Laboratory 3 - Performance Report

Israel Castro Gonzalez - A01228769

***Abstract*— This report shows the difference in performance between a simple lexer of the AC language (written in C) and another created with Lex.**

I. INTRODUCTION

The lexers scan a language called adding calculator (AC). I'll be using the following informal definition for AC.

In AC, there are only two data types: integer and float. An integer type is a sequence of decimal numerals, as found in most programming languages. A float type allows five fractional digits after the decimal point.

We have three reserved keywords, each limited for simplicity to a single character:

- f - declares a float variable.
- i - declares an integer variable
- p - prints the value of a variable.

In this AC we can have 23 possible variable names, drawn from the lowercase Roman alphabet and excluding the three reserved keywords f, i, and p. Variables must be declared prior to using them.

The definition of our AC tokens are the following:

Terminal	Regular Expression
floatdcl	f
intdcl	i
print	p
id	[a-eghj-oq-z]
assign	"="
plus	"+"
minus	"-"
multiplication	"*"
division	"/"
inum	[0-9]+
fnum	[0-9]+"."[0-9]{1,5}
comment	[/][/].*\n

II. PROBLEM DESCRIPTION

Lexical Analysis is the first phase of a compiler, this one converts the input program into a sequence of tokens. Having two lexers, one written in C and the other one created using Lex, which one proves to be more optimal in processing time and what are the factors that make the difference if there is any.

III. SOLUTION

A. Lexer in C

To make the scanning of characters more optimal than just using "if/else", we use a switch-case statement to identify the terminals.

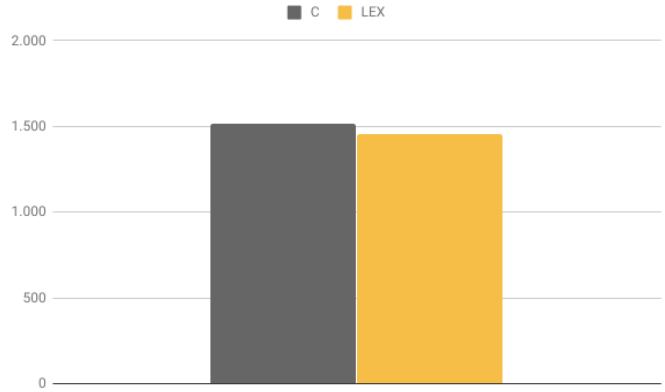
B. Lexer with Lex

With Lex, you just have to define the regular expression rules for the tokens and how they are to be processed. Lex then generates a complete scanner coded in C.

C. Evaluation

To evaluate the different performance in time between the two lexers, we are going to use an AC code_generator written in python, provided by the professor Victor Rodriguez, that creates a stress example with 600,000 lines of code. We use this random output as input for both lexers using the `time` command in the terminal to get the real time used to scan the AC program.

IV. RESULTS



The C Lexer's real time while scanning the stress AC code was of 1.518 seconds, while the Lex Lexer took 1.456 seconds. The difference is of 0.062 seconds.
This can be visualized in the following graph:

V. CONCLUSIONS

The difference is rather insignificant, but it's still a difference, this of course could vary depending on the system build that is being tested on. Creating the lexer with Lex proved to be less time consuming than programming it in C. We can conclude that the lexer produced with **Lex has a better performance than the one programmed in C**.

REFERENCES

[1] C. N. Fischer, R. K. Cytron & R. J. LeBlanc. Crafting a Compiler, 2nd ed.,Boston: Pearson Education, 2010.