CS335

Milestone IV

Course Instructor :

Dr. Swarnendu Biswas

Submitted By :

Nishant Roshan (200643)
Akash Biswas (200074)

Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

April 21, 2023

Development environment and tools used:

The program was developed in a system running **Ubuntu 22.04** with **ANTLR 4.7.2** installed. Python language was used to perform the listener traversal of the AST generated by antlr. The following python packages were used:

1. antlr - To prepare AST and do lexical and syntax analysis.
2. argparse - To take and parse argument flags from the command line.

Execution Instructions:

To install ANTLR4:
$ sudo apt install antlr4

To install the dependencies:
$ make req

To compile the lexer and parser:
$ make all

To execute and get the CSV dump of symbol tables and the text file containing the 3AC code (tac.txt), run:
$ python3 ./bin/main.py -input <input>.java -output <output>.txt

NOTE: The output file and output argument are optional. Provide the output argument to get the 3AC code dump in file with required name, else we get the 3AC code in out.txt by default if no output file and no output argument is provided. Also we get the **out.s** file which contains the **x86_64** code. **No manual changes** are required to compile the generated x86_64 code.

To compile the **x86_64** file,
$ gcc out.s

To runt he generated binary,
$ ./a.out

Functionalities implemented:

We have implemented a control stack which maintains a record of the function's local data, activation record, temporaries used and return address. When a function completes, we pop its activation record. Also, for local variable declaration, we first assign to variable to a temporary and then using load, we load the value into the temporary. In case of store, we use store instead.Precisely we have supported features for the following:

1. Support for all arithmetic binary operations like + - * / and unary operands like ++ and −−
2. Support for method call

3. Method call without parameters
4. Support for system.out.println()
5. Support for while and for loops
6. Support for conditional if else statement
7. Support for nested loops
8. Space for saved registers
9. Space for local variables

Functionalities NOT implemented:

1. Global variable: In x86_64, the global variables were declared for each instance in the above section. Differentiating the global variable from local variables was one of the concerns apart from its declaration and access in the code. Also tried calling variable from outside method but were presented garbage value.
2. Method with parameters: In the 3AC code generation, we lacked the support for parameters during function call. We vaguely implemented it and hence were unable to extend to the assembly code generation.
3. Array: We lacked support for array declaration and access in the 3AC code generation.
4. Support for multiple data types: For all operations we have used quad data type. While the structure of code remains same, we did not extend it for data type like float and boolean because we were not able to infer from it in the 3AC

Sample test cases and their description

1. **test_1.java**
   This is a simple java program to show the program execution and x86 assembly formation for method invocation inside functions.
2. **test_2.java**
   This is a sample java program to demonstrate invocation of method inside another method and illustrate print statement.
3. **test_3.java**
   This code is used to show the implementation of expressions and simple if else.
4. **test_4.java**
   This codes shows the execution of a for loop.
5. **test_5.java**
   This test case demonstrates the execution of while loops.
6. **test_6.java**
   This test case demonstrates the execution of complicated expressions in java.

7. **test_7.java**
   This test case demonstrates the execution nested for loops in java.
8. **test_8.java**
   This code illustrates the implementation of for loops inside if else construct.
9. **test_9.java**
   This test presents the example of function invocation inside if else constructs.
10. **test_10.java**
    This test case demonstrates the use of all the operators and parenthesis in java.

Member information			
Member Name	Roll number	Email id	Contribution
Akash Biswas	200074	abiswas20@iitk.ac.in	50%
Nishant Roshan	200643	nishantr20@iitk.ac.in	50%

.*Thank You*