

Assembly in C#

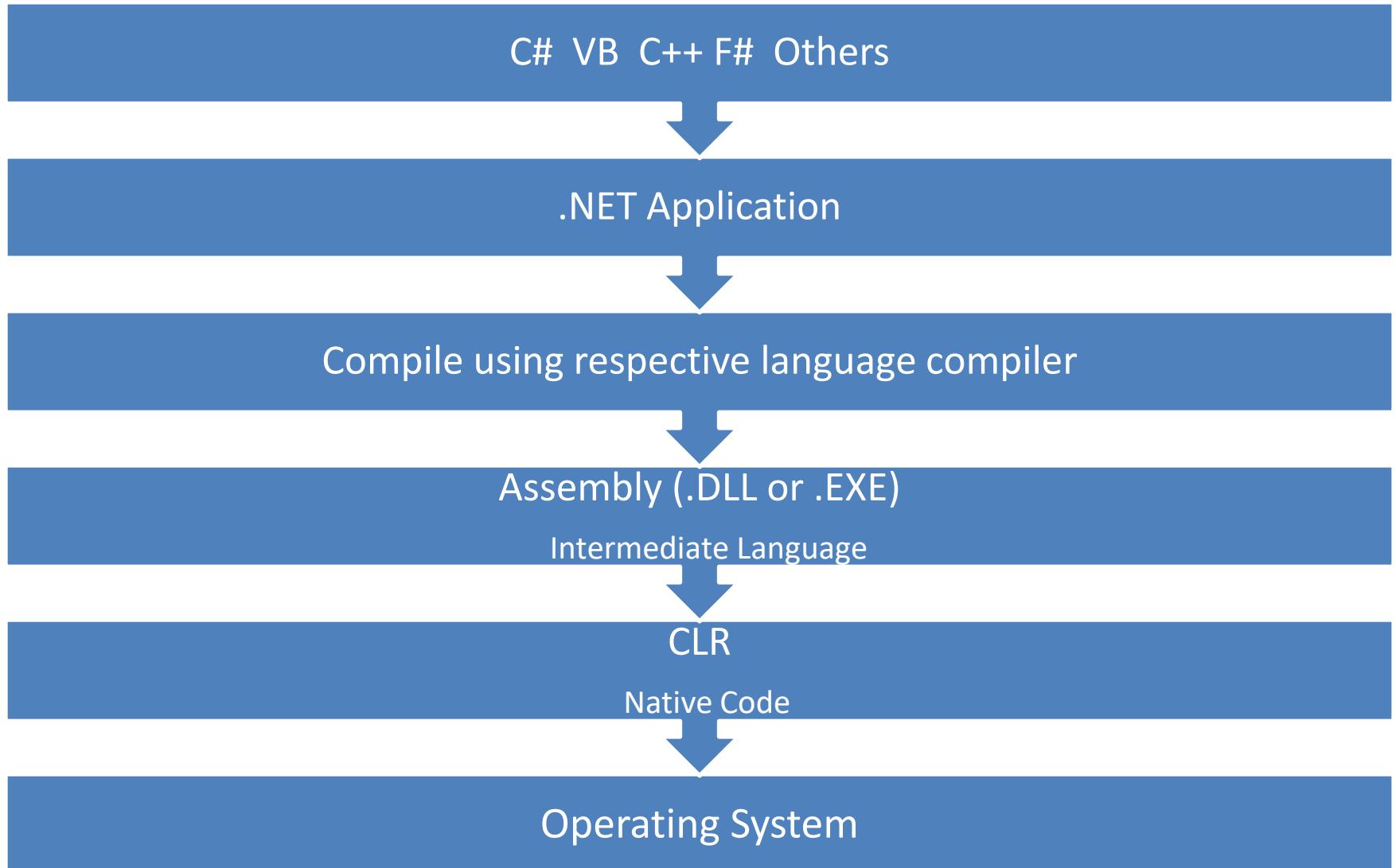
Prepared for Vth semester DDU-CE students
2025-26 WAD

Apurva A Mehta

TIFR

- Tata Institute of Fundamental Research
 - A National Centre of the Government of India, under the umbrella of the Department of Atomic Energy
 - A deemed University awarding degrees for master's and doctoral programs.
 - Carry out basic research in physics, chemistry, biology, mathematics, **computer science** and science education.

Program execution



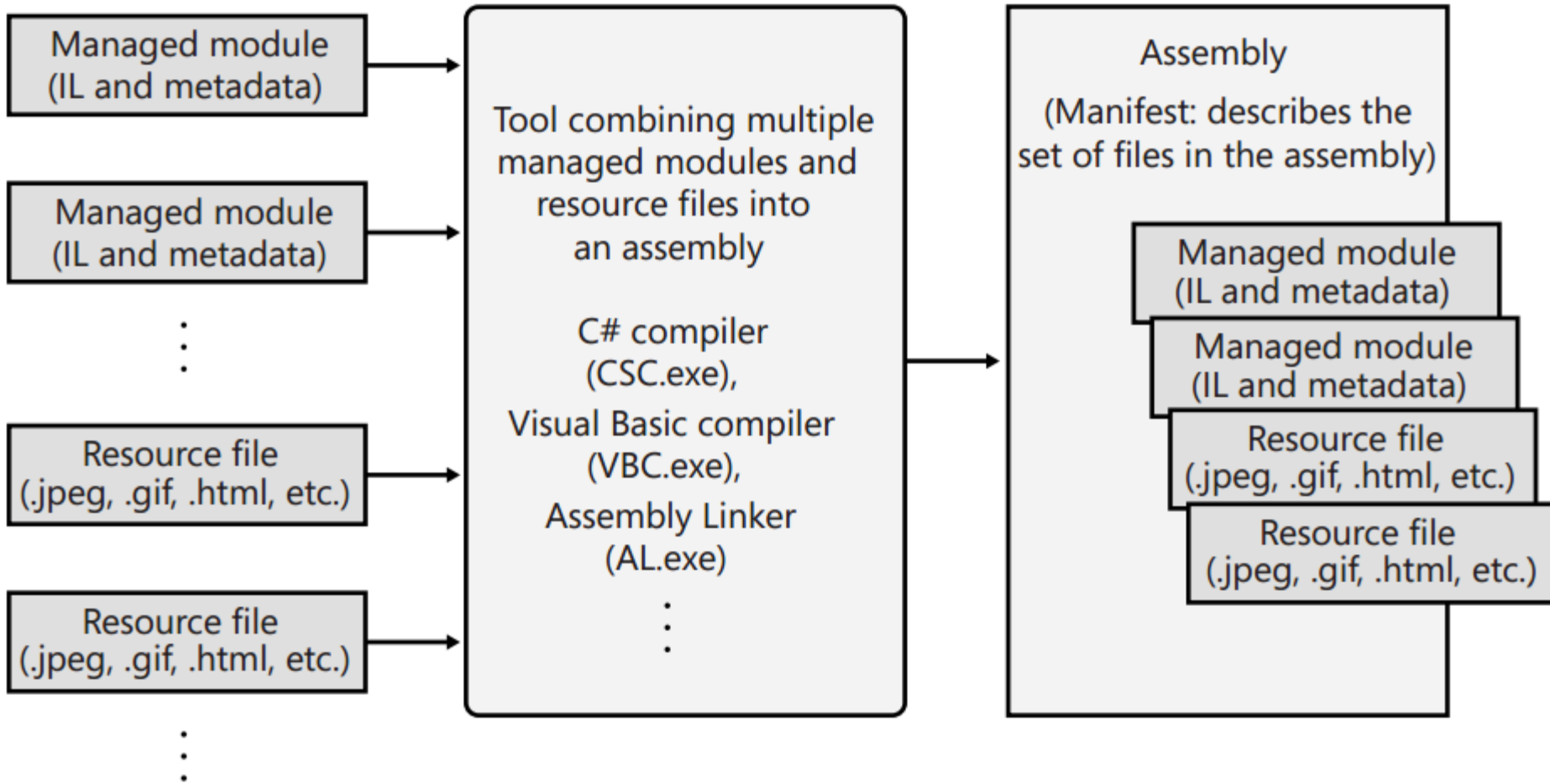
Introduction

- Assembly is the core part of the runtime.
 - Physical collection of classes.
- It is collection of all information required by the runtime to execute your application.
- IL + Metadata
- An assembly can be a **DLL file** or a **Portable Executable (EXE) file**

Safe vs Unsafe Code

- By default, Microsoft's C# compiler produces safe code. Safe code is code that is verifiably safe.
- Unsafe code is allowed to work directly with memory addresses and can manipulate bytes at these addresses.
- PEVerify.exe

Complete Definition



Purpose of Assembly

- Using assemblies allows you to semantically group functional units into a single file for purposes of deployment, versioning, and maintenance.

Manifest Data

- PE32(+) file contains a block of data called the manifest
- Manifest: Another set of metadata
- Manifest describes the set of files in assembly

Content of Manifest

- Assembly name
- Version number
- Culture
- Strong name information
- List of all files in the assemblies
- Type reference information
- Information on referenced assemblies

Assembly name	A text string specifying the assembly's name.
Version number	A major and minor version number, and a revision and build number. The common language runtime uses these numbers to enforce version policy.
Culture	Information on the culture or language the assembly supports. This information should be used only to designate an assembly as a satellite assembly containing culture- or language-specific information. (An assembly with culture information is automatically assumed to be a satellite assembly.)
Strong name information	The public key from the publisher if the assembly has been given a strong name.
List of all files in the assembly	A hash of each file contained in the assembly and a file name. Note that all files that make up the assembly must be in the same directory as the file containing the assembly manifest.
Type reference information	Information used by the runtime to map a type reference to the file that contains its declaration and implementation. This is used for types that are exported from the assembly.
Information on referenced assemblies	A list of other assemblies that are statically referenced by the assembly. Each reference includes the dependent assembly's name, assembly metadata (version, culture, operating system, and so on), and public key, if the assembly is strong named.

