Part 1: Introduction

- **What is an Operating System (OS)?**

- Types of Computing Systems

- Computer System Architecture (Review)

- Operating System Services
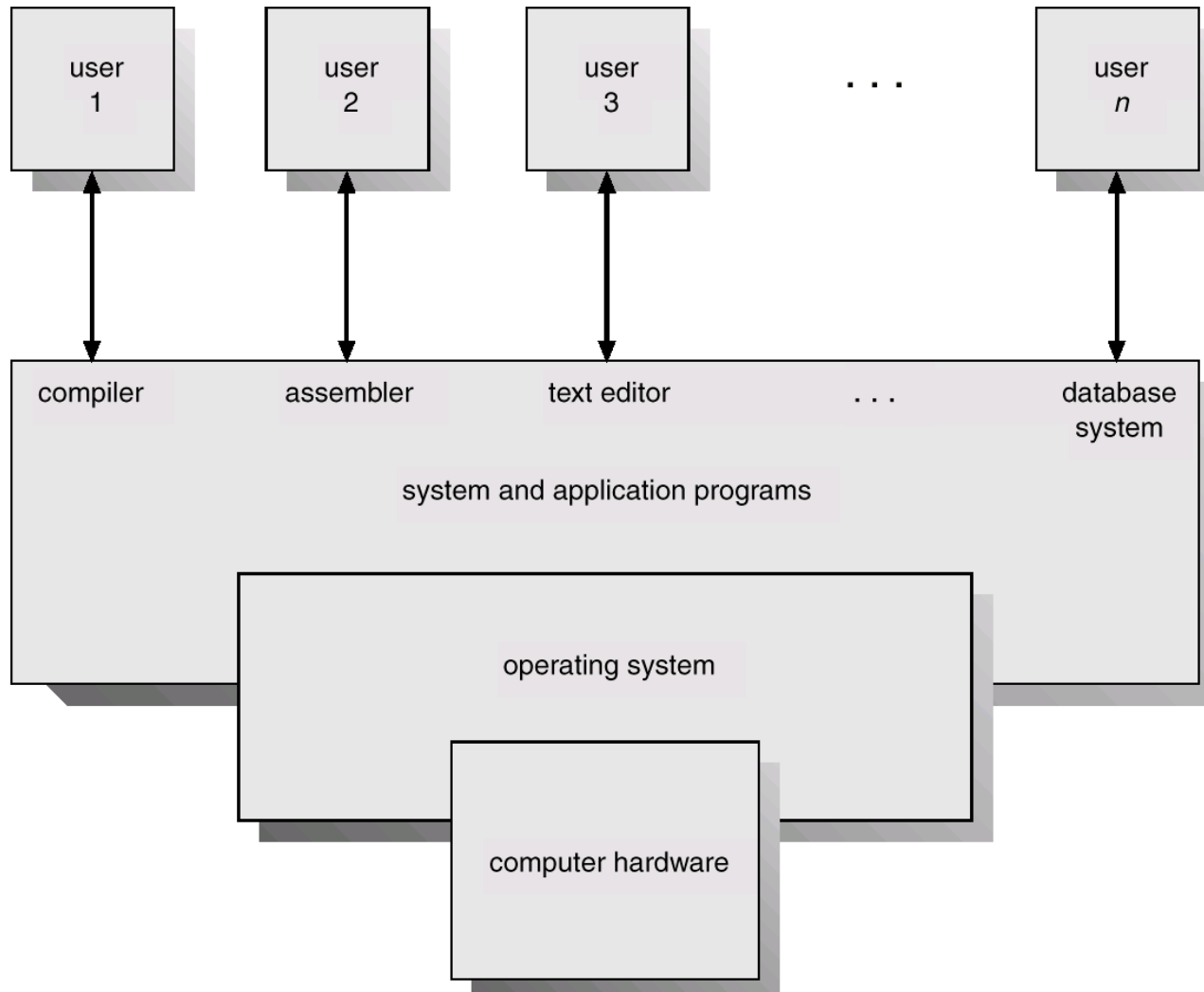
What is an Operating System?

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

- Two major goals: user convenience and efficient hardware utilization

  – Hide hardware complexity

  – Use hardware in an efficient manner

  ∗ Smart resource (e.g., Central Processing Unit (CPU), I/O devices, memory) allocation

- These two goals can be contradictory

  – Smart resource allocation may require lot of information about user programs

Computer System Components

1. **Hardware:** Provides basic computing resources (CPU, memory, I/O devices)

2. **Operating System(OS):** Controls and coordinates the use of the hardware among <u>various</u> application programs for <u>various</u> users

3. **Application programs:** Defines the ways in which system resources are used to solve computing problems for users (compilers, database systems, video games, business programs)

4. **Users:** People, machines, other computers

Abstract View of System Components

user 1   user 2   user 3   . . .   user *n*

compiler   assembler   text editor   . . .   database system

system and application programs

operating system

computer hardware

Operating System Definitions

- Resource allocator: Manages and allocates hardware resources

- Control program: Controls the execution of user programs and operations of I/O devices

- **Kernel:** The one "core" program that is always ready to accept new commands from the users as well as hardware (all else being application programs)

1.5

Part 1:   Introduction

- What is an Operating System (OS)?

- **Types of Computing Systems**

- Computer System Architecture (Review)

- Operating System Services

Types of Computing Systems

- Batch systems

- Multiprogrammed and Time-sharing systems
  - Desktop systems

- Embedded and Cyber-physical systems
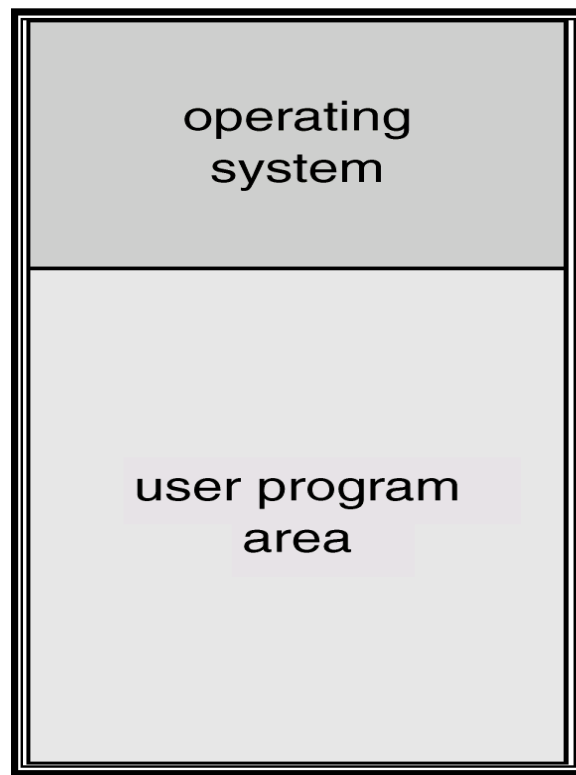 - Real-Time systems
  - Handheld systems

