Bilkent University
Department of Computer Engineering
CS342 Operating Systems

Introduction - 1

Last Update: Feb 3, 2023

Outline and Objectives

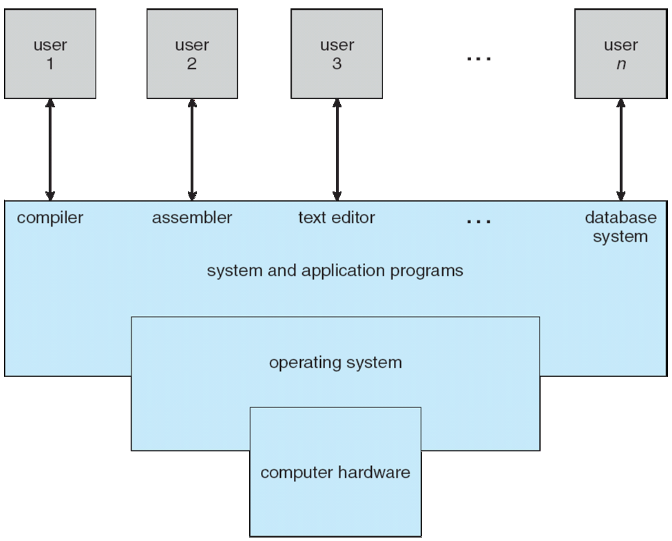
Outline

- What Operating Systems Do
- Computer-System Organization
- OS structure and operation
- Major OS Functions
  - Process Management
  - Memory Management
  - Storage Management
  - Protection and Security
- Computing Environments

Objectives

- To provide a grand tour of the major operating systems components
- To provide coverage of basic computer system

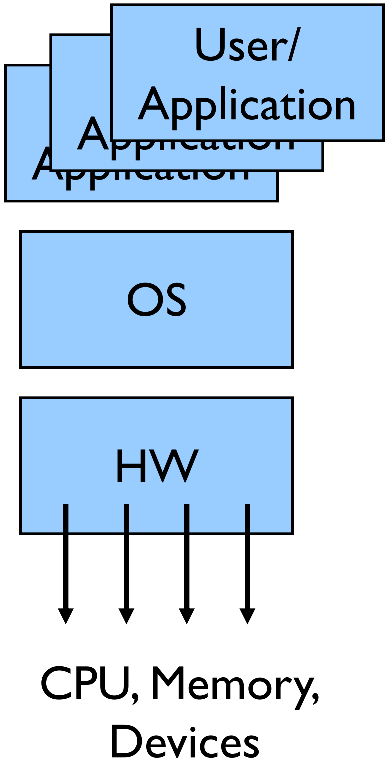
Basic components of a computer system: place of OS



- A computer system can be divided into four components
  - Hardware – provides basic computing and storage resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs –solve the problems of the users: use system resources
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers

What is an operating system?

- A program that manages hardware and acts as an *intermediary* between a users/applications and the hardware
- Operating system functionalities
  - Start, terminate, control executing user programs
  - Make system convenient to use
  - Control and coordinate use of hardware
    - Perform and manage I/O; setup devices
    - Manage and allocate resources
    - Use hardware efficiently
  - Implement common services

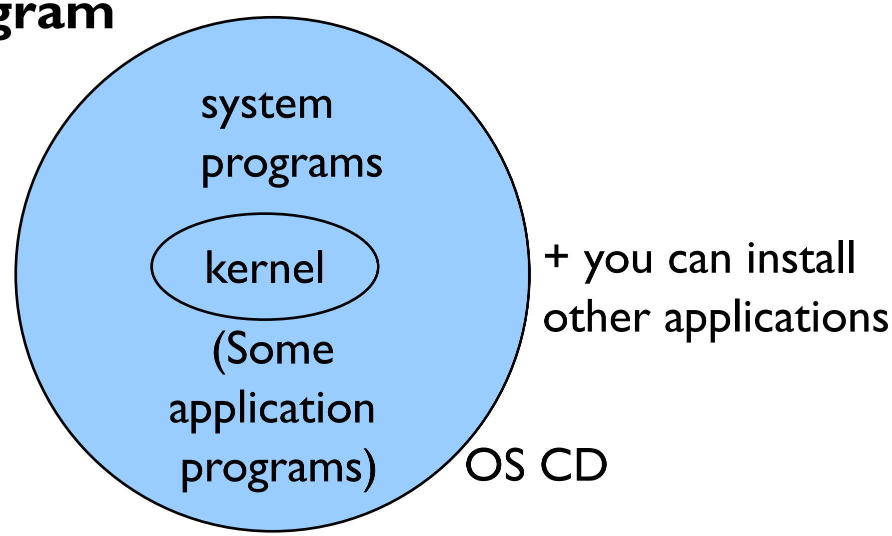


"operating system": *software responsible for proper "operation" of the computer.*

Operating System Definition (as a software)

- No universally accepted definition
  - "*Everything a vendor ships when you order an operating system*" is good approximation
    - But varies wildly.
- **Kernel:** running all the time; having most of the functionality
- Everything else: either a **system program** (associated with the operation of the system) or an **application program**

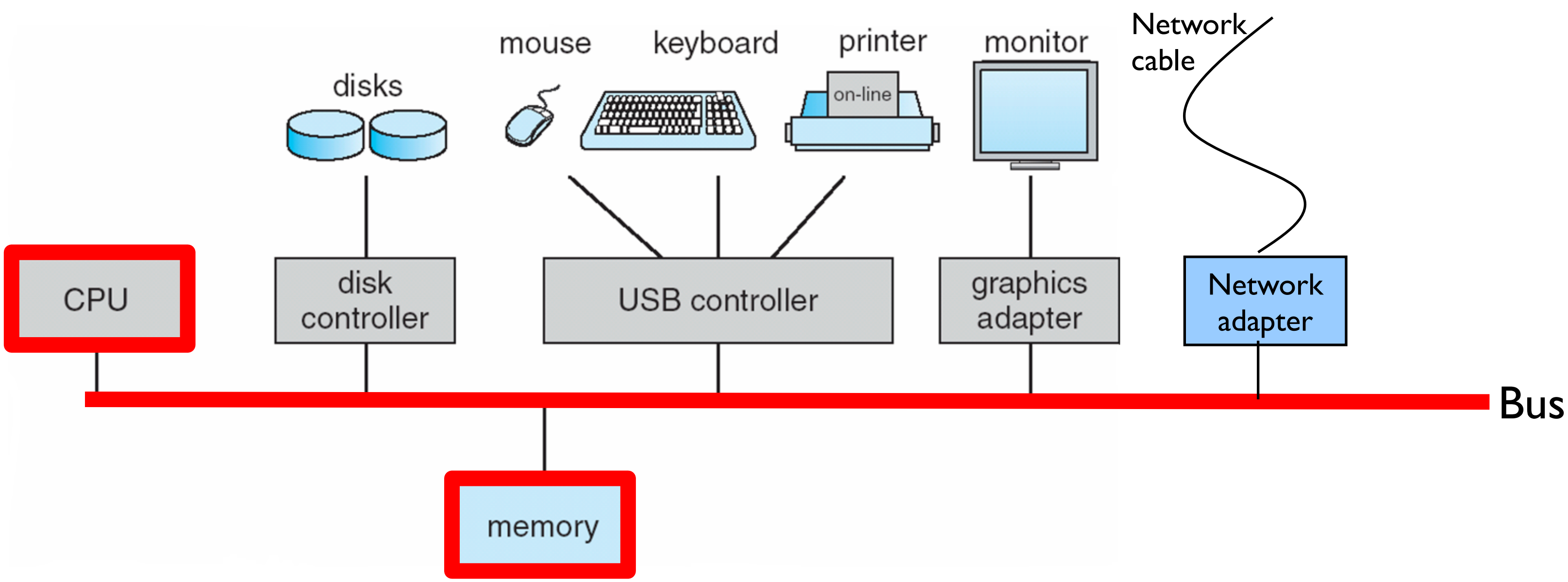
System programs: programs that are associated with the operating system.



Computer System Organization and Operation

Computer System Organization

- Computer-system operation
  - One or more **CPUs**, **device controllers** connect through **common bus** providing access to **shared memory**
  - **Concurrent execution** of CPUs and Device Controllers competing for memory cycles



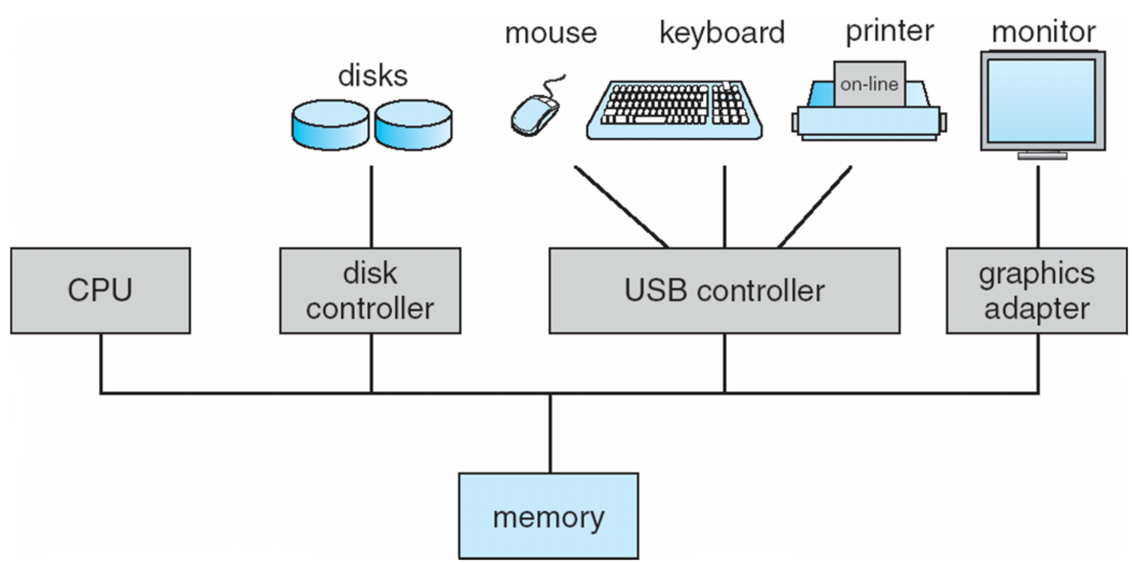
Computer Startup

- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM,
  - Generally known as **firmware**
  - Initializes all aspects of the system
  - Loads operating system **kernel** into memory and starts its execution.
- **Kernel** runs and make the system ready for running applications.
  - Kernel is always ready to run (always in memory)

