Computer Systems Organization

Jinyang Li

Some slides adapted from Tiger Wang's

Why study CSO?

Your career path in the next few years

graduation

interview

programmer

Your career path in the next few years

graduation

interview

apply for graduate school

programmer

programmer & researcher

Your career path in the next few years

graduation

startup

interview

graduate school

programmer & researcher

programmer

Hire / become

Your career path in the next few years

Your career path in the next few years

graduation

enjoy life

lawyer

startup

interview

graduate school

programmer & researcher

Hire / become

Be able to

programmer

Your career path in the next few years

graduation

interview

graduate school

startup

lawyer

enjoy life

programmer & researcher

Hire / become

Be able to

be in a relationship with

programmer

Taking CSO will affect each step in the path!

For Graduation

Required class

- For CS major and minor

Prepare for more advanced system classes

- Operating Systems
- Compilers
- Networks
- Architecture
- Distributed Systems

For Interview

This class adds to your CV

- C Programming, UNIX, X86 Assembly …

Interview related topics

- Basic knowledge of Array, String, Bit Manipulation

Topics Distribution From LeetCode

Topics ~30%

Some exercises in this class are derived from the real interview questions!

Our text books are considered as the bibles of job interview.

For Graduate School Application

This class adds to your CV

- A

Research related topics

- Performance optimization
  - Memory layout and allocation, concurrent programming
- Security
  - Buffer Overflow

Startup

The life you imagine

CEO

CTO

CFO

COO

Startup

Your real life: full stack programmer

Server

Website

Phone's App

Optimizations

My lawyer friend

Numerous hours spent on extracting information from documents

My lawyer friend

I want to study programming.

My lawyer friend

I want to study programming.

Ok, you need to study CSO first.

My lawyer friend

I want to study programming.

Ok, you need to study CSO first.

Hmm…, I want to marry a programmer.

My lawyer friend

I want to study programming.

Ok, you need to study CSO first.

Hmm…, I want to marry a programmer.

Ok, you need to study CSO first.

My lawyer friend

I want to study programming.

Ok, you need to study CSO first.

Hmm…, I want to marry a programmer.

Ok, you need to study CSO first.

…The user is offline

Conversation between programmers

Have you heard of the Meltdown attack?

No. Is it bad?

Meltdown lets an attacker read another process' address space!

What is an address space?

He does not know anything about computers...

Sorry I have to run now, bye!

For Programming

Understand how your program runs on the hardware

- Why it fails
- Why it is slow

Why it fails?

What is the result of 1000,000 * 1000,000 ?

Why it fails?

What is the result of 1000,000 * 1000,000 ?

Expected answer: 1000,000,000,000 (1 trillion)

Why it fails?

What is the result of 1000,000 * 1000,000 ?

Expected answer: 1000,000,000,000 (1 trillion)

```
int main()
{
   int a = 1000000;
   int b = 1000000;
   int r = a * b;
   printf("result is %d\n", r);
   return 0;
}
```

Why it is slow?

Example Matrix Multiplication

Both implementations have exactly the same operations count ($2n^3$)

What is CSO about?

Computer System Organization

Computer System Organization

Printed Circuit

Computer System Organization

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

The Scope of This Class

The Scope of This class

Focus on abstract interfaces exposed by

- CPU and Memory
- Operating System, Compilers

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
Code optimizations

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
Virtual memory: Address Spaces/ Translation, Goal
Virtual memory: Page table/physcial to virtual
Process

C Programming

↓

Assembly (X86)

↓

Virtual Memory

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

C Programming

↓

Assembly (X86)

↓

Virtual Memory

↓

Memory Management

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
Concurrent Programming I: thread, race
Concurrent Programming II: lock
Concurrent Programming III: conditional variable
Concurrent Programming IV: Other primitives

C Programming
↓
Assembly (X86)
↓
Virtual Memory
↓
Memory Management
↓
Concurrent Programming

Course Perspective

Most Systems Courses are Builder-Centric

- Computer Architecture
  - Design pipelined processor in Verilog
- Operating Systems
  - Implement large portions of operating system
- Compilers
  - Write compiler for simple language
- Networking
  - Implement and simulate network protocols

Course Perspective

Most Systems Courses are Builder-Centric

- Computer Architecture
  - Design pipelined processor in Verilog
- Operating Systems
  - Implement large portions of operating system
- Compilers
  - Write compiler for simple language
- Networking
  - Implement and simulate network protocols

Course Perspective (Cont.)

This course is programmer-centric

- Understanding of underlying system makes a more effective programmer
- Bring out the hidden hacker in everyone

Course workload

- Lectures: M/W 11am-12:15pm, 60 5th Ave C12
- Recitation: T 9:30-10:45am, 60 5th Ave C10

Starts tomorrow! Bring Laptop to class

- Homework:
  - Recitation exercises
    - to be finished on the day of recitation
  - 5 Programming labs

- Quizzes:
  - Quiz 1 (3/4)
  - Quiz 2 (4/3)
  - Final exam (5/15)

Grade Breakdown

Recitation exercises 15%

Labs 40%

Quiz-1 10%

Quiz-2 15%

Final 20%

Bonus I: lecture and piazza participation 5%

Bonus II: extra-credit lab questions

Optional class project

- To challenge yourself more, do a class project instead of taking quiz-2 and final.
- To qualify for the project option, you must:
  - Form a team of 2
  - Score >= 85% on quiz-1
  - Finish lab-1, 2, 3 before 3/13.
- 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 from this term: student level

We are here to help

Jinyang Li

Lecturer

Lingfan Yu

Recitation Leader
Head grader

Jingyu Liu

Grader

cso-staff@cs.nyu.edu

Before Class

Read the related sections in the text books

"Computer Systems: A Programmer's Perspective, 3nd Edition", http://csapp.cs.cmu.edu

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

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 the 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.