

# 宏结构程序设计

宏汇编

重复汇编

条件汇编

—— 统称宏结构

宏（Macro）是汇编语言的一个特点，它是与子程序类似又独具特色的另一种简化源程序的方法

# 宏汇编

宏——具有宏名的一段汇编语句序列  
——宏定义时书写

宏指令——这段汇编语句序列的缩写  
——宏调用时书写

宏展开——宏指令处用这段宏代替的过程  
——宏汇编时实现

宏的参数功能强大，颇具特色

配合宏，还有宏操作符和有关伪指令

### 定义

定义：是一段具有一定独立功能的汇编代码。该段代码起一个名称宏名。其使用与汇编指令类似。

定义形式：

宏名 **MACRO** 哑元表

宏定义体

**ENDM**

其中哑元表给出了宏定义中所用到的形式参数（或称虚参），每个哑元之间用逗号隔开

汇编指令（宏）代码

按照符号传送的形式参列表

### 定义

定义：是对已定义宏的使用，其使用与汇编指令类似。

调用形式：

宏名 **MACRO** 实元表

传给宏的参数，类似于过程参数

参数取代规则：参数传送时按照符号传送，实元按顺序依次送给哑元；当实元少于哑元时，缺的参数按空对待；实元多余哑元时，忽略多余的参数。

# 宏调用

Computer Architecture  
Group at PKU

**宏的取消：**

宏定义后，其优先顺序高于汇编指令。用PURGE可取消定义。

调用形式：

PURGE 宏名1，宏名2，。。。

宏展开：

- (1) 汇编时，将宏调用的实元取代哑元；
- (2) 将宏名用宏代码代替。

与过程调用的区别：

宏类似类型定义，汇编后消失，不能减小目标码，调用时不会引起控制转移，其参数替换为哑元形式。

## 举例

示例1 无变元

SAVER MACRO

PUSH AX

PUSH BX

PUSH CX

ENDM

程序中使用:

SAVER

## 宏的调用形式:变元是操作码

Computer Architecture  
Group at PKU

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

调用:

```
FOO BX,INC,AX
```

展开:MOV AX,BX  
INC AX

## 变元是操作码的一部分

Computer Architecture  
Group at PKU

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

宏调用:

```
LEAP C,THERE  
LEAP F,HERE
```

宏展开:

```
JC THERE  
JF HERE
```

变元是操作码的一部分，必须用‘&’符号作为分隔符



**&** 是一个操作符，它在宏定义体中可以作为哑元的前缀，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、操作数或是一个字符串。下面两个例子进一步具体说明这个问题。

宏定义：

```
FO      MACRO    P1
        JMP      TA&P1
        ENDM
```

宏调用：

```
FO      WORD_VAR
```

宏展开：

```
+      JMP      TAWORD_VAR
```

在这里，如果宏定义写为

```
FO      MACRO    P1
        JMP      TAP1
        ENDM
```

则在展开时，汇编程序把 TAP1 看作一个独立的标号，并不把 TAP1 中的 P1 作为哑元看待，这样就不能得到预期的结果。

## 变元是 ASCII 串的情况

宏定义：

```
MSGGEN MACRO LAB, NUM, XYZ  
        LAB&NUM DB 'HELLO MR. &XYZ'  
    ENDM
```

宏调用：

```
MSGGEN MSG, 1, TAYLOR
```

宏展开：

```
+ MSG1 DB 'HELLO MR. TAYLOR'
```

**例** 宏指令名可以与指令助记符或伪操作名相同,在这种情况下,宏指令的优先级最高,而同名的指令或伪操作就失效了。伪操作 PURGE 可以用来在适当的时候取消宏定义,以便恢复指令的原始含义。本例说明 PURGE 的用法。

宏定义:

```
ADD      MACRO   OPR1,OPR2,RESULT
          :
          :
          ENDM
```

宏调用:

```
          :
ADD      XX,YY ,ZZ
PURGE    ADD
          :
```

在宏调用后,用 PURGE 伪操作取消宏定义,以便恢复 ADD 指令的原始含义,在 PURGE ADD 后面所用的 ADD 指令,则服从机器指令的定义。

# 宏定义中含有标号

Computer Architecture  
Group at PKU

**ABSOL MACRO OPER**

**LOCAL NEXT**

**CMP OPER,0**

**JGE NEXT**

**NEG OPER**

**NEXT:**

**ENDM**

**宏调用:**

**ABSOL X0**

**宏展开:**

**CMP X0,0**

**JGE ??0000**

**NEG X0**

**??0000:**

...

