Report

CSE 3212 : COMPILER DESIGN LABORATORY

submitted to:

Dola Das Lecturer Department of Computer Science and Engineering Khulna University of Engineering and Technology, Khulna	Md. Ahsan Habib Nayan Lecturer Department of Computer Science and Engineering Khulna University of Engineering and Technology, Khulna

Submitted by:

Md. Likhon Sarker

Roll: 1607069

Third year, second semester

Department of computer science and engineering

Khulna university of engineering and technology, khulna

Submission date: 08-06-2021

LEX:

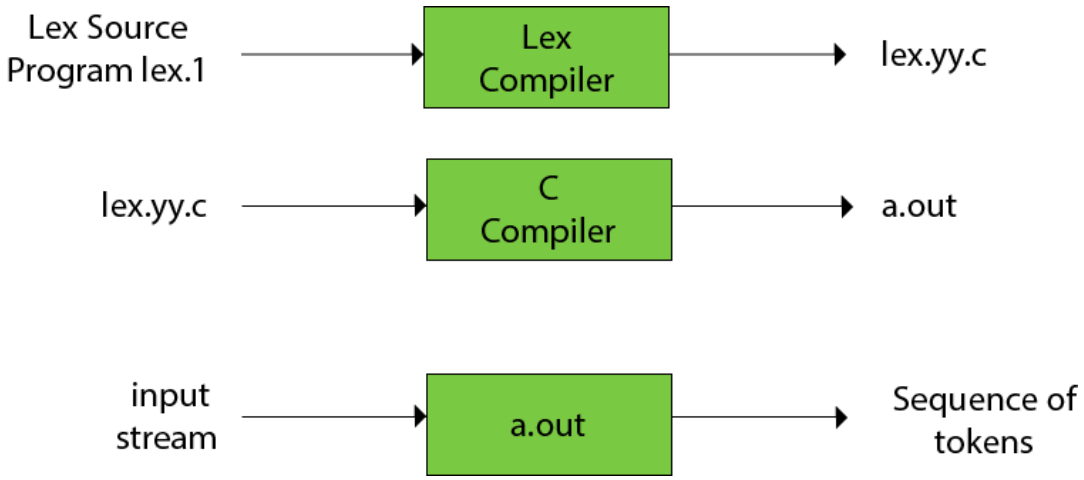
Lex is a program that generates lexical analyzer. It is used with YACC parser generator. The lexical analyzer is a program that transforms an input stream into a sequence of tokens. It reads the input stream and produces the source code as output through implementing the lexical analyzer in the C program.

The function of Lex is as follows:

1.Firstly lexical analyzer creates a program. suppose its lex.1 in the Lex language. Then Lex compiler runs the lex.1 program and produces a C program lex.yy.c.

2.Finally C compiler runs the lex.yy.c program and produces an object program a.out.

3.a.out is lexical analyzer that transforms an input stream into a sequence of tokens.



Bison:

Bison is a general-purpose parser generator that converts a grammar description for an LALR(1) context-free grammar into a C program to parse that grammar. Bison is upward compatible with Yacc. YACC stands for **Yet Another Compiler Compiler**. YACC provides a tool to produce a parser for a given grammar. YACC is a program designed to compile a LALR (1) grammar. It is used to produce the

source code of the syntactic analyzer of the language produced by LALR (1) grammar. The input of YACC is the rule or grammar and the output is a C program.

In bison file

1.main program calls yyparse( ).

2.yyparse( ) calls yylex when it wants a token.

3.yylex returns the type of the token.

4.yylex puts the value of the token in a global variable named yylval.

5. creates 2 new files, suppose its x.tab.h and x.tab.c

6. The file x.tab.h contains declarations.

Manual:

1. Variable declaration

Rule	Details
integer a,b,value;	Variable a,b,value is created All are integer type

2. Variable initialization:

Rule	Details
integer a=5,b=20,value=100;	Variable a is initialized to 5 b is initialized to 20 value is initialized to 100

3.Assign value to a variable

Rule	Details
integer value; **value=100;**	100 is assigned to a variable named "value"

4. Arithmetic operation

Operator	Operation	Details
+	Value=a plus b	Equivalent to value=a+b
-	Value=a minus 5	Equivalent to value=a-5
*	Value=a mul b	Equivalent to value=a*b
/	Value=a div b	Equivalent to value=a/b

5. For loop

Rule	Details
lb	Left brace '{'
rb	Right brace '}'
for_loop(a,b,c) **lb** **value=value plus 5** **rb**	For loop starts from a and continues till b, every time it will increment its value by c Inside for loop value=value+5 will be executed every time

6. While loop

Rule	Details
while_loop(var less_than num) **lb** **var= var plus 4;** **rb**	This loop will continue till variable var is less than num. inside loop, value of the variable will be incremented by 4 each time.

7. Comment

Type	Rule	Details
single line comment	comment: this is a comment	
multiple line comment	Comment lb This Is comment rb	Inside lb and rb, anything will be considered as a multiple line comment

8. If Else condition

Rule	Details
less_than	Its equivalent to '<'
greater_than	Its equivalent to '>'
if (a less_than b)	Condition inside if will be true when a<b
if (a greater_than b)	Condition inside if will be true when a>b
if(a less_than b) **lb** **a=a mul 6** **rb** **else** **lb** **a=a div b** **rb**	If a<b a will be multiplied by 6 otherwise a will be divided by b

9. Switch Case

Rule	Details
lb	Left brace '{'
rb	Right brace '}'
switch **case (a less_than b):** **a=a plus b** **case (a greater_than b):** **a=a mul b** **default: a=a div 10**	Inside switch case, there can be multiple cases. if any of the case condition become true, its corresponding expression will be executed and no other cases will be checked. If no case become true, then default case will be executed

10. Function

Rule	Details
main function ()	Whole program will be executed inside this function
print(variable)	It takes a single parameter and print its value
binary(num)	It takes a number/variable and print its binary representation
log(num)	It takes a number/variable and display its logarithmic value in base 'e'
factorial(num)	It takes a number/variable and print its factorial. N factorial=1*2*3…….*N
check_prime(variable)	It takes a number/variable and check whether it's a prime number or not
show_gcd(a,b)	It takes two number/variable and print their gcd

	gcd=greatest common divisor
show_lcm(a,b)	It takes two number/variable and print their lcm lcm=least common multiple
log10(a)	It takes a number/variable and print its logarithmic value in base '10'
sin(angle)	It takes an angle in degree and print its corresponding sin value
cos(angle)	It takes an angle in degree and print its corresponding cos value
tan(angle)	It takes an angle in degree and print its corresponding tan value
bigmod(a,b,m)	It takes three parameter ,suppose they are a,b,m. this function print (a^b)%m
get_remainder(a%b)	It takes two parameter and print remainder of these two. remainder=a%b;