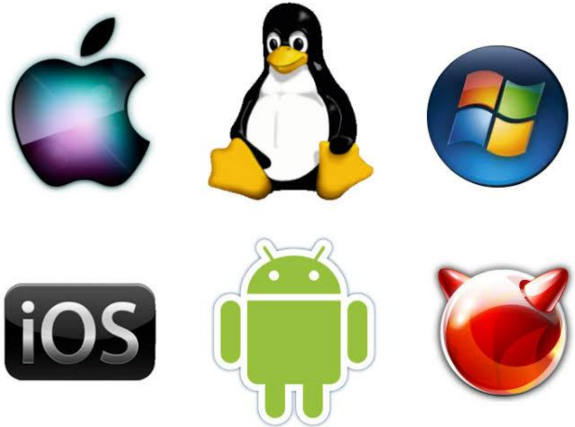
OPERATING SYSTEM

18 Aug 2025

Introduction & Background



Department of Computer Science and Engineering

International Institute of Information Technology, Bhubaneswar

Utkalika Satapathy

utkalika@iiit-bh.ac.in

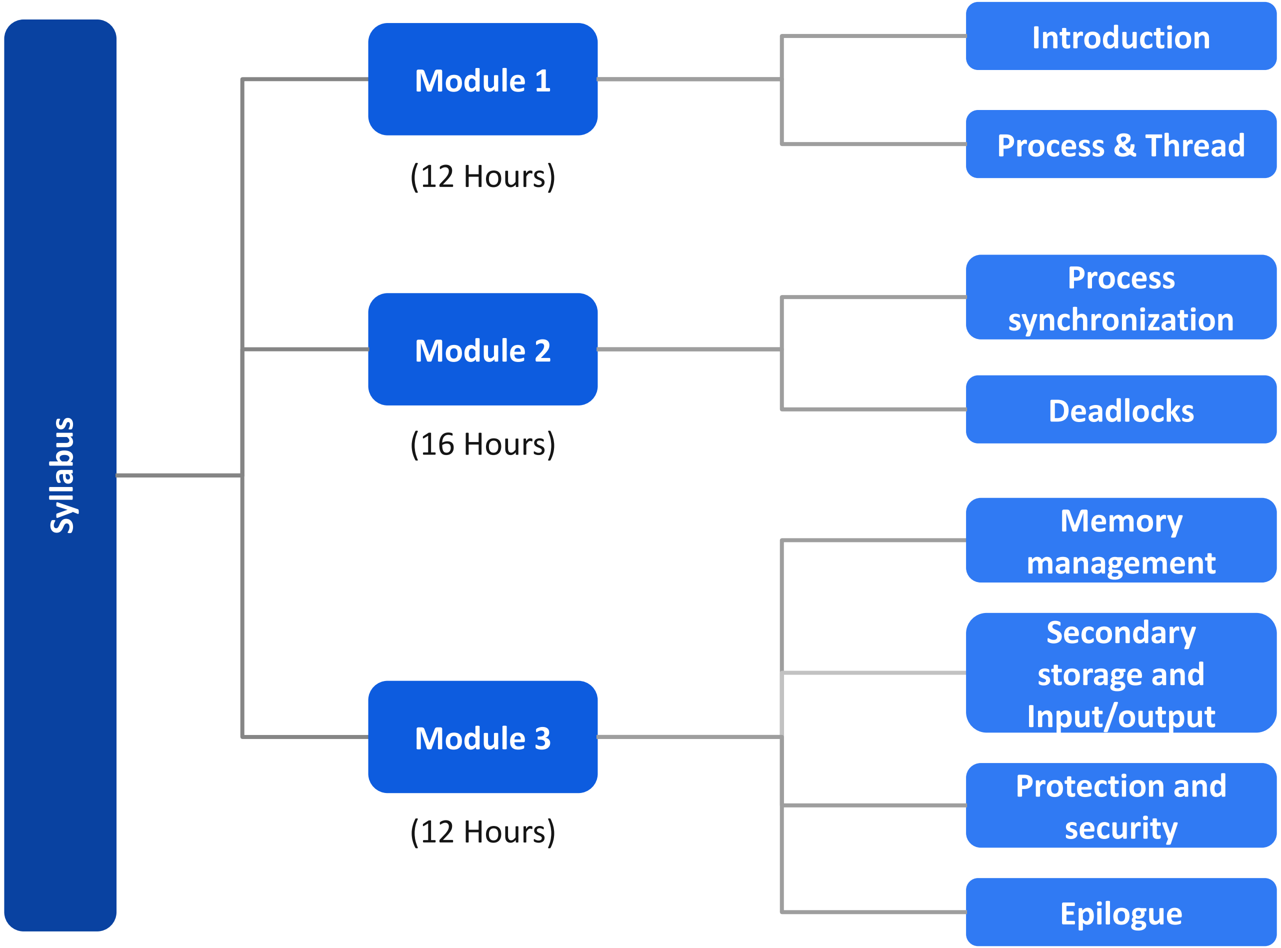
Prerequisites for this course

- Basic understanding of programming concepts
- Basic knowledge of computer hardware and system software

Course Objective

- Provides a comprehensive introduction to understanding the **principles**, **techniques** and **approaches** that constitute the structure of operating systems.
- To understand how the various elements of operating system **interact** and provide services for execution of application software.
- It focuses on OS support for **concurrency** (threads) and **synchronization**, **resource management** (e.g. CPU, memory, I/O), and distributed services etc.

Course Syllabus



Syllabus

Module 1

(12 Hours)

- Introduction
- Process & Thread

Module 2

(16 Hours)

- Process synchronization
- Deadlocks

Module 3

(12 Hours)

- Memory management
- Secondary storage and Input/output
- Protection and security
- Epilogue

Course Structure

