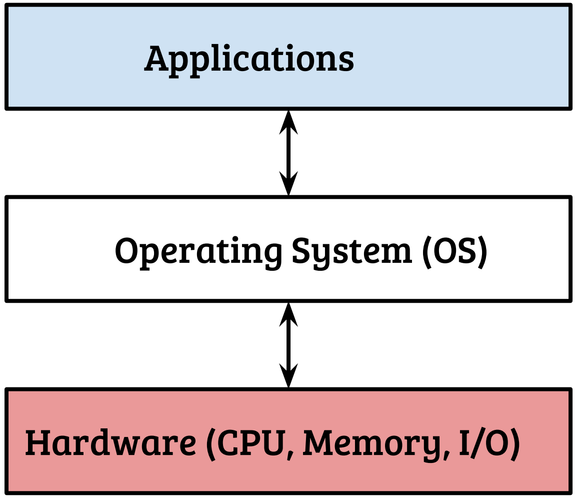
CS330: Operating Systems

Introduction

What is an Operating System?

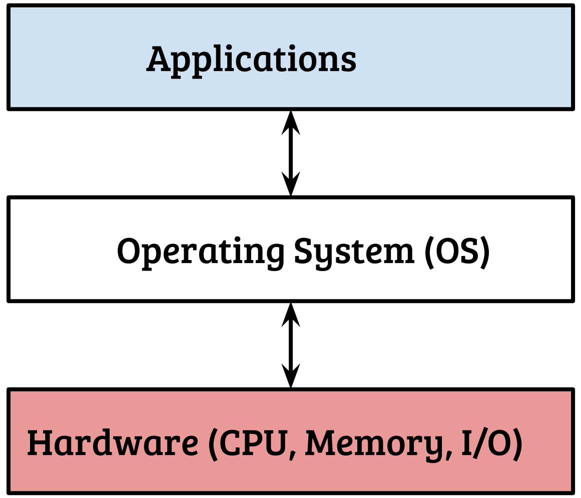


- Operating system is a software layer between the hardware and the applications
- What are the functions of this middleware?
  - Why is this intermediate layer necessary?

Even if it matters, why should we learn about this layer?

What if this software layer is removed from the scene?

What is an Operating System?

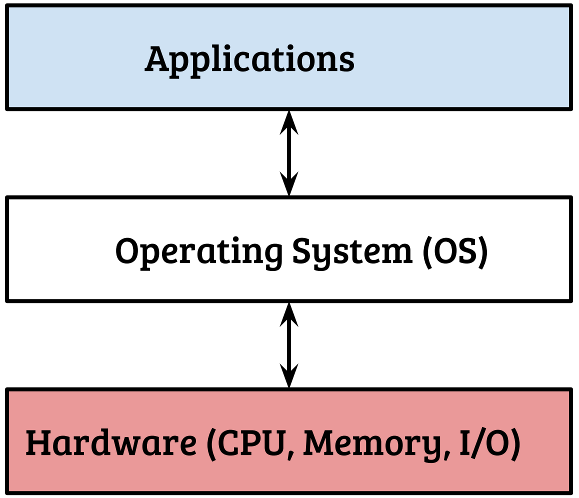


- Operating system is a <u>software layer</u> between the hardware and the applications
- What are the functions of this middleware?
  - Why is this intermediate layer necessary?

Even if it matters, why should we learn about this layer?

What if this software layer is removed from the scene?

What is an Operating System?

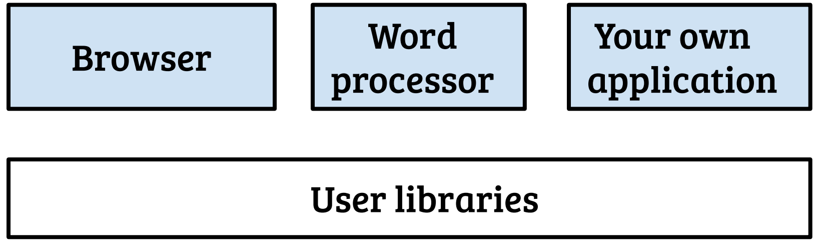


- Operating system is a <u>software layer</u> between the hardware and the applications
- What are the functions of this middleware?
  - Why is this intermediate layer necessary?

Even if it matters, why should we learn about this layer?

What if this software layer is removed from the scene?

What if we skip the OS layer?



Logic
Programming (C, Python etc.)
Data structures and Algorithms

Can build applications

Can even build libraries

What if we skip the OS layer?

Browser

Word processor

Your own application

User libraries

Logic
Programming (C, Python etc.)
Data structures and Algorithms

Oh! Need a computer to show my skills.

Can build applications

Can even build libraries

What if we skip the OS layer?

Browser	Word processor	Your own application

User libraries

Logic
Programming (C, Python etc.)
Data structures and Algorithms

Oh! Need a computer to show my skills.

I know logic gates to ISA

Can build a small computer for my program!

What if we skip the OS layer?

Browser | **Word processor** | **Your own application**

User libraries

Logic
Programming (C, Python etc.)
Data structures and Algorithms

Oh! Need a computer to show my skills.

I know logic gates to ISA

Can build a small computer for my program!

What is the role of the OS?

What if we skip the OS layer?

Applications

Library API

High-level programming language

(Will) know compilers to convert high-level language code to machine assembly

I understand only assembly code.

Hardware (CPU, Memory, I/O)

What if we skip the OS layer?