注意：如果程序中多次调用该宏定义时，展开后台出现标号的多重定义，这是不能允许的。汇编程序对 **LOCAL** 伪操作的局部标号表中的每一个局部标号建立唯一的符号（用？**0000**—？？**FFFF**）以代替在展开中存在的每个局部标号。必需注意，**LOCAL** 伪操作只能用在宏定义体内，而且必须是 **MACRO** 伪操作后的第一语句，在 **MACRO** 和 **LOCAL** 之间不允许有注释和分号标志。

ABSOL	MACRO	OPER
	LOCAL	NEXT
	CMP	OPER,0
	JGE	NEXT
	NEG	OPER
NEXT,		
	ENDM	

宏调用：

⋮
ABSOL     VAR
⋮
ABSOL     BX
⋮

宏展开：

⋮
+        CMP        VAR,0
+        JGE        ?? 0000
+        NEG        VAR
+?? 0000;
⋮
+        CMP        BX,0
+        JGE        ?? 0001
+        NEG        BX
+?? 0001;
⋮

宏定义中可含有嵌套,但宏必须先定义再使用

Computer Architecture  
Group at PKU

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

ENDM

调用:

```
DEFMAC ADDITION,ADD
```

展开:

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

```

DIF      MACRO  X,Y
          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

```

宏调用：

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

宏展开：

```

+      PUSH    DX
+      PUSH    AX
+      DIF     VAR1,VAR2
+      MOV     AX,VAR1
+      SUB     AX,VAR2
+      IMUL    AX
+      MOV     VAR3,AX
+      POP     AX
+      POP     DX

```

## 变元传值:%后必须是汇编变量

Computer Architecture  
Group at PKU

```
MSG MACRO COUNT,STRING
    MSG&COUNT DB STRING
ENDM
ERRMSG MACRO TEXT
    CNTR=CNTR+1
    MSG %CNTR,TEXT
ENDM
```

```
...
CNTR =0
ERRMSG 'SYNTAX ERROR'
```

...

宏展开:

```
...
MSG1 DB 'SYNTAX ERROR'
```

...

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，用这个数来取代哑元。



# 宏操作符总结

- **;;**——宏注释符，用于表示在宏定义中的注释。采用这个符号的注释，在宏展开时不出现
- **&**——替换操作符，用于将参数与其他字符分开。如果参数紧接在其他字符之前或之后，或者参数出现在带引号的字符串中，就必须使用该伪操作符

- **<>**——字符串传递操作符，用于括起字符串。在宏调用中，如果传递的字符串实参数含有逗号、空格等间隔符号，则必须用这对操作符，以保证字符串的完整
- **!**——转义操作符，用于指示其后的一个字符作为一般字符，不含特殊意义
- **%**——表达式操作符，用在宏调用中，表示将后跟的一个表达式的值作为实参，而不是将表达式本身作为参数

；宏定义

```
dstring    macro string
db '&string&',0dh,0ah,'$'
endm
```

；宏调用

```
dstring    < This is a example. >
dstring    < 0 !< Number !< 10 >
```

传递注释符

转义注释符

；宏展开

```
1          db 'This is a example.', 0dh,0ah,'$'
1          db '0 < Number < 10', 0dh,0ah, '$'
```