01 Introduction

02 Processes & Threads

03 Process Synchronization

04 Deadlocks

05 Memory management

06 Secondary storage and Input/output

07 Protection and security

08 Epilogue

Grading and Evaluation

30%	Mid Semester
50%	End Semester
15%	Quizzes
5%	Teacher’s Assessment

Quizzes and Tests

- Quiz 1 - 1st September 2025
- Mid Sem - 15 September 2025
- Quiz 2 - 25th October 2025
- End Sem - 21st November 2025

Books & Materials

1. Abraham Silberschatz, Peter B. Galvin, Greg Gagne, **Operating System Concepts**, 8th Ed., John Wiley, 2008 (pdf)
2. William Stallings, **Operating Systems: Internals and Design Principles**. Prentice-Hall, 6th Ed., 2008 (pdf)
3. A. S. Tanenbaum, **Modern Operating Systems**, 3rd Ed., Pearson, 2009 (pdf)
4. A. S. Tanenbaum, AS Woodhull, **Operating Systems Design and Implementation**, 3rd Ed., Prentice Hall, 2006. (pdf)

Books & Materials

1. Robert Love, **Linux Kernel Development**, 3rd Ed., Addison Wesley, 2010 (pdf)
2. Maurice J. Bach, **The Design of the UNIX Operating System**, Pearson Education Inc., 1986 (pdf)
3. Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau, **Operating Systems: Three Easy Pieces**, ARPACI-DUSSEAU, 2014 (pdf)

