



The Plan: Lab 1 preview

Program, Application

Programming Language

Compiler/Interpreter

Operating System

Instruction Set Architecture

Microarchitecture

Digital Logic

Devices (transistors, etc.)

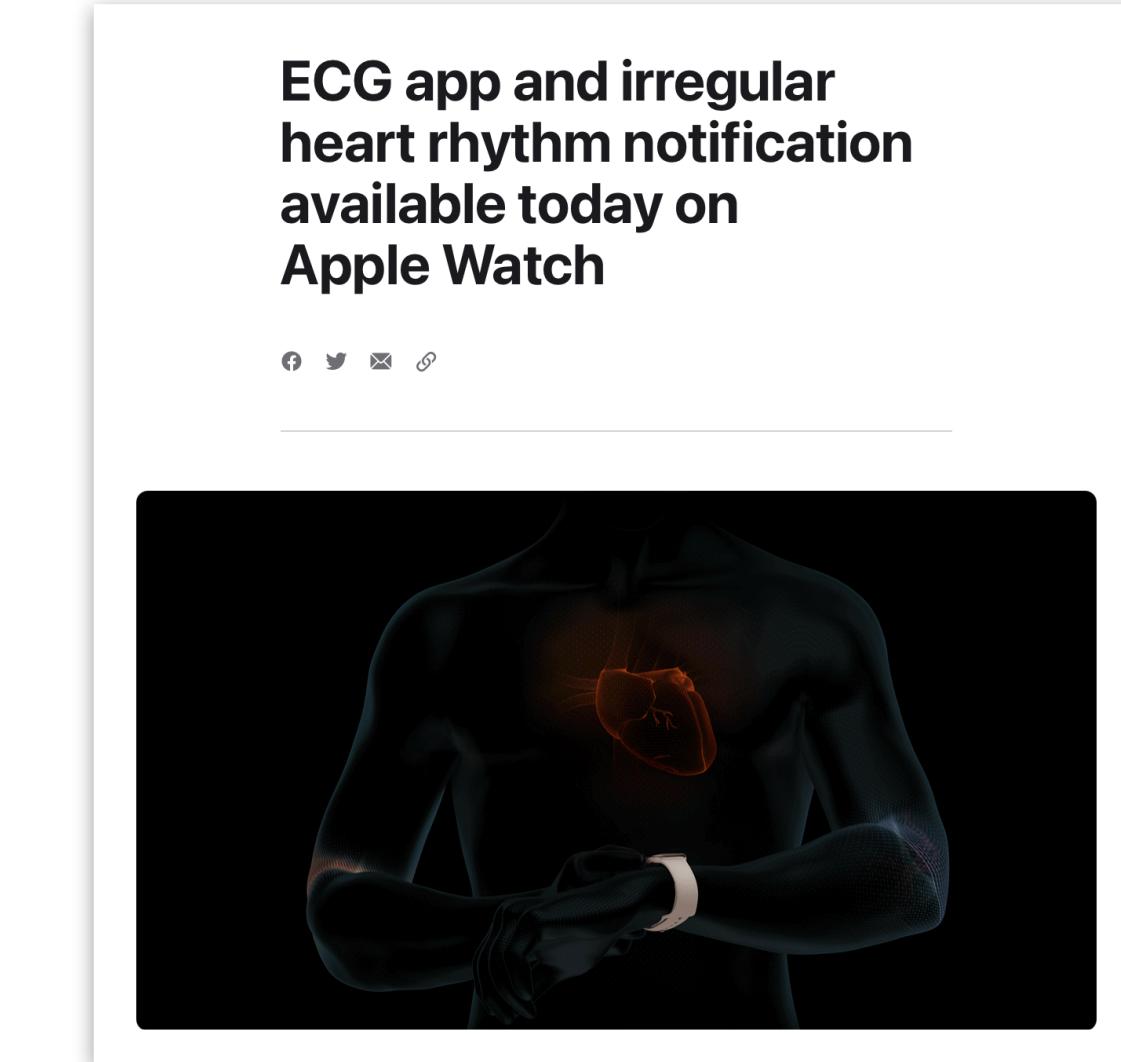
Solid-State Physics

Welcome to

CS 240: Foundations of Computer Systems!

Your lecture instructor: Alexa VanHattum

Note: you can call me “Alexa”, “Prof. Alexa”, or “Prof. VanHattum”



- 3rd year at Wellesley
- Research focus: programming languages & systems

Before Wellesley:

- PhD in Computer Science at Cornell
- Software engineer for Apple health (heart monitoring)
- **THIS CLASS** one of the most helpful across industry *and* research

Today

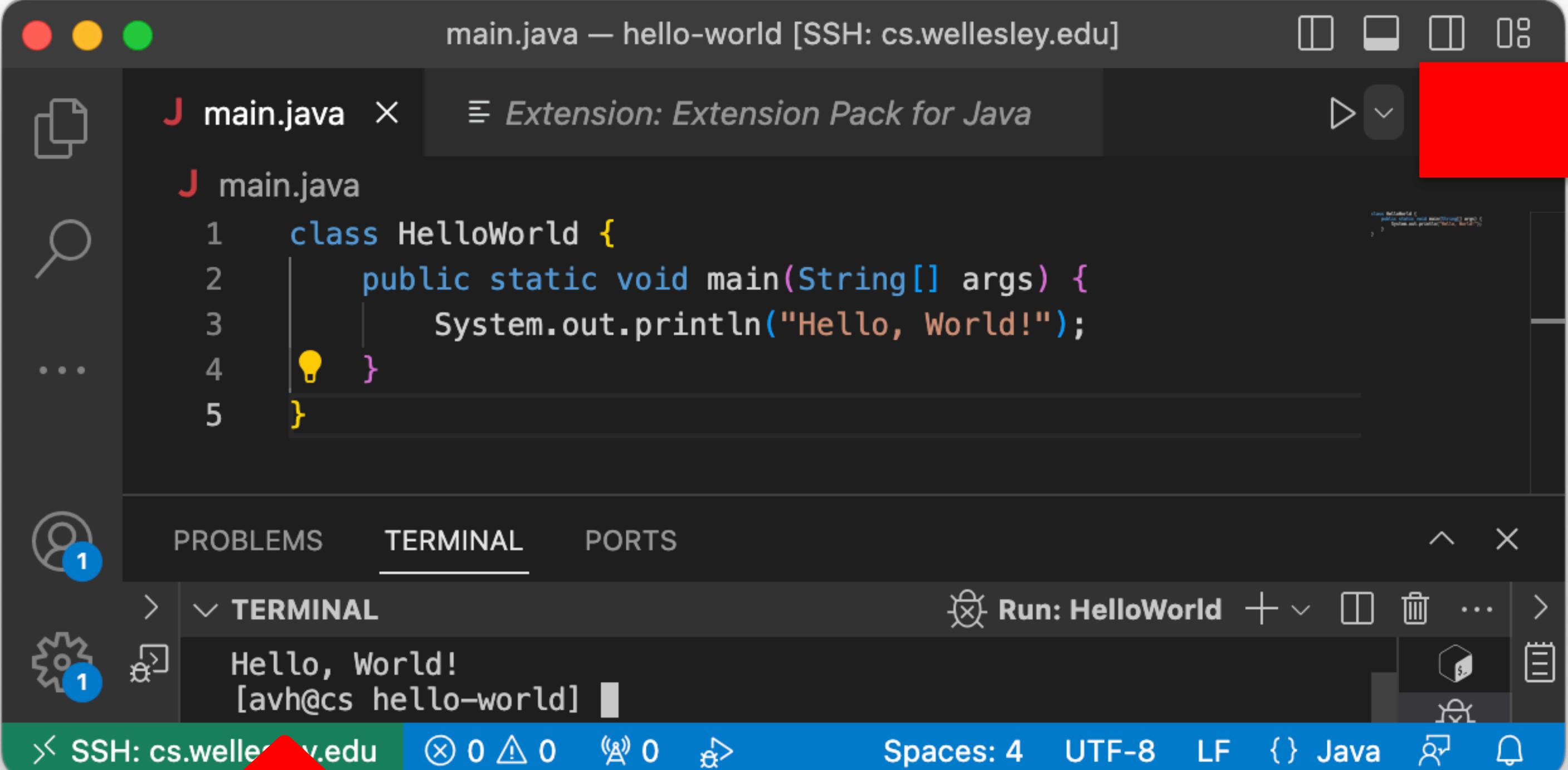
- 1 **What is CS 240?**
- 2 Why take CS 240? (in brief)
- 3 How does CS 240 work? (in brief)

CS 111, 230, 231, 235, 251:

- How do you use programming to solve a problem?
- How do you structure a program?
- How do you know it is correct or efficient?
- How hard is it to solve a problem?
- How is computation expressed?
- What does a program mean?
- ...

A BIG question is missing...

1



main.java — hello-world [SSH: cs.wellesley.edu]

J main.java X Extension: Extension Pack for Java

J main.java

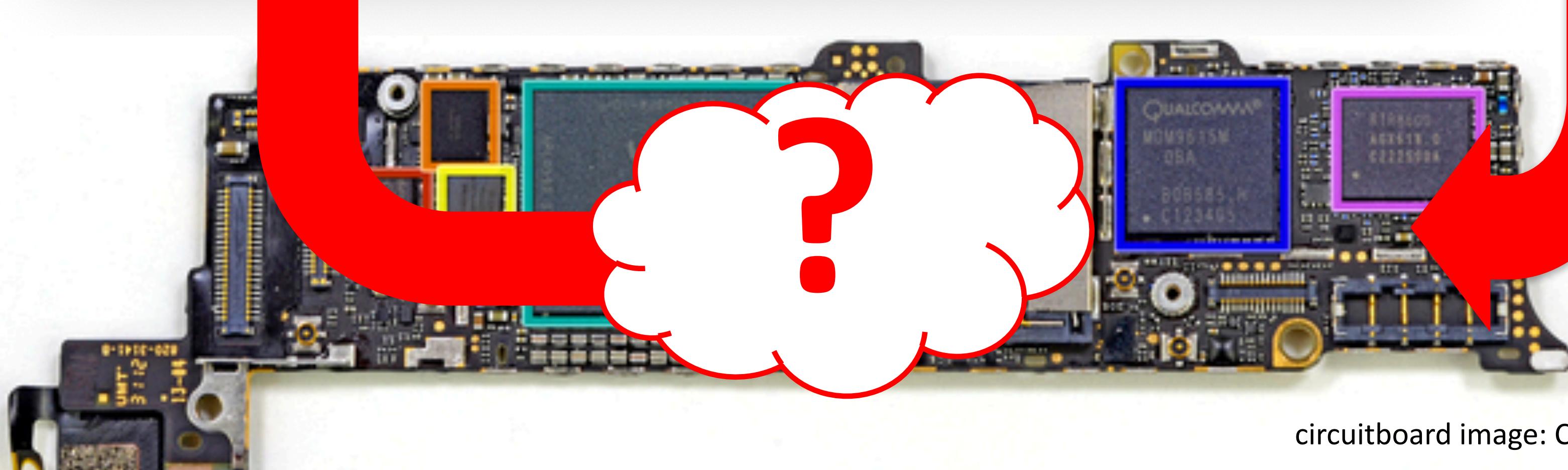
```
1 class HelloWorld {  
2     public static void main(String[] args) {  
3         System.out.println("Hello, World!");  
4     }  
5 }
```

PROBLEMS TERMINAL PORTS

1 > < TERMINAL Run: HelloWorld + < >

1 Hello, World!
[avh@cs hello-world]

SSH: cs.wellesley.edu 0 0 0 0 Spaces: 4 UTF-8 LF {} Java



CS 240

CS 111, 230,
231, 235, 251

Programming Language

Compiler/Interpreter

Operating System

Instruction Set Architecture

Microarchitecture

Digital Logic

Devices (transistors, etc.)

Solid-State Physics

Big Idea:
Abstraction

interface

implementation

*Layers manage
complexity.*

Programming Language

Compiler/Interpreter

Operating System

Instruction Set Architecture

Microarchitecture

Digital Logic

Devices (transistors, etc.)

Solid-State Physics

Big Idea: Abstraction

with a few recurring subplots

Simple, general interfaces:

- Hide complexity of efficient implementation.
- Make higher-level systems easy to build.

Representation of data and programs

0s and 1s,
electricity

Translation of data and programs

compilers,
assemblers,
decoders

Control flow within/across programs

branches,
procedures,
operating
system

Hardware

Software

Desired computation
in a programming language

Hardware/Software Interface

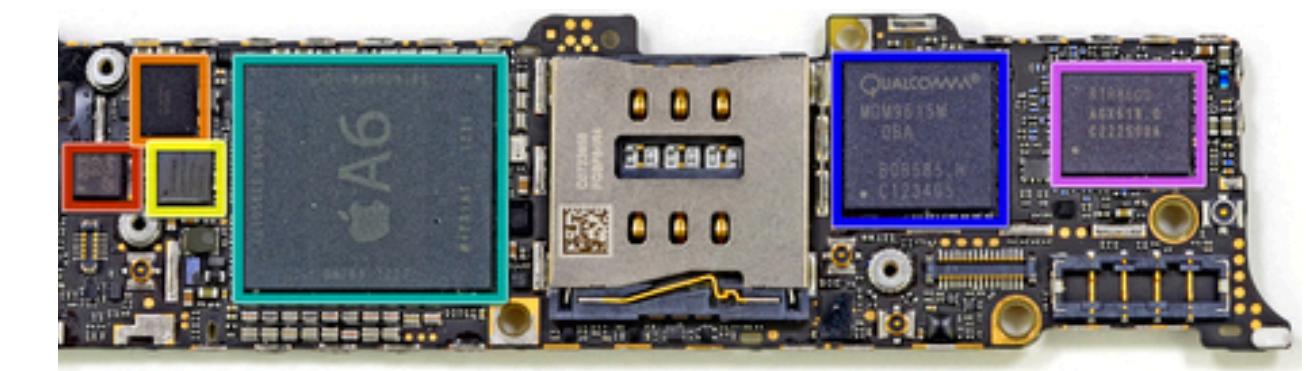
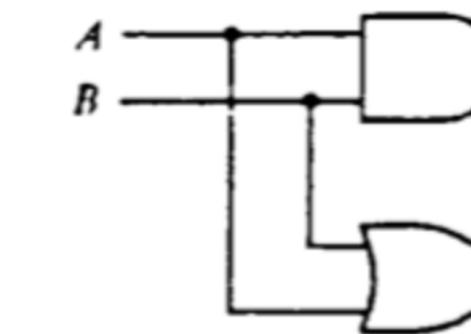
Abstraction!

Physical implementation
with circuits and electricity.

CS 240 in 3 acts (4-5 weeks each)

1. Hardware *implementation*

From transistors to a simple computer



2. Hardware-software *interface*

From instruction set architecture to programming in C

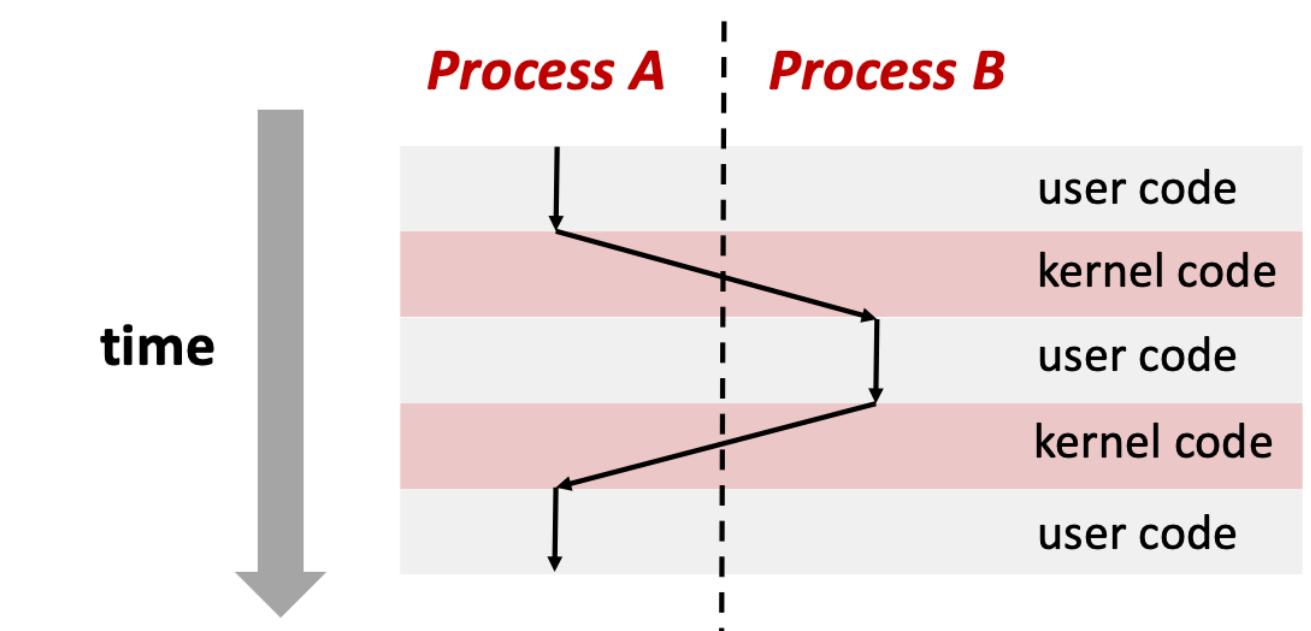
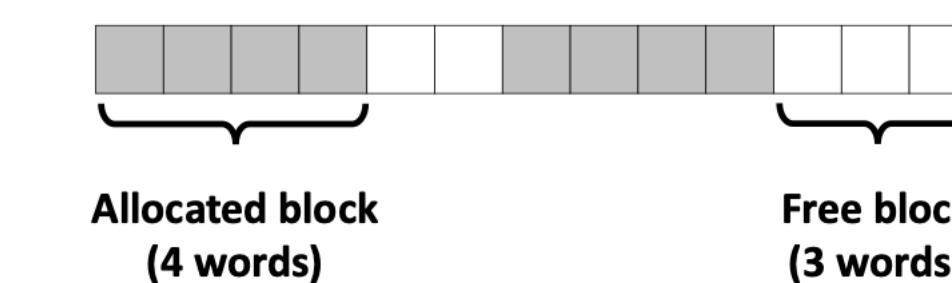
```
MOV x9, x10
ADD x12, x12, #1
*x = malloc(...);
```

3. Abstraction for practical systems

Memory hierarchy

Operating system basics

Higher-level languages and tools



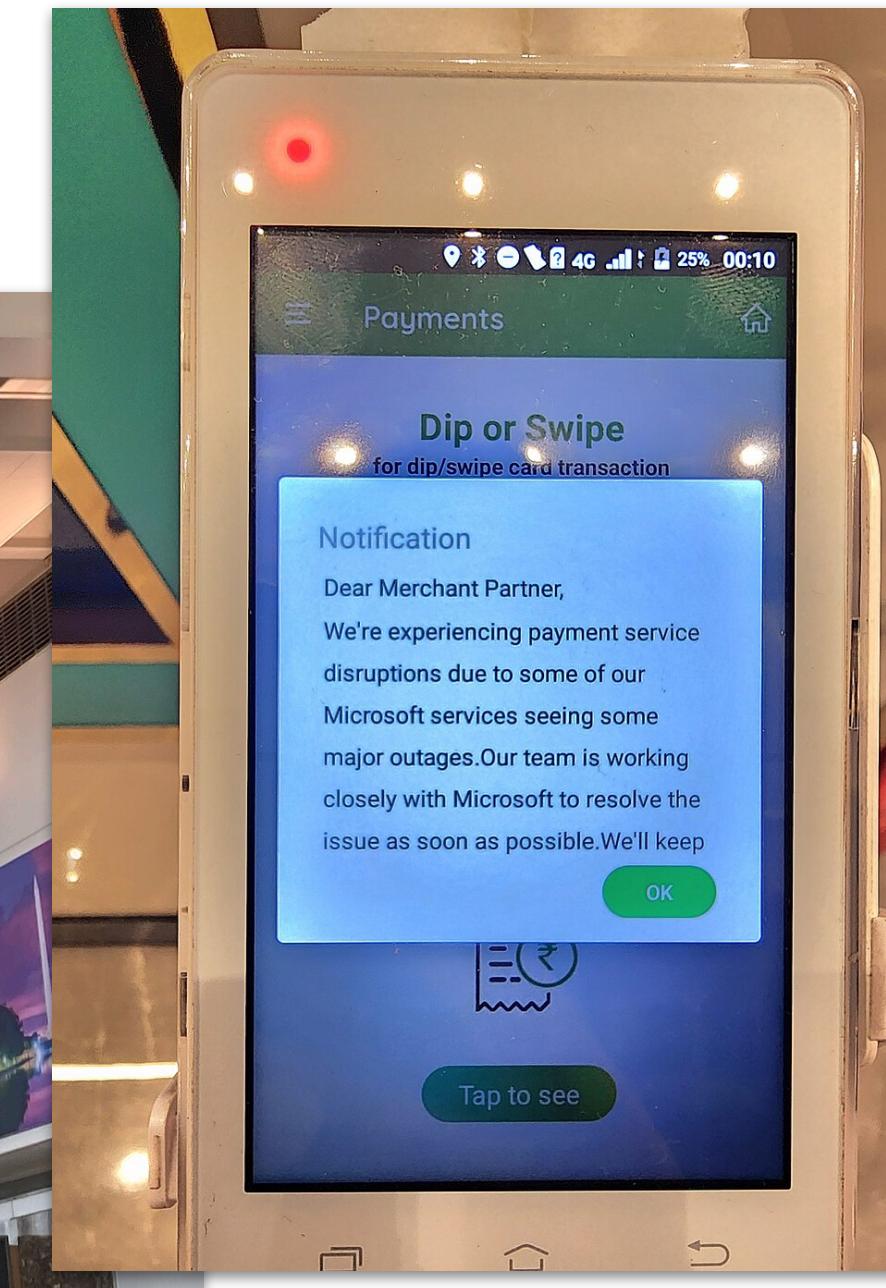
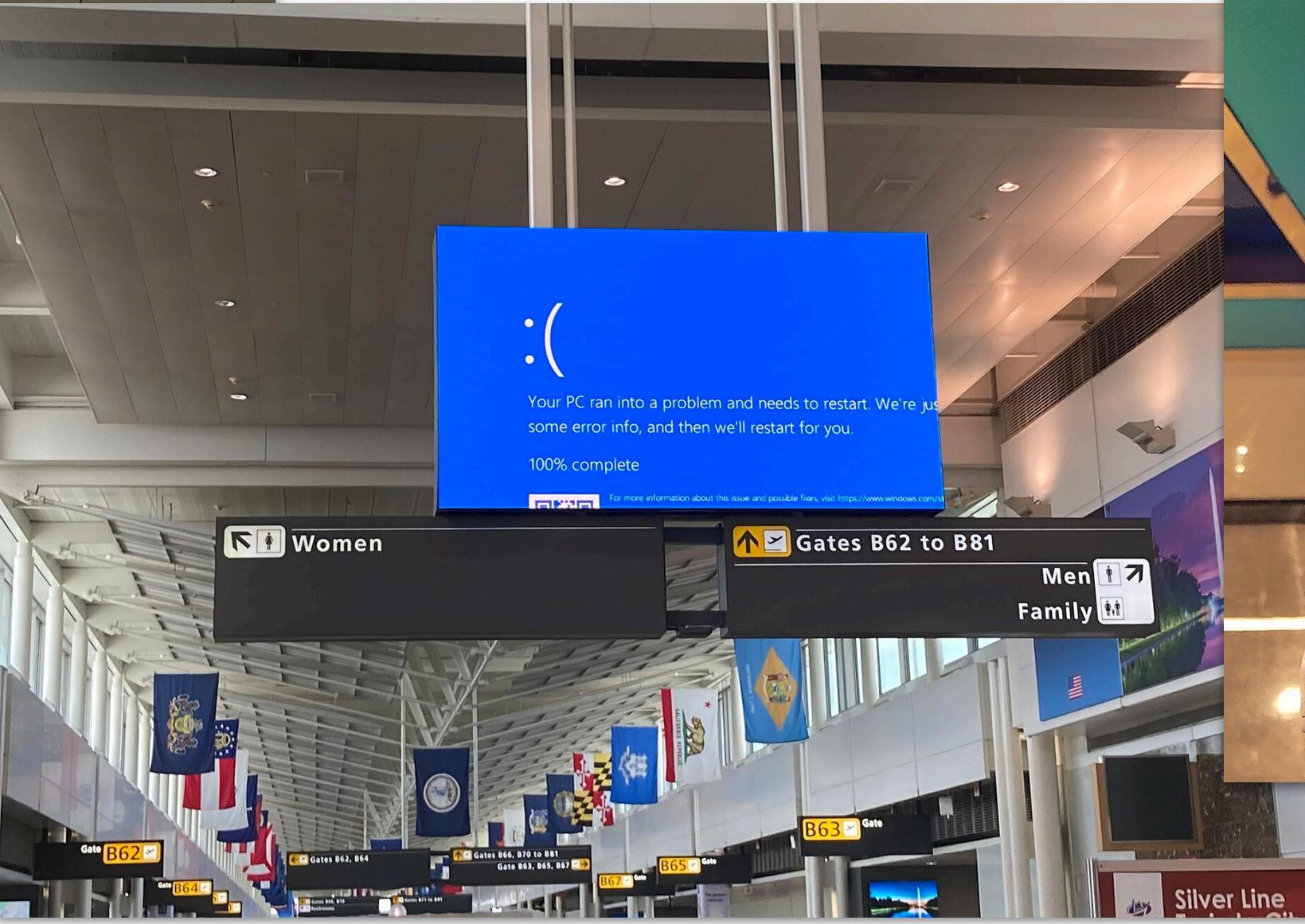
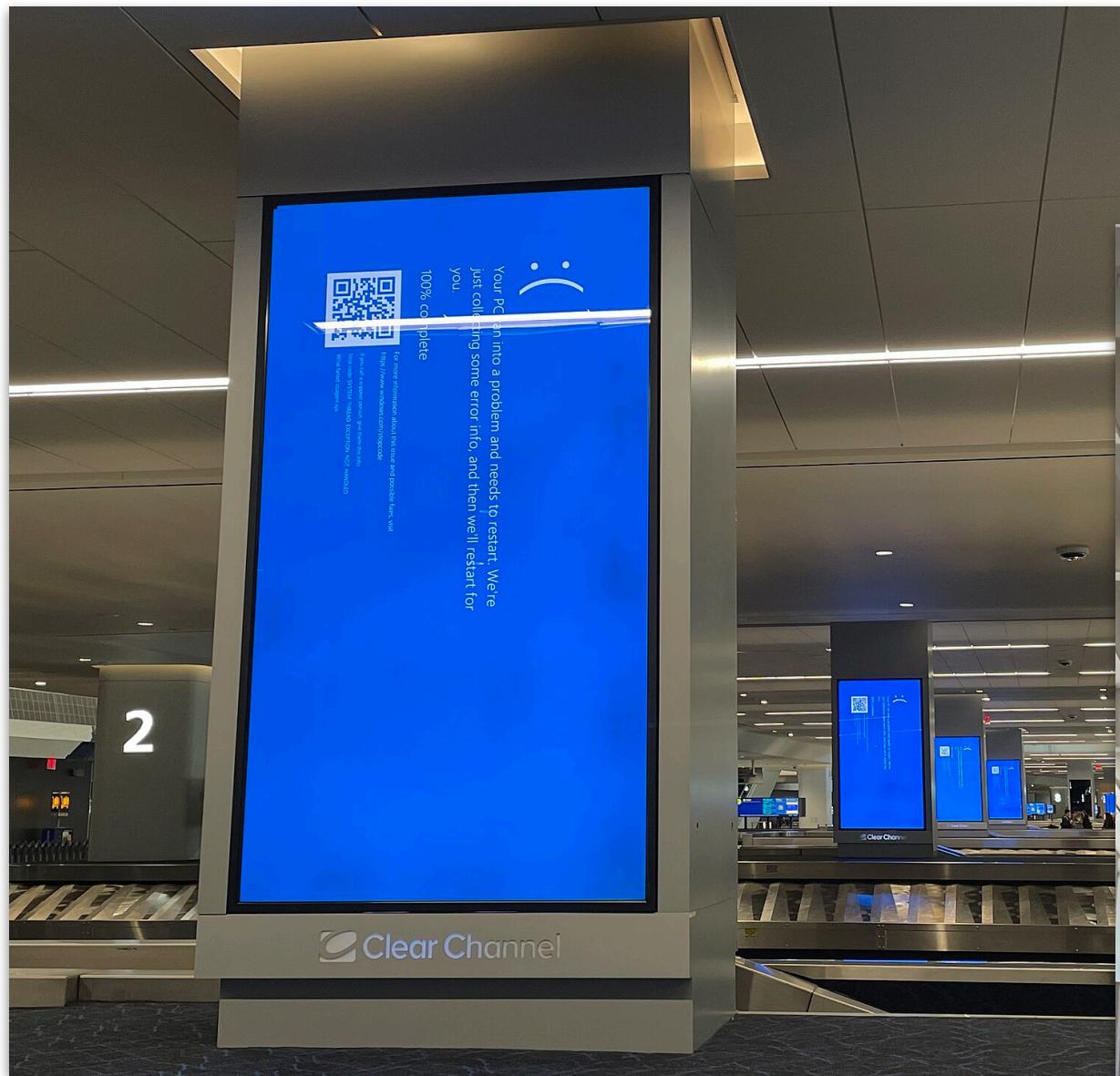
2

*I just like to program.
Why study the implementation?*



Does anyone remember what was noteworthy about July 19, 2024?

...was anyone trying to travel by plane around then?



What happened?

invalid memory access in C

unchecked array length

code running in OS kernel

limitations of processor multithreading

insufficient testing & validation

... all CS240 topics!

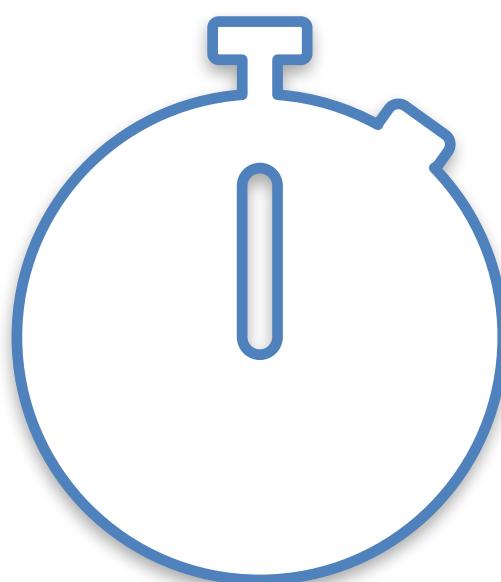
2

I just like to program. Why study the implementation?

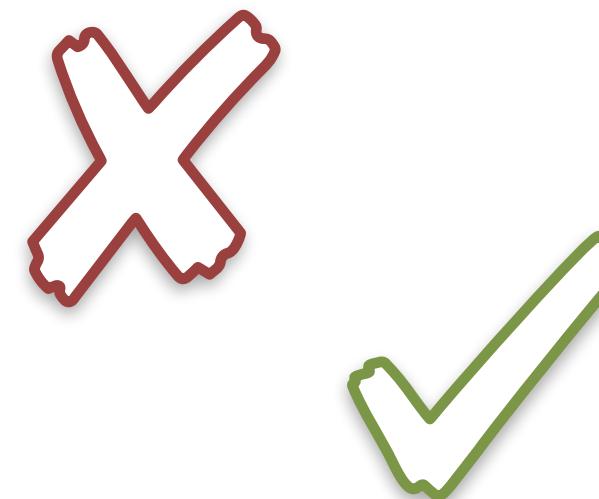
Most system abstractions "leak."

Implementation details affect your programs:

Their performance



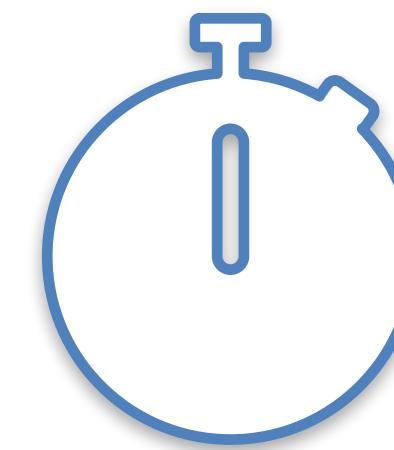
Their correctness



Their security



Performance



x / 973

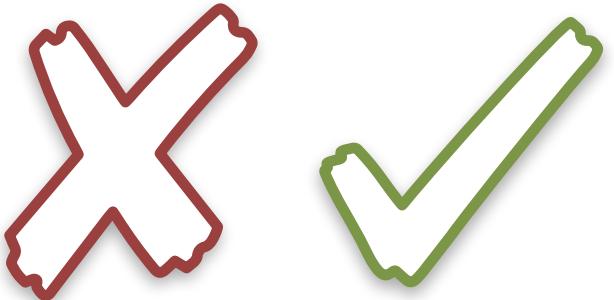
x / 1024

```
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];  
}
```

```
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];  
}
```

several times faster
due to hardware caches

Correctness



int ≠ integer
float ≠ real

Exploded due to **cast** of
64-bit floating-point number
to 16-bit signed number.
Overflow.



Boeing 787, 2015

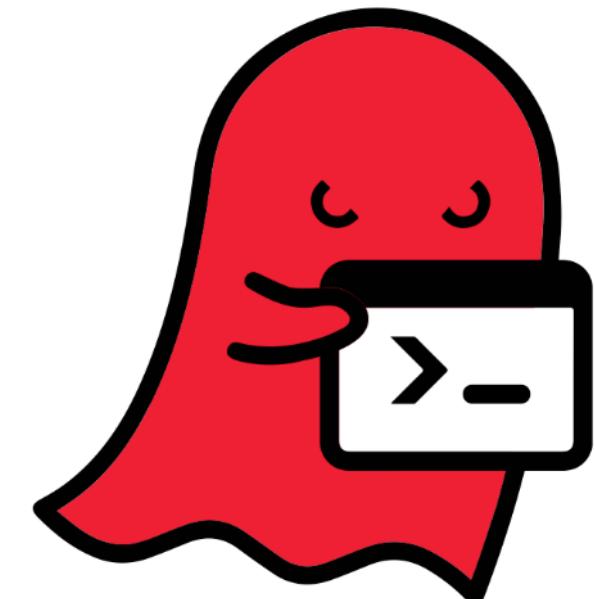


"... a **Model 787 airplane** ... can lose all
alternating current (AC) electrical power ...
caused by a **software counter** internal to the
GCUs that will **overflow** after **248 days** of
continuous power. We are issuing this AD to
prevent loss of all AC electrical power, which
could result in **loss of control of the airplane.**"
--FAA, April 2015

Security



The [GHOST vulnerability](#) is a buffer overflow condition that can be easily exploited locally and remotely, which makes it extremely dangerous. This vulnerability is named after the [GetHOSTbyname](#) function involved in the exploit.



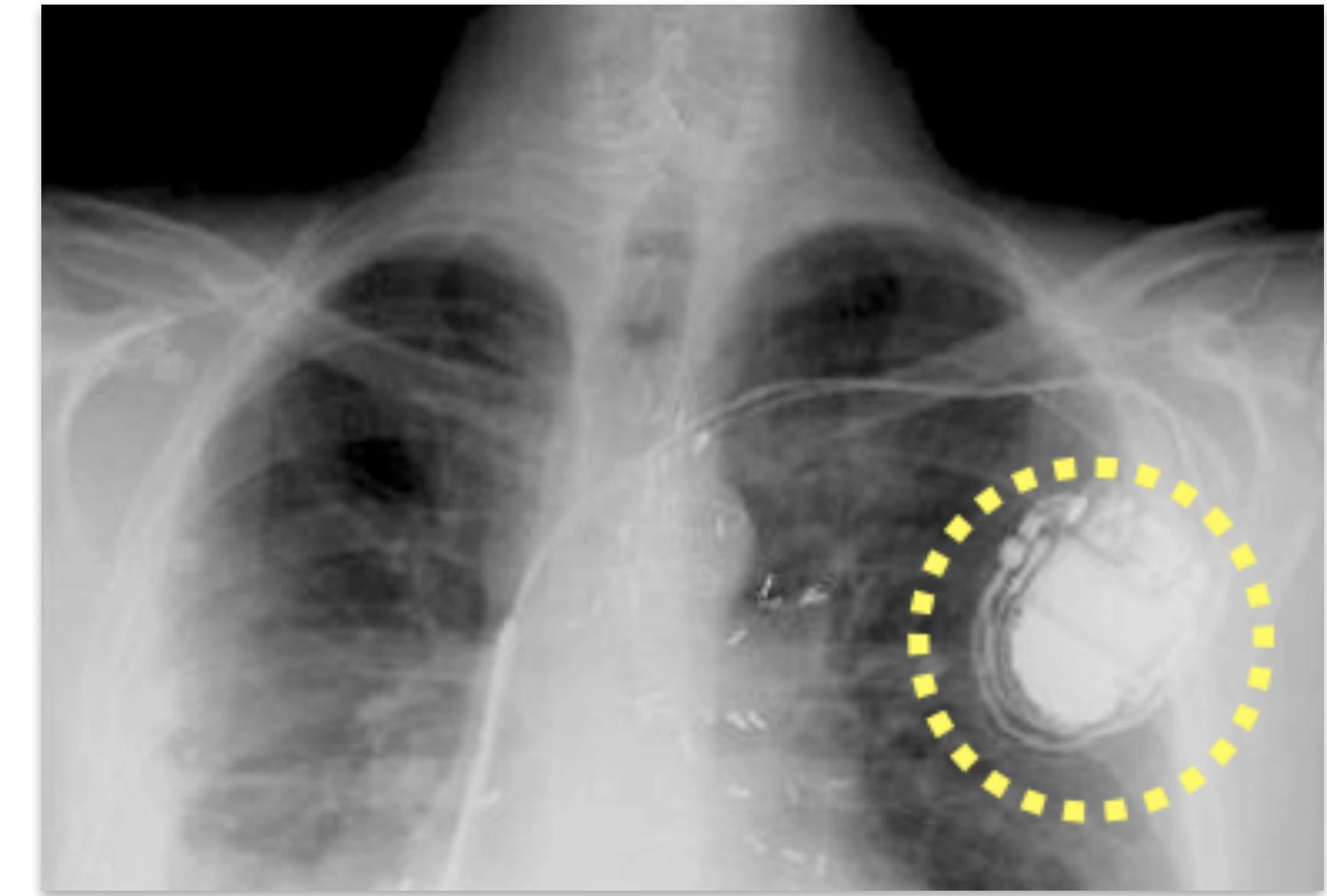
Cyber-Safe

All computers are flawed -- and the fix will take years

by Selena Larson [@selenalarson](#)

January 26, 2018: 12:07 PM ET

Meltdown and Spectre



[HOME PAGE](#) [MY TIMES](#) [TODAY'S PAPER](#) [VIDEO](#) [MOST POPULAR](#) [TIMES TOPICS](#)

The New York Times

Business

[WORLD](#) [U.S.](#) [N.Y. / REGION](#) [BUSINESS](#) [TECHNOLOGY](#) [SCIENCE](#) [HEALTH](#) [SPORTS](#) [OPINION](#)

[MEDIA & ADVERTISING](#) [WORLD BUSINESS](#) [SMALL BUSINESS](#) [YOUR MONEY](#) [DEALBOOK](#) [MARKETS](#) [RE](#)



indexology®
unmatched innovation

A Heart Device Is Found Vulnerable to Hacker Attacks

By BARNABY J. FEDER
Published: March 12, 2008

To the long list of objects vulnerable to attack by computer hackers, add the human heart.

The threat seems largely theoretical. But a team of computer security researchers plans to report Wednesday that it had been able to gain wireless access to a combination heart defibrillator and pacemaker.

[TWITTER](#)
[LINKEDIN](#)
[SIGN IN TO E-MAIL OR SAVE THIS](#)
[PRINT](#)
[REPRINTS](#)

Why take CS 240?

Learn **how** computers execute programs.

Deepen your appreciation of **abstraction**.

Learn enduring **system design principles**.

Improve your **critical thinking** skills.

Become a **better programmer**:

Think rigorously about execution models.

Identify limits and impacts of abstractions and representations.

Learn to use software development tools.

Foundations for:

Compilers, security, computer architecture, operating systems, ...

Have fun and feel accomplished!



<https://cs.wellesley.edu/~cs240/>

3

Long but *necessary*!



The Plan

Program, Application

Programming Language

Compiler/Interpreter

Operating System

Instruction Set Architecture

Microarchitecture

Digital Logic

Devices (transistors, etc.)

Solid-State Physics

Welcome to

CS 240: Foundations of Computer Systems!

What's a topic you are excited to learn more about in CS240?

0

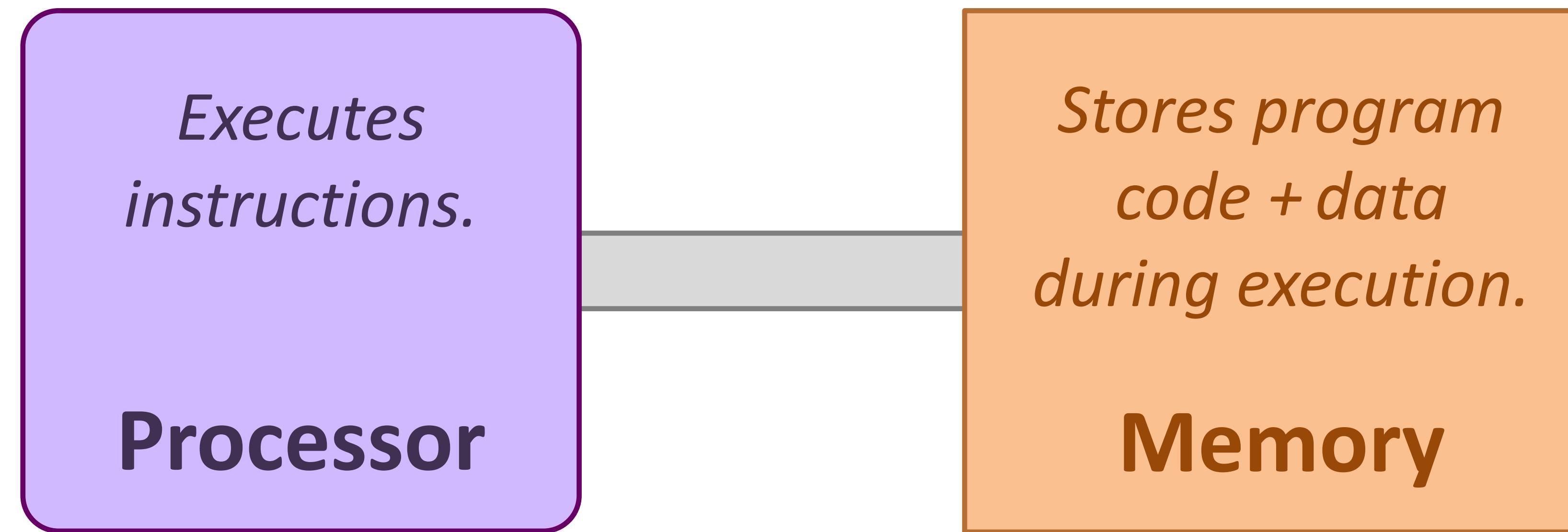
Nobody has responded yet.

Hang tight! Responses are coming in.

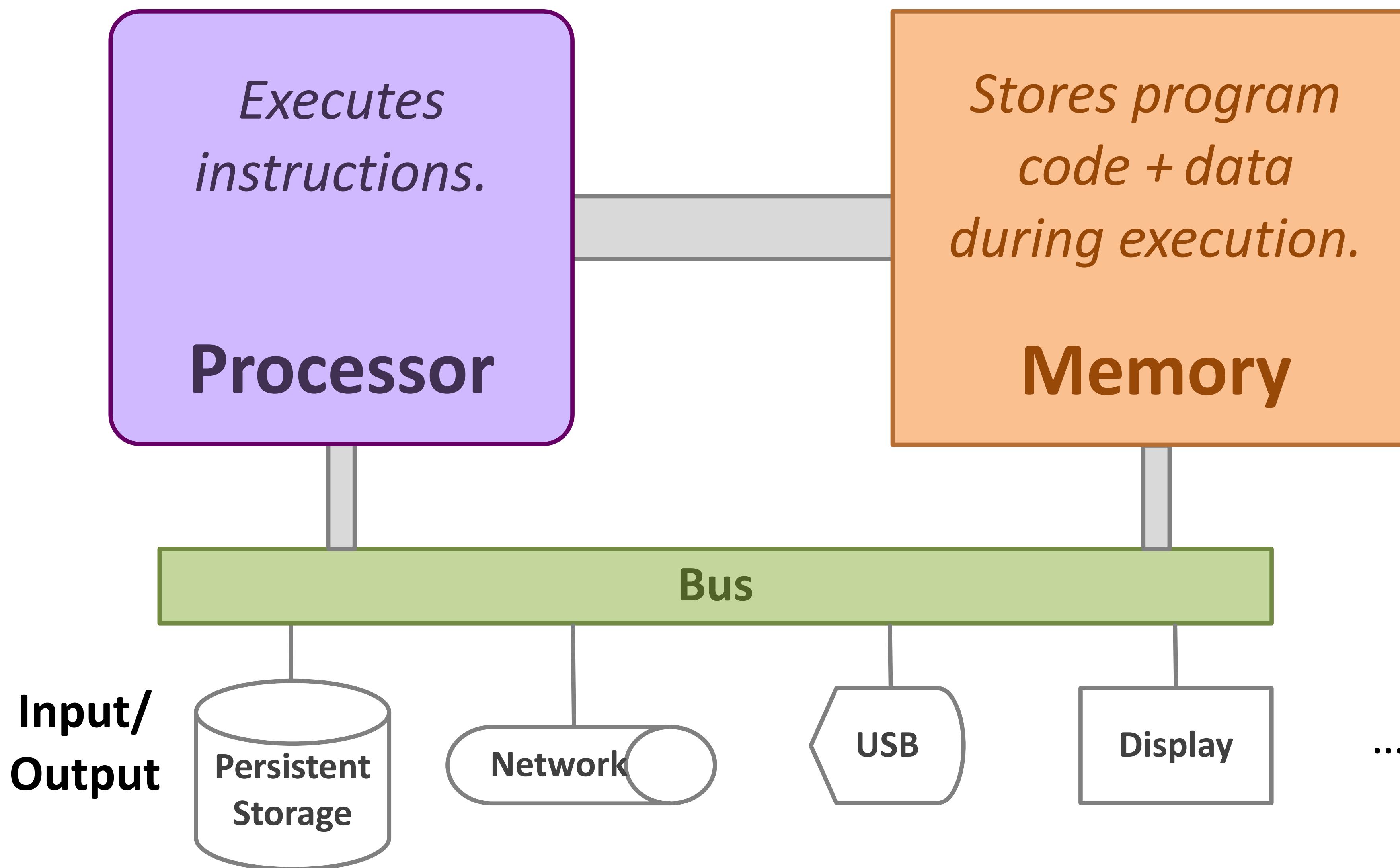
Today

- 1 What is CS 240?
- 2 How does CS 240 work?
- 3 Foundations of computer hardware

Modern Computer Organization



Modern Computer Organization

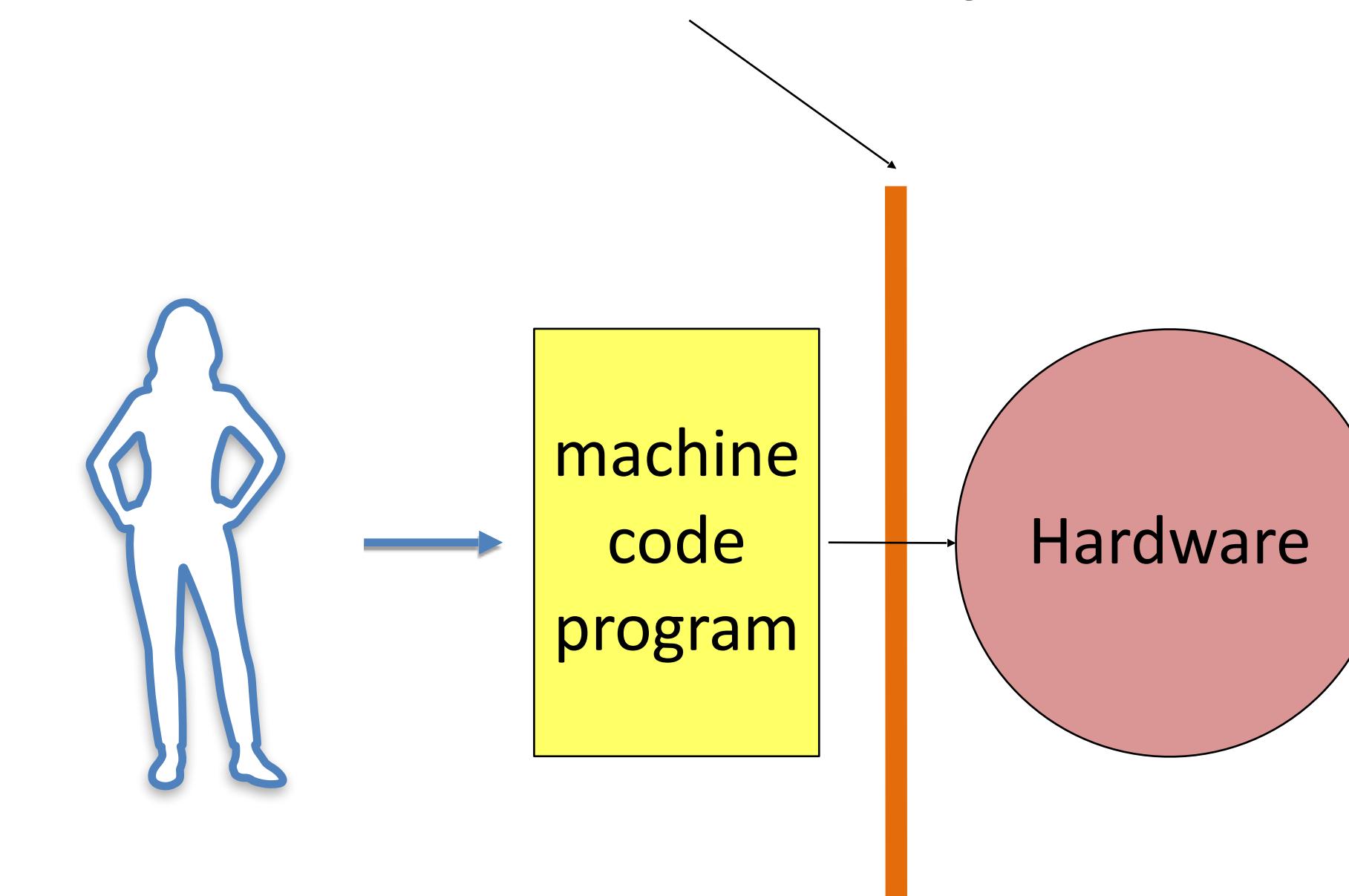


Machine Instructions

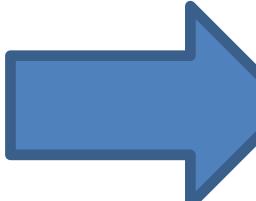
(adds two values and stores the result)

00000010100010101100100000010000

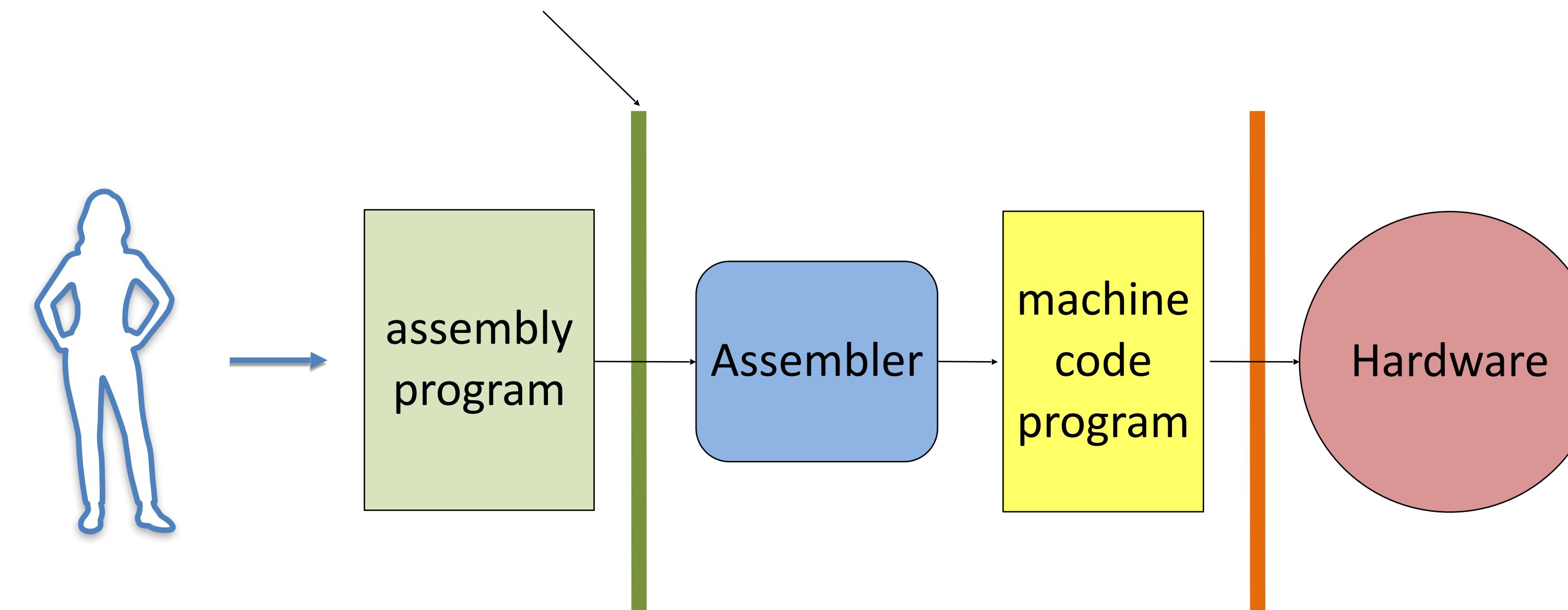
Instruction Set Architecture specification



Assemblers and Assembly Languages

addl %eax, %ecx  00000010100010101100100000010000

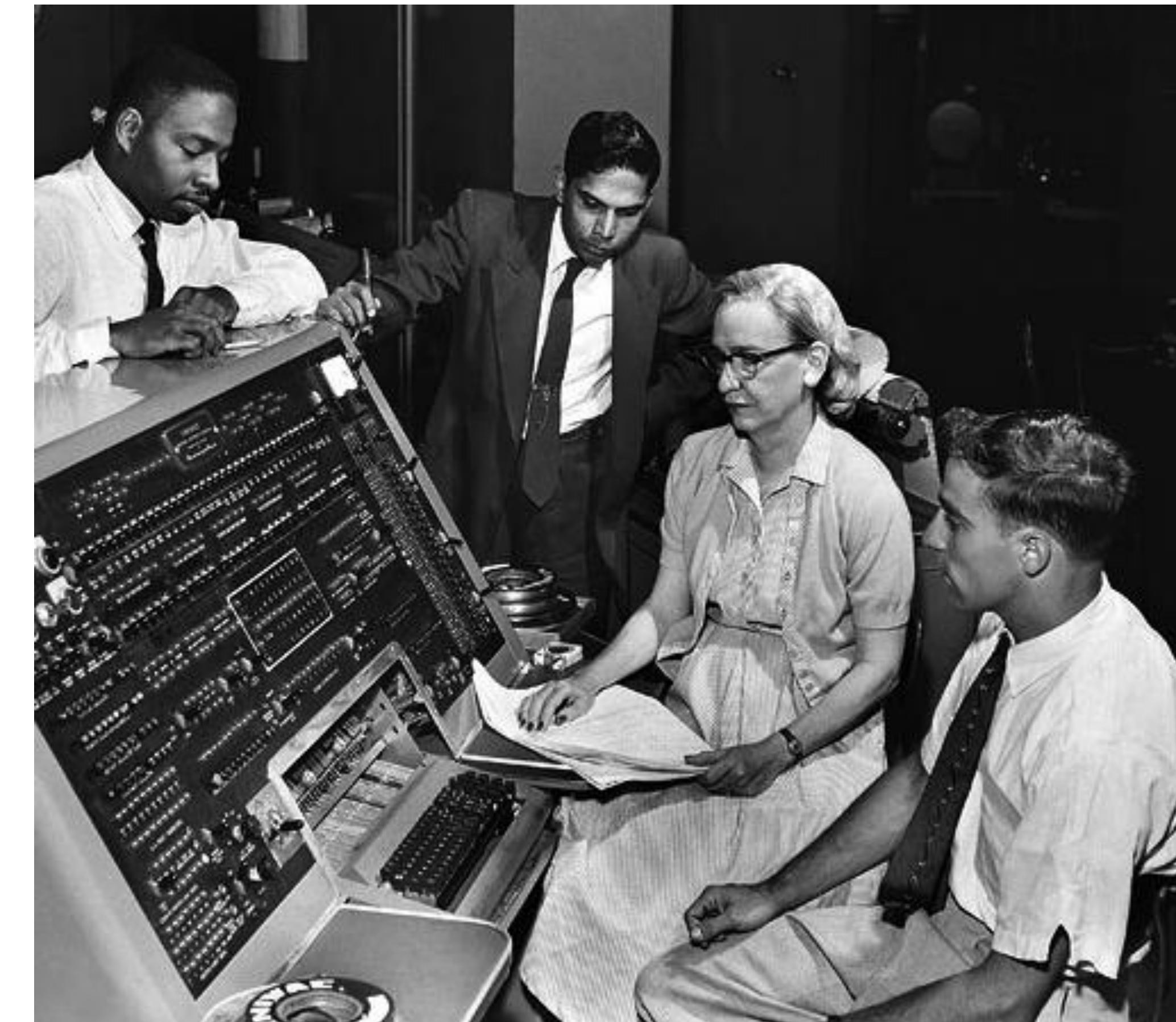
Assembly Language specification



A-0: first compiler, by Grace Hopper

Early 1950s

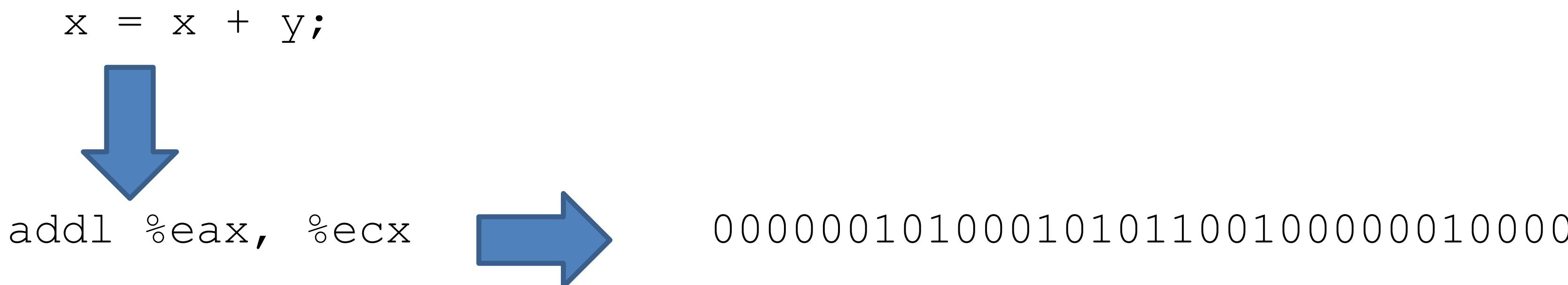
Maybe closer to assembler/linker/loader



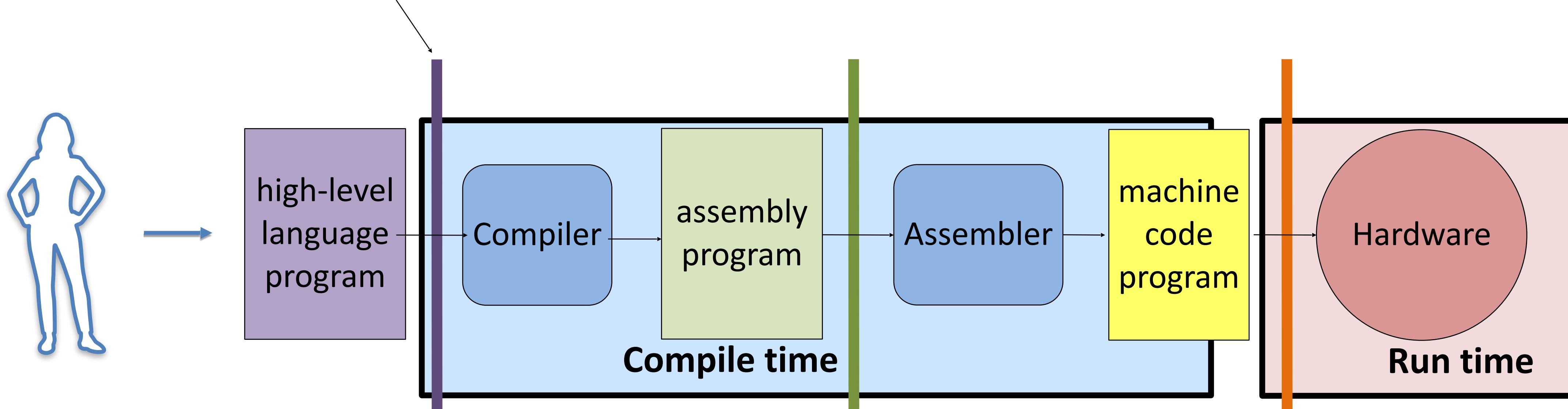
Jean Sammet also involved

- headed first sci comp group at Sperry in the '50s
- Later first female president of ACM
- Mount Holyoke alum, class of 1948

Higher-Level Programming Languages



Programming Language specification





<https://cs.wellesley.edu/~cs240/>

3

Long but *necessary*!