Lecture 1

Introduction

In the beginning...

CS 241: Foundations of Sequential Programs
Fall 2014

Troy Vasiga et al
University of Waterloo

What's in a name?

Foundations of Sequential Programs

- What really happens when I compile and run a program?
- By the end of the course, there should be very little mystery left about computers or computer programs.

Where do we start?

Assume computers exist.

About the course

due Wed 5pm

- Assignments
  - START EARLY!
  - Don't fall behind!!!
  - 11 assignments in total, each with 8 subparts: about 100 things to submit
- Design choice
- Outline
- Notes

① Data Structures/Algorithms
② Testing/Examples

① Low-Level
② High-Level
③ Memory

Marmoset

Due Wednesday 5PM

No Lates

- Public tests
- Release tests
  - Release tokens — 3 at the start
    regenerate every 24 hrs.

Linux

Your program must run correctly on the `linux.student.cs` environment.

use gvim, vi, emacs ~--

Purpose of the course

- To learn (to learn).
- Meta-thinking.
- Write a program that reads a program and outputs a program.

Abstraction

Marking

- Assignments: 25%
- Midterm: 25% written on Thursday, October 23, 4:30-6:30pm
- Final Exam: 50% written sometime in December

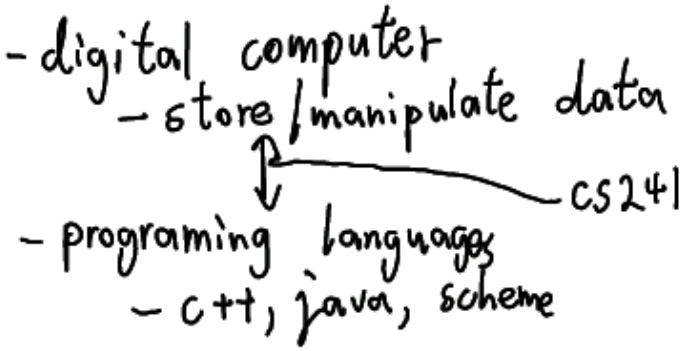
Personnel

- Instructors:
  - Troy Vasiga (troy.vasiga@uwaterloo.ca)✓
  - Gregor Richards (gregor.richards@uwaterloo.ca)
  - Adriel Dean-Hall (adeanhal@uwaterloo.ca)
- ISA: Maxim Bardakov (cs241@uwaterloo.ca)
- Instructional Support Coordinator: Gang Lu (glu@uwaterloo.ca)
- IAs/TAs: run tutorials

Non-human Resources

- Textbooks:
- Discussion Forum: Piazza

Computers to Programming Languages



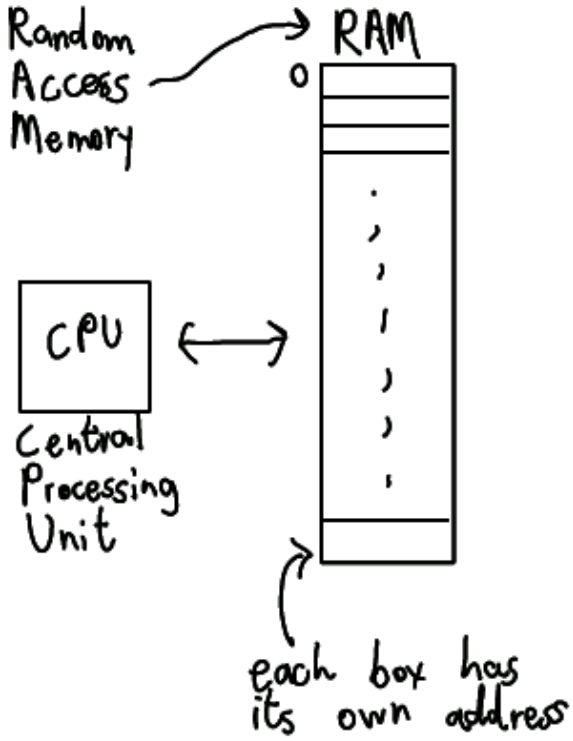
data → bits → binary digits

What is a computer?

CPU+RAM

store data

manipulate data



What is a program?

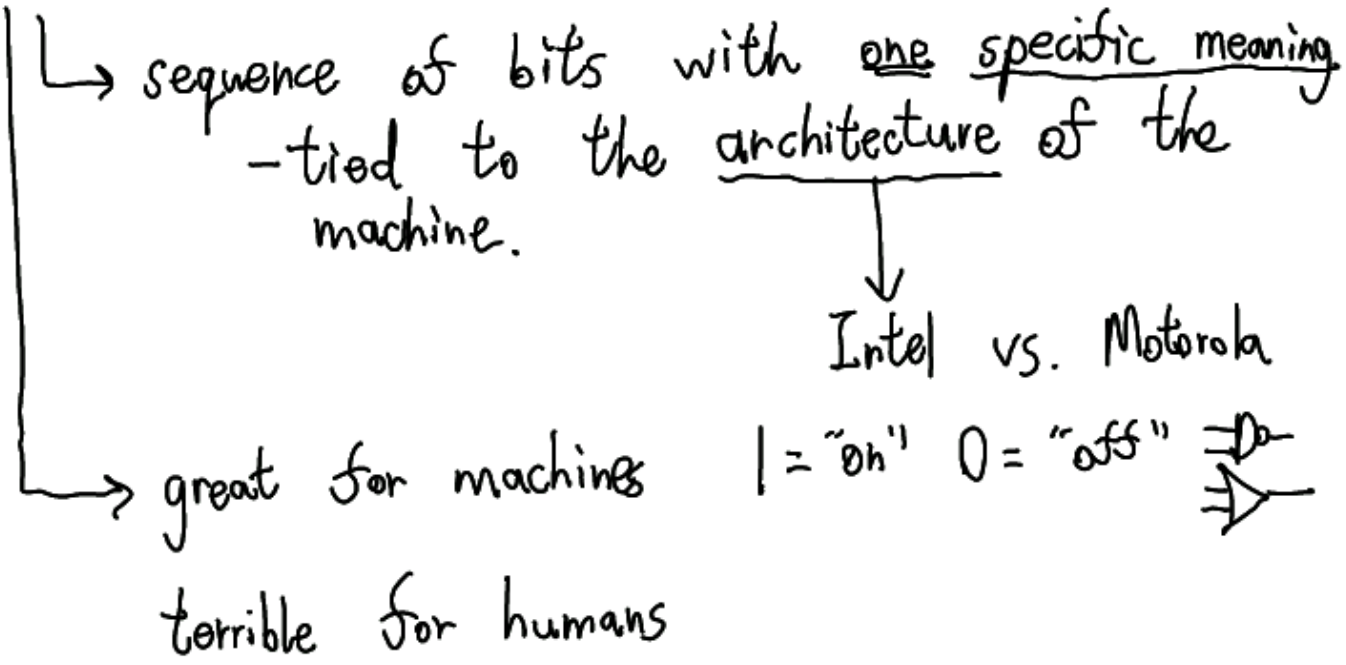
"Text" file $\Longrightarrow$ "Binary" file

Binary data

- sequence of bits
  - many interpretations for that sequence
    - many meanings

Interpretation is in the eye of the beholder.

Machine language



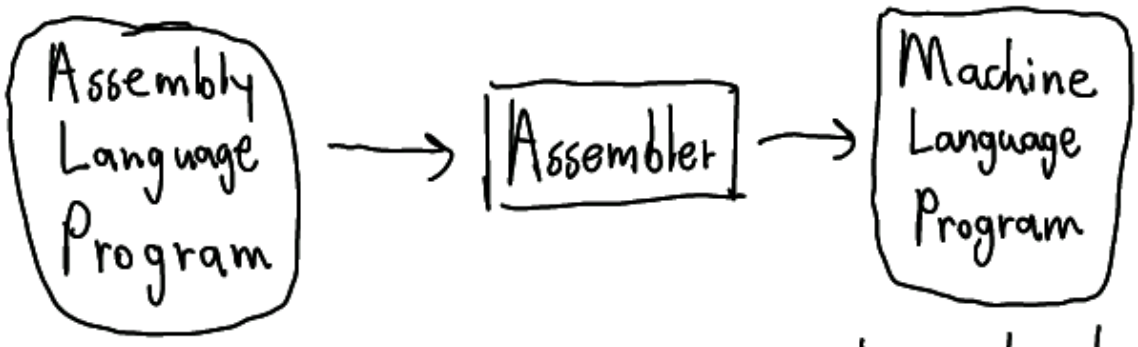
Assembly language

- simple, textual representation of machine language
  - more human readable.
- relatively easy to translate assembly to machine language.

Abstraction => hides details to see bigger picture

Assembler

↳ a program that translate assembly language into machine language.



Low-level chunk of the course.

Formal languages

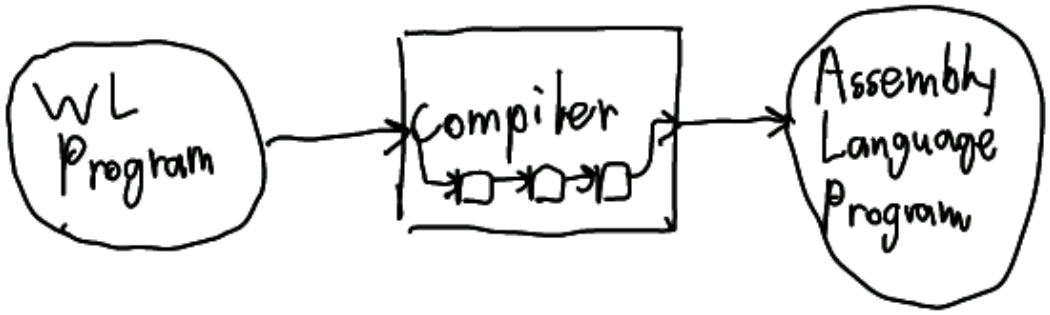
Regular Languages
Context-Free Languages

Higher level languages

C, Java, Scheme, Python, ...

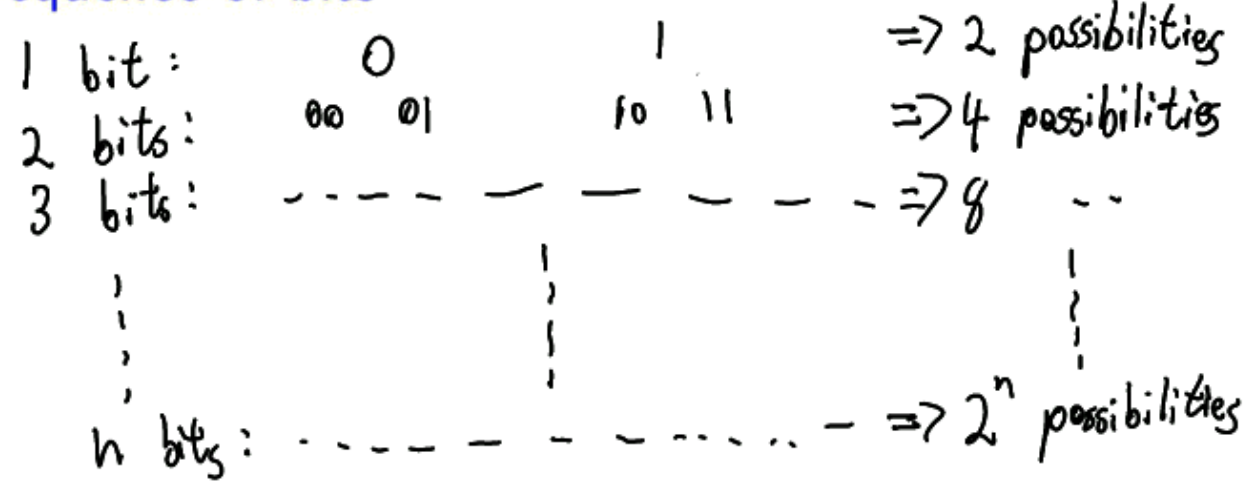
→ WL ("wool")
Waterloo Language

Write a compiler for WL / WLPP / WLP4



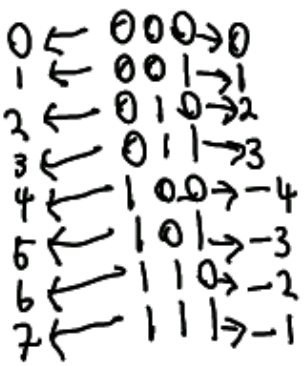
Back to Basics: Bits

Sequence of bits



Integers

3 bits: 8 possibilities



① ②

① Could be:
unsigned Integer
0 ... 7
$(0 \ldots 2^n - 1)$
(base-2 number)

② Could be:
2's complement
representation.
$(-4 \ldots 3)$
$(-2^{n-1} \ldots 2^{n-1} - 1)$

2s complement operation and 2s complement representation

−4 in 3 bits, 2's complement representation

① Write down the unsigned num in binary.

100

② negate the bits ⇒ 011

③ add 1 ⇒ 011 ⇒ 100

$$\begin{array}{r} 011 \\ +\ \ 1 \\ \hline 100 \end{array}$$

What is $\overset{8421}{1010}$?

① unsigned binary #: $(10)_{10}$

② 2's comp rep binary:

1010

↓ negate

0101

↓ +1

0110

↳ $(6)_{10}$ ⟹ $1010 \rightarrow (-6)_{10}$

Perform the 2's comp operation.

Characters

8 bits ↓

e.g. 00001010

↳ $(10)_{10}$ in unsigned binary

Hexadecimal

base 16

digits: 0, 1, ..., 9, A, B, C, D, E, F

10 11 15

1010 1111 1001 1001

A F 9 9 $\Rightarrow$ 0xAF99