# 与宏有关的伪指令

- 局部标号伪指令

LOCAL 标号列表

宏定义体采用了标号，应使用 LOCAL 加以说明  
它必须是宏定义 MACRO 语句之后的第一条语句

- 宏定义删除伪指令

PURGE 宏名表

不需要某个宏定义时，可以把它删除

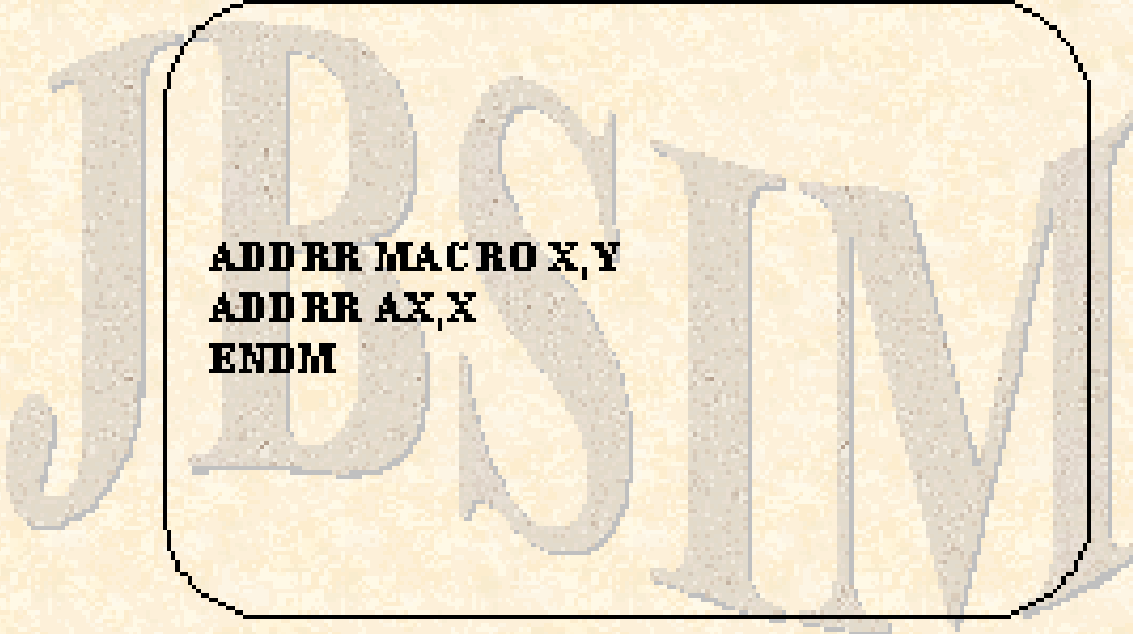
- 宏定义退出伪指令

EXITM

伪指令 EXITM 表示结束当前宏调用的展开

# 宏可以递归定义吗?

Computer Architecture  
Group at PKU



```
ADDRR MACRO X,Y  
ADDRR AX,X  
ENDM
```

## 7.2 重复汇编

Computer Architecture  
Group at PKU

### 完成重复代码汇编的伪操作

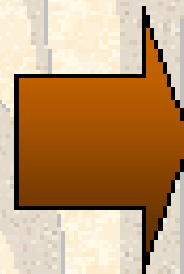
#### 7.2.1 确定次数的重复汇编

```
REPT expression  
...重复块  
ENDM
```

```
X=0  
REPT 3  
X=X+1  
DB X  
ENDM  
汇编后:  
DB 1  
DB 2  
DB 3
```

## 7.2.1 确定次数的重复汇编

```
PUSH TAB MACRO K  
PUSH TAB+K  
ENDM  
宏调用:  
I=0  
REPT 3  
PUSH TAB %I  
I=I+1  
ENDM
```



```
宏展开后:  
PUSH TAB+0  
PUSH TAB+1  
PUSH TAB+2
```

例 要求建立一个 100D 字的数组，其中每个字的内容是下一个字的地址，而最后一个字的内容是第一个字的地址。

**ARRAY LABEL WORD**

**REPT 99**

**DW \$+2**

**ENDM**

**DW ARRAY**



## 7.2.2 不定次数的重复汇编

**IRP 哑元,<变量表>**  
**...重复块**  
**ENDM**

重复次数由自变量表中的自变量个数来确定。自变量表必须用尖括号括起，它可以是常数、符号、字符串等

汇编时将重复块代码重复N次,每次重复用下一个变量代替重复块中的哑元

**IRP X,<1,2,3>**  
**DB X**  
**ENDM**

**DB 1**  
**DB 2**  
**DB 3**

## 7.2.2 不定次数的重复汇编

IRPC 哑元,字符串(<字符串>)  
...重复块  
ENDM

汇编时将重复块代码重复N次,每次重复用下一个字符代替重复块中的哑元

IRPC X,123  
DB X  
ENDM

DB 1  
DB 2  
DB 3

## 7.3 条件汇编

Computer Architecture  
Group at PKU

汇编程序根据条件选择目标是否包含一段程序

IF 条件

代码1

[ELSE]

代码2

ENDIF

条件为真

条件为假

IF *expr*

IFE *expr*

IFDEF *expr*

IFNDEF *expr*

IFB *expr*

IFNB *expr*

IFIND <串1>,<串2>

IFDIF <串1>,<串2>

IF expression	汇编程序求出表达式的值,如比值不为 0 则满足条件。
IFE expression	如求出表达式的值为 0 则满足条件。
IFDEF symbol	如符号已在程序中定义,或者已用 EXTRN 伪操作说明该符号是在外部定义的,则满足条件。
IFNDEF symbol	如符号未定义或未通过 EXTRN 说明为外部符号则满足条件。
IFB <argument>	如自变量为空则满足条件。
IFNB <argument>	如自变量不为空则满足条件。
IFIDN <arg-1>,<arg-2>	如果字符串<arg-1>和字符串<arg-2>相同,则满足条件。
IFDIF <arg-1>,<arg-2>	如果字符串<arg-1>和字符串<arg-2>不相同,则满足条件。

## 7.3 条件汇编

```
BRANCH MACRO X  
IF ($-X) LT 128  
JMP SHORT X  
ELSE  
JMP NEAR PTR X  
ENDIF  
ENDM
```

```
LAB:  
MOV AX,BX  
BRANCH LAB
```

JMP SHORT LAB

例 宏指令 **GOTO L, X, REL, Y**(其中 **REL** 可以是 **Z, NZ, L, NL** 等)可以根据不同情况产生无条件转移指令或比较和条件转移指令。

宏定义

<b>GOTO</b>	<b>MACRO</b>	<b>L,X,REL,Y</b>		
	<b>IFB</b>	<b>&lt;REL&gt;</b>		
	<b>JMP</b>	<b>L</b>		
	<b>ELSE</b>			<b>:</b>
	<b>MOV</b>	<b>AX,X</b>	<b>+</b>	<b>MOV</b> <b>AX,SUM</b>
	<b>CMP</b>	<b>AX,Y</b>	<b>+</b>	<b>CMP</b> <b>AX,15</b>
	<b>J&amp;REL</b>	<b>L</b>	<b>+</b>	<b>JNZ</b> <b>LOOP</b>
	<b>ENDIF</b>		<b>+</b>	<b>:</b>
				<b>JMP</b> <b>EXIT</b>
	<b>ENDM</b>			

宏展开:

宏调用:

	<b>:</b>
<b>GOTO</b>	<b>LOOP,SUM,NZ,15</b>
	<b>:</b>
<b>GOTO</b>	<b>EXIT</b>
	<b>:</b>

例 宏定义可允许递归调用，此时条件伪操作可用来结束宏递归。

宏指令 **POWER** 可以用来实现 **X** 和  $2^N$  相乘。这只需对 **X** 左移 **N** 次即可实现，可以设 **COUNT** 为递归次数的计数值，当该数与 **N** 相等时就可结束递归调用。

宏定义：

```
POWER      MACRO      X,N
            SAL        X,1
            COUNT=COUNT+1
            IF         COUNT=N
            POWER      X,N
            ENDIF
            ENDM
```

# 宏结构的作用

宏汇编、重复汇编和条件汇编

为源程序的编写提供了很多方便，  
灵活运用它们可以编写出非常  
良好的源程序来

汇编系统中有些以圆点起始的  
伪指令（如 `.startup`、`.exit` 等）  
实际上是一种宏结构



# 例题

```
dstring      MACRO string      ;; 定义字符串
              db '&string&',0dh,0ah,'$'
              ENDM

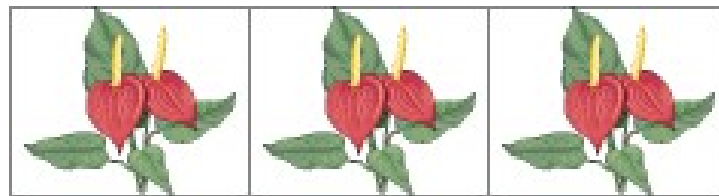
mainbegin    MACRO dsseg        ;; 设置数据段地址
              mov ax,dsseg
              mov ds,ax
              ENDM

dispmsg      MACRO message
              mov dx,offset message
              mov ah,09h
              int 21h
              ENDM
```



# 例题

```
mainend MACRO retnum    ;; 返回 DOS , 可不带参数
    ifb <retnum>
        mov ah,4ch      ;; 没有参数
    else
        mov ax,4c00h+(retnum AND 0ffh)
    ;; 有参数
endif
int 21h
ENDM
```



# 例题

```
.model small
.stack 256
.data
msg1      equ this byte
dstring   <Hello,Everybody !!>
msg2      equ this byte
dstring   <You see,I made it.>
.code
start:    mainbegin @data      ; 建立 DS 内容
          dispmsg msg1        ; 显示 msg1 字符串
          dispmsg msg2        ; 显示 msg2 字符串
          mainend              ; 返回 DOS
end start
```

ENTER

