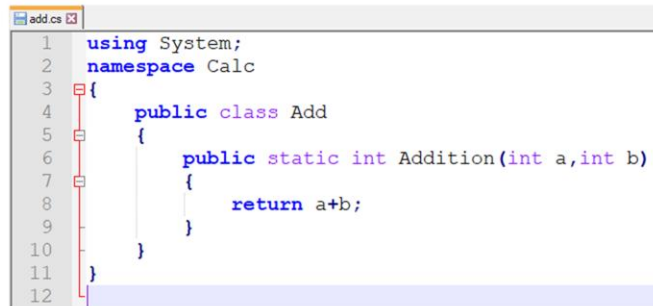


Multifile Assembly, Strongly Named Assembly

Prepared for Vth semester DDU-CE students
2022-23 WAD

Apurva A Mehta

Ex. 1

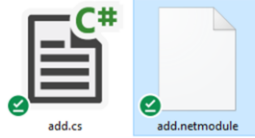


```
1  using System;
2  namespace Calc
3  {
4      public class Add
5      {
6          public static int Addition(int a,int b)
7          {
8              return a+b;
9          }
10     }
11 }
12
```

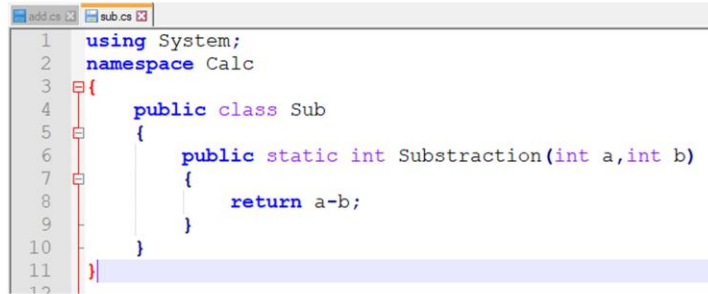
The image shows a screenshot of a C# code editor window titled 'add.cs'. The code defines a namespace 'Calc' containing a public class 'Add'. Inside the 'Add' class, there is a public static method 'Addition' that takes two integers 'a' and 'b' as parameters and returns their sum 'a+b'. The code is formatted with standard C# syntax highlighting: keywords like 'using', 'namespace', 'public', 'class', 'static', 'int', and 'return' are in blue; the class name 'Add' is in purple; and the method name 'Addition' is in purple. The method parameters 'a' and 'b' are in black. The code is enclosed in curly braces to define the scope of the namespace and the class. Line numbers 1 through 12 are visible on the left side of the editor.

Ex. 1

```
E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex1>csc /t:module add.cs
Microsoft (R) Visual C# Compiler version 3.4.1-beta4-19614-01 (16504609)
Copyright (C) Microsoft Corporation. All rights reserved.
```



Ex. 1



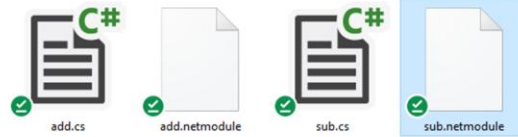
The image shows a screenshot of a C# code editor with two tabs: 'add.cs' and 'sub.cs'. The 'sub.cs' tab is active. The code is as follows:

```
1 using System;
2 namespace Calc
3 {
4     public class Sub
5     {
6         public static int Substraction(int a,int b)
7         {
8             return a-b;
9         }
10    }
11 }
```

The code defines a namespace 'Calc' containing a public class 'Sub'. Inside the 'Sub' class, there is a public static method 'Substraction' that takes two integers 'a' and 'b' as parameters and returns their difference 'a-b'. The method is implemented with a single return statement. The code is syntactically correct, although the spelling 'Substraction' is unusual.

