CLI

(Common Language Infrastructure)

.NET

*The **Common Language Infrastructure** (**CLI**) is an open specification developed by Microsoft and standardized by ISO and ECMA that describes executable code and a runtime environment that allows multiple high-level languages to be used on different computer platforms without being rewritten for specific architectures.*

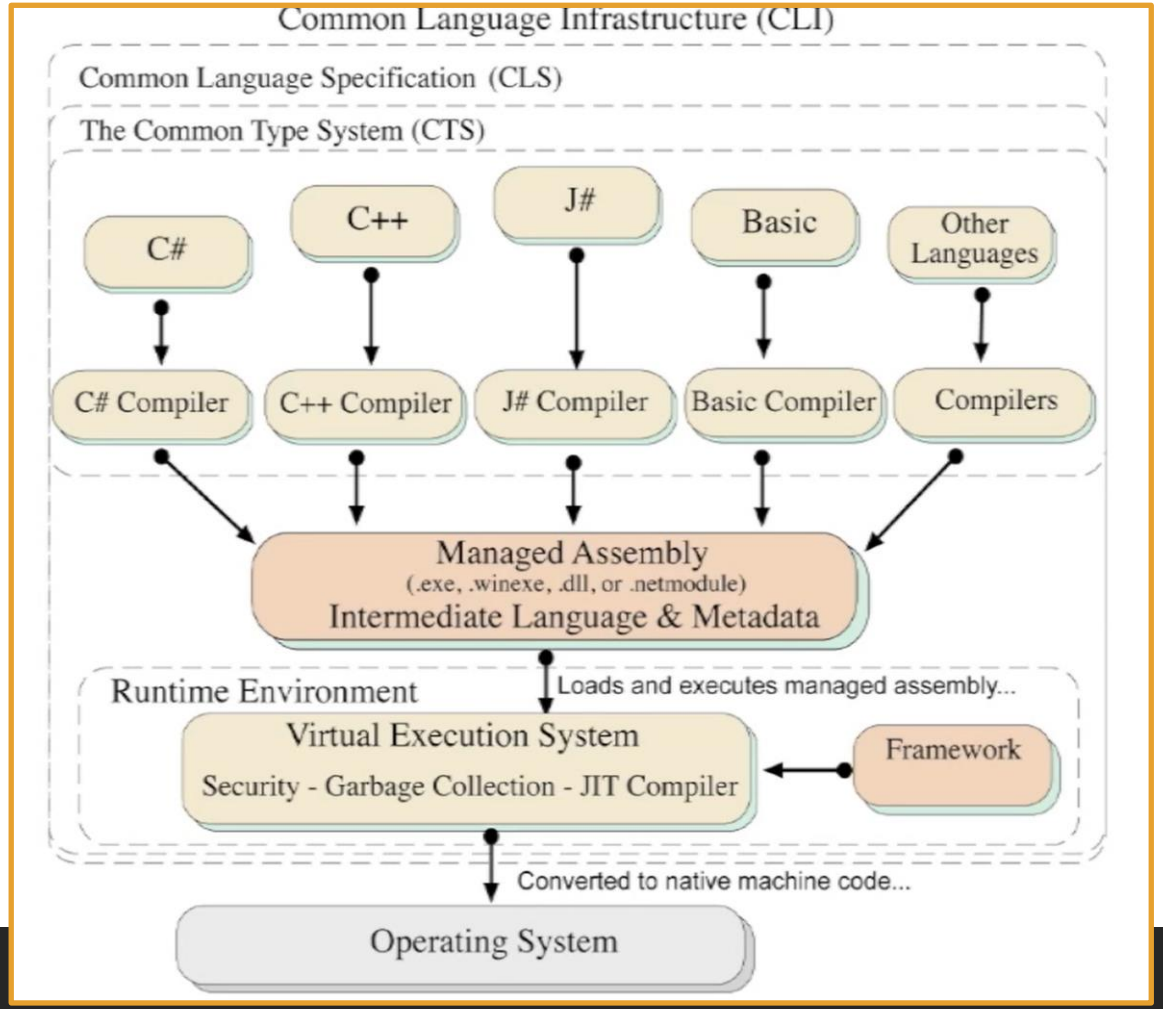
HTTPS://EN.WIKIPEDIA.ORG/WIKI/COMMON_LANGUAGE_INFRASTRUCTURE

CLI (Common Language Infrastructure)

https://searchapparchitecture.techtarget.com/definition/Common-Language-Infrastructure-CLI

Part of Microsoft's .NET strategy, ***Common Language Infrastructure (CLI)*** enables an application program written in any programming language to be run on any operating system (OS) using a "common" runtime program rather than a language-specific one.

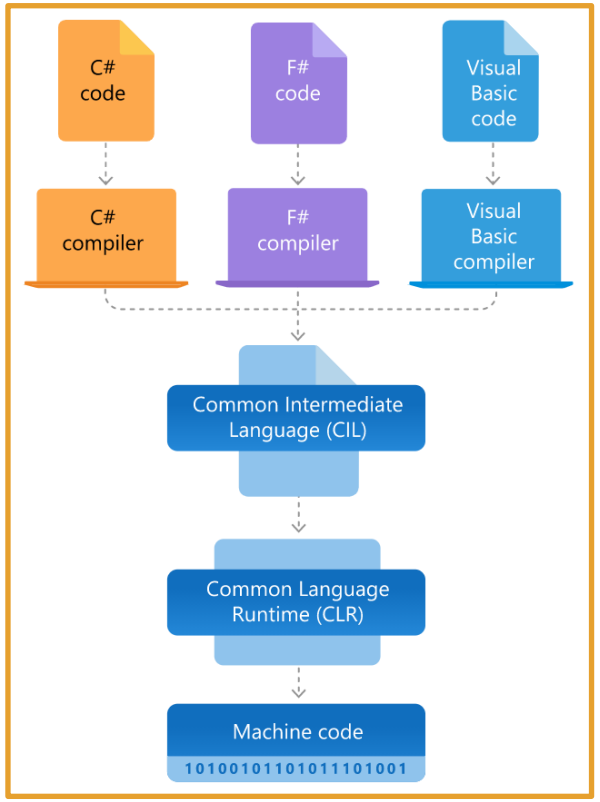
Common Language Infrastructure provides a virtual execution environment comparable to the one provided by Sun Microsystems for Java programs.

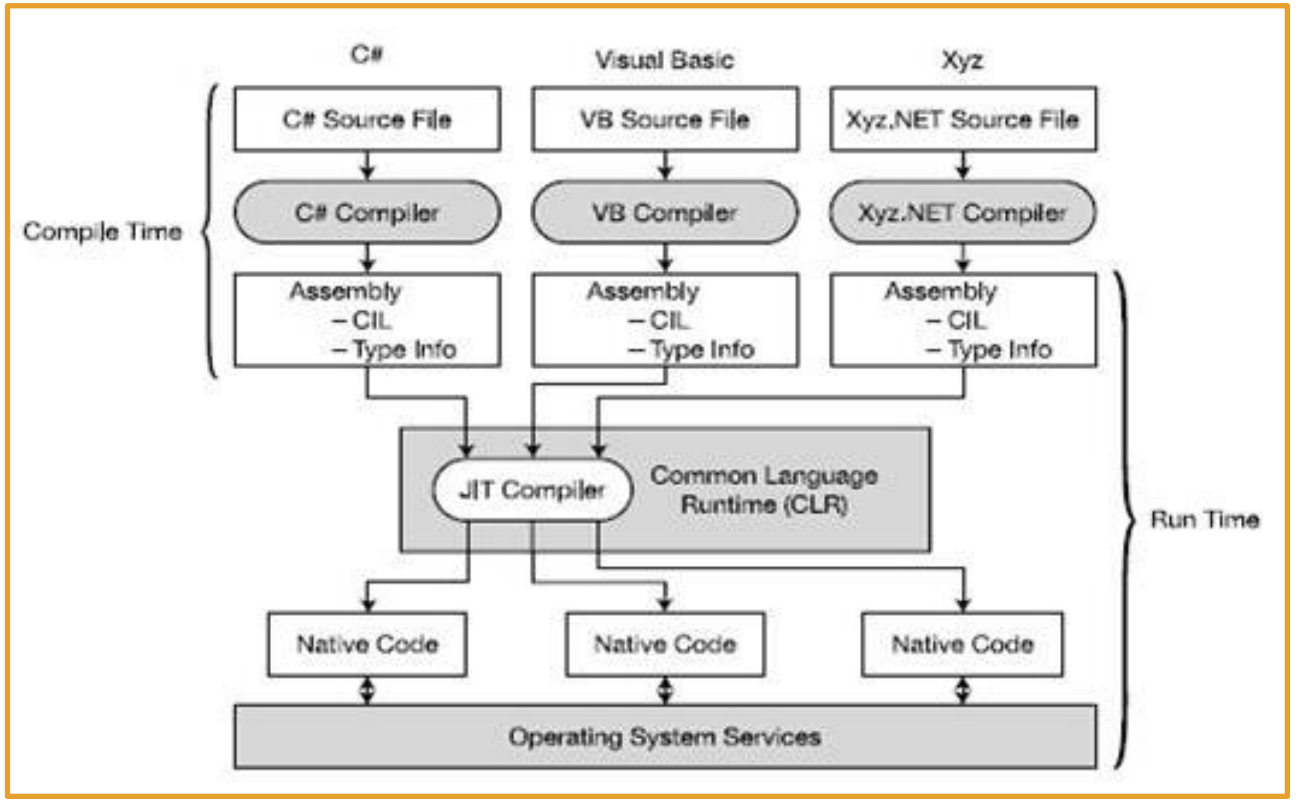


CLI (Common Language Infrastructure)

https://docs.microsoft.com/en-us/dotnet/standard/managed-execution-process
https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet-framework

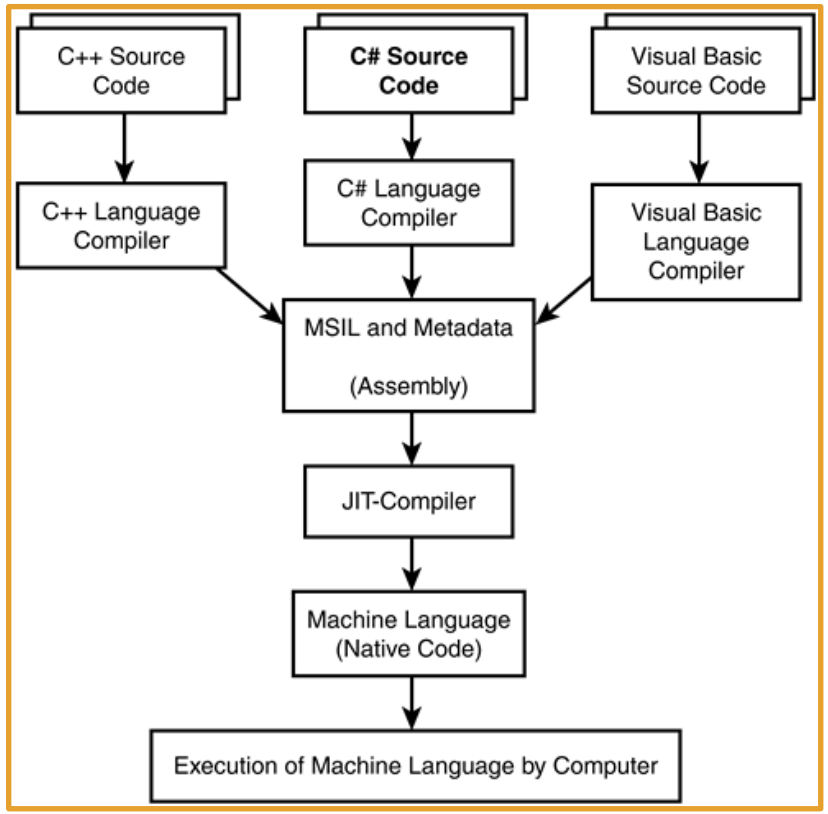
---

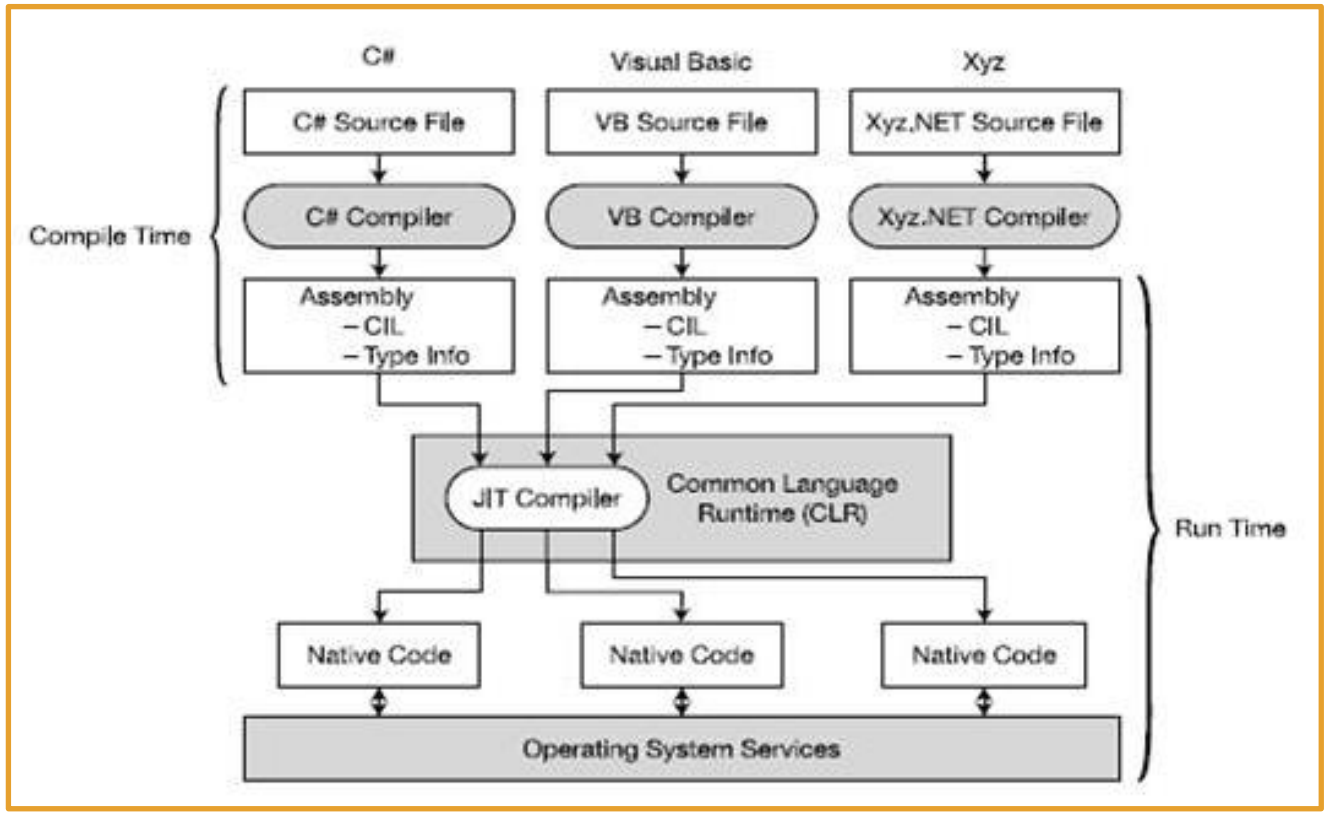




CLI - Two views, one model

https://docs.microsoft.com/en-us/dotnet/standard/managed-execution-process





CTS (Common Type System)

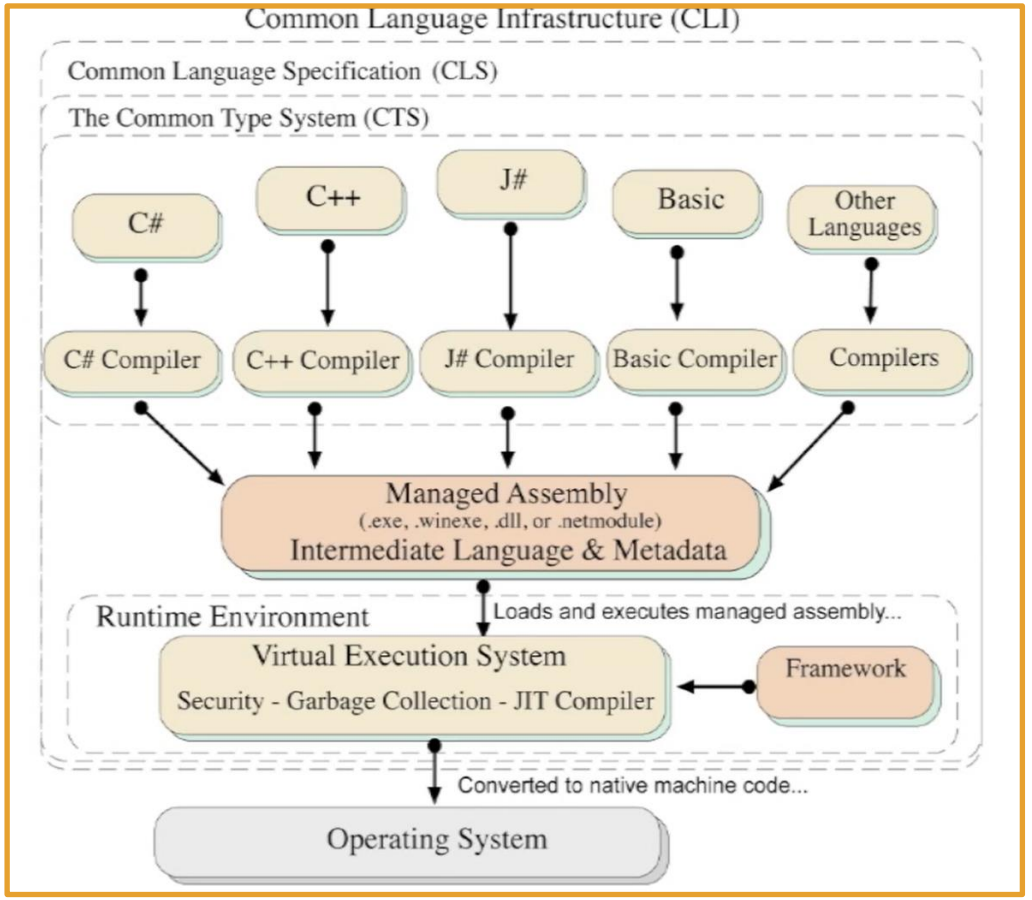
https://docs.microsoft.com/en-us/dotnet/standard/common-type-system
https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview
https://www.youtube.com/watch?v=b7L03h7nMwg

The ***CTS*** is used by the ***CLR*** to enforce strict ***typing*** and code verification.

To ensure that all compilers generate managed code that conforms to the ***CTS***, the ***CTS***:

1. describes all data types and all related constructs which are supported by the ***CLR***
2. details how they must be represented in the .NET metadata format
3. specifies how entities can interact with each other

This means that ***managed code*** can consume other managed types and instances, while strictly enforcing type fidelity and type safety.



CTS (Common Type System)

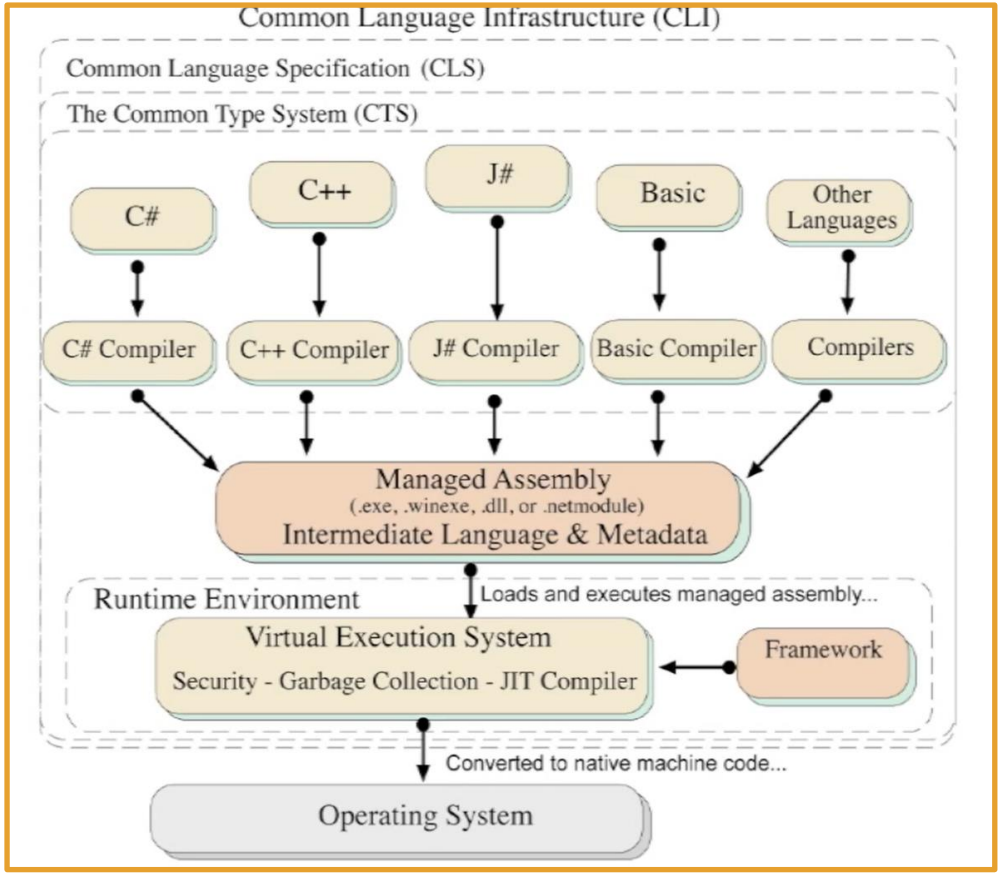
https://docs.microsoft.com/en-us/dotnet/standard/common-type-system
https://www.youtube.com/watch?v=b7L03h7nMwg

The ***CTS*** provides a library of the basic primitive data ***types*** to be used in application development.

- ***CTS*** defines the two main ***types*** (reference and value types) that should be supported.
- ***CTS*** defines several categories of ***types***, each with their specific semantics and usage:

Classes	Structs	Enums	Interfaces	Delegates

- ***CTS*** defines all other properties of the ***types*** (access modifiers, valid type members, how inheritance and overloading works, etc).



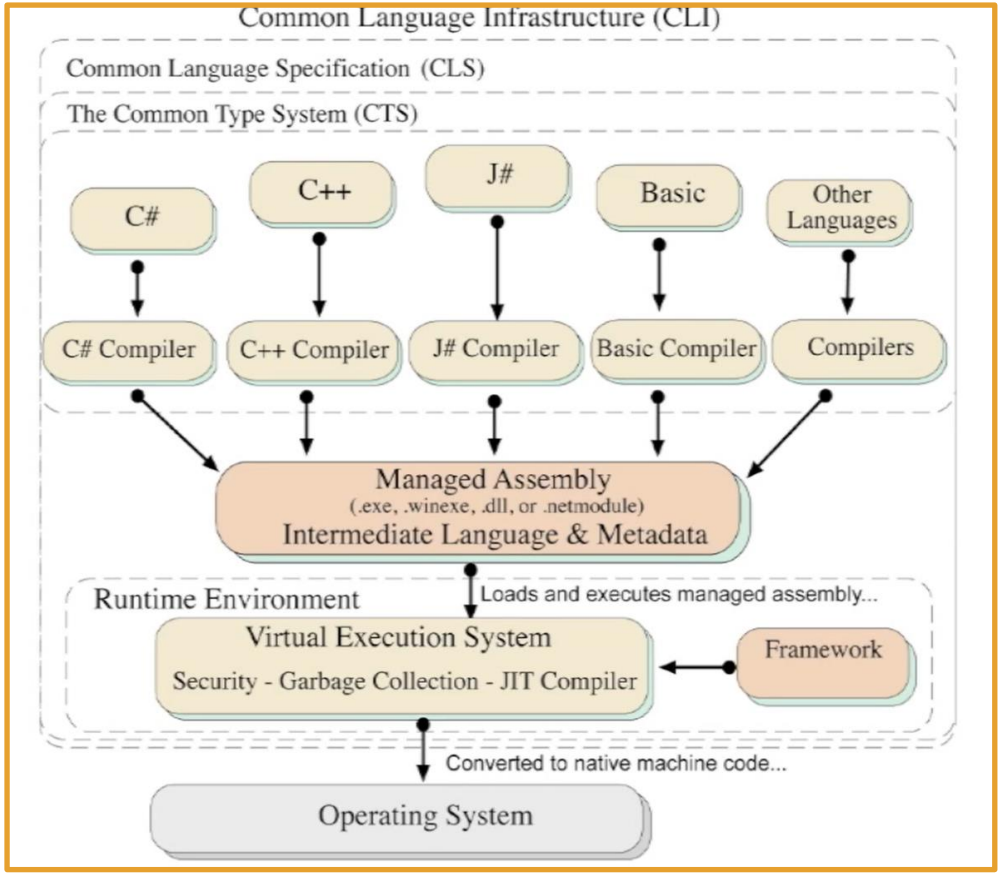
CLS (Common Language Specification)

https://docs.microsoft.com/en-us/dotnet/standard/common-type-system

Since there are numerous different languages, .NET has specified the commonalities required to enable full interoperability between languages into something called the ***Common Language Specification (CLS)***.

CLS is a subset of the ***CTS***. This means that all rules of the ***CTS*** apply to the ***CLS*** also.

CLS defines a set of rules and restrictions that every language must follow which runs under the .NET framework. It provides a sort of recipe for any language that is implemented on top of .NET on what it must support.



CIL (Common Intermediate Language)

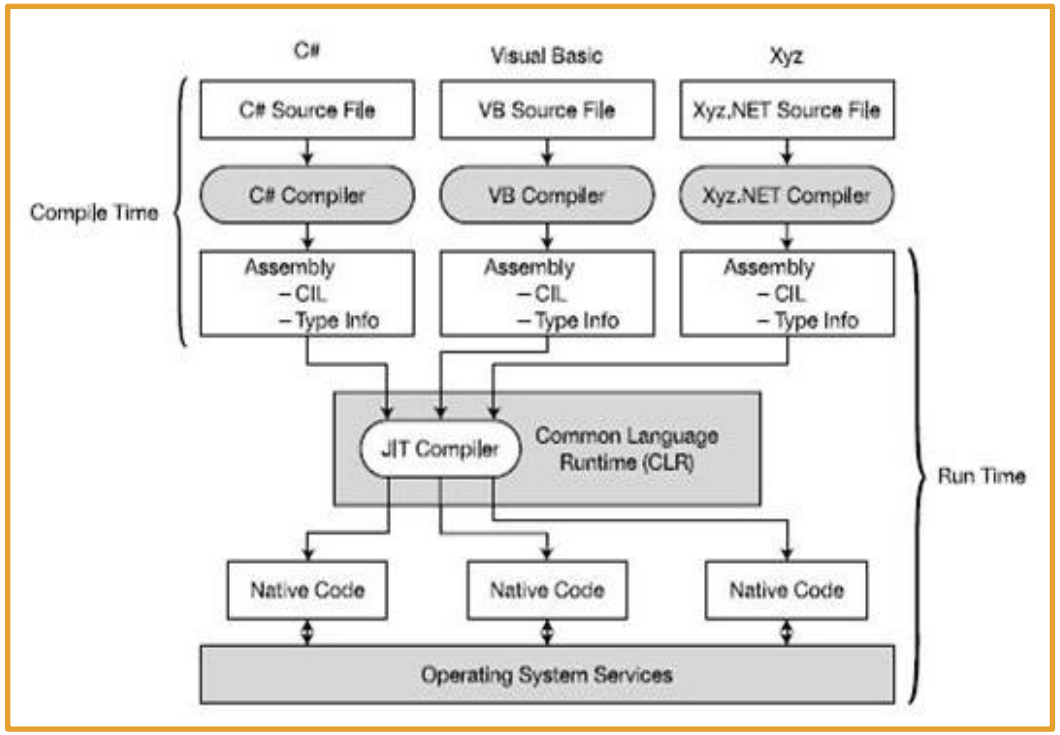
https://docs.microsoft.com/en-us/dotnet/standard/managed-execution-process#compiling-to-msil

Common Intermediate Language (CIL)* is the intermediate language binary instruction set defined within the ***Common Language Infrastructure (CLI)***.

CIL instructions are executed by the ***Common Language Runtime***.

Languages which target the ***CLI*** compile to ***CIL***.

Runtimes ***Just-In-Time(JIT)*** compile ***CIL*** instructions into native code.



*CIL was formerly called ***Microsoft Intermediate Language (MSIL)*** or ***Intermediate Language (IL)***.

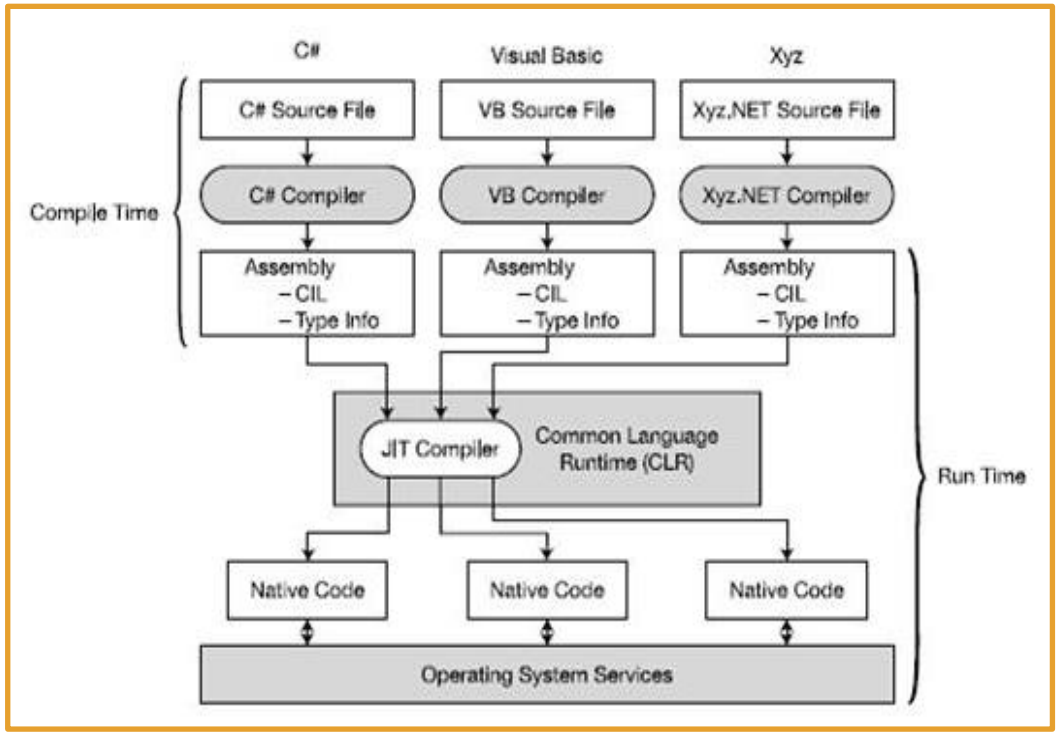
JIT (Just In Time Compiler)

https://docs.microsoft.com/en-us/dotnet/standard/managed-execution-process
https://docs.microsoft.com/en-us/dotnet/standard/managed-execution-process#compilation-by-the-jit-compiler

---

Just-In-Time (JIT) **compilation** involves compilation of a program at run time.

JIT compilation assumes that some code might never be called during execution. Instead of using time and memory to convert all the ***CIL*** in a ***PE (portable executable)*** file to native code, it converts the ***CIL*** as needed during execution and stores the resulting native code in memory so that it is accessible for subsequent calls in the context of that process.



VES (Virtual Execution System)

https://en.wikipedia.org/wiki/Virtual_Execution_System

A ***Virtual Execution System (VES)*** is a run-time system of the ***Common Language Infrastructure (CLI)*** which provides an environment for executing ***managed code*** where the ***Common Intermediate Language (CIL)*** instruction set can be executed.

The ***Common Language Runtime (CLR)*** is the .NET Framework's implementation of a **VES.** It provides direct support for a set of built-in data types. It defines:

- a hypothetical machine with an associated machine model and state,
- a set of control flow constructs, and
- an exception handling model.