Operating System

Chapter 3

I/O SYSTEMS

Khoa Công Nghệ Thông Tin
Trường Đại Học Khoa Học Tự Nhiên
ĐHQG-HCM

GV: Thái Hùng Văn

Outline

- I/O Devices
- I/O System
- I/O Bus & Port
- Device Drivers
- Device Controllers
- Device Independent
- Communication to I/O Devices
- Polling vs Interrupts
- BIOS & UEFI
- I/O Software Layers
- Polling vs Interrupts I/O
- I/O Request Process
- Popular Signal and I/O Devices
- Typical Ports Rate
- Disk Hardware
- Disk RAID
- Disk Arm Scheduling

Basic Concepts – I/O Device

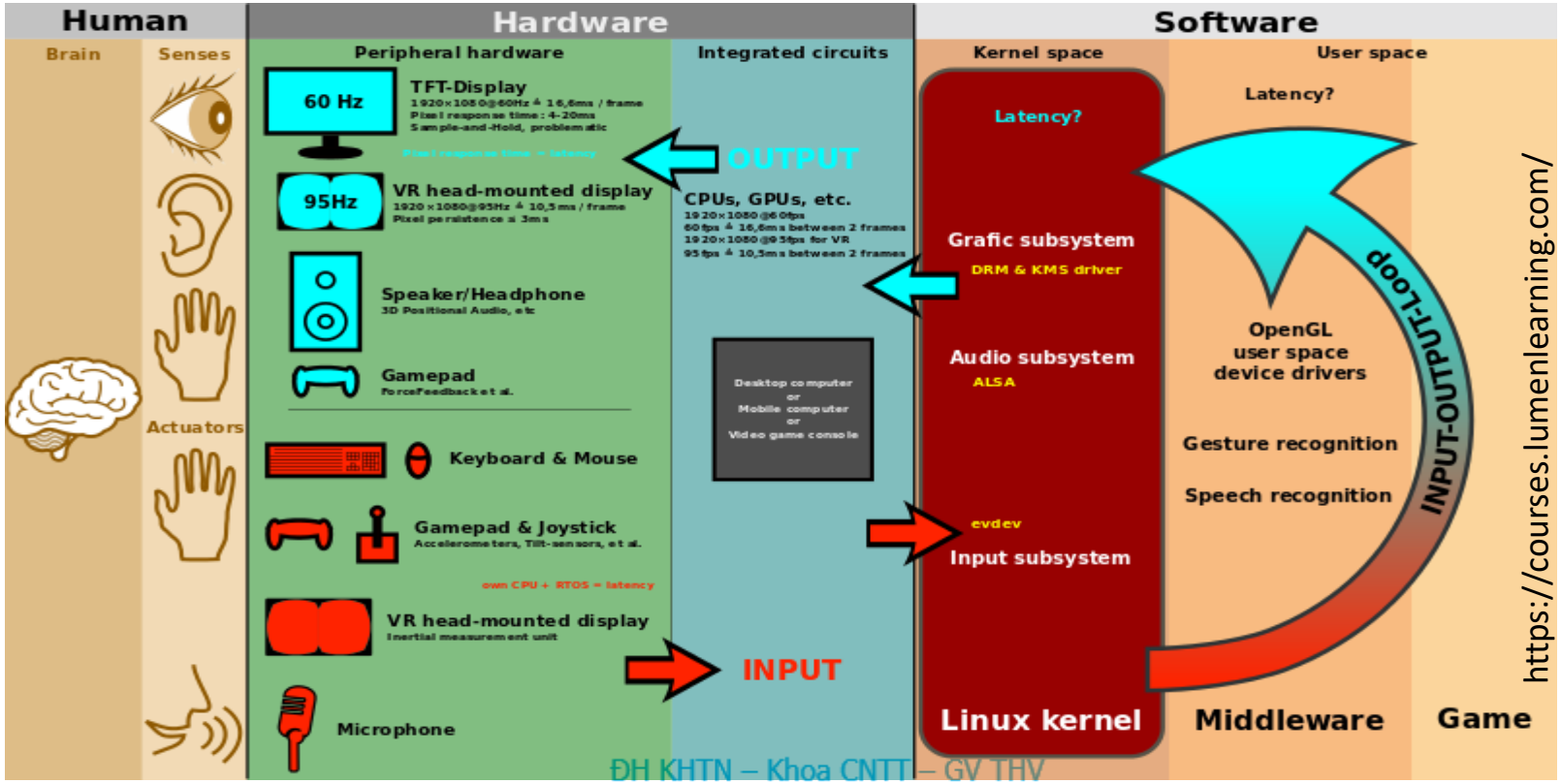
- I/O Devices
  - I/O devices (IODs) are the pieces of hardware used by a human (or other system) to communicate with a computer.
  - They are one of the three main components of a computer
  - Input or output devices that are connected to computer are called peripheral devices
- IOD Types

There are many types and functions. Some general categories:

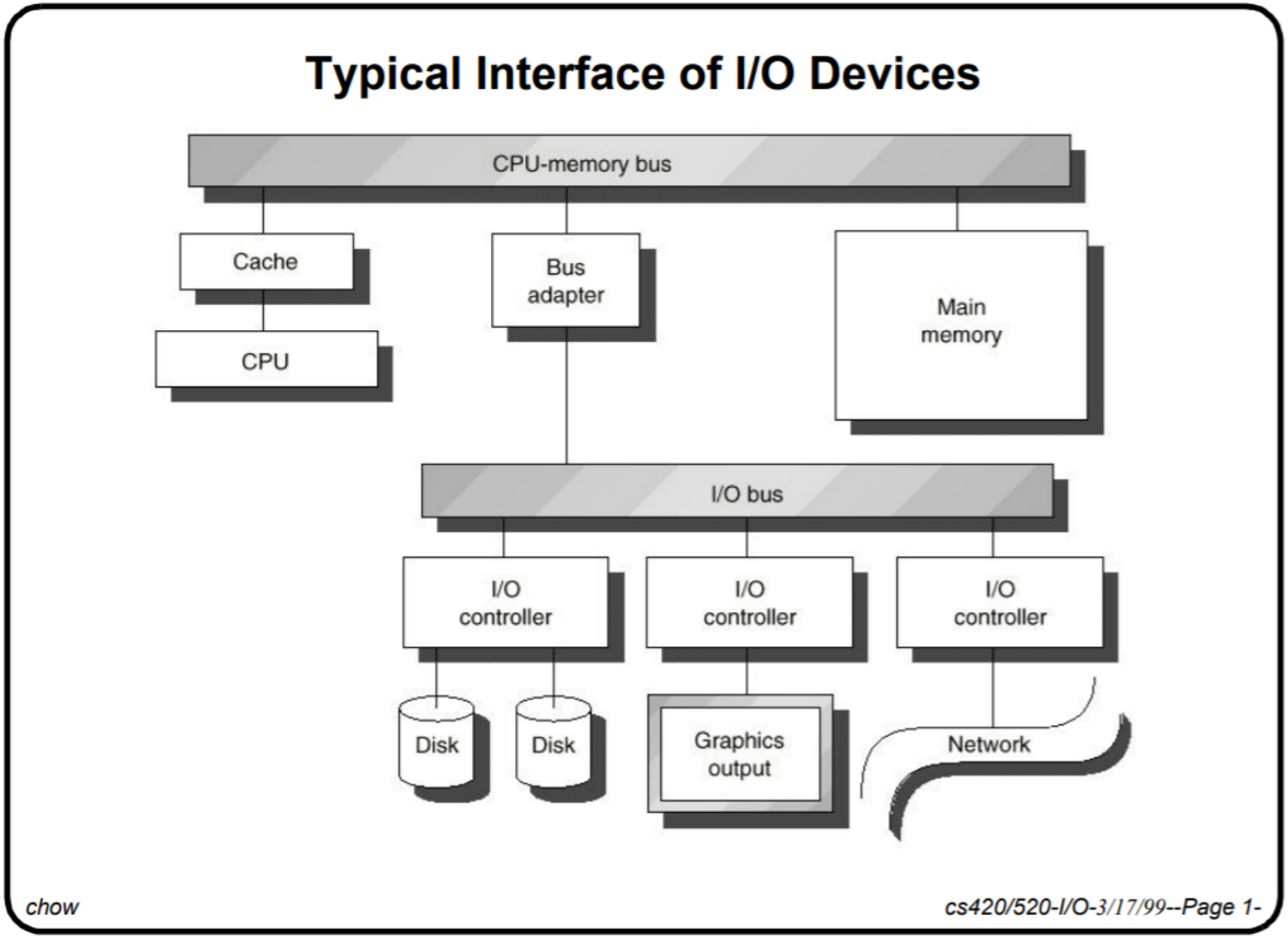
  - Input, Output and Input-Output peripherals
  - Storage *(disk,..)*, communications *(wifi,..)*, user-interface *(mouse,..)*
  - Block devices *(disk, camera,..)* and character devices *(kb, mouse,..)*
  - Logical devices and physical devices

Basic Concepts – I/O system

- I/O System:
  - Cover all stages - from building concepts, designing models, organizing systems, to functional implementation.
  - Management of IODs is a very important part of the OS.



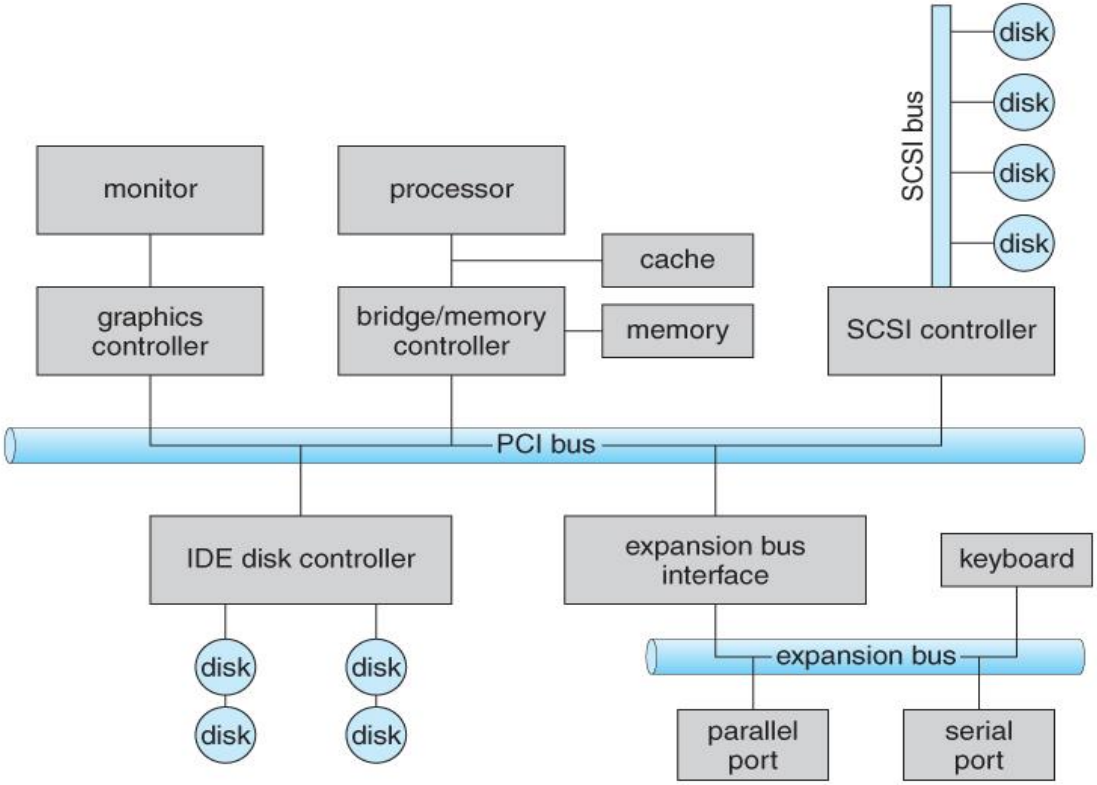
Basic Concepts – Bus



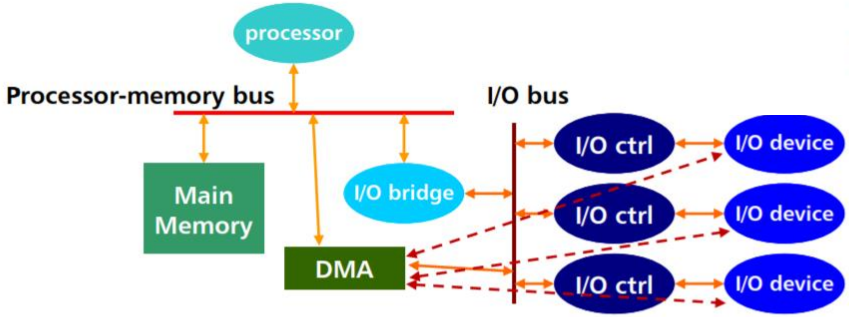
Basic Concepts – Bus

A typical PC bus structure:

["OS Concepts, 8th edition" –A. Silberschatz, Chapter 13]



