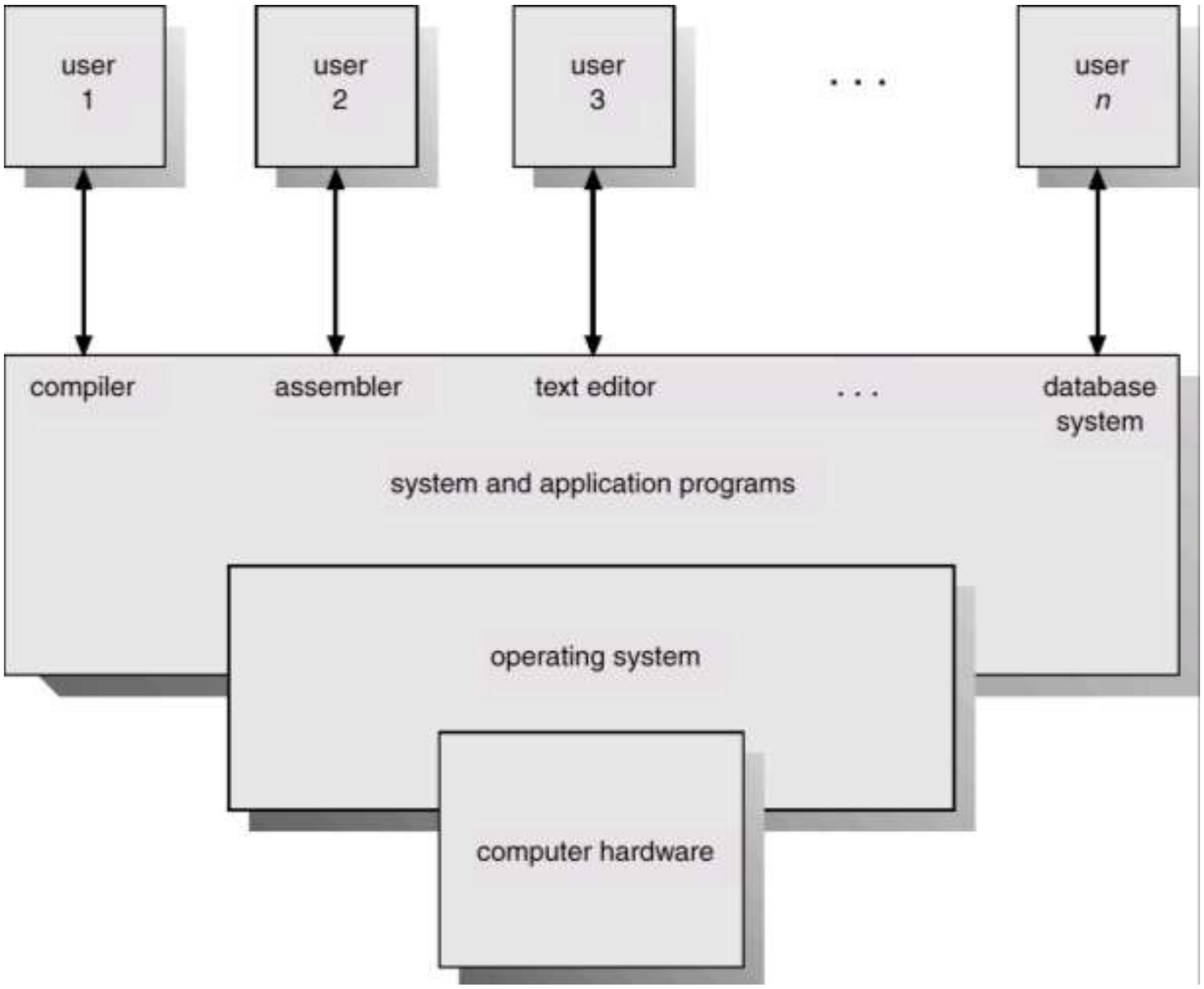
“Computer science is no more about computers than astronomy is about telescopes.“- E. Dijkstra

Edsger Wybe Dijkstra (Dutch pronunciation: [ɛtsxər ʋibə dɛikstra]); (1930–2002) was a Dutch computer scientist. He received the 1972 Turing Award for fundamental contributions to developing programming languages, and was the Schlumberger Centennial Chair of Computer Sciences at The University of Texas at Austin from 1984 until 2000.

Chapter 01: Introduction

These slides were compiled from the OSC textbook slides (Silberschatz, Galvin, and Gagne) and the instructors' class materials.

Computer System Components



Why Operating Systems?

Goals

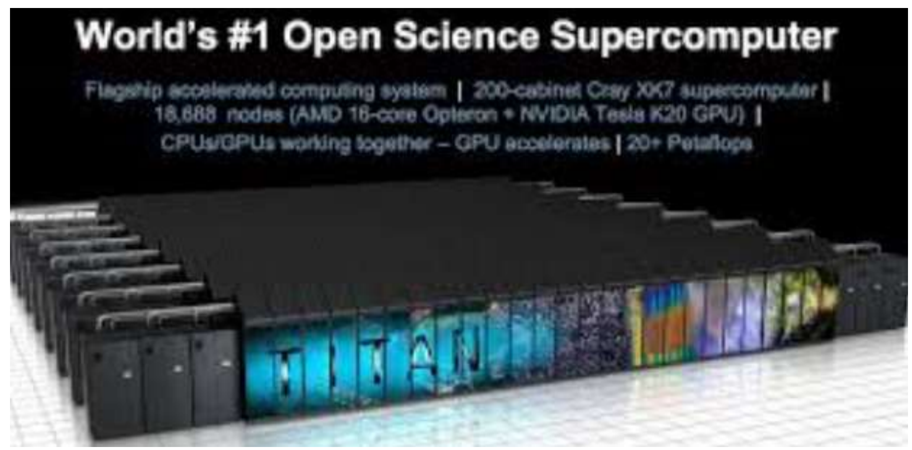
- Execute user **programs** and make solving user problems easier
- Making the computer system **convenient** to use
- Using computer **hardware** in an efficient manner

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)

Personal-Computer Systems

Generally speaking, the same OS concepts are appropriate for the various different classes of computers



Operating-System "Managements"

Protection

- Distinguishing between user and kernel

I/O Management

- Asynchronous I/Os

Process Management

- Launching, synchronizing, and terminating programs

Memory Management

- Giving each task a independent logical address space

Storage Management

- Giving logical views of disk spaces, (i.e. directories and files)

A bit of History

Batch systems

Multiprogramming

Time-sharing systems (multi-user)

PC Systems (protection!)

Symmetric Multiprocessor Architecture

Batch Systems

A job is assembled of the program, the data, and some control information (in control cards).

