CSCI 132:
Basic Data Structures and Algorithms

Lecture 2: Computers, Coding, and Java

Reese Pearsall

Spring 2023

Announcements

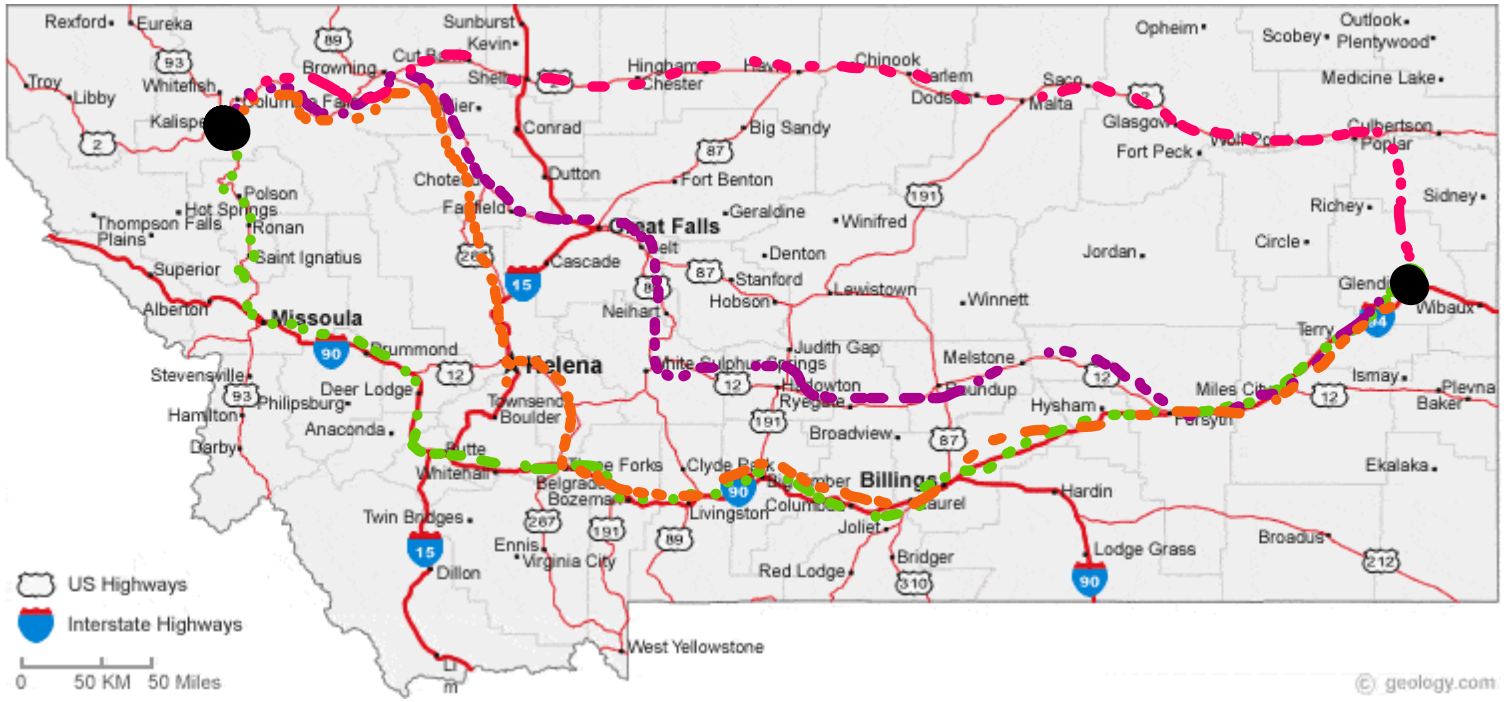
- Fill out the course questionnaire
- Lab 1 posted (Due 1/24) (really easy)
- Make sure you get an IDE downloaded before next week

You won’t be tested over any of today’s content

This is just helpful background information ☺

Computer Science is the study of computers, computation, and how we can use computers to solve *problems*

Computer Science is the study of computers, computation, and how we can use computers to solve *problems*



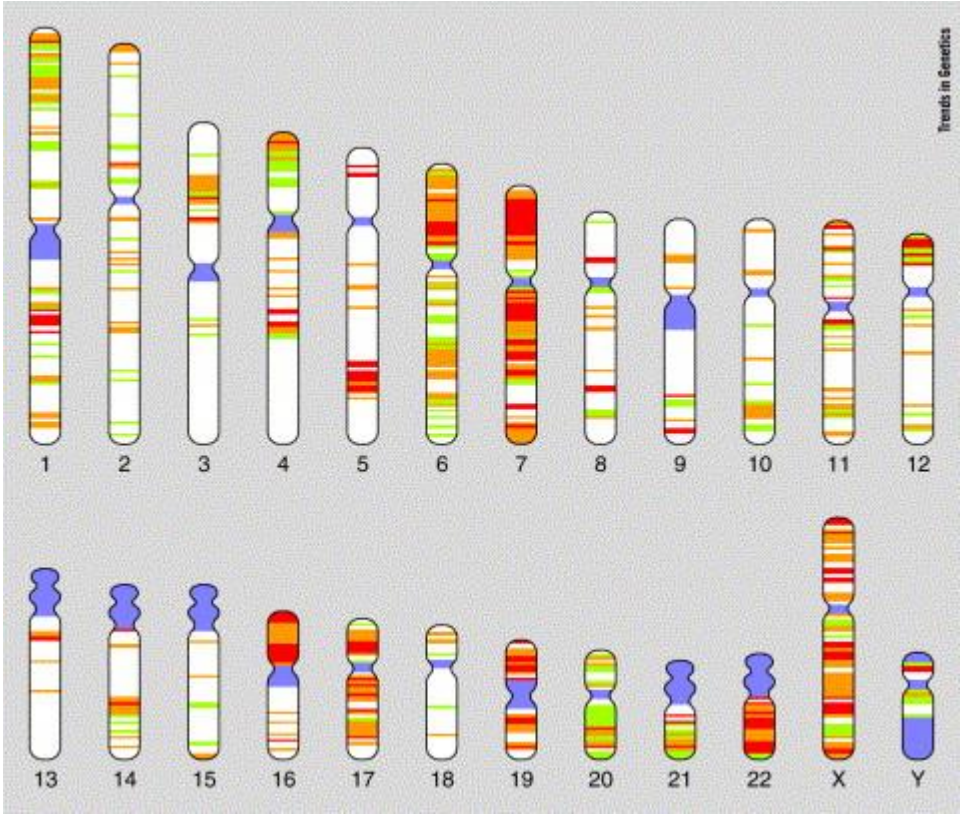
What is the optimal route from Kalispell to Glendive?

Computer Science is the study of computers, computation, and how we can use computers to solve *problems*

How to sort a list of integers from least to greatest?

11	99	0	55	5	14	89	23	7	1	10

Computer Science is the study of computers, computation, and how we can use computers to solve *problems*

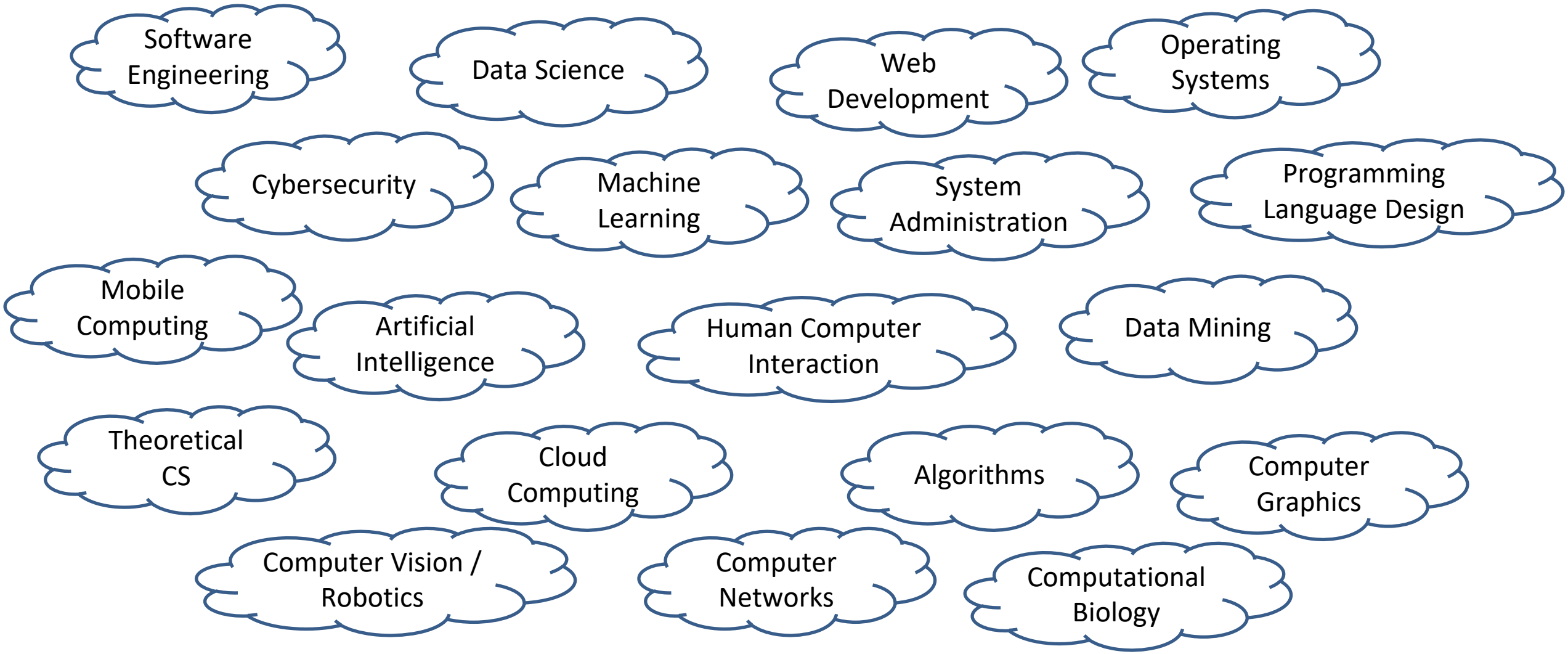


How to sequence the human genome?

Computer Science is the study of computers, computation, and how we can use computers to solve *problems*

How to defend a network and prevent intrusions?