Computer system operation: I/O and device interaction

- I/O devices (and controllers) and the CPU can execute concurrently
- Each device controller has a local buffer
  - Data movement (I/O) between **device** and **local buffer** (by device)
  - Data movement between **memory** and **local buffer** (by CPU)

Device **controller** informs CPU that it has finished its current output operation or it has some input data by causing an **interrupt.**

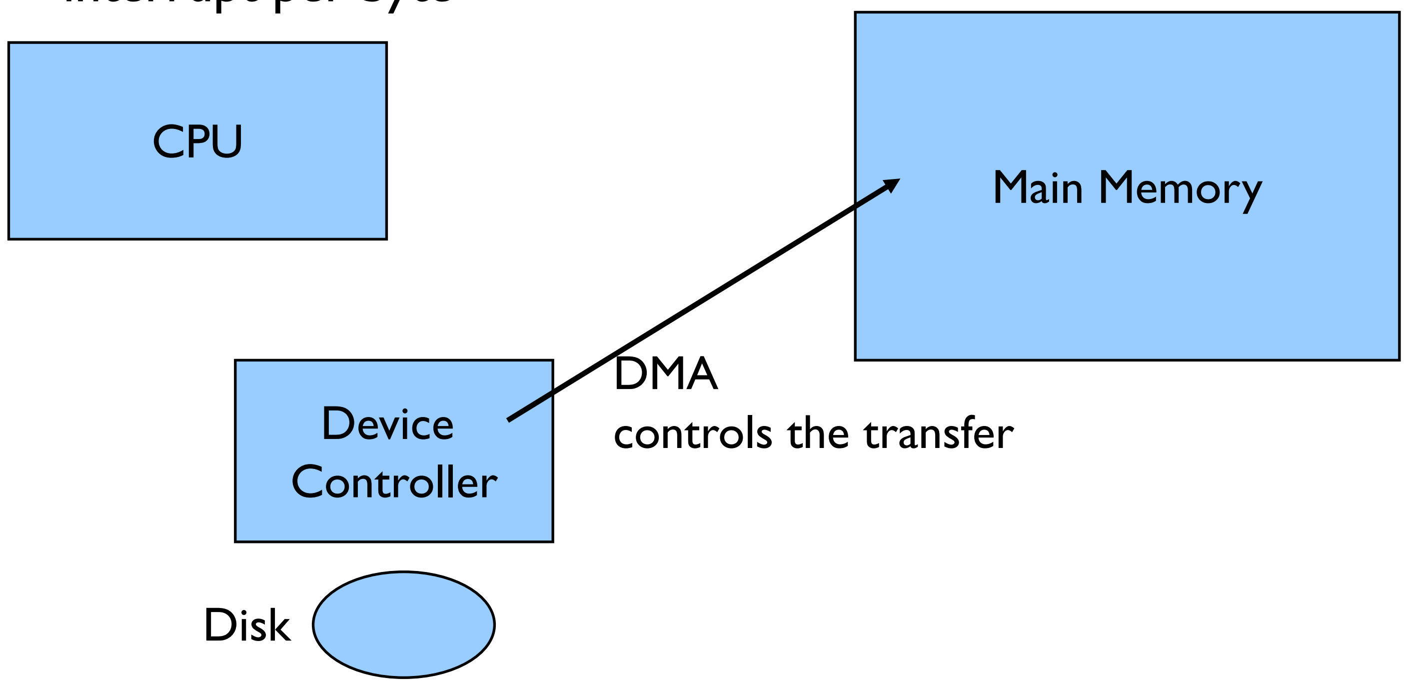


Hardware interrupts

- When interrupt occurs, hardware does the following:
  - CPU is interrupted
    - at that time application code or kernel code might be running
    - registers and the program counter saved into RAM to preserve CPU state
  - CPU starts running the respective Interrupt Service Routine (ISR)
    - ISR is a kernel routine
    - ISR is found through interrupt vector: a table containing addressed of ISRs

Direct Memory Access Structure

- With DMA, device controller transfers blocks of data from device buffer **directly** to main memory without CPU intervention
  - Only **one interrupt is generated per block**, rather than one interrupt per byte

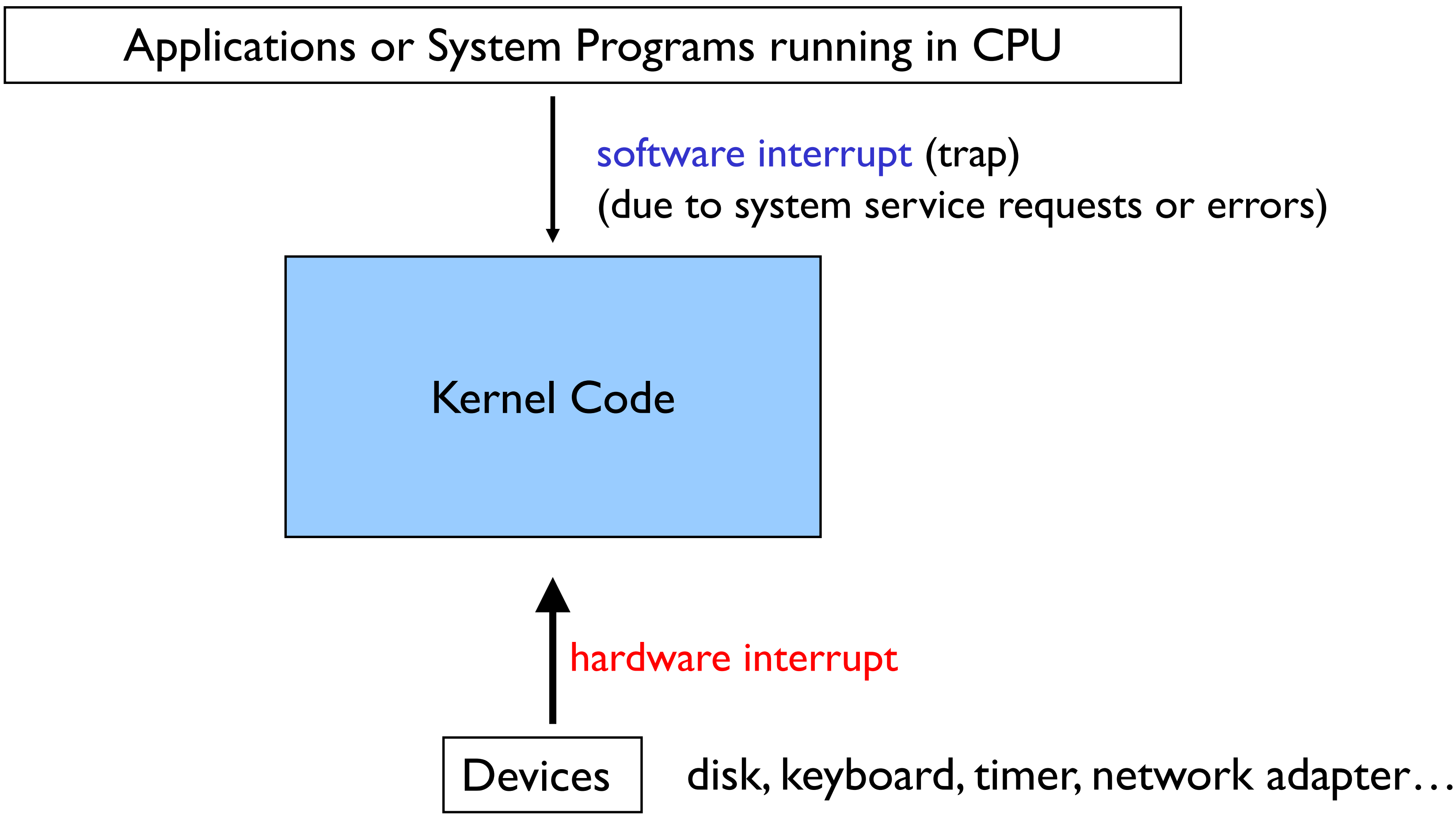


Software interrupts

- Running application software (executing program) may generate interrupts as well.
  - They are called software interrupts (trap)
    - 1. Exceptions (caused by errors, such as division by zero)
    - 2. System calls (service request)
      - syscall (or trap) instruction is used