A Short Review on Computers

In this video, we will briefly review the history of computers
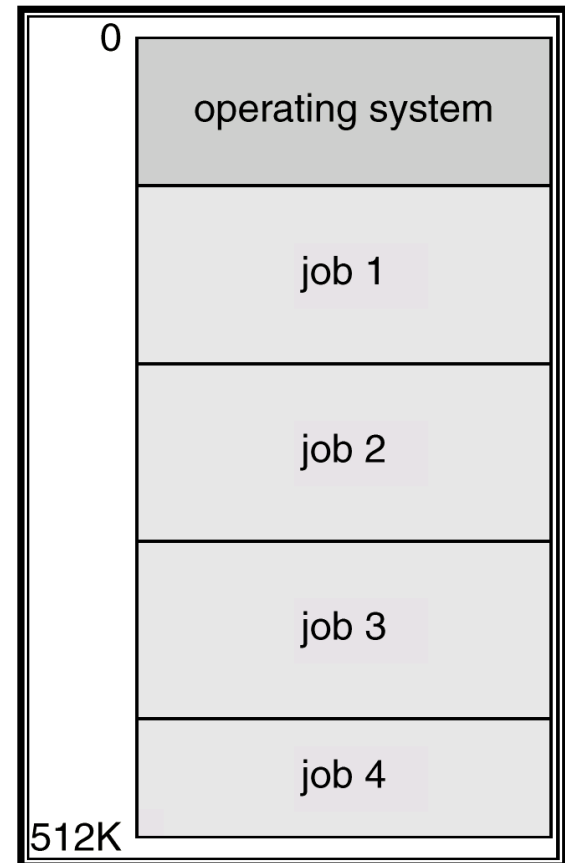
1.8

Simple Batch Systems

- Reduce setup time by <u>batching similar jobs</u>

- **Automatic job sequencing** – automatically transfers control from one job to another

- Simple memory layout
  - Only one user job in memory at any time point

- Not very efficient
  - When job waits for I/O, CPU idles

operating system
user program area

Multiprogrammed (Time-Sharing) Systems

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

- A job is swapped in and out of memory to the hard disk

- System is highly interactive; supports multiple online users

- Examples: desktops, servers

```
0
operating system

job 1

job 2

job 3

job 4
512K
```

1.10

OS Features Needed for Multiprogramming

- Memory management: To allocate memory to several jobs

- CPU scheduling: To choose among several jobs ready to run

- I/O device scheduling: Allocation of I/O devices to jobs

Desktop Systems

- Personal computers – computer system dedicated to a single user

- Several I/O devices – keyboard, mouse, display screens, printers, etc.

- User convenience and responsiveness is the main focus

- May run several different types of operating systems (Windows, MacOS, UNIX, Linux)

Embedded and Cyber-physical Systems

- **What are they?** Physical systems whose operations are monitored and controlled by a reliable computing and communication core

- **Resource constrained:** Low power, small memory, low bandwidth, etc.

- **Domain-specific OSes:** Real-time, Handheld, Automotive, Avionics, Industrial Controls, Sensor networks, etc.

Real-Time Systems

- Used as a control device in a dedicated application such as industrial controls, automotive, avionics, medical devices, etc.

- Well-defined fixed-time constraints

  – Job must be completed within a deadline

  – Example: Airbag control in cars
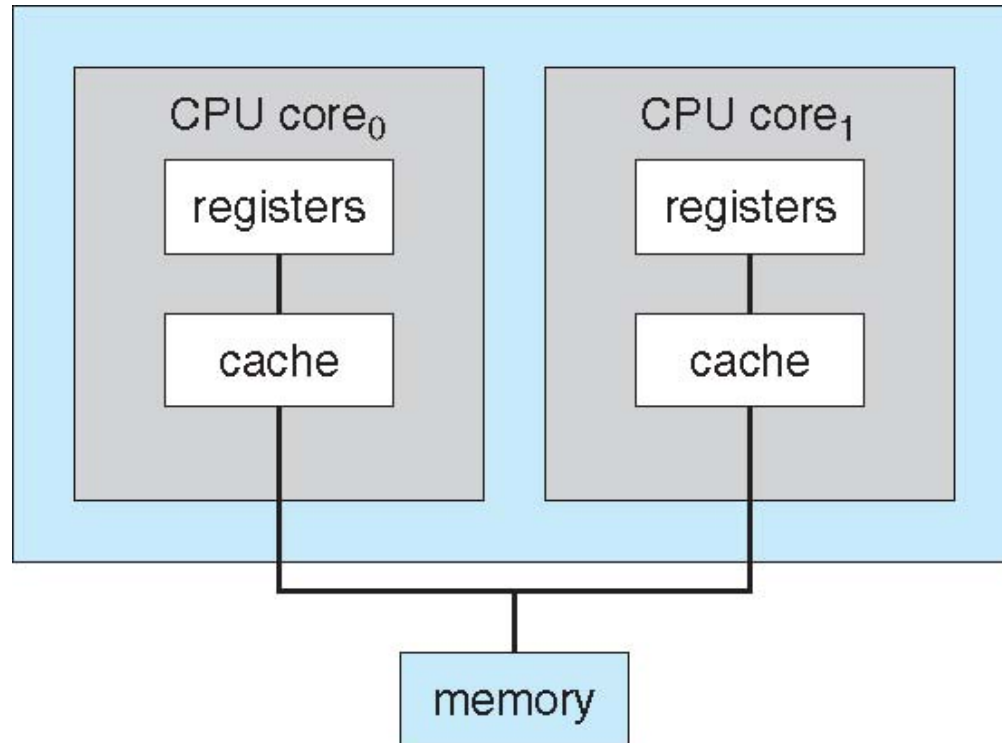
- Example real-time OSes

Handheld Systems

- Mobile phones, tablets

- Issues:
  - Limited memory
  - Slow processors
  - Small display screens

- Popular OSes: iOS, Android, Windows Phone

1.16

A Dual-Core CPU Design

Multiprocessor Systems

- Systems with more than one CPU, or CPU with multiple cores

- *Tightly coupled systems:* Communication usually takes place through shared memory

- Advantages of such parallel systems
  - Increased system *throughput*
  - Economical *due to* sharing of memory and I/O devices (as compared to multiple single CPU systems)
  - Increased reliability due to redundancy

Part 1: Introduction

- What is an Operating System (OS)?