Single file assembly

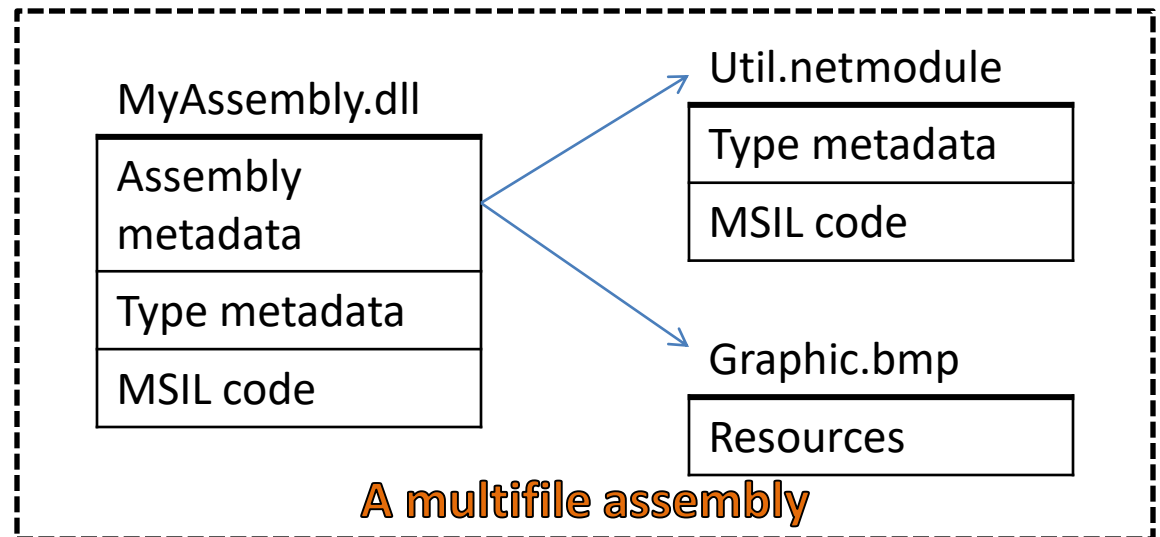
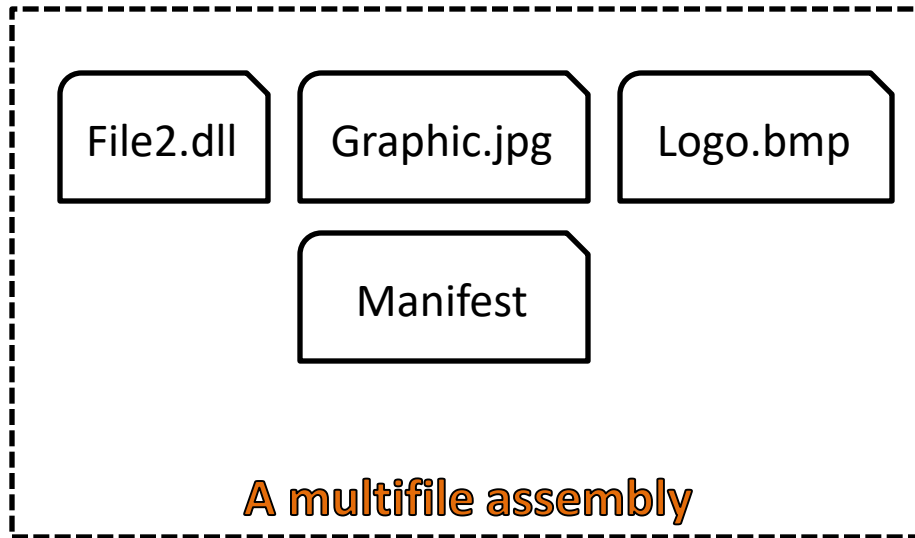
- If you were to compile a stand-alone application or DLL, the manifest would be incorporated into the resulting PE. This is known as a *single-file assembly*.

File1.dll
Manifest

File2.dll
Assembly metadata
MSIL code
Resources

Multifile Assembly

- A multifile assembly can also be generated, with the manifest existing as either a **standalone** entity within the assembly or as an **attachment** to one of the modules within the assembly.



Benefits of Assembly

- Assemblies afford the developer numerous benefits, including packaging, deployment, and versioning.
- Increased performance.
- Better code management and encapsulation.
- Introduces the n-tire concepts and business logic.

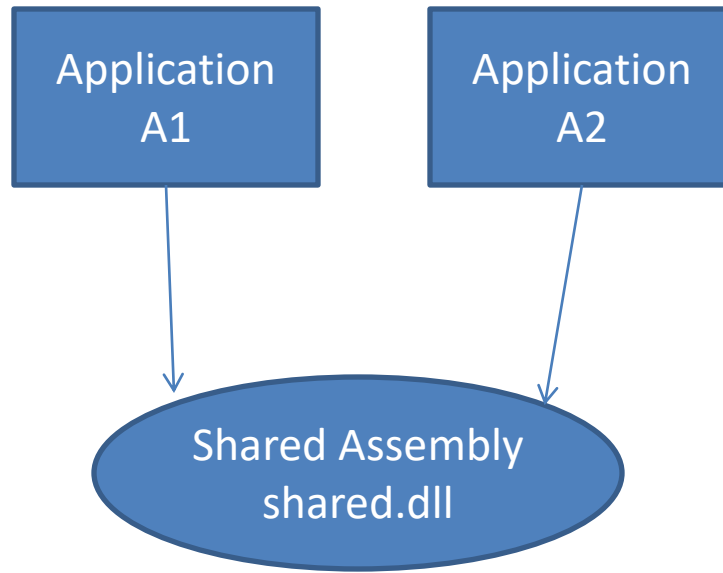
Deployment methods

- Private Assembly
 - An Assembly copied into same folder as referencing program.
- Shared Assembly
 - An Assembly deployed globally which can be consumed by any program.
- Assemblies can also be deployed worldwide via 3rd party package managers such as Nuget (Integrated with Visual Studio)

DLL Hell

- If the assembly is not signed with private-public key pair, the assembly is weak named and not guaranteed to be unique, and may cause DLL hell
- Strong named assemblies are guaranteed to be unique and solves DLL hell. You cannot install an assembly into GAC unless, the assembly is strongly named

DLL-Hell Problem



Continue...

- I have 2 applications. A1 and A2 installed on my machine
- Both of these applications use shared assembly shared.dll
- Now, I have a latest version of application A2 available on the internet
- I download the latest version of A2 and install it on my machine
- This new installation has over written shared.dll, which is also used by application A1
- Application A2 works fine, but A1 fails to work, because the newly installed shared.dll is not backward compatible

Continue...

- So, DLL HELL is a problem where one application will install a new version of the shared component that is not backward compatible with the version already on the machine causing all the other existing applications that rely on the shared component to break
- With .NET strong named assemblies we don't have DLL HELL problem any more

Justify: CLR is introduced for portability in .NET framework.

What is the purpose of using assembly?

Differentiate: Safe code and Unsafe code

Differentiate: Single file assemblies and Multifile assemblies.

Differentiate: Private assemblies and Shared assemblies.

Drink water, please!

- About 75% of the brain is made up of water. This means that dehydration, even in small amounts, can have a negative effect on the brain functions.
- Our body is composed of about 60% water.
- Blood is more than 90% water, and blood carries oxygen to different parts of the body.

Versioning

- Versioning makes sure that even if new version of DLL is installed, still old application can use old DLL.

The MSCLib Assembly

- **MSCLib.dll** is a special file in that it contains all the core types
 - Byte, Char, String, Int32, and many more.
 - In fact, these types are so frequently used that the C# compiler automatically references the MSCLib.dll assembly.

```
1 using System;
2
3 namespace ConsoleApplication1
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9             Console.WriteLine("DDU");
10            Console.ReadLine();
11        }
12    }
13 }
14
```


Cont...

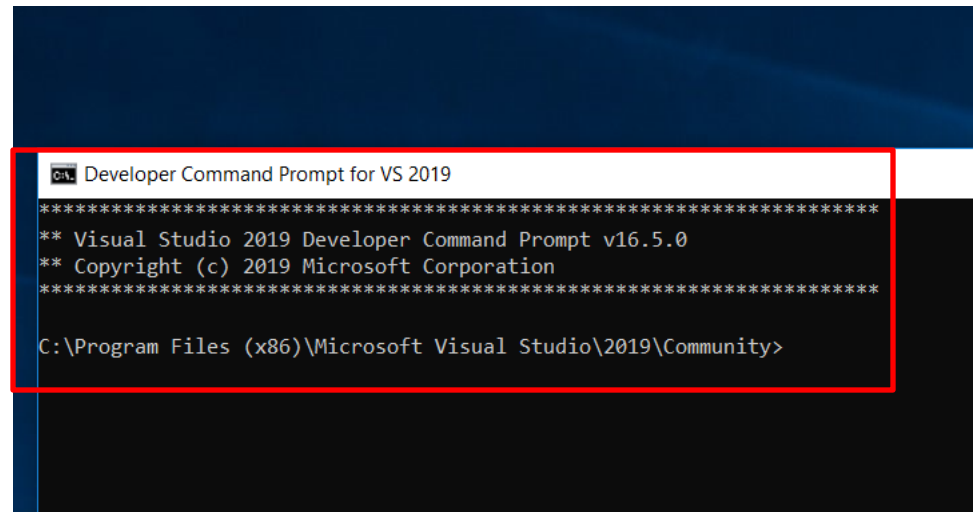
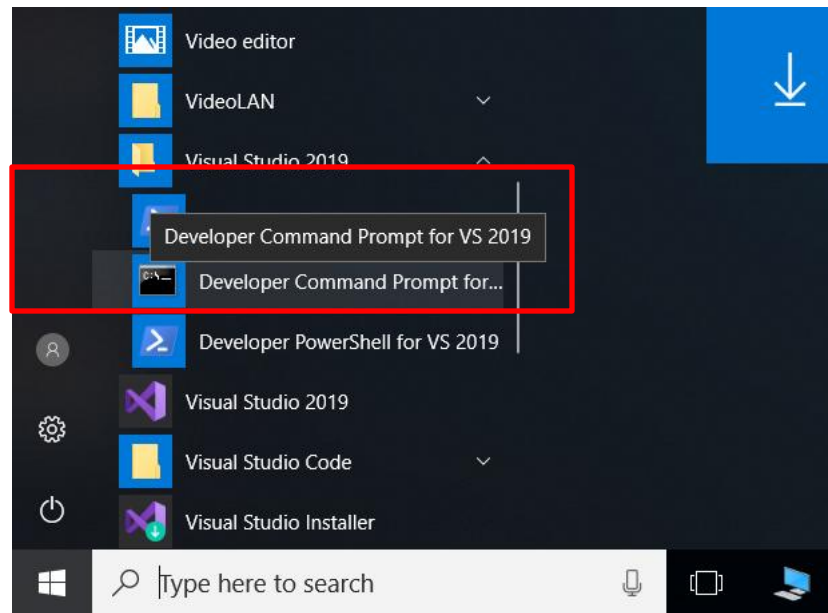
csc.exe /out:Program.exe /t:exe /r:mscorlib.dll Program.cs

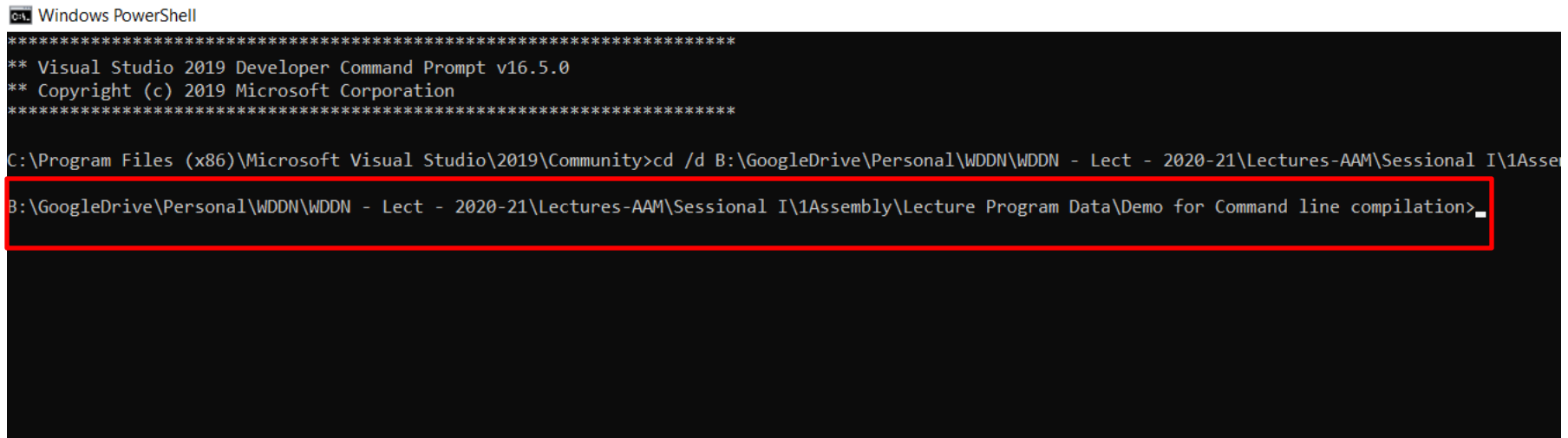
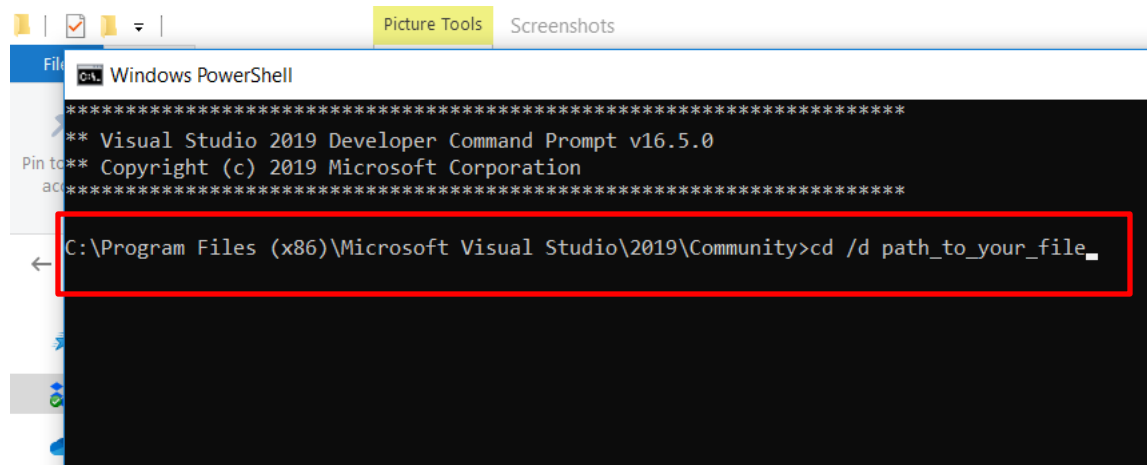
csc.exe /out:Program.exe /t:exe Program.cs

csc.exe Program.cs

/nostdlibswitch

csc.exe /out:Program.exe /t:exe /nostdlib Program.cs





mpilation

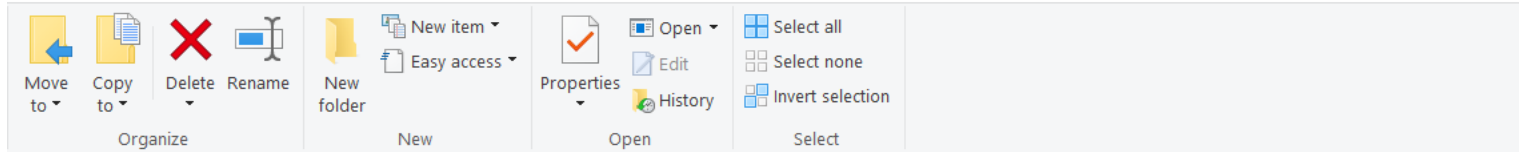
The screenshot shows a Windows File Explorer window with the following path: > GoogleDrive > Personal > WDDN > WDDN - Lect - 2020-21 > Lectures-AAM > Sessional I > 1Assembly > Lecture Program Data > Demo for Command line. The file list shows two files:

Name	Date modified	Type	Size
Program.cs	13-07-2018 11:02 ...	Visual C# Source F...	1 KB
Program.exe	02-07-2020 03:32 ...	Application	4 KB

Below the file list, a Windows PowerShell window is open, showing the command prompt. The command 'csc Program.cs' is entered and highlighted with a red box and a yellow number '1'. The output shows the Microsoft (R) Visual C# Compiler version 3.5.0-beta4-20153-05 (20b9af91) Copyright (C) Microsoft Corporation. All rights reserved.

Application Tools Demo for Command line compilation

Manage



;) > GoogleDrive > Personal > WDDN > WDDN - Lect - 2020-21 > Lectures-AAM > Sessional I > 1Assembly > Lecture Program Data > Demo for Command line

Name	Date modified	Type	Size
Program.cs	13-07-2018 11:02	Visual C# Source F	1 KB
Program.exe	02-07-2020 03:32 ...	Application	4 KB

B:\GoogleDrive\Personal\WDDN\WDDN - Lect - 2020-21\Lectures-AAM\Sessional I\1Assembly\Lecture Program Data\Demo for Co... - □ ×

DDU

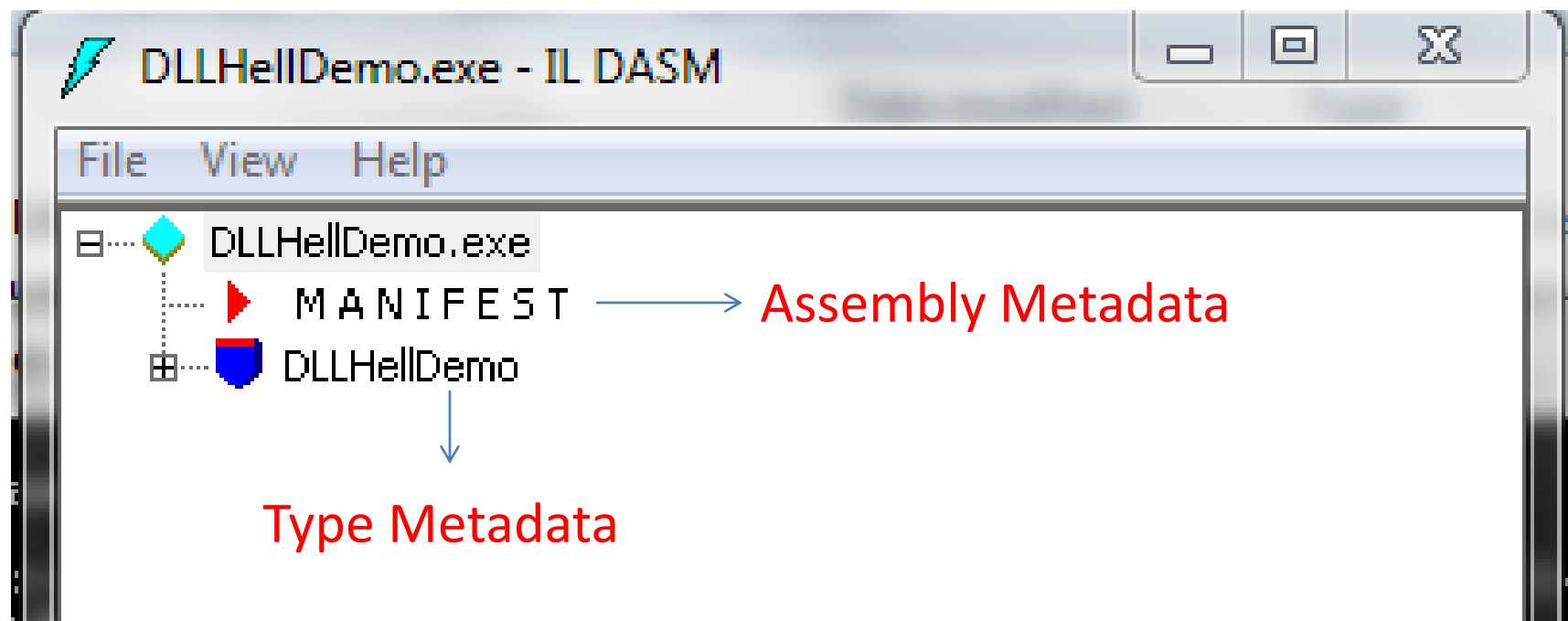
Cont...

- Windows supports three types of applications.
 - To build a console user interface (CUI) application, specify the **/t:exe** switch
 - To build a graphical user interface (GUI) application, specify the **/t:winexe** switch
 - To build a Windows Store app, specify the **/t:appcontainerexe** switch.

- Which component does convert IL to Native code?
- Which code is OS dependent?
- Which code is OS independent?
- Which components are required to be installed in order to execute any module containing managed code and managed data?

