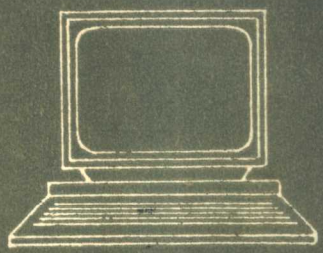
8086/8088



汇编语言实用程序

刘全忠　编著

天津大学出

8086／8088

汇编语言实用程序

刘全忠　编　著

天津大学出版社

内 容 简 介

本书是一本微型计算机8086/8088汇编语言实用程序集。第一部分通过对分支、循环、逻辑操作、字符串、过程、输入/输出、中断控制以及定时/计数电路等各种程序结构的研究，较系统地介绍了8086/8088汇编语言程序设计和调试的方法及技巧。同时也介绍了几种常用接口芯片的使用方法。第二部分较全面地介绍了在实时控制和检测中常用的各种多精度数据运算、码制转换、定时及滤波等实用子程序。所有程序都附有详细的硬件电路图，程序清单，操作数据及执行结果。全书通俗易懂，便于直接选用，实用性强。

本书可供从事微型计算机应用的科技人员阅读、使用，也可做为高等院校有关专业师生的参考教材。

(津)新登字012号

8086/8088
汇编语言实用程序
刘全忠 编著

天津大学出版社出版
(天津大学内)
河北省永清县印刷厂印刷
新华书店天津发行所发行

开本：787×1092毫米 1/16 印张：12 3/4 字数：315千字
1992年10月第一版 1992年10月第一次印刷
印数：1—6000
ISBN 7-5618-0368-0
TP·39
定价：7.00元

前　言

8086是美国Intel公司最早推出的16位微型计算机芯片。随后推出的8088，在软件上和8086完全兼容，它们配备有齐全的接口电路和丰富的软件，是目前国际上最流行的16位机之一，也是当前国内广泛采用的机型。

8086/8088和8位机相比，具有更高的可靠性和执行速度，并能容易地组成计算机网络，因而目前已广泛应用在文字处理、行政管理等事务工作中。可以预料，随着微机应用的深入和功能的进一步发挥，它必将在实时数据采集、自动控制和信号处理等各个领域得到广泛应用。在这些场合，使用汇编语言比任何高级语言在编程效率和执行速度上都优越得多。实践证明，掌握汇编语言程序设计是微机应用的一个基本技能。

本书全部的8086/8088汇编语言程序分两部分介绍。第一部分包括数据传送、算术运算、分支与循环、逻辑操作、数据串操作、过程、输入输出，中断及硬件定时计数等九章。它们以实验形式给出，着重程序的分析，并研究程序内部的执行过程，以培养设计、调试程序的能力，掌握初步的编程技巧。第二部分包括后四章，分别介绍了在工业控制和实时信号处理系统中常用的数据运算、转换及滤波等实用程序，并全部以子程序形式给出，用户可直接调用。

本书后三章及附录部分由天津纺织工学院王守茂编写。全部程序均经上机调试通过。由于作者水平有限，所列程序不一定是最佳程序，只供读者在使用或编程时参考，对其中不妥或错误之处敬请批评指正。

作　者

1991年6月于天津大学

目　录

第一部分

第二部分

第一部分

第一章 数据传送

数据传送是计算机最频繁、最基本的操作，8086/8088具有14种数据传送指令，其中包括存贮器和寄存器，寄存器（AL和AX）和I/O端口之间的字节或字的传送，以及传送状态标志和加载段寄存器的堆栈操作等指令。由于一个程序有相当大的部分是进行数据传送操作，因而，使用这类指令的熟练程度和技巧如何，将极大地影响程序的质量。

对于涉及到存贮器的数据传送，可以采用多种寻址方式，这就给用最少的指令实现各种繁杂的数据传送操作提供了充分的条件。本章的程序仅仅使用了直接寻址、寄存器寻址和变址寻址，其余的寻址方式在以后各章将陆续介绍。

