Computer Systems Organization

https://nyu-cso.github.io

Jinyang Li

Course information

Lecturer: Prof. Jinyang Li

MW 2-3:15pm CIWW-109

Recitation instructor (& course assistant):

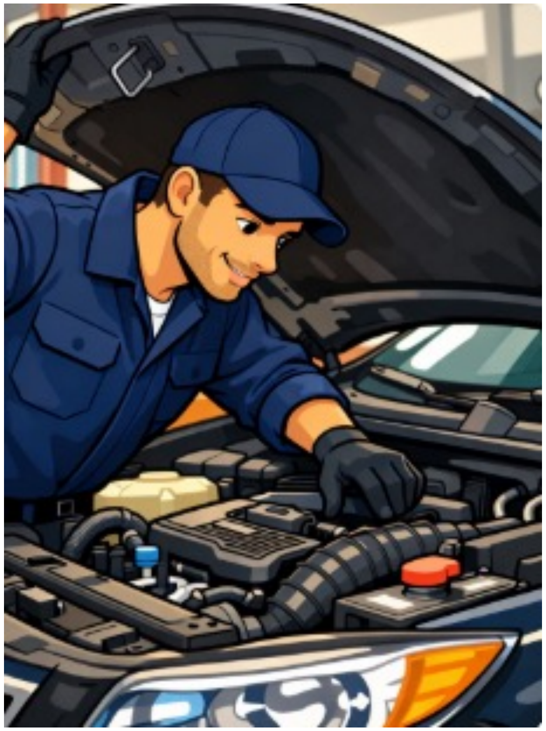
Haitian Jiang (3rd year Ph.D. student)

F 11am-12:15pm, Silv 207

Course Goal

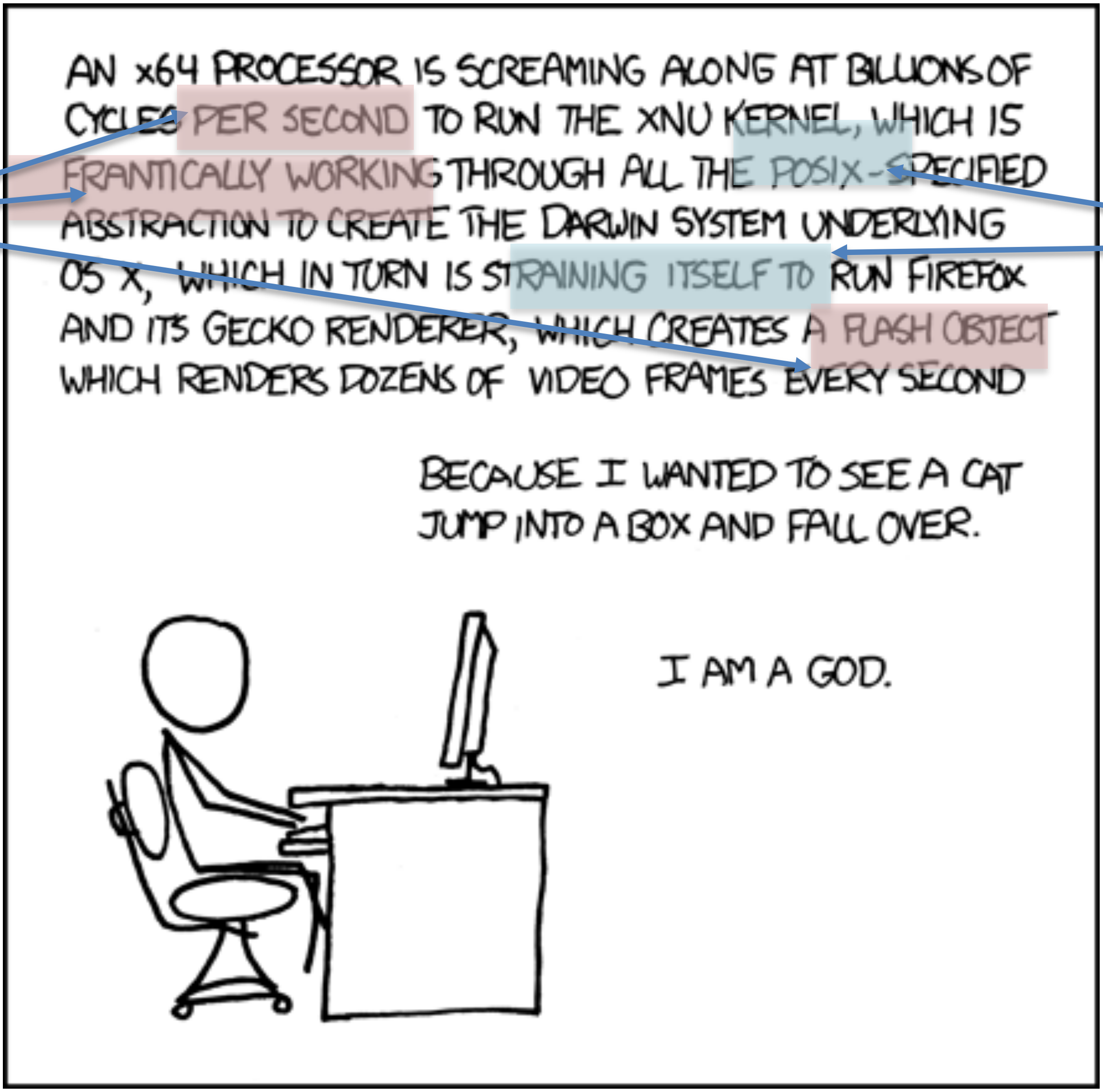
- Beyond learning how to program
  - Learn the gritty internals of how a computer system works

