5. How .Net runtime generates executable files from your code?

Ans-Code Execution Process

The Code Execution Process involves the following two stages:

1.Compiler time process.

2.Runtime process.

1.Compiler time process.-1. Compiler time process

The .Net framework has one or more language compliers, such as Visual Basic, C#, Visual C++, JScript, or one of many third-party compilers such as an Eiffel, Perl, or COBOL compiler.

Any one of the compilers translate your source code into Microsoft Intermediate Language (MSIL) code.

For example, if you are using the C# programming language to develop an application, when you compile the application, the C# language compiler will convert your source code into Microsoft Intermediate Language (MSIL) code.

In short, VB.NET, C# and other language compilers generate MSIL code. (In other words, compiling translates your source code into MSIL and generates the required metadata.)

Currently "Microsoft Intermediate Language" (MSIL) code is also known as "Intermediate Language" (IL) Code or "Common Intermediate Language" (CIL) Code.

SOURCE CODE -----.NET COMLIPER------> BYTE CODE (MSIL + META DATA)

2. Runtime process.

The Common Language Runtime (CLR) includes a JIT compiler for converting MSIL to native code.

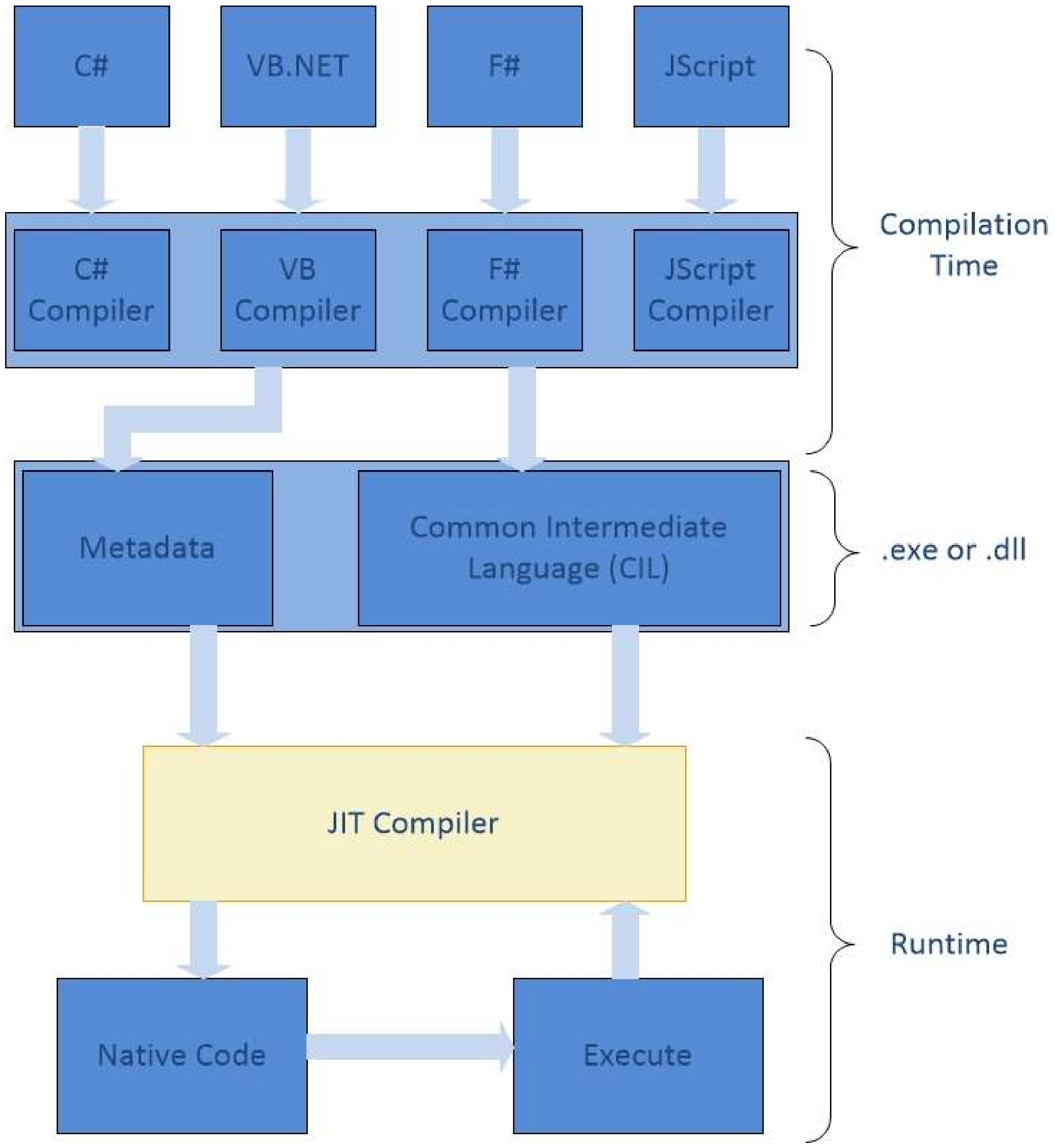
The JIT Compiler in CLR converts the MSIL code into native machine code that is then executed by the OS.

