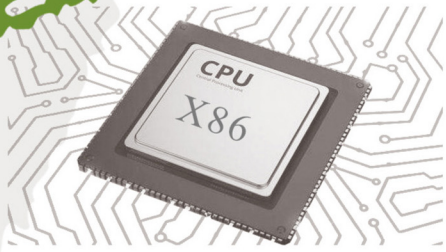
国家级精品课程主教材

X86汇编语言程序设计

许向阳　编著



華中科技大學出版社
http://www.hustp.com

x86 汇编语言程序设计

许向阳　编著

华中科技大学出版社
中国·武汉

内 容 简 介

本书立足于目前使用最为广泛的 Intel x86-32 和 x86-64 系列 CPU、Windows 操作系统及 Visual Studio 2019 开发平台，从汇编语言这种最直观和最直接的角度，揭示计算机工作的基本原理、C 语言语句和函数的处理过程、程序优化的技巧。

全书共分为 19 章。前 5 章介绍了汇编语言程序设计的基本知识，包括 CPU、内存、寻址方式和常用机器指令；第 6 章至第 11 章介绍了 x86-32 位控制台应用程序设计，包括顺序和分支、循环、子程序设计、多模块程序设计等；第 12 章为中断和异常处理；第 13 章是 Win32 窗口程序设计；第 14 章至第 17 章介绍了 x87 FPU、MMX、SSE、AVX 程序设计；第 18 章为 x86-64 位汇编程序设计；第 19 章为上机操作。

本书内容新颖，覆盖面广，重点突出，直观易懂，趣味性强，可供各类高等院校计算机及相关专业作为教材，也可供广大使用汇编语言的工程技术人员参考。

图书在版编目(CIP)数据

x86 汇编语言程序设计/许向阳编著. —武汉：华中科技大学出版社，2020.7
ISBN 978-7-5680-6311-1

Ⅰ.①x… Ⅱ.①许… Ⅲ.①汇编语言-程序设计-高等学校-教材 Ⅳ.①TP313

中国版本图书馆 CIP 数据核字(2020)第 123995 号

x86 汇编语言程序设计

x86 Huibian Yuyan Chengxu Sheji

许向阳 编著

策划编辑：徐晓琦 李 奥
责任编辑：陈元玉
封面设计：原色设计
责任监印：徐 露
出版发行：华中科技大学出版社(中国·武汉) 电话：(027)81321913
武汉市东湖新技术开发区华工科技园 邮编：430223
录 排：武汉市洪山区佳年华文印部
印 刷：湖北新华印务有限公司
开 本：787mm×1092mm 1/16
印 张：21.25
字 数：554 千字
版 次：2020 年 7 月第 1 版第 1 次印刷
定 价：48.80 元

前　言

毋庸置疑，现在的IT界很少使用汇编语言开发项目。作为一种程序设计语言，汇编语言似乎销声匿迹。这也让一些人怀疑学习汇编语言的必要性。在阅读本书后，读者完全可以打消这种疑虑了。不论是对后续课程学习、理解计算机的工作原理，还是编写高质量、高效率的应用程序，汇编语言都起着不可或缺的作用。

首先，汇编语言程序设计是计算机类专业的重要专业基础课，是从事计算机研究与应用，特别是软件研究的基础，是计算机人员必须接受的重要的专业基础训练课之一。汇编语言作为机器语言的符号表示，提供了最直观、最直接学习有关知识的方式。汇编程序可以看成是编译器对高级语言程序编译后的输出产物，也可以看成是在计算机上能直接加工处理的输入对象。因此，汇编语言就是连接高级语言程序和计算机硬件设备的桥梁和枢纽，为深入地理解计算机硬件、操作系统、应用程序之间的交互工作奠定基础。此外，汇编语言对于高级语言程序设计的学习和实践很有帮助。很多人在学习C/C++语言程序设计时，都会有很多疑问。例如，程序在运行中为什么会崩溃？程序运行中为什么会出现“莫名其妙”的结果？函数之间是如何传递参数和返回结果的？递归程序是如何运转的？为什么不要返回局部变量的地址？数组越界访问会造成什么后果？指针是如何实现的？地址类型转换和数据类型转换的含义是什么？在C++程序设计中，对象构造、对象析构、继承、多态、成员对象的引用、虚函数、类模板和函数模板等是如何实现的？汇编语言是揭开高级程序设计语言工作机制神秘面纱的强有力武器。本书给出了一些C语言程序的反汇编示例，直观展现了C语言语句和函数对应的机器指令序列，进而分析变量的空间分配方法、地址类型转换、数据结构中各组成部分的空间关系、函数参数和结果的传递方法、程序执行流程的转移、递归函数的执行过程等奥秘。这些知识有助于从本质上理解程序执行过程。在编程者深刻把握语句的执行原理后，编写程序时就可以少犯错误，且可以编写出执行效率高、形式优美的程序。

