# Part 1: Introduction

- What is an operating system?
- Mainframe Systems
- Desktop Systems
- Multiprocessor Systems
- Real-Time Systems
- Handheld Systems
- Computer System Structures
- 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 efficiency.
  - Make the computer system convenient to use.
  - Use the computer hardware in an efficient manner.

# Computer System Components

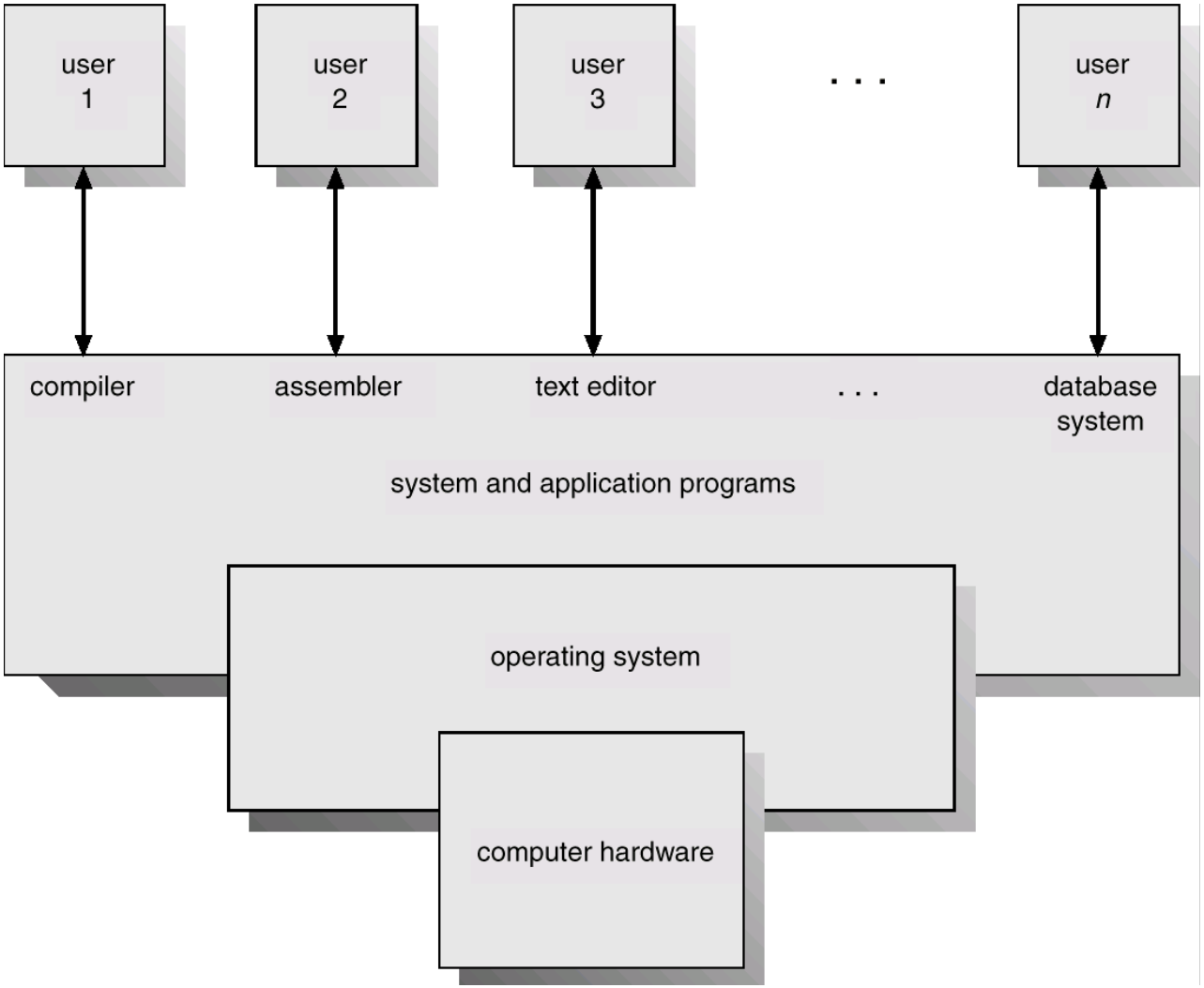
1. Hardware – provides basic computing resources (CPU, memory, I/O devices).

2. Operating system – controls and coordinates the use of the hardware among the various application programs for the various users.

3. Application programs – define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs).

4. Users (people, machines, other computers).

# Abstract View of System Components



# Operating System Definitions

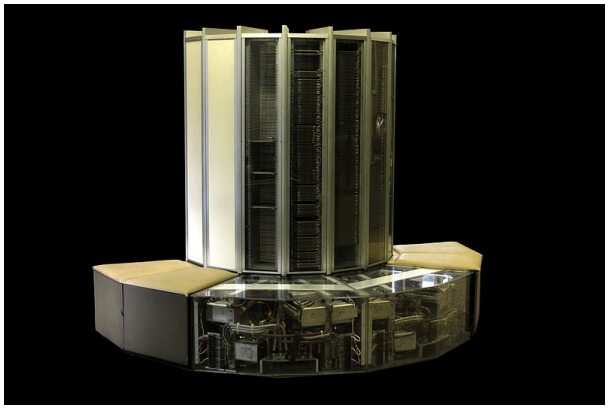
- Resource allocator – manages and allocates resources.
- Control program – controls the execution of user programs and operations of I/O devices.
- **Kernel** – the one program running at all times (all else being application programs).

