CS 3502
Operating Systems

Input/Output

Kun Suo

Computer Science, Kennesaw State University

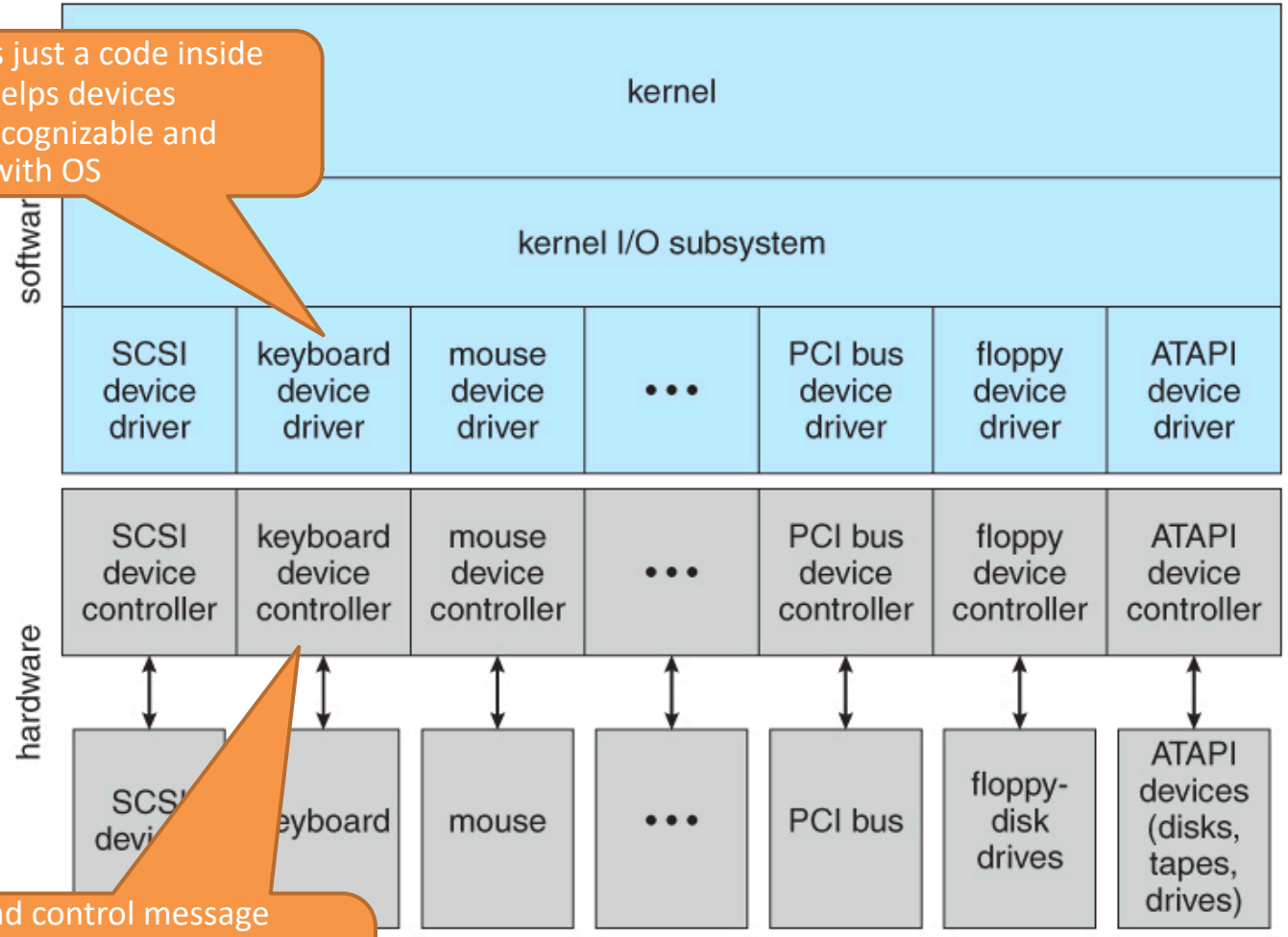
https://kevinsuo.github.io/

Outline

- I/O System
- Characteristics and Category of I/O Devices
- I/O Device Component
- How CPU works with I/O devices
- I/O Performance

I/O System Structure

Device driver is just a code inside the OS which helps devices compatible , recognizable and communicate with OS



Receive data and control message from the OS, and send back the data and status message to the OS
Convert digital signal into analog signal for the mechanical parts or vice verse

I/O System Structure

- Device Drivers : low level **software** written by hardware manufacturers which are loaded as kernel modules and provide the kernel with the knowledge on how to control the devices. It acts bridge between **kernel** and **device**.

- Device Controllers: A device controller (**hardware**) understands "software" input. It translates software input into signal a hardware device understands. It acts bridge between **OS** and **hardware mechanical parts**.

lsmod: list all drivers in the OS

```
ksuo@ksuo-VirtualBox ~> lsmod
Module                  Size  Used by
snd_intel8x0           45056  2
snd_ac97_codec        135168  1 snd_intel8x0
ac97_bus               16384  1 snd_ac97_codec
snd_pcm               102400  2 snd_intel8x0,snd_ac97_codec
crct10dif_pclmul       16384  1
crc32_pclmul           16384  0
snd_seq_midi           20480  0
ghash_clmulni_intel    16384  0
snd_seq_midi_event     16384  1 snd_seq_midi
snd_rawmidi            36864  1 snd_seq_midi
snd_seq                69632  2 snd_seq_midi,snd_seq_midi_event
aesni_intel           372736  0
aes_x86_64             20480  1 aesni_intel
crypto_simd            16384  1 aesni_intel
vmwgfx                290816  2
cryptd                 24576  3 crypto_simd,ghash_clmulni_intel,aesni_intel
glue_helper            16384  1 aesni_intel
snd_seq_device         16384  3 snd_seq,snd_seq_midi,snd_rawmidi
intel_rapl_perf        16384  0
snd_timer              36864  2 snd_seq,snd_pcm
joydev                 28672  0
ttm                   102400  1 vmwgfx
drm_kms_helper        180224  1 vmwgfx
input_leds             16384  0
serio_raw              20480  0
drm                   479232  5 vmwgfx,drm_kms_helper,ttm
snd                    86016  11 snd_seq,snd_seq_device,snd_intel8x0,snd_timer,snd_ac97_codec,s
fb_sys_fops            16384  1 drm_kms_helper
syscopyarea            16384  1 drm_kms_helper
sysfillrect            16384  1 drm_kms_helper
mac_hid                16384  0
sysimgblt              16384  1 drm_kms_helper
soundcore              16384  1 snd
vboxguest             335872  0
sch_fq_codel           20480  2
parport_pc             36864  0
ppdev                  24576  0
hid_generic            16384  0
usbhid                 53248  0
hid                   126976  2 usbhid,hid_generic
ahci                   40960  1
psmouse               151552  0
libahci                32768  1 ahci
i2c_piix4              28672  0
e1000                 139264  0
pata_acpi              16384  0
video                  49152  0
```

Keyboard driver

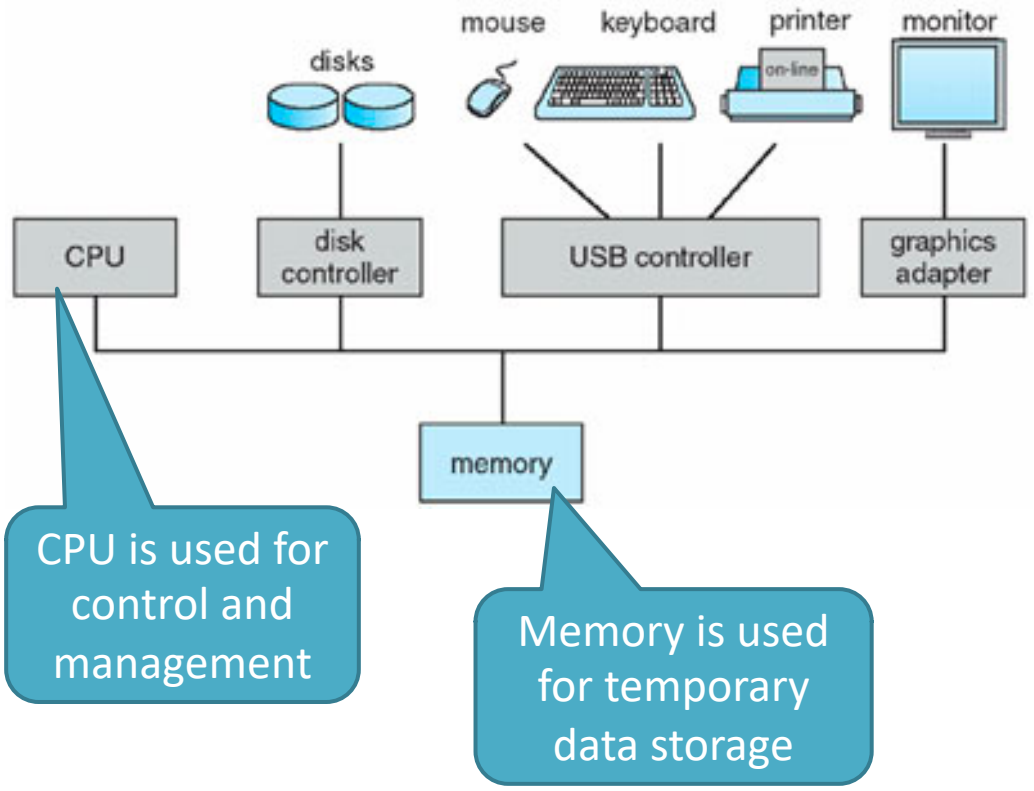
Mouse driver

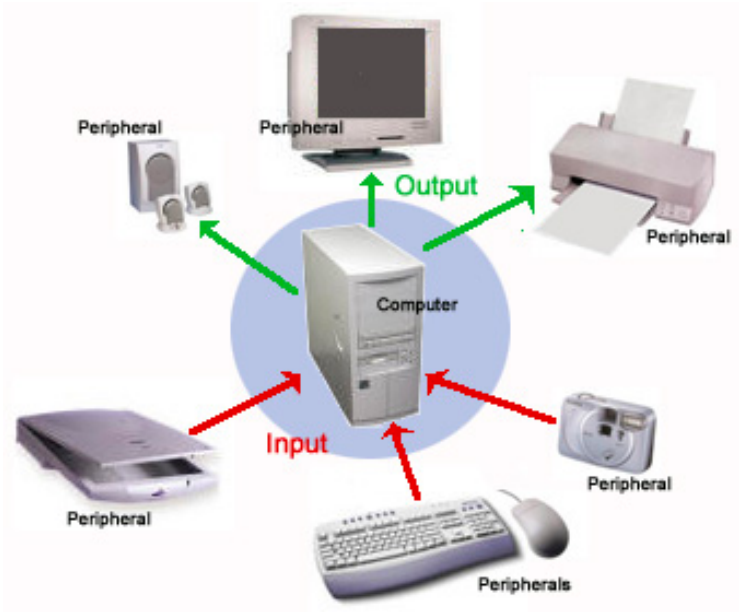
OS kernel:
Scheduling, interrupt, memory, refresh display, …