During the runtime of a program the "Just in Time" (JIT) compiler of the Common Language Runtime (CLR) uses the Metadata and converts Microsoft Intermediate Language (MSIL) into native code.

BYTE CODE (MSIL + META DATA) ----- Just-In-Time (JIT) compiler------> NATIVE CODE

Just-In-Time Compilation

The JIT compiler is part of the Common Language Runtime (CLR). The CLR manages the execution of all .NET applications. In addition to JIT compilation at runtime, the CLR is also responsible for garbage collection, type safety and for exception handling.

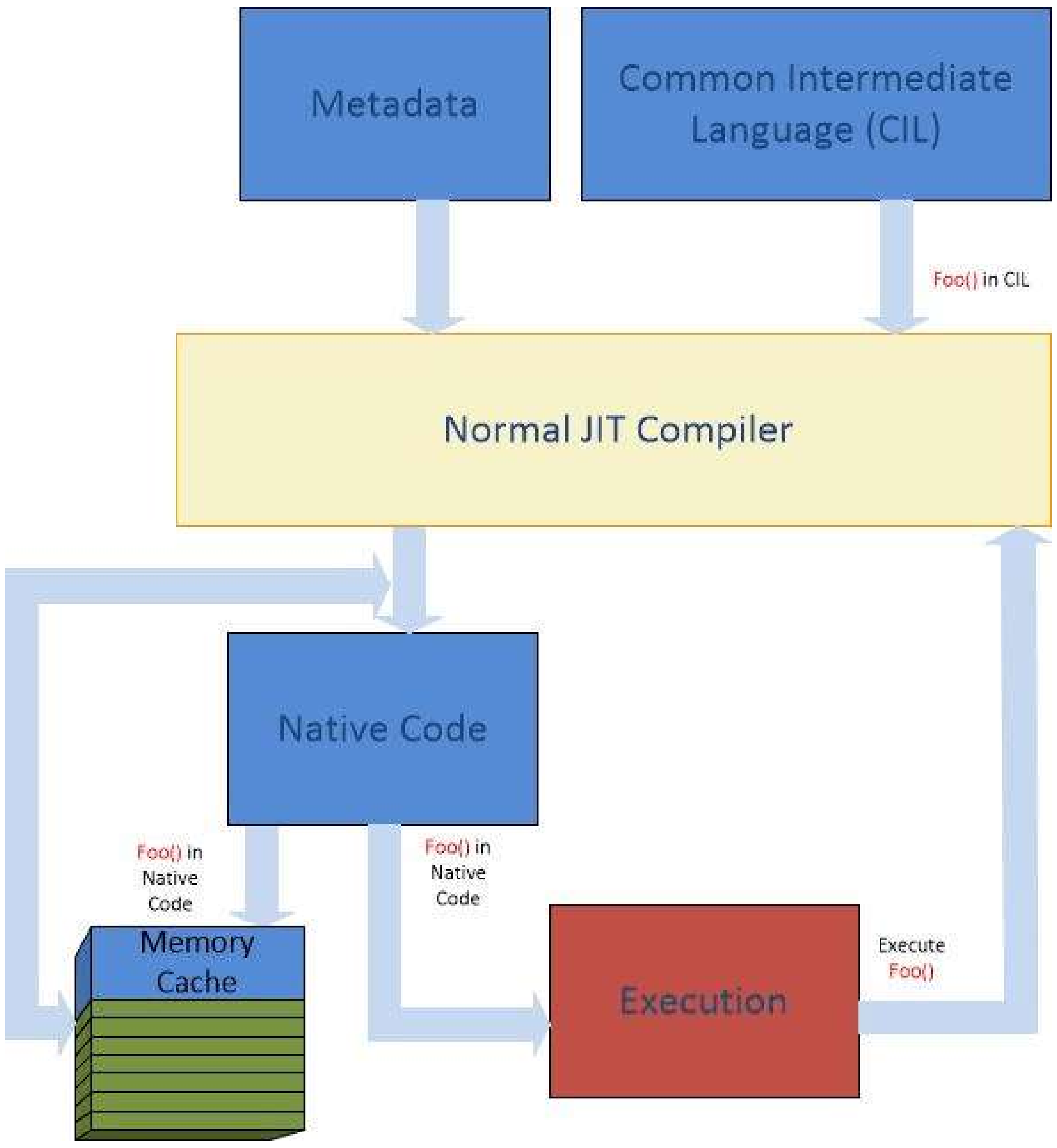


Different machine configurations use different machine level instructions. As Figure 1 shows, the source code is compiled to exe or dll by the .NET compiler. Common Intermediate Language (CIL) consists of instructions that any environment supporting .NET can execute and includes