Topics to be covered

01 - What is Computer System

02 - Introduction to popular operating systems

UNIX, Windows, etc.,

03 - Why Study OS

04 - What is OS

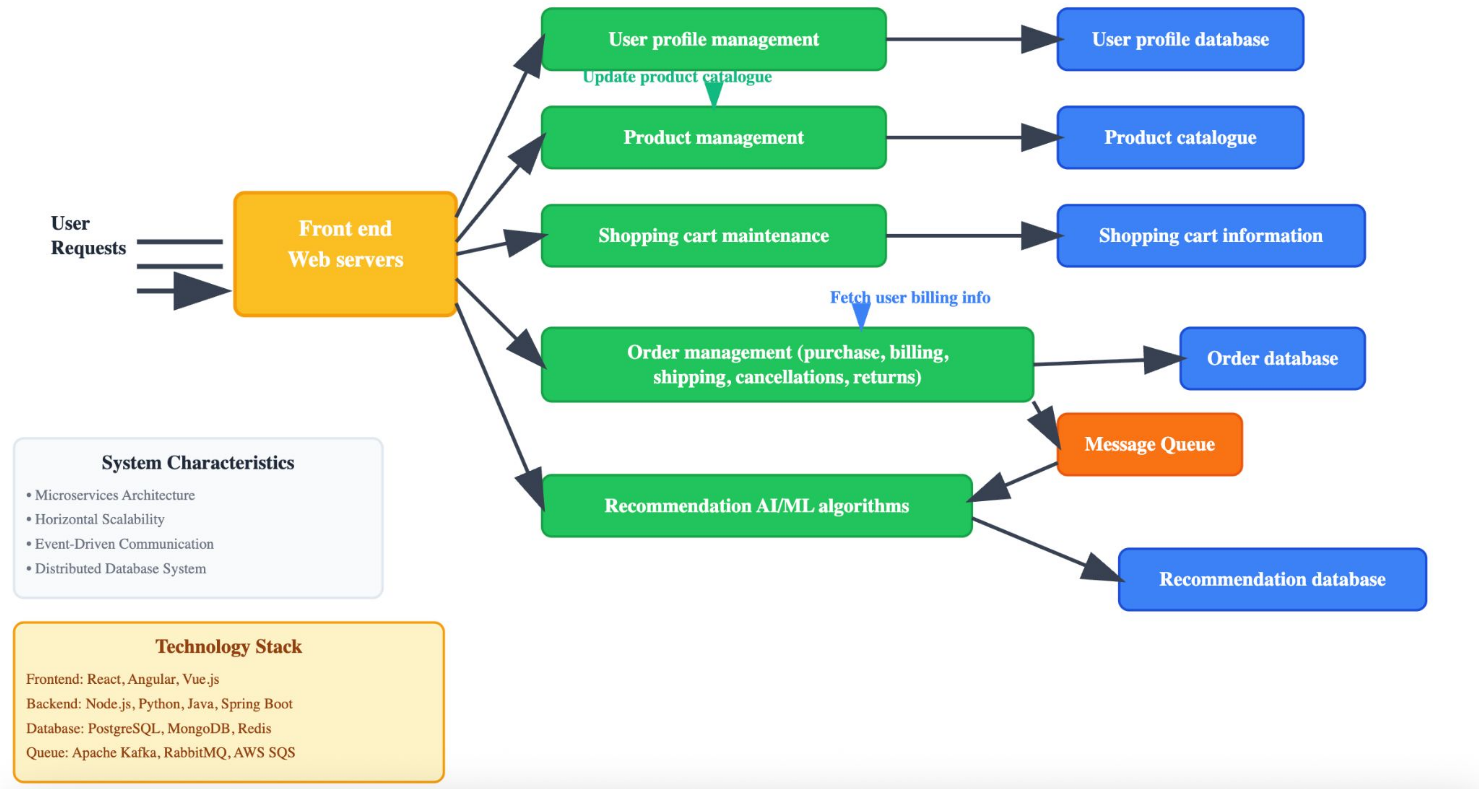
05 - History of OS

06 - Components of OS

What is a computer system?

- Real-world computer systems are complex
    - Multiple components/tiers distributed across several machines
    - Handle high number of user requests efficiently, reliably
- Example: consider an e-commerce application
    - Clients access multi-tier applications hosted in data centers or public clouds
    - Front-end components (e.g., web servers) receive user requests, reply to user with responses, consult various application servers to build responses
    - App servers contain business logic to process different types of user requests
    - Application data is stored in several database servers in the backend
    - Each of these components is built over one or more computers

Example: E-Commerce System



System Characteristics

- Microservices Architecture
- Horizontal Scalability
- Event-Driven Communication
- Distributed Database System

Technology Stack

Frontend: React, Angular, Vue.js

Backend: Node.js, Python, Java, Spring Boot

Database: PostgreSQL, MongoDB, Redis

Queue: Apache Kafka, RabbitMQ, AWS SQS

The Building Blocks

- A single computer system is the building block for all large, distributed computer systems that run real world applications
- What does a computer system contain?

The Building Blocks

- A single computer system is the building block for all large, distributed computer systems that run real world applications
- What does a computer system contain?
  - Software
  - Hardware
- We must understand the basic building blocks of a single system before we can build large-scale systems for real applications

The Building Blocks

- A single computer system is the building block for all large, distributed computer systems that run real world applications
- What does a computer system contain?
  - Software
    - User programs (instructions and data) to accomplish some tasks
    - System software like operating systems
  - Hardware
    - CPU (registers, ALU, caches)
    - Main Memory (DRAM)
    - I/O Devices, Secondary Storage …
- We must understand the basic building blocks of a single system before we can build large-scale systems for real applications

The Building Blocks

- Software written in high-level languages is compiled into binary files (executables/exe) containing instructions that the CPU hardware can execute
- OS written in high-level language like C

Why Study Operating System ?

