宏技术

## 宏

**宏**

宏是源程序中一段具有独立功能的代码。在源程序中定义宏后即可多次进行调用。

（文字描述）

**宏定义**

**宏定义格式**

宏定义需要用一组伪操作实现，其格式为：

宏指令名 MACRO [哑元表]

︙ (宏定义体)

ENDM

**说明**

* MACRO和ENDM

用于宏定义的伪操作。

* 宏指令名

用于给出该定义的名称，在进行宏调用时通过宏指令名来调用该宏。

宏指令名的第一个符号必须是字母，其后可以跟字母、数字或下划线字符。

* 宏定义体

一组有独立功能的程序代码。

* 哑元表

给出宏定义中需要用到的形式参数

（文字描述）

**宏调用**

在源程序中对宏指令的调用称为宏调用。

**宏调用格式**

宏指令名 [实元表]

（文字描述）

**宏展开**

当源程序被汇编时，汇编程序将对每个宏调用作宏展开。

**过程**

宏指令会被汇编程序用相应的代码替代，宏定义体将会取代源程序中的宏指令名，并且利用实元取代宏定义中的哑元。

**注意事项**

* 在进行取代时，实元和哑元是一一对应的，第一个实元对应第一个哑元，第二个实元对应第二个哑元，以此类推。
* 汇编程序不要求实元数目必须等于哑元数目，如果实元数目少于哑元数目，多余的哑元将做“空”处理；如果实元数目多于哑元数目，多余的实元将不予考虑。
* 宏展开后得到的语句应该是有效的，否则汇编程序会指示出错。

**（文字描述）**

**（动画展示：需要能具体地形象地表现出宏展开的过程）**

**宏定义、宏调用、宏展开解说例**

用宏指令定义两个字操作数相乘，得到一个16位的第三个操作数作为结果。

宏定义

MULTIPLY MACRO **OPR1**,**OPR2**,**RESULT**

PUSH DX

PUSH AX

MOV AX,**OPR1**

IMUL **OPR2**

MOV **RESULT**,AX

POP AX

POP DX

ENDM

宏调用

︙

MULTIPLY **CX**,**VAR**,**XYZ[BX]**

︙

MULTIPLY **240**,**BX**,**SAVE**

︙

宏展开

︙

+ PUSH DX

+ PUSH AX

+ MOV AX, **CX**

+ IMUL **VAR**

+ MOV **XYZ[BX]**,AX

+ POP AX

+ POP DX

︙

+ PUSH DX

+ PUSH AX

+ MOV AX, **240**

+ IMUL **BX**

+ MOV **SAVE**,AX

+ POP AX

+ POP DX

︙

汇编程序在所展开的指令前加上“+”号以示区别。

（文字说明）

（动画展示：重点展示宏展开过程）

**宏与子程序**

相同点

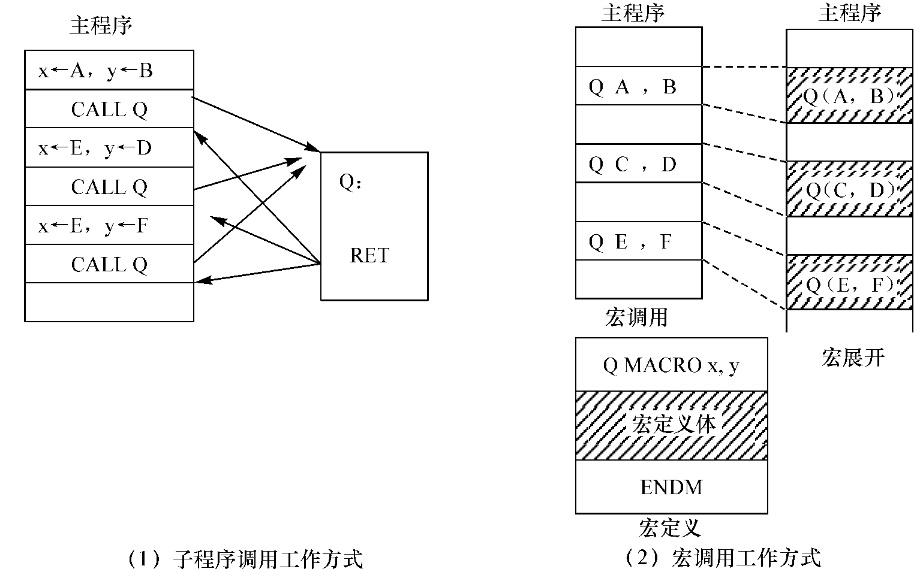
* 都可以定义为一段功能程序。
* 都可以被其他程序进行调用。
* 都利于提高程序的可维护性。

不同点

* 调用工作方式不同。

宏指令能简化源程序的书写。但是汇编程序处理宏指令时，是将宏定义体直接插入到宏调用处。所以宏指令没有并没有简化目标程序。每调用一次宏，目标程序中便会多一处代码的插入。

