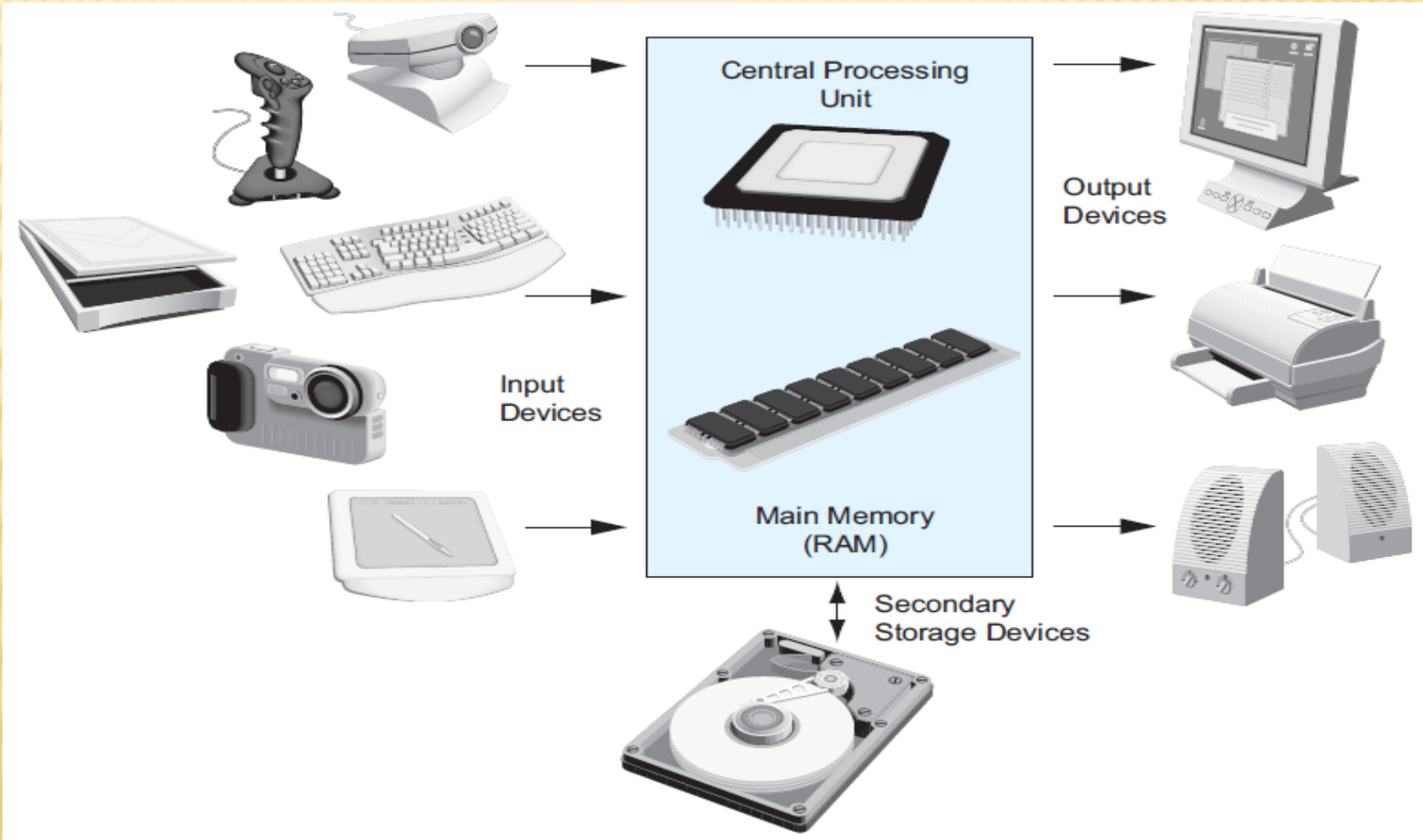
Advanced Operating Systems

Mahmoud Abou El-Mag Soliman
Department of Computer Science
Faculty of Computers and Artificial Intelligence
Sohag University

Introducing Operating Systems

A modern computer consists of one or more processors, some main memory, disks, printers, a keyboard, a mouse, a display, network interfaces, and various other input/output devices. If every application programmer had to understand how all these things work in detail, no code would ever get written. For this reason, computers are equipped with a layer of software called the operating system, whose job is to provide user programs with a better, simpler, cleaner, model of the computer and to handle managing all resources.

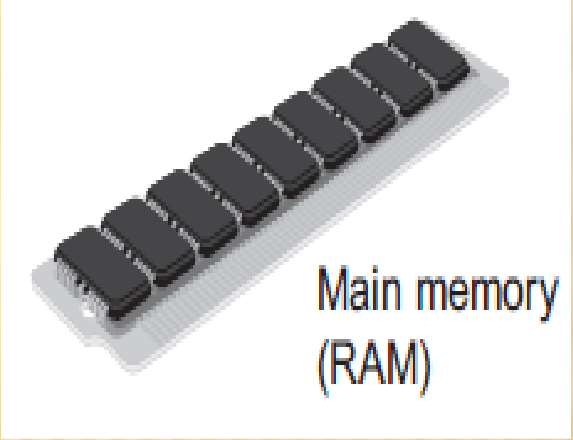
Typical Components of a Computer System (Computer Hardware)



Processing Devices

Main Memory:

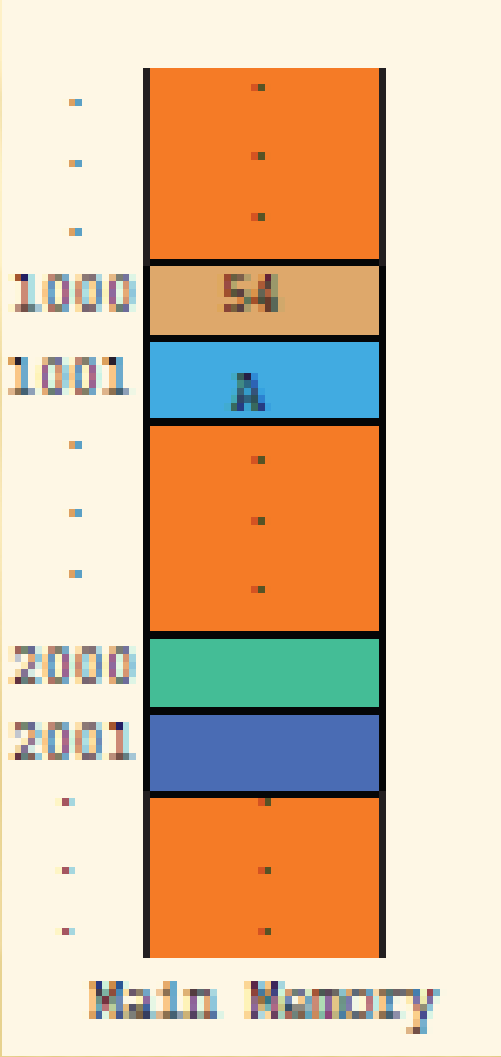
You can think of main memory as the computer's work area. This is where the computer stores a program while the program is running, as well as the data that the program is working with. Main memory is commonly known as random-access memory, or RAM. It is called this because the CPU is able to quickly access data stored at any random location in RAM. RAM is usually a volatile type of memory that is used only for temporary storage while a program is running. When the computer is turned off, the contents of RAM are erased.



Maim Memory (RAM)

RAM is similar in concept to a set of boxes in which each box can hold a 0 or a 1. Each box has a unique address that is found by counting across the columns and down the rows. A set of RAM boxes is called an array, and each box is known as a cell.

To find a specific cell, the RAM controller sends the column and row address down a thin electrical line etched into the chip. Each row and column in a RAM array has its own address line.

