**Khulna University of Engineering & Technology**

**Department of Computer Science & Engineering**

**Course No :** CSE 3112

**Course Title :** Compiler Design Laboratory

**Project Name** : Simple Compiler using FLEX & Bison

**Submission By:**

Samin Yasar

Roll : 1707006

Department of Computer Science and Engineering

Khulna University of Engineering & Technology

**Submission Date :** 15-06-2021

**Introduction**

A compiler is a computer program that translates computer code written in one programming language into another language. The name compiler is primarily used for programs that translate source code from a high-level programming language to a lower level language to create an executable program.

**Flex and Bison**

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

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. ... Compile the code output by Bison, as well as any other source files .

**Run the program in terminal**

1. bison -d 1707006.y
2. flex 1707006.l
3. gcc 1707006.tab.c lex.yy.c -o out
4. out

**Procedure**

1. The code is divided into two part flex file (.l) and bison file (.y) .

2. Input expression check the lex (.y) file and if the expression satisfies the rule then it check the CFG into the bison file .

3.it’s a bottom up parser and the parser construct the parse tree .firstly ,matches the leaves node with the rules and if the CFG matches then it gradually goes to the root .

**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.

**My compiler tokens**

IF, ELIF, ELSE, MAIN, INT, FLOAT, CHAR, COM, START, END, SWITCH, CASE, DEFAULT, BREAK, LOOP, PF, SIN, COS, TAN, LOG, LOG10

**CFG**

**Context-free grammars** (CFGs) are used to describe context-free languages. A context-free grammar is a set of recursive rules used to generate patterns of strings. A context-free grammar can describe all regular languages and more, but they cannot describe all possible languages.

**My compiler CFGs**

program: MAIN ':' START cstatement END

;

cstatement: cstatement statement {}

| statement {}

;

statement: ';'

| declaration ';'

| expression ';'

| VAR '=' expression ';'

| COM

| LOOP '(' expression '<' expression ';' expression '+''+' ')' START expression '=' expression ';' END | LOOP '(' expression '>' expression ';' expression '-''-' ')' START expression '=' expression ';' END

| LOOP '(' expression '<' expression ';' expression '-''-' ')' START expression '=' expression ';' END

| LOOP '(' expression '>' expression ';' expression '+''+' ')' START expression '=' expression ';' END {

| SWITCH '(' expression ')' START B END

| IF '(' expression ')' START expression ';' END

| IF '(' expression ')' START expression ';' END ELIF '(' expression ')' START expression ';' END ELSE START expression ';' END

B : C

| C D

;

C : C '+' C

| CASE NUM ':' expression ';' BREAK ';' ;

D : DEFAULT ':' expression ';' BREAK ';';

expression: NUM

| VAR

| expression '+' expression

| expression '-' expression

| expression '\*' expression

| expression '/' expression

| expression '%' expression

| expression '^' expression

| expression '<' expression

| expression '>' expression

| '(' expression ')'

| SIN expression

| COS expression

| TAN expression

| LOG10 expression

| LOG expression

;

declaration : TYPE ID1

;

TYPE : INT

| FLOAT

| CHAR

;

ID1 : ID1 ',' VAR '=' NUM

| ID1 ',' VAR

| VAR '=' NUM

| VAR

**Features of this compiler**

1. Main function

2. Comments

3.Single/Multiple Character Variable declaration

4. IF ELSE Block

5. Variable assignment

6. Loop

7. Print function

9. Switch Case

10. Mathematical Expression

Addition, Subtraction, Multiplication, Division, Power, Log () Operation, Sin () operation, Tan () operation,

Cos () operation.

**Discussion**

This is a bottom up parser and the parser generate a set of tokens.In a program Conditional logic, Loops, Variabledeclaration, Mathematicalfunction, array, header file are used. If any grammar matches with the input text then the compiler shows the token is declared. This compiler is written into C programming language.