

# 计算机原理第三次实验报告

张蔚桐 2015011493 自55

2017 年 4 月 20 日

## 1 实验目的

1. 练习使用子程序和宏
2. 练习使用功能调用（INT 21H）中关于字符输入及字符输出的部分

## 2 实验内容

实验完成了所有的必做和选做任务，下面按照程序执行的流程分块解释。

程序事先存储若干字符串方便后期显示的时候调用，同时将数据存储在1000H的偏移地址上。

程序开始执行后，首先完成必要的幅值工作和初始化工作，之后在Line30开始输出字符串“INPUT 10 DECIMAL NUMBER:”提示用户输入十个十进制数，Line33时，程序将DI指针移向输入内存块NUMBERS的最后一个单元，之后进入循环输入

整个循环退出的条件是DI指针移动到NUMBERS偏移地址之前，这样完成了对NUMBERS数据区的写入操作

在循环体内执行如下操作

首先调用INT 21H中断完成一个字符的读入工作，调用模式为(AH=08H)即输入不回显，检查输入的字符，如果为0DH（回车）的话更换为空格重新输出，保证输入格式的美观性，到此完成Line44之前的工作

之后开始对输入的字符进行检查，如Line47开始，首先检查是否为ASCII码的数字，如果为数字的话所执行的语句在Line53所示，检查DH（已经输入的数字）是否超过两位，如果超过两位，跳转到OVER处理块

如果输入正常，则左移DH4位，将DL从ASCII码转换为数字，之后将DL加到DH上，完成对DH的修改

如果输入的字符不是数字，如Line48等行所示，将跳转至NOTNUM处理块

下面叙述NOTNUM处理块的行为，首先检测输入的字符是否为空格，如果输入的字符为回车，因为已经在上面所述的步骤中进行了替换，因此可以一起处理。如果此时DH非零，说明结束了一个数字的输入，进入后续的OUTPUT处理块，否则返回循环开始，重新输入，相当于连续输入多个空格（回车）是无效行为

如果输入的非数字不是空格（或回车）的话，说明输入了错误字符，如Line64所示，程序显示错误原因“NOT DECIMAL NUMBER!”并通过调用CORRECT函数，显示当前的错误信息

现在说明OVER处理块的处理形式，和上面一样，具体的实现如Line69所示，程序显示错误原因“TOO MANY DIGITS”，并通过调用CORRECT函数，显示当前的错误信息

下面说明CORRECT函数的处理方式，具体实现Line137所示，首先程序调用SHOW函数显示当前已经输入的数字，其次程序提示用户还需要输入的数字，如Line147所示，首先程序将DI和数据域NUMBERS开始的地址的差赋给DX，则DL便是所要求的还需要输入的数字个数，接下来对DL的值进行分析，如果DL==10，则直接调用已经写好的数据域AUX输出10，否则DL加030H并利用INT 21H的AH=02H模式输出字符，在此前后，程序显示一些交互的信息，具体方式可见Line138之后的具体实现，主要方式是调用INT 21H的AH=09H的中断，具体叙述略去

下面简述SHOW函数的具体实现，首先仍然使用INT 21H的AH=09H中断进行一些用户交互信息的输出，之后程序将SI置为NUMBERS区尾地址，并逐字节读取直到DI==SI，在每次读取到DL后，拷贝到DH，其中DH去低4位，DL取高4位，分别移到相对应的低4位补加030H转ASCII码，先高位DL后低位DH调用INT 21H的AH=02H输出字符，之后程序依然完成一些用户交互上而定输出。其中具体实现见Line160

下面简述循环中最后一个模块OUTPUT的实现，首先将DH中的值移到DI所指示的内存中，递减DI，清空DH，等待下次输入，具体实现在Line78

至此，程序完成输入部分的操作

在Line84，程序开始调用WHOLE函数逐个显示输入的数值，WHOLE显示一个BX中信息，因此首先这里显示MESSAGE中的用户交互信息，WHOLE的具体实现在Line185，具体的方式和之前提到的SHOW大同小异，这里就不进行进一步的叙述了，唯一的不同是，输出的间隔从空格改变成为了换行符

在Line90程序开始执行冒泡排序，具体算法如下

首先程序将DL标志置零，在遍历中一旦发现有前面的字符大于后面的字符的情况，DL变为1，程序在Line101退出直到DL为0，说明整个数组已经从小到大排序，在遍历中，程序执行如下操作

内层循环开始时，程序将DI指向NUMBERS开始地址，并读取两位到AX，其中AL是前一位，AH是后一位，一旦AL大于AH，则将AL和AH交换顺序，重新将AX写入原位置，完成了检查和交换，之后DI递增，对于10个数的数组，程序执行9次。具体在Line94附近实现

在完成了冒泡排序后，数组从前到后是从小到大的排列，再次调用WHOLE函数，从后向前输出数组中的元素，完成排序

同时，直接输出数组的最后一个元素，完成最大数的输出工作

至此，程序完成要求的所有工作，当然在上面执行的过程中，存在着一些利用INT 21H，主要是AH=09H的输出用户交互的工作

程序在Line131行将通过INT 21H，AH=04CH将控制权转回系统，并返回AL=0，之前所述的操作在前面几行进行

程序的代码如下所示，运行时的截图如图1,2所示

```

01 DATA SEGMENT
02 MESSAGE0 DB 'INPUT 10 DECIMAL NUMBER:',0DH,0AH,'$'
03 MESSAGE1 DB 'YOU HAVE INPUT: ','$'
04 MESSAGE2 DB 0DH,0AH,'YOU HAVE TO INPUT ANOTHER $'
05 MESSAGE3 DB ' NUMBERS.',0DH,0AH,'$'
06 ERROR1 DB 0DH,0AH,'NOT DECIMAL NUMBER!',0DH,0AH,'$'
07 ERROR2 DB 0DH,0AH,'TOO MANY DIGITS',0DH,0AH,'$'
08 SORT DB '>>>> AFTER SORTED <<<',0DH,0AH,'$'
09 MAX DB '>>>> MAX NUMBER: ','$'
10 AUX DB '10'
11 ORG 1000H
12 NUMBERS DB 10 DUP(?)
13 DATA ENDS
14
15 STACK SEGMENT PARA STACK
16 DB 10 DUP(?)
17 STACK ENDS
18
19 CODE SEGMENT
20 ASSUME ES:DATA,DS:DATA,CS:CODE,SS:STACK

```

```
21 START:
22 MOV AX,DATA
23 MOV ES,AX
24 MOV DS,AX
25 MOV AX,STACK
26 MOV SS,AX
27 LEA DI,NUMBERS
28 MOV CL,4
29 ;READ IN 10 NUMBERS
30 MOV DX,OFFSET MESSAGE0
31 MOV AH,09H
32 INT 21H
33 ADD DI,09H
34 L1:
35 ;IO AND REFINE THE FORMAT
36 MOV AH,08H
37 INT 21H
38 MOV DL,AL
39 CMP DL,0DH
40 JNE L2
41 MOV DL,' '
42 L2:
43 MOV AH,02H
44 INT 21H
45 ;WORD IN DL
46 ;SET THE WORD IN DH
47 CMP DL,030H
48 JB NOTNUM
49 CMP DL,039H
50 JA NOTNUM
51 ;NOW THE INPUT SHOULD BE A NUMBER
52 ;CHECK THE DH
53 CMP DH,00FH
54 JA OVER
55 ;NOW EVERYTHING OK
56 AND DL,00FH
```

```
57  SHL DH,CL
58  ADD DH,DL
59  JMP L1
60  NOTNUM:
61  ;CHECK THE ' '
62  CMP DL,' '
63  JE L3
64  MOV DX,OFFSET ERROR1
65  MOV AH,09H
66  INT 21H
67  JMP L6
68  OVER:
69  MOV DX,OFFSET ERROR2
70  MOV AH,09H
71  INT 21H
72  L6:
73  CALL CORRECT
74  JMP L1
75  L3:
76  CMP DH,0
77  JE L1
78  OUTPUT:
79  MOV BYTE PTR [DI],DH
80  DEC DI
81  MOV DH,0
82  CMP DI,OFFSET NUMBERS
83  JNB L1
84  MOV BX,OFFSET MESSAGE1
85  CALL WHOLE
86  L11:
87  MOV DL,0
88  MOV CX,9
89  LEA DI,NUMBERS
90  L12:
91  MOV AX,[DI]
92  CMP AL,AH
```

```
93  JNA NORM
94  MOV DL,1
95  XCHG AL,AH
96  MOV [DI],AX
97  NORM:
98  INC DI
99  LOOP L12
100 CMP DL,0
101 JNE L11
102 MOV CL,4
103 MOV AH,02H
104 MOV DL,0DH
105 INT 21H
106 MOV DL,0AH
107 INT 21H
108 MOV DX,OFFSET MAX
109 MOV AH,09H
110 INT 21H
111 MOV AH,02H
112 LEA SI,NUMBERS
113 ADD SI,09H
114 MOV AH,02H
115 MOV DL,BYTE PTR [SI]
116 MOV DH,DL
117 SHR DL,CL
118 JZ L13
119 ADD DL,030H
120 INT 21H
121 L13:
122 AND DH,00FH
123 ADD DH,030H
124 MOV DL,DH
125 INT 21H
126 MOV DL,0DH
127 INT 21H
128 MOV DL,0AH
```

```
129 INT 21H
130
131 MOV BX,OFFSET SORT
132 CALL WHOLE
133 MOV AL,0
134 MOV AH,04CH
135 INT 021H
136
137 CORRECT:
138 CALL SHOW
139 MOV DX,OFFSET MESSAGE2
140 MOV AH,09H
141 INT 21H
142 MOV DX,DI
143 SUB DX,OFFSET NUMBERS
144 INC DL
145 CMP DL,0AH
146 JNE L4
147 MOV DX,OFFSET AUX
148 MOV AH,09H
149 JMP L5
150 L4:
151 ADD DL,030H
152 MOV AH,02H
153 L5:
154 INT 21H
155 MOV DX,OFFSET MESSAGE3
156 MOV AH,09H
157 INT 21H
158 RET
159
160 SHOW:
161 MOV DX,OFFSET MESSAGE1
162 MOV AH,09H
163 INT 21H
164 LEA SI,NUMBERS
```

```
165  ADD SI,09H
166  MOV AH,02H
167  L7:
168  MOV DL,BYTE PTR [SI]
169  MOV DH,DL
170  SHR DL,CL
171  JZ L10
172  ADD DL,030H
173  INT 21H
174  L10:
175  AND DH,00FH
176  ADD DH,030H
177  MOV DL,DH
178  INT 21H
179  MOV DL,' '
180  INT 21H
181  DEC SI
182  CMP SI,DI
183  JNE L7
184  RET
185  WHOLE:
186  MOV AH,02H
187  MOV DL,0DH
188  INT 21H
189  MOV DL,0AH
190  INT 21H
191  MOV DX,BX
192  MOV AH,09H
193  INT 21H
194  MOV AH,02H
195  MOV DL,0DH
196  INT 21H
197  MOV DL,0AH
198  INT 21H
199  LEA SI,NUMBERS
200  ADD SI,09H
```



```
201 MOV AH,02H
202 L8:
203 MOV DL,BYTE PTR [SI]
204 MOV DH,DL
205 SHR DL,CL
206 JZ L9
207 ADD DL,030H
208 INT 21H
209 L9:
210 AND DH,00FH
211 ADD DH,030H
212 MOV DL,DH
213 INT 21H
214 MOV DL,0DH
215 INT 21H
216 MOV DL,0AH
217 INT 21H
218 DEC SI
219 CMP SI,OFFSET NUMBERS
220 JNS L8
221 RET
222 CODE ENDS
223 END START
```

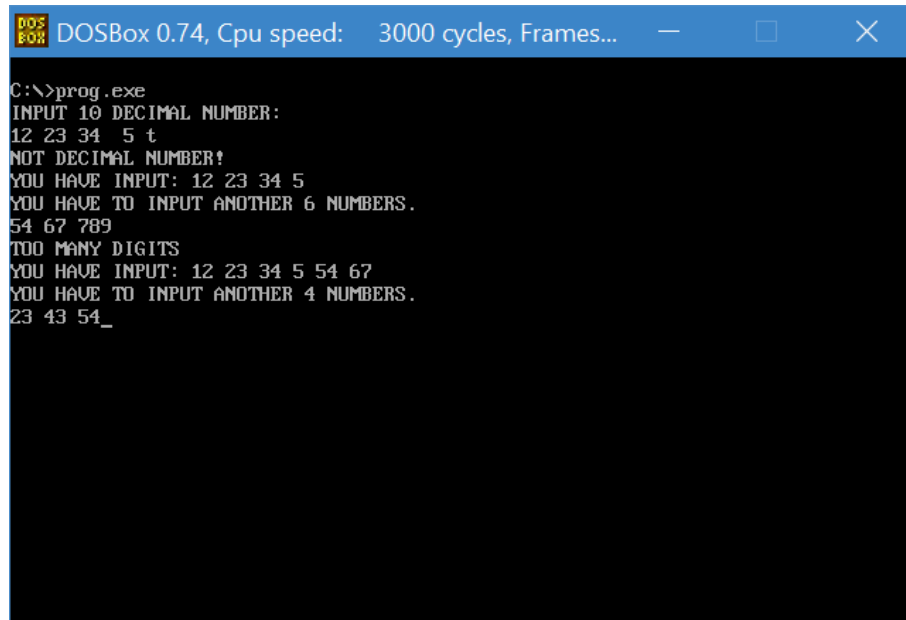
### 3 思考题

1. 若处理的是十六进制无符号数，程序应如何修改？

在字符存取的部分修改即可，加上判断，若字符为0到9，代码不变，存取时分别加减30H；若字符为 A到F，存取时加减37H。

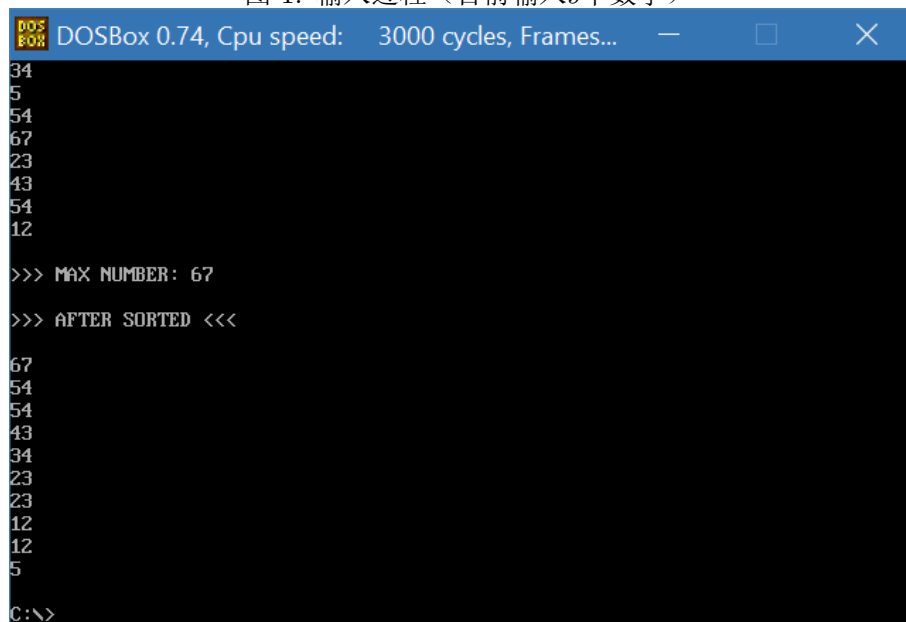
2. 若处理的是带符号数，程序又应作何修改？

首先需要判断“-”符号，然后根据正负号决定是否将两位数取补码后存储。取数时根据最高位判断正负，决定输出到显示屏时是否要加上符号，客观的说，处理带符号数的难度要上升很多



```
DOS FOR DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
C:\>prog.exe
INPUT 10 DECIMAL NUMBER:
12 23 34 5 t
NOT DECIMAL NUMBER!
YOU HAVE INPUT: 12 23 34 5
YOU HAVE TO INPUT ANOTHER 6 NUMBERS.
54 67 789
TOO MANY DIGITS
YOU HAVE INPUT: 12 23 34 5 54 67
YOU HAVE TO INPUT ANOTHER 4 NUMBERS.
23 43 54_
```

图 1: 输入过程 (目前输入9个数字)



```
DOS FOR DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
34
5
54
67
23
43
54
12

>>> MAX NUMBER: 67

>>> AFTER SORTED <<<

67
54
54
43
34
23
23
12
12
5

C:\>
```

图 2: 输出和排序过程

## 4 完成情况

本次实验因为准备充分，代码的调试已经事先完成，因此效果较好，在本次实验中，我理解了条件转移职能进行近地址跳转而不能进行远地址跳转并且使用函数解决了问题。同时，在使用函数的过程中，尽可能的避免了栈上空间的操作（程序中没有栈上空间的操作，所有的操作除了对数据块的读取外均在寄存器上进行），从最大程度上节约了运行时间，尽管，设计合适的压栈和函数可以进一步增加代码的可读性，但是作为低级语言，相对语言本身更在意的就是程序的执行效率，因此为了提高运行效率，可读性的牺牲也是可以理解的，虽然作为题目中的数据规模，避免内存访问所带来的运行时间减少是微乎其微的，但是作为对编程能力的锻炼仍然有着很大的意义