You after CS101, 102



You after CS201, 202

Goal: learn how computer systems work

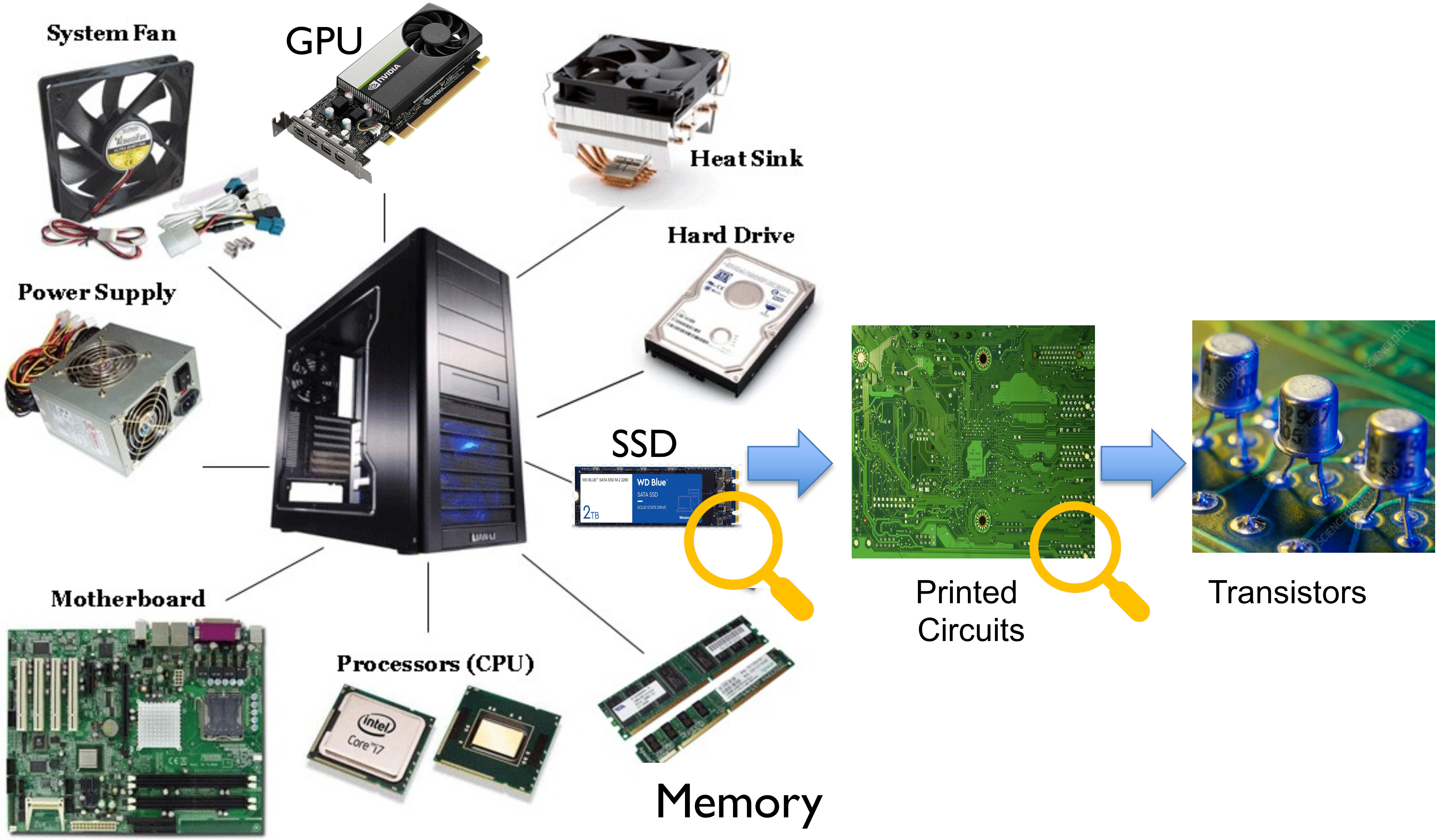

AN x64 PROCESSOR IS SCREAMING ALONG AT BILLIONS OF CYCLES PER SECOND TO RUN THE XNU KERNEL, WHICH IS FRANTICALLY WORKING THROUGH ALL THE POSIX-SPECIFIED ABSTRACTION TO CREATE THE DARWIN SYSTEM UNDERLYING OS X, WHICH IN TURN IS STRAINING ITSELF TO RUN FIREFOX AND ITS GECKO RENDERER, WHICH CREATES A FLASH OBJECT WHICH RENDERS DOZENS OF VIDEO FRAMES EVERY SECOND
BECAUSE I WANTED TO SEE A CAT JUMP INTO A BOX AND FALL OVER.
I AM A GOD.

Covered by CSO

To be covered by OS (202)

