第7章 分支与循环程序设计

- 程序设计的一般步骤:
  1.分析问题，确定算法和数据结构。
  2.根据算法绘制程序流程图。
  3.根据流程图编写程序。
  4.上机调试程序。

- 程序有顺序, 分支,循环,子程序4种结构。

7.1 分支程序设计

- 7.1.1 分支程序结构
- 7.1.2 单分支程序
- 7.1.3 复合分支程序
- 7.1.4 多分支程序

程序转移指令

- 无条件转移指令
- 条件转移指令
- 循环指令
- 子程序调用指令
- 中断调用指令

无条件转移指令

- JMP 跳转指令：无条件转移到指令指定的地址去执行程序。
- 转移的目标地址和本跳跳转指令在同一个代码段，则为段内转移；否则是段间转移。
- 转移的目标地址在跳转指令中直接给出，则为直接转移；否则是间接转移。

（1）段内直接转移

- 格式: JMP NEAR PTR OPR
- 操作: IP←IP+16位位移量
- NEAR PTR为目标地址OPR的属性说明，表明是一个近（段内）跳转，通常可以省略。
- 位移量是带符号数，IP的值可能减小（程序向后跳），也可能增加（程序向前跳）。
- 程序的重新定位并不影响程序的正确执行。

- 例5.47　关于程序的可重新定位的讨论。

```
    1000：JMP  P1        ；1000H是本条指令的所在偏移地址
    1002：MOV  AX，  BX
    1004：MOV  DX，  CX
P1：ADD  AX，  DX    ；P1是标号，其值为1006H
```

- 如果把这个程序放在内存中的另一个位置，如下所示：

```
    2000：JMP  P1        ；2000H是本条指令的所在偏移地址
    2002：MOV  AX，  BX
    2004：MOV  DX，  CX
P1：ADD  AX，  DX    ；P1是标号，其值为2006H
```

- 显然这两段程序是一样的，无论在内存什么位置，不应影响运行结果。

```
-a1000
073F:1000 jmp 1006
073F:1002 mov ax,bx
073F:1004 mov dx,cx
073F:1006 add ax,dx
073F:1008 nop
073F:1009
-a2000
073F:2000 jmp 2006
073F:2002 mov ax,bx
073F:2004 mov dx,cx
073F:2006 add ax,dx
073F:2008 nop
073F:2009
-u1000L2
073F:1000 EB04          JMP     1006
-u2000L2
073F:2000 EB04          JMP     2006
-_
```

（2）段内间接转移

- 格式: JMP WORD PTR OPR
- 操作: IP←（EA）
- 可以使用除立即数以外的任何一种寻址方式。

- 例5.48 如果BX=2000H，DS=4000H，(42000H)=6050H，(44000H)=8090H，TABLE的偏移地址为2000H，分析下面四条指令单独执行后IP的值。

  JMP BX ；寄存器寻址，IP=BX

  JMP WORD PTR [BX] ；寄存器间接寻址，IP=[DS:BX]

  JMP WORD PTR TABLE ；直接寻址，IP=[DS:TABLE]

  JMP TABLE[BX] ；寄存器相对寻址，IP=[DS:(TABLE+BX)]

  第一条指令执行后，IP=BX=2000H。

  第二条指令执行后，
  IP=(DS:2000H)=(40000H+2000H)=(42000H)=6050H。

  第三条指令执行后，
  IP=(DS:2000H)=(40000H+2000H)=(42000H)=6050H。

  第四条指令执行后，
  IP=(DS:4000H)=(40000H+4000H)=(44000H)=8090H。

（3）段间直接转移

- 格式: JMP FAR PTR OPR
- 操作: IP←OPR的偏移地址
  CS←OPR所在段的段地址

（4）段间间接转移

- 格式: JMP DWORD PTR OPR
- 操作: IP←（EA）
  CS←（EA+2）
- 可以使用除立即数和寄存器方式以外的任何一种寻址方式。

- 例5.49　如果BX=2000H，DS=4000H，(42000H)=6050H，(42002H)=1234H，指出下面指令执行后IP和CS的值。

  JMP　DWORD PTR [BX]

  指令执行后，
  IP=(DS:2000H)=(40000H+2000H)=(42000H)=6050H；CS=(42002H)=1234H。

条件转移指令

- 条件转移指令根据上一条指令所设置的标志位来判别测试条件，从而决定程序转向。
- 通常在使用条件转移指令之前，应有一条能产生标志位的前导指令，如**CMP**指令。
- 汇编指令格式中，转向地址由标号表示。
- 所有的条件转移指令都不影响标志位。

- 第一组: 根据单个条件标志的设置情况转移
- 第二组: 测试CX寄存器的值为0则转移
- 第三组: 比较两个无符号数,根据结果转移
- 第四组: 比较两个有符号数, 根据结果转移

(1) 根据单个条件标志的设置情况转移

- JZ（JE） 结果为零转移
  格式: JZ OPR
  测试条件:ZF=1
- JNZ（JNE） 结果不为零转移
  格式: JNZ OPR
  测试条件:ZF=0
- JS 结果为负转移
  格式: JS OPR
  测试条件:SF=1

- JNS OPR　结果不为负（为正）转移
  测试条件:SF=0
- JO OPR　结果溢出转移
  测试条件:OF=1
- JNO OPR　结果不溢出转移
  测试条件:OF=0
- JP (JPE)　奇偶位为1转移
  格式: JP　OPR
  测试条件:PF=1

- JNP (JPO)　奇偶位为0转移
  格式: JNP　OPR
  测试条件:PF=0
- JB　(JNAE,JC)　低于,(不高于等于,进位位为1),则转移.
  格式: JB　OPR
  测试条件:CF=1
- JNB　(JAE,JNC)　不低于,(高于等于,进位位为0),则转移.
  格式: JNB　OPR
  测试条件:CF=0

(2) 测试CX寄存器的值为0则转移

- 格式: JCXZ　OPR
- 测试条件:CX=0

(3) 比较两个无符号数,根据结果转移

- JB (JNAE,JC) 低于,(不高于或等于,进位位为1),则转移.
  格式: JB OPR
  测试条件:CF=1
- JNB (JAE,JNC) 不低于,(高于等于,进位位为0),则转移.
  格式: JNB OPR
  测试条件:CF=0
- JBE (JNA) 低于或等于,(不高于),则转移.
  格式: JBE OPR
  测试条件:CF OR ZF=1
- JNBE (JA) 不低于或等于,(高于),则转移.
  格式: JNBE OPR
  测试条件:CF OR ZF=0

(4) 比较两个带符号数,根据结果转移

- JL (JNGE) 小于,(不大于等于),则转移. <
  格式: JL OPR
  测试条件:SF XOR OF=1
- JNL (JGE) 不小于,(大于等于),则转移. >=
  格式: JNL OPR
  测试条件:SF XOR OF=0
- JLE (JNG) 小于等于,(不大于),则转移. <=
  格式: JLE OPR
  测试条件:(SF XOR OF) OR ZF=1
- JNLE (JG) 不小于等于,(大于),则转移. >
  格式: JNLE OPR
  测试条件:(SF XOR OF) OR ZF=0

表5-3有符号数的比较判断条件

****

为何针对有符号数和无符号数须用不同指令?

****

8位二进制数FFH 和 00H ,哪个大?

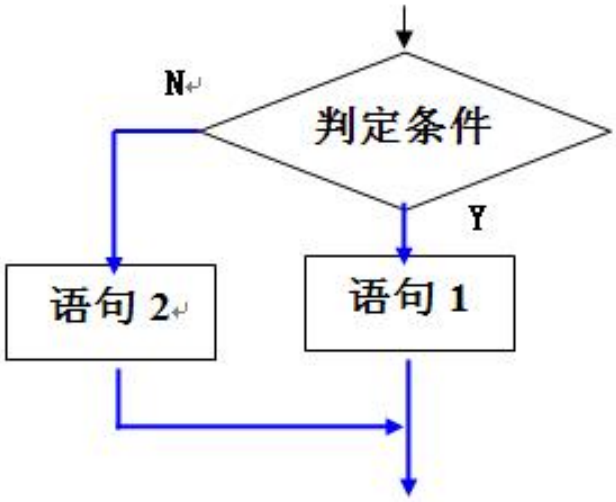
若为无符号数, FFH大,若为有符号数,00H大.

- 例5.50 有一个长为19字节的字符串，首地址为MESS。查找其中的‘空格’(20H) 字符,如找到则继续执行，否则转标号NO。

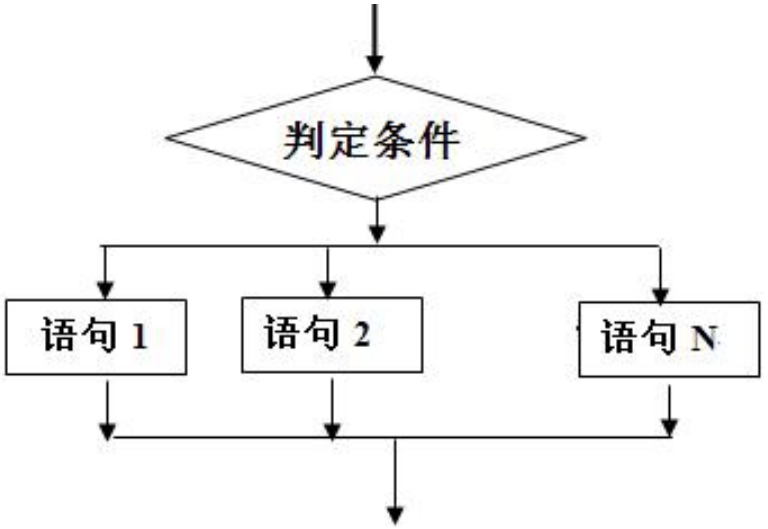
```
    MOV    AL，20H
    MOV    CX，19
    MOV    DI，-1
LK: INC    DI
    DEC    CX
    CMP    AL, MESS[DI]
    JCXZ   NO
    JNE    LK
    …
    …
```

7.1.1 分支程序结构

- 分支程序结构有二种形式: 两个分支和多个分支。



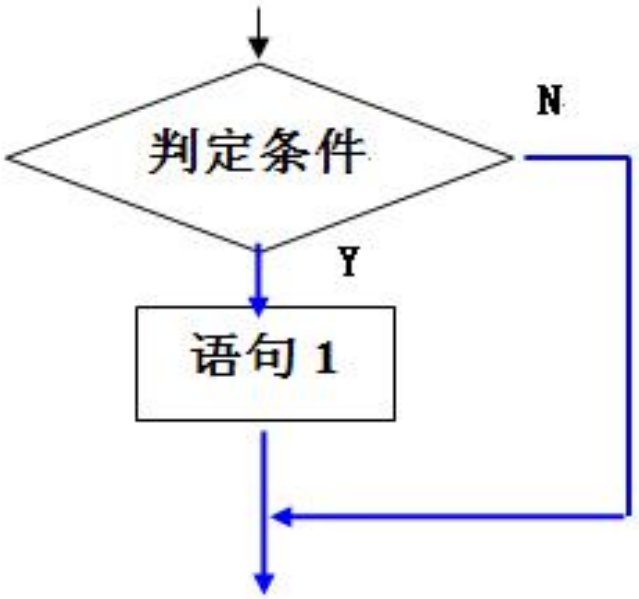
(1) IF THEN ELSE



(2) CASE

7.1.2 单分支结构程序

- 单分支结构程序（IF--THEN）：是分支结构程序的最简单形式。

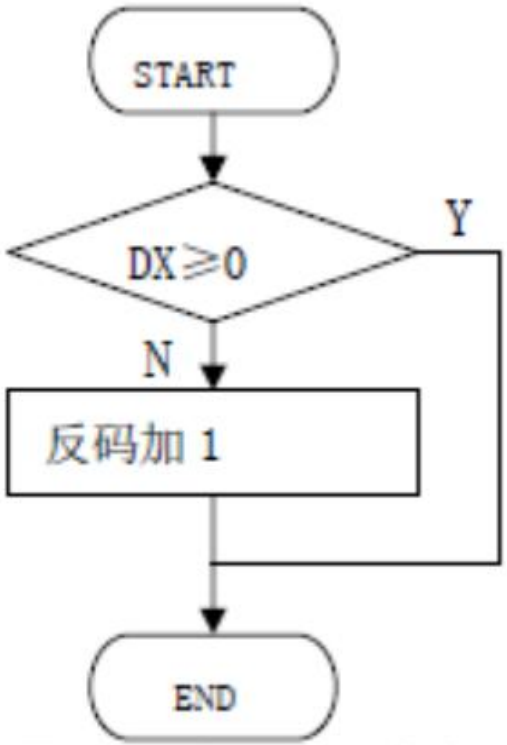


- 例7.1　双字长数存放在DX和AX寄存器中(高位在DX)，求该数的绝对值(用16位指令)。

- 算法分析：

  1. 双字长数高字在DX中，低字在AX中；

  2. 判该数的正负，为正数（最高位为0），该数不处理；为负数，就对该数求补（即反码加1）。


START
DX≥0
Y
N
反码加 1
END

```
code  segment
        assume  cs:code
start:
        test   dx，8000h        ；测试数的正负
        jz     exit             ；不为负数就退出
        not    ax
        not    dx
        add    ax，1
        adc    dx，0
exit:
        mov    ah, 4ch
        int    21h
code  ends
         end    start
```

7.1.3 复合分支程序

- 如果在分支结构中又出现分支，这就是复合分支结构。

- 例7.2　从键盘输入一位十六进制数，并将其转换为十进制数输出显示。

- 算法分析：

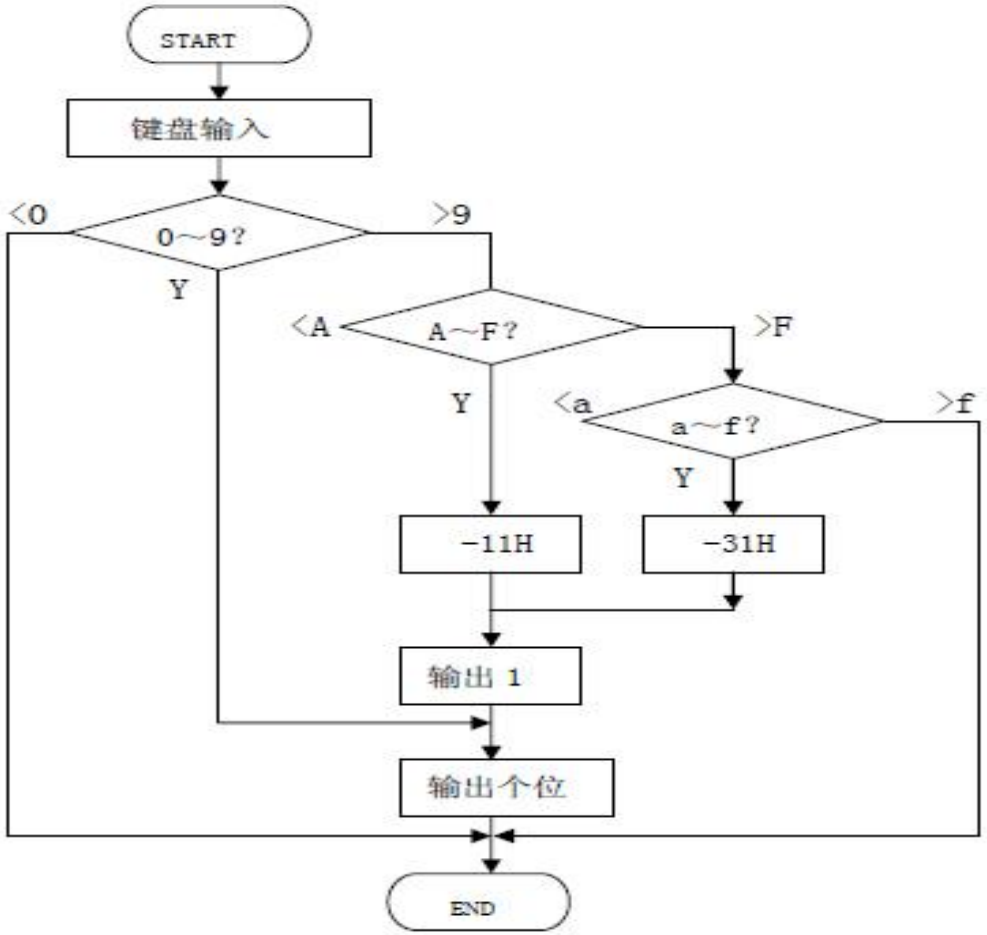
  从键盘输入一个十六进制数，有以下四种情况：

  1. 为数字0～9（ASCII码30～39H），无需处理，直接输出；

  2. 为大写字母A～F（ASCII码41～46H），先输出31H，再输出该数ASCII码-11H；

  3. 为小写字母a～f（ASCII码61～66H），先输出31H，再输出该数ASCII码-31H；

  4. 该数不为0～9、A～F、a～f，是非法字符，应退出程序或输出错误信息。


START
键盘输入
<0
0～9?
>9
Y
<A
A～F?
>F
Y
<a
a～f?
>f
Y
-11H
-31H
输出 1
输出个位
END

```
code    segment
        assume  cs:code
start:  mov   ah，1              ；键盘输入
        int   21h
        cmp   al，30h
        jl    exit               ；非法输入
        cmp   al，39h
        jle   dig                ；输入是数字0～9
        cmp   al，41h
        jl    exit               ；非法输入
        cmp   al，46h
        jle   print              ；输入是大写A～F
        cmp   al，61h
        jl    exit               ；非法输入
```

```
        cmp   al，66h
        jg    exit              ；非法输入
        sub   al，31h
        jmp   out1              ；输入是小写a～f
print：sub   al，11h
out1：mov   dl，31h             ；输出字符1
        mov   ah，2
        push  ax                ；暂存AX
        int   21h               ；int指令改写了AX
        pop   ax                ；恢复AX
dig：   mov   dl，al            ；输出个位
        mov   ah，2
        int   21h
exit：  mov   ah，4ch           ；程序终止并退出
        int   21h
code    ends
        end   start
```

7.1.4 多分支程序

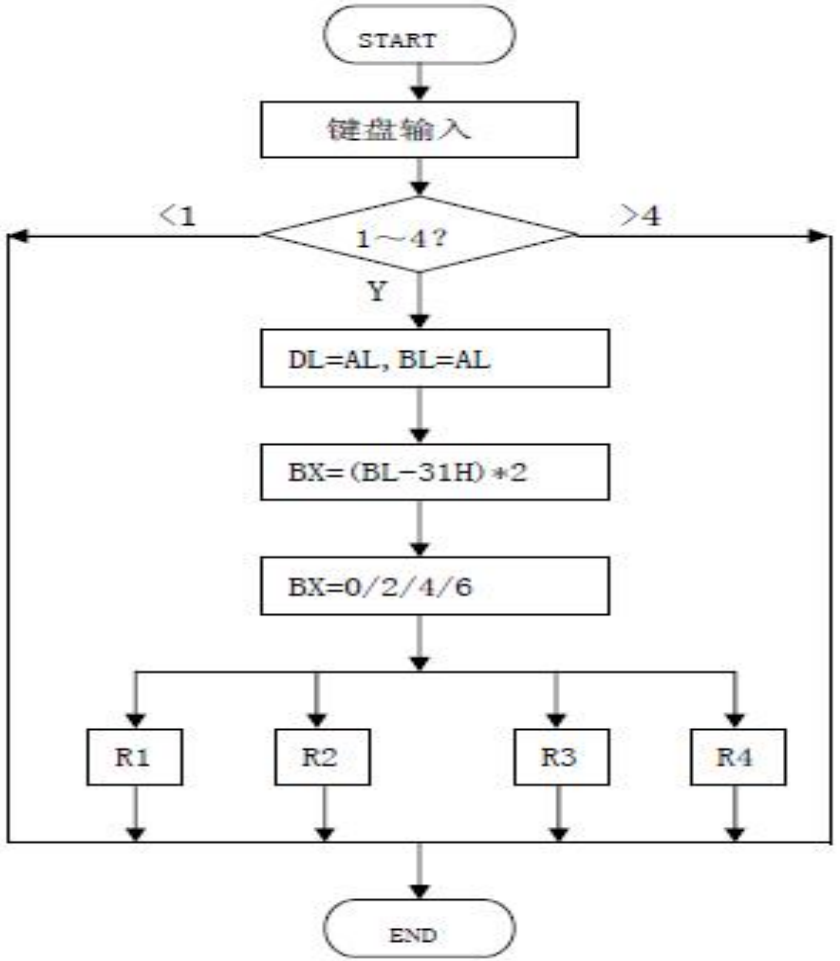
- 如果在分支结构中有超过两个以上的多个可供选择的分支，这就是多分支结构。

- 如果对多分支的条件逐个查询以确定是哪一个分支，只会增加代码和时间，为了尽快进入某个分支，可以采用分支向量表法。

- 例7.3 根据键盘输入的一位数字(1～4)，使程序转移到4个不同的分支中去，以显示键盘输入的数字。

- 算法分析：从键盘输入一个数1～4，

  1. 建立一个分支向量表branch ，集中存放四个分支的偏移地址；
  2. 每个偏移地址位16位，占用2个单元；
  3. 四个分支的偏移地址在转移地址表的地址是：
     转移地址表首址+输入数字（0～3）×2；
  4. 用间接寻址方式转向对应分支。


START
键盘输入
<1
1～4?
>4
Y
DL=AL, BL=AL
BX=(BL-31H)*2
BX=0/2/4/6
R1
R2
R3
R4
END

```
code    segment
        assume  cs:code，ds:code
start:  mov   ax,code          ；ds=cs
        mov   ds,ax
        mov   ah，7            ；键盘输入无回显
        int   21h
        cmp   al，31h
        jl    exit             ；非法输入
        cmp   al，34h
        jg    exit             ；非法输入
        mov   dl，al           ；放入dl，待显示
```

```
        mov   bl，al
        sub   bl，31h          ；转换ascii码为数值
        shl   bl，1            ；(bl)×2，指向分支向量表中某地址
        mov   bh，0
        jmp   branch[bx]       ；转向分支
r1：    mov   ah，2
        int   21h              ；显示键盘输入的数字
        jmp   exit
r2：    mov   ah，2
        int   21h
        jmp   exit
r3：    mov   ah，2
        int   21h
```

```
        jmp   exit
 r4:    mov   ah，2
        int   21h
        jmp   exit
 exit:  mov   ah，4ch          ；程序终止并退出
        int   21h
        branch  dw    r1
                dw    r2
                dw    r3
                dw    r4
code    ends
        end   start
```

7.2 循环程序设计

- 7.2.1 循环程序结构
- 7.2.2 计数循环程序
- 7.2.3 条件循环程序
- 7.2.4 条件计数循环程序
- 7.2.5 多重循环程序

循环指令

- **LOOP** 循环
- **LOOPZ / LOOPE** 为零或相等时循环
- **LOOPNZ / LOOPNE** 不为零或不相等时循环

- 指令：LOOP　OPR

  测试条件:CX ≠ 0，则循环
- 指令：LOOPZ / LOOPE　OPR

  测试条件:ZF=1 AND CX≠0，则循环
- 指令：LOOPNZ / LOOPNE　OPR

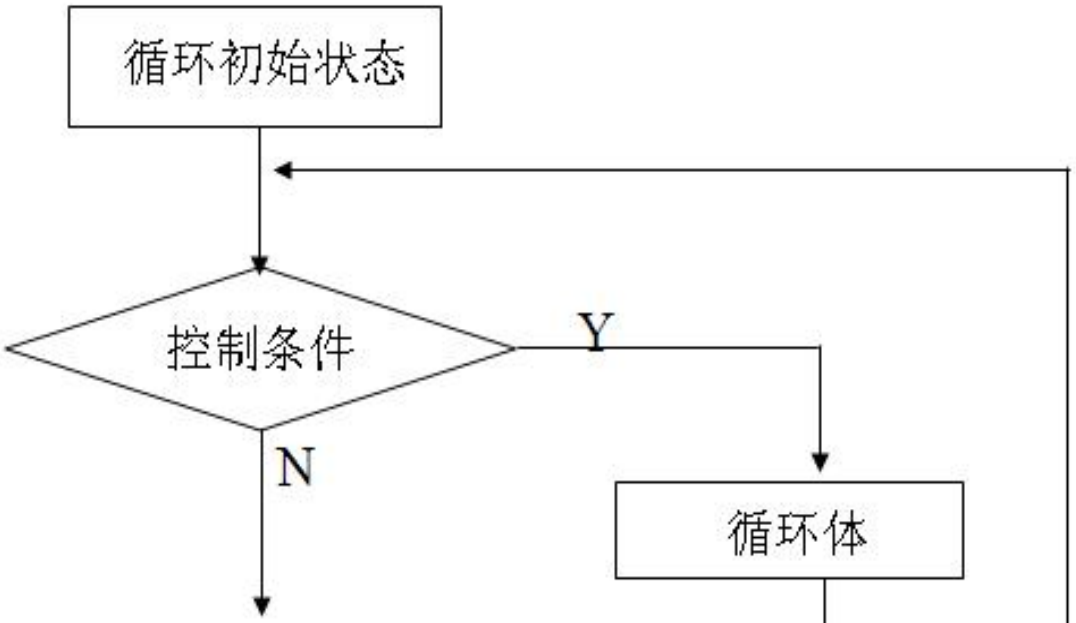
  测试条件:ZF=0 AND CX≠0，则循环
- 操作: 首先CX寄存器减1，然后根据测试条件决定是否转移。

- 例5.51 在首地址为MESS长为19字节的字符串中查找‘空格’(20H) 字符,如找到则继续执行，否则转标号NO。用循环指令实现程序的循环。

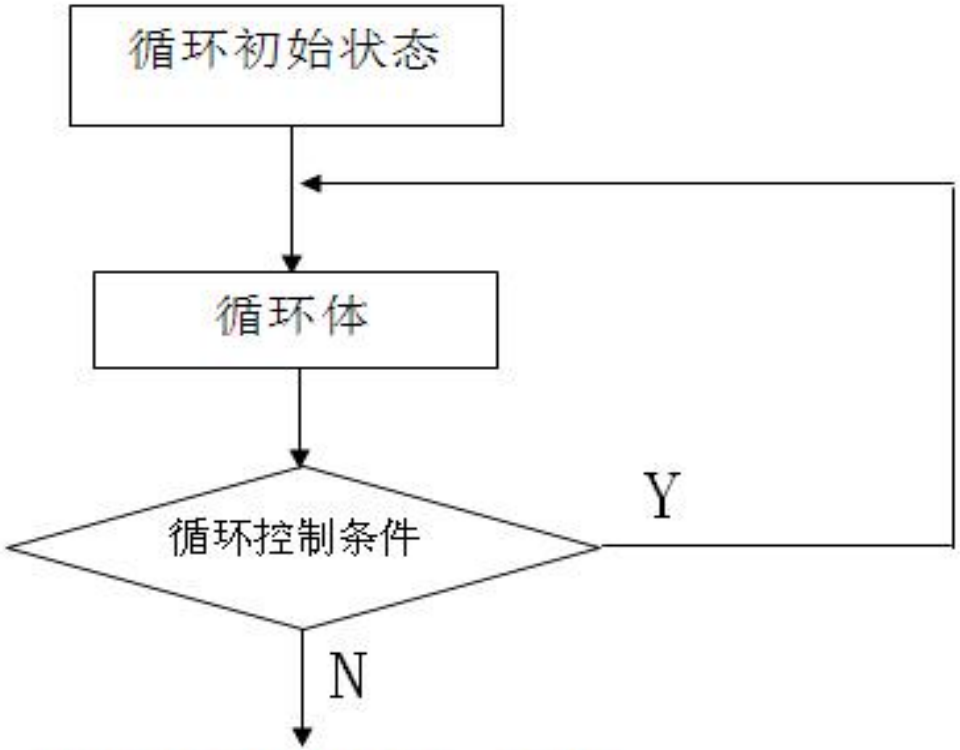
```
    MOV    AL，20H
    MOV    CX，19
    MOV    DI，-1
LK: INC    DI
    CMP    AL, MESS[DI]
    LOOPNE  LK
    JNZ    NO
    …
```

7.2.1 循环程序结构

- 循环程序有两种结构形式： DO---WHILE结构 和 DO---UNTIL结构。
- 循环程序由三部分组成：循环初始状态、循环控制、循环体。
- 循环控制条件有三类：计数循环、条件循环、条件计数循环。



（1）DO WHILE 结构



（2）DO UNTIL 结构

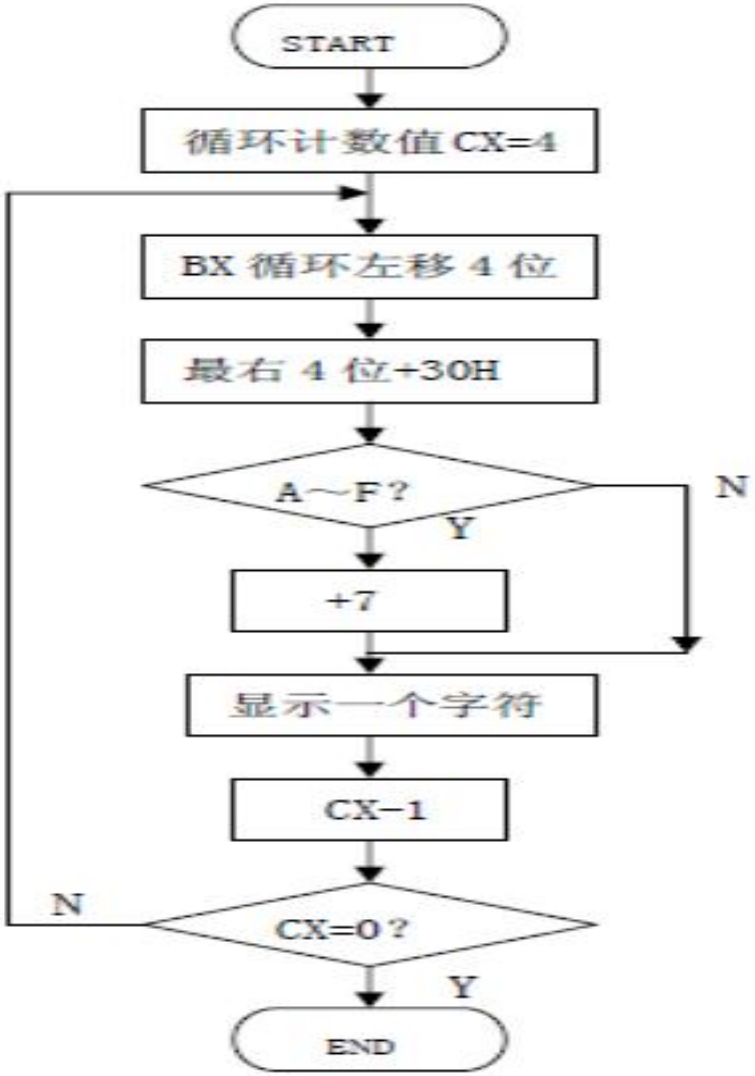
7.2.2 计数循环程序

- 计数循环：用循环计数器的值控制循环。
- 例7.4

- 例7.4　把BX中的二进制数用十六进制显示.（设BX=123AH）

- 算法分析

1. 屏幕显示字符用2号DOS功能调用，DL=输出字符。
2. 屏幕上如何显示 BX=123AH=0001 0010 0011 1010
3. BX=1　　2　　3　　A H,
   1-->31H,　2-->32H,　3-->33H,　A-->41H
4. BX循环左移4位。


START
循环计数值CX=4
BX 循环左移 4 位
最右 4 位+30H
A～F？
N
Y
+7
显示一个字符
CX−1
N
CX=0？
Y
END

```
code    segment
    assume  cs:code
start:  mov   cx，4
        shift：  rol   bx，1          ；连续循环左移4位
        rol   bx，1
        rol   bx，1
        rol   bx，1
        mov   al，bl
        and   al，0fh                 ；取最右4位
        add   al，30h                 ；转为ascii
        cmp   al，39h
```

```
        jle   dig                ；是0～9则转dig
        add   al，7              ；是A～F
dig：   mov   dl，al
      mov   ah，2
      int   21h
      loop  shift
      mov   ah，4ch
      int   21h
code    ends
        end   start
```

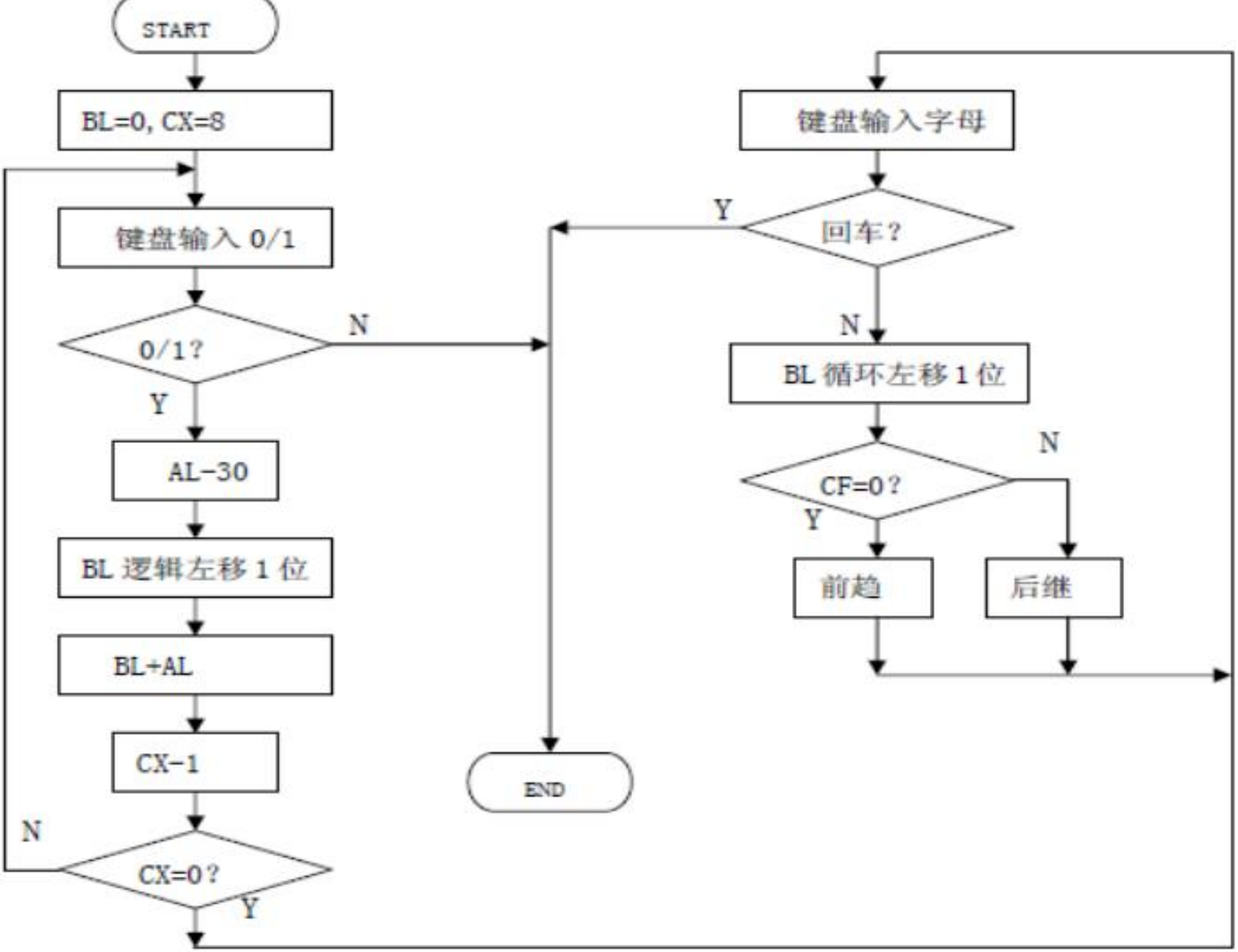
7.2.3 条件循环程序

- 在循环程序中，有时候每次循环所做的操作可能不同，即循环体中有分支的情况，需要依据某一个标志来决定做何操作。标志位为1表示要做操作A，标志位为0表示要做操作B，我们可把这种标志字称为逻辑尺。

- 例7.5 从键盘输入8位二进制数作为逻辑尺。再输入一个英文字母，根据逻辑尺当前的最高位标志显示输出该字母的相邻字符，标志位为0则显示其前趋字符，标志位为1则显示其后继字符。显示相邻字符后，逻辑尺循环左移一位，再接收下一个字母的输入，并依据逻辑尺显示相邻字符，直到回车键结束程序。

- 算法分析

  1. 循环次数已知，但每次循环所做的操作不同；
  2. 设置逻辑尺，循环中依据逻辑尺中的标志位选择操作。


START
BL=0, CX=8
键盘输入 0/1
0/1?
N
Y
AL-30
BL 逻辑左移 1 位
BL+AL
CX-1
CX=0?
N
Y
END
键盘输入字母
回车?
Y
N
BL 循环左移 1 位
CF=0?
N
Y
前趋
后继

```
code    segment
        assume  cs:code
start:  mov   bx，0                    ；初始化
        mov   cx，8
        inlog：  mov   ah，1           ；键盘输入0/1
        int   21h
        cmp   al，30h
        jb    exit                     ；非法输入
        cmp   al，31h
        ja    exit                     ；非法输入
        sub   al，30h                  ；输入是0/1
```

```
        shl   bl，1
        add   bl,al
        loop  inlog
        mov   ah，2
        mov   dl，10          ；输出换行
        int   21h
inchr：  mov   ah，1          ；键盘输入字母
        int   21h
        cmp   al，13
        je    exit            ；回车键
        mov   dl,al
```

```
        rol   bl，1
        jnc   k30                ；是0 则转k30
        inc   dl
        jmp   putc
k30：   dec   dl
putc：  mov   ah，2
        int   21h
        jmp   inchr
exit：  mov   ah，4ch            ；程序终止并退出
        int   21h
code    ends
        end   start
```

7.2.4 条件计数循环程序

- 例7.6 设置键盘缓冲区为16个字节，从键盘输入一串字符，然后再从键盘输入一个单个字符，查找这个字符是否在字符串中出现，如果找到，显示该字符串，否则显示‘NOT FOUND’。

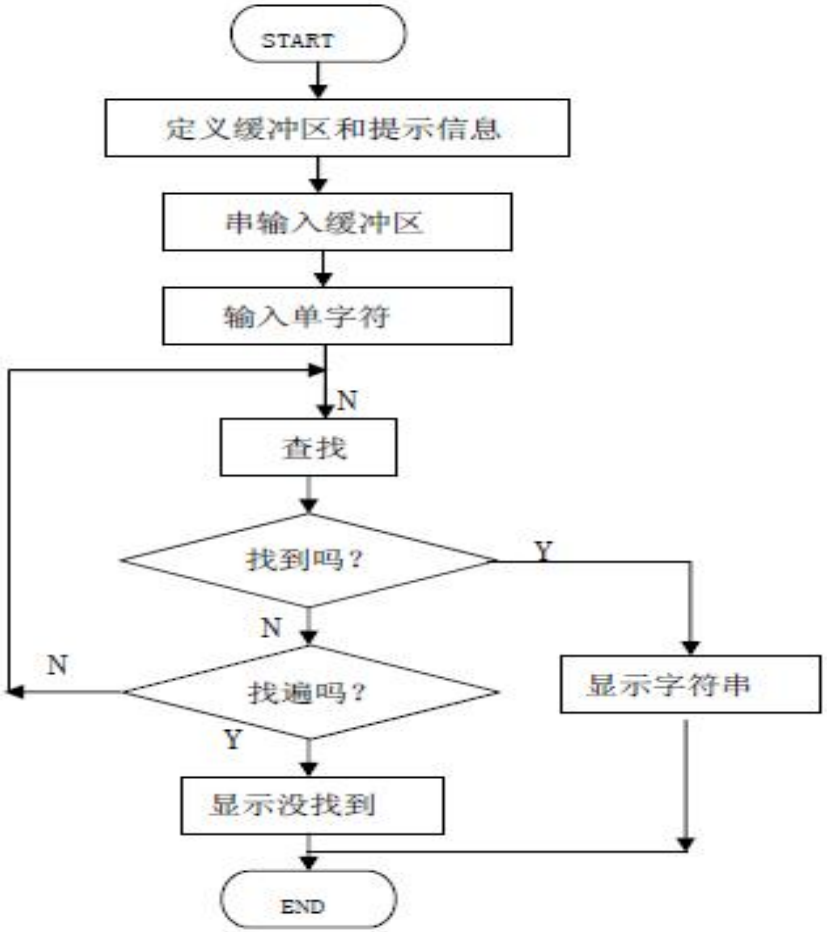
- 算法分析：

  1. 使用DOS系统功能调用10号功能实现键盘缓冲 区输入，

  2. 使用1号功能实现单个字符输入，

  3. 使用9号功能实现字符串显示输 出。

  4. 程序采用循环结构实现查找，最大计数值为实际输入的字符个数。


START
定义缓冲区和提示信息
串输入缓冲区
输入单字符
N
查找
找到吗？
Y
N
N
找遍吗？
显示字符串
Y
显示没找到
END

```
data    segment
        buffer   db   16,?,16 dup(?),13,10,'$'
        inputs   db   13，10，  'input string:$'
        getc     db   13，10，  'input char:$'
        output   db   13，10，  'not found$'
data    ends
code    segment
        assume  cs:code，ds:data
start:    mov   ax，data          ；ds赋值
          mov   ds，ax
          lea   dx，inputs        ；信息提示输入串
          mov   ah,9
          int   21h
          lea   dx，buffer        ；键盘输入串到缓冲区
```

```
        mov   ah,10
        int   21h
        lea   dx，getc          ；信息提示输入字符
        mov   ah,9
        int   21h
        mov   ah,1              ；输入字符到al
        int   21h
        mov   bl，al            ；保存到bl
        lea   di，buffer+1      ；di作为指针指向缓冲区
        mov   cl，buffer+1      ；cx设置计数值
        mov   ch，0
seek：inc   di
      cmp   bl，[di]
        loopne   seek           ；未完且没找到，转seek继续循环
```

```
    jne   nof                  ；没找到，转nof输出‘not found’
    mov   dl,10                ；输出换行
    mov   ah,2
    int   21h
    lea   dx，buffer+2         ；指向缓冲区，输出字符串
    mov   ah,9
    int   21h
    jmp   exit
nof：lea   dx，output
    mov   ah,9
    int   21h
exit：mov   ah，4ch
    int   21h
code   ends
       end   start
```

7.2.5 多重循环程序

- 例7.7 显示输出20H～7EH的ASCII码字符表，每行16个字符。

- 算法分析：

  1. 20H～7EH的ASCII字符共有95个，需6行显示。

  2. 程序需两重循环，内循环输出每行16个字符，循环计数初值为16，程序终止条件是显示最后一个字符。

高级语言程序描述

```
first=20h
last=7eh
char= first
x=1              ；行号
y=1              ；列号
do  while  char<last
      k=16
      do   while   k>0  and  char<last
        @ x, y  say  char
        char=char+1
         y=y+1
         k=k-1
      enddo
      x=x+1
      y=1
enddo
```

```
code    segment
    assume  cs:code
      k=16
      first=20h
      last=7eh
start:  mov   dx,first          ；从第一个开始
a10:    mov   cx，k             ；每行16个
a20:    mov   ah，2
        int   21h
```

```
cmp   dl，last          ；是最后一个则退出
je    exit
push  dx                ；暂存dx
mov   dl，20h           ；空2格
int   21h
int   21h
pop   dx                ；恢复dx
add   dx，1
loop  a20               ；进入内循环
push  dx                ；暂存dx
```

```
        mov   dl，13            ；回车
        int   21h
        mov   dl，10            ；换行
        int   21h
        pop   dx                ；恢复dx
        loop  a10               ；进入外循环
exit：  mov   ah，4ch
        int   21h
code    ends
        end   start
```