Keyboard driver:
key code → program logic

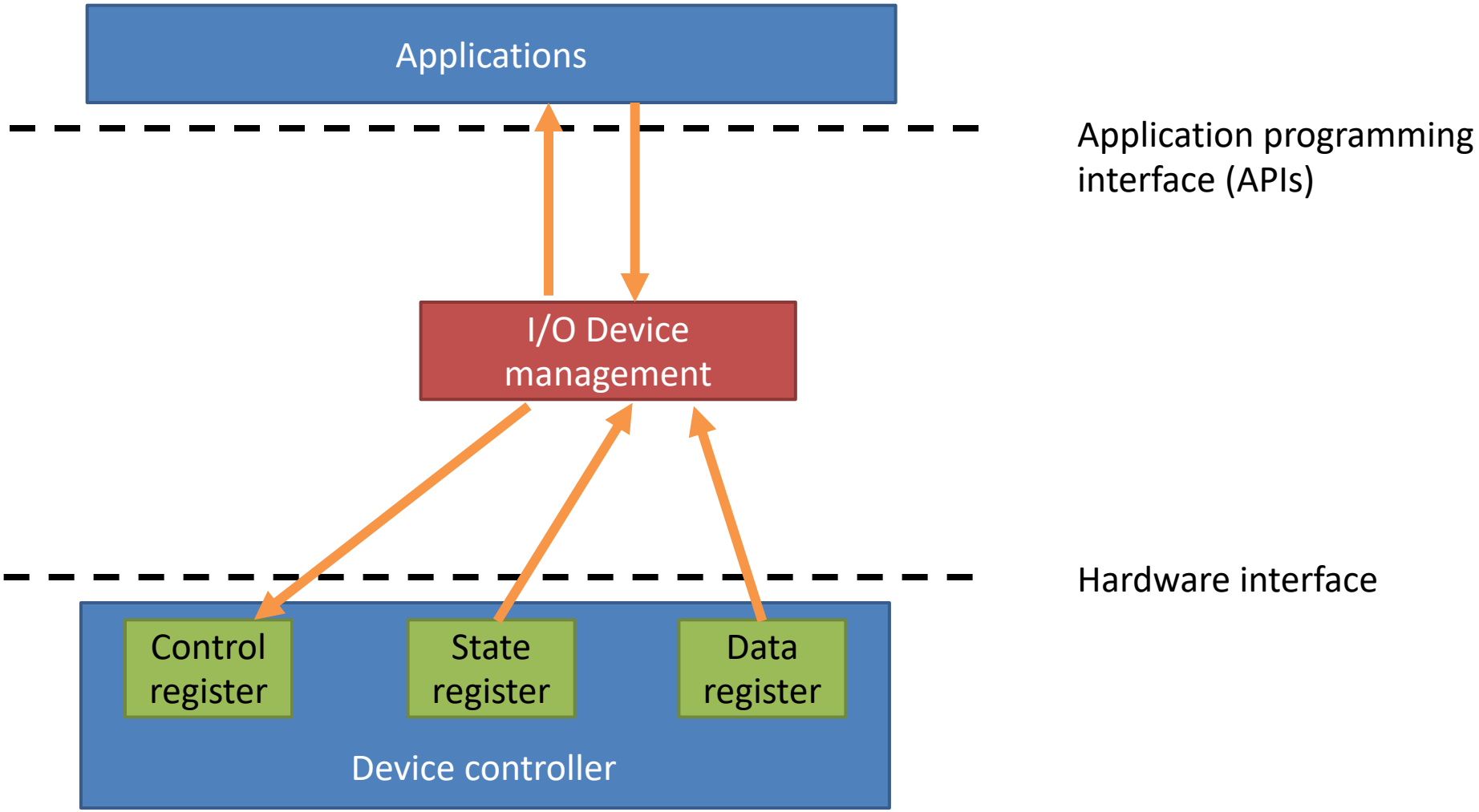
Keyboard controller:
Signal → key code

I/O System Structure

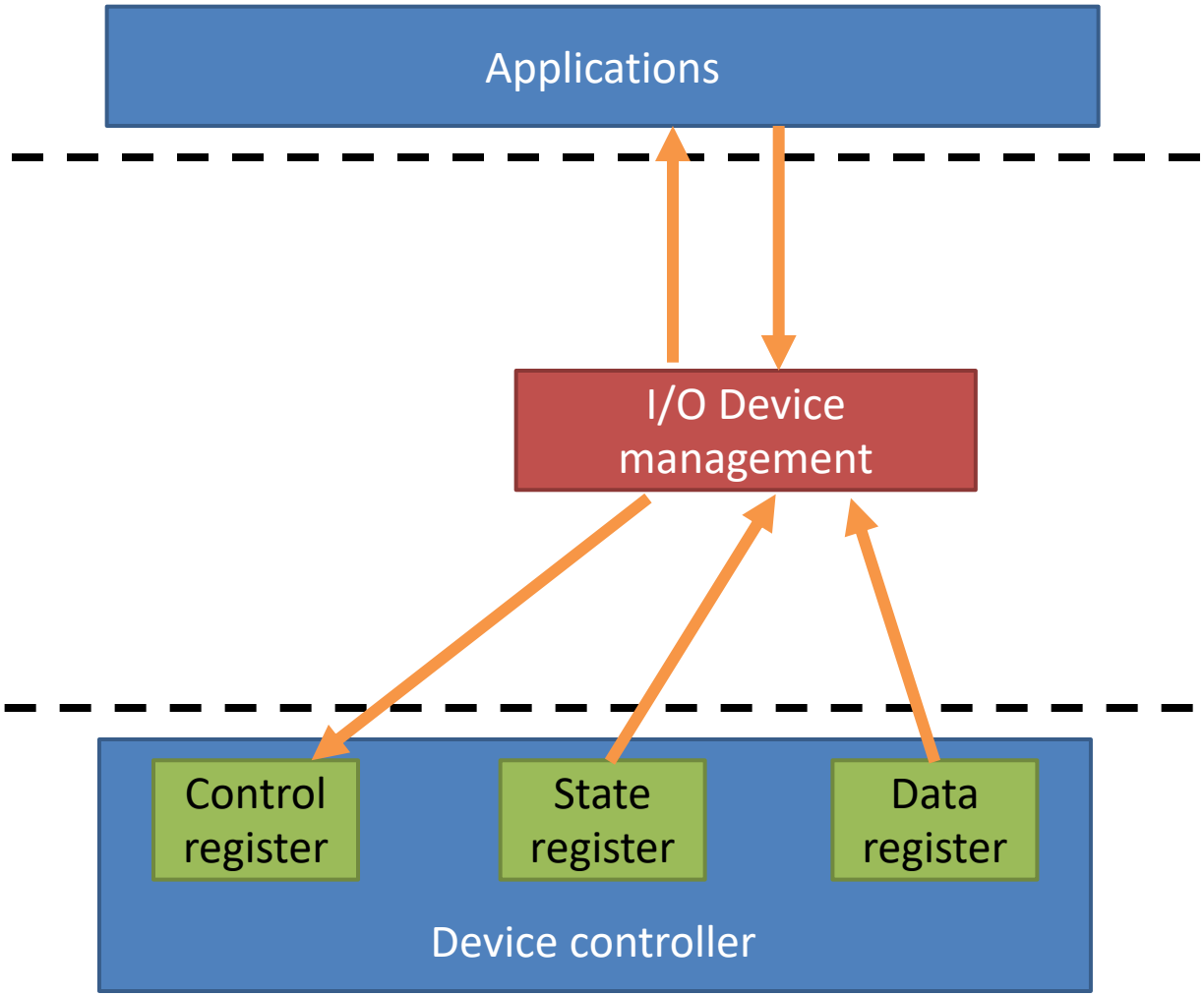




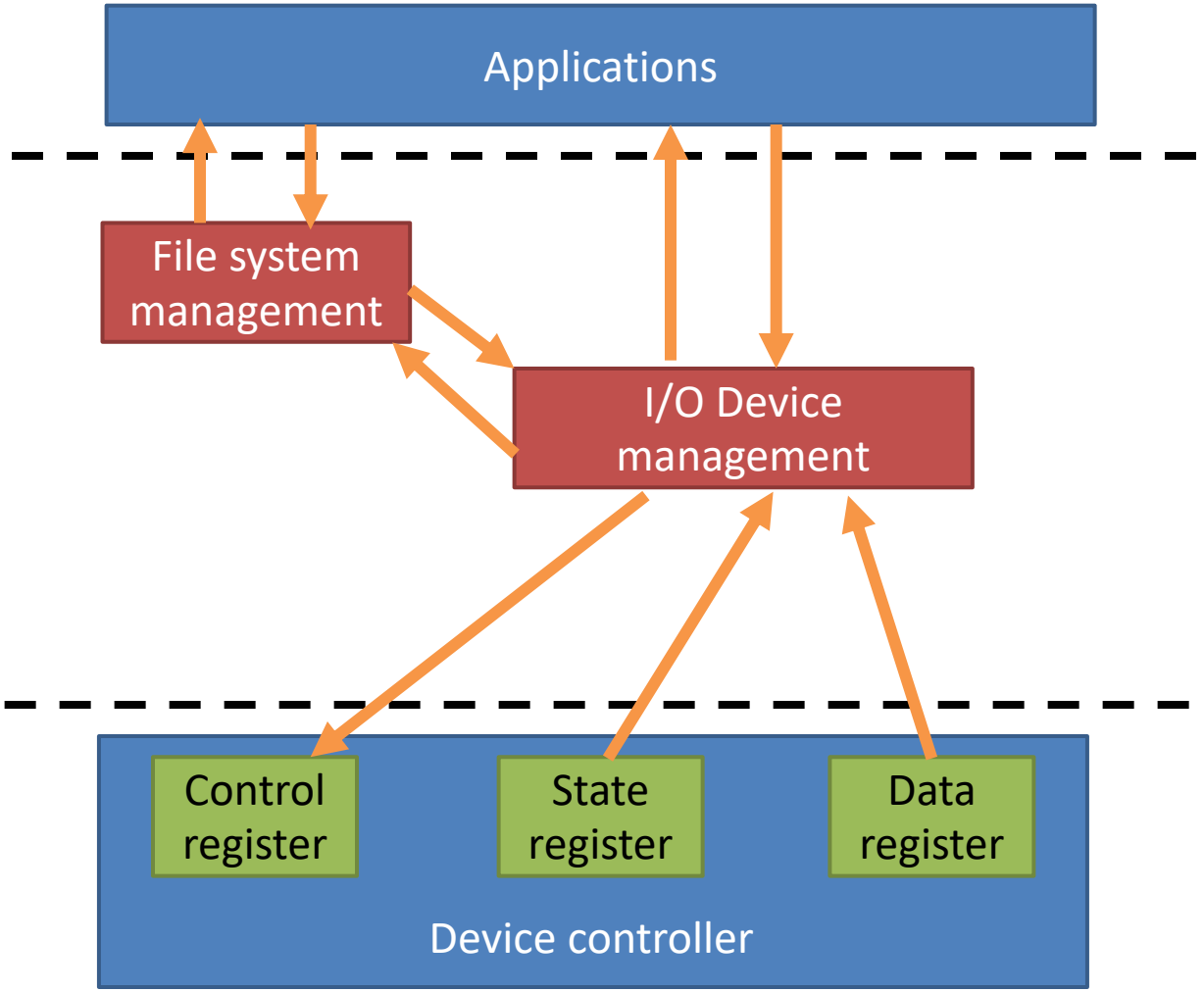
I/O Management



I/O Management Example



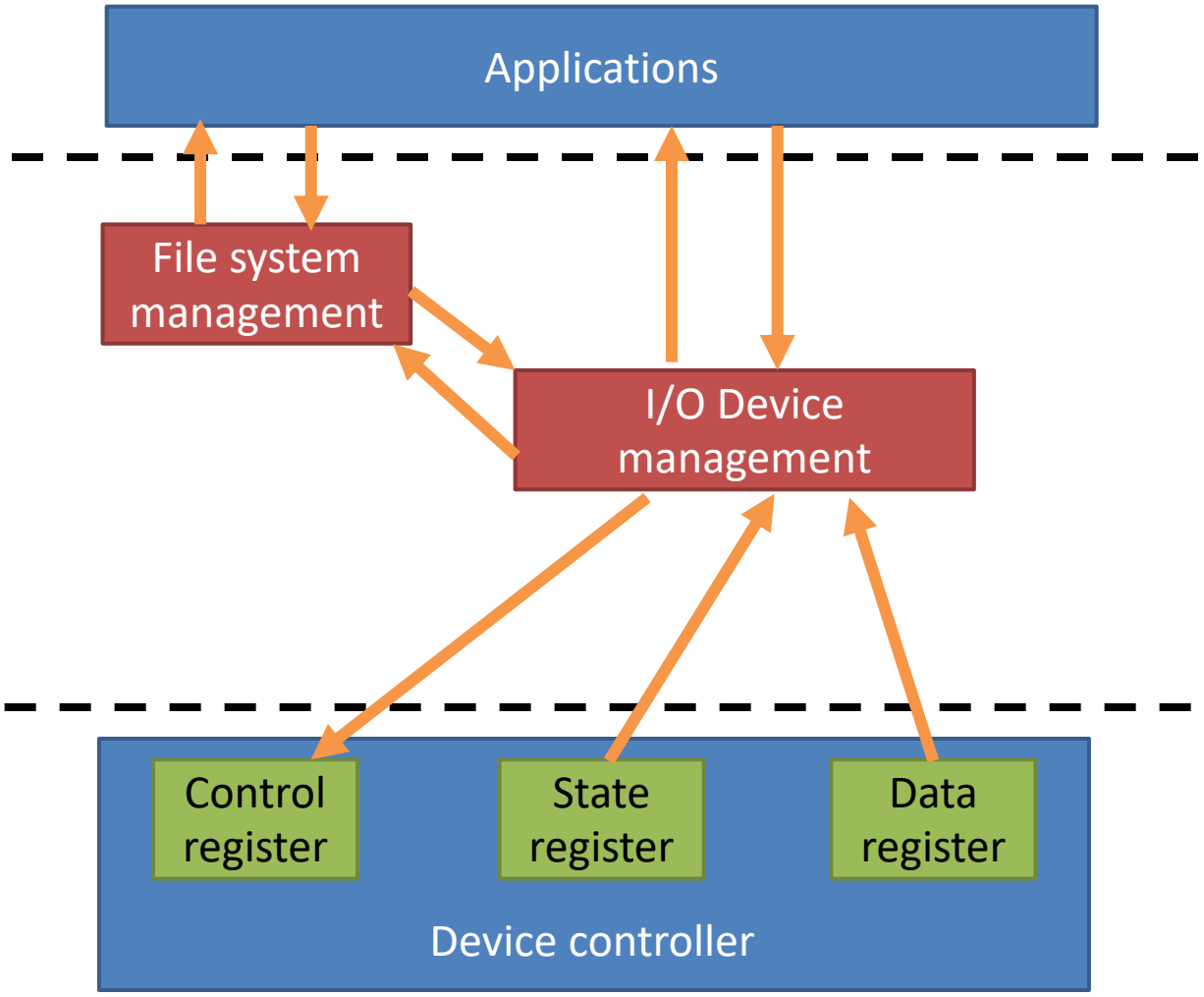
I/O Management



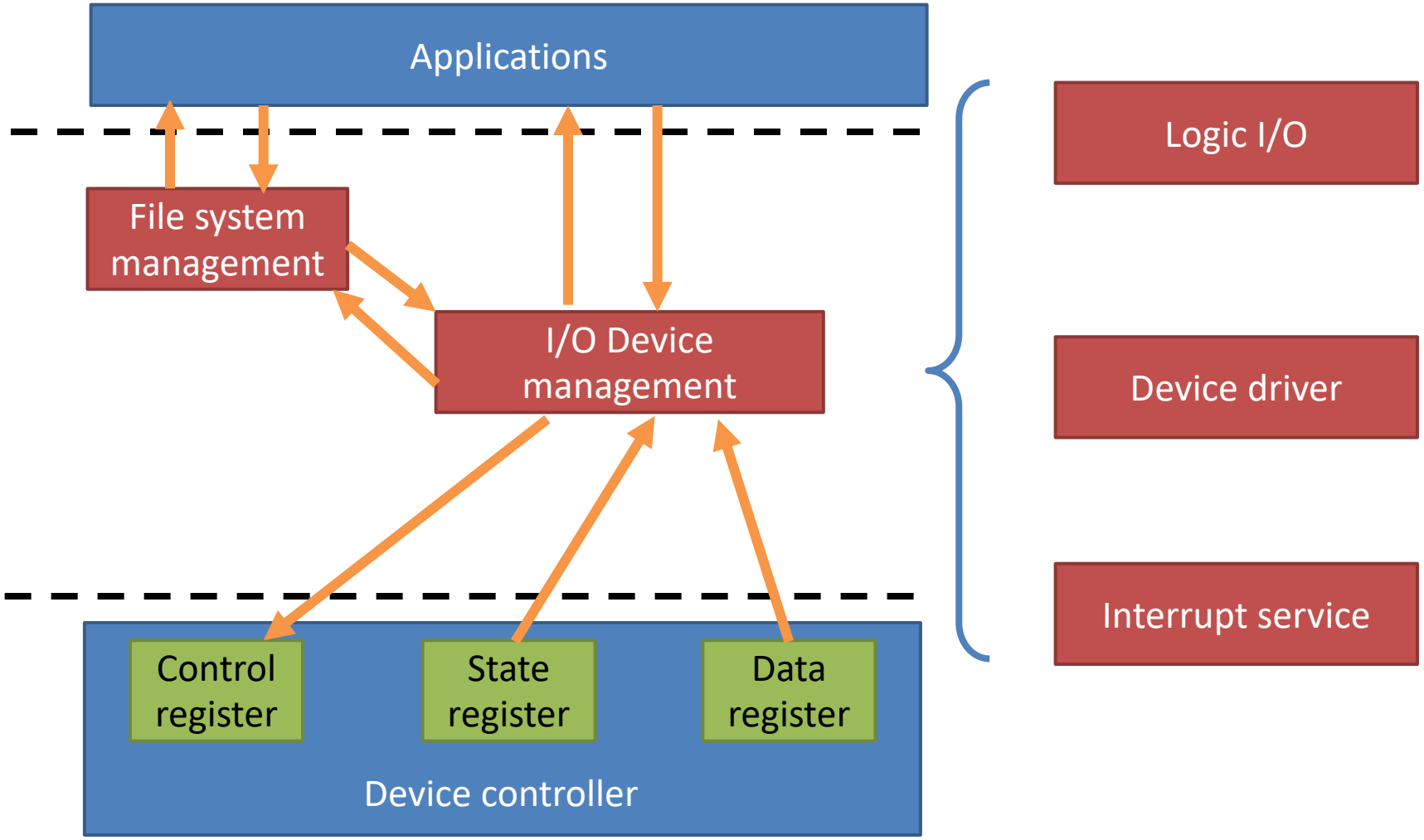
Application programming interface (APIs)

Hardware interface

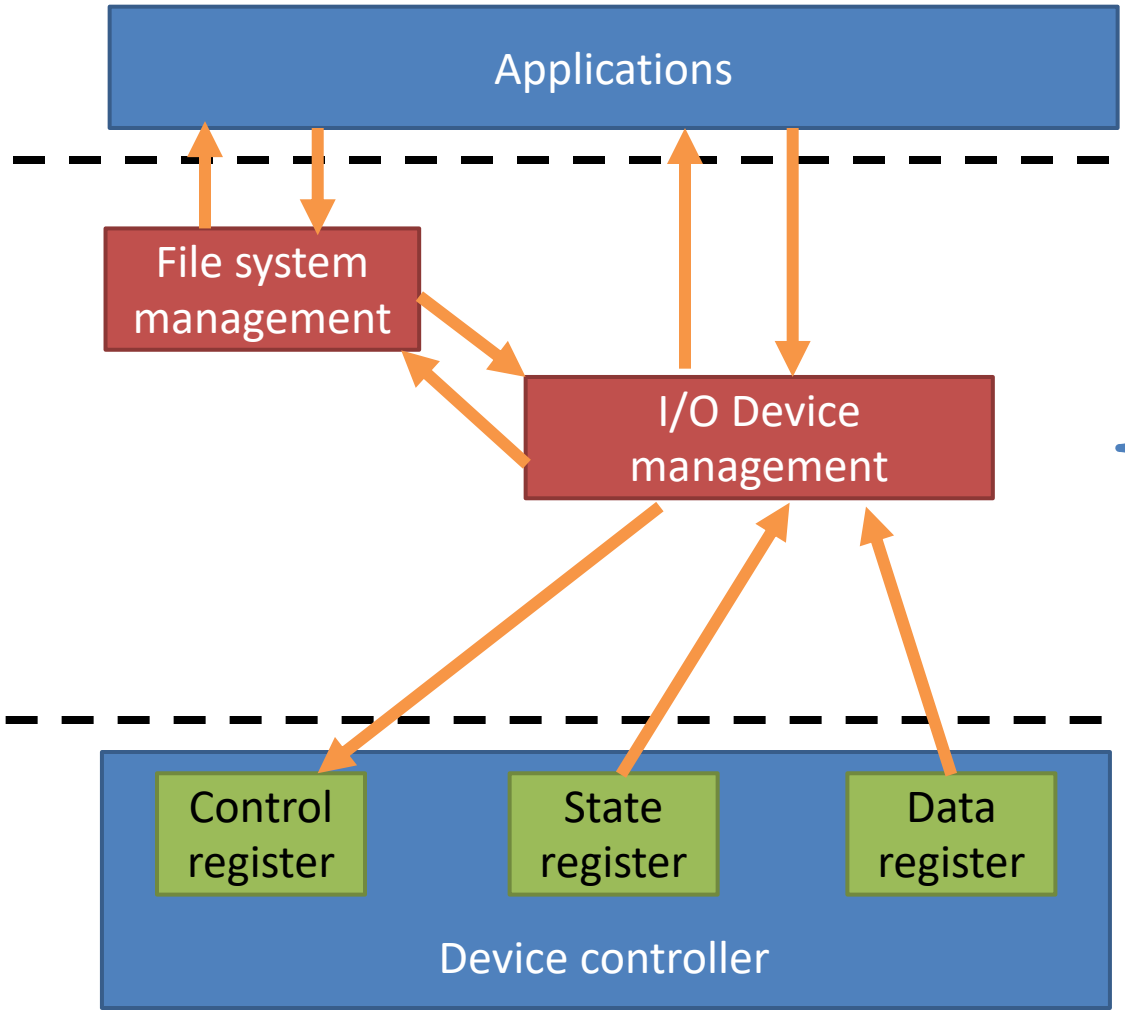
I/O Management Example



I/O Management



I/O Management



Logic I/O

- Device unrelated but unified operations for I/O
  - Resource allocation
  - Garbage collection
  - Data preparation
  - …