Programmers pass their jobs to an operator.

The operator batched together jobs.

OS transfers control from one job to another.

Each job output is sent back to the programmer.

Multiprogramming

Several jobs are kept in main memory at the same time.

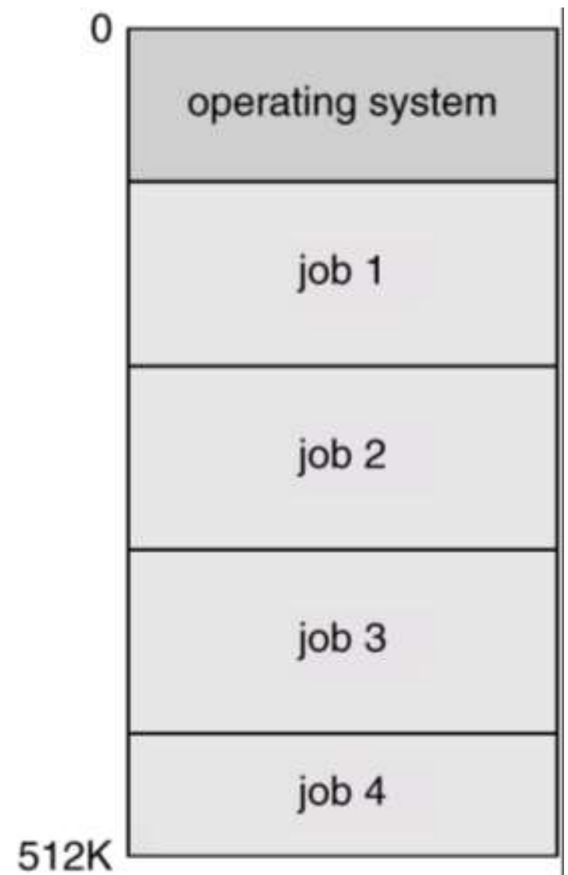
OS picks one of them to execute.

The job may have to wait for a slow I/O operation to complete.

OS switches to and executes another job.

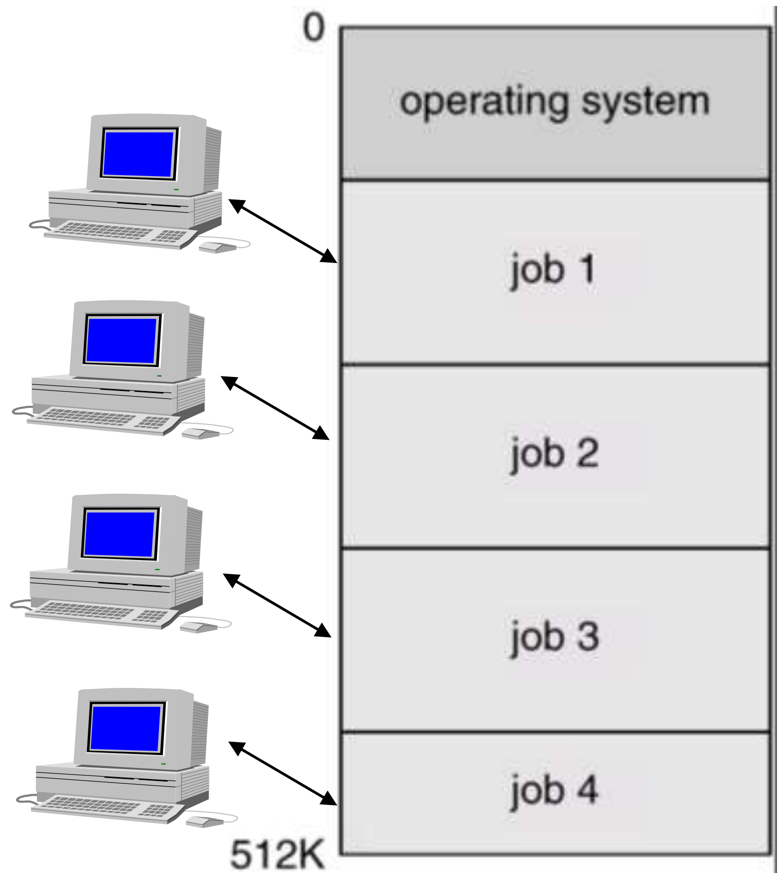
To facilitate multiprogramming, OS needs:

- Job scheduling
- Memory management



Time-Sharing Systems

- This is a logical extension of multiprogramming.
- Each user has at least one separate program in memory.
- A program in execution is referred to as a process.
- Process switch occur so frequently that the users can interact with each program while it is running.
- File system allows users to access data and program interactively.



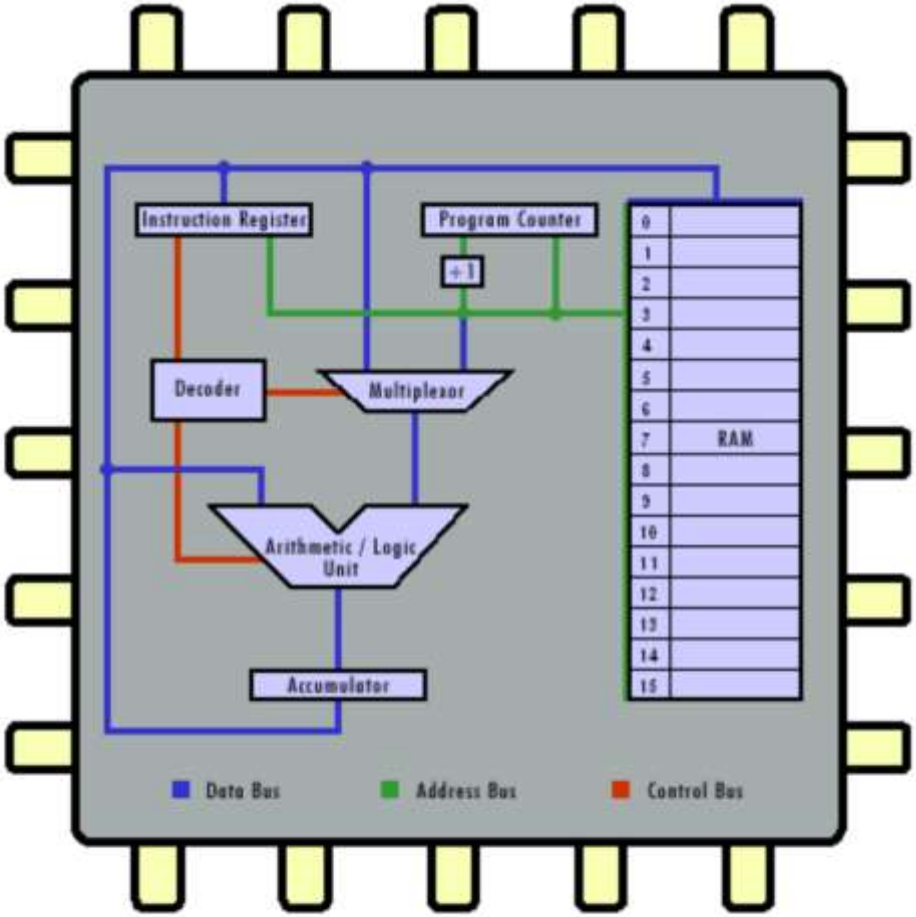
Personal-Computer Systems

- **Personal computers** – computer system dedicated to a single user.
- User **convenience** and **responsiveness**
- Can adopt technology developed for larger operating systems' some features.
- At its beginning, a single user system didn't not need advanced CPU utilization and protection.
- Later, file protection is necessary to avoid virus and bad stuff

