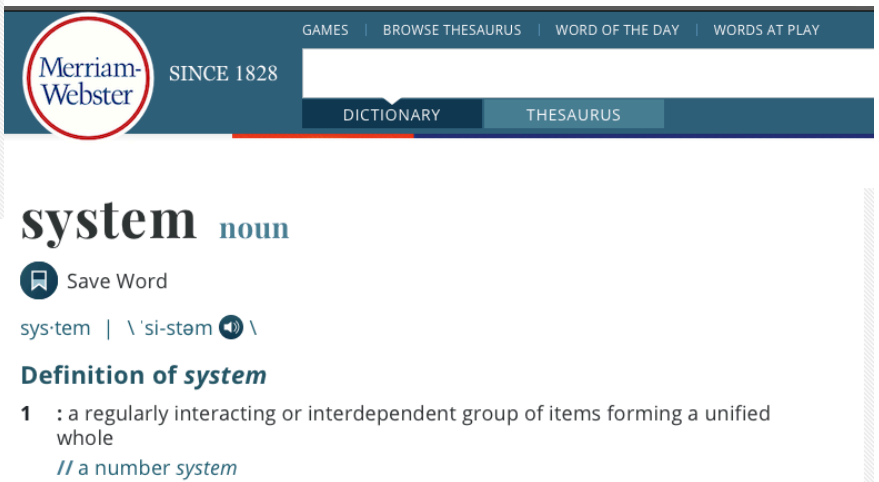
CS 210: Computer Systems

Course Introduction
5 September 2023

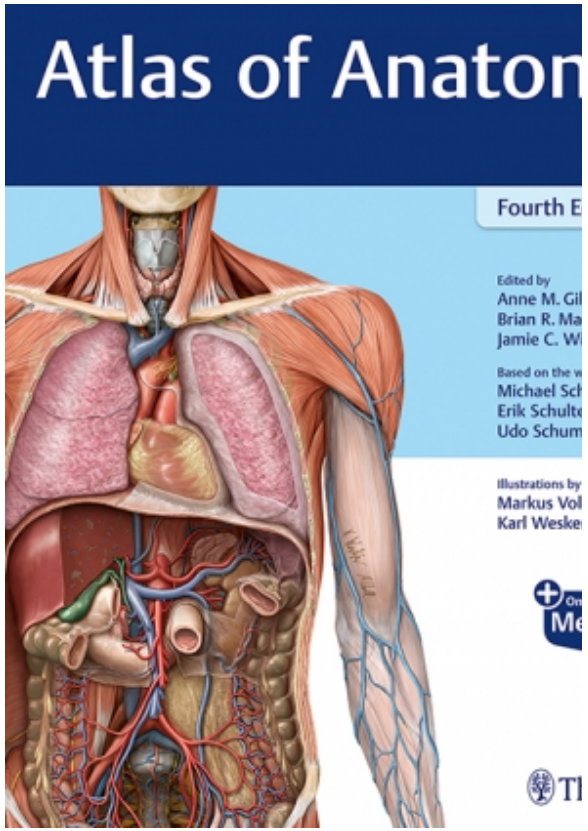
Vasiliki Kalavri, Assistant Professor

What is this course about?



Understanding the internals

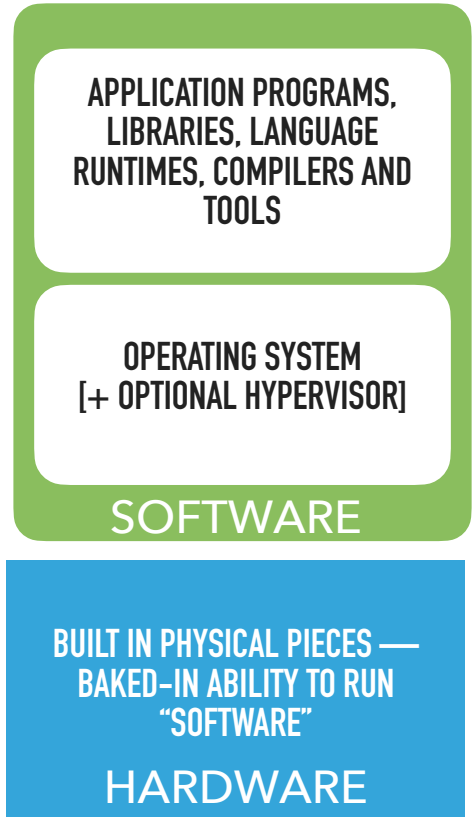
- What is a program and how it is executed
- How software interacts with the hardware
- What lies beneath the software abstractions
- How to be a “power”-programmer
  - inspect memory and internal processor state
  - detect, dissect, and fix bugs
  - write efficient code and fix performance issues



Computer System = Hardware + Software

- Hardware: the physical components
- Software: Layers of programs and operating information

The implementations may change but the underlying concepts do not
All computer systems have similar hardware
and software components that perform similar functions



Computer System = Hardware + Software

Client Computers

Server Computers

APPLICATION PROGRAMS, LIBRARIES, LANGUAGE RUNTIMES, COMPILERS AND TOOLS

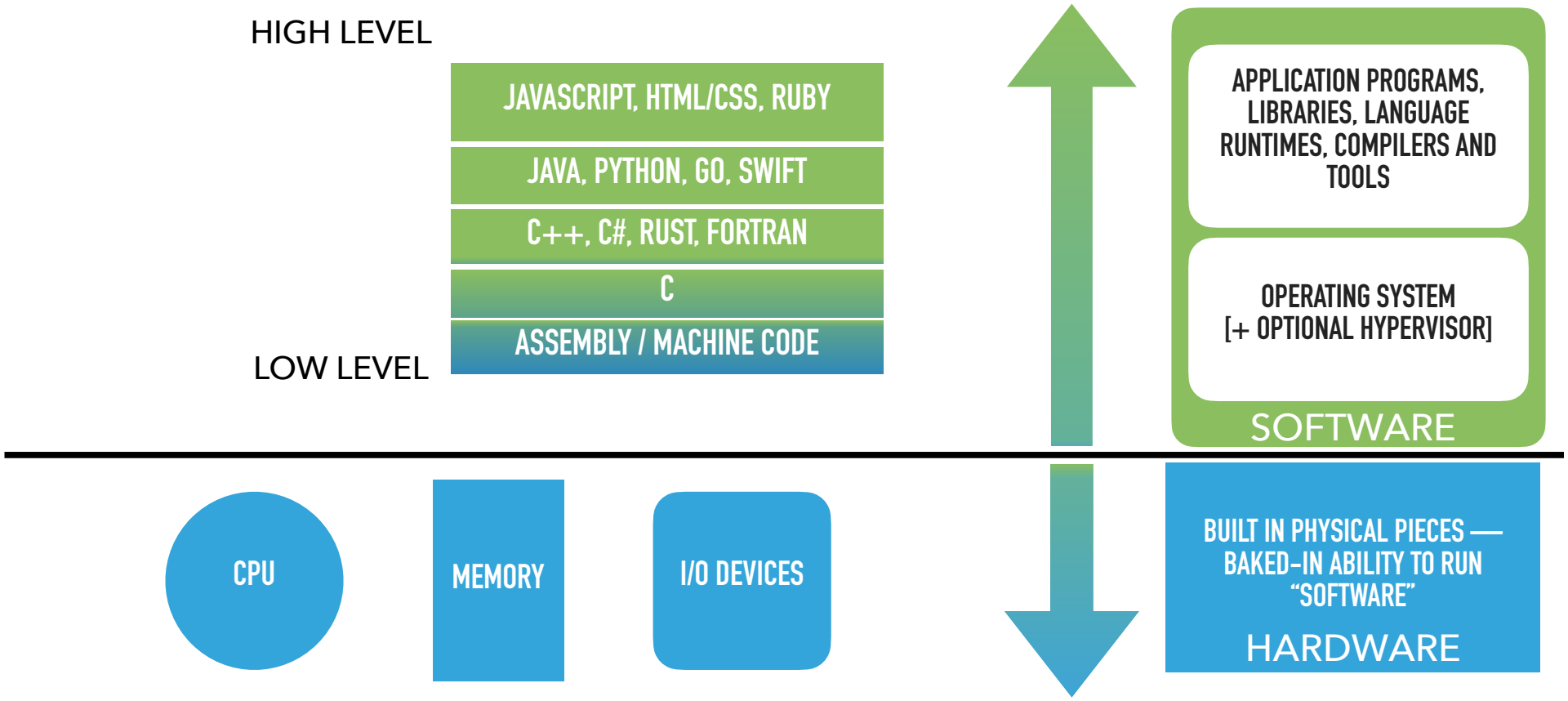
OPERATING SYSTEM [+ OPTIONAL HYPERVISOR]

SOFTWARE

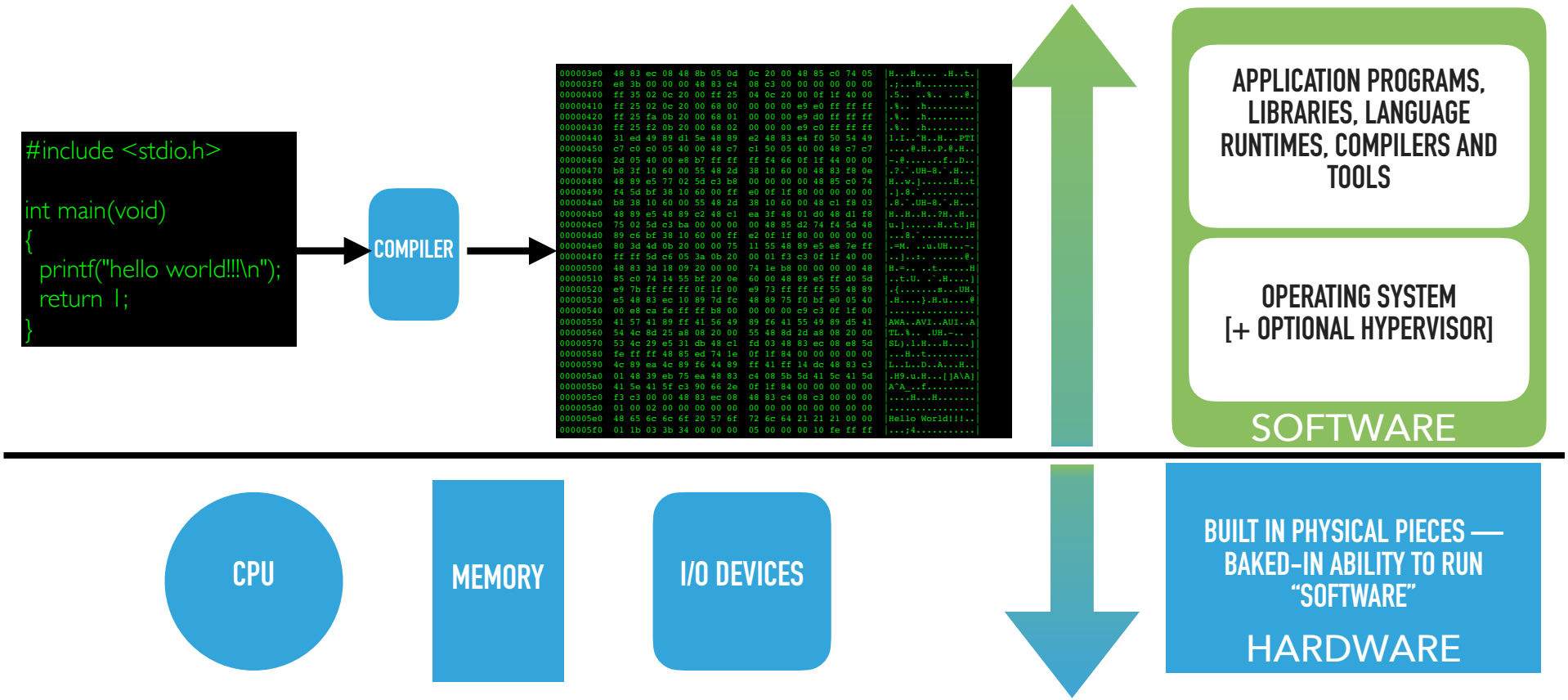
BUILT IN PHYSICAL PIECES — BAKED-IN ABILITY TO RUN "SOFTWARE"

HARDWARE

A programmer's view



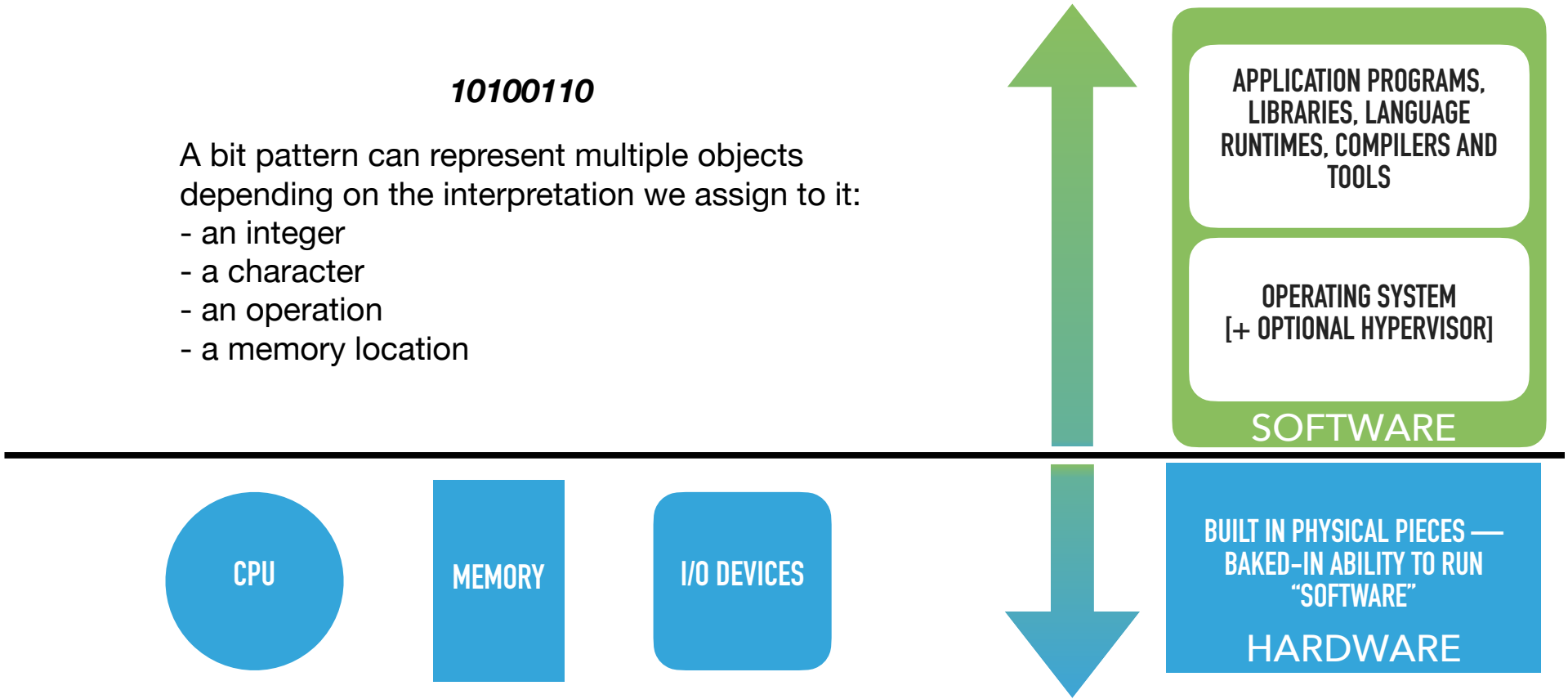
Programs are translated by other programs into different forms



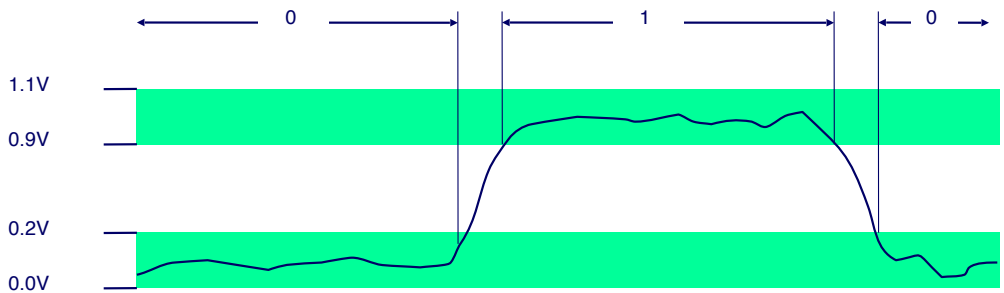
Information is bits + context

10100110

A bit pattern can represent multiple objects depending on the interpretation we assign to it:
- an integer
- a character
- an operation
- a memory location



Everything is bits and bytes



10100110 ON-OFF-ON-OFF-OFF-ON-ON-OFF

APPLICATION PROGRAMS, LIBRARIES, LANGUAGE RUNTIMES, COMPILERS AND TOOLS

OPERATING SYSTEM [+ OPTIONAL HYPERVISOR]

SOFTWARE

CPU

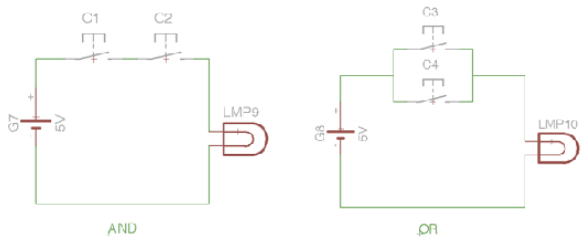
MEMORY

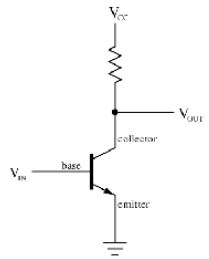
I/O DEVICES

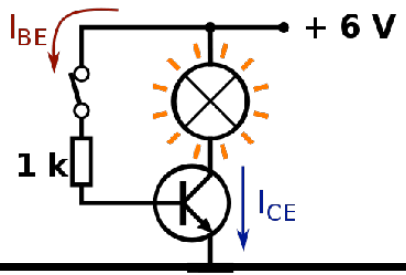
BUILT IN PHYSICAL PIECES — BAKED-IN ABILITY TO RUN "SOFTWARE"

HARDWARE

Electricity and transistors







APPLICATION PROGRAMS, LIBRARIES, LANGUAGE RUNTIMES, COMPILERS AND TOOLS

OPERATING SYSTEM [+ OPTIONAL HYPERVISOR]

SOFTWARE

CPU

MEMORY

I/O DEVICES

BUILT IN PHYSICAL PIECES — BAKED-IN ABILITY TO RUN "SOFTWARE"

HARDWARE

How did we get here?

"THOSE WHO CAN IMAGINE ANYTHING, CAN CREATE THE IMPOSSIBLE."
— ALAN TURING

BBC News Sport Reel Worklife Travel Future Culture Music More Search

Teach Home Primary Secondary Skillswise School Radio Bring the Noise Teache

Alan Turing: Creator of modern computing

BBC Teach > Secondary resources > KS3 Computing > GCSE Computing

Who was Alan Turing?

Alan Turing was not a well known figure during his lifetime.

But today he is famous for being an eccentric yet passionate British mathematician, who conceived modern computing and played a crucial part in the Allied victory over Nazi Germany in WW2.

He was also a victim of mid-20th Century attitudes to homosexuality – he was chemically castrated before dying at the age of 41.

1926

Discouraged at school

Alan Turing spent much of his early life separated from his parents, as his father worked in the British administration of India.

At 13 years old, he was sent to Sherborne School, a large boarding school in Dorset. The rigid education system gave his free-ranging scientific mind little encouragement, so Turing studied advanced modern scientific ideas, such as relativity, on his own, running far ahead of the school syllabus.

Alan Turing, aged 15, at Westcott House, Sherborne School.

1930

Devastated but inspired by his friend's death

The situation changed when Alan Turing became intensely attracted to another able pupil, Christopher Morcom.

He was inspired to communicate more and also to become an academic success. But Christopher died suddenly from tuberculosis. Devastated, Turing wanted to believe that Christop… emational turmoil involved a scientific… mind and brain that underlay his later…

Andrew Hodges talks abo… Morcom's de… n (BBC Two).

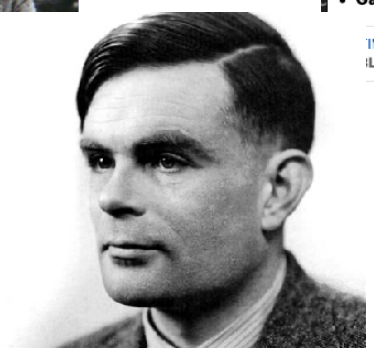
1936

Founder of modern computing

In 1936, Turing published a paper that is now recognised as the foundation of computer science.

Turing analysed what it meant for a human to follow a definite method or procedure to perform a task. For this purpose, he invented the idea of a 'Universal Machine' that could decode and perform any set of instructions. Ten years later he would turn this revolutionary idea into a practical plan for an electronic computer, capable of running any program.

Jim Al-Khalili explains how Turing invented the idea of feeding one machine different instructions. Clip from Order and Disorder (BBC Four).



Queen gives royal pardon to Alan Turing, the WWII codebreaking hero who was convicted for being gay then sterilised by the state

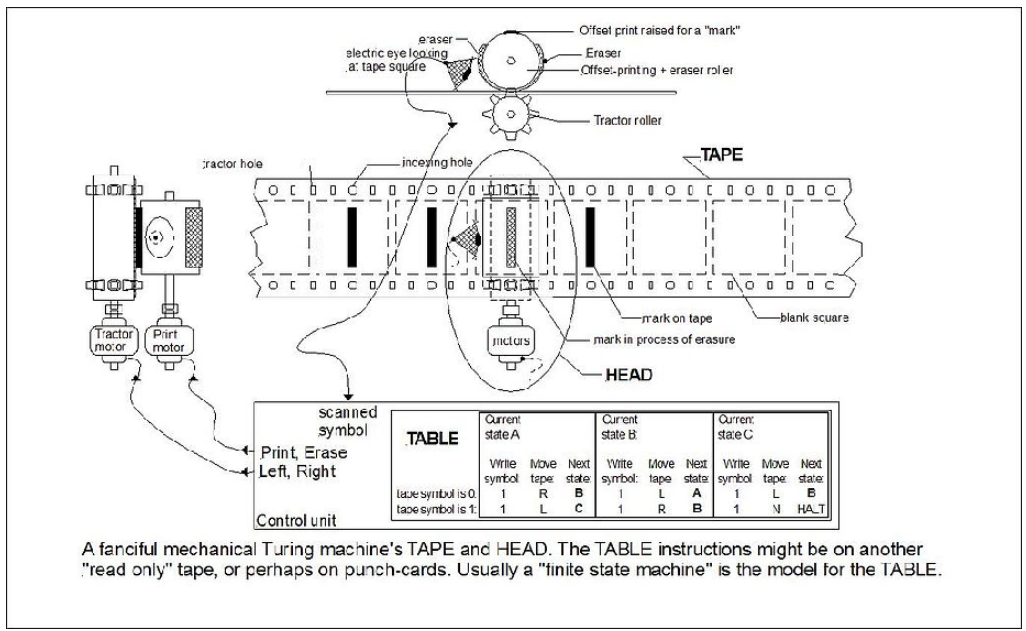
- **Alan Turing led the way in cracking the Enigma codes at Bletchley Park**
- **But he took his own life after 1952 conviction led to his chemical castration**
- **It is only the fourth Royal pardon since the end of the Second World War**
- **Gay rights campaigner Peter Tatchell said 50,000 more apologies are due**

'M SHIPMAN
LISHED: 19:25 EST, 23 December 20

https://www.bbc.co.uk/teach/alan-turing-creator-of-modern-computing/zhwp7nb

A **Turing machine** is a mathematical model of computation that defines an abstract machine,[1] which manipulates symbols on a strip of tape according to a table of rules.[2] Despite the model's simplicity, given any computer algorithm, a Turing machine capable of simulating that algorithm's logic can be constructed.[3]

Wikipedia contributors. "Turing machine." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 8 Jan. 2020. Web. 20 Jan. 2020.



A fanciful mechanical Turing machine's TAPE and HEAD. The TABLE instructions might be on another "read only" tape, or perhaps on punch-cards. Usually a "finite state machine" is the model for the TABLE.

It is indeed supremely difficult to effectively refute the claim that John von Neumann is likely the most intelligent person who has ever lived. By the time of his death in 1957 at the modest age of 53, the Hungarian polymath had not only revolutionized several subfields of mathematics and physics but also made foundational contributions to pure economics and statistics and taken key parts in the invention of the atomic bomb, nuclear energy and digital computing.

Known now as *"the last representative of the great mathematicians"*, von Neumann's genius was legendary even in his own lifetime. The sheer breadth of stories and anecdotes about his brilliance, from Nobel Prize-winning physicists to world-class mathematicians abound:

...as one famously ... the enjoyments of life. Twice married and wealthy, he loved expensive clothes, hard liquor, fast cars and dirty jokes, according to his friend Stanisław Ulam. Almost involuntarily, his posthumous biographer Norman Macrae recounts, people took a liking to von Neumann, even those who disagreed with his conservative politics (Regis, 1992).

Personality

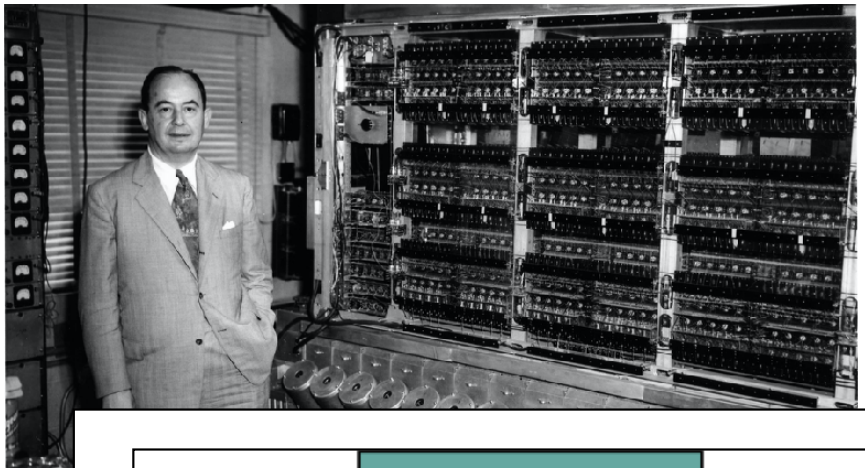
Despite his many appointments, responsibilities and copious research output, von Neumann lived a rather unusual lifestyle for a mathematician. As described by Vonnauman and Halmos:

> *"Parties and nightlife held a special appeal for von Neumann. While teaching in Germany, von Neumann had been a denizen of the Cabaret-era Berlin nightlife circuit."* — Vonneuman (1987)

> *The parties at the von Neumann's house were frequent, and famous, and long.* — Halmos (1958)

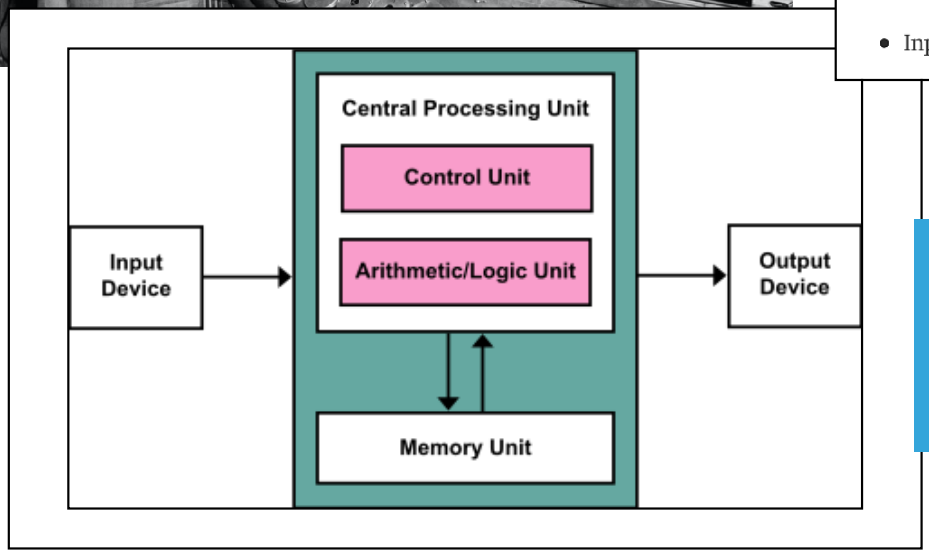
Although influential in his own right, throughout his life, von Neumann made sure to acknowledge that the central concept of the modern computer was indeed Turing's 1936 paper *On Computable Numbers, with an Application to the Entscheidungsproblem* (Fraenkel, 1972)

> "von Neumann firmly emphasised to me, and to others I am sure, that the fundamental conception is owing to Turing — insofar as not anticipated by Babbage, Lovelace and others." — Stanley Fraenkel (1972)



In 1945, von Neumann proposed a description for a computer architecture now known as the von Neumann architecture, which includes the basics of a modern electronic digital computer including:

- A processing unit that contains an arithmetic logic unit and processor registers;
- A control unit that contains an instruction register and a program counter
- A memory unit that can store data and instructions;
- External storage; and
- Input and output mechanisms;



THE STORED PROGRAM COMPUTER WAS REVOLUTIONARY AND HAS FUELED ALL THE DIGITAL TECHNOLOGY THAT HAS TOUCHED OUR LIVES

Grace Brewster Murray Hopper

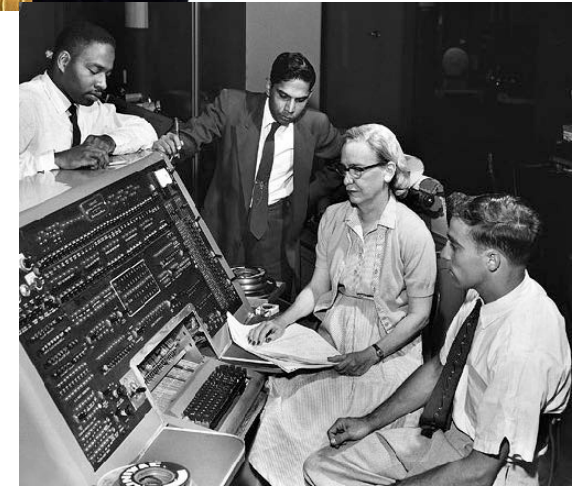
When Hopper recommended the development of a new programming language that would use entirely English words, she "was told very quickly that [she] couldn't do this because computers didn't understand English." Still, she persisted. "It's much easier for most people to write an English statement than it is to use symbols," she explained. "So I decided data processors ought to be able to write their programs in English, and the computers would translate them into machine code."[20]

Her idea was not accepted for three years. In the meantime, she published her first paper on the subject, compilers, in 1952. In the early 1950s, the company was taken over by the Remington Rand corporation, and it was while she was working for them that her original compiler work was done. The program was known as the A compiler and its first version was A-0.[21]:11

In 1952, she had an operational link-loader, which at the time was referred to as a compiler. She later said that "Nobody believed that," and that she "had a running compiler and nobody would touch it. They told me computers could only do arithmetic."[22]

https://en.wikipedia.org/wiki/Grace_Hopper#UNIVAC

> Humans are allergic to change. They love to say, "We've always done it this way." I try to fight that. That's why I have a clock on my wall that runs counter-clockwise.



WE OWE A GREAT DEAL TO THE ADMIRAL. NOT ONLY COULD SHE SEE WHAT WAS POSSIBLE BUT SHE MAKE IT REALITY — THE DETAILS MATTER

https://marhicks.com/syllabi.html

History of Computing

Illinois Institute of Technology: Fall 2011

Why do we think that we've witnessed a computing "revolution" in the 20th and 21st centuries? This course contextualizes Silicon Valley's current obsession with change-for-its-own-sake by showing how the fiction of disruption has a long and well-established history. From World War II codebreaking, to intra-national spying initiatives, this course asks students to look behind industry hype and explore the interconnectedness of computer technologies with the aims and goals of strongly centralized technocratic governments.

Women in Computing History (Current Syllabus)

First taught at Illinois Institute of Technology: Fall 2016 (Old Syllabus); offered again at the University of Wisconsin-Madison in Fall 2017

Perhaps the first of its kind in the U.S., this computer history course explicitly looks at computing's past through the experiences of women who worked in computing at all levels—from data input to programming to hardware design. It strives to be intersectional in its analysis, showing how gender is but one window into the historiography of computing and how it must be taken together with an analysis of class, race, sexuality, ability, and many other categories in order for us to truly understand how computing structures lives and whole economies. Students did projects in this class designed to engage the public on this still largely hidden history: In addition to a wiki-storming exercise, they also created public history projects, like video games, podcasts, and comic strips. See the news coverage of their projects in Chicago Inno or this post on their projects on the Digital History Lab.

Who are we?

We are scientists, researchers, professionals

Exploring the future of computer systems

- Designing new architectures, operating systems, data analytics systems
- Developing novel methods to improve the efficiency of existing systems

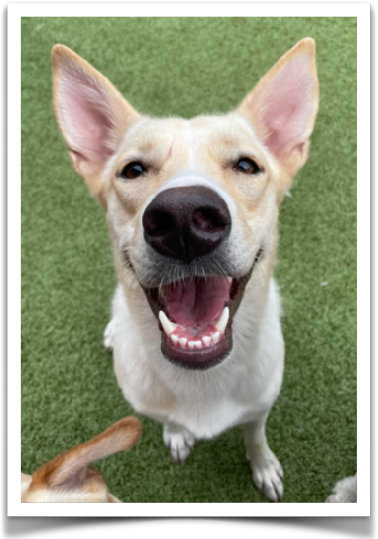
Applying our knowledge and expertise in the real world

Training the new generation of Computer Scientists

James Cadden, PhD (Jim)

About Me…

- PhD in CS from BU, 2019
  - Advised by Appavoo & Krieger
- IBM Research, Cambridge, 2020
  - Sr. Research Scientist
- Native of MA, Reading→Chelsea
  - Dog Owner, rescue
  - Climbing/Hiking
  - Special Talent? Rock Music Trivia!



Preethi Narayanan

- Lecturer
- MS in Computer Science, Georgia Tech
- Specializations:
  - Social Computing
  - Systems & Architecture
  - Modeling & Simulation
- Other things I do:
  - Sketching
  - Baking
  - Playing TOTK and Hades for too many hours

Vasiliki (Vasia) Kalavri

- Assistant Professor, co-leading the CASP Lab: https://sites.bu.edu/casp/
- My research focuses on designing efficient and scalable data analytics systems (graph processing, data streaming, machine learning, privacy-preserving analytics).
- Grew up and studied undergrad in Greece, went to Spain, Sweden, Germany, Belgium for graduate studies and PhD, Switzerland for post-doc.
- At BU since Jan.'20, teaching CS 210 and CS 551.

Course Info & Logistics

Course Goals

- Understand components of computer systems
- Look under the hood
- Understand software / hardware interaction
- Learn the principles and mechanisms that underlie all computer systems
- Use “real-world” examples to illustrate concepts

Course content: UNIX

- PART I (Lectures 2-6)
  - How software and the runtime are organized into applications and the Operating System
  - Tools of the trade:
    - ASCII Terminal Interface: The Shell
    - Editors, Make, Git

Course content: The von Neumann Architecture

- PART II (Lectures 7-16)
  - Binary representation and software anatomy
  - Executables
  - Assembly code and programming
  - Linking and Virtual Address Spaces
  - Cache Memories

Course content: C

- PART III (Lectures 17-24)
  - Bridging the human world with the binary worlds of von Neumann computers
  - Where and how high-level programming languages fit into things
  - Mapping C to Assembly
  - Dynamic memory allocation
  - Software optimizations

Why?

Classical Reasons

- You want to call yourself a “computer scientist”
- You want to build software people use (need correctness and performance)
- You need to make a purchasing decision or offer “expert” advice
- It’s Required ;-)

Other Reasons

- To be Masters of the Digital universe you must demystify the magic
- Computer Systems are just F’ing cool and a marvel and testament to human ingenuity
- Knowledge of how things work can unleash your creativity in new ways

Courseware

Website: https://cs-210-fall-2023.github.io/CS210-Website/

Syllabus: https://cs-210-fall-2023.github.io/CS210-Website/syllabus.pdf

- Course organization, policies, grading scheme
- Detailed lecture & discussion calendar
- Midterm dates: not flexible!

TopHat: For attendance & participation in lectures and discussions

Courseware (cont.)

Piazza: https://piazza.com/class/lkca521zyyb1wz

- Announcements, how-to guides, links to homework
- Ask questions, answer questions, get involved

Github Classroom:

- Create a github.com account if you don't have one
- Clone and push your work to assignment repositories

Gradescope: https://www.gradescope.com/courses/572191

- For assignment submission and grading

Readings

Two textbooks

- Under the Covers : The Secret Life of Software
  - https://cs-210-fall-2023.github.io/UndertheCovers/textbook/intro_tb.html
- K. N. King, “C Programming: A Modern Approach” (for Part III)

Lecture Notes

- https://cs-210-fall-2023.github.io/UndertheCovers/lecturenotes/intro_ln.html

Do the readings before the lecture. Come prepared with questions!

You personal online UNIX server

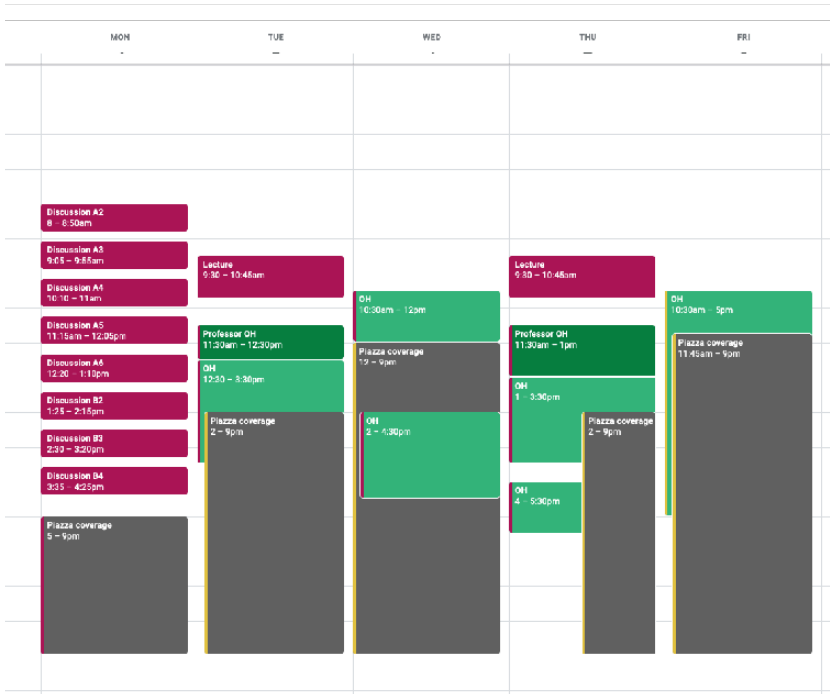
- Link to the environment.
- To start your server, follow the instructions in the book.
- Remember to commit and push your work
  - Your files will not be automatically persisted
  - Shut down your container when done working for the day
- Start your assignments early!

Discussions

- Discussions are a fundamental component of this course - a small setting where you can get the details sorted out
- Like in lectures, we will use Top Hat for questions to gauge participation and attendance

Attend the discussion you are registered for: We will record attendance based on the official roster!

Office Hours and Piazza coverage



- Google Calendar link provided in the Syllabus
- More than 25 hours of OH Tuesday-Friday
- Piazza coverage all weekdays

A note on Office Hours

- Office hours are not a replacement for attending lectures and discussions; Come with specific questions.

- It is not the responsibility of the TAs and CAs to give you an overview of the assignment. **Read the writeup BEFORE coming to OH**.

- It is not the responsibility of the TAs and CAs to debug your code or give you a crash-course on various tools. **Read the tutorials and use the tools BEFORE coming to OH**. Come to OH prepared to explain what you have tried so far and what it is you do not understand.

- Come to the Professor OH to clarify concepts, discuss logistics, ask advice. Professors will not help you debug your code or set up your environment.

How to reach us

Use **Piazza** for all communication with course staff

- Private post
- **Do not send email!**

General advice and heads-up

Becoming an expert at anything takes *a lot of practice*

- Probably the most time-consuming course you've taken so far.
- Need a solid foundation of topics covered in CS 111 and 112.

Superficial understanding won't cut it; you need to *understand the details*

- Read the material again and again until it "clicks".
- Draw diagrams.
- Follow the examples in the book and discussions multiple times.

This is your chance to explore your curiosity ... ask questions and “tinker”.

We love this stuff... make use of us, we are happy to help!

What's next

- PS0: Get set up on the infrastructure
  - PS0A: Read, sign, and submit the syllabus
  - PS0B: Get used to basic unix commands
- Will be released after lecture today
  - Keep an eye on Piazza for how-to posts