Introduction

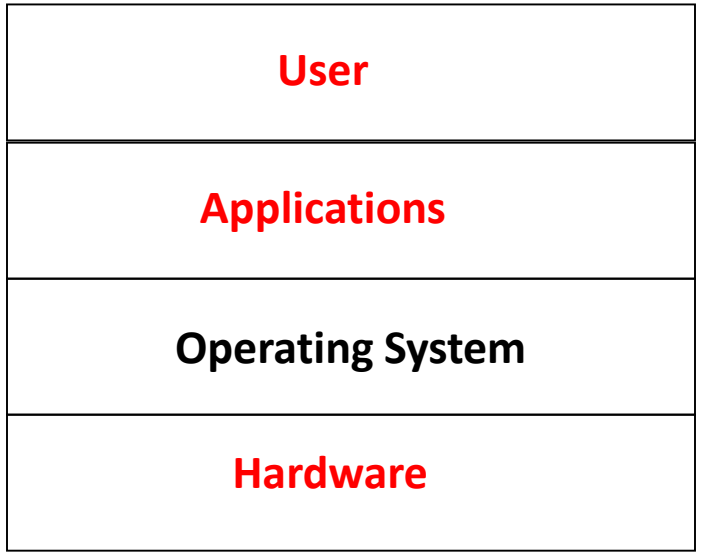
Sridhar Alagar

What will you learn?

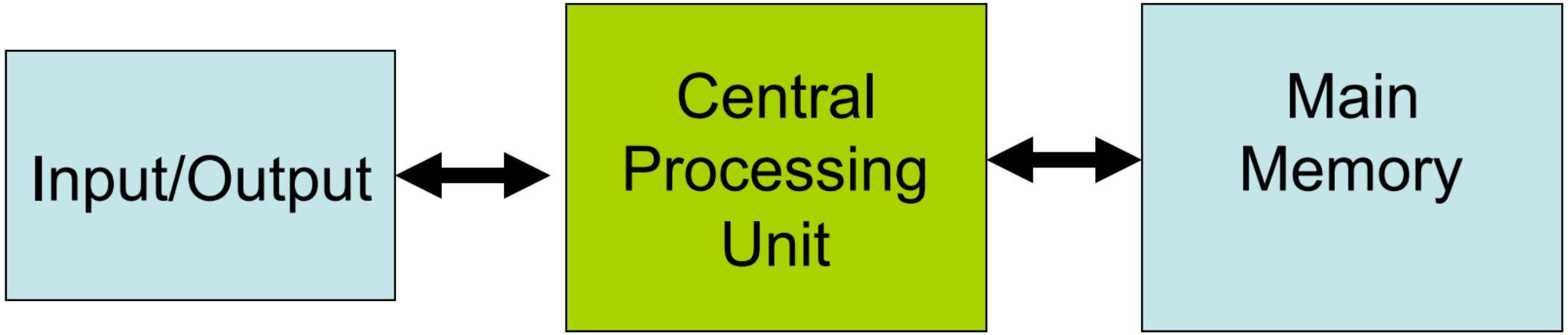
- OS design
  - Policy, Mechanism
- OS implementation
  - Look under the hood
  - xv6: a simple teaching operating system from MIT for x86
- Hands on OS programming in xv6,
  - 5 projects – 3 to 4 of them are kernel level programming
  - All projects should be implemented in C

What is an OS?

