MSIL 簡介

黃宣龍

asker_pachelbel@hotmail.com

何謂 MSIL

在.NET 平台中，為了達到跨語言甚至跨平台的可移植性，採取了和 Java 一樣的策略，也就是將程式碼先編譯成一種中介碼，再經由執行環境以 JIT 的方式編譯成可執行的原生碼(native code)並執行。相對於 Java 中的 byte code，.NET 平台則是將程式碼編譯成一種稱為 Microsoft Intermediate Language(MSIL)的中介碼。因此，當我們在.NET 平台上面開發程式時，不論我們使用的是 C#，VB.NET 還是 C++，都是先被轉換成 MSIL 再經由 CLR 編譯成原生碼並執行的。

MSIL 對我們有什麼意義

的確，對大多數的程式開發者而言，MSIL 通常都是交由編譯器自動產生，而且由於它的語法相對於 C#等高階語言顯得難以閱讀和除錯，也鮮少會有人直接使用 MSIL 來撰寫程式。既然如此，我們為何要去關心 MSIL 的細節呢？

事實上，您可以將 MSIL 當成 CLR 上的組合語言。透過閱讀 MSIL 我們可以瞭解編譯器對我們的程式碼做了什麼樣的處理，也可從中瞭解一些較為底層的實作細節，因此對於一個進階的.NET 平台程式開發者，掌握閱讀 MSIL 的能力是需要的。

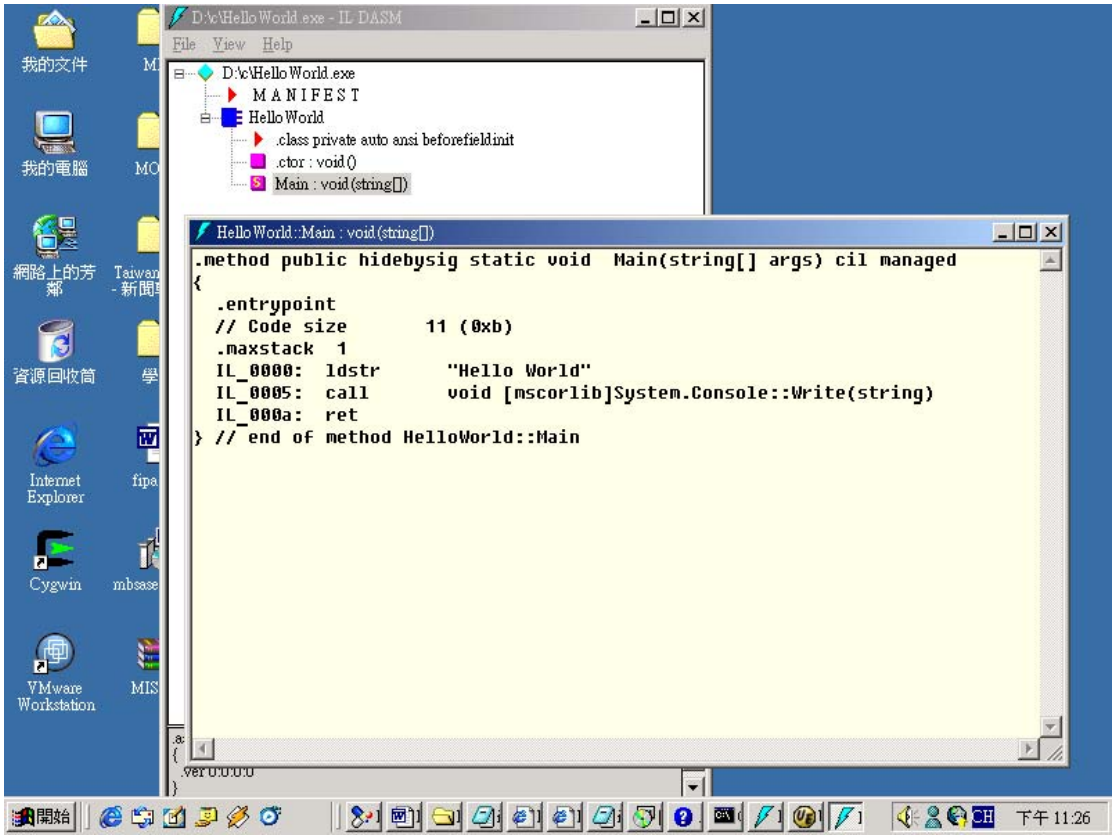
此外，透過 IL 碼，我們可以對程式碼進行反組譯工程。由於中介碼的結構較清楚，可讀性也比原生碼高的多，因此反組譯也相對得比較容易。事實上，對於 Java 的 byte code 就已經有很多反組譯的工具流傳於網路之上，而且能很精準的反組譯出原本的程式碼。

