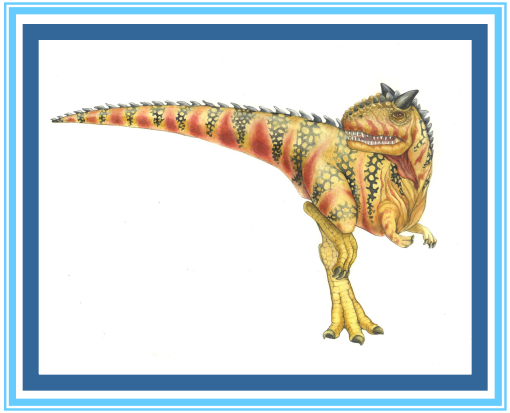
Chapter 1: Introduction (to Operating Systems)



Chapter 1: Introduction

- What we are going to discuss in this chapter:
  - What Operating Systems are & what they do.
  - Computer-System Organization & Architecture (very brief background)
  - Operating-System Structure (parts of an OS).
  - Operating-System Operations (tasks performed by each).
    - Process Management
    - Memory Management
    - Storage Management
    - Protection and Security
  - Some Computing Environments (traditional, mobile etc.).
  - Open-Source Operating Systems.

Objectives of Chapter 1

- To describe the basic organization of computer systems and the role of interrupts in a computer system.
- To give a tour of the major components of operating systems.
  - An introductory discussion.
  - More details about each component in the later chapters.
- Illustrate the dual-mode operation of an OS.
- To discuss how an OS is used in various computing environments.
- To discuss free and open-source operating systems.

Some Famous Operating Systems

- **For desktops & laptops?**
  - Microsoft Windows – 95, 98, 2000, XP, Vista, 7, 8, 10.
  - Ubuntu Linux – Trusty Tahr (kind of goat), Xenial Xerus (squirrel).
  - Apple Mac OS – Cheetah, Puma, Jaguar, Tiger, Leopard, Snow Leopard, Yosemite, El Capitan, Sierra, High Sierra.
- **For Servers.**
  - Windows Server, Red Hat Enterprise Linux Server.
- **Phones & tablets?**
  - Android – Jelly bean, Kit Kat, Marshmallow, Nougat.
  - iOS – 8, 9, 10, 11, 12…?.
- **Watches.**
  - Android wear.
  - iWatch OS.
- **Key point: operating systems are of various kinds, offering varied functionality (desktop vs mobile OS vs watch OS?).**

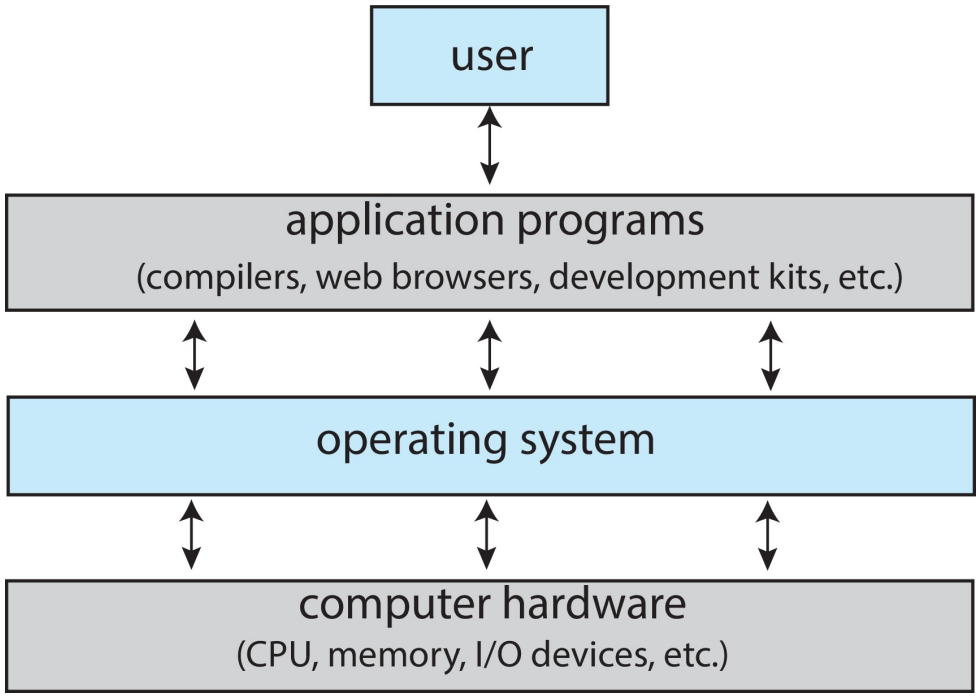
What is an Operating System?

- **A program that acts as an intermediary/mediator between a user of a computer and the computer hardware.**
- A large piece of software (code) that sits between the hardware & the user.
- Operating system goals:
  - Execute user programs (on user's behalf).
  - Make the computer system easy to use (for users).
  - Use the computer hardware in an efficient manner.
- **Note that some of the goals may be conflicting. Example?**
- Ease of use may come with a side effect of less than optimal resource utilization.
- Consider a command line vs. graphic user interface (GUI).
- GUI is more user friendly, but less efficient, as some of the hardware resources are being used to support it.
- Command line, though not so easy to use, is more efficient.

Computer System Structure

- **What's the role of an OS in the overall computer system?**
- A computer system can be divided into four components:
  - **Hardware** – provides basic computing resources**.**
    - CPU, memory, disk, I/O devices.
  - **Operating system.**
    - Controls and coordinates use of hardware among various applications and users
  - **Application programs** – define the ways in which the system resources are used to solve the computing problems of the users.
    - Word processors, web browsers, database systems, video games.
  - **Users**
    - People.
- Next slide shows the structure of a computer system.

Abstract View of Components of Computer



OS goals?

- **From the user's view.**
- Regular/common users want **convenience**/**ease of use** and **good performance.**
  - Don't care much about **resource utilization** (how hardware is shared, typically between multiple users).
- But, shared computers such as **mainframes/servers** must keep all users happy, it's more about **fairness** among users.
- Users of dedicated systems such as **workstations** have dedicated resources but frequently use shared resources from **servers** **(a combination of the above two options).**
- Handheld computers are resource poor (compared to desktops), optimized for **usability** and **battery life**.
  - Need user interfaces such as touch screens, voice recog.
- Some computers have little or no user interface (to interact with hardware), such as embedded computers in home devices (fridge) and automobiles.

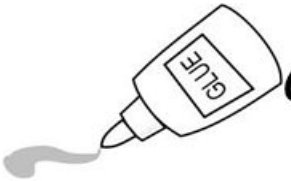
System View

- OS is a **resource allocator.**
  - Manages all hardware resources of a computer.
  - Decides between conflicting requests for efficient and fair resource use.
  - For example, if multiple programs want the CPU, how to allocate?
- OS is a **control program.**
  - Controls execution of programs & usage of various I/O devices to prevent errors and improper use of the computer.
  - One program cannot interfere in the execution of another program. What is wrong with this interference?

What is an Operating System?

- Referee
  - Manage sharing of resources, Protection, Isolation
    - » Resource allocation, isolation, communication

- Illusionist
  - Provide clean, easy to use abstractions of physical resources
    - » Infinite memory, dedicated machine
    - » Higher level objects: files, users, messages
    - » Masking limitations, virtualization



- Glue
  - Common services
    - » Storage, Window system, Networking
    - » Sharing, Authorization
    - » Look and feel

What does an OS contain?

- Debatable, no universally accepted theory.
- "*Everything a vendor ships when you order an operating system*" – a naïve/ simplistic approximation.
  - This varies a lot.
  - Some operating systems are < 1 MB (embedded devices, a microwave oven) with little/no GUI, others are several GBs (Windows 10 needs 15 GB, iOS 12 takes up 2.25 GB).
- Another theory: "*the one program running at all times on the computer*", called the **kernel/core**.
- Every other software is either:
  - A **system program:** ships with the operating system, but not part of the kernel, e.g. compiler, assembler.
  - An **application program:** all programs not associated with the operating system, e.g. Facebook, Twitter.

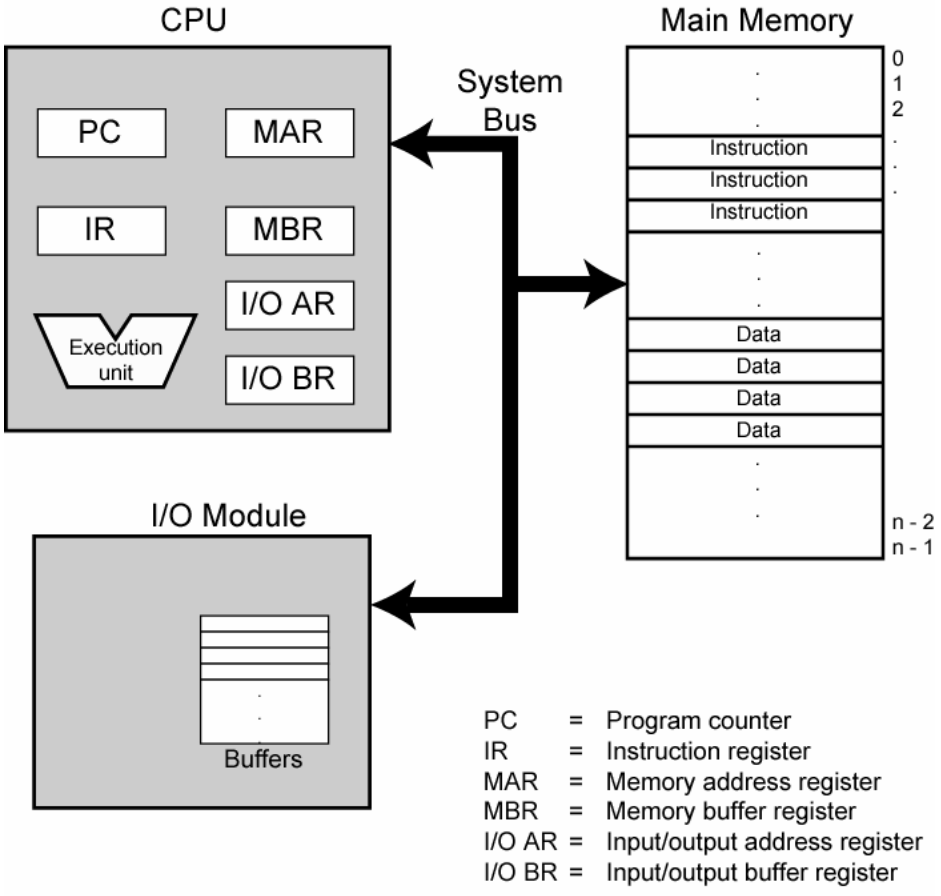
Computer Startup

- **Bootstrap program** is loaded at power-up or reboot.
  - To boot means to start.
  - Computer needs an initial program to run when it is powered on.
  - This program is typically stored in ROM (read only memory), generally known as **firmware.**
  - It typically initializes the computer system – CPU registers, device controllers, memory contents etc..
  - Bootstrap locates (on disk) & loads operating system kernel (into memory).
  - Once kernel is up & running, it waits for some user event to occur.
    - Connecting to a Wi-Fi network.
    - Opening a web browser.
  - Bootloader for Linux: GRUB (Grand Unified Bootloader).

