Compiler Design

Lex & Flex

Lexical Analyzer Generator

- **Lex is a program generator designed for lexical processing of character input streams.**
- **Lex source is a table of regular expressions and corresponding program fragments.**
- **The table is translated to a program which reads an input stream, copying it to an output stream and partitioning the input into strings which match the given expressions.**
- **The compiler writer uses specialized tools that produce components that can easily be integrated in the compiler and help implement various phases of a compiler.**

Contd…..

- The program fragments written by the user are executed in the order in which the corresponding regular expressions occur in the input stream.
- Lex can generate analyzers in either “C” or “Ratfor”, a language which can be translated automatically to portable Fortran.
- Lex is a program designed to generate scanners, also known as tokenizers, which recognize **lexical** patterns in text.
- Lex is an acronym that stands for "**lexical analyzer generator**.
- It is intended primarily for Unix-based systems.
- The code for Lex was originally developed by Eric Schmidt and Mike Lesk.

Contd……

- Lex can be used with a parser generator to perform lexical analysis.
- It is easy, for example, to interface Lex and Yacc, an open source program that generates code for the parser in the C programming language.
- Lex can perform simple transformations by itself but its main purpose is to facilitate lexical analysis.
- The processing of character sequences such as source code to produce symbol sequences called tokens for use as input to other programs such as parsers.

Contd……

Processes in lexical analyzers

- Scanning
  - Pre-processing
    - Strip out comments and white space
    - Macro functions
- Correlating error messages from compiler with source program
  - A line number can be associated with an error message
- Lexical analysis

Contd……

Terms of the lexical analyzer

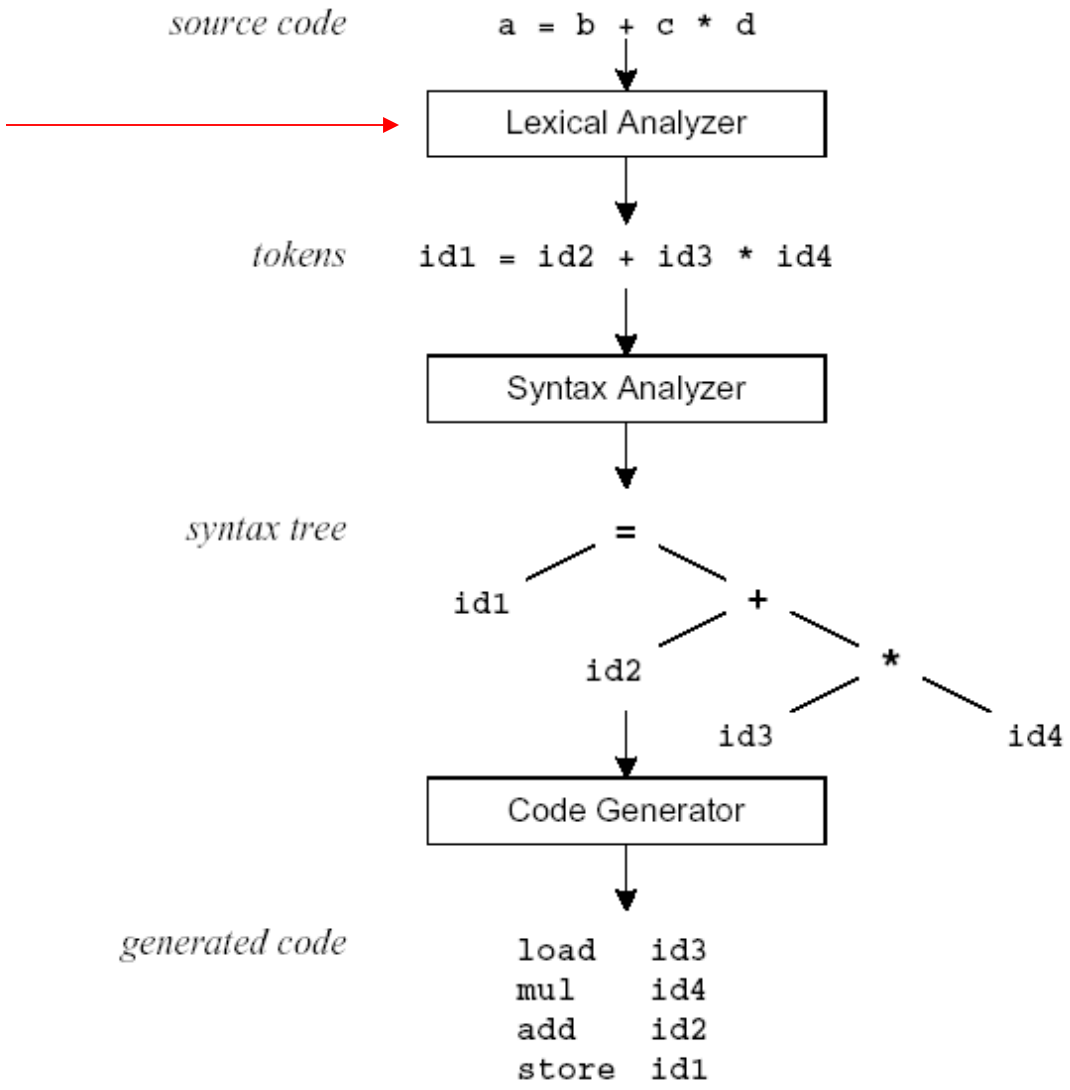
- Token
  - Types of words in source program
  - Keywords, operators, identifiers, constants, literal strings, punctuation symbols(such as commas, semicolons)
- Lexeme
  - Actual words in source program

Contd…..

- Pattern
  - A rule describing the set of lexemes that can represent a particular token in source program
  - **Relation** {<.<=,>,>=,==,<>}

  - Lexical Errors
- Deleting an extraneous character
- Inserting a missing character
- Replacing an incorrect character by a correct character
- Pre-scanning

Compilation Sequence



What is Lex?

- The main job of a *lexical analyzer (scanner)* is to break up an input stream into more usable elements (*tokens*)

```
a = b + c * d;
```

ID ASSIGN ID PLUS ID MULT ID SEMI

- Lex is an utility to help you rapidly generate your scanners

Lex – Lexical Analyzer

- **Lexical analyzers tokenize input streams**
- **Tokens are the terminals of a language**
  - **English**
    - **words, punctuation marks, …**
  - **Programming language**
    - **Identifiers, operators, keywords, …**

- **Regular expressions define terminals/tokens**
- **Some examples are**
- Flex lexical analyser
- Yacc
- Ragel
- PLY (Python Lex-Yacc)

Contd…..

- Lex turns the user's expressions and actions (called source in this memo) into the host general-purpose language.
- the generated program is named yylex.
- The yylex program will recognize expressions in a stream (called input in this memo) and perform the specified actions for each expression as it is detected. See the below fig….

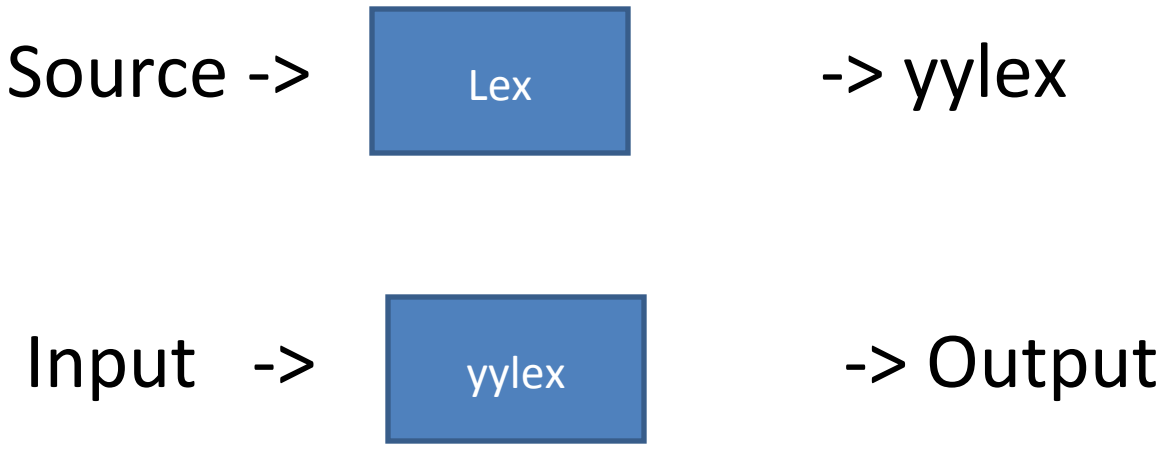
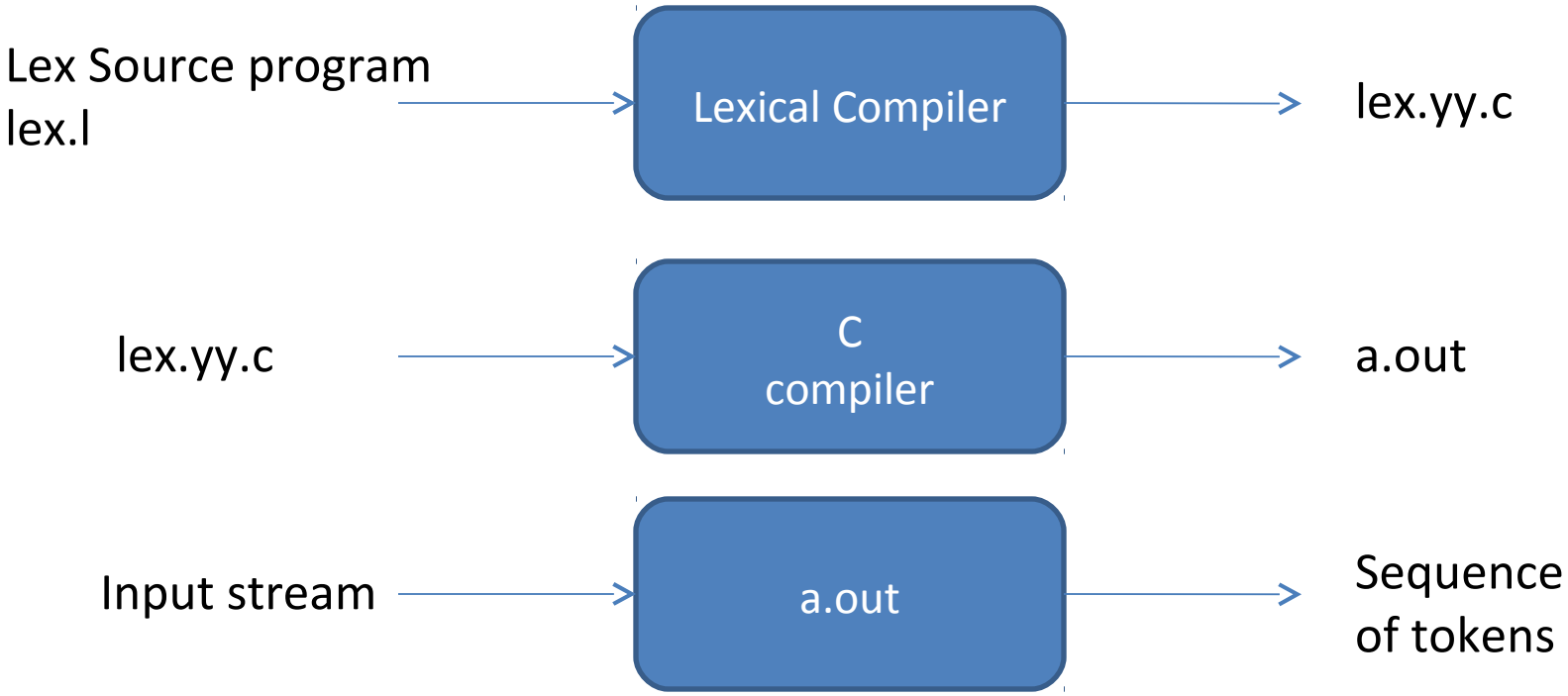


Fig. An overview of Lex

yylex()

- It implies the main entry point for lex.
- Reads the input stream generates tokens, returns 0 at the end of input stream.
- It is called to invoke the lexer or scanner.
- Each time yylex() is called, the scanner continues processing the input from where it last left off.
- Eg:
- yyout
- yyin
- yywrap

Lexical Analyzer Generator - Lex



Structure of a Lex file

- The structure of a Lex file is intentionally similar to that of a yacc file.
- files are divided into three sections, separated by lines that contain only two percent signs, as follows:
- ***Definition section***

%%

- ***Rules section***

%%

- ***C code section***

Contd…..

- The **definition** section defines macros and imports header files written in C. It is also possible to write any C code here, which will be copied verbatim into the generated source file.

- The **rules** section associates regular expression patterns with C statements. When the lexer sees text in the input matching a given pattern, it will execute the associated C code.

- The **C code** section contains C statements and functions that are copied verbatim to the generated source file. These statements presumably contain code called by the rules in the rules section. In large programs it is more convenient to place this code in a separate file linked in at compile time.

Example

- The following is an example Lex file for the flex version of Lex.
- It recognizes strings of numbers (positive integers) in the input, and simply prints them out.

```
/*** Definition section ***/
 %{
 /* C code to be copied  */
#include <stdio.h>
%}
 /* This tells flex to read only one input file
*/
%option noyywrap
%%
```

Contd……

```
/*** Rules section ***/
/* [0-9]+ matches a string of one or more digits */
[0-9]+  {
        /* yytext is a string containing the matched text.
   */
        printf("Saw an integer: %s\n", yytext);
         }
.|\n     { /* Ignore all other characters. */
    }
 %%
/*** C Code section ***/
```

Contd…..

```
int main(void)
{
        /* Call the lexer, then quit. */
       yylex();
        return 0;
  }
```

If this input is given to flex, it will be converted into a C file, lex.yy.c.

This can be compiled into an executable which matches and outputs strings of integers. Eg.

abc123z.!&*2gj6

the program will print:

Saw an integer: 123
Saw an integer: 2
Saw an integer: 6

Flex, A fast scanner generator

- flex is a tool for generating scanners:
- programs which recognized lexical patterns in text.
- flex reads the given input files,
- or its standard input if no file names are given, for a description of a scanner to generate.
- The description is in the form of pairs of regular expressions and C code, called rules.
- flex generates as output a C source file, `lex.yy.c', which defines a routine `yylex()'.
- This file is compiled and linked with the `-lfl' library to produce an executable.
- When the executable is run, it analyzes its input for occurrences of the regular expressions.
- Whenever it finds one, it executes the corresponding C code.

Contd…

- Flex (**fast lexical analyzer generator**) is a free and open-source software alternative to lex.
- It is a *computer program* that generates *lexical analyzers* (also known as "scanners" or "lexers").
- Flex was written in C by Vern Paxson around 1987.
- He was translating a ***Ratfor*** generator.
- It had been led by Jef Poskanzer.
- The tokens recognized are: '+', '-', '*', '/', '=', '(', ')', ',', ';', '.', ':=', '<', '<=', '<>', '>', '>=';
- numbers: 0-9 {0-9}; identifiers: a-zA-Z {a-zA-Z0-9} and keywords: begin, call, const, do, end, if, odd, procedur e, then, var, while.

Thanks