Lisp/Racket and Implementation (2)

Interpretation, Translation, and everything in between
Programs as Data
If time: Implementing Racket in Racket
- hands-on
- how Lisp was first implemented

How to implement a programming language

Interpretation

An **interpreter** written in the **implementation language** reads a program written in the **source language** and **evaluates** it.

Translation (a.k.a. compilation)

An **translator** (a.k.a. **compiler)** written in the **implementation language** reads a program written in the **source language** and **translates** it to an equivalent program in the **target language**.

But now we need implementations of:

implementation language

target language

How to implement a programming language

Interpreter Rule

$$\frac{\text{P-in-L program} \quad \text{L interpreter machine}}{\text{P machine}}$$

Translator Rule

$$\frac{\text{P-in-S program} \quad \text{S-to-T translator machine}}{\text{P-in-T program}}$$

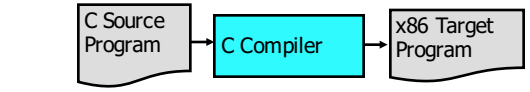
Prove how to implement a "251 web page machine" using:

- 251-web-page-in-HTML program (a web page written in HTML)
- HTML-interpreter-in-C program (a web browser written in C)
- C-to-x86-translator-in-x86 program (a C compiler written in x86)
- x86 interpreter machine (an x86 computer)

Metacircularity

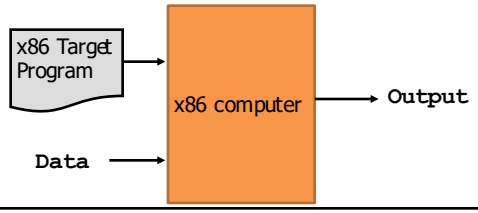
- Lisp in Lisp / Racket in Racket: eval
- Python in Python: PyPy
- Java in Java: Jikes RVM, Maxine VM
- …
- C-to-x86 compiler in C
  - Let's try to draw out that proof of existence…

Compiler



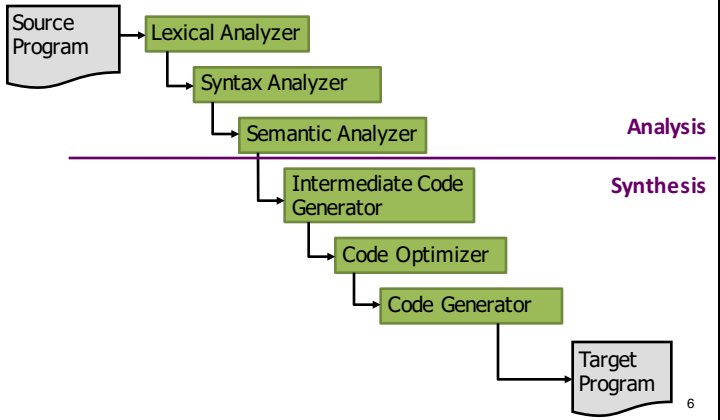

```
if (x == 0) {
  x = x + 1;
}
...
```

```
cmp (1000), $0
bne L
add (1000), $1
L:
...
```



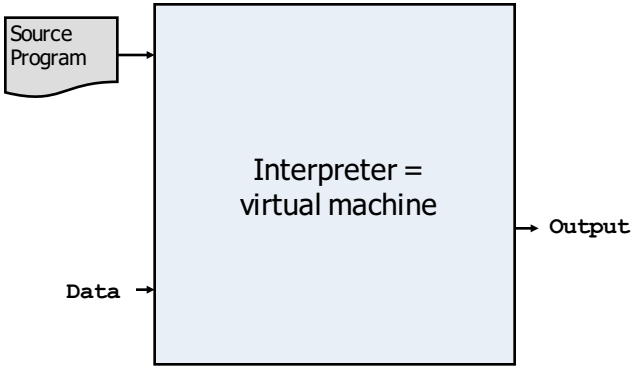
5

Typical Compiler



6

Interpreters



7

Interpreters vs Compilers

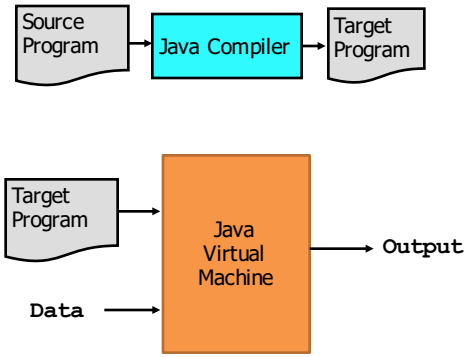
Interpreters

- No work ahead of time
- Incremental
- maybe inefficient

Compilers

- All work ahead of time
- See whole program (or more of program)
- Time and resources for analysis and optimization

Compilers… whose output is interpreted



Doesn't this look familiar?

