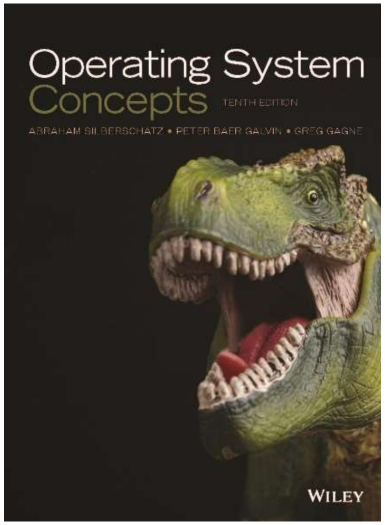
Chapter 1: Introduction

References

Operating System Concepts, 10th Edition

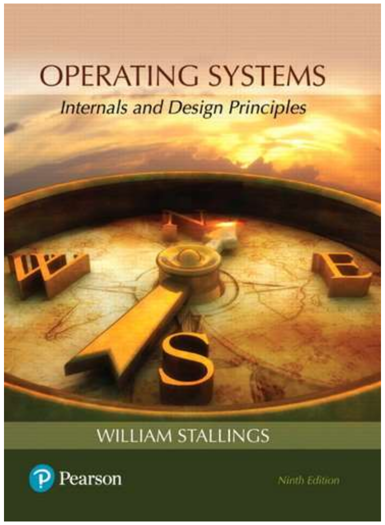
https://www.os-book.com/OS10/

Avi Silberschatz, Peter Baer Galvin, and Greg Gagne



Operating Systems: Internals and Design Principles, 9th Edition

William Stallings



What is an Operating System?

Silberschatz:

- Is a (most important) software that manages a computer’s hardware.
- A program that acts as an intermediary between a **user** of a computer and the **computer hardware** includes the CPU, memory, storage and I/O devices.
- A fundamental responsibility of an operating system is to **allocate** these hardware resources to programs.

Stalling:

- A program that controls the execution of application programs
- An interface between applications and hardware

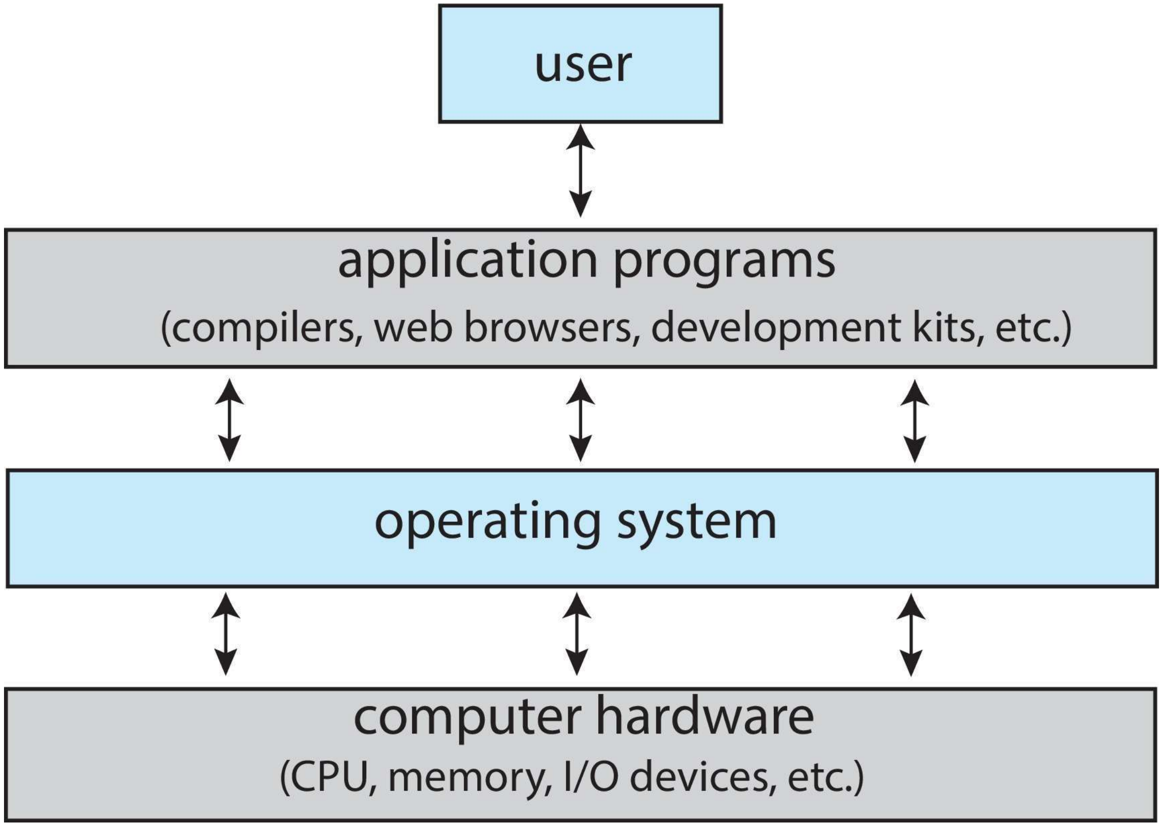
Operating System Goals

- Execute user programs and make solving user problems **easier**
- Make the computer system **convenient** to use
- Use the computer hardware in an **efficient** manner

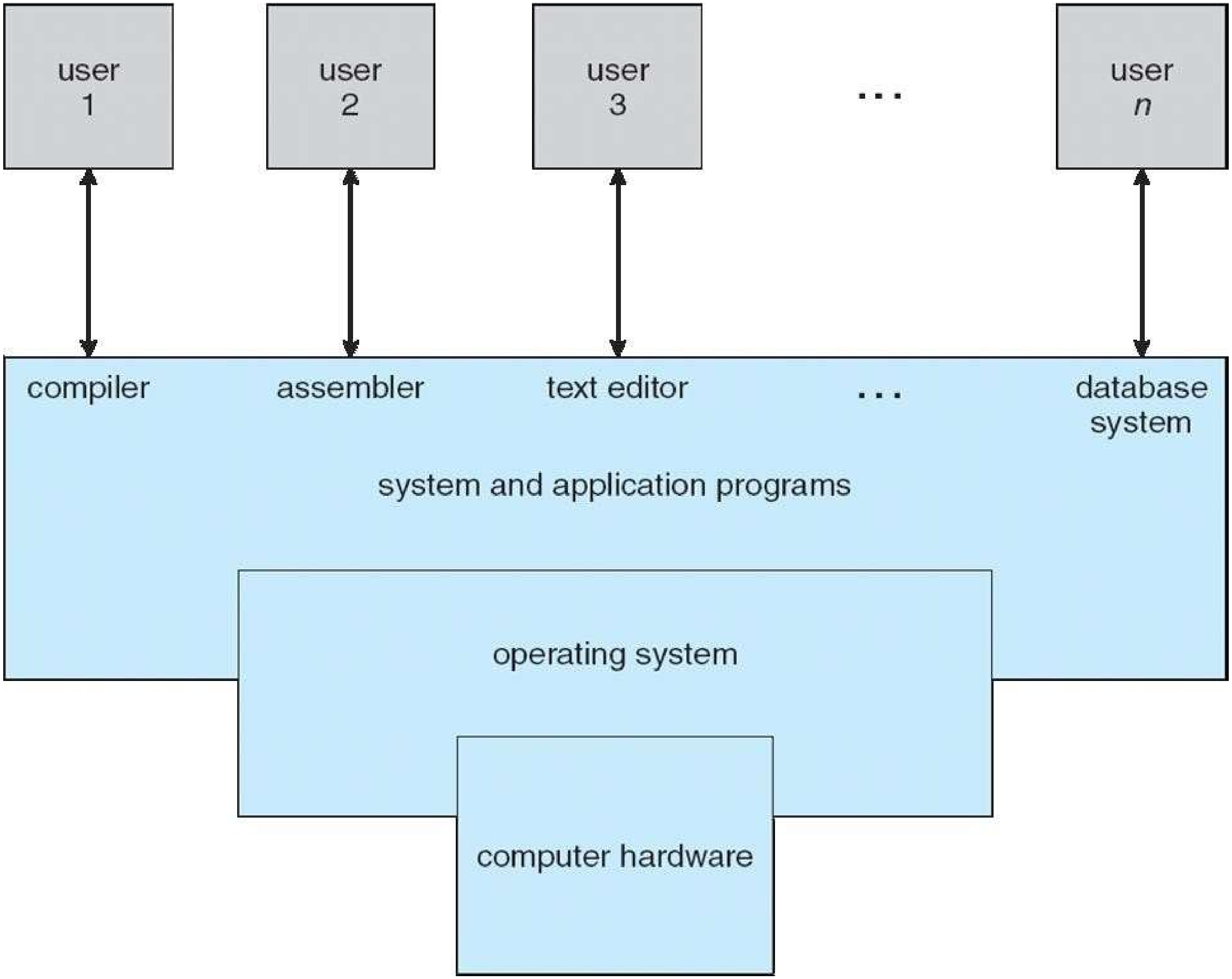
Computer System Structure

- Computer system can be divided into **four** components:
  - **Hardware** – provides basic computing resources
    - CPU, memory, 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, compilers, web browsers, database systems, video games
  - **Users**
    - People, machines, other computers

Abstract View of Components of Computer


user
application programs
(compilers, web browsers, development kits, etc.)
operating system
computer hardware
(CPU, memory, I/O devices, etc.)

