Administrivia

- Assignment 6 posted
  - Due Tuesday 25 July
- Course evaluations are open to students now
  - In myWSU, click on the Manage Classes tile and find the Course Evaluations tab on the left-hand side
  - Open until Friday 28 July
- Final Exam
  - Thursday 27 July
  - 1:10 – 2:00 p.m.
  - Review on Monday

CptS 355- Programming Language Design

Implementing Programming Languages

Java Virtual Machine

Instructor: Jeremy E. Thompson

Washington State University

World Class. Face to Face.

Implementing Programming Languages

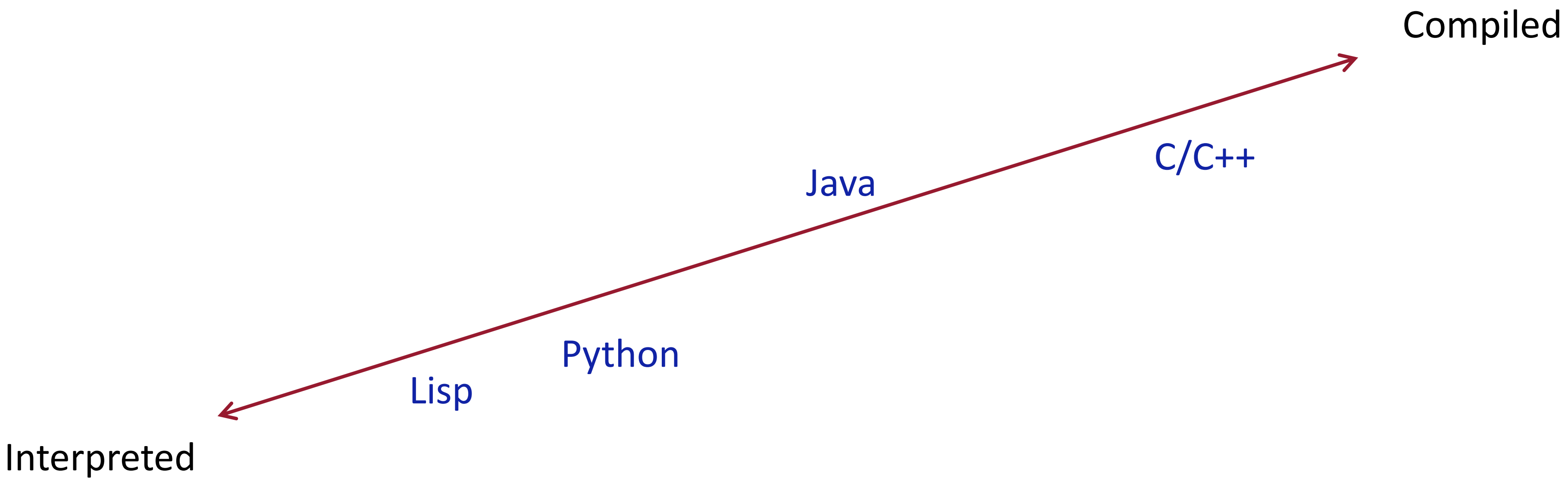
- How to implement programming models?
  - –Compiled languages:
    - Examples: C, C++, Fortran, Pascal, Haskell
  - –Interpreted languages:
    - Examples: LISP, Scheme, Python, MATLAB, Perl

Implementing Programming Languages

- Advantages of Interpreted Languages
  - Execute line by line in original source code
  - Easier to program and debug
    - Un-typed variables (*sometimes*)
    - On the fly variable creation
  - Easier to run on different architectures (portable):
    - Runs as a simulated environment that exists inside the interpreter process
  - All work done at run time
- Disadvantages of Interpreted Languages
  - Much slower to execute
  - Might be ineffective for large scale applications

