Chapter 1: Introduction

What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer 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

Four Components of a Computer System

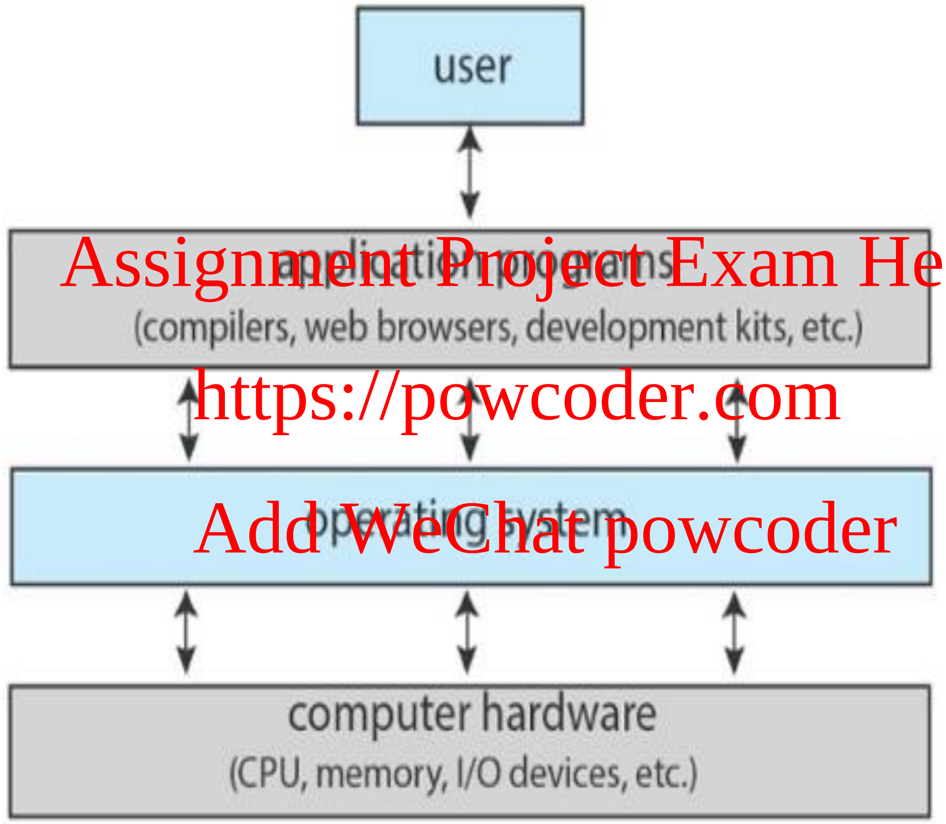


Figure 1.1 Abstract view of the components of a computer system.

What Operating Systems Do

- Depends on the point of view
- Users want convenience, **ease of use**
  - Don’t care about **resource utilization**
- But shared computer such as **mainframe** or **minicomputer** must keep all users happy
- Users of dedicate systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- Handheld computers are resource poor, optimized for usability and battery life
- Some computers have little or no user interface, such as embedded computers in devices and automobiles

Operating System Definition

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use

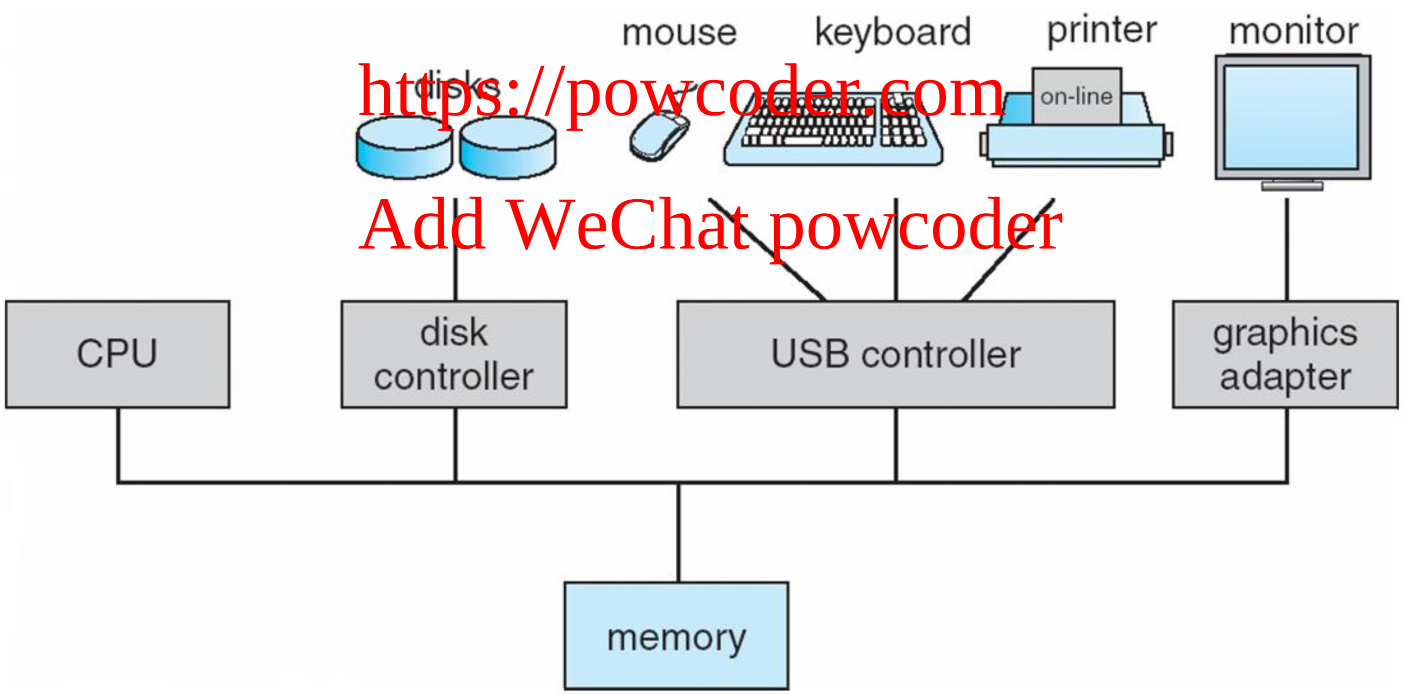
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

Operating System Definition (Cont.)

- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the **kernel**. Everything else is either a system program (ships with the operating system) or an application program.

Computer System Organization

- Computer-system organization
  - 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
- The device driver for each device moves data from/to main memory to/from local buffers
- The device controller is responsible for moving the data between the peripheral devices that it controls and its local buffer storage.
- Device controller informs CPU that it has finished its operation by causing an interrupt

Common Functions of Interrupts

- Hardware may trigger an **interrupt** at any time by sending a signal to the CPU, usually by way of the system bus.
- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request
- Software error or request creates **exception** or **trap**
  - Division by zero, request for operating system service
- Other process problems include infinite loop, processes modifying each other or the operating system
- An operating system is **interrupt driven**

Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter
- Separate segments of code determine what action should be taken for each type of interrupt
- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction

Interrupt Related Concepts

- **Interrupt number:** identifies the type of interrupt - provided by the interrupt h/w architecture whenever an interrupt occurs – is used as an index into the **interrupt vector** to lookup the address of the service routine for each interrupt.
- **Program counter**: contains address of instruction to be executed next by the CPU – a h/w CPU register – the interrupt h/w architecture automatically first saves PC value whenever an interrupt occurs.

- **Process**: a program in execution.

- **Process Control Block (PCB):** stores important system information about each process.

- **System call:** special instruction to invoke the OS, by generating an interrupt, to perform an OS-related service – a number i is associated with each type of system call, and is used as an index into a system call table to look up the address of the program which implements each type of system call.
- **CPU Scheduler:** selects a process from the processes in memory that are ready to execute and allocates the CPU to that process. (OS-ch-3.9-3.10)

Interrupt Timeline

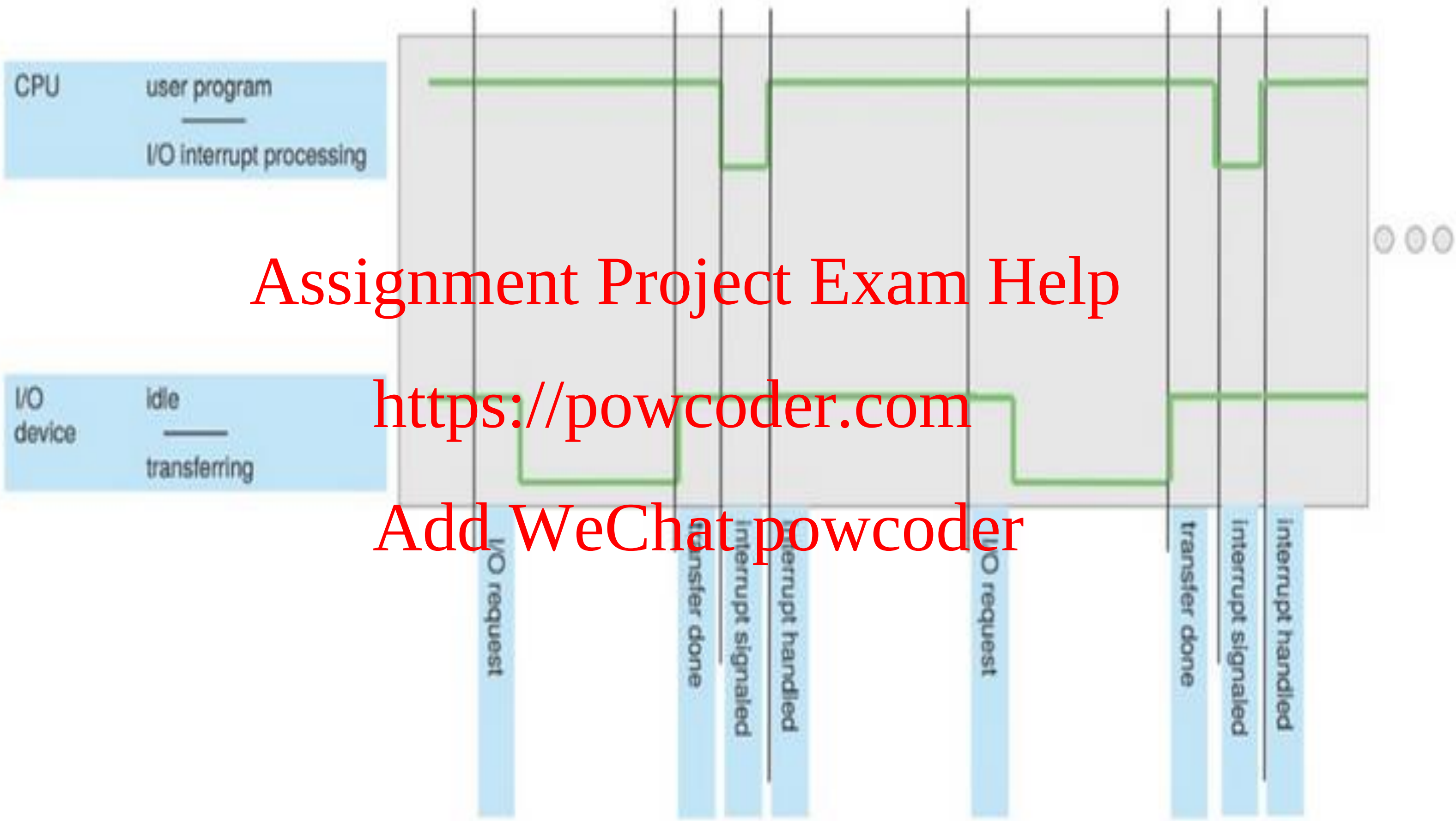


Figure 1.3 Interrupt timeline for a single program doing output.