And remember: the same OS concepts are appropriate for the various different classes of computers

A bit of hardware…

Inside the CPU



https://courses.cs.vt.edu/csonline/MachineArchitecture/Lessons/CPU/

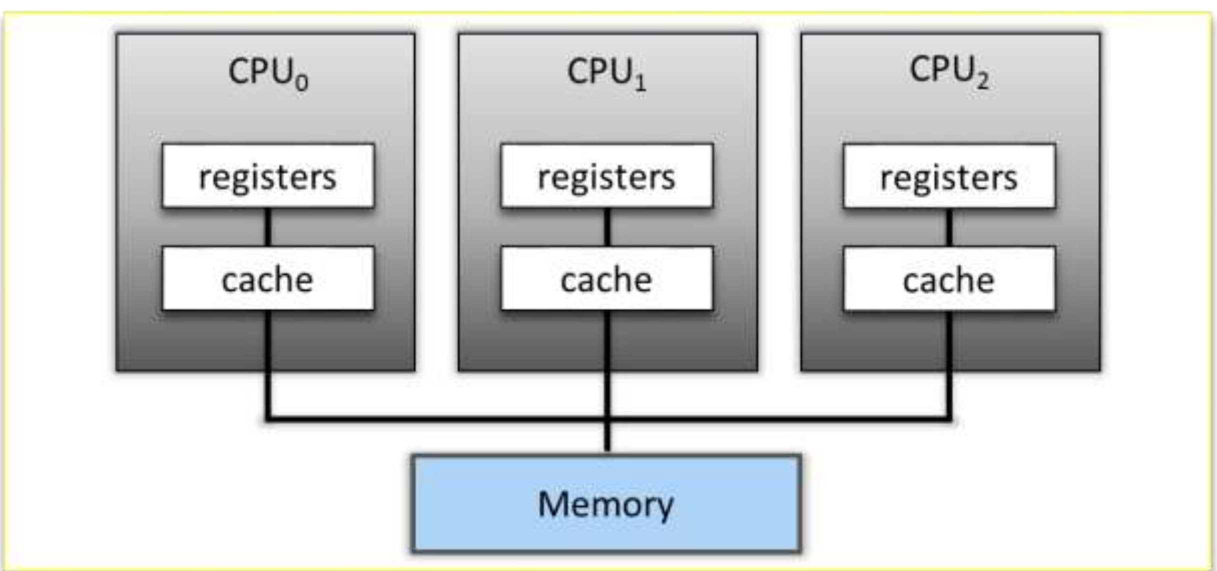
Symmetric Multiprocessing Architecture

Multiprocessor systems with more than one CPU in close communication (in one box).

Tightly coupled system – processors share memory and a clock; shared-memory-based communication.

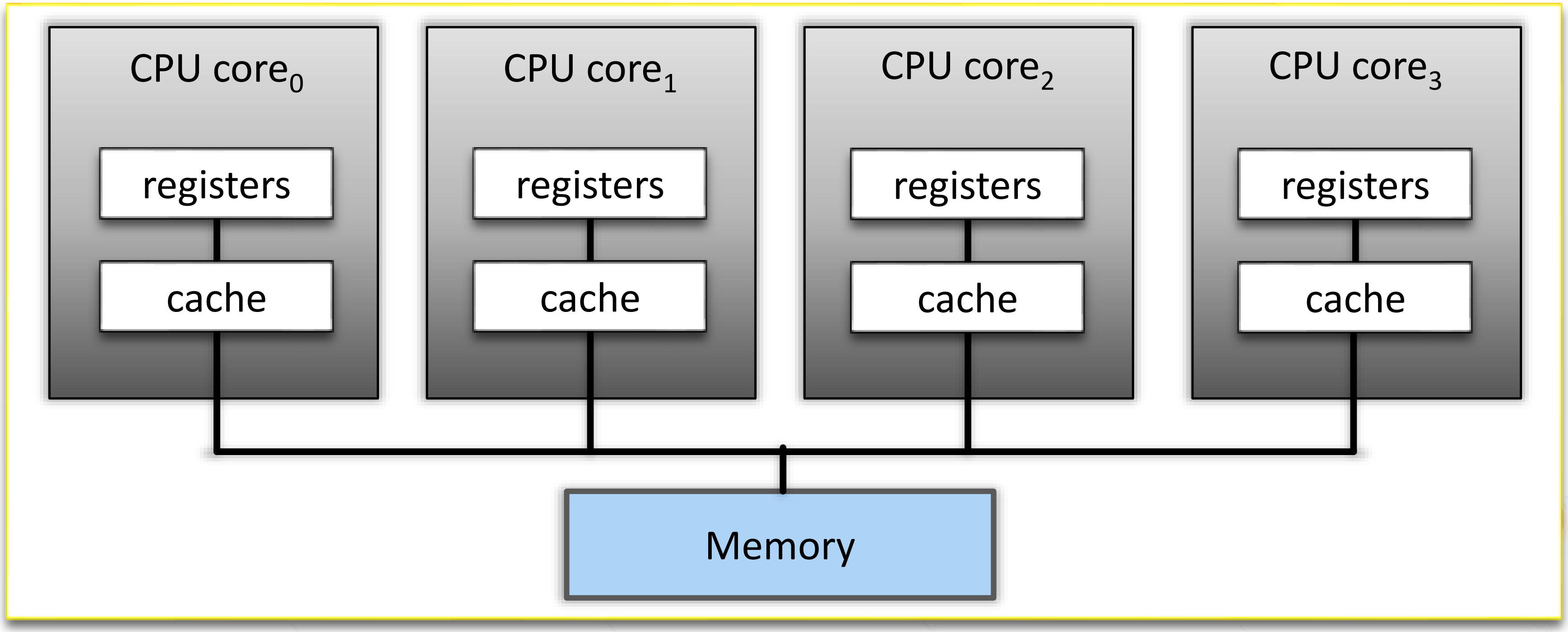
Advantages of parallel system:

- Increased *throughput*
- Economical
- Increased reliability



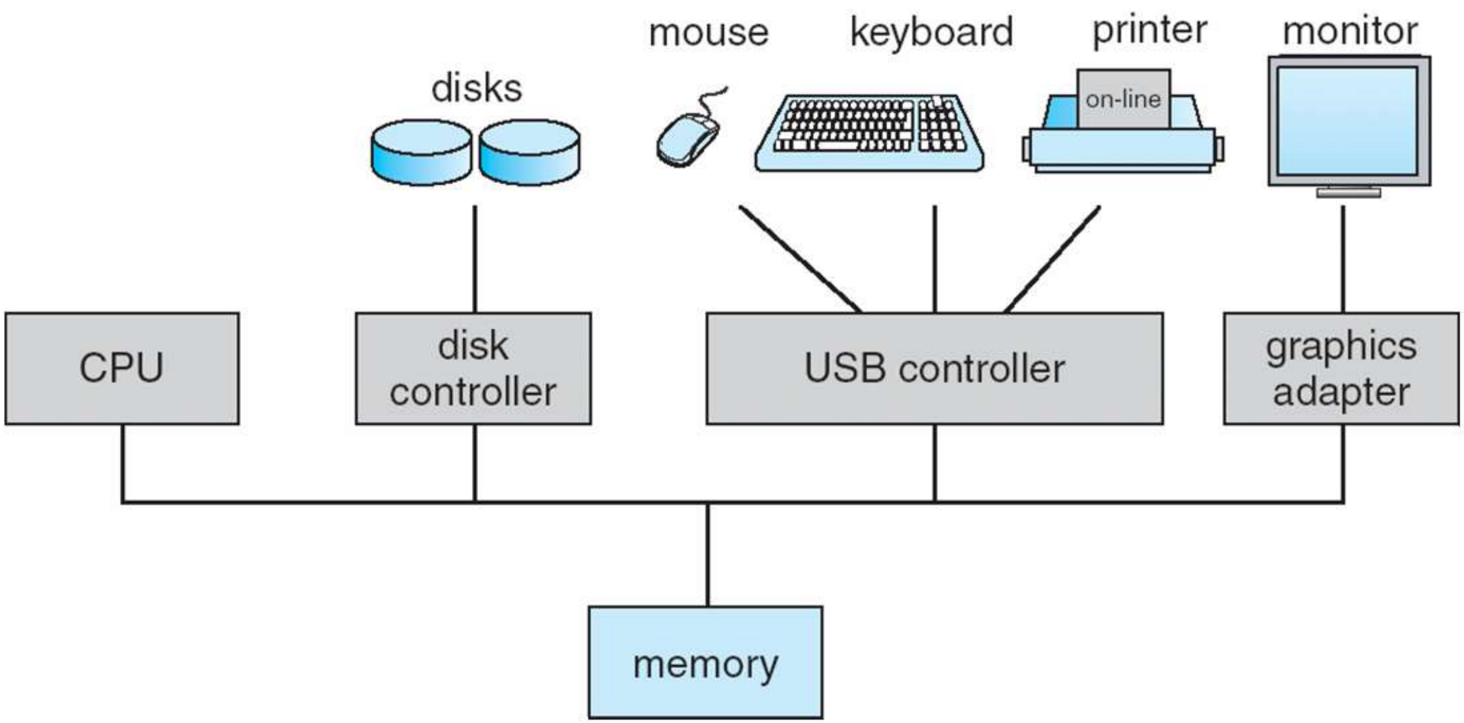
Quad-Core Design

- An MPU chip has four CPU cores, each:
  - ✓ owning its own small L1 cache,
  - ✓ sharing a large L2 cache, and
  - ✓ accessing the main memory through L2.



Computer System Organization

- Computer-system operation (hardware):
  - 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 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.

CPU moves data from/to main memory to/from local buffers.

I/O is from the device to local buffer of controller.

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

Computer-System Operation

OS Managements and Overview

I/O and interruptions

Processes

Storage (hierarchy)

Memory and DMA

Common Functions of Interrupts

- An operating system is **interrupt driven**
- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the **addresses** of all the service routines.
- Interrupt architecture must save the **address** of the interrupted instruction.
- 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.

Example of interrupt

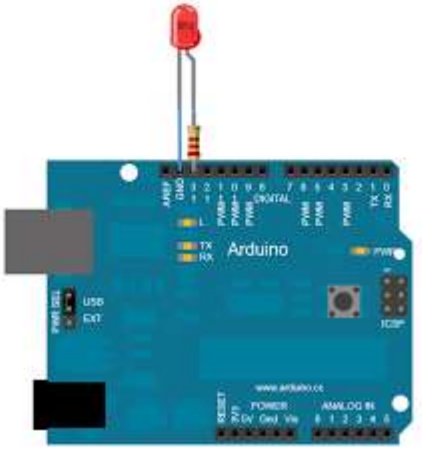
Using a very simple Arduino example

```
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte state = LOW;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
}

void loop() {
  digitalWrite(ledPin, state);
}

void blink() {
  state = !state;
}
```



Interrupt Handling

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

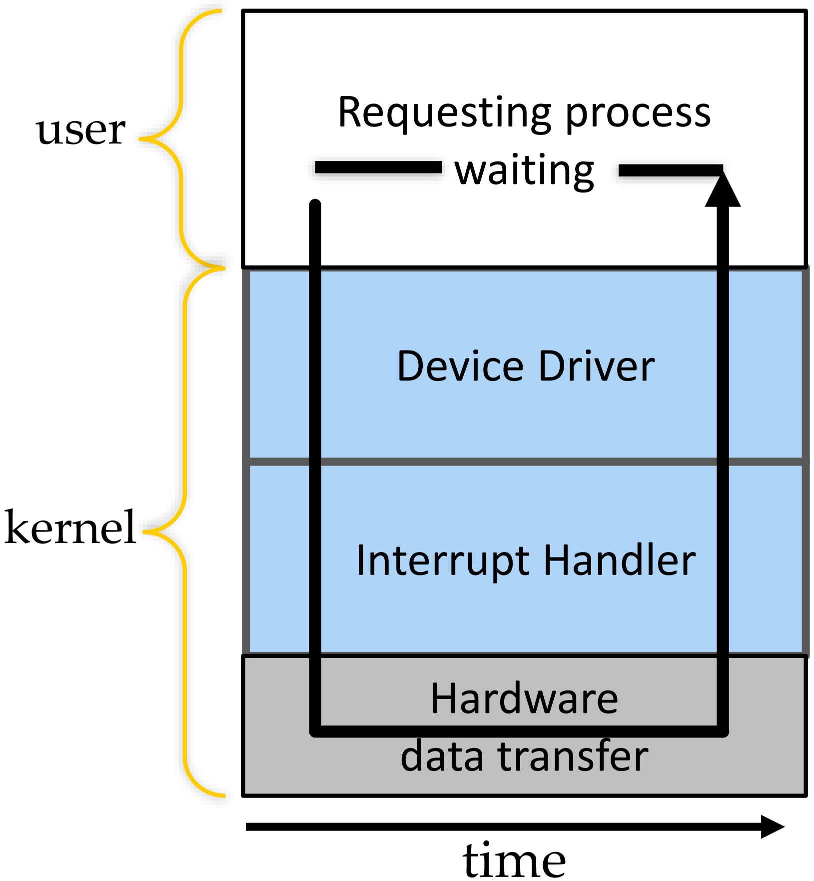
Determines which type of interrupt has occurred:

- polling
- vectored interrupt system

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

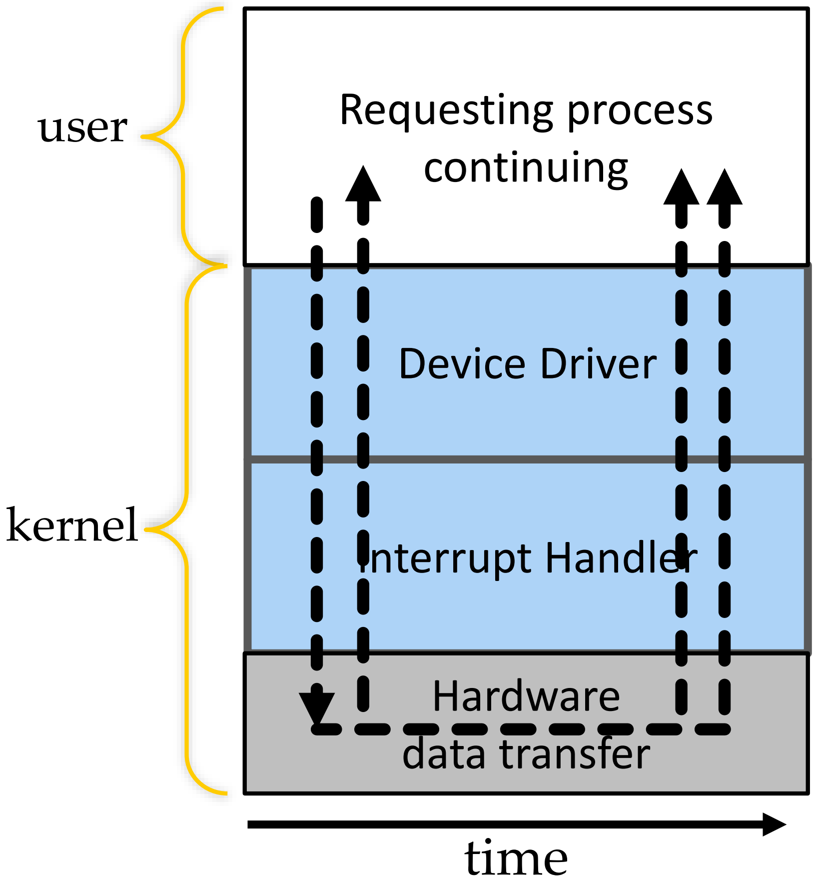
Synchronous vs. Asynchronous

Synchronous I/O



- During execution, each program needs I/O operations to receive *keyboard inputs, open files, and print out results.*
- In the early computer era, a program had to wait for an I/O operation to be completed. (Synchronous I/O)
- This frequently causes CPU idle.

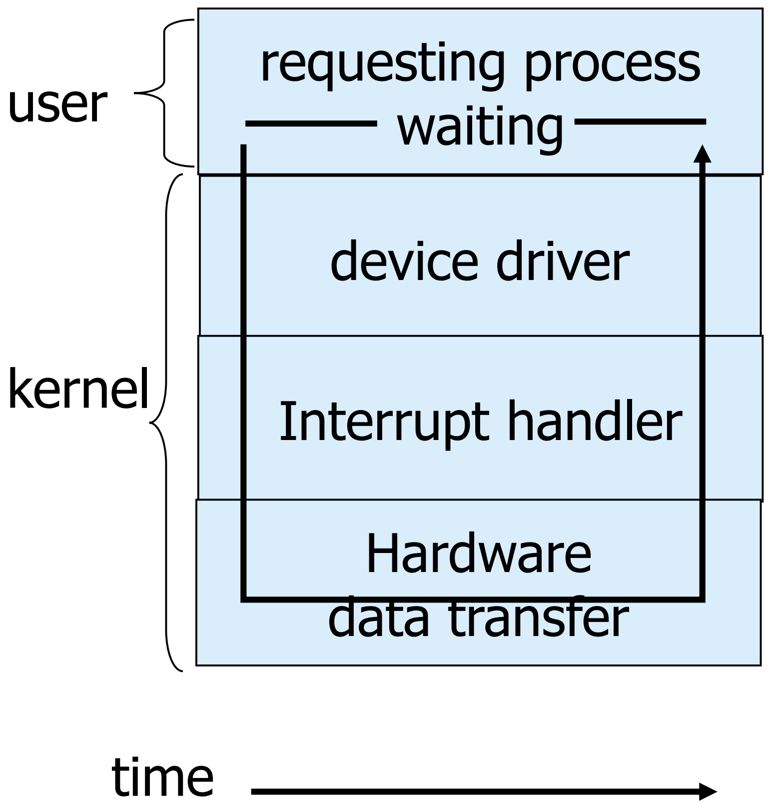
Async I/O and Interrupts

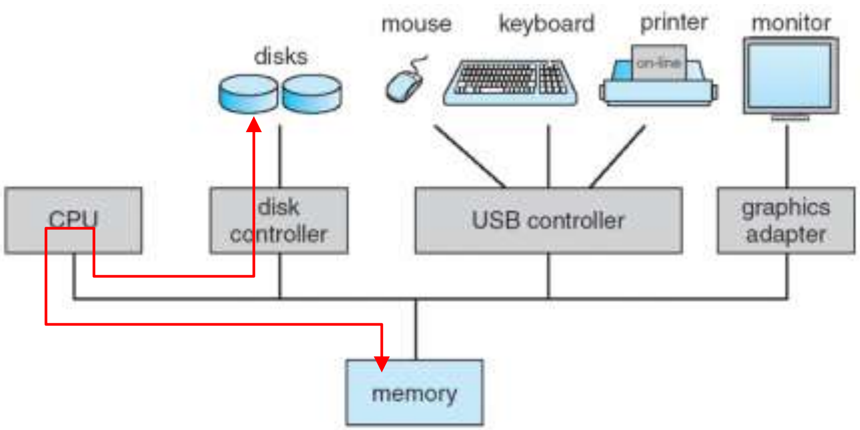


- Asynchronous I/O returns control to a user program without waiting for the I/O to complete.