Computer Science is the study of computers, computation, and how we can use computers to solve *problems*



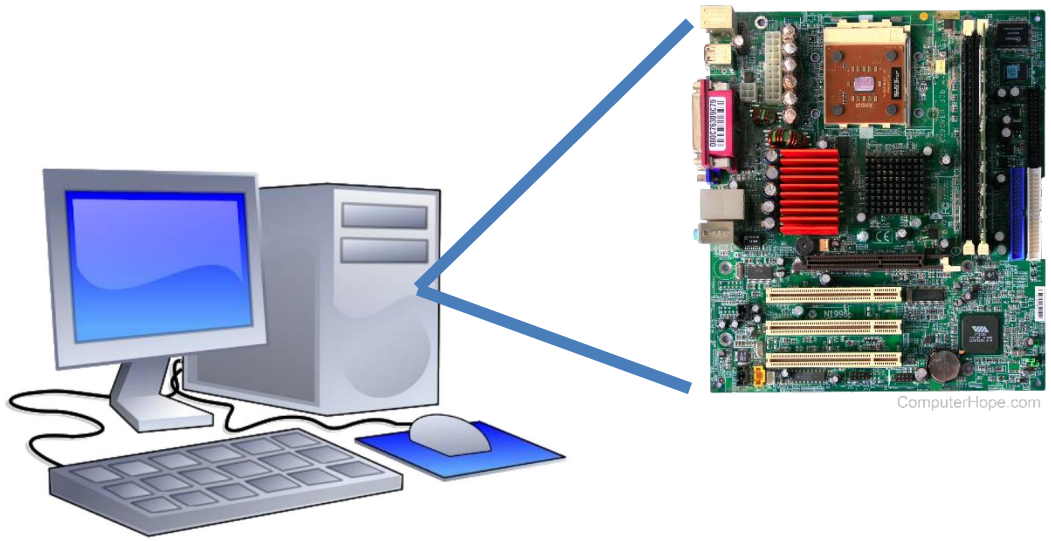
What is a computer?

What is a computer?

A magical box that does stuff

What is a computer?

A **semi-**magical box that ~~does stuff~~ **executes instructions**



What is so magical about a computer?

We use computers every day for many different things

But,

What is so magical about a computer?

We use computers every day for many different things

But,

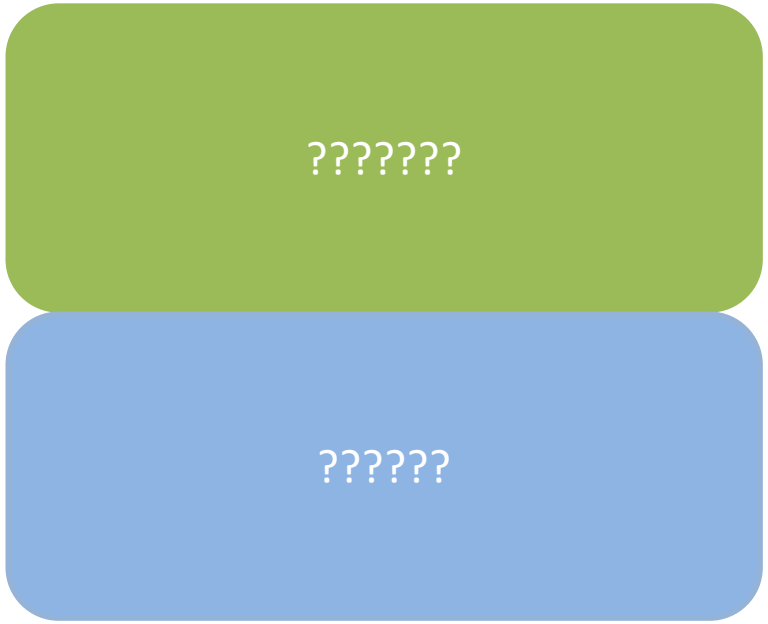
Big Idea

Computers only understand instructions in the form of 0s and 1s (binary)

~~Welcome to CSCI 132~~

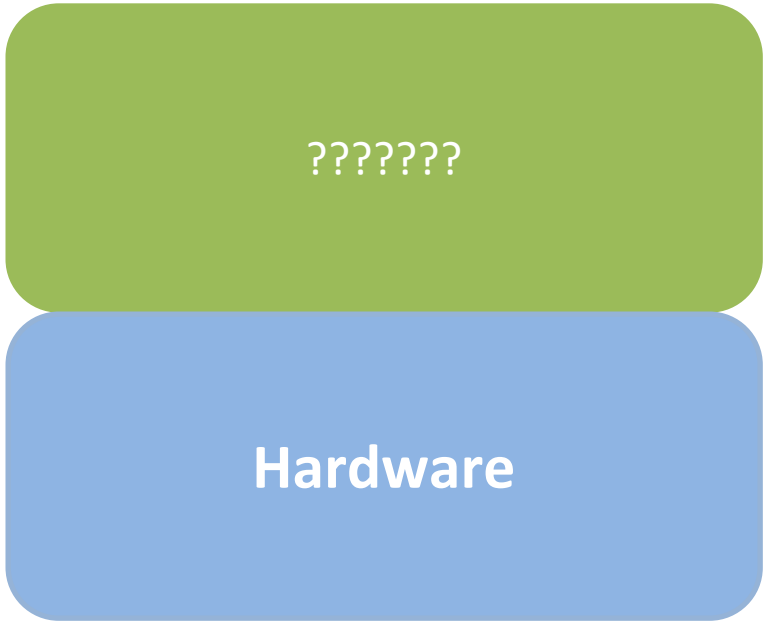
01010111 01100101 01101100 01100011 01101111
01101101 01100101 00100000 01110100 01101111
00100000 01000011 01010011 01000011 01001001
00100000 00110100 00110111 00110110

How does this happen?



From a high level, we will divide a computer system into two parts

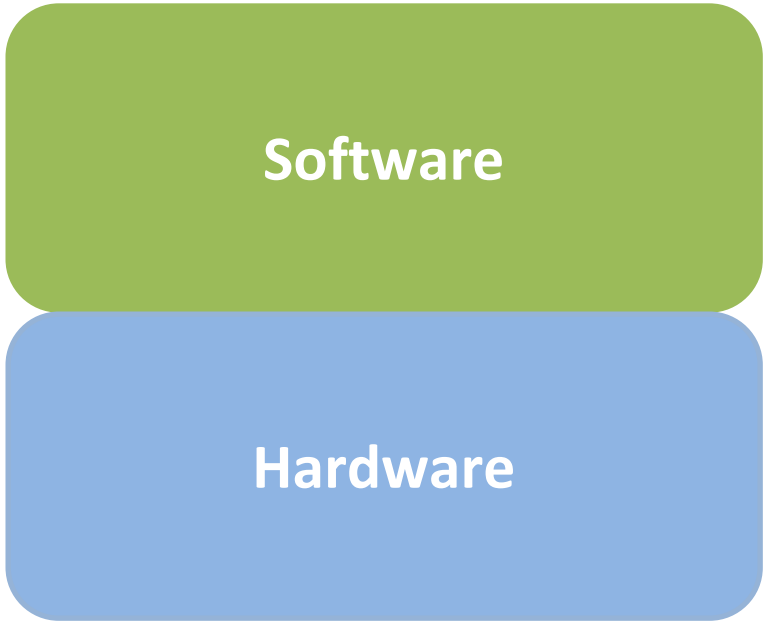
How does this happen?



From a high level, we will divide a computer system into two parts

I. Hardware

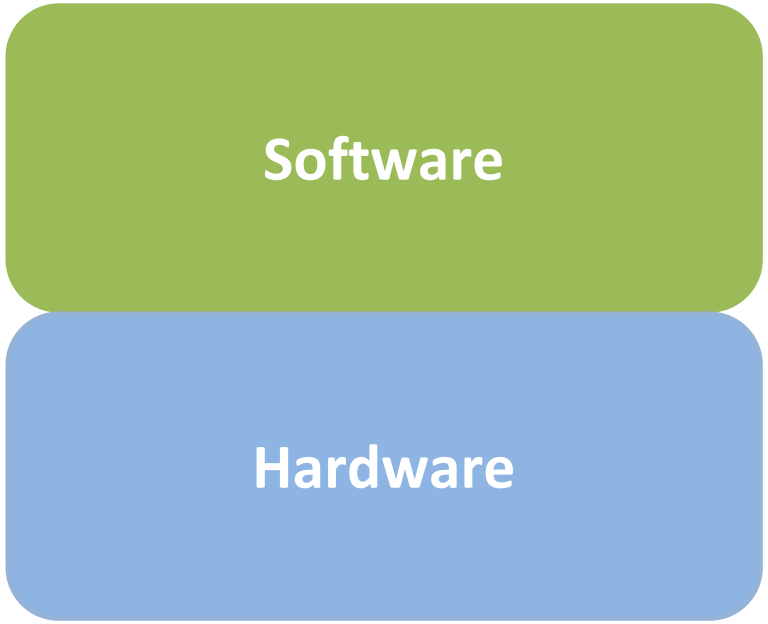
How does this happen?



From a high level, we will divide a computer system into two parts

I. **Hardware**
II. **Software**

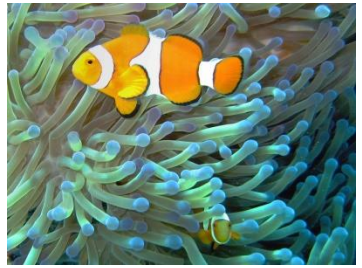
How does this happen?



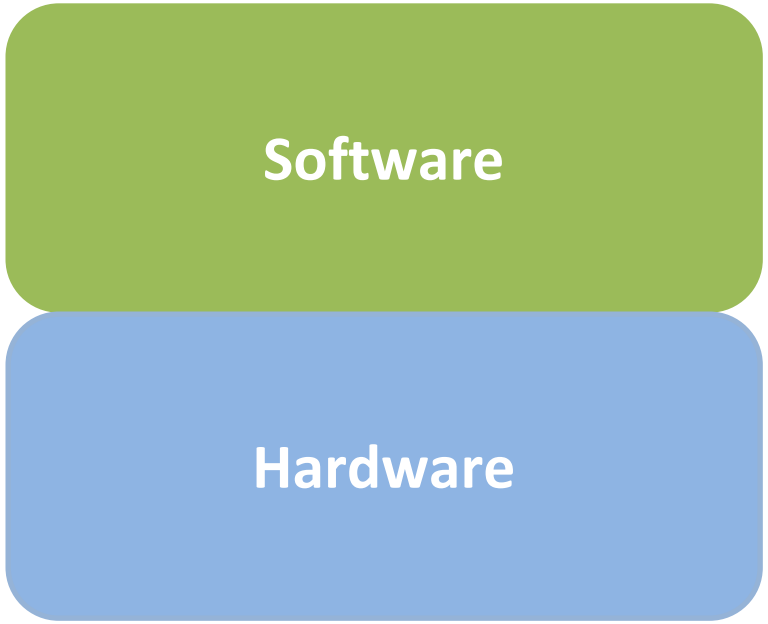
From a high level, we will divide a computer system into two parts

I. **Hardware**
II. **Software**

Symbiotic relationship



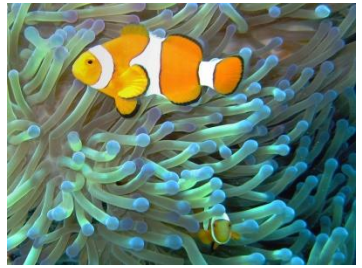
How does this happen?



