

实验三 简单编程练习

学号： 04022212

姓名： 钟 源

一、实验任务和实验结果

请写出 **已通过验收** 的各个实验任务的具体内容、调试通过的源程序（**加注释**）和实验结果。

实验结果请截图，并加以必要说明。

1. 在一个数据块中找出最大数。

假设数据块中的数据为 22、46、32、72、84、16、**156**，数据块的长度存放在 CX 寄存器中。

1) 数据块中的数据为 **无符号数**，找出其中的最大数存放在以 **MAXN1** 为符号的单元中。

2) 数据块中的数据为 **有符号数**，找出其中的最大数存放在以 **MAXN2** 为符号的单元中。

1) ① 源程序及注释：

```
DATA SEGMENT ;定义数据段,DATA 为段名
    NUM DB 22D,46D,32D,72D,84D,16D,156D ;初始化内存,用十进制表达
    MAXN1 DB ? ;设置 MAXN1 用于存放结果
DATA ENDS ;定义数据段结束

CODE SEGMENT PARA ;定义代码段
ASSUME CS:CODE,DS:DATA ;ASSUME 伪指令定义各段寄存器的内容
START: MOV AX,DATA
        MOV DS,AX ;DS 初始化为数据段首地址的段值 DATA

        MOV CX,07 ;CX 定义为数据段长度
        LEA SI,NUM ;NUM 的偏移地址赋给 SI
        MOV AL,[SI] ;将数据段的第一个数据放到 AL 里
AGAIN:  MOV BL,[SI] ;比较的第一步,数据移到 BL 上
        CMP AL,BL ;比较 AL 和 BL
        JAE NEXT ;如果 AL 大于等于 BL 则跳转到 NEXT
        MOV AL,BL ;如果相反,将 BL 的内容给 AL
NEXT:   INC SI ;SI 指向下一个空间
        LOOP AGAIN ;loop 循环,准备比较下一个数
        MOV MAXN1,AL ;循环完成后,把最大值给 MAXN1

        MOV AH,4CH ;功能号 4CH
```

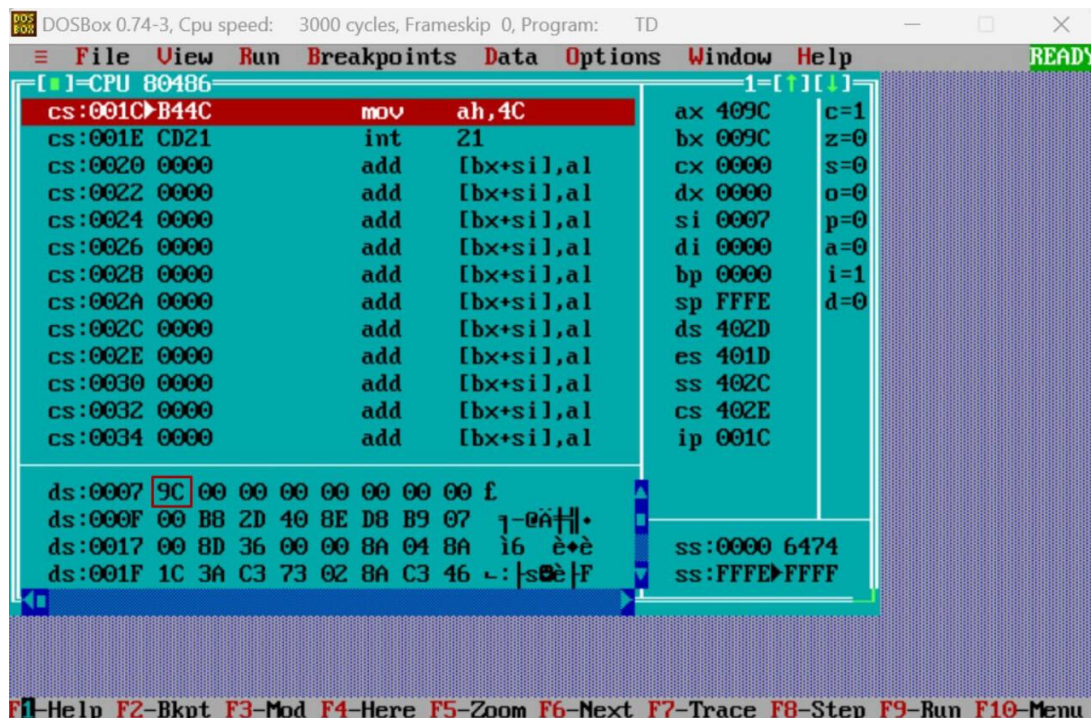
```

INT 21H ;类型号 21H, 利用中断向量返回 DOS

CODE ENDS ;代码段结束
END START ;整个程序汇编结束

