一、 实验目的

1、熟练掌握系统功能调用的语法，特别是输出一个字符串，等待一个按键，输出一个字符等常用的系统功能调用。
2、能够在输入一个按键的基础上实现输入一个字符串的操作。
3、熟练掌握分支和循环结构程序的设计，各种条件跳转指令。
4、熟练掌握各种串操作，以及串操作与 SI，DI 的联系，并能够熟练使用 rep 进行重复比较，清晰各种情况对标志位的影响。
5、熟练掌握基址变址寻址。
6、综合应用，在前 4 点的基础上，能够实现与控制台交互的程序的设计。

二、 实验设备

操作系统：Window10 个人 PC
软件环境：emu 8086 模拟器

三、 实验内容和要求

1、输入两串字符串 string1 和 string2，并比较两个字符串是否相等，相等就显示“match”，否则显示“no match”。 要求用两种实现方法。

2、

```
CONAME    DB    'SPACE EXPLORERS INC'
PRLINE    DB    20 DUP(' ')
```

用串指令编写程序段分别完成以下功能：

(1) 从左到右把 CONAME 中的字符串传送到 PRLINE。
(2) 从右到左把 CONAME 中的字符串传送到 PRLINE。
(3) 把 CONAME 中的第 3 和第 4 个字节装入 AX。
(4) 把 AX 寄存器的内容存入从 PRLINE+5 开始的字节中。
(5) 检查 CONAME 字符串中有无空格字符，如有则把它传送给 BH 寄存器。

四、 实验步骤

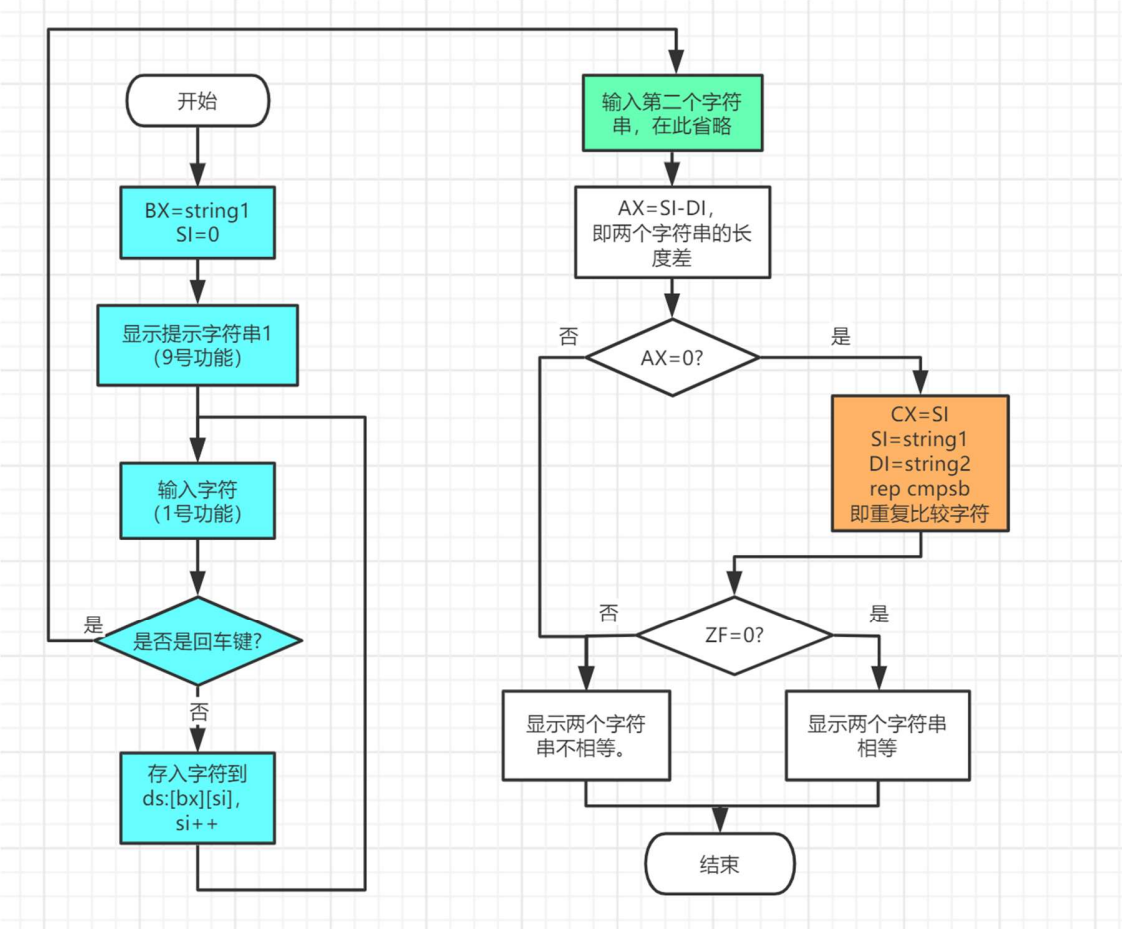
1、实验 1
实验设计思路：因为程序已经稍显复杂，所以在这里引入了多模块的设计，每个模块将实现一个专一的功能。有一定的输入和输出。在这里分成输入、比较判断和显示三个模块。每个模块都有输入和输出。
输入模块设计：输入两个字符串到 string1，string2 的存储区，结束后 si=string1 的长度，di=string2 的长度。

比较模块设计：比较两个字符串的长度。若不相等，则两个字符串不相等。否则，比较两个字符串的每一个字符，如果最后出现不相等的字符，则两个字符串不相等，否则两个字符串相等。比较模块的两种方法，有传统的循环方式和使用串比较命令的 REP 重复版本。两种代码段相对于整个代码来说占了很小的一部分，所以第二种方法只粘贴替换掉的代码。这部分在代码中用红色显示。

显示模块设计：根据判断的结果，显示两个字符串是否匹配。

输入模块分析：输入时需要调用系统的键入字符的指令，然后和 0dh(Enter key)进行比较，如果是回车，则结束输入，否则，重复调用系统输入字符的指令，等待下一次输入。

流程图：其中，蓝色和蓝绿色会输入模块，橙色为比较模块（只放了方法 1）



实验代码：

```
; multi-segment executable file template.
data segment
    ; strings
    pkey db "press any key...$"
    tip1 db "please input string1:$"
```

```
    tip2 db "please input string2:$"

    enter_key db 0dh, 0ah, "$"
    match db "match$"
    no_match db "no match$"
    ; datas
    max_len equ 120
    string1 max_len dup(?)
    len1 db 0
    len2 db 0
ends

extra segment
    string2 max_len dup(?)
ends

stack segment
    dw   128  dup(0)
ends

code segment
start:
; set segment registers:
    mov ax, data
    mov ds, ax
    mov ax, extra
    mov es, ax

    ; 初始化
    mov cx, 0
    mov si, 0
    mov di, 0

    ; module:输入模块
    lea dx, tip1
    mov ah, 9
    int 21h
    lea bx, string1 ;基址
input1_loop:
    ; 键入一个字符
    mov ah, 1
    int 21h

    cmp al, 0dh ;判断是否为enter
    jz input2
```

```
    cmp al, 08h ;判断是否是退格键
    jz back1
    mov ds:[bx][si], al
    inc si
    jmp input1_loop
back1:
    dec si
    jmp input1_loop


input2:
    mov dl, 0ah ;跳出循环时补足一个 LF。
    mov ah, 2
    int 21h ;调用输出字符的功能

    lea dx, tip2
    mov ah, 9
    int 21h
    lea bx, string2
input2_loop:
    mov ah, 1
    int 21h

    cmp al, 0dh
    jz input_out

    cmp al, 08h ;判断是否是退格键
    jz back2
    mov es:[bx][di],al
    inc di
    jmp input2_loop
back2:
    dec di
    jmp input2_loop
input_out:
    mov dl, 0ah ;跳出循环时补足一个 LF。
    mov ah, 2
    int 21h ;调用输出字符的功能

    ; 判断长度是否相等
    mov ax, si
    sub ax, di
    cmp ax, 0
    jnz show_noequal
```

```
    mov cx, si ; si 为比较次数
    lea si, string1
    lea di, string2
    rep cmpsb
    jz show_equal

show_noequal:
    lea dx, no_match
    jmp show_out
show_equal:
    lea dx, match
show_out:
    mov ah, 9
    int 21h

    lea dx, enter_key
    mov ah, 9
    int 21h

    lea dx, pkey
    mov ah, 9
    int 21h        ; output string at ds:dx

    ; wait for any key....
    mov ah, 1
    int 21h

    mov ax, 4c00h ; exit to operating system.
    int 21h
ends

end start ; set entry point and stop the assembler.
```

红色的代码段为串命令中的一种。

方法 2 的代码（仅不相同的部分）

```
    mov cx, si ; si 为比较次数
    lea si, string1
    lea di, string2
cmp_loop:
    cmpsb
    loop cmp_loop
```

2、实验 2

实验思路：（1）和（2）可以使用 MOVSB 以及和 SI,DI 进行配合来实现，也可以手动写循环程序，以[SI]->AL->[DI]的方式进行实现，但无论那种方式，都需要用到寄存器。（3）需要用到一次 MOV 指令，以及强制转换。（4）需要用到 STDSB，即串存储指令。（5）需要用到串扫描指令 SCASB。
由于此部分设计的关键代码很少，不需要进行复杂的流程设计，所以流程图和伪代码在此省略。
实验代码：

```
; multi-segment executable file template.

data segment
    ; strings
    pkey db "press any key...$"
    coname db "SPACE EXPLORERS INCO"
    prline db 20 dup(' ')
    len equ $-prline
ends

stack segment
    dw   128  dup(0)
ends

code segment
start:
; set segment registers:
    mov ax, data
    mov ds, ax
    mov es, ax

    ; the core code

    lea dx, pkey
    mov ah, 9
    int 21h        ; output string at ds:dx

    ; wait for any key....
    mov ah, 1
    int 21h

    mov ax, 4c00h ; exit to operating system.
    int 21h
ends
```

```
end start ; set entry point and stop the assembler.
```

（1）从左到右把 CONAME 中的字符串传送到 PRLINE。

```
    ; the core code
    lea si, coname
    lea di, prline
    mov cx, len
    rep movsb
```

或者是

```
    ; the core code
    std ; 设置 df 为 1
    mov si, coname+len-1
    mov di, prline+len-1
    mov cx, len
    rep movsb
```

（2）从右到左把 CONAME 中的字符串传送到 PRLINE。

```
    ; the core code
    lea si, coname ; si 指向 coname 的第 1 个单元
    mov di, prline+len-1 ; di 指向 prline 的最后一个单元
    mov cx, len
mov_loop:
    mov al, [si]
    mov [di], al ; 传送一个单元
    add si, 1
    sub di, 1 ; 上面两句实现偏移操作，其中 si+1,di-1
    loop mov_loop
```

（3）把 CONAME 中的第 3 和第 4 个字节装入 AX。

```
    ; the core code
    mov ax, word ptr [coname+3]
```

（4）把 AX 寄存器的内容存入从 PRLINE+5 开始的字节中。

```
    ; the core code
    mov ax, 6466h ; ax = d,f
    mov word ptr [prline+5], ax
```

（5）检查 CONAME 字符串中有无空格字符，如有则把它传送给 BH 寄存器。

```
    ; the core code
    lea di, coname  ; di 指向 coname
    mov cx, len ; 循环次数（最大）
    mov al, ' ' ; 或者为 20h
loop_cmp:
    scasb ; 将 al 与[di]比较，zf 保存比较结果，然后 di 自增
```

```
    jz equal ; 如果包含空格，直接跳到 equal 段
    loop loop_cmp
    jmp out_cmp
equal:
    ; 相等的逻辑
    lea bx, coname
out_cmp:
```

或者

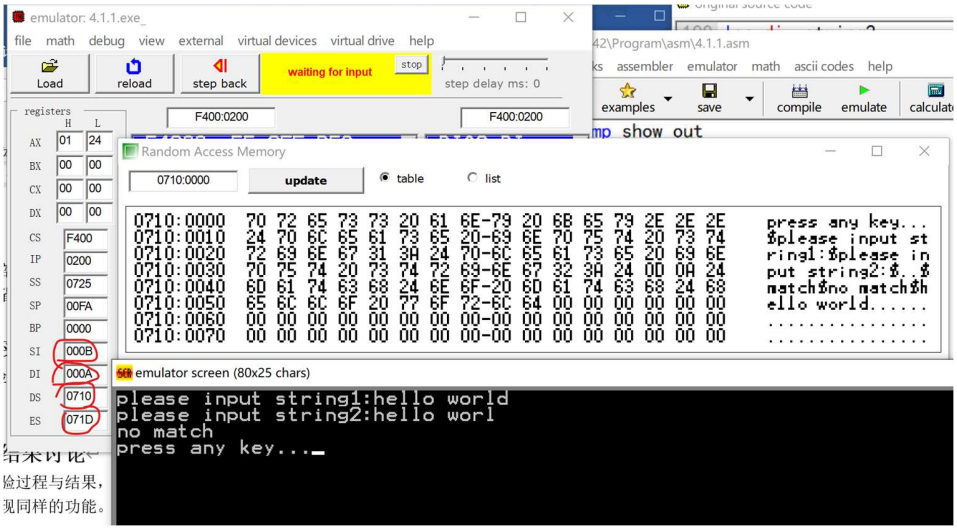
```
    ; the core code
    lea di, coname  ; di 指向 coname
    mov cx, len ; 循环次数（最大）
    mov al, ' ' ; 或者为 20h
    repnz scasb  ; 重复扫描，直到 al==[di]或者 cx=0
    jnz out_cmp  ; zf = 0 表示含有空格
    ; 相等的逻辑
    lea bx, coname

out_cmp:
```

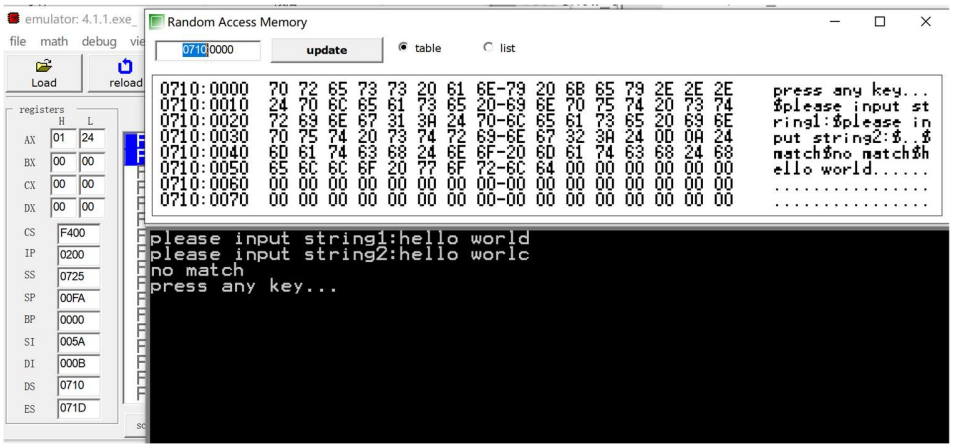
五、 实验结果分析

1、实验 1（方法 1）

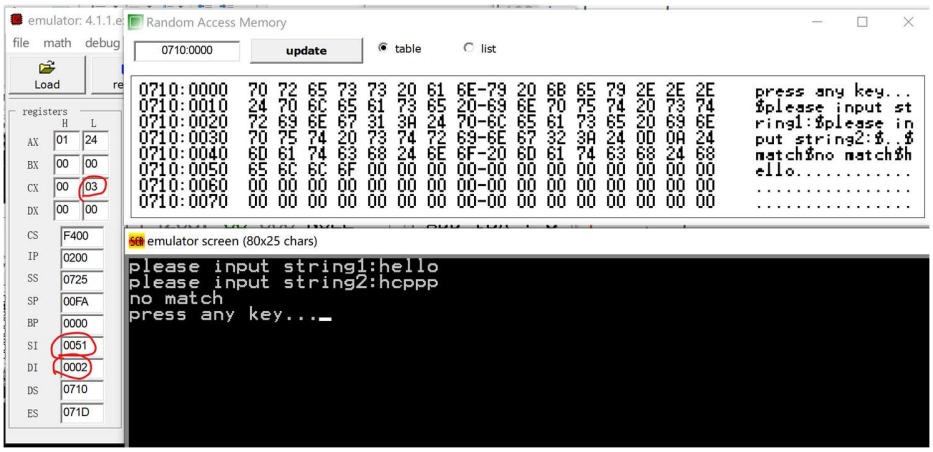
情况 1：两个字符串长度不相等，以 hello world 和 hello worl 进行测试



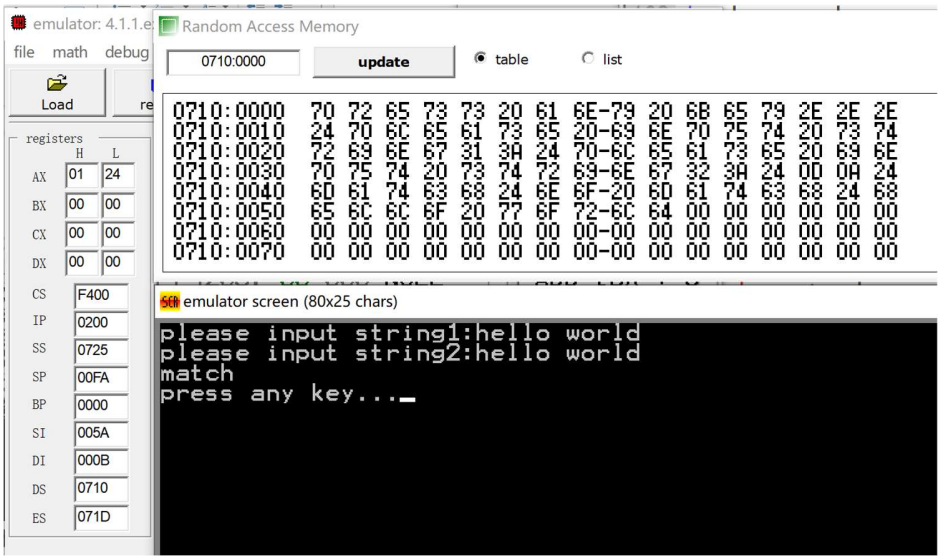
情况 2：两个字符串长度相等，但最后一个字符不相等，以 hello world 和 hello worc 进行测试。



情况 3：两个字符串长度相等，但不是情况 2，以 hello 和 hcppp 为例



情况 4：两个字符串相等，以 hello world 和 hello world 为例。

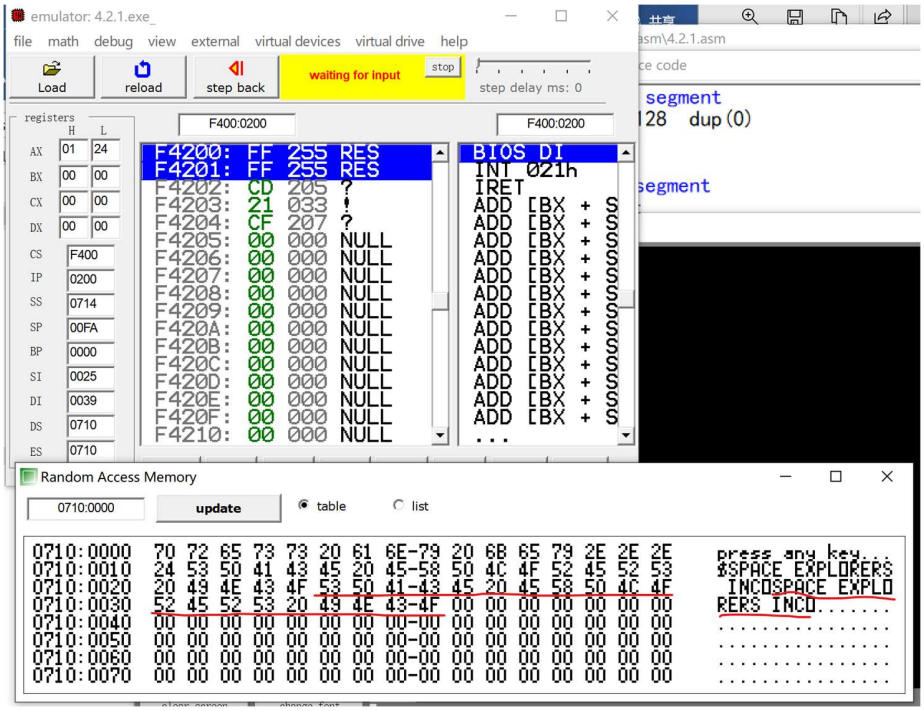


三种情况下的测试都符合程序的预期，且内存的分配以及各个寄存器的数据都正常。从结果分析可以得到，情况 2 和情况 4 的 CX=0，但是两种情况的 ZF 不一样，所以在进行判断时，应该根据 ZF 来进行条件跳转而不是 CX 来进行跳转。

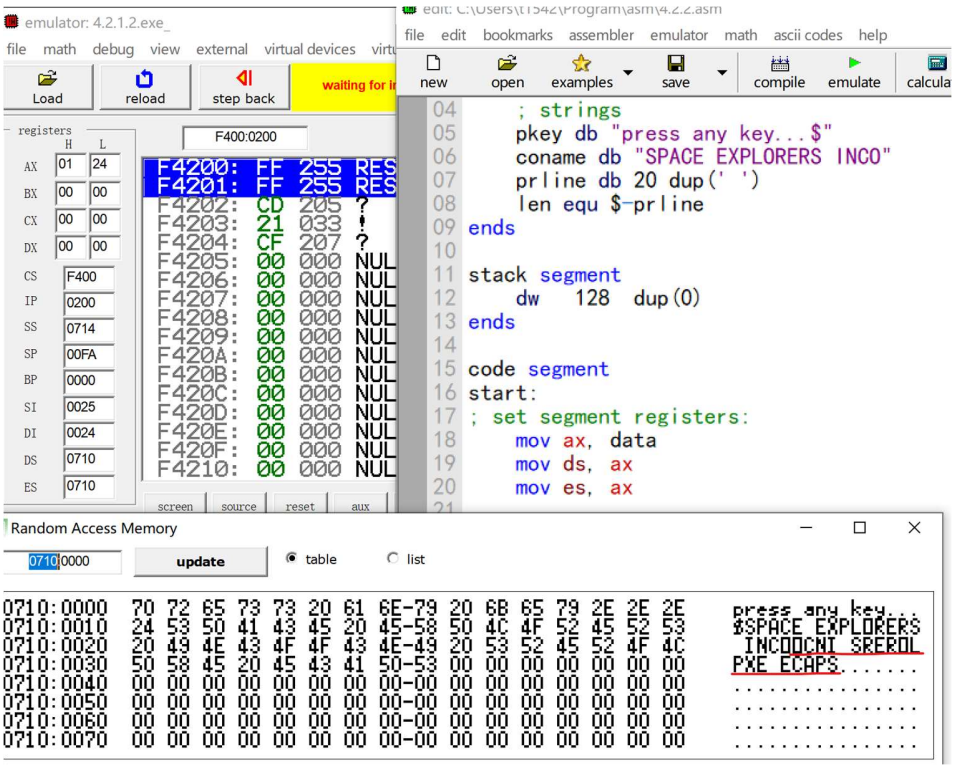
第二种方法不一样的地方仅仅在于比较字符串的地方，经验证，两种情况的结果相同，因此不再粘贴第二种方法的实验结果。

2、实验二

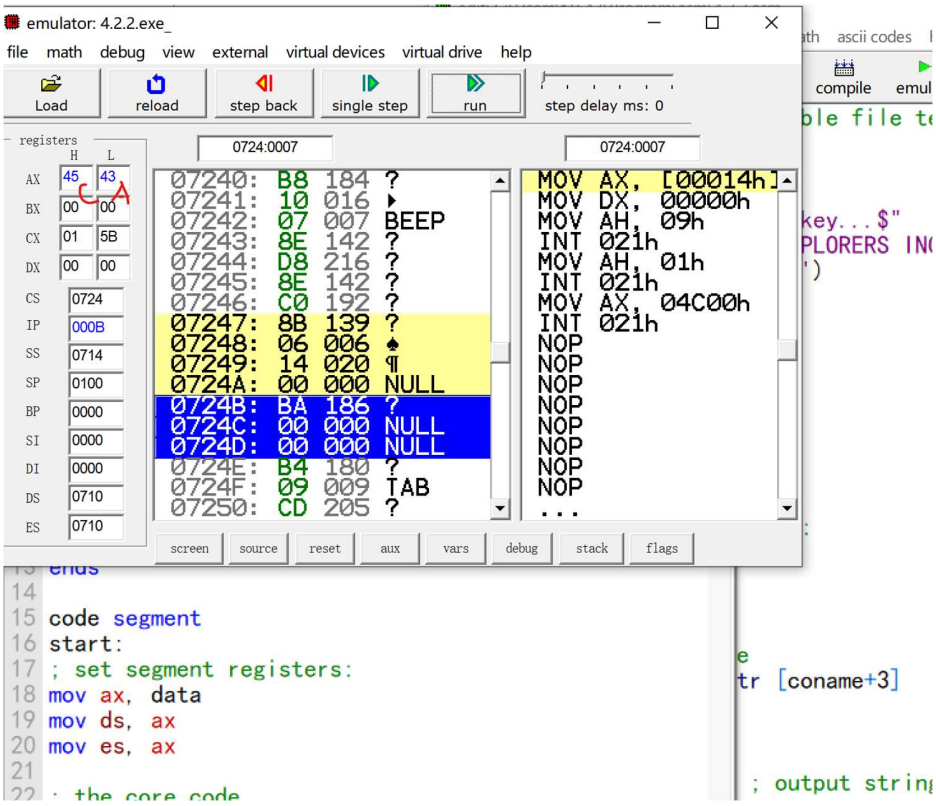
（1）从左到右把 CONAME 中的字符串传送到 PRLINE。



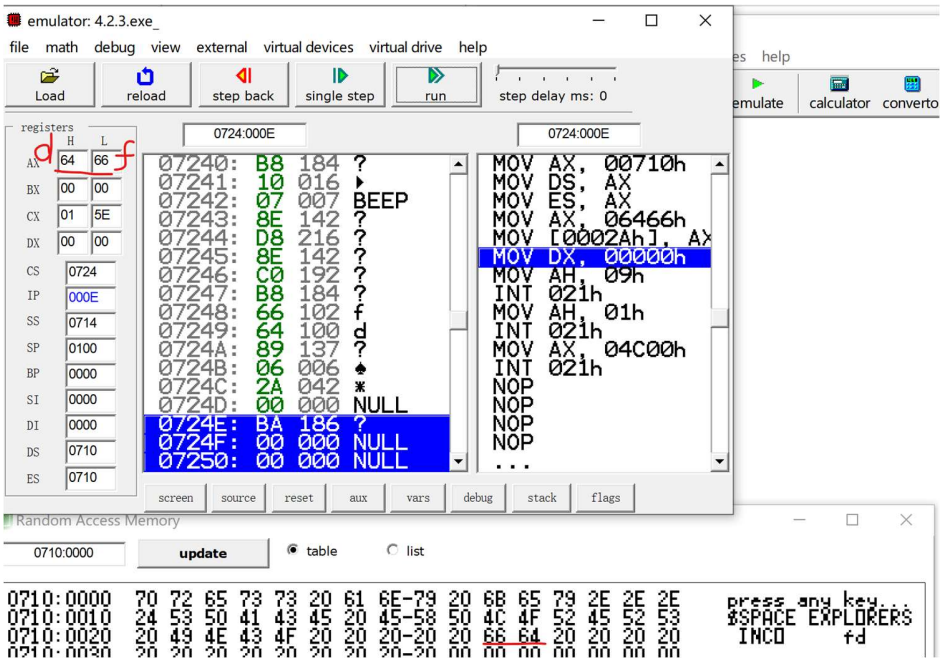
（2）从右到左把 CONAME 中的字符串传送到 PRLINE。



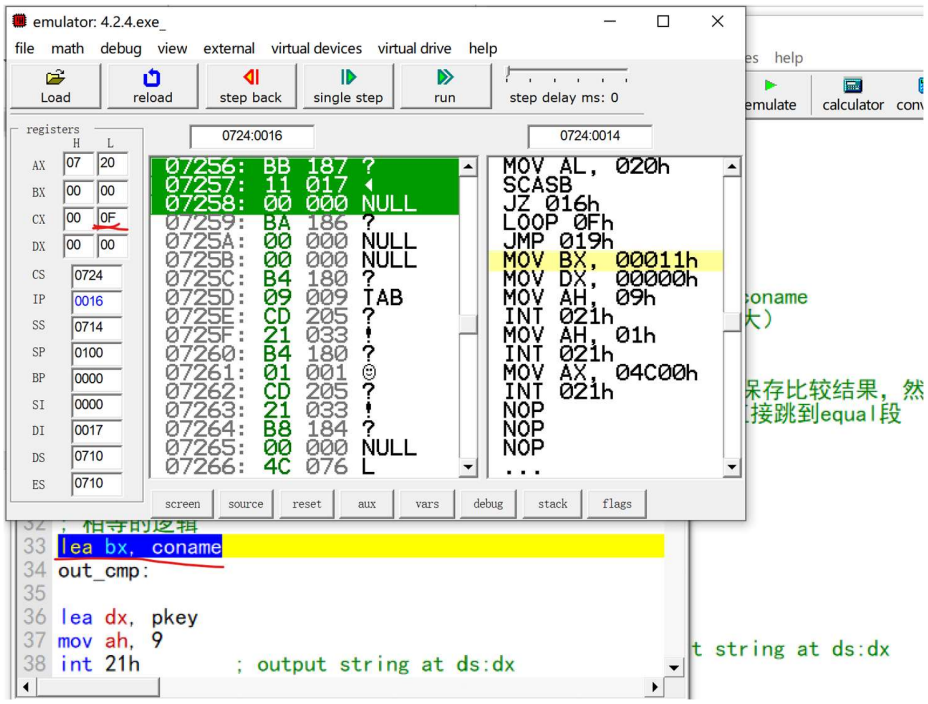
（3）把 CONAME 中的第 3 和第 4 个字节装入 AX。



（4）把 AX 寄存器的内容存入从 PRLINE+5 开始的字节中。



（5）检查 CONAME 字符串中有无空格字符，如有则把它传送给 BH 寄存器。



14h-0fh=05h，在 index=5 处为空格，符合预期。

六、 结果讨论

1、在第一个实验中，处理输入和输出的代码占了较大的比重。且多次使用到了调用系统功能的函数，在实验中，用到了显示一个字符，显示一串字符，输入一个字符等各个操作。可以在调用系统的输入一个字符的功能上拓展成输入一个字符串的

功能。在本实验中，以 Enter 键的输入作为输入字符串结束的标志。通过本实验知道，输入输出在汇编程序中需要复杂的代码，所以使用中断程序调用和子程序调用也成了一种需要。

2、在调用系统 1 号功能（即输入一个字符的功能）时，焦点会跑到控制台，在键入 1 个字符后，会在控制台打印该字符，并将该字符传到 AL。在系统中，有两个关键的字符 0AH(LF，line feed，new line)和 0DH(CR,carriage return)。在本模拟器中，0DH 表示将光标移到该行的开头处。0AH 表示将光标移到下一行，但列位置不变。这两个字符为控制字符，起源于行列打字器的控制输出。因此要按回车后到下一行还要追加一个 LF 字符。（不同的文件系统有不同的行尾序列，有的为 CRLF，有的为 LF。所以要按照不同的系统编写不同的换行处理指令）

3、在实验 1 中输入两个字符串写了两段相似的代码，在高级程序中，常常编写一个函数来实现相似的功能，在汇编中也是类似的，但是由于没有学过 call 语句，所以设计子函数有一定的难度。这也是这个程序需要改进的地方。

4、在串指令中，如果涉及到两个内存区块间的数据传递，必须使用到中间的寄存器（一般为 AL 和 AX），否则违反了不可以在两个内存区块传送数据的规定。

5、REP 指令后只可以接一条指令，其表示的含义是，执行指令，直到 CX=0 或者 ZF=1。与其相对的一条指令为 REPNZ，其含义为执行指令，直到 CX=0 或者 ZF=1，这条指令可以用来做第 2（5）题。（REPNZ 可以使用 repeat if not zero）来进行记忆。

6、在比较两个长度相等字符串是否相等的操作时（或者类似的操作），并不是依照 CX 是否等于 0 来段两个字符串是否相等，而应该依照 ZF 标志位来判断。反例为两个字符串只有最后一个字符不等时，CX=0，ZF=1。不相等。

7、一块内存的空间总是有限的，本实现将每个字符串的最大长度设置为 120，没有溢出检查的判断，因此，输入过长的字符串时容易导致内存边界的溢出现象。但是这种情况出现的几率比较小，因而并没有添加该功能。

实验的收获：

通过本实验，让之前相对模糊的概念通过实验有了更加精确的了解。特别是之前不太使用的 REP 前缀，通过实验和单步调试分析出了内部处理的逻辑。能够熟练在传统的 LOOP 和 REP（仅限能够转换的语句）之间进行转换。对相对复杂（指相对之前简单的代码）有了较为系统的设计方案，能够通过单步调试发现代码中存在的细小问题进行更正。