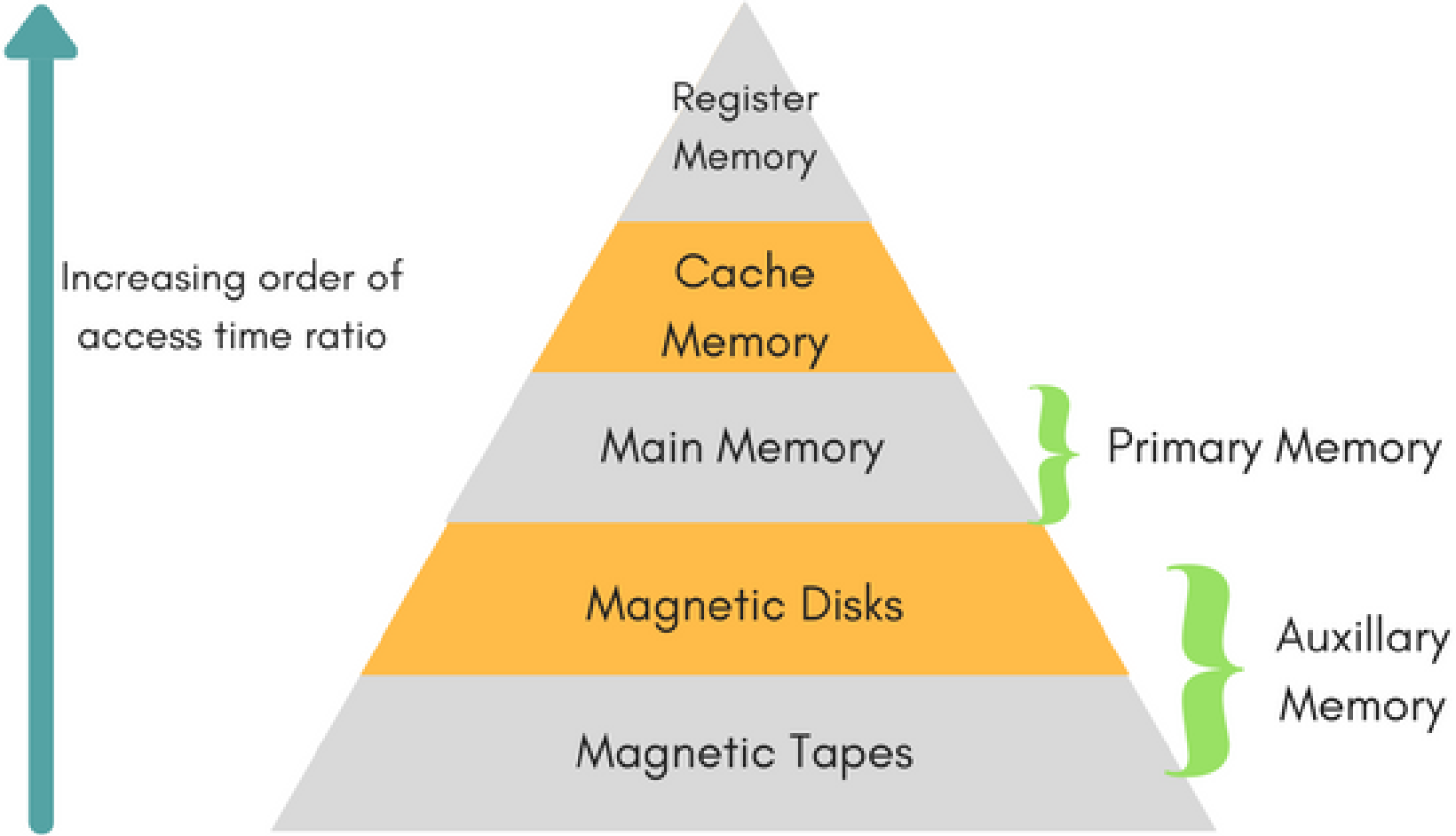


RAM vs. Virtual Memory

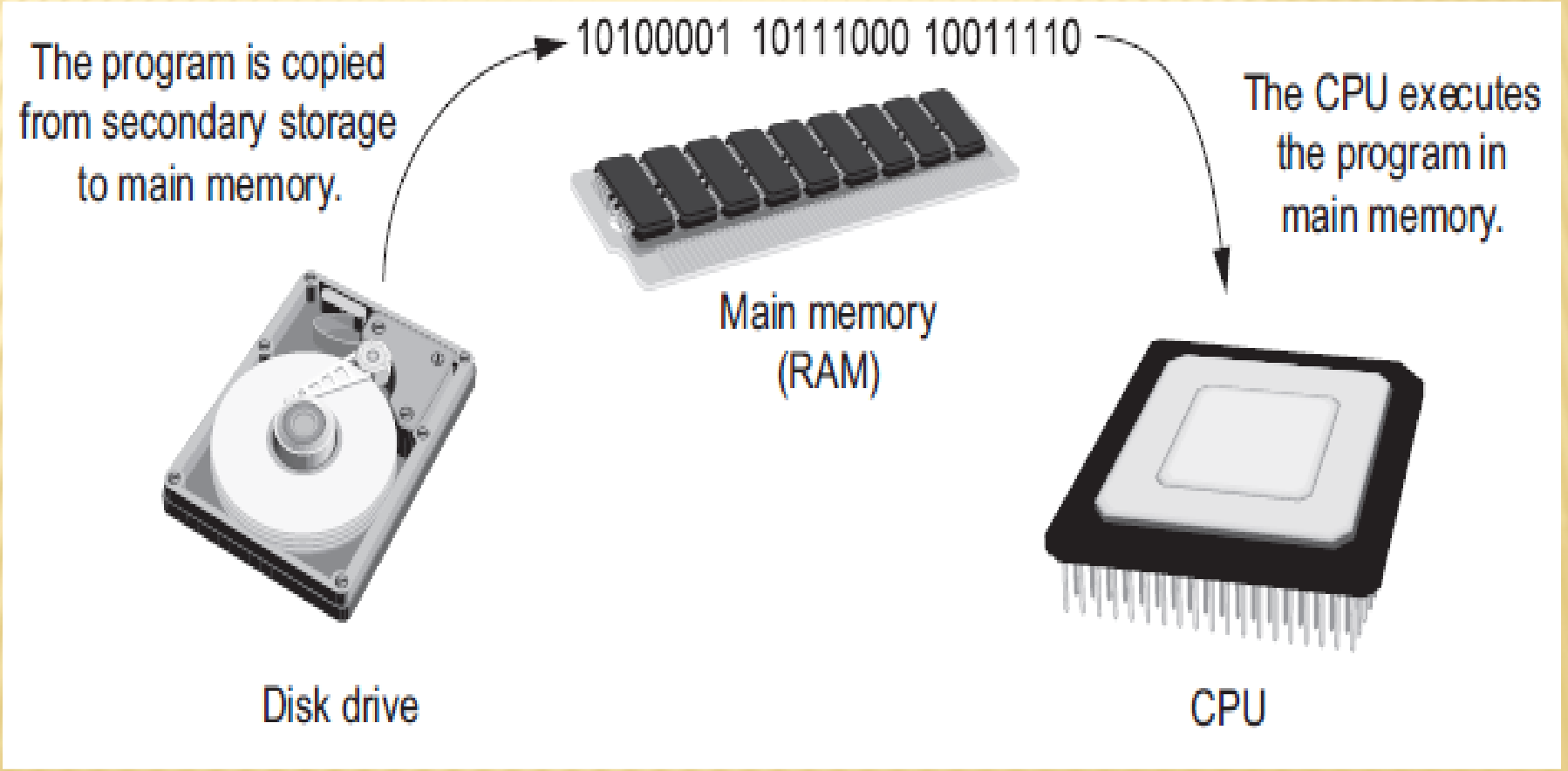
With virtual memory, data is temporarily transferred from RAM to disk storage, and virtual address space is increased using active memory in RAM and inactive memory in an HDD to form contiguous addresses that hold an application and its data. Using virtual memory, a system can load larger programs or multiple programs running at the same time, letting each operate as if it has infinite memory without having to add more RAM.

Virtual Memory = Main memory + Extended Memory

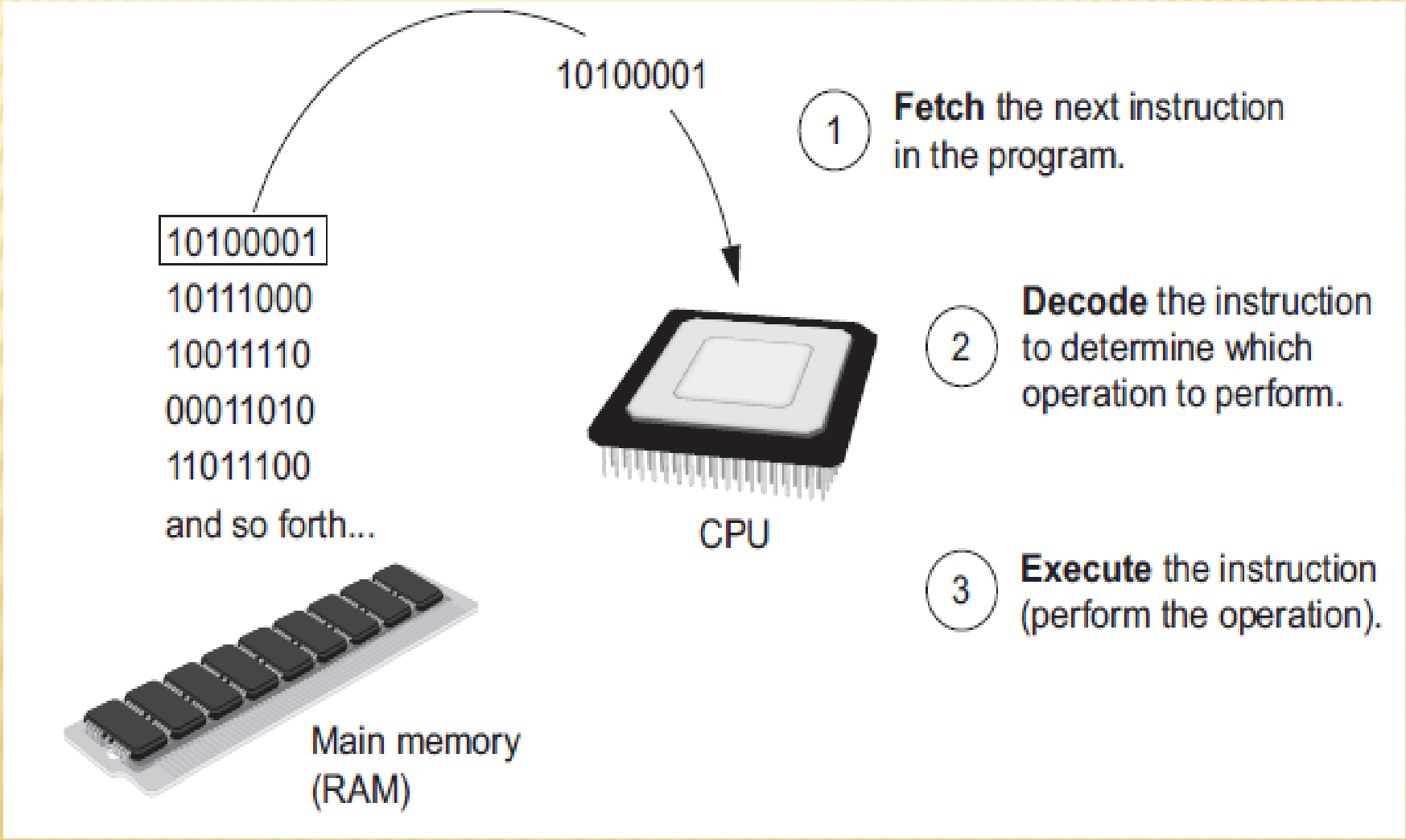
Memory Organization



A program is copied in main memory and then executed



The fetch-decode-execute cycle



RISC and CISC Processors

RISC Processor:

It is known as Reduced Instruction Set Computer. It is a type of microprocessor that has a limited number of instructions. They can execute their instructions very fast because instructions are very small and simple. RISC chips require fewer transistors which make them cheaper to design and produce.