Components of a Computer System



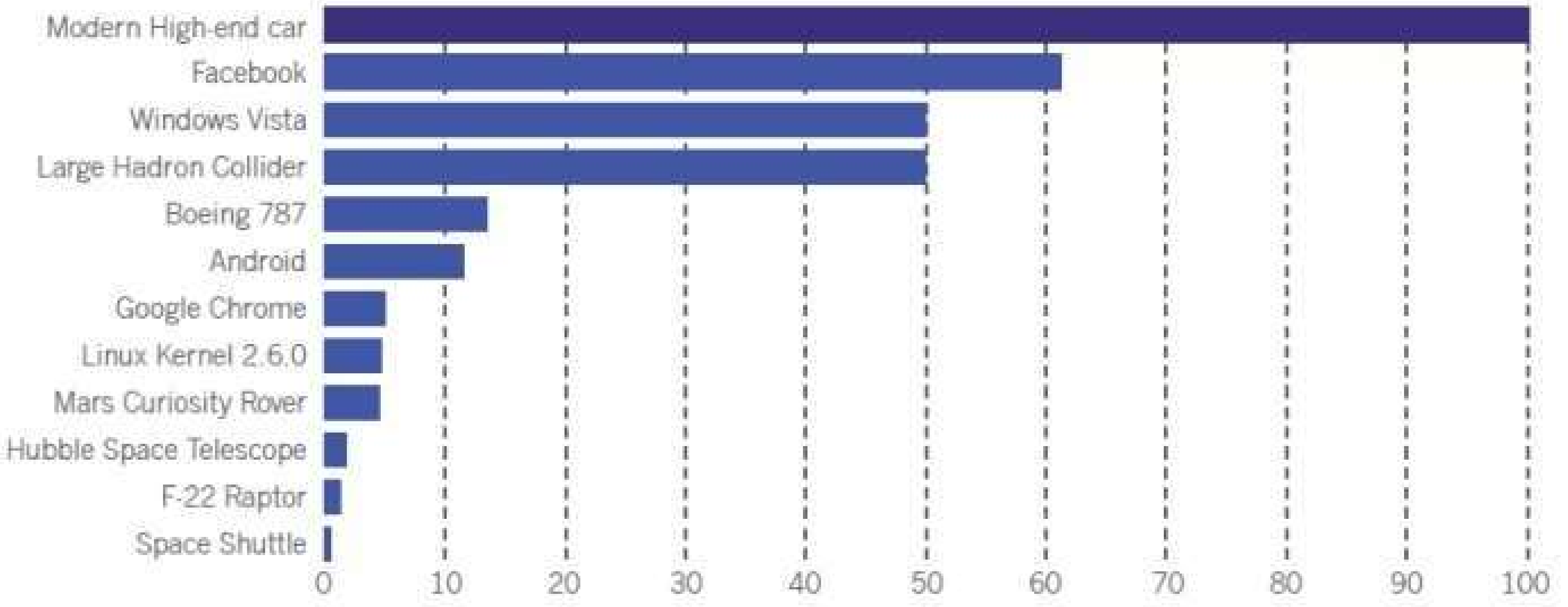
What is an Operating System?

- When we say “computers” run operating systems, we don’t just mean traditional desktop PCs and laptops.
- The smartphones is a computer, as are tablets, smart TVs, game consoles, smart watches, and Wi-Fi routers.
- Other devices, such as your Wi-Fi router, may run “**embedded operating systems**.” These are specialized operating systems with fewer functions than a typical operating system, designed specifically for a single task—like running a Wi-Fi router, providing GPS navigation, or operating an ATM.

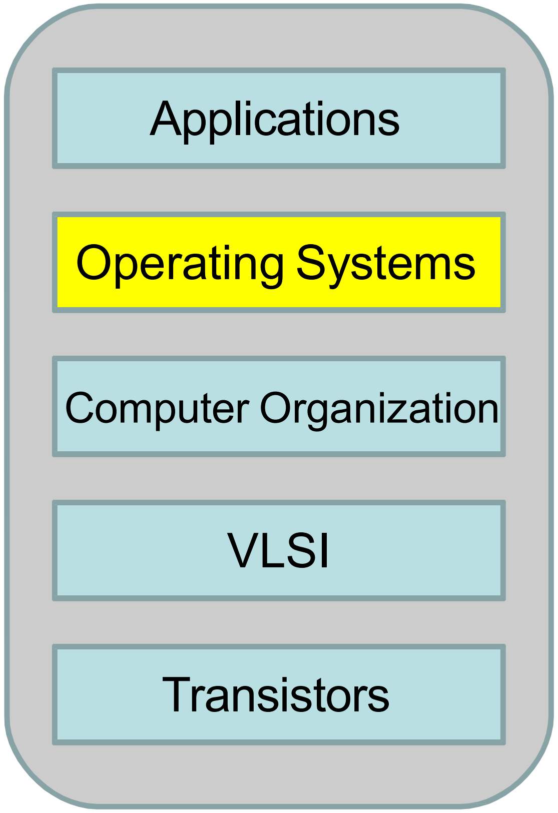
Modern OS is Complex

SOFTWARE SIZE (MILLION LINES OF CODE)

Source: NASA, IEEE, Wired, Boeing, Microsoft, Linux Foundation, Ohloh



The Layers in Systems



OS Definition (Silberschatz)

Silberschatz:

OS is a **resource allocator**

- Manages all resources
- Decides between conflicting requests for efficient and fair resource use

OS is a **control program**

- Controls execution of programs to prevent errors and improper use of the computer

Stallings:

An OS is a program that controls the execution of application programs, and acts as an interface between applications and the computer hardware.

OS Usage

- Hardware Abstraction (User/Computer Interface)

  Turns hardware into something that applications can use

- Resource Management

  manage system’s resources

OS as a User/Computer Interface

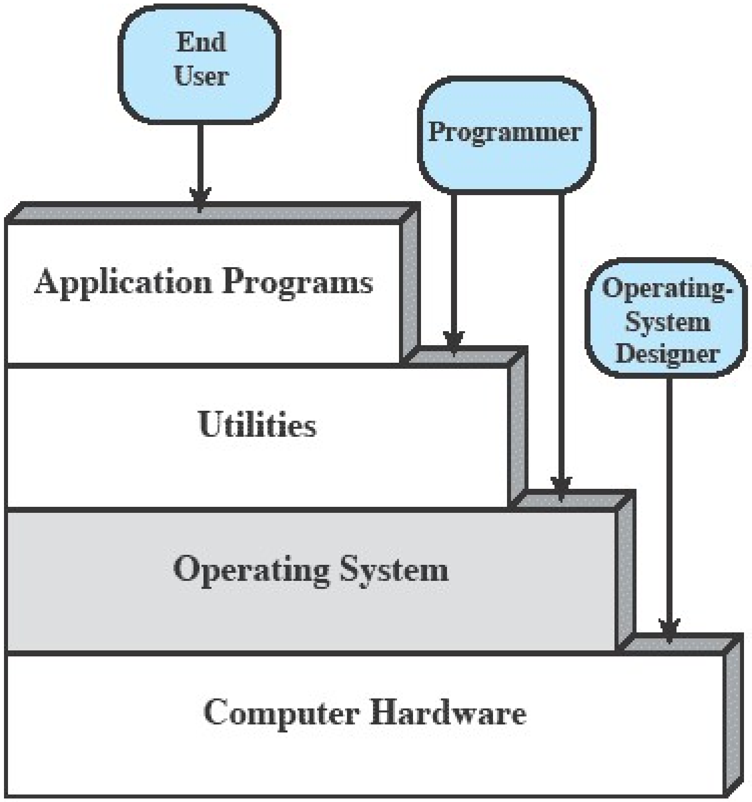
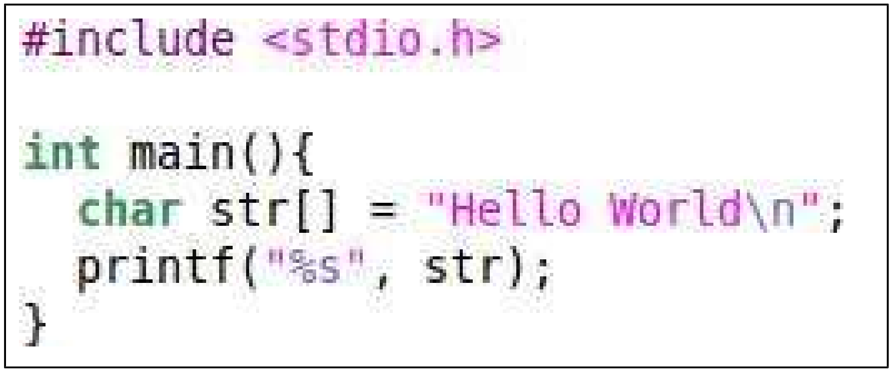


Figure 2.1 Layers and Views of a Computer System

A Simple Program

What is the output of the following program?

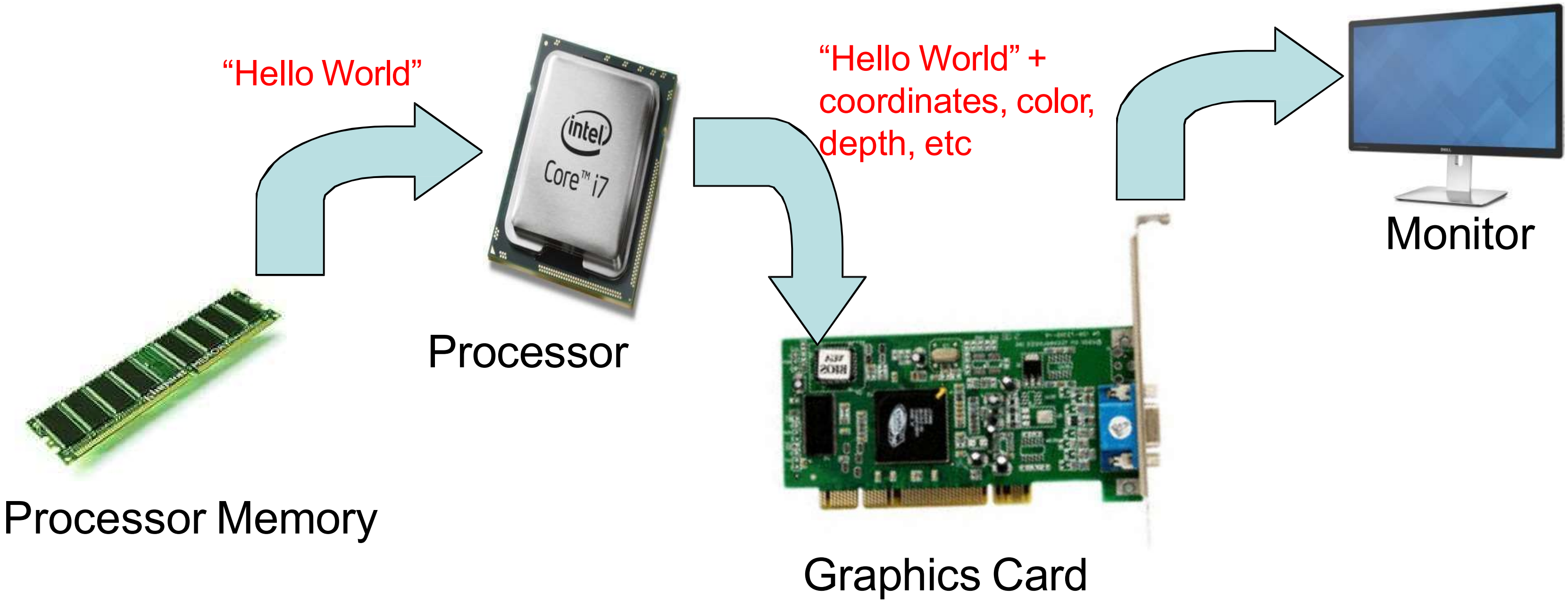


```c
#include <stdio.h>

int main(){
  char str[] = "Hello World\n";
  printf("%s", str);
}
```

How is the string displayed on the screen?

