Processamento de Linguagens e Compiladores (3º Ano LCC)

Project 2

Project Report

Bruno Dias da Gião
A96544
a96544@uminho.pt

Maria Filipa Rodrigues
A97536
a96536@uminho.pt

December 23, 2022

Resumo

O processo de compilação de uma linguagem de programação é um problema de extrema importância e de elevada complexidade. Através de Yacc adaptado a Python, o Projeto que este relatório documentará demostrará o processo do reconhecimento de uma linguagem inspirada em ANSI C e da geração de codigo de uma máquina virtual de stack, relativamente mais simples que uma máquina real, a partir dessa linguagem.

Abstract

The process of compiling a programming language is a problem of extreme importance and of increased difficulty. Through the use of Python adapted Yacc, the Project this report intends to document will demonstrate the process of recognizing an ANSI C inspired language and the generation of code for a stack based Virtual Machine, which is relatively simpler than a real machine, from the created language.

Contents

1 Report **2**
1.1 Introduction . . . 2
1.1.1 The “Not-Quite-C” language Compiler . . . 2
1.2 Methodology . . . 3
1.2.1 Theoretical Background . . . 3
1.2.2 Practical component . . . 3
1.3 Analysis . . . 3
1.3.1 Expected Results . . . 3
1.3.2 Testing the generated code . . . 3
1.4 Conclusion . . . 3
1.4.1 Future Work . . . 3

2 Appendix **4**
2.1 Codes . . . 4

Chapter 1

Report

§1.1 Introduction

1.1.1 The "Not-Quite-C" language Compiler

Introduction to the Report

The present document introduces a program that compiles text from a simplified version of C, as according to the C99 standard, into a stack based virtual machine that can be accessed via the World Wide Web [2] or in any UNIX-based machine [6].
The current chapter, chap:1, was structured with some goals in mind, where each section is representative of such goals:

1. In the Introduction, §1.1, we introduce and provide context to both the report and the project as it is presented.
2. In the Methodology section, §1.2, we present some contextualizing theory, the thought process that guided the elaboration of the Project and the State of the Art itself as it is presented. This is done with the hope of helping the reader best understand how this project was solutioned.
3. With the Analysis section, §1.3, we hope to "prove" very loosely but with inteligently chosen examples that show the correct functioning of the developed compiler.

   Note, we chose to limit our proof of compiler correctness to well chosen examples as the Formally correct method of verifying a compiler is a well known complex and extensive problem, resulting, thus in long proofs by derivation that would steal from the purpose of this report and far exceed the scope of the project. Thus, such a Formal Verification is best left for future work. [4] [5]
4. This chapter finishes with a Conclusion, §1.4, where we deem this report as terminated,as is costumary for any document of this format, with our thoughts on our work and future work considerations.

Following the report, this document comes annexed with the source code for the project's solution, chap:2.
As is costumary, a bibliography is also annexed at the very end of the present document.
In order to ease reader comprehension of this report, we have opted by the paradigm of "Literate Programming" in which the code shall always be accompanied by an explanation of the code wherever it is deemed necessary. In practice, what happens is, whenever a code segment is referenced it shall be presented as is in the code and a detailed explanation shall follow, in such a way where understanding of the program comes from reading directly the code and reading our thought process and explanations.

Historical background of the ALGOL-58 family, UNIX and C

While the first programming language was indeed FORTRAN, however, between FORTRAN and C, the differences are immense, so, in order to best analyse the history of this language we must look to ALGOL-58, Algorithmic Language.
ALGOL-58, a standard developed in 1958, one year after FORTRAN by the Association for Computing Machinery and has had 2 major revisions, ALGOL-60 and ALGOL-68, the latter of which was met with severe criticism [3], mainly due to it being compared to it's predecessor, which is the Language we shall be analysing, ALGOL-60.

Historical background of Lexx, Yacc and PLY

Importance of this project

Background of the Project

Expansions of the Project

§1.2 Methodology

1.2.1 Theoretical Background

1.2.2 Practical component

§1.3 Analysis

1.3.1 Expected Results

1.3.2 Testing the generated code

§1.4 Conclusion

1.4.1 Future Work

Chapter 2

Appendix

§2.1 Codes

Bibliography

[1] EPLDIUM. Portuguese Documentation for the Virtual Machine VM. https://eplmediawiki.di.uminho.pt/uploads/Vmdocpt.pdf.

[2] EPLDIUM. Web version of the Virtual Machine VM. https://ewvm.epl.di.uminho.pt.

[3] C Hoare. Critique of ALGOL 68. *ALGOL Bulletin*, 29:27–29, 1968.

[4] Xavier Leroy. Formal Verification of a Realistic Compiler. *Communications of The ACM*, 52(7):107–115, 2009.

[5] Xavier Leroy. Formally verifying a compiler: what does it mean, exactly? Talk at ICALP, 2016.

[6] Christine Paulin-Mohring. Source for the Virtual Machine VM. https://www.lri.fr/~paulin/COMPIL/introduction.html.