Operating Systems
CS 333/347

Autumn 2022

People

- Instructors
  - Puru (office: SIA 404) www.cse.iitb.ac.in/~puru
  - Umesh (office: SIA 220) www.cse.iitb.ac.in/~umesh

- TAs
  - Ashwin, Adarsh Varma, Nadeesh, Jatin
  - Aniket, Adarsh, Raja, Purvi, Rahul

Course Structure and Logistics

- Theory
  - Slot 2 (M - 9.30am, T - 10.30am, Th - 11.30am)
- Lab
  - Monday afternoons 2pm - 5pm
  - In lab + take home
- Piazza for interactions/discussion forum
  - ***piazza.com/iitb.ac.in/autumn2022/cs347333***
- Moodle for lab submissions

Syllabus & Course Text

- **CPU virtualization**
  - Limited direct execution
  - Processes and Process Management
  - CPU Scheduling
- **Memory virtualization**
  - Address spaces
  - Virtual memory
  - Address Translation
- **Concurrency**
  - Threads and thread scheduling.
  - Synchronization primitives
- **File systems**
  - Storage devices and disk scheduling.
  - iNodes
  - Journaling & Transactions
- **I/O**
  - Network stacks
  - RPC
- **Security**

Textbook

Operating Systems: Three Easy Pieces
by Remzi and Andrea Arpaci-Dusseau

available for online:
http://pages.cs.wisc.edu/~remzi/OSTEP

Labs - the xv6 Operating system

Linux based tools

Programming in C

xv6 — a simple Unix-like teaching operating system

Assessment

- Theory (CS347)
  - 2-3 scheduled quizzes (20%)
  - In-class *surprise* SAFE quizzes (5%)
  - Mid term and final exams (30% - 45 %)

- Lab (CS333)
  - 4 lab quizzes worth 100% of the lab grade

Plagiarism policy

Acceptable:

- Explaining a concept to someone in another group
- Discussing algorithms/testing strategies with other groups
- Helping debug someone else's code (in another group)
- Searching online for generic algorithms (e.g., hash table)

Unacceptable:
(will result in a report to DADAC + a zero on lab assignment/quiz/exam)

- Sharing code/answers with your friend
- Copying OR reading another's code/answers
- Copying online code or material from prior years OR from the Internet. (even reading this and typing it yourself is forbidden)

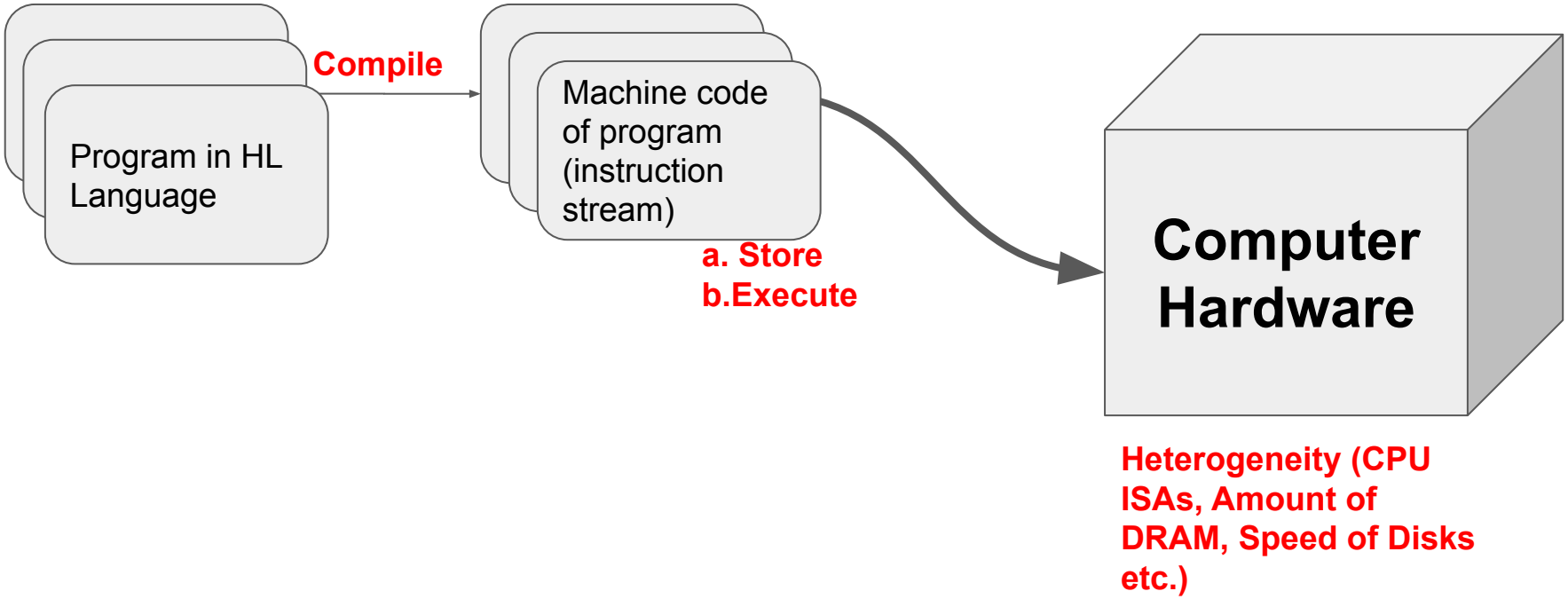
Ready, Set, OS!

