第 2 章　汇编语言实验程序的建立与执行

学习微型计算机原理必须接触汇编语言，因此要学习汇编语言程序的设计、调试和运行，并进行相应的实验。本章讨论汇编语言实验程序的建立与执行。

2.1　编辑和运行汇编源程序所必备的软件

在 PC 上运行汇编源程序，必须具备以下软件：

- DOS 2.0（或 CCDOS）以上版本操作系统
- EDIT 文本编辑程序（或其他文本编辑软件）
- MASM 宏汇编程序
- LINK 连接程序
- CREF 交叉引用文件处理程序（可选）
- LIB 库管理程序（可选）
- Debug 调试程序
- QBasic 编辑程序

2.2　建立与执行汇编源程序

在 TPC-H 微机接口实验系统中进行各项接口实验，一般都要用汇编语言（或 C 语言）来编写相应的实验程序。下面简要介绍汇编语言程序从建立到执行的过程。

2.2.1　建立与执行汇编源程序的基本步骤

用汇编语言编写的程序（即汇编源程序）要在机器上运行，必须经过以下几个步骤：

（1）在 DOS 环境下，调用任一编辑程序（如 EDIT）建立与修改源程序（扩展名必须为.ASM）。

（2）用宏汇编程序（MASM）对汇编源程序进行汇编，生成相应的目标文件（扩展名为.OBJ）。

（3）用连接程序（LINK）对目标文件进行再定位和连接，生成可执行文件（扩展名为.EXE）。

（4）运行可执行文件。

经过步骤（1）～（3）之后，可执行文件已生成并存放在磁盘上。此时，在 DOS 提示符下直接输入文件名（可不带扩展名.EXE），即可将该文件从磁盘上装入内存立即执行。

然而，设计一个较复杂的汇编源程序，不出现一点错误是不太可能的。如果出现错误，可调用 DOS 支持下的 Debug 程序（调试程序），对可执行文件进行动态跟踪调试运行，找出错误的地方。当发现错误后，要重复上述（1）～（4）步，即修改源程序中的错误，重新汇编、连接、运行程序，直至程序运行正确（或符合设计要求）为止。

2.2.2 建立与执行汇编源程序的过程细节

1. 编辑源程序

建立汇编语言源程序是程序开发的一步。可以使用的文本编辑器有 DOS 下的 EDIT、Windows 下的记事本，或者是 MASM 自带的 PWB（Programmer WorkBench），但不能用 Word 或写字板。下面以 DOS 下的 EDIT 为例来说明。

启动计算机，打开 C 盘，查看其中是否有 MASM 文件夹。然后从屏幕上选择“开始”→“运行”命令，在弹出的对话框中输入 cmd，进入到 DOS 状态的命令行模式，输入命令“cd\”、“cd masm”、“cd bin”、“masm”，屏幕显示 masm 的版本信息以及编译方法。

在编译中，输入源程序的文件名时，如果源程序就在当前目录下，则只输入文件名即可；如果启动 EDIT 时该文件已存在，源程序在其他目录下，则一定要输入源程序的路径，指明其盘符和子目录，将该文件调入编辑器内。

在当前目录下，可通过键盘输入自己编写的源程序并存盘，得到扩展名为.ASM 的汇编源程序文件。

再按 2.2.1 节所述的步骤对文件进行汇编、连接、运行。

2. 用 EDIT 编辑汇编源程序

命令格式为：

C:\MASM>EDIT[文件名]

其中，文件名为可选项。例如：

C:\MASM>EDIT

如果要带文件名，则必须是带有.ASM 扩展名的文件名全称。例如：

C:\MASM>EDIT Display.ASM

若输入过程有错误或对源程序进行汇编、连接、运行的过程中发现错误，都可以利用 EDIT 的一些命令对输入的源程序文件进行修改，修改完成后，再选择 File→Save As 命令保存。编辑完成后，选择 File→Exit 命令退出 EDIT，返回 DOS 系统。

3. 用 MASM 对源程序进行汇编

源程序建立后，可以用 Microsoft 宏汇编程序（MASM.EXE）对它进行汇编。通过汇编生成可重定位的二进制代码目标文件。

汇编过程中汇编程序一般采用两遍扫描的方法。第一遍扫描源程序产生符号表、处理伪指令等；第二遍扫描产生机器指令代码、确定数据等。如果源程序中有语法错误，则汇编结束时汇编程序将指出源程序中的错误。编程者根据出错提示信息修改源程序，直到通过为止。应该指出的是，汇编正确通过，只能说明源程序没有语法错误，但不能说明算法上或其他方面没有问题。

汇编完成后便建立扩展名为.OBJ 的目标文件，并且根据选择也可同时建立.LST 列表文件和.CRF 交叉索引文件。

对源程序进行汇编的过程有以下两种形式：

（1）提问方式。

在 DOS 状态下输入 MASM 命令，调入宏汇编程序之后即在屏幕上显示以下信息：

C:\MASM>MASM

Microsoft ® Macro Assembler Version 5.00

Copyright © Microsoft Corp 1981-1985.1987 All rights reserved.

```
Source lilenamc[.ASM]:Display          ;输入源程序文件名 Display.ASM
Object filename [Display.OBJ]:         ;汇编后得到目标文件名 Display.OBJ
Source Listing [NUL.LST]:Display,      ;输出列表文件（可选）
Cross-frcnce [NUL.CRF]:Display         ;输出交叉索引文件（可选）
0 Warmng errors                        ;警告性错误数目
0 Severe errors                        ;严重错误数目
```