From a high level, we will divide a computer system into two parts

I. **Hardware**
II. **Software**

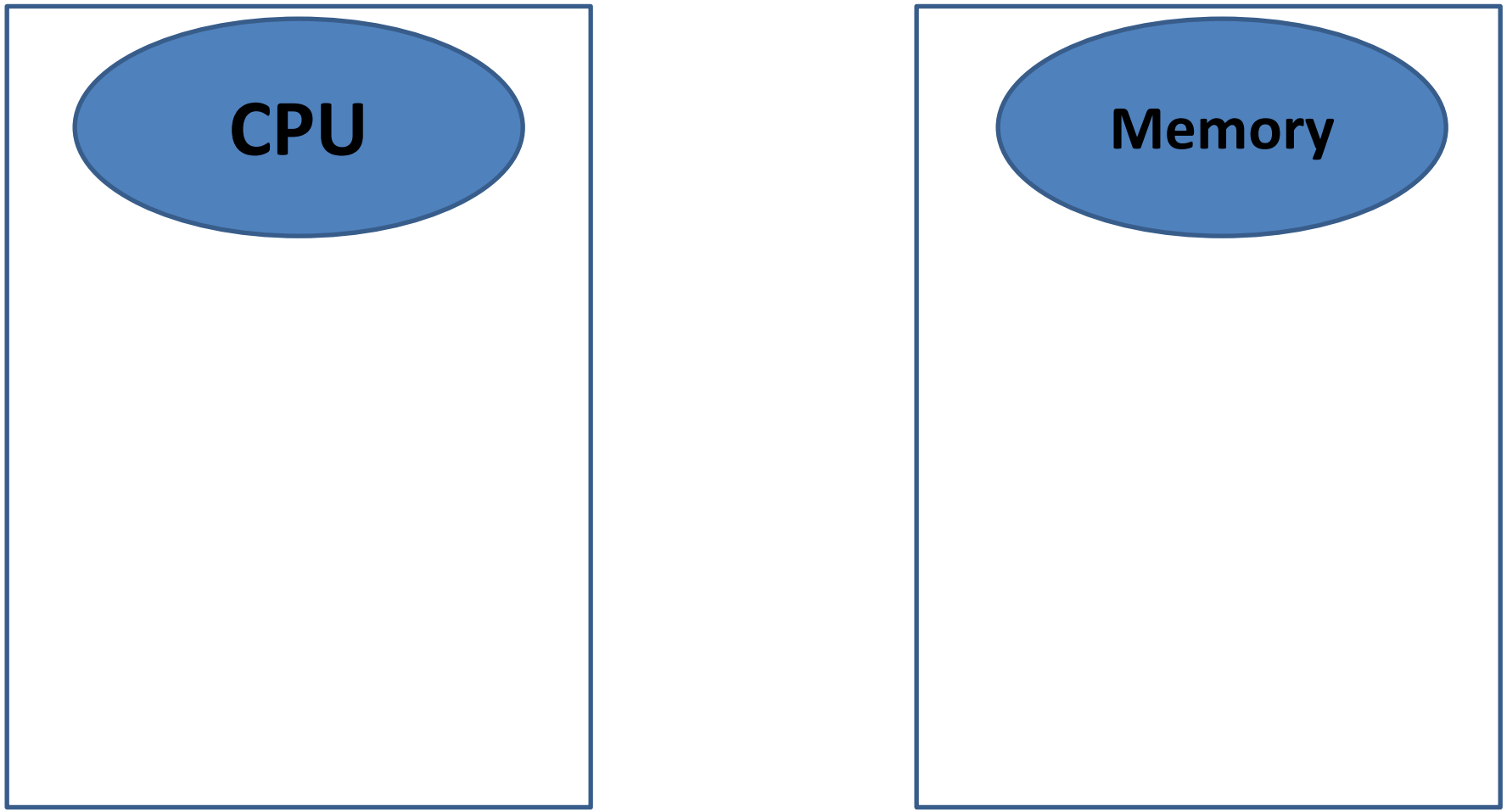
Symbiotic relationship



I. Hardware

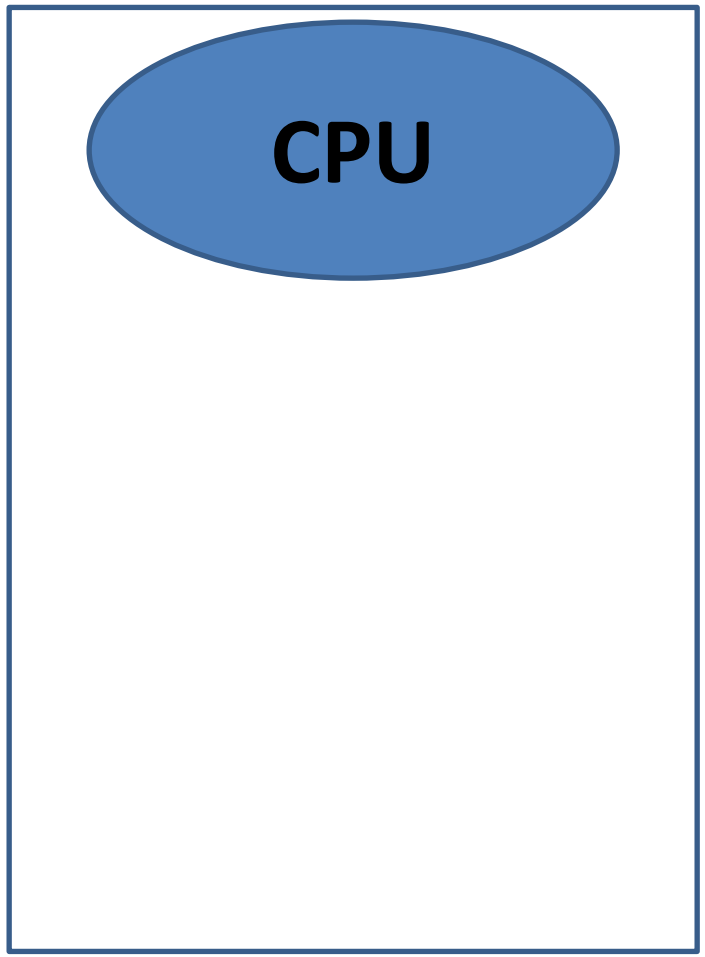
The **physical** parts of a computer

I. Hardware

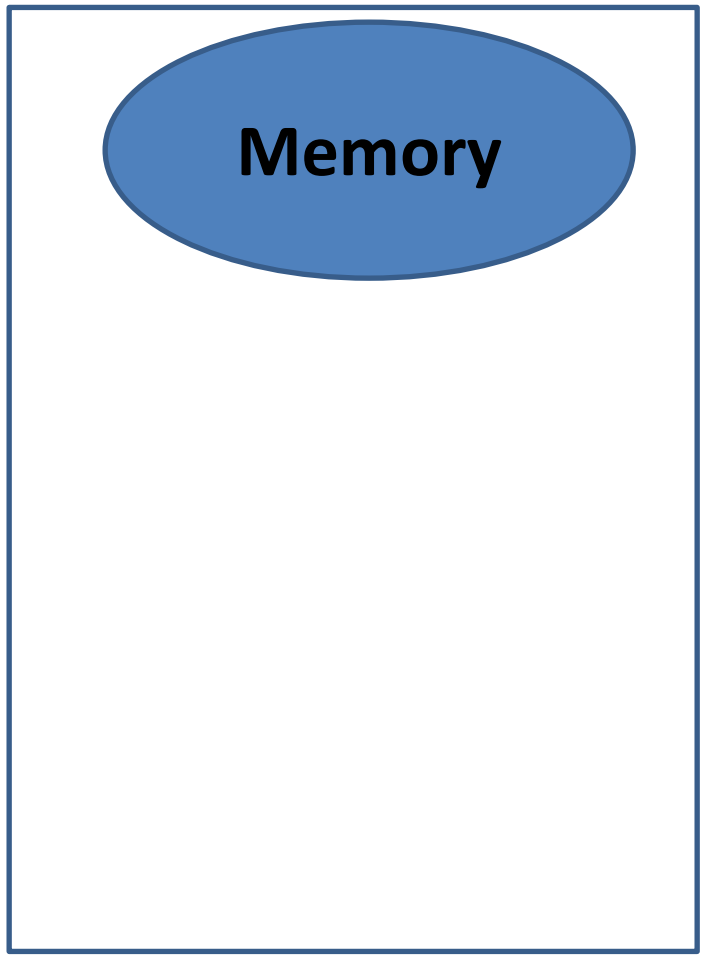


I. Hardware

Brain with no short-term memory

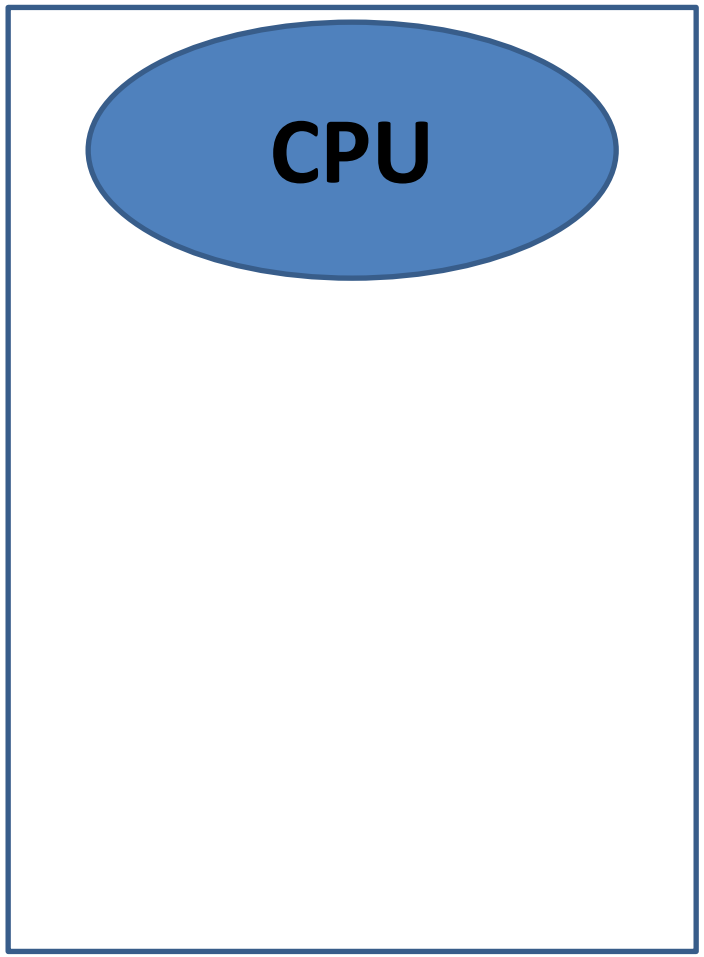


Scratch Pad



I. Hardware

Brain with no short-term memory



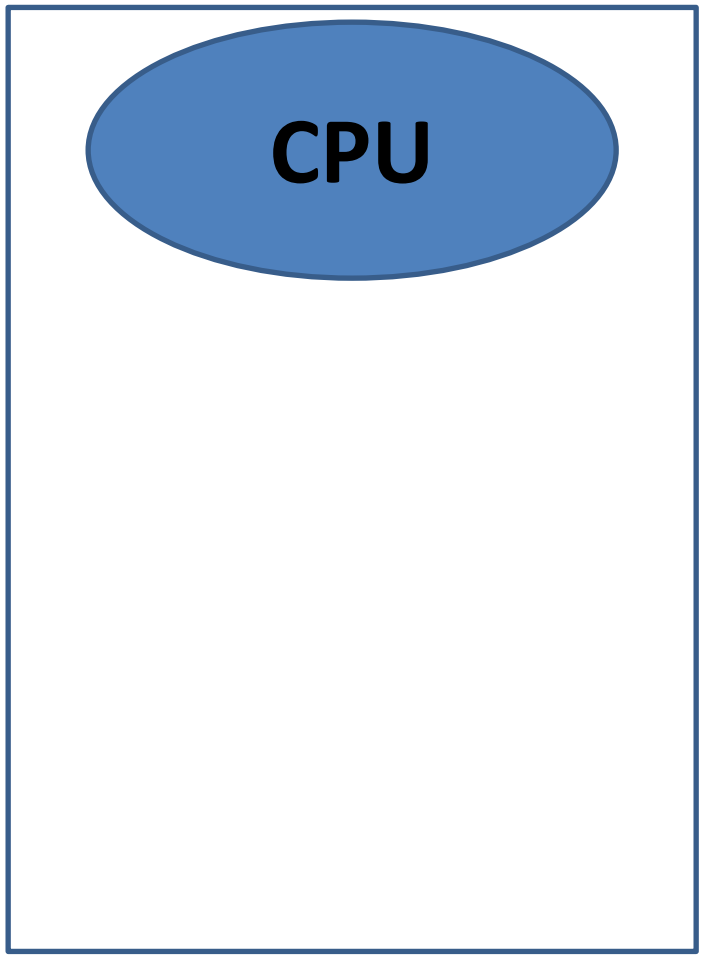
How does it "execute" instructions?

