CSC 360: Introduction to Operating Systems

Dr. Ahmad Abdullah
abdullah@uvic.ca

Department of Computer Science
University of Victoria
Spring 2023

Lectures: ECS 123, MWR 2:30 – 3:20 pm
Office: ECS 617, MR 11:30 – 1:00 pm, W 10:00 – 11:00 am

00- Intro

Ahmad Abdullah, PhD
abdullah@uvic.ca
https://web.uvic.ca/~abdullah/csc360

Lectures: MWR 2:30 – 3:20 pm
Location: ECS 123

Course Webpage

- https://bright.uvic.ca
- Updated periodically (check before class)
  - Announcements
  - Slides
  - Assignments
  - LAB materials
  - Other materials

Introduction to Operating Systems

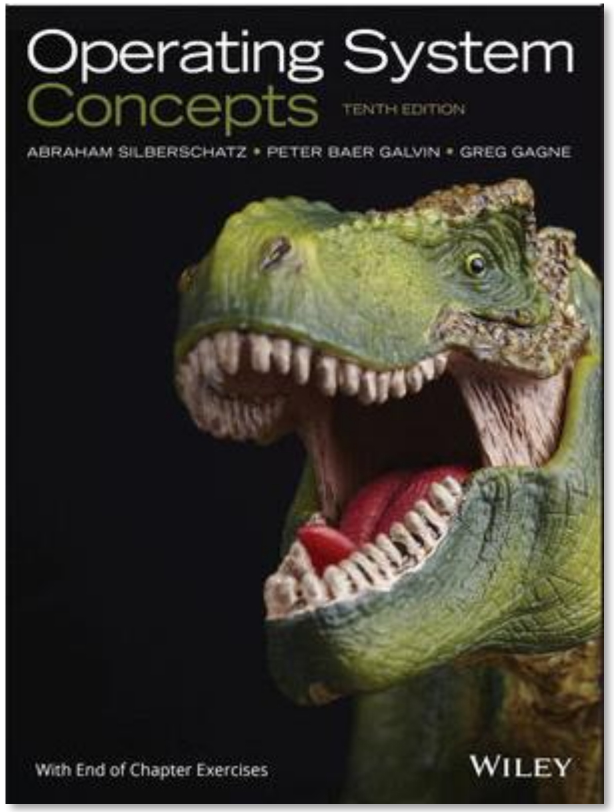
- This course aims to provide an introduction to major concepts of operating systems and study of the interrelationships between the operating system and the architecture of computer systems. Topics discussed include operating system structures, concurrent programming techniques, cpu scheduling, deadlocks, memory management, file systems and protection.

What this course covers?

- The purpose of an operating system
- Influence of computer-system organization and computer-system architecture
- Typical structures for an OS (including a kernel)
- Services and operations provided by an OS
- The process concept
- Support for concurrency (synchronization, communication)
- Memory management from the perspective of processes and OS services
- Management of different types of data storage
- Security

Textbook

- Title:
  - “Operating Systems Concepts”, 10th edition
- Authors:
  - Silberschatz, Galvin, and Gagne
- Publisher:
  - Wiley


Operating System Concepts
TENTH EDITION
ABRAHAM SILBERSCHATZ • PETER BAER GALVIN • GREG GAGNE
With End of Chapter Exercises
WILEY

Tutorials meant to help with assignments

- Tutorials will be used where necessary
  - This is unlike a lab-based course where there are labs every week
- Tutorials will be announced the week before they are to take place
  - Please attend your tutorial section for that week
  - Working plan right now is for the first tutorial to take place the week of Jan 23.

Evaluation

- Official course outline:
  - https://heat.csc.uvic.ca/coview/course/2023011/CSC360
  - Link to outline also available via Brightspace site
- Three programming assignments in C (15% each)
- Midterm exams (Thursday, February 16th; 20%)
- Final exam (scheduled by UVic admin, 35%)
- Please read the outline ASAP!
- Link to outline is provided at the Brightspace site for our course

Evaluation

Assignment/Exam	Weight	Assigned Date	Due Date
Assignment 1	15%	26-Jan	09-Feb
Midterm 1	20%	16-Feb	-
Assignment 2	15%	02-Mar	16-Mar
Assignment 3	15%	23-Mar	06-Apr
Final Exam	35%	TBD	-

SPRING 2023 Calendar

Week	Monday	Tuesday	Wednesday	Thursday	Friday
1	09-Jan First day of classes	10-Jan	11-Jan	12-Jan	13-Jan
2	16-Jan	17-Jan	18-Jan	19-Jan	20-Jan
3	23-Jan	24-Jan	25-Jan	26-Jan Assignment #1	27-Jan
4	30-Jan	31-Jan	01-Feb	02-Feb	03-Feb
5	06-Feb	07-Feb	08-Feb	09-Feb Assignment #1 due	10-Feb
6	13-Feb	14-Feb	15-Feb	16-Feb Midterm #1 [15%]	17-Feb
7	20-Feb	21-Feb	22-Feb	23-Feb	24-Feb
	Family Day & Reading Break				
8	27-Feb	28-Feb	01-Mar	02-Mar Assignment #2	03-Mar
9	06-Mar	07-Mar	08-Mar	09-Mar	10-Mar
10	13-Mar	14-Mar	15-Mar	16-Mar Assignment #2 due	17-Mar
11	20-Mar	21-Mar	22-Mar	23-Mar Assignment #3	24-Mar
12	27-Mar	28-Mar	29-Mar	30-Mar	31-Mar
13	03-Apr	04-Apr	05-Apr	06-Apr Assignment #3 due	07-Apr Last day of classes

Credits

- Special thanks to Professors Kui Wu and Mike Zastre for letting me use their course material.

Thank you!

Class Interaction

- We will be using www.menti.com a little bit during lectures this semester
  - Informal polls
  - Non-graded quizzes and questions
  - Anonymous
  - None of this on menti.comis for course credit!
- Link and code to Mentimeter will be shown when there is something to do.
- You do not need an account with Mentimeter to complete any of this.
  - Ignore everything at www.menti.comasking you to sign up for an account!

What is an Operating System?

https://www.menti.com/alwt1mf5dcq9

What is an Operating System?

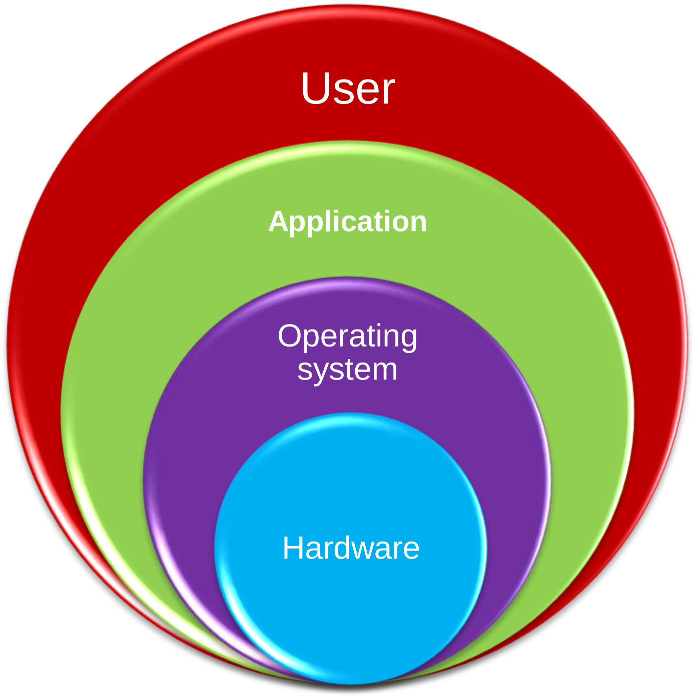
`“... A program that behaves as an intermediary between a user of a computer and the computer hardware”`

- Main goals for an OS:
  - Execute user tasks
  - Reduce the effort needed for a user to solve their problems with the computer system
  - Ensure the computer-system hardware and resources are utilized as “efficiently” as possible