Interrupt-Driven I/O Cycle

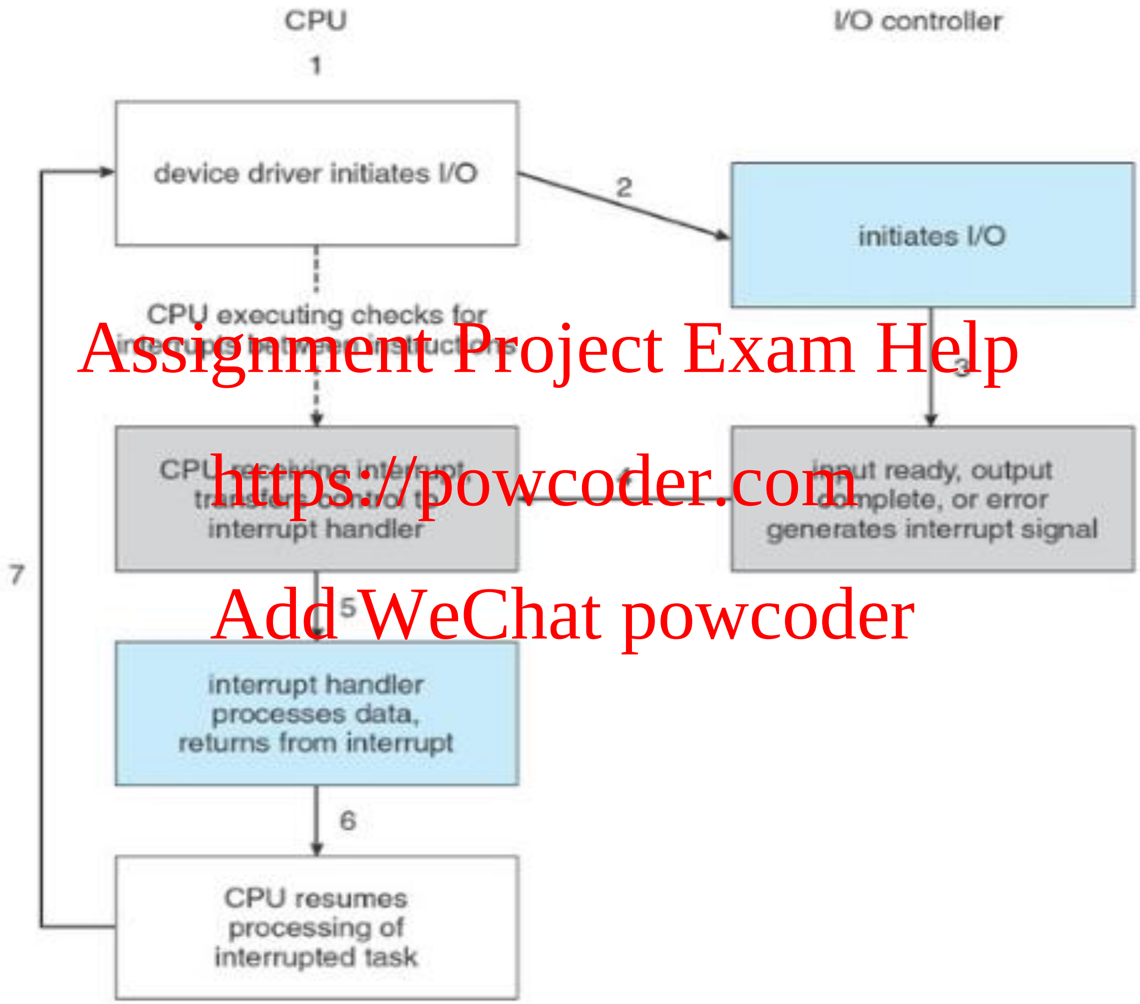


Figure 1.4 Interrupt-driven I/O cycle.

Interrupt-Vector Table

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Figure 1.5 Intel processor event-vector table.

Multiprogramming System

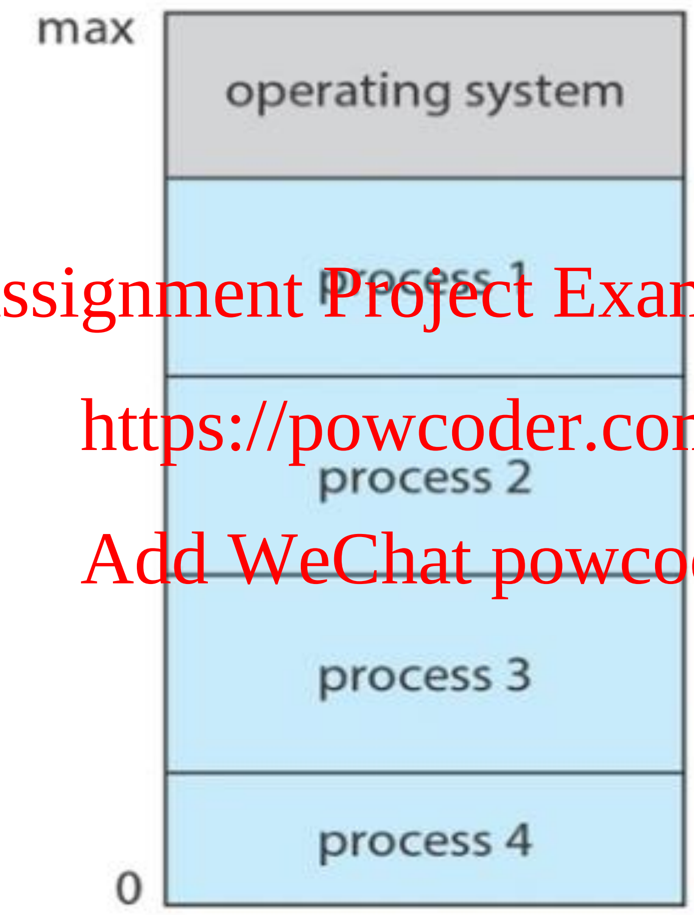


Figure 1.12 Memory layout for a multiprogramming system.

Process State

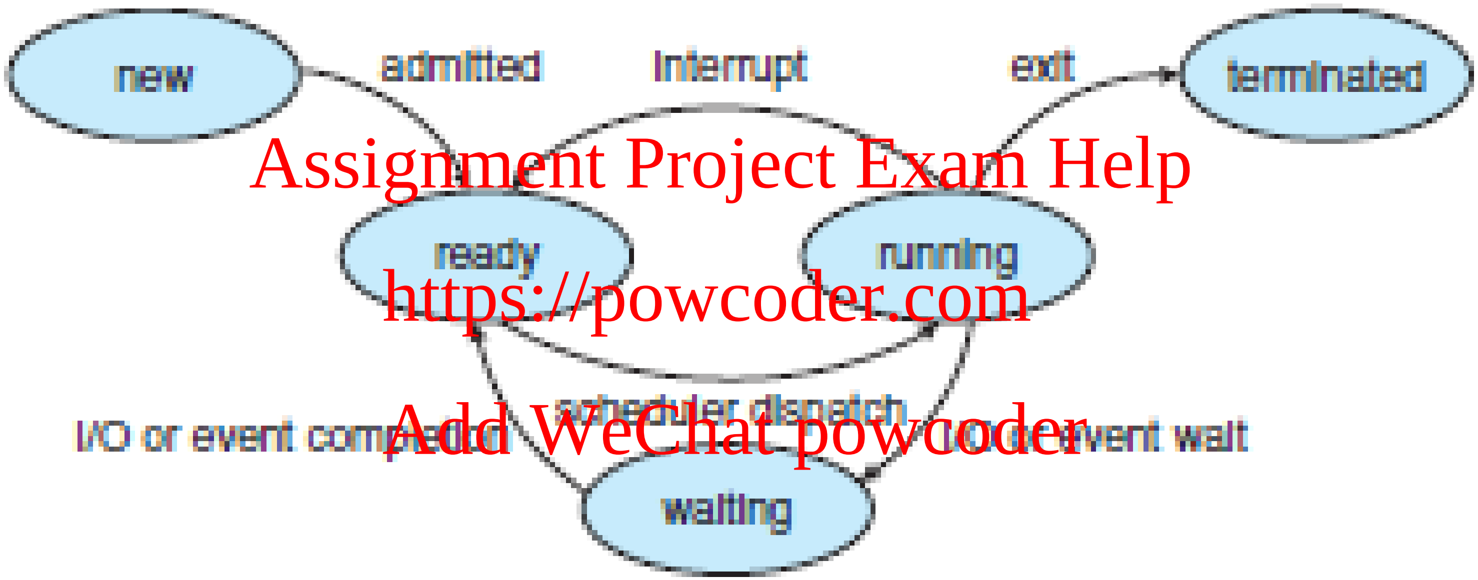


Figure 3.2 Diagram of process state.

Process Control Block (PCB)

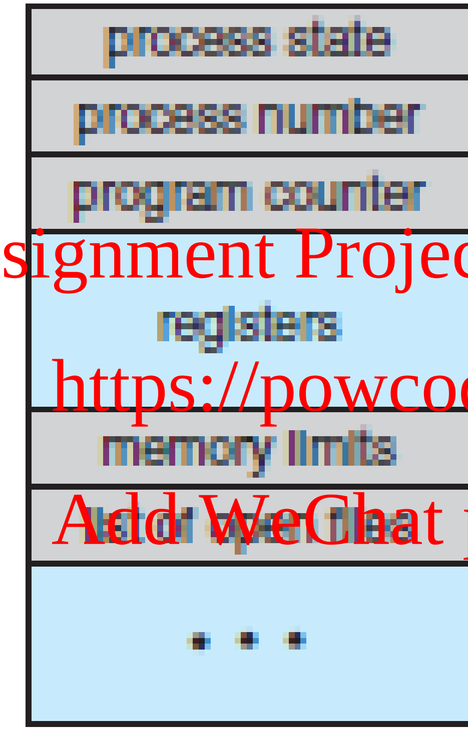


Figure 3.3 Process control block (PCB).

