Chapter 3 Operating Systems

Chapter 3 Operating Systems

3.1 The History of Operating Systems

3.2 Operating System Architecture

3.3 Coordinating the Machine’s Activities

3.4 Handling Competition Among Processes

3.5 Security

Computer System Structure

Hardware – provides basic computing resources

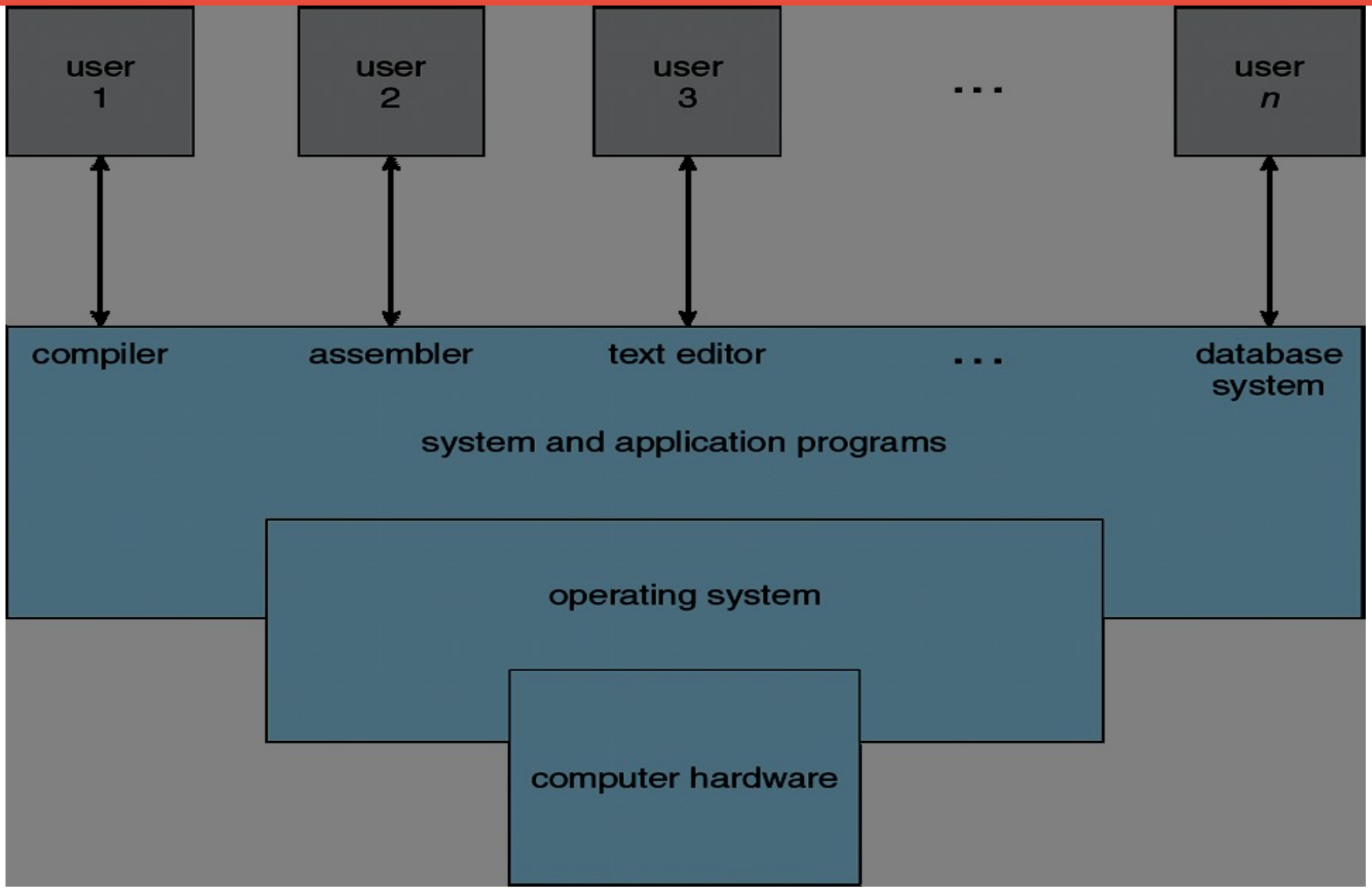
CPU, memory, I/O devices

Operating system – Controls and coordinates use of hardware among various applications and users

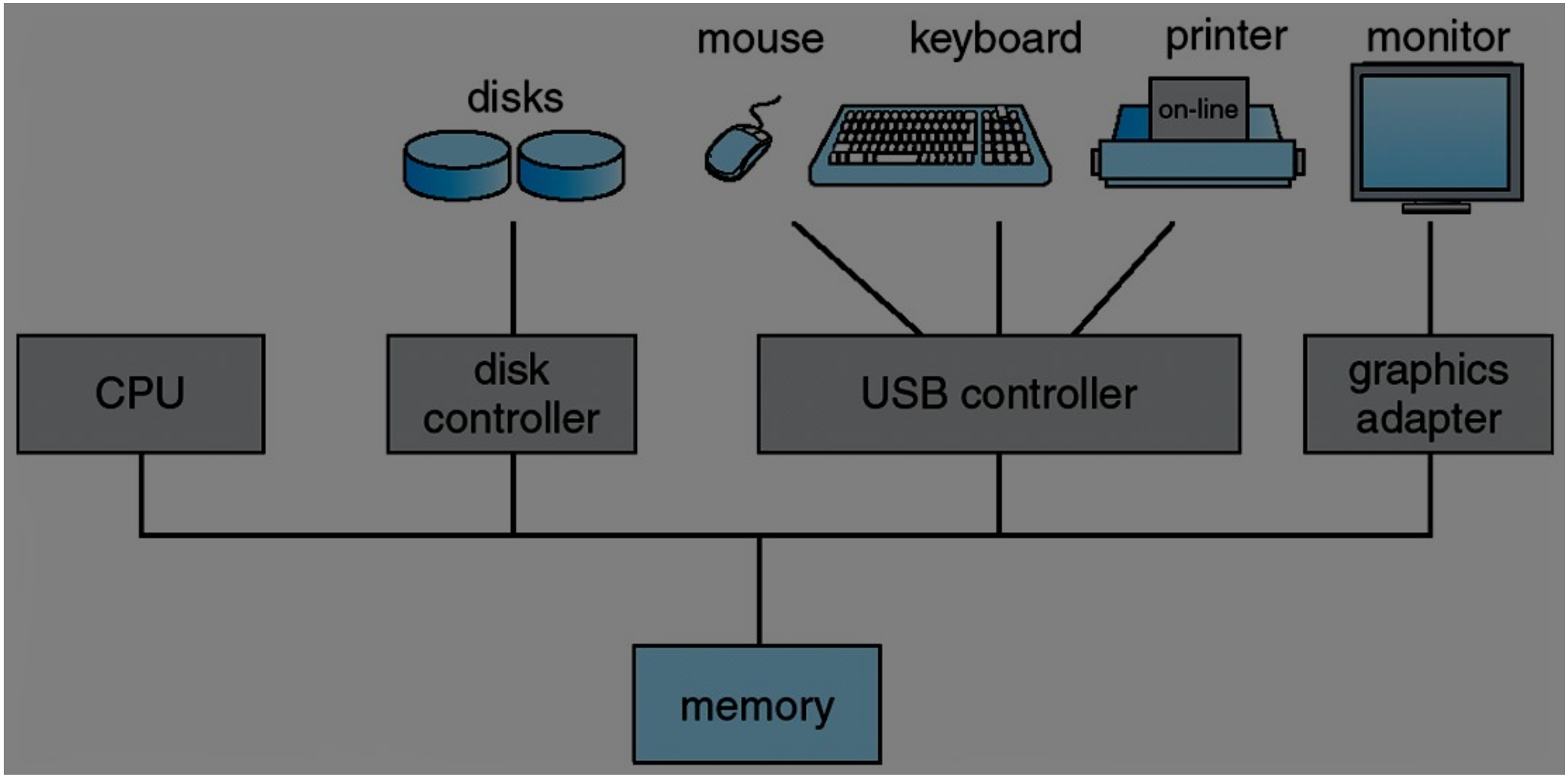
Application programs – define the ways in which the system resources are used to solve the computing problems of the users; Word processors, compilers, web browsers, database systems, video games