其次，开发的软件运行速度快是一个常规要求。除了在“大”方面选择性能高的算法外，还需要在很多“小”方面选择快速的实现方法。在学习汇编语言后，会发现计算机指令系统提供了一些能提高程序运行性能的指令，如串操作指令、单指令多数据流指令。它们比另外一些实现相同功能的指令的速度要快得多。这也就会让人们在使用高级语言开发程序时寻找“封装”高性能指令的函数或语句。本书在串操作和数据成组运算上给出了示例，并检测了不同实现方法的运行时间。当然，在实际项目开发中，有人直接利用汇编语言编写部分关键代码以提高系统的性能。汇编语言保持了机器语言的优点，具有直接和简捷的特点，可有效地访问和控制计算机的各种硬件设备，如磁盘、存储器、CPU、I/O端口等，且占用内存空间少，执行速度快，是高效的程序设计语言。

最后，汇编语言是逆向工程、解密程序、病毒与木马分析和防治的唯一选择。在不支持高级语言开发工具的特定场合下，编写汇编程序是一种必然选择。

本书的特色之一是使用Visual Studio 2019作为汇编语言程序的开发平台。该平台操作简单，与其前辈版本Visual Studio 2010、Visual Studio 2013、Visual Studio 2017等用法相似，

可以和 C/C++程序开发无缝衔接，在汇编语言程序中调用 C 标准库函数、Windows API 函数，或者在 C 程序中调用汇编语言编写的函数。特色之二是根据建构主义理论，将不同的知识关联起来形成网络，同步促进多种知识的学习。书上给出了一些大家熟悉的有代表性的 C 程序例子，通过研究这些例子的反汇编代码，由表及里揭示其内在的处理过程、编译技巧，进而加深对机器指令的运行过程和功能的理解，加深对计算机工作原理的理解。书中也给出了一些既用汇编语言又用 C 语言实现的例子，在学习汇编语言知识后，可以引导编写更高质量的 C 语言程序。特色之三是内容新颖、覆盖面广。本书介绍了目前使用最为广泛的 Intel x86-32 和 x86-64 系列 CPU 的指令系统，包括 x86-32、x87、MMX、SSE、AVX、x86-64 指令，以及机器指令的编码规则；介绍了Windows操作系统下控制台程序和窗口应用程序的开发；包含了多模块程序设计、C 和汇编混合、C 内嵌汇编、中断及异常处理程序开发、执行文件结构等内容。特色之四是趣味性强。在完成某一功能时，采用一题多解的策略，采用不同的寻址方式、不同的指令、不同的算法完成相同的任务，充分展现了编程的灵活性。同时，书上也给出了程序自我修改、机器语言编程、程序转移自主控制等特色例子。特色之五是重点突出。Intel CPU 的机器指令是非常多的，可编写的程序也非常多，本书并不是指令参考书，也不是编程集锦，未纠缠于一一介绍这些指令和过多地给出程序示例，相信读者对这些知识能够举一反三，融汇贯通。建议不要死记硬背那些机器指令，将大脑降档为一个存储器。随着时间的流逝，这些指令内容可能会被遗忘，留在脑海里的是基本原理、基本方法和基本技巧。

