EXPERIMENT 5

实验5

简单的汇编语言程序设计

从本实验起，就要真正开始设计汇编语言程序了。学过程序设计语言的读者都知道，每种程序设计语言如C、C++、Java等都有各自的指令、语法结构及程序的组织方式等。汇编语言作为一种程序设计语言，也有其独特的源程序书写格式。只有正确地编辑汇编语言源程序才能得到正确的执行结果，这就需要掌握汇编语言源程序的书写格式和数据组织方式。对于初学者，必须通过大量的上机练习，才能掌握汇编语言的编程知识。

5.1 实验目的

本实验通过几个简单的例子，展示了汇编语言源程序的书写格式，涉及汇编语言源程序指令的使用及数据的组织方式等，这些都是汇编语言编程的重要问题。

5.2 实验环境

本实验中所有参考程序的编辑，可以使用任何一款你熟悉的软件，如DOS下的Edit、UltraEdit、记事本、EMU8086等，只要将其文件保存为纯文本文件格式(.asm)，然后用微软的宏汇编工具MASM 5.0汇编、连接及执行即可，当然读者也可以在虚拟机下编辑运行程序。但要注意简化段定义的汇编语言源程序书写格式在EMU8086下不能通过编译。所以有关简化段定义格式的程序必须在TASM和高版本的MASM下运行。

5.3 实验过程

例5.1 用简化段定义、DOS 09H号和4CH号功能调用编写程序，在显示器上输出“I love NANKAI!”，并上机调试。

题目分析：本例题涉及简化段定义、DOS 09H号功能调用、4CH号功能调用以及要显示的数据“I love NANKAI!”的分配和预置。需要注意的是：简化段定义只适用TASM和高版本的MASM。

(1) 简化段定义的格式这里不再详述，读者可以查阅配套教材的相关章节。

(2) DOS 09H号功能调用：显示器显示字符串。

入口参数设置：DS 指向字符串所在的段，DX 指向字符串的串首，然后置 AH 为 9，用 INT 21H 指令实现 DOS 09H 号功能的调用。注意：DOS 9H 号功能要求字符串必须以“$”作为结尾标志。

注意：

这里的 DOS 功能调用是在执行 INT 21H 指令时实现的，通过 INT 21H 执行 DOS 中断服务程序，DOS 中断服务程序有很多，它们都是系统已定义好的用于实现特定功能的程序模块。例如，这里的 DOS 09H 号功能调用是显示一串字符，4CH 号功能调用用于结束程序。如果想实现输出一串字符，这时用户不需要自己写输出字符串的程序，因为系统已经写好了，并以 DOS 中断服务程序存在。只需要设定好入口参数，并调用相应的 DOS 中断服务程序就可以了。这里设置好 DOS 09H 号功能调用的入口参数，然后通过 INT 21H 即可实现字符串的显示器输出。

(3) 4CH 号功能调用：结束程序。

入口参数设置：AH 置功能调用号 4CH 或者 AX 置 4C00H，然后 INT 21H 指令实现 DOS 4CH 号功能调用。

(4) 按照简化段定义的格式，字符串数据“I love NANKAI!”放在数据段.DATA 中。代码段.CODE 是必需的，.STACK 段可以定义也可以不定义，但是为了使程序自己管理自己的堆栈而不致混乱，最好定义.STACK 段。

完整参考程序如下。

```
DOSSEG                                  ;用于独立的汇编语言模块
.MODEL SMALL                            ;定义汇编程序使用的内存模式
.STACK 100H                             ;定义堆栈段
.DATA                                   ;定义数据段
    MS DB 'I love NANKAI!',0DH,0AH,'$'  ;显示串并回车换行
.CODE                                   ;定义代码段
START: MOV AX,@DATA
       MOV DS,AX                        ;装填数据段寄存器
       MOV DX,OFFSET MS                 ; DX指向要显示串的串首
       MOV AH,09H
       INT 21H
       MOV AX,4C00H                     ;结束程序,返回DOS功能调用
       INT 21H
       END START                        ;汇编结束
```