Why Study Operating System ?

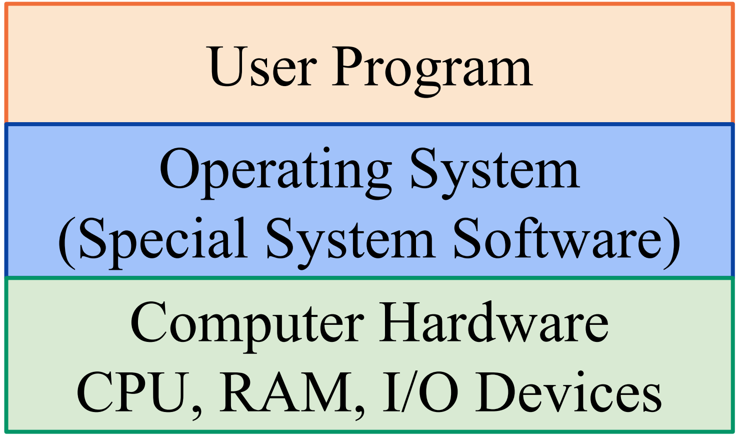
- Knowledge of hardware (architecture) + system software (OS), and how user programs interact with these lower layers, is essential to writing “good” (high performance, reliable) user programs
  - What exactly happens when you run a user program?
  - How to make your program run faster and more efficiently?
  - How to make your programs more secure, reliable, tolerant to failures?
  - Why is your program running slowly and how to fix it?
  - How much CPU/memory is your program consuming, and why?
- OS expertise is one of the most important skills when building high performance, robust, complex real life systems

What is an Operating System ?

What is an Operating System ?

- Middleware between user programs and system hardware
  - Not user application software but system software
  - Example: Linux, Windows, MacOS
- Manages computer hardware: CPU, main memory, I/O devices (hard disk, network card, mouse, keyboard etc.)
  - User applications do not have to worry about low-level hardware details
- Operating system has kernel + other extra useful software
  - Kernel = the core functionality of the OS
  - Other useful programs = shell, commands on shell, other utilities that help users interact with the OS

What is an Operating System ?



User Program

Operating System
(Special System Software)

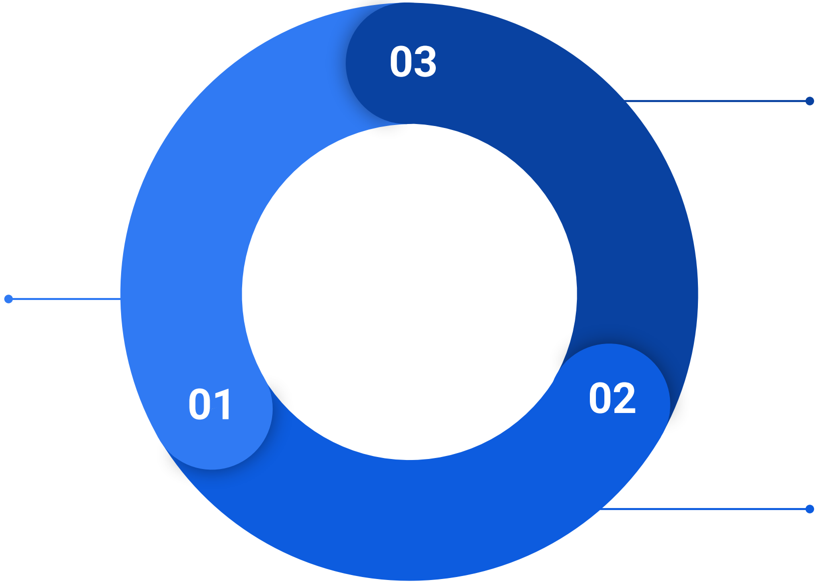
Computer Hardware
CPU, RAM, I/O Devices

The OS is a special system software between the user and the hardware. It manages the computer's hardware and implements abstractions to make the hardware easier to use.

