Understanding Managed Code

the .NET Framework and Common Language Runtime

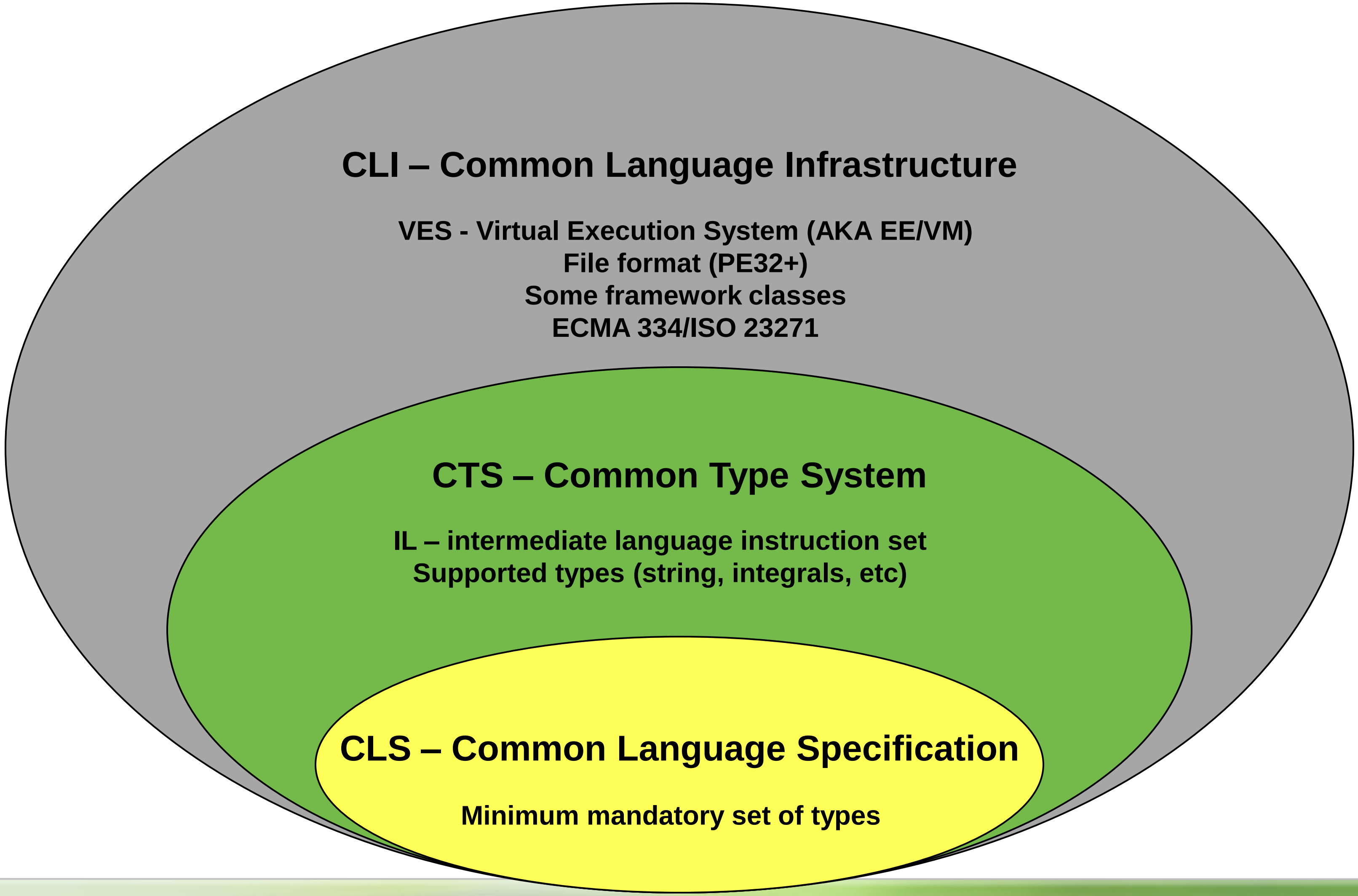
Outline

- **Overview of the .NET Framework**
  - □ elements of the framework
  - □ relevant standards
  - □ implementations
- **Overview of the Common Language Runtime (CLR)**
  - □ implementation overview
  - □ bootstrapping/initialization
  - □ intro to runtime services
- **Introduction to a few tools for analysis**
  - □ static code analysis
  - □ runtime/debugger-based analysis