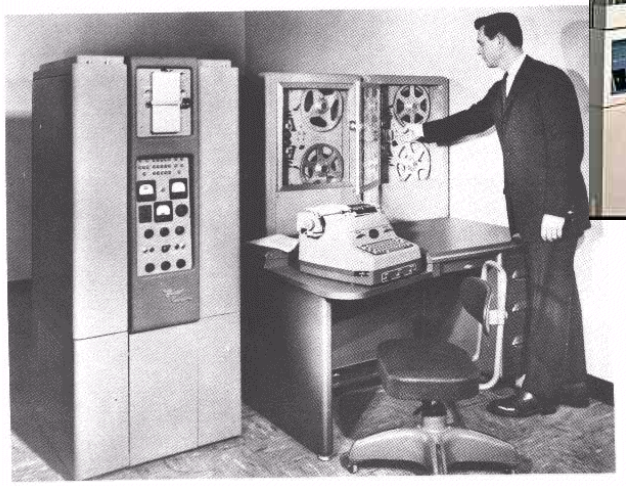
# Mainframe Systems

- Batch systems
- Multiprogrammed systems
- Time-sharing systems

# A Short Review on Computers

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





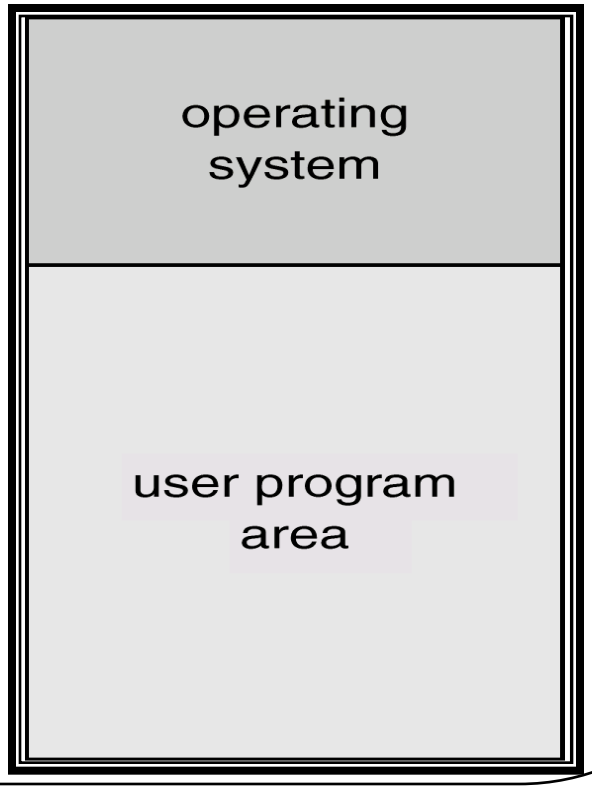
# Computer History Museum


**Source:**

http://www.youtube.com/watch?v=7e6EAAY-WfA

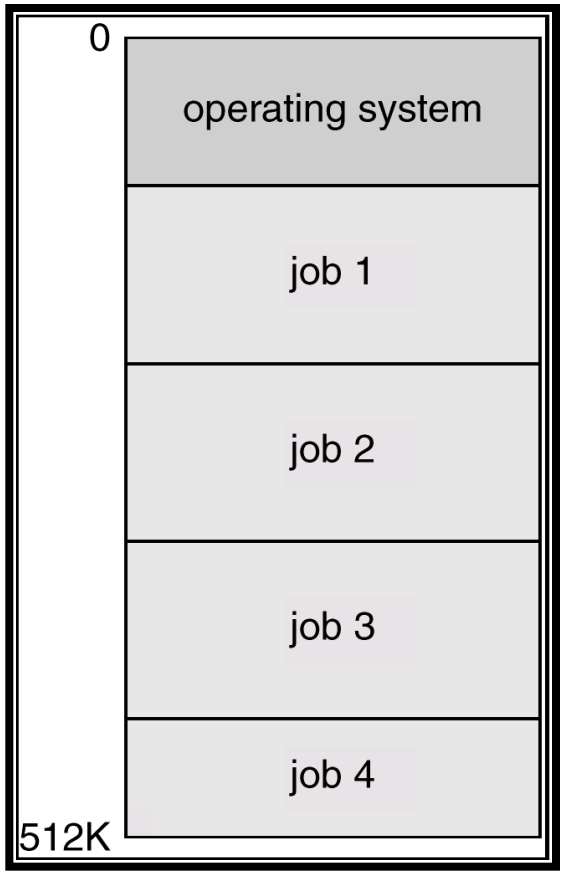
# Simple Batch Systems

- Reduce setup time by batching similar jobs.
- **Automatic job sequencing** – automatically transfers control from one job to another.
- Simple memory layout



# Multiprogrammed Systems

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



# OS Features Needed for Multiprogramming

- Memory management – to allocate memory to several jobs.
- CPU scheduling –to choose among several jobs ready to run.
- Allocation of devices.

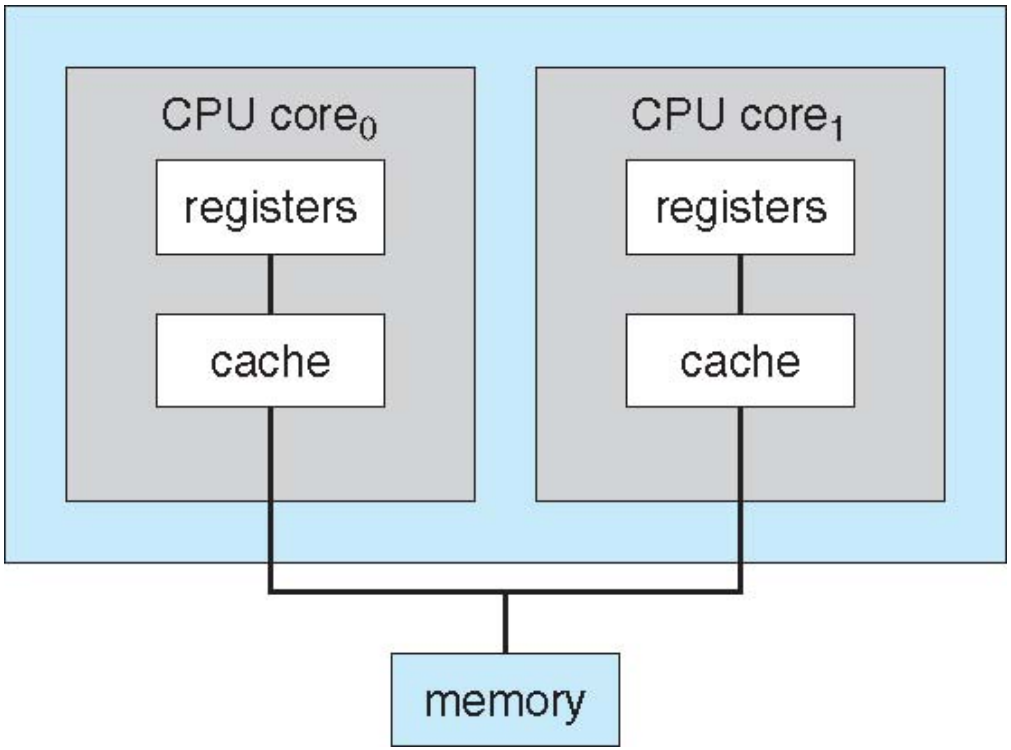
# Time-Sharing Systems

- The CPU is multiplexed among several jobs that are in memory.
- A job is swapped in and out of memory to the disk.
- On-line communication between the user and the system is provided; when the operating system finishes the execution of one command, it seeks the next command from the user.