8086/8088具有20条地址线，可以寻址2^{20}个字节（1M）的存贮器空间，但是，由于8086/8088只能处理16位的地址码，因而，为了对1M空间寻址，采用了用段寄存器CS、DS、SS和ES进行分段寻址的措施，即每个段寄存器（16位）指定64K空间，而在64K内部用指令计数器IP或堆栈指示器SP寻址。熟练地使用段寄存器是设计高质量汇编语言程序所必需的技能。本章将对其中的代码段寄存器CS和数据段寄存器DS的使用作简要的说明，其余的段寄存器将在以后的章节中介绍。

最后要指明的是，数据传送指令并不是本章所独有的，在以后各章的所有程序中，都要使用这类指令。因而，读者不必期望通过本章有限的程序就能掌握全部数据传送的技能。无疑，掌握这种技能是一个比较长期的任务。

§1-1 段寄存器的简单使用

程序1-1 代码段寄存器CS的使用

目的 在交换存贮器01000H单元和01050H单元内容的程序中，研究CS的使用方法。

实验步骤

表1-1 存贮单元交换程序一

行	地址(Hex)	内容(Hex)	助记符	注解
010	2000	A00010	MOV AL, 1000H	; 送数
020	2003	8A1E5010	MOV BL, 1050H	
030	2007	A25010	MOV 1050H, AL	; 交换
040	200A	881E0010	MOV 1000H, BL	
050	200E	F4	HLT	
060	1000	6A		; 数据
070	1050	7B		

(1) 复位，送入表1-1所列程序和数据。

(2) 复位，用GO 0:2000执行程序。复位后检查偏移量为1000H单元的内容是7BH，1050H单元的内容是6AH。

(3) 用GO 100:1000执行程序。复位后检查偏移量为1000H单元的内容是6AH，1050H单元的内容是7BH。

(4) 复位，用ST 2000单步执行程序。并检查CS（代码段基址）和IP（偏移量），计算出实际地址。

单步	CS	IP	实际地址
1	0000	2003	02003
2	0000	2007	02007
3	0000	200A	0200A
4	0000	200E	0200E
5	0000	200F	0200F

(5) 复位，检查CS=0，修改为CS=100H，用ST 1000单步执行程序。并检查CS和IP，计算出实际地址。

单步	CS	IP	实际地址
1	0100	1003	02003
2	0100	1007	02007
3	0100	100A	0200A
4	0100	100E	0200E
5	0100	100F	0200F

程序分析

程序本身比较简单，前四条指令都是用存贮器直接寻址方式进行数据传送。010行指令将01000H单元的内容送入AL，020行指令将01050H单元的内容送入BL，030和040行指令分别将AL和BL的内容送入01050H和01000H单元。

执行程序时使用的命令格式为：

$$\text{GO} \quad \underbrace{\times\times\times\times}_{\text{CS}} : \underbrace{\times\times\times\times}_{\text{IP}}$$

步骤(2)执行命令使段基址CS=0000H，偏移量IP=2000H，CPU从如下计算的实际地址处开始执行程序的指令代码：

$$\begin{array}{rl} & \text{CS} \quad \boxed{0\,0\,0\,0}\,0 \\ + & \text{IP} \quad \boxed{2\,0\,0\,0} \\ \hline & 0\,2\,0\,0\,0 \leftarrow \text{实际地址} \end{array}$$

无疑，设计的程序应该从存贮器的02000H单元开始安排，CPU顺序执行表1-1所列出的程

序，当执行完第040行指令，完成01000H和01050H单元互换之后，在050行执行动态停机。

步骤（3）执行命令使CS = 0100H，IP = 1000H，则实际地址为：

$$\begin{array}{rl} \text{CS} & \boxed{0\ 1\ 0\ 0}\ 0 \\ +\ \text{IP} & \boxed{1\ 0\ 0\ 0} \\ \hline & 0\ 2\ 0\ 0\ 0 \end{array}$$

步骤（3）与步骤（2）的实际地址完全相同，因而，执行的是同一个程序，其结果必然完成同一任务。

步骤（4）、（5）是单步执行程序的情况，单步执行的命令格式为：

$$\text{ST}\underbrace{\times\times\times\times}_{\text{IP}}$$

偏移量IP的段基址CS由复位置0，如步骤（4）；或者专门设置，如步骤（5）。从单步执行中可以发现，CS的内容没有改变，而每执行一条指令，IP的内容（偏移量）根据指令字长进行修改，以便指向下一条指令的地址。

