The .NET Architecture

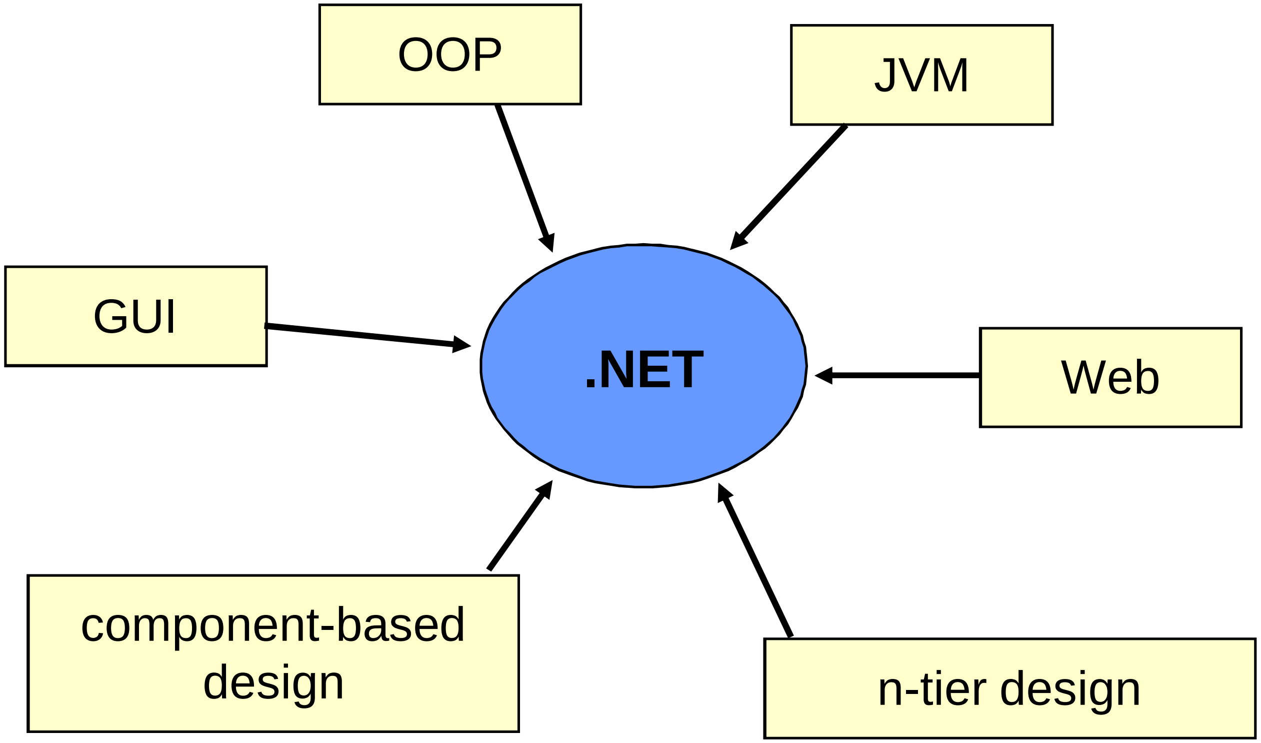
Objectives

“Microsoft .NET is based on the .NET Framework, which consists of two major components: the Common Language Runtime (CLR) and an extensive set of Framework Class Libraries (FCL). The CLR defines a common programming model and a standard type system for cross-platform, multi-language development.”

