CS 536
Programming Assignment P6
Hand-In Report
Nikhil Kumar
ID: 9073489388

CURRENT IMPLEMENTATION STATUS

For each ASTNode, multiple combinations were tested using both literal values and identifiers. No unexpected errors have been observed. The code generation does not support structs or any associated operations.

Please look at the instructions section below in order to correctly execute aspects of the overall project. This project will take in an input file for the *Moo* language and write the appropriate MIPS assembly code to an output file, and can run the MIPS code using the spim simulator.

Heavily nested function calls have not been tested thoroughly for, however.

INCLUDED FILES

Files to be included in this assignment are as follows:

- ast.java – File containing AST creation functions, as well as name analysis, type analysis, unparsing, and code generation functions
- Codegen.java – Contains methods for writing proper MIPS assembly code to a file
- configures.sh – Configuration script for environment
- DuplicateSymException.java – Necessary exception
- EmptySymTableException.java – Necessary exception
- ErrMsg.java – For reporting any type analysis errors
- Makefile – Makefile for project; usage found below
- moo.cup – JavaCUP specification
- moo.jlex – Jlex specification
- P6.java – Test program that parses input file (test.moo), performs name analysis, performs type checking, and generates corresponding MIPS assembly code in an output file (test.s)
- SemSym.java – Symbol specifications
- SymTable.java – Symbol table
- test.moo – Input file containing correct *Moo* language code
- test.s – Output file containing corresponding MIPS assembly code
- Type.java – Type object used to determine expression type during type checking phase

INSTRUCTIONS

To correctly run the project, navigate to the current working directory with the above files. Then execute the following commands:

- make clean – remove any files generated from old code
- make – create project
- make test – uses test.moo as input; file will correctly parse and pass name analysis and type checking, and corresponding MIPS assembly code will be written to an output file named test.s
- make run – uses test.s as input; spim command line simulator will execute the instructions given in test.s

MODIFICATIONS MADE

The main modifications made reside within the ast.java file.

- Static fields added to an ASTnode object
  - String exitLabel
    - Only one set of code is generated for a function exit; this is the label corresponding to that code
  - int localOffset
    - Used during name analysis in order to compute the relative offset of each variable for each scope
    - Is 0 for global variables
  - HashMap<String, String> stringLits
    - Used to prevent duplicate string literals being stored
- Functions added to an ASTnode object
  - void codeGen()
    - Used to generate code for specific node; is blank in the superclass ASTnode to prevent compiler errors, and overridden in the appropriate ASTnode subclasses
- Name Analysis
  - Computes offsets for each SemSym
  - Also sets these offsets as fields in the appropriate ASTnodes for ease of access

Smaller modifications reside within the Codegen.java file.

- Static fields added
  - final String T2 = “$t2”
    - Used in the Unary Minus node to help perform double subtraction (although this could have been avoided)

SemSym.java was modified to contain an offset field.

- Getter and setter functions for the offset field were added

P6.java was changed such that the file does not unparse, but appends the generated code. The output file is closed in here.