Ready Queues and Wait Queues

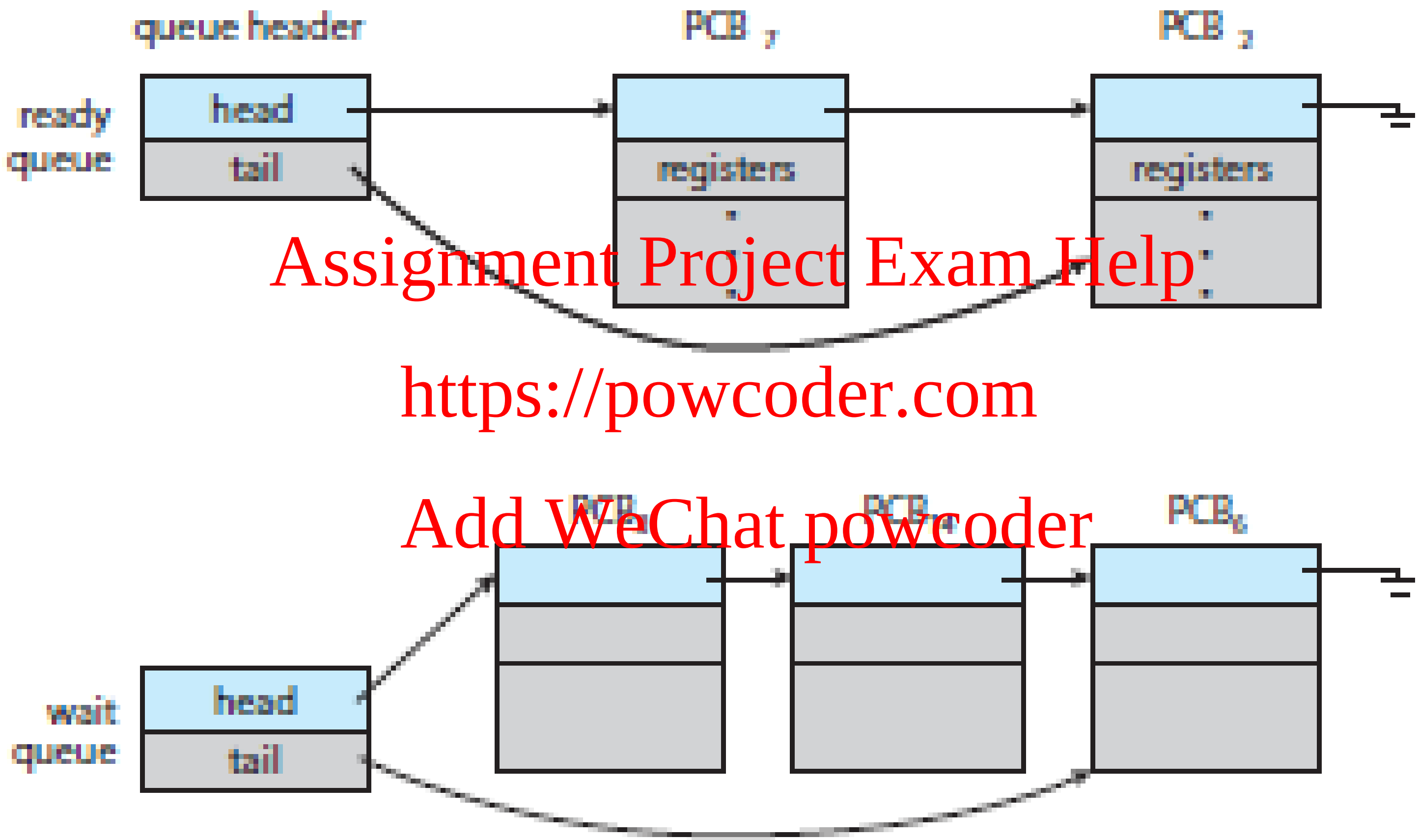


Figure 3.4 The ready queue and wait queues.

Process Scheduling Queueing Diagram

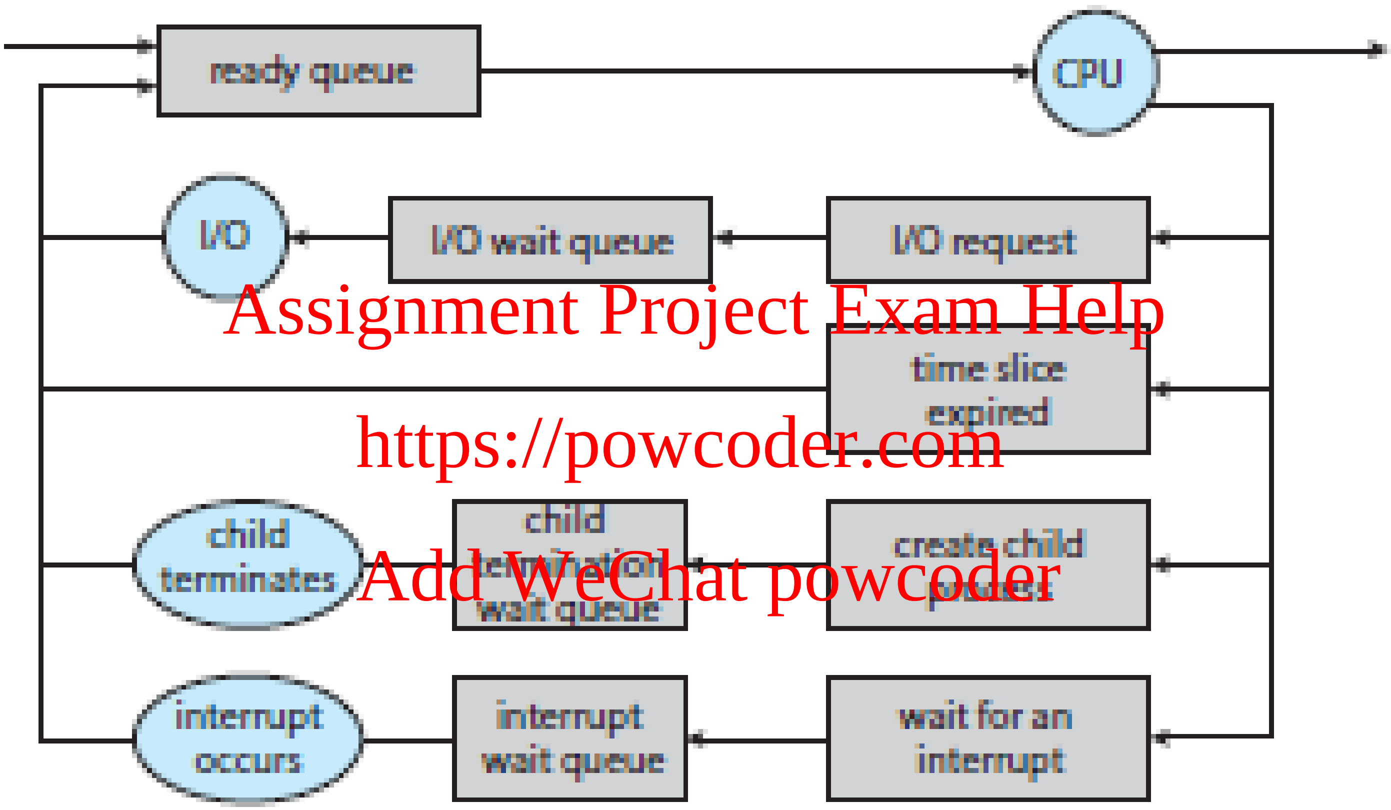


Figure 3.5 Queueing-diagram representation of process scheduling.

Context Switch

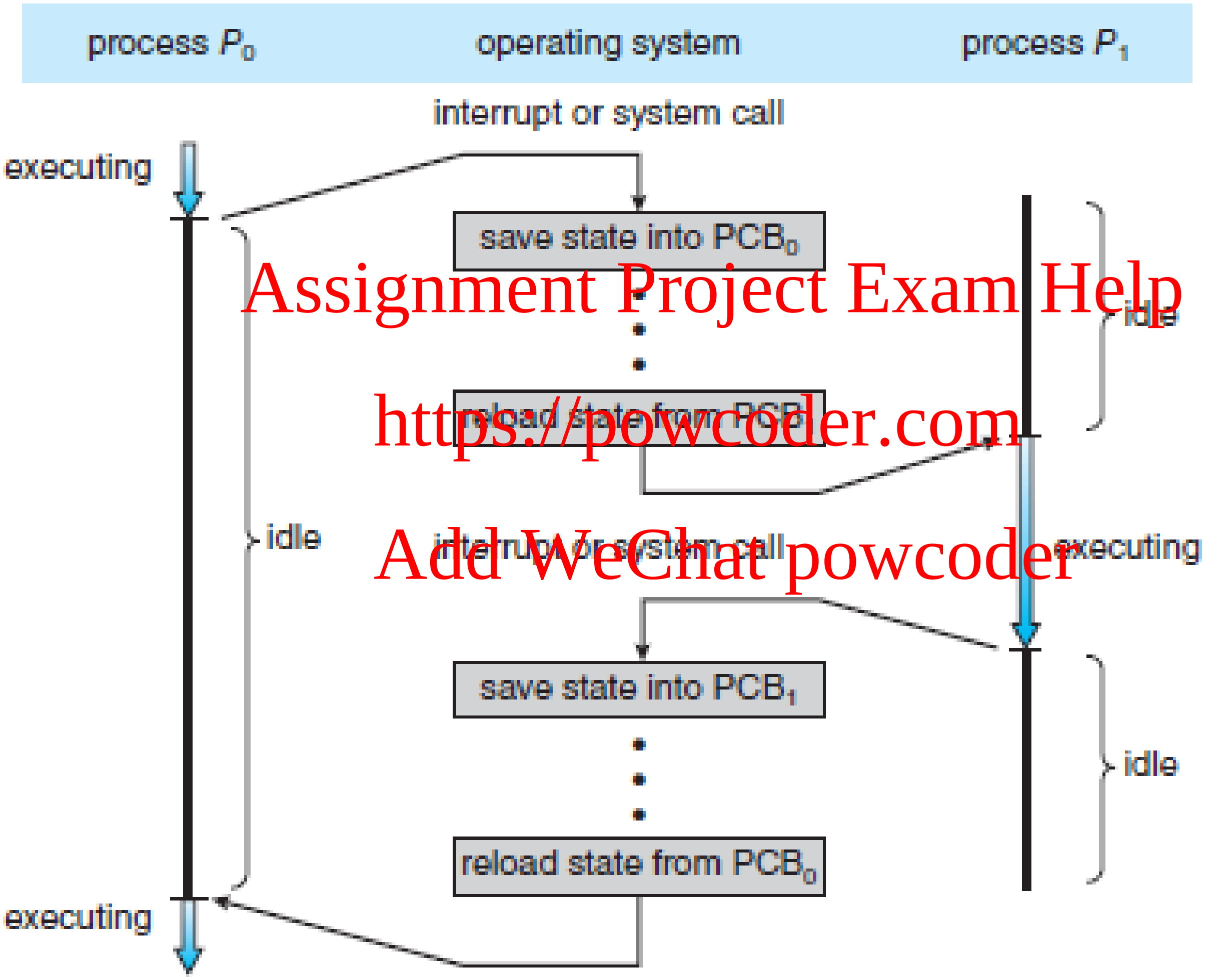


Figure 3.6 Diagram showing context switch from process to process.

