《汇编语言程序设计》

实

验

指

导

书

计算机科学与技术学院

二零一五年三月

目　录

实验一： 汇编环境与 DEBUG 调试

目的与要求

熟悉汇编环境与汇编源程序的调试，重点掌握常用 DEBUG 命令的用法

实验内容

1、熟悉汇编的编程环境
2、DEBUG 调试
3、 内存操作：
4、程序文件操作

实验步骤：

一、熟悉汇编的编程环境

1．汇编语言源程序的建立：

可以使用任何文本编辑器建立汇编源程序文件，如：TC、EDIT 等。但建立的文件扩展名建议为.asm，下面的源程序功能用于在屏幕上输出字符串“hello,world!”，我们通过此源程序的运行过程了解汇编的编程环境。

```
DATA    SEGMENT
        BUF     DB  “hello,world!$”
DATA    ENDS
CODE    SEGMENT
        ASSUME      CS:CODE,DS:DATA
START:  MOV   AX,DATA
        MOV   DS,AX
        LEA   DX，BUF
        MOV   AX,9
        INT 21H
        MOV   AH,4CH
        INT   21H
CODE    ENDS
        END   START
```

在 DOS 提示符下键入以下编辑器命令，如：

C:\MASM>EDIT 或 TC

然后在编辑器的窗口中通过键盘输入以上源程序。并保存为文件 abc.asm.后返回到 DOS 提示符下。

2． 将源程序文件汇编成目标程序文件：

汇编功能可以使用 Microsoft 的 MASM 进行，微软的汇编程序有多个版本，但使用方法大致相同，本实验将使用 MASM 5.0 进行汇编。

一般情况下，MASM 汇编程序的主要功能有以下 3 点：

（1）检查源程序中存在的语法错误，并给出错误信息。

（2）源程序经汇编后没有错误，则产生目标程序文件，扩展名为.OBJ。

（3）若程序中使用了宏指令，则汇编程序将展开宏指令。

源程序建立以后，在 DOS 状态下，可以采用宏汇编程序 MASM 对源程序文件进行汇编，方法是在 DOS 提示符下键入键入以下命令：

C:\MASM>MASM　abc；

此命令行后的；用于屏蔽 MASM 的参数选择提问，而使用其默认的参数，对于大多数汇编源程序，MASM 的默认参数选项是足够的，因此在此实验中将不介绍其参数选项的功能，有兴趣的同学可以去掉最后的分号进行汇编，并了解其细节。

汇编源程序若有错误，则汇编程序会给出错误信息，并在每条信息前有一数字 N，此数字标识源程序中第 N 行检测到错误。汇编过程的错误分警告错误（Warning Errors）和严重错误（Severe Errors）两种。其中警告错误是指汇编程序认为的一般性错误；严重错误是指汇编程序认为无法进行正确汇编的错误，并给出错误的个数、错误的性质。这时，就要对错误进行分析，找出原因和问题，然后再调用屏幕编辑程序加以修改，修改以后再重新汇编，一直到汇编无错误为止。

汇编成功后，将得到一个新的文件 abc.obj，此文件称为目标程序文件。

3．用连接程序生成可执行程序文件：

经汇编以后产生的目标程序文件（.OBJ 文件）并不是可执行程序文件，必须经过连接以后，才能成为可执行文件（即扩展名为.EXE）。连接过程使用以下命令进行：

C:\MASM>LINK　abc；

此命令行后的；功能同上面的 MASM。此命令运行成功后，就会得到最终的可执行文件 abc.exe。如果连接过程中出现错误，则显示出错误信息，根据提示的错误原因，要重新调入编辑程序加以修改，然后重新汇编，再经过连接，直到没有错误为止。连接以后，便可以产生可执行程序文件（.EXE 文件）。

4．程序的执行

当我们建立了正确的可执行文件以后，就可以直接在 DOS 状态下执行该程序。

C:\MASM＞abc

运行此命令后，我们会在屏幕上看到程序的输出结果：hello,world!。如果此时我们看到的结果同程序预期的结果不一致，就说明程序存在逻辑错误，这时就必须使用相应的调试程序进行调试，常用的调试程序有 DEBUG 与 Turbo　Debugger 等，此实验中，将只介绍 DEBUG 常用命令的使用方法。

二、DEBUG 调试

DEBUG 的命令格式：

DEBUG　[drive:][path][filename][.ext][param...]

其中：

drive:　是 DEBUG 将要调试的文件所在的磁盘驱动器。

path:　是查找 DEBUG 将要调试的文件所需的子目录路径，若未指定，DOS 使用当前目录。

filename[.ext]　是 DEBUG 将要调试的文件名。

param　是将被调试的程序(或文件)的命令行参数。

注：1、如果 DEBUG 命令行含有文件名，段寄存器 DS 和 ES 指向 PSP。寄存器 BX 和 CX 含有程序长度。

2、在 DEBUG 中所有输入与显示的数值数据均为十六进制数，且其后不加 H。

如果要调试上面的程序 ABC.EXE，可在 DOS 提示符下输入：

C:\MASM>DEBUG　ABC.EXE

—

此时的－为 DEBUG 的命令提示符，在此符号后可能输入许多 DEBUG 的调试命令。DEBUG 所有的命令都是单字符命令，格式为：命令名 [参数]。

在此实验中，重点介绍以下命令的功能与使用：

1．显示存储单元的命令 D，格式为：

－D [address] 或 －D[range]

例如，按指定范围显示存储单元内容的方法为：

－D 100 120

067C:0100 C7 D7 0D 0A 32 33 33 34 - D5 C5 B4 C6 30 10 42 0C ....2334....0.B……

067C:0110 03 41 42 43 44 45 46 47 - 48 49 4A 4B 4C 4D 4E 4F .ABCDEFGHIJKLMNO

067C:0120 41 A

说明：

左边是用十六进制表示的存储单元地址，0100 至 0120 是 DEBUG 要显示的存储单元段内偏移地址，而 067C 是当前数据段的段首址。中间是每个字节存储单元的内容。右边表示每个字节可以显示的字符，“.”表示是不可显示的字符。D 命令一行显示 16 个字节单元的内容，前 8 个字节与后 8 个字节之间用－连接。

这里没有指定段地址， D 命令自动显示 DS 段的内容。如果只指定首地址，则显示从首地址开始的 80 个字节的内容。如果完全没有指定地址，则显示上一个 D 命令的最后一个单元的内容。如：－D 显示下一片单元的内容

－D ES：1010 显示从 ES 段内偏移地址 1010H 开始的单元内容

－D 10L20 显示 DS 段内从偏移地址 10H 开始的连接 20H 个字节单元的内容

2．修改存储单元内容的命令 E：

E 命令有两种格式如下：

第一种格式可以用给定的内容表来替代指定范围的存储单元内容。命令格式为：

－E address [list]

例如，－E DS:100 F3'XYZ'8D

其中 F3，‘X’，‘Y’，‘Z’和 8D 各占一个字节，该命令可以用这五个字节来替代存储单元 DS：0100 到 0104 的原先的容。

第二种格式则是采用逐个单元相继修改的方法。命令格式为：

－E address

例如，－E CS:100

则可能显示为：

18E4：0100 89.

如果需要把该单元的内容修改为 78，则可以直接键入 78，再按空格键可接着显示下一个单元的内容，这样可以不断修改相继单元的内容，直到 Enter 键结束该命令为止。若要反向修改单元内容，则按减号－。

3．检查和修改寄存器内容的命令 R：

R 命令有三种格式：

1）显示 CPU 内所有寄存器内容和标志位状态，其格式为：

－R

AX=0000 BX=0000 CX=010A DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000

DS=18E4 ES=18E4 SS=18E4 CS=18E4 IP=0100 NV UP DI PL NZ NA PO NC

18E4：0100 C70604023801 MOV WORD PTR[0204]，0138 DS:0204=0000

其中：

第一行显示 8086 内通用寄存器的内容，第二行显示段寄存器及标志寄存器的内容，第三

行显示当前 IP 指向的将要被 CPU 执行的指令地址在 18E4：100 处，其机器码为 C70604023801，对应的汇编指令为 MOV　　WORD PTR[0204]，0138

2）显示和修改某个寄存器内容，其格式为：

－R　register_ name

例如，键入

－R　ax

系统将响应如下：

AX　F1F4

：

即 AX 寄存器的当前内容为 F1F4，如不修改则按 Enter 键，否则，键入欲修改的内容如：

－R　bx

BX　0369

：059F

则把 BX 寄存器的当前内容修改为 059F。

R 命令显示中标志位状态的含义如下表所示：

标 志 名	置　位	复　位
溢出 Overflow（是/否）	OV	NV
方向 Direction（减量/增量）	DN	UP
中断 Interrupt（允许/屏蔽）	EI	DI
符号 Sign（负/正）	NG	PL
零 Zero（是/否）	ZR	NZ
辅助进位 Auxiliary Carry（是/否）	AC	NA
奇偶 Parity（偶/奇）	PE	PO
进位 Carry（是/否）	CY	NC

3）显示和修改标志位状态，命令格式为：

－RF

系统将响应，如：

OV DN EI NG ZR AC PE CY －

此时如不修改其内容可按 Enter 键，否则，建入欲修改的内容，如：

OV DN EI NG ZR AC PE CY － PONZDINV

即可，键入的顺序是任意的。

4．运行命令 G：

G 命令格式为：

－G　[=address1][address2[address3　...]]

其中，地址 1 指定了运行的起始地址，如不指定则从当前的 CS:IP 开始运行。后面的地址均为断点地址，当指令执行到断点时，就停止执行并显示当前所有寄存器及标志位的内容，和下一条将要执行的指令。

5、跟踪命令 T：

T 命令有两种格式：

1）逐条指令跟踪

－T[=address]

从指定地址起执行一条指令后停下来，显示所有寄存器内空及标志位的值。如未指定则从当前的 CS:IP 开始执行。

2）多条指令跟踪

－T[=address][value]

从指定地址起执行 n 条指令后停下来，n 由 value 指定。

6、汇编命令 A：

A 命令的格式为：

－A [address]

该命令允许键入汇编语言语句，并能把它们汇编成机器代码，相继地存放在从指定地址开始的存储区中。

7、反汇编命令 U ：

U 命令有两种格式：

1）从指定地址开始，反汇编 32 个字节，其格式为：

－U[address]

例如：

－U 100

```
18E4:0100    C70604023801      MOV     WORD   PTR[0204],0138
18E4:0106    C70606020002      MOV     WORD   PTR[0206],0200
18E4:010C    C70608020202      MOV     WORD   PTR[0208],0202
18E4:0112    BB0402            MOV     BX,0204
18E4:0115    E80200            CALL    011A
18E4:0118    CD20              INT     20
18E4:011A    50                PUSH    AX
18E4:011B    51                PUSH    CX
18E4:011C    56                PUSH    SI
18E4:011D    57                PUSH    DI
18E4:011E    8B37              MOV     SI,[BX]
```

如果地址被省略则从上一个 U 命令的最后一条指令的下一个单元开始显示 32 个字节。

2）对指定范围内的存储单元进行反汇编，格式为：

－U[range]

例如：

－U 100　10C

```
18E4:0100    C70604023801      MOV   WORD PTR[0204],0138
18E4:0106    C70606020002      MOV   WORD PTR[0206],0200
18E4:010C    C70608020202      MOV   WORD PTR[0208],0202
```

8．命名命令 N：

N 的格式为：

－N　filespecs

命令把文件标识符 filespecs 格式化在文件控制块中，以便在其后用 L 或 W 命令把文件装入或存盘。filespecs 的格式可以是：[d:][path]　filename[.ext]

例如：

－N　myprog

9．装入命令 L：

L 命令的功能有两种，此实验只介绍其中一种功能：　装入指定文件。

其格式为：

－L [address]

此命令装入已在 N 命令中所指定的文件。如未指定地址，则装入 CS:0100 开始的存储区中。

10．写命令 W：

W 命令的功能有两种，此实验只介绍其中一种功能：把数据写入指定的文件中。

其格式为：

－W　[write address]

此命令执行前要求先利用 R 命令在 BX、CX (CX 为低字,BX 为高字)中存入程序段的长度(字节数)，可以利用 R BX　和 R CX 命令。此命令把指定的存储区中的数据写入由 N 命令所指定的文件中。如未指定地址则数据从 CS: 0100 开始。要写入文件的字节数应先放入 BX 和 CX 中。

11．退出 DEBUG 命令 Q：

格式为：

－Q

它退出 DEBUG，返回 DOS。本命令无存盘功能，如需存盘应先使用 W 命令。

三、 内存操作：

1.进入 DEBUG，执行命令：－eds:0 41"abc"42 43 后，可以修改数据段内的几个字节单元？如何用 D 命令查看这些单元的内容？这些内容对应的 ASCII 字符是什么？

2.可用什么命令查看从 ds:0 开始的连续 100 个字节单元的内容？

3.若要将 ds:10,ds:12,ds:14 三个存储单元的内容修改为４１，则应怎样操作？

四、程序文件操作

1． 进入 DEBUG，输入 A 命令汇编以下语句：

－A

1479:0100 mov dl,41

1479:0102 mov ah,2

1479:0104 int 21

1479:0106 int 20

1479:0108

2． 若要将上面的程序保存到当前路径中，并以 disp.com 命名，则应进行哪些操作，请详细给出每一步的命令行：

3． 退出 DEBUG 回到 DOS，输入 disp 回车后，此命令的运行结果是什么？

4． 请用 DEBUG 命令查出上面四条语句对应的机器指令码：

5． 此时 cs:0101 地址的内容是什么？可用什么命令将其修改为 61？

6． 运行命令：－ＷＣＳ：１００后，回到ＤＯＳ下输入 disp 回车后，此时运行的结果是什么？

实验二：8086 汇编语言顺序程序设计

实验目的：

熟悉顺序程序设计过程。

实验内容：

1、掌握顺序程序设计方法。

2、学习数据传送及算术和逻辑运算指令的用法。

3、熟悉在 PC 机上建立、汇编、连接、调试和运行 8086 汇编语言程序的过程。

实验步骤：

一. 编程实现以下功能：

将两个 32 位十进制数相加，被加数和加数存放在 DATA 段内的 N1 与 N2 两个变量中，要求将相加结果送入 N3 变量中。

源程序如下所示，使用编辑器建立源程序文件 LAB2.ASM。

```
DATA    SEGMENT
        N1    DD    12345678H
        N2    DD    87654321H
        N3    DD    ?
DATA    ENDS
CODE    SEGMENT
        ASSUME    CS：CODE，DS：DATA
START：
        MOV    AX，DATA
        MOV    DS，AX
        MOV    AX，WORD    PTR    N1
        MOV    DX，WORD    PTR    N1＋2
        MOV    CX，WORD    PTR    N2
        MOV    BX，WORD    PTR    N2＋2
        ADD    AX，CX
        ADC    DX，BX
        MOV    WORD    PTR    N3，AX
        MOV    WORD    PTR    N3＋2，DX
        MOV    AH，4CH
        INT    21H
CODE    ENDS
        END    START
```

使用相应的文本编辑器建立文件 LAB2.asm，内容如上所示。

二. 生成可执行文件：

1.汇编：

```
C:\masm> masm lab2;
```

2.连接：

C:\masm> link lab2;

三.运行及调试：

1. 运行：

C:\masm>debug lab2.exe

－U0　　；通过反汇编查找程序的断点

1425:0000 B82414　　MOV　　AX,1424
1425:0003 8ED8　　MOV　　DS,AX
1425:0005 A10000　　MOV　　AX,[0000]
1425:0008 8B160200　　MOV　　DX,[0002]
1425:000C 8B0E0400　　MOV　　CX,[0004]
1425:0010 8B1E0600　　MOV　　BX,[0006]
1425:0014 03C1　　ADD　　AX,CX
1425:0016 13D3　　ADC　　DX,BX
1425:0018 A30800　　MOV　　[0008],AX
1425:001B 89160A00　　MOV　　[000A],DX
1425:001F B44C　　MOV　　AH,4C　　；找到程序的断点为 001FH

－T=0 2　；加载数据段

－D0　　；查看原始数据是否正确，下划线上的数据就是 N1 与 N2 的值

1424:0000　**78 56 34 12 21 43 65 87**-00 00 00 00 00 00 00 00　　xV4.!Ce.....
1424:0010　B8 24 14 8E D8 A1 00 00-8B 16 02 00 8B 0E 04 00　　.$.....
1424:0020　8B 1E 06 00 03 C1 13 D3-A3 08 00 89 16 0A 00 B4　　.....

－G=0　1F　；运行程序至断点 1F 处

AX=9999　BX=8765　CX=4321　DX=9999　SP=0000　BP=0000　SI=0000　DI=0000
DS=1424　ES=1414　SS=1424　CS=1425　IP=001F　　NV UP EI NG NZ NA PE NC
1425:001F B44C　　MOV　　AH,4C

－D0　　；查看程序运行结果，下划线上的数据就是相加的和

1424:0000　78 56 34 12 21 43 65 87-99 99 99 99 00 00 00 00　　xV4.!Ce.....
1424:0010　B8 24 14 8E D8 A1 00 00-8B 16 02 00 8B 0E 04 00　　.$.....
1424:0020　8B 1E 06 00 03 C1 13 D3-A3 08 00 89 16 0A 00 B4　　.....

2．调试：

若要判断此程序对于其它 32 位数相加是否正确，则需要使用 DEBUG 进行程序调试。

例：判断 234B8074H＋658A1D61H 是否正确的方法如下：

－L　　；重新加载可执行文件
－T＝0　2　　；加载程序的数据段，以便修改 N1 与 N2 的值
－D0　　；查看数据段的内容
1424:0000　**78 56 34 12 21 43 65 87**-00 00 00 00 00 00 00 00　　xV4.!Ce.....
1424:0010　B8 24 14 8E D8 A1 00 00-8B 16 02 00 8B 0E 04 00　　.$.....
1424:0020　8B 1E 06 00 03 C1 13 D3-A3 08 00 89 16 0A 00 B4　　.....
－E0　　；修改 N1 为 234B8074H，N2 为 658A1D61H
1424:0000　78.74　56.80　34.4B　12.23　21.61　43.1D　65.8A　87.65
－G　1F　；带断点运行，此处不能使用 G＝0　1F

AX=9DD5 BX=658A CX=1D61 DX=88D5 SP=0000 BP=0000 SI=0000 DI=0000
DS=1424 ES=1414 SS=1424 CS=1425 IP=001F OV UP EI NG NZ AC PO NC
1425:001F B44C MOV AH,4C
－D0 ；查看运行结果，下划线上的数据就是相加的和
1424:0000 74 80 4B 23 61 1D 8A 65-D5 9D D5 88 00 00 00 00 t.K#a..e.....
1424:0010 B8 24 14 8E D8 A1 00 00-8B 16 02 00 8B 0E 04 00 .$.....
1424:0020 8B 1E 06 00 03 C1 13 D3-A3 08 00 89 16 0A 00 B4 .....

四．编程实现：将从 2000H 单元开始的连续 128 个单元的内容进行清零。

五．编程实现：将 3000H 单元的一个字节的内容进行拆分，高半字节放进 3001H 单元的低半部分，其低半字节放进 3002H 单元的低半部分。

实验三：8086 汇编语言分支程序设计

实验目的：

熟悉分支程序的调试方法。

实验内容：

编写一个程序，显示 AL 寄存器中的两位十六进制数

实验步骤：

一. 编程实现在显示器上输出 AL 中的内容：

源程序如下所示，编辑下面的源程序到文件 lab3.asm 中：

```
CODE SEGMENT
       ASSUME CS:CODE
START: MOV AL,3EH     ; 此处假设为 3EH
       MOV BL,AL
       MOV DL,AL
       MOV CL,4
       SHR DL,CL
       CMP DL,9
       JBE NEXT1
       ADD DL,7
NEXT1: ADD DL,30H
       MOV AH,2
       INT 21H                    ;显示高位 ASCII 码
       MOV DL,BL
       AND DL,0FH
       CMP DL,9
       JBE NEXT2
       ADD DL,7
NEXT2: ADD DL,30H
       MOV AH,2
       INT 21H                    ;显示低位 ASCII 码
       MOV AH,4CH
       INT 21H
CODE  ENDS                      ;返回 DOS
       END START
```

使用相应的文本编辑器建立文件 LAB3.asm，内容如上所示。

二. 生成可执行文件：

1.汇编：

```
C:\masm> masm lab3;
```

2.连接：

```
C:\masm> link lab3;
```

三.运行及调试：

1. 运行：

C:\masm>debug lab3.exe

－U0　　；通过反汇编查找程序的断点

－T=0 2　；加载数据段

－D0　　；查看原始数据是否正确

－G=0　XX　；运行程序至断点 XX 处

－R　　；查看程序运行结果

以上命令执行的细节可参照实验二中的说明。

2．调试：

修改 AL 的内容，判断此程序是否能正确显示其中的内容的方法。

例：修改 AL 内容为 9AH：

－L　　；重新加载可执行文件

－A0　　；重新修改 MOV　AL，3EH 指令

361E：0000　MOV　AL，9A

361E：0002

－G＝0　x x；带断点运行

－R　　；查看程序运行结果

四．编写一个数据区移动程序，要考虑源数据区与目的数据区有重叠的情况。

五．编程完成一个字符串的统计，要求分别统计出字母、数字、其它字符的个数。

实验四：8086 汇编语言循环程序设计

实验目的：

熟悉循环程序的调试方法。

实验内容：

1、熟悉循环程序的结构及循环指令的用法。

2、掌握循环程序的设计、调试方法

实验步骤：

一. 编写程序实现下述功能：

设有一段英文，其字符变量名为 ENG，并以$字符结束（如下定义）。程序检查单词 SUN 在文中出现的次数，并以格式“SUN echo times:”显示出次数。

源程序如下所示，编辑到文件 lab4.asm 中：

```
DATA SEGMENT
  ENG DB "asunbsunxysunbf$"
  COUNT DB 0
  STR DB "SUN'S NUMBER IS :$ "
DATA ENDS
CODE SEGMENT
     ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
            MOV DS,AX
            MOV SI,-1
NEXT: INC SI
           CMP ENG[SI],'S'
           JZ L1
           JMP L
L:      CMP ENG[SI],'$'
          JZ EXIT
          JMP NEXT
L1:    INC SI
          CMP ENG[SI],'U'
          JZ L2
          JMP L
L2:    INC SI
            CMP ENG[SI] ,'N'
            JZ L3
            JMP L
L3:    INC COUNT
```

```
        JMP NEXT
        MOV AX,COUNT+1
EXIT:
        LEA DX,STR
        MOV AH,9
        INT 21H
        MOV DL,COUNT
        MOV DH,0
        ADD DL,30H
        MOV AH,2
        INT 21H
        MOV AH,4CH
        INT 21H
    CODE ENDS
        END START
```

使用相应的文本编辑器建立文件 LAB4.asm，内容如上所示。

二. 生成可执行文件：

1.汇编：

C:\masm> masm lab4;

2.连接：

C:\masm> link lab4;

三.运行及调试：

1. 运行：

C:\masm>debug lab4.exe

－T=0 2 ；加载数据段

－D0 ；查看原始数据是否正确

－G=0 ；运行程序

；查看程序运行结果

以上命令执行的细节可参照实验二中的说明。

2. 调试：

通过 T 命令进行单步运行，以判断此程序执行的流程。

例：－T

四. 编程实现：将 DX 中的十六进制数转换为 ASCII 码，存放到 BUF 开始的内存单元中去，并在屏幕显示出数值。

五. 编程实现利用冒泡法对一组数据进行逆序排序。

实验五：8086 汇编语言子程序程序设计

实验目的：

熟悉子程序的设计方法。

实验内容：

1、熟悉子程序的结构及注意事项

2、掌握子程序的设计、调试方法。

实验步骤：

一. 使用子程序设计编程实现求数组 ARY 元素之和：

编辑下面的源程序到文件 lab5.asm 中：

```
  CODE SEGMENT
      ORG 100H
  ASSUME CS:CODE,DS:CODE,SS:CODE
  MAIN PROC FAR
    MOV AX,CODE
    MOV DS,AX
    MOV TABLE ,OFFSET ARY
    MOV TABLE+2,OFFSET COUNT
    MOV TABLE+4,OFFSET SUM
    MOV BX,OFFSET TABLE
    CALL PROADD
    MOV AX,4C00H
    INT 21H
 MAIN ENDP
  PROADD PROC NEAR
     PUSH AX
     PUSH CX
     PUSH SI
     PUSH DI
     MOV SI,[BX]
     MOV DI,[BX+2]
     MOV CX,[DI]
     MOV DI,[BX+4]
     XOR AX,AX
NEXT:ADD AX,[SI]
        ADD SI,2
        LOOP NEXT
        MOV [DI],AX
        POP DI
```

```
        POP SI
        POP CX
        POP AX
        RET
PROADD ENDP
ARY DW 1,2,3,4,5,6,7,8,9,10
COUNT DW 10
SUM DW ?
TABLE DW 3 DUP(?)
CODE ENDS
    END MAIN
```

使用相应的文本编辑器建立文件 LAB5.asm，内容如上所示。

二. 生成可执行文件：

1.汇编：

```
C:\masm> masm lab5;
```

2.连接：

```
C:\masm> link lab5;
```

三. 请写出此程序中的变量 ary,count,sum 的 EA，并判断此程序的功能：

四. 用 debug 调试此程序时,第一条指令的段内 EA 是多少?此程序数据段内偏移地址为 0 的字单元数据为多少?其对应的机器指令是什么?

－L　　；加载程序文件 lab5.exe

－R　　；查看 IP 与 CS 寄存器的内容

－D DS：0　　；查看当前数据段内偏移地址为 0 的字单元数据

－U　　；查看机器指令

五．编写一个子程序，实现在屏幕的指定位置，用指定颜色，显示一个用 0 结尾的字符串。

六．编写一个子程序，实现将 word 型数据以十进制形式显示出来。

实验六：8086 汇编语言中断程序设计

实验目的：

熟悉中断程序的设计方法。

实验内容：

1、熟悉 8086 中断的调用方法。

2、掌握 8086 中断的设计方法。

实验步骤：

一、请使用 DOS 中断设计编程实现向文件 ls.txt 中写入 100 个连续的字符 A：

源程序如下：

```
DATA SEGMENT
   BUF DB 100 DUP ('a')
   FNAME DB "LS.TXT",0
   FNUM DW ?
DATA ENDS
CODE SEGMENT
   ASSUME CS:CODE,DS:DATA
START:
        MOV AX,DATA
        MOV DS,AX
        MOV AH,3CH
        LEA DX,FNAME
        MOV CX,0          ；语句 1
        INT 21H
        JC EXIT
        MOV FNUM,AX
        MOV BX,AX         ；语句 2
        MOV CX,100
        MOV AH,40H
        LEA DX ,BUF
        INT 21H
        MOV BX,FNUM
        MOV AH,3EH
        INT 21H
EXIT:
        MOV AH,4CH
        INT 21H
CODE ENDS
   END START
```

使用相应的文本编辑器建立文件 LAB7.asm，内容如上所示。

二．汇编并运行此程序后，在当前目录建立的文件名是什么？其内容是什么？

1.汇编：

```
C:\masm> masm lab7;
```

2.连接：

```
C:\masm> link lab7;
```

3.运行：

```
C:\masm> lab7
```

三.若将语句 1 改为 mov cx,1，则运行情况与前面会有什么区别？

四.若将语句 1 改为 mov cx,2，则运行结果同上会有什么不同？并简要说明此语句的作用.

五.若将语句 2 改为 mov bx,1,则运行结果会有什么不同?简要说明则语句的作用.

六． 编写 0 号中断的处理程序，使得在除法溢出发生时，在屏幕中间显示字符串“divide error！”，然后返回到 DOS。

实验七：系统设计

实验目的：

熟悉汇编语言的综合编程。

实验内容：

使用汇编语言设计一个基于 8086 的简单系统。要求画出每个系统子功能的实现流程图，及相应代码的功能说明。

实验步骤：（略）

课程设计题目可从以下选择一个：

（1） 磁盘文件系统设计

（2） 数制转换系统

（3） 字符串处理系统

（4） 随机数产生及运算系统

（5） 难度与以上题目相当的自选题

附录 1 常用 DOS 命令

1. 显示目录命令 DIR

该命令用来列出指定盘、指定目录或指定文件的目录。 命令格式有一下三种：

DIR [<盘符>][/P][/W]

DIR [<目录路径名>][/P][/W]

DIR [<文件路径名>][/P][/W]

其中/P 表示分页显示； /W 表示紧缩格式显示，即一行显示多个文件。 [ ]中内容为任选项； <>中内容必须输入。

文件路径名为： [<盘符>][<路径>]<文件名>

目录路径名为： [<盘符>]<路径>

文件名中可用通配符“ *”和“ ?”，“ *”代表任意一串字符，“ ?”代表任意一个字符。例如：若要显示 D 盘 MASM 目录下的所有扩展名为 ASM 文件的 DOS 命令为：

DIR D:\MASM*.ASM

2. 文件改名命令 REN

该命令用来更改文件名，格式为：

REN <旧文件路径名> <新文件名>

旧文件路径名定义同第一点中的文件路径名，文件名中可用通配符“ *”和“ ?”。例如：若要将 D 盘 MASM 目录下的 test.lst 文件改名为 test.asm，其 DOS 命令为：

REN D:\MASM\TEST.LST TEST.ASM

3. 文件复制命令 COPY

该命令将一个或多个文件复制成副本，格式为：

COPY <文件路径名> <文件路径名>

文件路径名定义同第一点中的文件路径名，文件名中可用通配符“ *”和“ ?”。例如：若要将 D 盘 MASM 目录下的所有扩展名为 ASM 的文件复制到 C 盘的 EXAMPLE 目录下，其 DOS 命令为：

COPY D:\MASM*.ASM C:\EXAMPLE

若要将当前目录下的 TEST.ASM 文件复制成 TEST1.ASM，其 DOS 命令为：

COPY TEST.ASM TEST1.ASM

4. 文件删除命令 DEL

该命令将一个或多个文件删除，格式为：

DEL <文件路径名>

文件路径名定义同第一点中的文件路径名，文件名中可用通配符“ *”和“ ?”。例如：若要将 D 盘 MASM 目录下的所有扩展名为 ASM 的文件删除，其 DOS 命令为：

DEL D:\MASM*.ASM

5. 建立子目录命令 MD

该命令用于建立子目录，格式为：

MD <目录路径名>

目录路径名定义同第一点中的目录路径名。

例如： MD SUBDIR1

MD \SUBDIR1\SUBDIR2

MD \USER1

其中第一条命令在当前目录下建一个名为 SUBDIR1 的子目录；第二条命令在子目录 SUBDIR1 中建立 SUBDIR2 子目录；第三条命令表示在根目录中建立 USER1 子目录。

6. 删除子目录命令 RD

该命令用于删除一个空目录，但不允许删除根目录和当前目录，格式为：

RD <目录路径名>

目录路径名定义同第一点中的目录路径名。例如：若要删除根目录下的 USER1 子目录，其 DOS 命令为：

RD \USER1

在删除 USER1 子目录前，子目录 USER1 必须为空，且 USER1 不是当前目录。

7. 改变当前目录命令 CD

该命令用于显示或改变当前目录，格式为：

CD [<目录路径名>]

目录路径名定义同第一点中的目录路径名。

例如：

CD \USER1

CD \

CD ..

CD

其中第一条命令是将根目录下的 USER1 子目录设为当前目录；第二条命令是将根目录设为当前目录；第三条命令是将当前目录的上一级目录设为当前目录（即退回到上一级目录）。第四条命令为显示当前目录。

8. 设置可执行文件的搜索路径命令 PATH

用户在当前目录中工作时，时常会运行其它目录下的可执行文件（即扩展名为 EXE 或 COM 的文件），PATH 命令可满足这一要求。 PATH 命令用来指出，假如在当前目录中找不到可执行文件时应进一步去查找的目录。

格式为：

PATH [<目录路径名>][； <目录路径名>„]

若命令不带参数，将显示当前设置的搜索路径。例如将搜索路径设置为 C 盘的 DOS 目录以及 D 盘的 MASM 目录的 DOS 命令为：

PATH C:\DOS;D:\MASM

附录 2 DEBUG 主要命令

DEBUG 是为汇编语言设计的一种调试工具，它通过单步、设置断点等方式为汇编语言程序员提供了非常有效的调试手段。

1. DEBUG 程序的调用

在 DOS 的提示符下，可键入命令：

C:>DEBUG [d:][path][文件名][参数 1][参数 2]

其中文件名是被调试文件的名字，它必须是可执行文件（ EXE），两个参数是运行被调试文件所需要的命令参数，在 DEBUG 程序调入后，出现提示符“－”，此时，可键入所需的 DEBUG 命令。

2. DEBUG 的主要命令

⑴ 显示内存单元内容的命令 D，格式为：

-D [地址] 或 -D [范围]

例如，显示指定范围内存单元内容的方法为：

-D 100 1FF

18E4:0100 47 06 04 02 38 01 47 06 － 06 02 00 02 47 06 08 02 G. . . 8. G. . . . . G. . .

18E4:0110 02 02 3B 04 02 68 02 00 － 4D 20 50 51 56 57 8B 37 . . ; . . h. . M PQVW. 7

其中左边为十六进制表示形式，右边为 ASCII 码表示形式，“. ”表示不可显示字符。这里没有指定段地址， D 命令自动显示 DS 段的内容。

⑵ 修改内存单元内容的命令 E，它有两种格式

1) 用给定内容代替指定范围的单元内容，格式为：

-E 地址 内容表

例如： -E DS:100 F358595A8D，即用 F3， 58， 59， 5A， 8D 五个字节代替内存单元 DS:100 到 DS:104 的内容。

2) 逐个单元相继地修改，格式为：

-E 地址

例如： -E DS:100

18E4:0100 89.78

此命令是将 0100 单元内容 89 改为 78。 78 是程序员从键盘输入的。程序员在修改完一个单元后，可按“空格”键继续修改下一单元内容，直至按“回车”键结束该命令。

⑶ 检查和修改寄存器内容的命令 R，它有三种方式

1) 显示 CPU 内部所有寄存器内容和标志寄存器中的各标志位状态

-R

该命令可显示 AX， BX， CX， DX， SP， BP， SI， DI， DS， ES， SS， CS， IP 及标志寄存器内容。

R 命令显示中标志位状态的含义如下所示：

标 志 名	置 位	复 位
溢出 Overflow（是 / 否）	OV	NV
方向 Direction（减量） / 增量	DN	UP
中断 Interrupt（允许 / 屏蔽）	EI	DI
符号 Sign（负 / 正）	NG	PL
零 Zero（是 / 否）	ZR	NZ
辅助进位 Auxiliary Carry（是 / 否）	AC	NA
奇偶 Parity（偶 / 奇）	PE	PO
进位 Carry（是 / 否）	CY	NC

2) 显示和修改某个指定寄存器内容，格式为：

-R 寄存器名

例如： -R AX

系统响应如下：

AX F130

:

表示 AX 当前内容为 F130，此时若不对其作修改，可按“回车”键，否则，键入修改内容。

3) 显示和修改标志寄存器内容

-RF

系统将给出响应，例如：

OV DN EI NG ZR AC PE CY-

这时若不作修改可按“回车”，否则在“ -”之后键入修改值，键入顺序任意，各标志位的取值如上表所示。

⑷ 运行命令 G，格式为:

-G [=地址 1][地址 2[地址 3,,]]

其中，地址 1 指定了运行的起始地址，后面的均为断点地址，当指令执行到断点时，就停止执行并显示当前所有寄存器及标志位的内容和下一条要执行的指令。

⑸ 跟踪命令 T，它有两种格式：

1) 逐条指令跟踪，格式为：

-T[＝地址]

该命令从指定地址起执行一条指令后停下来，显示所有寄存器及标志位的内容，如未指定地址从当前的 CS:IP 开始执行。

2) 多条指令跟踪，格式为：

-T[＝地址][值]

该命令从指定地址起执行 n 条指令后停下来， n 由[值]确定。

⑹ 汇编命令 A，格式为：

-A [地址]

该命令允许输入汇编语言语句，并能把它们汇编成机器代码，相继地存放在从指定地址开始的存储区中。必须注意：输入的数字均默认为十六进制数。

⑺ 反汇编命令 A，它有两种格式：

1) 从指定地址开始，反汇编 32 个字节，其格式为：

-U [地址]

2) 对指定范围内的存储单元进行反汇编，其格式为：

-U [范围]

⑻ 命名命令 N，格式为：

-N 文件标识符 [文件标识符]

该命令将两个文件标识符格式化在 CS:5CH 和 CS:6CH 的两个文件控制块内，以便使用 L 或 W 命令把文件装入或者存盘。文件标识符格式为： [d:][path] 文件名[.扩展名]

⑼ 装入命令 L，它有两种功能：

1) 将磁盘上指定扇区的内容装入到内存指定地址起始的单元中，其格式为：

-L 地址 驱动器 扇区号 扇区数

2) 装入指定文件，其格式为：

-L [地址]

此命令装入已在 CS:5CH 中格式化的文件控制块所指定的文件。如未指定地址，则装入 CS:0100

开始的存储区中。

⑽ 写命令 W，它有两种功能：

1) 将数据写入磁盘的指定扇区，其格式为：

-W 地址 驱动器 扇区号 扇区数

2) 将数据写入指定文件中，其格式为：

-W [地址]

此命令把指定内存区域中的数据写入由 CS:5CH 处的文件控制块所指定的文件中。如未指定地址，则数据从 CS:0100 开始。要写入文件的字节数应先放入 BX 和 CX 中。

⑾ 退出 DEBUG 命令 Q，格式为：

-Q

该命令退出 DEBUG 程序，返回 DOS。

附录 3 汇编程序出错信息

编　码	说　明
0	Block nesting error 嵌套过程、段、结构、宏指令等不是正确结束。
1	Extra character on line 当一行上已接受了定义指令的足够信息，而又出现了多余的字符。
2	Register already defined 汇编内部出现逻辑错误。
3	Unknown symbol type 在符号语句的类型字段中，有些不能识别的东西。
4	Redefinition of symbol 在第二遍扫视时，接着又定义一个符号。
5	Symbol is multi-defined 重复定义一个符号。
6	Phase error between passes 程序中有模棱两可的指令，以至于在汇编程序的两次扫视中， 程序标号的位置在数值上改变了。
7	Already had ELSE clause 在 ELSE 从句子试图再定义 ELSE 从句。
8	Not in conditional block 在没有提供条件汇编指令的情况下，指定了 ENDIF 或 ELSE。
9	Symbol not defined 符号没有定义。
10	Syntax error 语句的语法与任何可识别的语法不匹配。
11	Type illegal in context 指定的类型在长度上不可接收。
12	Should have been group name 给出的组合不符合要求。
13	Must be declared in pass 1 得到的不是汇编程序所要求的常数值。例如，向前引用的向量长度。
14	Symbol type usage illegal PUBLIC 符号的使用不合法。
15	Symbol already different kind 企图定义与以前定义不同的符号。
16	Symbol is reserved word 企图非法使用一个汇编程序的保留定（例如，宣布 MOV 为一个变量）。
17	Forward reference is illegal 向前引用必须是在第一遍扫视中定义过的。
18	Must be register 希望寄存器作为操作数，但用户提供的是符号而不是寄存器。

编　码	说　明
19	Wrong type of register 指定的寄存器类型并不是指令中或伪操作中所要求的。例如 ASSUME AX。
20	Must be segment or group 希望给出段或组，而不是其它。
21	Symbol has no segment 想使用带有 SEG 的变量，而这个变量不能识别段。
22	Must be symbol type 必须是 WORD、 DW、 QW、 BYTE 或 TB，但接收的是其它内容。
23	Already defined locally 试图定义一个符号作为 EXTERNAL，但这个符号已经在局部定义过了。
24	Segment parameters are changed SEGMENT 的自变量与第一次使用这个段的情况不一样。
25	Not proper align/combine type SEGMENT 参数不正确。
26	Reference to mult defined 指令引用的内容已是多次定义过的。
27	Operand was expected 汇编程序需要的是操作数，但得到的却是其它内容。
28	Operator was expected 汇编程序需要的是操作符，但得到的却是其它内容。
29	Division by 0 or overflow 给出一个用 0 作除数的表达式。
30	Shift count is negative 移位表达式产生的移位计数值为负数。
31	Operand type must match 在自变量的长度或类型应该一致的情况下，汇编程序得到的并不一样。例如，交换。
32	Illegal use of external 用非法手段进行外部使用。
33	Must be record field name 需要的是记录字段名，但得到的是其它东西。
34	Must be record or field name 需要的是记录名或字段名，但得到的是其它东西。
35	Operand must have size 需要的是操作数的长度，但得到的是其它内容。
36	Must be var,label or constant 需要的是变量、标号或常数，但得到的是其它内容。
37	Must be structure field name 需要的是结构字段名，但得到的是其它内容。
38	Left operand must have segment 操作数的右边要求它的左边必须是某个段。

编　码	说　明
39	One operand must be const 这是加法指令的非法使用。
40	Operands must be same or 1 abs 这是减法指令的非法使用。
41	Normal type operand expected 当需要变量、标号时，得到的却是 STRUCT、 FIELDS、 NAMES、 BYTE、 WORD 或 DW。
42	Constant was expected 需要的是一个常量，得到的却是另外一个内容。
43	Operand must have segment SEG 伪操作使用不合法。
44	Must be associated with data 有关项用的是代码，而这里需要的是数据，例如一个过程的 DS 取代。
45	Must be associated with code 有关项用的是数据，而这里需要的是代码。
46	Already have base register 试图重复基地址。
47	Already have index register 试图重复变址地址。
48	Must be index or base register 指令需要基址或变址寄存器，而指定的是其它寄存器。
49	Illegal use of register 在指令中使用了 8088 指令中没有的寄存器。
50	Value is out of range 数值大于需要使用的，例如将 DW 传送到寄存器中。
51	Operand not in IP segment 由于操作数不在当前 IP 段中，因此不能存取。
52	Improper operand type 使用的操作数不能产生操作码。
53	Relative jump out of range 指定的转移超出了允许的范围（－128～＋127 字节）。
54	Index disp.must be constant 变址寄存器的位移量必须是常数。
55	Illegal register value 指定的寄存器值不能放入“ reg”字段中。（即“ reg”字段大于 7）
56	No immediate mode 指定的立即方式或操作码都不能接收立即数。例如： PUSH。
57	Illegal size for item 引用的项的长度是非法的。例如：双字移位。
58	Byte register is illegal 在上下文中，使用一个字节寄存器是非法的。例如： PUSH AL。

编　码	说　明
59	CS register illegal usage 试图非法使用 CS 寄存器。例如： XCHG CS， AX
60	Must be AX or AL 某些指令只能用 AX 或 AL。例如： IN 指令。
61	Improper use of segment reg. 段寄存器使用不合法。例如： 1 立即数传送到寄存器。
62	No or unreachable CS 试图转移到不可到达的标号。
63	Operand combination illegal 在双操作数指令中，两个操作数的组合不合法。
64	Near JMP/CALL to different CS 企图在不同的代码段内执行 NEAR 转移或调用。
65	Label can’ t have seg override 非法使用段取代。
66	Must have opcode after prefix 使用前缀指令之后，没有正确的操作码说明。
67	Can’ t override ES segment 企图非法地在一条指令中取代 ES 寄存器。例如：存储字符串。
68	Can’ t reach with segment reg. 没有使变量可达到的 ASSUME 语句。
69	Must be in segment block 企图在段外产生代码。
70	Can’ t use EVEN on BYTE segment 被提出的是一个字节段，但试图使用 EVEN。
71	Forward needs override 目前不使用这个信息。
72	Illegal value for DUP count DUP 计数必须是常数，不能是 0 或负数。
73	Symbol already external 企图定义一个局部符号，但此符号已经是外部符号了。
74	DUP is too large for linker DUP 嵌套太长，以至于从连接程序不能得到所要的记录。
75	Usage of ? (indeterminate) bad “？”使用不合适。例如：？ +5。

附录 4 常用字符 ASCII 码值

字符	ASCII 值	字符	ASCII 值	字符	ASCII 值	字符	ASCII 值
NUL	00	**4**	34	**M**	4D	**f**	66
BS	08	**5**	35	**N**	4E	**g**	67
LF	0A	**6**	36	**O**	4F	**h**	68
CR	0D	**7**	37	**P**	50	**i**	69
ESC	1B	**8**	38	**Q**	51	**j**	6A
SP	20	**9**	39	**R**	52	**k**	6B
!	21	**:**	3A	**S**	53	**l**	6C
“	22	**;**	3B	**T**	54	**m**	6D
#	23	**<**	3C	**U**	55	**n**	6E
$	24	**=**	3D	**V**	56	**o**	6F
%	25	**>**	3E	**W**	57	**p**	70
&	26	**?**	3F	**X**	58	**q**	71
‘	27	**@**	40	**Y**	59	**r**	72
(	28	**A**	41	**Z**	5A	**s**	73
)	29	**B**	42	**[**	5B	**t**	74
*****	2A	**C**	43	****	5C	**u**	75
+	2B	**D**	44	**]**	5D	**v**	76
,	2C	**E**	45	**^**	5E	**w**	77
-	2D	**F**	46	**_**	5F	**x**	78
.	2E	**G**	47	**`**	60	**y**	79
/	2F	**H**	48	**a**	61	**z**	7A
0	30	**I**	49	**b**	62	**{**	7B
1	31	**J**	4A	**c**	63	**|**	7C
2	32	**K**	4B	**d**	64	**}**	7D
3	33	**L**	4C	**e**	65	**~**	7E

NUL 空　　BS 退格
LF 换行　　CR 回车
ESC 退出　　SP 空格

附录 5 8088 / 8086 指令系统

附表 5.1 指令符号说明

符　号	说　明
r8	任意一个 8 位通用寄存器 AH、 AL、 BH、 BL、 CH、 CL、 DH、 DL
r16	任意一个 16 位通用寄存器 AX、 BX、 CX、 DX、 SI、 DI、 BP、 SP
reg	代表 r8、 r16
seg	段寄存器 CS、 DS、 ES、 SS
m8	一个 8 位存储器单元
m16	一个 16 位存储器单元
mem	代表 m8、 m16
i8	一个 8 位立即数
i16	一个 16 位立即数
imm	代表 i8、 i16
dest	目的操作数
src	源操作数
label	标号

附表 5.2 指令汇编格式

指令类型	指令汇编格式	指令功能简介	备 注
传送指令	MOV reg/mem,imm	dest←src	
	MOV reg/mem/seg,reg		CS 除外
	MOV reg/seg,mem		CS 除外
	MOV reg/mem,seg		
交换指令	XCHG reg,reg	reg←→reg/mem	
	XCHG reg/mem,reg		
转换指令	XLAT label	AL←[BX+AL]	
	XLAT		
堆栈指令	PUSH r16/m16/seg	入栈	
	POP r16/m16/seg	出栈	CS 除外
标志传送	CLC	CF←0	
	STC	CF←1	
	CMC	CF←$\overline{CF}$	
	CLD	DF←0	
	STD	DF←1	
	CLI	IF←0	
	STI	IF←1	
	LAHF	AH←标志寄存器低字节	
	SAHF	标志寄存器低字节←AH	
	PUSHF	标志寄存器入栈	
	POPF	出栈到标志寄存器	
地址传送	LEA r16,mem	r16←16 位有效地址	
	LDS r16,mem	DS:r16←32 位远指针	
	LES r16,mem	ES:r16←32 位远指针	

附表 5.2 指令汇编格式（续 1）

指令类型	指令汇编格式	指令功能简介	备 注
输入	IN AL/AX,i8/DX	AL/AX←I/O 端口 i8/DX	
输出	OUT i8/DX ,AL/AX	I/O 端口 i8/DX←AL/AX	
加法运算	ADD reg,imm/reg/mem	dest←dest+src	
	ADD mem,imm/reg		
	ADC reg,imm/reg/mem	dest←dest+src+CF	
	ADC mem,imm/reg		
	INC reg/mem	reg/mem←reg/mem+1	
减法运算	SUB reg,imm/reg/mem	dest←dest-src	
	SUB mem,imm/reg		
	SBB reg,imm/reg/mem	dest←dest-src-CF	
	SBB mem,imm/reg		
	DEC reg/mem	reg/mem←reg/mem-1	
	NEG reg/mem	reg/mem← $\overline{\text{reg / mem}}$+1	
	CMP reg,imm/reg/mem	dest-src	
	CMP mem,imm/reg		
乘法运算	MUL reg/mem	无符号数乘法	
	IMUL reg/mem	有符号数乘法	
除法运算	DIV reg/mem	无符号数除法	
	IDIV reg/mem	有符号数除法	
符号扩展	CBW	将 AL 符号扩展为 AX	
	CWD	将 AX 符号扩展为 DX.AX	
十进制调整	DAA	将 AL 中的加和调整为压缩 BCD 码	
	DAS	将 AL 中的减差调整为压缩 BCD 码	
	AAA	将 AL 中的加和调整为非压缩 BCD 码	
	AAS	将 AL 中的减差调整为非压缩 BCD 码	
	AAM	将 AX 中的乘积调整为非压缩 BCD 码	
	AAD	将 AX 中的非压缩 BCD 码转成二进制	
逻辑运算	AND reg,imm/reg/mem	dest←dest AND src	
	AND mem,imm/reg		
	OR reg,imm/reg/mem	dest←dest OR src	
	OR mem,imm/reg		
	XOR reg,imm/reg/mem	dest←dest XOR src	
	XOR mem,imm/reg		
	TEST reg,imm/reg/mem	dest AND src	
	TEST mem,imm/reg		
	NOT reg/mem	reg/mem← $\overline{\text{reg / mem}}$	
移位	SAL reg/mem,1/CL	算术左移 1 位/CL 指定的位数	
	SAR reg/mem,1/CL	算术右移 1 位/CL 指定的位数	
	SHL reg/mem,1/CL	逻辑左移 1 位/CL 指定的位数	
	SHR reg/mem,1/CL	逻辑右移 1 位/CL 指定的位数	
	ROL reg/mem,1/CL	循环左移 1 位/CL 指定的位数	

附表 5.2 指令汇编格式（续 2）

指令类型	指令汇编格式	指令功能简介	备 注
	ROR reg/mem,1/CL	循环右移 1 位/CL 指定的位数	
	RCL reg/mem,1/CL	带进位循环左移 1 位/CL 指定的位数	
	RCR reg/mem,1/CL	带进位循环右移 1 位/CL 指定的位数	
串操作	MOVS[B/W]	串传送	
	LODS[B/W]	串读取	
	STOS[B/W]	串存储	
	CMPS[B/W]	串比较	
	SCAS[B/W]	串扫描	
	REP	重复前缀	
	REPZ/REPE	相等重复前缀	
	REPNZ/REPNE	不等重复前缀	
控制转移	JMP label	无条件直接转移	
	JMP r16/m16	无条件间接转移	
	Jcc label 条件转移	/Z/NZ/S/NS/O/NO/B/NB/BE/NBE/L/NL/LE/NLE	cc 可为 C/NC
循环	LOOP label	CX←CX-1；若 CX≠0，则循环	
	LOOPZ/LOOPE label	CX←CX-1；若 CX≠0 且 ZF=1，则循环	
	LOOPNZ/LOOPNE label	CX←CX-1；若 CX≠0 且 ZF=0，则循环	
	JCXZ label	若 CX=0，则循环	
子程序	CALL label	直接调用	
	CALL r16/m16	间接调用	
	RET	无参数返回	
	RET i16	有参数返回	
中断	INT i8	中断调用	
	INTO	溢出中断调用	
	IRET	中断返回	
处理器控制	NOP	空操作指令	
	seg:	段跨越前缀	除 CS
	HLT	停机指令	
	LOCK	封锁前缀	
	WAIT	等待指令	
	ESC mem	换码指令	

附表 5.3 状态符号说明

符 号	说 明
-	标志位不受影响
0	标志位清 0
1	标志位置 1
x	标志位按定义功能设置
#	标志位按指令的特定说明设置
u	标志位不确定

附表 5.4 指令对状态标志的影响（未列出的指令不影响标志）

指 令	OF	SF	ZF	AF	PF	CF
SAHF	-	#	#	#	#	#
POPF/IRET	#	#	#	#	#	#
ADD/ADC/SUB/CMP/NEG/CMPS/SCAS	x	x	x	x	x	x
INC/DEC	x	x	x	x	x	-
MUL/IMUL	#	u	u	u	u	#
DIV/IDIV	u	u	u	u	u	u
DAA/DAS	u	x	x	x	x	x
AAA/AAS	u	u	u	x	u	x
AAM/AAD	u	x	x	u	x	u
AND/OR/XOR/TEST	0	x	x	u	x	0
SAL/SAR/SHL/SHR	#	x	x	u	x	#
ROL/ROR/RCL/RCR	#	-	-	-	-	#
CLC/STC/CMC	-	-	-	-	-	#

附录 6 IBM PC／AT 中断功能表

向 量 号	中 断 功 能 注 释
00H	除法错中断 其中 0～4 为 CPU 专用内部中断
01H	单步中断
02H	NMI 中断
03H	断点中断
04H	溢出中断
05H	屏幕打印
06H、 07H	保留
08H	定时器时钟中断（ IRQ_0） 08H～0FH 为外部可屏蔽中断
09H	键盘中断（ IRQ_1）
0AH	供 IRQ_8～IRQ_{15}串接（ IRQ_2） PC 和 PC/XT 机为保留
0BH	异步通信 COM2 中断（ IRQ_3）
0CH	异步通信 COM1 中断（ IRQ_4）
0DH	并行打印机 LPT2 中断（ IRQ_5） PC/XT 为硬盘中断
0EH	软盘中断（ IRQ_6）
0FH	并行打印机 LPT1 中断（ IRQ_7）
10H	显示 I/O 功能程序 10H～1FH 为 ROM BIOS 中断
11H	设备配置检测
12H	存储器容量检测
13H	磁盘 I/O 功能程序
14H	串行通信 I/O 功能程序
15H	盒式磁带机 I/O 功能程序 PC/XT 机不使用
16H	键盘 I/O 功能程序
17H	打印机 I/O 功能程序
18H	ROM BASIC 入口
19H	引导程序入口
1AH	日时钟 I/O 功能程序
1BH	键盘中止控制
1CH	定时器定时
1DH	显示器初始化参数
1EH	磁盘参数
1FH	图形字符集
20H	程序结束 20H～3FH 为 DOS 中断
21H	DOS 功能调用
22H	结束地址
23H	中止（ Ctrl+Break）处理
24H	关键性错误处理
25H	绝对磁盘读
26H	绝对磁盘写
27H	程序驻留结束
28H～3FH	DOS 内部使用或保留

向 量 号	中 断 功 能 注 释
40H 硬盘	I/O 功能程序 PC 机不使用
4lH	硬盘参数 PC 机不使用
42H～6FH	保留，用户使用或未使用
70H	实时时钟中断（ IRQ_0） 70H～77H 为外部可屏蔽中断
71H	软件使其重新指向 IRQ_2（ IRQ_9） PC 和 PC/XT 机无法使用
72H	保留（ IRQ_{10}）
73H	保留（ IRQ_{11}）
74H	保留（ IRQ_{12}）
75H	协处理机中断（ IRQ_{13}）
76H	硬盘中断（ IRQ_{14}）
77H	保留（ IRQ_{15}）
78H～FFH	未使用或保留给 BASIC 使用

附录 7 常用 DOS 功能调用（ INT 21H）

AH	功 能	入 口 参 数	出 口 参 数
00H	程序终止	CS=程序段前缀的段地址	
01H	键盘输入并回显		AL=输入字符
02H	显示输出	DL=输出字符	
03H	串行通信输入		AL=接收字符
04H	串行通信输出	DL=发送字符	
05H	打印机输出	DL=打印字符	
06H	直接控制台 I/O	DL=FFH（输入）， DL=字符（输出）	AL=输入字符
07H	键盘输入无回显		AL=输入字符
08H	键盘输入无回显 检测 Ctrl-Break 或 Ctrl-C		AL=输入字符
09H	显示字符串 DS:DX=字符串地址，字符串以 ‘ $’ 结尾		
0AH	键盘输入到缓冲区	DS:DX=缓冲区首址	(DS:DX)=缓冲区最大字符数 (DS:DX+1)=实际输入的字符数
0BH	检验键盘状态		AL=00H 有输入， AL=FFH 无输入
0CH	清输入缓冲区并执行指定的 输入功能	AL=输入功能号（ 1、 6、 7、 8、 0AH）	
0DH	磁盘复位		清除文件缓冲区
0EH	选择磁盘驱动器	DL=驱动器号（ 0=A， 1=B，…）	AL=系统中驱动器数
0FH	打开文件	DS:DX=FCB 首地址	AL=00H 文件找到， AL=FFH 文件未找到
10H	关闭文件	DS:DX=FCB 首地址	AL=00H 目录修改成功， AL=FFH 未找到
11H	查找第一个目录项	DS:DX=FCB 首地址	AL=00H 找到， AL=FFH 未找到
12H	查找下一个目录项	DS:DX=FCB 首地址，文件名中可带*或?	AL=00H 文件找到， AL=FFH 未找到
13H	删除文件	DS:DX=FCB 首地址	AL=00H 删除成功， AL=FFH 未找到
14H	顺序读文件	Ds:DX=FCB 首地址	AL=00H 读成功 AL=01H 文件结束，记录无数据 AL=02H DTA 空间不够 AL=03H 文件结束，记录不完整
15H	顺序写文件	DS:DX=FCB 首地址	AL=00H 写成功 AL=01H 磁盘满或只读文件 AL=02H DTA 空间不够
16H	创建文件	DS:DX=FCB 首地址	AL=00H 创建成功， AL=FFH 无磁盘空间
17H	文件改名	DS:DX=FCB 首地址 (DS:DX+1)=旧文件名 (DS:DX+17)=新文件名	AL=00H 改名成功， AL=FFH 不成功
19H	取当前磁盘驱动器		AL=当前驱动器号（ 0=A， 1=B，…）
1AH	设置 DTA 地址	DS:DX=DTA 地址	
1BH	取默认驱动器 FAT 信息		AL=每簇的扇区数 DS:BX=FAT 标识字符 CX=物理扇区的大小 DX=驱动器的簇数

AH	功 能	入 口 参 数	出 口 参 数
21H	随机读文件	DS:DX=FCB 首地址	AL=00H 读成功 AL=01H 文件结束 AL=02H 缓冲区溢出 AL=03H 缓冲区不满
22H	随机写文件	DS:DX=FCB 首地址	AL=00H 写成功 AL=01H 盘满 AL=02H 缓冲区溢出
23H	测定文件长度	DS:DX=FCB 首地址	AL=0 成功，文件长度填入 FCB AL=FFH 未找到
24H	设置随机记录号	DS:DX=FCB 首地址	
25H	设置中断向量	DS:DX=中断向量， AL=中断类型号	
26H	建立程序前缀 PSP	DX=新的 PSP	
27H	随机分块读	DS:DX=FCB 首地址 CX=记录数	AL=00H 读成功 AL=01H 文件结束 AL=02H 缓冲区溢出 AL=03H 缓冲区不满 CX=读取的记录数
28H	随机分块写	DS:DX=FCB 首地址 CX=记录数	AL=00H 写成功 AL=01H 盘满 AL=02H 缓冲区溢出
29H	分析文件名	ES:DI=FCB 首地址 DS:SI=ASCII 串 AL=控制分析标志	AL=00H 标准文件 AL=01H 多义文件 AL=FFH 非法盘符
2AH	取日期	CX:DH:DL=年:月:日	
2BH	设置日期	CX:DH:DL=年:月:日	AL=00H 成功， AL=FFH 无效
2CH	取时间		CH:CL=时:分， DH:DL=秒:百分秒
2DH	设置时间	CH:CL=时:分， DH:DL=秒:百分秒	AL=00H 成功， AL=FFH 无效
2EH	设置磁盘检验标志	AL=00 关闭检验， AL=01 打开校验	
2FH	取 DTA 地址		ES:BX=DTA 首地址
30H	取 DOS 版本号		AL=版本号， AH=发行号
31H	程序终止并驻留	AL=返回码， DX=驻留区大小	
33H	Ctrl-Break 检测	AL=00 取状态 AL=01 置状态（ DL=00H 关闭， DL=01H 打开）	DL=00H 关闭检测， DL=01H 打开检测
35H	获取中断向量	AL=中断类型号	ES:BX=中断向量
36H	取可用磁盘空间	DL=驱动器号 0=默认， 1=A， 2=B，…	成功： AX=每簇扇区数， BX=有效簇数， CX=每扇区字节数， DX=总簇数 失败： AX=FFFFH
38H	置/取国家信息	AL=00 取国别信息 AL=FFH 国别代码放在 BX 中 DS:DX=信息区首地址 DX=FFFFH 设置国别代码	BX=国别代码 DS:DX=返回的信息区首地址 AX=错误代码

AH	功 能	入 口 参 数	出 口 参 数
39H	建立子目录	DS:DX=ASCII 串地址	AX=错误码
3AH	删除子目录	DS:DX=ASCII 串地址	AX=错误码
3BH	改变当前目录	DS:DX=ASCII 串地址	AX=错误码
3CH	建立文件	DS:DX=ASCII 串地址， CX=文件属性	成功：AX=文件代号；失败：AX=错误码
3DH	打开文件	DS:DX=ASCll 串地址 AL=0/1/2 读/写/读写	成功：AX=文件代号；失败：AX=错误码
3EH	关闭文件	BX=文件号	失败： AX=错误码
3FH	读文件或设备	DS:DX=数据缓冲区地址 BX=文件号 CX=读取的字节数	成功： AX=实际读出的字节数， AX=0 已到文件尾 出错： AX=错误码
40H	写文件或设备	DS:DX=数据缓冲区地址， BX=文件号， CX=写入的字节数	成功： AX=实际写入的字节数 出错： AX=错误码
41H	删除文件	DX:DX=ASCII 串地址	成功： AX=00； 失败： AX=错误码
42H	移动文件指针	BX=文件号， CX:DX=位移量 AL=移动方式	成功： DX:AX=新指针位置 出错： AX=错误码
43H	读取/设置文件属性	DS:DX=ASCII 串地址 AL=0/1 取/置文件属性， CX=文件属性	成功： CX=文件属性 失败： AX=错误码
44H	设备 I/O 控制	BX=文件号； AL=0 取状态， AL=1 置状态， AL=2,4 读数据， AL=3,5 写数据， AL=6 取输入状态， AL=7 取输出状态	成功： DX=设备信息 失败： AX=错误码
45H	复制文件号	BX=文件号 1	成功： AX=文件号 2；出错： AX=错误码
46H	强制复制文件号	BX=文件号 1， CX=文件号 2	成功： AX=文件号 1；出错： AX=错误码
47H	取当前目录路径名	DL=驱动器号， DS:SI=ASCII 串地址	DS:SI=ASCII 串地址； 失败： AX=错误码
48H	分配内存空间	BX=申请内存容量	成功： AX=分配内存首址 失败： BX=最大可用空间
49H	释放内存空间	ES=内存起始段地址	失败： AX=错误码
4AH	调整已分配的内存空间	ES=原内存起始地址 BX=再申请的内存容量	失败： AX=错误码 BX=最大可用空间
4BH	装入/执行程序	DS:DX=ASCII 串地址 ES:BX=参数区首地址 AL=0/3 装入执行/装入不执行	失败： AX=错误码
4CH	带返回码终止	AL=返回码	
4DH	取返回码		AL=返回码
4EH	查找第一个匹配文件	DS:DX=ASCII 串地址，CX=属性	AX=错误码
4FH	查找下一个匹配文件	DS:DX=ASCII 串地址，文件名中可带*或?	AX=错误码
54H	读取磁盘写标志		AL=当前标志值
56H	文件改名	DS:DX=旧 ASCII 串地址 DS:DX=新 ASCII 串地址	AX=错误码
57H	设置/读取文件日期和时间	BX=文件号， AL=0 读取 AL=1 设置 (DX:CX)=日期:时间	DX:CX=日期:时间 失败： AX=错误码

附录 8 BIOS 功能调用

INT	AH	功能	调用参数	返回参数
10	0	设置显示方式	AL=00 40×25 黑白文本， 16 级灰度	
			=01 40×25 16 色文本	
			=02 80×25 黑白文本， 16 级灰度	
			=03 80×25 16 色文本	
			=04 320×200 4 色图形	
			=05 320×200 黑白图形， 4 级灰度	
			=06 640×200 黑白图形	
			=07 80×25 黑白文本	
			=08 160×200 16 色图形（ MCGA）	
			=09 320×200 16 色图形（ MCGA）	
			=OA 640×200 4 色图形（ MCGA）	
			=OD 320×200 16 色图形（ EGA／VGA）	
			=OE 640×200 16 色图形（ EGA／VGA）	
			=OF 640×350 单色图形（ EGA／VGA）	
			=10 640×350 16 色图形（ EGA／VGA）	
			=11 640×480 黑白图形（ VGA）	
			=12 640×480 16 色图形（ VGA）	
			=13 320×200 256 色图形（ VGA）	
10	1	置光标类型	$(CH)_{0\text{-}3}$=光标起始行， $(CL)_{0\text{-}3}$=光标结束行	
10	2	置光标位置	BH=页号， DH/DL=行/列	
10	3	读光标位置	BH=页号	CH=光标起始行
				CL=光标结束行
				DH/DL=行/列
10	4	读光笔位置		AX=0 光笔未触发
				=1 光笔触发
				CH/BX=像素行/列
				DH/DL=字符行/列
10	5	置当前显示页	AL=页号	
10	6	屏幕初始化或上卷	AL=0 初始化窗口	
			AL=上卷行数， BH=卷入行属性	
			CH/CL=左上角行/列号	
			DH/DL=右上角行/列	
10	7	屏幕初始化或下卷	AL=0 初始化窗口	
			AL=下卷行数， BH=卷入行属性	
			CH/CL=左上角行/列号	
			DH/DL=右上角行/列	
10	8	读光标位置的字符和属性	BH=显示页	AH/AL=字符/属性
10	9	在光标位置显示字符和属性	BH=显示页	
			AL/BL=字符/属性， CX=字符重复次数	
10	A	在光标位置显示字符	BH=显示页	
			AL=字符， CX=字符重复次数	

INT	AH	功能	调用参数	返回参数
10	B	置彩色调色板	BH=彩色调色板 ID BL=和 ID 配套使用的颜色	
10	C	写像素	AL=颜色值， BH=页号， DX/CX=像素行/列	
10	D	读像素	BX=页号， DX/CX=像素行/列	AL=像素的颜色值
10	E	显示字符（光标前移）	AL=字符， BH=页号， BL=前景色	
10	F	取当前显示方式		BH=页号， AH=字符列数 AL=显示方式
10	10	置调色板寄存器 (EGA/VGA)	AL=0， BL=调色板号， BH=颜色值	
10	11	装入字符发生器 (EGA/VGA)	AL=0～4 全部或部分装入字符点阵集 AL=20～24 置图形方式显示字符集 AL=30 读当前字符集信息	ES:BP=字符集位置
10	12	返回当前适配器设置的信息（ EGA/VGA）	BL=10H（子功能）	BH=0 单色方式 =1 彩色方式 BL=VRAM 容量 (0=64K， 1=128K，…) CH=特征位设置 CL=EGA 的开关设置
10	13	显示字符串	ES:BP=字符串地址 AL=写方式（ 0～3）， CX=字符串长度 DH/DL=起始行/列， BH/BL=页号/属性	
11		取设备信息		AX=返回值（位映像） 0=设备未安装 1=设备未安装
12		取内存容量		AX=字节数（ KB）
13	0	磁盘复位	DL=驱动器号 (00， 01 为软盘， 80h， 81h，…为硬盘)	失败： AH=错误码
13	1	读磁盘驱动器状态		AH=状态字节
13	2	读磁盘扇区	AL=扇区数 $(CL)_{6,7}(CH)_{0\sim7}$=磁道号 $(CL)_{0\sim5}$=扇区号 DH/DL=磁头号/驱动器号 ES:BX=数据缓冲区地址	读成功： AH=0， AL=读取的扇区数 读失败： AH=错误码
13	3	写磁盘扇区	同上	写成功： AH=0， AL=写入的扇区数 写失败： AH=错误码
13	4	检验磁盘扇区	AL=扇区数 $(CL)_{6,7}(CH)_{0\sim7}$=磁道号 $(CL)_{0\sim5}$=扇区号 DH/DL=磁头号/驱动器号	成功： AH=0 AL=检验的扇区数 失败： AH=错误码

INT	AH	功能	调用参数	返回参数
13	5	格式化盘磁道	AL=扇区数	
			$(CL)_{6,7}(CH)_{0\sim7}$=磁道号， $(CL)_{0\sim5}$=扇区号	成功： AH=0
			DH/DL=磁头号/驱动器号	失败： AH=错误码
			ES:BX=格式化参数表指针	
14	0	初始化串行口	AL=初始化参数	AH=通信口状态
			DX=串行口号	AL=调制解调器状态
14	1	向通信口写字符	AL=字符	写成功： $(AH)_7$=0
			DX=通信口号	写失败： $(AH)_7$=1
				$(AH)_{0\sim6}$=通信口状态
14	2	从通信口读字符	DX=通信口号	读成功： $(AH)_7$=0
				AL=字符
				读失败： $(AH)_7$=1
14	3	取通信口状态	DX=通信口号	AH=通信口状态
				AL=调制解调器状态
15	0	启动盒式磁带机		
15	1	停止盒式磁带机		
15	2	磁带分块读	ES:BX=数据传输区地址	AH=状态字节
			CX=字节数	=00 读成功
				=01 冗余检验错
				=02 无数据传输
				=04 无引导
				=80 非法命令
15	3	磁带分块读	DS:BX=数据传输区地址	AH=状态字节
			CX=字节数	（同上）
16	0	从键盘读字符		AL=字符码
				AH=扫描码
16	1	取键盘缓冲区状态		ZF=0
				AL=字符码， AH=扫描码
				ZF=1 键盘缓冲区空
16	2	取键盘状态字节		AL=键盘状态字节
17	1	初始化打印机	DX=打印机号	AH=打印机状态字节
		回送状态字节		
17	2	取打印机状态字节	DX=打印机号	AH=打印机状态字节
1A	0	读时钟		CH:CL=时:分
				DH:DL=秒:1/100 秒
1A	1	置时钟	CH:CL=时:分	
			DH:DL=秒:1/100 秒	
1A	6	置报警时间	CH:CL=时:分（ BCD）	
			DH:DL=秒:1/100 秒（ BCD）	
1A	7	清除报警		