Computer Systems Organization

Instructor:
Zhaoguo Wang
Why choose CSO?
The path of your next few years

1. Graduation
2. Interview
3. Programmer
The path of your next few years

1. Graduation

   - Interview
   - Apply for graduate school

   - Programmer
   - Programmer & researcher
The path for your next few years:

- Graduation
- Interview
- Graduate school
- Startup
- Programmer & researcher
- Programmer
- Hire / become
The path of your next few years

graduation

interview

graduate school

programmer & researcher

Hire / become

Be able to

programmer

startup

lawyer
The path of your next few years

graduation

interview

graduate school

startup

lawyer

enjoy life

programmer & researcher

Hire / become

Be able to

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

- graduation
- interview
- graduate school
- programmer & researcher
- Hire / become programmer
- startup
- lawyer
- 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
Complaints from a professor teaching Operating System

After 7 years teaching OS class, this is the first time I meet some students who even do not have any basic background knowledge. This is because they have never taken CSO before!!!
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

– 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 have a programmer boyfriend.
My lawyer friend

I want to study programming.

Ok, you need to study CSO first.

Hmm..., I want to have a programmer boyfriend.

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 have a programmer boyfriend.
Ok, you need to study CSO first.

...The user is offline
Conversation with a programmer

Do you program?

I do not program a lot. But I play with assembly sometime.

Let’s date !!!
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 $1000,000 \times 1000,000$?

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
- Power Supply
- Motherboard
- Processors (CPU)
- RAM Modules

Printed Circuit
Computer System Organization

- System Fan
- Floppy
- Print Sink
- Hard Drive
- Printed Circuit
- Power Supply
- Adobe
- Optical Drive
- Motherboard
- Processors (CPU)
- RAM Modules
- DirectX
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, ...)  Hardware

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\cdots)

Software

Hardware

CPU  Memory  I/O

Logical Circuits, Flip-Flops, Gates, \cdots

Transistors, Diodes, Resistors, \cdots
Layered Organization

User Applications

System Software

Software

Hardware

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

...

CPU

Memory

I/O

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

Transistors, Diodes, Resistors, ...
The Scope of This Class

- **Hardware**
  - Transistors, Diodes, Resistors, ...
  - Logical Circuits, Flip-Flops, Gates, ...
  - Transistors, Diodes, Resistors, ...
- **Software**
  - User Applications
  - System Software
  - Operating System
  - Compilers
- **Abstract Interface**
  - CPU
  - Memory
  - I/O
The Scope of This class

Focus on abstract interfaces exposed by

– CPU and Memory
– Operating System, Compilers

System Software

Software

Hardware

C Programming, OS Service, Memory Management, Concurrent Programming

Operating Systems and Compilers

Assembly, Virtual memory, Interrupt

CPU and Memory
Schedule of Our Class


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


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


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
Schedule of Our Class


C Programming
- Assembly (X86)
- Virtual Memory
- Memory Management

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

Dynamic Memory Allocation I: malloc, free
- Dynamic Memory Allocation II: design allocator
- Dynamic Memory Allocation III: further optimization
## Schedule of Our Class


<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></td>
</tr>
<tr>
<td>Virtual memory: Address Spaces/ Translation, Goal</td>
<td></td>
</tr>
<tr>
<td>Virtual memory: Page table/physical to virtual</td>
<td></td>
</tr>
<tr>
<td>Process</td>
<td></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
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 Perspective (Cont.)

This course is programmer-centric:

– Understanding of underlying system makes a more effective programmer
– Bring out the hidden hacker in everyone
This course is programmer-centric – understanding of underlying system makes a more effective programmer – bring out the hidden hacker in everyone.
This course is programmer-centric – understanding of underlying system makes a better programmer – bring out the hacker in everyone
This course is programmer-centric—understanding of underlying system makes a more effective programmer.

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

Enjoy Life!
To be a happy programmer, you need to

Take
- Lectures (M/W 3:30-4:45pm)
- Recitation (Thu 12:30-1:45 pm)
  - In-class exercises provide hands-on instruction
  - Please make sure you will attend tomorrow’s recitation

Do
- 5 (or 6) Programming labs
- Exercises

Pass
- 1 Quiz (before the mid-term)
- 1 Mid-term
- 1 Final exam
Grade Breakdown

Lecture  10%
Recitation and Exercises 15%
Labs 30%
Quiz before midterm 10%
Midterm 15%
Final  20%

Please make sure you will attend tomorrow’s recitation!
What are the challenges?
Challenges
Challenges

- Computer science: 26
- Other major/undecided: 12
Don’t Worry

We are here to help
Who are we?

Zhaoguo Wang
Lecturer
zhaoguo@nyu.edu

Chien-chin Huang
Recitation Leader / Grader
huang@cs.nyu.edu

HungWei Chen
Grader
hc2264@nyu.edu
Gossip Time

Chien-chin Huang

– 5th years PH.D, a perfect husband and father
Gossip Time

Chien-chin Huang
- 5th years PH.D, a perfect husband and father

HungWei Chen
- GPA: 4.0/4.0
- 8 years working experience
  - 4 years staff software engineer
Gossip Time

Chien-Chin Huang
– 5th years PH.D, a perfect husband and father

HungWei Chen
– GPA: 4.0/4.0
– 8 years working experience
  • 4 years staff software engineer

Zhaoguo Wang
– no gossip… 😊
Office Hours

Zhaoguo Wang
  – Monday 5:00 PM – 6:00 PM, 60 FA, Room 406

Chien-Chin Huang and HungWei Chen
  – Thursday 3:00 PM – 4:00 PM, 60 FA, Room 406
Who are we?
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
Reminder: please attend tomorrow's recitation!
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.
Integrity and Collaboration Policy

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