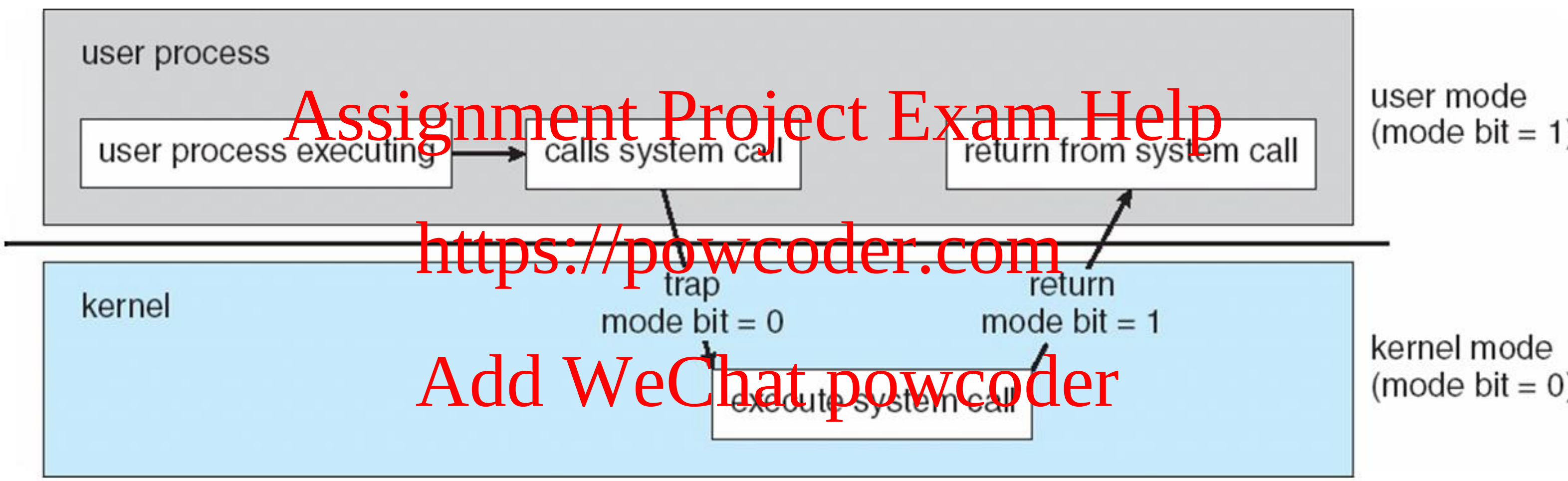
Computer Startup

- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

Dual Mode Operation

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user
- Increasingly CPUs support multi-mode operations
  - i.e. **virtual machine manager** (**VMM**) mode for guest **VMs**

Transition from User to Kernel Mode



API – System Call – OS Relationship

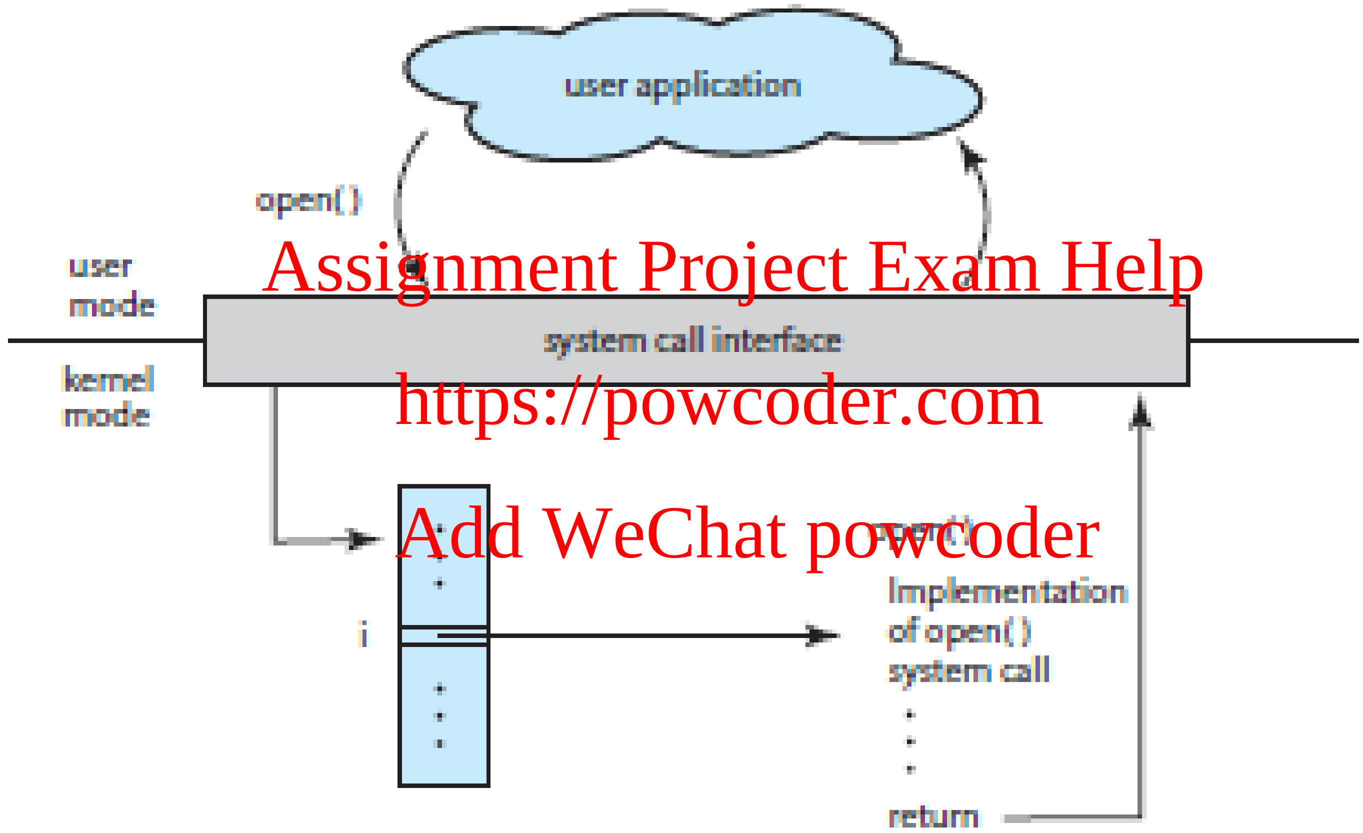
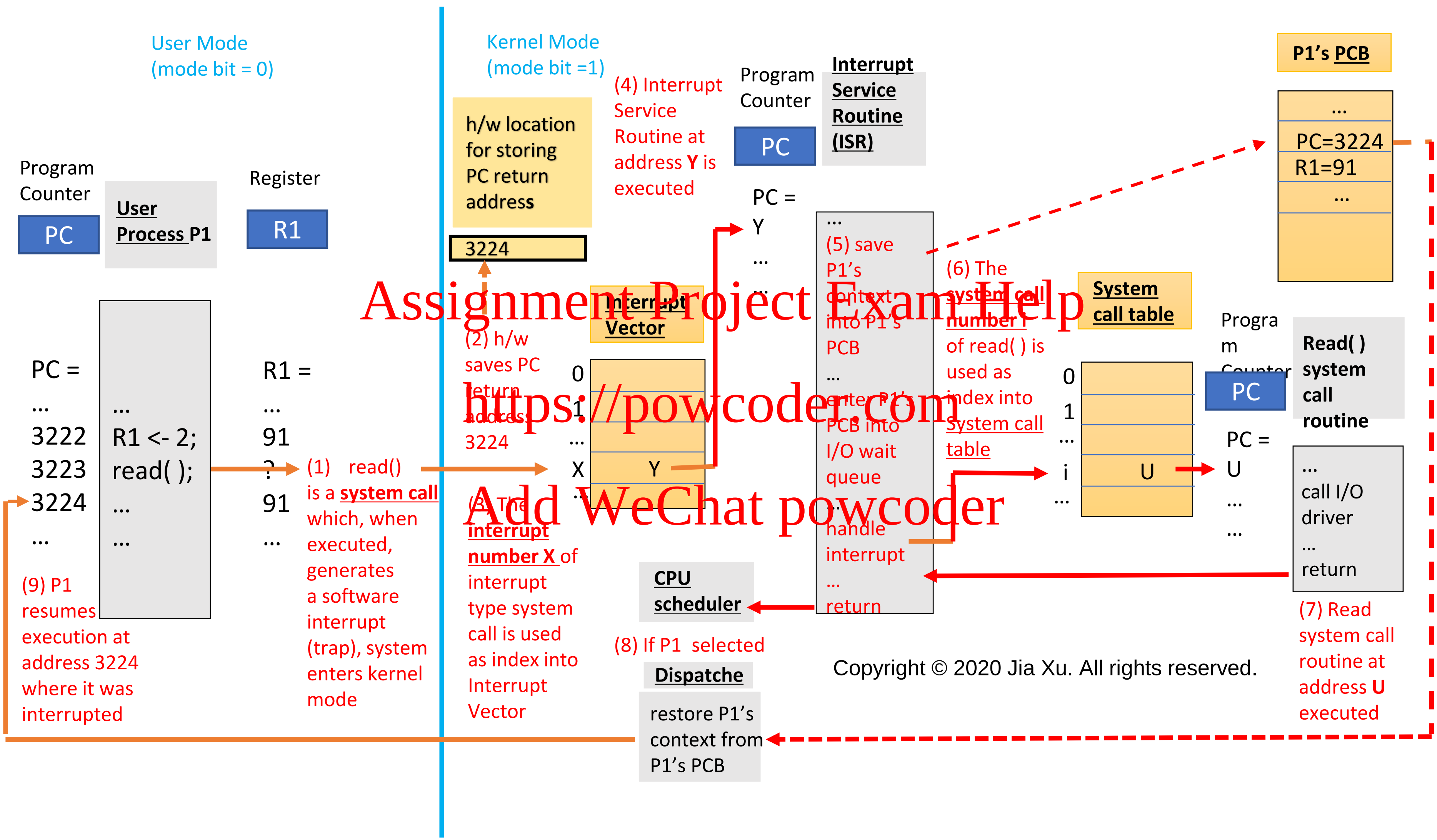


Figure 2.6 The handling of a user application invoking the open() system call.

