University of British Columbia
CPSC 111, Intro to Computation
2009W2: Jan-Apr 2010

Tamara Munzner

Hardware, Memory, Languages

Lecture 2, Wed Jan 6 2010

borrowing from slides by Kurt Eiselt, Wolfgang Heidrich, Alan Hu

http://www.cs.ubc.ca/~tmm/courses/111-10

News

- reminder: no class next time: this Friday Jan 8!

- UBC CS news

Events this week

How to Prepare for the Tech Career Fair

Date: Wed. Jan 6
Time: 5 – 6:30 pm
Location: DMP 110

Resume Writing Workshop (for non-coop students)

Date: Thurs. Jan 7
Time: 12:30 – 2 pm
Location: DMP 201

CSSS Movie Night

Date: Thurs. Jan 7
Time: 6 – 10 pm
Location: DMP 310
Movies: "Up" & "The Hangover" (Free Popcorn & Pop)

Drop-In Resume Edition Session

Date: Mon. Jan 11
Time: 11 am – 2 pm
Location: Rm 255, ICICS/CS Bldg

Industry Panel

Speakers: Managers from Google, IBM, Microsoft, TELUS, etc.
Date: Tues. Jan 12
Time: Panel: 5:15 – 6:15 pm; Networking: 6:15 – 7:15 pm
Location: Panel: DMP 110; Networking: X-wing Undergrad Lounge

Tech Career Fair

Date: Wed. Jan 13
Time: 10 am – 4 pm
Location: SUB Ballroom

Reading This Week

- Ch 1.1 - 1.2: Computer Anatomy

Correction / Recap: Prerequisites

- Mathematics 12 is the prerequisite
  - or any math course at UBC
  - if you have not taken it you will be dropped from the course
  - see CS advisors if you need prerequisite waived because of equivalent work
- current stuff
  - you cannot get credit for both 111 and new 110 course
  - you cannot get credit for 101 if you take it after or concurrently with 111
    - **you CAN get credit for 111 if you take it after 101!**

Recap: Processes, Procedures, and Programs

- process: what happens when a computer follows a procedure - it's a procedure in execution

- procedure: collection of instructions in some meaningful order that results in useful behavior on behalf of the device that executes the instructions

- program: when instructions are written in symbolic language that can be executed by a computer

Recap: Procedures and Algorithms

Here's why we get frustrated when we start to learn to write programs to make computers do stuff:

An algorithm is

- a finite procedure
- written in a fixed symbolic vocabulary
- governed by precise instructions
- moving in discrete steps, 1, 2, 3, ...
- whose execution requires no insight, cleverness, intuition, intelligence, or perspicuity
- and that sooner or later comes to an end

We don't have a lot of practice at being stupid!

Why Being Precise/Stupid Isn't Easy

- human languages are very different from computer languages: they're ambiguous
  - humans bring huge amounts of knowledge to understanding meaning of sentence
  - we apply it automatically and unconsciously
    - many meanings per word
    - sentence structure
    - context of conversation
    - how the world workd
    - how language is used
  - you count on listener to disambiguate without even noticing
  - we can get away with relatively short and imprecise sentences

Why Being Precise/Stupid Isn't Easy

- imagine a world where there is no ambiguity
  - that's computer programming!

  - everyone starts out imprecise
  - everyone gets frustrated while learning this stuff
    - you are not alone
    - you can succeed at this

Physical Hardware

- “Computer science is how to harness the physical world to help us think.” – Alan Hu

- harnessing the physical world to help us think
  - how to get things that have computational behavior?
  - technology dependent:
    - sticks, gears, relays, vacuum tubes, transistors, DNA,…
  - how to control that behavior to do interesting things

Computer Design

- it's hard to figure out how to make things do computation
- all digital computers for over 50 years have had:
  - same basic organization
  - binary representation of data
  - numerically addressed memory
  - fetch-decode-execute operation cycle

- we'll only have a brief glance here

Introduction to Computer Hardware

- Objectives:
  - to identify and explain the purpose of core hardware components
  - to understand the way data is represented in memory

- Understanding the hardware that runs our programs can help us understand the programs' behavior, especially when they misbehave.