Goals for this class

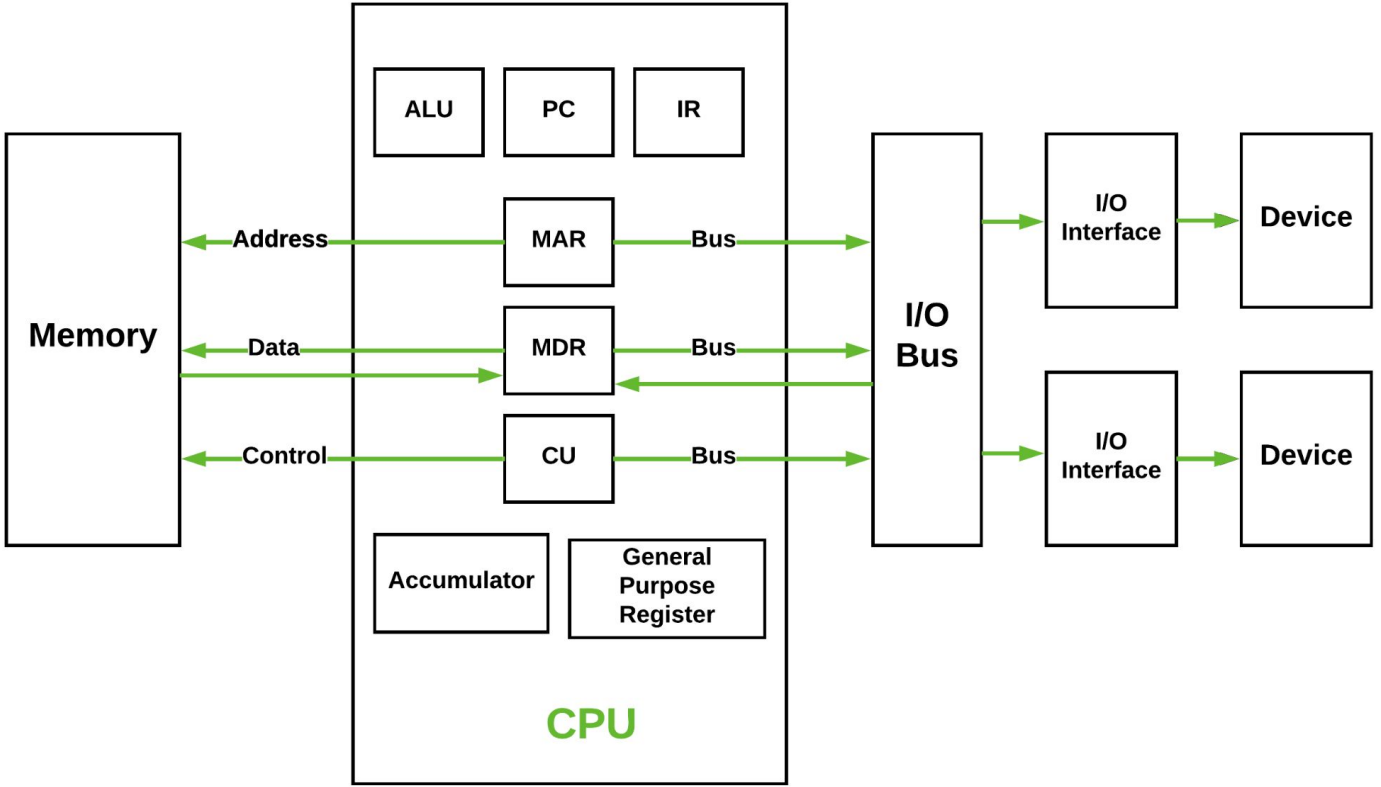
- Background - how do programs execute
- What is an Operating System?

