Computer Systems Organization

https://nyu-cso.github.io

Jinyang Li

Course staff

Lecturer: Prof. Jinyang Li

Zoom recitation instructor:

Ding Ding (PhD student)

In-person recitation instructor:

Jinkun Lin (PhD student)

Course Goal

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

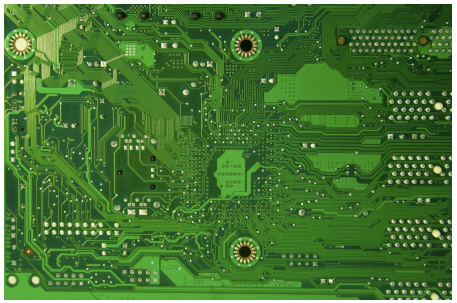
Goal: learn how computers really work

Covered by CSO

To be covered by OS (202)

Components of a computer: hardware

Components of a computer: hardware



Printed Circuit

Components of a computer: hardware + software

Layered Organization

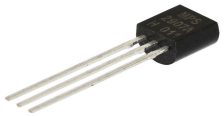
Software

Hardware

Layered Organization

Software

Hardware



Transistors

Diodes

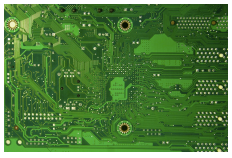
Resistors

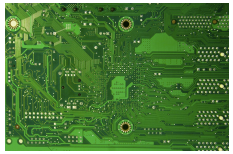
Layered Organization

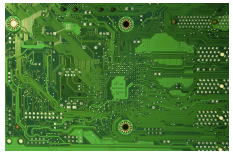
Software

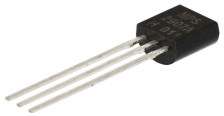
Hardware

Logical Circuits,
Flip-Flops, Gates









Transistors

Diodes

Resistors

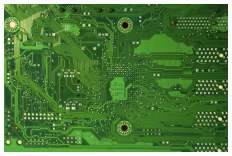
Layered Organization

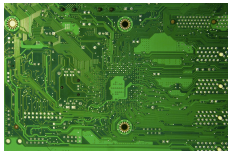
Software

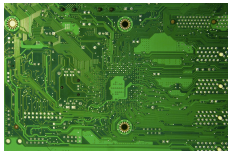
Hardware

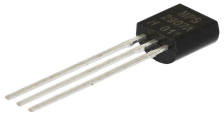
CPU, Memory, Disk

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, …

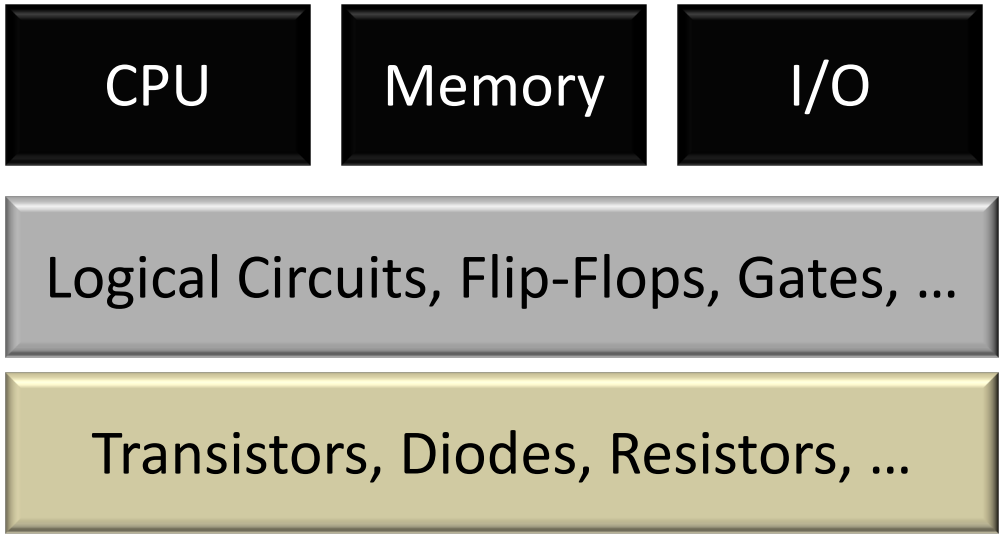
Layered Organization

User Applications

System Software
(OS, compiler, VM…)