Users; People, machines, other computers

Computer System Structure



Computer System Organization



Top-Level Components

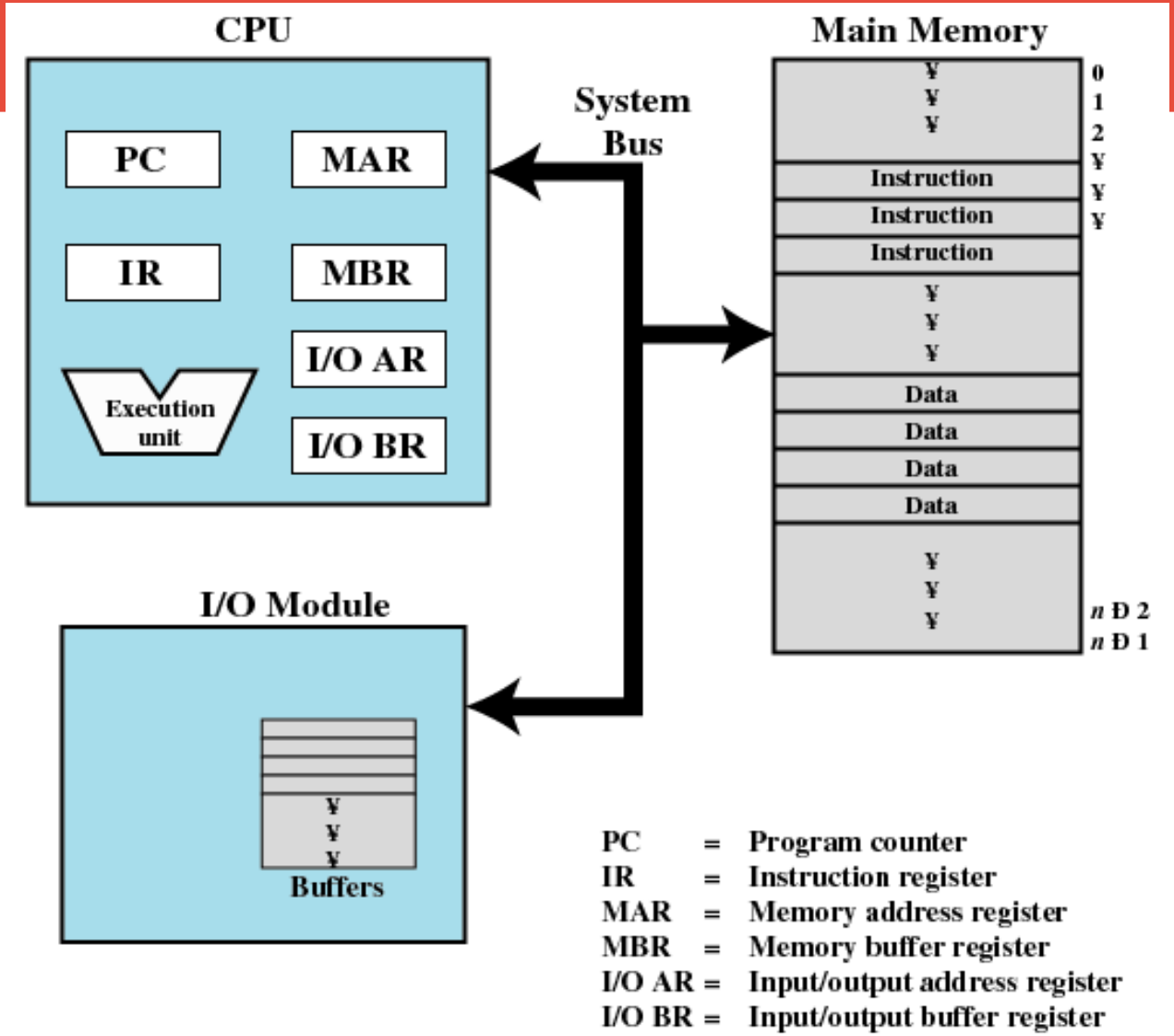


Figure 1.1 Computer Components: Top-Level View

Central Processing Unit (CPU)

Processor (CPU) to perform computations:

Executes a sequence of stored instructions called a program, program is kept in the main memory

Control unit : responsible for deciding which instruction in a program should be executed

Arithmetic and logic unit (ALU) : responsible for executing the actual instructions

Register : quickly accessible location available to CPU

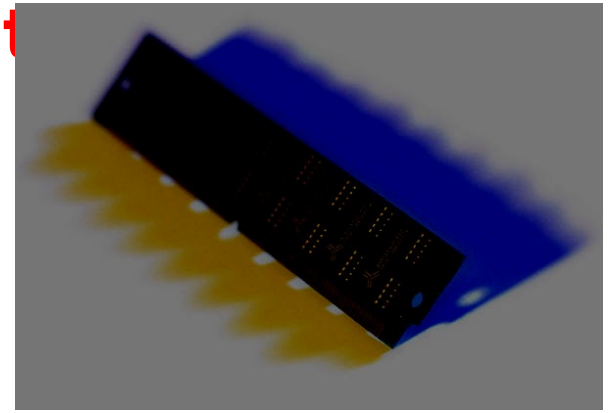
Main Memory

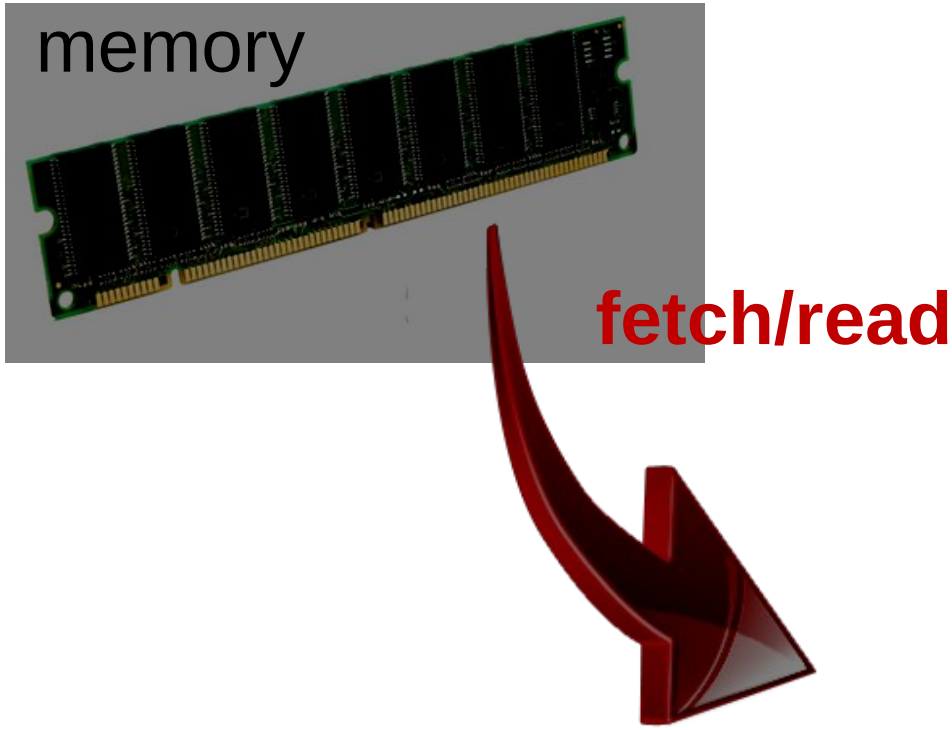
Collection of locations, each of which is capable of storing both instructions and data

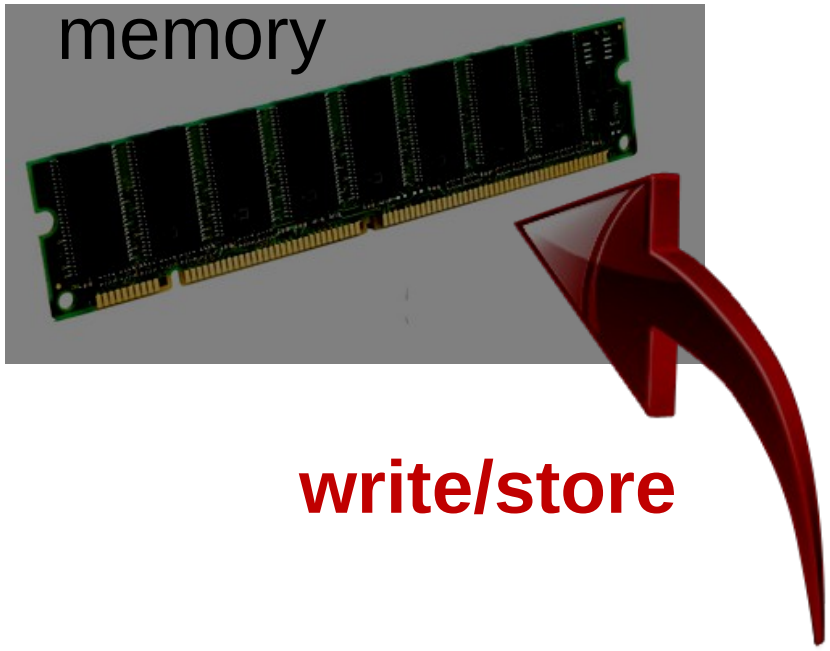
Every location consists of an address, which is used to access the location, and the contents of the location