The .NET Framework

- **The .NET Framework is a *managed execution platform***
  - an execution engine (EE)
    - AKA virtual machine (VM)
    - in charge of code execution (JIT compilation, security, …)
    - provides runtime services (memory management, I/O, …)
  - a set of class libraries
  - a set of standards describing scope of each

Standards



CLI – Common Language Infrastructure

VES - Virtual Execution System (AKA EE/VM)
File format (PE32+)
Some framework classes
ECMA 334/ISO 23271

CTS – Common Type System

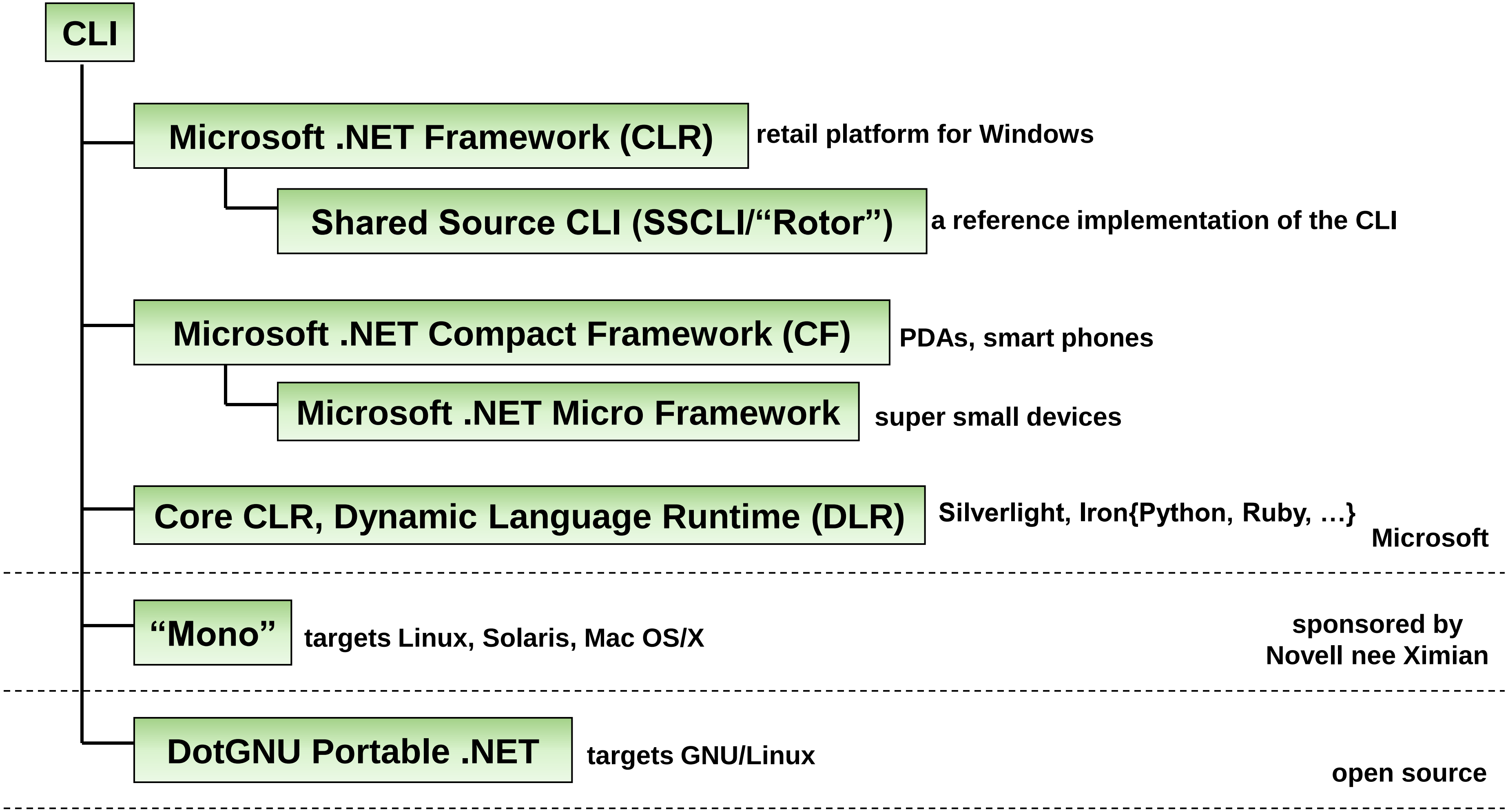
IL – intermediate language instruction set
Supported types (string, integrals, etc)

