Zery

让技术成为我们的一种能力，但不是所有能力

读懂IL代码就这么简单 (一)

一前言

感谢 @冰麟轻武 指出文章的错误之处，现已更正

对于IL代码没了解之前总感觉很神奇，初一看完全不知所云，只听高手们说，了解IL代码你能更加清楚的知道你的代码是如何运行相互调用的，此言一出不明觉厉。

然后开始接触IL，了解了一段时后才发现原来读懂IL代码并不难。进入正题

1.1 什么是IL

IL是.NET框架中中间语言（**Intermediate Language**）的缩写。使用.NET框架提供的编译器可以直接将源程序编译为.exe或.dll文件，但此时编译出来的程序代码并不是CPU能直接执行的机器代码，而是一种中间语言IL（Intermediate Language）的代码(来源百度)

1.2 为什么要了解IL

在很多时候不明白代码是如何操作时就可以通过IL指令来解释，比如，装箱，拆箱是否只是听别人说或者书上讲是怎么怎么实现的，自己是否证实过呢？了解IL指令你可清楚看到是每一步是如何处理的

1.3 怎么学IL

世上有个定律叫“二八定律”，80%的功能，只要用20%的技术就可以完成，但要完成另外20%可能就需要80%技术了，对于IL代码也是如此，有200多个指令，我们只需要用到其20%的指令就可以解决我们80%的问题了，所以我不会写太多，只是让大家能看懂普通的程序代码编译成IL代码后就行了，还有就是要多看，IL代码的每一条指令都是特定的意思，看得多了自然就懂了，当对自己代码有疑问时尝试看看它对应的IL代码，也许你会了解得更多。

IL指令大全 点这里

IL代码编译器 ILDasm 点这里

二 如何查看IL代码

2.1 步骤

1 编写代码并编译通过

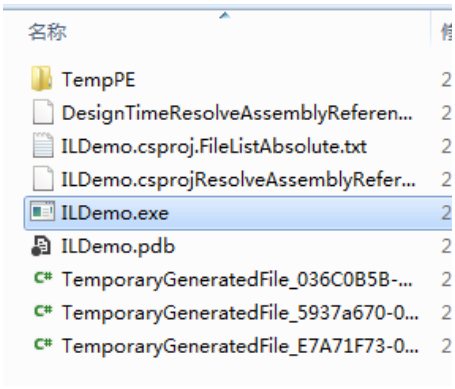
2 找到源文件的obj文件下的 .exe文件

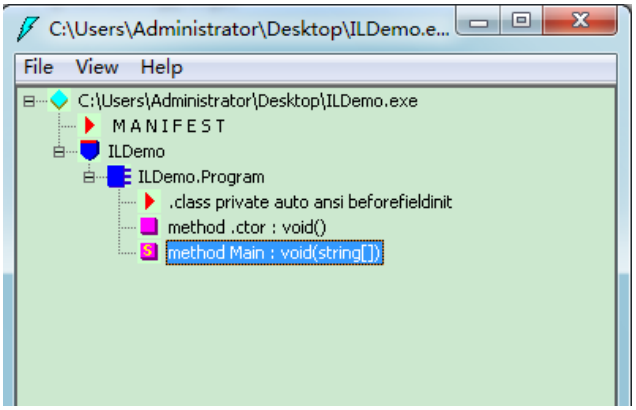
3 导入到ILDasm中反编译成IL代码

上图

1 -2步

3 导入到ILDasm中





ILDasm中图标含义

符号	含义
	更多信息
	命名空间
	类
	接口
	值类
	枚举
	方法
	静态方法
	字段
	静态字段
	事件
	属性
	清单或类信息项

三 如何读IL(大致了解)