OS sees and manages memory

Programs/data need to be brought t




memory
fetch/read
CPU


memory
write/store
CPU

Instruction Cycle

The processor fetches the instruction from memory

Program counter (PC) holds address of the instruction to be fetched next

Program counter is incremented after each fetch

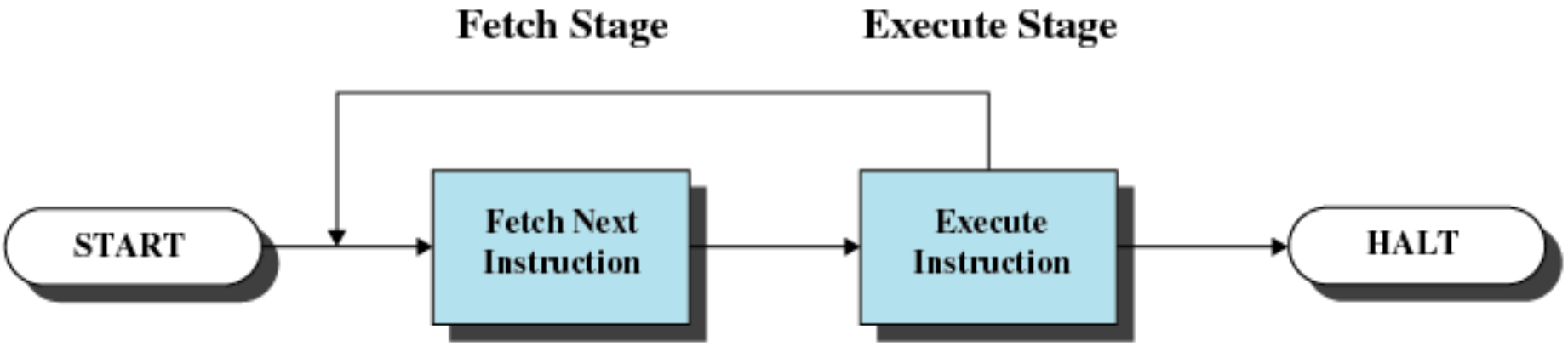
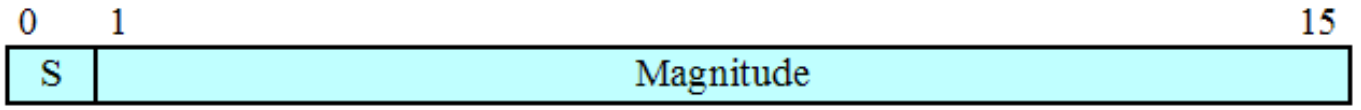


Figure 1.2 Basic Instruction Cycle

Characteristics of a Hypothetical Machine

(a) Instruction format



(b) Integer format

Program Counter (PC) = Address of instruction
Instruction Register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

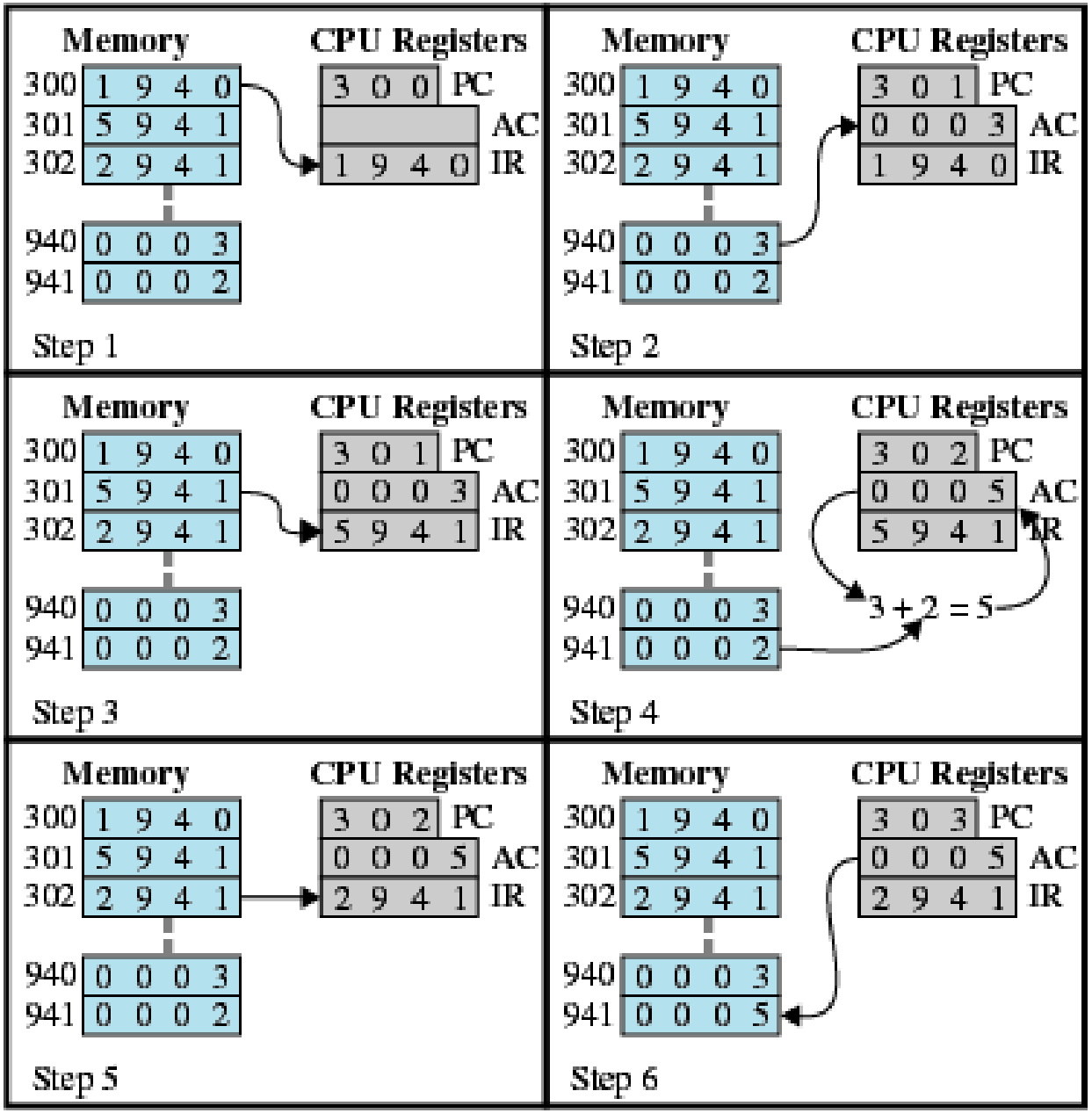
0001 = Load AC from Memory
0010 = Store AC to Memory
0101 = Add to AC from Memory

(d) Partial list of opcodes

Figure 1.3 Characteristics of a Hypothetical Machine

Example of Program Execution

0001 – Load AC from Memory
0010 – Store AC to Memory
0101 – Add to AC from Memory



I/O Devices

Large variety, varying speeds: Disk, monitor, printer

Each has a controller:

