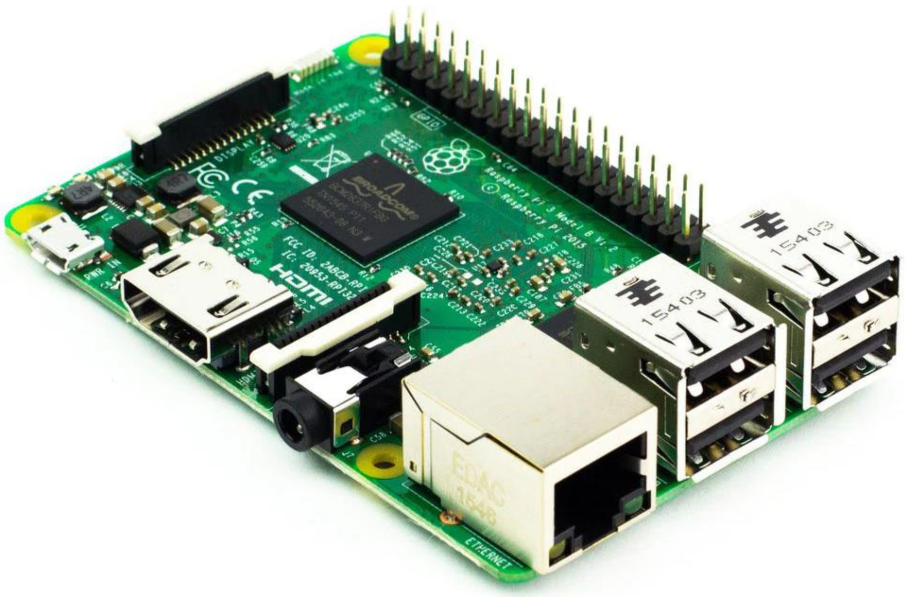
Types of I/O interfaces

- Custom I/O interfaces
  - devoted to one specific hardware device
  - usually, the device controller is on the same card of I/O interfaces.
- General-purpose I/O interface (GPIO)
  - used to connect several different hardware devices



Custom I/O interfaces

- Keyboard interface
- Graphic interface
- Disk interface
- Network interface



- https://www.coursera.org/lecture/iot-architecture/device-drivers-AL7YG
- Windows: Device Driver Development Tutorial – YouTube

https://www.youtube.com/watch?v=jC0B2kSyKAI

- Linux: Learning Linux Device Drivers Development : Find and Create Network Drivers | packtpub.com - YouTube

https://www.youtube.com/watch?v=4Njw_VJUhZ4

Practice: device driver development

- https://github.com/PacktPublishing/Linux-Device-Drivers-Development