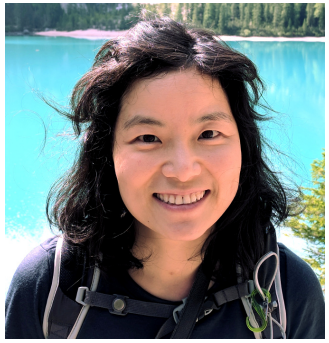
Computer Systems Organization

https://nyu-cso.github.io

Jinyang Li

Course staff

Lecturer: Prof. Jinyang Li



Zoom recitation instructor:

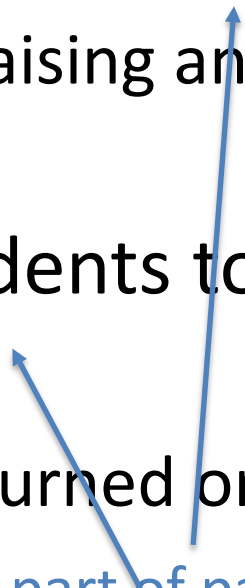
Anqi Zhang (PhD student)

In-person recitation instructor:

Arahant Ashok Kumar (M.S. student)

Zoom lecture

- Recorded.
- You are muted by default.
- For questions, type in (public) chat or raise hands
  - Anqi will monitor chat box / hand raising and interrupt me for Q&A.
- I will randomly select specific students to answer questions.
  - You'll be unmuted and have video turned on



part of participation grade

Course Goal

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

Covered by CSO

To be covered by OS (202)

and a bunch of other stuff

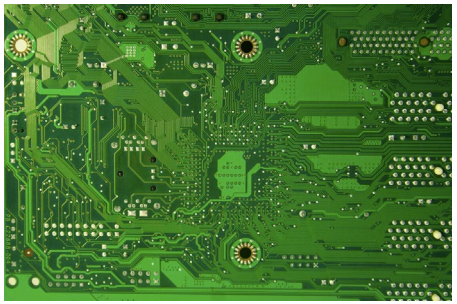
AN x64 PROCESSOR IS SCREAMING ALONG AT BILLIONS OF CYCLES PER SECOND TO RUN THE XNU KERNEL, WHICH IS FRANTICALLY WORKING THROUGH ALL THE POSIX-SPECIFIED ABSTRACTION TO CREATE THE DARWIN SYSTEM UNDERLYING OS X, WHICH IN TURN IS STRAINING ITSELF TO RUN FIREFOX AND ITS GECKO RENDERER, WHICH CREATES A FLASH 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.

Components of a computer: hardware

Components of a computer: hardware



Printed Circuit

Components of a computer: hardware + software

Layered Organization

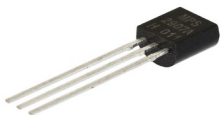
Software

Hardware

Layered Organization

Software

Hardware



Transistors

Diodes

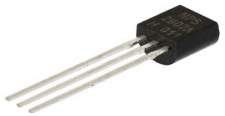
Resistors

Layered Organization

Software

Hardware

Logical Circuits,
Flip-Flops, Gates



Transistors

Diodes

Resistors

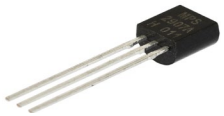
Layered Organization

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

Hardware

CPU | Memory | I/O

Logical Circuits, Flip-Flops, Gates, …

Transistors, Diodes, Resistors, …

Layered Organization

Users

User Applications

System Software
(OS, compiler, VM…)