Computer Hardware Overview

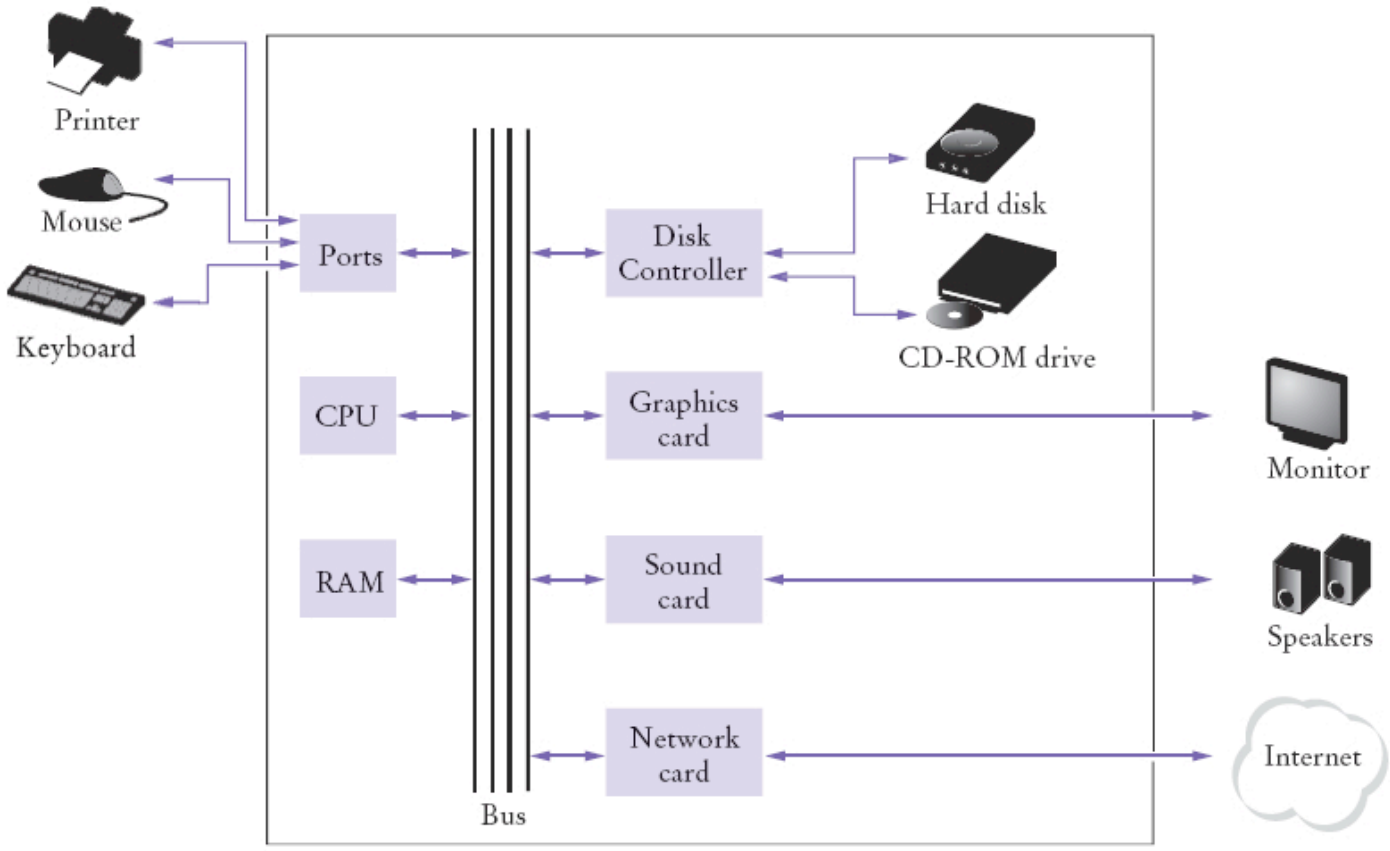
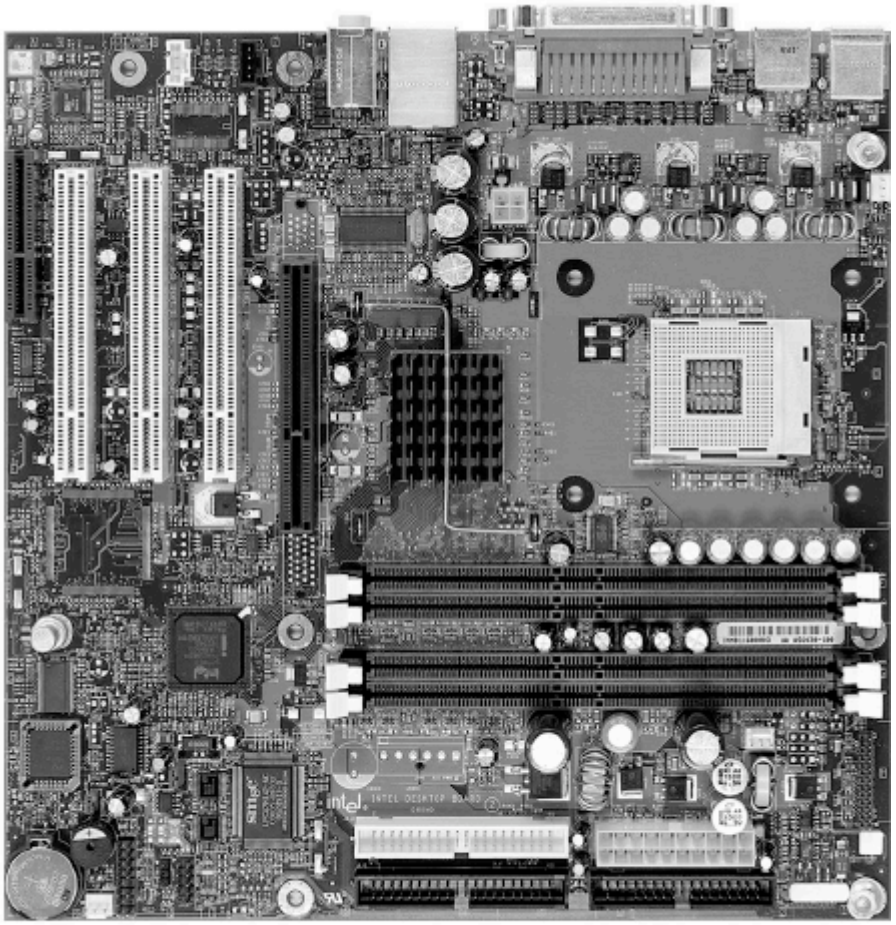


Figure 5 Schematic Diagram of a Computer

Computer Hardware Overview



Binary Data Representation

- All programs and data on a computer are represented using only symbols 0 and 1
- This simple binary system is encoded in all of our digital hardware devices:
  - Magnetic disks: magnetic material can be polarized to one of two extremes (north or south) to represent a 0 or a 1.
  - Memory: each byte consists of 8 bits; each bit is a kind of electronic switch that is either off or on representing a 0 or a 1.

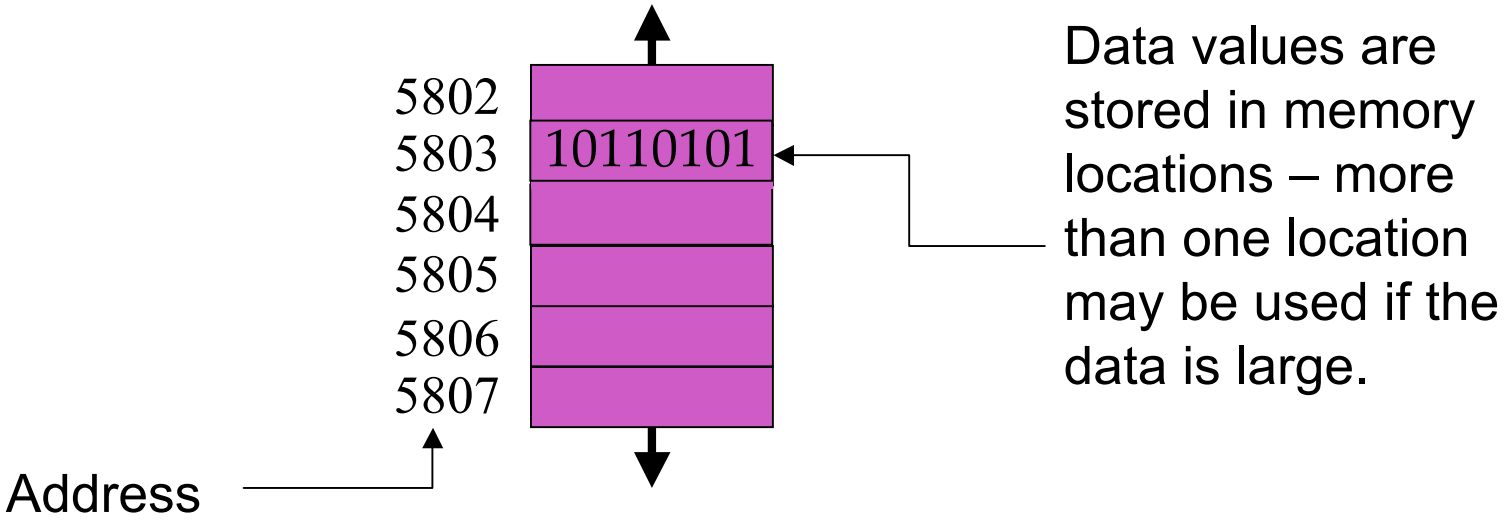
Memory

- Some of computer programming is resource management

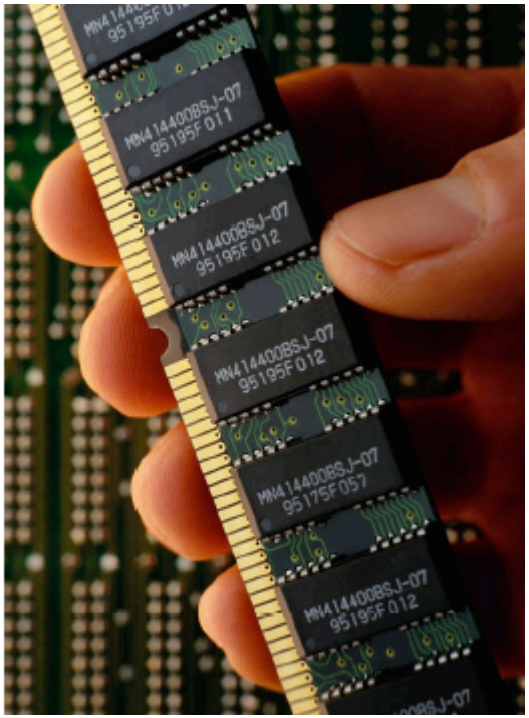
- As beginning programmers, the resource that you'll be concerned with most is memory
  - Most programming languages do a lot of the work for you
    - More on this soon

Memory

- Memory consists of a series of locations, each having a unique address, that are used to store programs and data.
- When data is stored in a memory location, the data that was previously stored there is overwritten and destroyed.
- Each memory location stores one byte (or 8 bits) of data.
  - Each bit is a 0 or a 1
    - More on this soon



Memory


MN4144008SJ-07
95195F011
MN4144008SJ-07
95195F012
MN4144008SJ-07
95195F012
MN4144008SJ-07
95175F057
MN4144008SJ-07
95195F012

Units of Memory Storage

We measure units of memory in terms of bytes:

Unit	Symbol	# of bytes
byte (8 bits)		$2^0 = 1$
kilobyte	KB	$2^{10} = 1024$
megabyte	MB	$2^{20} = 1024^2$
gigabyte	GB	$2^{30} = 1024^3$
terabyte	TB	$2^{40} = 1024^4$

What Can Be Represented By A Byte?

- 256 different characters from your keyboard
  - Java actually uses 2 bytes to represent a character
    - how many characters is that?
- 256 different shades of gray in a black and white image
- 256 colors or shades of color in a color image
- 256 frequencies or tones to be played through a speaker
- 256 of anything that can be represented as discrete entities
- part of an instruction for a computer

Memory

Macintosh SE in 1987
1 megabyte (MB) of memory

MacBook Pro in 2008
2 gigabytes (GB) of memory

1000 times more memory capacity in 20 years
1000 times greater processing speed
Approximately the same price

Mass storage/long-term memory

A disk drive without its protective case

Central processing unit

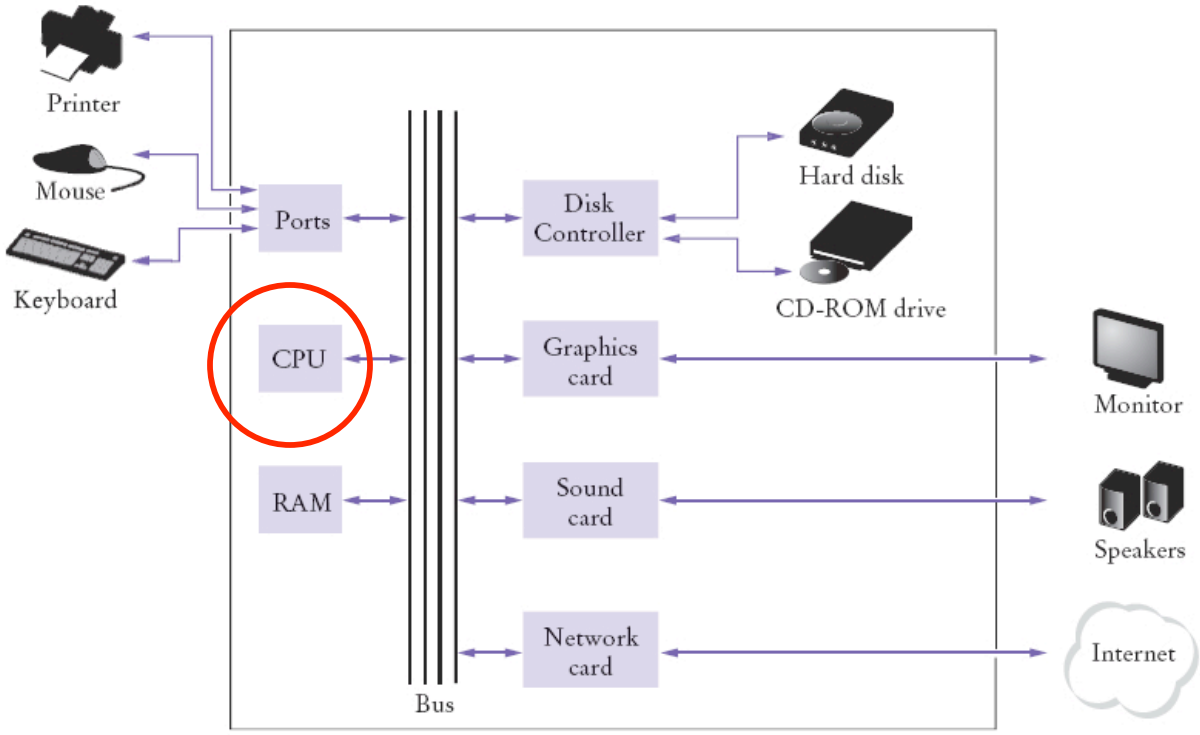
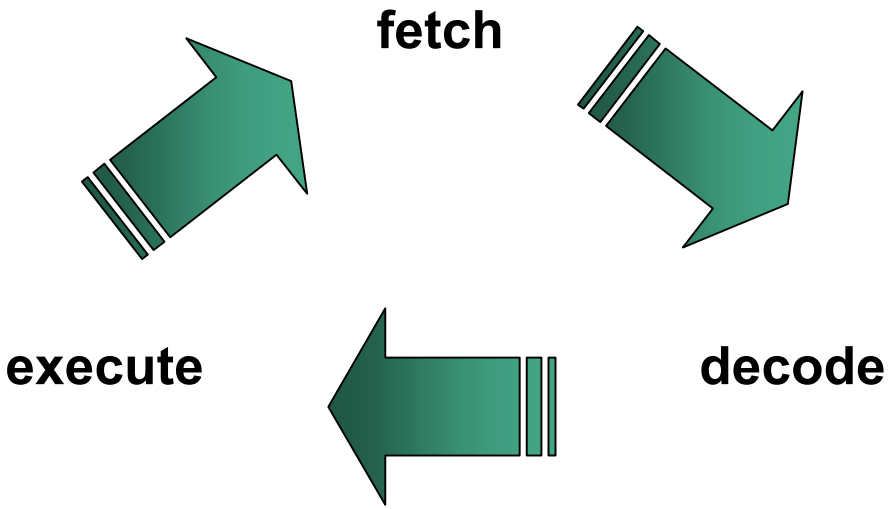


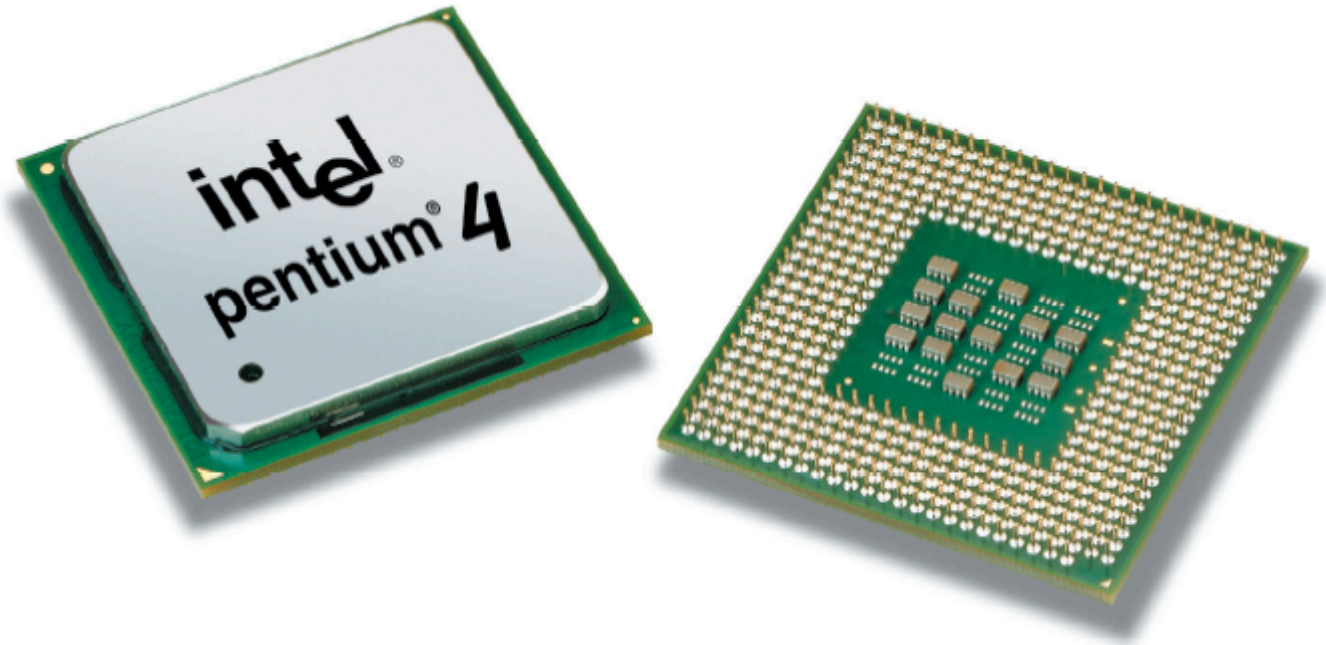
Figure 5 Schematic Diagram of a Computer

Central processing unit

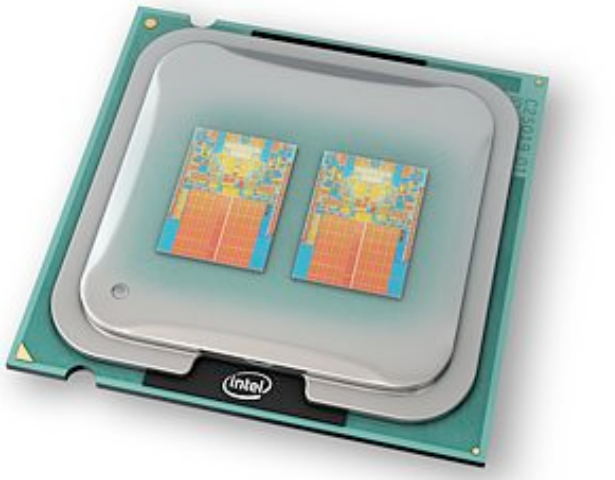
- CPU executes instructions in a continuous cycle
  - known as the “fetch-decode-execute” cycle
- CPU has dedicated memory locations known as **registers**
  - One register, the **program counter**, stores the address in memory of the next instruction to be executed



Central processing unit



Central processing unit



Instructions

- Implication of the fetch-decode-execute cycle
  - we control the computer to make it do what we want by giving it a sequence of little steps for it to do
  - these steps are the instructions in a programming language

Programming Languages

- Objectives
  - Understand what is meant by computer programming.
  - Understand the difference between machine/assembly language and a high-level computer language.
  - Understand what compilers and interpreters are, and why we use them.
  - Write, compile, and run a simple Java program.

What is Computer Programming?

What is a Computer?

How is a computer different from a video game console? Or a DVD player? Or a telephone? Or a bank machine?

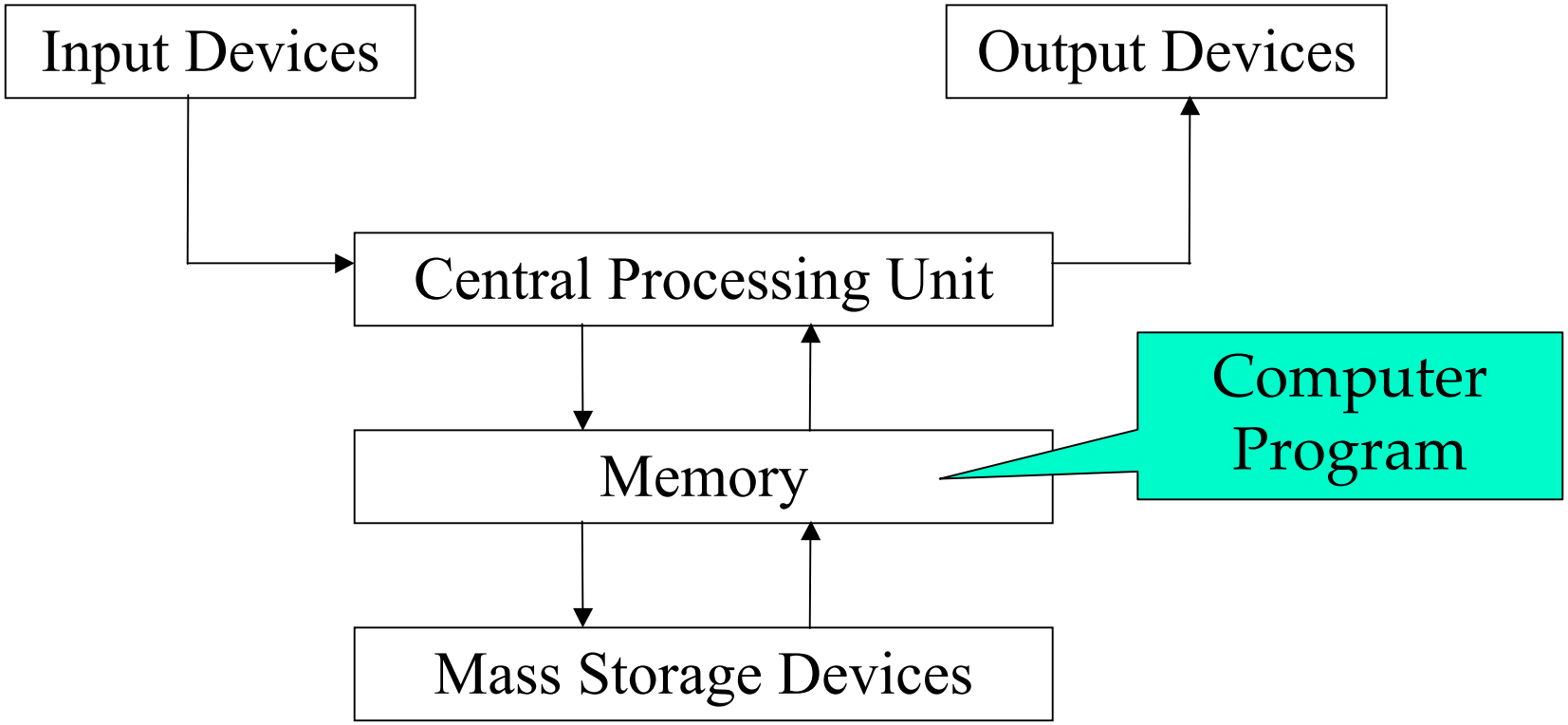
What is a Computer?

How is a computer different from a video game console? Or a DVD player? Or a telephone? Or a bank machine?

The computer is **general**. It can be all of the other devices.

Making the computer do what we want is called **programming** the computer.

Computer Programming



Computer Programming

- You can make the computer do anything that it's capable of. The only limits are space, time, I/O devices, and your skill and creativity
- It takes work.
  - The biggest program you'll write in 111 will be a few hundred lines long.
  - Windows XP is 40 million lines long.
- You have to write in a language the computer understands.

George and Stephen go to France

- George is American. He knows only English.
- Stephen is Canadian. He is bilingual in English and French.
- How can George communicate in France?

George and Stephen go to France

- How can George communicate in France?

1. If he wants to communicate quickly, then Stephen can **interpret** – translating French to English and English to French on-the-fly.
2. If there's a lot of stuff to translate (e.g., a speech, or a long document), then Stephen can translate the whole thing at once. Now, George can read it whenever he wants.

George and Stephen go to France

Translations can be combined:

In the Louvre, they see inscriptions in Egyptian hieroglyphics.

A museum sign gives a French translation.

Stephen interprets the sign for George.

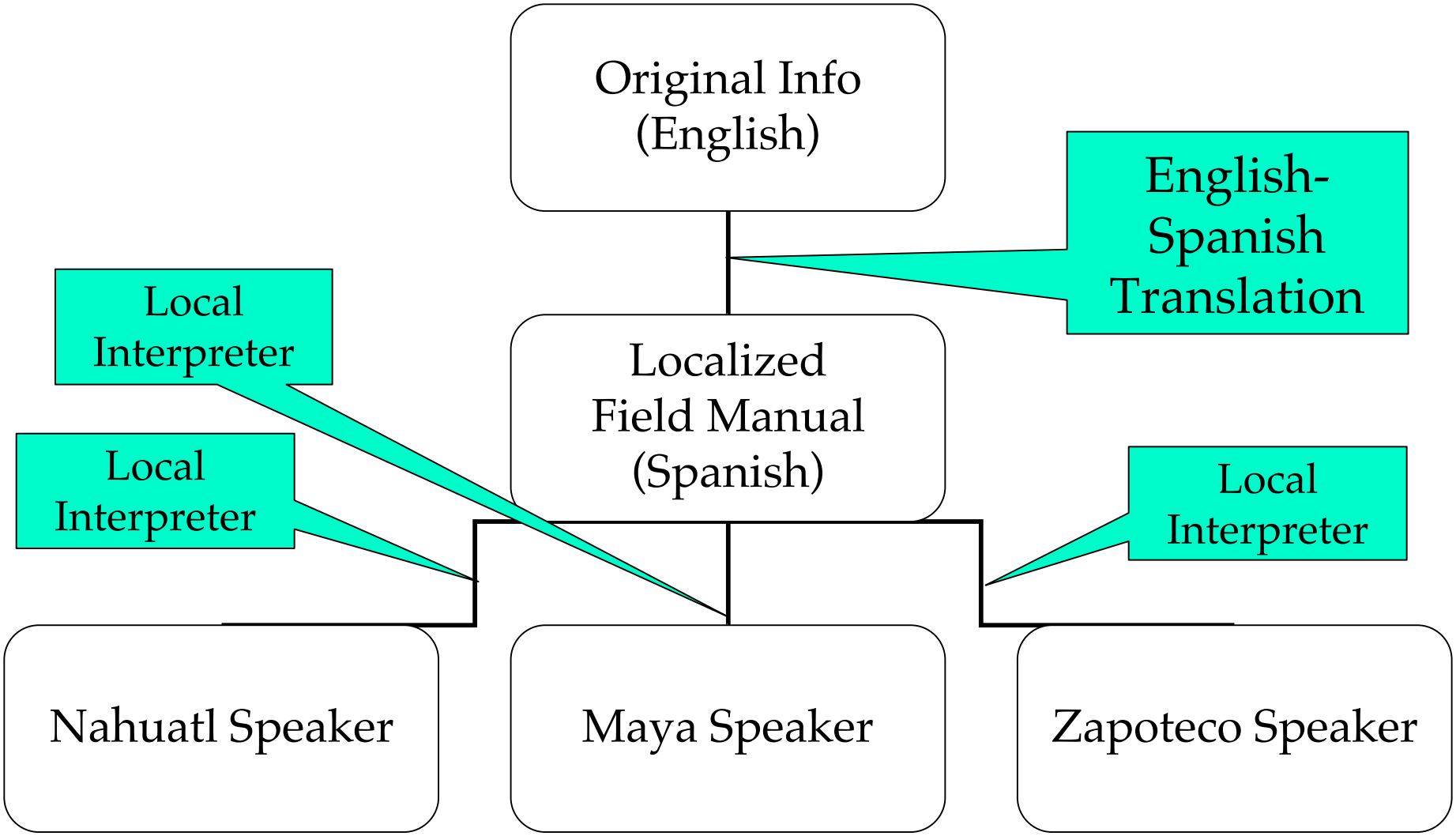
George can understand the hieroglyphics.

Health Education in Remote Areas

- In remote areas of the world, there are languages spoken by small groups of people, and also a national language spoken by the mainstream, e.g.,:
  - Many native languages vs. Spanish in Latin America
  - Minority languages vs. Mandarin in China
  - Regional languages vs. Hindi or English in India

- How do you provide health info (e.g., in English) to the isolated (e.g., in Latin America)?

Health Education in Remote Areas



Machine Language

- This is the “native language” of a computer.
- Each instruction does very little.
- The computer does them **very** fast.
- Each kind of processor has its own machine language, e.g.:
  - x86 (Intel, AMD), Windows and new Apples
  - PowerPC (Freescale, IBM), older Apples
  - SPARC (Sun), used in Sun servers
  - Many more…
- Remember: Everything is in binary!

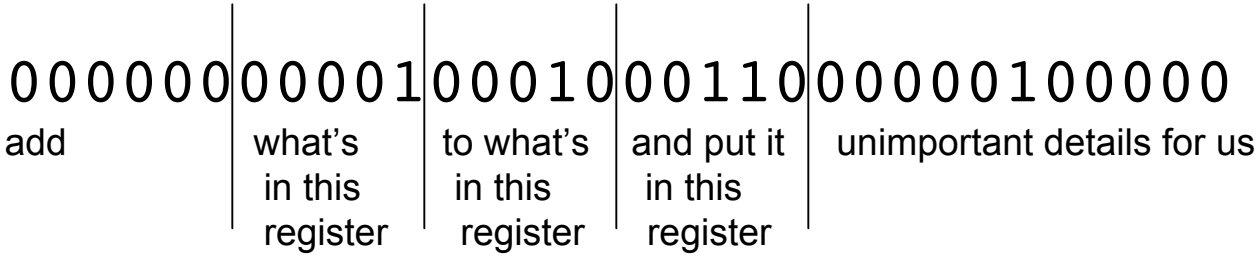
Machine Language

- First programming languages: machine languages
  - Most primitive kind
- Sample machine language instruction
  - what do you suppose it means?

Machine Language

- First programming languages: machine languages
  - most primitive kind

- Sample machine language instruction
  - Register: special purpose memory location inside CPU where real computation occurs



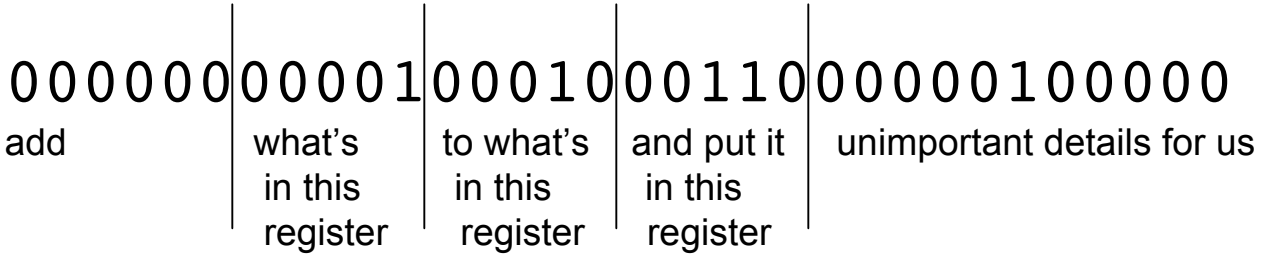
Machine Language

Digital Equipment Corporation PDP11/05 (circa 1974)

Machine Language

- First programming languages: machine languages
  - Most primitive kind

- Sample machine language instruction
  - Register: special purpose memory location inside CPU where real computation occurs

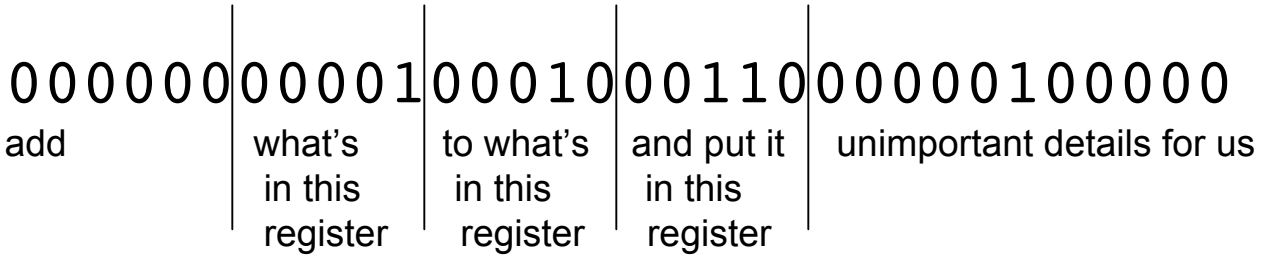


- Difficult to write programs this way
  - People created languages that were more readable

Assembly Language

- Next: assembly languages
  - Direct mappings of machine language instructions into helpful mnemonics, abbreviations
- Sample assembly language instruction
  - Corresponds to machine language instructions

`add r1,r2,r6`



Aside – Binary vs. Decimal Numbers

- We are used to representing numbers in the decimal system
  - have digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- For example:

4763

- Means (read above number right to left):

$$3*10^0+6*10^1+7*10^2+4*10^3$$

  - Note how the exponents count up from 0!

Aside – Binary vs. Decimal Numbers

- With binary numbers, we use 2 as the base
  - have digits 0,1
- Example:

$$10010111$$

- Means:

$$1*2^0+1*2^1+1*2^2+0*2^3+1*2^4+0*2^5+0*2^6+1*2^7$$

$$= 1+2+4+16+128 = 151$$

  - We have just converted a binary number to decimal
  - more details, decimal to binary conversion in CPSC 121

Aside – Other Bases

- The same principle works for other bases
- For example, *hexadecimal* (base 16)
  - uses digits 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
  - A-F correspond to values 10-15
- Example:

C350

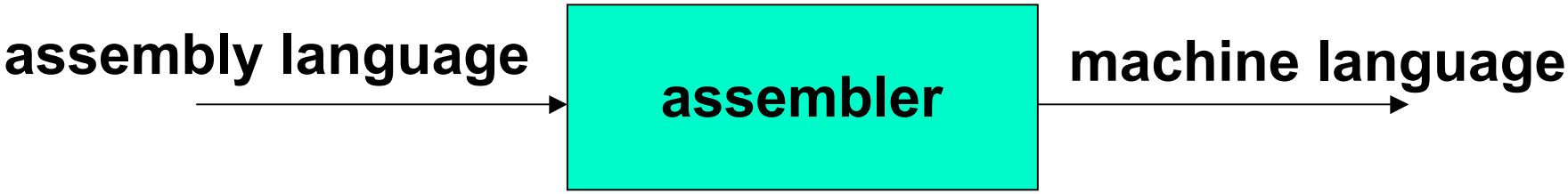
- Means:

$$0*16^0 + 5*16^1 + 3*16^2 + 12*16^3$$

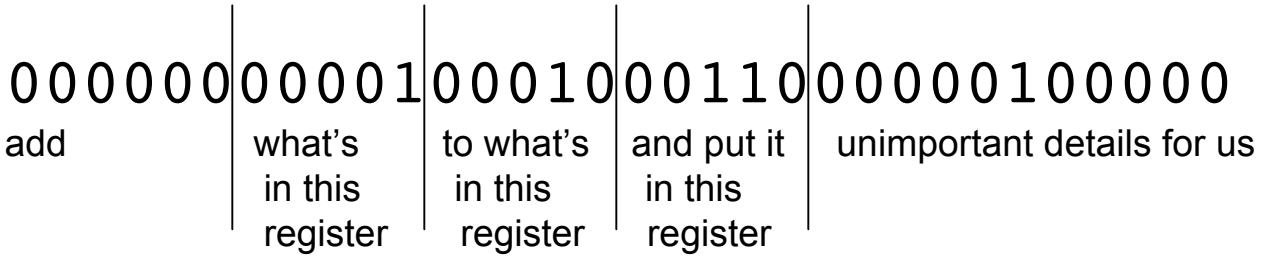
$$= 5*16 + 3*256 + 12*4096 = 50{,}000$$

Assembly Language

- Assembly language program converted into corresponding machine language instructions by another program called an assembler




```
add r1,r2,r6
```



Assembly Language

- Both machine and assembly languages pose big challenges for programmers
  - Difficult to read and write
  - Difficult to remember
- Each instruction does very little
  - Takes lots of instructions just to get something simple done
- Every machine or assembly language good for only one type of computer
  - Different to program IBM than Honeywell than Burroughs...

High-Level Language

- Next step: development of high-level languages
- You may have heard of some
  - Fortran, COBOL, Lisp, BASIC, C, C++, C#, Ada, Perl, Java, Python, Ruby, Javascript
- High-level languages intended to be easier to use
  - still a long way from English.
- A single high-level instruction gets more work done than a machine or assembly language instruction.
- Most high-level languages can be used on different computers

Java

- Java is the high-level language we'll use.
  - Modern, widely used, portable, safe.

- Developed by Sun in early 1990s
  - Originally intended for set-top boxes
  - Retargeted for the Web

High-Level Language

- Example of a high-level instruction
  - A = B + C

- Tells computer to
  - go to main memory and find value stored in location called B
  - go to main memory and find value stored in location called C
  - add those two values together
  - store result in memory in location called A

High-Level Language

- Must be translated into machine language so the computer can understand it.

- High-level instruction: A = B + C

becomes at least four machine language instructions!

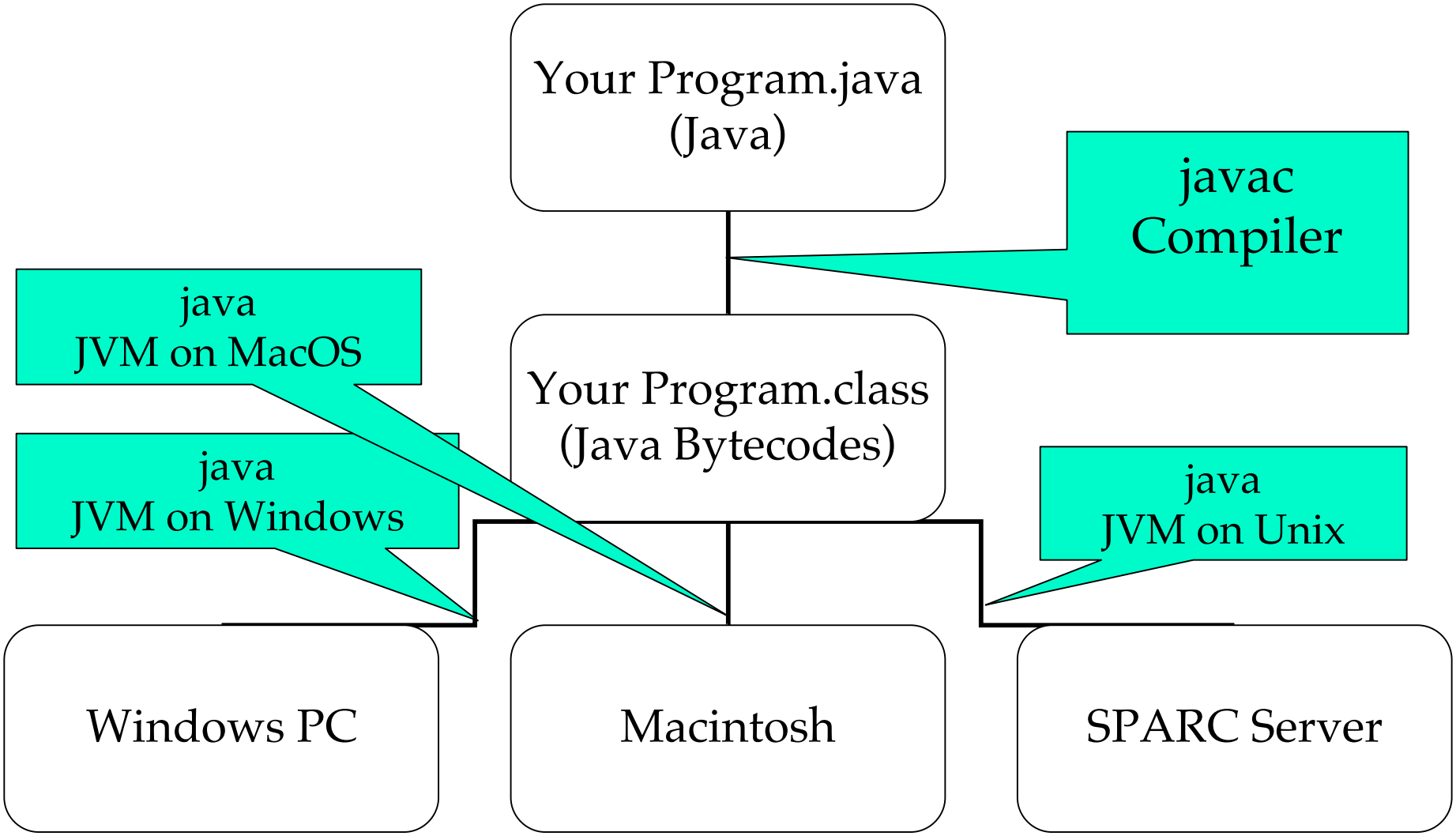
```
00010000001000000000000000000010   load B
00010000001000000000000000000011   load C
00000000001000100011000000100000   add them
00010100110000000000000000000001   store in A
```

- How?
  - You could translate it as you go (**interpreter**).
  - You could translate it in advance (**compiler**).

Interpreters and Compilers

- An interpreter translates the high-level language into machine language on-the-fly, executing the instructions as it goes.
- A compiler translates the high-level language program all at once in advance.
- Both compilers and interpreters are themselves computer programs.
- Which is better?
  - Remember George and Stephen in France?

Java Does Both!



A Simple Java Program

```
// Our first Java program.
/* Traditionally, one's first program in a new
   language prints out "Hello, World!"
*/
class HelloTester {
   public static void main(String[] args) {
    System.out.println("Hello, World!");
   }
}
```

Sample Java Application Program

```
//*********************************************************
// Oreo.java          Author:  Kurt Eiselt
//
// Demonstrating simple Java programming concepts while
// revealing one of Kurt's many weaknesses
//*********************************************************

public class Oreo
{
  //*******************************************************
  // demand Oreos
  //*******************************************************
  public static void main (String[] args)
  {
    System.out.println ("Feed me more Oreos!");
  }
}
```