COMP 442: Compiler Design

Assignment 4: Code Generation

Done by:

Abdulla Alhaj Zin, 40013496

Date:

17/04/2020

To:

Dr. Joey Paquet

Section 1: Analysis, List of implemented items

Implementation:

1. Memory allocation
    - 1.1 Allocate memory for basic types (integer, float).
        - Implemented
    - 1.2 Allocate memory for arrays of basic types.
        - Implemented
    - 1.3 Allocate memory for objects.
        - Implemented
    - 1.4 Allocate memory for arrays of objects.
        - Implemented
2. Functions
    - 2.1 Branch to a function's code block, execute the code block, branch back to the calling function.
        - Implemented
    - 2.2 Pass parameters as local values to the function's code block.
        - Implemented
    - 2.3 Upon execution of a return statement, pass the return value back to the calling function.
        - Implemented
    - 2.4 Call to member functions that can use their object's data members.
        - NOT Implemented

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.
        - Implemented
    - 3.2 Conditional statement: implementation of a branching mechanism.
        - Implemented
    - 3.3 Loop statement: implementation of a branching mechanism.
        - Implemented
    - 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 Moon console the result of evaluating the expression passed as a parameter to the output statement.

Implemented

4. Aggregate data members access

4.1 For arrays of basic types (integer and float), access to an array's elements.

NOT Implemented

4.2 For arrays of objects, access to an array's element's data members.

NOT Implemented

4.3 For objects, access to members of basic types.

NOT Implemented

4.4. For objects, access to members of array or object types.

NOT Implemented

5.Expressions

5.1 Computing the value of an entire complex expression.

Implemented

5.2 Expression involving an array factor whose indexes are themselves expressions.

NOT Implemented

5.3 Expression involving an object factor referring to object members.

NOT Implemented

Section 2: Design

The code generation phase depended on the AST generated in the syntactical analyzer along with the symbol table from the semantic analyzer phase. This phase is the most important phase in the compiler as all the components results are combined to produce an executable and useful program.

In this assignment, the tag-based memory system was implemented. To preserve uniqueness, special criteria were applied in the naming in addition to static numberings that are shared amongst all classes that would increment on every new variable or temporary variable. The code generator was implemented using the visitor design pattern as well, like the symbol table generator and AST generator. All generated code is moon code and the conventions of the code has been followed as to structure the file into sections and add comments.

Section 3: Use of Tools

Moon processor:

https://www.gel.usherbrooke.ca/khoumsi/Teaching/engineer/GEI_443/Moon.pdf

Moon libraries for getInt and putInt:

https://www.gel.usherbrooke.ca/khoumsi/Teaching/engineer/GEI_443/Moon.pdf

GCC-core for windows to compile moon:

https://cygwin.com/packages/summary/gcc-core.html

Built-in C# libraries and unit tests.
