**Project Report**

**CSE 3212: Compiler Design Laboratory**

**Topic:** Designing a Simple Compiler using flex and bison.

**Submission Date:** 15/06/2021

**Submitted By:**

**Md. Asif Ud Daula**

Roll: 1707031

Department of Computer Science and Engineering

Khulna University of Engineering & Technology, Khulna

**Introduction:**

In this project I have created my own compiler named Mycom compiler using Flex and Bison. Flex works as a scanner to match pattern and Bison generates rule and takes action according to matched items.

A **compiler** is a special program that processes statements written in a particular programming language and turns them into machine language that a computer's processor uses.

**Lex** is a computer program that generates lexical analyzers. Lex is commonly used with the yacc parser generator.

**Bison**, is a [parser generator](https://en.wikipedia.org/wiki/Parser_generator) that is part of the [GNU Project](https://en.wikipedia.org/wiki/GNU_Project). Bison reads a specification of a [context-free language](https://en.wikipedia.org/wiki/Context-free_language) and generates a parser that reads sequences of [tokens](https://en.wikipedia.org/wiki/Lexical_analysis#Token) and decides whether the sequence conforms to the syntax specified by the grammar.

**Objectives:**

• To know about the compiler

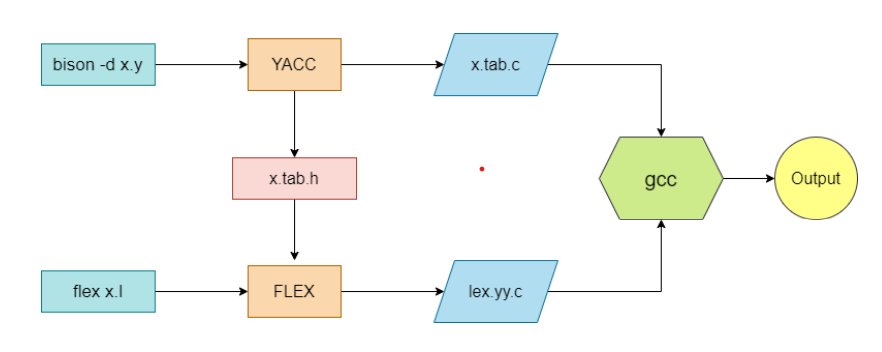
• To implement Context Free Grammar in the

compiler.

• To know the top-down parser and the bottom-up parser.

• To know about Flex and Bison and how to use them to create a compiler.

• To create a new language and it’s semantic and syntactic rules.

**Work Flow: **

**Command:**

**1.**bison -d Mycom.y

**2.**flex Mycom.l

**3.**gcc lex.yy.c Mycom.tab.c -o test

**4.**test

Shows output..

**Manual Table:**

|  |  |  |
| --- | --- | --- |
| **In C language** | **In Mycom input** | **Lfile token** |
| int main | dig prime | IntMain |
| ( ) | [ ] | LP3 RP3 |
| int | dig | Int |
| float | dec | Float |

|  |  |  |
| --- | --- | --- |
| **In C language** | **In Mycom input** | **Lfile token** |
| char | byte | Char |
| ; | / | sm |
| ++ | +\* | aadd |
| -- | -\* | ssub |
| + | # | add |
| - | -- | sub |
| / | | | divi |
| \* | & | mult |
| = | <- | assign |
| <= | =< | Lequal |
| >= | => | Gequal |
| > | >> | Gthan |
| < | << | Lthan |
| if | is | If |
| else | no | Else |
| // | !! | n/a |
| /\* ….. \*/ | !+ ….. +! | n/a |
| while | gofor | Loop |
| printf | out | Print |

