# Common Language Intermediate Language

* <https://www.codeproject.com/Articles/362076/Understanding-Common-Intermediate-Language-CIL>
* .maxstack directive by default is 8, says the maximum items that would be present on the stack
  + The maximum number of variables & parameters is usually the max stack.
* This code converts to this IL
* Ldc.\* load local variable x on the stack
* .locals : declare variables
* Ldloc.x : loads local variable x on stack
* Stloc.x: pops & stores in to variable
* Ldarg.x: loads arg x.
  + ldarg.0 Any instance level method, including ctor, behind scenes takes ‘this’ as first parameter. So ldarg.0 is this.

## OP codes

* <https://en.wikipedia.org/wiki/List_of_CIL_instructions>
* <https://msdn.microsoft.com/en-us/library/system.reflection.emit.opcodes_fields(v=vs.110).aspx>
* All op codes starting with ld are used for loading. i.e to push a value from memory to stack
* All op codes starting with st are used for storing. i.e pop from stack to memory

# Mono.Cecil

* [www.mono-project.com/Cecil](http://www.mono-project.com/Cecil) or <http://www.mono-project.com/docs/tools+libraries/libraries/Mono.Cecil/>
* <https://github.com/jbevain/cecil/wiki>
* Create new classes, interfaces, methods
* Write CIL instructions

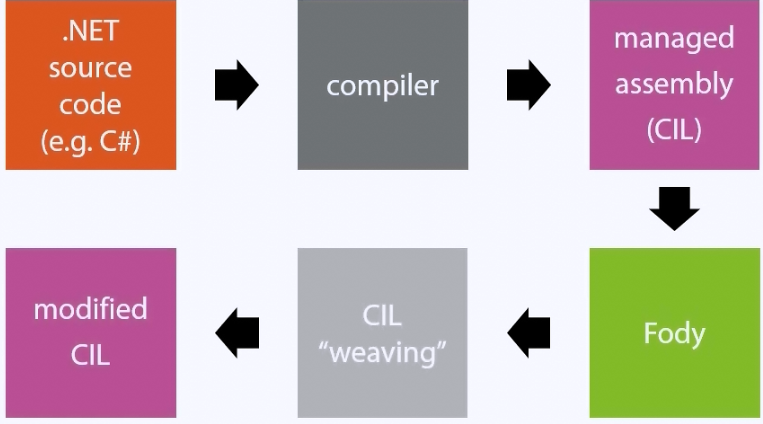
## Importing

* Used when you want to inject code that calls into other assembles

## References

* To get a type reference: ModuleDefinition.TypeSystem.String
* Adding a property : <https://stackoverflow.com/questions/7195260/using-mono-cecil-to-add-an-indexer-to-a-typedefinition>
* Adding a custom attribute: <https://stackoverflow.com/questions/10281104/how-do-i-add-a-custom-attribute-without-a-default-constructor-using-mono-cecil?noredirect=1&lq=1>

# Fody

* <https://github.com/fody>
* <https://github.com/Fody/Fody/wiki>
* 
* download [Basic fody addin](https://github.com/Fody/BasicFodyAddin)
  + see documentation
  + Test project helps to verify the injected code
  + Usage of attributes

## Addin

* <https://intellitect.com/creating-fody-addin/>
* The assembly must be sufficed with .Fody
* Install Fody nugget package to both your assembly and weaver project. No need of FodyWeavers.xml in later
* Add reference to Mono.Cecil.\*.dlls from package/fody directory to your weaver project
* Dependencies other than Mono should be combined using [ILMerge](http://research.microsoft.com/en-us/people/mbarnett/ilmerge.aspx) & the /internalize flag
* Must contain class named ModuleWeaver; Namespace does not matter.
* Build only your weaver project. Copy the dll to packages/YourWeaver. Fody looks here for the

### IL Merge

* Any dependencies should be merged
* <https://github.com/Microsoft/ILMerge/blob/master/ilmerge-manual.md>
* <http://blog.risingperfection.com/2013/03/how-to-use-ilmerge-and-how-to-build-portable-applications.html>

## In solution weaving:

* <https://github.com/Fody/Fody/wiki/InSolutionWeaving>
* <https://github.com/Fody/Fody/tree/master/Integration>
* Fody will look for binary Weavers.dll inside Weavers\bin
* Project name and assembly name should be Weavers ans must be in a directory called Weavers
* Change the build over, such that Weaver project is build before the projects using it.
  + Do not add a project reference.
* If weaver class is not named ModuleWeaver, FodyWeavers.xml should use the weaver class name

## ModuleWeaver class

* Public, not abstract, empty ctor
* ModuleDefinition property: Cecil representation of assembly being built
* Execute method: called as part of a build after properties have been set.