段基址和偏移量叫做逻辑地址，从上述分析可知，不同的逻辑地址可以指定同一个实际地址，如图1-1所示。另一方面，如果改变CS中的段基址，也可以使程序的起始实际地址相应改变。这样，就可以把编好的程序安放在存贮器的任一区域，而不必修改程序中的任何代码。例如，如果把表1-1的程序安置在存贮器03000H—0300EH中，只要按命令键：

GO 100:2000

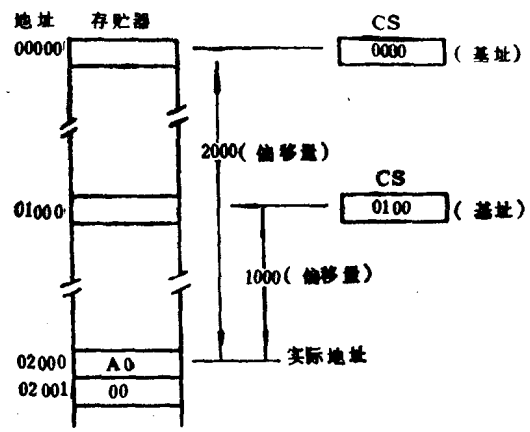


图1-1 多个逻辑地址可指定一个实际地址

便可执行表1-1的程序。这种特性叫可浮动再定位。在调试程序，尤其是在多道程序和多任务系统中很有用处。

为方便起见，编者声明：

（1）本书以后所有程序，如没有专门指出，CS的值由复位操作预置为0。

（2）表1-1和本书以后所有程序表中列出的地址，实际上是相对CS的偏移量。由于(1)中规定CS = 0000H，因而只要在偏移量前面添一个零，则为实际地址。如表1-1中列出的地址2000H，它为偏移量，而02000H则为实际地址。

程序1-2 数据段寄存器DS的使用

目的 在交换存贮器01000H单元和01050H单元内容的程序中，研究DS的使用方法。

实验步骤

（1）复位，送入表1-2所列程序和数据。

（2）复位，在2012H处设断点，执行程序，检查如下内容为：

表1-2 存贮单元交换程序二

行	地址(Hex)	内容(Hex)	助记符		注解
010	2000	B80000	MOV	AX, #0	; 设置DS
020	2003	8ED8	MOV	DS, AX	
030	2005	BE0010	MOV	SI, #1000H	; 建立指针
040	2008	8A04	MOV	AL, [SI]	; 送数
050	200A	8A5C50	MOV	BL, 50[SI]	
060	200D	884450	MOV	50[SI], AL	; 交换
070	2010	881C	MOV	[SI], BL	
080	2012	F4	HLT		
090	1000	6A			; 数据
100	1050	7B			

DS	SI	01000	01050
0000	1000	7B	6A

(3) 将010行指令改为MOV AX，#0010H（即2001H单元内容改为10H），030行指令改为MOV SI，#F00H（即2007H单元内容改为0FH）。复位，在2012H处设断点，执行程序。检查如下内容为：

DS	SI	01000	01050
0010	0F00	-6A	7B

程序分析

本程序和表1-1所列程序都是用来交换两个存贮单元的内容，操作过程也完全相同，所不同的是，本程序中040、070行指令存贮器用寄存器间接寻址，而050、060行指令用变址寻址进行数据传送。

无论是寄存器间接寻址、变址寻址，还是表1-1中的直接寻址，所要寻址的存贮器实际地址都由数据段寄存器DS和指令中规定的有效地址决定。在步骤（2），程序中010、020行指令设置DS = 0000H，则040行SI间接寻址的指令，按如下方法计算存贮器实际地址：

```
          ↙段基址
  | 0 0 0 0 | 0
+ SI  | 1 0 0 0 |  ←有效地址
-------------------
    0 1 0 0 0
```

并将该地址中的内容送入AL。而050行变址寻址指令中的实际地址为：

```
          ↙段基址
DS  | 0 0 0 0 | 0
SI    | 1 0 0 0 |  }
+ 位移量      5 0  }有效地址
-------------------
    0 1 0 5 0
```