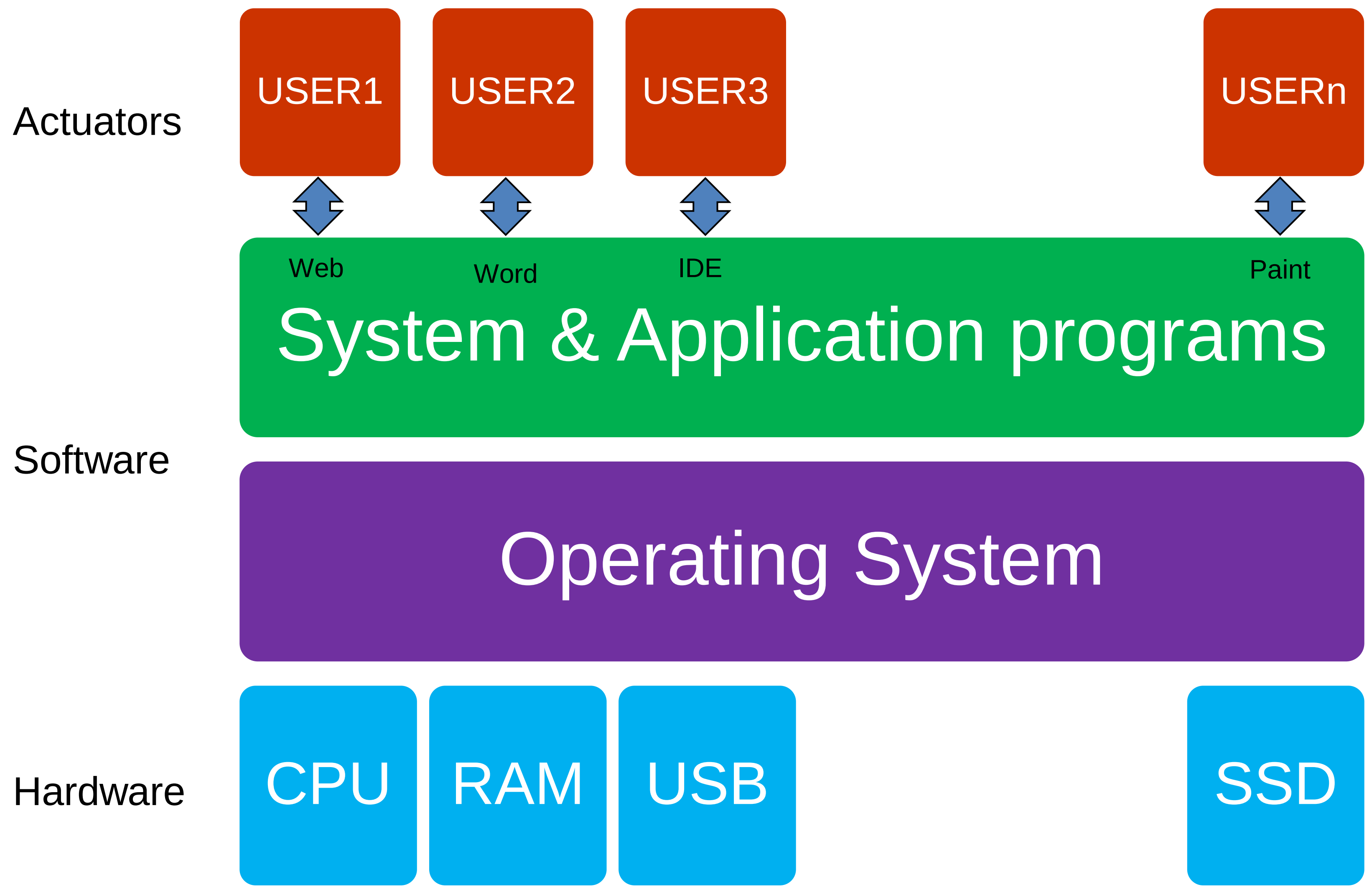
Notion of a “user” is very flexible here.

What is an Operating System?

“... A program that behaves as an intermediary between a user of a computer and the computer hardware”



What is an Operating System?



Users can be human, or they can be other running tasks/processes, or they can be other machines

System and application programs determine precise ways in which system resources are used to solve the computing problems of users

OS controls and coordinates the use of hardware among various applications and users

Hardware provides basic/raw resources to support computation

But what do operating systems do?

- This all depends upon the given point of view
- **Single regular human user using a personal computer:**
  - Wants ease of use and convenience
  - Wants good performance
  - Not necessarily concerned about resource allocation as all resources are used exclusively by the user
- **Users sharing resources in a data centre**
  - OS must somehow ensure service requirements of all users are met at all times
  - Some users pay for stricter requirements (i.e., lower latency) than others
- **Handheld computers (e.g., smartphones, tablets)**
  - Relative to desktop or laptops, somewhat resource poor (memory, CPU)
  - Devices optimized for power usage (i.e., maximize battery life)
  - Also optimized for usability (i.e., low latency when responding to touch-screen interface)

But what do operating systems do?

- **Embedded systems**
  - These often have little or no user interface
  - Can be found within industrial devices, consumer products
  - Also can be found in automobiles
  - (and much, much more)
  - System range from having no stringent requirements at all (i.e., minimal or no OS in a toy) to quite strict requirements (i.e., hard real-time OS)
  - This is a surprisingly complex area for OS design (i.e., combination of applications + RTOS requirements + target hardware = combinatorial explosion of configurations that must work correctly)

An operating system is...

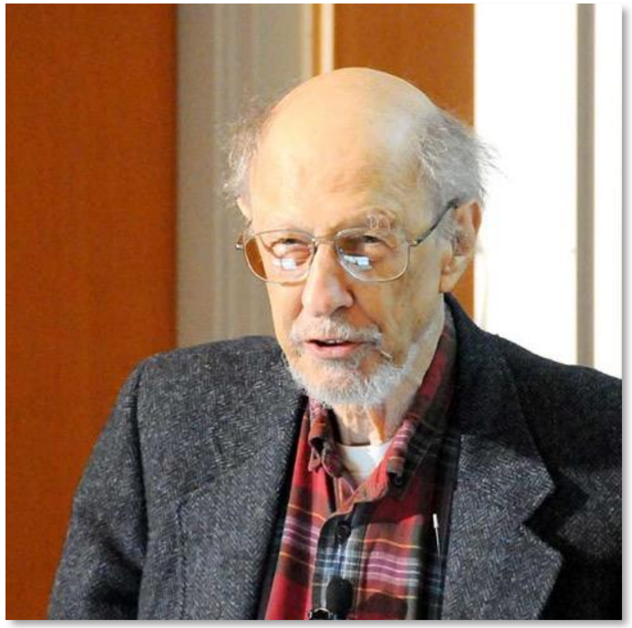
- **... a resource allocator:**
  - Manages nearly all computational, memory, storage devices
  - Arbitrates between conflicting requests for resources
  - Must somehow ensure efficient and fair use of resources
- **... a control program:**
  - Starts and stops user programs
  - Ensures errors in execution do not result in crashing system
  - Must also detect and prevent improper use of resources (i.e., access control for filesystem items)

An operating system is...

- **... surprisingly hard to define!**
  - No definition is accepted universally
- **Some have (somewhat ironically) defined it as “Everything a vendor ships when you order an OS.”**
- **Others have (also somewhat ironically) defined it as “The one program that runs at all times on the computer” (i.e., the kernel)**
- **Everything else is either:**
  - A system program(such as a compiler, assembler, etc.) that ships with the OS, or...
  - ... is an application program.

A bit of history...

Fernando Corbato (1926–2019)

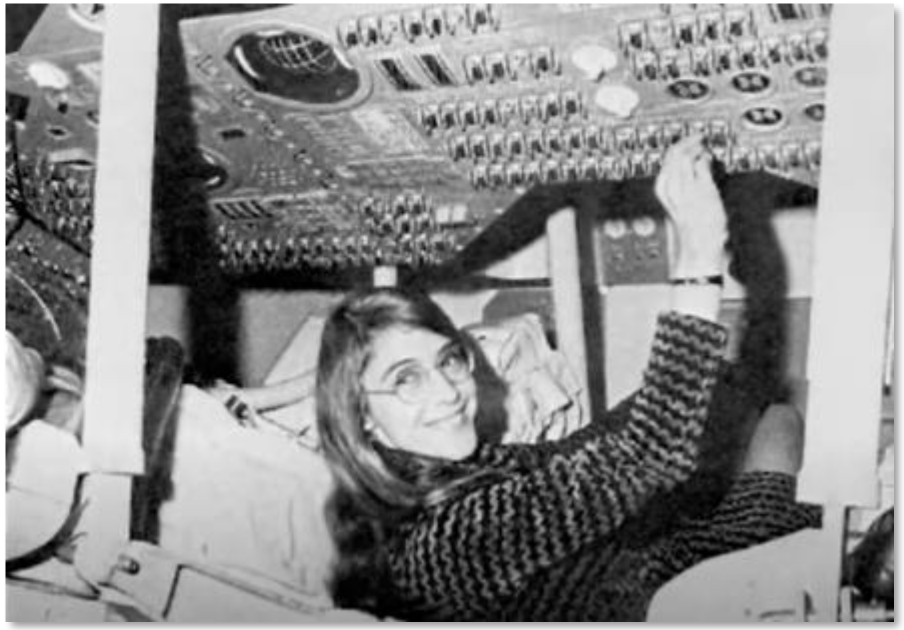


Won the ACM Turing Award in 1990 for his work on operating systems (Multics, CTSS, etc.)

https://youtu.be/Q07PhW5sCEk

A bit of history...

Margaret Hamilton (b. 1936)



https://youtu.be/4sKY6_nBLG0

Member of team the wrote the on-board flight software from the NASA Apollo program. She is also one of the creators of the term “software engineering”.

