THE UNIVERSITY *of* EDINBURGH
informatics

Operating Systems (INFR10079)

2023/2024 Semester 2

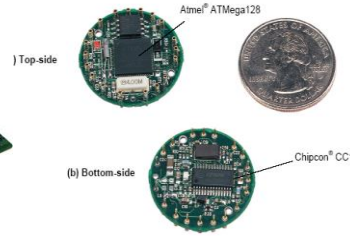
Introduction

(Operating Systems and Hardware)

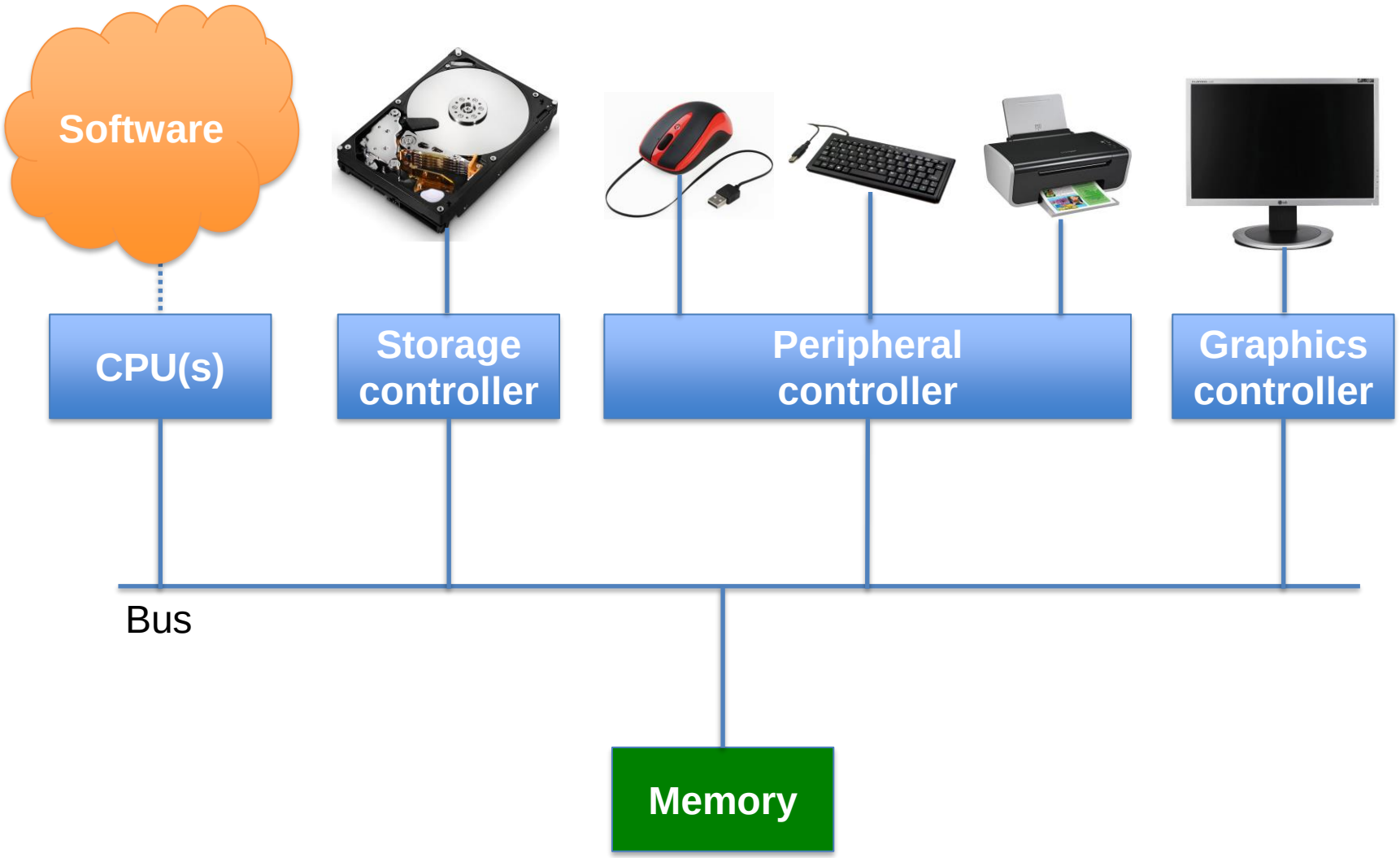
abarbala@inf.ed.ac.uk

Chapter 1.1, 1.2, 1.3.1, 1.3.2, 1.4.2

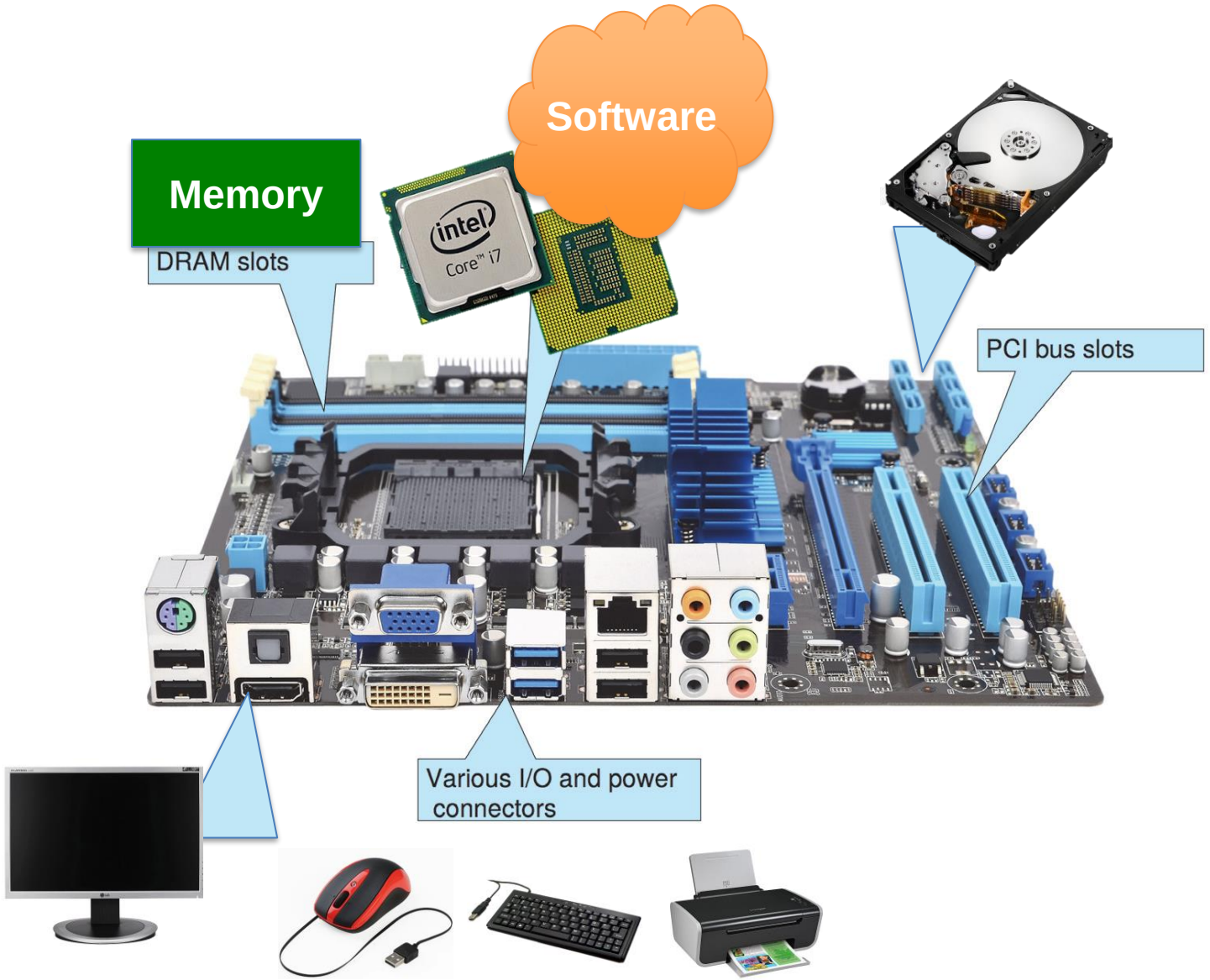
Computing Systems are Everywhere



Modern Computer System



Modern Computer System – PC Motherboard



Starting a modern computer with Linux Operating System

What is an Operating System?

- A program that manages a computer's hardware
- A program that acts as an intermediary between the user of a computer and computer hardware
- Likely, a big program
  - "The **Linux Kernel** Enters 2020 At **27.8 Million Lines** In Git But With Less Developers For 2019", **1 January 2020** at 09:14 AM EST
    - https://www.phoronix.com/scan.php?page=news_item&px=Linux-Git-Stats-EOY2019
  - **March 2021**, "Linux 5.12 Coming In At Around **28.8 Million Lines**"
    - https://www.phoronix.com/news/Linux-5.12-rc1-Code-Size

