About Programming

#2

CS 0007
Introduction to
Computer Programming

Luís Oliveira

Summer 2020

Announcements

- Starting next class (?) We will be doing programming! :D
  - If you want to write code along me, have your computers with you
  - During recitation, you will make sure you have Java installed and working
- Let me know if you have issues

- Recommended readings:
  - Chapter 1.1-1.4 of the Java book
  - Chapter 1 of https://www.cs.hmc.edu/~cs5grad/cs5book.pdf
    - This is a book about programming, they use Python, but the introduction is really interesting
    - I even took a couple of their examples ☺

Why do we want to program?

- A programming language is a hammer
  - But it is only useful if we have a nail
  - We need a problem to solve!
- So why do we program? Well, for many reasons. But mostly:
  - We want to automate a solution
  - We want to remove human error
    - And introduce computer error ;)
  - Computers are faster
- Most people need computers to efficiently solve their problems
  - Analyse data, track the status of machines, solve complex maths, store large volumes of data, simulate and model complex systems (how do chemicals react)

Can computers solve anything?

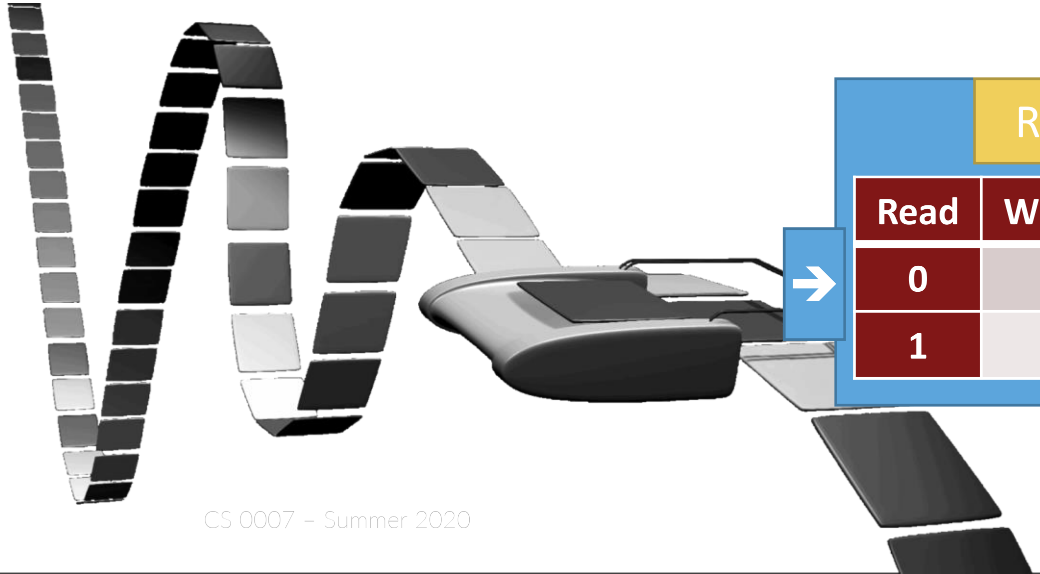
Nope

Alan Turing - Father of Computer Science

- Alan Turing (1912-1954)
  - Mathematician and cryptanalyst.
- Devised the scheme that broke the Enigma code in World War II
  - Bombe
- Published a thesis that provided a model of a computational machine
  - The Turing Machine

The Turing machine

- **Has infinite memory represented by a single tape.**
  - A head moves along the tape and can read and write values.
    - The movement (left or right) is based upon the value read and the state of the machine.
    - **Over simplified definition of state**: A rulebook ☺
- **A general, formal description of a computer.**
  - Theoretical and simple: all physical machines live up to this formal description.