CISC Processor:

It is known as Complex Instruction Set Computer. It contains large number of complex instructions. It was first developed by Intel.

Programs

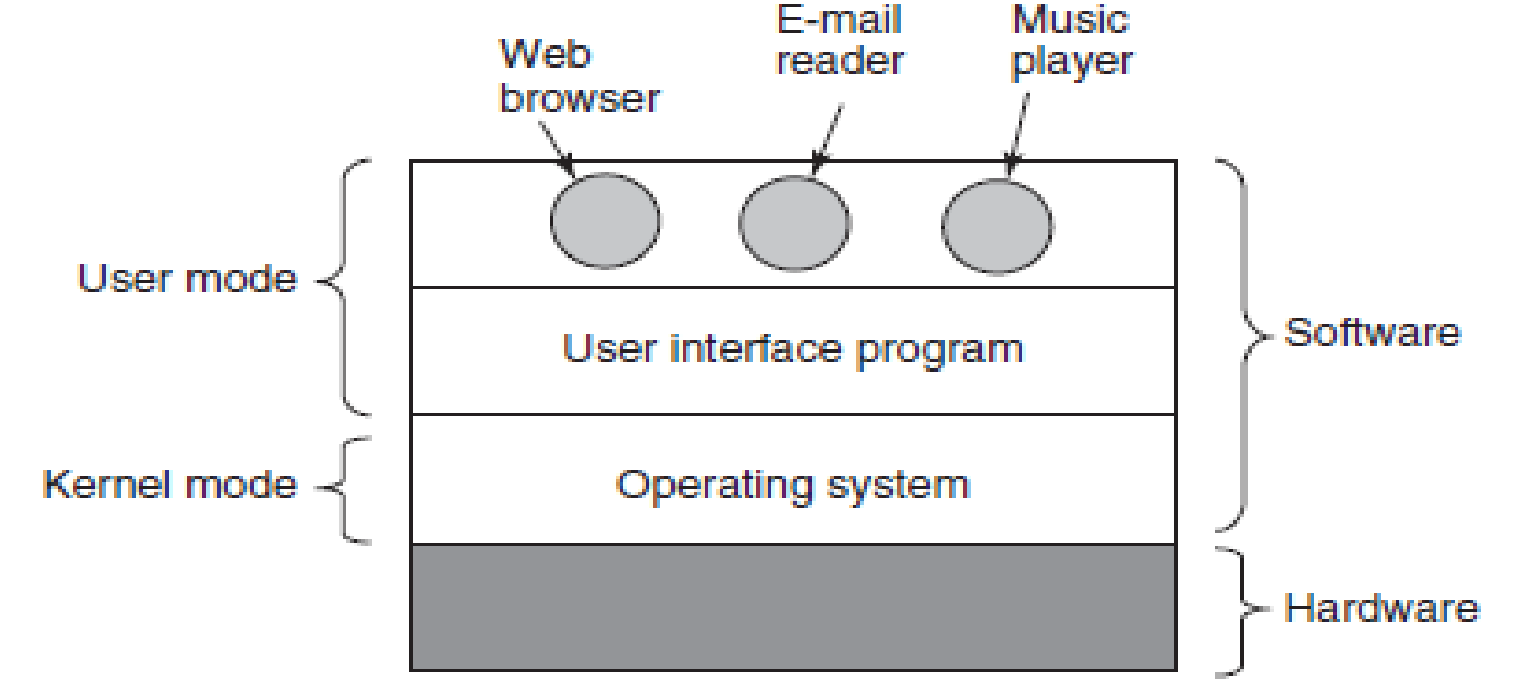
System Programs: System program is the type of software which is the interface between application software and system. System Software maintain the system resources and give the path for application software to run. An important thing is that without system software, system can not run. It is a general purpose software.

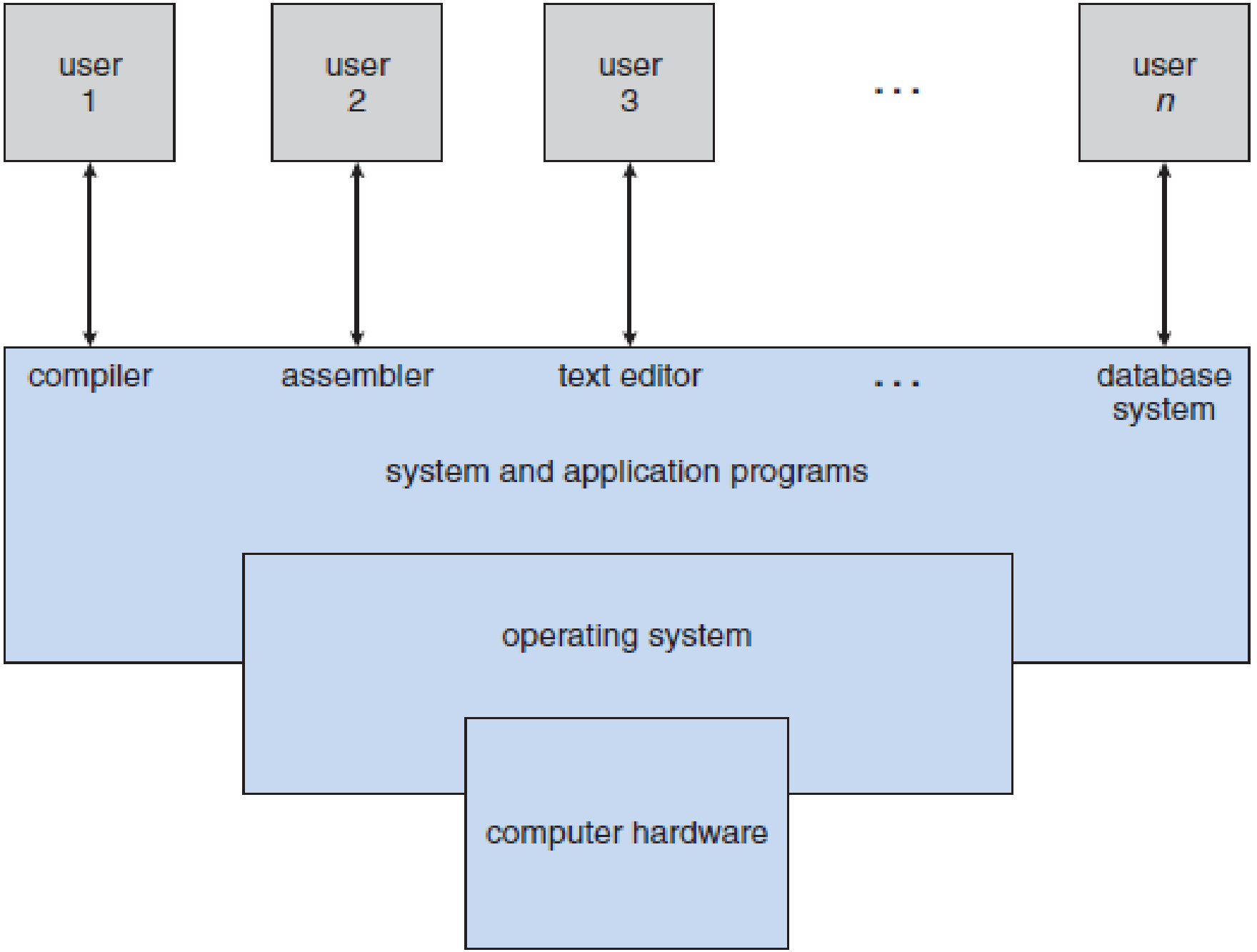
Application Programs: Application Program is the type of software which runs as per user request. It runs on the platform which is provide by system software. High level languages are used to write the application programs.

The main difference between System Program and Application Program is that without system program, system can not run on the other hand without application Program, system always runs.

Introducing Operating Systems

A simple overview of the main components of computer system is given in Figure. Here we see the hardware at the bottom. The hardware consists of chips, boards, disks, a keyboard, a monitor, and similar physical objects. On top of the hardware is the software.




user 1
user 2
user 3
...
user n
compiler
assembler
text editor
...
database system
system and application programs
operating system
computer hardware

User View

Such a system is designed for one user to monopolize its resources. The goal is to maximize the work (or play) that the user is performing. In this case, the operating system is designed mostly for ease of use.

In other cases, a user sits at a terminal connected to a mainframe or a minicomputer. Other users are accessing the same computer through other terminals. These users share resources and may exchange information. The operating system in such cases is designed to maximize resource utilization.

User View

In still other cases, users sit at workstations connected to networks of other workstations and servers. These users share resources such as networking and servers, including file, compute, and print servers. Therefore, their operating system is designed to compromise between individual usability and resource utilization.

User View

Some computers have little or no user view. For example, embedded computers in home devices and automobiles may have numeric keypads and may turn indicator lights on or off to show status, but they and their operating systems are designed primarily to run without user intervention.

System View

From the computer's point of view, the operating system is the program most intimately involved with the hardware. In this context, we can view an operating system as a resource allocator. A computer system has many resources that may be required to solve a problem: CPU time, memory space, file-storage space, I/O devices, and so on. The operating system acts as the manager of these resources.

A slightly different view of an operating system emphasizes the need to control the various I/O devices and user programs. An operating system is a control program.

Defining Operating Systems

A more common definition, and the one that we usually follow, is that the operating system is the one program running at all times on the computer—usually called the kernel. (Along with the kernel, there are two other types of programs: system programs, which are associated with the operating system but are not necessarily part of the kernel, and application programs, which include all programs not associated with the operation of the system.)

Computer-System Operation

A modern general-purpose computer system consists of one or more CPUs and a number of device controllers connected through a common bus that provides access to shared memory (Figure). Each device controller is in charge of a specific type of device (for example, disk drives, audio devices, or video displays). The CPU and the device controllers can execute in parallel, competing for memory cycles. To ensure orderly access to the shared memory, a memory controller synchronizes access to the memory.

