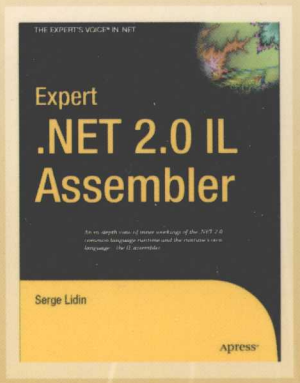
TURING 图灵程序设计丛书 微软技术系列

Apress®

Expert .NET 2.0 IL Assembler

.NET探秘 MSIL权威指南

[加] Serge Lidin 著
包建强 译



- 深入洞悉.NET技术内幕
- MSIL唯一权威指南，Amazon五星巨著
- 学习和参考皆宜

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 微软技术系列

Expert .NET 2.0 IL Assembler

.NET探秘
MSIL权威指南

[加] Serge Lidin 著
包建强 译

人民邮电出版社
北京

图书在版编目（CIP）数据

.NET探秘：MSIL权威指南 /（加）利丁（Lidin, S.）著；包建强译. —北京：人民邮电出版社，2009.9
（图灵程序设计丛书）
ISBN 978-7-115-20176-8

Ⅰ. N… Ⅱ. ①利…②包… Ⅲ. 软件工具－程序设计
Ⅳ. TP311.56

中国版本图书馆CIP数据核字（2009）第133398号

内 容 提 要

本书是IL汇编语言的权威之作，深入地讲解了IL的全部内容，是.NET底层开发人员必备的参考书，对于从概念设计到实现和维护等软件开发所有阶段都很有价值。本书内容基于.NET 2.0版本，同时，考虑到.NET 3.0/3.5在CLR/IL上没有作任何改动，且即将推出的.NET 4.0也只是在CLR上稍作修改而并没有涉及IL语言，所以，本书对于目前各个版本的.NET Framework而言都是适用的。

本书适合所有.NET开发人员、讲师、研究人员阅读。

图灵程序设计丛书

.NET探秘：MSIL权威指南

◆ 著 [加] Serge Lidin
译 包建强
责任编辑 傅志红
执行编辑 陈兴璐

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京顺义振华印刷厂印刷

◆ 开本：800×1000 1/16
印张：27.25
字数：650千字 2009年9月第1版
印数：1－3 000册 2009年9月北京第1次印刷

著作权合同登记号 图字：01-2009-2903号

ISBN 978-7-115-20176-8

定价：89.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

译 者 序

不要惊讶，坐在你面前娓娓而谈的，并不是一个技术Geek，而是流连于淮海路迤逦风光的性情中人。编程只是我诸多业余爱好中的一项。当然，有时会稍微热情些，比如说，从狄更斯的《双城记》中能联想到设计模式的精华。

一直在酝酿一篇有些不同的序言，即开头没有诸多名人的溢美之词，结尾也没有感谢父母家人的肺腑之言，我只想谈一谈阅读本书时所要注意的问题，以及本书的不足之处。

作者Serge Lidin是位慈眉善目的老伯，我们曾坐在微软总部的长椅上攀谈良久。他曾经是微软IL小组的成员，所写的内容都是第一手资料，所以写起本书来当仁不让，章节划分相当合理。但是，正如作者所说，他不善笔墨，所以本书读起来非常枯燥。当然，CLR底层本身就是非常枯燥的，要把它写得精彩生动，很难做到。除非借助C#来讲解IL和元数据，才会吸引人，如某位印度作家编著的*C# to IL*。建议大家先去阅读后面这本书，它只是基于.NET 1.1的，没有涉及泛型，内容没有前者丰富，但是，它要简单得多。

言归正传。本书是学习和研究IL和元数据的首选图书，它可以作为一本技术参考手册，睡觉前翻翻也是不错的。本书并没有介绍任何IL性能调优的方法，而这正是开发人员所关心的，这不能不说是一个遗憾。

IL和元数据，也就是PE结构体，是静态的内容。对于运行期间的动态执行过程，完全不是这么回事。本书没有涉及后面这部分内容，毕竟，这超出了本书的范围。有兴趣的读者可以去研究Rotor，可以参考*Shared Source CLI Essentials*一书。