该指令将01050H单元中的内容送入BL。可以看出，060、070行指令中的实际地址与上述计算方法一样，这里不再重复。

步骤（3）中，设置DS＝0010H，SI＝0F00H，因而040、050行指令中的实际地址分别为：

```
DS  0 0 1 0  0          DS  0 0 1 0  0
+ SI  0 F 0 0           SI  0 F 0 0
-------------           + 位移量  5 0
  0 1 0 0 0             -------------
                          0 1 0 5 0
```

可见，指令操作所需的存贮器实际地址和步骤（2）完全一样。因而，不难得出这样的结论：寻址存贮器变量时，其实际地址由数据段寄存器DS作为基址和以有效地址作为偏移量来决定，不同的基址和有效地址（称为逻辑地址）可以指向同一个存贮单元，如图1-2所示。

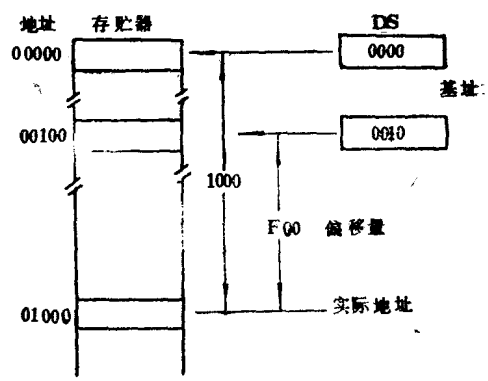


图1-2　存贮器变量的实际地址

另一方面，如果修改DS中的基址，也可以使存贮器变量（数据）移到存贮器的任何地方，而不必修改程序中的其它任一代码。例如，如果把表1-2程序中的DS值设置为0200H（即2002H单元的内容修改为02H），那么，该程序要变换的两个存贮单元的数据就可移到03000H和03050H单元，空出原来的01000H和01050H单元可另做它用。和前一程序分析过的代码段寄存器CS相对照，改变CS可以重新安排指令代码，而改变DS可以重新安排数据（存贮器变量），从而使整个程序具有可浮动再定位性。

实际上，执行程序前的复位操作，已设置DS＝0000H，因而，如程序中不另设置DS值便可认为它的值为零，这点请读者在阅读本书以后的程序时要特别注意。

8086除了代码段CS和数据段DS寄存器外，还有堆栈段寄存器SS和附加段寄存器ES，它们的使用方法将分别留在子程序和串操作两章中介绍。

§1-2　存贮单元内容互换

程序1-3 交换两个存贮单元的内容

目的　使用XCHG指令，实现存贮器01000H单元和01050H单元的内容互换。

实验步骤

（1）复位，送入表1-3所列程序和数据。

（2）单步执行程序，检查如下内容为：

单步	IP	AL	01000	01050
1	2003	6A	6A	7B
2	2007	7B	6A	6A
3	200A	7B	7B	6A

表1-3 交换两个存贮单元程序三

行	地址(Hex)	内容(Hex)	助记符	注解
010	2000	A00010	MOV AL, 1000H	; 送数
020	2003	86065010	XCHG AL, 1050H	; 交换
030	2007	A20010	MOV 1000H, AL	
040	200A	F4	HLT	
	1000	6A		; 数据
	1050	7B		;

程序分析

本程序是使用交换指令XCHG实现两个存贮单元互换的，XCHG指令类似MOV指令，但它不是将源送入目标中，而是两个操作数互换，这两个操作数不允许是立即数，而且至少有一个是寄存器。因而，实现两个存贮单元的互换必须用一个寄存器作为暂存单元，本程序是用AL作为暂存单元的。如单步执行程序表明的情况，首先将01000H单元的内容送入AL，然后用XCHG指令使AL与01050H单元的内容互换，第三步将AL的内容再送入01000H单元，完成两个存贮单元内容的互换。与表1-1和表1-2所列程序相比，本程序只需要三条指令和一个寄存器。显然，优于前两个程序。

