

Computer Systems Organization

Shuai Mu

Slides are based on Tiger Wang's and Jinyang Li'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 for your next few years



graduation



startup

interview

graduate school



programmer & researcher



programmer

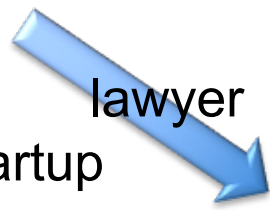


Hire / become

The path () for next few years

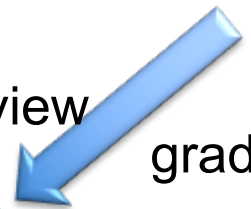


graduation



lawyer

startup



interview

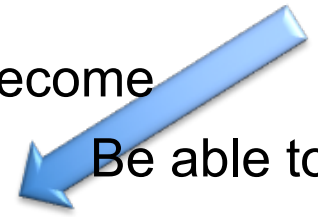


graduate school



programmer & researcher

Hire / become



Be able to

programmer



The path () your next few years



graduation

enjoy life

lawyer

startup

graduate school

interview



programmer & researcher

Hire / become

Be able to

programmer



~2M programmers in 2014 according to IDC

The path () your next few years



graduation

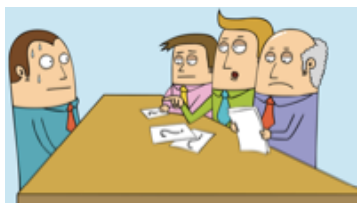
enjoy life

lawyer

startup

graduate school

interview



ENJOY ♥ LIFE



programmer & researcher

Hire / become

Be able to

programmer



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

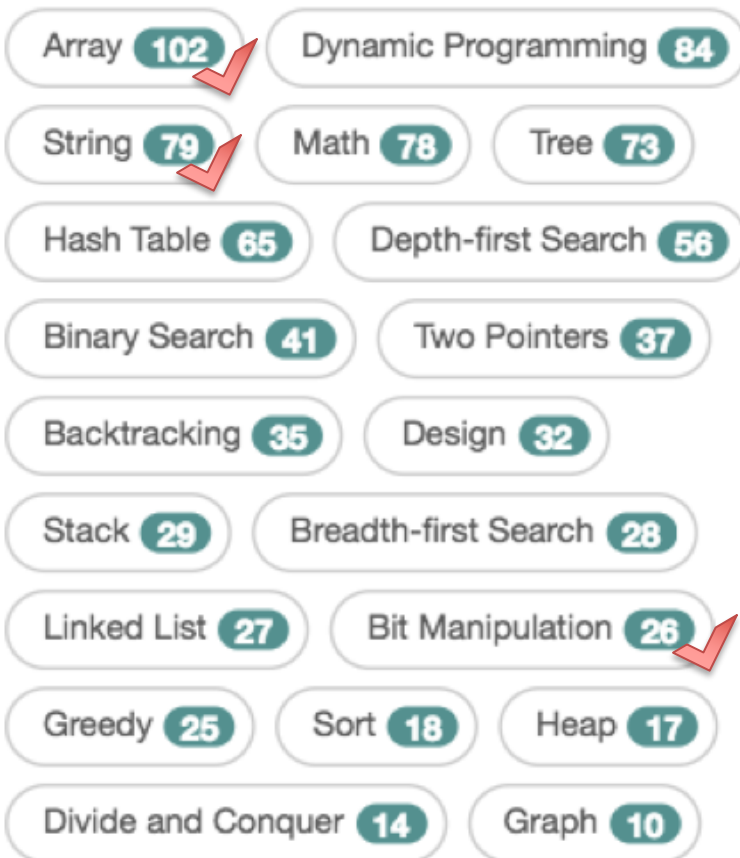
- C Programming, UNIX, X86 Assembly ...

Interview related topics

- Basic knowledge of Array, String, Bit Manipulation

Topics Distribution From LeetCode

🔖 Topics ~30%



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



CEO

CTO

CFO

COO

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 $1000,000 * 1000,000$?

Why it fails?

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

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

Why it fails?