在编写本书的过程中，得到了华中科技大学计算机科学与技术学院汇编语言程序设计课程组老师们的热情帮助和支持。汇编语言程序设计课程是国家级精品课程。在精品课程建设中，老师们集思广益，群策群力，使我收获颇丰。本书的编写也得到了华中科技大学出版社编辑的帮助，在此一并表示感谢。

由于作者水平有限，书中错误在所难免，恳请广大读者批评指正。同时也欢迎使用本书的老师、学生和其他读者，共同探讨汇编语言的教学内容和教学方法等问题。

许向阳

2019 年 12 月

目　录

第1章 绪论

本章主要介绍汇编语言和机器语言的概念，剖析汇编语言、机器语言和高级语言之间的关系，阐述学习汇编语言的重要性和方法。本章给出了汇编语言源程序示例，通过示例介绍汇编语言源程序的基本结构和格式。通过本章的学习，应深刻理解计算机世界是0-1世界的本质，计算机中所有的信息都是由0和1组成的，为了探索现实世界中的信息如何编码成计算机世界中的0-1串，以及0-1串又如何解码对应到现实世界的奥妙奠定基础。

1.1 什么是汇编语言

1.1.1 机器语言

学习过C语言程序设计的人都知道，在编写C语言程序之后，需要对这个程序进行编译和链接，生成可执行的exe文件之后，才能运行该程序。那么，执行程序是什么样子的呢？换句话说，执行程序里面存放的是什么呢？

设有如下C语言程序c_example.c，生成的可执行文件为c_example.exe。

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    int x, y, z;
    x=10;
    y=20;
    z=3*x+6*y+4*8;
    printf("3*%d+6*%d+4*8=%d\n",x,y,z);
    return 0;
}
```

使用二进制编辑器打开可执行文件c_example.exe，就会发现它都是一些十六进制数字串。

提示：在Visual Studio 2019中，单击“文件”→“打开文件”，在“打开文件”对话框中选择要打开的文件，并在“打开方式”中选择“二进制编辑器”，即可显示如图1.1所示的机器语言程序片段内容。

图1.1给出的是一个机器语言程序片段。机器语言程序是由机器指令组成的。机器指令(machine instruction)也常被称为硬指令，它是面向机器的，不同的CPU(central processing unit，中央处理器)都规定了自己所特有的、一定数量的基本指令，这批指令的全体即为计算机的指令系统。这种机器指令的集合就是机器语言(machine language)。使用机器语言编写的

```
4D 5A 90 00 03 00 00 00  04 00 00 00 FF FF 00 00
B8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 E8 00 00 00
0E 1F BA 0E 00 B4 09 CD  21 B8 01 4C CD 21 54 68
69 73 20 70 72 6F 67 72  61 6D 20 63 61 6E 6E 6F
74 20 62 65 20 72 75 6E  20 69 6E 20 44 4F 53 20
6D 6F 64 65 2E 0D 0D 0A  24 00 00 00 00 00 00 00
```

图 1.1　机器语言程序片段示例

程序称为机器语言程序。

在计算机开始出现的年代，将由 0 和 1 组成的程序打在纸带或卡片上，1 表示打孔，0 表示不打孔，再通过纸带机或卡片机将程序输入计算机中运行。那时，程序员编写由 0 和 1 组成的程序，然后由专门的人员负责在纸带打孔和填补误穿的孔。图 1.2 展现出来的是编码为“83H 06H”的纸带示意图。

图 1.2　编码为“83H 06H”的纸带

机器指令由一个字节或者多个字节组成，其中包括操作码字段、一个或多个有关操作数地址的字段、操作数(立即数)等。操作码和地址码均是由 0 和 1 组成的二进制代码。操作码指出了操作的种类，如加、减、乘、除、传送、移位、转移等；地址码指出了参与运算的操作数和运算结果存放的位置。除操作码和地址码外，指令中还有一些信息的编码。在第 4.9 节中将详细介绍 Intel CPU 机器指令的编码规则。

提示：

(1) 机器指令是 0-1 串，但并非任意 0-1 串都能解析成机器指令。

(2) 不同 CPU 的机器指令是不同的。

(3) 能够被 CPU 解释执行的只有机器指令。

(4) 可执行文件是有特定的结构的，除程序的机器指令外，还有许多其他信息。在第 11.6 节中将介绍可执行文件的结构。

1.1.2　汇编语言

虽然在 CPU 上能够被解释执行的只有机器语言程序，但是使用机器指令编写程序相当麻烦，编写出的程序也难以阅读和调试。为了解决这些问题，人们就想出了用助记符来表示机器指令的操作码；用变量代替操作数的存放地址；在指令前冠以标号来代表该指令的存放地址等方法。这种用符号书写的、其主要操作与机器指令基本上一一对应的，并遵循一定语法规则的计算机语言就是汇编语言(assembly language)。用汇编语言编写的程序称为汇编源程序。由此可见，汇编语言也是面向机器的语言。

为了让读者对汇编语言有一个感性的认识，我们首先以反汇编的形式显示可执行程序。所谓反汇编就是将由 0-1 组成的机器码翻译成用符号表示的形式。图 1.3 给出了第 1.1.1 节里 C 语言程序(c_example.exe)的反汇编窗口中显示内容的片段，显示信息包括指令在内存中的地址、机器码以及对应的汇编语言指令。

