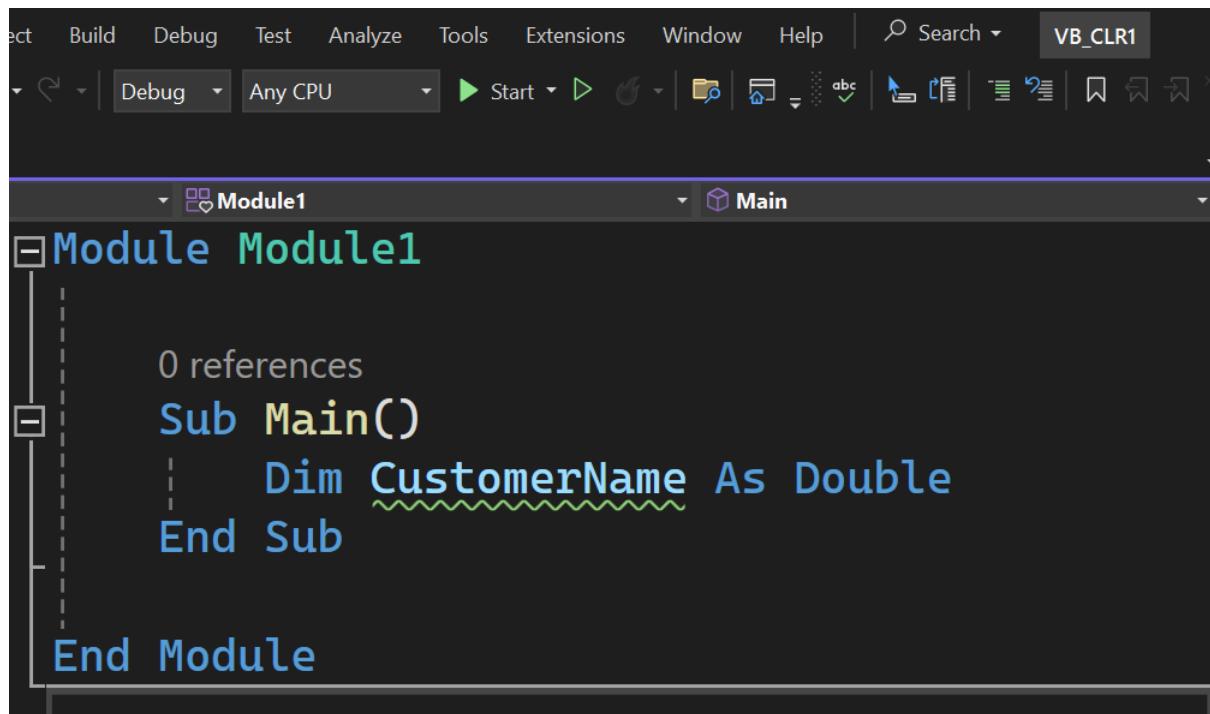


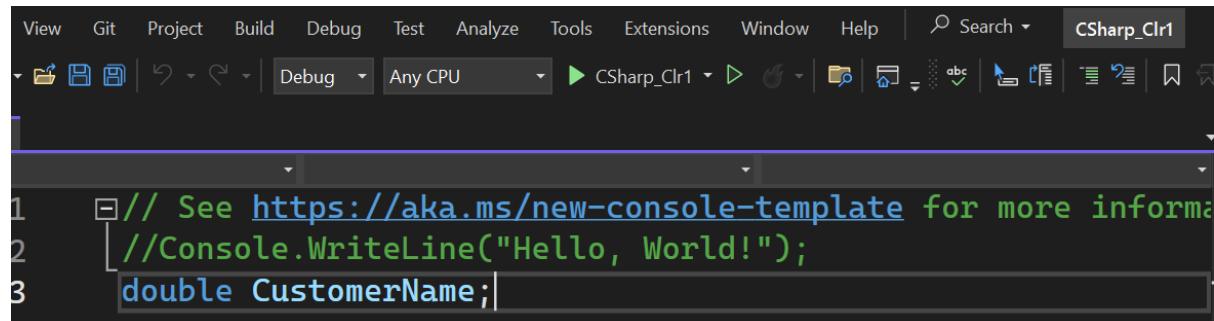
PT LAB:5 (CLR)

➤ VB:



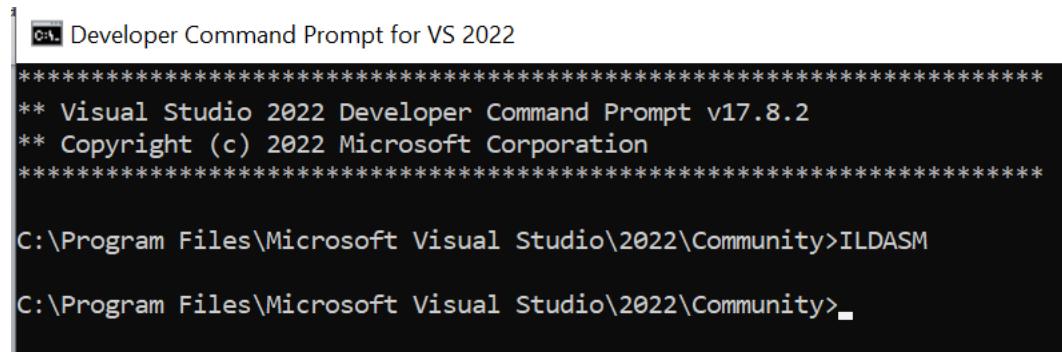
```
Module Module1
    Sub Main()
        Dim CustomerName As Double
    End Sub
End Module
```