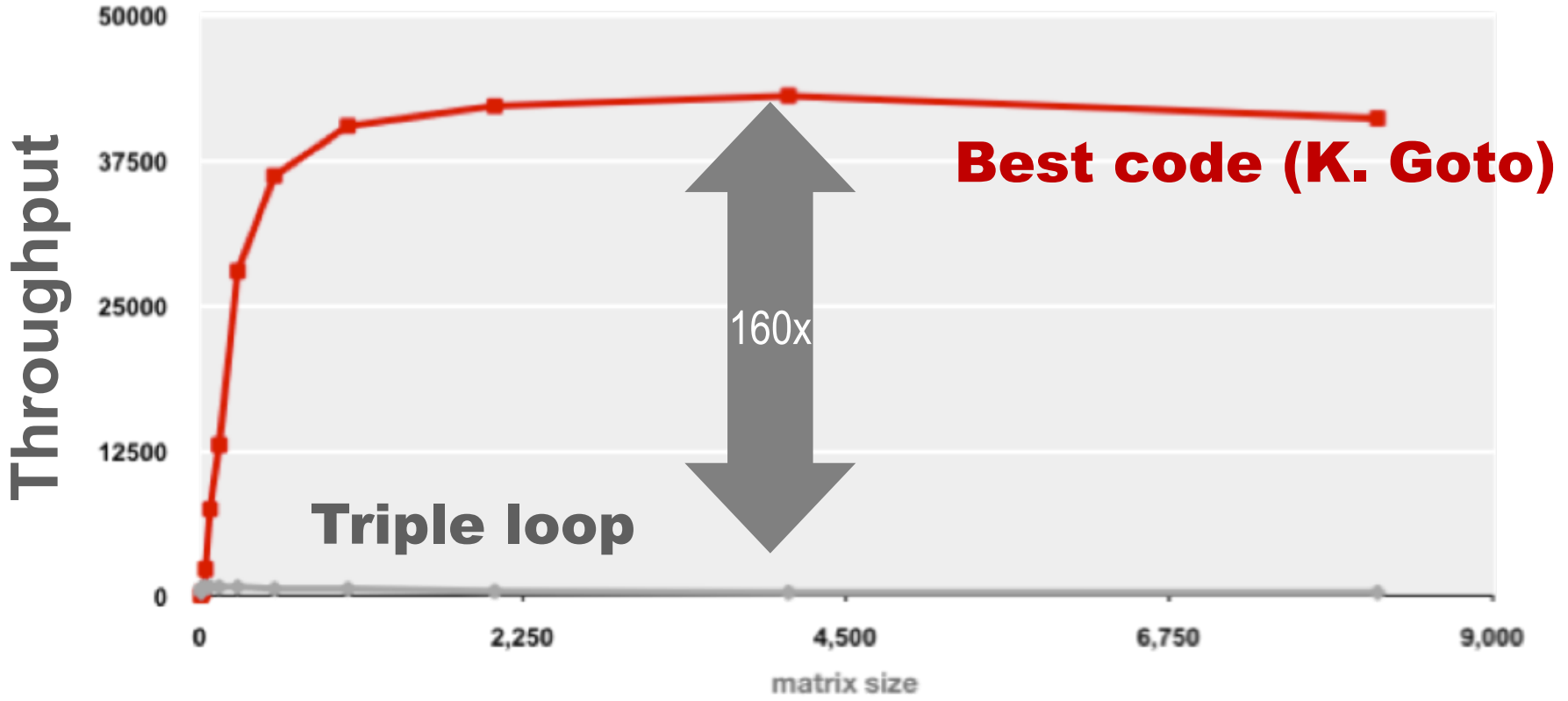
What is the result of $1000,000 * 1000,000$?

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

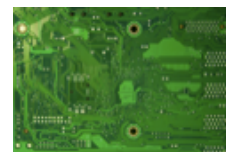
Layered Organization

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

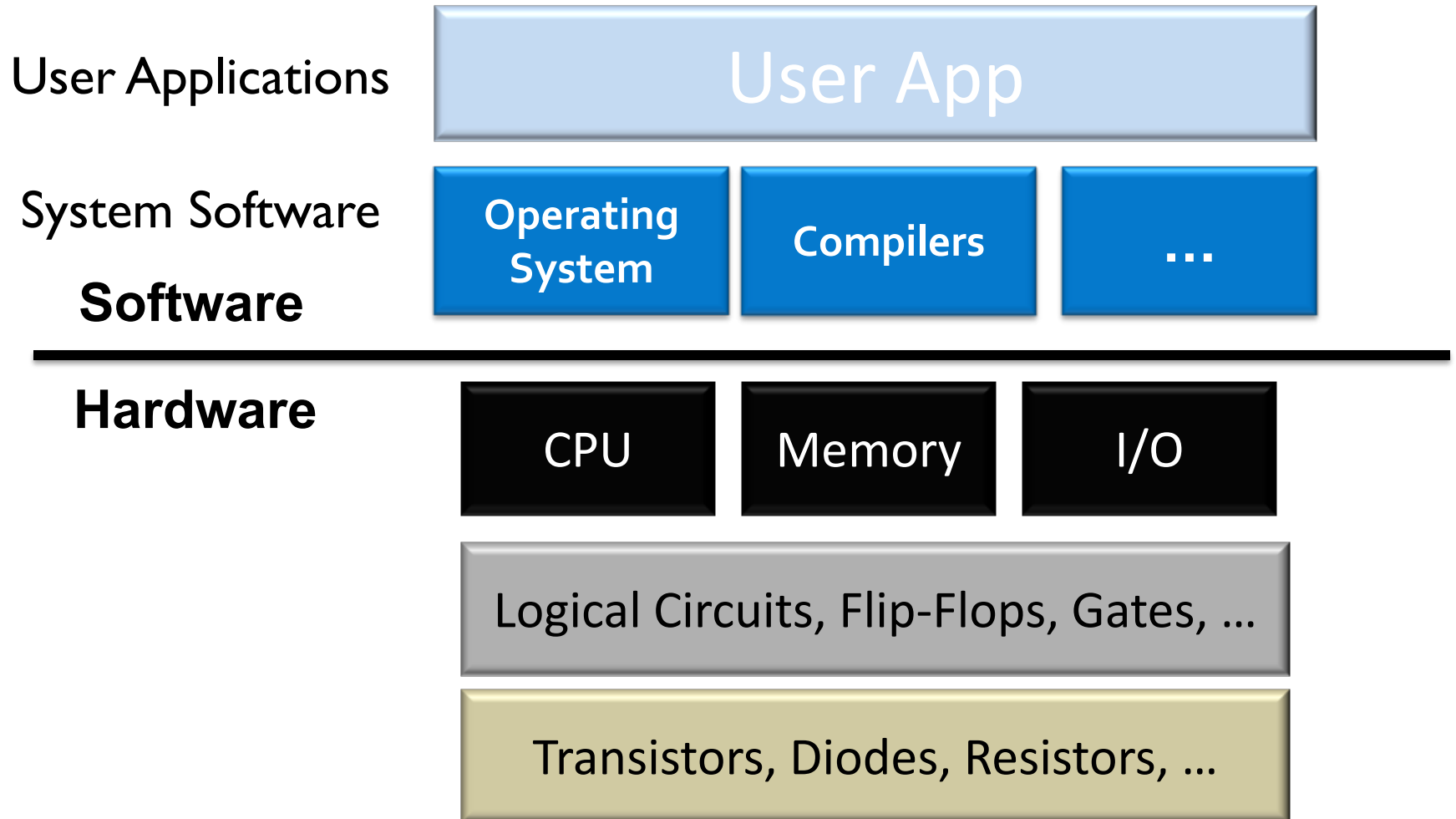
Memory

I/O

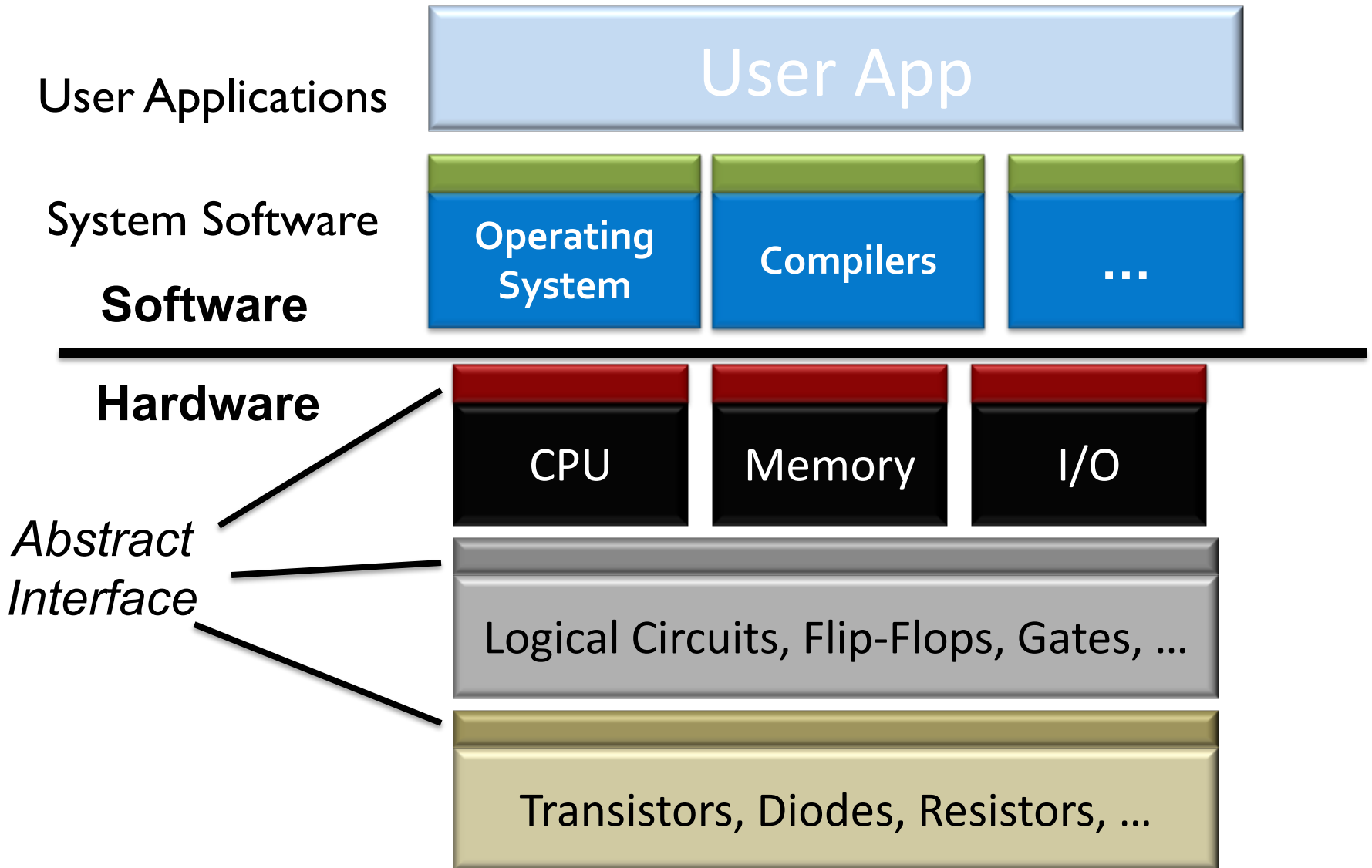
Logical Circuits, Flip-Flops, Gates, ...

Transistors, Diodes, Resistors, ...

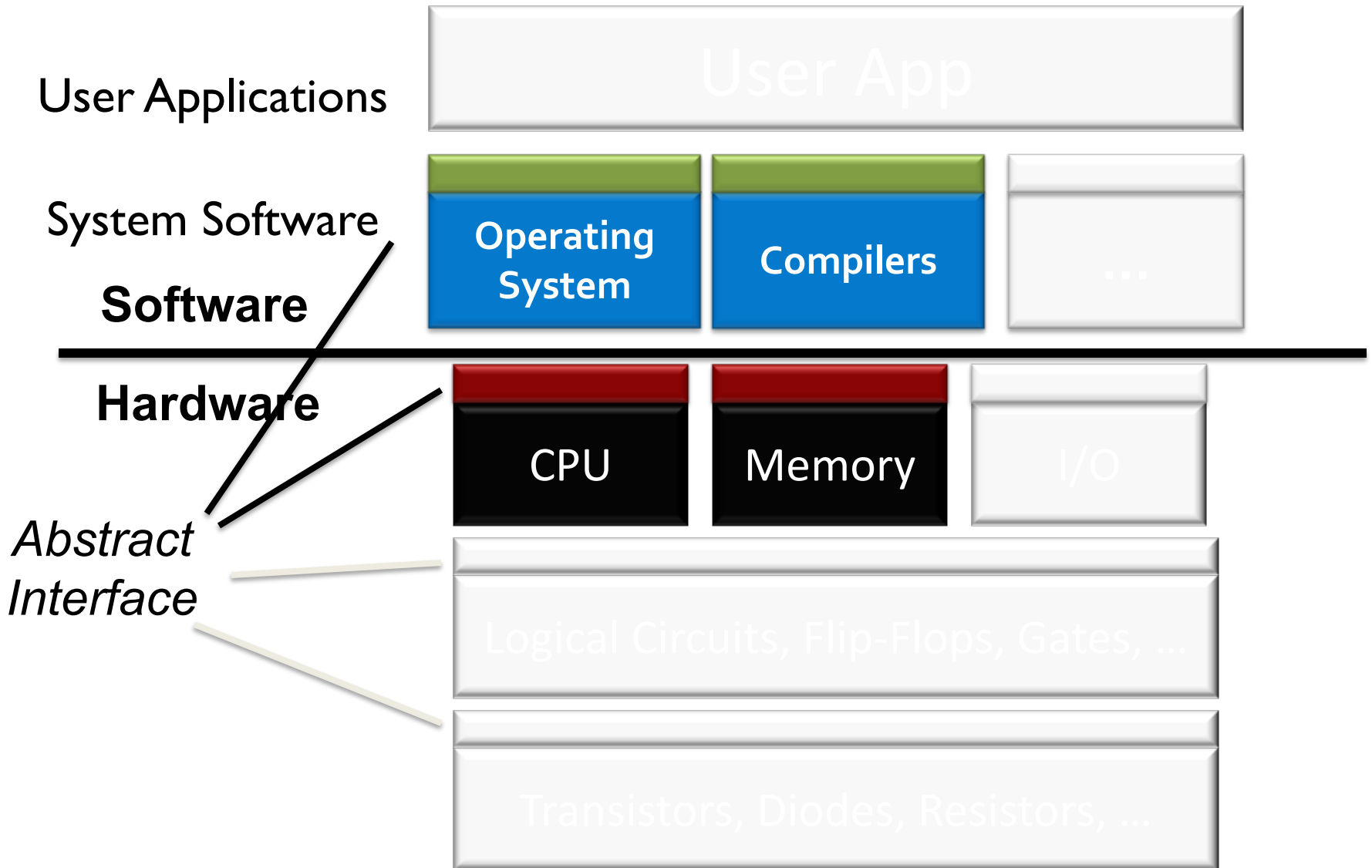
Layered Organization



Abstraction



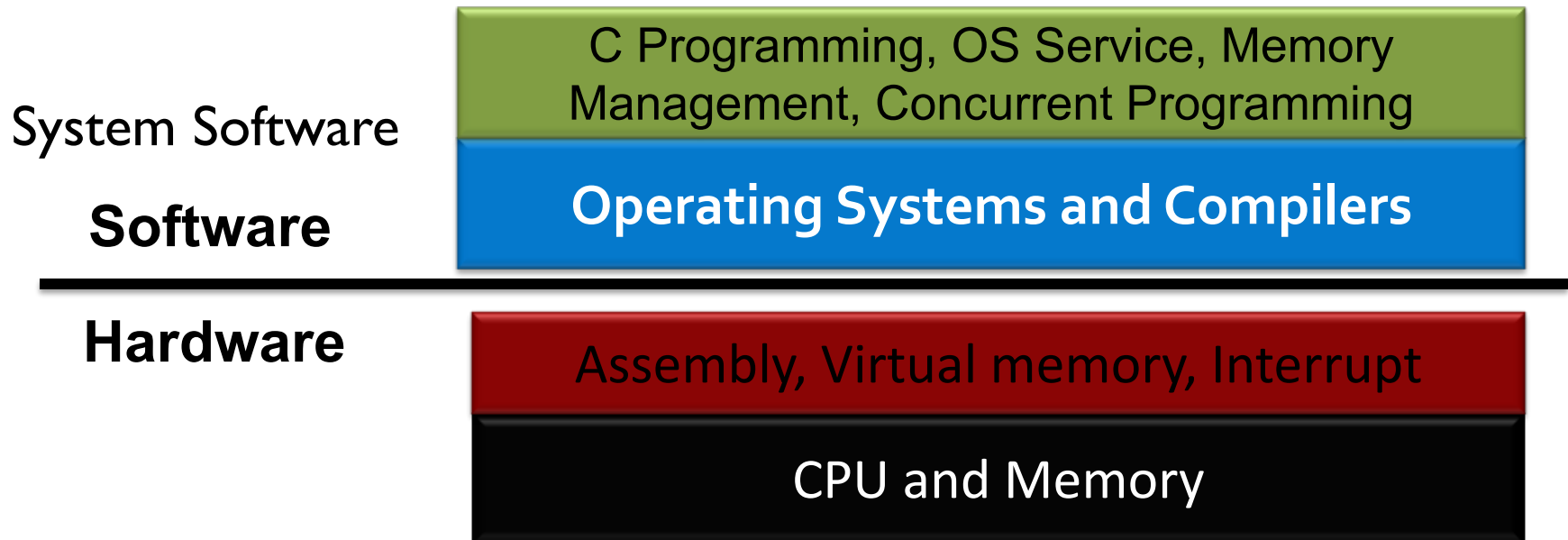
The Scope of This Class



The Scope of This class

Focus on abstract interfaces exposed by

- CPU and Memory
- Operating System, Compilers



Schedule of Our Class

<http://mpaxos.com/teaching/cso18spring/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)

C Programming

Schedule of Our Class

<http://mpaxos.com/teaching/cso18spring/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

C Programming



Assembly (X86)

Schedule of Our Class

<http://mpaxos.com/teaching/cso18spring/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



Assembly (X86)



Virtual Memory

Schedule of Our Class

<http://mpaxos.com/teaching/cso18spring/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

Dynamic Memory Allocation I: malloc, free

Dynamic Memory Allocation II: design allocator

Dynamic Memory Allocation III: further optimization

C Programming



Assembly (X86)



Virtual Memory



Memory Management

Schedule of Our Class

<http://mpaxos.com/teaching/cso18spring/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

Dynamic Memory Allocation I: malloc, free

Dynamic Memory Allocation II: design allocator

Dynamic Memory Allocation III: further 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



To be a happy programmer, you should

Attend

- Lectures (T/R 2:00-3:15pm)
- Recitation (W 8:00-9:15 am)
 - 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 1 10%

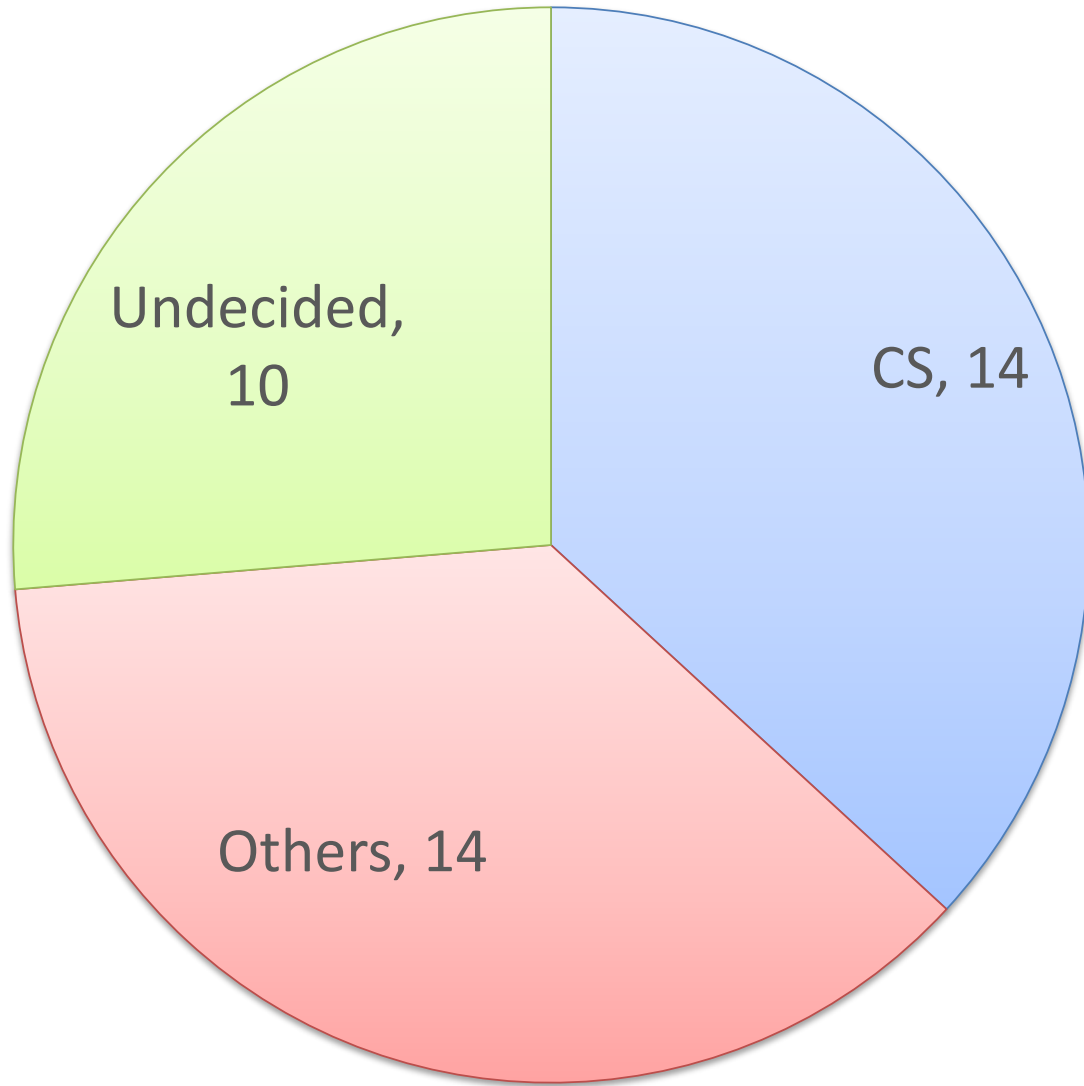
Quiz 2 15%

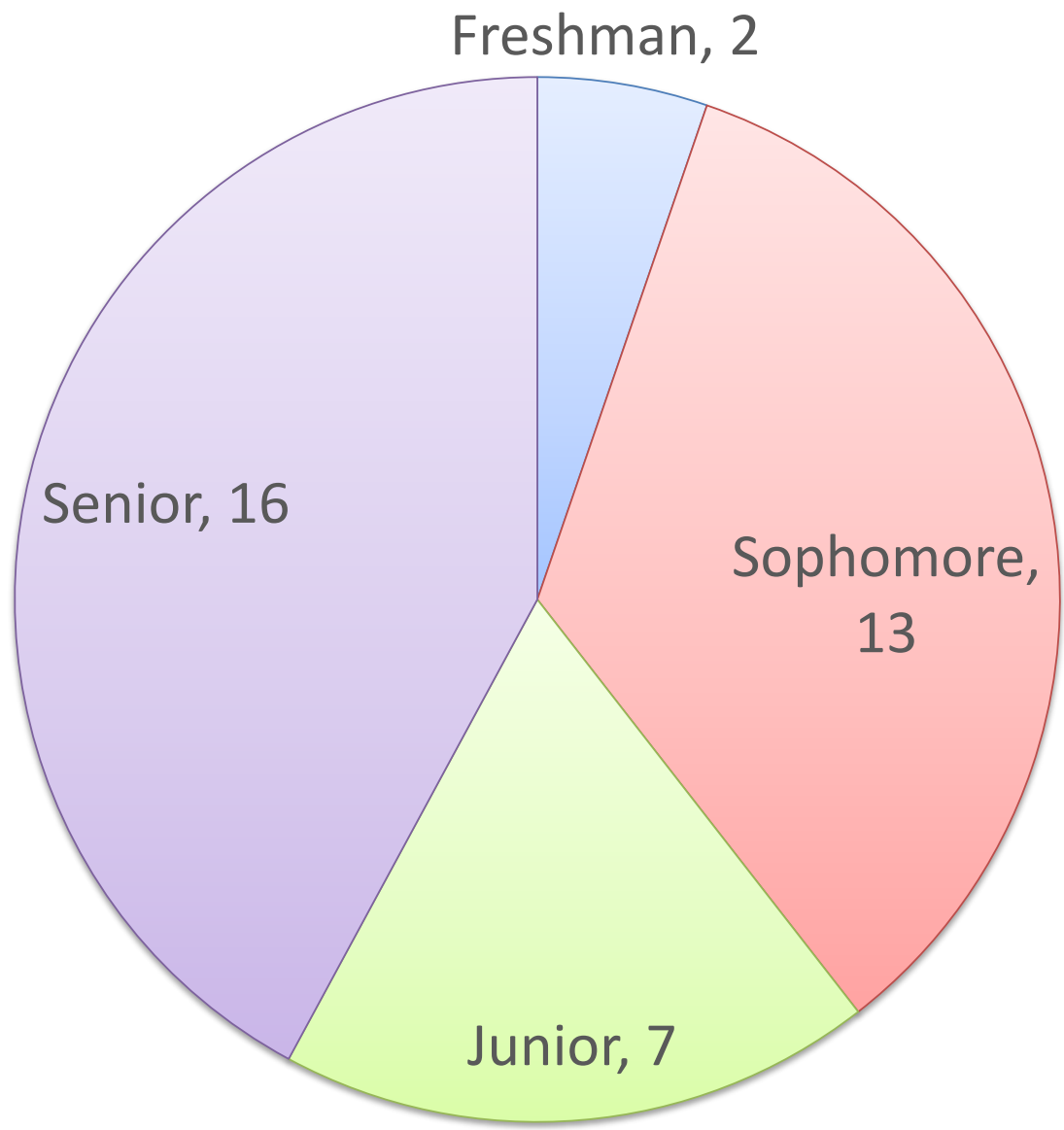
Final 20%

Bonus I: lecture and piazza participation 5%

Bonus II: extra-credit lab questions (points vary)

Is CSO going to be hard?





Time to work hard



We (the course staff) are here to help

Who are we?

Shuai Mu

Lecturer

Conrad Christensen

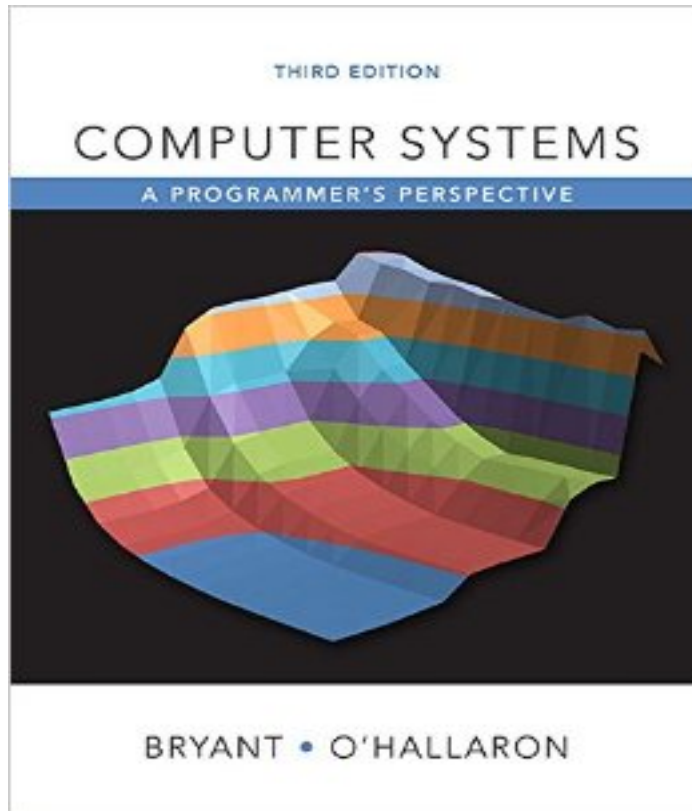
Recitation Leader & Grader

Lamont Nelson

Grader

Before Class

Read the related sections in the text books



“Computer Systems: A Programmer’s Perspective, 3rd 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

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://mpaxos.com/teaching/cso18spring/>

Recitation starts next Wed

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.