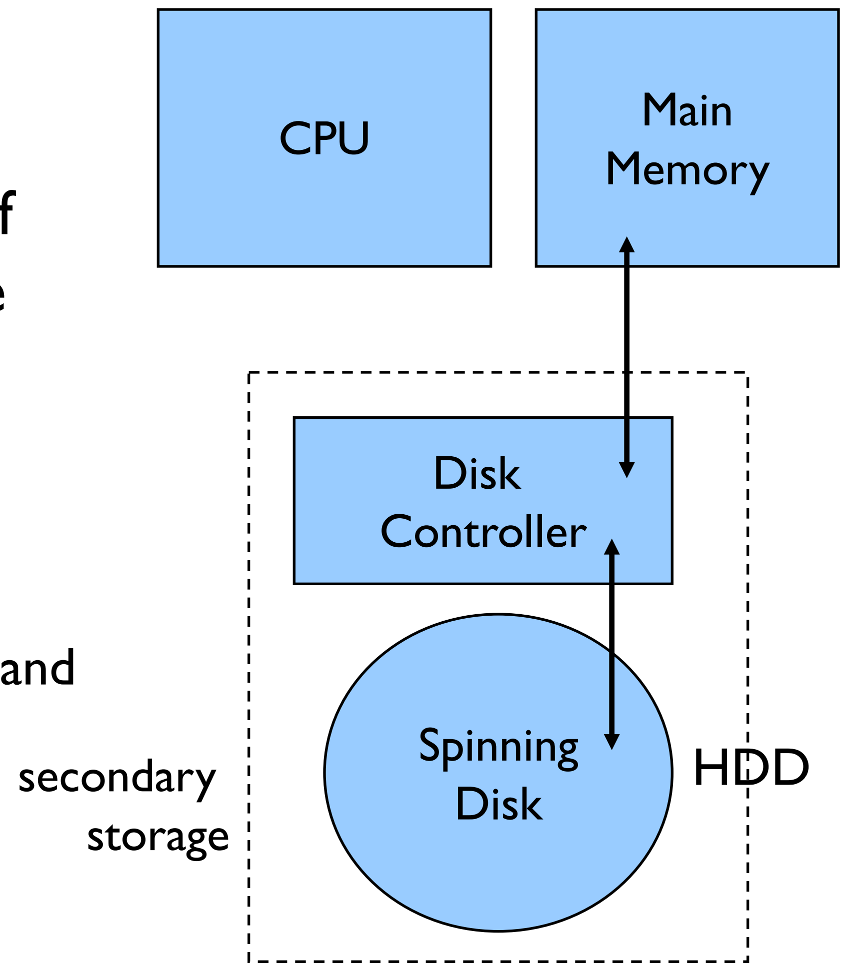
- *An operating system (kernel) is interrupt-driven (event driven)*

Interrupt-Driven OS



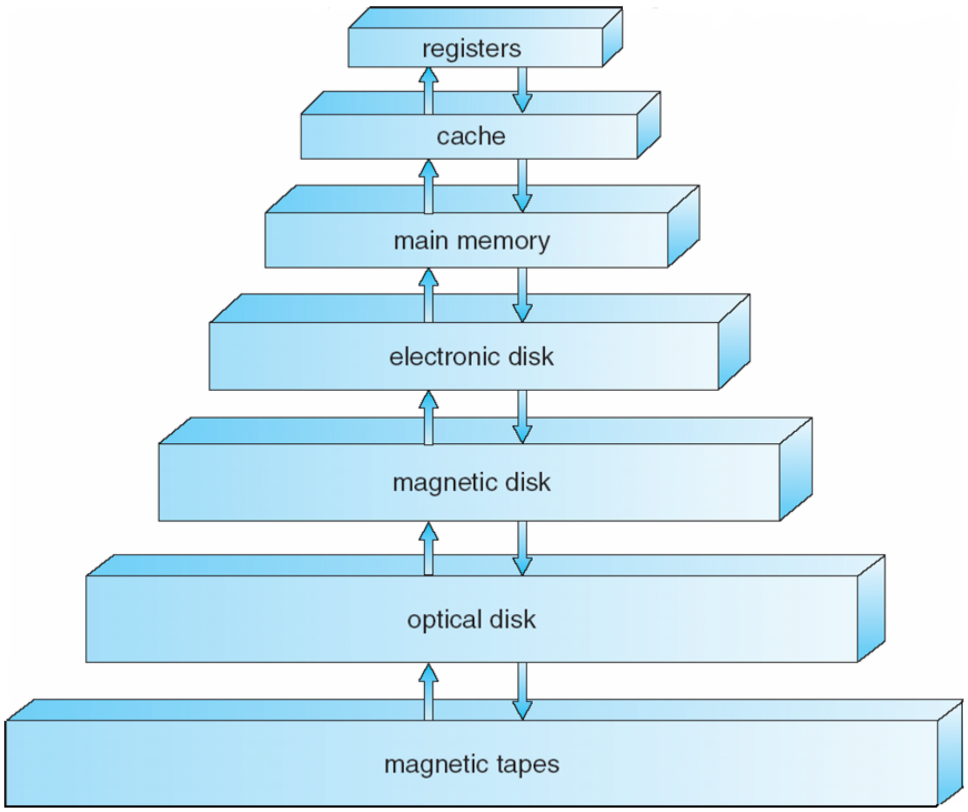
Storage Structure

- Main memory (primary storage)
  – CPU can access directly
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity
  - Magnetic disks (HDD disks)
    - Platters
    - The disk controller handles the interaction between the device and the computer
  - SSD disks



Storage Hierarchy

- Storage systems organized in hierarchy
  - They differ in
    - Speed (ms)
    - Cost ($)
    - Capacity (GB)
    - Volatility

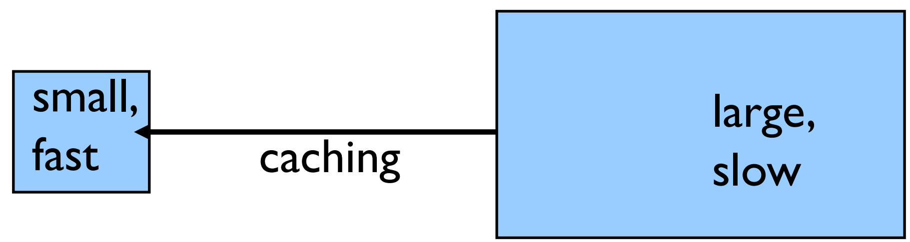


Storage units

- Bit: 0 or 1 (one of two value)
- Byte: 8 bits
- Word (architecture's native unit of data – e.g., 1 word = 4 bytes)
- 1 KB = 1024 bytes = 2^10 bytes
- 1 MB = 2^20 bytes = 1024^2 bytes
- 1 GB = 2^30 bytes = 1024^3 bytes
- 1 TB = 2^40 bytes = 1024^4 bytes
- 1 PB = 2^50 bytes = 1024^5 bytes

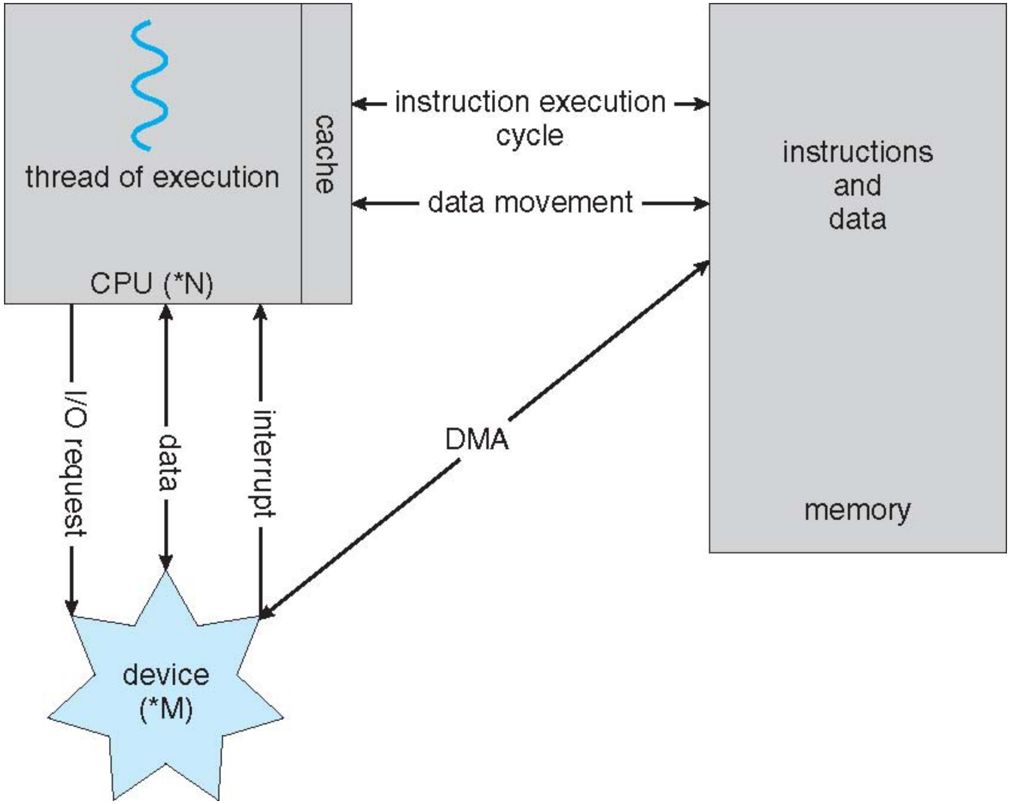
Caching

- Caching – copying information into faster storage
  - there is tradeoff between size and speed of storage devices.
- Performed at many levels in a computer
  (at hardware, operating system, or application level)
- Cache is checked first for an item.



