# 第七章 汇编语言简单应用程序设计

本章主要内容：

- 算术运算调整指令及其应用
- 串和表的处理
- 代码转换及其应用

# §7.1 算术运算调整指令及其应用

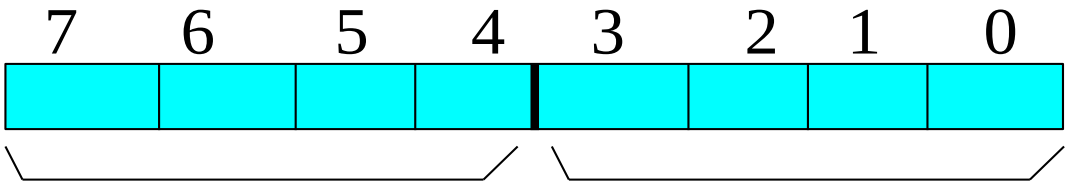
## 一. 十进制数的表示

在计算机中采用BCD码来表示十进制数。BCD码就是使用四位二进制数表示一位十进制数。

在8086/8088系统中，将BCD码分为两种格式：

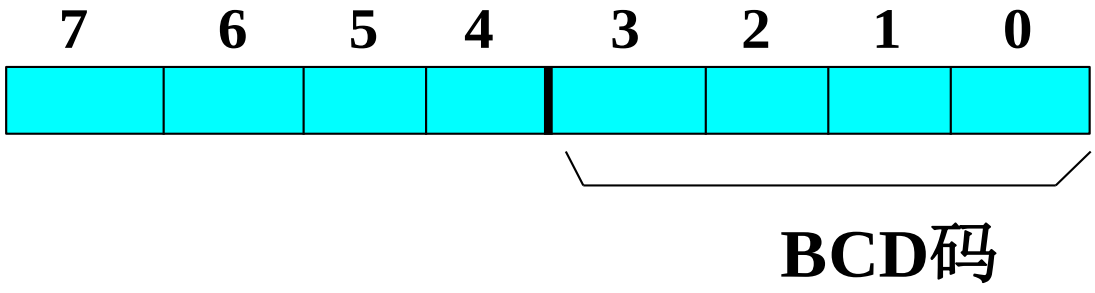
- 组合型（压缩型、装配型、PACKED）
- 非组合型（非压缩型、拆散型、UNPACKED）

组合型：一个字节表示两个BCD码，即两位十进制数。




例如：0010 0011 表示十进制数的23

非组合型：一个字节的低四位表示一个BCD码，而高四位对所表示的十进制数没有影响。常为0000B或0011B。




例如：0000 1001与0011 1001都是十进制数9的非组合型的BCD码

在计算机中实现十进制数(BCD码)的运算有两种方法:

1. 数制转换: 先把十进制数转换为二进制数,然后用计算机中的二进制运算指令进行运算。最后将结果由二进制数转换为十进制数。

2. 使用计算机中的BCD码指令，直接进行十进制数运算。

在计算机内部实现直接对BCD码运算的方法有两种：

（1）指令系统提供专门实现BCD码运算的加、减、乘、除运算指令。

（2）先用二进制数的加、减、乘、除运算指令对BCD码运算，再用BCD码校正指令对结果校正。在8086/8088系统中就是使用这种方法。

## 二、BCD码加减校正指令

8086/8088系统共有六条BCD码校正指令。本节先介绍四条加减校正指令。

### 1、非组合型加法校正指令AAA

AAA指令实现对一位十进制数进行校正。

在AAA指令执行前，应使用ADD或ADC指令完成了BCD码加法，且结果是在AL中。AAA指令对AL中内容进行校正。

校正过程为：

当AL中的低4位>9或者AF=1，则AL<=（AL）+6，AH<=（AH）+1，AL中高4位清0，AF和CF置1。

例如：从键盘输入两个一位数的十进制数，然后相加，结果放在AH和AL中。

```
MOV  AH,1
INT  21H
MOV  BL,AL ;BL中为输入的一位十进制数的ASCII码，低4位为该数的BCD码
MOV  AH,1
INT  21H ; AL中为输入的另一位十进制数的ASCII码
MOV  AH,0
ADD  AL,BL
AAA
```

## 2、组合型加法校正指令DAA

DAA指令实现对二位十进制数进行校正。

在执行DAA指令前，应使用ADD或ADC完成了二位BCD码的加法操作，且加的结果放在AL中。

其校正过程为：

若AL中低4位>9或AF=1，则 AL<=(AL)+6, AF<=1

若AL中高4位>9或CF=1，则AL<=(AL)+60H, CF<=1

例：实现两个4位十进制数的加法4678+2556

```
NUM1  DB  78H,46H
NUM2  DB  56H,25H
SUM    DB  ?,?
          ......
          MOV  AL,NUM1
          ADD  AL,NUM2   ;低字节BCD码相加
          DAA                    ;结果低字节校正
          MOV  SUM,AL
          MOV  AL,NUM1+1
          ADC  AL,NUM2+1;高字节BCD码相加
          DAA                    ;结果高字节校正
          MOV  SUM+1,AL
```

