ATISHAY JAIN

20CS30008

COA

ATISHAY JAIN
20CS30008

I/O device

The computer must be able to move data between itself and the outside world. The computer's operating system environment consists of the devices that serves as either sources or destination of data. When data is received from or delivered to a device that is directly connected to a computer, the process is known as input-output (I/O) and the device is refferred to as peripheral device.

⇒ From the point of view of a CPU, peripheral devices are viewed similar to memory. CPU sends some address through the Address Bus (not as flow, but like setting and unsetting some bits) After that, the CPU sets some specific control line to specify if the operation to be performed at the address is Input or Output.

In case of Input, the Data Bus bits are set by the peripheral device and the CPU can then decode them to read the input from the device.

In case of output, the CPU sets the bits of Data Bus and also sets specific control lines corresponding to which the bits on data bus alters the bits at the specified address (present in Address Bus) of the Peripheral device.

The interaction is very similar to the interaction b/w CPU and RAM. CPU simply has address for each Input/Output device which are unique among themselves and also different from the address of memory units, specifying a particular I/O device.

Programmer's View

The OS provides a high level interface to devices, greatly simplifying the programmer's job.

→ Standard interfaces are provided for related devices.

→ Device dependencies are encapsulated in device drivers.

→ New devices can be supported by providing new device drivers.

Device Characteristics:

→ Transfer Unit: Character or Block

→ Access Method: Sequential or Random Access

→ Timing: synchronous or asynchronous
Most devices are asynchronous, while I/O system calls are synchronous.
⇒ The OS implements blocking I/O.

→ Shareable and dedicated

→ Speed

→ Operations: Input, output or both

→ Examples: key board (sequential, character), disk (block, random or sequential)

Block and Character devices.

→ Block devices include disk drives

→ Character devices include keyboards, mice, serial ports.

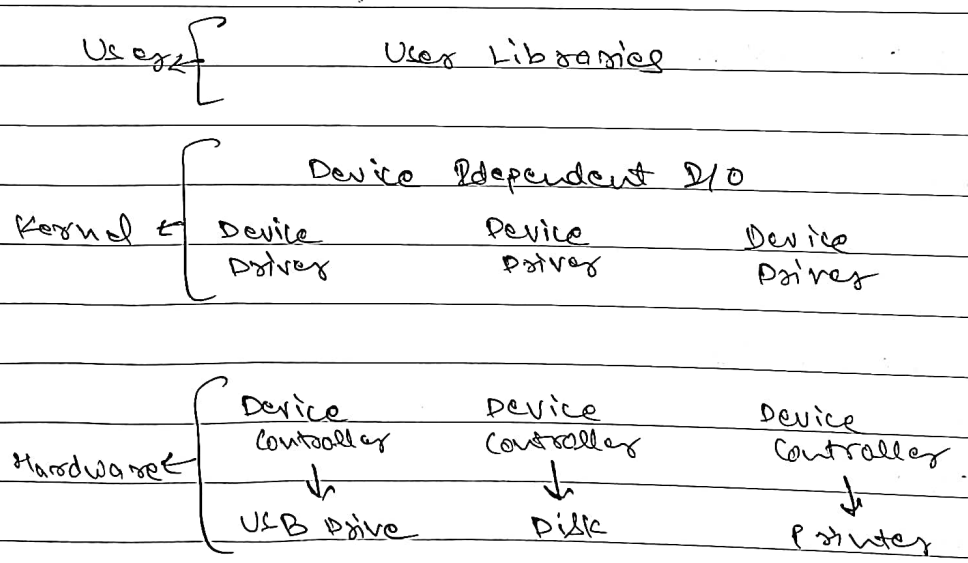
I/O software

I/O software is organised in the following layers:

→ User level libraries
→ Kernel level modules
→ Hardware

A key concept in the design of I/O software is that it should be device independent where it should be possible to write programs that can access any I/O device without having to specify the device in advance.

For instance, a software that accepts a file as input should be able to read a file from a CD-ROM, a hard drive or a floppy disc without requiring any modifications.