Displaying on the Screen



Processor Memory

“Hello World”

Processor

“Hello World” + coordinates, color, depth, etc

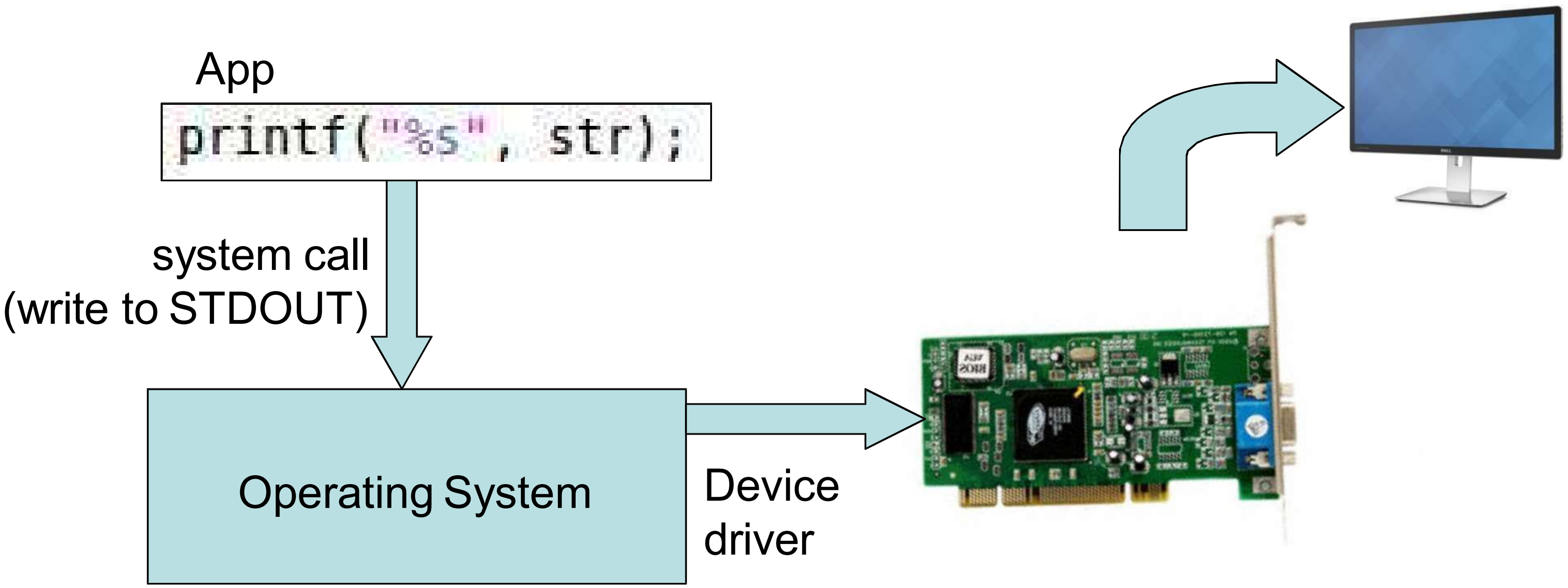
Graphics Card

Monitor

- Can be complex and tedious
- Hardware dependent

Without an OS, all programs need to take care of every nitty gritty detail

Operating Systems Provide Abstraction



```
printf("%s", str);
```

- **Easy to program** apps
  - No more nitty gritty details for programmers
- **Reusable functionality**
  - Apps can reuse the OS functionality
- **Portable**
  - OS interfaces are consistent. The app does not change when hardware changes

OS as Resource Manager

- OS must manage CPU, memory, network, disk etc…
- Resource Management
  - allows multiple apps to share resources
  - protects apps from each other
  - improves performance by efficient utilization of resources

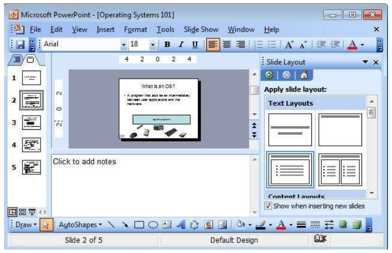
OS as a Resource Manager

- Multiple apps but limited hardware resources

Apps

```
#include <stdio.h>

int main(){
  char str[] = "Hello World\n";
  printf("%s", str);
}
```



Operating Systems

A few processors

What Operating Systems Do

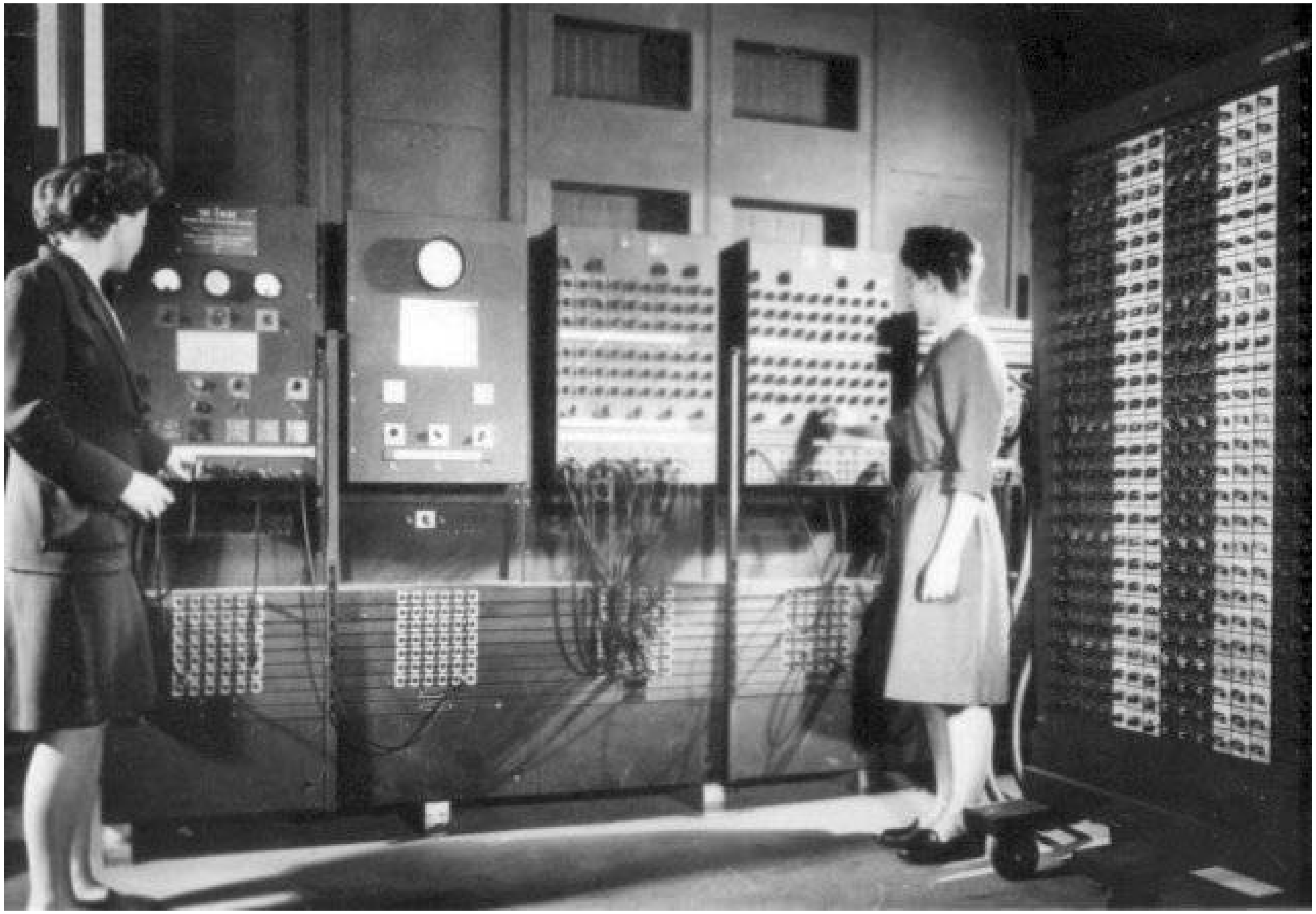
- Depends on the point of view
- Users want convenience, **ease of use** and **good performance**
  - Don’t care about **resource utilization**
- But shared computer such as **mainframe** or **minicomputer** must keep all users happy
  - Operating system is a **resource allocator** and **control program** making efficient use of HW and managing execution of user programs
- Users of dedicate systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- Mobile devices like smartphones and tables are resource poor, optimized for usability and battery life
  - Mobile user interfaces such as touch screens, voice recognition
- Some computers have little or no user interface, such as embedded computers in devices and automobiles
  - Run primarily without user intervention

Evolution of Operating Systems

1. The Evolution of Operating Systems
    1. Serial Processing: No Operating Systems
    2. Simple Batch Systems: Monitor
    3. Multiprogrammed Batch Systems
    4. Time-Sharing Systems

Serial Processing (1940s-1950s)

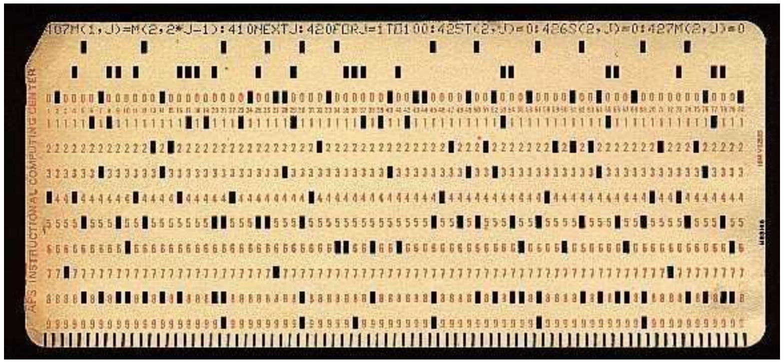
- Electronic Numerical Integrator and Computer (ENIAC)



https://en.wikipedia.org/wiki/ENIAC

Vacuum Tubes

- Hardware
  - Vacuum tubes and IO with punchcards
  - Expensive and slow
- User Apps
  - Generally straightforward numeric computations done in machine language



```
407M(1,J)=M(2,2*J-1):410NEXTJ:420FORJ=1TO100:425T(2,J)=0:426S(2,J)=0:427M(2,J)=0
```

IBM Punch card

ENIAC

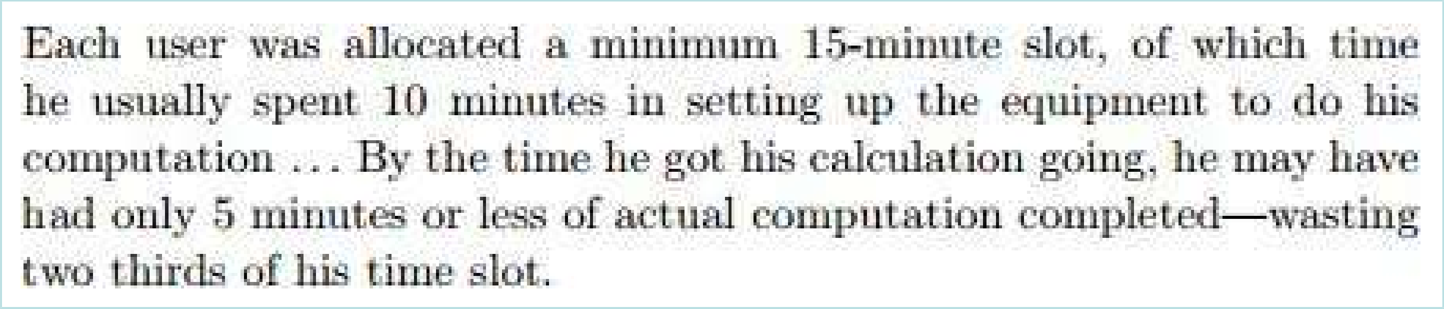
Serial Processing

- The user have to access the system in series.
- The programmer interacted directly with the computer hardware – No OS
- Machines run from a console with display lights, toggle switches, input device, and printer
- Problem
  - Scheduling: most installations used a hardcopy sign-up sheet to reserve computer time
  - Setup time: includes loading the compiler, source program, saving compiled program, and loading and linking

Serial Processing

- OS: Unheard of
- Human feeds program and prints output

George Ryckman, on IBM’s first computer



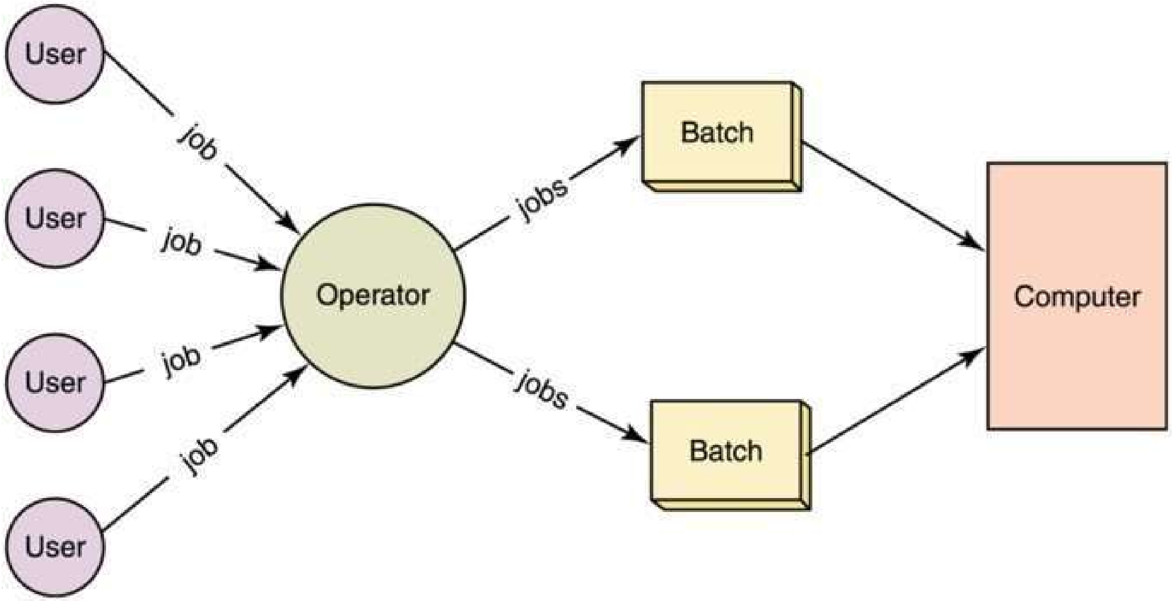
> Each user was allocated a minimum 15-minute slot, of which time he usually spent 10 minutes in setting up the equipment to do his computation . . . By the time he got his calculation going, he may have had only 5 minutes or less of actual computation completed—wasting two thirds of his time slot.

The cost of wastage was $146,000 per month (in 1954 US Dollars)

Simple Batch Systems

- The central idea behind the simple batch processing scheme is
  - Batch: made up by many jobs from users (batch jobs together)
    - A job may use several program
  - The use of a piece of software known as the monitor that controls the sequence of events.
  - Program returns control to monitor when finished
  - Job Control Language (JCL):
    - Special type of programming language
    - Provides instruction to the monitor
      - What compiler to use
      - What data to use

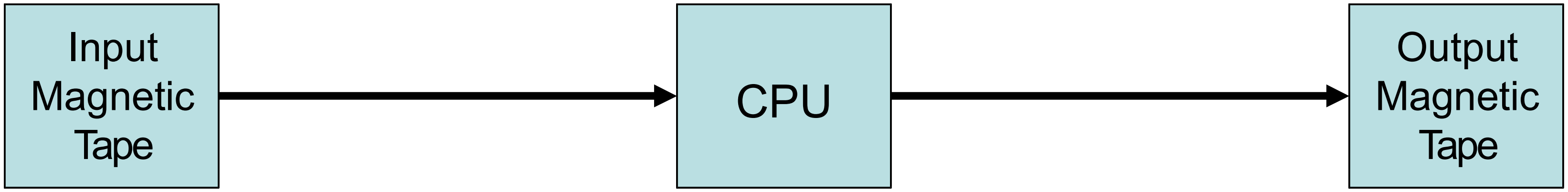
Simple Batch Systems



- Operator collects jobs (through punch cards) and feeds it into a magnetic tape drive
- Special Program reads a job from input tape drive and on completion writes result to output tape drive
- The next program is then read and executed
- Printing was done offline

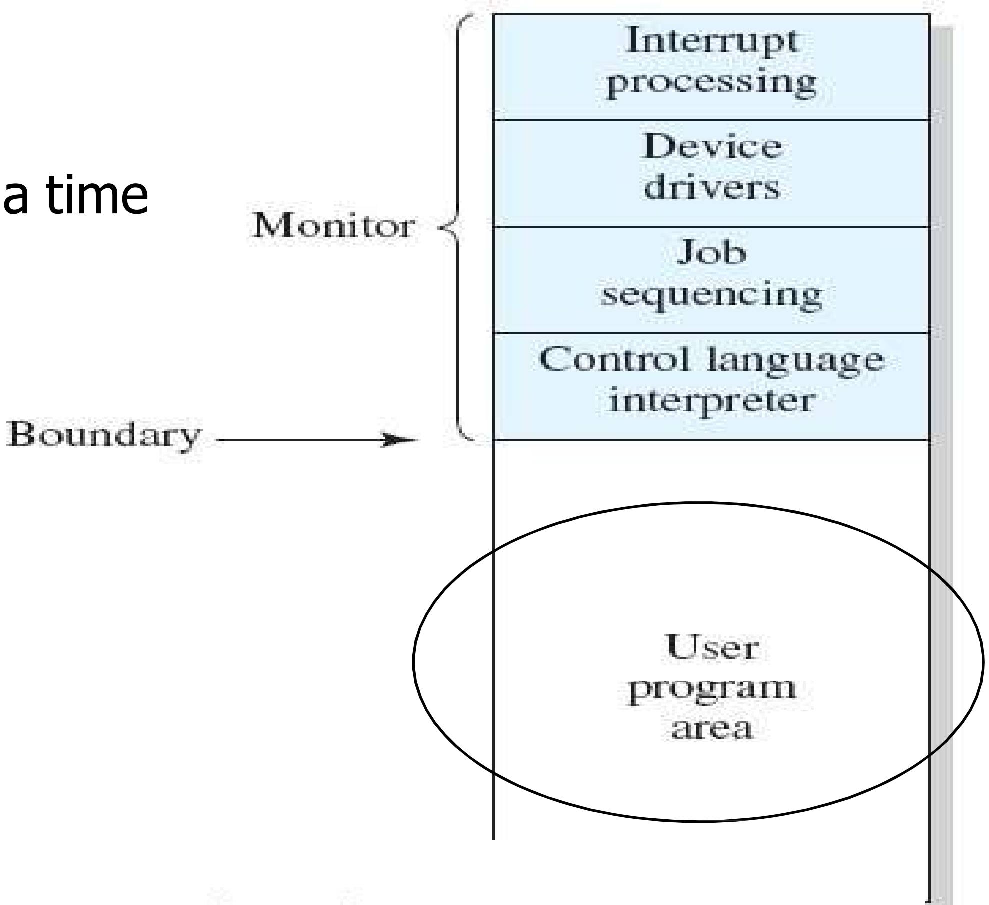
Simple Batch Systems

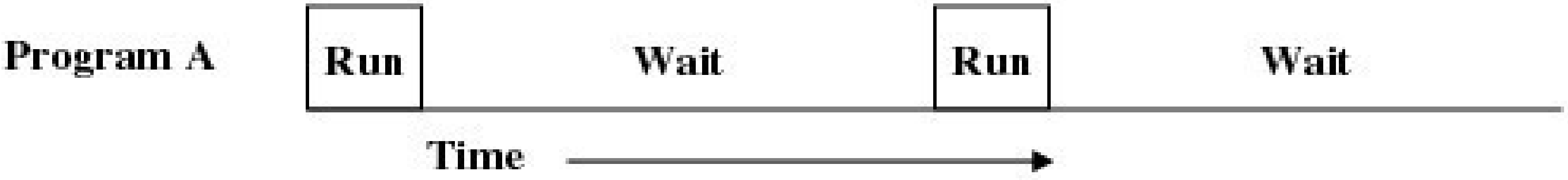
- Pros: Better utilization of machine (keep machine busy)
- Cons:
  - In Batch Systems execute time includes reading from input and writing to output.
  - I/O considerably **slower** than execution
    - Magnetic tapes were best read sequentially
  - Therefore programmer must **wait** for long time