# Desktop Systems

- Personal computers – computer system dedicated to a single user.
- I/O devices – keyboards, mice, display screens, small printers.
- User convenience and responsiveness.
- May run several different types of operating systems (Windows, MacOS, UNIX, Linux)

# A Dual-Core CPU Design



# Multiprocessor Systems

- Multiprocessor systems with more than one CPU, or CPU with multiple cores.
- *Tightly coupled system* – communication usually takes place through the shared memory.
- Advantages of parallel system:
  - Increased system *throughput*
  - Economical in sharing memory and I/O devices (as compared to multiple single systems)
  - Increased reliability

# Real-Time Operating Systems

- Used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.
- Well-defined fixed-time constraints.
- Some popular systems: LynxOS, RTLinux, VxWorks etc.



# Handheld Systems

- Mobile phones, Pads
- Issues:
  - Limited memory
  - Slow processors
  - Small display screens.
- Popular systems: iOS, Android, Windows Phone

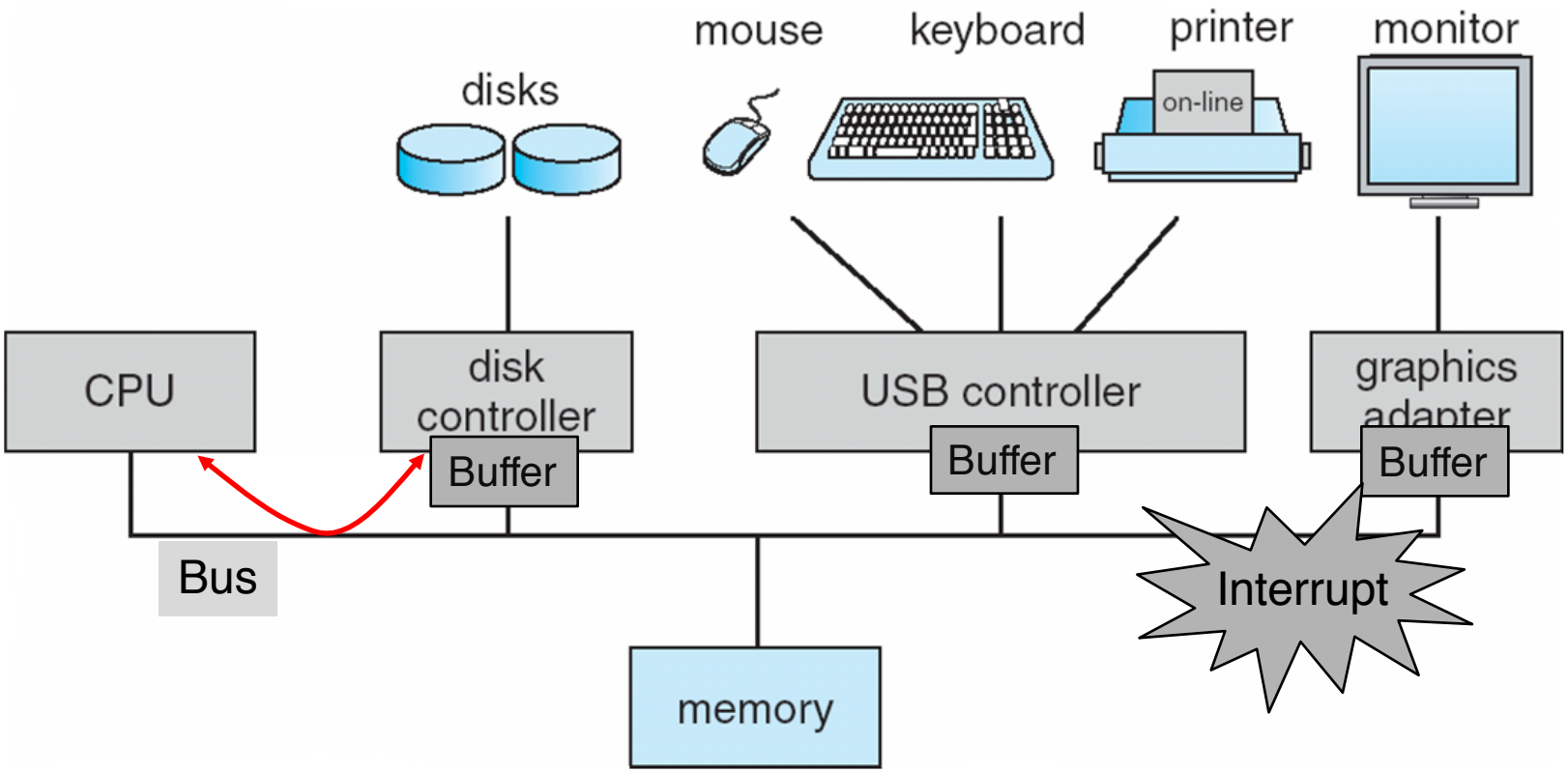

# Computer System Structures

- 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 to/from local buffer and then to/from memory
- Device controller informs CPU that it has finished its operation by causing an *interrupt*.