Timer to Prevent Infinite Loop or Process Hogging Resources

- Timer to prevent infinite loop / process hogging resources
  - Set interrupt after specific period
  - Operating system decrements counter
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time

Interrupt Handling Diagram



Direct Memory Access Structure

- Used for high-speed I/O devices able to transmit information at close to memory speeds
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only one interrupt is generated per block, rather than the one interrupt per byte

Storage Definitions and Notation Review

STORAGE DEFINITIONS AND NOTATION

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

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

Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
- Magnetic disks – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
  - The **disk controller** determines the logical interaction between the device and the computer
- **Solid-state disks** – faster than magnetic disks, nonvolatile
  - Various technologies
  - Becoming more popular

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

- **Device Driver** for each device controller to manage I/O
  - Provides uniform interface between controller and kernel

Storage-Device Hierarchy

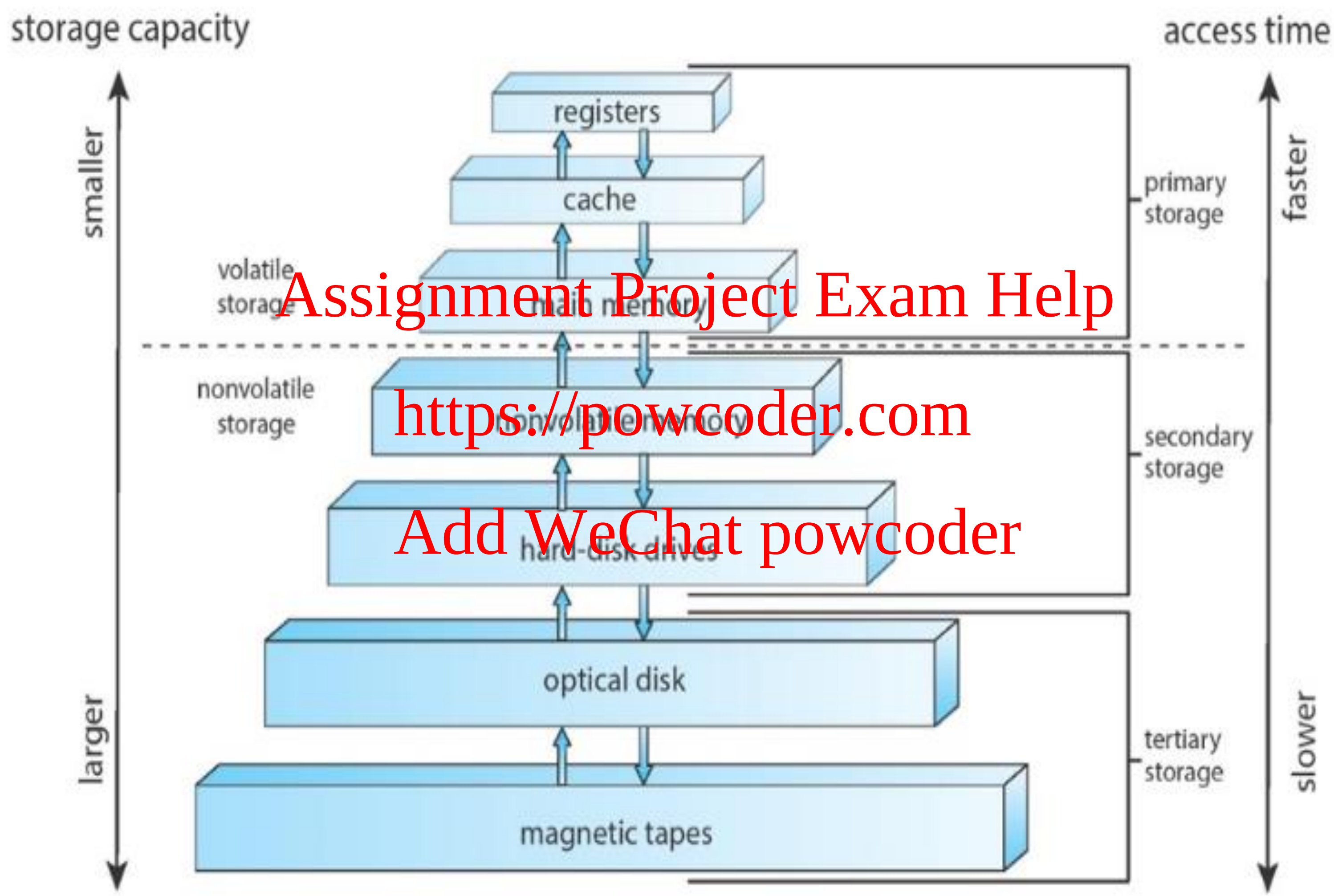


Figure 1.6 Storage-device hierarchy.

Performance of Various Levels of Storage

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

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

How A Modern Computer System Works

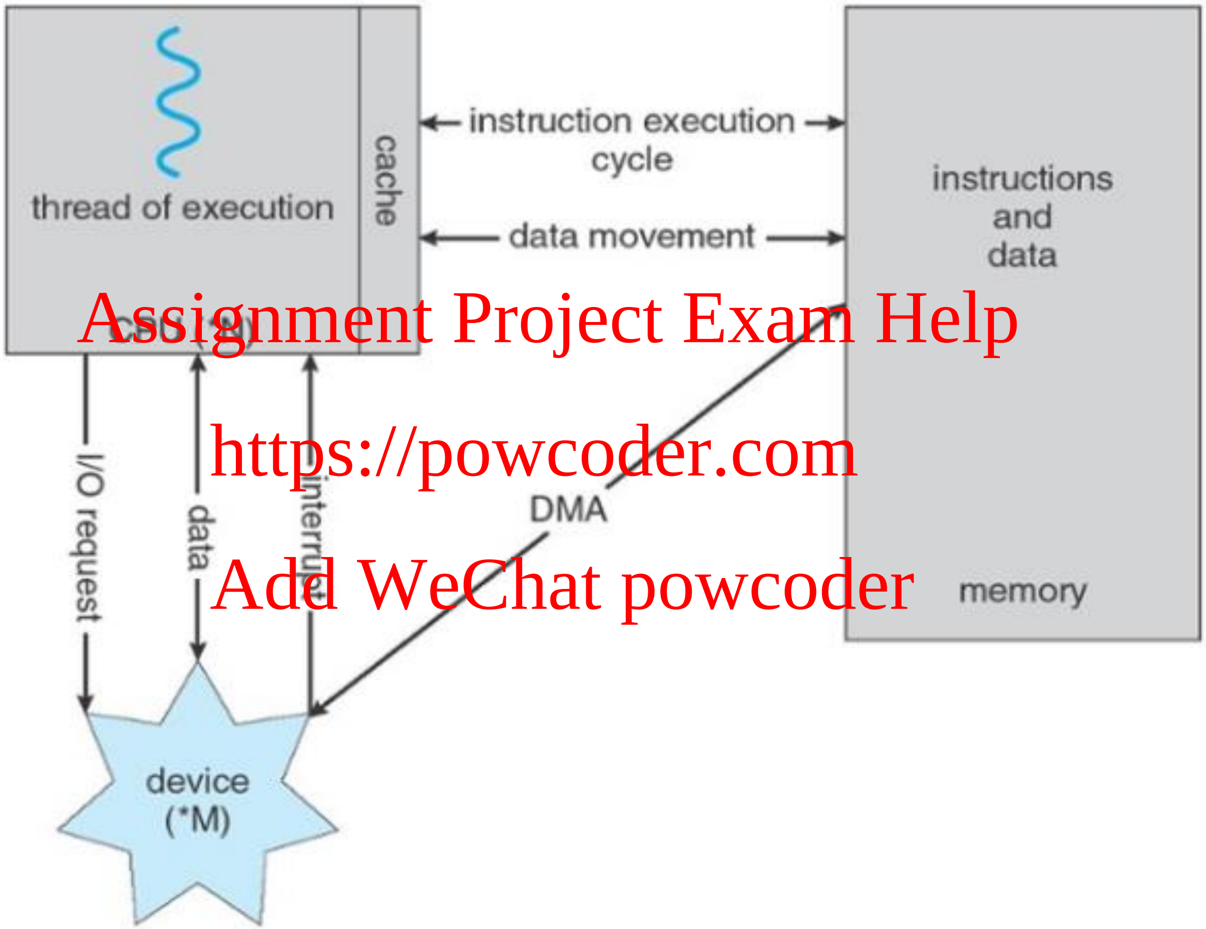


Figure 1.7 How a modern computer system works.

Computer-System Architecture

- Most systems use a single general-purpose processor (PDAs through mainframes)
  - Most systems have special-purpose processors as well
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Advantages include:
    1. **Increased throughput**
    2. **Economy of scale**
    3. **Increased reliability – graceful degradation** or **fault tolerance**
  - Two types:
    1. **Asymmetric Multiprocessing**
    2. **Symmetric Multiprocessing**

