## 1.3 CROSS-COMPILATION:

### What is Cross Compiler?

Compilers are the tool used to translate high-level programming language to low-level programming language. The simple compiler works in one system only, but what will happen if we need a compiler that can compile code from another platform, to perform such compilation, the cross compiler is introduced. In this article, we are going to discuss cross-compiler.

A cross compiler is a compiler capable of creating executable code for a platform other than the one on which the compiler is running. For example, a cross compiler executes on machine X and produces machine code for machine Y.

### Where is the cross compiler used?

In bootstrapping, a cross-compiler is used for transitioning to a new platform. When developing software for a new platform, a cross-compiler is used to compile necessary tools such as the operating system and a native compiler.

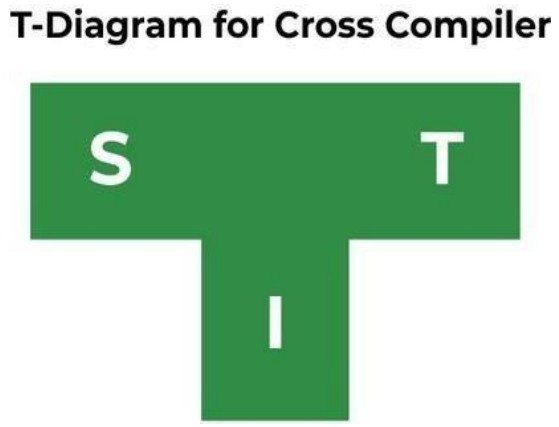
For microcontrollers, we use cross compiler because it doesn't support an operating system.

It is useful for embedded computers which are with limited computing resources.

To compile for a platform where it is not practical to do the compiling, a cross-compiler is used.

When direct compilation on the target platform is not infeasible, so we can use the cross compiler.

It helps to keep the target environment separate from the built environment.



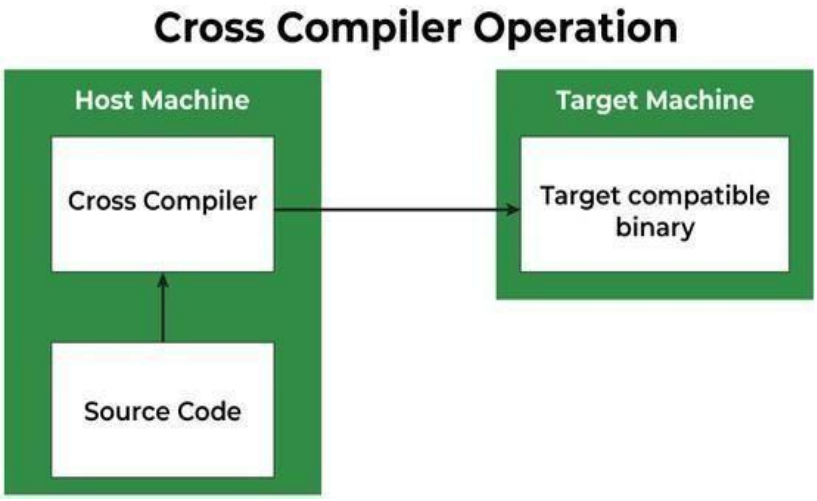

T-Diagram for Cross-compiler

### A compiler is characterized by three languages:

1. The source language that is compiled.
2. The target language T is generated.
3. The implementation language that is used for writing the compiler.

**How does cross-compilation work?**

A cross compiler is a compiler capable of creating executable code for a platform other than the one on which the compiler is running. In paravirtualization, one computer runs multiple operating systems and a cross compiler could generate an executable for each of them from one main source.




Working of Cross-Compiler

**What are the phases of the compiler?**

1. Lexical Analysis.
2. Syntactic Analysis (i.e. Parsing)
3. Intermediate Code Generation (and semantic analysis)
4. Optimization (optional)
5. Code Generation.

**How compiler is different from a cross-compiler?**

The native compiler is a compiler that generates code for the same platform on which it runs and on the other hand, a Cross compiler is a compiler that generates executable code for a platform other than one on which the compiler is running.

**Difference Between Native Compiler and Cross Compiler**

| Native Compiler | Cross Compiler |
|---|---|
| Translates program for same hardware/platform/machine on it is running. | Translates program for different hardware/platform/machine other than the platform which it is running. |
| It is used to build programs for same system/machine & OS it is installed. | It is used to build programs for other system/machine like AVR/ARM. |
| It is dependent on System/machine and OS. | It is independent of System/machine and OS. |
| It can generate executable file like .exe | It can generate raw code .hex |
| Generates machine code for the same platform it's running on. | Generates machine code for a different platform than it's running on. |
| Turbo C or GCC is native Compiler. | Keil is a cross compiler. |
| Used for development and testing on the same system. | Used for cross-platform development, porting, and cross-compiling. |
| Example :- GCC compiler on a Linux machine. | Example:- ARM compiler on a Windows machine for compiling code for a Raspberry Pi. |