Operating Systems

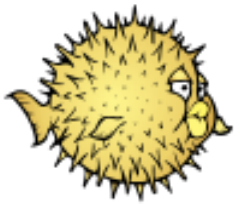
MS-DOS

solaris

iOS

Mac OS X

Windows



hp UX

MINIX 3

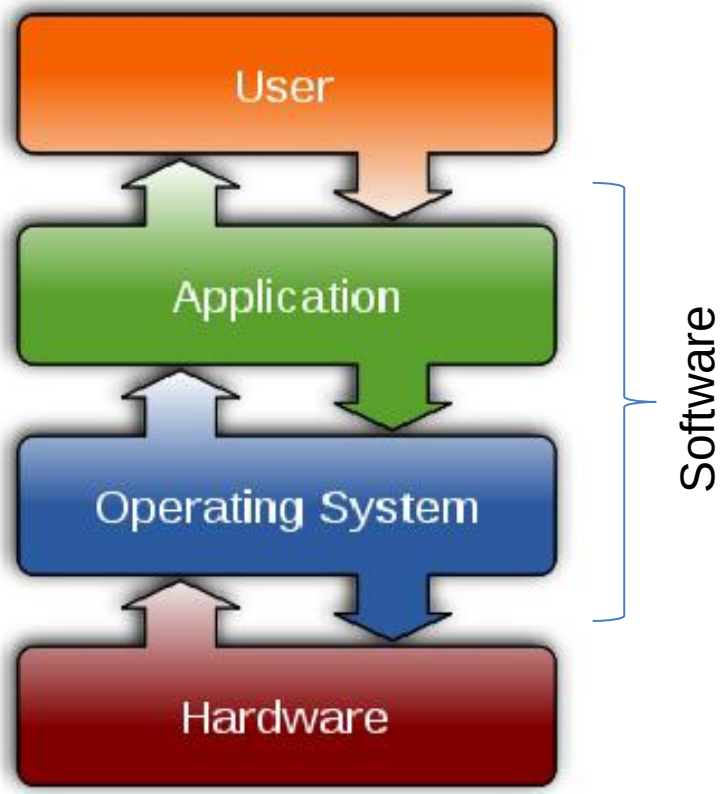
Linux™

Some Goals of Operating Systems

- Simplify the development of user programs and make solving user problems easier
  - Provide higher-than-hardware level programming abstractions
- Make application software portable and versatile
  - Write once, run everywhere (there is the same OS and ISA)
- Use computer hardware efficiently
  - Allow sharing of hardware and software resources
- Provide isolation, security, and protection among programs
  - What operations on resources a program can do?
- Improve overall system reliability
  - Error confinement, fault tolerance, reconfiguration

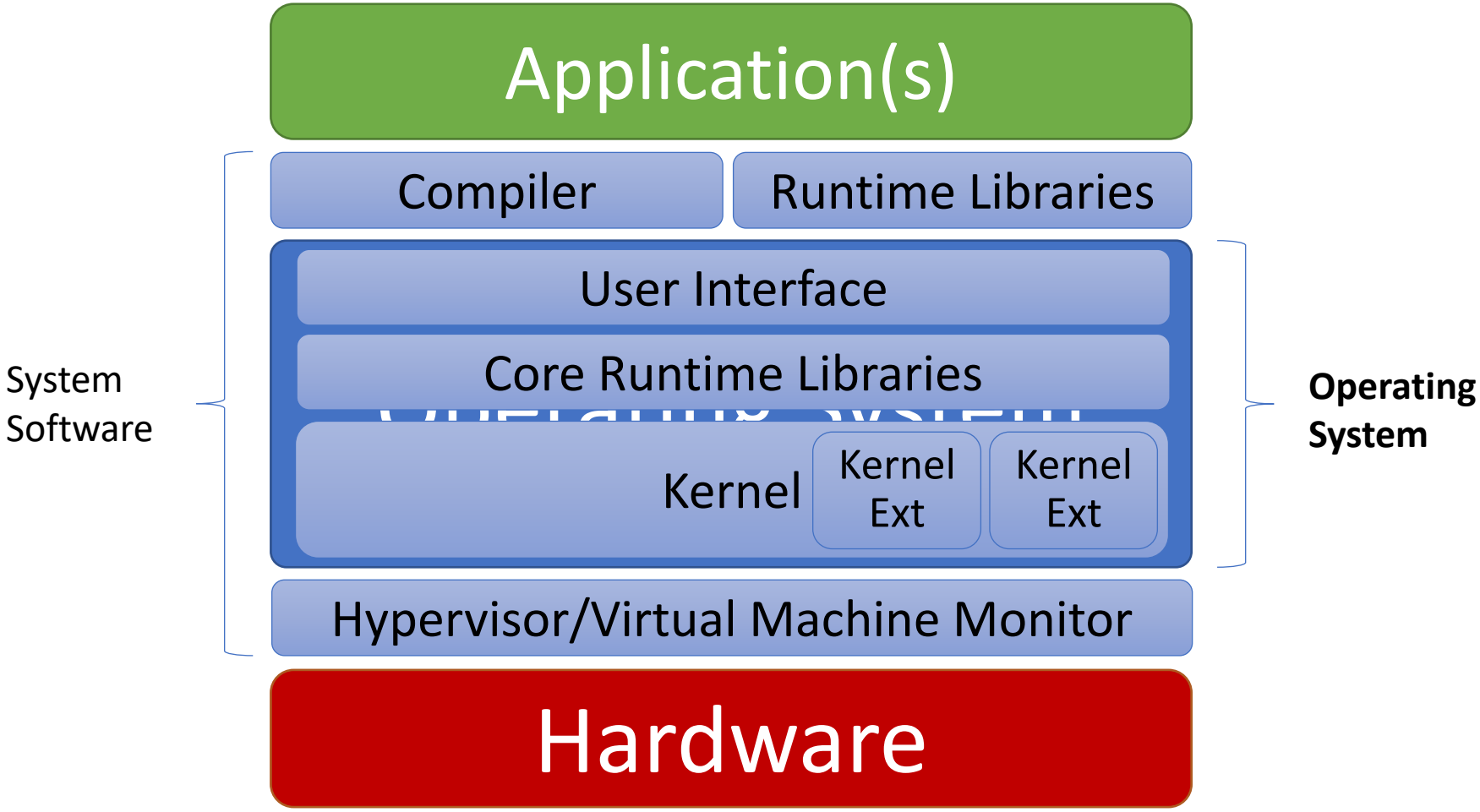
The Traditional Picture

- **“The OS is everything you don’t need to write in order to run your application”**
- Think OS as a library
  - In some ways, it is
    - all operations on I/O devices require OS calls (*syscalls)*
  - In other ways, it isn't
    - you use the CPU/memory without OS calls
    - it intervenes without having been explicitly called