以上步骤完成后我们就可以看到代码被编译后的IL代码，以下部份将会对每一条IL指令做详细的解释

随笔档案

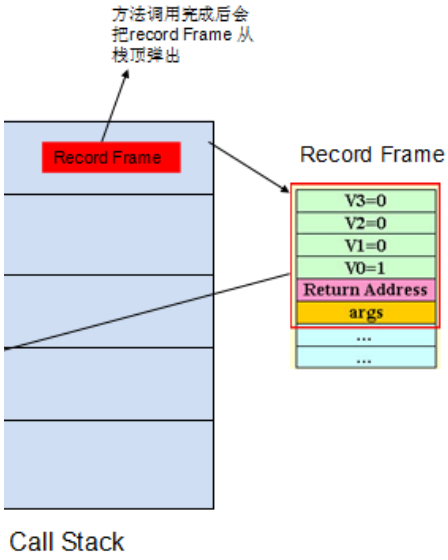
C#代码

```
1         static void Main(string[] args)
2         {
3             int i = 1;
4             int j = 2;
5             int k = 3;
6             Console.WriteLine(i+j+k);
7         }
```

IL代码

// Call Stack是一个栈，而Call Stack中的Record Frame则是一个局部变量列表，用于存储 .locals init (int32 V_0,int32 V_1,int32 V_2)初始化后的参数 V_0,V_1,V_2

因图中没有把Record Frame 标记出来，所以自己画了一张图



// Evaluation Stack 是一个栈 ldc.i4.2 这种指令都会先把值压入栈中等待操作

在第四段时大家可以理解得更清楚一点

另外@Learning hard 指出IL指令中第 9 11 13行容易让人误解值是从Record Frame中加载的

现强调IL指令中 第 9 11 13行的ldc.i4.1，ldc.i4.2，ldc.i4.3 执行这几条指令时 值是还没有加载到Record Frame中的，但是MSDN也没有指出从哪里加载

所以只能根据个人的想法解释，程序在编译后值类型数据会存在线程栈中，所以我认为此时的9 11 13行的值是从线程栈中取的

```
1  .method private hidebysig static void  Main(string[] args) cil
managed
2 {
3   .entrypoint  //程序入口
4   // Code size       19 (0x13)
5   .maxstack  3  //定义函数代码所用堆栈的最大深度，也指Evaluation Stackk中最多能同时存在3个值
```

2017年9月(3)
2017年7月(2)
2017年6月(3)
2017年5月(1)
2017年4月(5)
2017年2月(1)
2017年1月(1)
2016年12月(1)
2016年11月(3)
更多

文章档案
2016年11月(1)

阅读排行榜
1. 读懂正则表达式就这么简单(286069)
2. HTTPS 原理解析(67065)
3. 读懂IL代码就这么简单 (一)(53793)
4. Centos7 ping 未知的名称或服务 DNS 配置问题(49040)
5. C#操作XML方法集合(44638)
6. C#获取CPU占用率、内存占用、磁盘占用、进程信息(29194)
7. Linux jar包 后台运行(27601)
8. IL指令详细(25275)

```
 6    //以下我们把它看做是完成代码中的初始化
 7    .locals init (int32 V_0,int32 V_1,int32 V_2) //定义 int 类型参数
V_0,V_1,V_2  (此时已经把V_0,V_1,V_2存入了Call Stack中的Record Frame中)
 8    IL_0000:  nop //即No Operation 没有任何操作，我们也不用管它

 9    IL_0001:  ldc.i4.1    //加载第一个变量"i"的值            (压入
Evaluation Stack中)
10    IL_0002:  stloc.0     //从栈中把"i"的值弹出并赋值给Record Frame中第0个
位置(V_0)
11    IL_0003:  ldc.i4.2    //加载第二个变量"j"的值            (压入
Evaluation Stack中)
12    IL_0004:  stloc.1     //从栈中把"j"的值弹出并赋值给Record Frame中第1个
位置(V_1)
13    IL_0005:  ldc.i4.3    //加载第三个变量"k"的值             (压入
Evaluation Stack中)
14    IL_0006:  stloc.2     //从栈中把 "k"的值弹出并赋值给Record Frame中第2个
位置(V_2)
15
16     //上面代码初始化完成后要开始输出了，所以要把数据从Record Frame中取出
17
18    IL_0007:  ldloc.0     //取Record Frame中位置为0的元素(V_0)的值("i"的
值)并压入栈中    (相当于Copy一份值Call Stack中V_0的值。V_0本身的值是不变的)
19    IL_0008:  ldloc.1     //取Record Frame中位置为1的元素(V_1)的值("j"的
值)并压入栈中         (同上)
20    IL_0009:  add         // 做加法操作
21    IL_000a:  ldloc.2     // 取出Record Frame中位置为2的元素(V_2)的值
("k"的值)并压入栈中
22    IL_000b:  add         // 做加法操作
23    IL_000c:  call       void
[mscorlib]System.Console::WriteLine(int32) //调用输出方法
24    IL_0011:  nop
25    IL_0012:  ret         //即为  return  标记 返回值
26 } // end of method Program::Main
```

