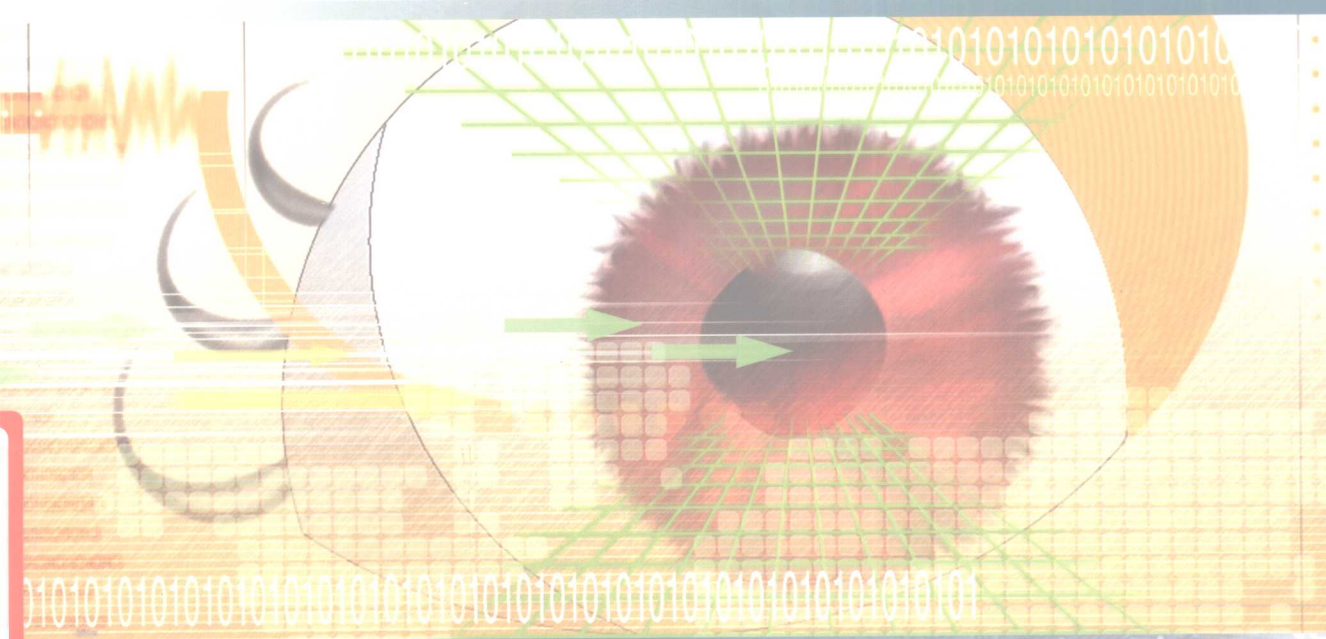


8086/8088和基于ARM核 汇编语言程序设计 实验教程及习题解答 (第二版)

主 编 李敬兆
副主编 纪 平 谢晓东
丁 刚 詹 林



中国科学技术大学出版社

8086/8088 和基于 ARM 核汇编 语言程序设计实验教程及习题解答

(第二版)

主 编 李敬兆
副主编 纪 平 谢晓东
丁 刚 詹 林

中国科学技术大学出版社

2008·合肥

内 容 简 介

汇编语言是提供给用户直接访问计算机系统最快而又最有效的一种编程语言,使用汇编语言编写程序能够充分发挥计算机硬件系统的功能,那些需要对计算机硬件进行控制或对运行时间和效率有要求的系统软件或应用软件,通常都是用汇编语言编写而成的,因此熟练掌握汇编语言程序设计方法是非常重要的。

本书是《8086/8088 和 ARM 核汇编语言程序设计》教材的配套实训教程及习题解答,共设 11 章 20 个实验。目的是使学生通过实验加深对理论课程的理解,增强学生的实际动手能力和实践能力。