# Computer-System Operation



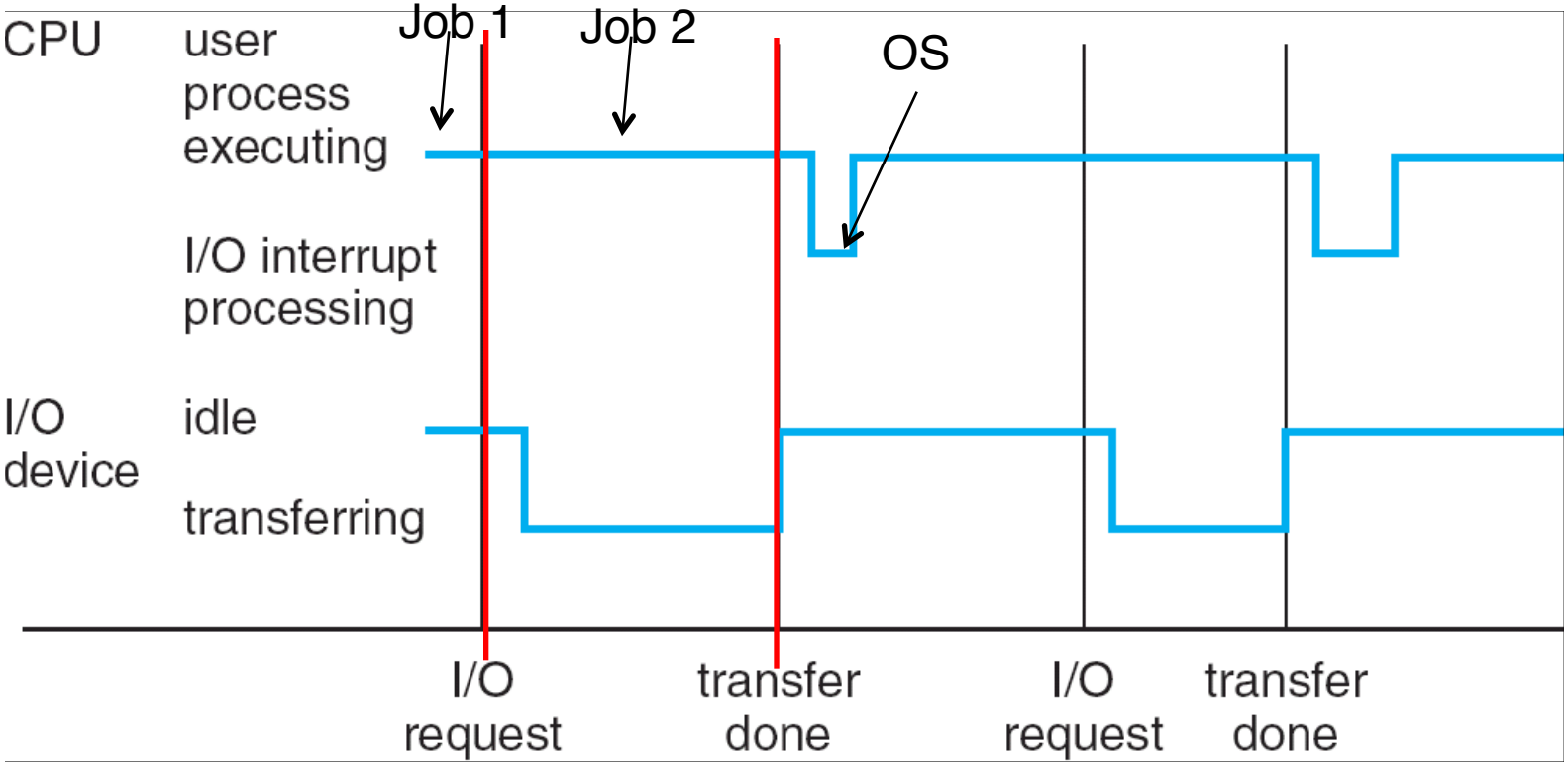
# Common Functions of Interrupts

- Interrupts 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 a *lost interrupt*.
- A *trap* is a software-generated interrupt caused either by an error or a user request.
  - Java program exceptions
  - Unhandled exceptions in user program
- An operating system is *interrupt* driven.
  - If the OS is not interrupt driven, it would require to poll for the task/event completion.

# Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter.
- Determines which type of interrupt has occurred.
- Separate segments of code determine what action should be taken for each type of interrupt.
- Example: interrupt driven I/O
  - Only for low-speed I/O devices.