## 3、非组合型减法校正指令AAS

执行AAS指令前，应使用SUB或SBB完成了BCD码的减法运算，且结果放在AL中。

其校正过程为：

若AL中低4位>9或AF=1，则AL<= (AL)-6 , AH<= (AH)-1, 同时将AL中高4位清零，CF和AF置1。

## 4、组合型减法校正指令DAS

执行DAS指令前，应使用SUB或SBB完成了二位BCD码减法运算，且结果放在AL中。

其校正过程为：

* 若AL中低4位>9或AF=1，则AL <=(AL)-6，AF置1；

* 若AL中高4位>9或CF=1，则AL<=(AL)-60H,CF置1。

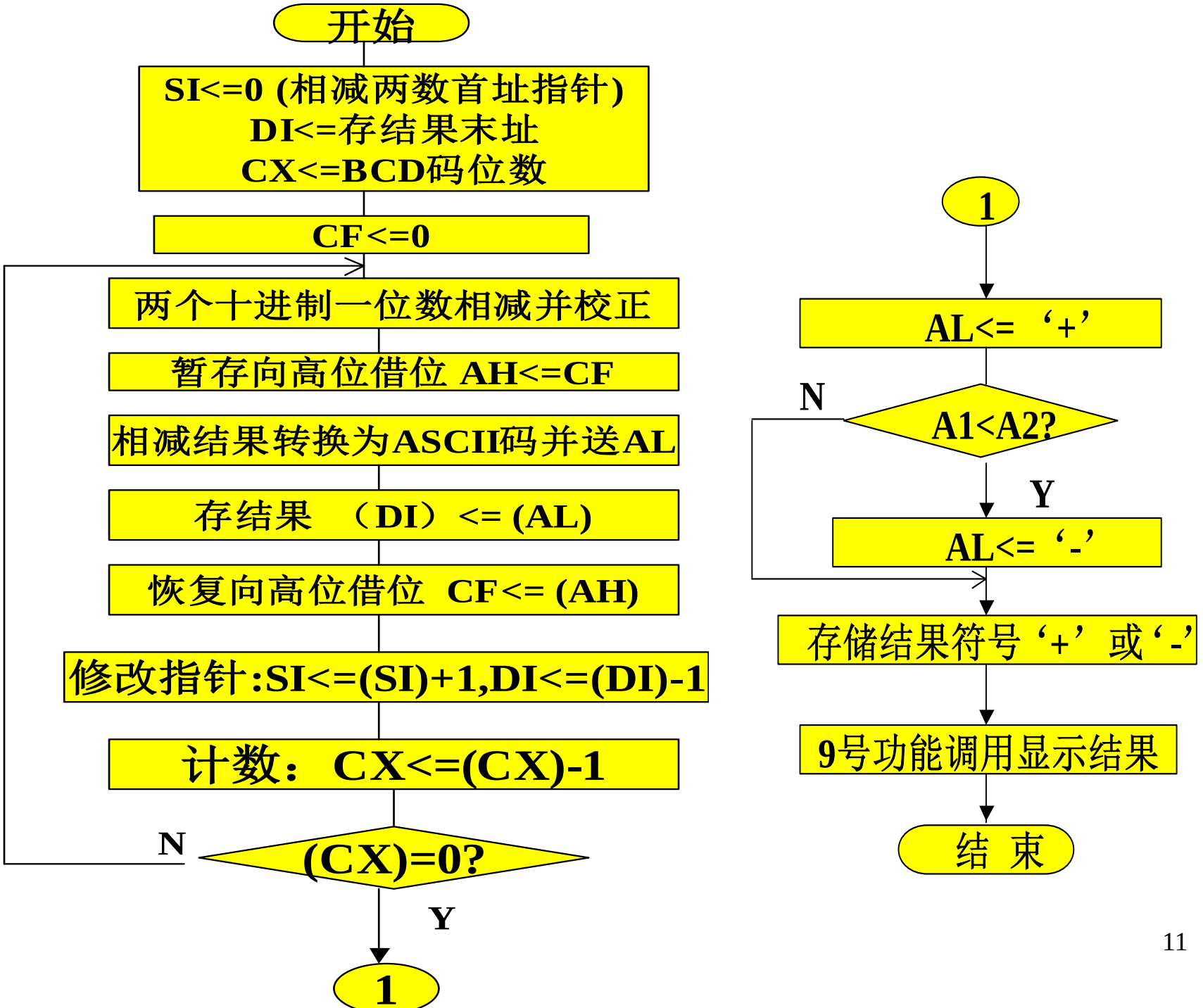
例1 试编制一程序，实现非组合型BCD码减法并显示结果。

设数据段有两个4位十进制数（非组合型BCD码）A1和A2。分别放在以DA1和DA2为首址的存储单元中（低字节放低位，高字节放高位）。

结果存放在以DA3为首址的存储单元中。为了显示方便，结果采用低字节放高位，高字节放低位。

为了表示A1和A2的相对大小，若A1≥ A2，则结果前加‘+’号，否则加‘-’号。

结果的显示使用9号DOS功能调用。


开始
SI<=0 (相减两数首址指针)
DI<=存结果末址
CX<=BCD码位数
CF<=0
两个十进制一位数相减并校正
暂存向高位借位 AH<=CF
相减结果转换为ASCII码并送AL
存结果 （DI）<= (AL)
恢复向高位借位 CF<= (AH)
修改指针:SI<=(SI)+1,DI<=(DI)-1
计数： CX<=(CX)-1
N
(CX)=0?
Y
1
1
AL<= ‘+’
N
A1<A2?
Y
AL<= ‘-’
存储结果符号‘+’或‘-’
9号功能调用显示结果
结 束

```
    TITLE  DECIMAL SUBTRACTION
DATA     SEGMENT
DA1      DB   1,2,3,4
DA2      DB   0,1,2,3
DA3      DB   5 DUP(0),'$'
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  SI,0
         LEA  DI,DA3+4     ;  存结果单元末址送DI
         MOV  CX,4         ;  十进制位数送CX
         CLC
```

```
LOP:  MOV  AL,DA1[SI]
      SBB  AL,DA2[SI]    ;  两数相减
      AAS                ;  校正
      LAHF               ;  暂存向高位的借位
      AND  AL,0FH        ;  转换成ASCII码
      OR   AL,30H
      MOV  [DI],AL       ;  存结果
      SAHF               ;  恢复向高位的借位
      INC  SI
      DEC  DI
      LOOP LOP
      MOV  AL,'+ '
      JNC  NEXT
      MOV  AL,'-'        ;  有向更高位的借位，  存 '-'号
```

```
NEXT: MOV  [DI],AL
      MOV  DX,OFFSET DA3
      MOV  AH,9        ；9号功能调用显示结果
      INT   21H
      MOV  AH,4CH
      INT   21H
CODE ENDS
      END  START
```

# 三. 乘除法指令及调整指令

## 1、无符号数乘法指令MUL

指令格式：**MUL　OPRD**

其中：**OPRD**提供乘法运算的一个操作数，它只能是寄存器或存储器操作数。另一操作数隐含使用**AL**或**AX**寄存器。

运算结果存放在**AX**（字节运算）或**DX：AX**（字乘法）中。

- 字节运算：AX<=(AL) ×(OPRD)
- 字运算 ：DX:AX<= (AX)×(OPRD)

MUL只影响CF和OF标志，其它标志位的值不确定。若结果的AH（字节运算）或DX（字运算）为全0，CF=0、OF=0，否则 CF=1、OF=1。

## 2、带符号数乘法指令IMUL

指令格式：IMUL　OPRD

该指令的功能除了操作数是带符号外，其余与MUL指令相同。

对标志位的影响：若乘积的高半部AH（字节乘法）或DX（字乘法）是低半部的符号扩展(不是有效数值），则CF=0、OF=0 。否则CF=1、OF=1 。

例如，对字节乘法有：

若乘积的(AH)=11111111，且AL最高为1，则表示符号扩展，则CF=0、OF=0；

若乘积的(AH)=00000000，且AL最高为0，则表示符号扩展，则CF=0、OF=0；

若乘积的(AH)=11111110，不是符号扩展，则CF=1、OF=1；

若乘积的(AH)=00000010，不是符号扩展，则CF=1、OF=1。

## 3、无符号数除法指令DIV

**指令格式：DIV OPRD**

其中OPRD是除法运算中的除数，它可以是字节(字节除法）或字（字除法）操作数，只能是寄存器或存储器操作数。

被除数和结果隐含使用以下的寄存器：

- 字节除法：被除数为AX，AL<=商 ，AH<= 余数
- 字除法：被除数为DX和AX，AX<=商, DX<=余数

对标志影响：该指令对标志各位无有效影响。

下面两种情况将产生0型中断，转入除法出错处理：

- （OPRD）=0
- 商>0FFH(字节运算）或商>0FFFFH（字运算）

## 4、带符号数除法指令IDIV

除操作数是带符号数外，该指令的功能与DIV指令相同。

当结果商的值超过-127~+127（字节运算）或-32767~+32767（字运算）范围时，将产生0型中断。

## 5、字节/字扩展指令CBW/CWD

这两条指令主要用于除法指令前，形成双倍长度的被除数。它们都是无操作数指令，隐含使用AX或DX。

指令功能：

CBW：扩展AL中符号位到AH中

CWD：扩展AX中符号位到DX中

两条指令对标志都无影响。

## 6. 非组合型乘法校正指令AAM

执行AAM指令前，必须是用MUL完成了无符号数乘法操作，且结果放在AL中。

调整规则： AH←AL/10(商)，AL←AL/10(余数)

对标志位的影响：根据结果设置PF、SF和ZF，AF、CF和OF等不确定。

## 7. 非组合型除法校正指令AAD

AAD用于除法指令DIV之前，将寄存器AH和AL组成的两位非组合型BCD码，调整成一个在AL中的二进制数。

调整规则： AL←AH×10+AL，AH←0

对标志位的影响：与指令AAM相同。

例2 设数据段有X、Y两变量（无符号数），且$X^Y$不超过一个字的表示范围（65535），试编制一计算$X^Y$的程序。

由于没有乘方指令，因此需要将乘方转换为乘法运算。即：

$$X^Y = \underbrace{X \times X \cdots\cdots \times X}_{Y}$$

```
    TITLE  Mathematical power
DATA   SEGMENT
VARX   DW  5
VARY   DW  6
POWER   DW  ?
DATA   ENDS
STACK1 SEGMENT PARA STACK
       DW  20H DUP(0)
STACK1 ENDS
```

```
CODE   SEGMENT
       ASSUME CS:CODE,DS:DATA,SS:STACK1
MATH:  MOV  AX,DATA
           MOV  DS,AX
           MOV  AX,VARX
            MOV  CX,VARY
            DEC  CX
           JE   EXIT
           MOV  DX,0  ;乘积的高位清0
LOP:    MUL VARX
           LOOP LOP
EXIT:  MOV  POWER,AX;存结果
           MOV  AH,4CH
           INT 21H
CODE  ENDS
            END MATH
```

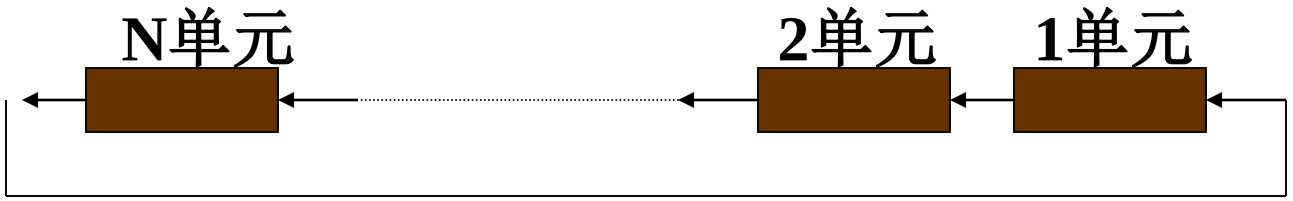
多精度数运算：8086/8088微处理器的每条指令只能处理8位或16位二进制数，其表示的数值范围有限，有时不能满足需要。为了提高运算精度，常用多字节或多字来表示一个完整的数据。

