Part 1: Introduction

- **What is an Operating System (OS)?**
- Types of Computing Systems
- Computer System Architecture (Review)
- Operating System Services

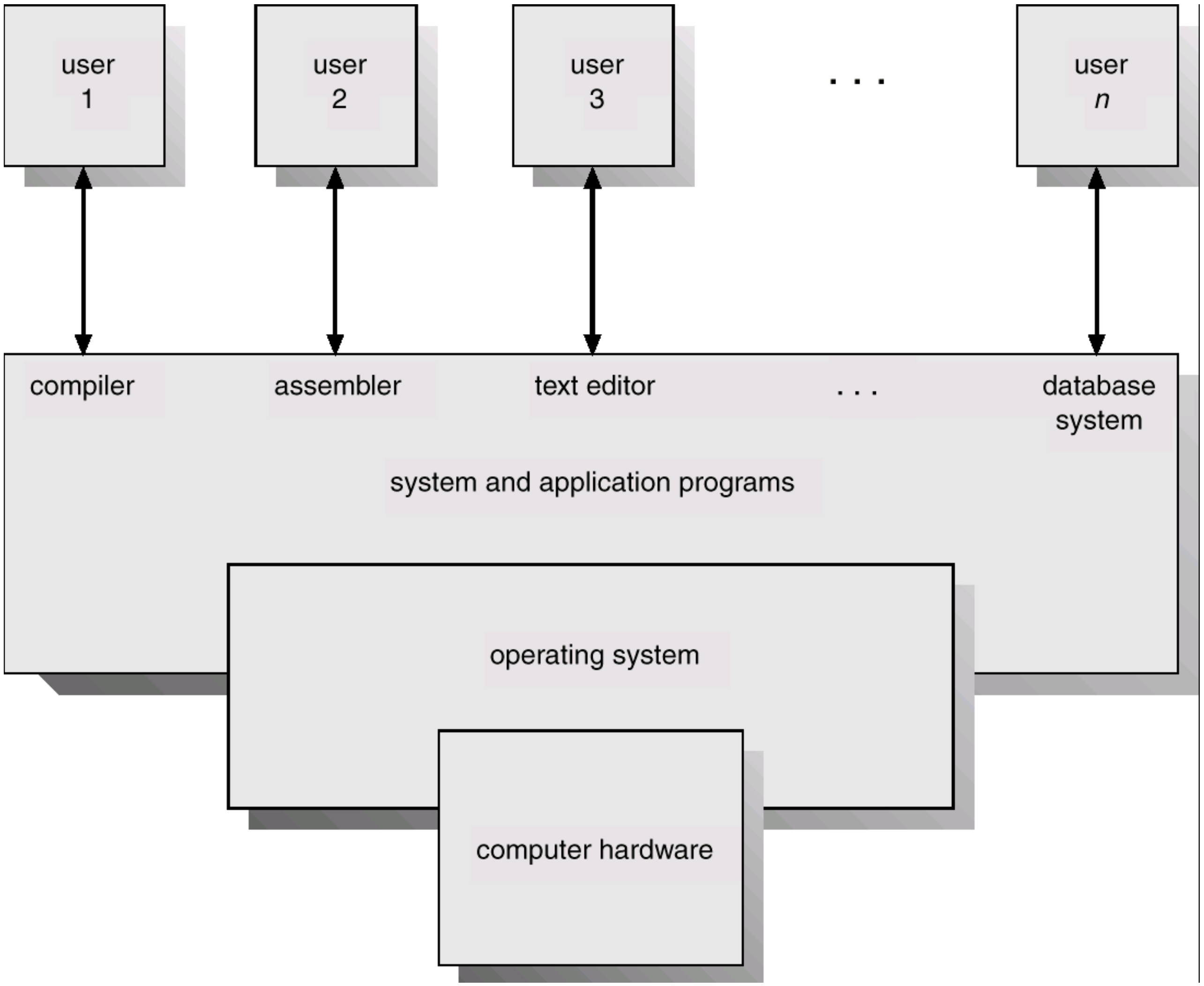
What is an Operating System?

- 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



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)

Part 1: Introduction

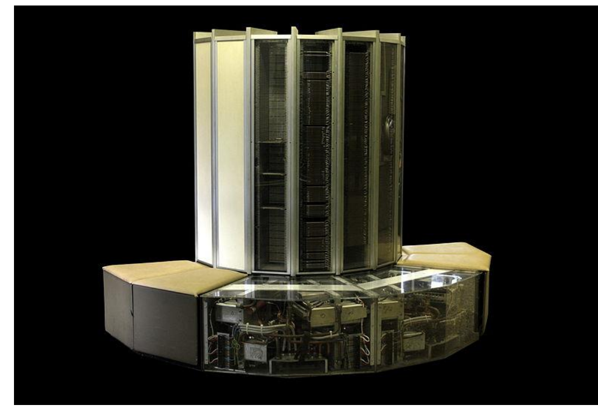
- What is an Operating System (OS)?
- **Types of Computing Systems**
- Computer System Architecture (Review)
- Operating System Services

Types of Computing Systems

- Batch systems
- Multiprogrammed and Time-sharing systems
  - Desktop systems
- Embedded and Cyber-physical systems
  - Real-Time systems
  - Handheld systems

