80386/80286 汇编语言程序设计

索梅　郑甫京　张鹿　编著

清 华 大 学 出 版 社

(京)新登字 158 号

内 容 提 要

本书针对 80386/80286 微型计算机及 80386/80286 CPU 的结构、指令系统与伪指令，全面介绍了宏汇编语言程序设计的基本概念、方法和技巧；介绍了输入/输出与中断的概念、DOS 和 BIOS 功能调用方法及其有关的程序设计；另外，还介绍了 80387/80287 协处理器的基本结构和指令系统以及宏汇编语言与高级语言的接口技术等内容。

本书内容丰富、实用性强。书中所举的大量实例具有代表性，简明易懂，并都经过上机验证。

本书可作为高等院校计算机及有关专业的教材，或作为技术人员的培训教材，也可作为广大从事微型计算机科研、生产、教学和应用开发的科技人员的自学参考书。

图书在版编目(CIP)数据

80386/80286 汇编语言程序设计/索梅等编著. - 北京：清华大学出版社，1994

ISBN 7-302-01490-6

Ⅰ.80… Ⅱ.索… Ⅲ.微型计算机-汇编语言-程序设计 Ⅳ.TP313

中国版本图书馆 CIP 数据核字(94)第 02079 号

出版者：清华大学出版社（北京清华大学校内，邮编 100084）
印刷者：人民文学印刷厂
发行者：新华书店总店北京科技发行所
开　本：787×1092 1/16　印张：21.5　字数：503 千字
版　次：1994 年 6 月第 1 版　1994 年 6 月第 1 次印刷
本社分类号：TP·595
印　数：00001—10000
定　价：19.00 元

前　言

近年来,广为流行与普及的微型计算机,多数是以 80386/80286 甚至是 80486 为 CPU 的。在这样的 CPU 上,虽然可以用高级语言来编制任意的程序,但是为了更充分地利用计算机的硬件特性,更多的直接控制计算机的基本资源、编制占用内存少、执行速度快、效率高的程序,就必须使用汇编语言了。

80386/80286 CPU 在功能、结构及存储管理上均高于 8086/8088 CPU,用与之相对应的汇编语言编制程序,能够更充分地利用 80386/80286 的特性。鉴于国内介绍 80386/80286 汇编语言程序设计方面的书籍较少或内容不全面,而且许多程序设计人员至今仍停留在用 8086/8088 汇编语言的基础上编制程序,没有充分发挥 80386/80286 的功能等方面的考虑,我们编写了此书,目的是帮助读者学习和掌握 80386/80286 汇编语言程序设计,为开发自己的实用程序奠定基础。

本书注重实用性,力图做到通俗易懂、概念清楚和一题多解。在章节的编排上,从简到难,循序渐进地引入各种概念,尽量避免对未介绍的内容提前使用以致读者难于理解的情形。

书中通过各种程序设计实例向读者介绍了汇编语言程序设计的思想、方法和技巧,并结合编者多次讲授汇编语言程序设计课程的经验,对于难于理解、易出差错、容易混淆的内容和问题给予了明确的说明和解释。书中的所有实例都是精心挑选并经过上机验证的,读者可通过阅读它们来加深自己对一些概念的理解。

本书共分十章。第一章为汇编语言所要用到的基础知识,对已具有一定程序设计经验及数制基础的读者来说,可跳过这一章。第二章简要介绍了 80386 及 80286 的体系结构、存储管理及内存的寻址方法。第三章详细介绍了 80386/80286 的指令系统,并给出了各种指令在程序中使用的例子。第四章介绍了汇编语言源程序的基本结构和常用的伪指令、汇编语言上机过程及调试程序的方法,至此,读者应能编写出一些简单、实用的完整汇编语言程序并可上机实践。第五章介绍了宏汇编程序的伪指令及其使用方法并结合实例予以说明。第六章主要介绍汇编语言程序设计的基本方法和技巧,并通过各种实例充分给予解释。第七章和第九章介绍了输入/输出、中断的概念及 DOS 和 BIOS 系统功能调用的使用方法以及有关的程序设计。第八章介绍了 80387/80287 的功能结构和指令系统。第十章简单介绍了汇编语言与高级语言的接口技术。

