第七章　数值运算程序设计

7.1.1　十进制数加减运算

- ＣＰＵ的运算以二进制数为基础，不能直接进行十进制运算。
- 若需使用十进制表示数据，则需做数制转换或运算调整。

数制转换法

- 将十进制表示的原始数据，例如ＡＳＣＩＩ码，转换为等价的二进制表示；
- 使用二进制运算指令完成二进制运算；
- 再将运算结果转换为十进制形式。

数制转换法

- 该方法原理简捷，但需要编制二、十进制数相互转换的子程序，使运算时间加长。

运算调整法

- 使用特定二进制编码表示十进制数，例如ＢＣＤ编码；
- 使用二进制运算指令实施运算；
- 搭配使用运算调整指令对运算结果进行十进制调整，使运算结果仍为ＢＣＤ码。

运算调整法

- 该方法原理较复杂，但使用非常方便，能够提高运算效率，因不需完成二、十进制数的相互转换。

（１）ＢＣＤ码

- ４个连续的二进制位可看作一个十六进制位，取值范围为０－Ｆ。
- 若将其变化范围加上限制，仅使用０－９的取值范围，则可将它看作一个十进制位。
- ＢＣＤ码即是用４个二进制位来表示１个十进制位，变化范围限定为０－９。

1）非组合类型ＢＣＤ码

- 一个字节的低４位表示一个ＢＣＤ码，高４位可以为０，也可以为０００１１B（ＡＳＣＩＩ码字节）。
- 数字字符的ＡＳＣＩＩ码低４位就是一个ＢＣＤ码。

2）组合类型ＢＣＤ码

- 一个字节的低４位和高４位分别表示两个ＢＣＤ码。
- ＢＣＤ码与数字字符ＡＳＣＩＩ码间存在必然的联系。
- 将ＢＣＤ码置于字节低４位，高4位指定为０００１１B，则此字节为对应十进制数字的ＡＳＣＩＩ码。

（２）ＢＣＤ码加法运算

- 考虑一位十进制数的加法运算会遇到哪些情况.
- １）
- 　　０２　Ｈ
- ＋　　０３　Ｈ
- 　　０５　Ｈ
- 特征：运算结果的低４位为０－９，ＡＦ＝０

- 调整：虽然是十六进制的加法，但是运算特征和十进制加法完全一致，不需要任何调整。

（２）ＢＣＤ码加法运算

- ２）
- 　　０８　Ｈ
- ＋　　０４　Ｈ
- 　　０Ｃ　Ｈ
- 特征：结果低４位大于９，ＡＦ＝０

- 调整：对于十六进制加法，逢１６进１，但对十进制加法，逢１０进１，此时两种加法产生不一致，需要作调整。

（２）ＢＣＤ码加法运算

- 对十进制加法来说，这种情况应该进位，却未进位，则使用十六进制方式迫使它进位，即加６，把逢１０强行变为逢１６。

- 　　０ＣＨ
- ＋　　０６Ｈ
- 　　１２Ｈ

- 如果把１２Ｈ看作十进制的运算结果，那么结果是正确的，因为８＋４＝１２

（２）ＢＣＤ码加法运算

- ３）
- 　　０９　Ｈ
- ＋　　０８　Ｈ
- 　　１１　Ｈ
- 特征：运算结果低４位为０－９，ＡＦ＝１

- 调整：对于十进制加法，应该进位，但应该向高位进１０，而不是进１６，所以多进了６，需要对运算结果作调整。

（２）ＢＣＤ码加法运算

- 这种情况应该进位，但多进了６，调整方法为加６，弥补进位的损失。

- １１　H
- ＋　０６　H
- １７　H

- 如果把１７H看作十进制运算的结果，那么结果是正确的，因为９＋８＝１７

（２）ＢＣＤ码加法运算

- 对于一位十进制数加法，可能遇到的进位情况仅有以上三种，对每种情况，都有相应的进位调整方法。

- 通过这些调整，可以把十六进制加法调整为十进制加法。

（２）ＢＣＤ码加法运算

- １）结果的低４位为０－９，ＡＦ＝０，不需要调整

- ２）结果低４位大于９，ＡＦ＝０，该进位但未进位，加６迫使运算进位。

- ３）结果低４位为０－９，ＡＦ＝１，该进位，但多进了６，加６作为补偿。

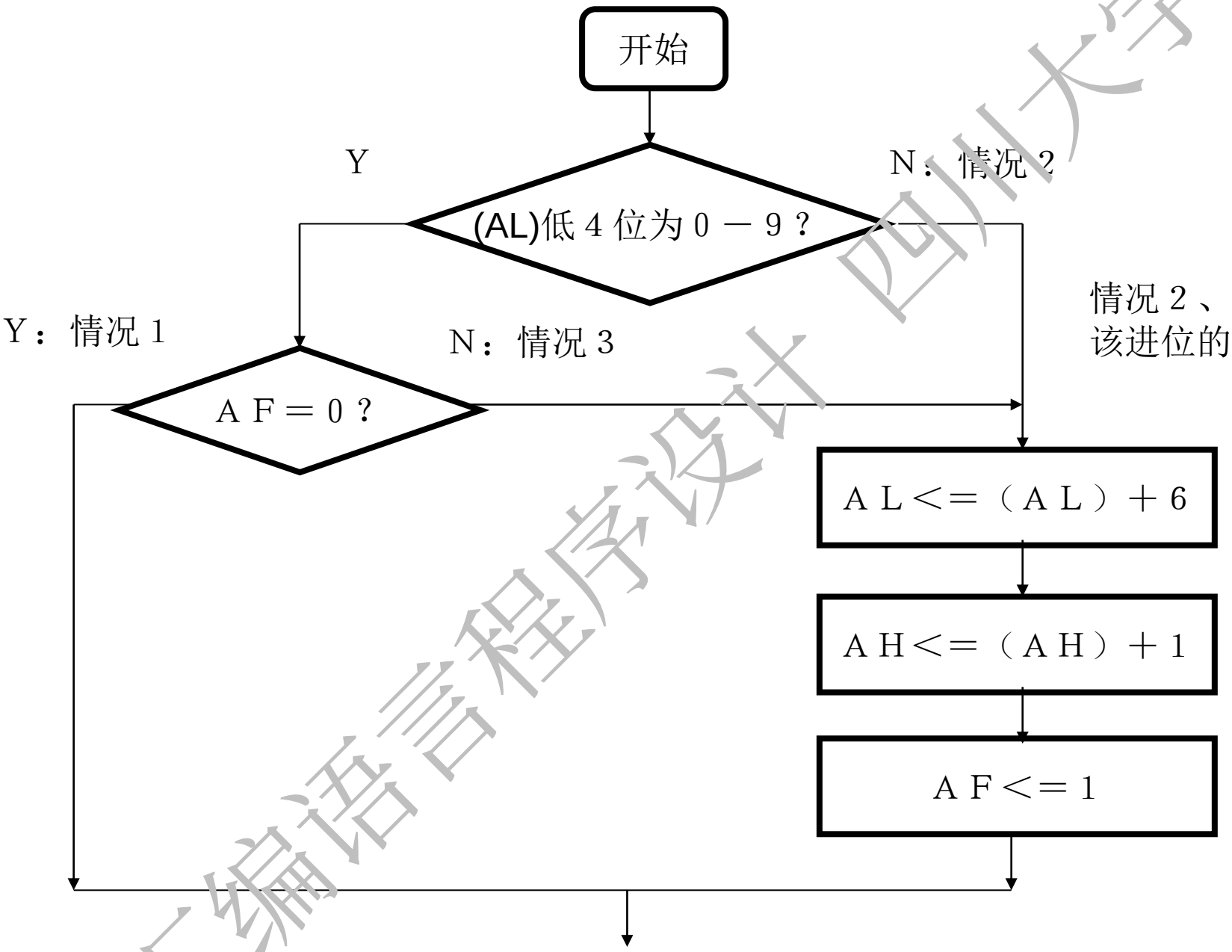
（２）非组合型ＢＣＤ码加法调整指令

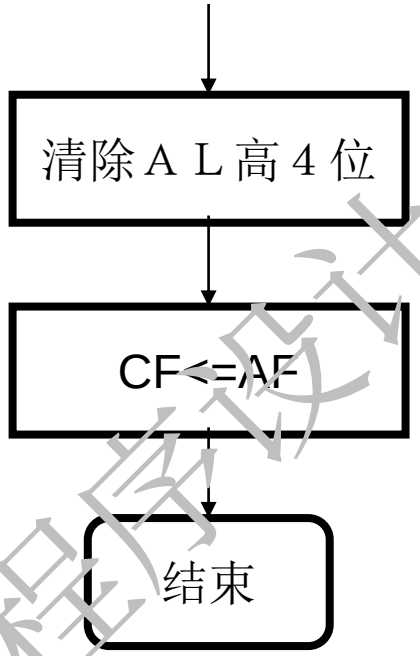
- 指令格式：ＡＡＡ
- （ASCII Adjust for Addition）

- 标志位影响：仅影响ＡＦ、ＣＦ标志。

- 功能：将非组合型BCD码加法运算结果调整为正确的非组合型BCD码，并且将十进制进位反映到AF、CF标志。

- 调整过程见如下框图：（注意，这是ＡＡＡ这条指令的处理流程，不是某个程序的流程）





CF、AF意义相同

（２）非组合型ＢＣＤ码加法调整指令

- AAA指令标志位解释：AF和CF取值一定相同，意义相同。

- 如果为0，表示对这一位BCD码进行加法时没有产生十进制进位。

- 如果为1，表示对这一位BCD码进行加法时产生了一个十进制进位。

执行AAA指令的前提条件

- 1）进行十进制加法调整以前，必须先使用ADD或ADC指令作二进制加法。
- 2）相加的两个操作数都必须是一位非组合型BCD码
- 3）相加的结果必须保存在寄存器AL中
- 只有满足上面3个前提条件，执行AAA指令才是有意义的。

非组合型BCD码加法示例

```
DATA  SEGMENT
DAA1        DB   5,9,5,9
DAA2        DB   9,7,9,7
RESULT    DB   'THE RESULT IS:'
RESULT1   DB   5  DUP(0), '$'
DATA  ENDS

STACK1  SEGMENT  STACK
              DW   20H DUP(0)
STACK1  ENDS
```

非组合型BCD码加法示例

```
CODE  SEGMENT
        ASSUME  CS:CODE, DS:DATA, SS:STACK1
MAIN:  MOV     AX,    DATA
       MOV     DS,    AX
       MOV     SI,     3
       LEA     DI,     RESULT1+4
       MOV     CX,     4
       CLC
LOP:   MOV     AL,     DAA1[SI]
       ADC     AL,     DAA2[SI]
       AAA
       LAHF
```

非组合型BCD码加法示例

```
OR      AL,   30H
MOV     [DI],  AL
SAHF
DEC     DI
DEC     SI
LOOP    LOP
MOV     BH,  30H
JNC     NEXT
MOV     BH,  31H
```

非组合型BCD码加法示例

```
NEXT:  MOV     [DI],  BH
       MOV     DX,  OFFSET  RESULT
       MOV     AH,  09H
       INT     21H
       MOV     AH,  4CH
       INT     21H
CODE   ENDS
       END     MAIN
```

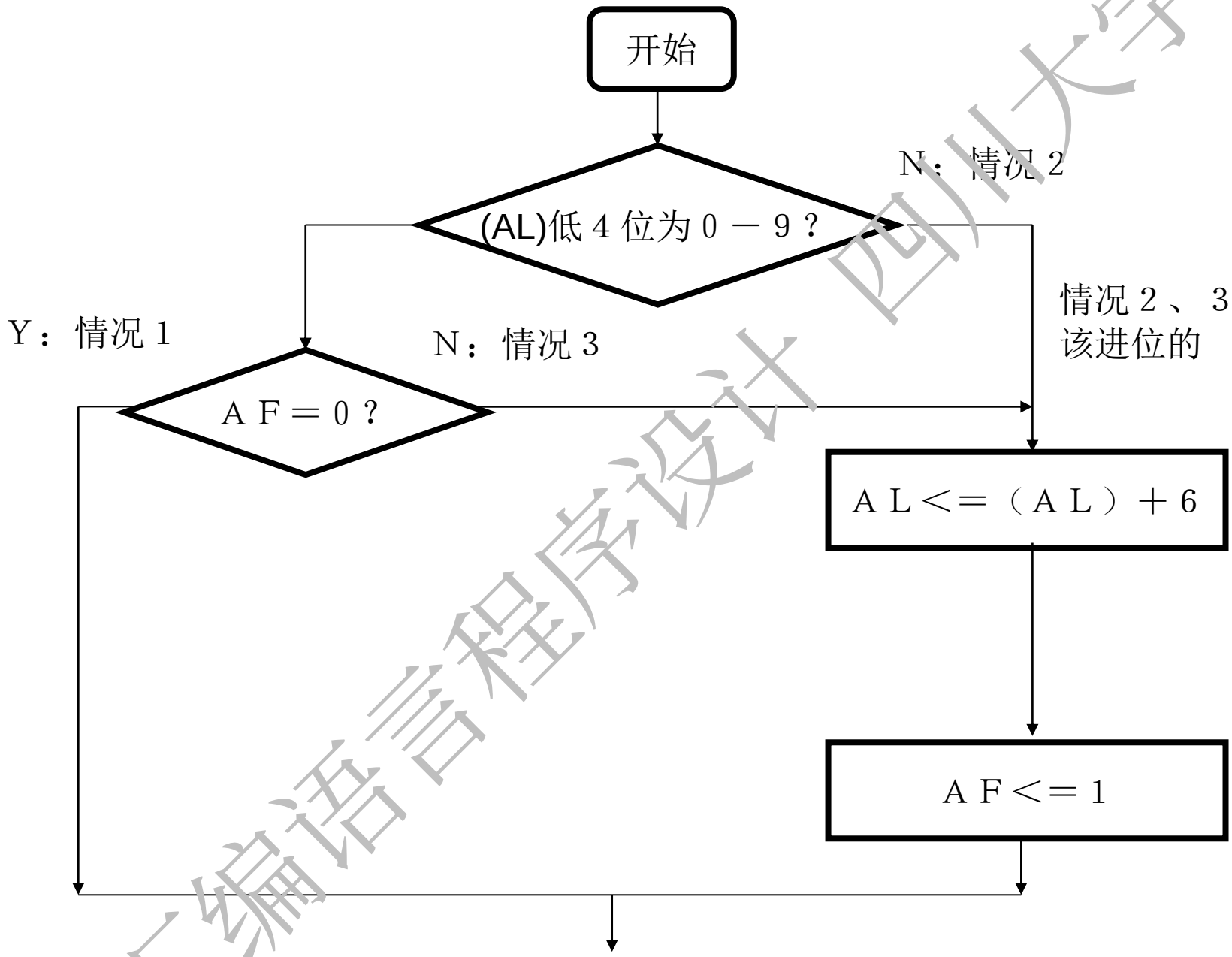
（２）非组合型ＢＣＤ码加法调整指令

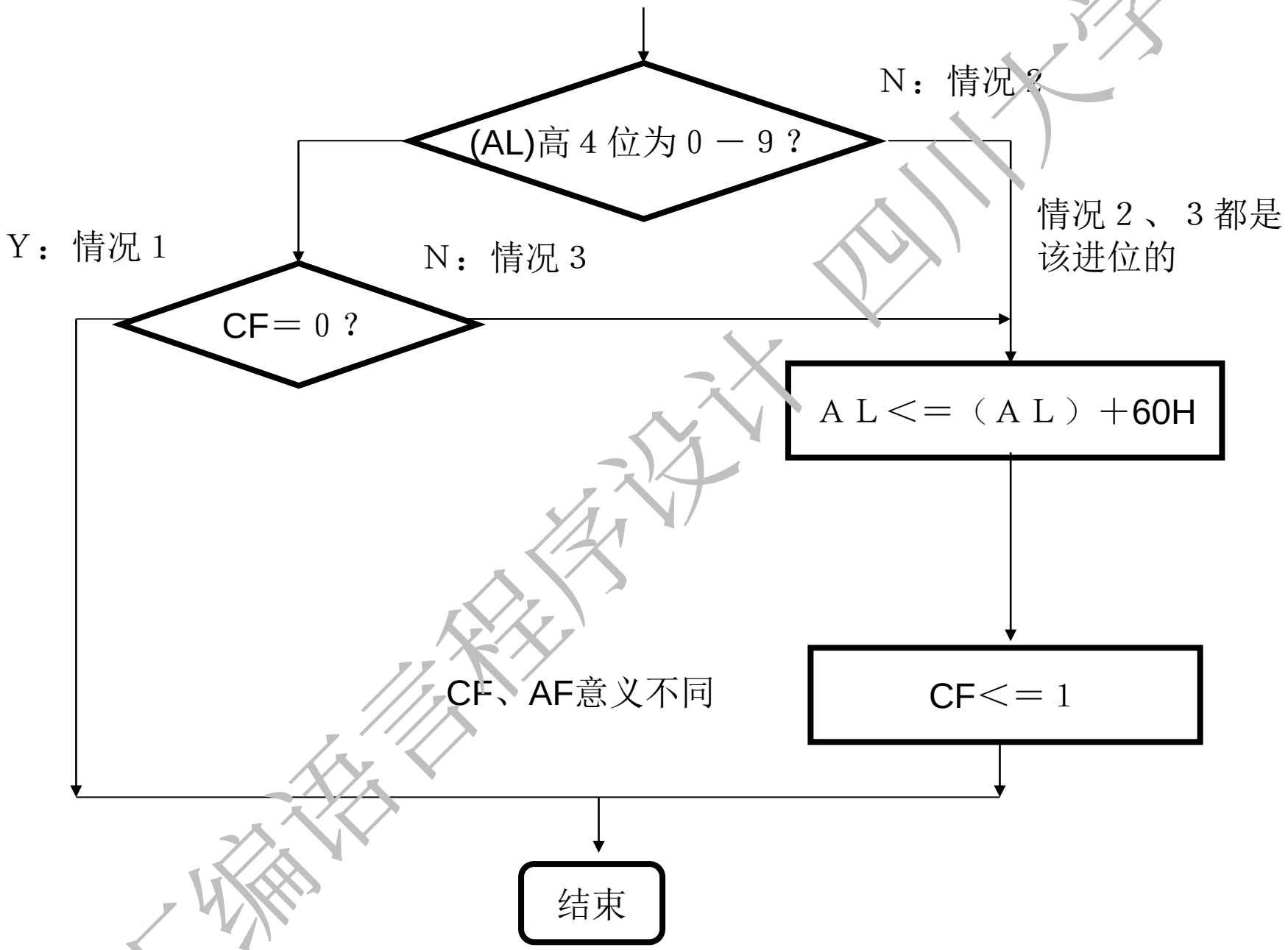
- BCD码加法需要注意：

- 两个N位BCD码数据相加时，可能产生一个N+1位的BCD码结果。因此在程序中应将存放结果的空间留够N+1位BCD码空间。

（3）组合型BCD码加法调整指令DAA

- 指令格式：DAA
- 标志位影响：AF、CF
- 功能：把组合型BCD码的加法运算结果调整为正确的组合型BCD码，并把十进制进位反映到AF、CF。
- 调整过程见如下流程图：





组合型BCD码加法调整指令的标志位解释

- AF：低位BCD码（低4位）是否向高位产生十进制进位，如果为0，表示没有进位，如果为1，表示产生了进位。

- CF：高位BCD码（高4位）是否向更高位产生十进制进位，如果为0，表示没有进位，如果为1，表示产生了进位。

组合型BCD码加法调整指令的标志位解释

- AF标志不常用，因为通过二进制加法运算，低位BCD码的进位已经反映到了高位BCD码的加法当中。

- CF标志会经常使用在ADC指令对多字节的组合型BCD码加法当中，反映低位向高位产生的十进制进位。

执行DAA指令的前提条件

- 1）进行十进制加法调整以前，必须先使用ADD或ADC指令作二进制加法。
- 2）相加的两个操作数都必须是两位BCD码构成的组合型BCD码。
- 3）相加的结果必须保存在寄存器AL中
- 只有满足上面3个前提条件，执行DAA指令才是有意义的。

（4）BCD码减法运算

- 首先考虑一位BCD码作减法会遇到哪些情况。

- 1）
- 05 H
- - 03 H
- 02 H
- 特征：运算结果为0-9，AF=0

- 调整：不需要调整，因为在这种情况，十六进制减法和十进制减法完全一致。

（4）BCD码减法运算

- 2）
- 01
- - 03
- 0E
- 特征：运算结果大于9，AF=1

- 调整：因为借位是十六进制的，所以对于十进制运算来说，多借了6，应该从结果中减去6。

（4）BCD码减法运算

- 0E H
- - 06 H
- 08 H

- 从十进制运算角度来看，调整以后的结果是正确的，1-3=8（从高位借位以后）。

（4）BCD码减法运算

- 3）
- 01 H
- - 09 H
- 08 H
- 特征：运算结果为0-9，AF=1，

- 调整：产生了十六进制的借位，多借了6，应该从结果中减掉6。

（4）BCD码减法运算

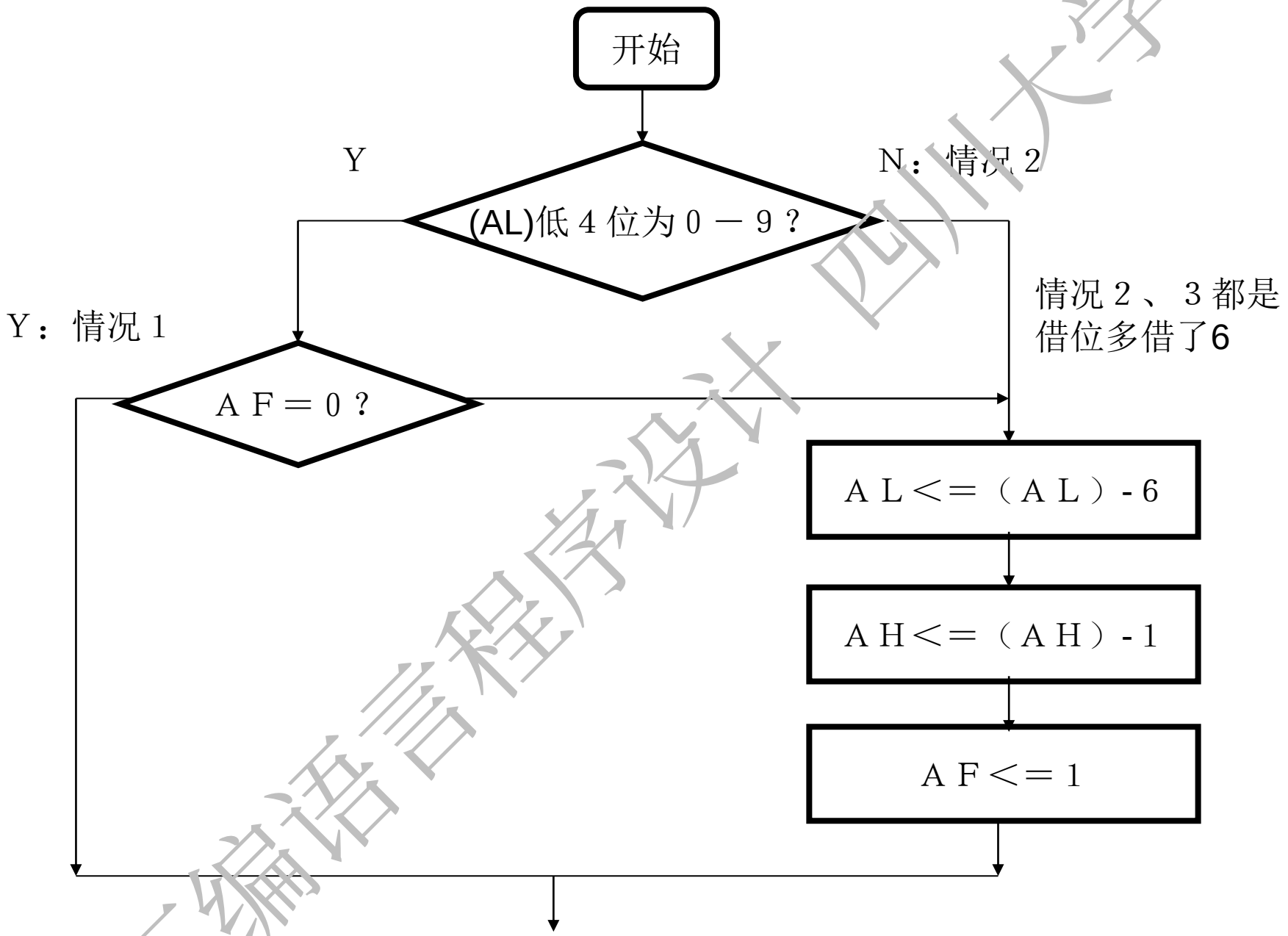
- 08 H
- - 06 H
- 02 H

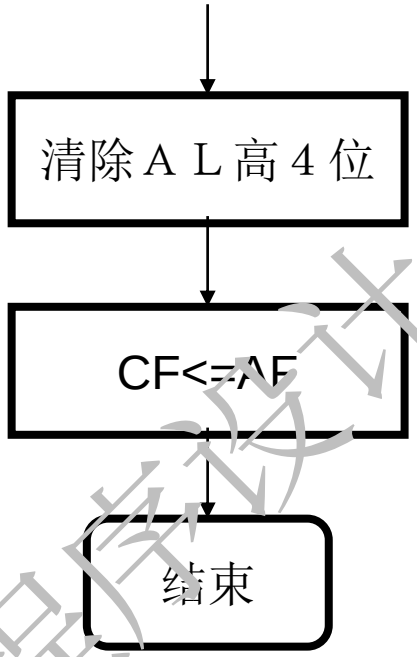
- 从十进制角度讲，调整后的结果是正确的，因为1-9=2（向高位借位以后）。

- 实际上，在减法运算中，第三种情况和第二种情况可以归纳为同一种情况，因为有共同的特征：AF=1，而且问题一致，借位时多借了6。

（5）非组合型BCD码减法调整指令AAS

- 指令格式：AAS
- 标志位影响：AF、CF
- 功能：把非组合型BCD码的减法结果调整为正确的非组合型BCD码，并且把十进制借位反映到AF、CF。
- AAS指令的调整过程见如下流程：（实际上减法调整比加法简单，但为保持同类指令的规整性，使用了和AAA指令相似的流程）





CF、AF意义相同

AAS指令标志位解释

- AF和CF取值一定相同，意义相同。
- 如果为0，表示对这一位BCD码进行减法时没有产生十进制借位。
- 如果为1，表示对这一位BCD码进行减法时产生了一个十进制借位。
- AAS指令和AAA指令一样,执行前必须有3个类似的必要条件,只是使用的运算指令为SUB指令或者SBB指令。

非组合型BCD码减法调整示例

```
DATA  SEGMENT
SUB1      DB   5, 2, 2
SUB2      DB   6, 2, 1
RESULT   DB   'THE RESULT IS:', 4 DUP(0), '$'
DATA  ENDS

STACK1   SEGMENT   STACK
              DW   20H   DUP(0)
STACK1   ENDS

CODE      SEGMENT
ASSUME   CS:CODE, DS:DATA, SS:STACK1
```

非组合型BCD码减法调整示例

```
MAIN:  MOV    AX,  DATA
       MOV    DS,  AX
       LEA    SI,   SUB1+2
       LEA    DI,   SUB2+2
       MOV    AH,  '+'
       MOV    BX,  0
       MOV    CX,  3
LOP1:  MOV    AL,  SUB1[BX]
       SUB    AL,  SUB2[BX]
       JZ     NEXT1
       JNC    NEXT2
```

非组合型BCD码减法调整示例

```
        MOV   AH, '-'
        XCHG  SI, DI
        JMP   NEXT2
NEXT1:  INC   BX
        LOOP  LOP1
NEXT2:  MOV   RESULT+14, AH
        LEA   BX, RESULT+17
        MOV   CX, 3
        CLC
```

非组合型BCD码减法调整示例

```
LOP2:   MOV     AL, [SI]
        SBB     AL, [DI]
        AAS
        LAHF
        OR      AL, 30H
        MOV     [BX], AL
        SAHF
        DEC     SI
        DEC     DI
        DEC     BX
        LOOP    LOP2
```

非组合型BCD码减法调整示例

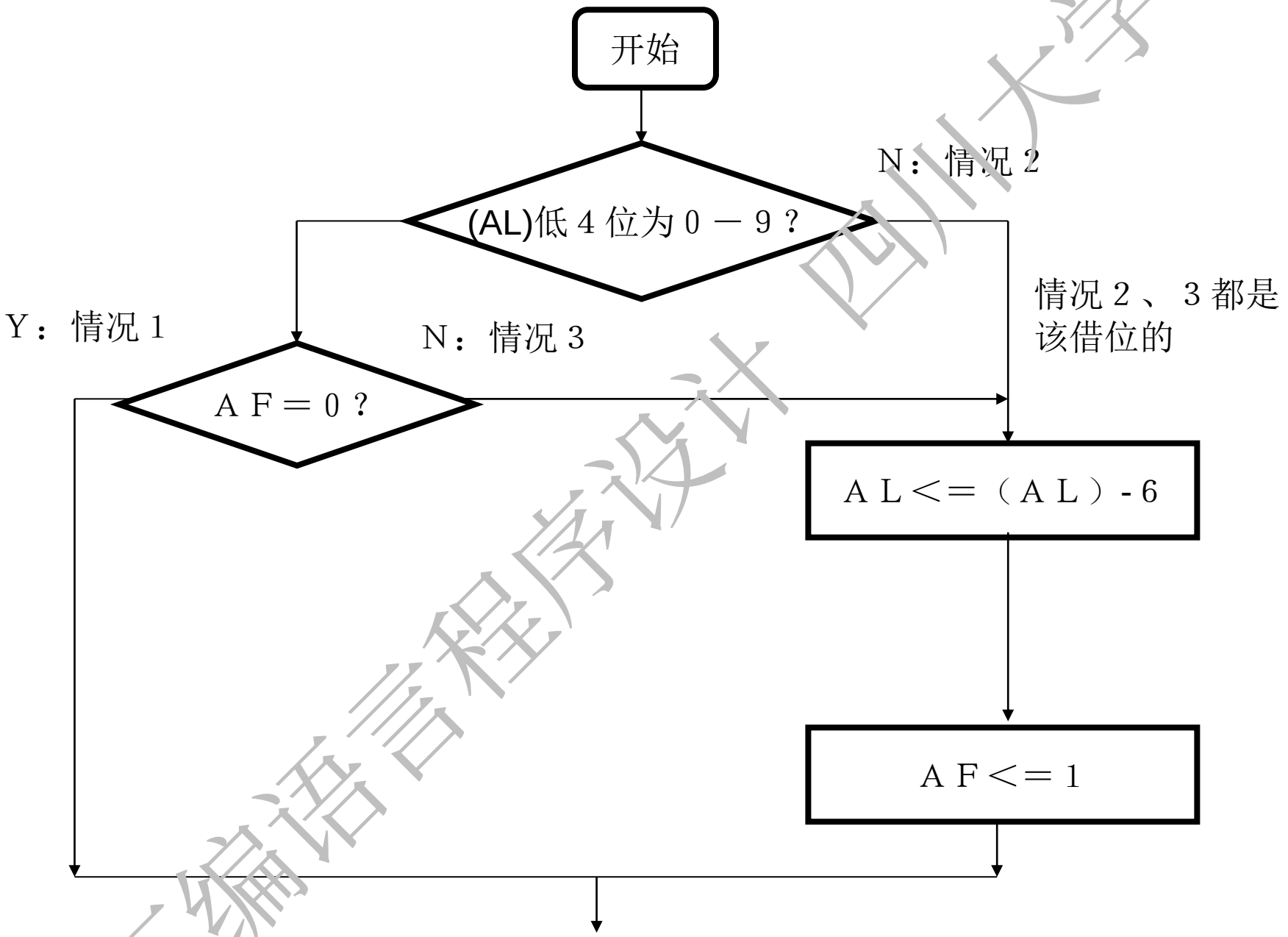
```
        MOV   DX, OFFSET RESULT
        MOV   AH, 09H
        INT   21H
        MOV   AH, 4CH
        INT   21H
CODE    ENDS
        END   MAIN
```

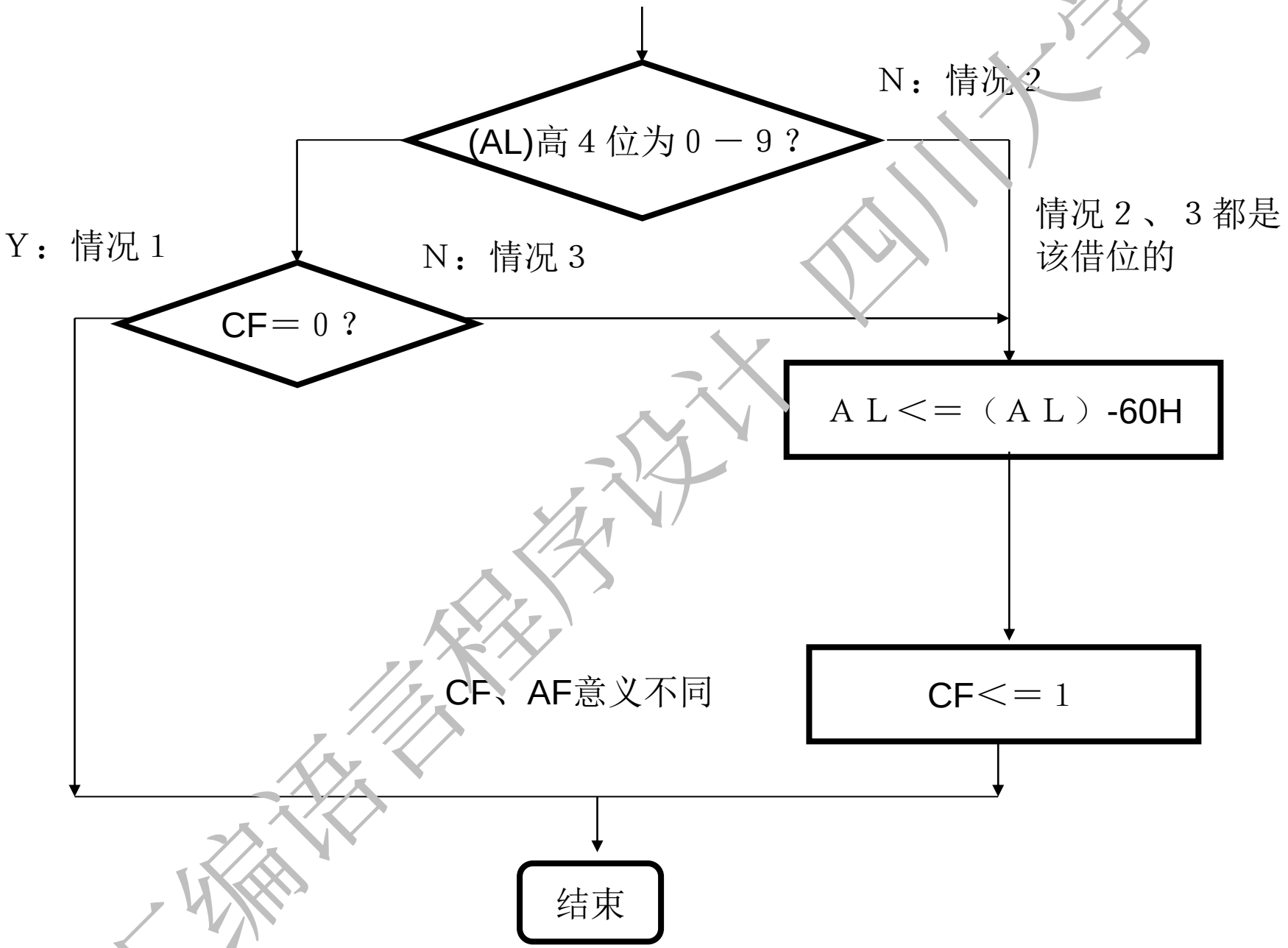
非组合型BCD码减法调整示例

- BCD码减法结果可能是负数，但BCD码运算本身没有考虑符号，所以在程序中需单独判断运算结果的符号，并合理选择减数和被减数，即绝对值较大的一个数作为被减数。

（6）组合类型BCD码减法调整指令DAS

- 指令格式：DAS
- 标志位影响：AF、CF
- 功能：把组合类型BCD码的减法结果调整为正确的组合类型BCD码，并且把十进制的借位反映到AF、CF标志中。
- DAS指令的调整过程如下图所示：





DAS指令的标志位解释

- AF：低位BCD码（低4位）是否向高位产生十进制借位，如果为0，表示没有借位，如果为1，表示产生了借位。

- CF：高位BCD码（高4位）是否向更高位产生十进制借位，如果为0，表示没有借位，如果为1，表示产生了借位。

- 和AAS指令相似，在多字节BCD码减法中，使用较多的是CF标志。

加减法调整的本质

- BCD码加减法的本质：把十六进制进位或借位调整为十进制的进位或借位。
- 如果BCD码加法产生了进位，那么一定多进了6，应该加上一个6作补偿。
- 如果BCD码减法产生了借位，那么一定多借了6，应该减去一个6作补偿。

多位BCD码运算

- 多位BCD码的加减法与多字节、多字二进制数的加减法相似，只是需要使用调整指令对每次运算进行调整。
- BCD码运算未考虑符号，若考虑带符号运算，则需单独考虑，一般用ASCII码表示符号。
- 若考虑符号，则需先将指定的运算转换为两个正数的等价运算，并且在运算前先判断结果的符号；若是减法，还需确定正确的被减数和减数，才能得到正确结果。

BCD码加法的符号判断

- 正+正 => 直接进行BCD码加法运算，结果符号为正
- 正+负 => 正 – 正，进行BCD码减法运算，需要单独判断结果的符号，并由此确定被减数和减数。
- 负+负 => -（正+正），进行BCD码加法运算，结果符号为负

BCD码减法的符号判断

- 正-正　直接进行BCD码减法运算，但是需要先判断结果的符号，并由此决定被减数和减数。
- 正-负 => 正+正　进行BCD码加法运算，结果符号为正。
- 负-负 => 负+正 => 正-正　和第一种情况相同

课后思考

- 如何编制一个程序，能完善解决BCD码的加减运算（BCD码位数可以自定）。

7.2.1 二进制乘除运算

- 无符号数的乘除运算
- 带符号数的乘除运算

（1）无符号数乘法指令MUL（Multiplication）

- 指令格式：MUL　SRC
- SRC可使用除立即数外的任一种操作数（字、字节）

- 功能：
- 字节乘法：AX<=（AL）*（SRC）
- 字乘法：DX：AX<=（AX）*（SRC）

- 标志位影响：CF、OF，如果运算结果为0，CF=OF=0；如果非0，CF=OF=1。

（2）带符号数乘法指令IMUL（Signed Integer Multiplication）

- 指令格式：IMUL　SRC

- 功能：与MUL指令相同，只是操作数都解释为补码，完成的乘法操作是补码乘法。

- 标志位影响：CF、OF，如果运算结果的高半部是低半部的符号扩展，那么CF=OF=0；如果不是，则CF=OF=1。

（3）无符号数除法指令DIV（Unsigned Division）

- 指令格式：DIV　SRC
- SRC为除数，可使用除立即数外的任意操作数（字、字节）
- 标志位影响：无

（3）无符号数除法指令DIV（Unsigned Division）

- 功能：
- 字节除法：ＡＬ<=（ＡＸ）／（ＳＲＣ）的商
- 　　ＡＨ<=（ＡＸ）／（ＳＲＣ）的余数

- 字除法：ＡＸ<=（DX:AX）／（ＳＲＣ）的商
- 　　ＤＸ<=（ＤＸ:AX）／（ＳＲＣ）的余数

0型中断

- 在以下情况中，除法指令会引起 0 型中断：

- １）除数（ＳＲＣ）＝＝０
- ２）商太大，超过了限定的范围

0型中断

- 一般出现0型中断，是因为程序中存在较严重的逻辑错误，通常是在循环结构中不断修改除数引起的。

- 0 型中断一般显示“ＯＶＥＲＦＬＯＷ”信息，用于提示用户，除法指令出现了严重的错误。

（４）带符号数除法指令ＩＤＩＶ（Signed Integer Division）

- 指令格式：ＩＤＩＶ　　ＳＲＣ
- 功能：和ＤＩＶ指令一致，只是将操作数看作补码，完成的除法操作是补码除法。
- 标志位影响：无
- ０型中断：如果（ＳＲＣ）＝０，或者运算结果超出了补码表示范围，那么会产生０型中断。

（５）字节、字扩展指令

- 指令格式：ＣＢＷ
- （Convert Byte to Word）

- 功能：扩展字节操作数的符号位，把它转变为字操作数。ＡＨ<=（ＡＬ）的符号位。

- 标志位影响：无

（５）字节、字扩展指令

- 指令格式：ＣＷＤ
- （Convert Word to Double word)

- 功能：扩展字操作数的符号位，把它转变为双字操作数。ＤＸ<=（ＡＸ）的符号位。

- 标志位影响：无

（5）字节、字扩展指令

- 由于除法指令的被除数只能是字或双字，所以有时需要使用符号位扩展指令扩展字节补码和字补码的符号位，得到更长的被除数以满足除法指令对被除数长度的要求。

7.2.2 十进制数乘除法运算

- 使用BCD码除了可以进行十进制的加减运算以外，同样可以进行乘除法，计算过程与手算方式类似。

（1）BCD码乘法

- 与分析BCD码加减法一样，首先考虑一位BCD码相乘的情况。

- 09 H
- * 09 H
- 51 H

（1）BCD码乘法

- 最大的两个BCD码相乘得到的结果小于0FFH，完全可以放在一个字节中；

- MUL指令的乘积存放在AX中，如果两个乘数都是一位的BCD码，则乘积完全在AL中，（AH）为全0。

- 调整过程：AH <=（AL）/ 0AH 的商
- AL <=（AL）/ 0AH 的余数

（1）BCD码乘法

- 51H / 0AH 的商为 08H（乘积十位上的BCD码），余数为01H（乘积个位上的BCD码）

- 通过除以0AH这一个过程，乘积已经被调整为十进制形式，9*9=81。

（2）非组合型BCD码乘法调整指令AAM

- 指令格式：AAM
- 功能：（AL）解释为乘积，调整为正确的BCD码后送到AX保存。
- AH <=（AL）/ 0AH 的商
- AL <=（AL）/ 0AH 的余数
- 标志位影响：SF、ZF、PF和其他算术运算指令中解释一致，OF、CF、AF不确定。

（2）非组合型BCD码乘法调整指令AAM

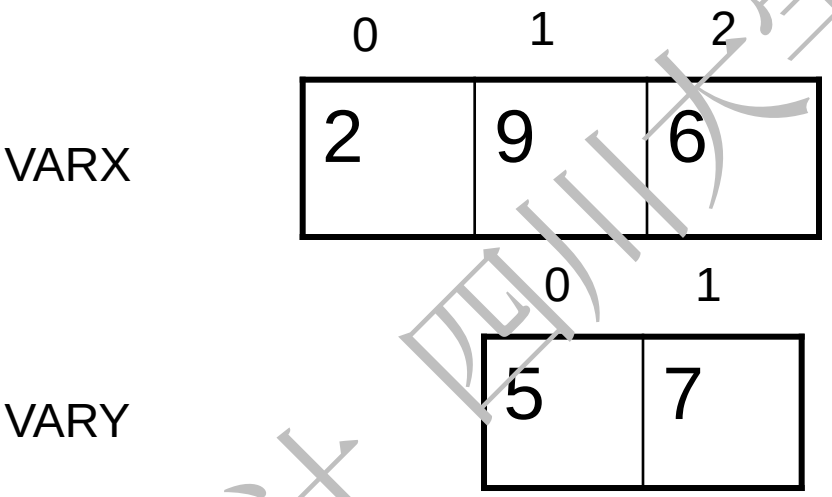
- AAM指令仅对一位BCD码乘法作调整，若要实现任意位的BCD码相乘，则需在程序设计时考虑整个乘法过程。

- 实际上BCD码乘法过程和手算乘法过程一致，为理解这一过程在程序中的具体实现，先看一个手算乘法的例子。

（2）非组合型BCD码乘法调整指令AAM

- 2 9 6
- * 5 7
-
- 4 2
- 6 3
- 1 4
- 3 0
- 4 5
- 1 0
- 1 6 8 7 2

BCD码乘法过程示意



BCD码乘法考虑符号

- 与加减法一样，BCD码乘法也需要单独考虑符号，具体的运算都是正数之间的乘法。
- 正*正　直接进行，运算结果符号为正
- 正*负 => -（正*正）直接进行，运算结果符号为负。
- 负*负 => 正*正 直接进行，运算结果符号为正

（3）BCD码除法调整指令

- BCD码除法和乘法相似，也是对手算过程的一种模仿。
- 首先理解2位BCD码除以1位BCD码的情况是理解多位BCD码除法的基础。
- 与前面介绍的十进制调整指令不同，BCD码除法调整指令是在除法指令执行前使用的。

（3）BCD码除法调整指令

- BCD码除法调整指令假定被除数（两位非组合类型的BCD码）是存放在AX中的，（AH）为高位BCD码，（AL）为低位BCD码。

- 调整过程：AL<=（AH）*0AH+（AL）
- AH <= 0

- 调整目的：将AX中的两位BCD码转换为对应的二进制数存放在AL中，这个过程正好和乘法调整是相反的。

手算除法的简单例子

- 176 / 5 ：（176和5都是BCD码表示）
- 1）01 / 5 = 商0 余 1
- 2）17 / 5 = 商3 余 2
- 3）26 / 5 = 商5 余 1
- 运算结果：把每次除法的商按照先后顺序连起来，为35，余数取最后一次除法的余数，为1。

手算除法的特征

- 1）每一次局部除法运算都是两位BCD码除以一位BCD码（除数）。
- 2）每一次得到的余数（一位BCD码），和被除数中下一次参加局部除法的BCD码组成新的2位BCD码被除数。
- 这些特征也就决定了BCD码除法的算法思路。

BCD码除法的局限

- 上面的例子中，只考虑了除数为一位BCD码的情况，若除数为多位BCD码，则在实行除法以前，需使用AAD指令调整为二进制数。

- 整个除法过程将变得更加复杂，并且对于除法操作，BCD码运算有局限性，因为二进制除数最大为一个字。

- BCD码的除法运算也可以使用逻辑上比较简单的方法：数制转换。

7.3 多精度数运算

- 多精度运算是指多字节、多字的算术运算。
- 由于8086/8088 CPU所能处理的最大数据为16位数据，如果超过这个限制，那么必须编制相应的程序来进行算术运算。
- 对于多精度数的加减运算，主要考虑将低字或低字节的进位、借位反映到高字或高字节的加减运算中。具体使用的指令为ADC、SBB。

多精度乘除法运算

- 多精度数乘除法运算和BCD码乘除法运算非常相似，也是对手算方法的一个模仿过程。

多精度乘除法运算

- BCD码乘法是以两个BCD码相乘为基础，算出乘数各个数位与被乘数各个数位相乘的部分积，把这些部分积错位相加得到最后的结果。

- 由于十进制数不能完全和几个二进制数位对应起来，十进制和二进制之间没有直接的联系，所以在每一次运算以后都必须进行十进制调整。

多精度乘除法运算

- 对于多精度数来说，原理是相通的，就字节来说，每一个字节可以看作是256进制数的一个数位；

- 同样可以求乘数各个数位与被乘数各个数位的部分积，最后把这些部分积错位相加得到最后的结果。

多精度乘除法运算

- 如果多精度数以字节为单位进行运算，那么解释为256进制数（8个二进制位），如果以字为单位进行运算，那么解释为65536进制数（16个二进制位）。

- 这些进制和16进制（4个二进制位）相似，在使用二进制运算指令时不需要任何调整。

多精度乘除法运算

- 对于除法，多精度数和BCD码也具有非常相似的特征。
- 对于BCD码，以两位BCD码除以一位BCD码作为基础模仿手算过程。
- 对于多精度数，也可以模仿这样的过程，2位256进制（或65536进制）数除以一位256进制数。