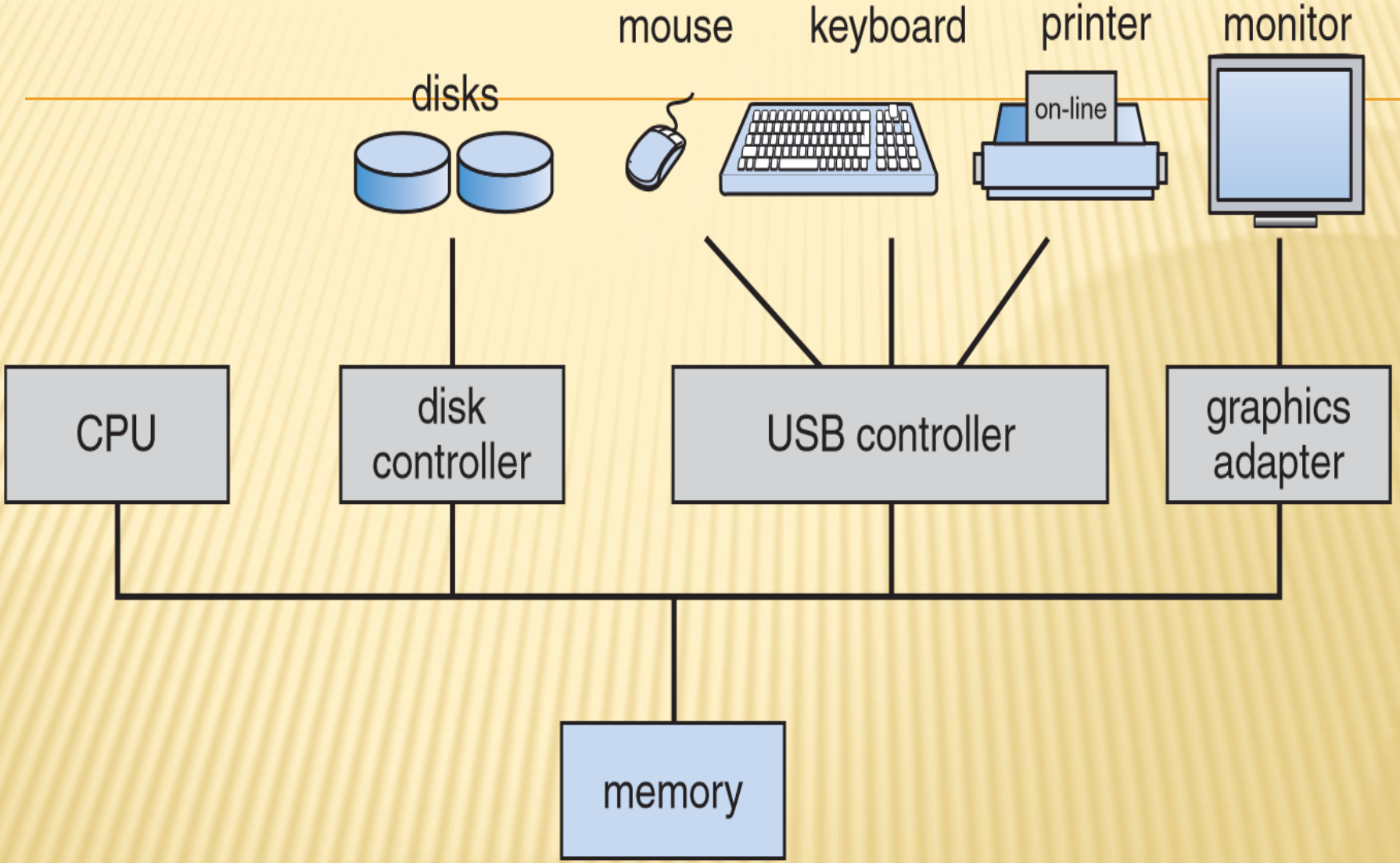


Figure 1.2 A modern computer system.

Computer-System Operation

For a computer to start running, when it is powered up or rebooted, it needs to have an initial program to run. This initial program, or bootstrap program. Typically, it is stored within the computer hardware in read-only memory (ROM). It initializes all aspects of the system, from CPU registers to device controllers to memory contents. The bootstrap program must know how to load the operating system and how to start executing that system. To accomplish this goal, the bootstrap program must locate the operating-system kernel and load it into memory.

Computer-System Operation

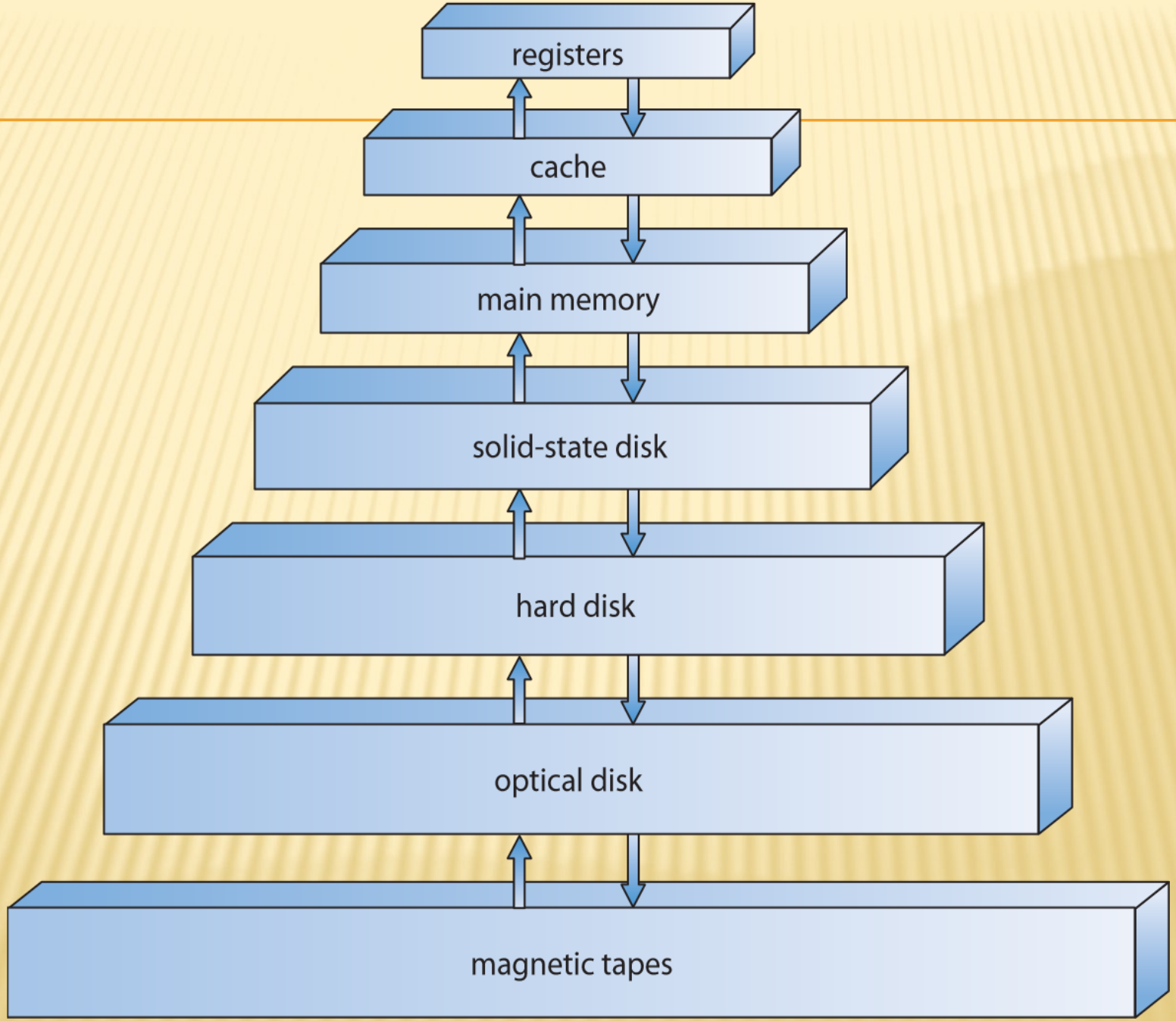
Once the kernel is loaded and executing, it can start providing services to the system and its users. Some services are provided outside of the kernel, by system programs that are loaded into memory at boot time to become system processes, or system daemons that run the entire time the kernel is running. On UNIX, the first system process is “init,” and it starts many other daemons. Once this phase is complete, the system is fully booted, and the system waits for some event to occur.

Computer-System Operation

The occurrence of an event is usually signaled by an interrupt from either the hardware or the software. Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by way of the system bus. Software may trigger an interrupt by executing a special operation called a system call (also called a monitor call).

Storage Structure

The CPU can load instructions only from memory, so any programs to run must be stored there. General-purpose computers run most of their programs from rewritable memory, called main memory (also called random-access memory, or RAM). Main memory commonly is implemented in a semiconductor technology called dynamic random-access memory (DRAM).

