实验三　子程序及宏的使用

实验报告

姓名：赵文亮

学号：2016011452

班级：自 64

日期：2018 年 4 月 23 日

目录

1 实验目的 **1**

2 实验准备 **1**

2.1 子程序 1

2.2 宏 1

2.3 DOS 功能调用 2

3 实验内容 **2**

4 输入 **2**

5 求最大值 **2**

6 排序 **3**

7 实验程序及说明 **3**

7.1 输入 6

7.2 求最大值 8

7.3 排序 8

7.4 其他子程序与宏 9

8 思考题 **9**

9 完成情况和心得体会 **11**

1 实验目的

- 练习使用子程序及宏。
- 练习使用功能调用 (INT 21H) 中关于程序终止、字符输入及字符输出部分。

2 实验准备

本实验中主要用到内容有主要有子程序、宏和 DOS 功能调用。

2.1 子程序

子程序（过程）是一种伪操作，类似于 C 语言中的函数，调用时 CS:IP 会跳转到子程序中，并在跳转前将 CS 和 IP 入栈保护。过程定义伪操作命令为 PROC/ENDP，格式如下：

```
过程名  PROC  [NEAR/FAR]
        ⋮
        RET
        ⋮
过程名  ENDP
```

程序中需要调用过程时，可以使用 CALL 指令。类型为 NEAR 的过程可以在段内调用，类型为 FAR 的过程可以被其他段调用。

```
CALL    过程名
```

2.2 宏

宏也是一种伪操作，在每个宏调用出，宏定义体被展开。宏类似于 C 语言中的#define。MACRO/ENDM 用来定义宏：

```
宏指令名    MACRO
            ⋮   (宏定义体)
            ENDM
```

或者包含参数：

```
宏指令名    MACRO   参数[,...]
            ⋮   (宏定义体)
            ENDM
```

调用时需要将实参传入宏。

宏与子程序的区别在于，宏的调用相当于在编译的时候就将宏定义展开，即只是减少了代码的编写量，并没有减少最后生成的机器码的量，比较占代码段空间。优点是在执行的过程中始终是顺序执行，运行速度快；子程序在调用的时候 CS:IP 会跳转，只会占少量的堆栈段空间，比较节约代码段。缺点是在调用的过程中有保护断点、恢复断点等操作，额外占用 CPU 的时间。

2.3 DOS 功能调用

DOS 系统功能调用通过INT 21H来实现。本次实验中用到的主要功能如表 1 所示。

表 1: DOS 系统功能调用表

功能号 (AH)	功能	入口参数	出口参数
01	输入字符并回显		AL= 输入字符的 ASCII 码
02	显示字符	DL= 被显示字符的 ASCII 码	
09	显示字符串 (串尾字符为 $，但不显示)	DS:DX= 被显示字符串首地址	
4C	带返回码结束	AL= 程序员自定的返回码	

3 实验内容

本次实验的可以分为十进制数输入、求最大值、排序这三个步骤。最终的代码可以在一个文件中完成，见第 7 节。

4 输入

处理十个十进制数的输入首先需要在一个循环中进行，每个循环中处理一个数的输入。对于每个两位十进制数，其输入过程可以分为以下几种情况。

1. 正常输入。此状态下，用户可以无限输入数字，直到按下回车或输入错误字符。
2. 错误输入。如果输入的字符不是 0-9 的数字且不是回车，则被视为错误输入。此时打印出错误信息，并重新开始本次输入。
3. 按下回车。选取本次输入的数字的后两位作为有效输入。例如在输入状态下用户输入了“1234”，则视为输入“34”。

每次输入后，将结果保存在内存中。十次输入结束后，将所有的数据顺次打印在屏幕上。

5 求最大值

求最大值的思路比较简单，步骤如下：

1. 假定第一个元素为最大元素。
2. 从前至后遍历，如果发现有元素比当前最大元素更大，则更新最大元素的值。

为了便于后续排序的实现，我每次找到较大的元素都将其放在内存区的第一个。

6 排序

在求最大值的基础上，即可方便地实现选择排序。令 SI 和 DI 分别指向第一个和最后一个元素，SI 从前至后移动，每次移动都调用求最大值的过程，使得 SI 至 DI 中的最大元素交换到 SI 中，即可实现选择排序。

7 实验程序及说明

三个程序联系紧密，我将三个程序的代码写在了一起。下面将分模块进行分析。

```
1  NAME     PROGRAM3
2  DATA     SEGMENT
3  TIP DB 'INPUT DECIMAL NUMBER ','$'
4  ERROR    DB 'NOT DECIMAL NUMBER!', 0DH, 0AH, '$'
5  ENDIN    DB 'ALL THE NUMBERS: ', 0DH, 0AH, '$'
6  SHOWMAX DB 'THE MAX NUM is:', '$'
7  NEWLINE DB 0DH, 0AH, '$'
8  SORTD    DB 'SORT DONE!', 0DH, 0AH, '$'
9  ORG      1000H; decide the position of the numbers
10 NUMLIST DB 10 DUP(?)
11 DATA     ENDS
12
13 STACK    SEGMENT PARA STACK
14          DB 100 DUP(?)
15 STACK    ENDS
16 CODE     SEGMENT
17          ASSUME  CS: CODE, DS: DATA, ES:DATA, SS: STACK
18
19 DISP     MACRO CONTENTS
20          LEA DX, CONTENTS
21          MOV AH, 9
22          INT 21H
23          ENDM
24
25 START:   MOV AX, DATA
26          MOV DS, AX
27          MOV ES, AX
28
29          MOV CH, 0H; initialize the registers
30          MOV CL, 0H
31          MOV BX, 0H
32
33          LEA SI, NUMLIST
34          LEA DI, NUMLIST
35
36
37 INPUTL: CALL INPUT; input loop
```

```
38          CMP CL, 10; for 10 times
39          JNE INPUTL
40
41          LEA SI, NUMLIST
42          MOV CL, 10
43          PUSH SI
44
45  DISP    ENDIN
46  SALL:   MOV DL, [SI]; show all the numbers
47          CALL DISPNUM
48          INC SI
49          DEC CL
50          JNZ SALL
51          POP SI
52
53          CALL FINDMAX
54  DISP    SHOWMAX
55          MOV DL, BL
56          CALL DISPNUM
57          CALL SORT
58
59          MOV AH, 4CH
60          INT 21H
61
62
63  INPUT   PROC    ; input one character
64          CMP CH, 0
65          JNE SKIPD; if not finished this number skip display
66
67  DISP    TIP
68          MOV DL, CL
69          MOV AL, DL
70          ADD AL, 1
71          DAA
72          MOV DL, AL
73          CALL DISPNUM; display the number
74  SKIPD:  MOV AH, 1; skip display the number
75          INT 21H
76
77          CMP AL, 0DH
78          JE CR
79          CMP AL, '0'; judge the range
80          JB INPUTE
81          CMP AL, '9'
82          JA INPUTE
83
84          SUB AL, '0'; AL is the newest digit
```

```
85              PUSH CX
86              MOV CL, 4
87              SHL BL, CL; update BL
88              POP CX
89              ADD BL, AL
90              CMP CH, 2
91              JE INPUTD
92              INC CH
93              JMP INPUT
94  INPUTE:
95  DISP        ERROR; input error
96              JMP INPUTD
97  CR:         MOV [DI], BL
98              INC DI
99              INC CL
100             MOV BL, 0
101             MOV CH, 0
102 INPUTD:     RET; input done
103 INPUT       ENDP
104
105
106
107 DISPNUM     PROC    ;display number
108             PUSH DX
109             PUSH CX
110             MOV CL, 4
111             SHR DL, CL
112             POP CX
113             CMP DL, 0
114             JE  SKIPZ;if the first digit is 0, skip the display
115             ADD DL, '0'
116             CALL DISPC
117 SKIPZ:      POP DX
118             AND DL, 0FH
119             ADD DL, '0'
120             CALL DISPC
121 DISP        NEWLINE
122             RET
123 DISPNUM     ENDP
124
125 DISPC       PROC    ;display character
126             MOV AH, 2
127             INT 21H
128             RET
129 DISPC       ENDP
130
131 FINDMAX     PROC    ;find the max number from si to di, and save to bl
```

```
132             ;[si] will be the maximum number after the proc
133             PUSH SI
134             MOV AX, DI
135             MOV DI, SI
136     NEXTMAX:INC DI
137             MOV BL, [SI]
138             CMP BL, [DI]
139             JNB NCMAX; jump not change the max number
140             CALL SWAP;change the max number
141     NCMAX:  CMP DI, AX; not change the max number
142             JNE NEXTMAX;     compare next
143             POP SI
144             MOV DI, AX
145             RET
146     FINDMAX ENDP
147
148     SWAP    PROC    ;swap the two numbers in [di] and [si]
149             PUSH BX
150             MOV BL, [DI]
151             MOV BH, [SI]
152             MOV [DI], BH
153             MOV [SI], BL
154             POP BX
155             RET
156     SWAP    ENDP
157
158     SORT    PROC    ;sort the numbers
159     SORTL:  CALL FINDMAX; the sort loop, using select sort
160             INC SI
161             CMP SI, DI
162             JNE SORTL
163     DISP    SORTD
164             RET
165     SORT    ENDP
166
167     CODE    ENDS
168     END     START
```

7.1 输入

输入部分的大循环为第 37 行的 INPUTL。每次循环调用 INPUT 过程来处理一个数的输入。

INPUT 过程位于第 63 行。首先判断是输入新的数还是继续输入上一个数。若是输入新的数则会显示一个提示信息，否则不显示。其中提示信息中包含了当前正在输入的数字的序号。这个序号通过 68-71 行计算而来，其中已经考虑到了十进制调整。

BL 用来保存当前的 2 位十进制数。每次输入一个字符，如果为回车，则结束本次输入；如果字符无

效（即不是 0-9），则显示错误信息（77-82 行）；如果字符有效，则将 BL 中的数左移四位，并将新输入的字符转为 BCD 存入 BL 的低四位（85-89）。

输入回车时，将当前 BL 中的十进制数写入内存中，并重置寄存器准备下一次输入（97-101）。这样，程序会认为用户没有输入数字就按下回车时写入的数字是 0，而不会出现闪退等问题。输入部分的流程图如图 1 所示。

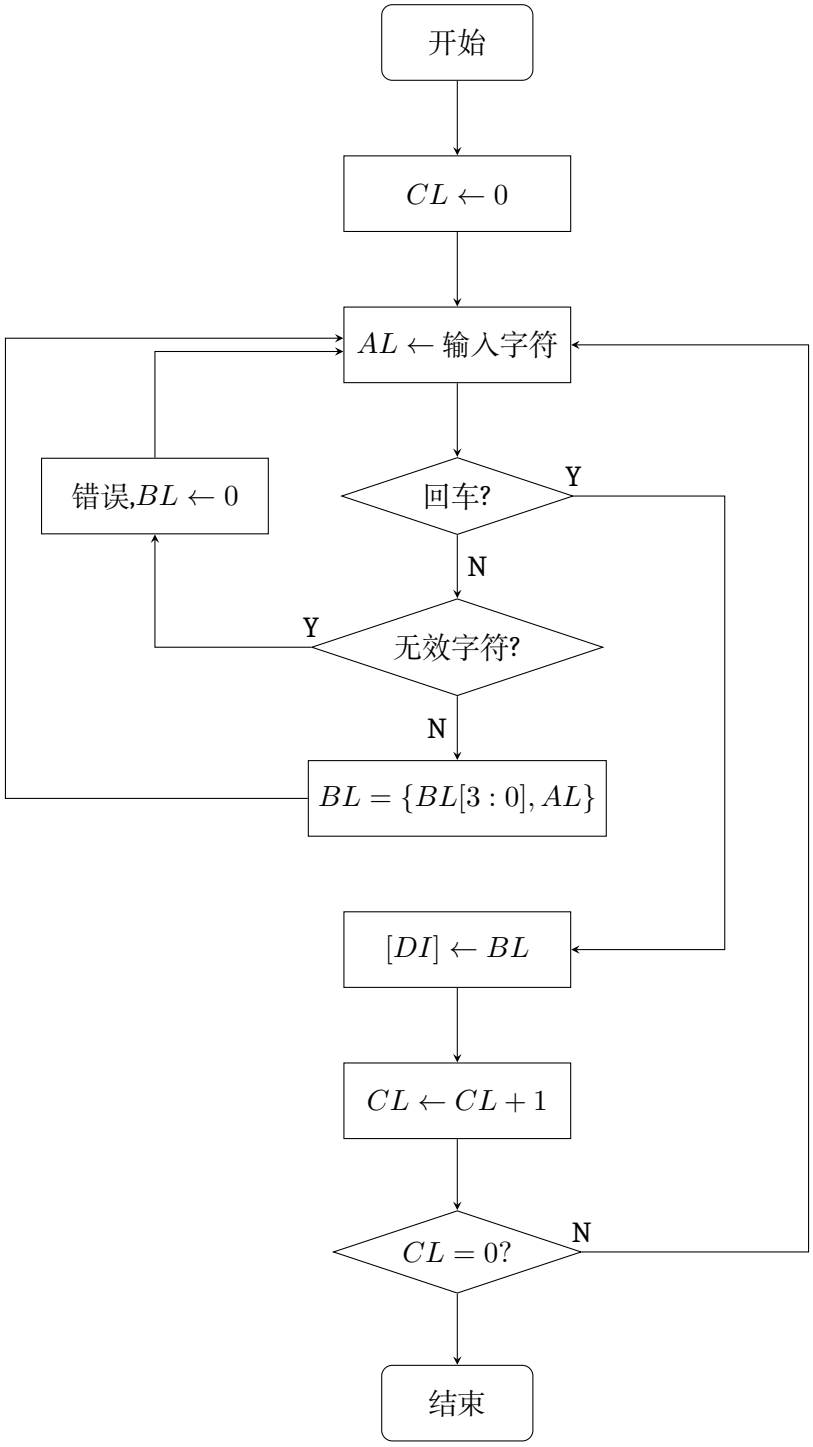

图 1: 输入部分流程图

7.2 求最大值

FINDMAX 是用来求最大值的子程序，见程序第 131 行。该程序实现了将 SI 到 DI 中所有数的最大值交换到 SI 的位置。执行结束后，SI 位置的元素将是最大元素。首先将 DI 暂存，用 DI 作为遍历的指针，从 SI 开始逐个元素比较，如果 [DI] 比 [SI] 小则跳转到 NCMAX（Not Change Max），否则调用交换过程。交换过程里会将 [SI] 和 [DI] 中的两个数进行交换。这样就可以实现最大值的选取。求最大值的流程图如图 2 所示。

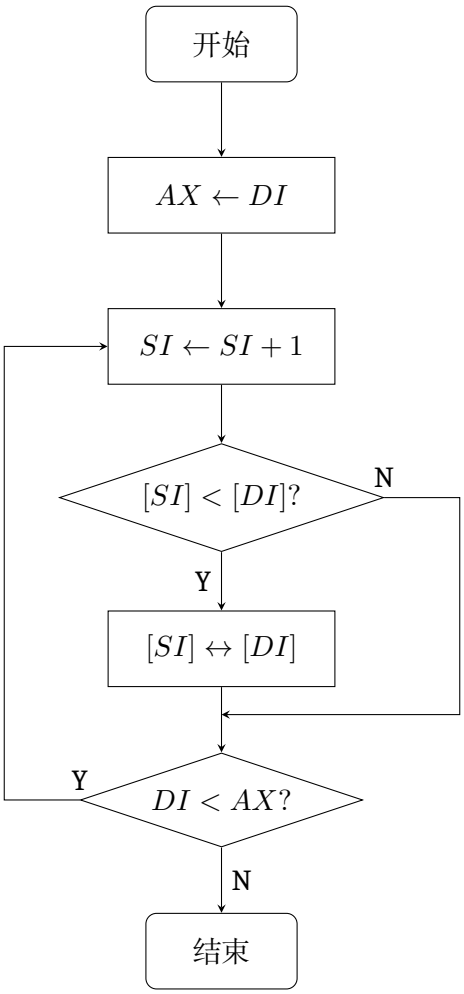

图 2: 取最大值流程图

7.3 排序

有了求最大值的程序，排序变得十分简单。排序的循环 SORTL 见第 159 行。只需每次递增 SI，并执行 FINDMAX 程序，即可将当前 SI-DI 中所有数的最大值换到 SI 位置处。遍历结束后内存区便是排好顺序的数据了。排序部分的流程图如图 3 所示。

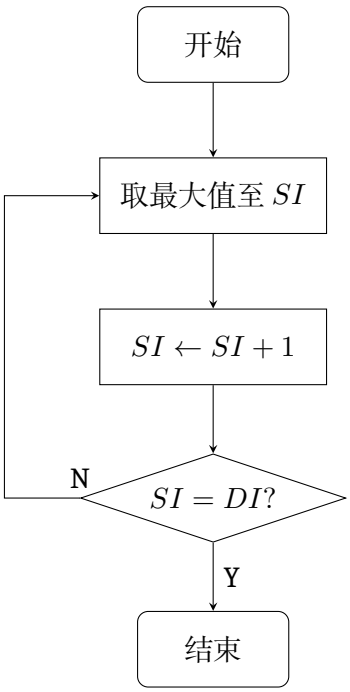

图 3: 排序流程图

7.4 其他子程序与宏

为了方便显示字符和字符串，我编写了输出字符串的宏 DISP（19 行）、输出单个字符的子程序 DISPC（125 行）、输出十进制数的子程序 DISPNUM（123 行）。

8 思考题

若处理的是十六进制无符号数，程序应如何修改？ 十六进制无符号数会包括 ASCII 字符 “abcdefABCDEF”，为了方便和用户友好，可以忽略大小写。只需在输入部分对这部分 ASCII 字符进行特殊处理即可。输入部分处理如下：

```
1  INPUT   PROC    ; input one character
2          CMP CH, 0
3          JNE SKIPD; if not finished this number skip display
4
5  DISP    TIP
6          MOV DL, CL
7          MOV AL, DL
8          ADD AL, 1
9          DAA
10         MOV DL, AL
11         CALL DISPNUM; display the number
12 SKIPD:  MOV AH, 1; skip display the number
13         INT 21H
14
```

```
15          CMP AL, 0DH
16          JE CR
17          CMP AL, '0'; judge the range
18          JB INPUTE
19          CMP AL, '9'
20          JA ATOF
21
22          SUB AL, '0'; AL is the newest digit
23          JMP UPDATE
24
25  ATOF:   CMP AL, 'A'
26          JB INPUTE
27          CMP AL, 'F'
28          JA SATOF
29          SUB AL, 37H
30          JMP UPDATE
31
32  SATOF:  CMP AL, 'a'
33          JB INPUTE
34          CMP AL, 'f'
35          JA INPUTE
36          SUB AL, 57H
37
38  UPDATE: PUSH CX
39          MOV CL, 4
40          SHL BL, CL; update BL
41          POP CX
42          ADD BL, AL
43          CMP CH, 2
44          JE INPUTD
45          INC CH
46          JMP INPUT
47  INPUTE:
48  DISP    ERROR; input error
49          JMP INPUTD
50  CR:     MOV [DI], BL
51          INC DI
52          INC CL
53          MOV BL, 0
54          MOV CH, 0
55  INPUTD: RET; input done
56  INPUT   ENDP
```

输出部分也应该做相应的处理，即判断是否大于 10，若大于 10 则应该输出 A~F 的数

```
1   DISPNUM PROC    ;display number
2           PUSH DX
3           PUSH CX
```

```
4           MOV CL, 4
5           SHR DL, CL
6           POP CX
7           CMP DL, 0
8           JE  SKIPZ;if the first digit is 0, skip the display
9           CMP DL, 10
10          JB  PLUS0
11          ADD DL, 7
12  PLUS0:  ADD DL, '0'
13          CALL DISPC
14  SKIPZ:  POP DX
15          AND DL, 0FH
16          CMP DL, 10
17          JB  PLUS1
18          ADD DL, 7
19  PLUS1:  ADD DL, '0'
20          CALL DISPC
21  DISP    NEWLINE
22          RET
23  DISPNUM ENDP
```

取最大值与排序的代码不变

若处理的是带符号数，程序又应作何修改？ 若处理的是带符号的数，则每个 2 位十进制或十六进制数不能够使用一个字节存下来。为此可以专门开辟两个字节的空间，存储各个数字的符号位。

```
1  SIGNS   DB 2 DUP(?)
```

输入的程序里面，判断输入符号是否为'-'，如果是则在 SIGNS 的对应位写 1。输出的程序先从 SIGNS 读取符号，如为 1 则输出一个'-'。

寻找最大值的程序需要相应改动：首先判断 SI 至 DI 是否所有的符号位均为 1，如果不是，则从第一个正数开始比较，遇到负数直接跳过；如果所有的符号位均为 1，则可以寻找所有数的最小值（即为带符号数的最大值）。

排序仍可以在寻找最大值的程序上进行，无需修改。

9 完成情况和心得体会

在实验课之前，我已经完成了代码并得到了正确的结果，所以整体来说比较顺利。然而在验收过程中出了一个小问题：我写代码的时候没有注意到实验任务中“输入完成后重新将十个数字输出在屏幕上”的要求，导致第一次验收的时候没有通过。于是我立即在机房的电脑上修改代码。由于之前已经将输出部分进行过封装，我很快就完成了这个简单的要求，并顺利通过了验收。

通过本次实验，我对宏和子程序的概念、作用和区别有了更加深刻的理解，了解了其优缺点和适用条件。有了这两大帮手，相信以后编写程序一定会更加简洁、可读性更好、接口更加规范。

我对 DOS 系统功能调用的部分也是很感兴趣。当看到自己的程序能够实现与用户的交互时，我是非常激动的，正如当初写 C 语言程序时的那样。相信以后的实验中一定有更加丰富的内容等着我们去发掘和探索。