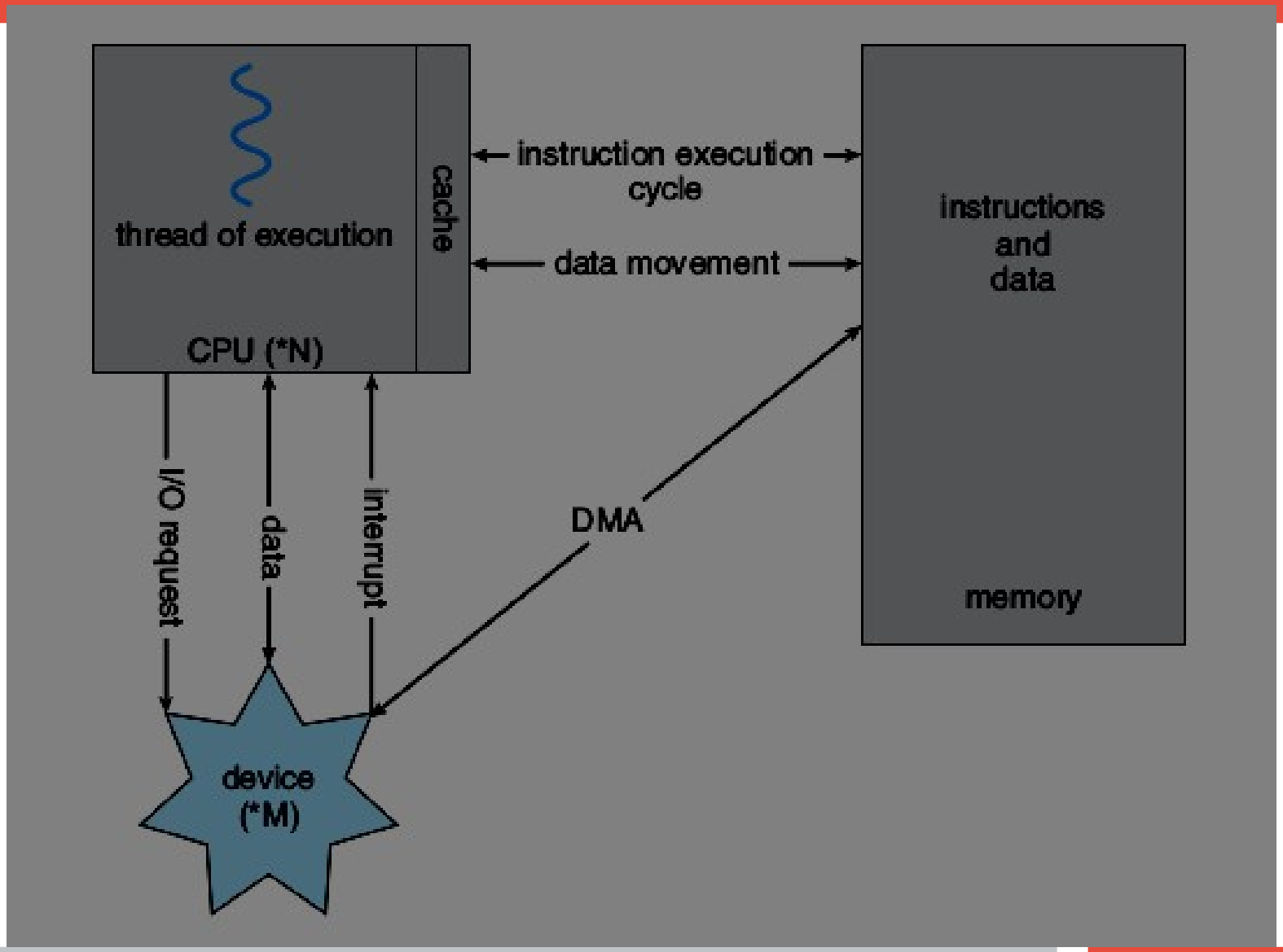
- Hides low-level details from OS
- Manages data flow between device and CPU/memory

Interconnects

Flow of data and information in the system

Switches map a fixed number of inputs to outputs, total number of ports on a switch

How a Computer Works?



Why to Learn How OS works?

Knowing how operating systems work is a fundamental and critical to anyone who is a serious software developer

When you write a program and it runs too slow, but you see nothing wrong with your code, where else will you look for a solution

Understand system performance

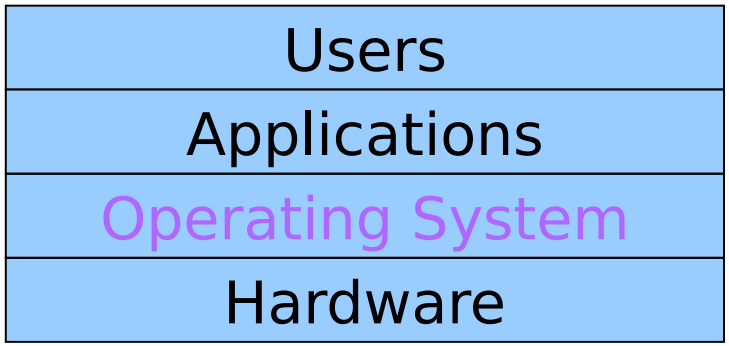
Operating System

No single definition

OS = all software minus applications

Operating system goals:

- **Helps the applications use the computer**
- **Helps manage the resources of the computer in an efficient manner**



Operating System Evolution

Single user

Batch processing

Multiprogramming

Graphical user interface

Single User

One user at a time on console

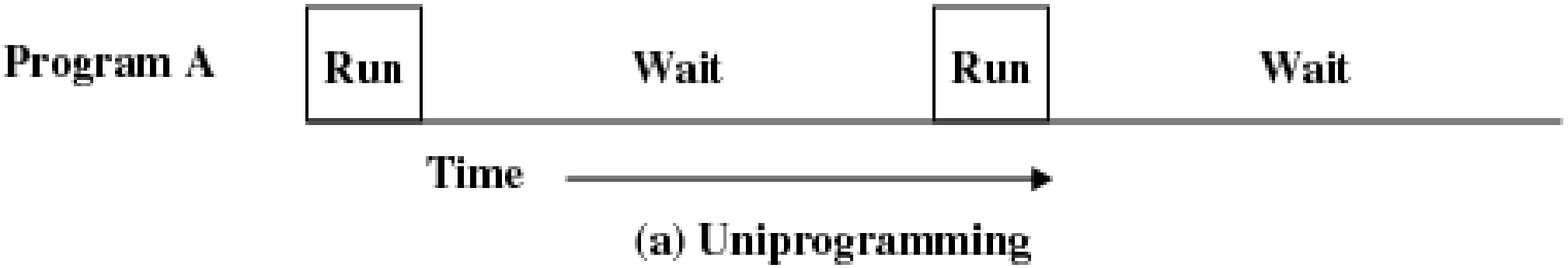
Computer executes one function at a time

No overlap: computation & I/O

User must be at console

Uniprogramming (~1960)

Processor must wait for I/O instruction to complete before preceding



(a) Uniprogramming

Batch Processing

Execute multiple consecutive “jobs” in batch:

- **Load program**
- **Run**
- **Print results, dump machine state**
- **Repeat**

Users submit jobs (on cards or tape)

Human schedules jobs

OS loads and runs jobs

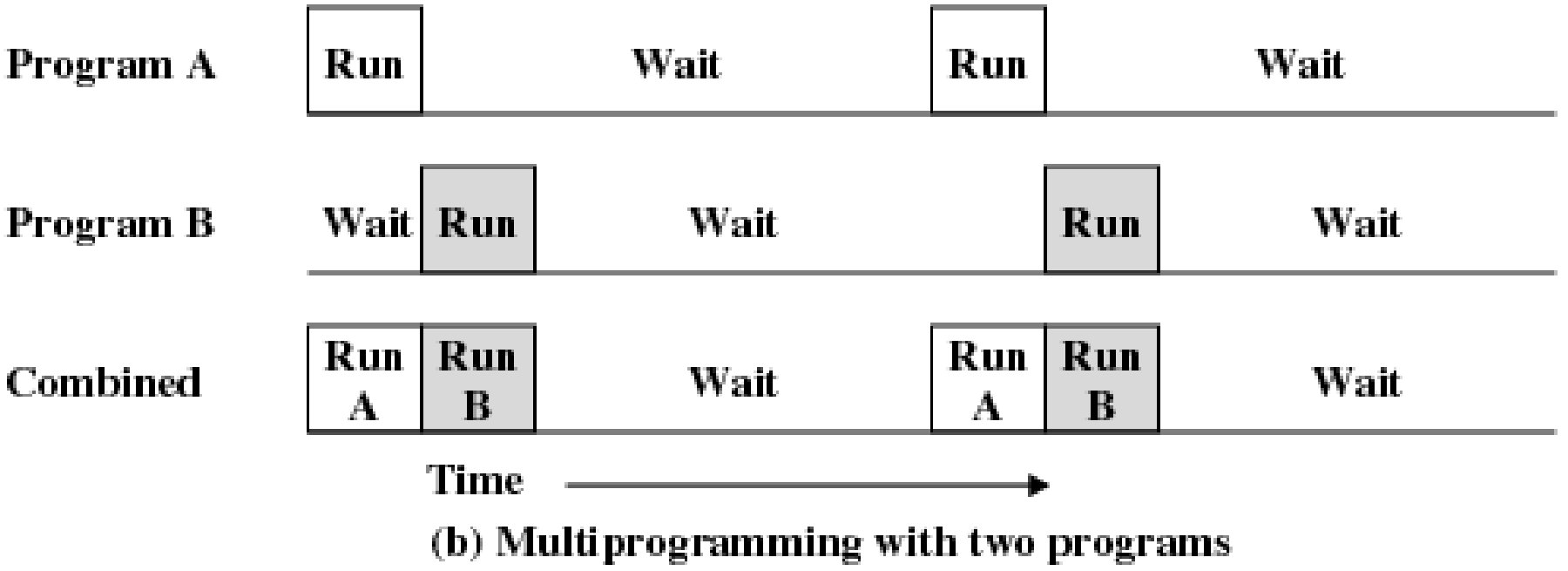
Multiprogramming

Allow several programs to run at the same time

- **Run one job until I/O**
- **Run another job, etc.**
- **OS manages interaction between programs**
  - **Which jobs to start**
  - **Protects program's memory from others**
  - **Decides which to resume when CPU available**
  - **TIME-SHARING**

Multiprogramming (~1961)

When one job needs to wait for I/O, the processor can switch to the other job



(b) Multiprogramming with two programs

Graphical User Interface

Separate window for each application

Menu-driven application environment

Operating System History

Multics (1960s), Unix

Users share system via terminals (mainframes)

