Computer Systems Organization

Jinyang Li

Slides are based on Tiger Wang’s class
Why study CSO?
The path of your next few years

- graduation
- interview
- programmer
The path of your next few years

- **Graduation**
- **Interview**
- **Apply for graduate school**
- **Programmer**
- **Programmer & researcher**
The path of your next few years

Hire / become

programmer & researcher

startup

graduate school

interview

graduation
The path of your next few years

- Graduation
- Interview
- Graduate school
- Programmer & researcher
- Startup
- Lawyer

Hire / become
Be able to
The path of your next few years

- graduation
- interview
- graduate school
- programmer & researcher
- programmer
- startup
- lawyer
- enjoy life

Be able to hire / become

~2M programmers in 2014 according to IDC
The path of your next few years

graduation

interview

graduate school

startup

lawyer

programmer & researcher

Hire / become

programmer

Enjoy life

Be able to

be in a relationship with
Taking CSO will affect each step in the path!
For Graduation

Required class
  – For CS Major
  – Also for CS minor 😞

Prepare for your later system classes
  – Operating Systems, Compilers, Networks, Computer 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
Some examples and 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, code optimization, memory allocation, concurrent programming

- Security
  - Buffer Overflow
Startup

The life you imagine
Startup
Your real life: full stack programmer

Server
Website
Phone’s App
Optimizations
My lawyer friend

Take >10 hours each day to extract information from the 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 1,000,000 * 1,000,000?
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)

```c
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

- System Fan
- Floppy
- Heat Sink
- Hard Drive
- Optical Drive
- Power Supply
- Motherboard
- Processors (CPU)
- RAM Modules
Computer System Organization

- System Fan
- Floppy
- Heat Sink
- Hard Drive
- Optical Drive
- Printed Circuit
- Power Supply
- Motherboard
- Processors (CPU)
- RAM Modules
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

**Hardware**
- Transistors, Diodes, Resistors, ...
- Logical Circuits, Flip-Flops, Gates, ...
- CPU
- Memory
- I/O

**Software**
Layered Organization

Hardware

Software

System Software (OS, compiler, VM...)

Transistors, Diodes, Resistors, …

Logical Circuits, Flip-Flops, Gates, …

CPU
Memory
I/O
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

Operating System
Compilers
...

CPU
Memory
I/O

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

Transistors, Diodes, Resistors, ...

Layered Organization

Hardware

Software

System Software

User Applications

User App

Operating System
Compilers
...

CPU
Memory
I/O

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

Transistors, Diodes, Resistors, ...
Abstraction

User Applications

System Software

Software

Hardware

Abstract Interface

User App

Operating System

Compilers

...
The Scope of This Class

User Applications

System Software

Software

Hardware

Abstract Interface

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

Transistors, Diodes, Resistors, ...

User App

Operating System

Compilers

CPU

Memory

I/O
The Scope of This class

Focus on abstract interfaces exposed by
– CPU and Memory
– Operating System, Compilers

<table>
<thead>
<tr>
<th>System Software</th>
<th>C Programming, OS Service, Memory Management, Concurrent Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>Operating Systems and Compilers</td>
</tr>
<tr>
<td>Hardware</td>
<td>Assembly, Virtual memory, Interrupt</td>
</tr>
<tr>
<td></td>
<td>CPU and Memory</td>
</tr>
</tbody>
</table>
Schedule of Our Class

http://news.cs.nyu.edu/~jinyang/sp18-cso/schedule.html

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)
Schedule of Our Class

http://news.cs.nyu.edu/~jinyang/sp18-cso/schedule.html

- 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
Schedule of Our Class

http://news.cs.nyu.edu/~jinyang/sp18-cso/schedule.html

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/physical to virtual
Process

C Programming
Assemble (X86)
Virtual Memory
Schedule of Our Class

http://news.cs.nyu.edu/~jinyang/sp18-cso/schedule.html

overview
bit, byte and int
float point
[C] basics, bitwise operator, control flow
[C] scopes rules, pointers, arrays
[C] structs, m allocate
[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: further optimization
# Schedule of Our Class

http://news.cs.nyu.edu/~jinyang/sp18-cso/schedule.html

<table>
<thead>
<tr>
<th>Overview</th>
<th>C Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit, byte and int</td>
<td>Assembly (X86)</td>
</tr>
<tr>
<td>Float point</td>
<td>Virtual Memory</td>
</tr>
<tr>
<td>[C] basics, bitwise operator, control flow</td>
<td>Memory Management</td>
</tr>
<tr>
<td>[C] scopes rules, pointers, arrays</td>
<td>Concurrent Programming</td>
</tr>
<tr>
<td>[C] structs, mallocs</td>
<td></td>
</tr>
<tr>
<td>[C] large program (linked list)</td>
<td></td>
</tr>
<tr>
<td>Machine Prog: ISA, Compile, movq</td>
<td></td>
</tr>
<tr>
<td>Machine Prog: Control Code (condition, jump instruction)</td>
<td></td>
</tr>
<tr>
<td>Machine Prog: Array allocation and access</td>
<td></td>
</tr>
<tr>
<td>Machine Prog: Procedure calls</td>
<td></td>
</tr>
<tr>
<td>Machine Prog: Structure, Memory Layout</td>
<td></td>
</tr>
<tr>
<td>Machine Prog: Buffer Overflow</td>
<td></td>
</tr>
<tr>
<td>Code optimizations</td>
<td>Memory Management</td>
</tr>
<tr>
<td>Virtual memory: Address Spaces/ Translation, Goal</td>
<td>Concurrent Programming</td>
</tr>
<tr>
<td>Virtual memory: Page table/physical to virtual</td>
<td></td>
</tr>
<tr>
<td>Process</td>
<td>Concurrent Programming</td>
</tr>
<tr>
<td>Dynamic Memory Allocation I: malloc, free</td>
<td></td>
</tr>
<tr>
<td>Dynamic Memory Allocation II: design allocator</td>
<td></td>
</tr>
<tr>
<td>Dynamic Memory Allocation III: further optimization</td>
<td></td>
</tr>
<tr>
<td>Concurrent Programming I: thread, race</td>
<td></td>
</tr>
<tr>
<td>Concurrent Programming II: lock</td>
<td></td>
</tr>
<tr>
<td>Concurrent Programming III: conditional variable</td>
<td></td>
</tr>
<tr>
<td>Concurrent Programming IV: Other primitives</td>
<td></td>
</tr>
</tbody>
</table>
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
This course is **programmer-centric**

- Understanding of underlying system makes a more effective programmer
- Bring out the hidden hacker in everyone
To be a happy programmer, you should

Attend
- Lectures (T/Th 11:00-12:15pm)
- Recitation (M 3:30-4:45 pm)
  - In-class exercises provide hands-on instruction

Do
- 5 Programming labs
- Recitation exercises

Pass
- Quiz 1 (2/27)
- Quiz 2 (3/27)
- Final exam
Grade Breakdown

Recitation and Exercises 15%
Labs 40%
Quiz before midterm 10%
Midterm 15%
Final 20%

Bonus I: lecture and piazza participation 5%
Bonus II: extra-credit lab questions (points vary)
Is CSO going to be hard?
- Senior: 23.5% (20 students)
- Freshman: 7.1% (6 students)
- Sophomore: 30.6% (26 students)
- Junior: 38.8% (33 students)
Time to work hard

We Can Do It!

We (the course staff) are here to help
Who are we?

Jinyang Li
Lecturer

Chien-chin Huang
Recitation Leader
Head grader

Gu Jin
Grader

Chengchen Li
Grader

Zekun Zhang
Grader

cso-staff@cs.nyu.edu
Before Class

Read the related sections in the text books

http://csapp.cs.cmu.edu

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

Have discussion and make friends with each others
After Class

Finish all labs / exercises
  – By yourself

Attend the recitations
  – Any issue of doing labs or exercises

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
  – Assignments due at 11:59pm on the due date
  – Everybody has 5 grace days
  – Zero score after the due
Class Info

http://news.cs.nyu.edu/~jinyang/sp18-cso/

Recitation starts next Mon
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 labs
Integrity and Collaboration Policy

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