Operating Systems

CMP-320

Instructor: Muhammad Adeel Nisar

Lecture: 1, 2

October 21, 23, 2014

Acknowledgement

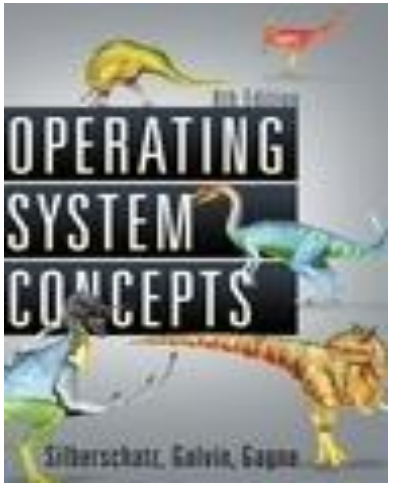
Slides and pictures are adapted from Lecture slides / Books of

- Dr. Syed Mansoor Sarwar.
- Mr. M. Arif Butt
- Text Book - OS Concepts by Silberschatz, Galvin, and Gagne.

Today's Agenda

- Basic Course information
- Course Outline
- Computer System Overview
  - Processor Registers
  - Instruction Execution
  - Memory Hierarchy & Cache Memory
  - Interrupts
  - I/O, Memory and CPU Protection
- Operating System Overview
  - Definition
  - goals

Course Information



- **Required Textbook:** Operating System Concepts, 8th Edition Silberschatz, Galvin, Gagne
- **Resources @:** \\printsrv\Adeel Nisar\
- **Grades Website:** http://online.pucit.edu.pk
- **Co-requisites :** Data Structures
- **Follow up:** None
- **Office:** Faculty Room (Ground Floor)
- **Students Counseling hours :**
  - **Mon, Wed : 8:30 – 10:30 & 4:00 – 5:00**
- **email:** adeel.nisar@pucit.edu.pk

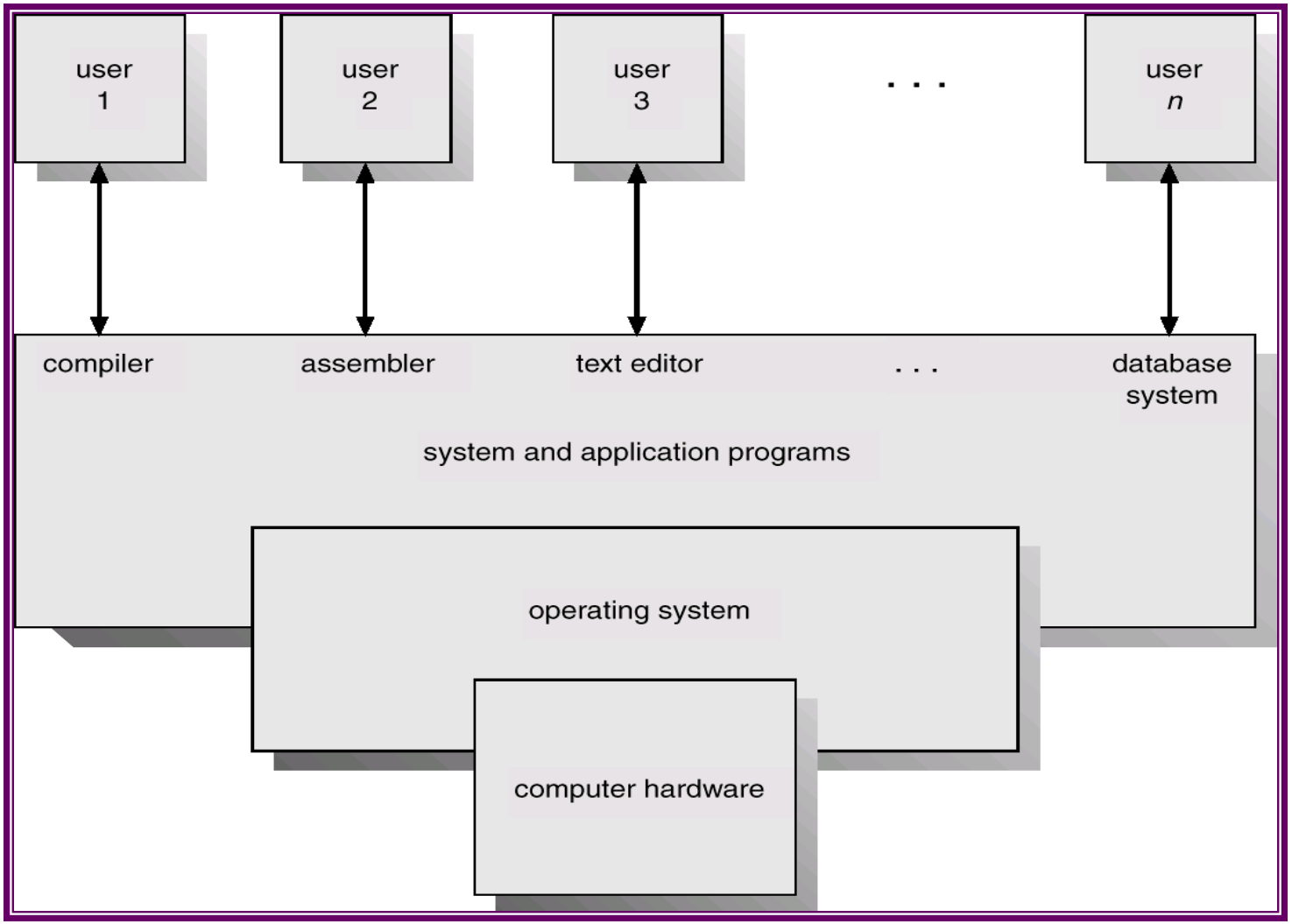
Course Outline

- Operating system concepts
- Operating system structures
- Processes and threads—scheduling, concurrency, synchronization, etc.
- Deadlocks
- Memory management
- Virtual memory
- File system
- Secondary storage management

Computer System

1. Hardware
2. Operating system
3. Applications programs
4. Users

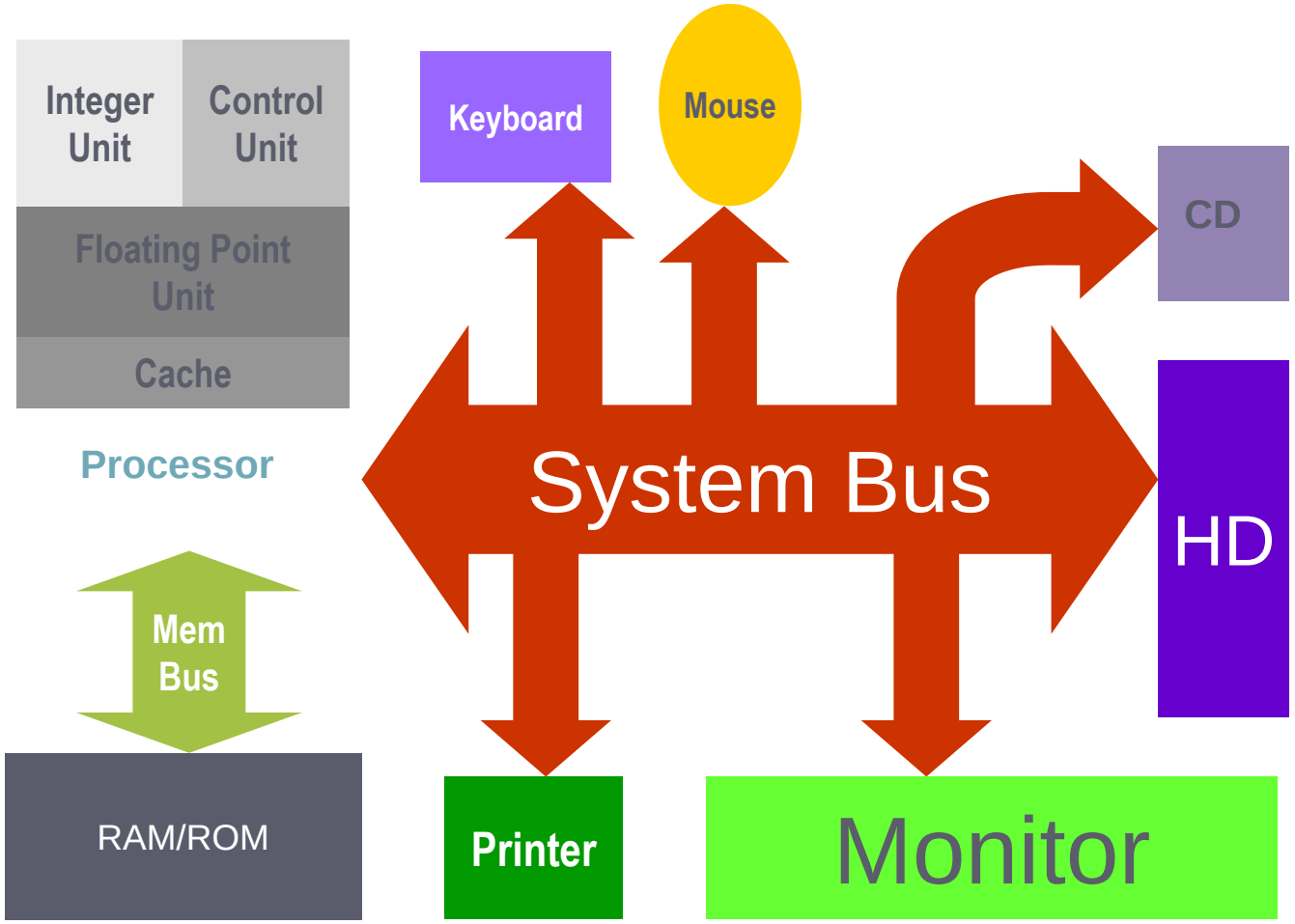
Layered View of a Computer System



Purpose of a Computer System

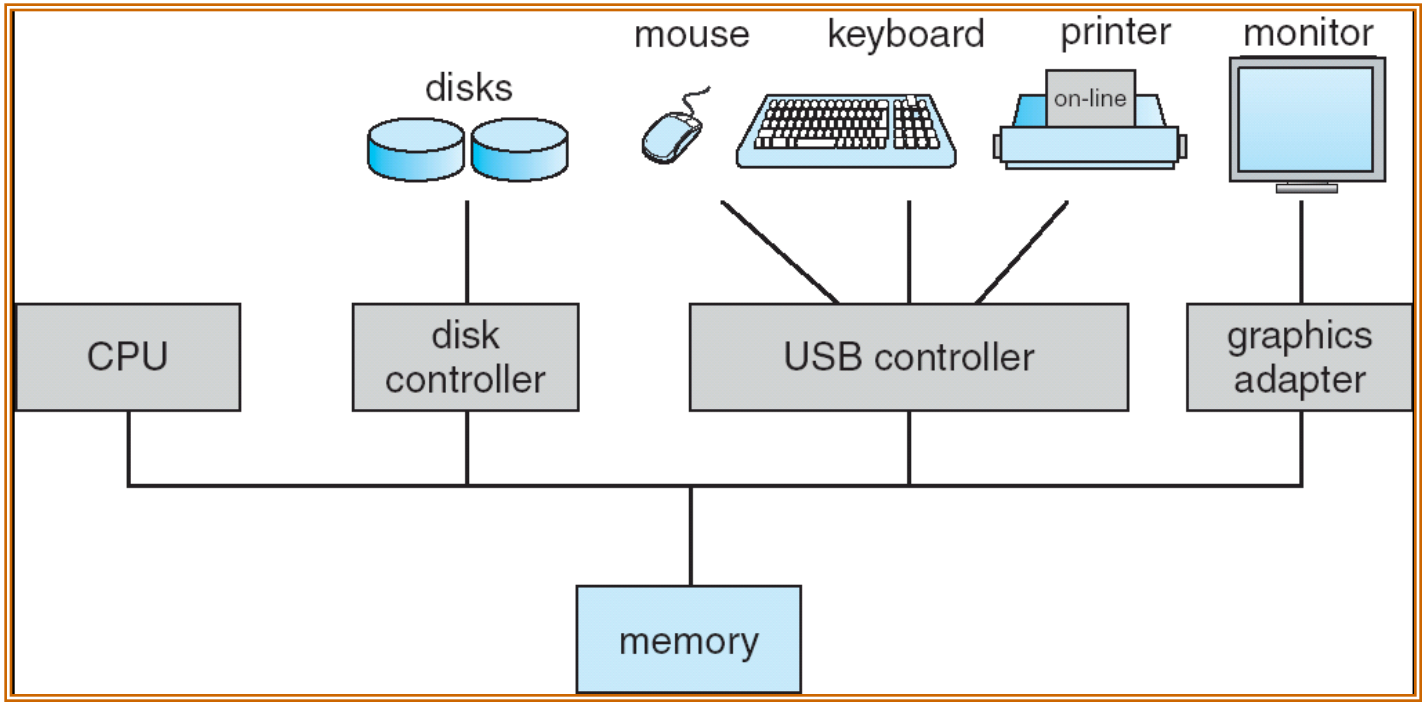
- Computer systems consist of software and hardware that are combined to provide a tool to solve specific problems in an efficient manner
- Execute programs