Architecture of I/O device

⇒ Key Components:

i) System Bus: It allows the device to communicate with the CPU. Typically, it is shared by multiple devices.

ii) A device port typically consisting of 4 registers:
- → status: It indicates a device busy, data ready or error condition
- → control: command to perform
- → Data-in: The data being sent from the CPU to the device.
- → Data-out: data being sent from the CPU to the device.

iii) Controller: received the commands from the system bus, translates them into device actions, and reads/write data onto the system bus.

iv) The device itself.

⇒ Traditional I/O devices:

i) Printer

ii) Keyboard

iii) Mouse

⇒ Modern I/O devices:

i) Joystick

ii) Touch-screen

iii) Robot actuators

Device Drivers

These are device drivers that can be plugged into an operating system to handle a particular device. It offers a common interface, encapsulate device-dependent code, and allow programmes to read/write device specific registers. These are typically created by the device's manufacturer and shipped on a CD-ROM with the device

It performs the following jobs.

→ Accepts request from the device independent software
→ Interact with the device controller to provide necessary error handeling and give I/O.
→ ensuring that the request is successfully carried out

Interrupt Handlers

An interrupt handler is a piece of software or more specifically a callback function in an OS whose execution is triggered by the reception of an interrupt.
The interrupt procedure does whatever it has to in order to handle the interrupt.

Device Independent I/O Software

The basic function this is to perform the I/O functions that are common to all devices and to provide a uniform interface to the user level software. We can not write completely device Independent software but use some common modules.

User-Space I/O software

These libraries offer a richer and more user-friendly interface to access the functionality of the kernel or ultimately interactive with the device drivers.

Kernel I/O subsystem

Kernel I/O subsystem is responsible to provide many services related to I/O.

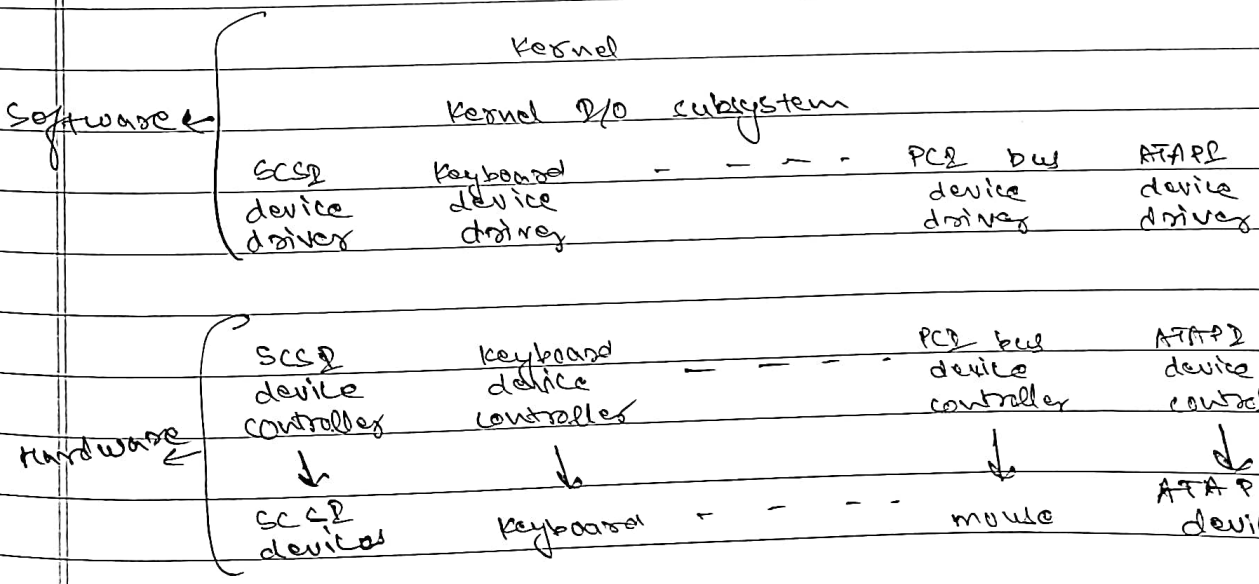
→ **Scheduling**: Kernel schedules a set of I/O requests to determine a good order to execute them to increase efficiency.

→ **Buffering**: It maintains a memory known as buffer that stores data temporarily while they are transferred to cope with mismatching speed.