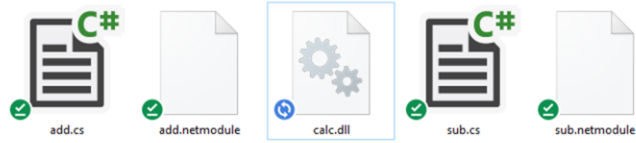
Ex. 1

```
E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex1>csc /t:module sub.cs  
Microsoft (R) Visual C# Compiler version 3.4.1-beta4-19614-01 (16504609)  
Copyright (C) Microsoft Corporation. All rights reserved.
```



Ex. 1

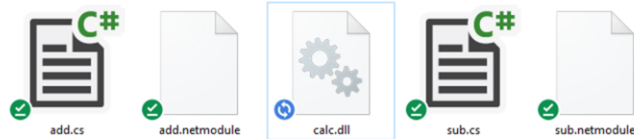
```
Developer Command Prompt for VS 2019
E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex1>csc /t:library /out:calc.dll /addmodule:add.net
module /addmodule:sub.netmodule
Microsoft (R) Visual C# Compiler version 3.4.1-beta4-19614-01 (16504609)
Copyright (C) Microsoft Corporation. All rights reserved.
```



Ex. 1

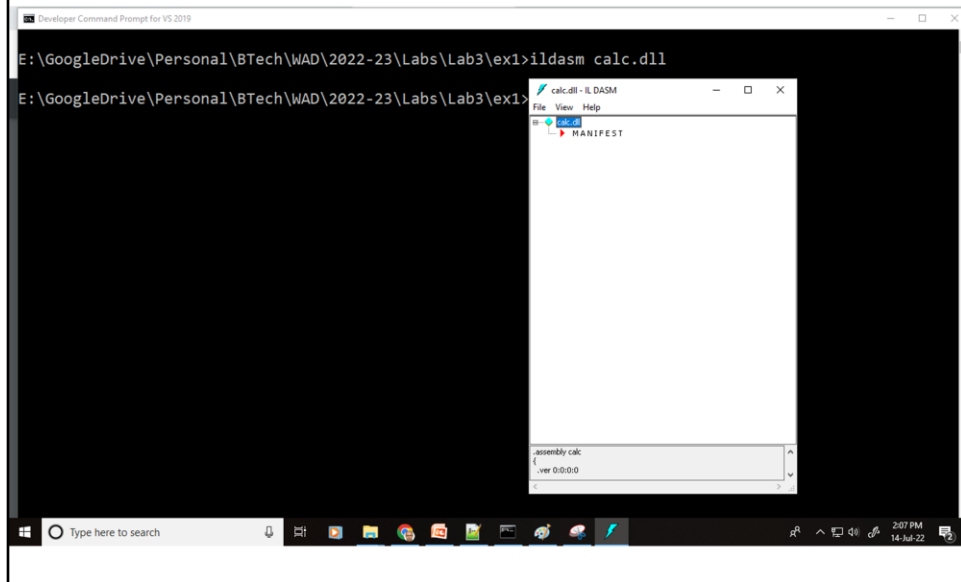
```
Developer Command Prompt for VS 2019
E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex1>al /t:library /out:calc.dll add.netmodule sub.netmodule
Microsoft (R) Assembly Linker version 14.8.3928.0
Copyright (C) Microsoft Corporation. All rights reserved.

E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex1>
```



