MS—DOS

80286汇编语言

【美】利奥丁·斯坎伦 著

周少柏 马腾阁 查良钿 译

中国科学院希望电脑公司

一九八八年五月

前　言

为什么要用汇编语言？

许多人用一种所谓高级语言（具体地，BASIC）写他们所有的计算机程序。BASIC易于学习、易于使用、而且对大多数计算机任务而言足够快。情况就是这样，但为什么某些人会想使用其它语言？一个理由是BASIC（象人类语言那样）不能很好地适应各种情况。某些任务用其它语言要容易得多。例如，设想一下，试图描述好的烹调而不用某些法语词汇或描述交响乐而不用某些意大利术语。类似地，特殊的计算任务（象作图、音乐、或字处理）用专用语言往往会更容易些。

再者，BASIC十分慢，术语"慢"可能使初学者感到惊奇，因为短的程序看来只在瞬间就运行完。但是，问题出现在下列的情况：

1.当涉及大量数据时。例如，当程序必须排序名字和地址或帐目的长表时，你将注意到BASIC是多么的慢。类似地，当程序必须搜索50页的报告或保留有关几千项目的存货记录时，BASIC也是十分慢的。

2.当涉及作图时，如果程序是在屏幕上绘制图画，它必须很快地工作，否则那样的延迟是无法容忍的。如果假设在图画中的实体是移动的，那末，程序必须足够快地使动作看起来自然。当图画包含许多实体时（象宇宙飞船、基地和外来侵略者等），这是特别困难的。它们全都向不同方向移动。

3.当要求不少判断或"思考"时。象在国际象棋或西洋棋那样的复杂游戏中，这常常是必须的。程序应该尝试许多可能情况并决定合理的移动。显然，有越多的可能情况和要求越多的分析，计算机为作出移动将要执行更长的时间。

BASIC为什么慢？首先，计算机实际上翻译每一BASIC语句为简单的内部命令（所谓机器或汇编语言）。在每次它运行BASIC程序时都做这件事。于是，许多计算机时间都消耗在翻译程序，而不是运行它。

有称为"编译程序"的各种BASIC版本，它们执行翻译一次，然后保留翻译的文本。但是，因为它有比较机械的特性，BASIC仍然是慢的。它实际上象带有自动变速的汽车；但没有足够的诱导使你能取得性能和节省燃料，而用手工变速的熟练驾驶员都能获得这些。人类显然是比自动变速或BASIC的解释程序或编译程序更灵活、更富于技巧，和更洒脱的操作员。

汇编语言就是人工变速的计算机等价物。它以更多的工作，更为繁琐和不很方便为代价，向程序员提供对计算机较大的控制。象自动变速那样，BASIC对大多数程序员在大部分时间里是足够好的。但对于那些必须从它们的计算机取得最大性能的人们而言，汇编语言是重要的。你将会发现大多数复杂的游戏，图形程序，以及大的事务程序，至少部分地是用汇编语言编写。

即使汇编程序是你喜爱的选择，你仍可能怀疑你是否已有足够的基础学习汇编语言程序设计。如果你已经做过某些种类的程序设计，你就一定可以。如果你知道BASIC或某些其它高级语言，那就很好。如果你已经用汇编语言开发过一些程序，那就更好

了。为了有利于那些高级语言的用户，本书有两个出发点。

如果你从未用汇编语言编写过程序，从第0章开始，它给你以“二进制”和“十六进制”数字表示系统的“速成课程”。否则，如果你已经知道这些术语是什么意思并理解如何使用它们，那末直接进到第一章。

本书的内容

在第一章我们介绍80286微处理器（计算机的大脑）并讨论它在系统中的作用。

第二章一般讨论汇编程序，然后描述Microsoft的“宏汇编程序”。所有80286兼容的汇编程序都做同样的工作：它们把你能理解的指令翻译为计算机的微处理器能理解的数码。因此，我们描述的大多数原理都将适用于你可能有的任何其它汇编程序。第二章还提供一个简单的程序，并告诉你怎样把它键入计算机，汇编它，并执行（运行）它。