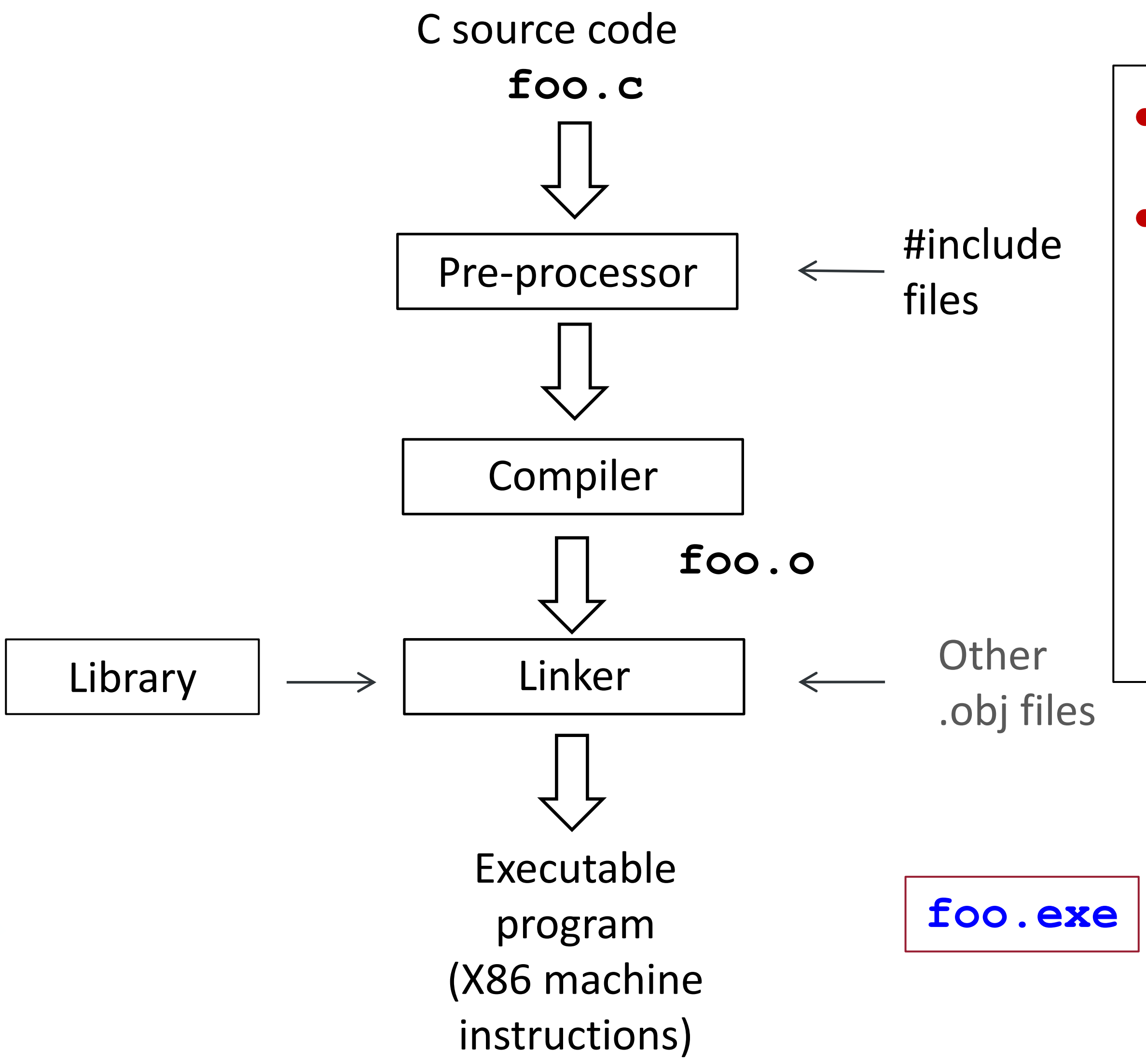
Interpreted vs. Compiled

- Work more or less done by interpreter/compiler



- Java programs are usually run by a virtual machine
  - VMs interpret an intermediate, “partly-compiled” language called bytecode