第二章　算术运算

使用汇编语言编制数据处理程序，无论表达式如何复杂，最终都要化为加、减、乘、除四则运算形式，而8086/8088不但具有加减指令，同时也具有乘除指令，这就给编制数据处理程序提供了极大的方便，在这点上8位机是无法比拟的。

8086/8088算术指令允许用四种不同类型的数进行运算，它们是：不带符号二进制数，带符号二进制数（补码形式），不带符号的压缩十进制（BCD码）数和不带符号的非压缩十进制（BCD码）数，这些类型数的四则运算都将是本章研究的重点。

执行算术运算指令以后，计算的结果按其特征将影响状态标志位，其中大部分状态标志位可以用条件分支指令进行检测。因而，在使用算术运算指令时务必要注意它们对状态标志位的影响，关于算术运算指令和条件分支指令的组合使用，已不属本章的内容，将在下一章叙述。

§2-1　二进制加减运算

程序2-1　单精度（16位）加减运算

目的　计算 $(x)+(y)-(z)+100$，并将结果存入 (w)

存贮器分配　见程序2-1图。

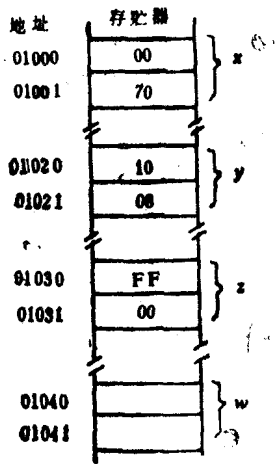
实验步骤

(1) 送入表2-1所列程序，按存贮器分配图送入数据。

(2) 单步执行程序，检查地址偏移量和AX的内容为

单步	地址	AX
1	2003	7000
2	2007	7810
3	200B	7711
4	200E	7775
5	2011	7775

检查存贮器01041H和01040H单元的内容为7775H。



程序2-1图

表2-1　单精度加减运算程序

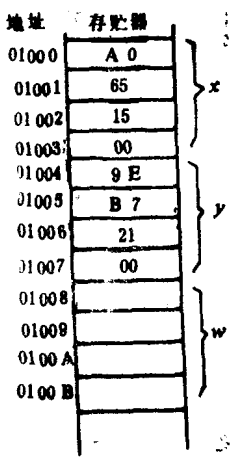
行	地址(Hex)	内容(Hex)	助记符	注　解
010	2000	A10010	MOV	AX, 1000H; 送(x)
020	2003	03062010	ADD	AX, 1020H; (x)+(y)
030	2007	2B063010	SUB	AX, 1030H; (x)+(y)−(z)
040	200B	056400	ADD	AX, #100; (x)+(y)−(z)+100
050	200E	A34010	MOV	1040H, AX; 送结果到(w)
060	2011	F4	HLT	

程序分析

程序中010行指令将01000H、01001H单元的内容送入AX，020、030行指令使AX与01020H、01021H单元的内容相加，并减去01040H、01041H单元的内容。这三条指令均在指令中给出存贮器地址的偏移量，因而为直接寻址方式。这种寻址方式可以在数据段内对64 k 内存进行寻址。040行指令是AX与立即数100相加，050行指令将AX的内容用直接寻址方式存入01040H、01041H单元。上述操作过程，从单步操作的记录中可以明显地看到。

程序中的操作数 x、y、z、w 均为一个字长（16位），这样的算术运算叫做单精度算术运算，它只能表示和容纳64 k 个不同的二进制数，因而，参加运算数的范围很有限，要想扩大数的范围，必须采用多精度运算。

程序2-2　双精度加法运算



程序2-2图

目的　计算（x）+（y），并将结果存入（w）。

存贮器分配　见程序2-2图。

实验步骤

（1）送入表2-2所列程序，按存贮器分配图送入数据。

（2）单步执行程序，检查地址偏移量，AX和进位标志位CF的内容为：

单步	地址	AX	CF
1	2003	65A0	×
2	2007	1D3E	1
3	200A	1D3E	1
4	200D	0015	1
5	2011	0037	0
6	2014	0037	0

