

Lecture 18+19

Code Generation

CS 241: Foundations of Sequential Programs
Fall 2009

Troy Vasiga et al
University of Waterloo

Big Picture of WL Compilation

From WL to MIPS

Zooming In On Compiler

Example

Input file:

```
int wain(int a, int b) {  
    println a;  
}
```

Example

Tokens from Scanner

Example

Parse Tree

Parse Tree

See `cfgr1.ss` or `CFGr1.java`

- ▶ structure
- ▶ traversals

Next steps

- ▶ processing a .wli file
- ▶ further error checking (context-sensitive analysis)
- ▶ code generation

Context-Sensitive Analysis

- ▶ Input:
- ▶ Output:

Rebuilding the parse tree from .wli file

- ▶ leaves correspond to terminals
- ▶ recall A7P7

Possible errors

If a program is syntactically valid, what else can go wrong?

Solving the “variable declaration” problems

We have already.

Solving the “duplicate variable” problem

Solving the “undeclared variable” problem

More advanced context-sensitive analysis

“Real” languages are much more complex.

- ▶
- ▶
- ▶
- ▶

Testing

- ▶ Make a script!
- ▶ Store expected output in a file and automatically diff against it.
- ▶ Start making many, small test cases now.
- ▶ Write a “wrapper” script to test all your tests as you progress along.

Code Generation

- ▶ Input:
- ▶ Output:

Number of different outputs:

Code Generation Issues

- ▶ Correctness
- ▶ Ease of writing compiler
- ▶ Efficiency of the compiler
- ▶ Efficiency of the compiled code

Code Generation via Syntax-Directed Translation

Fundamental idea:

Two simple rules

$S \rightarrow \text{BOF procedure EOF}$

$\text{procedure} \rightarrow \text{INT WAIN LPAREN dcl COMMA dcl COMMA...}$

Two more simple rules

$\text{expr} \rightarrow \text{term}$

$\text{statements} \rightarrow \text{statements statement}$

One more rule (A9P2)

factor \rightarrow LPAREN expr RPAREN

- ▶ What this gives us:
- ▶ The writing on the wall

A9P3: Full expressions

- ▶ `int wain(int a, int b) { return a+b; }`
- ▶ `int wain(int a, int b) { return a+b+a; }`

A9P3: Parse Tree

A9P3: Pseudocode

A9P4: Printing

statements \rightarrow PRINTLN LPAREN expr RPAREN SEMI

A first attempt:

```
return code(expr) +  
    move $3 to $1 +  
    call print
```

Problems:

Solving the current (and future) problems

Create some conventions: see website for one possible set.

Other advice

- ▶ comments
- ▶ rule-based structure of your code
- ▶ read the *WHOLE* assignment before beginning and *PLAN* ahead
- ▶ test, test, test