**Compiler Design Lab Project**

This project aims at developing a compiler that will accept high level instructionsfrom user based on our syntax & semantics, check it for errors(if any) and will computethe desired output.

**Figure 1 :**

http://www.cs.uaf.edu/~cs631/node4.html

**Figure 2:**

**http://www.math.bas.bg/~nkirov/2004/Horstman/ch01/images/compiler.png**

This project aims at developing a compiler that will accept high level instructionsfrom user based on our syntax & semantics, check it for errors(if any) and will computethe desired output.

**Figure 1 :**

http://www.cs.uaf.edu/~cs631/node4.html

**Figure 2:**

**http://www.math.bas.bg/~nkirov/2004/Horstman/ch01/images/compiler.png**

**(15B17CI671)**

**If - else Compiler in C**

Submitted by:

**Chaitanya Goel (9917103146)**

**Ayush Bansal (9917103145)**

**Batch – F5**

Submitted To:

**Dr. Mukesh Saraswat**

****

**Department of CSE/IT**

**Jaypee Institute of Information Technology University, Noida**

# FEB 2020

**PROBLEM STATEMENT**

Compiler is a software which converts a program written in high level language (Source Language) to low level language (Object/Target/Machine Language).So similarly are trying to design a simulation of front end phase of C Compiler involving if-else construct using lex and yacc tools in implemented from scratch.

**MOTIVATION**

This Project will be done as a part of Course for Compiler Design Lab. The main aim of this project is to understand the functioning of Compiler and its various phases including Lexical Analysis , Syntax Analyzer, Semantic Analyzer, Intermediate code generation and Compilation of code.

**OBJECTIVES**

This project aims at developing a compiler that will accept high level instructions from user based on our syntax & semantics, check it for errors(if any) and will compute the desired output.

**METHODOLOGY**

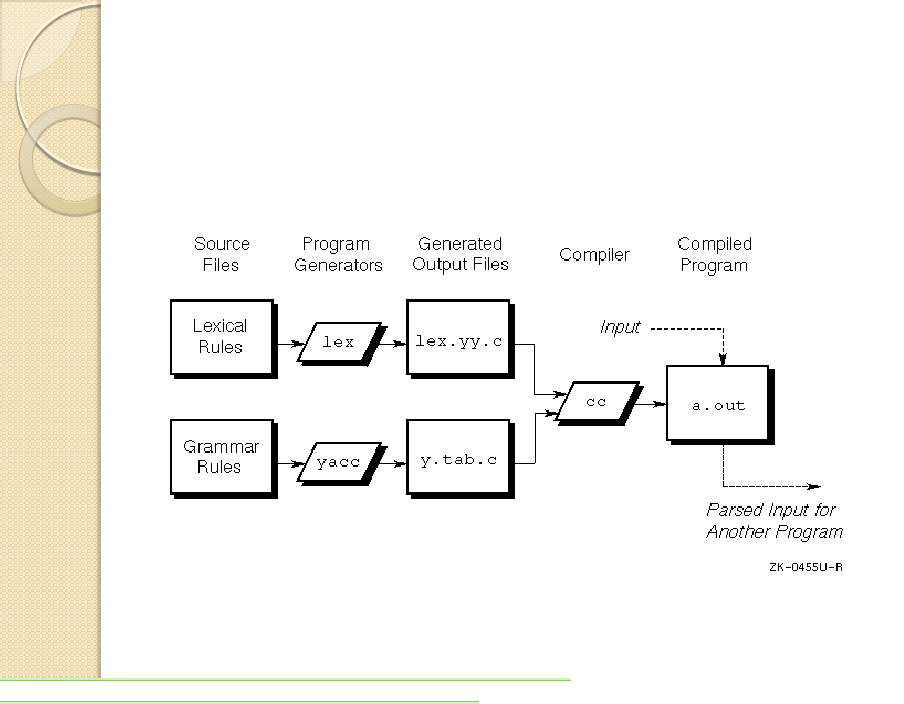
1. Lexical Analysis: Generation of tokens in lexer.l using regular expressions.

2. Syntax Analysis: Created grammar for entire C code that has IF-ELSE construct. Nested IFs are also taken into account. Parsing generates "Success" or "error" with line number.

3. Semantic Analysis: Annotate the grammar with actions to create symbol table, create Abstract Syntax Tree nodes, check for type, check for scope and return detailed errors if any of these fail. The symbol table contains the token name, token data type, token type, line number where it is defined.

4. Intermediate Code Generation: Generated intermediate code on the fly.

**DIAGRAM**



**Minimum Hardware Requirements**

• 32 bit machine,800 MHz processor 1.66GHz

• Windows/Linux OS

• 128 MB RAM

**Software Requirements**

• A lex tool

• A yacc parser tool

• Turbo C/GCC