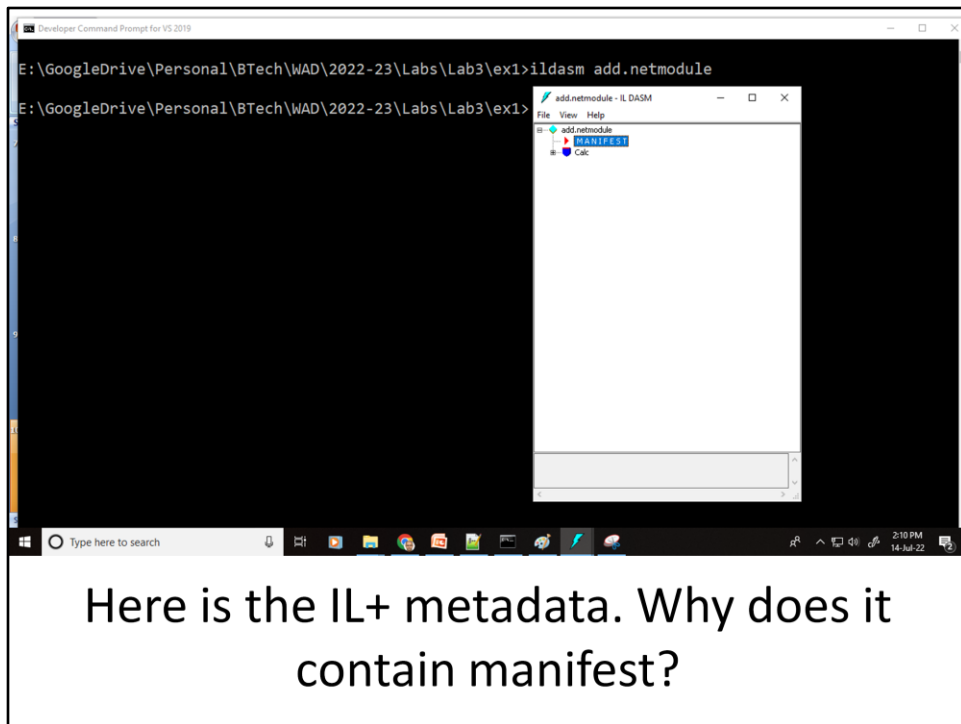
Is the manifest standalone or attached to one of the assembly?

Where is the IL, Metadata?




```
MANIFEST
Find Find Next

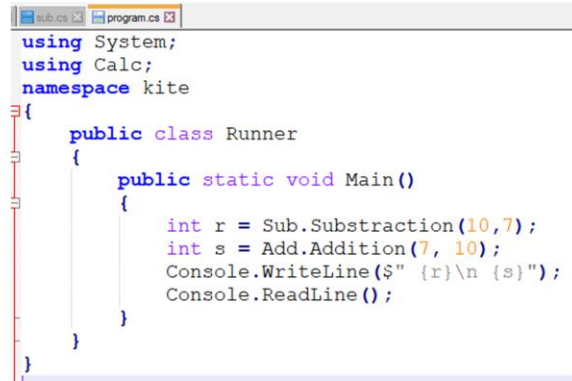
// Metadata version: v4.0.30319
.assembly externmscorlib
{
    publickeytoken = (B7 7A 5C 56 19 34 E0 89 ) // .NET
    .ver 4:0:0:0
}
.assembly calc
{
    .hash algorithm 0x00000004
    .ver 0:0:0:0
}
.file add.netmodule
{
    .hash = (C8 40 38 AD A3 7A 6F 17 27 9F F3 AC 4D E9 F1 02) // .J8HCzo...'...H...
    .hash = (B9 F8 58 1B ) // ..X.
}
.file sub.netmodule
{
    .hash = (3A E8 88 D2 7A 81 2C BF 02 82 81 A2 28 8A D0 23) // 4...Z......B
    .hash = (03 7C 1B FF ) // -|..
}
.class extern public Calc.Add
{
    .file add.netmodule
    .class 0x02000002
}
.class extern public Calc.Sub
{
    .file sub.netmodule
    .class 0x02000002
}
.module calc.dll
// GUID: {136D34A9-D87A-4763-9A3D-5D77D77283C6}
.imagebase 0x00000000
.file alignment 0x00000200
.stackreserve 0x00100000
.subsystem 0x0002 // WINDOWS_CUI
.corflags 0x00000001 // ILONLY
// Image base: 0x00CE0000
```



Here is the IL+ metadata. Why does it contain manifest?

It does not contain manifest. It is just the information about its own assembly.

