# COMPILER DESIGN - OVERVIEW

Computers are a balanced mix of software and hardware. Hardware is just a piece of mechanical device and its functions are being controlled by a compatible software. Hardware understands instructions in the form of electronic charge, which is the counterpart of binary language in software programming. Binary language has only two alphabets, 0 and 1. To instruct, the hardware codes must be written in binary format, which is simply a series of 1s and 0s. It would be a difficult and cumbersome task for computer programmers to write such codes, which is why we have compilers to write such codes.

Language Processing System

We have learnt that any computer system is made of hardware and software. The hardware understands a language, which humans cannot understand. So we write programs in high-level language, which is easier for us to understand and remember. These programs are then fed into a series of tools and OS components to get the desired code that can be used by the machine. This is known as Language Processing System.
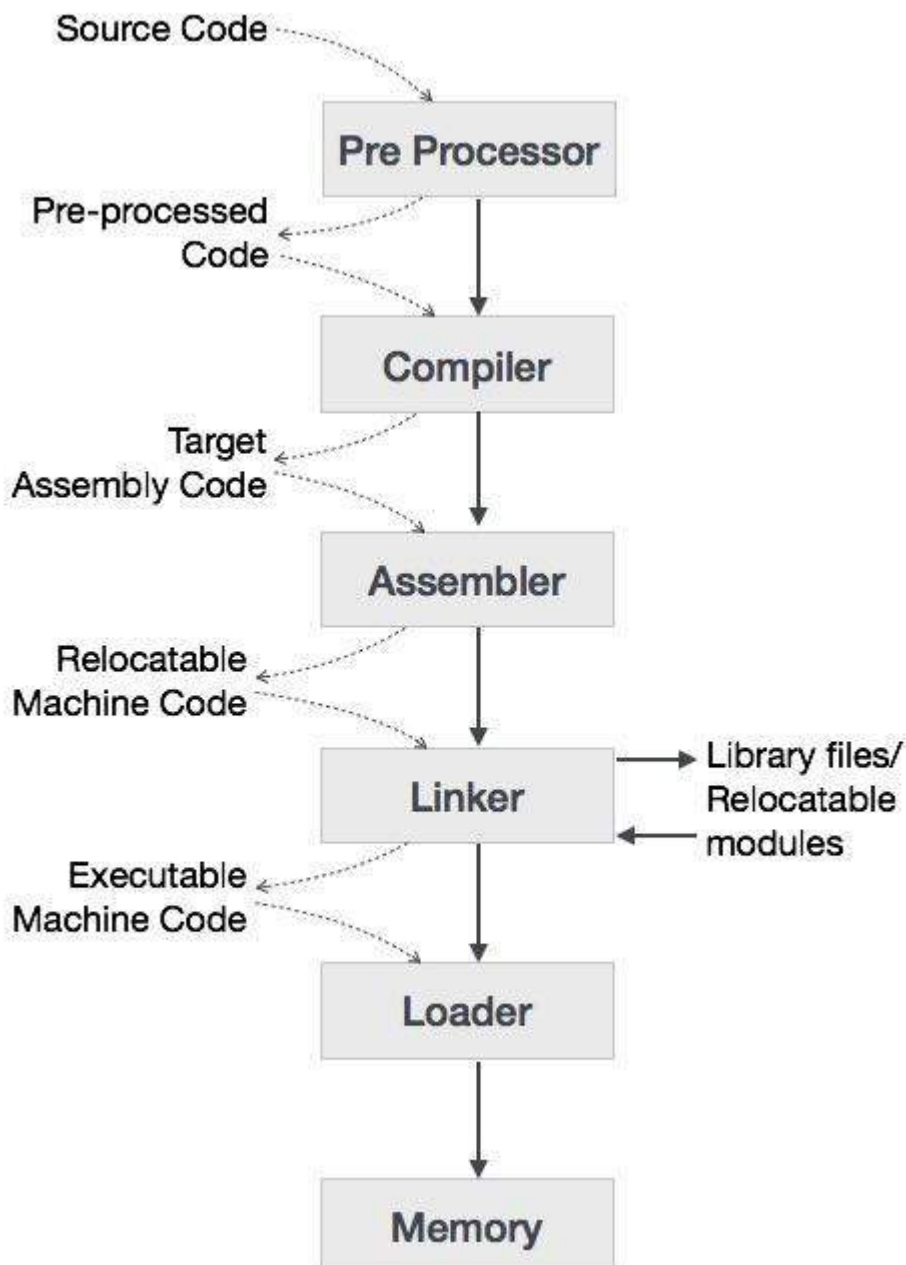
The high-level language is converted into binary language in various phases. A **compiler** is a program that converts high-level language to assembly language. Similarly, an **assembler** is a

program that converts the assembly language to machine-level language.

Let us first understand how a program, using C compiler, is executed on a host machine.

- User writes a program in C language *high − levellanguage*.
- The C compiler, compiles the program and translates it to assembly program *low − levellanguage*.
- An assembler then translates the assembly program into machine code *object*.
- A linker tool is used to link all the parts of the program together for execution *executablemachinecode*.
- A loader loads all of them into memory and then the program is executed.

Before diving straight into the concepts of compilers, we should understand a few other tools that work closely with compilers.

Preprocessor

A preprocessor, generally considered as a part of compiler, is a tool that produces input for compilers. It deals with macro-processing, augmentation, file inclusion, language extension, etc.

Interpreter

An interpreter, like a compiler, translates high-level language into low-level machine language. The difference lies in the way they read the source code or input. A compiler reads the whole source code at once, creates tokens, checks semantics, generates intermediate code, executes the whole program and may involve many passes. In contrast, an interpreter reads a statement from the input, converts it to an intermediate code, executes it, then takes the next statement in sequence. If an error occurs, an interpreter stops execution and reports it. whereas a compiler reads the whole program even if it encounters several errors.

Assembler

An assembler translates assembly language programs into machine code.The output of an assembler is called an object file, which contains a combination of machine instructions as well as the data required to place these instructions in memory.

Linker

Linker is a computer program that links and merges various object files together in order to make an executable file. All these files might have been compiled by separate assemblers. The major task of a linker is to search and locate referenced module/routines in a program and to determine the memory location where these codes will be loaded, making the program instruction to have absolute references.

Loader

Loader is a part of operating system and is responsible for loading executable files into memory and execute them. It calculates the size of a program *instructionsanddata* and creates memory space for it. It initializes various registers to initiate execution.

Cross-compiler

A compiler that runs on platform *A* and is capable of generating executable code for platform *B* is called a cross-compiler.

Source-to-source Compiler

A compiler that takes the source code of one programming language and translates it into the source code of another programming language is called a source-to-source compiler.