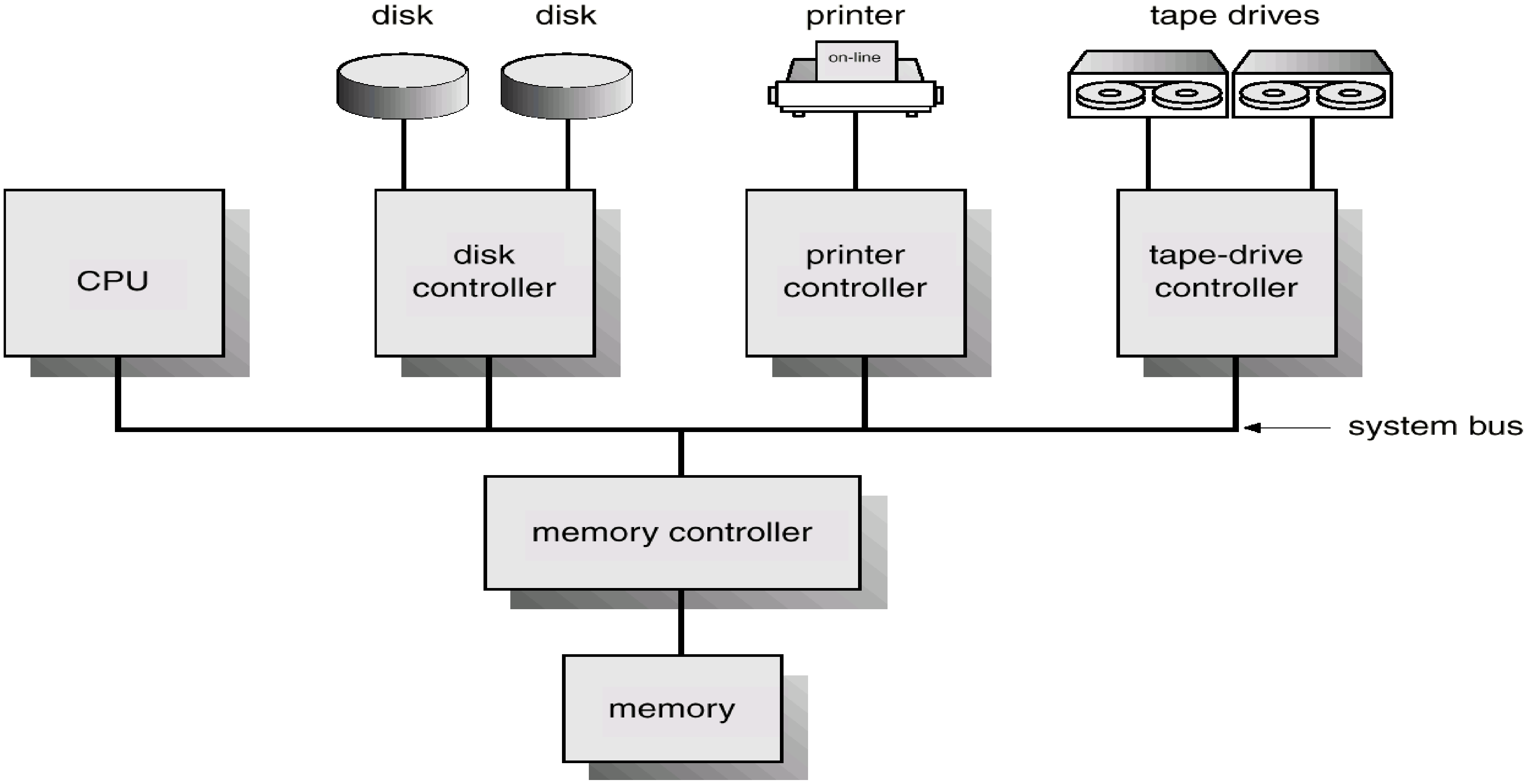
ch 2
Computer System Operation and Structure

By

Lecturer Ameen A.Noor

Computer System Operation

- A modern, general-purpose computer system consists of a CPU and a number of device controllers that are connected through a common bus that provides access to shared memory system.


disk
disk
printer
on-line
tape drives
CPU
disk controller
printer controller
tape-drive controller
system bus
memory controller
memory

Booting 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, tends to be simple. Typically, it is stored in read-only memory (ROM) such as firmware (coded instructions that are stored permanently in read-only memory) or EEPROM within the computer hardware. It initializes all aspects of the system, from CPU registers to device controllers to memory contents.