I/O bus

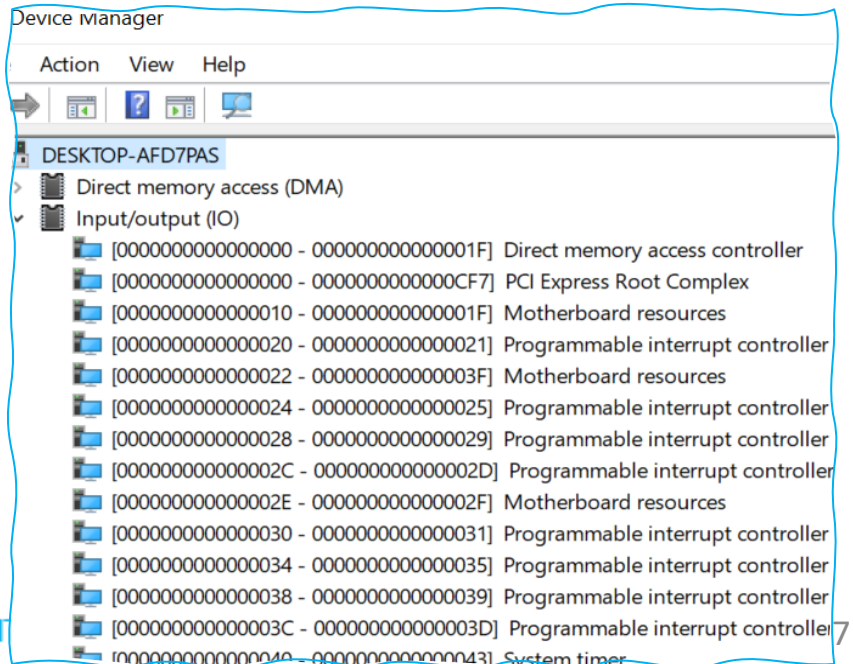


- Processor-memory bus
  - Masters: processor, I/O bridge (esp. DMA)
  - Slave: main memory
- I/O bus
  - Decouples memory and I/O bus transactions
  - Accommodates (slower) I/O devices and allows efficient data transfer using dedicated buffers and DMA

Basic Concepts – Port

- Physical Port: a socket on the machine that IOD cable is plugged into.
- (Logical) Port: an addresses used to transfer data. A port references a separate memory space on peripheral boards. *(similar to memory-mapped peripherals that use blocks of memory).*
- Peripherals often use both methods: an I/O address for passing control signals and memory for transferring data.
- IOD Port Locations on PCs:

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

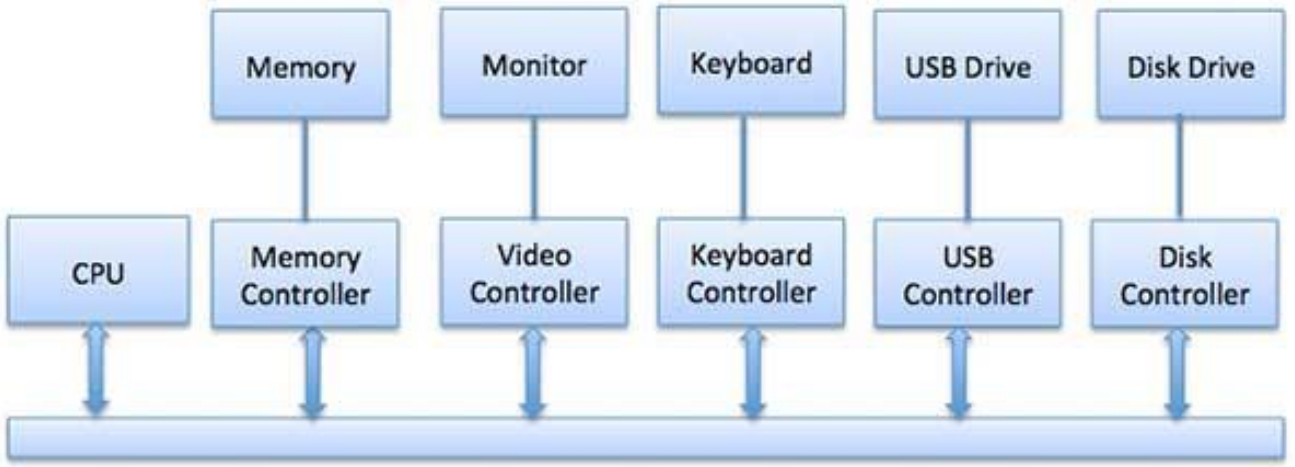


Basic Concepts – Drivers

- Device Driver is a software module that operates /controls a particular type of device that is attached to a computer.
- A driver provides a software interface to IODs, enabling OSs and other programs to access hardware functions without needing to know precise details about the hardware being used.
- Driver communicate with IODs through the I/O bus
- Drivers are hardware dependent and OS-specific. They usually provide the interrupt handling required for any necessary asynchronous time-dependent hardware interface.