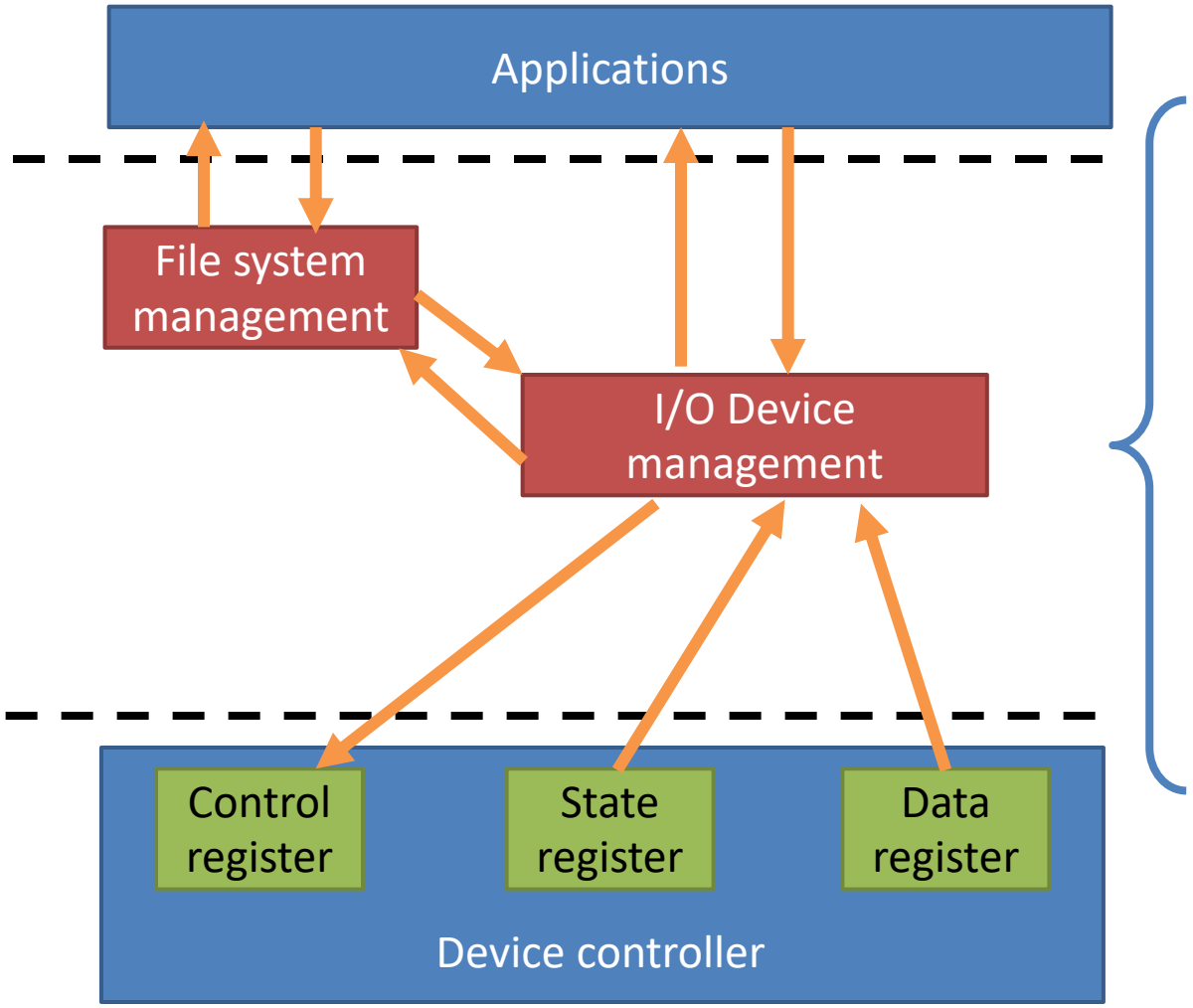
Like the VFS

Logic I/O

- The logic I/O is to perform the I/O functions that are common to all devices and to provide a uniform interface to the user-level software

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

I/O Management



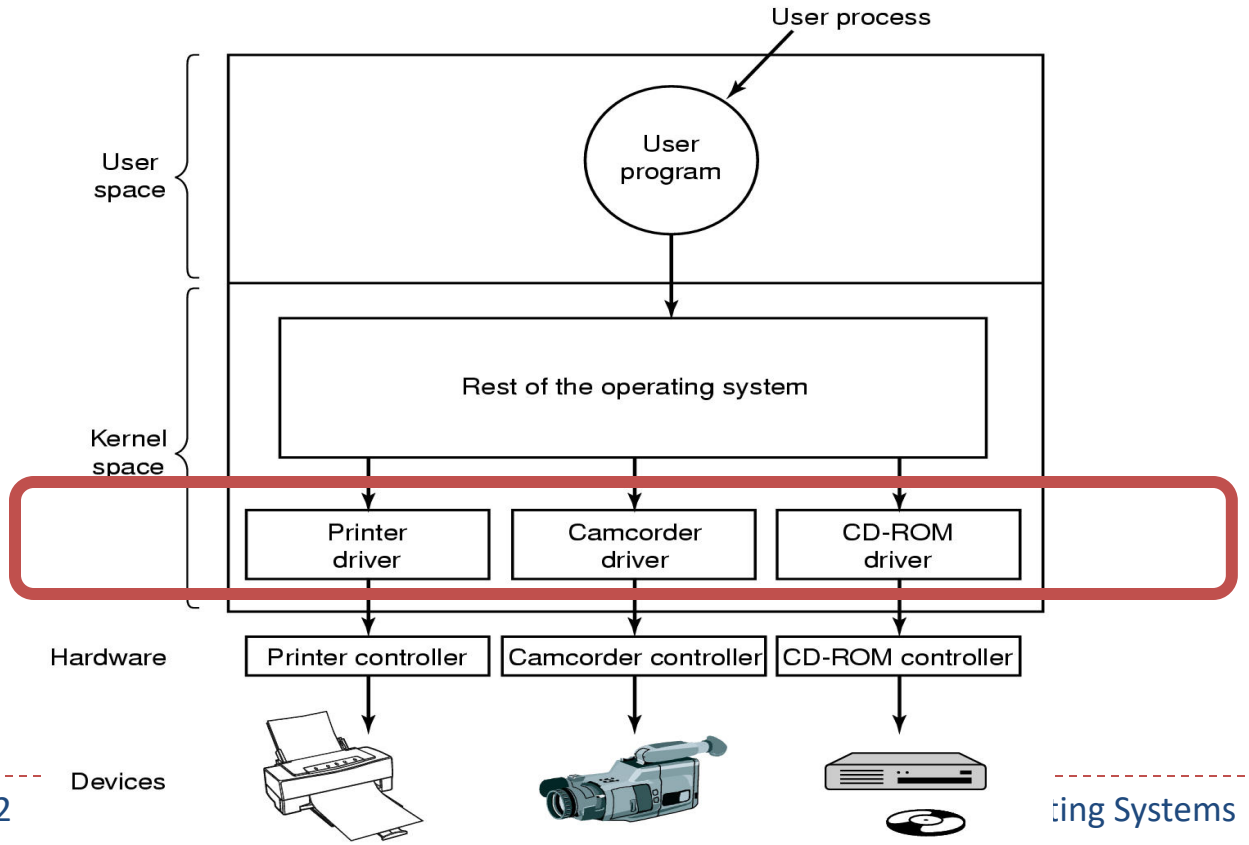
Logic I/O

Device driver

- Device driver is used to interact with and control specific I/O device
- Device driver implements the operation functions on different I/O devices
- Device driver code is large

Device driver

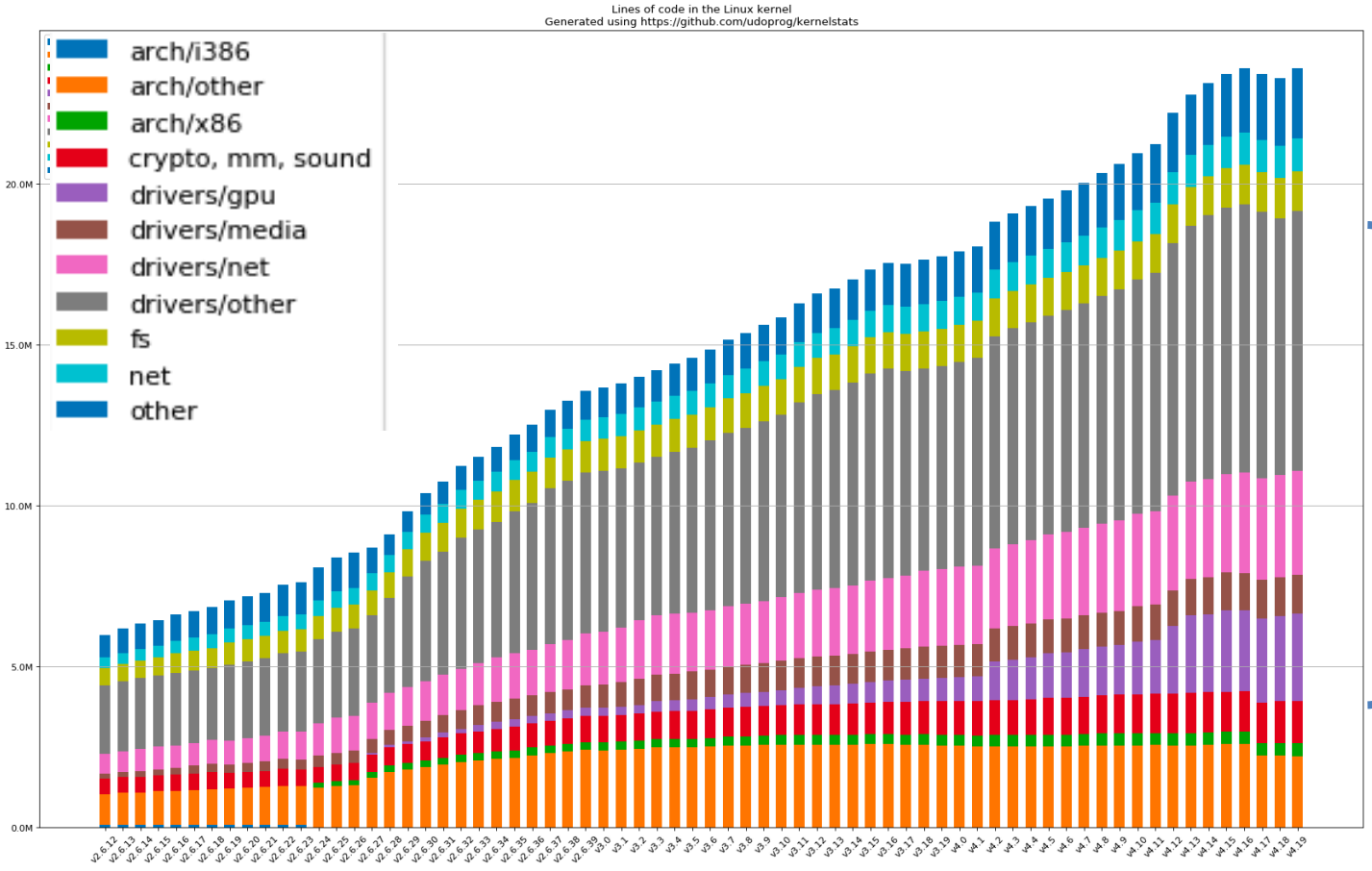
- Device drivers: provided by device's manufacturer, device-specific code for controlling the device (via device controller registers)



Device driver related code in OS

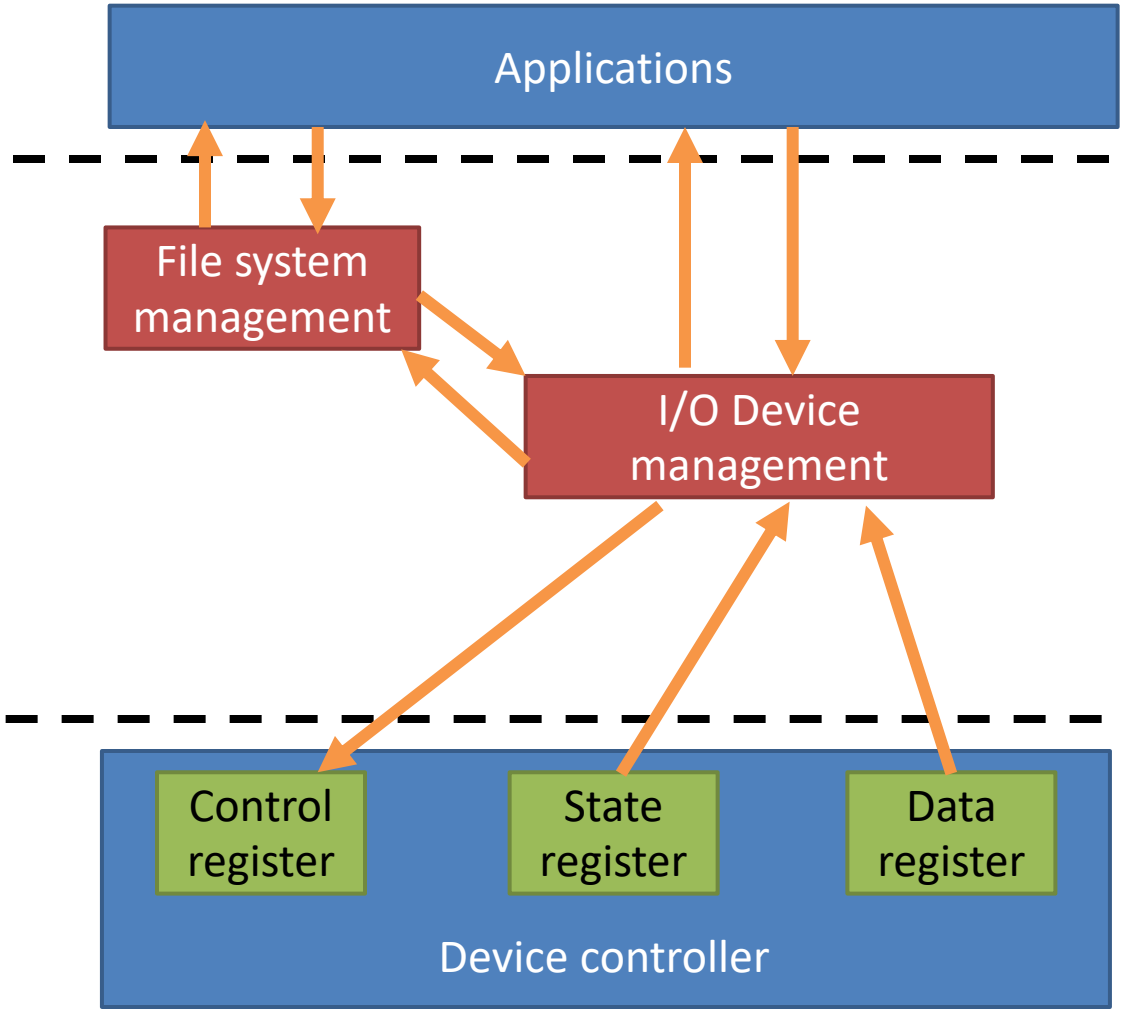
- ~ 20+ million lines of code in Linux

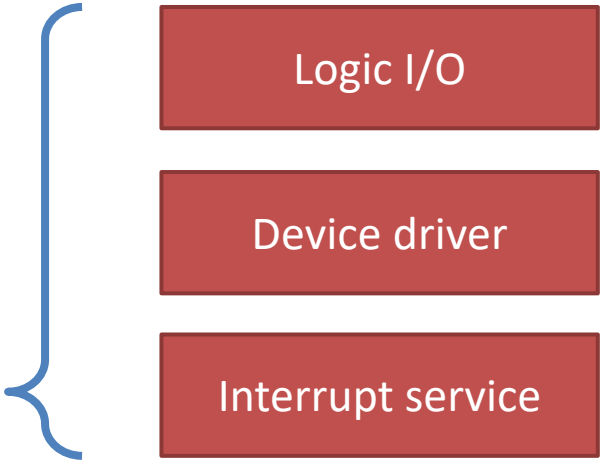
https://elixir.bootlin.com/linux/latest/source



Device driver

I/O Management





- Send interrupts when device is ready or has errors or …
- Send interrupts when data is successfully read, or transmission fails or …

Outline

- I/O System
- Characteristics and Category of I/O Devices
- I/O Device Component
- How CPU works with I/O devices
- I/O Performance

Characteristics of I/O: complexity

- I/O performance is usually bottleneck of the system performance

- I/O → amounts of devices, different performance, parallel read/write → making OS more complex

Characteristics of I/O: complexity → Speed

Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Telephone channel	8 KB/sec
Dual ISDN lines	16 KB/sec
Laser printer	100 KB/sec
Scanner	400 KB/sec
Classic Ethernet	1.25 MB/sec
USB (Universal Serial Bus)	1.5 MB/sec
Digital camcorder	4 MB/sec
IDE disk	5 MB/sec
40x CD-ROM	6 MB/sec
Fast Ethernet	12.5 MB/sec
ISA bus	16.7 MB/sec
EIDE (ATA-2) disk	16.7 MB/sec
FireWire (IEEE 1394)	50 MB/sec
XGA Monitor	60 MB/sec
SONET OC-12 network	78 MB/sec
SCSI Ultra 2 disk	80 MB/sec
Gigabit Ethernet	125 MB/sec
Ultrium tape	320 MB/sec
PCI bus	528 MB/sec
Sun Gigaplane XB backplane	20 GB/sec

Bytes/s

KBytes/s

MBytes/s

Characteristics of I/O: complexity → Application

One dimension

Dynamic move

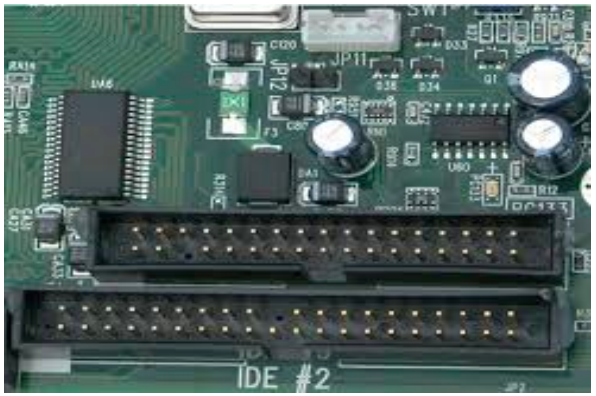
Two dimension

Static content

- Different characteristics of I/O devices make the OS interact with the drivers differently