**Syntax:**

**Declaration::::**

**Int I;** ---->dig I/

float d;----->dec d/

char c;----> byte c/

**assign::::**

a=12; ----> a<-12/

a=c; ----> a<-c/

**addition:::::**

c=12+23;------>c<-12#23/

c=a+b; -------> c<-a#b/

**subtraction::::**

c=12-23;------>c<-12--23/

c=a-b; -------> c<-a--b/

**other operators:::**

a>23------>a>>23

b<23----->b<<23

a++;------->a+\*/

b--;--------->b-\*/

**Printf:::::**

1. out “this is a print”/

2. out variable/

**if/else:::::**

is var>>12 [

………….statement/

]

No [

…………statement/

]

Loop::::

int i<-0/

gofor i<<10[

out i/

]

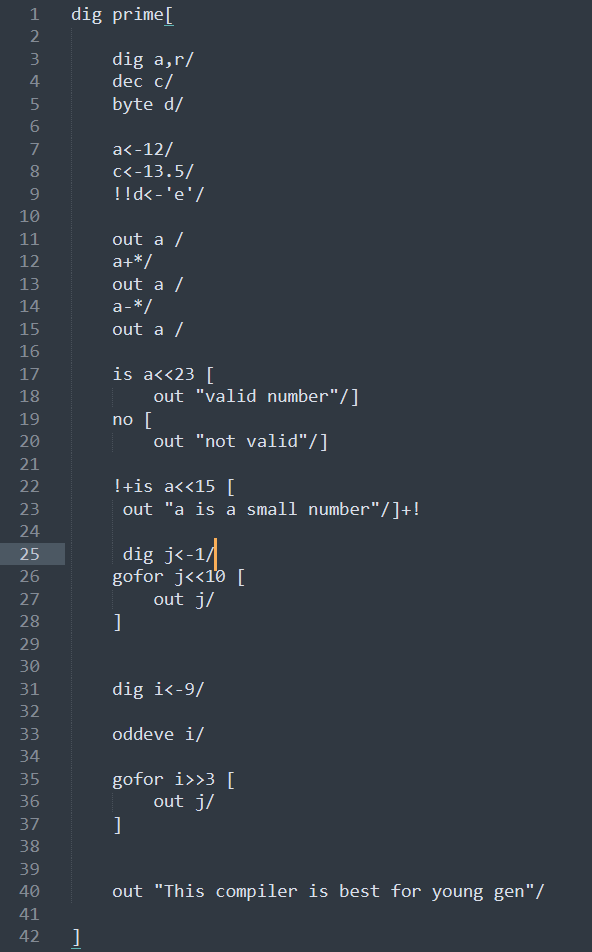
**Comments:::::**

!! -->single line comment.

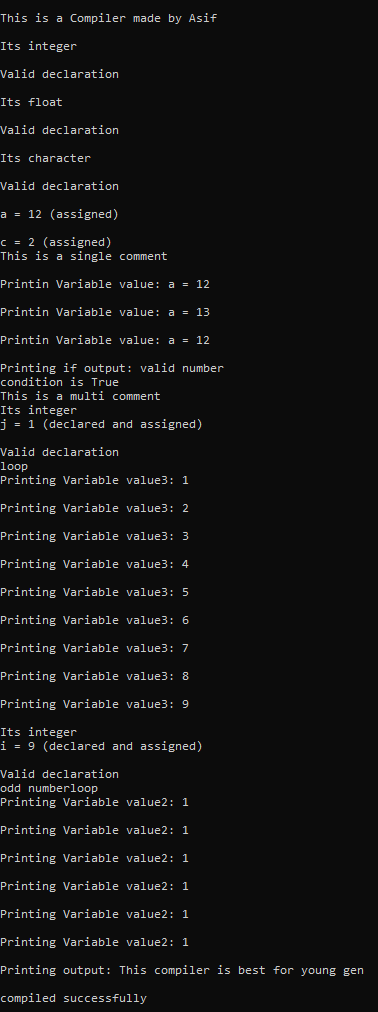
!+ ….. +! ------> multiple line comment.

**Built in Function::::::**

oddeve var/------> prints whether the var(variable) is odd or even;

**Sample input:**

Output:



**References:**

1.https://www.wikipedia.org/

2.Book:Flex and Bison by John Levine