Ex. 1



```
using System;
using Calc;
namespace kite
{
    public class Runner
    {
        public static void Main()
        {
            int r = Sub.Substraction(10,7);
            int s = Add.Addition(7, 10);
            Console.WriteLine($" {r}\n {s}");
            Console.ReadLine();
        }
    }
}
```

Ex. 1



```
Developer Command Prompt for VS 2019
E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex1>csc program.cs /r:calc.dll
Microsoft (R) Visual C# Compiler version 3.4.1-beta4-19614-01 (16504609)
Copyright (C) Microsoft Corporation. All rights reserved.

E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex1>program.exe
3
17

E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex1>
```

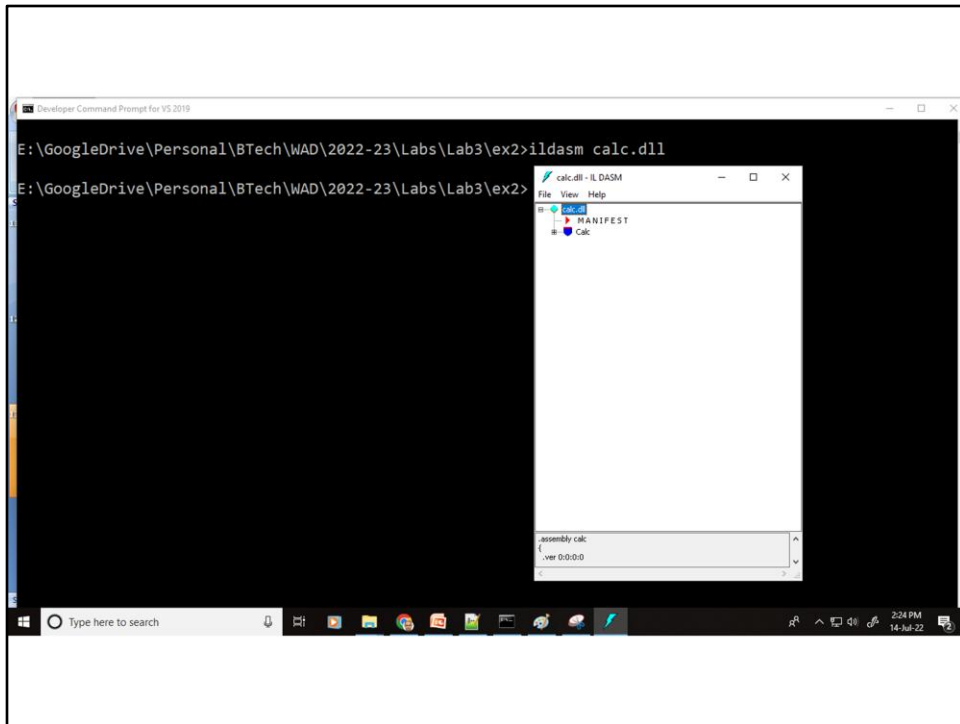
Ex. 2

- Create a multifile assembly such that manifest is attached to one of the module.

Step-1 `csc /t:module add.cs`

Step-2 `csc /out:calc.dll /t:library /addmodule:add.netmodule sub.cs`

Step-3 `csc /r:calc.dll program.cs`



Ex. 3

```
E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex2>SN -k MyCompany.snk

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

Failed to generate a strong name key pair -- Access is denied.
```

The first step in creating a strongly named assembly is to obtain a key by using the Strong Name utility, SN.exe, that ships with the .NET Framework SDK and Microsoft Visual Studio. This utility offers a whole slew of features depending on the command-line switch you specify. Note that all SN.exe's command-line switches are case-sensitive. To generate a public/private key pair, you run SN.exe as follows:

```
SN -k MyCompany.snk
```

This line tells SN.exe to create a file called MyCompany.snk. This file will contain the public and private key numbers persisted in a binary format. Public key numbers are very big. If you want to, after creating the file that contains the public and private key, you can use the SN.exe utility again to see the actual public key. To do this, you must execute the SN.exe utility twice. First, you invoke SN.exe with the `-p` switch to create a file that contains only the public key (MyCompany.PublicKey):