指令详解

.maxstack:评估堆栈(Evaluation Stack)可容纳数据项的最大个数

.locals init (int32 V_0,int32 V_1,int32 V_2)：定义变量并存入Call Stack中的Record Frame中

nop:即No Operation 没有任何操作，我们也不用管它，

ldstr.:即Load String 把字符串加压入Evaluation Stack中

stloc.：把Evaluation Stack中的值弹出赋值到Call Stack中的Record Frame中

ldloc.:把Call Stack中的Record Frame中指定位置的值取出(copy)存入 Evaluation Stack中 以上两条指令为相互的操作stloc赋值，ldloc取值

call: 调用指定的方法

ret: 即return 标记返回

每一句IL代码都加了注释后，是不是觉得IL代码其实并不难呢，因为它的每一条指令都是固定的，你只要记住了，看IL代码就比较轻松了。

四 如何读IL(深入了解)

4.1 提出问题

有了上面的一点IL基础后，现在我们来深入一点点，

9. Mysql 查看死锁，解除死锁 方式(17713)

10. 做为技术人员为什么要写博客(15433)

评论排行榜

1. 读懂IL代码就这么简单 (一)(104)

2. 2014年读书计划(102)

3. 做为技术人员为什么要写博客(77)

4. 读懂正则表达式就这么简单(53)

5. 采集博客园文章，用瀑布流+无限滚动展示(附源码)(49)

6. 百度广告 高亮 Chrome插件(附源码)(48)

7. 文件夹管理工具(MVC+zTree+layer)(附源码)(44)

8. 委托 你怎么看？(37)

9. 读懂IL代码就这么简单(二)(34)

10. 常用加解密方法汇总 工具 (附源码)(33)

推荐排行榜

1. 读懂正则表达式就这么简单(178)

2. 读懂IL代码就这么简单 (一)(167)

3. 做为技术人员为什么要写博客(138)

4. 常用加解密方法汇总 工具 (附源码)(100)

5. 让数据决策你的行为--拉勾网数据分析(87)

有如下几个问题：

1 当 ldc.i4.1 这一指定加载 “i” 这个变量后并没有马上赋值给Record Frame中的元素，而是要执行 stloc.0 后才赋值，那没赋值前是存在哪里的呢?

2 ldloc.0 把元素取出来后，存在哪里的?

3 add操作完成后值存在哪里?

4.2 概念引入

Managed Heap:：這是動態配置（Dynamic Allocation）的記憶體，由 Garbage Collector（GC）在執行時自動管理，整個 Process 共用一個

Managed Heap(**我理解为托管堆，存储引用类型的值**)。

Evaluation Stack:這是由 .NET CLR 在執行時自動管理的記憶體，每個 Thread 都有自己專屬的 Evaluation Stack(**我理解为类似一个临时存放值类型数据的线程栈**)

Call Stack:這是由 .NET CLR 在執行時自動管理的記憶體，每個 Thread 都有自己專屬的 Call Stack。每呼叫一次 method，就會使得 Call Stack 上多了一個 Record Frame；呼叫完畢之後，此 Record Frame 會被丟棄(**我理解为一个局部变量表，用于存放.locals init(int32 V_0)指令的参数值如：V_0**)

