# Computer system organization

Tran, Van Hoai

Faculty of Computer Science & Engineering
HCMC University of Technology

E-mail: hoai@hcmut.edu.vn
*(partly based on slides of Le Thanh Van)*

# Outline


1 What is operating system ?

2 Computer system organization
- Computer system operation
- I/O structure
- Storage structure
- Resource protection
- Kernel data structures

# Outline

1 What is operating system ?

2 Computer system organization
- Computer system operation
- I/O structure
- Storage structure
- Resource protection
- Kernel data structures

# Basic concepts

**Operating system**
**= Operating + system**

# Basic concepts

**Operating system**
**= Operating + system**

Systems can be

- human resources
- a computer hardware organization

# Basic concepts

**Operating system**
**= Operating + system**

Systems can be

- human resources
- a computer hardware organization


An operating system?

# OS - A manager

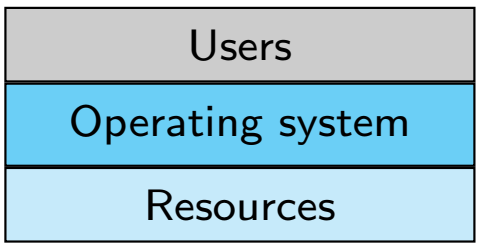
## Abstract view

Operating system is like a perfect manager which manages resources.

# OS - A manager

## Abstract view

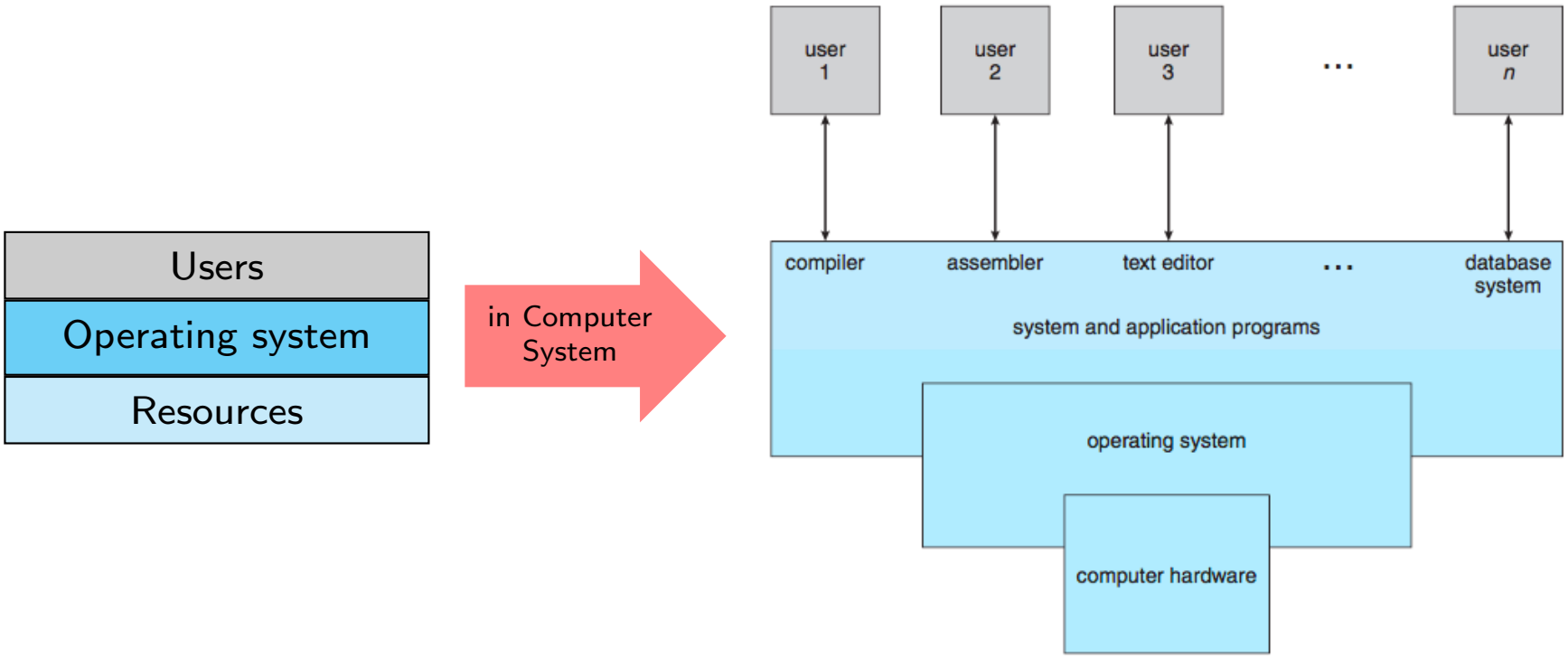
Operating system is like a perfect manager which manages resources.



# OS - A manager

## Abstract view

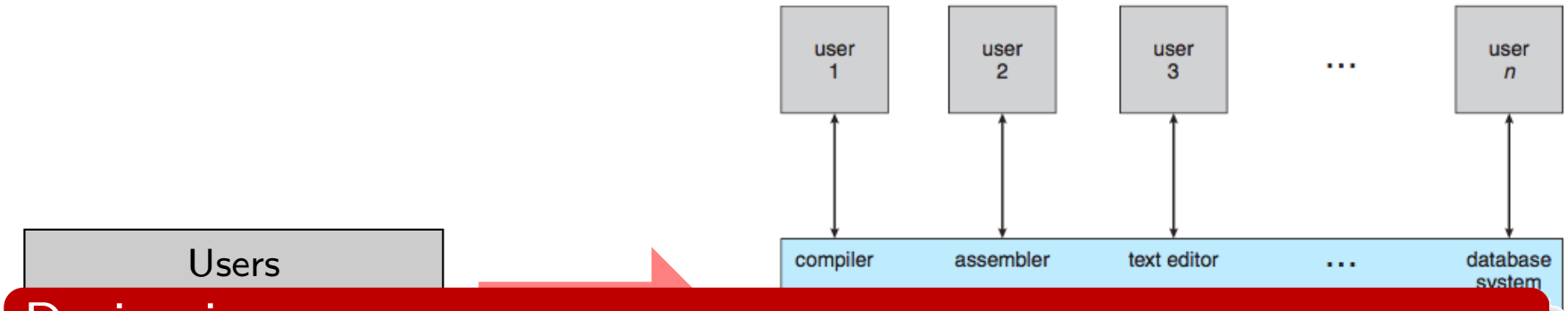
Operating system is like a perfect manager which manages resources.



# OS - A manager

## Abstract view

Operating system is like a perfect manager which manages resources.