Booting Operation

- The bootstrap program must know how to load the operating system and to start executing that system. To accomplish this goal, the bootstrap program must locate and load into memory the operating-system kernel. The operating system then starts executing the first process, such as "init," and waits for some event to occur. 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.
- **Booting:** is the operation of bringing operating system kernel from secondary storage and put it in main storage to execute it in CPU. There is a program bootstrap which is performing this operation when computer is powered up or rebooted.
- **Bootstrap:** is a simple initial program, it is stored in ROM such as firmware or EEPROM(**Electrically Erasable Programmable Read-Only Memory)** within the computer hardware.

Functions of Bootstrap program

1. Initialize all the aspect of the system, from CPU registers to device controllers to memory contents.
2. Locate and load the operating system kernel into memory then the operating system starts executing the first process, such as “init” and waits for some event to occur.

- Types of events are either software events (system call) or hardware events (signals from the hardware devices to the CPU through the system bus and known as an interrupt).

I/O Structure

- Each I/O device connected to the computer system through its controller. A device controller maintains some local buffer storage and a set of special-purpose registers. The device controller is responsible for moving the data between the peripheral devices that it controls and its local buffer storage.

I/O Interrupts

- To start an I/O operation, the CPU loads the appropriate registers within the device controller. The device controller, in turn, examines the contents of these registers to determine what action to take. For example, if it finds a read request, the controller will start the transfer of data from the device to its local buffer. Once the transfer of data is complete, the device controller informs the CPU that it has finished its operation.

Interrupt: Most devices send a signal (interrupt) to the processor when an event occurs. The operating system can respond a change in device status by notifying processes that are waiting on such events.

Trap or Exceptions: its interrupts generated in response to errors for example (division by zero or invalid memory access). The processor invokes the operating system to determine how to respond (terminate or restart).

Interrupt Vector (IV): it is a fixed locations (an array) in the low memory area (first 100 location of RAM) of operating system when interrupt occur the CPU stops what it is doing and transfer execution to a fixed location (IV) contain starting address of the interrupt of the interrupt service routine (ISR), on completion the CPU resume the interrupted computation.

Interrupt Service Routine (ISR): is it a routine provided to be responsible for dealing with the interrupt.

DMA Structure

- A high-speed device, however such as a tape, disk, or communications network-may be able to transmit information at close to memory speeds; the CPU needs two microseconds to respond to each interrupt and interrupts arrive every four microseconds, for example, that does not leave much time for process execution. To solve this problem, direct memory access (DMA) is used for high-speed I/O devices. After setting up buffers, pointers, and counters for the I/O device, the device controller transfers an entire block of data directly to or from its own buffer storage to memory, with no intervention by the CPU. Only one interrupt is generated per block, rather than the one interrupt per byte (or word) generated for low-speed devices. The DMA controller interrupts the CPU when the transfer has been completed.

Storage Structure

- Computer programs must be in main memory (RAM) to be executed. Main memory is the only large storage area that the processor can access directly. Each word has its own address.
- Interaction is achieved through a sequence of load or store instructions to specific memory addresses. The load instruction moves a word from main memory to an internal register within the CPU, whereas the store instruction moves the content of a register to main memory.
- We want the programs and data to reside in main memory permanently.

This arrangement is not possible for the following two reasons:

1. Main memory is usually too small to store all needed programs and data permanently.
2. Main memory is a volatile storage device that loses its contents when power is turned off or otherwise lost.