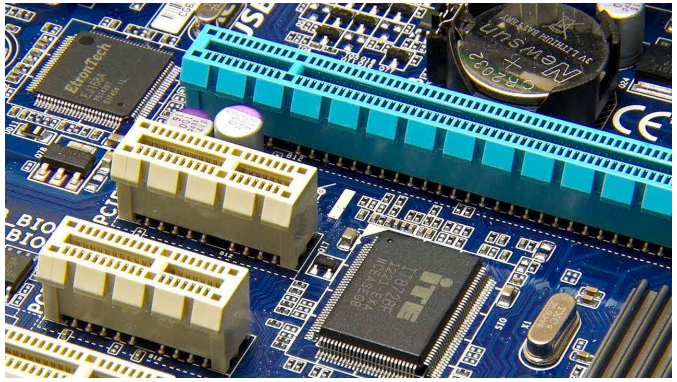
Characteristics of I/O: complexity → Control interface and transmission unit

CD-Rom



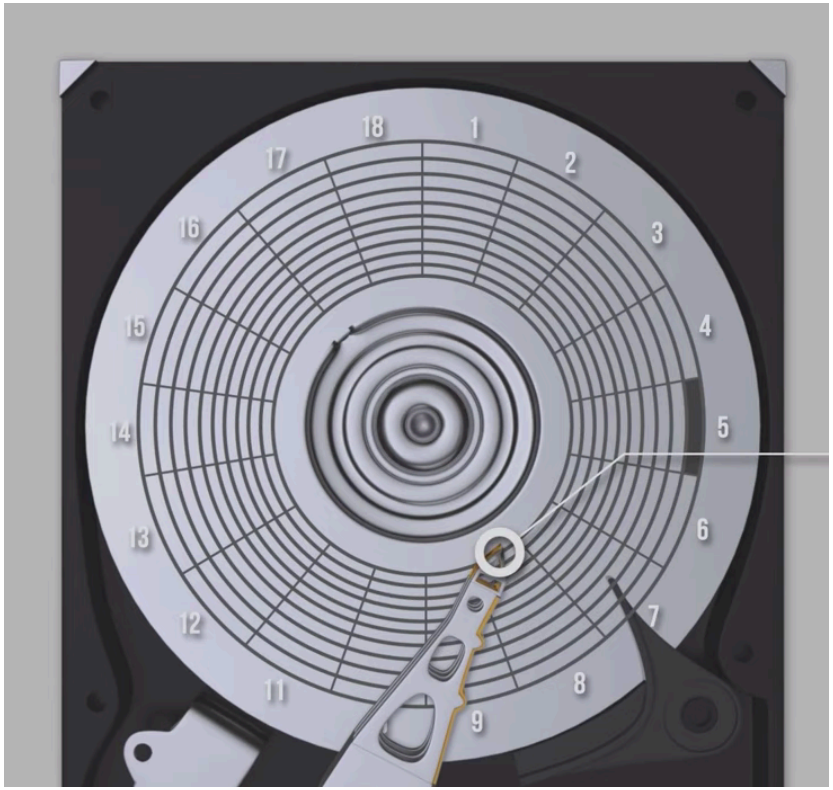
CD-Rom Interface

SSD



PCIe

Characteristics of I/O: complexity → Control interface and transmission unit



- For the same category of I/O device, such as HDD, the transmission unit could be different due to different manufacturer and drivers
  - Byte
  - Sector
  - Block

Category of I/O Devices

- Block devices:
  - ✓ Store or transfer data in blocks
  - ✓ High transmission speed, addressable, support seeking and the random access of data
  - ✓ E.g., CD, SSD, HDD
- Character devices:
  - ✓ Store or transfer data in characters
  - ✓ Low transmission speed, not addressable
  - ✓ E.g., Keyboard, Mouse, Printer

Category of I/O Devices

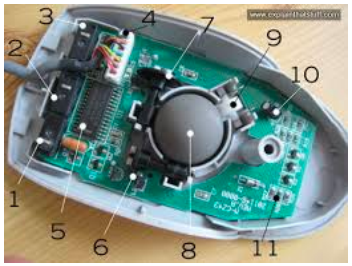
- Storage devices:
  - ✓ Used for temporary or persistent storage
  - ✓ E.g., Disk, Tape
- Transmission devices:
  - ✓ Used to transfer data from different locations
  - ✓ E.g., Ethernet card, Bluetooth, wireless card
- User interface devices:
  - ✓ Used to interact with the system
  - ✓ E.g., Monitor, keyboard, mouse, electronic pen, speaker

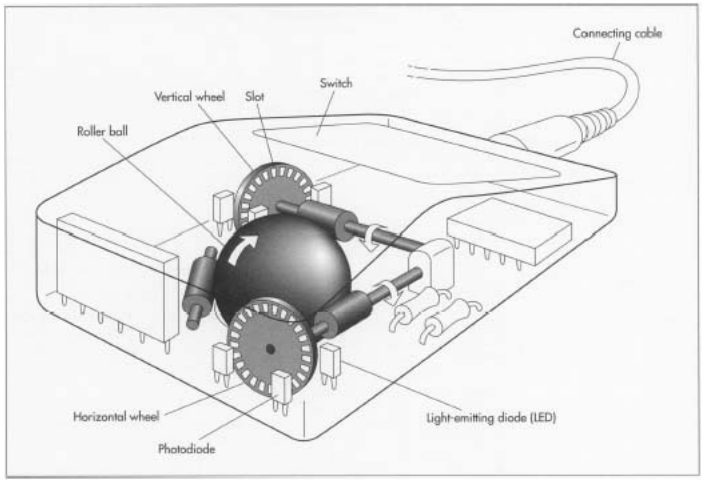
Outline

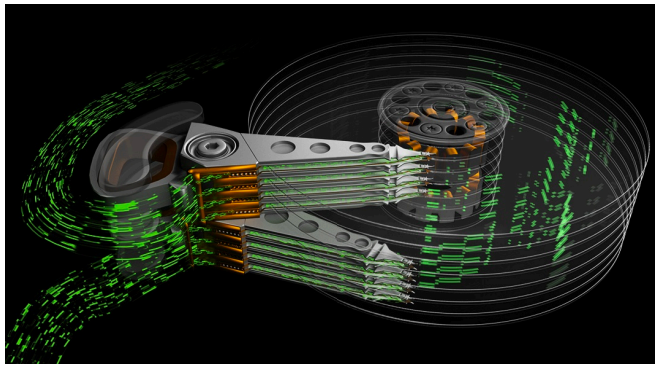
- I/O System
- Characteristics and Category of I/O Devices
- I/O Device Component
- How CPU works with I/O devices
- I/O Performance

I/O Device Component

- I/O devices have two components:
  - **mechanical component (physical part)**
  - electronic component (device controller)



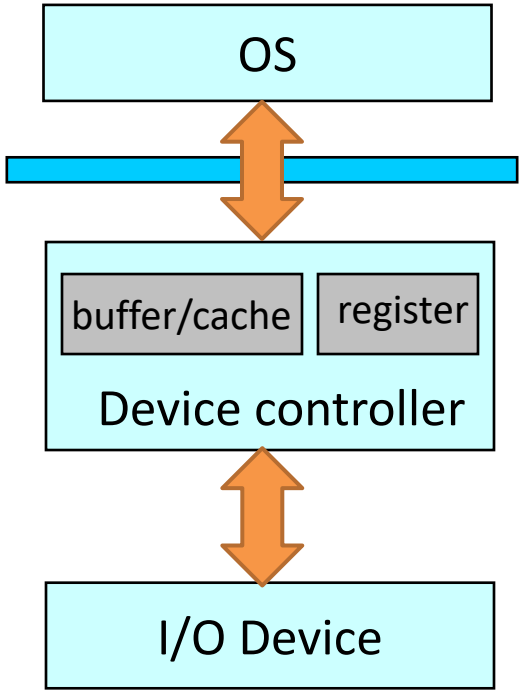




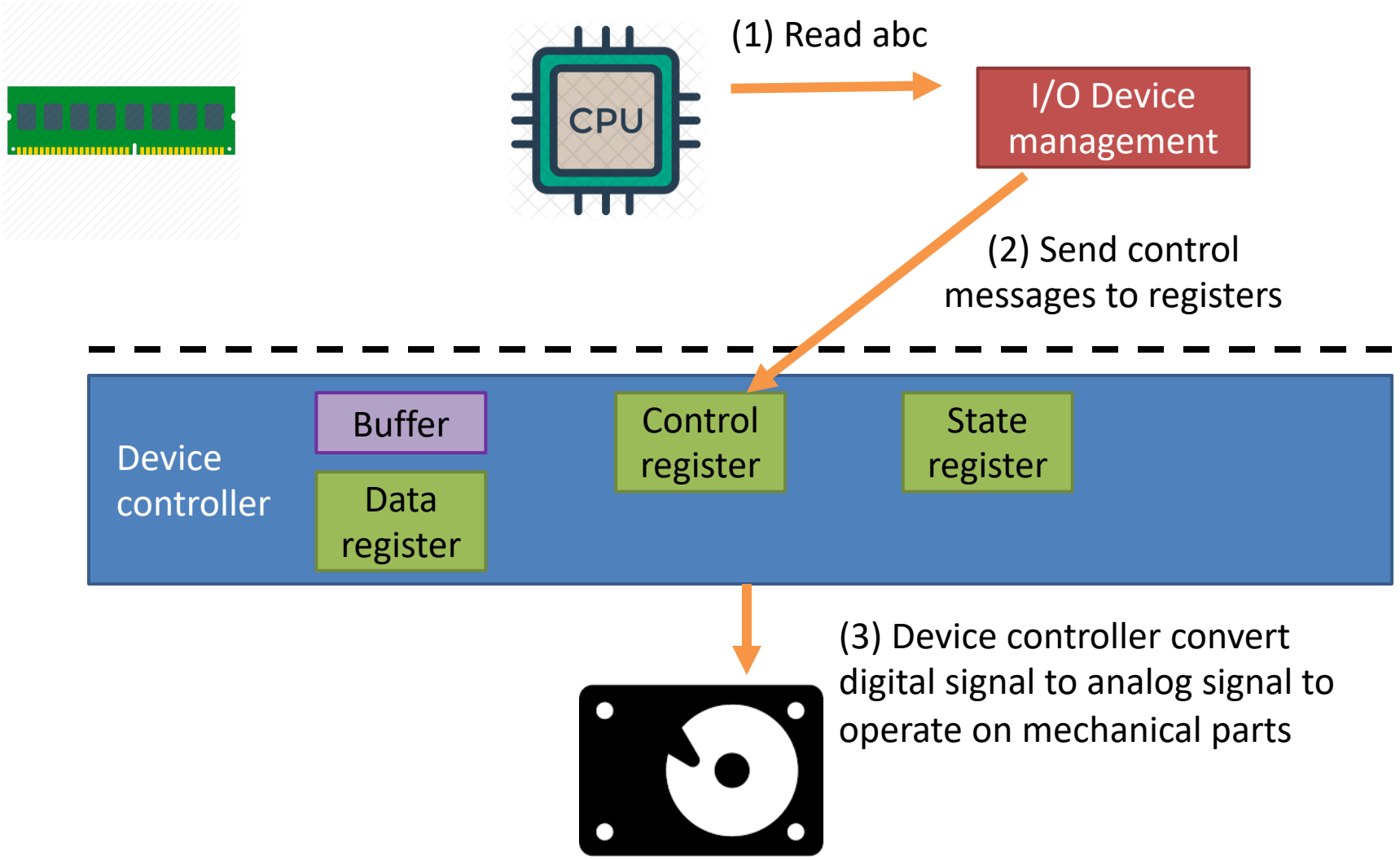
I/O Device Component

- I/O devices have two components:
  - mechanical component (physical part)
  - **electronic component (device controller)**

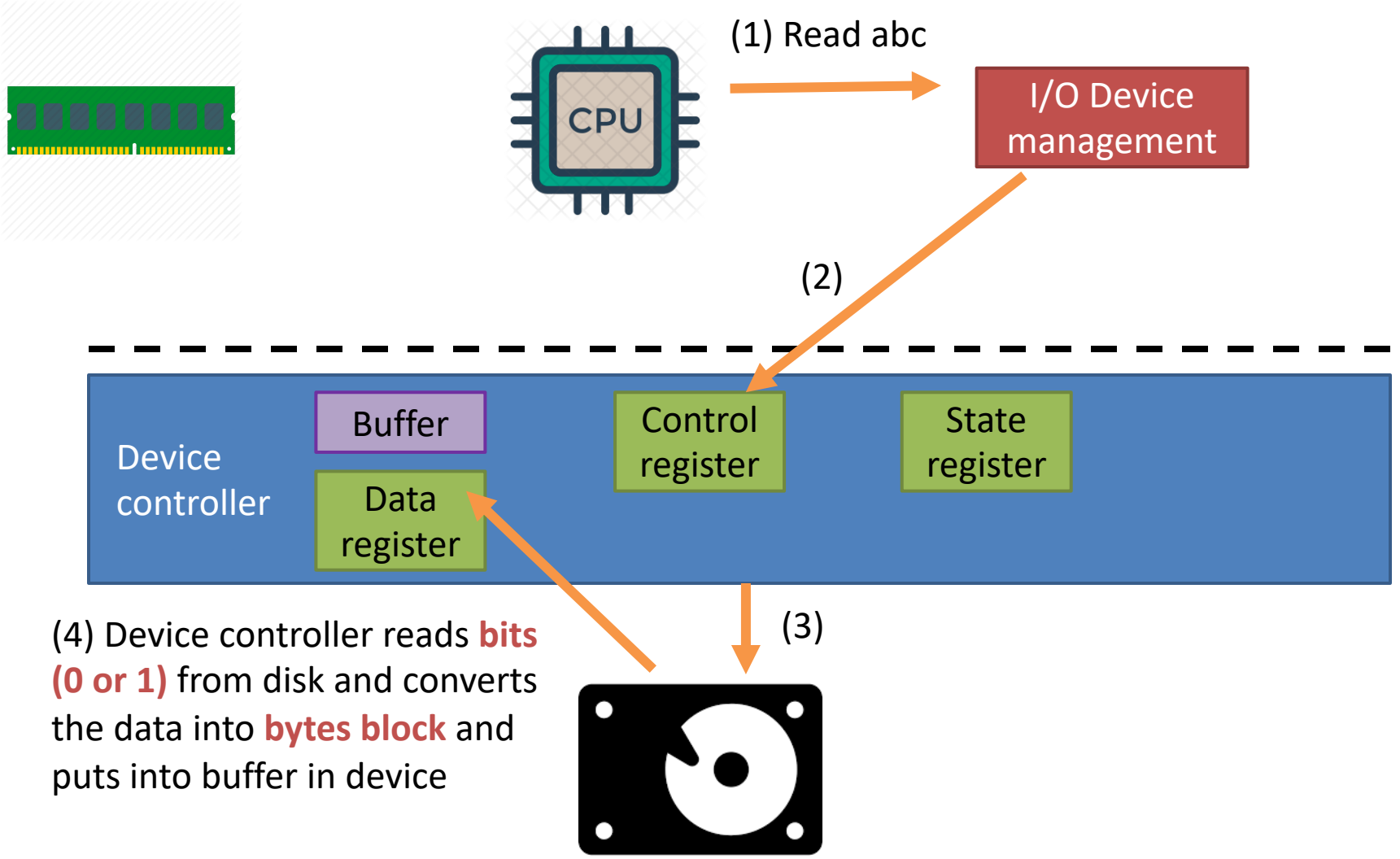
- Controller's tasks
  - Address transformation
  - Receive data and control message from the OS, and send back the data and status message to the OS
  - Convert digital signal into analog signal for the mechanical parts or vice verse
  - Memory buffer, data preprocess, perform error correction as necessary