第三章描述80286的指令系统，即那些你用以和你的计算机通信的汇编语言命令。本书按功能组而不是字母顺序陈述指令。即，我们组合加、减、乘和除，如此等等。用这个方法，你不仅能“理解”那些指令能做什么，而且你还能鉴别它们是怎样组合在一起的。

第四章告诉你怎样组合这些指令以执行扩充的数学操作，它们是微处理器的指令系统不直接提供的。

第五章涉及对各种表格的操作。除了一般的功能之外，MS—DOS还包括有用以和键盘，屏幕、磁盘驱动器、打印机以及其它外部设备“交谈”的程序。汇编语言向你提供访问这些程序的方法，从而节省你好几小时的程序设计时间。

第六章告诉你怎样做这些事情。

第七章涉及“宏”。宏是你可以简单地通过提出它的名字而把它插入主程序的一种小程序。宏可以使汇编语言程序几乎象BASIC程序那样易于开发。本章还向你示出怎样建立你最有用的宏的“库”。

第八章描述“目标库”（包含汇编子例行程序的磁盘文件）的使用。事实上，目标库是你的程序可以选择的，预先制成的工具集合。

第九章讨论你能把建立汇编语言程序的过程自动化的两个方法。一个方法是使用DOS批处理文件；另一个是使用Microsoft宏汇编程序的“程序维护程序”实用程序。（称为MAKE）。

最后，在第十章中，我们讨论80287数学协处理器，它是可以执行复杂算术操作的任选芯片。

为了你的方便，本书提供四个附录。附录A有一些表，它可以帮助把十六进数转换为十进数，或相反。附录B概述ASCII字符集（你可以读自键盘并显示在屏幕的字符）。附录C和D以字母顺序概述80286的指令系统，并示出执行每一指令需要多长时间，每一指令在存储器中占多少字节，以及它影响哪些标志等。

学习习题

大多数章节都包括一组问题和程序设计练习。其中某些测试你对本章资料的理解；其它是想把你的知识面扩大到附加的、有关的课题。

补充材料

本书用以补充宏汇编程序手册和计算机所带的一切手册，所以你大概不需要任何其

它的参考书。但是，你也许需要一本“Microsoft MS—DOS程序员参考手册”的复印本。它有有关磁盘组织的详细信息，你能从汇编语言调用的DOS特色、以及其它“吸引人的东西”。

至于80286微处理器和80287数学协处理器的全部细节，可参看包括“iAPX286数值参考”的“iAPX286程序员参考手册”。

目　录

前　言……（1）
0．计算机数字表示的速成课程……（1）
0.1 二进制数字表示……（1）
八位构成一字节……（2）
加二进数……（3）
带符号数……（3）
0.2 十六进制数字表示……（5）
使用十六进数……（6）
学习习题……（6）
1．引言……（7）
1.1 什么是汇编语言？……（7）
1.2 评价80286……（7）
1.3 80286微处理器的总述……（8）
操作模式……（8）
内部寄存器……（9）
分段……（9）
软件特点……（10）
输入／输出空间……（11）
存储分配……（11）
中断……（11）
数据和地址总线……（13）
1.4 内部寄存器……（13）
数据寄存器……（13）
段寄存器……（15）
指针和变址寄存器……（15）
内部部件……（15）
指令指针……（16）
标志……（16）
学习习题……（18）
2．使用汇编程序……（19）
2.1 引言……（19）
Microsoft的宏汇编程序……（19）
2.2 开发汇编语言程序……（19）

编辑程序……（20）
汇编程序……（20）
连接程序（LINK）……（20）
一种调试程序（SYMDEB）……（20）
自顶向下的程序设计……（21）
2.3 源语句……（21）
源语句中的常数……（22）
2.4 汇编语言指令……（22）
标号域……（22）
助记忆符域……（23）
操作对象域……（23）
注释域……（24）
2.5 汇编程序指示符……（24）
数据指示符……（25）
列表指示符……（34）
模式指示符……（34）
2.6 操作符……（35）
算术操作符……（38）
逻辑操作符……（38）
关系操作符……（39）
送回值操作符……（40）
属性操作符……（41）
2.7 编辑、汇编和运行一个程序……（42）
例子程序……（42）
键入程序……（43）
汇编该程序……（45）
列表源程序……（46）
建立运行文件……（47）
运行该程序……（47）
先进的列表任选项……（53）
2.8 构造程序的模型……（54）
主程序模块……（54）
副模块……（55）
使用这些模块……（56）
2.9 COM文件……（56）
建立COM文件的规则……（56）
建立COM文件……（57）
COM程序的模型……（58）
COM文件的pros和cons……（59）

