How Programming Works

cSplash 2011
Paul Gazzillo
Introduction
Introduction

• What programming languages do you use?
Introduction

• What programming languages do you use?
• Challenges of programming
Introduction

• What programming languages do you use?
• Challenges of programming
• Why programming languages
Introduction

- What programming languages do you use?
- Challenges of programming
- Why programming languages
- Build a expression evaluator together
Architecture

Processor

Memory
Architecture

Processor
Crunching numbers

Memory
Architecture

Processor
Crunching numbers

Memory
Storing numbers and programs
Instruction Set
Instruction Set

Add, subtract, etc two numbers
Instruction Set

Add, subtract, etc two numbers

Compare two numbers
Instruction Set

Add, subtract, etc two numbers

Compare two numbers

Jump to another part of the program
Instruction Set

Add, subtract, etc two numbers

Compare two numbers

Jump to another part of the program

Read and write a number to memory
Humans use *expressions*
Humans use expressions

\[ x = v_0 t + \frac{at^2}{2} \]
Humans use expressions

\[ x = v_0 t + \frac{at^2}{2} \]

Processors only do single steps
Humans use *expressions*

\[ x = v_0 t + \frac{a t^2}{2} \]

Processors only do *single steps*

\[
\begin{align*}
\text{temp1} &= v_0 \times t \\
\text{temp2} &= a \times t \\
\text{temp3} &= \text{temp2} \times t \\
\text{temp4} &= \text{temp3} / 2 \\
\end{align*}
\]

\[ x = \text{temp1} + \text{temp4} \]
Humans use expressions

\[ x = v_0 t + \frac{at^2}{2} \]

Processors only do single steps

\[
\begin{align*}
\text{temp1} &= v_0 \times t \\
\text{temp2} &= a \times t \\
\text{temp3} &= \text{temp2} \times t \\
\text{temp4} &= \text{temp3} / 2
\end{align*}
\]

\[ x = \text{temp1} + \text{temp4} \]

Translate human expressions to processor instructions
Humans use expressions

\[ x = v_0 t + \frac{at^2}{2} \]

Processors only do **single steps**

\[
\begin{align*}
\text{temp1} &= v_0 \times t \\
\text{temp2} &= a \times t \\
\text{temp3} &= \text{temp2} \times t \\
\text{temp4} &= \text{temp3} / 2 \\
\therefore x &= \text{temp1} + \text{temp4}
\end{align*}
\]

**Translate** human expressions to processor instructions
Programming Languages
Programming Languages

- Bridge human expression and the machine
Programming Languages

- Bridge human expression and the machine
- They *abstract away* repetitive and tedious tasks
Programming Languages

- Bridge human expression and the machine
- They *abstract away* repetitive and tedious tasks
- Eases implementing complex concepts
Programming Languages

• Bridge human expression and the machine

• They *abstract away* repetitive and tedious tasks

• Eases implementing complex concepts

• Less error-prone
Programming Languages

• Bridge human expression and the machine

• They *abstract away* repetitive and tedious tasks

• Eases implementing complex concepts

• Less error-prone

• Faster, more convenient programming
Programming Languages

• Bridge human expression and the machine
• They *abstract away* repetitive and tedious tasks
  • Eases implementing complex concepts
  • Less error-prone
• Faster, more convenient programming
Compiler vs Interpreter
Compiler vs Interpreter

Source program

Compiler

Input  Target program  Output
Compiler vs Interpreter

Source program

Compiler

Target program

Input → Output

Source program

Interpreter

Input → Output
Today’s Project
Today’s Project

• An interpreter that take an arithmetic expression, e.g.
Today’s Project

- An interpreter that take an arithmetic expression, e.g.
- \((8+2)\times3\)
Today’s Project

• An interpreter that take an arithmetic expression, e.g.

• (8+2)*3

• And computes the result, e.g.
Today’s Project

• An interpreter that take an arithmetic expression, e.g.
  • (8+2)*3
  • And computes the result, e.g.
    • 30
Today’s Project

• An *interpreter* that takes an arithmetic expression, e.g.
  • \((8+2)*3\)

• And computes the result, e.g.
  • 30

• We will be using a *stack*
Stack
Implementation Time!
Questions?