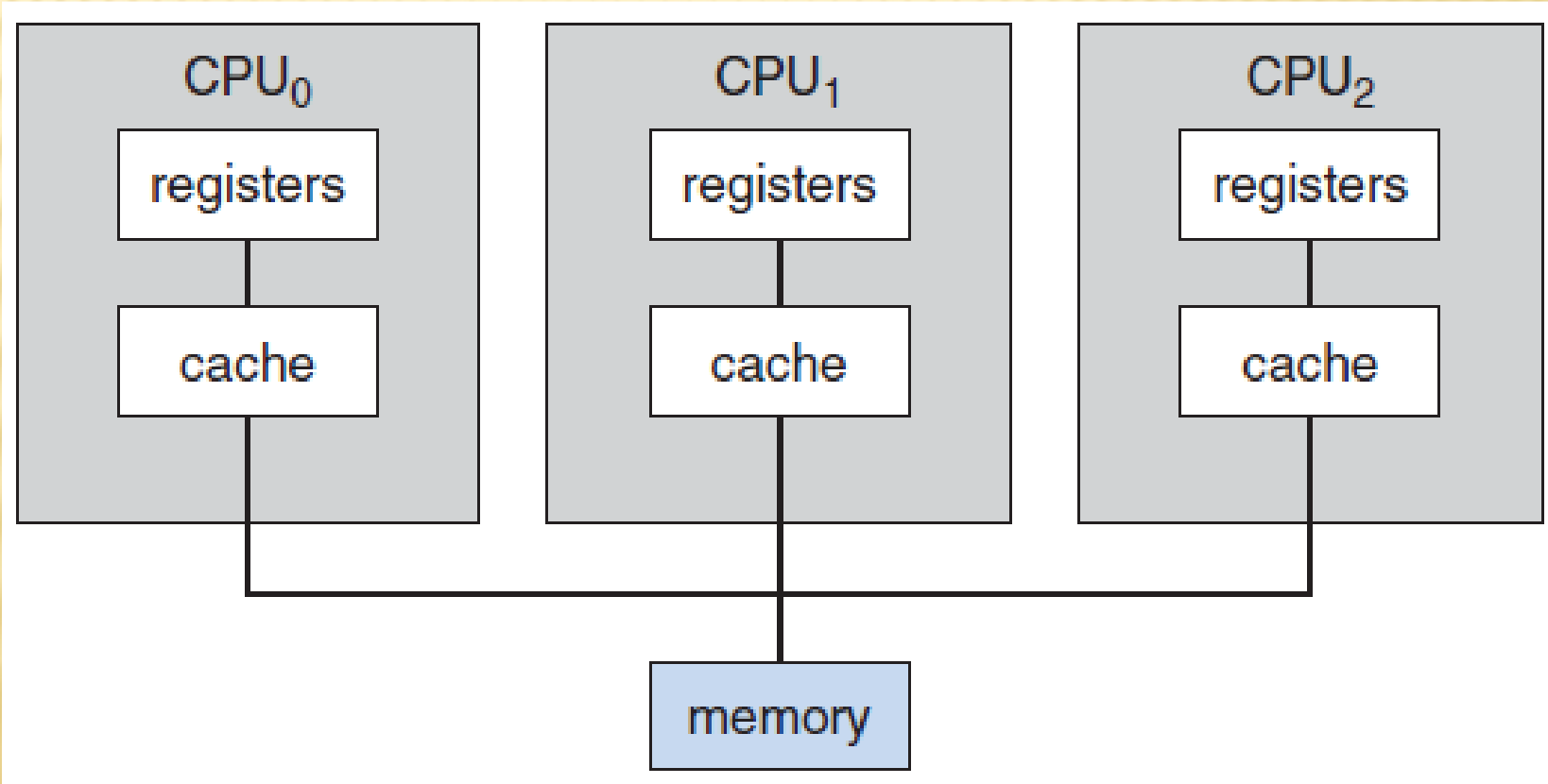

registers
cache
main memory
solid-state disk
hard disk
optical disk
magnetic tapes

I/O Structure

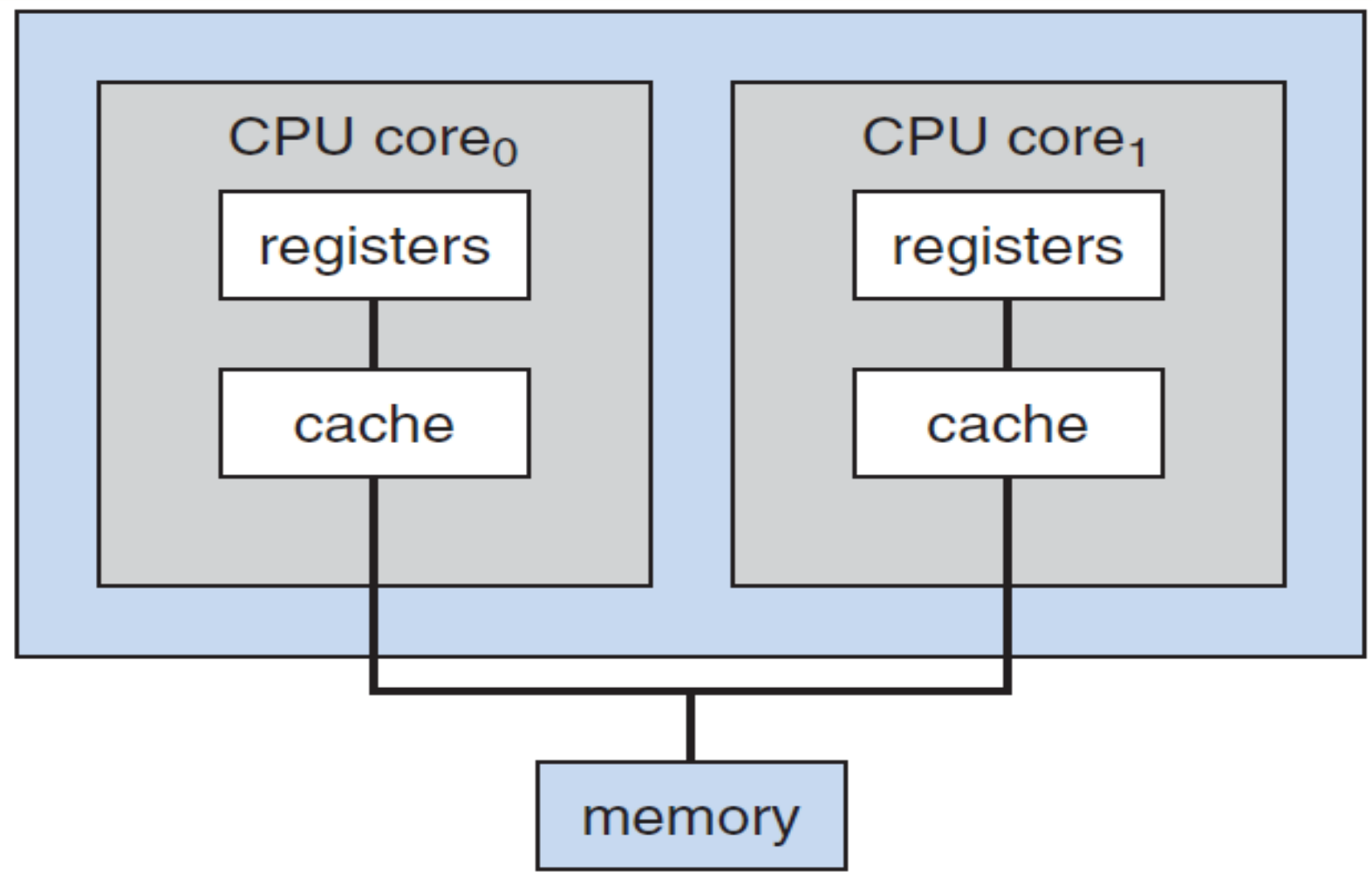
Storage is only one of many types of I/O devices within a computer. A large portion of operating system code is dedicated to managing I/O, both because of its importance to the reliability and performance of a system and because of the varying nature of the devices.

Computer-System Architecture

- Single-Processor Systems
- Multiprocessor Systems



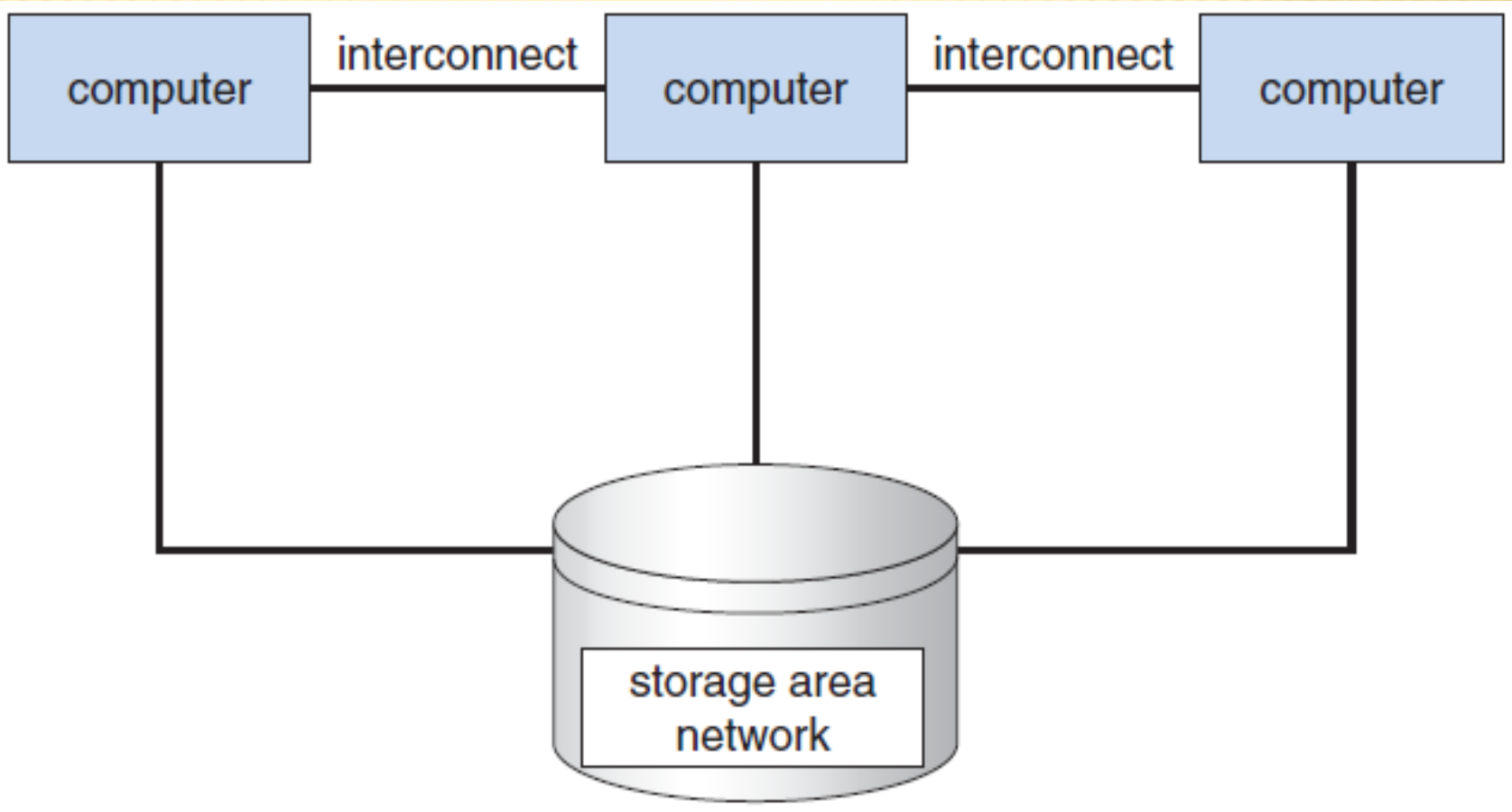
Computer-System Architecture



A dual-core design with two cores placed on the same chip.