Manifest vs Metadata

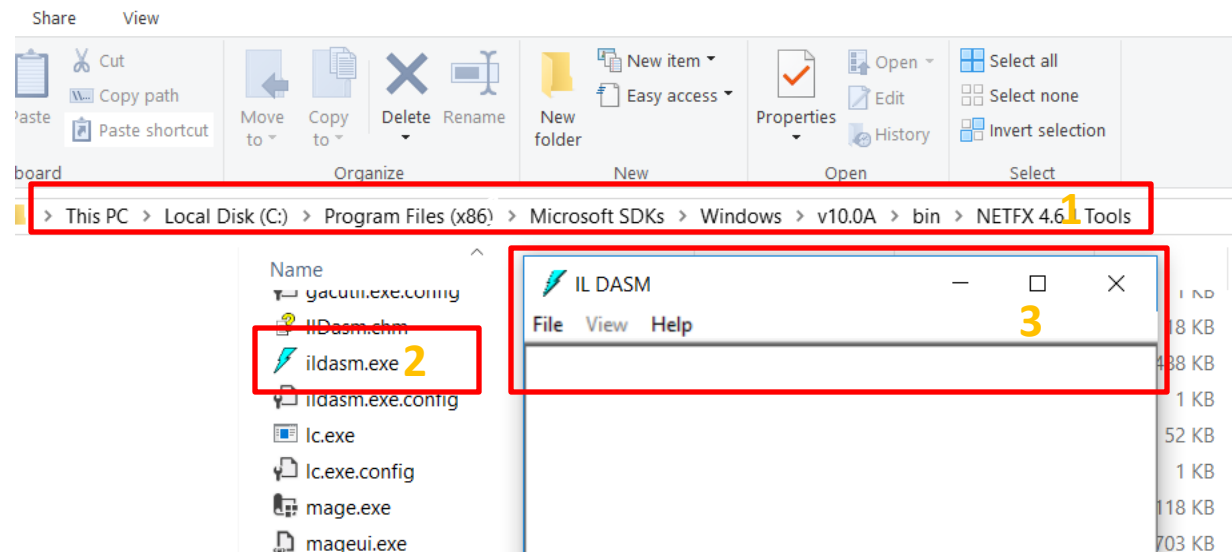
- Manifest maintains the information about the assemblies like version, name locale and an optional strong name that uniquely identifying the assembly. This manifest information is used by the CLR.
- Metadata means data about the data. Metadata yields the types available in that assembly, viz. classes, interfaces, enums, structs, etc., and their containing namespaces etc.

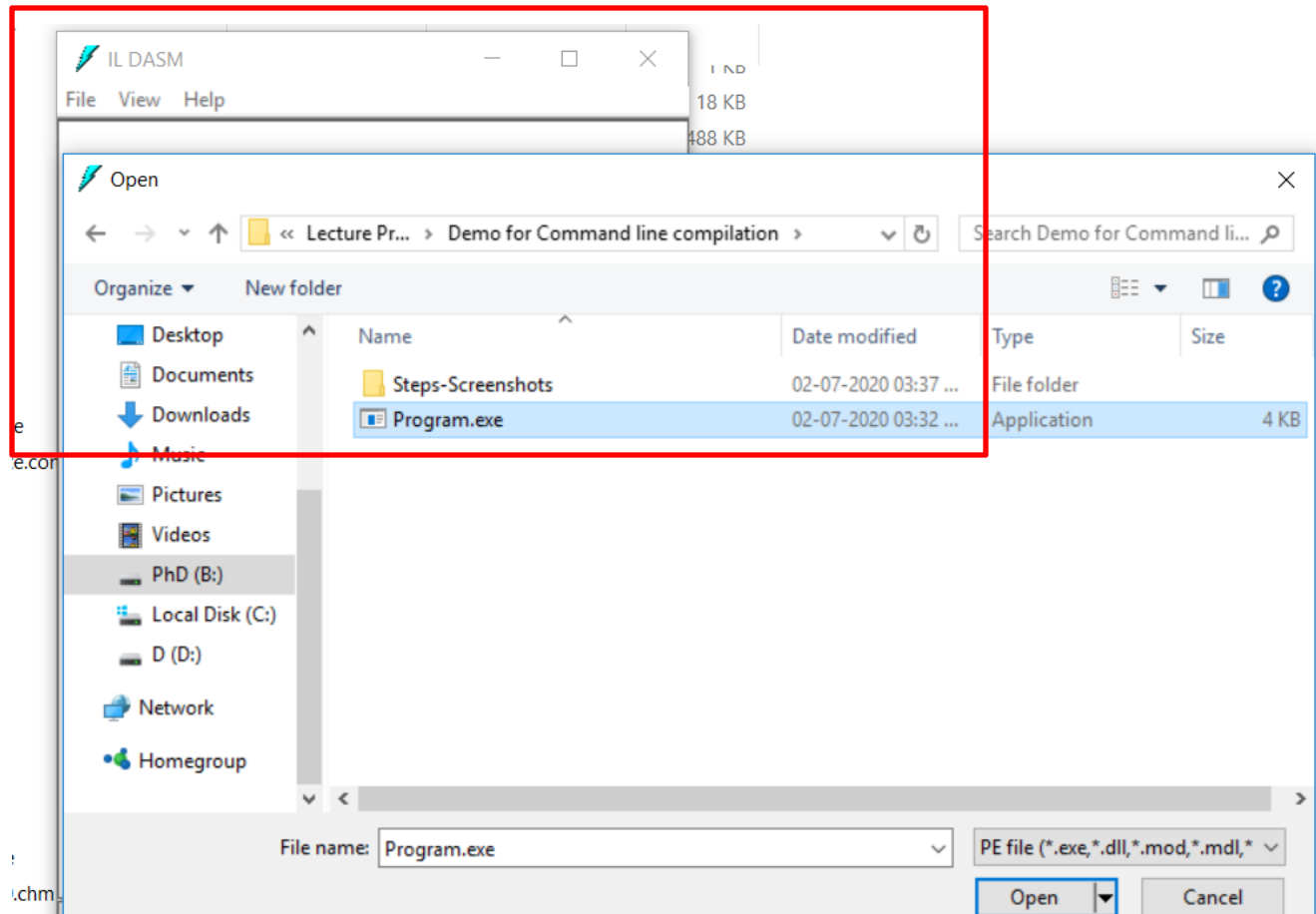


ILDASM.EXE

- We use ILDASM (intermediate language disassembler) to peek at the assembly manifest and IL.
- You can also use this tool to export manifest and IL to a text file
- MSIL + namespace + types + ...

ILFX 4.6.1 Tools





rd Organize New Open Select

> This PC > Local Disk (C:) > Program Files (x86) > Microsoft SDKs > Windows > v10.0A > bin > NETFX 4.6.1 Tools >

Name

- gacutil.exe.config
- ILDasm.chm
- ildasm.exe
- ildasm.exe.config
- lc.exe
- lc.exe.config
- mage.exe
- mageui.exe
- mgmtclassgen.exe
- mpgo.exe
- MSBuildTaskHost.exe
- MSBuildTaskHost.exe.config
- PEVerify.exe
- PEVerify.exe.config
- ResGen.exe
- SecAnnotate.exe
- sgen.exe
- sn.exe
- sn.exe.config
- SqlMetal.exe
- SqlMetal.exe.config
- StoreAdm.exe
- SvcConfigEditor.exe
- SvcConfigEditor_4.0.chm
- SvcTraceViewer.chm
- SvcTraceViewer.exe
- SvcUtil.exe
- TlbExp.exe

B:\GoogleDrive\Personal\WDDN\WDDN - Lect - 2020-21\Lectures-AAM\MANIFEST

MANIFEST

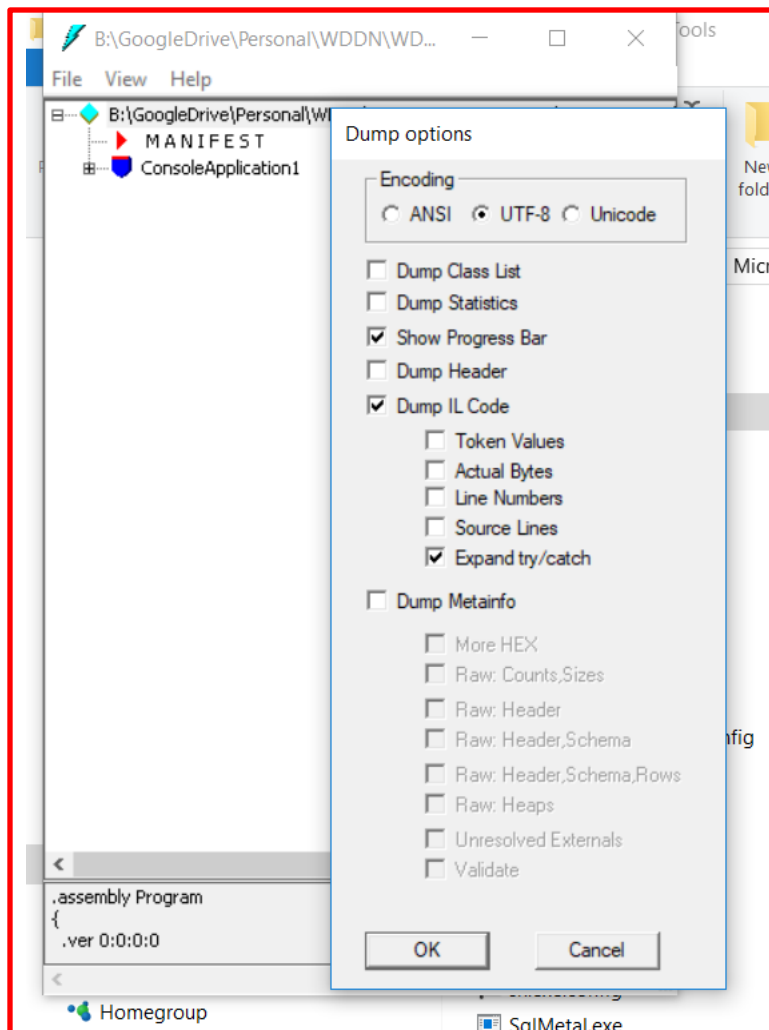
```
// Metadata version: v4.0.30319
.assembly extern mscorlib
{
    .publickeytoken = (B7 7A 5C 56 19 34 E0 89 ) // .z
    .ver 4:0:0:0
}
.assembly Program
{
    .custom instance void [mscorlib]System.Runtime.CompilerServices.CompilationS
    .custom instance void [mscorlib]System.Runtime.CompilerServices.RuntimeCom

    // --- The following custom attribute is added automatically, do not uncom
    // .custom instance void [mscorlib]System.Diagnostics.DebuggableAttribute

    .hash algorithm 0x00008004
    .ver 0:0:0:0
}
.module Program.exe
// MVID: {F8EFC768-83C4-4EBE-9EE0-9E68D021E5F0}
.imagebase 0x00400000
.file alignment 0x00000200
.stackreserve 0x00100000
.subsystem 0x0003 // WINDOWS_CUI
.corflags 0x00000001 // ILONLY
// Image base: 0x008E0000
```

05-11-2015 08:36 ... Application 63 KB

re to search



New item	Properties	Open	Select all
Easy access	Edit	History	Select none
			Invert selection
New	Open	Select	

Microsoft SDKs > Windows > v10.0A > bin > NETFX 4.6.1 Tools >

Date modified	Type	Size
05-11-2015 08:17 ...	XML Configuration...	1 KB
01-10-2015 02:15 ...	Compiled HTML H...	18 KB
05-11-2015 08:36 ...	Application	488 KB
05-11-2015 06:17 ...	XML Configuration...	1 KB
05-11-2015 08:36 ...	Application	52 KB
01-10-2015 02:21 ...	XML Configuration...	1 KB
05-11-2015 08:36 ...	Application	118 KB
05-11-2015 08:36 ...	Application	703 KB
05-11-2015 08:36 ...	Application	48 KB
05-11-2015 08:36 ...	Application	238 KB
05-11-2015 08:36 ...	Application	163 KB
22-10-2015 10:04 ...	XML Configuration...	2 KB
05-11-2015 08:36 ...	Application	215 KB
05-11-2015 06:17 ...	XML Configuration...	1 KB
05-11-2015 08:36 ...	Application	90 KB
05-11-2015 08:36 ...	Application	1,180 KB
05-11-2015 08:36 ...	Application	67 KB
05-11-2015 10:25 ...	Application	266 KB
05-11-2015 06:17 ...	XML Configuration...	1 KB
05-11-2015 08:36 ...	Application	285 KB

B:\GoogleDrive\Personal\WDDN\WD...
File View Help

Save As

« Lecture Pr... » Demo for Command line compilation »

Search Demo for Command li...

Organize New folder

This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
PhD (B:) **Local Disk (C:)**

Name	Date modified	Type	Size
Steps-Screenshots	02-07-2020 03:37 ...	File folder	

File name: Program

Save as type: IL file (*.il)

Hide Folders

Save Cancel

Paste shortcut
board
Organize
New
Open
Select
History
Invert selection

This PC > PhD (B:) > GoogleDrive > Personal > WDDN > WDDN - Lect - 2020-21 > Lectures-AAM > Sessional I > 1Assembly > Lecture Program Data > Demo for Command line compilation >

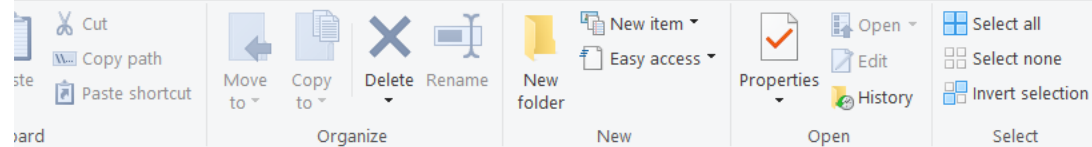
Name	Date modified	Type	Size
Steps-Screenshots	02-07-2020 03:37 ...	File folder	
Program.cs	13-07-2018 11:02 ...	Visual C# Source F...	1 KB
Program.exe	02-07-2020 03:32 ...	Application	4 KB
Program.il	02-07-2020 04:53 ...	IL File	3 KB
Program.res	02-07-2020 04:53 ...	Compiled Resourc...	2 KB

ILASM.EXE

- We use ILASM.EXE to reconstruct an assembly from a text file that contains manifest and IL

for Command line compilation

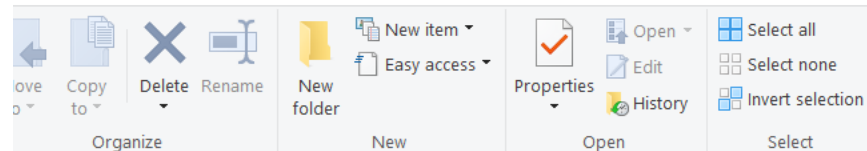
Share View



> This PC > PhD (B:) > GoogleDrive > Personal > WDDN > WDDN - Lect - 2020-21 > Lectures-AAM > Sessional I > 1Assembly > Lecture Program Data > Demo for Command line compilation >

Name	Date modified	Type	Size
Steps-Screenshots	02-07-2020 03:37 ...	File folder	
Program.cs	13-07-2018 11:02 ...	Visual C# Source F...	1 KB
Program.il	02-07-2020 04:53 ...	IL File	3 KB
Program.res	02-07-2020 04:53 ...	Compiled Resourc...	2 KB

ilation



> GoogleDrive > Personal > WDDN > WDDN - Lect - 2020-21 > Lectures-AAM > Sessional I > 1Assembly > Lecture Program Data > Demo for Command line compilat

Name	Date modified	Type	Size
Steps-Screenshots	02-07-2020 05:08 ...	File folder	
Program.cs	13-07-2018 11:02 ...	Visual C# Source F...	1 KB
Program.il	02-07-2020 04:53 ...	IL File	3 KB
Program.res	02-07-2020 04:53 ...	Compiled Resourc...	2 KB

```

C:\ Windows PowerShell

B:\GoogleDrive\Personal\WDDN\WDDN - Lect - 2020-21\Lectures-AAM\Sessional I\1Assembly\Lecture Program Data\Demo for Comm
and line compilation>ilasm Program.il
  
```

d line compilation

ath
ortcut

Move to

Copy to

Delete

Rename

Organize

New folder

New item

Easy access

New

Properties

Open

Edit

History

Open

Select all

Select none

Invert selection

Select

PhD (B:) > GoogleDrive > Personal > WDDN > WDDN - Lect - 2020-21 > Lectures-AAM > Sessional I > 1Assembly > Lecture Program Data > Demo for Command line compilation >

Name	Date modified	Type	Size
Steps-Screenshots	02-07-2020 03:37 ...	File folder	
Program.cs	13-07-2018 11:02 ...	Visual C# Source F...	1 KB
Program.exe	02-07-2020 05:06 ...	Application	2 KB
Program.il	02-07-2020 04:53 ...	IL File	3 KB
Program.res	02-07-2020 04:53 ...	Compiled Resourc...	2 KB

Multifile Assemblies

- With command line compilers, you can split an assembly into multiple parts - where a single assembly's Manifest contains the information required to find information that's part of the assembly, but stored in a separate file.
- e.g., you can keep a resource image (ie: a .bmp) that is a large resource in its own file, so that it isn't necessary to load it just to open the assembly.

Reasons to use MF Assemblies

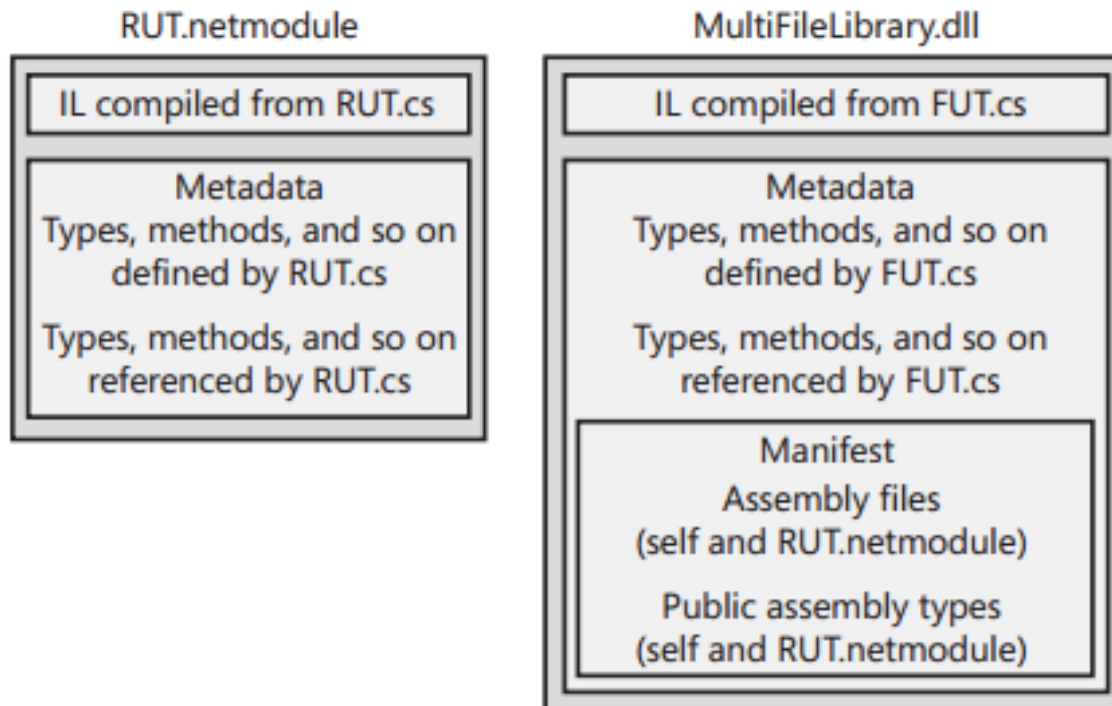
- You can partition your types among separate files, allowing for files to be incrementally downloaded
- You can add resource or data files to your assembly.
- You can create assemblies consisting of types implemented in different programming languages.

Creating MF Assemblies

- There are many ways to add a module to an assembly.
 - If you're using the C# compiler to build a PE file with a manifest, you can use the **/addmodule** switch.

Example

- `csc /t:module RUT.cs`
- `csc /out:MultiFileLibrary.dll /t:library /addmodule:RUT.netmodule FUT.cs`



Continue...

- Any client code that consumes the MultiFileLibrary.dll assembly's types must be built using the `/r[reference]:MultiFileLibrary.dll` compiler switch.
- If you were to delete the RUT.netmodule file, the C# compiler would produce the error.

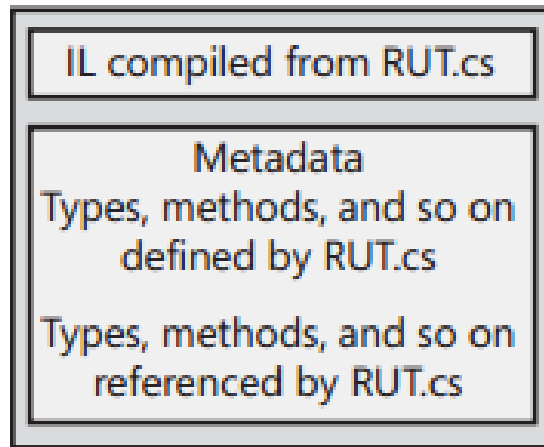
Assembly Linker Utility (al.exe)

- The Assembly Linker is useful if you want to create an assembly consisting of modules built from **different compilers**.
- You can also use AL.exe to build resource-only assemblies, called **satellite assemblies**, which are typically used for localization purposes.

