

CSC 252/452: Computer Organization

Fall 2024: Lecture 1

Instructor: Yanan Guo

Department of Computer Science
University of Rochester

Action Items:

- **Get CSUG account**
- **Make sure you have VPN setup!!!!**
- **Sign up for Blackboard & Piazza**

Outline: Class Introduction

- Introduction
 - What Are You Supposed to Learn in this Class?
 - Action Items
 - Instructor & TAs
 - Class Organization
 - How to Get Help?
 - How Will You Be Evaluated?
 - Policies

Abstraction is good, but don't forget reality

Abstraction is good, but don't forget reality



Problem

Who scores the highest
on the exam?

Abstraction is good, but don't forget reality



Problem

Who scores the highest
on the exam?

Algorithm

Quicksort

Abstraction is good, but don't forget reality



Problem

Algorithm

Program

Who scores the highest
on the exam?

Quicksort

Human-readable
language (Java, C)

Abstraction is good, but don't forget reality



Problem

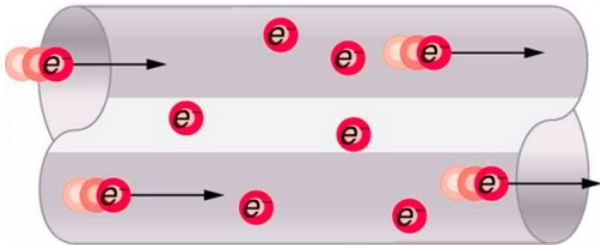
Who scores the highest
on the exam?

Algorithm

Quicksort

Program

Human-readable
language (Java, C)



Circuit

Electrons, Resistors,
Capacitors, etc.

Abstraction is good, but don't forget reality



Problem

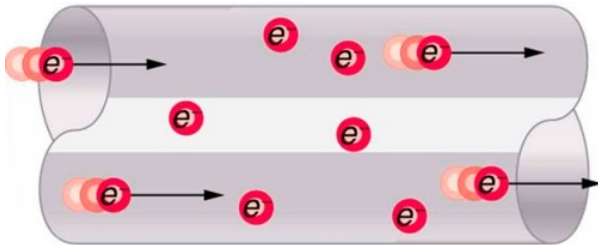
Who scores the highest
on the exam?

Algorithm

Quicksort

Program

Human-readable
language (Java, C)



Circuit

Electrons, Resistors,
Capacitors, etc.

Abstraction is good, but don't forget reality



Problem

Who scores the highest
on the exam?

Algorithm

Quicksort

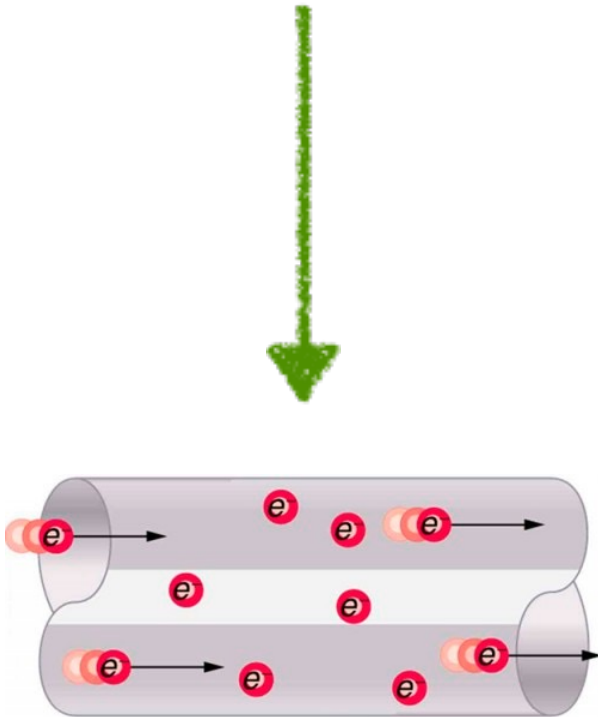
Program

Human-readable
language (Java, C)

CSC 252/452
Computer Systems--
"The Reality"

Circuit

Electrons, Resistors,
Capacitors, etc.



Course Theme:

(Systems) Knowledge is Power!

- **Systems Knowledge**
 - How hardware (processors, memories, disk drives) plus software (operating systems, compilers, libraries) combine to support the execution of application programs
 - How you as a programmer can best use these resources
- **Useful outcomes from taking 252/452**

Course Theme:

(Systems) Knowledge is Power!

- **Systems Knowledge**
 - How hardware (processors, memories, disk drives) plus software (operating systems, compilers, libraries) combine to support the execution of application programs
 - How you as a programmer can best use these resources
- **Useful outcomes from taking 252/452**
 - Prepare for later “systems” classes in CS & ECE
 - Compilers, Operating Systems, Networks, Computer Architecture, etc.

Course Theme:

(Systems) Knowledge is Power!

- **Systems Knowledge**
 - How hardware (processors, memories, disk drives) plus software (operating systems, compilers, libraries) combine to support the execution of application programs
 - How you as a programmer can best use these resources
- **Useful outcomes from taking 252/452**
 - Prepare for later “systems” classes in CS & ECE
 - Compilers, Operating Systems, Networks, Computer Architecture, etc.
 - Become more effective programmers
 - Able to find and eliminate bugs efficiently
 - Able to understand and tune for program performance

Reality #1: Data Representations

- You will learn in computers,
 - 2,147,483,647 → 01111111111111111111111111111111
 - 1 → 00000000000000000000000000000001

Reality #1: Data Representations

- You will learn in computers,
 - $2,147,483,647 \longrightarrow 01111111111111111111111111111111$
 - $1 \longrightarrow 00000000000000000000000000000001$
 - $2,147,483,647 + 1 = \text{-2,147,483,648}$
 $\longrightarrow 10000000000000000000000000000000$

Reality #1: Data Representations

- You will learn in computers,

➤ 2,147,483,647 → 01111111111111111111111111111111

➤ 1 → 00000000000000000000000000000001

➤ 2,147,483,647 + 1 = **-2,147,483,648**

→ 10000000000000000000000000000000

➤ -6.5 → 11000000110100000000000000000000

Reality #1: Data Representations

- You will learn in computers,
 - $2,147,483,647 \longrightarrow 01111111111111111111111111111111$
 - $1 \longrightarrow 00000000000000000000000000000001$
 - $2,147,483,647 + 1 = -2,147,483,648$
 $\longrightarrow 10000000000000000000000000000000$
 - $-6.5 \longrightarrow 11000000110100000000000000000000$
 - $(x+y)+z = (y+z)+x$? Yes and No

Reality #2: Assembly and Processors



Problem

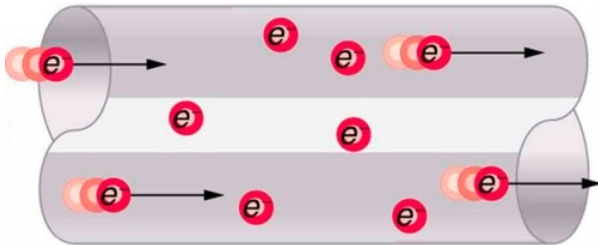
Who scores the highest
on the exam?

Algorithm

Quicksort

Program

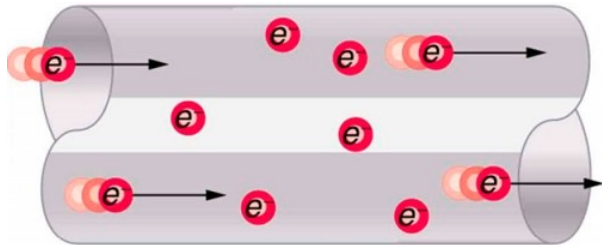
Human-readable
language (Java, C)



Circuit

Electrons, Resistors,
Capacitors, etc.

Reality #2: Assembly and Processors



Problem

Algorithm

Program

Instruction Set
Architecture

Microarchitecture

Circuit

Who scores the highest
on the exam?

Quicksort

Human-readable
language (Java, C)

Machine Language

Hardware Design

Electrons, Resistors,
Capacitors, etc.

Instruction Set Architecture

- The programmer's view of the computer is called the “instruction set architecture” (*ISA*)
 - E.g., x86, RICS-V, ARM...

Instruction Set Architecture

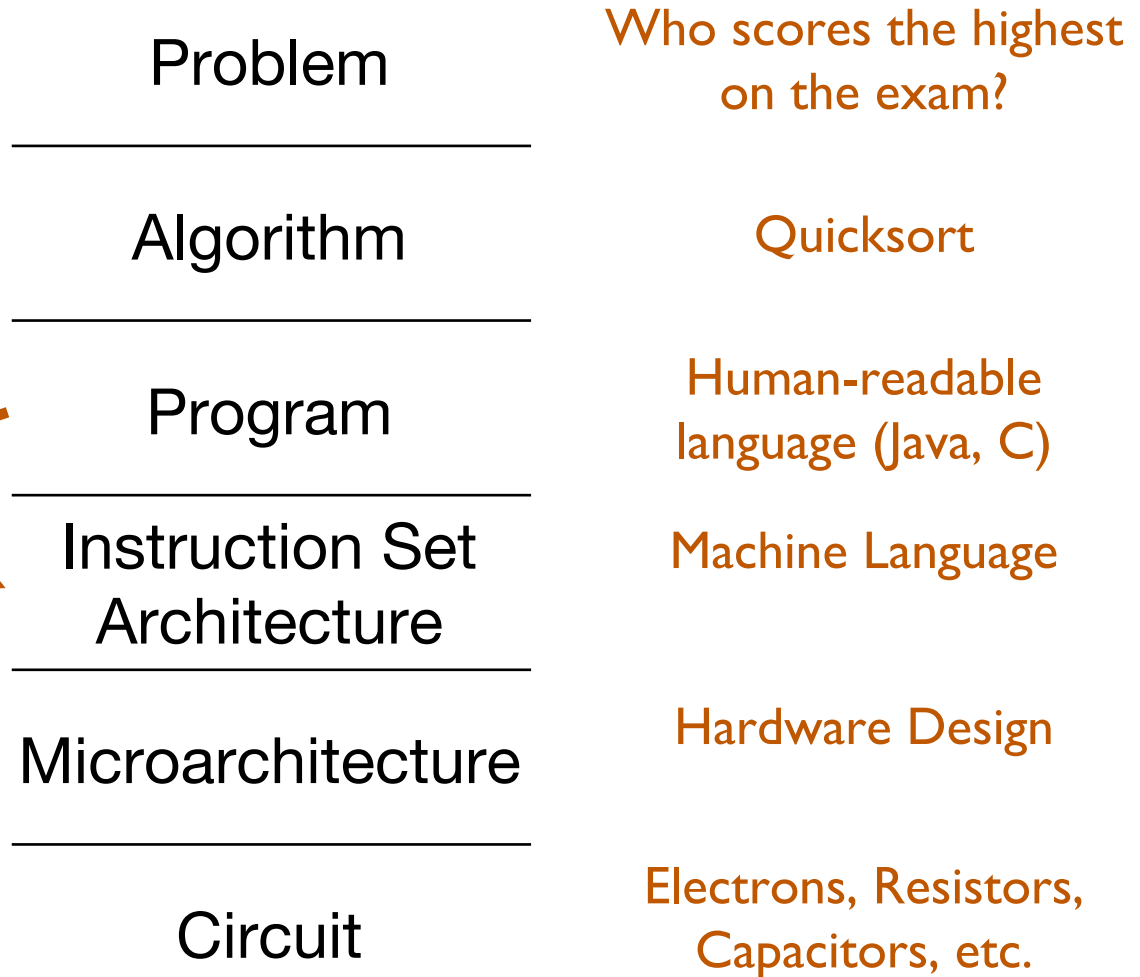
- The programmer's view of the computer is called the “instruction set architecture” (*ISA*)
 - E.g., x86, RISC-V, ARM...
- For programmer: no need to care how the instructions are implemented as long as they are somehow implemented

Two Fundamental Aspects of Computer Systems

Problem	Who scores the highest on the exam?
Algorithm	Quicksort
Program	Human-readable language (Java, C)
Instruction Set Architecture	Machine Language
Microarchitecture	Hardware Design
Circuit	Electrons, Resistors, Capacitors, etc.

Two Fundamental Aspects of Computer Systems

- How is a human-readable program translated to a representation that computers can understand?



The “Translation” Process, a.k.a., **Compilation**

C Program

```
void add() {  
    int a = 1;  
    int b = 2;  
    int c = a + b;  
}
```

**Pre-processor
Compiler**



Assembly program

```
movl    $1, -4(%rbp)  
movl    $2, -8(%rbp)  
movl    -4(%rbp), %eax  
addl    -8(%rbp), %eax
```

The “Translation” Process, a.k.a., **Compilation**

C Program

```
void add() {  
    int a = 1;  
    int b = 2;  
    int c = a + b;  
}
```

**Pre-processor
Compiler**



Assembly program

```
movl    $1, -4(%rbp)  
movl    $2, -8(%rbp)  
movl    -4(%rbp), %eax  
addl    -8(%rbp), %eax
```


The “Translation” Process, a.k.a., **Compilation**

Assembly program

```
movl    $1, -4(%rbp)
movl    $2, -8(%rbp)
movl    -4(%rbp), %eax
addl    -8(%rbp), %eax
```



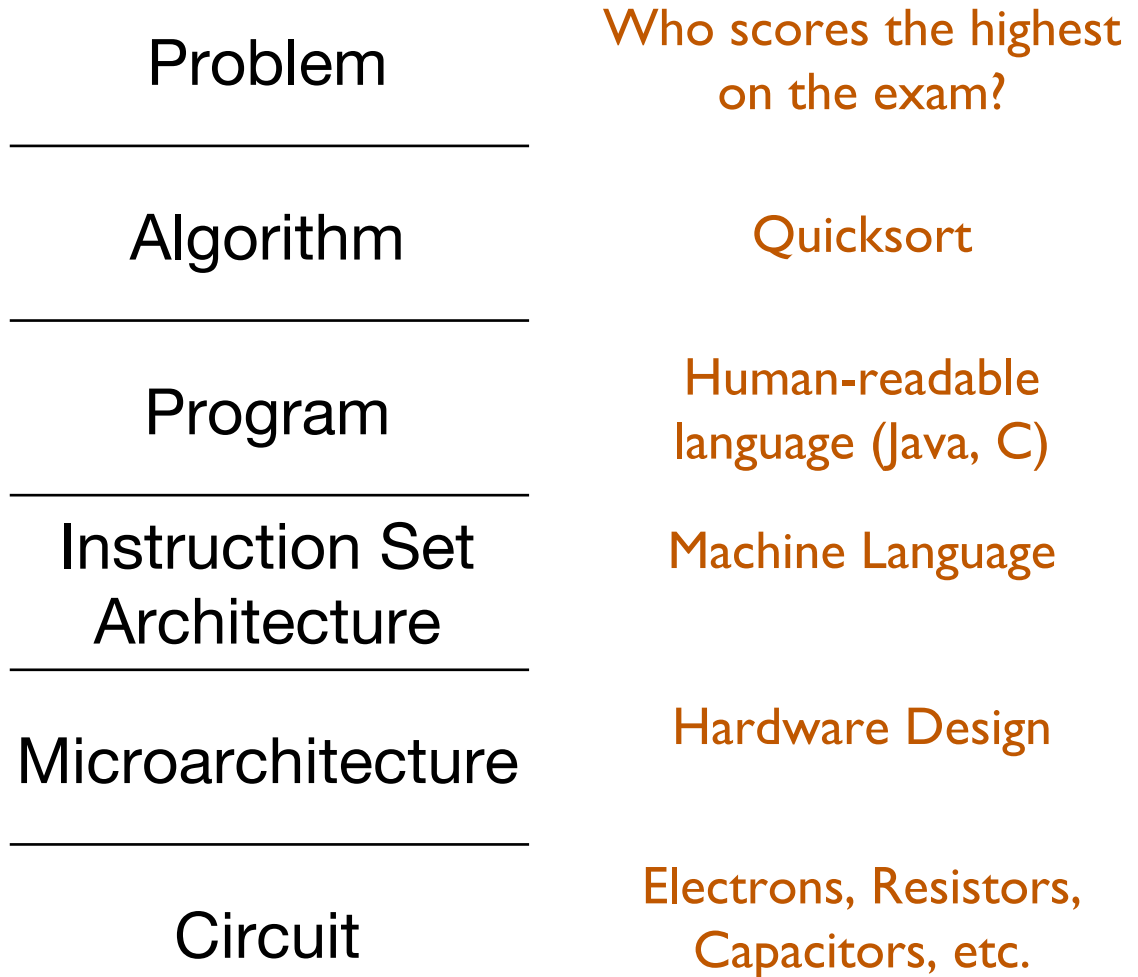
Executable Binary

```
00011001 ...
01101010 ...
11010101 ...
01110001 ...
```

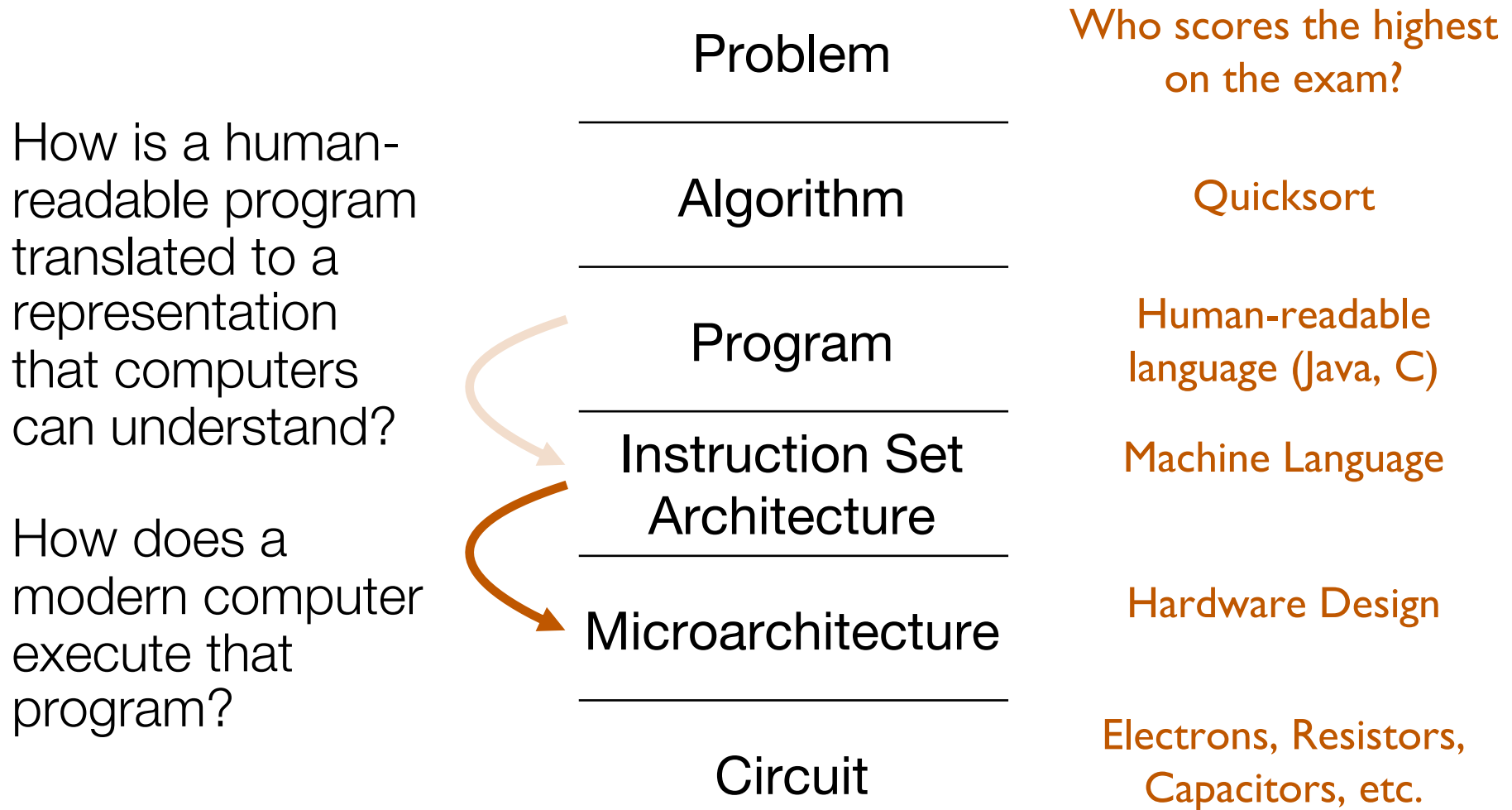
- It translates a text file to an executable binary file (a.k.a., executable) consisting of a sequence of **instructions**
- Why binary? Computers understand only 0s and 1s
 - The subject of next lecture

Back to Layers of Transformation...

How is a human-readable program translated to a representation that computers can understand?



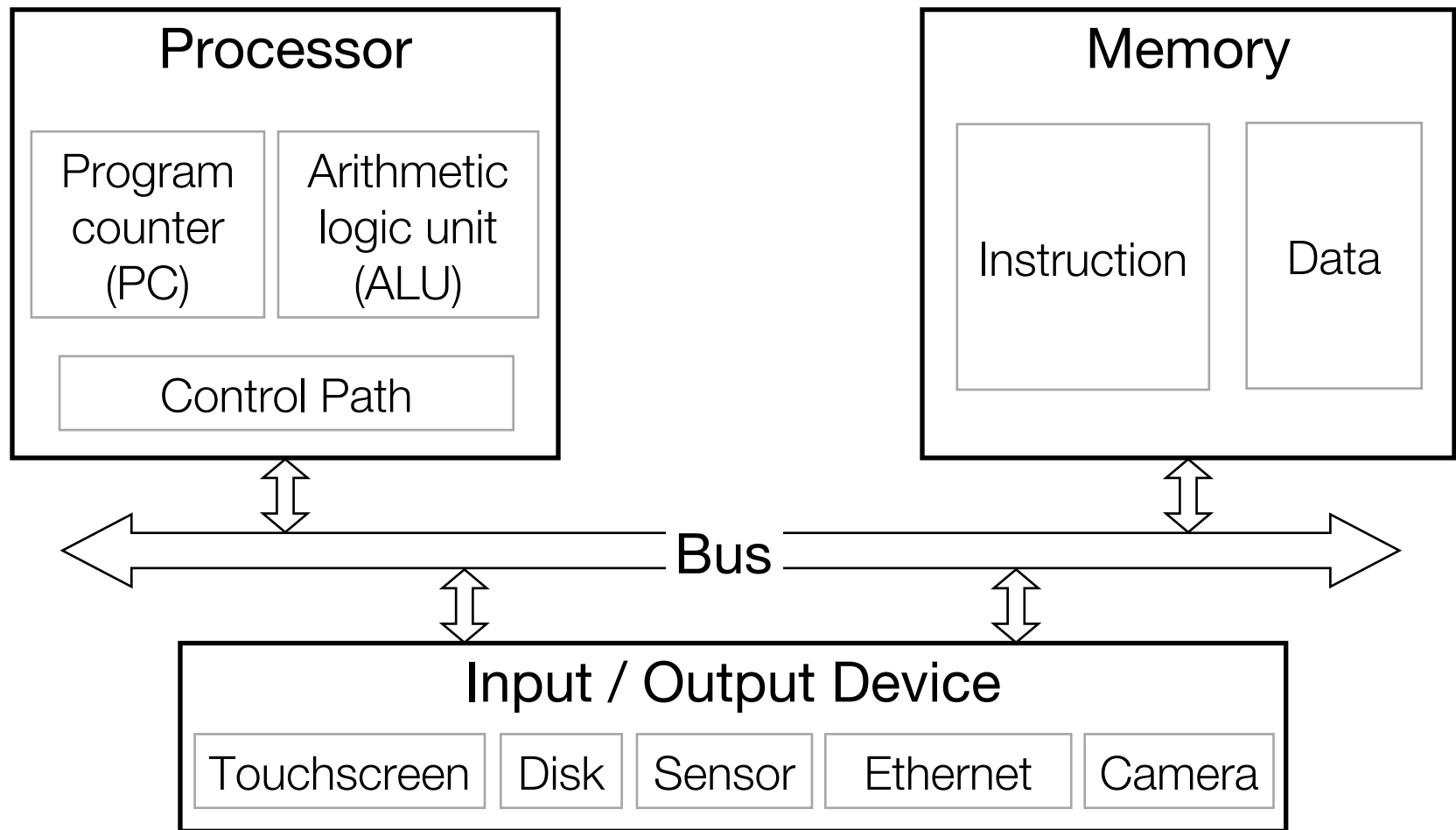
Back to Layers of Transformation...



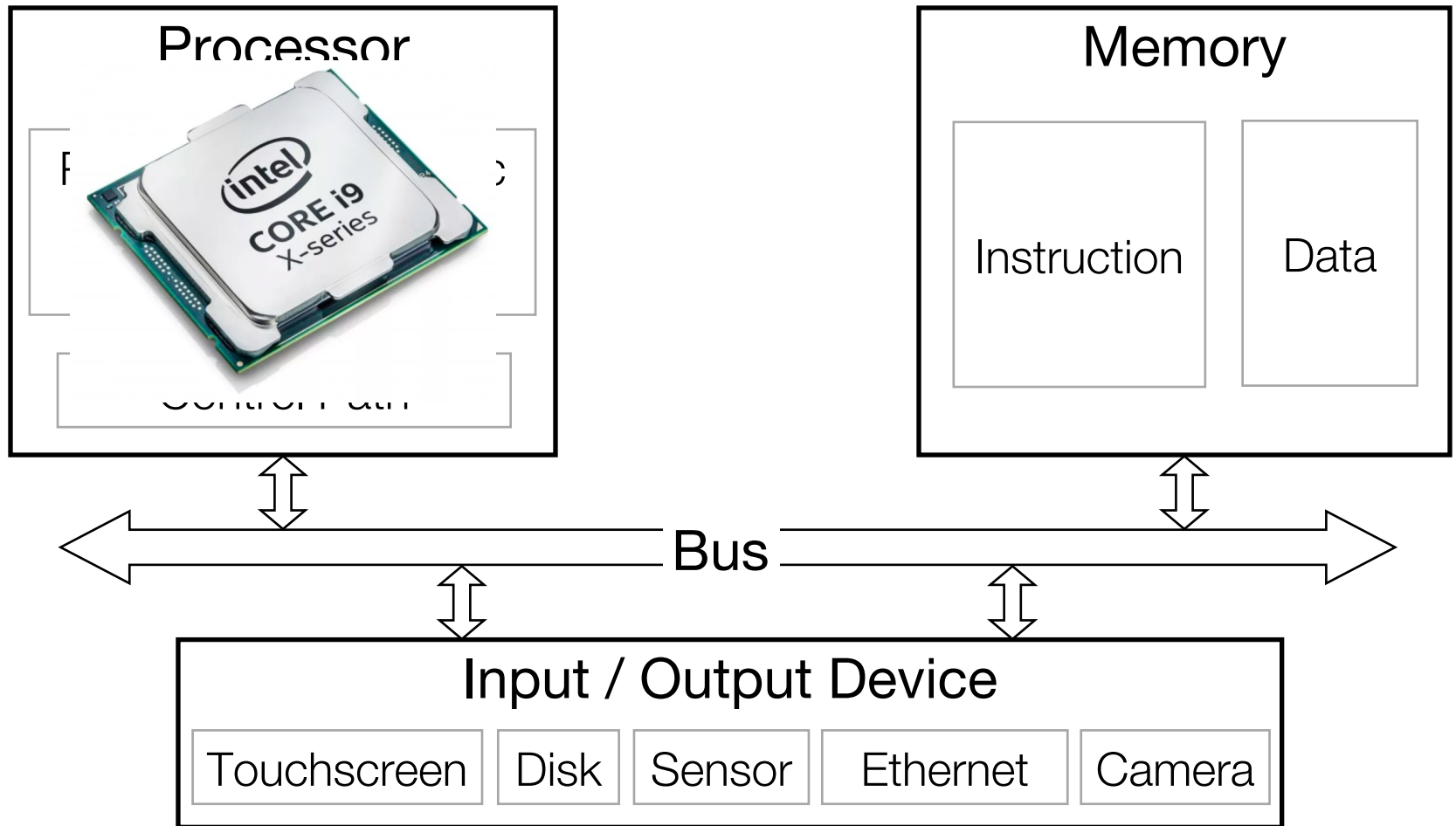
Instruction Set Architecture

- The programmer's view of the computer is called the “instruction set architecture” (*ISA*)
 - E.g., x86, RISC-V, ARM...
- For programmer: no need to care how the instructions are implemented as long as they are somehow implemented
- Implementation of an ISA is called *microarchitecture*
- ISAs *abstract* away details of microarchitecture

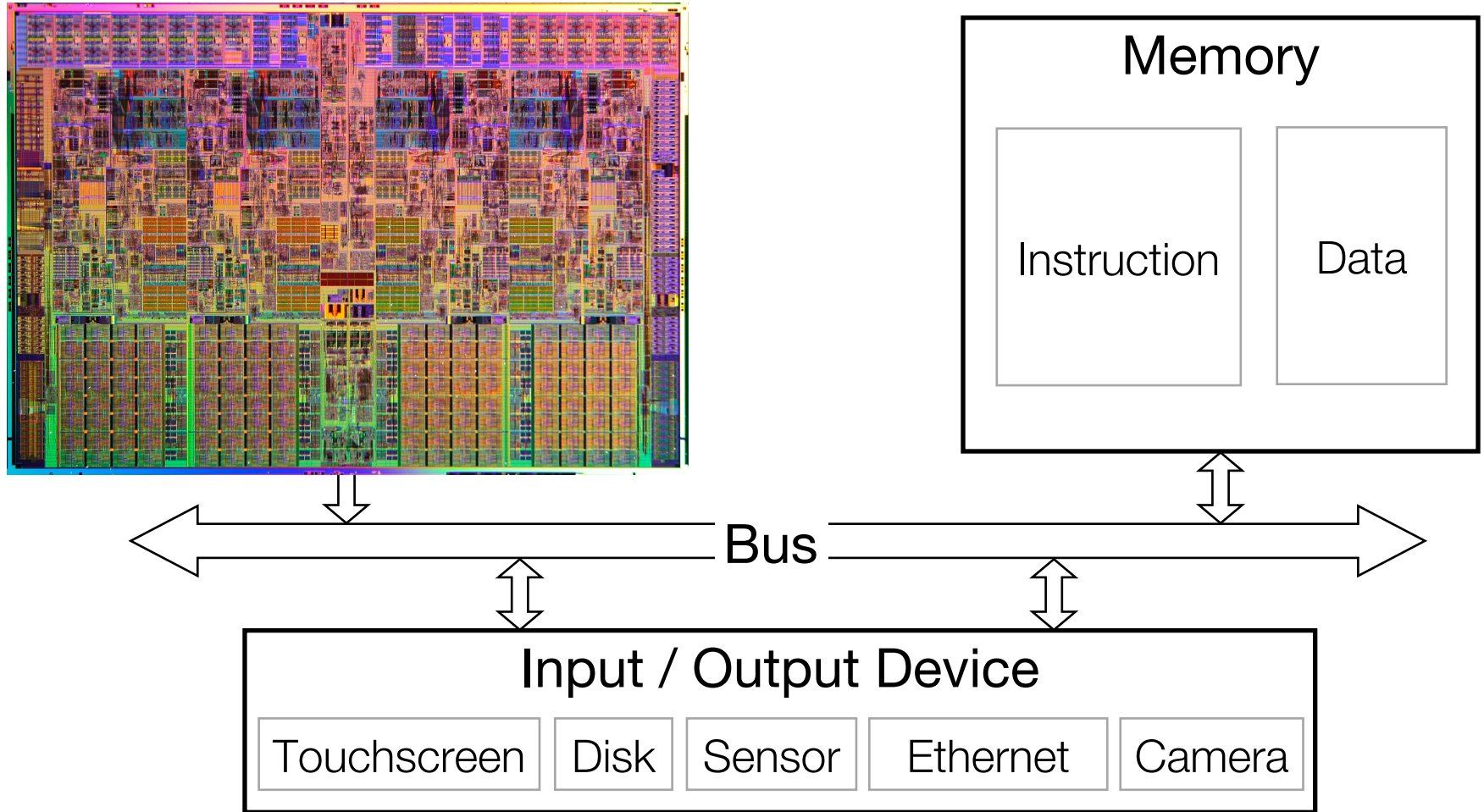
High-level Organization of Computer Hardware a.k.a., The Von Neumann Model



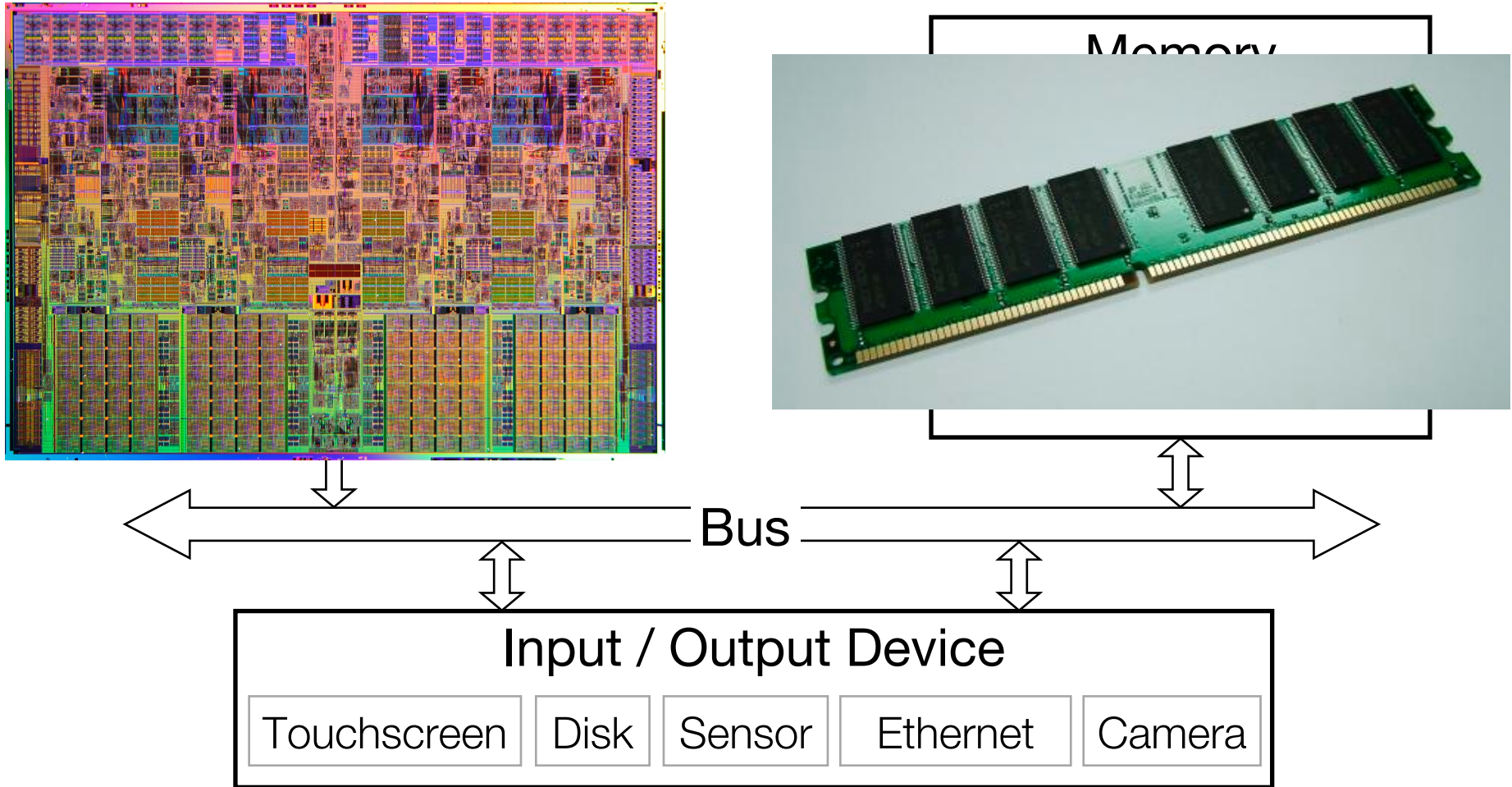
High-level Organization of Computer Hardware a.k.a., The Von Neumann Model



High-level Organization of Computer Hardware a.k.a., The Von Neumann Model



High-level Organization of Computer Hardware a.k.a., The Von Neumann Model



The Single Most Important Idea of Computers

- Executables (i.e., instructions) are stored in “memory”
- Processors read instructions from memory and execute instructions one after another

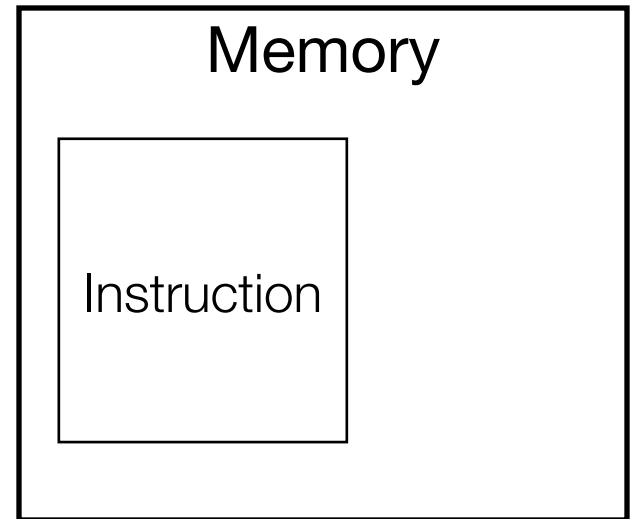
Assembly program: add.s

```
movl    $1, -4(%rbp)
movl    $2, -8(%rbp)
movl    -4(%rbp), %eax
addl    -8(%rbp), %eax
```

The Single Most Important Idea of Computers

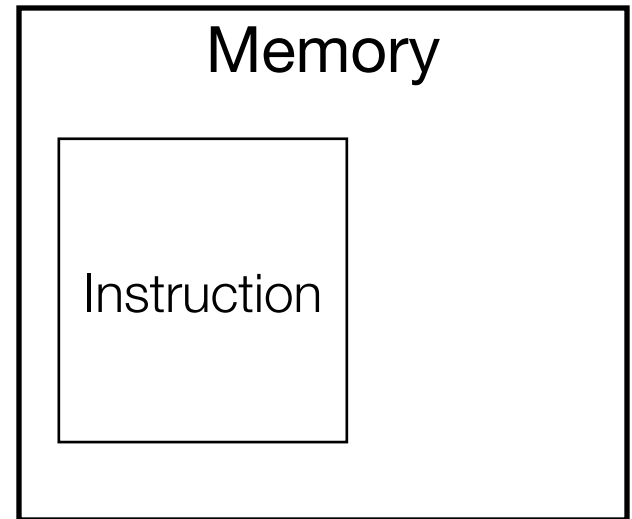
Assembly program: add.s

```
movl    $1, -4(%rbp)
movl    $2, -8(%rbp)
movl    -4(%rbp), %eax
addl    -8(%rbp), %eax
```

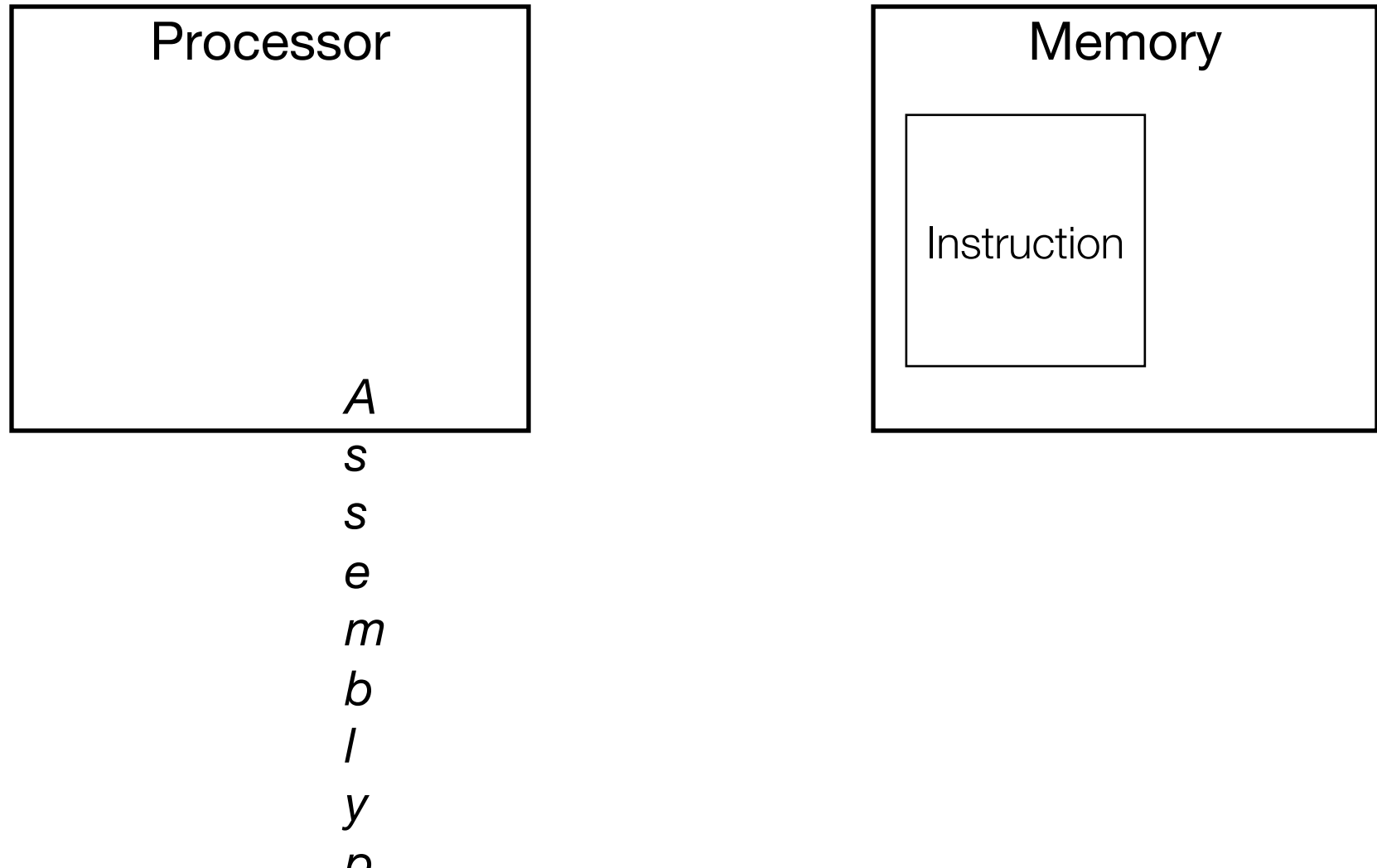


The Single Most Important Idea of Computers

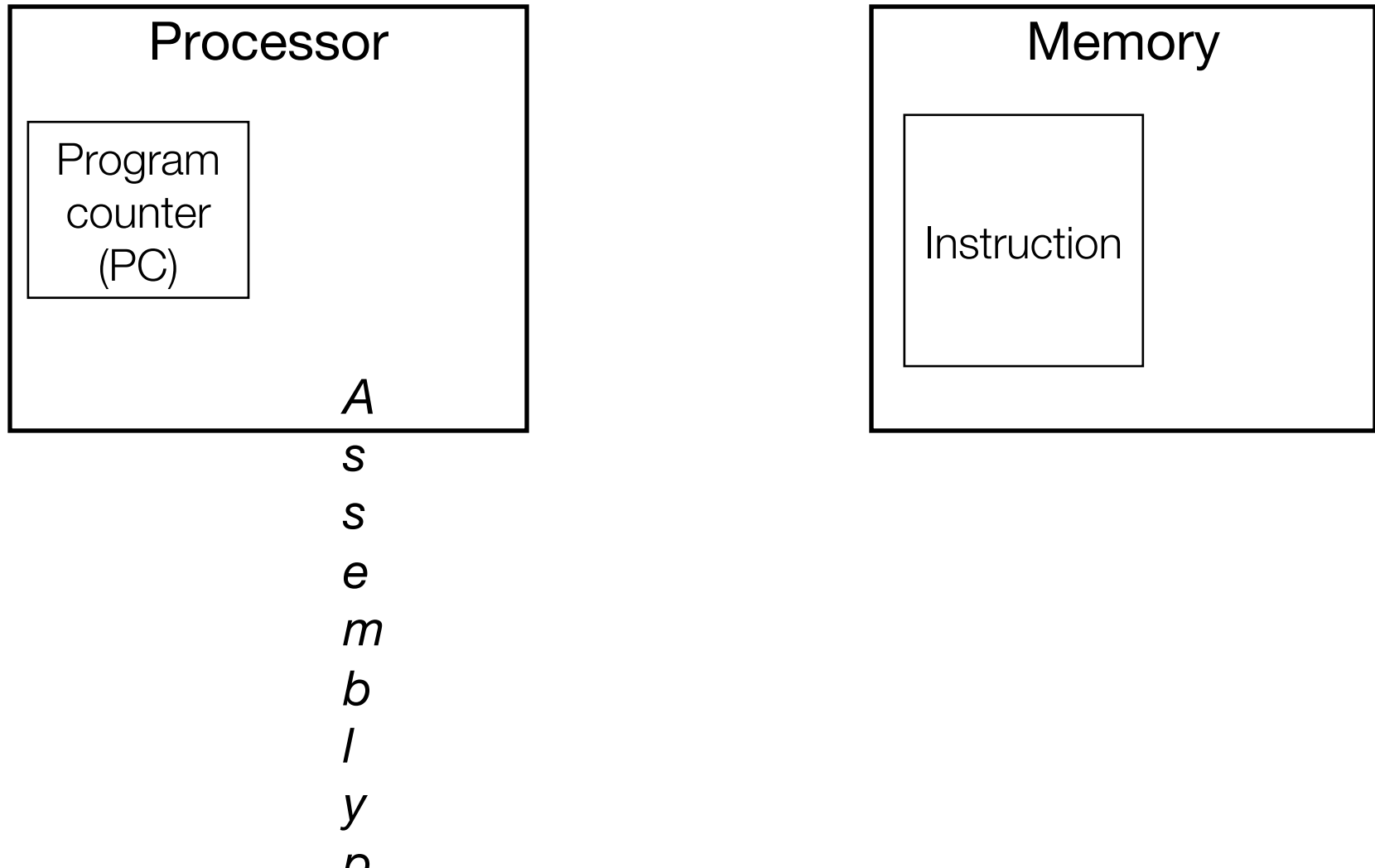
*A
s
s
e
m
b
l
y*



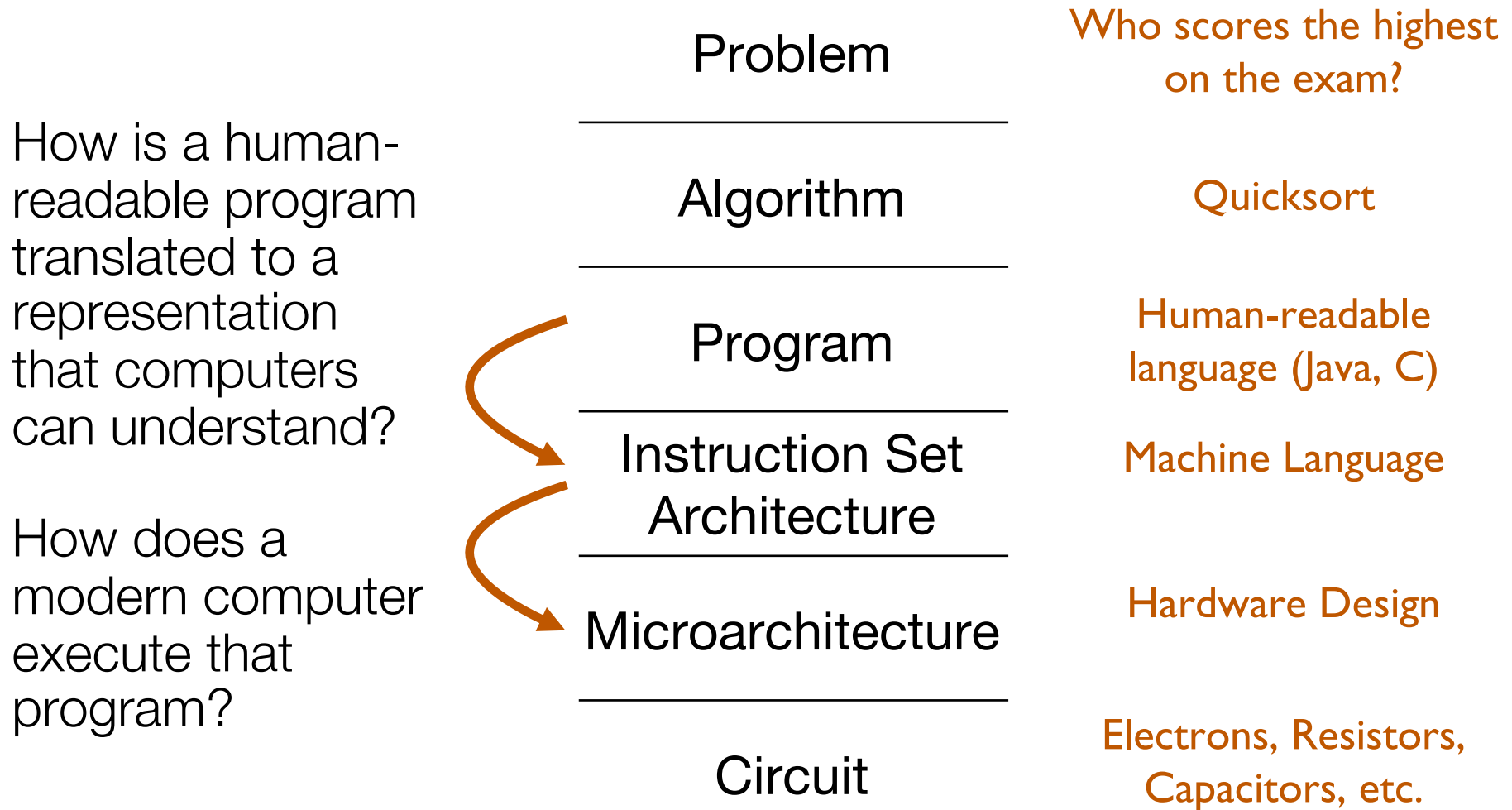
The Single Most Important Idea of Computers



The Single Most Important Idea of Computers



Back to Layers of Transformation...



Reality #3: There's more to performance than algorithmic complexity

- **Even exact op count does not predict performance**
 - Easily see 10:1 performance range depending on how code written
 - Must optimize at multiple levels: algorithm, data representations, procedures, and loops
- **Must understand system to optimize performance**
 - How programs compiled and executed
 - How to measure program performance and identify bottlenecks
 - How to improve performance without destroying code modularity and generality

Memory System Performance Example

```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (i = 0; i < 2048; i++)
        for (j = 0; j < 2048; j++)
            dst[i][j] = src[i][j];
}
```


Memory System Performance Example


```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (i = 0; i < 2048; i++)
        for (j = 0; j < 2048; j++)
            dst[i][j] = src[i][j];
}
```

```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (j = 0; j < 2048; j++)
        for (i = 0; i < 2048; i++)
            dst[i][j] = src[i][j];
}
```

Memory System Performance Example

```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (i = 0; i < 2048; i++)
        for (j = 0; j < 2048; j++)
            dst[i][j] = src[i][j];
}
```

```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (j = 0; j < 2048; j++)
        for (i = 0; i < 2048; i++)
            dst[i][j] = src[i][j];
}
```



Memory System Performance Example

```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (i = 0; i < 2048; i++)
        for (j = 0; j < 2048; j++)
            dst[i][j] = src[i][j];
}
```

4.3ms

```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (j = 0; j < 2048; j++)
        for (i = 0; i < 2048; i++)
            dst[i][j] = src[i][j];
}
```

81.8ms

2.0 GHz Intel Core i7 Haswell

Memory System Performance Example

```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (i = 0; i < 2048; i++)
        for (j = 0; j < 2048; j++)
            dst[i][j] = src[i][j];
}
```

4.3ms

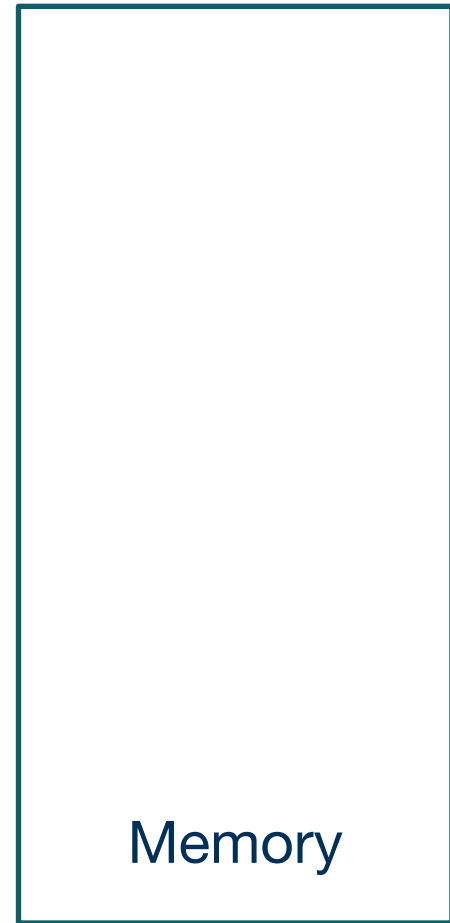
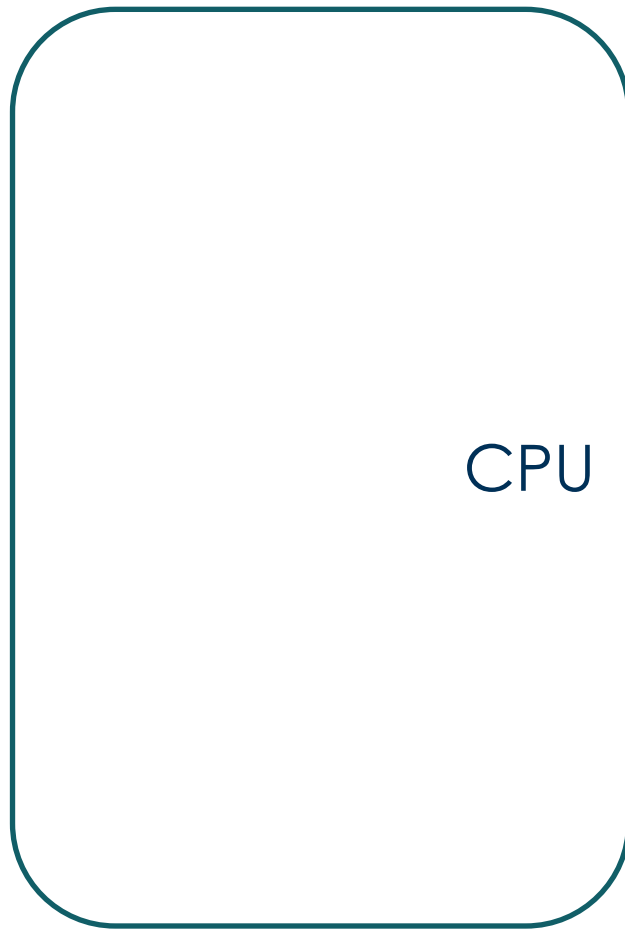
```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (j = 0; j < 2048; j++)
        for (i = 0; i < 2048; i++)
            dst[i][j] = src[i][j];
}
```

81.8ms

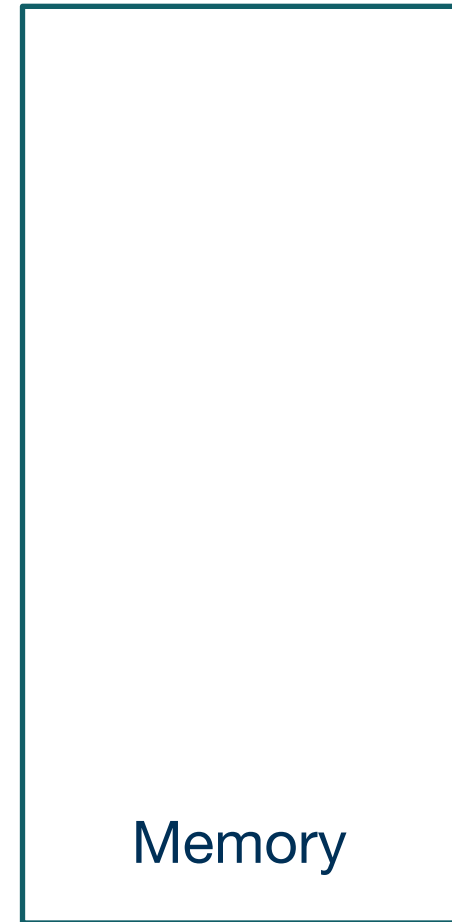
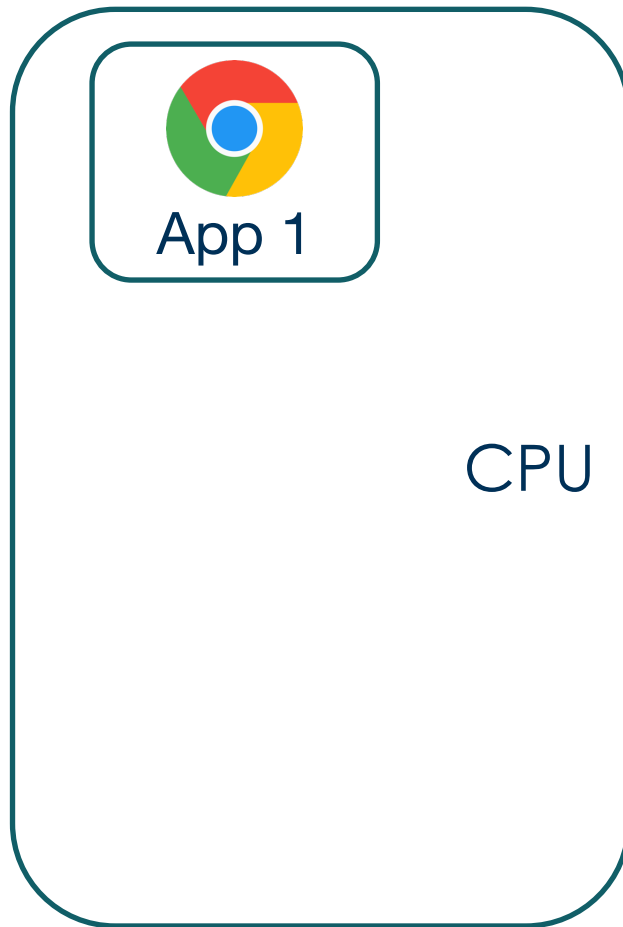
2.0 GHz Intel Core i7 Haswell

- Microarchitecture: Hierarchical memory organization
- Performance depends on access patterns
 - Including how step through multi-dimensional array

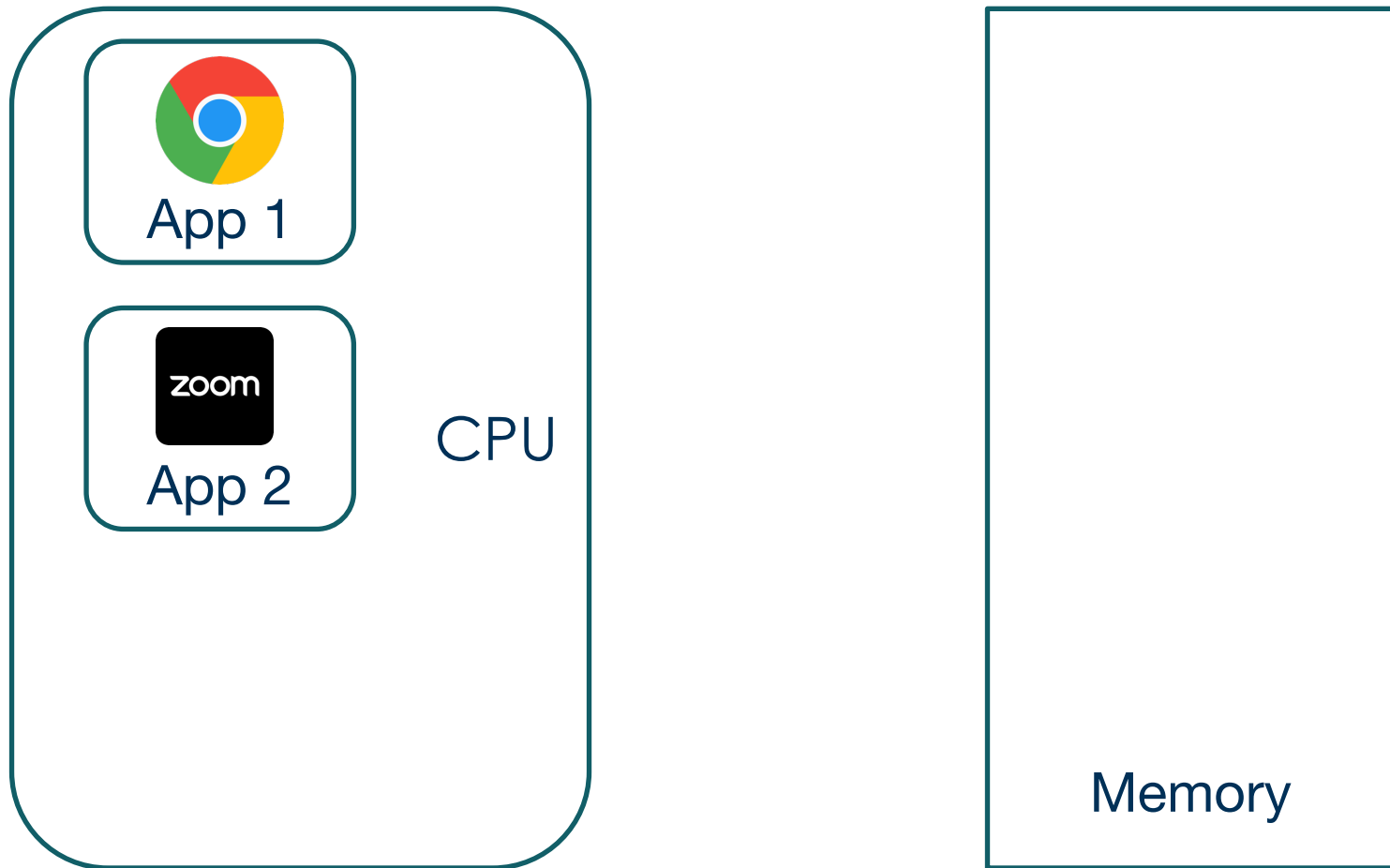
Reality #4: Processes and Memory Management



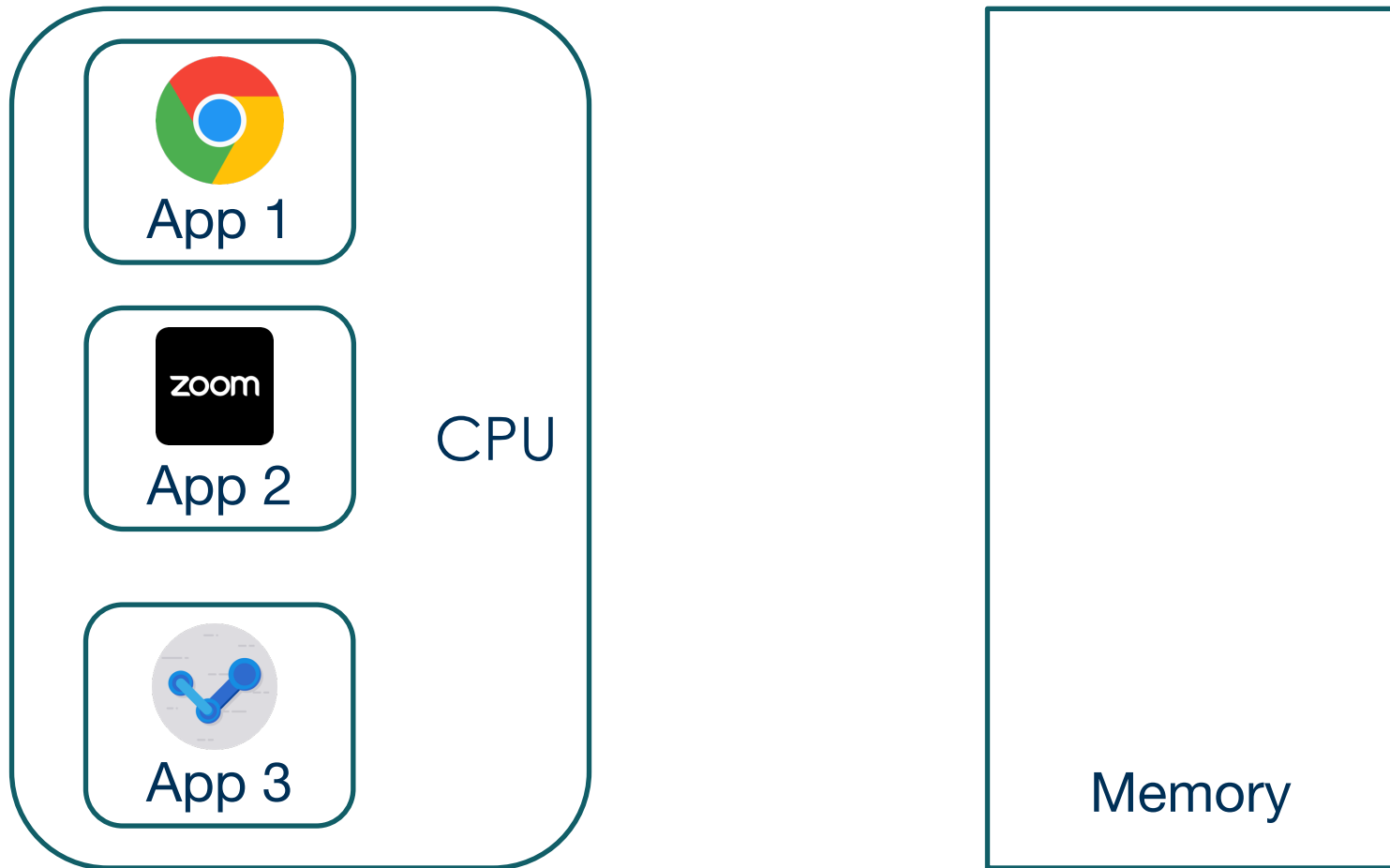
Reality #4: Processes and Memory Management



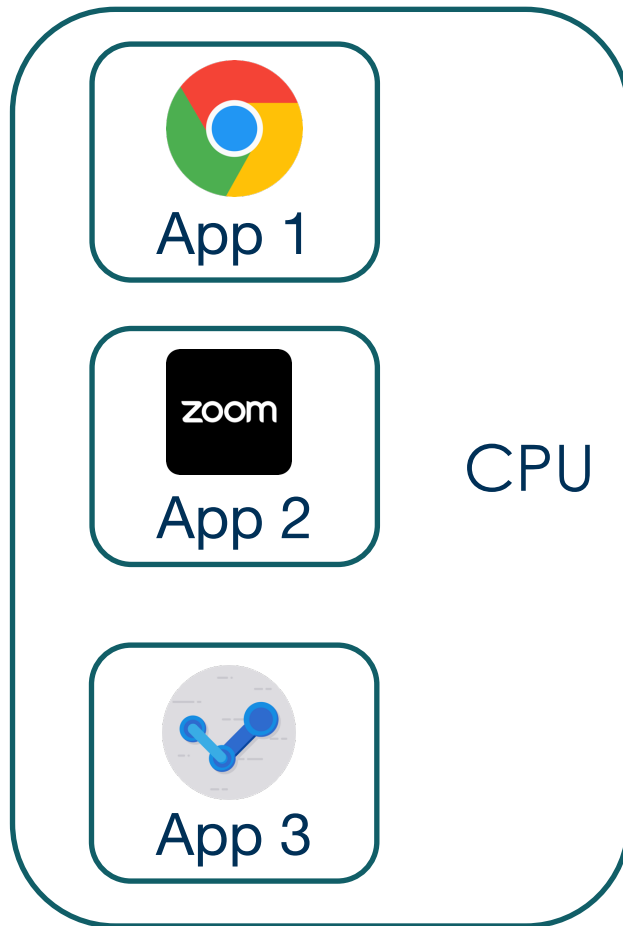
Reality #4: Processes and Memory Management



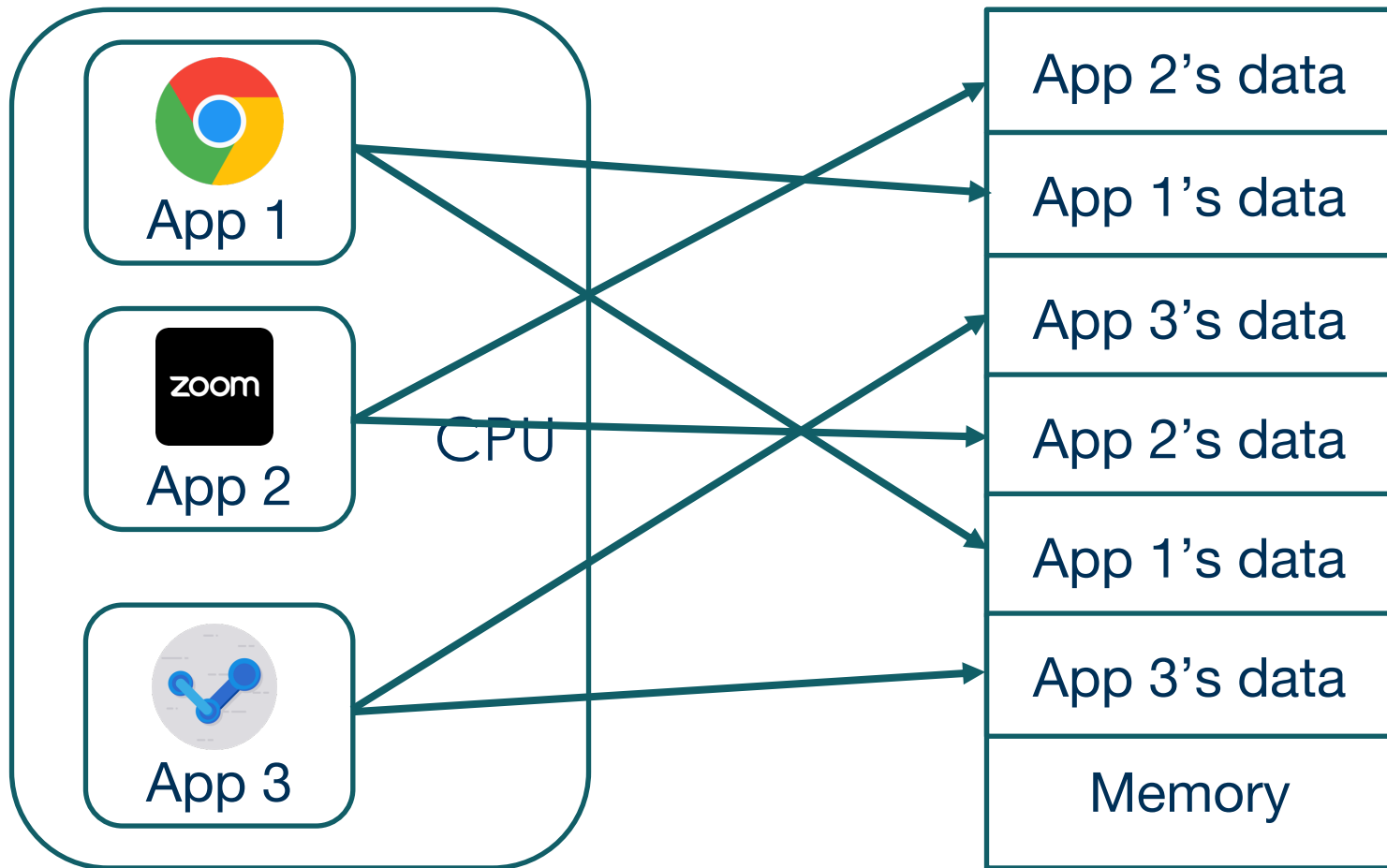
Reality #4: Processes and Memory Management



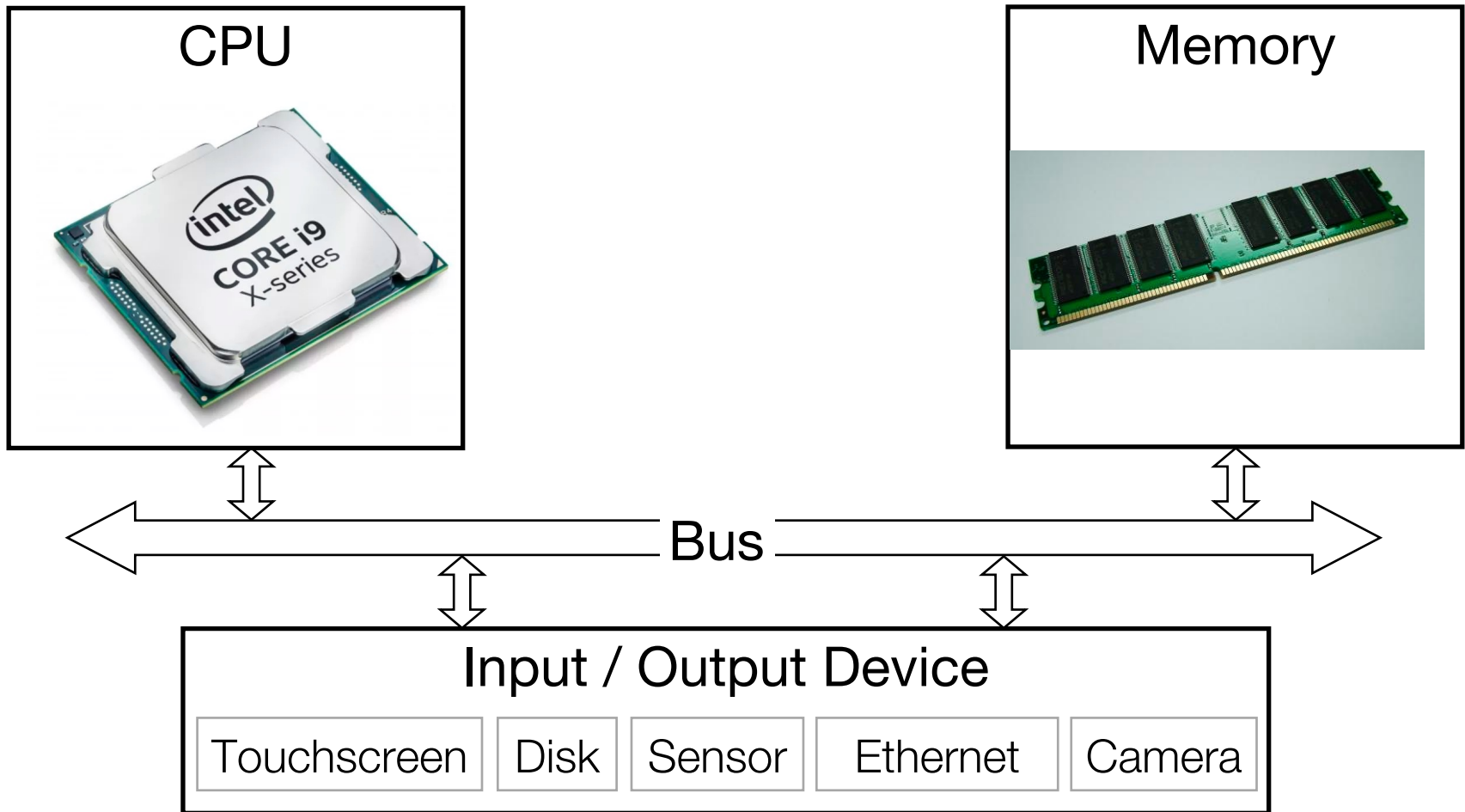
Reality #4: Processes and Memory Management



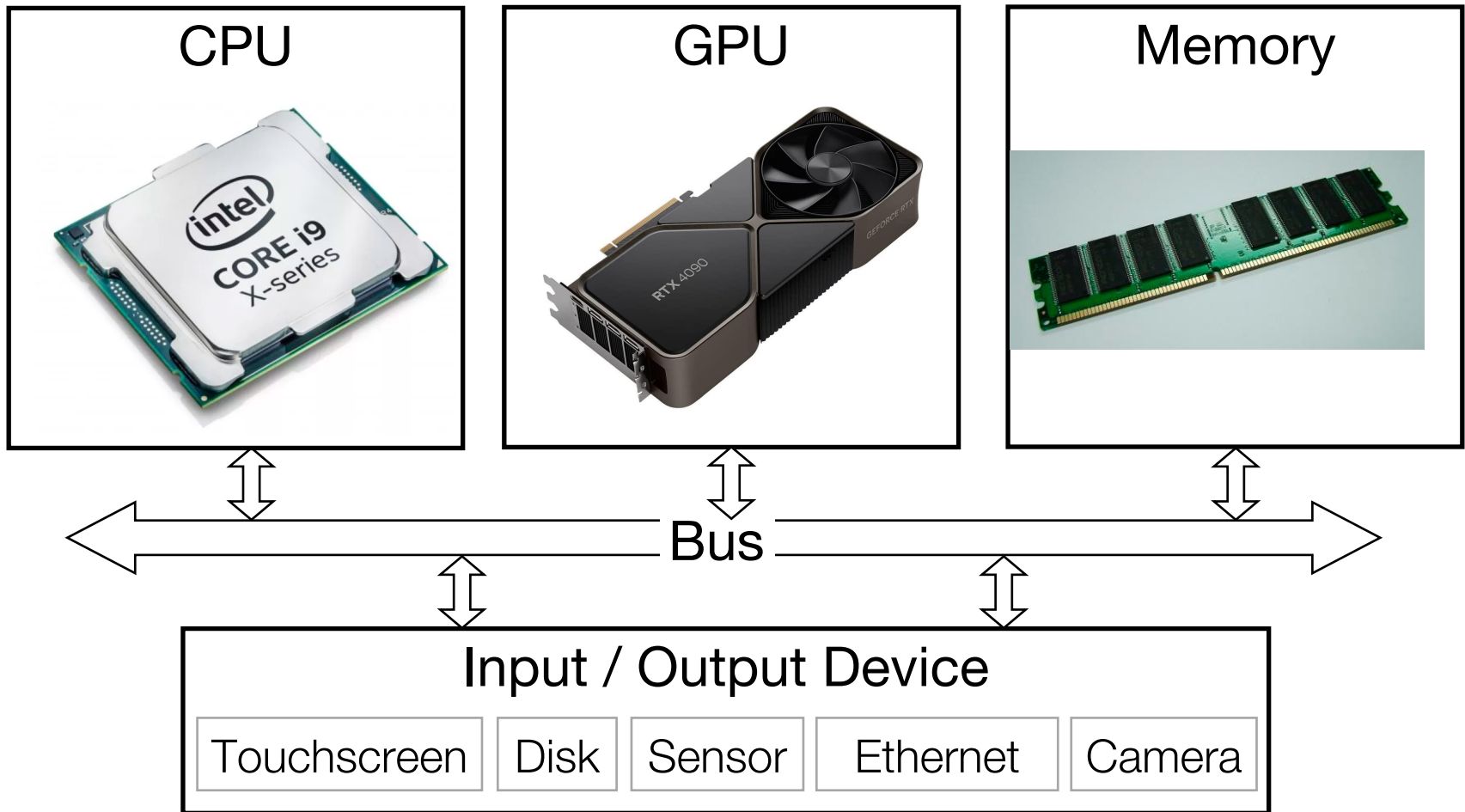
Reality #4: Processes and Memory Management



Reality #5: GPUs



Reality #5: GPUs

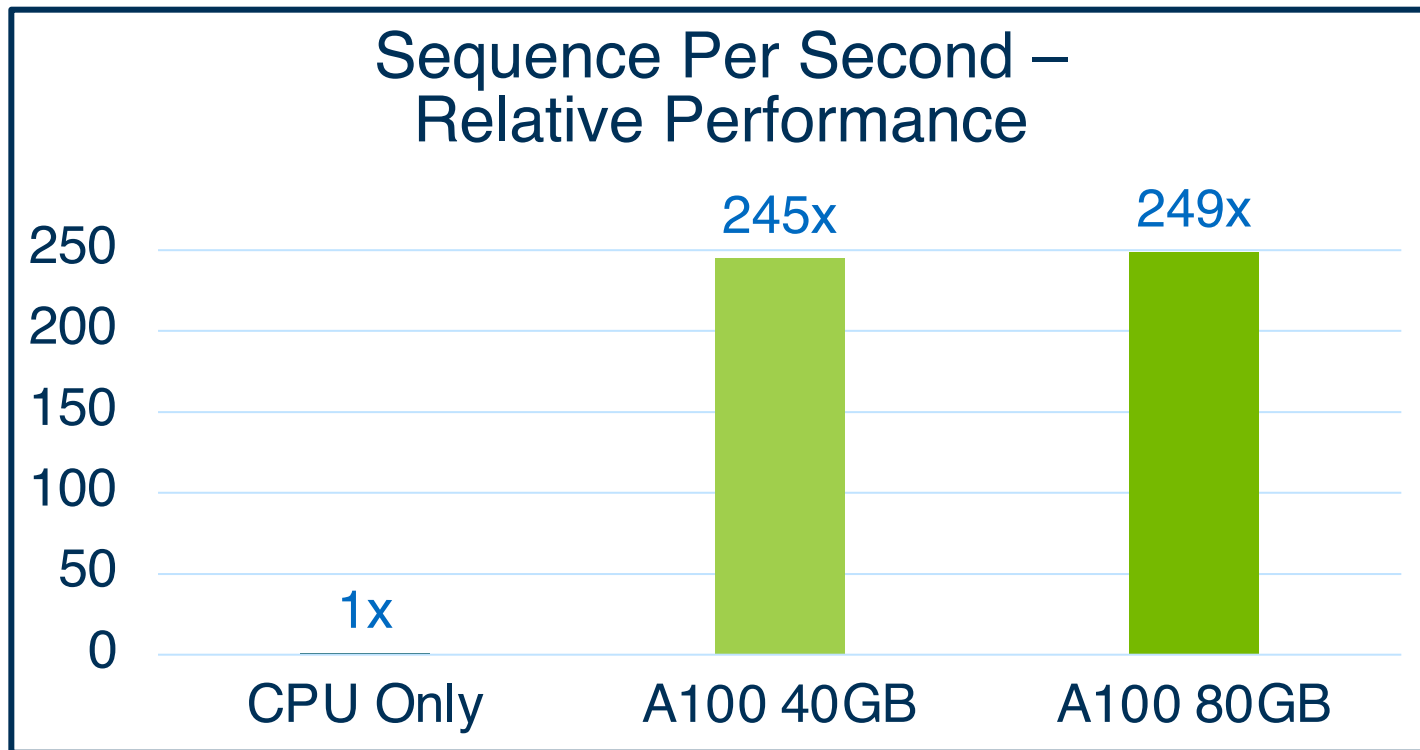


Reality #5: GPUs

- “Graphics” Processing Units are used as AI accelerators

Reality #5: GPUs

- “Graphics” Processing Units are used as AI accelerators



<https://www.nvidia.com/en-us/data-center/a100/>

Reality #5: GPUs

- “Graphics” Processing Units are used as AI accelerators

Reality #5: GPUs

- “Graphics” Processing Units are used as AI accelerators
 - How do GPUs work?
 - How are they different than CPUs?
 - Hardware design
 - Software design
 - How to program GPUs to achieve best performance?

Reality #6: Security

- Security was considered a software level problem.



Encryption



Permission control

...

- But all data are stored in hardware
 - What if hardware has security flaws, which it does...
 - These flaws can be exploited without physical access!
 - **This is what I work on**

Outline: Class Introduction

- Introduction
 - What Are You Supposed to Learn in this Class?
 - **Action Items**
 - Instructor & TAs
 - Class Organization
 - How to Get Help?
 - How Will you Be Evaluated?
 - Policies

Action Items

- Get a CSUG account.
 - cycle1.csug.rochester.edu (or cycle2, cycle3)
 - If you don't already have one, go to this link:
<https://accounts.csug.rochester.edu/>
 - **YOU WILL NEED VPN** to access these machines if you are not using campus WiFi!! Follow the instructions (<https://tech.rochester.edu/remote-access-vpn-tutorials/>) to set up the university VPN.
 - TAs will help with VPN setup too.
- Sign up for Blackboard (<https://learn.rochester.edu/>)
- Sign up for Piazza (<https://piazza.com/rochester/fall2024/csc252452>)

Outline: Class Introduction

- Introduction
 - What Are You Supposed to Learn in this Class?
 - Action Items
 - **Instructor & TAs**
 - Class Organization
 - How to Get Help?
 - How Will You Be Evaluated?
 - Policies

Who Are We?

- **Myself: Yanan Guo**
 - WH 3403, yanan.guo@rochester.edu
 - Office hours Thursday 11am - 12pm (zoom link on course website)
 - Got a PhD degree
 - Got some industry experience
 - Interested in computer systems security

Who Are We?

- **Myself: Yanan Guo**
 - WH 3403, yanan.guo@rochester.edu
 - Office hours Thursday 11am - 12pm (zoom link on course website)
 - Got a PhD degree
 - Got some industry experience
 - Interested in computer systems security
- ***TAs: 2 Grads + 6 UGs***
 - Office hours and contacts on course website
 - They share the same zoom link for office hours
 - Did very well themselves in this course before
 - Really care about you learning the material and succeeding
- **Coming to office hours does NOT mean you are weak!**

Outline: Class Introduction

- Introduction
 - What Are You Supposed to Learn in this Class?
 - Action Items
 - Instructor & TAs
 - **Class Organization**
 - How to Get Help?
 - How Will You Be Evaluated?
 - Policies

Class Organization -- Where to Find Stuff

- Course Website: <http://cs.rochester.edu/courses/252/fall2024/>
 - General info, syllabus, schedule, contacts & office hours
 - Programming assignments details
 - Slides
 - Practice problems, past exams
- Blackboard
 - Announcements
 - Grades
- Piazza: <https://piazza.com/rochester/fall2024/csc252452>
- CSUG machines for programming assignments submissions

Outline: Class Introduction

- Introduction
 - What Are You Supposed to Learn in this Class?
 - Action Items
 - Instructor & TAs
 - Class Organization
 - **How to Get Help?**
 - How Will You Be Evaluated?
 - Policies

Getting Help

- For technical (lectures, assignments, exam) or logistics (accounts) questions, post a question on Piazza.

Getting Help

- For technical (lectures, assignments, exam) or logistics (accounts) questions, post a question on Piazza.
 - Option 1: question visible to everyone; you can choose to remain anonymous to other students (**but not to the staff**).

Getting Help

- For technical (lectures, assignments, exam) or logistics (accounts) questions, post a question on Piazza.
 - Option 1: question visible to everyone; you can choose to remain anonymous to other students (**but not to the staff**).
 - Option 2: question only visible to you and the staff.
 - Staff members will also put posts about common questions.

Getting Help

- For technical (lectures, assignments, exam) or logistics (accounts) questions, post a question on Piazza.
 - Option 1: question visible to everyone; you can choose to remain anonymous to other students (**but not to the staff**).
 - Option 2: question only visible to you and the staff.
 - Staff members will also put posts about common questions.

Getting Help

- For technical (lectures, assignments, exam) or logistics (accounts) questions, post a question on Piazza.
 - Option 1: question visible to everyone; you can choose to remain anonymous to other students **(but not to the staff)**.
 - Option 2: question only visible to you and the staff.
 - Staff members will also put posts about common questions.
 - **Be sure to check the posted questions before contacting a staff member!**

Getting Help

- For technical (lectures, assignments, exam) or logistics (accounts) questions, post a question on Piazza.
 - Option 1: question visible to everyone; you can choose to remain anonymous to other students **(but not to the staff)**.
 - Option 2: question only visible to you and the staff.
 - Staff members will also put posts about common questions.
 - **Be sure to check the posted questions before contacting a staff member!**
 - Please answer questions posted by other students!

Getting Help

- For technical (lectures, assignments, exam) or logistics (accounts) questions, post a question on Piazza.
 - Option 1: question visible to everyone; you can choose to remain anonymous to other students **(but not to the staff)**.
 - Option 2: question only visible to you and the staff.
 - Staff members will also put posts about common questions.
 - **Be sure to check the posted questions before contacting a staff member!**
 - Please answer questions posted by other students!
- Going to office hours.
- Schedule meeting via email.

Textbook

- Required textbook
 - Bryant and O'Hallaron's *Computer Systems: A Programmer's Perspective* (3rd edition)
- Some recommended (but not required) textbooks
 - *Computer Organization and Design: The Hardware Software Interface*, ARM Edition. More emphasis on hardware. This is where I learnt Computer Systems.
 - *Introduction to Computing Systems: From Bits and Gates to C and Beyond*, 2/e.
 - *Structured Computer Organization*, 6/e. More emphasis on SW.

Outline: Class Introduction

- Introduction
 - What Are You Supposed to Learn in this Class?
 - Action Items
 - Instructor & TAs
 - Class Organization
 - How to Get Help?
 - **How Will You Be Evaluated?**
 - Policies

How Will You Be Evaluated?

- Programming Assignments: 40%
 - 5 assignments, 8% each
- 1 midterm exam, 25%
- 1 comprehensive final exam, 35%

Programming Assignments

- Check course webpage to figure out when they are due (there is a date and a time specified)
- They take time, so start early!

Programming Assignments

- Check course webpage to figure out when they are due (there is a date and a time specified)
- They take time, so start early!
 - Thinking/design time
 - Programming time
 - Test design + debug (and repeat)
- **3 slip days.** Use it wisely!

Programming Assignments

- Check course webpage to figure out when they are due (there is a date and a time specified)
- They take time, so start early!
 - Thinking/design time
 - Programming time
 - Test design + debug (and repeat)
- **3 slip days. Use it wisely!**
 - Tell us before the deadline, it's not automatically applied
 - Other than slip days, late submission counts 0 point
- You could work in pairs

Programming Assignments

- Check course webpage to figure out when they are due (there is a date and a time specified)
- They take time, so start early!
 - Thinking/design time
 - Programming time
 - Test design + debug (and repeat)
- **3 slip days. Use it wisely!**
 - Tell us before the deadline, it's not automatically applied
 - Other than slip days, late submission counts 0 point
- You could work in pairs
 - Only 1 submission per pair
 - Fill out a form if you work in pairs
- Share ideas but not artifacts (e.g., code, sketch)

Programming Environment

- Develop code (or at least test it) on the CSUG Linux boxes (csug.rochester.edu)
 - Microsoft Visual Studio could be nice, but it's not what we use
 - The lack of Unix knowledge is a major problem according to our industry contacts

Programming Environment

- Develop code (or at least test it) on the CSUG Linux boxes (csug.rochester.edu)
 - Microsoft Visual Studio could be nice, but it's not what we use
 - The lack of Unix knowledge is a major problem according to our industry contacts
- Projects will be mostly in C and x86 assembly.

Programming Environment

- Develop code (or at least test it) on the CSUG Linux boxes (csug.rochester.edu)
 - Microsoft Visual Studio could be nice, but it's not what we use
 - The lack of Unix knowledge is a major problem according to our industry contacts
- Projects will be mostly in C and x86 assembly.
- We only accept ANSI-C that can be compiled by the default GCC on the CSUG Linux boxes

Exams

- **Two exams: one in-class midterm and one final**
 - Midterm covers everything up until the second last lecture
 - Final will cover everything, including materials before midterm

Exams

- **Two exams: one in-class midterm and one final**
 - Midterm covers everything up until the second last lecture
 - Final will cover everything, including materials before midterm
- **No collaboration on exams**

Exams

- Two exams: one in-class midterm and one final
 - Midterm covers everything up until the second last lecture
 - Final will cover everything, including materials before midterm
- No collaboration on exams
- “I don’t know” is given 15% partial credit (from Yuhao)
 - You need to decide if guessing is worthwhile
 - Saves grading time
 - You have to write “I don’t know” and cross out /erase anything else to get credit: A blank answer doesn’t count

Exams

- Two exams: one in-class midterm and one final
 - Midterm covers everything up until the second last lecture
 - Final will cover everything, including materials before midterm
- No collaboration on exams
- “I don’t know” is given 15% partial credit (from Yuhao)
 - You need to decide if guessing is worthwhile
 - Saves grading time
 - You have to write “I don’t know” and cross out /erase anything else to get credit: A blank answer doesn’t count
- All exams are open book (means your book won’t help)
 - They will in fact probably **hurt**
 - **Memorization won’t help. Thinking will.**

Programming Assignments and Exams

The assignments were often very different from what we were learning in the course, causing a lot of frustration in the beginning. People had to rely on outside sources to get a grounding of how to do an assignment. ~~The test format was open book for which there were~~

The exams are structured to be significantly harder than the examples covered in class which means there is strong need for critical thinking on the spot during exams. ~~This can be hard when basic concepts themselves are difficult to grasp. There were also very few assignments to study from.~~

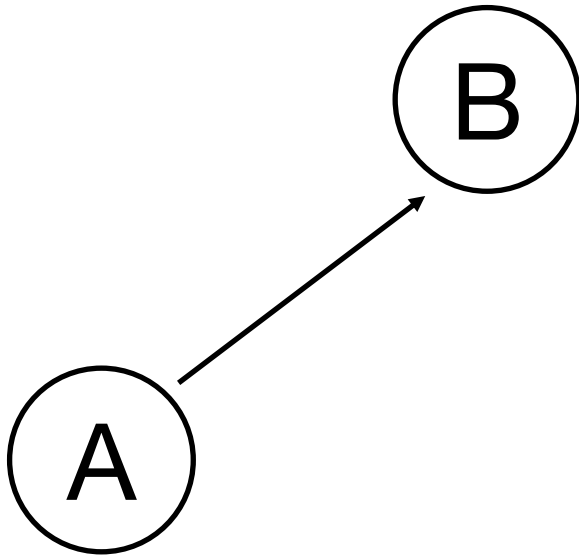
This is a feature, not a bug.

Here is how to think of the projects, exams, and lectures (loosely)...

- Lectures teach you how to go from **A** to **B**
- Projects ask you to go from **B** to **C**
- Exams test whether you can go from **A** to **C**

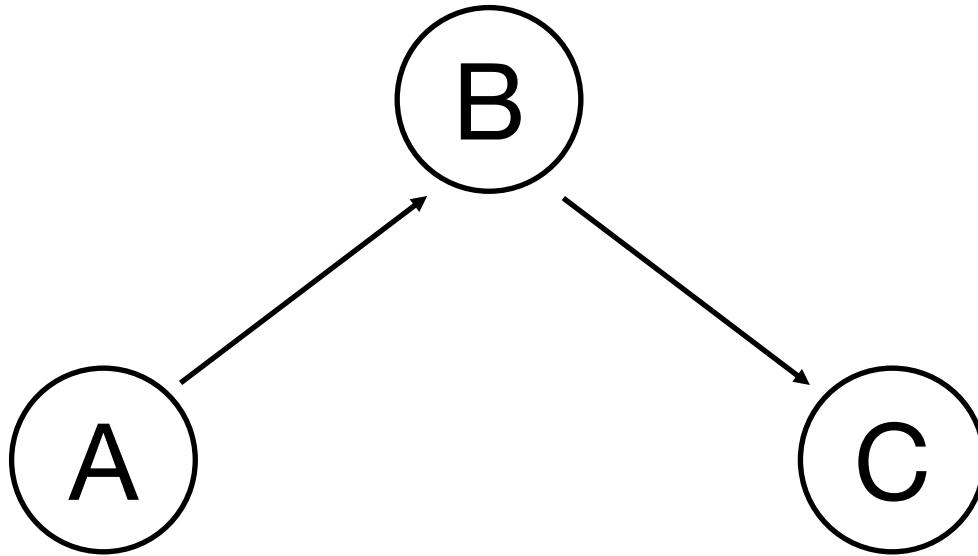
Here is how to think of the projects, exams, and lectures (loosely)...

- Lectures teach you how to go from **A** to **B**
- Projects ask you to go from **B** to **C**
- Exams test whether you can go from **A** to **C**



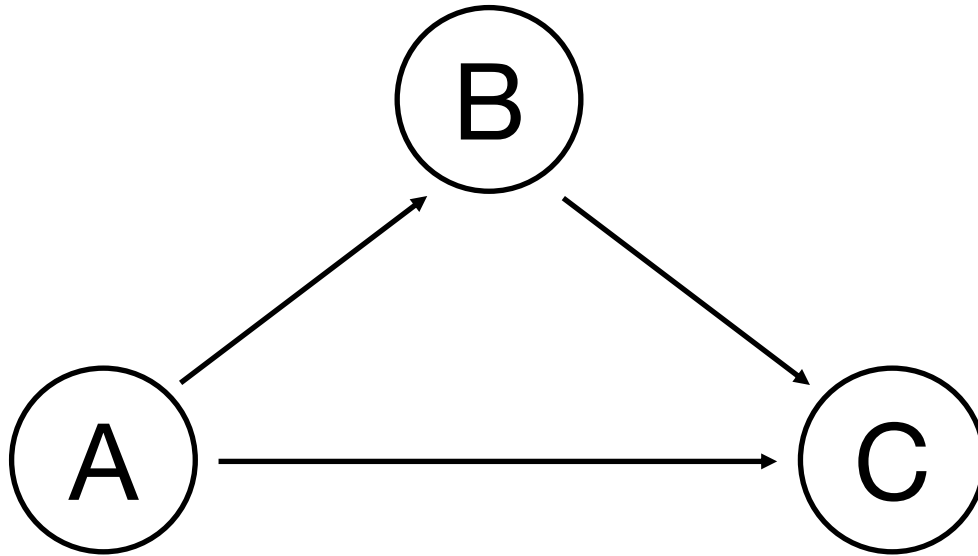
Here is how to think of the projects, exams, and lectures (loosely)...

- Lectures teach you how to go from **A** to **B**
- Projects ask you to go from **B** to **C**
- Exams test whether you can go from **A** to **C**



Here is how to think of the projects, exams, and lectures (loosely)...

- Lectures teach you how to go from **A** to **B**
- Projects ask you to go from **B** to **C**
- Exams test whether you can go from **A** to **C**



Example: A2--Bomblab

- What you need to do:
 - You will get a binary file (a bomb)
 - There are a few phases in the bomb
 - It expects to type in certain string in each phase when you execute the bomb
 - If you type in the correct string, you defuse the bomb
 - If you type in the wrong string, the bomb explodes and you lose some points (and I am notified)

Example: A2--Bomblab

- What you need to do:
 - You will get a binary file (a bomb)
 - There are a few phases in the bomb
 - It expects to type in certain string in each phase when you execute the bomb
 - If you type in the correct string, you defuse the bomb
 - If you type in the wrong string, the bomb explodes and you lose some points (and I am notified)
- How to do this?
 - Learn to use gdb
 - Use gdb to figure out the instructions in this binary
 - Guess the c code/meaning/logic of these instructions
 - Figure out what it wants you to type

Example: A2--Bomblab

- How to do this?
 - Learn to use gdb
 - Use gdb to figure out the instructions in this binary
 - Guess the c code/meaning/logic of these instructions
 - Figure out what it wants you to type

Example: A2--Bomblab

- What do we teach you?
 - Instructions VS Executable Binary
 - Instruction Format
 - Common Instructions
- How to do this?
 - Learn to use gdb
 - Use gdb to figure out the instructions in this binary
 - Guess the c code/meaning/logic of these instructions
 - Figure out what it wants you to type

Outline: Class Introduction

- Introduction
 - What Are You Supposed to Learn in this Class?
 - Action Items
 - Instructor & TAs
 - Class Organization
 - How to Get Help?
 - How Will You Be Evaluated?
 - **Policies**

Academic Honesty (TBC)

- Exams in CSC 252/452 must be strictly individual work.

Academic Honesty (TBC)

- Exams in CSC 252/452 must be strictly individual work.
- Collaboration on assignments across teams (or among individuals on non-team-based assignments) is encouraged **at the level of ideas**. Feel free to ask each other questions, brainstorm on algorithms, or work together at a whiteboard. You may not claim work as your own, however, unless you **transform the ideas into substance by yourself**. This means you must leave any brainstorming sessions with no written notes.

Academic Honesty (TBC)

- Exams in CSC 252/452 must be strictly individual work.
- Collaboration on assignments across teams (or among individuals on non-team-based assignments) is encouraged **at the level of ideas**. Feel free to ask each other questions, brainstorm on algorithms, or work together at a whiteboard. You may not claim work as your own, however, unless you **transform the ideas into substance by yourself**. This means you must leave any brainstorming sessions with no written notes.
- Similarly, you are welcome to read anything you find on the web, but you must close all web pages before beginning to write your code. You are not permitted to repeatedly consult a source. You can read it, understand it, put it away, and write your own similar code, but you **must not copy anything**. Both electronic copy-and-paste and copying through short-term memory are expressly forbidden.

Academic Honesty (Cont'd)

- To minimize opportunities to steal code, all students must protect the directories in which they do their work.

Academic Honesty (Cont'd)

- To minimize opportunities to steal code, all students must protect the directories in which they do their work.
- Any activity that has the effect of significantly impairing the ability of another student to learn is expressly forbidden. Examples here might include destroying the work of others, interfering with their access to resources, or deliberately providing them with misleading information. (Note too that grades in CSC 252/452 are assigned on the basis of individual merit, so there is no benefit, even a dishonest one, to be gained by sabotaging the work of others.

Academic Honesty (Cont'd)

- To minimize opportunities to steal code, all students must protect the directories in which they do their work.
- Any activity that has the effect of significantly impairing the ability of another student to learn is expressly forbidden. Examples here might include destroying the work of others, interfering with their access to resources, or deliberately providing them with misleading information. (Note too that grades in CSC 252/452 are assigned on the basis of individual merit, so there is no benefit, even a dishonest one, to be gained by sabotaging the work of others.
- Posting homework and project solutions to public repositories on sites like GitHub is a violation of the College's Academic Honesty Policy, Section V.B.2 "Giving Unauthorized Aid."

CSC 252/452 Will Be Hard

- Be prepared
- We will go back and forth between software and hardware layers
 - For ISA
 - ISA with C programming
 - ISA with microarchitecture
- It covers many aspects of computer systems
 - CSC 252/452 is programmer centric
 - It enables you to write programs that are more reliable and efficient

Welcome and Enjoy!