Multiprogrammed Batch Systems

- Simple Batch Systems
  - Only one job in memory at a time

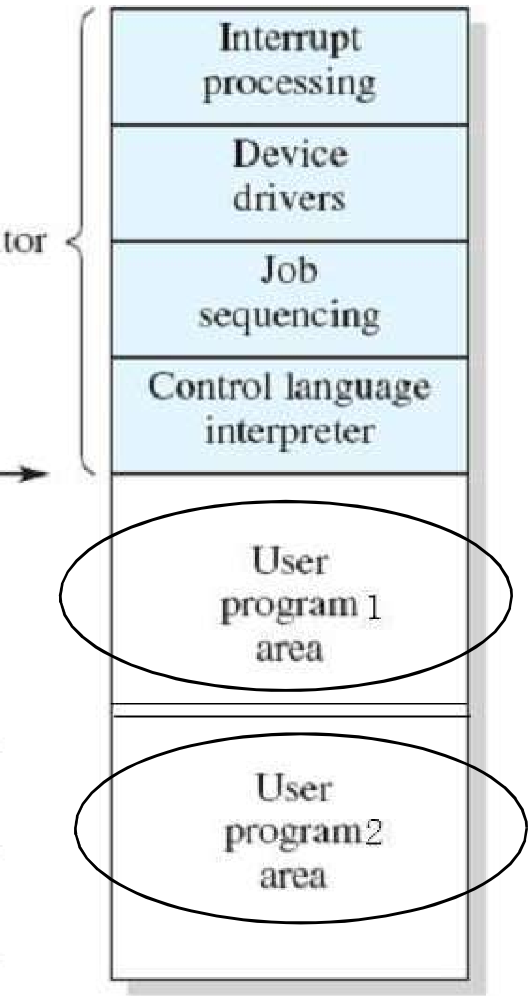


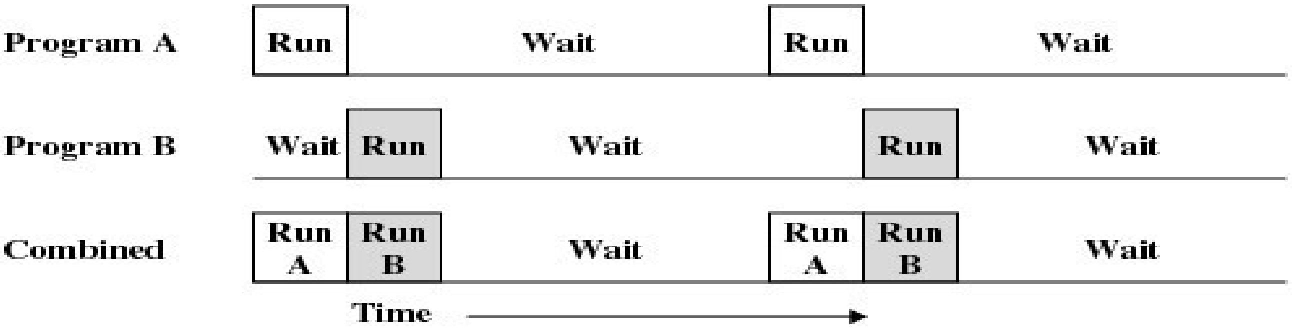


(a) Uniprogramming

Multiprogrammed Batch Systems

- Multiprogrammed Batch Systems
  - Multiple jobs in memory at a time
- Why ?
  - I/O devices are slow compare to the CPU
    - CPU is not busy





(b) Multiprogramming with two programs

Multiprogramming

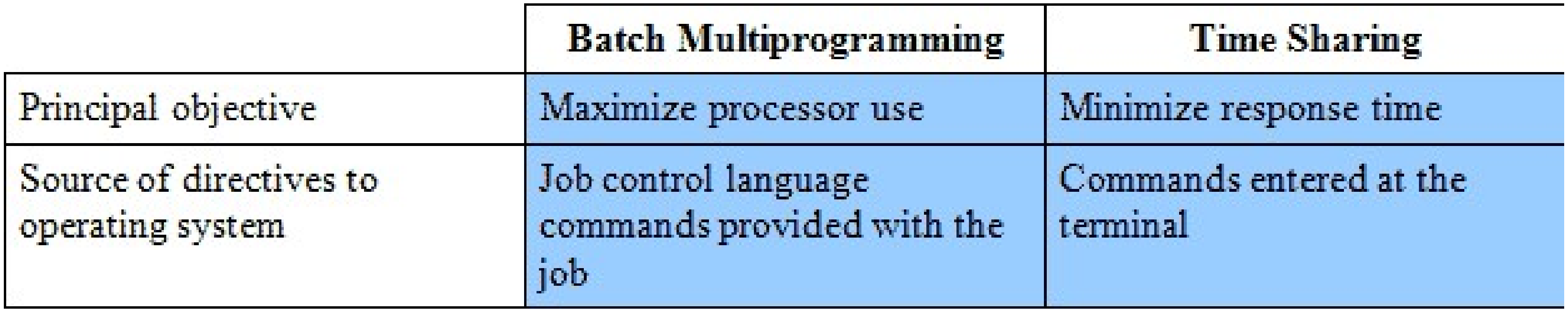
- Processor has more than one program to execute
- The sequence in which programs are executed depend on their relative priority and whether they are waiting for I/O
- After an interrupt handler completes, control may not return to the program that was executing at the time of the interrupt

Multiprogrammed Batch Systems

- Uniprogramming
  - Processor must wait for I/O instruction to complete before preceding
- Multiprogramming
  - When one job needs to wait for I/O, the processor can switch to the other job
  - Also known as Multitasking

Time-Sharing Systems

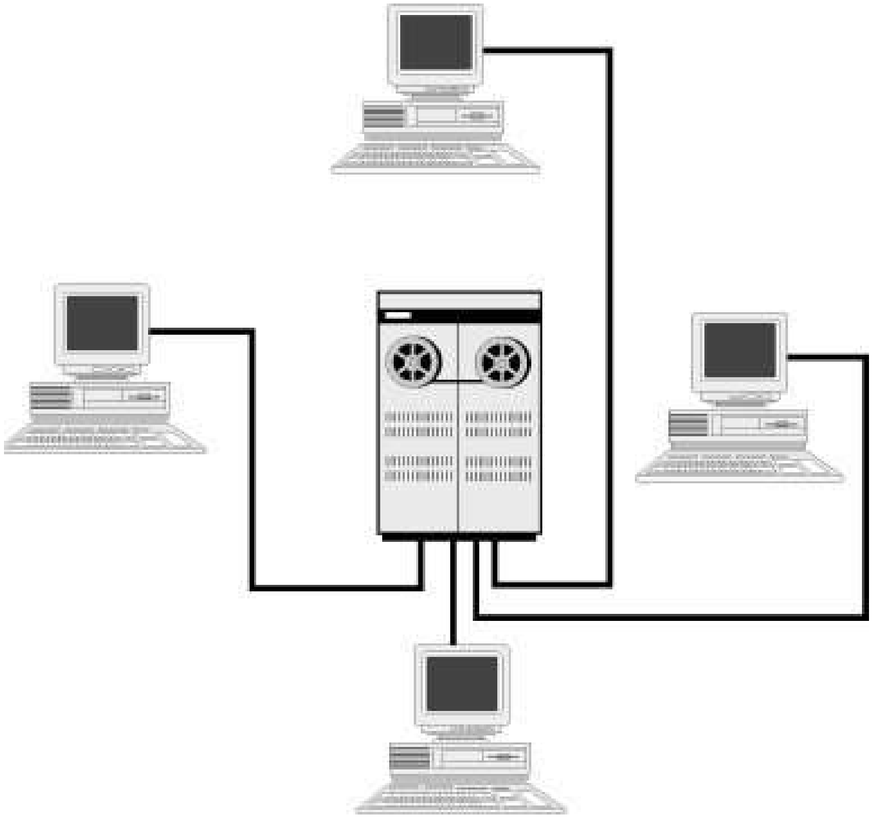
- Why?
  - Background: 1960s
    - Mainframe Computer System:
    - Multiple users simultaneously access the system through terminals
  - Requirement: handle multiple interactive users/jobs by using multiprogramming
- Idea
  - Processor's time is shared among multiple users/jobs



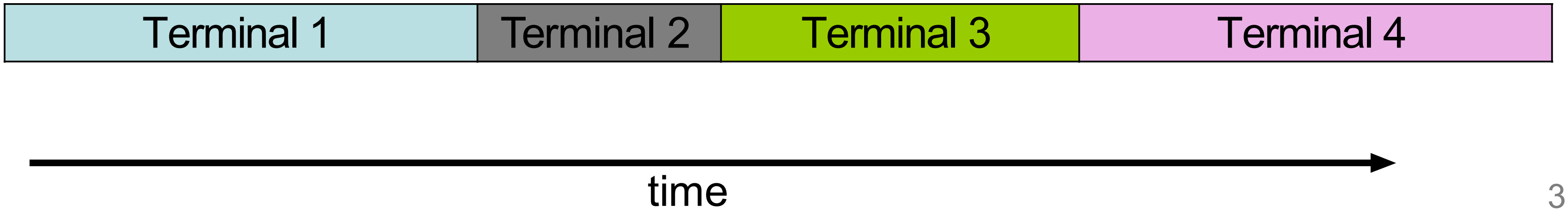
	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

Time-Sharing Systems - mainframe

Timesharing



Who uses the CPU?



Terminal 1

Terminal 2

Terminal 3

Terminal 4

time

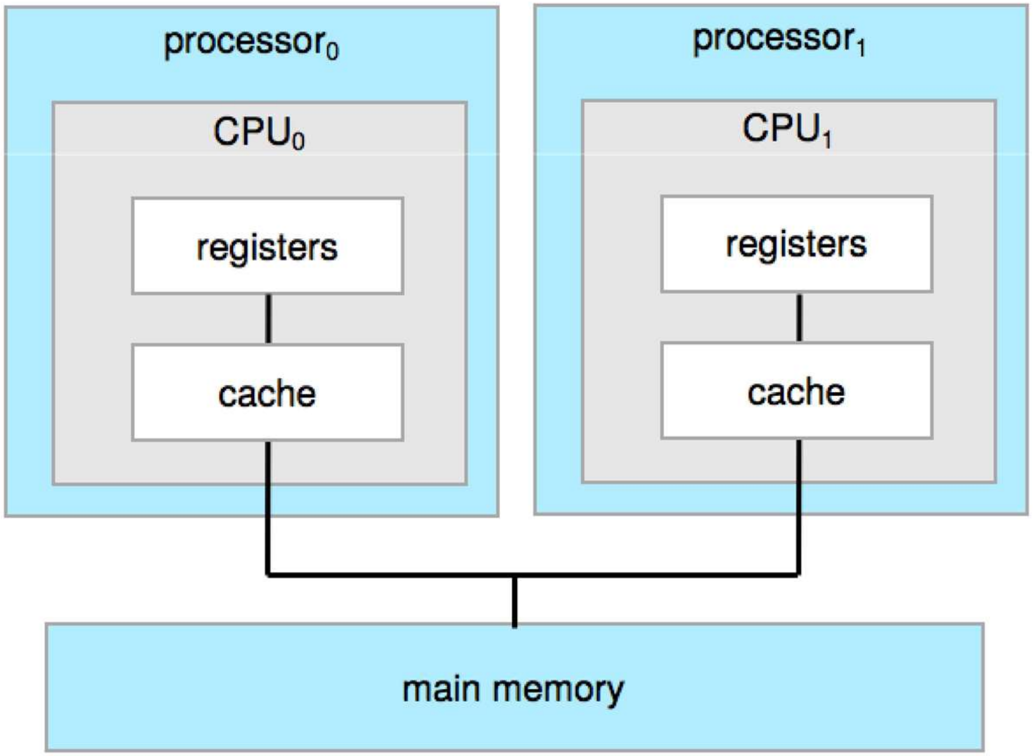
Personal Computers

- Hardware
  - VLSI ICs
- User Programs
  - High level languages
- Operating Systems
  - Multi tasking
  - More complex memory management and scheduling
  - Synchronization
  - Examples : Windows, Linux, etc