Rulebook 1

Read	Write	Move	Next
0	1	←	20
1	0	→	12

Rulebook 20

Read	Write	Move	Next
0	0	→	15
1	1	→	15

The Turing machine

- **The Turing machine performs terribly**
  - But it models the behaviour of every modern machines
    - Models: Simulates the behaviour of something. Often by abstracting details.
  - The machine reads data, compares that data to make a decision:
    1. what data is written and
    2. which direction it moves the head

- **Everything that can be computed, is computed by a Turing machine**
  - Turing-complete: Can do anything a Turing machine can do
  - So, can we use a Turing machine to simulate a Turing machine
    - Well, yes we can!
      - (Puts tin-foil hat on)

Will it stop?

- **But can everything be computed?**
  - E.g.: Can we even prove for **any** program if it’ll halt (stop)?
- **Turing proved that no, we cannot do that:**
  - Are There Problems That Computers Can't Solve?
    - https://www.youtube.com/watch?v=eqvBaj8UYz4
    - Thanks Tom Scott for the perfect timing ☺
- **So there are some problems that computers cannot solve**
  - But in this course, we’ll focus on those it can solve

And programming languages?

- Programming languages model this theoretical machine not the physical machine.
  - They assume infinite memory!
    - But you only have to guarantee that you have enough!
  - They assume infinite time!
  - The programmer, not the language, imposes and considers any extra limitations.
    - When we hit the limitations of the physical machine it crashes ☹

- Most (all) Programming languages are Turing-complete
  - Programming languages can compute anything computable

Algorithms

Delicious recipies!

Solving problems

- Presented with a computational problem, we need to find a solution
  - An **Algorithm**: A sequence of steps that carry out a task.
    - E.g. order your music collection by album, find webpages using the keywork “pumpkin pie”

The shampoo algorithm

1. Lather
2. Rinse
3. Repeat

The shampoo algorithm – better!

- The shampoo algorithm

1. Lather
2. Rinse
3. Repeat 1→2

- The shampoo algorithm

1. Lather
2. Rinse
3. Repeat until clean

- Here are the basics of the Turing machine
  - States
    - What are we doing now?
  - Read/write
    - Read (check status of hair)
    - Write (put on shampoo)
  - Make decisions
    - Do we repeat?

Different hammers for different nails

- Problems usually have several correct solutions ➔ different algorithms
  - Some are faster, some are smaller (less steps)
  - Some are better, some are worst
    - Is faster/smaller better or worst? → We'll discuss this later ☺
- They are often compared to recipes
  - Ingredients → data
  - Pumpkin pie recipe from https://www.cs.hmc.edu/~cs5grad/cs5book.pdf

1. Mix 3/4 cup sugar, 1 tsp cinnamon, 1/2 tsp salt, 1/2 tsp ginger and 1/4 tsp cloves in a small bowl.
2. Beat two eggs in a large bowl.
3. Stir 1 15-oz. can pumpkin and the mixture from step 1 into the eggs.
4. Gradually stir in 1 12 fl. oz. can evaporated milk into the mixture.
5. Pour mixture into unbaked, pre-prepared 9-inch pie shell.
6. Bake at 425◦F for 15 minutes.
7. Reduce oven temperature to 350◦F.
8. Bake for 30–40 minutes more, or until set.
9. Cool for 2 hours on wire rack.

Assuming we know how to measure, crack eggs, stir, etc.

Computer algorithms are not much different

- Let's replace Pie by π
  - Example also from https://www.cs.hmc.edu/~cs5grad/cs5book.pdf

1. Draw a square that is 2 by 2 feet.
2. Inscribe a circle of radius 1 foot (diameter 2 feet) inside this square.
3. Grab a bucket of *n* darts, move away from the dartboard, and put on a blindfold.
4. Take each dart one at a time and for each dart:
   a) With your eyes still covered, throw the dart randomly (but assume that your throwing skills ensure that it will land somewhere on the square dartboard).
   b) Record whether or not the dart landed inside the circle.
5. When you have thrown all the darts, divide the number that landed inside the circle by the total number, n, of darts you threw and multiply by 4. This will give you your estimate for π.

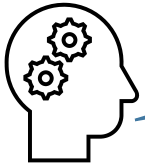
(Please don't try this at home)

Programming languages

Implementing algorithms

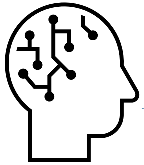
- To implement an algorithm we write a program
  - Algorithm: The list of instructions
  - Program (code): The software. Implementation of the algorithm
- And we need to learn a programming language
  - Programming language: The vocabulary and syntax rules.
  - We'll use Java. But the ideas are transferable to other programming languages.
- Why Java?
  - 1 → Java is a High-Level Language!
  - 2 → Java is portable

Different language levels



x=x+1

High-level language
Written by humans → Abstracted



```
lw   $t0, x
addi $t0, $t0, 1
sw   $t0, x
```

Assembly language
Written by humans→ No abstraction


```
10011101100110011001111101111001
11011110110010111011101001111001
10011100100110110001101111111011
```

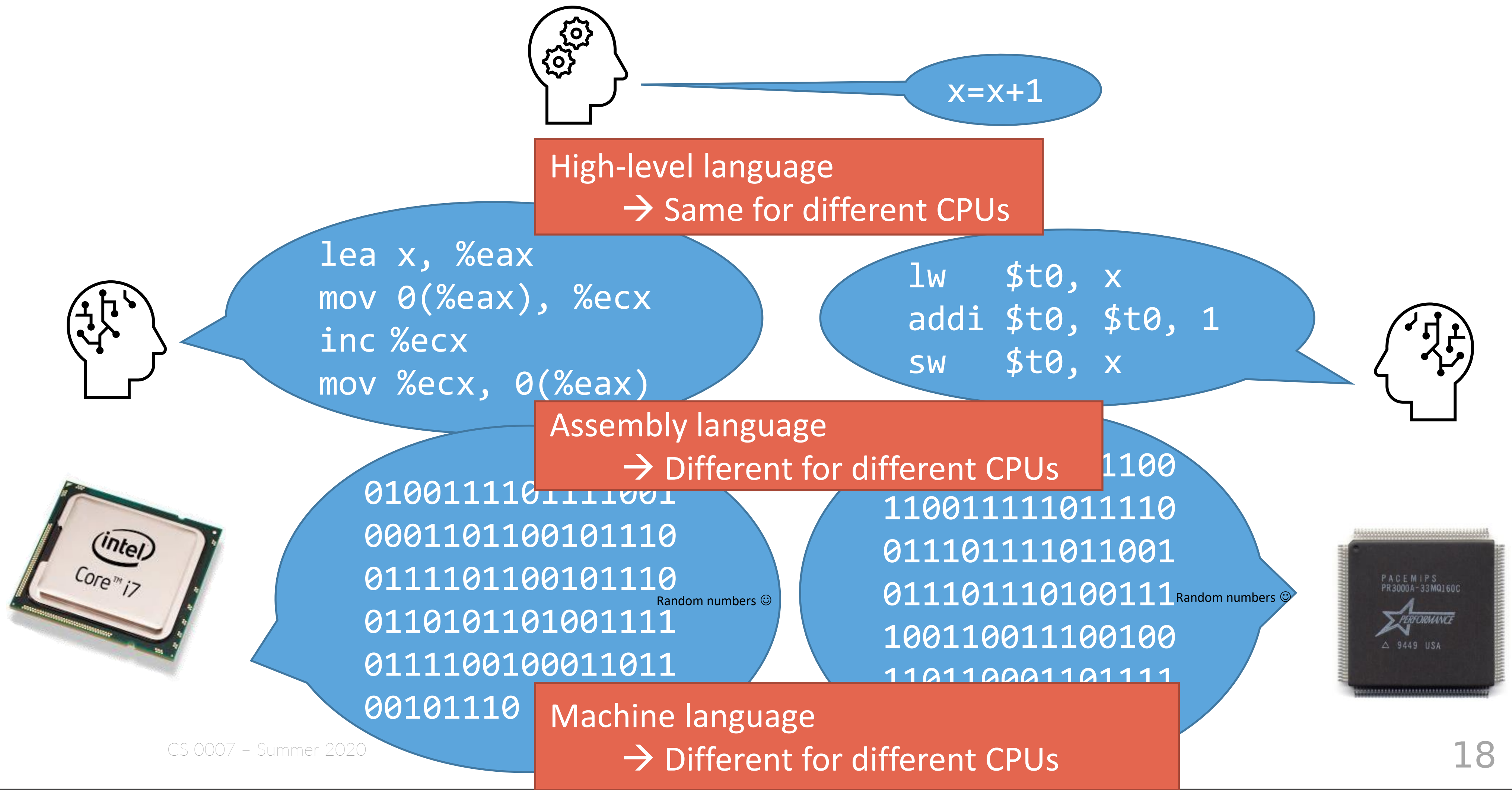
Random numbers ☺

Machine language
Spoken by the CPU→ Binary

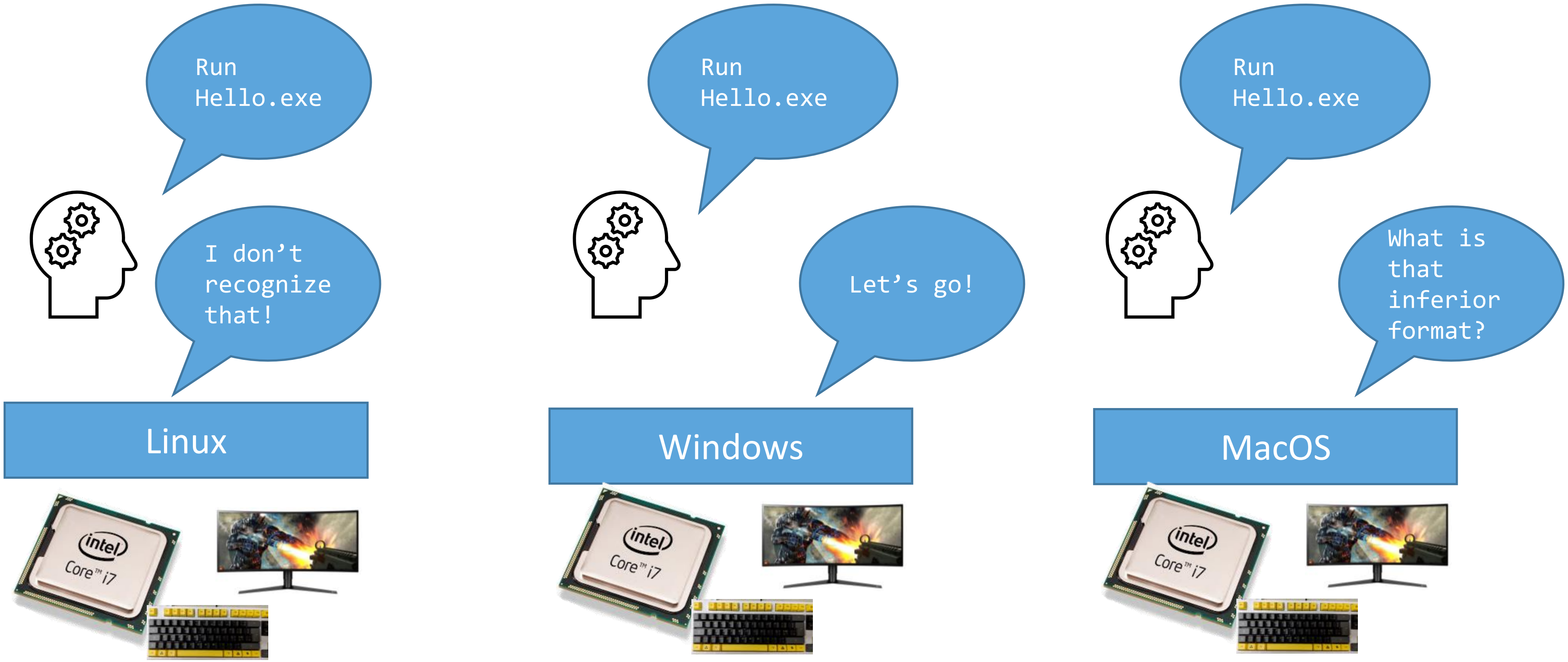
Easier to read

Closer to the hardware

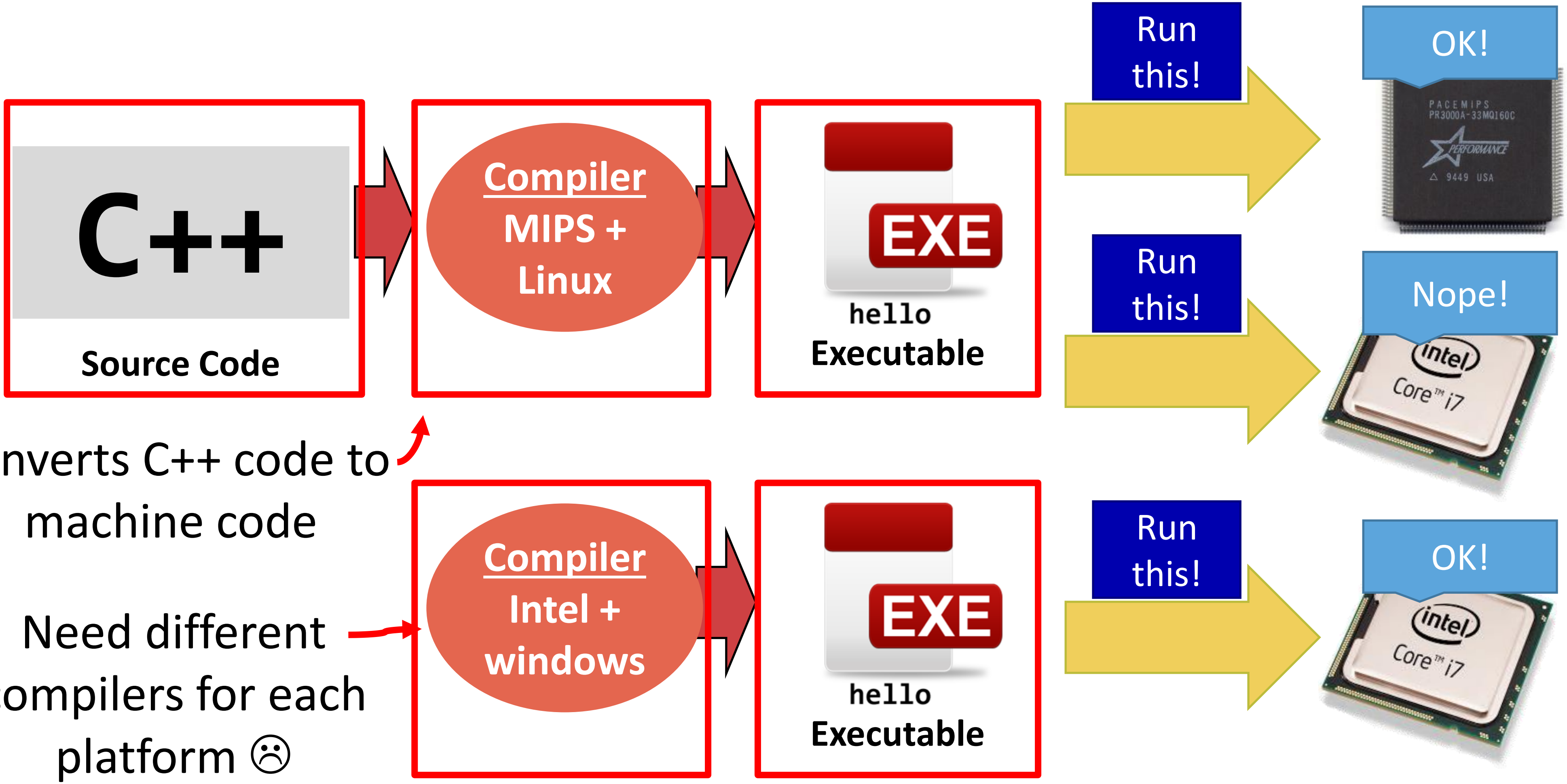
Different language levels



Different platforms



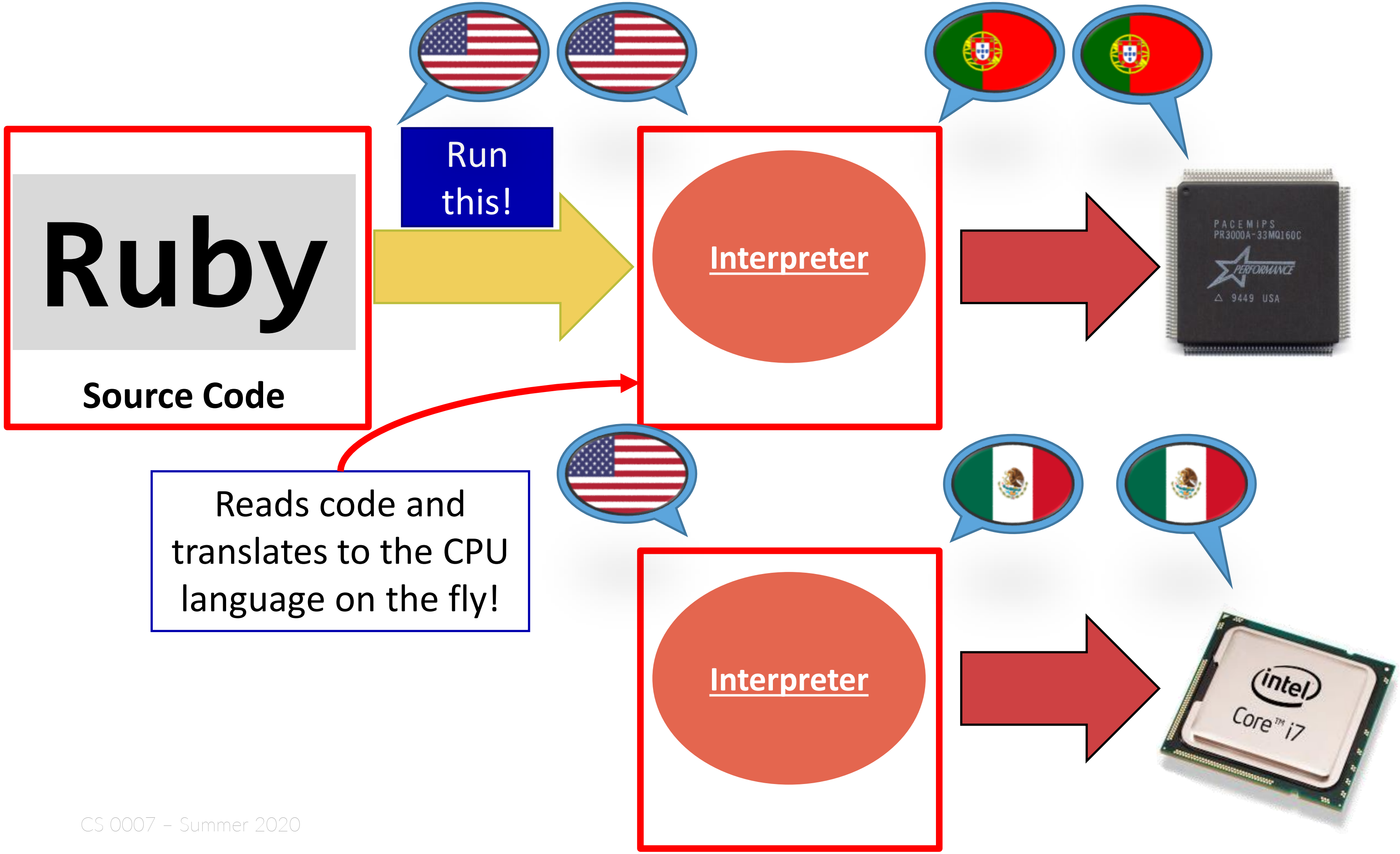
Some languages are compiled



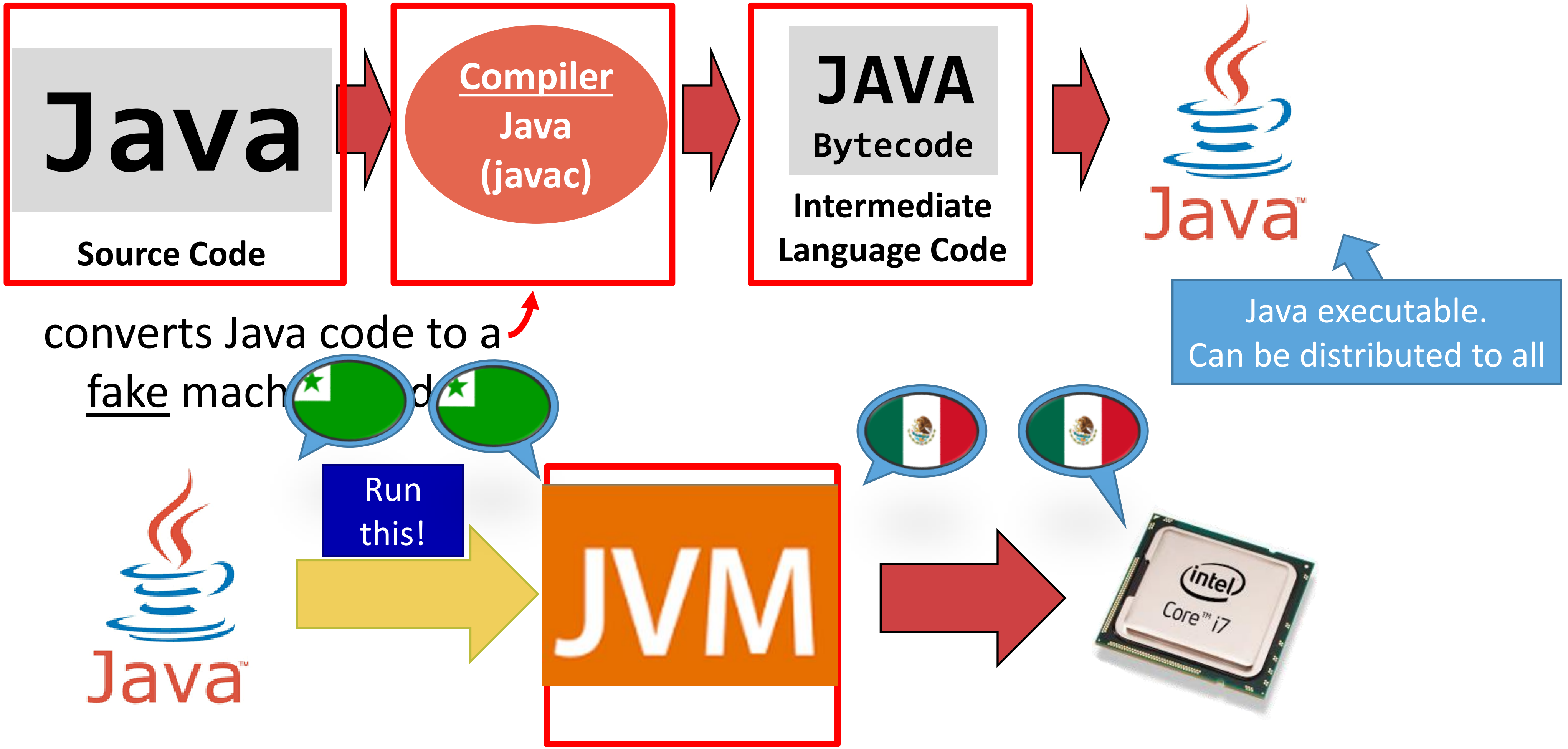
converts C++ code to machine code

Need different compilers for each platform ☹

Some languages are interpreted



Some languages are compiled and interpreted



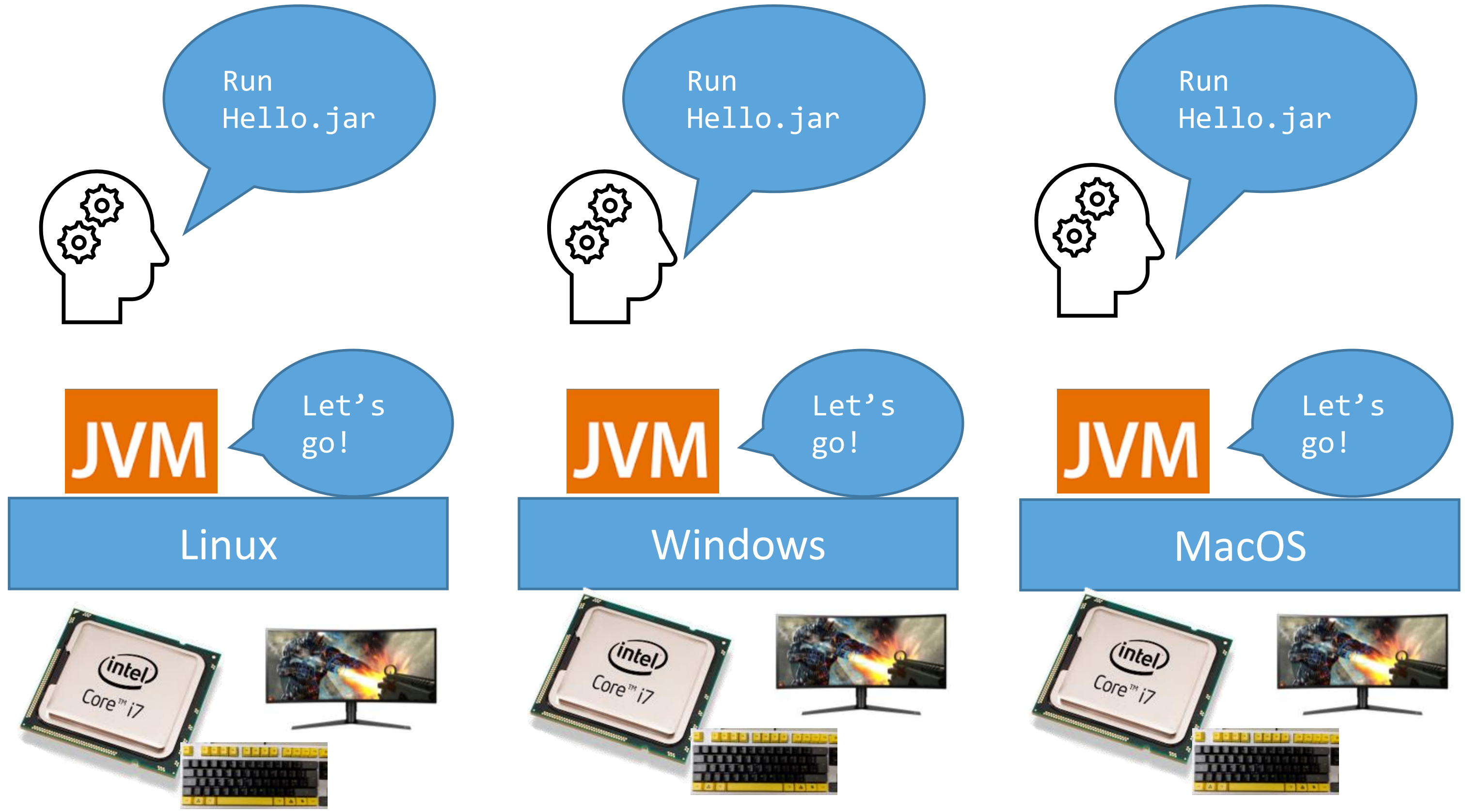
What’s the big deal

Java

Developer compiles

JVM

User installs interpreter



Picobot

Let's start programming :D