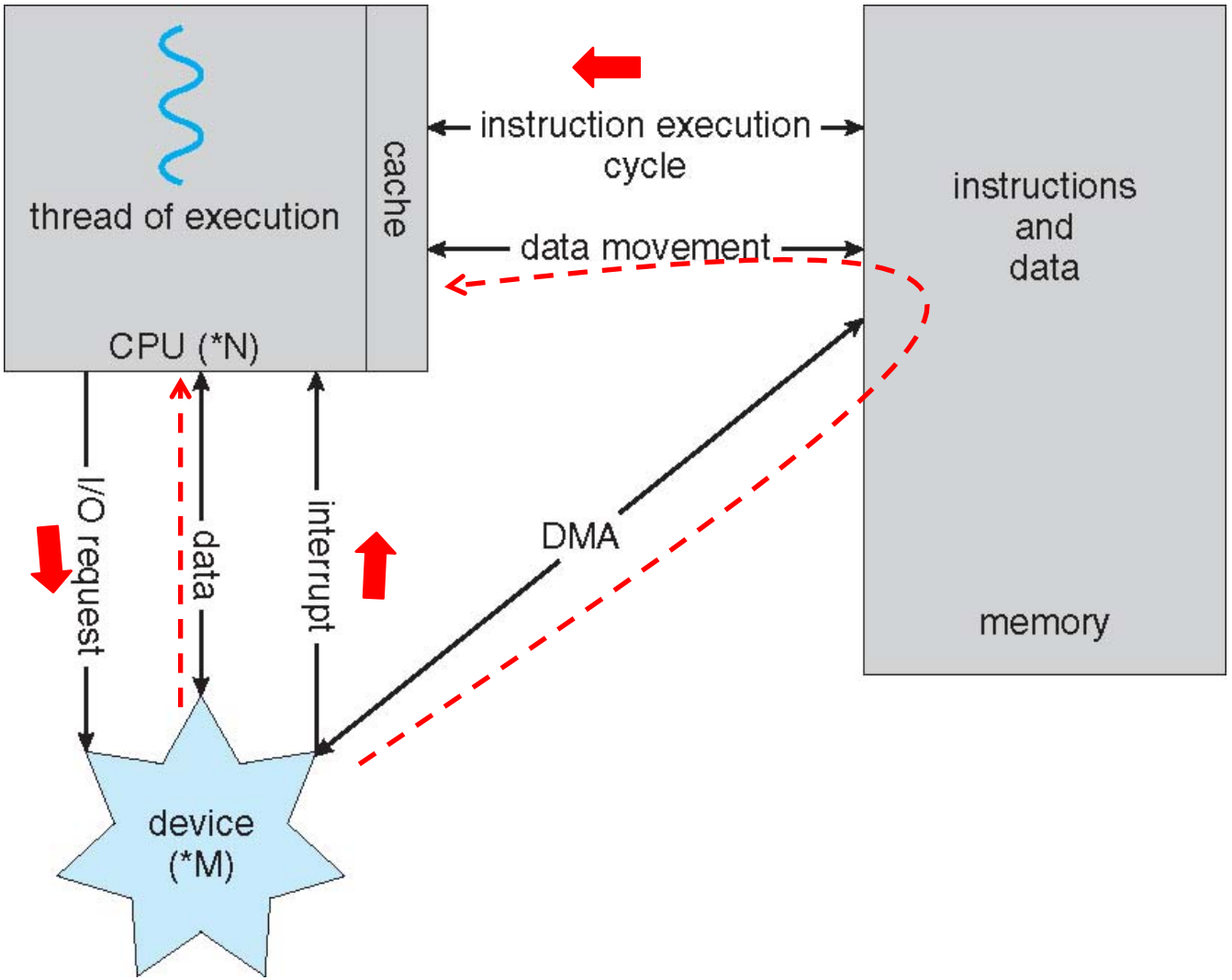
# Interrupt-Driven I/O Timeline



# Direct Memory Access (DMA) 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.

# How a Modern Computer Works



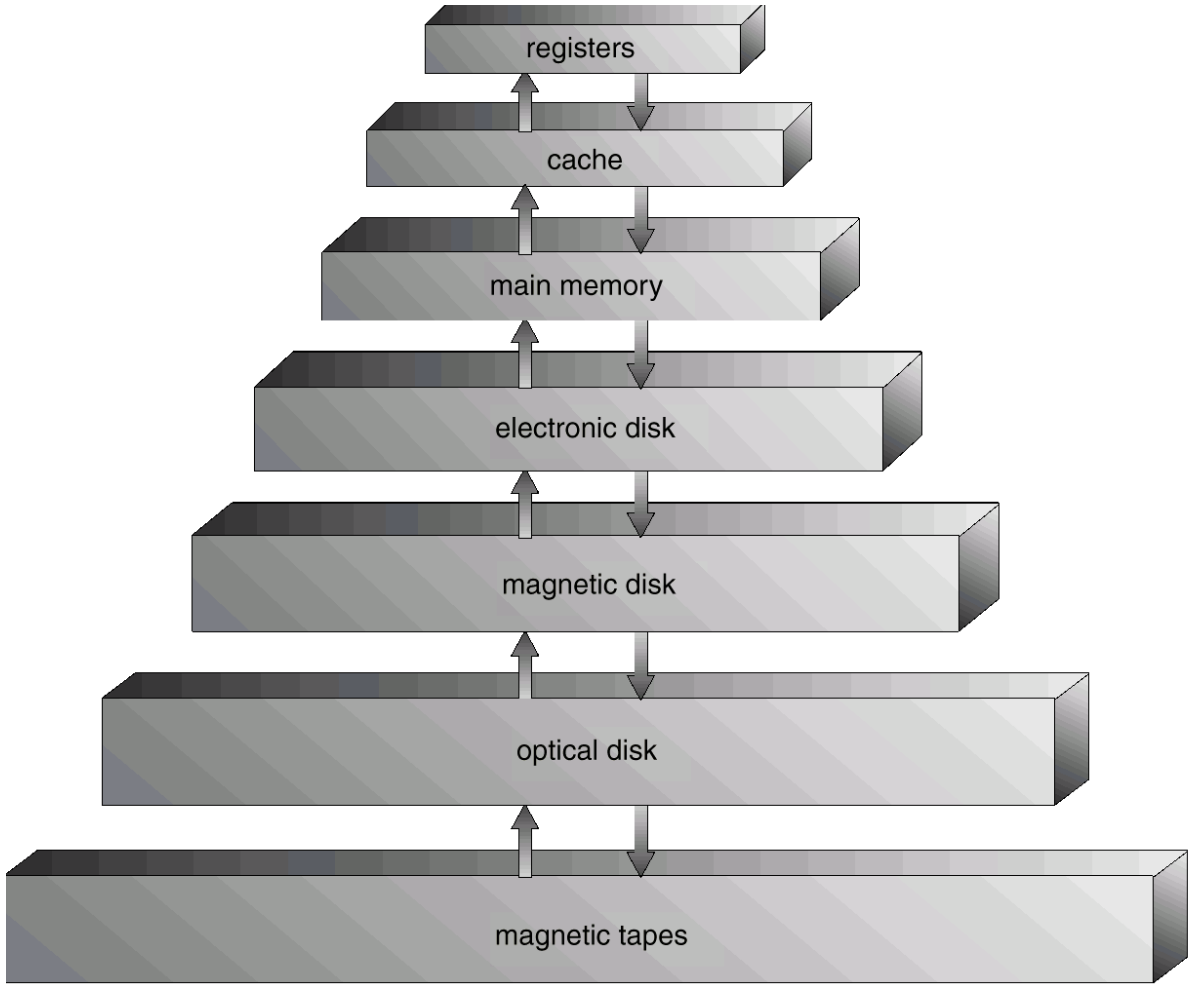
# Storage Hierarchy

- Hierarchy: CPU registers, CPU caches, main memory, hard disk …
- Storage systems organized in hierarchy.
  - speed
  - cost
  - volatility
  - size

This slide is not examinable.

- *Caching* – copying information into faster storage system; main memory can be viewed as a last *cache* for secondary storage.

# Storage Hierarchy




Size increases

Speed decreases

Cost/price decreases

This slide is not examinable.

# Hardware Protection

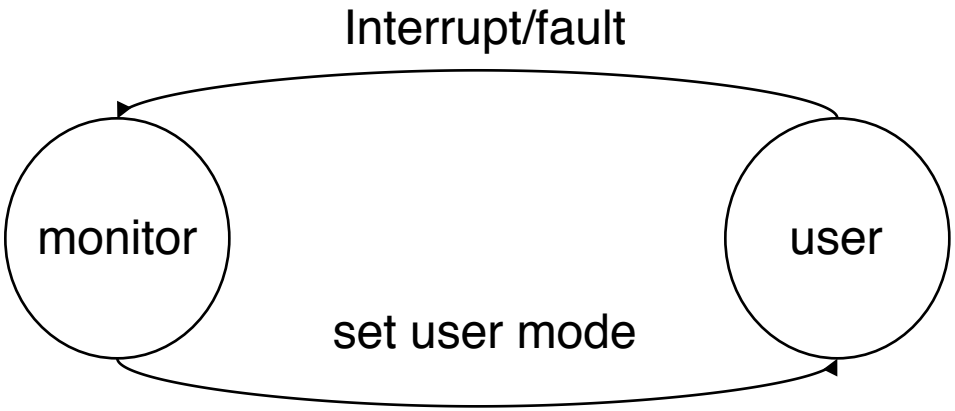
- Dual-Mode Operation
- I/O Protection
- Memory Protection

# Dual-Mode Operation

- Provide hardware protection to differentiate between at least two modes of operations.
  1. *User mode* – execution of user process.
  2. *Monitor mode* (also *supervisor mode* or *system mode or kernel mode*) – execution of operating system process.