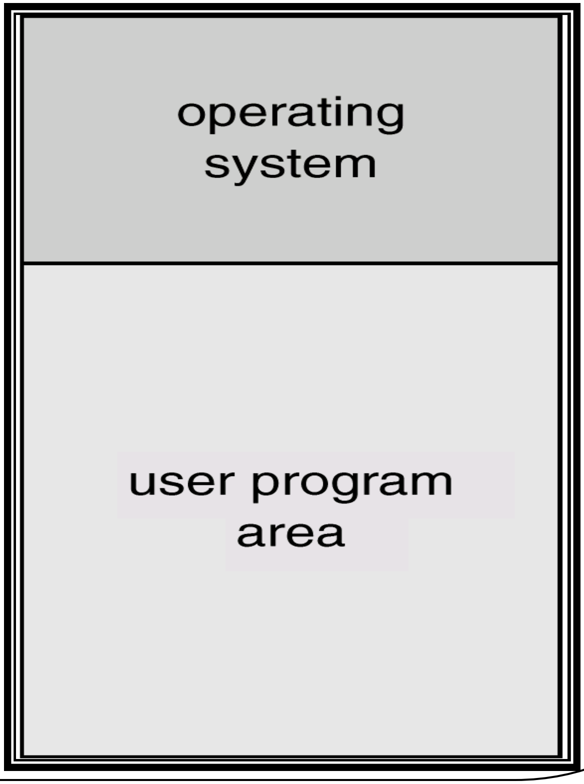
A Short Review on Computers

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



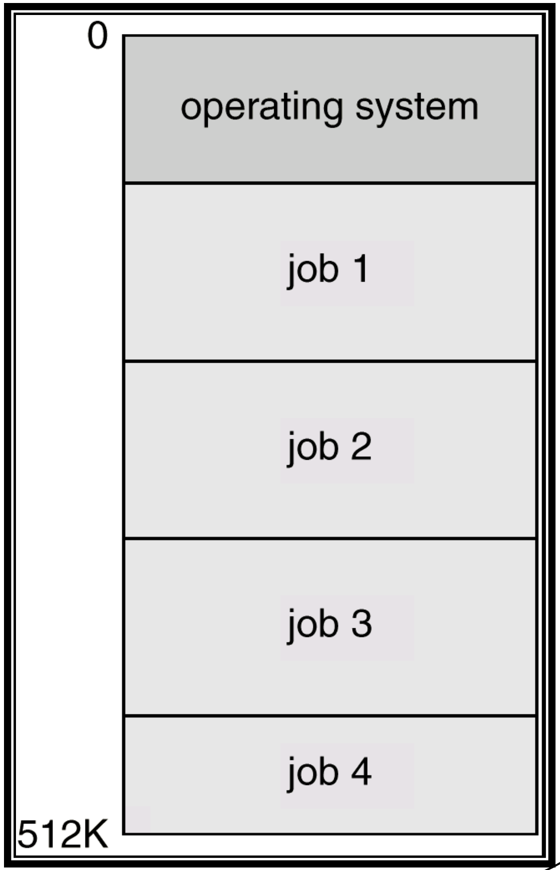
Simple Batch Systems

- Reduce setup time by batching similar jobs
- **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



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



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.

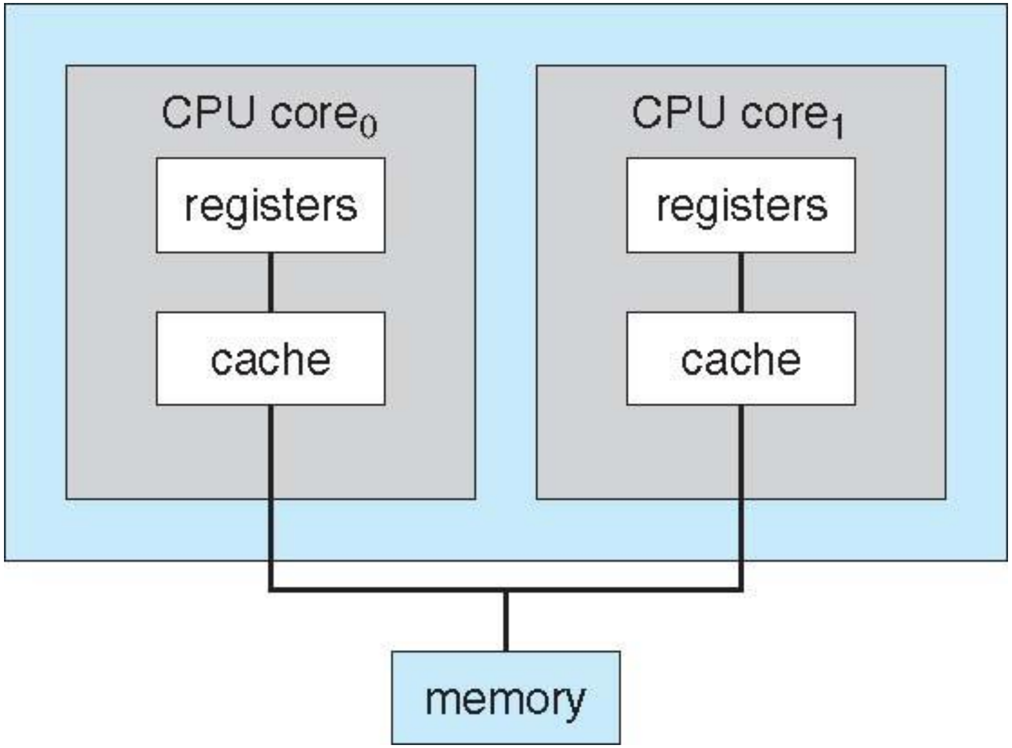
Real-Time Systems

- 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

A Dual-Core CPU Design



Multiprocessor Systems

- Systems with more than one CPU, or CPU with multiple cores (also called multi-core systems)