因时间紧迫,水平有限,书中的错误在所难免,敬请读者指正,以利改进。

编　者

1993 年 10 月

目　　录

第一章　基 础 知 识

为方便读者学习 80386/80286 汇编语言，我们先介绍一些与汇编语言有密切关系的基本知识，为后续章节的学习打下良好的基础。本章主要介绍微型计算机的基本结构、汇编语言的基本概念、使用汇编语言编程的目的、计算机的数据表示形式及 80386/80286 汇编语言处理的数据类型等内容。

1.1　微型计算机的基本结构

微型计算机与传统的大、中、小型计算机一样，是由五个基本部分组成的，它们是：输入设备、输出设备、运算器、控制器和存储器。其中：

1. 输入和输出设备：是实现人与计算机之间信息交换所必须的部件。

输入设备负责将用户要处理的数据及程序等信息输入到计算机存储器中。常用的输入设备有终端键盘、光电扫描仪等。

输出设备负责将处理后的结果或其它信息进行输出显示。常用的输出设备有屏幕终端、打印机、绘图仪等。

2. 存储器：用来存储计算机要处理的程序和数据，并将运算的中间结果以及处理后的结果储存起来。

在计算机系统中，存储器能储存的全部信息称为存储容量。从使用者的观点来看，存储器的容量越大越好，存取信息的时间越短越好。80286 CPU 能提供 24 位的地址，所以它可配置容量最多达 16 兆(2^{24})字节的存储器。80386 CPU 提供了 32 位的地址，它可配制容量最大为 4 千兆(2^{32})字节的存储器。

存储器分为随机存储器(RAM)和只读存储器(ROM)两种。RAM 既可以写入数据也可以从中读出数据，但断电后 RAM 不再保留原信息。RAM 主要用来存放用户数据和程序。

ROM 每个单元的信息是固化了的，用户可从中读出信息，但一般不能改变其信息。这种存储器中的信息在断电后不会丢失，只要再通电就又可读出原来的信息。ROM 主要用来存放计算机自身管理的系统程序，它可分成两种：PROM 和 EPROM。前者是可编程的只读存储器，后者是可擦除的可编程只读存储器，但这种擦除必须使用特殊工具，在特殊情况下才能将原信息更改和擦除。

3. 运算器：简称为 ALU，它是按指令对数据进行算术运算和逻辑运算的处理部件。ALU 通常由累加器和各种寄存器组成，用以暂存数据和完成各种最基本的算术运算和逻辑运算操作，同时也可在控制器的控制下，根据指令要求将处理结果送到存储器或输入/输出设备。

4. 控制器：通常由指令寄存器、指令计数器、译码器和各种控制线路组成。控制器完

成所有的输入/输出操作,以及对运算器的控制。它从存储器中读取程序指令,经过译码向计算机各部分发出执行微操作的控制信号,使指令得以执行。因此,控制器可实现对输入/输出设备、存储器和运算器的控制和管理,它将原始数据从输入/输出设备或存储器中取出后送到 ALU,又将 ALU 加工处理的结果送到存储器或输入/输出设备中去。

1.2 汇编语言的基本概念

1.2.1 汇编语言

汇编语言是一种面向机器的程序设计语言,它是一种低级语言。它直接利用机器提供的指令系统编写程序,该类程序的可执行指令是与机器语言程序的指令一一对应的。可以说,汇编语言是对机器语言的符号化描述。即:汇编语言的指令是用助记符来表示相应的用二进制数形式描述的机器语言指令的,因此便于记忆和使用。

1.2.2 汇编语言源程序

用汇编语言指令编写的程序称为汇编语言源程序或汇编源程序。通常,在以 DOS 为操作系统的 80386/80286 CPU 系列微型计算机上,汇编源程序以.ASM 为扩展名,这是汇编程序默认的扩展名。

1.2.3 目标程序

目标程序也称为机器语言程序,这是计算机 CPU 真正可以执行的程序。这种程序的指令都是用二进制代码来表示其操作码和操作数的机器语言指令。

由于汇编语言的指令是用助记符来表示机器语言指令,因此,汇编源程序同高级语言源程序一样,是不能直接运行的,必须将它翻译成目标程序(机器语言程序)才能在计算机中运行。

