哈爾濱工業大學(深圳)

HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

汇编语言程序设计
第9讲：高级汇编语言技术

裴文杰

第9讲：高级汇编语言技术

宏汇编

重复汇编

库的使用

wenjiecoder@outlook.com

宏汇编

重复汇编

库的使用

wenjiecoder@outlook.com

宏汇编

什么是宏?

宏（或宏指令）是源程序中一段有独立功能的程序代码，只需定义一次，可以多次调用。

为什么使用宏汇编?

在编制汇编语言程序过程中，有些功能程序段需要多次重复使用，所不同的只是参与操作的操作数。使用宏指令语句可以减少程序书写错误，缩短源程序长度，使源程序编写像高级语言一样清晰、简洁。特别是使用宏库后，可以提高编程效率。

子程序的缺点:
为调用子程序及返回、保存及恢复寄存器以及参数的传送等都要增加程序的开销。对于子程序本身较短或者需要传送的参数较多的情况下，使用宏汇编更加方便。

为了减少编程的工作量，通常采用两种方法:

①将程序段编写为独立的子程序;

②将程序段定义成宏。

wenjiecoder@outlook.com

宏汇编

宏定义：

由伪指令**MACRO**与**ENDM**实现。

形式如下：

```
宏指令名   MACRO   [形式参数表]
    ...                ；宏体
    ...                ；宏体
       ENDM（和子程序不同，ENDM之前不用写宏名）
```

说明：

- 宏指令名由编程者自定，但必须符合标号的命名规则。
- MACRO和ENDM是一对伪指令，分别表示宏定义的开始和结束。
- 宏体必须是指令、伪指令及宏指令构成的程序段。

实参和形参的个数可以不等，若调用时的实参个数多于形参个数，则多余的部分被忽略；若实参个数小于形参个数，则多余的形参假定为空（NULL）。

宏汇编

例：以下宏定义所定义的宏指令AX10可以实现寄存器AX内容乘以10的功能。（假设不溢出）

```
AX10  MACRO              ；宏名  AX10
      PUSH      DX
      SAL       AX, 1
       MOV      DX, AX
      SAL       AX, 1
      SAL       AX, 1
       ADD      AX, DX
      POP       DX
       ENDM
```

宏汇编

宏调用和宏展开

宏调用的格式：

宏指令名 [实参表]

说明：

- 宏指令名所指定的宏指令的定义必须放在该宏调用之前。
- 实参表通常与宏定义中形参表相对应。当需要使用多个实参时,各实参之间要用逗号分隔。
- 实参可以为空，也可以是常数、寄存器、存储单元、地址表达式、指令的操作码或者是操作码的一部分。

注意:宏应该先定义,再调用.

- 宏定义可以放在程序的任何地方（调用处之前），一般建议把宏定义放在程序的最前面。

```
xyz  MACRO  x,y,z
   MOV AL,x
   MUL y
   MOV z, AX
   ENDM
DATAS SEGMENT
   x db 10
   y db 20
   z dw 0
DATAS ENDS
CODES SEGMENT
   ASSUME CS:CODES
START:
   MOV AX,DATAS
   MOV DS,AX
   xyz x,y,z
   MOV AH,4CH
   INT 21H
CODES ENDS
   END START
```

```
DATAS SEGMENT
   x db 10
   y db 20
   z dw 0
DATAS ENDS
CODES SEGMENT
   ASSUME CS:CODES,DS:DATAS
START:
   MOV AX,DATAS
   MOV DS,AX
   xyz x,y,z
   MOV AH,4CH
   INT 21H
CODES ENDS
   END START
xyz  MACRO  x,y,z
   MOV AL,x
   MUL y
   MOV z, AX
   ENDM
```

```
GMENT
DS
GMENT
 CS:CODES,DS:DATAS
O  x,y,z
,x
AX
,DATAS
,AX
,4CH
DS
RT
```