CLS – Common Language Specification

Minimum mandatory set of types

C# = ECMA 335/ISO 23270

CLI Implementations & Derivatives



CLI

Microsoft .NET Framework (CLR) retail platform for Windows

Shared Source CLI (SSCLI/“Rotor”) a reference implementation of the CLI

Microsoft .NET Compact Framework (CF) PDAs, smart phones

Microsoft .NET Micro Framework super small devices

Core CLR, Dynamic Language Runtime (DLR) Silverlight, Iron{Python, Ruby, …}

Microsoft

“Mono” targets Linux, Solaris, Mac OS/X

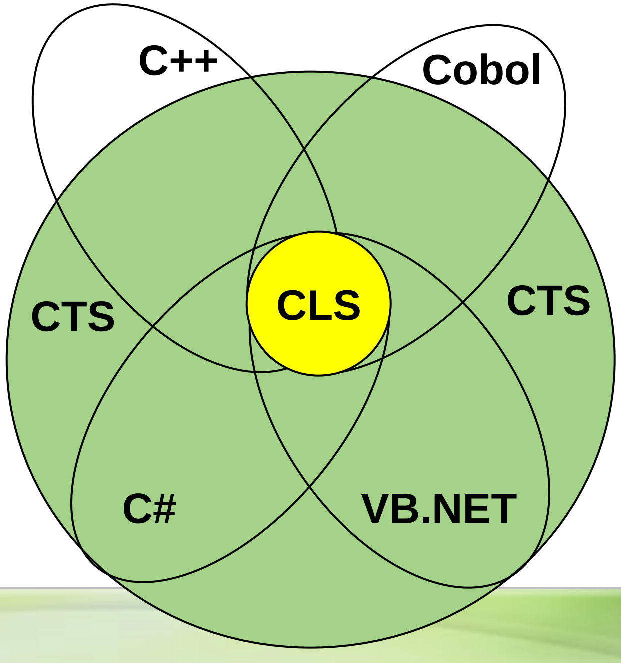
sponsored by
Novell nee Ximian

DotGNU Portable .NET targets GNU/Linux

open source

