# 计算机原理与应用小学期

## 实验报告


姓名：赵文亮

学号：2016011452

班级：自 64


日期：2018 年 7 月 4 日

# 目录

# 1 实验六：D/A 与 A/D 转换

## 1.1 实验目的

1. 学习 D/A 及 A/D 转换的基本原理。
2. 掌握 D/A 及 A/D 转换器 DAC0832，ADC0809 的使用方法。

## 1.2 实验准备

1. 复习有关 D/A 及 A/D 转换，DAC0832 及 ADC0809 的有关知识。
2. 对照 DAC0832，ADC0809 工作原理及时序，看懂实验接线图，理解实现 D/A、A/D 转换的工作过程。
3. 按实验内容要求编写汇编语言源程序并加注释。

## 1.3 实验内容

### 1.3.1 D/A 转换

用 DAC 0832 实现 D/A 转换，使产生的模拟电压波形分别为锯齿波、三角波和正弦波。运行程序后等待键入（可以显示提示信息）

- 若键入‘1’Ub 产生锯齿波；
- 若键入‘2’Ub 产生. 三角波；
- 若键入‘3’Ub 产生正弦波；
- 若键入‘4’退出用户程序，返回 DOS。

用户程序运行过程中没有新键入别的数字则维持原状不变。不要求识别‘1234’以外的其他键。

#### 1.3.1.1 编程提示

- 产生连续电压波形需周期性地重复对 DAC0832 输出变化的数字量。对于锯齿波和三角波而言，输出数据成等差序列。对于正弦波，事先将数据存放在数据表中。
- 为了得到满幅度的波形，要求输出数据的变化也充满 0~FFH。
- DAC0832 每两次转换的间隔可调用一个延时子程序来实现。
- 注意认真调节示波器图形的位移和幅度，以便在屏幕范围内显示整个波形。

#### 1.3.1.2 实验代码

```
name    ad
data    segment
tip     db 0dh, 0ah, 'C: continue', 0dh, 0ah
        db 'E: exit', 0dh, 0ah, '$'
head    db 0dh, 0ah, 'D/A    A/D', 0dh, 0ah, '$'
```

```
6   tab     db  '     ', '$'
7   newline db 0dh, 0ah, '$'
8
9   dist    db 0f1h,0e2h,0d3h,0c4h,0b5h
10          db 0a6h, 97h, 88h, 79h, 6ah
11          db  5bh, 4ch, 3dh, 2eh, 1fh
12          db  10h, 01h,0f2h,0e3h,0d4h
13          db 0c5h,0b6h,0a7h, 98h, 89h
14          db  7ah, 6bh, 5ch, 4dh, 3eh
15
16  num     db 00h
17  data    ends
18
19  stack   segment para stack
20          db 100 dup(?)
21  stack   ends
22
23  code    segment
24          assume cs:code, ds:data, ss:stack
25
26  dispstr macro mess; show string
27          lea dx, mess
28          mov ah, 9
29          int 21h
30          endm
31
32  input   macro
33          mov ah, 1
34          int 21h
35          endm
36
37  start:  mov ax, data
38          mov ds, ax
39
40  mloop:  dispstr tip
41          input
42
43          cmp al, 'C'; continue to transfer
44          je xsfer
45          cmp al, 'c'
46          je xsfer
47
48          cmp al, 'E'
49          je exit
50          cmp al, 'e'
51          je exit
52          jmp mloop
53
54  exit:   mov ah, 4ch
55          int 21h
56
57  xsfer:  dispstr head;transfer
58          xor cx, cx
59          mov cl, 20
60          mov al, [num];achieve different data
61          cmp al, 10
62          jb tloop
63          sub al, 10
```

```
64          mov [num], al
65
66  tloop:  mov bx, cx
67          add bl, [num]
68          mov al, [dist + bx]
69          call show
70
71          mov dx, 280h
72          out dx, al
73          call delay
74
75          dispstr tab
76          mov dx, 289h
77          out dx, al
78          call delay
79
80          in al, dx
81          call show
82          dispstr newline
83
84          loop tloop
85          add [num], 3
86          jmp mloop
87
88  show    proc
89          push ax
90          mov bl, al
91          shr al, 1
92          shr al, 1
93          shr al, 1
94          shr al, 1
95          cmp al, 9
96          jbe deci1
97          add al, 7
98  deci1:  add al, 30h
99          mov dl, al
100         mov ah, 2
101         int 21h
102         mov al, bl
103         and al, 0fh
104         cmp al, 9
105         jbe deci2
106         add al, 7
107 deci2:  add al, 30h
108         mov dl, al
109         mov ah, 2
110         int 21h
111         pop ax
112         ret
113 show    endp
114
115 delay   proc
116         push cx
117         mov cx, 0fffh
118 dloop:  dec cx
119         jnz dloop
120         pop cx
121         ret
```

```
122 delay   endp
123
124 code    ends
125 end     start
```

#### 1.3.1.3 调试中遇到的问题及解决

这是我做的第一个实验，首先遇到的问题是实验箱的 bug。在 TCP-USB 环境下运行程序后经常会报错，咨询了助教之后，我用重启实验箱的方式解决了问题。

后续的实验中这个问题也出现了好多次，每次都我通过重启实验箱的方式解决了，这多亏了我及时向助教咨询。

### 1.3.2 A/D 转换

使用 ADC0809 实现 A/D 转换。自动重复采集 20 组不同的数据，在 CRT 上将每组数据对应显示成如下格式：

| D/A | A/D |
|---|---|
| xx | xx |
| xx | xx |
| ⋮ | ⋮ |

然后等待键盘输入，若键入字母 C 则接着往下再做 20 组数据；若键入字母 E 退回 DOS。输入字母应大小写兼容。

#### 1.3.2.1 编程提示

- 在上述 D/A—A/D 过程中，数据输出给 0832 后要延迟一段时间才产生稳定的模拟电压，可调用延时子程序，或进行显示调用来实现
- 若通过 AH=2 的 DOS 功能调用显示数据，寄存器 AL 的内容会被破坏，要注意保护。

#### 1.3.2.2 实验代码

```
1  name    ad
2  data    segment
3  tip     db 0dh, 0ah, 'C: continue', 0dh, 0ah
4          db 'E: exit', 0dh, 0ah, '$'
5  head    db 0dh, 0ah, 'D/A    A/D', 0dh, 0ah, '$'
6  tab     db '      ', '$'
7  newline db 0dh, 0ah, '$'
8
9  dist    db 0f1h,0e2h,0d3h,0c4h,0b5h
10         db 0a6h, 97h, 88h, 79h, 6ah
11         db  5bh, 4ch, 3dh, 2eh, 1fh
12         db  10h, 01h,0f2h,0e3h,0d4h
13         db 0c5h,0b6h,0a7h, 98h, 89h
14         db  7ah, 6bh, 5ch, 4dh, 3eh
15
16 num     db 00h
```

```
17  data    ends
18
19  stack   segment para stack
20          db 100 dup(?)
21  stack   ends
22
23  code    segment
24          assume cs:code, ds:data, ss:stack
25
26  dispstr macro mess; show string
27          lea dx, mess
28          mov ah, 9
29          int 21h
30          endm
31
32  input   macro
33          mov ah, 1
34          int 21h
35          endm
36
37  start:  mov ax, data
38          mov ds, ax
39
40  mloop:  dispstr tip
41          input
42
43          cmp al, 'C'; continue to transfer
44          je xsfer
45          cmp al, 'c'
46          je xsfer
47
48          cmp al, 'E'
49          je exit
50          cmp al, 'e'
51          je exit
52          jmp mloop
53
54  exit:   mov ah, 4ch
55          int 21h
56
57  xsfer:  dispstr head;transfer
58          xor cx, cx
59          mov cl, 20
60          mov al, [num];achieve different data
61          cmp al, 10
62          jb tloop
63          sub al, 10
64          mov [num], al
65
66  tloop:  mov bx, cx
67          add bl, [num]
68          mov al, [dist + bx]
69          call show
70
71          mov dx, 280h
72          out dx, al
73          call delay
74
```

```
75          dispstr tab
76          mov dx, 289h
77          out dx, al
78          call delay
79
80          in al, dx
81          call show
82          dispstr newline
83
84          loop tloop
85          add [num], 3
86          jmp mloop
87
88  show    proc
89          push ax
90          mov bl, al
91          shr al, 1
92          shr al, 1
93          shr al, 1
94          shr al, 1
95          cmp al, 9
96          jbe deci1
97          add al, 7
98  deci1:  add al, 30h
99          mov dl, al
100         mov ah, 2
101         int 21h
102         mov al, bl
103         and al, 0fh
104         cmp al, 9
105         jbe deci2
106         add al, 7
107 deci2:  add al, 30h
108         mov dl, al
109         mov ah, 2
110         int 21h
111         pop ax
112         ret
113 show    endp
114
115 delay   proc
116         push cx
117         mov cx, 0fffh
118 dloop:  dec cx
119         jnz dloop
120         pop cx
121         ret
122 delay   endp
123
124 code    ends
125 end     start
```

#### 1.3.2.3 调试中遇到的问题及解决

本实验进行得较为顺利。在编程的过程中，为了实现两次的 20 组数据不完全一样，我事先在数据取定义了 30 组数据，并设置了 num 变量来控制偏移量。每次采集后 num 变为 (num+3) % 10。由于 3 和 10 互质，这

样的做法可以保证连续 10 次的采集数据都不完全相同，这个想法比较有创新性。

# 2 计算机原理及应用综合实验

## 2.1 实验目的

综合汇编语言编程及 I/O 接口的知识，提高实际应用的能力。

## 2.2 实验准备

在充分复习有关知识的基础上，根据各部分实验内容，画出实验电路原理图，编写汇编语言源程序。

## 2.3 实验内容

以可编程并行接口电路 8255 的应用为主，由易到难分成几个内容。

### 2.3.1 方式 0 I/O（必做 1）

将实验台上的 8255 电路 A 口设置成方式 0 输入，检测 8 只开关的状态；(拨到上面为“1”，拨到下面为“0”)；C 口设置成方式 0 输出，控制 8 只 LED(输入“1”信号时发光，输入“0”信号时熄灭)。实验程序运行后不断地读入 8 只开关状态，然后送对应 LED 显示。直至在计算机键盘上敲空格键退回 DOS。

#### 2.3.1.1 实验代码

```
data    segment
addr    dw 280h
data    ends
stack   segment para stack
        db 100 dup(?)
stack   ends
code    segment
        assume cs:code, ds:data, ss:stack

delay   proc
        push cx
        push dx

        mov dx, 01h
loop1:  mov cx, 0FFh
loop2:  loop loop2
        dec dx
        jnz loop1

        pop dx
        pop cx
        ret
delay   endp

chkkey  macro;check keyboard
        mov ah, 1
        int 16h
```

```
28          endm
29
30  readkey macro;read keyboard
31          mov ah, 0
32          int 16h
33          endm
34
35  start:  mov ax, data
36          mov ds, ax
37
38          mov dx, 283h; conntrol word
39          mov al, 10010000b
40          out dx, al
41
42  mloop:  chkkey
43          jz io
44          readkey
45          cmp al, ' '
46          jz exit
47  io:     mov dx, addr; A input
48          in al, dx
49          call delay
50          mov dx, 282h
51          out dx, al; C output
52          call delay
53          jmp mloop
54
55  exit:   mov ah, 4ch
56          int 21h
57  code    ends
58  end     start
```

#### 2.3.1.2 调试中遇到的问题及解决

本次实验中我遇到了这样的问题：运行程序后 LED 没有任何反应。我首先用导线将电源直接引到 LED 上发现 LED 亮了，排除了其故障的可能。于是我认为是我的 8255 没有设置好。我检查程序，发现之前我为了提高程序的可扩展性，将 280h 定义为 addr，后续通过 addr+3 来写控制字。我认为很可能是这个运算没有正常进行。于是我将 addr+3、addr+2 等语句均改成了显式的 283h、282h，产生了正确的结果。

后来我通过 debug 查看反汇编结果，发现 addr+3 这样的语句反汇编后出现了 [0003]，而不是 283h。这时我才意识到，由于 addr 定义在数据段的最开始，addr=0000。而 addr+3 其实是对 0000 附加 3 的偏移，而不是对内存中的数据 280h 附加 3 的偏移。

### 2.3.2 选做 1

修改程序，A 口保持方式 0 输入开关状态，C 口仍以方式 0 输出 LED：

- 若仅 K7=1 则 8 只 LED 的状态循环左移；
- 若仅 K6=1 则 8 只 LED 的状态循环右移；
- 若仅 K7=K6=1 则 8 只 LED 一起闪烁。

#### 2.3.2.1 实验代码

```
data    segment
addr    dw 280h
data    ends
stack   segment para stack
        db 100 dup(?)
stack   ends
code    segment
        assume cs:code, ds:data, ss:stack

delay   proc
        push cx
        push dx

        mov dx, 0FFh
loop1:  mov cx, 0FFFFh
loop2:  loop loop2
        dec dx
        jnz loop1

        pop dx
        pop cx
        ret
delay   endp

chkkey  macro;check keyboard
        mov ah, 1
        int 16h
        endm

readkey macro;read keyboard
        mov ah, 0
        int 16h
        endm

start:  mov ax, data
        mov ds, ax

        mov dx, 283h; conntrol word
        mov al, 10010000b
        out dx, al

        mov dx, addr
        in al, dx
        mov bl, al

mloop:  chkkey
        jz input
        readkey
        cmp al, ' '
        jz exit
input:  mov dx, addr; A input
        in al, dx
        call delay

        mov cl, al; get the k7 and k6
```

```
56          and al, 11000000b
57
58          cmp al, 11000000b;11
59          jnz next11
60          not bl
61  next11: cmp al, 01000000b;01
62          jnz next01
63          ror bl, 1
64  next01: cmp al, 10000000b;10
65          jnz output
66          rol bl, 1
67
68  output: mov al, bl
69          mov dx, 282h
70          out dx, al; C output
71          call delay
72          jmp mloop
73
74  exit:   mov ah, 4ch
75          int 21h
76  code    ends
77  end     start
```

#### 2.3.2.2 调试中遇到的问题及解决

本次实验刚开始看到的闪烁/循环频率过高。这一点很好解决，我修改了代码中延迟程序中的循环次数，就成功地将频率调整为适合人眼观察的频率了。

### 2.3.3 方式 1 I/O（必做 2）

输入选通信号 STBA 接实验台上单脉冲按键输出插孔，输入缓冲器满信号 IBFA 可接一 LED 显示其状态，中断请求信号 INTRA 接实验台总线区的 IRQ 插孔。实验中每按一次单脉冲按键，通过 8255 电路发一次中断请求。CRT 上显示一个 A 口的 ASC 码字符，直到 A 口数据为 FFH 退出。

#### 2.3.3.1 实验代码

```
1   data    segment
2   keep_ip dw 0
3   keep_cs dw 0
4   flag    db 0
5   data    ends
6
7   stack   segment para stack
8           db 100 dup(?)
9   stack   ends
10
11  code    segment
12          assume  cs:code, ds:data, ss:stack
13
14  int_pro proc
15          mov flag, 1
16          mov dx, 280h
17          in al, dx
```

```
18          mov cl, al
19          mov al, 20h         ;EOI
20          out 20h, al
21          iret
22  int_pro endp
23
24  start:  mov ax, data
25          mov ds, ax
26          ; 8255 init
27          mov dx, 283h
28          mov al, 10110000b   ; A mode 1, input
29          out dx, al
30
31          mov dx, 283h
32          mov al, 00001001b   ; set INTEA
33          out dx, al
34
35          ;keep interupt vector
36          mov ah, 35h
37          mov al, 0bh
38          int 21h
39          mov keep_cs, es
40          mov keep_ip, bx
41
42          ;load interupt vector
43          push ds
44          mov dx, offset int_pro
45          mov ax, seg int_pro
46          mov ds, ax
47          mov ah, 25h
48          mov al, 0bh
49          int 21h
50          pop ds
51
52          in al, 21h
53          and al, 0f7h        ; enable the IRQ
54          out 21h, al
55
56  mloop:  mov bl, flag
57          cmp bl, 1
58          jz  hasint            ;has interupting
59          jmp mloop
60
61  hasint: cmp cl, 0ffh
62          jz exit
63          mov dl, cl
64          mov ah, 2
65          int 21h
66          mov flag, 0
67          jmp mloop
68
69  exit:   mov al, 0ffh        ; disable IRQS
70          out 21h, al
71
72          ; reset interupt vector
73          push ds
74          mov dx, keep_ip
75          mov ax, keep_cs
```

```
        mov ds, ax
        mov ah, 25h
        mov al, 0bh
        int 21h
        pop ds

        mov ah, 4ch
        int 21h


code    ends
end     start
```

#### 2.3.3.2 调试中遇到的问题及解决

我在本实验的接线上遇到了一些困难。一开始我试图寻找 IRQ3 的接口，然而没有找到。后来我回忆起当时做的 8259 实验，知道了实验箱的 IRQ 其实在内部已经与 8259 的 IRQ3 相连了。正确接线后，我顺利地完成了实验。

### 2.3.4 选做 2

修改主程序实现密码检测功能：连续从 A 口拨入两次数据，与计算机内事先存放的两字节数比较，相符则在 CRT 上显示“OK”，否则重新输入。

#### 2.3.4.1 实验代码

```
data    segment
keep_ip dw 0
keep_cs dw 0
pwd     dw 0000H
yes     db 0dh, 0ah, 'OK', 0dh, 0ah, '$'
no      db 0dh, 0ah, 'WRONG', 0dh, 0ah, '$'
tip     db 'Input your password', 0dh, 0ah, '$'
tip1    db 'input next two digits', 0dh, 0ah, '$'
data    ends

stack   segment para stack
        db 100 dup(?)
stack   ends

code    segment
        assume  cs:code, ds:data, ss:stack

int_pro proc
        mov dx, 280h
        in al, dx
        mov cl, 1
        mov bl, al

        mov al, 20h         ;EOI
        out 20h, al
        iret
int_pro endp
```

```
28
29  dispstr macro string
30          lea dx, string
31          mov ah, 9
32          int 21h
33          endm
34
35  start:  mov ax, data
36          mov ds, ax
37          ; 8255 init
38          mov dx, 283h
39          mov al, 10110000b   ; A mode 1, input
40          out dx, al
41
42          mov dx, 283h
43          mov al, 00001001b   ; set INTEA
44          out dx, al
45
46          ;keep interupt vector
47          mov ah, 35h
48          mov al, 0bh
49          int 21h
50          mov keep_cs, es
51          mov keep_ip, bx
52
53          ;load interupt vector
54          push ds
55          mov dx, offset int_pro
56          mov ax, seg int_pro
57          mov ds, ax
58          mov ah, 25h
59          mov al, 0bh
60          int 21h
61          pop ds
62
63          in al, 21h
64          and al, 0f7h        ; enable the IRQ
65          out 21h, al
66
67          dispstr tip
68
69  next:   hlt                 ;first 2 digits
70          cmp cl, 1
71          jnz next
72          mov bh, bl
73          mov cl, 0
74          dispstr tip1
75
76  next1:  hlt                 ;next 2 digits
77          cmp cl, 1
78          jnz next1
79          mov cl, 0
80          cmp bx, pwd
81          jz right
82
83  wrong:  dispstr no
84          dispstr tip
85          jmp next
```

```
86
87  right:  dispstr yes
88          mov al, 0ffh        ; disable IRQS
89          out 21h, al
90
91          ; reset interupt vector
92          push ds
93          mov dx, keep_ip
94          mov ax, keep_cs
95          mov ds, ax
96          mov ah, 25h
97          mov al, 0bh
98          int 21h
99          pop ds
100
101         mov ah, 4ch
102         int 21h
103
104
105 code    ends
106 end     start
```

#### 2.3.4.2 调试中遇到的问题及解决

本实验在上一个实验的基础上改动不大，故进行得比较顺利，没有遇到问题。

### 2.3.5 选做 3

将 8255 电路 A 口改成方式 1 输出 (仅将 PA7 接一只 LED 示范即可)，修改前面程序实现每次中断后，通过 A 口输出数据控制 LED 状态在 0，1 之间来回翻转。

#### 2.3.5.1 实验代码

```
1   data    segment
2   keep_ip dw 0
3   keep_cs dw 0
4   data    ends
5
6   stack   segment para stack
7           db 100 dup(?)
8   stack   ends
9
10  code    segment
11          assume  cs:code, ds:data, ss:stack
12
13  int_pro proc
14          push ax
15          not bl              ;inverse
16          mov dx, 280h
17          mov al, bl
18          out dx, al
19
20          mov al, 20h         ;EOI
21          out 20h, al
```

```
22          pop ax
23          iret
24  int_pro endp
25
26  start:  mov ax, data
27          mov ds, ax
28
29          ; 8255 init
30          mov dx, 283h
31          mov al, 10100000b    ; A mode 1, output
32          out dx, al
33
34          mov dx, 283h
35          mov al, 00001101b; set INTEA
36          out dx, al
37
38          ;keep interupt vector
39          mov ah, 35h
40          mov al, 0bh
41          int 21h
42          mov keep_cs, es
43          mov keep_ip, bx
44
45          ;load interupt vector
46          push ds
47          mov dx, offset int_pro
48          mov ax, seg int_pro
49          mov ds, ax
50          mov ah, 25h
51          mov al, 0bh
52          int 21h
53          pop ds
54
55          in al, 21h
56          and al, 0f7h     ; enable the IRQ
57          out 21h, al
58
59          mov bl, 0ffh      ;init
60
61  mloop:  mov ah, 1
62          int 16h
63          jnz exit;
64          hlt
65          jmp mloop
66
67  exit:   mov al, 0ffh     ; disable IRQS
68          out 21h, al
69
70          ; reset interupt vector
71          push ds
72          mov dx, keep_ip
73          mov ax, keep_cs
74          mov ds, ax
75          mov ah, 25h
76          mov al, 0bh
77          int 21h
78          pop ds
79
```

```
        mov ah, 4ch
        int 21h


code    ends
end     start
```

#### 2.3.5.2 调试中遇到的问题及解决

由于本实验中 8255 改为方式 1 输出，其接线有所变化。一开始我没有注意到这一点，导致没有得到任何结果；后来我参照教材中的内容修改了接线，顺利地完成了任务。

### 2.3.6 扫描显示接口电路（必做 3）

利用扫描显示的方法，一片 8255 电路可以管理多位七段 LED 数码管显示不同数字。A 口以方式 0 输出控制数码管的段码输入端，C 口以方式 0 输出控制位码输入。每一位显示时调用延时，延时结束后再切换显示下一位。如此循环，只要切换的频率足够高，就可以看到“01”在两位数码管上同时显示。

#### 2.3.6.1 实验代码

```
data    segment
data    ends

stack   segment
        db 100 dup(?)
stack   ends

code    segment
        assume cs:code, ds:data, ss:stack

chkkey  macro;check keyboard
        mov ah, 1
        int 16h
endm

readkey macro;read keyboard
        mov ah, 0
        int 16h
        endm

delay   proc
        push cx
        push dx
        mov dx, 08h
loop1:  mov cx, 0ffffh
loop2:  loop loop2
        dec dx
        jnz loop1

        pop dx
        pop cx
        ret
delay   endp
```

```
34
35 start:  mov ax, data
36        mov ds, ax
37
38        mov dx, 283h
39        mov al, 10000000b
40        out dx, al
41
42 mloop:  chkkey
43        jz next
44        readkey
45        cmp al, ' '
46        jz exit
47 next:   mov dx, 282h    ;turn off both the digits
48        mov al, 00h
49        out dx, al
50        ; output 0
51        mov dx, 280h
52        mov al, 3fh
53        out dx, al
54        mov dx, 282h
55        mov al, 01h
56        out dx, al
57
58        call delay
59
60        ;turn off
61        mov dx, 282h
62        mov al, 00h
63        out dx, al
64
65        ;output 1
66        mov dx, 280h
67        mov al, 06h
68        out dx, al
69        mov dx, 282h
70        mov al, 02h
71        out dx, al
72
73        call delay
74
75        jmp mloop
76
77 exit:   mov ah, 4ch
78        int 21h
79 code    ends
80 end     start
```

#### 2.3.6.2 调试中遇到的问题及解决

由于之前在数电中曾经使用 FPGA 做过数码管的扫描显示，我对其原理比较熟悉，所以实验比较顺利。

### 2.3.7 选做 4

程序运行后，从计算机键盘上输入两位十进制数，分别在两个数码管上显示。若继续输入数字则更新显示。若发现输入了非数字键则退回 DOS。

#### 2.3.7.1 实验代码

```
data    segment
dig     db 3fh, 06h, 5bh, 4fh, 66h
        db 6dh, 7dh, 07h, 7fh, 6fh
data    ends

stack   segment
        db 100 dup(?)
stack   ends

code    segment
        assume cs:code, ds:data, ss:stack

chkkey  macro;check keyboard
        mov ah, 1
        int 16h
endm

readkey macro;read keyboard
        mov ah, 0
        int 16h
        endm

delay   proc
        push cx
        push dx
        mov dx, 02h
loop1:  mov cx, 0ffffh
loop2:  loop loop2
        dec dx
        jnz loop1

        pop dx
        pop cx
        ret
delay   endp

start:  mov ax, data
        mov ds, ax

        mov dx, 283h
        mov al, 10000000b
        out dx, al

        xor bx, bx
        xor cx, cx

mloop:  chkkey
        jz next
        readkey     ;read one digit
```

```
50          cmp al, '0'
51          jb exit
52          cmp al, '9'
53          ja exit
54          sub al, 30h
55          mov bh, bl  ;mov old digit to bh
56          mov bl, al  ;mov new digit to bl
57
58  next:   ; output lower digit
59          push bx
60          mov dx, 280h
61          mov bh, 0
62          mov al, [dig + bx]
63          out dx, al
64          mov dx, 282h
65          mov al, 01h
66          out dx, al
67          pop bx
68
69          call delay
70
71          ;turn off
72          mov dx, 282h
73          mov al, 00h
74          out dx, al
75
76          ;output higher digit
77          push bx
78          mov bl, bh
79          mov bh, 0
80          mov dx, 280h
81          mov al, [dig + bx]
82          out dx, al
83          mov dx, 282h
84          mov al, 02h
85          out dx, al
86          pop bx
87
88          call delay
89
90          ;turn off
91          mov dx, 282h
92          mov al, 00h
93          out dx, al
94
95          jmp mloop
96
97  exit:   mov ah, 4ch
98          int 21h
99  code    ends
100 end     start
```

#### 2.3.7.2 调试中遇到的问题及解决

我一开始设计的程序是一旦检测到键盘缓冲区有字符，就连着读入两位数字。这样的设计的问题在于，当输入第一个数字时，程序会等待输入下一位数字，这时数码管会熄灭，用户体验较差。于是我临时将程序改为

循环输入，即新输入的位会作为低位，而原来的低位变成高位。这样，每次只需读入一个字符，数码管上会实时显示两位数字。

### 2.3.8 选做 5

用 8253 定时来代替上面的软件延时，8253 定时器自动重复工作，每工作一个周期发出一次中断请求信号 (接至实验台上总线区的 IRQ 插孔)，在中断服务程序中同步更换段码和位码，即可实现扫描显示。

#### 2.3.8.1 实验代码

```
data    segment
keep_cs dw 0
keep_ip dw 0
data    ends

stack   segment
        db 100 dup(?)
stack   ends

code    segment
        assume cs:code, ds:data, ss:stack

chkkey  macro;check keyboard
        mov ah, 1
        int 16h
endm

readkey macro;read keyboard
        mov ah, 0
        int 16h
        endm
int_pro proc
        push ax
        mov dx, 282h    ;turn off both the digits
        mov al, 00h
        out dx, al

        cmp bl, 0
        jnz out1
        ; output 0
out0:   mov dx, 280h
        mov al, 3fh
        out dx, al
        mov dx, 282h
        mov al, 01h
        out dx, al
        mov bl, 1
        jmp int_end

out1:   ;output 1
        mov dx, 280h
        mov al, 06h
        out dx, al
        mov dx, 282h
        mov al, 02h
```

```
46          out dx, al
47          mov bl, 0
48
49  int_end:mov al, 20h      ;EOI
50          out 20h, al
51          pop ax
52          iret
53  int_pro endp
54
55  start:  mov ax, data
56          mov ds, ax
57          ;8253
58          mov dx, 293h
59          mov al, 00110111b
60          out dx, al
61
62          mov dx, 290h
63          mov al, 30h
64          out dx, al
65          out dx, al
66
67          ;8255
68          mov dx, 283h
69          mov al, 10000000b
70          out dx, al
71
72          ;interupt
73          mov ah, 35h
74          mov al, 0bh
75          int 21h
76          mov keep_ip, bx
77          mov keep_cs, es
78
79          push ds
80          mov dx, offset int_pro
81          mov ax, seg int_pro
82          mov ds, ax
83          mov ah, 25h
84          mov al, 0bh
85          int 21h
86          pop ds
87
88          in al, 21h
89          and al, 11110111b
90          out 21h, al
91
92          mov bl, 0
93  mloop:  hlt
94          chkkey
95          jz mloop
96          cmp al, ' '
97          jnz mloop
98
99
100
101 exit:   in al, 21h
102         or al, 00001000b
103         out 21h, al
```

```
104
105         ;reset interupt vector
106         push ds
107         mov dx, keep_ip
108         mov ax, keep_cs
109         mov dx, ax
110         mov ah, 25h
111         mov al, 0bh
112         int 21h
113         pop ds
114
115         mov ah, 4ch
116         int 21h
117 code    ends
118 end     start
```

#### 2.3.8.2 调试中遇到的问题及解决

有了必做 3 的基础，这个任务并不算困难。刚开始由于分频系数设置得过小，数码管出现了一些重影。不过这个问题很好解决，我经过几次尝试，对 8253 写入一个合适的分频系数，顺利地完成了任务。