Components of a computer

Components of a computer: hardware

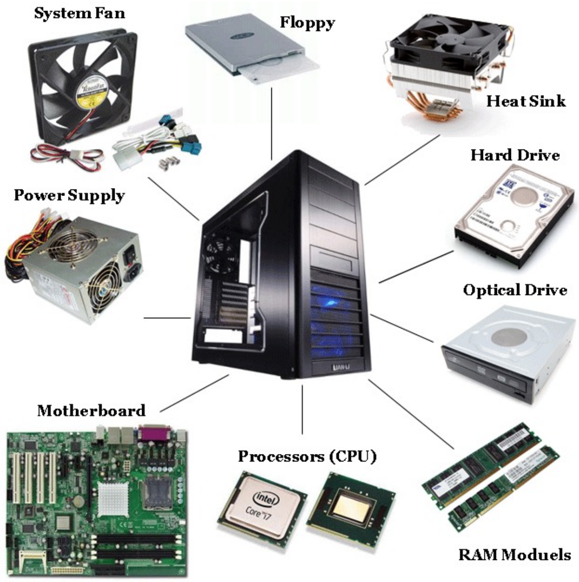


Components of a computer: hardware + software

Layered Organization

Software

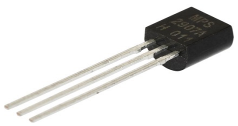
Hardware



Layered Organization

Software

Hardware


Transistors

Diodes

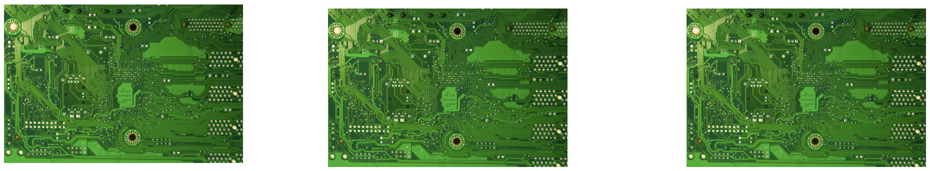
Resistors

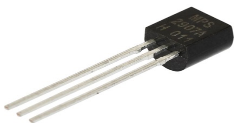
Layered Organization

Software

Hardware

Logical Circuits,
Flip-Flops, Gates





Transistors

Diodes

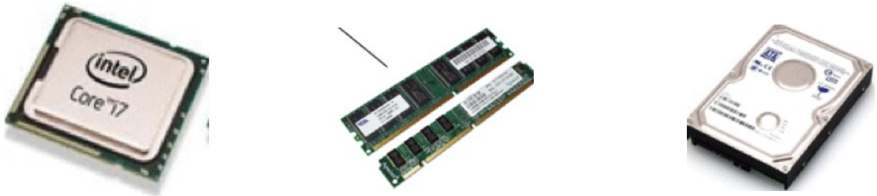
Resistors

Layered Organization

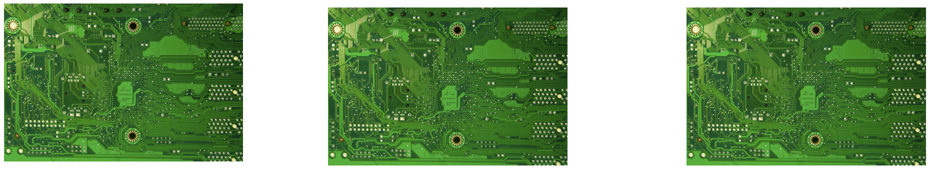
Software

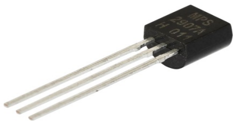
Hardware

CPU, Memory, Disk



Logical Circuits,
Flip-Flops, Gates





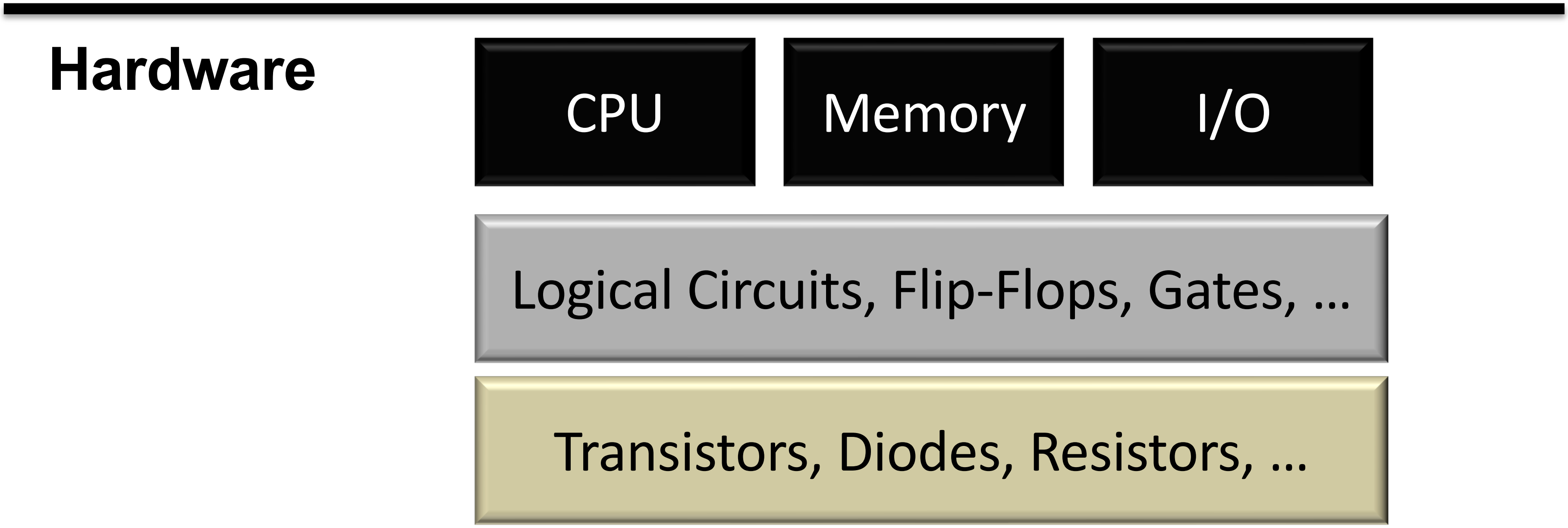
Transistors

Diodes

Resistors

Layered Organization

Software



Hardware

CPU

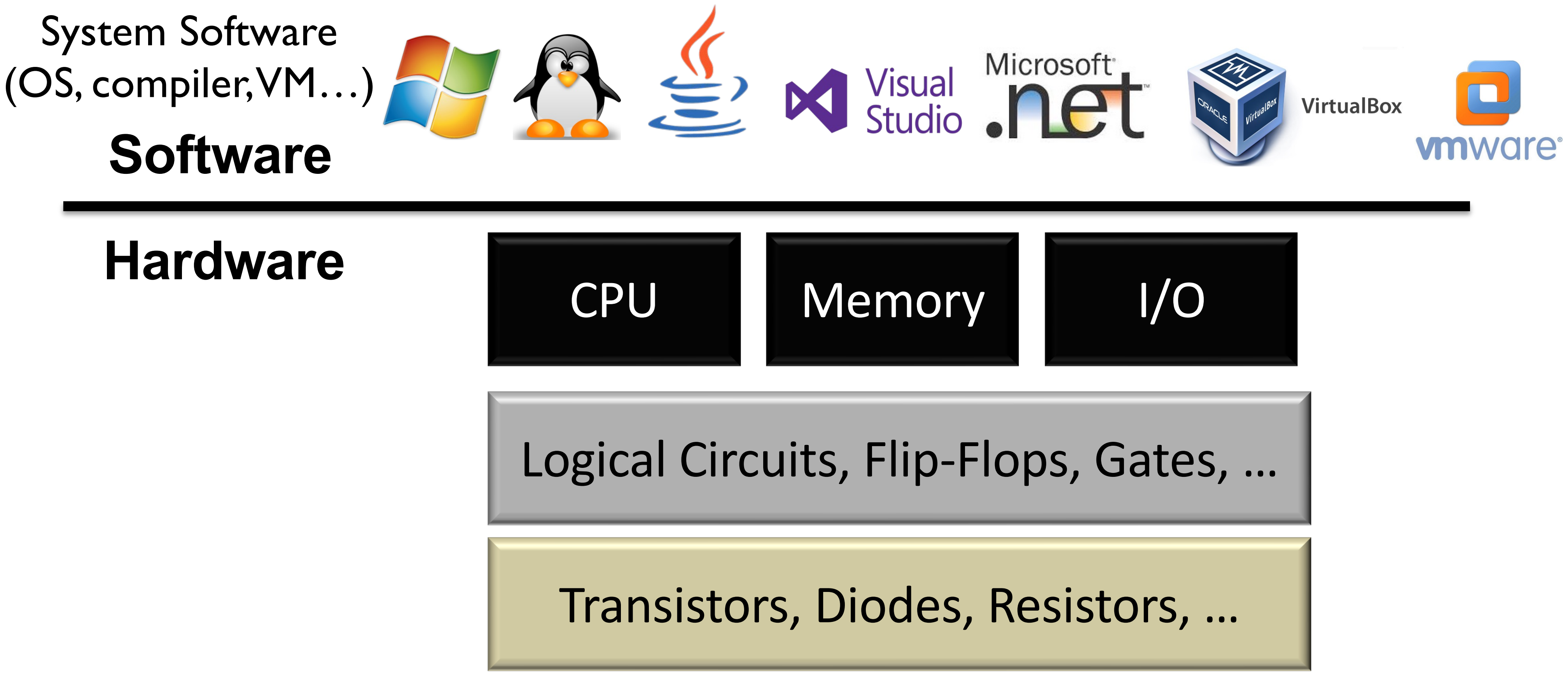
Memory

I/O

Logical Circuits, Flip-Flops, Gates, …

Transistors, Diodes, Resistors, …

Layered Organization



System Software
(OS, compiler, VM…)