- *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

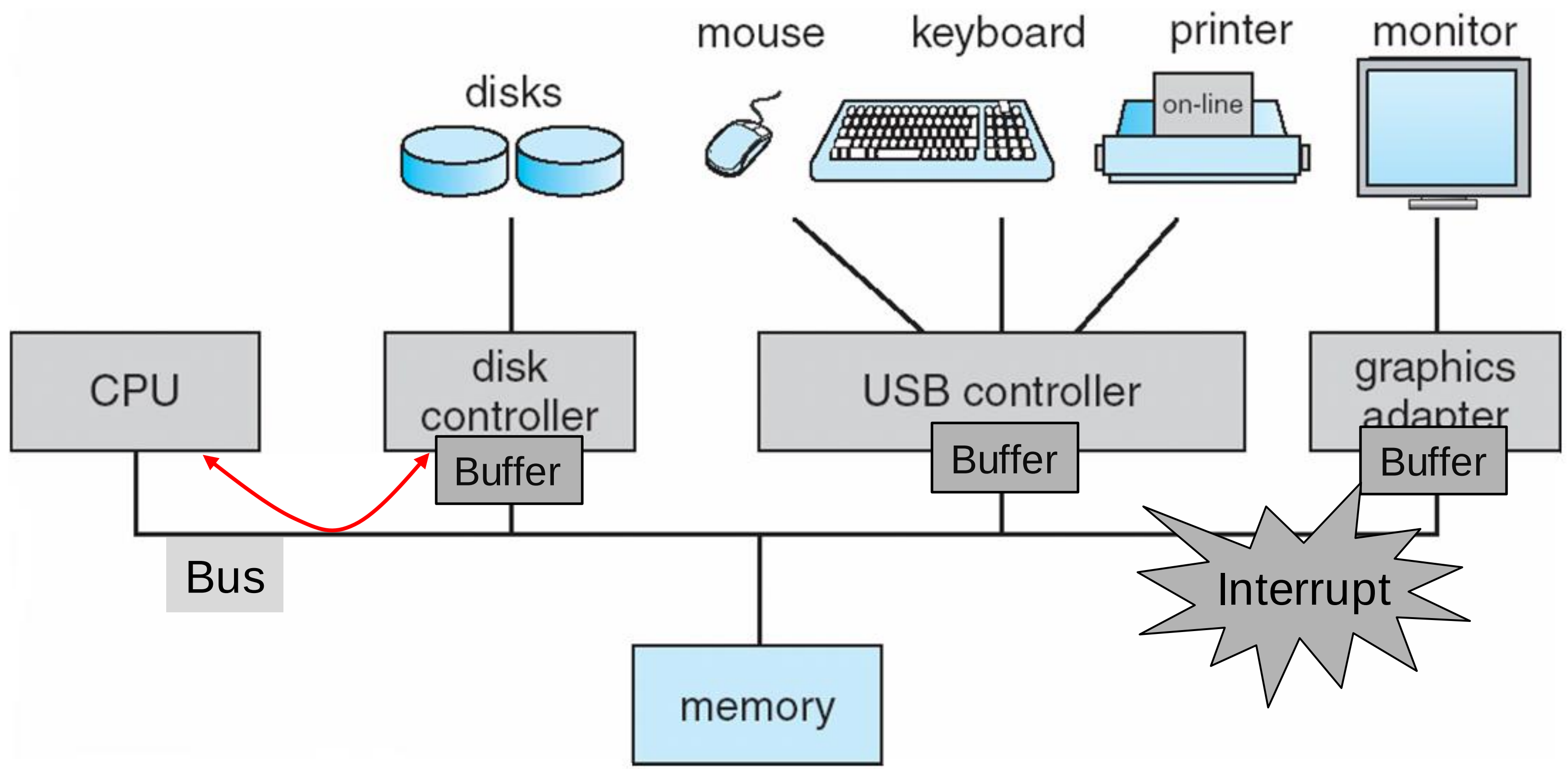
Computer System Architecture

- Computer System Operation
- Storage Hierarchy
- Hardware Protection

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



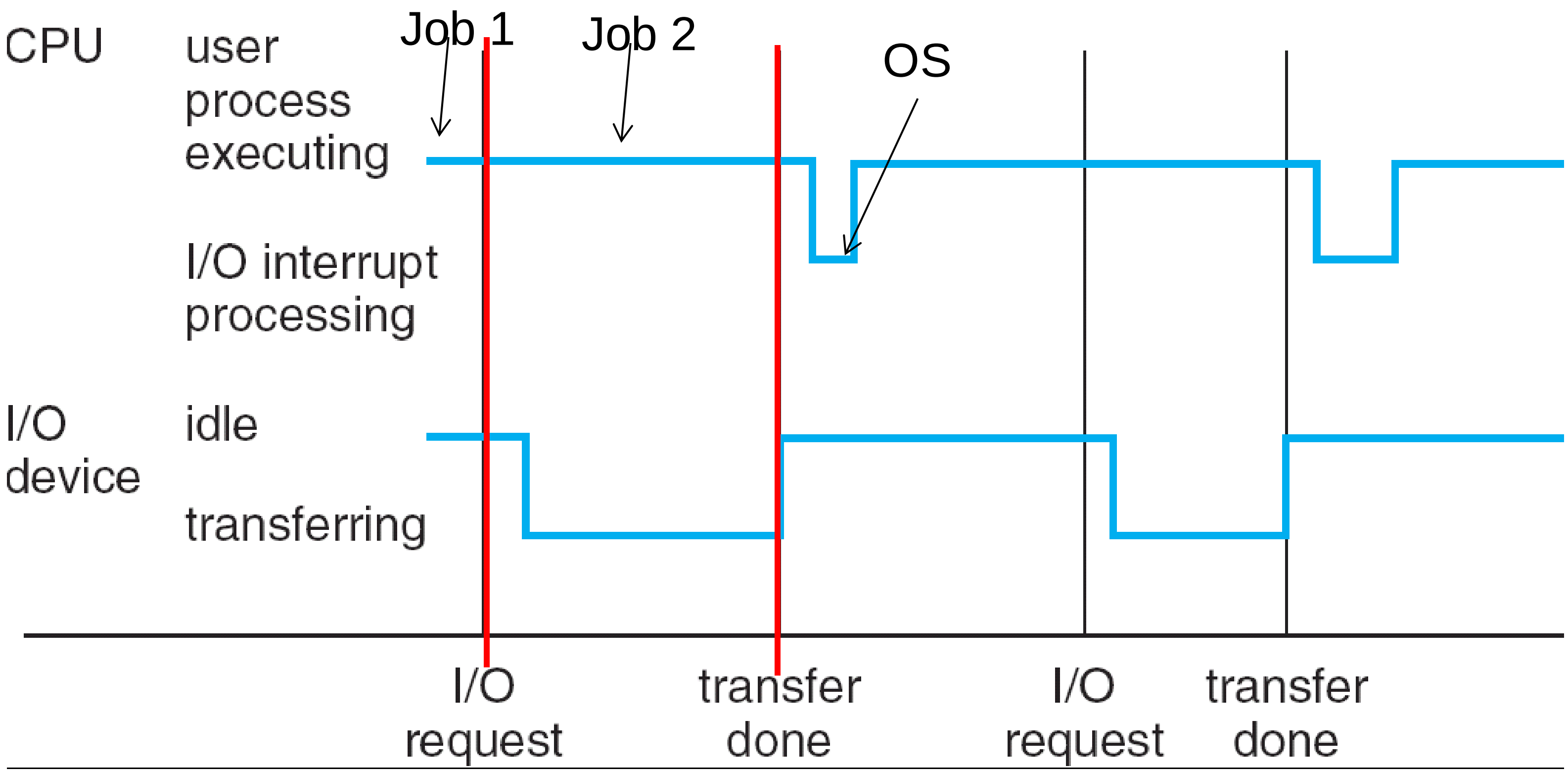
Common Functions of Interrupts

- 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*

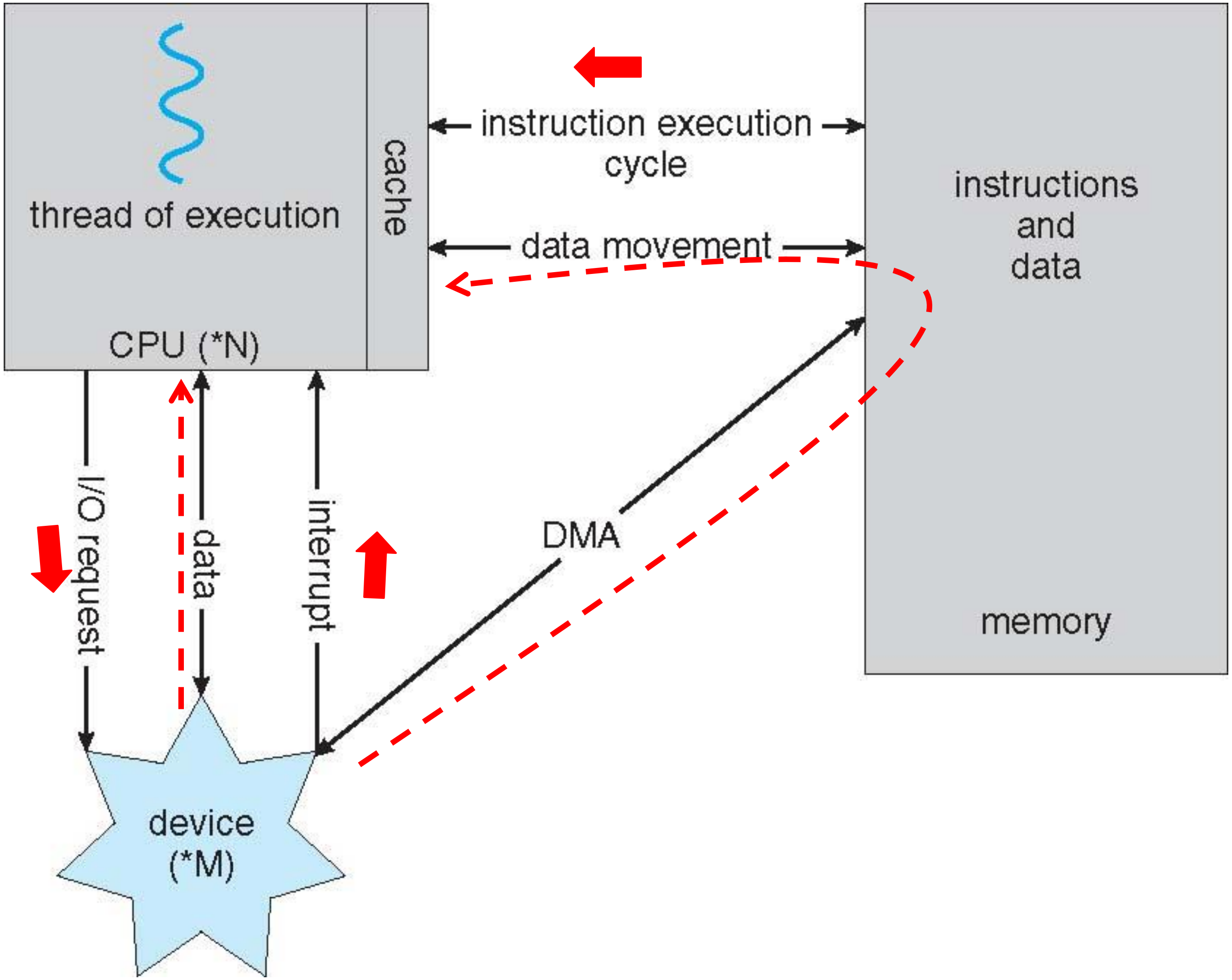