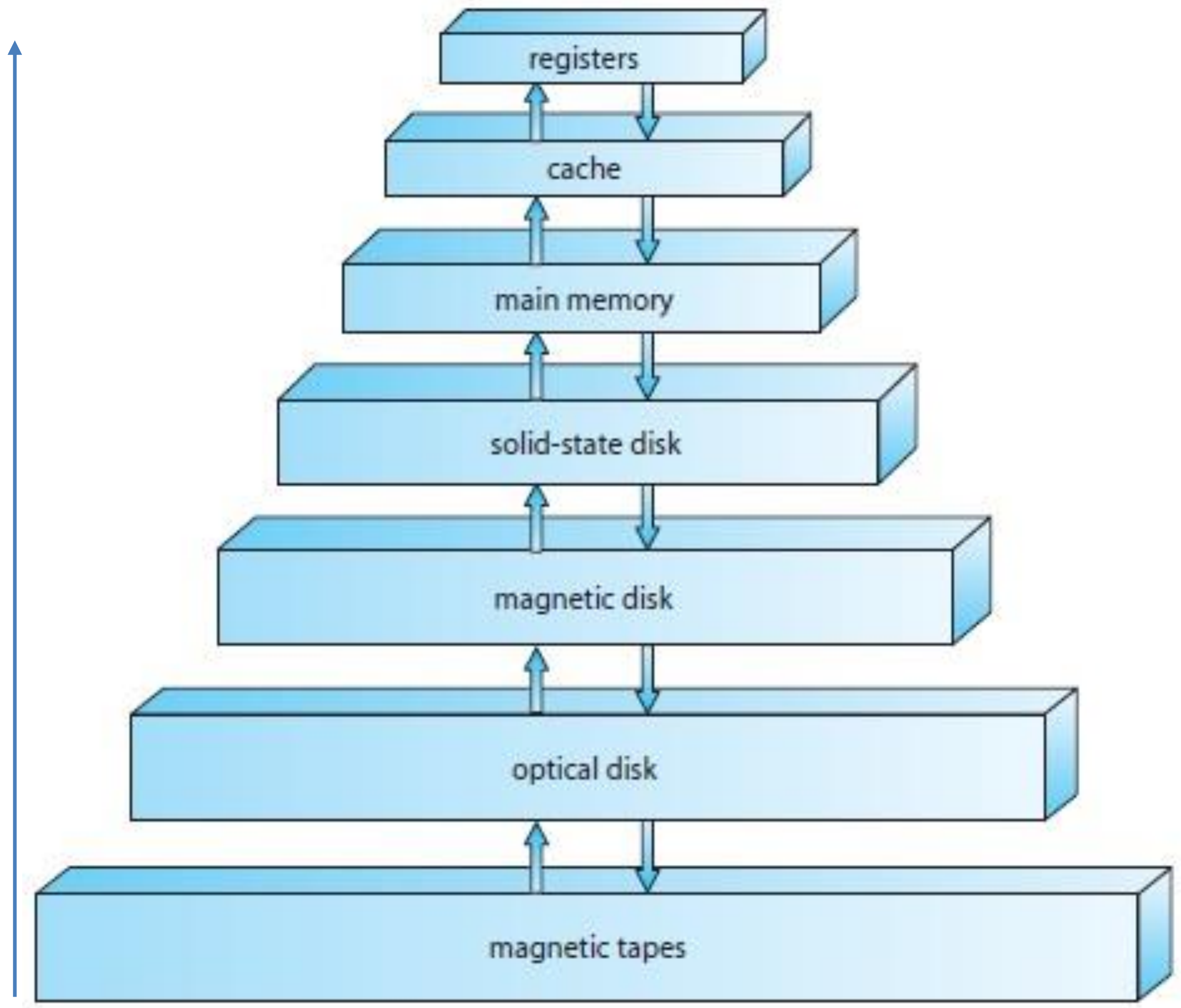
Most computer systems provide secondary storage as an extension of main memory. The main requirement for secondary storage is that it be able to hold large quantities of data permanently. The most common secondary-storage device is a magnetic disk, which provides storage of both programs and data. They are other many media such as floppy disks, hard disks, CD-ROMs, and DVDs.

STORAGE HIERARCHY

Low Capacity

High speed

High cost



Hardware Protection

- When we have single user any error occur to the system then we could determine that this error must be caused by the user program, but when we begin to dealing with spooling, multiprogramming, and sharing disk to hold many users data this sharing both improved utilization and increase problems. In multiprogramming system, where one error program might modify the program or data of another program, or even the resident monitor itself. MS-DOS and the Macintosh OS both allow this kind of error.
- A properly designed operating system must ensure that an incorrect (or malicious) program cannot cause other programs to execute incorrectly.
- Many programming errors are detected by the hardware. These errors are normally handled by the operating system.

Dual-Mode Operation

- To ensure proper operation, we must protect the operating system and all other programs and their data from any malfunctioning program. Protection is needed for any shared resource. The approach taken by many operating systems provides hardware support that allows us to differentiate among various modes of execution.
- A bit, called the mode bit, is added to the hardware of the computer to indicate the current mode: monitor (0) or user (1). With the mode bit, we are able to distinguish between a task that is executed on behalf of the operating system, and one that is executed on behalf of the user.

I/O Protection

- A user program may disrupt the normal operation of the system by issuing illegal I/O instructions, we can use various mechanisms to ensure that such disruptions cannot take place in the system.
- One of them is by defining all I/O instructions to be privileged instructions. Thus, users cannot issue I/O instructions directly; they must do it through the operating system, by execute a system call to request that the operating system performing I/O on its behalf. The operating system, executing in monitor mode, checks that the request is valid, and (if the request is valid) does the I/O requested. The operating system then returns to the user.