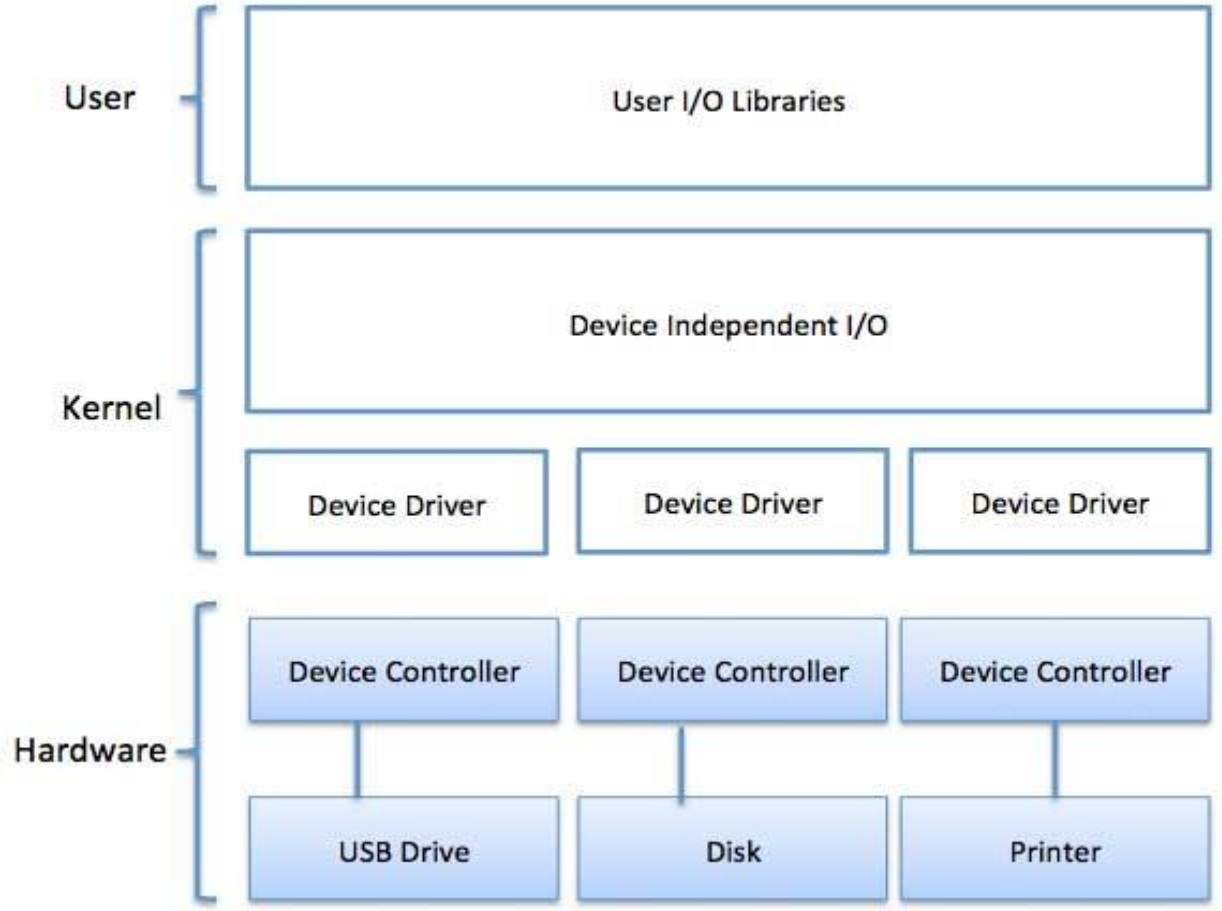
Basic Concepts – Controller

- Device Controller works like an interface between IOD & Device driver
- Controller's tasks: convert bit stream to block of bytes, perform error correction as necessary, make available to main memory.



Basic Concepts – Device Independent I/O

- A key concept in IO system design is known as device independence
- It is the process of making a software module able to function on a wide variety of devices regardless of the local hardware.

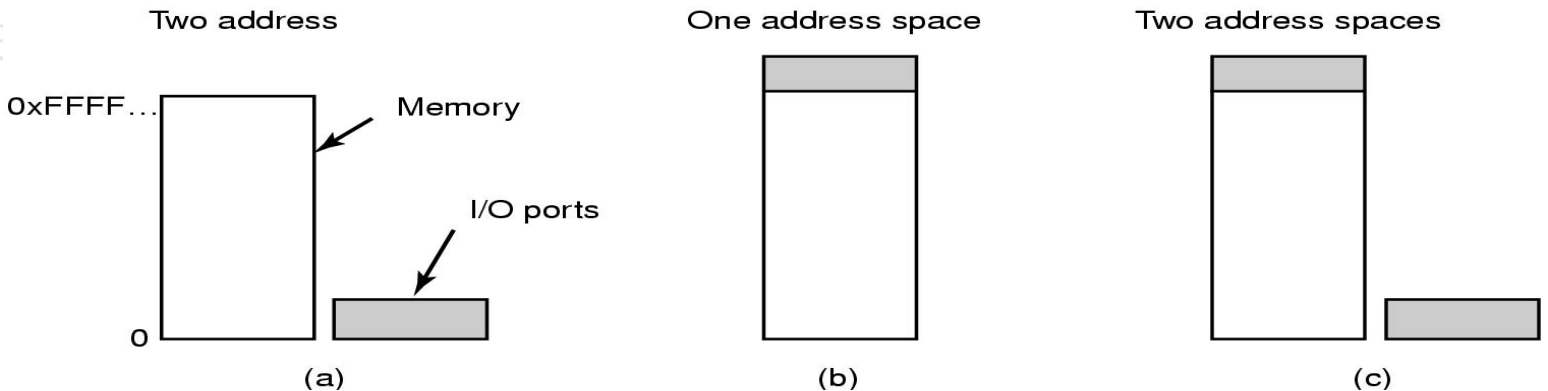


Communication to IODs

- Synchronous vs Asynchronous I/O
  - **Synchronous I/O** − In this scheme CPU execution waits while I/O proceeds
  - **Asynchronous I/O** − I/O proceeds concurrently with CPU execution
- The CPU must have a way to pass information to and from an IOD. There are some approaches available to communicate with the CPU and IOD.
  - Registers associated with Port
  - Memory-mapped I/O
  - Direct memory access (**DMA**)
  - Special Instruction I/O