### Design issues

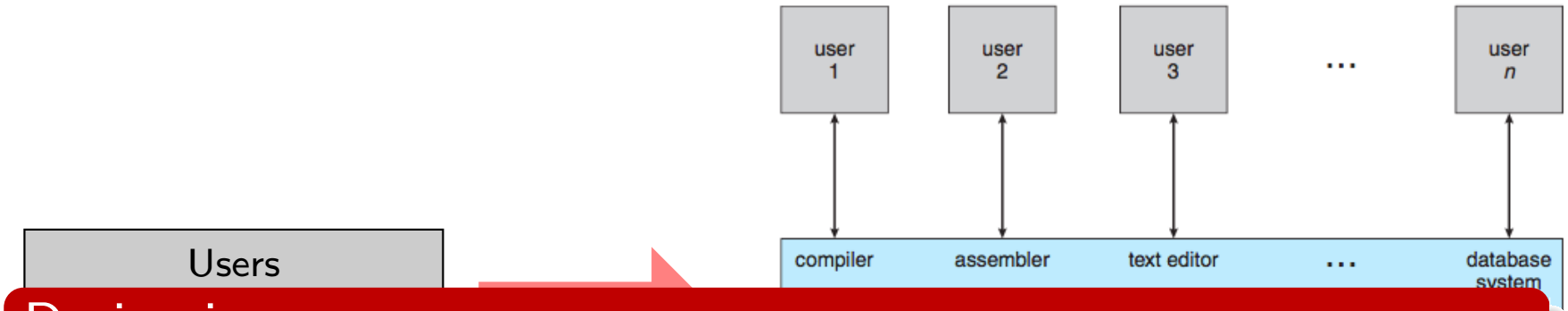
An OS is designed not only to "serve" users, but also to "serve" users well in terms of

- Convenience
- Efficiency

# OS - A manager

## Abstract view

Operating system is like a perfect manager which manages resources.




### Design issues

An OS is designed not only to "serve" users, but also to "serve" users well in terms of

- Convenience
- Efficiency

### Example: Mobile operating system

What do users expect for a mobile operating system? (Let's think!!!)

# Definition

## Operating system

- being a basis for application programs
- acting as an intermediary between the computer user and the computer hardware

# Definition

## Operating system

- being a basis for application programs
- acting as an intermediary between the computer user and the computer hardware

User view

System view

# Definition

## Operating system

- being a basis for application programs
- acting as an intermediary between the computer user and the computer hardware

User view

System view

- User view varies according to the interface
- Type of computers influences the design aspects
  - Personal computer: easy of use $>$ resource utilization
  - Computing Server: easy of use $<$ resource utilization
  - Shared Server: easy of use $\approx$ resource utilization

# Definition

## Operating system

- being a basis for application programs
- acting as an intermediary between the computer user and the computer hardware

### User view

- User view varies according to the interface
- Type of computers influences the design aspects
  - Personal computer: easy of use $>$ resource utilization
  - Computing Server: easy of use $<$ resource utilization
  - Shared Server: easy of use $\approx$ resource utilization

### System view

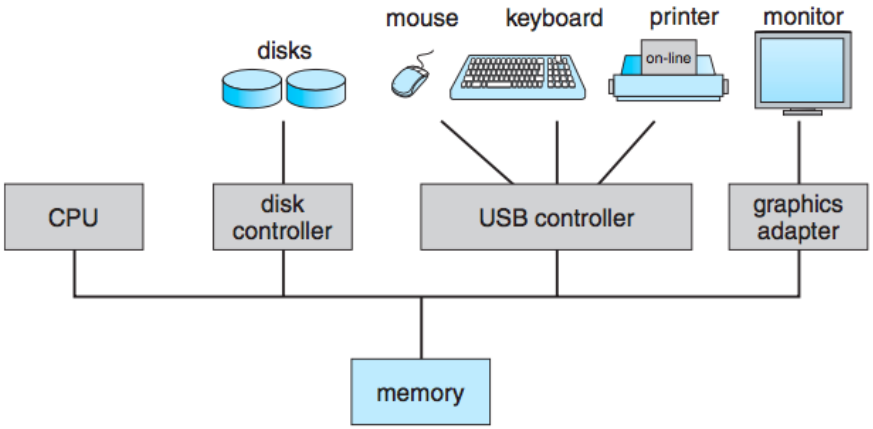
- OS as a resource allocator or a control program to manage various I/O devices and user programs
- OS = kernel + system programs

# Outline

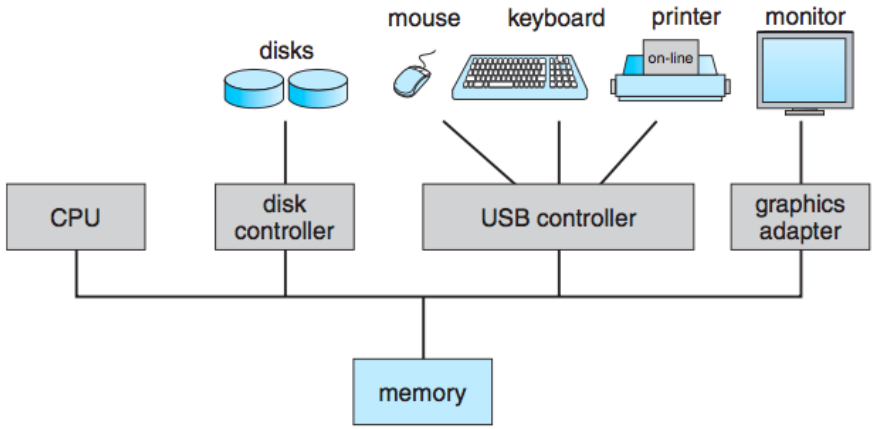
1 What is operating system ?