Computer Architecture (Very Fast Recap)

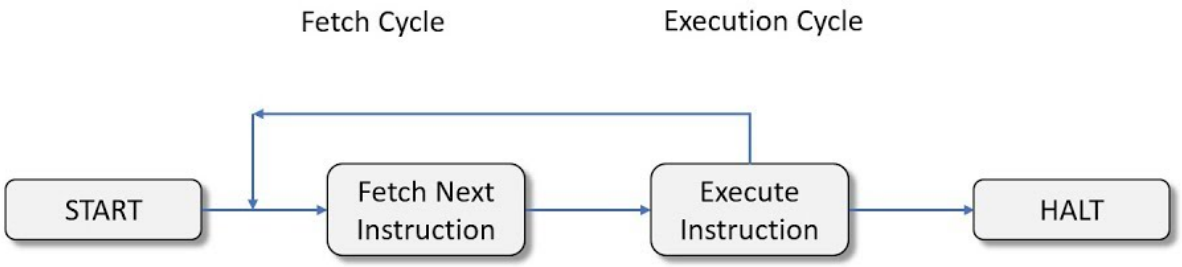
- **What are the major components of a computer?**
- CPU.
- Memory: main and disk.
- I/O components.
- All these are interconnected.
- **What is the main objective of a computer?**
- To execute user programs.
- CPU: executes the instructions.
- Memory: stores data and programs.
- I/O modules: move data between computer and external environment.

Computer Components



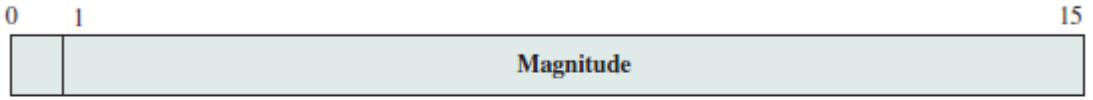
Instruction Cycle

Computer Instruction Cycle



Instructure Components

(a) Instruction format



(b) Integer format

Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

(d) Partial list of opcodes

Example of Program Execution

- Goal: add contents of memory word at address 940 to the contents of memory word at address 941, and store the results in the latter location.
- Three instructions (with three fetch and execute stages are needed):
- PC contains 300, address of 1st instruction. This instruction (1940) is loaded into the IR, and PC is incremented, to 301.
- 1st four bits of instruction in IR indicate AC to be loaded from memory. Remaining 12 bits specify memory address = 940. This will load contents of 940 (= 3) into AC.
- Next instruction (= 5941) is fetched from 301, and put in IR. Also, PC is incremented, to 302.
- 1st four bits of instruction in IR indicates add to AC from memory address 941 (which contains 2). We do 3 + 2 = 5 and put 5 back in AC.
- Next instruction (=2941) is fetched from 302, and put in IR. Also, PC is incremented, to 303.
- 1st four bits of IR instructions indicate to store AC contents to memory 941.
- We are now done.

Program Execution

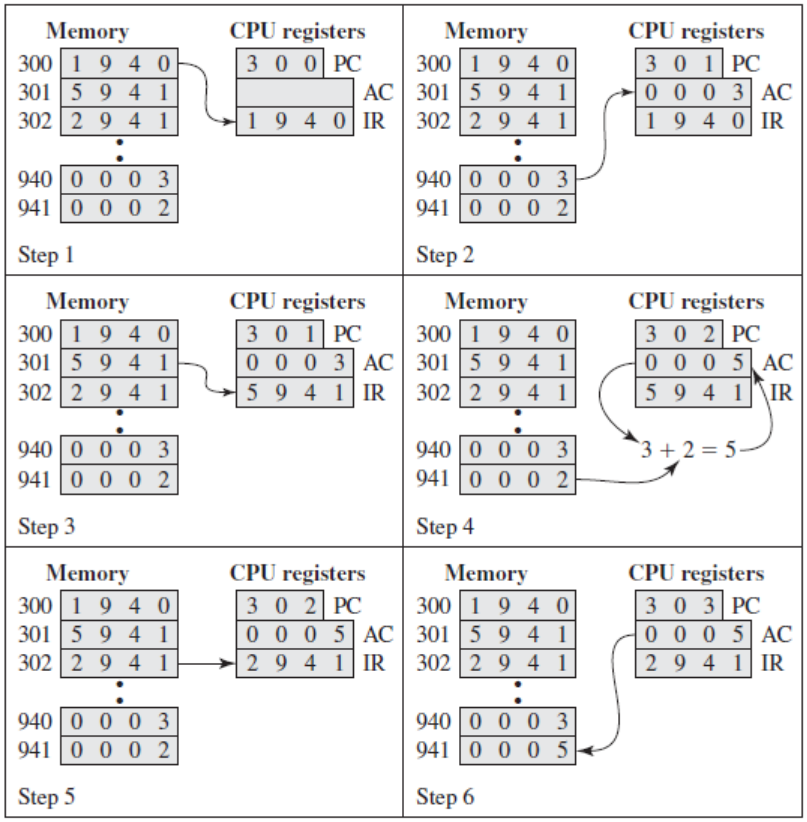
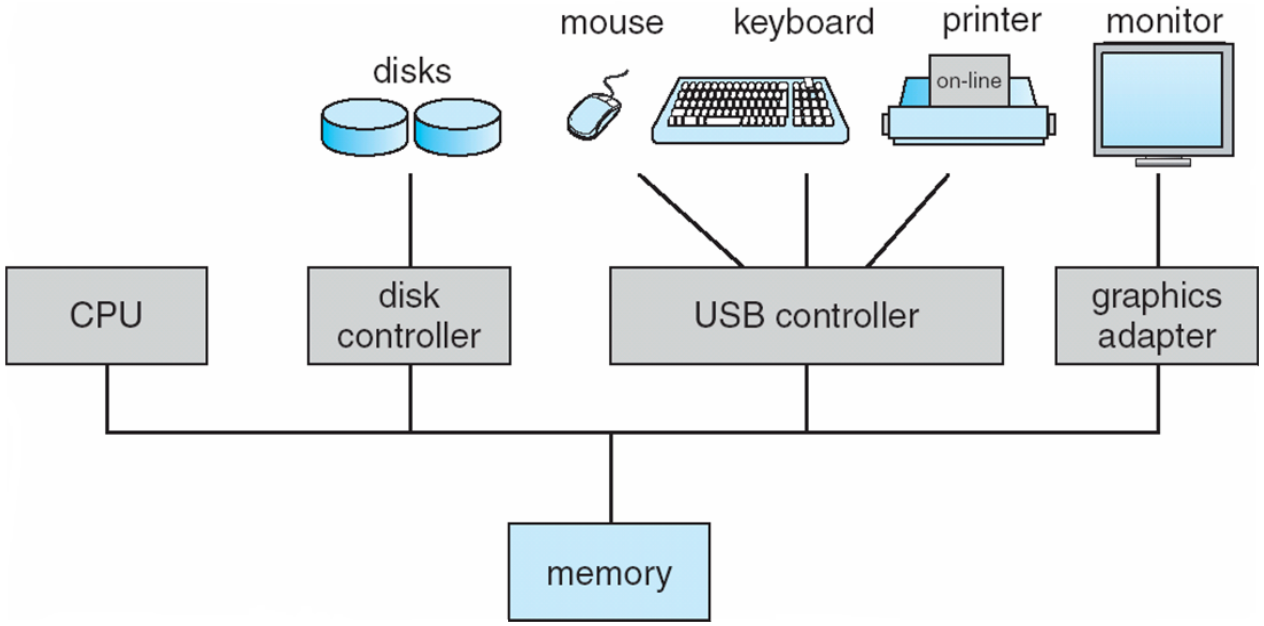


Figure 3.5 Example of Program Execution (contents of memory and registers in hexadecimal)

Computer System Organization

- Computer-system consists of:
  - One or more CPUs, device controllers connected through a common bus providing access to shared memory.
  - Concurrent execution of CPUs and devices competing for memory.



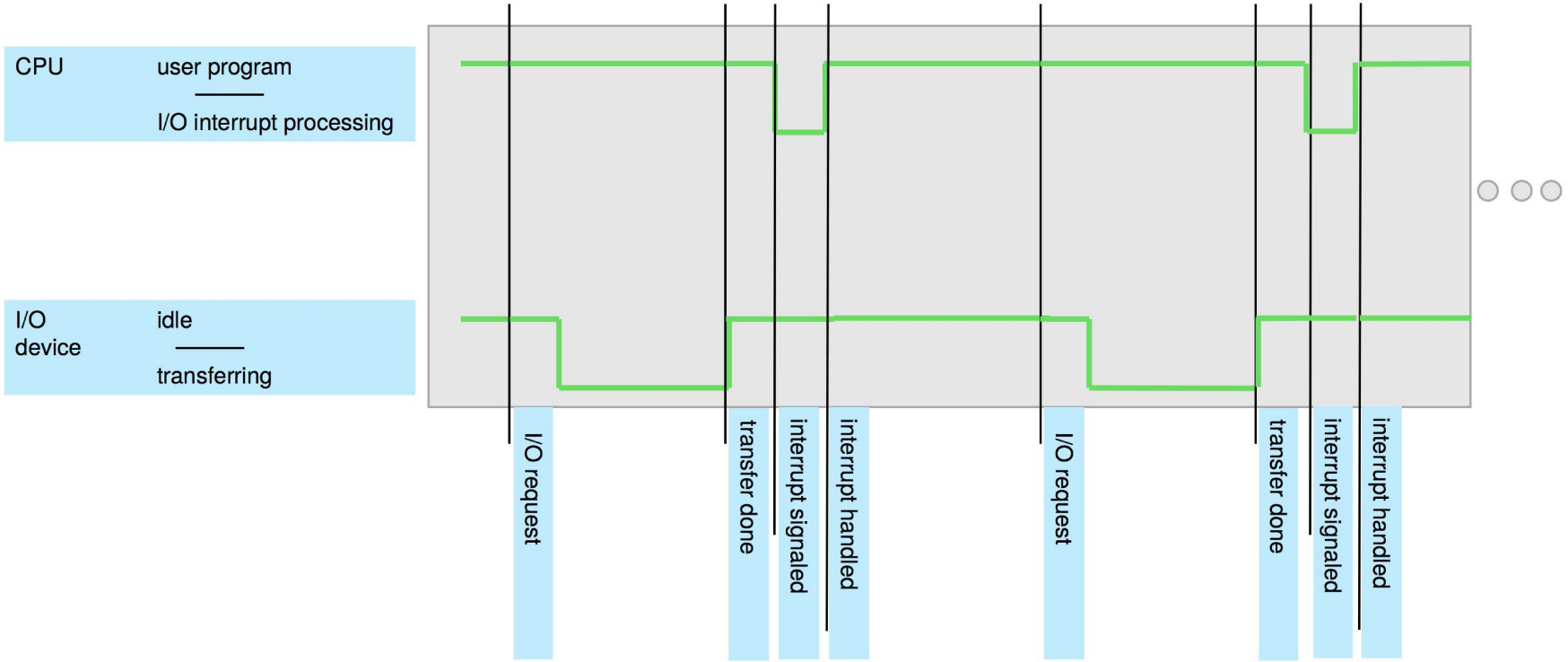
Device Controllers

- Note that each I/O device (disk, USB) has an associated controller.
- **A device controller is a piece of hardware that connects the operating system to the various devices, such as keyboard, mouse, printer etc.**
- Each device controller is in charge of a particular device type.
- **Goal of a device controller?**
- Make I/O data available to the OS.
  - So that the OS can act on it.
- Controller may have it's own local buffer to store data.
  - For example, data read by keyboard is put in a buffer, from where it can be read by the OS.
- Controller has a piece of code that accomplishes its task, and it is called a device driver.
  - Device driver is software that manages device (hardware) controllers.

