IBM-PC

汇编语言程序设计

实验教程

沈美明 温冬婵 张赤红

清华大学出版社

IBM PC 汇编语言程序设计
实 验 教 程

沈美明　温冬婵　张赤红

清华大学出版社

内 容 简 介

本书与清华大学出版社已出版的《IBM PC 汇编语言程序设计》和《IBM PC 汇编语言习题集》组成配套教材。全书共分五章：第一章介绍上机的基本方法；第二章为基本程序结构训练；第三章为I/O设备的编程技术；第四章为磁盘文件存取技术；第五章为高级汇编语言技术和连接技术。全书提供了22个例题及17个实验题。这组实验综合了顺序、循环、分支和子程序四种基本结构的编程技术，又增加了系统功能调用、BIOS调用、宏汇编及条件汇编功能、模块连接技术与中断程序设计技术等内容。又包括了表格查找、声音输出、键盘输入、显示及窗口技术、画线技术以及顺序式、随机式、文件代号式磁盘文件存取技术等汇编语言最经常使用的场合所需要的技术。因此本书既适于作为高等院校《汇编语言程序设计》课的实验指导书，又可供使用汇编语言的工程技术人员参考。

(京)新登字 158 号

JS430/3607

IBM PC 汇编语言程序设计实验教程

沈美明　温冬婵　张赤红

责任编辑　贾仲良

☆

清华大学出版社出版

北京　清华园

北京市昌平环球科技印刷厂印刷

新华书店总店科技发行所发行

☆

开本：787×1092 1/16　印张：12.5　字数：295 千字

1992 年 9 月第 1 版　1992 年 9 月第 1 次印刷

印数：00001—15000

ISBN 7-302-01033-1/TP；378

定价：3.70 元

前　言

本书与清华大学出版社已出版的《IBM PC 汇编语言程序设计》和《IBM PC 汇编语言习题集》一起组成配套教材，主要面向高等院校“汇编语言程序设计”的课程教学。“汇编语言程序设计”是一门实践性很强的课程，只有通过上机实践才有可能掌握程序设计技术并使其达到较高的水平，因此我们编写了这本《实验教程》，一方面为“汇编”课程的实验教学服务；另一方面，需要使用汇编语言的工程技术人员也可以根据本书内容进行上机实践，从中得到收益及提高。

本书共分五章：第一章介绍上机的基本方法，特别强调了 DEBUG 的使用；第二章为基本程序结构（循环、分支和子程序）训练；第三章介绍几种主要输入/输出设备的编程技术；第四章说明磁盘文件存取技术；第五章为以宏为主的高级汇编语言技术及连接技术训练。全书共给出了 22 个例题及 17 个实验题。这组实验的综合性较强。它综合了顺序、循环、分支和子程序四种基本结构的编程技术，同时又增加了系统功能调用、BIOS 调用、宏汇编及条件汇编功能、模块连接技术及中断程序设计技术等内容。它又包括了表格查找、声音输出、键盘输入、显示及窗口技术、画线技术以及顺序式、随机式、文件代号式磁盘文件存取技术等汇编语言最经常使用的场合所需要的技术。因此，这组实验对学生的训练是全面的。而对于工程技术人员则可以各取所需，协助解决你所遇到的实际问题。

在清华大学计算机系的“汇编”课中，安排了 32 机时的上机训练。要求学生完成 10 个属于基本要求的实验题，其余 7 个实验题（带 * 号）供有余力的学生选做。各兄弟院校可以根据自己的机时安排选择部分实验题供学生实验用。

本书的第一、二章及附录由沈美明同志编写，第三章由温冬婵同志编写，第四、五章由张赤红同志编写。书中如有错误和不当之处，欢迎读者批评指正。

目　录

第一章　实验的基本要求与方法

1.1　实验目的与要求

一、实验目的

学习程序设计的基本方法和技能，熟练掌握用汇编语言设计、编写、调试和运行程序的方法。为后续课程打下坚实的基础。

二、实验要求

1. 上机前要作好充分准备，包括程序框图、源程序清单、调试步骤、测试方法、对运行结果的分析等。

2. 上机时要遵守实验室的规章制度，爱护实验设备。要熟悉与实验有关的系统软件（如编辑程序、汇编程序、连接程序和调试程序等）的使用方法。在程序的调试过程中，有意识地学习及掌握 debug 程序的各种操作命令，以便掌握程序的调试方法及技巧。

为了更好地进行上机管理，要求用硬盘存储程序，并建立和使用子目录，以避免文件被别人删除。有关目录的操作命令见附录一 4。此外，为便于统一管理硬盘中的文件，要求实验者按以下形式命名实验文件：

字母学号.扩充名

其中字母取 a～z 的 26 个英文字母，按实验顺序从 a 至 z 排列。如学号为 850431 学生的第二个实验程序所对应的文件名应为 b850431.asm。

3. 程序调试完后，须由实验辅导教师在机器上检查运行结果，经教师认可后的源程序可通过打印机输出，并请教师在程序清单上签字。每个实验完成后，应写出实验报告。实验报告的要求如下：

(1)设计说明：用来说明程序的功能、结构。它包括：程序名、功能、原理及算法说明、程序及数据结构、主要符号名的说明等。

(2)调试说明：便于学生总结经验提高编程及调试能力。它包括：调试情况，如上机时遇到的问题及解决办法，观察到的现象及其分析，对程序设计技巧的总结及分析等；程序的输出结果及对结果的分析；实验的心得体会，以及诸如调试日期、文件存放的软盘号等需要记录的信息。

(3)使用说明：程序提供给用户使用时必须作出的说明。如：程序的使用方法，调用方式，操作步骤等；要求输入信息的类型及格式；出错信息的含义及程序的适用范围等。

(4)程序框图。

(5)经辅导教师签名后的程序清单。

1.2 实验方法

有关汇编语言程序的上机过程请读者参阅清华大学出版社 1991 年出版的《IBM PC 汇编语言程序设计》的 4.4 节。在这里，我们举例简要说明该过程以及程序的调试方法。

例 1.1 比较字符串 sample

试编写一程序：比较两个字符串 string1 和 string2 所含的字符是否相同。若相同则显示'Match'，否则，显示'No match'。

我们可以用串比较指令来完成程序所要求的功能。上机过程如下：

1. 调用字处理程序 wordstar 建立 asm 文件

当然也可以用其它编辑程序如 pced 或行编辑程序 edlin 来建立源文件。

c>ws

使用非文本文件方式（n 命令）建立以 sample.asm 为文件名的源文件如图 1.1 所示。然后用 CTRL K X 命令将文件存入磁盘，并使系统返回 DOS。

```
;PROGRAM TITLE GOES HERE--Compare string
;*********************************************
datarea segment                  ;define data segment
  string1      db      'Move the cursor backward.'
  string2      db      'Move the cursor backward.'
;
  mess1        db      'Match.',13,10,'$'
  mess2        db      'No match!',13,10,'$'
datarea ends
;*********************************************
prognam segment                   ;define code segment
;---------------------------------------------
main    proc      far
        assume cs:prognam,ds:datarea,es:datarea
start:                            ;starting execution address
;set up  stack for return
        push      ds              ;save old data segment
        sub       ax,ax           ;put zero in AX
        push      ax              ;save it on stack
;set DS register to current data segment
        mov       ax,datarea      ;datarea segment addr
        mov       ds,ax           ;  into DS register
        mov       es,ax           ;  into ES register
;MAIN PART OF PROGRAM GOES HERE
        lea       si,string1
        lea       di,string2
        cld
        mov       cx,25
        repz      cmpsb
```

```
        jz      match
        lea     dx,mess2
        jmp     short disp
match:
        lea     dx,mess1
disp:
        mov     ah,09
        int     21h
        ret                         ;return to DOS
main    endp                        ;end of main part of program
;-------------------------------------------------------------------
prognam ends                        ;end of code segment
;*******************************************************************
        end     start               ;end assembly
```

图1.1　例1.1的源文件 sample.asm

2. 用汇编程序 masm(或 asm)对源文件汇编产生目标文件 obj

```
C>masm sample;
The IBM Personal Computer MACRO Assembler
Version 1.00 (C)Copyright IBM Corp 1981

Warning Severe
Errors  Errors
0       0
```

如汇编指示出错则需重新调用编辑程序修改错误，直至汇编通过为止。如调试时需要用 lst 文件，则应在汇编过程中建立该文件。

3. 用连接程序 link 产生执行文件 exe

```
C>link sample;

IBM 5550 Multistation Linker 2.00
(C) Copyright IBM Corp. 1983

Warning: No STACK segment

There was 1 error detected.
```