作者非常认真负责。我们书信往来半年之久，修正了原书中的一些错误。同时，统一了.NET和汇编语言中的中文术语，其中，参考了侯捷先生和李建忠先生译作中的诸多神来之笔。此外，还在字里行间添加了大量的译注，有助于读者理解。

本书分为六个部分共19章。第1章～第3章是对IL语言的一个快速入门，通过一个IL小程序对这门技术进行快速的概览，强烈建议读者首先阅读这一部分，根据理解的层次以决定是否继续研读下去。第4章是全书中最乏味的一章（至少我是这么认为的，在翻译的过程中异常痛苦），毕竟PE文件头和CLR头的概念并不常见。第5章是元数据概览，这是全书的核心。接下来第6章～第12章，以及第15章～第17章是对所有45个元数据的详细介绍，是全书的精华所在。第13章是IL语句的集大成之作，可以作为参考手册日常查阅，不建议读者一页页学习。第14章则深入浅出地介绍了IL中的异常处理机制，非常重要。第18章是写给从事COM互操作的技术人员的，对于只专注于.NET平台的朋友，可以略过这一章。第19章，也就是双向解析，老实说，我是觉得意义不

大，当然这要看领域里了，对于那些从事.NET性能调优的朋友，这一章还是有很大参考价值的。5个附录都极具参考价值，建议你把附录部分独立装订成册常备案头。请不要嫌我啰嗦，要有选择地阅读本书。

如果你还是一位.NET新手，还是把它放回到书架上吧，3年之内你应该不会使用到这些技术。5年之后呢？那也未必。你起码要阅读一遍《深入理解.NET》[①]或者《.NET本质论》。否则，本书就是你噩梦的开始，轻则迷惘不前，重则走火入魔，甚至结束程序员的生涯。好吧，没那么夸张啦，但副作用还是有一点的。然而，即使你的修为达到了上述要求，在阅读本书时也要摆脱高级语言（如C#）的束缚，而重新建立一套CLR世界中的逻辑体系。“万物皆标记”，什么时候读者能领悟到这一层面，也就到达“庖丁解牛”的境界了。

是不是显得有些悲观了？好吧，下面说一点振奋人心的言语。

由于本书内容非常偏门，以至于国内出版社少有问津，非常感谢图灵公司能将其引进国内。我在翻译期间，参考了国内外大量有价值的关于IL的技术博客。

国外有：

Vijay Mukhi的博客：http://www.vijaymukhi.com/documents/books/。这就是我前面提到的那位印度作家的网站，上面有3本他参与编写的IL图书的电子版。

- *C# to IL*，对应子目录ilbook，这是最好的一本，可惜是1.1版本的。
- *Metadata*，对应子目录metadata，说的是如何用C#语法解读dll或exe文件。
- *IL Disassembler*，对应子目录ildasm。

国内有：

- Anders Liu（刘彦博）的博客：http://www.cnblogs.com/andersliu。他的最有名的文章莫过于《透过IL看C#》系列。
- 装配脑袋(施凡)的博客：http://www.cnblogs.com/ninputer。他在CLR上造诣极深，甚至自己编写新的.NET语言编译器，最喜欢VB.NET语言。
- Filer Lu（卢小海）的博客：http://www.cnblogs.com/flier。他是研究IL的先驱，他发表的最著名的文章是2001～2002年间在绿盟和程序员杂志上的《MS.NET CLR扩展PE结构分析》系列。

综上所述，国内在IL这方面的沉淀已经很深了，完全可以原创出一本通俗易懂且又博大精深的IL图书。让我们拭目以待，将IL进行到底。我很有信心。

① 英文版已由人民邮电出版社出版。——编者注

前　言

为什么要写这本书呢？

说实话，对此，我是责无旁贷的。这本书是我早期写的一本书——*Inside Microsoft .NET IL Assembler*（图书上市于2002年初，在.NET CLI 1.0版本发布1个月后）的修订和扩展。因此，时隔4年多，更加强大的.NET CLI 2.0版本公布之际，我的创作意图就显而易见了。而且，必须有人来写一本.NET CLI的内部工作原理的书，我只能当仁不让了。

.NET领域与其他信息技术领域类似，如同一个巨大的倒金字塔，根植于一种底层核心技术之上。.NET领域所依赖的底层就是CLR。CLR将IL二进制代码转换为特定于平台（本地）的机器代码来执行。位于CLR之上的是.NET Framework类库、编译器以及像Visual Studio这样的集成开发环境，再往上才是从工具到面向终端用户的应用开发层。这个金字塔正在迅速地向上蔓延，愈高而弥宽。

准确地说，这本书并不是关于CLR的。虽然CLR只是.NET这座金字塔的底层核心技术，但是CLR主题是如此庞大，以至于不可能在一本书的篇幅内对其进行详细描述。本书着重讨论另一个重要的主题：.NET IL汇编语言（ILAsm）。IL汇编语言是一门低级语言，专门用于描述CLR的各种特性。只要CLR具有某种特性，ILAsm就必须能够对其进行表述。

ILAsm不同于高级语言而类似于其他汇编语言，它是平台驱动而非概念驱动的。一门汇编语言通常会与底层平台之间存在精确的语言映射，对于ILAsm而言，这个底层平台就是CLR。事实上，这是一种非常精确的映射，甚至可以用这种语言来描述ECMA/ISO标准化文档中涉及.NET CLI运行环境的各个方面（ILAsm本身作为CLI的一部分，也是这个标准化文档的一个主题）。由于这种紧密的映射，如果不涉及底层平台的大量细节，也就不可能描述相应的汇编语言。所以，从很大程度上讲，这本书归根到底还是关于CLR的。

IL汇编语言在.NET开发者中非常流行。我并不是说所有的.NET开发者都更喜欢使用ILAsm编程，而不喜欢使用Visual C++/CLI、C#或者Visual Basic.NET。但是所有的.NET开发者都或多或少使用过IL反汇编器，而且许多人还会经常使用它。无论.NET开发者所偏好的语言和从事的开发领域是什么，在他们的计算机屏幕上都会有一道青色的闪电（IL反汇编器的图标）。而IL反汇编器的文本输出正是ILAsm源代码。

事实上所有基于.NET编程的书籍，只要是专注于高级编程语言的，如C#或者Visual Basic，或者是ADO.NET这样的技术，都会在有些时候提到IL反汇编器，把它作为一种可选的反汇编工具，用来分析.NET托管可执行程序的内部情况。但是这些书籍都没有充分说明这些反汇编文本

的含义以及如何解释它们。对这些书籍来说，这种选择是可以理解的，因为对元数据结构和IL汇编语言的详细描述是另外一个独立的话题。

现在，读者可能已经明白我责无旁贷写作此书的意义了。必须有人来承担这个任务，而我既然负责设计和开发IL汇编器以及IL反汇编器，就有义务善始善终地展示这些技术。

ILAsm 的历史

IL汇编器和IL反汇编器的第1个版本（分别名为Asm和DASM）由Jonathan Forbes在1998年初开发成功。当前的语言与它最初的形式已经有了很大的差异，二者所具有的唯一显著的共性就是伪指令关键字中的前导点号。这些汇编器和反汇编器最初是作为纯粹的内部工具而构建的，方便了正在进行的CLR开发，它们在CLR开发小组中得到了相当广泛的使用。

1999年初，Jonathan离开了CLR团队，把这些汇编器和反汇编器都交给了Larry Sullivan，Larry所领导的开发团队有一个很有意思的名字：CROEDT（Common Runtime Odds and Ends Development Team）。在那一年的4月，我加入了这个团队，Larry把这些汇编器和反汇编器转交给了我。当1999年5月CLR的alpha版本出现在技术预览会上的时候，这些汇编器和反汇编器引起了人们极大的关注，随后公司要求我改进这个工具，将其发展到产品的水平。在Larry、Vance Morrison和Jim Miller的帮助下，我完成了这项工作。因为这些工具当时还只是内部组件，所以我们（Larry、Vance、Jim和我）从根本上对这个语言进行了重新设计——并不是只将其作为工具实现。

主要的突破发生在1999年的下半年，当时IL汇编器的输入和IL反汇编器的输出已经可以实现足够的同步，进而可以获得有限的双向解析（round-tripping）。双向解析意味着用户可以获得由特定语言编译的托管（IL）可执行程序，对其进行反汇编、增加或者修改一些ILAsm代码，然后将其重新汇编到经过修改的可执行程序中。双向解析技术开创了新纪元，此后不久，它就开始应用于微软及其合作伙伴的特定产品的开发过程中。