2 Computer system organization
- Computer system operation
- I/O structure
- Storage structure
- Resource protection
- Kernel data structures

# Modern computer system

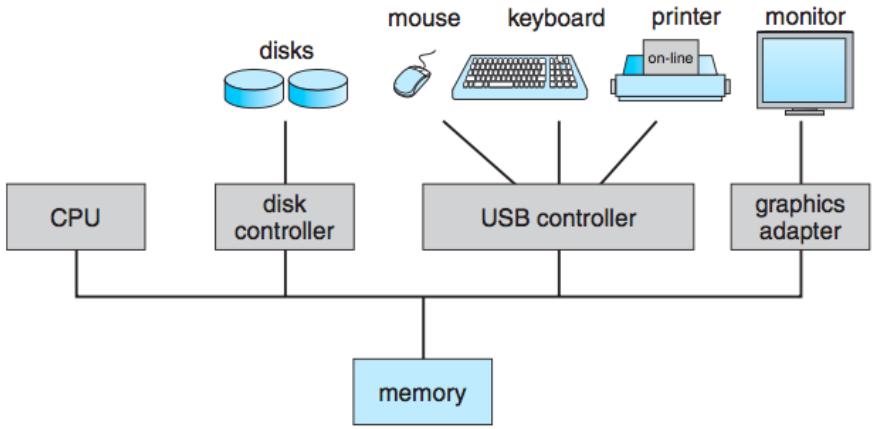


# Modern computer system




1. When computer is powered up (or rebooted): bootstrap program is an initial program (stored in ROM or EEPROM)
   - 1.1 to initialize all system aspects (registers, devices, ...)
   - 1.2 to load operating system (kernel) into memory and to start executing the system

# Modern computer system




1. When computer is powered up (or rebooted): bootstrap program is an initial program (stored in ROM or EEPROM)
   - 1.1 to initialize all system aspects (registers, devices, ...)
   - 1.2 to load operating system (kernel) into memory and to start executing the system
2. When being executed, kernel
   - 2.1 to load system programs and execute them as system daemons
   - 2.2 to wait for events

# Event detection

- More and more devices connecting to CPU: keyboard, mouse, screen, disk drives, printer, scanner, sound card, etc.
- Device occasionally need CPU service: but can't predict when
- Time between events should be busy time of CPU

# Event detection

- More and more devices connecting to CPU: keyboard, mouse, screen, disk drives, printer, scanner, sound card, etc.
- Device occasionally need CPU service: but can't predict when
- Time between events should be busy time of CPU

Need a way for CPU to find out devices requiring attention. Attentions are detected by (i) interrupt (ii) polling

# Event detection

- More and more devices connecting to CPU: keyboard, mouse, screen, disk drives, printer, scanner, sound card, etc.
- Device occasionally need CPU service: but can't predict when
- Time between events should be busy time of CPU

Need a way for CPU to find out devices requiring attention. Attentions are detected by (i) interrupt (ii) polling

**Polling**

"Polling is like picking up your phone every few seconds to see if you have a call. ..."

# Interrupt mechanism (1)

## Interrupt

An event is signaled by an interrupt from

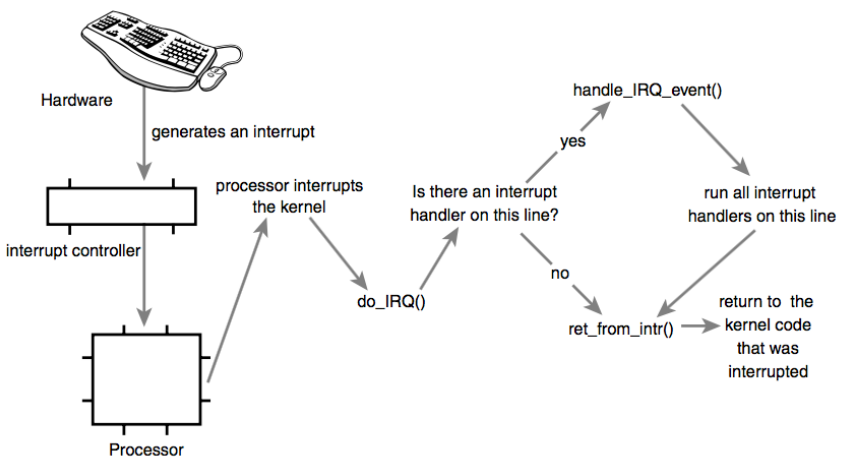
- Hardware: sending a signal to CPU, thru system bus
- Software: executing a special operation, called system call

# Interrupt mechanism (1)

## Interrupt

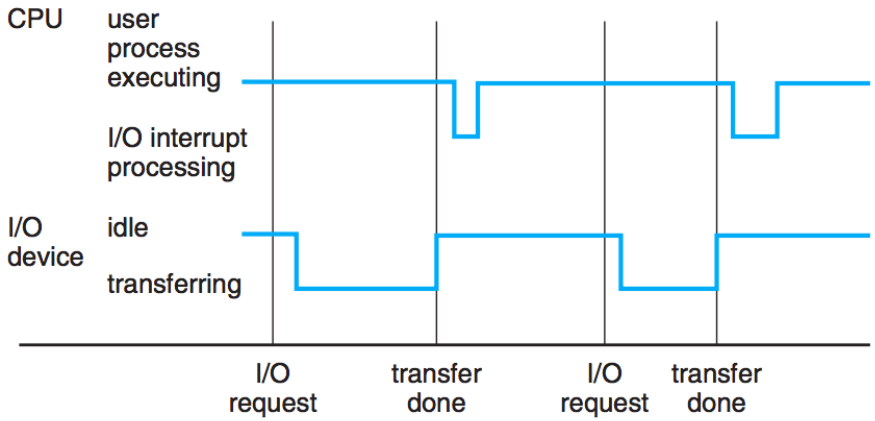
An event is signaled by an interrupt from

- Hardware: sending a signal to CPU, thru system bus
- Software: executing a special operation, called system call

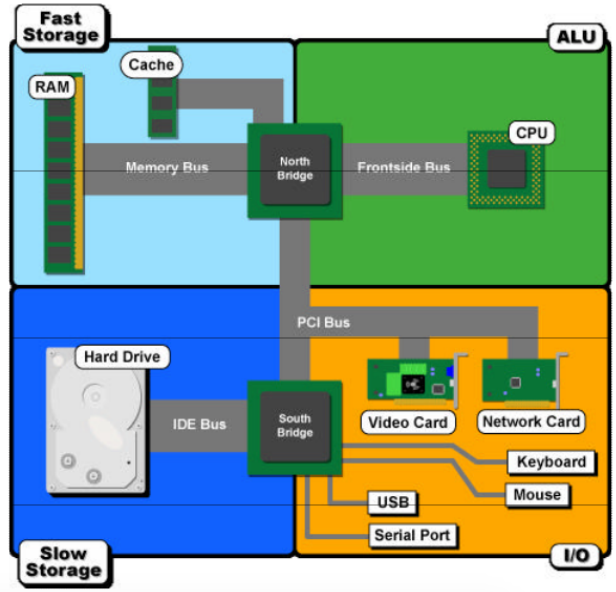



- When an interrupt occurs, a generic routine is called to check interrupt information. Return address of interrupted instructions must be stored in system stack
- Then the control is transfered to interrupt-specific handler (or interrupt handler). An interrupt vector keeps addresses to interrupt handlers