The C Programming System



C source code

foo.c

Pre-processor

#include files

Compiler

foo.o

Library

Linker

Other .obj files

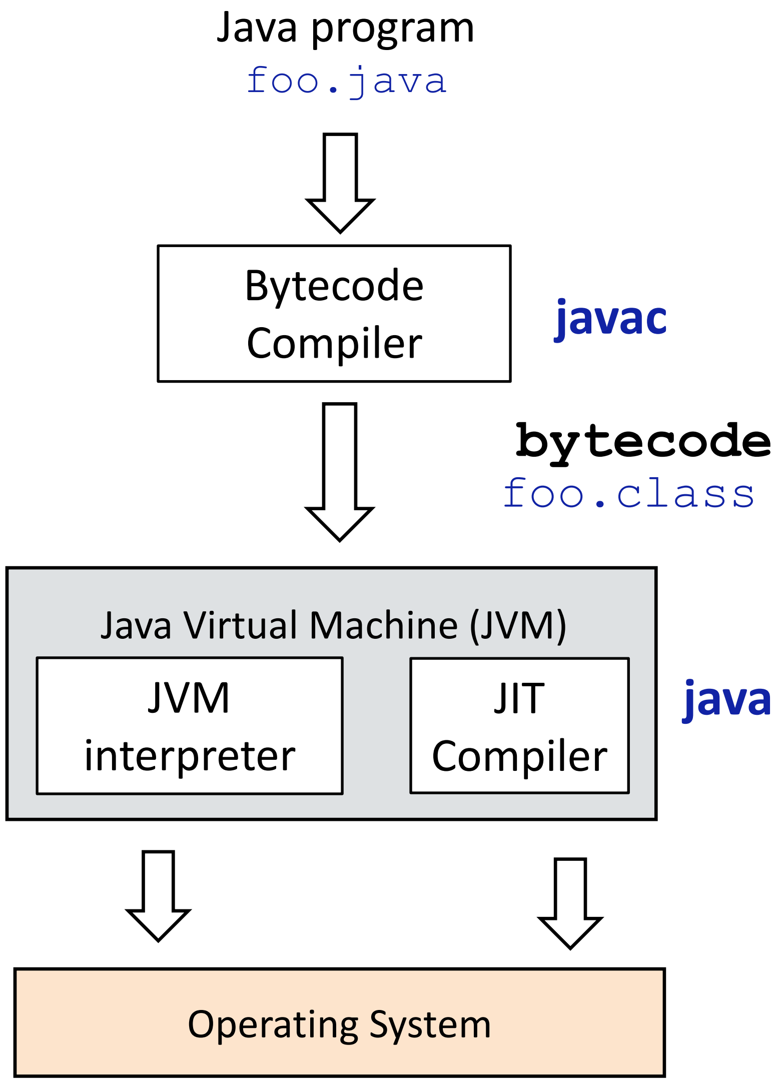
Executable program (X86 machine instructions)

foo.exe

- C is an example of a compiled language
- `gcc` is a script which hides steps
  - `gcc -c foo.c`
    - creates `foo.o`
  - `gcc -o foo foo.o`
    - links/creates `foo.exe`

Physical Machine

Virtual Machine Model



- Java Virtual Machine
  - Makes Java language machine-independent
  - Provides strong protection
  - Stack based execution model
  - There are many JVMs
    - Some interpret
    - Some compile into assembly
    - Usually *implemented* in C

Java Bytecodes

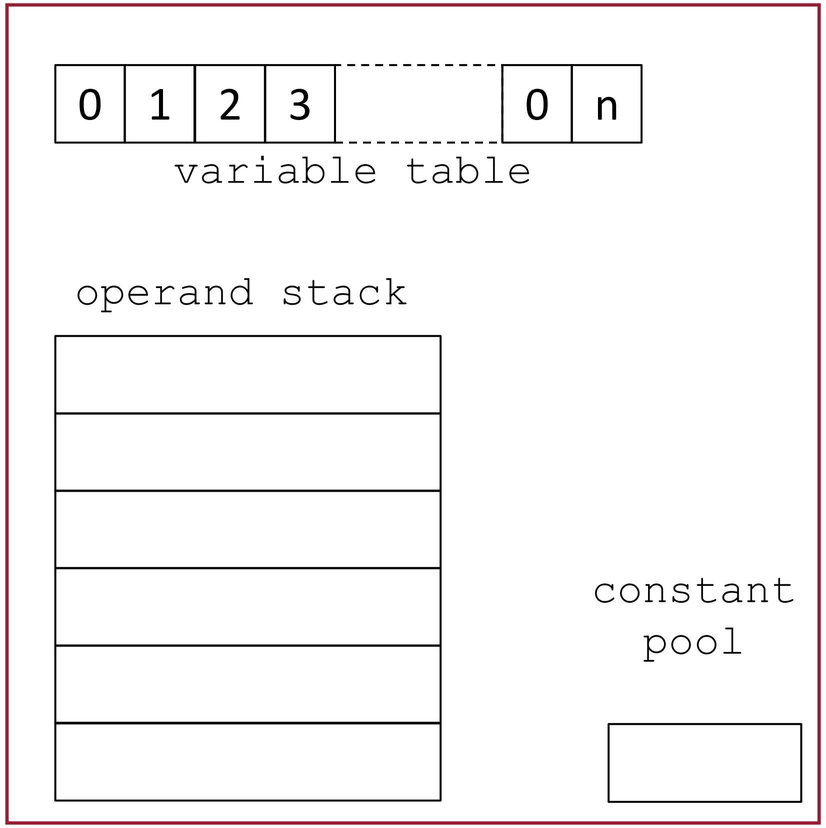
```
iload 1      //push 1st argument from table onto stack
iload 2      //push 2nd argument from table onto stack
iadd        //pop top 2 elements from stack
istore 2     //pop result and store in table
```