DOS, MacOS (1980s)

Simple OS in PCs without multiprogramming, concurrency, memory protection, virtual memory

NT, MacOS X, Linux (1990s)

“Real” operating systems on PCs

Today

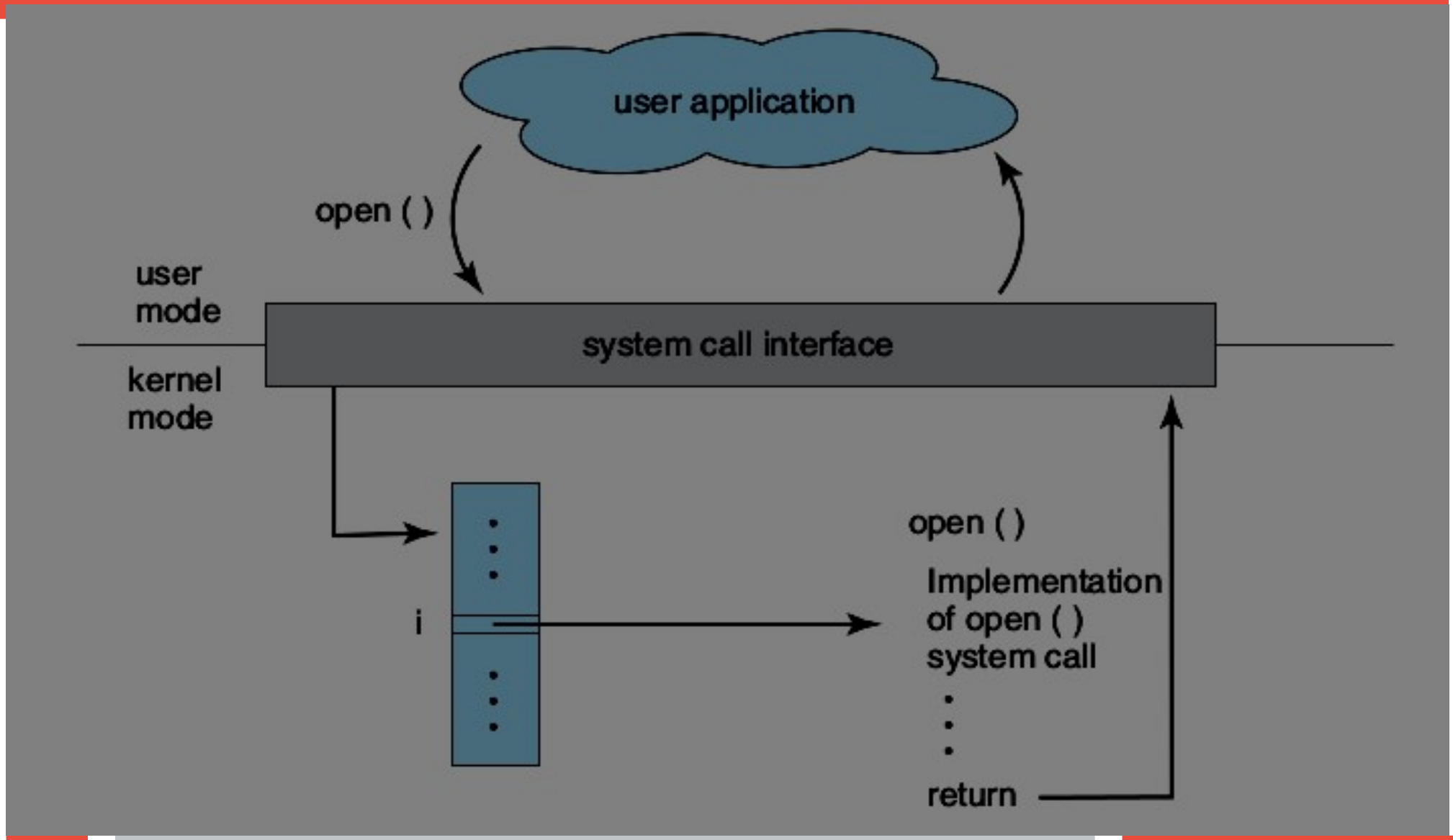
Parallel systems, mobile devices

System Calls

Programming interface to the services provided by the OS

Mostly accessed by programs via a high-level Application Programming Interface (API) rather than direct system call use

API – System Call – OS Relationship



Operating System Functionality

- **Certain services to users**
- **Load programs, run them**
- **Controls execution of programs to prevent errors and improper use of the computer**
- **Manages all resources (e.g., CPU, memory, disk)**
- **Decides between conflicting requests for efficient and fair resource use (prevent of applications from starvation and deadlock conditions)**

Operating System Services - Users

- User interface
- Program execution
- I/O operations
- File system manipulation
- Communication
- Error detection

Operating System Services - Resources

- **Resource allocation for processes (system and users)**
- **Accounting**
- **Protection and security**

Operating System Design Goals

User view:

- **convenient to use, easy to learn and to use, reliable, safe, and fast**
- **System view:**
- **easy to design, implement, and maintain; and flexible, reliable, error free, and efficient**
- **Specifying and designing an OS:**
  - **Highly creative task of software engineering**

Operating System Implementation

Early operating systems were written in assembly language

Now mostly C, C++

Operating System Components

- **Process scheduler - CPU**
  - **Determines when and for long each process executes**
- **Memory manager – Main memory**
  - **Determines when and how memory is allocated to processes**
  - **Decides what to do when main memory is full**
- **File system – I/O**
  - **Organizes named collections of data in persistent storage**
- **Networking - Interconnect**
  - **Enables processes to communicate with one another**
- **Protection and security**

Process Management

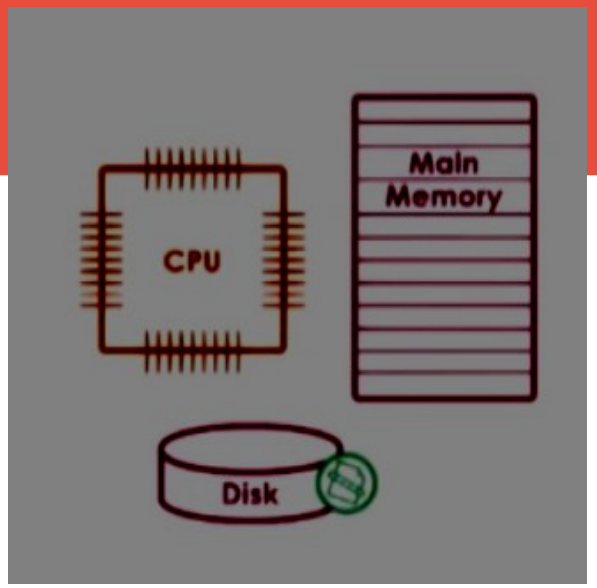
Process needs resources to accomplish its task; CPU, memory, I/O

Process management issues;

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

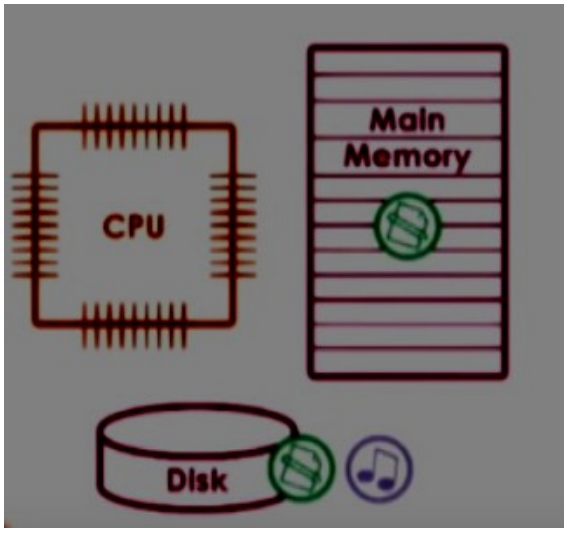
Process

application=program on disk



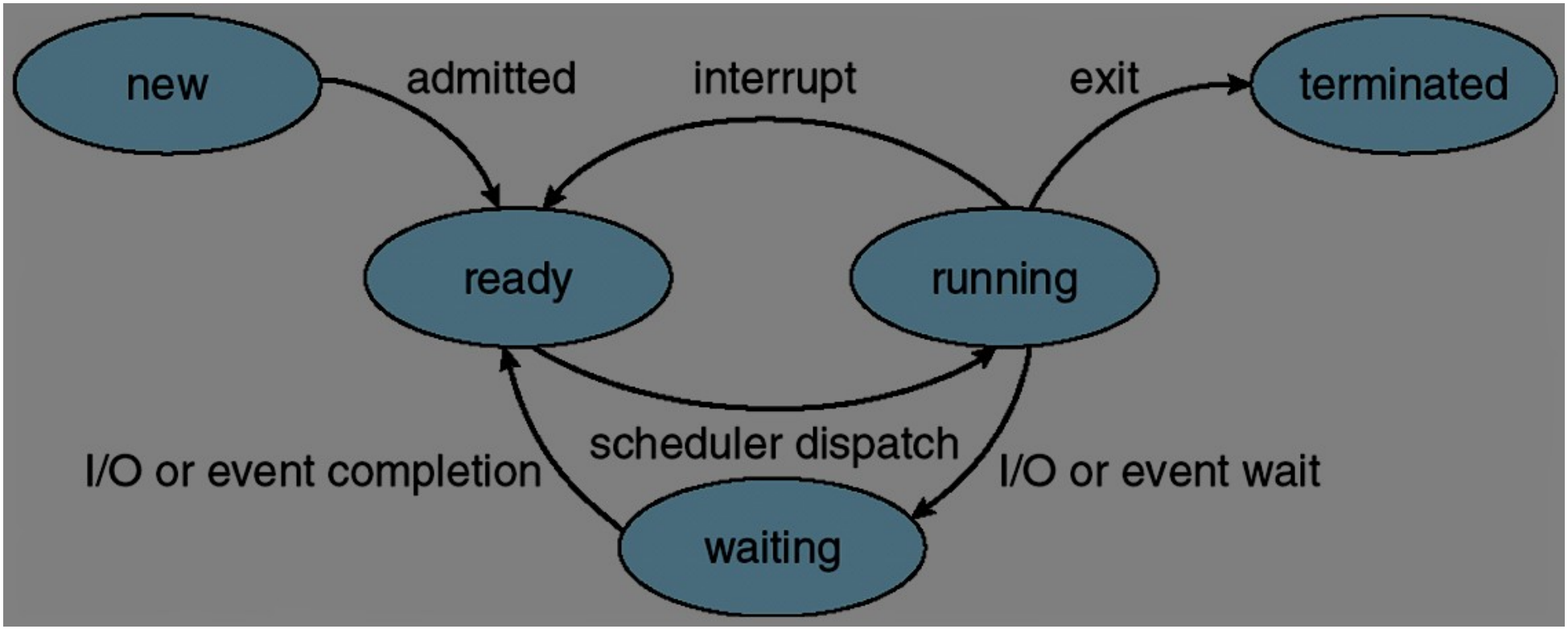
Program becomes process when executable file loaded

into memory

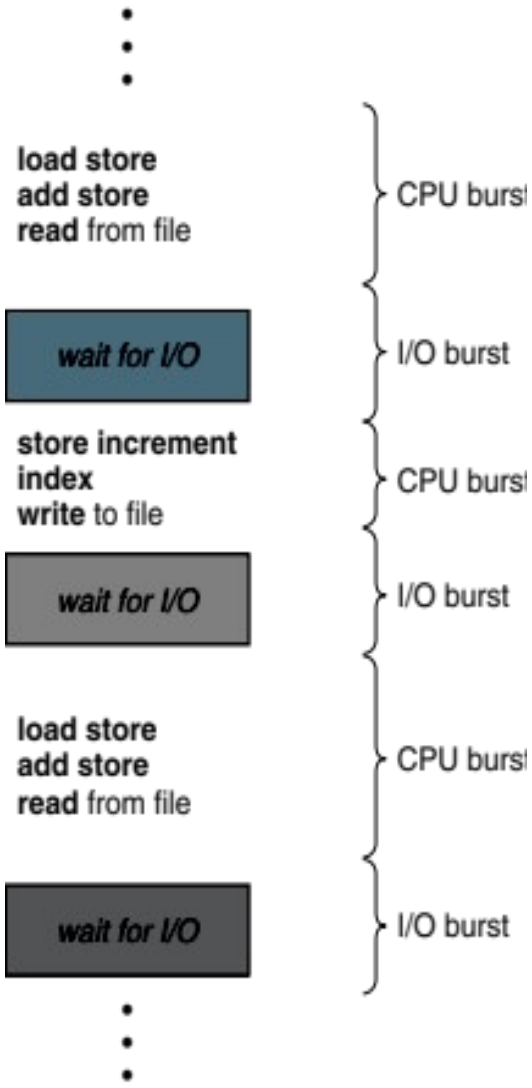


process=program in execution

Process Lifecycle



Burst Times


load store
add store
read from file
CPU burst
wait for I/O
I/O burst
store increment
index
write to file
CPU burst
wait for I/O
I/O burst
load store
add store
read from file
CPU burst
wait for I/O
I/O burst

Process Scheduler

Determines which one of the currently ready processes will be dispatched to the CPU to start running, and how long it should run for

OS must;

- **Preempt: interrupt and save current context**
- **Schedule: run scheduler to choose next process**
- **Dispatch: dispatch process and switch into its context**

Maximize CPU use, quickly switch processes onto CPU for time sharing

Be efficient!

First- Come, First-Served (FCFS) Scheduling

Process	Burst Time
P_1	24
P_2	3
P_3	3

Suppose that the processes arrive in the order: P_1 , P_2 , P_3

The Gantt Chart for the schedule is:

Waiting time for P_1 = 0; P_2 = 24; P_3 = 27

Average waiting time: (0 + 24 + 27)/3 = 17

First- Come, First-Served (FCFS) Scheduling

Suppose that the processes arrive in the order:

P_2 , P_3 , P_1

The Gantt chart for the schedule is:

Waiting time for $P_1 = 6; P_2 = 0, P_3 = 3$

Average waiting time: (6 + 0 + 3)/3 = 3

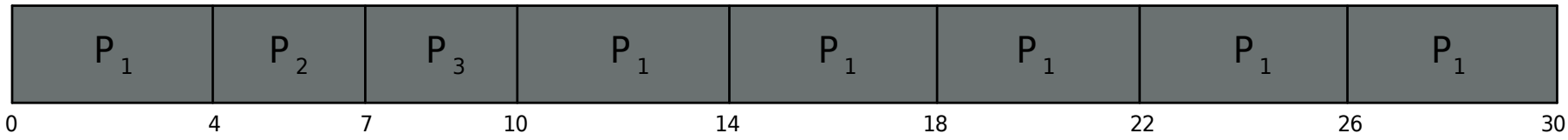
Much better than previous case

Round Robin

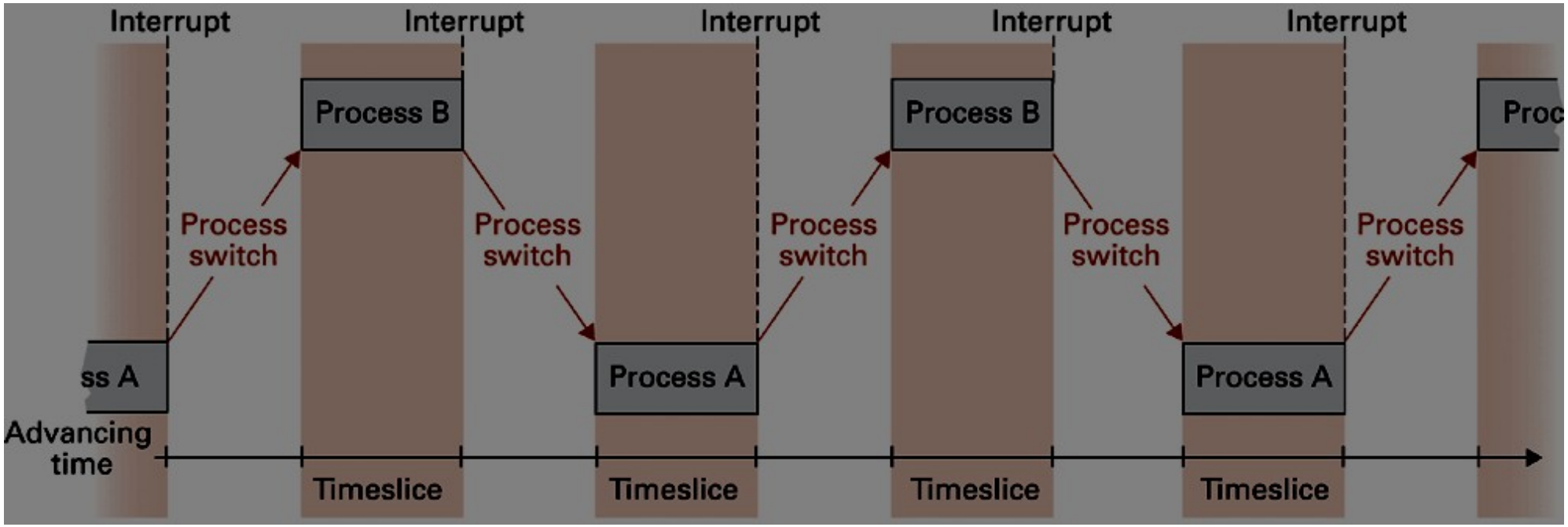
Each process gets a small unit of CPU time (time quantum *q*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.

RR with q=4:

The Gantt chart is:

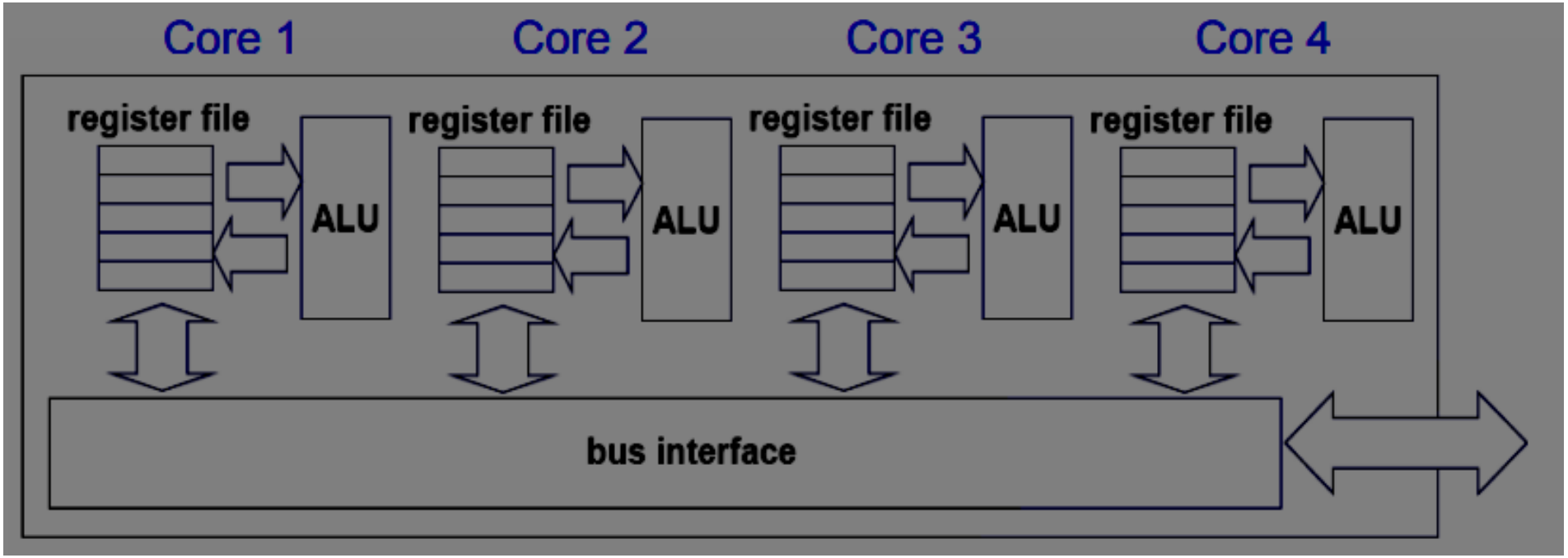


Time-Sharing Between Two Processes



Multicore Processors

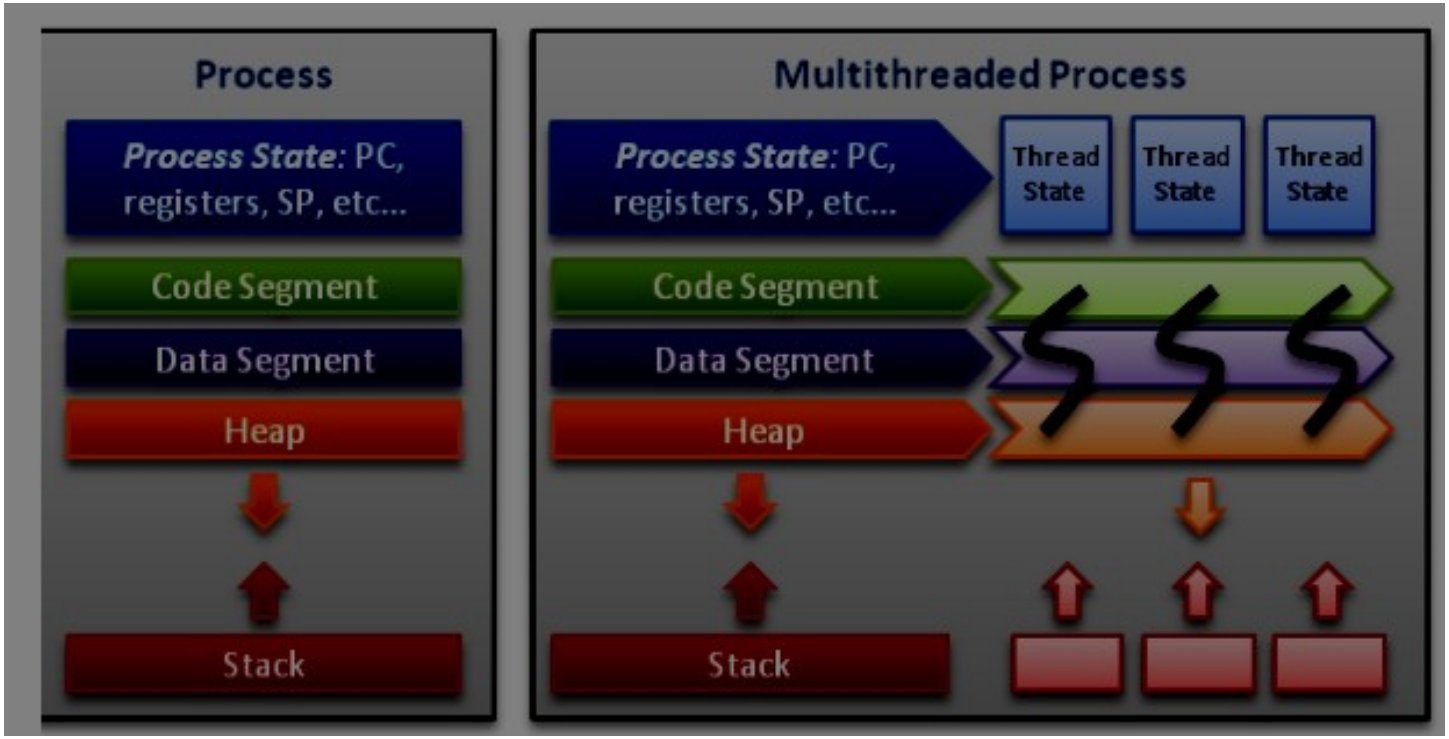
Multiple CPUs on a single chip



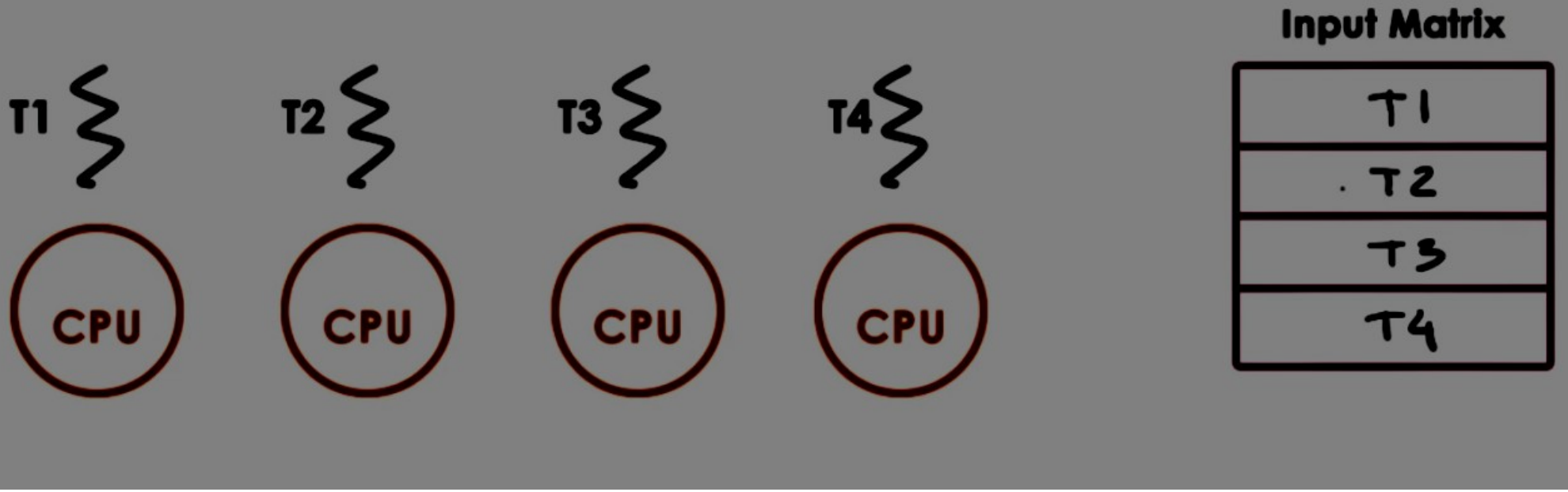
Threads

Threads use, and exist within, the process resources

Scheduled and run as independent entities



Benefits of Multithreading



Parallelization --> speed up

Specialization --> hot cache!

