

中山大學



微机实验

题 目：	作业3
上课时间：	第16-17周
授课教师：	何涛
姓 名：	周德峰
学 号：	21312210
日 期：	2023-6-10

题一

设数据段中有 10 个字节的无符号数(每个数 8 位)。要求统计这些 数中高电平(即 1)与低电平(即 0)的个数。将数据段中所有数高电平的 个数和与低电平的个数和分别存入 3000H 与 3100H 中。

要求：使用子程序完成 (样例: 51H, 3AH, 95H, 8DH, 90H, 0A7H, 0C1H, 77H, 24H, 0B1H)

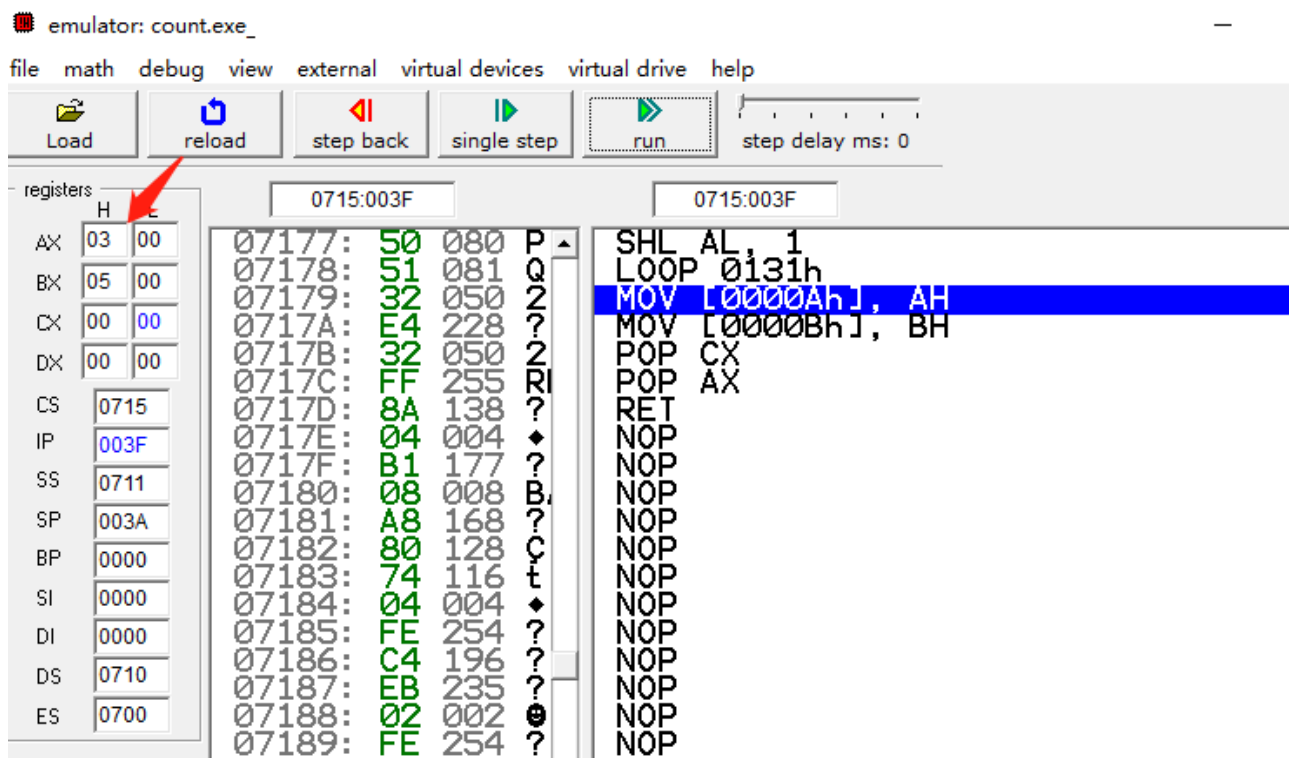
1 实验思路

在主程序中循环10次，每次将一个字节放入子程序中，计算其对应的高电平数与低电平数，然后将其累加到AH和AL中，AH存放高电平，AL存放低电平

2 实验结果

验证第一个数

当第一次循环时，将51H（0101 0001B）送入子程序中



当CX=0时，对应的AH=3，AL=5，符号结果

对应高电平个数为 $2 \times 16 + 5 = 37$ 个

Random Access Memory																			
0710:3000		update		<input checked="" type="radio"/> table		<input type="radio"/> list													
0710:3000	25	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0710:3010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0710:3020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0710:3030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0710:3040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0710:3050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0710:3060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0710:3070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

对应低电平个数为 $2*16+11=43$ 个

Random Access Memory																			
0710:3100		update		<input checked="" type="radio"/> table		<input type="radio"/> list													
0710:3100	28	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0710:3110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0710:3120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0710:3130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0710:3140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0710:3150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0710:3160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0710:3170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

总和刚好为80个，进一步验证程序的正确性

count.asm源码：

```

1  DATA SEGMENT
2      X DB 51H,3AH,95H,8DH,90H,0A7H,0C1H,77H,24H,0B1H
3      COUNT_HIGH DB 0
4      COUNT_LOW DB 0
5  DATA ENDS
6
7  STACK SEGMENT PARA  STACK
8      DW 20H DUP(0)
9
10 STACK ENDS
11
12
13 CODE SEGMENT
14     ASSUME DS:DATA,CS:CODE
15 START: MOV AX, DATA
16         MOV DS, AX

```

```

17         XOR AH,AH      ;CLEAR
18         XOR AL,AL
19         MOV CL,10      ; LOOP 10 TIMES
20         LEA SI,X
21 COUNT:
22         CALL COUNT_BITS ;调用子程序
23         ADD AH,[COUNT_HIGH]
24
25         ADD AL,[COUNT_LOW]
26         INC SI
27         LOOP COUNT
28         MOV [3000H],AH
29         MOV [3100H],AL
30         MOV AH,4CH
31         INT 21H
32 COUNT_BITS PROC
33         PUSH AX
34         PUSH CX
35         XOR AH, AH      ; 将AH寄存器清零, 用于存储高电平计数
36         XOR BH, BH      ; 将BH寄存器清零, 用于存储低电平计数
37         MOV AL, [SI]    ; 读取数据段中的字节
38         MOV CL,8
39 SHL_LOOP:
40         TEST AL, 80H    ; 检查字节的最高位是否为1
41         JZ LOW_BIT      ; 如果最高位为0, 跳转到LOW_BIT标签
42
43 HIGH_BIT:
44         INC AH          ; 如果最高位为1, 增加高电平计数
45         JMP NEXT        ; 跳转到NEXT标签
46
47 LOW_BIT:
48         INC BH          ; 如果最高位为0, 增加低电平计数
49
50 NEXT:
51         SHL AL, 1
52         LOOP SHL_LOOP
53
54 BIT_END:
55         ;INC SI          ; 增加SI寄存器的值, 指向下一个字节
56         ;LOOP COUNT_LOOP ; 循环COUNT_LOOP标签, 直到CX寄存器为0
57         MOV [COUNT_HIGH], AH ; 将高电平计数存储到COUNT_HIGH变量
58         MOV [COUNT_LOW], BH  ; 将低电平计数存储到COUNT_LOW变量
59         POP CX
60         POP AX
61         RET
62

```

```
63 | COUNT_BITS ENDP
64 |
65 | CODE ENDS
66 | END START
```

题二

设数据段中初始存放了 10 个字节的无符号数，要求使用冒泡排序法 将这些数从小到大排序。排序完成后放在以 3000H 为初始单元的地址中。

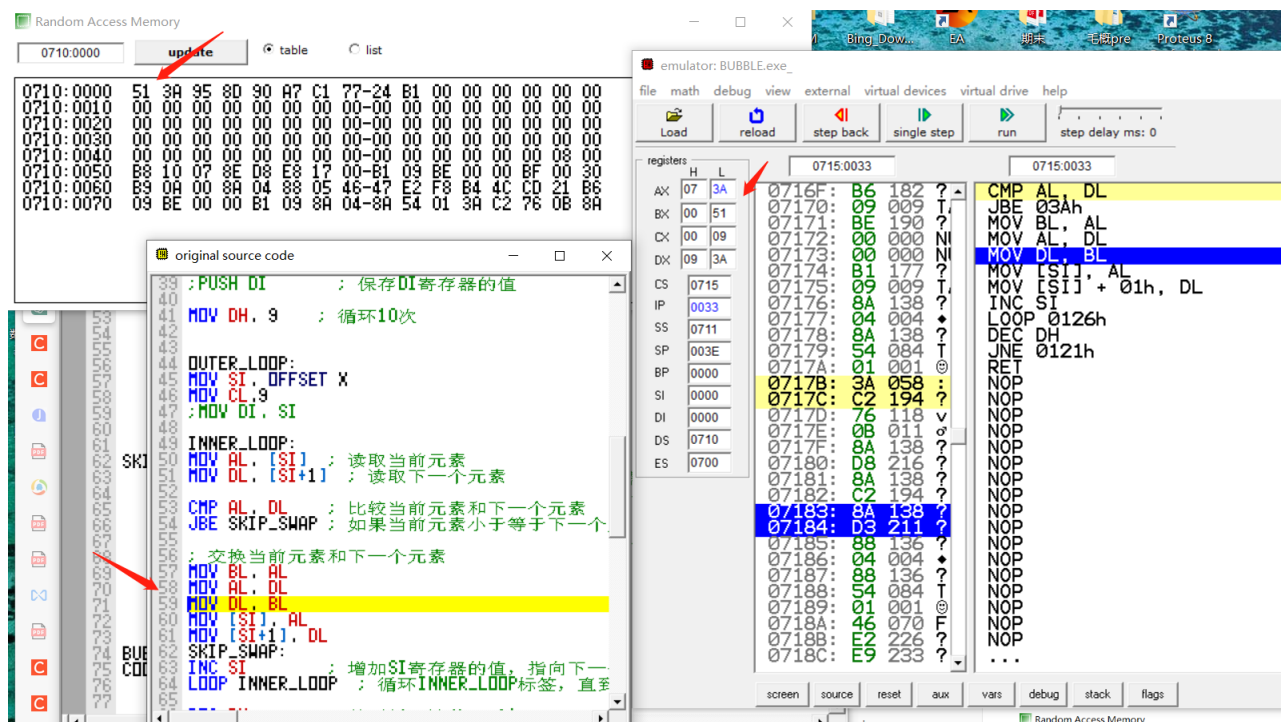
(样例: 51H, 3AH, 95H, 8DH, 90H, 0A7H, 0C1H, 77H, 24H, 0B1H)

1 实验思路

将数据段放在DS: 0000处，在子程序段中进行初步冒泡排序，然后再将所有数据迁移到对应地址中

2 实验结果

判断第一个数51H和3AH时，3AH小于51H，因此进行交换，将BL作为数据中转寄存器



可以看到，交换后，3AH排到了51H前

Random Access Memory

0710:0000	3A	51	95	8D	90	A7	C1	77-24	B1	00	00	00
0710:0010	00	00	00	00	00	00	00	00-00	00	00	00	00
0710:0020	00	00	00	00	00	00	00	00-00	00	00	00	00
0710:0030	00	00	00	00	00	00	00	00-00	00	00	00	00
0710:0040	00	00	00	00	00	00	00	00-00	00	00	00	00
0710:0050	B8	10	07	8E	D8	E8	17	00-B1	09	BE	00	00
0710:0060	B9	0A	00	8A	04	88	05	46-47	E2	F8	B4	4C
0710:0070	09	BE	00	00	B1	09	8A	04-8A	54	01	3A	C2

```

53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77

41 MOV DH, 9 ; 循环10次
42
43
44 OUTER_LOOP:
45 MOV SI, OFFSET X
46 MOV CL, 9
47 ; MOV DI, SI
48
49 INNER_LOOP:
50 MOV AL, [SI] ; 读取当前元素
51 MOV DL, [SI+1] ; 读取下一个元素
52
53 CMP AL, DL ; 比较当前元素和下一个元素
54 JBE SKIP_SWAP ; 如果当前元素小于等于下一个元素
55
56 ; 交换当前元素和下一个元素
57 MOV BL, AL
58 MOV AL, DL
59 MOV DL, BL
60 MOV [SI], AL
61 MOV [SI+1], DL
62 SKIP_SWAP:
63 INC SI ; 增加SI寄存器的值
64 LOOP INNER_LOOP ; 循环INNER_LOOP
65

```

可以看到，当冒泡排序结束后，对应初地址的结果已经按照从小到大的顺序排好

Random Access Memory															
0710:0000		update		table		list									
0710:0000	24	3A	51	77	8D	90	95	A7-B1	C1	00	00	00	00	00	00
0710:0010	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0710:0020	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0710:0030	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0710:0040	00	00	00	00	00	00	00	00-00	00	1F	00	15	07	07	02
0710:0050	88	10	07	8E	D8	E8	17	00-B1	09	BE	00	00	BF	00	30
0710:0060	89	0A	00	8A	04	88	05	46-47	E2	F8	B4	4C	CD	21	B6
0710:0070	09	BE	00	00	B1	09	8A	04-8A	54	01	3A	C2	76	0B	8A

对应偏移地址也已经成功排好序

Random Access Memory															
0710:3000		update		table		list									
0710:3000	24	3A	51	77	8D	90	95	A7-B1	C1	00	00	00	00	00	00
0710:3010	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0710:3020	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0710:3030	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0710:3040	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0710:3050	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0710:3060	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0710:3070	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00

Note: 本文比较采用无符号比较，因为对于16进制而言，无符号更容易判断程序是否正确，若要进行有符号数的冒泡排序，只需将CODE中第54行的比较结果跳转更换成 `JLE`

BUBBLE.asm源码：

```

1  DATA SEGMENT
2      X DB 51H, 3AH, 95H, 8DH, 90H, 0A7H, 0C1H, 77H, 24H, 0B1H
3  DATA ENDS
4
5  STACK SEGMENT PARA STACK
6      DW 20H DUP(0)
7  STACK ENDS
8
9
10 CODE SEGMENT
11     ASSUME DS:DATA, CS:CODE
12     START: MOV AX,DATA
13             MOV DS,AX
14             ;LEA SI,X           ; 将SI寄存器指向数据段的起始地址
15
16             CALL BUBBLE_SORT    ; 调用子程序BUBBLE_SORT进行冒泡排序
17             MOV CL,9

```

```

18         MOV SI, OFFSET X
19         MOV DI, 3000H
20         MOV CX, 10      ; 循环10次
21
22 COPY_LOOP:
23         MOV AL, [SI]    ; 读取当前元素
24         MOV [DI], AL    ; 将当前元素存储到目标地址
25
26         INC SI          ; 增加SI寄存器的值, 指向下一个元素
27         INC DI          ; 增加DI寄存器的值, 指向下一个目标地址
28         LOOP COPY_LOOP ; 循环COPY_LOOP标签, 直到CX寄存器为0
29
30         MOV AH, 4CH     ; 程序结束
31         INT 21H
32
33
34 ; 子程序: BUBBLE_SORT
35 ; 使用冒泡排序法将数据段中的字节从小到大排序
36 BUBBLE_SORT PROC
37     ; PUSH CX          ; 保存CX寄存器的值
38     ; PUSH SI          ; 保存SI寄存器的值
39     ; PUSH DI          ; 保存DI寄存器的值
40
41     MOV DH, 9          ; 循环10次
42
43
44 OUTER_LOOP:
45     MOV SI, OFFSET X
46     MOV CL, 9
47     ; MOV DI, SI
48
49 INNER_LOOP:
50     MOV AL, [SI]      ; 读取当前元素
51     MOV DL, [SI+1]    ; 读取下一个元素
52
53     CMP AL, DL        ; 比较当前元素和下一个元素
54     JBE SKIP_SWAP    ; 如果当前元素小于等于下一个元素, 跳过交换
55
56     ; 交换当前元素和下一个元素
57     MOV BL, AL
58     MOV AL, DL
59     MOV DL, BL
60     MOV [SI], AL
61     MOV [SI+1], DL
62 SKIP_SWAP:
63     INC SI            ; 增加SI寄存器的值, 指向下一个元素

```



```

64      LOOP INNER_LOOP    ; 循环INNER_LOOP标签, 直到CX寄存器为0
65
66      DEC DH              ; 外层循环计数器减1
67      JNZ OUTER_LOOP     ; 如果外层循环计数器不为0, 继续外层循环
68
69      ;POP DI             ; 恢复DI寄存器的值
70      ;POP SI             ; 恢复SI寄存器的值
71      ;POP CX             ; 恢复CX寄存器的值
72      RET
73
74 BUBBLE_SORT ENDP
75 CODE      ENDS
76      END START
77

```

题三

设数据段中初始存放了分数在 1~100 的 10 个成绩, 将这些成绩放入 初始地址为 3000H 的单元当中, 3000H+I 表示第 I 位同学的成绩。编写程序, 将排出的学生成绩名次放在初始地址为 3100H 的单元中, 3100H+I 表示第 I 位同学的名次。本题编号 I 的范围是 0~9

(样例: 56H, 4DH, 5DH, 52H, 64H, 47H, 51H, 5BH, 4FH, 61H)

1 实验思路

创建Rank_segment来存放排名

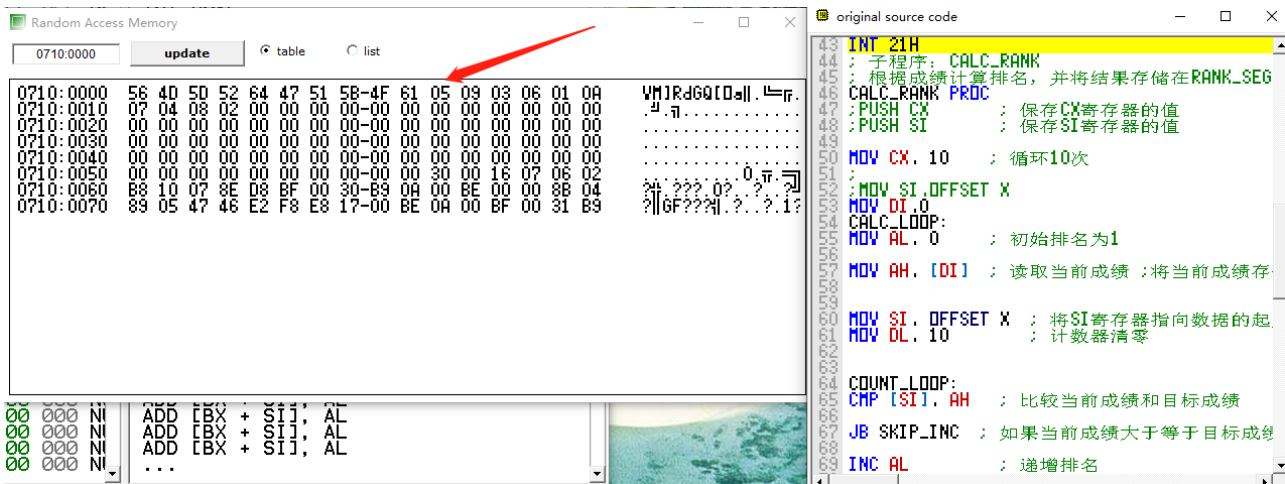
如何计算排名: AL存放排名, AH存放当前成绩, 计算每位数据的排名时, 将其初始化为1, 然后从头开始遍历, 如果有成绩大于当前成绩, 将当前成绩排名+1

如此循环十次, 可以得到每个数据的排名

2 实验结果

可以看到当程序运行结束后, 每个数据对应的排名, 且范围为1-10

因此在将排名迁移到3100H时, 要将其减1



所以在3100H的排名范围为0-9



源码score_proc.asm

```

1 DATA SEGMENT
2     ;Y DB 3000H DUP(0)
3     X DB 56H,4DH,5DH,52H,64H,47H,51H,5BH,4FH,61H ;定义数据
4     RANK_SEGMENT DB 10 DUP(0) ; 用于存储排名结果
5     ;TIMES DB 0
6 DATA ENDS
7
8
9 STACK SEGMENT PARA STACK
10     DW 20H DUP(0)
11 STACK ENDS
12
13 CODE SEGMENT
14     ASSUME DS:DATA,CS:CODE
15 START:MOV AX, DATA
16         MOV DS, AX
17         MOV DI, 3000H ; 存储成绩的起始地址
18         MOV CX, 10    ; 循环10次
19
20
21         LEA SI, X      ; 将SI寄存器指向数据的起始地址

```

```

22 STORE_LOOP: MOV AX,[SI]
23             MOV [DI],AX
24             INC DI
25             INC SI
26             LOOP STORE_LOOP
27
28             CALL CALC_RANK ; 调用子程序CALC_RANK进行排名计算
29
30             MOV SI, OFFSET RANK_SEGMENT
31             MOV DI, 3100H ; 存储排名的起始地址
32             MOV CX, 10 ; 循环10次
33
34 COPY_LOOP:
35             MOV AL, [SI] ; 读取当前排名
36             MOV [DI], AL ; 将当前排名存储到目标地址
37             DEC [DI],1
38             INC SI ; 增加SI寄存器的值, 指向下一个排名
39             INC DI ; 增加DI寄存器的值, 指向下一个目标地址
40             LOOP COPY_LOOP ; 循环COPY_LOOP标签, 直到CX寄存器为0
41
42             MOV AH, 4CH ; 程序结束
43             INT 21H
44 ; 子程序: CALC_RANK
45 ; 根据成绩计算排名, 并将结果存储在RANK_SEGMENT中
46 CALC_RANK PROC
47             ;PUSH CX ; 保存CX寄存器的值
48             ;PUSH SI ; 保存SI寄存器的值
49
50             MOV CX, 10 ; 循环10次
51             ;
52             ;MOV SI,OFFSET X
53             MOV DI,0
54 CALC_LOOP:
55             MOV AL, 0 ; 初始排名为1
56
57             MOV AH, [DI] ; 读取当前成绩 ;将当前成绩存储在AH寄存器中
58
59
60             MOV SI, OFFSET X ; 将SI寄存器指向数据的起始地址
61             MOV DL, 10 ; 计数器清零
62
63
64 COUNT_LOOP:
65             CMP [SI], AH ; 比较当前成绩和目标成绩
66
67             JB SKIP_INC ; 如果当前成绩大于等于目标成绩, 跳过递增排名

```

```

68
69     INC AL           ; 递增排名
70
71 SKIP_INC:
72     INC SI           ; 增加SI寄存器的值, 指向下一个成绩
73
74     DEC DL           ; 递减DL, 循环次数为0
75     JNZ COUNT_LOOP
76     MOV [RANK_SEGMENT + DI], AL ; 将排名存储在RANK_SEGMENT中
77     INC DI
78     LOOP CALC_LOOP   ; 循环CALC_LOOP标签, 直到CX寄存器为0
79     RET
80
81 CALC_RANK ENDP
82
83 CODE ENDS
84     END START

```

3 心得

NOTE:

- 判断符号要是无符号数,
- 在子程序中, 如果没有PUSH, 不能有POP, push与pop是成对的