‘i’ stands for integer
‘a’ for reference
‘b’ for byte
‘c’ for char
‘d’ for double

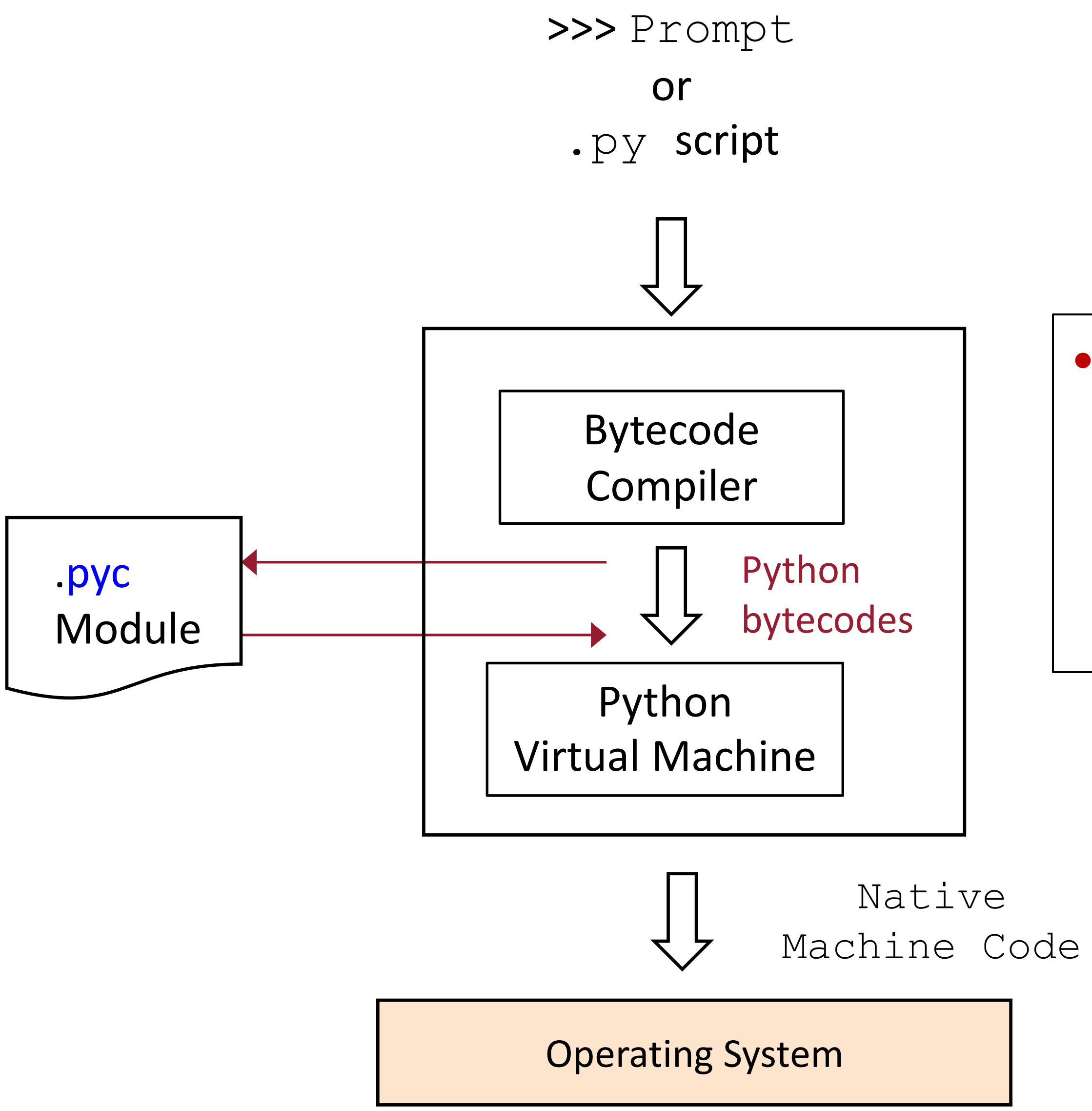
No knowledge of integer’s memory locations (each instruction is 1 byte – bytecode)