Symmetric Multiprocessing Architecture

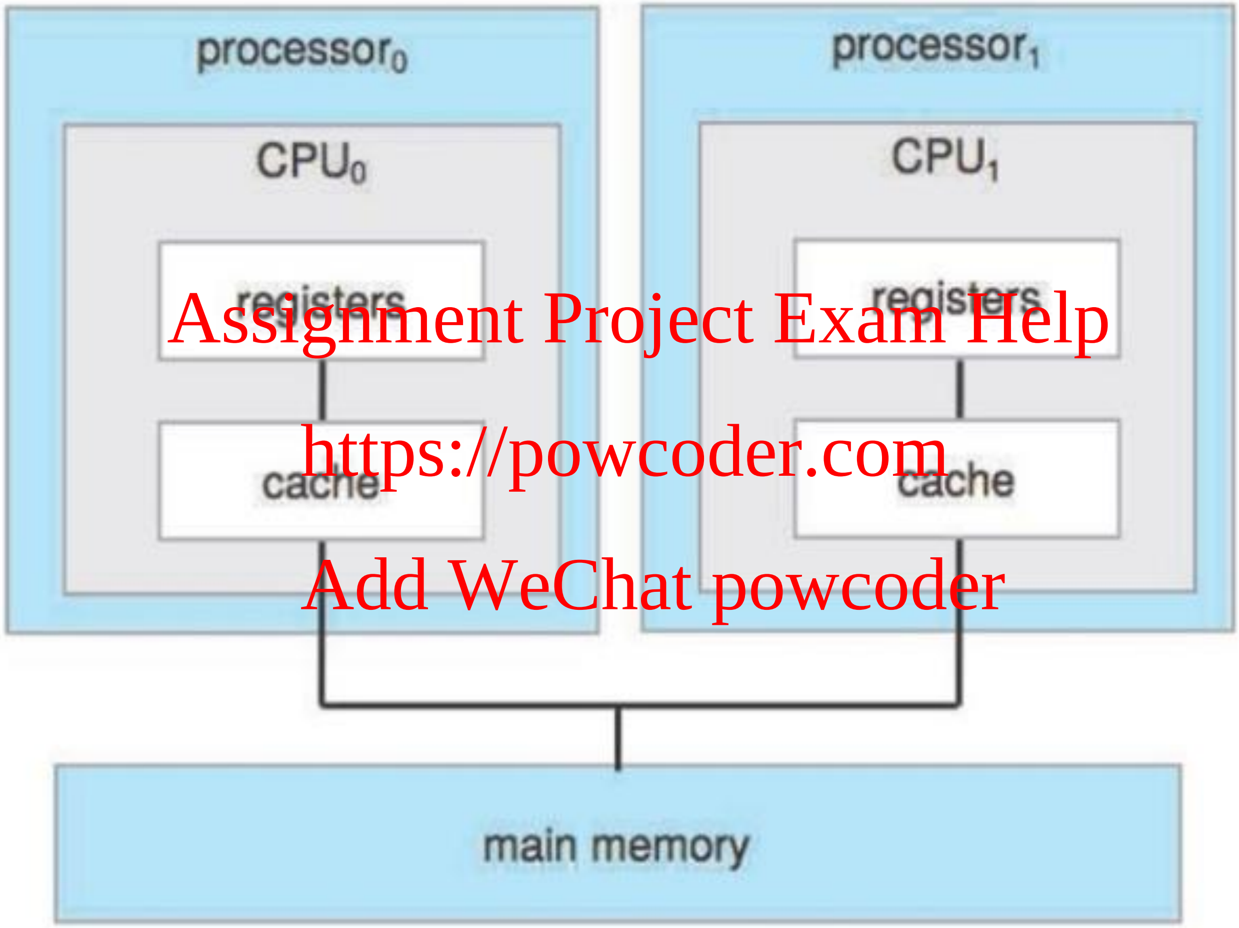


Figure 1.8 Symmetric multiprocessing architecture.

A Dual-Core Design

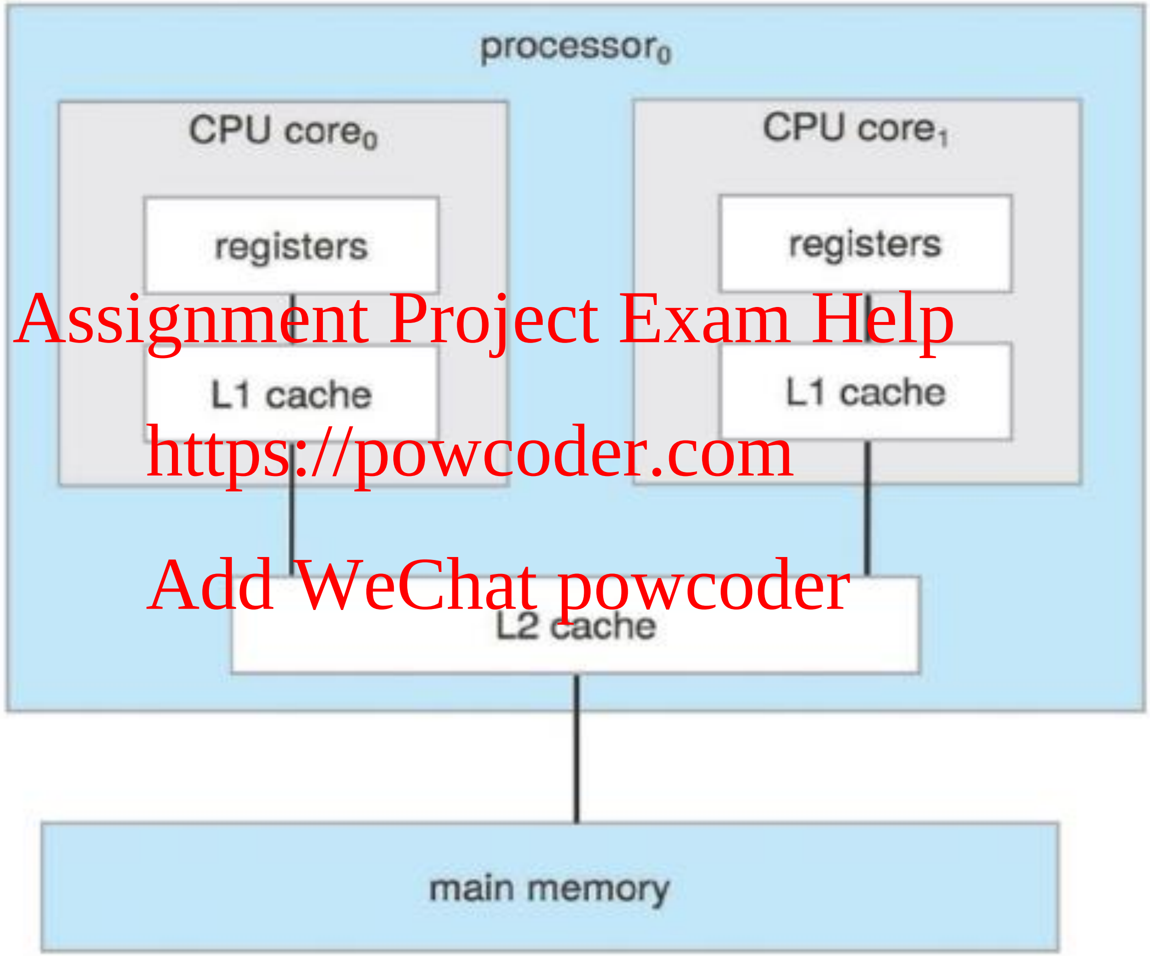


Figure 1.9 A dual-core design with two cores on the same chip.

Computer System Components

DEFINITIONS OF COMPUTER SYSTEM COMPONENTS

- CPU—The hardware that executes instructions.
- Processor—A physical chip that contains one or more CPUs.
- Core—The basic computation unit of the CPU.
- Multicore—Including multiple computing cores on the same CPU.
- Multiprocessor—Including multiple processors.

Although virtually all systems are now multicore, we use the general term *CPU* when referring to a single computational unit of a computer system and *core* as well as *multicore* when specifically referring to one or more cores on a CPU.

Non-Uniform Memory Access

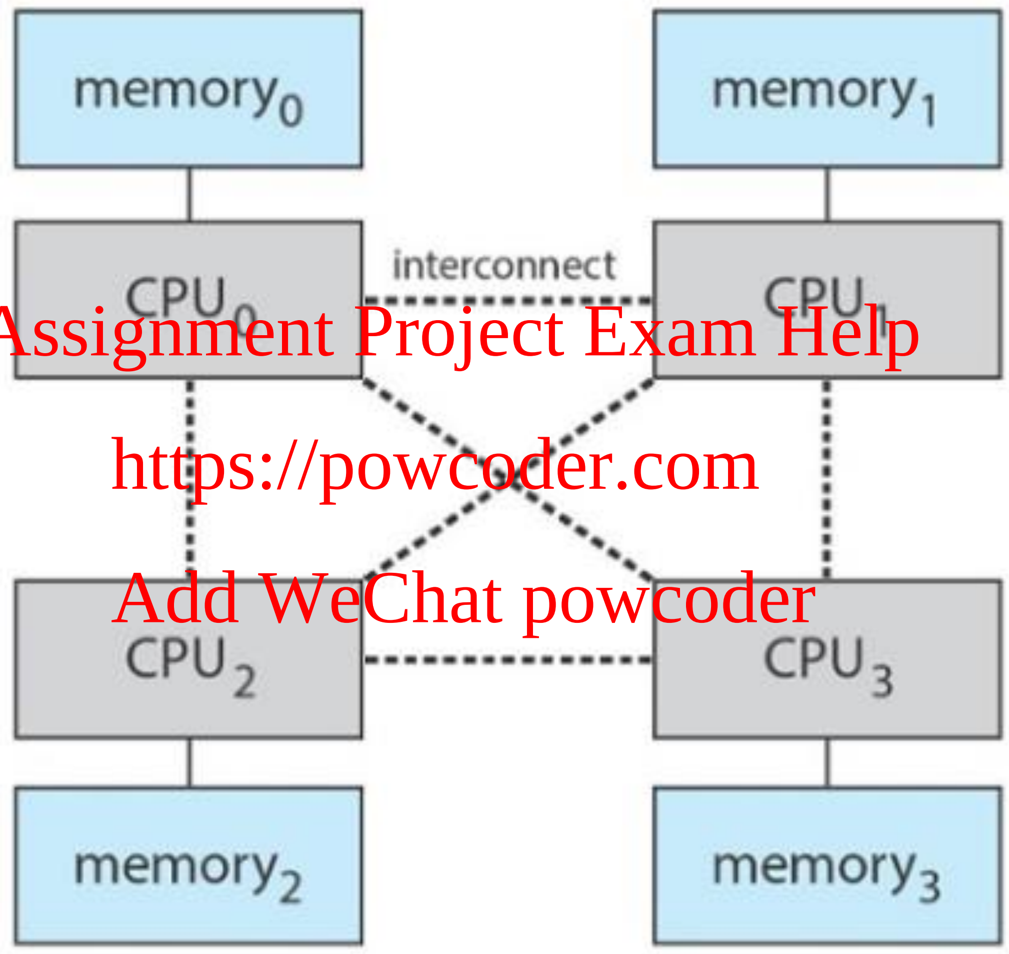
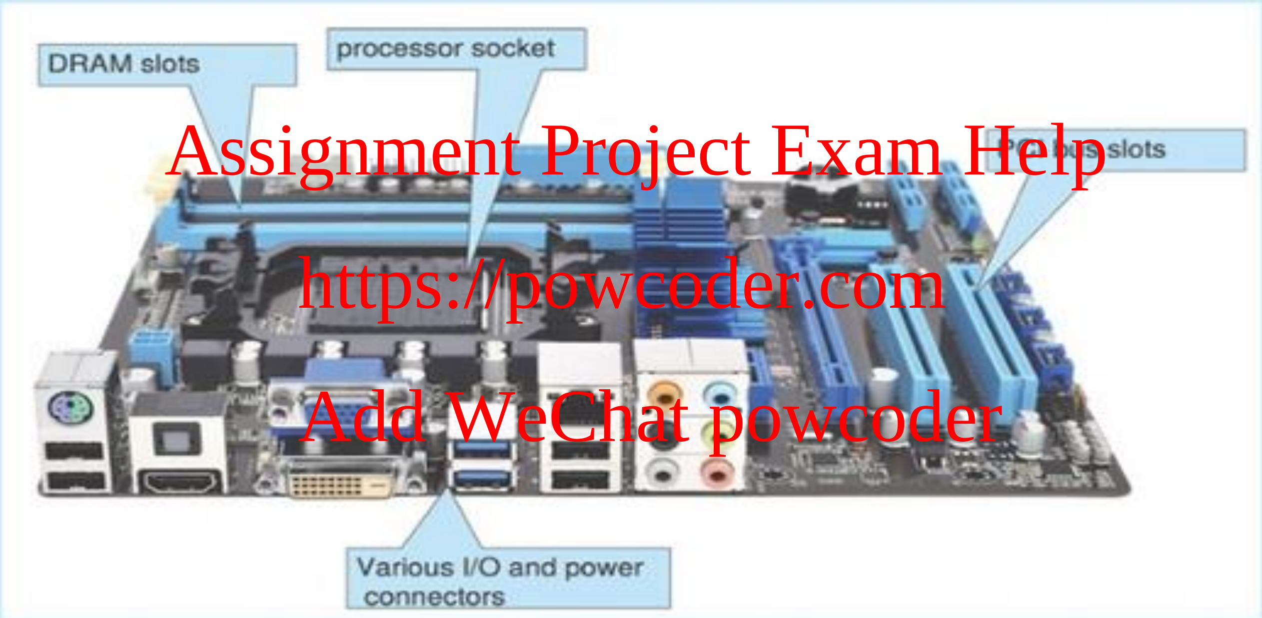


Figure 1.10 NUMA multiprocessing architecture.

