

CS 152: *Programming Language Paradigms*

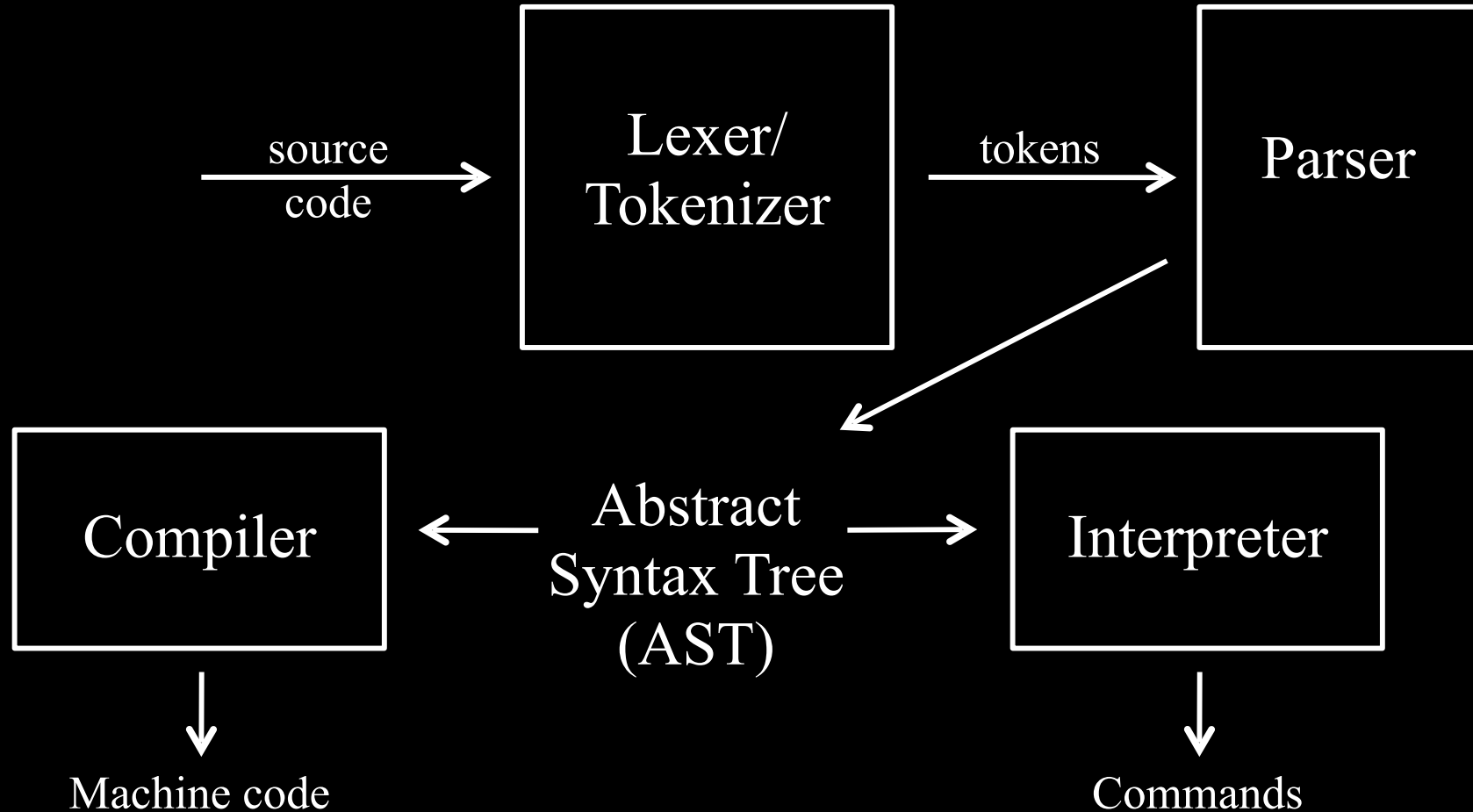


Virtual Machine
Lab

Prof. Tom Austin

San José State University

A Review of Compilers



Virtual Machines (VM)

- Code is compiled to *bytecode*
 - low-level
 - platform independent
- The VM interprets bytecode

Lab: Scheme VM

In today's lab, you will implement:

- a compiler for Scheme
- a stack-based VM

Input program

```
(println (+ 2 3 4))
```

```
(println (- 13 (* 2 4)))
```

```
(println (- 10 4 3))
```

Supported VM Operations

- **PUSH** – adds argument to stack
- **PRINT** – pops & prints top of stack
- **ADD**
 - pops top two elements
 - adds them together
 - places result on stack
- **SUB** – subtraction
- **MUL** – multiplication

Bytecode Output

PUSH 2	MUL
PUSH 3	SUB
ADD	PRINT
PUSH 4	PUSH 10
ADD	PUSH 4
PRINT	SUB
PUSH 13	PUSH 3
PUSH 2	SUB
PUSH 4	PRINT

Lab, part 1: Write a VM

- Starter code is provided.
- PUSH and PRINT are functional.
- Your job: add support for the other opcodes

Compiler or Interpreter?

- Compilers
 - efficient code
- Interpreters
 - runtime flexibility
- Can we get the best of both?

Just-in-time compilers (JITs)

- interpret code
- "hot" sections are compiled
at run time

JIT tradeoffs

- + Speed of compiled code
- + Flexibility of interpreter
- Overhead of both approaches
- Complex implementation

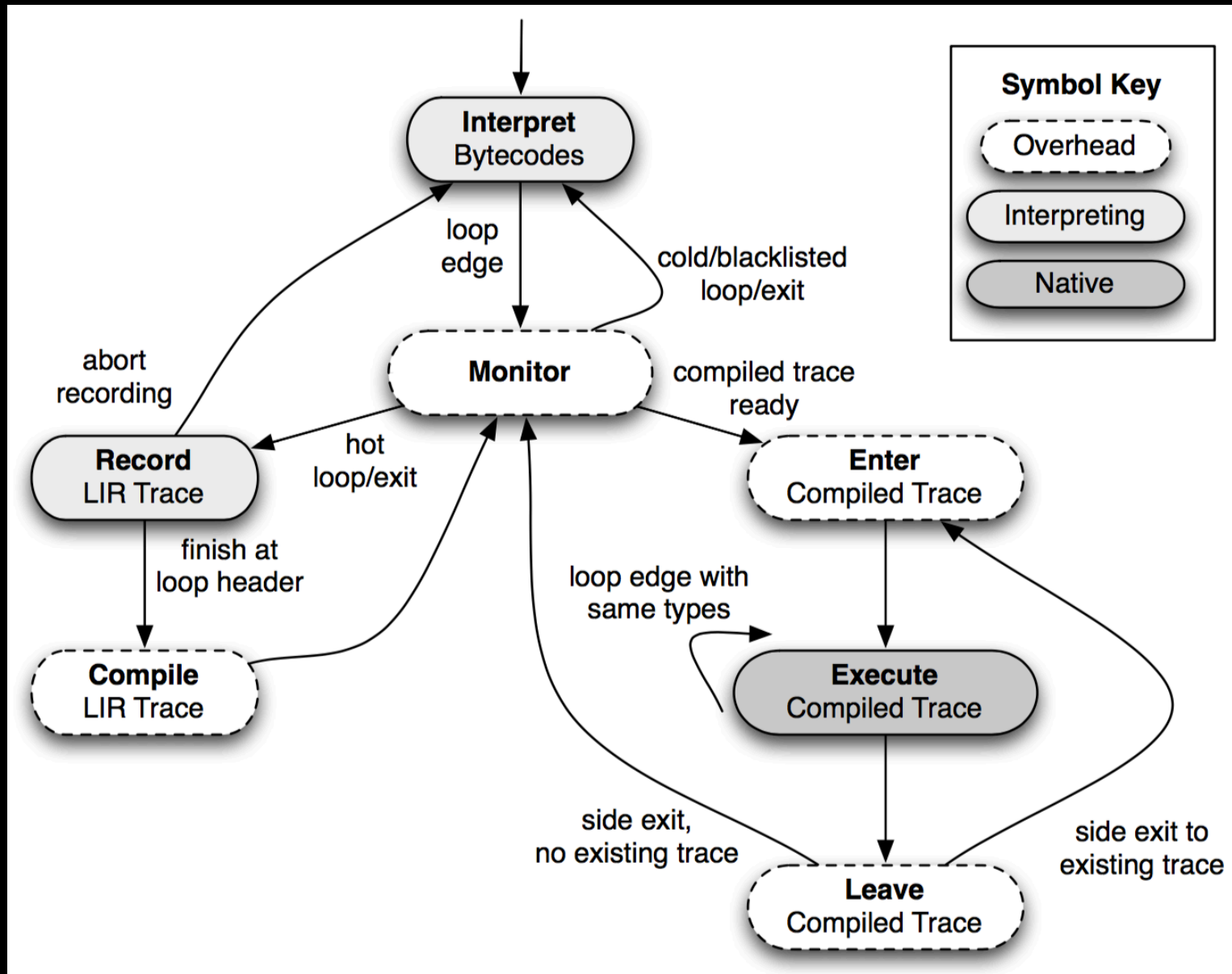
Dynamic recompilation

- JIT pursues aggressive optimizations
 - make assumptions about code
 - guard conditions verify assumptions
- Unexpected cases interpreted
- Can outperform static compilation

Types of JITs

- Method based
 - Compiles methods
- Trace based
 - Compiles loops
 - Gal et al. 2009 [http://
www.stanford.edu/class/cs343/
resources/tracemonkey.pdf](http://www.stanford.edu/class/cs343/resources/tracemonkey.pdf)

Trace-based JIT design (Gal et al. 2009)



How can a language designer make use of a JIT?

1. Become an expert in JITs
 - study the latest techniques
 - build large code bases to test
 - profile your code execution
2. Use someone else's JIT-ed VM

Lab, part 2 – Write a Compiler

- Starter code is provided.
- `println` is functional.
- Your job: update `to_bytecode` to add support for the mathematical operators.

Review of compiler.rb

(in class)

EXTRA CREDIT

Add compiler support for

- if expressions
- boolean variables
- let expressions

Add VM support for

- labels
- Jump (JMP / JZ / JNZ) operations
- STOR / LOAD operations