Interrupt-Driven I/O Timeline



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



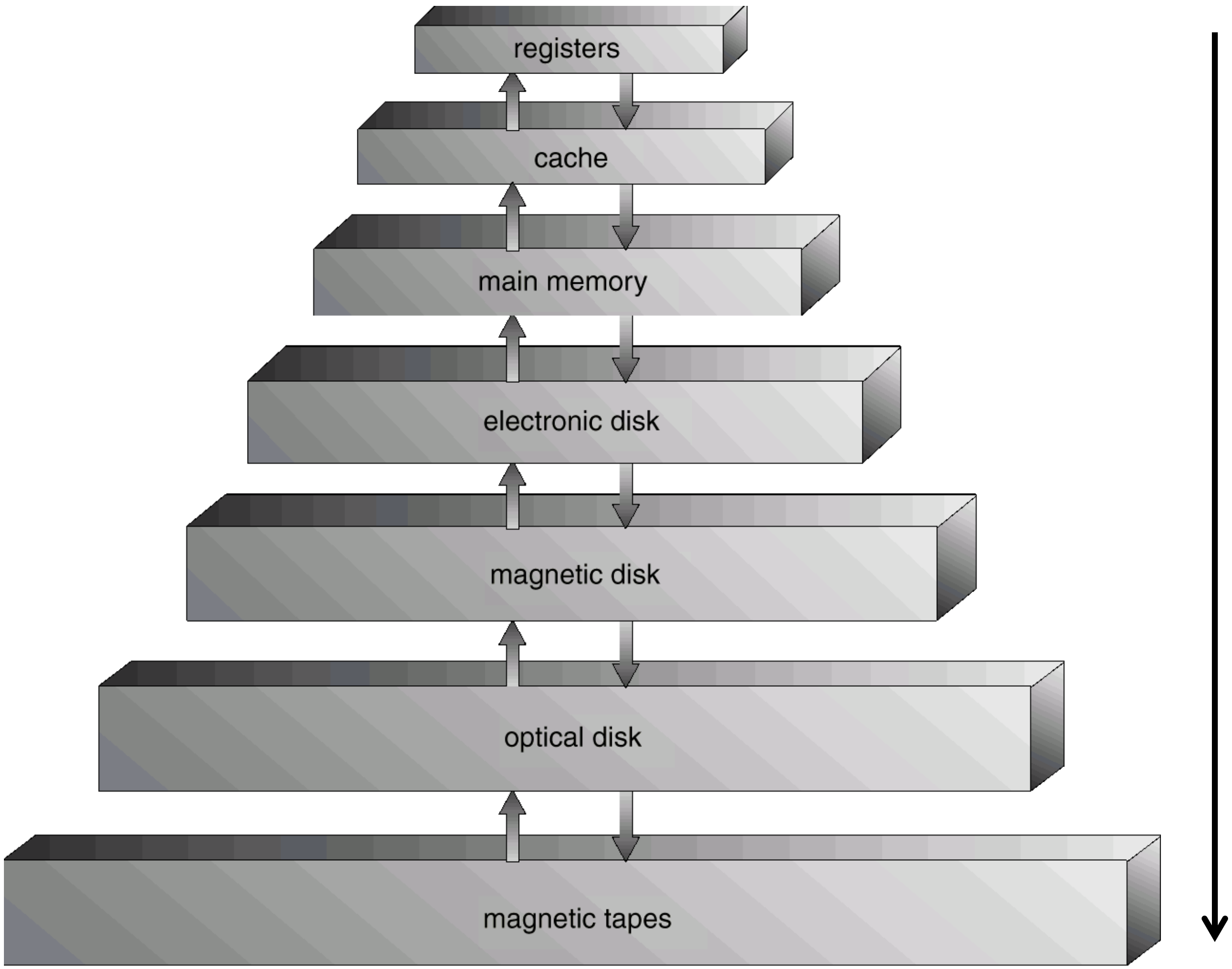
Storage Hierarchy

- 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.

