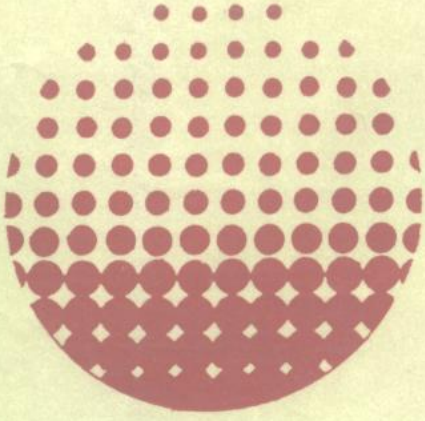
MS—DOS
汇编语言子程序精选

L. J. 斯堪隆　　编著
栾毓敏 钟礼花　　合译
林定基　　审校



北京科海培训中心
新地文化事业有限公司

MS—DOS
汇编语言子程序精选

L. J. 斯堪隆　　编著
栾毓敏 钟礼花　　合译
林定基　　审校

北京科海培训中心
新地文化事业有限公司
一九九二年八月

内容提要

本书包含上百个汇编语言子程序的源程序清单。其中除了六个子程序是IBM－PC专用以外，其余均可在任一MS－DOS或PC－DOS的计算机上运行，而不管这些计算机所采用的微处理器是8086，8088，80186，80286，还是80386。

本书可以作为一本工具书使用，程序员可以从这里找到所需要的子程序。它节省了有经验程序员的时间，使他们不必去花精力建立这些子程序。而且，对刚入门的和非专业的程序员来说，它也是有用的，这是因为本书不仅给出了预先经过测试的程序代码，而且阐明了使用它们的正确方法。实际上，本书不仅仅是一本工具书，它也是关于汇编语言程序设计的一本简明教材，因此也适合大专院校和各种培训班使用。

Assembly Language
Subroutines for MS－DOS
Leo J. Scanlon
Windcrest，1991

目　录

前　言

我在编写一个汇编语言程序时，常常需要一些能完成某种公共任务的短程序，如象 显示一些信息，或者读取一个键盘字符等。早期我必须花费不少精力去研究 IBM 或 Microsoft 公司的程序员参考手册，找出适合该任务用的 DOS 功能调用，然后将它的数字 代码插入到我的程序中，再继续下一步工作。以后，我熟悉了一些供特定应用的程序代码 段，但有时仍然需要亲自查找程序员参考手册，以便找出那些我还不知道的程序代码段。

最后当积累到一定程度时，我决心花费若干小时来建立最常用作业的子程序。有了 它们就可以从我的主程序中直接调用一个文件，并且在继续进行程序设计时，不必再记住功能调用码(例如字符显示调用码 2 或 3)，更不必首先去查找程序员参考手册！

开始我只有五、六个子程序，以后随时添加。我还开发了一些非功能调用的子程序，用于完成如象二—十进制转换，或在存储器中移动一个数据块等。于是，我终于有了一个子程序库。

有一天，我将这些子程序的有关情况告诉一位程序员朋友，他向我索取这些子程序的拷贝。几星期后，他对我说，他发现这些子程序很有用，并且建议再增加一些。这就给我一种设想：为什么不再增加一些人们经常需要的子程序，并且写成一本书来发表它们呢？现在你正在阅 读的这本书就是上述设想的结果。

我不能保证，你要做的每一件事情都能从本书中找到现成的子程序，但我相信大多数有用的子程序 已经包括在本书之中。如果你能建议一些其它的子程序，或者找到一些能完成其中某些任务的 简便方法，我将十分感激并请你写信给我。我衷心地希望能发现这些子程序(或本书中的任何 从而段落)中的差错，尽管它们已经通过测试，但偶尔的差错也在所难免。最后，我利用 IBM 和 Microsoft 宏汇编对这些程序进行汇编。如果你曾用过别的汇编程序成功地对它们进行了汇编，那我将感谢你向我提供有关情况。

序　言

本书包含一百多个汇编语言子程序的源程序清单。你可以利用 IBM 或 Microsoft 的宏汇编程序或与其兼容的宏汇编程序（如 Borland 的 Turbo 汇编程序）来汇编这些子程序。其中有六个子程序是 IBM－PC 专用的，其余的可以在任何一个采用 MS－DOS 或 PC－DOS 的计算机上运行，而不管这些计算机采用的微处理器是 8086，8088，80186，80286，还是 80386 。

本书可以作为一本工具书使用，程序员可以从这里找到所需要的子程序。它节省了有经验的程序员的时间，使他们不必再花精力去建立这些子程序。可以证明，它对于刚入门的和非专业的程序员来说都是有用的，这是因为本书不仅给出了预先经过测试的没有错误的程序代码，而且阐明了使用它们的正确方法。

然而本书不只是一本工具书，它也是一本关于汇编语言程序设计的简明教材。请注意我说的是"简明"教材，我并不打算提供有关汇编语言程序设计的全部内容，因为在市场上这类好书已经很多。我所做的只是把关于微处理器，宏汇编与 Turbo 汇编以及汇编语言指令系统的最主要细节归纳在一起。

在所提供的教材中，我已经把所有基本的细节放在一起。省得你在本书和你用的任何手册之间来回翻阅。当然，它也有助于在你的主要参考资料中填补某些空隙。

读者对象

本书假定你已经在 MS－DOS 的计算机上完成过一些汇编语言程序设计。还假定你至少阅读过这方面的一本教材（并且对它有一般的了解），或者你已经阅读过你的汇编程序随带的手册。

使用本书时你需要些什么

为了使用本书中的子程序，你需要一台带有 MS－DOS（或 PC－DOS）2.1 版本以上的计算机。你还需要一个汇编程序，我已经用 IBM 和 Microsoft 宏汇编程序，以及用 Borland 的 Turbo 汇编程序对这些子程序进行了测试。它们也可能在其它的汇编程序下满意地工作，但是如果你的汇编程序的伪指令与 IBM、Microsoft 以及 Borland 的汇编程序伪指令有所区别，那么你可能需要作某些修改。

本书包含哪些内容

本书共分二十一章，第一章介绍当前流行的 MS－DOS 计算机中使用的微处理器（Intel 8086，8088，80186，80286 和 80386），并且讨论了它们的内部寄存器和寻址方式。

第二章概述宏汇编程序的命令或指令，并且讨论了开发汇编语言程序所需要的步骤.这包括从程序输入计算机，到产生最后可执行的程序：运行模块。

第三章按功能分组来介绍整个微处理器的指令集。通过按功能将指令分组的方法，你不仅能够知道这些指令作什么用，而且你还能理解怎样将它们结合在一起。

第四章介绍两个程序模块，你可以用它们作为建立你自己的程序的起点。这些模块包含汇编程序要求每个程序所应有的"规范格式"。

第五章至第二十一章分别按功能分组介绍子程序。对于每个子程序除了提供源码清单以外，还描述了它是怎样工作的，并且说明它的入口参数值，以及如何将它连接到调用程序，和它调用任何其它子程序所需要的命令。在很多场合，我还提供了使用子程序的例子。

第五章介绍执行存储器操作的子程序.这些子程序包括：用一个值去填满一个数据块，移动和比较数据块，检索指定值，以及对一些连续的单元取平均值。

第六，七和八章包括一些用来扩展微处理器指令实现多精度操作的子程序，这些子程序包括32位二进制运算，16位十进制运算，以及32位移位和旋转操作。

第九章提供了一些代码转换子程序。

第十章涉及字符串处理，它包括将一个串插入到另一个串中间，或者将一个串添加到另一个串上，以及查找、删除、拷贝、或者移动一个串等。本质上，这一章提供了类似字处理的功能。