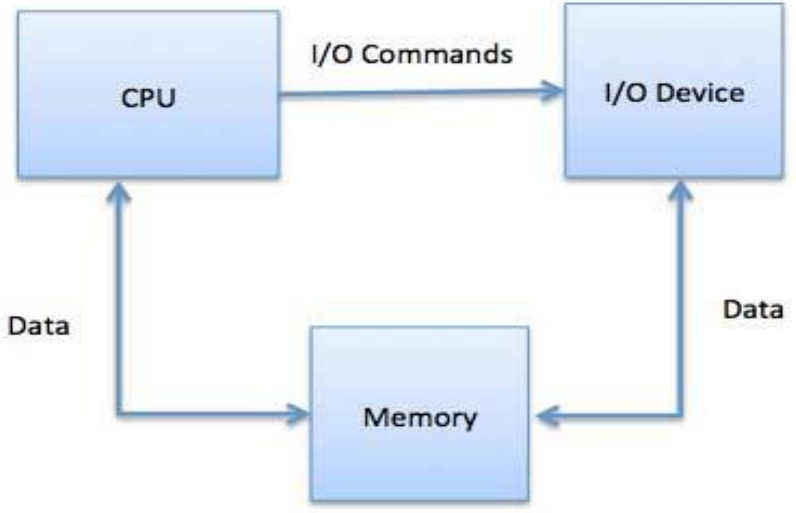
Communication to IODs by Port

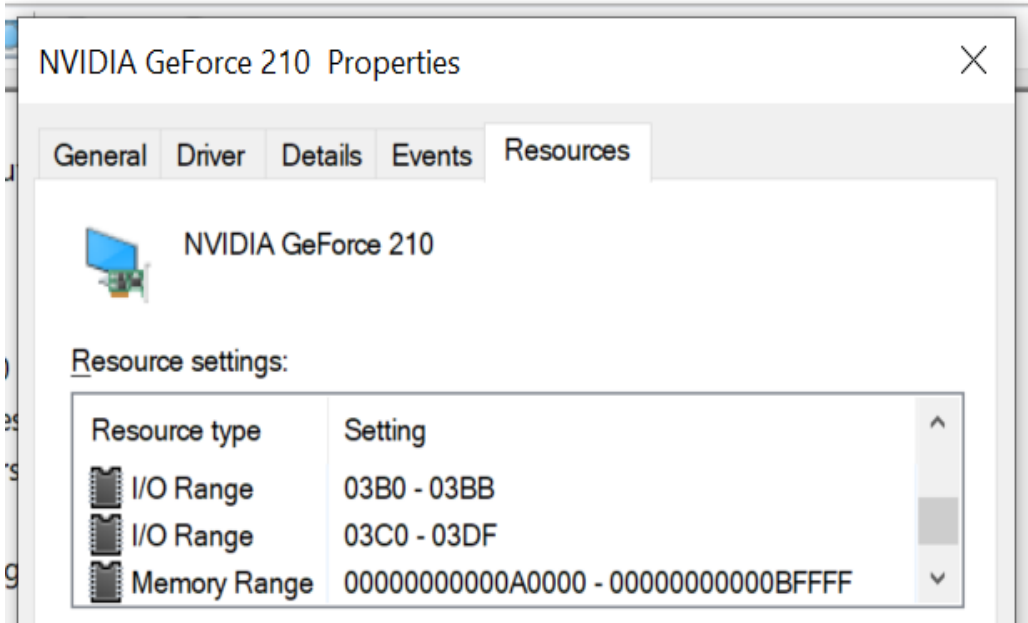
- One simple way of communicating with devices is through ***Registers*** associated with each Port:
  - ***data-in register*** is read by the host to get input from IOD.
  - ***data-out register*** is written by the host to send output.
  - ***status register*** has bits read by the host to ascertain the status of IOD, such as idle, ready for input, busy, error, transaction complete, etc.
  - ***control register*** has bits written by the host to issue commands or to change settings of the device such as parity checking, word length, or full- versus half-duplex operation
- The



Communication to IODs by Memory-mapped IO

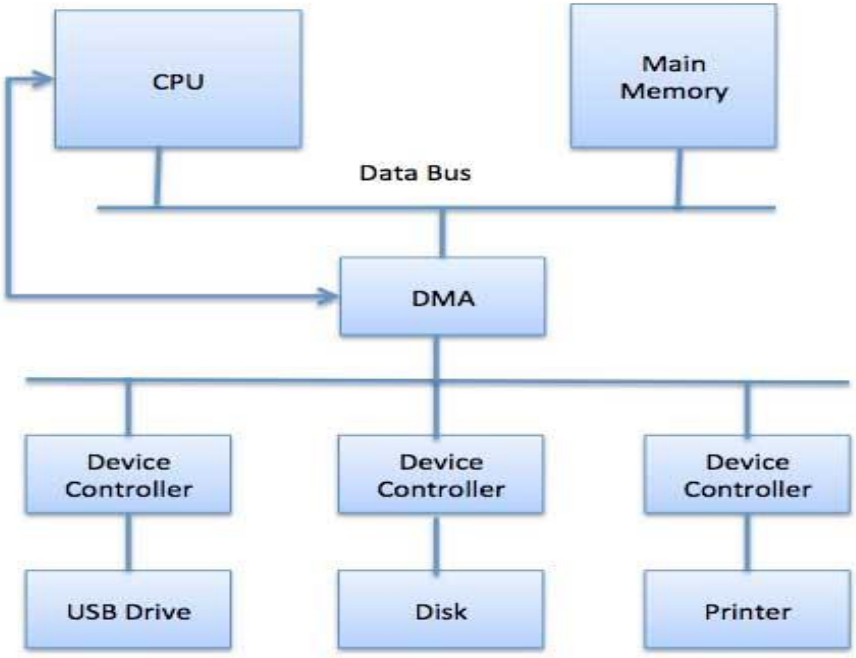
- Most IODs can be considered as memories, with an “address” for reading /writing. A same address space is shared by memory & IODs
- The device is connected directly to main memory locations so that IOD can transfer block of data to/from memory without going through CPU.
- OS allocates buffer in memory and informs IOD to use that buffer to send data. IOD operates asynchronously with CPU, interrupts CPU when finished.

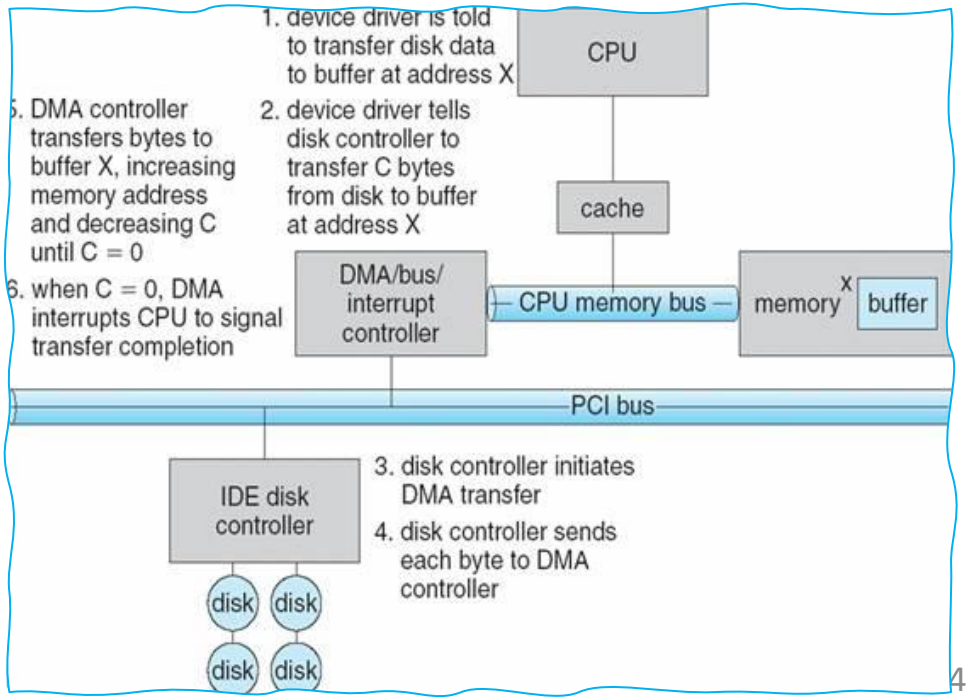




Communication to IODs by DMA

- Slow devices like keyboard, mouse will generate an interrupt to the CPU after each byte is transferred.
- If a fast device *(such as a disk)* generated an interrupt for each byte, the OS will be overloaded and it must use DMA
- DMA module controls exchange of data between memory and IOD. CPU is only involved at the beginning & end of the transfer.





Communication to IODs by Special Instruction

- This uses CPU instructions that are specifically made for controlling IODs.
- These instructions typically allow data to be sent to an IOD or read from an IOD.
  - In some case, data are stored in CPU register - no need to use memory.
- Basic IODs are usually programmed to control /read / write data via interrupt functions in ROM BIOS.