Hardware Protection

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

Dual-Mode Operation

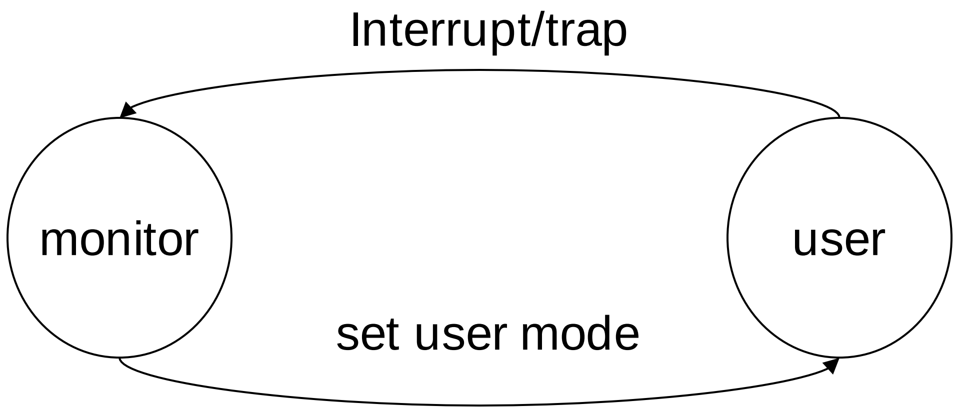
- Provides hardware protection by differentiating between at least two modes of operations