4.3 IL指令详解

对三个名词做解释后现在我们再来仔细看看执行IL指令时，对应的变量是如何存放的

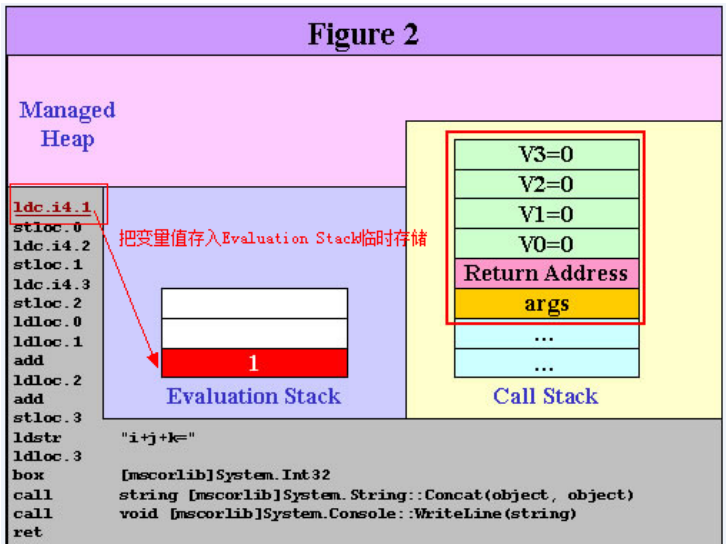
IL_0001: ldc.i4.1 //加载第一个变量i

首先对 ldc.i4.1 做下细解：变量的值为1 时IL指令就是ldc.i4.1 ,变量值为2 时IL指令就是ldc.i4.2，依此类推一直到ldc.i4.8

当为-1 时IL指令为ldc.i4.M1,当超过8时就是一个统一指令 ldc.i4.S

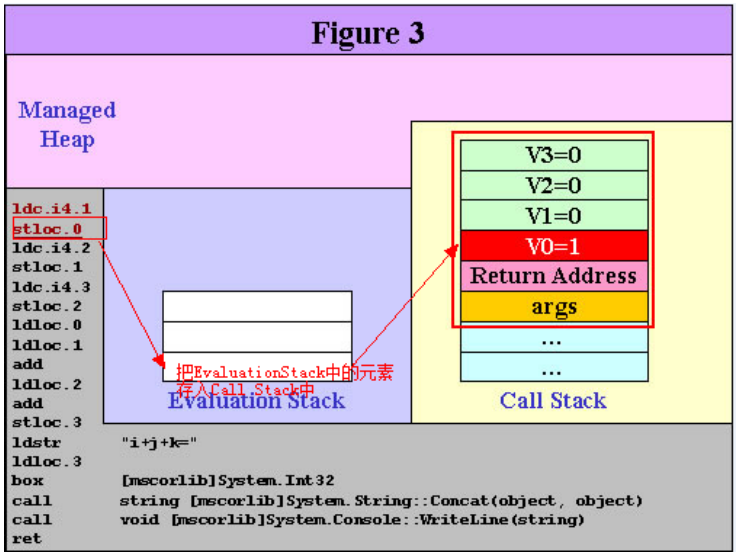
IL_0001: ldc.i4.1 //加载第一个变量i

当执行这一条指令时会把变量i 的值压入Evaluation Stack中做临时存储



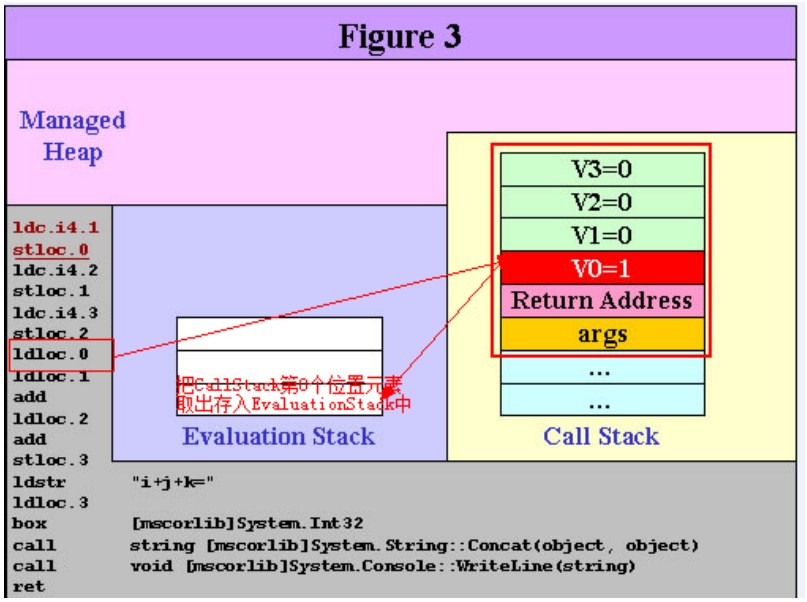
IL_0002: stloc.0 //把i 赋值给Call Stack中第0个位置

当执行这一条指信时会把Evaluation Stack 中的 i 弹出赋值给Record Frame中的第0个位置



IL_0007: ldloc.0 //取出Record Frame位置为0的元素 (i)

当执行这条指令时会将 Record Frame中的位置为0的元素的值取出(copy)压入 Evaluation Stack 等待做加法的指令 Add



IL_000b: add // 做加法操作

add这一操作完成后，会把结果存在Evaluation Stack中等待下一步的指令操作

4.4 问题回答

以上内容看完开始的问题相应也解决了

1 ldc.i4.1 把值取出来后先存在 Evaluation Stack中 执行了stloc.0 后才会存入Record Frame中指定的元素中

2 ldloc.0 把取出来后也是先压入 Evaluation Stack 等持指令

3 add 操作完成后值是暂存于 Evaluation Stack中的