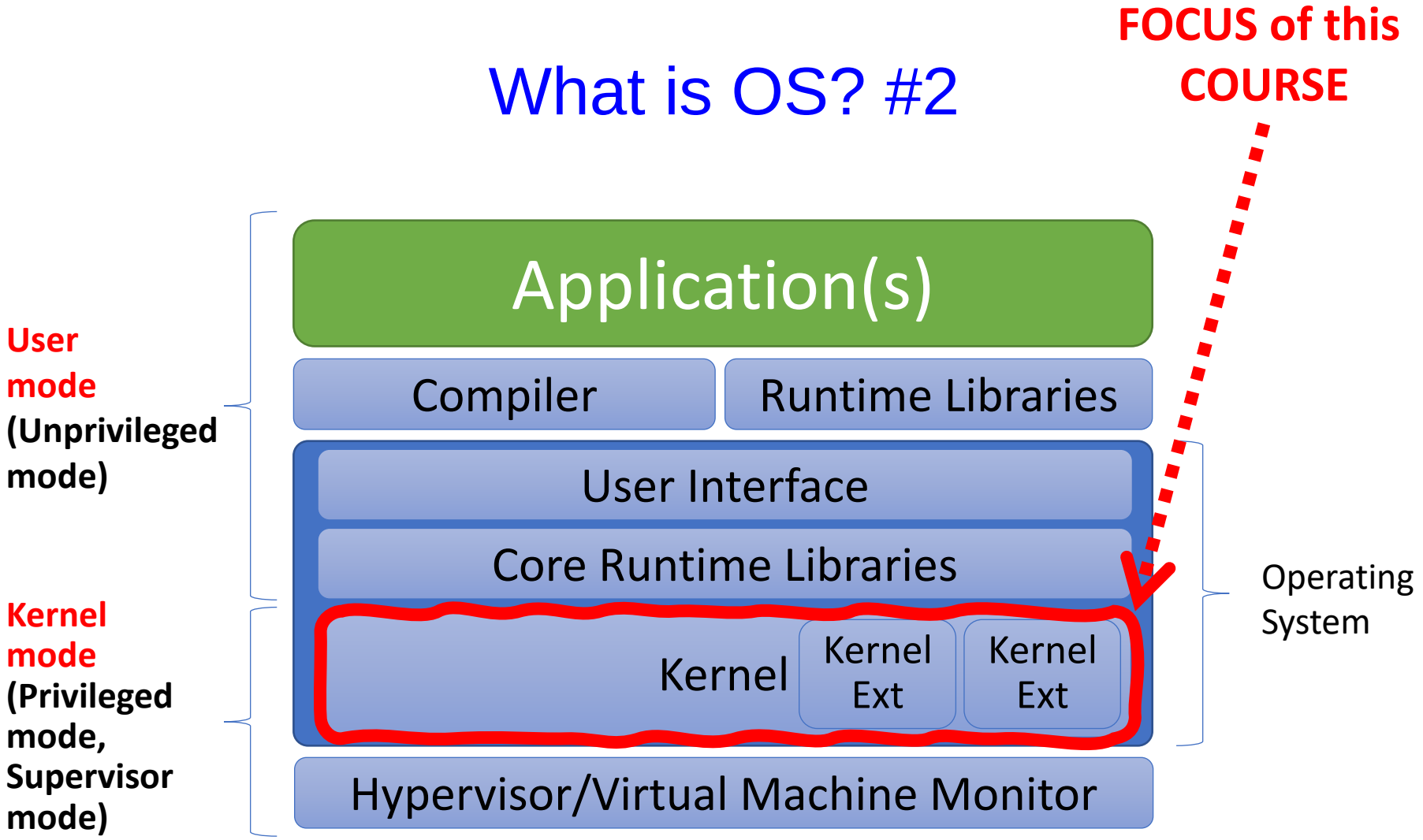


https://en.wikipedia.org/wiki/File:Operating_system_placement.svg

What is OS? #1


Application(s)
Compiler
Runtime Libraries
User Interface
Core Runtime Libraries
Kernel
Kernel Ext
Kernel Ext
Hypervisor/Virtual Machine Monitor
Hardware
System Software
Operating System

What is OS? #2



NOTE there exist OSes that do not use modes, there is hardware that doesn't support modes