Interrupts

- How does a device driver informs the CPU that it has finished its operation?
- By causing an interrupt.
- **Events in an OS are signaled by interrupts.**
- Interrupts (to an OS) may be signaled by either the hardware or the software (called traps or exceptions or system calls).
- Hardware interrupts CPU by sending a signal.
  - Signal sent through the system bus.
- Software interrupts CPU by executing a system call.
  - This is a piece of code.
- Key concept: **an operating system is interrupt driven**.
  - This is how, for example, the CPU gets to know about a program that wants to start execution.
  - Many other events signaled by interrupts, as we will soon see.

Interrupt Timeline



Interrupt Time Line

- Here's the sequence of events:

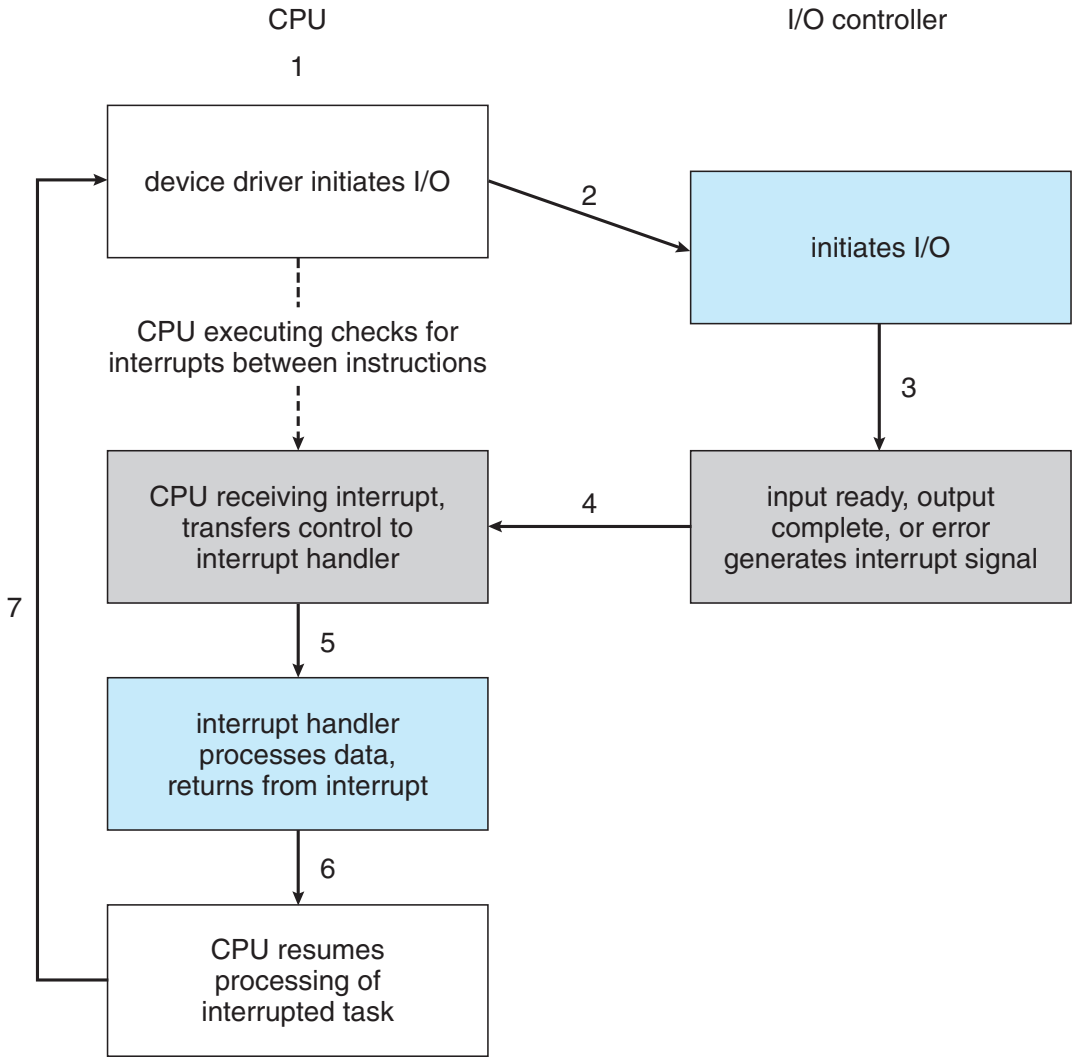
1. CPU is busy executing some instructions of a program.
2. CPU is interrupted.
3. It stops it's execution, & transfers to a fixed location in memory, that contains the **interrupt service routine**.
4. This routine/code is executed.
5. On completion, CPU resumes execution of interrupted program.

- Table of pointers is maintained.
- Each pointer points to a specific interrupts service routine.
- This table of pointers is stored in low memory locations, & is called an **interrupt vector, say *iv[100].**
- Vector – "to direct to a desired point".
- Both Windows and Linux service interrupts in this manner.

Interrupt Implementation

- CPU hardware has a wire (interrupt request line) that it senses after executing every instruction.
- When CPU detects a device controller has sent a signal on the line, it reads interrupt number and jumps to relevant isr.
- CPU executes the isr code (to service the interrupt).
- **Note that current state of CPU needs to be saved (and restored later).**
- After servicing the interrupt, CPU goes back to what it was doing (execute next instruction of the halted program). **Need a state restore for this.**
- In modern OSes, interrupts are assigned a priority level (which one to service now, which one can wait a bit?).
- Vector number 0: divide error, 6: invalid opcode, 7: device not available, 14: page fault etc.
- Integers here are indexes to array of pointers.

Interrupt I/O Cycle



Types of Interrupts

Program interrupts	As a result of instruction execution e.g. arithmetic overflow, out of bounds instruction.
Timer interrupts	Generated by a timer in the processor to perform functions at regular intervals.
I/O interrupts	Normal I/O completion or error conditions.
Hardware failure interrupts	Power failure.

Storage Structure

- CPU can load instructions only from main memory.
- Any program to be run must be stored there.
- All memory provides an array of bytes.
- Each byte is addressable using a memory location.
- Memory – CPU: load instruction.
- CPU – memory: store instruction.
- Main memory is too small to accommodate all programs.
- Hence, we have external memory, also known as disks.

Storage Structure

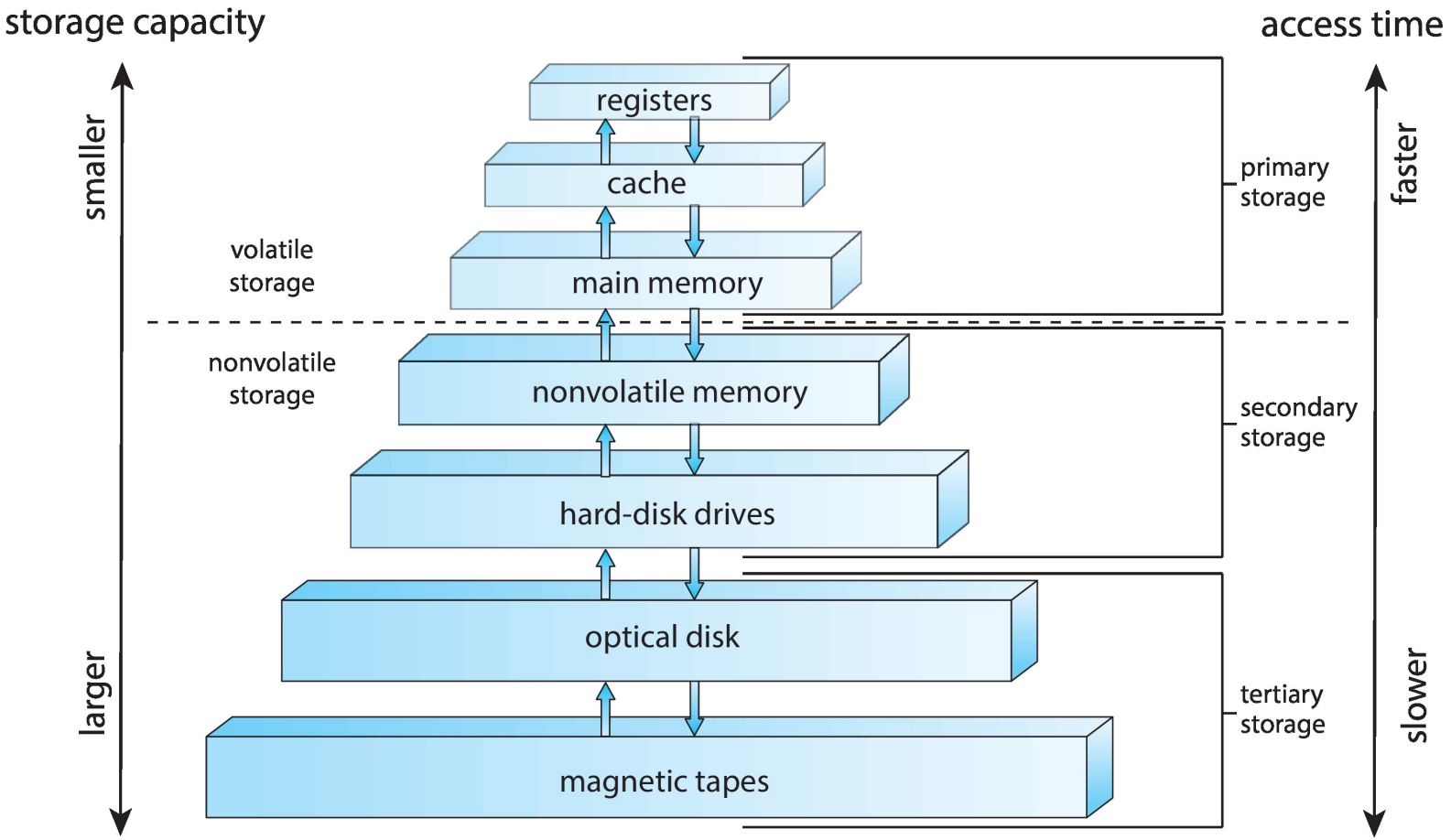
- Main memory – only large storage that the CPU can access directly.
  - **Random access** (any memory cell accessed in same amount of time).
  - Typically **volatile** (contents lost on power down).
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity. This consists of:
- **Hard disks (HD)** – 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** takes care of the interaction between the device and the computer.
- **Solid-state disks (SSD)** – faster than hard disks, nonvolatile.
  - Various technologies.
  - Becoming more popular.
- More on this later in chapters on memory management.

