**.NET ASSIGNMENT DONE BY MAYANK GOYAL**

Instructions:

1. Please submit your answers in Google Docs.
2. In case of Pictorial representations, either share the online link or upload the hand drawn models.
3. Submit your assignment by **30th July EOD.**
4. Mail your assignment submissions to [raghuvardhan.karanam@accoliteindia.com](mailto:raghuvardhan.karanam@accoliteindia.com).
5. Please feel free to ping me if you have any questions.

Assignment:

1. Demonstrate the process of conversion of Source code into the native machine code in .Net framework with the help of a flowchart.
2. Explain in detail the CTS and how the .net framework implements CTS.
3. Name at least 3 runtime services provided by CLR and explain their role in .net framework.
4. What are the differences between Library vs DLL vs .Exe? Explain.
5. How does CLR in .net ensure security and type safety? Explain.

By

Raghu Karanam.

ASSIGNMENT START:-

Q1. Demonstrate the process of conversion of Source code into the native machine code in .Net framework with the help of a flowchart.

[.NET Framework Compilation Process](http://geekswithblogs.net/manjunath.k/archive/2013/06/12/.net-framework-compilation-proces.aspx)

**Compilation Process:**

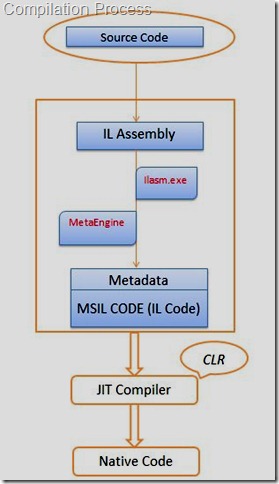
When any application is written targeting .net framework, the code will get compiled with respective compiler (say csc compiler OR vs compiler), based on whether we are using C# as a language or VB. After compilation of the code, what we get is an **IL Assembly** (Intermediate Language Assembly) later which will be converted into **IL Code or Byte Code**(Intermediate Language Code). The process of converting IL Assembly to IL Code will be done with the help of two tools they are **ilasm.exe** and **Meta Engine.**The .exe or .dll (dynamic link library) file what we get contains IL Code, later which will converted in to native code.

Where ilasm.exe converts IL Assembly to MSIL Code, meanwhile Meta Engine creates a metadata for the IL Assembly. Hence, once source code is compiled the IL code generated contains metadata as well as Inter mediate format code.

Any .exe generated for the code written contains IL code. When this .exe is executed on any system which has .NET Framework, then IL Code will be converted in to **native code (Machine understandable code)**. The Process of Converting IL Code to native code will be done with the help of **JIT compiler (Just-In-Time Compiler).**

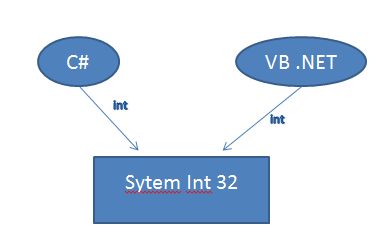
**JIT** will be present at **CLR** (Common Language Runtime) level.

For better understanding purpose, I am giving pictorial representation.

[](https://gwb.blob.core.windows.net/manjunathk/.netcompilation.jpg)

Q2. Explain in detail the CTS and how the .net framework implements CTS.

**CTS**  
The Common Type System (CTS) standardizes the data types of all programming languages using .NET under the umbrella of .NET to a common data type for easy and smooth communication among these .NET languages.  
  
**How CTS converts the data type to a common data type**

To implement or see how CTS is converting the data type to a common data type, for example, when we declare an int type data type in C# and VB.Net then they are converted to int32. In other words, now both will have a common data type that provides flexible communication between these two languages.  
  
  
  
Let's take a live example using a Double data type and we will see how .NET helps to make the data types of C# and VB.NET common to each other for easy communication.

Q3. Name at least 3 runtime services provided by CLR and explain their role in .net framework.

CLR provides multiple services to execute processes, like memory management service and security services. CLR performs multiple tasks to manage the execution of .NET applications. **Following responsibilities of CLR are given below:**

1. Automatic memory management
2. Code access security
3. Garbage collection
4. JIT compilation

### **1) Automatic memory management**

CLR calls various predefined functions of .NET framework to allocate and de-allocate memory of .NET objects. So that, developers need not to write code to explicitly allocate and de-allocate memory.

### **2) Code access security**

CLR allows access to code to perform only those tasks for that it has permission. It also checks user’s permissions using authentication and configuration files of .NET applications.

### **3) Garbage collection**

GC is used to prevent memory leaks or holes. Garbage collector of CLR automatically determines the best time to free the memory, which is allocated to an object for execution.

Q4. What are the differences between Library vs DLL vs .Exe? Explain.

Libraries are used because you may have code that you want to use in many programs. For example, if you write a function that counts the number of characters in a string, that function will be useful in lots of programs. Once you get that function working correctly you don't want to have to recompile the code every time you use it, so you put the executable code for that function in a library, and the linker can extract and insert the compiled code into your program. Static libraries are sometimes called 'archives' for this reason.

Dynamic libraries take this one step further. It seems wasteful to have multiple copies of the library functions taking up space in each of the programs. Why can't they all share one copy of the function? This is what dynamic libraries are for. Rather than building the library code into your program when it is compiled, it can be run by mapping it into your program as it is loaded into memory. Multiple programs running at the same time that use the same functions can all share one copy, saving memory. In fact, you can load dynamic libraries only as needed, depending on the path through your code. No point in having the printer routines taking up memory if you aren't doing any printing. On the other hand, this means you have to have a copy of the dynamic library installed on every machine your program runs on. This creates its own set of problems.

The terms EXE and DLL are very common in programming. When coding, you can either export your final project to either a DLL or an EXE. The term EXE is a shortened version of the word executable as it identifies the file as a program. On the other hand, DLL stands for Dynamic Link Library, which commonly contains functions and procedures that can be used by other programs.

A DLL is a library and therefore can not be executed. If you try to run it you'll get an error about a missing entry point. An EXE is a program that can be executed. It has an entry point. A flag inside the PE header indicates which file type it is (irrelevant of file extension).

A DLL would share the same process and memory space of the calling application while an EXE creates its separate process and memory space.

Q5. How does CLR in .net ensure security and type safety? Explain.

#### Safe Execution Environment

With the CLR, a compiler generates MSIL instructions, not native code. It is this managed code that runs. Hence, the CLR can perform runtime validations on this code before it is translated into native code. Types are verified. Subscripts are verified to be in range. Unsafe casts and uninitialized variables are prevented.

The CLR performs memory management. Managed code cannot access memory directly. No pointers are allowed. This means that your code cannot inadvertently write over memory that does not belong to it, possibly causing a crash or other bad behavior.

The CLR can enforce strong security. One of the challenges of the software world of third party components and downloadable code is that you open your system to damage from executing code from unknown sources. You might want to restrict Word macros from accessing anything other than the document that contains them. You want to stop potentially malicious Web scripts. You even want to shield your system from bugs of software from known vendors. To handle these situations, .NET security includes *Code Access Security* (CAS).