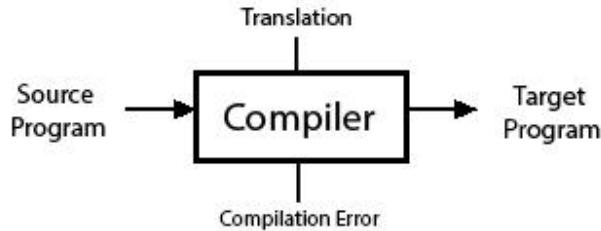


What Is a Compiler?

A **compiler** is a system software that translates source code written in a high-level programming language (like C, C++, Java) into machine code, assembly code, or an intermediate code that a computer can execute.



🔍 Key Objectives:

- ❖ **Translation:** Converts human-readable code into machine-readable instructions.
- ❖ **Error Detection:** Identifies syntax and semantic errors before execution.
- ❖ **Optimization:** Enhances performance and efficiency of the final executable.
- ❖ **Portability:** Allows the same source code to run on different platforms by targeting different machine architectures.

Why Do We Need a Compiler?

- ❖ Computers only understand **binary machine code**.
- ❖ High-level languages are designed for human readability and productivity.
- ❖ A compiler bridges this gap by converting high-level code into low-level instructions.

Types of Compilers

The four main types of compilers are as follows –

- ❖ **Single-Pass Compiler** – A single-pass compiler processes the source code in a single pass, from start to finish, generating machine code as it goes. It is efficient but may not catch all errors or perform extensive optimization.

- ❖ **Multi-Pass Compiler** – A multi-pass compiler makes multiple passes over the source code, analyzing it in different stages. This allows for more thorough error checking and optimization but can be slower than a single-pass compiler.
- ❖ **Just-In-Time (JIT) Compiler** – A JIT compiler translates code into machine language while the program is running, on-the-fly. It is used in languages like Java and JavaScript to improve performance by converting code as needed during execution.
- ❖ **Ahead-of-Time (AOT) Compiler** – An AOT compiler translates code into machine language before the program is run, producing an executable file. This approach is common in languages like C and C++, providing fast execution but requiring compilation before running the program.