History of OS

- Started out as a library to provide common functionality to access hardware, invoked via function calls from user program
  - Convenient to use OS instead of each user writing code to manage hardware
  - Centralized management of hardware resources is more efficient
- Later, computers evolved from running a single program to multiple processes concurrently
  - Multiple untrusted users must share same hardware
- So OS evolved to become trusted system software providing isolation between users, and protecting hardware
  - Multiple users are isolated and protected from each other
  - System hardware and software is protected from unauthorized access by users

Key Concepts in Operating System



Virtualization

OS gives a "virtual" or logical view of the hardware to the users, hiding the messy real "physical" view, so that each user/program has the illusion of having entire hardware to itself

Persistence

OS stores user data persistently on external I/O devices

Concurrency

OS runs multiple user programs at the same time, while sharing the system resources across users efficiently and securely

What is a Program

- User program = code (instructions for CPU) + data to do a specific task
- Stored program concept
    - User programs stored in main memory (instructions + data)
    - Memory is byte-addressable: data accessed via memory address / location / byte#
    - CPU fetches code/data from memory using address, and executes instructions
- CPU runs processes = running programs
- Modern CPUs have multiple cores for parallel execution
    - Each core runs one process at a time each
    - Modern CPUs have hyper-threading (one physical core can appear as multiple logical cores by sharing hardware, and hence run multiple processes at once)

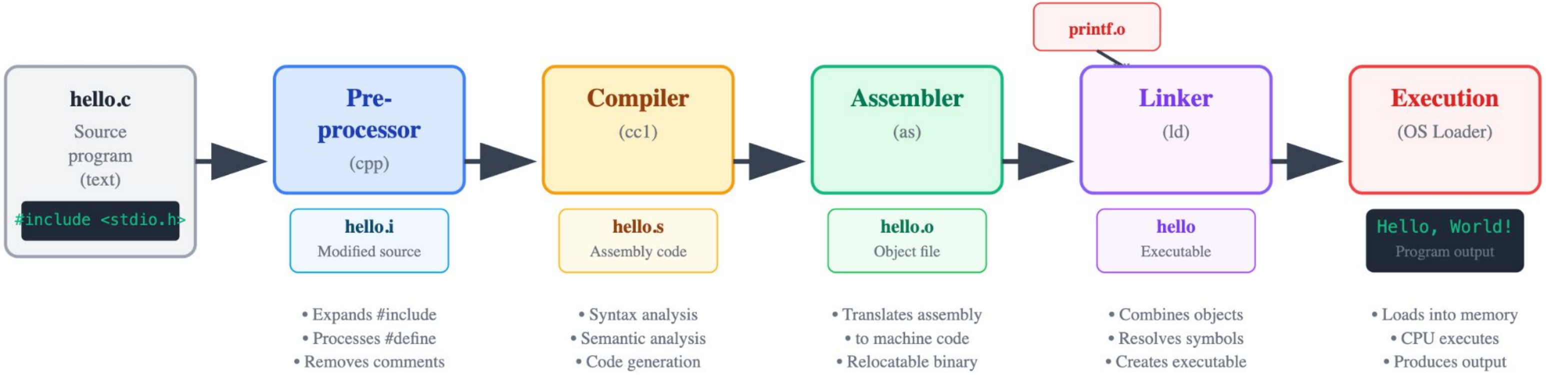
Running a Program

- What happens when you run a C program?

Running a Program

- What happens when you run a C program?
  - C code translated into executable by compiler
  - Executable file stored on hard disk (say, “a.out”)
  - When executable is run, a new process is created
  - Process allocated space in RAM to store code and data (compile time data allocated at start, runtime data allocated as program runs)
  - CPU starts executing the instructions of the program
- When CPU is running a process, CPU registers contain the execution context of the process
  - PC points to instruction in the program, general purpose registers store data in the program, and so on

Running a Program



printf.o

hello.c

Source program (text)

```
#include <stdio.h>
```

Pre-processor (cpp)

hello.i

Modified source

- Expands #include
- Processes #define
- Removes comments

Compiler (cc1)

hello.s

Assembly code

- Syntax analysis
- Semantic analysis
- Code generation

Assembler (as)

hello.o

Object file

- Translates assembly
- to machine code
- Relocatable binary

Linker (ld)

