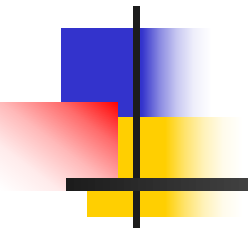

第四章

汇编语言程序设计

4.1 汇编语言类型与格式

4.1.1 程序设计语言类型

1. 机器语言

2. 汇编语言

3. 高级语言

4.1.2 汇编语句

4.1 汇编语言类型与格式

4.1.3 汇编语句基本格式

```
标号:   操作符    操作数    ;      注释
START:  MOV   AX，5AH    ;  立即数 5AH 送累加器 AX
```

1. 标号

- 段地址。
- 偏移地址。
- 距离。

2. 操作符

3. 操作数

4. 注释

4.2 汇编语言的基本语法

4.2.1 常数

1. 数值常数

2. 字符串常数

3. 符号常数

4.2 汇编语言的基本语法

4.2.2 数字与文字字符

4.2.3 标识符

4.2.4 变量

4.2 汇编语言的基本语法

4.2.5 变量和复制操作符定义语句

1. 定义变量

<变量名> 变量定义伪指令 常量、变量或表达式

```
DATA_A   DB   10，10H，'AB'  ；定义了 4 个字节的变量，其中有一字符串
DATA_B   DW   200H，-2，'$'，？；问号?被用于定义不确定的值，用于存放运算结果
DATA_C   DD   3*20，0FFFBH   ；定义了两个双字变量
DATA_D   DB     'GOOD'     ；定义字符串,GOOD 是 4 个 ASCII 码
DATA_E   DW     'GOOD'     ；
```

4.2 汇编语言的基本语法

4.2.5 变量和复制操作符定义语句

1. 定义变量

图 4-1 字节字符串存储图

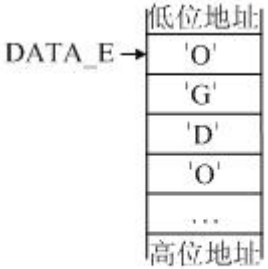

图 4-2 字型字符串存储图

2. 复制操作符DUP

4.2 汇编语言的基本语法

4.2.6 表达式

1. 数字表达式

2. 地址表达式

4.2 汇编语言的基本语法

4.2.7 一般运算符

1. 算术运算符

表 4-1　汇编软件支持的运算符

运　算　符			运算结果	示例
类型	符 号	名 称		
算术运算符	+	加法	和	7+9 = 16
	-	减法	差	7-3 = 4
	*	乘法	乘积	4*6 = 24
	/	除法	商	25/6 = 4
	MOD	模除	余数	12 MOD 3 = 0
	SHL	左移	左移 1 至多位二进制数	0011B SHL 2　= 1100B
	SHR	右移	右移 1 至多位二进制数	1101B SHR 1　= 0110B
逻辑运算符	NOT	非运算	逻辑非按位运算	NOT　1010B = 0101B
	AND	与运算	逻辑与按位运算	1011B AND 1101B = 1001B
	OR	或运算	逻辑或按位运算	1001B OR 1100B = 1101B
	XOR	异或运算	逻辑异或按位运算	1010B XOR 1100B = 0110B
关系	EQ	相等	结果为真输出全 1	6 EQ 11B=全 0

4.2 汇编语言的基本语法

4.2.7 一般运算符

1. 算术运算符

运算符	NE	不等	结果为真输出全 1	6 NE 11B=全 1
	LT	小于	结果为真输出全 1	6 LT 4=全 0
	LE	小于等于	结果为真输出全 1	6 LE 110B=全 1
	GT	大于	结果为真输出全 1	6 GT 101B=全 1
	GE	大于等于	结果为真输出全 1	6 GE 110B=全 0
数值返回运算符	SEG	返回段地址	段地址	SEG M1:M1 所在段段地址
	OFFSET	返回偏移地址	偏移地址	OFFSET M1:M1 的偏移地址
	LENGTH	返回变量单元数	单元数	LENGTH M1:M1 单元数
	TYPE	返回元素字节数	字节数	TYPE M1:M1 中字节数
	SIZE	返回变量总字节数	总字节数	SIZE M1:M1 总字节数
修改属性	PTR	修改类型属性	修改后类型	BYTE PTR [BX]
	THIS	指定类型/距离属性	指定后类型	ALPHA EQU THIS BYTE
	段寄存器名	段前缀	修改段	ES：[BX]
其他运算符	HIGH	分离高字节	高位字节	HIGH 1234H : 12H
	LOW	分离低字节	低位字节	LOW 1234H : 34H
	SHORT	短转移		JMP SHORT LABEL

4.2 汇编语言的基本语法

4.2.7 一般运算符

1. 算术运算符

【例 4-1】 比较以下两段相同的程序段。

右侧程序段是经汇编软件编译后所产生的机器码所能直接对应的等效程序段。

汇编源程序	编译后机器码直接对应的等效程序

B1 EQU 2200H+3300H
STA : MOV AX, B1－1200H
MOV BX, 22H*5H
MOV DX, B1/200
	B1 EQU 5500H
	STA : MOV AX, 4300H
	MOV BX, 0AAH
	MOV DX, 001BH

4.2 汇编语言的基本语法

4.2.7 一般运算符

2. 逻辑运算符

3. 关系运算符

【例 4-2】考察以下指令经汇编软件编译后生成的等效指令：

```
MOV   BX, 110B EQ 06H      ；编译后的等效指令是：MOV   BX, 0FFFFH
MOV   AL, 1111B GT 16      ；编译后的等效指令是：MOV   AL, 00H
ADD   AH, 254 LT 0FFH      ；编译后的等效指令是：ADD   AH, 0FFH
```

4.2 汇编语言的基本语法

4.2.8 数值返回运算符

数值返回运算符　变量 或 标号　；　如：TYPE 变量 或 标号

1. SEG运算符

2. OFFSET运算符

3. LENGTH运算符

【例 4-3】 对于以下左端源程序中对变量 D1, D2 和 D3 的定义表述，试比较源程序中的指令和经汇编软件编译后所产生对应的等效指令。

源程序中的数据定义	源程序中的指令	编译后的等效指令
D1　DW　8 DUP(0)	MOV　DL，LENGTH　D1	MOV　DL, 8
D2　DB　4 DUP(?)	MOV　BL，LENGTH　D2	MOV　BL, 4
D3　DD　5, 2, -3	MOV　AL，LENGTH　D3	MOV　AL, 1
D4　DB　'123ABC'	MOV　CL，LENGTH　D4	MOV　CL, 1

4.2 汇编语言的基本语法

4.2.8 数值返回运算符

数值返回运算符　变量 或 标号　；　如：TYPE 变量 或 标号

4. TYPE运算符

【例 4-4】对于以上例 4-3 和例 4-1 中对数据定义和标号定义，试比较以下指令经汇编软件编译后所产生的对应等效指令。

```
MOV   AL，TYPE   D1        ;    编译后的等效指令是：MOV   AL, 2
MOV   AL，TYPE   D2        ;    编译后的等效指令是：MOV   AL, 1
MOV   BL，TYPE   D3        ;    编译后的等效指令是：MOV   BL, 4
MOV   BL，TYPE   D4        ;    编译后的等效指令是：MOV   BL, 1
MOV   BL，TYPE   STA       ;    编译后的等效指令是：MOV   BL, 0FFH
```

4.2 汇编语言的基本语法

4.2.8 数值返回运算符　　数值返回运算符　变量 或 标号　　；　如：TYPE 变量 或 标号

5. SIZE运算符

【例 4-5】对于例 4-3 的定义，试比较以下指令经汇编软件编译后所产生的对应等效指令。

```
MOV   SI，SIZE  D1        ；编译后的等效指令是：MOV  SI, 16
MOV   BX，SIZE  D2        ；编译后的等效指令是：MOV  BX, 4
MOV   AX，SIZE  D3        ；编译后的等效指令是：MOV  AX, 4
```

4.2 汇编语言的基本语法

4.2.9 合成运算符

1. PTR运算符

<新类型> PTR <变量、标号、存储器操作数>

【例 4-6】根据例 4-3 的定义，考察以下指令的正确性：

```
MOV   [DI]，3AH              ；  指令错误，存储器操作数 DI 类型不确定
MOV   [DI]，D2               ；指令错误，存储器操作数类型不确定
MOV   [SI]，AX               ；指令正确，因为 AX 的内容本身限制其为一个字
MOV   DX，[D1]               ；指令正确，将 D1 地址单元的一个字送入 DX
MOV   BYTE PTR [DI]，45H     ；正确。把字节 3AH 送入 DI 指定的地址单元
MOV   BYTE PTR [D1]，AL      ；  正确。AL 内字节送入 DI 指定的地址单元
MOV   WORD PTR [DI]，45H     ；正确。把字 3AH 送入[DI]单元，将 00H 送入[(DI)+1]单元
MOV   AX，WORD PTR D2        ；正确。从 D2 地址开始读取一个字的数据送 AX
MOV   AL，BYTE   PTR D1      ；正确。从 D1 地址开始读取一个字节的数据送 AL
MOV   DX，D1                 ；与以上的指令功能相同
```

4.2 汇编语言的基本语法

4.2.9 合成运算符

2. THIS运算符

THIS　新类型

【例 4-7】 考察以下程序段中 THIS 运算符的功能。由于伪指令 EQU（有等于含义）的定义不占实际的地址单元，因此变量 DAT1 和 DAT2 的地址是相同的。

```
ORG    4000H                    ；ORG 是定义程序段起始地址的伪指令
DAT1   EQU  THIS  BYTE     ；  定义在此地址段，变量 DAT1 是字节类型
DAT2   DW   1234H，5678H    ；定义在此地址段，变量 DAT2 是字类型
MOV    DX，DAT2  ；编译后的等效语句是 MOV  DX，[4000H] ，执行指令后(DX)=1234H
MOV    CL，DAT1  ；编译后的等效语句是 MOV  CL，[4000H] ，执行指令后(CL)=34H
```

4.2 汇编语言的基本语法

4.2.9 合成运算符

3. 分离运算符

```
MOV    BL,  LOW   0ABCDH    ；执行指令后，(BL)=0CDH
MOV    AH,  HIGH  0ABCDH    ；执行指令后，(AH)=0ABH
```

4. 段操作码

```
MOV   DX, ES: [BX][DI]   ；此指令等同于 MOV   DX, ES: [BX+DI]
```

4.2 汇编语言的基本语法

4.2.10 SHORT运算符

```
LP1: JMP SHORT LP2
...
LP2: MOV CX, AX
```

4.3 汇编程序结构形式

4.3.1 简化段定义格式的汇编程序结构

```
TITLE    <标题>                ；标题可以省略
COMMENT                        *这是一个简化段定义结构实例*
.MODEL SMALL                   ；定义程序的存储模式
.STACK   nH                    ；定义堆栈段，其中 n 为以字节为单位的栈长
                               ；需要写具体值，如 10 个字的堆栈，n 应该写为 14H
.DATA                          ；定义数据段
   …                           ；定义数据
.CODE                          ；定义代码段
START:  MOV   AX，@DATA        ；程序开始
        MOV   DS，AX           ；加载数据段寄存器
         …                     ；程序代码
        MOV   AH，4CH
        INT   21H              ；程序结束点，返回 DOS
         …                     ；程序代码
        END   START            ；结束，程序起始点为标号 START 处
```

4.3 汇编程序结构形式

4.3.2 简化段定义伪指令

1. 置标题伪指令TITLE

[TITLE <标题>]

2. 内存模式伪指令.MODEL

.MODEL　<内存模式>

4.3 汇编程序结构形式

表 4-2 内存模式

存储模型	功能	适用操作系统
Tiny （最小模式）	所有数据和代码都放在一个≤64KB 的段内，访问都为 NEAR 型，并会产生.COM 文件。在 8088 IP 核编程中，常采用 Tiny 模型	MS-DOS
Small （小模式）	代码在一个 64KB 段中，数据在另一个 64KB 段中，通常编写的汇编程序使用小模式就已经足够了	MS-DOS Windows
Medium （中模式）	代码可以大于 64KB，转移或调用可为 FAR 型，但数据只能在一个单一的 64KB 段中	MS-OS Windows
Compact （紧凑模式）	代码在一个单一的 64KB 段中，转移或调用可为 NEAR 型，而数据可以大于 64KB	MS-OS Windows
Large （大模式）	代码和数据都可以大于 64KB，但任何数组不能大于 64KB，数据和代码的访问类型都是 FAR 型	MS-DOS Windows
Huge （特大模式）	代码和数据都可以大于 64KB，数组也可以超过 64KB	MS-DOS Windows
Flat （平展模式）	所有代码和数据放置在一个段中，但段地址是 32 位的，所以整个程序可为 4GB。MASM 6.0 支持该模型	OS/2 WindowsNT

4.3 汇编程序结构形式

4.3.2 简化段定义伪指令

3. 定义堆栈段伪指令**.STACK**

.STACK [<堆栈字节数>]

4. 定义数据段伪指令 **.DATA** .DATA

.DATA? 或 .CONST

4.3 汇编程序结构形式

4.3.2 简化段定义伪指令

5. 定义起始地址伪指令**ORG**

```
ORG   表达式
```

6. COMMENT 成片注释伪指令

```
[COMMENT   * 注释 *]
```

7. 分号的作用

```
MOV   AX，@DATA
MOV   DS，AX
```

4.3 汇编程序结构形式

【例 4-8】简化段定义格式的源程序如下。在显示器上输出显示字符串 Hello World。可用 DOS 9 号功能调用在显示器上输出显示串字符。

```
TITLE   EX4-1
COMMENT                    * 这是一个简化段定义程序实例 *
.MODEL   SMALL             ; 定义存储器类型
.STACK   100H              ; 定义堆栈段
.DATA                      ; 定义数据段
   HW   DB   'Hello World', 13, 10, '$'
.CODE                              ; 定义代码段
START:  MOV   AX, @DATA
        MOV   DS, AX               ; 加载数据段寄存器
        MOV   DX, OFFSET HW        ;DX 指向串首
        MOV   AH, 9                ;DOS 9 号功能调用
        INT   21H                  ; 显示指定字符串
        MOV   AH, 4CH              ; 结束程序返回 DOS 功能调用
        INT   21H
        END   START                ;汇编结束
```

4.3 汇编程序结构形式

【例 4-9】在 FPGA 中基于 8088 IP 软核构建 IBM 微机 SOC 系统，设存储器 RAM 容量为 256B，地址范围是 0000H~00FFH（即系统复位后，第一条被执行指令的地址是 00F0H）。以下程序以一个简单的乘法运算说明简化段定义的程序格式：

```
        title TEST
          .model tiny
          .code
          .8086
      ORG   00010H
      DAT1  DB    56H            ；被乘数
      DAT2  DB    0E7H           ；乘数  56H×E7H=4D9AH
    RESULT  DW    4567H          ；存放乘积。这里预先放置的 4567H 将会被存入的乘积覆盖
   START:   MOV   AL，DAT1
            MOV   BL，DAT2
            MUL   BL              ；相乘后的乘积被存入 AX 中
            MOV   RESULT，AX      ；运算结果放入存储单元
     RND :  JMP     RND
      ORG    00F0H          ;    BOOT CODE，复位后执行指令的起始地址：00F0H
       DB    0EAH           ;  EAH 是长跳转的指令操作码：JMP  FAR PTR START
       DW    START          ; 最后编译结果相当于：JMP 0000：0014 ,指令码：EA00000014H
       DB    00H,00H        ；START 的偏移地址是 0014H
       END                  ;
```

4.3 汇编程序结构形式

4.3.3 完整段定义格式的汇编程序结构

```
; filename.asm
...                                  ; 用 EQU 定义的等价语句区
...                                  ; 若有外部模块，用 EXTRN 定义外部说明
STACK SEGMENT                        ; 定义堆栈段 STACK
...                                  ; 分配堆栈段的大小
STACK ENDS                           ; 堆栈段结束
DATA SEGMENT                         ; 定义数据段 DATA
...                                  ; 定义数据
DATA ENDS                            ; 数据段结束
CODE SEGMENT   'CODE'                ; 定义代码段
ASSUME CS:CODE,DS:DATA,SS:STACK      ; 确定 CS/DS/SS 指向的逻辑段
START:    MOV   AX,DATA
          MOV   DS,AX                ; 加载数据段寄存器
           ...                       ; 程序代码
          CODE   ENDS                ; 代码段结束
          END START                  ; 汇编结束，程序起始点为 START
```

4.3 汇编程序结构形式

4.3.4 完整段定义伪指令

1. 定义逻辑段伪指令SEGMENT/ENDS

<段名> SEGMENT [段定位类型][组合类型][类别]
<汇编语言语句>
<段名> ENDS

表 4-3 段定位类型

类 型	说 明
BYTE	逻辑段的起始地址可以是以字节为单位，即段可以从任何地址开始
WORD	逻辑段的起始地址是以字为单位，即段的起始地址必须为偶数，用二进制表示最低一位是 0
PAGE	逻辑段从页的边界开始，段址是 256 的倍数，用二进制表示低 8 位均为 0
PARA	逻辑段的起始地址是以节（16 个字节）为单位的，即为 16 的倍数，即该段起始物理地址为 XXXX0H，此类型为默认类型

4.3 汇编程序结构形式

4.3.4 完整段定义伪指令

1. 定义逻辑段伪指令SEGMENT/ENDS

表 4-4 组合类型

类　　型	说　　明
PUBLIC	把同名的 PUBLIC 段按连接顺序组合成一个物理段，形成一个更大的段，公用一个段地址，段的总长度是所有同名段的长度和
COMMON	把所有同名同类别的段重叠起来形成共同的物理段址，共享相同的存储区域。通常不同模块采用公用缓冲区时使用这种组合类型。
AT<表达式>	按绝对地址定位，段地址即表达式的值，即把本段装在表达式的值所指定的地址上，这样可直接对内存操作，但代码段不能用 AT 指定
STACK	把所有的 STACK 段连接成一个物理堆栈段，段基址在 SS 中
MEMORY	把本段定位在所有段的上面，即程序的最高地址处。如有多个 MEMORY 段，则遇到的第一个是 MEMORY 段，其它的作为 COMMON 段处理
PRIVATE	默认的设置方式，表示本段与其他逻辑段没有关系，每段均各自设置地址

4.3 汇编程序结构形式

4.3.4 完整段定义伪指令

2. 定义寻址关系伪指令ASSUME

ASSUME <段寄存器名>：<逻辑段名>[，…]

```
MOV    AX，DATA1
MOV    DS，AX              ；加载 DS 寄存器，确定数据段段地址
MOV    AX，DATA2
MOV    ES，AX              ；加载 ES 寄存器
MOV    AX，STACK
MOV    SS，AX              ；加载 SS 寄存器，确定堆栈段段地址
LEA    SP，STACKTOP        ；栈顶存入 SP
```

4.3 汇编程序结构形式

4.3.4 完整段定义伪指令

3. 汇编结束伪指令END

END [表达式]

4. 等价语句伪指令EQU

<标识符> EQU <表达式或字符串>

```
CNT   EQU   4ABH     ; 以下的程序代码中 CNT 就代表常数 4ABH
MOVE  EQU   MOV      ; 此后的语句中可用 MOVE 代替指令助记符 MOV
```

4.3 汇编程序结构形式

4.3.4 完整段定义伪指令

5. 定义过程伪指令PROC/ENDP

<过程名> PROC [NEAR/FAR]
<过程名> ENDP

4.3 汇编程序结构形式

4.3.4 完整段定义伪指令

6. 完整段定义格式的程序实例

【例 4-10】

```
DATA   SEGMENT                          ; 定义数据段 DATA
  HW   DB   'Hello world', 13, 10, '$'      ; 定义数据
  DATA ENDS                             ; 数据段结束
CODE   SEGMENT                          ; 定义代码段
  ASSUME  CS:CODE, DS:DATA              ; 确定 CS 和 DS 指向的逻辑段
START:  MOV   AX, DATA
        MOV   DS, AX                    ; 加载 DS 寄存器
        MOV   DX, OFFSET HW             ; DX 指向串首
        MOV   AH, 9                     ; DOS 9 号功能调用
        INT   21H                       ; 显示指定字符串
        MOV   AH, 4CH                   ; 结束程序返回 DOS 功能调用
        INT   21H
CODE    ENDS                            ; 代码段结束
        END   START                     ;汇编结束，程序起始点为 START
```

4.4 DOS和BIOS功能调用

4.4.1 DOS系统功能调用

表 4-5 常用的 DOS 系统功能调用

INT 21H 功能号	功　能	入口参数	出口参数
01H	带回显的字符输入(单字符输入)		(AL)＝输入字符
02H	字符显示（单字符输出）	(DL)＝输出字符	
09H	字符串显示（字符串输出）	(DS:DX)＝缓冲区首地址	
0AH	字符串缓冲输入(字符串输入)	(DS:DX)＝缓冲区首地址	

4.4 DOS和BIOS功能调用

4.4.1 DOS系统功能调用

【例 4-11】从键盘输入一串字符，字符数小于 20，汇编程序如下：

```
DATA    SEGMENT
    BUF  DB   25
         DB   20
         DB   25   DUP(?)
DATA    ENDS
CODE    SEGMENT
    …
    MOV    AX，DATA   ；数据区段基址赋值
    MOV    DS，AX
    …
    LEA      DX，BUF
    MOV    AH，AH
    INT     21H
    …
CODE    ENDS
```

4.4 DOS和BIOS功能调用

4.4.2 BIOS功能调用

```
MOV     AH, <功能号>
<设置入口参数，即将参数放在指定寄存器中>
INT     n          ；n 是中断类型类型码，n 通常为 10H~1AH
```

1. 从键盘输入单个字符（使用INT 16H中断）

```
MOV   AH，0              ；功能号 0 送 AH
INT   16H                ；返回键盘输入的字符 ASCII 码进入 AL
```

2. 输出一个字符到显示器（使用INT 10H中断）

```
MOV    AL，‘A’    ；AL 中置需要显示的字符码
MOV    BX，0      ；默认方式
MOV    AH，0EH    ；置功能号
INT    10H
```

4.5 汇编语言程序设计

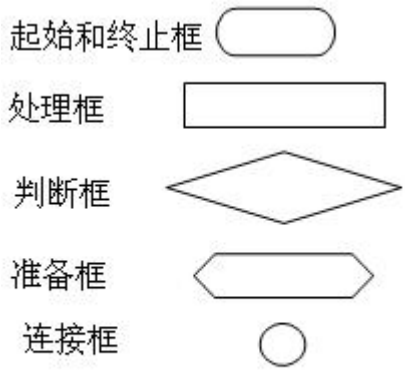

图 4-3 流程图符号

4.5.1 顺序结构程序设计

4.5 汇编语言程序设计

4.5.2 分支结构程序设计

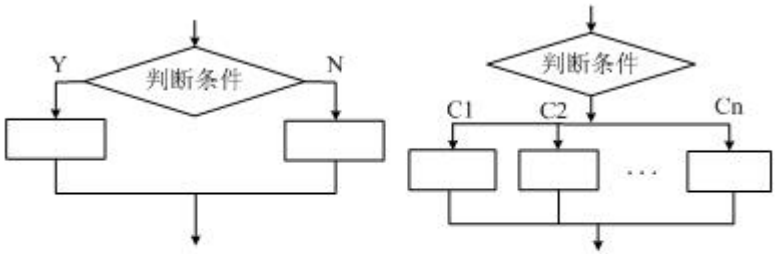

图 4-4 分支程序结构

4.5 汇编语言程序设计

4.5.2 分支结构程序设计

【例 4-12】根据符号函数式 4-1 编写汇编程序，程序根据 X 的值确定函数 Y 的值，并将 Y 值存于指定存储单元中。

$$Y=\begin{cases}1 & \text{当} X>0\text{时}\\ 0 & \text{当} X=0\text{时}\\ -1 & \text{当} X<0\text{时}\end{cases} \qquad (4\text{-}1)$$

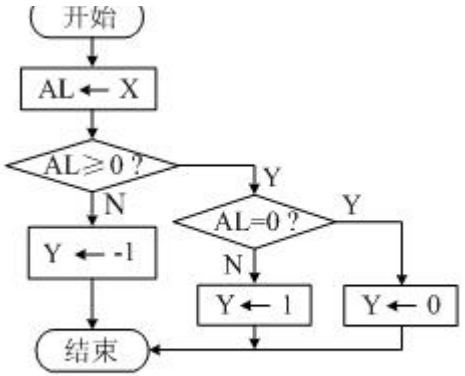

图 4-5 例 4-12 程序流程图

```
DATA   SEGMENT
    XD   DB   -5      ；设 XD 单元存有待判断的值 X=-5
    YD   DB   ?       ；设 YD 单元存储 Y 值
DATA   ENDS
CODE   SEGMENT
    ASSUME   CS:CODE，DS:DATA
START：  MOV   AX，DATA
    MOV   DS，AX
    MOV   AL，XD        ；X 值送 AL
    CMP   AL，0
    JGE   NEXT          ；X 大于等于 0 转 NEXT
    MOV   AL，0FFH      ；X 小于 0，-1 送 AL
    JMP   HALT
NEXT：   JE   ZRO       ；X 等于 0，转 ZRO
    MOV   AL，1         ；  X 大于 0，1 送 AL
    JMP   HALT
ZRO：    MOV   AL，0    ；X 等于 0，0 送 AL
HALT：   MOV   YD，AL ；Y 值送存储单元 YD
    MOV   AH，4CH   ；赋值带返回码后结束功能码
    INT   21H           ；执行完程序后返回 DOS 系统
CODE   ENDS
    END   START
```

4.5 汇编语言程序设计

4.5.2 分支结构程序设计

【例 4-13】试编汇编程序对考试成绩进行统计。设有 20 个学生的成绩分别是：37、47、55、59、61、63、67、72、74、78、73、79、80、83、88、82、91、93、98、100 分。统计要求：分别统计低于 60 分，及处于 60~69、70~79、80~89、90~99 分之间及 100 分的人数，并存放到 RLS60、RLS69、RLS79、RLS89、RLS100 及 RE100 存储单元中。

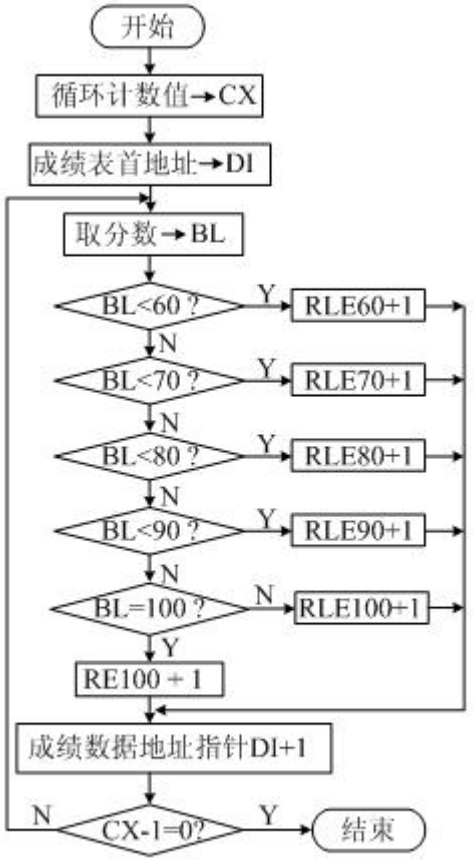

图 4-6 例 4-13 成绩统计流程图

```
 DATA  SEGMENT
  RCD  DB 37, 47, 55, 59, 61, 63, 67
       DB 72,74, 78, 73, 79, 80, 83
       DB 88, 82, 91, 93, 98, 100
   RLS60    DB 0
   RLS70    DB 0
   RLS80    DB 0
   RLS90    DB 0
   RLS100   DB 0
   RE100    DB 0
DATA  ENDS
CODE  SEGMENT
ASSUME  CS : CODE, DS : DATA
 START:  MOV  AX, DATA
         MOV  DS, AX
         MOV  CX, 20
         LEA  DI, RCD
STEPCMP :  MOV  BL,[DI]
         CMP  BL,60
         JL   RL6
         CMP  BL,70
         JL   RL7
         CMP  BL,80
         JL  RL8
         CMP  BL,90
         JL  RL9
         CMP  BL,100
         JNE  RL10
         INC  RE100
         JMP  NEXT_RCD
RL10 :  INC  RL100
         JMP  NEXT_RCD
RL9 : INC  RLS90
         JMP  NEXT_RCD
RL8 : INC  RLS80
         JMP  NEXT_RCD
RL7 : INC  RLS70
         JMP  NEXT_RCD
RL6  : INC RLS60
NEXT_RCD : INC DI
         LOOP STEPCMP
         MOV  AH,4CH
          INT  21H
CODE  ENDS
          END  START
```

4.5 汇编语言程序设计

4.5.3 循环结构程序设计

1. 循环程序的结构

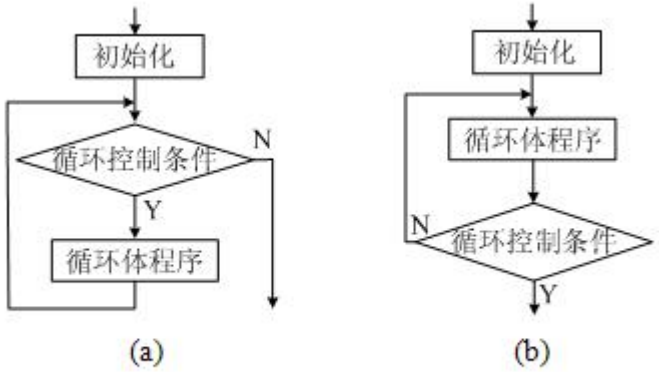

图 4-7 不同循环程序结构的流程图

4.5 汇编语言程序设计

4.5.3 循环结构程序设计

2. 循环程序的控制方法

① 计数控制循环法

② 条件控制循环法

4.5 汇编语言程序设计

4.5.3 循环结构程序设计

3. 循环程序示例

【例 4-14】设计一个程序，完成从 1 连加到 100 的操作，结果放在数据段 SUM 单元。要求使用图 4-7(b)的 DO UNTIL 结构形式的循环控制流程。

4.5 汇编语言程序设计

```
DATA   SEGMENT
 SUM   DW   0
DATA   ENDS
CODE   SEGMENT
ASSUME   CS : CODE, DS : DATA
  START:   MOV   AX，DATA
           MOV   DS，AX
           XOR   AX，AX          ；累加器 AX 清 0
           MOV   CX，100         ；计数值 100 送计数寄存器
  LP1   :  INC   AX
           ADD   SUM，AX
           DEC   CX              ；计数器减 1
           JNZ   LP1             ；若(CX)不等于 0，则转 LP1，继续执行循环程序
           MOV   AH，4CH         ；(CX)=0，终止循环，返回 DOS
           INT   21H             ；用 DEBUG 断点调试测试可知最后结果是=13BAH=5050
    CODE   ENDS
           END   START
```

【例 4-15】 从自然数 2 开始累加，累加序列是 2+4+6+…，直到累加和大于 2000。要求统计所累加的偶数的个数，并把统计的个数送入 EN 单元，把累加和送入 SUM 单元。

```
DATA   SEGMENT
 EN    DW   ?
SUM   DW   ?
DATA   ENDS
CODE   SEGMENT
ASSUME   CS:CODE，DS:DATA
START :    MOV   AX，DATA
           MOV   DS，AX
           MOV   BX，1        ；BX 纪录计数过程中偶数的个数
           MOV   AX，2        ；AX 作为累加和寄存器
           MOV   DX，2        ；作为加数，DX 生成偶数
 LOOP 1:   INC    BX
           ADD   DX，2        ；生成偶数
           ADD   AX，DX       ；累加
           CMP   AX，2000     ；
           JBE   LP           ；  AX 小于等于 2000，则转 LP
           MOV   EN，BX       ；把参与累加的偶数的个数值送 EN 缓
冲单元
           MOV   SUM，AX      ；把累加和送 SUM 缓冲单元
           MOV   AH，4CH      ；最后结果是：(AX)=816H，(BX)=2DH
           INT   21H
CODE       ENDS
END        START
```

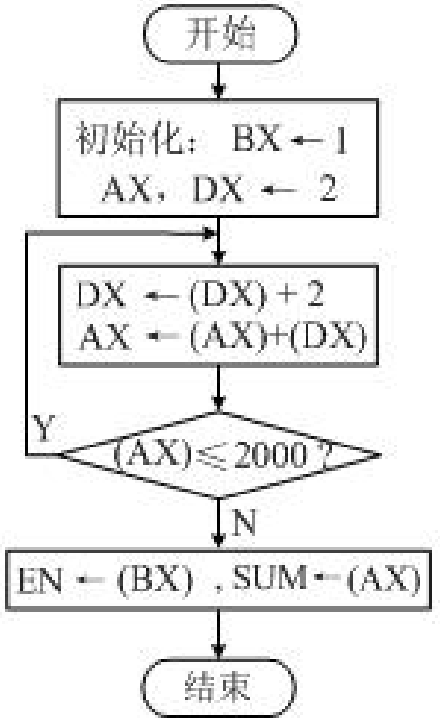

图 4-8 例 4-15 流程图

4.5 汇编语言程序设计

4.5.3 循环结构程序设计

4. 多重循环程序设计

① 内循环必须完整地包含在外循环内，内外循环不能相互交叉；

② 嵌入在外循环体中的内循环可以数个内循环并列存在，也可逐层嵌入；

③ 可以从内循环中直接跳到外循环，但不允许从外循环直接跳入内循环；

④ 应该防止出现“死循环”。无论是外循环还是内循环，一定不能使循环返回到初始值设置部分。

⑤ 每次由外循环再次进入内循环时，初始条件须有所改变，以免进入“死循环”。

【例 4-16】 编写程序对内存单元 DBUF 开始的以字节为单位的 M（=20）个无符号数进行从小到大排序。

```
DATA   SEGMENT
   M    EQU  20            ; 数组中数据的个数
DBUF   DB  6，180，68，178，46，84，4，186，26，112
       DB  100，49，172，22，88，222，149，180，14，60
DATA   ENDS
CODE   SEGMENT
ASSUME  CS：CODE，DS：DATA
START:  MOV  AX，DATA
        MOV  DS，AX
        MOV  DX，19  ;   M-1=19
   S1:  MOV  BL，0
        MOV  CX，DX
        MOV  SI，OFFSET DBUF
   S2:  MOV  AL，[SI]      ; 内循环，将数组中的数据送 AL
        CMP  AL，[SI+1]    ; 与下一数据比较
        JNA  NXC           ; 若前一数据小于后一数据转 NXC
       XCHG  [SI+1]，AL    ; 后一数据大，交换位置
       XCHG  [SI]，AL      ; 送回存储单元
        MOV  BL，1         ;  置 1，表示数据发生过交换
  NXC:   INC  SI           ; 数据单元地址加 1
        LOOP  S2     ; 返回内循环
        DEC  DX            ; 减去已比较换位的数据
        CMP  BL，00H  ;  检测此轮内循环是否有数据发生交换
        JNE  S1       ; 若 BL 中交换标志不为 0，表示此内循环数据有交换，转 S1，继续换位
        MOV  AH，4CH  ; 否则结束数据比较和换位
        INT  21H
CODE   ENDS
 END   START
```

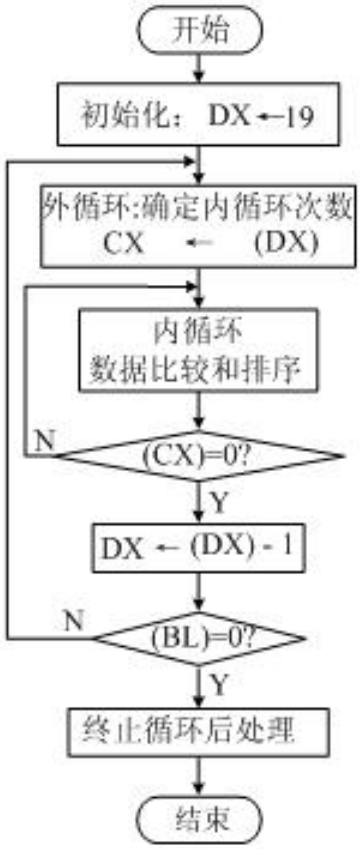

图 4-9 例 4-16 程序流程

4.5 汇编语言程序设计

4.5.4 子程序设计

1. 子程序的定义和调用

```
过程名   PROC    类型
         ......
         RET
过程名   ENDP
```

```
CODE1   SEGMENT
        …
        CALL   SUBPRO1          ；调用子程序 1
        …
        CALL   FAR   PTR   SUBPRO2     ；调用子程序 2
        …
SUBPRO1   PROC                  ；子程序 1
        …
          RET
SUBPRO1   ENDP
        …
CODE1   ENDS
        …
CODE2   SEGMENT
        …
SUBPRO2   PROC   FAR     ；子程序 2
        …
          RET
SUBPRO2   ENDP
        …
CODE2   ENDS
```

4.5 汇编语言程序设计

4.5.4 子程序设计

2. 寄存器内容的保护和恢复

```
SUBPRO  PROC  FAR
        PUSH  AX      ; 保护 AX 的内容
        PUSH  BX      ; 保护 BX 的内容
        PUSH  CX
        PUSH  DX
        PUSH  DI      ；注意压栈和出栈数据的先后次序
        …
        POP   DI
        POP   DX
        POP   CX
        POP   BX      ; 恢复 BX 的内容
        POP   AX      ; 恢复 AX 的内容
        RET
SUBPRO  ENDP
```

4.5 汇编语言程序设计

4.5.4 子程序设计

3. 子程序的嵌套与递归调用

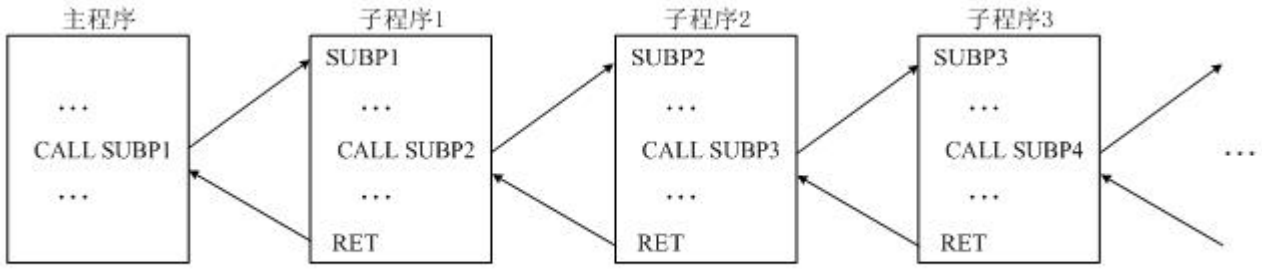

图 4-10　子程序嵌套示意图

4.5 汇编语言程序设计

4.5.4 子程序设计

4. 子程序调用示例

【例 4-17】在某存储单元中存放着 4 位十六进制数值，试编写程序将此 4 位数值分别转换为对应的 ASCII 码，并依次存放到指定的 4 个存储单元中。要求用子程序实现。

```
STACK    SEGMENT              ；若不定义堆栈，则采用 DOS 默认堆栈
         DW 128 DUP    (?)
STACK    ENDS
DATA     SEGMENT
 ASC4    DB   4 DUP(?)        ；准备好存放 4 个 ASCII 码的字节单元
 HEX4    DW   89ABH           ；设 HEX4 单元存放有待转换的 4 位 16 进制数值
 DATA    ENDS
 CODE1   SEGMENT              ；定义第一个代码段，存放主程序
ASSUME   CS: CODE1，DS: DATA
```

```
START :   MOV  AX. DATA
          MOV  DS. AX
          MOV  AX. STACK
          MOV  SS. AX
          MOV  CX. 4
          LEA    DI. ASC4
          MOV  BX. HEX4     ; 将待转换的 4 位 16 进制数据送入 BX
  NEXT:   MOV  AX. BX
          AND  AL. 0FH
          CALL FAR PTR HEX2ASC    ; 长调用子程序 HEX2ASC
          MOV  [DI]. AL           ; AL 中有子程序的出口参数，即已转换好的数据
          INC  DI
          PUSH CX           ; 由于右移指令要用到 CX，将 CX 中的计数值入栈保护
          MOV  CX. 4        ; 送入移位位数
          SHR  BX. CL       ; 右移 4 位
          POP  CX           ; 弹出控制循环次数的寄存器
          LOOP   NEXT       ; 如果(CX)不等于 0，继续循环。
          MOV  AH. 4CH
          INT  21H
  CODE1   ENDS
  CODE2   SEGMENT                ; 定义另一代码段
 ASSUME   CS: CODE2
HEX2ASC   PROC  FAR       ; 在另一代码段中的子程序的类型须定义为 FAR
          CMP  AL. 0AH
          JB     ADD30    ; 如果数值小于 AH，只需直接加 30H 即转换成对应的 ASCII 值
          ADD  AL. 07H    ; 如果数值大于等于 AH，需加 37H
ADD30 :   ADD AL. 30H
          RET
HEX2ASC   ENDP
  CODE2   ENDS
          END  START
```

【例 4-18】 递归子程序可对应数学上对递归函数的定义，所以利用递归子程序完成相关算法，可完成较复杂的计算。此例以一个无符号数 N（=8）的阶乘计算为例，说明递归子程序的设计方法。即编制计算：N! =N*（N-1）*（N-2）*…*1 的程序。

```
DATA   SEGMENT
    N    DW   8        ；定义 N 值=8
  RSV    DW   ?        ；存放结果
 DATA    ENDS
 CODE    SEGMENT
ASSUME    CS : CODE，  DS : DATA
START:    MOV    AX，DATA
          MOV    DS，AX
          MOV    AX，N        ；AX←N 值
          CALL   MULN         ；调用 N!递归子程序
          MOV    RSV，AX
          HLT
 MULN     PROC                ；递归子程序
          PUSH   AX
          CMP    AX，0
          JE     NE0      ；N= 0 时退出子程序
          DEC    AX
          JNZ    NN0
 NE0 :    POP    AX
          MOV    AX，  1    ；0!= 1
          JMP    HM
 NN0 :    CALL   MULN
          POP    BX
          MUL    BX
  HM :    RET
 MULN     ENDP
 CODE     ENDS
          END    START
```

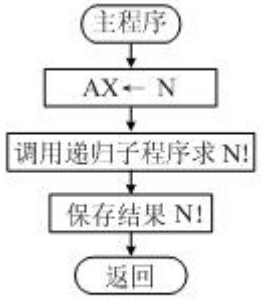

图 4-11 主程序流程图

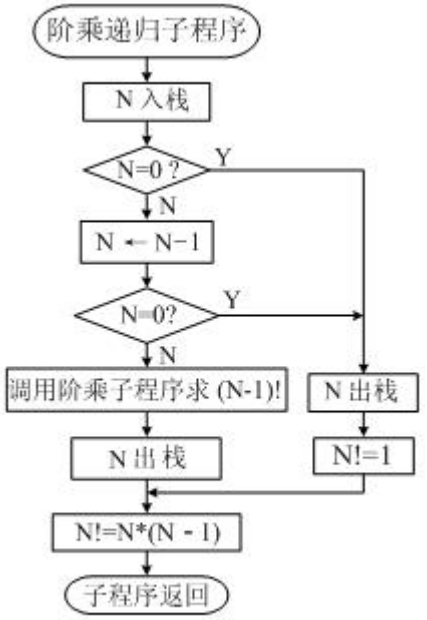

图 4-12 求 N!子程序流程图

4.6 宏指令

宏指令名　MACRO　[形式参数 1，形式参数 1，…]
　　<宏定义体>
　　ENDM

宏指令名　[实际参数 1，实际参数 1，…]

```
DAD   DL，BUF0
DAD   BUF1，BUF2    ；注意，实际参数的名称可以不同
```

```
DAD     MACRO    X，Y
        MOV  AL，X
        ADD  AL，Y
        DAA
        MOV  X，AL
    ENDM
```

```
MOV   AL，DL
ADD   AL，BUF0
DAA
MOV   DL，AL
MOV   AL，BUF1
ADD   AL，BUF2
DAA
MOV   BUF，AL
```