Software

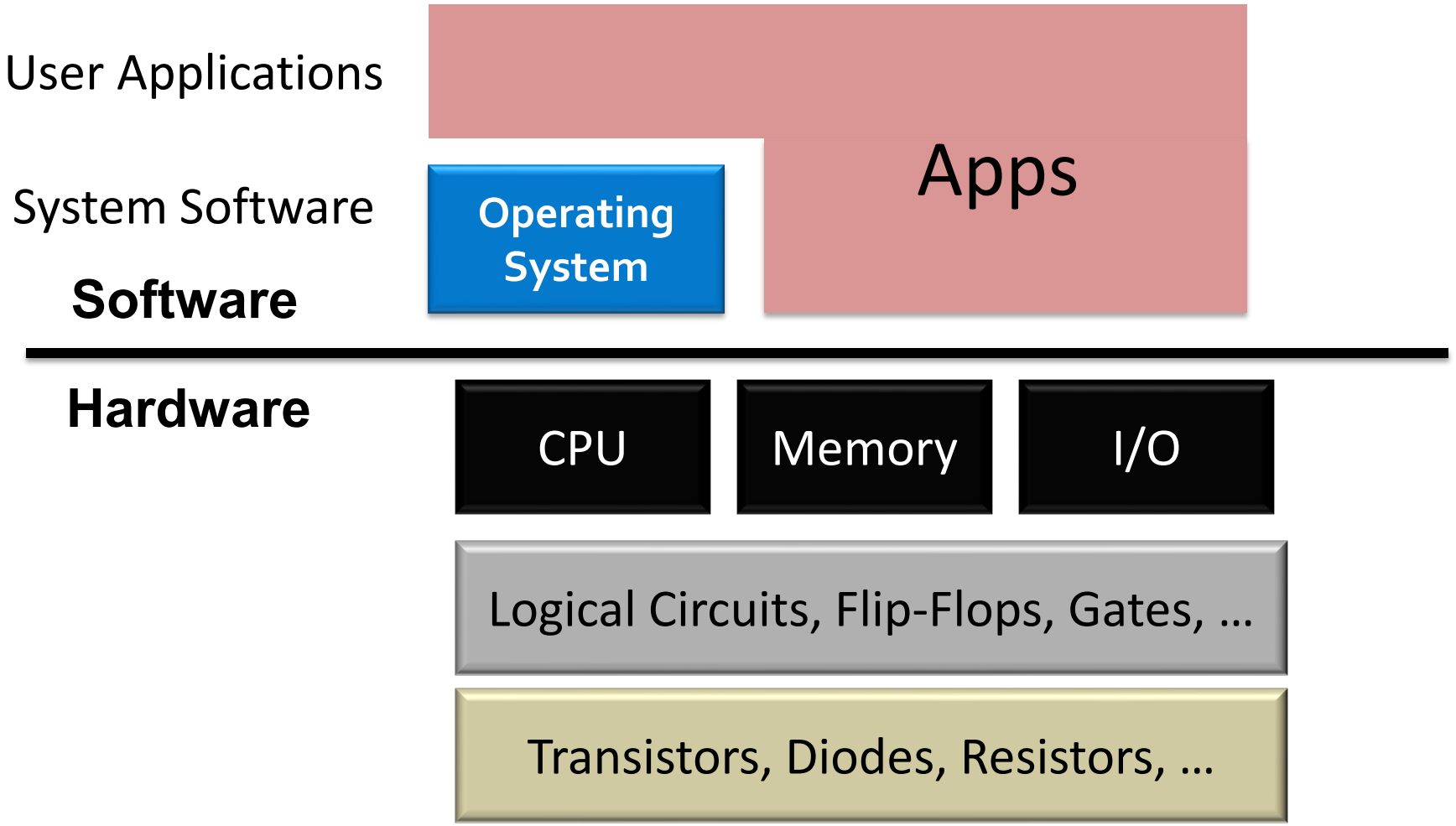
Hardware

CPU | Memory | I/O

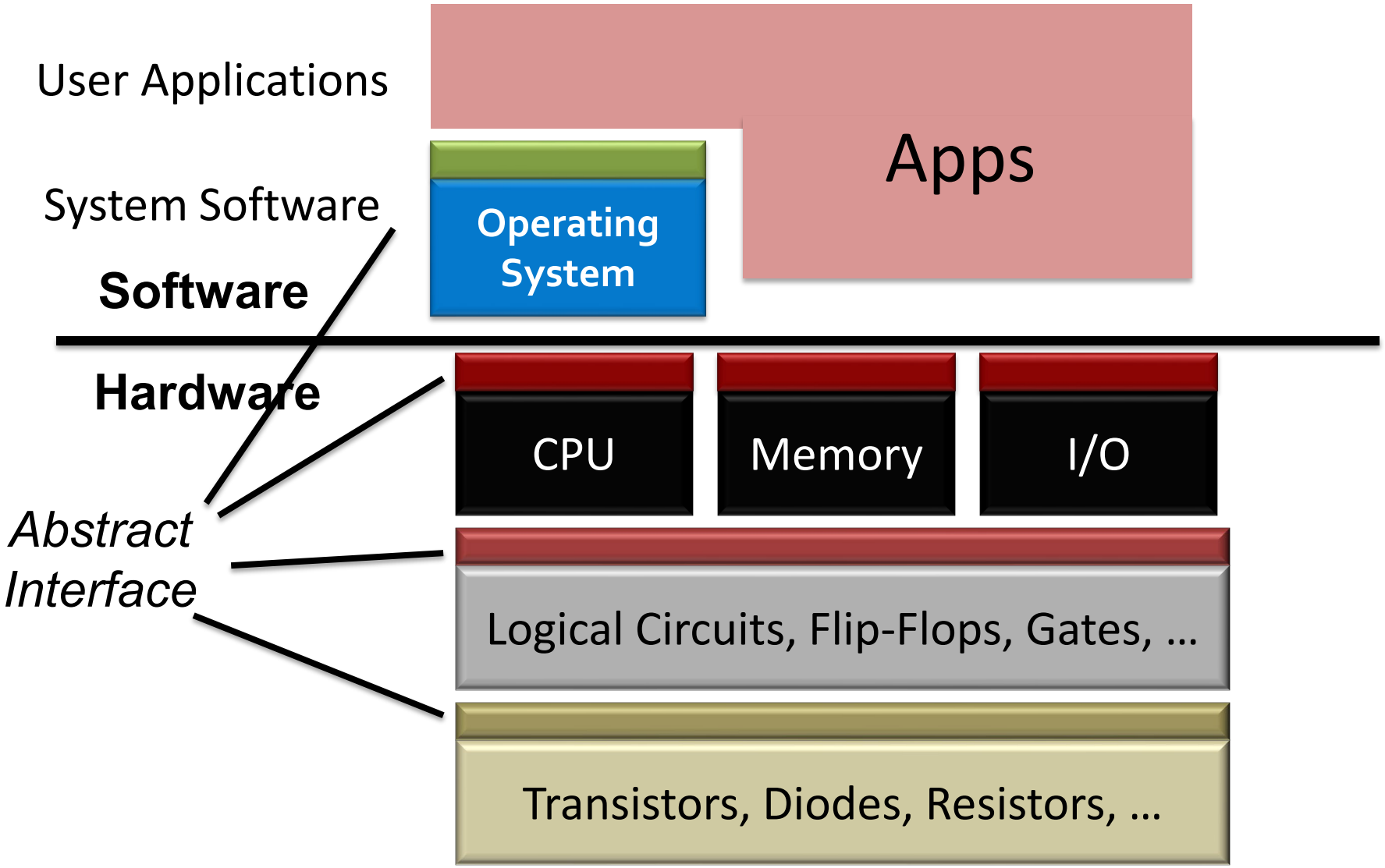
Logical Circuits, Flip-Flops, Gates, …

Transistors, Diodes, Resistors, …

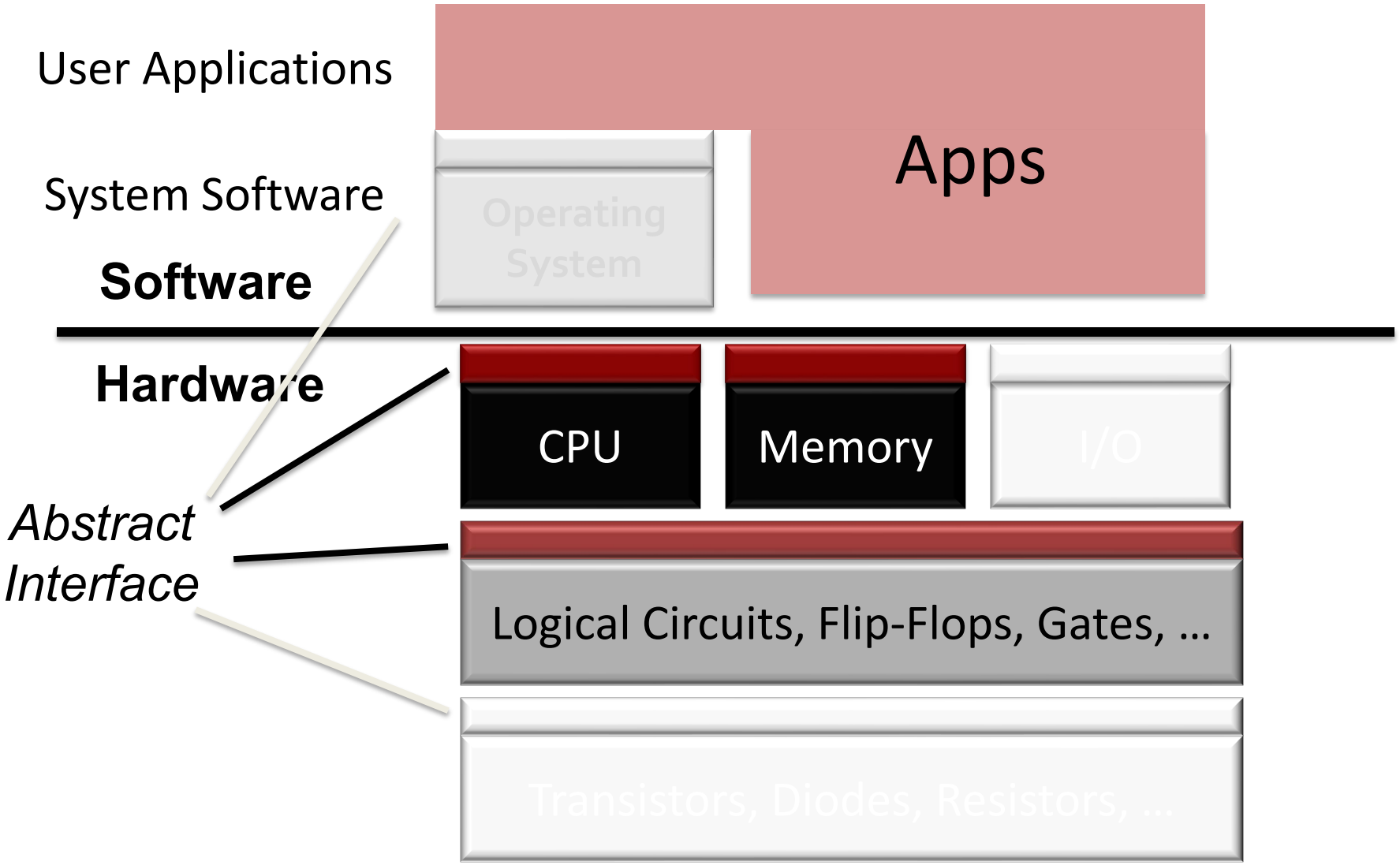
Layered Organization



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)