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

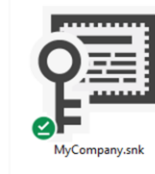
Ex. 3

```
Administrator: Developer Command Prompt for VS 2019
E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex3>SN -k MyCompany.snk

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

Key pair written to MyCompany.snk

E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex3>
```



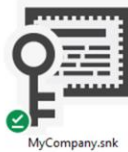
Ex. 3

```
Administrator: Developer Command Prompt for VS 2019
E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex3>SN -p MyCompany.snk MyC
ompany.PublicKey sha256

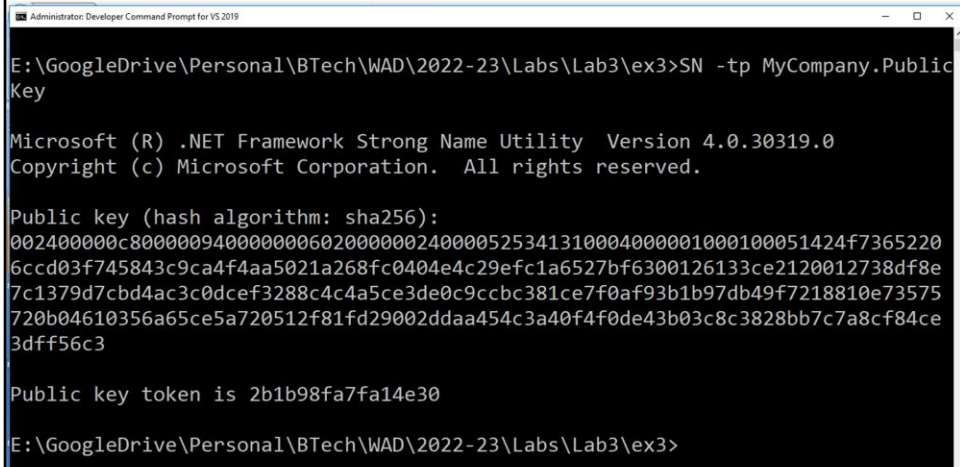
Microsoft (R) .NET Framework Strong Name Utility Version 4.0.30319.0
Copyright (c) Microsoft Corporation. All rights reserved.

Public key written to MyCompany.PublicKey

E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex3>
```



Ex. 3



```
Administrator: Developer Command Prompt for VS 2019
E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex3>SN -tp MyCompany.Public
Key

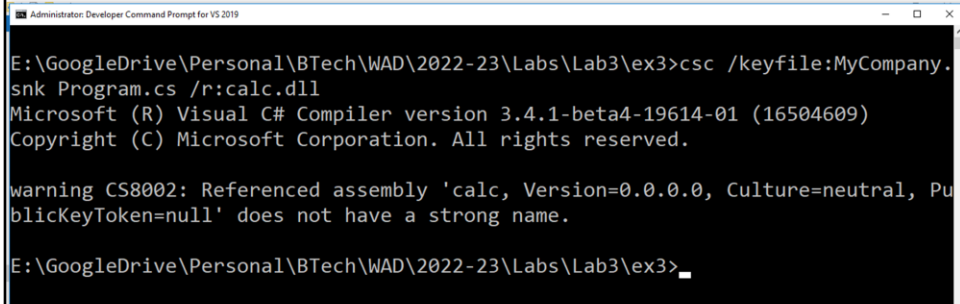
Microsoft (R) .NET Framework Strong Name Utility  Version 4.0.30319.0
Copyright (c) Microsoft Corporation.  All rights reserved.

Public key (hash algorithm: sha256):
002400000c800000940000000602000000240000525341310004000001000100051424f7365220
6ccd03f745843c9ca4f4aa5021a268fc0404e4c29efc1a6527bf6300126133ce2120012738df8e
7c1379d7cbd4ac3c0dcef3288c4c4a5ce3de0c9ccbc381ce7f0af93b1b97db49f7218810e73575
720b04610356a65ce5a720512f81fd29002ddaa454c3a40f4f0de43b03c8c3828bb7c7a8cf84ce
3dff56c3

Public key token is 2b1b98fa7fa14e30

E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex3>
```

