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" (Dragon Book, 2006)

A. Structure of a Compiler

II. Lexical Analyzer

A lexical analyzer is the first front-end step in compilers, matching keywords, comments, operators, etc, and generating an input token stream for parsers. The so called 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. 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. 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

References