- Types of Computing Systems

- **Computer System Architecture (Review)**
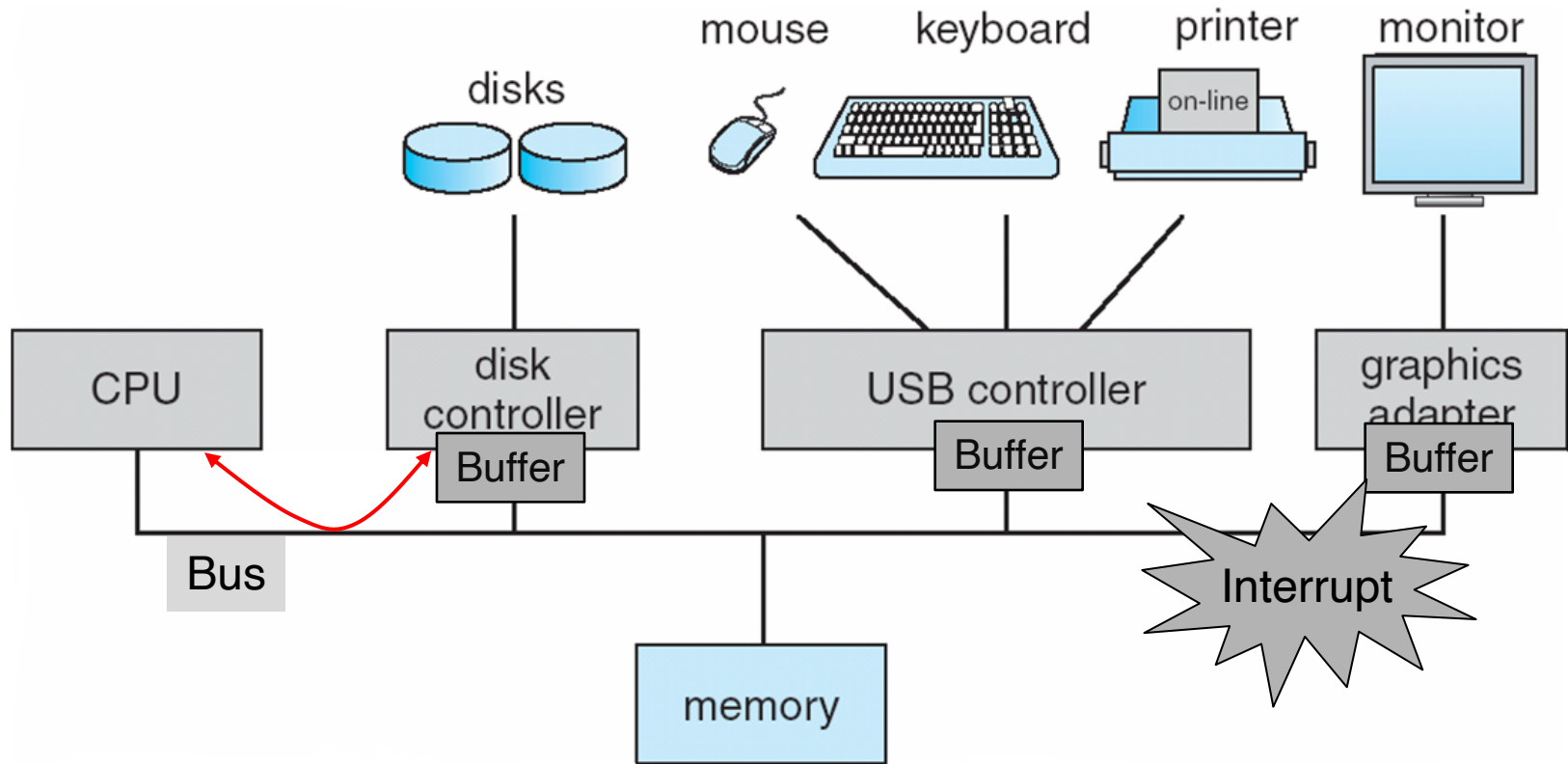
- Operating System Services

Computer System Architecture

- Computer System Operation

- Storage Hierarchy

- Hardware Protection

1.20

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

- Device controller moves data between local buffer and memory

- Device controller informs CPU that it has finished its operation by causing an *interrupt*