Computer-System Architecture

- Clustered Systems



General structure of a clustered system.

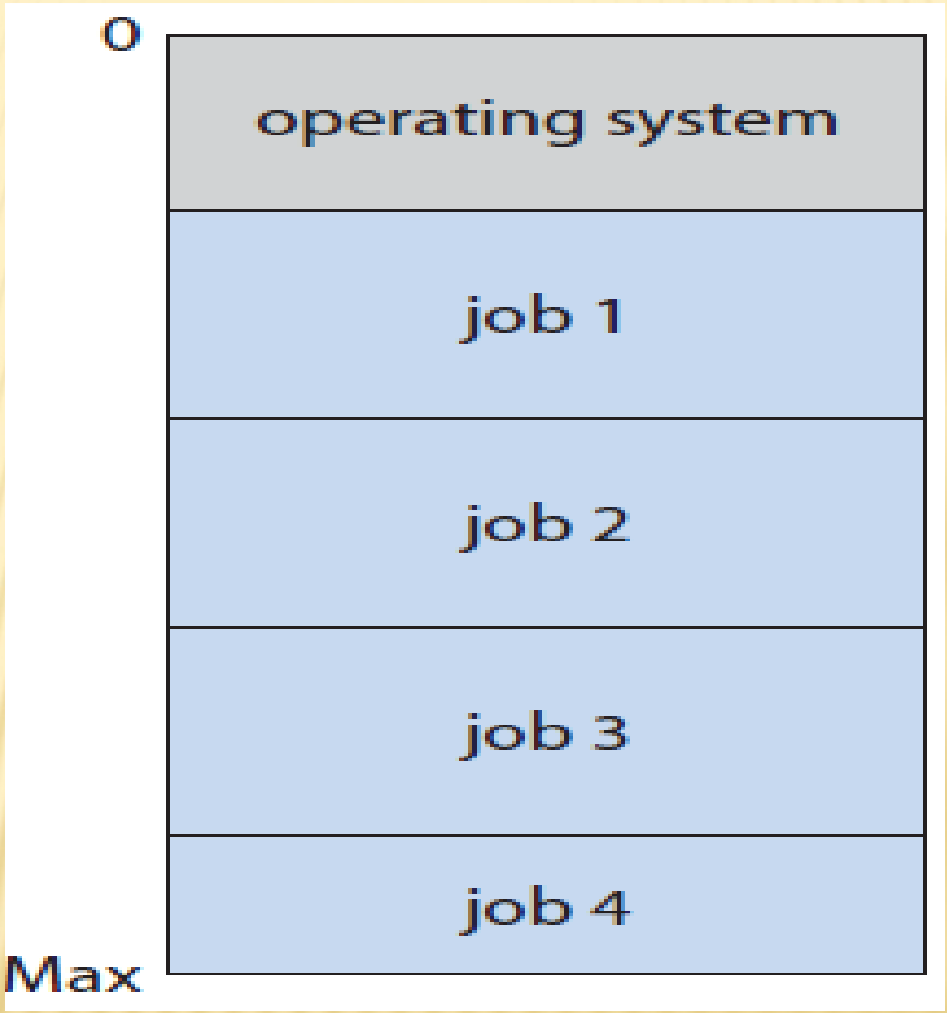
Operating-System Structure

One of the most important aspects of operating systems is the ability to multiprogram. A single program cannot, in general, keep either the CPU or the I/O devices busy at all times. Single users frequently have multiple programs running. Multiprogramming increases CPU utilization by organizing jobs (code and data) so that the CPU always has one to execute.

Operating-System Structure

The idea is as follows: The operating system keeps several jobs in memory simultaneously (Figure). Since, in general, main memory is too small to accommodate all jobs, the jobs are kept initially on the disk in the job pool. This pool consists of all processes residing on disk awaiting allocation of main memory.

Operating-System Structure



Memory layout for a multiprogramming system.

Operating-System Operations

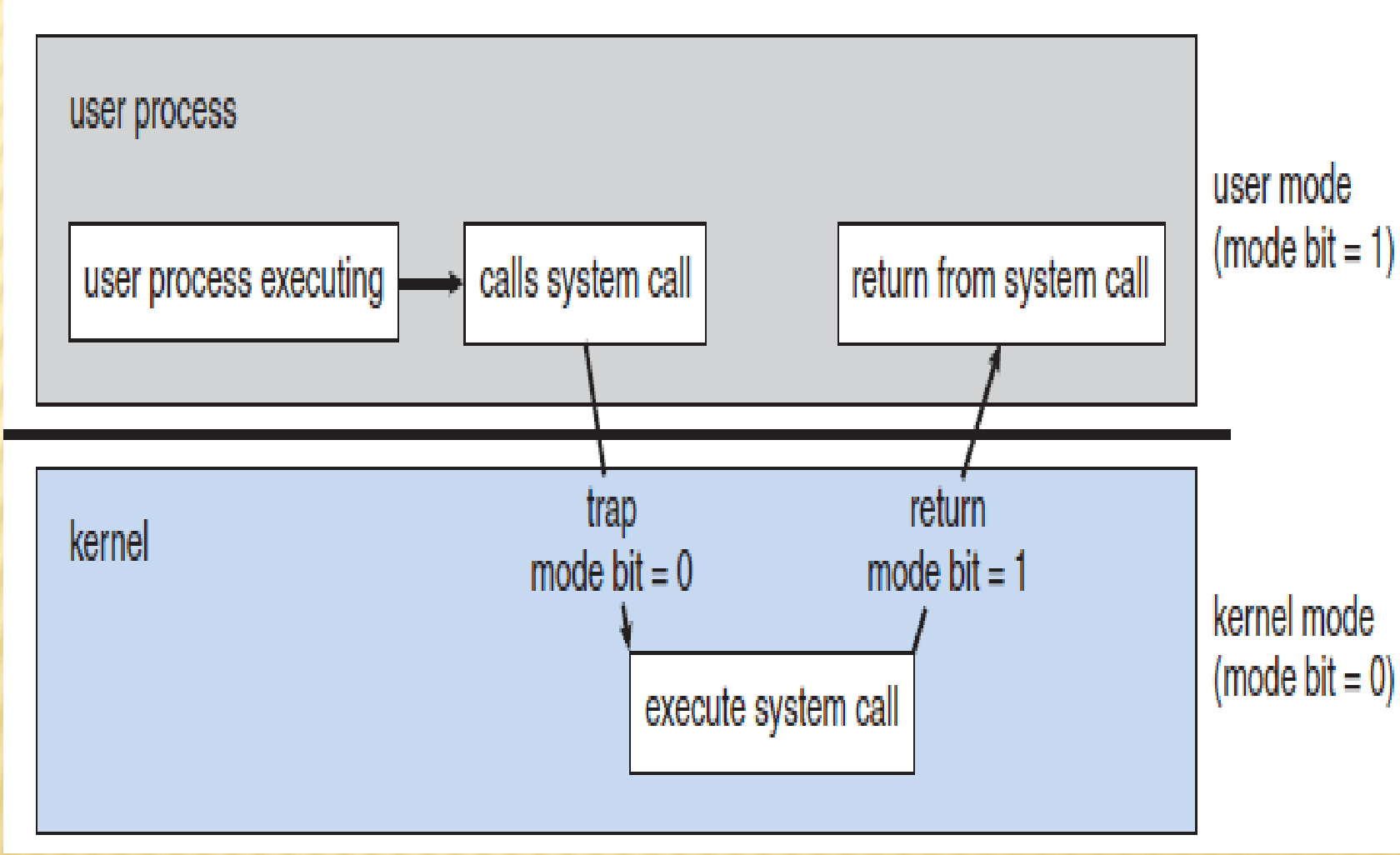
As mentioned earlier, modern operating systems are interrupt driven. Events are almost always signaled by the occurrence of an interrupt or a trap. A trap (or an exception) is a software-generated interrupt caused either by an error (for example, division by zero or invalid memory access) or by a specific request from a user program that an operating-system service be performed. For each type of interrupt, separate segments of code in the operating system determine what action should be taken.

Operating-System Operations

- Dual-Mode and Multimode Operation

In order to ensure the proper execution of the operating system, we must be able to distinguish between the execution of operating-system code and user defined code. The approach taken by most computer systems is to provide hardware support that allows us to differentiate among various modes of execution.

Operating-System Operations



Transition from user to kernel mode.

Operating-System Operations

- Timer

We must ensure that the operating system maintains control over the CPU. We cannot allow a user program to get stuck in an infinite loop or to fail to call system services and never return control to the operating system. To accomplish this goal, we can use a timer. A timer can be set to interrupt the computer after a specified period. The operating system sets the counter. Every time the clock ticks, the counter is decremented. When the counter reaches 0, an interrupt occurs.

Operating-System Operations

- Process Management

A program does nothing unless its instructions are executed by a CPU. A program in execution, as mentioned, is a process. A time-shared user program such as a compiler is a process. A word-processing program being run by an individual user on a PC is a process. A system task, such as sending output to a printer, can also be a process.

