Computer Systems Organization

Jinyang Li

Course staff

Lecturer: Prof. Jinyang Li
Research: distributed systems

Recitation instructor:

John Westhoff (2nd PhD student)

Research: distributed systems

TA:

Itamar Reif (senior)

Past CSO survivor

Important notice

- We are experimenting with a new curriculum!
- How this affects you:
  - This class is different from other CSO sections
  - This class prepares you for "Operating Systems" (Spring 2020) taught by Prof. Aurojit Panda

CS majors: Register for Prof. Panda's OS in the spring!!

Course Goal

- How does a computer really 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.

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

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? New!
   - 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 workload

- Lectures: M/W 12:30am-1:45pm, CIWW 102
- Recitation: R 8:00-9:15am, CIWW 101

Starts tomorrow! Bring a laptop to class

- Homework:
  - Recitation exercises
  - 5 Programming labs
- Quizzes:
  - Quiz 1
  - Quiz 2
  - Final exam

Grade Breakdown

- Recitation exercises 12%
- Labs 43%
  - Breakdowns: 7%, 7%, 7%, 11%, 11%
- Quiz-1 10%
- Quiz-2 15%
- Final 20%

Bonus I: lecture and piazza participation 5%

Bonus II: extra-credit lab questions

Optional class project

- For extra challenge, do a class project instead of taking the final.
- To qualify for the project option, you must:
  - Form a team of 2
  - Score >= 85% on quiz-1, 2
  - Finish lab-1, 2, 3, 4 before 11/19.
- How your project will be evaluated:
  - Presentation + demo

Is CSO going to be hard?

Statistics from last term: Final

It’s time to work hard

You must be prepared to devote >15 hours/week

Statistics from this term: major vs. minor

Statistics of this term:

Before Class

Read the related sections in the text books

"The C Programming Language, 2nd Edition", Prentice Hall, 1988, Reserved at NYU library

Computer organization and design (RISC-V edition)

Be Active In Class

Raise your hand at any time

- Ask me to repeat, repeat and repeat
- Ask questions
- Answer questions from me or others

Befriend your classmates and form study groups

After Class

Finish all labs / recitation exercises by yourself

Attend recitations

Getting help

- Office hour, Piazza

Policies

You must work alone on all assignments

- You may post questions on Piazza,
- You are encouraged to answer others' questions, but refrain from explicitly giving away solutions.

Labs & Exercises

- Everybody has 5 grace days (incremented at half day granularity)

Class Info

https://nyu-cso.github.io

Recitation starts tomorrow,
bring your laptop to class

Integrity and Collaboration Policy

We will enforce the policy strictly.

1. The work that you turn in must be yours
2. You must acknowledge your influences
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. If there are inexplicable discrepancies between exam and lab performance, we will over-weight the exam and interview you

Do not turn in labs/exercises that are not yours

You won't fail because of one missing lab

Integrity and Collaboration Policy

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