**Experiment No.: 8**  **Date: 23/11/ 2020**

**Common Intermediate Language**

**Aim:** To study the Common Intermediate Language (CIL) of .NET binaries.

**Theory:**

.NET binaries contain platform-agnostic Intermediate Language (IL). The \*.dll or \*.exe created using a .NET compiler is termed an assembly. An assembly contains CIL code, which is conceptually like Java bytecode – not compiled to platform-specific instructions until necessary. CIL is a .NET programming language with its own syntax, semantics, and compiler (ilasm.exe). CIL is a stack-based programming language. The Virtual Execution Stack holds a set of values to be evaluated. Data must be explicitly loaded onto the stack and popped when needed. CIL is compiled on demand. During compilation, many implementation redundancies are optimized.

**Programs**

1)Create a new project in Visual Studio choosing Template as Visual C#/ Visual Basic and Console Application.

2) Compile using the C#/VB compiler.

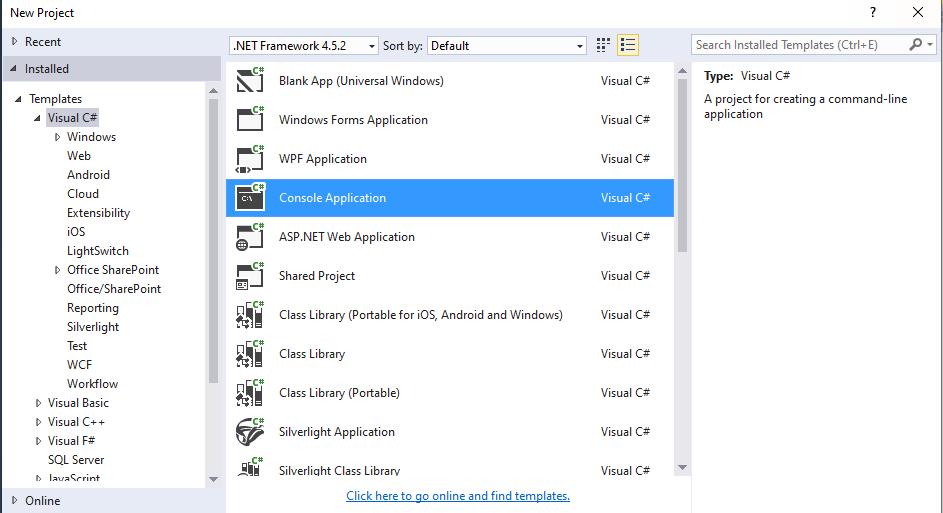
3) Open the assembly using ildasm.exe – it is a CIL decompiler. It translates an assembly’s binary opcodes into their corresponding CIL mnemonics.

[C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.6.1 Tools]

File-Open {Go to your\_folder\bin\Debug and open the \*.exe}

Double click on Main to see the CIL.

**C#**



using System;

namespace IntLang

{

class Program

{

static void Main(string[] args)

{

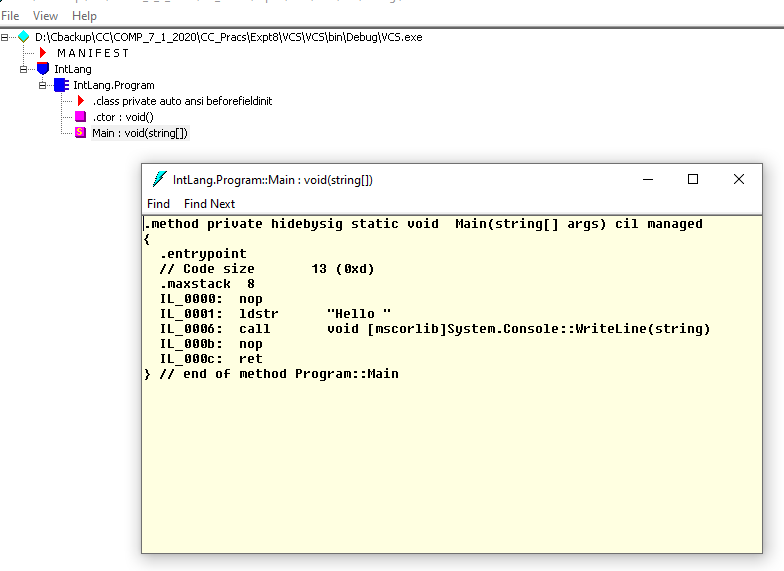
Console.WriteLine("Hello ");