Applications

Library API

High-level programming language

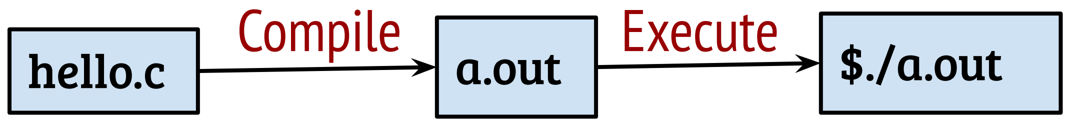
(Will) know compilers to convert high-level language code to machine assembly

I understand only assembly code.

Hardware (CPU, Memory, I/O)

Conclusion: do not need the OS. Hang-on, may be there is something else!

Program execution

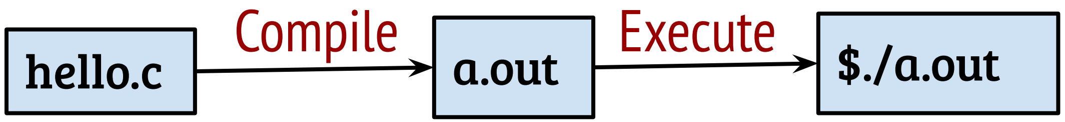


Program execution

$$\begin{tikzcd}
\boxed{\text{hello.c}} \arrow[r, "\text{Compile}"] & \boxed{\text{a.out}} \arrow[r, "\text{Execute}"] & \boxed{\text{\$./a.out}}
\end{tikzcd}$$

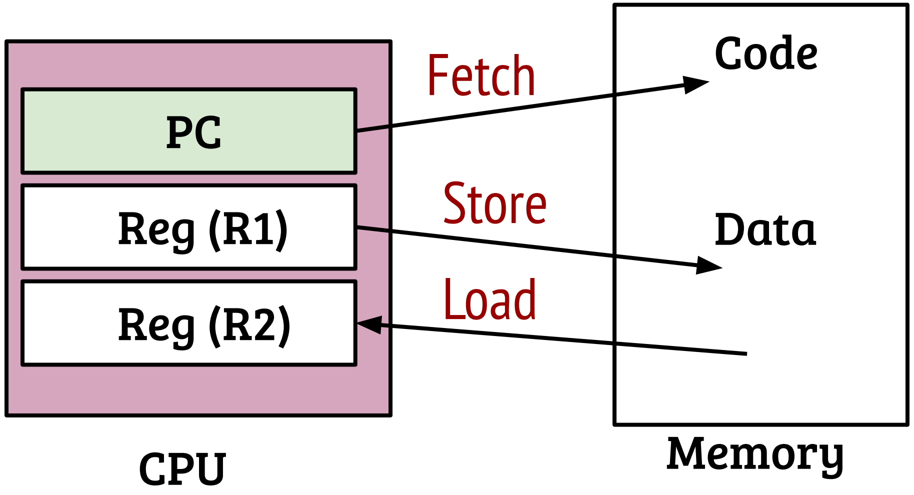
You said only CPU can execute!

Inside program execution



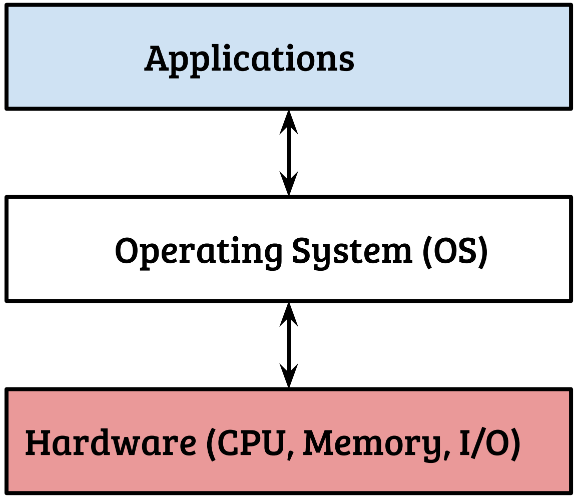
You said only CPU can execute!

CPU execution (from CS220)



- Loads instruction pointed to by PC
- Decode instruction
- Load operand into registers
- Execute instruction (ALU)
- Store results

What is an Operating System?



- OS bridges the *semantic gap* between the notions of application execution and real execution
  - OS loads an executable from disk to memory, allocates/frees memory dynamically
  - OS initializes the CPU state i.e., the PC and other registers
  - OS provides interfaces to access I/O devices
- OS facilitates hardware resource sharing and management (How?)

Resource virtualization

- OS provides virtual representation of physical resources
  - Easy to use abstractions with well defined interfaces
  - Examples:

Physical resource	Abstraction	Interfaces
CPU	Process	Create, Destroy, Stop etc.
Memory	Virtual memory	Allocate, Free, Permissions
Disk	File system tree	Create, Delete, Open, Close etc.

What is virtualization of resources?

- Definition [1] “Not physically existing as such but made by software to appear to do so.”
- By implication
  - OS multiplexes the physical resources
  - OS manages the physical resources
- Efficient management becomes more crucial with multitasking

1. Oxford dictionary : https://en.oxforddictionaries.com/definition/virtual

Design goals of OS abstractions

- Simple to use and flexible
- Minimize OS overheads
  - Any layer of indirection incurs certain overheads!
- Protection and isolation
- Configurable resource management policies
- Reliability and security

Next lecture: The process abstraction