Computer-System Architecture

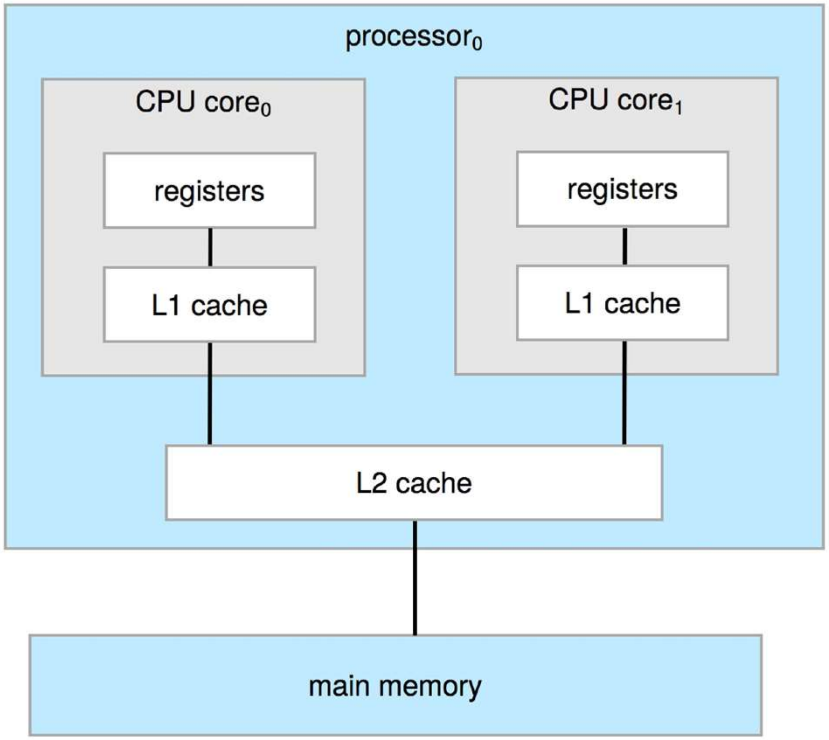
- Most systems use a single general-purpose processor
  - Most systems have special-purpose processors as well
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - There are multiple processors
  - These processors share same main memory and I/O facilities
  - Advantages include:
    1. **Increased throughput**
    2. **Economy of scale**
    3. **Increased reliability** – fault tolerance
  - Two types:
    1. **Asymmetric Multiprocessing** – each processor is assigned a specie task.
    2. **Symmetric Multiprocessing** – each processor performs all tasks

Symmetric Multiprocessing (SMP)



Dual-Core Design

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



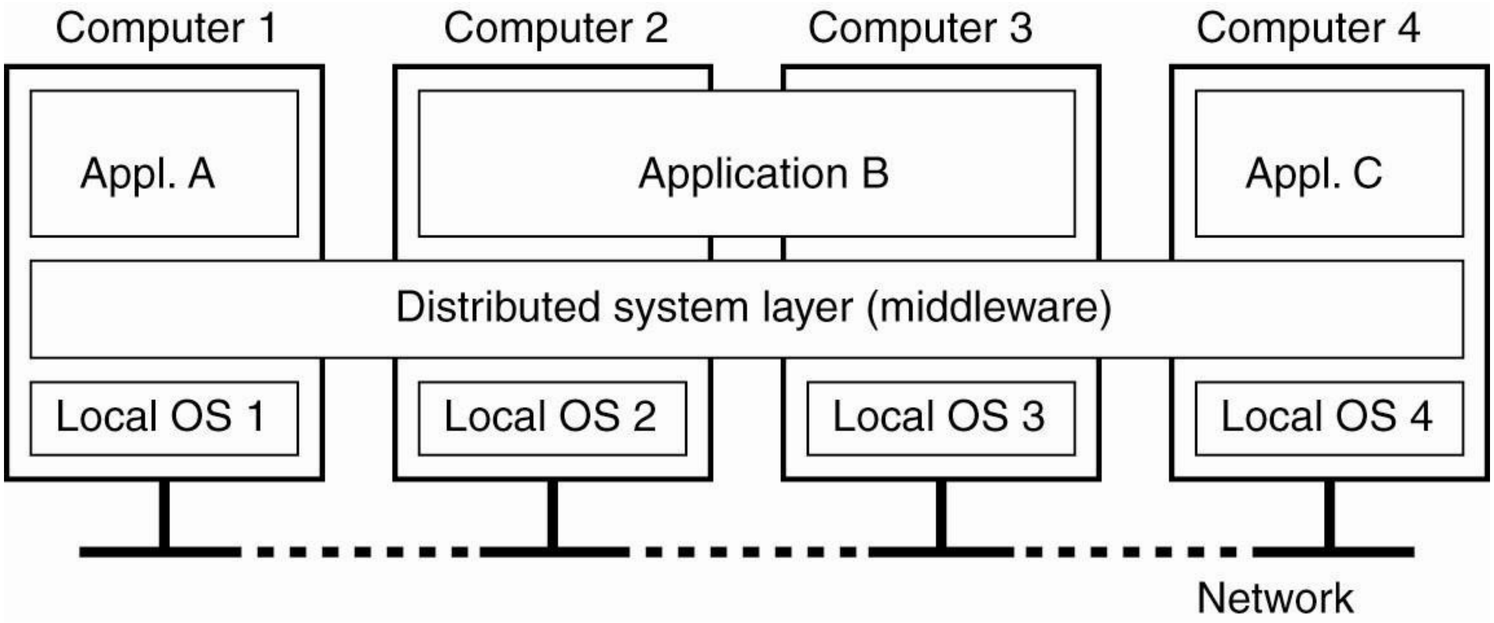
Advanced OS

- Distributed OS
- Network OS
- Real-time OS
    - Each process has a deadline and must be finished by its **deadline**

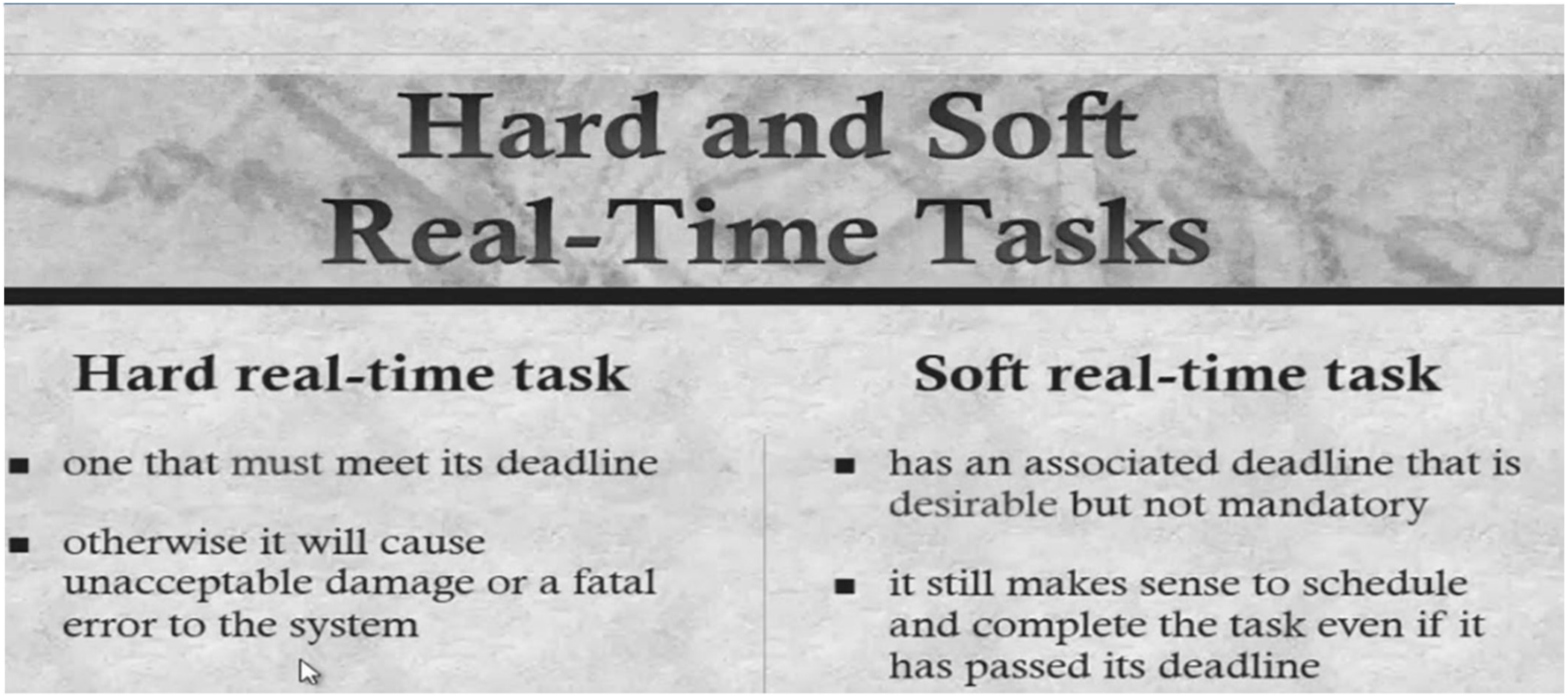
Distributed systems

A collection of independent computers that appears to its users as a single coherent system.

Example: High Performance Computing (HPC) clusters.



Real-Time Systems



Hard and Soft Real-Time Tasks

Hard real-time task

- one that must meet its deadline
- otherwise it will cause unacceptable damage or a fatal error to the system

Soft real-time task

- has an associated deadline that is desirable but not mandatory
- it still makes sense to schedule and complete the task even if it has passed its deadline

Hard deadline examples: nuclear power plant control system, weapon system, flight air traffic control system

Soft deadline example: multimedia systems (film = 24-32 frames per second)

Overview of Computer System Structure

Computer System Organization

- □ You should know the structure of computer systems from COMP 222: Computer Organization course.
- □ Review the required concepts from Chapter 1: Computer System Overview, Stallings.