The OS and Hardware

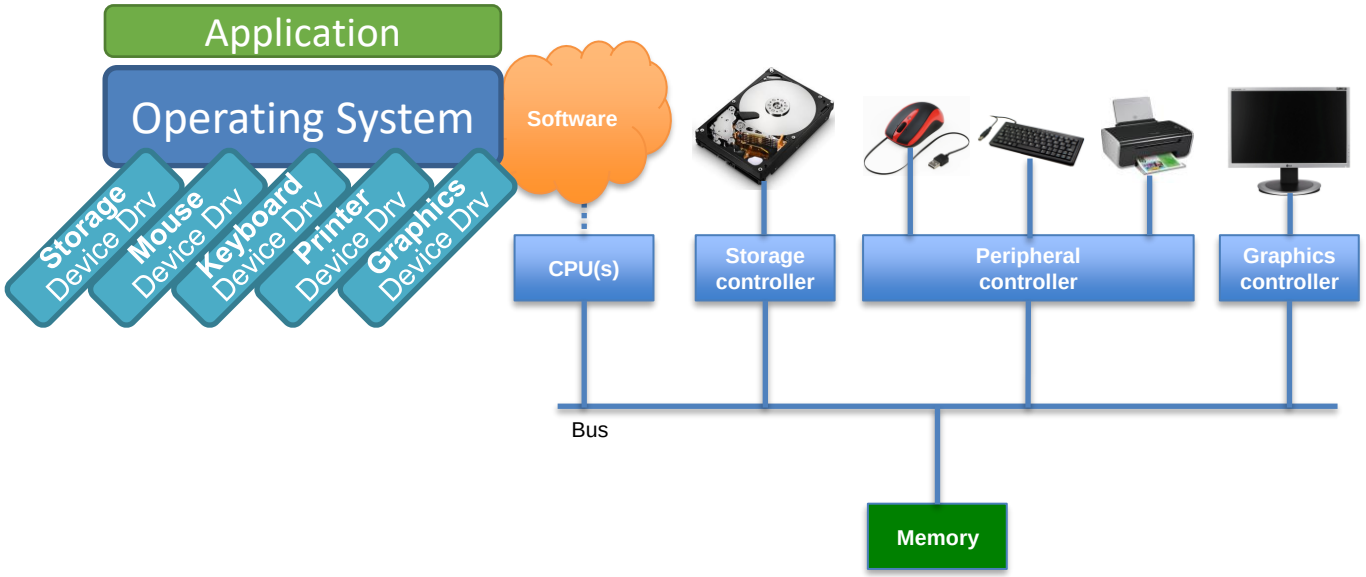
- An OS mediates programs' access to hardware resources (*sharing* and *protection*)
  - computation (CPU)
  - volatile storage (memory) and persistent storage (disk, etc.)
  - network communications (TCP/IP stacks, Ethernet cards, etc.)
  - input/output devices (keyboard, display, sound card, etc.)
- The OS abstracts hardware into logical resources and well-defined interfaces to those resources (*ease of use*)
  - processes (CPU)
  - address space (memory)
  - files (disk)
  - sockets (network)

Why bother with an OS?

- Application benefits
  - programming simplicity
    - see high-level abstractions (files) instead of low-level hardware details (device registers)
    - abstractions are reusable across many programs
  - portability (across machine configurations or architectures)
    - device independence: 3com card or Intel card?
- User benefits
  - safety
    - program “sees” its own (virtual) machine, thinks it “owns” the computer
    - OS protects programs from each other
    - OS fairly multiplexes resources across programs
  - efficiency (cost and speed)
    - share one computer across many users
    - concurrent execution of multiple programs

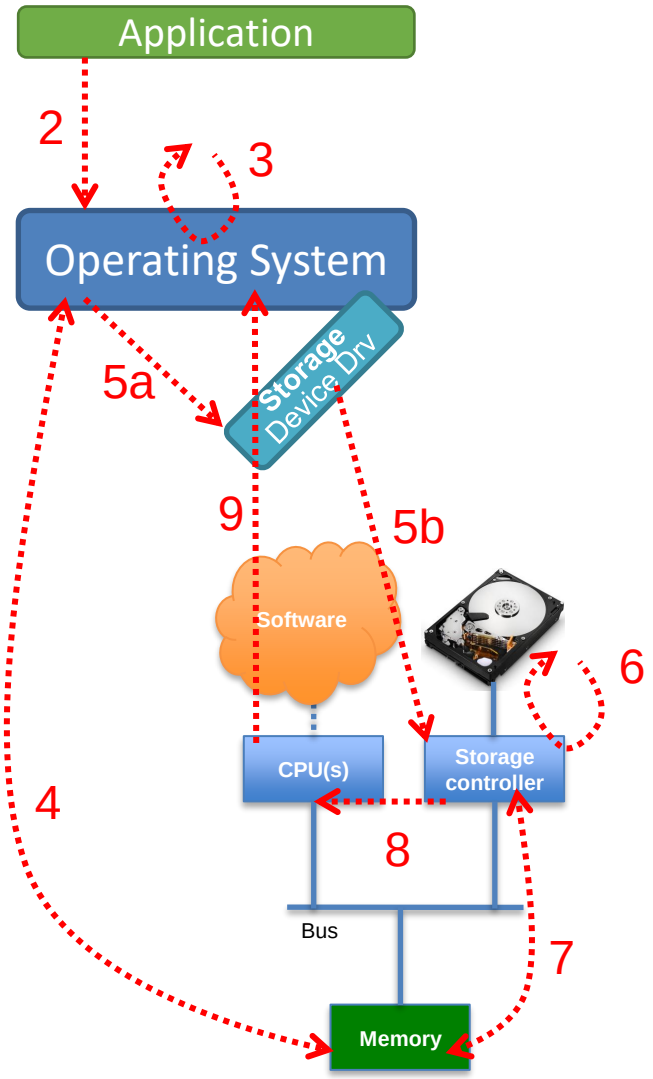
Hardware Recap: Devices

- To interact with the external world (e.g., with the user)
- Every device has a device controller, which
  - May move data to main memory, like the CPU(s)
  - Run in parallel to the CPU
  - Have buffers for data (thus, local memory)
- Operating Systems have device drivers per device controller