以上把IL指令是如何操作内存中的值做了一点很基本的介绍，让大家在了解IL指令时，知道是如何操作内存中的值的。我想对于理解IL指令或许更透彻一点。

五 总结

这一篇只写了IL中最基本的几个指令，然后讲解了IL指令是如何操作内存中数据的。古人云：水得一口一口喝，路得一步一步走，步子迈得大了容易扯着蛋，慢慢来内容虽然少了点，但是还会有下篇的。下一篇还是会写IL的一些基本指令，我会结合我自己的理解，尽量把文字写得通俗一点，让大家更容易理解。

另外本人水平有限，难免会有理解错误的地方，如有发现，请指出！我会马上修改，以免误导他人。

如果您觉得本文能给您带来一点收获不妨点下 推荐 让更多的人了解IL，您的推荐是我源源不断的写作力
如果觉得我的博客还不错，那就关个注吧~

成长在于积累

参考资料：《你必须知道的.net》,MSDN

Record Frame

分类： C#基础知识

标签： IL 系列

« 上一篇： C#操作XML方法集合
» 下一篇： 读懂IL代码就这么简单(二)

posted @ 2013-10-15 08:56 Zery 阅读(53794) 评论(104) 编辑 收藏 举报

编辑推荐:
· 流量录制与回放技术实践
· 熟悉而陌生的新朋友——IAsyncDisposable
· 对象池在 .NET (Core)中的应用[3]: 扩展篇
· 奇思妙想 CSS 3D 动画 | 仅使用 CSS 能制作出多惊艳的动画?
· 一个测试工程师的成长复盘

最新新闻:
· 字节实习生开发的AI，实现4K60帧视频实时抠图，连头发丝都根根分明（2021-08-31 13:05）
· 关键指标下行，有赞的故事不好讲了（2021-08-31 12:50）
· 快递小哥还差70万，加价1毛钱能堵上这个缺口吗？（2021-08-31 12:38）
· 暂缓IPO，网易云音乐该“抑郁”了（2021-08-31 12:24）
· 途牛：一直被唱衰，就是没倒下（2021-08-31 12:10）
» 更多新闻...