hello

Executable

- Combines objects
- Resolves symbols
- Creates executable

Execution (OS Loader)

```
Hello, World!
```

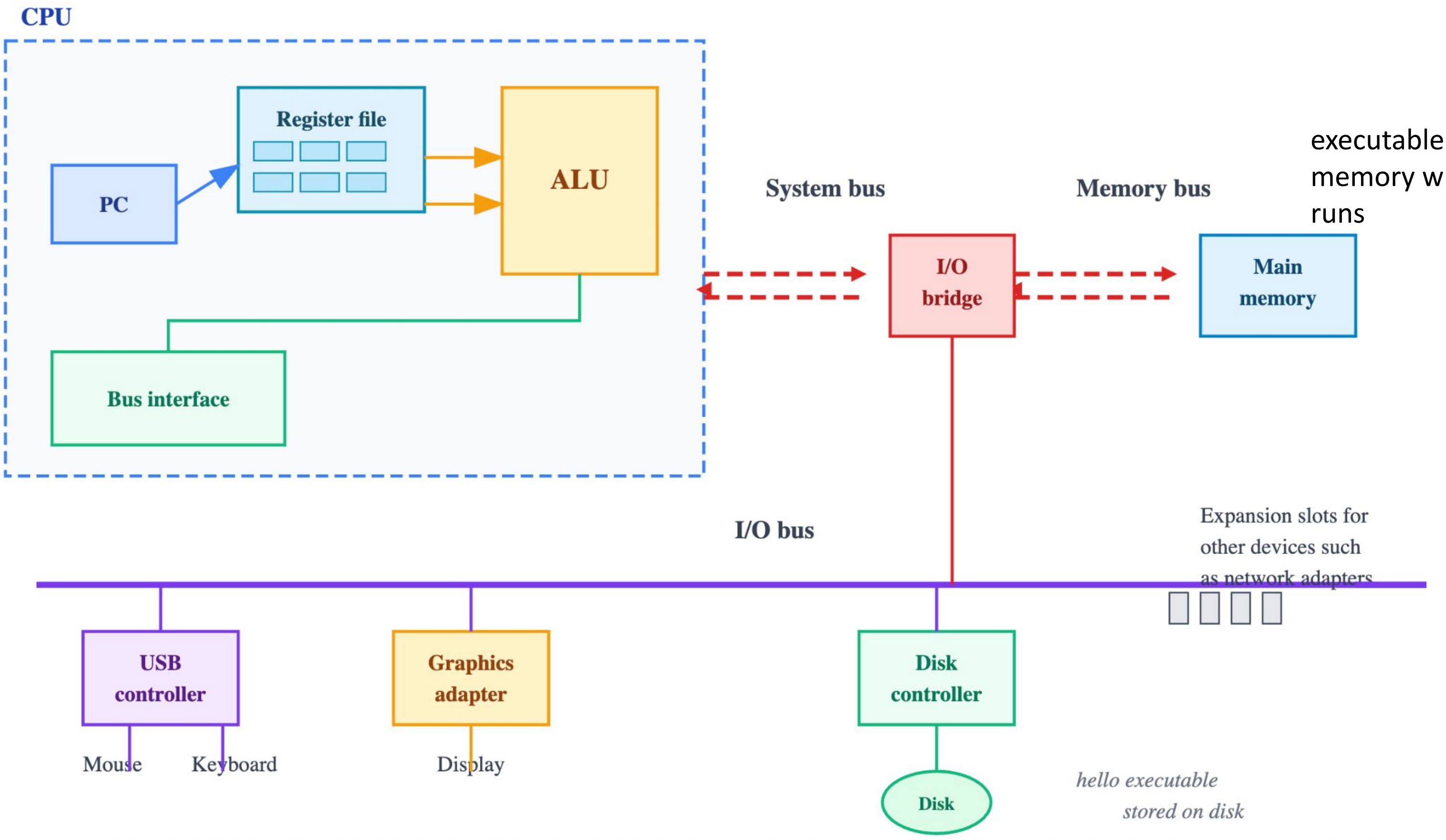
Program output

- Loads into memory
- CPU executes
- Produces output

File Types & Characteristics

Source (.c)	Preprocessed (.i)	Assembly (.s)	Object (.o)	Executable	Running Program
Human-readable	Expanded source	Mnemonic code	Machine code	Ready to run	In memory

Hardware Organization



executable loaded in memory when program runs

Next Class We Will Talk About

- Hardware Organization
- CPU ISA (Instruction Set Architecture)
- Memory/Storage Hierarchy
- Parts of Program Memory
- Memory Allocation
- Pointers and addresses
- Stack vs Heap

Happy Learning !