图书在版编目(CIP)数据

8086/8088 和 ARM 核汇编语言程序设计实验教程及习题解答/李敬兆主编. —2 版. —合肥:中国科学技术大学出版社,2008. 12

ISBN 978-7-312-02384-2

I. 8… II. 李… III. 汇编语言-程序设计-高等学校-教学参考资料 IV. TP313

中国版本图书馆 CIP 数据核字(2008)第 127691 号

出版发行 中国科学技术大学出版社

地址:安徽省合肥市金寨路 96 号,邮政编码:230026

网址:<http://www.press.ustc.edu.cn>

电话:发行部:0551-3602905 邮购部:0551-3602906

印 刷 合肥现代印务有限公司

经 销 全国新华书店

开 本 787mm×1092mm 1/16

印 张 10.75

字 数 270 千

版 次 2006 年 9 月第 1 版 2008 年 12 月第 2 版

印 次 2008 年 12 月第 2 次印刷

印 数 4001—8000 册

定 价 17.00 元



前 言

汇编语言是提供给用户直接访问计算机系统最快而又最为有效的一种编程语言,使用汇编语言编写程序能够充分发挥计算机硬件系统的功能,具有占用存储空间少、运行速度快以及代码质量高等优点,那些需要对计算机硬件进行控制或对运行时间和效率有要求的系统软件或应用软件,通常都是用汇编语言编写的,因此熟练掌握汇编语言源程序的设计方法是非常重要的。

目前,基于 80X86 系列微处理器的 PC 机和基于 ARM 核微处理器的嵌入式系统应用最为广泛,本书是与《8086/8088 和 ARM 核汇编语言程序设计》教材配套的实训教程和习题解答,目的是使学生通过做实验和做习题加深对理论课程的理解。本书在整个编写过程中,努力做到实验简单、明了,习题解答简捷、完整,具有一定的代表性,文字解释清晰,通俗易懂。

全书实验部分共 11 章 20 个实验,其中第 1 章至第 7 章是基于 8086/8088 汇编语言的实验,共有 16 个实验。第 1 章介绍了 8086/8088 汇编语言运行环境和方法,了解如何使用汇编语言编制程序,熟悉 DEBUG 有关命令的使用,掌握 DEBUG 有关指令的功能,利用 DEBUG 运行简单的程序段。第 2 章介绍了不同数码转换编程及程序调试。第 3 章介绍了汇编语言存储器操作的程序设计实验。第 4 章介绍了汇编语言程序设计基本结构实验,即顺序程序实验、分支程序实验。对每类实验都进行了详细的问题分析,实验验证。第 5 章介绍了数据运算类如双精度加法、BCD 减法、乘法等程序设计实验。第 6 章介绍了 I/O 程序实验,对键盘扫描实验、绘图实验进行了详细论述。第 7 章介绍了汇编语言与 C/C++ 语言的混合程序设计方式,重点叙述了 C/C++ 嵌入汇编程序实验和 C/C++ 调入汇编程序模块实验。第 8 章至第 11 章是基于 ARM 核微处理器汇编语言的实验,共有 4 个实验,分别对 ARM ADS 集成开发环境、ADS 1.2 应用实例、ARM 汇编语言程序设计和 Thumb 汇编语言程序设计进行了介绍。

本书的第 1 章、第 6 章的实验二、第 7 章至第 11 章由安徽理工大学徐辉老师、江静老师、李敬兆老师编写。其余各章实验均由承担相应章节教材的安徽工业大学张雷老师、纪平老师,安徽建筑工业学院丁刚老师、夏巍老师,安徽工程科技学院谢晓东老师编写。全书习题解答部分由承担《8086/8088 和 ARM 核汇编语言程序设计》相应章节的老师编写。安徽理工大学李敬兆教授任主编。

由于编者水平所限,书中难免存在错误和不妥之处,敬请广大读者批评指正。

编 者

2008.7

目 录

第一部分 8086 和基于 ARM 核汇编语言程序设计实验教程

第 1 章 汇编语言程序调试方法	(3)
实验一 汇编语言运行环境及方法、简单程序设计	(3)
第 2 章 数码转换程序设计	(17)
实验一 数码转换编程及程序调试	(17)
实验二 二进制到 BCD 转换	(22)
实验三 二进制到 ASC II 码转换	(24)
第 3 章 存储器操作程序设计	(27)
实验一 存储器块清零	(27)
实验二 内存块移动	(28)
第 4 章 基本程序结构练习	(31)
实验一 循环程序实验	(31)
实验二 分支程序实验	(32)
第 5 章 数据运算程序设计	(35)
实验一 二进制双精度加法运算	(35)
实验二 十进制数的 BCD 码减法运算	(36)
实验三 乘法运算	(38)
第 6 章 I/O 程序设计	(41)
实验一 绘制三角形图形程序	(41)
实验二 键盘输入程序	(44)
第 7 章 汇编语言与 C/C++ 的混合编程	(52)
实验一 在 Turbo C 2.0 中求两个整数的最大值	(53)
实验二 键盘与显示	(55)
实验三 在 Visual C++ 6.0 中用两种方法实现求两个整数中的最大值	(56)
实验四 嵌入汇编求字符串长度	(58)
第 8 章 ARM ADS 集成开发环境介绍	(60)
第 9 章 ADS 1.2 应用实例	(66)



实验一 运算程序设计	(66)
第 10 章 ARM 汇编语言程序设计	(70)
实验一 ARM 汇编语言程序实验	(70)
第 11 章 Thumb 汇编语言程序设计	(76)
实验一 Thumb 汇编语言程序实验	(76)

第二部分 8086 和基于 ARM 核汇编语言程序设计习题解答

第 1 章 汇编语言基础知识习题解答	(83)
第 2 章 IBM-PC 系统结构习题解答	(85)
第 3 章 8086/8088 寻址方式和指令系统习题解答	(88)
第 4 章 8086/8088 汇编语言(格式)习题解答	(93)
第 5 章 基本程序设计习题解答	(97)
第 6 章 输入/输出程序设计习题解答	(100)
第 7 章 8086/8088 循环程序设计习题解答	(103)
第 8 章 8086/8088 分支程序设计习题解答	(113)
第 9 章 子程序设计习题解答	(120)
第 10 章 数据运算程序设计习题解答	(128)
第 11 章 高级汇编语言技术习题解答	(135)
第 12 章 BIOS 和 DOS 中断调用习题解答	(138)
第 13 章 汇编语言与 C/C++ 的混合编程习题解答	(144)
第 14 章 80286/386/486/Pentium 汇编语言简介习题解答	(149)
第 15 章 ARM CPU 概述习题解答	(152)
第 16 章 ARM 系统结构习题解答	(153)
第 17 章 ARM/Thumb 汇编语言(格式)习题解答	(154)
第 18 章 ARM 指令系统习题解答	(156)
第 19 章 Thumb 指令系统习题解答	(158)
第 20 章 ARM 汇编语言设计习题解答	(159)

第一部分 8086 和基于 ARM 核 汇编语言程序设计实验教程

第1章 汇编语言程序调试方法

实验一 汇编语言运行环境及方法、简单程序设计

一、实验目的

1. 熟悉汇编语言运行环境和方法。
2. 了解如何使用汇编语言编制程序。
3. 熟悉 DEBUG 有关命令的使用方法。
4. 利用 DEBUG 掌握有关指令的功能。
5. 利用 DEBUG 运行简单的程序段。

二、实验内容

1. 学会输入、编辑汇编语言程序。
2. 学会对汇编语言程序进行汇编、连接和运行。
3. 进入和退出 DEBUG 程序。
4. 学会 DEBUG 中的 D 命令、E 命令、R 命令、T 命令、A 命令、G 命令等的使用。对于 U 命令、N 命令、W 命令等,也应试一下。

三、实验准备

1. 仔细阅读有关汇编语言环境的内容,事先准备好使用的例子。
2. 准备好源程序清单、设计好调试步骤、测试方法、对运行结果的分析。
3. 编写一个程序:比较 2 个字符串所含的字符是否相同。若相同则显示'Match.', 否则显示'No match!'。
4. 仔细阅读有关 DEBUG 命令的内容,对有关命令,都要事先准备好使用的例子。

四、实验步骤

1. 在 DOS 提示符下,进入 MASM 目录。
2. 在 MASM 目录下启动 EDIT 编辑程序,输入源程序,并对其进行汇编、连接和运行。
 - (1) 调用 edit 输入、编辑源程序并保存在指定的目录中。例:
edit abc.asm
 - (2) 用汇编程序 masm 对源程序汇编产生目标文件 obj。例:
masm abc



不断修改错误,直至汇编通过为止。

(3) 用连接程序 link 产生执行文件 exe。例:link abc

(4) 执行程序

可直接从 DOS 执行程序,即在 DOS 环境中,输入文件名即可。

3. 详细记录每一步所用的命令,以及查看结果的方法和具体结果。

五、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《IBMPC 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

当然也可以用其他编辑程序如 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下所示。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
```

```
;*****
```

```
dataarea segment      ;define data segment
string1  db           'Move the cursor backward.'
string2  db           'Move the cursor backward.'
```

```
;
```

```
mess1    db           'Match.',13,10,'$'
mess2    db           'No match!',13,10,'$'
```

```
dataarea ends
```

```
;*****
```

```
prognam segment      ;define code segment
```

```
;
```

```
main      proc      far
```

```
assume cs:prognam,ds:dataarea,es:dataarea
```

```
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,dataarea ;dataarea 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
```

```
;
```

```
progam ends ;end of code segment
```

```
;*****
```

```
end start ;end assembly
```

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;

Microsoft (R) Macro Assembler Version 5.00

Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

51520 + 423584 Bytes symbol space free

0 Warning Errors

0 Severe Errors

3. 用连接程序 link 产生执行文件 exe

C:\>link sample;



Microsoft (R) Overlay Linker Version 3.60

Copyright (C) Microsoft Corp 1983—1987. All rights reserved.

LINK : warning L4021: no stack segment

4. 执行程序

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:

```
; *****
dataarea segment      ;define data segment
    string1    db      'Move the cursor backward.'
    string2    db      'Move the cursor backward.'
;
    mess1      db      'Match.',13,10,'$'
    mess2      db      'No match!',13,10,'$'
dataarea 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。显示结果如下所示。

—u

0D36:0000 1E	PUSH	DS
0D36:0001 2BC0	SUB	AX,AX
0D36:0003 50	PUSH	AX
0D36:0004 B8310D	MOV	AX,0D31
0D36:0007 8ED8	MOV	DS,AX
0D36:0009 8EC0	MOV	ES,AX

```

0D36:000B 8D360000    LEA    SI,[0000]
0D36:000F 8D3E1900    LEA    DI,[0019]
0D36:0013 FC          CLD
0D36:0014 B91900      MOV    CX,0019
0D36:0017 F3          REPZ
0D36:0018 A6          CMPSB
0D36:0019 7406        JZ      0021
0D36:001B 8D163B00    LEA    DX,[003B]
0D36:001F EB04        JMP     0025
-u
0D36:0021 8D163200    LEA    DX,[0032]
0D36:0025 B409        MOV    AH,09
0D36:0027 CD21        INT     21
0D36:0029 CB          RETF
0D36:002A 8A4608      MOV    AL,[BP+08]
0D36:002D 98          CBW
0D36:002E 50          PUSH   AX
0D36:002F 8B4604      MOV    AX,[BP+04]
0D36:0032 03C6        ADD    AX,SI
0D36:0034 50          PUSH   AX
0D36:0035 E858FF      CALL   FF90
0D36:0038 83C406      ADD    SP,+06
0D36:003B 8BF8        MOV    DI,AX
0D36:003D 83FFFF      CMP    DI,-01
0D36:0040 750C        JNZ    004E

```

将端点设置在程序的主要部分以前。

-g0b

```

AX=0D31 BX=0000 CX=007A DX=0000 SP=FFFC BP=0000 SI=0000 DI=0000
DS=0D31 ES=0D31 SS=0D31 CS=0D36 IP=000B NV UP EI PL ZR NA PE NC
0D36:000B 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

```

0D31:0000  4D 6F 76 65 20 74 68 65-20 63 75 72 73 6F 72 20    Move the cursor
0D31:0010  62 61 63 6B 77 61 72 64-2E 4D 6F 76 65 20 74 68    backward. Move th
0D31:0020  65 20 63 75 72 73 6F 72-20 62 61 63 6B 77 61 72    e cursor backwar

```



```

0D31:0030  64 2E 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61  d. Match... $ No ma
0D31:0040  74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 00  tch!.. $. ....
0D31:0050  1E 2B C0 50 B8 31 0D 8E-D8 8E C0 8D 36 00 00 8D  .+. P. 1. .... 6...
0D31:0060  3E 19 00 FC B9 19 00 F3-A6 74 06 8D 16 3B 00 EB  >.....t...;...
0D31:0070  04 8D 16 32 00 B4 09 CD-21 CB 8A 46 08 98 50 8B  ...2....!..F..P.

```

可用 e 命令修改数据区的字符串,操作如下:

—e29

```

0D31:0029  62.66  61.6f  63.72  6B.77  77.61  61.72  72.64
0D31:0030  64.2e  2E.20

```

再次用 d 命令查看修改结果。

—d0

```

0D31:0000  4D 6F 76 65 20 74 68 65-20 63 75 72 73 6F 72 20  Move the cursor
0D31:0010  62 61 63 6B 77 61 72 64-2E 4D 6F 76 65 20 74 68  backward. Move th
0D31:0020  65 20 63 75 72 73 6F 72-20 66 6F 72 77 61 72 64  e cursor forward
0D31:0030  2E 20 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61  . Match... $ No ma
0D31:0040  74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 00  tch!.. $. ....
0D31:0050  1E 2B C0 50 B8 31 0D 8E-D8 8E C0 8D 36 00 00 8D  .+. P. 1. .... 6...
0D31:0060  3E 19 00 FC B9 19 00 F3-A6 74 06 8D 16 3B 00 EB  >.....t...;...
0D31:0070  04 8D 16 32 00 B4 09 CD-21 CB 8A 46 08 98 50 8B  ...2....!..F..P.

```

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 装入程序时,仍是原来在磁盘中的内容。操作如下:

C:\>debug sample.exe

—g0b

```

AX=0D31  BX=0000  CX=007A  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=0D31  ES=0D31  SS=0D31  CS=0D36  IP=000B  NV UP EI PL ZR NA PE NC
0D36:000B 8D360000  LEA  SI,[0000]  DS:0000=6F4D

```

—d0

```

0D31:0000 4D 6F 76 65 20 74 68 65-20 63 75 72 73 6F 72 20 Move the cursor
0D31:0010 62 61 63 6B 77 61 72 64-2E 4D 6F 76 65 20 74 68 backward. Move th
0D31:0020 65 20 63 75 72 73 6F 72-20 62 61 63 6B 77 61 72 e cursor backwar
0D31:0030 64 2E 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61 d. Match... $ No ma
0D31:0040 74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 00 tch!.. $.
0D31:0050 1E 2B C0 50 B8 31 0D 8E-D8 8E C0 8D 36 00 00 8D .+.P.1.....6...
0D31:0060 3E 19 00 FC B9 19 00 F3-A6 74 06 8D 16 3B 00 EB >.....t...;...
0D31:0070 04 8D 16 32 00 B4 09 CD-21 CB 46 FA 8B 5E FC C1 ...2....!.F..^..
-e29 'forward.'20
-d0
0D31:0000 4D 6F 76 65 20 74 68 65-20 63 75 72 73 6F 72 20 Move the cursor
0D31:0010 62 61 63 6B 77 61 72 64-2E 4D 6F 76 65 20 74 68 backward. Move th
0D31:0020 65 20 63 75 72 73 6F 72-20 66 6F 72 77 61 72 64 e cursor forward
0D31:0030 2E 20 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61 . Match... $ No ma
0D31:0040 74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 00 tch!.. $.
0D31:0050 1E 2B C0 50 B8 31 0D 8E-D8 8E C0 8D 36 00 00 8D .+.P.1.....6...
0D31:0060 3E 19 00 FC B9 19 00 F3-A6 74 06 8D 16 3B 00 EB >.....t...;...
0D31:0070 04 8D 16 32 00 B4 09 CD-21 CB 46 FA 8B 5E FC C1 ...2....!.F..^..
-g
No match!

```

Program terminated normally

可见这种 e 命令的方式避免使用 ASC II 码进入,对用户是比较方便的。其中最后一个 20 是空格键的 ASC II 码,以补足原来的内容恢复原状,具体如下:

```

-a0d31:29
0D31:0029 db 'backward.'
0D31:0032
-d0
0D31:0000 4D 6F 76 65 20 74 68 65-20 63 75 72 73 6F 72 20 Move the cursor
0D31:0010 62 61 63 6B 77 61 72 64-2E 4D 6F 76 65 20 74 68 backward. Move th
0D31:0020 65 20 63 75 72 73 6F 72-20 62 61 63 6B 77 61 72 e cursor backwar
0D31:0030 64 2E 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61 d. Match... $ No ma
0D31:0040 74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 00 tch!.. $.
0D31:0050 1E 2B C0 50 B8 31 0D 8E-D8 8E C0 8D 36 00 00 8D .+.P.1.....6...
0D31:0060 3E 19 00 FC B9 19 00 F3-A6 74 06 8D 16 3B 00 EB >.....t...;...
0D31:0070 04 8D 16 32 00 B4 09 CD-21 CB 56 FA 89 7E EE 8B ...2....!.V..~..
-g

```



Match.

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=0D31 BX=0000 CX=007A DX=0000 SP=FFFC BP=0000 SI=0000 DI=0000
DS=0D31 ES=0D31 SS=0D31 CS=0D36 IP=000B NV UP EI PL ZR NA PE NC
0D36:000B 8D360000 LEA SI,[0000] DS:0000=6F4D

—d0

0D31:0000	4D 6F 76 65 20 74 68 65—20 63 75 72 73 6F 72 20	Move the cursor
0D31:0010	62 61 63 6B 77 61 72 64—2E 4D 6F 76 65 20 74 68	backward. Move th
0D31:0020	65 20 63 75 72 73 6F 72—20 62 61 63 6B 77 61 72	e cursor backwar
0D31:0030	64 2E 4D 61 74 63 68 2E—0D 0A 24 4E 6F 20 6D 61	d. Match... \$ No ma
0D31:0040	74 63 68 21 0D 0A 24 00—00 00 00 00 00 00 00 00	tch!... \$.
0D31:0050	1E 2B C0 50 B8 31 0D 8E—D8 8E C0 8D 36 00 00 8D	. +. P. 1. 6...
0D31:0060	3E 19 00 FC B9 19 00 F3—A6 74 06 8D 16 3B 00 EB	>. t. . . ;..
0D31:0070	04 8D 16 32 00 B4 09 CD—21 CB 56 FA 89 7E EE 8B	...2. . . . !. V. ~..

—f29 19 'forward.'20

—g

No match!

Program terminated normally

—q

f 命令中的 29 为修改区的首地址,19 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 码提出。该程序汇编、连接后,进入得不够调试如下:

C>debug sample.exe

—g

No match!

Program terminated normally

结果是错误的(因源文件中的两个字符是相同的)。为检查程序的错误,将断点设在比较串之后。

—g19

AX=0D31 BX=0000 CX=0000 DX=0000 SP=FFFC BP=0000 SI=0019 DI=0032
DS=0D31 ES=0D31 SS=0D31 CS=0D36 IP=0019 NV UP EI PL ZR NA PE NC
0D36:0019 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=0D31 BX=0000 CX=0000 DX=0000 SP=FFFC BP=0000 SI=0019 DI=0032
DS=0D31 ES=0D31 SS=0D31 CS=0D36 IP=001B NV UP EI PL ZR NA PE NC
0D36:001B 8D163B00 LEA DX,[003B] DS:003B=6F4E
```

为查到 003B 单元的内容, 可查数据区如下:

—d0

```
0D31:0000 4D 6F 76 65 20 74 68 65-20 63 75 72 73 6F 72 20 Move the cursor
0D31:0010 62 61 63 6B 77 61 72 64-2E 4D 6F 76 65 20 74 68 backward. Move th
0D31:0020 65 20 63 75 72 73 6F 72-20 62 61 63 6B 77 61 72 e cursor backwar
0D31:0030 64 2E 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61 d. Match... $ No ma
0D31:0040 74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 00 tch!.. $. ....
0D31:0050 1E 2B C0 50 B8 31 0D 8E-D8 8E C0 8D 36 00 00 8D .+. P. 1. .... 6...
0D31:0060 3E 19 00 FC B9 19 00 F3-A6 75 06 8D 16 3B 00 EB >..... u...;..
0D31:0070 04 8D 16 32 00 B4 09 CD-21 CB 46 FA 8B 5E FC C1 ... 2.... !. F. ^..
```

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。可用 a 命令修改, 应用 u 命令检查修改结果。运行结果说明程序修改正确。

—a19

```
0D36:0019 jz 0021
```

```
0D36:001B
```

—u0

```
0D36:0000 1E PUSH DS
0D36:0001 2BC0 SUB AX,AX
0D36:0003 50 PUSH AX
0D36:0004 B8310D MOV AX,0D31
0D36:0007 8ED8 MOV DS,AX
0D36:0009 8EC0 MOV ES,AX
0D36:000B 8D360000 LEA SI,[0000]
0D36:000F 8D3E1900 LEA DI,[0019]
0D36:0013 FC CLD
0D36:0014 B91900 MOV CX,0019
0D36:0017 F3 REPZ
0D36:0018 A6 CMPSB
0D36:0019 7406 JZ 0021
0D36:001B 8D163B00 LEA DX,[003B]
```