I/O Device Component: Device controller

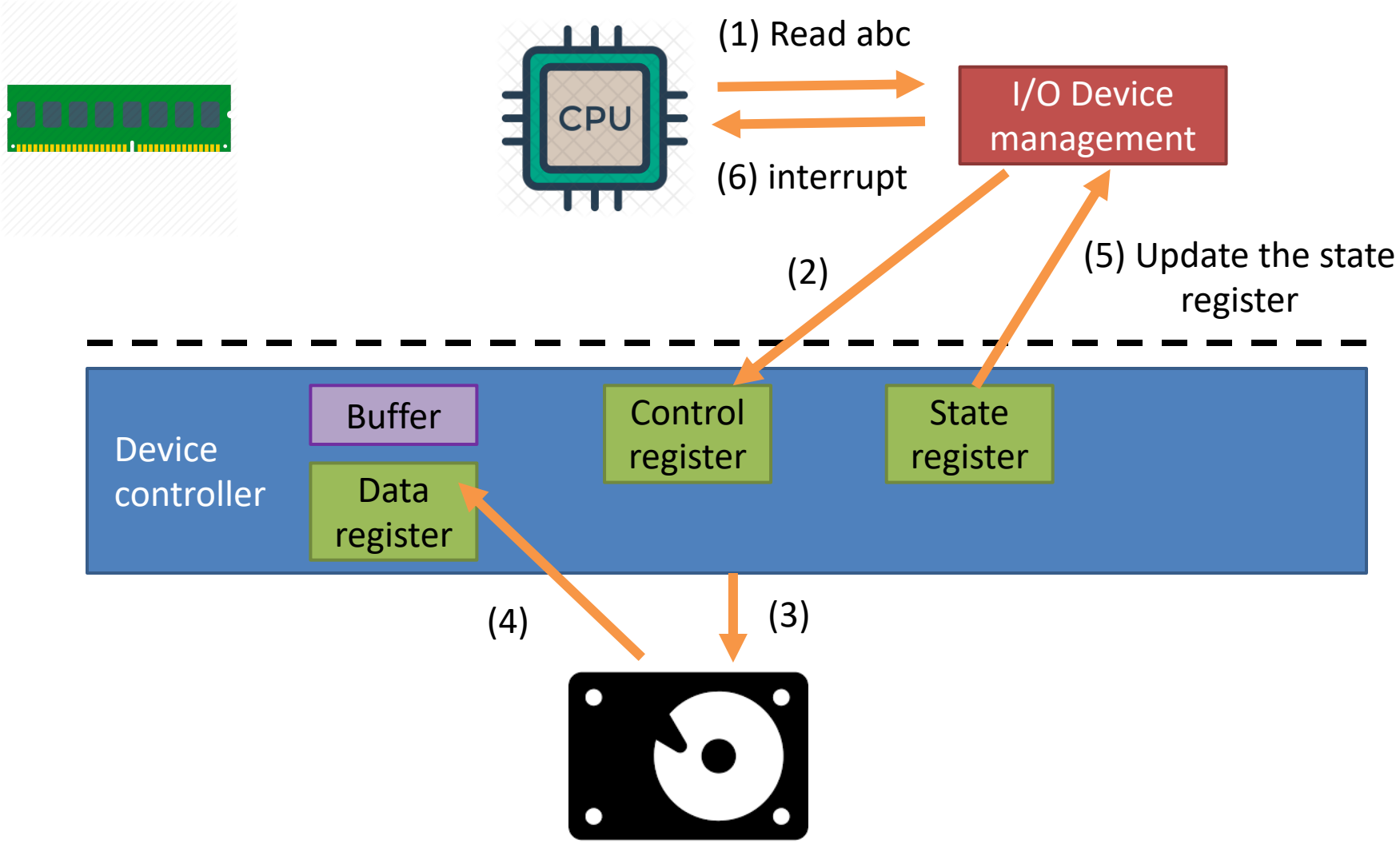


I/O Device Component: Device controller

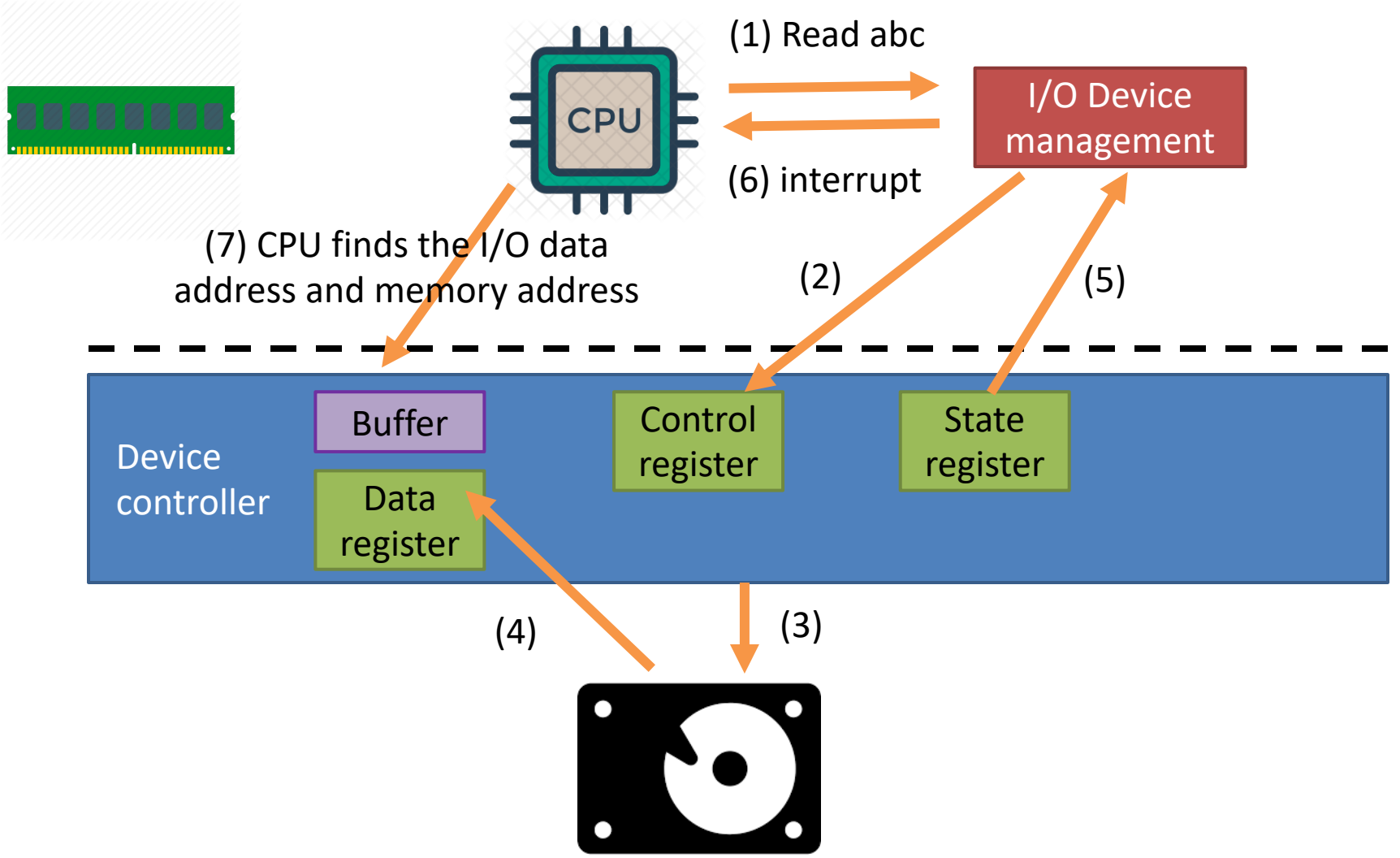


(4) Device controller reads **bits (0 or 1)** from disk and converts the data into **bytes block** and puts into buffer in device

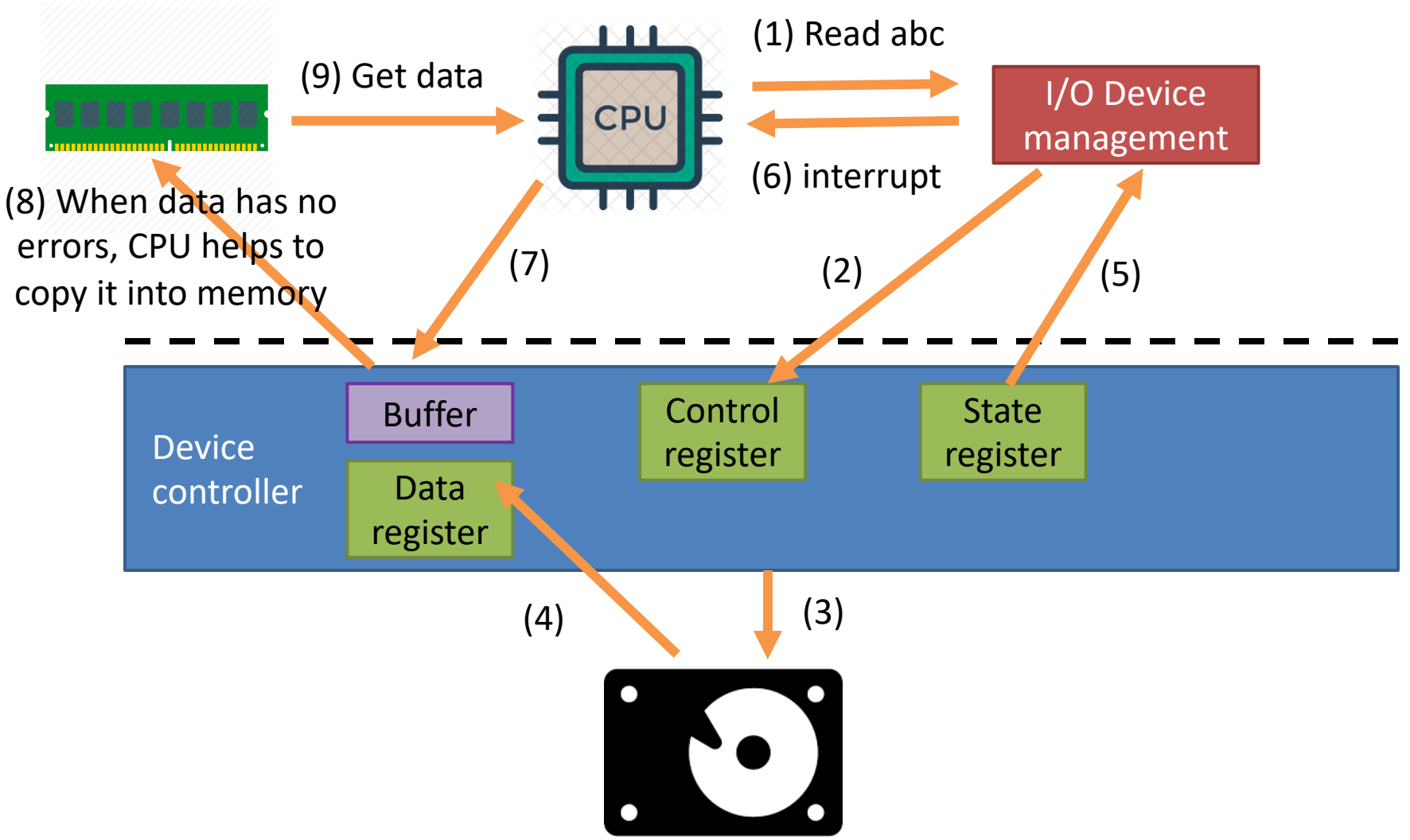
I/O Device Component: Device controller



I/O Device Component: Device controller



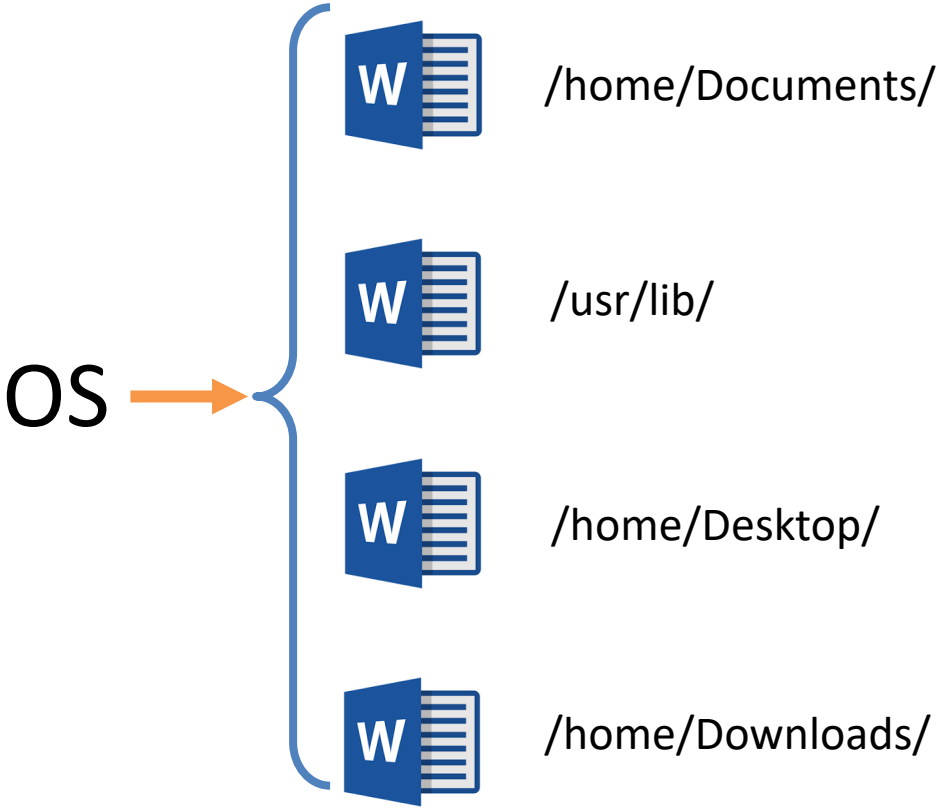
I/O Device Component: Device controller



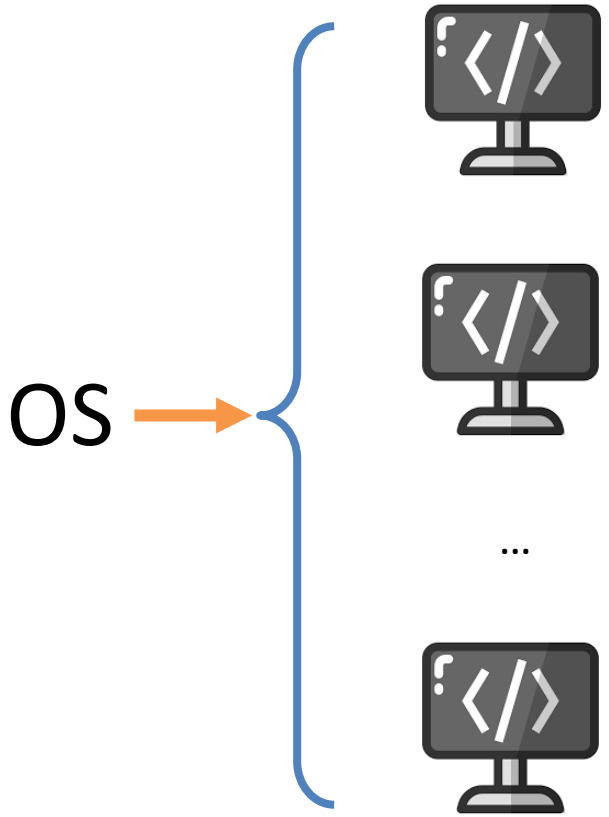
Outline

- I/O System
- Characteristics and Category of I/O Devices
- I/O Device Component
- How CPU works with I/O devices
- I/O Performance

How does the CPU(OS) address the I/O devices?



Different files have its unique address on disk



Addressing?

How does the CPU(OS) address the I/O devices?

- Method 1: set fixed address for I/O devices

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)

How does the CPU(OS) address the I/O devices?

- Method 1: set fixed address for I/O devices

I/O address

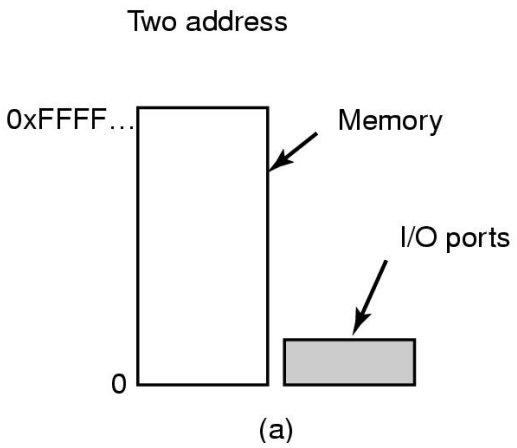
I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

memory

Data and instruction

How does the CPU(OS) address the I/O devices?

- Method 1: set fixed address for I/O devices



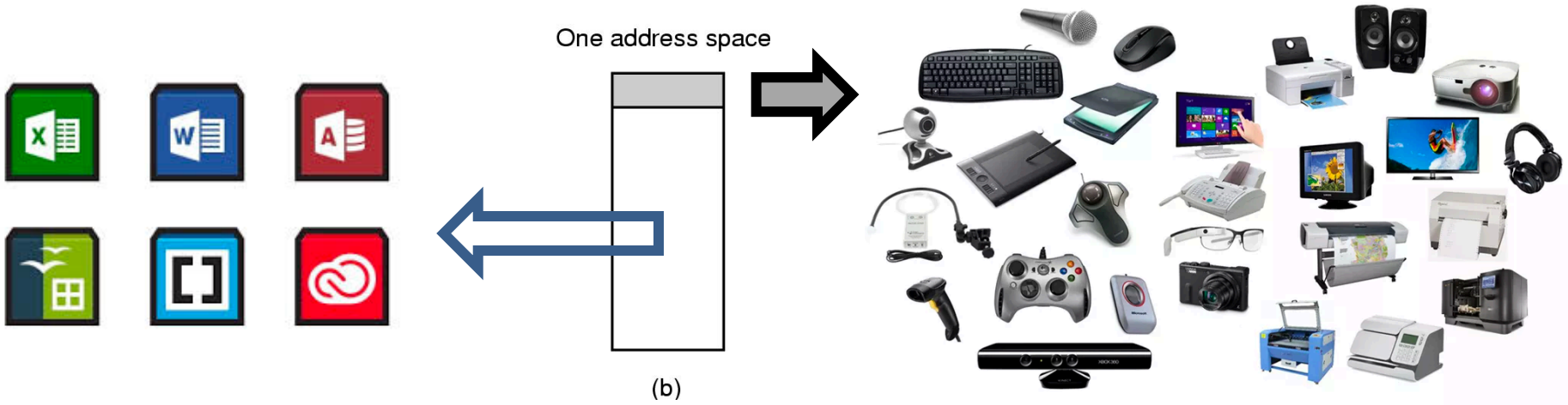
(a) Separate I/O and memory space

- Advantages:
  - Independent from main memory address
  - Easy to program on I/O
- Disadvantages:
  - Fixed range and size, not flexible, extra space for I/O

How does the CPU(OS) address the I/O devices?