- **CLR-based execution**
- **Application designs**

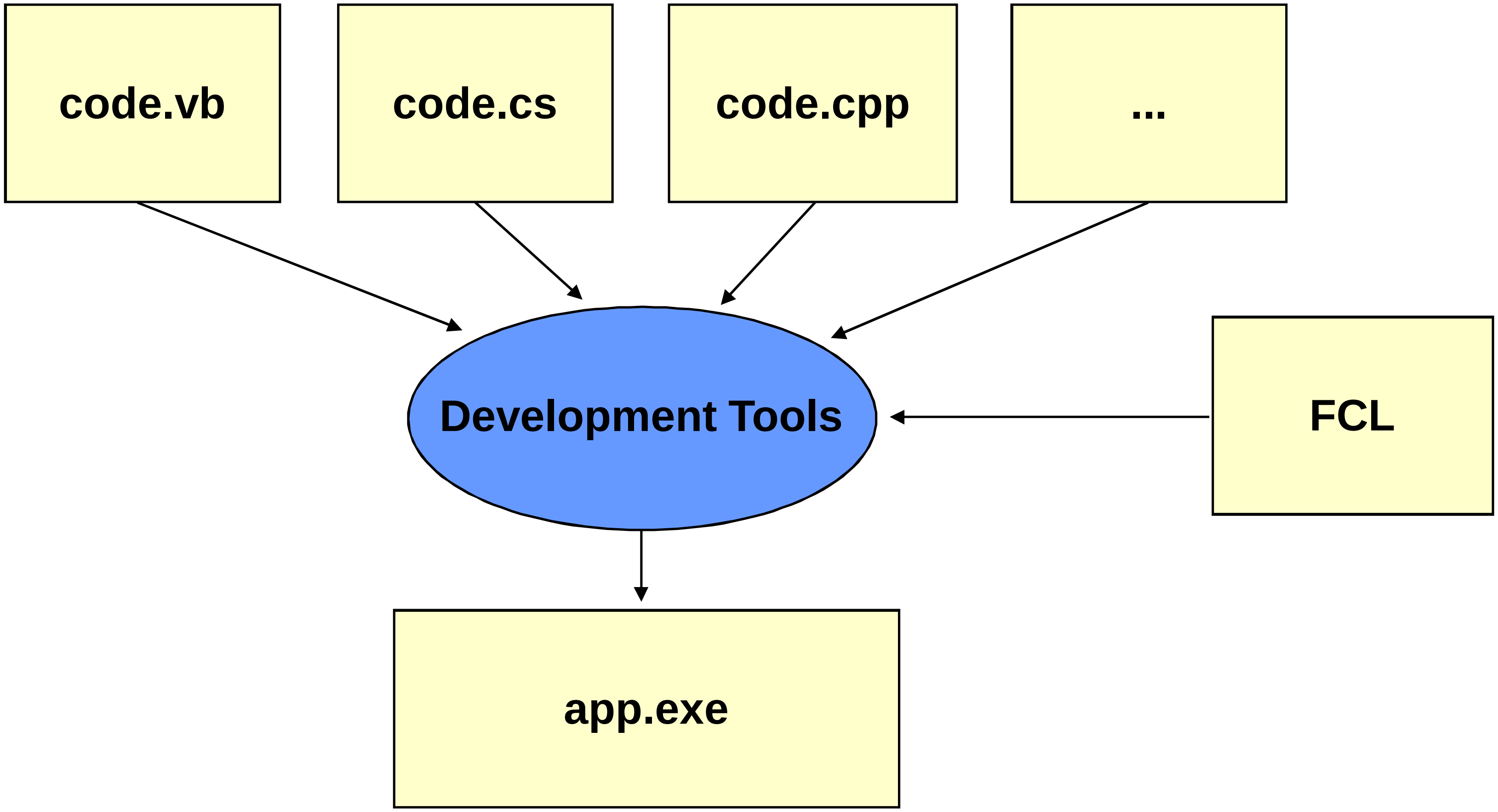
CLR-based execution…
Influences

- .NET is the result of many influences…



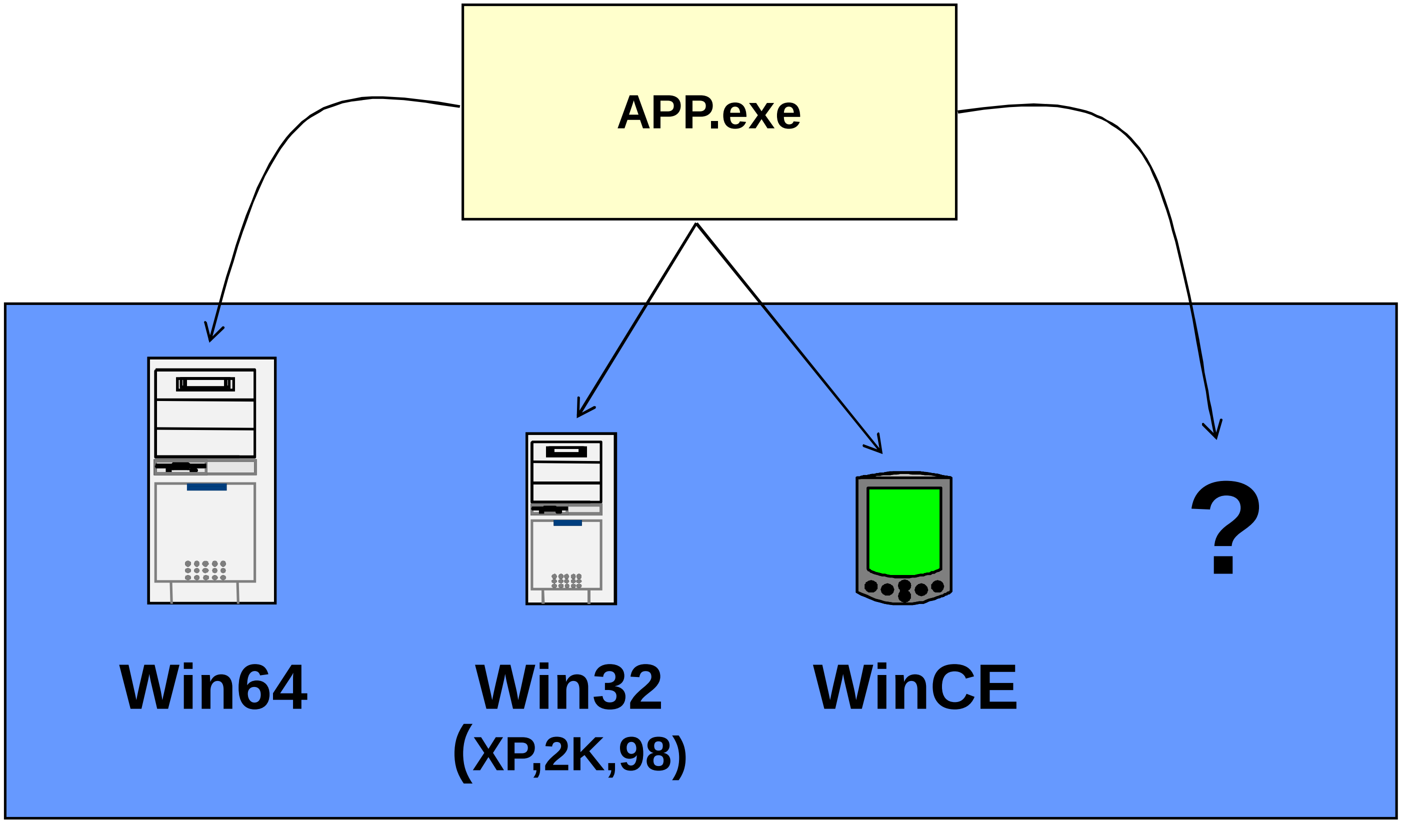
.NET is multi-language

- .NET supports VB, C# (C-sharp), C++, J# (Java 1.2), Eiffel, etc.



.NET is cross-platform

- Compiled .NET apps run on any supported platform:



How is cross-platform achieved?

- **Cross-platform execution realized in two ways:**

  1. apps are written against *Framework Class Library* (FCL), not underlying OS

  2. compilers generate generic assembly language which must be executed by the *Common Language Runtime* (CLR)