Memory Protection

- To ensure correct operation, we must protect the interrupt vector from modification by a user program. This protection must be provided by the hardware. We need the ability to determine the range of legal addresses that the program may access, and to protect the memory outside that space. We could provide this protection by using two registers, usually a base and a limit register. The base register holds the smallest legal physical memory address; the limit register contains the size of the range. This protection is accomplished by the CPU hardware comparing every address generated in user mode with the registers. Any attempt by a program executing in user mode to access monitor memory or other users' memory results in a trap to the monitor, which treats the attempt as a fatal error

CPU Protection

- In addition to protecting I/O and memory, we must ensure that the operating system maintains control. We must prevent a user program from getting stuck in an infinite loop or not calling system services, and never returning 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 period may be fixed (for example, 1/60 second) or variable (for example, from 1 millisecond to 1 second). A variable timer is generally implemented by a fixed-rate clock and a counter.
- We can use the timer to prevent a user program from running too long. A simple technique is to initialize a counter with the amount of time that a program is allowed to run.
- A more common use of a timer is to implement time sharing. In the most straightforward case, the timer could be set to interrupt every N milliseconds, where N is the time slice that each user is allowed to execute before the next user gets control of the CPU. The operating system is invoked at the end of each time slice to perform various housekeeping tasks. This procedure is known as a context switch. Following a context switch, the next program continues with its execution from the point at which it left off.

Process Management

- A process can be thought of as a program in execution. A process needs certain resources to accomplish its task.
- A process is the unit of work in a system. Such a system consists of a collection of processes, some of which are operating-system processes others are user processes.
- All these processes can potentially execute concurrently, by multiplexing the CPU among them

Process Management

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

Main-Memory Management

- The main memory is central to the operation of a modern computer system. For a program to be executed, it must be mapped to absolute addresses and loaded into memory. 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 by whom.
  - Deciding which processes are to be loaded into memory when memory space becomes available.
  - Allocating and deallocating memory space as needed.

File Management

- For convenient use of the computer system, 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. A file is a collection of related information defined by its creator. These files are normally organized into directories to ease their use. The operating system is responsible for the following activities in connection with file management:
  - Creating and deleting files.
  - Creating and deleting directories.
  - Supporting primitives for manipulating files and directories.
  - Mapping files onto secondary storage.
  - Backing up files on stable (nonvolatile) storage media.

I/O System Management

- One of the purposes of an operating system is to hide the peculiarities of specific hardware devices. The O.S responsible for the following activities in connection with I/O system management:
  - A memory-management component that includes buffering, caching, and spooling.
  - A general device-driver interface.
  - Drivers for specific hardware devices.

Secondary-Storage Management

- the computer system must provide secondary storage to back up main memory because that are hold by MM are lost when power is switched of and the main memory is too small to accommodate all data and programs. The operating system is responsible for the following activities in connection with disk management:
    - Free-space management.
    - Storage allocation.
    - Disk scheduling.

Networking

- A distributed system collects physically separate, possibly heterogeneous, systems into a single coherent system, providing the user with access to the various resources that the system maintains. Access to a shared resource allows computation speedup, increased functionality, increased data availability, and enhanced reliability.

Protection System

- Protection is any mechanism for controlling the access of programs, processes, or users to the resources defined by a computer system. This mechanism must provide means for specification of the controls to be imposed and means for enforcement. Protection can improve reliability by detecting latent errors at the interfaces between component subsystems.

Command-Interpreter System

- Command-Interpreter System is the interface between the user and the operating System. Some of these Command-Interpreter System are user-friendly such as mouse-based window and- Menus. In other shells, commands are typed on a keyboard and displayed on a screen or printing terminal.

Operating System Services

- An operating system provides an environment for the execution of programs. It provides certain services to programs and to the users of those programs. The specific services provided, of course, differ from one operating system to another, but we can identify common classes. These operating-system services are provided for the convenience of the programmer, to make the programming task easier.
- 1-Program execution . 2- I/O operat.ion .
- 3- File-system manipulation. 4- Communication.
- 5- Error detection. 6- Resource allocation.
- 7- Accountimg. 8- Protection

System Calls

- System calls provide the interface between a process and the operating system. These calls are generally available as assembly-language instructions, and they are usually listed in the various manuals used by the assembly-language.

System Programs

- System programs provide a convenient environment for program development and execution. Some of them are simply user interfaces to system calls; others are considerably more complex. They can be divided into these categories:
  - File management.
  - Status information.
  - File modification.
  - Programming-language support.
  - Program loading and execution.
  - Communications.

System Structure

- A system as large and complex as a modern operating system must be engineered carefully if it is to function properly and be modified easily. There are three different system structures:
  - Simple Structure.
  - Layered Approach.
  - Microkernels.

System Design and Implementation

- The problems and steps of system design and implementation are as follows:
  - Design Goals.
  - Mechanisms and Policies.
  - Implementation.

End of chapter two