下面三章涉及表处理。第十一章介绍处理无序表的子程序，第十二章包含一些排序子程序，第十三章中的子程序则用来处理有序表。

其余各章中的子程序大部分使用DOS的资源(这里不管微处理器的类型)去完成它们的工作。第十四章中的子程序提供通用的输入和输出操作，如读和显示字符和串。

第十五章介绍时间和日期操作，第十六章中的子程序用于访问IBM PC中的ROM,以便实现DOS尚未提供的某些有用的任务。例如移动光标的子程序和清屏的子程序。

第十七、十八、十九章中的子程序用于完成有关磁盘驱动、子目录和磁盘文件的类似DOS的操作。第十九章还包括这样一些子程序，它们使你能实现文件的隐含和取消隐含，写保护和取消写保护。

第二十章的子程序用于实现对磁盘文件的操作，如建立文件、打开和关闭文件,从文件读取数据，向文件写入数据。

第二十一章所包含的四个子程序是不能归并到任何其它类别的。第一个子程序用于报告DOS的版本，第二个和第三个子程序用于读取和改变中断向量的内容，最后一个子程序用于报告你的计算机是否带有数学协处理器芯片(8087,80287或80387)。

本书最后有三个附录，附录A中的表格用于将十六进制数转换成十进制，或相反。附录B是ASCII字符集的一览表，这些字符都是你从键盘上和屏幕显示上可能遇到的。附录C按照字母顺序归纳了微处理器的指令系统。对于每一条指令列出了有效的汇编格式，并且指明了它所影响的那些状态标志。

补充材料

当起动 DOS 时，几个中断子程序被装入存储器。对程序员来说，最重要的 DOS 中断是 21H，它提供大量的功能调用选项号，我已经把一些最有用的功能调用放在这些子程序中。关于 DOS 及其功能调用的详细内容请参考 IBM 的 DOS 技术参考手册或者 Microsoft 类似的 MS—DOS 的程序员参考手册。关于你的计算机的微处理器及其指令系统的详细内容可向下列地点索取相应的程序员参考手册：

Intel Books, Inc.

P. O. Box 58130

Santa Clara, CA 95052—8130

在美国和加拿大你可以直接打电话给 Intel 公司。

最后，对于使用 80386 微处理器的读者，则建议参考 Penn Brumm 和 Don Brumm 编著的 " A programming and Design Handbook " 。

第一章 微处理机概述

本书中的程序可以在任何有 MS－DOS 的计算机上运行，而不管该计算机所采用的微处理器是 Intel 8088,8086,80186,80286，还是 80386。因为汇编语言程序设计要求读者具备微处理器方面的工作常识，所以这里首先概述这五种型号并指出它们的差异是有价值的。

1.1 从程序员的观点来看微处理器

市场上出售的 Intel 公司手册和大多数芯片说明书均详细描述了微处理器的系统结构。它们讨论如何产生地址，如何根据功能说明段（代码，数据，附加值和堆栈）来处理存储器，如何进行中断操作等等。因为我假定你自己已经有了这些书，而且已经读过它或多少了解它。所以，我省略了大部分技术细节。我直接从程序员的观点来归纳微处理器。也就是说，我将简要讨论程序员需要知道的细节，而不是硬件设计师所要知道的细节。

程序员可以认为 8086 和 8088 是相同的。它们具有同样的指令集，同样的内部寄存器，以及同样的状态标志等。两者的寻址能力均能达到一兆字节(1024K 字节)。它们之间的主要区别在于 8086 具有一组 16 位的数据总线，而 8088 具有一组 8 位的数据总线。当然，这实际上是一种硬件的特性，但它意味着如果这两个芯片在同样的速度下运行，那么 8086 在传送 16 位的字时，将比 8088 快一倍。

80186 基本上与 8086 相近，只是增加了几条新的指令以及改进了某些性能。它还包括了 15 个其它芯片的电路，使之成为一个"单片计算机"。

80286 的一个重要进展是它能工作在二种方式，即所谓实际地址方式和带保护的虚拟地址方式（简称保护方式）。在实际地址方式中，80286 的操作象 80186。在保护方式中，它提供了某些用于数据保护和存储管理的优异特性。保护方式最重要的特点在于允许 80286 利用所谓"虚拟寻址"方法去访问大容量的存储器。

具有虚拟寻址的 80286 提供两类存储器：物理地址空间和虚拟地址空间。物理地址空间是 286 可直接寻址工作的存储器，而虚拟地址空间则是相当大的可用的存储空间。物理地址空间可以大到 16 兆字节，虚拟地址空间可以大到十亿字节或一千兆字节。

80386 具有 80286 的所有特性，但它是 32 位微处理器，它比 80286 快（每秒能处理 3－4 百万条指令），而且能管理多个存储器。特别是 80386 的物理地址空间可以大到 4 千兆字节，它的虚拟地址空间可以大到 64 兆兆字节。

1.2 内部寄存器

图 1－1 指出了 8086，8088，80186 和 80286 所提供的 16 位寄存器组。80386 具有 32 位宽的寄存器，它的寄存器是那些老的寄存器的简单扩展，那里每个 16 位寄存器可以成

为相应的 32 位寄存器的低半部（右半部）。

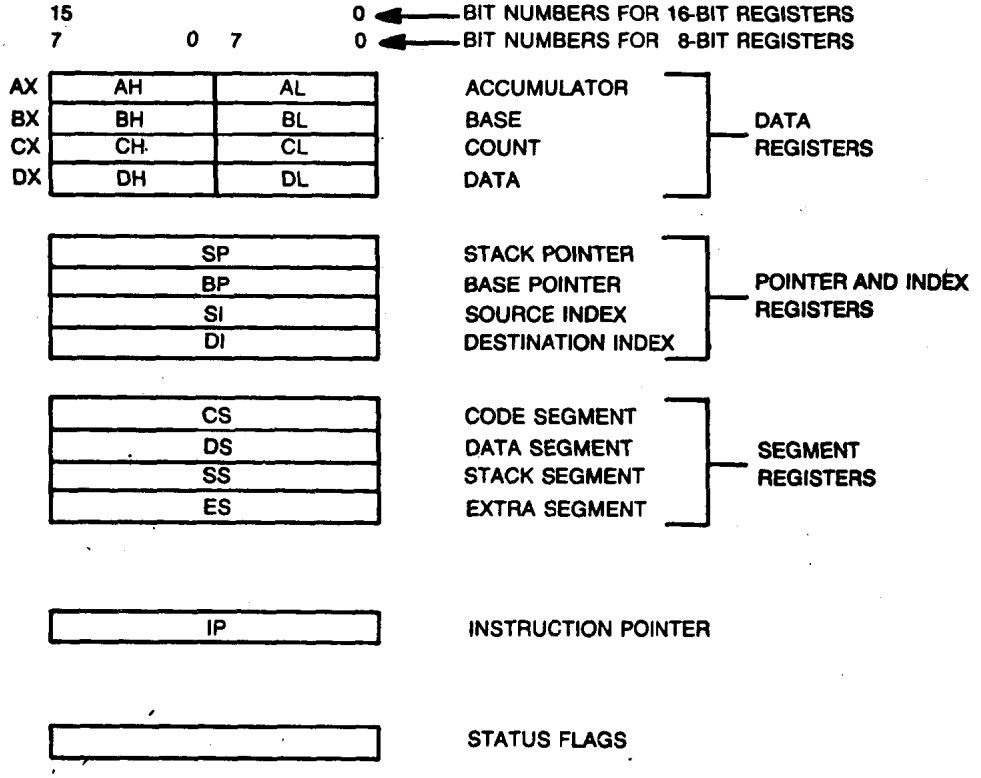


图 1—1 内部寄存器

注意，图 1—1 上部的几个寄存器按功能可以分成三组：数据，指针和变址，以及段。

数据寄存器

你可以把数据寄存器当作四个 16 位寄存器或八个 8 位寄存器来处理，这取决于对 16 位字还是对 8 位字节进行操作。16 位寄存器命名为 AX,BX,CX 和 DX。在这些带"X"的寄存器中包含名为 AL,AH,BL,BH,CL,CH,DL 和 DH 等一些 8 位寄存器。