9

Java Compiler

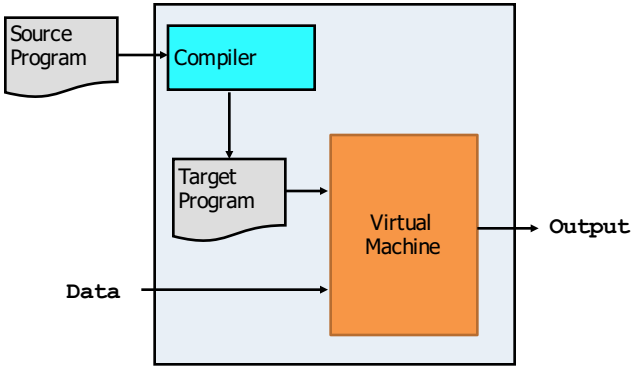

```
if (x == 0) {
  x = x + 1;
}
...
```

```
load 0
ifne L
load 0
inc
store 0
L:
...
```

(compare compiled C to compiled Java)

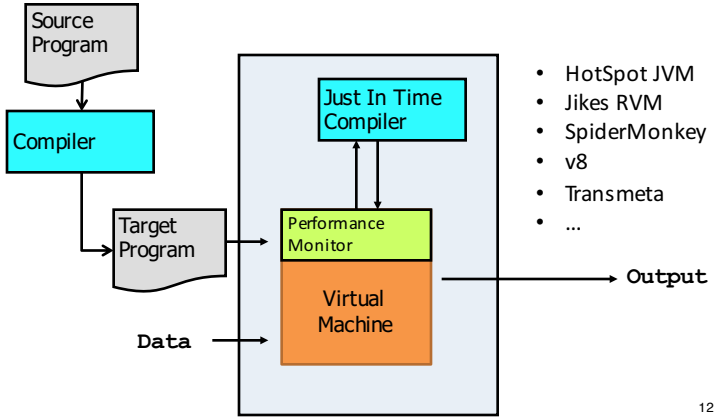
10

Interpreters… that use compilers.



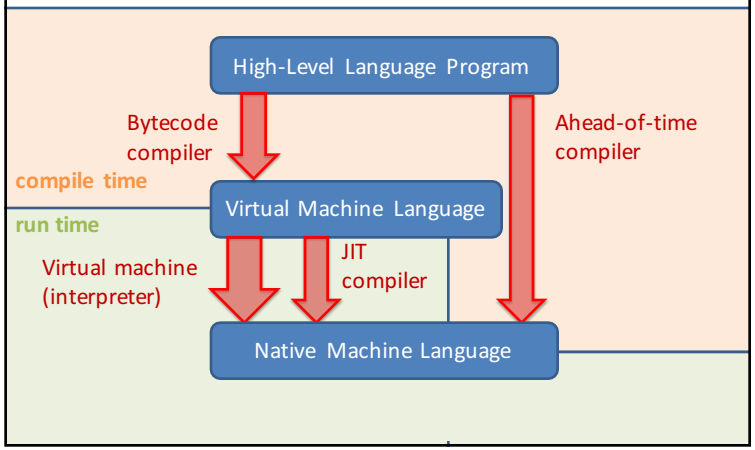
11

JIT Compilers and Optimization



12

Virtual Machine Model



Remember: language != implementation

Easy to confuse "the way this language is usually implemented" or "the implementation I use" with "the language itself."

Java and Racket can be compiled to x86

C can be interpreted in Racket

x86 can be compiled to JavaScript

Programs as Data

Racket programs look a lot like…

Symbols: `'a`

- Number and boolean symbols identical to values: `'#f` is `#f`

Atoms: symbols, numbers, booleans, null

General quoting:

- `(list 1 2 3)` produces a value DrRacket draws as `'(1 2 3)`
- `'(cons 1 2)` is the same as `(list 'cons '1 '2)`
- `'(lambda (x) (+ x x))`
  is the same as
  `(list 'lambda (list 'x) (list '+ 'x 'x))`