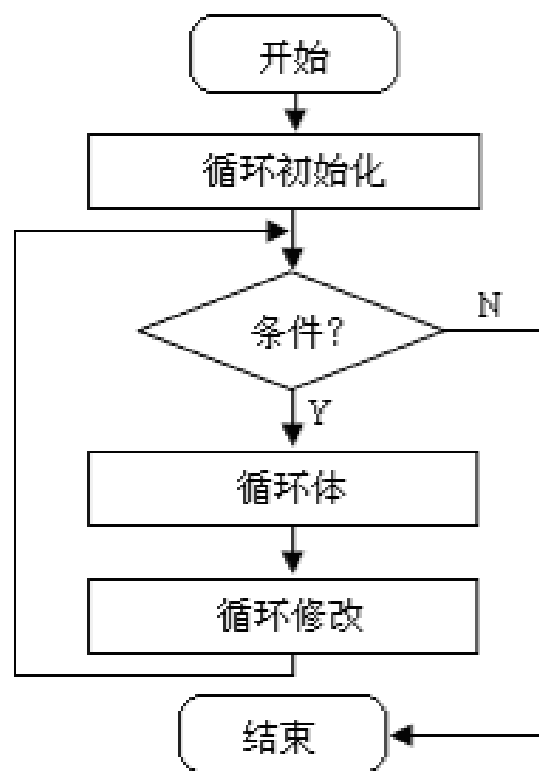
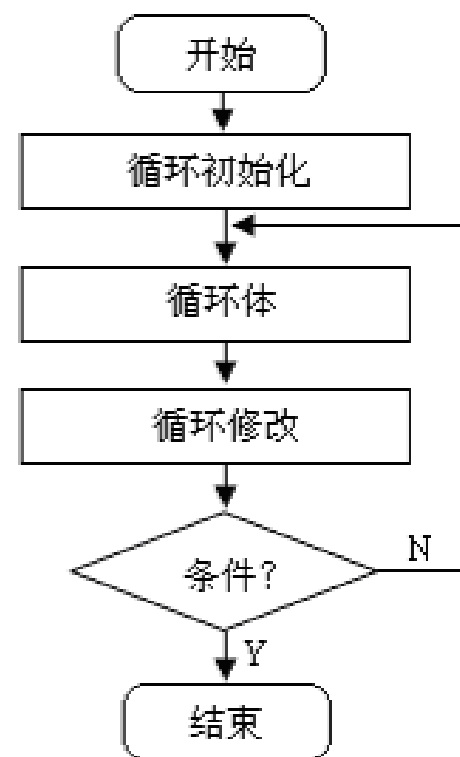


- 控制转移指令
- 循环结构程序设计
- 分支结构程序设计

循环结构程序设计



DO-WHILE 结构



DO-UNTIL 结构

基本循环结构示意图

循环结构程序设计

- 循环初始化部分

这是循环准备工作阶段，如建立地址指针、设置循环次数、必要的
数据保护以及为循环体正常工作而建立的初始状态等。

- 循环体

循环体是在循环过程中反复执行的部分。它是循环的核心部分，是
循环程序所要完成的若干操作的全部指令。

- 循环修改部分

循环修改主要是指对一些运算控制单元（变量、寄存器）的修改，
如修改操作数地址、修改循环计数器、改变变量的值等。

- 循环控制部分

根据给定的循环次数或循环条件，判断是否结束循环。若未结束，
则转去重复执行循环工作部分。

循环结构程序设计

重点：循环控制部分。

- ◆循环控制是循环体的一部分，它是循环程序设计的关键。
- ◆每个循环程序必须选择一个循环控制条件来控制循环的运行和结束。
 - ◆有时循环次数已知，此时可以用循环次数作为控制条件，`loop`指令使得这种循环程序设计很容易实现。
 - ◆有时循环次数已知，但有可能使用其他特征或条件使循环提前结束，此时可用`loopz`或`loopnz`指令。
 - ◆有时循环次数未知，那就需要根据具体情况找出控制循环结束的条件，采用转移指令来实现循环。