Software

Hardware



Layered Organization

Users

User Applications

System Software
(OS, compiler, VM…)

Software

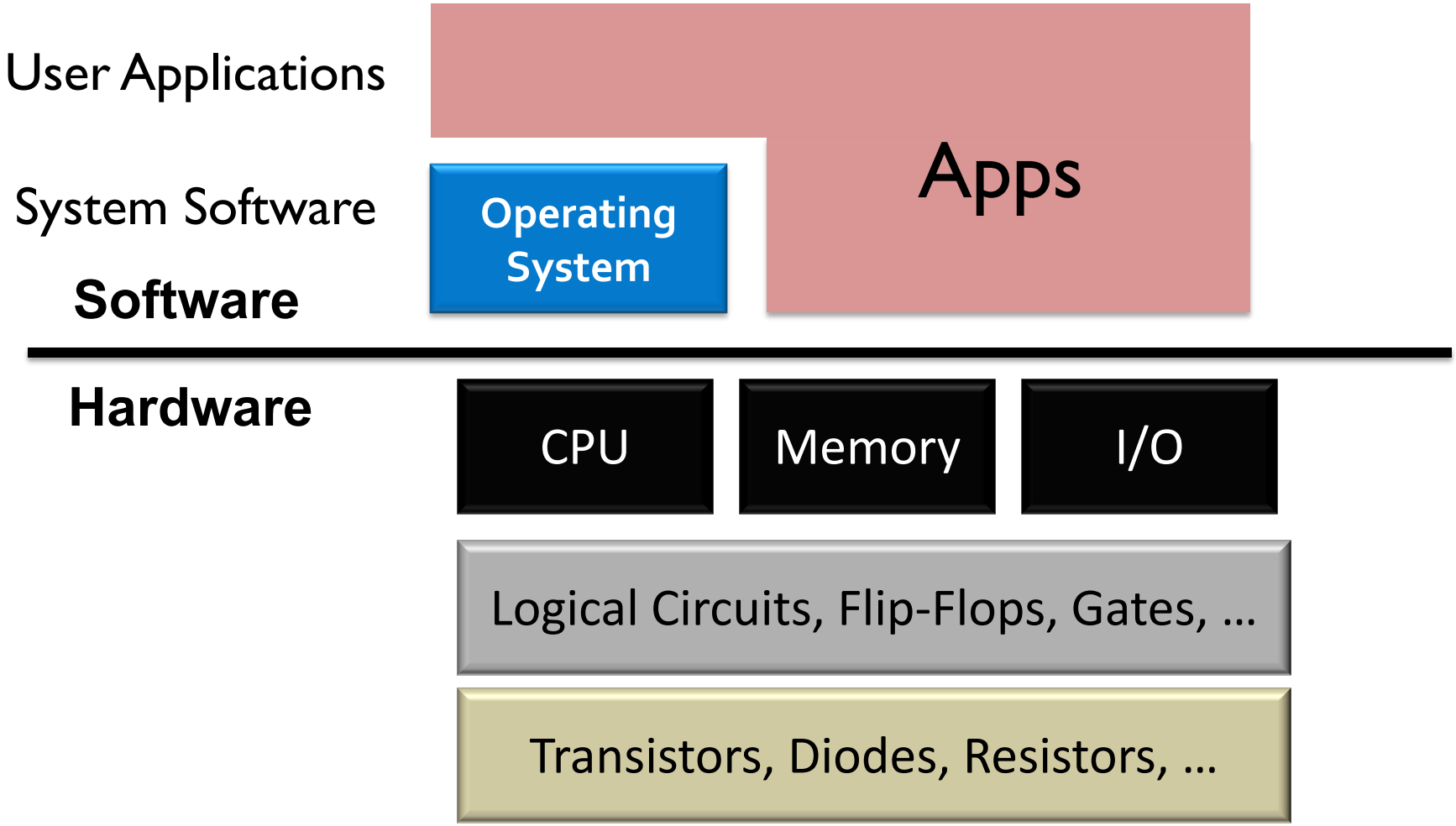
Hardware

CPU | Memory | I/O

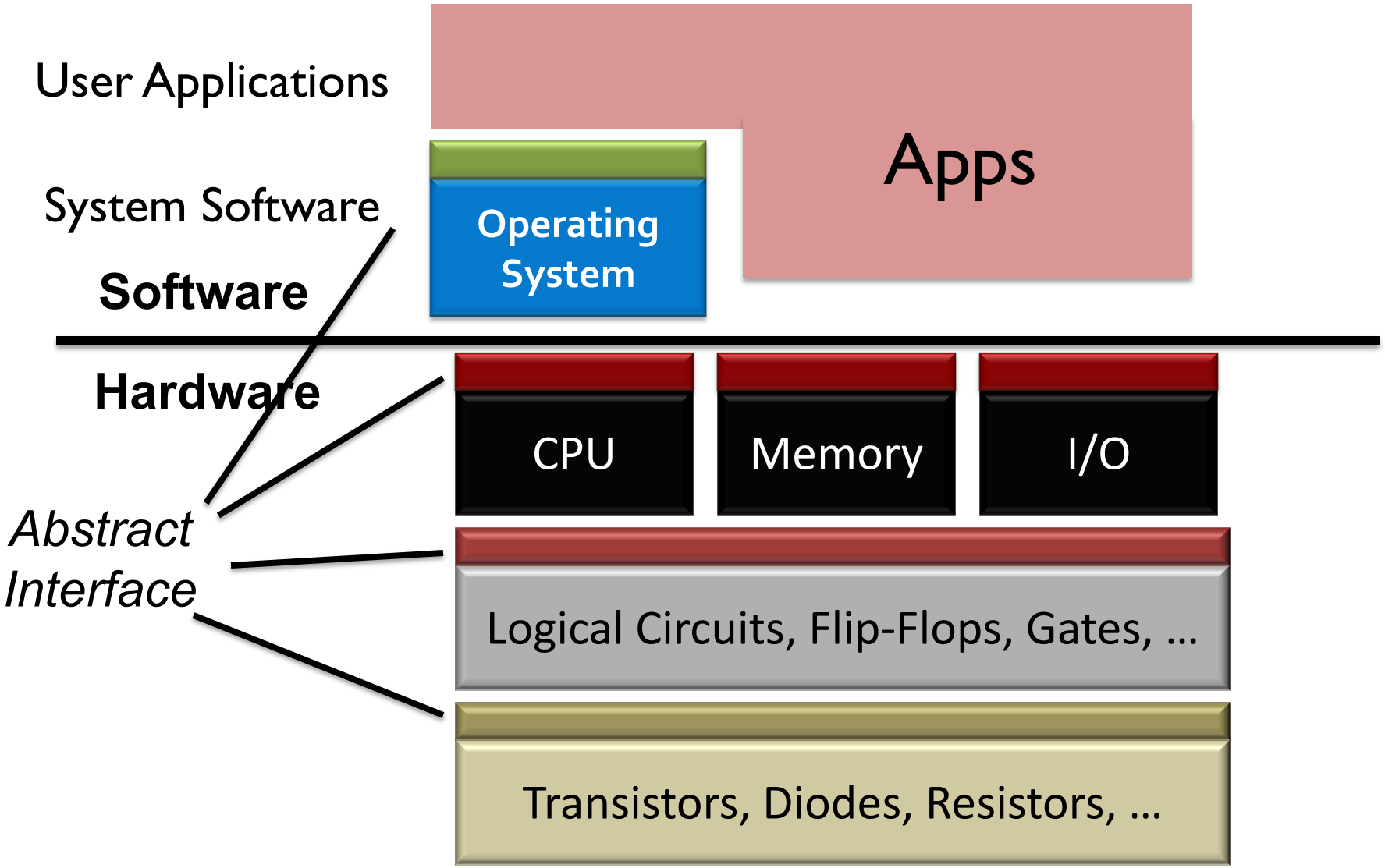
Logical Circuits, Flip-Flops, Gates, …

Transistors, Diodes, Resistors, …

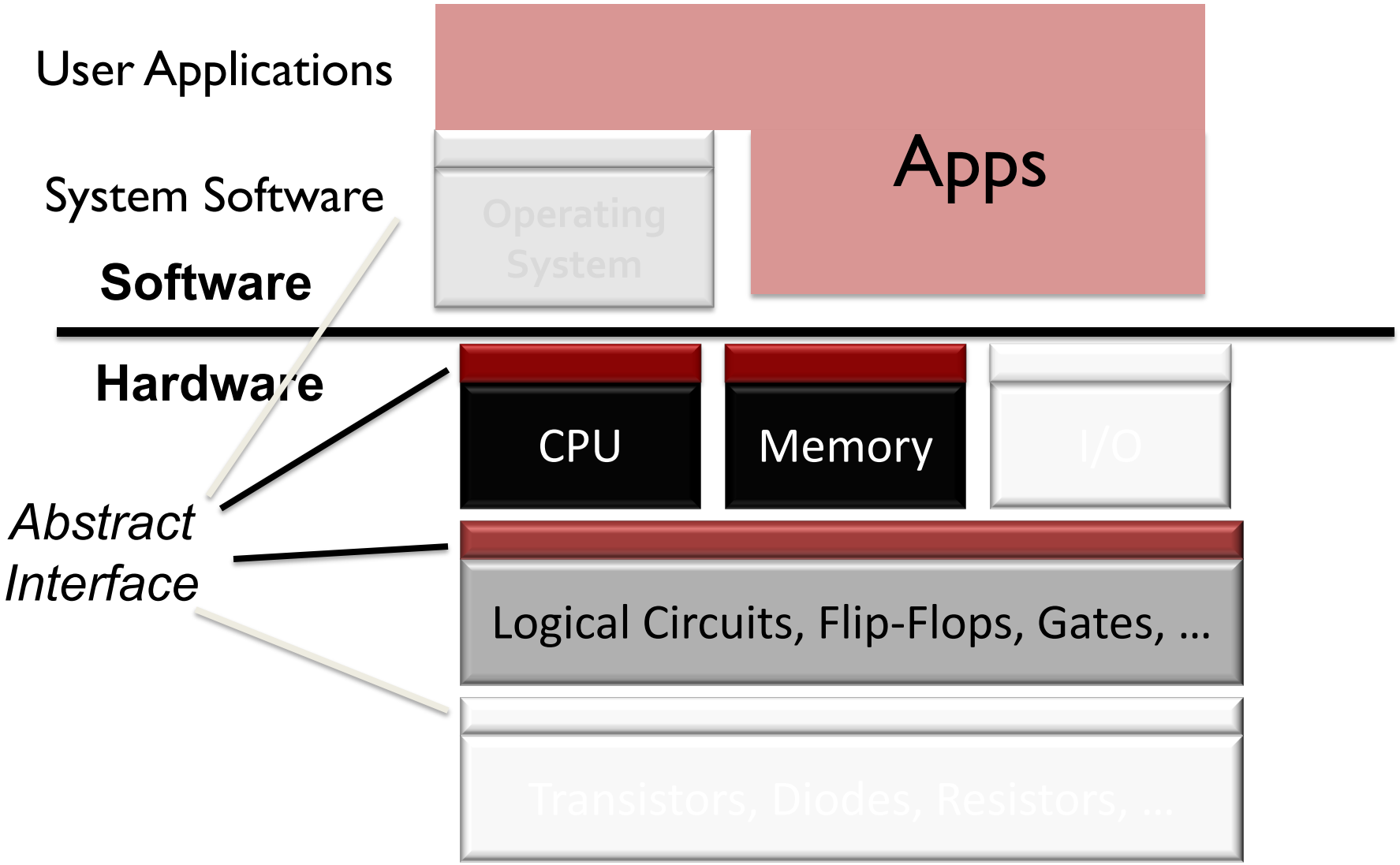
Layered Organization



Abstraction



Scope of this class



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

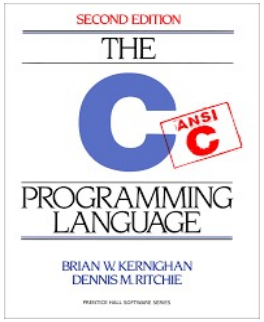
↓

Memory & Cache

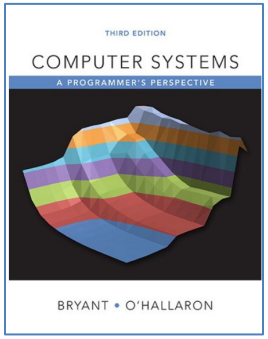
Course logistics

- Lectures (in-person): T/Th 8-9:15am
- Recitation(in-person or zoom): W 8-9:15am
- Website: https://nyu-cso.github.io
  - Syllabus
    - Reading preparation
    - lecture/recitation slides
    - Lab instructions
- Forum: https://campuswire.com/c/G93B4D45D
  - Q&A
- NYU Brightspace
  - Gradescope
    - Lab submission, weekly assessments
  - Zoom links, Zoom recordings
  - Use Campuswire instead of Brightspace for Q&A.

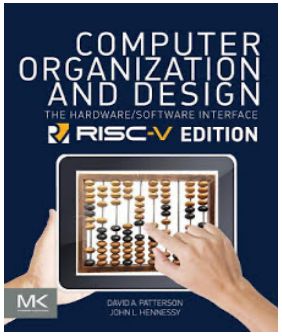
Textbooks



The C Programming Language 2nd ed, Kernighan and Ritchie



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



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

Grade Breakdown

- 6 programming labs
  - Lab-1,2,3: 8%
  - Lab-4,5,6: 9%
- Weekly assessments (take-home)
  - 14 total, starting next week
  - 2% each
- Final exam (80 minutes)
  - 16%
- Participation: 5%
  - Includes participation in lecture, recitation, online forum (Campuswire)

6 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.

Weekly assessment (mini-quiz)

- Start next week
- Done via Gradescope:
  - Multiple choice questions and short answers
  - Mostly on the current week's materials
- Open-book individual assessments
  - Do not consult your classmates or anyone else.
- Quiz duration:
  - 24-hours.
  - Thu 9pm to Fri 9pm (EST). No late submission.
- Answers discussed in the following week's recitation

To survive/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 and weekly assessment.
  - Start early
- Getting help:
  - Campuswire
  - Office hours (TBA next week)

Integrity and Collaboration Policy

1. The work that you turn in must be yours
2. You must acknowledge your influences
   - E.g., if you are inspired by a code snippet, include the URL to the snippet in the lab you turn in.
3. You must not look at, or use, solutions from prior years or the Web, or seek assistance from the Internet
4. You must take reasonable steps to protect your work
   - You must not publish your solutions
5. We reserve the right to randomly pick students for oral assessment and over-weight oral assessment if it does not match your quiz/lab performance.

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