汇编程序运行后首先显示版本号，然后依次有 4 个提示。其中，第一个提示询问要汇编的源程序文件名。用户根据提示输入源文件名 Display（可不带.ASM 扩展名）后出现第二个提示，询问目标程序文件名，括号内的信息为系统规定的默认文件名（Display.OBJ），通常直接按 Enter 键，表示采用默认文件名。接着出现第三个提示，询问是否要建立列表文件（默认为空），若要则输入文件名，若不要则直接按 Enter 键。最后发出第四个提示，询问是否要建立交叉索引文件（默认为空），若要则输入文件名；若不要则直接按 Enter 键。在回答了第四个询问之后，汇编程序便对汇编源程序进行汇编。若汇编过程中发现源程序中有语法错误，则给出错误的行号和错误信息的提示，最后列出警告错误数及严重错误数。此时，应分析错误，然后再调用 EDIT 加以修改，改正后重新汇编，直至汇编后无错误为止。

（2）命令行方式。

命令行格式为：

```
C:\MASM>MASM 源文件名,目标文件名,列表文件名,交叉索引文件名;
```

格式中扩展名都可不给出，汇编程序会按照默认情况使用或产生。若只想对部分提示给出回答，则在相应位置用逗号隔开；若不想对剩余部分回答，则用分号结束。例如，以下的命令与前面的分行回答是等效的：

```
C:\MASM>MASM Display, ,Display,Display;
```

该命令行表示调用 MASM 程序对源文件 Display.ASM 进行汇编，生成目标文件 Display.OBJ、列表文件 Display.LST 和交叉索引文件 Display.CRF。

再如：

```
C:\MASM>MASM Display;（后面各项均不列出）
```

该命令行表示调用 MASM 程序对源文件 Display.ASM 进行汇编，仅生成目标文件 Display.OBJ。

4. 执行连接程序（LINK）

用宏汇编程序 MASM 对源程序进行汇编后产生的二进制目标文件（.OBJ 文件）还不能直接在 PC 上运行，因为在该目标文件中有可浮动的相对地址（逻辑地址），必须经过连接程序（LINK）连接后才能执行。连接程序（LINK）的功能是把一个或多个独立的目标程序模块装配成一个可重定位的可执行文件，即扩展名为.EXE 的文件。此外，还可产生一个内存映像文件，扩展名为.MAP。

执行连接程序 LINK 的过程与执行汇编程序 MASM 类似，也有以下两种方式：

（1）提问方式。

在 DOS 状态下输入 LINK 回车（或 LINK Display 回车），调入连接程序 LINK，在屏幕上显示版本信息后，依次提出 4 个问题。以 Display.OBJ 为被连接的目标文件为例，显示如下：

```
C:\MASM>LINK
Microsoft  ® Overlay Linker Version 3.60
Copyright © Microsoft Corp 1 983 • 1 987 All rights reserved
Object Modules [.OBJ]:Display          ;输入目标文件 Display.OBJ
```

```
Run File[DISPLAY.EXE]:          ;建立 Display.EXE 文件
List File[NUL.MAP]:Display      ;建立 Display.MAP 文件
Libraries [.LIB]                ;连接其他程序库文件
```

LINK 程序运行后首先询问要连接的目标文件名，操作员输入文件名作为回答（此处为 Display），如果有多个要连接的目标文件，应一次输入，各目标文件名之间用“+”号隔开，最后按 Enter 键。第二个提示，询问要产生的可执行文件的文件名，一般直接按 Enter 键表示采用方括号内规定的默认文件名。第三个提示，询问是否要建立内存地址映像文件，输入文件名再按 Enter 键表示要建立；直接按 Enter 键表示不要建立。最后询问是否用到库文件，如果没有库文件，则直接按 Enter 键即可（如果用户自己建立了库文件，则输入库文件名）。

回答以上询问后，LINK 便开始进行连接。若连接过程有错，则显示错误信息。此时，应根据错误性质重新调用 EDIT 编辑程序修改源程序，重新汇编、连接，直至无错误为止。

（2）命令行方式。

命令行格式为：

```
C:\MASM>LINK 目标文件名,执行文件名,内存映像文件名,库文件名;
```

格式中扩展名都可以不给出，LINK 程序会按默认情况使用和产生。若只想对部分提示给出回答，则在相应位置用逗号隔开：若不想对剩余部分作答，则用分号结束。例如，下列命令行与前边的分行回答操作是等效的：

```
C:\MASM>LINK Display, ,Display;
```

该命令行表示执行 LINK 程序对目标程序文件 Display.OBJ 进行连接，生成可执行文件 Display.EXE（默认值）和内存映像文件 Display.MAP。

再如：

```
C:\MASM>L1NK Display;
```

该命令行表示执行 LINK 程序对 Display.OBJ 文件进行连接，仅生成一个可执行文件 Display.EXE。如果除 Display.EXE 文件外，还要产生 Display.MAP 文件，则在分号前加两个逗号，如下：

```
C:\.MASM>LINK Display, , ;
```

运行 LINK 程序后，即产生两个输出文件：一个是可执行文件 Display.EXE；另一个是扩展名为 .MAP 的内存映像文件，它指出每个段在内存中的地址分配情况及长度。通常，在 DOS 状态下用 TYPE 命令显示打印出来。例如：

```
C:\MASM>TYPE Display.MAP
 Start         Stop        Length      Name
00000H     0000FH      0010H       DATA
00010H     0004FH      0040H       STACK
00050H     0005FH      0010H       CODE
Origin   Group
Program entry point at 0005:0000
```

从内存映像文件可知，源程序 Display 中定义了 3 个段：数据段（DATA），起始地址为 00000H，终止地址为 0000FH，长度为 0010H 字节；堆栈段（STACK），起始地址为 00010H，终止地址为 0004FH，长度为 0040H 字节；代码段（CODE），起始地址为 00050H，终止地址为 0005FH，长度为 0010H 字节。

5. 运行程序

当用连接程序 LINK 将目标程序文件（.OBJ）连接定位成功后，在磁盘上就已生成一个可执行文件（.EXE）。此时，即可在 DOS 状态下运行该文件。其执行操作如下：

C:\MASM>Display 或 C:\MASM>Display.EXE

在源程序 Display 中如果有显示结果的指令，则执行程序后可以在屏幕上或打印机上看到执行结果；如果没有显示结果的命令，要想看到结果，则只有通过 Debug 程序来查看了。

2.3　调试程序 Debug 及其使用

在上述编写和运行汇编源程序的过程中，会遇到一些错误和问题，需要对源程序进行分析和调试。通常采用 Debug 调试程序。Debug 是专为汇编语言设计的交互式机器语言程序的调试工具。它有较强的功能，能使程序设计者接触到机器内部，可以单步执行程序，也可以在程序中设置断点，能观察和修改寄存器与内存单元的内容，并能监视目标程序的执行情况，便于寻找程序的错误。

2.3.1　Debug 程序的调用

以 Windows 为操作系统的 PC 都带有 Debug 调试程序。当 PC 进入 MS-DOS 工作方式后，就可以调用 Debug 调试程序了。

1. 调用格式

C:\MASM>DEBUG[d:][Path][filename.ext]

其中，[]的内容为可选项；[d:]为驱动器号，指要调入 Debug 状态下的可执行文件所在的驱动器；[Path]为路径，指要调入 Debug 状态下的可执行文件所在的目录或子目录；[filename.ext]指要调入 Debug 状态下的可执行文件的文件名，该文件名的扩展名只能是.EXE 或.COM，即 Debug 只能对扩展名为.EXE 和.COM 的文件进行调试。

在启动 Debug 时，如果输入文件名，则 Debug 程序就把指定文件装入内存，用户可以通过 Debug 的命令对指定文件进行调试。如果没有带文件名，则可以当前内存的内容工作，或者用命名命令（Name）或装入命令（Load）把需要的文件装入内存，然后再通过 Debug 命令进行调试。

当启动 Debug 程序成功后，屏幕上出现“－”提示符，说明系统已进入 Debug 状态。

2. Debug 程序对寄存器和标志位的初始化

在调入 Debug 程序后，各寄存器和标志位置成为以下状态：

- 段寄存器（CS、DS、ES、SS）置于自由存储空间的底部，也就是 Debug 程序结束后的第一个段。
- 指令指针（IP）置为 0100H。
- 堆栈指针置于段的结尾处或是装入程序的临时底部。哪一个更低，SP 就指向哪一个。
- 余下的寄存器（AX、BX、CX、DX、BP、SI 和 DI）均置为 0。但是，若调用 Debug 时包含一个要调试的可执行文件名，则 CX 中包含以字节表示的文件长度，若文件大于 64KB，则文件长度包含在 BX 和 CX 中（高位部分在 BX 中）。
- 标志位都置为清除状态。
- 默认磁盘的传送地址置成代码段的 80H。

注意：若由 Debug 调入的程序具有扩展名为.EXE 的文件，则 Debug 必须进行再分配，把段寄存器、堆栈指针置为程序中所规定的值。

2.3.2 Debug 命令的有关规定

（1）Debug 命令都是由一个英文字母，后面跟着一个或多个有关参数组成的。

（2）命令和参数可以用大写或小写或混合方式输入。

（3）命令和参数之间以及参数与参数之间可以用定界符（逗号或空格）分隔。

下列命令是等效的：

dcs:100 110

d cs:100 110

d.cs:100.110

（4）每一个命令，只有按了 Enter 键以后才有效。

（5）参数中的地址和数据均用十六进制数表示，但十六进制数据后面不跟“H”标志。

（6）可以用 Ctrl 和 Break 键来停止一个命令的执行，返回到 Debug 的提示符“-”下。

（7）若一个命令产生相当多的输出行，则为了能在屏幕上看清输出内容，可在显示过程中用 Ctrl+NumLock 键暂停上卷动作，也可通过按任意键来继续上卷动作。

2.3.3 Debug 的主要命令

1. 汇编命令 A

格式：A[address]

格式中的[address]可以是[段寄存器名]:[偏移地址]，也可以是[段地址]:[偏移地址]，或者只有[偏移地址]，或者缺省。

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单元开始的存储区中。若在命令中没有指定地址，但前面用过汇编命令，则接着上一个汇编命令汇编结果存放的最后一个单元开始存放。若前面未用过汇编命令，则从“CS:100”单元开始存放。若输入的语句中有错误，则 Debug 会显示“Error”信息，然后重新显示现行的汇编地址，等待新的输入。

注意：Debug 把输入的数字看成十六进制数，如果要输入十进制数，则应说明，如 100D。

2. 显示内存单元内容的命令 D

格式：D[address]或 D[range]

格式中 address 为地址，range 为地址范围。

该命令能显示指定范围的内存单元的内容。显示的内容为两种形式：一种为十六进制内容（每一字节用两位十六进制数显示）；另一种是用相应的 ASCII 码字符显示，句号“。”表示不可显示的字符。

3. 修改内存单元内容的命令 E

有两种格式：

E[address][list]

E[address]

格式中 address 为地址，1ist 为内容列表。

第一种格式可以用给定的内容表来代替指定存储单元的内容，如：

-EDS:1 00　F3 'XYZ'8D

其中 F3、'X'、'Y'、'Z'和 8D 各占一个字节，该命令可以用这 5 个字节来代替内存单元 DS:0100～0104 的原先内容。内容表中的内容可以是一个字节的十六进制数，也可以是用单引号括起来的一串字符。

第二种格式则是采用逐个单元相继修改内存单元内容的方法，如：

-E CS:100

则可以显示为：

1 8E4:0100 89

如果想把该单元的内容修改为 78，则可直接输入 78，再按空格键，可接着显示下一单元的内容如下：

18E4:0100　89 78 1B

这样，可不断修改相继内存单元的内容，直至按 Enter 键结束该命令为止。

4. 检查和修改寄存器内容的命令 IR

它有 3 种格式：

R

R[register name]

RF

格式中的 register name 为寄存器名。

第一种格式是显示 CPU 内所有寄存器的内容和标志位状态，如：

-r

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

DS=18E4ES=18E4 SS=18E4CS=18E4IP=0100 NV UP DI PL NZ NA PO NC

18E4:0100C70604023801　MOV WORD PTR[024],0138 DS:0204=0000

第二种格式是显示和修改某个寄存器的内容，如：

-r ax

则系统显示如下：

AX　F1F4

即 AX 寄存器中当前内容为 F1F4，如不修改，则按 Enter 键；否则，可输入欲修改的内容，如：

-r bx

BX 0369

:059F

则把 BX 寄存器的内容修改为 059F。

第三种格式是显示和修改标志位状态，如：

- r F

则系统显示为：

OV　DN　El　NG　ZR　AC　PE　CY

此时，如不修改其内容，则可按 Enter 键；否则，可输入欲修改的内容。

注意：对于状态标志寄存器 FLAG 是以位的形式显示的，8 个状态标志的显示次序和符号如表 2-1 所示。

表 2-1 状态标志寄存器 8 个标志的显示次序和符号

标志位	状态	显示形式（置位/复位）
溢出标志 OF	有/无	OV/NV
方向标志 DF	增/减	DN/UP
中断标志 IF	开/关	EI/DI
符号标志 SF	负/正	NG/PL
零标志 ZF	零/非零	ZR/NZ
辅助进位 AF	有/无	AC/NA
奇偶标志 PF	偶/奇	PE/PO
进位标志 CF	有/无	CY/NC

5. 跟踪与显示命令 T

有两种格式：

T[=address] ;逐条指令跟踪

T[=address][value] ;多条指令跟踪

第一种格式是执行一条指定地址处的指令后停下来，显示 CPU 所有寄存器的内容和全部标志位的状态。如未指定地址，则从当前 CS:IP 开始执行。

第二种格式是从指定地址起，执行 n 条指令后停下来，n 由 value 指定。

6. 反汇编命令 U

有两种格式：

U[address]

U[range]

第一种格式是从指定地址开始，反汇编 32 个字节。如果命令中没有指定地址，则从上一个 U 命令的最后一条指令的下一单元开始显示 32 个字节。

第二种格式是对指定范围的存储单元的内容进行反汇编。范围可以由起始地址、结束地址来规定，也可以由起始地址与长度来规定。

7. 运行命令 G

格式：G[=addressl][address2[address3…]]

其中，address1 指定了运行的起始地址，如不指定，则从当前 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。

8. 退出 Debug 命令 Q

格式：Q

程序调试完后退出 Debug，返回到 DOS 状态下。

2.4 汇编语言程序设计实验

2.4.1 Debug 调试汇编语言程序的方法

1. 实验目的

（1）熟悉汇编语言在 Debug 模式下编辑、汇编、调试、生成的全过程。

（2）掌握部分常用的 Debug 命令，对 Debug 调试环境有一个直观的认识。

2. 实验原理及相关知识

参看 2.3 节调试程序 Debug 及其使用。

3. 实验内容

在 Debug 中调试以下两段代码：

```
（1）MOV     AX,FEDC
     MOV     BX,CDAE
     ADD     AX,BX
     SUB     AX,BX
（2）MOV     AH,01
     INT     21
     Sub     AL,20
     MOV     DL,AL
     MOV     AH,02
     INT     21
```

4. 实验步骤

第一段的代码调试过程如下：

（1）选择“开始”→“运行”命令，在弹出的对话框中输入 CMD 命令，进入黑屏的 DOS 状态，屏上显示光标和命令行模式。

（2）在命令行窗口中输入 Debug 回车。

（3）进入 Debug，可以看到 Debug 模式的标识和命令行模式是不一样的。

（4）这时在 Debug 中输入命令：A100，进入编辑状态，如图 2-1 所示。

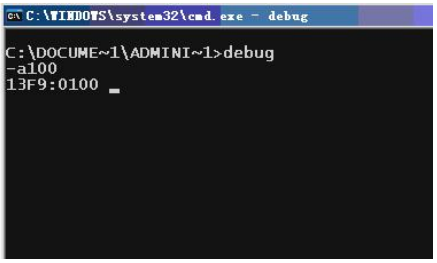

图 2-1　Debug 编辑状态

（5）这时可以看到在窗口中显示出 13F9:0100 这样一个提示，这个 13F9 代表的是段地址，而 0100 是偏移地址，也就是说指令将从这个地址开始存放（不同的计算机在不同的时间分配的段地址可能会有些不同）。

（6）输入第一段代码，输入完后按两次 Enter 键结束。接着输入命令：t=100 4，开始调试，如图 2-2 所示。

（7）注意观察 AX 和 BX 以及 PSW 寄存器的值变化。

第二段代码调试过程如下：

（1）按照调试第一段代码的过程，使用命令 a100 输入第二段代码。

（2）由于第二段代码中调用了中断不能直接调试，需要在 Debug 中编译存盘成.COM 的文件格式运行，下面介绍这种调试方式。

（3）输入完代码后输入以下命令：

R CX
0c
N demo.com（demo.com 是编译后存盘的文件名，可以自己随意定）
W
Q

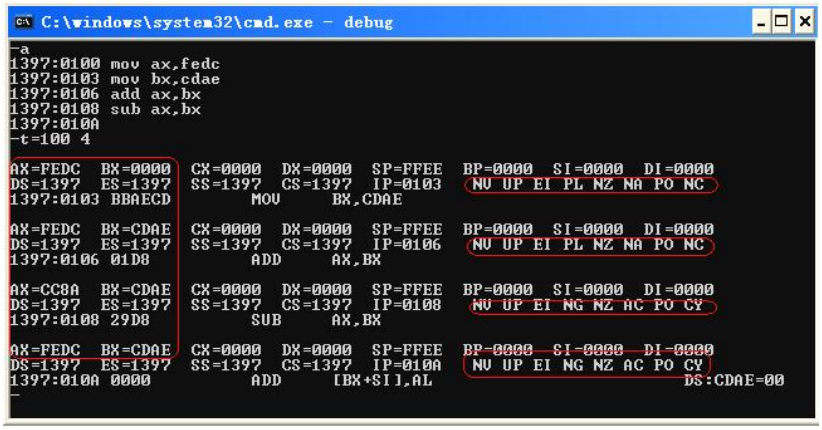

图 2-2　调试界面

当显示 Writing 0000C bytes 时表示存盘成功，如图 2-3 所示。

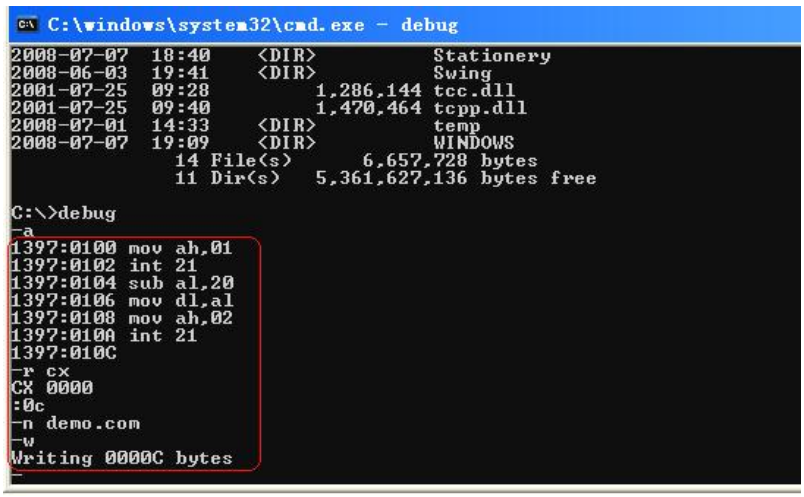

图 2-3　存盘成功界面

（4）退出 Debug 后，在 DOS 状态下调用该程序，如图 2-4 所示。

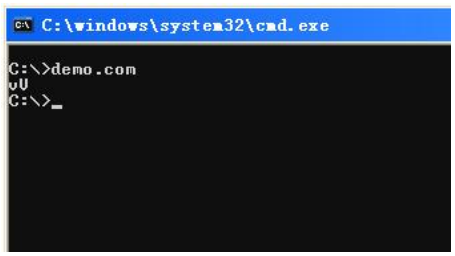

图 2-4　DOS 状态

输入一个小写 v，输出大写 V，实现了大小写转换。

5. 实验记录

记录程序调试过程出现的问题及解决过程，注意观察 AX、BX 及 PSW 等寄存器值的变化，并加以简要说明。

2.4.2　传送指令

1. 实验目的

（1）熟悉 8086 传送指令的格式。

（2）熟悉 MOV、POP、PUSH、XLAT 等指令的使用方法。

（3）熟悉 MASM 编译过程。

2. 实验原理及相关知识

温习 8086 传送指令。

3. 实验内容

在 Masm 中调试以下代码：

```
DATA    SEGMENT
        WVAR    DW    ?,?
        XVAR    DW    5286H
DATA ENDS
CODE SEGMENT
        ASSUME    DS:data,CS:code
START:
        MOV     AX,DATA
        MOV     DS,AX
        MOV     AX,5286H
        MOV     AX,XVAR
        MOV     BX,OFFSET XVAR
        MOV     AX,[BX]
        MOV     BX,OFFSET WVAR
        MOV     AX,4[BX]
        MOV     BX,OFFSET WVAR
        MOV     SI,4
        MOV     AX,[BX+SI]
        MOV     BX,2
        MOV     SI,2
        MOV     AX,WVAR[BX+SI]
        PUSH    AX
        POP     BX
        MOV     AH,4CH
        INT     21H
CODE ENDS
END START
```

4. 实验步骤

（1）首先在记事本中输入代码，然后以.ASM 格式保存至 C 盘的 Masm 中。

（2）确认计算机上已安装 Masm 编译器，然后使用汇编语言编译程序，编译源代码产生

OBJ 文件，如图 2-5 所示。

C:\masm>masm demo.asm

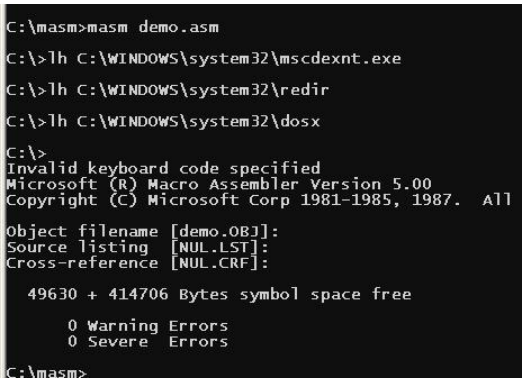

图 2-5　Masm 编译器环境

（3）使用 LINK 连接.OBJ 文件产生.EXE 文件，如图 2-6 所示。

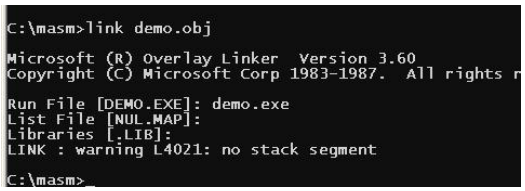

图 2-6　产生.EXE 文件

（4）调试。由于这个程序没有输出，所以要在 Debug 中进行调试。确定在 C:\masm>目录下输入以下命令：

Debug demo.exe

进入 Debug 后输入以下命令，观察结果（如图 2-7 所示）：

L

U

G

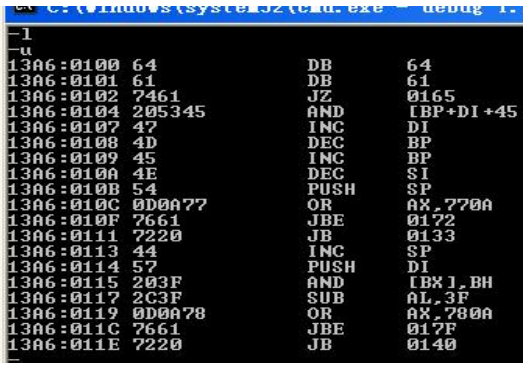

图 2-7　命令运行结果

5. 实验记录

记录程序调试过程中出现的问题及解决过程，注意观察 AX、BX 及 PSW 寄存器值的变化，

并加以简要说明。

2.4.3　逻辑与移位指令

1. 实验目的

（1）掌握 8086 逻辑指令的使用，注意区分汇编语言中逻辑指令与其他指令的不同，理解并掌握按位逻辑的概念。

（2）掌握移位指令的使用，了解不同的操作数之间不同的移位方式。

2. 实验原理及相关知识

温习 8086 逻辑指令和移位指令。

3. 实验内容

（1）假定 AL 中值为 78H，将 AL 中低 4 位与高 4 位交换。

（2）假定 AL 中值为 90H，将 AL 中的 D0 位和 D7 位交换，D3 位和 D4 位交换。

4. 实验步骤

实验内容一：

（1）编程思路分析：用自己的语言加以说明。

分析示例：将 AL 中的值进行高 4 位与低 4 位交换，可以使用循环移位指令。

（2）编写程序。

```
MOV   AL,78H
MOV   CL,4
ROL   AL,CL
```

实验内容二：

（1）编程思路分析：用自己的语言加以说明。

分析示例：通过分析可以知道如果一次实现 D0 和 D7 位交换以及 D3 和 D4 位交换是很困难的，可以采取先将 AL 的值分别赋值到 BL 和 DL 中去，然后对 BL 中的值除 D0 位和 D7 位外其他值都屏蔽为 0，进行 D0 位与 D7 位交换，同样对 DL 也进行上述方式处理。将处理完后的 DL 和 BL 的值叠加至 AL 中。

（2）编写程序。

```
MOV     AL,90H
MOV     BL, AL
MOV     DL, AL
AND     BL, 80H
AND     DL, 01H
ROL     BL, 1
ROR     DL, 1
OR      BL, DL
MOV     DL, AL
AND     DL, 10H
SHR     DL, 1
OR      BL, DL
MOV     DL,AL
AND     DL,08H
SHL     DL,1
```

```
OR      BL,DL
MOV     AL,BL
```

（3）调试。

① 打开 EMU8086 模拟器，在编辑区域中输入实验内容一的代码。

② 确认代码输入无误后，单击工具栏上的 emulate 按钮编译执行，如图 2-8 所示。

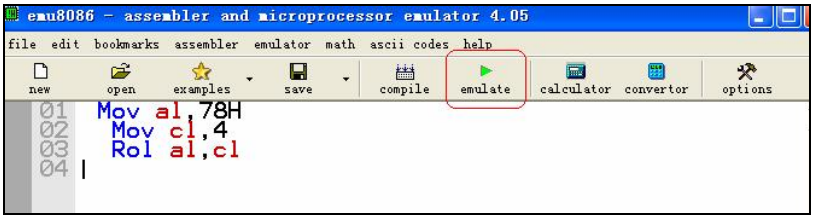

图 2-8　单击 emulate 按钮编译执行

③ 在弹出的 emulate 窗口中，单击 run 按钮，开始执行代码。观察窗口左侧的寄存器栏，从中可以看到 8086 所有的寄存器在执行程序时值的变化，如图 2-9 所示。

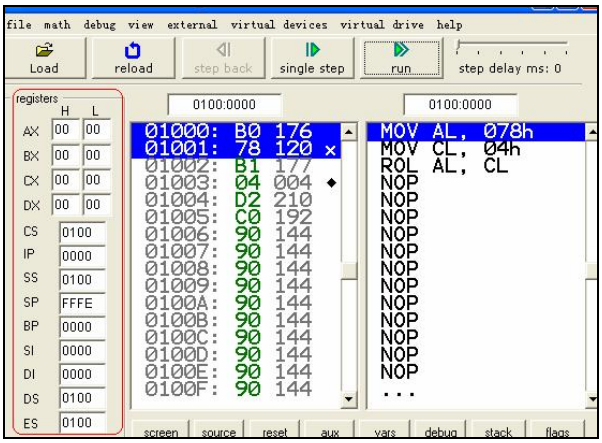

图 2-9　寄存器的值在执行程序时有所变化

（4）测试运行结果，如图 2-10 所示。

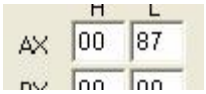

图 2-10　测试运行结果

按照调试实验内容一的代码的步骤方法调试实验内容二的代码并测试运行结果，如图 2-11 所示。

5. 实验记录

记录程序调试过程中出现的问题及解决过程，并加以简要说明。

2.4.4　子程序调用——字符串处理程序设计

1. 实验目的

（1）练习用汇编语言实现对字符串复制、比较、求长度及大小写转换的方法。

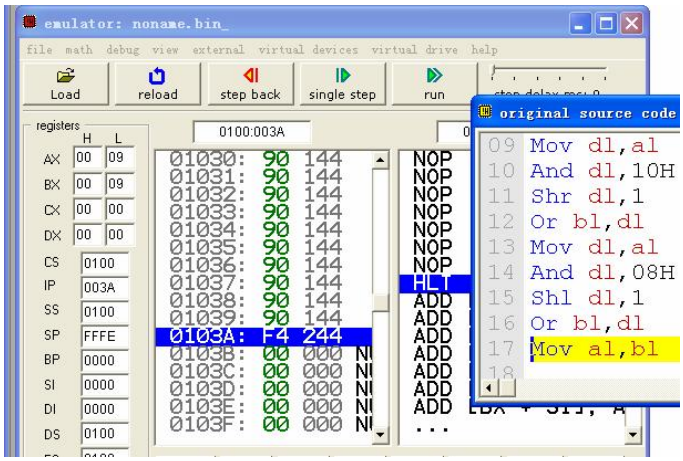

图 2-11　调试代码并测试运行结果

（2）掌握字符串处理方式以及子程序调用的使用。

2. 实验原理及相关知识

温习 8086 字符串处理指令，以及子程序调用的知识。

3. 实验内容

编程实现“从键盘输入一字符串，求出字符串的长度”的功能。

4. 实验步骤

（1）编程思路分析：用自己的语言加以说明。

分析示例：从键盘输入一字符串可以通过调用中断 01 号功能，首先实现输入单个字符，然后利用 STOB 串存储指令和循环实现输入多个字符，也就是输入字符串。当用户按 Enter 键时，输入结束。

（2）编写程序。

```
DATA    SEGMENT
            STR   DB   100   DUP(?)
            COUNT   DB   ?
DATA ENDS
CODE SEGMENT
            ASSUME   CS:CODE,DS:DATA
START:      MOV     AX,DATA
            MOV     DS,AX
            MOV     ES,AX
            LEA     DI,STR
            CALL    INPUT              ;调用输入子程序
            LEA     SI,STR
            CALL    COUNTPRC           ;调用字符串统计子程序
            MOV     AH,4CH
            INT     21H
            COUNTPRC   PROC            ;统计字符串长度子程序
            CLD
            MOV     CL,0
```

```
LOP2:   LODSB
        CMP     AL,0DH
        JZ      QUIT
        INC     CL
        JMP     LOP2
QUIT:   MOV     COUNT,AL
        RET
        COUNTPRC    ENDP
        INPUT   PROC            ;输入子程序
        CLD
LOP1:   MOV     AH,01
        INT     21H
        STOSB
        CMP     AL,0DH
        JZ      ENDSTR
        JMP     LOP1
ENDSTR: RET
        INPUT   ENDP
        CODE    ENDS
        END     START
```

（3）程序调试。打开 EMU8086 输入以上代码，测试相应的结果，如图 2-12 所示。

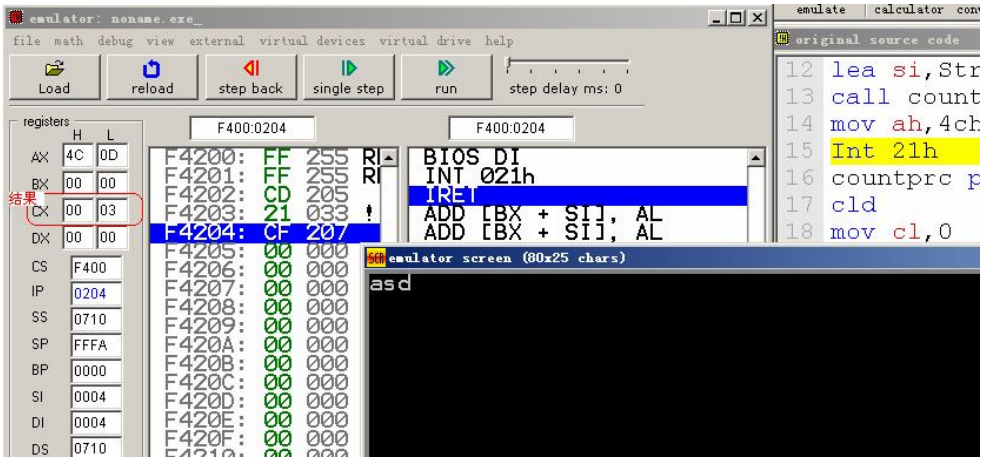

图 2-12　输入代码测试相应结果

5. 实验记录

记录程序调试过程中出现的问题及解决过程，并加以简要说明。

2.4.5　中断处理程序开发

1. 实验目的

（1）掌握中断调用原理、中断向量表的概念及作用。

（2）了解中断服务程序的开发流程及技术要领。

2. 实验原理及相关知识

温习中断调用和中断向量表的相关知识。

3. 实验内容

利用 DOS 中断中的 25H 功能和 35H 功能修改 8086 CPU 的 65H 中断，功能变为在屏幕上显示一字符串“This is New Int 65H No.”

4. 实验步骤

（1）编程思路分析：用自己的语言加以说明。

分析示例：中断调用程序大部分都是由 BIOS 和 DOS 系统提供的，当用户编写自己的中断处理程序时，需要考虑到使用的中断号码是否已经被使用；由于中断涉及对硬件底层的直接操作，难度较高，而且从 Windows 2000 开始系统采用纯 32 位模式，DOS 模式都由 32 系统来模拟出来，所以修改中断难以实现，程序也难以通过编译。利用 EMU 8086 模拟器可以通过模拟的方式进行编译，实现修改中断的效果。

（2）编写程序。

```
CODE    SEGMENT
            ASSUME    CS:CODE,DS:CODE
START:      MOV    AL,65H              ;取出原有 65H 号中断保存
            MOV    AL,35H
            INT    21H
            PUSH   ES
            PUSH   BX
            MOV    AX,SEG   LIST
            MOV    DS,AX
            MOV    DX,OFFSET   LIST
            MOV    AL,65H
            MOV    AH,25H              ;指定新的中断地址
            INT    21H
            INT    65H
            MOV    AH,4CH
            INT    21H
            STRING   DB   'THIS   IS   INT65H NO.',0DH,0AH,'$'
            LIST     PROC              ;中断处理子程序
            PUSH   AX
            PUSH   BX
            PUSH   CX
            PUSH   DX
            PUSH   SI
            PUSH   DI
            PUSH   BP
            PUSH   DS
            PUSH   ES
            STI
            PUSH   CS
            PUSH   DS
            MOV    DX,OFFSET   STRING
            MOV    AH,09H
```

```
        INT     21H
        CLI
        POP     ES
        POP     DS
        POP     BP
        POP     DI
        POP     SI
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        IRET
        LIST    ENDP
        CODE    ENDS
        END     START
```

（3）程序调试：调试步骤与实验一类似。

5. 实验记录

记录程序调试过程中出现的问题及解决过程，并加以简要说明。

2.4.6 磁盘处理程序

1. 实验目的

（1）了解 BIOS 调用与 DOS 中断在磁盘调度中的处理方式，熟悉 DOS 磁盘调用的步骤。

（2）试着编写一个 DOS 磁盘调用程序。

2. 实验原理及相关知识

温习 BIOS 调用和 DOS 磁盘调用的相关知识。

3. 实验内容

编程实现从键盘输入 20 个字符存放到利用扩充文件管理方式建立的文件中。

4. 实验步骤

（1）编程思路分析：用自己的语言加以说明。

（2）编写程序。

```
DATA SEGMENT
FNAME DB 'C:\MASM\FILE1.DAT',0
DAT     DB 80H DUP(0)
DAT1    DB 80H DUP(0)
DATA    ENDS
CODE    SEGMENT
        ASSUME CS:CODE,DS:DATA,ES:DATA
START:    MOV AX,DATA
          MOV DS,AX
          MOV ES,AX
          MOV DX,OFFSET FNAME
          MOV CX,0
          MOV AH,3CH
          INT 21H
          MOV SI,AX
```

```
NEW:    MOV BX,0
        MOV CX,14H
ERA:    MOV AH,01H
        INT 21H
        MOV DAT[BX],AL
        INC BX
        LOOP ERA
        MOV DAT[BX],0AH
        MOV DX,OFFSET DAT
        MOV CX,14H
        MOV BX,SI
        MOV AH,40H
        INT 21H
        MOV BX,SI
        MOV AH,3EH
        INT 21H
        MOV AH,4CH
        INT 21H
CODE ENDS
END START
```

5. 实验记录

记录程序调试过程中出现的问题及解决过程，分析与编程思路的联系，详细注释所编程序。

2.4.7　编程综合练习

1. 实验目的

通过本次实验掌握汇编语言的程序开发步骤，即分析－设计－编写的整个流程。

2. 实验原理及相关知识

温习 ASCII 码的相关知识。

3. 实验内容

将一个有符号十进制数转换为 ASCII 码形式的二进制数输出。

4. 实验步骤

（1）分析问题。对于有符号数，要考虑符号位的处理。处理流程如下：

① 判断数据是零、正数或负数，若是零，则显示“0”退出。

② 若是负数，则显示“-”号，并求数据的绝对值。

③ 接着将数据除以 10，余数加 30H 转换为 ASCII 码，压入堆栈。

④ 重复③步，直到余数为 0 结束。

⑤ 依次从堆栈弹出各位数字显示。

（2）设计算法。涉及子程序调用、数码转换、堆栈操作和循环操作。

（3）编写程序。

```
DATA    SEGMENT
        ARRAY   DW   1204,2334,8975
DATA    ENDS
CODE    SEGMENT
```

```
        ASSUME  DS:DATA,CS:CODE
START:  MOV     AX,DATA
        MOV     DS,AX
        LEA     BX,ARRAY
        MOV     CX,3
AGAIN:
        CALL    OUTSTREAM
        INC     BX
        INC     BX
        CALL    ENTER
        LOOP    AGAIN
OUTSTREAM  PROC
        PUSH    AX
        PUSH    BX
        PUSH    DX
        MOV     AX,[BX]
        TEST    AX,AX
        JNZ     WRITE1
        MOV     DL,30H
        MOV     AH,02
        INT     21H
        JMP     WRITEEND
WRITE1: MOV     BX,10
        PUSH    BX
WRITE2: CMP     AX,0
        JZ      WRITE3
        XOR     DX,DX
        DIV     BX
        ADD     DL,30H
        PUSH    DX
        JMP     WRITE2
WRITE3: POP     DX
        CMP     DL,10
        JE      WRITEEND
        MOV     AH,02
        INT     21H
        JMP     WRITE3
WRITEEND: POP   DX
        POP     BX
        POP     AX
        RET
OUTSTREAM  ENDP

ENTER PROC
        PUSH    DX
        MOV     AH,02H
        MOV     DL,0DH
        INT     21H
```

```
                MOV     DL,0AH
                INT     21H
                POP     DX
                RET
ENTER   ENDP
                MOV     AH,4CH
                INT     21H
CODE    ENDS
END     START
```

（4）程序调试。程序调试结果如图 2-13 所示。

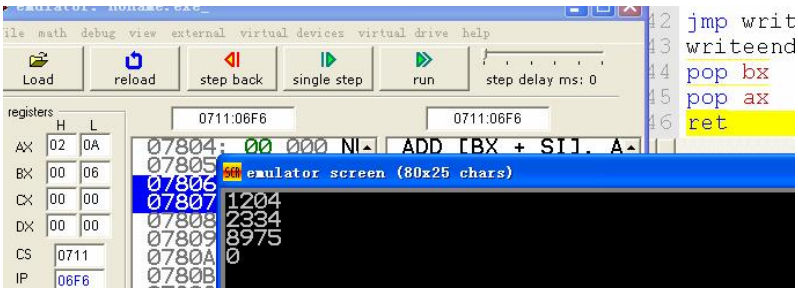

图 2-13　程序调试结果

5. 实验记录

记录程序调试过程中出现的问题及解决过程，分析与编程思路的联系，详细注释所编程序。