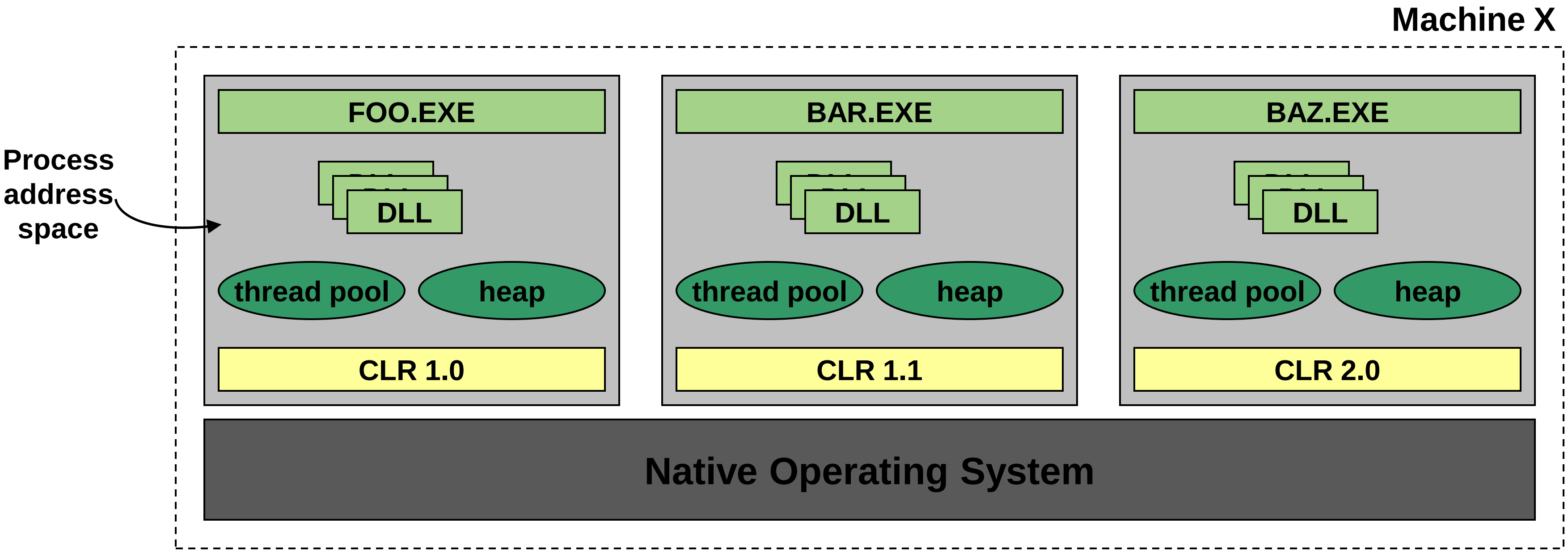
Language Support for the CTS

- **Language support for the CTS can vary**
  - languages do not have to support 100% of the CTS
    - each can choose a different subset of the CTS to support
  - languages do not have to limit themselves to the CTS
  - support for the CLS is the only shared requirement



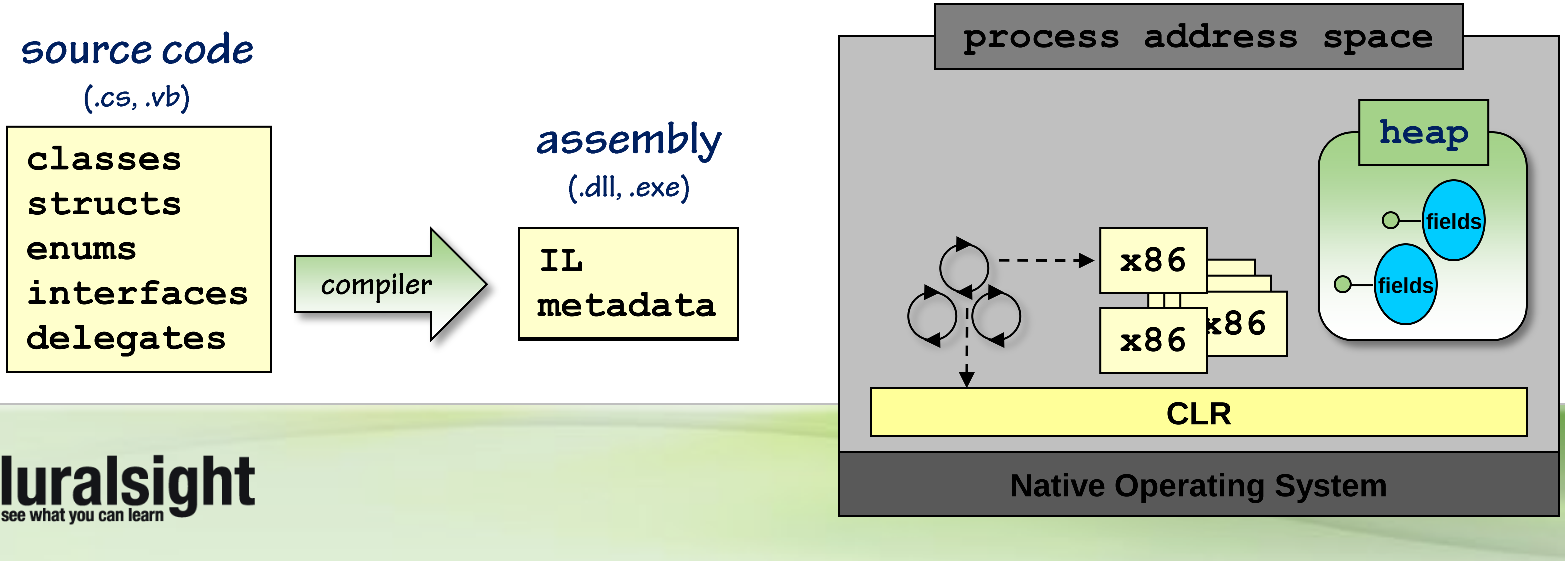
The Common Language Runtime

- **The CLR is implemented as a set of in-process DLLs**
  - loaded only into processes that run managed code
  - different apps can load different versions of the CLR
  - each process has its own runtime-specific resources
    - e.g.: heap, thread pool



From Development to Execution

- **Managed execution is characterized by…**
  - *types* described using a managed language (C#)
  - compiler produces an *assembly*
    - contains *intermediate language* (IL) and metadata
  - *assembly resolver* locates & loads assemblies
  - IL is just-in-time (JIT) compiled at runtime as needed
  - runtime services influence and/or facilitate execution
    - garbage collection (GC), security (CAS), reflection



Getting Started

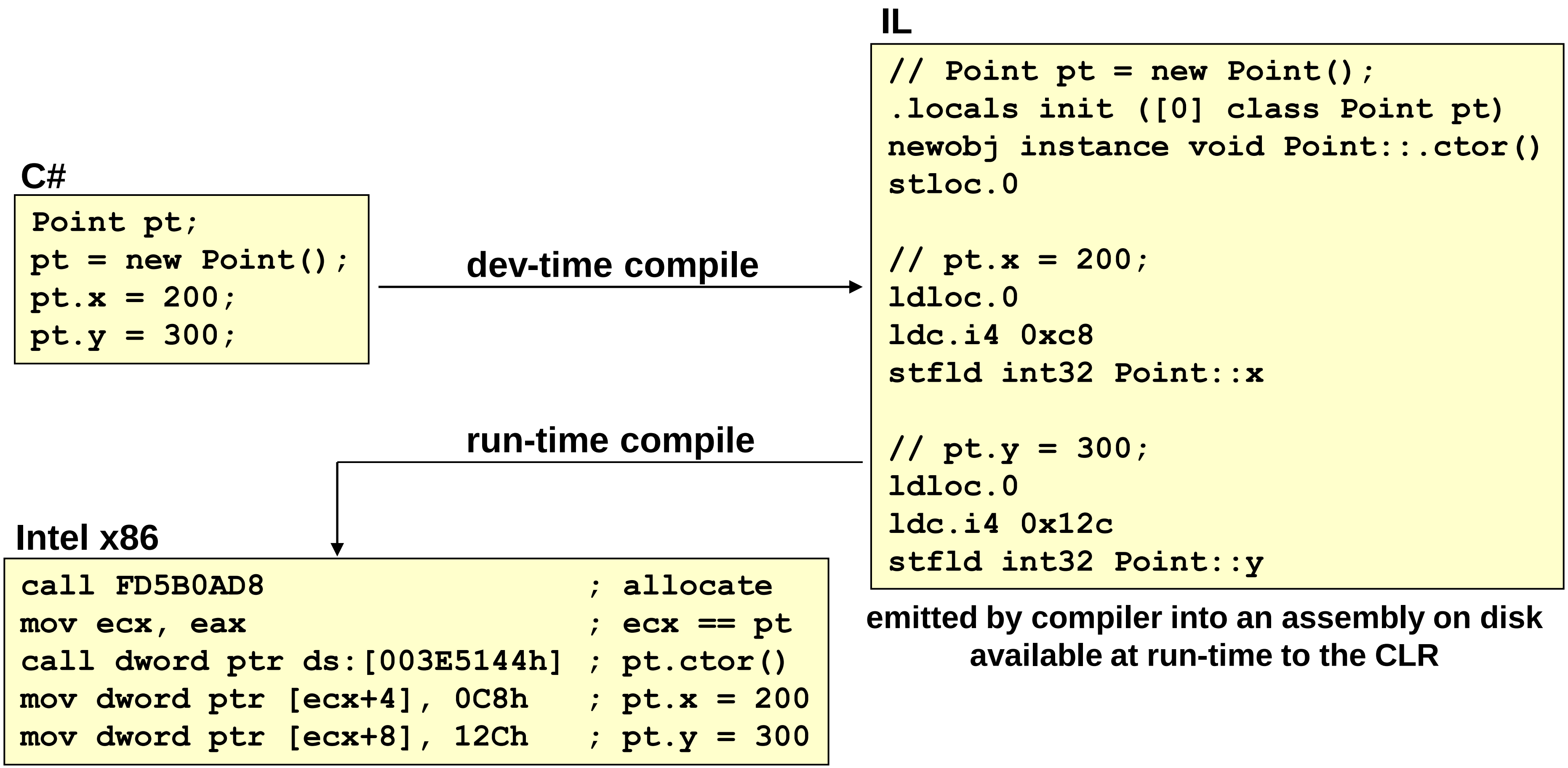
- **Each .NET program consists of a set of classes (types)**
  - Some classes you write
  - Thousands of existing classes available in the Framework Class Library (FCL)
    - String, Stack, Socket, etc.
- **A static entrypoint is where things get started**
  - Called “Main” by default in C#

```
class Program {
    static void Main() {
        System.Console.WriteLine("Hello, world!");
    }
}
```

Just-in-Time (JIT) Compilation

- **Processor-specific code is generated at *runtime***
  - IL is verified to be type safe
  - accommodates evolution of types
  - optimized for target machine, not dev machine
  - not interpreted
  - by default, on a method-by-method basis
    - ngen.exe supports “preJIT”

JIT Compilation



C#

```
Point pt;
pt = new Point();
pt.x = 200;
pt.y = 300;
```

dev-time compile

IL

```
// Point pt = new Point();
.locals init ([0] class Point pt)
newobj instance void Point::.ctor()
stloc.0

// pt.x = 200;
ldloc.0
ldc.i4 0xc8
stfld int32 Point::x

// pt.y = 300;
ldloc.0
ldc.i4 0x12c
stfld int32 Point::y
```

emitted by compiler into an assembly on disk
available at run-time to the CLR

run-time compile

Intel x86

```
call FD5B0AD8                     ; allocate
mov ecx, eax                      ; ecx == pt
call dword ptr ds:[003E5144h]     ; pt.ctor()
mov dword ptr [ecx+4], 0C8h       ; pt.x = 200
mov dword ptr [ecx+8], 12Ch       ; pt.y = 300
```

transient in-memory product of JIT compilation

Garbage Collection

- **Unmanaged applications require considerable memory mgmt effort**
  - Difficult to reason about
    - e.g.: different ownership models
  - Error prone & difficult to debug
    - failure to release (leak)
    - multiple release (undefined)
    - use after release (undefined)
- **The CLR's heap manager provides an efficient allocator**
  - Dynamically tuned acquisition of underlying virtual memory resources
  - Prevents/reduces fragmentation of underlying virtual memory
- **The CLR collects garbage from time to time**
  - Traversal of rooted references results in identification of garbage
  - Compaction improves locality of reference & alleviates fragmentation

You may never write Main()

- **Managed code is often hosted by a framework, for example:**
- **Desktop applications**
  - Windows Forms
  - Windows Presentation Foundation (WPF)
- **Services**
  - ASP.NET applications & web services
  - Windows Communication Foundation (WCF)
- **Other hosted application scenarios**
  - Silverlight
  - SQL 2005+ stored procedures & user-defined functions
  - PowerShell cmdlets
- **The .NET Framework must be installed for any of this to work**
  - You can ship the redistributable with your product if you like

Summary

- **The Common Language Infrastructure (CLI) spec defines**
  - an execution engine (EE/VES)
  - a type system (CTS)
  - minimal type system support requirements (CLS)
- **The Microsoft .NET Framework implements the CLI++**
  - generally referred to as “the CLR” (Common Language Runtime)
  - the CLR is hosted within each process running managed code
- **Several tools support analysis of the CLR and managed apps**
  - static analysis: ILDASM, Reflector
  - execution analysis: VS.NET with SOS debugger extension DLL

References

- Specs & .NET Framework Variations
    - □ http://link.pluralsight.com/netspecs
    - □ http://www.microsoft.com/net
    - □ http://www.microsoft.com/netmf
    - □ http://link.pluralsight.com/netcf
    - □ http://link.pluralsight.com/sscli2
- Tools
    - □ http://www.microsoft.com/whdc/devtools/debugging
    - □ http://www.red-gate.com/products/reflector
- Instructor-Led Courses
    - □ http://www.pluralsight.com/main/ilt/Courses.aspx?category=framework