Registers	Main Memory
Hardware cache L1, L2, etc.	Main Memory
Main Memory	Hard Disk
Hard Disk	Remote Internet Servers

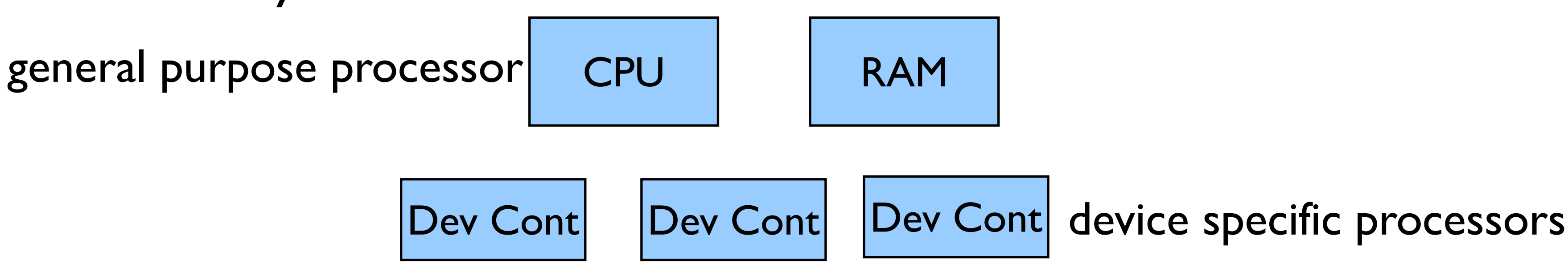
Interplay of all Hardware Components



Computer System Architecture

Single processor systems

- A lot of systems use a single general-purpose processor (CPU) or a limited number of CPUs
  - Most systems have special-purpose processors as well
- CPU is capable of executing a general purpose instruction set, including instructions from user programs.
- Computers have device-specific processors as well.
  - They don't run user programs.
  - Some may be commanded/managed by the operation system.
    - By the device drivers.

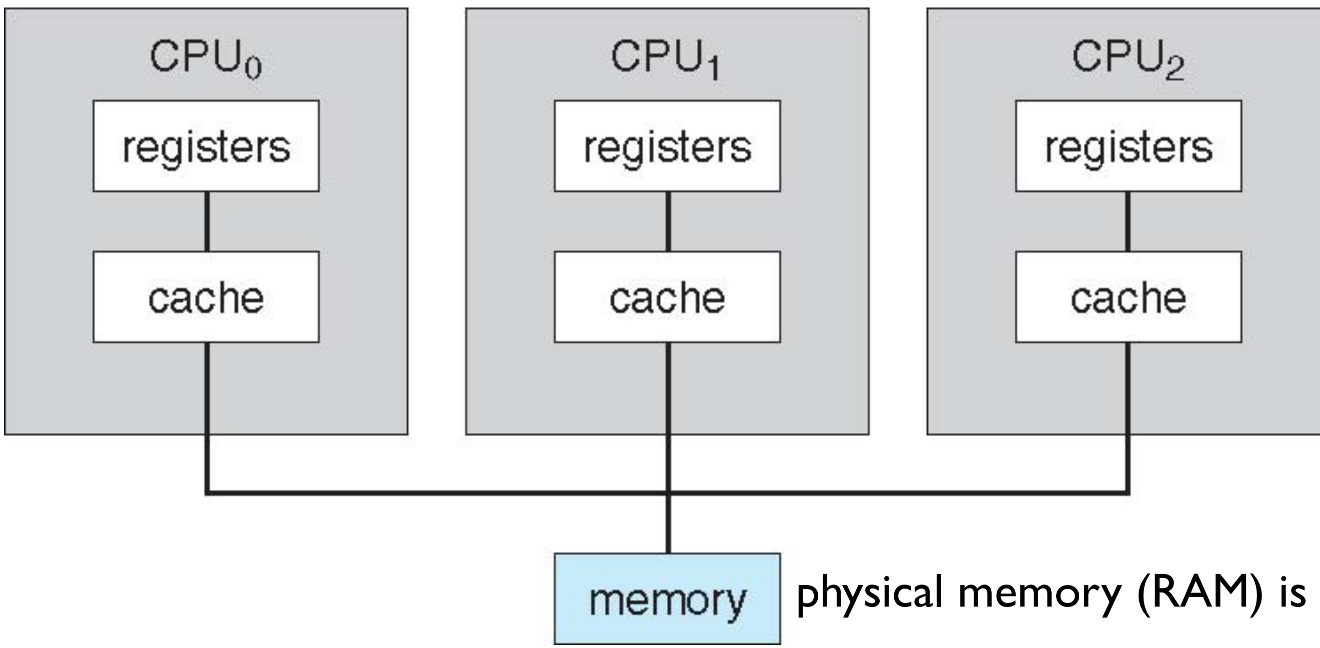


Multiprocessor Systems

- Multiprocessor systems growing in use and importance
  - They are parallel systems
  - Tightly-coupled systems
  - Advantages include
    - Increased throughput
    - Economy of scale (cheaper than using multiple computers)
    - Increased reliability – graceful degradation or fault tolerance
  - Two types of multiprocessor architecture
    1. Asymmetric Multiprocessing
    2. Symmetric Multiprocessing (SMP) (very common)

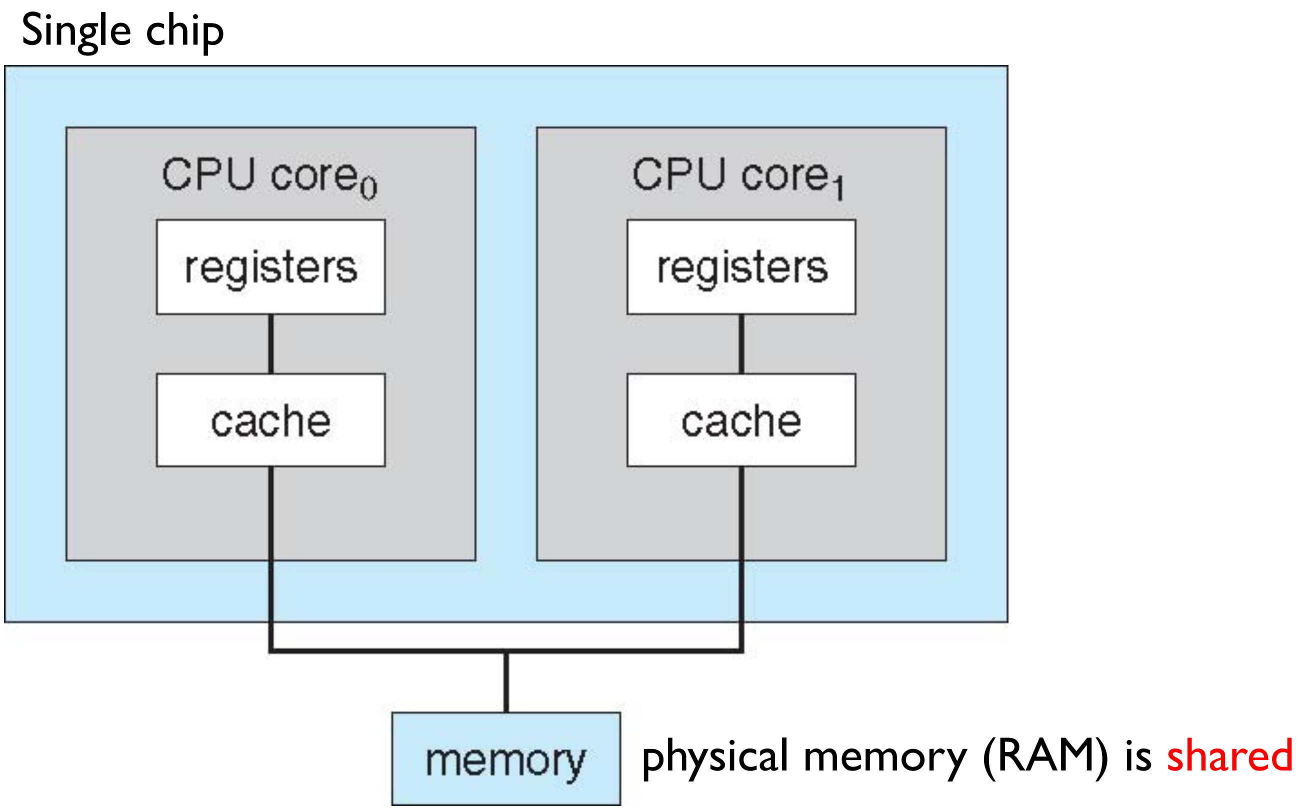
Symmetric Multiprocessing Architecture (SMP)

Each CPU has equal role
(can execute a user program or kernel)