1.2.4 汇编程序

汇编程序是将汇编源程序转变为相应目标程序的翻译程序。这个转变过程称为汇编。

由于汇编指令助记符与机器语言指令是一一对应的等价关系,所以汇编程序很容易将汇编源程序迅速、准确、有效地翻译成目标程序。

在本书中,我们主要介绍 DOS 操作系统支持下的宏汇编程序 MASM。

用 MASM 宏汇编语言编写的汇编源程序的盘文件应取扩展名为.ASM。

1.3 为什么要用汇编语言编写程序

汇编语言是面向机器的程序设计语言,与具体的计算机硬件有着密切的关系,因此,用它们编写出的程序只适用于某一系列的计算机,可移植性差。但由于汇编指令与机器语言指令一一对应,即一条汇编语言的可执行指令对应着一条机器语言指令,反之亦然。因此,汇编语言可直接利用机器硬件系统的许多特性,如寄存器、标志位以及一些特殊指

令等,执行速度快、占用内存少。

而高级语言(如 PASCAL、FORTRAN、BASIC)是面向问题的,它与机器的硬件无关,可以在各种不同的计算机上运行,因此可移植性好。但是用高级语言编写的程序是不能直接执行的,需要由编译程序或解释程序将它们翻译成对应的目标程序,机器才能接受。由于高级语言指令与机器语言指令不是一一对应的,往往一条高级语言指令要对应着多条机器语言指令,因此,这个翻译过程要比将汇编源程序翻译成目标程序花费的时间要长的多,产生的目标程序也较冗长,占用存储空间大,执行的速度也较汇编语言慢。虽然采用高级语言编写程序可以节省软件开发的时间,但它不允许程序员直接利用寄存器、标志位等这些计算机硬件特性,因而影响了许多程序设计技巧的发挥。

用汇编语言编写程序可以充分发挥机器硬件的特性,提高编程的质量和运行速度。但由于汇编语言要求对计算机的组成部分进行操作,因此学习汇编语言程序设计除了要掌握机器的指令系统外,还要熟悉机器的内部结构,特别是中央处理器(CPU)和存储器的结构,以及与编程有关的其它部分(芯片)的结构,如中断系统、视频显示、键盘接收、定时功能、通信等。

是否采用汇编语言编写程序,要视具体的应用场合而定。一般来说,某些对时间和存储器容量要求较高的程序用汇编语言来书写,如系统软件、实时控制系统、智能化仪器仪表软件等。

1.4 计算机数据表示

人们在日常的生活和工作中,习惯于用十进制进行计数,但计算机则不同,它采用的是二进制计数法。在计算机中的所有运算和判断都是通过二进制体现的,从数据到指令、从地址到内容也均是用二进制来表现的。但是二进制数书写起来太长、不方便,故在计算机应用中也常使用十六进制数,特别是在汇编语言程序设计中使用的更广泛。

本节是为了那些不熟悉数据的这些表示法的读者而写的,对已熟悉它们的读者可跳过这一节。

1.4.1 二进制数的表示

我们知道,任意一个十进制数上的每一位数,都可以取 0～9 这十个数码中的一个,并且是“逢十进一”的。而二进制数每个数位只可能取两个不同的数码“0”或“1”,并且是“逢二进一”的。

正如一个十进制数 2345 可表示成如下形式:

$$2345 = 2\times 10^3 + 3\times 10^2 + 4\times 10^1 + 5\times 10^0$$

一样(10 为基数),二进制数 1010 可表示成:

$$(1010) = 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 0\times 2^0$$

的形式(2 为基数)。而 2^i(i 为正整数)称为每一位数字对应的权。所以二进制数 1010 自左至右每一位数字对应的权分别为 2^3、2^2、2^1、2^0。

为了与十进制数相区别,我们常在二进制数后面写一个字母 B。例如:

1010B

而在十进制数后面写上字母 D, 例如:

2345D

但一般对十进制数来说, 都省略字母 D。

由于二进制只取两个数码 0 和 1, 因此二进制数的每一数位都可以用任何具有两个不同稳定状态的元件来表示, 而制造具有两个稳定状态的元件比制造多稳定状态的元件容易的多, 并且也最节省。比如电灯的亮和灭、继电器的闭合和断开、晶体管的截止和导通、脉冲的有无、电位的高低等等。只要规定其中的一种状态为“1”, 另一种状态为“0”表示, 就可用它们表示二进制了。

另外, 二进制数的运算很简单, 但要记住是“逢二进一”的。

二进制加法表示为:

0 + 0 = 0

1 + 0 = 0 + 1 = 1

1 + 1 = 10

二进制乘法表示为:

0* 0 = 0

1* 0 = 0* 1 = 0

1* 1 = 1

下面举例说明二进制正整数的算术运算规则:

```
   111B          1101B
 + 011B        - 110B
  1010B          111B

   10111B          11001B ———商
 *  1010B     111B|10110001B
  00000B          111
 10111 B          1000
00000  B           111
10111   B          1001
11100110B           111
                     10B ———余数
```

在这几个例子中, 凡是向前产生的进位都是“逢二进一”的, 而产生的借位, 借过来的都是 2。

1.4.2 十六进制数的表示

由于二进制数书写起来很长, 念起来不易懂, 不方便, 因此我们常用十六进制来表示。为区别起见, 十六进制数后面用字母“H”作标记。

十六进制数以 16 为基数，每一数位都有十六种状态，即用数码 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F 来表示，恰好 4 位二进制数可以对应一位十六进制数，其中 A～F 表示十进制的 10～15，并且是“逢十六进一”的。

例如：十六进制数 308BH 可表示成如下形式：

$$308BH = 3 * 16^3 + 0 * 16^2 + 8 * 16^1 + B * 16^0$$

十六进制算术运算举例：

```
  85A0H        95E8H        4BH          4BH
+ 1048H      - 1048H      * 13H      13H|591
  95E8H        85A0H        E1          4C
                          4B           D1
                          591H         D1
                                        0
```

表 1-1 列出了十进制、二进制和十六进制之间的相互关系。

表 1-1　十进制、二进制与十六进制数字对照表

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

有一点请读者注意：汇编语言中，凡是字母 A～F 打头的十六进制数，都要以数字“0”作为开头，以避免与标识符相混淆。例如：十六进值数 B0H 应写成 0B0H。

1.4.3　数制转换

在计算机内部处理的数据都是二进制数，但人们编程时，往往为了使用上的方便，需要用到一些其它进制的数据，所以，为了用不同进制正确表示一个数据，就需要掌握各进制数之间的转换方法。

1. 二进制、十六进制数转换成十进制数

在上面对二和十六进制数的讨论中，我们分别用按权相加的方法对两种进制的数据进行了表示，这实际上也就是将它们转换成十进制数的方法。例如：

$$101011B = 1 * 2^5 + 0 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 = 43$$

$$58D6H = 5 * 16^3 + 8 * 16^2 + D * 16^1 + 6 * 16^0 = 22742$$

从中我们不难看出，不管是哪一种进制的数据，在将它们转换成十进制数时，都是以

该进制作为基数,然后将它们的各数位按权相加,结果便是其对应的十进制数。这种转换比较容易和方便,反之,若将一个十进制数转换成某一进制数,则该过程要复杂一些。

2. 十进制数转换成二进制数

下面我们以具体例子来说明这种转换。

例: 将十进制整数 365 转换成对应的二进制数。

设: $$365 = j_n j_{n-1} \cdots j_1 j_0 B$$

现在的问题是要决定 $j_n, j_{n-1}, \cdots, j_1, j_0$ 的数值。因为是二进制,所以它们只能是 0 或 1,下面我们将逐个地确定 $j_n, \cdots, j_0$ 的值。

由于任何一个二进制数都可以写成按权的展开式,因此也可将上式写成:

$$365 = j_n \cdots j_0 B = j_n * 2^n + \cdots + j_0 * 2^0$$
$$= 2(j_n * 2^{n-1} + \cdots + j_1) + j_0$$

等式两边同时除以 2 得到:

$$365/2 = 182 + 1/2$$
$$= (j_n * 2^{n-1} + j_{n-1} * 2^{n-2} + \cdots + j_1) + j_0/2 \qquad (1)$$

由于等式两边整数与小数必须对应相等,所以(1)式表明 $j_0 = 1$, 它正好是 365/2 的余数。

再将(1)式写成:

$$365/2 - j_0/2 = 2(j_n * 2^{n-2} + \cdots + j_2) + j_1$$

即:
$$(365 - 1)/2 = 182$$
$$= 2(j_n * 2^{n-2} + \cdots + j_2) + j_1$$

同样,将该式两边除以 2, 可知 j_1 为 182/2 的余数,即 $j_1 = 0$。

用类似的方法继续下去,就可将 $j_n, j_{n-1}, \cdots, j_1, j_0$ 都确定下来。现将整个过程与结果表示如下:

2	365	
2	182	余数 = 1 = j_0
2	91	余数 = 0 = j_1
2	45	余数 = 1 = j_2
2	22	余数 = 1 = j_3
2	11	余数 = 0 = j_4
2	5	余数 = 1 = j_5
2	2	余数 = 1 = j_6
2	1	余数 = 0 = j_7
	0	余数 = 1 = j_8

因此,转换结果为:

$$365=j_8j_7j_6j_5j_4j_3j_2j_1j_0B=101101101B$$

这种转换我们可以简单地记为:“除以 2 取余数,逆序排列得整数。”

3. 十进制数转换成十六进制数

这种转换的方法与将十进制数转换成二进制数类似,可以记为:“除以 16 取余数,逆序排列得整数。”

4. 二进制数与十六进制数之间的转换

因为 $2^4=16$,所以一位 16 进制数相当于四位二进制数,这样,二进制数与十六进制数之间的转换就极为方便:从二进制数转换成十六进制数时,只要将二进制数从右到左每四位用一个十六进制数表示即可;从十六进制转换成二进制时,只要将每位十六进制数用四位二进制数表示即可。例如,二进制数转换成十六进制数:

1010110110B= 10, 1011, 0110B= 2B6H

— — — — —

↓ ↓ ↓

2 B 6

十六进制数转换成二进制数:

67AB5H= 0110, 0111, 1010, 1011, 0101B= 01100111101010110101B

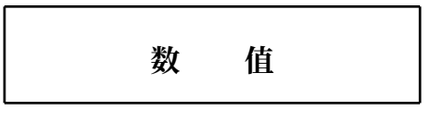
— — — — — — — — — —

↓ ↓ ↓ ↓ ↓

6 7 A B 5

1.4.4 带符号二进制数的表示

到目前为止,我们所说的二进制数均为无正、负符号的数,不管这个数含有多少个二进制数位,这些数位都是该数中的一个组成部分。例如,8 位无符号数的形式为:



7 0

数值范围:00～0FFH,即 0～255。

16 位无符号数的形式:

15 0

数值范围:0000H～0FFFFH,即 0～65535。

32 位无符号数形式:

31 0

数值范围: 00000000H～0FFFFFFFFH, 即 0～4294967295 或 0～2^{31}- 1。

由于二进制数上每一数位只有 0 和 1 两种状态, 因此, 对于一个带有正号“ + ”或负号“ - ”二进制数的符号来说, 也只能用这两种不同的状态来区别。在机器内部, 符号实际上是被“ 数码化 ”了。

带符号数在机器中的表示, 通常是利用该数的最高有效二进制位来作为符号位(S), S 为 0 则表示该数为正数, S 为 1 则表示该数为负数。

由于符号位占据了一个二进制位, 所以, 这样的二进制数比相对应位数的无符号二进制数来说, 其数值范围减少了一半。例如:

8 位有符号数表示形式为:

数值范围: 80H～7FH, 即- 128～+ 127。

16 位有符号数表示形式为:

数值范围: 8000H～7FFFH, 即- 32768～+ 32767。

32 位有符号数表示形式:

数值范围: 80000000H～7FFFFFFFH, 即- 2147483648～+ 2147483647。

对于一个带符号的数(尤其是负数), 它在机器内部如何表示呢? 请看下面的例子是否正确:

+ 5 的 8 位二进制形式: 00000101—— —— —— ——— (1)

- 5 的 8 位二进制形式: 10000101—— —— —— ——— (2)

由于带符号数的最高有效位是符号位(S), 所以(1)式是正确的, (2)式错误的原因是因为在机器内部, 对任意一个带符号的负数来说, 都是按补码的形式来存储和处理的。请看下面的介绍。

1.4.5 数的补码表示

为得到一个负数数值的补码形式, 方法可以是先写出该负数所对应正数的二进制形式, 使该正数的每一个二进制位变反(即 0 变 1, 1 变 0), 然后再将变反后的结果加 1, 最后得到的就是这个负数的补码形式。

例如: 对于- 5 来说, 其所对应正数的二进制形式为: