

ECE 2400 / ENGRD 2140
Computer Systems Programming
Course Overview

Anne Bracy

School of Electrical and Computer Engineering
Cornell University

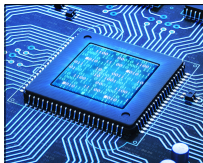
<http://www.csl.cornell.edu/courses/ece2400>



Application-Level
Software



System-Level
Software



ECE 2400 / ENGRD 2140

Computer Systems Programming

What is Computer Systems Programming?

Course Logistics

Applications vs. Technology

Application



Gap too large to bridge in one step
(but there are exceptions,
e.g., a magnetic compass)



Technology

Applications vs. Technology

Application



Technology

Applications vs. Technology

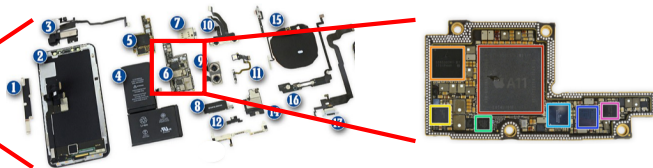
Application



Technology

Applications vs. Technology

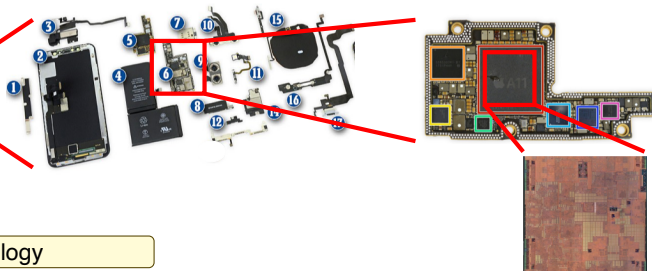
Application



Technology

Applications vs. Technology

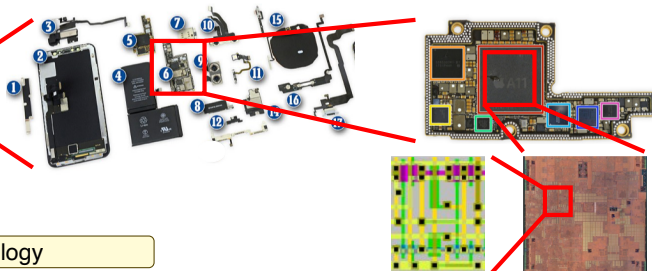
Application



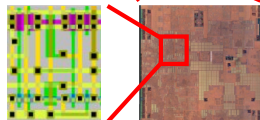
Technology

Applications vs. Technology

Application

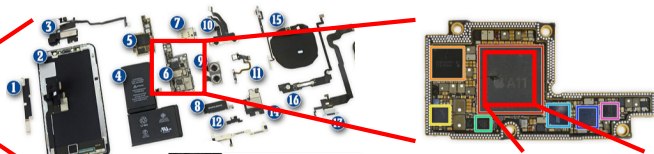


Technology

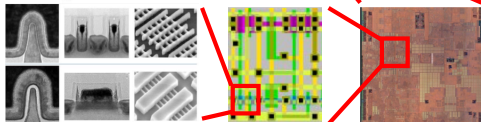


Applications vs. Technology

Application

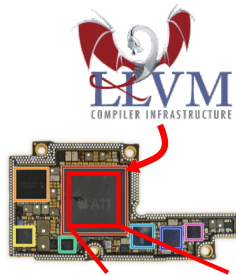
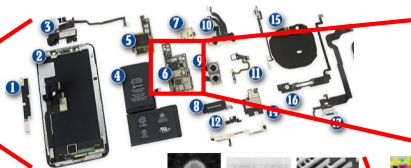


Technology

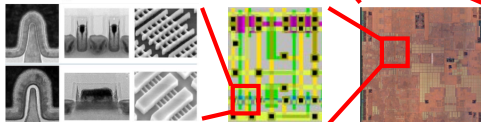


Applications vs. Technology

Application

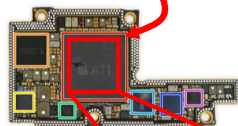
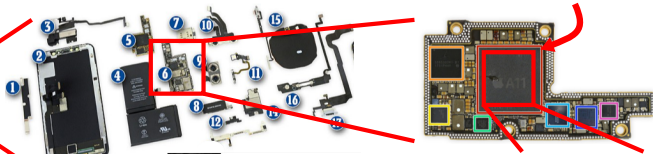


Technology

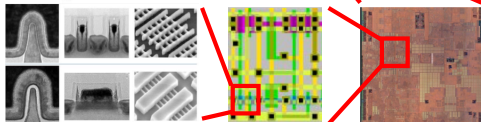


Applications vs. Technology

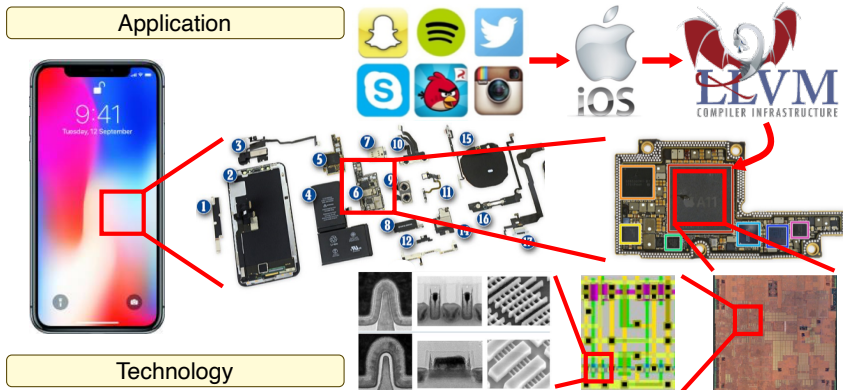
Application



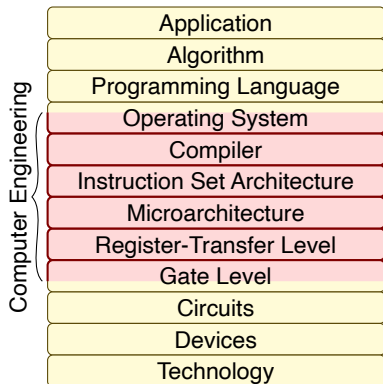
Technology



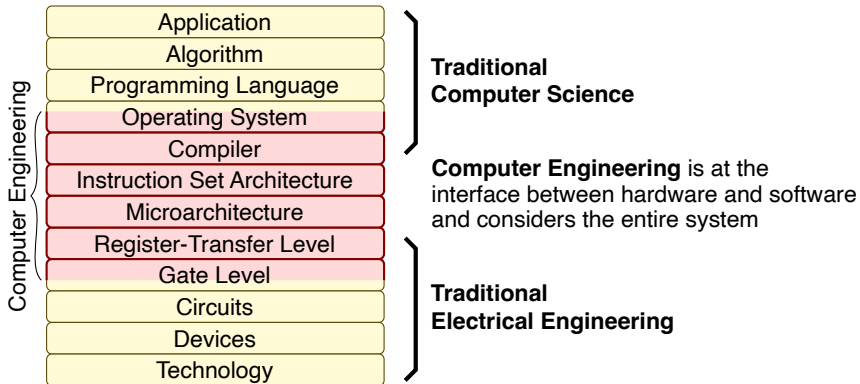
Applications vs. Technology



The Computer Systems Stack



The Computer Systems Stack

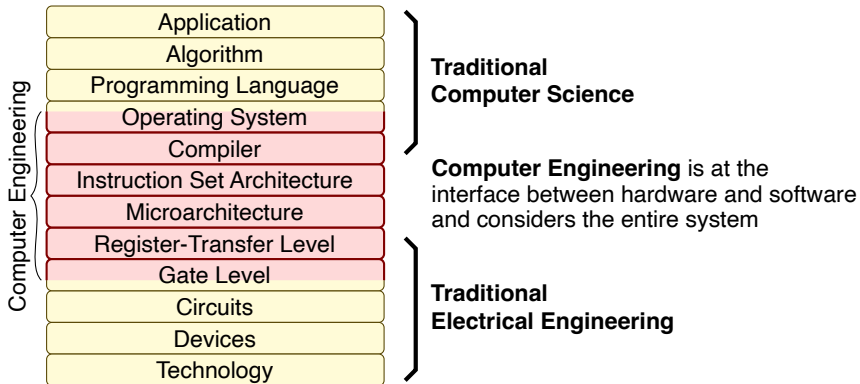


**Traditional
Computer Science**

Computer Engineering is at the interface between hardware and software and considers the entire system

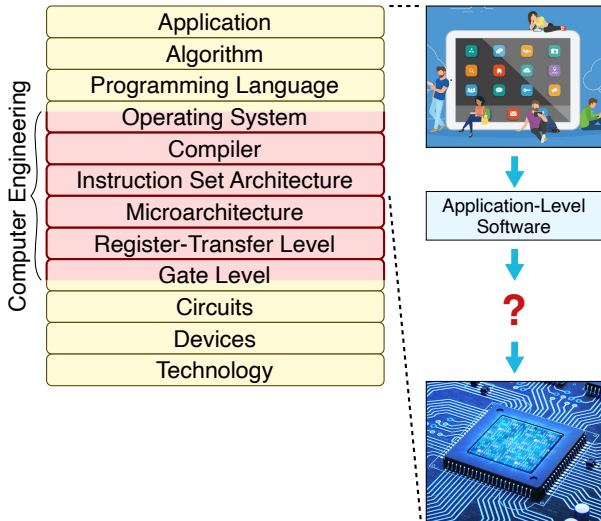
**Traditional
Electrical Engineering**

The Computer Systems Stack



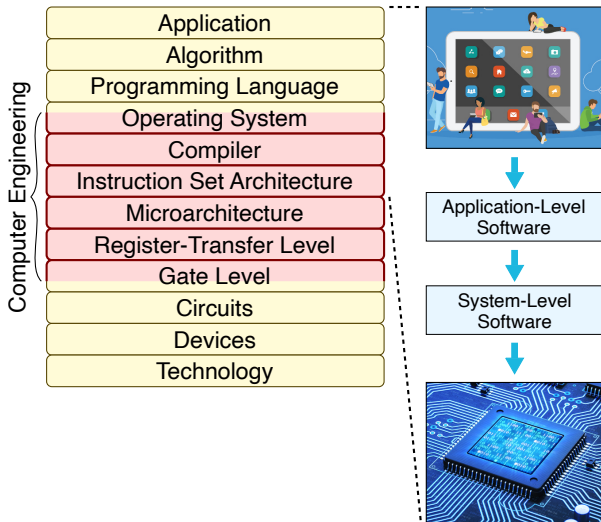
In its broadest definition, computer engineering is the **development of the abstraction/implementation layers** that allow us to execute information processing **applications** efficiently using available manufacturing **technologies**

Python for Application-Level Programming



- ▶ High-level, user-facing software
- ▶ Enable productively developing applications that provide new functionality to users
- ▶ Enable productively collecting, analyzing, visualizing data
- ▶ Sometimes called a productivity-level language

C/C++ for System-Level Programming

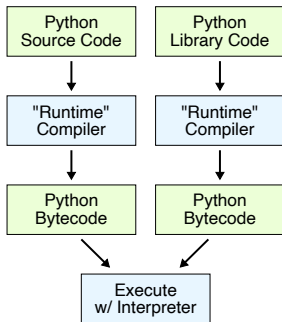


- ▶ Connects application software to the low-level computer hardware
- ▶ Enables carefully managing performance and resource constraints
- ▶ Sometimes called an efficiency-level language

Dynamically Interpreted vs. Statically Compiled

```
def min(a,b):
    if a < b:
        c = a
    else:
        c = b
    return c
```

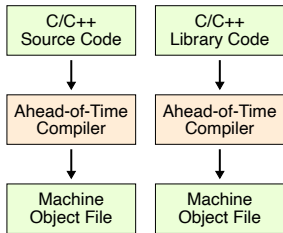
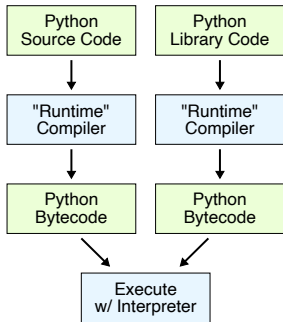
```
LOAD_FAST
LOAD_FAST
COMPARE_OP
POP_JUMP_IF_F
LOAD_FAST
STORE_FAST
JUMP_FORWARD
LOAD_FAST
STORE_FAST
LOAD_FAST
RETURN_VALUE
```



Dynamically Interpreted vs. Statically Compiled

```
def min(a,b):
    if a < b:
        c = a
    else:
        c = b
    return c
```

```
LOAD_FAST
LOAD_FAST
COMPARE_OP
POP_JUMP_IF_F
LOAD_FAST
STORE_FAST
JUMP_FORWARD
LOAD_FAST
STORE_FAST
LOAD_FAST
RETURN_VALUE
```



```
int min( int a,
        int b )
{
    int c;
    if ( a < b )
        c = a;
    else
        c = b;
    return c;
}
```

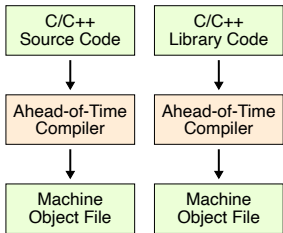
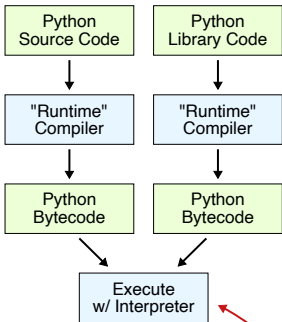
```
pushq %rbp
movq %rsp,%rbp
cmpl %esi,%edi
cmovl %edi,%esi
movl %esi,%eax
popq %rbp
retq
```

```
1010101
100100010001001
11100111110111
111101001110111
100010011111000
1011101
11000011
```

Dynamically Interpreted vs. Statically Compiled

```
def min(a,b):
    if a < b:
        c = a
    else:
        c = b
    return c
```

```
LOAD_FAST
LOAD_FAST
COMPARE_OP
POP_JUMP_IF_F
LOAD_FAST
STORE_FAST
JUMP_FORWARD
LOAD_FAST
STORE_FAST
LOAD_FAST
RETURN_VALUE
```



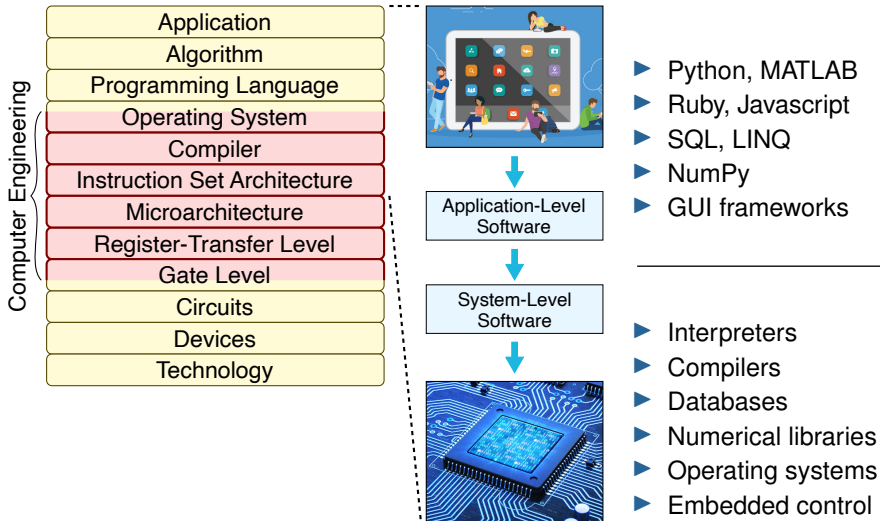
```
int min( int a,
        int b )
{
    int c;
    if ( a < b )
        c = a;
    else
        c = b;
    return c;
}
```

```
pushq %rbp
movq %rsp,%rbp
cmpl %esi,%edi
cmovl %edi,%esi
movl %esi,%eax
popq %rbp
retq
```

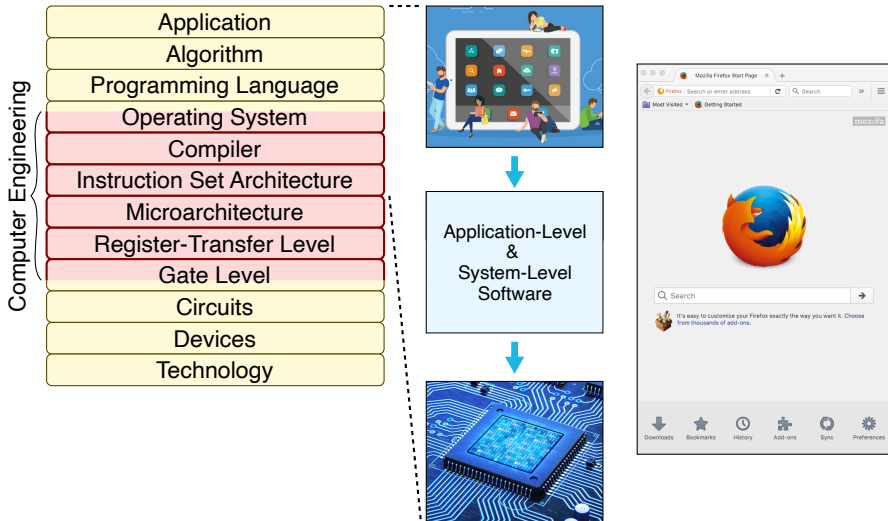
```
1010101
100100010001001
11100111110111
111101001110111
100010011111000
1011101
11000011
```

The standard Python interpreter is called CPython and it is written in C!

Computer Systems Programming is Diverse



Aside: C/C++ for Application-Level Software



A Tale of Two Programming Languages



Python Programming Language

- ▶ Introduced: 1991
- ▶ Most of the machine details are hidden from programmer
- ▶ Programmer gives up some control for improved productivity
- ▶ Easily supports multiple programming paradigms
- ▶ Extensive standard library is included
- ▶ Slow and memory inefficient

C/C++ Programming Language

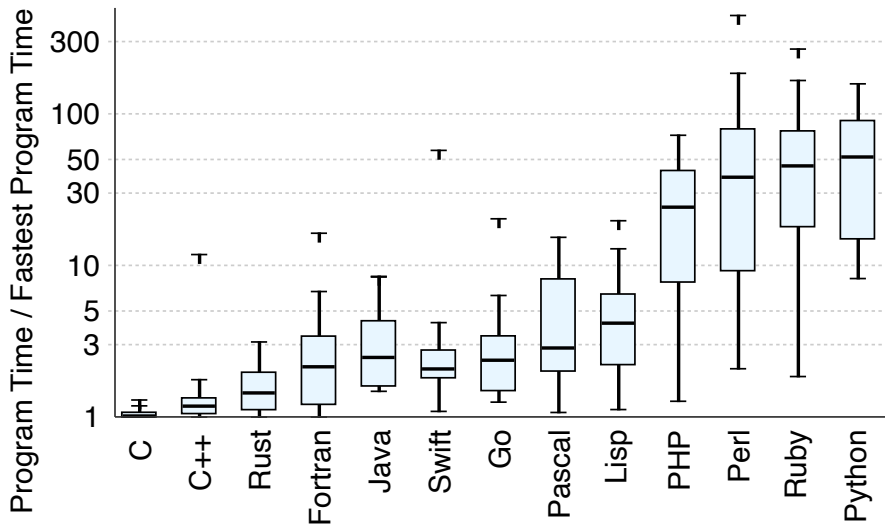
- ▶ Introduced: 1972(C), 1979(C++)
- ▶ Most of the machine details are exposed to the programmer
- ▶ Programmer is in complete control for improved efficiency
- ▶ Easily supports multiple programming paradigms
- ▶ More limited standard library is included
- ▶ Fast and memory efficient

Comparing the Popularity of Python vs. C/C++

Rank	Language	Type	Score
1	Python	  	100.0
2	Java	  	95.4
3	C	  	94.7
4	C++	  	92.4
5	JavaScript		88.1
6	C#	   	82.4

The 2021 Top Programming Languages, IEEE Spectrum

Comparing the Performance of Python vs. C/C++



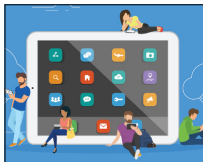
The Computer Language Benchmarks Game

Program = Algorithm + Data Structure

While this course covers C/C++ and system-level programming, this course also builds off of your prior programming experience to further develop your understanding of algorithms and data structures



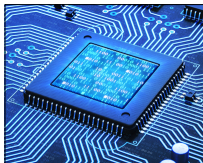
- ▶ **Algorithm:** Clear set of steps to solve any problem instance in a particular class of problems
- ▶ **Data Structure:** Way of efficiently organizing and storing data along with operations for accessing and manipulating this data



Application-Level
Software



System-Level
Software



ECE 2400 / ENGRD 2140

Computer Systems Programming

What is Computer Systems Programming?

Course Logistics

Who am I?



*Stanford
University*

- BA, German, BS, Sym Sys
- MS, Computer Science



*University of
Pennsylvania*

- PhD, Computer Science



Intel Labs

- Research Scientist



*Washington
University in
St. Louis*

- Principal Lecturer



Cornell University

- ~~Senior Lecturer~~ Teaching Prof.
 - 1110, 2110, 3410, 4410, 4411
 - 4750, 2400, 2300 (labs), ... ??

<https://canvas.cornell.edu/courses/82877/pages/meet-the-course-staff>

Graduate Teaching Assistants

**Anusha
Muralidharan**

Jifeng Wu

Colin Muessig

Undergraduate Teaching Assistants

Elizabeth Kuntz

Joseph Folsom

Linh Anh Nguyen

Ludvig Fellstrom

Mason Drew

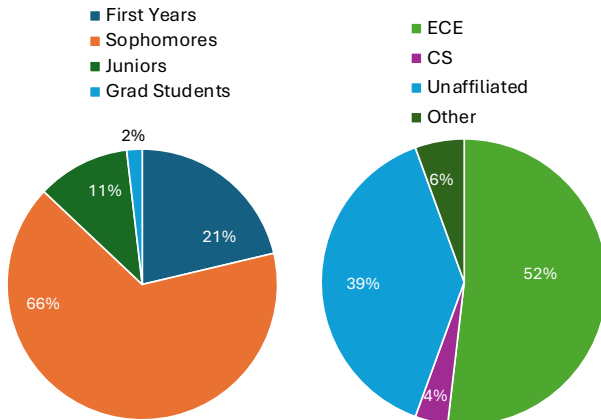
Natalie Yeung

Rachael Godwin

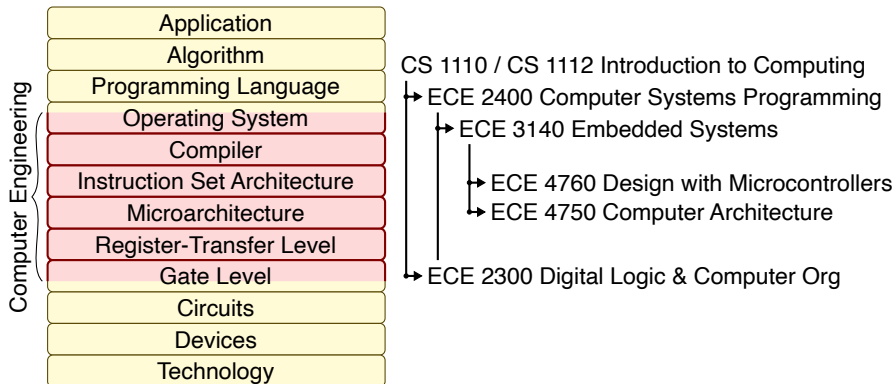
Sara Liu

Sean Tao

Who are you?



ECE 2400 Within the Engineering Curriculum



ECE 2400 is also an ENGRD and thus satisfies the engineering distribution requirement

ECE 2400 can be an excellent way to generally incorporate programming into your non-ECE engineering curriculum

Course Objectives

- ▶ **describe** a variety of algorithms and data structures and how to analyze these algorithms and data structures in terms of time and space complexity
- ▶ **apply** the C/C++ programming languages to implement algorithms and data structures using different programming paradigms
- ▶ **evaluate** algorithm and data structure alternatives and make a compelling qualitative and/or quantitative argument for one approach
- ▶ **create** non-trivial C/C++ programs (roughly 1,000 lines of code) and the associated testing strategy from an English language specification
- ▶ **write** concise yet comprehensive technical reports that describe a program implemented in C/C++, explain the testing strategy used to verify functionality, and evaluate the program to characterize its performance and memory usage

Course Structure

▶ Part 1: Procedural Programming

- ▷ introduction to C; variables; expressions; functions; conditional & iteration statements; recursion; static types; pointers; arrays; dynamic allocation

▶ Part 2: Basic Algorithms and Data Structures

- ▷ lists; vectors; complexity analysis; sorting algorithms: insertion, selection, merge, quick, radix; ADTs: stacks, queues, priority queues, sets, maps

▶ Part 3: Multi-Paradigm Programming

- ▷ transition to C++; namespaces; flexible function prototypes; references; exceptions; new/delete; *object oriented programming*: C++ classes and inheritance for dynamic polymorphism; *generic programming*: C++ templates for static polymorphism; *concurrent programming*: C++ threads and atomics

▶ Part 4: More Algorithms and Data Structures

- ▷ trees (binary search trees; binary heaps); tables (lookup tables; hash tables); graphs (DFS, BFS, shortest path, minimum spanning trees)

zyBook: Interactive, Online Textbook

The screenshot shows a web browser window displaying the zyBook interface. The browser address bar shows the URL `learn.zybooks.com/zybook/BattenC++CDSEAug2020/chapter/1/section/2`. The page title is "zyBooks" and the navigation path is "My library > Programming in C with zyLabs home > 1.2: Programming basics".

On the left side, there is a navigation menu with a search bar and a list of topics. The current topic, "1.2 Programming basics", is highlighted. Other topics include "1.1 Programming (general)", "1.3 Comments and whitespace", "1.4 Errors and warnings", "1.5 Variables and assignments (general)", "1.6 Variables (int)", "1.7 Identifiers", "1.8 Arithmetic expressions (general)", "1.9 Arithmetic expressions (int)", "1.10 Example: Health data", "1.11 If-else branches (general)", "1.12 If-else", "1.13 Equality and relational operators", "1.14 Logical operators", "1.15 Example: Toll calculation", "1.16 Loops (general)", "1.17 Switch statements", and "1.18 While loops".

The main content area displays the chapter title "1.2 Programming basics" with "Present" and "Note" icons. Below the title is the section "A first program" with the text "A simple C program appears below." followed by a bulleted list:

- A **program** starts in `main()`, executing the statements within `main`'s braces {}, one at a time.
- Each statement typically appears alone on a line and ends with a **semicolon**, as English sentences end with a period.
- The `int wage` statement creates an integer variable named `wage`. The `wage = 20` statement assigns `wage` with 20.
- The `printf` statements output various values.
- The `return 0` statement ends the program (the 0 tells the operating system the program ended without error).

Below the list, it says "The following code (explained later) at the top of a file enables the program to get input and put output:" followed by a code block:

```
#include <stdio.h>
```

Below the code block, there is a "PARTICIPATION ACTIVITY" section titled "1.2.1: Program execution begins with `main`, then proceeds one statement at a time." with a "Start" button and a "2x speed" checkbox. To the right of the activity, there is a diagram showing a memory stack with a box labeled "20" and the variable name "wage" next to it.

Likely Programming Assignments

▶ PA1–3: Fundamentals

- ▶ PA1: Math functions
- ▶ PA2: List and Vector Data Structures
- ▶ PA3: Sorting Algorithms

▶ PA4–5: Handwriting Recognition System

- ▶ PA4: Linear vs. Binary Searching
- ▶ PA5: Trees vs. Tables

▶ Every programming assignment involves

- ▶ C/C++ “agile” programming
- ▶ State-of-the-art tools for build systems, version control, continuous integration, code coverage
- ▶ Performance measurement
- ▶ Short technical report



Exam Groups

- ▶ **Two-phased Prelims:**
 - ▷ part 1: by yourself
 - ▷ part 2: in groups
 - ▷ your prelim score will be a 90/10 weighted average
 - ▷ (but your group score cannot lower your grade)

- ▶ **Who makes these groups?**
 - ▷ We do. (mostly random)

- ▶ **How will I get to know my group?**
 - ▷ a few in-class activities/quizzes
 - ▷ these will be announced in advance
 - ▷ for example: first quiz will be next Wednesday

Frequently Asked Questions

- ▶ I have not taken CS 1110 nor CS 1112, can I take this class?
 - ▷ We assume some basic programming experience, discuss with instructor
- ▶ **ECE Majors** – How does ECE 2400 satisfy degree requirements?
 - ▷ ECE 2400 can count as your second ENGRD course
 - ▷ ECE 2400 can count as an outside-ECE technical elective
 - ▷ ECE 2400 satisfies the ECE advanced programming requirement
- ▶ **CS Majors** – Can I use ECE 2400 in place of CS 2110?
 - ▷ Yes but you should probably take CS 2110 (and they can't both be your ENGRD)
- ▶ **ECE/CS Dual Majors** – Can I use ECE 2400 in place of CS 2110?
 - ▷ Yes! (but it may delay your CS affiliation)
- ▶ **CS Minors** – Can I use ECE 2400 in place of CS 2110?
 - ▷ Absolutely!

Frequently Asked Questions

- ▶ **Other Majors** – How does ECE 2400 satisfy degree requirements?
 - ▷ ECE 2400 can count as one of your two required ENGRD courses
 - ▷ CS 2110 and ECE 2400 are in the same ENGRD category, so you cannot use both of them as your two ENGRD courses

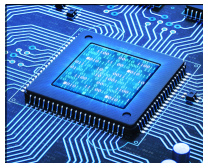
- ▶ Can I take both ECE 2400 and CS 2110?
 - ▷ Sure! (recall popularity and performance data)



Application-Level
Software



System-Level
Software



Take-Away Points

- ▶ Computer systems programming involves developing software to **connect** the low-level computer hardware to high-level, user-facing application software and usually **requires careful consideration of performance and resource constraints**
- ▶ We are entering an **exciting era** where computer systems programming will play a **critical role in enabling both cloud computing and the internet-of-things**