**Experiment no. 05**

**Aim** - study of phases of compiler.

compiler is a program that converts a source program written in high level programming language into target program which is machine understandable language. The most common reason for converting source code is to create an executable program. Generally, the target program is an executable program that can be used by the user to process the input and produce the related output. The compiler works with two prime features of the language syntax and semantics. If the compiled program can run on a computer whose CPU or operating system is different from the one on which compiler runs, the compiler is known as a cross-compiler. More generally, compilers are the specific type of translator. A program that translates from a low level language to a higher level language is a de-compiler.

Phases of compiler

Each phase transforms the source program from one representation into another representation. They communicate with error handlers and symbol table. But in this customized compiler, we are going through only necessary three phases of the compiler i.e. Lexical Phase, Syntax Phase and Target Code Generation.



**Phase I**

(Lexical Analyzer) - This phase reads the source code as a stream of characters and converts it into meaningful lexemes. A token describes a pattern of characters having same meaning in the source program (such as identifiers, operators, keywords, and numbers).

**Phase II**

(Syntax Analyzer) - This phase generates the syntax tree according to the already known CFG. A syntax analyzer is also called a parser. A parse tree describes the syntactic structure.

**Phase III**

(Semantic Analyzer) - This phase checks the source program for semantic errors and collects the type of information for the code generation. The main functionality is type checking.

**Phase IV**

(ICG) - ICG stands for Intermediate Code Generator. After semantic analysis intermediate code is generated, which is in between high level language and machine language. These codes are generally machine architecture independent, but the level of intermediates code is close to the level of machine codes.

**Phase V**

(Code Optimizer) - This phase removes unnecessary code lines, and arranges the sequence of statements in order to speed up the program execution without wasting resources (CPU, Memory).

**Phase VI**

(Code Generation) - In this phase the code generator takes the optimized representation of intermediate code and converts it into machine understandable code (Target Code).