Operating-System Operations

The operating system is responsible for the following activities in connection with process management:

- Scheduling processes and threads on the CPUs
- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication

Operating-System Operations

- Memory Management

Main memory is a repository of quickly accessible data shared by the CPU and I/O devices. The central processor reads instructions from main memory during the instruction-fetch cycle and both reads and writes data from main memory during the data-fetch cycle (on a von Neumann architecture). As noted earlier, the main memory is generally the only large storage device that the CPU is able to address and access directly. For example, for the CPU to process data from disk, those data must first be transferred to main memory by CPU-generated I/O calls.

Operating-System Operations

The operating system is responsible for the following activities in connection with memory management:

- Keeping track of which parts of memory are currently being used and who is using them
- Deciding which processes (or parts of processes) and data to move into and out of memory
- Allocating and deallocating memory space as needed

Operating-System Operations

- Storage Management

To make the computer system convenient for users, the operating system provides a uniform, logical view of information storage. The operating system abstracts from the physical properties of its storage devices to define a logical storage unit, the file. The operating system maps files onto physical media and accesses these files via the storage devices.

Operating-System Operations

The operating system is responsible for the following activities in connection with file management:

- Creating and deleting files
- Creating and deleting directories to organize files
- Supporting primitives for manipulating files and directories
- Mapping files onto secondary storage
- Backing up files on stable (nonvolatile) storage media

Operating-System Operations

- Mass-Storage Management

As we have already seen, because main memory is too small to accommodate all data and programs, and because the data that it holds are lost when power is lost, the computer system must provide secondary storage to back up main memory. Most modern computer systems use disks as the principal on-line storage medium for both programs and data.

Operating-System Operations

The operating system is responsible for the following activities in connection with disk management:

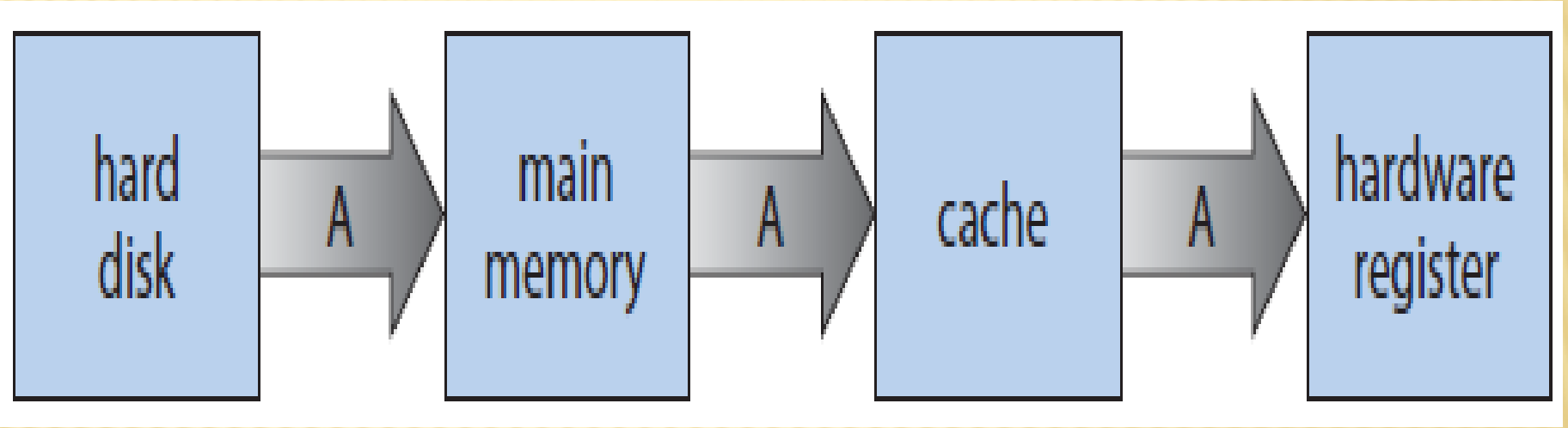
- Free-space management
- Storage allocation
- Disk scheduling

Operating-System Operations

- Caching

Caching is an important principle of computer systems. Here's how it works. Information is normally kept in some storage system (such as main memory). As it is used, it is copied into a faster storage system—the cache—on a temporary basis. When we need a particular piece of information, we first check whether it is in the cache. If it is, we use the information directly from the cache. If it is not, we use the information from the source, putting a copy in the cache under the assumption that we will need it again soon.

Operating-System Operations



Migration of integer A from disk to register.

Operating-System Operations

- I/O Systems

One of the purposes of an operating system is to hide the peculiarities of specific hardware devices from the user. The I/O subsystem consists of several components:

- A memory-management component that includes buffering, caching, and spooling
- A general device-driver interface
- Drivers for specific hardware devices

Operating-System Operations

- Protection and Security

If a computer system has multiple users and allows the concurrent execution of multiple processes, then access to data must be regulated. For that purpose, mechanisms ensure that files, memory segments, CPU, and other resources can be operated on by only those processes that have gained proper authorization from the operating system. The timer ensures that no process can gain control of the CPU without eventually relinquishing control.

Computing Environments

Traditional Computing

At home, most users once had a single computer with a slow modem connection to the office. Today, network-connection speeds once available only at great cost are relatively inexpensive in many places, giving home users more access to more data. These fast data connections are allowing home computers to serve up Web pages and to run networks that include printers, client PCs, and servers. Many homes use firewalls to protect their networks from security breaches.

Computing Environments

Mobile Computing

Mobile computing refers to computing on handheld smartphones and tablet computers. These devices share the distinguishing physical features of being portable and lightweight. Historically, compared with desktop and laptop computers, mobile systems gave up screen size, memory capacity, and overall functionality in return for handheld mobile access to services such as e-mail and web browsing.

Computing Environments

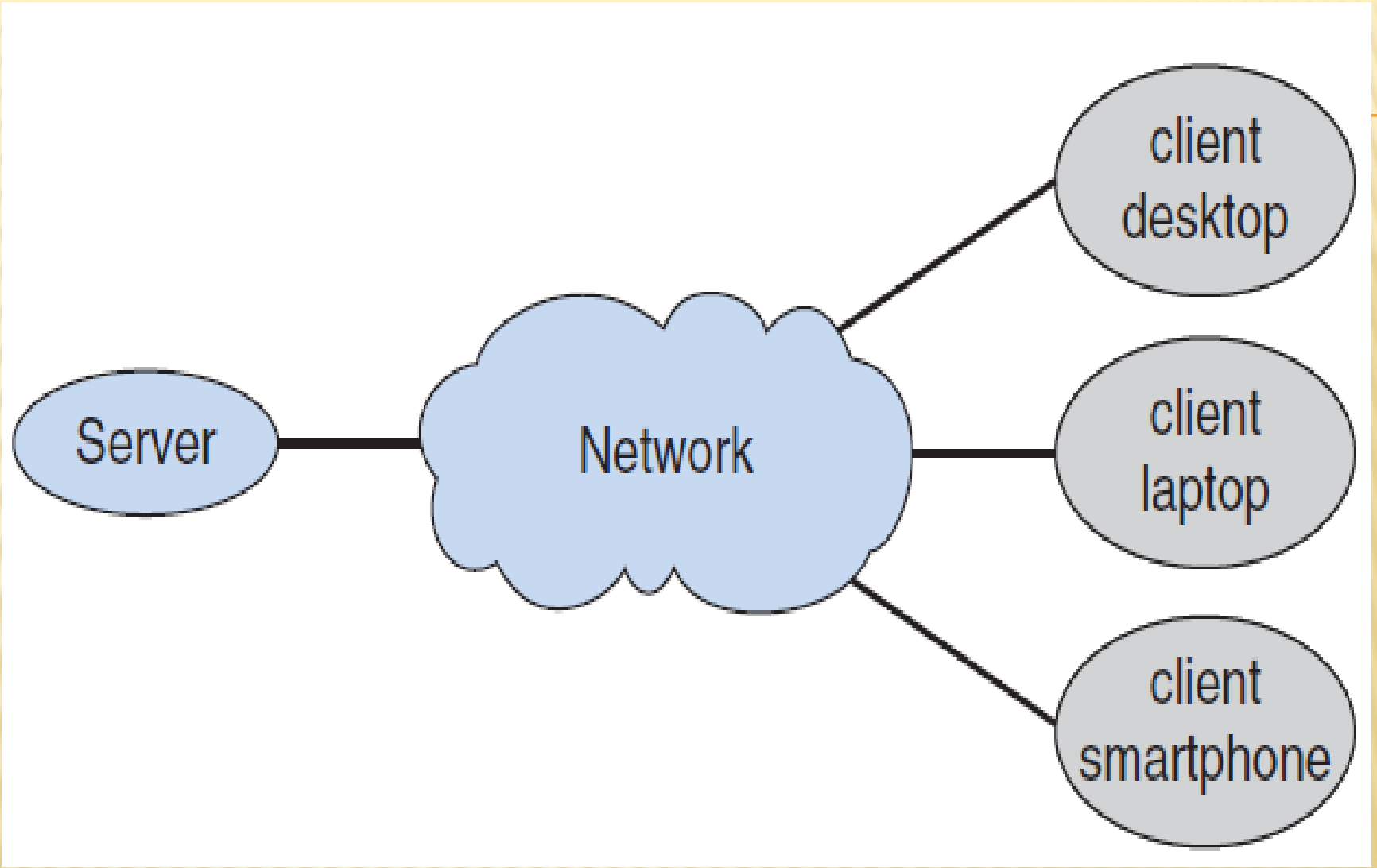
- Distributed Systems

A distributed system is a collection of physically separate, possibly heterogeneous, computer systems that are networked to provide users with access to the various resources that the system maintains. Access to a shared resource increases computation speed, functionality, data availability, and reliability. Some operating systems generalize network access as a form of file access, with the details of networking contained in the network interface's device driver.

Computing Environments

- Client–Server Computing

As PCs have become faster, more powerful, and cheaper, designers have shifted away from centralized system architecture. Terminals connected to centralized systems are now being supplanted by PCs and mobile devices. Correspondingly, user-interface functionality once handled directly by centralized systems is increasingly being handled by PCs, quite often through a web interface. As a result, many of today's systems act as server systems to satisfy requests generated by client systems.

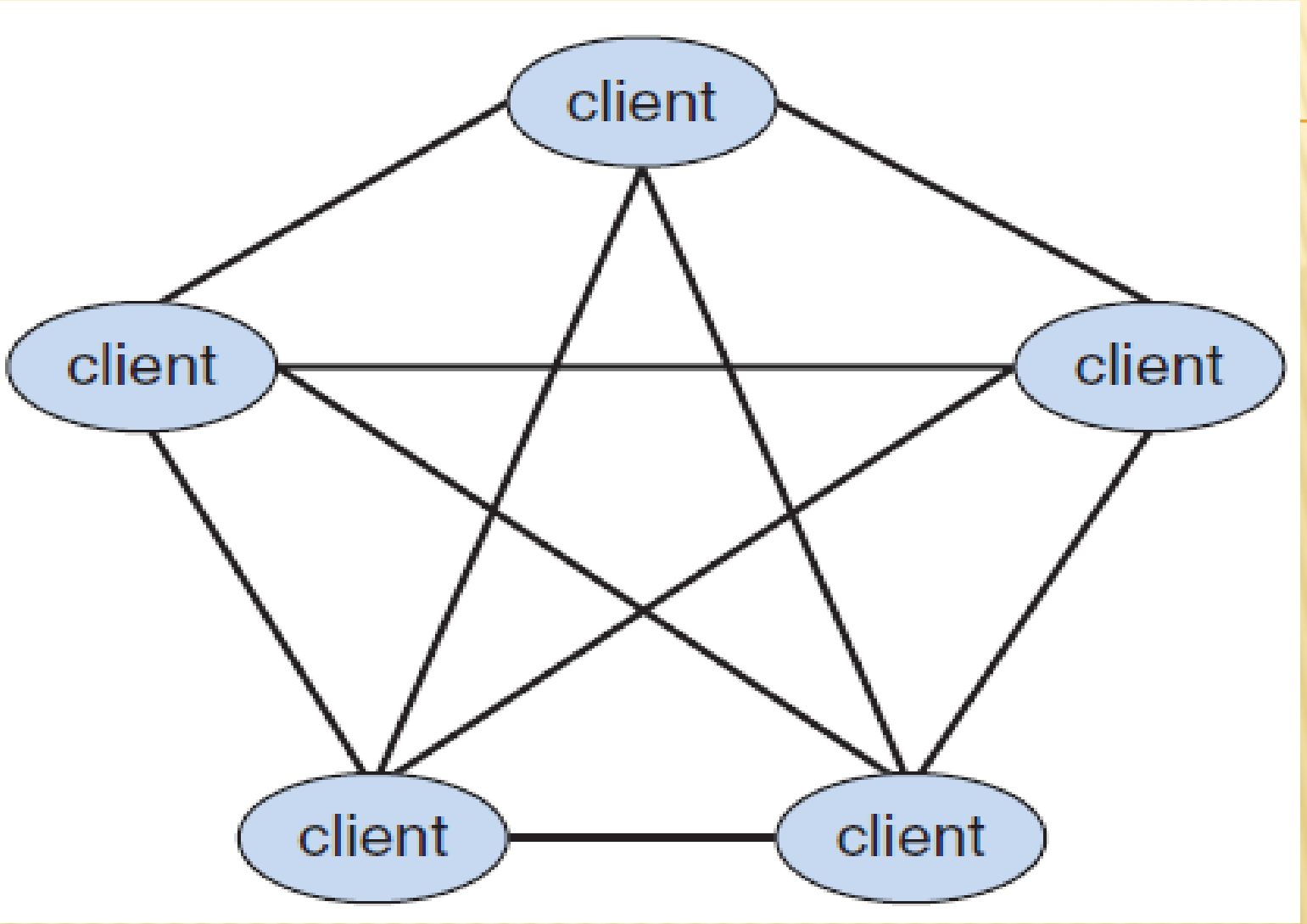


General structure of a client–server system

Computing Environments

Peer-to-Peer Computing

Another structure for a distributed system is the peer-to-peer (P2P) system model. In this model, clients and servers are not distinguished from one another. Instead, all nodes within the system are considered peers, and each may act as either a client or a server, depending on whether it is requesting or providing a service. Peer-to-peer systems offer an advantage over traditional client-server systems. In a client-server system, the server is a bottleneck.



Peer-to-peer system with no centralized service

Computing Environments

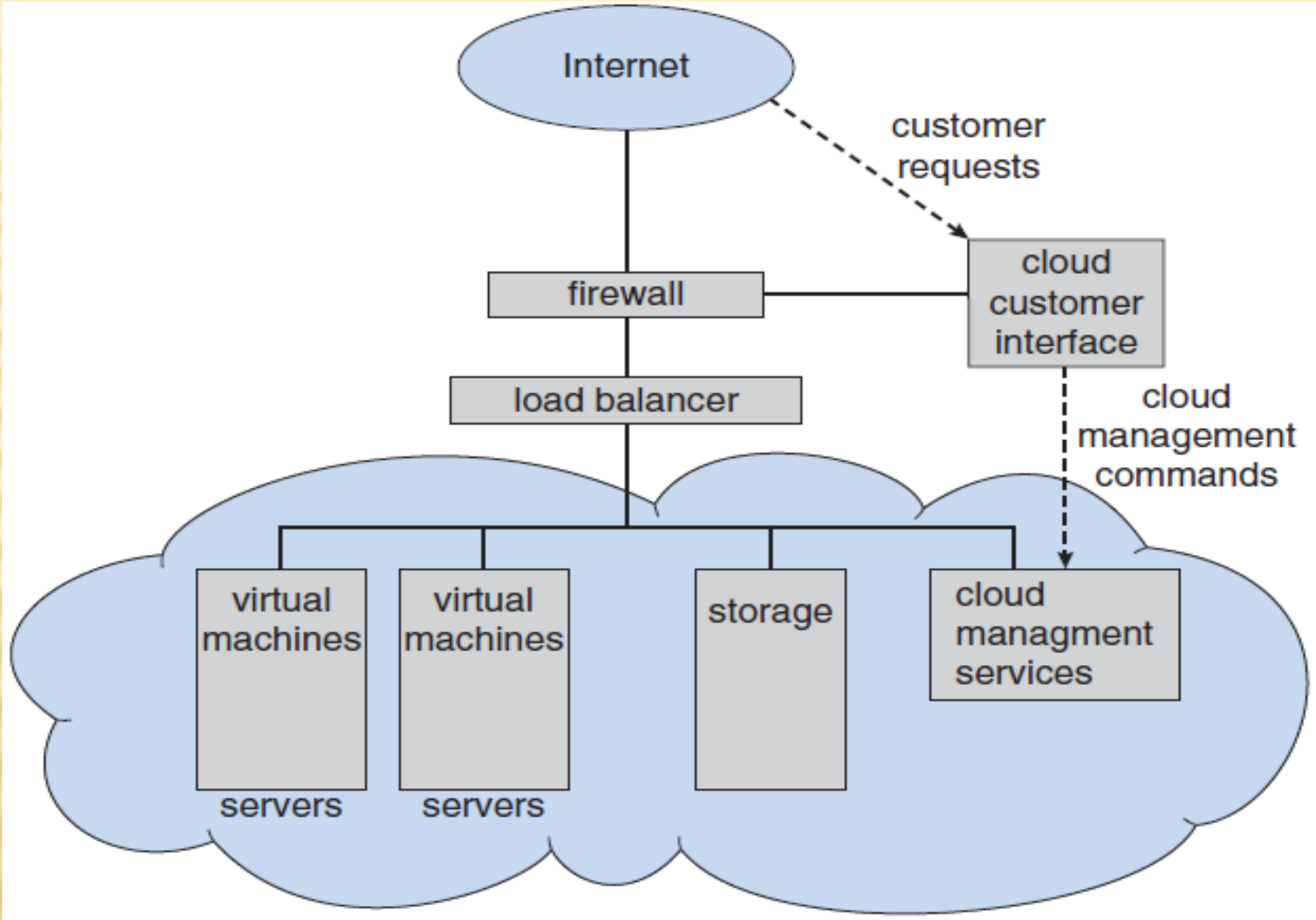
Virtualization

Virtualization is one member of a class of software that also includes emulation. Emulation is used when the source CPU type is different from the target CPU type. For example, when Apple switched from the IBM Power CPU to the Intel x86 CPU for its desktop and laptop computers, it included an emulation facility called "Rosetta," which allowed applications compiled for the IBM CPU to run on the Intel CPU. That same concept can be extended to allow an entire operating system written for one platform to run on another.

Computing Environments

- Cloud Computing

Cloud computing is a type of computing that delivers computing, storage, and even applications as a service across a network. In some ways, it's a logical extension of virtualization, because it uses virtualization as a base for its functionality. For example, the Amazon Elastic Compute Cloud (EC2) facility has thousands of servers, millions of virtual machines, and petabytes of storage available for use by anyone on the Internet.



Cloud computing

Computing Environments

- Real-Time Embedded Systems

Embedded computers are the most prevalent form of computers in existence. These devices are found everywhere, from car engines and manufacturing robots to DVDs and microwave ovens. They tend to have very specific tasks. The systems they run on are usually primitive, and so the operating systems provide limited features. Usually, they have little or no user interface, preferring to spend their time monitoring and managing hardware devices, such as automobile engines and robotic arms.

Open-Source Operating Systems

Open-source operating systems are those available in source-code format rather than as compiled binary code. Linux is the most famous opensource operating system, while Microsoft Windows is a well-known example of the opposite closed-source approach. Apple's Mac OS X and iOS operating systems comprise a hybrid approach. They contain an open-source kernel named Darwin yet include proprietary, closed-source components as well.