第五章　程序控制结构及其设计技术

本章主要内容：

- 顺序程序设计
- 无条件转移指令
- 条件转移指令
- 分支程序设计
- 循环控制指令
- 循环程序设计

§5.1　顺序程序设计

顺序程序是指程序的结构从开始到结尾一直是顺序执行，中途没有分支。

例 5.2.1 试编写程序计算以下表达式:

Z=(3X+Y-5)/2

设X、Y的值放在字变量VARX、VARY中，结果存放在VARZ中。

算法分析:

1、乘2^n和除2^n可以使用算术左移和右移实现

2、其它非2^n的乘除运算可以用移位和加减组合运算来实现。如 3X可以分解成2X+X

```
TITLE  EQUATION COMPUTE
DATA   SEGMENT
VARX   DW  15
VARY   DW  10
VARZ    DW  ?
DATA    ENDS
STACK1 SEGMENT PARA STACK
                DW 20H DUP (0)
STACK1   ENDS
CODE   SEGMENT
          ASSUME CS:CODE,DS:DATA,SS:STACK1
```

```
COMP  PROC  FAR
      PUSH  DS
      MOV  AX,0
      PUSH AX
      MOV AX,DATA
      MOV DS,AX
      MOV AX,VARX
      SHL AX,1              ;  2*X
      ADD AX，VARX    ;  3*X
      ADD AX，VARY    ;  3X+Y
      SUB  AX，5            ;  3*X+Y-5
      SAR  AX，1            ;  （3*X+Y-5）/2
      MOV VARZ，AX   ;  存结果
      RET
COMP  ENDP
CODE   ENDS
       END COMP
```

例5.2.2 利用学号查学生的数学成绩表。

算法分析：首先在数据段中建立一个成绩表TABLE，在表中各学生的成绩按照学号从小到大的顺序存放。要查的学号存放在变量NUM中，查表的结果放在变量MATH中。

```
    TITLE   TABLE      LOOKUP
DATA     SEGMENT
TABLE    DB   81,78,90,64,85,76,93,82,57,80
         DB   73,62,87,77,74,86,95,91,82,71
NUM      DB    8
MATH     DB    ?
DATA     ENDS
STACK1   SEGMENT  PARA  STACK
         DW   20H    DUP(0)
STACK1   ENDS
COSEG    SEGMENT
         ASSUME  CS:COSEG,DS:DATA,SS:STACK1
```

```
START:  MOV   AX,DATA
        MOV   DS,AX
        MOV   BX,OFFSET  TABLE    ;BX指向表首址
        XOR   AH,AH
        MOV   AL,NUM
        DEC   AL                  ；实际学号是从1开始的
        ADD   BX,AX               ；BX加上学号指向要查的成绩
        MOV   AL,[BX]             ；查到成绩送AL
        MOV   MATH,AL             ；存结果
        MOV   AH,4CH              ；返回DOS
        INT    21H
COSEG   ENDS
        END   START
```

在上述程序中，如果使用换码指令XLAT，可以简化程序。

换码指令格式为：

XLAT 表首址 ；功能为：AL<=((BX)+(AL))

其中表首址可以省略。

在XLAT指令执行前，要求将表首址的偏移量送入BX中，待查项与表首址之间的字节距离 (0~255)送入AL中。

§5.2分支程序设计

分支程序结构是指程序的执行顺序将根据某些指令的执行结果，选择某些指令执行或不执行。

分支程序的实现主要是由转移指令完成。

一.转移指令

1. 无条件转移指令

格式：JMP 目标

目标是程序中的一个标号，表示转移指令所转移的目的地指令的地址。

程序结构：

```
        :
    JMP   TARGET
        :
TARGET:  .....
        :
```

根据目标所在的位置，JMP指令分为段内转移和段间转移。

（1） 段内转移

JMP指令与转移目标位于同一个代码段内。转移时IP寄存器内容被改变，而CS保持不变。

目标地址可以有两种提供方法：

A. 段内转移直接寻址---- 指令中直接给出转移目的地标号

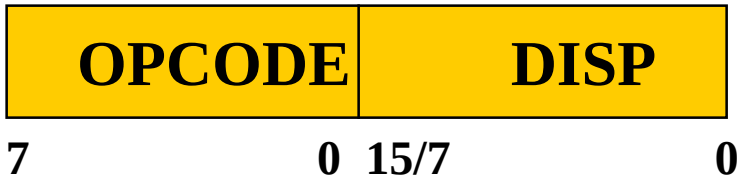
例如

```
        ……
        JMP  LABEL1
        ……
LABEL1: ……
```

或者

```
        ……
LABEL2: ……
        ……
        JMP  LABEL2
```

指令编码:



OPCODE字段长度为一个字节, DISP字段根据OPCODE字段为不同编码时分别为1个或2个字节。 DISP 为相对位移量，用补码表示。

OPCODE=
- 0EBH时， 为短转移， DISP为8位， 转移偏移量：-128—+127 字节
- 0E9H时， 为长转移， DISP为16位， 转移偏移量：-32768—+32767

指令的功能为： IP <=（ IP ）+ DISP

注意：转移偏移量是相对转移指令的下一条指令的起始地址

如果是相对于该转移指令的地址而言，则相对偏移量的值为：

-126~+129字节 或 -32765~+32770

B. 段内转移间接寻址——指令中指定一个*16位*通用寄存器*或*字存储单元，其内容为转移目标地址。

例如：JMP CX
JMP WORD PTR [BX][SI]

指令编码格式：

指令执行时，由MOD、R/M以及位移量确定一个寄存器或有效地址EA，将所指寄存器或存储单元内容送入IP中。

IP<=(通用寄存器） **或** **IP<=(EA)**

（2）段间转移——JMP指令与目标地址不在同一个段内

执行该转移指令，将同时改变CS和IP的内容。

A、段间转移直接寻址

在JMP指令中，目标地址符前面加属性说明符FAR。

例如：

```
COSEG1 SEGMENT                    COSEG2  SEGMENT
          :
        JMP  FAR PTR TARGET       TARGET：  .........
          :                             :
COSEG1  ENDS                      COSEG2  ENDS
```

指令编码格式：

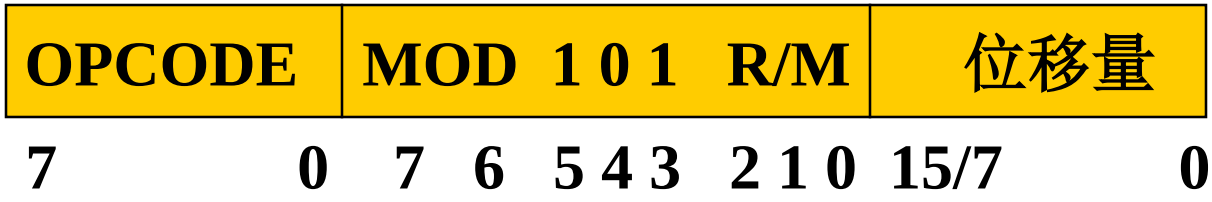
指令执行时，将有：

IP<= 目标地址偏移量

CS<=目标地址段基值

B. 段间间接寻址——目标地址存放在一个双字存储单元中。低地址字单元内容为偏移量，高地址字单元内容为段基值。

指令编码格式：



指令执行时，将有：

IP<= (EA)　　EA字单元内容

CS<= (EA+2)　　EA+2字单元内容

例：JMP DWORD PTR ADDR1；双字单元ADDR1的内容为转移目的地的偏移量和段基值。

JMP DWORD PTR [BX] ；由BX所指向的一个双字存储单元内容为转移目的地的偏移量和段基值。

2. 条件转移指令

8086/8088指令系统有18条条件转移指令

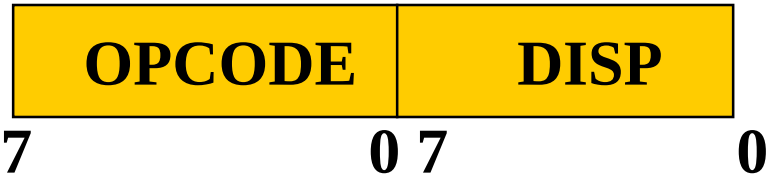
一般格式为：JXX 目标

其中：XX为1—2个字母组合，用来表示各种条件。

执行该指令时，若指定的条件成立，则转移至目标处。否则顺序执行。

条件用标志寄存器中的一个或几个标志位的状态来表示。

指令编码格式：



指令格式与段内无条件转移直接寻址指令的情况相似.但是，该指令中的DISP的长度为一个字节。 因此转移范围为 -128—+127字节。

条件转移指令分为三大类：

（1）简单条件转移指令——条件为单个标志位的状态

标志位	指　　令	转移条件	含　　义
CF	JC	CF=1	有进位/借位转移
	JNC	CF=0	无进位/借位转移
ZF	JE/JZ	ZF=1	相等/等于0转移
	JNE/JNZ	ZF=0	不相等/不等于0转移
SF	JS	SF=1	是负数转移
	JNS	SF=0	是正数转移
OF	JO	OF=1	有溢出转移
	JNO	OF=0	无溢出转移
PF	JP/JPE	PF=1	有偶数个1转移
	JNP/JPO	PF=0	有奇数个1转移

（2）无符号数条件转移——在转移指令前执行了两个无符号数A和B相减的指令（A—B）

指　令	转移条件	含　义
JA/JNBE	CF=0 且 ZF=0	A > B 转移
JAE / JNB	CF=0 或 ZF=1	A ≥B 转移
JB /JNAE	CF=1 且 ZF=0	A < B 转移
JBE / JNA	CF=1 或 ZF=1	A ≤ B 转移

(3) 带符号数条件转移指令——在转移指令之前执行了两个带符号数相减（A—B）的指令

指令	转移条件	含　义
JG / JNLE	SF=OF 且 ZF=0	A > B 转移
JGE / JNL	SF=OF 或 ZF=1	A ≥B 转移
JL / JNGE	SF≠OF 且 ZF=0	A < B 转移
JLE / JNG	SF≠OF 或 ZF=1	A ≤ B 转移