Storage Definitions and Notation Review

The basic unit of computer storage is the **bit** . A bit can contain one of two values, 0 and 1. All other storage in a computer is based on collections of bits. Given enough bits, it is amazing how many things a computer can represent: numbers, letters, images, movies, sounds, documents, and programs, to name a few. A **byte** is 8 bits, and on most computers it is the smallest convenient chunk of storage. For example, most computers don't have an instruction to move a bit but do have one to move a byte. A less common term is **word**, which is a given computer architecture's native unit of data. A word is made up of one or more bytes. For example, a computer that has 64-bit registers and 64-bit memory addressing typically has 64-bit (8-byte) words. A computer executes many operations in its native word size rather than a byte at a time.

Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes. A **kilobyte** , or KB , is 1,024 bytes; a **megabyte** , or **MB** , is $1,024^2$ bytes; a **gigabyte** , or GB , is $1,024^3$ bytes; a **terabyte** , or **TB** , is $1,024^4$ bytes; and a **petabyte** , or **PB** , is $1,024^5$ bytes. Computer manufacturers often round off these numbers and say that a megabyte is 1 million bytes and a gigabyte is 1 billion bytes. Networking measurements are an exception to this general rule; they are given in bits (because networks move data a bit at a time).

Storage-Device Hierarchy



Caching

- Would have studied this in CSL 211.
- What is caching?
- Important principle, performed at many levels in a computer (in hardware, operating system, other software).
- Data in use copied from slower to faster storage temporarily.
- Faster storage (cache) checked first to determine if data is there.
  - If it is, data used directly from the cache (fast).
  - If not, data may be copied to cache and used.
- Challenge: cache smaller than data being cached.
  - Cache management important design problem.
  - Cache size and replacement policy.

Operating-System Operations

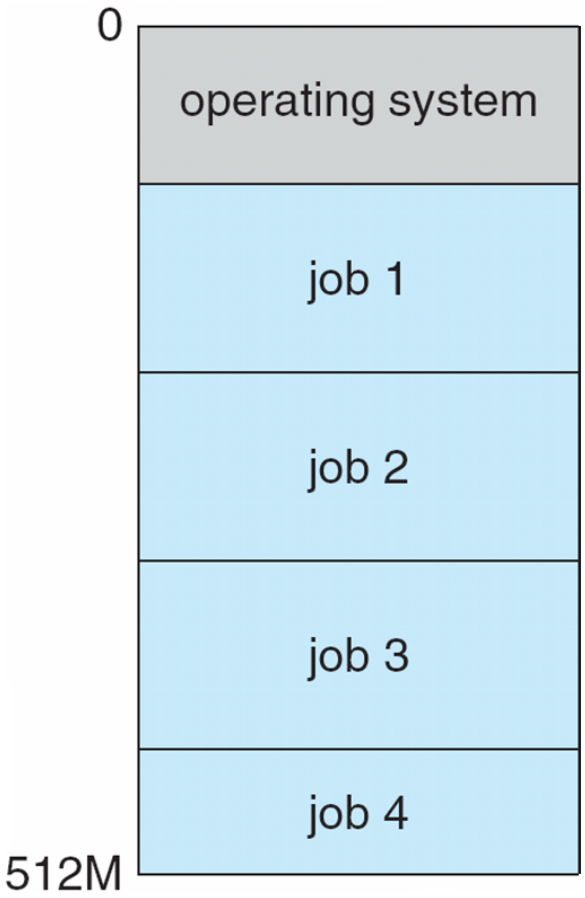
- After review of architecture, let's focus back on operating systems.
- Bootstrap program – simple code to initialize the system, load the kernel (discussed earlier).
- No user programs to run, no I/O to process, kernel simply waits for an event, signaled by in interrupt.
- **Interrupt driven** (hardware 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 -> infinite loop, processes modifying each other or the operating system.

Multiprogramming and Multitasking

- **Multiprogramming**.
  - 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.
  - Advantage: > CPU utilization, keeps users satisfied.

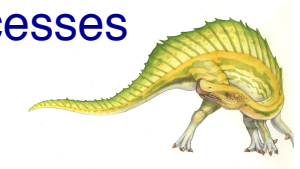
- **Timesharing** (**multitasking**) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing.
  - Illusion is that system executes multiple programs "simultaneously".
  - **Response time** should be less, say, < 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



Dual Mode

- Software interrupt from a user program that an OS service be provided. This interrupt is called a system call.
- The OS and user programs share computer hardware.
- Incorrect or malicious program should not cause other programs, or the OS to operate incorrectly.
  - E.g. process hogging an IO device or the CPU.
- Need a security mechanism that distinguishes between user code and OS code.
- Such a mechanism is called – dual mode operation.
- **Dual-mode** operation allows OS to protect itself and other programs from each other.
  - **Two modes: User mode** and **kernel mode.**
  - In kernel mode, CPU can execute all instructions & access all areas of memory (unrestricted access).
  - In user mode, CPU is restricted to certain instructions & accesses certain areas of memory.

