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

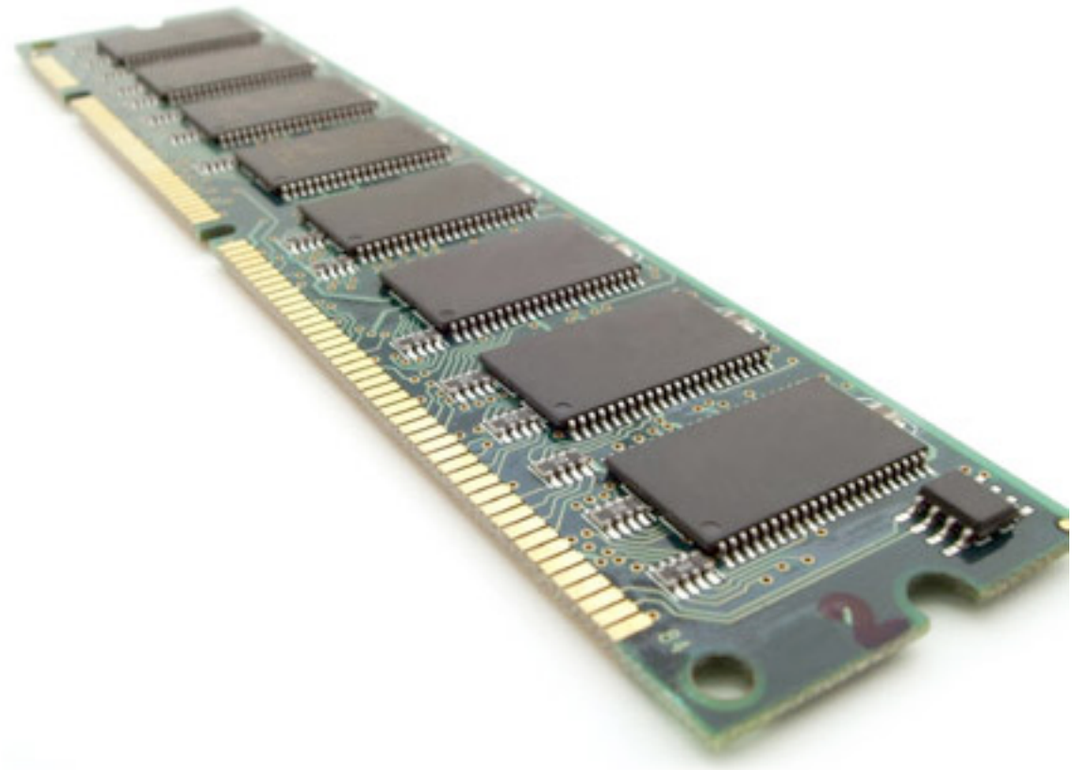
- What programming languages do you use?
- Challenges of programming
- Why programming languages

