An Introduction to Computer Science

CS 8: Introduction to Computer Science
Lecture #2

Ziad Matni
Dept. of Computer Science, UCSB

A Word About Registration for CS8

FOR THOSE OF YOU NOT YET REGISTERED:

- This class is currently **FULL**
- If you are on the waitlist, you will be added automatically as others drop the course
  - **THE WAITLIST WILL CLOSE ON FRIDAY AT 5 PM!**
  - **IF YOU'RE NOT REGISTERED BY THEN, YOU'RE NOT IN THE CLASS!**
- If you are not on the waitlist, you will not get into this class

Disabled Students Program Notetaker Needed

CMPSC 8 TR 3:30

$25 per unit (of the class)

(prorated based on the number of weeks selected)

Questions: Please contact WANDA THOMAS:

Phone: 805-893-2668

Email: thomas-w@sa.ucsb.edu

Please apply online at http://dsp.sa.ucsb.edu/services

Administrative

- **You must register on Piazza**
  - **https://piazza.com/ucsb/spring2017/cs8**
  - You will not get my class announcements otherwise!
    - I'm not using GauchoSpace

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

- Class webpage: **https://ucsb-cs8-s17.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, <u>YOU MUST</u>:

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

What is this “Computer” you speak of?

Let’s define a “computer”

- Computer (n.): a computing device

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

Algorithms!

Algorithm

- A *step-by-step* logical procedure to *solve a problem*
  - Like a very precise recipe!

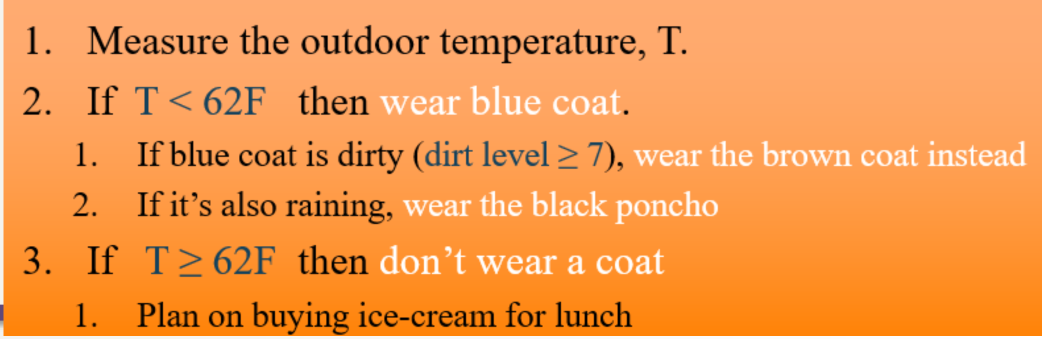
- Named after famed 9th-century Persian mathematician Al-Khawarizmi who put a name to the practice and published a lot on it

Examples of Everyday Algorithms

- **Problem to Solve**: What to wear today?
  - – More precisely, “what coat should I wear today?”
- **Algorithm**: *(assuming problem is only weather-related)*

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

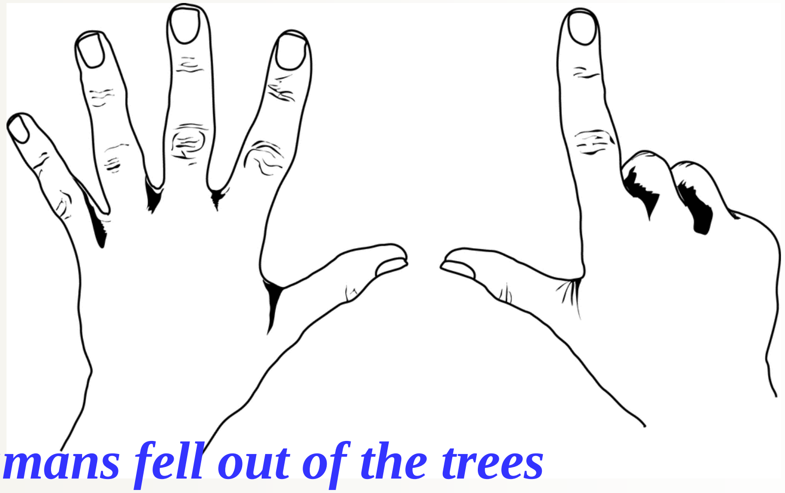
```
Define decision =    1. wear blue coat,
                     2. wear brown coat,
                     3. wear black poncho,
                     4. wear nothing

Get T

If ( (T < 62) AND (Dirt_Level < 7) ) then (decision = 1)
If ( (T < 62) AND (Dirt_Level ≥ 7) ) then (decision = 2)
If ( (T < 62) AND (Rain = True) ) then (decision = 3)
Otherwise (decision = 4) and (ice_cream_lunch = True)

The End
```

Computers = Computing Devices

Compute

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

- What was the first computing tool ever?



Invented around when humans fell out of the trees

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

What is “Computer Science”?

The study of :

1. The designs and uses of computers
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

Images from Wikimedia.org

B. Pascal (1623 – 1662)

"Pascaline" : a calculating machine (1652)

- **Blaise Pascal**
  - Mechanical device that could add, subtract, divide & multiply using gears
- **Joseph Jacquard**
  - Jacquard's Loom, used punched cards to describe patterns

J. Jacquard (1752 – 1834)

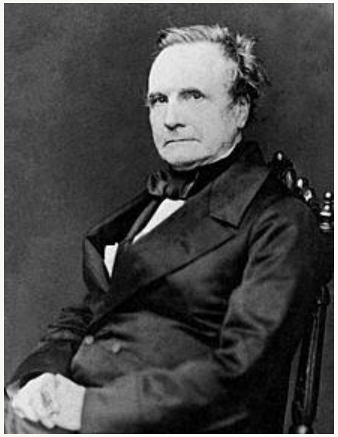
Jacquard Loom (invented 1801)

Computing Devices for General Purposes

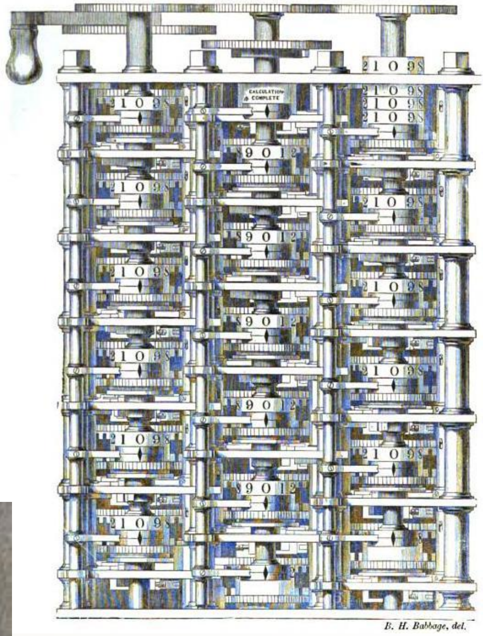
Images from Wikimedia.org

- **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)

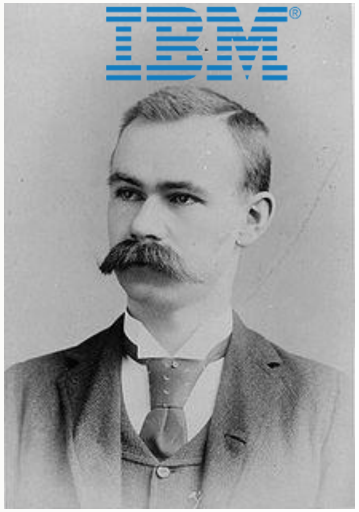


Part of Babbage’s Analytical Engine

A. Byron Lovelace (1815 – 1852)

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 single-purpose calculating machines

Images from Wikimedia.org

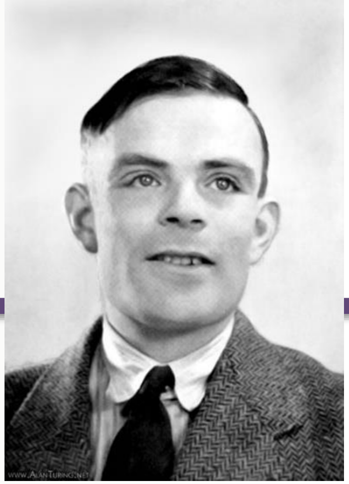
The Modern Digital Computer

Alan Turing

- 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

A. Turing (1912 – 1954)

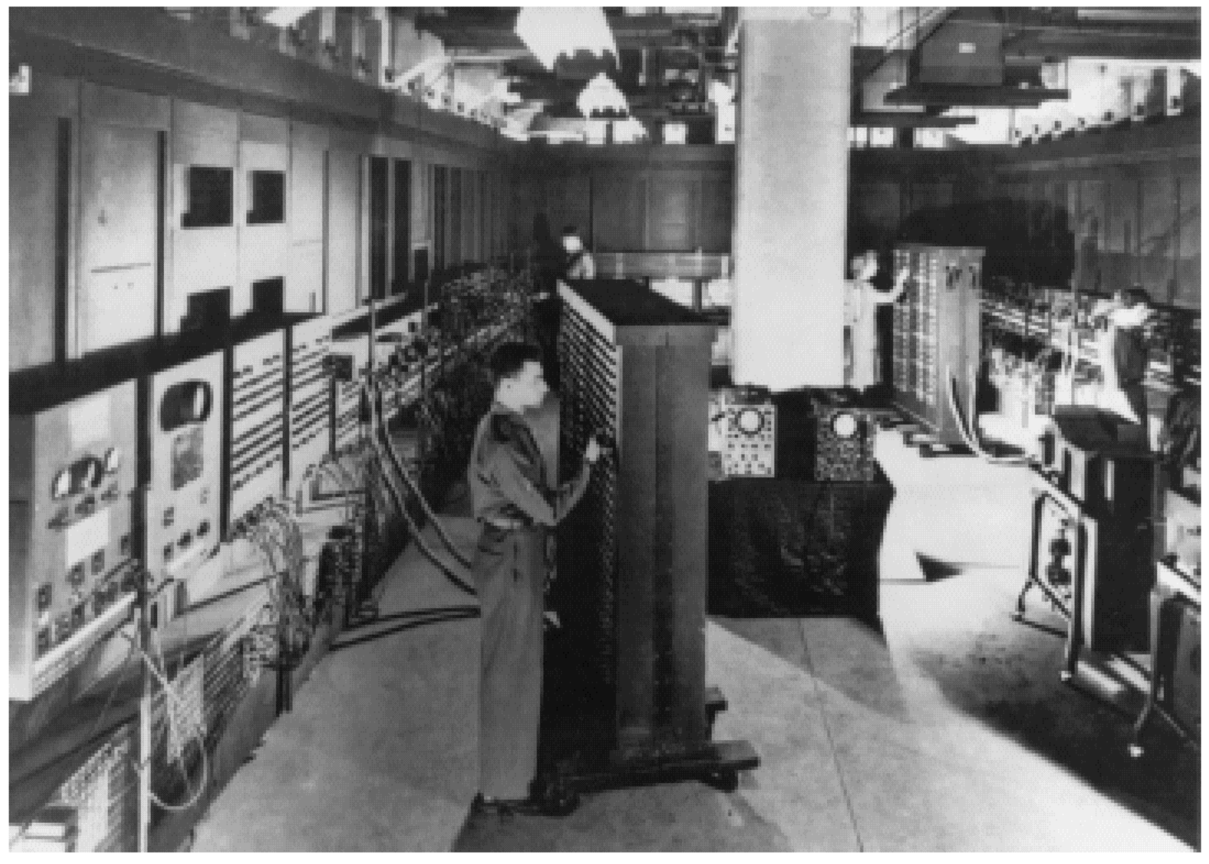
Turing’s Legacy



- 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
- 30 tons!
- Trajectories (for bombs) computed in 30 seconds instead of 40 hours
- Slowly replaced human "computers"

Computers
Since the Mid-20th Century

- UNIVAC (1951)
  - The 1st general purpose computers (private use and commercial use, respectively)
  - 1st to be developed by a private corporation and sold to other companies
  - Enormous machines – took up entire floors of a building

- The invention of *high-level* computer languages and compilers (1950s & 1960s)

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

- Computer instructions became less “1”s and “0”s and more “English”-friendly
  - Needed “translator” programs to handle these “high-level” languages a.k.a ***compilers***

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

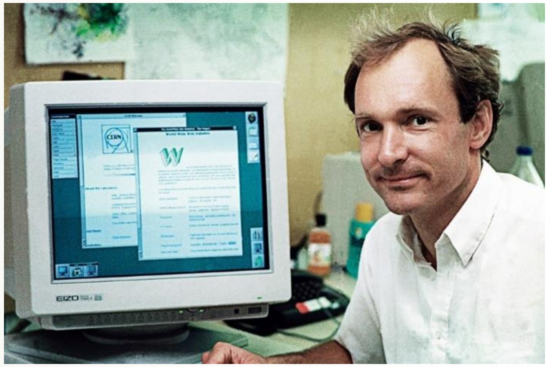
The Age of The Personal Computer

- 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…)

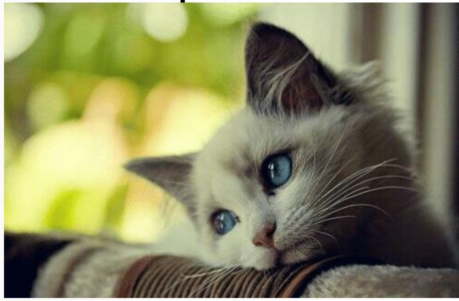
The Individual Computer Gives Way to the Networked Computer

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

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



- Invention of the hyper-text document (and hence the WWW) in early 1990s
- 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…

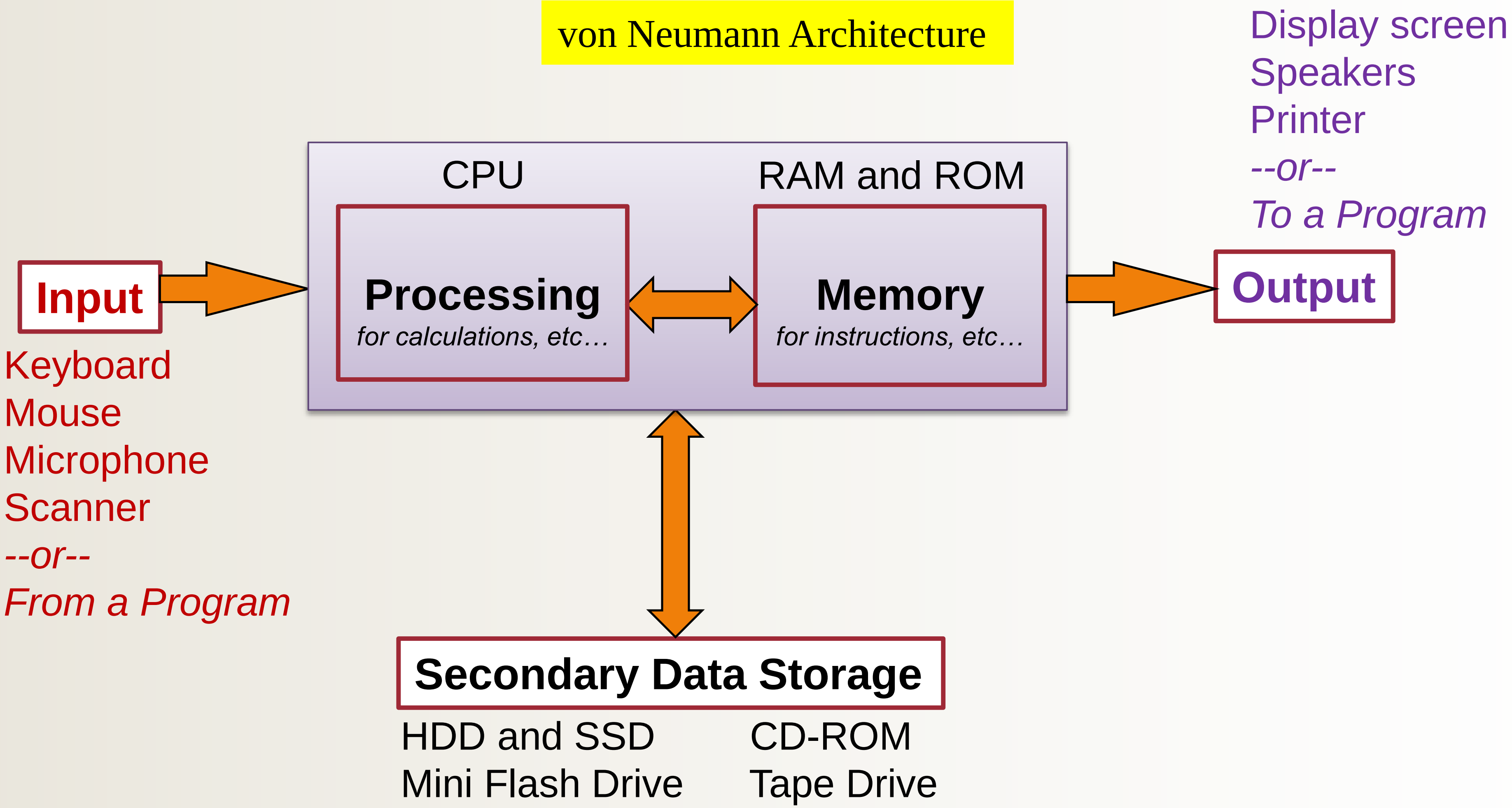


Computer Systems

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

A Map of Computer Components (Modern Computer Architecture)

von Neumann Architecture



Display screen
Speakers
Printer
--or--
To a Program

CPU

Processing
for calculations, etc…

RAM and ROM

Memory
for instructions, etc…

Input

Keyboard
Mouse
Microphone
Scanner
--or--
From a Program

Output

Secondary Data Storage

HDD and SSD
Mini Flash Drive
CD-ROM
Tape Drive

CPU = Central Processing Unit
RAM = Random-Access Memory
HDD = Hard Disk Drive
SSD = Solid State Drive
CD-ROM = Compact Disk – Read-Only Memory
OS = Operating System

5 Main Components to Computers

1. Inputs
2. Outputs
3. Processor
4. Main memory
   - Usually inside the computer, volatile
5. Secondary memory
   - More permanent memory for mass storage of data

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 – i.e., computer "readable"
    - e.g., `43065932752`, might mean add (operation# **`43`**) value at memory address **`065`** to value at address **`932`** and store result at memory address **`752`**
    - But in binary form, of course – `1001101`…`etc`…
  - Specific to particular computer types – not portable

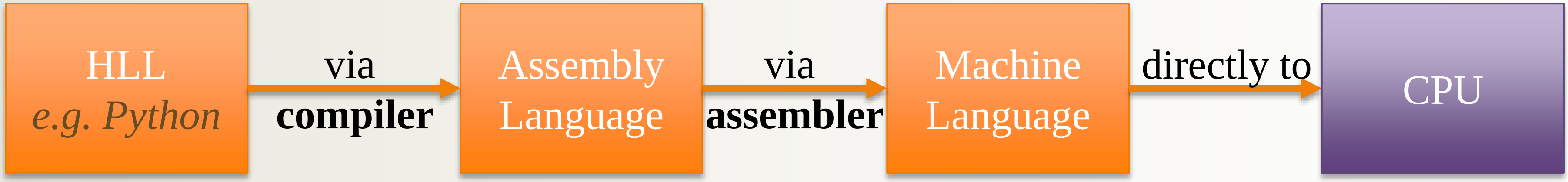
Programming Languages: **Assembly** (Low-Level Language)

- **Assembly language** – 1st real advance
  - Instead of instructions that looked like …*0101111001*…
  - Human-readable instructions (mnemonics)
    – translated to machine language by ***assembler programs***
    - e.g., *ADD X Y T*
    - Symbolic names represent operations and memory addresses
  - Very basic – lots of instructions to do simple things
  - Still processor-specific

    (so different A.Ls for different computers)

Programming Languages: High-Level Languages

- **High-level languages** – a much bigger advance
  - Easier to write/read:
    - `e.g.` ***`result = (first + second)`*** *`instead of`* ***`"ADD X Y Z"`***
  - Translated to assembly language by ***compiler programs***
    - Now the same code works on many types of processors!



High-Level Language Paradigms

- Procedural languages – focus on *functions*
- **Fortran** (by IBM, 1957) – first high level language
  - Easy to learn – spawned thousands of new programmers
- **C**, **Pascal**, **BASIC** – developed through 1970s
  - Even easier to learn/use – ever more programmers 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** (1991… and still evolving)

python™

~1990…2017…

BDFL Guido (1956 -)

- Derived from ABC – a language designed for learning how to program
  - By Guido van Rossum (an ABC designer) – to be a more general purpose language than ABC
- Open sourced since version 1.0 (1991)
  - So it is free!
  - 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/

The Python Interpreter

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

1) It reads Python instruction statements
   - From a standard input (a.k.a. `stdin`; usually keyboard)
   - Or from a text file (usually has file ending `.py`)
2) It executes Python commands
3) It prints results of commands if there are any

Try some arithmetic with it!

Numbers are Objects to Python

- Each object *type* has: data and related operations

- 2 basic number types and one derived type
  - **Integers** (like `5`, `-72`) – add, subtract, multiply, …
  - **Floating point** numbers (like `0.005`, `-7.2`) – operations similar but *not exactly the same as integer* operations
  - **Complex** numbers (like `3.4 + j5`) – have *two* floating point parts, but operations are specific to complex numbers

- Expect many ***non-number object*** types later
  - But they still will have data and related operations

YOUR TO-DOs

- ❑ **Sign up on Piazza if you haven’t yet**
- ❑ Read the rest of **Chapter 1**
- ❑ Do **Homework1** (due next **Thursday 4/13**)
- ❑ Turn in your **Lab0** by **Friday (tomorrow)**
- ❑ I’ll put up **Lab1** online by Mon/Tue:
  give it a look when it’s there to prep for Wed.

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

</LECTURE>