- When the I/O is completed, an interrupt occurs to CPU that temporarily suspends the user program and handles the I/O device.

I/O Management



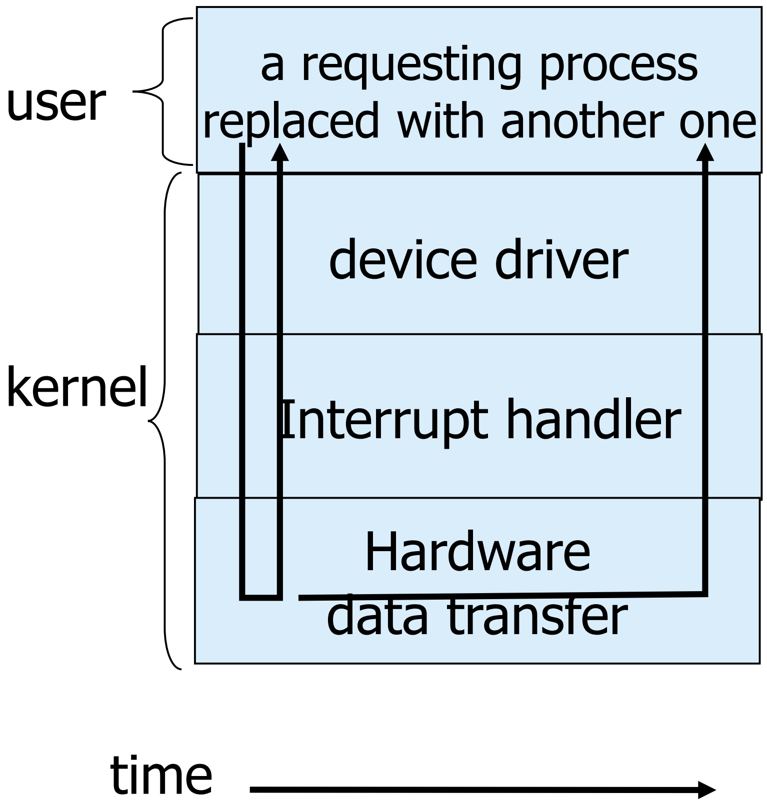


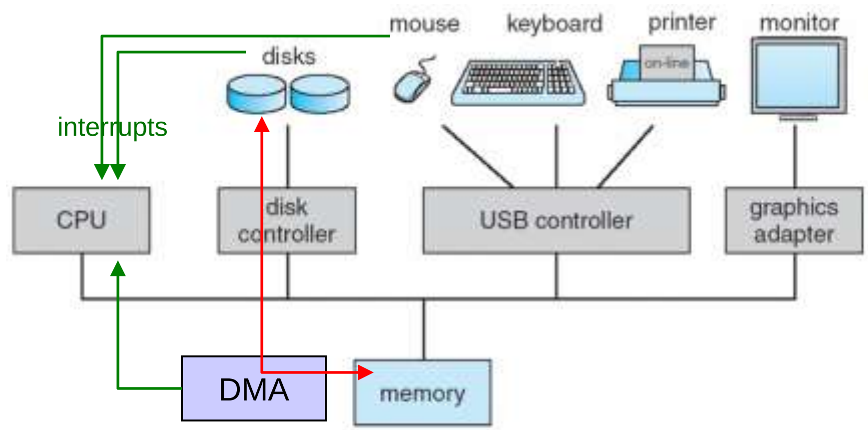
Faster CPU/Memory versus Slower I/Os

Synchronous I/O or CPU I/O

- CPU takes care of All I/O operations.
- Polling wastes CPU time.

I/O Management (cont.)





- Device interrupts and DMA
- Asynchronous I/O
  - DMA takes care of I/O data transfers.
  - Devices interrupt CPU when they are ready.

OS Managements and Overview

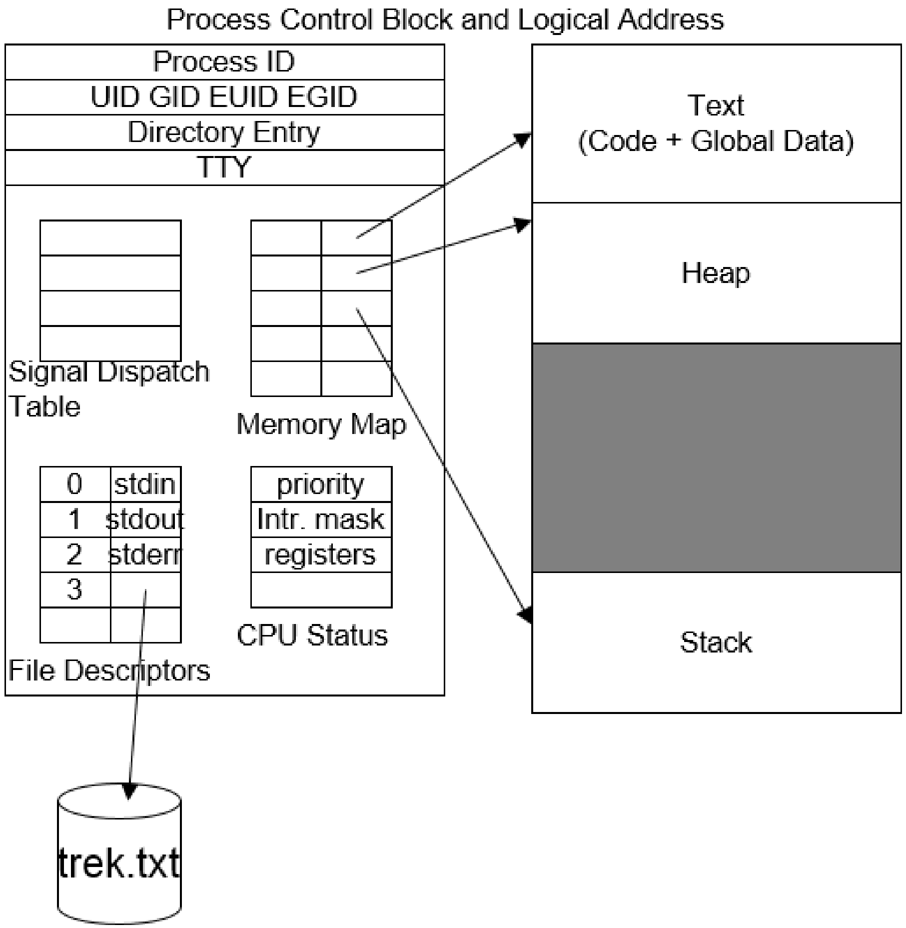
I/O and interruptions

Processes

Storage (hierarchy)