4. 执行程序

可直接从 DOS 执行程序如下：

```
C>sample
Match.
```

终端上已显示出程序的运行结果。为了调试程序的另一部分，可重新进编辑程序修改两个字符串的内容，使它们互不相同。如修改后的数据区为：

```
;*******************************************************************
datarea segment                     ;define data segment
  string1       db      'Move the cursor backward.'
```

```
  string2       db      'Move the cursor forward.'
;
  mess1         db      'Match.',13,10,'$'
  mess2         db      'No match!',13,10,'$'
datarea ends
;*************************************************
```

然后，重新汇编、连接、执行，结果为：

```
C>sample
No match!
```

至此，程序已调试完毕，运行结果正确。

另一种调试程序的方法是使用 debug 程序。可调用如下：

```
C>debug sample.exe
-
```

此时，debug 已将执行程序装入内存，可直接用 g 命令运行程序。

```
-g
Match.

Program terminated normally
```

为调试程序的另一部分，可在 debug 中修改字符串内容。可先用 u 命令显示程序，以便了解指令地址。显示结果如图1.2所示。

```
-u
19F3:0000  1E            PUSH    DS
19F3:0001  2BC0          SUB     AX,AX
19F3:0003  50            PUSH    AX
19F3:0004  B8EE19        MOV     AX,19EE
19F3:0007  8ED8          MOV     DS,AX
19F3:0009  8EC0          MOV     ES,AX
19F3:000B  8D360000      LEA     SI,[0000]
19F3:000F  8D3E1900      LEA     DI,[0019]
19F3:0013  FC            CLD
19F3:0014  B91900        MOV     CX,0019
19F3:0017  F3            REPZ
19F3:0018  A6            CMPSB
19F3:0019  7406          JZ      0021
19F3:001B  8D163B00      LEA     DX,[003B]
19F3:001F  EB04          JMP     0025
-u
19F3:0021  8D163200      LEA     DX,[0032]
19F3:0025  B409          MOV     AH,09
19F3:0027  CD21          INT     21
19F3:0029  CB            RETF
19F3:002A  FF7501        PUSH    [DI+01]
19F3:002D  40            INC     AX
19F3:002E  5A            POP     DX
19F3:002F  22C2          AND     AL,DL
19F3:0031  50            PUSH    AX
19F3:0032  807EDC20      CMP     BYTE PTR [BP-24],20
19F3:0036  B0FF          MOV     AL,FF
19F3:0038  7201          JB      003B
```

```
19F3:003A  40          INC     AX
19F3:003B  5A          POP     DX
19F3:003C  22C2        AND     AL,DL
19F3:003E  50          PUSH    AX
19F3:003F  80F920      CMP     CL,20
```

图1.2　例1.1用 debug 调试时，u 命令的显示情况

将断点设置在程序的主要部分运行以前。

```
-g0b
AX=19EE  BX=0000  CX=007A  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=000B      NV UP DI PL ZR NA PE NC
19F3:000B  8D360000        LEA        SI,[0000]                  DS:0000=6F4D
```

根据其中指示的 ds 寄存器内容查看数据段的情况如下：

```
-d0
19EE:0000  4E 6F 76 65 20 74 68 65-20 63 75 72 73 6F 72 20   Move the cursor
19EE:0010  62 61 63 6B 77 61 72 64-2E 4D 6F 76 65 20 74 68   backward.Move th
19EE:0020  65 20 63 75 72 73 6F 72-20 62 61 63 6B 77 61 72   e cursor backwar
19EE:0030  64 2E 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61   d.Match...$No ma
19EE:0040  74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 00   tch!..$.........
19EE:0050  1E 2B C0 50 B8 EE 19 8E-D8 8E C0 8D 36 00 00 8D   .+@P8n..X.@.6...
19EE:0060  3E 19 00 FC B9 19 00 F3-A6 74 06 8D 16 3B 00 EB   >..|9..s&t...;.k
19EE:0070  04 8D 16 32 00 B4 09 CD-21 CB FF 75 01 40 5A 22   ...2.4.M!K.u.@Z"
```

可用 e 命令修改数据区的字符串，操作如下：

```
-e29
19EE:0029  62.66    61.6f    63.72    6B.77    77.61    61.72    72.64
19EE:0030  64.2e    2E.20
```