PC Motherboard

PC MOTHERBOARD

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



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

Clustered Systems

- 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**

Clustered Systems

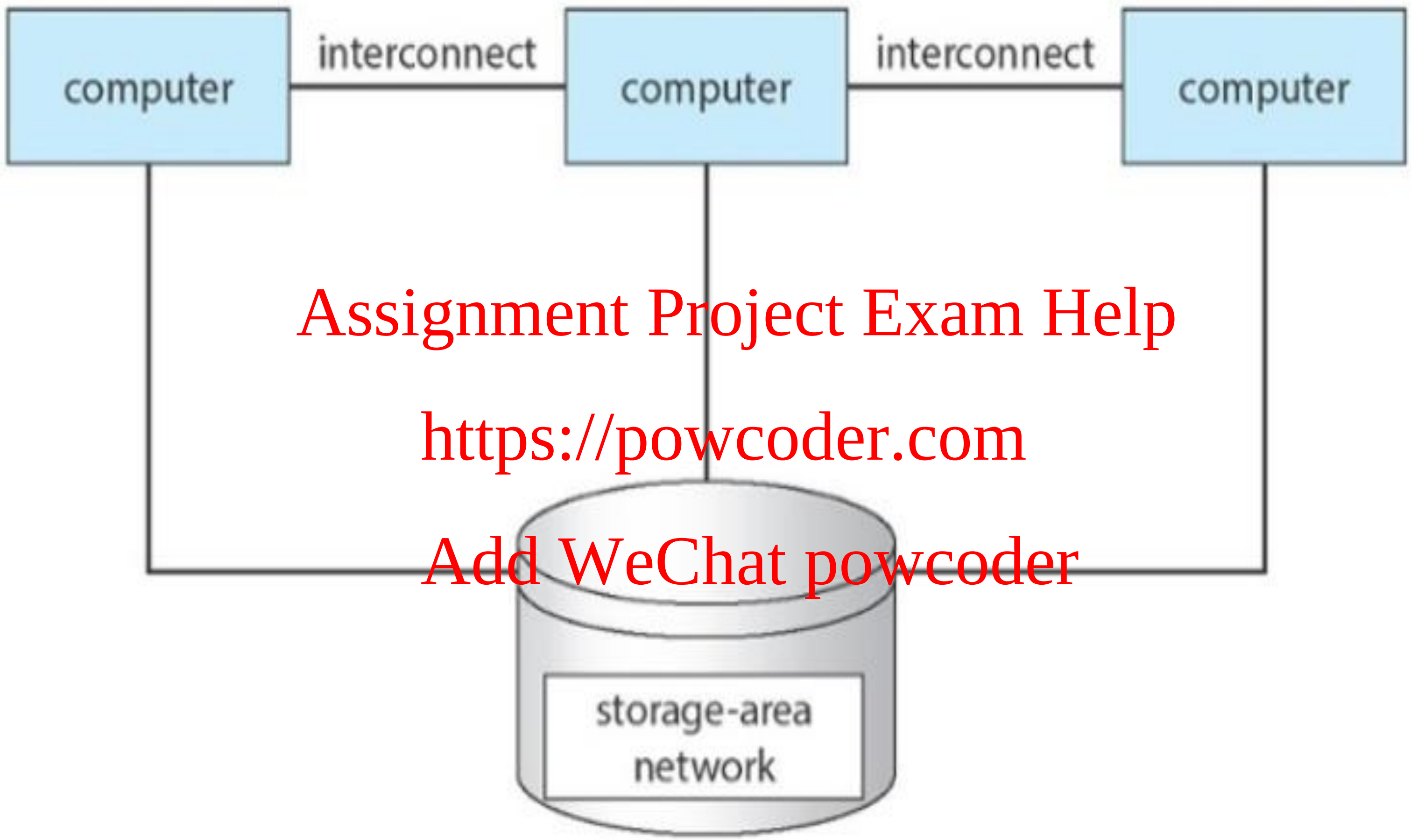


Figure 1.11 General structure of a clustered system.

Computing Environments

- Many different kinds of computing environments
  - Traditional computing
  - Mobile computing
  - Client Server computing
  - Peer-to-Peer computing
  - Cloud computing
  - Virtualization
  - Real-Time Embedded Systems
  - Open Source Operating Systems

Client Server Computing

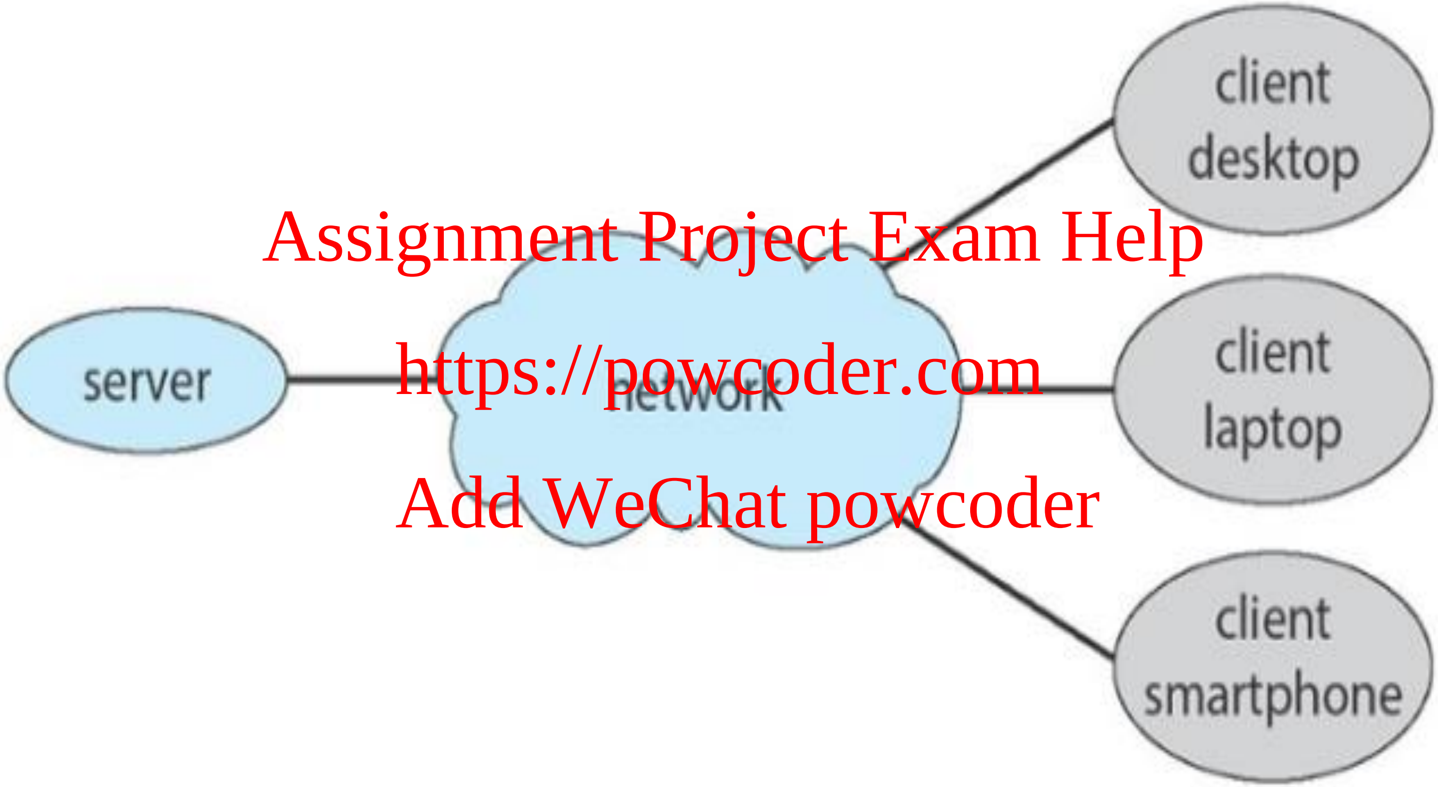


Figure 1.22 General structure of a client–server system.

Peer-to-Peer Computing

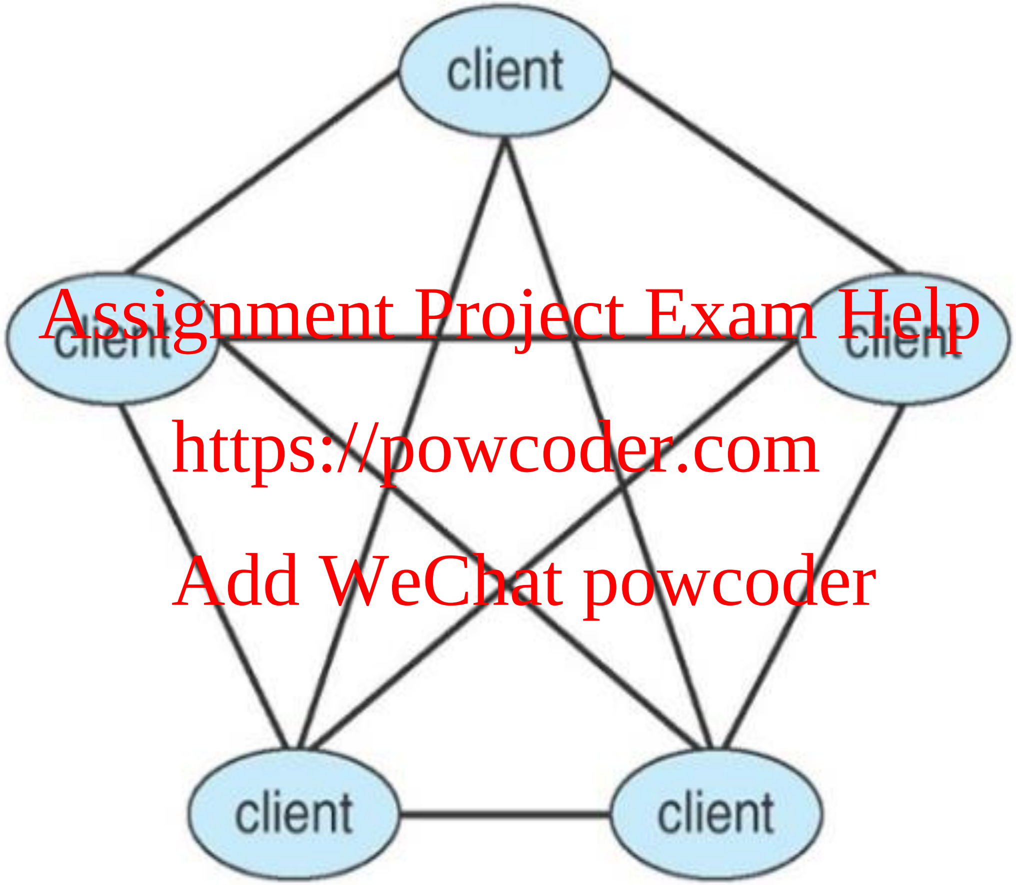


Figure 1.23 Peer-to-peer system with no centralized service.