➤ C#:



```
// See https://aka.ms/new-console-template for more information
//Console.WriteLine("Hello, World!");
double CustomerName;
```

➤ Developer Command Prompt for VS 2022:



```
Developer Command Prompt for VS 2022
*****
** Visual Studio 2022 Developer Command Prompt v17.8.2
** Copyright (c) 2022 Microsoft Corporation
*****

C:\Program Files\Microsoft Visual Studio\2022\Community>ILDASM

C:\Program Files\Microsoft Visual Studio\2022\Community>_
```

PT LAB:5 (CLR)

VB:

The screenshot shows the ILSpy interface. At the top, it displays the file path: C:\source\repos\VB_CLR1\VB_CLR1\bi... The menu bar includes File, View, and Help. Below the menu is a tree view of the assembly structure:

- C:\source\repos\VB_CLR1\VB_CLR1\bin\Debug\VB_CLR1.exe
- MANIFEST
- VB_CLR1
- VB_CLR1.My
- VB_CLR1.Module1
 - .class private auto ansi sealed
 - .custom instance void [Microsoft.VisualBasic]Microsoft.Visu
 - Main : void()

Below the assembly tree, there is a search bar with "Find" and "Find Next" buttons. The main pane displays the IL code for the Main method:

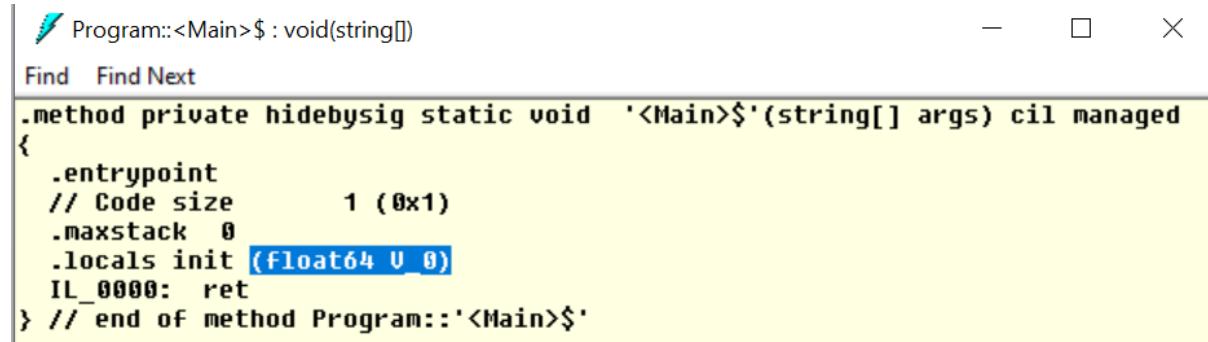
```
.method public static void Main() cil managed
{
    .entrypoint
    .custom instance void [mscorlib]System.STAThreadAttribute::ctor() = ( 01 0
    // Code size      2 (0x2)
    .maxstack 0
    .locals init ([0] float64 CustomerName)
    IL_0000:  nop
    IL_0001:  ret
} // end of method Module1::Main
```

C#:

The screenshot shows the ILSpy interface. At the top, it displays the file path: C:\source\repos\CSharp_Clr1\CSharp_Clr1\bin\Debug\net8.0\CSharp_Clr1.exe. The menu bar includes File, View, and Help. Below the menu is a tree view of the assembly structure:

- C:\source\repos\CSharp_Clr1\CSharp_Clr1\bin\Debug\net8.0\CSharp_Clr1.exe
- MANIFEST
- Program
- .class private auto ansi beforefieldinit
- .custom instance void [System.Runtime]System.Runtime.Com
- .ctor : void()
- <Main>\$: void(string[])

PT LAB:5 (CLR)



The screenshot shows a debugger interface with assembly code. The code is for a method named 'Program::<Main>\$'. It includes directives like .method private hidebysig static void '<Main>\$'(string[] args) cil managed, .entrypoint, and .locals init (float64 v_0). The assembly code ends with IL_0000: ret. The interface has standard window controls (minimize, maximize, close) and menu items 'Find' and 'Find Next'.

```
Program::<Main>$ : void(string[])
Find Find Next
.method private hidebysig static void  '<Main>$'(string[] args) cil managed
{
    .entrypoint
    // Code size      1 (0x1)
    .maxstack  0
    .locals init (float64 v_0)
    IL_0000:  ret
} // end of method Program::<Main>$
```

Discussion:

Here, in our PROGRAMMING TECHNOLOGY (PT)-Lab 5, we explored the Common Language Runtime (CLR) in .NET, which is like the "brain" behind the scenes that helps run our programs smoothly.

Moreover, in this lab, we dived into the Common Language Runtime (CLR) in .NET using C# and VB.NET. We learned how to declare a variable with the double data type in both languages. In C#, it's as simple as writing "double" followed by the variable name, while in VB.NET, we use "Dim" and specify "As Double" after the variable name. This hands-on experience helped us see the language differences in declaring double variables and reinforced our understanding of how CLR supports various languages within the .NET framework.

Conclusion:

Hence, this lab helped us to understand the practical aspects of how the "CLR" makes our DOT NET applications work effectively and securely.