# Interrupt mechanism (2)

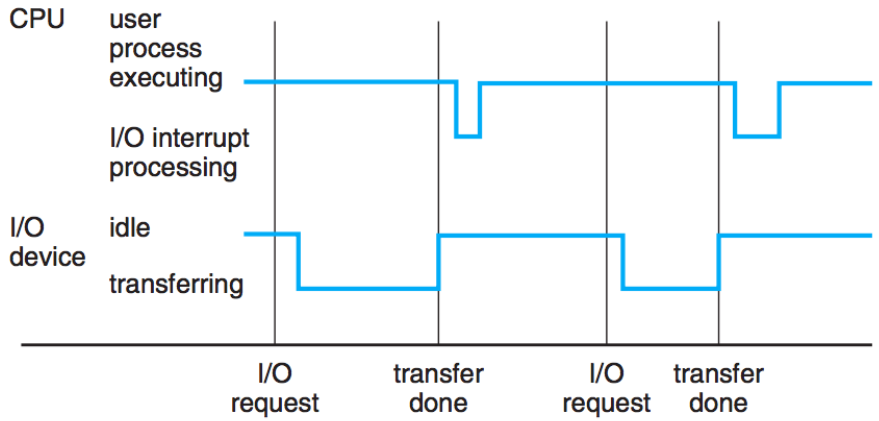



Interrupt timeline for a single process doing output

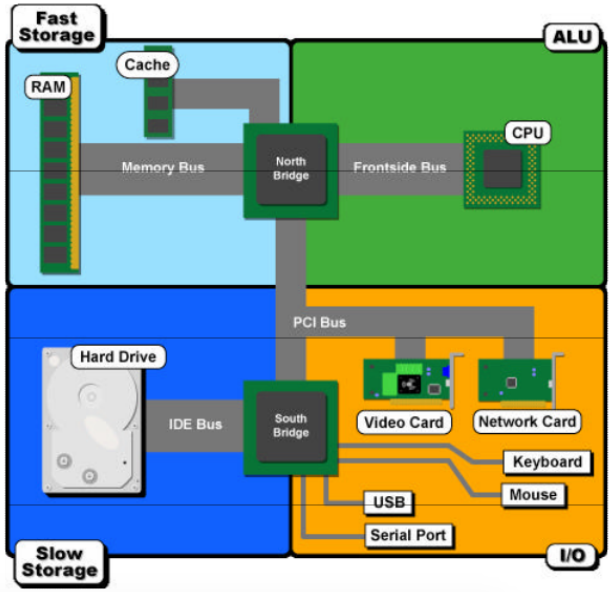



Programmable interrupt controller (PIC) $\in$ 8259A chip

# Interrupt mechanism (2)




Interrupt timeline for a single process doing output



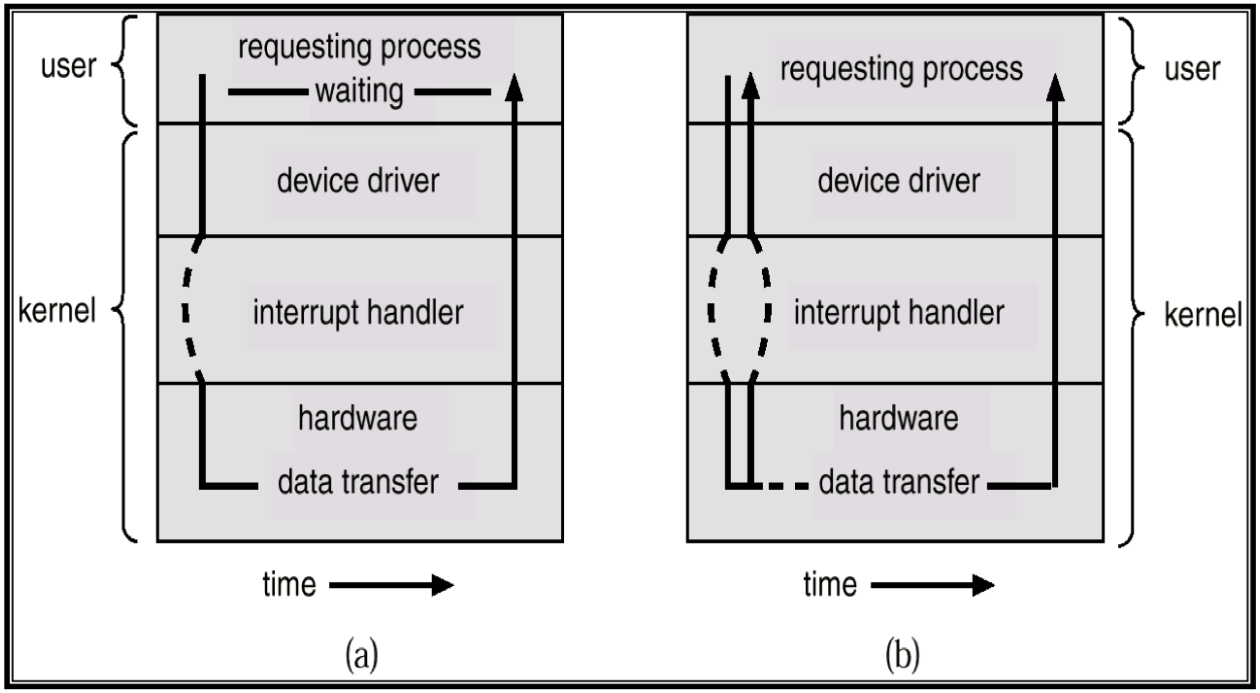

Programmable interrupt controller

(PIC) $\in$ 8259A chip

## Types of interrupts

- Program: arithmetic overflow, division by zero, invalid memory access
- Timer: CPU performs a task periodically
- I/O: finish of an I/O operation, failures in I/O operation
- Hardward failure: power failure, memory parity
- Trap (software interrupt): a system call