例3 试编写对一个多精度数求补的子程序。

设多精度数的首址放在SI中。数据存储时，低字节放低地址单元，高字节放高地址单元。多精度数的字节数在CL中。程序中，求补采用的是“变反加1”的方法。

```
COMP  PROC
          MOV  CS:RESV,SI    ;暂存多精度数首址
          XOR  CH,CH
          MOV  AL,CL         ;暂存多精度字节数
LOP1: NOT  BYTE PTR [SI]    ;变反
          INC  SI
          LOOP  LOP1
          MOV  SI,CS:RESV   ;恢复多精度数首址
          MOV  CL,AL        ;恢复字节数
          STC                ;  置CF为1
LOP2: ADC  BYTE PTR [SI],0    ;完成最低位加1，
                                ;其它方式？
          INC  SI
          LOOP  LOP2
          RET
RESV  DW  0
COMP  ENDP
```

例4 编制一个多精度数的循环左移子程序。



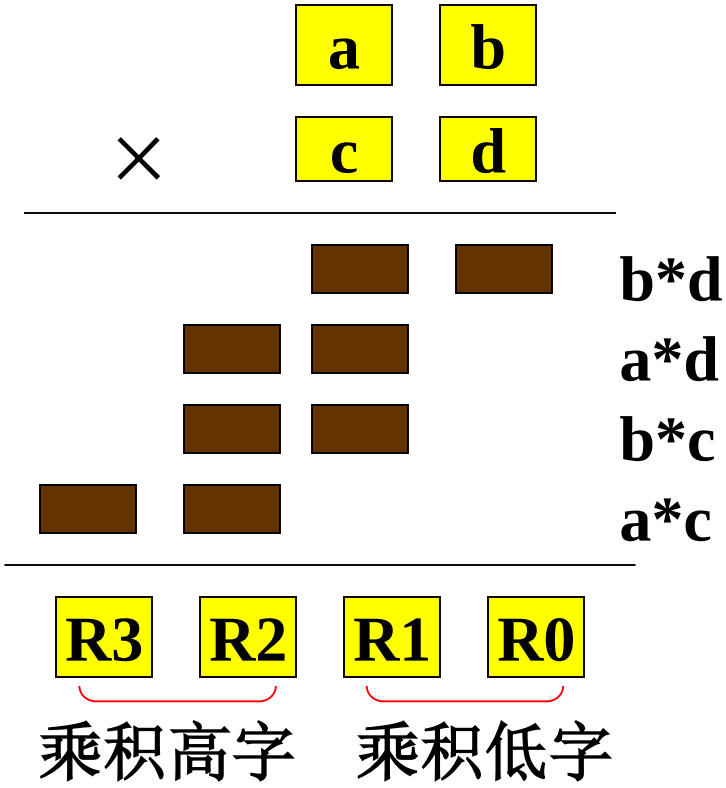

为了将最后一个单元（N单元）的最高位移入第一个单元（1单元）的最低位，先测试N单元的最高位，将测试的结果记录在CF中，在1单元做循环左移时，将其移入最低位。

设多精度数的首址在SI中，字节数在CL中。

```
ROTATE  PROC
        XOR  CH,CH
        MOV  BX,CX
        TEST   BYTE PTR [BX-1][SI],80H
         ;测试最后单元的最高位，并清CF
        JNS  LOP              ;最高位=0,保持CF=0
        STC                   ;最高位=1,CF<=1
LOP:  RCL   BYTE PTR [SI],1  ;循环左移一位
        INC  SI
        LOOP  LOP
        RET
ROTATE  ENDP
```

## 例5 用乘法指令实现32位（双字长）二进制数的乘法

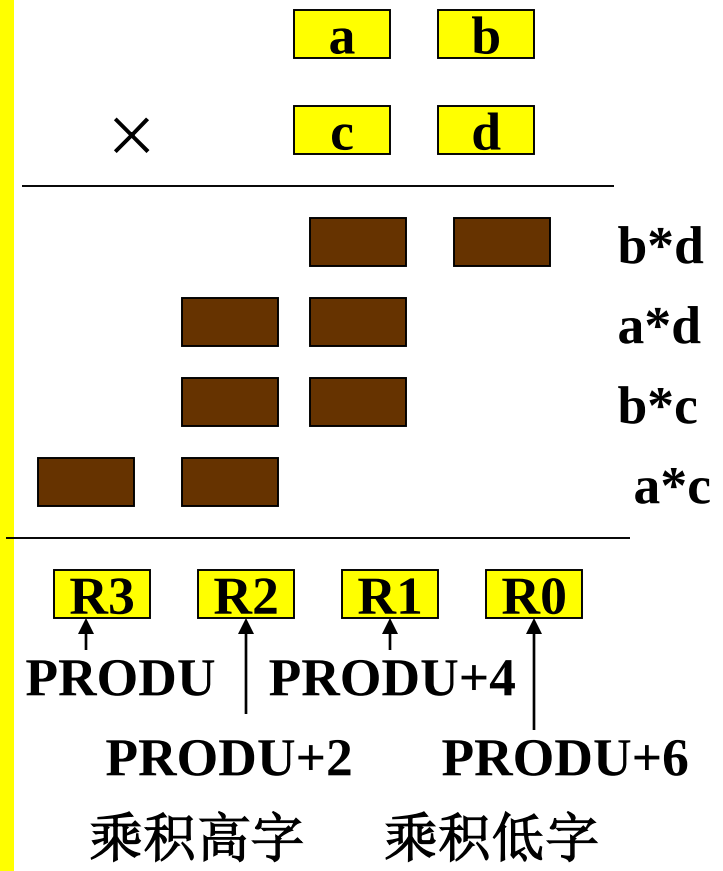
设被乘数两个字用a、b表示，乘数两个字用c、d表示。乘积则为64位（4个字长）。由于乘法指令只能完成单字乘法，对于双字乘法的处理过程如下图所示。




设乘数和乘积存放为:低字存于高地址单元,高字存于低地址单元。

```
 TITLE  32-BIT MULTIPLICATION
DATA   SEGMENT
NUM1   DW  1220H,48A2H
NUM2   DW  2398H,0AE41H
PRODU  DW  4 DUP(0)
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  DX,DX
       MOV  AX,NUM2+2
       MUL  NUM1+2   ;完成b*d
       MOV  PRODU+6,AX;  存R0
       MOV  PRODU+4,DX;  暂存R1的部分
```

```
    MOV  AX,NUM2+2
    MUL  NUM1            ;完成a*d
    ADD  PRODU+4,AX;
    ADC  PRODU+2,DX
    ADC  PRODU,0 ;将进位送PRODU中
    MOV  AX,NUM2
    MUL  NUM1+2          ;完成b*c
    ADD  PRODU+4,AX;  完成R1存放
    ADC  PRODU+2,DX
    ADC  PRODU,0
    MOV  AX,NUM2
    MUL  NUM1            ;完成a*c
    ADD  PRODU+2,AX ;  完成R2存放
    ADC  PRODU,DX     ;  完成R3存放
    MOV  AH,4CH
    INT 21H
CODE   ENDS
    END START
```



# §7.2 串和表的处理

## 一. 串操作指令

“串”是指存储器中的一段连续的字单元或字节单元。这些单元中存放的内容可以是字符或数据。

“串操作”就是对这些单元进行某种相同的操作。比如将一段数据从一个存储区传送到另一个存储区。

串操作指令具有以下共有特点：

（1）串操作指令使用专用的寻址方式：源操作数地址由DS:[SI]提供，目的操作数由ES:[DI]提供。

（2）串操作指令每次只处理串中的一个字或一个字节单元，并自动修改SI和(或)DI，使其指向下一个字或字节单元。

（3）当标志位DF=0时，SI和DI的修改方向为递增，即加2（字操作）或加1（字节操作）。当DF=1时，SI和DI的修改为递减，即减2或减1。

## 1. 取串指令

指令格式：　LODS　源串

功能：　将源串中的一个字（或字节）内容送入AX（或AL）中，并根据DF修改SI。

由于源串是由SI指定，如果程序中在执行该指令时已经明确是字或字节，则可以用无操作数指令LODSB（字节操作）或LODSW（字操作）替代。

该指令执行后对标志无影响。

## 2. 存串指令

指令格式：　**STOS　目的串**

功能：将**AX**(或**AL**)中的内容送入目的串中的一个字单元(或字节单元)，并根据DF修改DI。

同样，指令可以用无操作数指令**STOSB**或**STOSW**替代。

该指令对标志无影响。

## 3. 串传送指令

指令格式：**MOVS 目的串，源串**

功能：将由SI指向的源串的一个字(或字节)传送到DI所指向的目的串中。并根据DF修改SI和DI。

同样，指令可以用无操作数指令MOVSB或MOVSW替代。

指令对标志无影响。

## 4. 串比较指令

指令格式：　CMPS　源串，目的串

功能：　将源串中的一个字(或字节)减去目的串中的一个字(或字节)，结果不回送。但将影响标志寄存器。同时，将根据DF修改SI和DI。

同样，指令可以用无操作数指令CMPSB或CMPSW替代。

## 5、串搜索指令

指令格式：　SCAS　目的串

功能：　在目的串中查找AX或AL指定的内容。

查找的方法：用AX(或AL)的内容减去目的串中的一个字(或字节)，相减的结果反映在标志寄存器中。每查找一次，将按照DF修改DI。

同样，指令可以用无操作数指令SCASB或SCASW替代。

## 6、重复前缀指令

指令格式：　**REP**

为了方便对若干个字或字节进行多次同样的操作，可在上述各种指令的前面加上**REP**指令。

重复操作的次数由**CX**控制，每执行一次串操作指令，**CX**内容减**1**，直到**CX**内容为**0**。

例如：**REP MOVSB**

设在执行该指令前,DF=0,(SI)=0020H,(DI)=100H,(CX)=0030H。

执行该指令将使数据段中从0020H开始的30H个字节传送到附加段从0100H开始的存储区。

如果改成不使用串操作指令，则它相当于下面的程序段：

```
LOP:MOV  AL,[SI]
    MOV  ES:[DI],AL
    INC  SI
    INC  DI
    LOOP LOP
```

另外还有两条重复前缀指令：

**REPE/REPZ**

**重复执行串操作指令的条件是：（CX）≠0和ZF=1**

**REPNE/REPNZ**

**重复执行串操作指令的条件是：（CX）≠0和ZF=0**

**由于这两条指令的执行要由标志位ZF来控制结束，而LODS、STOS和MOVS三条指令不影响标志，因此不适合与这些指令配合使用。**

# 二. 串操作指令应用举例

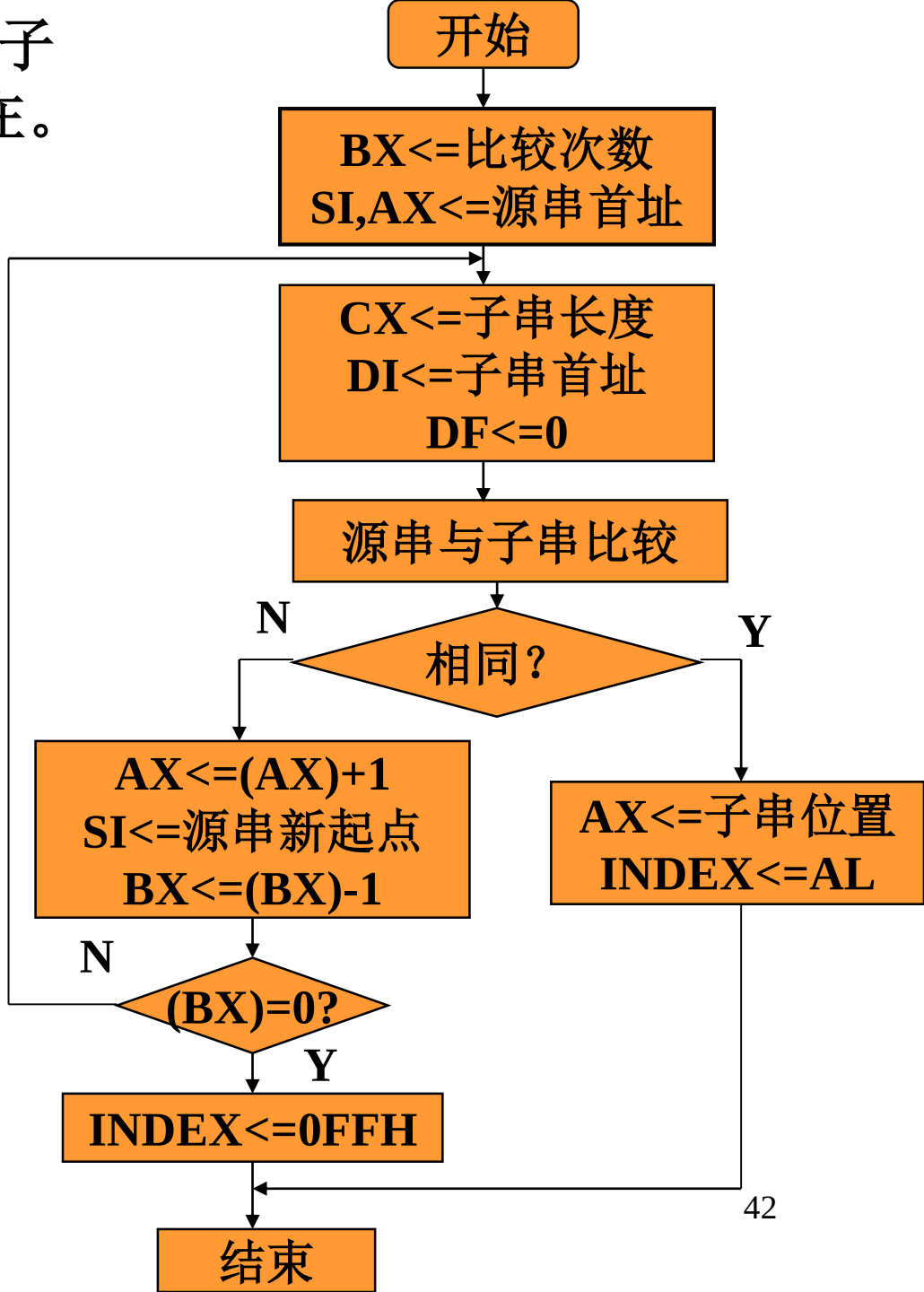
例1 试编制一程序，在TXTBUF字符串中查找STRING变量指定的字符。若查到，则把该字符所在位置（1～n）送入INDEX单元中。若未查到，则把0FFH送INDEX单元中。

```
DATA     SEGMENT
TXTBUF  DB 'ABCDEFGHIJKLMNOP'
CUNT      EQU $-TXTBUF
STRING    DB 'G'
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   ES,AX
       MOV DI,OFFSET TXTBUF;取目的串首址
       MOV CX,CUNT
       MOV AL,STRING
       CLD
      REPNE SCASB;查找字符
      MOV BL,0FFH
      JNE END0   ;未查到，ZF=0，转移
       SUB DI,OFFSET TXTBUF;DI指向了后一个字符
       MOV BX,DI
END0:MOV INDEX,BL
       MOV AH,4CH
       INT 21H
CODE   ENDS
 END START
```

例2 试编写一程序，确定某子字符串是否在另一字符串中存在。若在，则记录其所在起始位置。若不在则设置标志0FFH。

该例是上例的更一般情况。

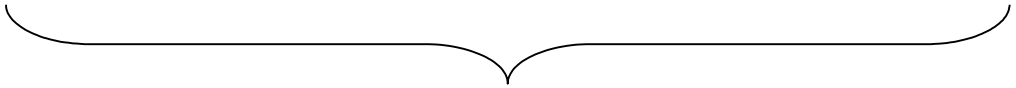


最大比较次数=（源串长度-子串长度）+1

第1次比较：

A B C D E F G H I J K L M N O P

E F



比较次数-1

第2次比较：

A B C D E F G H I J K L M N O P

E F

第3次比较：

A B C D E F G H I J K L M N O P

E F

```
DATA   SEGMENT
TXTBUF DB 'ABCDEFGHIJKLMNOP'
CUNT1  EQU $-TXTBUF
STRING DB 'EF'
CUNT2  EQU $-STRING ；子串长度
INDEX  DB 0FFH
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 ES,AX
       MOV BX,CUNT1-CUNT2+1;最大比较次数
       MOV SI,OFFSET TXTBUF;取源串首址
       MOV AX,SI ； 信息保护，SI要变化
```

```
LOP:   LEA DI,STRING;取子串首址
       MOV CX,CUNT2
       CLD ;SI,DI递增
       REPZ CMPSB; 比较(SI和DI同时？ )
       JZ  MATCH   ;（CX)=0且ZF=1,相同,转MATCH
       INC AX    ; 未匹配，比较位置往后移一位
       MOV SI,AX
       DEC BX  ;  比较次数计数
       JNZ LOP
       JMP EXIT
MATCH:  SUB AX,OFFSET TXTBUF;位置序号从0开始
       INC  AL                          ;位置序号从1开始
       MOV INDEX,AL
EXIT:   MOV AH,4CH(?,INDEX初始化的值为0FFH)
       INT 21H
CODE   ENDS
       END START
```

# 三. 表的排序与查找

排序是指将一组没有规律的无序数据，排列成有序数据。
查找是从一组数据中搜索指定的数据。

排序和查找一般都是在表中进行的。数据表是指一组连续存放在存储器中的数据。

数据表分有序表和无序表。有序表指各数据项在存储器中按顺序（递增或递减）排列。否则称为无序表。

为了缩短数据查找时间，一般应先将无序表排列成有序表后再查找。

## 1. 气泡排序算法

设有一组无序数据为$b_1$、$b_2$、.....$b_{n-1}$、$b_n$，要求以递减顺序排列。气泡排序法的处理过程如下：

（1）将表中第一个数$b_1$与第二数$b_2$比较，若b1<b2，则交换两数，否则不交换。然后将b2与b3比较，这样重复比较，直至最后两个数。这称为一个循环。

（2）重复执行上述步骤，直至一次循环中没有数据交换为止。

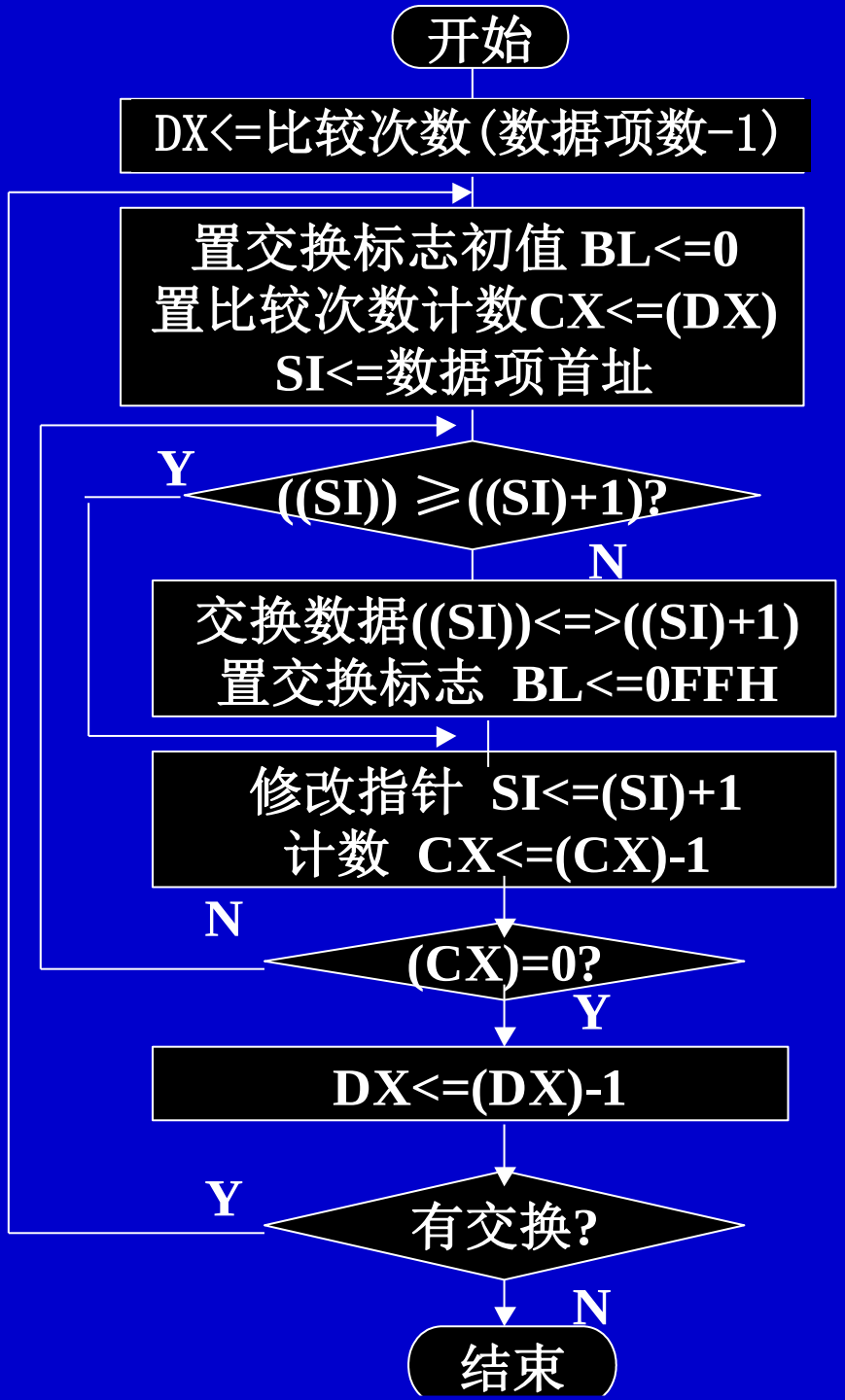
执行完第一次比较循环，将使得最小一个数移动到最后，即一个气泡上浮。

以后每次循环比较将使一个较小的数上浮。并且比较的次数将依次递减。这样直至整个表排列成一个有序表为止。

如果采用递增方式进行排序，其处理方法与上述递减排列方法类似，只需将交换条件变为前面的数大于后面的数。

例如有10个数，经过5次循环比较后，第6次循环比较时已没有数据交换，表明已排列完成。为此，设置一个标识，以确定每次循环比较中是否有数据交换。

| 遍数 / 数据 | | 一 | 二 | 三 | 四 | 五 | 六 |
|---|---|---|---|---|---|---|---|
| b10 | 42 | -32 | -32 | -32 | -32 | -32 | -32 |
| b9 | -32 | 42 | -20 | -20 | -20 | -20 | -20 |
| b8 | 62 | -20 | 42 | -6 | -6 | -6 | -6 |
| b7 | -6 | 62 | -6 | 42 | 3 | 3 | 3 |
| b6 | 120 | -6 | 62 | 3 | 42 | 9 | 9 |
| b5 | 9 | 120 | 3 | 62 | 9 | 42 | 42 |
| b4 | 116 | 9 | 120 | 9 | 62 | 62 | 62 |
| b3 | -20 | 116 | 9 | 120 | 80 | 80 | 80 |
| b2 | 3 | 3 | 116 | 80 | 120 | 116 | 116 |
| b1 | 80 | 80 | 80 | 116 | 116 | 120 | 120 |


开始
DX<=比较次数(数据项数-1)
置交换标志初值 BL<=0
置比较次数计数CX<=(DX)
SI<=数据项首址
Y
((SI)) ≥((SI)+1)?
N
交换数据((SI))<=>((SI)+1)
置交换标志 BL<=0FFH
修改指针 SI<=(SI)+1
计数 CX<=(CX)-1
N
(CX)=0?
Y
DX<=(DX)-1
Y
有交换?
N
结束

```
 TITLE BUBBLE SORT
DATA     SEGMENT
DA       DB 80,3,-20,116,9,120,-6,62,-32,42
COUNT    EQU $-DA
DATA     ENDS
STACK1 SEGMENT PARA STACK
         DW 20H DUP(0)
STACK1 ENDS
COSEG    SEGMENT
       ASSUME CS:COSEG,DS:DATA,SS:STACK1
SORT:    MOV  AX,DATA
         MOV  DS,AX
         MOV  DX,COUNT-1  ;置比较次数初值
SORT1:   MOV  BL,0        ;置交换标志初值
         MOV  CX,DX      ;置内循环比较次数
         MOV  SI,0       ;置数据指针初值
```

```
SORT2:    MOV  AL,DA[SI]
          CMP  AL,DA[SI+1] ;比较
          JGE  NOXCHG      ;大于等于则转不交换
          XCHG AL,DA[SI+1] ;交换
          MOV  DA[SI],AL
          MOV  BL,0FFH   ;置已交换标志
NOXCHG:INC  SI         ;修改地址
          LOOP SORT2
          DEC  DX        ;修改比较次数
          CMP  BL,0      ;检查交换标志
          JNE  SORT1     ;有交换,继续
          MOV  AH,4CH
          INT  21H
COSEG     ENDS
          END  SORT
```

## 2. 二分法查找算法

二分法查找算法是对有序表进行的查找方法。如果是一个无序表，则必须先将其排列成一个有序表。

算法要点：将整个数据表对分成两个部分，判断所查找的数据属于哪个部分。再将所属部分对分成两个区域，并判断所查找数据属于哪个区域。如此重复操作，直至找到数据或区域长度<1。

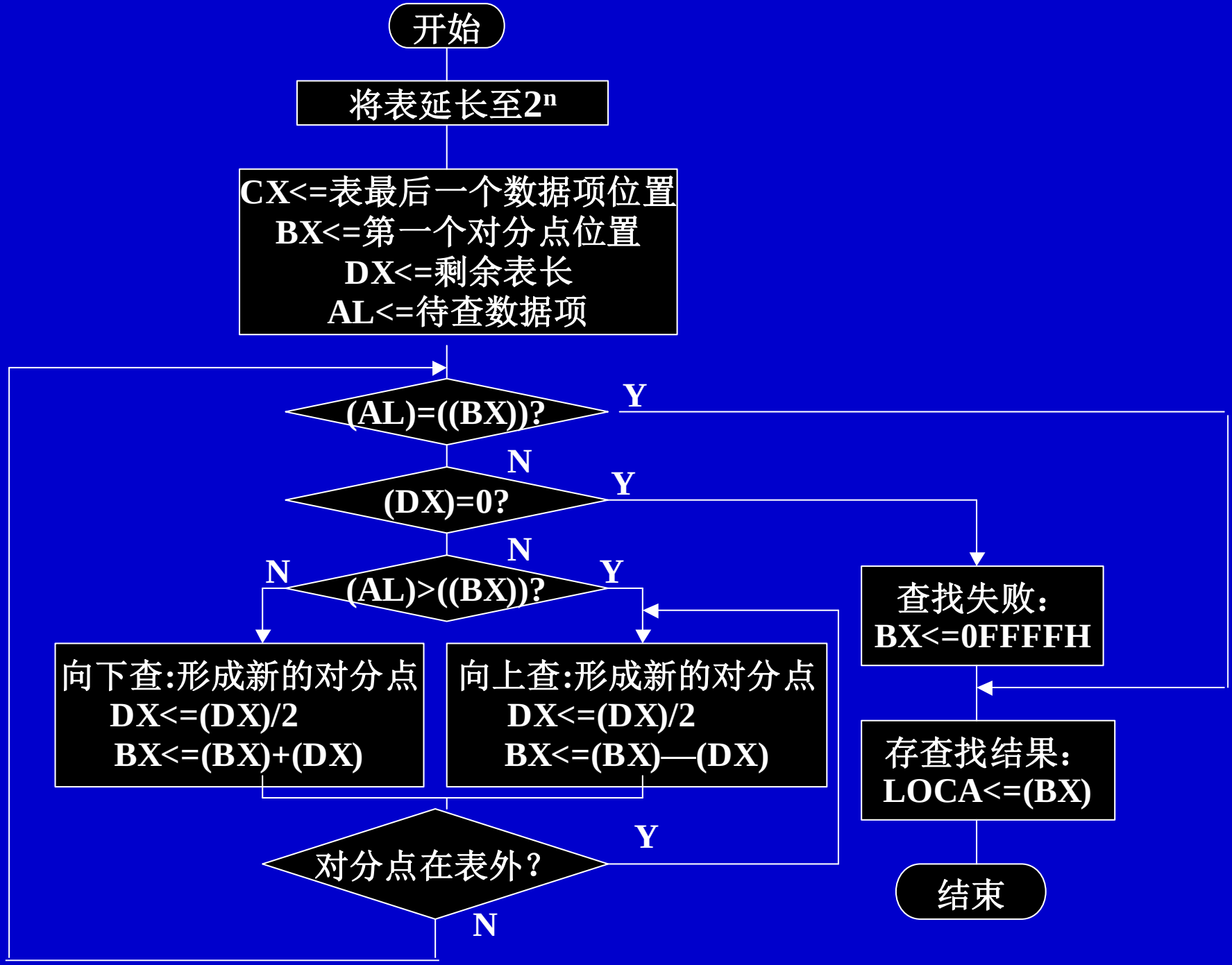
二分法查找算法与顺序查找算法相比，可以减少查找次数，特别是对数据表很大的查找特别明显。对于数据项为N的数据表，二分法查找算法的最多查找次数为：$Log_2N$。

例如当N=1024时，采用顺序查找的平均查找次数为1024/2=512，而二分查找算法的最多查找次数为$Log_2 2^{10}=10$ 。

为了达到对数据表的合理对分，数据表的长度应为$2^n$。为此，对不满足该长度要求的数据表，将其延长至$2^n$。但是，在进行对分时不能使对分的中点落入延长部分。

例　试用二分查找算法在有序表中查找指定的数据。设数据表已经是递减排列。

使用BX作对分中点地址指针，DX存放对分后的查找范围长度。（DX）/2就是下次查找范围长度。若(DX)=0则查找结束。如果找到数据，则将其位置（1~n）送LOCA单元，如果未找到，则送全“1”到LOCA单元。


开始
将表延长至2n
CX<=表最后一个数据项位置
BX<=第一个对分点位置
DX<=剩余表长
AL<=待查数据项
(AL)=((BX))?
Y
N
(DX)=0?
Y
N
N
(AL)>((BX))?
Y
查找失败:
BX<=0FFFFH
向下查:形成新的对分点
DX<=(DX)/2
BX<=(BX)+(DX)
向上查:形成新的对分点
DX<=(DX)/2
BX<=(BX)—(DX)
存查找结果:
LOCA<=(BX)
对分点在表外?
Y
N
结束

```
      TITLE  BINARY  SEARCH
DATA     SEGMENT
TABLE    DB CUNT-1,7FH,7AH,79H,73H,70H,6EH,6BH,6AH
         DB 5DH,5CH,5AH,59H,55H,54H,50H,4DH,4CH,4AH
         DB 49H,48H,44H,41H,40H,3EH,3DH,3AH,39H,36H
         DB 35H,32H,30H,2CH,25H,23H,1FH,19H,14H,00H
CUNT     EQU $-TABLE
DA0      DB  32H  ;被查找数据
LOCA     DW  ?
DATA     ENDS
STACK1 SEGMENT PARA STACK
         DW 20H DUP(0)
STACK1 ENDS
COSEG    SEGMENT
         ASSUME CS:COSEG,DS:DATA,SS:STACK1
BINARY:MOV AX,DATA
         MOV DS,AX
```

```
        MOV BX,1
        MOV AX,CUNT-1 ;CUNT-1为最后一个数的指针?
LENG:   CMP BX,AX ;此循环实现将表延长为2^n
        JAE SEARCH
        SAL BX,1  ;BX此时角色还不是对分点?
        JMP LENG
SEARCH:MOV CX,CUNT-1;CX<=最后一个数据项位置
        SHR BX,1     ;?,第一个对分点
        MOV DX,BX    ;DX<=对分后表长
        MOV AL,DA0   ;AL<=待查找数据
COMP:  CMP AL,TABLE[BX]
       JE END0  ;查到,转结束
       PUSHF    ;未查到，暂存标志
       CMP DX,0 ;表已查完?
       JE   NOFUND;是,转结束
       SHR DX,1 ;?,折半计算
       POPF
```

```
            JG  UP    ;大于对分点数据
            ADD BX,DX ; 向下查，计算出下半部的对分点
             JMP NEXT
UP:       SUB BX,DX ;  向上查，计算出上半部的对分点
NEXT:    CMP BX,CX ;检查对分点是否在表外（比较地址）
            JB COMP  ;在表内,转继续查找
            JMP   UP ;在表外,向上找对分点
NOFUND:MOV BX,0FFFFH;未查到,BX置常数
             POPF ;该操作的结果虽无用,但能保持堆栈操作的正确
END0:    MOV LOCA,BX ;存结果
            MOV AH,4CH
            INT 21H
COSEG  ENDS
           END BINARY
```

# §7.3 代码转换及其应用

代码转换是在计算机程序设计中经常碰到的问题。如二进制数与十进制数的转换，ASCII码表示的各种进制数与二进制数之间的转换等等。

代码转换可以用硬件快速实现，但更常用的方法还是用软件的方法来实现。用软件处理代码转换的方法通常有以下两种方法：

1.查表法：这种方法主要用于代码之间的转换关系比较复杂，但码元的数量必须是有限的情况。

2.直接转换法：依据转换规律，采用一定的算术运算或逻辑运算进行转换。

# 一．十六进制数的ASCII码与二进制数之间的转换

在编制源程序时，常用十六进制数。而从键盘输入时，在计算机中得到的是每个数符的**ASCII码**。因此需要将这些ASCII码表示的数转换为二进制数。

十六进制的每个数符所对应的ASCII码如下表所示。

从表中可以看出以下规律：

1、对于数字符0～9，其ASCII码的低4位就等于对应的二进制值。转换时，只需要将ASCII码的高4位去掉，就是其对应的二进制数。而在二进制数前加上0011B，就是ASCII码。

2、对于数符A～F，各个ASCII码值与对应的二进制数值之差都为37H。

3、对于数符a～f，各个ASCII码值与对应的二进制数值之差都为57H。

| 16 进制数符 | ASCII 码 | 二进制数 |
|---|---|---|
| 0 | 30H | 0000 |
| 1 | 31H | 0001 |
| 2 | 32H | 0010 |
| 3 | 33H | 0011 |
| 4 | 34H | 0100 |
| 5 | 35H | 0101 |
| 6 | 36H | 0110 |
| 7 | 37H | 0111 |
| 8 | 38H | 1000 |
| 9 | 39H | 1001 |
| A | 41H | 1010 |
| B | 42H | 1011 |
| C | 43H | 1100 |
| D | 44H | 1101 |
| E | 45H | 1110 |
| F | 46H | 1111 |
| a | 61H | 1010 |
| b | 62H | 1011 |
| c | 63H | 1100 |
| d | 64H | 1101 |
| e | 65H | 1110 |
| f | 66H | 1111 |

例1 将4位十六进制数的ASCII码分别转换为对应的4位二进制数，然后将它们组合成一个16位长的二进制数。

例如，十六进制数1A2CH，它的ASCII码的表示形式为：31H,41H,32H,43H，而对应的二进制数为0001101000101100B

对于数字0～9的ASCII码，将其高4位二进制数去掉，就是对应的二进制数，对应字母A～F或 a～f ，将其ASCII码减去7，则其低4位与对应的4位二进制数相同。再去掉高4位即可。

```
 TITLE  CONVERT HEX TO BINARY
DATA     SEGMENT
PROMPT DB 'INPUT HEXADECIMAL (4DIGIT):$'
HEX       DB 5,0,5DUP(0)
BIN       DW ?
ERR       DB 0AH,0DH,'ERROR ! NO-HEXADECIMAL ! $'
DATA     ENDS
STACK1 SEGMENT PARA STACK
            DW 20H DUP(0)
STACK1 ENDS
COSEG   SEGMENT
           ASSUME CS:COSEG,DS:DATA,SS:STACK1
HEXBIN:MOV  AX,DATA
            MOV  DS,AX
            LEA    DX,PROMPT;显示提示信息
            MOV  AH,09H
            INT 21H
```

```
        LEA   DX,HEX
        MOV AH,0AH  ;输入4位十六进制数
        INT 21H
        LEA SI,HEX+2 ;取输入字符首地址
        MOV CH,HEX+1        ;取字符数
        MOV AX,0
CONV:  MOV BL,[SI] ;代码转换
        CMP  BL,'0'
        JB      ERROR；<0，出错
        CMP BL,'9'
      JBE  BIN1  ;是0-9，转移
        CALL HEX1 ;是字母符，调用子程序
        JC  ERROR ;是错误的字符
BIN1:  AND BL,0FH
        MOV CL,4
        SAL AX,CL;空出低4位装新转换的值
        OR  AL,BL;  类似于加？
        INC SI
```

```
            DEC CH ；转换字符计数
            JNE CONV
            MOV BIN,AX ;存结果
             JMP END0
ERROR: MOV BIN,0
            LEA DX,ERR
            MOV AH,09H
            INT 21H
END0:    MOV AH,4CH
            INT 21H
;判断大小写子程序
HEX1     PROC
            CMP  BL,'F'
            JA  SMALL
            CMP  BL,'A' ;
            JB  ERROR1；<‘A’,出错
            JMP OUT1；是十六进制数符A-F
SMALL:CMP BL,‘a’ ;检查是否为a~f？
```

```
            JB  ERROR1；<‘a’，出错
            CMP BL,'f'
            JA  ERROR1；>’f’,出错
OUT1:    SUB BL,07H ;
            CLC  ;无错误CF<=0
            RET
ERROR1:STC;设置出错标志CF<=1
            RET
HEX1     ENDP
COSEG  ENDS
            END HEXBIN
```

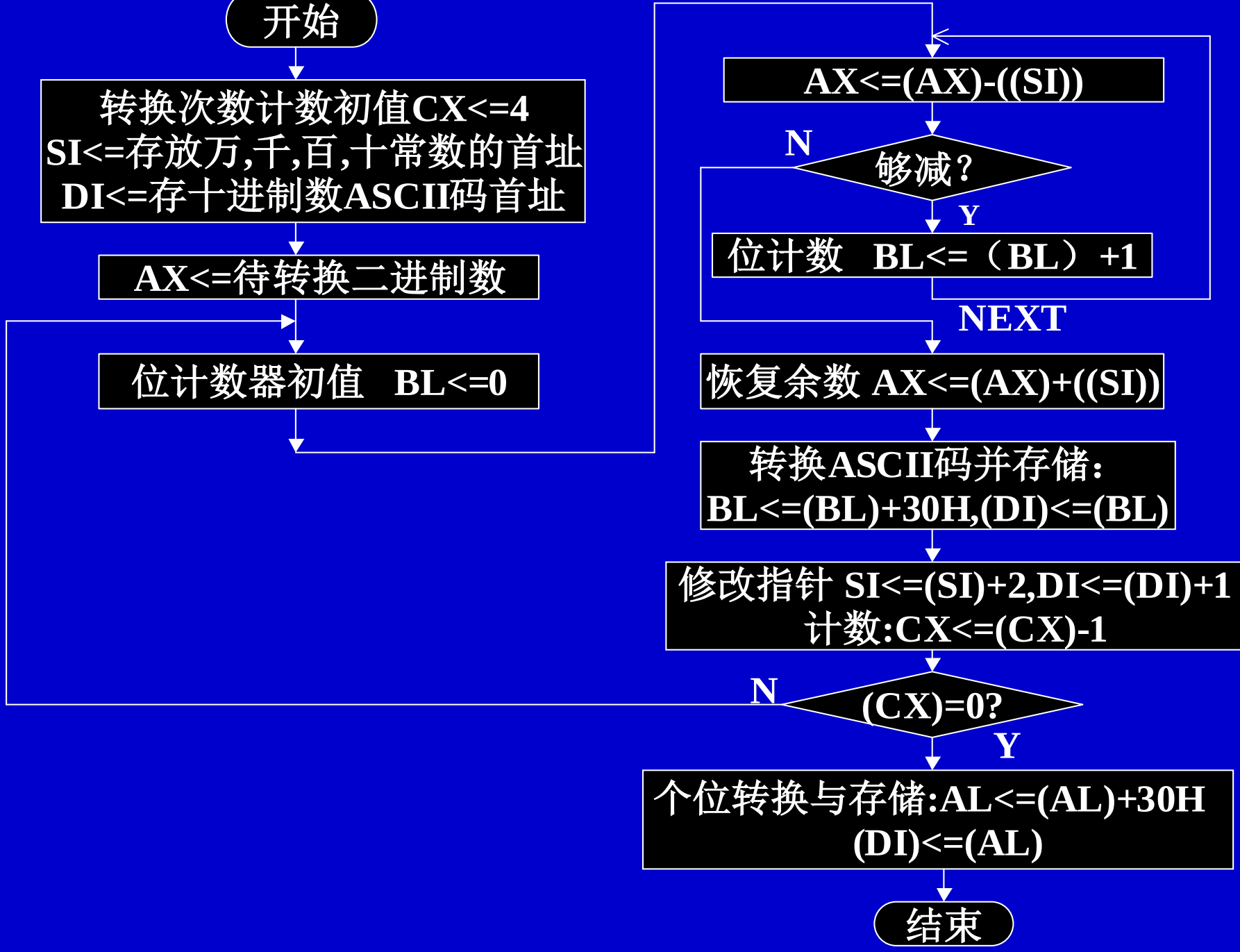
# 二. 二进制数与十进制数之间的转换

例1 将16位无符号二进制数转换为用ASCII码表示的十进制数

算法分析：

1. 16位无符号二进制数表示的十进制数范围为0~65535，需要分别求出万位、千位、百位、十位和个位的值。

2. 从16位二进制数中能够减10000的次数就是万位的值，剩下的数再用1000去减，这样依次进行下去，直至个位。


开始
转换次数计数初值CX<=4
SI<=存放万,千,百,十常数的首址
DI<=存十进制数ASCII码首址
AX<=待转换二进制数
位计数器初值 BL<=0
AX<=(AX)-((SI))
N
够减?
Y
位计数 BL<=（BL）+1
NEXT
恢复余数 AX<=(AX)+((SI))
转换ASCII码并存储:
BL<=(BL)+30H,(DI)<=(BL)
修改指针 SI<=(SI)+2,DI<=(DI)+1
计数:CX<=(CX)-1
N
(CX)=0?
Y
个位转换与存储:AL<=(AL)+30H
(DI)<=(AL)
结束

```
DATA      SEGMENT
BIN1      DW 0110110110101001B
CONST     DW 10000,1000,100,10
DEC5      DB  5 DUP(0)
DATA      ENDS
STACK1  SEGMENT PARA STACK
          DW  20H DUP(0)
STACK1  ENDS
CODE      SEGMENT
          ASSUME CS:CODE,DS:DATA,SS:STACK1
BINDEC: MOV  AX,DATA
          MOV  DS,AX
          MOV  CX,4  ;转换后十进制位数-1
          LEA    SI,CONST ;常数首址
          LEA    DI,DEC5 ;取存十进制数ASCII码的首址
          MOV  AX,BIN1 ;取待转换数
CONV3:  MOV   BL,0  ;位计数器初始化
```

```
LOP1:   SUB   AX,[SI];试减
        JC      NEXT  ;不够减，转NEXT
        INC     BL    ;够减，计数
        JMP     LOP1
NEXT:   ADD   AX,[SI];不够减，恢复余数
        OR      BL,30H ;形成ASCII码
        MOV   [DI],BL;存结果
        INC     SI  ; 取下一个常数地址
        INC     SI
        INC     DI ;修改存结果指针
        LOOP  CONV3  ;继续
       OR    AL,30H ;形成个位的ASCII码
       MOV [DI],AL ;存个位数
       MOV AH,4CH
       INT 21H
CODE   ENDS
       END  BINDEC
```

例2 将16位二进制数转换为非组合型BCD码表示的十进制数。

本例中将16位二进制数转换为5位十进制数的方法是：被转换二进制数除以10，所得余数为十进制数的个位。其商再除以10，所得余数为十位，如此反复，直到商为0。

程序中,用循环实现将二进制数转换为十进制数。十进制数的高位存放在高地址单元，低位存放在低地址单元。

```
DATA    SEGMENT
BIN16    DW  365AH      ;待转换的二进制数
DEC5   DB  5 DUP(0)   ;存转换后的十进制数BCD码
DATA    ENDS
STACK1  SEGMENT PARA  STACK
        DW  20H DUP(0)
STACK1  ENDS
COSEG   SEGMENT
        ASSUME CS:COSEG,DS:DATA,SS:STACK1
MAIN:MOV  AX,DATA
        MOV  DS,AX
        LEA  DI,DEC5 ;取存十进制数个位的单元地址
        MOV  AX,BIN16
        MOV  BX,10
LOP:XOR  DX,DX
        DIV  BX       ;二进制数除以10，余数在DX中
        MOV  [DI],DL  ;存1位十进制数
        INC  DI
```

```
        CMP  AX,0        ;商是否为0?
        JNE  LOP    ;否,则继续转换
        MOV  AH,4CH
        INT  21H
COSEG   ENDS
        END  MAIN
```

# 三. 十六进制数与BCD码的转换

从键盘上输入一个十六进制数字的ASCII码串，将它转换为十进制数的BCD码表示形式。

其转换过程通常分为两个步骤：先把十六进制数ASCII码转换为二进制数，再将二进制数转换为BCD码。这两个步骤我们在前面的例题中已经学习了。下面学习另一种将二进制数转换为BCD码的方法。

设有4位二进制数为$a_3a_2a_1a_0$，每个数符$a_i$的取值为0或1。该二进制数对应的十进制数可用以下公式计算：

$$(((0+a_3)\times 2+a_2)\times 2+a_1)\times 2+a_0$$

公式中需要作4次加法和3次乘法。如果是8位二进制数，则需要作8次加法和7次乘法，其余依此类推。

例1 从键盘输入4位十六进制数（它对应的二进制数是补码表示的带符号数），试编制一程序，把它们转换为带符号非组合型BCD码，并在屏幕上显示出来。

算法分析：

1、从键盘输入4位十六进制数，存放在以HEXBUF+2为首址的4个字节单元中。其中HEXBUF+1单元中为输入的数据个数。

2、将ASCII码表示的4位十六进制数转换为16位二进制数，并暂时存放在BX中；

3、确定十进制数的符号，并把符号(+或-)存放在BCDBUF单元中；

**4、对BX中的二进制数采用前述算法转换为十进制数（非组合型BCD码），转换结果存放在以BCDBUF+1为首址的5个字节单元中；**

**5、把转换结果的5个非组合型BCD码形成相应的ASCII码**

**6、显示结果。**

```
 TITLE  CONVERT HEXADECIMAL TO BCD
DATA    SEGMENT
PROMPT  DB "INPUT HEXADECIMAL(4DIGITS):$'
HEXBUF  DB 5,0,5 DUP(0)
DISP        DB 0AH,0DH
BCDBUF  DB 6 DUP(0),'$'
DATA        ENDS
STACK1  SEGMENT   PARA  STACK
                DW 20H DUP(0)
STACK1  ENDS
```

```
COSEG   SEGMENT
        ASSUME CS:COSEG,DS:DATA,SS:STACK1
HEXBCD: MOV AX,DATA
        MOV DS,AX
;输入4位十六进制数
        LEA DX,PROMPT   ;显示提示信息
        MOV AH,09H
        INT 21H
        LEA DX,HEXBUF   ;输入数据
        MOV AH,0AH
        INT 21H
;ASCII码转换为16位二进制数并存入BX中
        LEA SI,HEXBUF+2   ;取十六进制数ASCII码首址
        MOV BX,0   ;暂存二进制数的寄存器清零
        MOV CH,HEXBUF+1    ;取输入数据个数
HEX1:   MOV AL,[SI]        ;取一个十六进制数字符
        CMP AL,'9'
        JBE NUMB           ;小于等于‘9’是数字符
```

```
        SUB AL,07H          ;是字母符
NUMB:   AND AL,0FH
        MOV CL,4
        SAL BX,CL
        OR  BL,AL
        INC SI
        DEC CH
        JNE HEX1
;确定十进制数的符号
        MOV BCDBUF,'+'
        TEST BX,8000H
        JNS PLUS        ;是正数
        MOV BCDBUF,'-' ;是负数
        NEG BX          ;求补后变为原码
```

```
;将二进制数转换为非组合型BCD码，从二进制
;数高位起,进行15次加和乘
PLUS: MOV CH,0FH     ;“加乘”运算的次数
LOP0: SHL BX,1       ;最高位二进制数送CF
      CALL ADDIT     ;先加1位二进制数
      CALL MULTI     ;再乘2
      DEC CH
      JNE LOP0
      SHL BX,1       ;加最低位二进制数
      CALL ADDIT ;与上一起？
;把非组合型BCD码转换为ASCII码形式
      LEA DI,BCDBUF+1
      MOV CX,5
LOP1: OR  BYTE PTR [DI],30H
      INC DI
      LOOP LOP1
```

显示结果

```
        LEA DX,DISP
        MOV AH,09H
        INT 21H
        MOV AH,4CH ;程序结束
        INT 21H
;多字节BCD码加1位二进制数子程序
ADDIT   PROC
        LEA DI,BCDBUF+5 ;从低位开始
        MOV CL,5        ;取字节数
ADD1:   MOV AL,[DI]     ;取BCD码
        ADC AL,0        ;加二进制数位(CF)
        AAA             ;十进制数运算校正
        MOV [DI],AL     ;存BCD码
        DEC DI
        DEC CL
        JNE ADD1
        RET
ADDIT   ENDP
```

```
;多字节BCD码乘2子程序
MULTI   PROC
        LEA DI,BCDBUF+5   ;从低位开始
        MOV CL,5
        CLC
MUL1:   MOV AL,[DI]       ;取BCD码
        ADC AL,AL         ;乘2
        AAA               ;十进制数运算校正
        MOV [DI],AL       ;存BCD码
        DEC DI
        DEC CL
        JNE MUL1
        RET
MULTI   ENDP
COSEG   ENDS
        END  HEXBCD
```