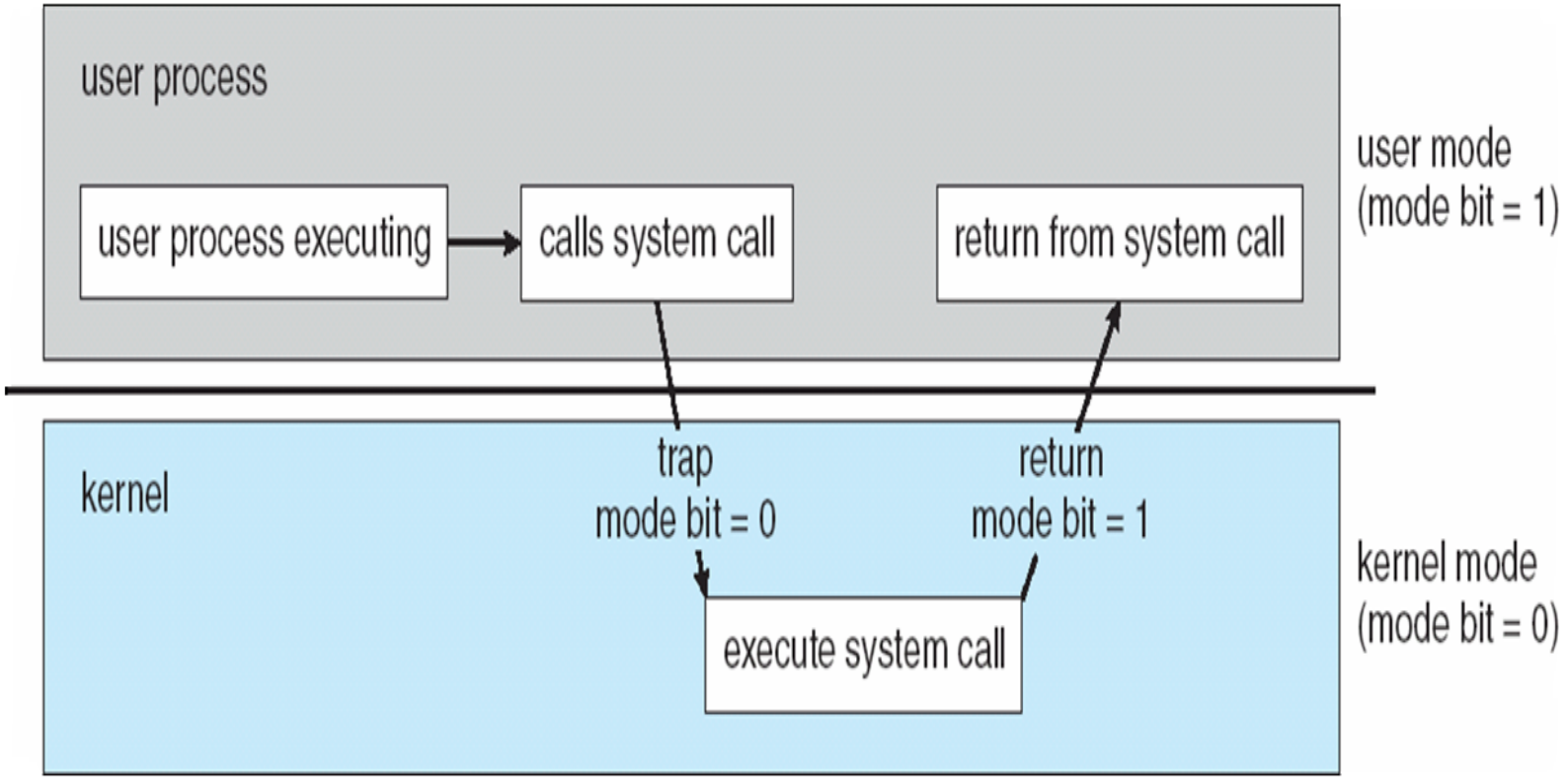


Dual Mode Operation

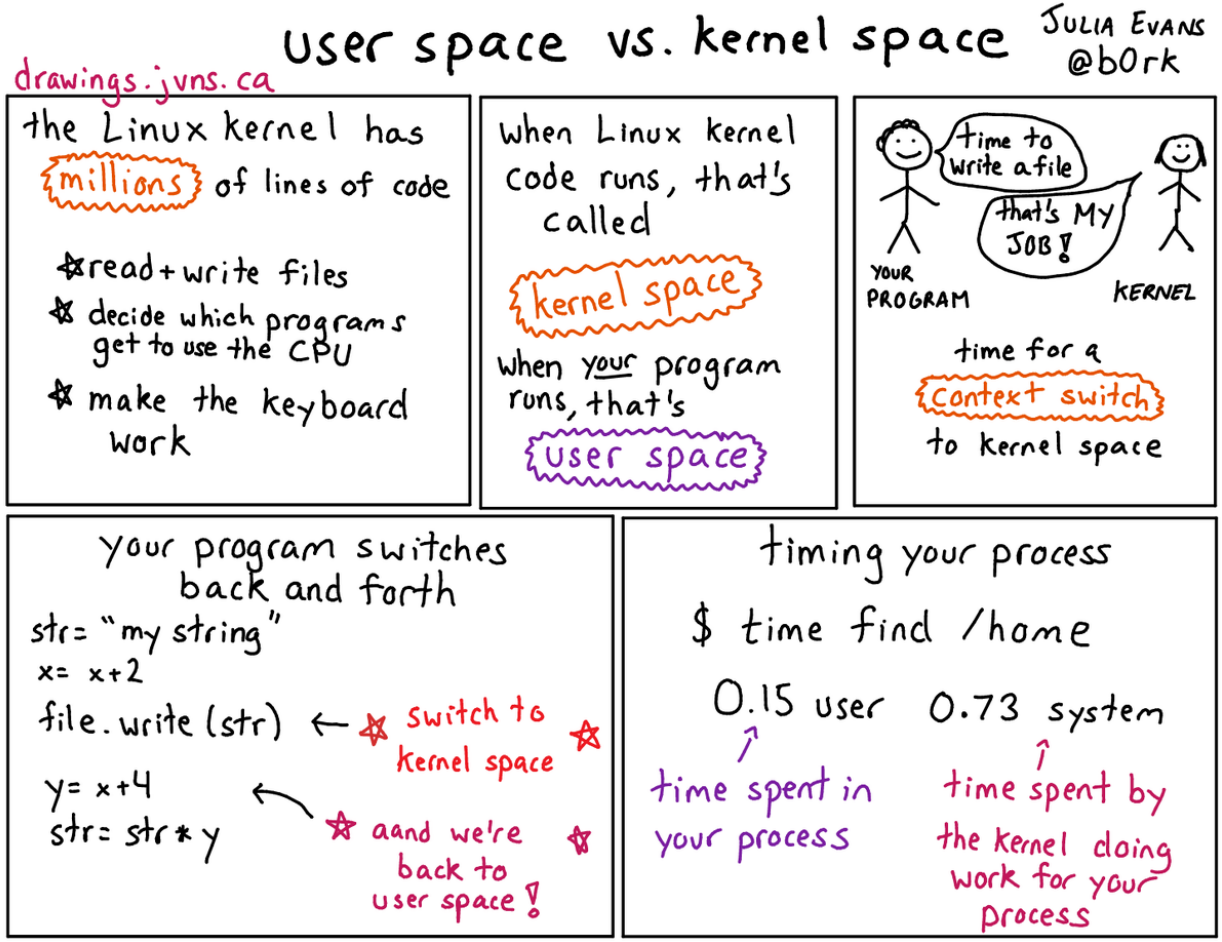
- **Mode bit** provided by hardware.
  - Provides ability to distinguish when system is running user code or kernel code.
  - Some instructions designated as **privileged**, only executable in kernel mode.
  - System call changes mode to kernel, return from call resets it to user.
- What if user code tries to execute a privileged instruction?
- Hardware traps to the OS, which may terminate the process, dump it's code to memory for examination (core dump).

- Example of privileged instructions:
  - Go from user → kernel mode.
  - Interrupt management.
  - I/O device management.

Transition from User to Kernel Mode



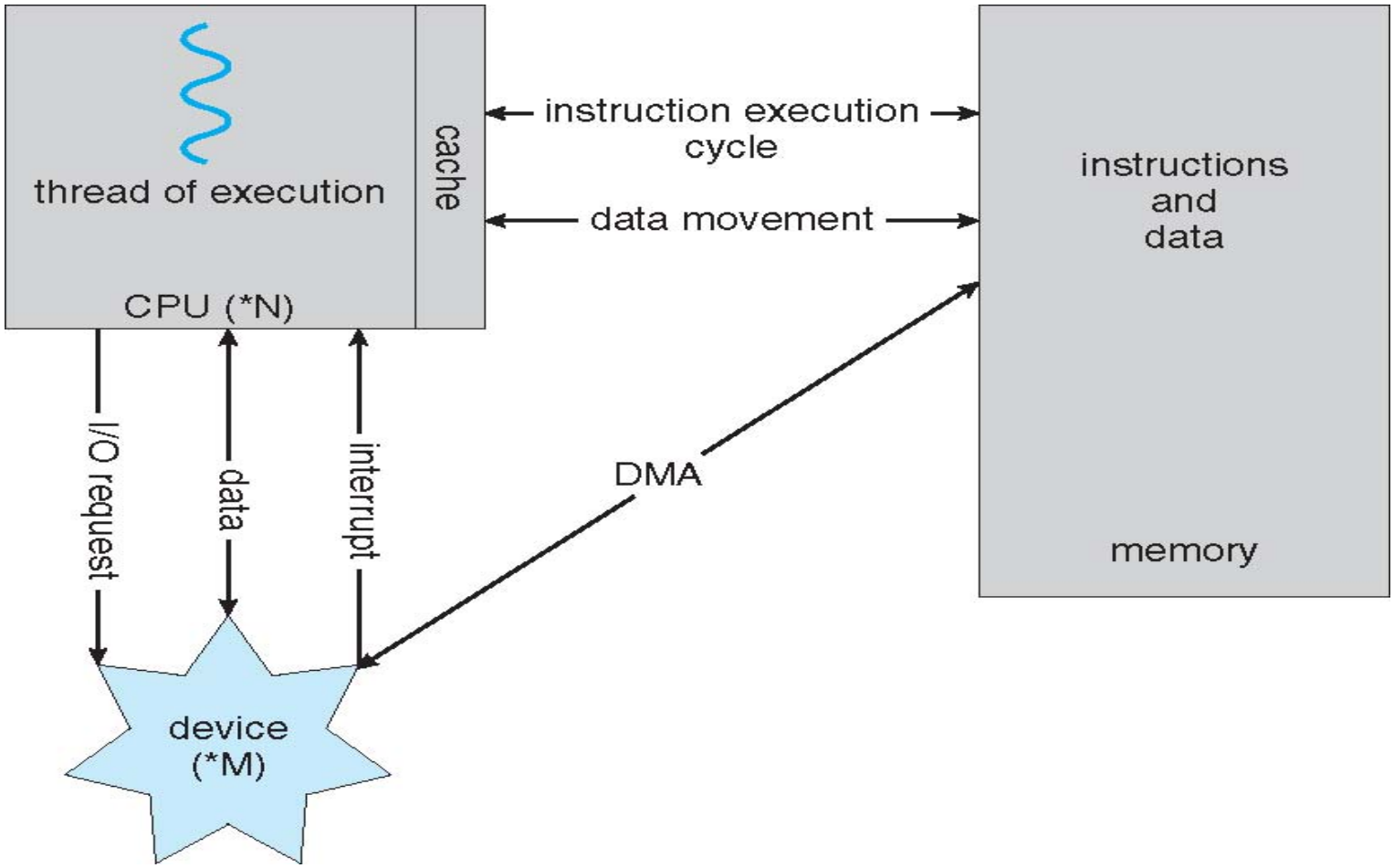
Source: Julia Evans, Twitter.



UNIX time utility

- Also try: time find /usr.
- Three kids of time shown in the output:
- User time – time spent executing user code.
- Sys time – time spent executing kernel code.
- Real time – total time spent, including wait times.
- User + Sys = CPU time used by process, across all CPUs.

How a Modern Computer Works



A von Neumann architecture

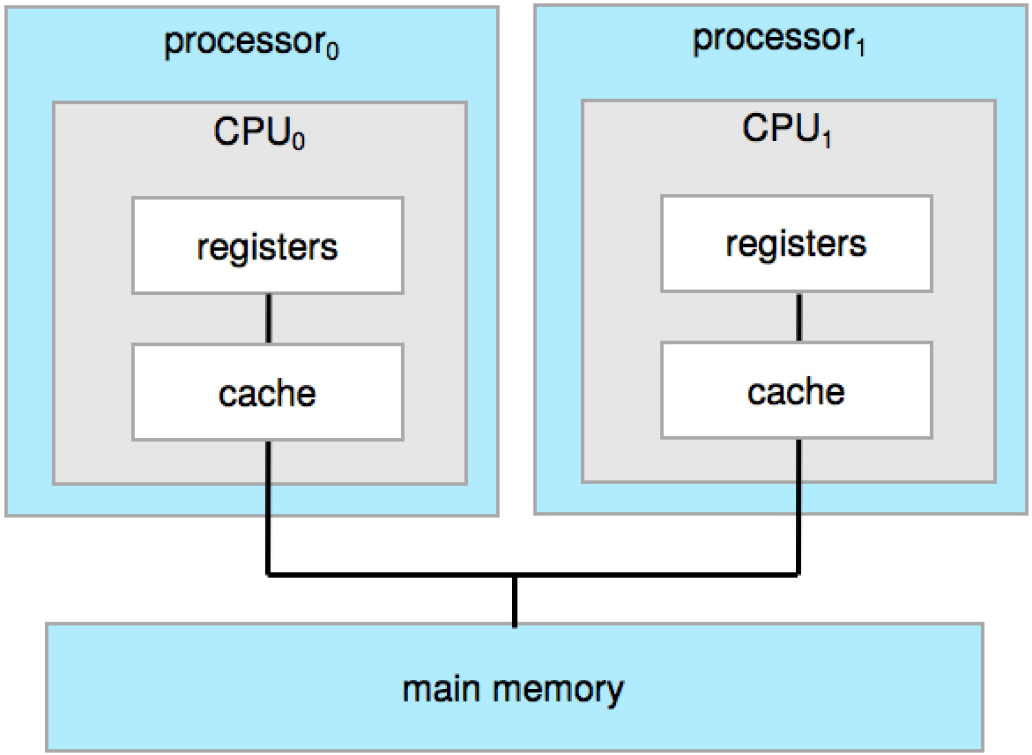
1. Single CPU systems

- We now briefly discuss some popular computer architectures.
- When hardware was expensive, most systems used to have a single **general-purpose** processor (CPU).
- As opposed to general purpose CPUs, we now have special purpose CPUs.
- For example, for graphics (GPUs) or for capturing motion (motion co-processor) on the iPhone.
  - The motion co-processor collects data from motion sensors (accelerometer, compass), used by various motion based apps.
- In a single CPU machine, at any point of time, only one program may run (instructions of only one program may be executed).
  - However, we can still simultaneously support multiple programs, even with a single CPU. How?
  - Using multiprogramming.

