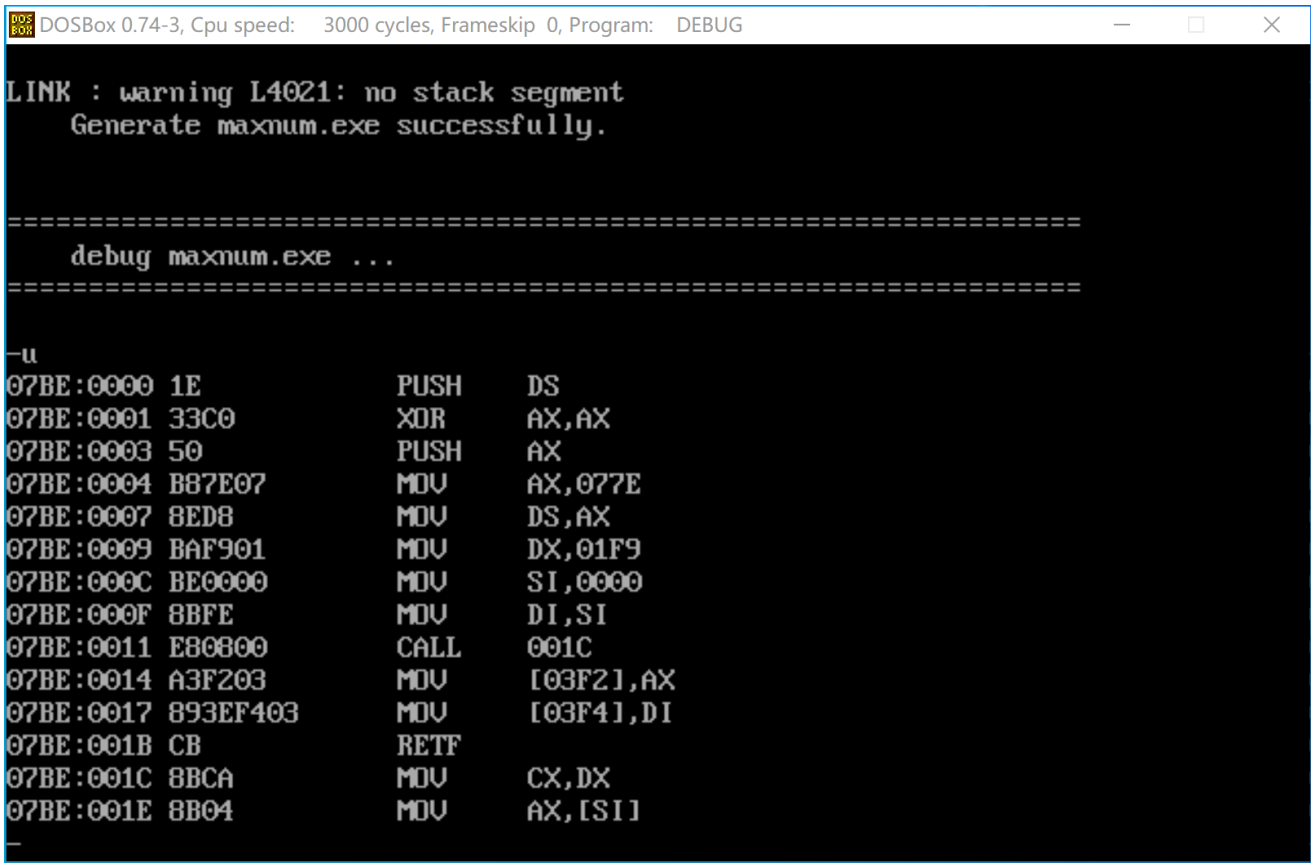
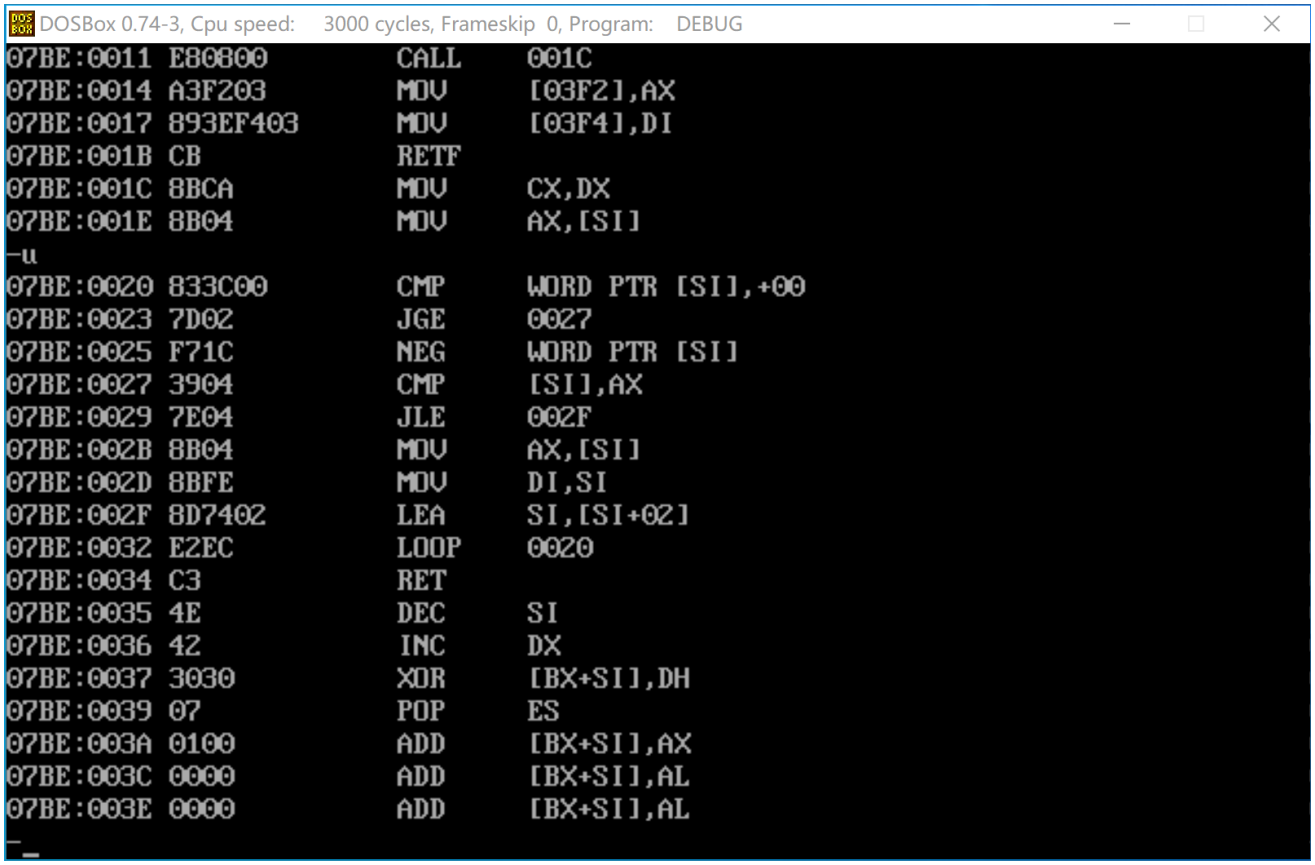
汇编语言第2次上机

班级	学号	姓名
计算机2205	2204112913	李雨轩

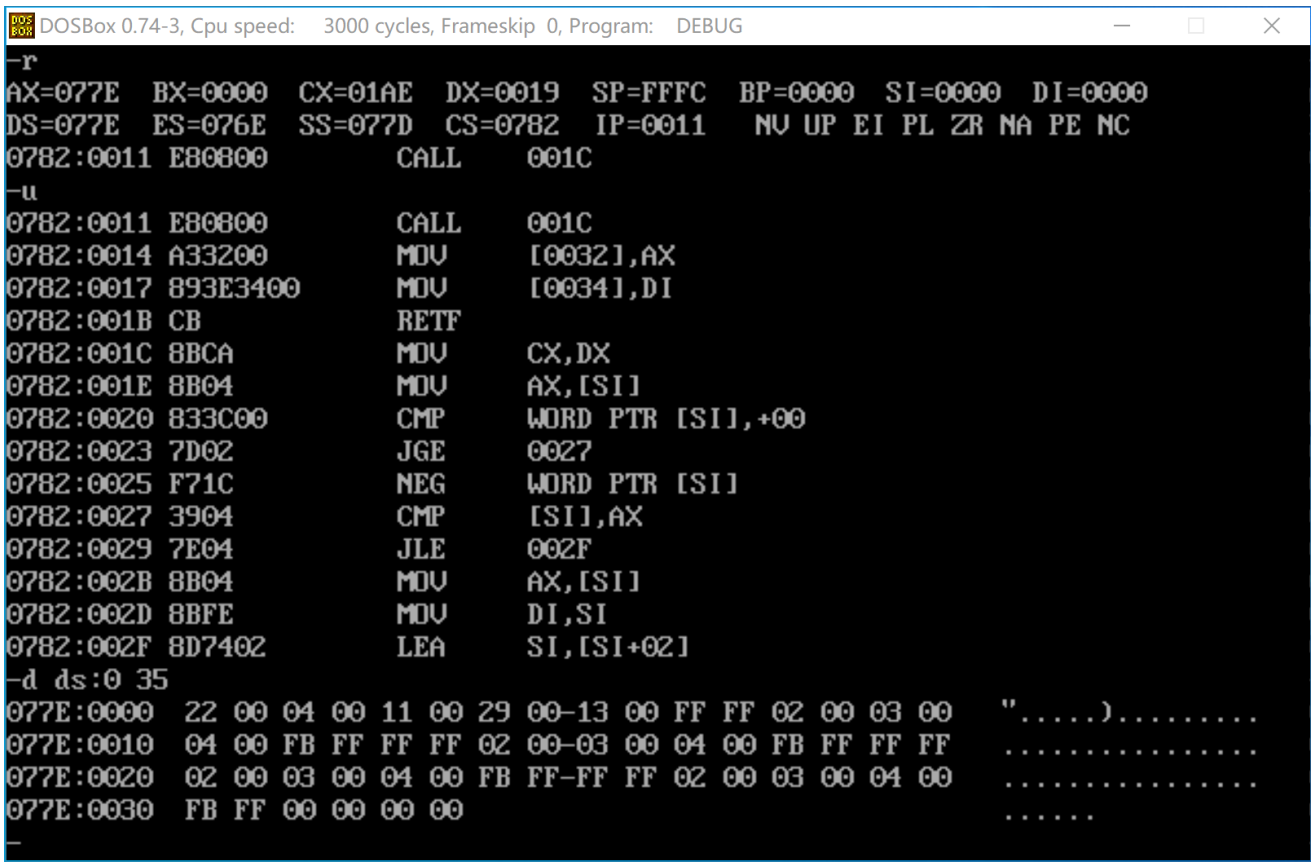
1. 循环程序设计

(1). 反汇编的截图

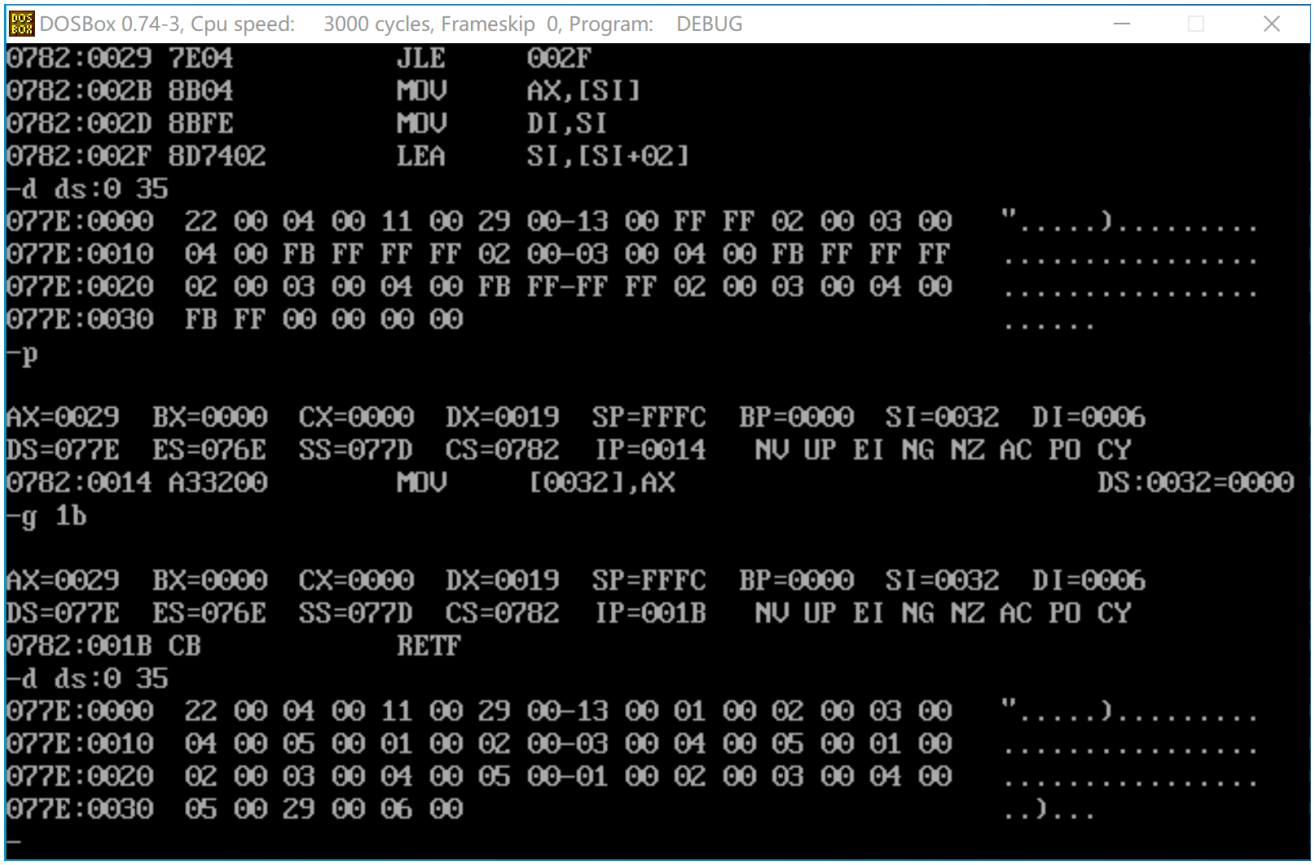




(2). 在进行计算前，显示数组M开始的n+2个字的内存值的截图（只能显示这n+2个字的内存值，多显示、少显示均扣分）



(3). 执行完计算后，显示数组M开始的n+2个字的内存值的截图（只能显示这n+2个字的内存值，多显示、少显示均扣分）



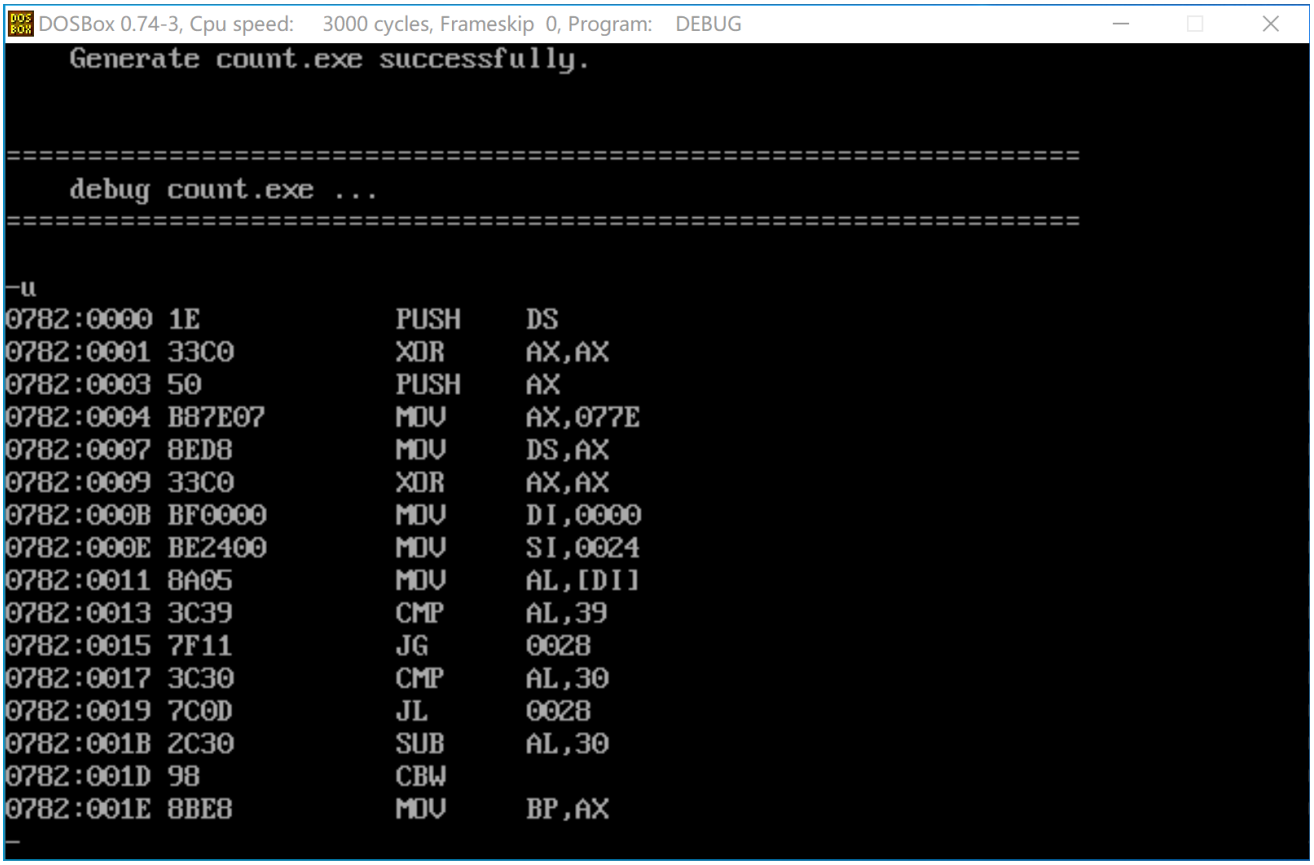
(4). 源代码

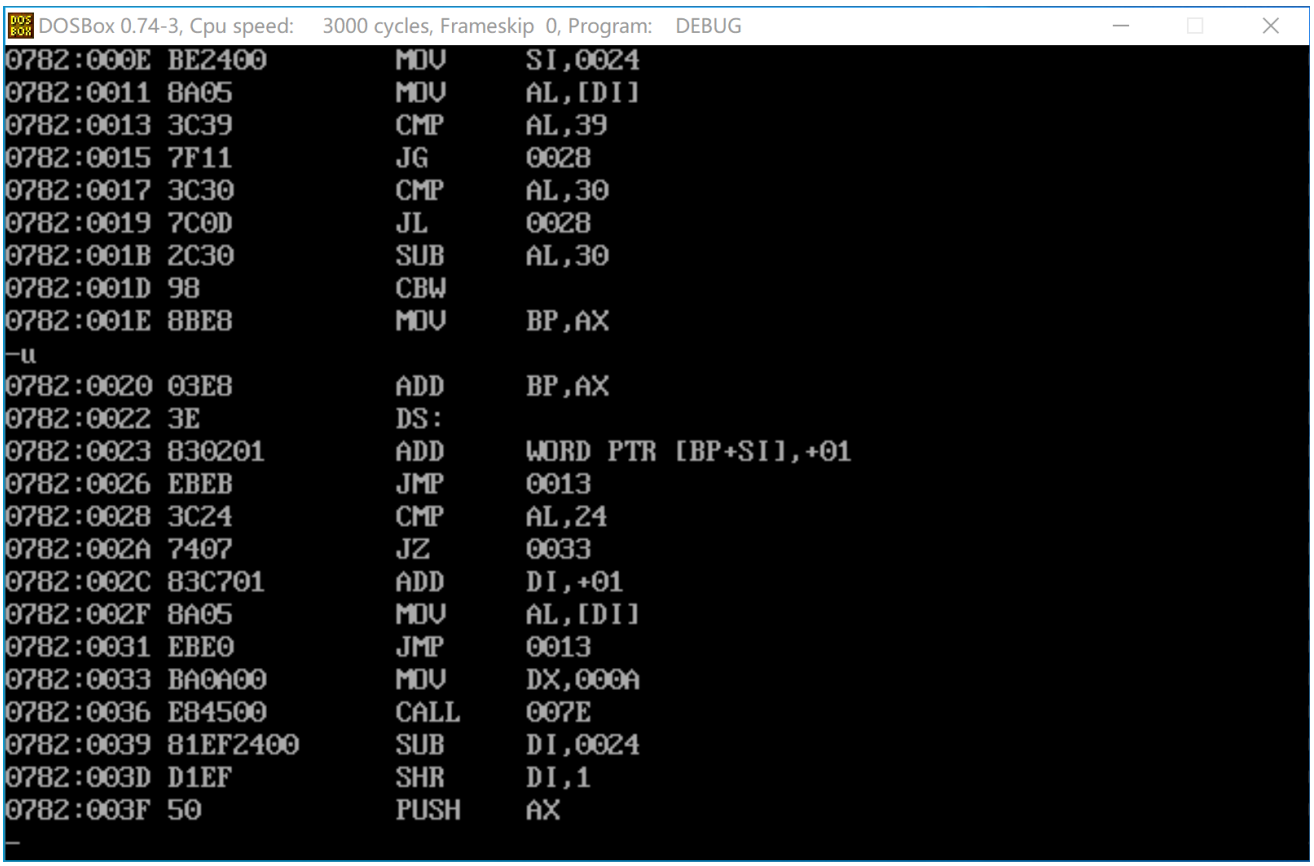
```
name MaxNumber
title Find Max Number

data segment
;    length dw 16
     array label word
     dw 22h, 04h, 11h, 29h, 13h
     dw 4  dup(-1,2,3,4,-5)
     arrend label word
     max dw ?
     ofs dw ? ;store the first max number
data ends


code segment
     assume cs:code, ds:data

     main proc far
          push ds
          xor ax, ax
          push ax
          mov ax, data
```

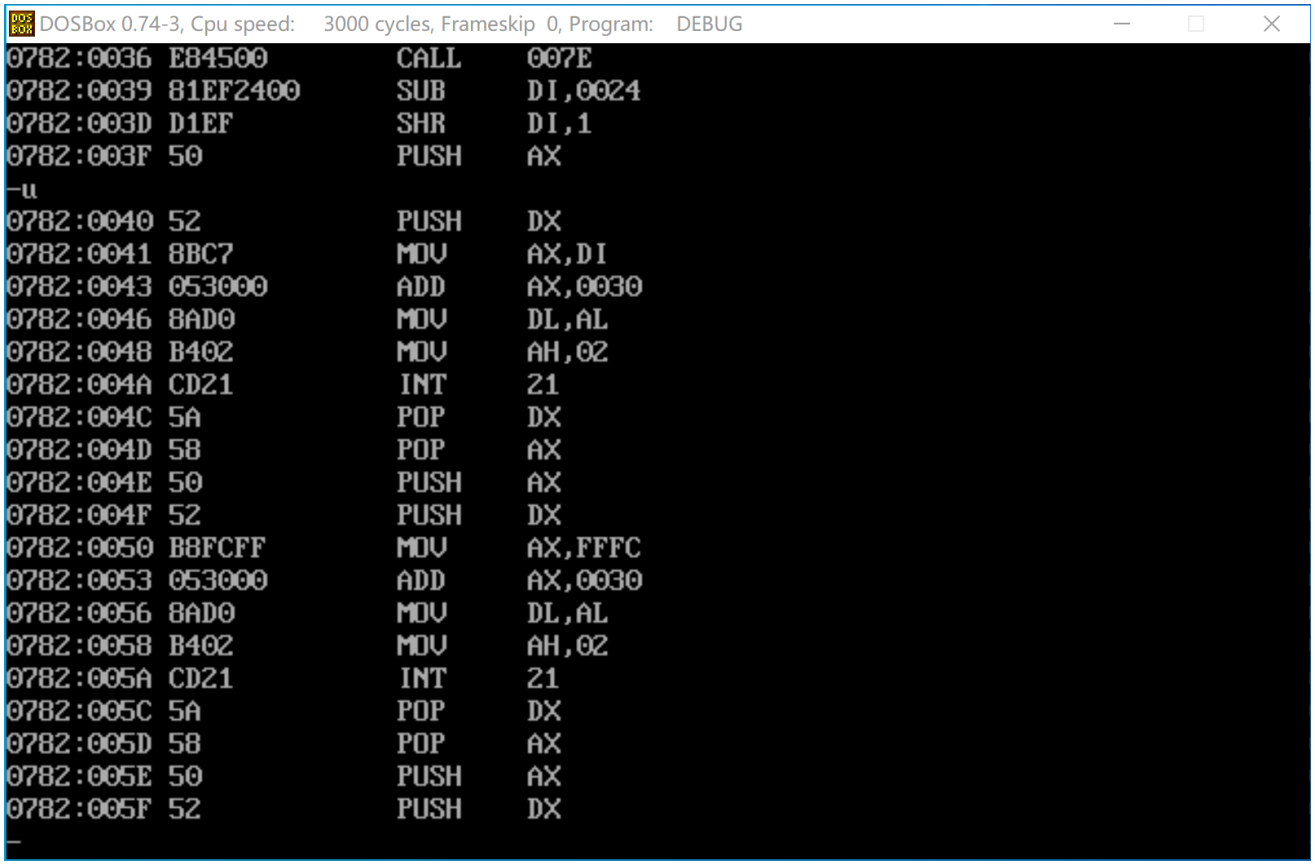
```
        mov ds, ax
        mov dx, (arrend - array)/2; get array's length
        mov si, offset array
        mov di, si
        call findMax
        mov max, ax
        mov ofs, di

        ret
    main endp

    findMax proc near
    ; dx - length, si - array, di - max_index
    ; return the max num in ax

        mov cx, dx
        mov ax, [si]; ax stores the max number

loopH:  cmp word ptr [si], 0;
        jnl short whennl
        neg word ptr [si]
whennl: cmp [si], ax
        jng short whenng
        mov ax, [si]
        mov di, si
whenng: lea si, 2[si]
        loop loopH

        ret
    findMax endp

code ends
    end main
```

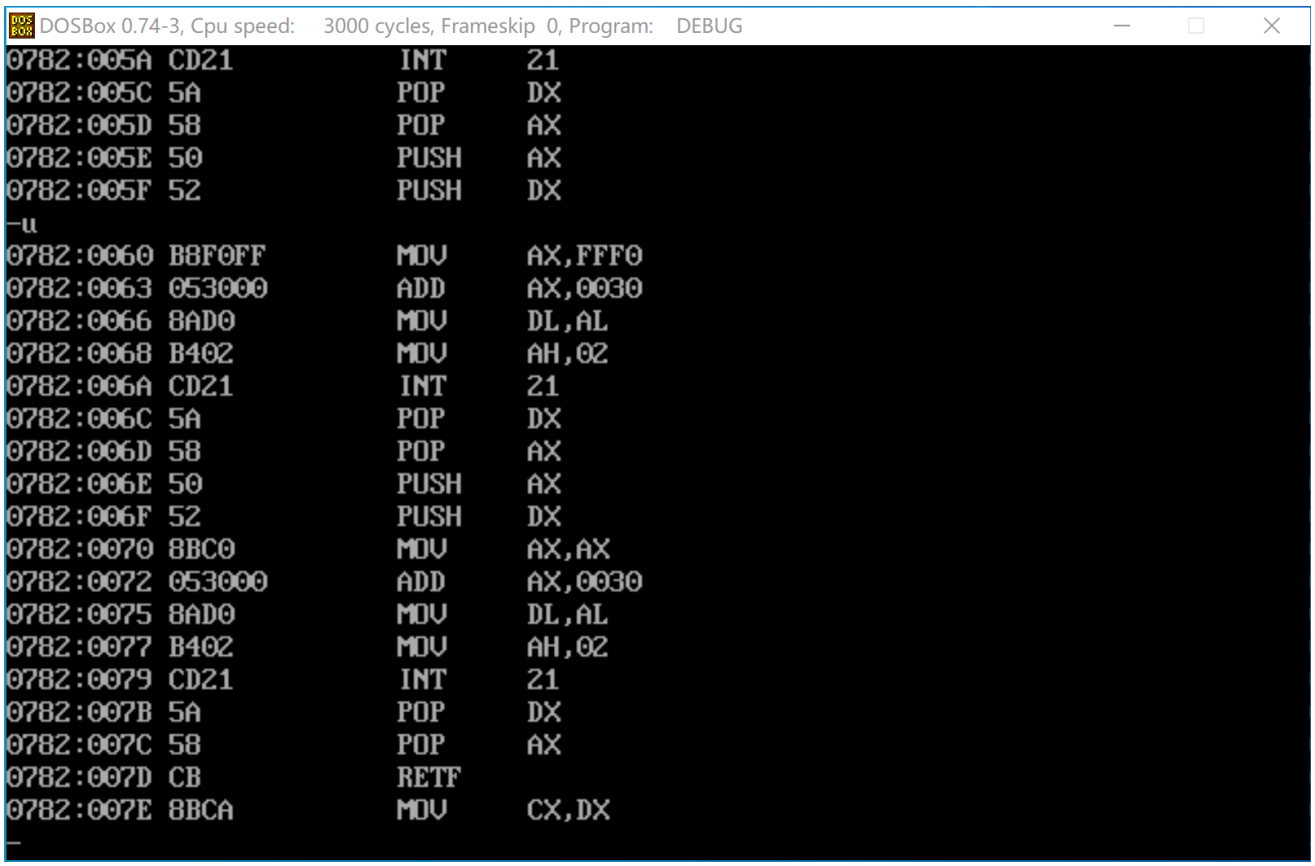
2. 分支程序设计

(1). 反汇编的截图

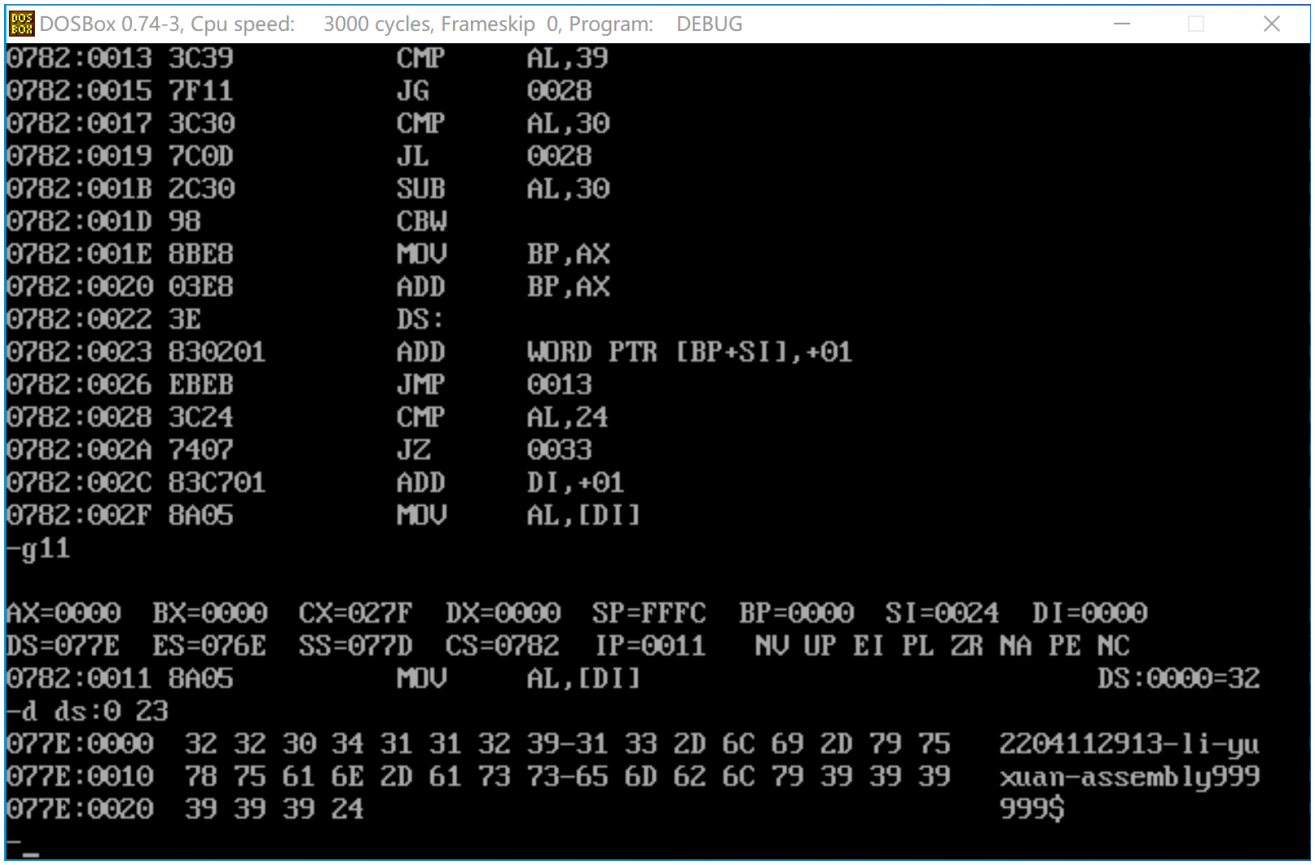

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
Generate count.exe successfully.
debug count.exe ...
-u
0782:0000 1E PUSH DS
0782:0001 33C0 XOR AX,AX
0782:0003 50 PUSH AX
0782:0004 B87E07 MOV AX,077E
0782:0007 8ED8 MOV DS,AX
0782:0009 33C0 XOR AX,AX
0782:000B BF0000 MOV DI,0000
0782:000E BE2400 MOV SI,0024
0782:0011 8A05 MOV AL,[DI]
0782:0013 3C39 CMP AL,39
0782:0015 7F11 JG 0028
0782:0017 3C30 CMP AL,30
0782:0019 7C0D JL 0028
0782:001B 2C30 SUB AL,30
0782:001D 98 CBW
0782:001E 8BE8 MOV BP,AX


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
0782:000E BE2400 MOV SI,0024
0782:0011 8A05 MOV AL,[DI]
0782:0013 3C39 CMP AL,39
0782:0015 7F11 JG 0028
0782:0017 3C30 CMP AL,30
0782:0019 7C0D JL 0028
0782:001B 2C30 SUB AL,30
0782:001D 98 CBW
0782:001E 8BE8 MOV BP,AX
-u
0782:0020 03E8 ADD BP,AX
0782:0022 3E DS:
0782:0023 830201 ADD WORD PTR [BP+SI],+01
0782:0026 EBEB JMP 0013
0782:0028 3C24 CMP AL,24
0782:002A 7407 JZ 0033
0782:002C 83C701 ADD DI,+01
0782:002F 8A05 MOV AL,[DI]
0782:0031 EBE0 JMP 0013
0782:0033 BA0A00 MOV DX,000A
0782:0036 E84500 CALL 007E
0782:0039 81EF2400 SUB DI,0024
0782:003D D1EF SHR DI,1
0782:003F 50 PUSH AX

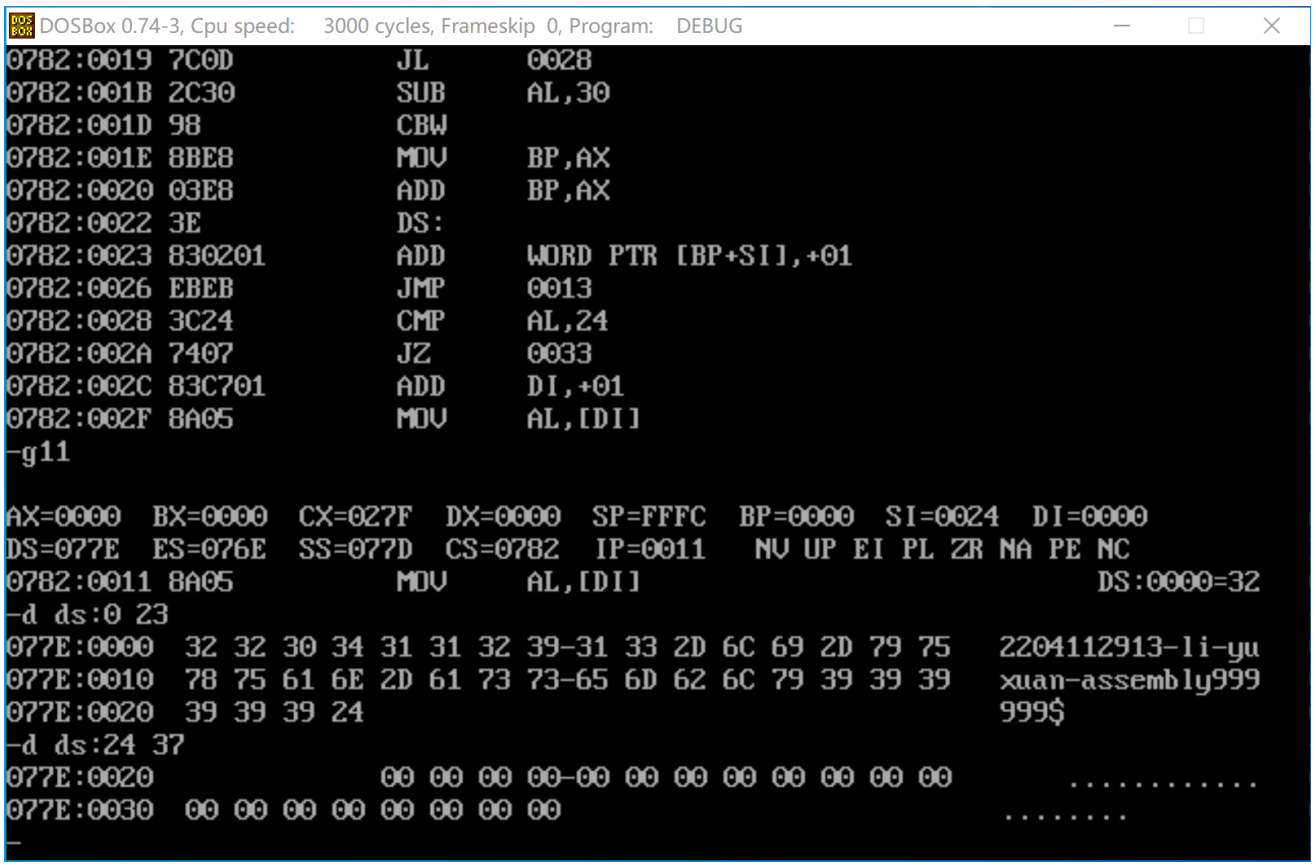




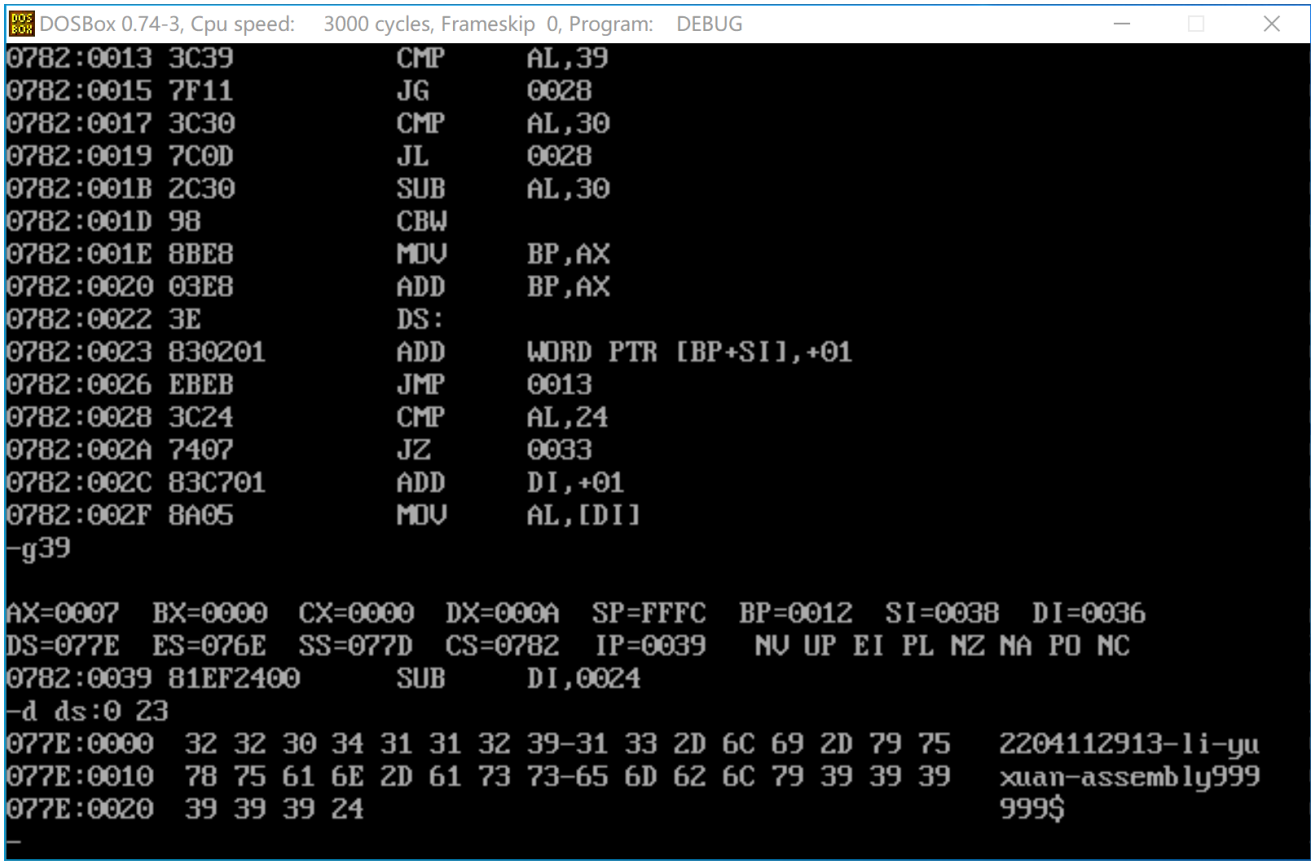
(2). 在进行计算前，显示在数据段中定义的含学号的字符串的内存值的截图（只能显示该完整的字符串，多显示、少显示均扣分）



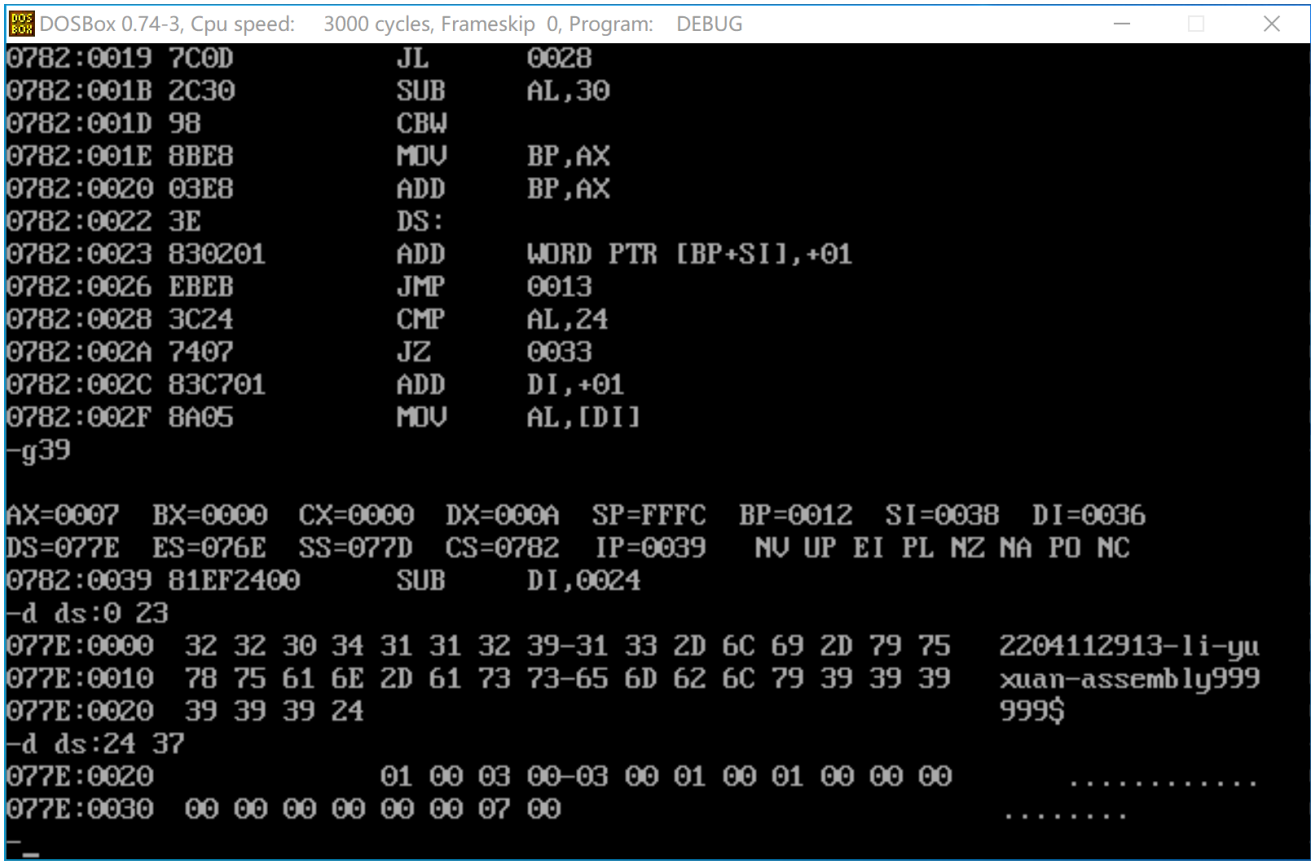
(3). 在进行计算前，显示在数据段中定义的COUNT数组的内存值的截图（只能显示完整的COUNT数组内容，多显示、少显示均扣分）



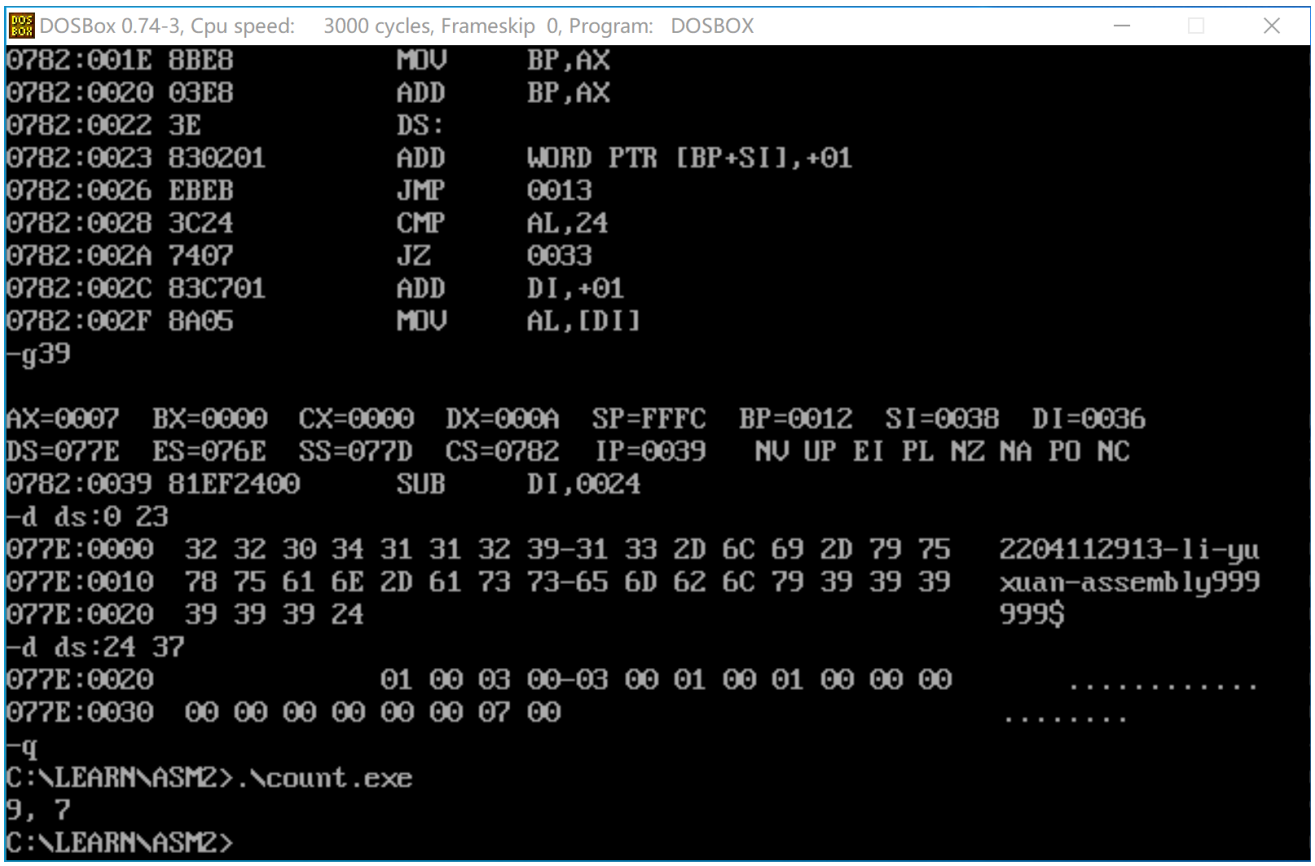
(4). 执行完计算后，显示在数据段中定义的含学号的字符串的内存值的截图（只能显示该完整的字符串，多显示、少显示均扣分）



(5). 执行完计算后，显示在数据段中定义的COUNT数组的内存值的截图（只能显示完整的COUNT数组内容，多显示、少显示均扣分）



(6). 程序在DOSBox下直接运行的截图



(7). 源代码

```
printnum macro num
    push ax
    push dx
    mov ax, num
    add ax, 30h
    mov dl, al
    mov ah, 02h
    int 21h
    pop dx
    pop ax
endm

name CountString
title Count String

data segment
    mystring db '2204112913-li-yuxuan-assembly999999$'
    countarray dw 10 dup(0)
    tests db 9
data ends


code segment
    assume cs:code, ds:data

    main proc far
        ; init
        push ds
        xor ax, ax
        push ax
        mov ax, data
        mov ds, ax
        xor ax, ax

        ; count number
        mov di, offset mystring
        mov si, offset countarray
        mov al, ds:[di]

begin:  cmp al, 39h
        jg short incr
        cmp al, 30h
        jl short incr
        sub al, 30h

        cbw
        mov bp, ax
        add bp, ax
        add word ptr ds:[bp+si], 1
```

```
        jmp short begin

incr:   cmp al, '$'
        je short endstr
        add di, type mystring
        mov al, ds:[di]
        jmp short begin

endstr: mov dx, length countarray

        call findMax
        sub di, offset countarray
        shr di, 1
        printnum di
        printnum ','-30h
        printnum ' '-30h
        printnum ax

        ret
    main endp


    findMax proc near
    ; dx - length, si - array, di - max_index
    ; return the max num in ax

        mov cx, dx  ; length
        mov ax, [si]; ax stores the max number

loopH:  cmp word ptr [si], 0;
        jnl short whennl
        neg word ptr [si]
whennl: cmp [si], ax
        jl short whenl
        mov ax, [si]
        mov di, si
whenl:  lea si, 2[si]
        loop loopH

        ret
    findMax endp



code ends
    end main
```