.NET keretrendszer
Bevezető

Jánosi-Rancz Katalin Tünde

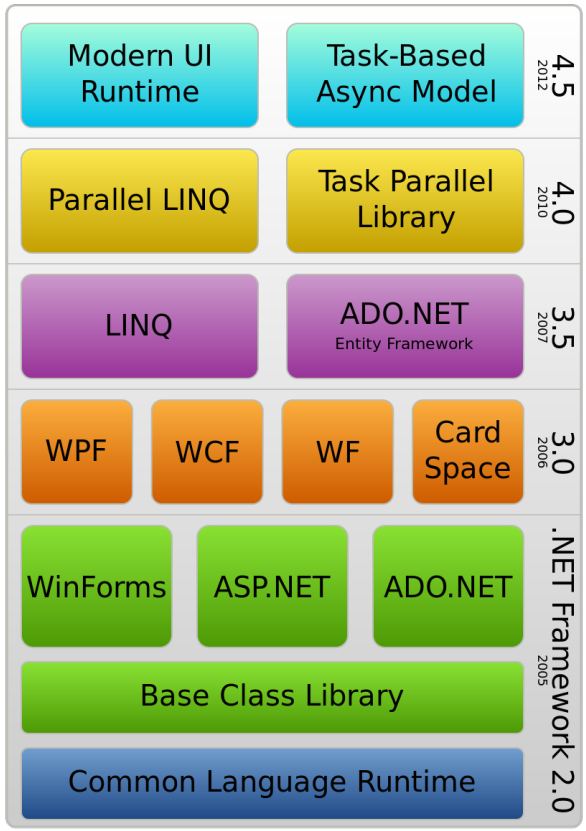
Sapientia EMTE
tsuto@ms.sapientia.ro

Történelem

- A Java (1995) gyökerei a C++ (1983, 1992)-ból erednek, C++ számos kellemetlen szintaktikai jellemvonását kiküszöbölte
- Java: jelentős méretű osztálykönyvtár minden lehetséges feladatra
- A Java kismértékben nyújt nyelvi integritást (sok forráskód létezik, s ezeket előnyös lenne Java-kódból felhasználni)
- A Java sok tekintetben a C++ letisztult verziója
- a C# -t (2000) a Java letisztult verziójának tekinthetjük

Miért C# ?

