第9章 宏汇编及其它高级伪操作

宏汇编语言提供了类似于高级语言的某些复杂功能，如宏汇编、重复汇编与条件汇编，提高了使用汇编语言进行程序设计的质量和效率。

9.1 宏汇编

- 宏是源程序中一段有独立功能的程序代码。它只需要在源程序中定义一次，就可以多次调用，调用时只需要用一个宏指令语句就可以了。

- 宏功能既可以实现程序复用，又能方便的传递多个参数。

- 子程序优点：
  - 省存储空间
  - 优化程序结构
  - 便于调试和修改
- 子程序缺点：
  - 调用、返回、保存、恢复需花开销
  - 参数传递复杂
- 因此，在程序较短，或需要传送的参数较多的情况下，用宏汇编更加有利。

9.1.1 宏定义、宏调用和宏展开

- 宏定义：宏指令名 **MACRO** [形参1,形参2,…]
  <宏定义体>
  ENDM

- 宏定义体是一组有独立功能的程序代码。

- 宏指令名给出宏定义的名称，调用时就使用宏指令名来调用宏定义。第一个符号必须是字符。

- 哑元表给出了宏定义中所用到的形式参数，每个哑元之间用逗号隔开。

- 宏调用：宏指令名 [实参1,实参2,…]
- 宏指令被定义后，在源程序中就可以直接调用，称为宏调用。
- 一般说来，实参的个数应和形参的个数相等，但汇编程序并不要求它们必须相等。若实参个数大于形参个数，则多余的实参不予考虑，若实参个数小于形参个数，则多余的形参取空值。

- 宏展开：在对源程序的汇编阶段，汇编程序对源程序中的每个宏调用都进行了宏展开，即用宏定义体取代源程序中宏指令名，用实参取代形参。

- 应该注意，用实参取代形参后，所得到的语句应该是有效的，否则汇编程序将会指示出错。

- 例9.1 用宏指令实现两数的相加。

- 宏定义：

```
sumn   MACRO   x, y, result
               mov ax, x
               add  ax, y
               mov result, ax
               ENDM
```

- 宏调用：

  sumn 34,25, bx

- 宏展开：

  1 mov ax,34

  1 add ax,25

  1 mov bx, ax

- 例9.2 用宏指令实现两个八位有符号数的乘法。

- 宏定义：

```
imultiply   MACRO   x,y,result
            push ax
            mov al,x
            imul y
            mov result,ax
            pop  ax
            ENDM
```

- 宏调用：

```
imultiply   cl,dl, [bx]
…
imultiply   ary,var,save
```

- 宏展开：

```
1   push ax
1   mov  al,cl
1   imul  dl
1   mov  [bx], ax
1   pop  ax
      ⋮
1   push ax
1   mov al,ary
1   imul var
1   mov save,ax
1   pop  ax
```

- 例9.3 某工厂工人的周工资由计时工资和计件工资组成，计时工资按每小时工资率RATE乘以工作小时数计算；计件工资按超定额部分乘以SUP计算（超定额=实际完成的工件数MADE-定额工件数PART），工资总额放在WAGE中。

- 宏定义如下：

```
wages  macro  rate, hout, made, part, sup
       wage=rate*hout+(made-part)*sup
       endm
rate=5
part=100
sup=4
```

- 宏调用:

  wages rate,40,120,part,sup

- 宏展开为:

  wage=rate*40+(120-part)*sup

- 从以上两个例子可以看出,子程序和宏指令具有类似的功能,但又有以下区别:
  - 空间的区别：宏指令并不节省目标程序的空间，而子程序在目标程序中只有一段；
  - 时间的区别：宏指令在运行时不需要其他额外的CPU开销，而子程序的调用、返回和保护现场、恢复现场需要占用时间；
  - 参数的区别：宏调用可实现多个参数的直接代换，方式简单灵活，而子程序参数传递麻烦。

9.1.2 宏定义的嵌套

- 这种嵌套结构的特点是外层宏定义的宏体中又有宏定义，只有调用外层宏定义一次后，才能调用内层宏指令。

- 例9.4　用嵌套的宏定义实现两个八位数的算术运算。

- 宏定义：

```
math   MACRO  mathname, action, num
       mathname   MACRO  x, y, resule
                  push ax
                  mov num,x
                  action y
                  mov result ax
                  pop ax
                  ENDM
        ENDM
```

- 宏调用:

  math imultiply , imul, al

- 宏展开:

```
imultiply    MACRO  x, y, result
             push ax
             mov  al, x
             imul  y
             mov result, ax
             pop ax
             ENDM
```

- 宏调用:

  math divide, div, ax

- 宏展开:

```
divide   MACRO  x, y, result
         push ax
         mov  ax, x
         div  y
         mov result, ax
         pop ax
         ENDM
```

- 接下来可使用宏调用:

  divide ary, var, save

- 则宏展开如下：

```
1   push ax
1   mov  ax, ary
1   div   var
1   mov  save, ax
1   pop   ax
```

9.1.3 宏定义中使用宏调用

- 宏定义中使用的宏调用必须已经定义。

- 例9.5　用宏指令实现显示字符。
- 宏定义：

```
INT21   MACRO   FUNCTN
        MOV     AH，FUNCTN
        INT     21H
        ENDM
DISPC   MACRO   CHAR
        MOV     DL，CHAR
        INT21    2
        ENDM
```

- 宏调用：
  DISPC　‘A’

- 宏展开：
  1　　MOV　DL，‘A’
  2　　MOV　AH，2
  2　　INT　 21H

- 这里的2表示第二层展开结果。

9.1.4 带间隔符的实参

- 在宏调用中，有时实参使用的是字符串（不是单引号括起来的），而是字符串中包含间隔符（如空格、逗号等），为使间隔符成为实参的一部分，则要用尖括号将字符串括起来作为一个实参的整体来替换形参。

- 例9.6　在数据段中定义40个字节的存储空间。

```
defdb  macro  buf,x
       buf  x
       endm
 data  segment
       defdb  array,<db 40 dup(?)>
 data  ends
 …
```

宏展开为：

```
       1   array  db  40  dup(?)
```

9.1.5 连接操作符&

- 用连接操作符&可连接实参，形成一个完整的符号或字符串。

- 例9.7　用操作符&连接实参，生成指令中的操作码。

- 宏定义：

```
shift   MACRO   rig, m, n
            mov cl,n
            s&m  rig, cl
        ENDM
```

- 宏调用：

  shift ax, hl, 4

  shift dx, hr, 2

- 宏展开：

  1 mov cl,4

  1 shl ax,cl

  1 mov cl, 2

  1 shr dx, cl

- 例9.8　用操作符&连接实参，生成指令中的操作数。

- 宏定义：

```
student   macro   rec，n，name，tel
          rec    db   &n
          rec1   db ‘&name&,&tel’
          rec2   db ‘&computer’
          endm
```

- 宏调用：

  student　msg，1，wang, 12345678

- 宏展开：

  1　msg　db　1

  1　rec1　db ‘wang,12345678’

  1　rec2　db ‘&computer’

9.1.6 宏替换操作符%

- 宏替换：汇编程序把跟在%后的表达式的值转换为当前基数下的数，在展开期间用这个数来取代形参（哑元）。
- 格式：% 表达式
- 例9.9 用操作符%将实参的值替换形参。

- 宏定义：

```
student   macro   rec，n，msg
          rec&n   db   msg
          endm
```

- 宏调用：

```
n=1
student   numb，%n，  ‘wang,12345678’
n=n+1
student   numb，%n，  ‘zhou,56781234’
```

- 宏展开：

```
 1    numb1  db   ‘wang,12345678’
 1    numb2  db   ‘zhou,56781234’
```

9.1.7 LOCAL伪操作

- 在宏定义中，常常使用标号，当多次宏调用后，就会出现标号重复定义的错误。
- 显然在程序中多次调用该宏定义时，则展开后会出现标号的多次重复定义，这是不允许的。为此，系统提供了**LOCAL**伪操作。
- 例**9.10**　宏定义体中使用标号。

错误形式

- 宏定义：

```
cmpdata   macro   r1，r2，max
            cmp     r1，r2
            jge     mr1
            mov     max，r2
            jmp     mr2
      mr1：  mov     max，r1
      mr2：
            endm
```

- 宏调用：

```
cmpdata   ax，bx，var
cmpdata   dx，cx，value
```

- LOCAL伪操作格式：LOCAL 符号[,符号]
- 汇编程序对LOCAL伪操作定义的符号说明为局部标号，对每一个局部标号建立唯一的符号（用？？0000～？？FFFF）来代替每个局部标号。
- 注意：LOCAL伪操作只能用在宏定义体内，而且必须是MACRO伪操作后的第一个语句。

正确形式

- 宏定义：

```
cmpdata   macro  r1，r2，max
          local  mr1，mr2
          cmp    r1，r2
          jge    mr1
          mov    max，r2
          jmp    mr2
    mr1：  mov    max，r1
    mr2：
          endm
```

- 宏调用：

```
cmpdata   ax，bx，var
cmpdata   dx，cx，value
```

- 宏展开：

```
1                cmp    ax，bx
1                jge    ??0000
1                mov    var，r2
1                jmp    ??0001
1    ??0000：    mov    var，r1
1    ??0001：
1                cmp    dx，cx
1                jge    ??0002
1                mov    var，r2
1                jmp    ??0003
1    ??0002：    mov    var，r1
1    ??0003：
```

9.1.8 使用宏库文件

- 如果程序中定义了多个宏，用户可以把它们集中建立在一个独立的宏库文件中。
- 宏库文件为文本文件，扩展名可为**MAC**。
- 用**INCLUDE**伪指令加入宏库文件，格式：
  INCLUDE 宏库文件名
- 宏库文件和当前源程序中定义的标识符不能重复。

- 例9.11 宏库文件STDIO.MAC 是关于输入/输出的文件。内容如下:

```
cr  equ  13
lf  equ  10
getchar    macro
           mov     ah, 1
           int     21h
           endm
putchar    macro   asc
           mov     ah, 2
           mov     dl, asc
           int     21h
           endm
prints     macro   msg
           mov     ah, 9
```

```
            mov     dx, offset  msg
            int     21h
            endm
inputs      macro   conbuf
            mov     ah, 10
            mov     dx, offset  conbuf
            int     21h
            endm
crlf        macro
            putchar  cr
            putchar  lf
            endm
exit        macro
            mov     ah，4ch
            int     21h
            endm
```

- 例9.12 在程序中加入宏库文件STDIO.MAC，并使用其中的宏指令。程序内容如下:

```
include  stdio.mac
data     segment
          string  db  16, ?,16 dup(?)
          msgbox  db  '输入字符串请用$结束',13,10,'$'
data     ends
code     segment
          assume  cs:code,ds:data
start:
          getchar                ；输入一个字符
          crlf                   ；输出回车换行
```

```
        putchar  'a'          ; 输出一个字符
        crlf                  ; 输出回车换行
        inputs   string       ; 输入字符串
        crlf                  ; 输出回车换行
        prints   string+2     ; 输出字符串
        exit                  ; 退出程序
code    ends
        end  start
```

9.2 其它高级伪操作

9.2.1 PURGE伪操作

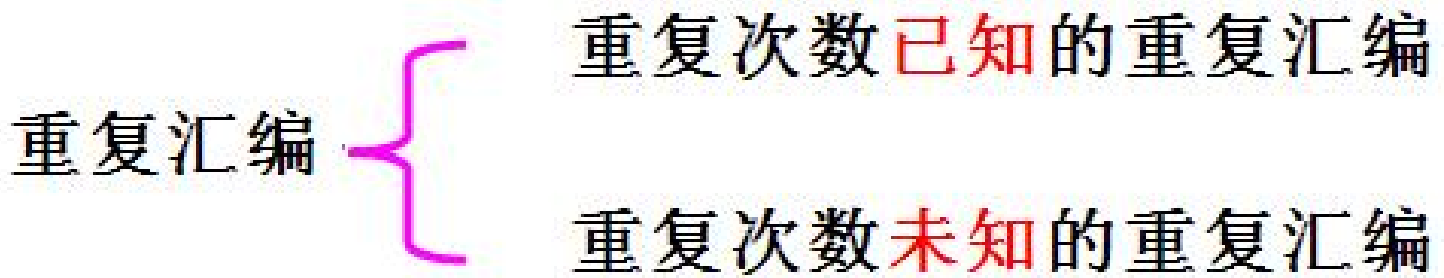
- 一个宏定义可以用**PURGE**伪操作来取消，然后可以再重新定义。
- 格式：**PURGE** 宏指令名 [，宏指令名]
- 取消宏定义的作用是使该宏定义成为空，如果程序中对已被取消宏定义进行宏调用，汇编程序则忽略该宏调用，不会进行宏展开。

9.2.2 列表伪操作

- 列表伪操作可以控制其后的宏调用是否在列表文件中出现宏展开，并不影响宏展开的实际产生。
- **.LISTMACRO\XALL:**列出产生目标码的宏展开，默认情况。
- **.LISTMACROALL\LALL:**列出包括注释在内的所有宏展开。
- **.NOLISTMACRO\SALL:**不列出任何宏展开。

9.2.3 重复汇编

- 有时程序中，需要得到连续相同的或者格式相同的一组代码，这时可使用重复汇编。



1. 重复伪操作

- 格式：REPT 表达式
  …　（重复块）
  ENDM

- 表达式的值表示重复块的重复次数。

- 例9.13　在数据段产生字节数据，首地址为ARRAY。

- 宏定义：

```
X=0
ARRAY  LABEL  BYTE
        REPT  99
        DB  X
        X=X+1
        ENDM
```

- 汇编后产生：

```
1  DB   0
1  DB   1
1  DB   2
    …
1  DB   98
```

- 例9.14 在代码段产生一组代码，该组代码的功能是从键盘输入9个字符，放入数组ARRAY中。程序如下：

```
getchar  macro
         mov  ah,1
         int   21h
         endm
data     segment
         array  db  10  dup(?)
data     ends
code     segment
         assume  cs:code,
ds:data
start:
         mov  ax, data
         mov  ds, ax
         n=0
         rept  9
         getchar
         mov  array+n, al
         n=n+1
         endm
         mov  ah, 4ch
         int   21h
code  ends
         end  start
```

- 重复汇编中使用了宏调用，这部分经汇编后产生：

```
2    mov  ah,1
2    int   21h
1    mov  array+n,al
1    n=n+1
2    mov  ah,1
2    int   21h
1    mov  array+n,al
1    n=n+1
       …
2    mov  ah,1
2    int   21h
1    mov  array+n,al
1    n=n+1
```

2. 不定次数的重复汇编伪操作

（1）IRP伪操作

- 格式：IRP 形参，〈实参1,实参2,…,实参N〉

  …　（重复块）

  ENDM

- 功能：宏汇编程序将重复块的代码重复汇编，重复的次数由实参个数确定。注意：将实参表用尖括号括起来，实参可以是常数、符号、字符串。

- 例9.15　用不定次数的重复汇编在数据段产生字节数据，首地址为ARRAY。

```
array  label  byte
       irp   x,<3,5,7,22,6,8,19>
       db  x
       endm
```

- 汇编后产生：

```
1   db   3
1   db   5
1   db   7
1   db   22
1   db   6
1   db   8
1   db   19
```

（2）IRPC伪操作

- 格式：IRPC 形参，字符串
  …　（重复块）
  ENDM

- 功能：宏汇编程序将重复块重复汇编，重复的次数由字符串的字符个数确定。并在每次重复时，依次用相应位置的字符代换形参。字符串不需用引号，可以用尖括号括起来。

- 例9.16　用IRPC伪操作在数据段产生字节数据，首地址为ARRAY。

```
array  label  byte
        irpc   x,5678
        db  x
 endm
 irpc   x,<1234>
        db  x
 endm
 char='a'
        irpc   n,abcd
        n  db  char
        char=char+1
        endm
```

汇编后产生：

```
1  db  5
1  db  6
1  db  7
1  db  8
1  db  1
1  db  2
1  db  3
1  db  4
1  a  db  61
1  b  db  62
1  c  db  63
1  d  db  64
```

- 例9.17　用IRPC伪操作定义寄存器清0指令。

```
irpc    reg, abcd
mov   reg&x, 0
endm
```

汇编后产生：

```
1  mov   ax, 0
1  mov   bx, 0
1  mov   cx, 0
1  mov   dx, 0
```

9.2.4 条件汇编

- 让汇编程序根据某些条件是否成立来决定是否把一段汇编语句包括在程序中或者排除在外，满足条件的那部分语句生成目标代码。可以用条件汇编伪操作来实现。

- 一般格式：

  IF 条件表达式
  <语句体1>
  [ELSE]
  <语句体2>
  ENDIF

- 功能: 若条件成立,则汇编<语句体1>中的语句;否则对<语句体2>进行汇编。

- 条件汇编伪指令有八条：

  IF　表达式　　；表达式不为0，则满足条件

  IFE　表达式　　；表达式为0，则满足条件

  IFDEF　符号　　；符号已定义或被说明为EXTRN，则满足条件

  IFNDEF　符号　　；符号未定义或未说明为EXTRN，则满足条件

  IFB　<变量>　　；变量为空,则满足条件

  IFNB　<变量>　　；变量不为空,则满足条件

  IFIDN　<串变量1>,<串变量2>　　；两串相等,则满足条件

  IFNIDN　<串变量1>,<串变量2>　　；两串不等,则满足条件

- 表达式可以是用关系操作符EQ，NE，LT，LE，GT，GE和逻辑运算符AND，OR连接的布尔表达式。

- 例9.18　用宏指令MAX把三个变元中最大值放在AX中。变元个数不同产生的程序段也不同。

```
max  macro  k, a, b, c
       local  next,out1
       mov   ax, a
       if (k ge 2) and (k le 3)
            if  k  eq  3
              cmp  c,ax
              jle  next
              mov  ax,c
            endif
next: cmp  b, ax
       jle   out1
       mov  ax, b
       endif
out1:endm
```

- 宏调用：

```
max  1,x
max  2,x,y
max  3,x,y,z
```

- 宏展开：

```
        max  1,x
1       mov  ax, x
1  ??0001:
        max  2,x,y
1       mov  ax, x
1  ??0002:
1       cmp  y, ax
1       jle   ??0003
1       mov  ax, y
1  ??0003:
```

```
        max  3,x,y,z
1       mov  ax, x
1       cmp  z, ax
1       jle   ??0004
1       mov  ax, z
1  ??0004:
1       cmp  y,ax
1       jle   ??0005
1       mov  ax, y
1  ??0005:
```

- 例9.19 求K的阶乘，结果放在AX中。宏定义和子程序一样，也可以递归调用。用条件伪操作可结束宏递归。

```
pow  macro  k
       pop  ax
       mov  bl, k
       mul   bl
       push  ax
       k=k-1
       if  k  ge  1
          pow  k
      endif
endm
```

- 宏调用：

```
mov  ax,1
push  ax
n=4
pow  n
```

- 宏展开：

```
1   pop  ax
1   mov  bl, n
1   mul   bl
1   push  ax
1   n=n-1
2   pop  dx
2   mov  bl, n
2   mul   bl
2   push  ax
2   n=n-1
3   pop  ax
3   mov  bl, n
3   mul   bl
3   push  ax
3   n=n-1
4   pop  ax
4   mov  bl, n
4   mul   bl
4   push  ax
4   n=n-1
```