

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

目 录

前 言

第一章 实验的基本要求与方法	1
1.1 实验目的与要求	1
一、实验目的	1
二、实验要求	1
1.2 实验方法	2
例 1.1 比较字符串 sample	2
第二章 程序的基本结构练习	13
2.1 循环程序设计.....	13
一、示例	13
例 2.1 表格查找 tabsrch	13
例 2.2 建立学生名次表 rank	16
二、实验题	19
实验 2.1 用表格形式显示字符	19
实验 2.2 查找匹配字符串	20
2.2 分支程序设计.....	21
一、示例	21
例 2.3 统计学生成绩 result	21
例 2.4 显示月份名 direct	24
例 2.5 显示错误信息 show_err	27
二、实验题	33
实验 2.3* 分类统计字符个数	33
2.3 子程序设计.....	34
一、示例	34
例 2.6 显示学生名次表 rank	34
例 2.7 计算工资 scremp	40
例 2.8 HANOI 塔题 hanoi	52
二、实验题	67
实验 2.4 查找电话号码	67
实验 2.5* 求 Fibonacci 数	68
第三章 I/O 程序设计	70
3.1 发声系统程序设计.....	70
一、示例	71

例 3.1 枪声程序 gun	71
例 3.2 演奏音阶程序 musex	73
二、实验题	74
实验 3.1* 乐曲程序(1)	74
实验 3.2 乐曲程序(2)	76
3.2 显示器 I/O 程序设计	76
一、示例	78
例 3.3 光标轨迹程序 draw	78
例 3.4 窗口控制程序 wdex	81
例 3.5 画横竖线程序 grnd	86
二、实验题	88
实验 3.3* 字符图形程序	88
实验 3.4 屏幕窗口程序	89
实验 3.5* 画栅栏线程序	91
3.3 键盘输入程序设计	91
一、示例	93
例 3.6 键盘处理演示程序 kbdio	93
例 3.7 键盘输入程序 keyboard	97
例 3.8 字处理演示程序 wspp	100
二、实验题	105
实验 3.6 扩充键盘处理功能的程序	105
实验 3.7* 扩充字处理功能的程序	106
3.4 中断程序设计	107
一、示例	108
例 3.9 打字计时程序 type_ex	108
二、实验题	116
实验 3.8 中断练习程序	116
第四章 文件管理	118
4.1 文件代号方式下的文件管理	118
一、示例	118
例 4.1 分页显示文件 ex_41	118
例 4.2 删除页 ex_42	126
二、实验题	136
实验 4.1 页拷贝	136
4.2 文件控制块方式下的文件管理	137
一、示例	137
例 4.3 个人档案文件管理 ex_43	137
二、实验题	147

实验 4.2 个人档案管理系统	147
第五章 高级汇编语言技术与连接技术	149
5.1 高级汇编语言技术	149
一、示例	149
例 5.1 用宏和高级汇编技术实现 IF 和 while 语句功能 ex_51	149
二、实验题	159
实验 5.1' 扩展 if 和 while 条件表达功能	159
5.2 连接技术	159
一、示例	159
例 5.2 可回卷的页显示 ex_52	159
二、实验题	173
实验 5.2 菜单使用	173
附录一 上机基本操作	175
附录二 全屏编辑程序 WordStar	177
附录三 全屏编辑程序 pced	178
附录四 行编辑程序 EDLIN	182
附录五 调试程序 DEBUG	184
附录六 汇编程序出错信息	186
附录七 IBM PC ASCII 码字符表	191

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. 上机前要做好充分准备,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. 调用串处理程序 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
;*****
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
;*****
program segment                ;define code segment
;-----
main proc far
    assume cs:program,ds:dataarea,cs: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
;-----
program 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:

```

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

```

```

string2      db      'Move the cursor forward.'
;
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。显示结果如图 1.2 所示。

—u

```

19F3:0000 1E          PUSH     DS
19F3:0001 2B00          SUB     AX,AX
19F3:0003 50          PUSH     AX
19F3:0004 B8E119       MOV     AX,19EF
19F3:0007 8ED8       MOV     DS,AX
19F3:0009 8EC9       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:

-do

```

19EE:0000 4D6F 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 64.2c 2E.20

```

再次用 d 命令查看修改结果。

--do

```

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  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"

```

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
--gdb
```

```
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 - 6F1D
```

--do

```
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 2F 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61 d.Match...$No ma
19EE:0040 74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 tch!...$.
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 77 01 40 5A 22 ...2.4.M!K.u.@Z"
```

--c29 /forward.'20

--do

```
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 d.Match...$No ma
19EE:0040 74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 tch!...$.
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
```

k

No match!

Program terminated normally

可见这种 c 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,以补是原来的字节数。

也可使用 a 命令把数据区的内容恢复原状,具体如下:

--a19ec:29

```
19EE:0029 db 'backward.'
```

19EE:0032

--do

```
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...$No ma
19EE:0040 74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 tch!...$.
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 77 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=0000 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 21 4E 6F 20 6D 61 d.Watch...$No ma
19EE:0040 74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 tch!..$.
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 >..19..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.0Z"
```

```
-f29 19 '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 . Watch...$No ma
19EE:0040 74 63 68 21 0D 0A 21 00-00 00 00 00 00 00 00 tch!..$.
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 >..19..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.0Z"
```

```
-g
```

```
No match!
```

```
Program terminated normally
```

```
-q
```

f 命令中的 29 为所修改区的首地址,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 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 backwar
19EE:0030 64 2E 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 8D 61 d.Match...$No ma
19EE:0040 74 63 68 21 0D 0A 24 00 00 00 00 00 00 00 00 tch!..$.
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 C1-D1 21 CB FF 75 01 40 5A 22 ...2.4.MIX.u.0Z"

```

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 2F 20 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61 . Match...$No ma
19EE:0040 74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 tch!..$......
19EE:0050 1E 2B C0 50 B8 EE 19 8E-D8 8E C0 8D 36 00 00 8D .*0P8n..X.0.6...
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 49 5A 22 ...2.4.W!K.u.0Z"

```

—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												

```

AX=19EE BX=0000 CX=000C DX=0000 SP=FFFC BP=0000 SI=000D DI=0026
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=000B DX=0000 SP=FFFC BP=0000 SI=000E DI=0027
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=000A DX=0000 SP=FFFC BP=0000 SI=000F DI=0028
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=0009 DX=0000 SP=FFFC BP=0000 SI=0010 DI=0029
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=0008 DX=0000 SP=FFFC BP=0000 SI=0011 DI=002A
DS=19EE ES=19EE SS=19EE CS=19F3 IP=0019 NV UP DI NG NZ AC PE CY
19F3:0019 7406 JZ 0021

```

-t

```

AX=19EE BX=0000 CX=0008 DX=0000 SP=FFFC BP=0000 SI=0011 DI=002A
DS=19EE ES=19EE SS=19EE CS=19F3 IP=001B NV UP DI NG NZ AC PE CY
19F3:001B 8D163B00 LEA DX,[003B] DS:003B=6F4E

```

-t

```

AX=19EE BX=0000 CX=0008 DX=003B SP=FFFC BP=0000 SI=0011 DI=002A
DS=19EE ES=19EE SS=19EE CS=19F3 IP=001F NV UP DI NG NZ AC PE CY
19F3:001F EB04 JMP 0025

```

-t

```

AX=19EE BX=0000 CX=0008 DX=003B SP=FFFC BP=0000 SI=0011 DI=002A
DS=19EE ES=19EE SS=19EE CS=19F3 IP=0025 NV UP DI NG NZ AC PE CY
19F3:0025 B409 MOV AH,09

```

-t

```

AX=09EE BX=0000 CX=0008 DX=003B SP=FFFC BP=0000 SI=0011 DI=002A
DS=19EE ES=19EE SS=19EE CS=19F3 IP=0027 NV UP DI NG NZ AC PE CY
19F3:0027 CD21 INT 21

```

-g

No match!

Program terminated normally

-q

从这一过程可清楚地看出每次比较的结果，一旦比较不等，则立即从串指令退出，执行下面的指令。应该注意，如果遇到系统功能调用，则不能再使用 t 或 p 命令跟踪，而应该用断点停在功能调用完成之后，然后再接着跟踪。本例中，由于不需要再跟踪，所以直接用 g 命令运行到程序结束。

前面已经提到, debug 调试期间所修改的数据段或代码段的内容只是修改了内存中的内容,而磁盘文件中的内容并未修改。如果你的执行文件是 .com, 则可在 debug 中用 n、w 命令直接把经修改后的内存单元中的内容存入磁盘,但是 .exe 文件则不允许这样做,因此,应该重新进入编辑程序,根据调试结果把源文件修改正确,经汇编、连接、执行检查正确后再保存下来。

第二章 程序的基本结构练习

2.1 循环程序设计

一、示例

例 2.1 表格查找 tabsrch

仓库管理中,总共存有有关库存品的编号、名称、数量、价格等情况的表格,根据用户提供的编号可以找到有关材料。假设表格中共有 6 种库存品,表格的格式为:

```
STOKTAB DB      '05', 'Excavators'  
          '08', 'lifters'  
          .  
          .  
          .
```

试编写一程序,根据用户提供的编号在终端上显示其名称。

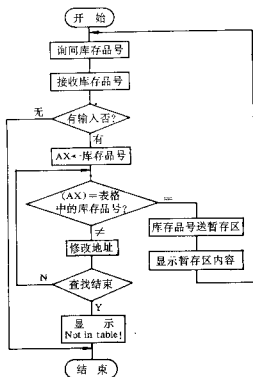


图 2.1 例 2.1 的程序框图