大约在同一时期，使用ILAsm作为基本语言的第三方.NET编译器也已经开始出现。人所共知的可能就是富士通的NetCOBOL，它在2000年7月的专业开发者会议上大出风头，这次会议向开发者们发布了最初的CLR的pre-beta版本并附带.NET Framework类库、编译器和工具。

自从2000年底beta 1版本发布以来，IL汇编器和IL反汇编器的功能日渐完善，它们已经可以反映元数据和IL的所有特性，支持完整的双向解析，而且能够将自身的改变与CLR的改变保持同步。

ILAsm 的推进

现如今IL汇编器越来越多地应用于编译器和工具实现、教育以及学术研究。以下这些编译器，既有纯学术项目又有工业强度[①]系统，均生成ILAsm代码作为输出，并让IL汇编器负责产生托管的可执行体。

- Ada#（科罗拉多美国空军研究院）
- Alice.NET（德国萨尔布吕肯州萨尔兰德大学）

① industrial strength，表示其强健和可靠程度可以胜任工业级的考验，这里翻译为工业强度。——译者注

- Boo（codehaus.org）
- NetCOBOL（富士通公司）
- COBOL2002 for .NET Framework（NEC/日立）
- NetExpress COBOL（Microfocus）
- CommonLarceny.NET（波士顿东北大学）
- CULE.NET（CULEPlace.com）
- Component Pascal（澳大利亚昆士兰理工大学）
- Fortran（Lahey/富士通）
- Hotdog Scheme（芝加哥西北大学）
- Lagoona.NET（加州大学欧文分校）
- LCC（ANSI C）（微软雷蒙德研究院）
- Mercury（澳大利亚墨尔本大学）
- Modula-2（澳大利亚昆士兰理工大学）
- Moscow ML.NET（丹麦皇家兽医农业大学）
- Oberon.NET（苏黎士瑞士联邦理工大学）
- S#（Smallscript.com）
- SML.NET（英国剑桥微软研究院）

IL汇编器和IL反汇编器协同工作产生了大量有趣的工具和技术，这些都基于“创造性的双向解析”的托管可执行体：反汇编—文本处理—重新汇编。例如，Preemptive Software（一家以面向Java和.NET混淆和代码优化而著名的公司）就在此基础上建造了自己的DotFuscator系统。DotFuscator是一款商业的、行业实用的混淆和优化系统，在市场上非常出名。我会在第19章讨论另外一些有趣的“创造性的双向解析”的应用程序示例。

实际上，所有关于.NET编程的大学课程在一定程度上都使用到了ILAsm。（这些课程开发者还能怎样展示.NET托管可执行体的内部呢？）一些课程是完全基于ILAsm的，如Regeti Govindarajulu博士在印度海得拉巴的国际信息技术研究所开发的课程；Andrey Makarov、Sergey Skorobogatov和Andrey Chepovskiy博士在俄罗斯莫斯科的罗蒙诺索夫国立大学和包曼工业大学开发的课程。

读者对象

本书面向所有的.NET开发人员，因为他们在非常高级的层次上工作，所以可能需要考虑程序的编译结果，或者希望分析所编写程序的最终结果。这些读者可以在本书中找到解释反汇编文本以及元数据结构概要所需的信息，这可以帮助他们开发出更有效率的编程技术。

因为分析反汇编信息和元数据结构对于评估任何面向.NET的编译器的正确性和效率都十分关键，所以本书也非常适用于.NET编译器的开发者。还有一些目前数量较少但在逐渐增加的读者将发现本书对他们相当有帮助，包括直接使用IL汇编语言的开发人员，例如，将ILAsm作为中间步骤的编译器开发人员、策划多语言项目的开发人员，以及希望驾驭CLR（这种能力无法从高级语言中获得）的开发人员。