工欲善其事，必先利其器

當您安裝.NET Framework 時，也會同時安裝進一些內附的輔助工具，而現在我們要用到的是 ilasm.exe(MSIL Assembler)和 ildasm.exe(MSIL Disassembler)這兩個工具程式。由於在.安裝 NET Framework 時，預設會將這些工具所在的路徑加入系統的 path 變數中，所以您只要打開命令提示字元，鍵入 ilasm 和 ildasm 就可以使用這兩個內附的工具了。

Ilasm.exe 讓我們能由 il 檔產生 exe 檔，更精確的說法是一個 PE 檔(portable executable file)，或一個組件(assembly)。Ilasm 的用法如下：**ilasm** [*options*] *filename* [[*options*] *filename*...]。舉例來說，當您鍵入 ilasm HelloWorld /exe 時，意味著您要由一個名為 HelloWorld.il 的 IL 檔來產生一個名為 HelloWorld.exe 的 PE 檔。

而 ildasm 則是能讓您由 PE 檔中將 IL 碼分離出來，並提供了一個圖形介面讓您能方便的閱讀這些 IL 碼(見圖一)。



圖一

由 Hello World 開始閱讀 MSIL 碼吧

首先，我們使用 C#撰寫一個 Hello World 程式。程式碼如下：

```
using System;

class HelloWorld
{
    public static void Main(string[] args)
    {
            Console.Write(“Hello World”);
      }
}
```

然後我們使用 csc.exe 來編譯它，並得到一個 HelloWorld.exe 的 PE 檔。接下來我們使用 ildasm 來開啟這個 HelloWorld.exe(使用 File->Open，然後選取 HelloWorld.exe)，我們將會看到如圖一的畫面，其中白色背景色的視窗會顯示出這個 IL 檔的樹狀結構。在其上敲擊後，會顯示出一個以鵝黃色為底色的視窗，用來顯示其中的 IL 碼。讓我們在 Main:void(string[])上雙擊滑鼠，並取得其中的 IL 碼如下。

```
1.method public hidebysig static void  Main(string[] args) cil managed
2{
3  .entrypoint
4  // Code size       11 (0xb)
5  .maxstack  1
6  IL_0000:  ldstr      "Hello World"
7  IL_0005:  call       void [mscorlib]System.Console::Write(string)
8  IL_000a:  ret
9} // end of method HelloWorld::Main
```

我們可以發現，IL 碼的內容其實大多都能由字面理解。我們很容易可以看出這是一個公開(public)的靜態(static)方法(method)。其中方法的名稱，傳入值及傳回值，還有程式的進入點(.entrypoint)等等都是一目瞭然的。第六行的指令是將一個內容為 Hello World 的字串載入(load)堆疊，接下來第七行呼叫了 System.Console 的 Write 方法，而其中的[mscorlib]代表這個類別定義在

mscorlib.dll 這個檔裡。第八行的 ret 則是代表 return 的意思。

幾個常用的 MSIL 指令

接著介紹幾個較常出現的 MSIL 指令，詳細的 MSIL 指令說明您可以在 D:\Program Files\Microsoft.NET\FrameworkSDK\Tool Developers Guide\docs(請依照您的.NET Framework 安裝路徑自行修正)下的 Partition III CIL.doc 找到。
最常見到的應該是一群以 ld 和 st 開頭的指令，ld 代表載入(load)，而 st 則意味著儲存(store)。以 ld 開頭的指令都是用來將某些東西放入堆疊以讓 CPU 處理的，而 st 開頭的指令則是用來將堆疊上的變數存回到記憶體中。此外，接在 ld 或 st 後面的字元通常是用來描述其處理的變數型別。

Ldstr，這在剛剛的範例程式中有出現，是用來將字串型別的變數載入堆疊中，開頭的 ld 代表載入(load)的意思。
Ldc，用來載入數字型別的變數至堆疊中。用法如下：
ldc.i4.2　//i4 代表這是一個 4byte 的整數(integer)，而最後的 2 則是它的值。也就是說，這行指令會將一個內容為 2 的 4byte 整數放入堆疊之中。
ldc.r4　3.1415926　//r4 代表這是一個 4byte 的浮點數(float)，而最後面的 3.1415926 是它的值。也就是說，這行指令會將一個值為 3.1415926 的浮點數放入堆疊之中。

Stloc，用來將堆疊中的變數移回記憶體的區域變數之中，用法如下：
stloc.1　//用來將堆疊中最上面的數值移到區域變數 1 中，stloc.0 stloc.1 stloc.2 stloc.3 這四個指令分別是指前四個區域變數，若是要存到後面的

區域變數，要用 **stloc index** 語法(index 為 4 到 255)，也就是說，如果想要將數值載入到區域變數 6 中，則需要使用 stloc 6 這個指令。

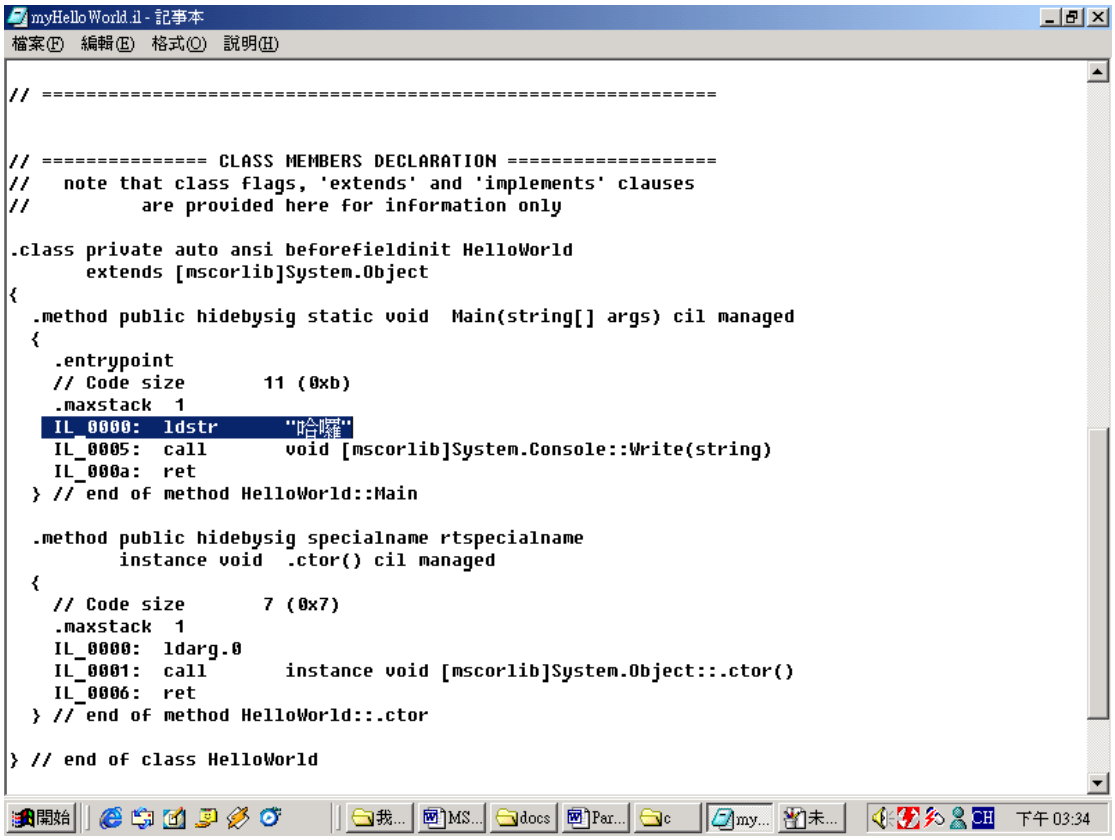
此外，newobj 用來實體化一個類別，newarr 用來實體化一個陣列，ldelem 和 stelem 則能用來操弄陣列。詳細的語法內容您都可以在文件中找到

除了看，可以改嗎？

現在你稍微能讀懂一些 IL 碼了，不過光是看實在不過癮，讓我們試著動手修改 IL 碼吧。假設您手上有一個以.NET 平台開發出的.exe 檔，而您想要將其中的英文訊息中文化，卻苦於沒有原始檔可改，那麼不妨試試由 IL 碼著手吧。讓我們以之前的 Hello World 為例，在我們用 ildasm 打開 HelloWorld.exe 後，使用 File->Dump 將它的 IL 碼存成一個檔案，在範例中，我們存成 myHelloWorld.IL，接著用 notepad 開啟，找到您之前在 ildasm 中看過的那行

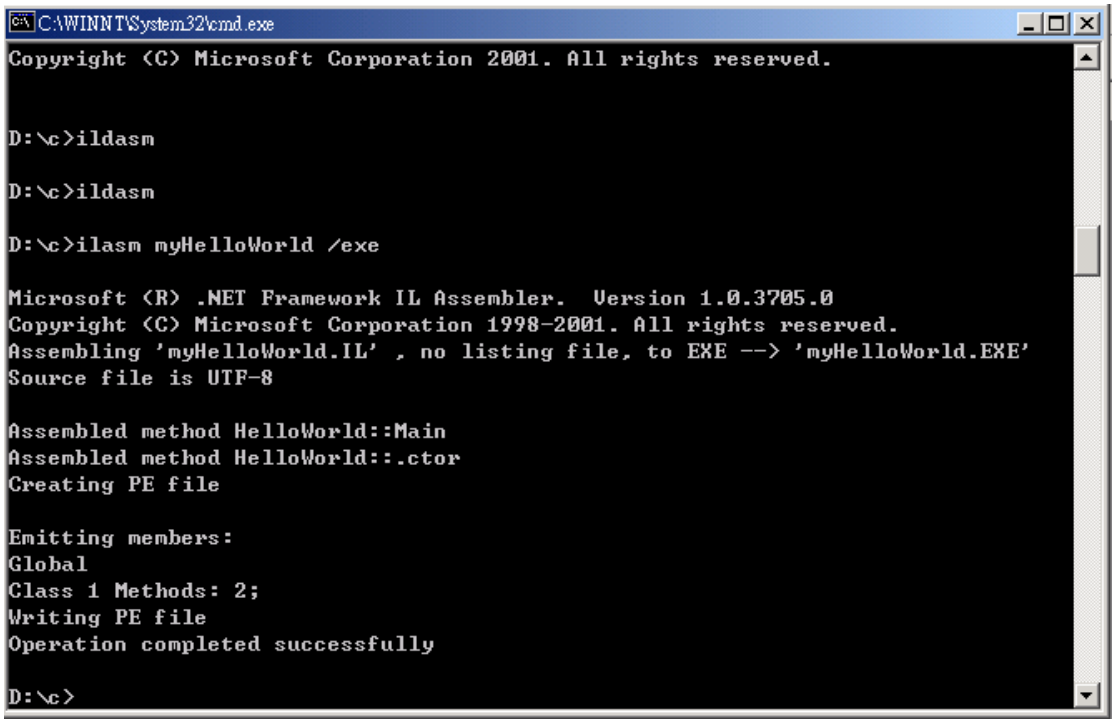
```
IL_0000:  ldstr      "Hello World"
```

並把其中的 Hello World 改成「哈囉」，然後存檔。(見圖二)