  And – what is it not?
- Some OS designs
- Why study Operating Systems?

The Bigger Picture



Recall: How do instructions execute?

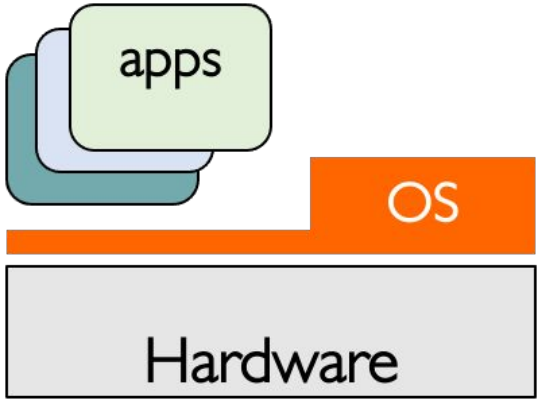


Executing a program

1. Load the program from stable store (File) into memory (process)
2. Initiate execution of the process at its entry point (main function)
3. Read input by interfacing with Input devices (keyboard, disks, network cards etc.)
4. Allocate resources as and when required - memory for example
5. Use auxiliary devices when needed - run CUDA kernels on GPUs for example.
6. Write output by interfacing with Output devices (disks, displays, network cards etc.)
7. Allow other programs to execute simultaneously without interfering with their execution.

What is an OS?

Provides user programs with a more convenient *abstract machine* interface rather than the underlying *physical machine*



Bridging the Gap - the Operating System

Requirements

- **Virtualization** capable - Allow multiple programs to run simultaneously.
- **Robust** - a single program should not crash the computer
- **Secure** - each program should not interfere with another program's resources, instructions or data.
- **Programming Simplicity**
  - High level of abstraction to low level hardware - example: Bitmapped display to windowing system.
- **Uniform Interface** to heterogeneous hardware.
  - If it looks like a disk and quacks like a disk, it IS a disk.
- **Network Transparency** - distributed operating systems

A few *Abstractions* and *Mechanisms*

CPU Virtualization
Abstraction: Processes and Threads

Multiple Programs
Context Switching
Protection
Isolation

Memory Virtualization
Abstraction: Address spaces
Mechanisms: Segmentation, Address translation, Paging, Swapping etc.

Handling I/O
Interrupts, DMA, TCP/UDP Stacks, Disk Scheduling

Security and Protection
Access control (on files), Segmentation faults on illegal access

What is an OS?

- No universal definition
- MUST haves:
  - CPU Scheduling
  - Memory management
  - Multi programming mechanisms (concurrency and synchronization)
  - I/O management
  - Network communication
- Nice to haves:
  - File system
  - Windowing system
  - ????
- “Kernel” vs the rest
  -

Unix OS Structure