第五章　程序设计的基本方法

本章将介绍的程序设计方法有以下几种：

① 顺序程序设计；
② 分支程序设计；
③ 循环程序设计；
④ 子程序设计

其它程序设计方法有以下几种：

① 宏指令

② 模块化程序设计

5．1 程序设计的基本步骤

汇编语言程序设计的一般步骤如下：

（1）分析问题，确定算法
（2）根据具体问题，确定存储空间，工作单元和寄存器。
（3）根据算法，编制程序流程图
（4）根据流程图编写程序
（5）上机调试程序

5．2 顺序程序设计

5.2.1算术运算类指令

- 加法指令
- 减运算指令
- 乘运算指令
- 除运算指令

1. 加法指令

ADD	加法
ADC	带进位加法
INC	加 1
XADD	交换并相加

(1) ADD 加法指令

格式：ADD OPD, OPS

执行的操作：OPD + OPS → OPD

例如：ADD EAX,EBX ;EAX + EBX→EAX

若指令执行前 CF = 0,EAX = 00000054H,EBX = 00120330H;

则指令执行后 EAX = 00120384H,CF = 0。

（2）ADC 带进位加法指令

格式：　　ADC　OPD,OPS

执行的操作：　OPD + OPS + C F→ OPD

其中 CF 为进位位的值。

例如：　　ADC　AX,DX　;AX + DX→AX

若指令执行前　　CF = 1,AX = 1234H,DX = 0112H;

则指令执行后　　AX = 1347H,CF = 0。

（3）INC 加 1 指令

格式：　　INC　OPD

执行的操作：　OPD + 1→OPD

例如：　　INC　DX　　;DX + 1→DX

若指令执行前　　CF = 1,DX = 3562H;

则指令执行后　　DX = 3563H,CF = 1。

以上 3 条指令都可做字或字节运算，386 及以上机型还可做双字运算，而且除 INC 指令不影响 CF 标志外，其他两条指令都影响条件标志位。

（4）XADD 交换并相加指令

2. 减法指令

SUB	减法
SBB	带借位减法
DEC	减 1
NEG	求补
CMP	比较
CMPXCHG	比较并交换
CMPXCHG8B	比较并交换 8 字节

（1）SUB 减法指令

格式：　　SUB　OPD,OPS

执行的操作：　OPD – OPS→OPD

例如：　　SUB　AX,DX;AX – DX→AX

若指令执行前　　CF = 1,AX = 1234H,DX = 0100H;

则指令执行后　　AX = 1134H,CF = 0。

(2) SBB 带借位减法指令

格式：　　SBB　OPD, OPS

执行的操作：　OPD - OPS - CF→OPD

其中 CF 为进位位的值。

例如：　　SBB　AX,DX　　;AX - DX - CF→AX

若指令执行前　　CF = 1, AX = 1234H, DX = 0100H;

则指令执行后　　AX = 1133H, CF = 0。

(3) DEC 减 1 指令

格式：　　DEC　OPD

执行的操作：　OPD - 1→OPD

例如：　　DEC　CX　　;CX - 1→CX

若指令执行前　　CF = 0, CX = 1000H;

则指令执行后　　CX = 0FFFH, CF = 0。

(4) NEG 求补指令

格式：　　NEG　OPD

执行的操作：　- OPD→OPD

亦即把操作数按位求反后末位加 1,因而执行的操作也可表示为：

0FFFFH - OPD + 1→OPD

例如：　　NEG　AX　　;AX 求反加 1→AX

若指令执行前　　AX = 0612H;

则指令执行后　　AX = 0F9EEH。

(5) CMP 比较指令

格式：　　CMP　OPD,OPS

执行的操作：　OPD - OPS

该指令与 SUB 指令一样执行减法操作，但它并不保存结果，只是根据结果设置条件标志位。CMP 指令后往往跟一条条件转移指令，根据比较结果产生不同的程序分支。

例如：　　CMP　　AX,BX　　;AX - BX

若指令执行前　　CF = 1,AX = 1000H,BX = 0100H;

则指令执行后　　CF = 0,ZF = 0,OF = 0,SF = 0,PF = 1。

(6) CMPXCHG 比较并交换指令

3. 乘法指令

MUL　　无符号数乘法

IMUL　　带符号数乘法

(1) MUL 无符号数乘法指令

格式：　　MUL　OPS

执行的操作：　字节操作数：AL * OPS→AX

字操作数：AX * OPS→DX:AX

双字操作数：　EAX * OPS→EDX:EAX

例如：　　MUL　CL

若指令执行前　　AL = 0B4H,CL = 11H;

则指令执行后　　AX = 0BF4H。

(2) IMUL 带符号数乘法指令

格式：　　IMUL　OPS

执行的操作与 MUL 相同，但必须是带符号数，而 MUL 是无符号数。

例如：　　IMUL　CL

若指令执行前　　AL = 0B4H，CL = 11H；

则指令执行后　　AX = 0FAF4H。

在乘法指令里，目的操作数必须是累加器，字运算为 AX，字节运算为 AL。两个 8 位数相乘得到的是 16 位乘积存放在 AX 中，两个 16 位数相乘得到的结果是 32 位乘积，结果存放在 DX:AX 中。其中 DX 存放高位字，AX 存放低位字。386 及以上机型可做双字运算。累加器为 EAX，两个 32 位数相乘得到 64 位乘积存放于 EDX:EAX 中。EDX 存放高位双字，EAX 存放低位双字。指令中的源操作数可以使用除立即数方式以外的任何一种寻址方式。

在 286 以后的机型中又增加了如下指令：

```
IMUL  OPD,OPS      ; OPD * OPS→OPD
IMUL  OPD,OPS,d    ; OPS * d→OPD,d 为立即数
```


4. 除法指令

DIV　　无符号数除法指令

IDIV　　带符号数除法指令

(1) DIV 无符号数除法指令

格式：　　DIV　OPS

执行的操作：

字节操作：16 位被除数在 AX 中，8 位除数为源操作数，结果的 8 位商在 AL 中，8 位余数在 AH 中。表示为：

AX/ OPS 的商→AL

AX/ OPS 的余数→AH

字操作：32 位被除数在 DX:AX 中，其中 DX 为高位字，AX 为低位字。16 位除数为源操作数，结果的 16 位商在 AX 中，16 位余数在 DX 中。表示为：

DX:AX/ OPS 的商→AX

DX:AX/ OPS 的余数→DX

双字操作：64 位被除数在 EDX:EAX 中。其中 EDX 为高位双字;32 位除数为源操作数,结果的 32 位商在 EAX 中,32 位余数在 EDX 中。表示为:

EDX:EAX/ OPS 的商→EAX

EDX:EAX/ OPS 的余数→EDX

商和余数均为无符号数。

例如：　　DIV ECX

若指令执行前　　EDX = 000005D2H, EAX = 60F40000H, ECX = 00BC5200H;

则指令执行后　　EAX = 0007EA00H(商), EDX = 00000000H(余数)。

(2) IDIV 带符号数除法

格式：　　IDIV　OPS

执行的操作:与 DIV 相同,但操作数必须是带符号数,商和余数也都是带符号数,且余数的符号和被除数的符号相同。

例如：　　IDIV　BL

若指令执行前　　AX = 0400H, BL = 0B4H;

则指令执行后　　AL = 0F3H(商), AH = 24H(余数)。

除法指令的寻址方式和乘法指令相同。其目的操作数必须存放在 AX、DX:AX 或 EDX:EAX 中;而其源操作数可以是除立即数以外的任一种寻址方式。

除法指令对所有条件码位均无定义。

5. 类型转换指令

由于除法指令的字节操作要求被除数为16位,字操作要求被除数为32位,双字操作要求被除数为64位。然而,有些被除数的操作数不能满足格式要求。因此,有时在进行除法时,需要调整被除数,使被除数成为除法指令所需要的操作数格式。

当进行无符号数的除法时,通常将其高位补0。

当进行带符号数的除法时,须用类型转换指令。类型转换指令有:

CBW	字节转换为字
CWD/CWDE	字转换为双字
CDQ	双字转换为4字
BSWAP	字节交换

(1) CBW 字节转换为字指令

格式: CBW

执行的操作:AL的内容符号扩展到AH,形成AX中的字。即如果AL的最高有效位为0,则AH=0;如果AL的最高有效位为1,则AH=0FFH。

例如: CBW

若指令执行前 AL=98H;

则指令执行后 AX=0FF98H。

(2) CWD/CWDE 字转换为双字指令

格式: CWD

执行的操作:AX的内容符号扩展到DX,形成DX:AX中的双字。即如果AX的最高有效位为0,则DX=0;如果AX的最高有效位为1,则DX=0FFFFH。

格式: CWDE

执行的操作:AX的内容符号扩展到EAX的高16位,形成EAX中的双字。

例如: CWD

若指令执行前 AX=1234H;

则指令执行后 AX=1234H,DX=0000H。

(3) CDQ 双字转换为4字指令

格式: CDQ

执行的操作:EAX的内容符号扩展到EDX,形成EDX:EAX中的4字。

例如: CDQ

若指令执行前 EAX=56781234H;

则指令执行后 EDX=00000000H,EAX=56781234H。

(4) BSWAP 字节交换

格式：　　BSWAP　R32

执行的操作：使指令指定的 32 位寄存器的字节次序变反。具体操作：第 1 与第 4 字节互换，第 2 与第 3 字节互换。

该指令只能用于 486 及以上机型，R32 指 32 位寄存器。

例如：　　BSWAP　EAX

若指令执行前　　EAX = 56781234H;

则指令执行后　　EAX = 34127856H。

6. 十进制调整指令

前面介绍的所有算术运算指令都是二进制数的算术运算指令，但是人们最常用的是十进制数。这样，当计算机进行计算时，必须先把十进制数转换成二进制数进行计算，计算结果再转换成十进制数输出。为了便于十进制数的计算，在 8086/8088 及后继的机型中还提供了一组十进制数调整指令。这组指令在二进制计算的基础上给予十进制调整，可直接得到十进制数的结果。十进制调整指令有：

DAA	加法的压缩 BCD 码十进制调整指令
DAS	减法的压缩 BCD 码十进制调整指令
AAA	加法的非压缩 BCD 码十进制调整指令
AAS	减法的非压缩 BCD 码十进制调整指令
AAM	乘法的非压缩 BCD 码十进制调整指令
AAD	除法的非压缩 BCD 码十进制调整指令

(1) DAA 加法的压缩 BCD 码十进制调整指令

格式：　　DAA

执行的操作：把 AL 中的两个压缩 BCD 码之和调整成压缩 BCD 码的格式→AL。

注意，这条指令之前必须执行 ADD 或 ADC 指令。加法指令必须把两个压缩 BCD 码相加，并把相加结果存放在 AL 寄存器中。本指令的调整方法是：

如果 AF 标志（辅助进位位）为 1，或者 AL 寄存器的低 4 位是 16 进制的 A～F 中的任意一位数，则 AL 寄存器内容加 06H 修正，且将 AF 位置 1；

如果 CF 标志为 1，或者 AL 寄存器的高 4 位是 A～F 中的任意一位数，则 AL 寄存器内容加 60H 修正，并将 CF 位置 1；

如果 AL 寄存器的高 4 位和低 4 位都满足以上条件，则将 AL 寄存器的内容加 66H。

DAA 指令对 OF 标志无定义，但影响其他条件标志。

例如：　　DAA

若指令执行前　　AL＝3AH，AF＝0，CF＝0；

则指令执行后　　AL＝40H，AF＝1，CF＝0。

如：68H+89H=F1
执行DAA：F1H+66H=157H
因为低4位有进位，高4位大于9，所以加66H，结果就好象68+89按十进制计算一样！

低位大于9，所以加06H

(2) DAS 减法的压缩 BCD 码十进制调整指令

格式：　　DAS

执行的操作：把 AL 中的两个压缩 BCD 码之差调整成压缩 BCD 码的格式→AL。

注意，在执行这条指令之前，必须先执行 SUB 或 SBB 指令，减法指令必须把两个压缩 BCD 码相减，并把结果存放在 AL 寄存器中。本指令的调整方法是：

如果 AF 标志为 1，或者 AL 寄存器的低 4 位是 16 进制的 A～F 中的任意一位数，则 AL 寄存器的内容减 06H，且将 AF 位置 1；

如果 CF 标志为 1，或者 AL 寄存器的高 4 位是 16 进制的 A～F 中的任意一位数，则 AL 寄存器内容减 60H，并将 CF 位置 1。

如果 AL 寄存器的高 4 位和低 4 位都满足以上条件，则将 AL 寄存器的内容减 66H。

DAS 指令对 OF 标志无定义，但影响其他条件标志。

例如：　　DAS

因为AF=1，低位减06H

若指令执行前　　AL＝49H，AF＝1，CF＝0；

则指令执行后　　AL＝43H，AF＝1，CF＝0。

如：86H-07H=86H+F9H =7FH, CF=0
低位大于9，所以执行DAS要减06H，即：7FH-06H=79H
结果就好象86-07按十进制计算一样！

5.2.2 处理机控制指令

在前面介绍的指令中，都是对某一个数或两个数进行操作，产生相应的结果，然后根据运算的结果影响标志位。而在80X86 CPU的指令系统中，还提供了一组指令，它们只是对标志寄存器(PSW)的标志位置1、清0，或者是对处理机的状态进行控制。

1. 标志处理指令

CLC；进位位清0指令，0→CF

CMC；进位位求反指令，$\overline{CF}$→CF

STC；进位位置1指令，1→CF

CLD；方向位清0指令，0→DF

STD；方向位置1指令，1→DF

CLI；中断标志位清0指令，0→IF

STI；中断标志位置1指令，1→IF

2. 其他处理机控制与杂项操作指令

NOP　空操作

HLT　停机

ESC　换码

WAIT　等待

LOCK　封锁

BOUND　界限

（1）NOP　空操作指令

指令格式：NOP

该指令不做任何操作，机器码占有一个字节单元。其作用是在调试程序时往往用这一条指令占有一定的存储单元，以便在正式的程序中用其他的指令来取代它。它的另一个作用是在延时程序中作为延时时间的调节。

（2）HLT　停机指令

指令格式：HLT

该指令可以使机器暂时停止工作，使处理机处于动态停机的状态，等待外部设备的中断请求。如果外部设备没有中断请求，则处理机继续等待；如果外部设备有中断请求，则处理机转移到中断处理程序执行中断处理。当中断处理程序执行完之后，程序将继续执行停机指令的下一条指令。

（3）ESC　换码指令

指令格式：ESC　OP，R/M

在计算机的硬件系统中，如果使用了协处理器8087（或80287、80387等）进行浮点运算，该指令指定协处理器接收指令和数据。协处理器接收的第一个数据为操作码（OP），第二个数据为操作数（寄存器R或存储器M中的内容）。并且此时CPU将控制权交给协处理器进行控制。自80486 CPU之后的机型中，由于浮点处理部件已经封装在CPU芯片中，系统可以直接支持协处理器指令。因此，ESC指令是一条未定义的指令。如果在程序中遇到ESC指令，将引起一次异常中断处理。

（4）WAIT　等待指令

指令格式：WAIT

该指令使CPU处于空转等待状态，它可以用来等待外部的中断请求，当外部的中断请求处理完之后，仍然返回到WAIT指令继续等待；它也可以用来等待协处理器的操作，直到协处理器完成其指令的操作为止。

（5）LOCK　封锁指令

指令格式：LOCK　联合使用的指令

该指令是一条前缀指令，它可以与其他指令联合使用。在执行这一条指令时，它可以使CPU总线的LOCK信号在执行联合指令期间维持有效。与LOCK前缀指令联合使用的指令有：

① ADD/ADC/SUB/SBB/AND/OR/XOR

5．2．3 顺序程序设计方法

顺序程序设计是程序设计中最简单的一种程序设计方法。从程序的流程图看，顺序程序是以一个开始框开始，以一个结束框终止，中间有一个或多个顺序的执行框组成的程序结构形式。只要遵照算法步骤依次写出相应的指令即可。这种程序设计方法称为直流方法。

例5.1　设X、Y、Z为有符号字变量，编写程序计算R=（（X ＊Y+5）+4 ＊X）/Z 表达式　　。

数据段定义如下：

```
DATA  SEGMENT
X           DW    -45
Y           DW    70
Z           DW    -12
R           DW    2   DUP（0）
DATA  ENDS
```


例5.1　程序　　R=（（X ＊Y+5）+4 ＊X）/Z

```
CODE    SEGMENT
        ASSUME  CS：CODE，DS：DATA
START：MOV  AX，DATA
        MOV  DS，AX
        MOV  AX，X
        IMUL Y              ；计算X＊Y
        ADD  AX，5          ；计算X＊Y+5
        ADC   DX，0
        MOV  CX，DX
        MOV  BX，AX
        MOV  AX，X          ；计算4＊X
```

例5.1

```
        MOV   SI，4
        IMUL  SI
        ADD   AX，BX        ；算（X＊Y+5）+4＊X
        ADC   DX，CX
        IDIV  Z             ；计算（（X＊Y+5）+4＊X）/Z
        MOV   R，AX
        MOV   R+2，DX
        MOV   AH，4CH
        INT   21H
CODE    ENDS
        END   START                    例5.2  （不看）
```


例5.3

将BUF中的一字节压缩BCD码转换为ASCII码显示输出。

程序清单如下：

```
DATA    SEGMENT
BUF     DB      34H
DATA    ENDS
CODE    SEGMENT
        ASSUME      CS：CODE，DS：DATA
START：         MOV    AX，DATA
                MOV    DS，AX
                MOV    DL，BUF
                MOV    CL，4
                SHR    DL，CL
```

例5.3

```
        ADD     DL，30H
        MOV     AH，2
        INT     21H
        MOV     DL，BUF
        AND     DL，0FH
        ADD     DL，30H
        MOV     AH，2
        INT     21H
        MOV     AH，4CH
        INT     21H
CODE    ENDS
        END     START
```


例5.4

■设A、B变量中存放的是64位的二进制数，请用386及其后继机型的相应指令编写A＋B→C的程序。

```
. 586
STACK SEGMENT       STACK
DW    200 DUP（0）
STACK ENDS
DATA  SEGMENT       USE16
A     DQ    372869AF63DE3710H
B     DQ    1239876ABCFDE020H
C     DQ    0
DATA  ENDS
CODE  SEGMENT   USE16
      ASSUME    CS：CODE，DS：DATA，SS：STACK
```

例5.4

```
START:  MOV   AX, DATA
        MOV   DS, AX
        MOV   AX, STACK
        MOV   SS, AX
        MOV   EAX, DWORD  PTR  A
        ADD   EAX, DWORD   PTR  B
        MOV   DWORD   PTR  C, EAX
        MOV   EAX, DWORD  PTR  A+4
        ADC   EAX, DWORD   PTR  B+4
        MOV   DWORD   PTR  C+4 , EAX
        MOV   AX, 4C00H                ; 返回DOS
21      INT   21H
CODE    ENDS                  END   START
```


作业：5.1 、5.2

5．3 分支程序设计

根据不同情况做出判断，有选择地执行相应的处理程序。通常称这类程序为分支程序。实现这类程序的设计过程称为分支程序设计。

在分支程序中，不同的条件往往是通过标志寄存器中条件标志的不同状态反映的。因而，分支程序设计中一个至关重要的问题是根据各标志的不同状态选用合适的转移指令。

5．3．1 转移指令

1．简单的条件转移指令

1） JZ（或JE） 结果为零（或相等）则转移； 测试条件：ZF=1

2）JNZ（或 JNE）结果不为零（或不相等）则转移；
测试条件：ZF=0

3） JS 结果为负则转移 测试条件：SF=1

4） JNS 结果为正则转移 测试条件：SF=0

5） JO 结果溢出则转移； 测试条件：OF=1

6） JNO 结果不溢出则转移; 测试条件：OF=0

7） JP（或JPE） 奇偶位为1则转移； 测试条件：PF=1

8） JNP（或JPO）奇偶位为0则转移； 测试条件：PF=0

9） JC 进位则转移； 测试条件：CF=1

10）JNC 没有进位则转移； 测试条件：CF=0

2. 无符号数比较条件转移指令

1） JB（或 JNAE） 低于或者不高于等于则转移;
测试条件：CF=1且ZF=0

2） JNB（或JAE） 不低于或者高于等于则转移;
测试条件：CF=0或ZF=1

3） JA（或JNBE） 高于或者不低于等于则转移;
测试条件： CF=0且 ZF=0

4） JNA（或JBE）不高于或者低于等于则转移;
测试条件：CF=1或 ZF= 1

记忆方法：A：大于；B小于；N：否定

3. 有符号数比较条件转移指令

1） JL（或 JNGE） 小于或者不大于等于则转移;
测试条件： SF ≠ OF∧ZF=0

2） JNL（或 JGE）不小于或者大于等于则转移;
测试条件： SF ≠ OF∨ZF=1

3） JG（或 JNLE） 大于或者不小于等于则转移;
测试条件：SF = OF∧ZF=0

4） JNG（或 JLE） 不大于或者不小于等于则转移
测试条件：SF = OF∨ ZF=1

记忆方法：G：大于；L小于；N：否定

4．测试CX或ECX为0则转移指令 （不看）

1） JCXZ CX寄存器的内容为0则转移指令
测试条件：CX=0

2） JECXZ ECX寄存器的内容为0则转移指令
测试条件：ECX=0

5．循环指令

1） **** LOOP 当计数器的值不为0时循环指令
测试条件：CX－1→CX≠0 或 ECX－1→ ECX≠0

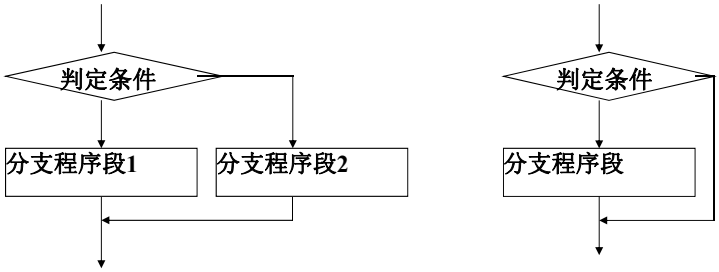
以下2条很少使用，可以不看,重点看上面的LOOP指令！

2） LOOPZ / LOOPE 计数器的值不为0或相等时循环指令
测试条件： ZF=1且CX－1→CX≠0 或 ZF=1且ECX－1→ ECX≠0

3）LOOPNZ / LOOPNE 当计数器的值不为0或不相等时循环指令
测试条件： ZF=0且CX－1→CX≠0
或 ZF=0且 ECX－1→ ECX≠0

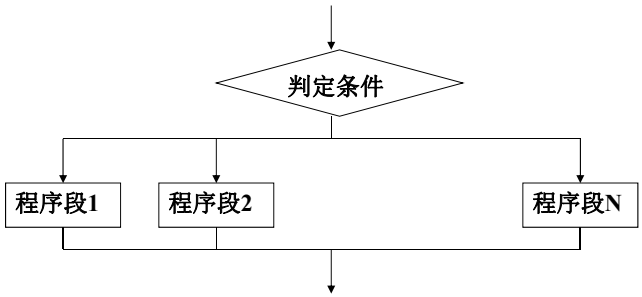
5．3．2 分支程序的结构形式

第一种形式称为双分支程序结构；



5．3．2 分支程序的结构形式

第二种形式称为多分支程序结构；



5．3．3 分支程序设计方法

1. 双分支程序设计

例5.5　设有三个单字节无符号数存放在BUF开始的缓冲区中，编写一个能将它们按大到小重新排列的程序。

由于BUF缓冲区中只有三个数据，有多种方法可实现三个数据的排序。我们采用交换法，找到三个数中的最大数，然后再找到剩下两个数的大数，最后将三个数按大小依次存放。为了方便，先把要排序的三个数取到三个寄存器中，然后再对三个数据进行比较排序。

例5.5

```
DATA   SEGMENT
BUF    DB  87，234，123
DATA        ENDS
CODE   SEGMENT
       ASSUME   CS：CODE，DS：DATA
START：        MOV    AX，DATA
               MOV    DS，AX
             MOV      SI，OFFSET BUF  ；（或  LEA SI, BUF）
             MOV      AL，［SI］
             MOV      BL，［SI＋1］
             MOV   CL，［SI＋2］
             CMP      AL，BL
             JAE      NEXT1
             XCHG     AL，BL
```


例5.5

```
NEXT1：CMP      AL，CL
       JAE      NEXT2
       XCHG     AL，CL
NEXT2：CMP      BL，CL
       JAE      NEXT3
       XCHG     BL，CL
NEXT3：MOV      [SI]，AL
       MOV      [SI＋1]，BL
       MOV      [SI＋2]，CL
       MOV      AH，4CH
       INT      21H
CODES    ENDS
         END      START
```


例5.6

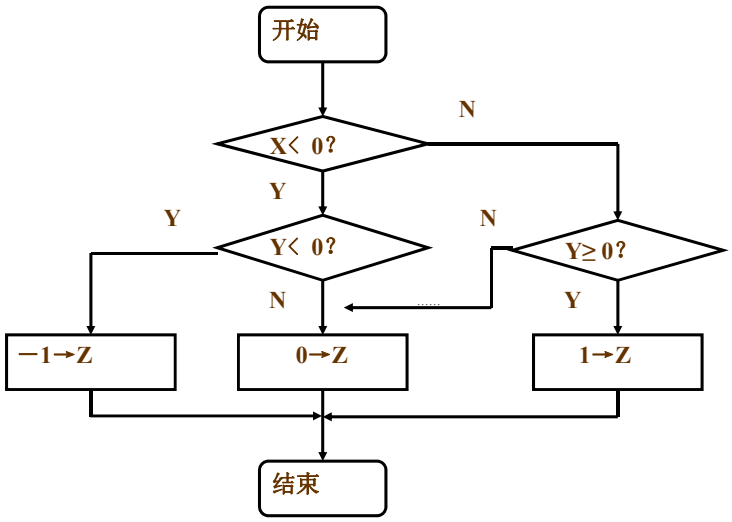
编写计算下面函数值的程序（x、y的值均在—128～＋127之间）。

$$Z=\begin{cases}1 & x\geq 0，y\geq 0\\ -1 & x<0，y<0\\ 0 & x、y异号\end{cases}$$

输入数据为x、y，结果数据为Z。

存储单元分配如下：变量X中存放x的值，变量Y中存放y的值，变量Z用来存放函数值。以上各变量均为字节类型。程序流程图如下：

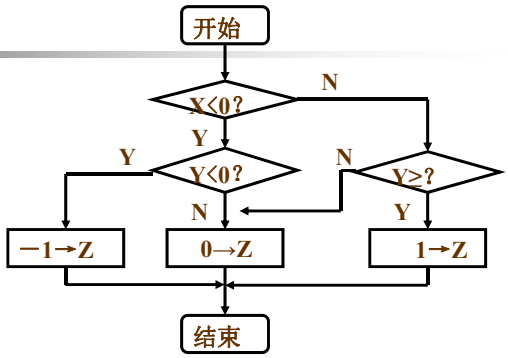
例5. 6



例5. 6

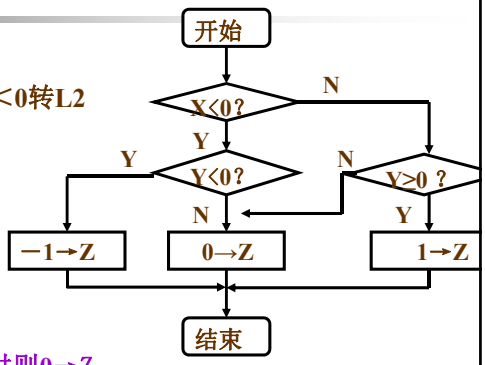
程序清单如下：

```
DATA        SEGMENT
    X       DB    23
    Y       DB    −10
    Z       DB    0
    DATA    ENDS
CODE        SEGMENT
ASSUME  CS：CODE，DS：DATA
START：     MOV AX，DATA
            MOV  DS，AX
            CMP   X，0
            JS      L1
```



例5.6

```
            CMP  Y，0
            JL L2              ；y＜0转L2
            MOV Z，1
            JMP    EXIT
L1：    CMP  Y，0   ;y≥0转L2
            JGE        L2
           MOV  Z，－1
           JMP EXIT
L2：    MOV  Z，0  ；x、y异号时则0→Z
EXIT：  MOV        AH，4CH
           INT 21H
CODE    ENDS
              END    START
```



例5.7

从键盘输入0～9中任一自然数X，若输入的字符是0～9中的某一数，则将X的立方值送Y；若输入的字符不是0～9中的某一数，则显示“INPUT ERROR！”，表明输入错。

程序清单如下：

```
STACK SEGMENT     STACK
                  DB     100 DUP（0）
STACK ENDS
DATA  SEGMENT
INPUT DB   ‘PLEASE INPUTX（0～9）：$’
TAB    DW     0，1，8，27，64，125，216，343，512，729
X              DB     ?
Y              DW     0
ERR            DB     0DH，0AH，“INPUT ERROR！$”
DATA  ENDS
```

例5. 7

```
CODE        SEGMENT
        ASSUME  CS：CODE，DS：DATA，SS：STACK
START：     MOV  AX，DATA
            MOV  DS，AX
            LEA   DX，INPUT
            MOV  AH，9
            INT    21H
            MOV  AH，1
            INT    21
            CMP  AL，  ‘0’
            JB     CHR
            CMP   AL，  ‘9’
             JA     CHR
```


例5. 7

```
            AND    AL，0FH
            MOV    X，AL
            SHL     AL，1
            MOV    BL，AL
             MOV    BH，0
             MOV    AX，TAB[BX]
             MOV    Y，AX
EXIT：MOV AH，4CH
            INT      21H
CHR：LEA   DX，ERR；显示“INPUT ERROR！”
             MOV    AH，9
           INT     21H
             JMP    EXIT
CODE   ENDS
          END     START
```


例5. 7

从键盘输入0～9中任一自然数X，若输入的字符是0～9中的某一数，则将X的立方值送Y；若输入的字符不是0～9中的某一数，则显示“INPUT ERROR！”，表明输入错。

程序清单如下：

```
STACK SEGMENT     STACK
                  DB     100 DUP（0）
STACK ENDS
DATA  SEGMENT
INPUT DB   ‘PLEASE INPUTX（0～9）：$’
X                 DB     ?
F                 DW     0
DATA  ENDS
```


例5. 7

```
CODE    SEGMENT
        ASSUME  CS：CODE，DS：DATA，SS：STACK
START：MOV      AX，DATA
                                MOV         DS，AX
          LEA   DX，INPUT
          MOV   AH，9                                          INT
          21H
          MOV   AH，1
          INT   21
          CMP   AL，‘0’
          JB    L1
          CMP   AL，‘9’
          JA    L1
          MOV   F, 1
L1:       CMP   AL，‘A’
          JB    L2
          CMP   AL，‘Z’
          JA    L2
```


例5. 7

```
        AND    AL，0FH
        MOV    X，AL
        SHL    AL，1
        MOV    BL，AL
        MOV    BH，0
        MOV    AX，TAB[BX]
        MOV    Y，AX
EXIT：MOV AH，4CH
        INT    21H
CHR：LEA   DX，ERR；显示“INPUT ERROR！”
        MOV    AH，9
       INT    21H
        JMP    EXIT
CODE   ENDS
       END    START
```

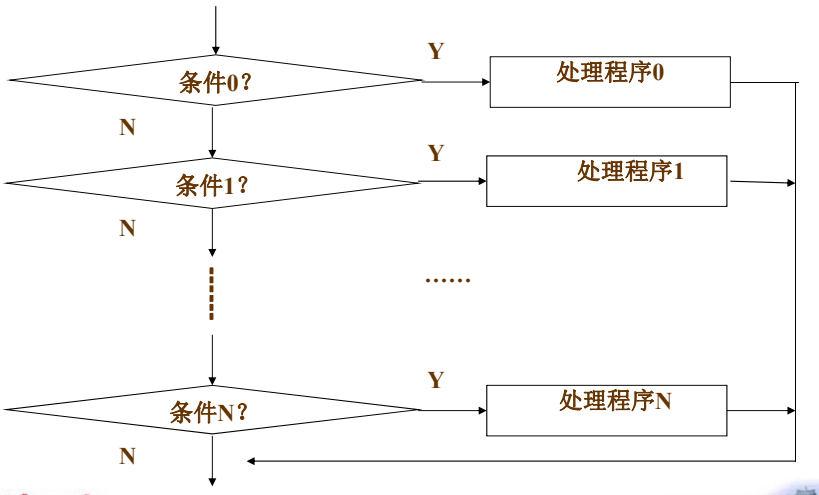

2．多分支程序设计

多分支程序结构框图如图5．3所示。从图中可以看出，设计多分支程序的关键是如何按条件对多分支进行判断，从而根据不同的条件转移到不同的入口去执行各自的程序段。

下面结合具体示例介绍多分支结构程序设计的三种基本方法：逻辑分解法、地址表法和转移表法。

1）逻辑分解法多分支程序设计

程序流程图如图5．5所示。



例5. 8

如果X字节变量中是处理程序的序号（0～9），P0、P1、P2、… 、P9是序号（0～9）的对应的处理程序。试编写一程序，根据X中值的不同，分别转移到相应的处理程序。

```
DATA   SEGMENT
  X      DB      ?
DATA   ENDS
CODE   SEGMENT
                ASSUME   CS：CODE，DS：DATA
START：         MOV    AX，DATA
                MOV    DS，AX
                MOV    AL，X
                CMP    AL，0          ；如果（X）=0则转P0
                JNZ    L1
                JMP    P0
```

例5.8

```
L1:     CMP  AL，1      ；如果（X）=1则转P1
        JNZ  L2
        JMP  P1
L2:     CMP  AL，2      ；如果（X）=2则转P2
        JNZ  L3
        JMP  P2
        ┆
L9:     CMP  AL，9      ；如果（X）=9则转P9
        JNZ  L10
        JMP  P9
L10:    JMP   EXIT
```


例5.8

```
P0:  …
        ┆
P1:  …
        ┆
P9:  …
        ┆
EXIT:   MOV  AH，4CH    ；返回DOS
        INT   21H
 CODE   ENDS
          END    START
```

2）转移表法多分支程序设计

逻辑分解法需用多条比较判断转移指令才能实现多分支的程序设计。这样使得编写的程序太长。如果将转移各分支程序段的转移指令依次存放在一张表中，这张表称为转移表。各分支转移指令在表中的位置，即离表首地址的位移量作为条件，当进行多分支条件判断时，把当前条件的位移量加上表首地址作为转移地址，转移到表的相应位置，继续执行无条件转移指令，达到多分支的目的。这种程序设计的方法称为转移表法多分支程序设计。

例5.9

编写程序，根据键盘输入的值，若输入数字0则显示“INPUT DIGIT 0！”，若输入数字1则显示“INPUT DIGIT 1！”，...，若输入数字9则显示“INPUT DIGIT 9！”；否则显示“INPUT CHARACTER！”。

```
DATA   SEGMENT
DIS0   DB     'INPUT    DIGIT     0! $'
DIS1   DB     'INPUT    DIGIT     1! $'
        ⋮
DIS9   DB     'INPUT    DIGIT     9! $'
DIS10  DB     'INPUT    CHARACTER! $'
DATA   ENDS
```

例5.9

```
CODE    SEGMENT
        ASSUME   CS：CODE，DS：DATA
START：  MOV   AX，DATA
        MOV   DS，AX
        MOV   AH，1        ；调用键盘输入DOS中断
        INT   21H
        CMP   AL，30H      ；如果输入的是非数字则转M
        JB    M
        CMP   AL，39H
        JA    M
        AND   AL，0FH
         JMP  N ；
M：      MOV   AL，10
```


例5.9

```
N：      LEA   BX，TAB     ；
        MOV   AH，0
        SHL   AL，1
        ADD   BX，AX
        JMP   BX
EXIT：   MOV   AH，4CH
        INT   21H
TAB：    JMP   SHORT   P0   ；转移地址表
        JMP   SHORT   P1
        JMP   SHORT   P2
        ⋮
        JMP   SHORT   P9
        JMP   SHORT   P10
```

例5.9

```
P0:   LEA   DX，DIS0；显示输出字符串INPUT   DIGIT   0!
      MOV   AH，9
      INT   21H
      JMP   EXIT
P1:   LEA   DX，DIS1；显示输出字符串INPUT   DIGIT   1!
      MOV   AH，9
      INT   21H
      JMP   EXIT
      ⋮
P10:  LEA   DX，DIS10；显示输出字符串INPUT  CHARACTER!
      MOV   AH，9
      INT   21H
      JMP   EXIT    CODE  ENDS      END  START
```


3）地址表法多分支程序设计

如果将每个分支处理程序的入口地址存放一个地址表中，然后根据不同的条件，分别从地址表中取出对应分支处理程序的入口地址，转移到相应的分支执行程序。按例9的要求，可编写程序如下：

```
DATA    SEGMENT
DIS0    DB      'INPUT   DIGIT    0! $'
DIS1    DB      'INPUT   DIGIT    1! $'
DIS2    DB      'INPUT   DIGIT    2! $'
        ⋮
DIS9    DB      'INPUT   DIGIT    9! $'
DIS10   DB      'INPUT   CHARACTER! $'
TAB     DW        P0, P1, P2, P3, P4, P5, P6, P7, P8, P9, P10
DATA    ENDS
```

3）地址表法多分支程序设计

```
          CODE SEGMENT
          ASSUME    CS：CODE，DS：DATA
START：   MOV   AX，DATA
          MOV   DS，AX
          MOV   AH，1
          INT   21H
          CMP   AL，30H
          JB    M
          CMP   AL，39H
          JA    M
          AND   AL，0FH
          JMP   N；
```


3）地址表法多分支程序设计

```
M：       MOV   AL，10
N：       LEA   BX，TAB
          MOV   AH，0
          SHL   AL，1
          ADD   BX，AX
          JMP   WORD PTR [BX]
EXIT：    MOV   AH，4CH
          INT   21H
```


3）地址表法多分支程序设计

```
P0:   LEA   DX, DIS0
      MOV   AH, 9
      INT   21H
      JMP   EXIT
P1:   LEA   DX, DIS1
      MOV   AH, 9
      INT   21H
      JMP   EXIT
      ┆
P10: LEA    DX, DIS10
      MOV   AH, 9
      INT   21H
      JMP   EXIT
CODE ENDS
      END START
```


5．4 循 环 程 序 设 计

如果对多次重复执行的语句由程序控制执行，这类程序设计称为循环程序设计。

5．4．1 循环程序的结构

循环程序一般由四部分组成：置初值部分、工作部分、修改部分和控制部分。

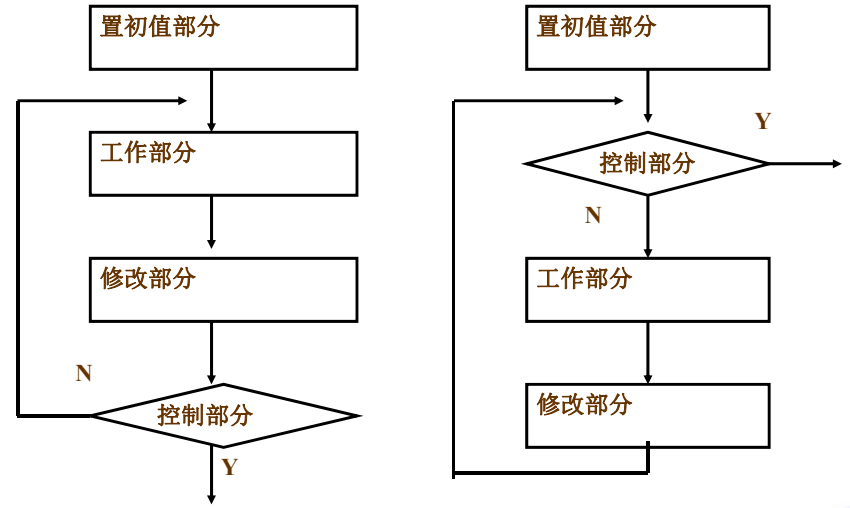
置初值部分：为了保证循环程序能正常进行循环操作而必须做的准备工作。循环初值分两类，一类是为循环体的循环次数设置的计数初值，另一类是为工作部分、修改部分正常工作设置的初值。

工作部分：即需要重复执行的程序段，这是循环程序的核心，称之为循环体。

修改部分：按一定规律修改操作数地址及控制变量，以使每次执行循环体时得到新的数据。

控制部分：用来保证循环程序按一定的次数或特定条件正常循环。

5．4．1 循环程序结构



5．4．2 循环控制方法

1． 计数控制法（循环次数已知的循环控制方法）

假设循环次数为N，常常用以下三种计数控制方法实现循环次数的控制：

1）

```
MOV  CX，N              ；置初值部分
 ┆
L:    …                 ；工作部分
 ┆                ；修改部分
DEC   CX          ；控制部分    } = LOOP L
JNZ   L
```

1. 计数控制法

```
2）MOV      CX，－N      ；置初值部分
 ┆
L：  …                  ；工作部分
 ┆                      ；修改部分
INC   CX                ；控制部分
CMP   CX, 0
JNZ   L

3）MOV      CX，0        ；置初值部分
 ┆
L：  …                  ；工作部分
 ┆                      ；修改部分
INC   CX                ；控制部分
CMP   CX，N
JNZ   L
```


2. 条件控制法

有些情况下，循环次数无法事先确定，但它与问题中的某些条件有关。这些条件可以通过指令来测试。若测试比较的结果表明满足循环条件，则继续循环，否则结束循环。

2．条件控制法

例如： 统计AX寄存器中为1的位数，并将统计的结果存放在CL寄存器中。

如果要解决这一问题，可以采用计数控制法控制循环，其程序段如下：

```
        MOV  CL，0        ；置循环的初值
        MOV  CX，16
L：      SAL  AX，1
        JNC   N
        INC   CL
    N：  LOOP  L

EXIT：…
```


2．条件控制法

无论AX中内容为何值，程序都要循环16次。显然，解决这一问题时，此种方法并非是一种好方法。如果采用条件控制法编写程序，其程序段如下：

```
        MOV  CL，0
    L：  AND   AX，AX；  CMP  AX，0
        JZ     EXIT      ；AX=0时，结束循环转EXIT
        SAL    AX，1     ；将AX中的最高位移入CF中
        JNC    L         ；如果 CF=0则转L
        INC    CL        ；如果 CF=1则CL+1→CL
        JMP    L         ；转L处继续循环
EXIT：…
```


5．4．3　单重循环程序设计

所谓单重循环，即其循环体内不再包含循环结构。

1．循环次数已知的循环程序设计

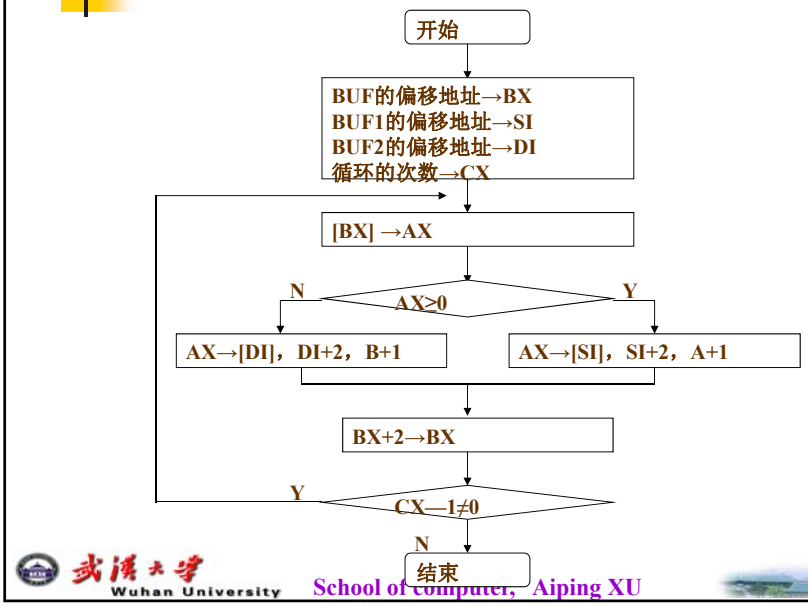
对于循环次数已知的情况，通常采用计数控制方法实现循环。

例5.10　已知以BUF为首地址的字存储区中存放着N个有符号二进制数，试编写程序将其中大于等于0的数依次送BUF1为首地址的字存储区中，小于0的数依次送以BUF2为首地址的字存储区中。同时将大于等于0的数的个数送A字变量，将小于0的个数送B字变量。

例5. 10　程序清单

```
DATA SEGMENT
BUF   DW   23，123，–12，–210，45，0，90，–453
N          =      （$–BUF）/2
BUF1  DW   N  DUP（0）
BUF2  DW   N  DUP（0）
A          DW   0
B          DW   0
DATA ENDS
CODE SEGMENT
           ASSUME  CS：CODE，DS：DATA
START：    MOV  AX，DATA
           MOV  DS，AX
```


例5.10　程序框图



例5.10　程序清单

```
        LEA   BX，BUF     ；置循环初值
        LEA   SI，BUF1
        LEA   DI，BUF2
        MOV   A，0
        MOV   B，0
        MOV   CX，N
L0：    MOV   AX，[BX]
        CMP   AX，0
        JGE   L1
        MOV   [DI]，AX
        ADD   DI，2
        INC   B
        JMP   NEXT
```

例5.10 程序清单

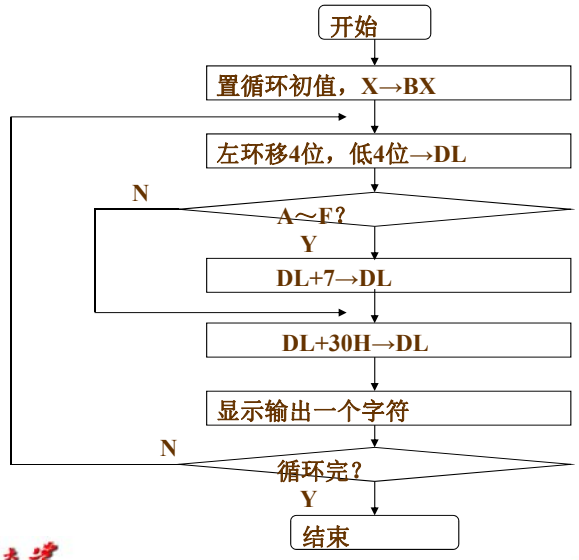
```
     L1:  MOV  [SI]，AX
          ADD  SI，2
          INC  A
NEXT:     ADD  BX，2          ；BUF的地址修改
          LOOP L0             ；循环次数修改，未完则转L0
          MOV  AH，4CH
          INT  21H
CODE      ENDS
          END  START
```


例5.11 *****

试编制一个程序，把字变量X中的16位二进制数用十六进制数的形式在屏幕上显示出来。

这里采用了循环移位的方法把所要显示的4位二进制数移到最右边（最低4位），以便作数字到字符的转换工作。另外，由于数字0～9的 ASCII为30H～39H，如果要将数字转换为ASCII码只需加30H；而字母 A～F的ASCII码为41H～46H，如果要将字母转换为对应的ASCII码则需加37H。所以在把4位二进制数转换为ASCII码的时候，需要判断4位二进制所表示的是0～9还是A～F。在转换的过程中，首先将4位二进制数加30H，然后再作一次判断，如果为数字，则此时的值就是0～9的ASCII码；如果为字符A～F，则还应加上7才是A～F的ASCII码。只有将4位二进制数转换为ASCII码后才能显示出正确的十六进制数。

例5. 11　　程序框图



例5. 11　　程序清单

```
DATA SEGMENT
X          DW    4F59H
DATA ENDS
CODE SEGMENT
           ASSUME  CS：CODE，DS：DATA
START：    MOV  AX，DATA
           MOV  DS，AX
           MOV  CH，4              ；置循环初值
           MOV  BX，X
L：        MOV  CL，4        ；BX中的内容左环移4位
           ROL  BX，CL
           MOV  DL，BL       ；低4位送DL
           AND  DL，0FH      ；DL的低4位转换为ASCII码
```

例5. 11　程序清单

```
        CMP   DL，10
        JB    NEXT
        ADD   DL，7
NEXT:   ADD   DL，30H
        MOV   AH，2              ；输出显示
        INT   21H
        DEC   CH
        JNZ   L                  ；循环次数修改，未完则转L

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


例5. 12

设A变量中存放着10个双字长的二进制数，试用386及其后继机型的相应指令编写将A送B的程序。

在进行32位数据传送的程序设计过程中，首先应该使用32位的数据传送指令。当数据传送后，为了使地址指针指向下一数据，需对地址进行修改。由于一个32位的数据占用4字节，在做地址修改时，需将地址加4。

```
 .586
DATA SEGMENT   USE16
A              DD    10 DUP（？）
B              DD    10 DUP（？）
DATA ENDS
CODE SEGMENT   USE16
               ASSUME  CS：CODE，DS：DATA
```

例5. 12

```
START：   MOV   AX，DATA
          MOV  DS，AX
          MOV  CX，10              ；置循环初值
          LEA   ESI，A
          LEA   EDI，B
       L：MOV  EAX，[ESI]  ；传送一个双字长的二进制数
          MOV  [EDI]，EAX
          ADD   ESI，4             ；地址修改
          ADD   EDI，4
          LOOP L              ；循环控制
          MOV  AX，4C00H
          INT   21H
CODE      ENDS
          END   START
```


例5. 13

例5.13　设A、B变量中分别存放着10个双字长的多精度二进制数，其中低地址存放的是低位数，高地址存放的是高位数。试用386及其后继机型的相应指令编写将A＋B送C的程序。

```
.586
DATA SEGMENT   USE16
A              DD    10 DUP （？）
B              DD    10 DUP （？）
C              DD    11  DUP（0）
DATA ENDS
CODE SEGMENT   USE16
               ASSUME CS：CODE，DS：DATA
START：    MOV    AX，DATA
           MOV    DS，AX
```

例5. 13

```
        CLC                 ；cf=0 ，置循环初值
        MOV  CX，10
        LEA  ESI，A
        LEA  EDI，B
        LEA  EBX，C
L：     MOV  EAX，[ESI]  ；两个双字长的二进制数相加
        ADC  EAX，[EDI]
        MOV  [EBX]，EAX；结果送目的单元
        ADD  ESI，4              ；地址修改
        ADD  EDI，4
        ADD  EBX，4
        LOOP L              ；循环控制
```


例5. 13

```
        MOV  EAX，0       ；最高进位的处理
        ADC  EAX，0
        MOV  [EBX]，EAX

        MOV  AX，4C00H
        INT  21H
CODE    ENDS
        END  START
```

2. 循环次数未知的循环程序设计

例5.14 设STR字符串是以0结尾。试编写一个把字符串中的所有大写字母改为小写字母的程序，并将转换后的字符串显示输出。

由于字符串是以0结尾的，所以字符串的长度是一个未知数，它的循环次数是不确定，需根据字符串尾这个条件来控制程序的循环。

如果是大写字母，需将对应字母的ASCII码加20H；如果是其他字符，则其字符保持不变。编写的程序清单如下：

```
DATA    SEGMENT
  STR DB     'HOW   arE   YoU!' , 0；假设的字符串
DATA  ENDS
CODE  SEGMENT
            ASSUME    CS：CODE，DS：DATA
```


例5. 14

```
START：  MOV  AX，DATA
         MOV  DS，AX
         MOV  SI，OFFSET STR   ；LEA  SI, STR
AGAIN：  MOV  DL，[SI]         ；取一字符
         OR   DL，DL           ；是否到字符串尾？
         JZ   OK               ；到字符串尾，转OK
         CMP  DL，'A'   ；否则，判是否为大写字母
         JB   NEXT             ；否，转继续
         CMP  DL，'Z'
         JA   NEXT             ；否，转继续
         ADD  DL，20H    ；是大写字母，改为小写字母
         MOV  [SI]，DL         ；送回到字符串中
```

例5. 14

```
NEXT:   MOV     AH，2
        INT     21H
        INC     SI                  ；调整指针
        JMP     AGAIN                   ；继续
OK:     MOV   AH，4CH
        INT     21H
CODE    ENDS
        END     START
```

例5. 15（ 不看）

例5. 16

例5.16　设ARY中存放着以0作为结束的双字数组。请用386及其后继机型的相应指令编写出统计数组中大于0和小于0元素的个数，并将统计的个数分别存放在P和N变量中的程序。

```
.586
STACK       SEGMENT   USE16  STACK
DW    200 DUP （0）
STACK       ENDS
DATA SEGMENT   USE16
ARY   DD     1234,456,789,333,-6678,-999,640,  0
P             DW    0
N             DW    0
DATA ENDS
```

例5.16

```
CODE SEGMENT   USE16
               ASSUME CS：CODE，DS：DATA, SS：STACK
START：        MOV AX，DATA
               MOV DS，AX
               MOV CX，100              ；置循环初值
               LEA ESI，ARY
          L：  MOV EAX，[ESI]
               CMP EAX，0
               JZ  EXIT                 ；为0则转EXIT
               JNS PLUS                 ；为正数则转PLUS
               INC N                    ；为负数则N+1
               JMP CONT
```


例5.16

```
PLUS：         INC  P                   ；为正数则P+1
CONT：         ADD  ESI，4      ；地址加4使ESI指向下一个数
               JMP  L                   ；未完则继续循环


EXIT：         MOV AX，4C00H
               INT  21H
               END  START
```


4 多重循环程序设计

当内层循环设计完之后，用其替换外层循环体中被视为一个处理粗框的对应部分，再对外层循环进行置初值、工作、修改和控制部分的设计，这样一层一层地进行设计，就构成了一个多重循环结构。

例5.17　已知M×N矩阵A的元素A i j 按行序存放在以A为首地址的字节存储区中，试编写程序，求每行元素之和S i 。

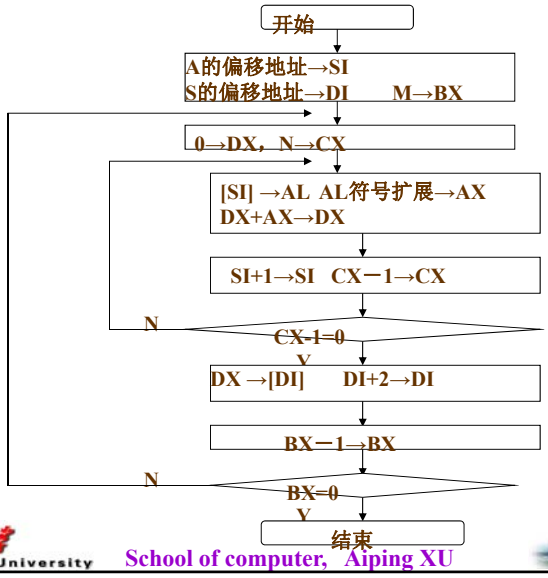
每行元素之和S i 的计算公式为：

$$S_i = \sum_{i=1}^{n} A_{ij} \qquad (i = 1, 2, \ldots, m)$$

例5. 17　程序清单

```
DATA    SEGMENT
        A   DB  11，12，13，14，15，16
            DB  17，18，19，20，21，22
            DB  23，24，25，26，27，28
            DB  29，30，31，32，33，34
            DB  35，36，37，38，39，40
M           =   5
N           =   6
S           DW  M  DUP（0）
DATA    ENDS
CODE    SEGMENT
            ASSUME  CS：CODE，DS：DATA
```

例5. 17　　程序流程图



例5. 17　　程序清单

```
START:    MOV  AX，DATA
          MOV  DS，AX
          LEA   SI，A          ；A的首地址送SI
          LEA   DI，S          ；和数的地址送DI
          MOV  BX，M           ；将行数送BX作为循环的次数
 L1：     MOV  DX，0           ；将行的和数初值送0
          MOV  CX，N           ；将行元素个数（列数）送CX作为循环的次数
L2：      MOV  AL，[SI]        ；求行元数的和
          CBW
          ADD   DX，AX
          INC    SI
          LOOP L2              ；一行的元素未完则转L2 继续
```

例5. 17　程序清单

```
        MOV  [DI]，DX      ；和数送结果单元
        ADD  DI，2         ；地址加2指向下一个数
        DEC  BX
        JNZ  L1            ；行数未完则转L1继续
        MOV  AH，4CH
        INT  21H
CODE    ENDS
        END      START
```


5．5　算术运算与代码变换程序设计

5．5．1　算术运算程序

例 5.18　设A、B两个变量都是6字节的无符号数据。编写程序计算A+B→C。

```
DATA SEGMENT
A           DB    91，12，13，14，15，16
B           DB    37，18，19，20，21，22
C           DB    7  DUP（0）
 DATA       ENDS
CODE SEGMENT
            ASSUME  CS：CODE，DS：DATA
START：     MOV  AX，DATA
            MOV  DS，AX
```

例 5.18 程序

```
        LEA   SI，A          ；置初始参数地址值
        LEA   DI，B
        LEA   BX，C
        MOVCX，6             ；置初始参数个数
        CLC                  ；清进位位
L：     MOV   AL，[SI]       ；两数相加送结果单元
        ADC   AL，[DI]
        MOV   [BX]，AL
        INC   SI             ；地址修改
        INC   DI
        INC   BX
        LOOP  L              ；循环控制
```


例 5.18 程序

```
        MOV   AL，0          ；最高进位位的处理
        ADC   AL，0
        MOV   [BX]，AL
        MOV       AH，4CH
        INT   21H
CODE    ENDS
        END   START
```

例 5.19 （不看）

例 5.20 程序

例5.20　设变量BCD1、BCD2为压缩型BCD码，编写计算表达式 S=BCD1＋BCD2的程序，同时将计算结果显示输出。

```
DATA SEGMENT
BCD1 DB    56H，92H
BCD2 DB    73H，81H
S          DB    3  DUP (0)
F          DB    5  DUP（‘’），0DH，0AH，‘$’
DATA ENDS
CODE SEGMENT
           ASSUME  CS：CODE，DS：DATA
START：    MOV  AX，DATA
           MOV  DS，AX
```


例 5.20 程序

```
        MOV  AL，BCD1          ；低位数相加
        ADD  AL，BCD2
        DAA                    ；BCD调整
        MOV  S，AL             ；送结果单元
        MOV  BL，AL
        MOV  AL，BCD1+1              ；高位数相加
        ADC  AL，BCD2+1
        DAA                    ；BCD调整
        MOV  S+1，AL           ；送结果单元
        MOV  CL，0             ；高位进位送结果的最高位
        ADC  CL，0
        MOV  S+2，CL
        ADD  CL，30H           ；将结果转换为ASCII码送F
```

例 5.20 程序

```
        MOV  F，CL
        MOV  AH，AL ；AH=AL=74H
        MOV  BH，BL  ;BH=BL=29H
        MOV  CL，4
        SHR   AH，CL ；  AH=07
        AND   AL，0FH ；AL=04
        SHR   BH，CL   ；BH =02
        AND   BL，0FH    ; BL = 09
        OR     AX，3030H ；AX=3734H
        OR     BX，3030H ; BX=3239H
        XCHG        AH，AL ; AX=3437H
        XCHG        BH，BL ;BX= 3932H
```


例 5.20 程序

```
        MOV  WORD      PTR  F+1，AX ; AX=3437H
        MOV  WORD      PTR  F+3，BX;BX= 3932H
        LEA   DX，F                    ；将F的内容显示输出
        MOV  AH，9
        INT    21H
        MOV   AH，4CH
        INT    21H
CODE    ENDS
        END   START
```


5．5．2　　代码转换程序设计

例5.21　编写程序，将BIN字变量中的16位有符号二进制数转换成十进制数，然后将十进制数的ASCII码存入存储器的BUF字节缓冲区中，并显示输出转换的结果。

其解题分为四步：

一是处理16位有符号二进制数的符号位；

二是完成二进制转换成十进制的操作；

三是ASCII码的转换；

四是显示输出转换的结果。

四步中最主要的是完成二进制转换成十进制的操作。

其转换的方法有多种，下面介绍一种除十取余的方法。此转换方法是将二进制数每次除以10。

第一次将二进制数除以10得到的余数是个位数；

第二次将所得的商除以10得到的余数是十位数；

第三次再将所得的商除以10得到的余数是百位数；

第四次再将所得的商除以10得到的余数是千位数；

第五次再将所得的商除以10得到的余数是万位数。

例5.21

每次将所得的余数依次保存在堆栈中。ASCII码的转换操作是将堆栈中的内容依次弹出，将每个数加30H后送BUF字节缓冲区。最后用DOS功能调用将BUF字节缓冲区的内容显示输出。编写的程序清单如下：

```
STACK SEGMENT     STACK
      DW     100 DUP（0）
STACK ENDS
DATA  SEGMENT
BIN   DW     7462
BUF   DB     6    DUP（0），0DH，0AH，‘$’
TEN   DW     10
DATA  ENDS
CODE  SEGMENT
```

例5.21

```
        ASSUME   CS：CODE，DS：DATA，SS：STACK
START：MOV    AX，DATA
       MOV    DS，AX
       MOV    AX，BIN        ；符号位处理
       OR     AX，AX
       JNS    PLUS
       NEG    AX
       MOV    BUF，  ‘—’
       JMP    NEXT
PLUS：MOV    BUF，  ‘+’
NEXT：MOV    CX，5
L1：   MOV    DX，0
       DIV    TEN
       PUSH   DX
       LOOP   L1
       MOV    CX，5                ；ASCII码的转换
```


例5.21

```
       LEA    BX，BUF+1
L2：   POP    AX
        ADD   AL，30H
        MOV   [BX]，AL
        INC   BX
        LOOP  L2
        LEA   DX，BUF            ；显示字符串
        MOV   AH，9
        INT   21H
        MOV         AH，4CH
        INT   21H
CODE   ENDS
       END    START
```

例5.22

例5.22　编写将STR字符串中存放的十进制ASCII码数字（－32768～32767）转换为二进制数字，并把转换的结果送BUF缓冲区的程序。

该程序的功能是要将有符号的十进制数（连同符号位）转换成二进制数。有符号十进制数的书写形式有如下三种：

d1 d2 …dn ，　+d1　d2　…dn　，　－d1　d2　…dn

其中di （i =1, 2, …, n)为0～9中的一位十进制数字。

将有符号十进制数转换二进制数须分三步完成。第一步是对符号的处理；第二步是对数字的转换；第三步是对错误信息的处理。对数字的转换可以把十进制数字d1 d2 …dn写成如下形式：

d1 d2 …dn=（…（0）×10+ d1）×10+ d2+…）×10+ dn

由于di和10在计算机中都是以二进制形式表示和运算的，所以上式在计算机中计算所得的值为二进制。如果d1 d2 …dn为正数，则计算所得的值就是要转换的二进制；如果d1 d2 …dn为负数，则计算所得的值须求补后才是转换的二进制。编写的程序清单如下：

例5.22

```
STACK   SEGMENT     STACK
        DW      100 DUP (0)
STACK   ENDS
DATA    SEGMENT
STR     DB      '－2510'
N               =       $－STR
BUF     DW      2  DUP   (0)
FLAG    DB      0
TEN     DW      10
ERR1    DB      '十进制数的ASCII码串有非数字字符的错误！ $'
ERR2    DB      '十进制数字超出数字范围错误！ $'
DATA    ENDS
CODE    SEGMENT
                        ASSUME     CS：CODE, DS：DATA, SS：STACK
START:  MOV     AX, DATA
        MOV     DS, AX
```

例5.22

```
        MOV     FLAG，0；置初值
        MOV     AX，0
        LEA     SI，STR
        MOV     CX，N
        MOV     BL，[SI]
        CMP     BL，‘－’      ；第一个字符为非‘－’则转A
        JNZ     A
        MOV     FLAG，1；第一个字符为‘－’则FLAG置1
        JMP     B
A:      CMP     BL，‘＋’      ；第一个字符无‘－’、‘＋’则转N2
        JNZ     N2
B:      DEC     CX            ；跳过第一个符号位字符
N1:     INC     SI
        MOV     BL，[SI]      ；取出一个ASCII码数
```


例5.22

```
N2:     CMP     BL，30H       ；为非数字则转ER1进行出错处理
        JB      ER1
        CMP     BL，39H
        JA      ER1
        SUB     BL，30H       ；将ASCII码转换为数字
        MOV     BH，0
        MUL     TEN           ；将上次的数乘以10
        JO      ER2           ；超出范围则转ER2进行出错处理
        ADD     AX，BX        ；将本次的数加到结果中
        JC      ER2           ；超出范围则转ER2进行出错处理
        LOOP    N1
        CMP     FLAG，1       ；为负数则将转换的结果求补
        JNZ     RE
        NEG     AX
RE:     MOV     BUF，AX       ；保存结果
```

例5.22

```
            JMP     EXIT
ER1:        LEA     DX，ERR1     ；显示非数字的出错信息
            MOV     AH，9
            INT     21H
            JMP     EXIT
ER2:        LEA     DX，ERR2；显示超出范围的出错信息
            MOV     AH，9
            INT     21H
            JMP     EXIT
EXIT:       MOV       AH，4CH
            INT     21H
CODE  ENDS
            END     START
```

例5.23

编写程序，将BUF缓冲区中5个字节的压缩型的BCD码转换成非压缩型的BCD码（ASCII）送STR缓冲区，并将转换结果显示输出。

```
STACK  SEGMENT     STACK
       DW      100 DUP（0）
STACK  ENDS
DATA   SEGMENT
       BUF     DB            23H，65H，28H，91H，66H
       STR     DB      10  DUP   （0）
       DISBUF DB       10  DUP  （0），0DH，0AH，  ‘$’
DATA   ENDS
CODE   SEGMENT
                       ASSUME     CS：CODE，DS：DATA，SS：STACK
START：MOV     AX，DATA
       MOV     DS，AX
       LEA     SI，BUF        ；送循环的初值
       LEA     DI，STR
```

例5.23

```
        LEA     BX, DISBUF+9
        MOV     DX, 5
L:      MOV     AL, [SI]        ；将压缩的BCD码的低位转换为ASCII
        AND     AL, 0FH
        ADD     AL, 30H
        MOV     [DI], AL
        INC     DI
        MOV     [BX], AL
        DEC     BX
        MOV     AL, [SI]        ；将压缩的BCD码的高位转换为ASCII
        MOV     CL, 4
        SHR     AL, CL
        ADD     AL, 30H
        MOV     [DI], AL
```


例5.23

```
        INC     DI
        MOV     [BX], AL
        DEC     BX
        DEC     DX
        JNZ     L
        LEA     DX, DISBUF
        MOV     AH,  9
        INT     21H
EXIT:   MOV     AH, 4CH
        INT     21H
CODE    ENDS
        END     STAR
```

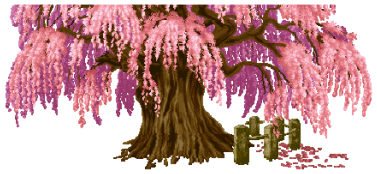
本章结束

WHU作业：

顺序程序设计：5.1　5.3

分支程序设计：5.5、5.6

循环程序设计：　5.7、5.13、　5.15 、5.17、5.20、5.21、5.22



实验1

- 做 习题 5.20，用5.21的代码显示结果