对于带符号数的比较，需要使用符号标志位SF、溢出标志位OF和零标志位ZF来判断。下面以A>B的情况为例进行分析。

A>B可以分为以下几种情况：

1.A和B都为负数

若要A>B，则A-B的结果一定是正数（SF=0),也不会发生溢出（OF=0），且结果不为零（ZF=0）。

2. A和B都为正数

若要A>B，则A-B的结果一定是正数（SF=0），也不会发生溢出（OF=0），并且结果不为零（ZF=0）。

3.A为正数，B为负数

执行A-B的结果可能有两种情况：

（1）不发生溢出。这时结果为正数（SF=0），即有SF=OF。

（2）发生溢出。这时结果变为负数（SF=1），即有SF=OF。

二、分支程序设计

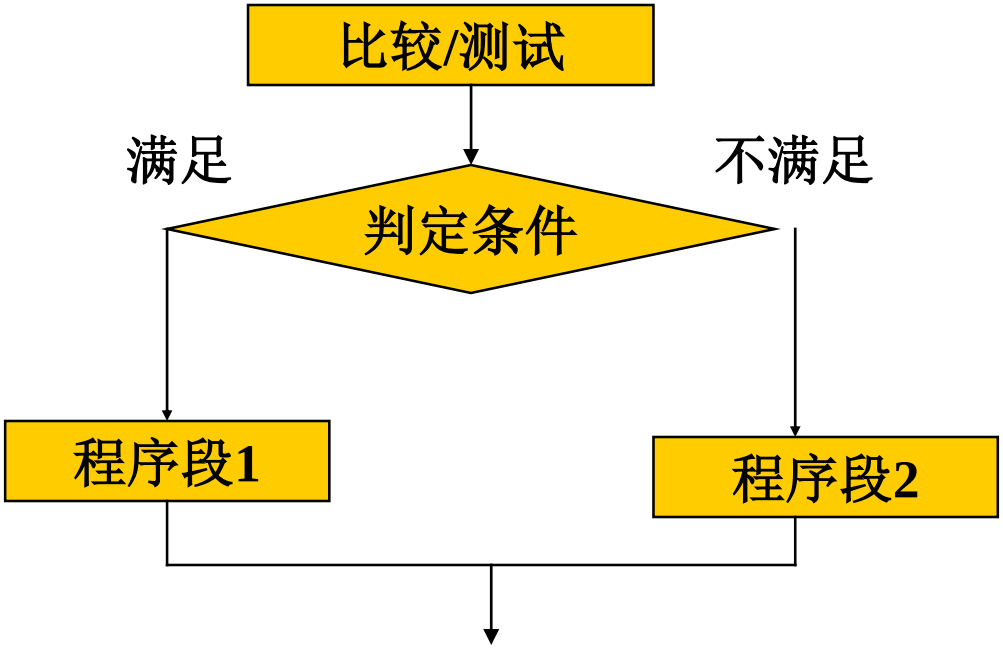
分支程序的结构有两种常见结构：

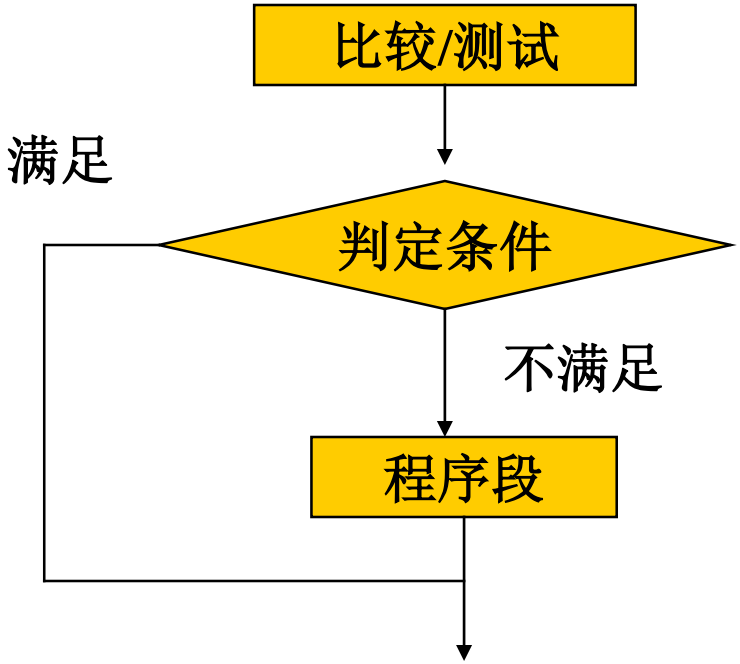
1、用比较/测试指令+条件转移指令实现分支

比较指令：**CMP DEST，SRC**

该指令的功能与减法指令SUB相似，区别是(DEST)-(SRC)的差值不送入DEST。而其结果影响标志位。

这种类型的分支程序有两种结构





一条条件转移指令只能实现两条分支程序的设计。要实现更多条分支的程序，需使用多条条件转移指令。

例5.3.2 数据段的ARY数组中存放有10个无符号数，试找出其中最大者送入MAX单元。

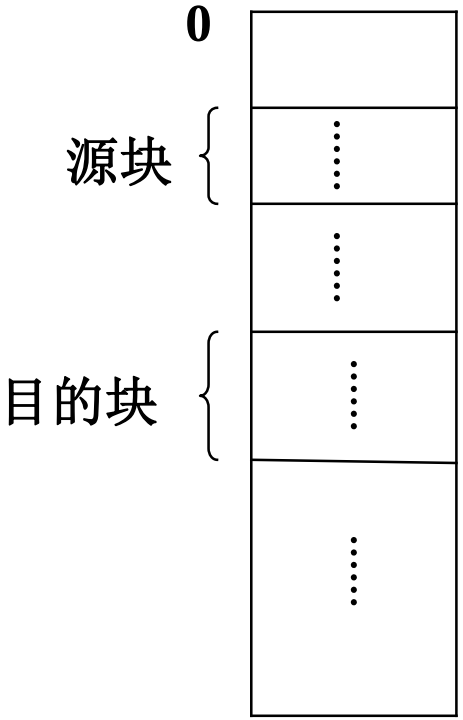
算法分析：

- 依次比较相邻两数的大小，将较大的送入AL中。
- 每次比较后，较大数存放在AL中，相当于较大的数往下传。
- 比较一共要做9次。
- 比较结束后，AL中存放的就是最大数。

```
DATA  SEGMENT
ARY   DB   17，5，40，0，67，12，34，78，32，10
MAX   DB   ？
DATA  ENDS
        ……
        MOV   SI，OFFSET ARY ；SI指向ARY的第一个元素
        MOV   CX，9          ；CX作次数计数器
        MOV   AL，[SI]       ；取第一个元素到AL
LOP：   INC   SI             ；SI指向后一个元素
        CMP   AL，[SI]       ；比较两个数
        JAE   BIGER          ；前元素≥后元素转移
        MOV   AL，[SI]       ；取较大数到AL
BIGER： DEC   CX             ；减1计数
        JNZ   LOP            ；未比较完转回去，否则顺序执行
        MOV   MAX，AL        ；存最大数
        ……
```

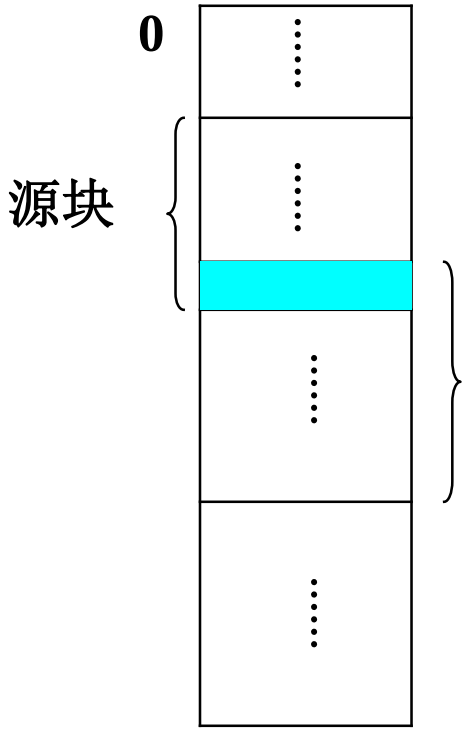
例5.3.4　编写一程序，实现将存储器中的源数据块传送到目的数据块。

在存储器中两个数据块的存放有三种情况：两个数据块分离和有部分重叠。



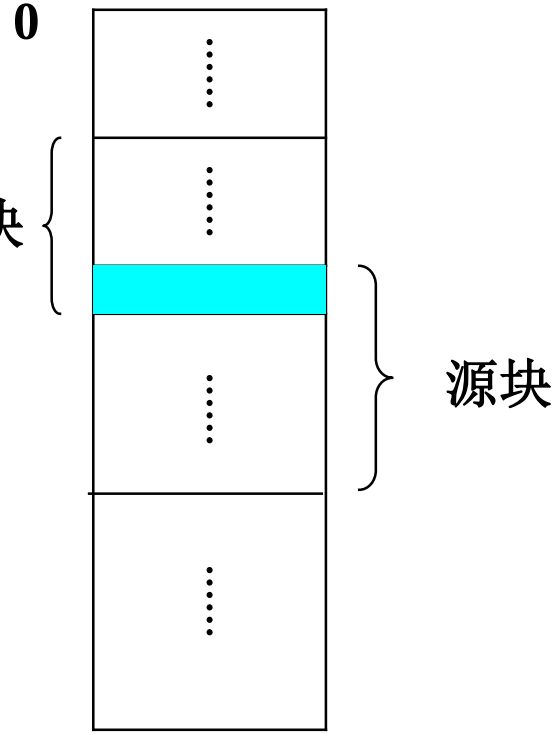
两个数据块分离

可以从首址或末址开始传送



源块首址 < 目的块首址

必须从数据块末址开始传送



源块首址 > 目的块首址

必须从数据块首址开始传送

三种相对位置情况的传送方法：

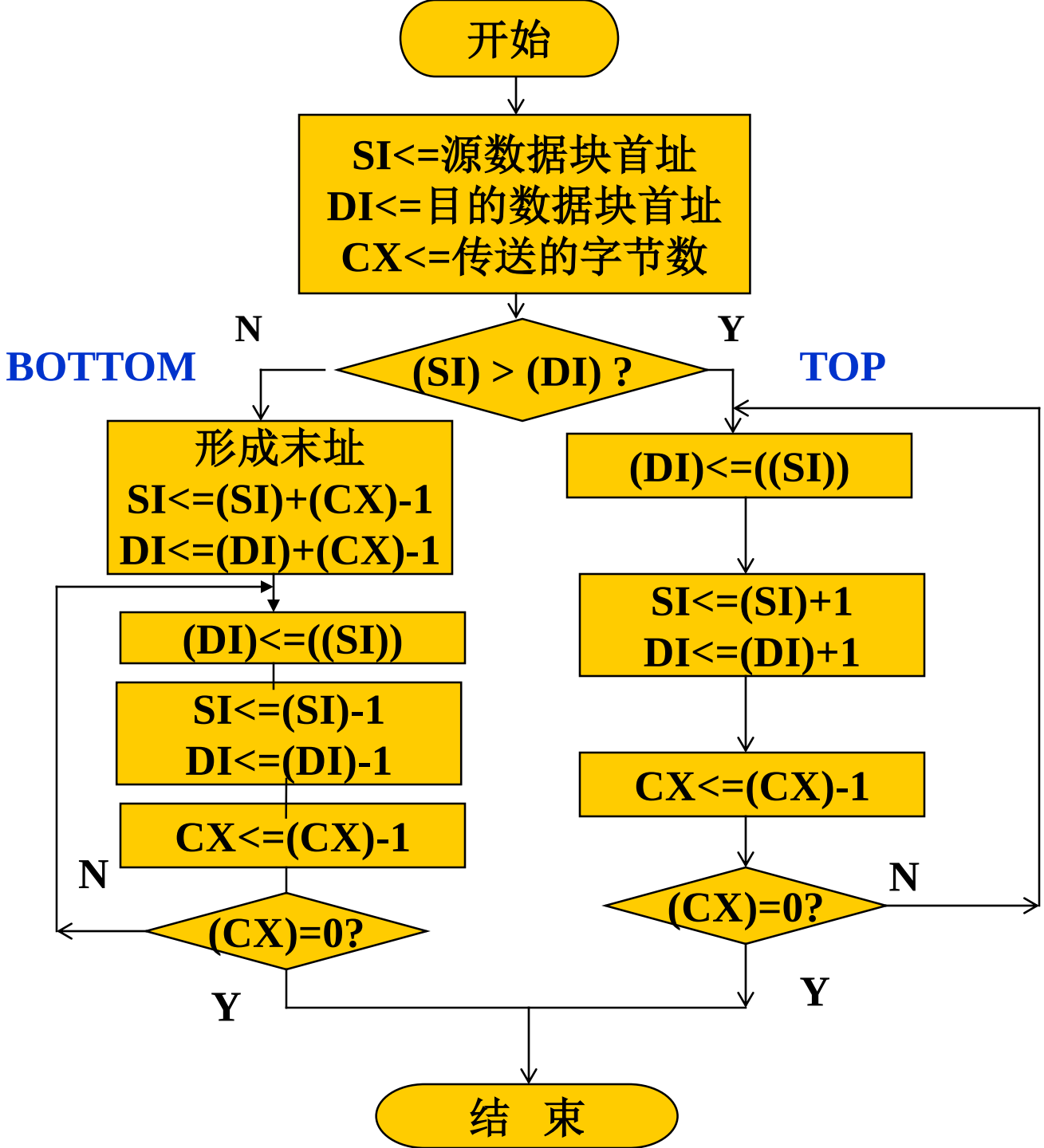
对于源块和目的块分离的情况，不论是从数据块的首址还是末址开始传送都可以。

对于源块与目的块有重叠且源块首址<目的块首址的情况，必须从数据块末址开始传送。

对于源块与目的块有重叠且源块首址>目的块首址的情况，必须从数据块首址开始传送。

将上述三种情况综合起来，只考虑源块和目的块的地址相对大小，传送方法如下：

当源块首地址<目的块首地址时，从数据块末地址开始传送。反之，则从首地址开始传送。


开始
SI<=源数据块首址
DI<=目的数据块首址
CX<=传送的字节数
N
Y
BOTTOM
(SI) > (DI) ?
TOP
形成末址
SI<=(SI)+(CX)-1
DI<=(DI)+(CX)-1
(DI)<=((SI))
SI<=(SI)-1
DI<=(DI)-1
CX<=(CX)-1
N
(CX)=0?
Y
(DI)<=((SI))
SI<=(SI)+1
DI<=(DI)+1
CX<=(CX)-1
N
(CX)=0?
Y
结 束

```
 TITLE  DATA BLOCK MOVE
DATA   SEGMENT
            ORG  $+20H
STRG      DB   'ABCDEFGHIJ' ;数据块
LENG    EQU  $-STRG            ;数据块字节长度
BLOCK1 DW    STRG              ;源块首址
BLOCK2 DW   STRG-5          ;目的块首址
DATA   ENDS
STACK1 SEGMENT STACK
              DW   20H DUP(0)
STACK1 ENDS
```

```
COSEG  SEGMENT
        ASSUME CS:COSEG,DS:DATA,SS:STACK1
BEGIN:  MOV AX, DATA
        MOV DS,AX
        MOV CX,LENG        ;设置计数器初值
        MOV SI,BLOCK1      ;SI指向源块首址
        MOV DI,BLOCK2      ;DI指向目的块首址
        CMP  SI,DI         ;源块首址>目的块首址吗?
        JA  TOP            ;大于则转到TOP处，否则顺序执行
        ADD SI,LENG-1      ;SI指向源块末址
        ADD DI,LENG-1      ;DI指向目的块末址
```

```
BOTTOM: MOV AL，[SI]        ;从末址开始传送
        MOV [DI], AL
        DEC SI
        DEC DI
        DEC CX
        JNE   BOTTOM
        JMP   END1
  TOP: MOV AL,[SI]          ;从首址开始传送
        MOV [DI],AL
        INC SI
        INC DI
        DEC CX
        JNE TOP
END1:  MOV AH,4CH
        INT 21H
COSEG  ENDS
        END BEGIN
```

2、用跳转表形成多路分支

当程序的分支数量较多时，采用跳转表的方法可以使程序长度变短， 跳转表有两种构成方法：

（1）跳转表用入口地址构成

在程序中将各分支的入口地址组织成一个表放在数据段中，在程序中通过查表的方法获得各分支的入口地址。

例5.3.5 设某程序有10路分支，试根据变量N的值（1~10），将程序转移到其中的一路分支去。

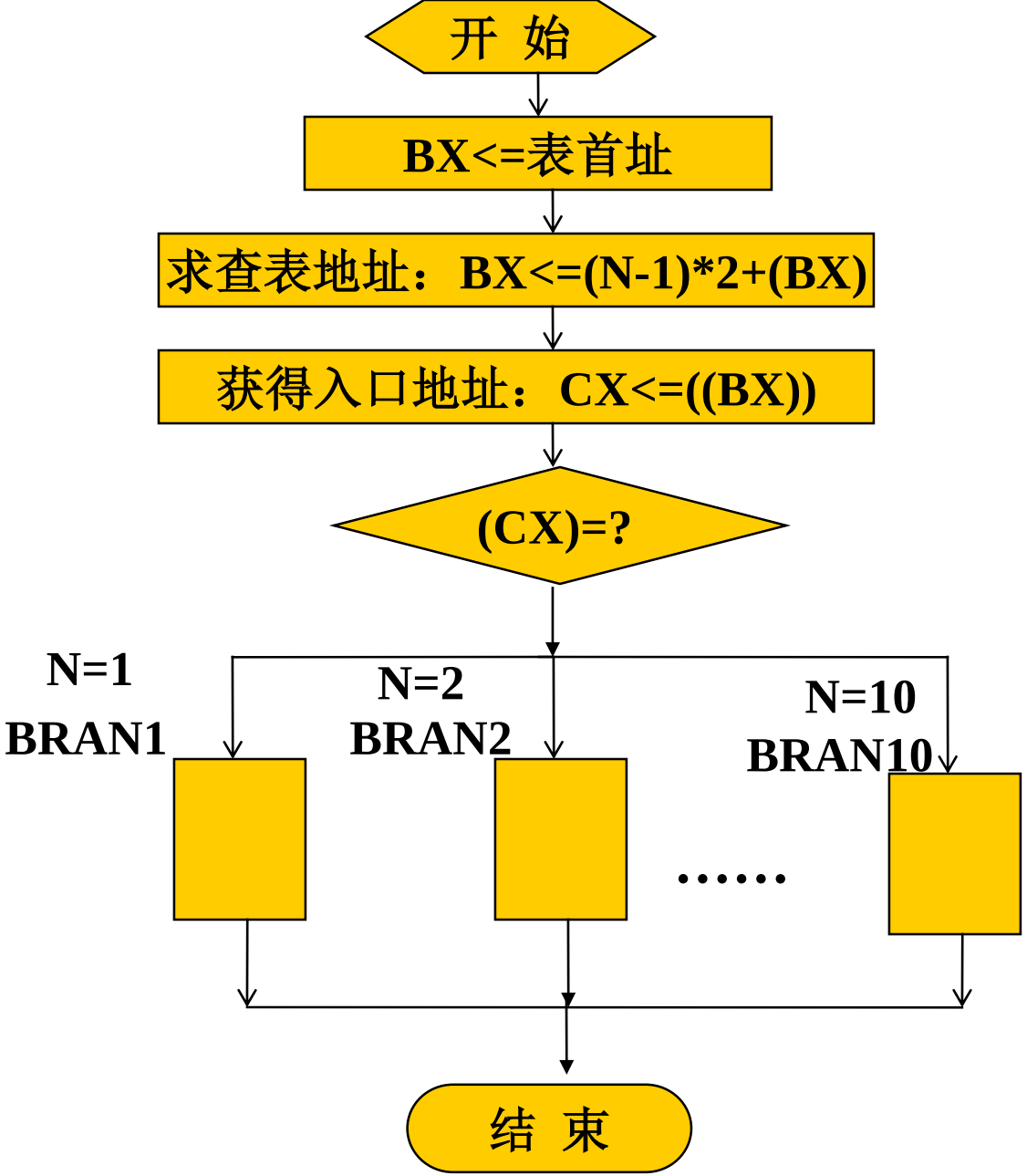
跳转表

设10路分支程序段的入口地址分别为：BRAN1、BRAN2.....BRAN10。

当变量N为1时，转移到BRAN1；N为2时，转移到BRAN2，依次类推。

在跳转表中每两个字节存放一个入口地址的偏移量，如右图所示。

程序中，先根据N的值形成查表地址：（N-1）×2+表首址。


开 始
BX<=表首址
求查表地址：BX<=(N-1)*2+(BX)
获得入口地址：CX<=((BX))
(CX)=?
N=1
BRAN1
N=2
BRAN2
N=10
BRAN10
……
结 束

多路分支结构流程图

```
        TITLE   JUMP TABLE OF ADDRESS
DATA    SEGMENT
ATABLE  DW      BRAN1，BRAN2，BRAN3，...，BRAN10
N       DB      3
DATA    ENDS
STACK1  SEGMENT PARA STACK
        DW      20H DUP（0）
STACK1  ENDS
CODE    SEGMENT
        ASSUME  CS：CODE，DS：DATA，SS：STACK1
START：  MOV     AX，DATA
        MOV     DS，AX
          ...
```

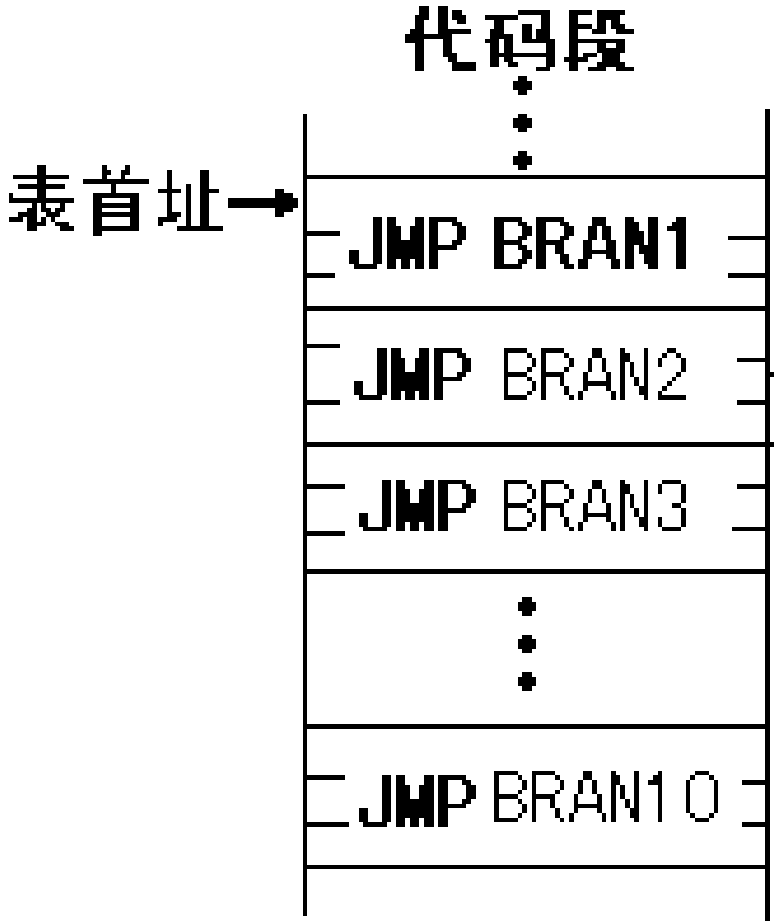
```
XOR    AH，AH
MOV    AL，N
DEC     AL
SHL     AL，1
MOV     BX，OFFSET ATABLE ；BX指向表首址
ADD     BX，AX          ；BX指向查表地址
MOV     CX，[BX]        ；将N对应的分支入口地址送到CX中
JMP     CX              ；转移到N对应的分支入口地址
```

```
BRAN1：  ⋮
         JMP    END1
BRAN2：  ⋮
         JMP    END1
BRAN3：  ⋮
         JMP    END1
           ⋮
BRAN10：⋮

END1：  MOV    AH，4CH
         INT    21H
CODE     ENDS
         END    START
```

（2）跳转表用无条件转移指令构成

跳转表的每一个项目就是一条无条件转移指令。这时跳转表是代码段中的一段程序。



例 5.3.5的源程序可修改为如下程序：

```
TITLE  JUMP TABLE OF INSTRUCTION
DATA     SEGMENT
N          DB  3
DATA     ENDS
STACK1  SEGMENT PARA STACK
           DW   20H DUP(0)
STACK1  ENDS
CODE     SEGMENT
          ASSUME  CS:CODE,DS:DATA,SS:STACK1
START:  MOV  AX,DATA
           MOV  DS,AX
             ...
           MOV  BH,0
           MOV  BL,N
```

```
        DEC   BL              ;四条指令实现(N-1)*3
        MOV  AL,BL
         SHL  BL,1
        ADD  BL,AL
        ADD  BX,OFFSET  ITABLE ;BX指向查表地址
        JMP  BX               ;转移到N对应的JMP指令
ITABLE: JMP  BRAN1            ;JMP指令构成的跳转表
        JMP  BRAN2
        JMP  BRAN3
               :
               :
        JMP  BRAN10
```

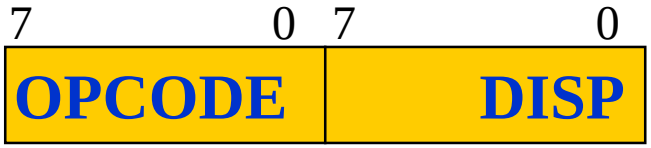
```
BRAN1: ...
          :
          JMP  END1
BRAN2: ...
          :
          JMP  END1
          :
          :
BRAN10: ...
          :
          :
END1:  MOV  AH,4CH
       INT  21H
CODE  ENDS
       END  START
```

5.3 循环程序设计

一、循环控制指令

8086/8088指令系统中有4条循环控制指令，长度都是2字节。

指令编码格式为：



DISP：8位补码表示本指令的下一条指令的首址与目标单元之间的字节距离。

指令中指定一定的条件，若条件满足，则将DISP加入到IP中，即IP<=(IP)+DISP使程序转移到目的指令执行。

指令使用CX寄存器做循环计数。循环控制指令的执行对标志位没有影响。

1、LOOP指令

格式： LOOP 目标

其中目标是程序中的一个标号。

执行一次LOOP指令将使：CX <= (CX)—1

若（CX）≠0，则转到目标处执行，否则顺序执行。

例 5.4.1 在例5.3.2中， 数据段的ARY数组中存放有10个无符号数，试找出其中最大者送入MAX单元。若使用循环指令，则程序可修改如下：

源程序结构如下：

```
DATA    SEGMENT
ARY     DB  17,5,40,0,67,12,34,78,32,10
LEN     EQU  $-ARY
MAX     DB  ?
DATA    ENDS
          :
          :
        MOV  SI,OFFSET  ARY ;SI指向ARY的第一个元素
        MOV  CX,LEN-1       ;  CX作循环(比较)次数计数
        MOV  AL,[SI]
LOP:    INC  SI
        CMP  AL,[SI]
        JAE  BIGER
        MOV  AL,[SI]
BIGER:LOOP LOP
        MOV  MAX,AL
          ……
```

2、LOOPE / LOOPZ指令

格式：LOOPE 目标 或 LOOPZ 目标

指令执行：CX <= (CX) —1，若(CX) ≠0且ZF=1，则转到目标处执行，否则顺序执行。

例 5.4.2 编写一程序，在一字符串中查找第一个非空格字符，并将其在字符串中的序号（1～n)送入INDEX单元中。若未找到，则将INDEX单元置为全1。

```
DATA     SEGMENT
STRG      DB ‘      CHECK NO_SPACE’
LENG      EQU $－STRG
INDEX     DB ?
DATA      ENDS
STACK1   SEGMENT  PARA STACK
          DW  20H DUP(0)
STACK1   ENDS
```

```
CODE    SEGMENT
        ASSUME  CS:CODE,DS:DATA,SS:STACK1
START:  MOV  AX,DATA
        MOV  DS,AX
        MOV  CX,LENG    ;字符串长度送入CX
        MOV  BX,－1     ;设地址指针初值
NEXT:   INC  BX
        CMP  STRG[BX],' '
        LOOPE NEXT   ;是空格字符且计数不为0，继续查找
        JNZ  FOUND     ;找到非空格字符，转FOUND
        MOV  BL，0FEH   ;未找到非空格字符
FOUND:  INC  BL              ;使位置序号从1开始
        MOV  INDEX,BL   ;存结果
        MOV  AH,4CH
        INT  21H
CODE    ENDS
        END  START
```

3、LOOPNE / LOOPNZ指令

使用格式：LOOPNE 目标　或 LOOPNZ 目标

指令执行：CX <= (CX) —1，若(CX) ≠0且ZF=0，则转到目标处执行，否则顺序执行。

例 5.4.3 编写程序，计算两个字节数组ARY1和ARY2对应元素之和，一直计算到两数之和为0或数组结束为止。并将和存入数组SUM中，将该数组的长度存放在NUM单元中。

源程序如下：

```
DATA      SEGMENT
ARY1      DB 12,10,3,5,-1,7,34,8,9,10
ARY2      DB 14,23,6,-2,1,9,45,21,8,24
LENG      EQU  ARY2-ARY1
SUM       DB LENG DUP(?)
NUM       DB ?
DATA      ENDS
STACK1  SEGMENT PARA STACK
          DW 20H DUP(0)
STACK1  ENDS
CODE      SEGMENT
          ASSUME CS:CODE,DS:DATA,SS:STACK1
BEGIN:   MOV AX,DATA
          MOV DS,AX
```

```
        MOV CX,LENG
        MOV BX,－1   ;设置指针初值
NZERO:  INC BX
        MOV AL,ARY1[BX]  ;取被加数
        ADD AL,ARY2[BX]
        MOV SUM[BX]，AL
        LOOPNE NZERO  ；和不为0转到NZERO处
        JZ  ZERO      ；和为0转到ZERO处
        INC BL
ZERO：  MOV NUM，BL  ；存结果数组长度
        MOV AH，4CH
        INT 21H
CODE    ENDS
        END BEGIN
```

4、JCXZ指令

指令格式：JCXZ　目标

该指令测试CX的内容是否为0，如果（CX）=0，则转移到目标处指令，否则顺序执行。

该指令相当于条件转移指令。它一般用在一个循环的开始，当循环次数计数寄存器CX为0时，就不执行该循环。如果没有这个控制，将使得循环次数变得非常大（0-1=0FFFFH），从而产生错误结果。

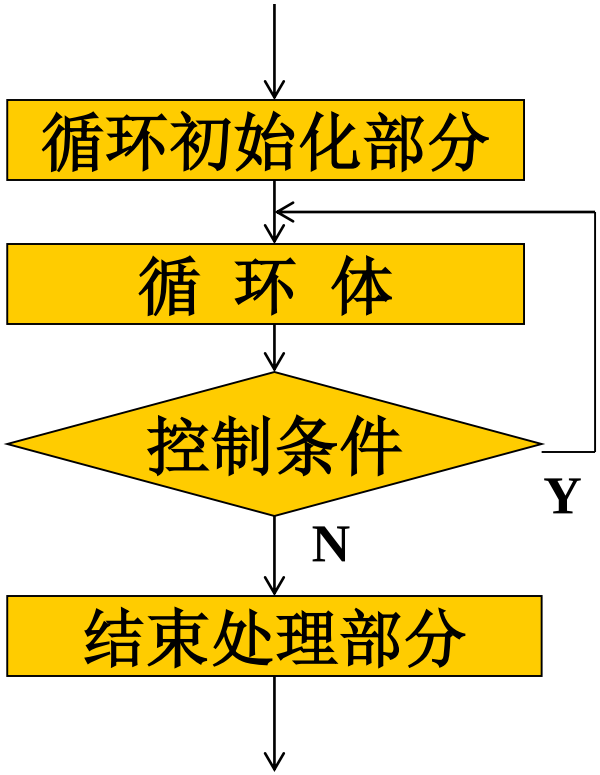
程序结构为：

```
        ..........
        MOV  CX,COUNT
        JCXZ   NEXT
LOP:    ..........
        LOOP   LOP
NEXT:  ..............
```

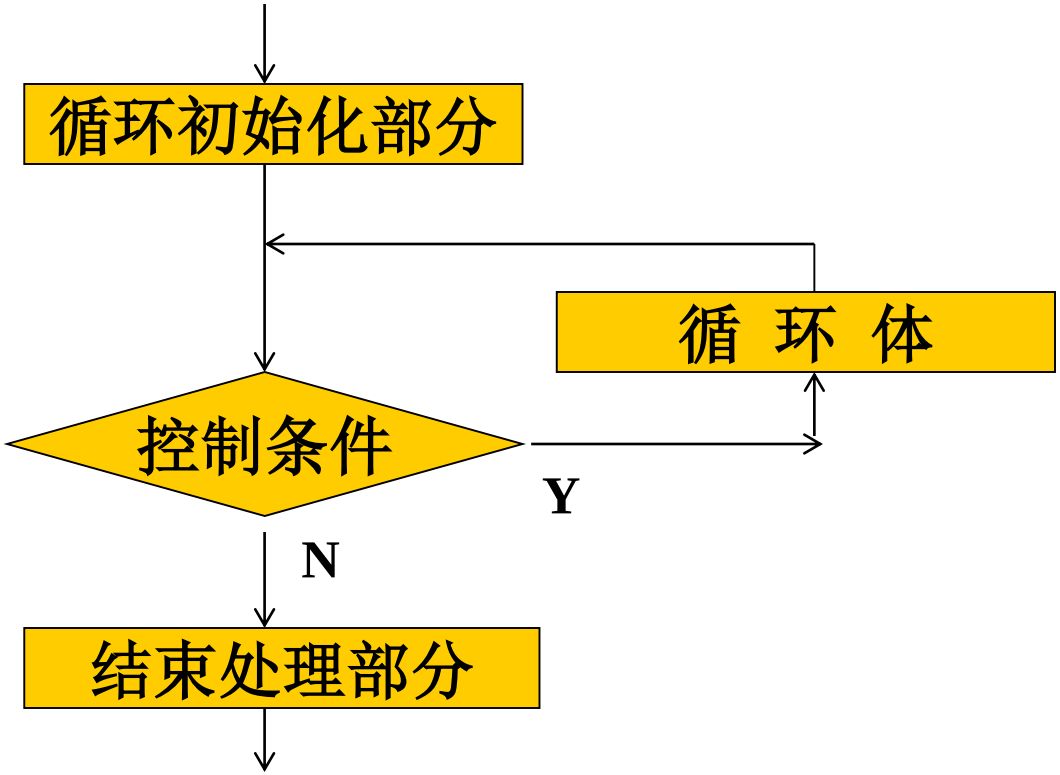
二、循环程序的结构

循环程序有两种结构形式

1、先执行后判断结构



2、先判断后执行结构



在循环程序中主要包括以下四个部分：

1、初始化部分

用于建立循环的初始状态。包括：循环次数计数器、地址指针以及其它循环参数的初始设定。

2、循环体

循环程序完成的主要任务。包括工作部分和修改部分。

工作部分：是完成循环程序任务的主要程序段。

修改部分：为循环的重复执行，完成某些参数的修改。

3、循环控制部分

判断循环条件是否成立。可以有以下两种判断方法：

（1）用计数控制循环——循环次数已知

（2）用条件控制循环——循环次数未知

4、结束处理部分

处理循环结束后的结果。如存储结果等。

三、单循环程序设计

单循环程序的循环体由顺序结构或分支结构组成

1、计数控制循环

常选用CX作计数器，可选用LOOP、LOOPE或LOOPNE等循环控制指令。

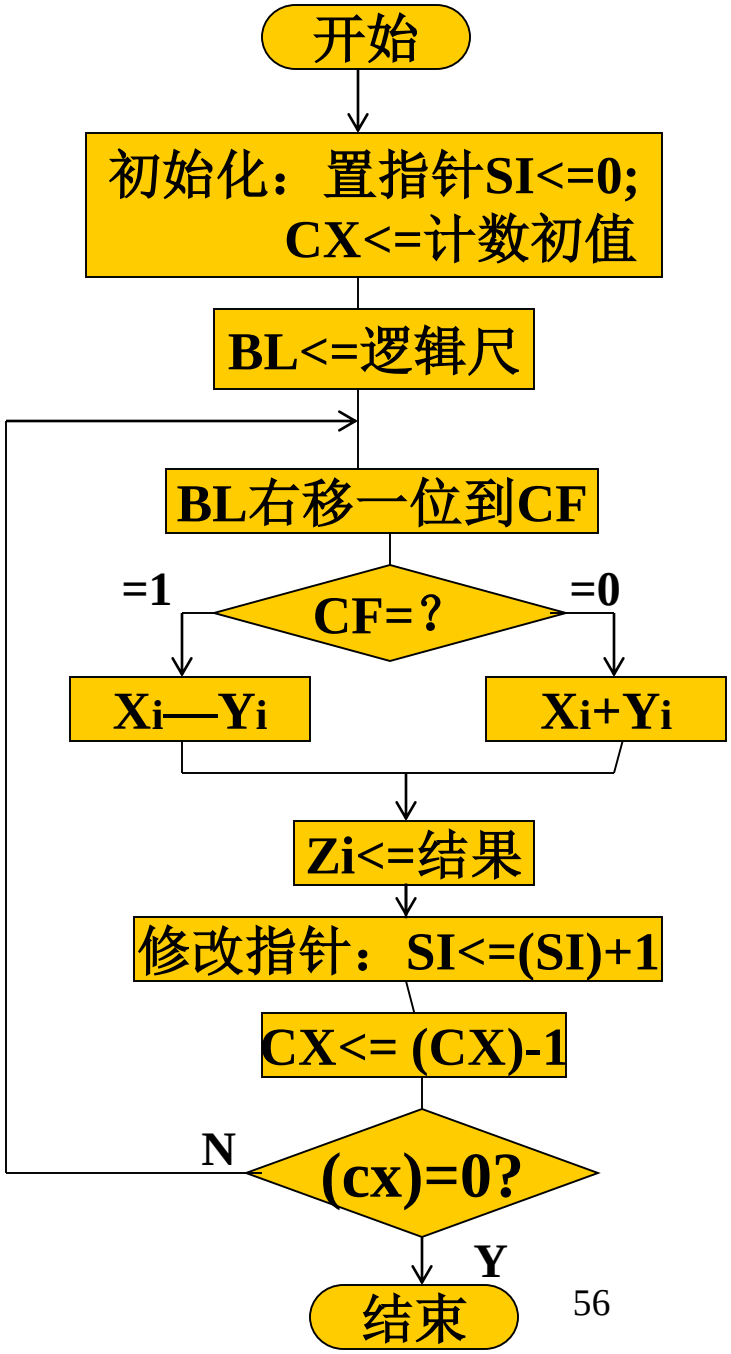
例 5.4.4 设有两个数组X和Y，它们都有8个元素，其元素按下标从小到大的顺序存放在数据段中。试编写程序完成下列计算：

Z1=X1+Y1　Z2=X2-Y2　Z3=X3+Y3

Z4=X4-Y4　Z5=X5-Y5　Z6=X6+Y6

Z7=X7+Y7　Z8=X8-Y8

由于循环体中有“+”和“-”两种可能的运算，通过设置标志0和1来判断。八个运算表达式由8位逻辑尺：10011010B来识别。



```
DATA    SEGMENT
X       DB  0A2H,7CH,34H,9FH,0F4H,10H,39H,5BH
Y       DB  14H,05BH,28H,7AH,0EH,13H,46H,2CH
LEN     EQU $ —Y
Z       DB  LEN DUP(?)
LOGR    DB  10011010B
DATA    ENDS
STACK0 SEGMENT PARA STACK
       DW  20H DUP(0)
STACK0 ENDS
COSEG   SEGMENT
        ASSUME CS:COSEG,DS:DATA,SS:STACK0
BEGIN:   MOV AX,DATA
         MOV DS,AX
         MOV CX,LEN  ;  初始化计数器
         MOV SI,0    ;  初始化指针
         MOV BL,LOGR ;初始化逻辑尺
```

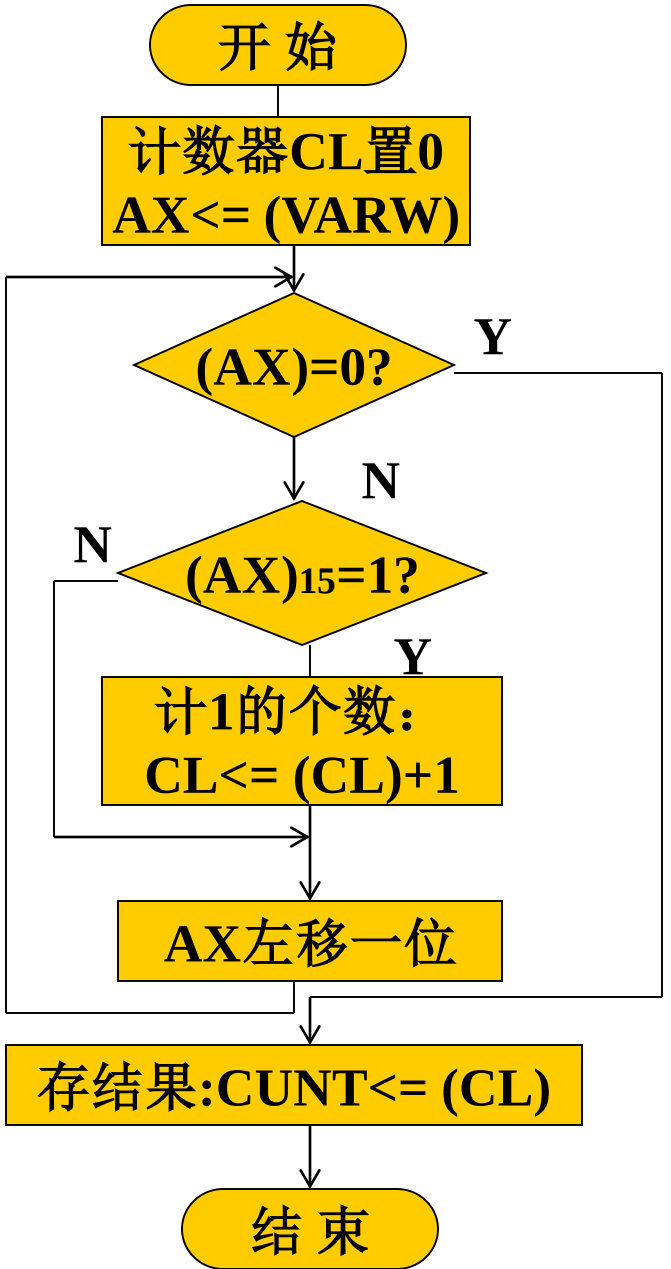
```
LOP:   MOV  AL,X[SI]
       SHR   BL,1          ;标志位送CF
       JC      SUB1        ；为1，转做减法
       ADD   AL,Y[SI] ；为0，做加法
       JMP     RES
SUB1:  SUB    AL,Y[SI]
RES:    MOV   Z[SI],AL ；存结果
        INC     SI          ；修改指针
        LOOP   LOP
        MOV   AH,4CH
        INT      21H
COSEG  ENDS
          END BEGIN
```

2、条件控制循环

例 5.4.5 编写一程序，将用二进制数表示的字单元VARW 中“1”的个数统计出来，存入CONT单元中。

本例中通过将字单元各位逐位移入最高位来判断，而最高位即符号位。为了减少循环次数，循环中加上了判断各位是否全为0，这样可使低位为全0时的循环次数减少。（优化性能，效率高）

其它算法？



```
DATA    SEGMENT
VARW    DW   1101010010001000B
CONT     DB   ?
DATA    ENDS
STACK1  SEGMENT PARA STACK
            DW  20H DUP(0)
STACK1  ENDS
CODE     SEGMENT
           ASSUME  CS:CODE,DS:DATA,SS:STACK1
BEGIN:   MOV  AX,DATA
           MOV  DS,AX
           MOV  CL,0
           MOV  AX,VARW
LOP:      TEST AX,0FFFFH  ；测试（AX）是否为0
           JZ   END0                ；为0，循环结束
           JNS  SHIFT               ；判最高位，为0则转SHIFT
           INC  CL                  ；最高位为1，计数
```

```
SHIFT:   SHL AX,1
         JMP LOP
END0:    MOV  CONT,CL ；存结果
         MOV  AH,4CH
         INT  21H
CODE     ENDS
         END  BEGIN
```

四、多重循环程序设计

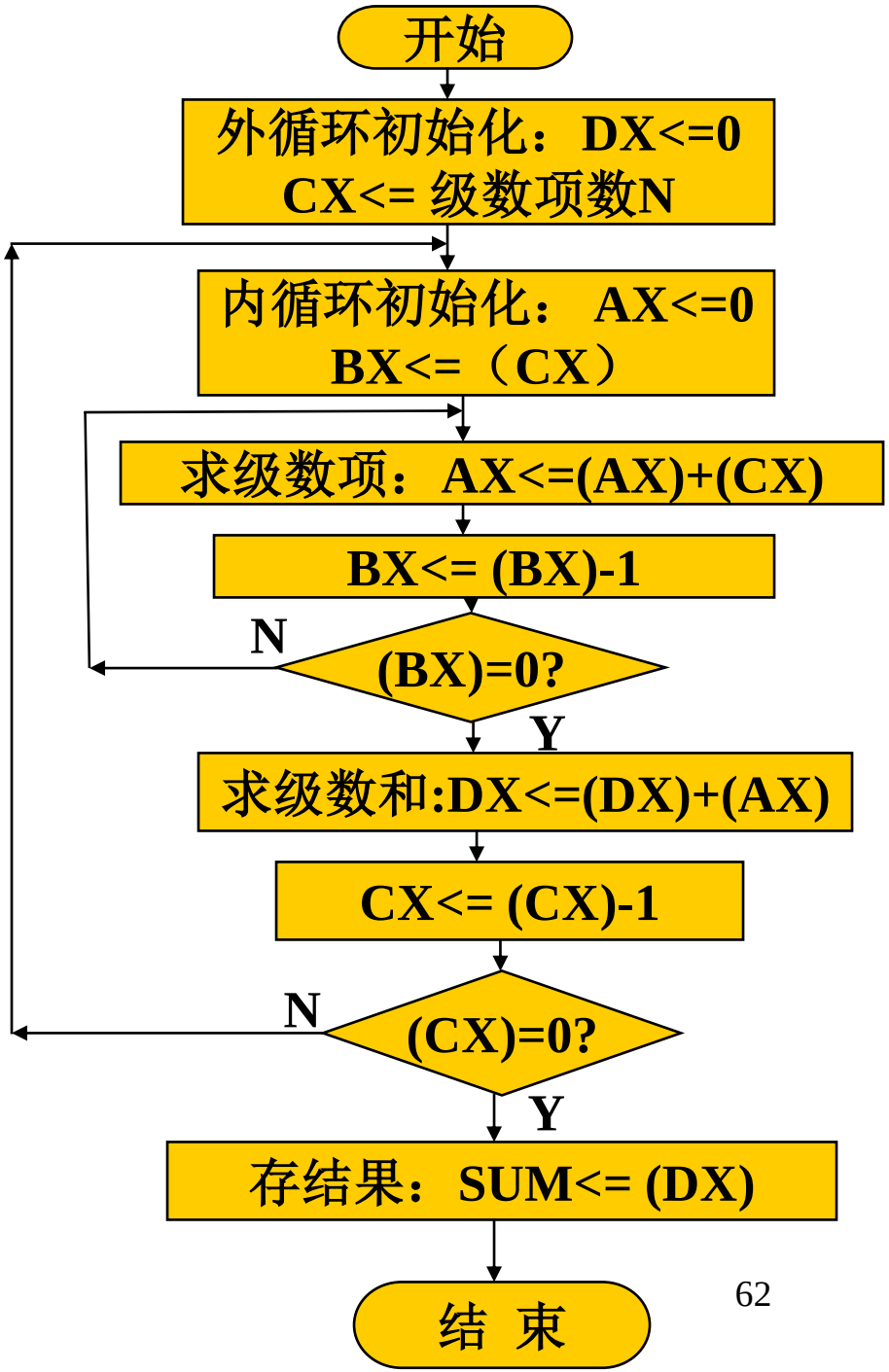
多重循环结构是指循环程序的循环体中又包含了另一个循环

例5.4.6 编写一程序，求级数$1^2+2^2+3^2+\ldots$的前N项和。

对于N^2的计算采用连加的方法，即：

$$N^2 = N \times N = \underbrace{N+N+\ldots+N}_{N}$$

本题程序采用双重循环（顺序:外循环倒序）。内循环计算级数各项的值，外循环计算各级数项之和。



```
DATA   SEGMENT
SUM    DW  ?
N      DB  20
DATA   ENDS
STACK1 SEGMENT PARA STACK
       DW 20H DUP(0)
STACK1 ENDS
CODE   SEGMENT
    ASSUME CS:CODE,DS:DATA,SS:STACK1
START: MOV AX,DATA
    MOV DS,AX
    MOV DX,0
    MOV CX,0
    MOV CL,N ;设置外循环次数
LOP1:  MOV AX,0
    MOV BX,CX;  设置内循环次数
```

```
LOP2: ADD AX,CX；求级数项的值
      DEC BX
      JNZ LOP2；BX计数不为0，继续内循环
      ADD DX,AX ；累加级数项
      LOOP  LOP1；CX计数不为0，继续循环
      MOV SUM,DX；存级数和
      MOV AH,4CH
      INT 21H
CODE  ENDS
       END  START
```