Efficiency --> lower memory requirement, cheaper communication overhead

Memory Management

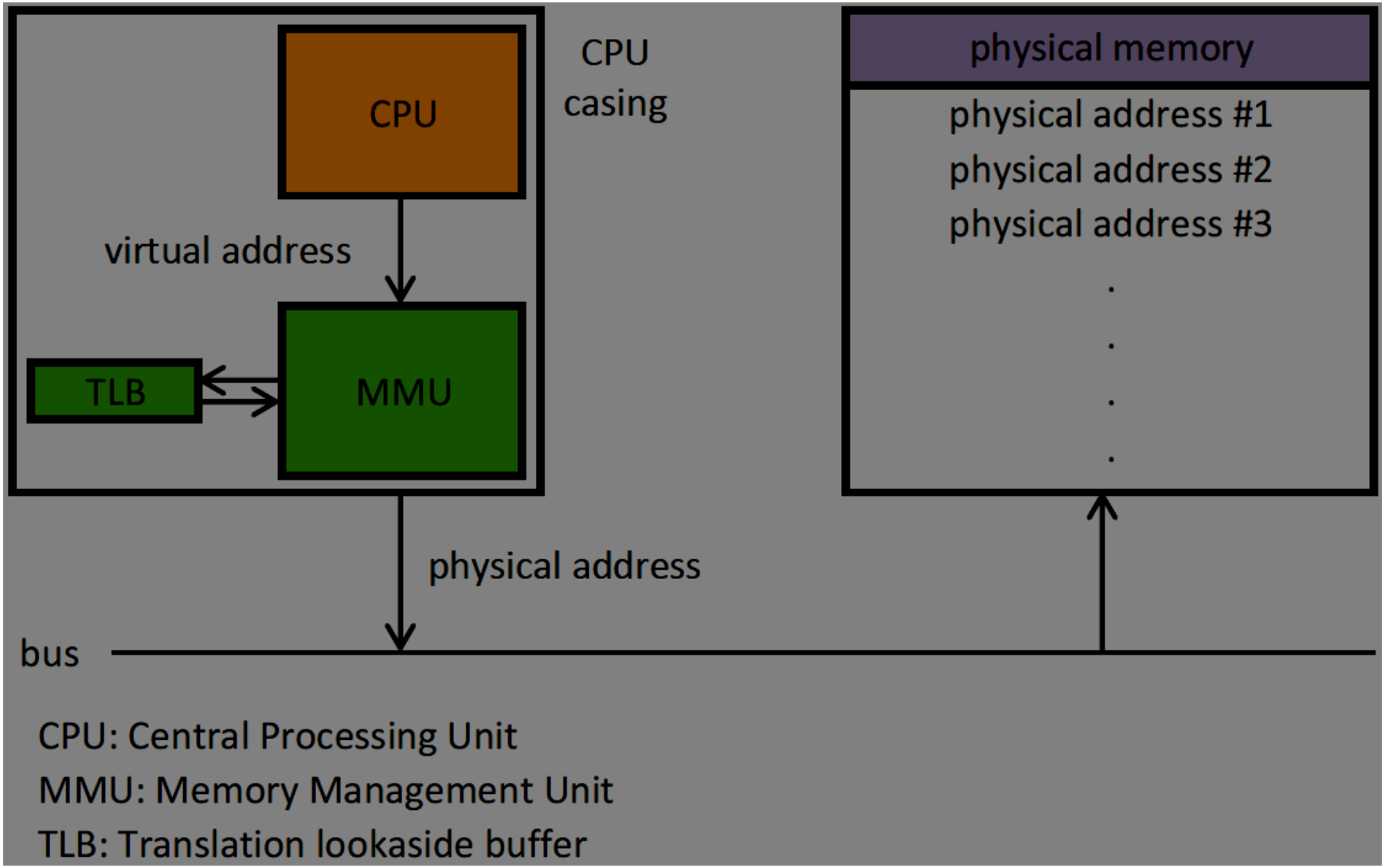
To execute a program all (or part) of the instructions must be in memory

All (or part) of the data that is needed by the program must be in memory

Memory management issues:

- **Keeping track of which parts of memory are currently being used and by whom**
- **Deciding which processes and data to move into and out of memory**
- **Allocating and deallocating memory space as needed**

Memory Management Unit



Storage Management

File system management:

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

Mass 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

I/O subsystem:

- To hide features of hardware devices from the user
- Drivers for hardware devices

Networking

Let another process know that some event has occurred

Transfer data from one process to another

Protection and Security

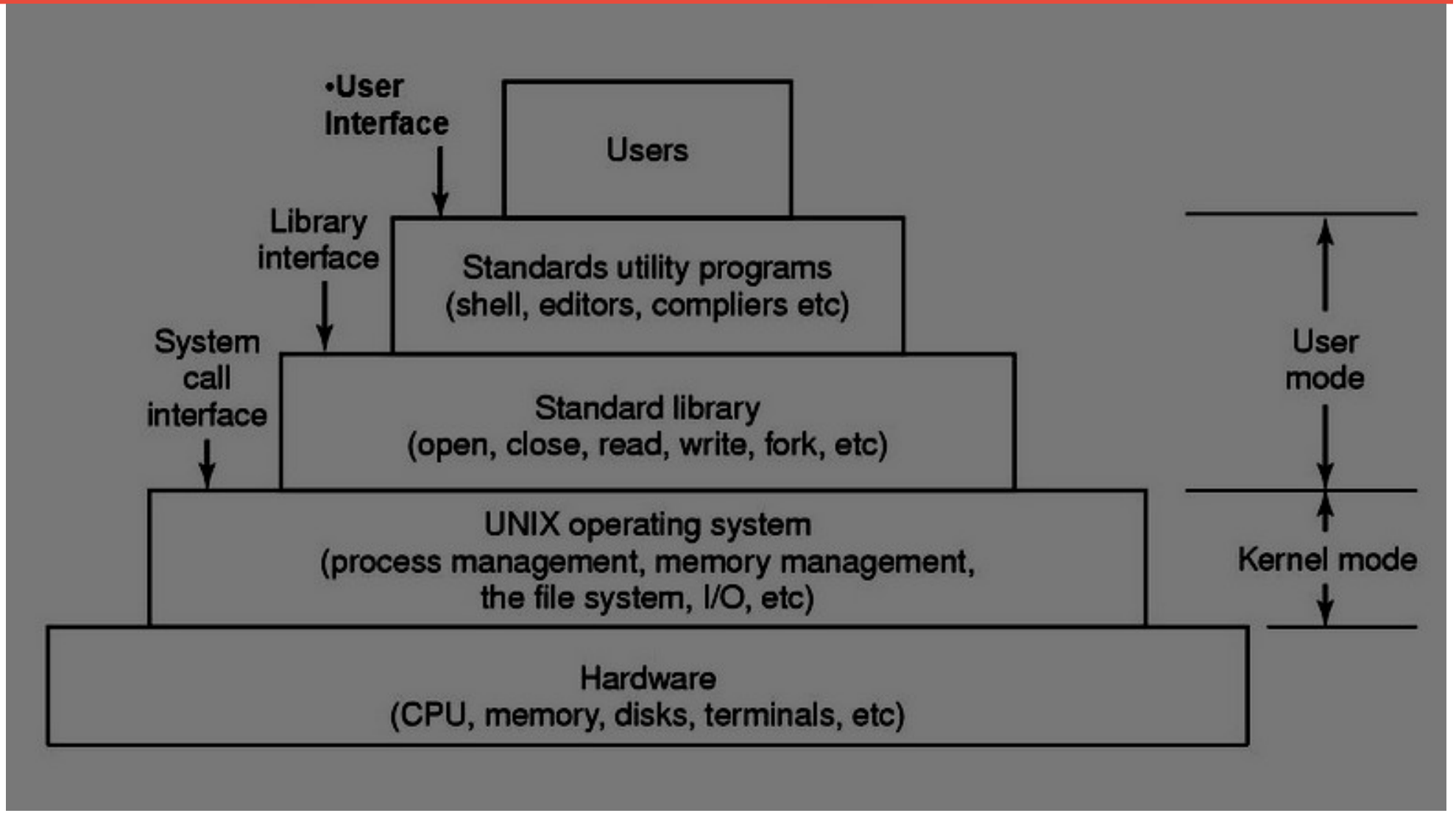
Protection

- 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
- Insecure passwords, auditing software
- Unruly processes

Linux Architecture



When You Press the Power Button?