Software

Hardware

CPU

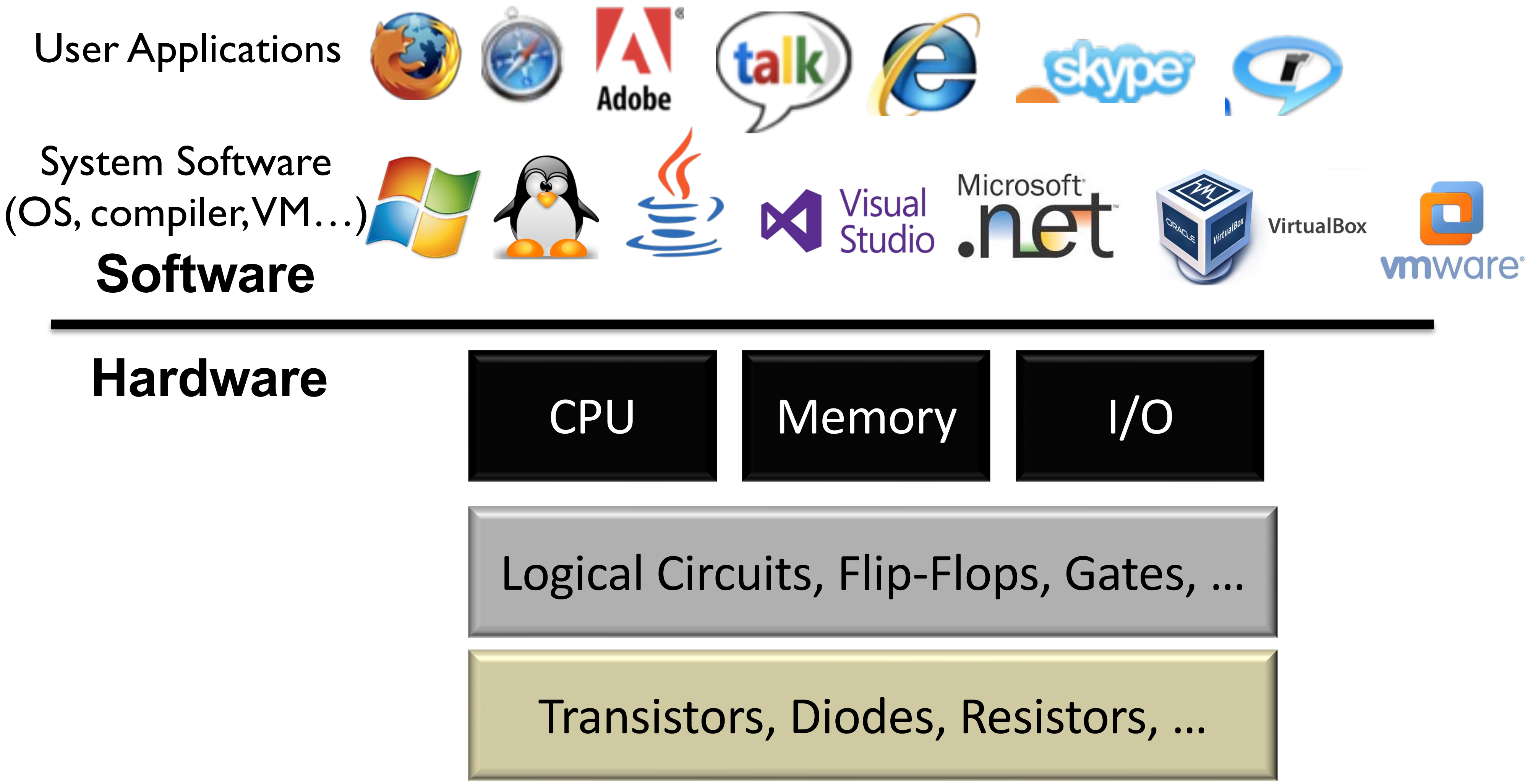
Memory

I/O

Logical Circuits, Flip-Flops, Gates, …

Transistors, Diodes, Resistors, …

Layered Organization



User Applications

System Software (OS, compiler, VM…)

Software

Hardware

CPU

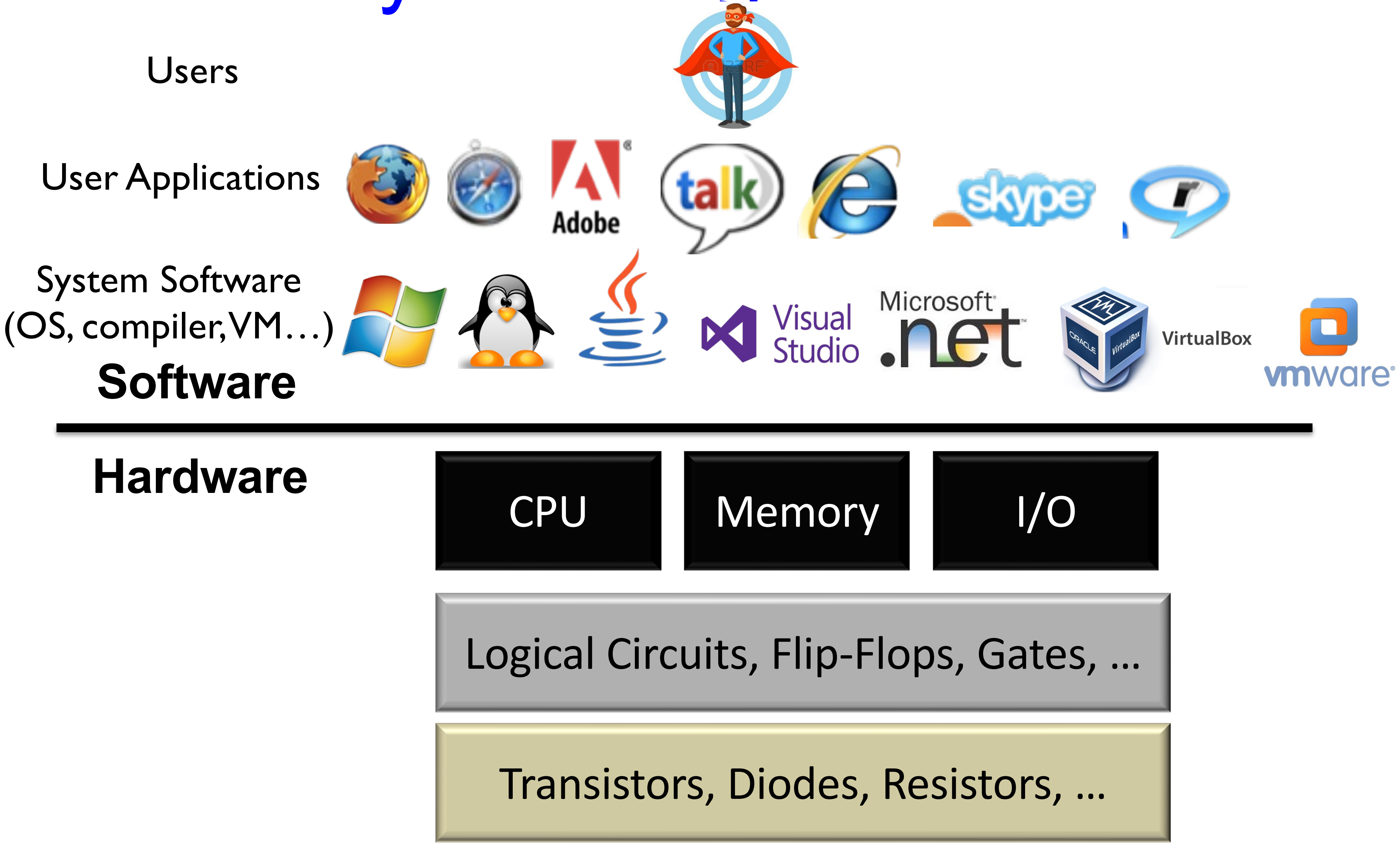
Memory

I/O

Logical Circuits, Flip-Flops, Gates, ...