- Method 2: memory-mapped I/O

- A portion of memory address space is mapped to the device, and communications occur by reading and writing directly to/from those memory areas

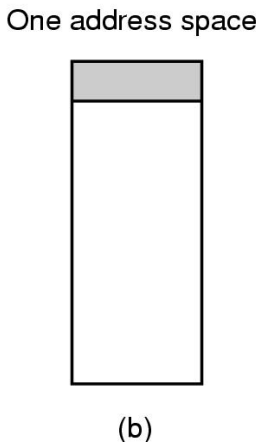


(b) Memory-mapped I/O;

How does the CPU(OS) address the I/O devices?

- Method 2: memory-mapped I/O

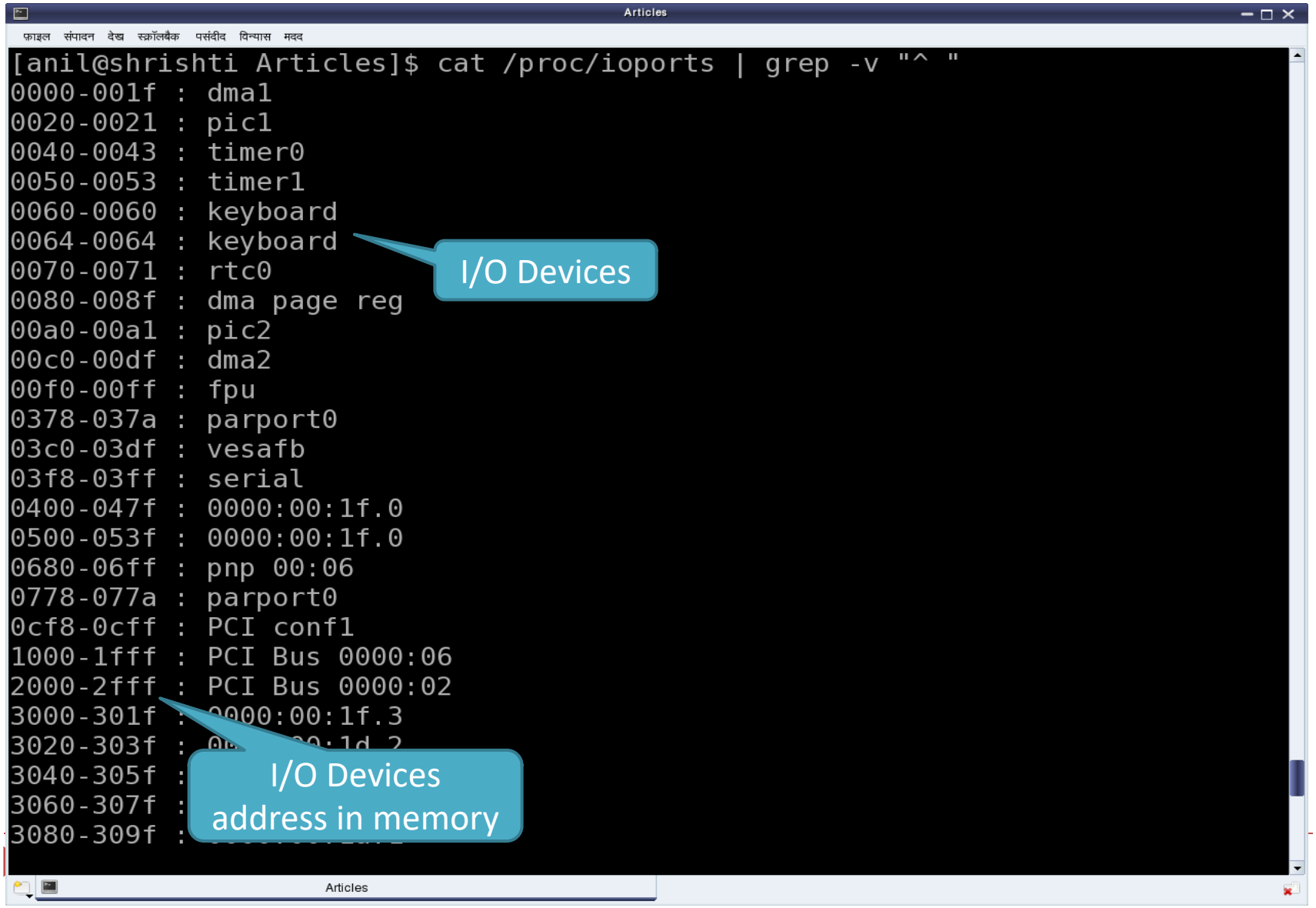
- Advantages:
  - No need extra space for I/O, unified under memory operation
  - I/O can take large address space



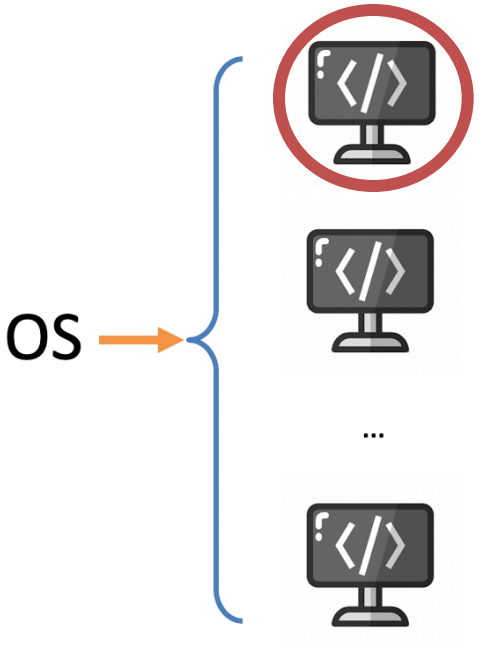
(b) Memory-mapped I/O;

- Disadvantages:
  - Take part of the main memory address space

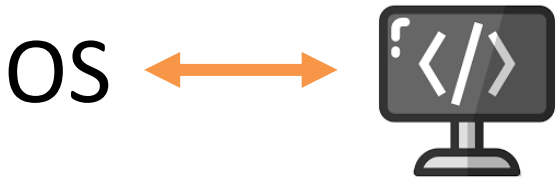
Memory-mapped I/O Devices in Linux



How does the CPU(OS) communicate with the I/O devices?



After the OS finds one I/O device, next is interaction.

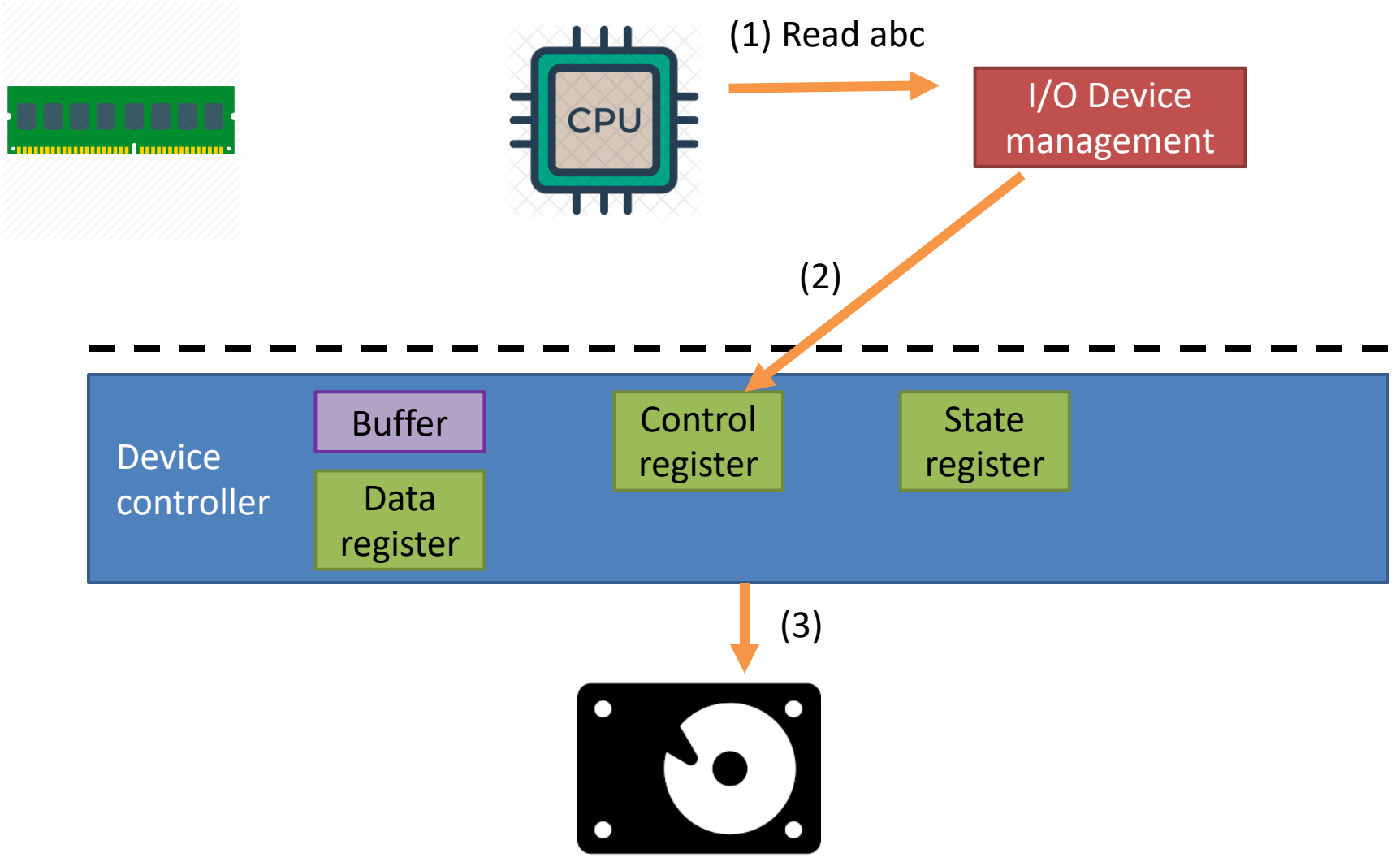


- The OS needs to know when:
  - The I/O device has completed an operation
  - The I/O operation has encountered an error

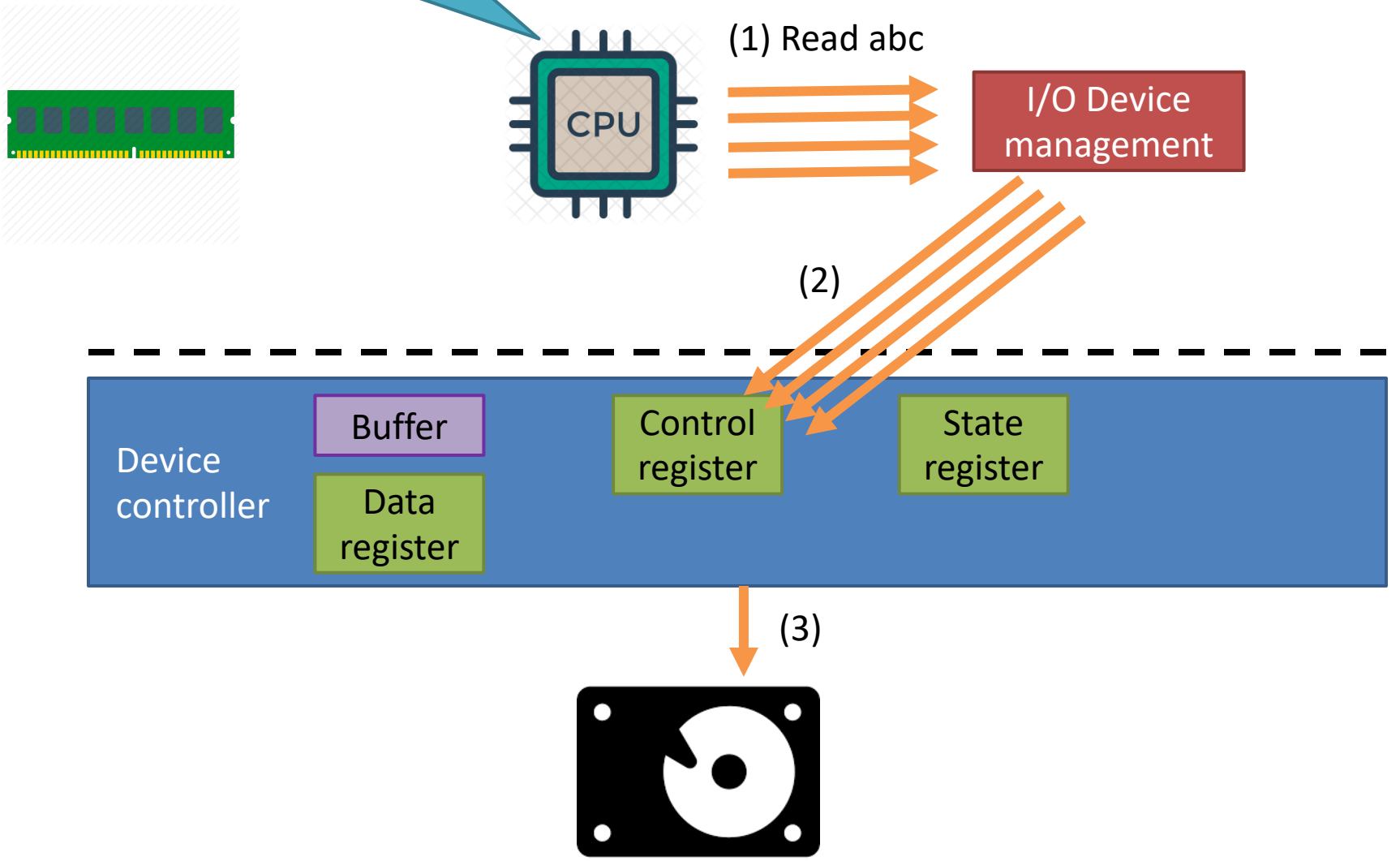
How does the CPU(OS) communicate with the I/O devices?

- Option 1: Polling (Busy Waiting):
  - The I/O device puts information in a status register
  - The CPU periodically or continuously checks the status register

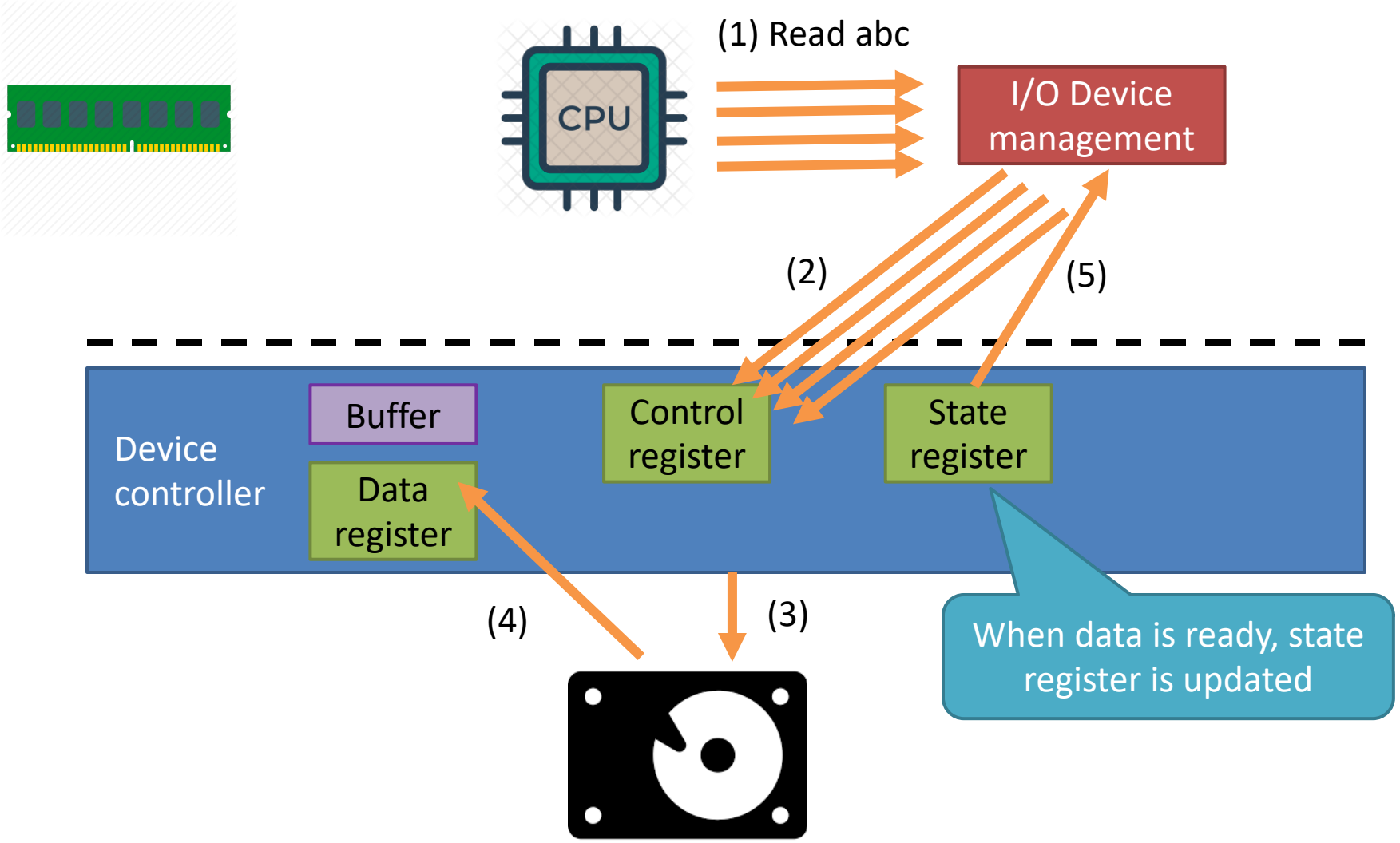
How does the CPU(OS) communicate with the I/O devices? Polling if data not ready