圖二

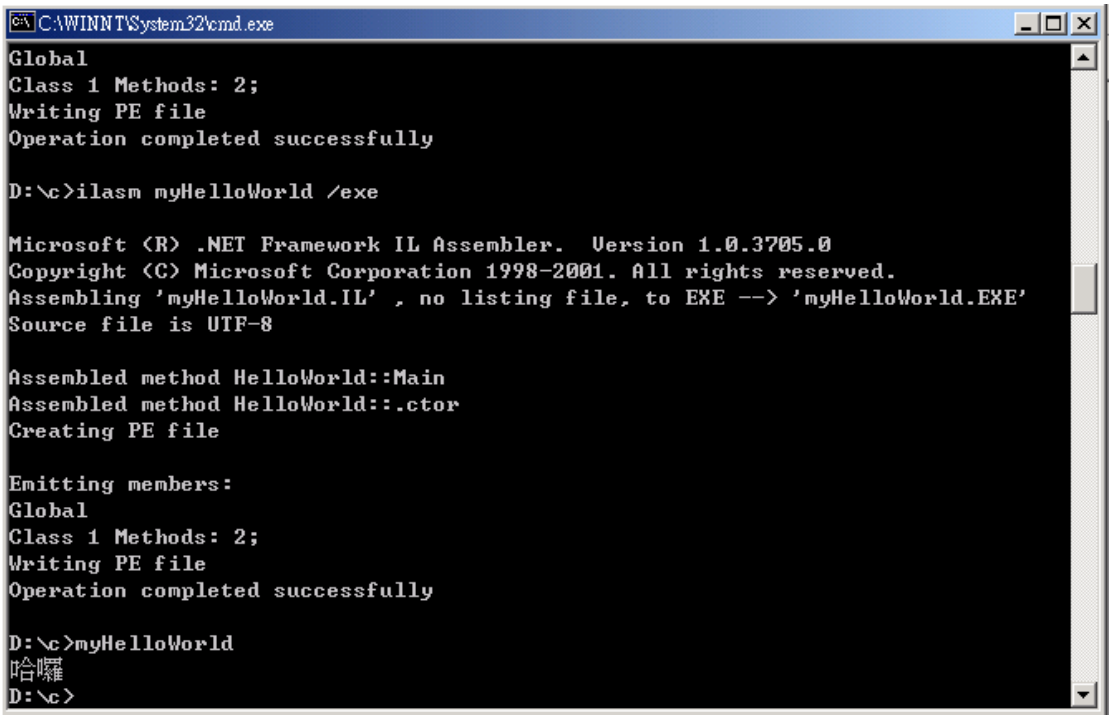
接著使用 ilasm 將這個 il 檔轉成.exe 檔，語法如下：ilasm myHelloWorld /exe



圖三

我們會得到如圖三的訊息，並得到一個

myHelloWorld.exe，執行看看吧。



圖四

執行 myHelloWorld 後我們會得到如圖四的結果，訊息已經由原本的「Hello World」變成我們填入的「哈囉」了。在這個範例中我們也可以發現，我們可以用 ilasm 和 ildasm 順暢的讓程式在 IL 檔和 PE 檔間互相轉換，這種性質我們稱之為”round-trip”。

結語

您覺得 MSIL 有趣嗎？不論如何，擁有 IL 檔的閱讀能力將能讓我們對.NET 平台上程式碼的運作方式有更深入的瞭解，並撰寫出更好的程式碼。在某些不得已的情況下，我們也可以透過 ildasm 和 ilasm 的協助，使用修改 IL 碼的方式來改變一個程式的行為。在這篇文章中，您應該會瞭解到何謂 MSIL，並學習使用 ildasm 來觀看 IL 檔並進行修改，接著使用 ilasm 將其轉換為可執行的 PE 檔。

參考資料

- Partition III CIL.doc
- ILAssemblyProg.doc
- http://msdn.microsoft.com/msdnmag/issues/01/05/bugslayer/bugslayer0105.asp
- http://www.devx.com/premier/mgznarch/vbpj/2001/05may01/df0501/df2_0501.asp

附註

本文已刊載於 .NET 電子雜誌 http://www.netmag.com.tw