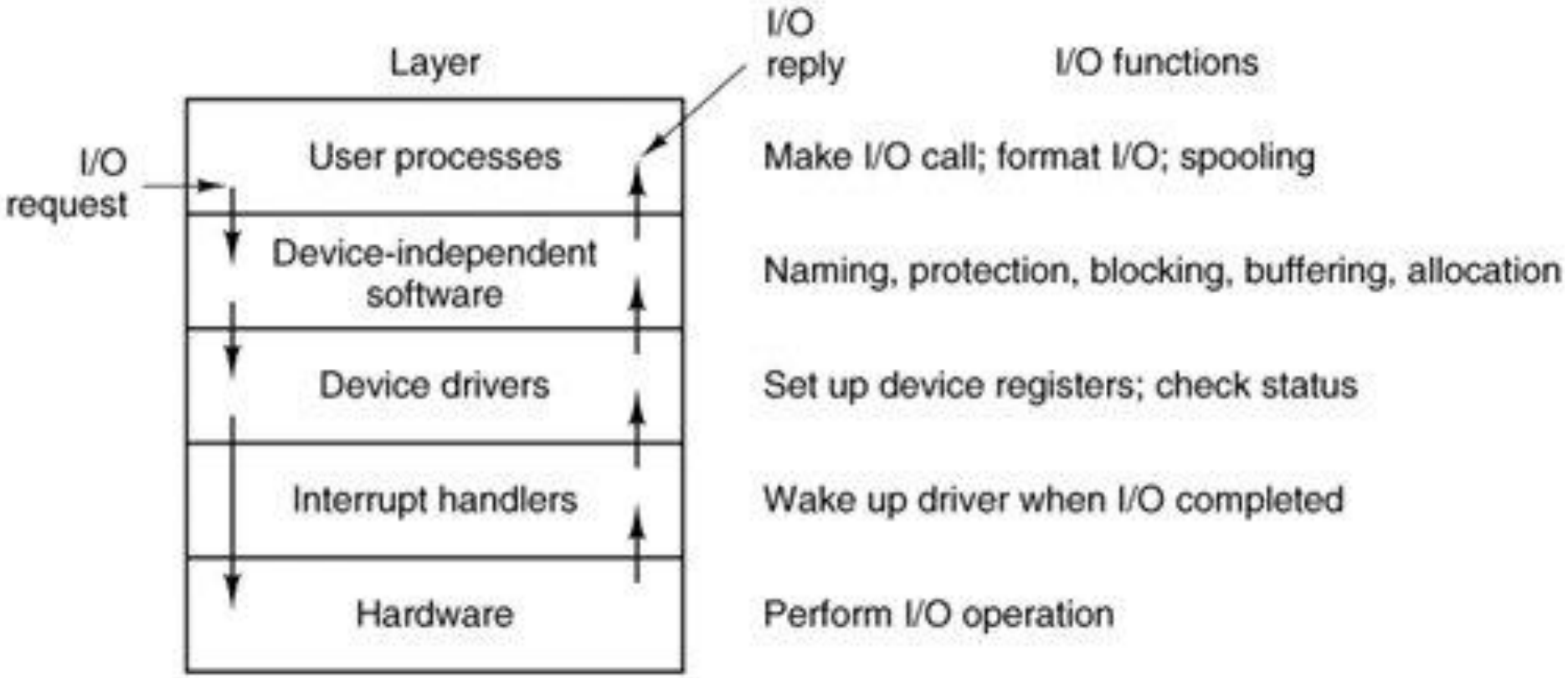
Polling vs Interrupts I/O

- Polling
  - The process of periodically checking status of the IOD to see if it is time for the next I/O operation, is called polling.
  - The IOD simply puts the information in a Status register, and the processor must come and get the information.
  - This is an inefficient method and much of the processors time is wasted on unnecessary polls.
- Interrupts I/O
  - A IOD controller puts an interrupt signal on the bus when it needs CPU's attention.
  - When CPU receives an interrupt, it saves current state & invokes the appropriate interrupt handler using the interrupt vector.
  - When the interrupting device has been dealt with, the CPU continues with its original task as if it had never been interrupted.