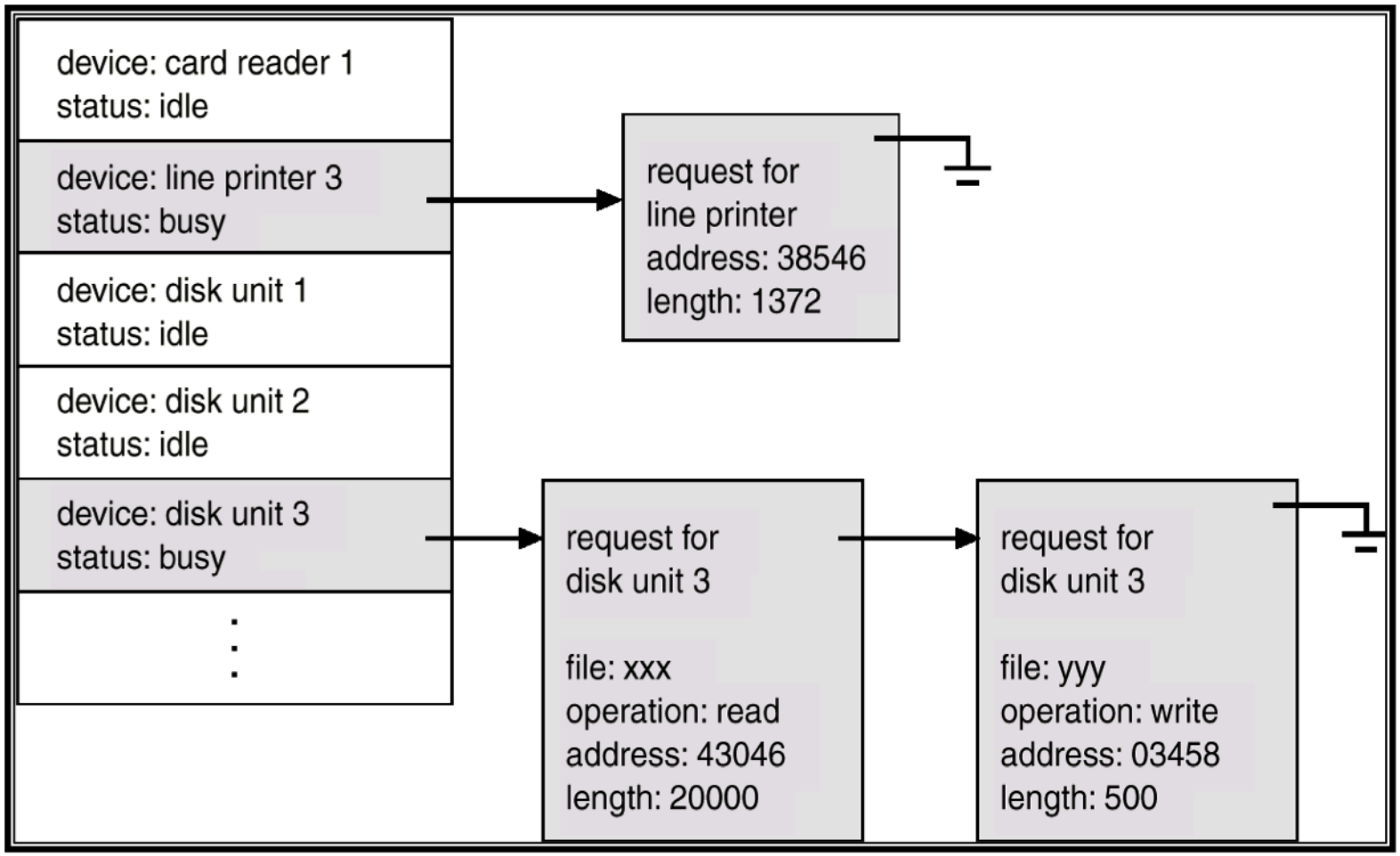
# I/O methods by interrupt




- Control returns to user program only upon I/O completion
- No simultaneous I/O processing

- Control returns to user program without waiting for I/O completion
- System call – request to the operating system to allow user to wait for I/O completion
- Device-status table contains entry for each I/O device indicating its type, address, and state
- Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt

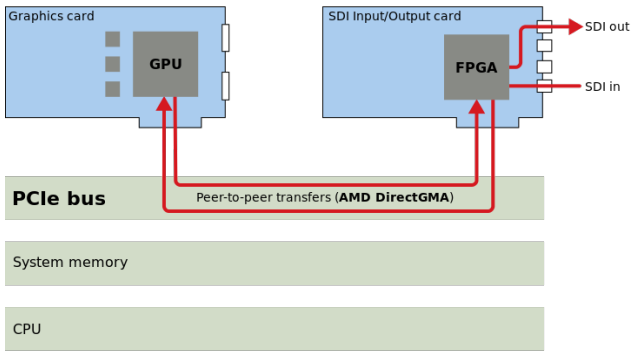
# Device status table



# Direct memory access structure

## Direct memory access

Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention

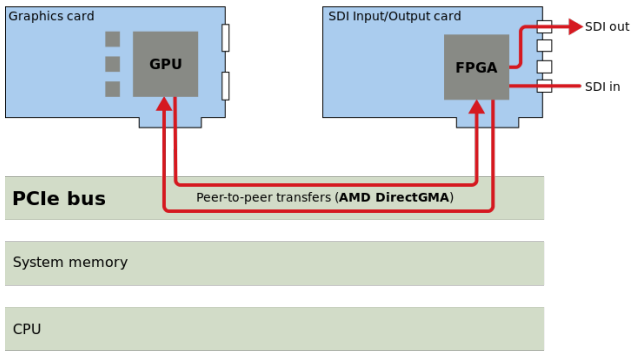



(*Source: wikipedia*)

# Direct memory access structure

## Direct memory access

Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention



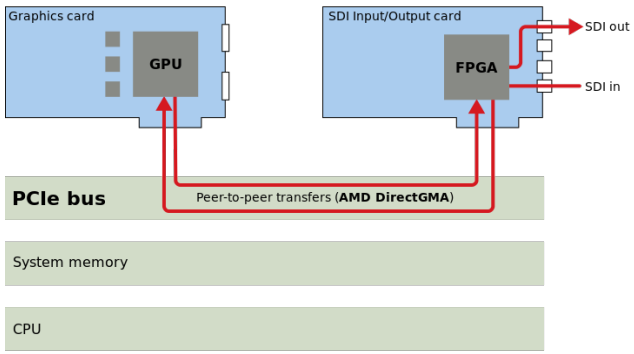

(*Source: wikipedia*)

- Used for communication at close to memory speeds
- Only interrupted per block, not per byte