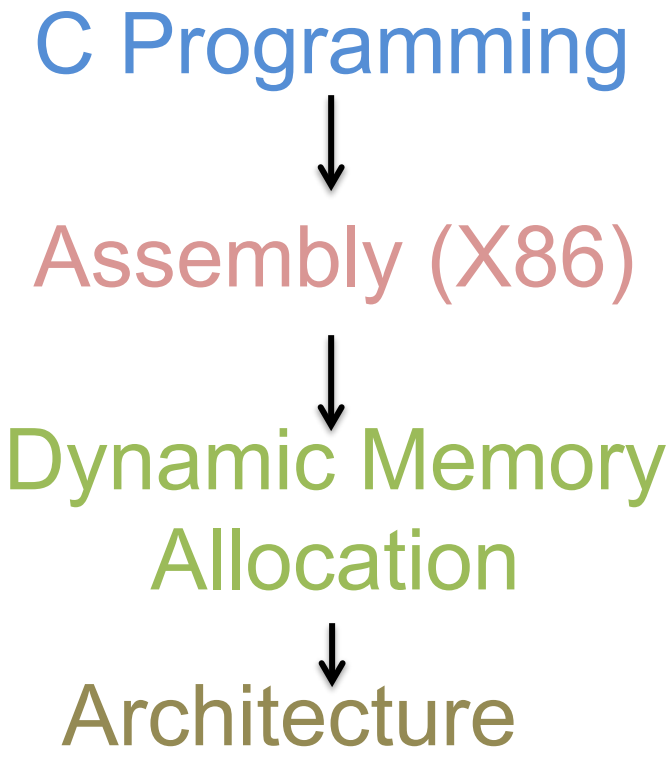
↓

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



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

Breakout activity

- You'll be assigned to a breakout room
- 12 minutes
- TODO:
  - Introduce oneself to each other
  - Nominate a group leader
  - Complete the given Google form as a group.
    - Only leader should submit the form

Course structure

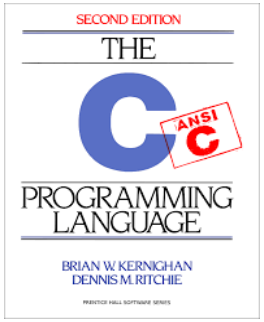
- Zoom Lectures: M/W 12:30-1:45pm
- Zoom Recitation: R 8-9:15am
  - All must attend.
  - Recorded.
- (Optional) In-person recitation: W 11am-12:15pm
  - Only for students registered for *CSCI-UA.0201-010*
  - Content is in addition to those in Zoom recitation.
  - Not recorded.

Starts tomorrow

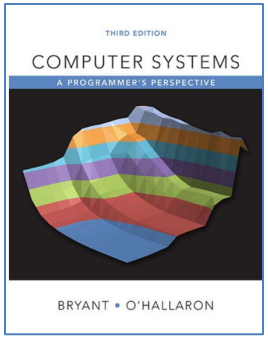
Course websites

- Main website: https://nyu-cso.github.io
  - Syllabus
    - Reading preparation
    - lecture/recitation slides
    - Lab instructions
- Forum: https://campuswire.com/c/G4A62FCF2
  - Q&A
- NYU-classes
  - Zoom links, Zoom recordings
  - Gradescope
    - Lab submission, weekly assessments
  - Use Campuswire instead of NYU-classes for Q&A.

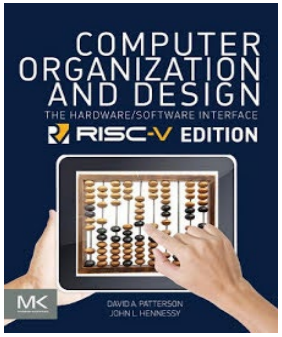
Textbooks



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 assessment
  - 14 total, starting next week
  - 3% each
- Participation: 7%
  - Includes participation in lecture, recitation, online forum (Campuswire)

6 individual programming labs

- Programming environment:
  - Virtual machine running on your laptop
  - 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)

- Starting 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.
- Answered discussed in the following week's zoom recitation

To do well in CSO, you should …

Before lecture:

Read assigned book chapters before lecture

Duration lecture/recitation:

Ask me to repeat, repeat and repeat

– Ask questions

– Answer questions from me or others

Befriend your classmates and form study groups

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 later this 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