Computer-System Operation

disks   mouse   keyboard   printer   monitor

on-line

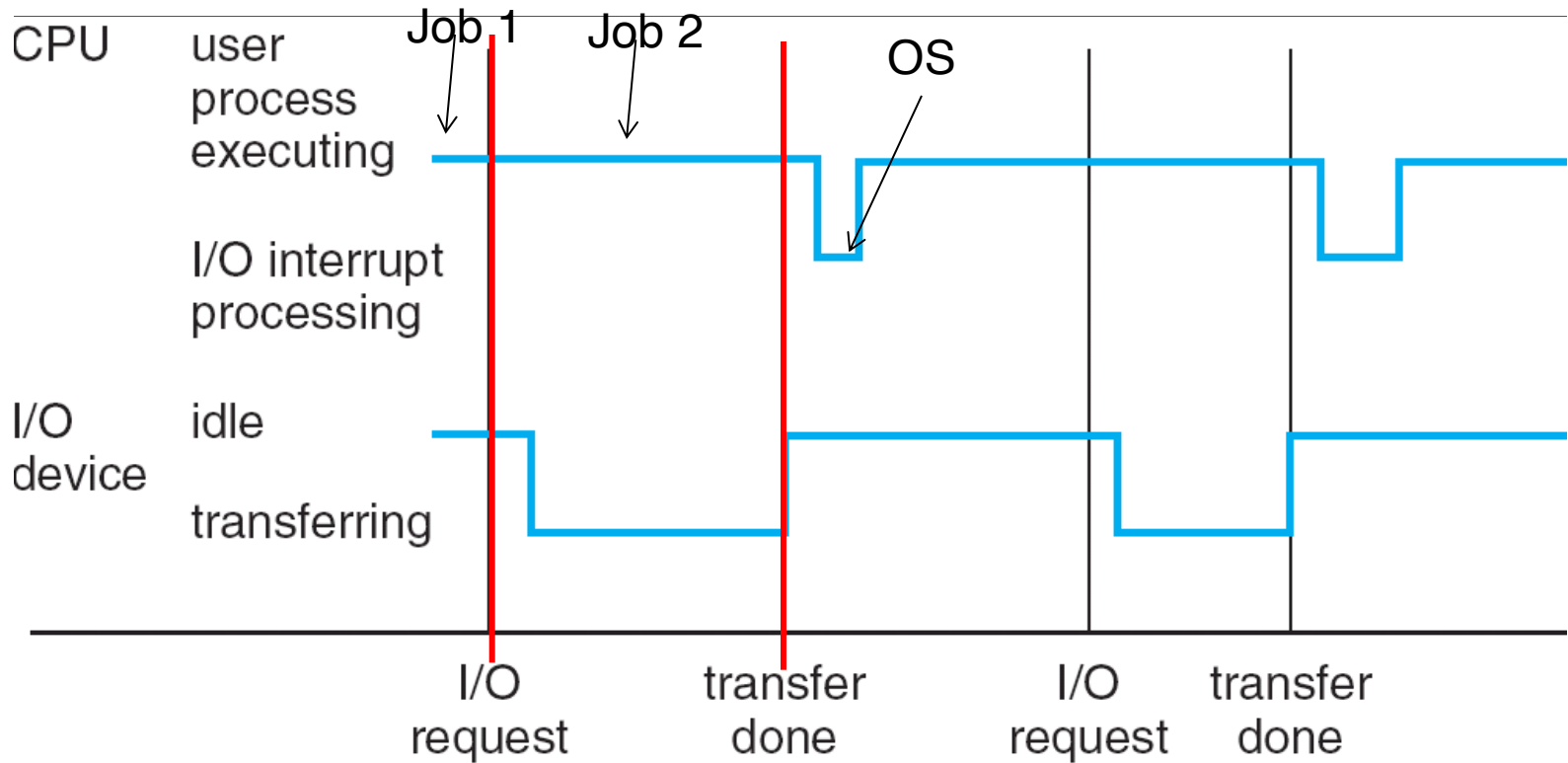| CPU	disk controller	USB controller	graphics adapter
	Buffer	Buffer	Buffer

Bus

Interrupt

memory

Common Functions of Interrupts

- An interrupt transfers control to the interrupt service routine generally through the *interrupt vector*, which contains the addresses of all the service routines

- Incoming interrupts are *disabled* while another interrupt is being processed to prevent any *loss of interrupts*

- A *trap* is a CPU generated interrupt caused either by a software error or request
  - Unhandled exceptions in user program

- An operating system is typically *interrupt* driven
  - If the OS is not interrupt driven, it would be required to poll for task/event completions

Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter

  - Also called a *context switch*

- It then determines which type of interrupt has occurred

  - Separate segments of code determine what action should be taken for each type of interrupt

- Based on the interrupt type, it identifies the appropriate *interrupt service routine* to execute
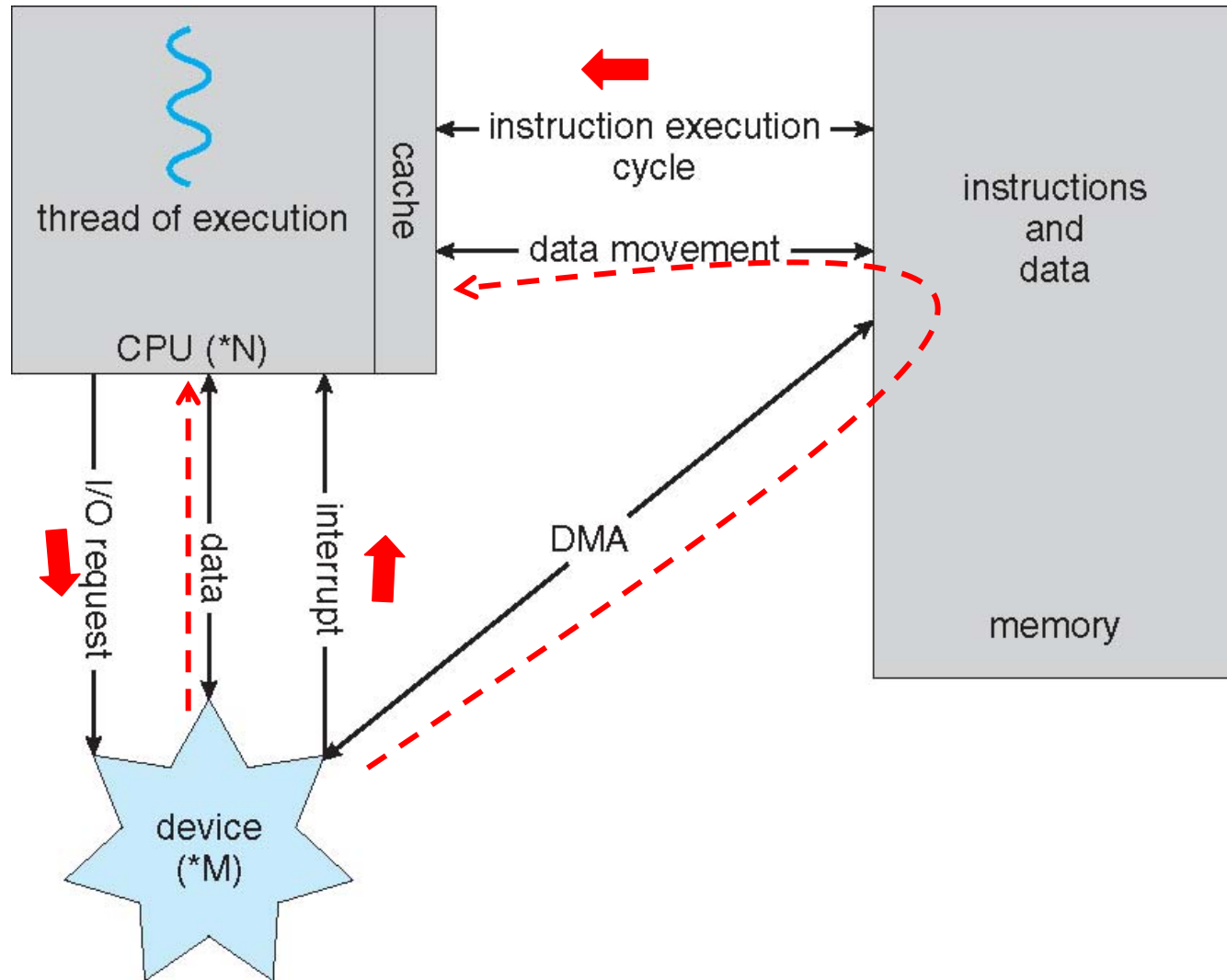
  - Obtained from the *interrupt vector table*