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



- *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

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

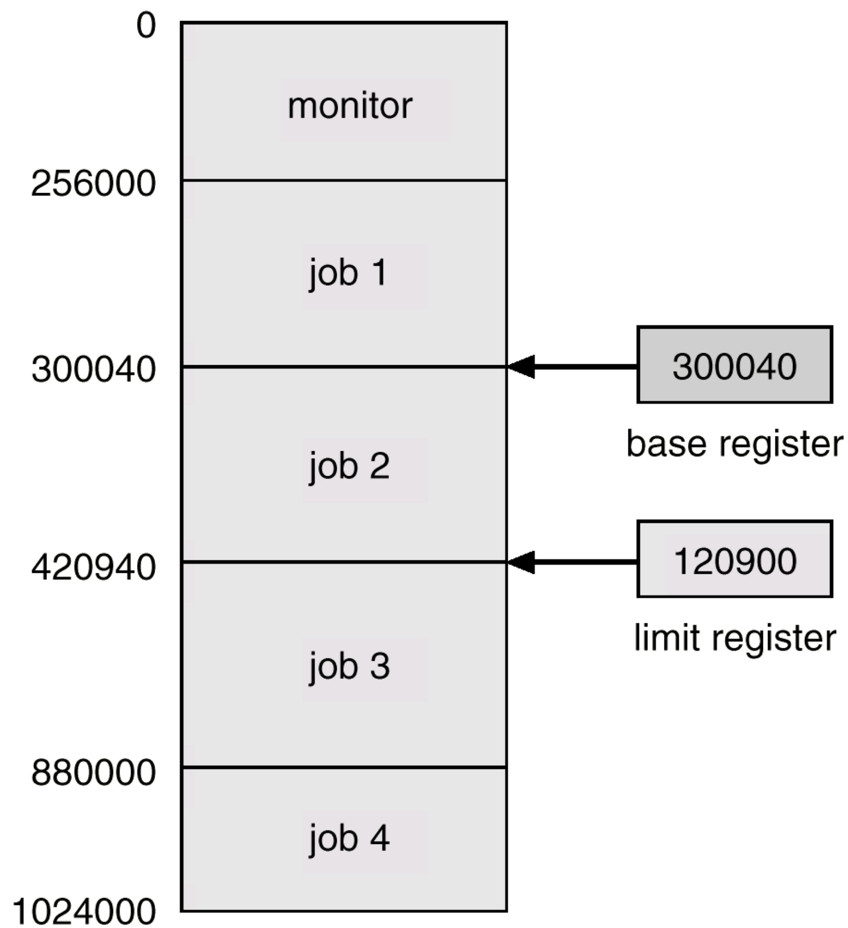
Memory Protection

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

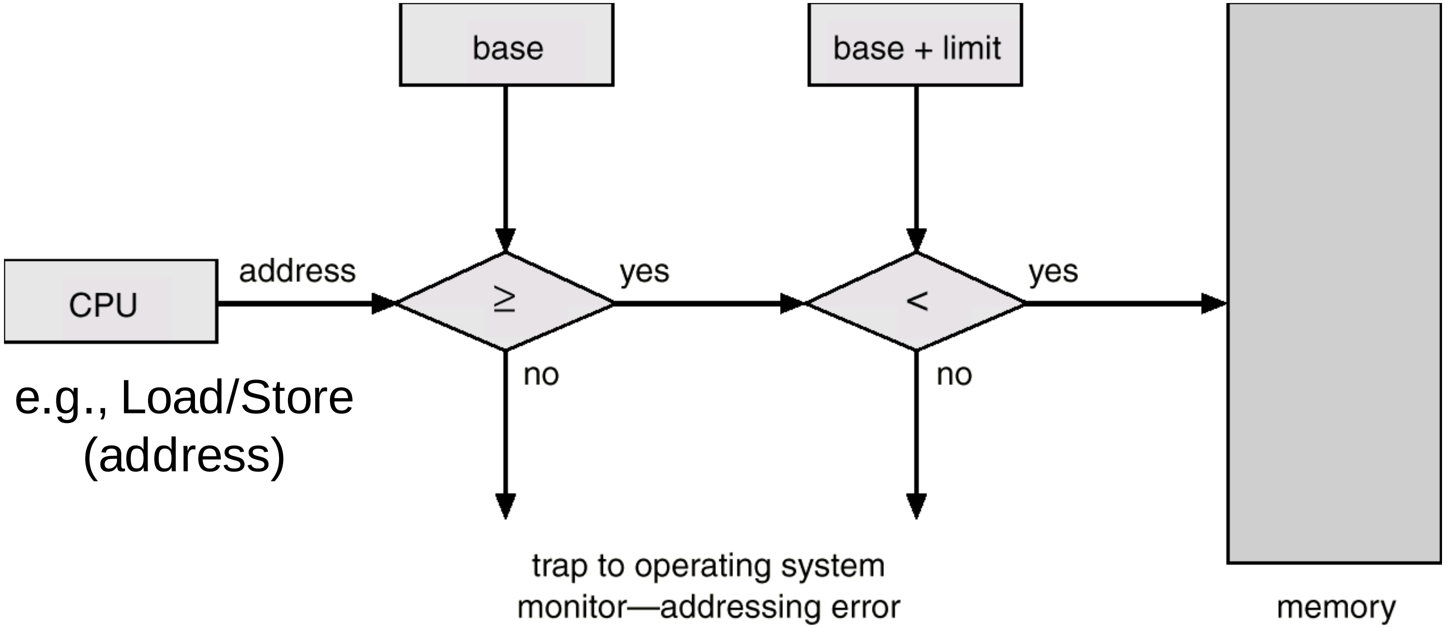
- Two CPU registers determine the range of legal addresses a program may access:
  - **Base register**: Holds the first legal physical memory address
  - **Limit register**: Contains the size of legal range

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

Memory Protection (Cont.)



Memory Protection (Cont.)



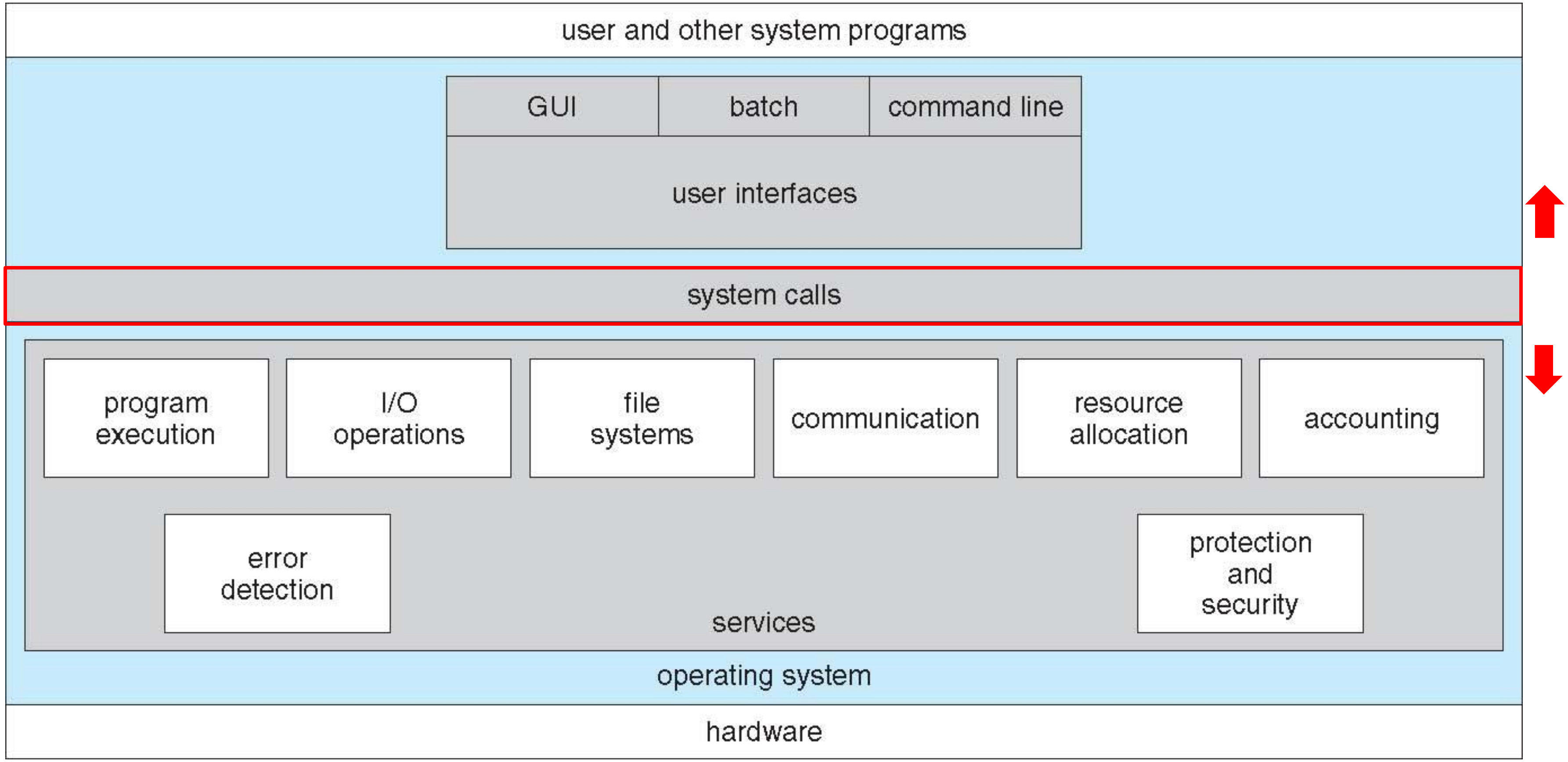
e.g., Load/Store (address)

