

## 实验二 顺序程序设计

### 一、实验目的

1. 熟练掌握汇编语言程序的建立、汇编、连接、调试和运行的过程。
2. 掌握顺序程序设计方法。

### 二、实验内容

1. 复习教材中顺序结构程序设计的相关内容。
2. 分析程序 1：下面程序完成将 HEX 为起始地址的 2 位十六进制(ASCII 码)转换成 8 位二进制数存入 BIN 单元，请阅读程序，找出其中的错误之处。

```
DATA    SEGMENT
        ORG    1000H
BIN      DB ?
HEX      DB '6B'
        ORG    1030H
        DB 00H,01H,02H,03H,04H,05H,06H,07H,08H,09H
        ORG    1040H
        DB 0AH,0BH,0CH,0DH,0EH,0FH
DATA     ENDS
CODE     SEGMENT
START:   MOV    AX,DATA
        MOV    DS,AX
        MOV    AL,HEX
        MOV    BX,OFFSET BIN
        SUB    AH,AH
        ADD    BX,AX
        MOV    DL,[BX]
        MOV    CL,4
SHL      DL,CL
        MOV    AL,HEX+1
        MOV    BX,OFFSET BIN
        XOR    AH,AH
        ADD    BX,AX
        MOV    AL,BX
        OR     AL,DL
        MOV    BIN,AL
        MOV    AH,4CH
```

CODE      ENDS

- 编写程序 2 实现  $W=X+Y+Z$ （其中  $X=5$ ， $Y=6$ ， $Z=18$ ）。
- 编写程序 3 实现  $W=X+2Y+5Z$ （其中  $X=5$ ， $Y=6$ ， $Z=18$ ）。

### 三、 实验结果（截图）

#### 1. 代码找错

```
0871:002E 0000      ADD     [BX+SI],AL
0871:0030 0000      ADD     [BX+SI],AL
0871:0032 0000      ADD     [BX+SI],AL
0871:0034 0000      ADD     [BX+SI],AL
0871:0036 0000      ADD     [BX+SI],AL
0871:0038 0000      ADD     [BX+SI],AL
0871:003A 0000      ADD     [BX+SI],AL
0871:003C 0000      ADD     [BX+SI],AL
0871:003E 0000      ADD     [BX+SI],AL
0871:0040 0000      ADD     [BX+SI],AL
-g 26

AX=006B BX=1042 CX=1004 DX=0060 SP=0000 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076B CS=0871 IP=0026  NU UP EI PL NZ NA PO NC
0871:0026 B44C      MOV     AH,4C
-d 1000
076C:1000 6B 36 42 00 00 00 00 00-00 00 00 00 00 00 00 00  k6B.....
076C:1010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076C:1020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076C:1030 00 01 02 03 04 05 06 07-08 09 00 00 00 00 00 00  .....
076C:1040 00 0A 0B 0C 0D 0E 0F 00-00 00 00 00 00 00 00 00  .....
076C:1050 B8 6C 07 8E D8 A0 01 10-BB 00 10 2A E4 03 D8 8A  .l.....*.
076C:1060 17 B1 04 D2 E2 A0 02 10-BB 00 10 32 E4 03 D8 8A  .....2....
076C:1070 07 0A C2 A2 00 10 B4 4C-CD 21 00 00 00 00 00 00  .....L.!.....
```

上图为修改完后代码的运行结果，可以发现 076C:1001、076C:1002 处为‘6B’的 ASCII 码，操作完之后 076C:1000 处存的二进制内容就是‘6B’本身。

#### 2. 程序 2

```
077D:002B 8BEC      MOV     BP,SP
077D:002D 81ECB600    SUB     SP,00B6
077D:0031 57          PUSH    DI
077D:0032 56          PUSH    SI
077D:0033 B8BE05      MOV     AX,05BE
077D:0036 50          PUSH    AX
077D:0037 E8C371      CALL    71FD
077D:003A 83C402      ADD     SP,+02
077D:003D 8BF0      MOV     SI,AX
077D:003F 0BF6      OR      SI,SI
-g 15

AX=001D BX=0000 CX=012A DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076D CS=077D IP=0015  NU UP EI PL NZ NA PE NC
077D:0015 B8004C      MOV     AX,4C00
-d 0
076C:0000 05 00 06 00 12 00 1D 00-00 00 00 00 00 00 00 00  .....
076C:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076C:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076C:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076C:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076C:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076C:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076C:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
```

从截图中可以看到从 076C:0000 开始，分别为 5、6、12、1D，最终结果为

1D，即 29，是正确的。

3. 程序 3

```
077D:002D 81EC8600      SUB     SP,0086
077D:0031 57          PUSH    DI
077D:0032 56          PUSH    SI
077D:0033 B8BE05      MOV     AX,05BE
077D:0036 50          PUSH    AX
077D:0037 E8C371      CALL    71FD
077D:003A 83C402      ADD     SP,+02
077D:003D 8BF0      MOV     SI,AX
077D:003F 0BF6      OR      SI,SI
077D:0041 7461      JZ      00A4
-g 23

AX=005A BX=006B CX=013B DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076D CS=077D IP=0023  NU UP EI PL NZ NA PO NC
077D:0023 B8004C      MOV     AX,4C00
-d 0
076C:0000  05 00 06 00 12 00 6B 00-00 00 00 00 00 00 00 00  .....k.....
076C:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076C:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076C:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076C:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076C:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076C:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076C:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
```

从截图中可以看到从 076C:0000 开始，分别为 5、6、12、6B，最终结果为 6B，即 107，是正确的。

四、 实验报告要求（习题）

1. 列出源程序（加以注释），说明程序的基本结构。

(1) 程序 1

下方的代码为修改之后的正确代码，其中标红的为删除部分，标蓝的为添加部分。

- ① 数据段定义部分的“ORG 1040H”改为“ORG 1041H”
- ② 操作第二个 ASCII 码时的“MOV AX,BX”改为“mov al, [bx]”

```
DATA    SEGMENT
        ORG    1000H
BIN      DB      0
HEX      DB      '6B'
        ORG    1030H
        DB      00H,01H,02H,03H,04H,05H,06H,07H,08H,09H
        ORG    1040H
        ORG    1041H
        DB      0AH,0BH,0CH,0DH,0EH,0FH
DATA     ENDS
CODE     SEGMENT
```

```
    assume  cs:code,ds:data
START:
    MOV AX,DATA
        MOV  DS,AX
        MOV  AL,HEX
        MOV  BX,OFFSET BIN
        SUB  AH,AH
        ADD  BX,AX
        MOV  DL,[BX]
        MOV  CL,4
    SHL DL,CL
        MOV  AL,HEX+1
        MOV  BX,OFFSET BIN
        XOR  AH,AH
        ADD  BX,AX
MOV  AX,BX
    mov  al, [bx]
        OR   AL,DL
        MOV  BIN,AL
        MOV  AH,4CH
        int  21H
CODE    ENDS
    END START
```

这段代码完成的操作就是将“6B”从 ASCII 码（HEX 处）转为“6B”这个字节并存到内存（BIN）中。使用的方法就是建立一个从 ASCII 码到二进制本身的映射。在本段代码中，1030H~1039H（十进制下低 8 位为 48~57）中的内容为 0~9；1041H~1046H（十进制下低 8 位为 65~70）中的内容为 A~F，完成了从 ASCII 值到其表示的字符本身的二进制的映射

## （2）程序 2

```
DATA SEGMENT
    X DW 5
    Y DW 6
    Z DW 18
    W DW 0
DATA ENDS

STACK SEGMENT STACK
```

```
DW 128 DUP(0)
STACK ENDS

CODE SEGMENT
    ASSUME cs:CODE, DS:DATA, SS:STACK
START:
; 设置段寄存器
    mov ax, DATA
    mov ds, ax
; 将 ax 清零
    xor ax, ax
; 完成 X+Y+Z 计算，最终结果保持存在 ax 中
    mov ax, X
    add ax, Y
    add ax, Z
; 将计算结果存入 W 中
    mov W, ax
; 设置 21H 的带返回值退出的功能码
    mov ax, 4C00H
    int 21H
CODE ENDS

END START
```

### (3) 程序 3

```
DATA SEGMENT
    X DW 5
    Y DW 6
    Z DW 18
    W DW 0
DATA ENDS

STACK SEGMENT STACK
    DW 128 DUP(0)
STACK ENDS

CODE SEGMENT
```

```
        ASSUME cs:CODE, DS:DATA, SS:STACK
START:
; 设置段寄存器
        mov ax, DATA
        mov ds, ax
; 将 ax 清零
        xor bx, bx
; 始终保持计算结果存在 bx 中
; X
        mov bx, X
; X+2Y
        mov ax, Y
        mov dx, 2
        imul dx
        add bx, ax
; X+2Y+5Z
        mov ax, Z
        mov dx, 5
        imul dx
        add bx, ax
; 结果存于 W
        mov W, bx

        mov ax, 4C00H
        int 21H
CODE ENDS

END START
```

## 2. 说明如何用 Debug 相关命令查看程序 1、2、3 的运行结果。

下面叙述本次报告使用 debug 查看运行结果的过程

- ① 使用 debug [exe 程序名]，进入程序的调试
- ② 使用 g [断点]运行程序至断点处，断点可以使用相对地址指定，需要注意的是，指定的断点位置的语句并不会运行
- ③ 使用 r 命令观察寄存器，计算过程中把结果存在了 AX 中
- ④ 使用 d [内存地址]查看指定内存单元以后一段的内容，该命令中的地址也可以使用相对地址进行指定，默认查看的是 DS:指定偏移地址的位置

### 3.上机调试过程中遇到的问题是如何解决的。

原先使用的是 emu8086 软件进行汇编代码模拟，但发现有些代码即使正确也无法正常模拟汇编与调试。在查阅相关资料并实机测试后得出结论：emu8086 不支持 ORG 伪指令手动设置偏移地址，并且在数据段定义中不能使用？作为初始值。对于第二个问题，只需要将暂时不给入值的赋为 0 即可。

## 五、 个人体会与总结

本次实验中给出了一道程序理解与改错题，对于这种改错的问题，若给出了程序的功能说明就简单地将其理解，若没给出，就需要先整体浏览一遍代码，大致搞清楚做了些什么事。与高级程序设计语言不同，汇编程序源代码在粗略地浏览下很可能无法找到头绪，可以在之后的调试中逐步理解。

首先必然是通过编译手段查看代码是否有语法错误，有的话需要先将其改正。将代码跑起来之后，对其进行调试，注意观察各个量的初始值与内存的变化，从理解代码意义出发，逐步改正其逻辑上的错误。