vs. machine code

```
mov 0x8001, %eax
mov 0x8002, %edx
add %edx, %eax
mov %eax, 0x8003
```

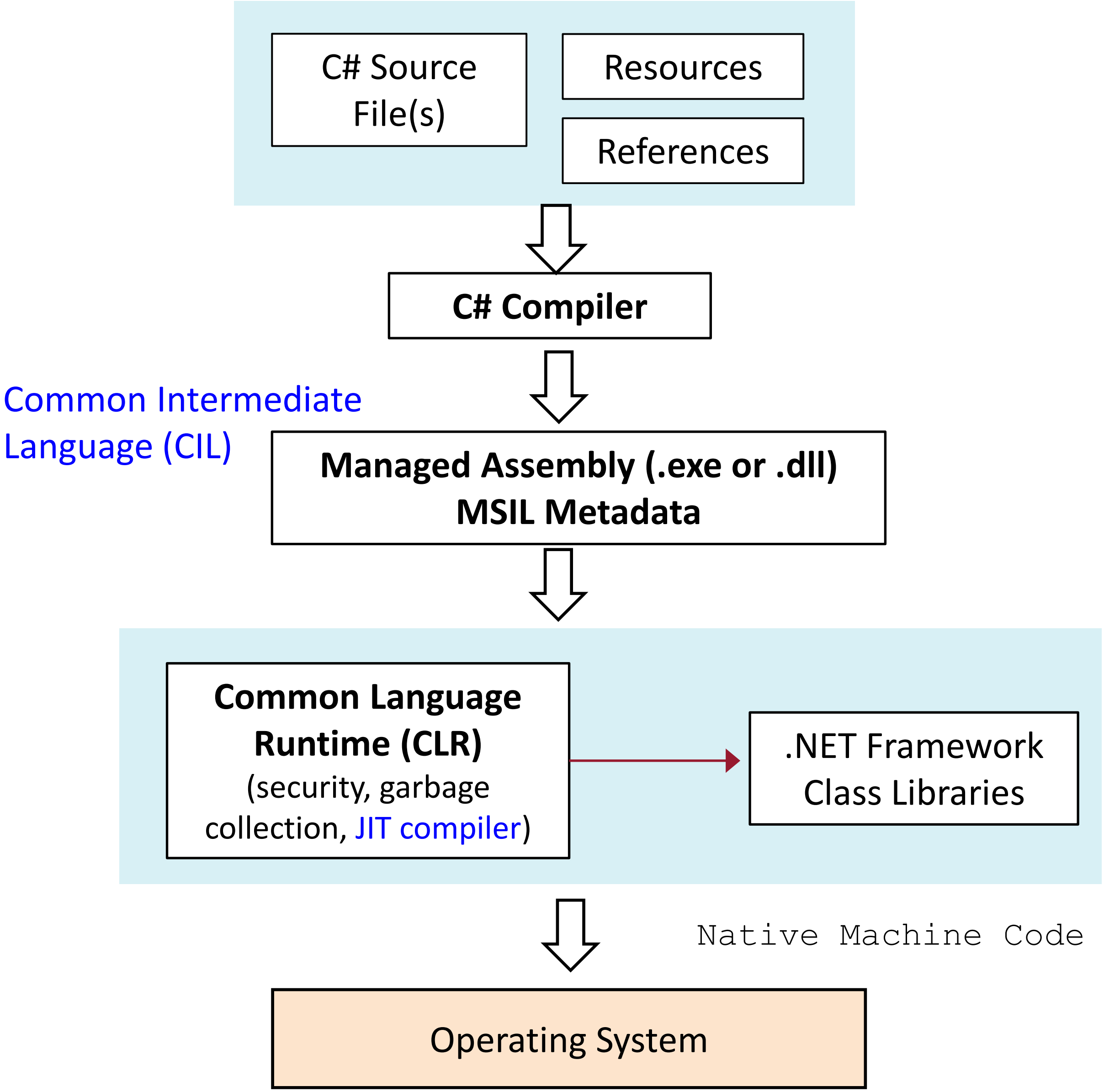


Python Interpreter



- The Python interpreter consists of two parts
  - A Python bytecode compiler
  - A virtual machine which executes Python

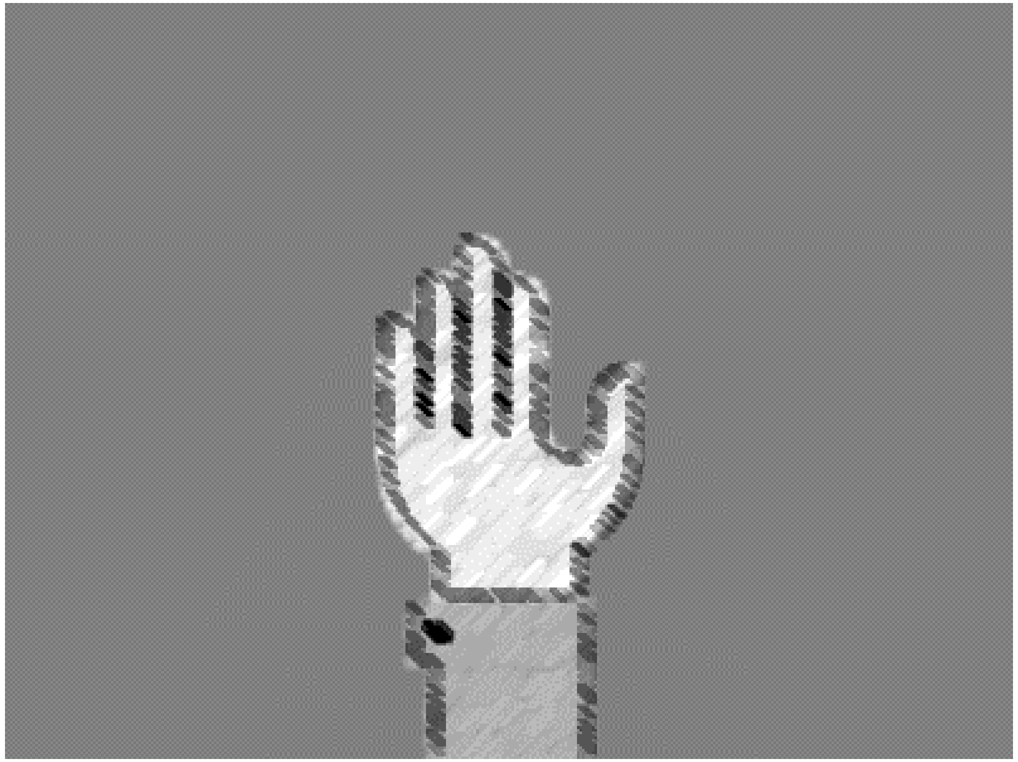
C# and .NET Framework Platform


C# Source File(s)
Resources
References
C# Compiler
Common Intermediate Language (CIL)
Managed Assembly (.exe or .dll) MSIL Metadata
Common Language Runtime (CLR) (security, garbage collection, JIT compiler)
.NET Framework Class Libraries
Native Machine Code
Operating System

Language interoperability is a key feature of the .NET Framework. Because the IL code produced by the C# compiler conforms to the Common Type Specification (CTS), IL code generated from C# can interact with code from the .NET versions of Visual Basic, Visual C++, or any of more than 20 other CTS-compliant languages.

A single assembly may contain multiple modules written in *different* .NET languages, and the types can reference each other just as if they were written in the *same* language

Questions?



End of testable material