- 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



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

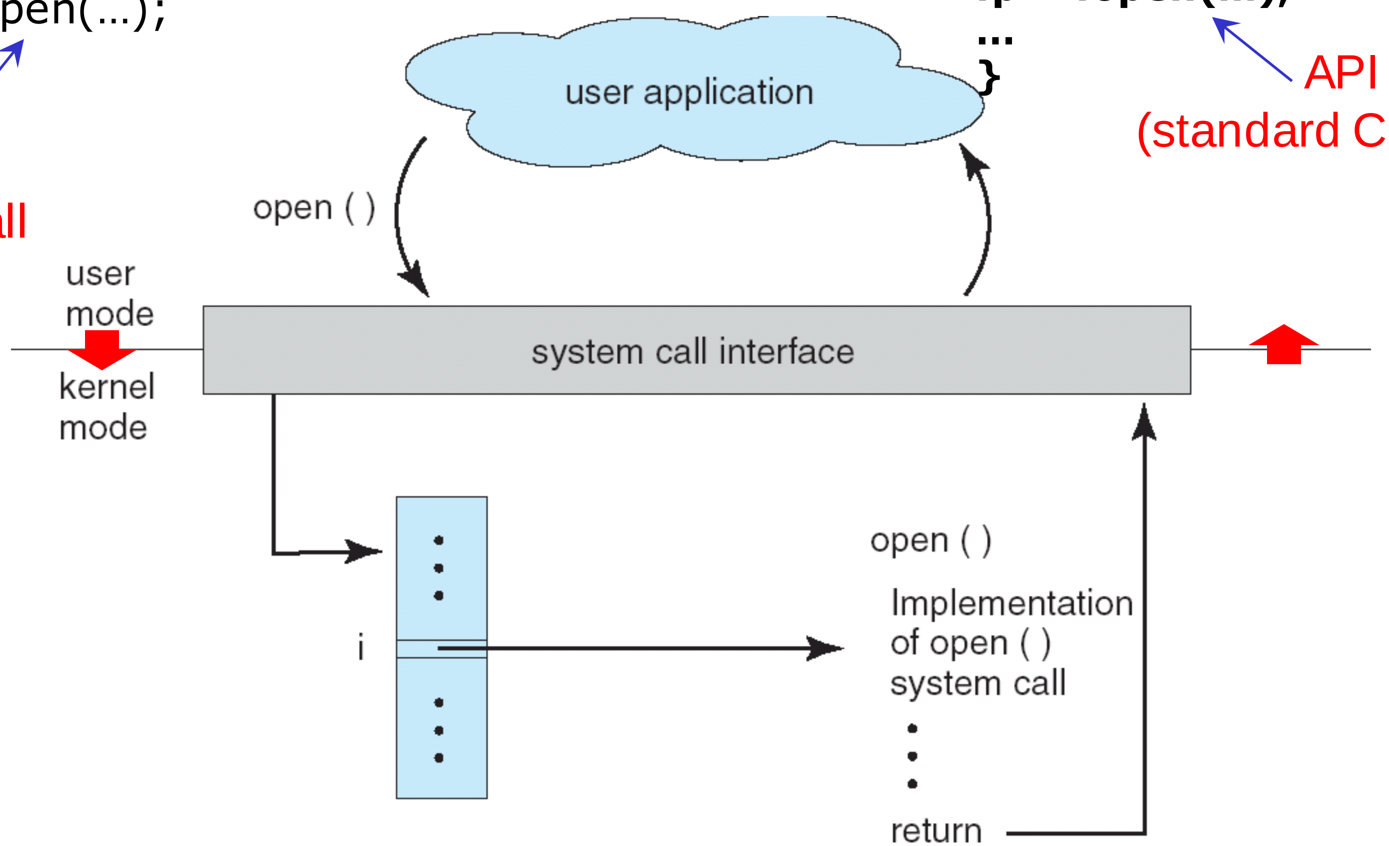
API – System Call – OS Relationship

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

System Call

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

API
(standard C library)



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)

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/