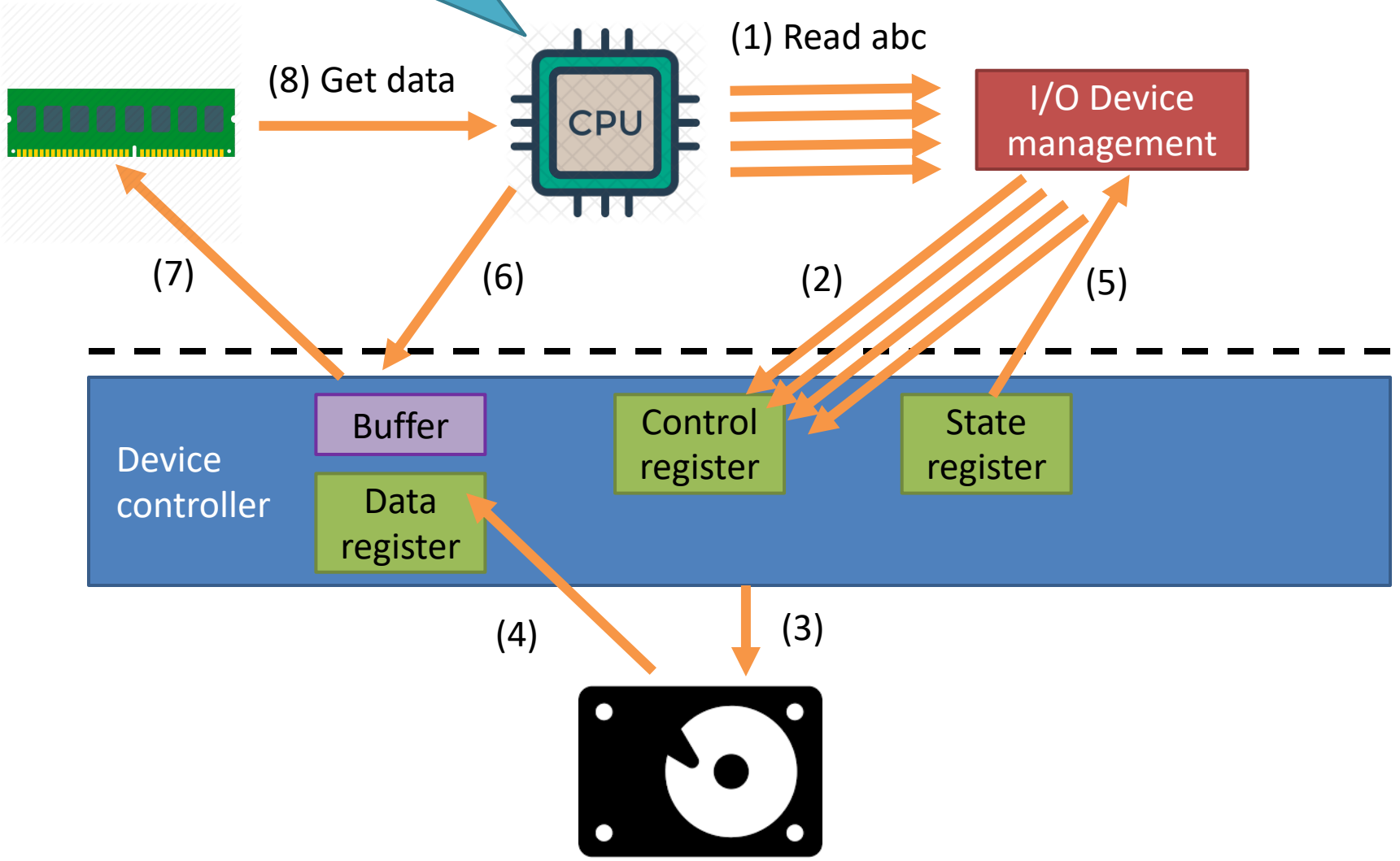
PU(OS) communicate with
? Polling if data not ready



How does the CPU(OS) communicate with the I/O devices? Polling if data is ready



Ho PU(OS) communicate with th ? Polling if data is ready



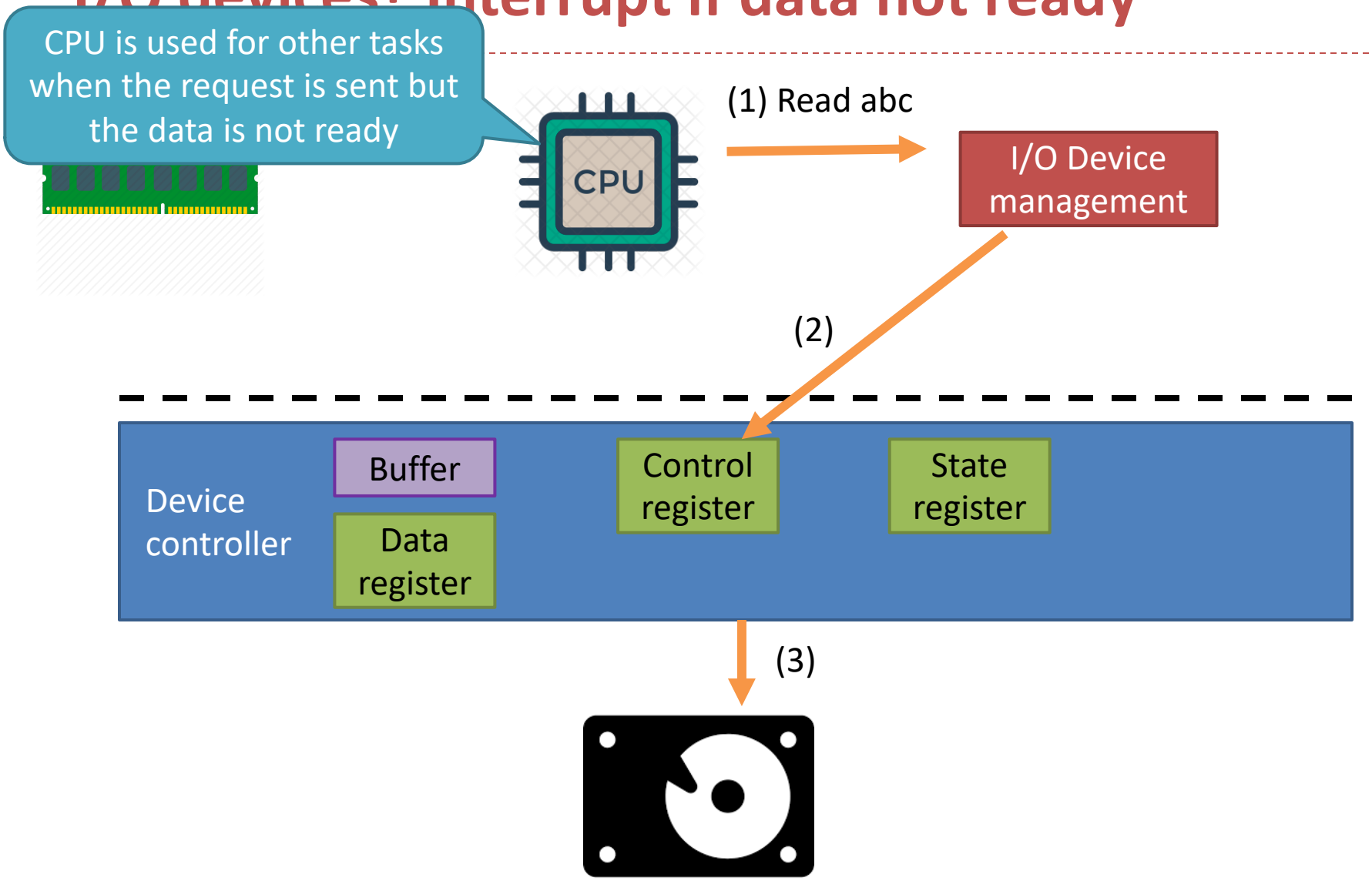
How does the CPU(OS) communicate with the I/O devices?

- Option 1: Polling (Busy Waiting):
  - The I/O device puts information in a status register
  - The CPU periodically or continuously checks the status register
- Advantage:
  - Simple
- Disadvantage:
  - Polling overhead can consume a lot of CPU time

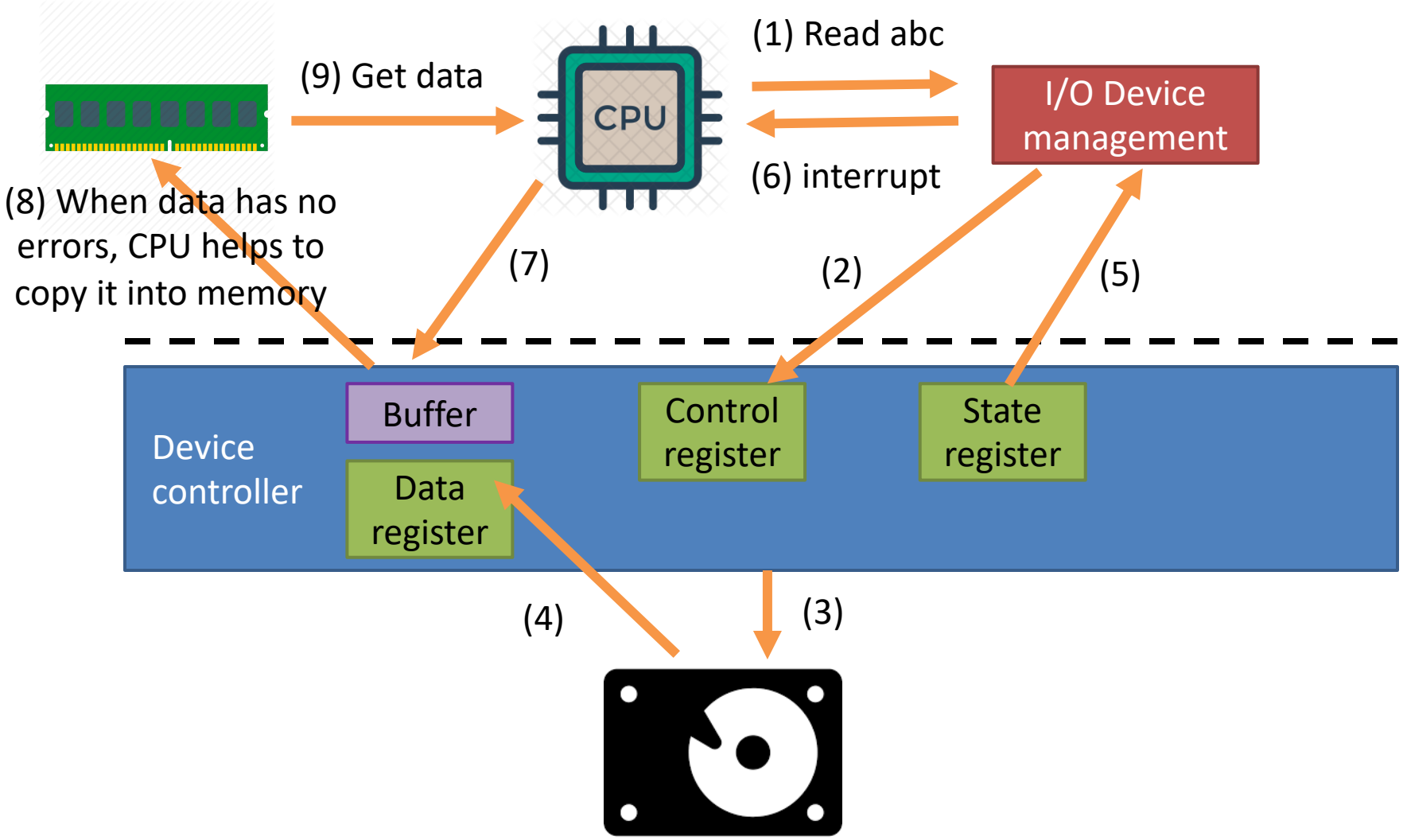
How does the CPU(OS) communicate with the I/O devices?

- Option 2: I/O Interrupt:
  - Whenever an I/O device needs attention from the processor, it interrupts the processor from what it is currently doing.

How does the CPU(OS) communicate with the I/O devices? Interrupt if data not ready



How does the CPU(OS) communicate with the I/O devices? Interrupt if data is ready

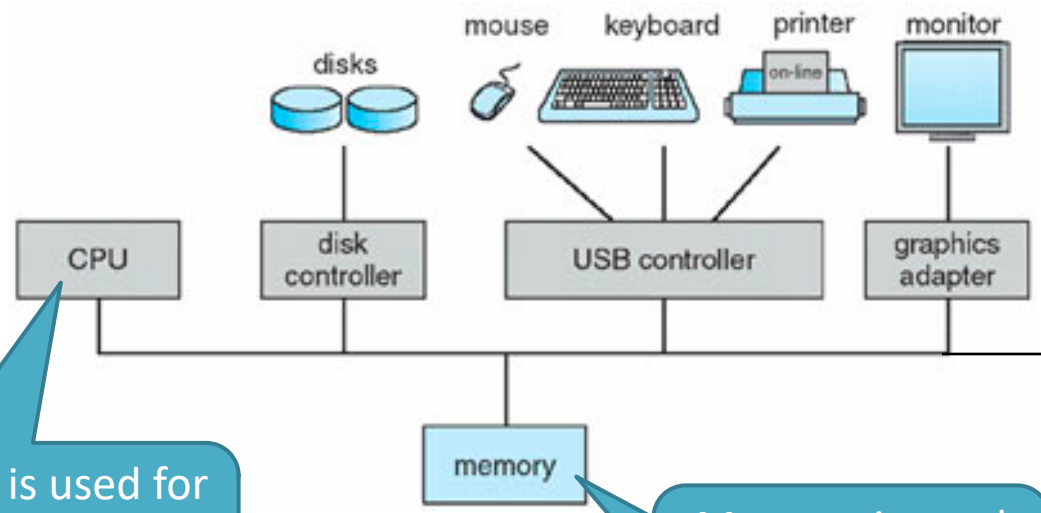


How does the CPU(OS) communicate with the I/O devices?

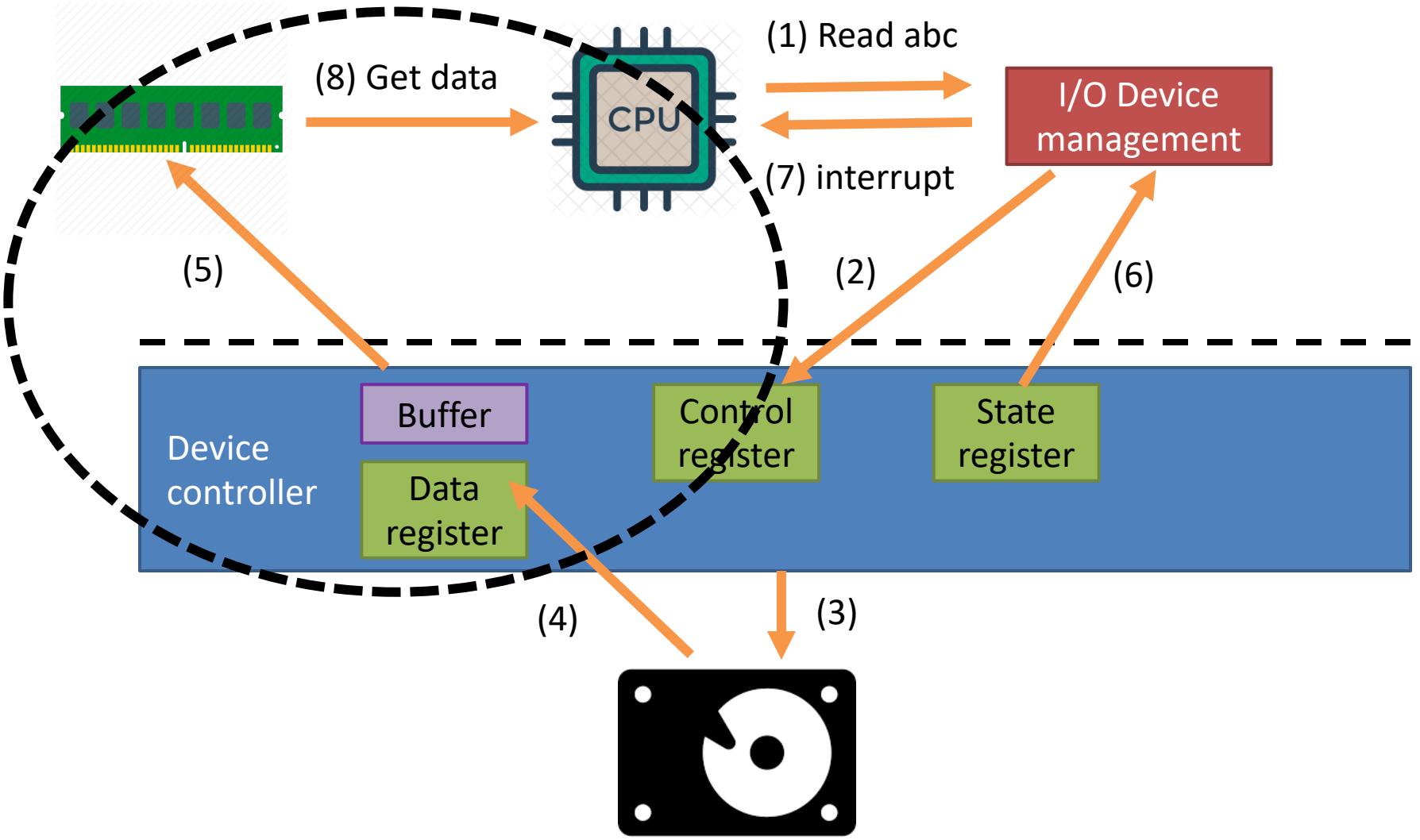
- Option 2: I/O Interrupt:
  - Whenever an I/O device needs attention from the processor, it interrupts the processor from what it is currently doing.
- Advantage:
  - User program progress is only halted during actual transfer
- Disadvantage
  - Save the proper states to resume after the interrupt (processor)
  - Too many interrupts will cause lots of context switch overhead

How does the CPU(OS) communicate with the I/O devices?