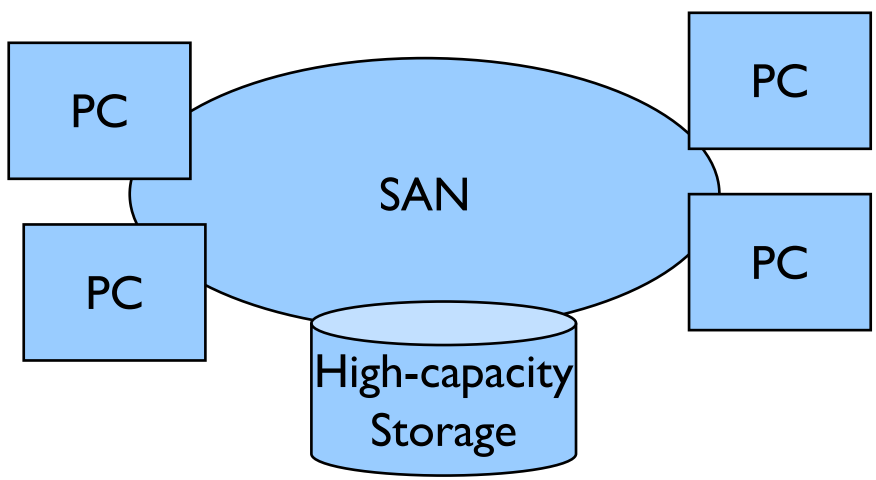
physical memory (RAM) is shared

A Dual Core Design



Clustered Systems: multicomputers

- Like multiprocessor systems, but **multiple systems** working together
  - Usually sharing storage via a storage-area network (SAN)
  - Provides a **high-availability** service which survives failures
    - Asymmetric clustering has one machine in hot-standby mode
    - Symmetric clustering has multiple nodes running applications, monitoring each other
  - Some clusters are for high-performance computing (HPC)
    - Applications must be written to use parallelization

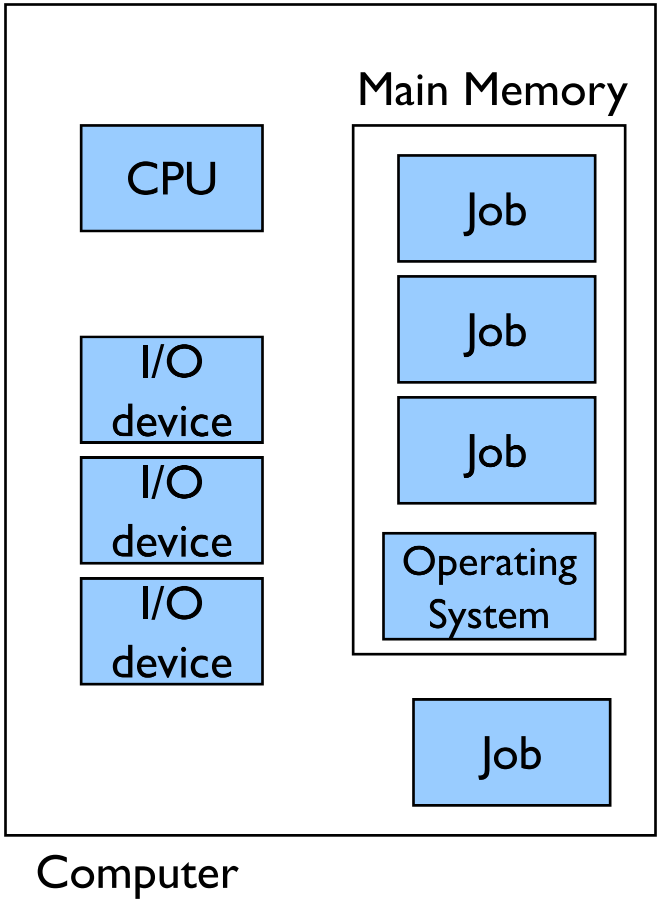


Storage

Operating System Operations

Operating Systems: providing multiprogramming

- Multiprogramming: multiple programs can be started and loaded.
- A subset of total jobs in system is kept in memory.
- It is convenient
- It is efficient:
  - Single user cannot keep CPU and I/O devices busy at all times
- – One job selected and run via job scheduling
  - OS selects which job
  - When the job has to wait (for I/O for example), CPU is given to another job.
- "job", "process", "running program" will be used interchangeably.



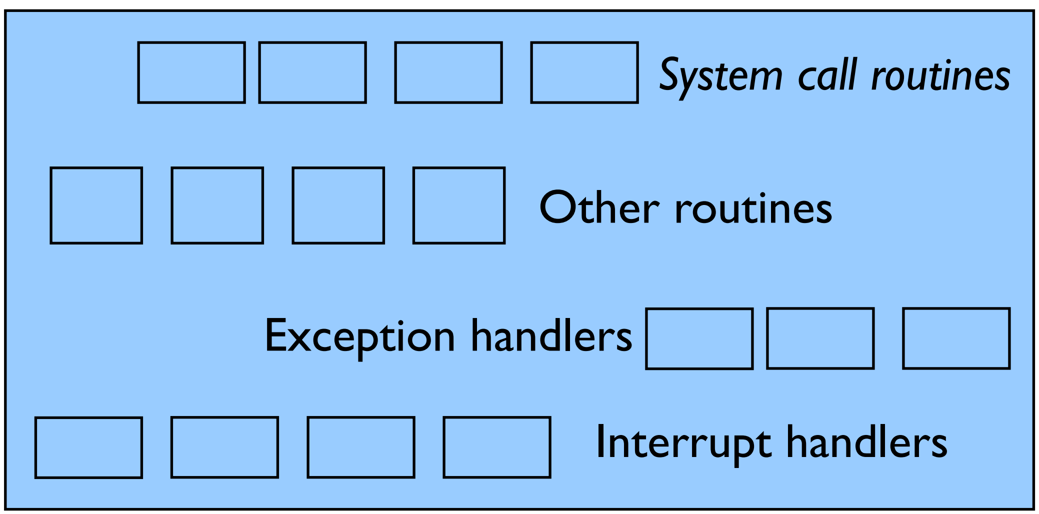
Operating Systems: providing time sharing

- 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
  - program loaded in memory ⇨ process
  - If several processes ready to run at the same time ⇨ CPU scheduling

Operating System execution

- OS system kernel is interrupt driven
  - Hardware interrupt causes ISR to run (which is a routine of OS)
  - Software error or system request (system call) causes exception handler or system call handler to run
    - example: “division by zero” (exception)
    - Example: request for an operating system service (“open a file”) (system call)

OS Code
(Kernel Code)



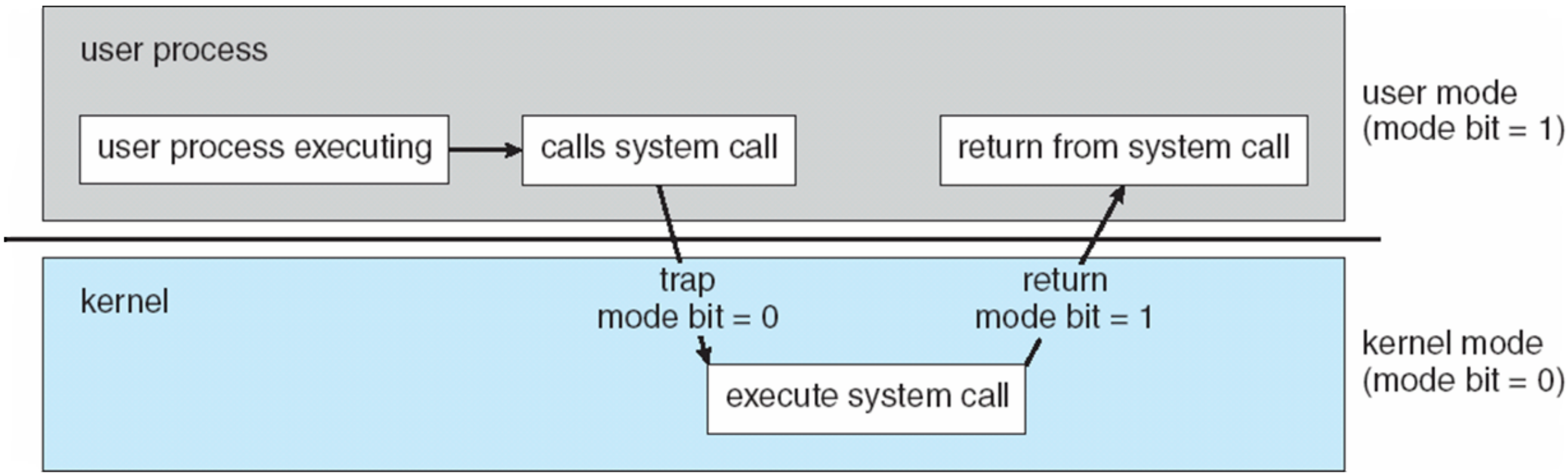
Dual mode operation

- Dual-mode operation (a hardware property) allows OS to protect itself and programs and other system components
- CPU can run in one of two (at least) modes:
  - User mode and kernel mode
  - Mode bit provided by hardware (CPU)
- User code runs in user mode;
- Kernel code runs in kernel mode.
- Some machine instructions designated as privileged/special, only executable in kernel mode; other instructions are normal instructions.
- In kernel mode, where all instructions (normal + privileged) can be executed.
- In user mode, only normal instructions are allowed to execute.