# Direct memory access structure

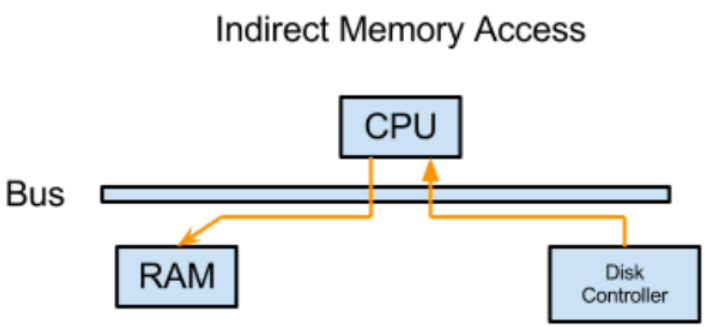
## Direct memory access

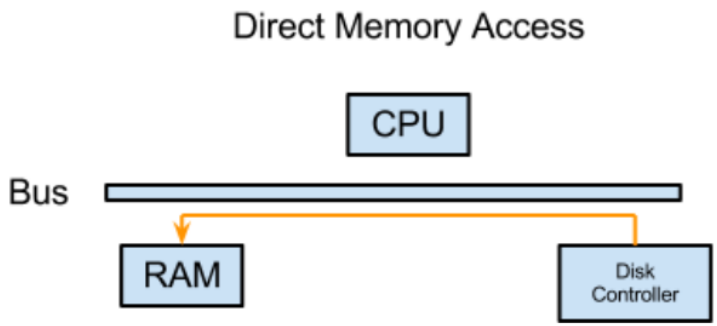
Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention




(*Source: wikipedia*)

- Used for communication at close to memory speeds
- Only interrupted per block, not per byte



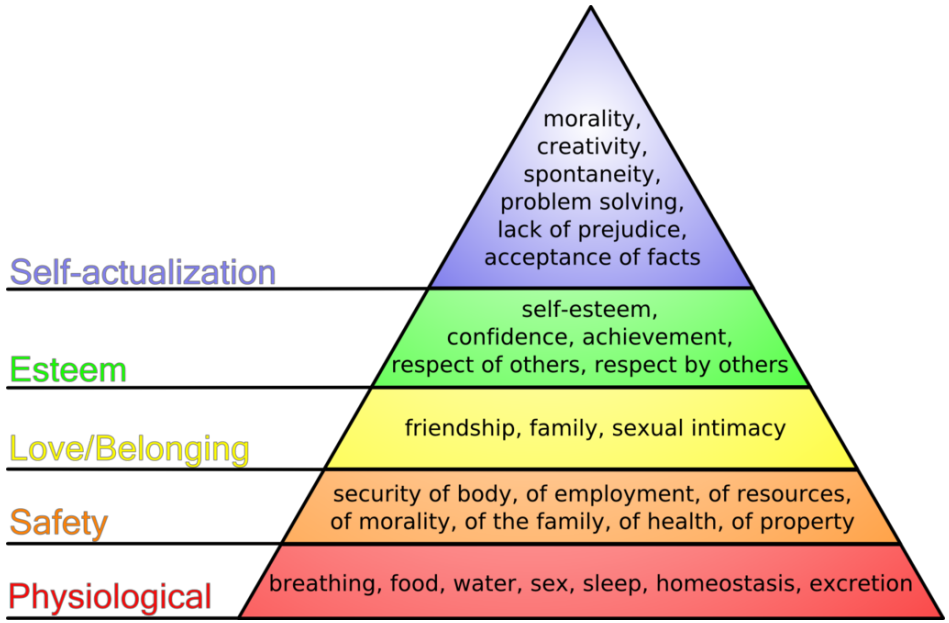




(*Source: UCLA*)

# Storage hierarchy

*"Hierarchies are celestial. In hell all are equal."*

Nicolás Gómez Dávila



# Storage hierarchy

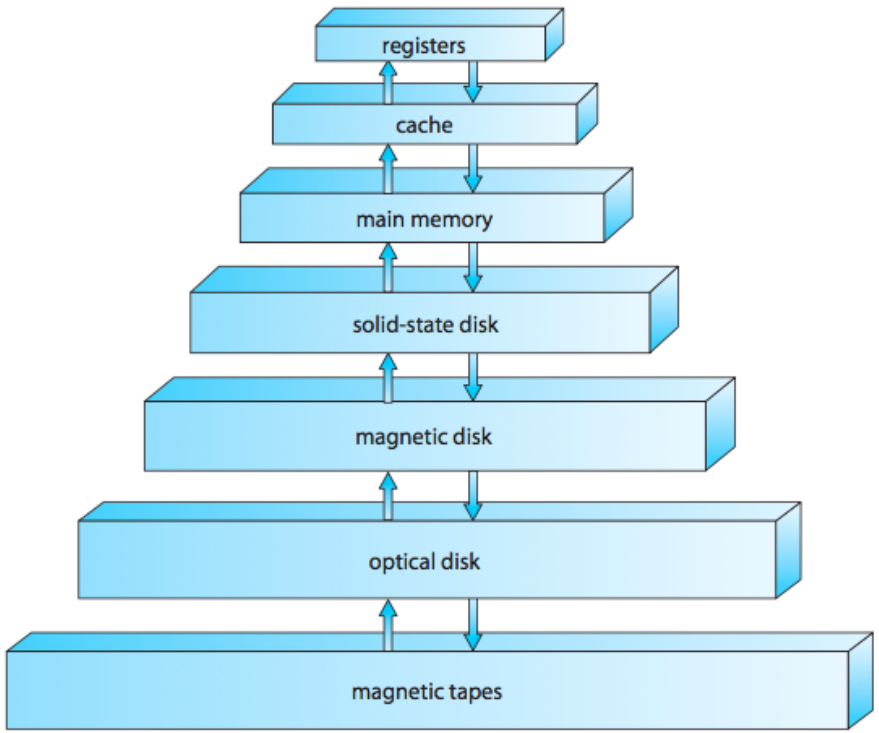
Storage systems organized in hierarchy, w.r.t.

- Speed
- Cost
- Volatility

# Storage hierarchy

Storage systems organized in hierarchy, w.r.t.

- Speed
- Cost
- Volatility




- Main memory – only large storage media that the CPU can access directly
- Secondary storage – extension of main memory that provides large non-volatile storage capacity

