Lab Report

‘Project of Simple Compiler Design Using Flex And Bison’

Course Name: Compiler Design Laboratory

Course No: CSE 3212

Submitted To:

Dola Das
Asistant Professor
Khulna University of
Engineering & Technology,Khulna

Dipannita Biswas
Lecturer
Khulna University of
Engineering & Technology,Khulna

Submitted By:

Sazid Bin Hayat
Roll: 1807092
Department of Computer Science and Engineering
Khulna University of Engineering & Technology, Khulna

Submission Date: 20 / 12/ 2022.

Introduction:

compiler is a special program that translates a programming language's source code into machine code, bytecode or another programming language. Compiler Design is the structure and set of principles that guide the translation, analysis, and optimization process of a compiler.

Flex:

Flex(fast lexical analyzer generator) is a tool for generating tokens. Flex source is a table of regular expressions and corresponding program fragments. It is a program that generates lexical analyzers. It is used with the YACC parser generator. Flex takes a program written in a combination of Flex and C, and it writes out a file (called lex.yy.c) that holds a definition of function yylex() .

Flex Specification:

Flex is written following structure given below:

```
{ definitions }
%%
 { rules }
%%
{ user subroutines }
```

Bison:

Bison is a general-purpose parser generator that converts a grammar description (Bison Grammar Files) for an LALR(1) context-free grammar into a C program to parse that grammar. The Bison parser is a bottom-up parser. It is a powerful and free version of Yacc. Grammars for yacc are described using BNF.

Bison Input File Format:

```
%{
  C declarations { types , variables , functions , preprocessorcommands  }
 %}
 /* Bison Declarations (grammar symbols , operator precedence
                          declaration , attribute data type) */
%%
  /* grammar rules go here */
%%
/* additional c code goes here */
```

Equipment:

PC, Flex , Bison , VS Code.

Procedure:

- The code is divided into two parts: flex file (.l) and bison file (.y) .
- Input expression checks the lex (f.y) file and if the expression satisfies the rule then it checks the CFG into the bison file .
- It's a bottom up parser and the parser constructs the parse tree firs ,matches the leaves node with the rules and if the CFG matches then it gradually goes to the root .
- There is an input file called "input.txt" from which inputs are gained.
- Output is shown in "output.txt".

To Run the files:

1. Bison -d f,y
2. Flex f.l
3. gcc f.tab.c lex.yy.c -o exe
4. exe

Description:

A lexical analyzer divides the input into meaningful units or Token. A token is the smallest element(character) of a computer language program that is meaningful to the compiler. The parser has to recognize these as tokens: identifiers, keywords, literals, operators,punctuators, and other separators .

The Keywords that I have used in my compiler project are given below **:**

1. **start()** = Indicates start of main function.
2. **Addlib** = To add library needed for the program
3. **Int** = For declaring integer type
4. **char** = For declaring character type
5. **Float** = For declaring float type
6. **//char//** = For declaring character type array
7. **//int//** = For declaring integer type array
8. **//float//** = For declaring float type array
9. **plus** = This is used for addition operation
10. **minus** = This is used for subtraction operation
11. **mul** = This is used for multiplication operation
12. **div** = This is used for division operation
13. **++** = Increment operation
14. **--** = Decrement operation

15.^ = Power raise to operation

16.rem = Modulus or Finding remainder

18.gt = ‘Greater than’ relational operator

19.gtoe = ‘Greater than or equal’ relational operator

20.lt = ‘Less than’ relational operator

21.ltoe = ‘Less than or equal’ relational operator

22.equal = ‘Equal to ‘ relational operator

25.!equal = ‘Not Equal to ‘ relational operator

26.fun =To Start user defined function

28.array =To declare an array

30,display()=To print something

31.fun=To Start user defined function

32.; =End of a statement

333.floop =start loop (for loop and while loop)

34.go =Works as ‘switch’ in switch-case

35.if = Works as ‘case’ in switch-case

39.default = Works as ‘default’ in switch-case

40.is = Works as ‘if’ in if-else conditional statement

41.elis = Works as ‘else if’ in if-else conditional statement

42.none = Works as ‘else’ in if-else conditional statement

45.**isprime** = Takes a number to check whether it is prime or not

46.**max** = Takes two numbers and give maximum among them

47.**min** = Takes two numbers and give minimum among them

48.**convert_to_binary** = Takes a number and gives it's equivalent binary

49.**xchange**= Takes two variables and swap their values

50.**#....** = To add single line comment

51.**#...#** =To add multiple line comment

52.**factorial_of** = To Find factorial of a given number

53. All tab , whitespace, new line is ignored.

54. [//=,():;] these returs the exact thing.

55. If anything except above tokens are present in input.txt 'Syntax error, is printed out output.

Discussion and Conclusion:

Above keywords were used to design the compiler and input code was also written using keywords given. Shift-reduce error occurred during compilation. If any grammar matches with the input text then the compiler matches the token with each other in bison file .This compiler is designed taking the reference of C programming language . Some keywords like crack and assign were tried to implement but failed. Except these other keywords were successfully implemented and executed.