Dual mode operation

Dual mode system operation

Transition from User to Kernel Mode and Vice Versa



system-call instruction *(executed by user program)* changes the mode to kernel mode

return-from-system-call *(executed by the kernel)* instruction resets the mode to user mode

Periodic timer interrupts

- Timer device to prevent an infinite loop / process hogging resources
  - 1) Set the timer device to interrupt after a while later
    - Can be a fixed (for example 10 ms) or variable time period
  - 2) CPU executes a program (a process)
  - 3) Timer device sends an interrupt after that period
  - 4) CPU starts executing timer handler: OS gains control
  - 5) OS can schedule the same process or other process
  - 6) OS sets the timer again before giving the CPU to the scheduled process

Major OS Functionalities

- Process management
- Memory management
- Storage (HDD or SDD) management
  - File concept, file mapping to disk blocks, disk scheduling
- I/O control and management
  - Device derivers (doing I/O), buffering, providing uniform access interface
- Protection and security
  - Controlled access to resources,
  - Preventing processes interfering with each other and OS

Process Management

- A process is a program in execution. (unit of work) (active)
- Process executes instructions sequentially, one at a time, until completion
- Process needs resources
  - CPU, memory, I/O, files
- Typically system has many processes running concurrently
  - Some of them may be OS processes
- Upon termination, resources are released

For process management:

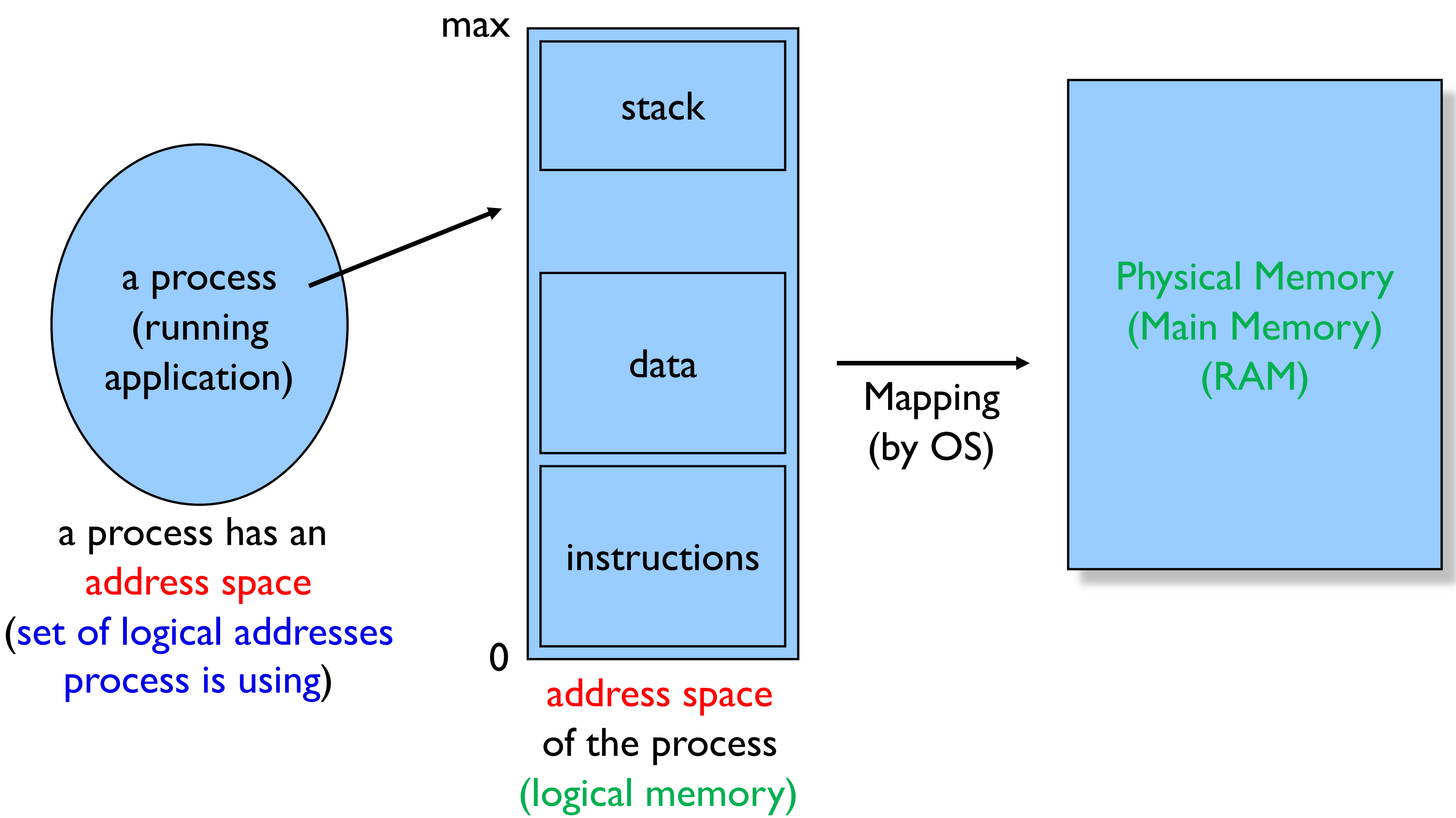
- Creating and deleting both user and system processes and
- Suspending/resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory, where and when

- Memory management activities
  - Keeping track of which parts of memory are currently being used and by which program (process)
  - Deciding which processes (or parts of a process) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Process Address Space



File-System Management

- OS provides uniform, logical view of information storage
  - Abstracts physical storage to logical storage unit (a file)
  - Various storage device types varying in medium type, 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;
  - Primitives to manipulate files/dirs;
  - Mapping files onto secondary storage

Mass-Storage Management

- Mass Storage:
  HDD disks, SDD disks (secondary storage);
  CDs, tapes, etc. (tertiary storage)

- Proper management of mass storage devices is of central importance
  - For improving performance of the computer system
  - Since they are slow devices.

- OS activities
  - Free-space management; Storage allocation
  - Disk scheduling
  - Uniform naming ….

Performance of various levels of storage

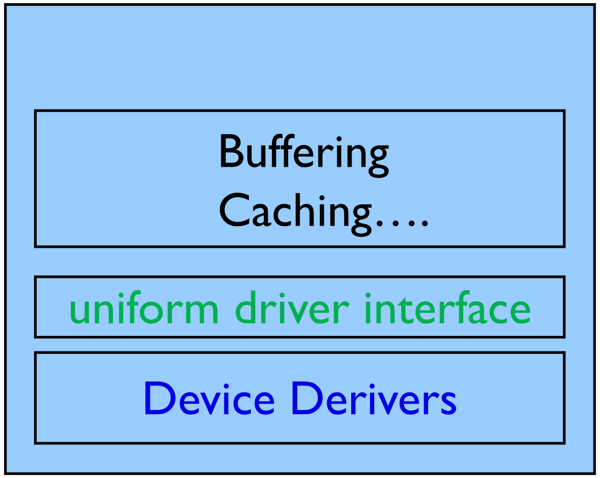
- Movement between levels of storage hierarchy can be explicit or implicit.

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

Input/Output Subsystem

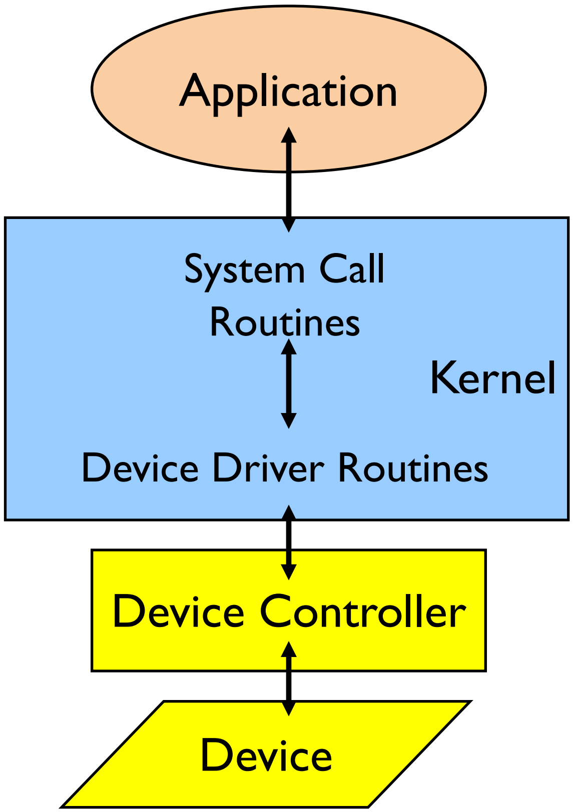
- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Buffering, caching,
  - General device-driver interface
  - Drivers for specific hardware devices
    Interacting with the device and doing I/O

I/O sub-system of Kernel



I/O Structure

- Application programs do I/O via OS.
  - The request is done by calling a system call (OS routine)
  - System call routine in OS performs the I/O via the help of device driver routines in OS.
  - After issuing a system call, an application
    - may wait for the call to finish (blocking call), or
    - may continue to do something else (non-blocking call)



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 of the user is then associated with all the files and processes of the user to do access control

Kernel Data Structures

- Lists
  - Singly linked lists
  - Double linked lists
  - Circular linked lists
- Queues
- Stacks
- Trees
  - Binary search tree
  - Balanced binary search tree (red-black tree)
- Hash functions and hash tables
- Bitmaps

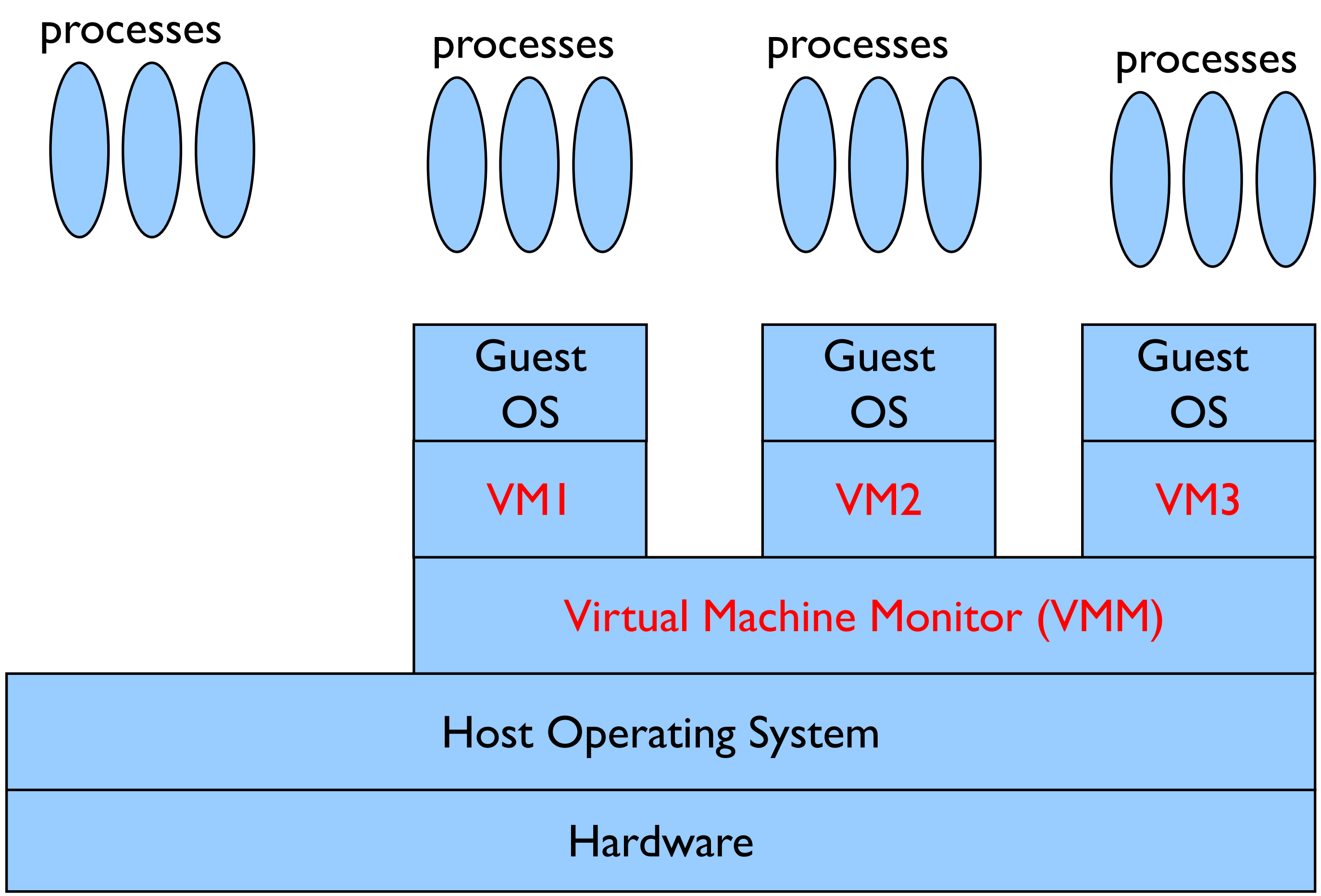
LINUX KERNEL DATA STRUCTURES

The data structures used in the Linux kernel are available in the kernel source code. The *include* file `<linux/list.h>` provides details of the linked-list data structure used throughout the kernel. A queue in Linux is known as a `kfifo`, and its implementation can be found in the `kfifo.c` file in the `kernel` directory of the source code. Linux also provides a balanced binary search tree implementation using *red-black trees*. Details can be found in the include file `<linux/rbtree.h>`.

Virtual Machines

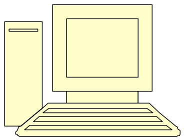
- Hardware is abstracted into several different execution environments
  - Virtual machines
- Each virtual machine provides an interface that is identical to the bare hardware
- A *guest* kernel (and processes) can run on top of a virtual machine.
  - We can run several operating systems on the same *host*.
  - Each virtual machine can run another operating system.

Virtual Machines

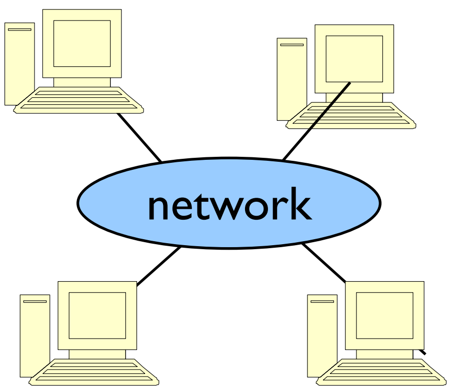


Different Types of Computer Systems and Applications (Computing Environments)

Distributing Computing



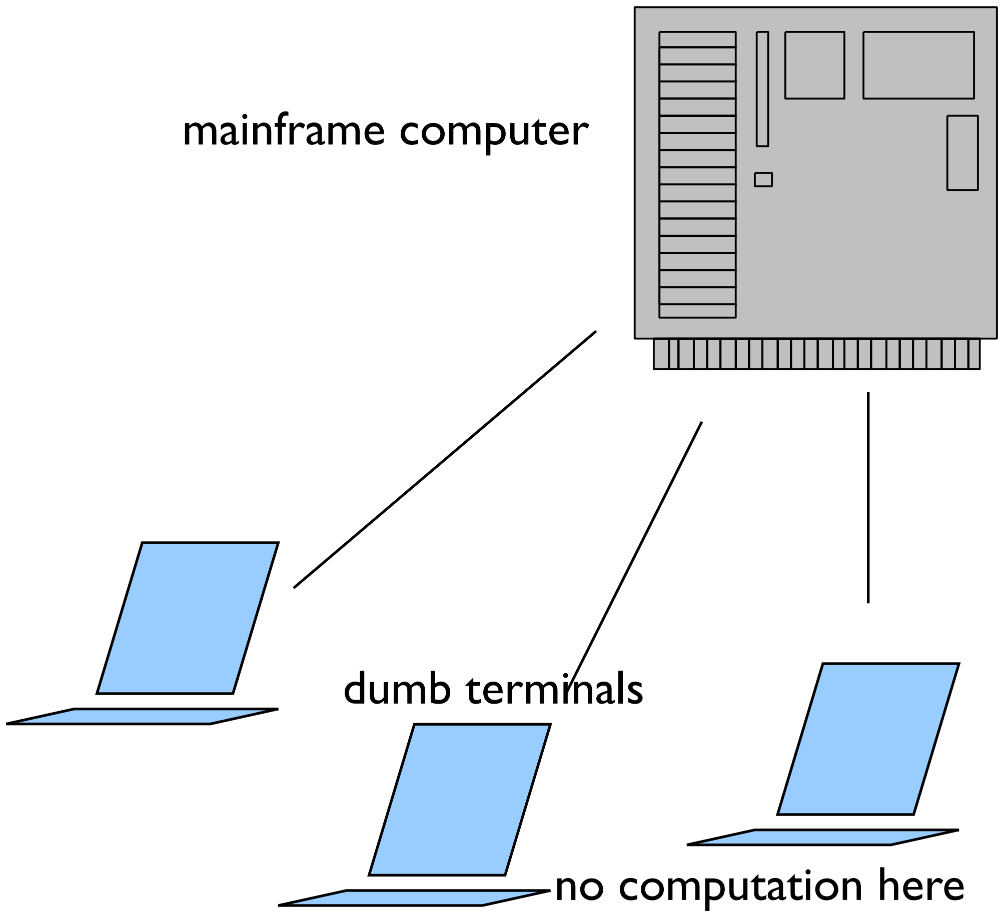
- Earlier systems executed tasks on a single system
- Now we have systems interconnected (networked) Operating systems have now support for networking multiple systems,
  - enabling data communication
  - Enabling distributed computing
  - Enabling resource sharing
  - Enabling distributing file storage
- Therefore, the computing environment is no longer a single system.



Computing Environments

- Traditionally

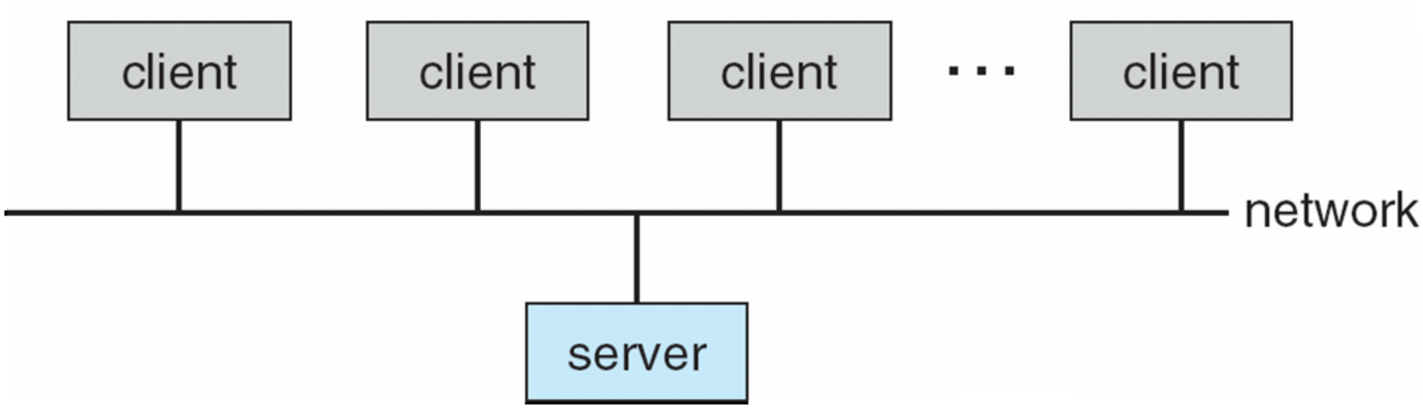
a single system with a user



Computing and OS
in a single machine

Computing Environments

- Client-Server Computing
  - Dumb terminals replaced by smart PCs
  - Many systems now are servers, responding to requests generated by clients
  - A compute-server provides an interface to clients to request services (i.e., database) executed in the server
  - File-server provides interface for clients to store and access files

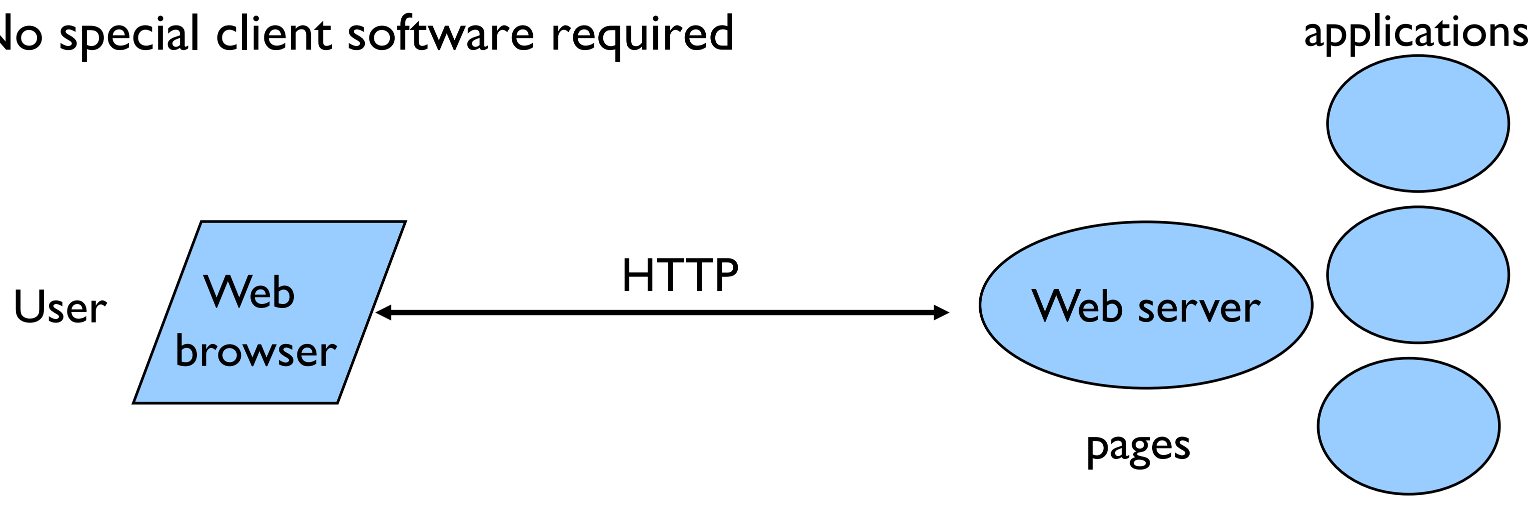


Peer-To-Peer Computing

- Another model of distributed system.
- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - Each may act as a client, as a server, or both
  - A node must join P2P network
  - A peer registers its service with central lookup service on network, or
  - Peer broadcasts requests for service and the serving peer(s) responds (resource discovery/lookup protocol)

Web Based Computing

- Web has become ubiquitous
- More devices becoming networked to allow web access
- OSs run web servers and web clients
- Web based applications can be developed to run over web servers and clients.
  - Having a browser at the client is enough to run an application.
  - No special client software required



Mobile Computing

- Computing on smart phones and tablets.
- Potable and lightweight devices: mobile devices
- Many sensors: GPS, accelerometers, gyroscope, etc.
- Small screen, touch screen, no keyboard/mouse
- Wireless interfaces (port): 3G/4G, WiFi, Bluetooth.
- Mobile OS: iOS or Android

Applications:

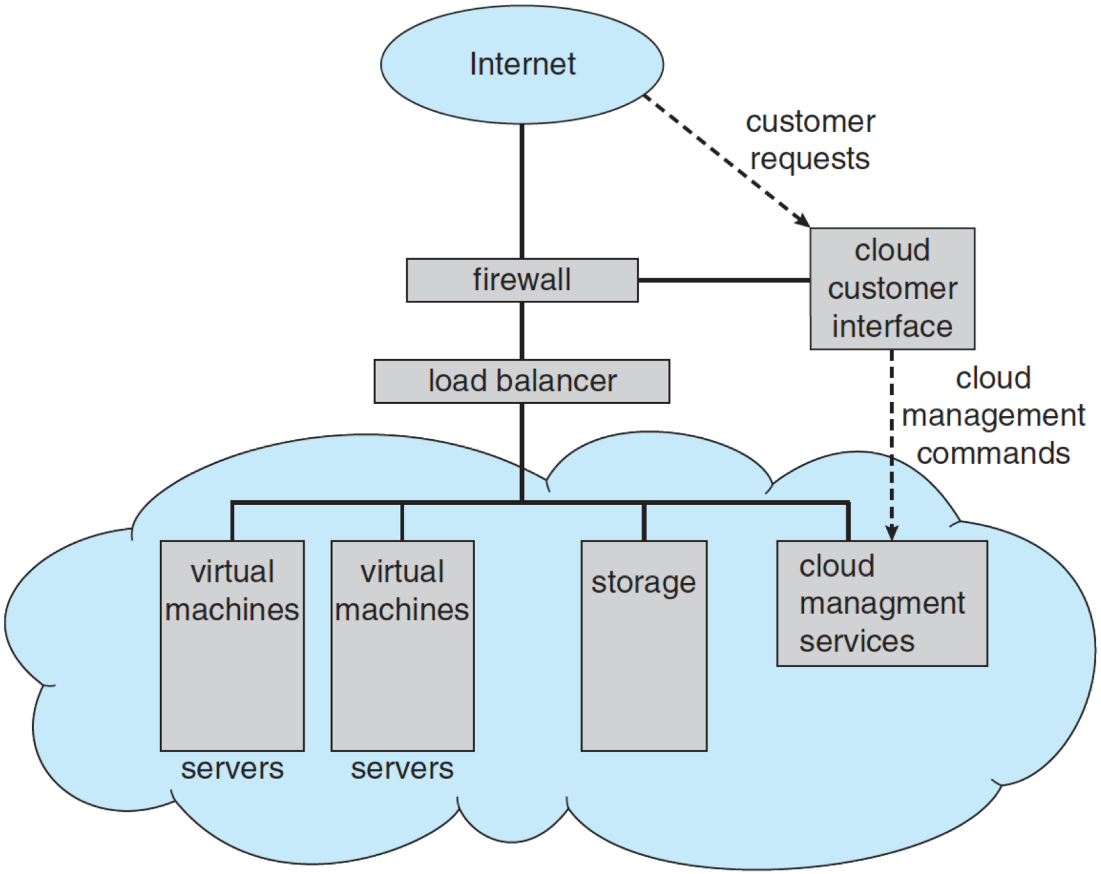
- New types:
  - Applications that use sensors
  - Location based applications
- How they are developed and run:
  - Web based applications or
  - Native applications

Cloud Computing

- Type of computing that delivers computing, storage, applications as a service across a network.
- Different types of services:
  - *Computing as a service:* remote virtual machine instances or platforms-APIs (software stacks) - IaaS or PaaS
  - *Storage as a service*: block storage (remote virtual disks), object storage (blob storage) - IaaS
  - *Software as a service:* Internet services, email services, web based services, … - SaaS

- Public Cloud: Can use anyone
- Private Cloud: internal to a company

Cloud Computing



A public cloud providing IaaS

Real Time Embedded Systems

- Embedded Computing
- Embedded computers in car engines, robots, microware ovens, …
- They do specific tasks.
- Little or no interface (no monitor)
- Some use general purpose processors (CPUs) and OSs (Linux)
- Some use ASICs – No OS
- OS is real time OS
  - Rigid timing requirements for tasks to be performed

Open-Source Operating Systems

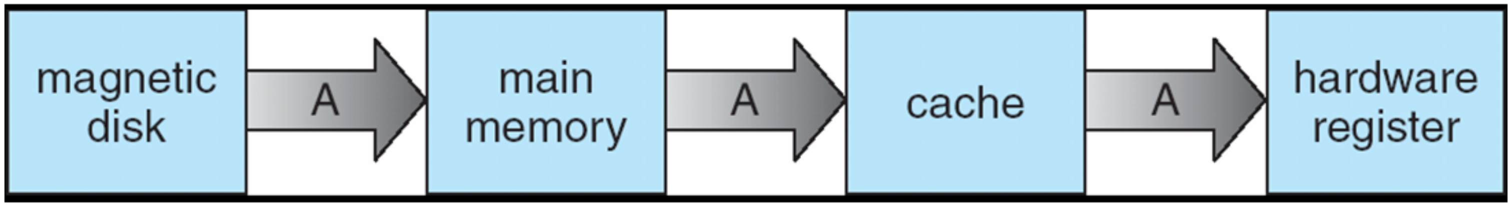
- Some operating systems made available in source-code format rather than just binary closed-source
- Examples include
  - GNU/Linux,
  - BSD Unix (FreeBSD, etc.)
  - Sun Solaris
- Closed source: Windows
- Hybrid: Mac OS X, iOS

References

- Operating System Concepts, Silberschatz et al.
- Modern Operating Systems, Andrew S. Tanenbaum et al.
- OSTEP, Remzi Arpaci-Dusseau et al.

Migration of Integer 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