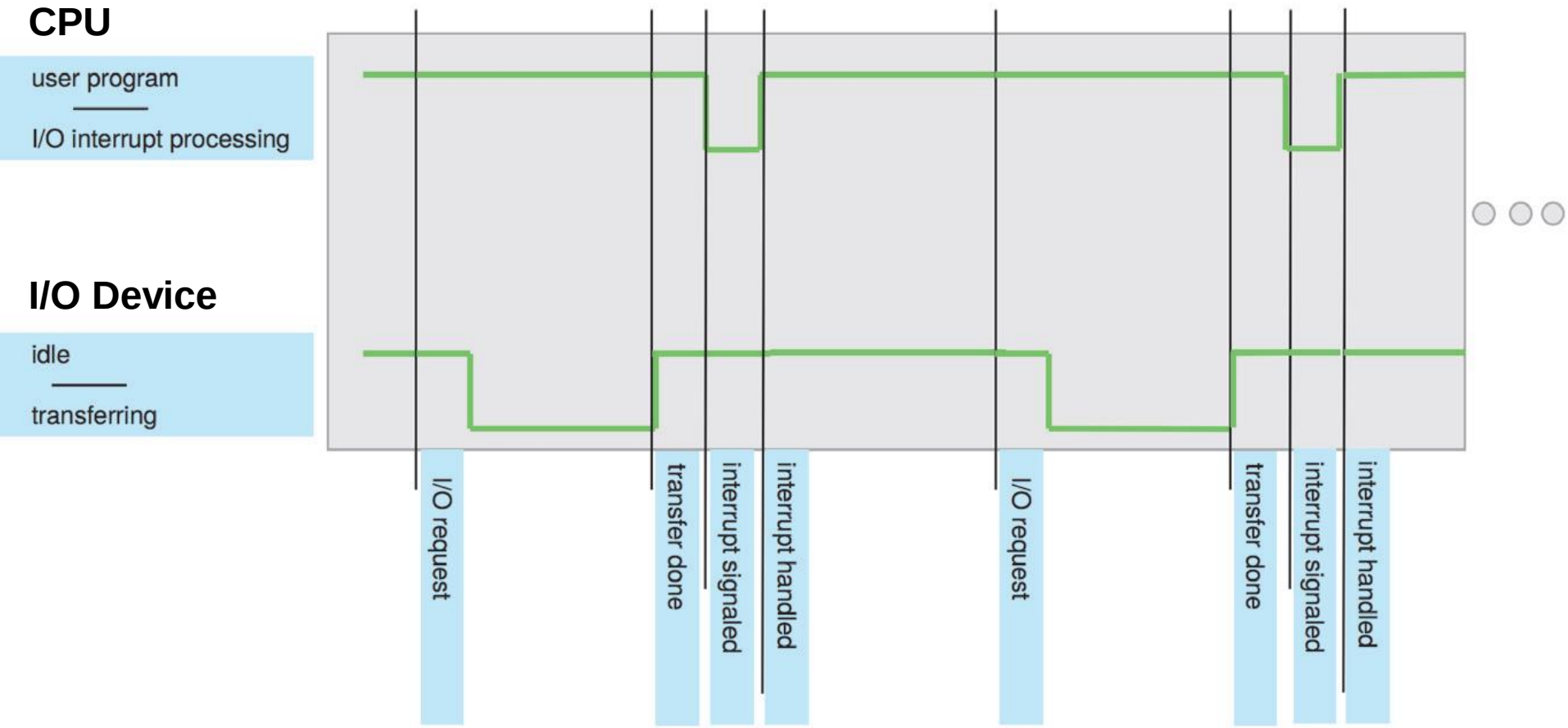


Hardware Recap: Devices – Disk Example

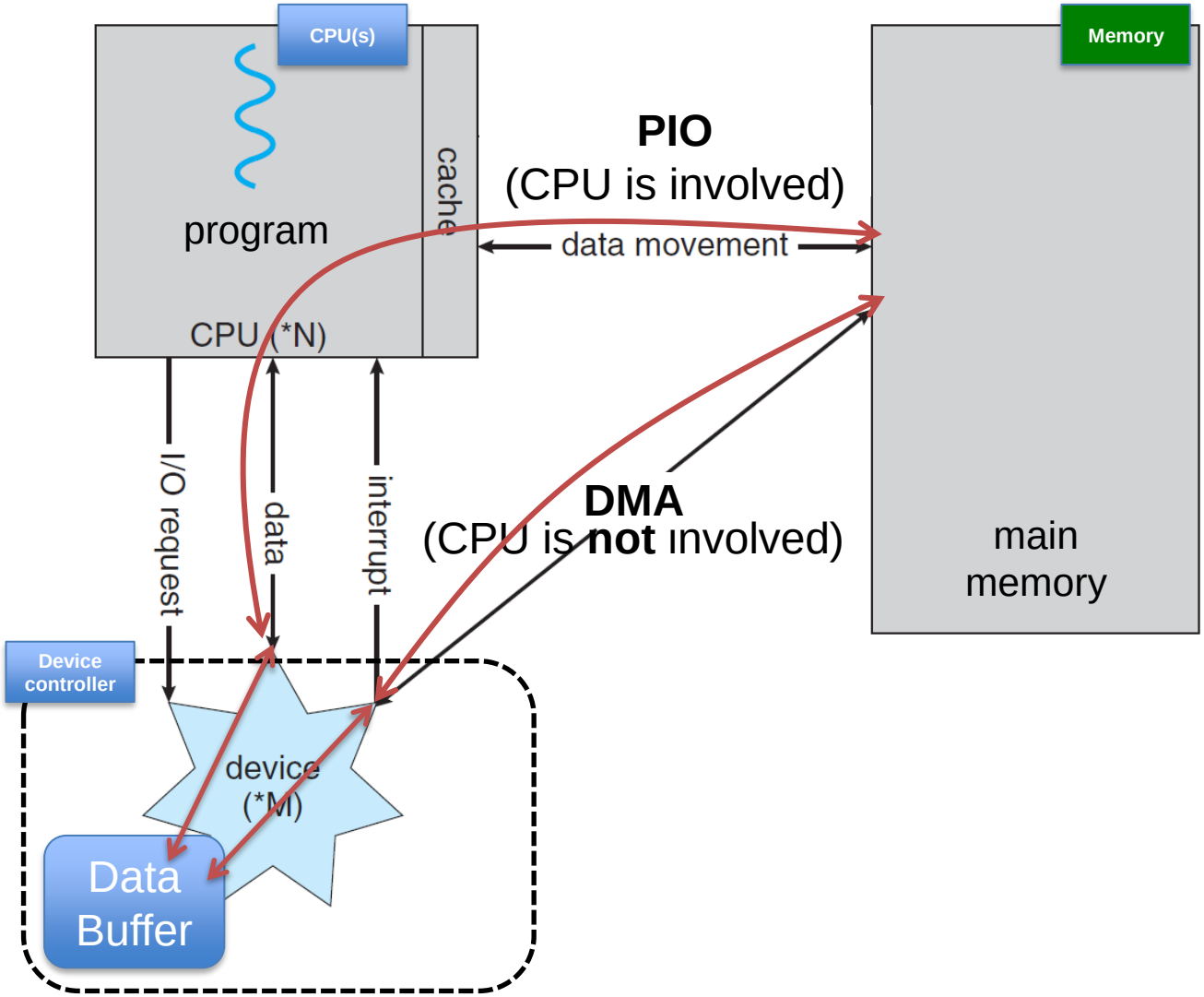
1. An application wants to read a file on disk
2. Application asks to read to the OS
3. OS converts the abstraction of file into a disk address
4. OS allocates memory space to receive the data
5. OS' disk device driver sends low-level command to disk
6. Disk reads data into internal buffers
7. Disk transfers data into memory space
8. When data transfer is finished, disk issues an interrupt
9. The OS receives the **interrupt** and runs the **interrupt handler**
10. …