# Dual-Mode Operation (Cont.)

- *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 used only in monitor mode.

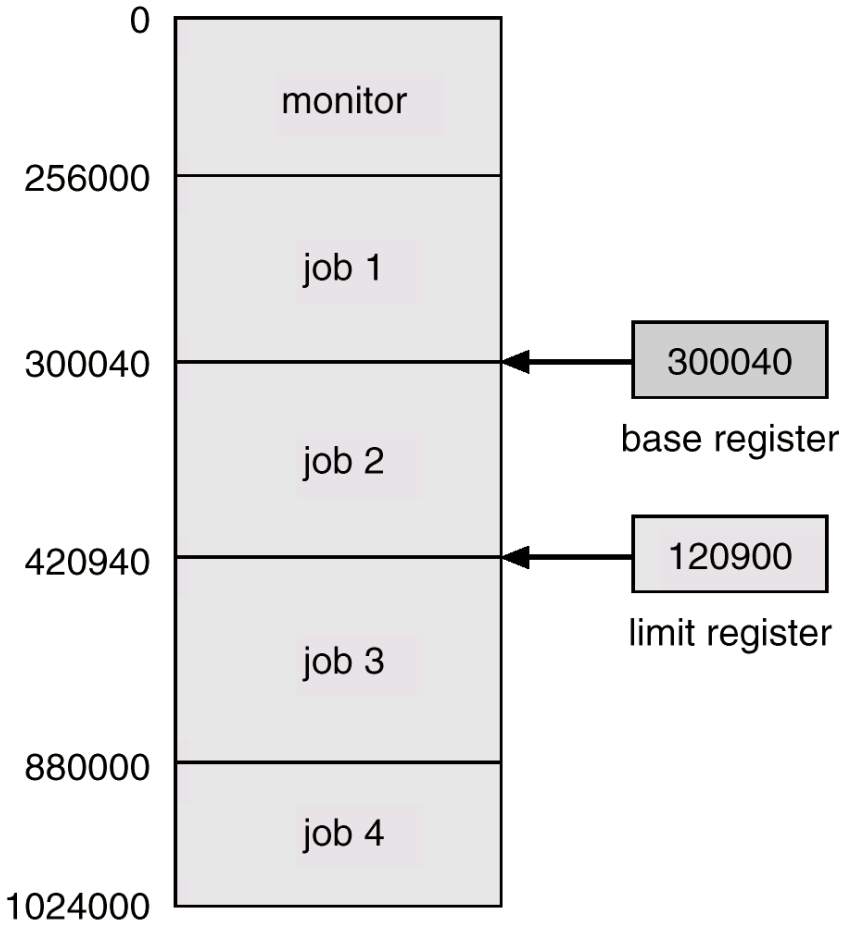
# I/O Protection

- 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 accesses to the devices.
- All I/O instructions are privileged instructions.
- I/O operations must go thru OS to ensure its correctness and legality.
  - OS **handles** a trap for illegal I/O operations (e.g., Java exceptions).