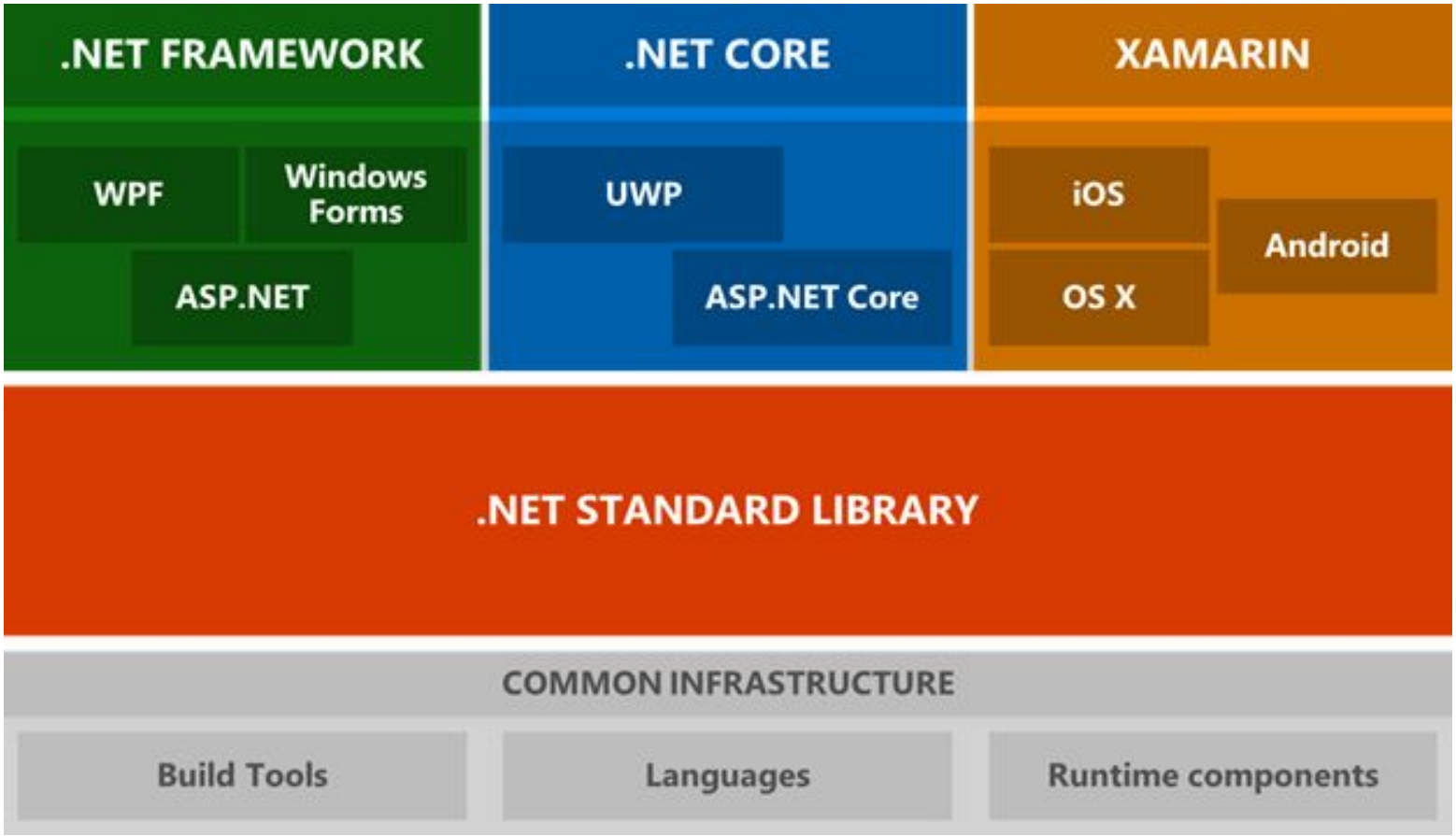
- Együttműködési képesség a meglévő forráskóddal (pl. C-alapú könyvtárak hívása a .NET-kódból)
- Teljes nyelvi integráció
- Bájtkód szinten az egymás kódját tudjuk egy másik programozási nyelvben használni
- Támogatja a web-es protokollokat (HTTP, XML, XPath, SOAP)
- Átfogó alaposztálykönyvtár
- Egy valóban leegyszerűsített telepítési modell (a .NET lehetővé teszi, hogy egyetlen gépen ugyanazon .dll több verziója legyen jelen)