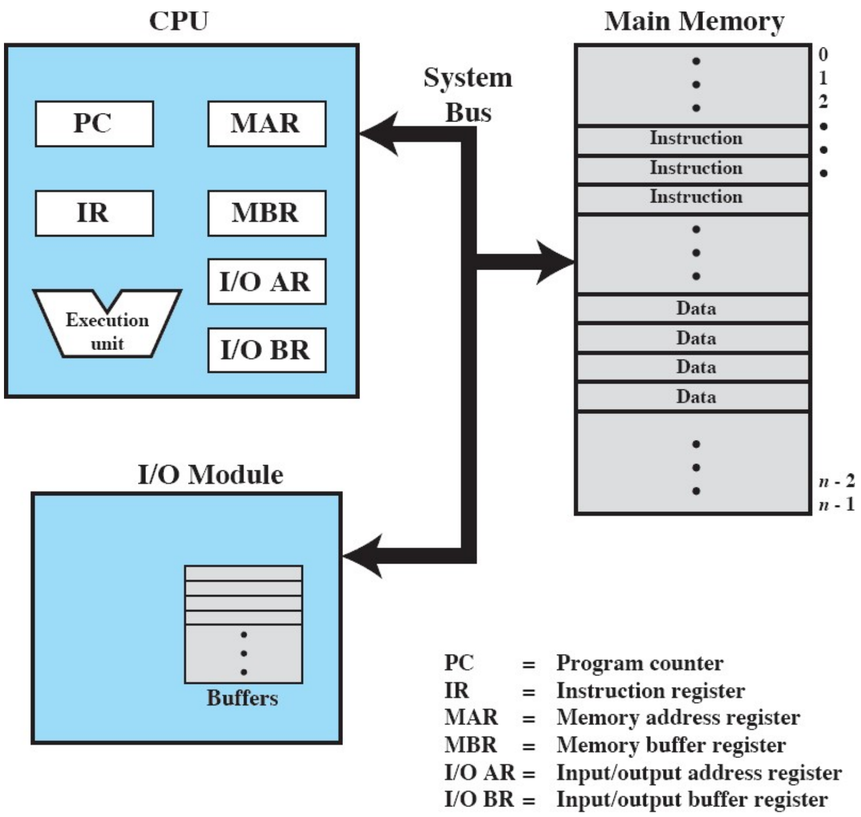
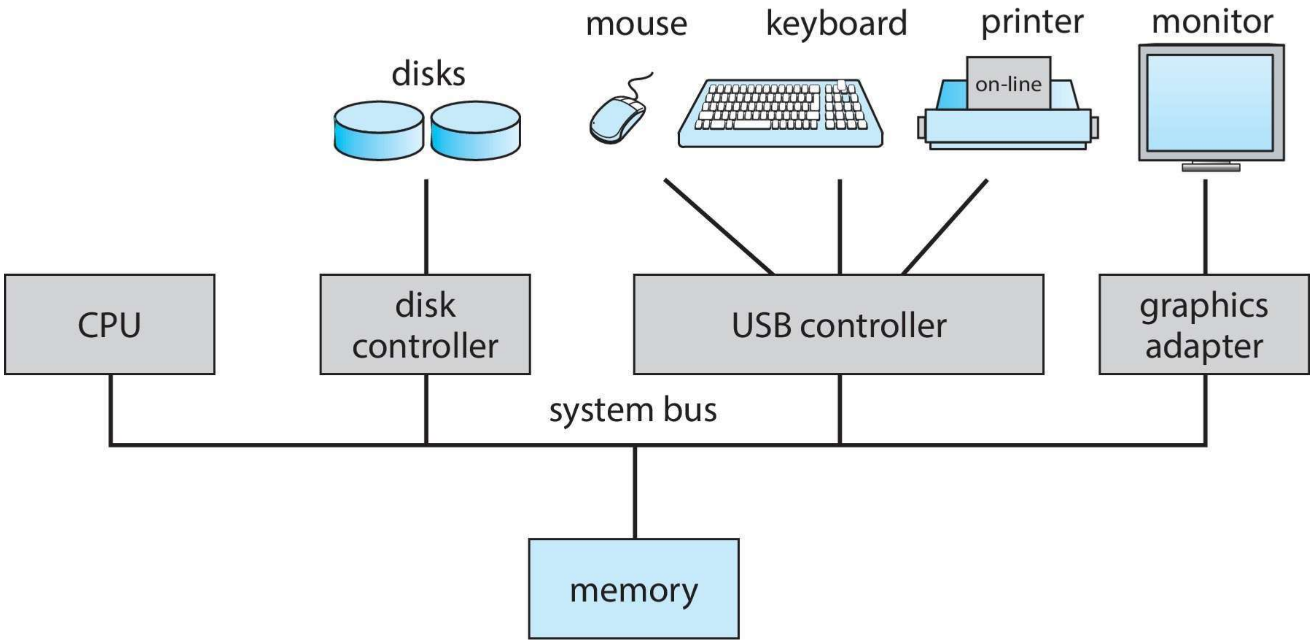


Figure 1.1 Computer Components: Top-Level View

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 devices competing for memory cycles



Computer-System Operation

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

Common Functions of Interrupts

- Interrupt architecture must save the address of the interrupted instruction
- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request
- An operating system is **interrupt driven**

Interrupts

- Mechanism by which other modules may interrupt the normal sequencing of the processor
- Provided to improve processor utilization
  - Most I/O devices are slower than the processor
  - Processor must pause to wait for device
  - Wasteful use of the processor

Classes of Interrupts

Program Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.

Timer Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.

I/O Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.

Hardware failure Generated by a failure, such as power failure or memory parity error.

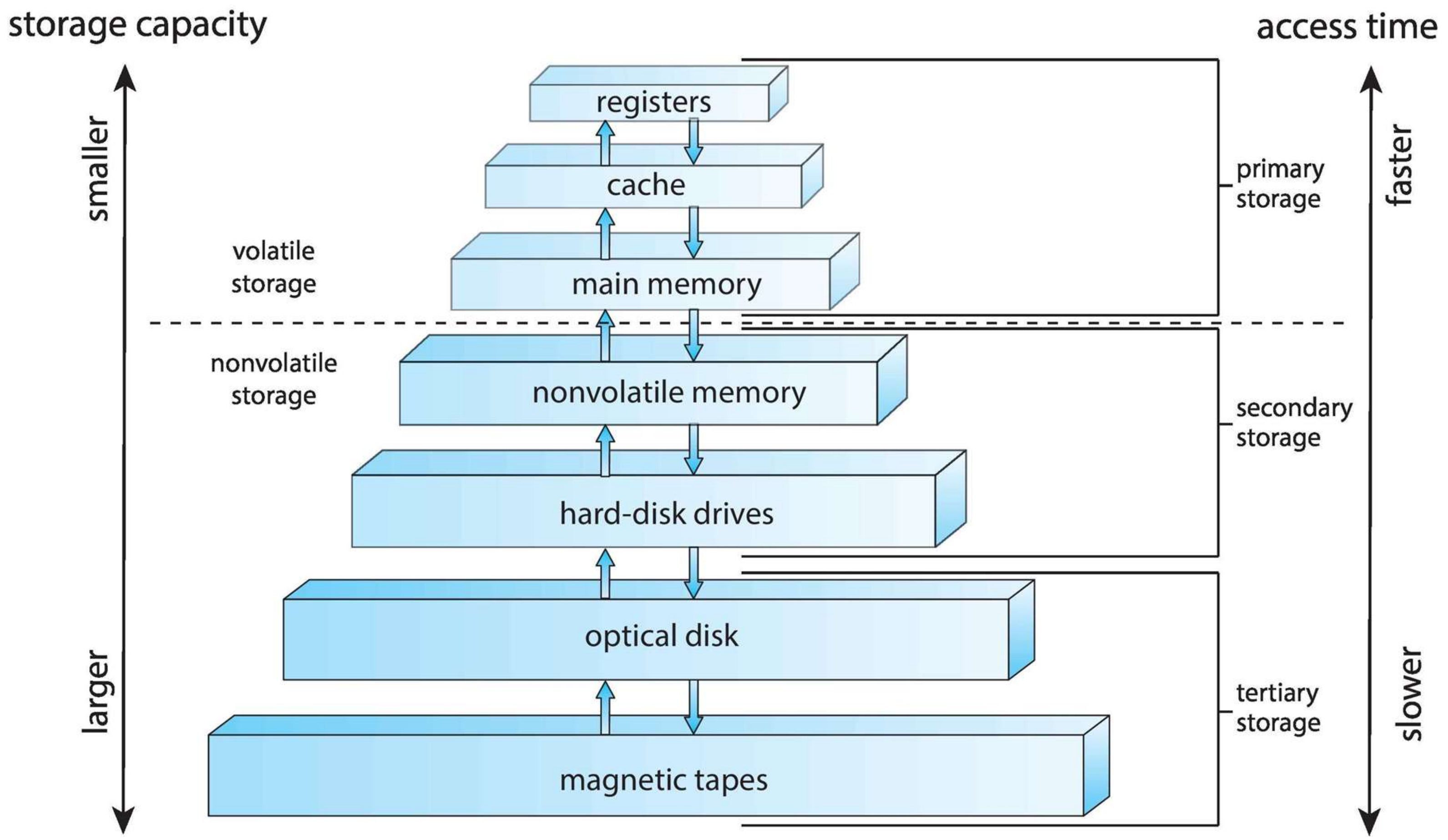
Interrupt Handling

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

Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- **Caching** – copying information into faster storage system; main memory can be viewed as a cache for secondary storage

Storage-Device Hierarchy



Caching

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

Operating System Operations

- Computer Startup
  - **bootstrap program** is loaded at power-up
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution
- Kernel
  - Portion of operating system that is in main memory
  - Contains most frequently used functions

Operating System Operations

- The operating system then starts executing the first process, 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

- Modern operating systems are interrupt driven.
  - Events are signaled by the occurrence of an interrupt

- Types of Interrupts
  - **Hardware** Interrupts: hardware sending a signal to the CPU, usually by way of the system bus.
  - **Software** Interrupts (exception or traps): software may trigger an interrupt by executing a special operation called a **system call**, or by causing an error.

Operating-System Operations

Software interrupt (**exception** or **trap**):

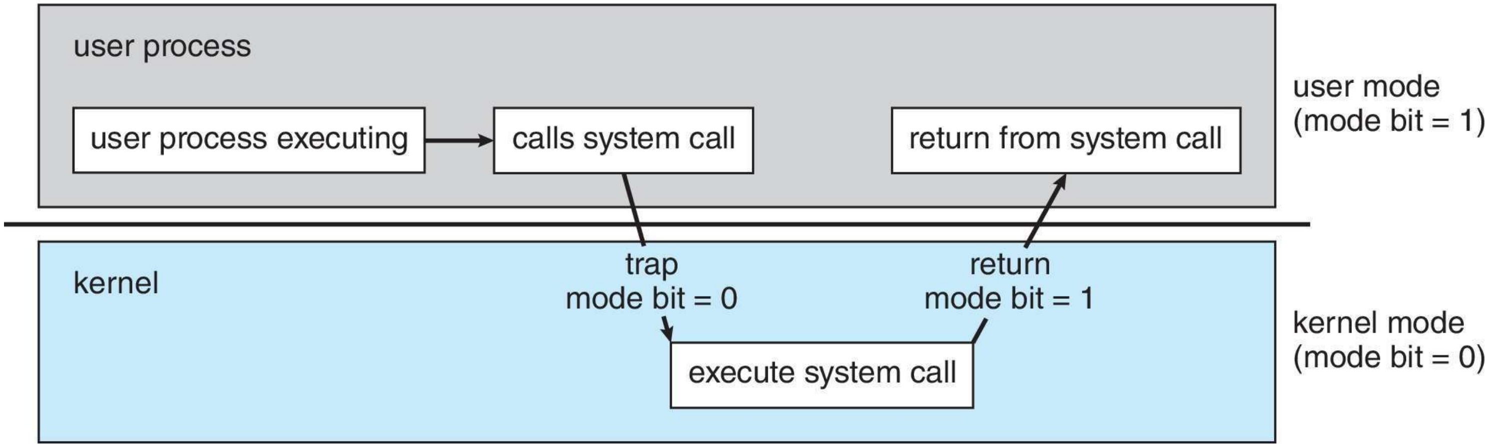
- Software error (e.g., division by zero)
- Request for operating system service – **system call**
- Other process problems include infinite loop, processes modifying each other or the operating system

Dual-mode Operation

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode** (supervisor mode, system mode or privileged mode)
  - In user mode, there is restricted access to the HW resources, however in kernel mode, there is full access to the HW resources.