metadata describing structures of both data and code. The JIT Compiler processes the CIL instructions into machine code specific for an environment. Program portability is ensured by utilizing CIL instructions in the source code. The JIT compiler compiles only those methods called at runtime. It also keeps track of any variable or parameter passed through methods and enforces type-safety in the runtime environment of the .NET Framework.

There are three types of JIT compilation in the .NET framework:-

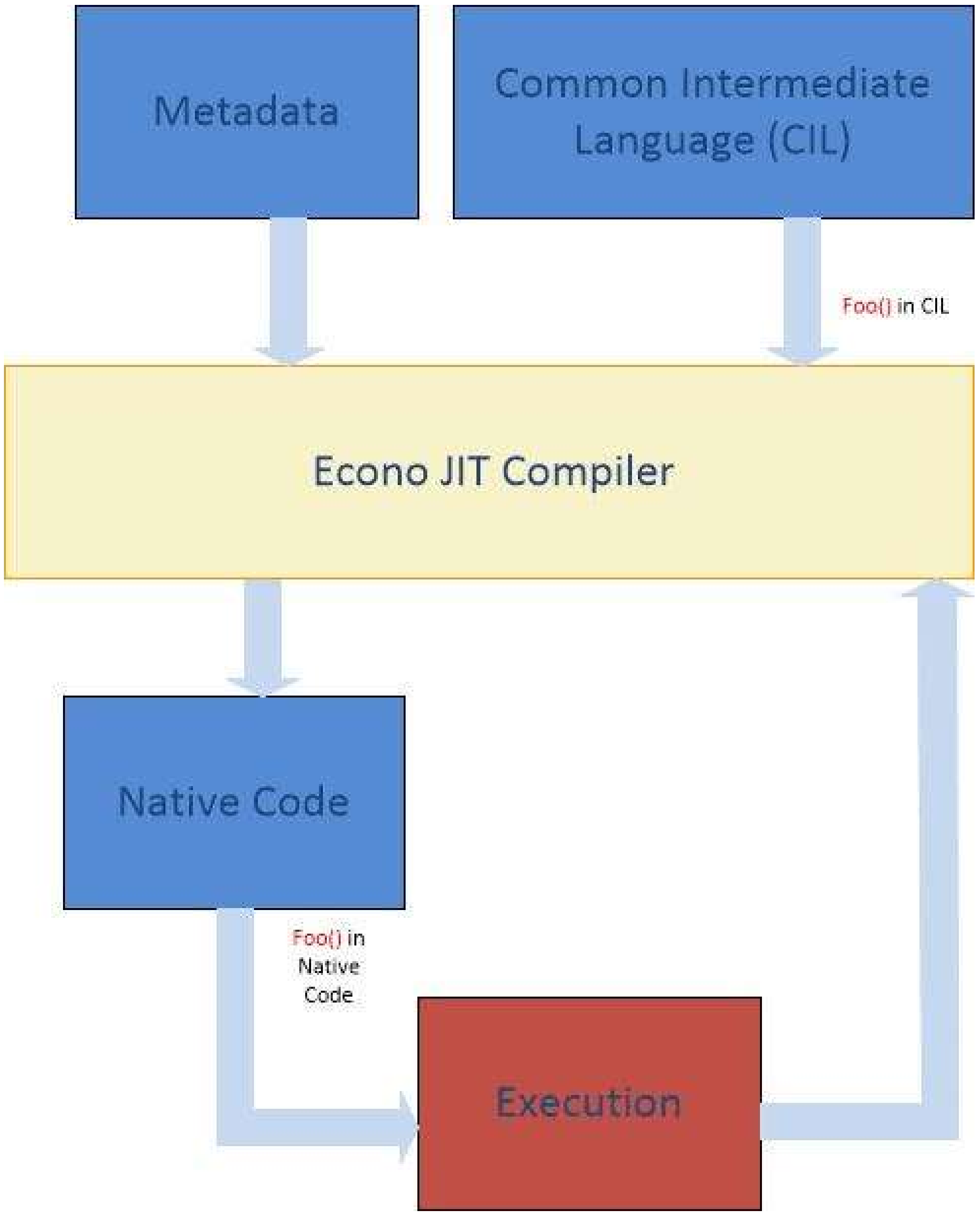
Normal JIT Compilation

With the Normal JIT Compiler (figure 2) methods are compiled when called at runtime. After execution this method is stored in the memory and it is commonly referred as "jitted". No further compilation is required for the same method. Subsequent method calls are accessible directly from the memory cache.



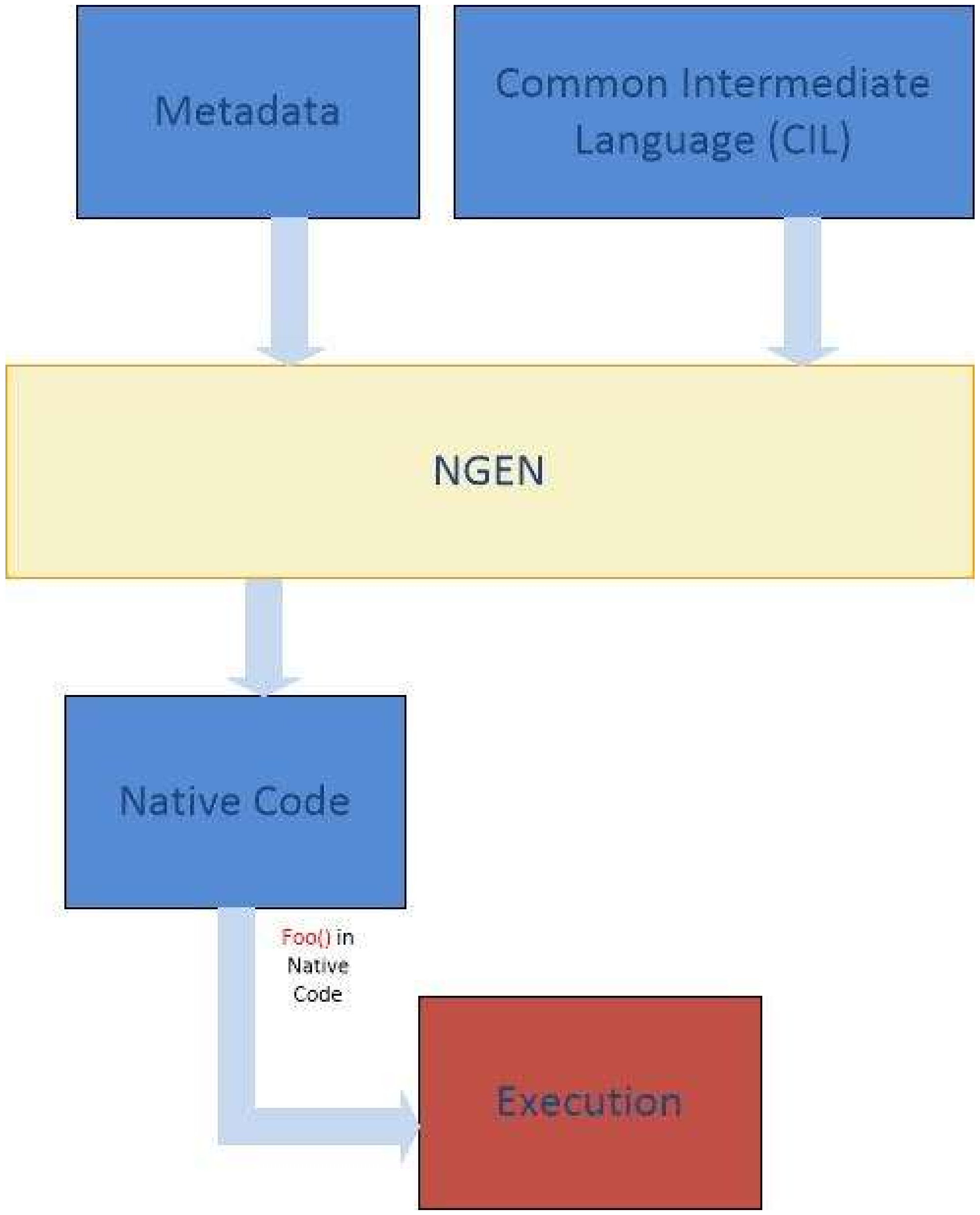
Econo JIT Compilation

The Econo JIT Compiler is displayed in figure 3. It compiles methods when called at runtime and removes them from memory after execution.



Pre-JIT Compilation

Another form of compilation in .NET is called Pre-JIT compilation. It compiles the entire assembly instead of used methods. In .NET languages, this is implemented in Ngen.exe (Native Image Generator). All CIL instructions are compiled to native code before startup, as shown in figure 4. This way the runtime can use native images from the cache instead of invoking the JIT Compiler.


Metadata
Common Intermediate Language (CIL)
NGEN
Native Code
Foo() in Native Code
Execution