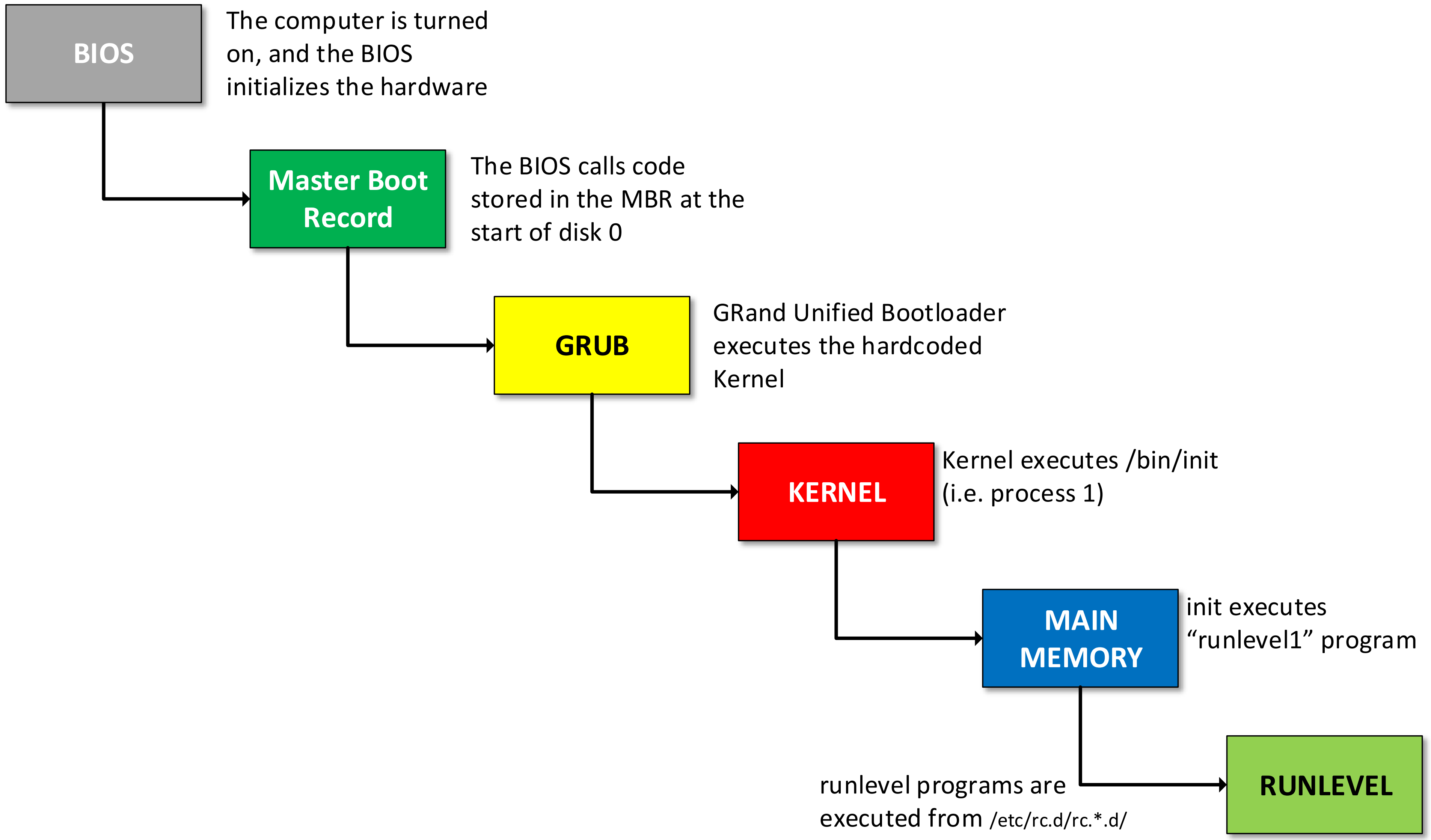
Some aspects of OS operation

- Given that a top-down definition is out of reach...
- **... we will instead approach our study of operating systems by looking at them from different perspectives.**
- Some initial perspectives (i.e., not an exhaustive list):
  - What happens at startup?
  - What are the consequences of physical concurrency?
  - What are the consequences of different data storage devices (speeds, capacities)?
  - What are some typical OS structures?
  - What are foundational OS abstractions?

What happens at startup?

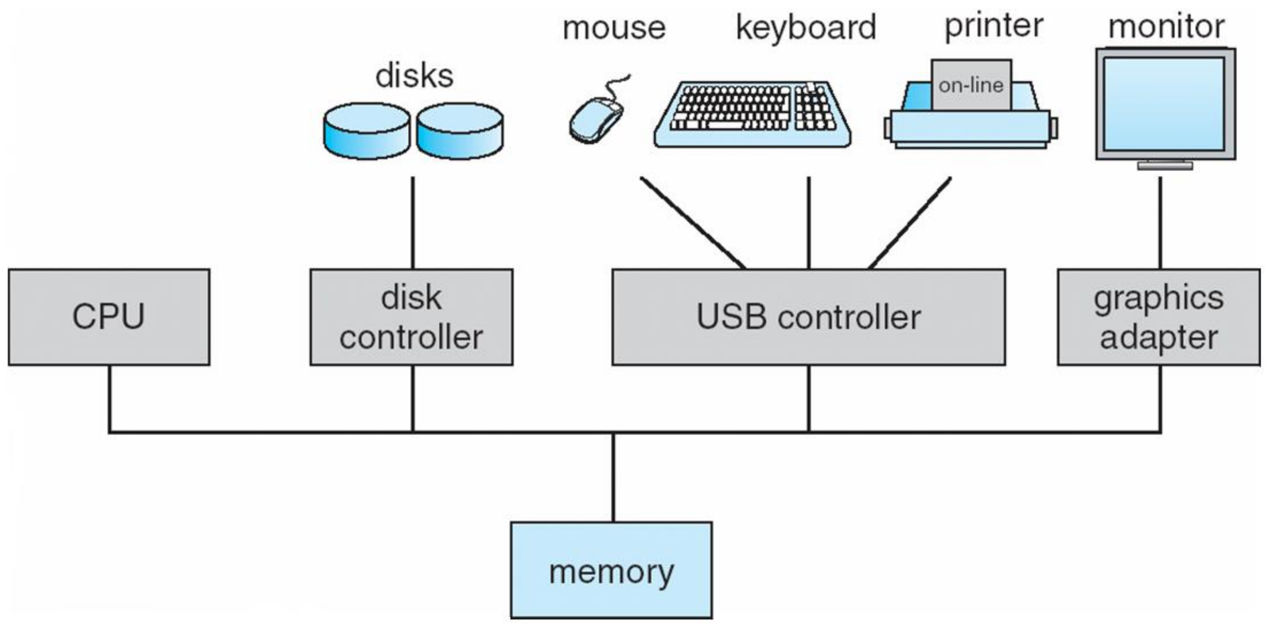
- **Bootstrap program**
  - Minimal program loaded into computer at power-up or reboot
  - Normally stored in ROM or EPROM (aka firmware)
- These programs load in progressively more powerful programs
  - Initializes all aspects of the computer system
  - Final step is to load OS kernel and begin its execution (i.e., in Unix, start process 0, which forks itself to create in process 1, such that this latter process is the ancestor of all other processes).
  - Note: We will go into gruesome detail this semester on the concept of a **process**

What happens at startup?


BIOS
The computer is turned on, and the BIOS initializes the hardware
Master Boot Record
The BIOS calls code stored in the MBR at the start of disk 0
GRUB
GRand Unified Bootloader executes the hardcoded Kernel
KERNEL
Kernel executes /bin/init (i.e. process 1)
MAIN MEMORY
init executes “runlevel1” program
RUNLEVEL
runlevel programs are executed from /etc/rc.d/rc.*.d/

Computer System Organization

- One or more CPUs, device controllers connect through a common bus
- Access a shared memory
- Physically concurrent CPUs and devices compete for memory cycles