检查（w）＝00371D3EH。

表2-2　双精度加法运算程序

行	地址(Hex)	内容(Hex)	助记符	注　解
010	2000	A10010	MOV AX,1000H	；(x)低位送AX
020	2003	03060410	ADD AX,1004H	；(x)低位加(y)低位
030	2007	A30810	MOV 1008H,AX	；存低位和
040	200A	A10210	MOV AX,1002H	；(x)高位送AX
050	200D	13060610	ADC AX,1006H	；(x)高位加(y)高位
060	2011	A30A10	MOV 100AH,AX	；存高位和
070	2014	F4	HLT	

程序分析

本程序是双精度（2个16位，即32位）运算，利用累加器AX首先求低16位和，并存入低位 w（010—030行指令），然后求高16位和，再存入高　　指令）。由于低位和可能向高位有进位，因而程序中的050行必须用带进位加法指令ADC。如单步2所示，当两个低位字相加有进位时，使进位标志位CF = 1，则在高位字相加中，除两个高位字本身相加

外，还要包含CF的1，如单步5的结果。编制程序时，如不慎将ADC写为ADD，那么，结果必然因高16位少1而导致错误。

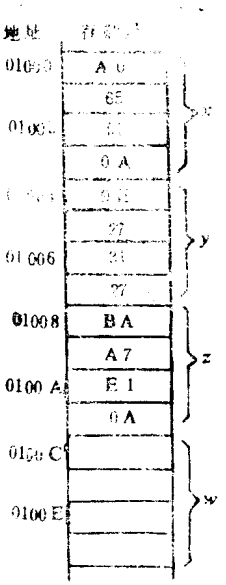
双精度运算能容纳：$2^{32}=2^2\cdot(2^{10})^3\approx4$亿个不同的数，若按整数操作时，一般是足够使用了。

本程序存贮器寻址均采用直接寻址方式，这对于双精度运算是简便的。但对于超过两个字的多精度运算一般要用间接寻址，这在以后的程序中会逐渐遇到。

程序2-3　双精度加减法运算

目的　计算（x）+（y）-（z）+100，并将结果存入（w）。

存贮器分配　见程序2-3图。



程序2-3图

实验步骤

（1）送入表2-3所列程序，按存贮器分配图送入数据。

（2）单步执行程序，检查地址偏移量、DX、AX和进位标志位CF的内容为：

单步	地址	DX	AX	CF
1	2003	××××	65A0	×
2	2007	0A51	65A0	×
3	200B	0A51	8D3E	0
4	200F	3182	8D3E	0
5	2013	3182	E584	1
6	2017	26A0	E584	0
7	201A	26A0	E5E8	0
8	201E	26A0	E5E8	0
9	2021	26A0	E5E8	0
10	2025	26A0	E5E8	0

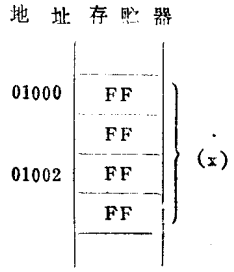
存在（w）=26A0E5E8H。

表2-3　双精度加减法运算程序

行	地址(Hex)	内容(Hex)	助记符	注　解
010	2000	A10010	MOV AX,1000H	；送低16位
020	2003	8B160210	MOV DX,1002H	；送高16位
030	2007	03060410	ADD AX,1004H	；加低16位
040	200B	13160610	ADC DX,1006H	；加高16位
050	200F	2B060810	SUB AX,1008H	；减低16位
060	2013	1B160A10	SBB DX,100AH	；减高16位
070	2017	056400	ADD AX,#100	；加立即数
080	201A	81D20000	ADC DX,#0	
090	201E	A30C10	MOV 100CH,AX	；存低16位
100	2021	89160E10	MOV 100EH,DX	；存高16位
110	2025	F4	HLT	

程序分析

这是一个双精度加减法运算，首先用010、020行指令将被加数（x）的低16位送入AX，高16位送入DX。然后，用AX进行低16位的加减运算，并保存其和或差。用DX进行高16位的加减运算，同样保存其和或差。低位和高位运算用进位标志位CF连接，当低位有进位或借位时，CF = 1，使在高位加1或减1。如单步执行记录的第5步，进行低位相减时有借位，使CF = 1，当第6步进行高位相减时，从DX中多减1（被低位借去1），使高位的结果为26A0H。



程序2-4图

程序2-4　不同精度的带符号加法运算

目的　BX寄存器中单精度带符号数与（x）中的双精度带符号数相加，其和再存入（x）。

存贮器分配　见程序2-4图。

实验步骤

（1）送入表2-4所列程序，按存贮器分配图送入数据，将0002H送入BX寄存器。

表2-4　不同精度带符号加法程序

行	地址(Hex)	内容(Hex)	助记符	注　解
010	2000	8BC3	MOV AX,BX	；取被加数
020	2002	99	CWD	；扩展符号
030	2003	03060010	ADD AX,1000H	；低16位相加
040	2007	A30010	MOV 1000H,AX	；存低16位和
050	200A	13160210	ADC DX,1002H	；高16位相加
060	200E	89160210	MOV 1002H,DX	；存高16位和
070	2012	F4	HLT	

（2）单步执行程序，检查地址偏移量、DX、AX和CF标志位的内容为：

单步	地址	DX	AX	CF
1	2002	××××	0002	×
2	2003	0000	0002	×
3	2007	0000	0001	1
4	200A	0000	0001	1
5	200E	0000	0001	1
6	2012	0000	0001	1

检查（x）= 00000001H。

程序分析

带符号加减运算一般采用补码运算，规定参加运算的操作数和结果均为补码形式。本程序两个不同精度的补码运算，必须首先用扩展符号指令，将短数的符号扩展到与长数位数相等，然后按补码进行运算。扩展符号指令有CBW和CWD指令。CBW指令是将AL中补码数的符号扩展到AH，在AX中得到一个等值一字长的补码数。类似地，CWD指令是将AX中补码数的符号扩展到DX，这样，在DX:AX中形成一个双字补码数。程序中的020行使用CWD指令，将AX中的0002H扩展为00000002H，如单步2记录的数据。由于AX中为正数，符号为0，故扩展后DX为全0。若AX为负数，符号为1，则扩展后DX必为全1。

程序中030、040行指令进行低位补码相加，并将其和存入（x）的低位。050、060行指令进行高位补码相加，其和存入（x）的高位，由于要考虑从低位来的进位，因而高位相加必须使用ADC指令。

本程序带符号数按2的补码进行运算，其中：

（BX）=0002H=2

经符号扩展为00000002H

而　　（x）=FFFFFFFFH=-1

所以

	DX	AX
	0000	0002
	+FFFF	FFFF
CF=	10000	0001

CF=1

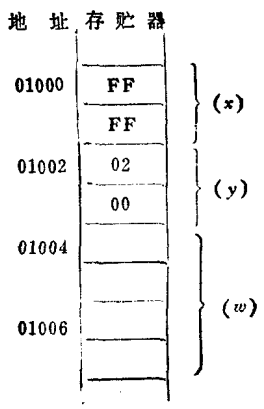
如单步3、5所示，低位和高位相加，均使进位标志位CF=1。最后计算结果（x）=00000001H。

§2-2　二进制乘除运算

程序2-5　单精度乘法运算

目的　两个单精度数相乘（x）*（y），并将乘积存入（w）。

存贮器分配见程序2-5图。



程序2-5图

实验步骤

（1）送入表2-5所列程序，按存贮器分配图送入数据。

（2）执行程序，检查（w）=0001FFFEH。

（3）将020行改为IMUL指令（2004H单元改为2EH），重新执行程序，检查（w）=FFFFFFFEH。

程序分析

乘法指令的操作数如图2-1所示，它可以进行两个8位数相乘，积为16位。也可以进行两个16位数相乘，积为32位。但必须注意，乘法指令有带符号乘法和不带符号乘法两种，使用时不能混淆。

MUL是无符号乘法指令，它按无符号数对待操作数，并产生无符号的积。IMUL是带

表2-5　单精度乘法程序

行	地址(Hex)	内容(Hex)	助记符	注解
010	2000	A10010	MOV AX,1000H	;送(x)
020	2003	F7260210	MUL 1002H	;(x)*(y)
030	2007	A30410	MOV 1004H,AX	;存低位积
040	200A	89160610	MOV 1006H,DX	;存高位积
050	200E	F4	HLT	