The .NET Framework Stack

http://en.wikipedia.org/wiki/File:DotNet.svg

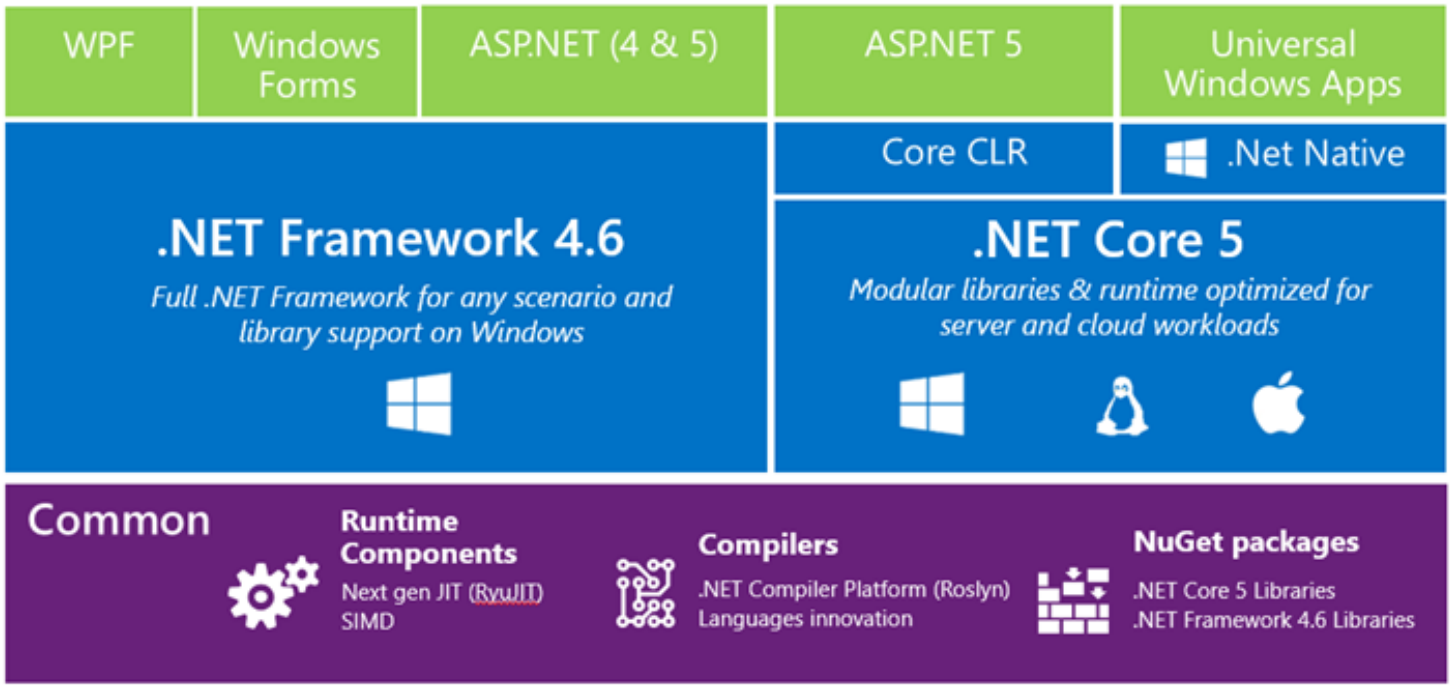
.NET Ecosystem



.NET Core - a .NET jövője, egy cross-platform, egységes, gyors, könnyű, modern, nyílt forráskódú keretrendszer. Mobil, webes és Windows alkalmazások és szolgáltatások fejlesztésére szolgál, Windows, Linux és Mac operációs rendszereken.

Miért C# ?

- A .NET nem korlátozódik a Windows operációs rendszerre
- platform független, létezik Linux és Mac fordító is
- .NET Framework 4.7, .NET Core 5
- 8-10 milló .NET fejlesztő
- Open source!



.NET keretrendszer

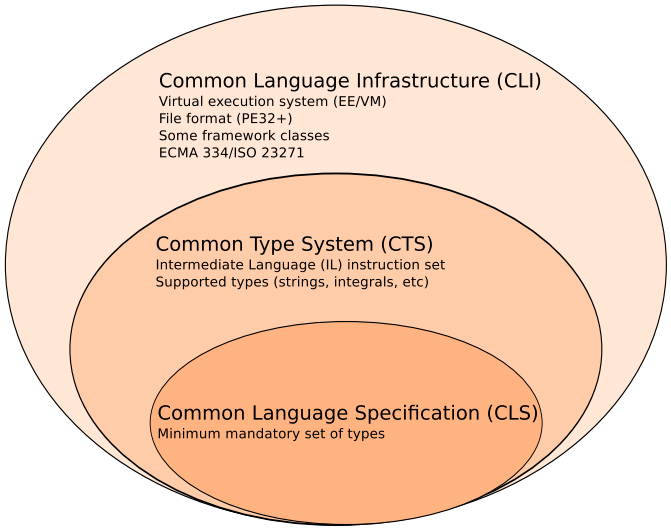
- A .NET **NEM** egy programozási nyelv, hanem egy környezet!

Szoftver keretrendszer

- A hordozhatóságot, gyors futást és hibamentességet a **futásidejű fordítás** (Just In Time Compilation) biztosítja
  - az eredeti kód egy alacsony szintű, de gépfüggetlen kódra fordul, az a köztes nyelvű kód (Intermediate Language)
  - a köztes nyelvű kódot külön kell értelmezni, erre szolgál a futtatórendszer - CLR, Javaban virtuális gép -Virtual Machine, ami egyúttal lehetőséget ad a memóriafelügyeletre
- Szoftver keretrendszernek nevezzük a kiemelt programkönyvtár és a futtatórendszer együttesét
  - tartalmazza az API gépfüggetlen, absztrakt lefedését
  - felügyeli a programok futásának folyamatát

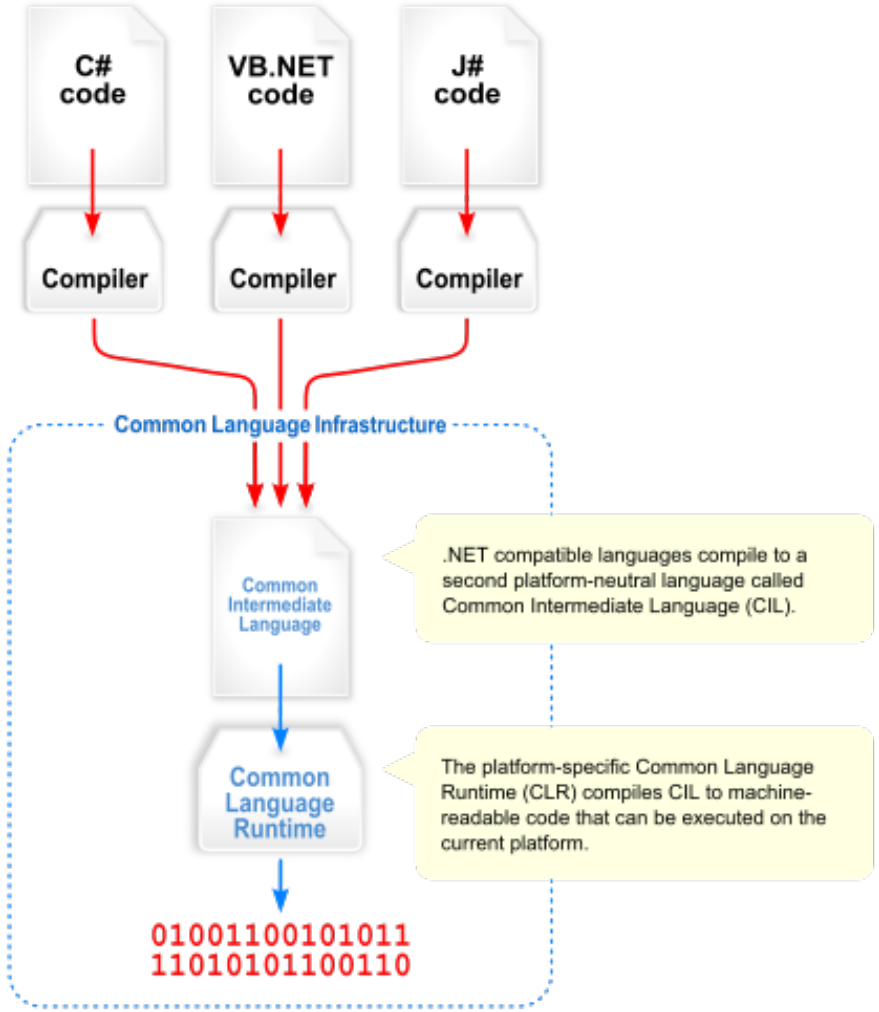
.NET platform építőelemei (CLI, CTS, CLS, CLR)

- A .NET = futtatókörnyezet + átfogó alaposztálykönyvtár
  - közös nyelvi futtatórendszer: Common Language Runtime (CLR)
    - feladata: megkeresse, betöltse, kezelje a .NET típusokat; memóriakezelés
  - platformfüggetlen köztes nyelv: Common Intermediate Language (CIL)
  - egységes típusrendszer: Common Type System (CTS)
    - feladata: leírja az összes adattípust, meghatározza ezek hogyan kommunikáljanak egymással
  - teljes körű programkönyvtár: Base Class Library (BCL), Framework Class Library (FCL)



Platformfüggetlen köztes nyelv - CIL

- A Common Language Infrastructure (CLI) a különböző programozási nyelven írt programok futtatására szolgáló alrendszere a .NET Keretrendszernek.
- .NET-programozási nyelvek
  - Kb 50 nyelv támogatott:
  - Microsoft: C# , Visual Basic .NET, F#, J#, C++/CLI, JScript .NET
  - más: Smalltalk, Cobol, Pascal, etc



A közös köztes nyelv (CIL) szerepe

- tekintsük a következő példát:

```
//Calc.cs
using System;
class Program {
  static void Main() {
    Calc c = new Calc();
    int ans = c.Add(10, 84);
    Console.WriteLine(”10 + 84 is {0}.”, ans);
  }
}

class Calc {
  public int Add(int x, int y){
    return x + y;
  }
}
```

- compilation: `csc.exe Calc.cs` $\longrightarrow$ `Calc.exe`

A Source Code-tól a CIL-ig

- `Add()` C# metódus `ildasm.exe`-t használva

```
.method public hidebysig
      instance int32
Add(int32 x, int32 y) cil
      managed
{
//  A kód mérete 9 (0x9)
.maxstack 2
.locals init (int32 V_0)
IL_0000: nop
IL_0001: ldarg.1
IL_0002: ldarg.2
IL_0003: add
IL_0004: stloc.0
IL_0005: br.s      IL_0007
IL_0007: ldloc.0
IL_0008: ret
} // a Calc::Add metódus vége
```

- `Add()` VB metódus `vbc.exe`-t használva

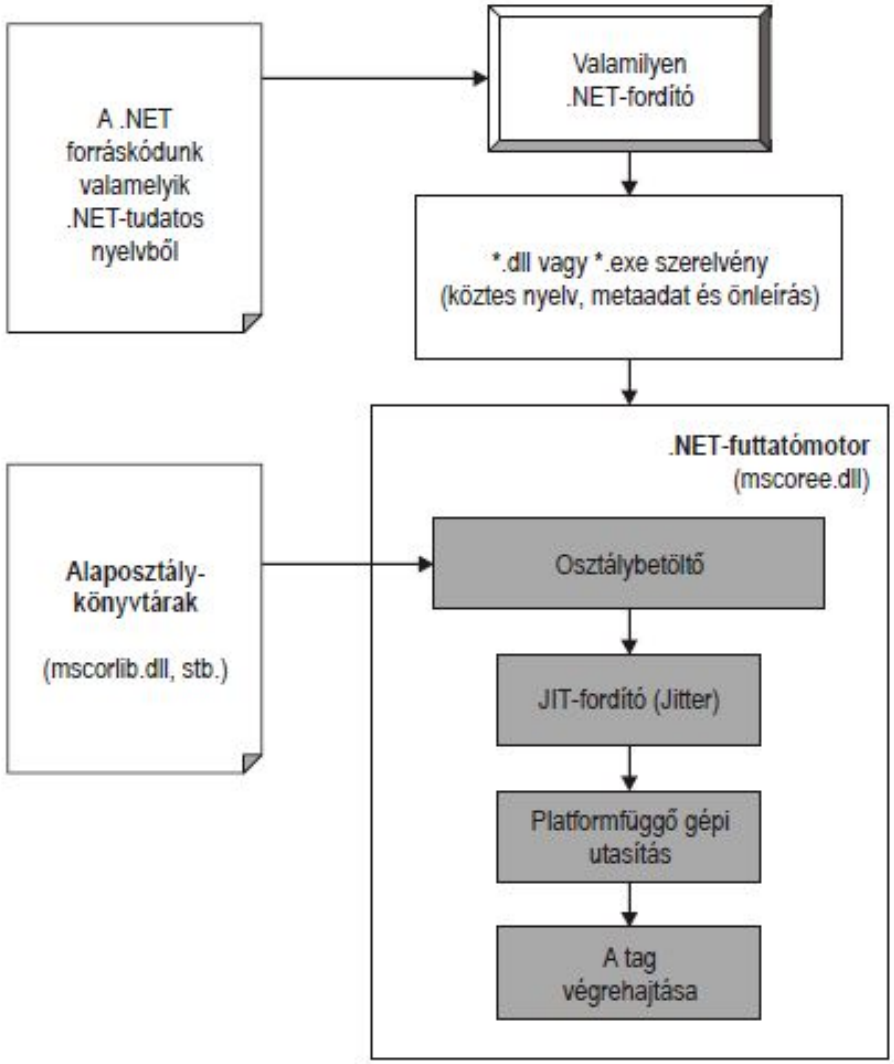
```
.method public
instance int32
Add(int32 x, int32 y) cil
      managed
{
// A kód mérete 8 (0x8)
.maxstack 2
.locals init (int32 V_0)
//
IL_0000: ldarg.1
IL_0001: ldarg.2
IL_0002: add.ovf
IL_0003: stloc.0
IL_0004: br.s      IL_0006
IL_0006: ldloc.0
IL_0007: ret
} // a Calc::Add metódus vége
```

CIL előnye

- Nyelvi integráció
- Minden egyes .NET-alapú fordító közel azonos CIL-utasításokat hoz létre
- Az összes nyelv egy jól meghatározott bináris térben kommunikálhat egymással
- a CIL platformfüggetlen (vagyis egy szimpla kódbázis fut több operációs rendszeren).

CLR - közös nyelvi futtatórendszer

- a kód egy adott lefordított egységének a végrehajtásához szükségesek
- jól meghatározott futtatási réteg, amelyet az összes .NET-alapú nyelv és -platform használ
- mscoree.dll
- CLR $\neq$ virtuális gép
  - nincs .NET virtuális gép, felügyelt (vagy managed) kódot használ
  - a program a processzoron fut
  - mellette a keretrendszer (felelős a memóriafoglalásért, kivételek kezeléséért...)



forráskód, egy adott .NET-fordító és a .NET-futtatómotor közötti munkafolyamat

Nyelvsemleges .NET-kódkönyvtárak

- a C# nem rendelkezik nyelvfüggő kódkönyvtárral!
- bármelyik nyelv, amelyik a .NET-futtatórendszert használ, **ugyanazokat** a névtereket és **ugyanazokat** a típusokat használja

```
// Hello world C# nyelven
using System;
public class MyApp
{
    static void Main()
    {
        Console.WriteLine(”Hi from C#”);
    }
}
```

```
// Hello world VB nyelven
Imports System
Public Module MyApp
    Sub Main()
        Console.WriteLine(”Hi from VB”)
    End Sub
End Module
```

```
// Hello world C++/CLI nyelven
#include ”stdafx.h”
using namespace System;
int main(array<System::String ^> ^args)
{
    Console::WriteLine(L”Hi from C++/CLI”);
    return 0;
}
```

- mindhárom ugyanazt a System névteret használja

A C# hátrányai

- megnövelt erőforrásigény
- lassabb programindítás és futtatás
- kód visszafejthetőség

Integrált fejlesztőkörnyezet (IDE)

- Visual Studio
  - csapatmunka támogatás (verziókövetés, feladatkövetés)
  - tervezés (architektúra, végrehajtás, terv-kód szinkronizálás)
  - kód-kiegészítés, kód-újratervezés, kód-stílus követés
  - hibakeresés, tesztelés, kódelemzés
- kiegészítő eszközök:
  - natív kódelemzés (IL Disassembler, .NET Reflector)
  - statikus kód elemzés (FxCop, NDepend, StyleCop)
  - kódrészletek előállítása (Snipper Compiler, Regulator)
  - kód-újratervezés (Resharper)
  - automatizált tesztelés (NUnit, TestDriven.NET CruiseControl.NET, Moq)
  - teljesítménytesztelés (dotTrace)
  - automatikus dokumentálás (GhostDoc, NDoc, Sandcastle)

Hibakeresés

- Hibakeresés (debugging) során a programot futás közben elemezzük, követjük a változók állapotait, a hívás helyét, felfedjük a lehetséges hibaforrásokat
- fontosabb hibakövetési funkciók:
  - megállási pontok (breakpoint) elhelyezése, ahol a program futása megáll, és várakozó állapotba lép
  - változókövetés (watch), amely automatikus a lokális változókra, szabható rá feltétel
  - hívási lánc (callstack) kezelése, a felsőbb szintek változóinak nyilvántartásával
  - léptetés (step-in): az utasítások lépésenkénti végrehajtása
- kódolás: Unicode 3.0

Fordítás és futtatás

- A natív programok (pl. C++) $\Rightarrow$ gépi kódra fordulnak le
- amíg a .NET forráskódokból $\Rightarrow$ CIL futtatható állomány (Java-ban is van köztes nyelv)
- A fordító nem platformfüggő gépi utasításokat ad ki, hanem CIL-re fordít
- futtatva a CIL állományokat
  - először JIT (Just–In–Time) fordító lefordítja őket gépi kódra
  - a processzor már képes kezelni
- "első" fordításnál keletkezik egy Assembly (vagy szerelvény)
  - Tartalmazza a Metadata-t (felhasznált típusok adatait, osztályok szerkezete, metódusai, stb.)
  - egy vagy több fileból is állhat (.exe vagy .dll)
  - a .NET-szerelvények csak olyan gépeken futtathatók, amelyekre telepítve van a .NET-keretrendszer!

Fordítás és futtatás

- A fordítás módja konfigurálható (VS-ban)
- 2 üzemmód támogatott:
  - Debug: nyomkövetéshez optimalizált, így a kód számos kiegészítést tartalmaz
  - Release: végleges használatra optimalizált, így a kód jelentős optimalizáláson megy keresztül
- Megadhatjuk a célplatformot (x86, x64, vagy tetszőleges)
- A lefordított program tartalmazza az eredeti kódot, és visszafejthető (pl. .NET Reflector segítségével)

C# kód töredékek (Snippets)

- Kód töredék és 2 TAB beszúr egy kódrészletet
  - ctor – constructor
  - prop – property
  - propfull – full property
  - if – if format
  - switch – switch format
  - foreach – foreach format
  - ex - saját kivételosztályt generál

```
Lásd Code/Conventions kódolási egyezményt is
```

Feladat

- Parancssorból nézzünk meg az Add metódus CIL kódját
  - csc Calc.cs

```
csc.exe: C:\Windows\Microsoft.NET\Framework\v3.5
```

  - isdalm Calc.exe

```
isdalm.exe:   C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0
    A\Bin
```