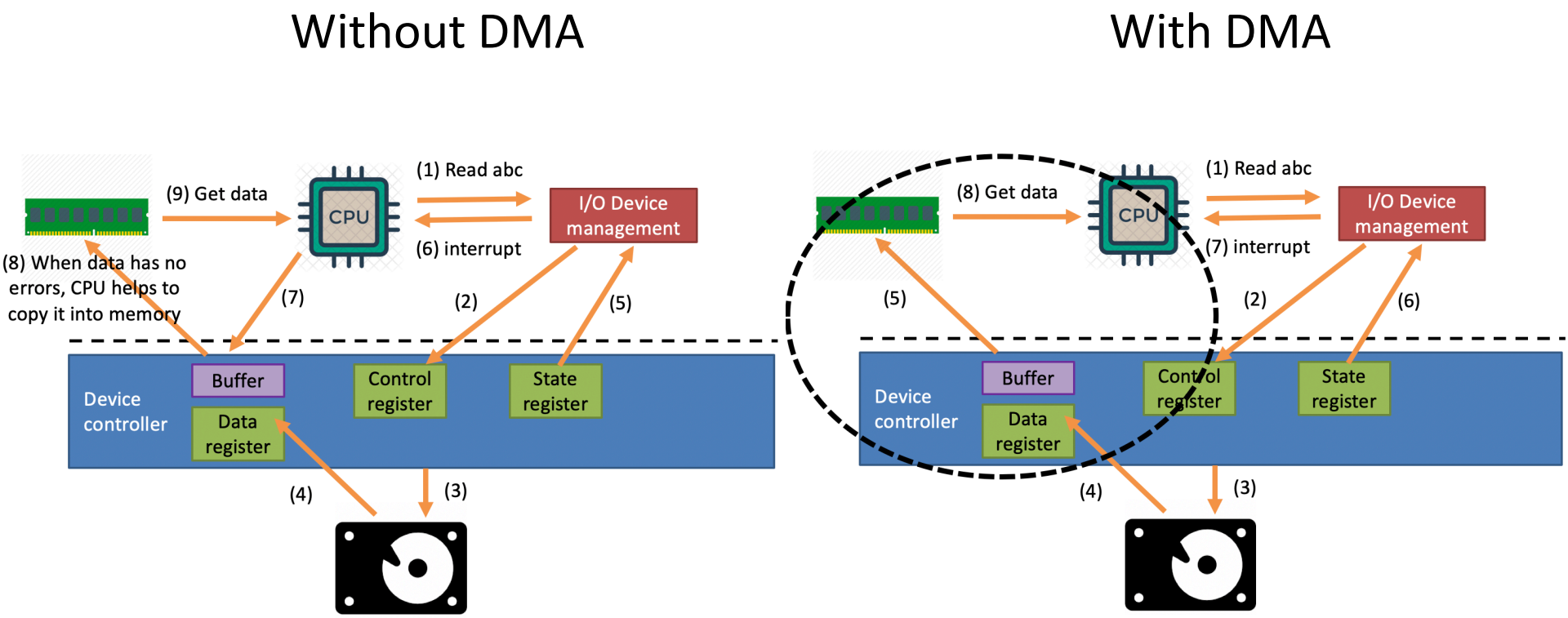
- Option 3: Direct Memory Access (DMA):
  - DMA controller has access to system bus independent of CPU
  - The I/O can transfer data to memory directly



How does the CPU(OS) communicate with the I/O devices? DMA



How does the CPU(OS) communicate with the I/O devices? DMA



How does the CPU(OS) communicate with the I/O devices? DMA

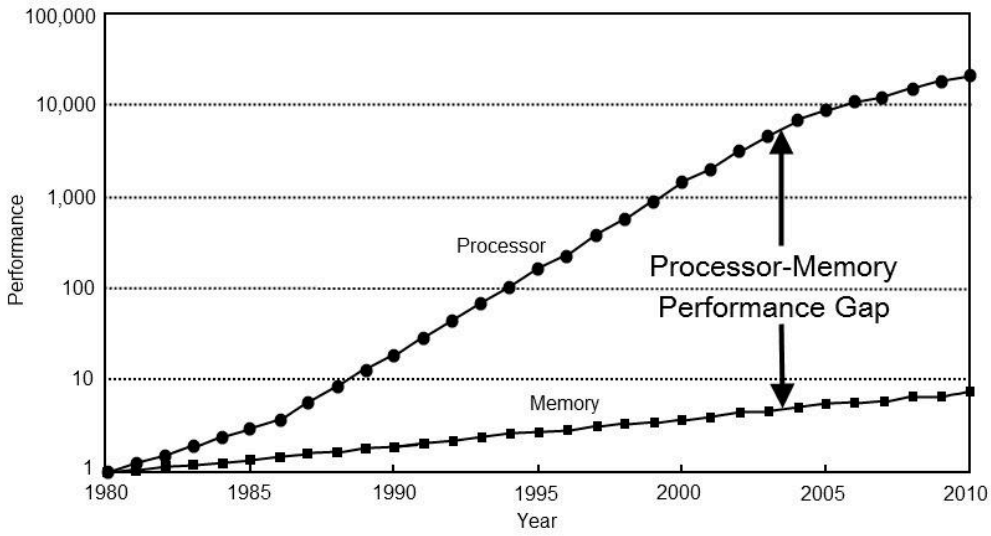
	Without interrupt	With interrupt
Using CPU to transfer data between memory and I/O	Polling I/O	Interrupt I/O
Transfer data between memory and I/O directly	/	DMA

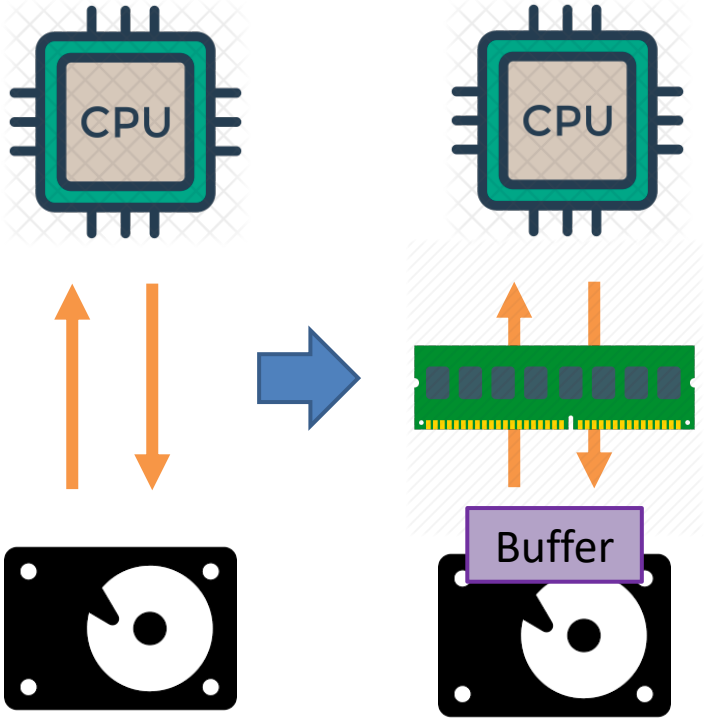
Outline

- I/O System
- Characteristics and Category of I/O Devices
- I/O Device Component
- How CPU works with I/O devices
- I/O Performance

I/O performance - Buffering

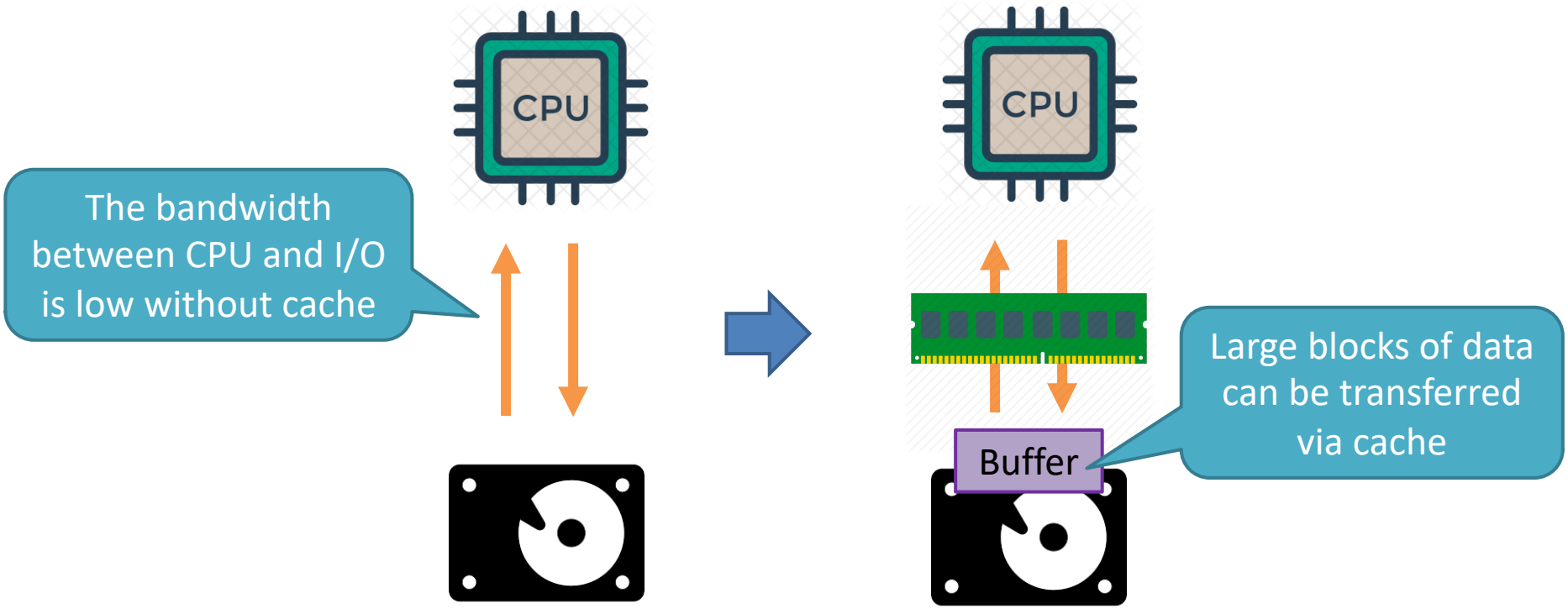
- Bridge the gap between CPU and I/O speed





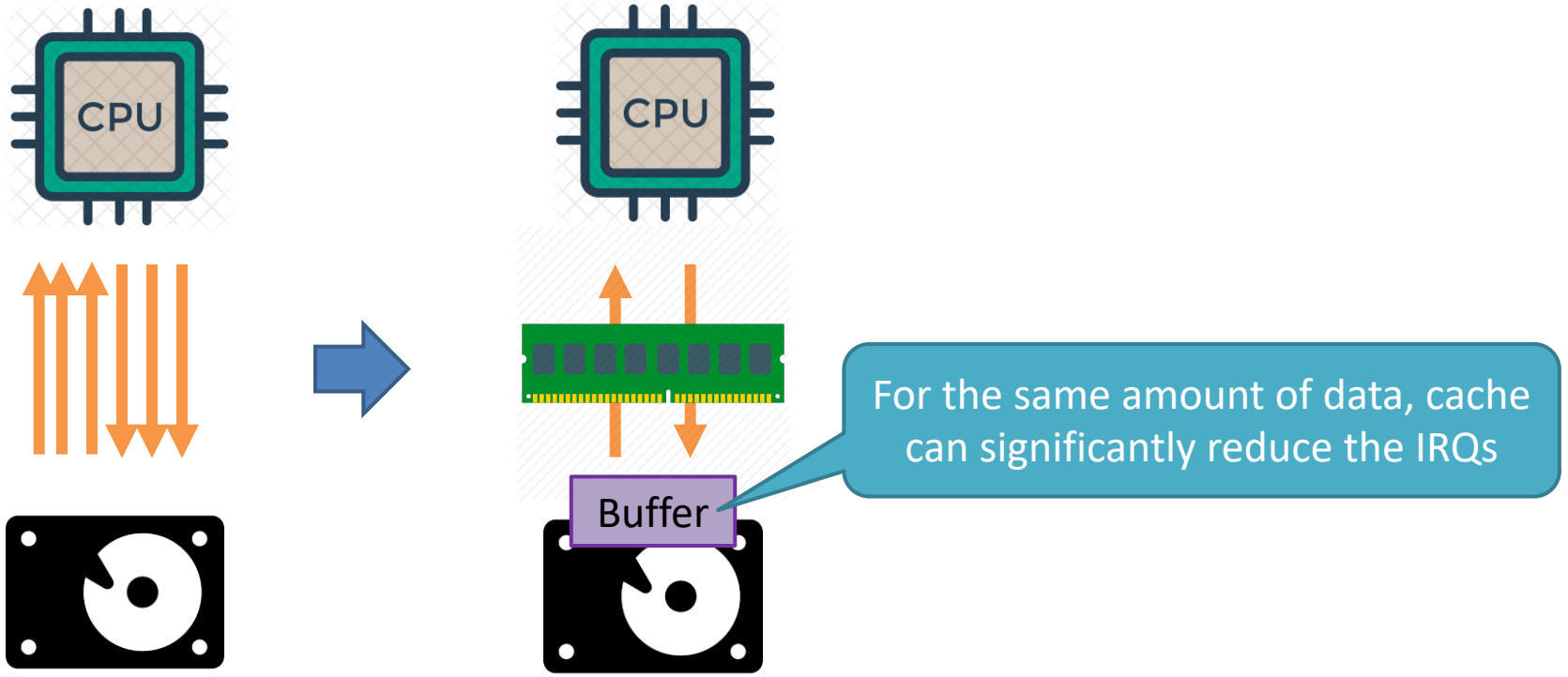
I/O performance - Buffering

- Increase the parallelism between CPU and I/O



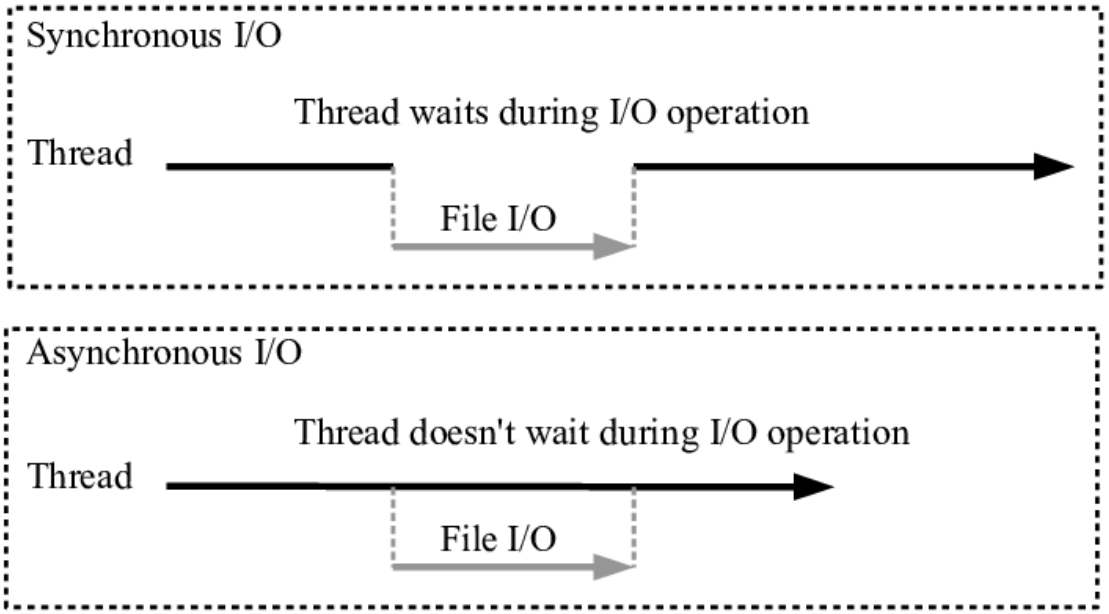
I/O performance - Buffering

- Reduce the interrupts from I/O to CPU



I/O performance – synchronous/asynchronous I/O

- **Synchronous I/O:** application is blocked until the I/O operation finishes
- **Asynchronous I/O:** application is not blocked and can does other tasks during the I/O operation is processing



I/O Performance Summary

- I/O execution is complex and slow compared to the CPU/memory

- Ways to improve the I/O performance
  - Bridge the speed gap --> Buffering
  - Do not let CPU wait for I/O --> asynchronous I/O
  - Make I/O access to memory directly without CPU --> DMA

Conclusion

- I/O System
  - Structure and management
- Characteristics and Category of I/O Devices
  - Block vs character, storage vs transmission vs UI
- I/O Device Component
  - Mechanical parts and device controller
- How CPU works with I/O devices
  - Addressing and communication (poll, interrupt, DMA)
- I/O Performance
  - Buffering, asynchronous I/O, DMA