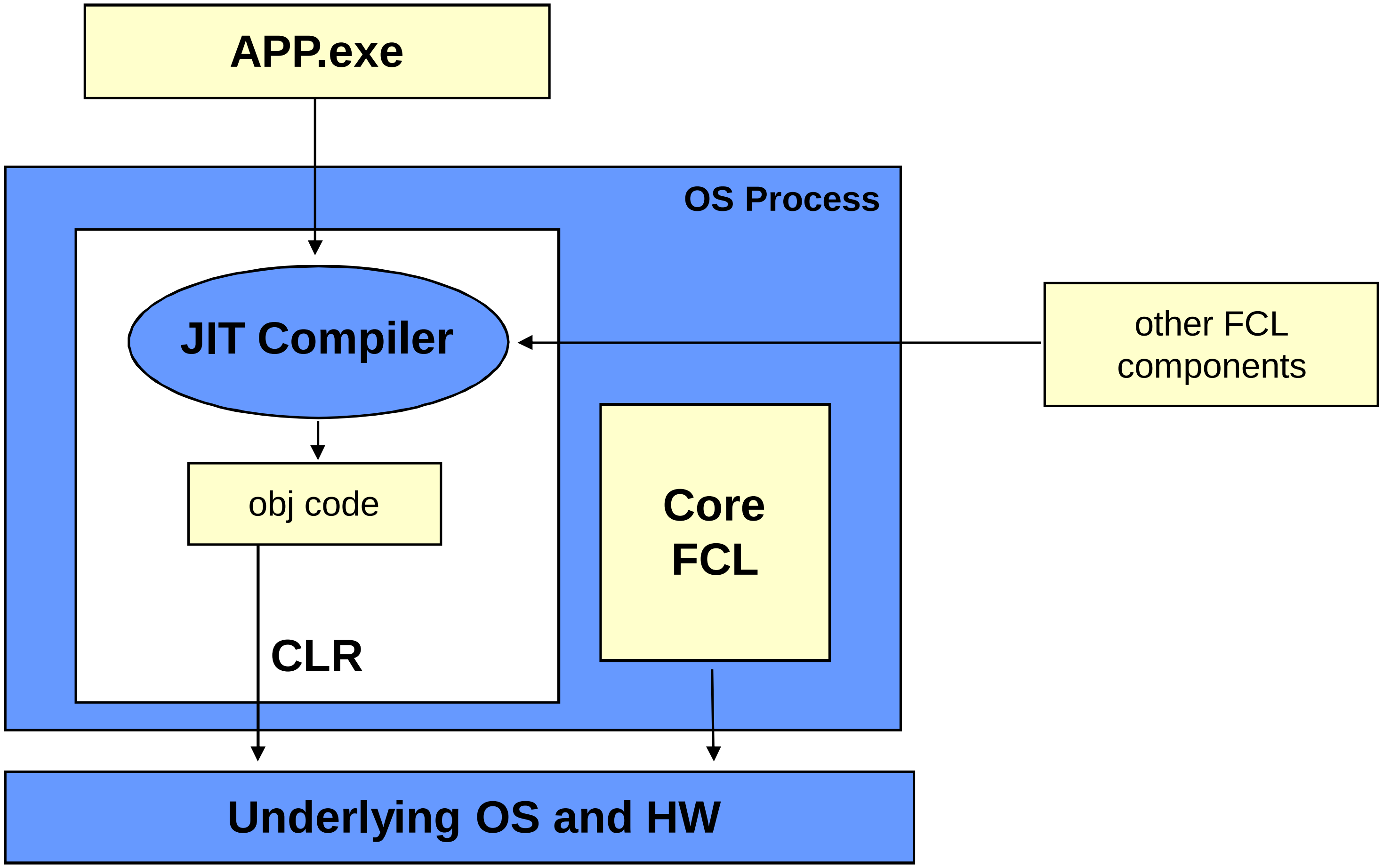
(1) FCL

- **Framework Class Library**
  - 1000's of predefined classes
  - common subset across all platforms & languages
  - networking, database access, XML processing, GUI, Web, etc.