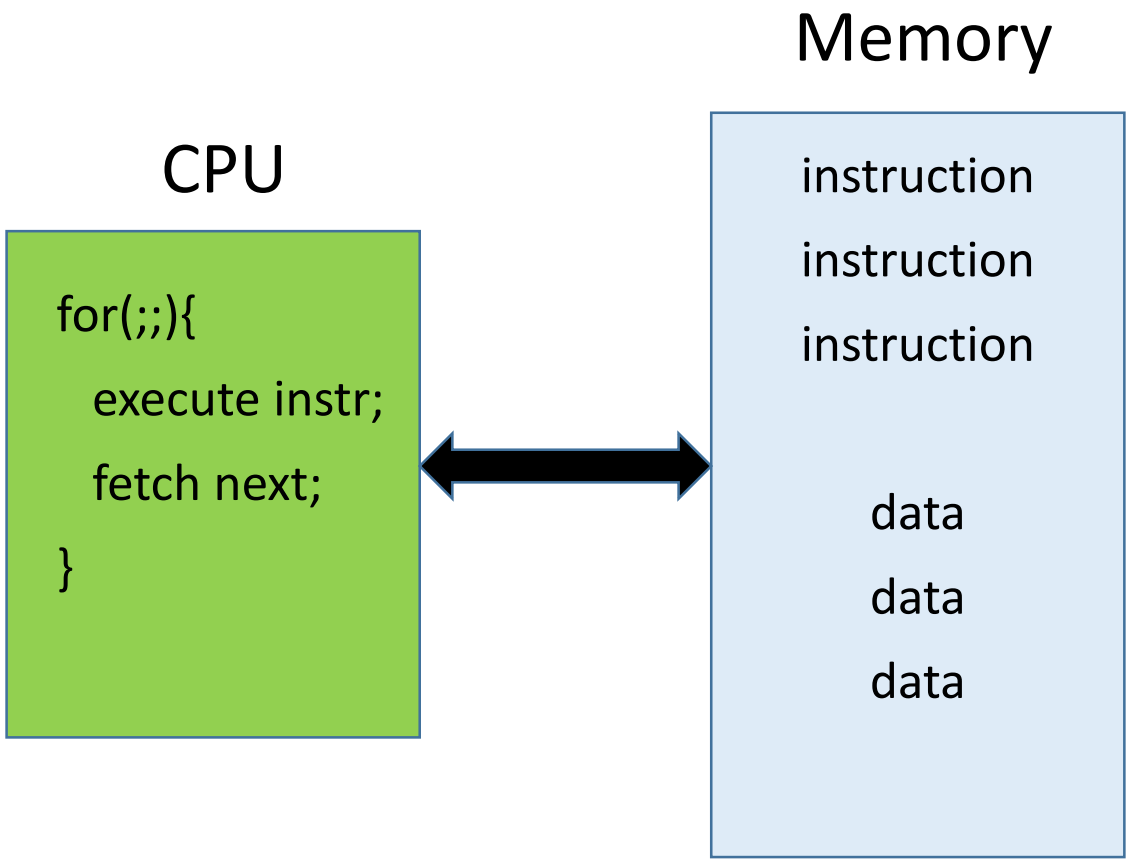
- Many definitions
  - Interface between hardware and application
  - Software that makes hardware into a useful form for applications



An abstract model of hardware



Stored Program Execution



What does OS provide?

- Abstraction
  - Common library for using resources
- Advantages of abstraction?
  - Make different devices look the same
  - Higher level functionality
- What abstraction?
  - CPU - processes
  - Memory – address space
  - Disk - files

What does OS provide?

- Resource Management
  - Share resources well
- Advantages of resource management?
  - Protect one application from another
  - Efficient use of resource
  - Fair access to resource
- Challenges
  - Policy
  - Mechanism

Three pieces: First

- Virtualization
  - Each application believes it has the resource for itself

Three pieces: Second

- Concurrency
  - OS must handle simultaneously occurring events
  - Easier if applications are independent
  - Trickier if they interact

Three pieces: Third

- Persistency
  - Access information permanently
  - Life time of information is longer than any running program
  - Provide abstraction so that application need not know how/where data is stored
  - Performance
  - Handle failures

Design Goals

- Efficiency
  - Low level view: a h/w management library
- Better usability
  - High level view: physical machine to an abstract one with better properties
- Protection: among applications, between OS and applications
  - Isolate process from one another
- Other goals:
  - Reliability
  - Security
  - Mobility

OS design/implementation is hard/fun

- Many trade-offs
  - Must be efficient (low level), but abstract (high level)
  - Powerful (many features), but simple (few building blocks)
- Features/behaviors interact
- Difficult environment: peculiar h/w, weak debugger

You will be glad you took OS

- If you want to
  - work on the above problems
  - know what is under the hood
  - build/modify an OS
  - Understand large and complex systems