文件保存为 ex5_1.asm，其调试过程如图 5.1 所示。

例 5.2 用简化段定义、DOS 40H 号和 4CH 号功能调用编写程序，在显示器上输出“I love NANKAI!”，并上机调试。

题目分析：本例题涉及简化段定义、DOS 40H 号功能调用、4CH 号功能调用以及要显示的数据“I love NANKAI!”的分配和预置。

(1) DOS 40H 号功能调用：显示器显示字符串。

入口参数设置：寄存器 DS 指向要显示的字符串所在的段，DX 指向要显示的字符串的串首，CX 指明字符串的长度，BX 置标准输出设备号为 1，AH 置功能调用号 40H，最后用 INT 21H 指令实现 DOS 40H 号功能调用。

图 5.1　例 5.1 的调试过程

(2) 4CH 号功能调用：结束程序。

入口参数设置见例 5.1 中的介绍。

(3) 按照简化段定义的格式，字符串数据“I love NANKAI!”放在数据段. DATA 中。同时 DOS 40H 号功能调用需要的字符串的长度也应该放在数据段. DATA 中。代码段. CODE 是必需的，. STACK 段可以定义也可以不定义，但是为了使程序自己管理自己的堆栈而不致混乱，最好定义. STACK 段。

完整参考程序如下。

```
DOSSEG                              ;用于独立的汇编语言模块
.MODEL SMALL                        ;定义汇编程序使用的内存模式
.STACK 100H                         ;定义堆栈段
.DATA                               ;定义数据段
    MS DB 'I love NANKAI!',0DH,0AH  ;显示串并回车换行
    LM EQU $-MS                     ;显示串的长度：当前位置减去显示串首
.CODE                               ;定义代码段
START: MOV AX,@DATA
       MOV DS,AX                    ;装填数据段寄存器
       MOV BX,1                     ; BX 置标准输出设备号 1
       MOV CX,LM                    ;CX 指向显示串长度
       MOV DX,OFFSET MS             ; DX 指向要显示串的串首
       MOV AH,40H
       INT 21H
       MOV AX,4C00H                 ;结束程序，返回 DOS 功能调用
       INT 21H
       END START                    ;汇编结束
```

文件保存为 ex5_2. asm，其调试过程如图 5.2 所示。

程序说明：在上面简化段定义的汇编语言源程序书写方式中，出现了伪指令 END、DOSSEG、. DATA、. STACK、. CODE、. MODEL，而且以后的程序中还会出现不少伪指令。

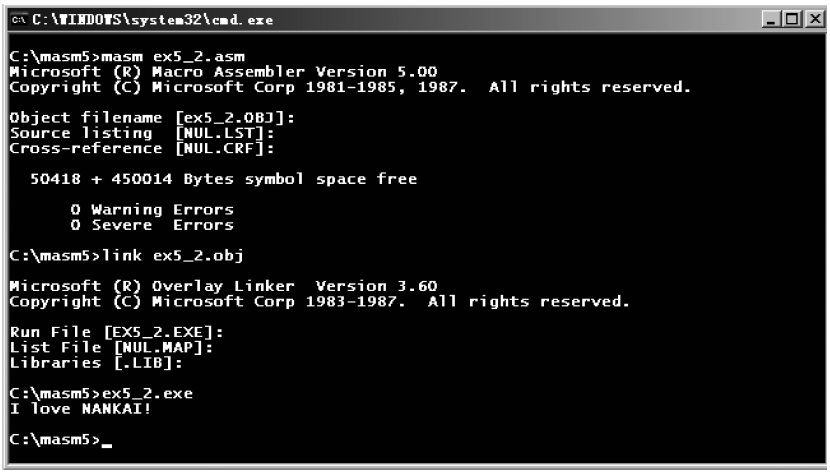

图 5.2 例 5.2 的调试过程

伪指令是汇编语言源程序格式中不可或缺的一部分。那么什么是伪指令呢？它们又和指令有什么区别呢？伪指令又称为伪操作，它们不是 80X86 芯片上的指令，没有对应的机器指令，也就没有对应的硬件动作，它们只是在源程序汇编期间由汇编程序处理以实现某种功能。

