An Introduction to Computer Science

CS 8: Introduction to Computer Science, Winter 2018
Lecture #2

Ziad Matni
Dept. of Computer Science, UCSB

A Word About Registration for CS8

FOR THOSE OF YOU NOT YET REGISTERED:

- **If you are <u>not on the waitlist</u>, you will not get into this class**
- I will be going by the waitlist as I decide to let people in the class
- There *are a <u>few</u>* spots opening up – I will let you know between today and Wednesday about getting in

Administrative

- **You must register on Piazza**
  - **https://piazza.com/ucsb/winter2018/cs8**
  - You will not get my class announcements otherwise!
    - **I'm not using GauchoSpace**

- Remember: Lab0 is due on Wednesday!
  - Use the Turnin service as shown in lab on Tue.

- Class webpage: **https://ucsb-cs8-matni-w18.github.io**

Switching About In The Labs…

… is frowned upon ☹

- Please stick to the lab time that you have per your registration
  - The labs are pretty full and at capacity

IF YOU WANT TO SWITCH LAB SECTIONS, YOU MUST:

1. **Find a person in the other lab to switch with you**
2. Get the OK from *BOTH* T.A.s

What is this "Computer" you speak of?

<u>Let's define a "computer"</u>

- Computer (n.): a computing device

- A device **that can be instructed** to carry out an **arbitrary** set of **arithmetic or logical operations** automatically

Algorithms!

Computers = Computing Devices

Compute

(v) To make sense of ; to **calculate** or reckon

- What was the first computing tool ever?

Likely invented around when humans fell out of the trees…

Using **Abstraction** is Key to Using Computers (or any Complex Machine)

Abstraction: *(n)* A **mental model** that *removes complex details*

Do you need to know this?

To know how to do this?

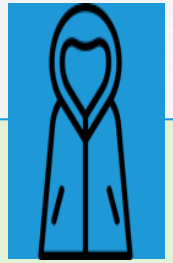
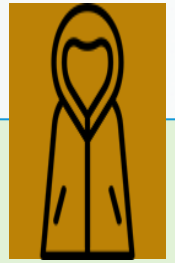
Images from jblearning.com

Algorithm

- A *step-by-step* logical procedure to *solve a problem*
  - Like a very precise recipe!
- Named after famed 9th-century Persian mathematician Al-Khawarizmi who put a name to the practice and published a lot on it

Examples of Everyday Use of Algorithms

- **Problem to Solve**: What coat, if any, should I wear today?
- **Algorithm**:


1. Measure the outdoor temperature, T.
2. If T < 62F then wear my blue coat.
   1. If blue coat is *dirty* (dirt level ≥ 7), wear my brown coat instead
   2. If it's also *raining* (Now raining = True), wear my black poncho instead
3. If T ≥ 62F then don't wear a coat
   1. Plan on buying ice-cream for lunch!

And Now, With More Detail…

1. Measure the outdoor temperature, T.
2. If T < 62F then wear my blue coat.
   1. If blue coat is *dirty* (dirt level ≥ 7), wear my brown coat instead
   2. If it's also *raining* (Now raining = True), wear my black poncho instead
3. If T ≥ 62F then don't wear a coat
   1. Plan on buying ice-cream for lunch!

a) Define outcomes:
1. wear blue coat,
2. wear brown coat,
3. wear black poncho,
4. wear nothing and get ice-cream for lunch!

b) Define conditions:
1. T < 62 or not
2. Dirt_Level < 7 or not
3. Now_Rain = True or not

b) Get measures/values for T, Dirt_Level, Now_Rain

c) If ((T < 62) AND (Dirt_Level < 7)) then (outcome = 1)
d) If ((T < 62) AND (Dirt_Level ≥ 7)) then (outcome = 2)
e) If ((T < 62) AND (Now_Rain = True)) then (outcome = 3)
Otherwise (outcome = 4)
f) The End

And Now, With "Language"…

1. Measure the outdoor temperature, T.
2. If T < 62F then wear my blue coat.
   1. If blue coat is *dirty* (dirt level ≥ 7), wear my brown coat instead
   2. If it's also *raining* (Now raining = True), wear my black poncho instead
3. If T ≥ 62F then don't wear a coat
   1. Plan on buying ice-cream for lunch!

```
Measure(T)
Get(Dirt_Level)
Assess(Now_Raining)

if (T < 62) AND (Dirt_Level < 7)
    then Outcome = 1
if (T < 62) AND (Dirt_Level >= 7)
    then Outcome = 2
if (T < 62) AND (Now_Raining = True)
    then Outcome = 3
else
    Outcome = 4

End Program
```

…that has specific form and syntax

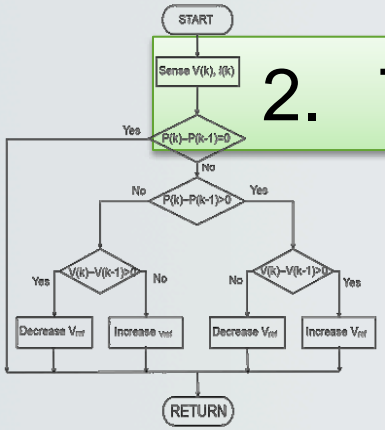
(like any "language" would!)

This is often called "pseudo-code" and is the pre-cursor to writing a program in a specific computer language

What is “Computer Science”?

The study of :

1. The designs and uses of computers as useful ***tools*** in our daily lives



2. The use of algorithms to solve problems

mostly around the creation, processing, interpreting, communication, etc… of information

Some Historical Background…

The First Modern Computing Devices
(As a Novelty or For Specific Commercial Purposes)

B. Pascal (1623 – 1662)

Blaise Pascal

Mechanical device that could
add, subtract, divide & multiply using gears

"Pascaline" : a calculating machine (1652)

J. Jacquard (1752 – 1834)

Joseph Jacquard

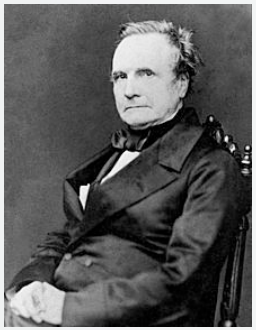
Jacquard's Loom,
used punched cards to describe patterns

Jacquard Loom (invented 1801)

Images from Wikimedia.org

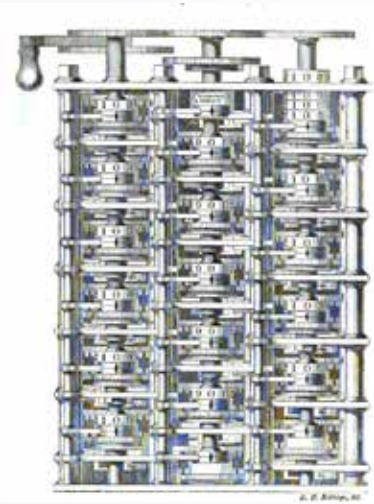
Computing Devices for General Purposes
(For Serious Math and Engineering Purposes)

- **Charles Babbage**
  - *Analytical Engine* could calculate polynomial functions and differentials
  - Calculated results, but also *stored intermediate findings* (i.e. precursor to computer memory)
  - "Father of Computer Engineering"
- **Ada Byron Lovelace**
  - Worked with Babbage and foresaw computers doing much more than calculating numbers
  - Loops and Conditional Branching
  - "Mother of Computer Programming"



C. Babbage (1791 – 1871)

A. Byron Lovelace (1815 – 1852)

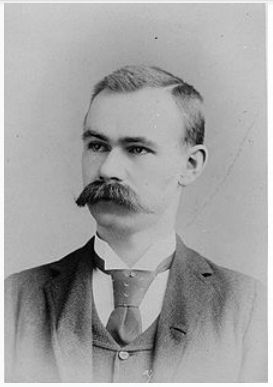


Part of Babbage's Analytical Engine

Images from Wikimedia.org

Punched Card Data Processors

- **Herman Hollerith**
  - Developed a “mechanical tabulator” in the early 1900s and used it very successfully to do the census for the US government
  - His Tabulating Machine Company (with 3 others) became **International Business Machines Corp. (*IBM*)** in 1911



H. Hollerith (1860 – 1929)

IBM punched card “Accounting Machines”, pictured in 1936.

But these were all *mostly* single-purpose calculating machines

Images from Wikimedia.org

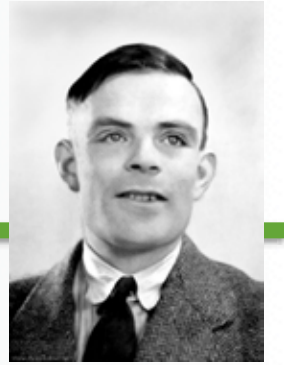
The Modern Digital Computer

A. Turing (1912 – 1954)

Alan Turing (UK)

- Theorized the possibility of computing machines capable of performing *any* conceivable mathematical computation as long as this was representable as an *algorithm*
  - Called "*Turing Machines*" (1936)
  - Lead the effort to create a machine to successfully decipher the German "Enigma Code" during World War II
    - As seen in the movie "The Imitation Game"

Turing's Legacy



The REAL
A. Turing (1912 – 1954)

- Turing Machine : An abstract model
  - Calculating machine that can "read" in symbols on a medium and "writes" out results on another, based on a "table" of instructions
  - What we call "computers" today owe a lot to this concept

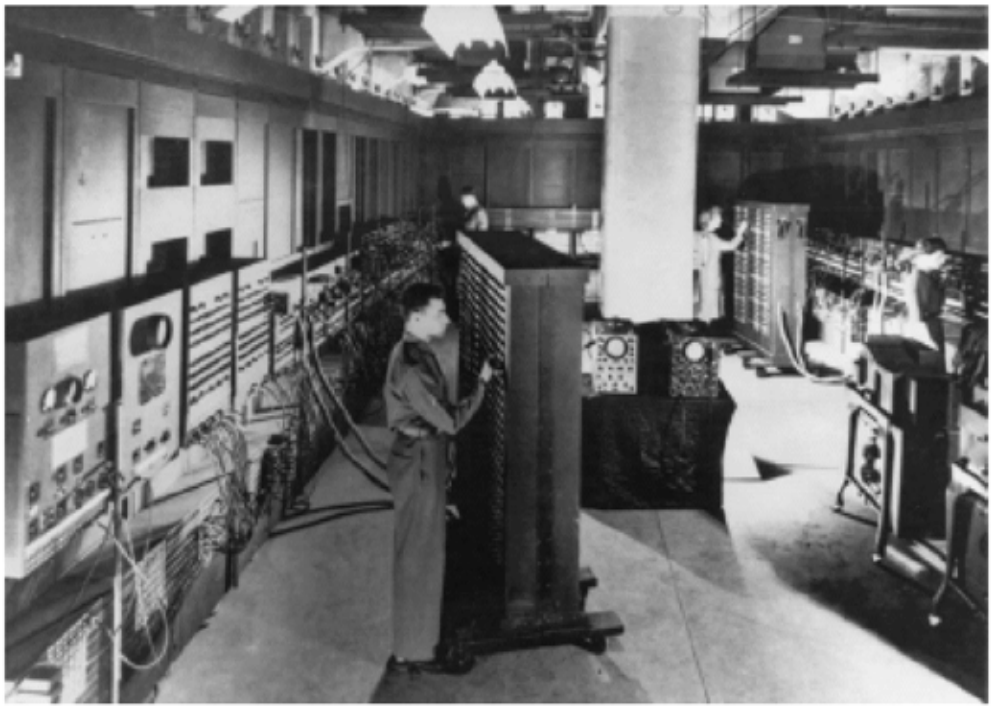
- The *Turing Test* : Asks "Can Machines Think?"

  - A test to see if a machine can exhibit intelligent behavior like a human
  - Example: CAPTCHA
    - Completely Automated Public Turing test to tell Computers and Humans Apart

- The Turing Award
  - Called the "Nobel Prize" for computing
  - For contributions of lasting and major technical importance to the computer field
  - https://en.wikipedia.org/wiki/Turing_Award

The ENIAC

electronic numerical integrator and computer – 1945



100 feet long,
by 10 feet high,
by 3 feet deep
(took up a whole big room)

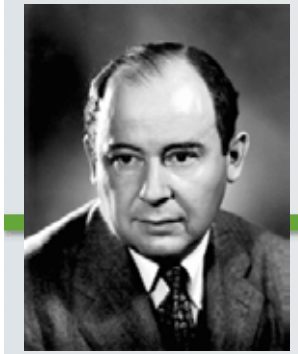
Weighed 30 tons!

Used by the military to calculate trajectories (for bombs)

Could compute in 30 seconds instead of 40 hours

Slowly replaced human “computers”

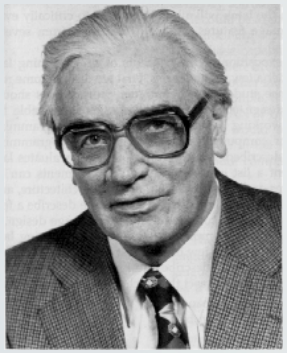
Computers Since the Mid-20th Century



John Von Neumann (1903 - 1957)
His computer architecture is what we still use today

Grace Hopper (1906 -1992)
Inventor of the first high-level computer language & compiler

- The invention of *high-level* computer languages and compilers (1950s & 1960s)
  - Up until then, operators fed these machines "1"s and "0"s for their instructions
  - Required very abstract thinking and re-arrangement of the computer "architecture"

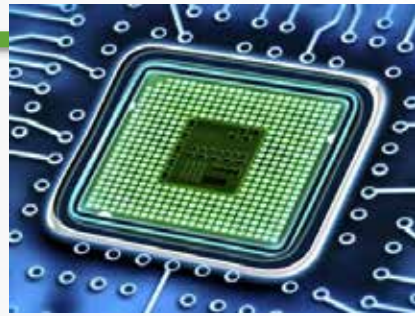


Konrad Zuse (1910 - 1995)
Built the first modern computer with high-level programming

- Computer instructions became more English-language friendly: Computers became *practical to use*
  - This needed "translator" programs (or ***compilers***) to be the go-betweens for the "high-level" languages and the machines

Katherine Johnson (1918 -)
NASA "Computer"

The Age of the Transistor



- Transistors (1947) are
  semi-conducting electronic elements
  - Replace bulky “vacuum tubes” for switching functions
  - Could now create faster AND smaller computer machines
  - The basis for all ***modern digital technology***

- Transistors: The lynchpins of modern technology
  - Kept shrinking in size while getting cheaper to produce
  - We still talk about “Moore’s Law” as the concept behind computers’ progress:
    the number of transistors in a dense integrated circuit **doubles** approximately every **two years**

The Age of The Personal Computer

- Commercialization of personal computers (1970s and 1980s)
  - Made the machines a ***lot*** smaller and cheaper
  - Apple I and II, Macintosh (Apple), PC (IBM)
  - Lots of software created to help run the hardware for everyday uses (Microsoft's DOS and Windows, Lotus' 123, etc…)

The Individual Computer Gives Way to the Networked Computer

- Invention of computer networking protocols
  - *Ethernet* and *TCP/IP* (1980s)

- Invention of the hyper-text document (and hence the WWW) in early 1990s by Berners-Lee and others

Tim Berners-Lee (1955 -)
Inventor of the hyper-text doc and WWW

- Deployment of ARPANET in the 1970s/80s (predecessor of the Internet)
  - At first, mostly just for university research use and the military
  - Once released to the public in the early 90s, it enabled us to swap pictures of cats… and world was never the same…

Computer Systems

- **Hardware**
  - The physical computer
    - CPU, Memory ICs, Printed circuit boards
    - Plastic housing, cables, etc…
- **Software**
  - The instructions and the data fed to/generated by the computer
    - Programs and applications
    - Operating systems

What is Programming?

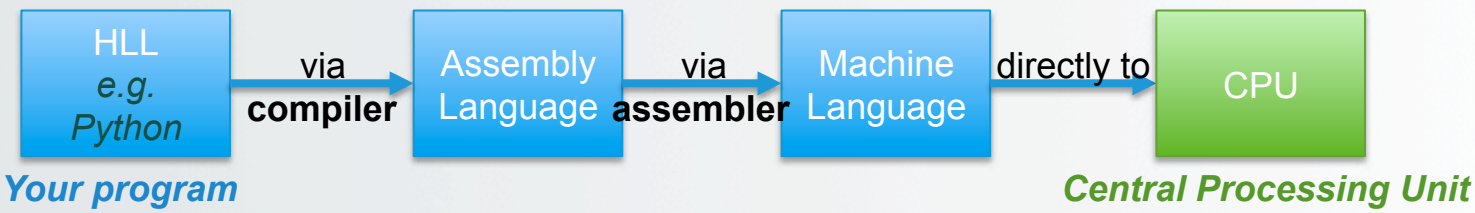
Instructing a computer what to do

- Programs – a.k.a. "Software"
  - Includes operating system, utilities, applications, …
  - Computer just sits there until instructions fed to CPU

- **Machine language** – basic CPU instructions
  - Completely numeric (as binary numbers) – i.e., computer "readable"
  - Specific to particular computer types – not portable

High-Level Computer Languages

- **A way to program computers using "human-like" language**
  - Easier to write/read (than 1s and 0s…):
    - `e.g.` ***`result = (first + second)`*** `instead of` ***"1001111010101011011 0"***
  - Translated to machine language by ***compiler programs***
    - Advantage: the same H-LL Program can be used on different machines!



High-Level Language Paradigms

- Procedural languages – focus is on *functions and process*
- **FORTRAN** (by IBM, 1957) – first commercially used high level language
  - Easy to learn – spawned thousands of new programmers
- **1970s**: Golden Age of Programmers: **C**, **PASCAL**, **BASIC**
  - Even easier to learn/use – went into use well into 1990s

- Object-oriented languages – focus on *objects*
  - **C++** (early 1980s), …, **Java** (1996)
  - Idea is to build *objects* – then let them perform tasks

- Multi-paradigm languages – combined features
  - e.g., **Python** (invented 1991… and still evolving)

~1991…2018…

BDFL Guido (1956 -)

- Derived from **ABC** – a language designed for learning how to program
  - Python designed by Guido van Rossum (an ABC designer) – to be a more general purpose language than ABC
- Python is Open Sourced since it's first version (1991)
  - So it is free!
  - Has a huge community of volunteer developers
  - Guido still the BDFL (Benevolent Dictator for Life)
- Lots of handy modules ready to use at **http://docs.python.org/**
  - More on modules later…

The Python Interpreter

A program that performs three steps over and over and …
… until `exit()` happens

1) It **reads** Python instruction statements
   - From a standard input (a.k.a. `stdin` --- usually a keyboard)
   - Or from another file (usually a text file ending in `.py`)

2) It **executes** Python commands

3) It **shows results (outcomes)** of commands, if any

Let's Fire It Up And Try Some Arithmetic With It! (demo time!)

YOUR TO-DOs

- **Sign up on Piazza if you haven't done so**
  - https://piazza.com/ucsb/winter2018/cs8
- Read the rest of **Chapter 1**
  - Get your textbook!!!

- Homework:
  - Do **Homework0** (turn it in in LAB on Tuesday 1/23)
  - Do **Homework1** (due next Monday 1/29)
- Lab:
  - Read Lab0 and prepare for it
  - Go to lab on Tuesday 1/23 and do it!

- Solve world hunger yet? Global warming?
- Eat at least half of your vegetables

</LECTURE>