循环结构程序设计

【例5.1】 试编制一个程序把 **BX** 寄存器内的二进制数用十六进制数的形式在屏幕上显示出来。

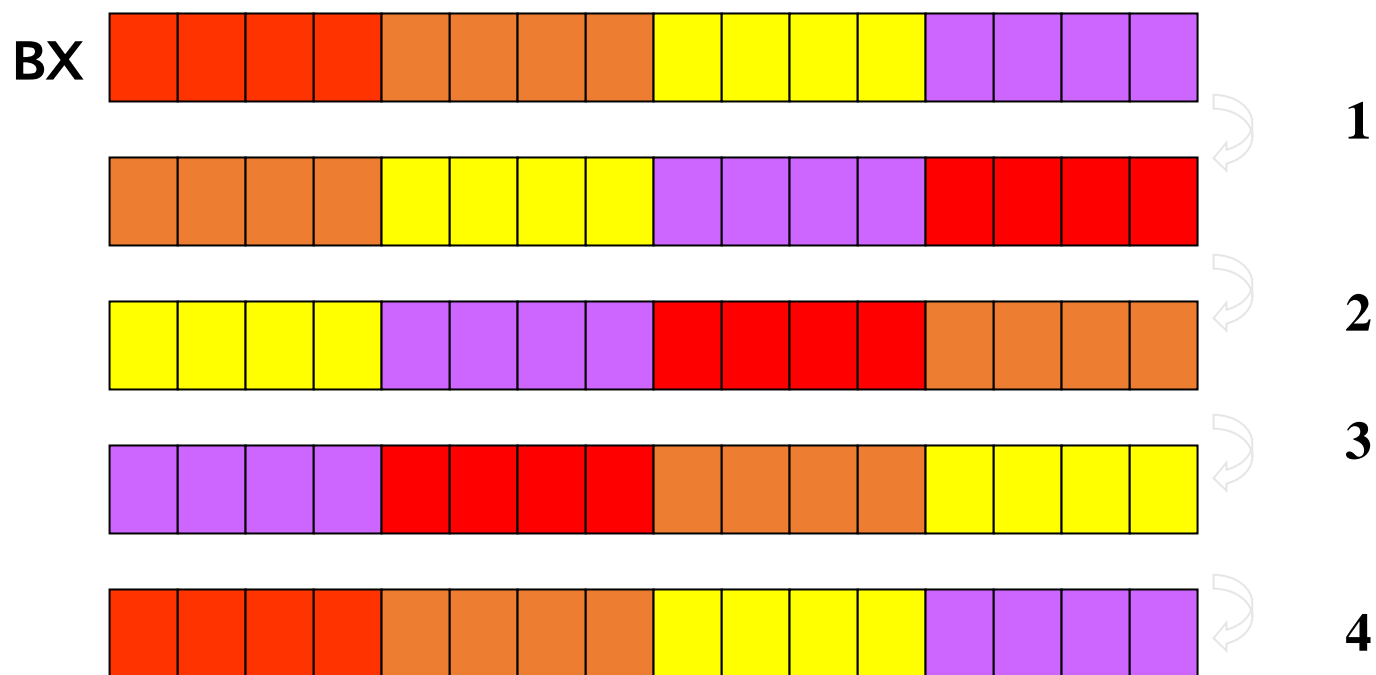
如：1011 0010 1111 1010 B → 0B2FAH

循环结构程序设计

分析： (1)程序结构的确定

由题意应该把BX的内容从左到右每4位为一组在屏幕上显示出来，显然这可以用循环结构来完成，每次显示一个十六进制数位，因而循环次数是已知的，计数值为4。

如： 1011 0010 1111 1010 B → 0B2FAH



循环结构程序设计

分析： (1)程序结构的确定

由题意应该把BX的内容从左到右每4位为一组在屏幕上显示出来，显然这可以用循环结构来完成，每次显示一个十六进制数位，因而循环次数是已知的，计数值为4。

(2)循环体的构成（算法确定）

循环体应该包括：二进制到所显示字符的ASCII之间的转换，以及每个字符的显示。

(3)需要了解相关知识

◆ 字符和其ASCII码之间的关系？

“0”~“9” → 30H~39H, “A”~“F” → 41H~46H

循环结构程序设计

分析： (1)程序结构的确定

由题意应该把BX的内容从左到右每4位为一组在屏幕上显示出来，显然这可以用循环结构来完成，每次显示一个十六进制数位，因而循环次数是已知的，计数值为4。

(2)循环体的构成（算法确定）

循环体应该包括：二进制到所显示字符的ASCII之间的转换，以及每个字符的显示。

(3)需要了解相关知识

◆ 字符和其ASCII码之间的关系？

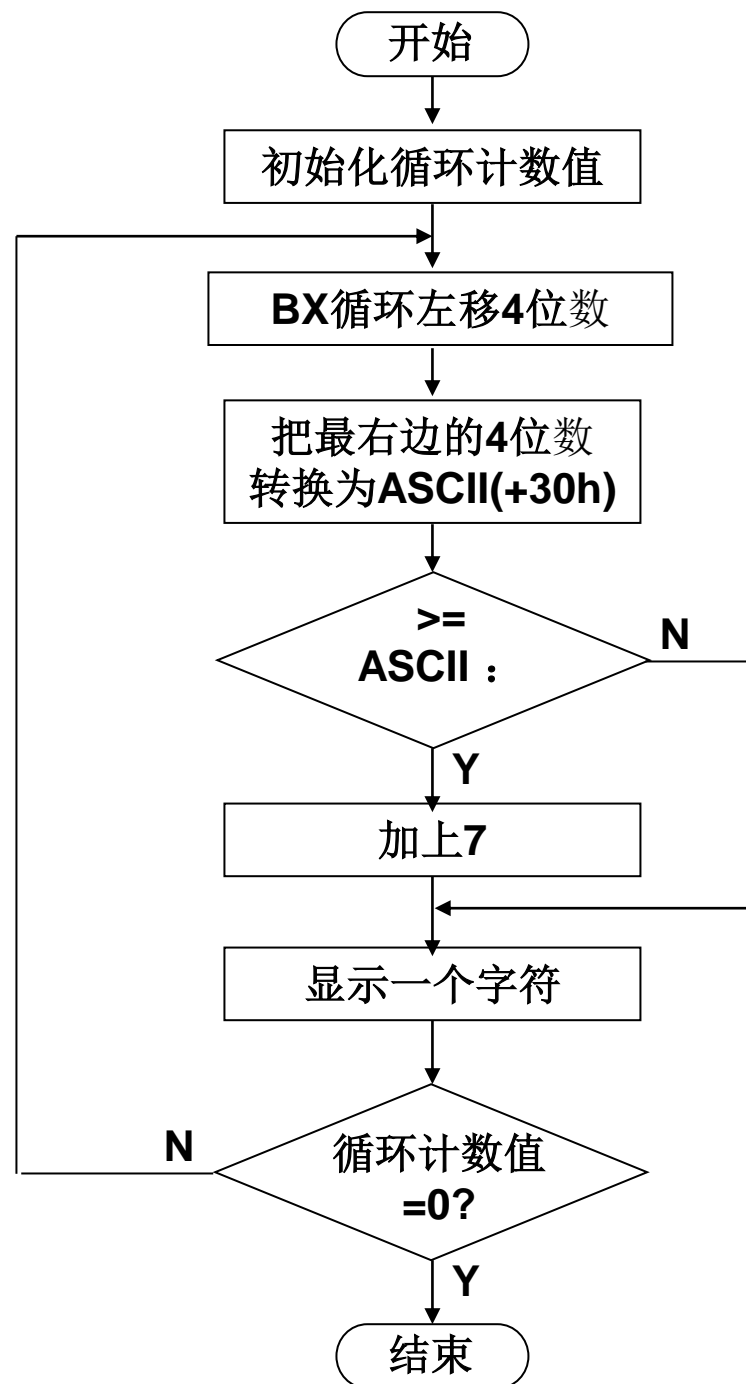
“0”~“9” → 30H~39H, “A”~“F” → 41H~46H

◆ 如何显示一个字符？

(a) 将显示字符的ASCII码放入DL寄存器；(b)将AH的内容置为2（功能号）；(c) 执行INT 21H（DOS 功能调用）。

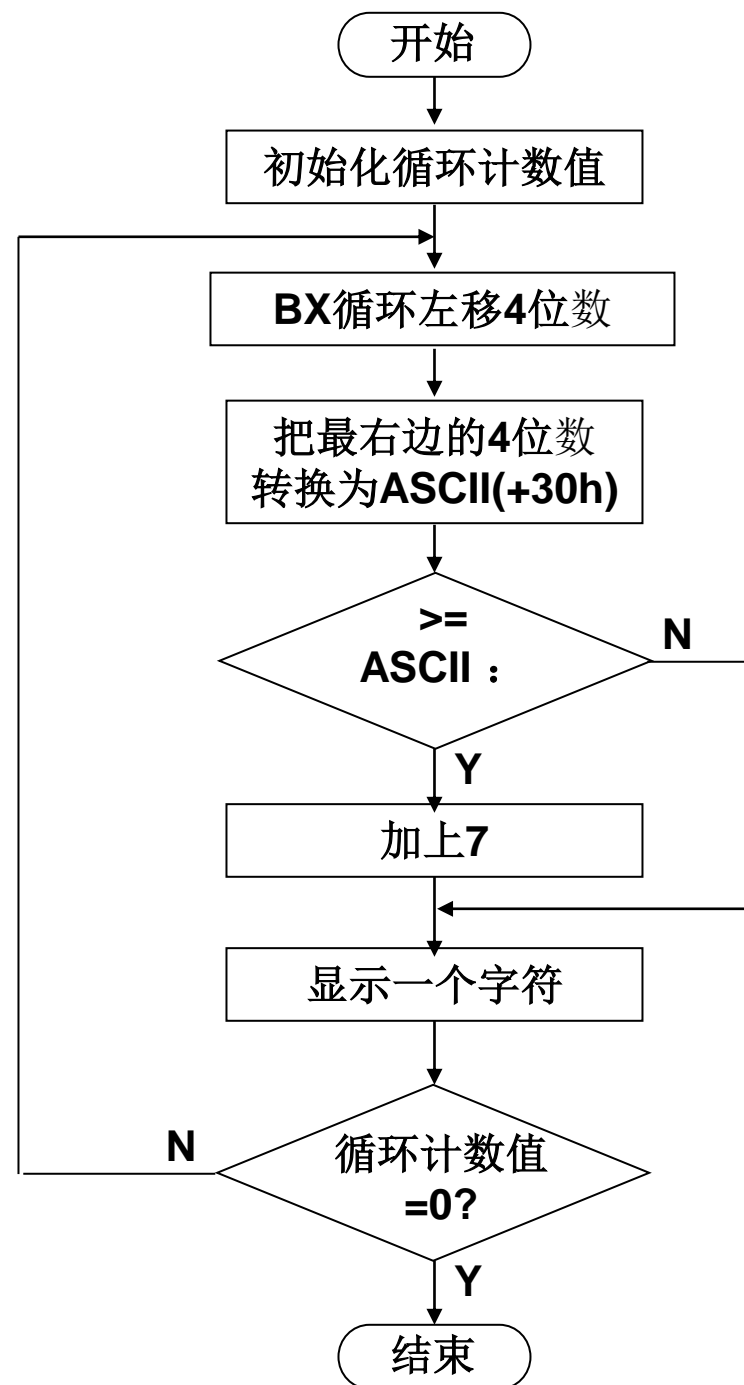
循环结构程序设计

十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符
48	0	64	@	80	P	96	`	112	p
49	1	65	A	81	Q	97	a	113	q
50	2	66	B	82	R	98	b	114	r
51	3	67	C	83	S	99	c	115	s
52	4	68	D	84	T	100	d	116	t
53	5	69	E	85	U	101	e	117	u
54	6	70	F	86	V	102	f	118	v
55	7	71	G	87	W	103	g	119	w
56	8	72	H	88	X	104	h	120	x
57	9	73	I	89	Y	105	i	121	y
58	:	74	J	90	Z	106	j	122	z
59	;	75	K	91	[107	k	123	{
60	<	76	L	92	\	108	l	124	
61	=	77	M	93]	109	m	125	}
62	>	78	N	94	^	110	n	126	~
63	?	79	O	95	_	111	o	127	△



循环结构程序设计

```
datas segment
    temp dw 0B2FAH
datas ends
codes segment
    assume cs:codes, ds:datas
```



循环结构程序设计

datas segment

temp dw 0B2FAH

datas ends

codes segment

assume cs:codes, ds:datas

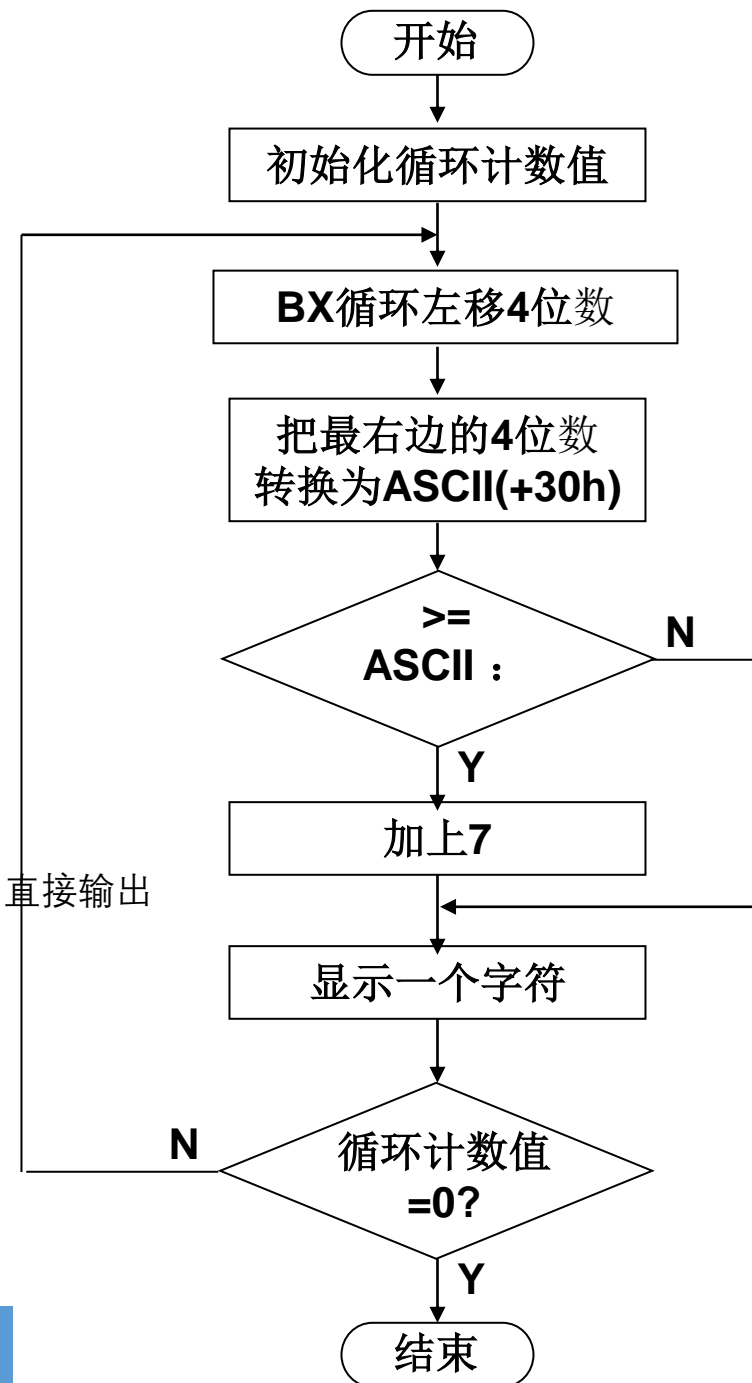
start:

```
rotate:  mov     ax, datas
         mov     ds, ax
         mov     bx, temp
         mov     ch, 4      ; 循环次数
         mov     cl, 4      ; 移位次数
         rol     bx, cl      ; bx循环左移4位
         mov     al, bl      ; 移位后的低8位送al
         and     al, 0fh     ; 取al的低四位
         add     al, 30h     ; 0-9转ascii码, 加30h
         cmp     al, 3ah     ; 比较是否是a-f
         jl      printit    ; 如果小于3ah, 说明是0-9, 直接输出
         add     al, 7h      ; 反之说明是a-f, 则加7
```

codes ends

end start

思考：为什么不用LOOP指令实现？



循环结构程序设计

datas segment

temp dw 0B2FAH

datas ends

codes segment

assume cs:codes, ds:datas

start:

```

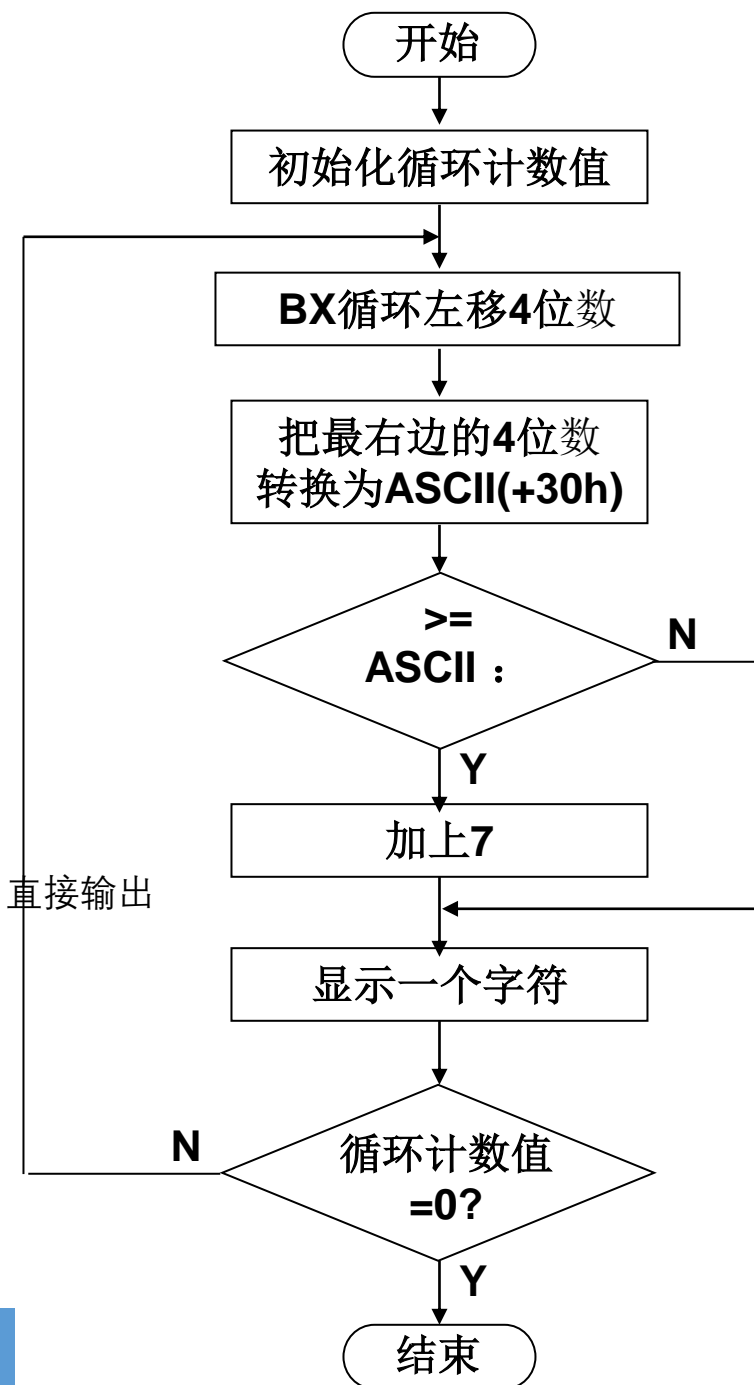
        mov     ax, datas
        mov     ds, ax
        mov     bx, temp
        mov     ch, 4      ; 循环次数
rotate:  mov     cl, 4      ; 移位次数
        rol     bx, cl     ; bx循环左移4位
        mov     al, bl     ; 移位后的低8位送al
        and     al, 0fh    ; 取al的低四位
        add     al, 30h    ; 0-9转ascii码, 加30h
        cmp     al, 3ah    ; 比较是否是a-f
        jl      printit   ; 如果小于3ah, 说明是0-9, 直接输出
        add     al, 7h     ; 反之说明是a-f, 则加7
printit: mov     dl, al
        mov     ah, 2
        int     21h
        dec     ch        ; 循环次数减1
        jnz     rotate    ; 若zf不为0则循环
        mov     ah, 4ch
        int     21h

```

codes ends

end start

思考：为什么不用LOOP指令实现？



循环结构程序设计

```
.....
        mov     cx, 4          ; 初始化
rotate:  push    cx
        mov     cl, 4
        rol     bx, cl
        mov     al, bl
        and     al, 0fh
        add     al, 30h        ; '0'~'9' ASCII 30H~39H
        cmp     al, 3ah
        jnl     printit
        add     al, 7h         ; 'A'~'F' ASCII 41H~46H
printit: mov     dl, al
        mov     ah, 2
        int     21h
        pop     cx
        loop    rotate
.....
```

方法2 (LOOP)

循环结构程序设计

【例5.5】 有数组 $x(x_1, x_2, \dots, x_{10})$ 和 $y(y_1, y_2, \dots, y_{10})$, 编程计算 $z(z_1, z_2, \dots, z_{10})$

$$z_1 = x_1 + y_1$$

$$z_2 = x_2 + y_2$$

$$z_3 = x_3 - y_3$$

$$z_4 = x_4 - y_4$$

$$z_5 = x_5 - y_5$$

$$z_6 = x_6 + y_6$$

$$z_7 = x_7 - y_7$$

$$z_8 = x_8 - y_8$$

$$z_9 = x_9 + y_9$$

$$z_{10} = x_{10} + y_{10}$$

循环结构程序设计

【例5.5】 有数组 $x(x_1, x_2, \dots, x_{10})$ 和 $y(y_1, y_2, \dots, y_{10})$, 编程计算 $z(z_1, z_2, \dots, z_{10})$

$$z_1 = x_1 + y_1$$

$$z_2 = x_2 + y_2$$

$$z_3 = x_3 - y_3$$

$$z_4 = x_4 - y_4$$

$$z_5 = x_5 - y_5$$

$$z_6 = x_6 + y_6$$

$$z_7 = x_7 - y_7$$

$$z_8 = x_8 - y_8$$

$$z_9 = x_9 + y_9$$

$$z_{10} = x_{10} + y_{10}$$

逻辑尺: 0 0 1 1 1 0 1 1 0 0

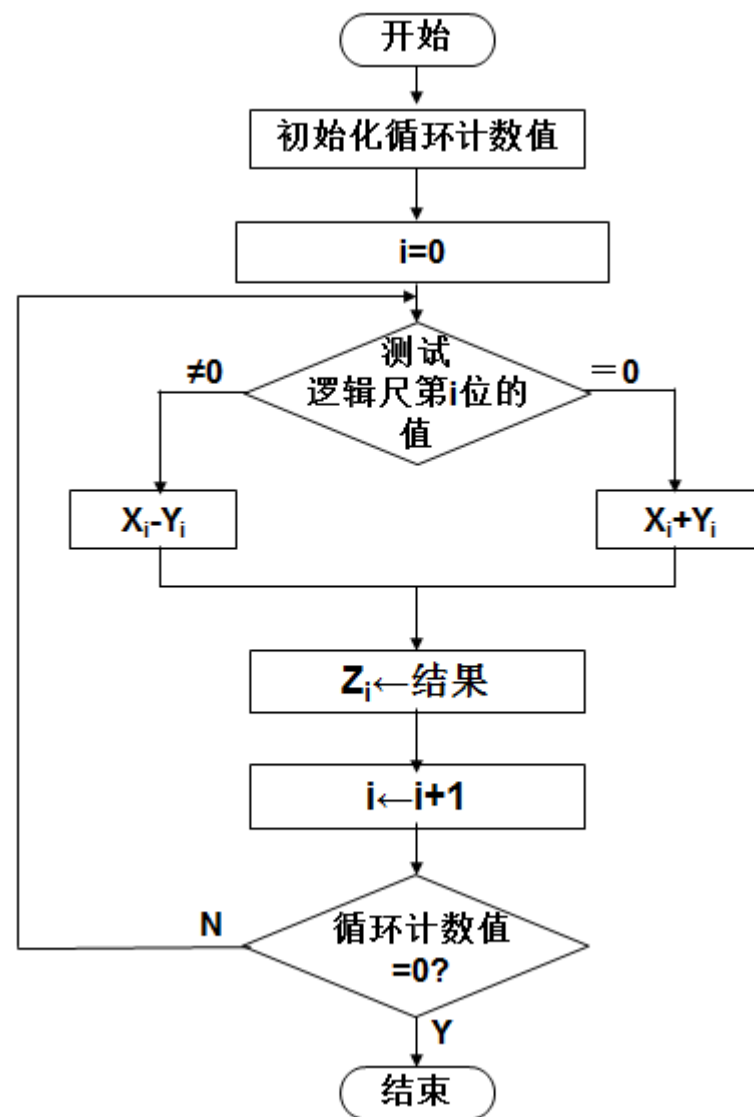
1 减法

0 加法

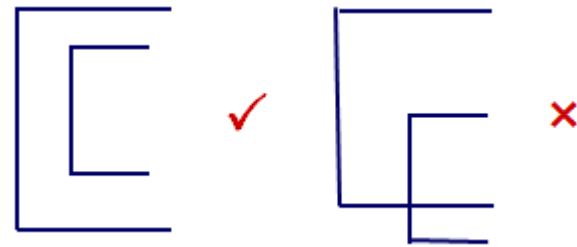
循环结构程序设计

逻辑尺: 0011101100

```
x    dw x1,x2,x3,x4,x5,x6,x7,x8,x9,x10
y    dw y1,y2,y3,y4,y5,y6,y7,y8,y9,y10
z    dw z1,z2,z3,z4,z5,z6,z7,z8,z9,z10
logic_rule dw 0DCH ;0000,0000,1101,1100
.....
mov  bx, 0          ; 数组索引
mov  cx, 10         ; 循环次数
mov  dx, logic_rule
next: mov ax, x[bx]
      shr dx, 1      ; 逻辑尺逻辑右移
      jc  subtract   ; CF=1, 减法
      add ax, y[bx]   ; CF=0, 加法
      jmp short result ; 跳转
subtract:
      sub ax, y[bx]
result:
      mov z[bx], ax   ; 输出到Z
      inc bx          ; 修改数组索引
      loop next       ; CX不为0则循环
.....
```



循环结构程序设计



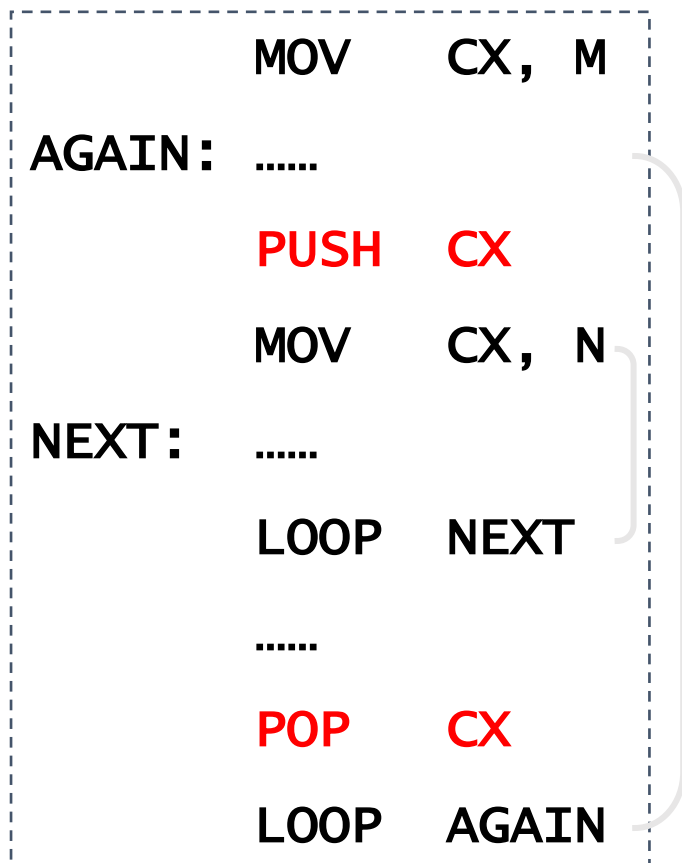
多重循环程序设计:

循环可以有多重结构。多重循环程序设计的基本方法和单重循环程序设计基本一致的，应分别考虑各重循环的控制条件及其程序实现，相互之间不能混淆。

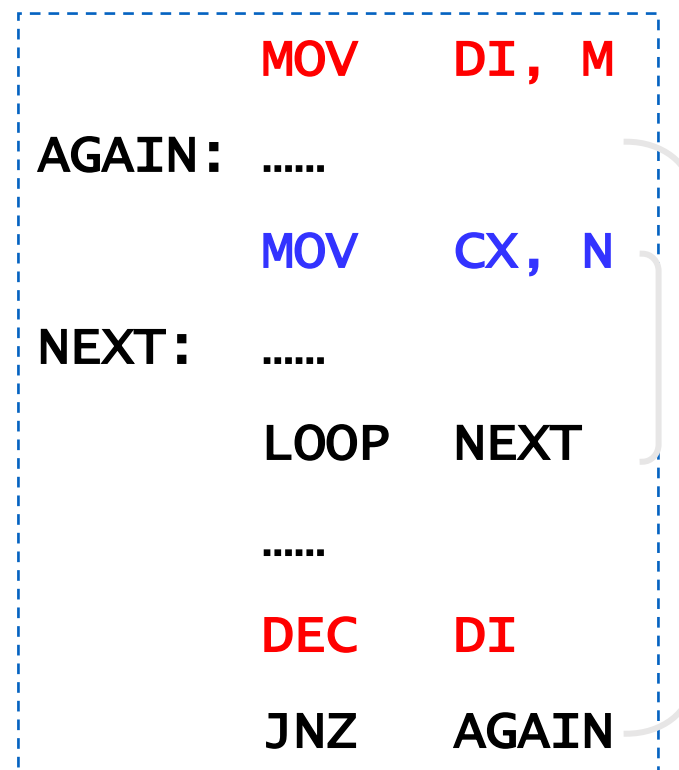
- 内循环必须完整地包含在外循环内，内外循环不能相互交叉。
- 内循环在外循环中的位置可根据需要任意设置，在设计内、外循环时要避免出现混乱。
- 多个内循环可以拥有一个外循环，这些内循环间的关系可以是嵌套的，也可以是并列的。当通过外循环再次进入内循环时，内循环中的初始条件必须重新设置。
- 程序可以从内循环中直接跳到外循环，但不能从外循环直接跳到内循环中。
- 无论是外循环，还是内循环，注意不要使循环返回到初始部分，以避免出现“死循环”情况。

循环结构程序设计

注意：在多重循环的程序结构中，要注意CX 计数器的保存和恢复



利用堆栈保存和恢复CX



分别用不同的寄存器来进行内外循环计数，避免出错。

循环结构程序设计

【例5.7】 有一个首地址为A的N字数组，编写程序使该数组中的数按照从大到小的次序整序。

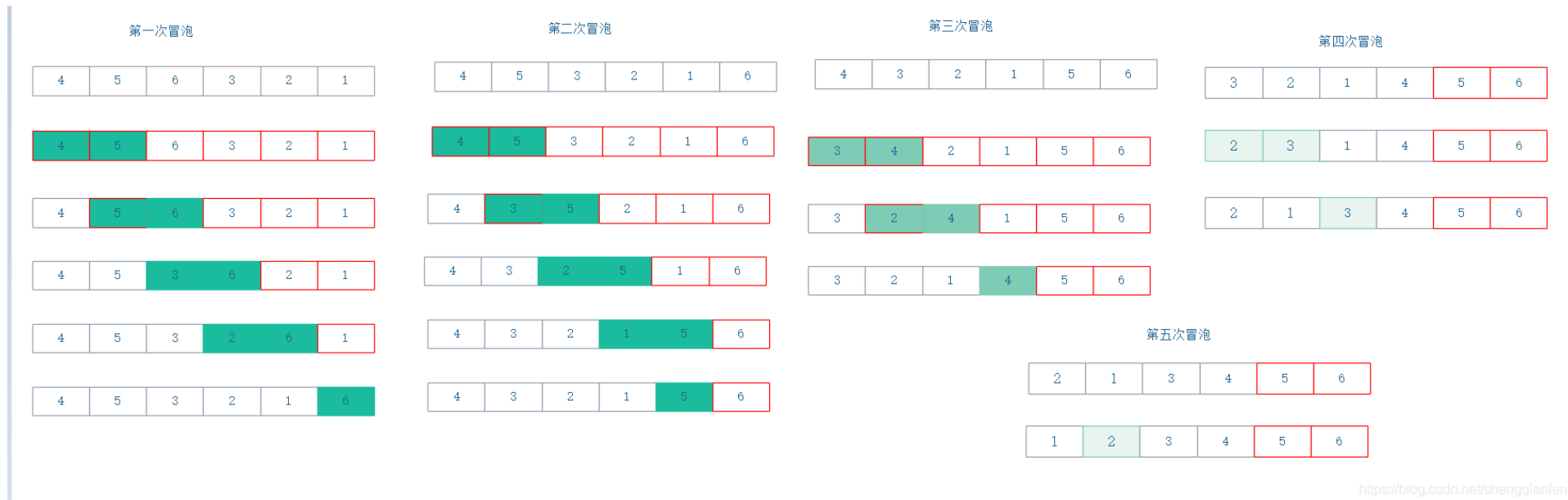
(冒泡算法，多重循环)

冒泡法排序算法思想：

初始时，数组为无序排列。在排序过程中，数组会有一部分元素处于无序状态，称为无序集合，另一部分元素处于有序状态，称为有序集合。

冒泡法算法有一基本操作步骤，使得无序集合减少一个元素，有序集合增加一个元素。

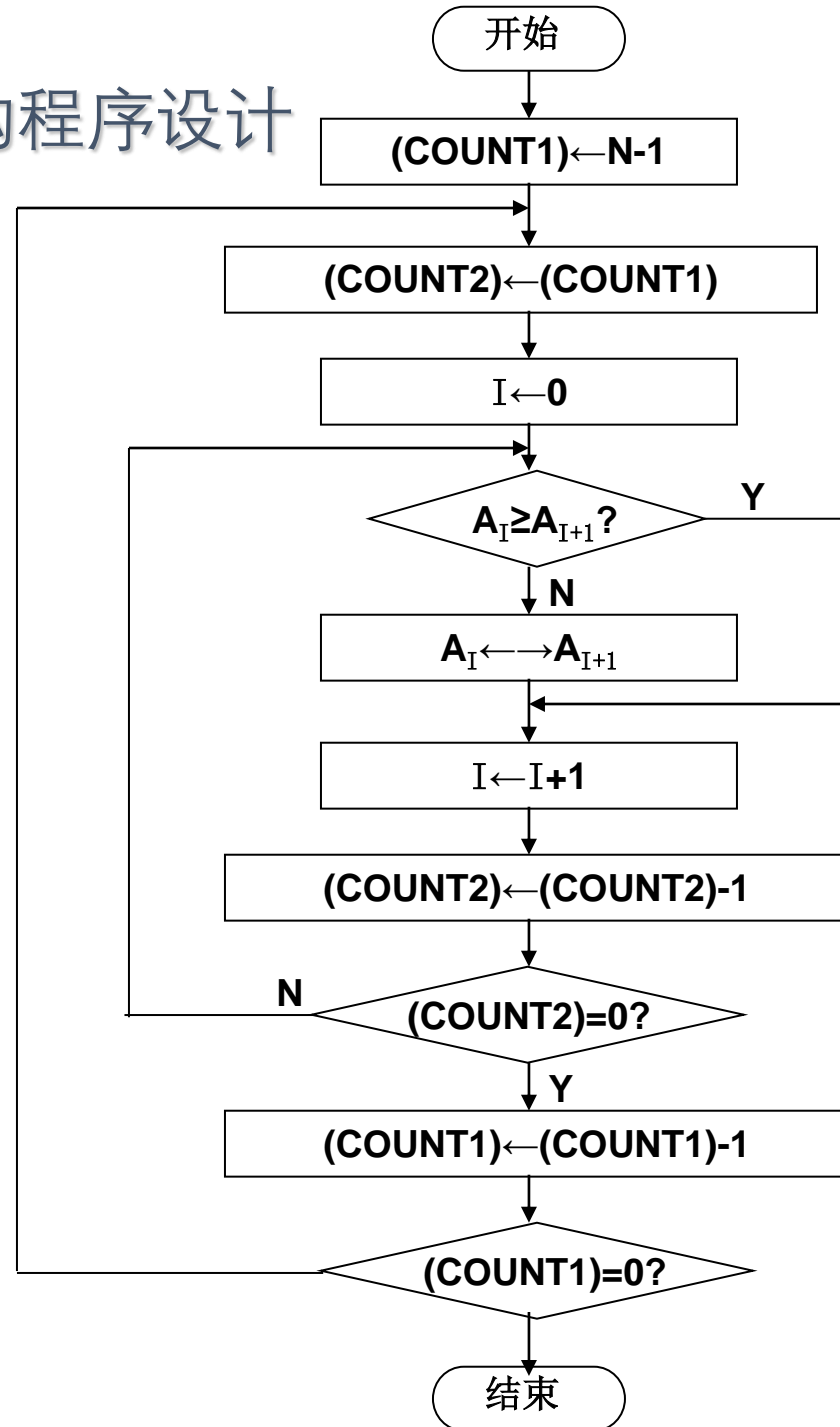
循环结构程序设计



<https://blog.csdn.net/shengqianfeng>

- ◆ N个数排序，第一遍比较了N-1次，把最大的数排到了第N的位置。
- ◆ 第二遍比较了N-2次，把第二大的数排到了第N-1的位置。以此类推
- ◆ 最终只需比较N-1遍，每遍比较次数递减。

循环结构程序设计



Count1:

比较遍数

Count2:

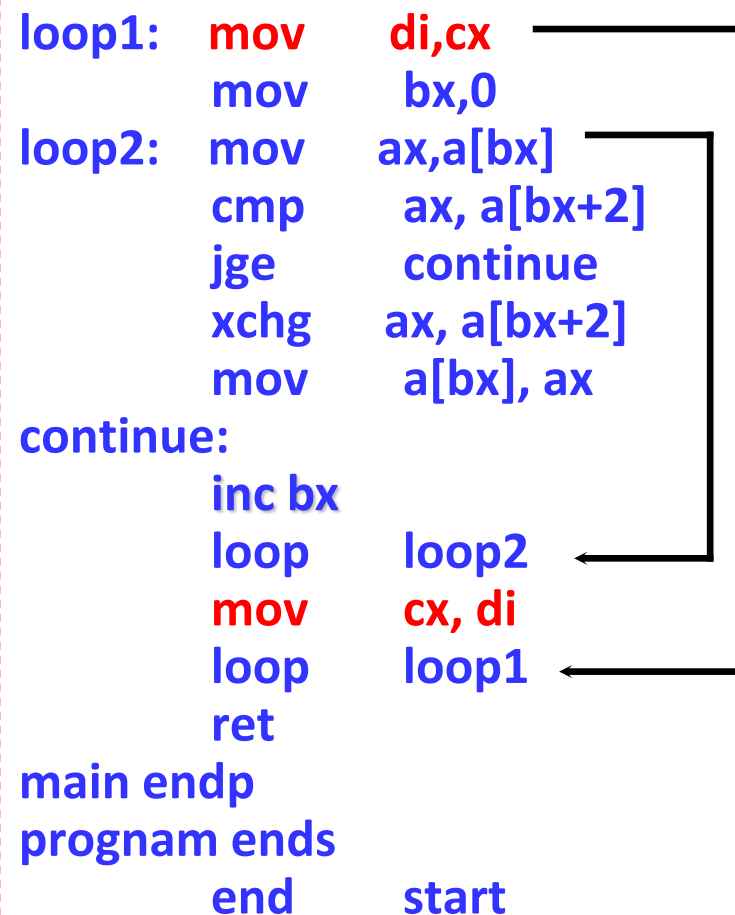
每遍比较次数

循环结构程序设计

```
data    segment
a       dw      4,5,6,3,2,1
n       equ     ($-a)/2
data    ends

prognam segment
main    proc     far
        assume  cs:prognam,ds:data
start:
        mov     ax, data
        mov     ds, ax
        mov     cx, n
        dec     cx      ; 比较遍数
```

```
loop1:  mov     di,cx
        mov     bx,0
loop2:  mov     ax,a[bx]
        cmp     ax, a[bx+2]
        jge     continue
        xchg    ax, a[bx+2]
        mov     a[bx], ax
continue:
        inc     bx
        loop    loop2
        mov     cx, di
        loop    loop1
        ret
main endp
prognam ends
        end     start
```



注意：Loop指令会自动修改CX的值，多重循环的时候要注意CX的保存和恢复。

分枝结构程序设计

<pre>CODE SEGMENT ASSUME CS:CODE START:LEA BX,TAB MOV AH,1 INT 21H SUB AL,30H MOV AH,0 ADD AX,AX ADD BX,AX JMP BX TAB:JMP SHORT MODE0 ; 转移表 JMP SHORT MODE1 JMP SHORT MODE2 JMP SHORT MODE3 JMP SHORT MODE4</pre>	<pre>MODE0:MOV DL,30H JMP EXIT MODE1:MOV DL,31H JMP EXIT MODE2:MOV DL,32H JMP EXIT MODE3:MOV DL,33H JMP EXIT MODE4:MOV DL,34H EXIT:MOV AH,2 INT 21H MOV AH,4CH INT 21H CODE ENDS END START</pre>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

说明：转移表中每条转移指令占用2个字节（段内短转移），所以有以下计算公式：表地址=模式字*2+表首地址

Q&A



Fall 2023