伪指令可以完成定义程序模块、定义数据、分配存储区及指示程序结束等功能。例如，伪指令 DOSSEG 指示该程序用于独立的汇编语言程序模块，而不是其他高级语言调用的汇编模块；伪指令.DATA、.STACK、.CODE 分别是数据段、堆栈段、代码段的开始；伪指令 END 表示程序结束；伪指令.MODEL 用于指出汇编语言源程序所使用的内存模式。

除伪指令外每一条汇编语言指令在汇编之后都生成相应的机器代码，然后在程序运行期间由 CPU 来执行。汇编指令的通用格式为：

[标号:]操作码[目的操作数][,源操作数][;注释]

其中：

(1) []中的内容可有可无。

(2) 标号不能是保留字，必须是以字母开头的字母或数字组成的字符串，是提供转移指令转移的目标，在程序中必须唯一，一般指令中可以没有标号。

(3) 指令末尾的分号“;”后面直到＜ENTER＞都是注释部分，注释是为了方便阅读，汇编指令翻译成机器码时会忽略这部分内容。输入源程序时，每条汇编指令末尾必须有＜ENTER＞，表示一条指令的结束，另一条指令的开始。

(4) 操作码是汇编指令的关键字，指示指令完成什么样的功能，一条汇编成机器代码的汇编指令必须有唯一的操作码。

(5) 目标操作数指示指令的结果放在何处，在许多指令中目标操作数既表示处理的对象之一来自何处，又指出处理的结果放在何处。源操作数指示指令处理的对象来自何处。

注：指令中可以没有源操作数和目的操作数，在这种情况下是对某一固定的或隐含的操作数的操作。指令中也可以是没有源操作数而仅含目标操作数的单操作数指令，这时目标操作数可能既指出处理对象来自何处，又指出结果置于何处。

1）无操作数指令

例如：

```
AAA     ;固定操作数为 AL
```

该指令没有源操作数，也没有目标操作数，此指令隐含使用目标操作数 AL，分号后直到回车符是注释部分。

2）单操作数指令

例如：

```
INC AL
```

该指令是单操作数指令，AL 是目标操作数，其功能是将 AL 的内容加 1 后送 AL。

3）双操作数指令

指令中既含有源操作数又有目标操作数的指令，称为双操作数指令。在双操作数指令中，目标操作数一定是在源操作数的左边，而且源操作数和目标操作数的类型必须一致。

例如：

```
NEXT: MOV AL,BL     ;BL->AL
```

NEXT 是标号，MOV 指令是数据传送指令，MOV 为操作码，AL 是目标操作数，BL 是源操作数，分号后面直到换行符是注释部分。该指令的功能是将源操作数 BL 中的内容送到目标操作数 AL 中。

例 5.3　用简化段定义、DOS 40H 号功能调用在显示器上输出“I love NANKAI!”，并用 RET 形式返回，最后上机调试。

题目分析：本例题涉及用 RET 的形式返回 DOS、DOS 40H 号功能调用以及要显示的数据“I love NANKAI!”的分配和预置。

(1) RET 形式返回 DOS

除了用

```
MOV AH,4CH
INT 21H
```

即 DOS 4CH 号功能调用结束程序，返回 DOS 系统外，还可以用 RET 的形式返回 DOS 系统，这时必须将执行模块定义为远(FAR)过程，一般过程的标准定义格式为：

```
过程名 PROC [NEAR/FAR]
    过程体
    RET
过程名 ENDP
```

