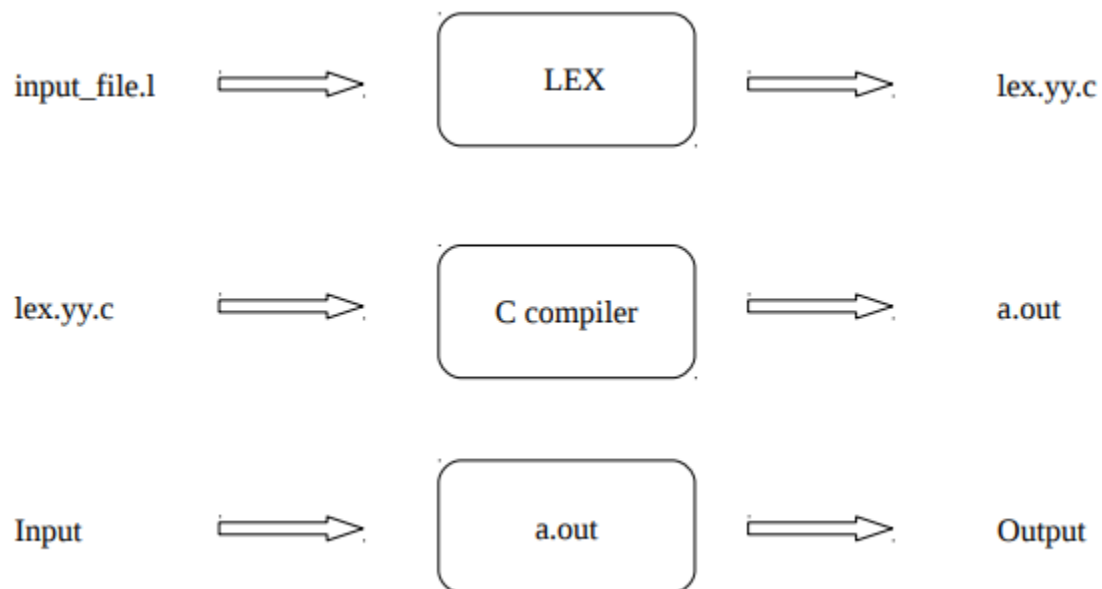## 1. Introduction to LEX

LEX is a tool used to generate a lexical analyzer. This document is a tutorial for the use of LEX for SIL Compiler development. Technically, LEX translates a set of regular expression specifications (given as input in input_file.l) into a C implementation of a corresponding finite state machine (lex.yy.c). This C program, when compiled, yields an executable lexical analyzer.

input_file.l	⇒	LEX	⇒	lex.yy.c
lex.yy.c	⇒	C compiler	⇒	a.out
Input	⇒	a.out	⇒	Output

C compiler = GCC in your case

2. The structure of LEX programs

A LEX program consists of three sections : Declarations, Rules and Auxiliary functions

```
DECLARATIONS
%%
RULES
%%
AUXILIARY FUNCTIONS
```

Example of LEX program

\\ Declarations
```
%{
#include <stdio.h>
%}
```

\\Rules
Rules in a LEX program consist of two parts:
i. The pattern to be matched
ii. The corresponding action to be executed

```
%%
(go|GO|gO|eat|EAT|eAT)  { printf("Verbs: %s\n", yytext); }
[0-9]+                  { printf("NUMBER: %s\n", yytext); }
[a-zA-Z]+               { printf("WORD: %s\n", yytext); }
[ \t\n]+                ;  /* ignore whitespace */
.                       { printf("others: %s\n", yytext); }
%%
```

\\ Auxiliary functions
```
int main(void)
{
    yylex();
    return 0;
}

int yywrap(void)
{
    return 1;
}
```

References :
1. https://silcnitc.github.io/lex.html#navintro
2. https://cse.iitkgp.ac.in/~bivasm/notes/LexAndYaccTutorial.pdf
3.