再次用 d 命令查看修改结果。

```
-d0
19EE:0000  4D 6F 76 65 20 74 68 65-20 63 75 72 73 6F 72 20   Move the cursor
19EE:0010  62 61 63 6B 77 61 72 64-2E 4D 6F 76 65 20 74 68   backward.Move th
19EE:0020  65 20 63 75 72 73 6F 72-20 66 6F 72 77 61 72 64   e cursor forward
19EE:0030  2E 20 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61   . Match...$No ma
19EE:0040  74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 00   tch!..$.........
19EE:0050  1E 2B C0 50 B8 EE 19 8E-D8 8E C0 8D 36 00 00 8D   .+@P8n..X.@.6...
19EE:0060  3E 19 00 FC B9 19 00 F3-A6 74 06 8D 16 3B 00 EB   >..|9..s&t...;.k
19EE:0070  04 8D 16 32 00 B4 09 CD-21 CB FF 75 01 40 5A 22   ...2.4.M!K.u.@Z"
```

用 g 命令运行程序，结果为：

```
-g
No match!

Program terminated normally
```

用 q 命令退出 debug。

```
-q
```

至此，程序已调试完毕。为了进一步说明 debug 命令的使用方法，我们再次重复上述程序的调试过程，只是使用 e、a 和 f 命令来修改数据区的内容而已。必须注意，由于在用 debug 调试程序时，只能修改当时有关的内存单元内容，因此重新用 debug 装入执行程序时，仍

是原来在磁盘中的文件内容。操作如下：

```
C>debug sample.exe
-g0b

AX=19EE  BX=0000  CX=007A  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=000B   NV UP DI PL ZR NA PE NC
19F3:000B 8D360000        LEA       SI,[0000]                    DS:0000=6F4D

-d0
19EE:0000  4D 6F 76 65 20 74 68  65-20  63 75 72 73 6F 72 20   Move the cursor
19EE:0010  62 61 63 6B 77 61 72  64-2E  4D 6F 76 65 20 74 68   backward.Move th
19EE:0020  65 20 63 75 72 73 6F  72-20  62 61 63 6B 77 61 72   e cursor backwar
19EE:0030  64 2E 4D 61 74 63 68  2E-0D  0A 24 4E 6F 20 6D 61   d.Match...$No ma
19EE:0040  74 63 68 21 0D 0A 24  00-00  00 00 00 00 00 00 00   tch!..$.........
19EE:0050  1E 2B C0 50 B8 EE 19  8E-D8  8E C0 8D 36 00 00 8D   .+@P8n..X.@.6...
19EE:0060  3E 19 00 FC B9 19 00  F3-A6  74 06 8D 16 3B 00 EB   >..|9..s&t...;.k
19EE:0070  04 8D 16 32 00 B4 09  CD-21  CB FF 75 01 40 5A 22   ...2.4.M!K.u.@Z"

-e29      'forward.'20

-d0
19EE:0000  4D 6F 76 65 20 74 68  65-20  63 75 72 73 6F 72 20   Move the cursor
19EE:0010  62 61 63 6B 77 61 72  64-2E  4D 6F 76 65 20 74 68   backward.Move th
19EE:0020  65 20 63 75 72 73 6F  72-20  66 6F 72 77 61 72 64   e cursor forward
19EE:0030  2E 20 4D 61 74 63 68  2E-0D  0A 24 4E 6F 20 6D 61   . Match...$No ma
19EE:0040  74 63 68 21 0D 0A 24  00-00  00 00 00 00 00 00 00   tch!..$.........
19EE:0050  1E 2B C0 50 B8 EE 19  8E-D8  8E C0 8D 36 00 00 8D   .+@P8n..X.@.6...
19EE:0060  3E 19 00 FC B9 19 00  F3-A6  74 06 8D 16 3B 00 EB   >..|9..s&t...;.k
19EE:0070  04 8D 16 32 00 B4 09  CD-21  CB CC CC CC CC CC CC   ...2.4.M!KLLLLLL

-g
No match!

Program terminated normally
```

