COMPILER LAB

MINI PROJECT

MINI C COMPILER

A compiler is a special program that processes statements written in a particular programming language and turns them into machine language or code that a computer's processors use. A compiler can broadly be divided into two phases based on the way they compile.

1. **Analysis phase :** Known as the front-end of the compiler, the analysis phase of the compiler reads the source program, divides it into core parts and then checks for lexical, grammar and syntax errors.The analysis phase generates an intermediate representation of the source program and symbol table. This phase consists of:

- ➢ Lexical Analysis
- ➢ Syntax Analysis
- ➢ Semantic Analysis
- ➢ Intermediate Code Generation

2. **Synthesis phase** : Known as the back-end of the compiler, the synthesis phase generates the target program with the help of intermediate source code representation and symbol table. This phase consists of:

- ➢ Code Optimization
- ➢ Code Generator

In this mini project we built only the **Analysis phase** which includes **:**

1.**Lexical Analysis :**

Lexical analysis is the first phase of a compiler. It takes the modified source code from language preprocessors that are written in the form of sentences. The lexical analyzer breaks these syntaxes into a series of tokens, by removing any whitespace or comments in the source code.

If the lexical analyzer finds a token invalid, it generates an error. The lexical analyzer works closely with the syntax analyzer. It reads character streams from the source code, checks for legal tokens, and passes the data to the syntax analyzer when it demands.

2.**Syntax Analysis :**

Syntax analysis or parsing is the second phase of a compiler. It takes the token produced by lexical analysis as input and generates a parse tree or syntax tree.

3.**Semantic Analysis :**

Semantic analysis is the third phase of a compiler. Semantic analyzer checks whether the parse tree constructed by the syntax analyzer follows the rules of language.

4.**Intermediate Code Generation :**

After semantic analysis the compiler generates an intermediate code of the source code for the target machine. It represents a program for some abstract machine. It is in between the high-level language and the machine language. This intermediate code should be generated in such a way that it makes it easier to be translated into the target machine code.

1)Senthuran R(2017103590)
2)R Romeyo(2017103617)