A compiler is a program that reads a program written in one language and translates it into another language. Make executable code for machine.

Interpreter directly execute the operations in source program on inputs supplied by the user.

Preprocessor performs operations on the source files prior to compilation.

Assembler translates assembly language to machine code. Source code is written in assembly language

The machine-language target program produced by a compiler is usually much faster than an interpreter at mapping inputs to output.

An interpreter, however, can usually give better error diagnostics than a compiler, because it executes the source program statement by statement.

Analysis Phase: front end Breaks up source program into constituent pieces and produces an internal representations of it called intermediate code.

a) Lexical b)Syntax c)Semantic d)Intermediate code generator

The analysis part is often called the front end of the compiler; the synthesis part is the back end.

Synthesis Phase: back end Translates the intermediate code into the target program.

a) Code optimizer b) Code generator

Lexical Analyzer: Scanning, The symbol-table entry for an identifier holds information about the identifier, such as its name and type

Lexical Error: A lexical error is a mistake in a lexeme, for examples, typing.

Syntax Analyzer: Parsing,The second phase of the compiler is syntax analysis or parsing.

Interior node represents an operation

Children of the node represent the arguments of the operation.

Syntax Error: A grammatical error is a one that violates the (grammatical) rules of the language, for example if x = 7 y:= 4 (missing then)

Semantic analyzer: type checking , The semantic analyzer uses the syntax tree and the information in the symbol table to check the source program for semantic consistency with the language definition

Semantic Errors: Datatype mismatch, Undeclared variable, Multiple variable declare, Actual and formal mismatch.

Intermediate Code generator: After syntax and semantic analysis of the source program, many compilers generate an explicit low-level or machine-like intermediate (three-address) representation, which we can think of as a program for an abstract machine.

Syntax trees are a form of intermediate representation; they are commonly used during syntax and semantic analysis

*It should be easy to produce and

*It should be easy to translate into the target machine.

Intermediate form called three-address code, which consists of a sequence of assembly-like instructions with three operands per instruction. Each operand can act like a register

Each three-address assignment instruction has at most one operator on the right side

The compiler must generate a temporary name to hold the value computed by a three-address instruction

Three-address instructions like the above, have fewer than three operands.

Code-Optimization: The machine-independent code-optimization phase attempts to improve the intermediate code so that better target code will result.

A simple intermediate code generation algorithm followed by code optimization is a reasonable way to generate good target code

Code Generator: The code generator takes as input an intermediate representation of the source program and maps it into the target language.

If the target language is machine code, registers or memory locations are selected for each of the variables used by the program. Then, the intermediate instructions are translated into sequences of machine instructions that perform the same task. A crucial aspect of code generation is the judicious assignment of registers to hold variables.

The organization of storage at run-time depends on the language being compiled. Storage-allocation decisions are made either during intermediate code generation or during code generation.

The symbol table is a data structure containing a record for each variable name, with fields for the attributes of the name.

These attributes may provide information about the storage allocated for

A name, its type, its scope (where in the program its value may be used), , how much memory it occupies,

And in the case of procedure names

such things as the number and types of its arguments, the method of passing each argument (for example, by value or by reference), and the type returned.

The data structure should be designed to allow the compiler to and the record for each name quickly and to store or retrieve data from that record quickly.

Information is usually put into the symbol table throughout the analysis phase and used for synthesis phase

Linker: A linker, also called link editor or binder, is a program that combines the object modules to form an executable program. In general, in case of a large program, programmers prefer to break the code in to smaller modules, as this simplifies the programming task. Eventually, when the source code of all the modules has been converted in to object code, all the modules need to be put together, which is done by the linker\

A loader is a special type of a program that copies programs from a storage device to the main memory, where they can be executed.

Back end & Front End :

Retargeting: Build a compiler for a new machine by attaching a new code generator to an existing front-end

Optimization: Reuse intermediate code optimizers in compilers for different languages and different machines.