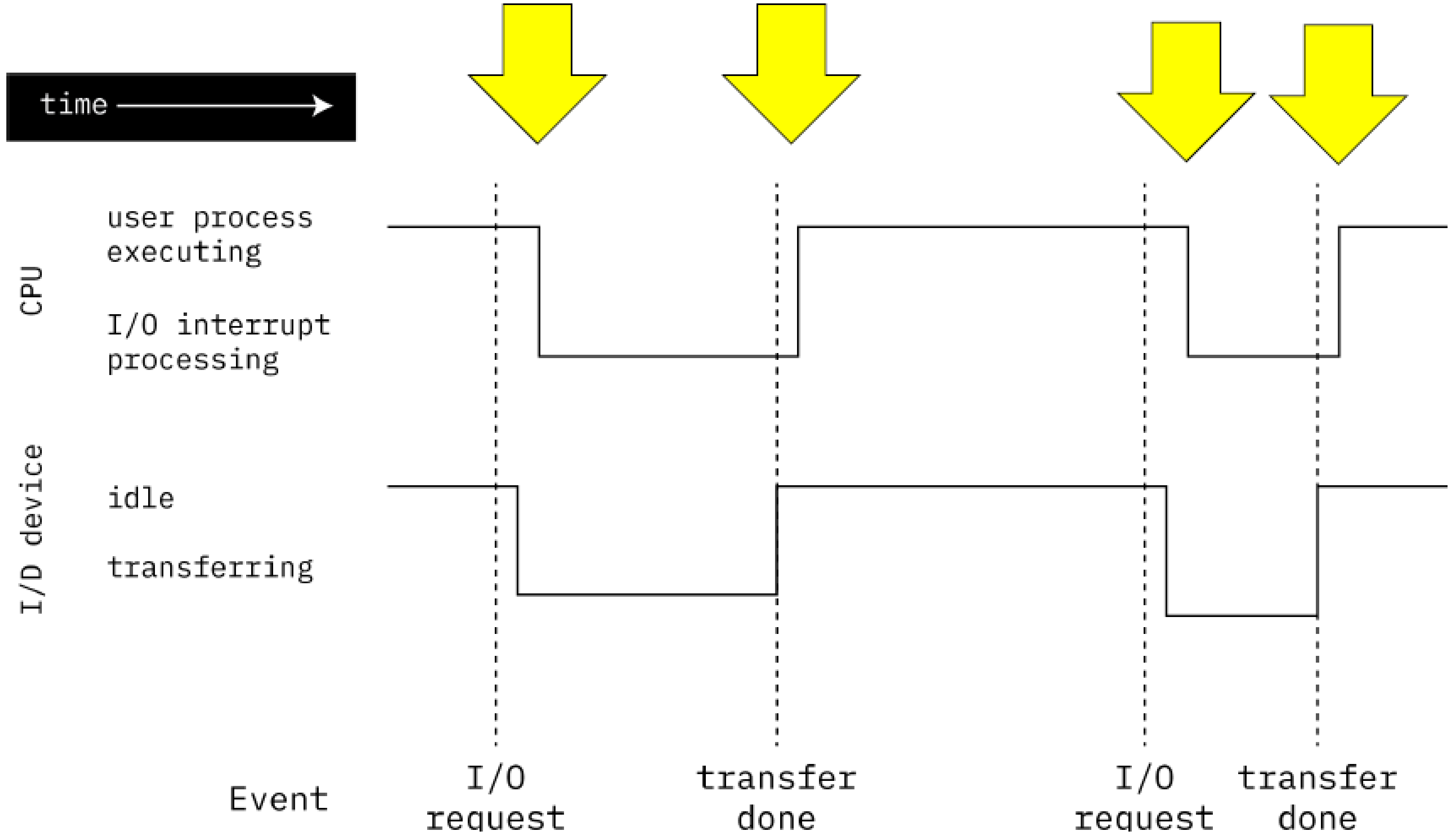
Computer-System Operation

- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller
- Device controller informs CPU that it has finished its operation by causing an **interrupt**

Common functions of interrupts

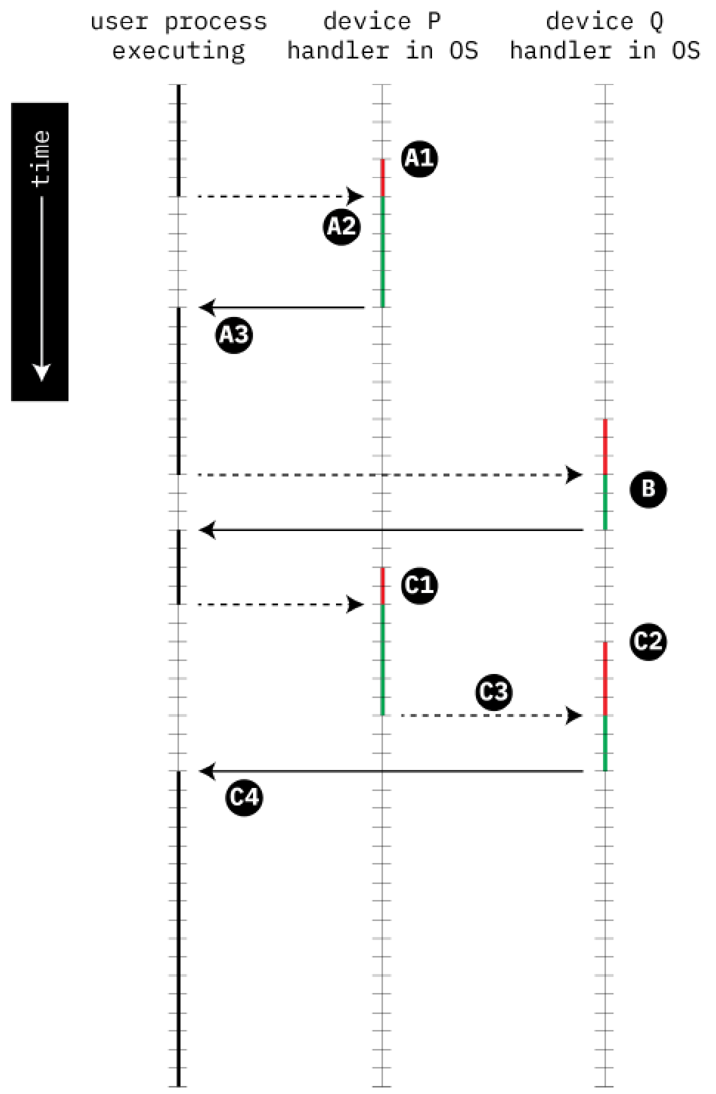
- **An interrupt event** causes control to be transferred to the **interrupt service routine** (aka **interrupt handler**)
  - This is normally found through the **interrupt vector**
  - The vector contains the mapping / addresses of all hardware interrupt code in the OS
- Interrupt hardware must also cause **address of interrupted instruction** to be saved
  - If interrupt priorities are possible, then interrupted instruction might be within a handler!
- A **trap** or **exception** is a software-generated interrupt
  - Caused by an error or a user request
- **Bottom line: At its core, an operating system is *interrupt driven***

Interrupt Timeline



User program suspended while I/O in progress

Interrupt Timeline



A:
Device P generates an event, it is handled after a short delay

B:
Device Q generates an event, it is handled after a short delay

C:
Device P generates an event, while device Q generates an event when P's handler is executing.

Notice that at any one point in time, only a thick black line or thick green line is active.

Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter
- Determines which type of interrupt has occurred:
  - **polling**
  - **vectored** interrupt system
- Separate segments of code determine what action should be taken for each type of interrupt

Interrupt Handling

- Before handling an interrupt, operating **system preserves the state of the CPU**
  - Values in CPU registers (general purpose, status, etc.)
  - Value of program counter
- The OS must determine precisely what interrupt has occurred
  - Approach 1: **poll** / ask the interrupt controller for device information
  - Approach 2: device is associated with a particular spot in the **interrupt vector**, therefore the fact of the interrupt directly indicates starting address of handler’s code.
  - The approach used depends upon OS, hardware architecture, device type,
- Code in the handler for the interrupt determine actions to take
- After handler’s work is completed, **CPU state is restored**

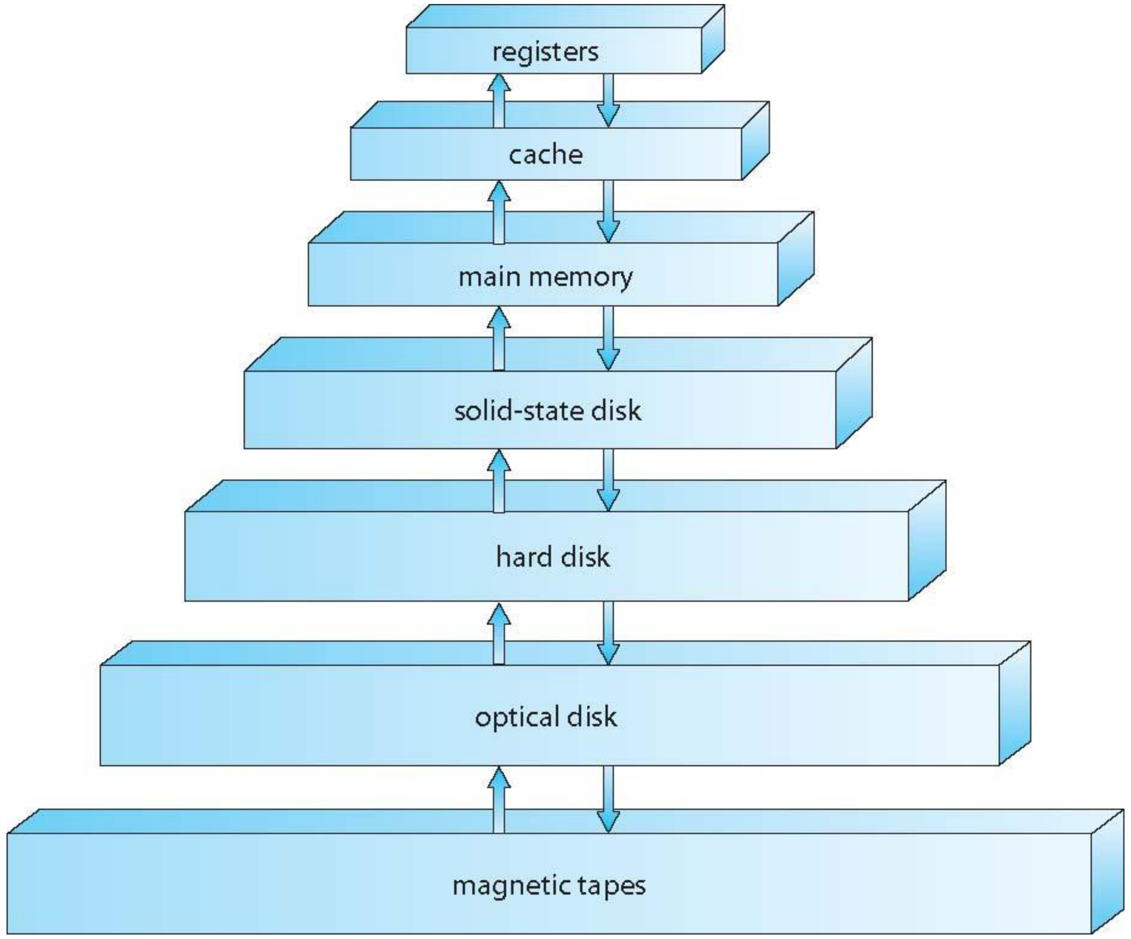
Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
- Hard disks – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
  - The **disk controller** determines the logical interaction between the device and the computer
- **Solid-state disks** – faster than hard disks, nonvolatile
  - Various technologies
  - Becoming more popular

Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- **Caching** – copying information into faster storage system; main memory can be viewed as a cache for secondary storage
- **Device Driver** for each device controller to manage I/O
  - Provides uniform interface between controller and kernel

Storage-Device Hierarchy



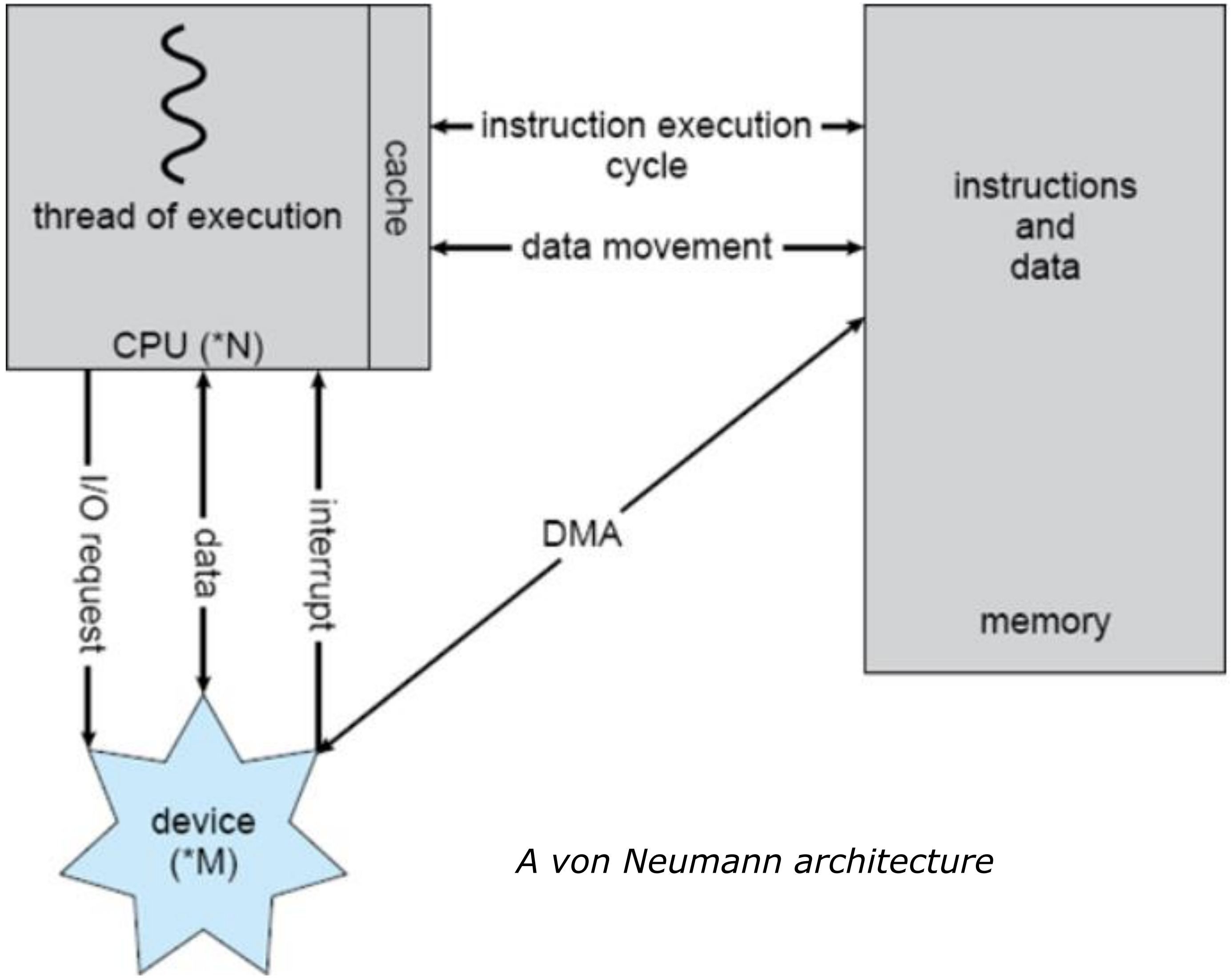
Caching

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache smaller than storage being cached
  - Cache management is an important design problem
  - Cache size and replacement policy

Direct Memory Access Structure

- Used for high-speed I/O devices that are able to transmit information at close to memory speeds
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only one interrupt is generated per block, rather than an interrupt per each byte

How a Modern Computer Works



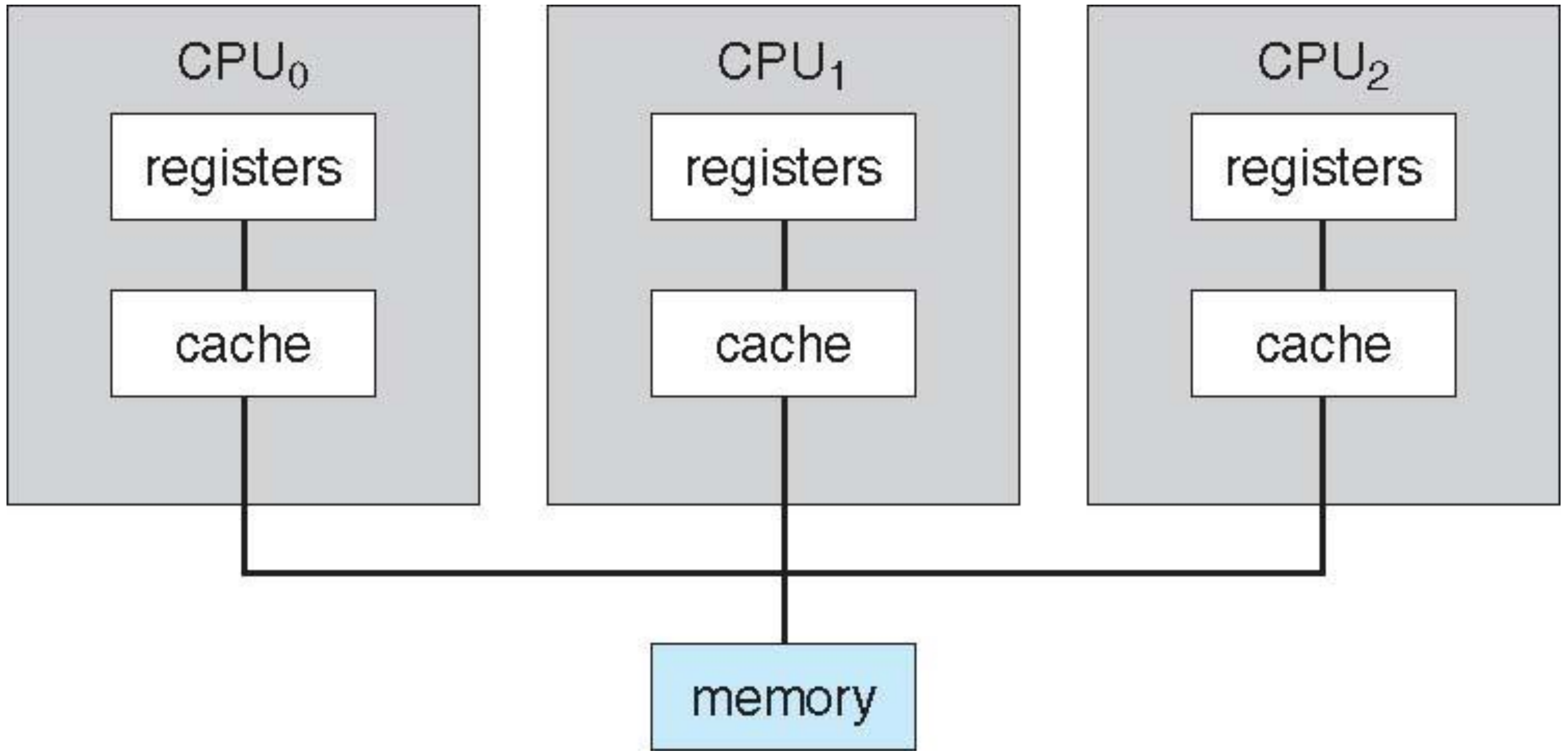
A von Neumann architecture

I/O only involves CPU when needed to transfer to/from device/main memory

Computer-System Architecture

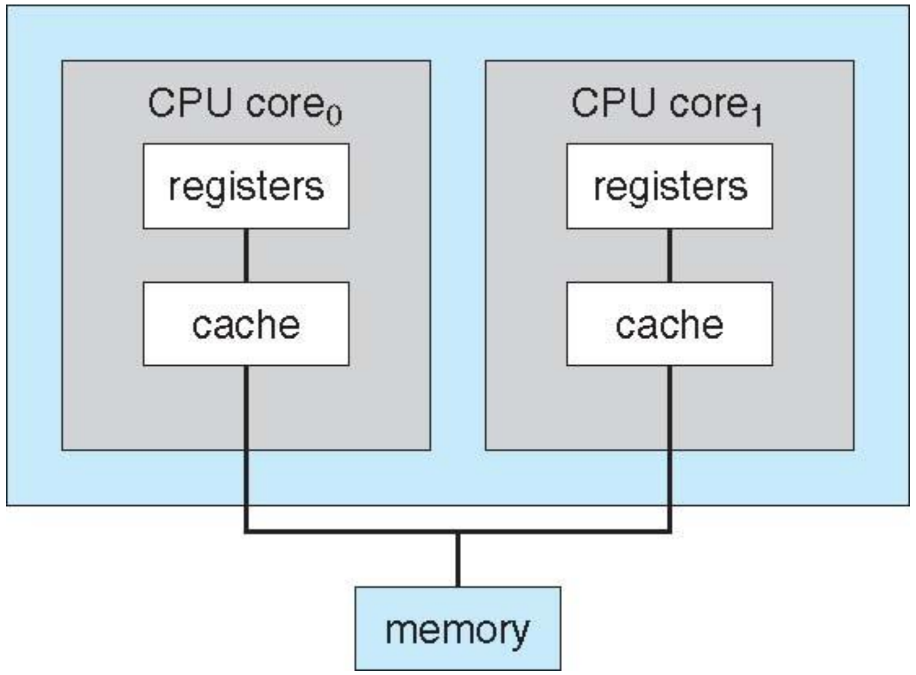
- Most systems use a single general-purpose processor
  - Most systems have special-purpose processors as well
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems, tightly-coupled systems**
  - Advantages include:
    1. **Increased throughput**
    2. **Economy of scale**
    3. **Increased reliability** – graceful degradation or fault tolerance
  - Two types:
    1. **Asymmetric Multiprocessing** – each processor is assigned a specie task.
    2. **Symmetric Multiprocessing** – each processor performs all tasks

Symmetric Multiprocessing Architecture



A Dual-Core Design

- Multi-chip and **multicore**
- Systems containing all chips
  - Chassis containing multiple separate systems



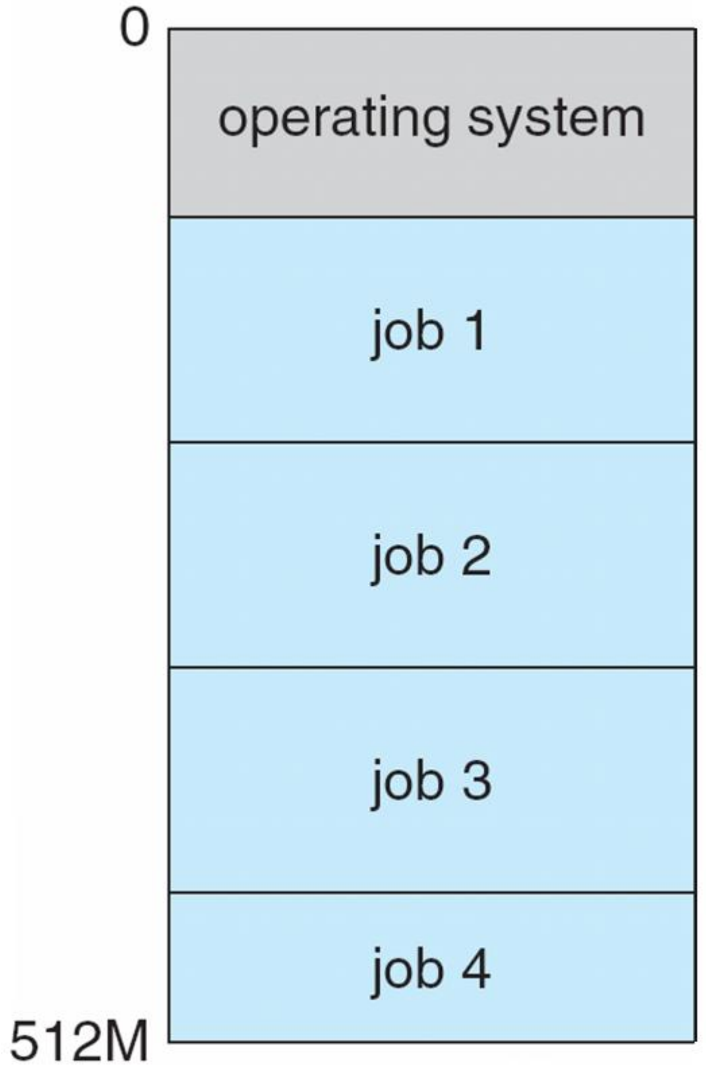
Operating System Structure

- **Multiprogramming (Batch system)** needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via **job scheduling**
  - When it has to wait (for I/O for example), OS switches to another job

Operating System Structure

- **Timesharing (multitasking)** is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
  - **Response time** should be < 1 second
  - Each user has at least one program executing in memory ⇨ **process**
  - If several jobs ready to run at the same time ⇨ **CPU scheduling**
  - If processes don’t fit in memory, **swapping** moves them in and out to run
  - **Virtual memory** allows execution of processes not completely in memory

Memory Layout for Multiprogrammed System



Operating-System Operations

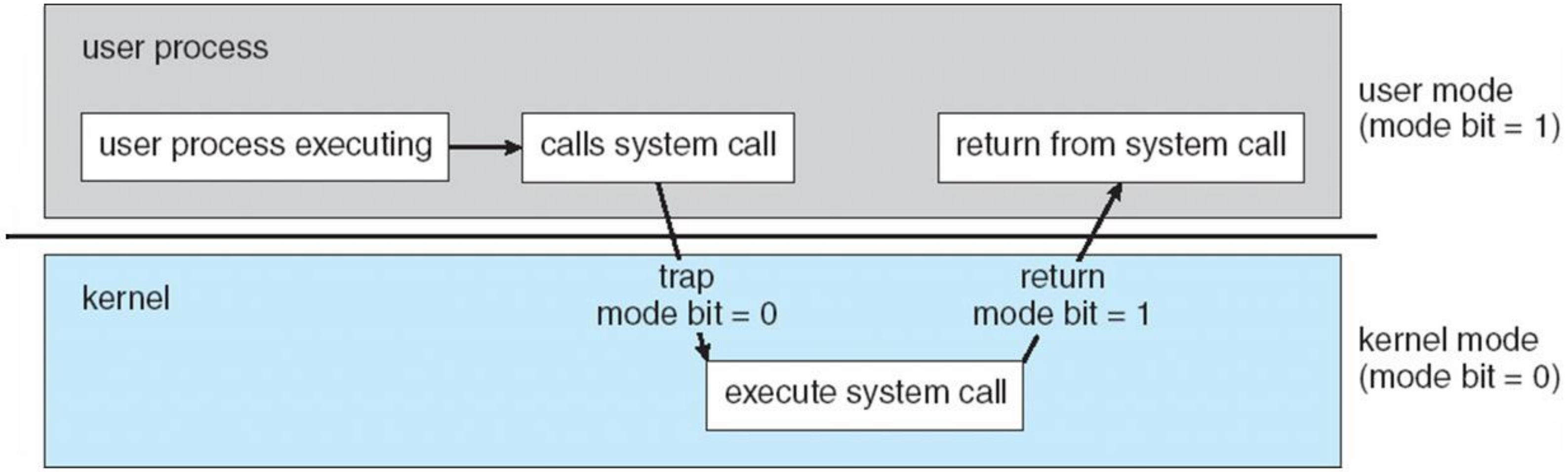
- **Interrupt driven** (hardware and software)
  - Hardware interrupt by one of the devices
  - Software interrupt (**exception** or **trap**):
    - Software error (e.g., division by zero)
    - Request for operating system service
    - Other process problems include infinite loop, processes modifying each other or the operating system

Operating-System Operations (cont.)

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user
- Increasingly CPUs support multi-mode operations
  - i.e. **virtual machine manager (VMM)** mode for guest **VMs**

Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
  - Timer is set to interrupt the computer after some time period
  - Keep a counter that is decremented by the physical clock.
  - Operating system set the counter (privileged instruction)
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time



Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a ***passive entity***, process is an ***active entity***.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one **program counter** specifying location of next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
  - Concurrency by multiplexing the CPUs among the processes / threads

Process Management Activities

- The operating system is responsible for the following activities in connection with process management:
  - Creating and deleting both user and system processes
  - Suspending and resuming processes
  - Providing mechanisms for process synchronization
  - Providing mechanisms for process communication
  - Providing mechanisms for deadlock handling

Memory Management

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory.
- Memory management determines what is in memory and when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

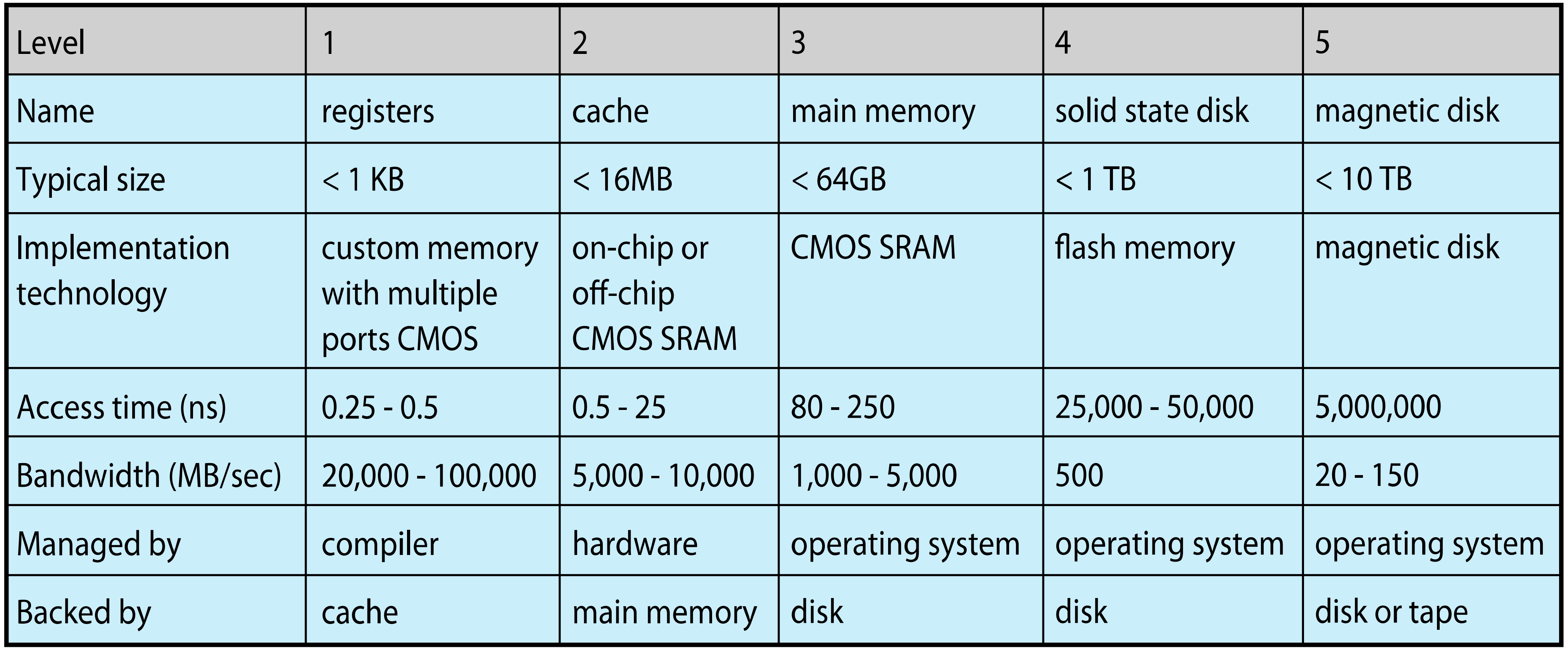
Storage Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - **file**
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and directories
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

Mass-Storage Management

- Usually, disks are used to store data that does not fit in main memory or data that must be kept for a “long” period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed – by OS or applications
  - Varies between WORM (write-once, read-many-times) and RW (read-write)

Performance of Various Levels of Storage

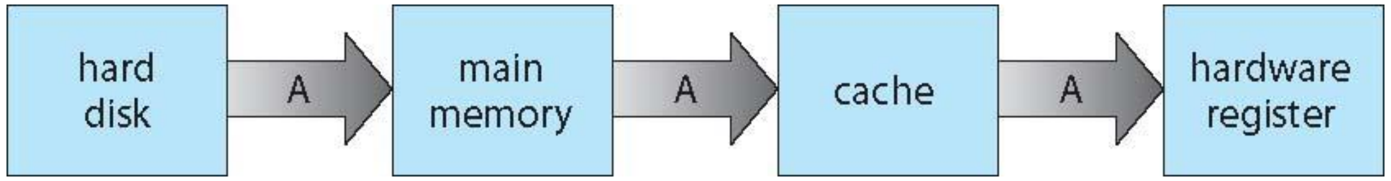


Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Movement between levels of storage hierarchy can be explicit or implicit

Migration of data “A” from Disk to Register

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy



- Multiprocessor environment must provide cache coherency in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
  - Several copies of a datum can exist (i.e., on physically different computers)
  - Later in the course we may have an opportunity to look at strategies for dealing with this.

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
  - General device-driver interface
  - Drivers for specific hardware devices

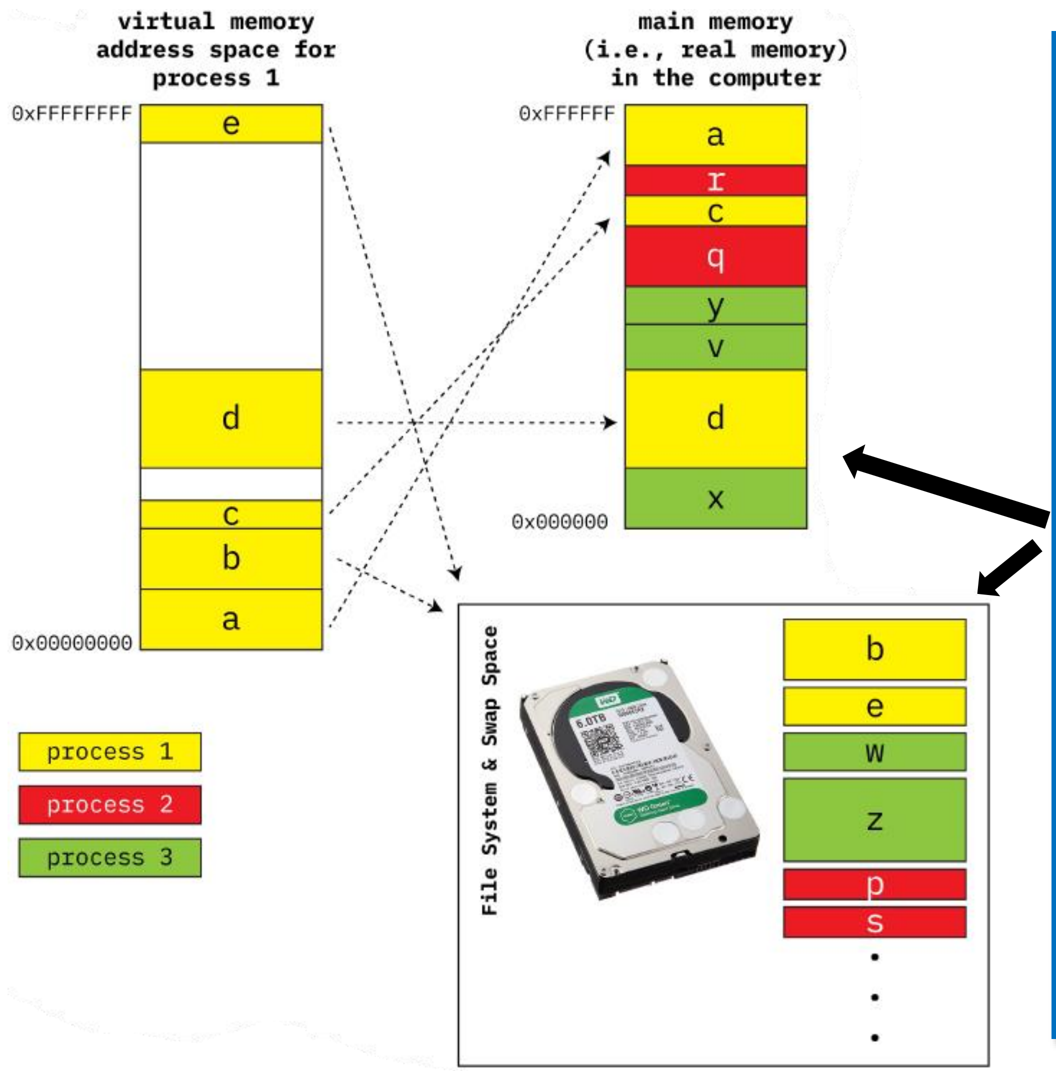
Protection and Security

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
  - User identities (**user IDs**, security IDs) include name and associated number, one per user
  - User ID then associated with all files, processes of that user to determine access control
  - Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file
  - **Privilege escalation** allows user to change to effective ID with more rights

Virtual memory

- Multiprogramming intuition:
  - if a task is in memory, then the CPU can be switch to work on the task
- Problem:
  - What if there is not enough memory available to hold a task?
- Chosen solution:
  - **Logical view** of memory does not need to correspond to **physical view** of memory
  - That is, **only those parts of a task that are needed in physical memory need be resident.**
  - This is known as **virtual memory**.

Virtual memory (cont.)



The code within a process sees a single address space.

That single address space, however, is not real (i.e., it is virtual). It is an illusion implemented by the computer + operating system.

In reality, different parts of the virtual address space are stored in real memory, and some (not needed at present) are stored on disk.

Only one virtual address space is shown here, but there are as many virtual address spaces as there are processes.

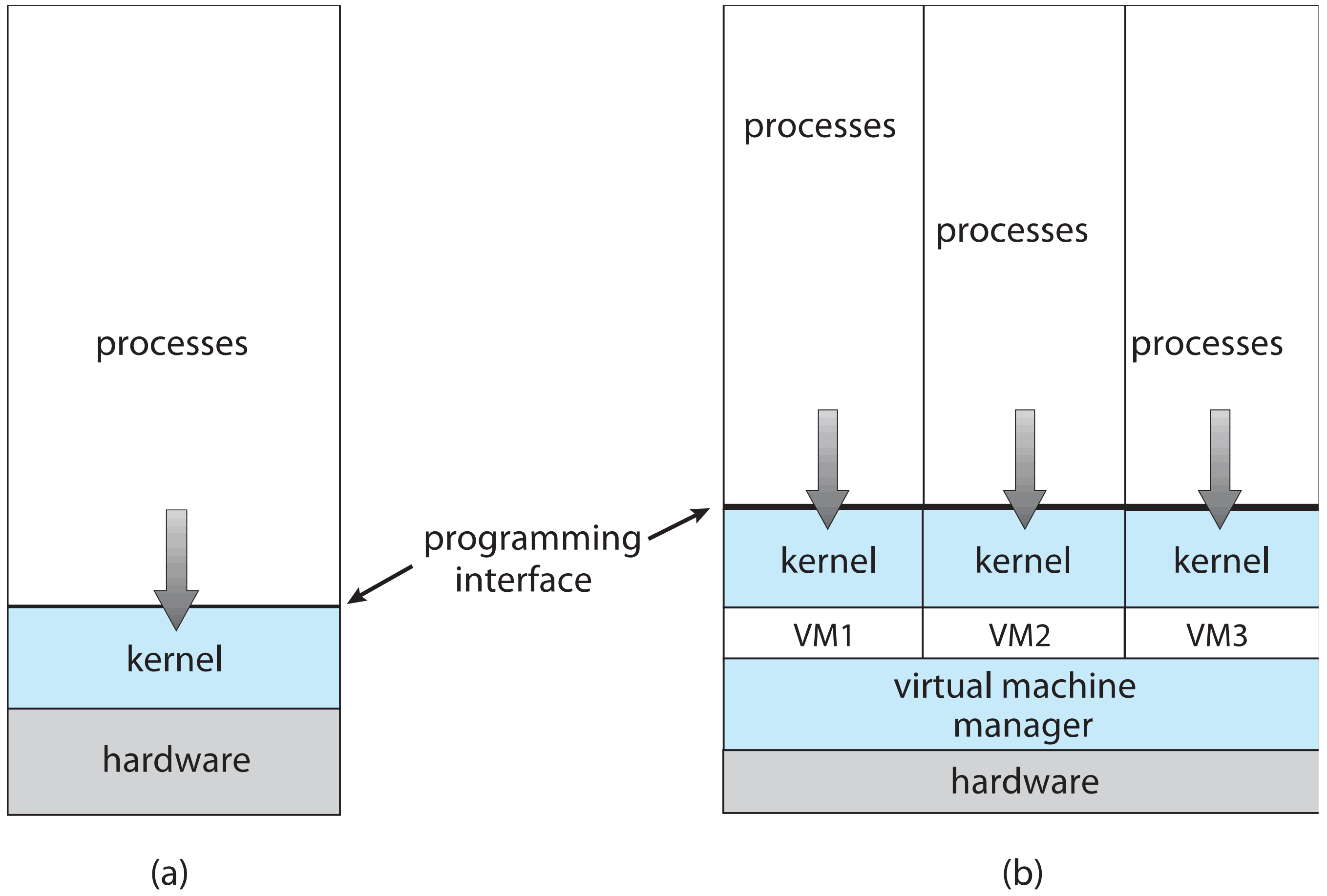
Virtualization

- Allows operating systems to run applications within other OSes
  - Vast and growing industry
- **Emulation** used when source CPU type different from target type (i.e. PowerPC to Intel x86)
  - Generally slowest method
  - When computer language not compiled to native code – **Interpretation**
- **Virtualization** – OS natively compiled for CPU, running **guest** OSes also natively compiled
  - Consider VMware running Windows 10 guests, each running applications, all on native Windows 10 **host** OS
  - **VMM** (virtual machine Manager) provides virtualization services

Virtualization (cont.)

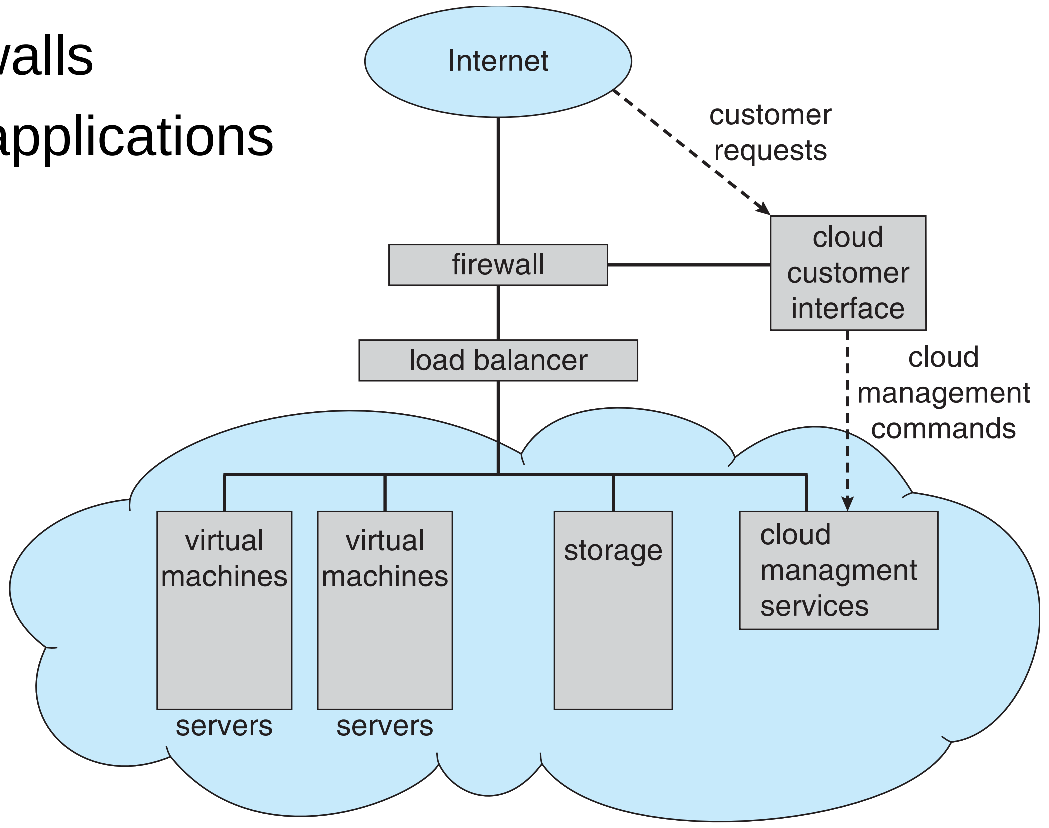
- Use cases involve laptops and desktops running multiple OSes for exploration or compatibility
  - Apple laptop running Mac OS X host, Windows as a guest
  - Developing apps for multiple OSes without having multiple systems
  - QA testing applications without having multiple systems
  - Executing and managing compute environments within data centers
- VMM can run natively, in which case they are also the host
  - There is no general purpose host then (VMware ESX and Citrix XenServer)

Virtualization (cont.)



Cloud Computing

- Cloud computing environments composed of traditional OSes, plus VMMs, plus cloud management tools
  - Internet connectivity requires security like firewalls
  - Load balancers spread traffic across multiple applications



Real-Time Embedded Systems

- Real-time embedded systems most prevalent form of computers
  - Vary considerable, special purpose, limited purpose OS,   real-time OS
  - Use expanding
- Many other special computing environments as well
  - Some have OSes, some perform tasks without an OS
- Real-time OS has well-defined fixed time constraints
  - Processing must be done within constraint
  - Correct operation only if constraints met

Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary **closed-source**
- Counter to **the copy protection** and **Digital Rights Management (DRM)** movement
- Started by **Free Software Foundation (FSF)**, which has “copyleft” **GNU Public License (GPL)**
- Examples include **GNU/Linux** and **BSD UNIX** (including core of **Mac OS X**), and many more
- Can use VMM like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - http://www.virtualbox.com)
    - Use to run guest operating systems for exploration

Any Questions?