Computer System Organization

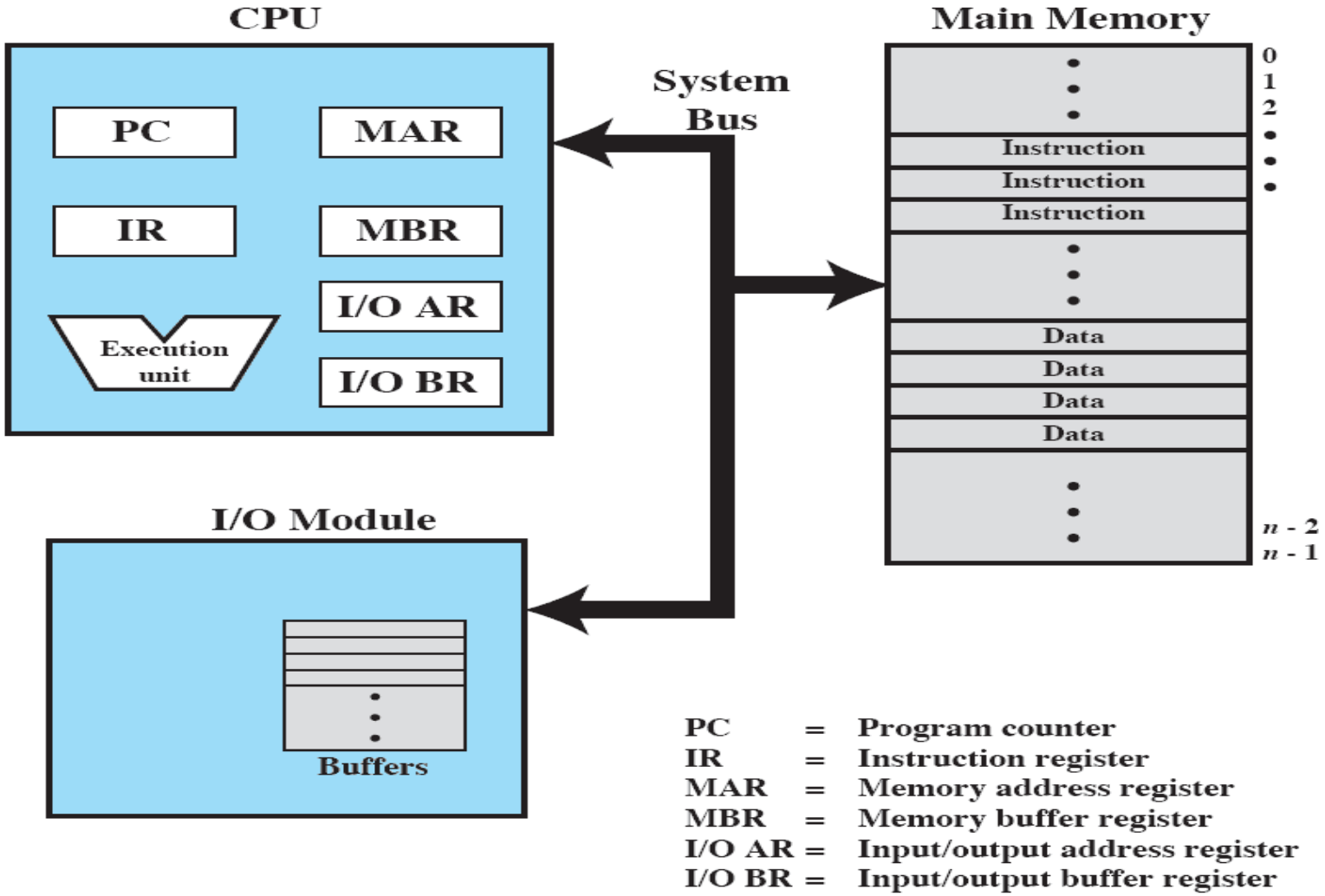


Computer System Organization

- 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 Organization

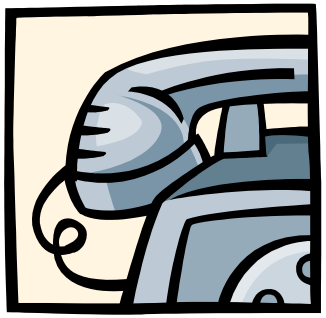


Instruction Execution

- Fetch Instruction
- Decode Instruction
- Fetch Operand(s)
- Execute Instruction
- Store Operand(s)

Interrupts, Traps, and Signals

- The occurrence of an event is usually signaled by an interrupt from either the hardware or the software.
- Interrupt is an event generated by an I/O device to get the attention of CPU and control goes to the OS. State of the CPU is saved, ISR is executed and then state of CPU is restored.
- Software may trigger an interrupt by executing a special operation called a system call.
- Trap is an event generated by CPU and control goes to the OS. Normally when an instruction is executed that may cause a division by zero or protection error. State of CPU is not saved and TSR is executed.



Answer the Phone

.
.
.

Resume

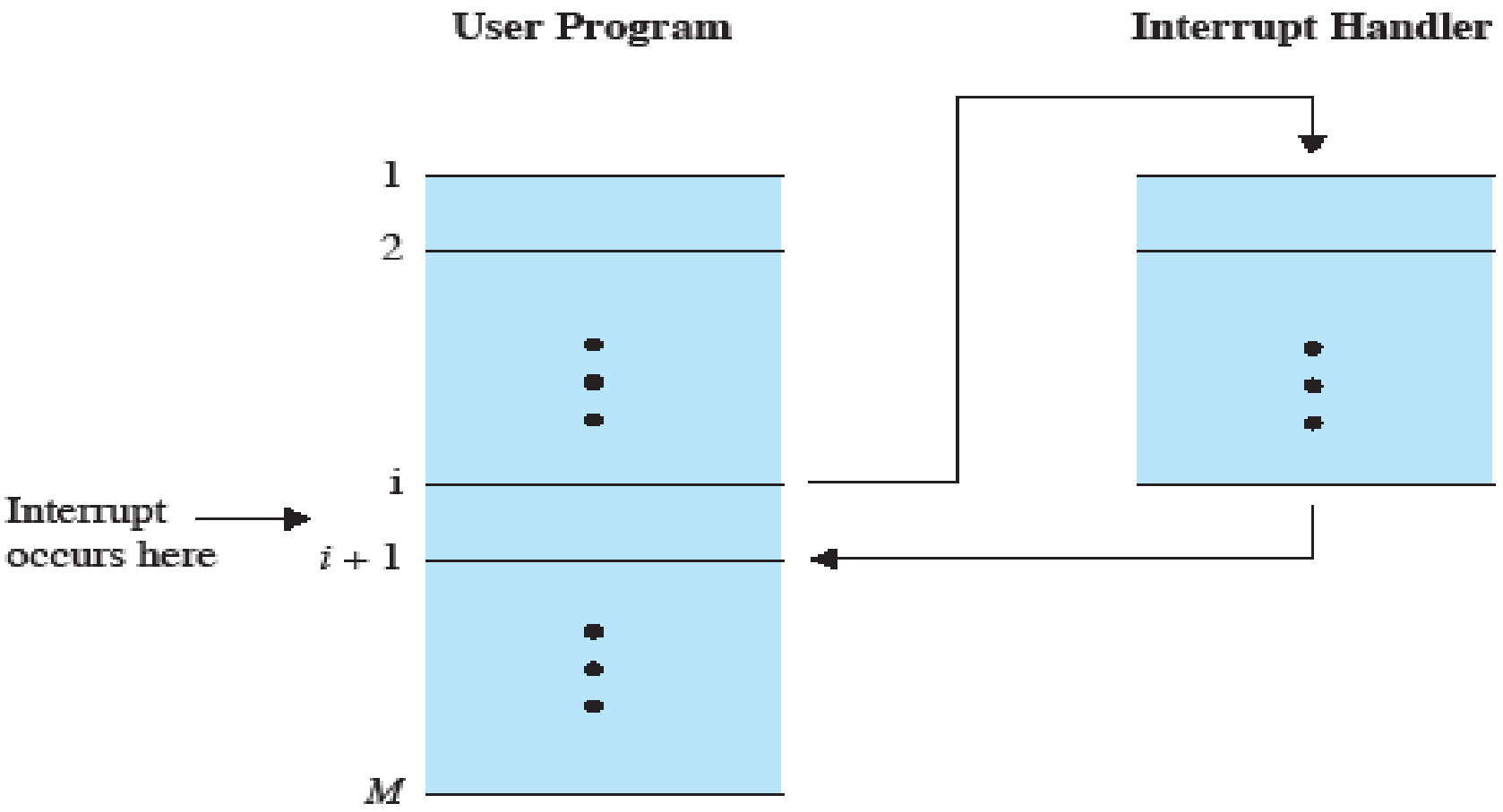
Interrupts, Traps, and Signals

- Signal. An event given to a process not to CPU. Since a process is a soft entry so signal is called soft interrupt. In case of a signal the process can take one of there possible actions:
  - A default action defined by OS.
  - A programmer specified action for signal handling.
  - Ignore signal.

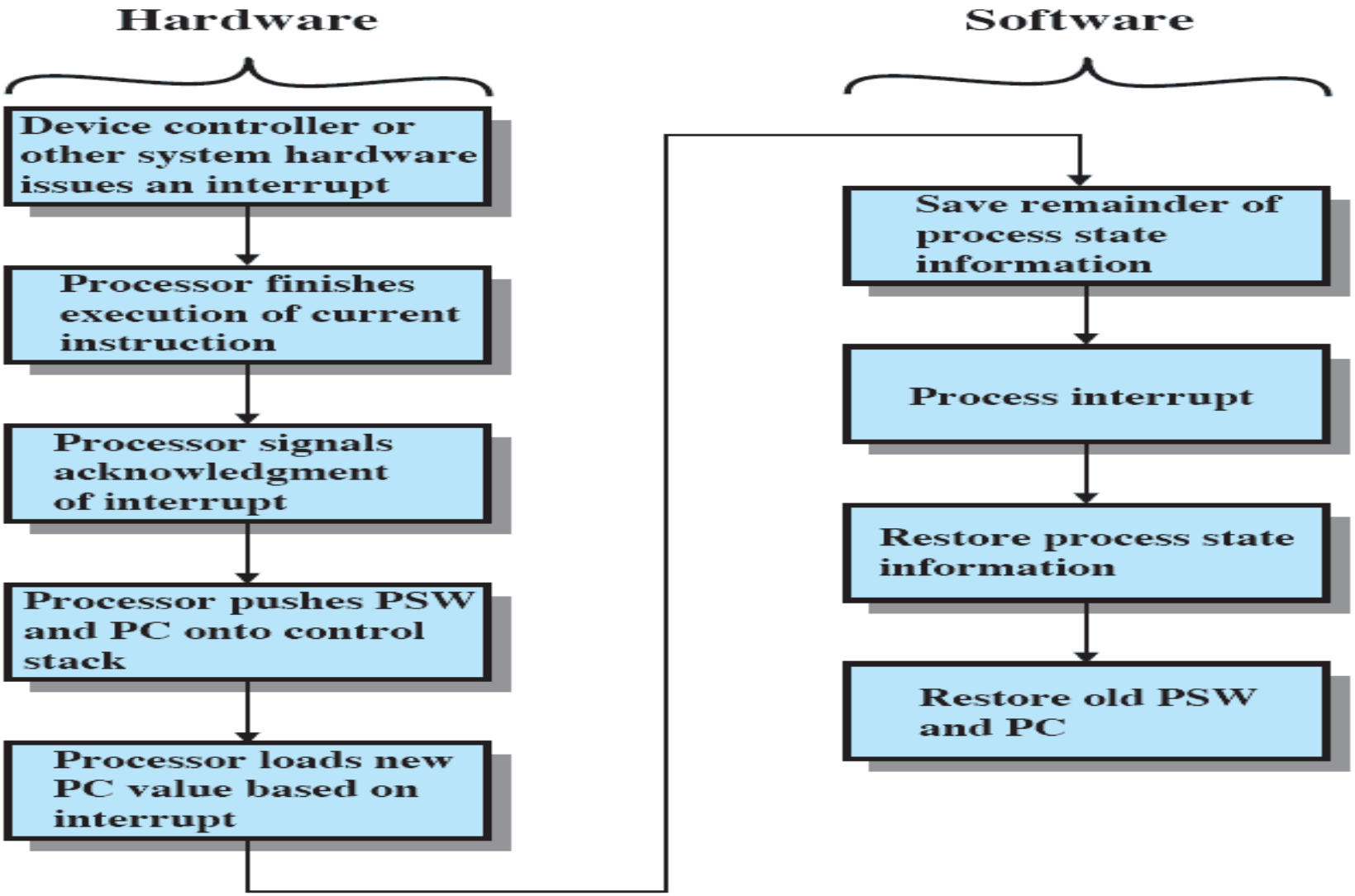
Interrupt Handling

- Interrupt transfers control to the interrupt service routine, generally, through the *interrupt vector*, which contains addresses of all the interrupt service routines.
- Interrupt architecture must save the address of the instruction after the interrupted instruction and the CPU state so that execution of the interrupted process may continue after the interrupt has been serviced.
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent *lost interrupts*.
- An operating system is *interrupt* driven.

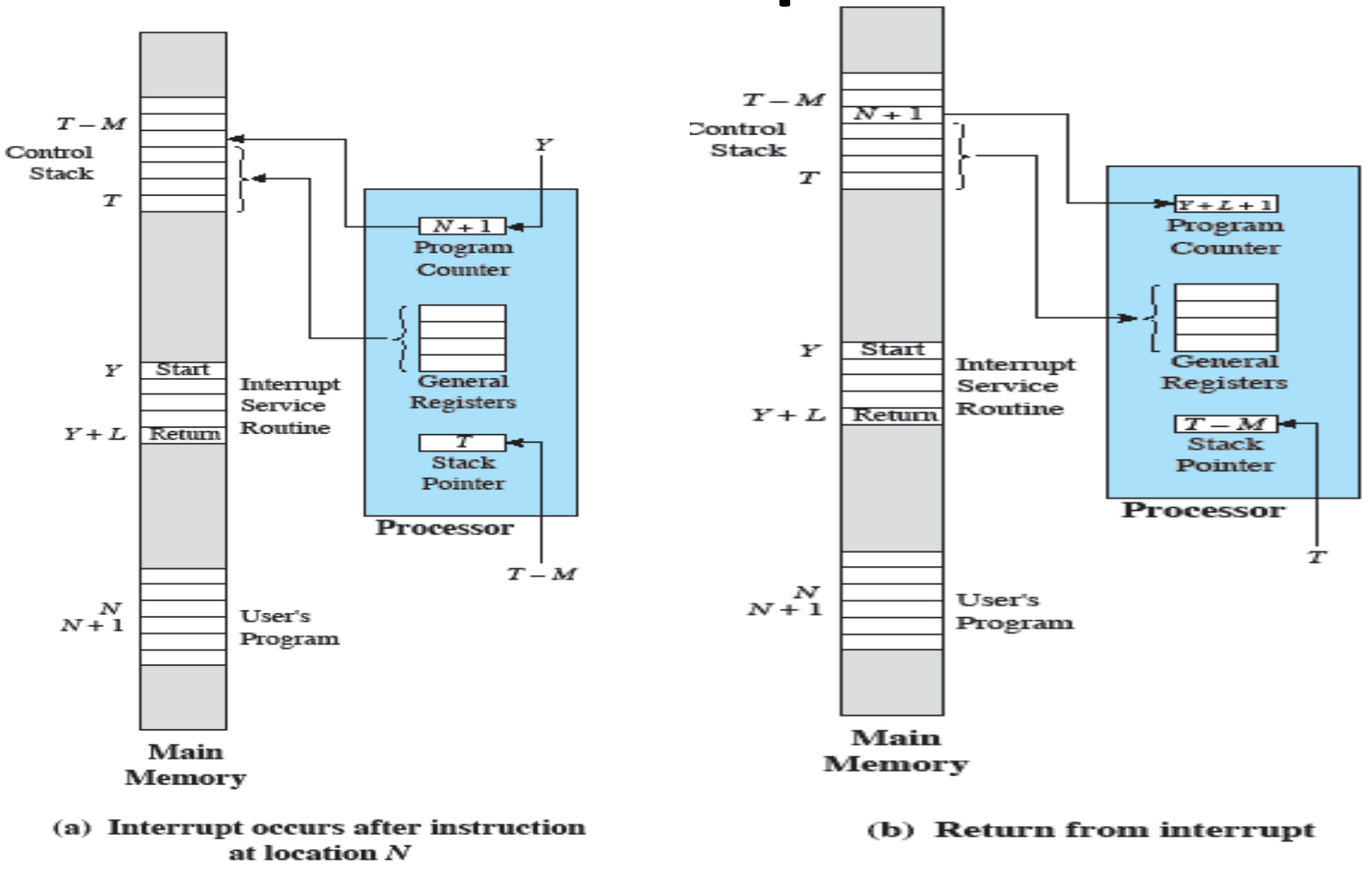
Transfer of Control via Interrupts



Simple Interrupt Processing



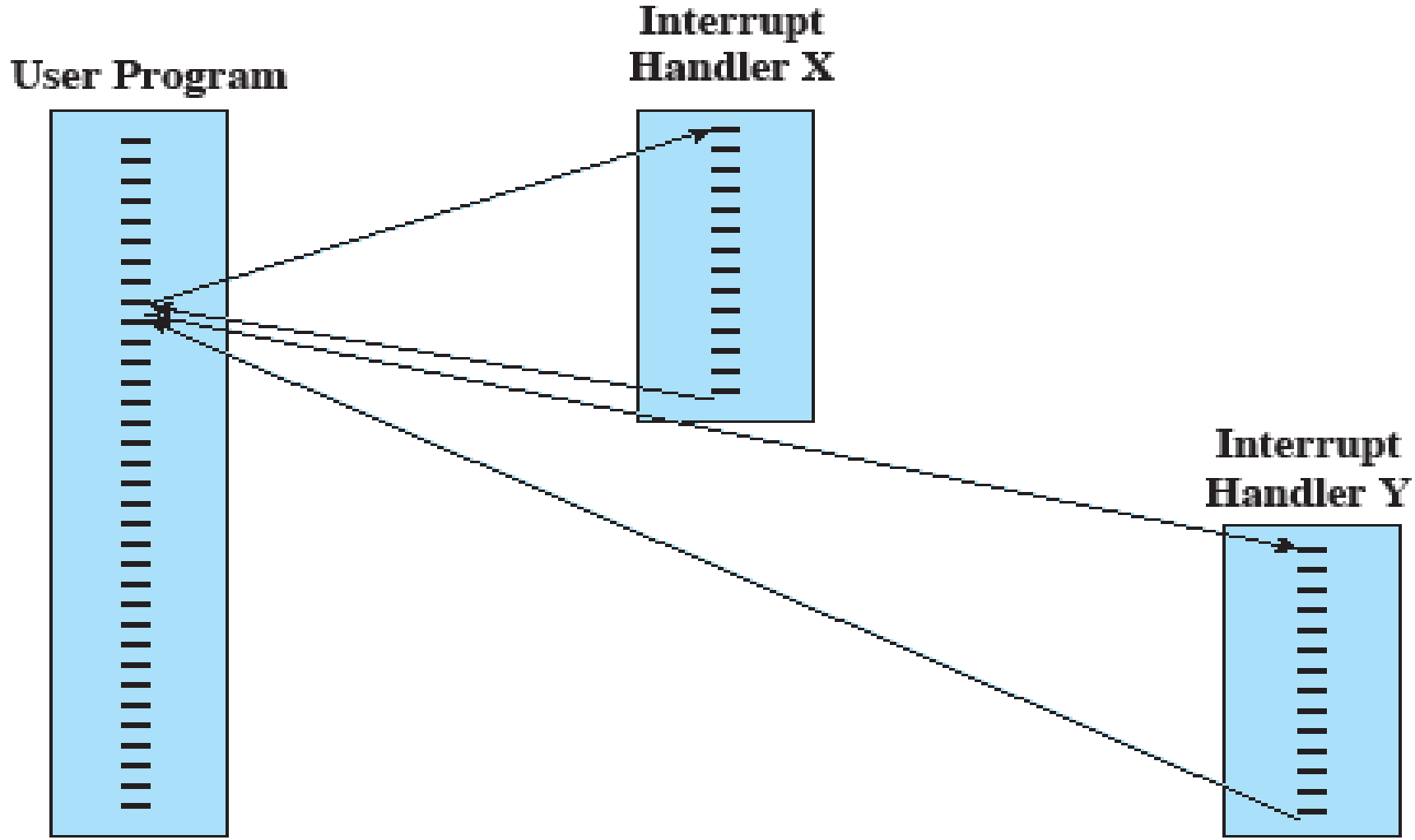
Changes in Memory and Registers for Interrupt



Multiple Interrupts

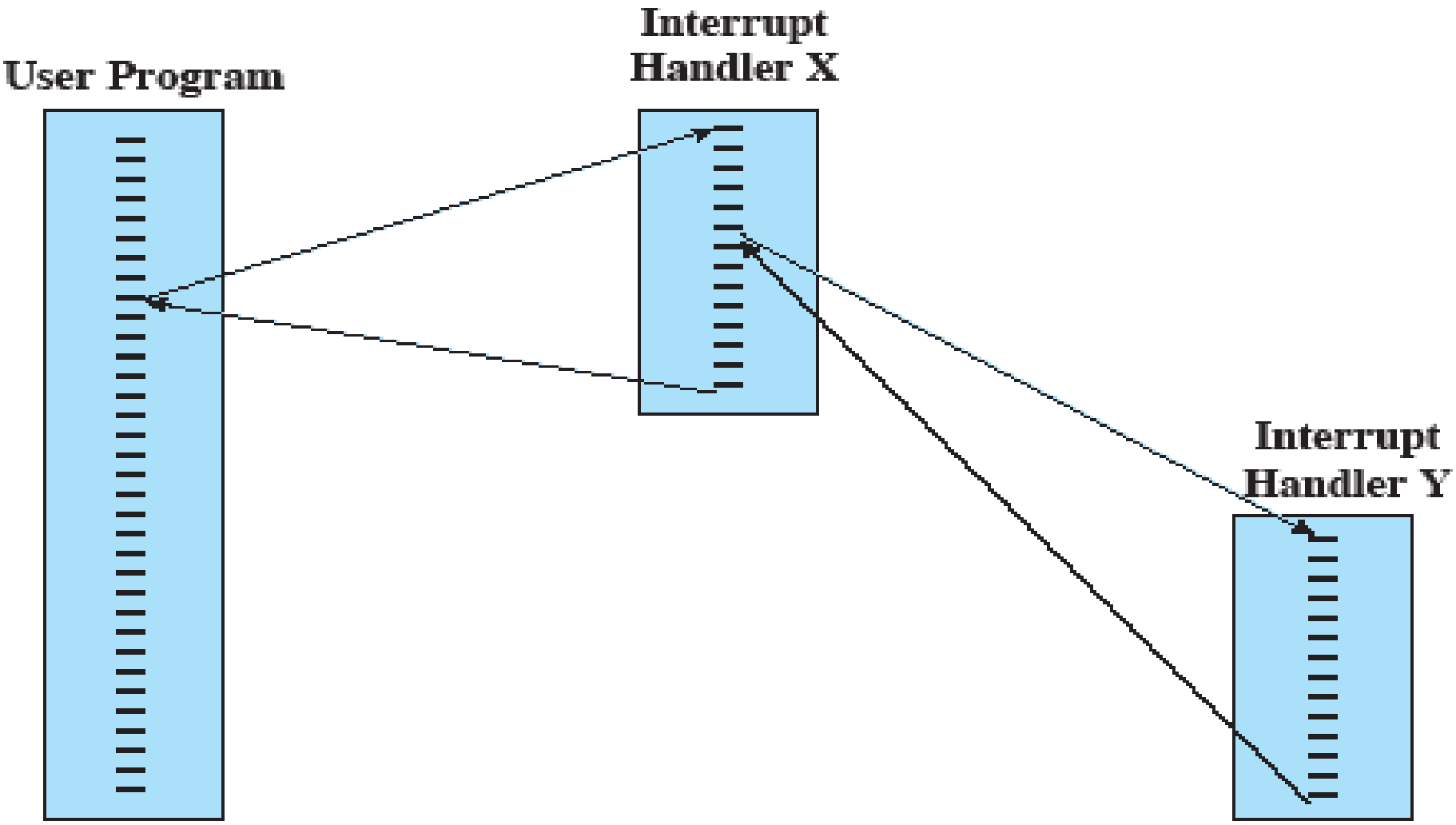
- Suppose an interrupt occurs while another interrupt is being processed.
  - E.g. During printing, data being received via communications line.
- Two approaches:
  - Disable interrupts during interrupt processing
  - Use a priority scheme.

Sequential Interrupt Processing



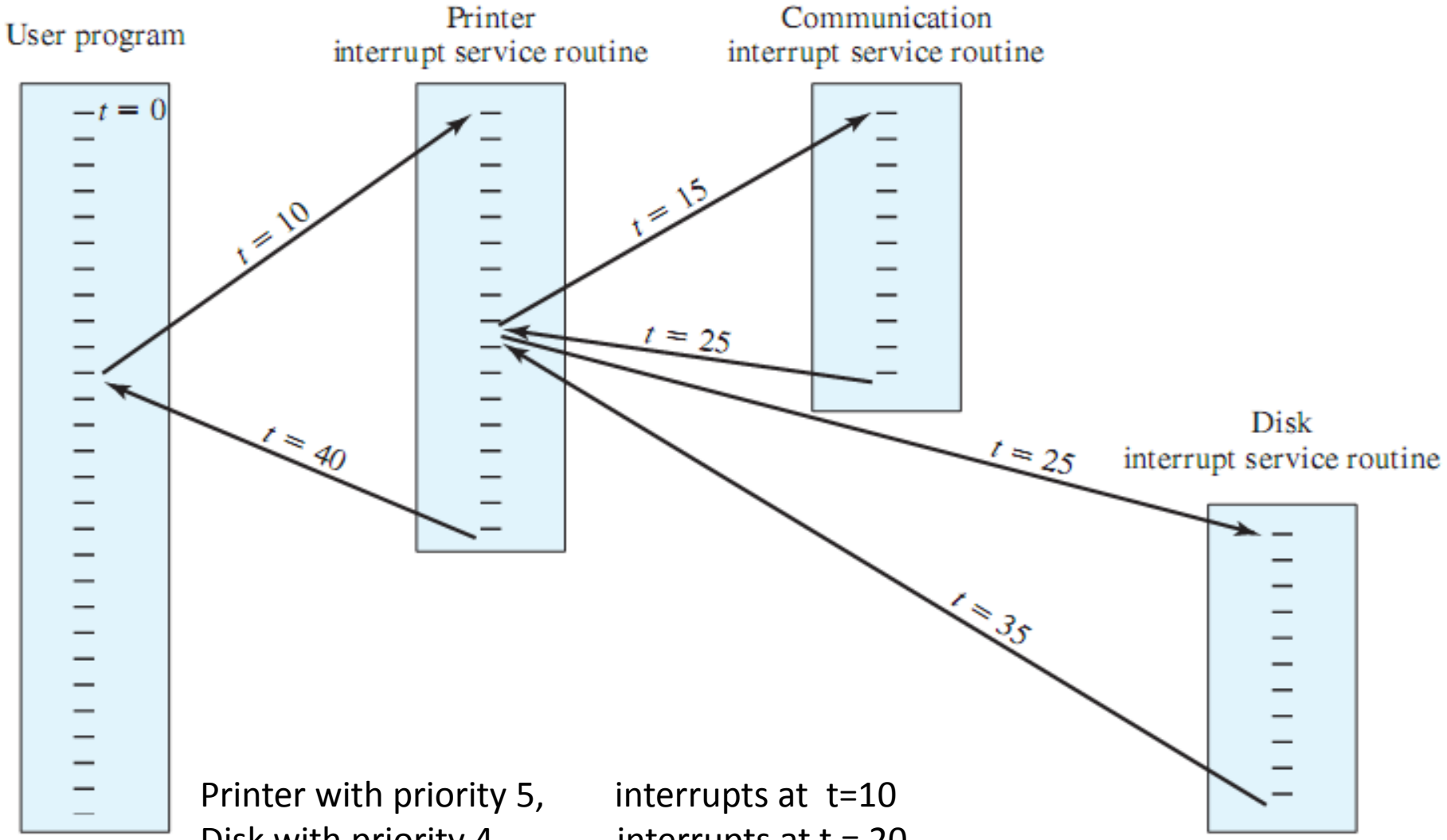
(a) Sequential interrupt processing

Nested Interrupt Processing



(b) Nested interrupt processing

Example of Nested Interrupt



Printer with priority 5, interrupts at t=10
Disk with priority 4, interrupts at t = 20
Comm line with priority 3, interrupts at t = 15

Memory Hierarchy

- Major constraints in memory
  - Amount
  - Speed
  - Expense
- Faster access time, greater cost per bit
- Greater capacity, smaller cost per bit
- Greater capacity, slower access speed

Memory Hierarchy

- Going down the hierarchy
  - Decreasing cost per bit
  - Increasing capacity
  - Increasing access time
  - Decreasing frequency of access to the memory by the processor

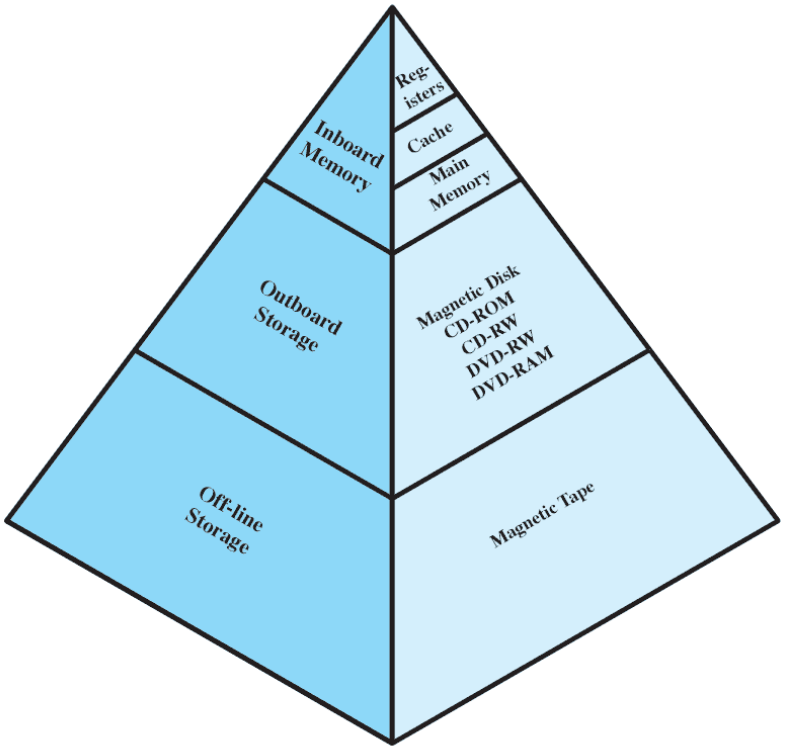
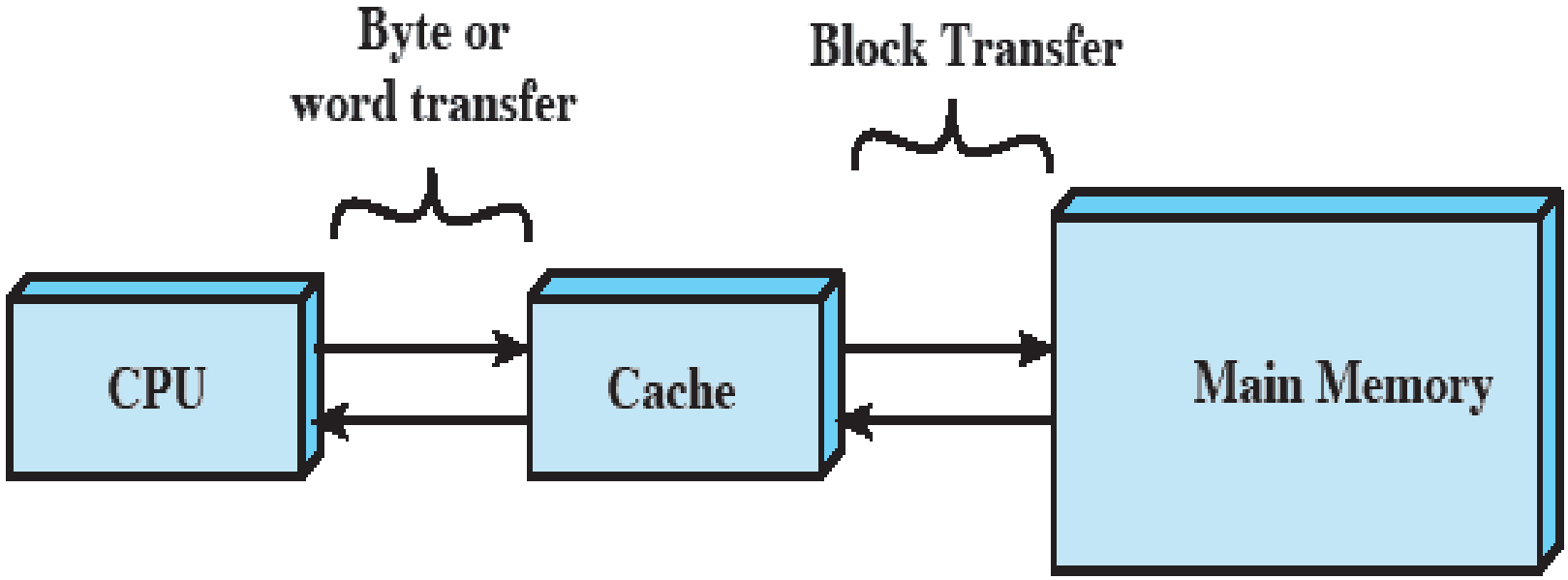


Figure 1.14 The Memory Hierarchy

Cache Memory

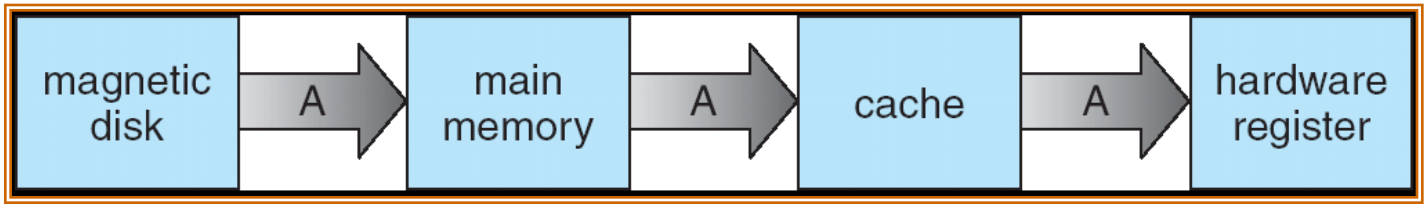
- Invisible to the OS
  - Interacts with other memory management hardware
- Processor must access memory at least once per instruction cycle
  - Processor speed faster than memory access speed
- Exploit the principle of locality (Temporal and Spatial)
  - **Temporal Locality.** If data is referenced at time **t**, then it might get referenced at time **t+1**
  - **Spatial Locality.** Data which is required soon is often close to the current data

Cache and Main Memory



- Caches Contains copy of a portion of main memory
- Processor first checks cache
  - If not found, block of memory read into cache
- Because of locality of reference, likely future memory references are in that block

Migration of Data from Disk to Register



Computer System Architecture

- Single Processor Systems
- Multiprocessor Systems
  - Increased throughput
  - Economy of scale
  - Increased reliability
- Clustered Systems

What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware—provides the user a simpler (virtual) machine to work with
- A program that allocates and deallocates computer system resources in an efficient, fair, and secure manner—a resource manager

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.

History of Operating Systems

(Reading Assignment)

- **First generation 1945 – 1955**
  - vacuum tubes, plug boards (no OS, signup sheets or punch cards)
- **Second generation 1955 – 1965**
  - transistors, *batch* systems
- **Third generation 1965 – 1980**
  - ICs and multiprogramming
- **Fourth generation 1980 – 1995**
  - personal computers, networks and distributed systems
- **Fifth generation 1995 – present**
  - Mobile systems

Types of Operating Systems

- Single programming
- Multiprogramming
- Time sharing
- Real time
  - Hard real time
  - Soft real time

Single User Systems

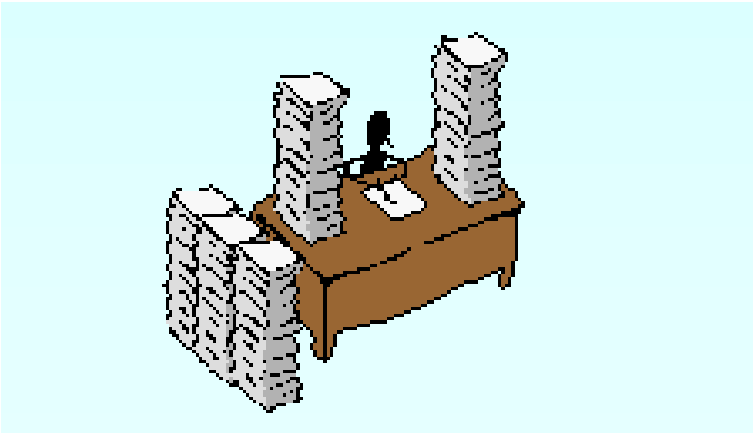
- *Personal computers* – computer system dedicated to a single user.
- Interactive
- User convenience and responsiveness.

Single User Systems

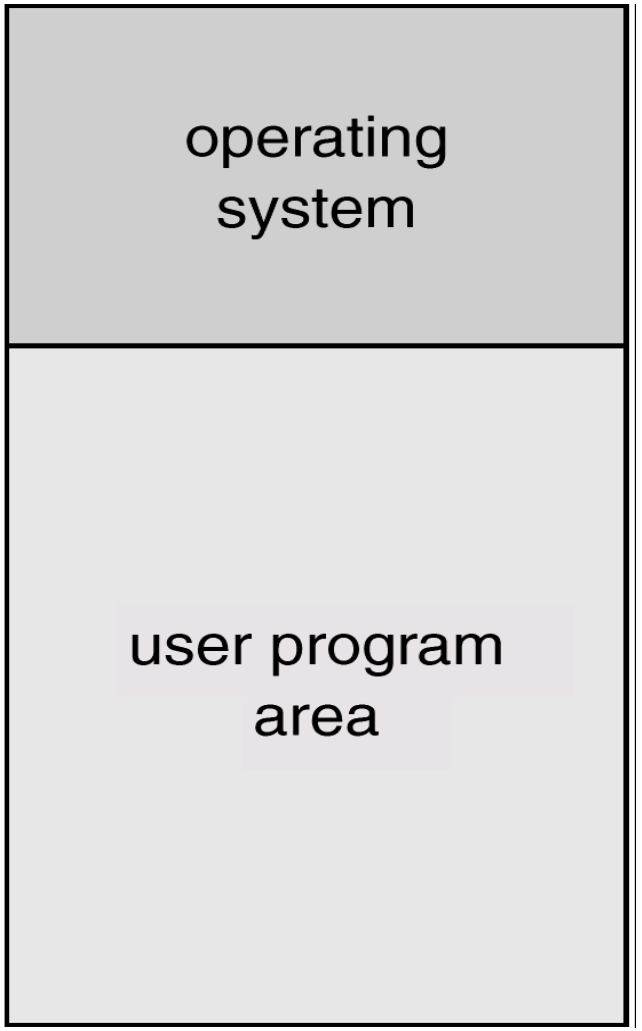
- Can adopt technology developed for larger operating systems—multi-process, multi-user
- Individuals usually have sole use of computer and do not need advanced protection features.
- May run several different types of operating systems (Windows, MacOS, UNIX, Linux)

Batch Systems

- First rudimentary system.
- User ≠ operator
- Reduce setup time by batching similar jobs
- Automatic job sequencing – automatically transfers control from one job to another.
- Resident monitor :
  - initial control in monitor
  - control transfers to job
  - when job completes transfers back to monitor

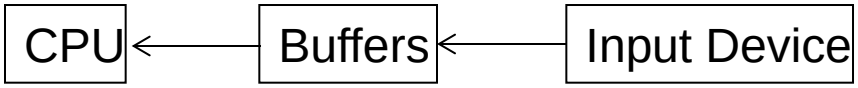


Memory Layout

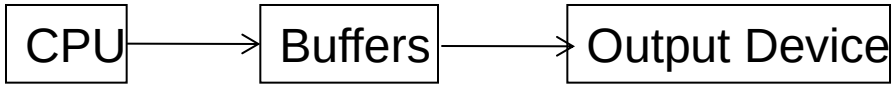


Buffering VS Spooling

- **Buffering** is a method of overlapping I/O and processing of a single job
  - Input



  - Out put

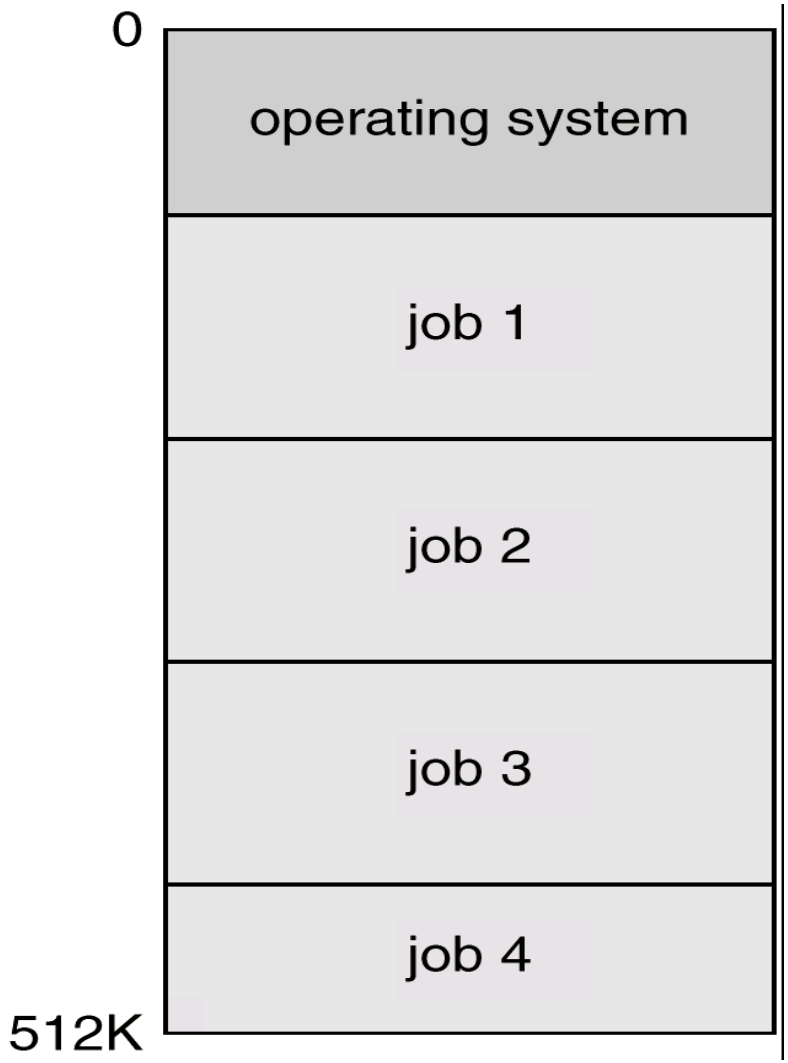


- Since the CPU is faster than the I/O device, the speed of execution is controlled by the I/O device and not by the speed of CPU

- **Simultaneous Peripheral Operations Online** allows CPU to overlap the input of one job with the computation and output of other jobs
- It essentially use the disk as a large buffer for reading and for storing output files
- Advantages
  - Speedy computation at the cost of some disk space and few tables kept by OS
  - Keeps both CPU and I/O devices working at much higher rates

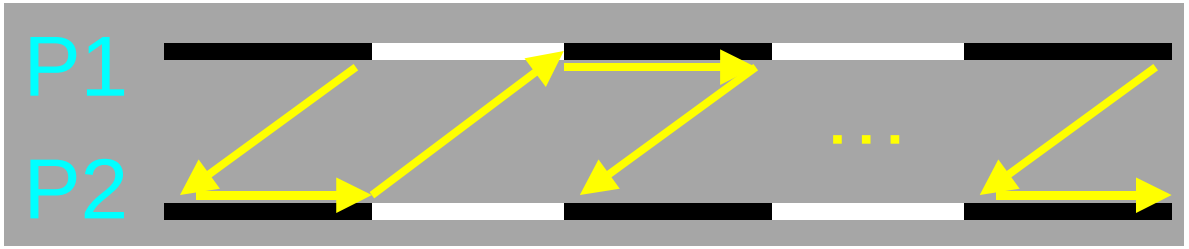
Multiprogrammed Systems

Several jobs are kept in main memory at the same time, and the CPU is multiplexed among them.



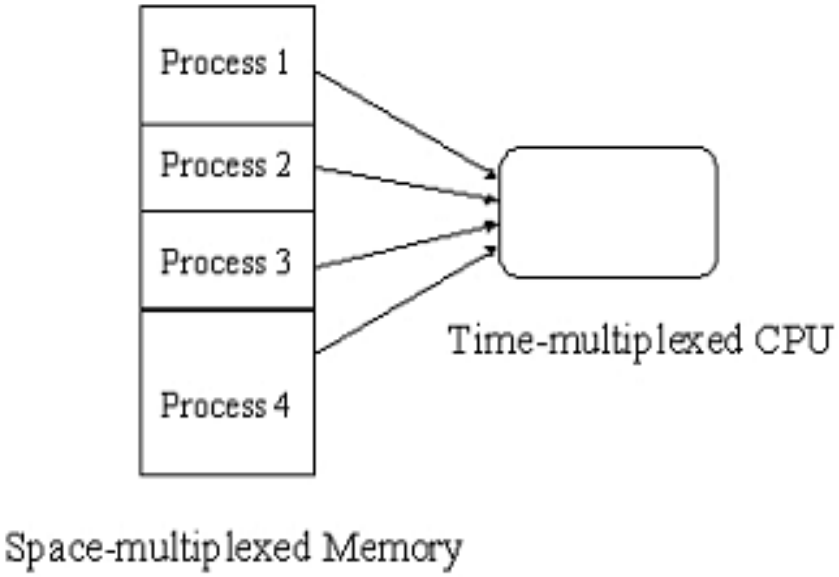
Multiprogrammed Systems

Example: Two processes P1 and P2 with CPU and I/O bursts of one time unit each



OS Features Needed for Multiprogramming

- SPOOLing (Simultaneous Peripheral Operation On-Line)
- Memory management
- CPU scheduling



Time-sharing Systems

- An interactive system with multiprogramming
- A job is swapped in and out of memory to the disk if needed.
- On-line file system must be available for users to access data and code.

Real-time Systems

- Well-defined fixed-time constraints.
- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.
- Real-Time systems may be either *hard* or *soft* real-time.

Real-time Systems …

- Hard real-time systems:
  - Secondary storage limited or absent, data stored in short term memory, or read-only memory (ROM)
  - No virtual memory—time cannot be “wasted” on translation of logical to physical addresses
  - OS code structured for efficiency
  - Plane landing systems, process control in nuclear power plants, respirators, etc.

Real-time Systems ...

- Soft real-time systems
  - Output should be produced within the given time constraints but if it is not, the result is not life threatening
  - Useful in applications (multimedia, virtual reality) requiring advanced operating-system features.

Hardware Protection

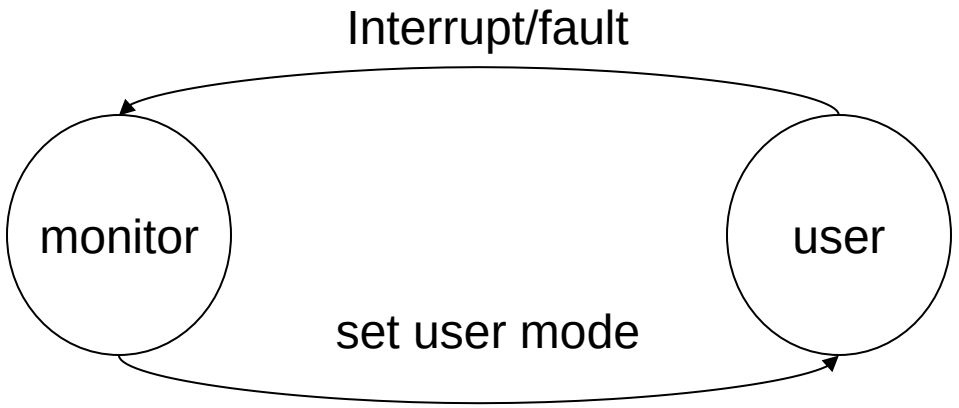
- Dual-Mode Operation
- I/O Protection
- Memory Protection
- CPU Protection

Dual-Mode Operation

- Sharing system resources requires operating system to ensure that an incorrect program cannot cause other programs to execute incorrectly.
- Provide hardware support to differentiate between at least two modes of operations.
  - *User mode* – execution done on behalf of a user.
  - *Monitor mode* (also *kernel mode* or *system mode*) – execution done on behalf of operating system.

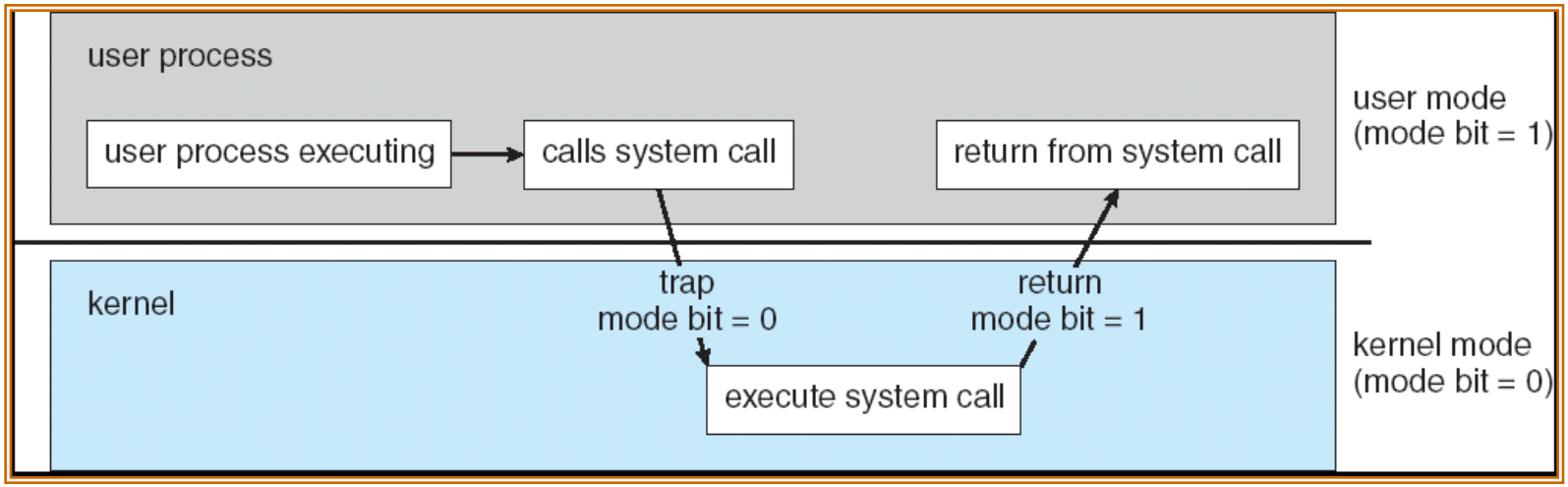
Dual-Mode Operation …

- *Mode bit* added to computer hardware to indicate the current mode: monitor (0) or user (1).
- When an interrupt or fault occurs hardware switches to monitor mode.



***Privileged instructions* can be issued only in monitor mode.**

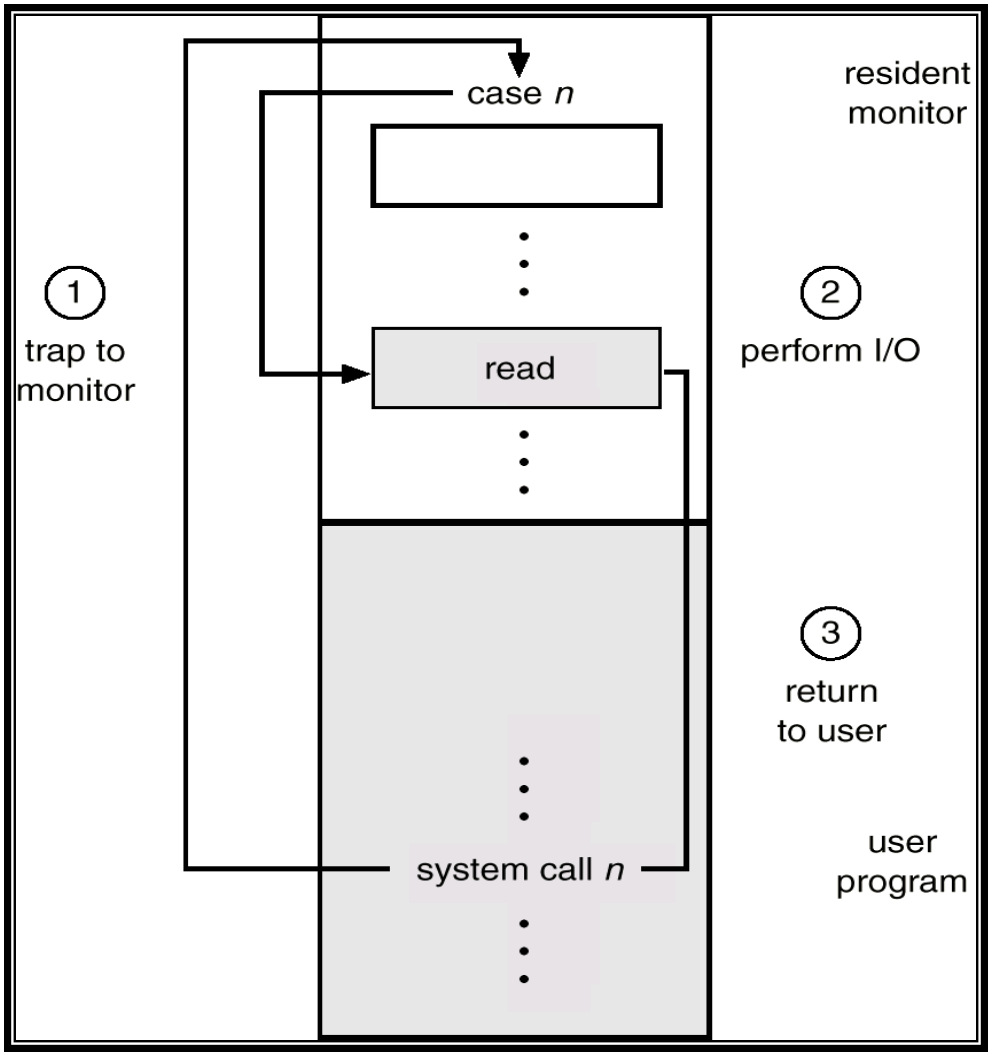
Dual-Mode Operation…



I/O Protection

- All I/O instructions are privileged instructions.
- Must ensure that a user program could never gain control of the computer in monitor mode (i.e., a user program that, as part of its execution, stores a new address in the interrupt vector).

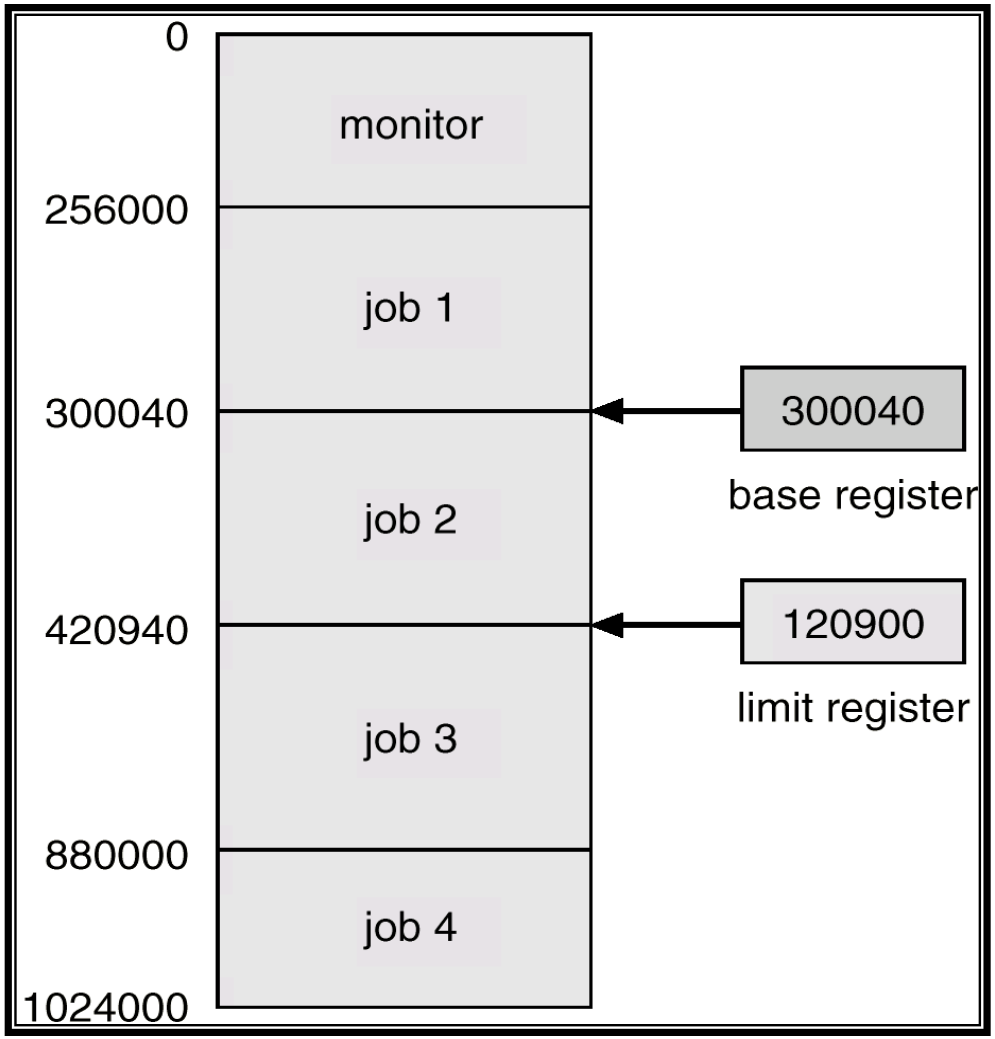
Use of A System Call to Perform I/O



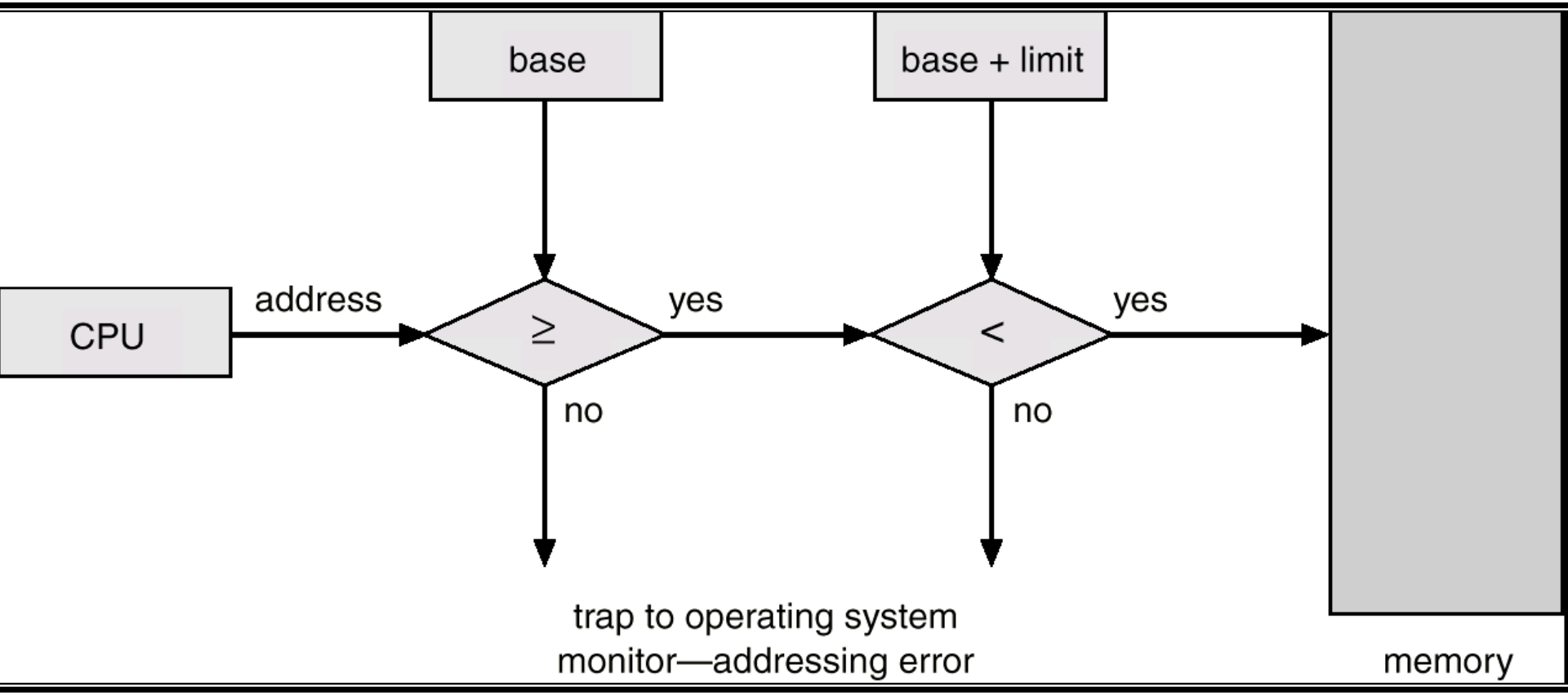
Memory Protection

- Must provide memory protection outside the address space of a process.
- In order to have memory protection, add two registers that determine the range of legal addresses a program may access:
  - **Base register** – holds the smallest legal physical memory address.
  - **Limit register** – contains the size of the range
- Memory outside the defined range is protected.

Use of Base and Limit Register



Hardware Support



CPU Protection

- *Timer* – interrupts computer after specified period to ensure operating system maintains control.
  - Timer is decremented every clock tick.
  - When timer reaches the value 0, an interrupt occurs.
- Timer commonly used to implement time sharing.
- Time also used to compute the current time.
- Load-timer is a privileged instruction.

OS Components

- Process management
- Main memory management
- Secondary storage management
- I/O system management
- File management
- Protection system
- Networking
- Command-line interpreter (shells)

Operating System Services

Services for user and users of programs:

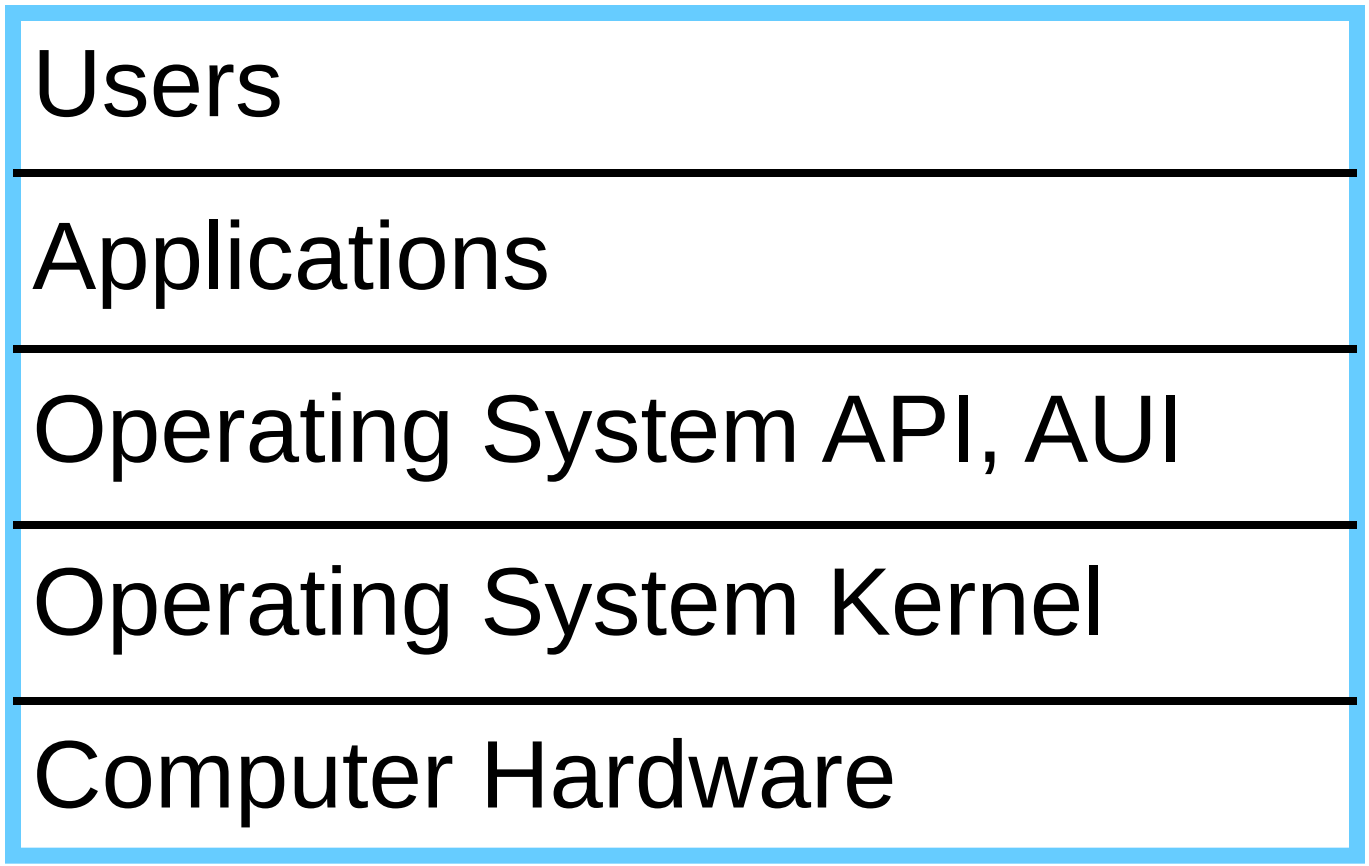
- **Program execution**
- **I/O Operations**
- **File System Manipulation**
- **Communications** between processes/users
- **Error detection and handling**

Operating System Services ...

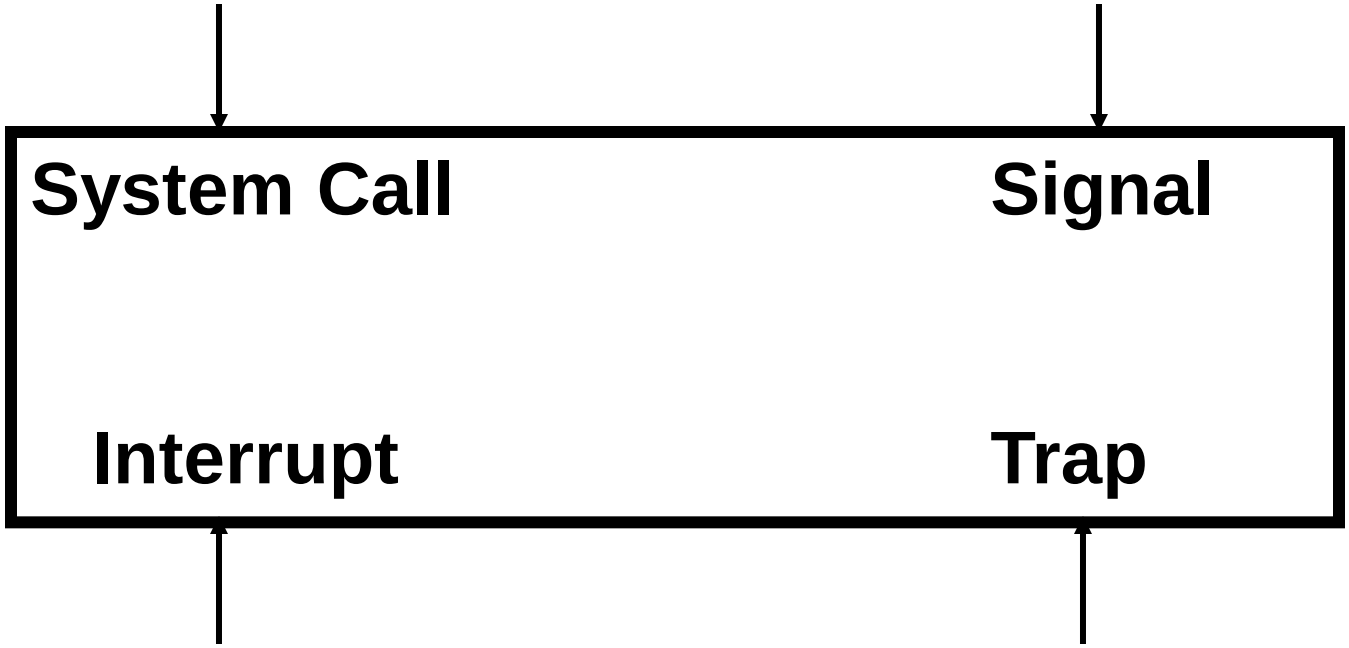
Services for efficient system operation:

- **Resource management**
- **Accounting**
- **Protection**

OS Kernel


Users
Applications
Operating System API, AUI
Operating System Kernel
Computer Hardware

Entry Points into Kernel



System Calls

- User processes must not be given open access to the kernel code
- The system call interface layer contains entry point in the kernel code
- Any user or application request that involves access to any system resource must be handled by the kernel code

Types Of System Calls

- Process Control
  - Load, execute
  - Create process, terminate process
  - Get/Set process attributes
  - Wait for time, wait event, signal event
  - Allocate, free memory
- File Management
  - Create, delete
  - Open, close
  - Read, write
  - Get/Set file attributes

Types Of System Calls

- Device Management
  - Request device, release device
  - Read, write, reposition
  - Get/Set device attributes
  - Logically attach or detach devices
- Information Maintenance
  - Get/Set date, time, or system data
  - Get/set process, file or device attributes
- Communication
  - Create/Delete communication connection
  - Send/Receive message
  - Attach/detach remote devices

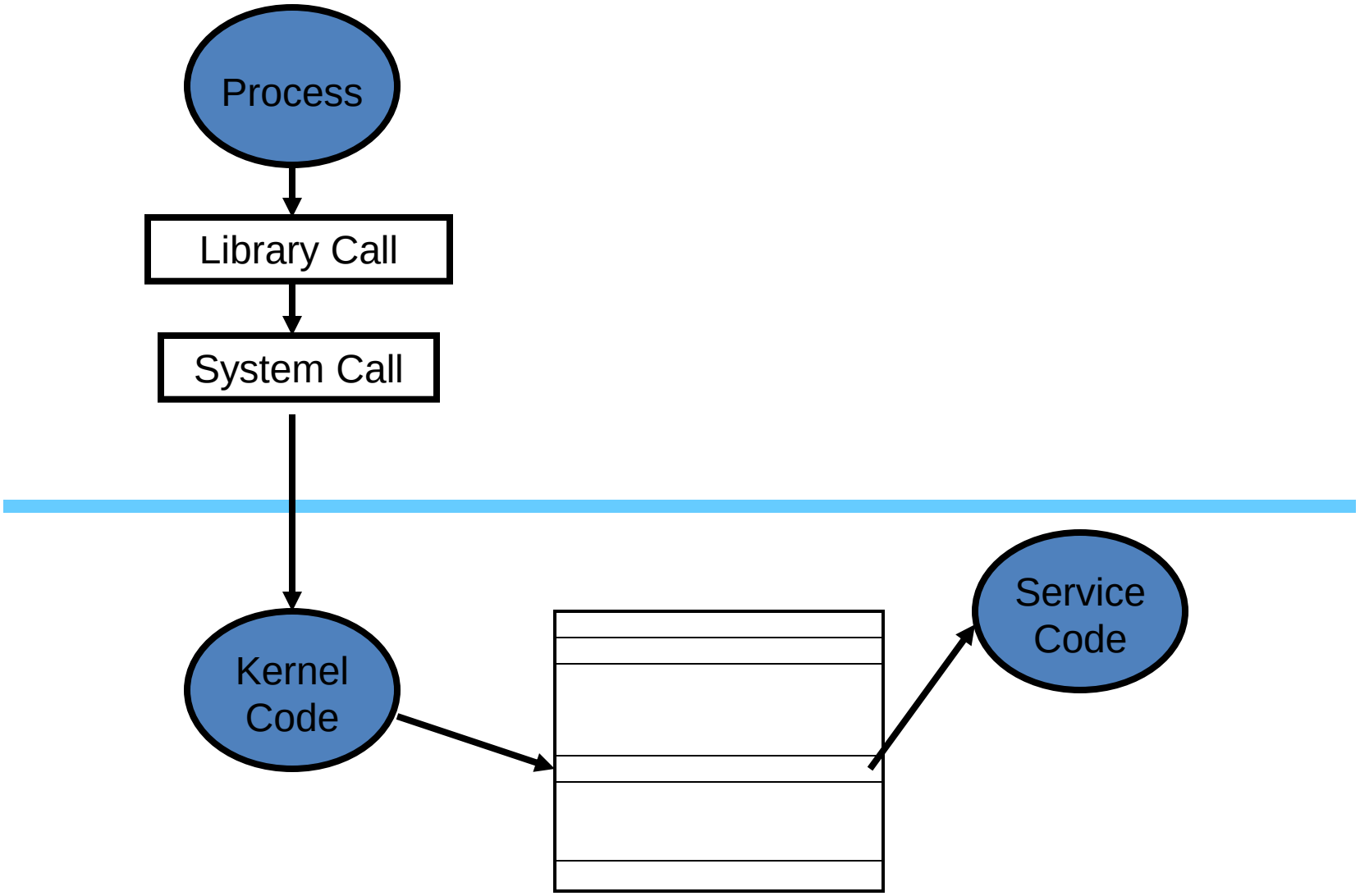
System Call Execution

- The user program makes a call to a library function.
- Library routine puts appropriate parameters at a well-known place (registers, stack, or a table in memory).
- The `trap` instruction is executed to change mode from user to kernel.
- Control goes to operating system.
- Operating system determines which system call is to be carried out.

Semantics of System Call Execution ...

- Kernel indexes the **dispatch table**, which contains pointers to service routines for system calls.
- Service routine is executed and return parameter or error code placed at well-known places (usually a CPU register).
- Control given back to user program.
- Library function executes the instruction following `trap`.

System Call ...

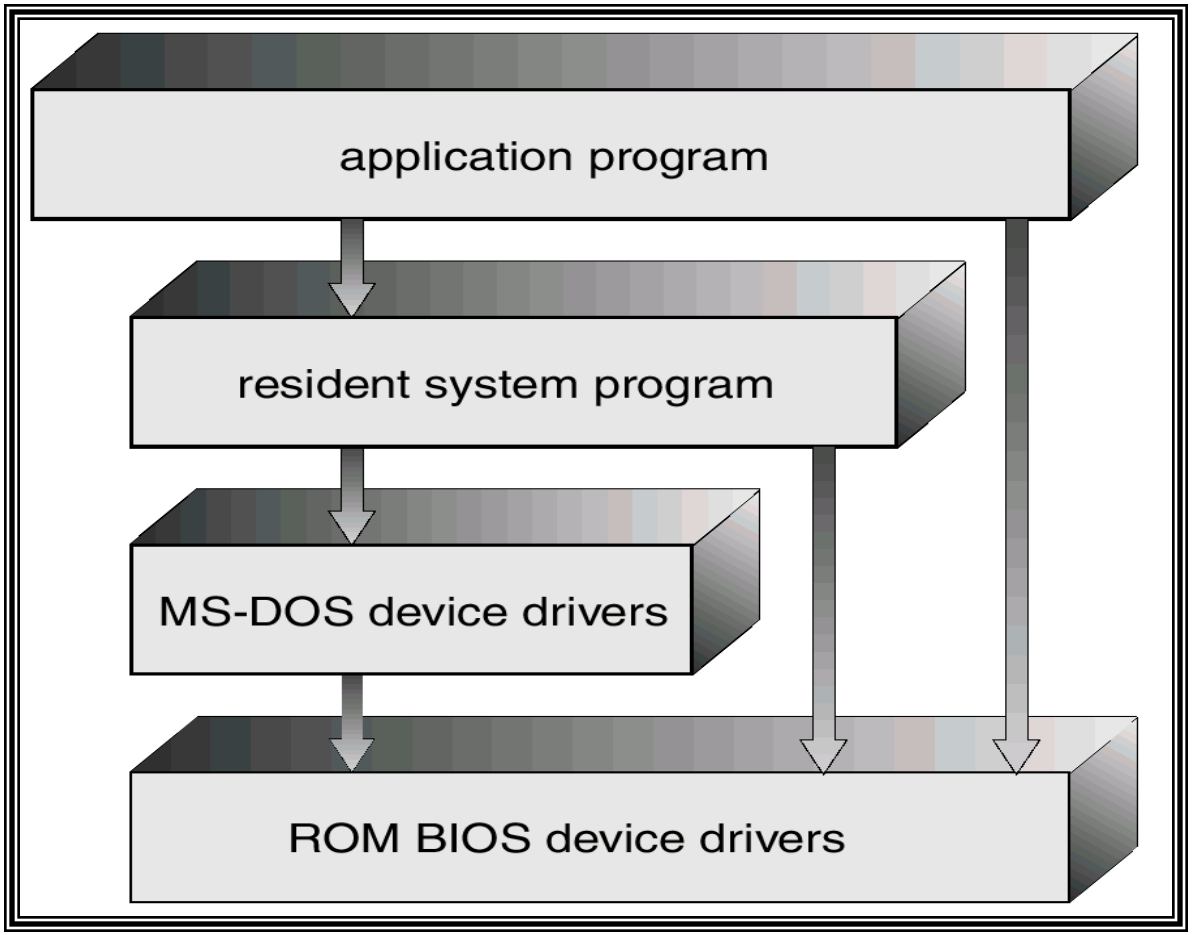


Reading Assignment

(Article 2.7, Page 70)

- Operating System Structures
  - Simple Structure
  - Layered Approach
  - Microkernels
  - Module based Structure

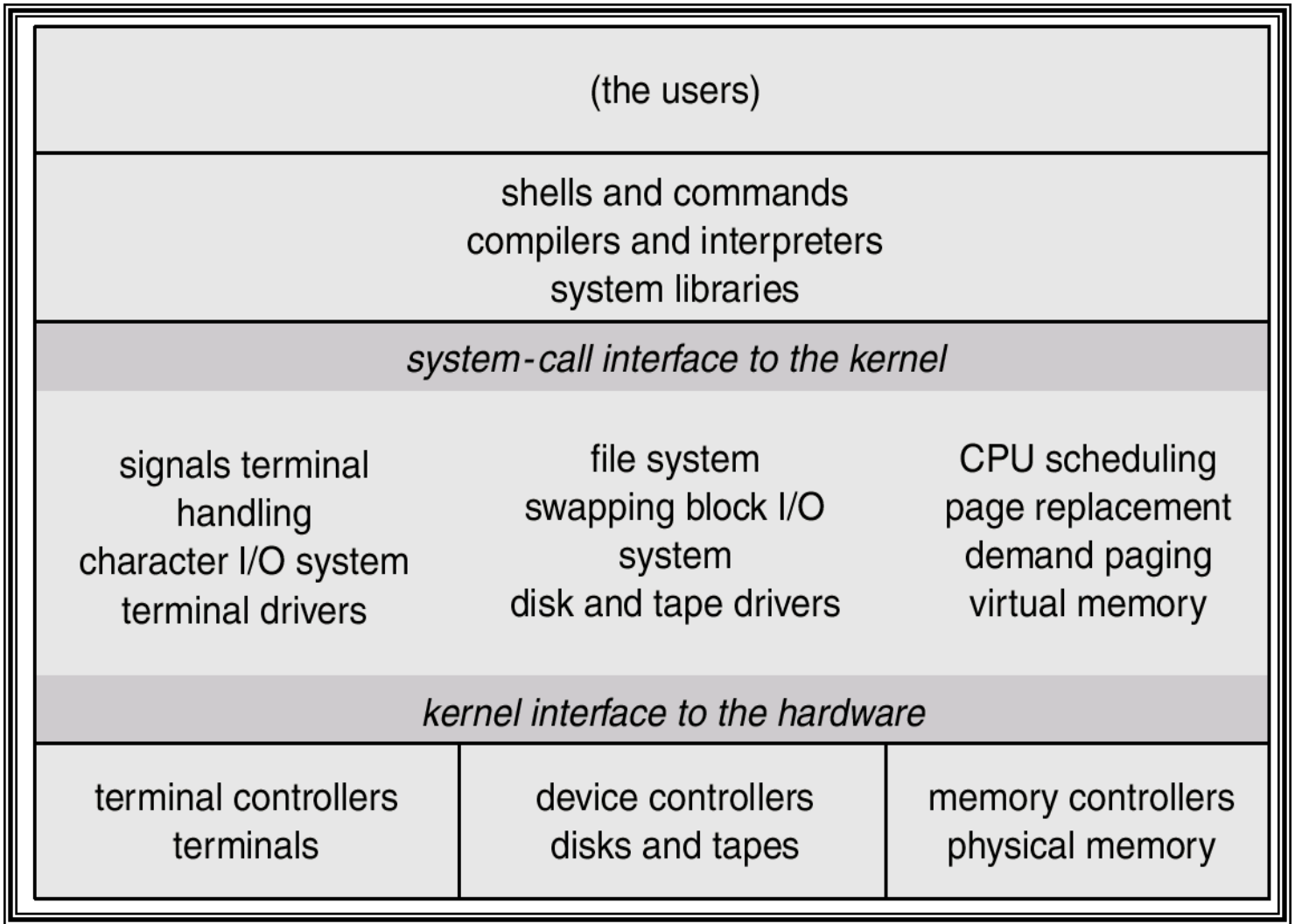
Simple Structures



Simple Structures …

- **UNIX** consists of two separable parts, the kernel and the system programs.
- Every thing below the system call interface and above the physical hardware is the kernel.
- An enormous amount of functionality combined in one level, UNIX is difficult to enhance as changes in one section could adversely affect other areas.

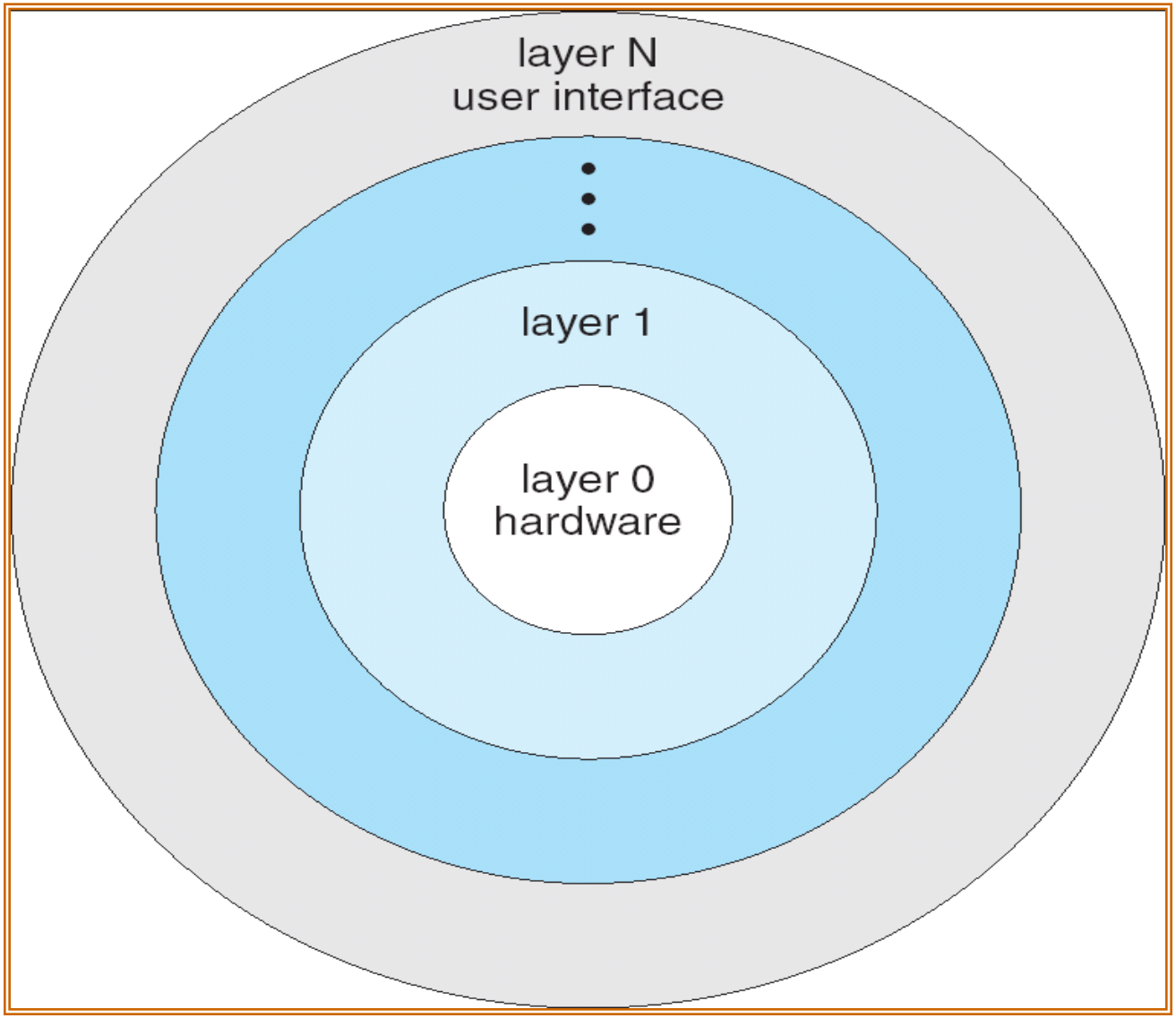
UNIX System Structure



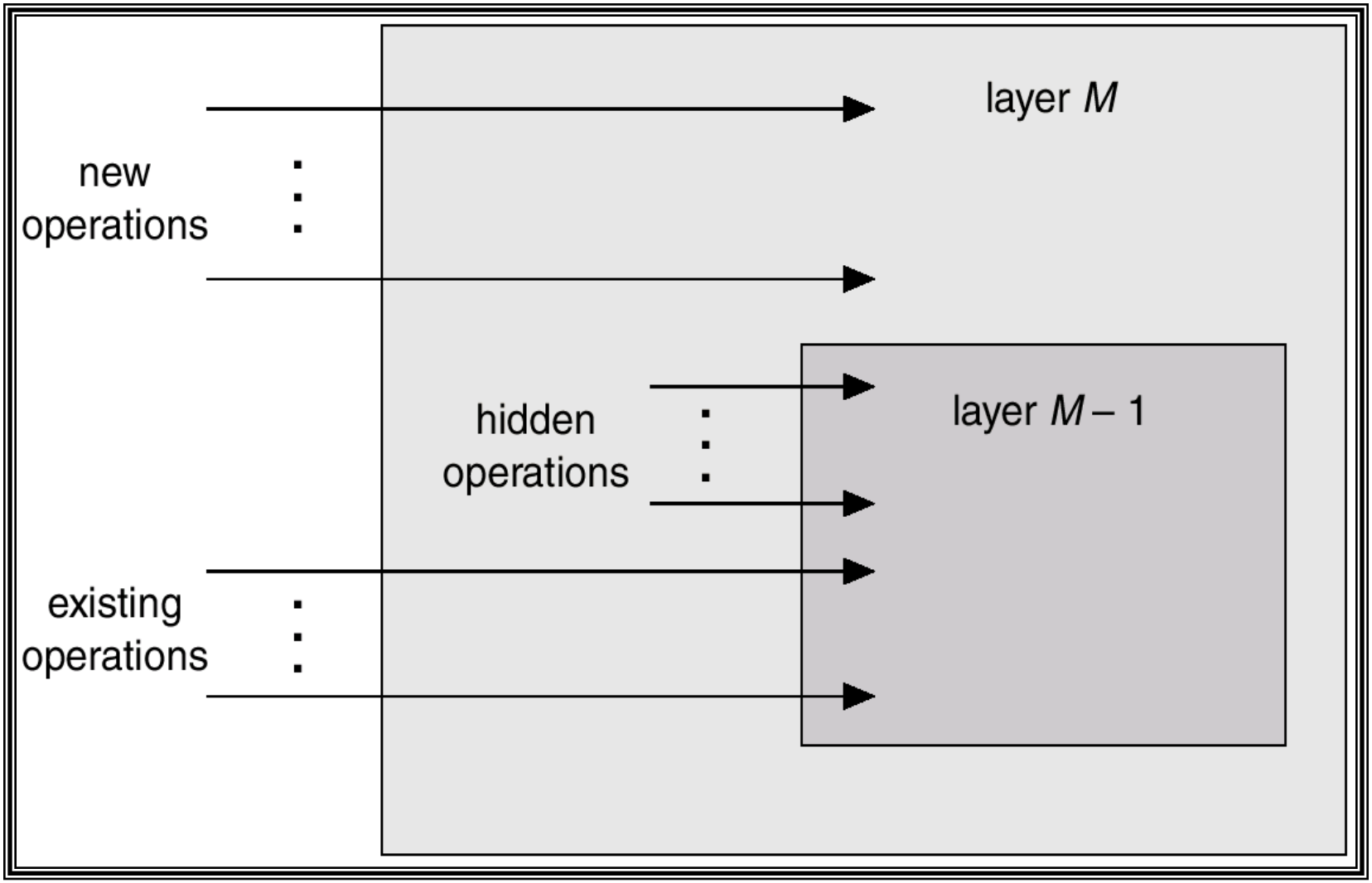
Layered Approach

- The OS is broken up into a number of layers
- Bottom layer is hardware and the topmost layer (layer N) is the user interface
- A typical layer consists of data structures and a set of routines to service the layer above it
- THE operating system by Dijkstra
- IBM’s OS/2

Layered Approach ...



Layered Approach ...



Layered Approach …

- Modularity
- Each layer uses functions and services of only lower layers
- Simplifies debugging and system verification.
- The major difficulty with layered approach is careful definition of layers, because a layer can only use the layers below it
- Less efficient than other approaches

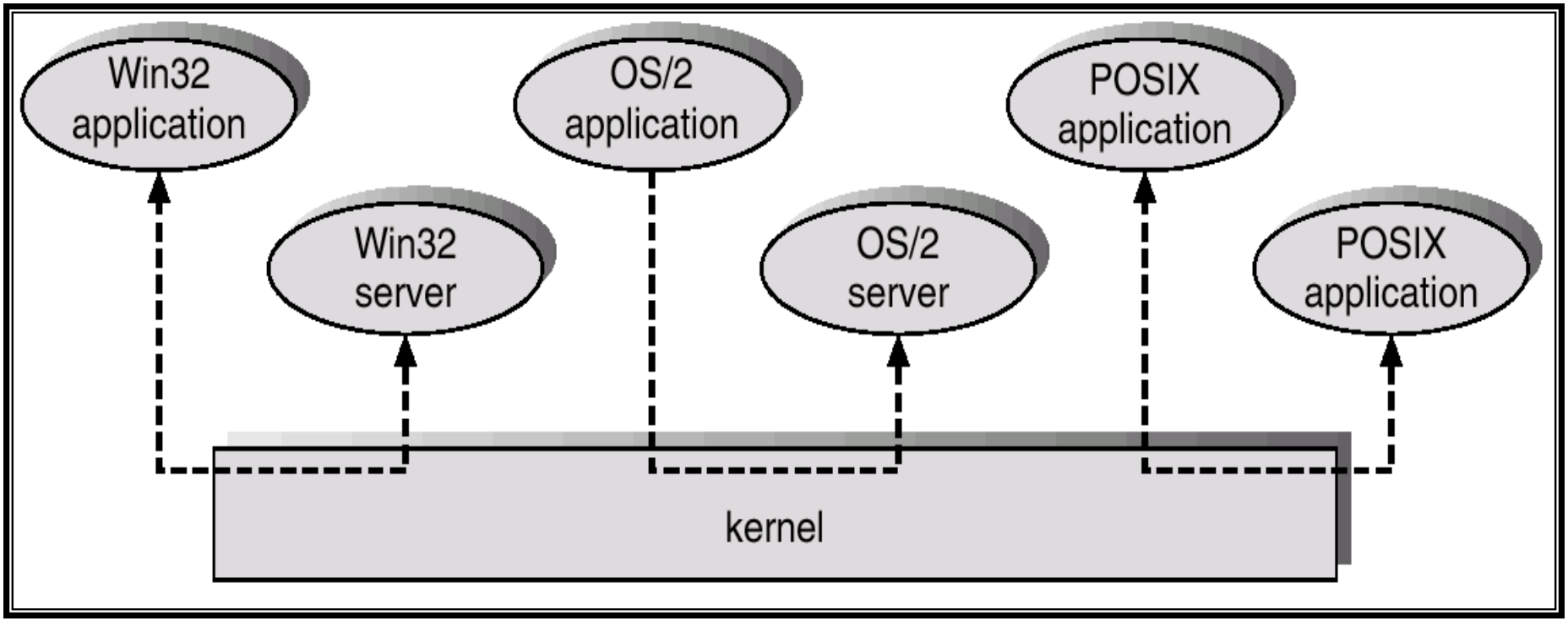
Microkernel

- Structures the operating system by removing all non-essential components from the kernel and implementing them as system and user level programs
- Smaller kernel
- Main function is to provide a communication facility between client programs and the various services that are also running in the user space.

Microkernel ...

- Easier to extend the OS—new services are added to user space and consequently do not require modification of the kernel and/or its recompilation
- Easier to maintain operating system code (enhancement, debugging, etc.)
- OS is easier to port from one hardware to another
- More security and reliability
- Mach, MacOS X Server, QNX, OS/2, and Windows NT

Windows NT Client-Server Structure



Modules Based Structure

- Most modern operating systems implement modules:
  - Uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel
- Overall, similar to layers but with more flexibility
- Solaris OS