2. Multiprocessor Systems

- **Multiprocessors (parallel)** systems growing in use and importance.
  - Have multiple CPUs on the same machine.
  - Advantage of this?
  - Can support more programs at a time.
  - However, the CPUs do share the memory & I/O devices (contention reduces expected gain).
  - Multiprocessors first appeared in servers, but are now available on basic desktops, laptops, & even mobile phones.
- Advantages of multiprocessors:
  1. **Increased throughput** (more work done/unit time).
  2. **Economy of scale** (multiple CPUs sharing memory & IO devices vs. single CPU machines, each with their own memory & IO devices, good for programs that share data).
  3. **Increased reliability** – graceful degradation or fault tolerance. If one of the CPUs breaks down, load of that CPU can be re-distributed to remaining CPUs.

Symmetric Multiprocessing Architecture



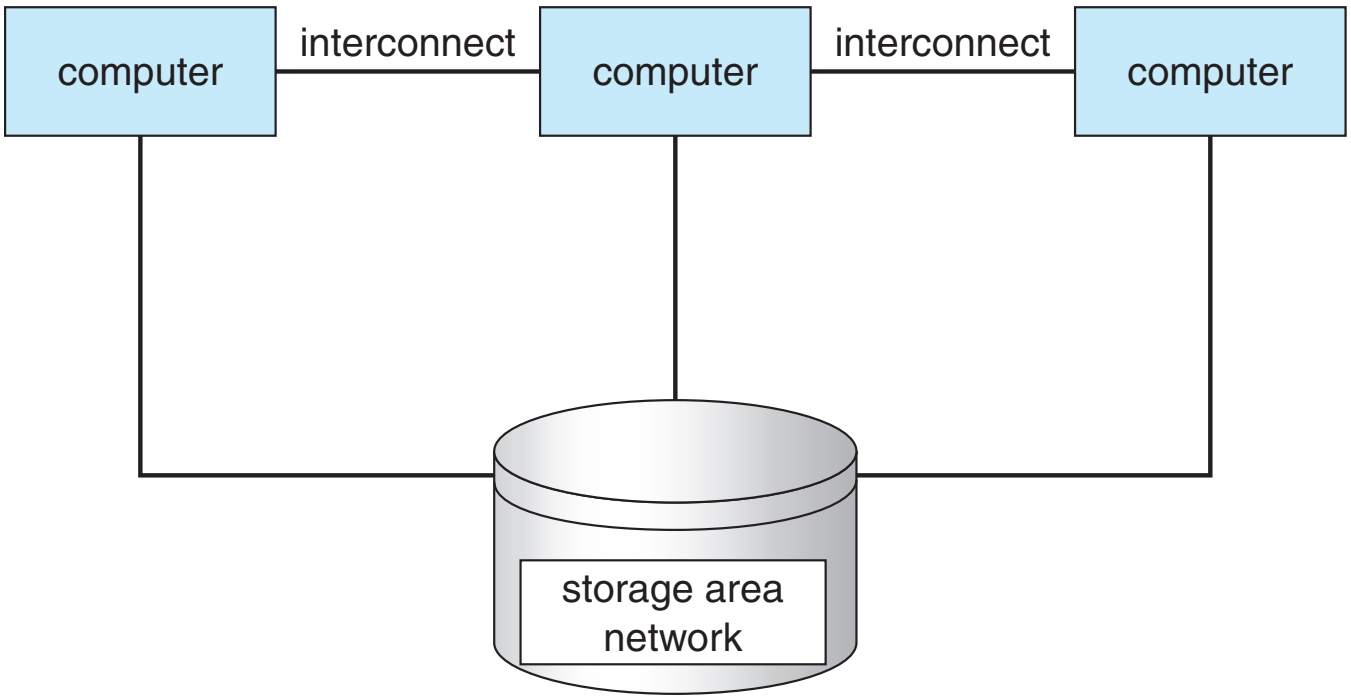
3. Clustered Systems

- **Like multiprocessor systems, but multiple (possibly multiprocessor) systems working together, & are generally distributed in nature.**
  - A bunch of machines, connected geographically apart, connected by a network.
  - Network may be in a building, in a city, or across countries & continents.
  - Example applications: Facebook, Google.
  - Usually sharing storage via a **storage-area network (SAN).**
  - Provides a **high-availability** service which survives failures.
  - One master node may monitor many ordinary nodes, that do computation.
  - If one node fails (hardware failure), master node sends request that was running on failed machine to another machine.
  - Request is re-started, only a brief interruption of service for users.

Clusters (continued)

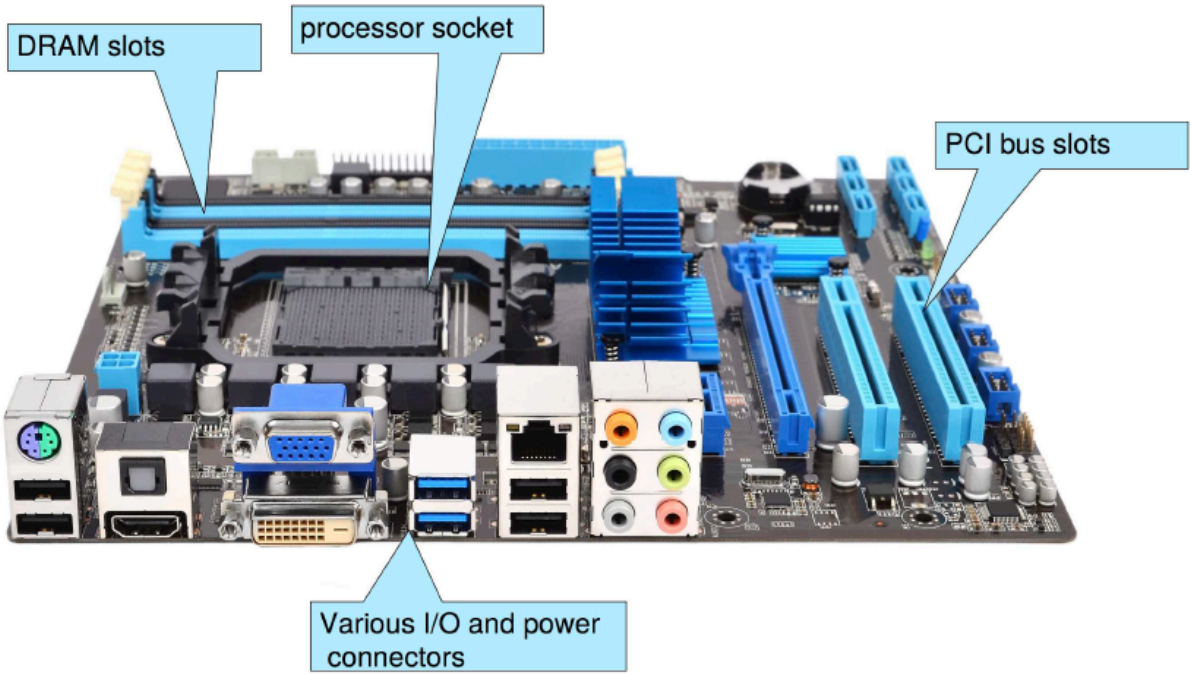
- Clusters generally support **high-performance computing (HPC).**
- Can use power of hundreds of nodes.
- Can run computationally expensive applications on them, say DNA encoding, weather simulations.
- Divide the application into parts, & run each part in parallel on a separate node/machine (multi threading).
- Applications must be written to use **parallelization.**
- Example application for databases – Oracle Real Application cluster.
- It is a version of Oracle's database that can be run on a parallel cluster.
- Multiple nodes can simultaneously access a database, speeding up operations & providing reliability.
- Storage, however, is shared. Can have a Storage Area Network (SAN).
  - Distributed storage to which many node can attach.
  - Cluster software can assign app on SAN to any connected node for running.
- Need to be careful with writes, reads are OK (Distributed Lock Manager).

Clustered Systems



PC Motherboard

Consider the desktop PC motherboard with a processor socket shown below:

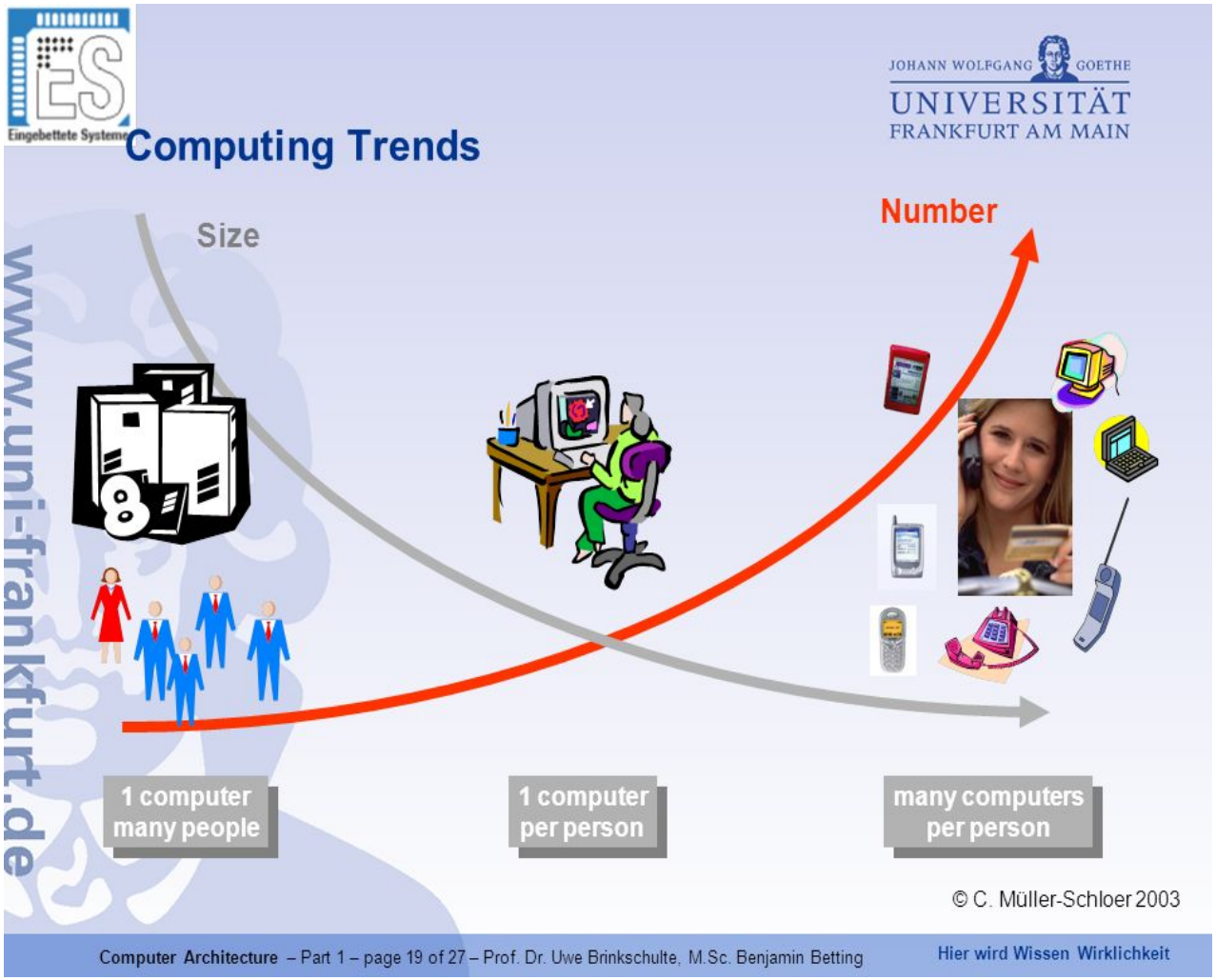


This board is a fully-functioning computer, once its slots are populated. It consists of a processor socket containing a CPU, DRAM sockets, PCIe bus slots, and I/O connectors of various types. Even the lowest-cost general-purpose CPU contains multiple cores. Some motherboards contain multiple processor sockets. More advanced computers allow more than one system board, creating NUMA systems.

Computing Environments - Traditional

- Stand-alone, general purpose machines.
- For example, a desktop or a laptop at home or in the office.
- Definition is blurred as most systems interconnect with others (networked).
- Earlier, hardware was expensive, so we had time shared systems.
  - One machine accessed by multiple users.
  - **Use multiprogramming to provide fairness among users.**
- Today, time shared systems are uncommon in a home or in one office.
- Think about the quantity -> number of people/machine. How has it changed over the years?
- **Multiprogramming can still be used among various programs of a single user (multiple processes for single user).**
- **User processes & system processes each get a slice of the CPU time.**

Source: Computer Architecture, Univ. of Frankfurt, 2003.



Computing Environments - Mobile

- How many mobile phone users in India?
  - 800 million (~ 80 crores) !
- Mobile devices: handheld smartphones, tablets, etc.
- What is the functional difference between them and a "traditional" desktop/laptop?
- More portable, lightweight.
- Less hardware capability (relatively).
- User interface.
- Extra features – more OS features (GPS, accelerometer.)
- Use IEEE 802.11 wireless standard, or cellular data networks for connectivity (EDGE, GPRS, 2G, 3G, 4G, 5G).
- Leaders are: **Apple iOS** and **Google Android.**
- **More on these in chapter 2.**

Computing Environments – Distributed

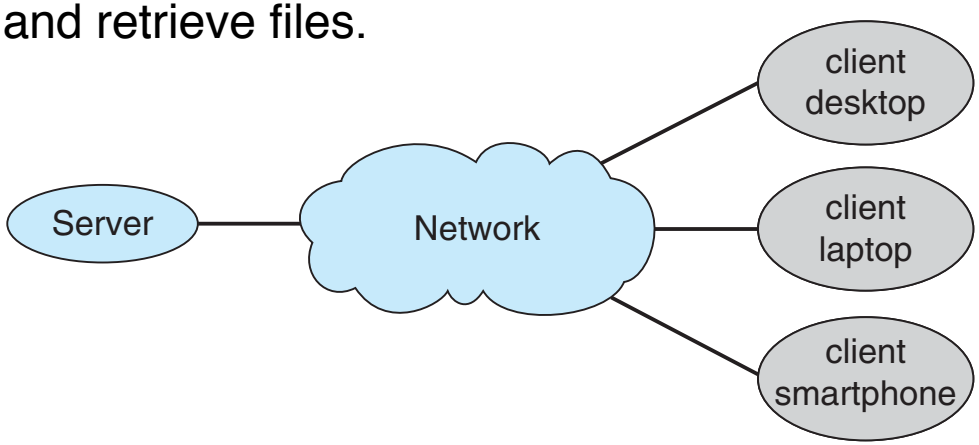
- Distributed computing.
  - Collection of cluster of separate, possibly heterogeneous, systems networked together.
    - **Network** is a communications path, **TCP/IP** most common.
      - **Local Area Network** (**LAN**): within a room/building/campus.
      - **Wide Area Network** (**WAN**): between buildings/cities/countries.
      - **Metropolitan Area Network** (**MAN**): link buildings within a city.
      - **Personal Area Network** (**PAN**): within a room (between a headset & phone, between wireless mouse & keyboard)
  - Operating System needs to support various networking functions & needs to interface with various network hardware devices.

Distributed Computing - II

- Advantage of a distributed system?
- Because we can (fast internet speeds, up to GBPS).
  - Use the computation power of hundreds & thousands of machines to run large applications.
- Data availability/reliability.

Computing Environments – Client-Server

- Client-Server Computing.
  - Dumb terminals replaced by smart PCs, replaced by dumb terminals again?
  - Google server, Facebook server.
  - Many systems now use **servers**, responding to requests generated by **clients.**
    - **Compute-server system** provides an interface to client to request services (i.e., database search).
    - **File-server system** provides interface for clients to store and retrieve files.



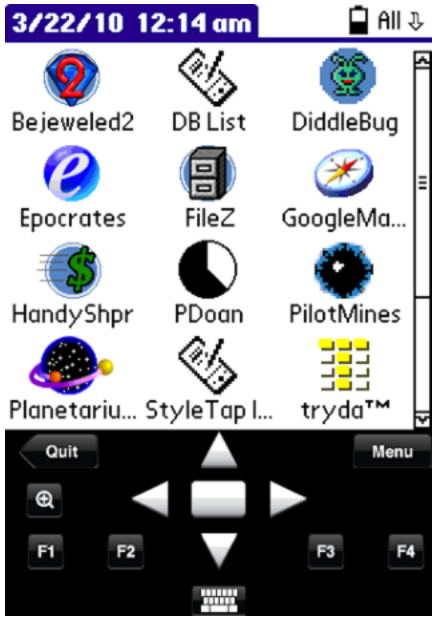
Computing Environments - Peer-to-Peer

- Another model of a distributed system.
- P2P does not distinguish clients and servers.
  - Instead, all nodes are considered peers.
  - Each node acts as client, server or both.
  - Node must join P2P network.
    - Registers its service with central lookup service on network, or
    - Broadcast request for service and respond to requests for service
- Examples include Napster and Gnutella, **Voice over IP** (**VoIP**) such as Skype, torrents.

Computing Environments - Virtualization

- Allows whole operating systems to run as applications within other operating systems.
  - Vast and growing industry.
- **Is a form of emulation,** used when source CPU type different from target CPU type.
  - Some time back, Apple changed CPU architecture, from PowerPC to Intel x86.
  - Included a utility called “Rosetta” to let users run PowerPC programs on Intel.
  - Generally, this method is slow.
  - Every instructions needs to be translated from target -> source architecture.
  - Examples: Nintendo emulators (Android), Palm emulator (Android), DOS Box (Linux, Mac OS).

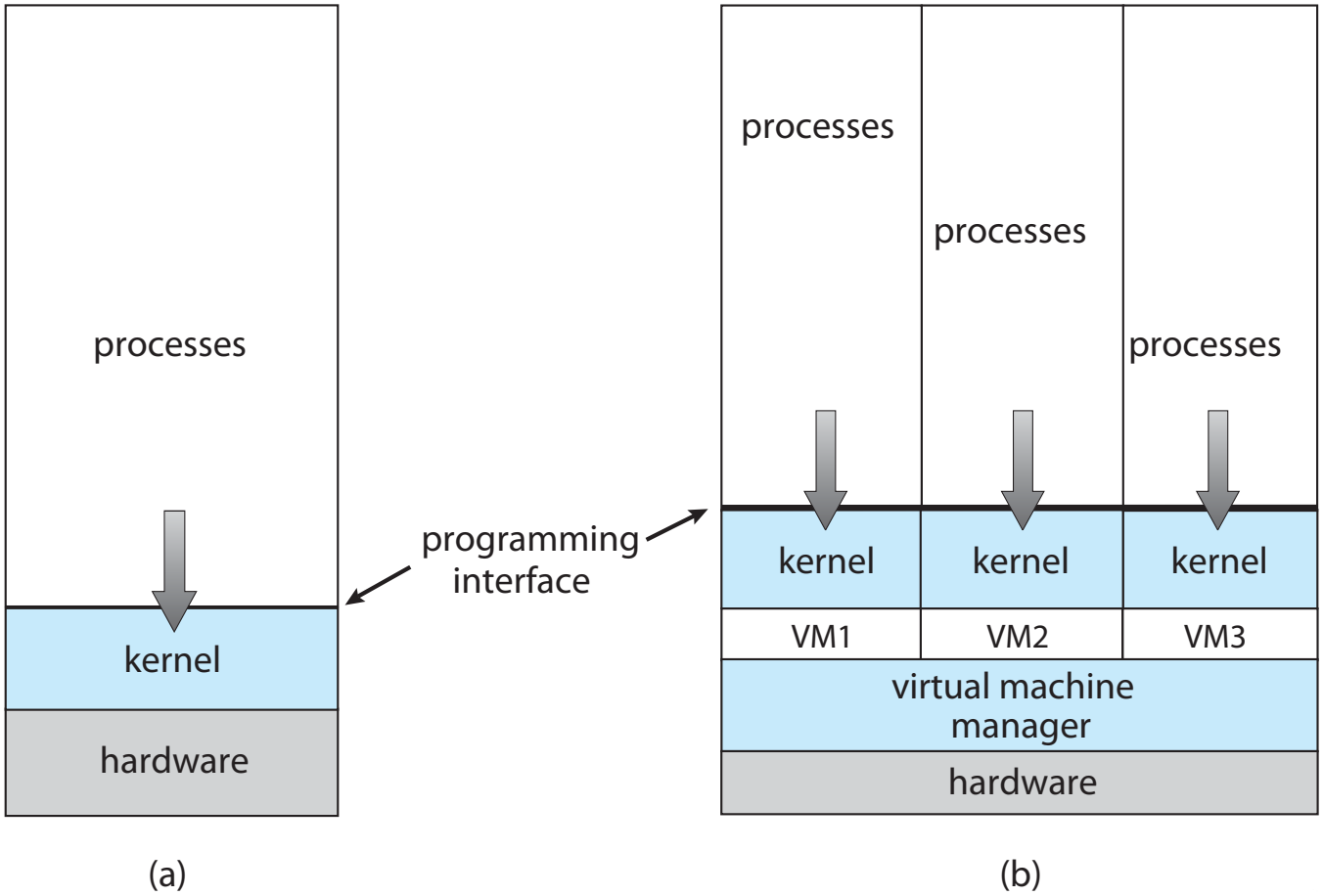
Emulators for Nintendo and Palm on Android



Virtualization - II

- **Virtualization** – OS natively compiled for CPU, running **guest** Oses, which may also be natively compiled.
  - Same set of hardware shared among multiple OSes.
  - To enable a computing environment to run multiple independent systems at the same time.
  - Both OSes need same hardware to run.
  - Consider VMware running WinXP guests, each running applications, all on native WinXP **host** OS: many users using same system.
  - This was how virtualization started initially, at IBM in the 1960s and 1970s.
  - **VMM** (virtual machine Manager) software provides virtualization services.
  - It has an application that runs on the host OS.
  - Over it, many copies of guest OSes can be run.
  - Next slide shows a picture.

Computing Environments - Virtualization



Computing Environments - Virtualization

- Use cases involve laptops and desktops running multiple OSes for exploration.
  - Apple laptop running Mac OS X host, Windows as a guest.
  - Developing apps for multiple OSes without having multiple systems.
  - Quality Assurance testing applications without having multiple systems.

Computing Environments – Cloud Computing

- **What is cloud computing?**
- **Delivers computing, storage, even apps as a service across a network.**
- Any example of cloud based applications that you can think of?
- Logical extension of virtualization because it uses virtualization as the base for it's functionality.
  - Example: Amazon **Elastic Compute Cloud 2 (EC2)** has thousands of servers, millions of virtual machines, petabytes of storage available across the Internet, pay based on usage.
  - Another example is: Microsoft Azure.
- Many types of clouds in the market:
  - **Public cloud** – available via Internet to anyone willing to pay.
  - **Private cloud** – run by a company for the company's own use.
  - **Hybrid cloud** – includes both public and private cloud components.
  - Software as a Service (**SaaS**) – one or more applications available via the Internet (i.e., word processor).
  - Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e., storage available for backup use).
- Advantage of cloud computing? Disadvantage?

Computing Environments – Real-Time Embedded Systems

- Embedded systems are very common today.
- It is a “computer system with a dedicated/specialized function within a larger mechanical or electrical system” – source: Wikipedia.
- Examples: microwave ovens, music players, parts of automobiles, sensors & so on.
- Many other special computing environments as well
  - They have simple OSes, with little or no user interface.
- Usage expanding with the internet (e.g. a sensor network in a smart home). New field -> Internet of Things (IoT)
- Many times, embedded systems run a Real-time OS, that has well-defined fixed time constraints
  - Processing ***must*** be done within constraint.
  - Correct operation only if timing constraints met.

Real-Time Systems

- Examples:
  - Braking system of an airplane.
  - Sensor network that warns about enemy movement in a military application.
- Need the results of computation in time, else, we may have a disaster.
- In a real-time OS, processes must be completed within a deadline, else, the usefulness is zero.
- Note that in non real-time systems, it is desirable (not essential) to finish tasks early.
- Also note that a real-time is not the same as an “as fast as possible system”.
- Such systems require special scheduling algorithms, as we shall see in a later chapter.

1. Process Management

- What is a process?
- 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.
- Process termination requires reclaim of 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 (activity monitor/task manager)
  - Concurrency by sharing 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 processes (user + system).
- Suspending and resuming processes.
- Scheduling processes and threads.
- Providing mechanisms for process synchronization.
- Providing mechanisms for process communication.

2. Memory Management

- Memory management determines what is in memory and when.
  - Goal is: 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 de-allocating memory space as needed.

3. Storage Management

- OS provides uniform view of information storage.
  - Abstracts physical properties to logical storage unit - **file.**
  - Each storage medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method.
- 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.
    - Disk scheduling
    - Mounting and unmounting directories.
    - Storage allocation.
    - Backup files onto stable (non-volatile) storage media.

IBM Disk from 1956 (business insider)

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

4. I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user.
  - Provide users with a simple interface for performing I/O.
- 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).
  - Device drivers for specific hardware devices, that provide the above interface.
- Chapter 13 discusses how the I/O system:
  - Interfaces to other parts of the OS.
  - Manages I/O devices.
  - Detects I/O completion.

Open Source Operating Systems

- Two broad categories of operating systems:
  - Windows – closed source.
  - Linux – open source.
- What do we mean by open source?
- They are available in source code format (e.g. C).
- Hence, they can be changed & compiled.
- Can also be copied and distributed for free.
- Closed source, on the other hand, is available in compiled binary code.
- Hard to derive source code from it.
- Apple's OSX has a hybrid approach.
  - Kernel (called Darwin) is open source.
  - Some components are close source.

Open Source Operating Systems

- Advantage of open source operating systems?
- Ideal for learning operating systems.
  - Can go into the detailed code of each module.
- Anyone can make a contribution to the OS to make it better.
  - For example: improve a scheduling algorithm, cache policy.
- They are generally free.
  - For original OS & upgrades as well.
- Dedicated community of programmers, online forums to help out users.
- Arguably more secure.
  - Since source code is in public domain, anyone can suggest changes that enhance the security of the OS.

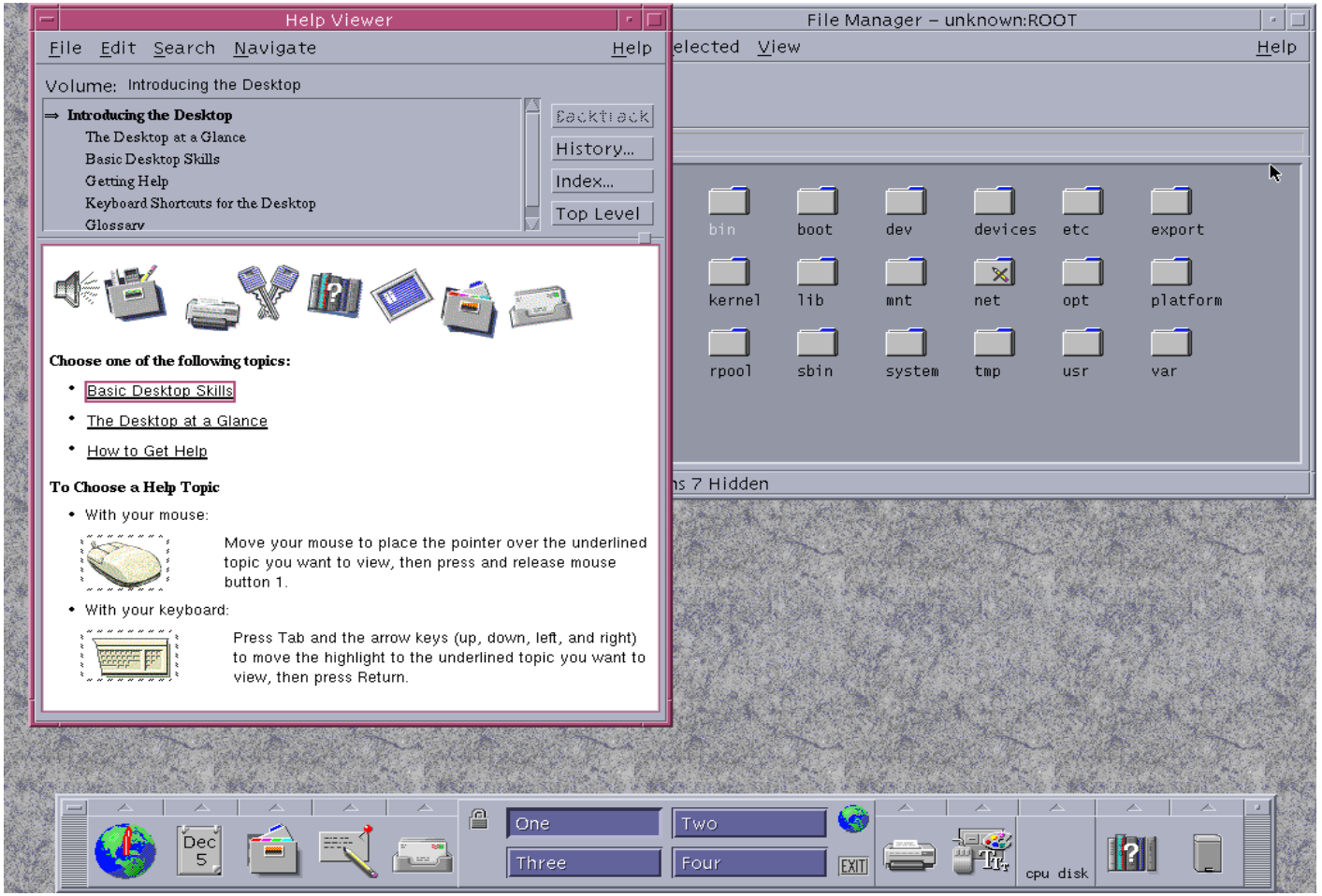
A Bit of History of Open Source

- In the 1950s, MIT had a Tech Model club, that used to write programs & leave them in drawers, inviting other members to look at the code & suggest changes.
- Companies, such as the Digital Equipment Corporation (DEC) accepted contributions from people for source code, collected such programs, & distributed them to interested users.
- Companies such as Microsoft, Apple did not like this. Why?
- They wanted to make money of the OS.
- Initial effort to design the OS.
- Later, only need to make copies of it.
- Some tied the OS to the hardware.
- To use iOS, need to buy an iPhone.
- For Android, there are many phone options.

Open Source Operating Systems

- **Linux**, started by Linus Torvalds, a research student from Finland.
- Comes in many flavors – Red Hat, Fedora, Debian, Ubuntu.
- Many Linux projects are provided at: http://distrowatch.com.
- Can be run on a Windows/Apple machine using virtualization.
- Install Vmware player or Virtual Box tool.
- Download a Linux distribution.
- Run it as an application.
- **Unix**, on which Apple's OS is based.
- Some open source & some closed source parts.
- Darwin kernel is available for download at: https://opensource.apple.com.
- **Solaris**, by Sun Systems.
- This is another Unix variant.

Solaris OS (Wikepedia)



Open Source Software

- Open Office: equivalent of Word, Excel, Power Point.
- Emacs : programming editor.
- Mozilla Firefox: web browser.
- My SQL: database.
- Scilab: equivalent of Matlab, for numerical computation.
- VLC player: music, videos.
- Gimp: image editor.
- Many more.

Open-Source Operating Systems - Recap

- Operating systems made available in source-code format rather than just binary **closed-source**
- 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

Summary of Chapter 1

- The OS is a software that – (a) manages the hardware resources and (b) provides a convenient environment for users to run programs.
- Some broad goals of a general purpose OS: convenient user interface, efficient utilization of resources.
- OS has many jobs: process management, memory management, storage management, I/O management, protection & security.
- A (very) brief overview of a computer system and various architectures: single CPU, multiple CPU, distributed, peer-to-peer, cloud, real-time, multimedia, mobile.
- An introduction to some OS features – multiprogramming/multitasking, dual-mode (user & kernel).
- Discussed various architectures: uniprocessor, multiprocessor, distributed.
- Discussed various computing environments: traditional, mobile, cloud.
- Talked about open source operating systems, mostly Unix/Linux.

End of Chapter 1