可见这种 e 命令的方式避免使用 ASCII 码进入，对用户是比较方便的。其中最后一个20是空格键的 ASCII 码，以补足原来的字节数。

也可使用 a 命令把数据区的内容恢复原状，具体如下：

```
-a19ee:29
19EE:0029              db          'backward.'
19EE:0032

-d0
19EE:0000  4D 6F 76 65 20 74 68  65-20  63 75 72 73 6F 72 20   Move the cursor
19EE:0010  62 61 63 6B 77 61 72  64-2E  4D 6F 76 65 20 74 68   backward.Move th
19EE:0020  65 20 63 75 72 73 6F  72-20  62 61 63 6B 77 61 72   e cursor backwar
19EE:0030  64 2E 4D 61 74 63 68  2E-0D  0A 24 4E 6F 20 6D 61   d.Match...$No ma
19EE:0040  74 63 68 21 0D 0A 24  00-00  00 00 00 00 00 00 00   tch!..$.........
19EE:0050  1E 2B C0 50 B8 EE 19  8E-D8  8E C0 8D 36 00 00 8D   .+@P8n..X.@.6...
19EE:0060  3E 19 00 FC B9 19 00  F3-A6  74 06 8D 16 3B 00 EB   >..|9..s&t...;.k
19EE:0070  04 8D 16 32 00 B4 09  CD-21  CB 6F 72 77 61 72 64   ...2.4.M!Korward

-g
Match.
AX=0924  BX=0000  CX=0000  DX=0032  SP=FFFC  BP=FFFF  SI=0019  DI=0032
```

```
DS=19EE  ES=19EE  SS=19EE  CS=19EE  IP=0096     NV UP DI NG NZ AC PE NC
19EE:0096 CC                INT       3
-q
```

由于 a 命令是汇编命令，因此信息是用汇编格式进入的。如果修改的是程序中的语句，方法也是相同的，下面我们还会看到这类的操作。现在再看一下用 f 命令修改数据区的方法。

```
C>debug sample.exe
-g0b

AX=19EE  BX=0000  CX=007A  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=000B     NV UP DI PL ZR NA PE NC
19F3:000B 8D360000        LEA       SI,[0000]                    DS:0000=6F4D

-d0
19EE:0000  4D 6F 76 65 20 74 68 65-20 63 75 72 73 6F 72 20   Move the cursor
19EE:0010  62 61 63 6B 77 61 72 64-2E 4D 6F 76 65 20 74 68   backward.Move th
19EE:0020  65 20 63 75 72 73 6F 72-20 62 61 63 6B 77 61 72   e cursor backwar
19EE:0030  64 2E 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61   d.Match...$No ma
19EE:0040  74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 00   tch!..$.........
19EE:0050  1E 2B C0 50 B8 EE 19 8E-D8 8E C0 8D 36 00 00 8D   .+@P8n..X.@.6...
19EE:0060  3E 19 00 FC B9 19 00 F3-A6 74 06 8D 16 3B 00 EB   >..|9..s&t...;.k
19EE:0070  04 8D 16 32 00 B4 09 CD-21 CB FF 75 01 40 5A 22   ...2.4.M!K.u.@Z"

-f29 1 9  'forward.'20

-d0
19EE:0000  4D 6F 76 65 20 74 68 65-20 63 75 72 73 6F 72 20   Move the cursor
19EE:0010  62 61 63 6B 77 61 72 64-2E 4D 6F 76 65 20 74 68   backward.Move th
19EE:0020  65 20 63 75 72 73 6F 72-20 66 6F 72 77 61 72 64   e cursor forward
19EE:0030  2E 20 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61   . Match...$No ma
19EE:0040  74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 00   tch!..$.........
19EE:0050  1E 2B C0 50 B8 EE 19 8E-D8 8E C0 8D 36 00 00 8D   .+@P8n..X.@.6...
19EE:0060  3E 19 00 FC B9 19 00 F3-A6 74 06 8D 16 3B 00 EB   >..|9..s&t...;.k
19EE:0070  04 8D 16 32 00 B4 09 CD-21 CB FF 75 01 40 5A 22   ...2.4.M!K.u.@Z"

-g
No match!

Program terminated normally
-q
```

f 命令中的29为所修改区的首地址，l 9表示需要修改的长度为9个字节。

为进一步说明程序的调试过程，现假设程序编制错误：在源文件中把 jz match 改为 jnz match。该程序经汇编、连接后，进入 debug 调试如下：

```
C>debug sample.exe
-g
No match!

Program terminated normally
```

结果是错误的(因源文件中两个字符串是相同的)。为检查程序的错误，将断点设在比较串之后。

```
-g19
```

```
AX=19EE  BX=0000  CX=0000  DX=0000  SP=FFFC  BP=0000  SI=0019  DI=0032
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=0019     NV UP DI PL ZR NA PE NC
19F3:0019 7506              JNZ         0021
```

此时零标志为ZR，即ZF＝1，即表示比较结果相等，说明比较结果是正确的。现在可用p命令再执行一条指令以观察指令的转向。

```
-p

AX=19EE  BX=0000  CX=0000  DX=0000  SP=FFFC  BP=0000  SI=0019  DI=0032
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=001B     NV UP DI PL ZR NA PE NC
19F3:001B 8D163B00          LEA         DX,[003B]                DS:003B=6F4E
```

为查到003B单元的内容，可查数据区如下：

```
-d0
19EE:0000  4D 6F 76 65 20 74 68 65-20 63 75 72 73 6F 72 20   Move the cursor
19EE:0010  62 61 63 6B 77 61 72 64-2E 4D 6F 76 65 20 74 68   backward.Move th
19EE:0020  65 20 63 75 72 73 6F 72-20 62 61 63 6B 77 61 72   e cursor backvar
19EE:0030  64 2E 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61   d.Match...$No ma
19EE:0040  74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 00   tch!..$.........
19EE:0050  1E 2B C0 50 B8 EE 19 8E-D8 8E C0 8D 36 00 00 8D   .+@P8n..X.@.6...
19EE:0060  3E 19 00 FC B9 19 00 F3-A6 75 06 8D 16 3B 00 EB   >..|9..s&u...;.k
19EE:0070  04 8D 16 32 00 B4 09 CD-21 CB FF 75 01 40 5A 22   ...2.4.M!K.u.@Z"
```

可见003B单元的内容为4E，即N的ASCII码，后面跟的是No match!，这说明jnz指令使用错误，应该改为jz match。可用a命令修改，并用u命令检查修改结果。运行结果说明程序修改正确。

```
-a19
19F3:0019          jz          0021
19F3:001B
-u0
19F3:0000  1E              PUSH    DS
19F3:0001  2BC0            SUB     AX,AX
19F3:0003  50              PUSH    AX
19F3:0004  B8EE19          MOV     AX,19EE
19F3:0007  8ED8            MOV     DS,AX
19F3:0009  8EC0            MOV     ES,AX
19F3:000B  8D360000        LEA     SI,[0000]
19F3:000F  8D3E1900        LEA     DI,[0019]
19F3:0013  FC              CLD
19F3:0014  B91900          MOV     CX,0019
19F3:0017  F3              REPZ
19F3:0018  A6              CMPSB
19F3:0019  7406            JZ      0021
19F3:001B  8D163B00        LEA     DX,[003B]
19F3:001F  EB04            JMP     0025
-rip
IP 001B
:0
-g
Match.
```

```
AX=0924  BX=0000  CX=0000  DX=0032  SP=FFFC  BP=FFFF  SI=0019  DI=0032
DS=19EE  ES=19EE  SS=19EE  CS=19EE  IP=0096   NV UP DI NG NZ AC PE NC
19EE:0096 CC              INT       3
-q
```

在这里应该注意，在使用 a 命令修改数据区时，必须给出数据段的地址，而在修改程序区时，由于 a 命令的缺省段为代码段，所以直接给出偏移地址就可以了。

在调试过程中，也可以用 t 命令逐条跟踪程序的执行。下面列出断点停在0b 后，用 f 命令修改数据区中字符串的内容，然后用 t 命令逐条执行指令的情况。

```
-f29 1 9 'forward.'20

-d0
19EE:0000  4D 6F 76 65 20 74 68 65-20 63 75 72 73 6F 72 20   Move the cursor
19EE:0010  62 61 63 6B 77 61 72 64-2E 4D 6F 76 65 20 74 68   backward.Move th
19EE:0020  65 20 63 75 72 73 6F 72-20 66 6F 72 77 61 72 64   e cursor forward
19EE:0030  2E 20 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61   . Match...$No ma
19EE:0040  74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 00   tch!..$.........
19EE:0050  1E 2B C0 50 B8 EE 19 8E-D8 8E C0 8D 36 00 00 8D   .+@P8n..X.@.6...
19EE:0060  3E 19 00 FC B9 19 00 F3-A6 74 06 8D 16 3B 00 EB   >..|9..s&t...;.k
19EE:0070  04 8D 16 32 00 B4 09 CD-21 CB FF 75 01 40 5A 22   ...2.4.M!K.u.@Z"

-t

AX=19EE  BX=0000  CX=007A  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=000F   NV UP DI PL ZR NA PE NC
19F3:000F 8D3E1900        LEA       DI,[0019]                DS:0019=6F4D
-t

AX=19EE  BX=0000  CX=007A  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0019
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=0013   NV UP DI PL ZR NA PE NC
19F3:0013 FC              CLD
-t

AX=19EE  BX=0000  CX=007A  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0019
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=0014   NV UP DI PL ZR NA PE NC
19F3:0014 B91900          MOV       CX,0019
-t

AX=19EE  BX=0000  CX=0019  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0019
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=0017   NV UP DI PL ZR NA PE NC
19F3:0017 F3              REPZ
19F3:0018 A6              CMPSB
-t

AX=19EE  BX=0000  CX=0018  DX=0000  SP=FFFC  BP=0000  SI=0001  DI=001A
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=0017   NV UP DI PL ZR NA PE NC
19F3:0017 F3              REPZ
19F3:0018 A6              CMPSB
-t

AX=19EE  BX=0000  CX=0017  DX=0000  SP=FFFC  BP=0000  SI=0002  DI=001B
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=0017   NV UP DI PL ZR NA PE NC
19F3:0017 F3              REPZ
19F3:0018 A6              CMPSB
-t

AX=19EE  BX=0000  CX=0016  DX=0000  SP=FFFC  BP=0000  SI=0003  DI=001C
```

```
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=0017     NV UP DI PL ZR NA PE NC
19F3:0017 F3                REPZ
19F3:0018 A6                CMPSB
-t

AX=19EE  BX=0000  CX=0015  DX=0000  SP=FFFC  BP=0000  SI=0004  DI=001D
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=0017     NV UP DI PL ZR NA PE NC
19F3:0017 F3                REPZ
19F3:0018 A6                CMPSB
-t

AX=19EE  BX=0000  CX=0014  DX=0000  SP=FFFC  BP=0000  SI=0005  DI=001E
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=0017     NV UP DI PL ZR NA PE NC
19F3:0017 F3                REPZ
19F3:0018 A6                CMPSB
-t

AX=19EE  BX=0000  CX=0013  DX=0000  SP=FFFC  BP=0000  SI=0006  DI=001F
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=0017     NV UP DI PL ZR NA PE NC
19F3:0017 F3                REPZ
19F3:0018 A6                CMPSB
-t

AX=19EE  BX=0000  CX=0012  DX=0000  SP=FFFC  BP=0000  SI=0007  DI=0020
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=0017     NV UP DI PL ZR NA PE NC
19F3:0017 F3                REPZ
19F3:0018 A6                CMPSB
-t

AX=19EE  BX=0000  CX=0011  DX=0000  SP=FFFC  BP=0000  SI=0008  DI=0021
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=0017     NV UP DI PL ZR NA PE NC
19F3:0017 F3                REPZ
19F3:0018 A6                CMPSB
-t

AX=19EE  BX=0000  CX=0010  DX=0000  SP=FFFC  BP=0000  SI=0009  DI=0022
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=0017     NV UP DI PL ZR NA PE NC
19F3:0017 F3                REPZ
19F3:0018 A6                CMPSB
-t

AX=19EE  BX=0000  CX=000F  DX=0000  SP=FFFC  BP=0000  SI=000A  DI=0023
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=0017     NV UP DI PL ZR NA PE NC
19F3:0017 F3                REPZ
19F3:0018 A6                CMPSB
-t

AX=19EE  BX=0000  CX=000E  DX=0000  SP=FFFC  BP=0000  SI=000B  DI=0024
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=0017     NV UP DI PL ZR NA PE NC
19F3:0017 F3                REPZ
19F3:0018 A6                CMPSB
-t

AX=19EE  BX=0000  CX=000D  DX=0000  SP=FFFC  BP=0000  SI=000C  DI=0025
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=0017     NV UP DI PL ZR NA PE NC
19F3:0017 F3                REPZ
19F3:0018 A6                CMPSB
-t
```