第5章 循环与分支程序设计

例5.2:p163

在ADDR单元中存放着数Y的地址，试编制一程序把Y中1的个数存入COUNT单元中

- 变量定义：

```
addr      dw    number
number    dw    Y
```

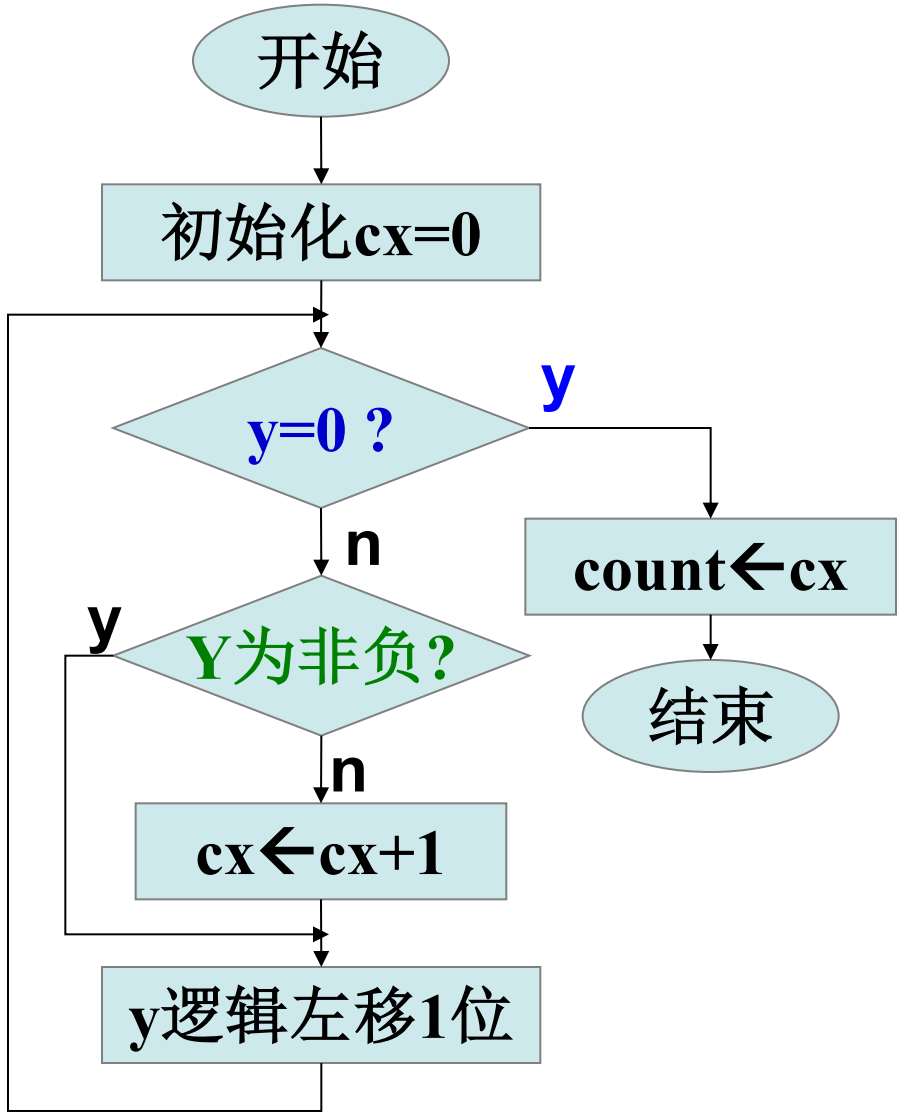
- 程序处理：

（1）采用逐位测试：用移位方法把各位数逐次移到最高位去，根据最高有效位是否为1（负数）来记数

（2）循环的结束条件：

①计数值为16

②Y=0



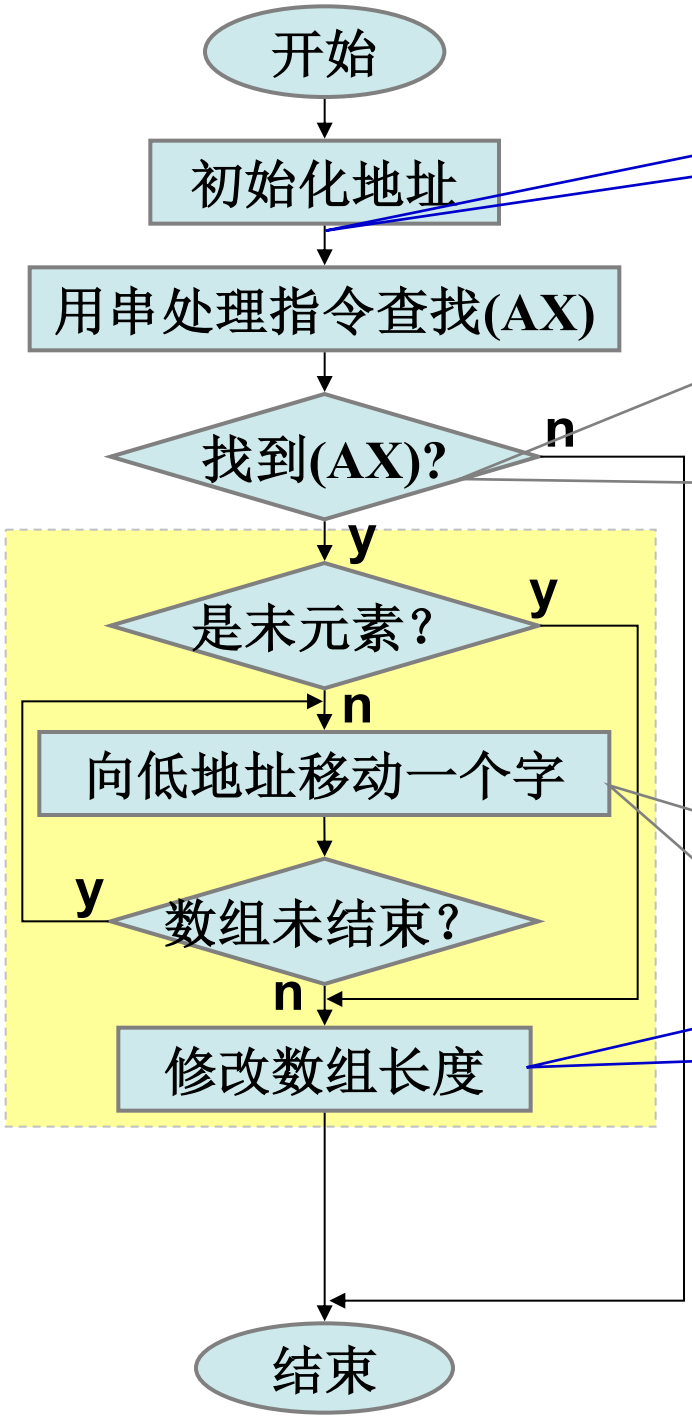

```
addr  dw    number
number      dw    78a4h
count dw    ?
… …
              mov   cx,0
              mov   bx,addr
              mov   ax,[bx]
repeat:       test  ax,0ffffh
              jz    exit
              jns   shift
              inc   cx
shift:        shl   ax,1
              jmp   repeat
exit:         mov   cx,count
```

例5.3：p164

在附加段中，有一个首地址为LIST且未经排序的字数组。在数组的第一个字中，存放着该数组的长度，数组的首地址存放在DI寄存器中，AX寄存器中存放着一个数。要求编制一程序：在数组中查找该数，如果找到此数，则把它从数组中删除

```
data    segment
        number      dw      23
data    ends

extra   segment
        list    dw      10
                dw      15,37,5,3,23
                dw      99,65,52,78,49
extra   ends
```



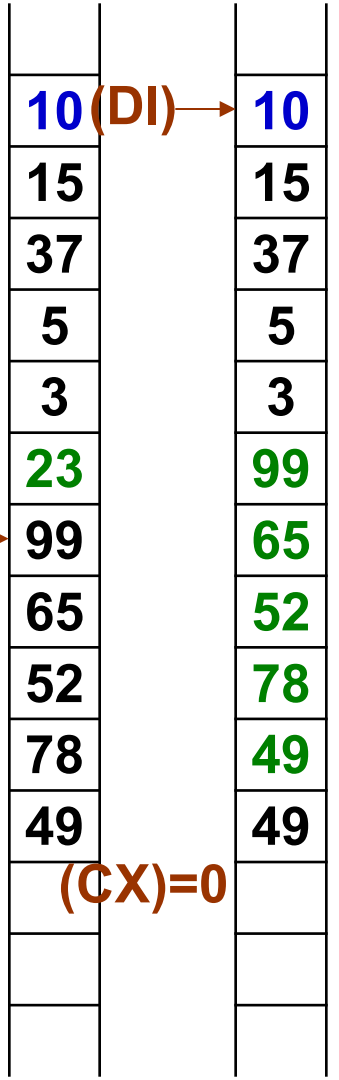
保留数组首地址
PUSH DI或MOV SI,DI

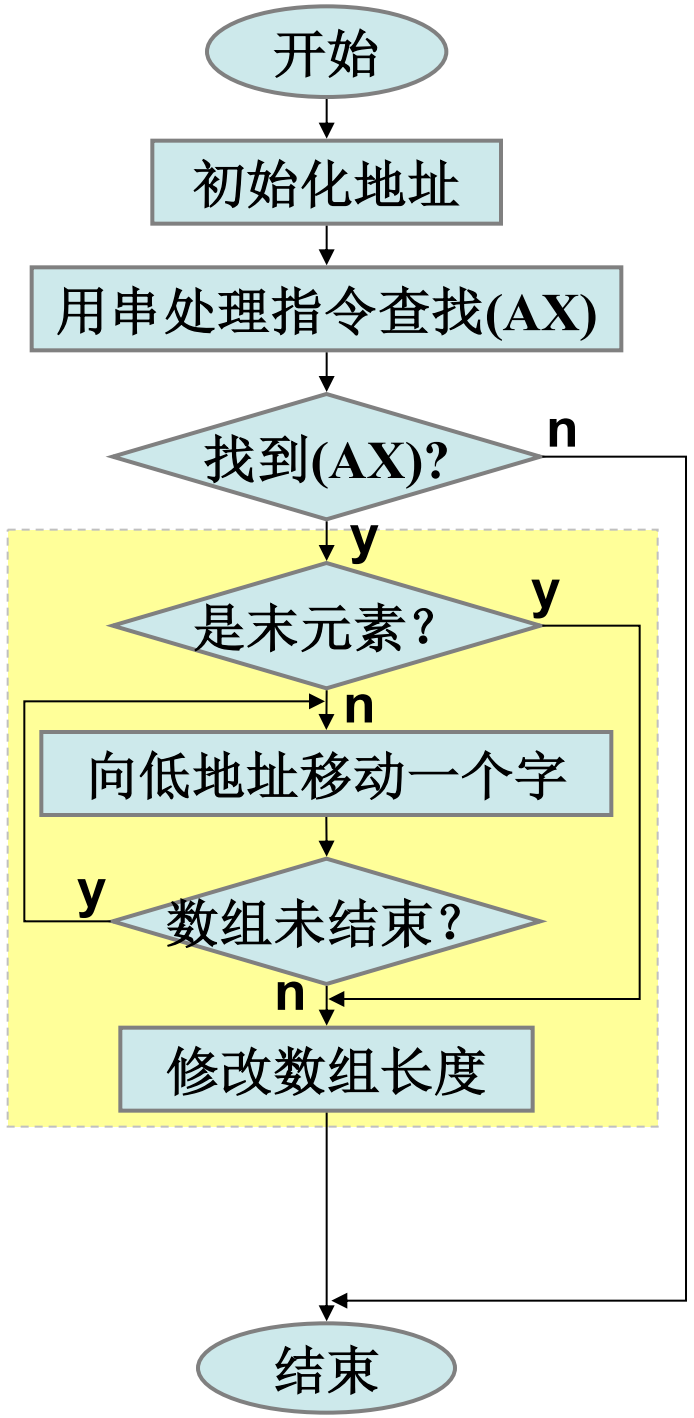
查找结果：
找到：ZF=1
CX剩余元素
(DI)指向下一个字
未找到:ZF=0
(CX)=0

找到中间元素：
[DI]及其前元素后移1个字
循环次数：CX

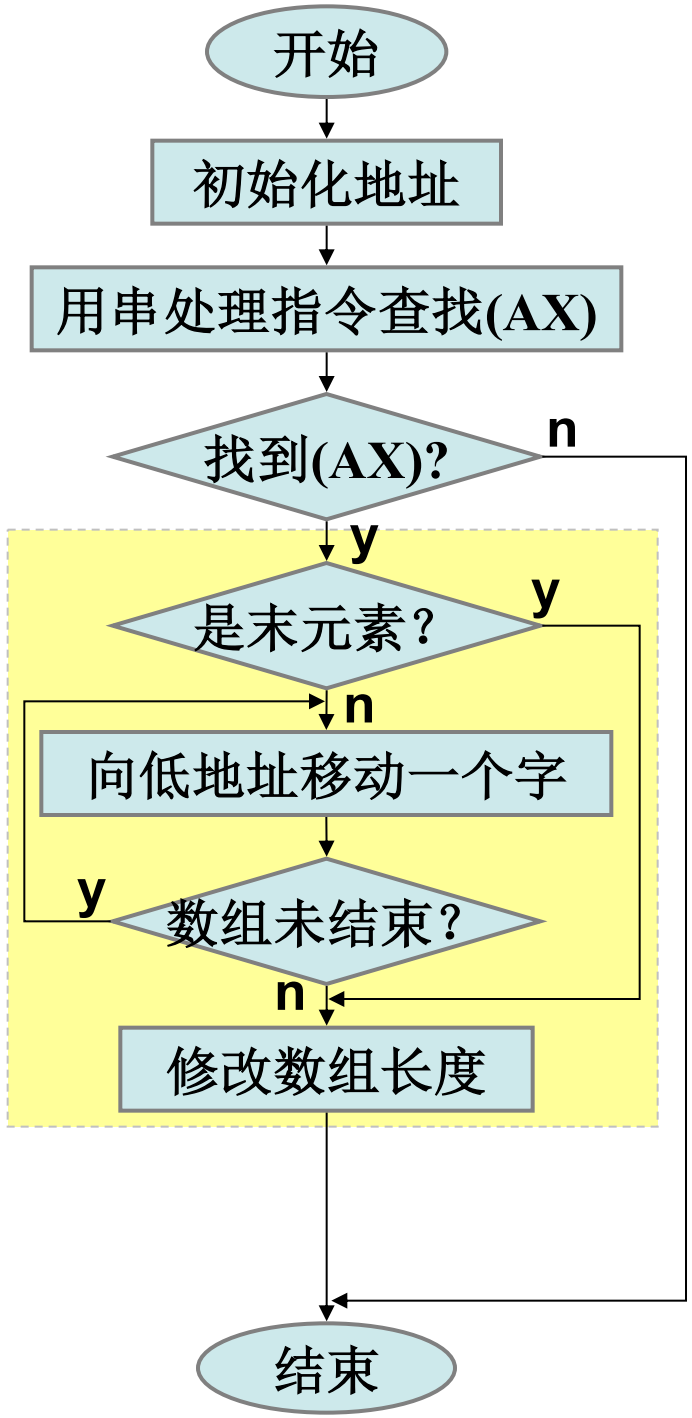
恢复数组首地址
POP DI或MOV DI,SI

约定：
前移：向高地址移动
后移：向低地址移动






```
data    segment
        number      dw      23
data    ends
extra   segment
        list    dw      10
                dw      15,37,5,3,23
                dw      99,65,52,78,49
extra   ends
code    segment
  assume  cs:code,ds:data,es:extra
start:
        mov     ax,data
        mov     ds,ax
        mov     ax,extra
        mov     es,ax
        lea     di,list
        mov     ax,number
        … …     ;查找删除
code    ends
        end     start
```




```
            cld
            push  di
            mov   cx,es:[di]
            add   di,2
            repne scasw
            je    delete
            pop   di
            jmp   exit
delete:     jcxz  dec_cnt
next_el:    mov   bx,es:[di]
            mov   es:[di-2],bx
            add   di,2
            loop  next_el
dec_cnt:    pop   di
            dec   word ptr es:[di]
exit:       mov   ax,4c00h
            int   21h
```

例5.4：p166

- 将正数N插入一个已整序的字数组的正确位置。该数组的首地址和末地址分别为**array_head**和**array_end**，其中所有数均为正数且已按递增的次序排列

```
data    segment
        array_head  dw      3,5,15,23,37,49,52,65,78,99
        array_end   dw      105
        n       dw      32
data    ends
```

约定：
前移：向高地址移动
后移：向低地址移动

例5.4：p166

- 程序分析—三步：

  1) 寻找插入位置：

  a.从首部开始→ 尾部

  b.从尾部开始→ 首部

  2) 插入位置前的字前移:从尾部开始→首部

  3) 插入

- 程序处理：

  1) 从数组的尾部逐字取出数组中的一个数K与N比较，如K>N，则把K前移一个字，继续比较；如K≤N则把N插在数组中K之前

  2) 循环结束条件： K≤N

例5.4：p166

- 程序分析—三步：

  1) 寻找插入位置：

  a.从首部开始→ 尾部

  b.从尾部开始→ 首部

  2) 插入位置前的字前移:从尾部开始→首部

  3) 插入

- 程序处理：

  1) 从数组的尾部逐字取出数组中的一个数K与N比较，如K>N，则把K前移一个字，继续比较；如K≤N则把N插在数组中K之前

  2) 循环结束条件： K≤N

  3) 边界：

例5.4：p166

- 程序分析—三步：

  1) 寻找插入位置：

  a.从首部开始→ 尾部

  b.从尾部开始→ 首部

  2) 插入位置前的字前移:从尾部开始→首部

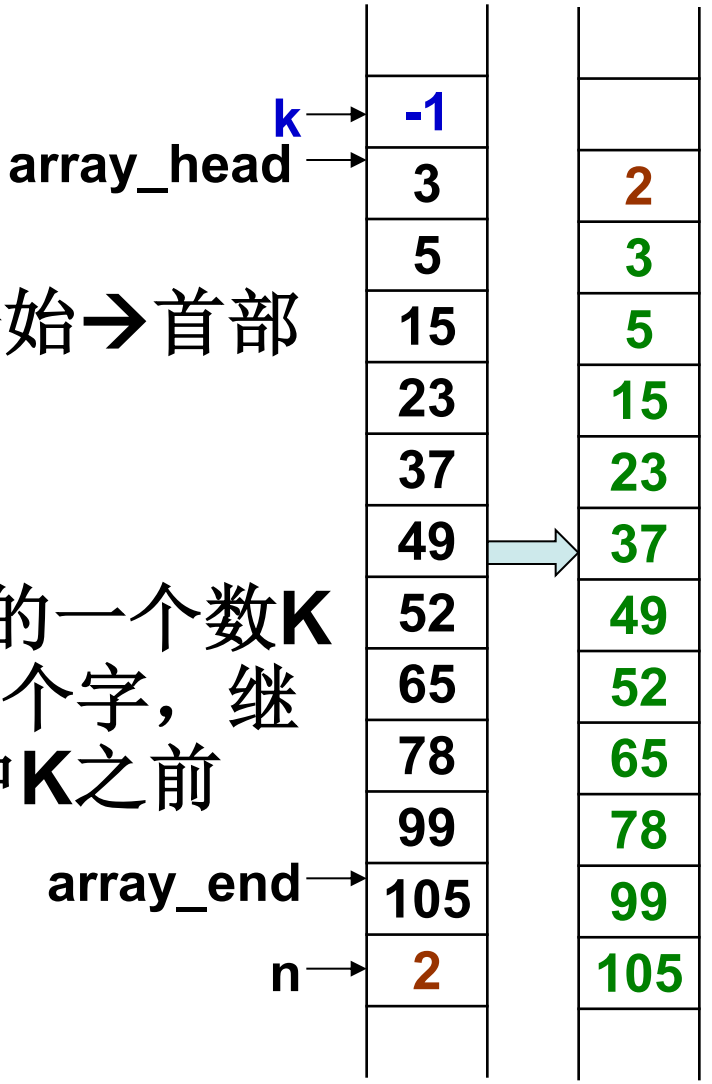
  3) 插入

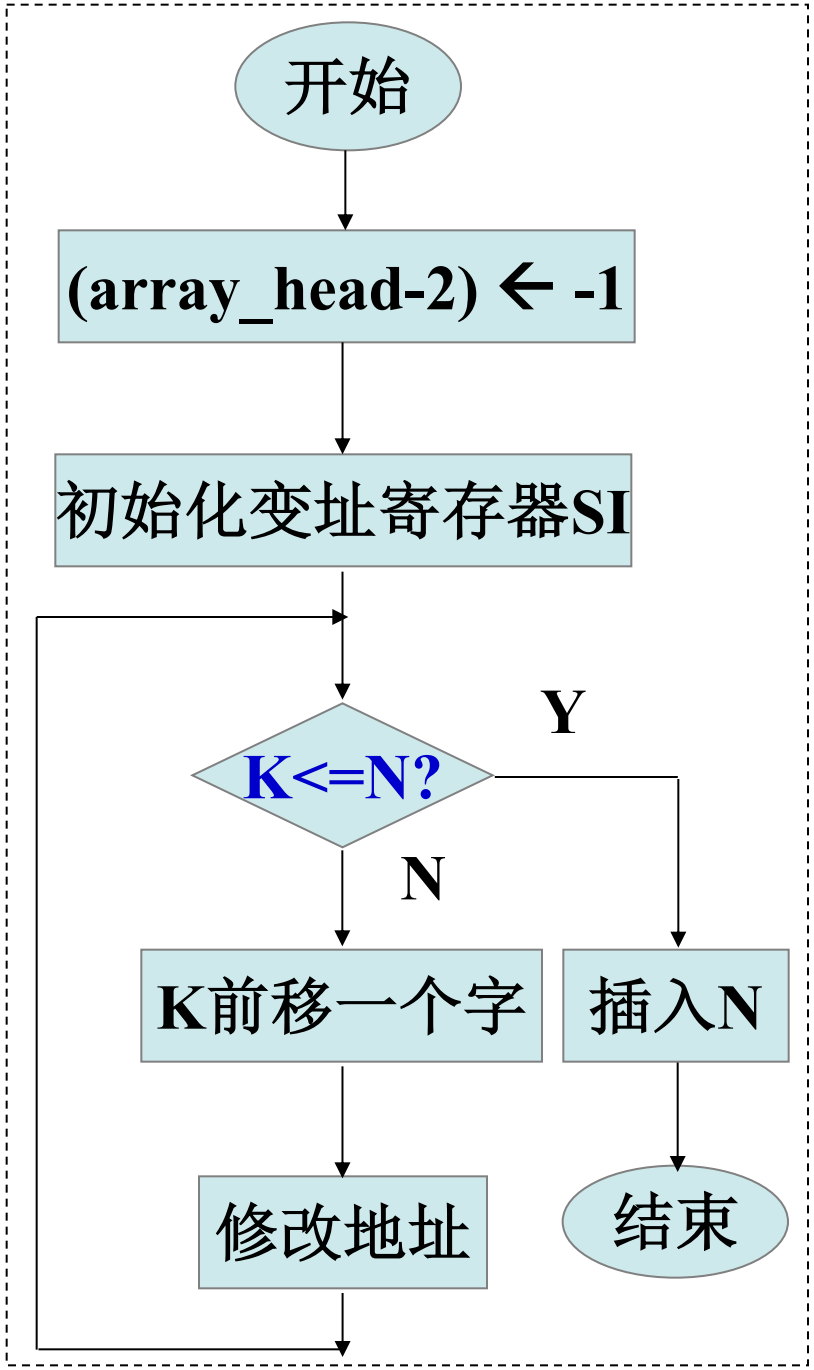
- 程序处理：

  1) 从数组的尾部逐字取出数组中的一个数K与N比较，如K>N，则把K前移一个字，继续比较；如K≤N则把N插在数组中K之前

  2) 循环结束条件： K≤N

  3) 边界：array_head-2 ← -1






```
x           dw      ?           ;-1
array_head  dw      3,5,15,23,37
array_end   dw      105
n           dw      32
            … …
            mov     ax,n
            mov     x,-1        ;0ffffh
            mov     si,0
compare:
            cmp     array_end[si],ax
            jle     insert
            mov     bx,array_end[si]
            mov     array_end[si+2],bx
            sub     si,2
            jmp     compare
insert:
            mov     array_end[si+2],ax
```

例5.7 冒泡法排序：p172

有一个首地址为A的N字数组，编制程序使该数组中的数按照从大到小的次序整序

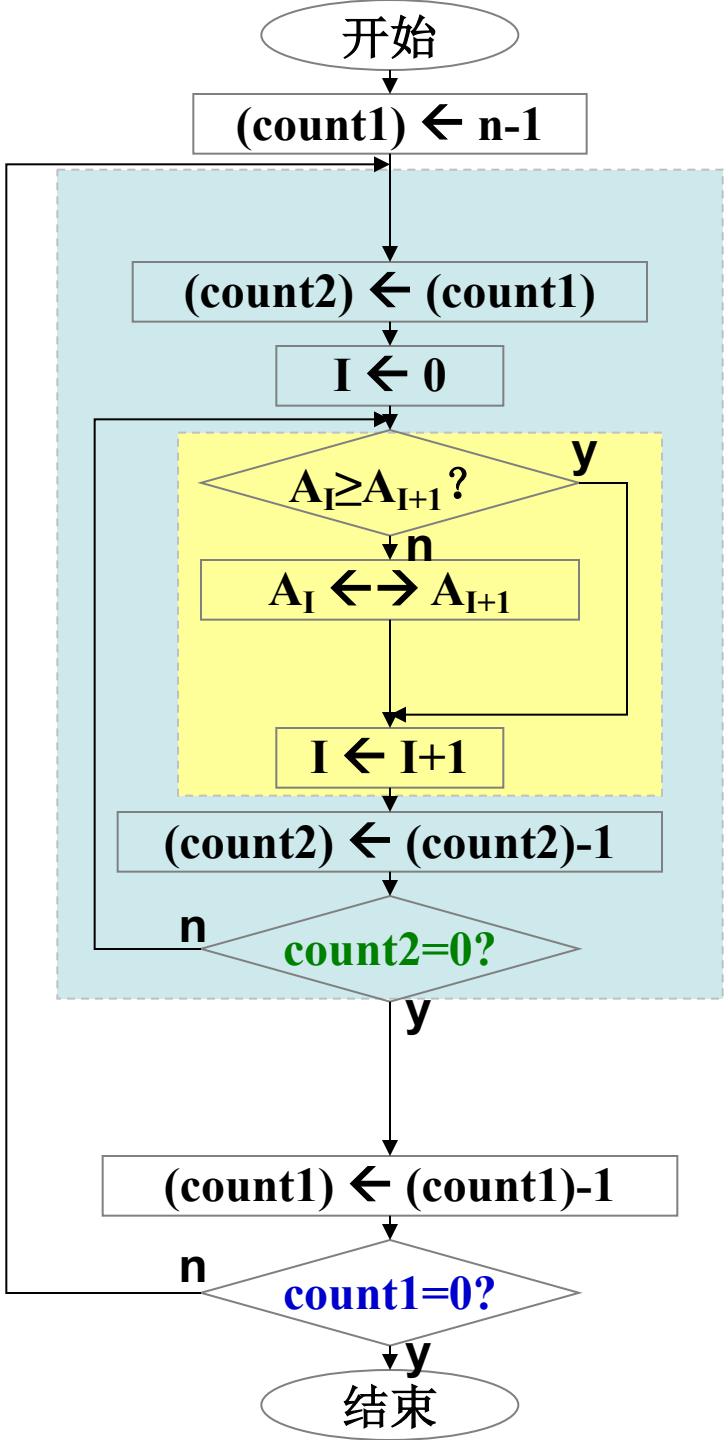
第几趟：	1	2	3	4
所需趟数：	4	3	2	1
俩俩比较次数：	4	3	2	1

5	8	8	8	8	8	16	16	16	16	32	32	32	84	
8	5	16	16	16	16	8	32	32	32	16	84	84	32	
16	16	5	32	32	32	32	8	84	84	84	16	16	16	
32	32	32	5	84	84	84	84	8	8	8	8	8	8	
84	84	84	84	5	5	5	5	5	5	5	5	5	5	

所需趟数(n-1)构成外层循环

每趟中俩俩比较次数构成内层循环

内层循环次数与外层循环次数相同




```
                mov   cx,n
                dec   cx
loopout:        mov   di,cx

                mov   bx,0
loopin:         mov   ax,a[bx]
                cmp   ax,a[bx+2]
                jge   continue
                xchg  ax,a[bx+2]
                mov   a[bx],ax

continue:       add   bx,2
                loop  loopin

                mov   cx,di
                loop  loopout
```

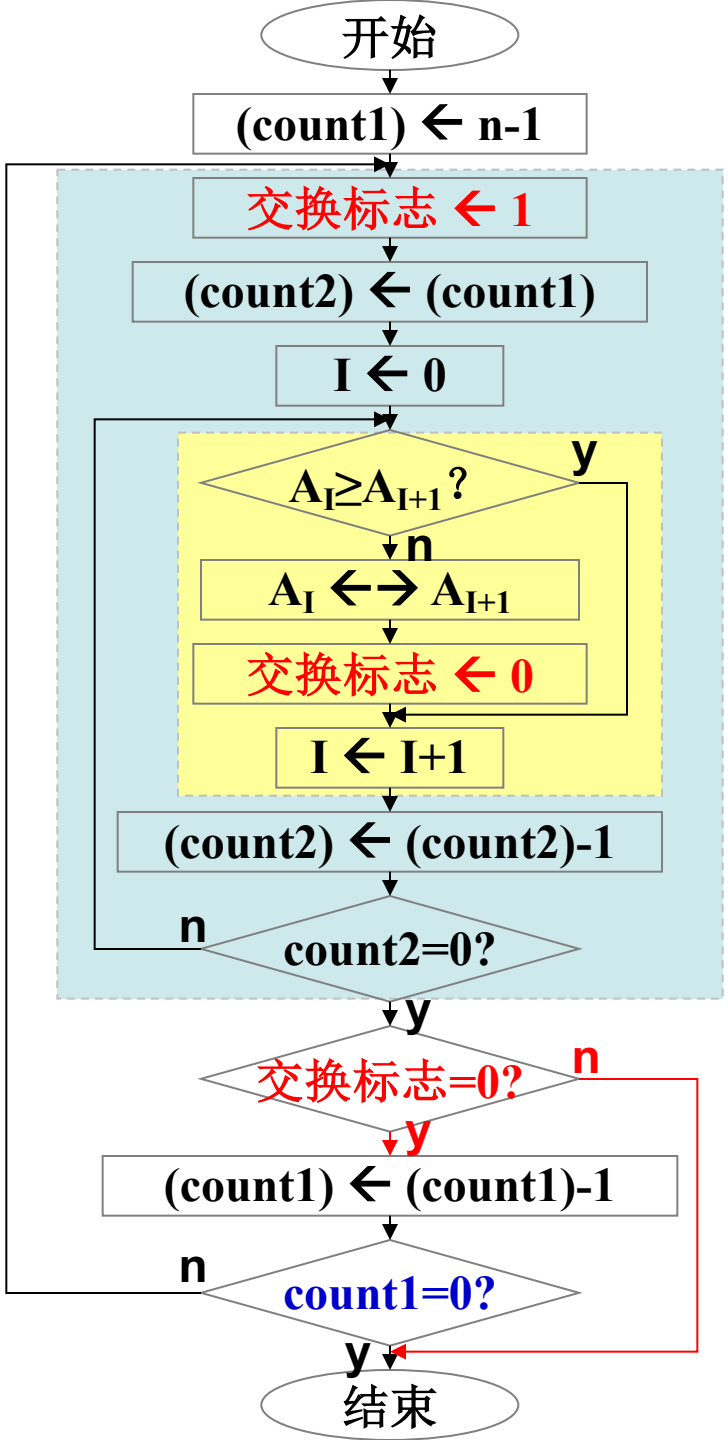
第几趟		1	2	3	4
所需趟数		4	3	2	1
16		84	84	84	84
84		16	32	32	32
5		32	16	16	16
32		8	8	8	8
8		5	5	5	5
交换	初始值	1	1	1	
标志	未值	0	0	1	

第几趟		1	2	3	4
所需趟数		4	3	2	1
84		84	84	84	84
32		16	32	32	32
16		32	16	16	16
8		8	8	8	8
5		5	5	5	5
交换	初始值	1			
标志	未值	1			

- 提前结束外循环：
  - 设立交换标志位，初始值为1，交换后置0
  - 开始下一次外循环前检查交换标志位的值，若仍为1，说明已有序，即可提前结束循环

例5.8:p174

在附加段中有一个字数组，其首地址已存放在DI寄存器中，在数组的第一个字中存放着该数组的长度。要求编制一个程序使该数组中的数按照从小到大的次序排列整齐




```
            mov   cx,n
            dec   cx
loopout:    mov   di,cx
            mov   si,1
            mov   bx,0
loopin:     mov   ax,a[bx]
            cmp   ax,a[bx+2]
            jge   continue
            xchg  ax,a[bx+2]
            mov   a[bx],ax
            sub   si,si
continue:   add   bx,2
            loop  loopin
            cmp   si,0
            jne   finish
            mov   cx,di
            loop  loopout
finish:     ret
```

第5章 循环与分支程序设计

- 5.1 循环程序设计
- 5.2 分支程序设计

例5.5：p167

设有数组X和Y。X数组中有X_1，…，X_{10}。Y数组中有Y_1，…，Y_{10}。试编制程序计算

	标志位
$Z_1=X_1+Y_1$	0
$Z_2=X_2+Y_2$	0
$Z_3=X_3-Y_3$	1
$Z_4=X_4-Y_4$	1
$Z_5=X_5-Y_5$	1
$Z_6=X_6+Y_6$	0
$Z_7=X_7-Y_7$	1
$Z_8=X_8-Y_8$	1
$Z_9=X_9+Y_9$	0
$Z_{10}=X_{10}+Y_{10}$	0

逻辑尺

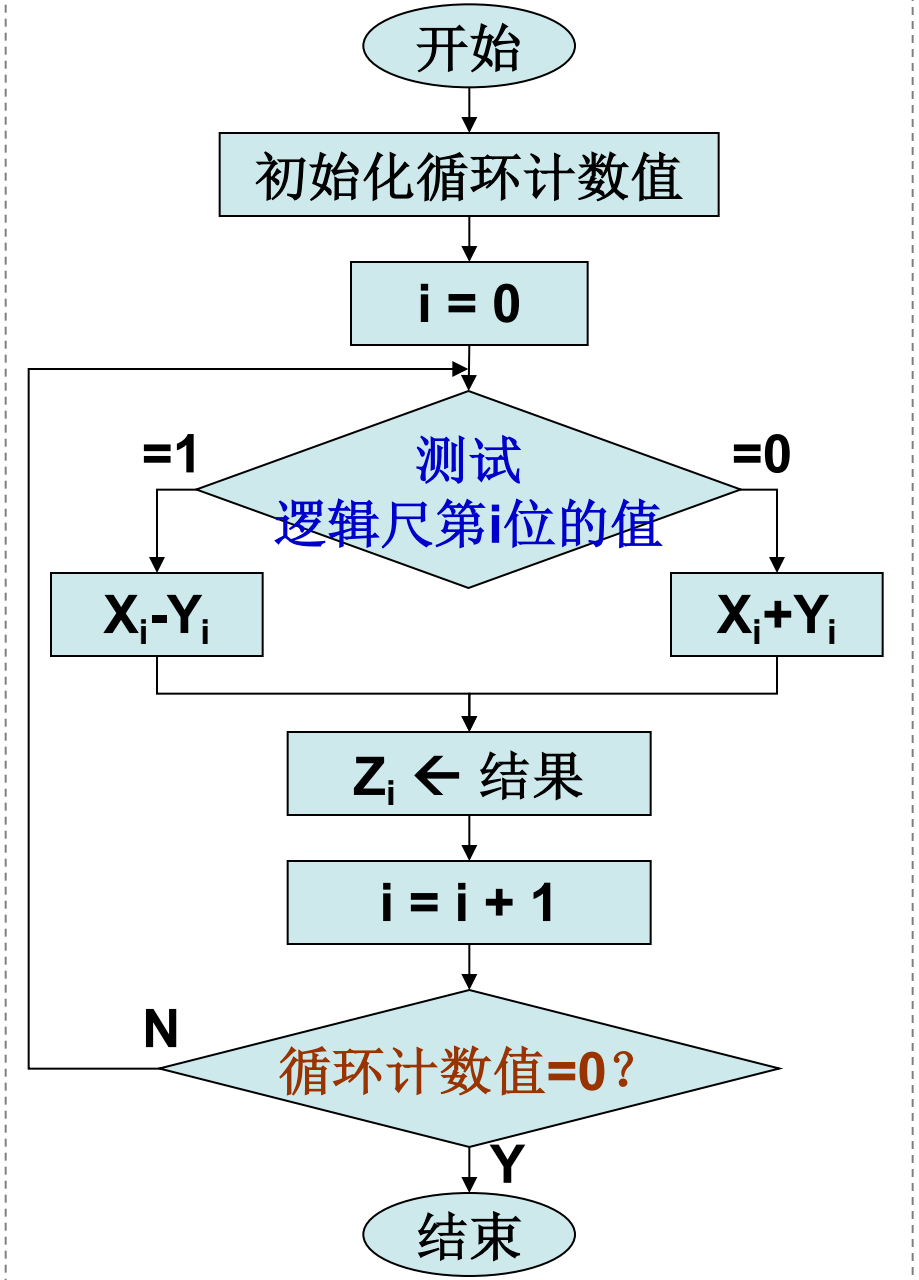
0000 0000 1101 1100

00dch

shr

cf=1 或cf=0

```
data    segment
        x       dw      10,20,30,40,50,60,70,80,90,100
        y       dw      11,22,33,44,55,66,77,88,99,111
        z       dw      10 dup(?)
        logic_rule      dw      00dch
data    ends
;*************************************************************
program         segment
;-------------------------------------------------------------
main    proc    far
        assume          cs:program,ds:data
start:
        push    ds
        sub     ax,ax
        push    ax
        mov     ax,data
        mov     ds,ax
        … …             ;计算
        ret
main    endp
;-------------------------------------------------------------
program         ends
;*************************************************************
        end     start
```




```
        mov   bx,0
        mov   cx,10
        mov   dx,logic_rule
next:
        mov   ax,x[bx]
        shr   dx,1
        jc    subtract
        add   ax,y[bx]
        jmp   result
subtract:
        sub   ax,y[bx]
result:
        mov   z[bx],ax
        add   bx,2
        loop  next
```

例5.9 折半查找：p177

在附加段中，有一个按从小到大顺序排列的无符号数数组，其首地址存放在DI寄存器中，数组中的第一个单元存放着数组长度。在AX中有一个无符号数，要求在数组中查找(AX)，如找到，则使CF= 0, 并在SI中给出该元素在数组中的偏移地址；如未找到，则使CF=1。

折半查找法

mid=(low+high)/2
中间元素地址=array+mid*2

(ax)=55

下标	数据
0	12
1	11
2	22
3	33
4	44
5	55
6	66
7	77
8	88
9	99
10	111
11	222
12	333

low	high	mid
1	12	6
1	5	3
4	5	4
5	5	5

❷ (ax)>array[mid] low=mid+1
❸ (ax)>array[mid] low=mid+1
❹ (ax)=array[mid]
❶ (ax)<array[mid] high=mid-1

(si)=0ah
cf=0

折半查找法

mid=(low+high)/2
中间元素地址=array+mid*2

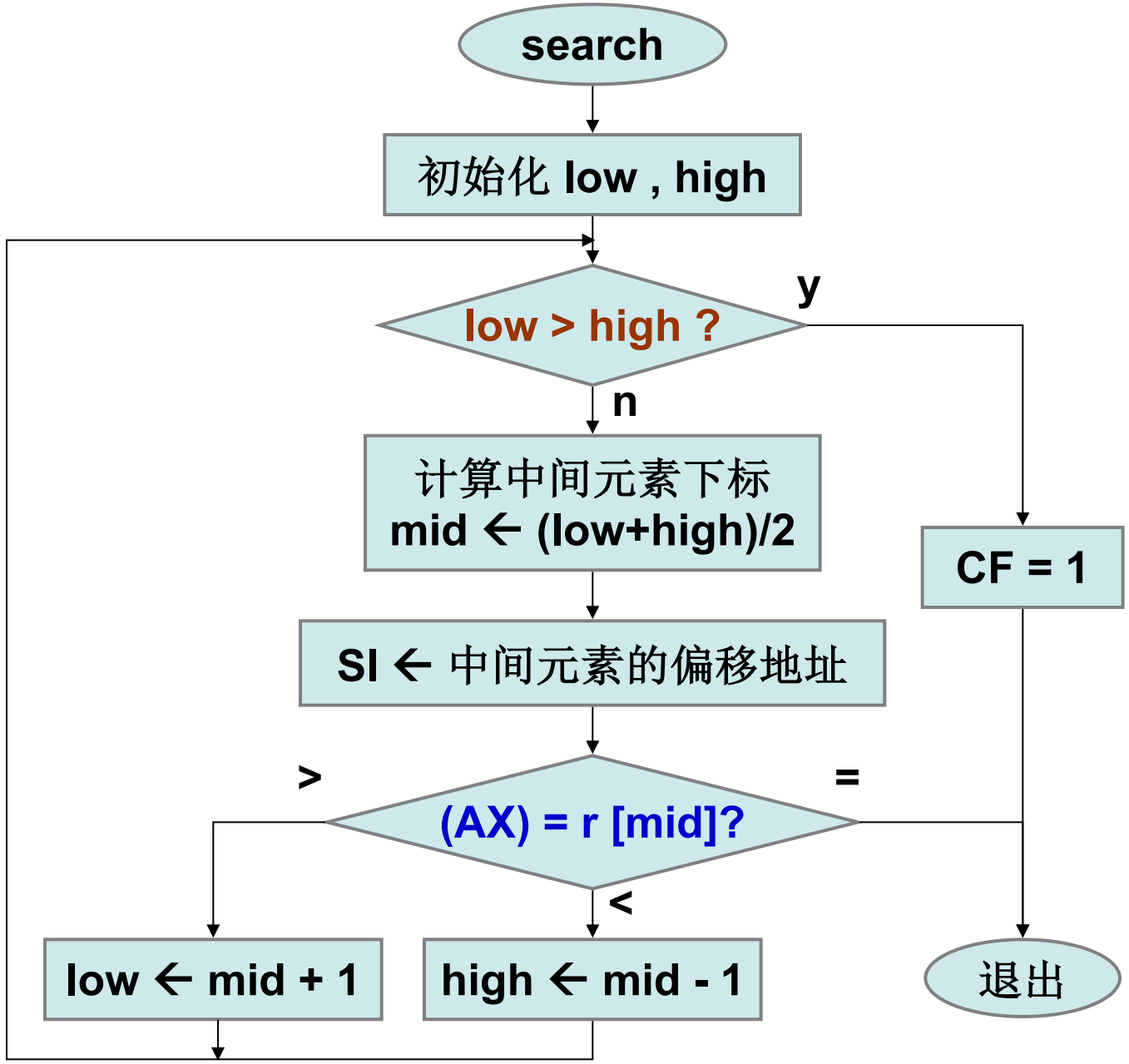
(ax)=90

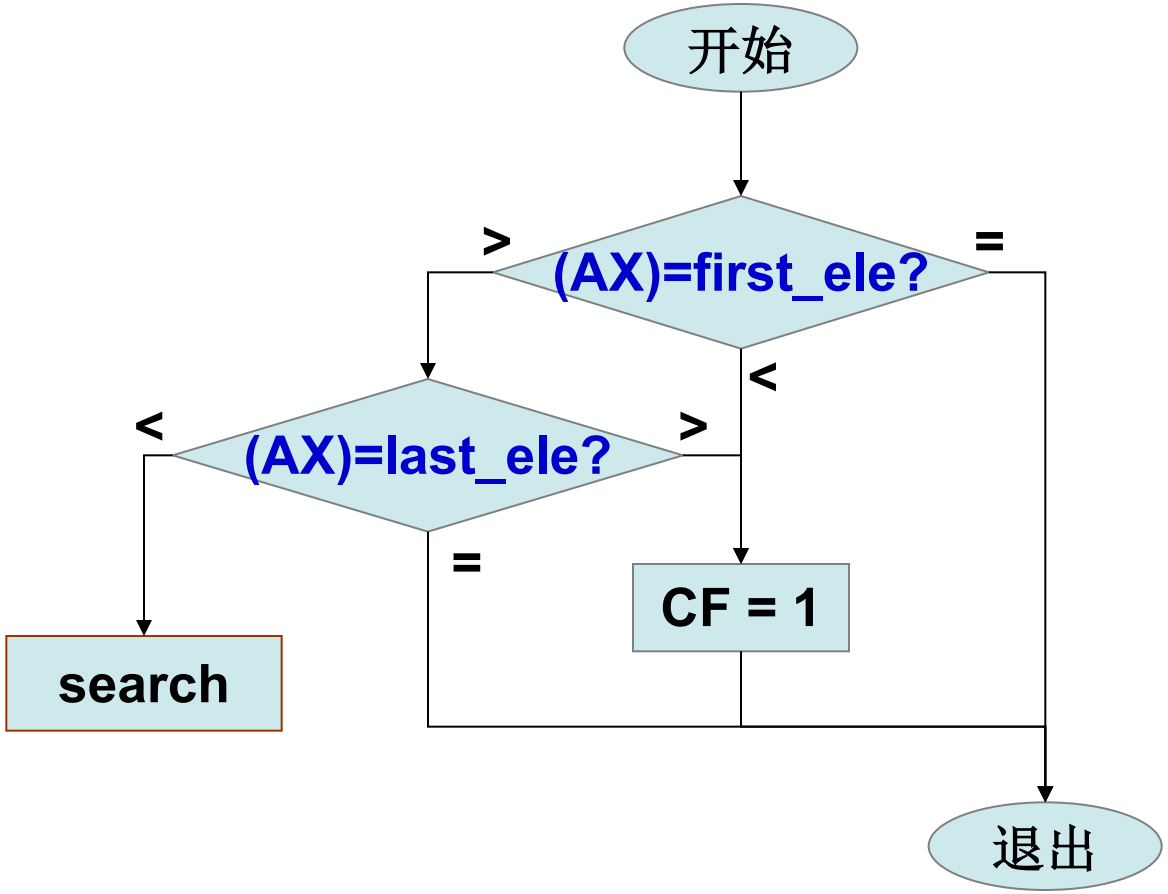
下标	数据
0	12
1	11
2	22
3	33
4	44
5	55
6	66
7	77
8	88
9	99
10	111
11	222
12	333

low	high	mid
1	12	6
7	12	9
7	8	7
8	8	8
9	8	结束

❶ (ax)>array[mid] low=mid+1
❸ (ax)>array[mid] low=mid+1
❹ (ax)>array[mid] low=mid+1
❷ (ax)<array[mid] high=mid-1

(si)=10h
cf=1


search
初始化 low , high
low > high ?
y
n
计算中间元素下标
mid ← (low+high)/2
CF = 1
SI ← 中间元素的偏移地址
>
=
(AX) = r [mid]?
<
low ← mid + 1
high ← mid - 1
退出


开始
(AX)=first_ele?
>
=
<
(AX)=last_ele?
<
>
=
search
CF = 1
退出

```
data    segment
        number      dw      55
        low_index   dw      ?
        high_index  dw      ?
data    ends
extra   segment
        array   dw      12
                dw      11,22,33,44,55,66
                dw      77,88,99,111,222,333
extra   ends
code    segment
        assume cs:code,ds:data,es:extra
start:
        mov     ax,data
        mov     ds,ax
        mov     ax,extra
        mov     es,ax
        … …             ;折半查找
exit:   mov     ax,4c00h
        int     21h
code    ends
        end     start
```

```
            lea     di,array        ;(di)←数组首地址
            mov     ax,number       ;(ax)←待查找数据
            cmp     ax,es:[di+2]    ;(ax)与第一个元素比较
            ja      chk_last        ;分支1
            lea     si,es:[di+2]    ;(si)←第一个元素地址,分支2
            je      exit
            stc                     ;置cf=1,分支3
            jmp     exit
chk_last:
            mov     si,es:[di]      ;(si)←元素个数
            shl     si,1
            add     si,di           ;计算最后一个元素的地址
            cmp     ax,es:[si]      ;(ax)与最后一个元素比较
            jb      search          ;分支1
            je      exit            ;分支2
            stc                     ;分支3
            jmp     exit
```

```
search:
        mov   low_index,1      ;low_index←1
        mov   bx,es:[di]
        mov   high_index,bx    ;high_index←数组长度
        mov   bx,di            ;(bx)←数组首地址
mid:
        mov   cx,low_index
        mov   dx,high_index
        cmp   cx,dx
        ja    no_match         ;low_index>high_index
        add   cx,dx
        shr   cx,1             ;(cx)←mid_index
        mov   si,cx            ;(si)←mid_index
        shl   si,1             ;(si)←中间元素在数组中地址
```

```
compare:
        cmp   ax,es:[bx+si]   ;(ax)与中间元素比较
        je    exit            ;分支1
        ja    higher          ;分支2
lower:                        ;分支3
        dec   cx
        mov   high_index,cx   ;high_index←mid_index-1
        jmp   mid
higher:
        inc   cx
        mov   low_index,cx    ;low_index←mid_index+1
        jmp   mid
no_match:
        stc                   ;置cf=1
```