2.10 先进的指示符……（60）
数据指示符……（60）
条件指示符……（62）
列表指示符……（64）
2.11 要点一览……（65）
学习习题……（67）
3．80286指令系统……（68）
3.1 编址模式……（68）
寄存器和立即编址……（69）
存储编址模式……（70）
3.2 指令类型……（74）
3.3 数据传输指令……（78）
通用指令……（78）
输入和输出指令……（83）
地址传输指令……（83）
标志传输指令……（84）
3.4 算术指令……（85）
数据格式……（85）
加法指令……（87）
减法指令……（89）
乘法指令……（92）
除法指令……（94）
符号扩展指令……（95）
3.5 位处理指令……（95）
逻辑指令……（96）
移位和旋转指令……（98）
3.6 控制转移指令……（101）
无条件转移指令……（101）
条件转移指令……（106）
循环指令……（110）
3.7 字符串指令……（111）
方向指令……（112）
重复前缀……（112）
传送字符串指令……（113）
重置段分配……（114）
比较字符串指令……（115）
扫描字符串指令……（116）
装载字符串和存放字符串指令……（117）
输入／输出字符串指令……（118）

3.8 中断指令……（119）
3.9 处理器控制指令……（121）
标志操作……（121）
外部同步指令……（122）
不操作指令……（123）
3.10 高级指令……（123）
3.11 保护模式指令……（124）
3.12 要点一览……（124）
80286和8086/8088之间的差异……（126）
学习习题……（127）
4．高精度数学……（129）
4.1 乘法……（129）
无符号32位×32位乘法……（129）
带符号32位×32位乘法……（132）
4.2 除法……（133）
处理溢出……（136）
4.3 平方根……（137）
学习习题……（138）
5．对数据结构的操作……（140）
5.1 无序表……（140）
增加元素到无序表……（140）
从无序表中删除元素……（141）
无序表中的最大和最小值……（143）
5.2 排序无序数据……（144）
冒泡排序……（144）
5.3 有序表……（149）
搜索有序表……（149）
增加元素到有序表……（153）
从有序表中删除元素……（153）
5.4 查寻表……（155）
查寻表可以代替方程……（156）
查寻表可以执行代码转换……（159）
转移表……（159）
5.5 正文文件……（161）
学习习题……（162）
6．使用DOS资源……（164）
6.1 系统中断……（164）
6.2 DOS中断……（165）
DOS类型21功能调用……（166）

功能调用错误报告……（170）
中断向量功能……（172）
目录功能……（172）
扩展的文件管理功能……（173）
DOS错误信息程序……（174）
6.3 时间和日期操作……（176）
计时程序……（176）
生成延迟……（177）
6.4 视频功能调用……（179）
ASCII……（179）
视频功能调用的概述……（182）
6.5 键盘功能调用……（183）
读单个的键……（184）
读字符串……（184）
响应提示符……（185）
6.6 ASCII／二进制代码转换……（187）
把ASCII字符串转换为二进数……（187）
把二进数转换为字符串……（193）
学习习题……（194）
7．宏……（195）
7.1 宏的简介……（195）
宏与过程……（195）
宏的内容……（196）
7.2 宏指示符……（197）
通用指示符……（199）
重复指示符……（199）
条件指示符……（200）
列表指示符……（202）
7.3 宏操作符……（202）
7.4 在源程序中定义宏……（203）
7.5 宏库……（203）
建立宏库……（203）
把宏库读入程序……（204）
8．目标库……（205）
8.1 建立目标库……（205）
8.2 对目标库的操作……（205）
获取库的目录……（206）
8.3 使用目标库……（206）
9．自动化汇编过程……（207）