Introduction

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

Architecture

Processor

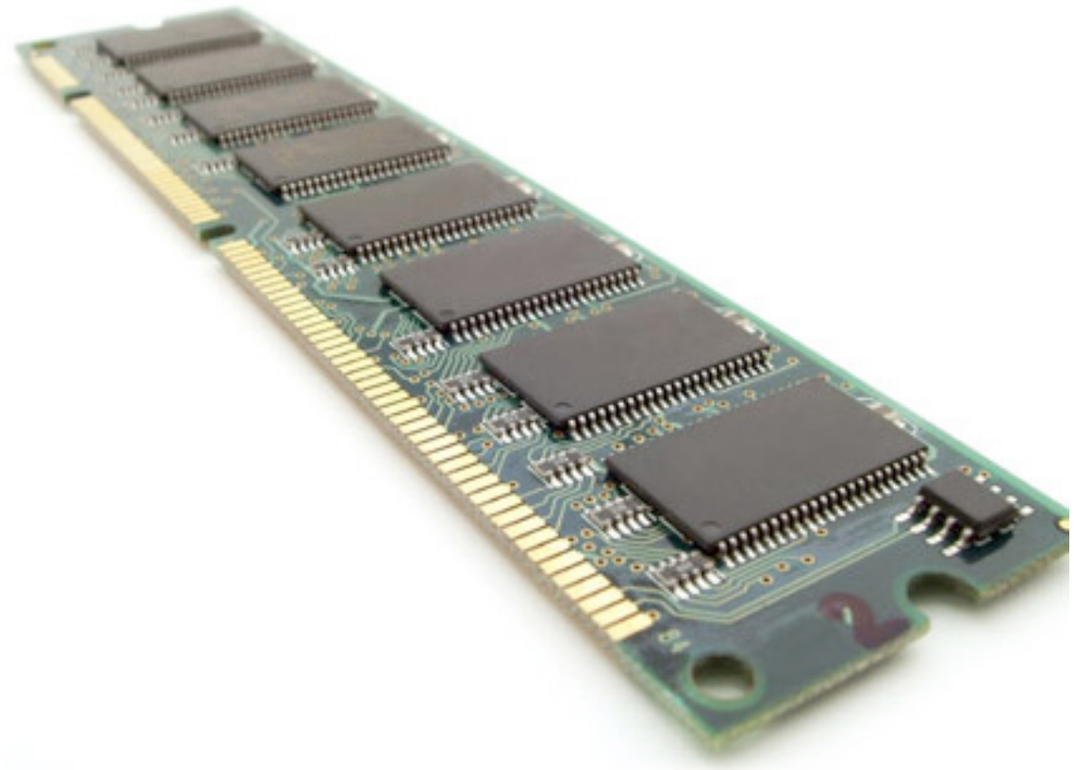


Memory

Architecture

Processor

Crunching numbers

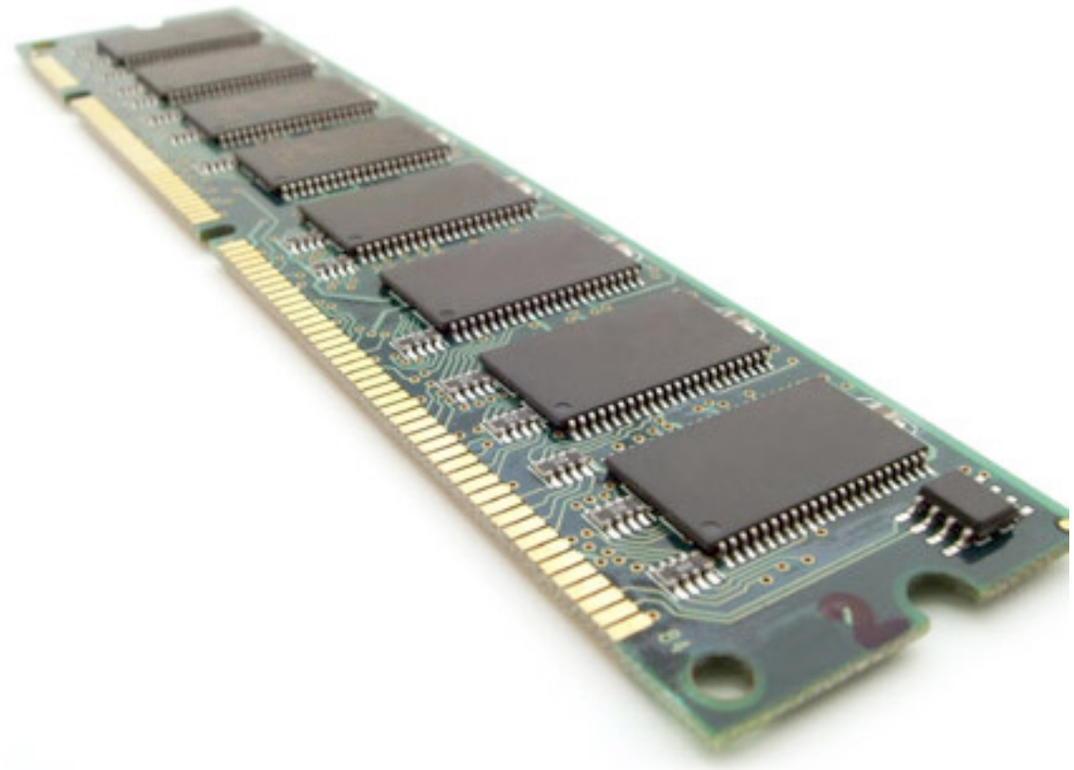


Memory

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*

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

Processors only do *single steps*

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*

$$temp1 = v_0 * t$$

$$temp2 = a * t$$

$$temp3 = temp2 * t$$

$$temp4 = temp3/2$$

$$x = temp1 + temp4$$

Humans use *expressions*

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

Processors only do *single steps*

$$temp1 = v_0 * t$$

$$temp2 = a * t$$

$$temp3 = temp2 * t$$

$$temp4 = temp3/2$$

$$x = temp1 + temp4$$

Translate human expressions to processor instructions

Humans use *expressions*

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

Compiler

Processors only do *single steps*

$$temp1 = v_0 * t$$
$$temp2 = a * t$$
$$temp3 = temp2 * t$$
$$temp4 = temp3/2$$
$$x = temp1 + temp4$$

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

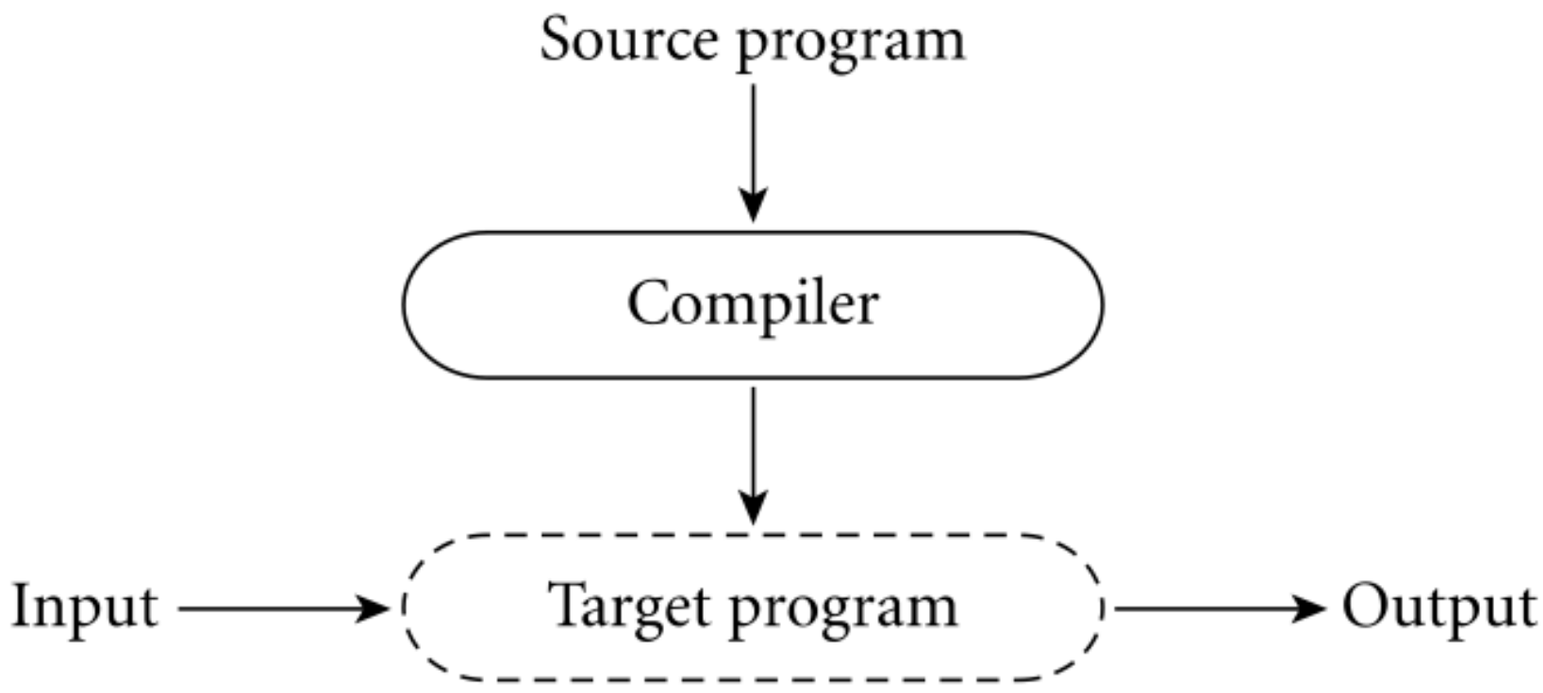
- Bridge human expression and the machine
- They *abstract away* repetitive and tedious tasks
- Eases implementing complex concepts
- Less error-prone
- Faster, more convenient programming

Programming Languages

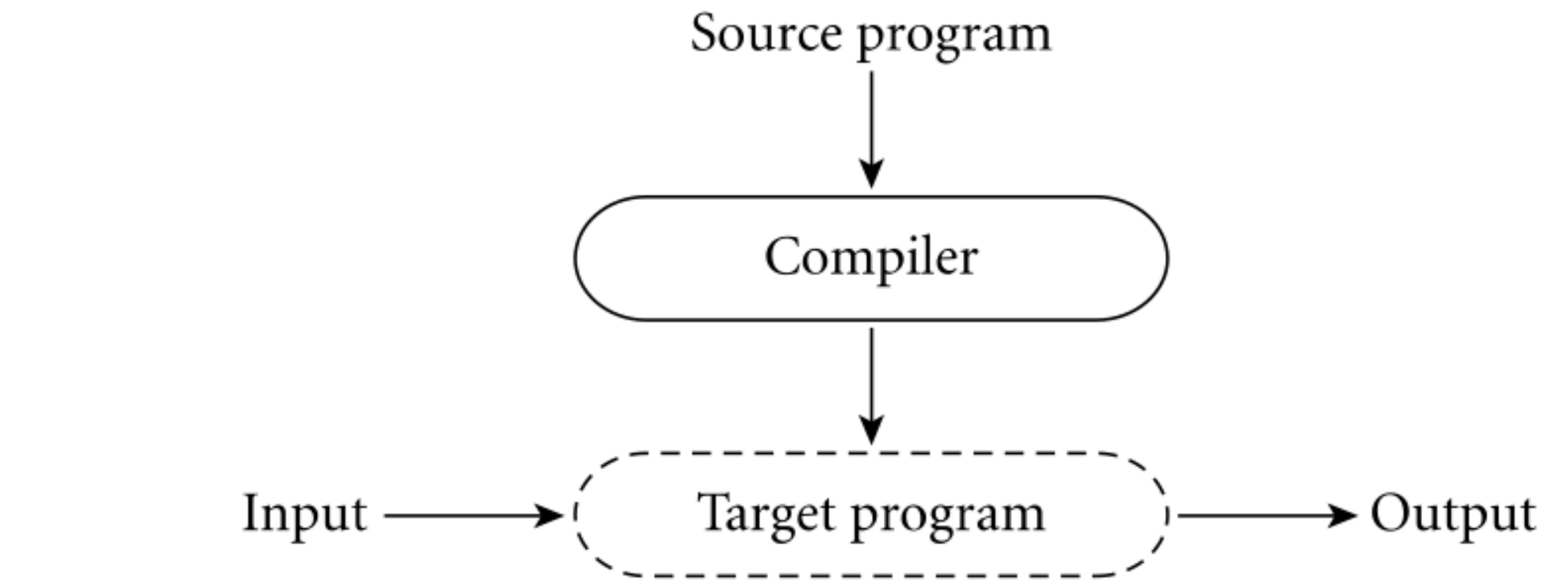
- Bridge human expression and the machine
- They *abstract away* repetitive and tedious tasks
- Eases implementing complex concepts
- Less error-prone
- Faster, more convenient programming

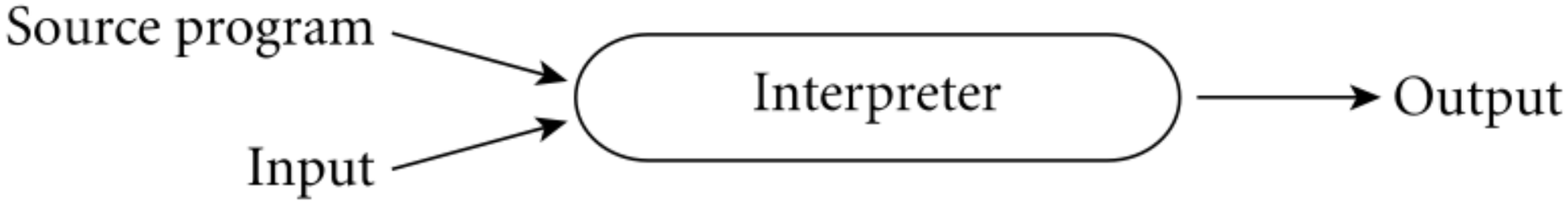
Compiler vs Interpreter

Compiler vs Interpreter



Compiler vs Interpreter





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)*3

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 take 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?