Performance Results for Compilers' Lab 4

Miguel Angel Montoya Zaragoza - A01226045

***Abstract*— The purpose of this report was to compare the results of executing a lexical analyzer made completely in C and a lexical analyzer made using LEX/FLEX.**

I. INTRODUCTION

We were given the task of creating a lexical analyzer for our class language called ac (For adding calculator). The professor aided us by creating a Python script that generates a .ac file for testing ad a longer one for stress-testing.

II. PROBLEM

The problem it's to create the lexical analyzer using FLEX. The usage of Regular Expression is essential, as simple ifs and else won't serve as much. Plus, we need to find a way to read from a file (input stream) and write to a file (Output tokens) from a FLEX file.

The reason we are looking to output the tokens to a file instead of to console it's because the time it takes to print in console it's way longer than writing to a file (A compiler doesn't outputs the tokens it gathers from code).

III. SOLUTION

14 REGEXES were used to match declarations, functions and operations. REGEXES we can remark are the match to a newline using the regex [\r\n] and to blank space [ ].

It was very important to match the newline and blank space as ignoring them would make LEX to output blank spaces and newline to console, instead of to the file. This would make the output file wrong and it would make the file take more time because of the print to console.

It's necessary to point out that a re-declaration of the function yylex() was necessary. The reason for this is that yylex doesn't has in its declaration a FILE parameter. Given that the code to execute if a REGEX matched was to write to a file, a redeclaration was in order.

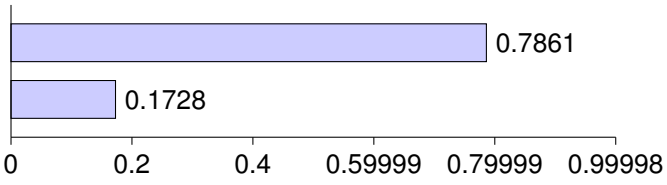
```
#define YY_DECL int yylex(FILE *outFile)
.
.
.
FILE *outFile = fopen("lex.out", "w");
yylex(outFile);
fclose(outFile);
```

IV. RESULTS

Both the lexical analyzer in C and the lexical analyzer in FLEX were provided with the same generated .ac code generated by the python script. After that, the codes were compiled and executed 10 times so we could get an average of the data.

C lexical analyzer	FLEX lexical analyzer
0.862	0.215
0.688	0.172
0.828	0.129
0.832	0.156
0.799	0.178
0.715	0.177
0.730	0.173
0.807	0.204
0.800	0.161
0.800	0.163
Average	
0.7861	0.1728

As you can observe, in average, the lexical analysis made with C it's 4.5 times slower than the one made with FLEX.



V. CONCLUSIONS

It's clear the superiority of LEX/FLEX when trying to process input streams. Even though the C code used for comparison already used Regular Expressions, FLEX it's quite superior.