# Caching

**Cache**

A high-speed memory to hold recently-accessed data

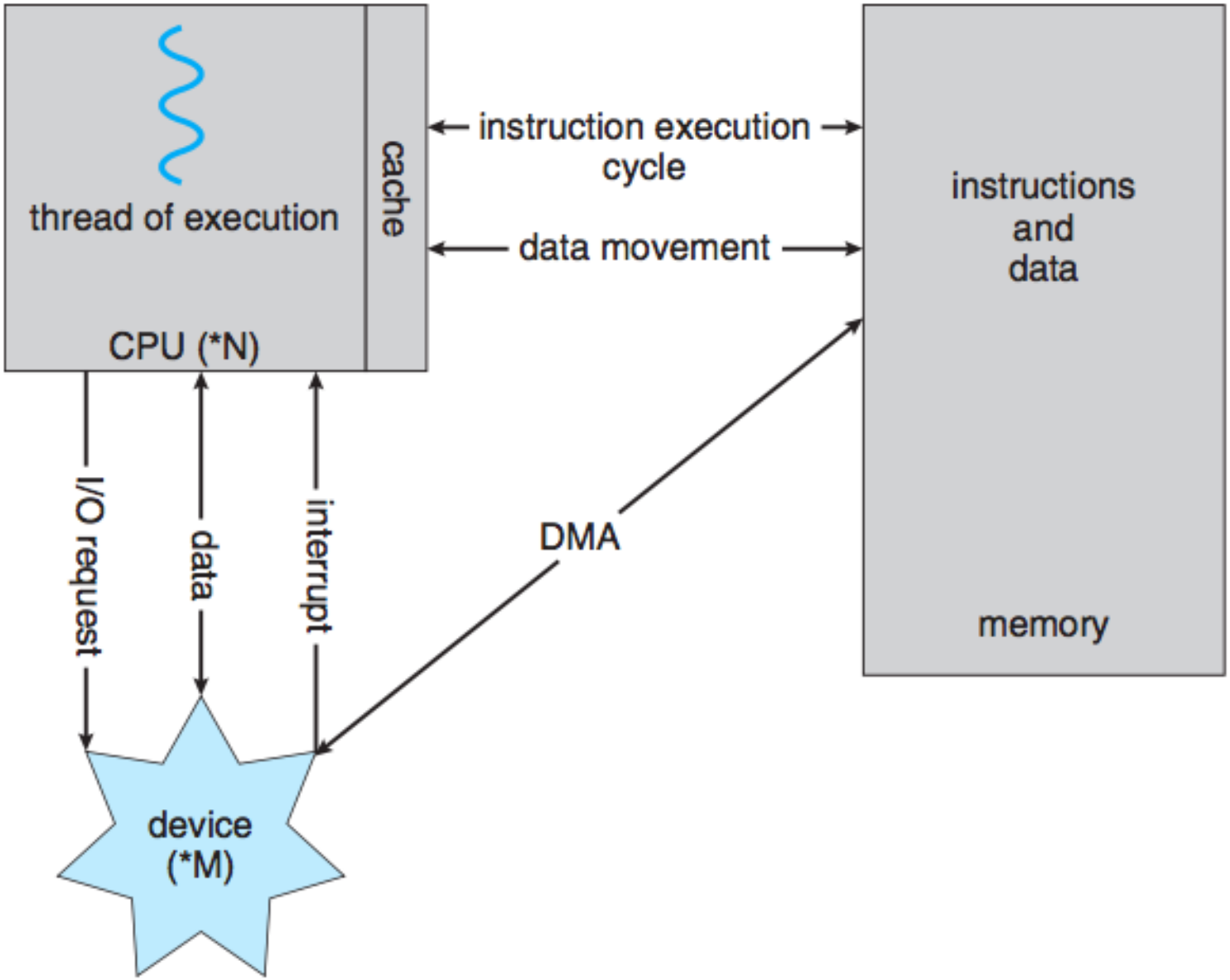
- Caching – a mechanism of copying information into faster storage system and it is controlled by a cache management policy
- Main memory can be viewed as a last cache for secondary storage
- Data is stored in more than one level, therefore, cache should be kept consistent

# Caching

## Performance of various levels of storage

| Level | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Name | registers | cache | main memory | solid state disk | magnetic disk |
| Typical size | < 1 KB | < 16MB | < 64GB | < 1 TB | < 10 TB |
| Implementation technology | custom memory with multiple ports CMOS | on-chip or off-chip CMOS SRAM | CMOS SRAM | flash memory | magnetic disk |
| Access time (ns) | 0.25 - 0.5 | 0.5 - 25 | 80 - 250 | 25,000 - 50,000 | 5,000,000 |
| Bandwidth (MB/sec) | 20,000 - 100,000 | 5,000 - 10,000 | 1,000 - 5,000 | 500 | 20 - 150 |
| Managed by | compiler | hardware | operating system | operating system | operating system |
| Backed by | cache | main memory | disk | disk | disk or tape |

# How a modern computer system works



# Hardware protection

In a multiprogrammed time-shareing system, operating system and users share hardware and software resources
$\Rightarrow$ We need a way to protect an error (of a user program) not cause problems to OS and other programs.

# Hardware protection

In a multiprogrammed time-shareing system, operating system and users share hardware and software resources
$\Rightarrow$ We need a way to protect an error (of a user program) not cause problems to OS and other programs.

- Dual-mode protection
- I/O protection
- Memory protection
- CPU protection

# Dual-mode protection

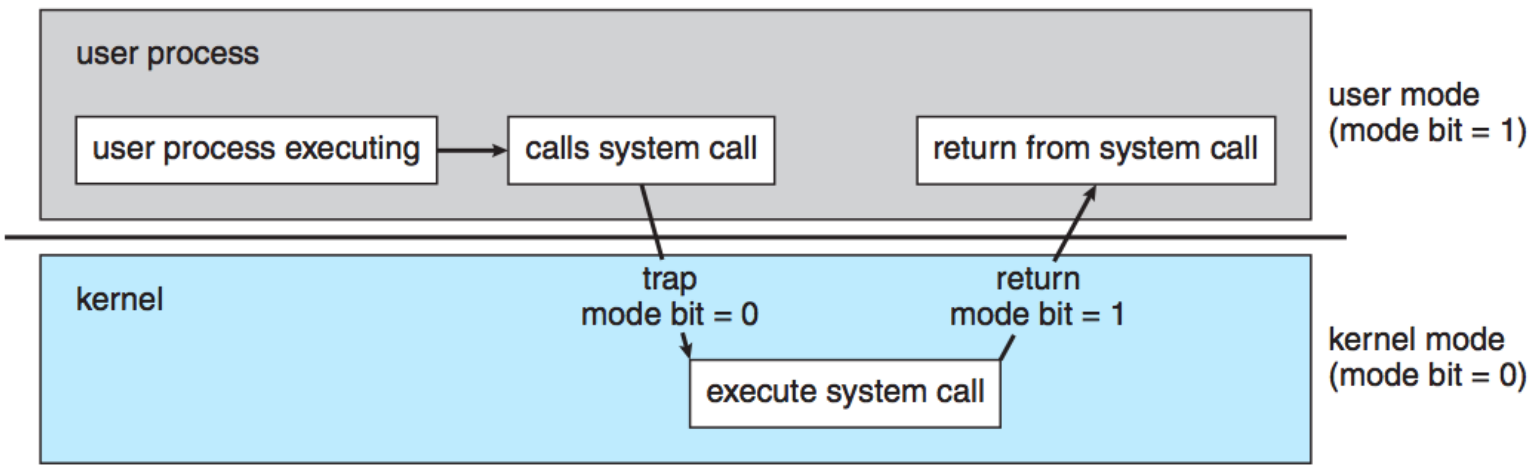
- Execution of OS code and user-defined code must be distinguished

# Dual-mode protection

- Execution of OS code and user-defined code must be distinguished
- There are at least 2 separate modes of operation:
  - User mode - execution done on behalf of a user
  - Kernel mode (also monitor, system, supervisor, privileged mode) - execution done on behalf of operating system

# Dual-mode protection

- Execution of OS code and user-defined code must be distinguished
- There are at least 2 separate modes of operation:
  - User mode - execution done on behalf of a user
  - Kernel mode (also monitor, system, supervisor, privileged mode) - execution done on behalf of operating system
- Mode bit added to computer hardware to indicate the current mode.