注意：正如前面提到的，在 80386 中寄存器的字长是 32 位，早期的微处理器寄存器只有它的低 16 位。那些"扩展的"数据寄存器被称为 EAX,EBX,ECX 和 EDX。

AX 是累加器，它用于 16 位字长的乘法，除法，I/O 操作，以及某些串操作。它的低字节 AL 用于与上述相应的字节操作，以及用于转换和十进制算术运算操作。AH 寄存器也用于字节乘法和除法。

基址寄存器 B 通常用于在存储器中寻址数据。计数寄存器 CX 用作循环操作的重复次数计数器，以及用作串操作的元素个数计数器。CL 寄存器用于保存多位移位和旋转操作中的移位次数。

数据寄存器 DX 用于全字长的乘法和除法操作，它也能提供 I/O 操作中的端口号。

指针与变址寄存器

堆栈指针（堆栈指示器）SP 用于记录存储器内微处理器堆栈上的数据和地址。通常，

你在程序中从来不直接引用它。其它一些寄存器（BP，SI 和 DL）均用于对存储器中的操作数进行寻址。SI 和 DL 也用于串指令，它们可以分别作为"源"指针和"目的"指针。

段寄存器

段寄存器用于 EXE 类型的程序。但本书中只用 COM 类型的程序，它只用到一个段。

指令指示器(IP)

这是一个内部寄存器，它指向处理器下一次将要执行的指令。你在程序中从不引用 IP。

标志

16 位标志寄存器用于报告各种不同的状态，这些状态可以帮助你的程序作出判定，如图 1—2 所示。其中六位用于保存状态，另外三位允许你在程序中去控制微处理器的操作。

必须指出：

1、当 80286 工作在它的保护方式时，标志寄存器用了三个附加位，第 12 位至第 14 位。

2、在 80386 中，上述标志寄存器构成了 32 位的 IFLAGS(扩展标志)寄存器的低半部。

这里简单归纳一下标志寄存器中的各位：

1、进位标志(CF)。当一次加法产生进位或一次减法产生借位时，CF 为 1 。CF 也保存一个操作数在移位或旋转操作中曾经移出的位，以及作为乘法的一个结果指示器。

2、奇偶标志(PF)。如果一次操作的结果有偶数个 1 ，则 PF 为 1 。它主要用于数据通信。

3、辅助进位标志(AF)。它反应第三位的进位或借位输出。AF 对于压缩的十进制数操作也是有用的。

4、零标志(ZF)。如果操作的结果为零，则 ZF 为 1 。

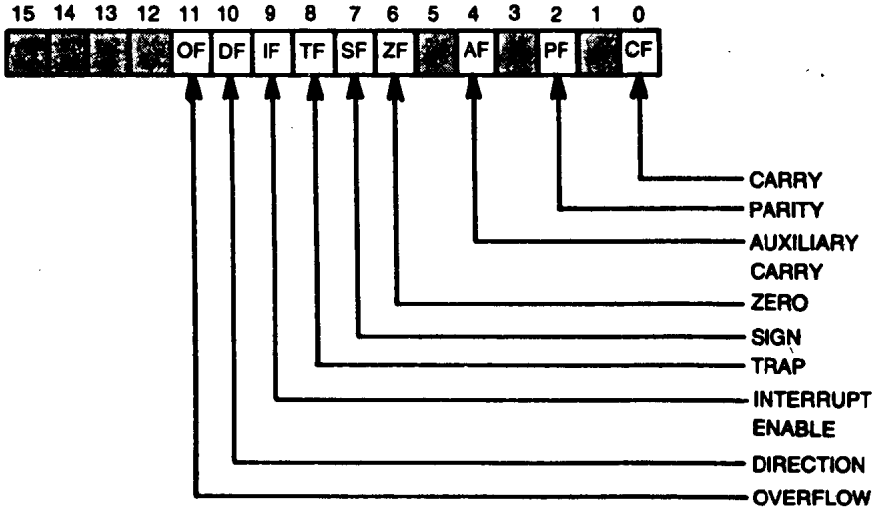


图 1—2　状态标志

5、符号标志(SF)。符号标志仅在对带符号数进行操作期间才有意义。如果结果的最高有效位为 1 ，即表示是一个负数，则 SF 为 1 。

6、陷阱标志(TF)。它使处理器按"单步"方式通过一个程序，可用于调试。

7、中断使能标志(IF)。当处理器认可中断请求时，IF 为 1 ；忽略中断请求时 IF 为零。
8、方向标志(DF)。它决定处理器在通过串时的前进方向；1 使它正向；0 使它反向。
9、溢出标志(OF)。它主要是带符号数操作期间的一个出错指示器。

1.3 寻址方式

所有五种处理器都提供同样的寻址方式。总共有七种寻址方式，简述如下：
1、寄存器操作数方式：操作数装入寄存器。例，INC BX。
2、立即操作数方式：操作数就是指令中包含的一个数。例，MOV CL,5。
3、直接方式：操作数的偏移量作为一个 8 位，16 位或（在 80386 中）32 位的数包含在指令中。例，MOV BX,TABLE。
4、寄存器间址方式：操作数的偏移量是 SI，DI，BX 或 BP。宏汇编程序要求用方括号括住寄存器名。例，MOV [BX],AX。
5、基址相对方式：操作数的偏移量是位移值和 BX 或 BP 的内容的总和。例，MOV AX,[BX]+4。
6、变址方式：操作数的偏移量是位移值和 SI 或 DI 的内容的总和。例，MOV AX,[SI]+4。
7、基址变址方式：操作数的偏移量是基址寄存器（BX 或 BP）的内容，变址寄存器（SI 或 DI）的内容，以及可选的位移值的总和。例，MOV AX,[BX] [SI]+2。

第二章 汇编程序的使用

当用汇编语言进行程序设计时，你可以用类似英语的缩写词来写一些指令。然后你可以运行汇编程序，将这些缩写转换成微处理器能够认识的数字代码。用缩写词缩写的程序被称为源程序，而与微处理器可兼容的数字形式的程序称为目标程序。因此，汇编程序的工作是将源程序转换成目标程序。

2.1 汇编语言指令

在源程序中，每一条汇编语言指令可以多到四个字段（又称域），如下：

[标号:] 助记符 [操作数] [;注释]

其中只有助记符字段是经常需要的。标号和注释字段通常是可选项——这就是为什麼要用方括弧来表示。你可以在一行上的任何地点输入这些字段，但是你至少必须用一个空格（或者一个制表符 TAB）来分隔它们。

标号字段

标号字段用一个名字赋给一条汇编语言指令。在汇编语言程序中，标号的作用与 BASIC 程序中行号的作用相同。

一条指令的标号可以长到 31 个字符，它必须用一个冒号(:) 来结尾。它可以包含字母 A 到 Z(或 a 到 z，在汇编程序中不分大小写)，数字 0 到 9，或者以下几个字符：

?，@，—，$

你可以用数字以外的任何字符作为标号的头一个字符。但如果你使用句号，则它必须是第一个字符。

助记符字段

助记符字段是包含 2 至 7 个字母的指令缩写词。汇编程序利用一张内部的表将每一个助记符翻译成它的等价的数字代码。

操作数字段

操作数字段告诉处理器到哪里去找要操作的数据。并非所有指令都需要操作数。那些需要操作数的指令可能需要一个或二个操作数。在助记符和操作数之间至少必须插入一个空格或制表符 TAB。如果一条指令需要二个操作数，则在它们中间必需放一个逗号。

在一条双操作数指令中，第一个操作数是"目的"，第二个操作数是"源"，源操作数所说明的值是指：微处理器应该加到它上面去，或者从其中减去，或者与它比较，或者要将它拷贝到目的操作数。例如：MOV 指令

```
MOV CX,DX
```

告诉处理器将寄存器 DX 的内容(源操作数)拷贝到寄存器 CX（目的操作数)。

注释字段

如象 BASIC 中的 REM，你可以利用这个可选字段来描述源程序中的一些语句，使得