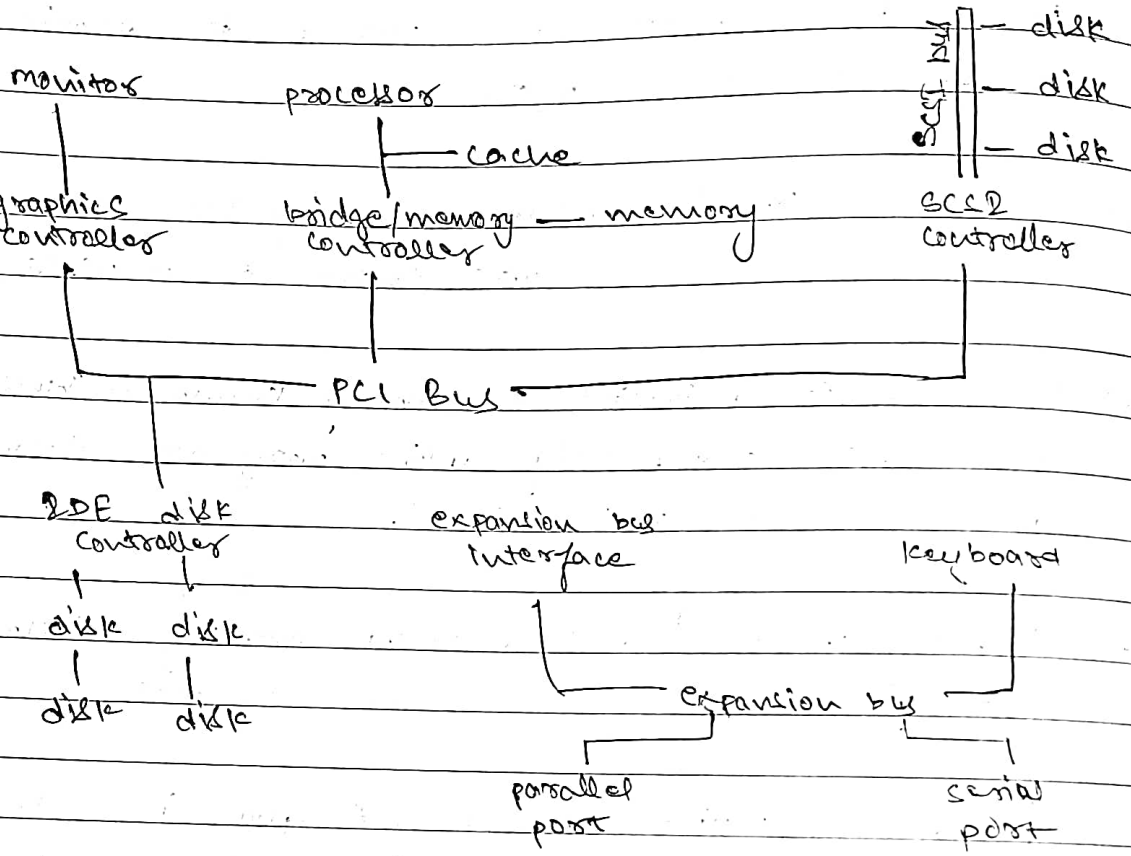
→ **Caching**: It maintains a cache memory which is region of fast memory, that holds copies of data

→ **Spooling and Device Reservation**: A spool is a buffer that holds the output for a devices that can not accept interleaved data streams.

→ **Error Handling**: Guard against many errors.



PCI bus structure



Interaction b/w CPU, Keyboard and Speakers

Step-1: CPU sets address bus to address of keyboard and set control lines to read from keyboard.

Step-2: The keyboard corresponding to the current control lines signal writes on the data bus which are almost simultaneously available to the CPU. (mostly stored in some register).
NOTE that only the device/memory location available on the address bus is allowed to read/write from the data bus.

Step-3: The CPU sets address Bus to the address of speakers, and set control lines to "write on speakers". It would also sets the data bus to the data it received from keyboard earlier (the data was probably stored in some register).

Step-4: The speakers, corresponding to the current control signal received the bits from data Bus. (The receiving is generally done by over-writing bits in some flip-flop inside the monitor.)

NOTE: There are also other schemes of implementing I/O devices where we have different data, address and control lines for memory and I/O devices.

A high-level diagram: