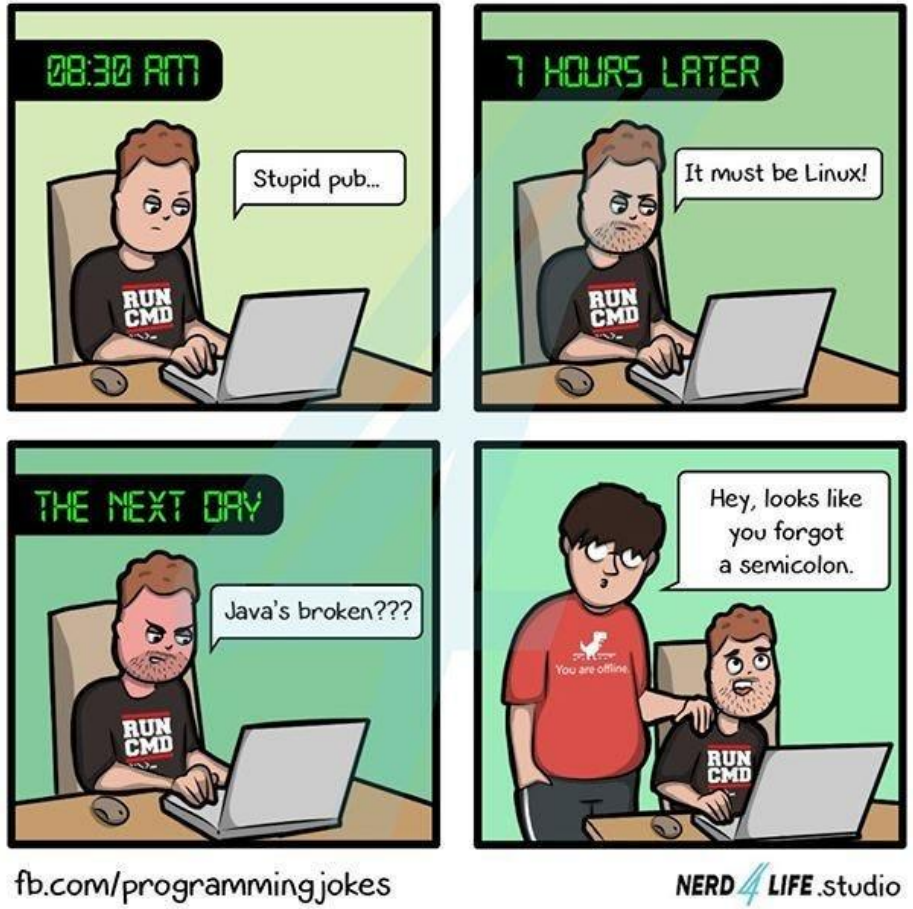
CS 170: Programs, Files, and Memory

English Teachers: You will almost never use the semicolon.
Java programmers:



Announcements & Reminders

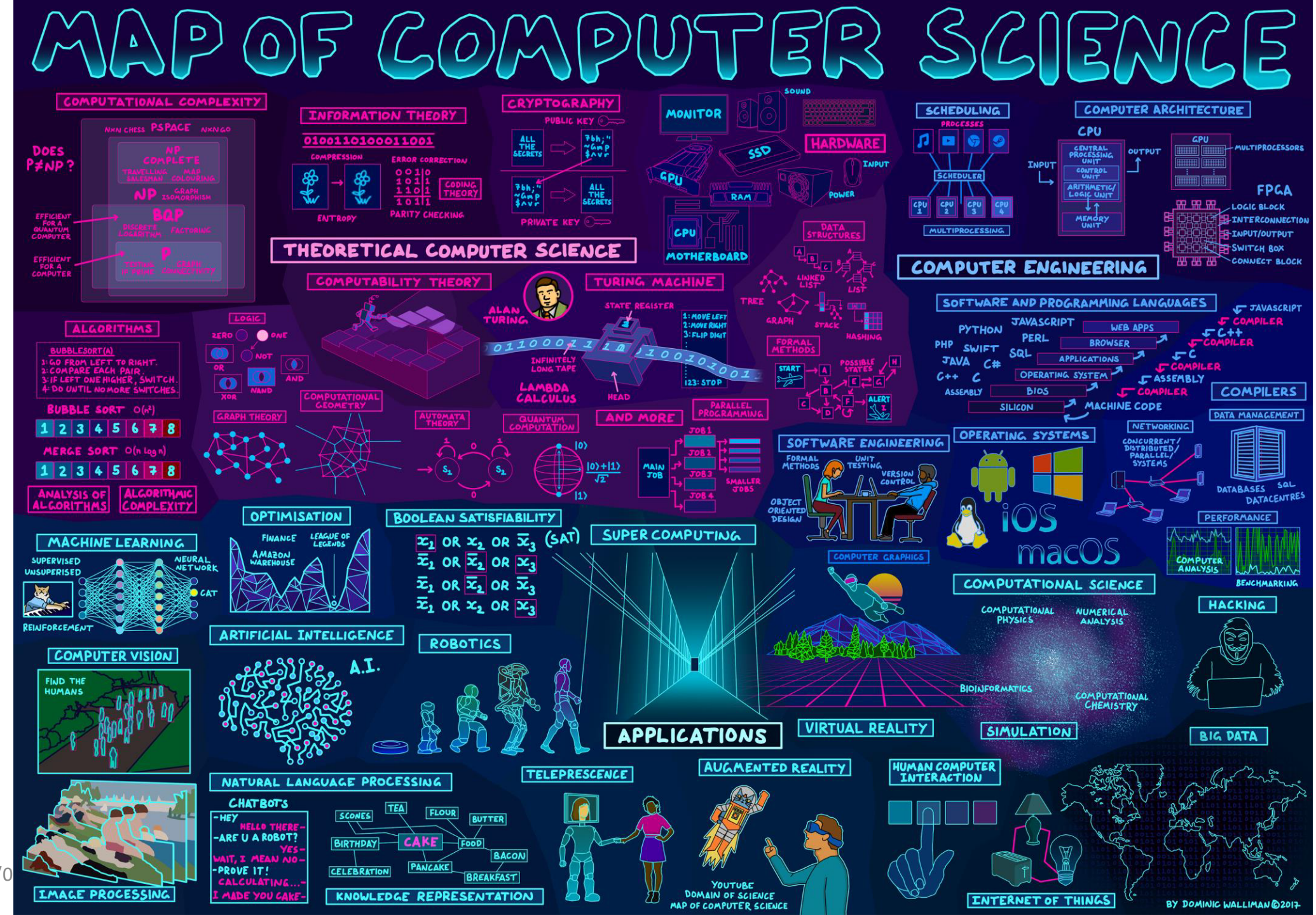
- Complete Beginning of Semester Survey on Canvas
- Join 170 Slack (link on Canvas)
- Read through the syllabus
- Labs begin Friday (MSC E308A – computer lab classroom)
- CWP 1 due Friday at 4pm
  - Gradescope overview at end of class
- If you have a DAS letter, contact me to discuss

What is a computer?

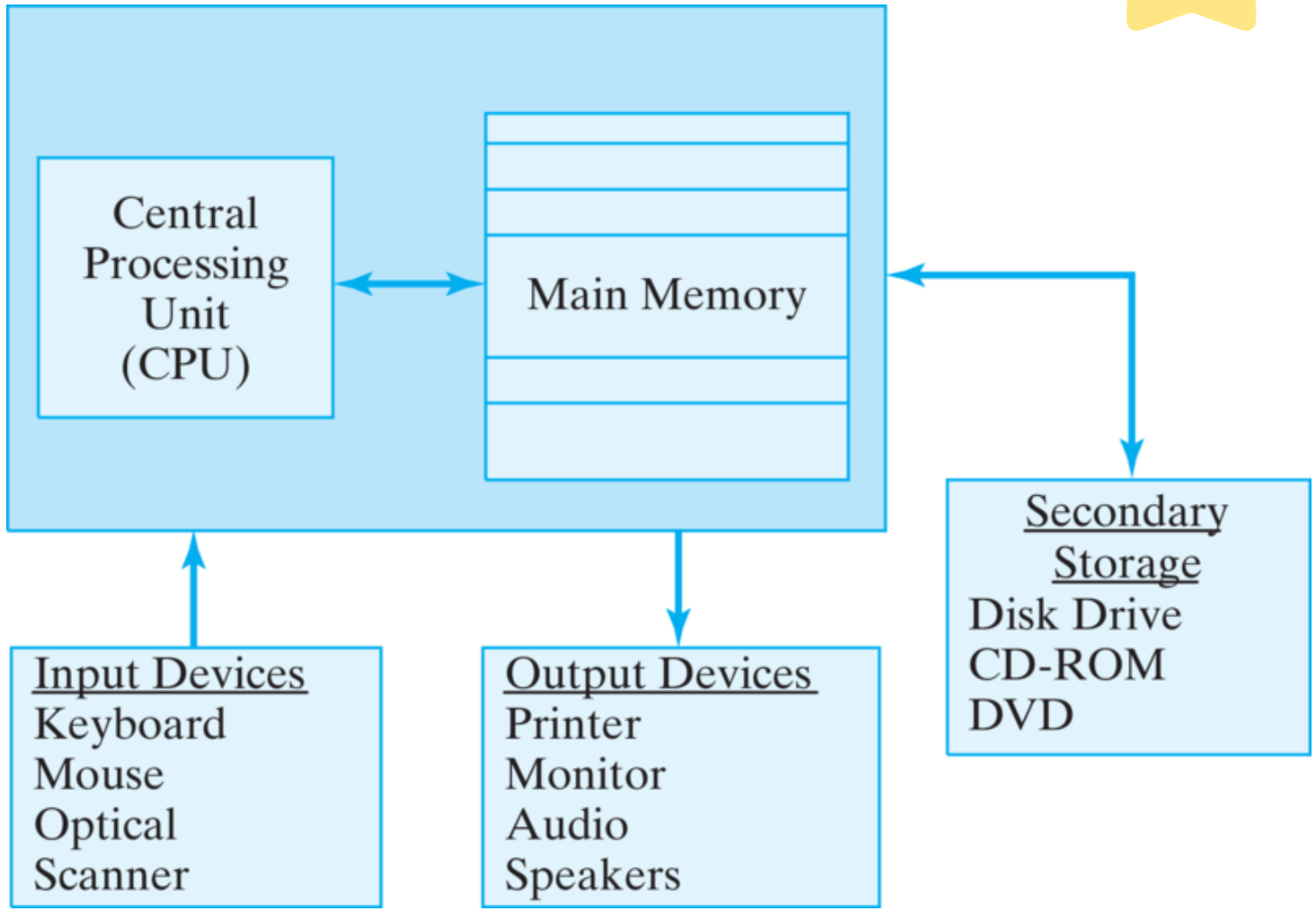
- **Computer**: machine that performs calculations and processes information
- **Computer program**: set of instructions that tell a computer what to do
- **Hardware**: electronic and mechanical components of a computer
- **Software**: programs that control the hardware
- **General-purpose computer**: can store and execute many different programs (e.g. word processor, internet browser, etc.)
- **Special-purpose computer**: single, fixed program (e.g. microwave, calculator, watch, etc.)

What is Computer Science?

- **Computer Science**: study of computers and computational systems
- Not just programming!
- Includes:
  - Theory (algorithms, complexity, logic, quantum computing, …)
  - Systems (operating systems, computer architecture, compilers, …)
  - Applications (machine learning, AI, robotics, natural language processing, big data/ data science, …)


MAP OF COMPUTER SCIENCE
THEORETICAL COMPUTER SCIENCE
COMPUTATIONAL COMPLEXITY
DOES P≠NP?
NXN CHESS
PSPACE
NXNGO
NP COMPLETE
TRAVELLING SALESMAN
MAP COLOURING
NP
GRAPH ISOMORPHISM
BQP
DISCRETE LOGARITHM
FACTORING
P
TESTING IF PRIME
GRAPH CONNECTIVITY
EFFICIENT FOR A QUANTUM COMPUTER
EFFICIENT FOR A COMPUTER
INFORMATION THEORY
0100110100011001
COMPRESSION
ENTROPY
ERROR CORRECTION
PARITY CHECKING
CODING THEORY
CRYPTOGRAPHY
PUBLIC KEY
ALL THE SECRETS
PRIVATE KEY
COMPUTABILITY THEORY
ALAN TURING
TURING MACHINE
STATE REGISTER
INFINITELY LONG TAPE
HEAD
LAMBDA CALCULUS
1: MOVE LEFT
2: MOVE RIGHT
3: FLIP DIGIT
123: STOP
ALGORITHMS
BUBBLESORT(A)
1: GO FROM LEFT TO RIGHT.
2: COMPARE EACH PAIR.
3: IF LEFT ONE HIGHER, SWITCH.
4: DO UNTIL NO MORE SWITCHES.
BUBBLE SORT O(n²)
MERGE SORT O(n log n)
ANALYSIS OF ALGORITHMS
ALGORITHMIC COMPLEXITY
LOGIC
ZERO
ONE
NOT
OR
AND
XOR
NAND
GRAPH THEORY
COMPUTATIONAL GEOMETRY
AUTOMATA THEORY
QUANTUM COMPUTATION
AND MORE
PARALLEL PROGRAMMING
MAIN JOB
SMALLER JOBS
JOB 1
JOB 2
JOB 3
JOB 4
DATA STRUCTURES
LINKED LIST
TREE
GRAPH
STACK
HASHING
FORMAL METHODS
START
POSSIBLE STATES
ALERT
COMPUTER ENGINEERING
MONITOR
SOUND
HARDWARE
SSD
GPU
RAM
POWER
INPUT
CPU
MOTHERBOARD
SCHEDULING
PROCESSES
SCHEDULER
CPU 1
CPU 2
CPU 3
CPU 4
MULTIPROCESSING
COMPUTER ARCHITECTURE
CPU
CENTRAL PROCESSING UNIT
CONTROL UNIT
ARITHMETIC/ LOGIC UNIT
MEMORY UNIT
INPUT
OUTPUT
GPU
MULTIPROCESSORS
FPGA
LOGIC BLOCK
INTERCONNECTION
INPUT/OUTPUT
SWITCH BOX
CONNECT BLOCK
SOFTWARE AND PROGRAMMING LANGUAGES
PYTHON
JAVASCRIPT
PERL
PHP
SWIFT
SQL
JAVA
C#
C++
C
ASSEMBLY
WEB APPS
BROWSER
APPLICATIONS
OPERATING SYSTEM
BIOS
SILICON
MACHINE CODE
JAVASCRIPT
COMPILER
C++
COMPILER
C
COMPILER
ASSEMBLY
COMPILER
COMPILERS
DATA MANAGEMENT
DATABASES
SQL
DATACENTRES
NETWORKING
CONCURRENT/ DISTRIBUTED/ PARALLEL/ SYSTEMS
PERFORMANCE
COMPUTER ANALYSIS
BENCHMARKING
SOFTWARE ENGINEERING
FORMAL METHODS
UNIT TESTING
VERSION CONTROL
OBJECT ORIENTED DESIGN
OPERATING SYSTEMS
iOS
macOS
MACHINE LEARNING
SUPERVISED
UNSUPERVISED
NEURAL NETWORK
CAT
REINFORCEMENT
OPTIMISATION
FINANCE
LEAGUE OF LEGENDS
AMAZON WAREHOUSE
BOOLEAN SATISFIABILITY
(SAT)
SUPER COMPUTING
ARTIFICIAL INTELLIGENCE
A.I.
ROBOTICS
COMPUTER VISION
FIND THE HUMANS
IMAGE PROCESSING
APPLICATIONS
COMPUTER GRAPHICS
COMPUTATIONAL SCIENCE
COMPUTATIONAL PHYSICS
NUMERICAL ANALYSIS
BIOINFORMATICS
COMPUTATIONAL CHEMISTRY
SIMULATION
HACKING
BIG DATA
VIRTUAL REALITY
NATURAL LANGUAGE PROCESSING
CHATBOTS
-HEY
-HELLO THERE-
-ARE U A ROBOT?
-YES-
-WAIT, I MEAN NO-
-PROVE IT!
-CALCULATING...-
-I MADE YOU CAKE-
SCONES
TEA
FLOUR
BUTTER
BIRTHDAY
CAKE
FOOD
BACON
CELEBRATION
PANCAKE
BREAKFAST
KNOWLEDGE REPRESENTATION
TELEPRESCENCE
AUGMENTED REALITY
YOUTUBE
DOMAIN OF SCIENCE
MAP OF COMPUTER SCIENCE
HUMAN COMPUTER INTERACTION
INTERNET OF THINGS
BY DOMINIC WALLIMAN ©2017

Parts of a Computer



Main Memory vs. Secondary Memory

- **Main memory** is where programs and data are kept when the processor is actively using them
- **Secondary memory** is “long-term” storage
- When programs/ data become active, they are copied from secondary memory to main memory
  - A copy remains in secondary memory
- Nothing permanent is kept in main memory
- Main memory is connected to the processor so accessing it is very fast

RAM – Random Access Memory

This acronym just helps us know what's going on better now

- Another term for main memory
- Random => memory can be accessed in any order
- RAM also refers to the type of silicon chip used to implement main memory
- Size of RAM: how big the main memory is
  - E.g. 512 megabytes, 8 gigabytes, etc.
  - 1 megabyte can hold approximately one million (10^6) characters of a word processing document

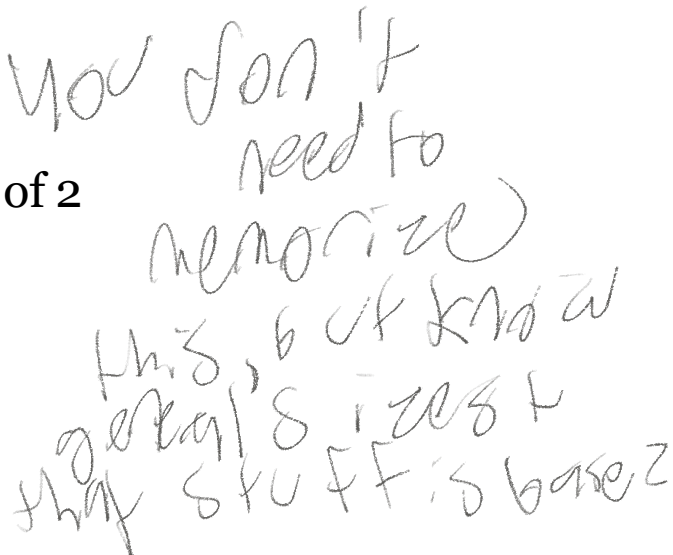
Secondary Memory

- Long-term storage, e.g. hard drive
- Hard drives have a much larger storage capacity than main memory
- Data is organized into **files** (a collection of data on disk that has a name)
- Accessing/ moving data in secondary memory is much slower than main memory

Memory Storage

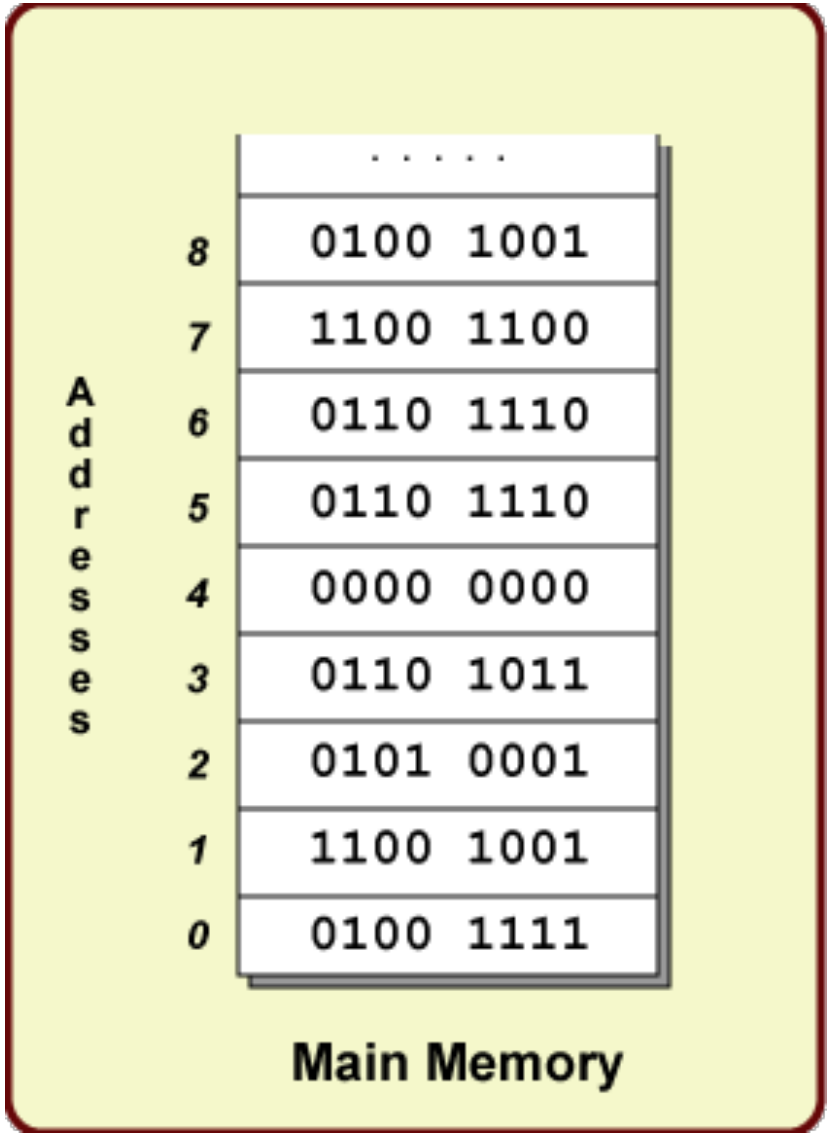
- In both main and secondary memory, information is stored as patterns of **bits**
- A **bit** is a single *on/off* value (e.g. true/false, 0/1)
- Data is usually organized in groups of 8 bits called a **byte**
- One byte can hold a single character (e.g. ‘a’)
- Four bytes represent integers (e.g. 125)

Name	Number of Bytes	power of 2
byte	1	2^0
kilobyte	1024	2^{10}
megabyte	1,048,576	2^{20}
gigabyte	1,073,741,824	2^{30}
terabyte	1,099,511,627,776	2^{40}



Memory Addressing

- Each box in the picture represents a single byte with an address
- Stored at each memory address are the 8 bits (0s and 1s) making up the byte
- This binary pattern represents something – a character, a number, instructions to run a program, etc.
- Writing in a programming language like Java, you do not (usually) need to worry about where things are in memory during program execution because Java takes care of that for you



Programs

- Programs are just files that contain information the processor can use to perform certain actions
- Example: Operating System
  - Always running when the computer is on
  - Coordinates operation of hardware/ software components
  - Responsible for starting applications, running them, managing resources, etc.
  - When the computer is first started, starting the OS is called **booting**
    - The OS needs to be involved itself in starting; "pulling themselves up by their bootstraps"

The role of the OS

The OS needs to start itself using itself. It needs itself to start running

Starting a Program

There's a lot of stuff going on underneath for apps...

1. User asks to run an application by clicking on an icon, making a menu choice, etc.
2. The OS determines the name of the application
3. The OS finds the files on the hard disk where the application and its data are stored
4. The OS finds an unused section of main memory large enough for the application
5. The OS copies the application and data to main memory
   - Remember this is just a copy; the software on hard disk is unchanged
6. The OS sets up resources and starts running the application

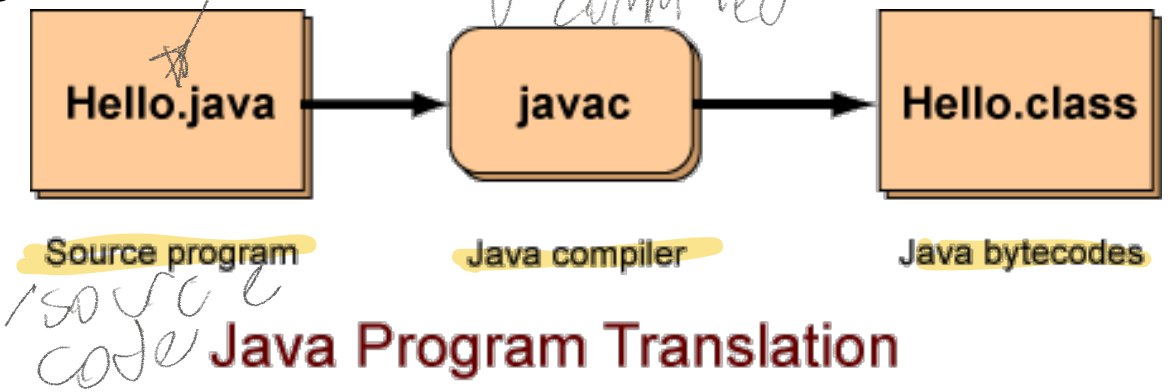
Java Source Code

- **Source Code**: text file containing a program written in a programming language (e.g. HelloWorld.java)
- Contains ordinary text stored as bytes
- Can be printed, displayed, altered with a text editor, etc.
- Cannot be directly executed (run) by computer system

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

Java Bytecode

- To run the program, the source file is first translated into a file of **bytecodes**
- **Java Bytecode**: machine instruction for a Java processor
- A file of bytecodes (.class file) is a machine language program suitable for execution by the Java processor
- The **compiler** (javac) program translates the source program to file of bytecodes; this process is called **compilation**

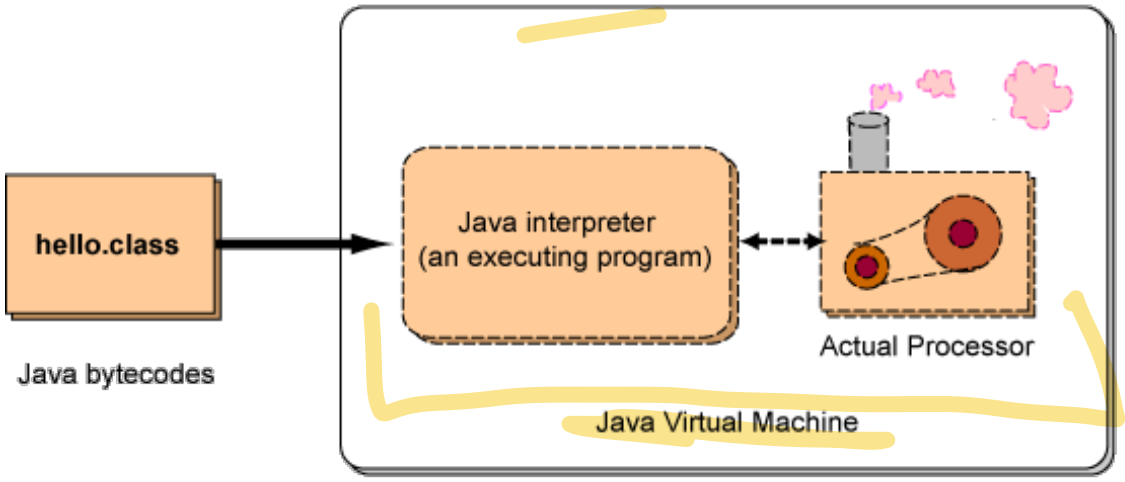


Java Program Translation

part of the anatomy of a Java program

Java Processor: Java Virtual Machine (JVM)

- The Java processor is (usually) not a physical processor like the processor on your computer
- The Java bytecodes are executed by another program that is a software version of a Java processor called the **Java Virtual Machine (JVM)**
- This software is also called a **Java interpreter**



Interpreting Java Bytecode on a Virtual Machine

Parts of a Java Source File

- Every Java file contains a **class** with the same name as the file
- The compiler looks for a **main method** as the entry point for program execution
- The lines of code inside the main method are executed in order

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