9.1 批处理文件……（207）
例子……（207）
9.2 Microsoft程序维护程序（MAKE）……（208）
使用MAKE……（208）
例子……（209）
9.3 比较这两种技术……（209）
结论……（210）
10. 80287数学协处理器……（211）
10.1 内部寄存器……（211）
80287的堆栈……（211）
浮点格式……（212）
10.2 数据类型……（212）
10.3 指令系统……（213）
10.4 用宏汇编程序进行80287程序设计……（216）
常数……（216）
数据定义指示符……（216）
发现80287……（216）
10.5 梗概……（216）
学习习题答案……（218）
附录A.十六进数／十进数转换……（225）
附录B.ASCII字符集……（226）
附录C.80286指令时间……（226）
附录D.80286指令系统一览……（232）

第0章　计算机数字表示的速成课程

除非你来自别的星球到这里访问，不然，你已耗费你整个一生在使用十进制计算东西。数学家们称十进数为“基10”数字表示系统，因为它有10个数字（0到9）。

人类十分习惯用十进制计数（大概是因为我们有十个手指和脚指），但计算机就不是这样了。代替的是，它们用基2（或二进制）数字表示系统计数，这个系统只有两个数字，0和1。于是，为了在它本身的水平上和计算机通信（象你用汇编语言编程序时所作的那样），你必须熟悉二进制数字表示。除了二进制，汇编语言还使用其它的数字表示系统——基16（或十六进制），所以，你也必须熟悉它。

本章是计算机数字表示系统的“速成课程”，是为了那些从前没有接触过它们的读者而写的。这就是为什么我们称之为第0章的缘故。如果你已经理解二进制和十六进制数字表示，可以试试跳过这一章而从第一章开始。

0.1 二进制数字表示

计算机从它的“存储器”取得所有程序指令和数据。存储器是由包含几千个电子元件的集成电路(或，”芯片”)组成的。象光学开关那样，这些元件只有两个可能的设置：“开”或“关”。然而，只用这两种设置，存储器元件的组合可以表示任何大小的数。怎样表示呢？请继续往下阅读。

存储器元件的开和关设置相应于“二进制数字表示系统”（计算机的基本系统）的两个数字。只有两个数字——1（ON）和0（Off），所以，二进制数字表示系统是“基2”系统。再者，它不同于标准的十进制数字表示系统——它有10个数字（0到9）。

存储器的类开关元件称为“位”（二进制数字的简称）。根据约定，具有值1的位是“ON”，而具有值0的位是“Off”。这似乎是有点难受的限制，直到你想起十进数也不过只能安排0到9为止。就象你能组合十进制数字以形成大于9的数，你也可以组合二进制数字以形成大于1的数。

如你所知，为表示大于9的十进数要求附加的“十位”数字。同样地，为表示大于99的十进数要求“百位”数字，如此等等。每一个你添加的十进制数字都有10乘它直接右边数字的“数”。

例如，你可以表示十进数324为

$$(3\times100)+(2\times10)+(4+1)$$

或作为

$$(3\times10^2)+(2\times10^1)+(4\times10^0)$$

于是，每个十进制数字都比前面数字大10的幂。

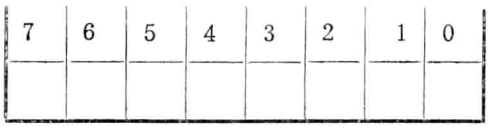
类似的规则适用于二进制数字表示系统。这里，“每个二进制数字都比前面数字大2的幂。”最右位有2^0（十进数1）的数，次一位有2^1（十进数2）的权，如此等等。例如，二进数101有5的十进值，因为

$$101_2=(1\times2^2)+(0\times2^1)+(1\times2^0)$$

$=(1\times4)+(0\times2)+(1\times1)$

$=5_{10}$

你现在是否已经明白二进数是怎样构造的呢？为找出任一给定位的位置的值，你加倍前一位的位置的权。于是，前八位的二进制权是1，2，4，8，16，32，64，和128，如图0—1所示。



7	6	5	4	3	2	1	0	位的位置
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2的幂
128	64	32	16	8	4	2	1	十进值

图0—1　八位二进数字的权

为把十进值转换为二进制，你要作一系列简单的减法。每一次减法产生单个的二进制数字（位）的值。

开始，从十进值减最大的可能二进制权，并在该位位置打入1。然后从结果减次一最大的可能二进制权，并在该位的位置打入1。再然后从结果减次一最大的可能二进制权，并在该位的位置打入1。继续直到结果为零。在它的权不能减自当前十进值的某些位的位置打入0。例如，为把十进数50转换为二进数：

```
  50
 -32          位的位置5 = 1
  18
 -16          位的位置4 = 1
   2
  - 2         位的位置1 = 1
   0
```

在其它位的位置（位3，2，和0）打入0，产生110010的最后结果。

为验证十进数50的二进制等价物确实是110010，把“1”位置的十进制权相加：

```
  30    （位5）
  16    （位4）
+  2    （位1）
  50
```

八位构成一字节

苹果Ⅱ系列，Commodore64和VIC—20，Radip Shack TRS—80，以及许多其它流行的微计算机都设计成围绕“8位微处理器”。八位微处理器之所以这样命名是因为它们每次八位地处理信息。为处理多于八位，它们必须执行附加的操作。

用计算机的术语，信息的八位单位称为“字节”。使用八位，一个字节可以表示十进值从0（二进数00000000）到255（二进数11111111）。

因为字节是处理的基本单位，微计算机用它们的存储器能容多少字节（而不是位）的数目来描述。微计算机制造商以1，024字节的块来构造存储器。这个具体的量反映计算机的二进制倾向在于它表示为2^{10}字节。

值1，024有标准的缩写：字母k。于是，有“256k存储器”的计算机包含256×1,024（或262，144）字节。

加二进数

你可以用加十进制的同样方法加二进数：通过把任何来自一列的超额进位到下一列。例如，为加十进值7和9，你必须进1到“十位”列以产生正确的结果（16）。类似地，为加“二进”值1和1，你必须进1到“二位”列以产生正确的结果（10）。

加多位数稍微复杂些，因为你必须包括来自前一列的进位。例如，这个操作涉及两个进位：

```
  1011
+   11
------
  1110
```

最右列的加法（1+1）产生0的结果和向第二列加1的进位。带着进位，第二列的加法（1+1+1）产生1的结果和向第三列的进位。

二进制加法的一般规则在此表示出：

输入		结果		
操作对象 #1	操作对象 #2	进位	和	进位
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

带符号数

至此，我们已讨论怎样用二进制表示“无符号数。如同我们早些时候提到的那样，无符号数中的每一位都有反映它的位置的权。最右位（最低有效位）有1的权，而每一更高有效位有两倍它的先驱的权。因此，如果所有字节中的八位都是0，那末，该字节有值0；如果它们全都是1，那末，该字节有值255。但是，你常常需要对正或负值（即，对“带符号数”）操作。当字节包含带符号数时，只有七个最低有效位（0到6）表示数据；最高有效位（7）说明数的符号。“如果数是正或零，那末，符号位为0；如果它是负，那末，符号位为1”。图0—2示出带符号和无符号字节的安排。

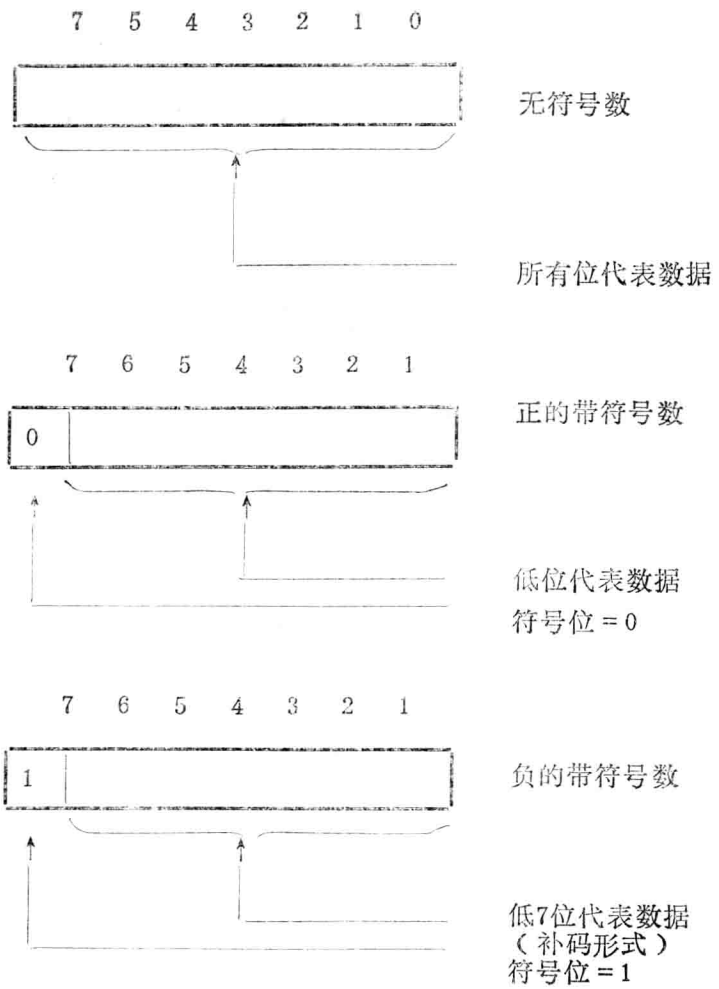


图0—2　带符号和无符号数的表达

当保留带符号数时，单个字节可以表示0（二进数00000000）和 + 127（二进数01111111）之间的正值以及 - 1（二进数11111111）和 - 128（二进数10000000）之间的负值。

请注意，二进数的 - 1是11111111。使它正好是10000001（即带负号位的1）不是更简单吗？不，它会产生错误的答案。例如，考虑如果你加 + 1和 - 1，会出现些什么。答案当然应该是0，但你却得到：

$$\begin{array}{ll} 00000001 & = +1 \\ 10000001 & = -1 \\ \hline 10000010 & = -2 \end{array}$$

于是，我们需要的是表示 - 1的某种方法，以便当你把它和 + 1相加时你能得到0。这就是为什么数学家们用11111111代表 - 1的缘故：它能产生正确的答案。

为测试这个，让我们再做一次我们的加法：

```
  00000001      = +1
  11111111      = -1
 ---------
 100000000      = 0
```

开头额外的1位是“进位”（加法的左超出位），我们简单地忽略它。

补码

象-1那样，所有负的带符号数都以一种特殊的形式表示，它使加法产生正确的答案。这被称为“补码形式”。

为找出负数的二进制表达（即，找出它的补码形式），简单地取数的正的形式，并逆转每一位（改变每一个1为0以及每一个0为1），然后把结果加1。下面的例子示出如何计算-32的补码二进制表达。

```
  00100000      +32
  11011111      逆转每一位
+        1
 ---------
  11100000      加1
                -32（补码形式）
```

当前，补码约定使负数难于译解。幸运的是，你可以使用我们刚才给出的同一过程寻找（补码）负数的正的形式。例如，为找出值11010000有什么样的正的形式，可如下处理：

```
  00101111      逆转每一位
+        1      加1
 ---------
  00110000      = 16 + 32 = +48
```

汇编程序允许你以十进制形式键入数（带符号或无符号的），并自动地进行所有转换。但是，你可能需要解释存放在存储器或寄存器的负数，所以，你自己应该知道怎样进行这些转换。

0.2 十六进制数字表示

虽然二进制数字表示系统是表示存储器中数的精确方法。但只有1和0的字符串却很难处理。况且，它们也易出错，因为象10110101这样的数是极端容易打错的。

若干年前，程序员发现他们一般都是对位的“组”而不是各别的一些位操作。最早的微处理器是4位设备（它们每次四位地处理信息），所以对二进制符合逻辑的替补是以四个为组的数字位系统。

如你所知，四位可以表示二进值0000到1111（它等价于十进值0到15），总共16个可能的组合。如果数字表示系统是表示这16个组合的，那末，它必须有16个数字，即，它必须是“基16”系统。

如果基2数字称为“二进制”，而基10数字称为“十进制”，那末，对于基16系统采用什么术语合适呢？不知是谁组合希腊字“hex”（表示六）和拉丁字“decem”（表示十）以形成字“hexadecimal”（十六进制）命名了基16系统。于是，基16系统就称为“十六进制数字表示系统”。

在十六进制数字表示系统的16个数字当中，前十个标为0到9（十进值0到9），而后六个标为A到F（十进值10到15）。表0—1列出每一个十六进数字的二进和十进值。

表0—1　十六进制数字表示系统

十六进数字	二进值	十进值	十六进数字	二进值	十进值
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	A	1010	10
3	0011	3	B	1011	11
4	0100	4	C	1100	12
5	0101	5	D	1101	13
6	0110	6	E	1110	14
7	0111	7	F	1111	15

象二进和十进数字，每一个十六进数字都有一个权，它是它的基的某些倍数。由于十六进数字表示系统基于16，每一个数字都要比它直接右边的数字加权16倍。即，最右的数字有16^0的权，次一有16^1的权，如此等等。例如，十六进值3AF有943的十进值，因为

$$(3\times16^2)+(A\times16^1)+(F\times16^0)$$

化为十进制形式：

$$(3\times256)+(10\times16)+(15\times1)=943$$

使用十六进数

当BASIC和其它高级语言通常以十进制形式显示数的时候，汇编语言一般以十六进制形式显示它们。这包括地址，指令码，以及存储单元和寄存器的内容。因此，为从你的程序设计中取得最大的效益，试一试“设想十六进制”。开始，这是困难的，但当你获得更多经验时，它就变得比较容易了。为了帮助你向前，附录A提供把十进数转换为十六进数和相反转换的表。

学习习题

1.把下面的十进值转换为二进值：

（a）12　（b）17　（c）45　（d）72

2.把下面的无符号二进值转换为十进值：

（a）1000　（b）10101　（c）11111

3.你怎样把习题2中的三个二进数表示为十六进数？

4.试列出十六进数D8的十进制等价物，如果：

（a）D8表示无符号数

（b）D8表示带符号数

第一章 引 言

1.1 什么是汇编语言？

如同BASIC一样，汇编语言是词的集合，它向计算机说明该做些什么事情。但是，汇编语言指令系统当中的词直接涉及计算机的元件。就象告诉某些人走到邮筒处和精确地告诉他们怎样移动肌肉，举步穿过障碍之间的差异一样。显然，在大多数情况下，一条简单的命令已很充分；只有运动员和登山者才需要更详细的指示。

汇编语言给计算机以详细的命令，象“把32装入AX寄存器”，“把CL寄存器的内容传输到DL寄存器”，和“把DL寄存器中的数存放到存储单元3,456”。如同你所看到的那样，BASIC和汇编语言差别在于你怎样指示计算机，“用BASIC，你一般地讲；用汇编语言，你具体地讲。”

虽然汇编语言比BASIC需要更多的时间和工作写程序，但它们也运行得更快。这里，详细的程度是关键。想法是和下面的事情一致的，即一个能注意他或她所做的每一步骤的运动员将跑得更快或跳得更远。为取得最大的性能，精确的形式是重要的。

因为汇编语言要求你在计算机的内部元件上操作，你必须了解含有这些元件的集成电路（或“芯片”），即计算机的微处理器的特性和能力。在本书，我们将研究Intel 80286微处理器。但在我们探究这种芯片的细节之前，让我们先简要地看一看它是怎样诞生的。

1.2 评价80286

最早的微处理器是4位的装置，这意味着它们每次只传输4位的信息。为传输多于4位，它们执行若干独立的传输操作。当然，这也就使得它们变慢。

1972年推出的Intel 8008，是第一个商业化的8位微处理器（它每次传输8位信息），并仍然被认为是第一流的“第一代”8位微处理器。用类计算器的体系结构设计，8008有一个累加器，6个暂时寄存器，一个堆栈指针（作临时存储用的专用地址寄存器），8个地址寄存器，以及执行输入和输出的专用指令。在1973年，Intel推出8008的“第二代”版本，称为8080。

8080是增强的8008；它具有更多的编址和输入／输出（I／O）能力，更多的指令，并且执行指令更为快速。虽然Intel在8080中维持总的8008体系结构原理，但内部的组织比较好。8080在历史上是第二代微处理器事实上的标准——当人们提到微处理器时，许多人仍然首先想到的一种。

1976年，技术上的进步使Intel能推出8080的增强版本，称为8085。基本上是重组装8080，8085加进了象通电时重置（初始化微处理器），向量中断（服务于外部设备的需要），串行I／O端口（附加指针和其它外部设备），以及单个的+5伏电源（8080要求两个电源）那样的一些特性。