|  |
| --- |
| **Name:**Liya Elizabeth Joe  **Roll No:**48 |
| **Exp. No:**1 **Date:** 4/08/2020 |
| **Study of Phases of Compilation** |
| **Aim:** To familiarize with different phases of compilation. |
| Compiler is a software which converts a program written in high level language (Source Language) to low level language (Object/Target/Machine Language).The compilation process is a sequence of various phases. Each phase takes input from its previous stage, has its own representation of the source program, and feeds its output to the next phase of the compiler.  **Phases of a Compiler :**  There are two major phases of compilation -   1. Analysis Phase 2. Synthesis Phase   **Analysis Phase :**  In this phase an intermediate representation is created from the given source code . This phase is also 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, which should be fed to the Synthesis phase as input. Front-end constitutes :   1. Lexical Analyzer 2. Syntax Analyzer 3. Semantic Analyzer 4. Intermediate Code Generation     1. Lexical Analyzer  Lexical analyzer phase is the first phase of the compilation process. It takes source code as input. It reads the source program one character at a time and converts it into meaningful lexemes. A Lexical analyzer represents these lexemes in the form of tokens.  2.Syntax Analyzer  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). In this phase, token arrangements are checked against the source code grammar, i.e. the parser checks if the expression made by the tokens is syntactically correct.  3.Semantic Analyzer  Semantic analysis checks whether the parse tree constructed follows the rules of language. For example, assignment of values is between compatible data types, and adding string to an integer. Also, the semantic analyzer keeps track of identifiers, their types and expressions, whether identifiers are declared before use or not etc. The semantic analyzer produces an annotated syntax tree as an output.  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.  **Synthesis Phase :** This phase is 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. Back-end constitutes.   1. Code Optimization 2. Code Generation 3. Code Optimization   The next phase does code optimization of the intermediate code. Optimization can be assumed as something that removes unnecessary code lines, and arranges the sequence of statements in order to speed up the program execution without wasting resources (CPU, memory)   1. Code Generation  In this phase, the code generator takes the optimized representation of the intermediate code and maps it to the target machine language. The code generator translates the intermediate code into a sequence of (generally) relocatable machine code. Sequence of instructions of machine code performs the task as the intermediate code would do. |
| **Result:** Familiarized with various phases of compiler. |
| **Remarks:**(To be filled by faculty)  **Diagram** |