宏展开

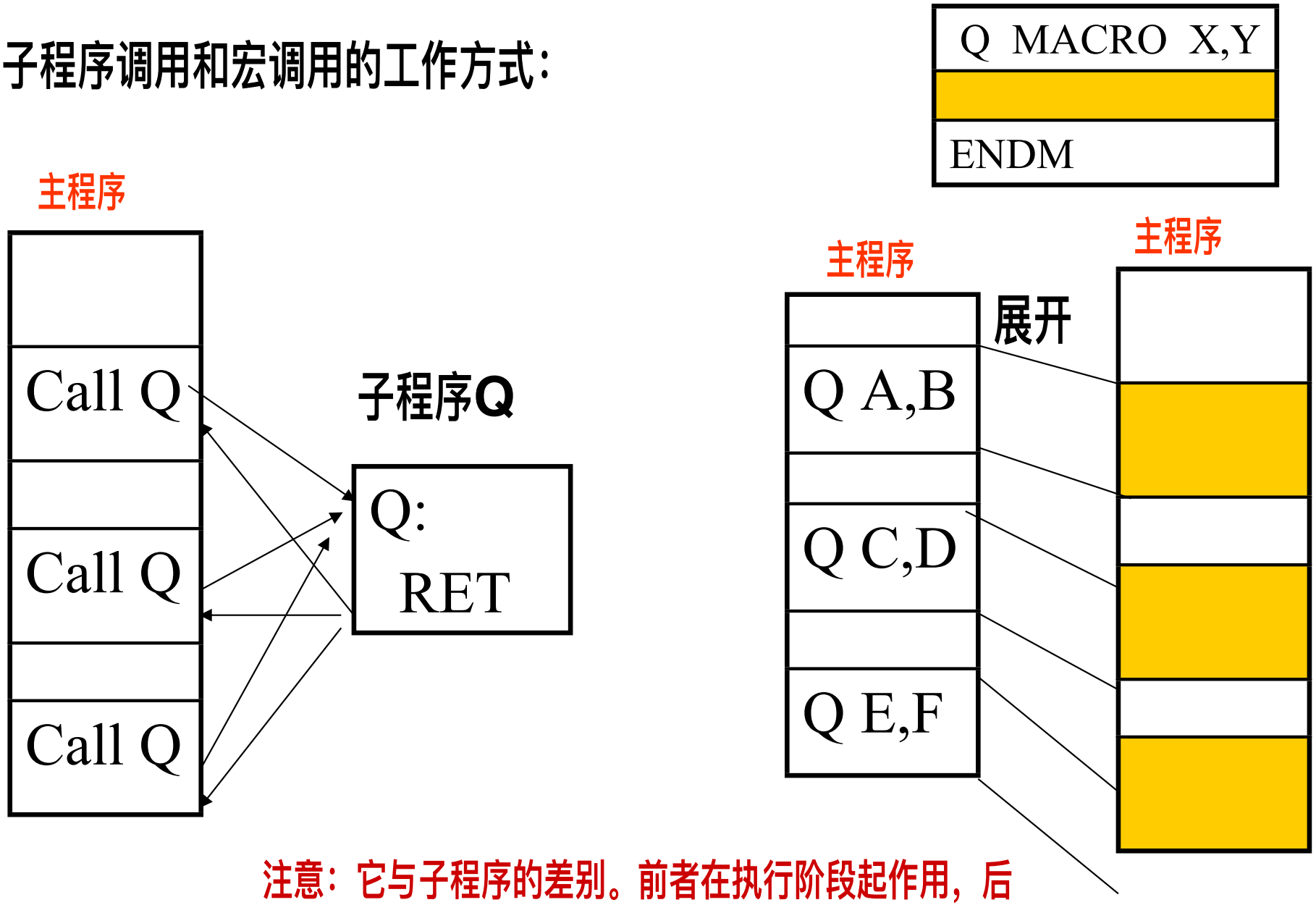
在对源程序进行汇编时，自动用宏定义的内容(宏体)代替宏指令，叫宏展开。

宏展开的具体过程是：
当汇编程序扫描源程序遇到已有定义的宏调用(宏指令)时，即用相应的宏定义体取代源程序的宏指令，同时用位置匹配的实参对形参进行取代。

这样,在程序的目标代码中，每个宏指令语句位置上都包含有相应宏体的目标代码，因此宏指令的使用不会减少程序的目标代码长度。

wenjiecoder@outlook.com

子程序调用和宏调用的工作方式：



注意：它与子程序的差别。前者在执行阶段起作用，后者在汇编阶段做宏展开。

宏汇编

两个字操作数相乘，得到一个双字的乘积结果。

宏定义：

```
MULTIPLY    MACRO     OPR1，OPR2，RESULT
            PUSH      DX
            PUSH      AX
            MOV       AX，OPR1
            IMUL      OPR2
            MOV       RESULT，AX
            MOV       RESULT+2，DX
            POP       AX
            POP       DX
            ENDM
```

宏调用：

```
            …
            MULTIPLY  CX，VAR，XYZ[BX]
            …
            MULTIPLY  240，BX，SAVE
```

宏展开：

```
            …
1           PUSH  DX
1           PUSH  AX
1           MOV   AX, CX
1           IMUL  VAR
1           MOV   XYZ[BX], AX
1           MOV   XYZ[BX]+2, DX
1           POP   AX
1           POP   DX
            …
1           PUSH  DX
1           PUSH  AX
1           MOV   AX,240
1           IMUL  BX
1           MOV   SAVE, AX
1           MOV   SAVE+2, DX
1           POP   AX
1           POP   DX
            …
```

宏定义和宏调用中参数的使用：

宏展开后，即用实参取代形参后，所得到的语句应该是有效的，否则汇编程序会提示出错！

wenjiecoder@outlook.com

宏汇编

宏体可以没有形参：

宏定义：

```
SaveREG     MACRO
            PUSH    AX
            PUSH    BX
            PUSH    CX
            PUSH    DX
            PUSH    SI
            PUSH    DI
            ENDM
```

宏调用：

```
            SaveREG
```

宏定义：

```
BEGIN   MACRO
            MOV     AX, DATAS
            MOV     DS, AX
            MOV     AX, STACKS
            MOV     SS, AX
            ENDM
```

```
EXIT    MACRO
            MOV     AH, 4CH
            INT     21H
            ENDM
```


wenjiecoder@outlook.com

宏汇编

形参可以是操作码：

宏定义：

```
FOO    MACRO        P1, P2, P3
       MOV  AX, P1
       P2       P3
       ENDM
```

宏调用：

```
       FOO   WORD-VAR, INC, AX
```

宏展开：

```
1              MOV  AX, WORD-VAR
1              INC    AX
```

宏汇编

与宏有关的操作符

(1) 连接操作符（&）

- 在宏定义中，可以用连接操作符&作为形参的前缀。在宏展开时，&符前后的两个符号连接在一起构成一个新的符号。这个连接的功能对修改某些符号是很有用的。

(2) 表达式操作符（%）

- 表达式操作符的格式为：%表达式
- 在宏调用时，表达式操作符%强迫后面的表达式立即求值，并把表达式的结果作为实参替换，而不是表达式本身。

(3) 转义操作符（!）

- 转义操作符的格式为：!字符
- 转义操作符指示汇编程序，把后面的字符当成普通的字符对待，而不使用它的特殊含义。宏调用的实参中若包含一些特殊字符（如宏操作符），就可以使用转义操作符。
- 例如，“!&”表示“&”不作为连接操作符使用，只作为符号“&”使用；“!%”表示“%”不作为表达式操作符使用，只作为百分号使用。

wenjiecoder@outlook.com

宏汇编

&操作符

形参可以是操作码的一部分：

宏定义：

```
Leap        MACRO  COND, LAB
            J&COND     LAB
            ENDM
```

这里，&是一个操作符，它在宏体中可以作为形参的前缀，展开时可以把&前后两个符号合并而形成一个符号。这个符号可以是操作码、操作数，或者是一个字符串。

宏调用：

```
        LEAP   Z, THERE
        …
        LEAP   NZ, HERE
        …
```

宏展开：

```
  JZ  THERE
        …
  JNZ         HERE
        …
```

【例7.5】

宏定义：

```
        FO  MACRO  P1
            JMP TO&P1
        ENDM
```

宏调用：

```
        FO   ONE
        ……
        FO   TWO
```

宏展开：

```
        1   JMP TOONE
        1   ……
        1   JMP TOTWO
```

%操作符

格式：%表达式
汇编程序把跟在%之后的表达式的值转换成当前基数下的数。在展开期间，用这个数来取代形参。

%的使用

```
DISP  MACRO  X
      String      DB     'ANSWER:', '&X','$'
       ENDM
```

形参出现在字符串中，前面加“&”，替代后形成新的符号或字符串

宏调用：DISP %(2*11-8)产生的宏扩展为：

```
  1  String   DB    'ANSWER:', '14', '$'
```

不使用符号“%”的宏调用：DISP 2*11-8
产生的宏扩展却是：

```
  1  String   DB    'ANSWER:', '2*11-8', '$'
```


wenjiecoder@outlook.com

！操作符

注意：
在实参中使用“&”、“%”等符号，但不作宏运算符时，就必须在其前使用！”。

如：

```
DISP  MACRO  X
      String     DB     ‘ANSWER:’, ‘&X’,‘$’
ENDM
```

宏调用： DISP   !%(2*11-8)

产生的宏扩展为：

```
1     String   DB    ‘ANSWER:’,  ‘%(2*11-8)’, ‘$’
```

宏汇编

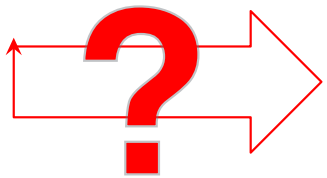
【例7.8】宏定义体中允许使用标号

宏定义:

```
ABSOL  MACRO  OPER
       CMP  OPER,0
       JNS   NEXT
       NEG  OPER
   NEXT:
       ENDM
```

宏调用:

```
       ABSOL var
       …
       ABSOL BX
```



宏展开:

```
1          CMP  var, 0
1          JNS   NEXT
1          NEG  var
1     NEXT:

…

1          CMP  BX, 0
1          JNS   NEXT
1          NEG  BX
1     NEXT:
```


wenjiecoder@outlook.com

宏汇编

宏指令一经定义便可在源程序中调用，若宏体中使用了标号或变量，在多次宏调用时就会出现多个相同标号或出现变量的重复定义，使用LOCAL伪指令可以解决这一问题。

LOCAL伪指令的使用方法及功能如下：

LOCAL伪指令的一般格式：LOCAL 标号及变量表。

各标号、变量之间均用逗号分隔。

LOCAL伪操作只用在宏定义体内，且

LOCAL伪指令必须紧接MACRO伪指令之后，它们中间不能有注释和分号标志。

在处理各个宏调用时，汇编程序将自动以??0000，??0001，……，??FFFF替代LOCAL从伪指令列出的各个标号或变量，从而避免多次宏调用时出现多个相同标号或出现变量重复定义的问题。

wenjiecoder@outlook.com

宏汇编

本例应定义为:

```
ABSOL  MACRO  OPER
    LOCAL NEXT
    CMP  OPER,0
    JNS   NEXT
    NEG  OPER
  NEXT:
ENDM
```

宏调用:

```
ABSOL  VAR
ABSOL  BX
```

宏展开:

```
 1      CMP VAR,0
 1      JNS   ??0000
 1      NEG VAR
1 ??0000:

 1      CMP BX,0
 1      JNS   ??0001
 1      NEG BX
1 ??0001:
```


wenjiecoder@outlook.com

宏汇编

在上例中，宏定义体内只使用了一个标号，如果宏定义体内的标号数多于1个，则可把他们列在LOCAL伪操作之后：

LOCAL　　Next, Out, Exit

在展开时：

第1次宏调用

标号	编号
Next	??0000
Out	??0001
Exit	??0002

第2次宏调用

标号	编号
Next	??0003
Out	??0004
Exit	??0005

wenjiecoder@outlook.com

宏嵌套

宏嵌套象子程序一样包括两种情况：

其一：宏体中包括宏定义。

其二：宏定义的宏体中包括宏调用，即在宏体中调用宏体外定义的宏指令。在这种情况下要注意，其调用的宏指令必须先行定义；

注意：像所有的编程语言一样，不能在源程序中直接调用内层定义的宏指令。换言之，在源程序中只有通过外层宏指令的调用才能调用内层宏指令。

宏汇编

宏嵌套有两种形式：宏定义中嵌套宏定义和宏定义中嵌套宏调用，两种宏嵌套的深度不限。

1. 宏定义中嵌套宏定义

```
MAC1    MACRO
        …
MAC2    MACRO
        …
MAC3    MACRO
        …
        ENDM
        …
        ENDM
        …
        ENDM
```

当宏定义中嵌套宏定义时，必须首先调用最外层宏定义，然后才能调用内层宏定义。

wenjiecoder@outlook.com

宏汇编

2.宏定义中嵌套宏调用

```
MACA   MACRO
       …
       ENDM

MACB   MACRO
       …
       MACA
       …
       ENDM
```

√

各宏定义可单独调用。

```
MACA   MACRO
       …
       MACB   MACRO
       …
       ENDM
       …
       ENDM

MACC   MACRO
       …
       MACB
       …
       ENDM
```

X

先调用MACA，从而形成MACB的定义，然后才能调用MACB。

wenjiecoder@outlook.com

宏汇编

【例7.9】宏定义中允许使用宏，条件是先定义后调用

宏定义：

```
DIF     MICRO X, Y              ; (X-Y)->AX
        MOV   AX, X
        SUB   AX, Y
ENDM

DIFSQR      MACRO   OPR1, OPR2, RESULT
        PUSH  DX
        PUSH  AX
        DIF   OPR1, OPR2
        IMUL  AX
        MOV   RESULT, AX
        POP   AX
        POP   DX
ENDM
```

RESULT=(OPR1-OPR2)²

wenjiecoder@outlook.com

宏汇编

宏调用：

```
DIFSQR    VAR1,VAR2,VAR3
```

宏展开：

```
1         PUSH DX
1         PUSH AX
2         MOV  AX,VAR1
2         SUB  AX,VAR2
1         IMUL AX
1         MOV  VAR3,AX
1         POP  AX
1         POP  DX
```


wenjiecoder@outlook.com

宏汇编

【例7.10】

宏定义：

```
INT21   MACRO   FUNCTN
        MOV     AH, FUNCTN
        INT     21H
ENDM

DISP    MACRO   CHAR
        MOV     DL, CHAR
        INT21   02H
ENDM
```

宏调用：

```
        DISP    '?'
```

宏展开：

```
1               MOV     DL, '?'
2               MOV     AH,02H
2               INT     21H
```


wenjiecoder@outlook.com

宏汇编

宏指令与子程序的区别：程序中重复出现的程序段既可以用宏指令，也可以用子程序（过程）来编写，两者都可以简化源程序的编写，而且都是一次编写，可多次调用。但是它们是两个完全不同的概念，它们之间有一些异同之处，主要有：

处理的时间不同：

宏调用是在源程序被汇编时由汇编程序处理的；而子程序调用是在程序执行期间由CPU直接执行的。

处理的方式不同：

两者都必须先定义后使用，但宏调用是用宏体替换宏调用伪指令，实参代替形参，源程序被翻译成目标代码后宏定义随着消失；而子程序则没有这样的替换操作，是以CALL指令将控制权由调用者转给子程序并执行。

wenjiecoder@outlook.com

宏汇编

参数处理不同：

宏调用是以实参代替形参，参数的形式不受限制，可以是任何合法字符；子程序的参数需要寄存器或存储单元进行传递，而且需要附加的指令实现参数传递。

执行速度不同：

子程序调用时需要执行CALL指令和RET指令，还要执行实现参数传递的附加指令，因而会比宏的执行速度稍慢。

占用的存储器空间大小不同：

宏指令在每次调用时都要展开，把宏体中的程序段复制一遍，因而用宏指令编写的程序在目标代码中会重复出现相同或相似的程序段，占用内存空间较大；而子程序是由CALL指令调用的，无论调用多少次，子程序的目标代码只在程序中出现一次，目标代码相对较短。

wenjiecoder@outlook.com

宏汇编

宏是源程序级的简化，子程序是目标程序级的简化。

宏指令可以调用子程序，子程序也可以包括宏指令。究竟什么情况下使用宏指令或子程序进行设计，可权衡内存空间、执行速度、参数的多少和使用的程序设计方法来确定。

一般而言，采用模块结构设计的程序多用子程序（或过程）。对于一些非标准程序段且程序较短，调用次数又不太频繁，或者是显示打印之类的明确功能，可用宏指令设计，以便增加程序的可读性。

wenjiecoder@outlook.com

宏汇编

重复汇编

库的使用

wenjiecoder@outlook.com

重复汇编

有时候汇编语言程序需要连续的重复完成相同的或几乎完全相同的一组代码，这时候可以使用重复汇编。

格式：REPT 整数表达式
　　重复体
　　ENDM

功能：使汇编程序对重复体作重复汇编，以整数表达式的值作为重复次数。

例如：

```
X=0
REPT  10
   X=X+1
   DB  X
ENDM
```

汇编后：

```
1   DB   1
1   DB   2
1   DB   3
    ……
1   DB   10
```


wenjiecoder@outlook.com

例7.14：把字符‘A’到‘Z’的ASCII码填入数组TABLE

```
      CHAR = ‘A’
TABLE  LABEL  BYTE
      REPT   26
      DB     CHAR
      CHAR = CHAR+1
      ENDM
```

汇编后：

```
1   DB   41H
1   DB   42H
1   DB   43H
    ……
1   DB   5AH
```

重复汇编

不定重复伪操作

IRP伪操作

格式：IRP　形参，<实参表>

　　重复体

　ENDM

功能：使汇编程序对重复体作重复汇编，每作一次汇编就依次将实参表中的一个实参取代重复体中的形参。

例如：

```
IRP    REG, <AX, BX, CX, DX>
  PUSH REG
ENDM
```

其结果等价于：

```
PUSH  AX
PUSH  BX
PUSH  CX
PUSH  DX
```


wenjiecoder@outlook.com

重复汇编

IRPC伪操作

格式：IRPC 形参，字符串
　　重复体
　　ENDM

功能：使汇编程序对重复体作重复汇编，每作一次汇编就依次用字符串中的一个字符取代重复体中的形参。与IRP相似，但实参必须是字符串。

例如：

```
IRPC X, 0123456789
      DB X
ENDM
```

其结果等价于：

```
DB 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
```


wenjiecoder@outlook.com

重复汇编

例7.19：生成存储字符串的汇编语句

```
array  label  byte
IRPC   K, 12345
db     'NO.&K'
ENDM
```

汇编后：

```
1  db  'NO.1'
1  db  'NO.2'
1  db  'NO.3'
1  db  'NO.4'
1  db  'NO.5'
```

例7.20：

```
IRPC   K, ABCD
push   K&X
ENDM
```

汇编后：

```
1  push AX
1  push BX
1  push CX
1  push DX
```

重复汇编

重复汇编与循环程序结构的比较

	重 复 汇 编	循 环 程 序
目标代码所占空间	重复体将重复指定的次数，故并不简化目标代码	重复部分的目标代码只出现一次，故目标代码短
程序运行速度	无需循环控制，程序的运行速度快	需要循环控制，程序的运行速度慢
处理时机	在汇编时由汇编程序对重复体作重复汇编；CPU执行的是经过重复汇编的各个重复体目标代码	在执行时，CPU在循环控制指令的控制下确定是否重复执行循环体
灵活性	重复体可以包括指令、伪指令及宏指令；重复汇编所得到的各个重复体目标代码可以完全相同也可以有所区别，此法较灵活	循环体只能是指令或宏指令，但不能是伪指令；每次重复执行的目标代码完全相同。此法相对来说欠灵活
应用场合	程序运行速度是主要考虑因素，需重复的部分有伪指令，以及各个重复体目标代码有所区别的场合	目标代码所占空间是主要考虑因素，重复的部分不含伪指令以及每次执行的目标代码完全相同的场合

wenjiecoder@outlook.com

宏汇编

重复汇编

库的使用

wenjiecoder@outlook.com

库的使用

宏库的建立与使用：

编程中将经常使用的，带有通用性的宏定义集中放在一个单独的磁盘文件——宏指令库（宏库）中，既可以减少程序的输入量，又方便程序修改。

1. 建立宏库

为了在宏指令库中存放一个或多个宏指令定义（宏定义），可以用EDIT或其它文本编辑器建立宏库。

宏库的扩展名是*.mac。

wenjiecoder@outlook.com

库的使用

macro.mac文件：

```
initz   macro
        mov     ax,@data
        mov     ds,ax
        mov     es,ax
endm
prompt  macro   message
        mov     ah,09h
        lea     dx,message
        int     21h
endm
finish  macro
        mov     ax,4c00h
        int     21h
endm
```


wenjiecoder@outlook.com

库的使用

2. 调用插入伪指令INCLUDE

INCLUDE伪指令用来告诉MASM程序，将语句INCLUDE指出的文件完整地、全部地插入到它所在的位置。该文件是由汇编语言编写的源程序文件，包括宏库文件。语句INCLUDE的格式如下：

INCLUDE [驱动器名：][目录路径]文件名.扩展名

例如：

INCLUDE MYFILE.MAC

在源程序中需要插入程序段的地方使用伪指令语句INCLUDE，就可以指示汇编程序将指定的文件读入，一起进行汇编。当该文件的语句汇编完后，再接着汇编INCLUDE语句后面的语句。

使用INCLUDE要求源程序文件正确无误，否则出错后修改比较麻烦。

wenjiecoder@outlook.com

库的使用

3.删除宏库中部分宏伪指令PURGE

INCLUDE语句将文件的所有宏指令定义全部读到内存，但是有的宏定义对当前程序无用，却占用空间。为了删除汇编时引入到内存的无用宏定义，可在INCLUDE之后直接使用PURGE语句实现，但是库文件中相应的宏指令定义并没有删除。

例如，如果不用宏库中的“INOUTM”宏和“LRCF”，则可以使用下列形式就可以将它们删除。

```
INCLUDE    MACROIO.mac
PURGE      INOUTM , LRCF
```

该删除操作的目的是使该宏定义内容为空，程序汇编时不再展开它。

luguangm@gmail.com

库的使用

macro.mac文件:

```
initz   macro
        mov     ax,@data
        mov     ds,ax
        mov     es,ax
endm
prompt  macro   message
        mov     ah,09h
        lea     dx,message
        int     21h
endm
finish  macro
        mov     ax,4c00h
        int     21h
endm
```

主程序:

```
        include  macro.mac
        .model   small
        .stack   200h
        .data
mess1 db 'Customer name?',13,10,'$'
mess2 db 'Customer address?',13,10,'$'
        .code
begin   proc     far
        initz
        prompt  mess1
        prompt  mess2
        finish
begin   endp
        end
```


luguangm@gmail.com

库的使用

宏指令名可以与指令助记符及伪指令名同名。在此情况下，宏指令的优先级较高，同名的指令或伪指令的原有功能失效。在利用这一方法改变了某个指令助记符或伪指令名的原有功能后，可以通过宏调用来使用新定义的功能。若要恢复其原有功能，可以使用清除宏定义的伪指令。

例如：CBW是一个已定义宏名那么下面：

```
CBW                    ;宏调用
……
PURGE  CBW             ;清除对CBW的宏定义
CBW                    ;将(AL)的符号扩展到AH
```

宏定义时也要注意现场的保护和恢复。

注意宏扩展后程序的一致性、完整性。

第9讲作业：

Page 278-280：7.8，7.15

wenjiecoder@outlook.com

宏汇编

【例】
定义一条INOUT宏指令，既可以引用它输入一串字符，也可引用它显示一串提示字符。

```
INOUT  MACRO  X, Y
       MOV   AH , X
       LEA    DX , Y
       INT     21H
       ENDM
```

```
LF      MACRO
    MOV         DL , 10  ;换行
    MOV         AH , 2
    INT         21H
    MOV         DL , 13   ;回车
    MOV         AH , 2
    INT         21H
    ENDM
```


wenjiecoder@outlook.com

宏汇编

宏调用：

```
DATAS    SEGMENT
INPUT    DB       'PLEASE INPUT ANY CHARACTERS:', '$'
KEYBUF   DB       10, 11 DUP(?), 13, 10, '$'
DATAS    ENDS
CODES    SEGMENT
         ASSUME   CS:CODES, DS:DATAS
START:   MOV      AX, DATAS
         MOV      DS, AX
         INOUT    9, INPUT                ; 显示一串提示符的宏指令调用
         LFCR                             ; 换行、回车
         INOUT    10, KEYBUF              ; 输入一串字符的宏指令调用
         LFCR
         INOUT    9, KEYBUF+2             ; 显示输入的一串字符
         MOV      ah, 4ch
         INT      21h
CODES    ENDS
END      START
```

缓冲区第一个字节保存最大字符数，第二个字节是实际输入字符的个数，两个字节之后的是实际内容。

（DOS中断调用会讲）

wenjiecoder@outlook.com

宏汇编

上述INOUT宏指令调用后，宏展开后的语句如：

```
CODES   SEGMENT
        ASSUME  CS:CODES, DS:DATAS
START:
        MOV     AX, DATAS
        MOV     DS, AX
        INOUT   9, INPUT
LFCR
        INOUT   10, KEYBUF
        LFCR
        INOUT   9, KEYBUF+2
        MOV     ah, 4ch
        INT     21h
CODES   ENDS
END     START
```

```
+       MOV     AH, 9
+       LEA     DX, INPUT
+       INT     21H
+       MOV     DL, 10
+       MOV     AH, 2
+       INT     21H
+       MOV     DL, 13
+       MOV     AH, 2
+       INT     21H
+       MOV     AH, 10
+       LEA     DX, KEYBUF
+       INT     21H
+       MOV     DL, 10
+       MOV     AH, 2
+       INT     21H
+       MOV     DL, 13
+       MOV     AH, 2
+       INT     21H
+       MOV     AH, 9
+       LEA     DX, KEYBUF+2
+       INT     21H
```

例：用嵌套的形式定义将AL中的数据转换为两个十六进制数的ASCII码的宏指令。

```
BHTOA1   MACRO
         MOV     AH, AL
AHHN     MACRO
         LOCAL   AHHN1
         MOV     CL, 4
         SHR     AH, CL
         CMP     AH, 10
         JC      AHHN1
         ADD     AH, 7
AHHN1:   ADD     AH, 30H
         ENDM
ALLN     MACRO
         LOCAL   ALLN1
         AND     AL, 0FH
         CMP     AL, 10
         JC      ALLN1
         ADD     AL, 7
ALLN1:   ADD     AL, 30H
         ENDM
         AHHN
         ALLN
         ENDM
```

有如下宏调用：

```
BHTOA1
```

则宏扩展后得到如下程序段：

```
                BHTOA1
1                       MOV     AH,AL

1                       MOV     CL,4
1                       SHR     AH,CL
1                       CMP     AH,10
1                       JC      ??0000
1                       ADD     AH,7
1               ??0000: ADD     AH,30H

1                       AND     AL,0FH
1                       CMP     AL,10
1                       JC      ??0001
1                       ADD     AL,7
1               ??0001: ADD     AL,30H
```

宏汇编

【例7.11】宏定义体里还可以包含宏定义

宏定义：

```
        DEFMAC      MACRO       MACNAM
    , OPERATOR
                MACNAM      MACRO       X, Y, Z
                        PUSH        AX
                        MOV         AX, X
                        OPERATOR    AX, Y
                        MOV         Z, AX
                        POP         AX
                ENDM
        ENDM
```

宏调用：

```
        DEFMAC      ADDITION, ADD
```

形成加法宏定义：

```
        ADDTION  MACRO      X, Y, Z
                PUSH  AX
                MOV   AX, X
                ADD   AX, Y
                MOV   Z, AX
                POP   AX
        ENDM
```


wenjiecoder@outlook.com

宏汇编

宏调用：

```
        DEFMAC      SUBTRACT, SUB          ;能形成减法宏定义
```

宏调用：

```
        DEFMAC      LOGOR, OR      ;能形成逻辑或宏定义
```

在形成这些宏定义以后，就可以使用宏调用，如加法宏定义宏调用：

```
        ADDITION                   VAR1, VAR2, VAR3
```

宏展开：

```
1               PUSH    AX
1               MOV     AX, VAR1
1               ADD     AX, VAR2
1               MOV     VAR3, AX
1               POP     AX
```


wenjiecoder@outlook.com

宏汇编

```
DEFMAC MACRO   MACNAM,OPERATOR
       MACNAM          MACROX,Y,Z
                ...
       ENDM
ENDM
DATAS SEGMENT
   VAR1 dw 100
   VAR2 dw 50
   VAR3 dw 0
DATAS ENDS
CODES SEGMENT
   ASSUME CS:CODES,DS:DATAS
START:
   MOV AX,DATAS
   MOV DS,AX
   DEFMAC    ADDITION,ADD
   ADDITION  VAR1,VAR2,VAR3
   MOV AH,4CH
   INT 21H
CODES ENDS
   END START
```

```
0000                    CODES SEGMENT
                            ASSUME CS:CODES,DS:DATAS,SS:STACKS
0000                    START:
0000  B8 ---- R              MOV AX,DATAS
0003  8E D8                       MOV DS,AX

                            DEFMAC   ADDITION,ADD
                            ADDITION VAR1,VAR2,VAR3
0005  50              1                 PUSH  AX
0006  A1 0000 R       1                 MOV         AX,VAR1
0009  03 06 0002 R          1                ADD    AX,VAR2
000D  A3 0004 R       1                 MOV         VAR3,AX
0010  58              1                 POP         AX

0011  B4 4C                     MOV AH,4CH
0013  CD 21                     INT 21H
0015                    CODES ENDS
                            END START
```


wenjiecoder@outlook.com