Privileged instructions can be issued only in kernel mode

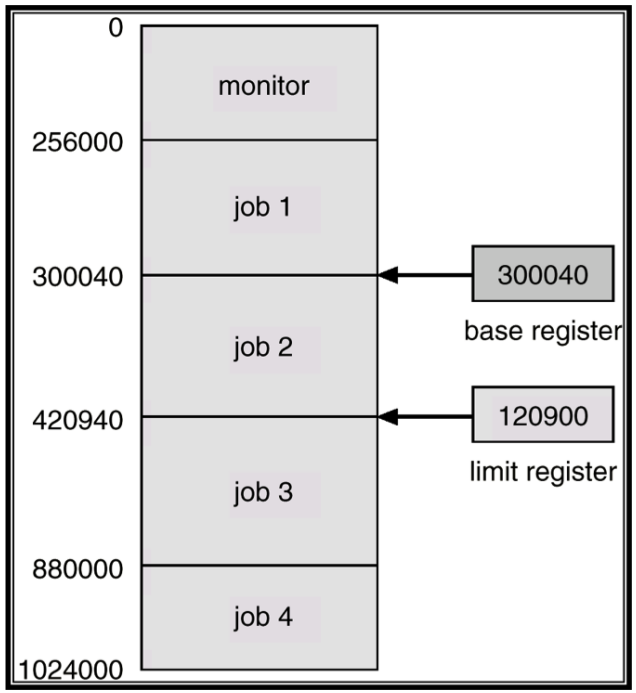
# I/O protection

- All I/O instructions are privileged instructions
- All I/O requests are done by calling system calls
- Guarantee that a user program could never gain control of the computer in kernel mode

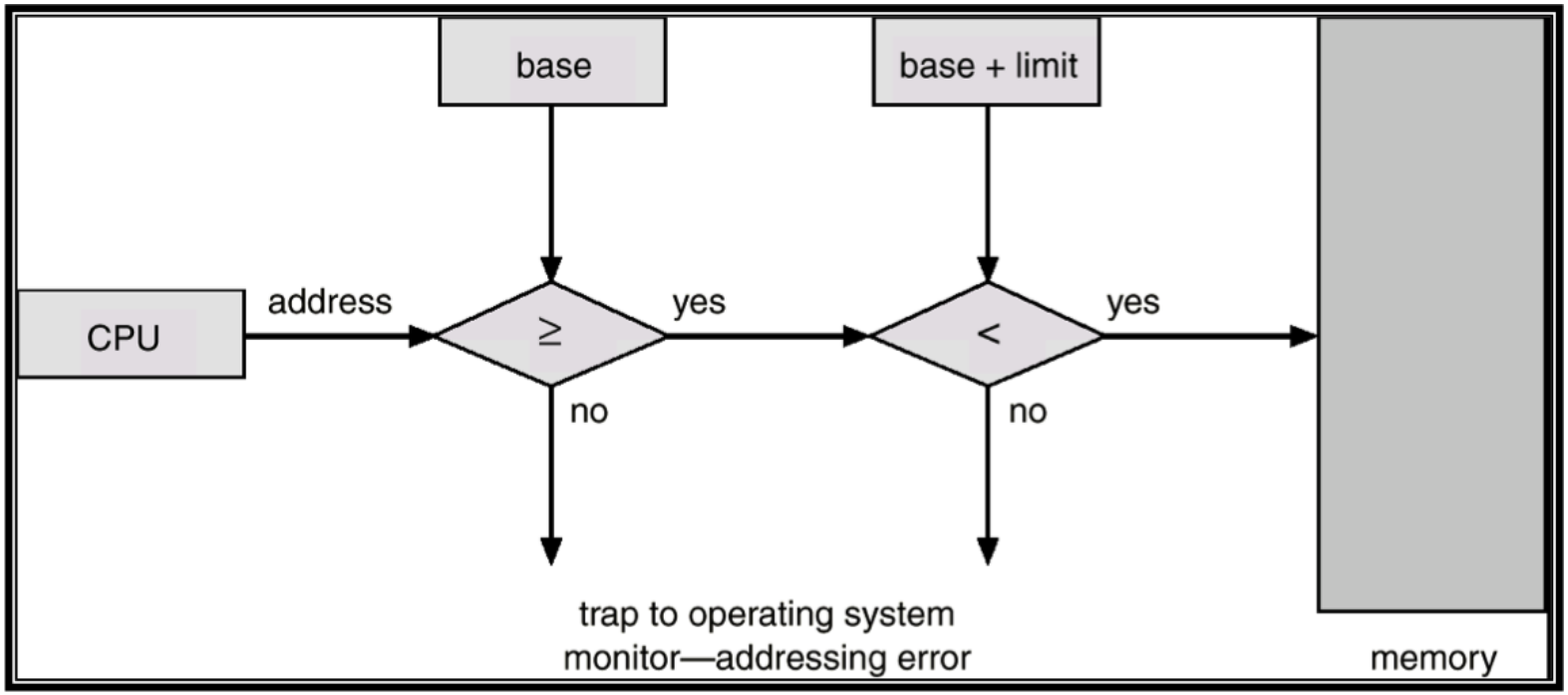
  Example: a user program that, as part of its execution, stores a new address in the interrupt vector

# Memory protection (1)

- Must provide memory protection at least for the interrupt vector and the interrupt service routines
- 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 (2)




- When executing in monitor mode, the operating system has unrestricted access to both monitor and user's memory
- The load instructions for the base and limit registers are privileged instructions

# 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
- Load-timer is a privileged instruction

# Kernel data structures

- Singly linked list, doubly-linked list, circularly linked list
- Stack, queues
- Trees
- Hash functions and Maps
- Bitmaps