Memory and DMA

Process Management



A program in execution

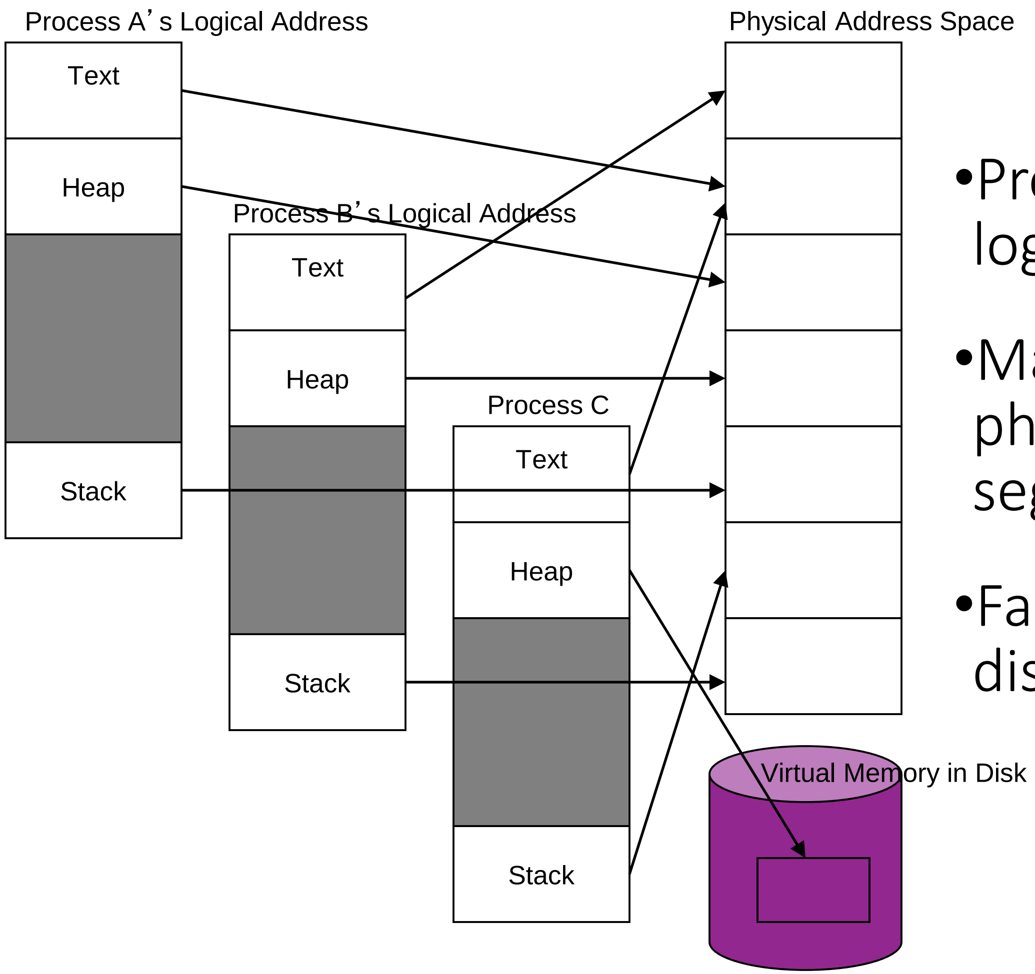
Process needs:

- CPU, memory, Files, I/Os etc.

OS must supports:

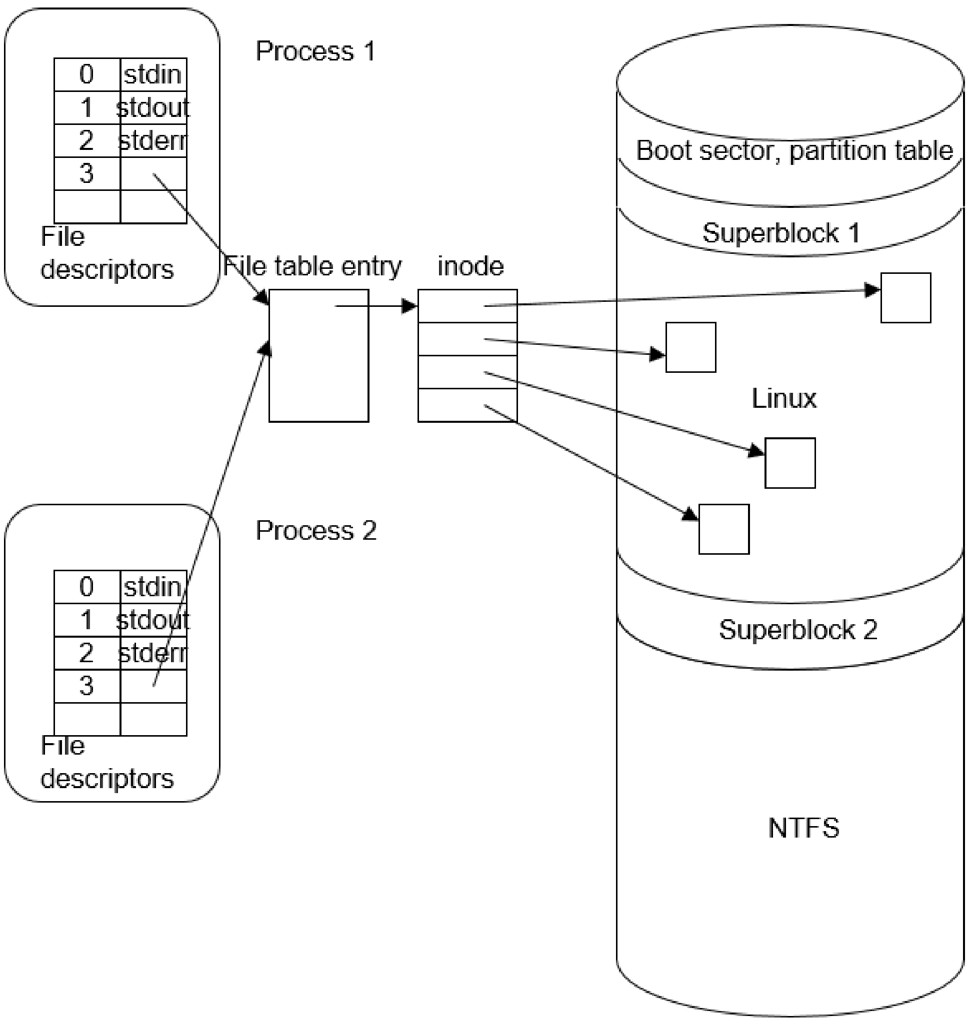
- Creation: loading it in memory
- Synchronization/communication:
  - having a process wait for another.
- Termination: clean it up from memory

Memory Management



- Providing each process with a logical address space
- Mapping logical address to physical address (paging and segmentation)
- Facilitating virtual memory using disk

Storage Management

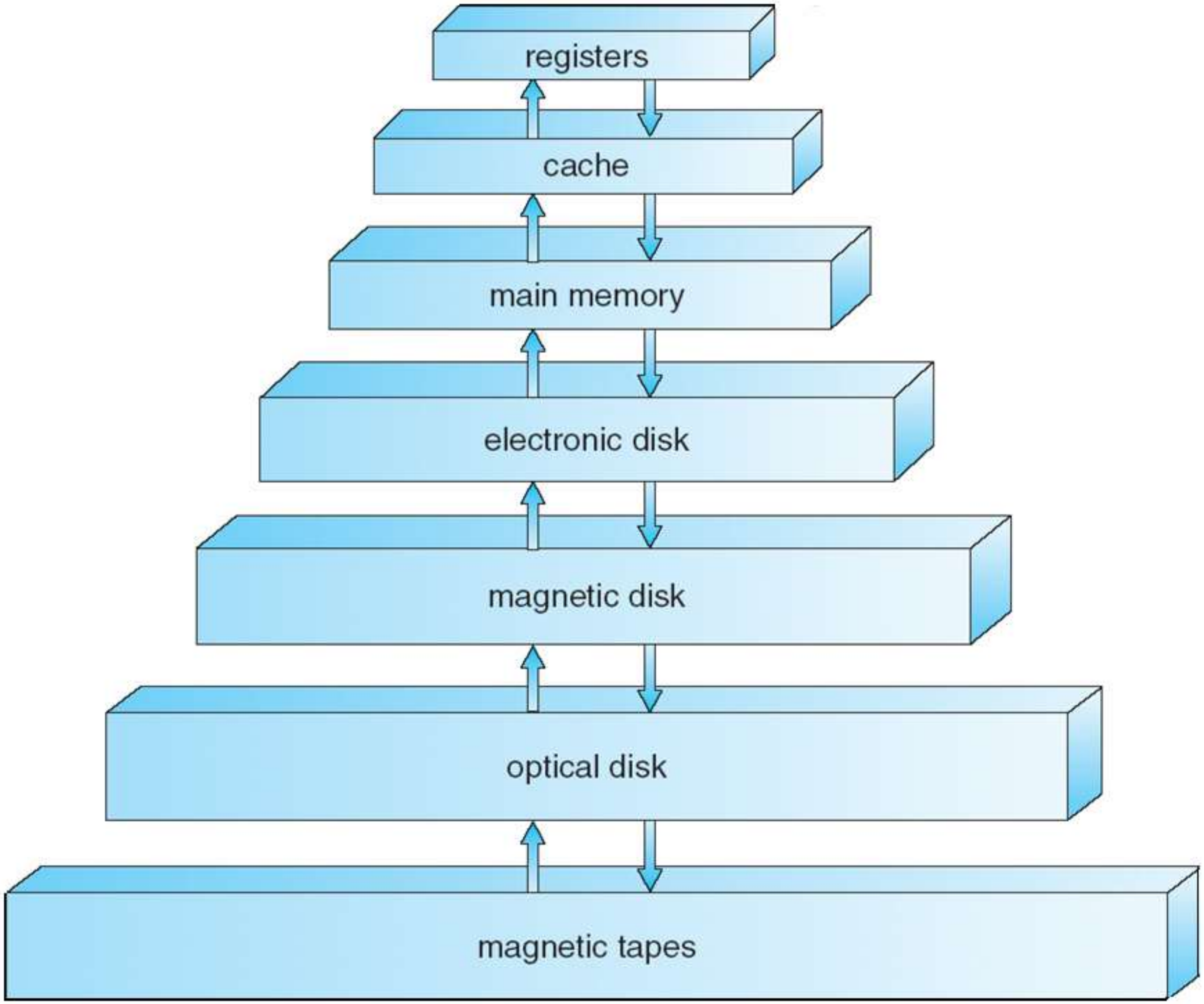


Disk partitioning
- Allowing multiple file systems

File and directory management
- Logical to physical mapping

File operations
- open, read, write, seek, close

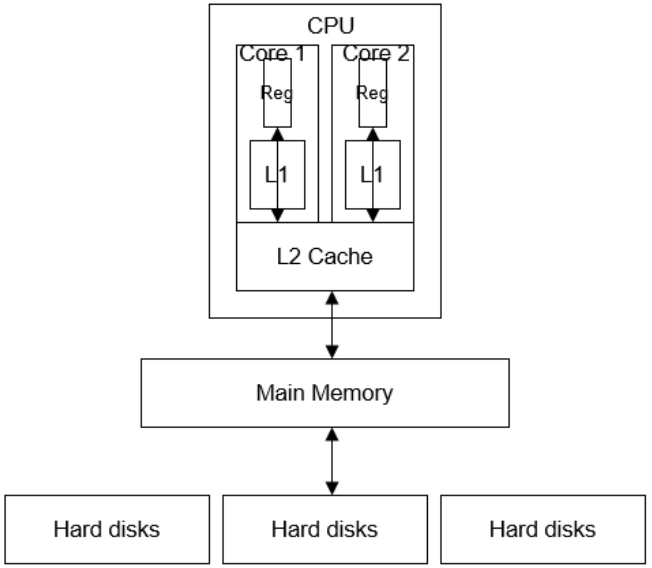
Storage-Device Hierarchy



Memory Hierarchy

Level	Size	Speed (ns)	Managed by
Registers	1KB	0.25 ~ 0.5	Compiler
Cache	16MB	0.5 ~ 25	Hardware
Main memory	16GB	80 ~ 250	OS
Hard disks	Several TB	5000	OS

- Important Rules on Storage Operations
  - Inclusion property
  - Memory coherence



Discussions

Solve Text Exercises:

- 1.8 (Privileged Instructions)
- 1.10 (CPU Mode)
- 1.22 (DMA)
- 1.25 (Cache Memory)