- **Goal?**
  - FCL is a portable operating system

(2) CLR-based execution

- **Common Language Runtime must be present to execute code:**



Implications of .NET's execution model

1. **Clients need CLR & FCL to run .NET apps**
   - available via *Redistributable .NET Framework*
   - 20MB download
   - runs on 98 and above, NT (sp6a) and above

2. **Design trade-off…**

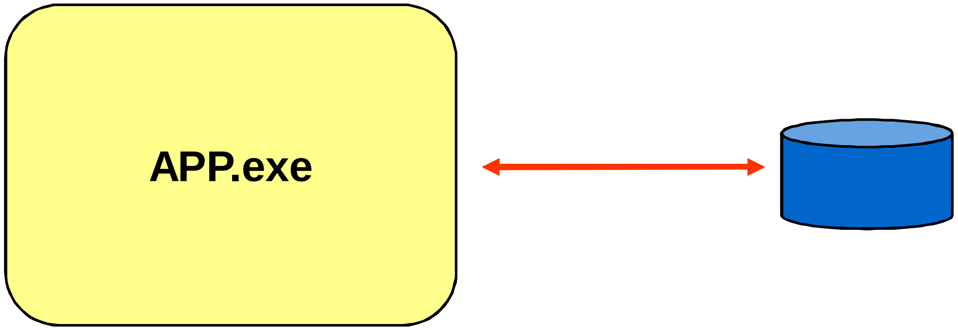
   + managed execution (memory protection, verifiable code, etc.)

   + portability:

   – slower execution?

Application design…
Monolithic

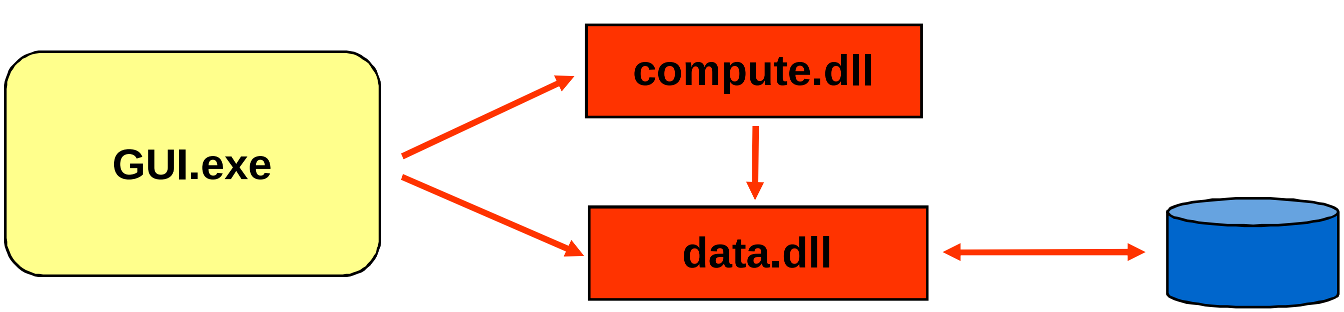
- **Monolithic app: all source code compiled into one .EXE**



  - ***not*** the norm on Windows…

Component-based

- **Component-based app: .EXE + 1 or more .DLLs**



  - standard practice on Windows…

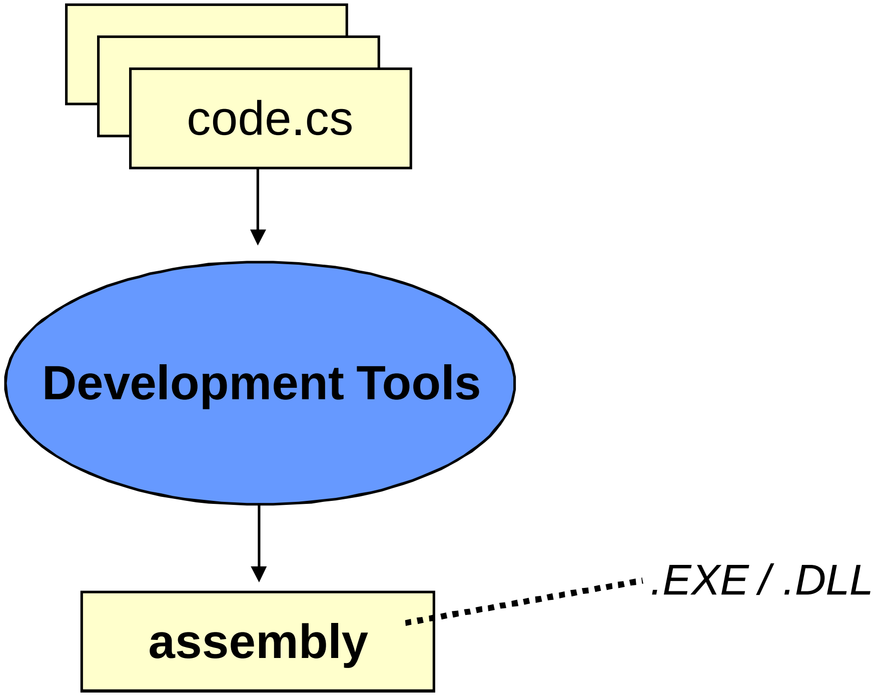
Why component-based?

- **Many motivations:**
  - team programming
  - multi-language development (I like VB, you like C#)
  - code reuse (e.g. across different .EXEs)
  - independent updating (update just component X)

- **FCL ships as a set of components!**

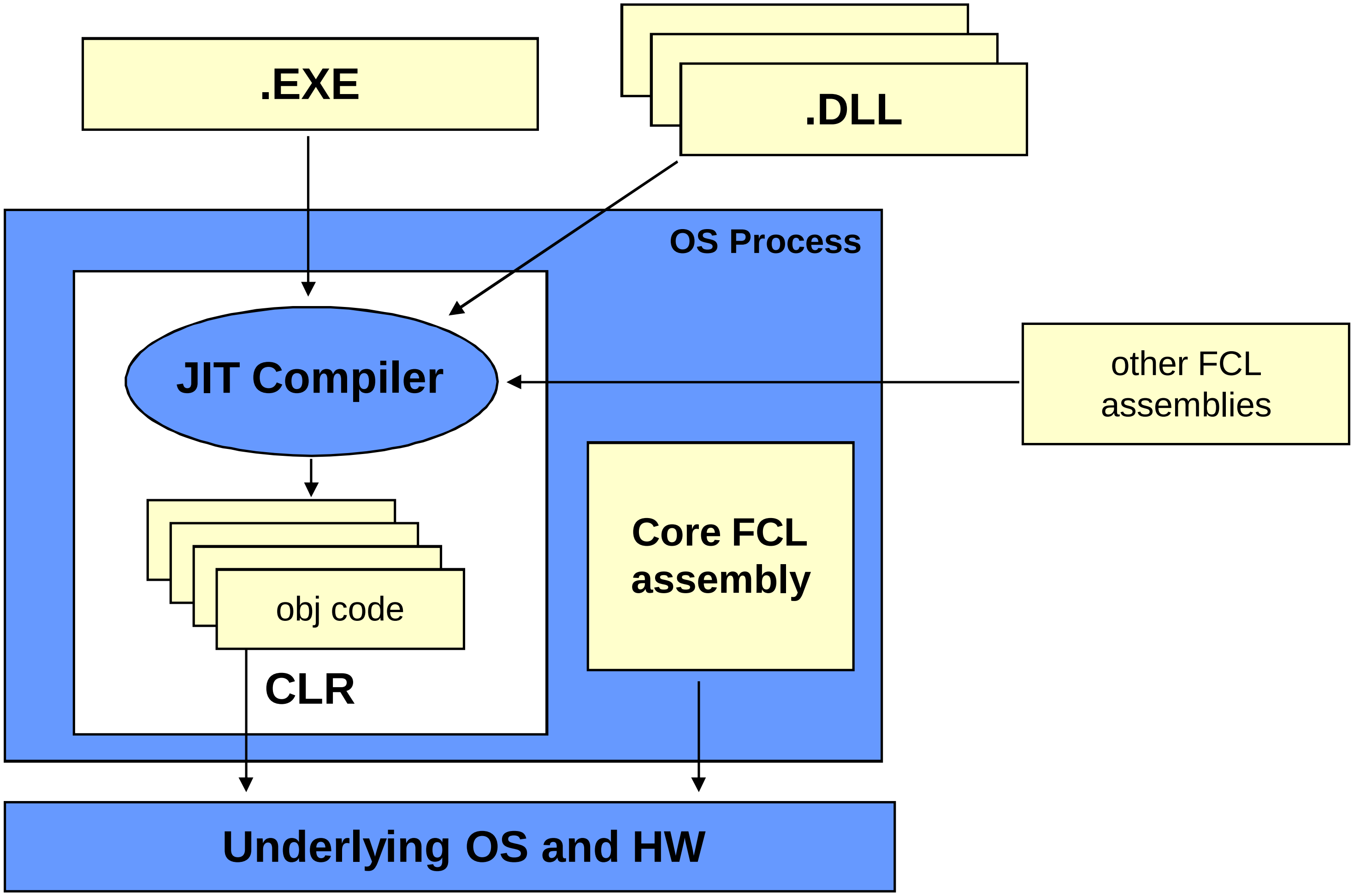
Assemblies

- **.NET packages components into *assemblies***
- **1 assembly = 1 or more compiled classes**
  - .EXE represents an assembly with classes + Main program
  - .DLL represents an assembly with classes



CLR-based execution revisted

- **CLR must be able to locate all assemblies:**



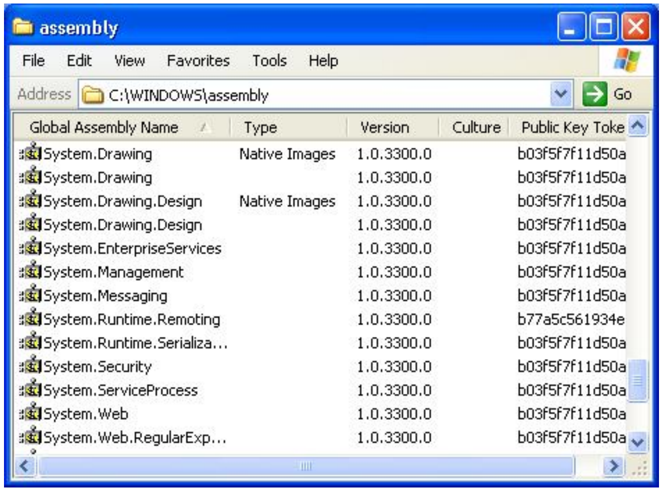
Assembly resolution

- **How does CLR find assemblies?**

- **For now, simple answer is sufficient:**
  - our DLLs must reside in same directory as our EXE
  - FCL assemblies reside in GAC
  - CLR looks in GAC first, then EXE's directory…

GAC?

- **GAC = Global Assembly Cache**
  - C:\Windows or C:\WinNT directory



C:\WINDOWS\assembly

Global Assembly Name	Type	Version	Culture	Public Key Toke
System.Drawing	Native Images	1.0.3300.0		b03f5f7f11d50a
System.Drawing		1.0.3300.0		b03f5f7f11d50a
System.Drawing.Design	Native Images	1.0.3300.0		b03f5f7f11d50a
System.Drawing.Design		1.0.3300.0		b03f5f7f11d50a
System.EnterpriseServices		1.0.3300.0		b03f5f7f11d50a
System.Management		1.0.3300.0		b03f5f7f11d50a
System.Messaging		1.0.3300.0		b03f5f7f11d50a
System.Runtime.Remoting		1.0.3300.0		b77a5c561934e
System.Runtime.Serializa...		1.0.3300.0		b03f5f7f11d50a
System.Security		1.0.3300.0		b03f5f7f11d50a
System.ServiceProcess		1.0.3300.0		b03f5f7f11d50a
System.Web		1.0.3300.0		b03f5f7f11d50a
System.Web.RegularExp...		1.0.3300.0		b03f5f7f11d50a

- **Observations:**
  - explorer yields a flat view of GAC
  - command-shell yields actual representation
  - GAC can hold different versions of the same assembly
  - some assemblies have been pre-JIT ("native image")
  - tamper proof via digital signatures…

Summary

- **.NET architecture is:**
  - multi-language
  - cross-platform
  - based on the CLR, FCL, and JIT technology

- **Application designs are typically multi-tier**
- **Application designs yield component-based development**
  - .NET components are packaged as assemblies