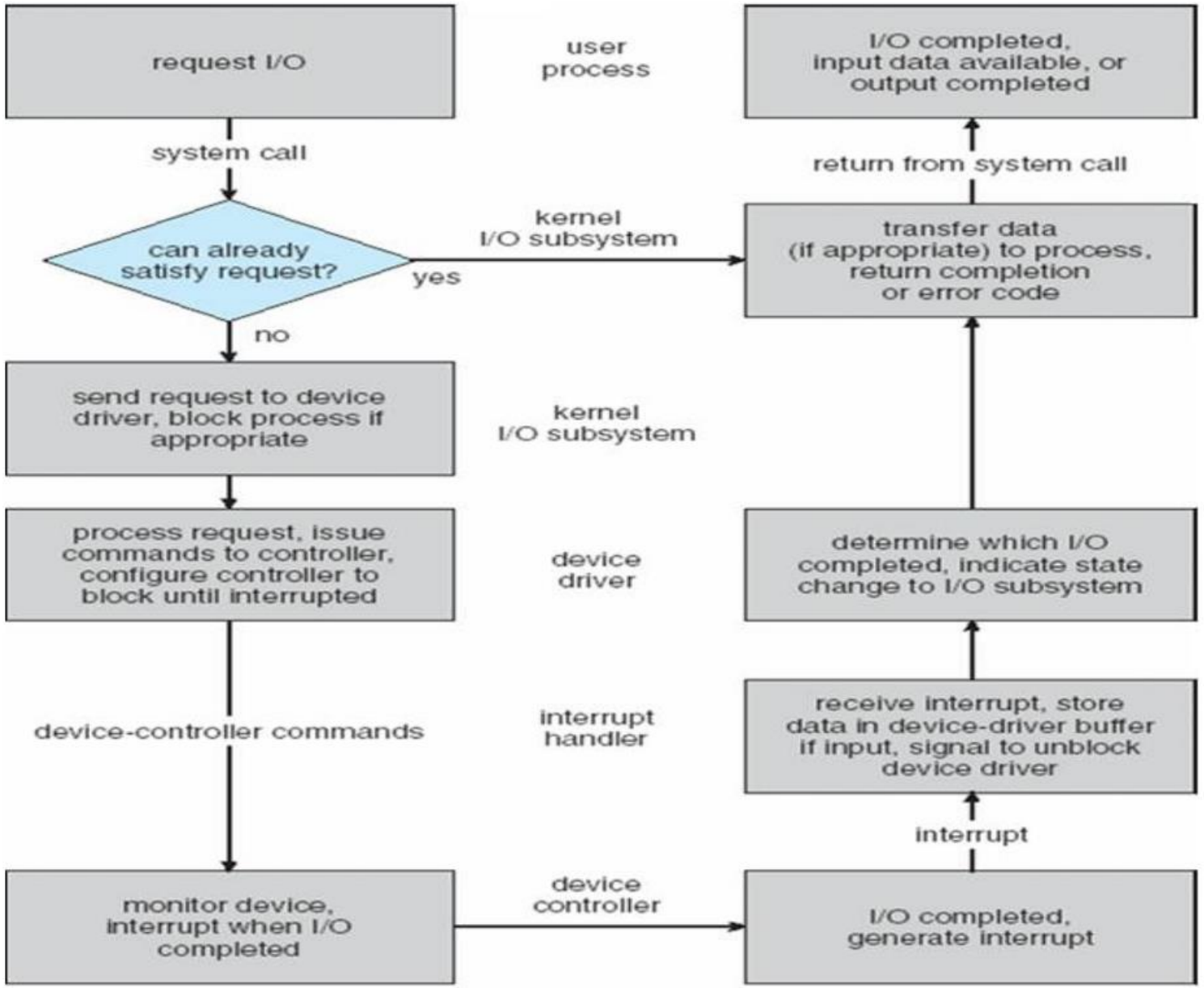
BIOS & UEFI

- BIOS *(Basic Input Output System)* is a firmware used to perform hardware initialization during the booting process (power-on startup), and to provide runtime services for OSs and programs.
- The BIOS firmware comes pre-installed on PC's system board, and it is the first software to run when powered on. It initializes and tests hardware components, loads a boot loader from a mass memory device which then initializes an OS.
- Originally, BIOS firmware was stored in a ROM chip on the Mobo. After that, the BIOS contents are stored on flash memory *(ok/!ok?)*
- UEFI *(Unified Extensible Firmware Interface)* is a successor to the legacy BIOS. It provides several technical advantages over BIOS: Ability to use large disks (>2TB) with a GPT, CPU-independent architecture & drivers, Flexible pre-OS environment (including network capability), Modular design, Backward and forward compatibility.

I/O Software Layers



I/O Request Process



Popular Signal and IODs

- Popular signal on computer?
- Popular physical ports on PC?
- Popular input devices on PC?
- Popular output devices on PC?

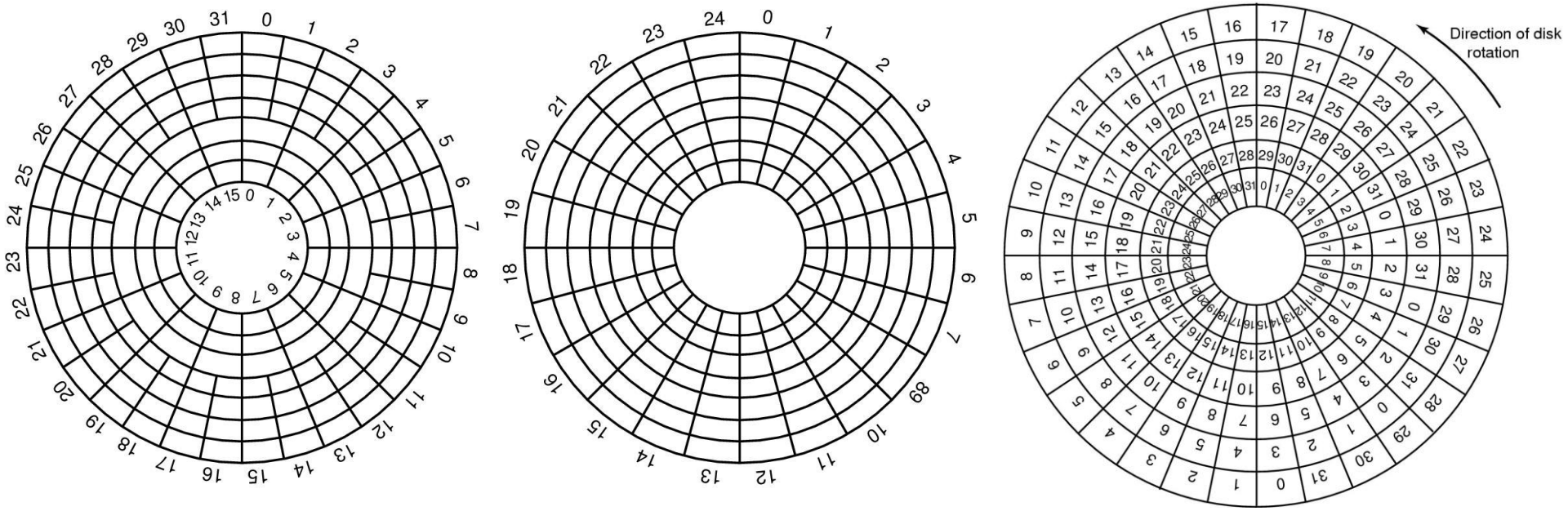
Typical Ports Rate

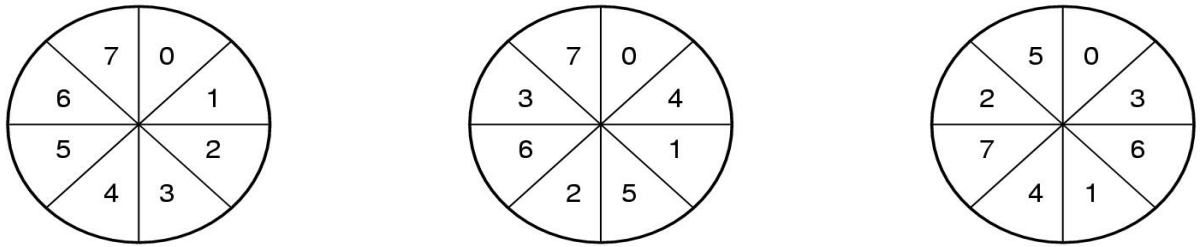
PORT	Rate	
PS/2 port	12.0 kbit/s	1.5 kB/s
Serial RS-232 max	230.4 kbit/s	28.8 kB/s
Serial EIA-485 max.	35 Mbit/s	4.375 MB/s
Parallel (Centronics/IEEE 1284)	1 Mbit/s	125 kB/s
USB 1.0 full speed	12 Mbit/s	1.5 MB/s
USB 2.0 high speed	480 Mbit/s	60 MB/s
USB 3.0 SuperSpeed (aka usb3.1 Gen 1)	5 Gbit/s	625 MB/s
USB 3.1 SuperSpeed (aka usb3.1 Gen 2)	10 Gbit/s	1.25 GB/s
USB 3.2 SuperSpeed (USB 3.2 Gen 2×2)	20 Gbit/s	2.5 GB/s
USB 4	40 Gbit/s	5 GB/s
Thunderbolt	2 × 10 Gbit/s	2 × 1.25 GB/s
Thunderbolt 2	20 Gbit/s	2.5 GB/s
Thunderbolt 3 two links	40 Gbit/s	5 GB/s