最后，本书对于软件开发的所有阶段——从概念设计到实现和维护都很有参考价值。

组织结构

第一部分“快速入门”利用一个简单的示例程序对ILAsm和CLR的特性进行快速的概述。这个概述并不完善，主要用来勾画CLR和ILAsm语言的轮廓。

接下来的部分将会使用自底向上的方式，详细讨论CLR的特性以及相应的ILAsm结构。第二部分“底层结构”将会描述托管可执行文件的结构以及通用的元数据组织。第三部分“基本组件”专门介绍构成应用程序的必备基础组件：程序集、模块、类、方法、字段以及相关主题。第四部分“深入执行引擎”将会带领用户深入讨论执行引擎，分析IL指令的执行以及托管异常处理。第五部分“特殊组件”将会讨论其他组件的元数据表示和使用：事件、属性以及自定义特性和安全特性。第六部分“互操作”将会描述托管代码和非托管代码之间的互操作，并且讨论IL汇编器和IL反汇编器在多语言项目中的实际应用。

本书的5个附录提供了有关ILAsm语法、元数据组织以及IL指令集和工具特性的参考，包括IL汇编器、IL反汇编器和离线元数据有效性检查工具。

致谢

首先我要感谢与我合作出版本书的Apress出版社编辑团队：Ewan Buckingham、Sofia Marchant、Kim Wimpsett和Laura Cheu。我很荣幸能与这样一个专业的团队合作，这是件很愉快的事情。

我还要感谢我的同事Jim Hogg和Vance Morrison，他们是本书主要的技术审稿人。Jim在CLR小组工作过很长一段时间，并且是制定.NET CLI的ECMA/ISO标准的核心人员。Vance在1998年CLR小组成立伊始就加入了，他领导了JIT编译器小组很长一段时间，并且在IL汇编器方面对我的帮助很大。Jim和Vance尽其所能地对本书的初稿提供了非常有价值的反馈。

当然，我还要感谢所有帮助过我写作本书（包括本书的第一版）的同事们，感谢他们解答了我的问题并和我一起深入地探讨规范文档和源代码：Larry Sullivan、Jim Miller、Bill Evans、Chris Brumme、Mei-Chin Tsai、Erik Meijer、Thorsten Brunklaus、Ronald Laeremans、Kevin Ransom、Suzanne Cook、Shajan Dasan、Craig Sinclair等。

目　录

第五部分 特 殊 组 件

Part 1 第一部分

快速入门

本部分内容

- 第 1 章　简单示例
- 第 2 章　代码增强
- 第 3 章　使代码更简单

第1章 简单示例

本章概要介绍ILAsm——MSIL汇编语言。（MSIL表示微软中间语言，本章稍后将会讨论。）本章先讨论一个相对简单的ILAsm程序，随后提出了一些修改意见，展示了如何用ILAsm表示.NET程序设计中的概念和元素。

本章并没有讲如何用ILAsm编写程序，但是会帮助用户理解IL汇编器（ILASM）和IL反汇编器（ILDASM）做了什么，以及如何通过这些理解并借助这些工具来分析.NET程序的内部结构。你还将了解到一些发生在CLR中的神秘而有趣的事件——相当吸引人，希望这能促使你读完本书的其他部分。

注解 简单起见，在本书中将IL汇编语言简写为ILAsm。不要将其与ILASM相混淆，后者是IL汇编器（也就是ILAsm汇编器）在.NET文档中的简写形式。

1.1 CLR基础

.NET CLR不仅是.NET诸多概念中的一个，更是.NET的核心。（注意，有时候我会将CLR称为运行库。）我们将重点关注.NET中真正发生行为的部分——CLR，而不是.NET平台的完全描述。

注解 关于.NET及其组件的总体结构的精彩讨论，参见David S.Platt所著的*Introducing Microsoft .NET, Third Edition*（Microsoft Press, 2003），以及Tom Archer和Andrew Whitechapel合著的*Inside C#, Second Edition*（Microsoft Press, 2002）。

简而言之，CLR是供.NET应用程序使用的运行时环境。它在.NET应用程序和底层操作系统之间提供了一个操作层。从原理上讲，CLR类似于GBasic这样的解释型语言的运行库。但是，这种相似性仅仅只是原理上的，CLR并不是解释器。

由面向.NET的编译器（如Microsoft Visual C#、Microsoft Visual Basic.NET、ILAsm以及其他编译器）生成的.NET应用程序，表现为一种抽象的中间形式，独立于原始的编程语言、目标机器以及它的操作系统。正因为它们表现为这种抽象的形式，所以使用不同的语言写成的.NET应