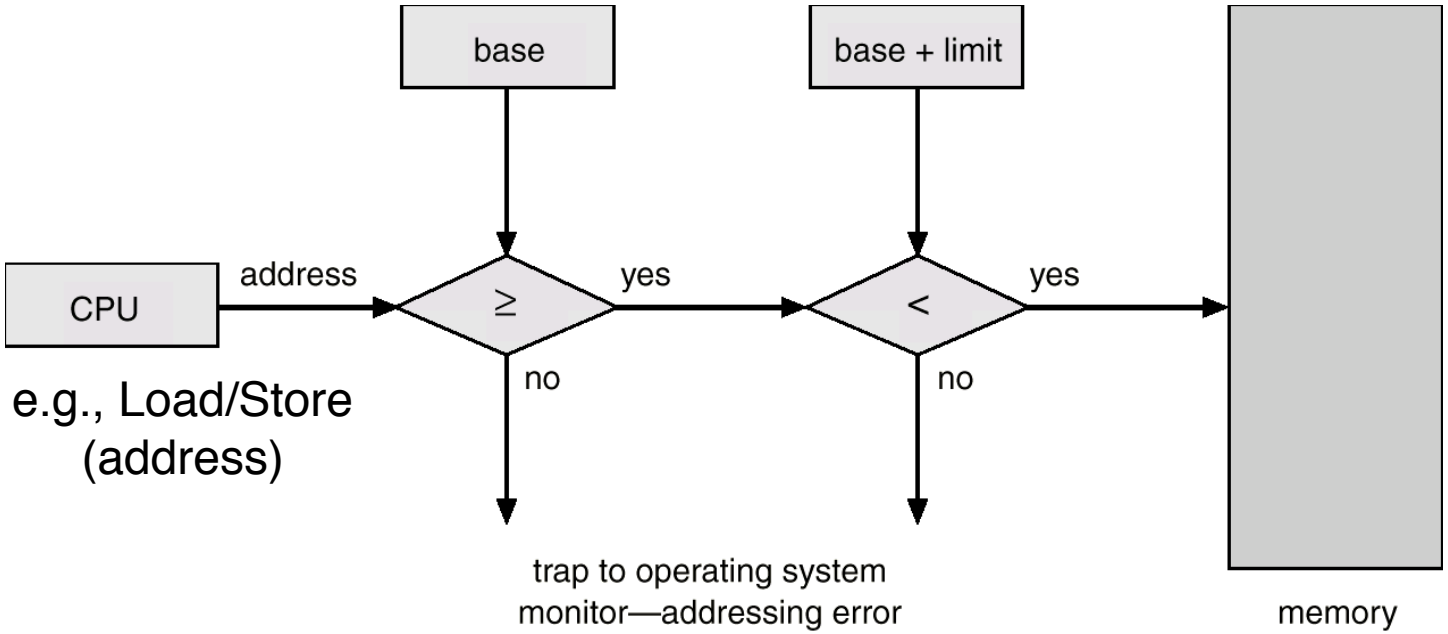
# Memory Protection

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

# Memory Protection (Cont.)

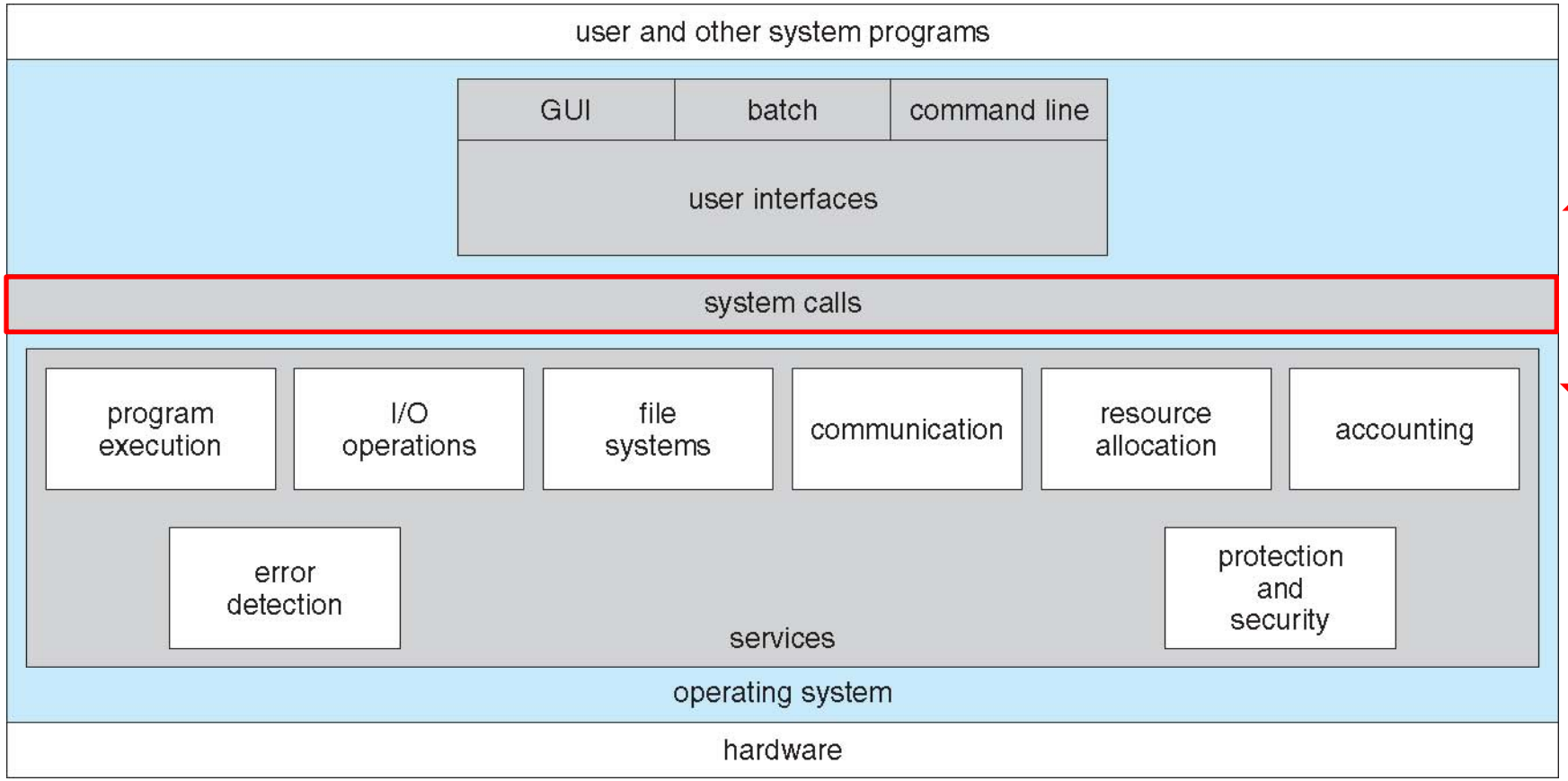


# Memory Protection (Cont.)




- The load instructions for the *base* and *limit* registers are privileged instructions.
- A trap is issued to operating system if either instruction fails.

# 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.
  - Languages defined to replace assembly language for systems programming to allow system calls to be made directly (e.g., C/C++.)
- The execution of a system call requires the switch between user mode and kernel mode.