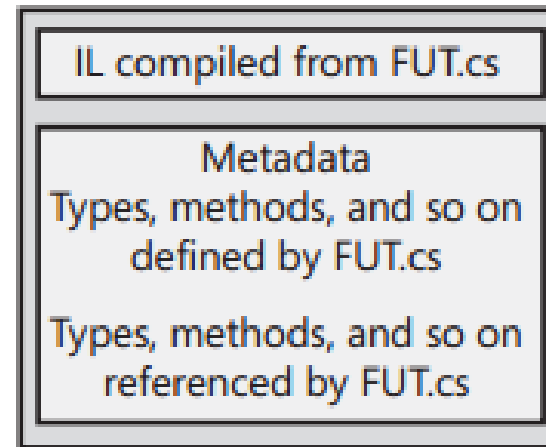
Example

- `csc /t:module RUT.cs`
- `csc /t:module FUT.cs`
- `al /out: MultiFileLibrary.dll`
`/t:library FUT.netmodule RUT.netmodule`

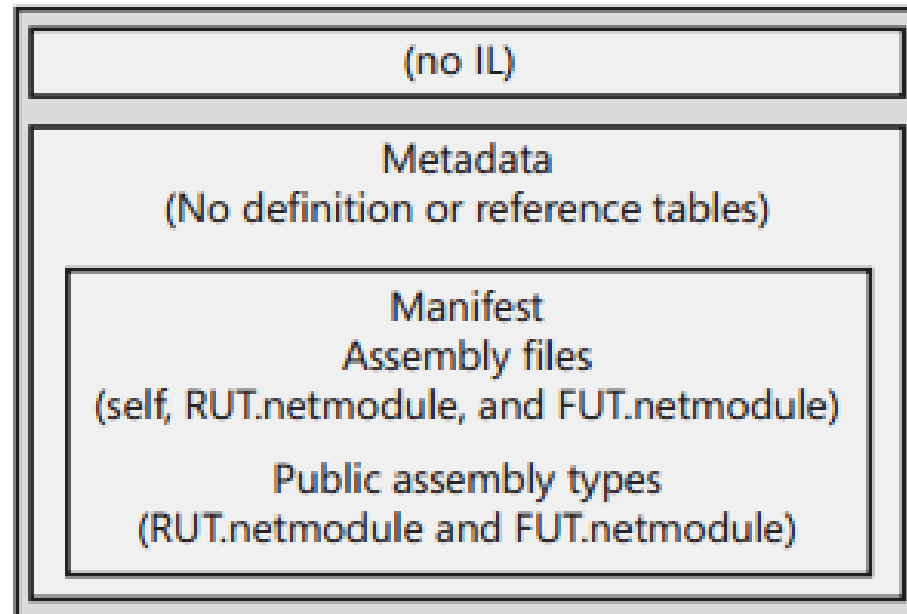
RUT.netmodule



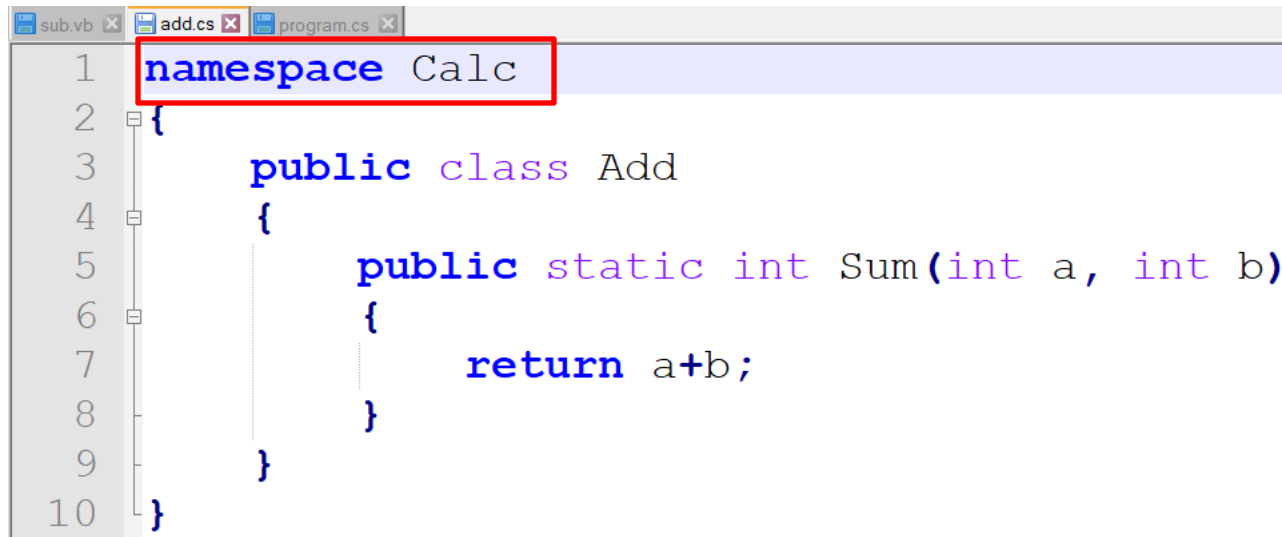
FUT.netmodule



MultiFileLibrary.dll

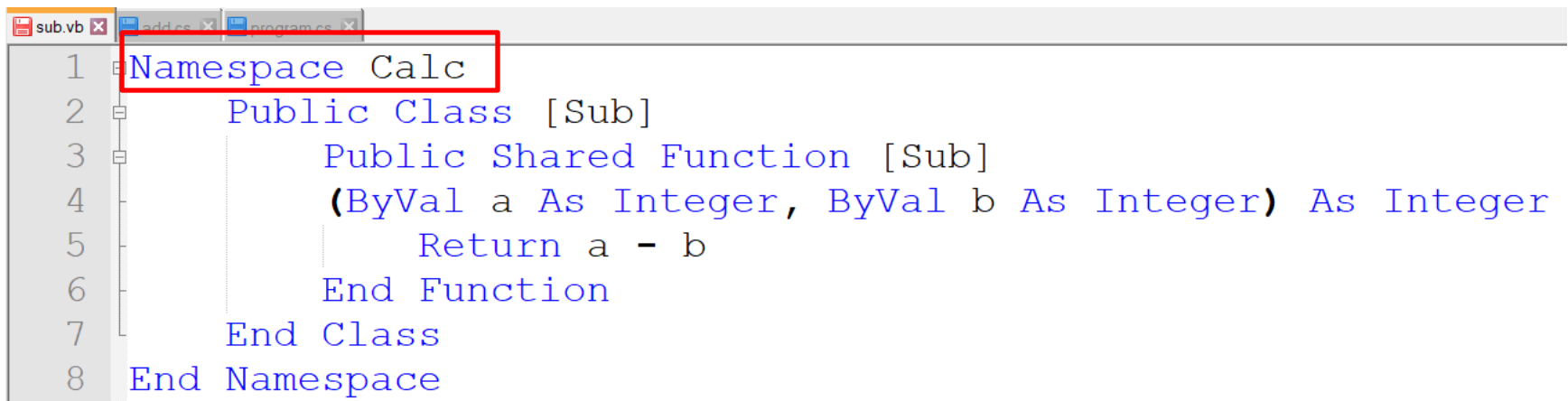


Multifile Assembly - Demo






A screenshot of a Visual Studio code editor window with three tabs: sub.vb, add.cs, and program.cs. The 'add.cs' tab is active and shows C# code. Line 1 contains 'namespace Calc', which is highlighted with a red rectangle. The code defines a 'public class Add' with a 'public static int Sum(int a, int b)' method that returns 'a+b'.

```
1 namespace Calc
2 {
3     public class Add
4     {
5         public static int Sum(int a, int b)
6         {
7             return a+b;
8         }
9     }
10 }
```



A screenshot of a Visual Studio code editor window with three tabs: sub.vb, add.cs, and program.cs. The 'sub.vb' tab is active and shows VB.NET code. Line 1 contains 'Namespace Calc', which is highlighted with a red rectangle. The code defines a 'Public Class [Sub]' with a 'Public Shared Function [Sub]' that takes 'a' and 'b' as integers and returns 'a - b'.

```
1 Namespace Calc
2     Public Class [Sub]
3         Public Shared Function [Sub]
4             (ByVal a As Integer, ByVal b As Integer) As Integer
5             Return a - b
6         End Function
7     End Class
8 End Namespace
```

Name	Date modified	Type	Size
 add.cs	06-07-2019 11:16 AM	Visual C# Source File	1 KB
 program.cs	06-07-2019 11:15 AM	Visual C# Source File	1 KB
 sub.vb	06-07-2019 11:17 AM	Visual Basic Source File	1 KB

```
sub.vb x add.cs x program.cs x
1 using System;
2 using Calc;
3
4 namespace Test
5 {
6     public class MyProgram
7     {
8         public static void Main(string[] args)
9         {
10             int sum = Add.Sum(1,2);
11             int sub = Sub.Sub(1,2);
12
13             Console.WriteLine("Sum: {0}, Sub: {1}", sum, sub);
14             Console.ReadLine();
15         }
16     }
17 }
```

Name	Date modified	Type	Size
add.cs	06-07-2019 11:16 AM	Visual C# Source File	1 KB
add.netmodule	03-07-2020 10:57 AM	NETMODULE File	3 KB
program.cs	06-07-2019 11:15 AM	Visual C# Source File	1 KB
sub.netmodule	03-07-2020 10:57 AM	NETMODULE File	3 KB
sub.vb	06-07-2019 11:17 AM	Visual Basic Source File	1 KB

Windows PowerShell

```
B:\GoogleDrive\Personal\WDDN\WDDN - Lect - 2020-21\Lectures-AAM\Sessional I\1  
assembly\Method 0>csc /t:module add.cs  
Microsoft (R) Visual C# Compiler version 3.5.0-beta4-20153-05 (20b9af91)  
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
B:\GoogleDrive\Personal\WDDN\WDDN - Lect - 2020-21\Lectures-AAM\Sessional I\1  
assembly\Method 0>vbc /t:module sub.vb  
Microsoft (R) Visual Basic Compiler version 3.5.0-beta4-20153-05 (20b9af91)  
Copyright (C) Microsoft Corporation. All rights reserved.
```









Name	Date modified	Type	Size
add.cs	06-07-2019 11:16 AM	Visual C# Source File	1 KB
add.netmodule	03-07-2020 10:57 AM	NETMODULE File	3 KB
calc.dll	03-07-2020 11:14 AM	Application extension	3 KB
program.cs	06-07-2019 11:15 AM	Visual C# Source File	1 KB
sub.netmodule	03-07-2020 10:57 AM	NETMODULE File	3 KB
sub.vb	06-07-2019 11:17 AM	Visual Basic Source File	1 KB

Windows PowerShell

```
B:\GoogleDrive\Personal\WDDN\WDDN - Lect - 2020-21\Lectures-AAM\Sessional I  
ssembly\Method 0>al /t:library /out:calc.dll add.netmodule sub.netmodule  
Microsoft (R) Assembly Linker version 14.8.3928.0  
Copyright (C) Microsoft Corporation. All rights reserved.
```

