Report
Compiler Design
comp-6421

Assignment 5
Code Generation

Professor: Dr. Joey Paquet

Neona Sheetal Pinto
40172626

1. Memory allocation	*Status*
1.1 Allocate memory for basic types (integer, float).	X
1.2 Allocate memory for arrays of basic types.	X
1.3 Allocate memory for objects.	X
1.4 Allocate memory for arrays of objects.	X
2. Functions	
2.1 Branch to a function's code block, execute the code block, branch back to the calling function.	X
2.2 Pass parameters as local values to the function's code block.	X
2.3 Upon execution of a return statement, pass the return value back to the calling function.	X
2.4 Call to member functions that can use their object's data members.	X
3. Statements	
3.1 Assignment statement: assignment of the resulting value of an expression to a variable, independently of what is the expression to the right of the assignment operator.	X
3.2 Conditional statement: implementation of a branching mechanism.	X
3.3 Loop statement: implementation of a branching mechanism.	X
3.4 Input/output statement: execution of a keyboard input statement should result in the user being prompted for a value from the keyboard by the Moon program and assign this value to the parameter passed to the input statement. Execution of a console output statement should print to the	X

Moon console the result of evaluating the expression passed as a parameter to the output statement.	
4. Aggregate data elements access	
4.1. For arrays of basic types (integer and float), access to an array's elements.	X
4.2. For arrays of objects, access to an array's element's data members.	X
4.3. For objects, access to members of basic types.	X
4.4. For objects, access to members of array or object types.	X
5. Expressions	
5.1. Computing the value of an entire complex expression.	X
5.2. Expression involving an array factor whose indexes are themselves expressions.	X
5.3. Expression involving an object factor referring to object members.	X

Design

The visitor package has 2 new visitors namely :

- *ComputeMemSizeVisitor* class which is used to compute the memory and create an offset for each class data member, function member, data variables as well as for the literals which are used.
- *StackBasedVisitor* class is used to generate moon code for simple expressions and assignment and put statements, including code for function calls using a stack-based model.
- *CodeGeneration* class is responsible for dealing with every file having the StackBased visitor objects to do tree traversal starting from the root node(progNode).
- *ComplierDriver* class is a driver which opens every file(.src) in a folder or a single file(.src), first creates a SyntacticAnalyzer object and then a SemanticAnalyzer object to handle the task of every visitor and output reports to files along with the CodeGeneration object to handle the final visitor node.

Tools

1. ***IntelliJ IDEA***: IDE platform for writing the source code. The IDE has smooth, efficient and clever code suggestions which makes it easy to type. Debugging is much faster compared to other IDE's.
2. ***Java.util***: Used util files in the source code for creating tables, hashmaps,. Etc.
3. ***Visitor pattern***: Used for following the code example the professor provided, which gives a general idea to implement the pattern in this assignment. Since different semantic actions are needed to do semantic checking and the next assignment also contains semantic actions, this pattern separates different groups of operations, making the implementation much easier.