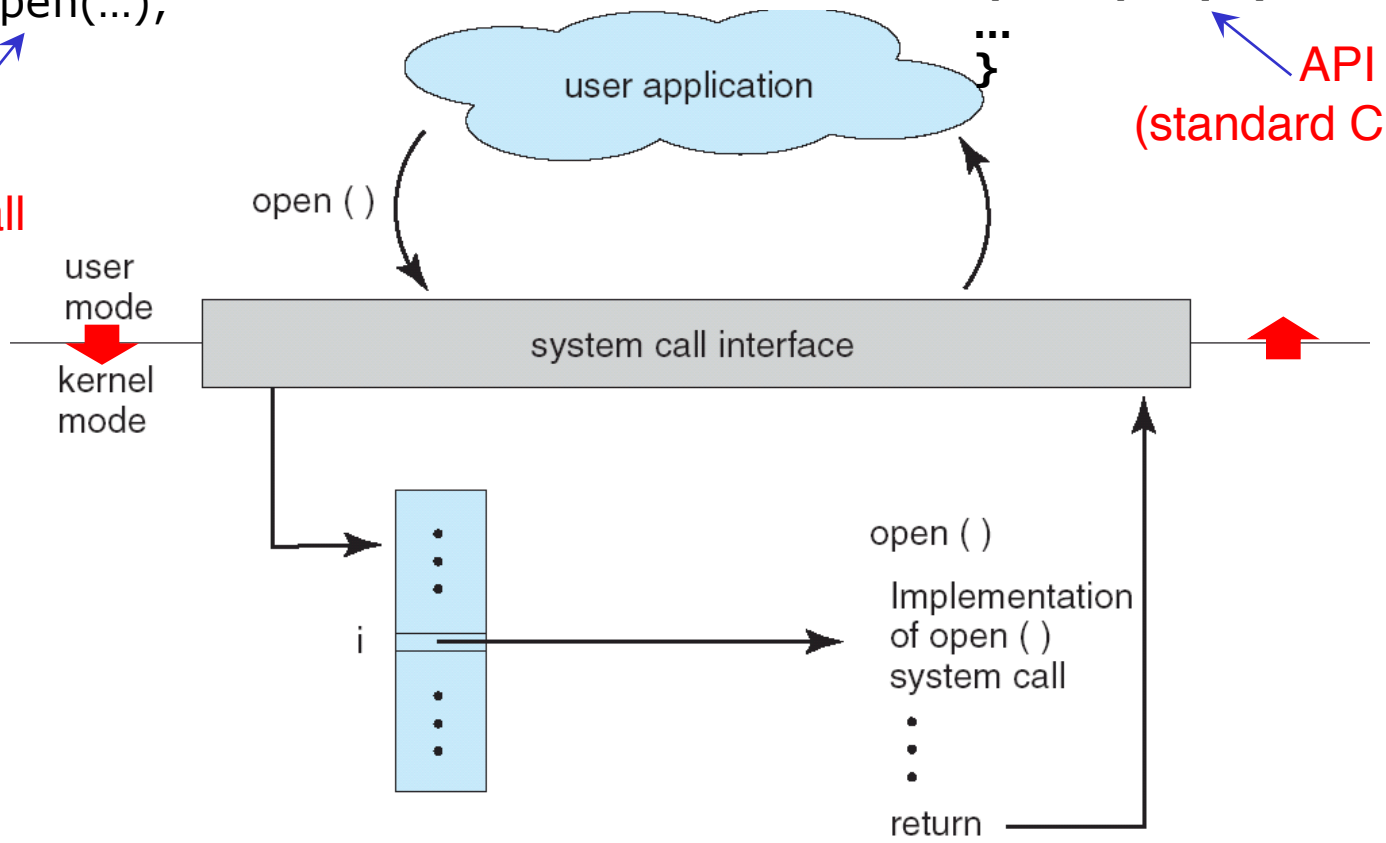
# API – System Call – OS Relationship

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

System Call

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

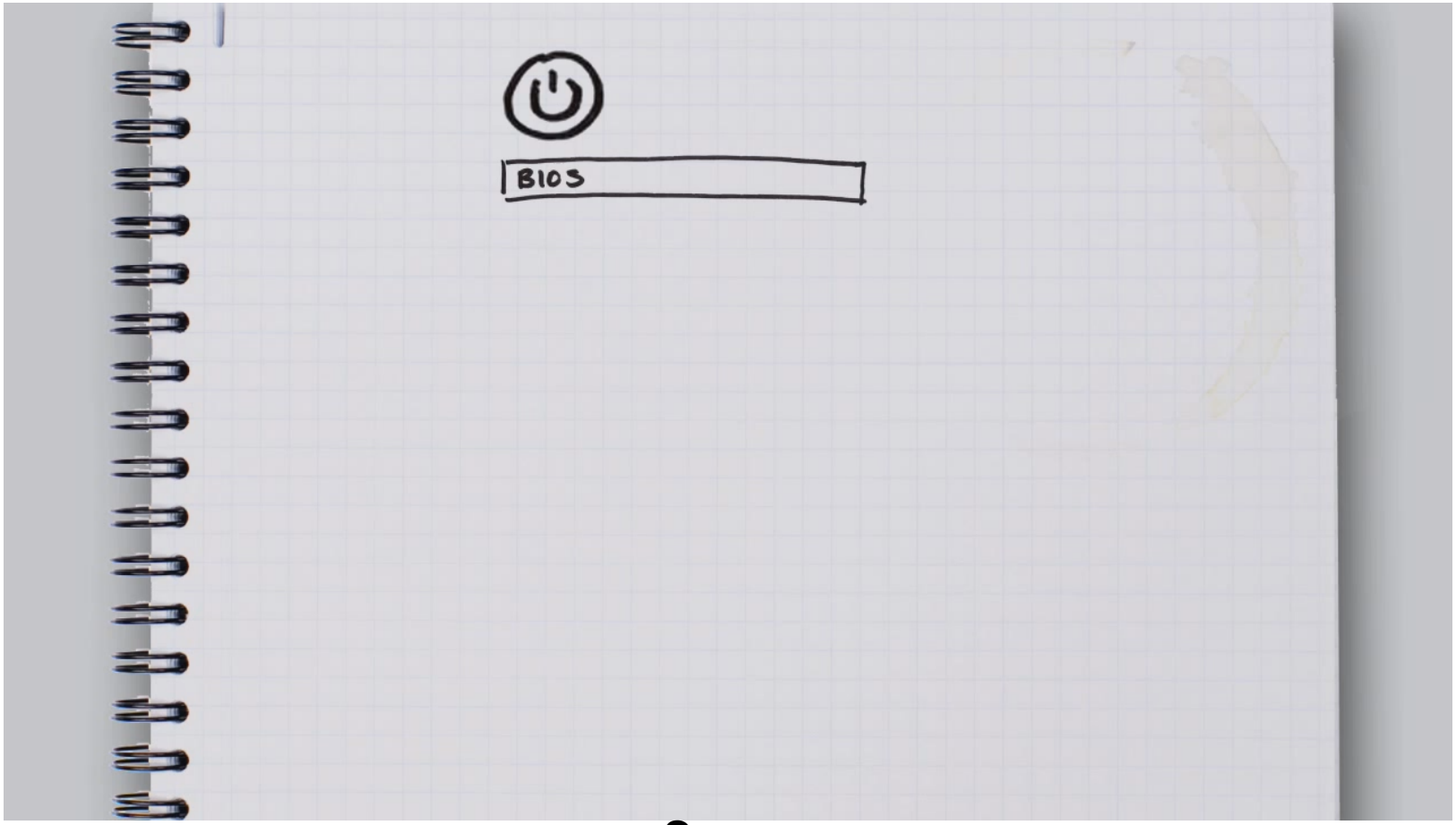
API
(standard C library)



# 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)

# New Trend in OS




**Source:**

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

# Future Operating Systems

- Operating systems are **not** limited to Windows/Linux/MacOS etc.
- Operating system development and research never stops.
  - Hardware evolvement.
  - User requirements.
- Examples:
  - Microkernel (also known as μ-kernel), https://en.wikipedia.org/wiki/Microkernel
  - Formally verified OS kernel, https://sel4.systems/