The CPU receives instructions from another part of the computer

00110110101101010001

I. Hardware

Brain with no short-term memory



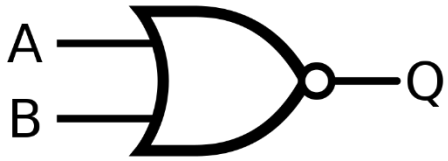
How does it "execute" instructions?

The CPU receives instructions from another part of the computer

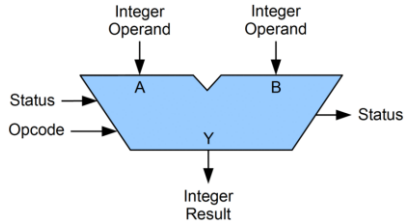
0011011010110101001

CPU deciphers these instructions by using

Electricity™

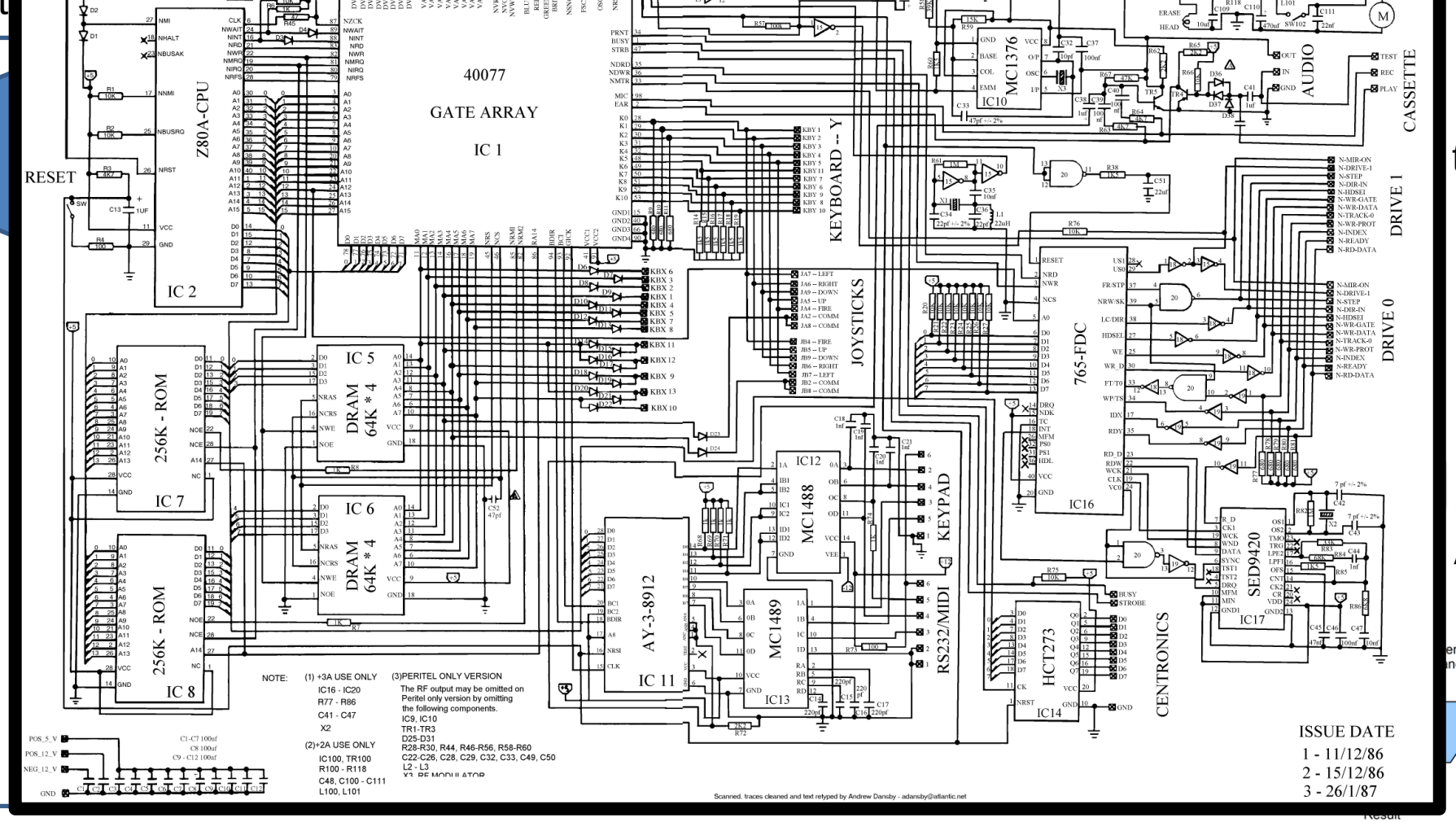


The CPU then executes the appropriate **operation** based on the instruction

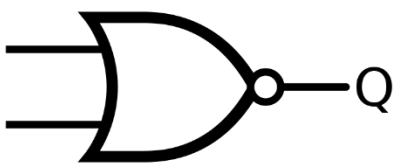


I. Ha

Brain wit



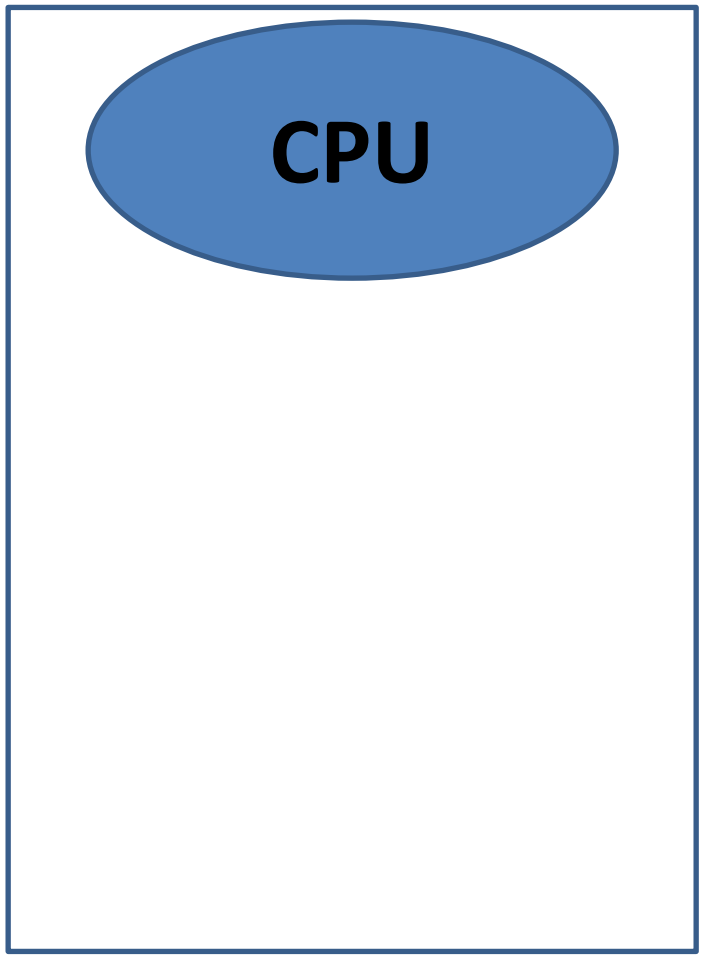
the computer



Status

I. Hardware

Brain with no short-term memory



How does it "execute" instructions?

The CPU receives instructions from another part of the computer

0011011010110101001

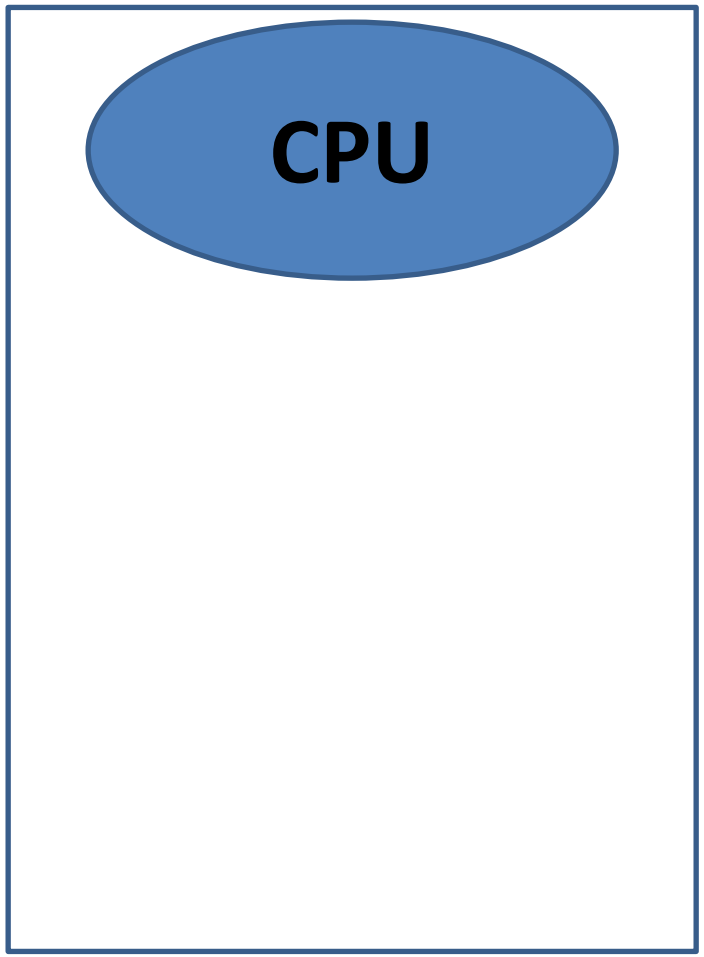
CPU deciphers these instructions by using

Electricity™

The CPU then executes the appropriate **operation** based on the instruction

I. Hardware

Brain with no short-term memory



The CPU then executes the appropriate **operation** based on the instruction

Data Transfer Instruction	
`MOV`	Move byte or word to register or memory
`IN, OUT`	Input/output byte or word
`LEA`	Load Effective Address
`PUSH, POP`	Push/Pop word on/from stack
Arithmetic and Logical Instructions	
`NOT`	Logical NOT of byte or word
`AND`	Logical AND of byte or word
`OR`	Logical OR of byte or word
`XOR`	Logical XOR of byte or word
`ADD, SUB`	Add, subtract byte or word
`INC, DEC`	Increment, decrement byte or word
`NEG`	Negate byte or word (two's complement)
`MUL, DIV`	Multiply, divide byte or word (unsigned)
Control Flow Instructions	
`JMP`	Unconditional jump
`JE, JNE`	Jump if equal/Jump if not equal
`LOOP`	Loop unconditional, count in `CX`, short jump to target address
`CALL, RET`	Call, return from procedure

Instructions consists of a small set (200-400) very basic operations

(Add stuff, move stuff, check a condition, etc)

I. Hardware

Brain with no short-term memory

CPU

Must decipher what instruction to execute

The CPU can add, subtract, logic, move stuff around

Scratch Pad

Memory

I. Hardware

Scratch Pad

Memory

Instructions are stored here and prepared for the CPU

Instruction1
Instruction2
Instruction3
Instruction4
..
..
...

When computer programs are executed, their instructions will eventually get stored in memory

Temporary storage

RAM (Random Access Memory)

I. Hardware

Brain with no short-term memory

CPU

Must decipher what instruction to execute

The CPU can add, subtract, logic, move stuff around

Scratch Pad

Memory

Temporary Storage

Sends instructions to the CPU

I. Hardware

Brain with no short-term memory

CPU

Must decipher what instruction to execute

The CPU can add, subtract, logic, move stuff around

Fetch Next Instruction

Decode and Execute

Scratch Pad

Memory

Temporary Storage

Sends instructions to the CPU

I. Hardware

Brain with no short-term memory

CPU

Must decipher what instruction to execute

The CPU can add, subtract, logic, move stuff around

Do this forever…

Fetch Next Instruction

Decode and Execute

Scratch Pad

Memory

Temporary Storage

Sends instructions to the CPU

I. Hardware

This happens very fast

… like REALLY fast

Brain with no short-term memory

Scratch Pad

CPU

Must decipher what instruction to execute

The CPU can add, subtract, logic, move stuff around

Do this forever…

Fetch Next Instruction

Decode and Execute

Memory

Temporary Storage

Sends instructions to the CPU

I. Hardware

This happens very fast

… like REALLY fast

Intel i7 = **3 BILLION** instructions per second

Brain with no short-term memory

CPU

Must decipher what instruction to execute

The CPU can add, subtract, logic, move stuff around

Do this forever…

Fetch Next Instruction

Decode and Execute

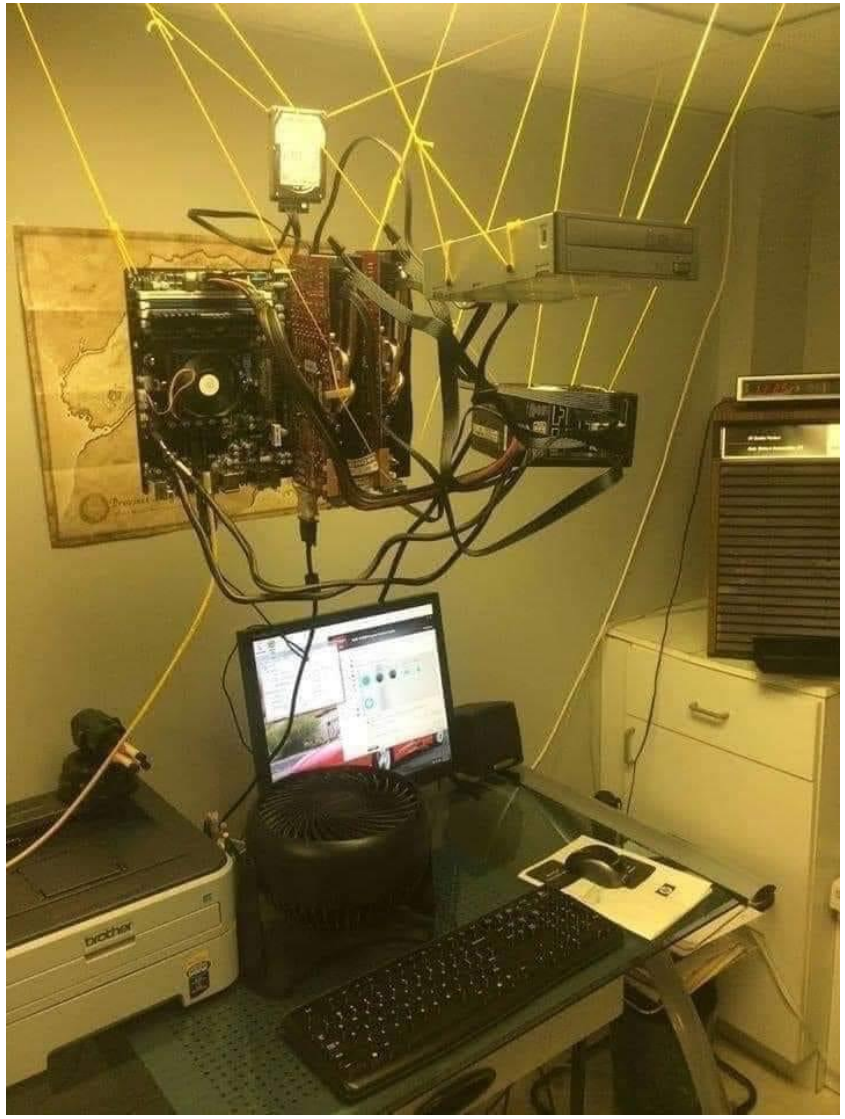
Scratch Pad

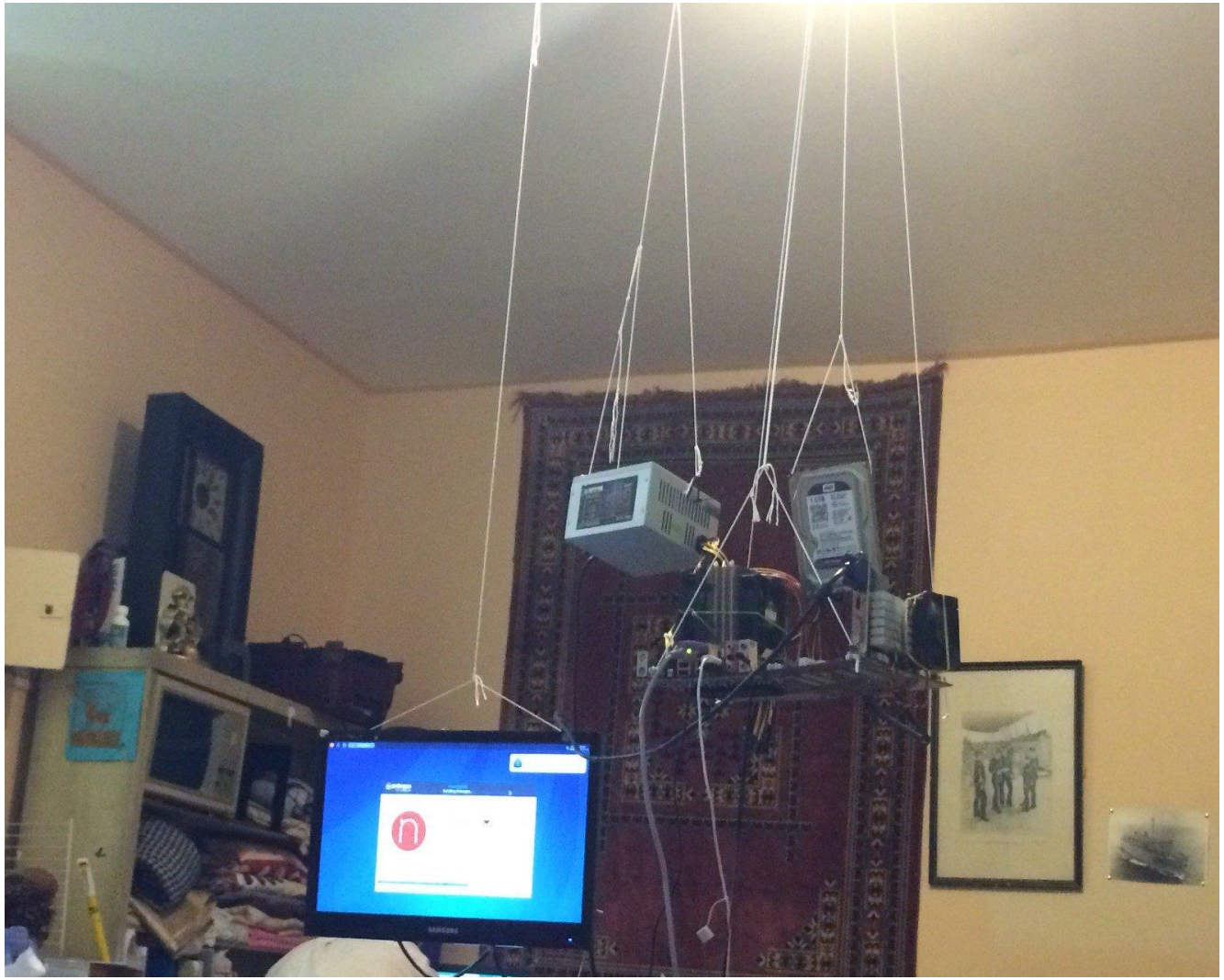
Memory

Temporary Storage

Sends instructions to the CPU

I. Hardware





I. Hardware

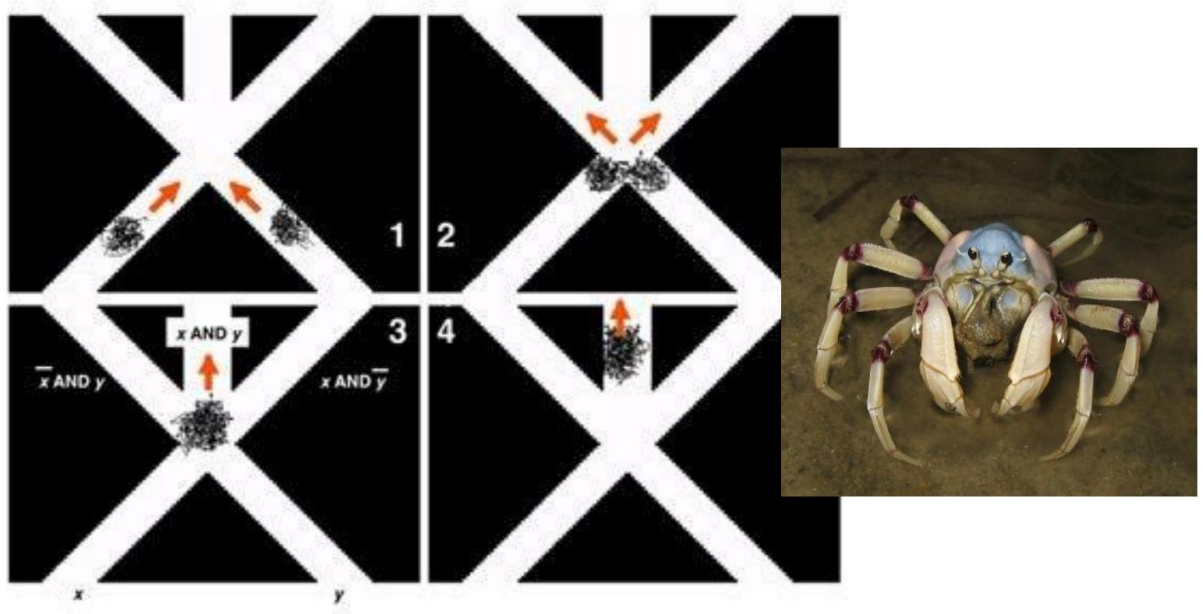
People have been able to create CPU components and fully functional, multi-core computers in games such as Minecraft

I. Hardware

WIRED STAFF BUSINESS 04.14.2012 03:20 PM

Computer Built Using Swarms Of Soldier Crabs

Computer scientists at Kobe University in Japan have built a computer that draws inspiration from the swarming behavior of soldier crabs. The computer is based on theories from the early 1980s that studies how it could be possible to build a computer out of billiard balls. Proposed by Edward Fredkin and Tommaso Toffoli, the mechanical [...]

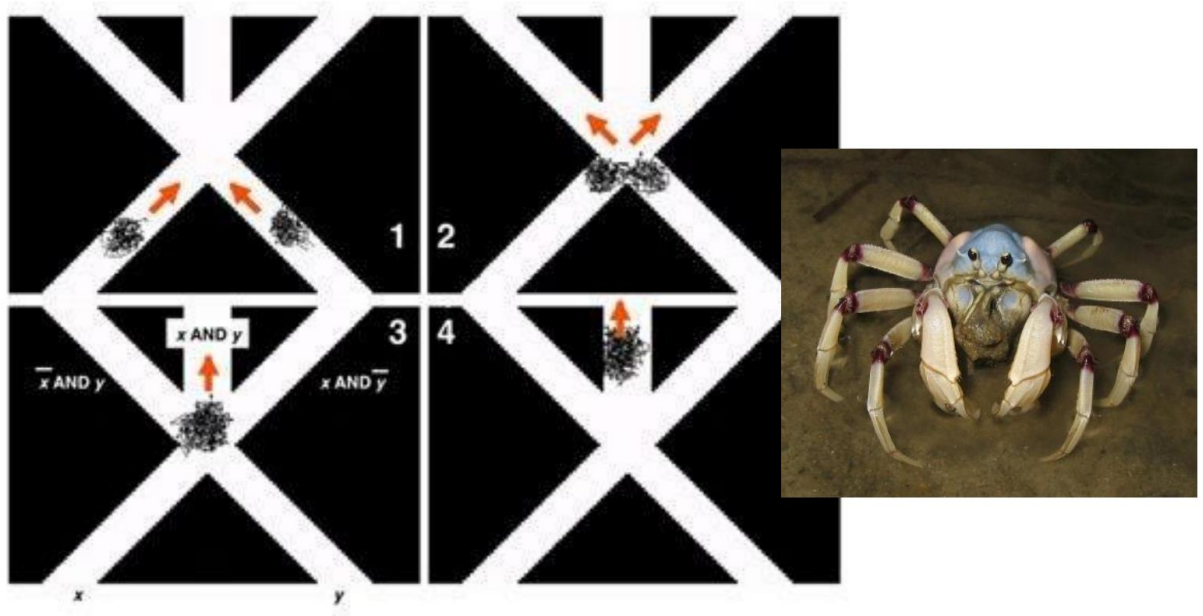


I. Hardware

WIRED STAFF BUSINESS 04.14.2012 03:20 PM

Computer Built Using Swarms Of Soldier Crabs

Computer scientists at Kobe University in Japan have built a computer that draws inspiration from the swarming behavior of soldier crabs. The computer is based on theories from the early 1980s that studies how it could be possible to build a computer out of billiard balls. Proposed by Edward Fredkin and Tommaso Toffoli, the mechanical [...]



This is very real

Robust Soldier Crab Ball Gate

Yukio-Pegio Gunji
Yuta Nishiyama
Department of Earth and Planetary Sciences
Kobe University
Kobe 657-8501, Japan

Andrew Adamatzky
Unconventional Computing Centre
University of the West of England
Bristol, United Kingdom

Soldier crabs *Mictyris guinotae* exhibit pronounced swarming behavior. Swarms of the crabs are tolerant of perturbations. In computer models and laboratory experiments we demonstrate that swarms of soldier crabs can implement logical gates when placed in a geometrically constrained environment.

1. Introduction

All natural processes can be interpreted in terms of computations. To implement a logical gate in a chemical, physical, or biological spatially extended medium, Boolean variables must be assigned to disturbances, defects, or localizations traveling in the medium. These traveling patterns collide and the outcome of their collisions are converted

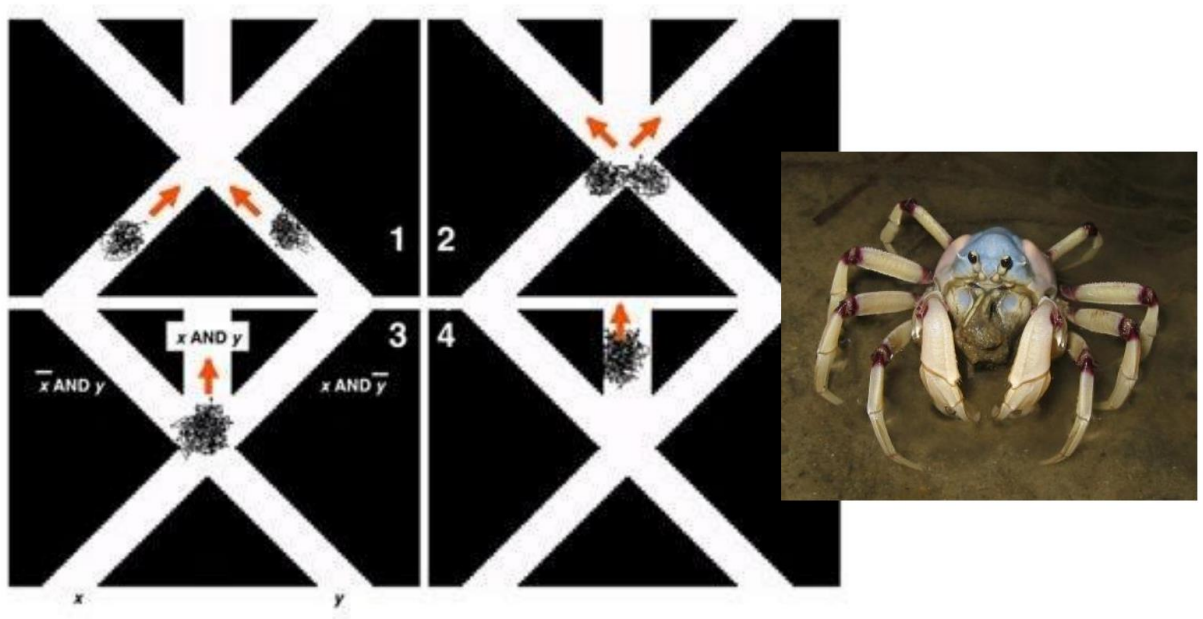
https://wpmedia.wolfram.com/uploads/sites/13/2018/02/20-2-2.pdf

I. Hardware

WIRED STAFF BUSINESS 04.14.2012 03:20 PM

Computer Built Using Swarms Of Soldier Crabs

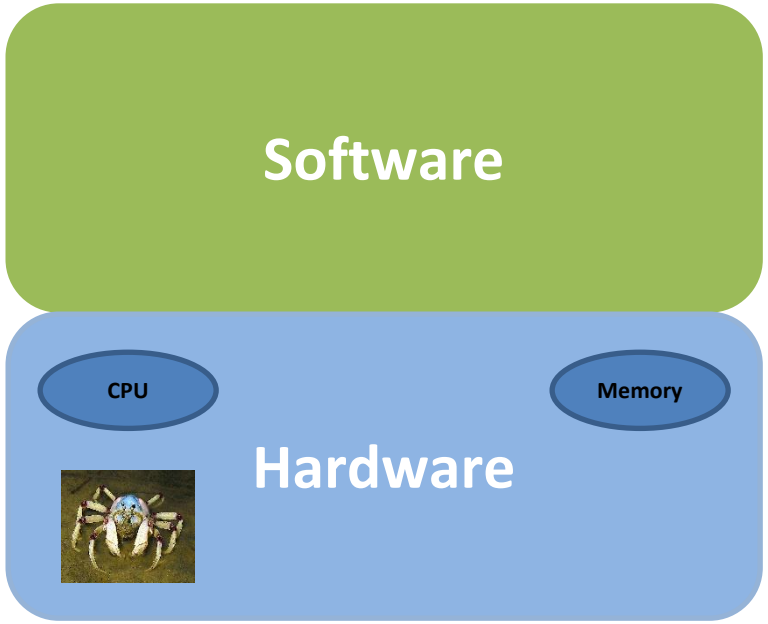
Computer scientists at Kobe University in Japan have built a computer that draws inspiration from the swarming behavior of soldier crabs. The computer is based on theories from the early 1980s that studies how it could be possible to build a computer out of billiard balls. Proposed by Edward Fredkin and Tommaso Toffoli, the mechanical [...]



(In theory) If you wanted to play Doom (1993) using a CPU made from soldier crabs, you would need 22 million crabs

(source: twitter)

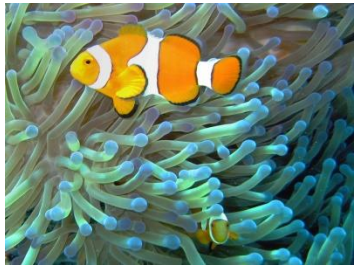
How does this happen?



From a high level, we will divide a computer system into two parts

I. **Hardware**
II. **Software**

Symbiotic relationship



Software

A sequence of instructions, or **program**, that tells the computer how to work






Software

A sequence of instructions, or **program**, that tells the computer how to work

Humans (computer programmers) write software

- So, do we have to write programs in 0s and 1s ???

Software

A sequence of instructions, or **program**, that tells the computer how to work

Humans (computer programmers) write software

- So, do we have to write programs in 0s and 1s ???

NO!!!

(thank goodness!!)

Software

We write programs in a **high-level** programming language

These are languages that are very easy for humans to read

Software

We write programs in a **high-level** programming language

These are languages that are very easy for humans to read

Software

```
#Basic Program

number = 7

if number > 0:
    print("This is a positive number")

print("Goodbye!")
```

A computer doesn't understand what this means…

We need to translate it to 0s and 1s

It eventually gets translated to binary by the **compiler***

*not true for every language

Software

```
#Basic Program

number = 7

if number > 0:
    print("This is a positive number")

print("Goodbye!")
```

```
10100001 10111100 10010011 00000100  10100001 10111100 10010011 00000100
00001000 00000011 00000101 11000000  00001000 00000011 00000101 11000000
10010011 00000100 00001000 10100011  10010011 00000100 00001000 10100011
11000000 10010100 00000100 00001000  11000000 10010100 00000100 00001000
```

A computer doesn't understand what this means…

We need to translate it to 0s and 1s

*not true for every language

Software

An **algorithm** is a recipe, or a sequence of steps, for solving a problem

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The quadratic formula is an **algorithm** for finding the roots of a quadratic equation

Software

An **algorithm** is a recipe, or a sequence of steps, for solving a problem

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The quadratic formula is an **algorithm** for finding the roots of a quadratic equation

We have algorithms for many different problems:

- Finding greatest common denominator between two values
- Searching and Sorting arrays
- Encrypting and compressing data
- Finding the shortest path in a network
- Medical diagnosis
- Scheduling jobs

Software

An **algorithm** is a recipe, or a sequence of steps, for solving a problem

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

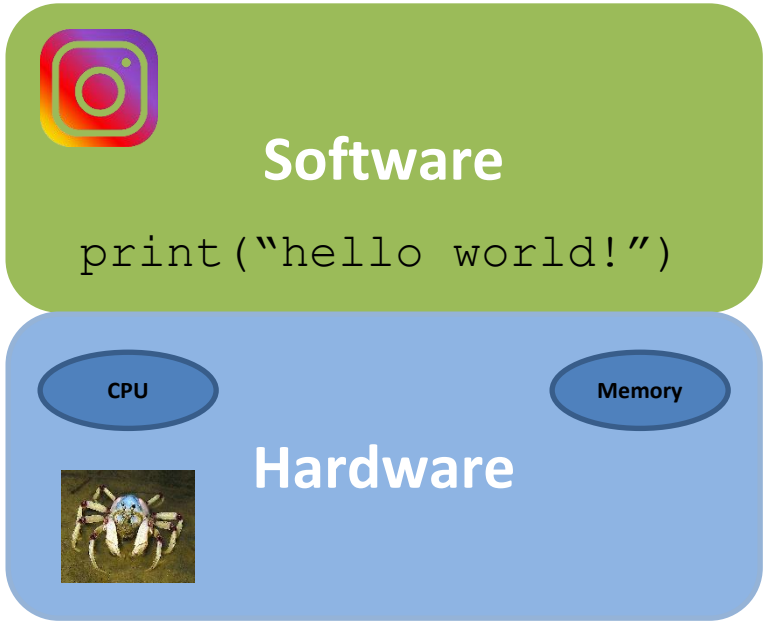
The quadratic formula is an **algorithm** for finding the roots of a quadratic equation

We implement algorithms in a **program,** which is a set of instructions that a computer can understand

```
# Solve the quadratic equation ax**2 + bx + c = 0

# import complex math module
import cmath

a = 1
b = 5
c = 6

# calculate the discriminant
d = (b**2) - (4*a*c)

# find two solutions
sol1 = (-b-cmath.sqrt(d))/(2*a)
sol2 = (-b+cmath.sqrt(d))/(2*a)

print('The solution are {0} and {1}'.format(sol1,sol2))
```

A Python **program** that uses the quadratic formula algorithm

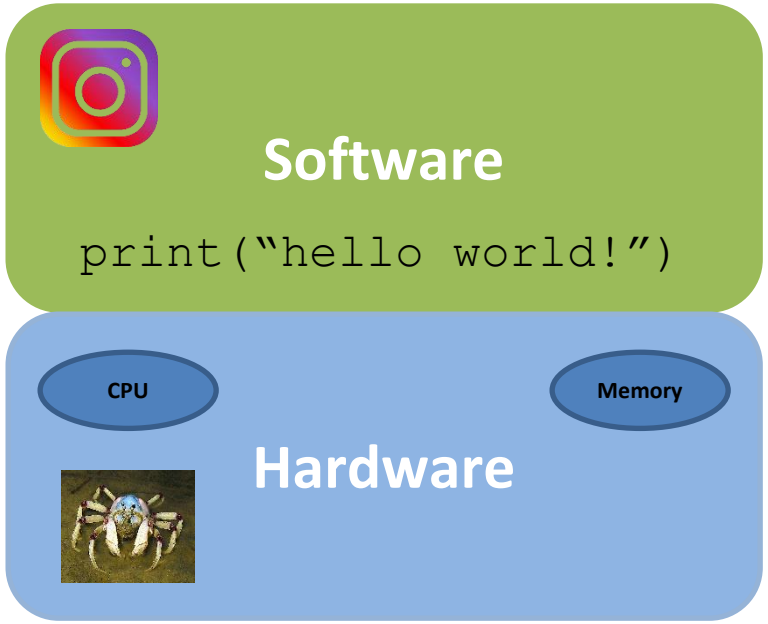
How does this happen?



From a high level, we will divide a computer system into two parts

I. **Hardware**
II. **Software**

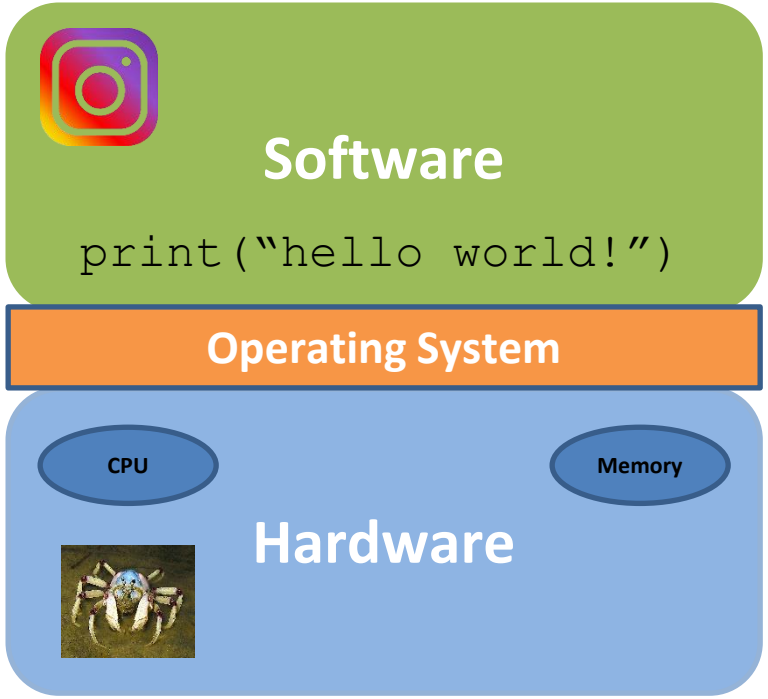
How does this happen?



From a high level, we will divide a computer system into two parts

I. **Hardware**
II. **Software**
III. **???**

How does this happen?



From a high level, we will divide a computer system into two parts

I. **Hardware**
II. **Software**
III. **Operating System**

Java

In this class, we will use **Java** as our programming language

Why do we need more than one programming language?

```
public void processData() {
   do {
      int data = getData();

      if (data < 0)
         performOperation1(data);
      else
         performOperation2(data);
   } while (hasMoreData());
}
```

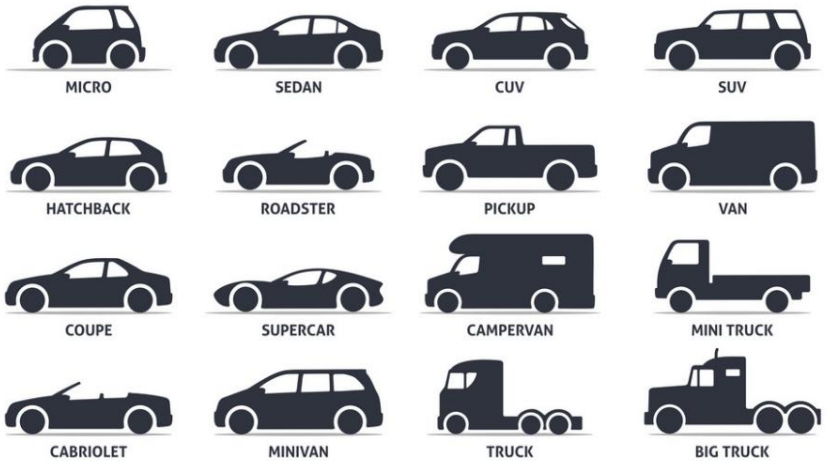
Java

In this class, we will use **Java** as our programming language

Why do we need more than one programming language?

```
public void processData() {
   do {
      int data = getData();

      if (data < 0)
         performOperation1(data);
      else
         performOperation2(data);
   } while (hasMoreData());
}
```

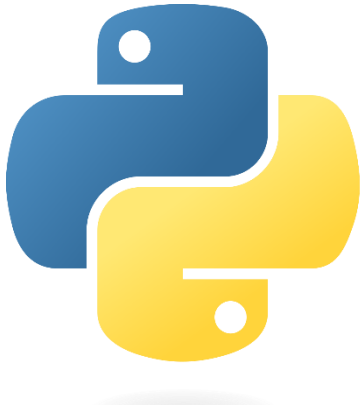
Different programming languages are better for different things



Java vs Python

Good for developing large, commercial, distributable software

Very flexible. Good for shorter jobs, data analysis, Web development,

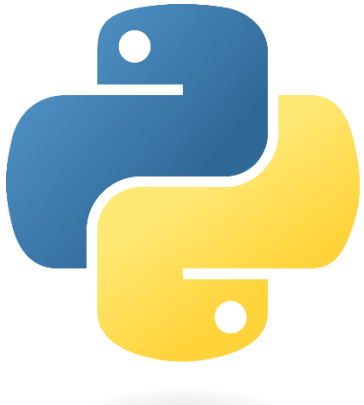


Java vs Python

Good for developing large, commercial, distributable software

Faster than Python

Very flexible. Good for shorter jobs, data analysis, Web development,



Slower than Java

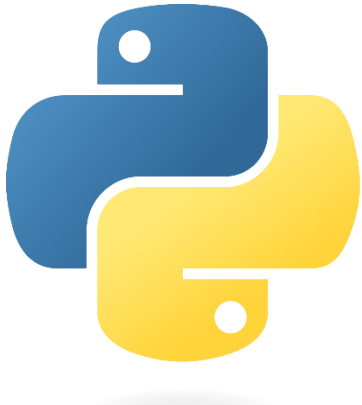
Java vs Python

Good for developing large, commercial, distributable software

Faster than Python

OOP Language

Very flexible. Good for shorter jobs, data analysis, Web development,



Slower than Java

Functional programming language

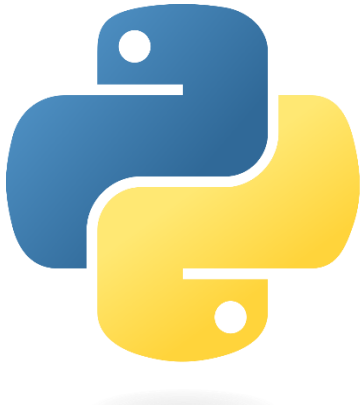
Java vs Python

Good for developing large, commercial, distributable software

Faster than Python

OOP Language

Very flexible. Good for shorter jobs, data analysis, Web development,



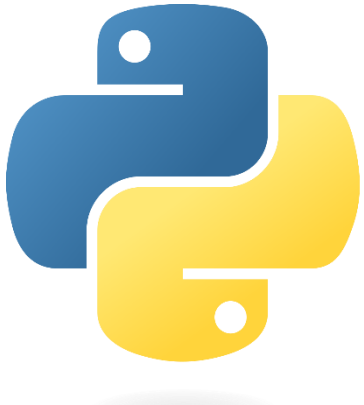
Slower than Java

Functional programming language

Java vs Python

Good for developing large, commercial, distributable software

Very flexible. Good for shorter jobs, data analysis, Web development,



Faster than Python

Slower than Java

OOP Language

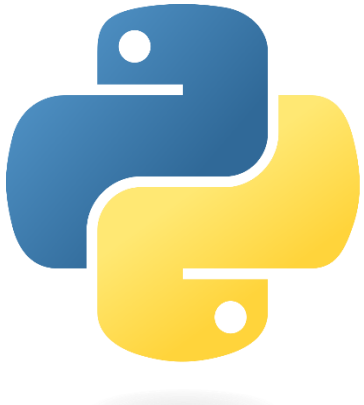
Functional programming language

Verbose (sigh)

Simple (but requires whitespace)

Java vs Python

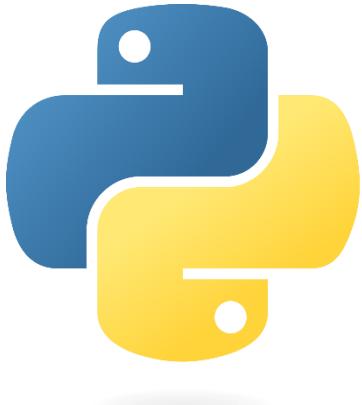
Good for developing large, commercial, distributable software	Very flexible. Good for shorter jobs, data analysis, Web development,
Faster than Python	Slower than Java
OOP Language	Functional programming language
Verbose (sigh)	Simple (but requires whitespace)
Static Typed	Dynamic Typed



Java vs Python

Good for developing large, commercial, distributable software

Very flexible. Good for shorter jobs, data analysis, Web development,



Faster than Python

Slower than Java

OOP Language

Functional programming language

Verbose (sigh)

Simple (but requires whitespace)

Static Typed

Dynamic Typed

Object Oriented Programming

```
class Student():

    def __init__(self,name,gpa,major):
        self.name = name
        self.gpa = gpa
        self.major = major

    def getName(self):
        return self.name

    def getGPA(self):
        return self.gpa

    def getMajor(self):
        return self.major
```

Object Oriented Programming

```
class Student():

    def __init__(self,name,gpa,major):
        self.name = name
        self.gpa = gpa
        self.major = major

    def getName(self):
        return self.name

    def getGPA(self):
        return self.gpa

    def getMajor(self):
        return self.major
```

We write **classes** that is a blueprint of something

Object Oriented Programming

```
class Student():

    def __init__(self,name,gpa,major):
        self.name = name
        self.gpa = gpa
        self.major = major

    def getName(self):
        return self.name

    def getGPA(self):
        return self.gpa

    def getMajor(self):
        return self.major
```

We write **classes** that is a blueprint of something

Classes consist of two important things:

1. Instance Fields/Attributes
2. Methods/Behaviors

Object Oriented Programming

```
class Student():

    def __init__(self,name,gpa,major):
        self.name = name
        self.gpa = gpa
        self.major = major

    def getName(self):
        return self.name

    def getGPA(self):
        return self.gpa

    def getMajor(self):
        return self.major
```

We write **classes** that is a blueprint of something

Classes consist of two important things:

1. Instance Fields/Attributes
2. Methods/Behaviors

This program does nothing until we start **creating objects**

Object Oriented Programming

```
class Student():

    def __init__(self,name,gpa,major):
        self.name = name
        self.gpa = gpa
        self.major = major

    def getName(self):
        return self.name

    def getGPA(self):
        return self.gpa

    def getMajor(self):
        return self.major

student1 = Student("Reese", 4.0, "Computer Science")

student2 = Student("Susan", 3.5, "Chemistry")
```

`student1` and `student2` are instances of the `Student` class.

Object Oriented Programming

```
class Student():

                                1.    2.    3.
    def __init__(self,name,gpa,major):
        self.name = name
        self.gpa = gpa
        self.major = major

    def getName(self):
        return self.name

    def getGPA(self):
        return self.gpa

    def getMajor(self):
        return self.major
```

`student1` and `student2` are instances of the `Student` class.

To create an object, we called the class name, and then passed the necessary **parameters/arguments**

This triggers the constructor, which will *create* our objects

```
                    1.       2.            3.
student1 = Student("Reese", 4.0, "Computer Science")

                    1.       2.          3.
student2 = Student("Susan", 3.5, "Chemistry")
```

Object Oriented Programming

```
class Student():

                                1.    2.    3.
    def __init__(self,name,gpa,major):
        self.name = name
        self.gpa = gpa
        self.major = major

    def getName(self):
        return self.name

    def getGPA(self):
        return self.gpa

    def getMajor(self):
        return self.major
```

`student1` and `student2` are instances of the `Student` class.

To create an object, we called the class name, and then passed the necessary **parameters/arguments**

```
                    1.       2.              3.
student1 = Student("Reese", 4.0, "Computer Science")

                    1.       2.          3.
student2 = Student("Susan", 3.5, "Chemistry")
```

Object Oriented Programming

```
class Student():

    def __init__(self,name,gpa,major):
        self.name = name
        self.gpa = gpa
        self.major = major

    def getName(self):
        return self.name

    def getGPA(self):
        return self.gpa

    def getMajor(self):
        return self.major

student1 = Student("Reese", 4.0, "Computer Science")

student2 = Student("Susan", 3.5, "Chemistry")
```

`student1` and `student2` are instances of the `Student` class.

An object is an encapsulation of information…

```
print(student1)
<__main__.Student object at 0x000002010BD0E0D0>
```

Printing/accessing an object doesn't do much on its own…

Object Oriented Programming

```
class Student():

    def __init__(self,name,gpa,major):
        self.name = name
        self.gpa = gpa
        self.major = major

    def getName(self):
        return self.name

    def getGPA(self):
        return self.gpa

    def getMajor(self):
        return self.major

student1 = Student("Reese", 4.0, "Computer Science")

student2 = Student("Susan", 3.5, "Chemistry")
```

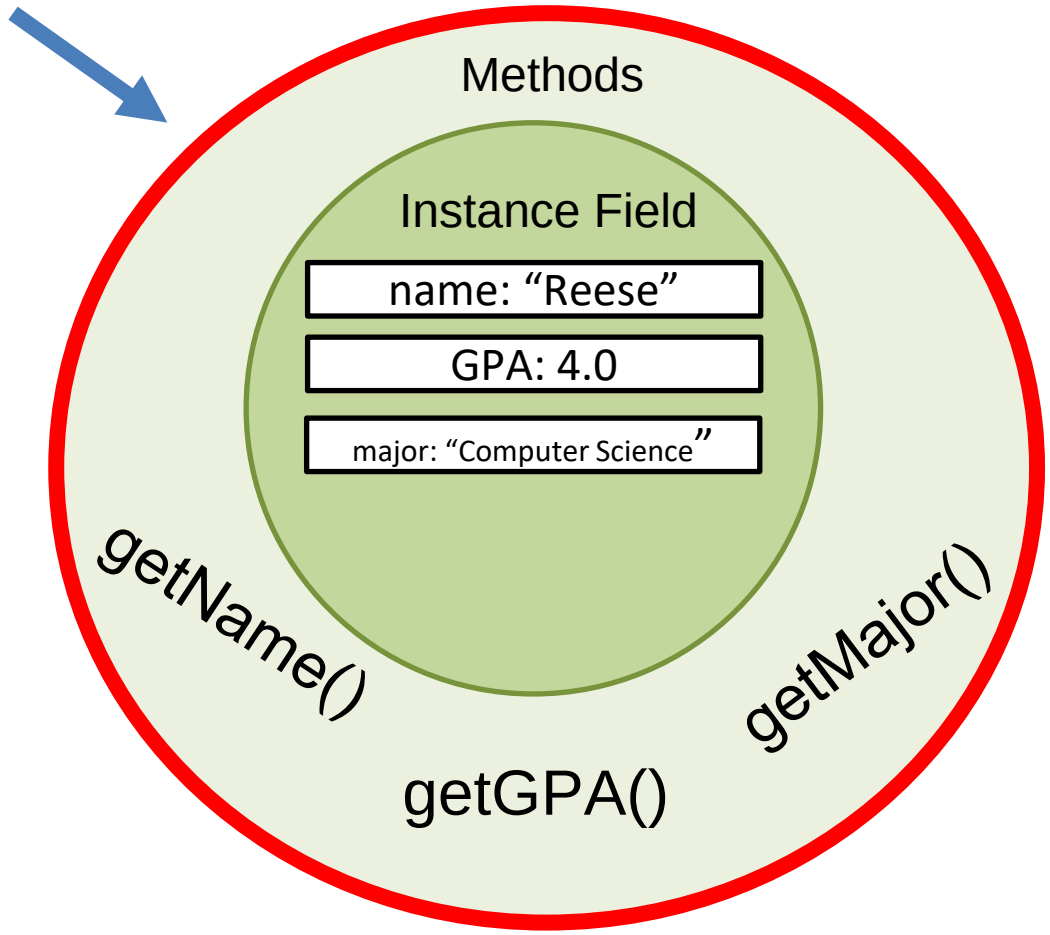
`student1` and `student2` are instances of the `Student` class.

```
print(student1.getName())
```

```
Reese
```

Object Oriented Programming

```
class Student():

    def __init__(self,name,gpa,major):
        self.name = name
        self.gpa = gpa
        self.major = major

    def getName(self):
        return self.name

    def getGPA(self):
        return self.gpa

    def getMajor(self):
        return self.major

student1 = Student("Reese", 4.0, "Computer Science")

student2 = Student("Susan", 3.5, "Chemistry")
```



Object Oriented Programming

```python
class Student():

    def __init__(self,name,gpa,major):
        self.name = name
        self.gpa = gpa
        self.major = major

    def getName(self):
        return self.name

    def getGPA(self):
        return self.gpa

    def getMajor(self):
        return self.major


student1 = Student("Reese", 4.0, "Computer Science")

student2 = Student("Susan", 3.5, "Chemistry")
```

```java
public class Student {

    String name;
    double GPA;
    String major;

    public Student(String name, double GPA, String major){
        this.name = name;
        this.GPA = GPA;
        this.major = major;
    }

    public String getName(){
        return this.name;
    }

    public double getGPA(){
        return this.GPA;
    }

    public String getMajor(){
        return this.major;
    }
```

```java
Student student1 = new Student("Reese",4.0,"Computer Science");

Student student2 = new Student("Susan",3.5,"Chemistry");
```

Java is only OOP,
all our code will be going inside of a class

IDE Install and creating a very basic Java Program