CSCI 360
Introduction to Operating Systems

I/O System

Humayun Kabir
Professor, CS, Vancouver Island University, BC, Canada

Outline

- I/O Concepts
  - I/O Devices
  - Device Controllers
  - I/O Ports
  - Memory Mapped I/O
  - Programmed I/O
  - Interrupt Driven I/O
  - Direct Memory Access (DMA)
  - I/O Using DMA
- I/O Software Layers
  - User I/O Layer
  - Device Independent I/O Layer
  - Device Driver
  - Interrupt Handler
- I/O Buffering

I/O Devices

- Mainly 2 types of I/O devices
  - Block Devices: Hard Disk, Blue-ray Disk, and USB Stick
  - Character Devices: Printer, Network Interface Card, and Mouse.

I/O Devices

- Block Devices
  - Stores information in fixed-size blocks, each one with its own address.
  - Transfers are in units of entire blocks.
  - Allows to read or write each block independently.

I/O Devices

- Character Devices
  - Transfers stream of characters, without regard to block structure
  - Not addressable, does not have any *seek* operation

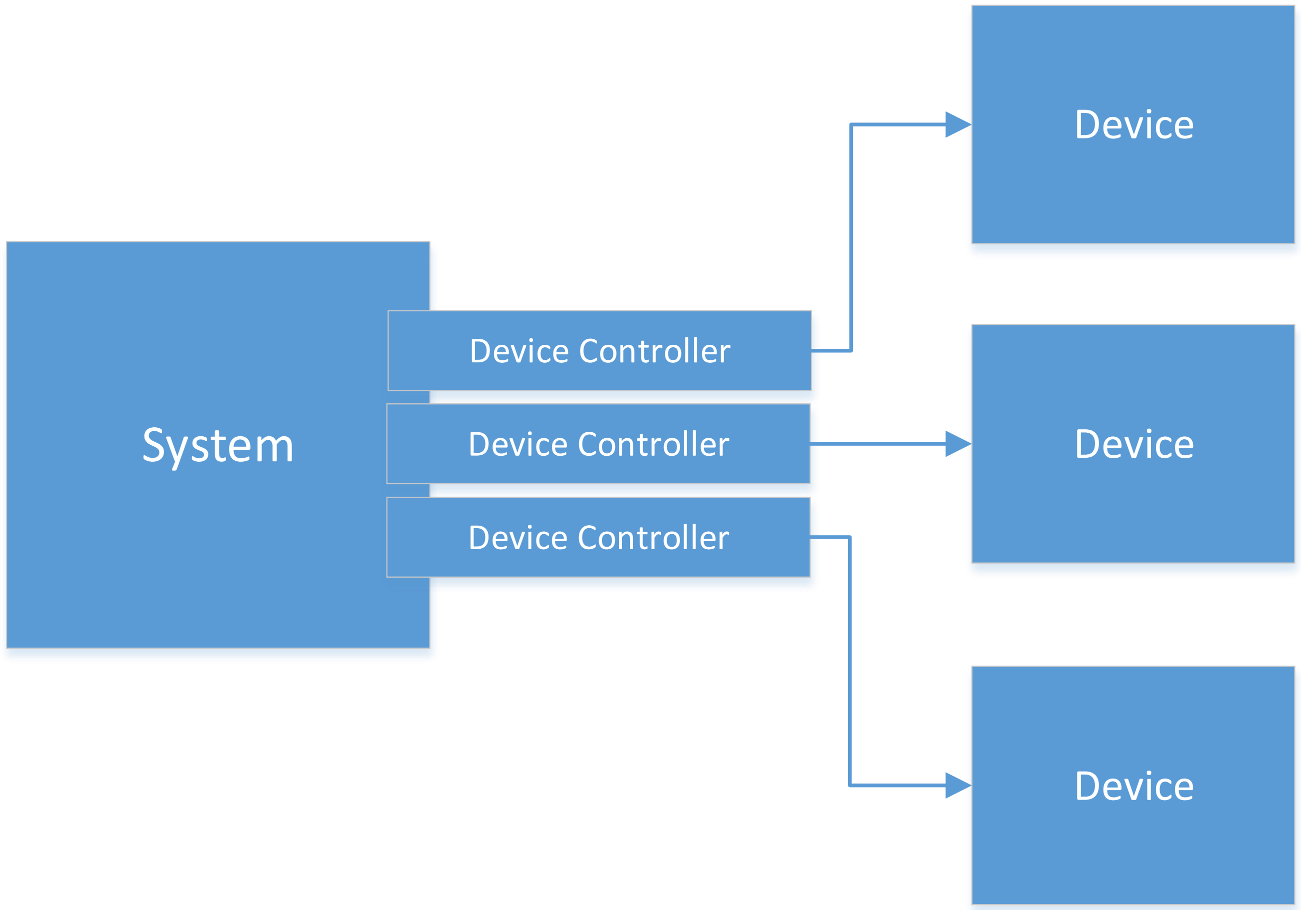
I/O Devices

Come with fixed **data rate**.

Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Scanner at 300 dpi	1 MB/sec
Digital camcorder	3.5 MB/sec
4x Blu-ray disc	18 MB/sec
802.11n Wireless	37.5 MB/sec
USB 2.0	60 MB/sec
FireWire 800	100 MB/sec
Gigabit Ethernet	125 MB/sec
SATA 3 disk drive	600 MB/sec
USB 3.0	625 MB/sec
SCSI Ultra 5 bus	640 MB/sec
Single-lane PCIe 3.0 bus	985 MB/sec
Thunderbolt 2 bus	2.5 GB/sec
SONET OC-768 network	5 GB/sec

Device Controller

Device Controllers connect devices to the systems



Device Controller

- Each Device Controller has **control registers** that the system can use to write **control commands** to the device.
- Control registers can also be read to know the **status** of the device.
- Some devices may have **data buffer** in addition to control registers.
- Control registers and data buffer can be addressed in two ways:
  - Using **I/O port numbers**
  - **Mapping to memory** addresses.

I/O Port Numbers

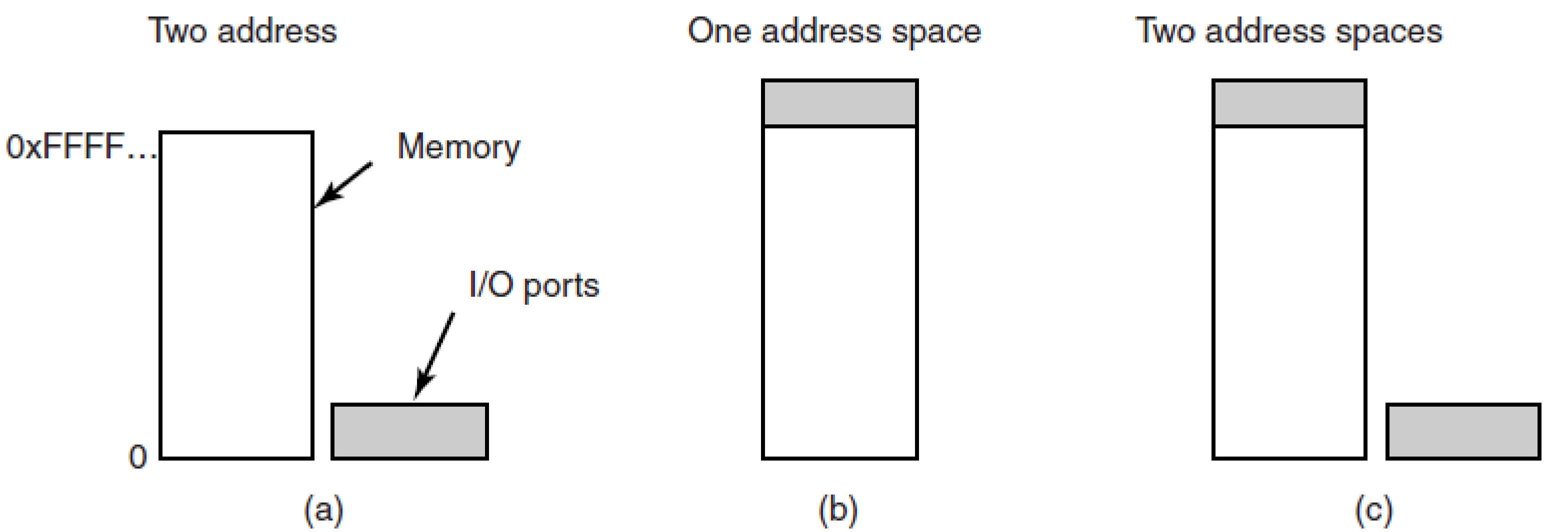
- Controller registers are assigned 8 or 16-bit port numbers to address.
- **I/O port space** is separate from memory address space.
- System access I/O ports by using special I/O instructions.