Interrupt-Driven I/O Timeline

1.25

Direct Memory Access (DMA)

- Used for high-speed I/O devices that are able to transmit information at close to memory speeds

- OS sets up the memory blocks, counters, etc.

- Device controller transfers data blocks from buffer to main memory without CPU intervention

- Only one interrupt is generated per block, rather than one interrupt per byte

How a Modern Computer Works

Storage Hierarchy

- Memory Hierarchy: CPU registers, CPU caches, main memory, hard disk …

- Storage system organization based on
    - speed
    - cost
    - volatility
    - size

    This slide is not examinable.

- *Caching:* Copying information into faster storage system; main memory can be viewed as the last *cache* before secondary storage

Storage Hierarchy

Size increases

Speed decreases

Cost/price decreases

This slide is not examinable.

1.29

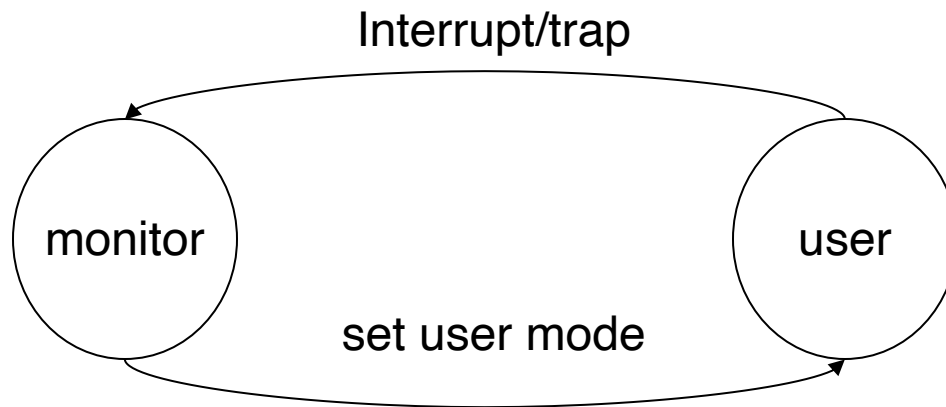# Hardware Protection

- Dual-Mode Operation

- I/O Protection

- Memory Protection

Dual-Mode Operation

- Provides hardware protection by differentiating between at least two modes of operations

  1. *User mode:* Execution of user processes

  2. *Monitor mode* (*supervisor mode* or *system mode or kernel mode*): Execution of operating system processes

Dual-Mode Operation (Cont.)

- *Mode bit* added to computer hardware to indicate the current mode: monitor (0) or user (1)

- When an interrupt or trap occurs hardware switches to monitor mode

Interrupt/trap

monitor   user

set user mode

- *Privileged instructions* can be used only in monitor mode

Kernel mode vs. root/admin.

- Are they the same?
  - **No, they are not the same**

- Kernel or user mode is a hardware operation mode

- Root/Administrator is a user account in an OS
  - Jobs still execute in user mode, even when executed by a root/admin. user
  - This user may execute code in kernel mode indirectly → e.g., by loading a kernel module

1.33
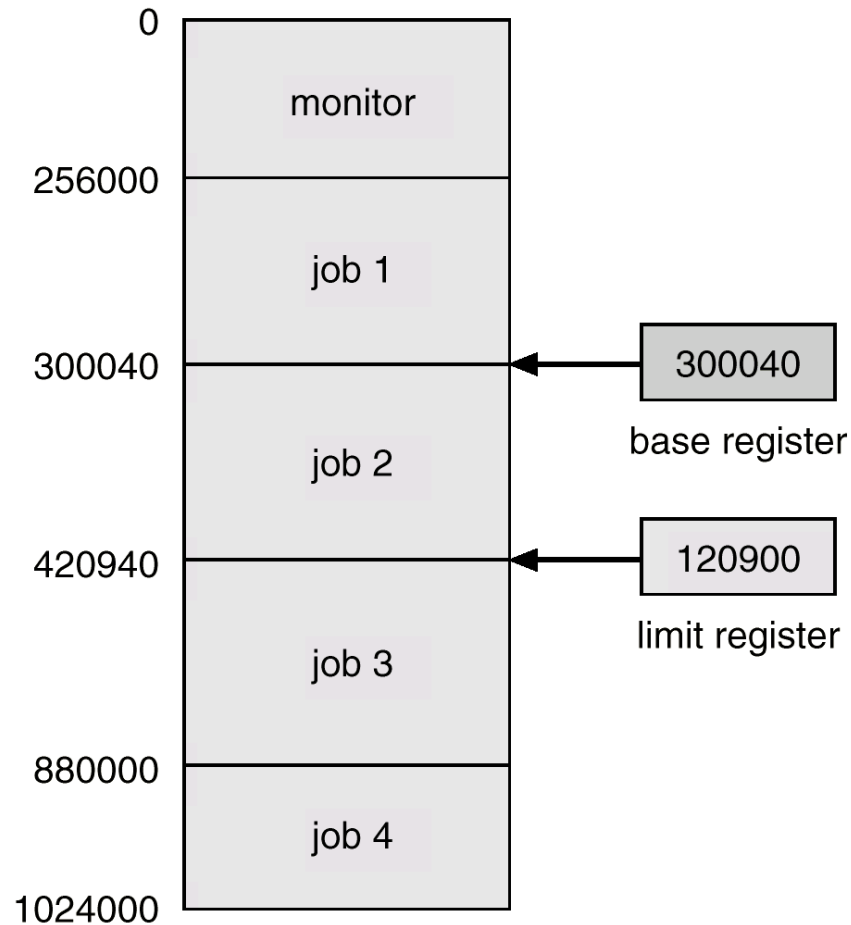
I/O Protection

- A user program may issue illegal I/O operation; hence I/O must be protected
    - Case 1: read a file that does not exist
    - Case 2: unauthorized access to a device

- All I/O instructions are privileged instructions

- All I/O operations must go through the OS to ensure its correctness and legality
    - CPU generates a trap for I/O operations that try to bypass the OS

Memory Protection
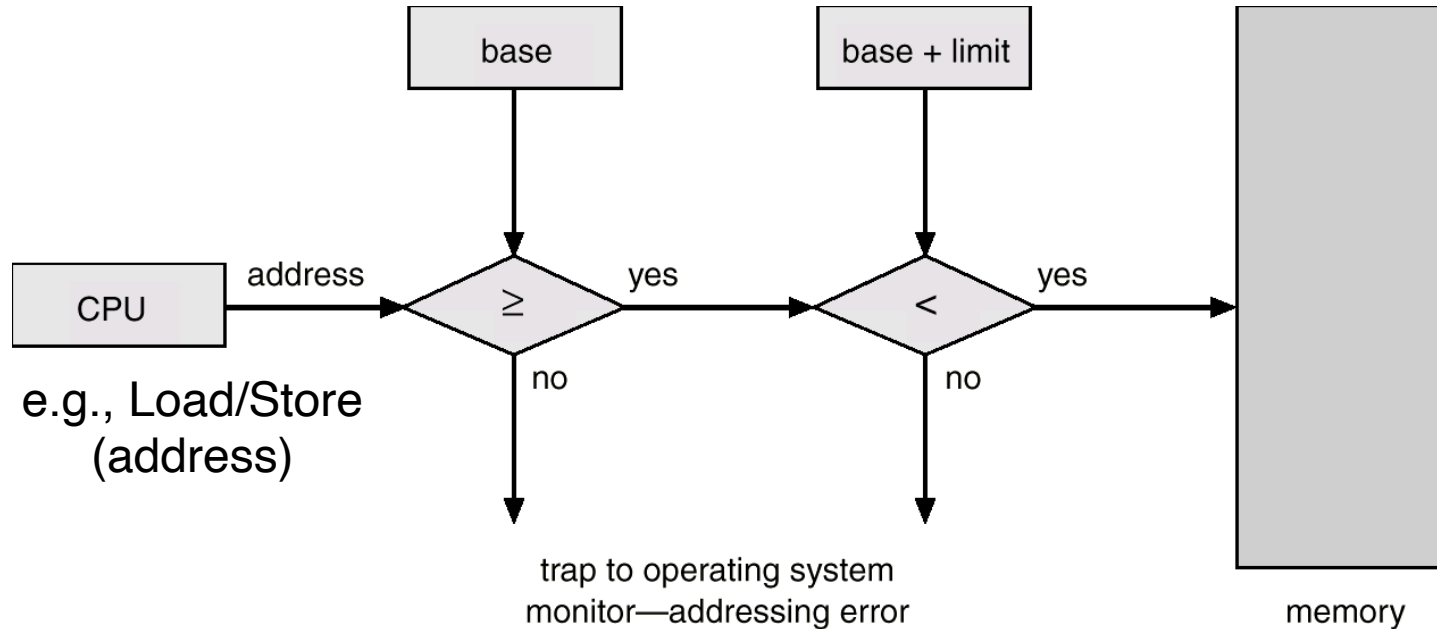
- Must provide memory protection, at least for the interrupt vector and the interrupt service routines

- Two CPU registers determine the range of legal addresses a program may access:
  - **Base register:** Holds the first legal physical memory address
  - **Limit register:** Contains the size of legal range

- Memory outside the defined range is protected and cannot be accessed

1.35

Memory Protection (Cont.)

```
          0  ┌─────────────┐
             │             │
             │   monitor   │
             │             │
     256000  ├─────────────┤
             │             │
             │    job 1    │
             │             │
     300040  ├─────────────┤  ◄─────  ┌──────────┐
             │             │          │  300040  │
             │    job 2    │          └──────────┘
             │             │           base register
     420940  ├─────────────┤  ◄─────  ┌──────────┐
             │             │          │  120900  │
             │    job 3    │          └──────────┘
             │             │           limit register
     880000  ├─────────────┤
             │    job 4    │
    1024000  └─────────────┘
```

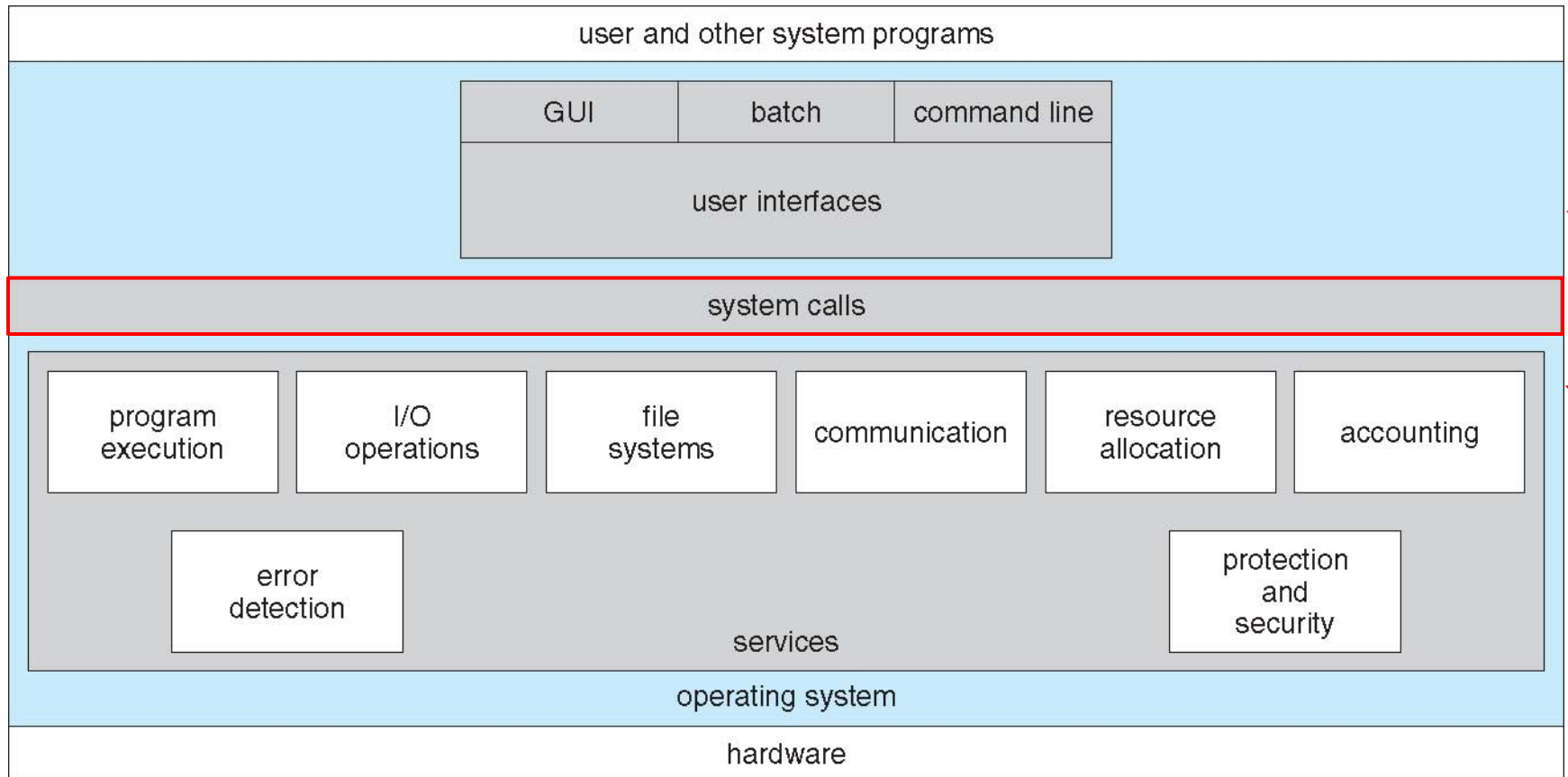# Memory Protection (Cont.)

e.g., Load/Store (address)

- The load instructions for the *base* and *limit* registers are privileged instructions (only in monitor mode)

- CPU issues a trap to the OS if above checks fail

Part 1:   Introduction

- What is an Operating System (OS)?

- Types of Computing Systems

- Computer System Architecture (Review)
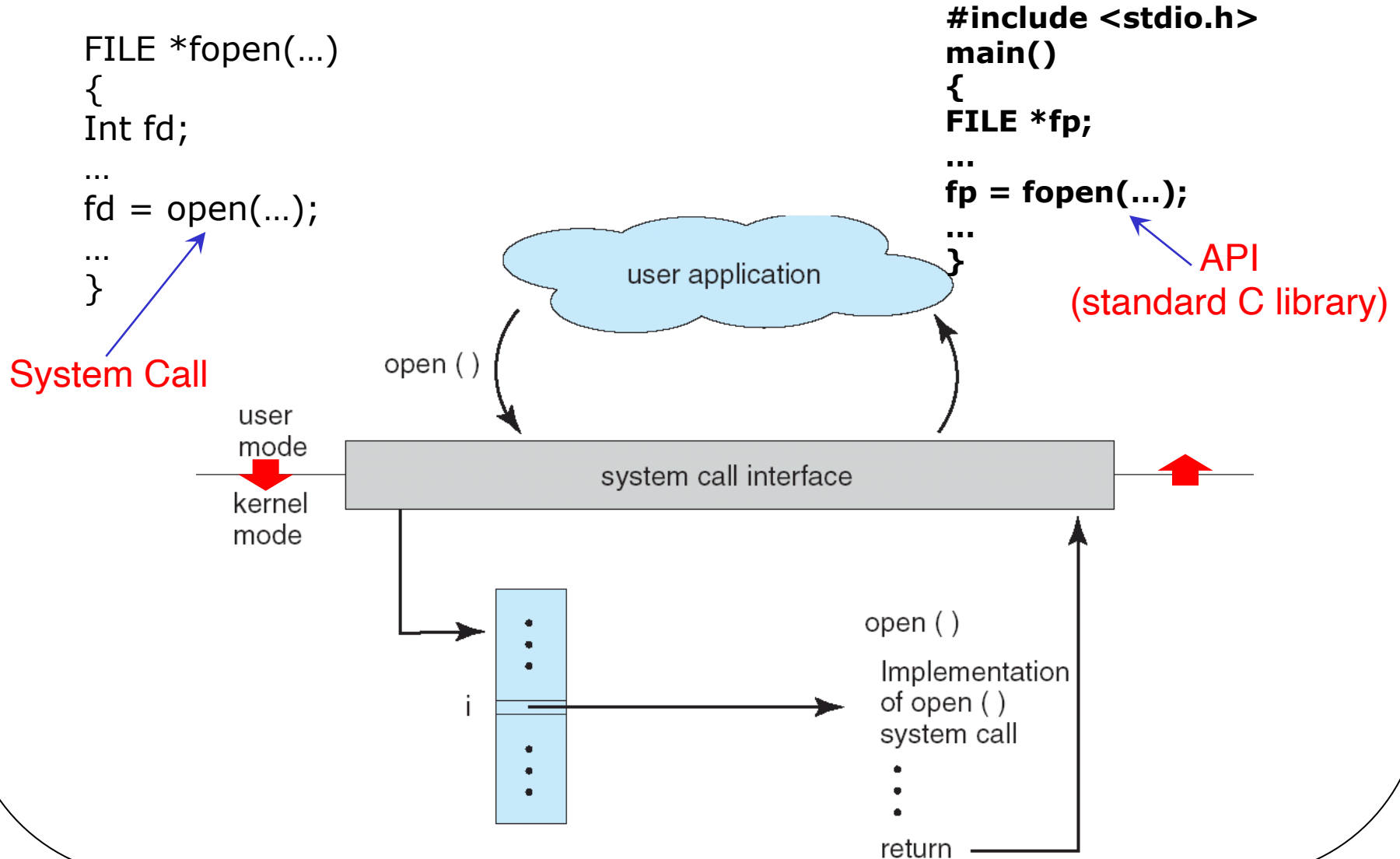
- **Operating System Services**

A View of Operating System Services

user and other system programs		

GUI	batch	command line
user interfaces		

system calls

program execution	I/O operations	file systems	communication	resource allocation	accounting

error detection	protection and security
services	

operating system

hardware

System Calls

- System calls provide the interface between a user program and the operating system
  - Generally available as assembly-language instructions
  - Possible to replace assembly language for systems programming to allow system calls to be made directly (e.g., in C/C++)

- The execution of a system call requires the switch from the user to the kernel mode
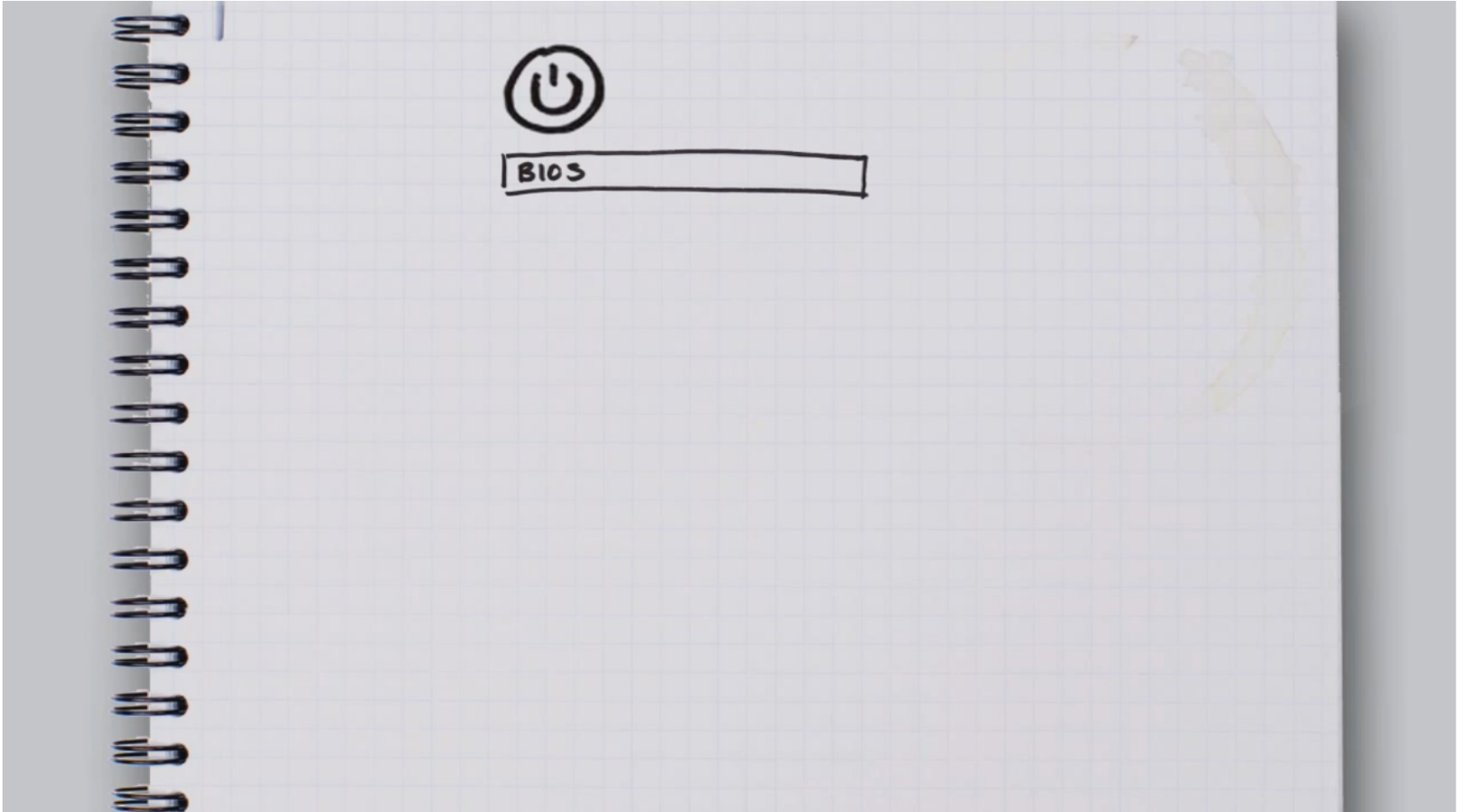
API – System Call – OS Relationship

FILE *fopen(…)
{
Int fd;
…
fd = open(…);
…
}

System Call

```
#include <stdio.h>
main()
{
FILE *fp;
…
fp = fopen(…);
…
}
```

**API
(standard C library)**

user application

open ( )

user
mode

kernel
mode

system call interface

i

open ( )

Implementation
of open ( )
system call

return

Advanced Readings

- Mobile operating systems
    - Apple iOS, http://en.wikipedia.org/wiki/IOS
    - Google Android, http://en.wikipedia.org/wiki/Android_(operating_system)
    - Microsoft Windows Phone, http://en.wikipedia.org/wiki/Windows_Phone

- "History of Operating Systems", by Ayman Moumina (pdf in NTULearn)

1.42

New Trend in OS

http://www.youtube.com/watch?v=0QRO3gKj3qw

Future Operating Systems

- Operating systems are **not** limited to Windows/Linux/MacOS etc.

- Operating system development never stops
  - Changes in hardware
  - Changes in user requirements

- Examples:
  - Microkernel (also known as μ-kernel), https://en.wikipedia.org/wiki/Microkernel
  - Formally verified OS kernel, https://sel4.systems/