Computing Environments – Cloud Computing

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization as based on virtualization
  - Amazon **EC2** has thousands of servers, millions of VMs, PBs of storage available across the Internet, pay based on usage
- Many types
  - **Public cloud** – available via Internet to anyone willing to pay
  - **Private cloud** – run by a company for the company's own use
  - **Hybrid cloud** – includes both public and private cloud components
  - Software as a Service (**SaaS**) – one or more applications available via the Internet (i.e. word processor)
  - Platform as a Service (**PaaS**) – software stack ready for application use via the Internet

    Ex: LAMP (Linux (OS), Apache (web server), MySQL (DB), PHP, Perl or Python (programming languages)
  - Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e. storage available for backup use)

Cloud Computing

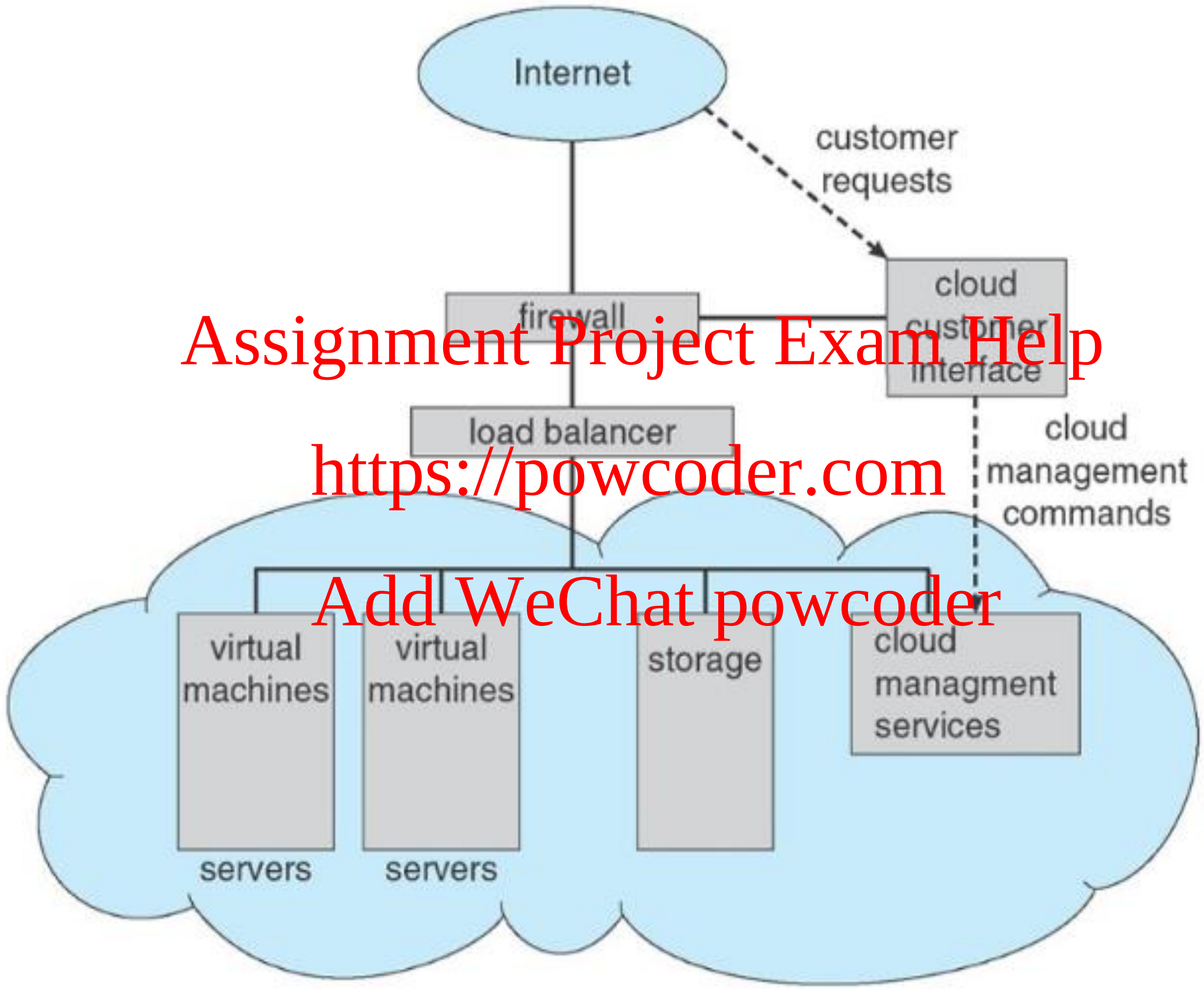


Figure 1.24 Cloud computing.

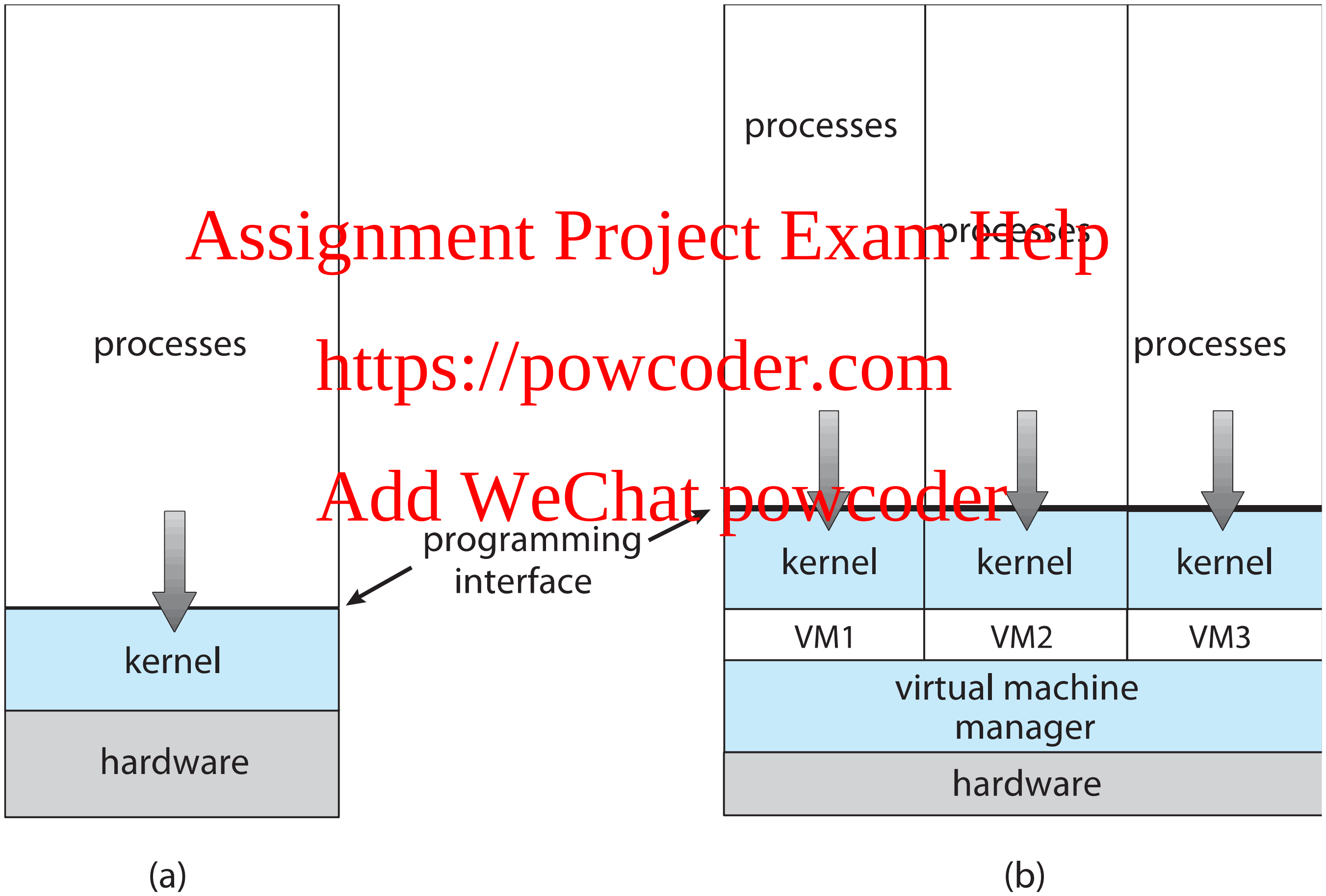
Computing Environments - Virtualization

- Allows operating systems to run applications within other OSes
  - Vast and growing industry
- **Virtualization** – OS natively compiled for CPU, running **guest** OSes also natively compiled
  - Consider VMware running WinXP guests, each running applications, all on native WinXP **host** OS
  - **VMM** provides virtualization services

Computing Environments - Virtualization

- Use cases involve laptops and desktops running multiple OSes for exploration or compatibility
  - Apple laptop running Mac OS X host, Windows as a guest
  - Developing apps for multiple OSes without having multiple systems
  - QA testing applications without having multiple systems
  - Executing and managing compute environments within data centers

Computing Environments - Virtualization



Computing Environments – Real-Time Embedded Systems

- Real-time embedded systems most prevalent form of computers
  - Vary considerable, special purpose, limited purpose OS, **real-time OS**
  - Use expanding
- Many other special computing environments as well
  - Some have OSes, some perform tasks without an OS
- Real-time OS has well-defined fixed time constraints
  - Processing ***must*** be done within constraint
  - Correct operation only if constraints met

Open-Source Operating Systems

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

- Counter to the **copy protection** and **Digital Rights Management (DRM)** movement

- Started by **Free Software Foundation (FSF)**, which has “copyleft” **GNU Public License (GPL)**

- Examples include **GNU/Linux** and **BSD UNIX** (including core of **Mac OS X**), and many more

End of Chapter 1