- **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
  - How do we guarantee that user does not explicitly set the mode bit to “kernel”?
    - System call changes mode to kernel, return from call resets it to user

Transition from User to Kernel Mode



Timer

- It prevents a user program from getting stuck in an infinite loop or not calling system services and never returning control to the operating system.
- A timer can be set to interrupt the computer after a specified period.
- Before turning over control to the user, the operating system ensures that the timer is set to interrupt.

Operating System Components

- Process Management
- Memory Management
- Storage Management
  - File-System Management
  - Mass-Storage Management
  - Caching
  - I/O Systems
- Protection and Security

Process Management

- A process is a program in execution.
- It is a unit of work within the system.
  - The entity that can be assigned to and executed on a processor
- An instance of a program running on a computer
- Program is a passive entity, process is an active entity.
- Process needs resources to accomplish its task.
  - CPU, memory, I/O, files

Process Management Activities

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

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

Memory Management

- For a program to be executed, it must be mapped to absolute addresses and loaded into memory.
- As the program executes, it accesses program instructions and data from memory by generating these absolute addresses.
- Eventually, the program terminates, its memory space is declared available.
- General-purpose computers keep several programs in memory (for multiprogramming and time-sharing), creating a need for memory management.

Memory Management

- Memory management determines what is in memory and when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Storage Management

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

Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a "long" period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Mounting and unmounting
  - Free-space management
  - Storage allocation
  - Disk scheduling
  - Partitioning
  - Protection

I/O Subsystem

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

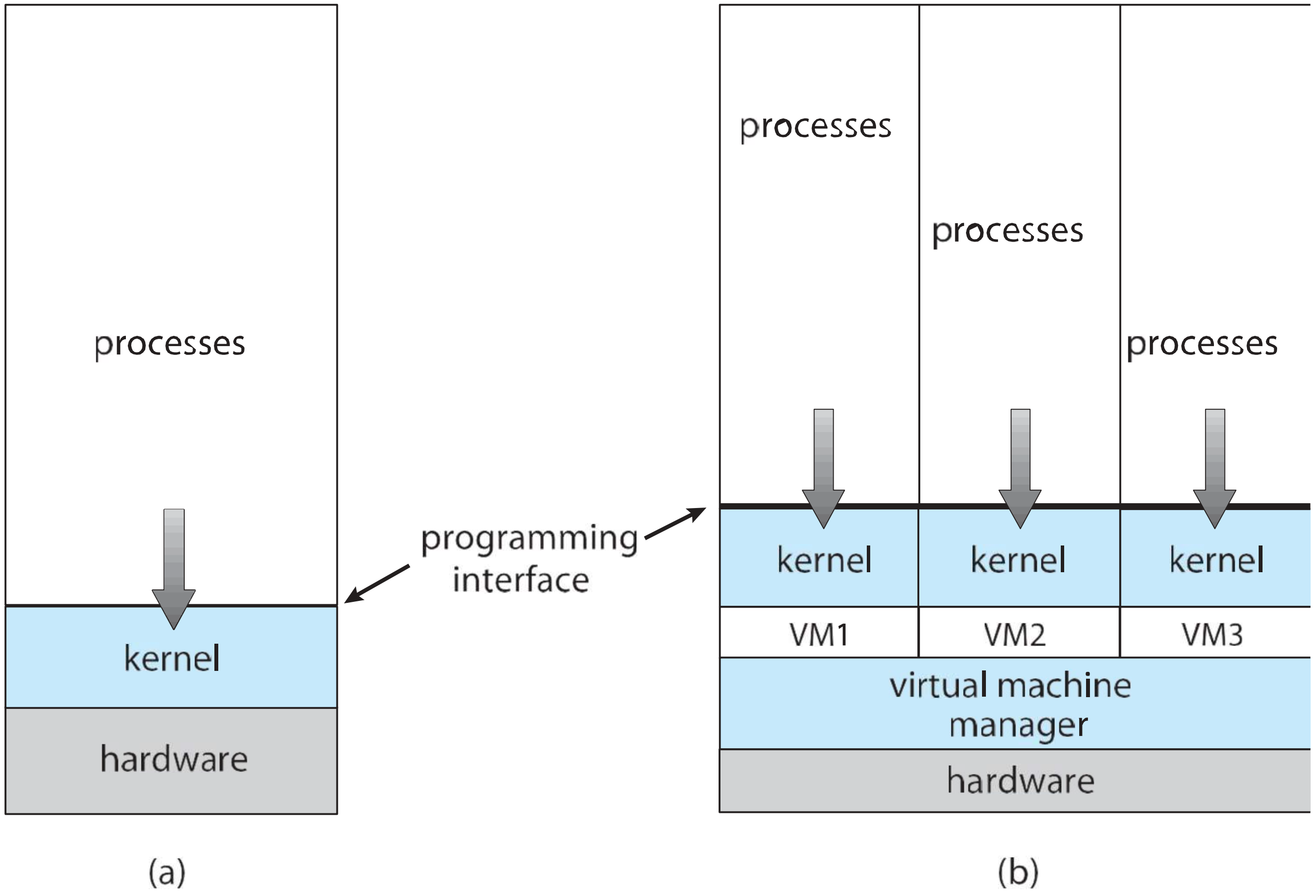
Protection and Security

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

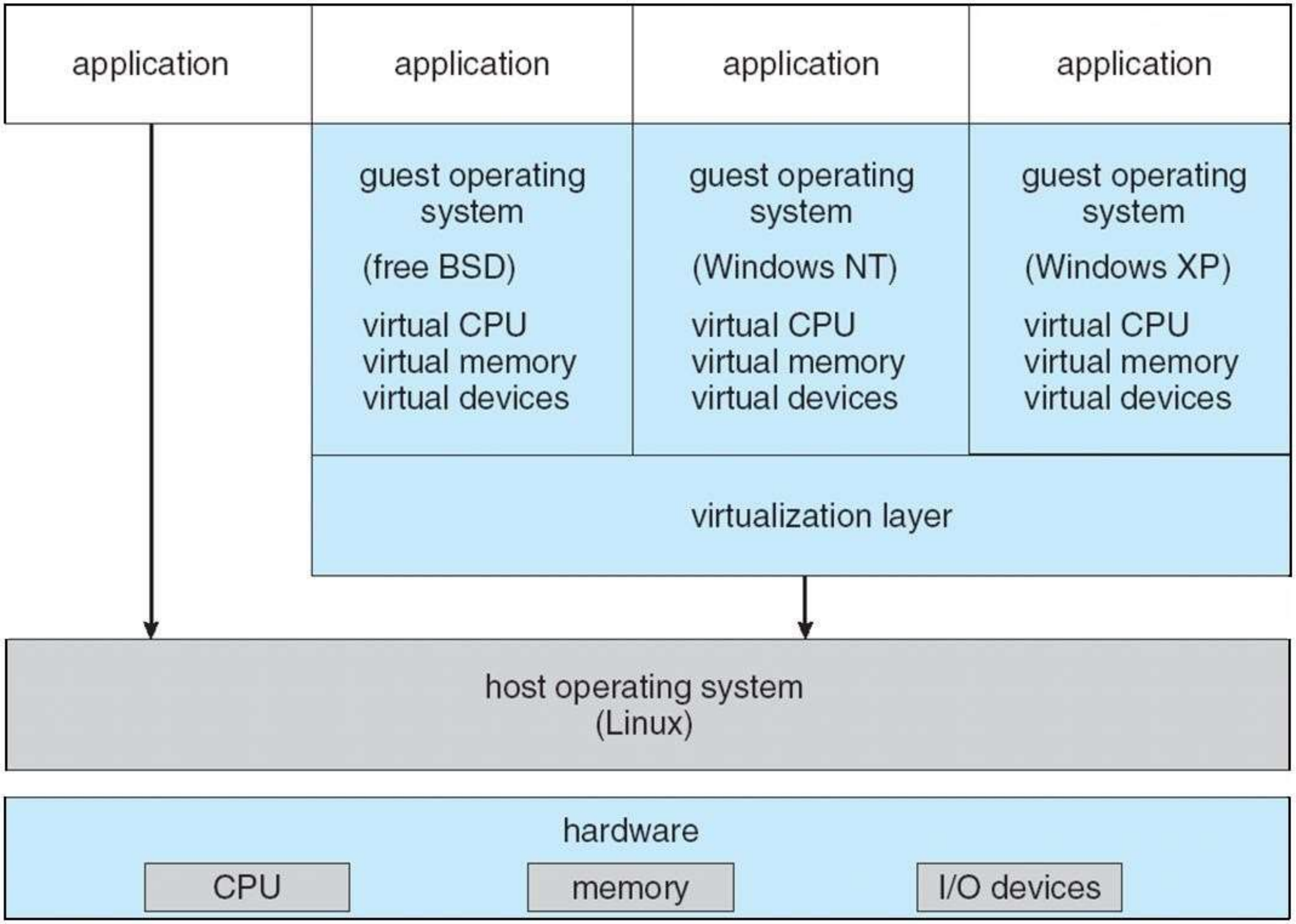
Virtual Machines

- Main idea
  - To abstract the hardware of a single computer (the CPU, memory, disk drives, and so forth) into several different execution environments, thereby creating the illusion that each separate execution environment is running its own private computer.
- A virtual machine provides an interface *identical* to the underlying bare hardware.
- Benefit:
  - to share the same hardware yet run several different execution environments (that is, different operating systems) concurrently.

Computing Environments - Virtualization



VMWare


application
application
application
application
guest operating system
(free BSD)
virtual CPU
virtual memory
virtual devices
guest operating system
(Windows NT)
virtual CPU
virtual memory
virtual devices
guest operating system
(Windows XP)
virtual CPU
virtual memory
virtual devices
virtualization layer
host operating system
(Linux)
hardware
CPU
memory
I/O devices