```

② 实验结果:



MAXN1 偏移地址为[0007], 存储结果为 9CH (156D)。

2) ① 源程序及注释:

```

DATA SEGMENT ;定义数据段, DATA 为段名
NUM DW 22D, 46D, 32D, 72D, 84D, 16D, 156D ;初始化内存, 用十进制表达
MAXN2 DW ? ;设置 MAXN2 用于存放结果
DATA ENDS ;定义数据段结束

CODE SEGMENT PARA ;定义代码段
ASSUME CS:CODE, DS:DATA ;ASSUME 伪指令定义各段寄存器的内容
START: MOV AX, DATA ;NUM 的偏移地址赋给 SI
        MOV DS, AX ;DS 初始化为数据段首地址的段值 DATA

        MOV CX, 07 ;CX 定义为数据段长度
        LEA SI, NUM ;NUM 的偏移地址赋给 SI
        MOV AX, 0000 ;AX 清零
        MOV AX, [SI]

AGAIN: MOV BL, [SI] ;比较的第一步, 数据移到 BL 上

```

```

    CMP AX,BX          ;比较 AX 和 BX
    JGE NEXT          ;如果 AL 大于 BL 则跳转到 NEXT
    MOV AX,BX          ;如果相反, 将 BX 的内容给 AX
NEXT:  INC SI          ;SI 指向下一个空间
    INC SI             ;由于每单位为字, 故需要操作两次
    LOOP AGAIN        ;loop 循环, 准备比较下一个数
    MOV MAXN2,AX       ;循环完成后, 把最大值给 MAXN2

    MOV AH,4CH         ;功能号 4CH
    INT 21H            ;类型号 21H, 利用中断向量返回 DOS
CODE ENDS             ;代码段结束
END START             ;整个程序汇编结束

```

② 实验结果：

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TD

File View Run Breakpoints Data Options Window Help

[CPU 80486] 1=[↑][↓]

cs:000C B80000	mov	ax,0000	ax 009C	c=1
cs:000F 8B04	mov	ax,[si]	bx 009C	z=0
cs:0011 8A1C	mov	bl,[si]	cx 0000	s=0
cs:0013 3BC3	cmp	ax,bx	dx 0000	o=0
cs:0015 7D02	jnl	0019	si 000E	p=0
cs:0017 8BC3	mov	ax,bx	di 0000	a=0
cs:0019 46	inc	si	bp 0000	i=1
cs:001A 46	inc	si	sp FFFE	d=0
cs:001B E2F4	loop	0011	ds 402D	
cs:001D A30E00	mov	[000E],ax	es 401D	
cs:0020 B44C	mov	ah,4C	ss 402C	
cs:0022 CD21	int	21	cs 402E	
cs:0024 0000	add	[bx+si],al	ip 0020	

ds:000E 9C 00 B8 2D 40 8E D8 B9 f 7-0Ä+||

ds:0016 07 00 8D 36 00 00 B8 00 * i6

ds:001E 00 8B 04 8A 1C 3B C3 7D i+è-; -}

ds:0026 02 8B C3 46 46 E2 F4 A3 0i|FFrú

ss:0000 6474

ss:FFFE FFFF

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

MAXN2 偏移地址为[000E], 存储结果为 009CH (156D)。

3. 求无符号字节数据之和, 和数为 16 位二进制数。

假设有数据 58、25、45、73、64、43, 数据块的长度存放在 CX 寄存器中, 和数存放在以

SUM 为符号的字单元中。

(1) 源程序及注释：

```

DATA SEGMENT
    NUM DB 58D,25D,45D,73D,64D,43D    ;NUM 指定的内存单元（字节型）赋初值
    SLENGTH EQU 6                      ;定义 NUM 长度为 6
    SUM DW ?                            ;定义 SUM
DATA ENDS

CODE SEGMENT PARA                      ;定义代码段
ASSUME CS:CODE,DS:DATA                ;ASSUME 伪指令定义各段寄存器的内容
START:  MOV AX,DATA                    ;NUM 的偏移地址赋给 SI
        MOV DS,AX                      ;DS 初始化为数据段首地址的段值 DATA

        MOV CX, SLENGTH                ;CX 定义为数据段长度
        LEA SI, NUM                    ;NUM 的偏移地址赋给 SI
        MOV AX, 0000                  ;AX 清零
        MOV BX, 0000                  ;BX 清零
AGAIN:  MOV BL, [SI]                  ;将 SI 所指向的内放到 BL 中
        ADD AX, BX                     ;将 BX 里面的数加到 AX 中
        INC SI                         ;SI 指向下一个数
        LOOP AGAIN                    ;这里 cx-=1, 再判断 cx? =0, 决定是否跳过 loop
        MOV SUM, AX                   ;将和数存放在以 SUM 为符号的单元中

        MOV AH,4CH                    ;功能号 4CH
        INT 21H                       ;类型号 21H, 利用中断向量返回 DOS
CODE ENDS                              ;代码段结束
END START                              ;整个程序汇编结束

```

(2) 实验结果：

The screenshot shows the DOSBox 0.74-3 interface with the CPU window open. The assembly code is displayed on the left, and the register and memory values are shown on the right. The current instruction being executed is `mov ah,4C` at address `cs:001C`. The registers show `ax=0134`, `bx=002B`, `cx=0000`, `dx=0000`, `si=0006`, `di=0000`, `bp=0000`, `sp=FFFE`, `ds=402D`, `es=401D`, `ss=402C`, and `ip=001C`. The memory window shows the contents of `ds` and `ss` segments.

Address	Code	Comment	Register/Value
cs:0000	B82D40	mov ax,402D	ax 0134
cs:0003	8ED8	mov ds,ax	bx 002B
cs:0005	B90600	mov cx,0006	cx 0000
cs:0008	8D360000	lea si,[0000]	dx 0000
cs:000C	B80000	mov ax,0000	si 0006
cs:000F	BB0000	mov bx,0000	di 0000
cs:0012	8A1C	mov bl,[si]	bp 0000
cs:0014	03C3	add ax,bx	sp FFFE
cs:0016	46	inc si	ds 402D
cs:0017	E2F9	loop 0012	es 401D
cs:0019	A30600	mov [0006],ax	ss 402C
cs:001C	B44C	mov ah,4C	cs 402E
cs:001E	CD21	int 21	ip 001C
cs:0020	0000	add [bx+si],al	
cs:0022	0000	add [bx+si],al	

SUM 偏移地址为[0006], 存储结果为 0134H。

4.求两个十进制数相乘的积($56093 \times 5 = ?$)改为($53348 \times 9 = ?$), 被乘数和乘数均以非压缩 BCD 码

表示, 并存放在内存中, 乘积以非压缩 BCD 码的格式存放在以 SUM 为起始符号的单元中。

(1) 源程序及注释:

```
DATA SEGMENT PARA
NUM1 DB 03,09,00,06,05,00 ;存放被乘数 56093 的非压缩 BCD 码
NUM2 DB 05 ;存放乘数的非压缩 BCD 码
NUM3 DB 08,04,03,03,05,00 ;存放被乘数 53348 的非压缩 BCD 码
NUM4 DB 09 ;存放乘数的非压缩 BCD 码
SUM1 DB 8 DUP(?) ;定义一个未知长度的空间存放和的非压缩 BCD 码形式
SUM2 DB ? ;定义一个未知长度的空间存放和的非压缩 BCD 码形式
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
        MOV DS,AX ;DS 初始化为数据段首地址的段值 DATA

        LEA SI, NUM2 ;将乘数的偏移地址给 SI
        MOV BL,[SI] ;将乘数赋值给 BL,BL=05
        LEA SI, NUM1 ;将被乘数的偏移地址给 SI
        MOV AL,[SI] ;将被乘数的非压缩 BCD 码最低位赋值给 AL
        LEA DI,SUM1 ;将结果偏移地址给 DI,DI 指向最终结果[000E]
        MOV CX,06 ;循环 6 次
        MOV DL, 00 ;将 DL 清零
AGAIN1: MUL BL ;当前被乘数与乘数相乘
        AAM ;二化十指令, 将乘法结果调整为非压缩 BCD 码表示
        ADD AL,DL ;将前一位的进位加上当前结果
        AAA ;加法结果调整为非压缩 BCD 码, 将进位加到 AH 上
        MOV DL, AH ;将当前的进位存到 DL
        MOV AH, 00H ;清零 AH
        MOV [DI],AL ;将当前位的值存到存储单元
        INC SI ;SI 指向下一个单元
        INC DI ;DI 指向下一个单元
        MOV AL,[SI] ;将被乘数下一位的非压缩 BCD 码赋值给 AL
        LOOP AGAIN1 ;在指定的循环次数内循环计算

        LEA SI, NUM4 ;将乘数的偏移地址给 SI
```

```

MOV BL,[SI]           ;将乘数赋值给 BL,BL=05
LEA SI, NUM3          ;将被乘数的偏移地址给 SI
MOV AL,[SI]           ;将被乘数的非压缩 BCD 码最低位赋值给 AL
LEA DI,SUM2           ;将结果偏移地址给 DI,DI 指向最终结果[0016]
MOV CX,06             ;循环 6 次
MOV DL, 00            ;将 DL 清零
AGAIN2: MUL BL         ;当前被乘数与乘数相乘
AAM                   ;二化十指令，将乘法结果调整为非压缩 BCD 码表示
ADD AL,DL             ;! 将前一位的进位加上当前结果
AAA                   ;加法结果调整为非压缩 BCD 码，将进位加到 AH 上
MOV DL, AH            ;! 将当前的进位存到 DL
MOV AH, 00H           ;清零 AH
MOV [DI],AL           ;将当前位的值存到存储单元
INC SI                ;SI 指向下一个单元
INC DI                ;DI 指向下一个单元
MOV AL,[SI]           ;将被乘数下一位的非压缩 BCD 码赋值给 AL
LOOP AGAIN2           ;在指定的循环次数内循环计算

MOV AH,4CH            ;功能号 4CH
INT 21H               ;类型号 21H，利用中断向量返回 DOS
CODE ENDS             ;代码段结束
END START              ;整个程序汇编结束

```

(2) 实验结果：

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TD

File View Run Breakpoints Data Options Window Help

[CPU] 80486

cs:004D 8805	mov	[di],al	ax 0009	c=0
cs:004F 46	inc	si	bx 0009	z=0
cs:0050 47	inc	di	cx 0000	s=0
cs:0051 8A04	mov	al,[si]	dx 0000	o=0
cs:0053 E2ED	loop	0042	si 000D	p=0
cs:0055 B44C	mov	ah,4C	di 001C	a=0
cs:0057 CD21	int	21	bp 0000	i=1
cs:0059 0000	add	[bx+si],al	sp FFFE	d=0
cs:005B 0000	add	[bx+si],al	ds 402D	
cs:005D 0000	add	[bx+si],al	es 401D	
cs:005F 0000	add	[bx+si],al	ss 402C	
cs:0061 0000	add	[bx+si],al	cs 402F	
cs:0063 0000	add	[bx+si],al	ip 0055	

ds:000E 05 06 04 00 08 02 00 00

ds:0016 02 03 01 00 08 04 00 00

ds:001E 00 00 B8 2D 40 8E D8 8D

ds:0026 36 06 00 8A 1C 8D 36 00

ss:0000 6474

ss:FFFE FFFF

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

SUM1 偏移地址为[000E], 存储结果为 02 08 00 04 06 05H (56093×5);

SUM2 偏移地址为[0016], 存储结果为 04 08 00 01 03 02H (53348×9)。

5.请用串传送指令编写程序,将以 STR1 为首地址的字节存储单元中的数据 30H、31H、32H、33H、34H、35H、36H、37H、38H、39H、40H、41H、42H、43H、44H、45H, 传送到以 STR2 为首地址的字节存储单元中。

(1) 源程序及注释:

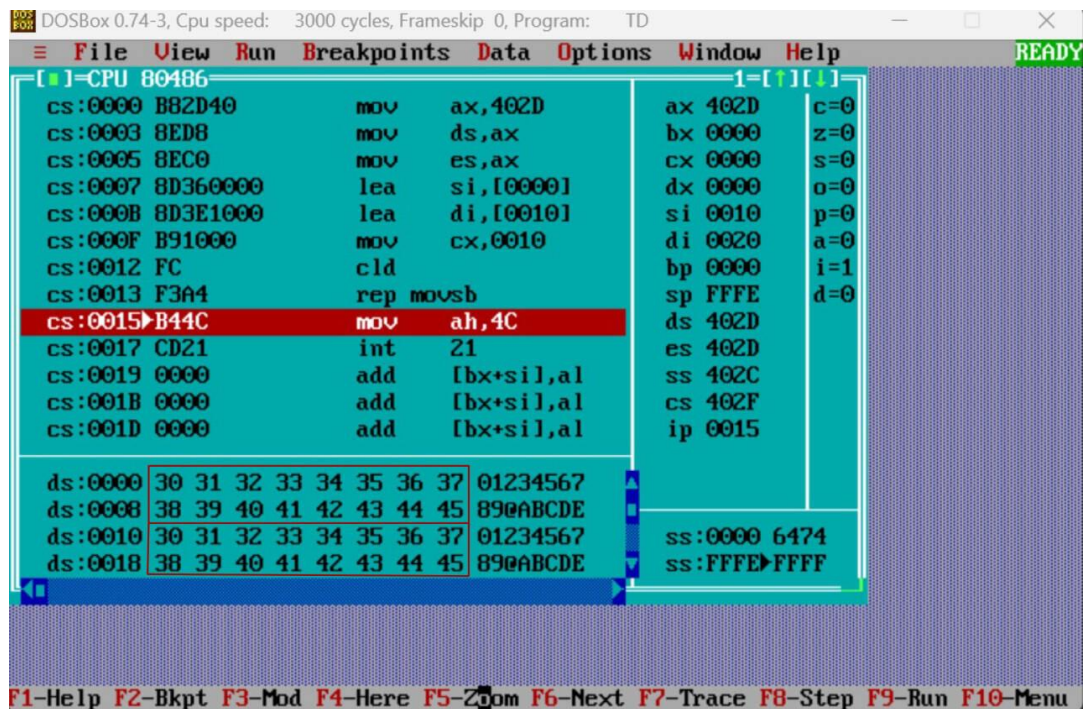
```
DATA SEGMENT
STR1 DB 30H,31H,32H,33H,34H,35H,36H,37H,38H,39H,40H,41H,42H,43H,44H,45H
SLENGTH EQU $-STR1           ;直接获取元素个数
STR2 DB SLENGTH DUP(0)       ;STR2 为将要移动的目标地址
DATA ENDS

CODE SEGMENT
ASSUME DS: DATA, CS: CODE, ES: DATA
START:  MOV AX,DATA
        MOV DS,AX             ;DS 初始化为数据段首地址的段值 DATA

        MOV ES, AX
        LEA SI, STR1          ;取 STR1 地址
        LEA DI, STR2          ;取 STR2 地址
        MOV CX, SLENGTH       ;CX 定义为要移动的字符串长度
        CLD                   ;设为指针向后移动,地址加
        REP MOVSB             ;移动字符串

        MOV AH,4CH            ;功能号 4CH
        INT 21H               ;类型号 21H, 利用中断向量返回 DOS
CODE ENDS                     ;代码段结束
END START                     ;整个程序汇编结束
```

(2) 实验结果:



STR1 偏移地址为[0000],存储结果为 30H,31H32H,33H,34H,35H,36H,37H,38H39H, 40H,41H,42H,43H,44H,45H;

STR2 偏移地址为[0010],存储结果为 30H,31H32H,33H,34H,35H,36H,37H,38H39H, 40H,41H,42H,43H,44H,45H。

附加任务（选做）：

6. 将任务 4 的乘积在屏幕上显示出来。

提示：用 DOS 系统功能调用的字符显示或字符串显示的功能。

(1) 源程序及注释：

```
DATA SEGMENT PARA
    NUM1 DB 03,09,00,06,05,00 ;存放被乘数 56093 的非压缩 BCD 码
    NUM2 DB 05 ;存放乘数的非压缩 BCD 码
    NUM3 DB 08,04,03,03,05,00 ;存放被乘数 53348 的非压缩 BCD 码
    NUM4 DB 09 ;存放乘数的非压缩 BCD 码
    SUM1 DB 8 DUP(?) ;定义一个未知长度的空间存放和的非压缩 BCD 码形式
    SUM2 DB ? ;定义一个未知长度的空间存放和的非压缩 BCD 码形式
    STR2 DB 6 DUP(?),'$' ;定义以$结尾的字符串
```



```

DATA ENDS

CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:  MOV AX,DATA
        MOV DS,AX          ;DS 初始化为数据段首地址的段值 DATA

        LEA SI, NUM2       ;将乘数的偏移地址给 SI
        MOV BL,[SI]        ;将乘数赋值给 BL,BL=05
        LEA SI, NUM1       ;将被乘数的偏移地址给 SI
        MOV AL,[SI]        ;将被乘数的非压缩 BCD 码最低位赋值给 AL
        LEA DI,SUM1        ;将结果偏移地址给 DI,DI 指向最终结果[000E]
        MOV CX,06          ;循环 6 次
        MOV DL, 00         ;将 DL 清零
AGAIN1: MUL BL              ;当前被乘数与乘数相乘
        AAM                ;二化十指令，将乘法结果调整为非压缩 BCD 码表示
        ADD AL,DL          ;将前一位的进位加上当前结果
        AAA                ;加法结果调整为非压缩 BCD 码，将进位加到 AH 上
        MOV DL, AH         ;将当前的进位存到 DL
        MOV AH, 00H        ;清零 AH
        MOV [DI],AL        ;将当前位的值存到存储单元
        INC SI             ;SI 指向下一个单元
        INC DI             ;DI 指向下一个单元
        MOV AL,[SI]        ;将被乘数下一位的非压缩 BCD 码赋值给 AL
        LOOP AGAIN1        ;在指定的循环次数内循环计算

        LEA SI, NUM4       ;将乘数的偏移地址给 SI
        MOV BL,[SI]        ;将乘数赋值给 BL,BL=05
        LEA SI, NUM3       ;将被乘数的偏移地址给 SI
        MOV AL,[SI]        ;将被乘数的非压缩 BCD 码最低位赋值给 AL
        LEA DI,SUM2        ;将结果偏移地址给 DI,DI 指向最终结果[0016]
        MOV CX,06          ;循环 6 次
        MOV DL, 00         ;将 DL 清零
AGAIN2: MUL BL              ;当前被乘数与乘数相乘
        AAM                ;二化十指令，将乘法结果调整为非压缩 BCD 码表示
        ADD AL,DL          ;! 将前一位的进位加上当前结果
        AAA                ;加法结果调整为非压缩 BCD 码，将进位加到 AH 上
        MOV DL, AH         ;! 将当前的进位存到 DL
        MOV AH, 00H        ;清零 AH
        MOV [DI],AL        ;将当前位的值存到存储单元
        INC SI             ;SI 指向下一个单元
        INC DI             ;DI 指向下一个单元
        MOV AL,[SI]        ;将被乘数下一位的非压缩 BCD 码赋值给 AL
        LOOP AGAIN2        ;在指定的循环次数内循环计算

```

```

SHOW:
    MOV CX,06          ;给 CX 赋值，CX=结果的字节数
    DEC DI             ;先让 DI 指向结果的最高位
    LEA SI,STR2        ;取 STR2 的地址到 SI
TRAN:
    MOV DL,[DI]        ;将[DI]的值赋给 DL
    ADD DL,30H         ;将数字转化为 ASC 码
    MOV [SI],DL        ;将转换结果赋给[SI]
    DEC DI             ;将 DI 指向低一个字节字符
    INC SI             ;将 SI 指向高一个字节字符
    LOOP TRAN          ;循环转换

    LEA SI,STR2        ;取 STR2 地址
    MOV DX,SI          ;赋值地址给 DX
    MOV AH,09          ;功能号 09H
    INT 21H            ;类型号 21H，显示 DS: DX 指向的以$结尾的字符串

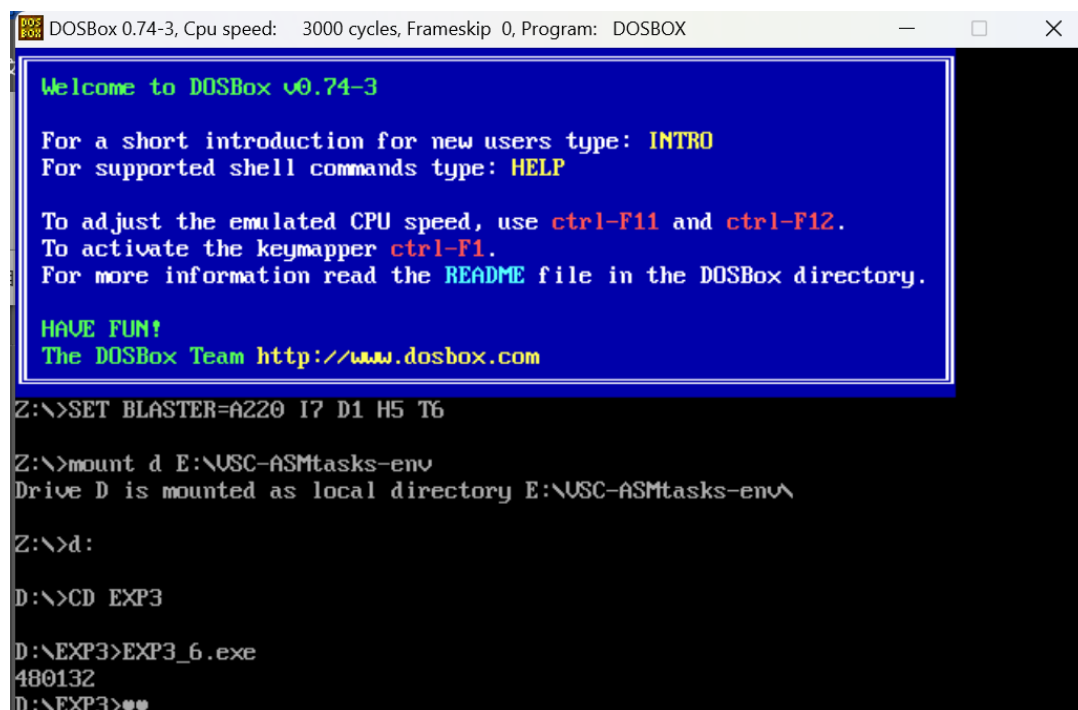
    MOV AH,4CH         ;功能号 4CH
    INT 21H            ;类型号 21H，利用中断向量返回 DOS

CODE ENDS             ;代码段结束
END START             ;整个程序汇编结束

```

注：SHOW 之前的部分与第四题代码一致（除了加了 STR2 的定义）

(2) 实验结果：



```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

Welcome to DOSBox v0.74-3

For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount d E:\USC-ASMTasks-env
Drive D is mounted as local directory E:\USC-ASMTasks-env\

Z:\>d:

D:\>CD EXP3

D:\EXP3>EXP3_6.exe
480132
D:\EXP3>

```

输出结果为 480132，与第 4 题的 SUM2 结果一致。

7. 在数据段和附加数据段中各定义 10 字节的字符串，请编程比较这两个字符串是否完全相同。

若两串完全相同，则将数据段中存放比较结果的 RESULT1 单元赋值为 0；

若两串不同，则将源串中第 1 个不相同字节的地址赋给数据段中的 RESULT1 单元，并将该字节内容送到数据段中的 RESULT2 单元。

(1) 源程序及注释：

```
DATA SEGMENT
    STR1 DB 'HelloWorld$' ; 第一个字符串
    STR2 DB 'HelloSEUer$' ; 第二个字符串
    RESULT1 DW ?          ; 用于存放比较结果的单元（地址或 0）
    RESULT2 DB ?          ; 用于存放不同字节内容的单元
DATA ENDS

CODE SEGMENT                ; 定义代码段
ASSUME CS:CODE,DS:DATA      ; ASSUME 伪指令定义各段寄存器的内容
START:  MOV AX, DATA;
        MOV DS, AX          ; 给 DS 附上初值

        MOV SI, OFFSET STR1 ; 将字符串 1 的偏移地址加载到 SI
        MOV DI, OFFSET STR2 ; 将字符串 2 的偏移地址加载到 DI
        MOV CX, 10          ; 循环次数，字符串长度为 10

COMPARE_LOOP:
        MOV AL, [SI]         ; 将 SI 指向的字符串 1 中的字符加载到 AL
        CMP AL, [DI]         ; 将 DI 指向的字符串 2 中的字符与 AL 比较
        JNE NOT_EQUAL        ; 如果不相等，跳转到不相等处理部分
        INC SI               ; 指向下一个字符
        INC DI               ; 指向下一个字符
        LOOP COMPARE_LOOP    ; 继续比较直到循环结束

        ; 如果执行到这里，说明字符串完全相同
        MOV RESULT1, 0       ; 将结果设置为 0
        JMP END_CODE         ; 跳转，结束程序

NOT_EQUAL:
        MOV RESULT1, SI      ; 将 SI 中的地址存入 RESULT1，表示第一个不相等字节的地址
        MOV AL, [SI]         ; 将第一个不相等的字符加载到 AL
        MOV RESULT2, AL      ; 将该字符存入 RESULT2

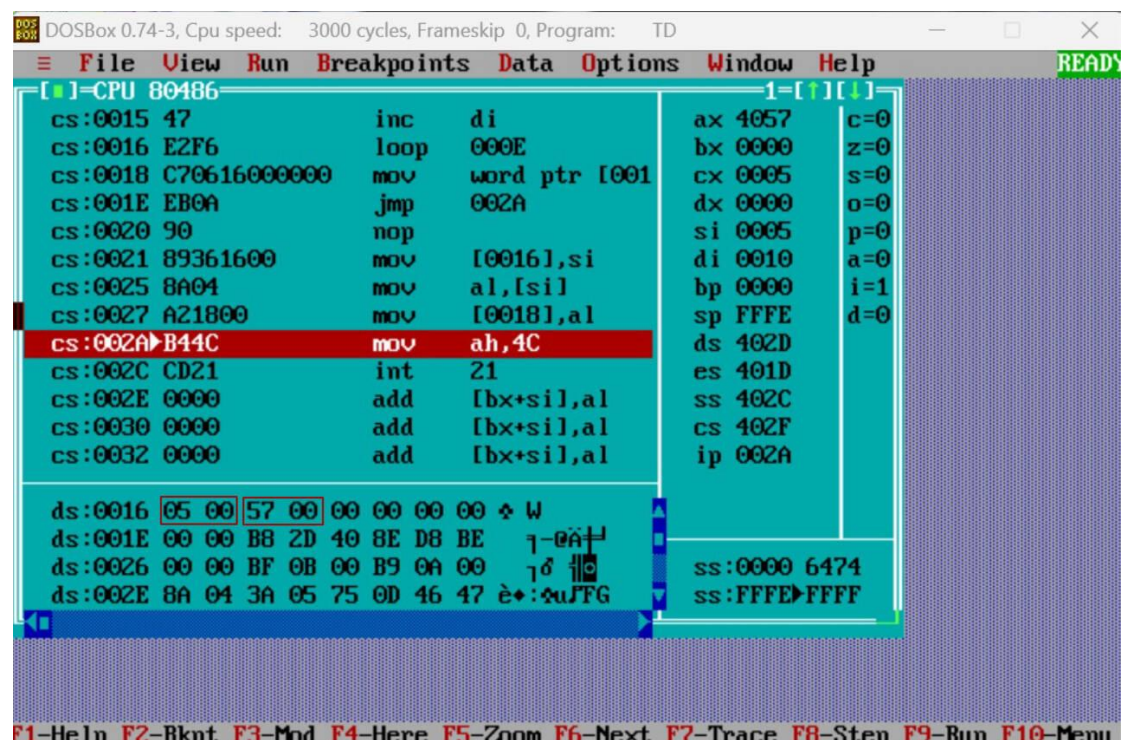
END_CODE:
```

```

MOV AH,4CH      ;功能号 4CH
INT 21H         ;类型号 21H, 利用中断向量返回 DOS
CODE ENDS      ;代码段结束
END START      ;整个程序汇编结束

```

(2) 实验结果：



RESULT1 偏移地址为[0016], 存储结果为 00 05H, 说明 STR1 中与 STR2 不相等的第一个字节的偏移地址为[0005], 与结果相符;

RESULT2 偏移地址为[0018], 存储结果为 00 57H, 说明 STR1 中与 STR2 不相等的第一个字节的 ASCII 码值为 57H (W), 与结果相符。

注：按电子版实验三指导，4 个基础任务和 2 个附加任务，实验验收了几个任务，实验报告就相应写几个任务。

二、实验总结（实验中遇到的问题、解决方法和实验收获）

如，上述实验任务的程序调试中遇到什么错误或问题，是如何查找、改正程序错误或如何解决问题的。

1.问题：查找数据时，由于在程序定义的数据地址为？，在具体执行时产生了疑惑。

解决方法：查看代码在 TD 中的具体形式，从而得到数据的偏移地址，再通过 goto 指令查找。

2.问题：对于串传送指令掌握不够全面，对于 CLD 等附加指令不清楚，且忘记将 DI 指向附加段 ES。

解决方法：温习书中关于串操作指令的知识点，适当与老师同学交流，利用微机作业反复练习。

3.我的收获：

1) 通过对汇编语言代码的设计，更深刻理解了计算机底层逻辑。

2) 通过简单的编程练习，加深了对汇编语言的理解与应用。

三、回答思考题：

请按电子版实验三指导，第 1 个思考题必答，后面五个附加思考题可选择回答部分问题或全部问题。

1. 在程序实例（求负数个数）中，“CMP BL,00”指令有何作用？“CMP BL,00”指令是否可以用其它指令代替？

答：比较指令，比较 BL 和 00H，判断 BL 正负，为后续条件转移准备。CMP 功能相当于减法，但只影响标志位。只要 BL 具体的值不影响下列程序，则可用“SUB BL, 00”代替。

附加（可选择部分问题或全部问题回答）：

2. 无符号数和有符号数比较大小时，用到的条件跳转指令有何不同？

答：无符号数：比较结果不低于用 JNB/JA，不高于用 JNA/JB；

有符号数：比较结果不小于用 JNL/JG，不大于用 JNG/JL。

3. 在程序实例（求负数个数）中，指令 LEA SI, NUM 中，源操作数是什么寻址方式？该指令可用什么指令替换？

答：源操作数为存储器直接寻址，可替换 MOV [SI], OFFSET NUM。

4. 实验任务 4 中，非压缩 BCD 码乘法和加法分别用了什么调整指令？简要说明非压缩 BCD 码乘法和加法调整指令的调整方法。并写出执行该程序进行(53348×9)的乘法运算时，第一次执行乘法的 BCD 码调整指令后的调整结果和第一次执行加法的 BCD 码调整指令后的调整结果。

答：1) 乘法用了调整指令 AAM，加法用了调整指令 AAA。

2) 乘法调整指令（AAM）调整方案：

将 AL 寄存器中的结果除以 10，所得商数即为高位十进制数置入 AH 中，所得余数即为低位十进制数置入 AL 中。

加法调整指令（AAA）调整方案：

① 若 AL 的低四位是 0 至 9，且 AF=0，则 AL 的高四位清零；

② 若 AL 的低四位是 0AH 至 0FH，或 AF=1，则 AL 进行加 6 调整，AH 加 1，AL 的高四位为 0。

3) 第一次乘法调整指令后的结果，AX=0702H，第一次执行加法调整指令后，因为无进位，AX 仍为 0702H，但第二次执行 AAA 后，AX=0403H。

5.请说明在串操作时，方向标志 DF 的作用，并分别写出 DF 清零和置 1 的指令。

答：1)方向标志 DF 决定串处理的方向：DF=0 时，地址指针 SI 和 DI 增量 (+1 或+2)；DF=1 时，地址指针 SI 和 DI 减量 (-1 或-2)。如果从串的首地址开始操作，则置 DF=0,如果从末地址开始操作，则置 DF=1。

2)CLD ;DF 置 0

STD ;DF 置 1

注：实验报告可提交 word 或 pdf 文档，实验报告的文件名格式：学号-姓名-实验三报告。