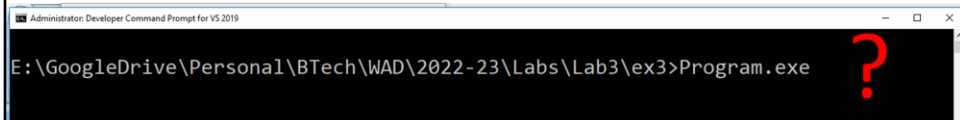
Ex. 3



```
Administrator: Developer Command Prompt for VS 2019
E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex3>csc /keyfile:MyCompany.snk Program.cs /r:calc.dll
Microsoft (R) Visual C# Compiler version 3.4.1-beta4-19614-01 (16504609)
Copyright (C) Microsoft Corporation. All rights reserved.

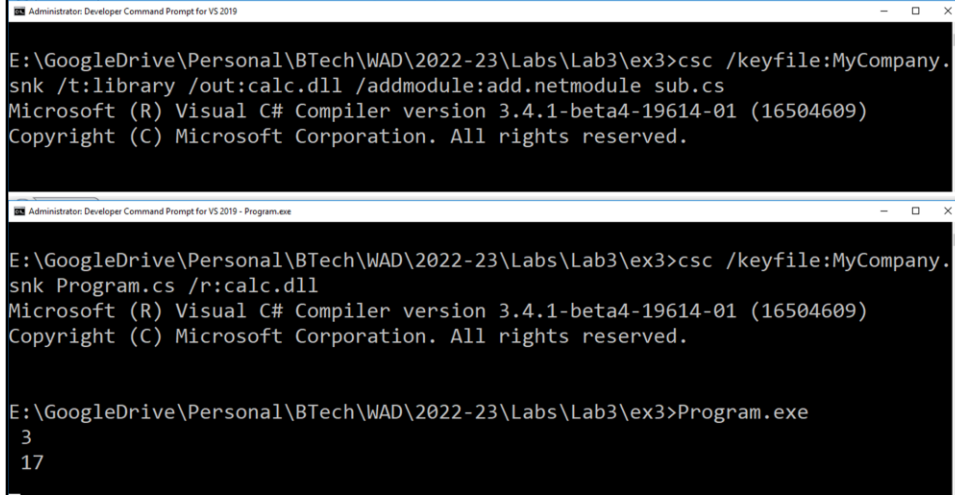
warning CS8002: Referenced assembly 'calc, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null' does not have a strong name.

E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex3>
```



```
Administrator: Developer Command Prompt for VS 2019
E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex3>Program.exe ?
```

Ex. 3



The image shows two screenshots of a Windows Developer Command Prompt window. The top screenshot shows the compilation of a C# program using the Visual C# Compiler (csc). The command is: `E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex3>csc /keyfile:MyCompany.snk /t:library /out:calc.dll /addmodule:add.netmodule sub.cs`. The output shows the compiler version (3.4.1-beta4-19614-01) and the copyright notice. The bottom screenshot shows the execution of the compiled program. The command is: `E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex3>Program.exe`. The output shows the numbers 3 and 17 on separate lines.

```
E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex3>csc /keyfile:MyCompany.snk /t:library /out:calc.dll /addmodule:add.netmodule sub.cs
Microsoft (R) Visual C# Compiler version 3.4.1-beta4-19614-01 (16504609)
Copyright (C) Microsoft Corporation. All rights reserved.

E:\GoogleDrive\Personal\BTech\WAD\2022-23\Labs\Lab3\ex3>Program.exe
3
17
```

How can we verify that assemblies are strongly named?

ildasm