其中，这里定义的远过程体的开始必须是：

```
PUSH DS
MOV AX,0
PUSH AX
```

为什么用 RET 指令返回时，过程体的开始必须是以上 3 条指令呢？这与程序段前缀(PSP)有关。每个应用程序加载入内存后都会在前 100H(256)个字节建立一个 PSP，真正的程序是从 PSP 之后才开始的。PSP 是 DOS 操作系统与应用程序通信的接口，供 DOS 正确加载程序时传递信息，保存参数，保存重要的 DOS 中断入口地址等。

在 PSP 中，DS 寄存器保存的是 PSP 的起始地址，偏移地址为 0 的地方存放的是 INT 20H 指令，即 0CD20H，那么 DS:0 指向的就是 INT 20H 指令，该指令用于 DOS 终止程序。所以当用 RET 指令返回操作系统时，过程是这样的：

```
PUSH DS
MOV AX,0
PUSH AX
…
RET
```

第一条指令必须是将 DS 压栈，因为程序刚加载时 DS 正好指向 PSP 段基址，然后将 0 送 AX，接着将 AX 也压栈，这样 DS:AX 被保存在栈顶，它们所指向的内存正好是 INT 20H 指令所在的位移。

RET 指令程序返回，该指令的实现正好是先将后入栈的 AX(值为 0)弹出送 IP，再将最先入栈的 DS 弹出送 CS，这样 CS:IP 正好指向了 PSP 的位移 00H，于是就将控制转移给了 INT 20H，执行该指令，程序结束，返回 DOS 操作系统。

(2) DOS 40H 号功能调用：显示器显示字符串。

入口参数设置见例 5.2 的介绍。

(3) 按照简化段定义的格式，字符串数据“I love NANKAI!”放在数据段.DATA 中。同时 DOS 40H 号功能调用需要的字符串长度也应该放在数据段.DATA 中。代码段.CODE 是必需的，.STACK 段可以定义也可以不定义，但是为了使程序自己管理自己的堆栈而不致混乱，最好定义.STACK 段。

完整参考程序如下。

```
DOSSEG                              ;用于独立的汇编语言模块
.MODEL SMALL                        ;定义汇编程序使用的内存模式
.STACK 100H                         ;定义堆栈段
.DATA                               ;定义数据段
    MS DB 'I love NANKAI!',0DH,0AH  ;显示串并回车换行
    LM EQU $-MS                     ;显示串的长度：当前位置减去显示串首
.CODE                               ;定义代码段
GO PROC FAR                         ;过程体开始，过程名为 GO
    PUSH DS
    MOV AX,0
    PUSH AX
    MOV AX,@DATA
    MOV DS,AX                       ;装填数据段寄存器
    MOV BX,1                        ;BX 置标准输出设备号 1
    MOV CX,LM                       ;CX 是输出串的长度
    MOV DX,OFFSET MS                ;DX 指向串首
```

```
    MOV AH,40H
    INT 21H                          ;DOS 40 号功能调用实现字符串的显示
    RET                              ;用过程返回 DOS
GO ENDP                              ;主程序结束
    END GO                           ;汇编结束
```

文件保存为 ex5_3.asm,其调试过程如图 5.3 所示。

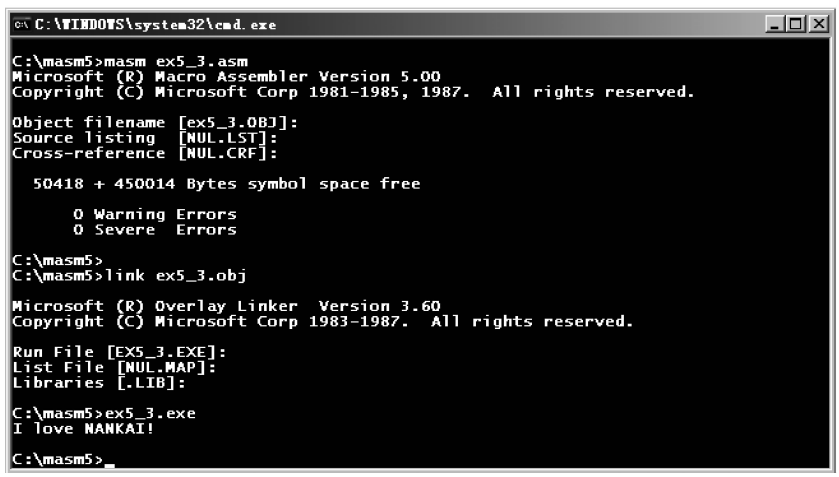

图 5.3　例 5.3 的调试过程

例 5.4　用简化段定义、DOS 09H 号功能调用在显示器上输出“I love NANKAI!”,并用 RET 形式返回,最后上机调试。

完整参考程序如下。

```
DOSSEG                               ;用于独立的汇编语言模块
.MODEL SMALL                         ;定义汇编程序使用的内存模式
.STACK 100H                          ;定义堆栈段
.DATA                                ;定义数据段
    MS DB 'I love NANKAI!',0DH,0AH,'$'   ;显示串并回车换行
.CODE                                ;定义代码段
GO PROC FAR                          ;过程体开始,过程名为 GO
    PUSH DS
    MOV AX,0
    PUSH AX
    MOV AX,@DATA
    MOV DS,AX                        ;装填数据段寄存器
    MOV DX,OFFSET MS                 ;DX 指向串首
    MOV AH,9H
    INT 21H                          ;DOS 09 号功能调用实现字符串的显示
    RET                              ;用过程返回 DOS
GO ENDP                              ;主程序结束
    END GO                           ;汇编结束
```

文件保存为 ex5_4.asm,其调试过程如图 5.4 所示。

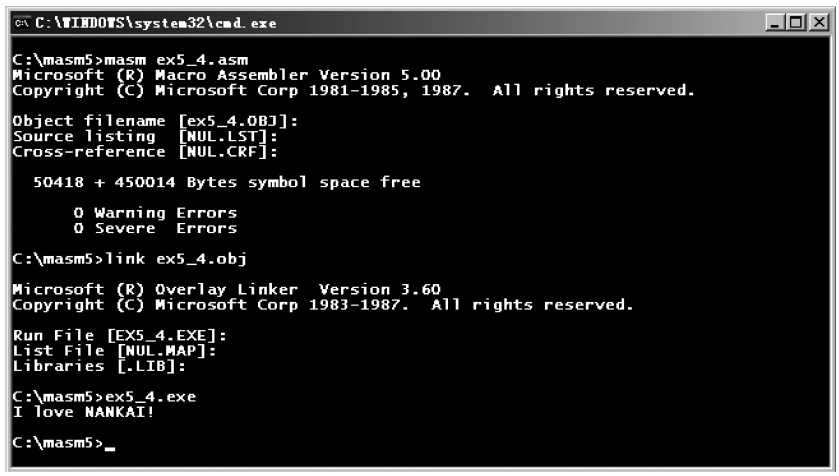

图 5.4 例 5.4 的调试过程

例 5.5 用完整段定义、RET 形式返回、DOS 9H 号功能调用在显示器上输出“I love NANKAI!”，并上机调试。

题目分析：本例题涉及完整段定义、用 RET 形式返回 DOS、DOS 9H 号功能调用以及要显示的数据“I love NANKAI!”的分配和预置。

(1) 完整段定义的格式这里不再详述，读者可以查阅课本的相关章节。

(2) 用 RET 形式返回，过程如下：

```
PUSH DS
MOV AX,0
PUSH AX
…
RET
```

(3) DOS 9H 号功能调用：显示器显示字符串。

入口参数设置见例 5.1 的介绍。

完整参考程序如下。

```
DATA SEGMENT                          ;定义逻辑数据段
    LM DB'I Love NANKAI!',0DH,0AH,'$'  ;显示字符串回车换行
DATA ENDS                             ;数据段定义结束
CODE SEGMENT                          ;定义逻辑代码段
    ASSUME CS:CODE,DS:DATA
GO PROC FAR                           ;过程体开始,过程名为 GO
    PUSH DS
    MOV AX,0
    PUSH AX
    MOV AX,DATA
    MOV DS,AX                         ;装填数据段寄存器
    MOV DX,OFFSET LM
    MOV AH,9H
```

```
    INT 21H                          ; DOS 9 号功能调用实现字符串的显示
    RET                              ;用过程返回 DOS
GO ENDP                              ;主程序结束
CODE ENDS                            ;代码段结束
    END GO                           ;汇编结束
```

程序说明：

(1) 完整段定义格式可以随意地给各段起名，如本程序的数据段和代码段分别命名为DATA 和 CODE，不像简化段定义中必须用. DATA . CODE 来表示。随意起名导致MASM 不知道各个 SEGMENT 分别是什么段，所以需用 ASSUME 伪指令描述 MASM 需要寻址的段与什么寄存器相联，通常 ASSUME 语句放在整个代码段的开始，如本例中的ASSUME CS:CODE,DS:DATA。但是 ASSUME 语句只是一个伪指令，由 MASM 汇编程序对源程序汇编期间进行处理，伪指令都没有对应的机器码，CPU 执行指令时并不知道伪指令的存在，所以语句 ASSUME CS:CODE,DS:DATA 并不能替代数据段的装填，程序中必须有装填数据段的语句，如程序中的：

```
MOV AX,DATA
MOV DS,AX
```

(2) 同简化段定义格式一样，完整段定义的汇编语言书写格式也有相应的伪指令，在上面程序中出现的 SEGMENT/ENDS、ASSUME、PROC/ENDP 都是伪指令。ASSUME、PROC/ENDP 在前面都有简要介绍。而伪指令 SEGMENT/ENDS 是为任何一个逻辑段命名，并指出该逻辑段的开始及结束位置，其格式为：

```
段名 SEGMENT
…
段名 ENDS
```

其中，段名由用户自己定义，但要注意 SEGMENT 和 ENDS 前的“段名”必须一致。

(3) 从寻址方式(寻找指令中的操作数的方式，寻址主要是寻找内存数据的地址)来看，因为立即数(字节常数或字常数)不能直接送给寄存器，而 DATA 代表段地址，是立即数，所以直接用 MOV DS,DATA 是错误的，因此必须通过通用寄存器 AX(或 BX,CX,DX 但一般用 AX)中转。这里装填寄存器 DS 用了两条指令，代码如下：

```
MOV AX,DATA
MOV DS,AX
```

(4) 程序中涉及的数据，需要在程序设计时进行预置和分配，即在一定的逻辑段中，将这些数据以一定的形式存放起来，并给出访问原则。为了方便程序设计，8088 汇编语言中给出了以变量的方式命名数据的方法。本例题中，名为“LM”的变量存放着要显示的字符串数据“I love NANKAI!”，同时变量存放在逻辑段 DATA 中。变量放在逻辑段中，逻辑段与段寄存器联系在一起，这样通过段寄存器和这些变量所在的偏移位置以及数据段中字节、字或双字的存储类型就可以找到并访问变量及变量中的数据了。

变量的定义：

```
<变量名> DB|DW|DD <表达式>
```

本例中，变量 LM 的定义：

```
LM DB'I Love NANKAI!',0DH,0AH,'$'
```

表示变量 LM 是按字节方式存储的。注意：两个以上的 ASCII 码字符串只能用 DB 定义，即只能按照字节方式存储，字符与字符或字符串之间用逗号隔开。其中'I love NANKAI!'为要显示的字符串，0DH 和 0AH 分别是回车符和换行符的 ASCII 码。

在逻辑代码段中通过以下两条指令，将段寄存器 DS 指向逻辑数据段 DATA 段首。

```
MOV AX,DATA
MOV DS,AX
```

在指令 MOV DX,OFFSET LM 中，OFFSET LM 为取变量 LM 的位移，这样 DX 的值就是变量 LM 的偏移地址，于是 DS:DX 就指向了逻辑数据段 DATA 的变量 LM 的起始处，即显示串'I love NANKAI!'的串首。

以上程序保存为 ex5_5.asm，程序调试过程如图 5.5 所示。

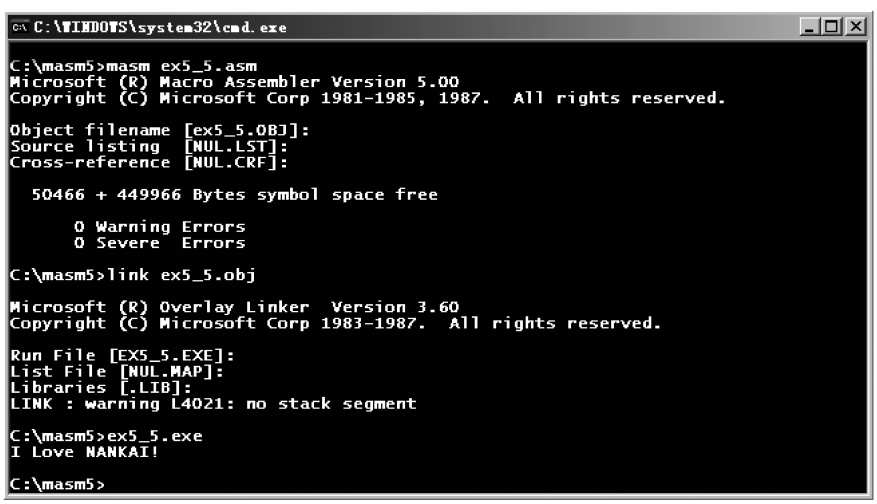

图 5.5 例 5.5 的调试过程

例 5.6 写一个人机交互程序，并且上机调试通过，具体要求如下。

题目：计算机输出 What is your name?

键盘输入你的名字，如 Zhu Yaoting(提示：用 0AH 号功能调用)

计算机输出：Hello 你的名字，这里针对刚才的输入应该是 Hello Zhu Yaoting

题目分析：本例题涉及字符串的输出和字符串的输入。

(1) 首先，计算机输出“What is your name?”，前面已讲过，DOS 9H 和 DOS 40H 均可实现字符串的显示输出，用哪个都可以，只要设置好相应的入口参数就可以了。

(2) 其次，从键盘输入你的名字，如 Zhu Yaoting。输入字符串的 DOS 功能调用是：DOS 0AH 号功能调用。其入口参数设置：调用前 DS:DX 应该指向一个缓冲区，这个缓冲区的首字节(位移为 0)用来定义缓冲区所能容纳的字符数，须预先定义好，不能为 0；第二个字节(位移为 1)为缓冲区接受的实际字符数，第二个字节不需要预先设置，系统会根据用户输入的字符数自动填充；缓冲区的第三个字节开始存放用户输入的字符，从键盘输入的

字符，逐一放到缓冲区，直到遇见＜ENTER＞键为止。

（3）输出名字：DOS 9H 和 DOS 40H 号功能调用均可。但需要注意的是：名字是用户通过键盘输入的，根据 DOS 0AH 号功能调用输入时的缓冲区存放形式，输出时必须从缓冲区的第三个字节（位移为 2）开始输出。

完整参考程序如下。

```
DATAS SEGMENT                          ;定义数据段
    STR DB 'What is your name? $'      ;定义输出串，这里用 DOS 9H 功能调用
    BUF DB 20                          ;定义键盘输入缓冲区，缓冲区大小为 20
        DB ?                           ;实际接受字符数待定
    DB 20 DUP('$')                     ;输入字符的存放位置
    CRLF DB 0AH,0DH,'$'                ;定义回车换行
    LM DB 'Hello $'                    ;字符串"Hello"
DATAS ENDS
CODES SEGMENT                          ;定义代码段
    ASSUME CS:CODES,DS:DATAS
START:
    MOV AX,DATAS
    MOV DS,AX                          ;装填数据段寄存器
    MOV DX,OFFSET STR
    MOV AH,9
    INT 21H                            ;DOS 9H 功能调用，输出“What is your name?”
    MOV DX,OFFSET CRLF
    MOV AH,9
    INT 21H                            ;DOS 9H 功能调用，输出回车换行
    MOV DX,OFFSET BUF
    MOV AH,0AH
    INT 21H                            ;DOS 0AH 功能调用，输入字符串
    MOV DX,OFFSET CRLF
    MOV AH,9
    INT 21H                            ;DOS 9H 功能调用，输出回车换行
    MOV DX,OFFSET LM
    MOV AH,9
    INT 21H                            ; DOS 9H 功能调用，输出"Hello"
    MOV DX,OFFSET BUF+2
    MOV AH,9
    INT 21H                            ;DOS 9H 功能调用，输出名字
    MOV AH,4CH                         ;结束程序，返回 DOS 功能调用
    INT 21H
CODES ENDS
    END START                          ;汇编结束
```

将程序保存为 ex5_6.asm，程序调试过程如图 5.6 所示。

说明：一个逻辑数据段中可以定义多个变量，只不过每个变量的偏移地址不同，本题中逻辑数据段 DATAS 定义了 4 个变量 STR、BUF、CRLF 和 LM，对这些变量的访问通过“OFFSET 变量名”取变量的位移值。当然一个程序也可以定义多个逻辑数据段。有关变量的定义及访问的细节这里不再详述，读者可查阅课本的相关章节。

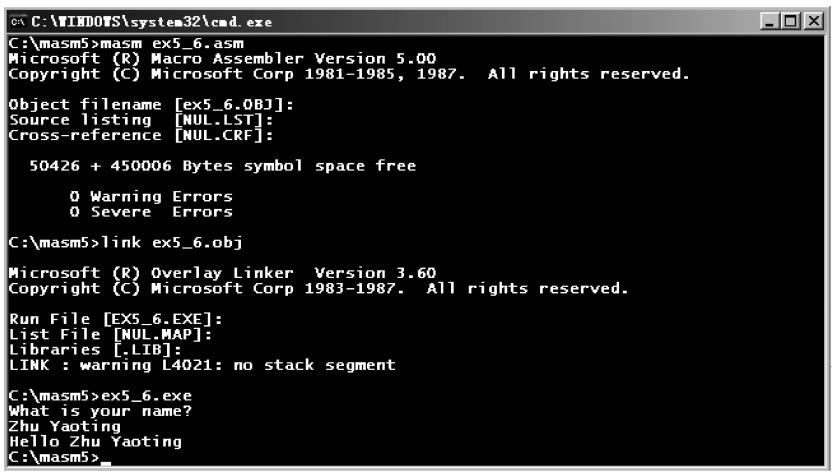

图 5.6 例 5.6 的调试过程

5.4 实 验 总 结

本实验通过几个简单的例子，使读者初步掌握了汇编语言源程序的书写格式(简化段定义、完整段定义)和数据的组织方式，程序正常结束的两种方式以及字符串输入输出的 DOS 功能方法等，为以后编写复杂汇编语言程序奠定了基础。

练 习 题 5

1. 用简化段定义、RET 形式返回、DOS 9H 号功能调用在显示器上输出带回车换行的下列字符串：

```
Hello World,
It's a miracle!
```

2. 将题 1 的字符串用完整段定义格式、DOS 40H 号功能调用显示输出、4CH 号功能调用结束返回，改写程序。

3. 参照例 5.6 自己写一个人机交互程序并上机调试通过。