NATIONAL INSTITUTE OF TECHNOLOGY, DELHI

Department of Computer Science and Engineering

CSB353: Compiler Design

Project synopsis

Project name: C ladder

Submitted to:

Dr. Shelly Sachdeva
Department of Computer Science and Engineering

Submitted by:

Gourav Bansal (191220020)
Kishan Srivastava (191220030)
Vardan Agarwal (191220051)

INDEX

C Ladder

INTRODUCTION:

A compiler is a translating program that translates the instructions of high-level language to machine level language. A program which is input to the compiler is called a **Source program**. This program is now converted to a machine level language by a compiler is known as the **Object code.**

Following students of CSE 3rd year are the part of a team working on this project:

1. **Gourav Bansal – 191220020**
2. **Kishan Srivastava – 191220030**
3. **Vardan Agarwal – 191220051**

PROBLEM STATEMENT:

To build a complete compiler that accepts a high-level language as input and produces working assembly code as output.

DOMAIN:

For now, we have thought of building a compiler that can accept a basic C program with limited keywords, conditional statements and loops.

Our compiler will specify productions for the following:

- **Arithmetic operators**
- **Relational operators**
- **Conditional constructs**: if, else if and else.
- **Loop constructs**: for, while.
- **Keywords of C (necessary ones)**
- **Single and multi-line Comments**
- **Data types:** int/float

TEST CASE:

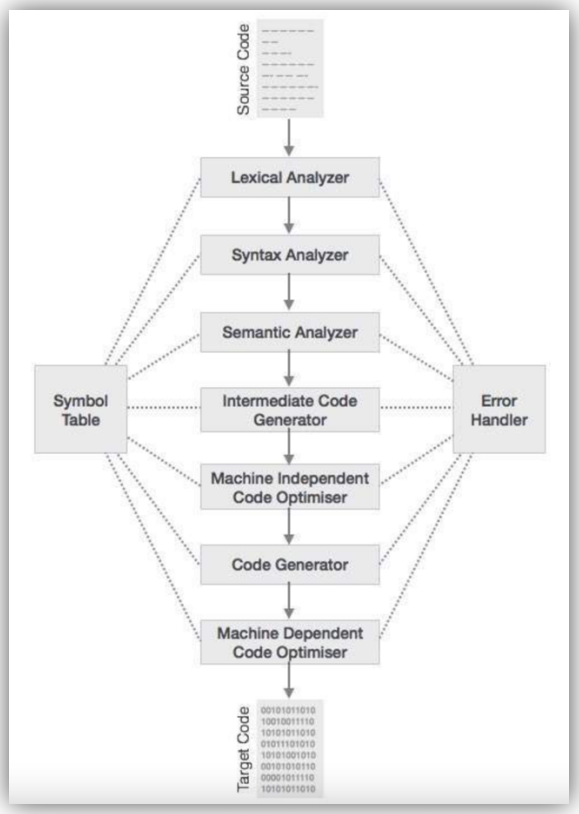
```c
#include<stdio.h>
int main()
{
    int n=5;
    int flag=1;
    if(flag==1)
    {
        // first n positive odd numbers
        int j = 0;
        for(int i=1; j<n; i+=2,j+=1)
            printf("%d",i);

    }
    else
    {
        // first n positive even numbers
        int j = 0;
        for(int i=2; j<n; i+=2,j+=1)
            printf("%d",i);
    }

}
```

DESIGN IDEA:

The compilation process is a sequence of various phases. Each phase takes input from its previous stage, has its own representation of source program, and feeds its output to the next phase of the compiler.



Lexical Analysis

The first phase of scanner works as a text scanner. This phase scans the source code as a stream of characters and converts it into meaningful lexemes.

Implementation planned using a lex program.

Syntax Analysis

The next phase is called the syntax analysis or **parsing**. It takes the token produced by lexical analysis as input and generates a parse tree (or syntax tree).

Implementation of parser planned using lex and yacc programs.

Semantic Analysis

Semantic analysis checks whether the parse tree constructed follows the rules of language.

Implementation of parser planned using lex and yacc programs.

Intermediate Code Generation

After semantic analysis the compiler generates an intermediate code of the source code for the target machine.

Implementation planned using lex and yacc programs.

*Test cases will be .c type files.

Code Optimization

The next phase does code optimization of the intermediate code.

Code Generation

In this phase, the code generator takes the optimized representation of the intermediate code and maps it to the target machine language.

FURTHER ADD-ONs

We will also display the symbol Table, Constant table and intermediately generated code

HARDWARE REQUIREMENTS

Assuming, that we are designing a very basic C compiler, it will have to take very few information as input as far as memory is concerned. Further the whole information has to be processed. So, there is no need for a high specs system for running this.

All this suggest that the minimum hardware requirements should be: -

• Memory (RAM): 512 MB of RAM required.

• Hard Disk Space: 100 MB of free space required.

• Processor: Intel Pentium 4 or later.

• Cache: 512KB

SOFTWARE REQUIRMENTS

Following software should be installed for smooth functioning:

- Operating System: Windows 7/8/8.1/10/11/MacOs/LINUX
- FLEX
- BISON (yacc)
- gcc compiler
- Code editor for editing files