总复习

第一章 汇编语言基础知识

1.2 数据表示

1. 十进制数转换二进制数；

整数部分不断除以2，记下每次得到的余数，直到商为零；②余数倒排，即最后得到的余数排在最高位，第一个余数排在最低位。例如将十进制数13转换成二进制数：

小数部分转换：乘2取整，顺序排列得到的整数。例如将0.8125转换成二进制数

2. 二进制数十六进制数互相转换

1101001.101B=68.AH （~~68.5H~~）

3. BCD 码（8421码、二——十进制数）

- 解决十进制数在计算机内部如何表示。BCD码规定用四位二进制数表示一位十进制数。
- 对多位十进制数，只要把每一位十进制数分别表示为四位二进制数即可。
- 压缩BCD码和非压缩BCD码

4. ASCII码

- 解决字母、符号在计算机内部如何表示。
- 基本ASCII码（标准ASCII码）用七位二进制数表示一个符号（共128个）；
- 书写：用两位十六进制数书写，如41H----A；
- 种类：1）控制字符（前32个和最后一个）：

  0D--- 回车，0A--- 换行；

  2）其他为打印字符（可显示字符）；
- 应记住的ASCII码：30H~39H，41H，61H
- 扩展ASCII码用八位二进制数表示一个符号（共256个）。

5. 有符号数表示方法

原码、反码、补码总结：

1）正数的原码反码补码相同；负数的原码反码补码各不相同，但符号位都是1。

2）设字长为八位，原码反码的表数范围为-127～+127，补码的表数范围为-128～+127。

3）已知某负数的补码，求该负数的真值，方法如下：

①符号位不动，其余位求反加一，得到的是该负数的原码；

②根据原码即可写出该负数的真值。

例：$[X]_{补}$=11111100B

$[X]_{原}$=10000011B+1=10000100B

X= -0000100= - 4

1.5 8086微处理器

1. 8086的功能结构

总线接口单元、执行单元、指令预取队列

2. 8086的寄存器组

❖8086通用寄存器

(1) AX-累加器。 (2) BX-基地址寄存器。

(3) CX-计数寄存器。 (4) DX-数据寄存器。

(5) SI-源变址寄存器。 (6) DI-目的变址寄存器。

(7) BP-栈基地址寄存器。 (8) SP-栈顶指针。

❖专用寄存器 IP、标志寄存器FLAG（PSW）

3. 段寄存器使用规定

访问存储方式	默认段寄存器	可超越的段寄存器	偏移地址
取指令	CS	无	IP
堆栈操作	SS	无	SP
一般数据访问	DS	CS, ES, SS	有效地址EA
串的源操作数	DS	CS, ES, SS	SI
串的目的操作数	ES	无	DI
BP作基址时	SS	CS, ES, SS	有效地址EA

1.6 8086的寻址方式

1. 立即数寻址方式
2. 寄存器寻址方式
3. 存储器寻址方式
   - 寄存器间接寻址
   - 寄存器相对寻址
   - 基址变址寻址
   - 直接寻址
   - 基址变址相对寻址

mem操作数的各种形式

①[2050H];VAR_ADDR

②[BX];[BP];[si];[di]

③[BX+disp];[BP+disp];[si+disp];[di+disp]

disp[BX]; disp[BP]; disp[SI]; disp[DI]

④[BX+SI];[BX+DI];[BP+SI];[BP+DI]

[BX][SI];[BX][DI];.....

⑤[BX+SI+disp];[BX+DI+disp];

[BP+SI+disp];[BP+DI+disp];

第二章 8086的指令系统

2.1 数据传送指令

2.1.1 通用数据传送指令
MOV/XCHG/XLAT

- MOV DEST, SRC

①立即数只能作源操作数,且要与目的操作数匹配。
②两个操作数类型要匹配。
③如汇编程序无法确定操作类型，要加类型说明符。
④CS一般不能作目的操作数（用转移指令改变）。
⑤如果指令有两个操作数，不允许两个都是存储器数。
⑥所有“MOV”类指令均不影响标志。

2.1.2 堆栈操作指令

PUSH r16/m16/seg

POP r16/m16/seg

①堆栈操作是字操作指令

②在“POP”指令中，“POP CS”为非法指令。

③堆栈操作指令不影响标志。

2.1.3 标志传送指令

LAHF/SAHF/PUSHF/POPF

2.1.4 地址传送指令

LEA/LDS/LES

LEA r16, mem

2.1.5 输入输出指令

IN/OUT

2.2 算术运算指令

ADD/ADC/INC

SUB/SBB/DEC/NEG/CMP

①加法指令对标志的影响：ADD和ADC指令对所有的6个状态标志都有影响，INC指令不影响CF，影响其他五个标志。学习加减法指令要会设置标志、使用标志。

②加减法运算要注意OF和CF的意义不同，用法不同。

③注意NEG/CMP指令的用法。

MUL/IMUL

①指令指定的是乘数，被乘数是隐含的。如乘数类型为字节，则被乘数为AL，16位乘积用AX；如乘数类型为字，则被乘数为AX，32位乘积用DX，AX。

②影响 CF 、OF 标志：如果乘积的高一半为零，或高一半为低一半的符号扩展，则 CF=OF=0，否则，CF=OF=1。对其他标志无定义。

③单操作数指令，涉及mem时，须指定类型。

MUL BYTR PTR[BX+SI]

MUL WORD PTR[BX+SI]

DIV / IDIV

①指令指定的是除数，被除数是隐含的；除数为字节，则称为“字节除”，被除数使用AX；除数为字，称为“字除”，被除数使用DX AX。

②操作数的格式与乘法指令相同。

③指令对状态标志无定义。

④当产生除法溢出，CPU自动产生“0号”中断，运行相应中断服务程序。程序设计时，应避免产生除法溢出。

CBW/CWD

①注意符号扩展涉及的对象是AL和AH以及AX和DX，与其他寄存器无关。

②用来为有符号数除法准备被除数（16位被除数和32位被除数）。

③为无符号数除法准备被除数，用“0扩展”。

DAA/DAS/AAA/AAS/AAM/AAD

- 如果做BCD码加法运算，ADD、ADC指令后应紧跟DAA指令，以保证结果正确。
- 调整对象只能是AL寄存器（BCD码运算只能使用以AL寄存器为目的操作数的8位数运算指令）。
- 对OF标志无定义，设置其他标志

2.3 位操作指令

AND/OR/XOR/TEST/NOT

①操作数格式同加减法指令。

②设置CF=OF=0，影响SF、ZF、PF，对AF无定义。

③ TEST不影响目的操作数，只根据运算结果设置标志。

④NOT reg/mem 不影响任何标志。

①屏蔽若干位。（常用指令）

AND AL，01H；屏蔽AL的D7~D1，保留D0

AND AL，0FH；屏蔽AL高4位，保留低4位

②使若干位置1（常用指令）

OR BL，0F0H；使BL高4位置1，低4位不变

③清除CF、OF 或 设置标志

AND AL,AL (AND AL,0FFH；OR BL,BL；……）

;类似指令没有改变目的操作数，但使CF=OF=0，也可能纯粹以设置其他标志（ZF）为目的。

④求反　NOT AL／ NOT WORD PTR[BX+DI]

⑤对指定位求反

XOR AL，0FH；AL高4位不变，低4位求反

XOR CL，55H；CL偶数位求反，奇数位不变

⑥清除寄存器及CF（常用指令）

XOR AX,AX / XOR BX,BX

XOR BYTE PTR[BX]，BYTE PTR[BX] ×

⑦不改变操作数，测试操作数或操作数的指定位

```
TEST  AL，0FFH；由ZF标志判断AL是否为零
                ；也可以用CMP指令。
TEST  AL，01；由ZF标志判断AL的D0是否为零，
              ；ZF=1，则AL.D0=0
              ；ZF=0，则AL.D0=1
AND   AL，01；也可完成上述功能，但是改变了
              目的操作数
```

SHL/SAL/SHR/SAR

①标志设置　AF：对AF无定义；CF：按移入的值或为0或为1；

根据移位后的结果设置SF、ZF、PF；

OF：当移动一位时，移位后如果符号位发生变化，则OF=1,符号位不发生变化，则OF=0，移位次数大于一时，OF不定。

②操作数左移一位，相当于乘2（对有符号数同理，只要OF=0，结果就对）。

③操作数逻辑右移（SHR）一位，相当于无符号数除以2；
操作数算术右移（SAR）一位，相当于有符号数除以2。

ROL/ROR/RCL/RCR

移位指令和循环移位指令结合，可实现32位数左移右移。

2.4 控制转移类指令

2.4.1 无条件转移指令

- JMP SHORT LABEL;
  JMP NEAR PTR LABEL;
- JMP r16 ; JMP WORD PTR mem
- JMP （FAR PTR） LABEL
- JMP FAR PTR mem (JMP DWORD PTR mem)

2.4.2 条件转移指令

1.判断单个状态标志

助记符	标 志	说 明
JZ/JE	ZF=1	结果为0；两数相等
JNZ/JNE	ZF=0	不为0；不相等
JC/JB/JNAE	CF=1	加有进位；减有借位；其他
JNC/JNB/JAE	CF=0	无进位； 无借位；其他
JS	SF=1	结果为负
JNS	SF=0	结果为正
JP/JPE	PF=1	结果的低8位含偶数个“1”
JNP/JPO	PF=0	结果的低8位含奇数个“1”
JO	OF=1	运算结果溢出
JNO	OF=0	运算结果不溢出
★JCXZ	（CX=0）	串操作是否处理完所有

2. 比较无符号数高低(条件为一个标志或标志组合)

助记符	标志	说 明
JB/JNAE/JC	CF=1	低于/不高于不等于(<)
JNB/JAE/JNC	CF=0	不低于/高于或等于(≥)
JBE/JNA	CF=1或ZF=1	低于或等于/不高于(≤)
JNBE/JA	CF=0且ZF=0	不低于不等于/高于(>)

3. 比较有符号数大小(条件为标志组合)

助记符	标志	说 明
JL/JNGE	SF≠OF	小于/不大于且不等于(<)
JNL/JGE	SF＝OF	不小于/大于或等于 (≥)
JLE/JNG	SF≠OF或ZF=1	小于或等于/不大于 (≤)
JNLE/JG	SF＝OF且ZF=0	不小于且不等于/大于(>)

2.4.3 循环控制指令

```
LOOP   label          ; CX≠0，循环；否则退出
LOOPZ/LOOPF  label    ; CX≠0 且 ZF=1，循环
                      ; 否则退出
LOOPNZ/LOOPNE  label  ; CX≠0 且 ZF=0，循环
                      ; 否则退出
```

2.4.4 子程序调用及返回指令

```
CALL near ptr label          ; 段内直接调用
CALL r16/word ptr m16        ; 段内间接调用
CALL far ptr  label          ; 段间直接调用
CALL dword ptr mem           ; 段间间接调用
RET
RET  i16
```

2.4.5 中断控制指令

1. 8086中断类型

(1)外部中断

中断源来自CPU之外（两种）

- ❖可屏蔽中断：响应与否，受IF标志控制。
  涉及指令：CLI　STI
- ❖非屏蔽中断：不受IF控制的中断源。（2#）

(2)内部中断（4种）

中断源为：程序执行过程中程序自身引发的事件

- 除法错中断：除数为0或除法溢出。（0#）
- 单步中断：若单步标志TF=1，则每条指令执行行后产生单步中断。（1#）
- 溢出中断：执行中断指令INTO时，如OF=1，则产生溢出中断。（4#）
- 指令中断：执行中断调用指令INT n ，产生指令中段，类型号n=0～255。

2. 8086的中断过程

①中断向量表②进入中断服务程序③返回断点

- CPU取类型号;外中断:中断源提供
  内中断:指令提供、预定义
- 类型号*4：对应向量在表中的首地址
- 保护断点：PSW、CS、IP依次入栈
- 从向量表读取服务程序入口地址：
  IP←低位字 CS←高位字
- 进入中断服务程序

3. 8086的中断指令

INT i8 /INTO /IRET / CLI / STI

2.4.6 系统功能调用

- ❖在AH中设置调用的功能号；
- ❖在指定的寄存器中设置入口参数；
- ❖执行INT 21H指令，调用功能子程序；

1. 单个字符的输出 AH=02；
   入口参数：DL=字符的ASCII码
2. 字符串输出 AH=09；
   入口参数：DS:DX=字符串首地址
   字符串必须以$（24H）结尾
3. 单个字符输入 AH=1；
   入口参数：无；出口参数：AL=字符ASCII码

2.5 串操作指令

1. 串传送指令 MOVSB/MOVSW
2. 串存储指令 STOSB/STOSW
3. 串读取指令 LODSB/LODSW
4. 串比较指令 CMPSB/CMPSW
5. 串扫描指令 SCASB/SCASW
6. 重复前缀指令 REP REPZ/REPE 和REPNZ/REPNE

1）源数据串可以段跨越，目的串不可。

2）DF=0，地址指针+1或+2，DF=1，指针-1或-2。

3）注意重复前缀的使用。

2.6 处理机控制类指令

1）NOP(同XCHG AX,AX)

预留空间　删除指令　软件延时

2）段超越前缀指令　段寄存器：如 CS：，SS：...

3）指令封锁前缀指令LOCK（有相应引脚 LOCK）：

LOCK MUL

4）暂停指令 HLT

反复执行NOP，等待复位或中断信号。（慎用）

5）交权指令ESC 将浮点指令交给浮点处理器。

6）等待指令WAIT（有相应引脚信号 TEST）

用于与8087同步 （5T）

TEST=1 保持WAIT状态，TEST=0 退出WAIT状态。

第三章 3.1/3.2

1. 变量定义伪指令

1）字节定义伪指令DB

2）定义字单元伪指令DW

3）定义双字单元伪指令DD

4）其他数据定义伪指令

1）DF、DQ、DT 略

2）MASM6.0建议使用：BYTE/WORD/DWORD等

3）SBYTE/SWORD/SDWORD：有符号数专用。

5）DUP / ? / $

2. 基数控制伪指令（RADIX）

RADIX　n；n取2~16内的任意整数。

3. 符号常数定义伪指令（EQU、=）

- EQU

符号名　EQU　数值表达式

符号名　EQU　<字符串>；5.X版用双引号。

- “=”号伪指令

```
   .model small
   .stack
   .data
bvar db 16
wvar dw 4*3
dvar dd 4294967295
     db 1,2,3,4,5
abc  db 'a','b','c'
msg  db 'hello',13,10,'$'
bbuf db 12 dup('month')
dbuf dd 25 dup(?)
B1   DW  BVAR    ; [B1]=0000H
B2   DW  WVAR    ; [B2]=0001H
B3   DD  WVAR    ; [B3]=0001H, [B3+2]=段地址
```

4.定位伪指令ORG（/EVEN/ALIGN）

控制数据或指令的偏移地址。

1）ORG 参数

使地址计数器指向参数表达的偏移地址。

ORG 100H；从0100H单元开始分配存储器。

ORG $+10；$表示地址计数器的当前值，$+10

；表示由当前地址向前跳过10个字节。

2）EVEN ；使它后面的数据或指令从偶地址开始。

3）ALIGN n；使它后面的数据或指令从n的整数倍

；地址开始（可被n整除）。

n是2的乘方（2，4，8...）且小于所在段的定位属性值。如“ALIGN 4”，使下一个地址开始于双字边界

定位伪指令举例：

```
DATA SEGMENT
D01 DB 1,2,3          ; D01偏移地址为0,  $=0003H
   EVEN（ALIGN 2）;                      $为0004H
D02 DW 5              ; D02偏移地址为04H,  $=0006H
   ALIGN 4    ; 最接近6的、可被4整除的数是8
                ;                        $=0008H
D03 DD 6              ; D03的偏移地址为08H,  $=000CH
   ORG $+10H   ; 000CH+0010H=001CH   $=001CH
D04 DB ‘abc’   ; D04的偏移地址为001CH  $=001FH
   LEN EQU $－D04; LEN=001F－001C=3
                        ; 变量D04所占的字节数。
```

3.3程序段的定义和属性

1. 简化段定义标准格式：

```
.model small
.stack
.data
.....
.code
.startup
  .....
.exit 0
end
```

2. 完整段定义格式

1）.段定义伪指令

段名 SEGMENT［定位］［组合］［段字］［‘类别’］

……

段名 ENDS

(1)定位属性：指定逻辑段的起始地址

BYTE：为下一个可用的字节地址(xxxx xxxxB)

WORD：为下一个可用的偶数地址(xxxx xxx0B)

DWORD：下一个可被4整除的地址(xxxx xx00B)

PARA：下一个可被16整除的地址(xxxx 0000B)

PAGE：下一个可被256整除的地址(0000 0000B)

2）指定段寄存器伪指令

ASSUME 段寄存器:段名[，段寄存器:段名.....]

建立段寄存器与段之间的缺省关系，改变这种缺省关系可使用段跨越前缀。

如 ASSUME CS：CODE，DS：DATA

ASSUME 段寄存器：NOTHING

对指令给出的段寄存器取消已经指定的缺省关系。

注意：ASSUME伪指令并不能为段寄存器赋值。

3）段组伪指令

组名 GROUP 段名[，段名].....

将多个同类但不同名的段合并为一个不超过64KB的物理段，并使用组名统一访问它。可理解为组合属性PUBLIC的补充。

.MODEL SMALL具有下面语句的作用：

```
dgroup  GROUP  _data, _bss, stack
```

4）汇编结束伪指令

END [标号]

第三章 表达式/运算符/操作符

1. 数值表达式及运算符

常数、寄存器、变量及标号等用运算符连接起来即构成表达式，如细分，有算术表达式、逻辑表达式、关系表达式、地址表达式等，但是由于前三种表达式或由它们构成的综合型表达式其结果都是数值，故一律算做**数值表达式**；如果一个表达式的结果从物理意义来说，代表存储器单元的地址，则称其为**地址表达式**。

常见运算符分成五类：

1）算术运算符：+ 、- 、* 、/ 、MOD

2）逻辑运算符：AND 、OR 、XOR 、NOT

3）移位运算符：SHL 、SHR

4）关系运算符：EQ、NE、GT、LT、GE、LE

5）高低分离符：HIGH、LOW、HIGHWORD、LOWWORD

2. 地址操作符及地址表达式

- OFFSET 变量|标号；返回变量或标号的偏移地址
- SEG 变量|标号；返回变量或标号的段地址

3. 类型操作符（数值表达式）

对变量或标号的类型属性进行操作。

PTR/THIS/LABEL/SHORT/TYPE/

SIZEOF/LENGTHOF

例 3.4 属性及应用

```
        .model small
        .stack
        .data
V_byte  equ this byte
V_word  dw 3332h,3735h
Target  dw 5 dup(20h)
Crlf    db 0dh,0ah, '$'
Flag    db 0
N_point dw offset s_label
        .code
        .startup
        mov al,byte ptr v_word
        dec al
        mov  v_byte ,al;v_word=3331h
```

```
N_label:cmp flag,1 ;flag单元=0
        jz s_label
        inc flag  ; flag=1
        jmp short n_label
S_label: cmp flag,2  ;flag=1
        jz next
        inc flag   ;flag=2
     jmp n_point ;段内间接转移，等同于jmp s_label
Next: mov ax,type v_word  ;ax=0002h
        mov cx,lengthof target;5个数据项，cx=5
        mov si,offset target
W_again: mov [si],ax
        inc si
        inc si
        loop w_again;对target填充5个字：0002h
```

```
mov cx,sizeof target;cx=10
      mov al,'?'
      mov di,offset target
B_again: mov [di],al
      inc di
      loop b_again ;对target填充10个 '?'
      mov dx,offset v_word
      mov ah,9
      int 21h   ;显示结果为1357??????????
      .exit 0
      end
```

第四章 基本汇编语言程序设计

1. 按程序结构分类：

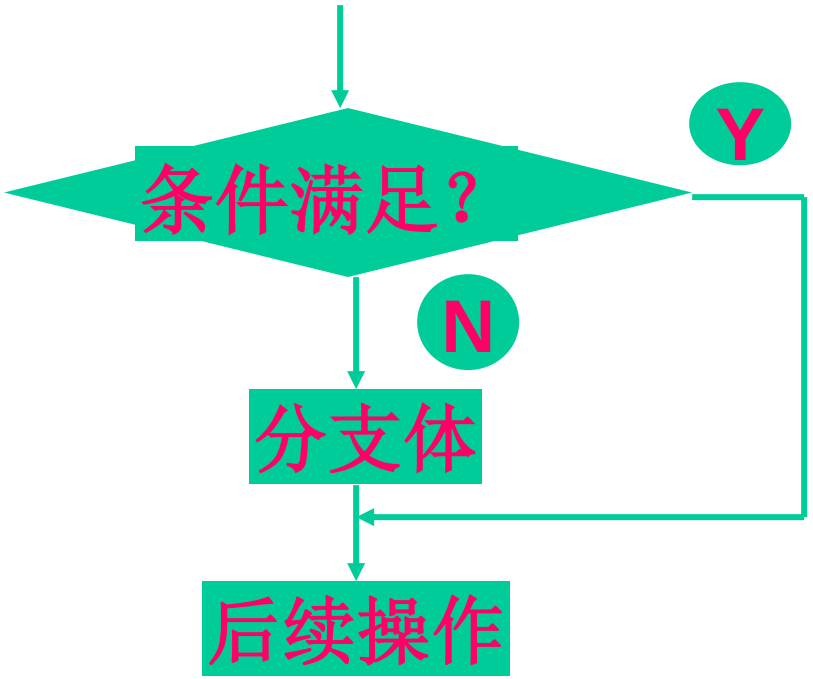
- 4.1 顺序程序设计
- 4.2 分支程序设计
- 4.3 循环程序设计
- 4.4 子程序设计

2. 按程序功能分类

4.1 顺序程序设计

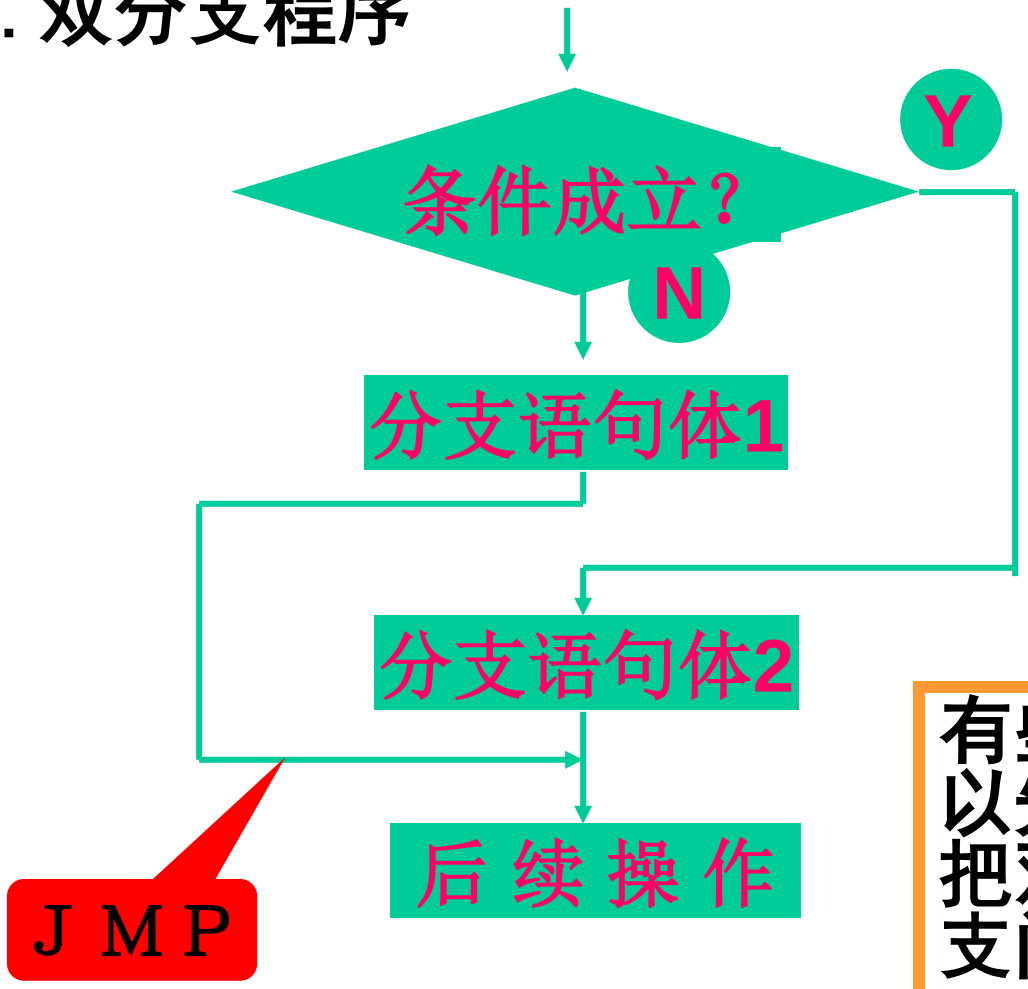
4.2 分支程序设计

1. 单分支类型



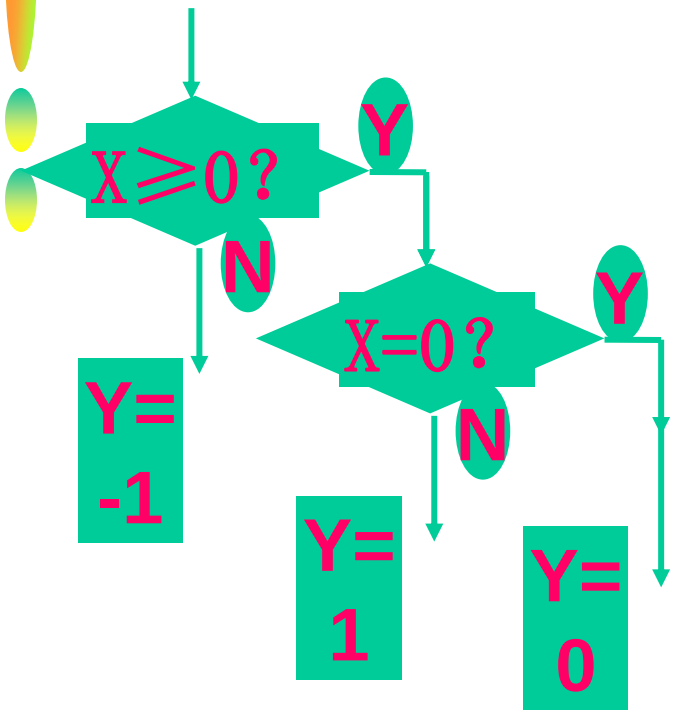
对同一个问题，根据选择的条件不同，单分支结构的流程图有两种画法，对应的程序也有两种编法。

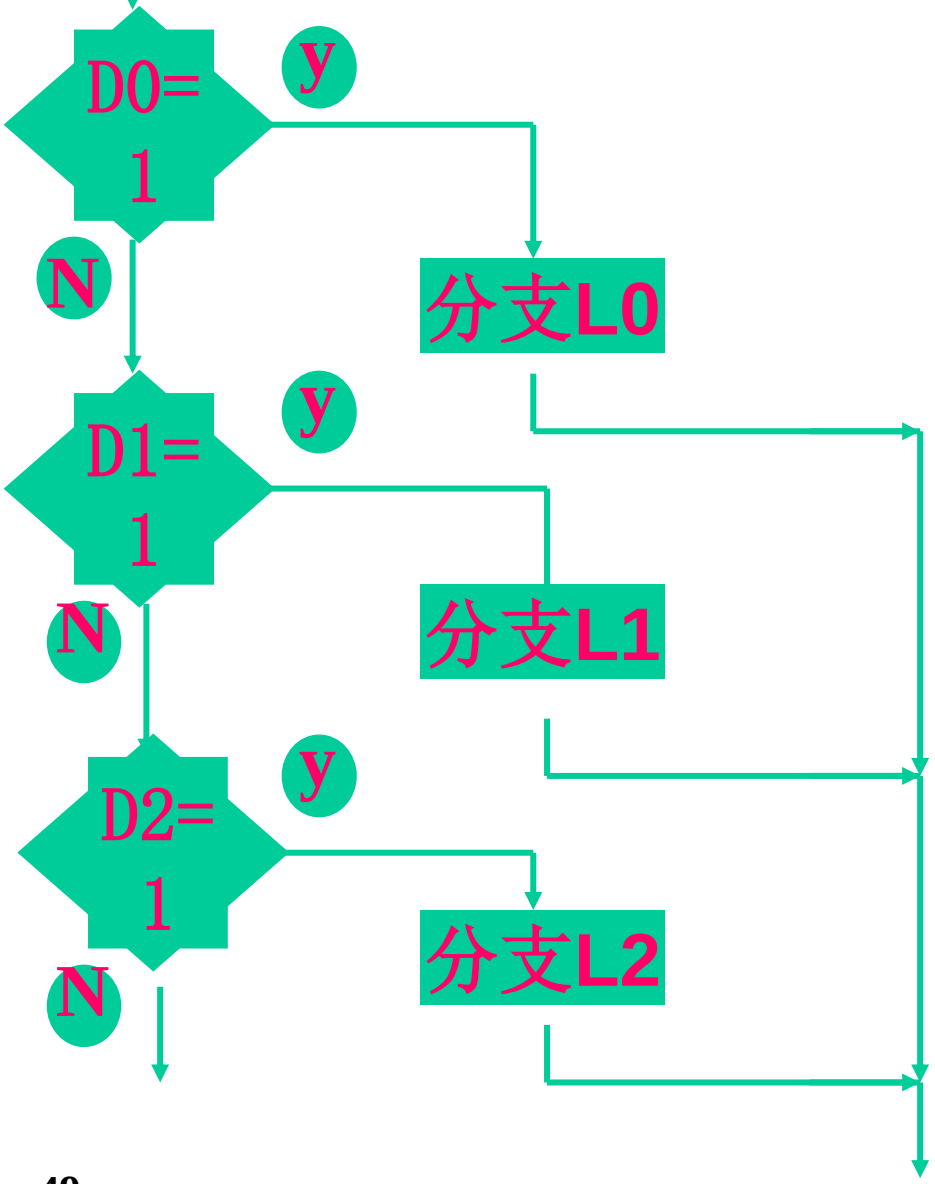
2. 双分支程序



有些双分支问题可以先假设一种情况，把双分支改成单分支问题。

3.多分支程序





4.3 循环程序设计

1. 两种循环结构

1） “先循环、后判断”结构

相当于高级语言的“直到型”循环

2） “先判断、后循环”结构

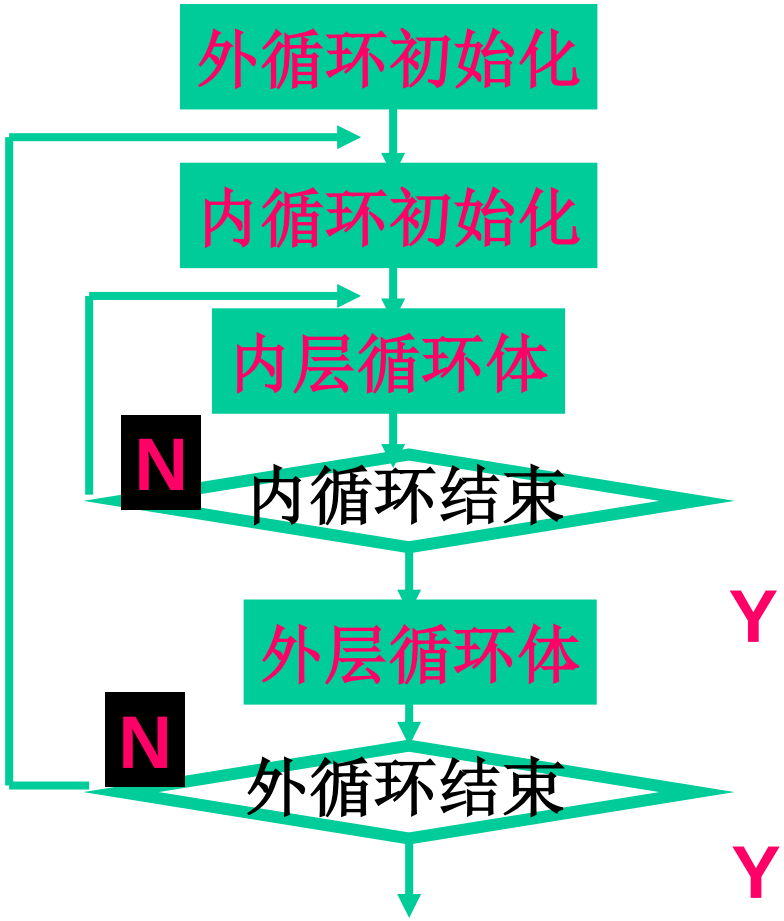
相当于高级语言的“当型循环”，可实现0次循环。

2. 循环程序设计

编写循环程序的关键在于循环的控制。

- 循环次数已知：可用LOOP指令，CX计数。
  如教材例4.5。
- 循环次数和ZF标志：可用LOOPZ、LOOPNZ指令。
  如教材例4.6。
- 循环次数未知：通常要采用比较指令和各类条件转移指令实现循环控制。

3. 多重循环：循环程序的嵌套构成多重（多层）循环。最常见的多重循环为两重循环。



4.4 子程序设计

把功能相对独立的程序段单独编写和调试，作为一个相对独立的模块供程序使用，就形成子程序。

使用子程序：简化源程序结构；提高编程效率。

4.4.1 过程定义伪指令

4.4.2 子程序的参数传递

4.4.3 子程序的嵌套递归重入

4.4.4 子程序的应用

1 过程定义伪指令

过程名 PROC [NEAR|FAR]
过程体
RET（RET N）
过程名 ENDP

一个完整的子程序，特别是供其他编程人员使用的子程序，必须附有一个详细说明：

子程序名（过程名）
子程序功能介绍
子程序的入口参数
子程序的出口参数
子程序内使用的寄存器（存储单元）
使用该子程序的范例

2. 子程序的参数传递

主程序和子程序之间通常需要传递参数：

- 入口参数（输入参数）：主程序提供给子程序
- 出口参数（输出参数）：子程序返回给主程序
- 参数的形式：
  - ① 数据本身（传值）
  - ② 数据的地址（传址）
- 传递的方法：
  - ① 寄存器　② 变量　③ 堆栈

2. 按程序功能分类

- 多精度运算、查表（查代码、特定值等）、排序
- 数据范围判断（0～9、A～Z、a～z）
- 字母大小写转换；字符串传送、比较等操作
- 求最小最大值、数据求和、统计字符个数
- ASCII、BCD及十六进制数据间的代码转换

二进制数转换成ASCII码总结：

1. 二进制数转换成二进制形式的ASCII码

如习题4.16：01000101B→30H 31H 30H...

2. 二进制数转换成十六进制形式ASCII码

4位二进制数→0-9，A-F

3. 二进制数转换成十进制形式的ASCII码

（二进制数转换成BCD码）

二进制数除以10，得到个位，再除以10，得到十位，...

也可参照习题4.21的方法，先得到最高位，然后依次得到低位。

5.1 高级语言特性

5.1.1 条件控制伪指令

5.1.2 循环控制伪指令

5.1.3 过程声明和调用伪指令

1. 条件控制伪指令

```
.IF 条件表达式          ; 条件为真，执行分支体1
    分支体1
 [.ELSEIF 条件表达式    ; 前面IF[及前面ELSEIF]为假，
      分支体2 ]         ; 当前条件为真，执行分支体2
 [.ELSE                 ; 前面IF[及前面ELSEIF]为假，
      分支体3 ]         ; 执行分支体3
 .ENDIF                 ; 分支结束
```

表达式中的操作符：

表5.1列出的操作符用于伪指令的条件表达式，第三章介绍的操作符用于数值表达式和地址表达式（构成指令的操作数），两类操作符不可混淆。

操作符可分为：比较、逻辑运算和测试三类，其中测试又可分为：标志测试、位测试、寄存器测试、存储单元测试（reg\mem）。

条件表达式中比较的两个数据可能是无符号数，也可能是有符号数，可分成以下情况：

- 数据为变量：用DB、DW、DD等定义的变量一律作为无符号数，若需要进行有符号数比较，必须使用SBYTE、SWORD、SDWORD定义。
- 数据为寄存器或存储单元：默认为无符号数，若需要进行有符号数比较，必须使用操作符SBYTE PTR或SWORD PTR指明类型。

2. 循环控制伪指令

.WHILE和 .ENDW ：当型循环结构

.REPEAT和 .UNTIL/.REPEAT和 .UNTILCXZ：直到型

.BREAK：退出循环；.CONTINUE ：转向循环体开始

格式 1

```
.WHILE 条件表达式；表达式为真，执行循环体
    循环体                ;
.ENDW                     ；循环体结束
```

格式 2

```
.REPEAT                    ; 重复执行循环体
   循环体
.UNTIL 条件表达式 ; 直到条件表达式为真
```

格式 3

```
.REPEAT                          ; 重复执行循环体
   循环体
.UNTILCXZ [条件表达式]; CX←CX-1, 直到CX=0
                                 ; 或条件表达式结果为真
```

3. 带参数的过程定义、过程声明和过程调用伪指令

利用堆栈传递参数为常用方式，但传统的编程方法容易出错。MASM6.X扩充了PROC伪指令的功能，并新增了几条伪指令，使调用子程序具有高级语言的特性。

- 过程声明 PROTO

  过程名 PROTO 语言类型 ,[参数名]:类型,.....

❖过程定义PROC

过程名 PROC 语言类型 [USES 寄存器列表]
,形参1:类型,形参2:类型...
[LOCAL 参数表]
……
……
过程名 ENDP

❖过程调用伪指令 INVOKE

INVOKE 过程名 , 实参1, 实参2, … …

5.2宏结构程序设计

对常用的、具有独立功能的程序段，除了可定义为过程外，还可定义为宏结构或宏指令。宏指令提供了简化程序设计的另一种方法。

通常与宏指令配合使用的伪指令还有重复汇编和条件汇编。宏指令、重复汇编和条件汇编统称宏结构。

5.2.1 宏汇编

5.2.2 重复汇编

5.2.3 条件汇编

宏汇编

宏定义

宏名 MACRO ［形参1，形参2，……］

宏定义体；指令语句的组合

ENDM

宏调用

宏名 ［实参1，实参2，… …］

宏展开

汇编时，用宏体取代宏调用，用实参取代形参，称为宏展开。

宏的参数

参数的形式灵活多变，可以是常数、变量、存储单元、指令操作码或它们的一部分，也可以是表达式，使用灵活多变的参数，同一个宏定义甚至可以执行不同的操作。

几个宏操作符

&：替换操作符：用在宏体中。

<>：字符串传递操作符：用在宏调用的实参中。

！：转义操作符：用在宏调用的实参中。

%：表达式操作符：用在宏调用的实参中。

宏与子程序

相同点：简化源程序的设计和结构

不同点：

※ 处理时间段不同。

※ 对目标程序的长度影响不同。

※ 对目标程序执行速度的影响不同。

※ 传递参数的方式不同；传递参数过程中如出现错误，错误的性质不同。（宏：语法；子程序：逻辑）

通常，当程序段较短，要求较快执行时，用宏定义。当程序段较长，或为减小目标代码长度，用子程序。

从程序的“易读性”考虑，子程序使用的更多。