  IN REG PORT

  OUT PORT REG

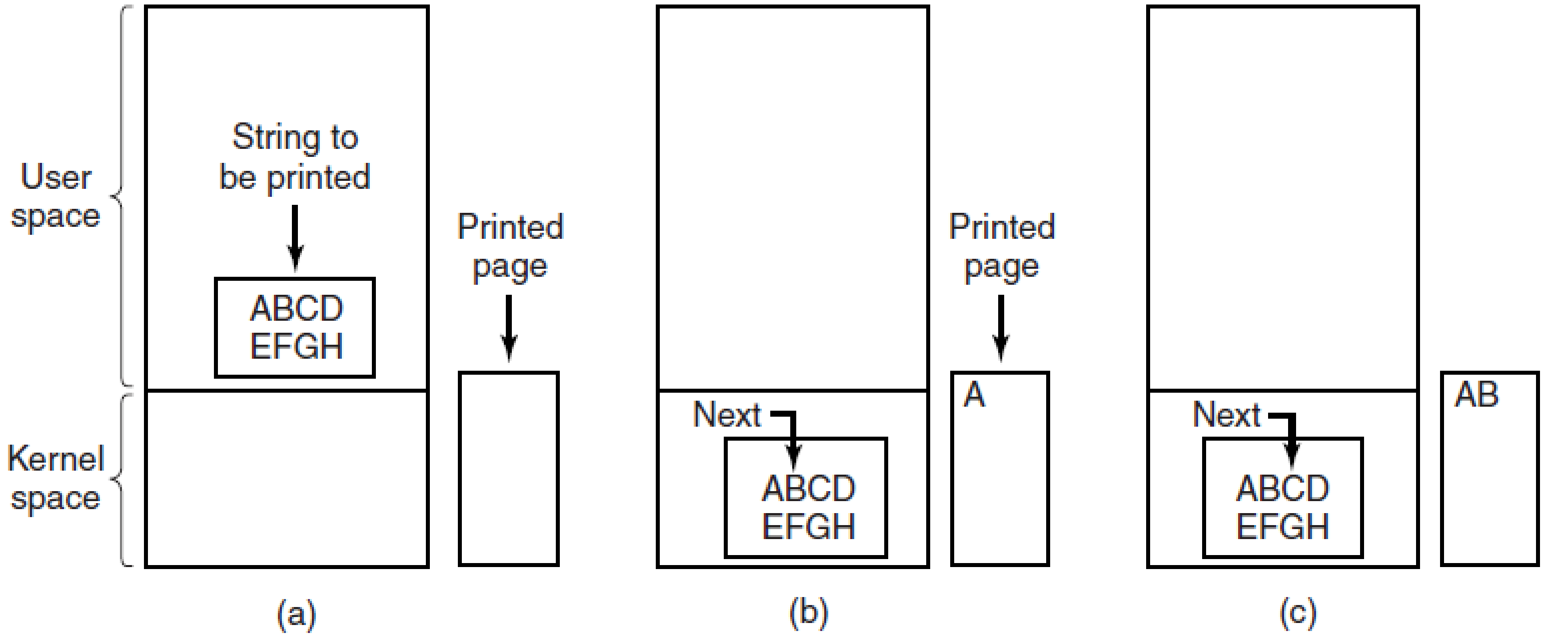
Memory-Mapped I/O

Each control register is mapped to a unique memory address to which no memory is assigned.



(a) Separate I/O and memory space.
(b) Memory-mapped I/O. (c) Hybrid.

Programmed I/O



Steps in printing a string.

Programmed I/O

```
copy_from_user(buffer, p, count);              /* p is the kernel buffer */
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.

Programmed I/O

- Programmed I/O is **Synchronous** or **Blocking**.
- CPU is busy with I/O operation until the I/O transfer is complete.

Interrupt Driven I/O

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

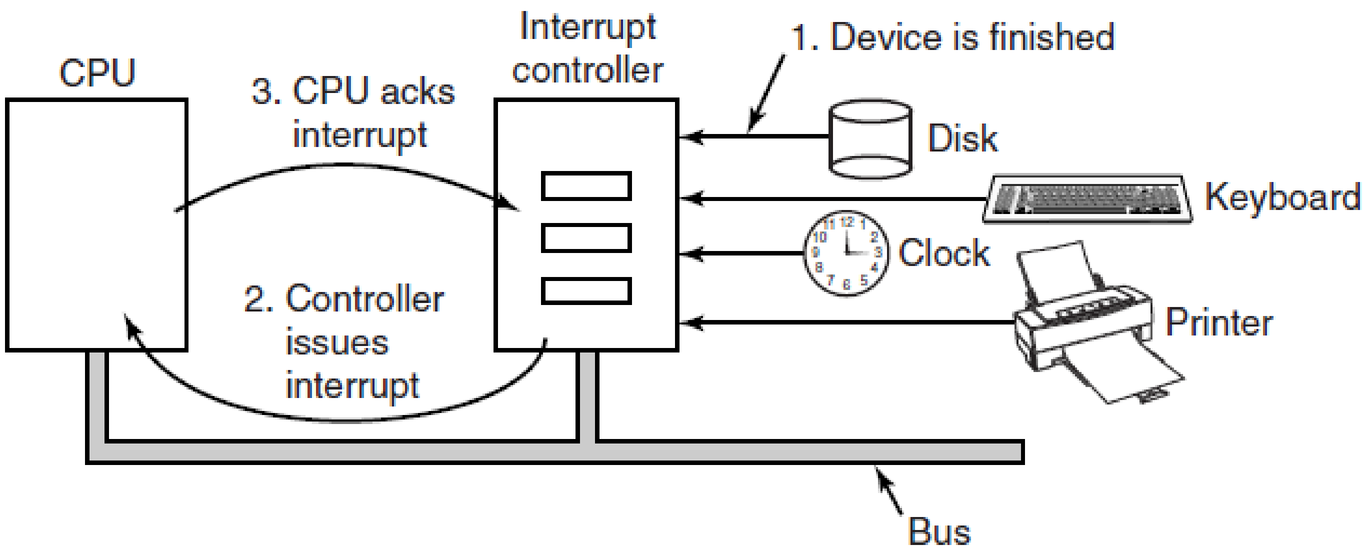
(a)

```
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. (a) Code executed at the time the print system call is made. (b) Interrupt service procedure for the printer.

Interrupt Driven I/O



- Interrupt driven I/O is **asynchronous** or **non-blocking**.
- CPU proceeds with other jobs until interrupted by the device controller.

Interrupt-Driven Disk I/O

- System writes a read command on **disk controller**.
- Disk controller
  - Reads the data block from the drive serially, bit by bit, until the entire block is in the controller's **internal buffer**.
  - Computes the **checksum** to verify that no read errors have occurred.
  - Assert **interrupt** to the CPU
- System (**Interrupt Handler**) transfers data from the controller buffer to the memory.

Direct Memory Access

- Getting I/O data one byte at a time wastes CPU time.
- Using **Direct Memory Access (DMA)** CPU time waste is avoided.
- System needs a **DMA Controller**, which has direct access to the system bus to transfer data from I/O buffer to memory without involving CPU.
- DMA Controllers come with **control registers**, **memory address registers**, and **byte count register**.

I/O Using DMA

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

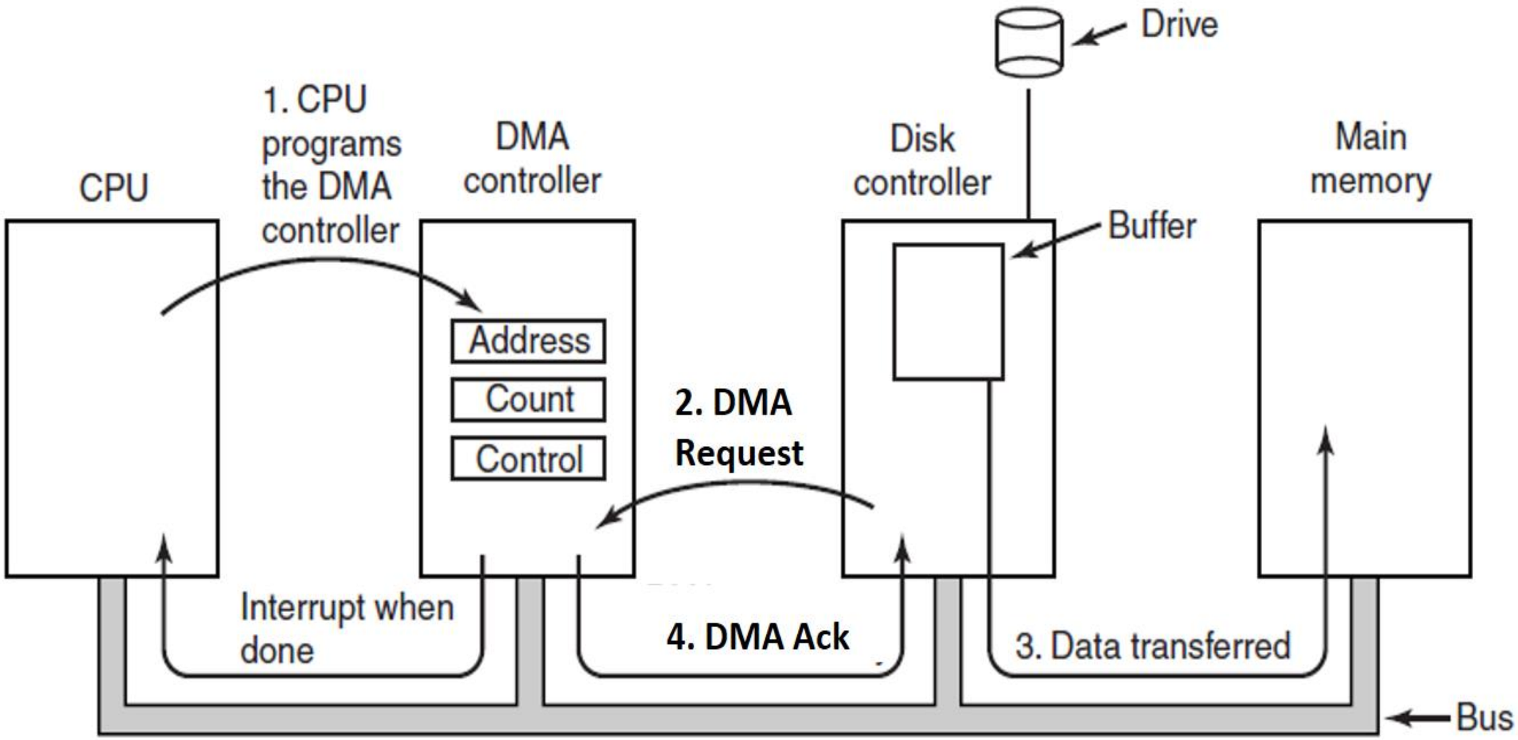
(a)

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

(b)

Printing a string using DMA. (a) Code executed when the print system call is made. (b) Interrupt service procedure.

Disk I/O with DMA



Operation of a DMA transfer.

Disk I/O with DMA

- System instructs DMA controller by setting **source** and **destination** addresses and the **byte count**.
- System also instructs the **disk controller** to read a block of data from the disk.
- The disk controller reads the whole block into its **internal buffer** and asserts a **DMA request** to DMA controller.
- DMA Controller requests for system bus access.

Disk I/O with DMA

- DMA controller completes the **data transfer** from the disk controller buffer to the memory after acquiring system bus access.
- Once the transfer is complete, DMA controller asserts **DMA acknowledgement** to the disk controller and **interrupt** to the system.
- System (**interrupt handler**) asserts **interrupt acknowledgement** to DMA controller and unblock the user process that was waiting for the I/O to complete.

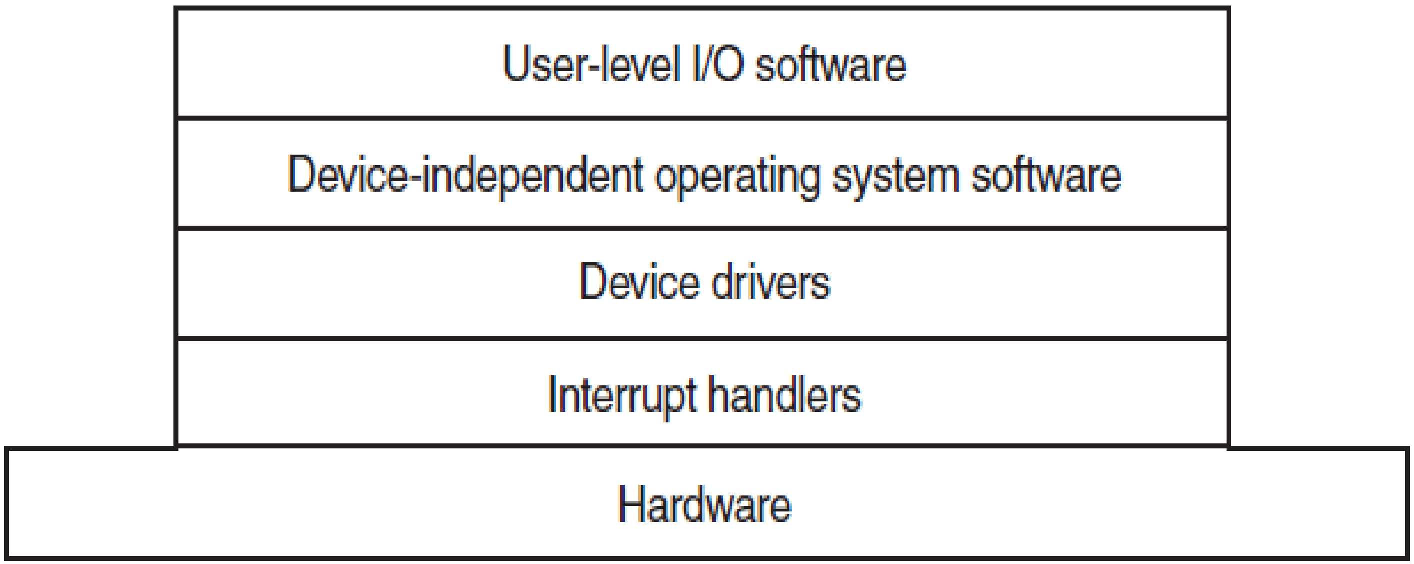
Goals of the I/O Software

- **Device independence**
  - Similar methods to access different types of devices.
- **Uniform naming**
  - Similar naming scheme for different types of devices.
- **Error handling**
  - Device controller must handle and conceal as many errors as possible

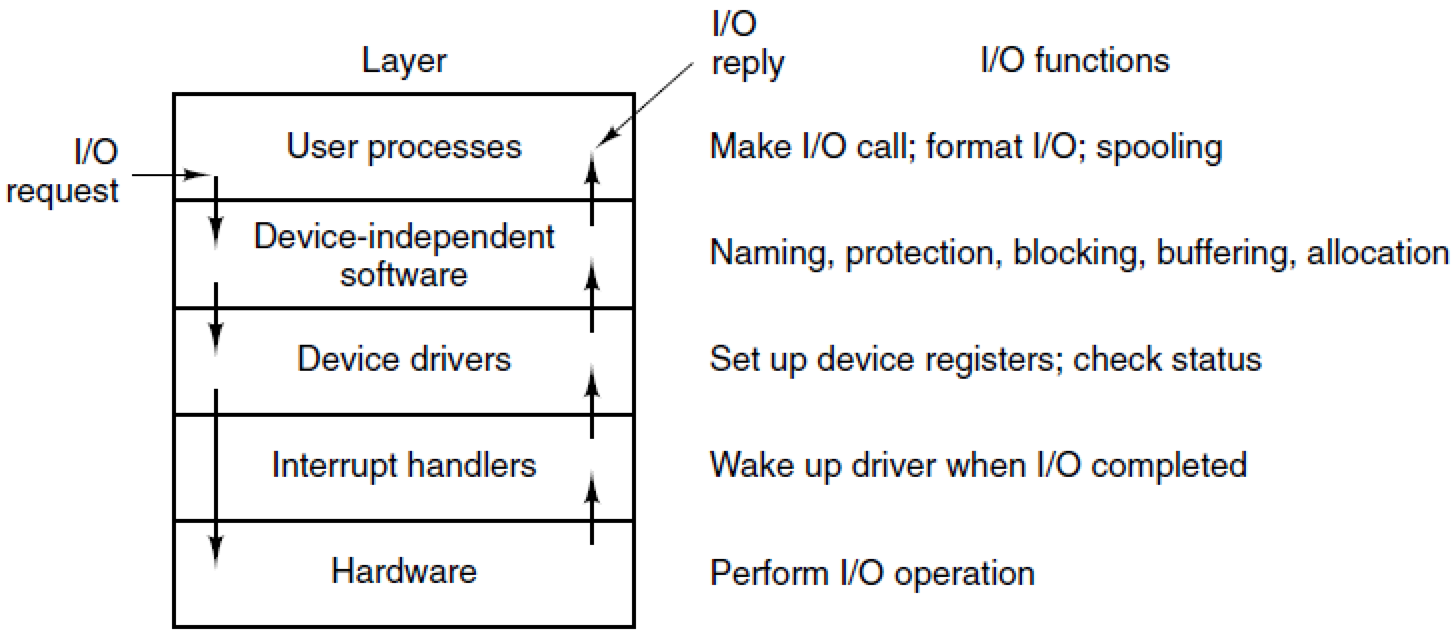
Goals of the I/O Software

- I/O operation may be **synchronous** (blocking) or **asynchronous** (interrupt driven)
- **Buffering**
  - Device controller should employ buffer to decouple system and I/O speeds

I/O Software Layers



User-Space I/O Software



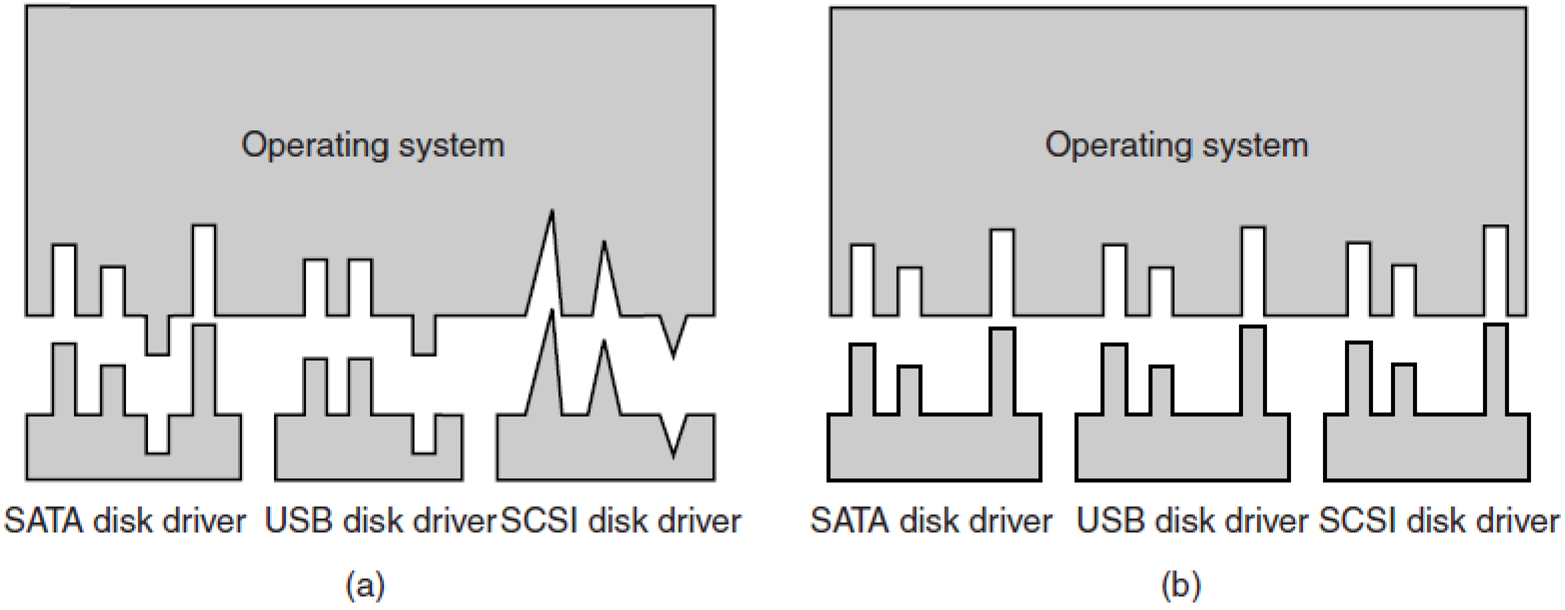
Layers of the I/O system and
the main functions of each layer.

Device-Independent I/O Software

Uniform interfacing for device drivers
Buffering
Error reporting
Allocating and releasing dedicated devices
Providing a device-independent block size

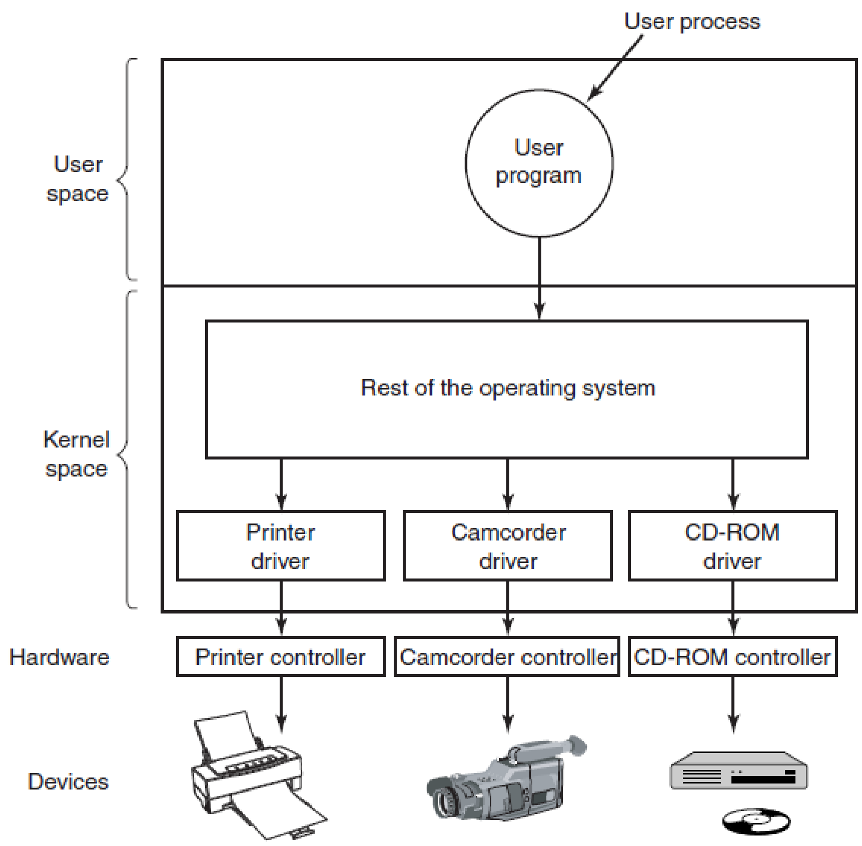
Functions of the
device-independent I/O software.

Uniform Interfacing for Device Drivers



(a) Without a standard driver interface.
(b) With a standard driver interface.

Device Drivers



- Logical positioning of device drivers.
- In reality all communication between drivers and device controllers goes over the bus.

Interrupt Handlers

- **Interrupt hardware** flips the mode bit in **PSW** to **kernel mode**.
- Pushes **PC** onto **stack**.
- Jumps to the **interrupt handler** corresponds to the **interrupt vector**.

Interrupt Handlers

Interrupt handler (I/O software) steps:

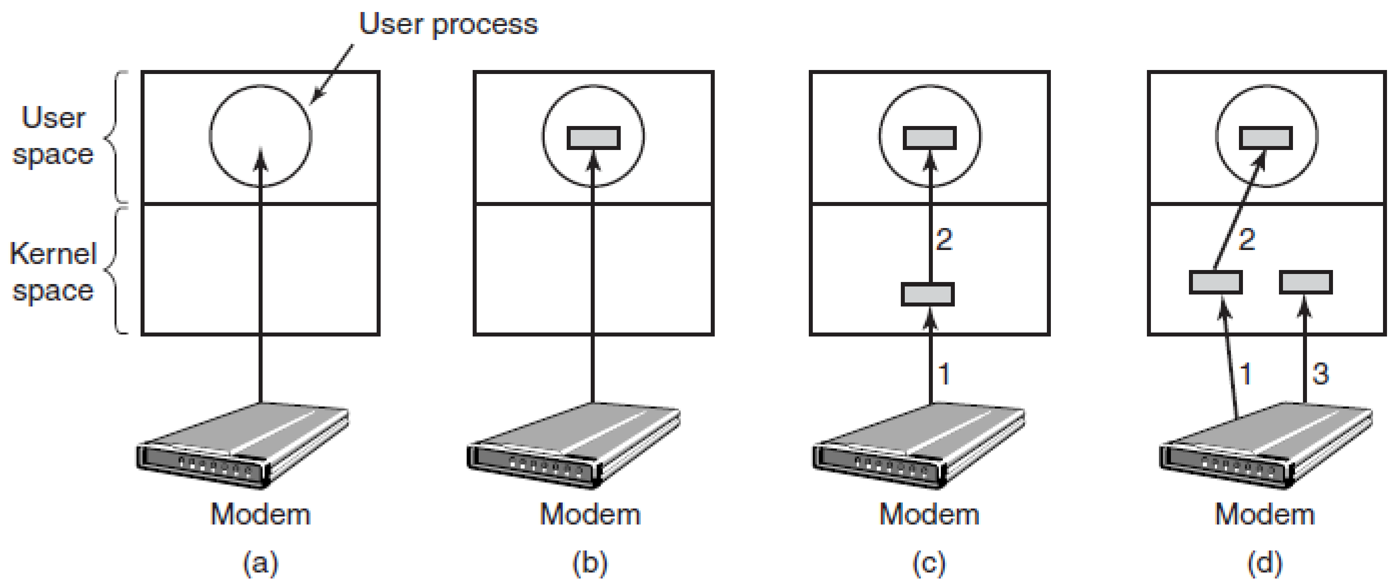
1. Pushes **registers** (including the **PSW**) that are not saved by interrupt hardware onto the stack.
2. Set up context for **interrupt service procedure**.
3. Set up a **stack** for the interrupt service procedure.
4. Acknowledge interrupt controller. If no centralized interrupt controller, re-enable interrupts.
5. Copy saved registers from the stack into **process table**.

Interrupt Handlers

Interrupt handler (I/O software) steps:

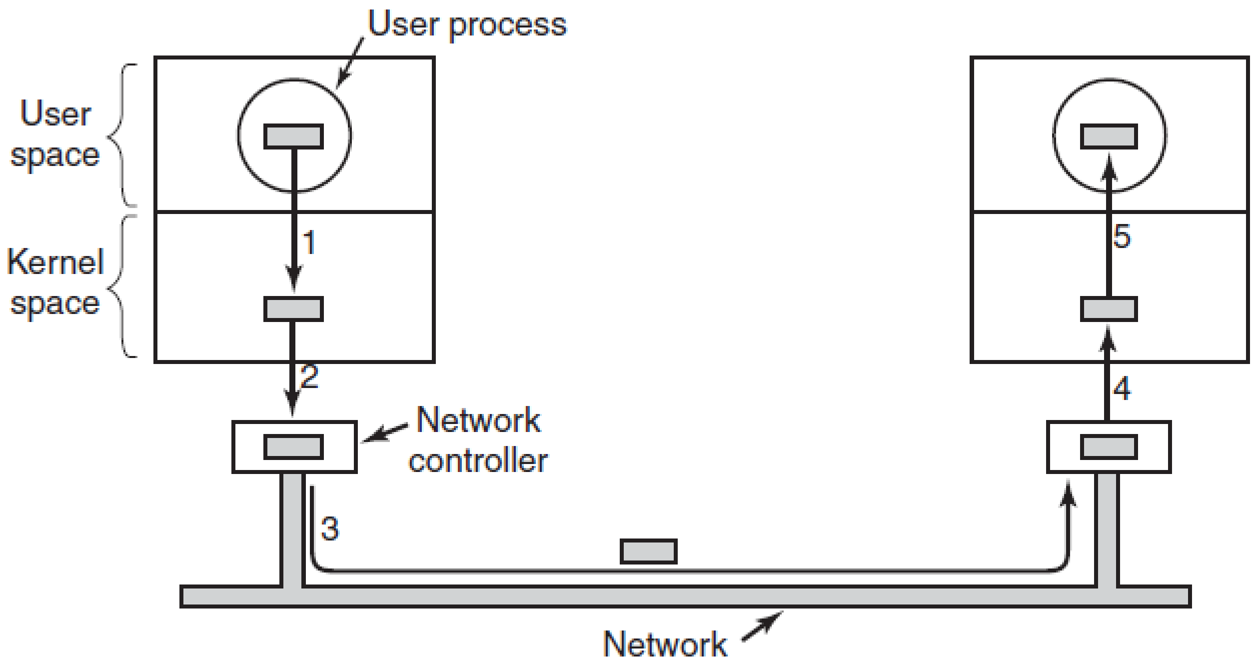
6. Run **interrupt service procedure**. Extract information from interrupting device controller's registers.
7. Get **next process** to run from CPU scheduler.
8. Set up the **MMU context** for the next process to run.
9. Load new process's registers, including its PC, PSW.
10. Return from interrupt calling IRET, as a consequence hardware flips the mode bit to **user mode**.
11. Start running the new process.

Buffering



(a) Unbuffered input. (b) Buffering in user space. (c) Buffering in the kernel followed by copying to user space. (d) Double buffering in the kernel.

Buffering



Networking may involve many copies of a packet.

Summary

- I/O Concepts
  - I/O Devices
  - Device Controllers
  - I/O Ports
  - Memory Mapped I/O
  - Programmed I/O
  - Interrupt Driven I/O
  - Direct Memory Access (DMA)
  - I/O Using DMA
- I/O Software Layers
  - User I/O Layer
  - Device Independent I/O Layer
  - Device Driver
  - Interrupt Handler
- I/O Buffering

Next

Protection

- Protection Domain
- Access Control List
- Capabilities