FireWire (IEEE 1394) 100	98.304 Mbit/s	12.288 MB/s
FireWire (IEEE 1394) 200	196.608 Mbit/s	24.576 MB/s
FireWire (IEEE 1394) 400	393.216 Mbit/s	49.152 MB/s
FireWire (IEEE 1394b) 800	786.432 Mbit/s	98.304 MB/s
FireWire (IEEE 1394b) 1600	1.573 Gbit/s	196.6 MB/s
FireWire (IEEE 1394b) 3200	3.1457 Gbit/s	393.216 MB/s
eSATA (SATA 300)	3 Gbit/s	300 MB/s
eSATA (SATA 600)	6 Gbit/s	600 MB/s
External PCI Express 2.0 ×1	4 Gbit/s	500 MB/s
External PCI Express 2.0 ×2	8 Gbit/s	1 GB/s
External PCI Express 2.0 ×4	16 Gbit/s	2 GB/s
External PCI Express 2.0 ×8	32 Gbit/s	4 GB/s
External PCI Express 2.0 ×16	64 Gbit/s	8 GB/s

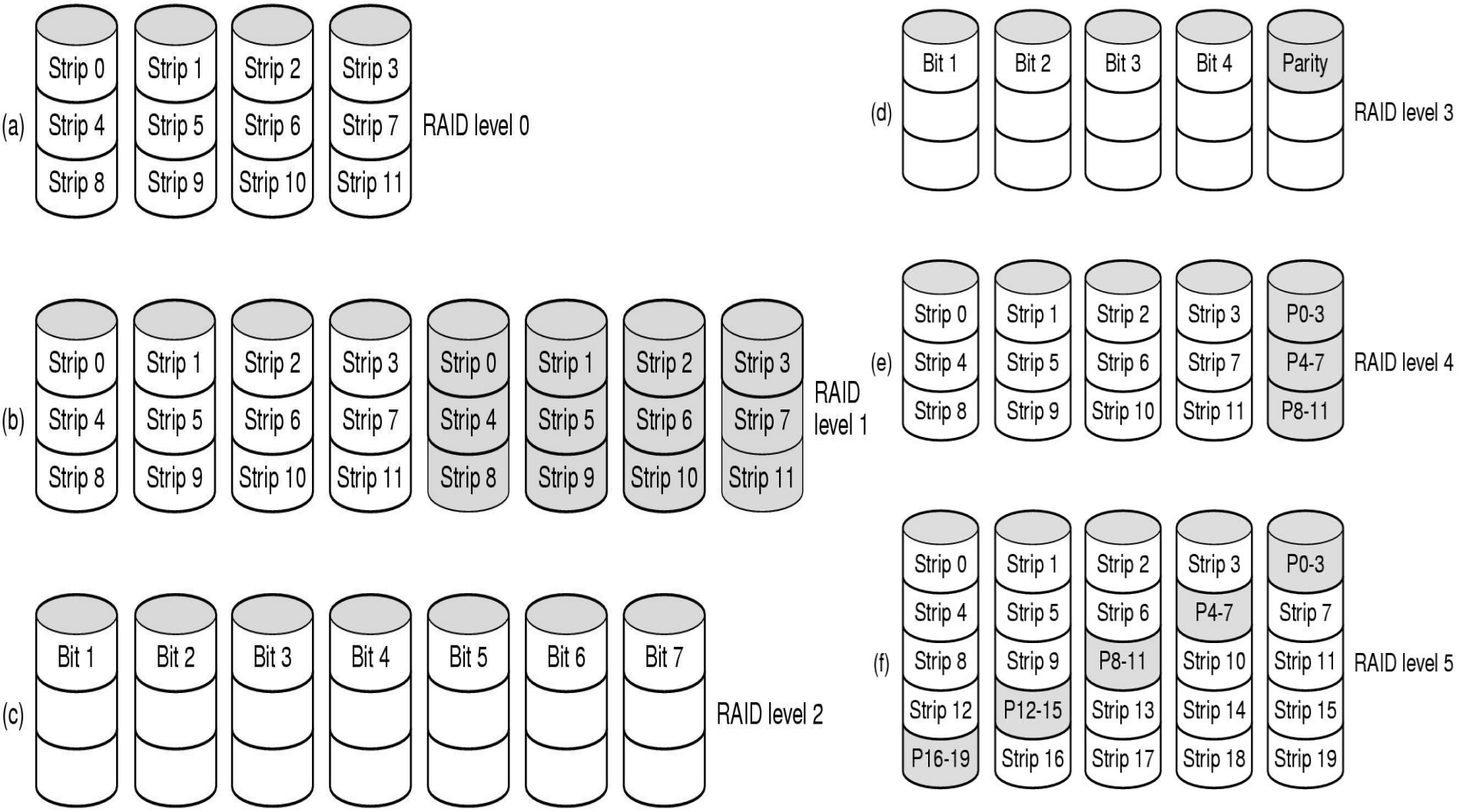
Disk Hardware

- Physical geometry of a disk with two zones
- A possible virtual geometry for this disk
- (a) No interleaving (b) Single interleaving (c) Double interleaving





Disk RAID


(a)
Strip 0
Strip 1
Strip 2
Strip 3
Strip 4
Strip 5
Strip 6
Strip 7
Strip 8
Strip 9
Strip 10
Strip 11
RAID level 0
(b)
Strip 0
Strip 1
Strip 2
Strip 3
Strip 4
Strip 5
Strip 6
Strip 7
Strip 8
Strip 9
Strip 10
Strip 11
Strip 0
Strip 1
Strip 2
Strip 3
Strip 4
Strip 5
Strip 6
Strip 7
Strip 8
Strip 9
Strip 10
Strip 11
RAID level 1
(c)
Bit 1
Bit 2
Bit 3
Bit 4
Bit 5
Bit 6
Bit 7
RAID level 2
(d)
Bit 1
Bit 2
Bit 3
Bit 4
Parity
RAID level 3
(e)
Strip 0
Strip 1
Strip 2
Strip 3
P0-3
Strip 4
Strip 5
Strip 6
Strip 7
P4-7
Strip 8
Strip 9
Strip 10
Strip 11
P8-11
RAID level 4
(f)
Strip 0
Strip 1
Strip 2
Strip 3
P0-3
Strip 4
Strip 5
Strip 6
P4-7
Strip 7
Strip 8
Strip 9
P8-11
Strip 10
Strip 11
Strip 12
P12-15
Strip 13
Strip 14
Strip 15
P16-19
Strip 16
Strip 17
Strip 18
Strip 19
RAID level 5

Disk Arm Scheduling

- Time required to read or write a disk block determined by factors: Seek time, Rotational delay, Actual transfer time
- Error checking is done by controllers
- Shortest Seek First (SSF) and elevator disk scheduling: