Building a Compilator for Java code

J. Agustín Barrachina
IEEE Student Member
École Polytechnique
Universit Saclay-Paris

***Abstract*—In this project, a compilator was created to generate a x86-64 assembler code from a C fragment called mini-C. This is a 100% C-compatible fragment, in the sense that any Mini C program is also a C program.**

I. Introduction

"Optimizing compilers are so difficult to get right that we dare say that no optimizing compiler is error-free! Thus, the most important objective in writing a compiler is that it is correct" [1]

A. Structure of a Compiler

II. Lexical Analyzer

A lexical analyzer (*Lexer*) is the first front-end step in compilers, matching keywords, comments, operators, etc, and generating a token stream for parsers called *lexemes*. The Lexer reads input from the programming language to compile (mini c in our case) and matches it against regular expressions and runs a corresponding action if such an expression is matched.

To make the Lexer are going to use:

- Regular Expresions: To describe the lexemes
- Finite Automata: To recongnize the expresions

A. Regular Expresions

The concept of regular expression arose in the 1950's when the American mathematician Stephen Cole Kleene formalized the description of a regular language.

A regular expression is a sequence of characters that define a search pattern. In other words, there are a conjunction of letters and digits that follow a certain rule.

Let us define *¡letter¿* as any letter in the Latin alphabet and *¡digit¿* any number [0-9]. Then we can define rules as follow:

$$0|[1-9](<digit>*|[]) \tag{1}$$

The corresponding action will be just to generate an abstract symbol (normally called "token name") that can be read by the next phase (Syntax Analyzer).

For the lexical analyzer, a flex library was used [2]. A .flex file was created and then, by means of jflex, converted to the final java class.

III. Syntax Analyzer

IV. Semantic Analyzer

V. Code Generation

VI. Conclusion

Acknowledgment

References

[1] A. V. A. M. S. L. R. S. J. D. Ullman, *Compilers. Principles, Techiniques & Tools*, 2nd ed. Addison Wesley, 2006.
[2] J. Team. (2015, mar) Jflex - the fast scanner generator for java. [Online]. Available: http://jflex.de/