```
x=10;
00251828 C7 45 F8 0A 00 00 00    mov  dword ptr [ebp-8],0Ah
y=20;
0025182F C7 45 EC 14 00 00 00    mov  dword ptr [ebp-14h],14h
z=3*x+6*y+4*8;
00251836 6B 45 F8 03             imul eax,dword ptr [ebp-8],3
0025183A 6B 4D EC 06             imul ecx,dword ptr [ebp-14h],6
0025183E 8D 54 08 20             lea  edx,[eax+ecx+20h]
00251842 89 55 E0                mov  dword ptr [ebp-20h],edx
```

图 1.3　执行程序反汇编后的片段

提示：① 显示反汇编窗口的操作方法请参见第 19 章。② 指令在内存中的地址是不能重现的，每一次运行程序都会发生变化，程序在内存的位置由操作系统控制。

在图 1.3 中：语句“x＝10；”对应的机器指令是“C7 45 F8 0A 00 00 00”，对应的汇编语言语句是“mov dword ptr [ebp－8]，0Ah”。该语句完成的功能是将 10（十六进制为 0Ah）送到地址为[ebp－8]的双字存储单元中。其中，mov 为数据传送指令的助记符，代表了机器指令中的操作符；[ebp－8]表示在当前堆栈段中的一个单元，它是变量 x 的地址；“dword ptr”说明了这个目的操作数是 32 位的二进制数，而源操作数是 0Ah，用 32 位来表示等同于 0000000AH。

在上面的机器指令“C7 45 F8 0A 00 00 00”中，“0A 00 00 00”表示一个数 0000000AH，即十进制数值 10，它是一个双字的数据，数据的低字节放在地址小的单元中，数据的高字节放在地址大的单元中。将 F8H 当成一个字节的有符号数的补码表示，它对应的是－8。在前面还有 C7H 45H，表示要进行“数据传送”操作，还指明了以何种方式取得源操作数和目的操作数。较详细的机器指令编码规则请参见第 4.9 节。

在机器指令的左边，有数字 00251828，这是指令在内存中存放的起始地址。程序调试时，打开内存显示窗口，输入内存的起始地址，会显示该地址为起始地址的一块存储单元中的内容，如图 1.4 所示。

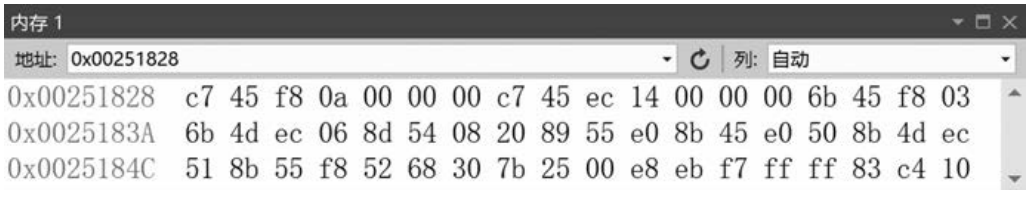


图 1.4　内存显示窗口

“内存 1”窗口中的第一行的开头是“c7 45 f8 0a 00 00 00”，之后的“c7 45 ec 14 00 00 00”是“mov dword ptr [ebp－14h]，14h”的机器指令编码，是 C 语句“y＝20；”编译的结果。“6b 45 f8 03”是“imul eax，dword ptr [ebp－8]，3”的机器码。

由语句“x＝10；”的地址是 00251828、该指令占 7 个字节、下一条指令紧接在前一条指令之后可知，语句“y＝20；”的地址是 0025182FH（00251828H＋7＝0025182FH）。

“z＝3＊x＋6＊y＋4＊8；”被翻译成了 4 条指令，如下：

```
imul eax,dword ptr [ebp-8],3      ;(x)*3->eax
imul ecx,dword ptr [ebp-14h],6    ;(y)*6->ecx
```

```
lea edx,[eax+ecx+20h]              ;(eax)+(ecx)+20h ->edx
mov dword ptr [ebp-20h],edx        ;(edx)->z
```

直观上看，实现“z＝3＊x＋6＊y＋4＊8；”要分成几个步骤来完成，首先是执行 3＊x，结果放在寄存器 eax 中；再执行 6＊y，结果放入另一个寄存器 ecx 中；4＊8 是一个常量表达式，在编译时，计算该表达式的值为 32，即 20H，之后执行三个数相加操作，结果存放在寄存器 edx 中；最后将 edx 中的内容送入变量 z 中。

汇编语言是为了方便用户而设计的一种符号语言，因此，用它编写出的源程序并不能直接被计算机识别，必须将它翻译成由机器指令组成的程序后，计算机才能识别并执行。这种由源程序经过翻译转换生成的机器语言程序也称目标程序。目标程序中的二进制代码（即机器指令）称为目标代码。这个翻译工作一般都由计算机自己去完成，但人们事先必须将翻译方法编写成一个语言加工程序作为系统软件的一部分，在需要时让计算机执行这个程序才可完成对某一汇编源程序的翻译工作。这种把汇编源程序翻译成目标程序的语言加工程序称为汇编程序（即编译器）。汇编程序进行翻译的过程称为汇编（编译）。

在这里，汇编程序相当于一个翻译器，它加工的对象是汇编源程序，而加工的结果是目标程序。汇编程序与汇编源程序、目标程序之间的关系如图 1.5 所示。当然，在 Visual Studio 中集成了编译器，可直接在该开发平台下编写汇编源程序，然后编译、链接、调试和执行。

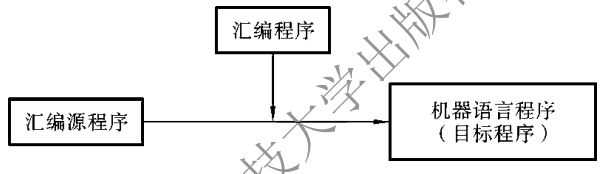


图 1.5　汇编程序与汇编源程序、目标程序之间的关系

为了能让汇编程序正确地完成翻译工作，必须告诉汇编程序，源程序应从什么位置开始安放，汇编到什么位置结束，数据放在什么位置，数据的类型是什么，留多少内存单元作为临时存储区等。这就要求源程序中应该有一套告诉汇编程序如何进行汇编工作的命令，这种命令称为伪指令（或称为汇编控制命令）。由此可见，指令助记符、语句标号、数据变量、伪指令及它们的使用规则构成整个汇编语言的内容。

由于汇编语句基本上与机器指令对应，因此它的编写也是相当麻烦的。为了简化程序的编写，提高编程效率，Visual Studio 提供的编译器有很多新特性，支持很多接近于 C 语言的伪指令，使得用汇编语言写出的程序有点像 C 语言程序。当然，对于初学者，首先要掌握机器指令，在使用条件流控制、函数调用等类似于高级语言语句的伪指令时，要清楚这些伪指令的编译结果，即它们与机器语言的对应关系。

本书中所介绍的是 Intel x86 汇编语言，适用对象是 Intel 公司的 x86 系列 CPU。虽然 Intel公司的 CPU 的型号很多，新产品不断涌现，但它们的机器语言和汇编语言保持了很好的兼容性，新型的 CPU 一般都是在原有的指令系统基础上增加一些新的机器指令。

1.2　为什么学习汇编语言

与机器语言相比，汇编语言易于理解和记忆，所编写的源程序也容易阅读和调试，所占用

的存储空间、执行速度与机器语言的相仿。虽然与机器语言相比，使用汇编语言编写程序要简单很多，但是与高级语言相比，编写程序还是相当烦琐的。

例如，对于一条C语句“z=3*x+6*y+4*8;”，对应的汇编语句如下：

```
imul eax,dword ptr [x],3        ;将x单元中的内容与3相乘，结果放在eax中
imul ecx,dword ptr [y],6        ;将y单元中的内容与6相乘，结果放在ecx中
lea  edx,[eax+ecx+20H]          ;将(eax)+(ecx)+20H相加，结果放入edx中
mov  dword ptr [z],edx          ;将edx中的内容放入z单元中
```

提示：上述语句与图1.3所示的反汇编语句略有不同，但是机器码是完全相同的，只是在反汇编窗口的“查看选项”中勾选了“显示符号名”，即用符号名x代替了地址表达式[ebp-8]。虽然采用符号名更直观一些，但其本质是一个存储单元地址的符号表示。

纵观计算机程序设计语言的发展历史，是从机器语言、汇编语言向高级语言发展的。其目标就是要让程序员编写程序越来越简单。高级语言接近于自然语言，易学、易记、便于阅读、容易掌握、使用方便、通用性强，且不依赖于具体的计算机。在使用高级语言编写程序后，只需要配备相应的编译器，将其翻译成机器语言程序即可。从C语言到面向对象的C++语言，再到Java和Python语言以及各种开发平台的出现，都反映了将程序员从繁重的一般性的脑力劳动中解放出来的进步。越来越好的编程语言和开发工具减少了编写程序这一部分的工作量，同时提高了程序开发效率，能够让人们更多地集中精力分析需求、设计和研究高效算法。

毋庸置疑，在现在的IT界很少使用汇编语言编写程序，那么我们为什么要学习汇编语言呢？对于计算机科学与技术专业的学生而言，学习汇编程序设计语言的必要性或者价值主要体现在以下几个方面。

(1) 逆向工程、程序解密、病毒和木马分析及防治的唯一选择。

当进行程序解密、病毒和木马分析等工作的时候，我们并没有源程序，而只有可执行程序。此时，需要对可执行程序进行反汇编而得到汇编语言程序，然后进行阅读和分析。

(2) 理解高级语言的最好途径。

随着高级程序设计语言的不断进步，它们离机器语言越来越远，使得编程者看不到程序中语句的执行过程，看不见程序运行的奥秘。这对一种语言的初学者掌握相关的知识带来了很大的挑战，束缚了灵活应用编程语言的手脚。汇编语言就是揭开程序语言工作机理神秘面纱的最佳工具。

在C语言程序设计中，我们关心的问题很多。例如，地址类型转换和数据类型转换的含义是什么？为什么调用一个函数后能够返回到调用语句的下一行？函数之间是如何传递参数和返回结果的？为什么局部变量的作用域只在函数内部？递归程序如何理解？数组越界访问是怎么回事？指针是如何指向相应对象的？UNION结构中各成员是什么关系？程序运行中崩溃时，导致崩溃的原因是什么？在C++程序设计中，对象构造、对象析构、继承、多态、成员对象的引用、虚函数等更复杂的内容是如何实现的？泛型程序设计是如何实现的？在编程者深刻把握语句的执行原理后，编写程序时就能少犯错误，也能写出执行效率高且形式优美的程序。掌握这些学科的基础知识，有助于程序员提升编程水平。

通过对执行程序的反汇编，或者分析在编译时生成汇编语言程序，可以清楚地看到每一条语句对应的执行序列、变量的空间分配方法、数据结构中各组成部分的空间关系、数据的传递方法、程序执行流程的转移方法，从而分析高级语言程序的运转原理。相信读者在学习本书并