Transistors, Diodes, Resistors, ...

Layered Organization



Users

User Applications

System Software (OS, compiler, VM…)

Software

Hardware

CPU

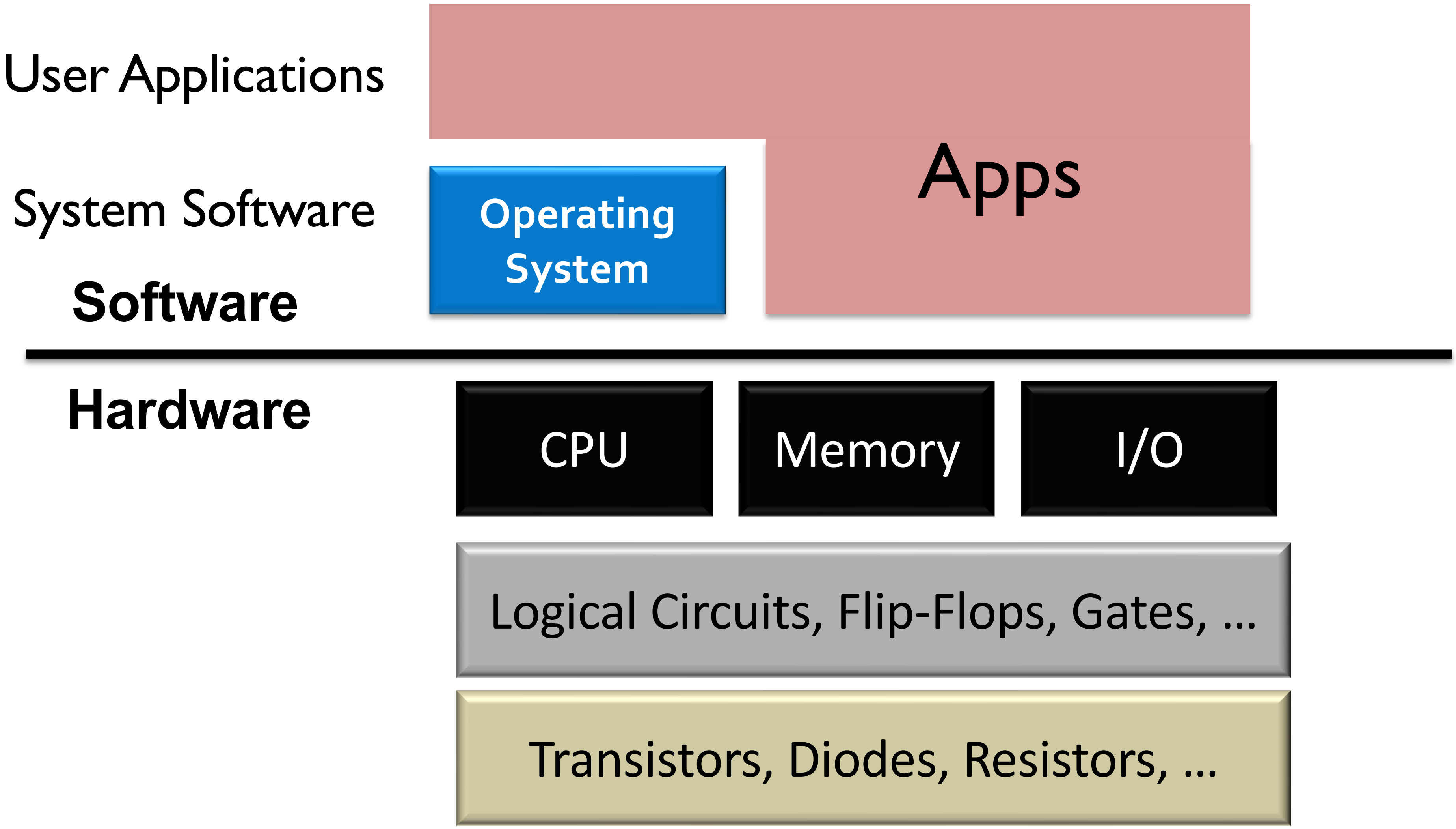
Memory

I/O

Logical Circuits, Flip-Flops, Gates, …

Transistors, Diodes, Resistors, …

Layered Organization



User Applications

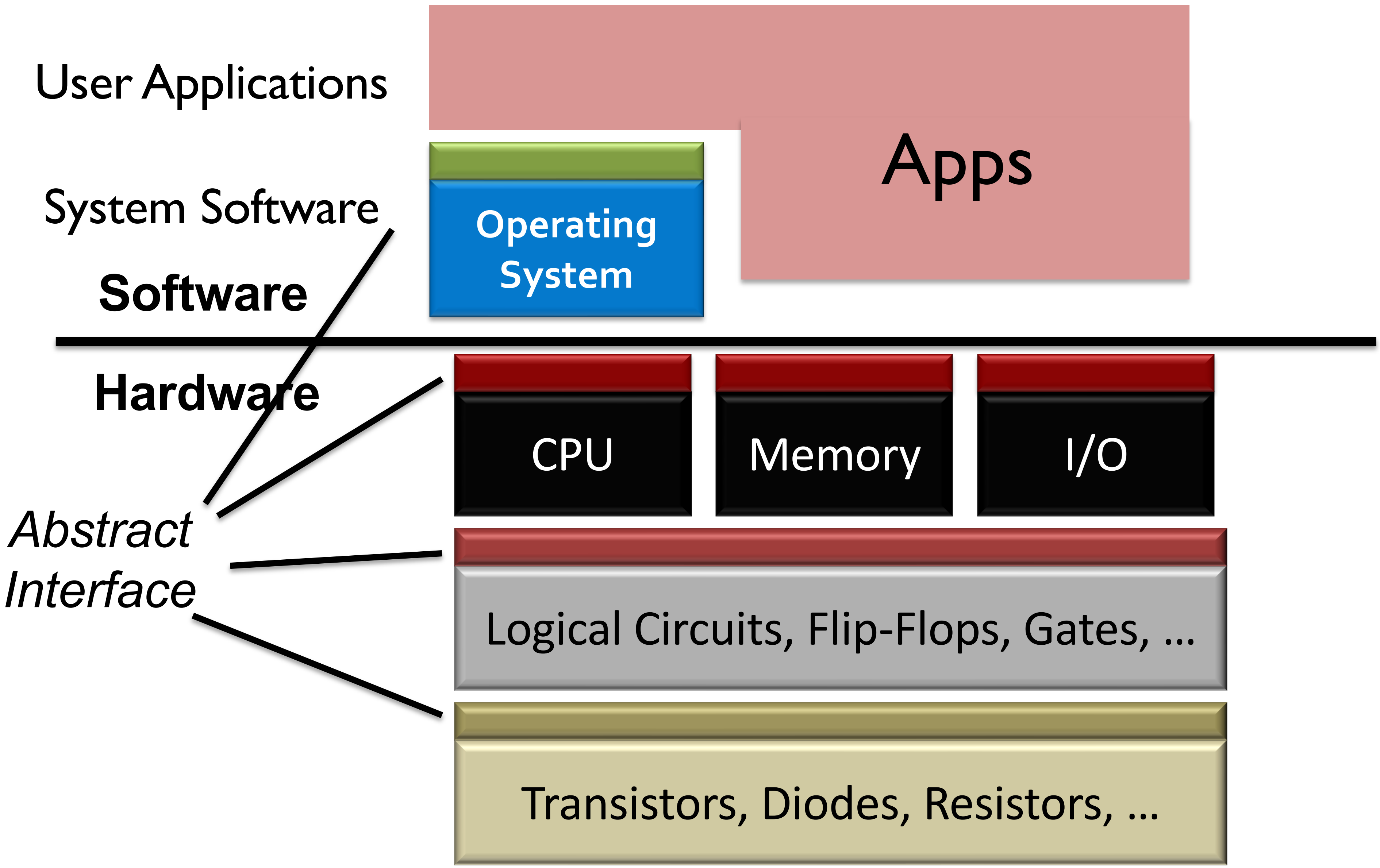
System Software

Software

Hardware

Apps
Operating System
CPU
Memory
I/O
Logical Circuits, Flip-Flops, Gates, …
Transistors, Diodes, Resistors, …

Abstraction



User Applications

System Software

Software

Hardware

Abstract Interface

Apps

Operating System

CPU

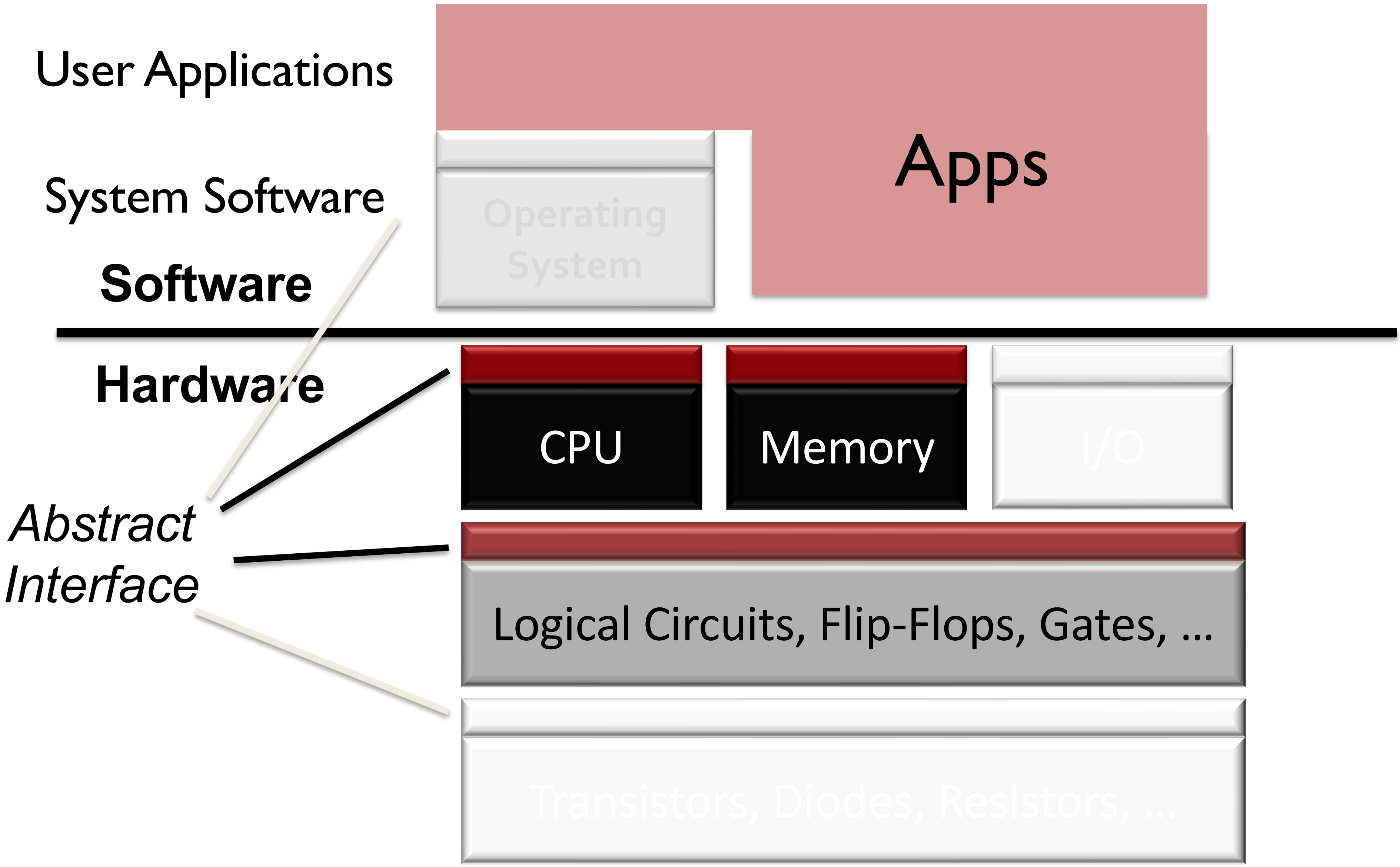
Memory

I/O

Logical Circuits, Flip-Flops, Gates, …

Transistors, Diodes, Resistors, …

Scope of this class



User Applications

System Software

Software

Hardware

Abstract Interface

Apps

Operating System

CPU

Memory

I/O

Logical Circuits, Flip-Flops, Gates, ...

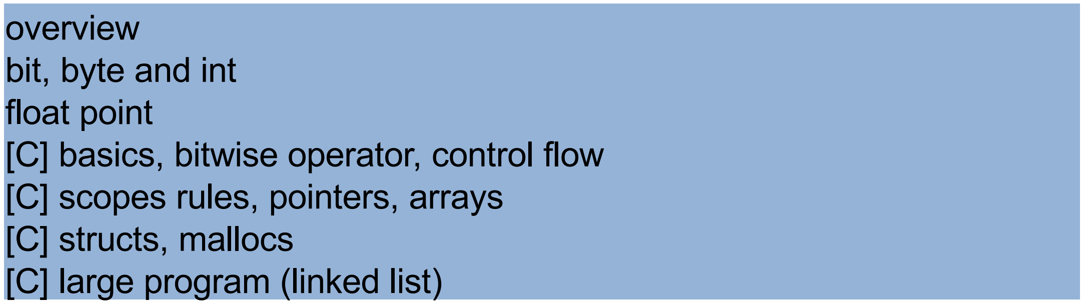
Transistors, Diodes, Resistors, ...

Scope of this class

1. How do applications run on a computer?
   – Hardware/software interface
2. How do CPU/memory work?
   – overview of computer architecture

Schedule

https://nyu-cso.github.io



overview
bit, byte and int
float point
[C] basics, bitwise operator, control flow
[C] scopes rules, pointers, arrays
[C] structs, mallocs
[C] large program (linked list)

C Programming

Schedule

https://nyu-cso.github.io

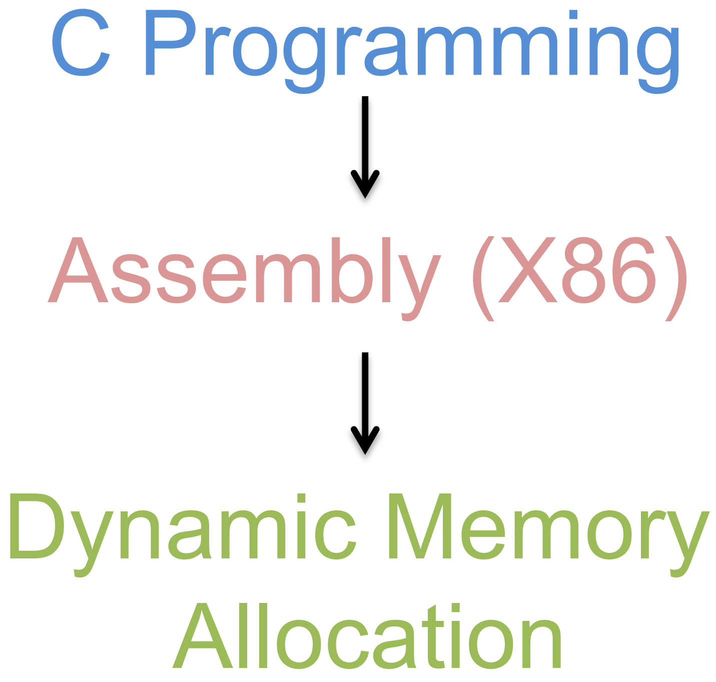
overview
bit, byte and int
float point
[C] basics, bitwise operator, control flow
[C] scopes rules, pointers, arrays
[C] structs, mallocs
[C] large program (linked list)
Machine Prog: ISA, Compile, movq
Machine Prog: Control Code (condition, jump instruction)
Machine Prog: Array allocation and access
Machine Prog: Procedure calls
Machine Prog: Structure, Memory Layout
Machine Prog: Buffer Overflow

C Programming
↓
Assembly (X86)

Schedule

https://nyu-cso.github.io

overview
bit, byte and int
float point
[C] basics, bitwise operator, control flow
[C] scopes rules, pointers, arrays
[C] structs, mallocs
[C] large program (linked list)
Machine Prog: ISA, Compile, movq
Machine Prog: Control Code (condition, jump instruction)
Machine Prog: Array allocation and access
Machine Prog: Procedure calls
Machine Prog: Structure, Memory Layout
Machine Prog: Buffer Overflow
Code optimizations
Dynamic Memory Allocation
Dynamic Memory Allocation continued



C Programming

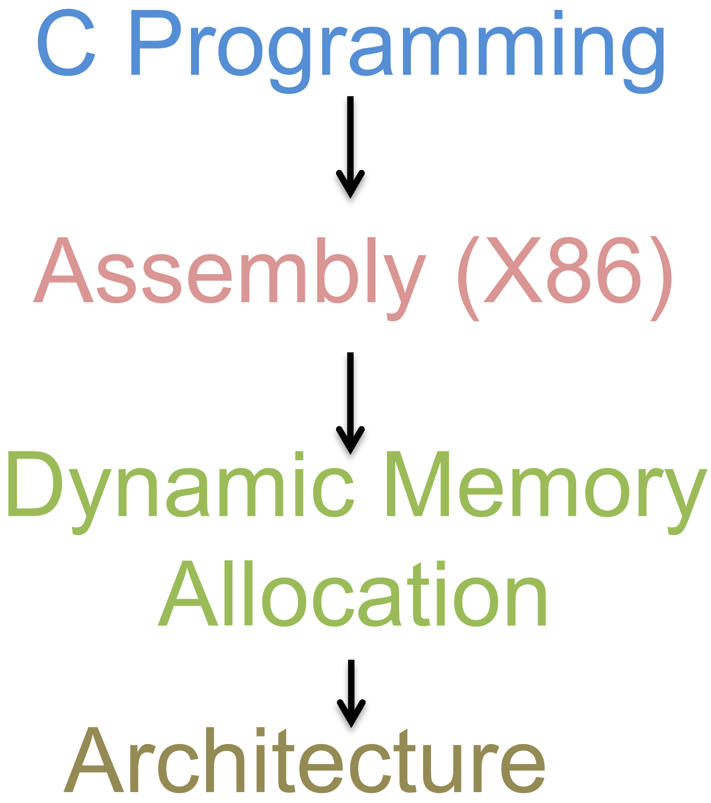
Assembly (X86)

Dynamic Memory Allocation

Schedule

https://nyu-cso.github.io

overview
bit, byte and int
float point
[C] basics, bitwise operator, control flow
[C] scopes rules, pointers, arrays
[C] structs, mallocs
[C] large program (linked list)
Machine Prog: ISA, Compile, movq
Machine Prog: Control Code (condition, jump instruction)
Machine Prog: Array allocation and access
Machine Prog: Procedure calls
Machine Prog: Structure, Memory Layout
Machine Prog: Buffer Overflow
Code optimizations
Dynamic Memory Allocation
Dynamic Memory Allocation continued
Logic Design
Logic Design continued
Sequential implementation
Pipelined implementation



C Programming

Assembly (X86)

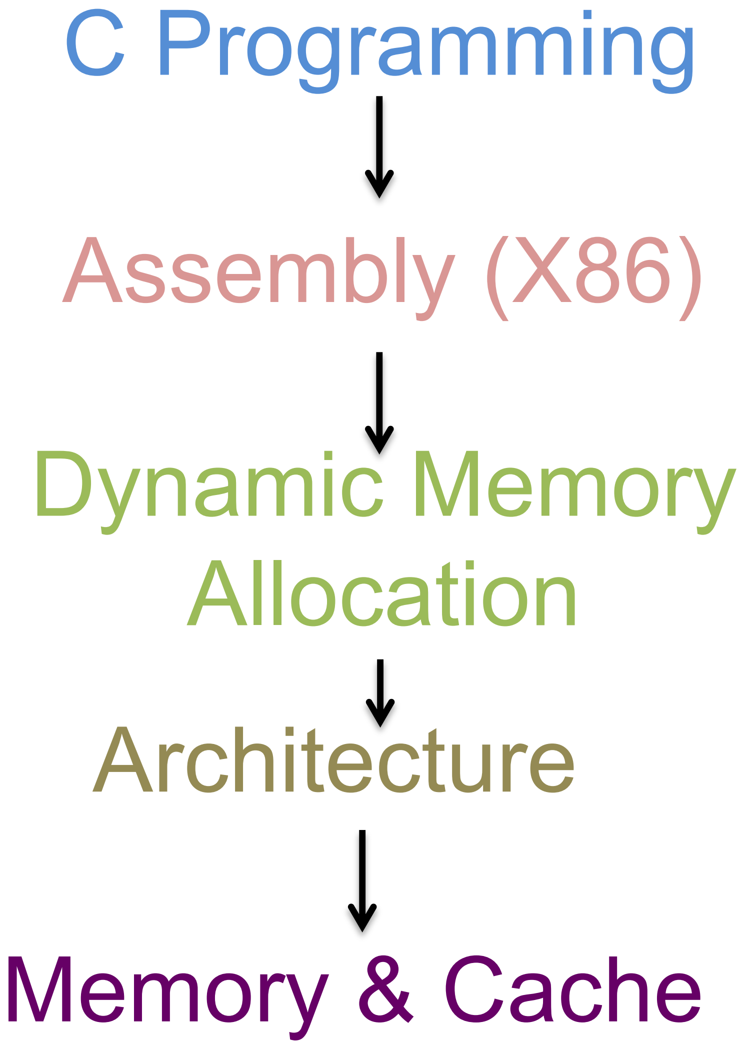
Dynamic Memory Allocation

Architecture

Schedule

https://nyu-cso.github.io

overview
bit, byte and int
float point
[C] basics, bitwise operator, control flow
[C] scopes rules, pointers, arrays
[C] structs, mallocs
[C] large program (linked list)
Machine Prog: ISA, Compile, movq
Machine Prog: Control Code (condition, jump instruction)
Machine Prog: Array allocation and access
Machine Prog: Procedure calls
Machine Prog: Structure, Memory Layout
Machine Prog: Buffer Overflow
Code optimizations
Virtual memory: Address Spaces/ Translation, Goal
Virtual memory: Page table/physcial to virtual
Process
Dynamic Memory Allocation I: malloc, free
Dynamic Memory Allocation II: design allocator
Dynamic Memory Allocation III: futher optimization
Memory, cache
Memory, cache



C Programming

Assembly (X86)

Dynamic Memory Allocation

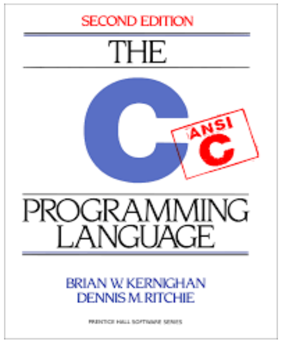
Architecture

Memory & Cache

Course logistics

- Website: https://nyu-cso.github.io
  - Syllabus
    - Reading preparation
    - lecture/recitation slides
    - Lab instructions
- Forum: Campuswire
  - Q&A
- NYU Brightspace
  - Gradescope
    - Lab submission, weekly assessments
  - Zoom links, Zoom recordings
  - Use Campuswire instead of Brightspace for Q&A.

Textbooks



SECOND EDITION

THE C

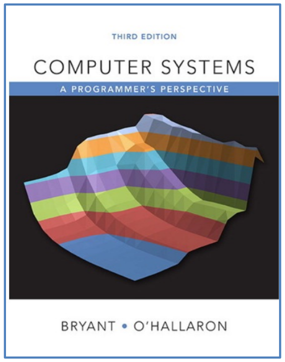
ANSI C

PROGRAMMING LANGUAGE

BRIAN W. KERNIGHAN
DENNIS M. RITCHIE

PRENTICE HALL SOFTWARE SERIES

The C Programming Language 2nd ed,
Kernighan and Ritchie



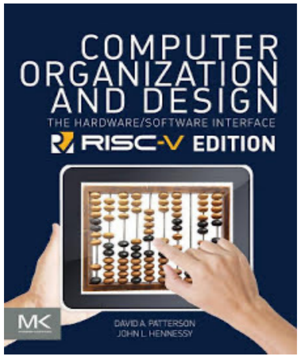
THIRD EDITION

COMPUTER SYSTEMS

A PROGRAMMER'S PERSPECTIVE

BRYANT • O'HALLARON

Computer Systems -- A programmer's perspective,
3rd ed, Bryant and O'Hallaron.



COMPUTER ORGANIZATION AND DESIGN

THE HARDWARE/SOFTWARE INTERFACE

RISC-V EDITION

MK

DAVID A. PATTERSON
JOHN L. HENNESSY

Computer organization and design (RISC-V
edition), Patterson and Hennessy

Grade Breakdown

- 4 programming labs
  - 1% each
- 4 in-person tests
  - Each test is based on the corresponding programming lab plus related concepts.
  - 11% each
- Class participation
  - Recitation participation (5%)
  - Other participation (lecture, online forum): 2%
- In-person final exam (80 minutes)
  - 45%

4 individual programming labs

- Programming environment:
  - Use Courant’s compute server (snappy1)
  - Learn to use:
    - a text editor to write code
    - git for version control
- Optional bonus exercises.
- Submission:
  - Push to github
  - Submit and have it graded via Gradescope
- Late policy:
  - 6 (cumulative) grace days in total over the semester.
  - 3 max. grace days for each lab.

Recitation mini-quiz

- Start next week
- To be done in-class in-person via Gradescope:
  - 15 minutes in the beginning
    - Multiple choice questions and short answers
    - Based on current week’s lecture materials
  - Answers discussed immediately during recitation
- Counts towards recitation participation: 5%

To thrive in CSO, you should …

- Before lecture:
  - Read assigned book chapters
- During lecture/recitation:
  - Ask questions
  - Don’t be shy to ask me to repeat.
- Labs
  - Start early
- Getting help:
  - Campuswire
  - Office hours (see post on Campuswire)

Integrity, Collaboration and AI Policy

Almost all the evaluation is based on in-person test/exam. This means “yes” to:

- All forms of collaboration.
  - E.g. work on programming labs with your buddies
  - Study together etc.
- Unrestricted of AI tools (including coding agents) and traditional web search and other online materials
  - Acknowledge your usage for such tools for each lab.
  - You have no AI nor general internet access during tests/exams.

Warning: Don’t ask AI the instant you get stuck. Think first. Train your brain—or you’ll struggle on exams.

Integrity and Collaboration Policy

We will enforce integrity policy strictly and report violators to the department and Dean.

Do not turn in labs/quiz that are not yours
You won’t fail because of one missing lab/quiz