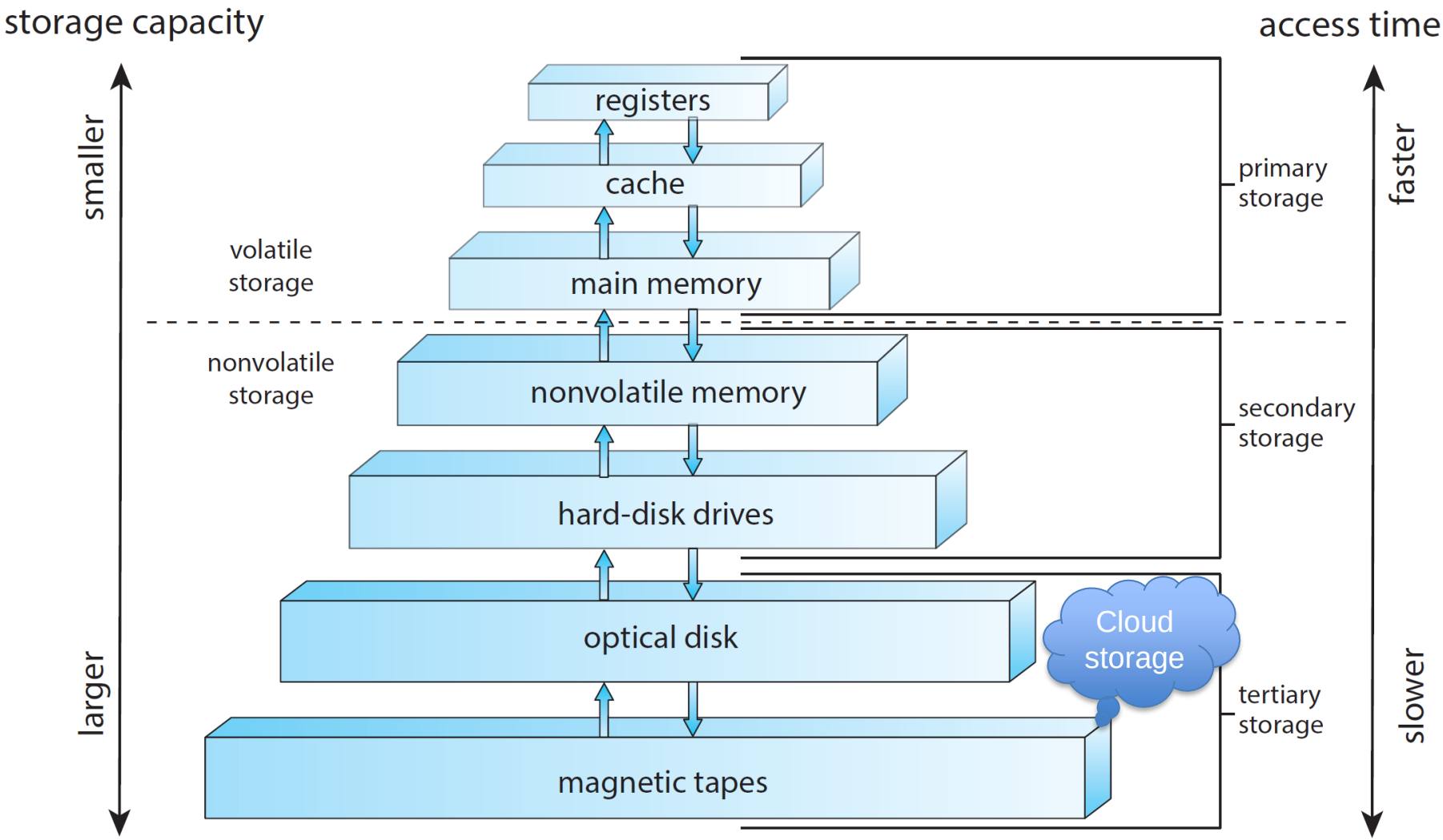
Hardware Recap: Interrupts



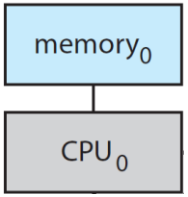
Hardware Recap: DMA



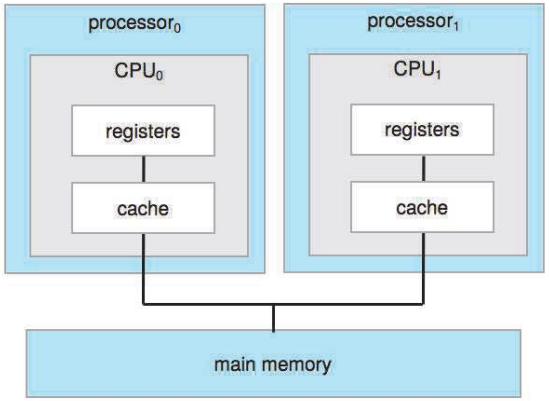
Hardware Recap: Storage Structure



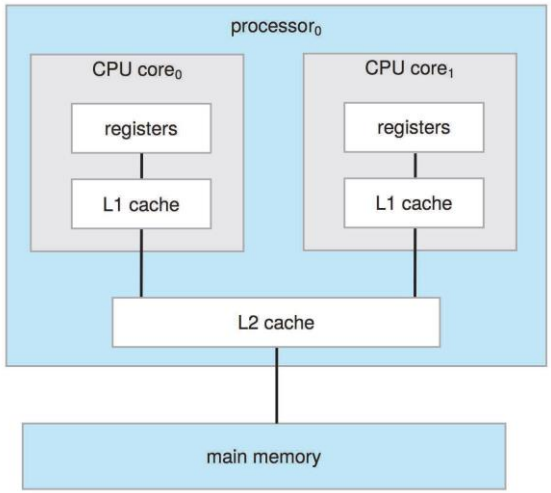
Hardware Recap: Memory and CPU



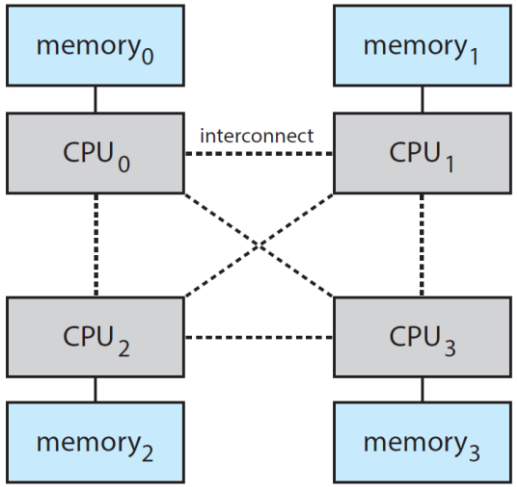
Single-core



Multiprocessor



Multicore



NUMA