Will it work?









```
>csc /t:library /out:calcD.dll /addmodule:add.netmodule /addmodule:sub.netmodule
```


Name	Date modified	Type	Size
 add.cs	06-07-2019 11:16 AM	Visual C# Source File	1 KB
 add.netmodule	03-07-2020 10:57 AM	NETMODULE File	3 KB
 calc.dll	03-07-2020 11:14 AM	Application extension	3 KB
 calcD.dll	03-07-2020 11:16 AM	Application extension	3 KB
 program.cs	06-07-2019 11:15 AM	Visual C# Source File	1 KB
 program.exe	03-07-2020 11:20 AM	Application	4 KB
 sub.netmodule	03-07-2020 10:57 AM	NETMODULE File	3 KB
 sub.vb	06-07-2019 11:17 AM	Visual Basic Source File	1 KB

Windows PowerShell

```
B:\GoogleDrive\Personal\WDDN\WDDN - Lect - 2020-21\Lectures-AAM\Sessional  
sembly\Method 0>csc /r:calc.dll program.cs  
Microsoft (R) Visual C# Compiler version 3.5.0-beta4-20153-05 (20b9af91)  
Copyright (C) Microsoft Corporation. All rights reserved.
```


GoogleDrive > Personal > WDDN > WDDN - Lect - 2020-21 > Lectures-AAM > Sessional I >

Name ^	Date modified	Type	Size
 add.cs	06-07-2019 11:16 AM	Visual C# Source File	1 KB
 add.netmodule	03-07-2020 10:57 AM	NETMODULE File	3 KB
 calc.dll	03-07-2020 11:14 AM	Application extension	3 KB
 calcD.dll	03-07-2020 11:16 AM	Application extension	3 KB
 program.cs	06-07-2019 11:15 AM	Visual C# Source File	1 KB
 program.exe	03-07-2020 11:20 AM	Application	4 KB
 sub.netmodule	03-07-2020 10:57 AM	NETMODULE File	3 KB
 sub.vb	06-07-2019 11:17 AM	Visual Basic Source File	1 KB

 B:\GoogleDrive\Personal\WDDN\WDDN - Lect - 2020-21\Lectures-AAM\Sessional I\1Assembly

Sum: 3, Sub: -1

Please, try other methods for creating multifile assemblies, as given in PDF. Inspect, .dll/.netmodule with ILDASM.exe, it will give you more clarification.

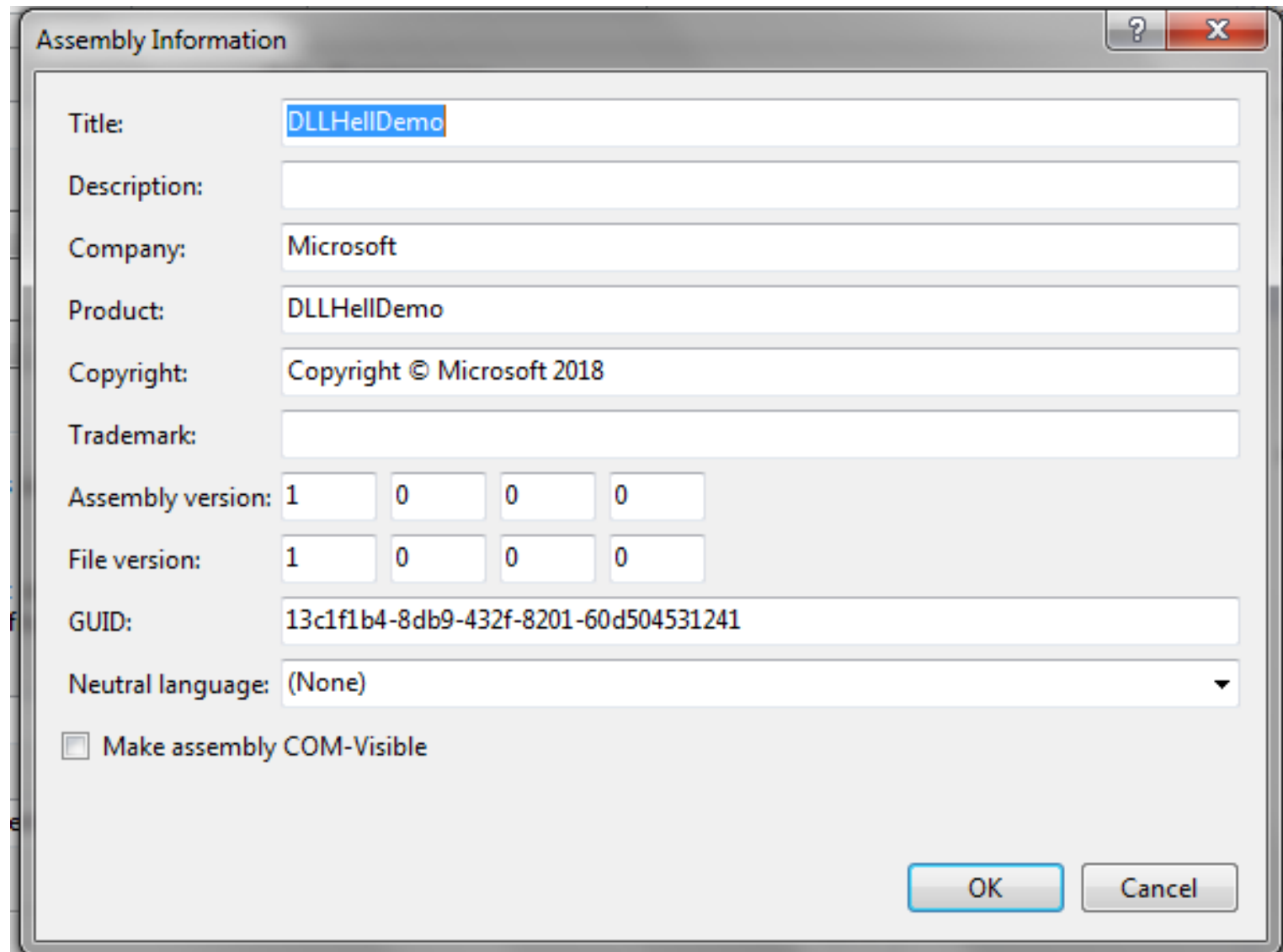
Assembly Version Resource Info

- When AL.exe or CSC.exe produces a PE file assembly, it also embeds into the PE file a standard Win32 version resource.
 - Users can examine this resource by viewing the file's properties.
- Application code can also acquire and examine this information at run time by calling System.Diagnostics.FileVersionInfo's static GetVersionInfo method.

Continue...

- When building an assembly, you should set the version resource fields by using custom attributes that you apply at the assembly level in your source code.
- Version can be set in source code or using assembly linking (al.exe) process.
 - al /version:0.0.0.0

Assembly Info in C#



The image shows a screenshot of the 'Assembly Information' dialog box, a standard Windows interface for setting assembly metadata. The dialog has a title bar with a question mark and a close button. The main area contains several text boxes and a checkbox. The 'Title' field is highlighted with a blue selection box and contains the text 'DLLHellDemo'. The 'Company' field contains 'Microsoft'. The 'Product' field contains 'DLLHellDemo'. The 'Copyright' field contains 'Copyright © Microsoft 2018'. The 'Trademark' field is empty. The 'Assembly version' and 'File version' fields are each composed of four small text boxes, all containing '1', '0', '0', and '0' respectively. The 'GUID' field contains the value '13c1f1b4-8db9-432f-8201-60d504531241'. The 'Neutral language' field is a dropdown menu showing '(None)'. At the bottom left, there is a checkbox labeled 'Make assembly COM-Visible' which is currently unchecked. At the bottom right, there are 'OK' and 'Cancel' buttons.

Assembly Information

Title: DLLHellDemo

Description:

Company: Microsoft

Product: DLLHellDemo

Copyright: Copyright © Microsoft 2018

Trademark:

Assembly version: 1 0 0 0

File version: 1 0 0 0

GUID: 13c1f1b4-8db9-432f-8201-60d504531241

Neutral language: (None)

☐ Make assembly COM-Visible

OK Cancel

Assembly Version System

	Major Number	Minor Number	Build Number	Revision Number
Example:	2	5	719	2

- What is the difference between .netmodule and .dll file?
- When should we promote usage of .netmodule file over .dll file?
- When should we use multifile assembly?

Culture

- Like version numbers, assemblies also have a **culture** as part of their **identity**.
- Cultures are identified via a string that contains a primary and a secondary tag

Primary Tag	Secondary Tag	Culture
De	(none)	German
De	CH	Swiss German
En	US	U.S. English
...

Continue...

- Culture neutral assembly + Satellite assembly
- You should create one satellite assembly for each culture you intend to support.
- You'll usually use the AL.exe tool to build a satellite assembly. You won't use a compiler:

Why?

```
// Set assembly's culture to Swiss German  
[assembly:AssemblyCulture("de-CH")]
```

```
>al /t:library /embed:resource /culture:de /out:MyApp.dll
```

```
al /t:lib /embed:gujarati.png /c:gu-IN /out:guj.dll
```


Two Kinds of Assemblies

- Weakly (*Not strongly*) named assemblies
- Strongly named assemblies.
- Both are structurally identical
 - PE file format
 - PE32(+) header
 - CLR header
 - Metadata
 - Manifest
 - IL

Difference

- A strongly named assembly is signed with a publisher's public/private key pair that uniquely identifies the assembly's publisher.
- This key pair allows the assembly to be uniquely identified, secured, and versioned, and it allows the assembly to be deployed anywhere on the user's machine or even on the Internet.

Two kinds of deployment

Kind of Assembly	Can Be Privately Deployed	Can Be Globally Deployed
Weakly named	Yes	No
Strongly named	Yes	Yes

Giving an assembly strong name

- Why?
- Is file name enough for differentiating assemblies?
- Strongly named assembly
 - Some mechanism that helps uniquely identify an assembly

Attributes of Strongly named assembly

- a file name (w/o extension)
- a version number
- a culture identity
- a public key
 - public key token

Assembly display name assembly identity strings

"MyTypes, Version=1.0.8123.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"

"MyTypes, Version=1.0.8123.0, Culture="en-US", PublicKeyToken=b77a5c561934e089"

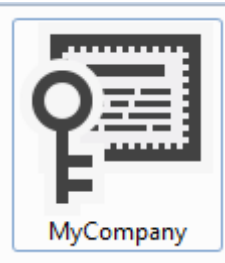
"MyTypes, Version=2.0.1234.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"

"MyTypes, Version=1.0.8123.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"

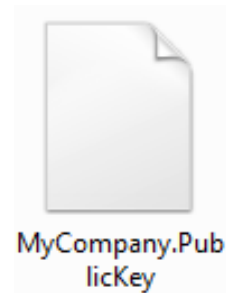
Create strongly named assembly

- Strong Name utility, SN.exe

SN -k MyCompany.snk



**SN -p MyCompany.snk MyCompany.PublicKey
sha256**



Continue...

SN -tp MyCompany.PublicKey

```
Microsoft (R) .NET Framework Strong Name Utility Version 4.0.30319.0
Copyright (c) Microsoft Corporation. All rights reserved.

Public key (hash algorithm: sha256):
0024000000c800000094000000060200000002400000525341310000400000010001005bfb9e986c0247
58b4b2052fb8ce1937081334186289530d18d6e56fb5db7118cf981e549d7c05c6788679435647
b4ac435ccc33f2c3103bf43cc5202d6a03d4cf1c328bd3f4de7256943725ed393a9fa4a3a633a9
3a6ad544fcea98f6a45f613ae8f130becececeff4853e15964935c9cc1a9f544d7117212a07cb3
16fa24a2

Public key token is 7f7f6c090563eda9
```

To display private key? 😊

Continue...

`csc /keyfile:MyCompany.snk Program.cs`

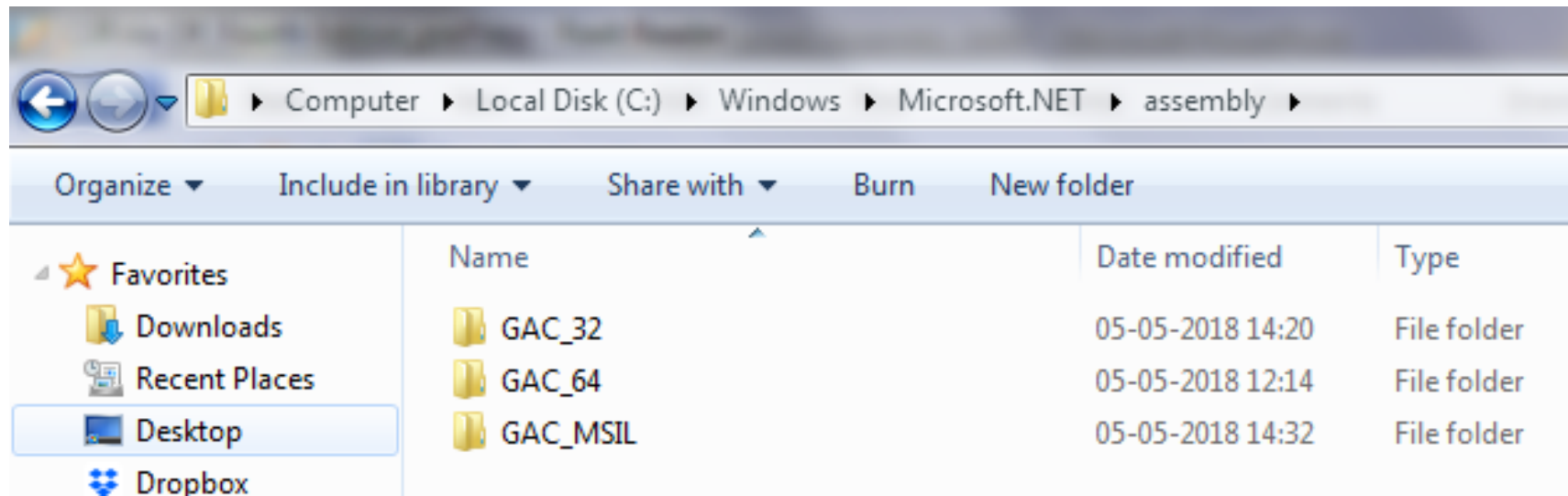
How do we create key and sign to project in visual studio?

Public key and Public key token

Does CLR use public key token?

The Global Assembly Cache

- An assembly can be accessed by multiple applications.
- Exact location of GAC is implementation details.
- GAC directory is structured.



GACUtil.exe

- Only strongly named assembly can be placed into GAC
- Need administrative rights
- /i to install
- /u to uninstall

Registering Assembly

- Two companies each produce an OurLibrary assembly consisting of one file: OurLibrary.dll.
- Obviously, both of these files can't go in the same directory
- When you install an assembly into the GAC, dedicated subdirectories are created under the %SystemRoot%\Microsoft.NET\Assembly directory, and the assembly files are copied into one of these subdirectories.

Steps

```
sn -k Company1.snk
```

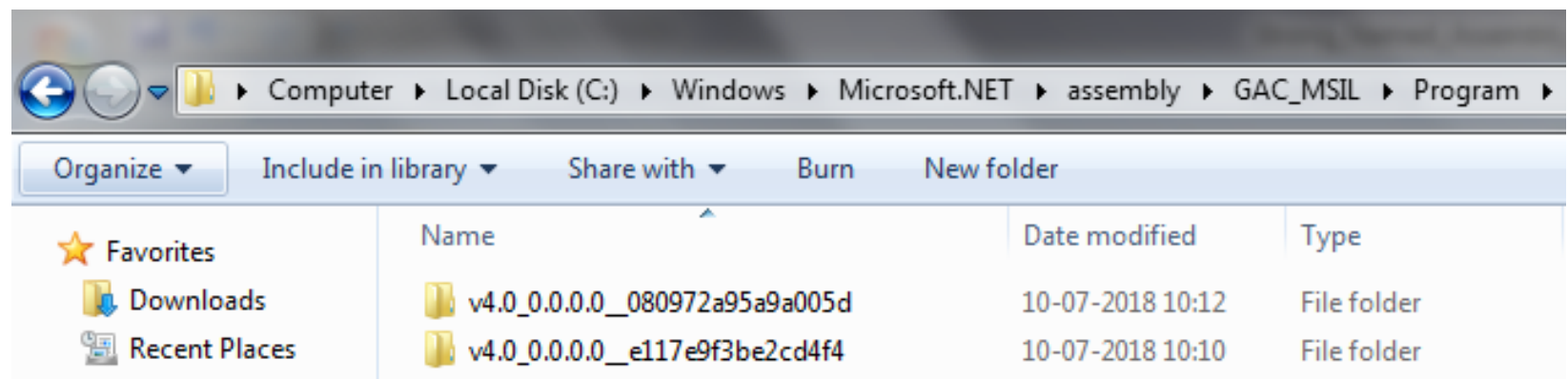
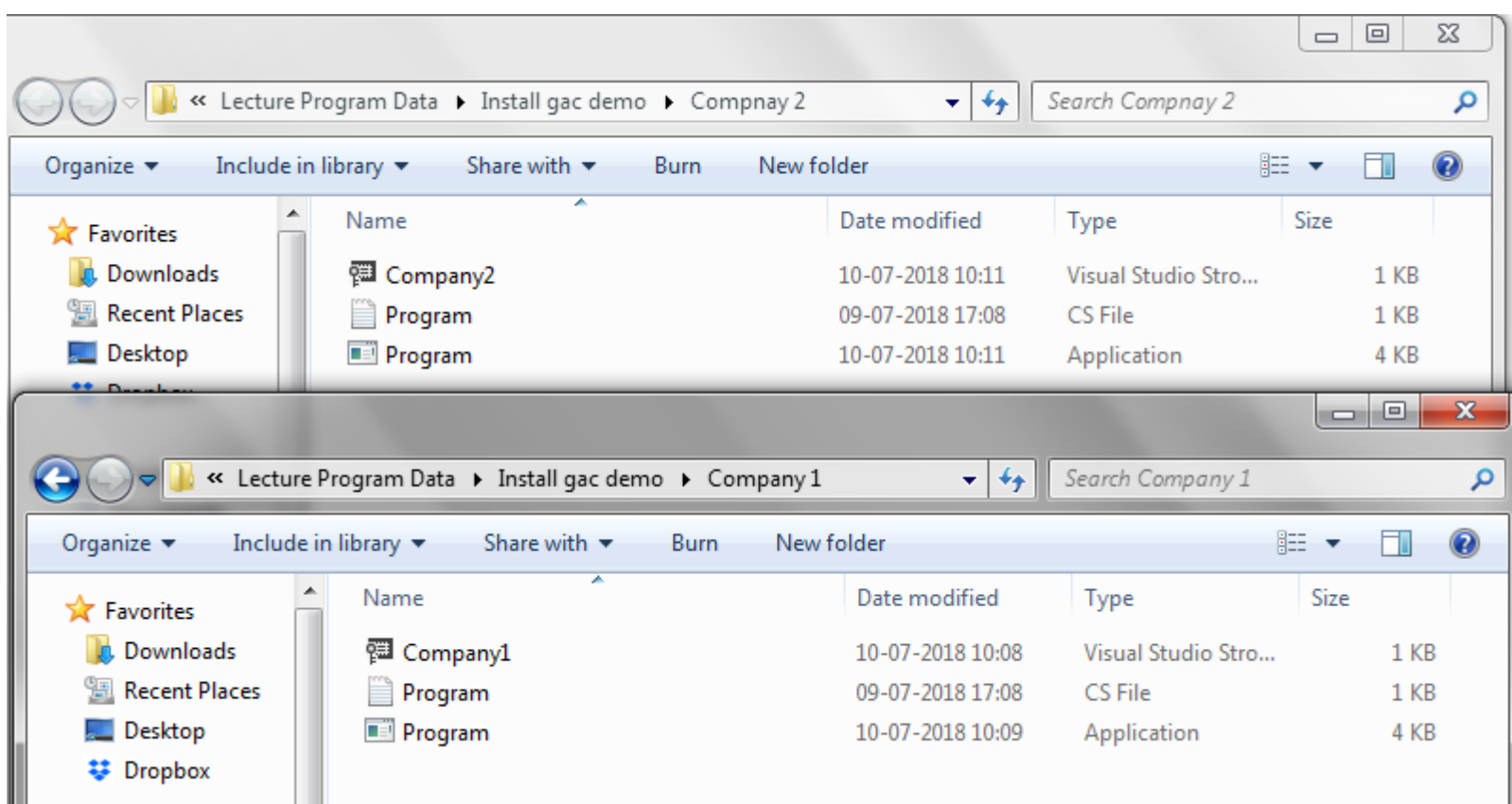
```
csc /keyfile:Company1.snk Program.cs
```

```
gacutil /i Program.exe
```

```
sn -k Company2.snk
```

```
csc /keyfile:Company2.snk Program.cs
```

```
gacutil -i Program.exe
```



When you build an assembly, the assembly will have references to other strongly named assemblies.

True

False

Building source Program

- How does csc.exe find assemblies?
- When you install the .NET Framework, two copies of Microsoft's assembly files are actually installed.