}

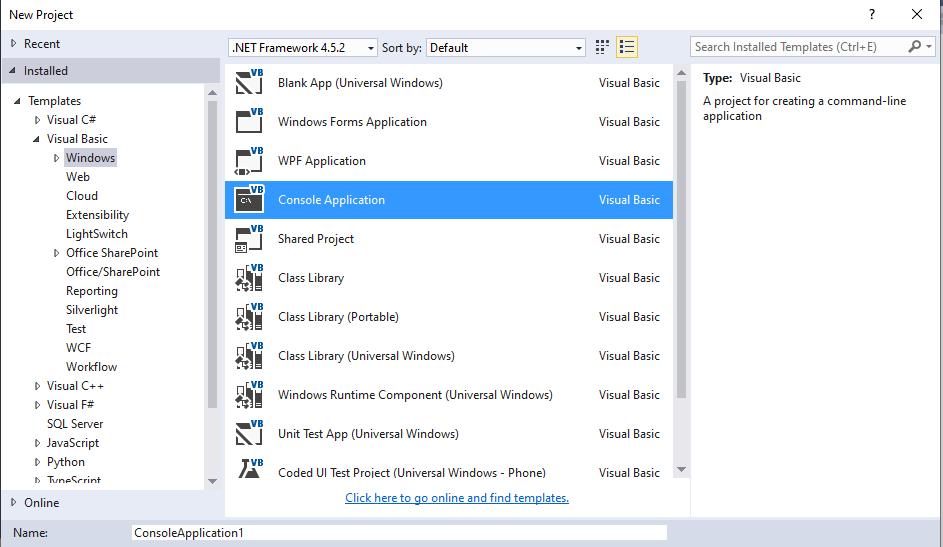
}

}

**Output**



**VB**



Module Module1

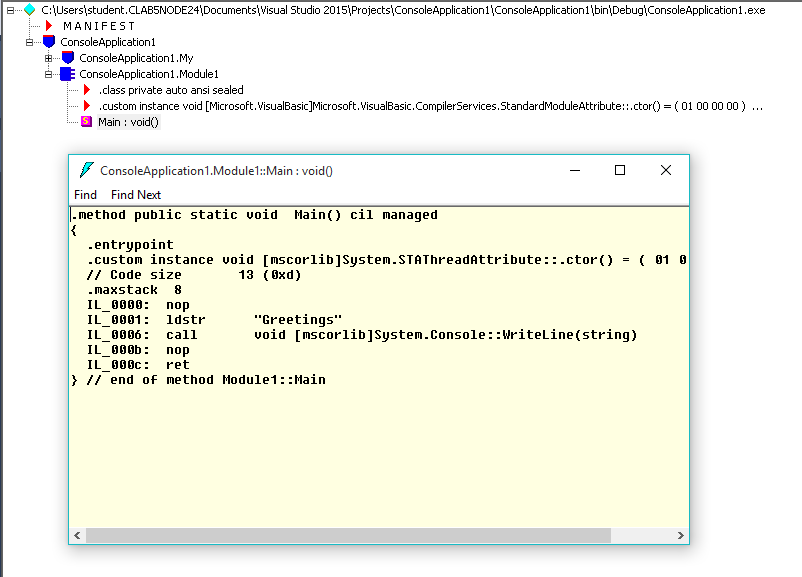
Sub Main()

Console.WriteLine("Greetings")

End Sub

End Module

**Output**



**Conclusion:** Common Intermediate Language program was successfully implemented.