而子程序的执行，是由CPU直接执行的。在源程序中无论调用多少次，在目标程序中都只有调用指令的目标代码，子程序的目标代码在目标程序中只有一段。



* 储存大小不同

从宏和子程序的调用工作方式可以看出，对于宏指令，每进行一次宏调用，目标代码都会多出一段相应的代码；而子程序无论调用多少次，目标代码中都只有一段子程序目标代码。两者相比，子程序消耗的内存空间会比宏少。

* 执行速度不同

虽然子程序产生的目标代码少，占用内存少，但是通常调用子程序时要先保护断点，还要保护现场；在返回时要恢复现场再恢复断点。这一过程会额外增加执行的时间，执行速度相应变慢。而宏虽然占用内存增加，但是不需要额外的操作，因而执行速度会快些。

* 参数处理不同

宏指令中的参数可以是常数、寄存器、内存变量、操作码或操作码的一部分等，这是子程序做不到的。同时宏指令的参数传递十分方便，而子程序的参数传递是依靠寄存器、堆栈等方式实现的，实现起来不方便。

（文字说明）

（视频讲解：需要一部分动画展示子程序调用的工作方式和宏调用工作方式的区别）

## 宏的参数

**变元**

哑元和实元统称为变元

（文字说明）

**常见使用分类**

**无变元**

宏定义：

SAVEREG MACRO

PUSH AX

PUSH BX

PUSH CX

PUSH DX

PUSH SI

PUSH DI

ENDM

宏调用：

SAVREG

宏展开时将宏定义体中的所有内容全部列出。

**变元为操作数或储存单元**

宏定义：

MULTIPLY MACRO **OPR1**,**OPR2**,**RESULT**

PUSH DX

PUSH AX

MOV AX,**OPR1**

IMUL **OPR2**

MOV **RESULT**,AX

POP AX

POP DX

ENDM

宏调用：

︙

MULTIPLY **CX**,**VAR**,**XYZ[BX]**

︙

MULTIPLY **240**,**BX**,**SAVE**

︙

宏展开：

︙

+ PUSH DX

+ PUSH AX

+ MOV AX, **CX**

+ IMUL **VAR**

+ MOV **XYZ[BX]**,AX

+ POP AX

+ POP DX

︙

+ PUSH DX

+ PUSH AX

+ MOV AX, **240**

+ IMUL **BX**

+ MOV **SAVE**,AX

+ POP AX

+ POP DX

︙

**变元为操作码**

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：

+ MOV **AX**,WORD\_VAR

+ **INC** **AX**

其中把替换哑元P2的实元INC为操作码。

**变元为操作码的一部分**

宏定义：

LEAP MACRO **COND**,**LAB**

J&**COND**  **LAB**

ENDM

宏调用：

︙

LEAP **Z**,**THERE**

︙

LEAP **NZ**,**HERE**

︙

宏展开：

︙

+ J**Z**  **THERE**

︙

+ J**NZ**  **HERE**

︙

**变元为字符串**

宏定义：

MSGGEN MACRO **LAB**,**NUM**,**XYZ**

**LAB**&**NUM** DB ‘HELLO MR.&**XYZ**’

ENDM

宏调用：

MSGGEN **MSG**,**1**,**WHITE**

宏展开：

+ **MSG1** DB ‘HELLO MR.**WHITE**’

（文字说明）

（PPT分页演示各种用例）

（动画演示：主要演示宏展开的过程）

## 宏的运算符

**&**

在宏定义体中可以作为哑元的前缀，在宏展开时，汇编程序会将&前后两个符号合并形成一个符号。

使用例：

宏定义：

DISTR MACRO **SS**

DB ‘Exam:&**SS**’,0AH,0DH,’$’

ENDM

宏调用：

DISTR **book**

宏展开：

+ DB ‘Exam:**book**’,0AH,0DH,’$’

**<>**

当变元为字符串时，如果实元是含有空格的字符串，则实元需要使用<>括起。

使用例：

宏定义：

DISTR MACRO **SS**

DB ‘Exam:&**SS**’,0AH,0DH,’$’

ENDM

宏调用：

DISTR **<My name is Peter.>**

宏展开：

+ DB ‘Exam:**My name is Peter.**’,0AH,0DH,’$’

**!**

当字符串中含有<或>字符时，为避免产生冲突，在宏调用的实元中需要用到!将字符转义。

使用例：

宏定义：

DISTR MACRO **SS**

DB ‘Exam:&**SS**’,0AH,0DH,’$’

ENDM

宏调用：

DISTR **2!<3**

宏展开：

+ DB ‘Exam: **2<3**’,0AH,0DH,’$’

**%**

在宏调用的实元中如果有表达式，%会将表达式的值转换为当前基数下的值。

使用例：

宏定义：

DISTR MACRO **SS**

DB ‘Exam:&**SS**’,0AH,0DH,’$’

ENDM

宏调用：

DISTR **%35+42**

宏展开：

+ DB ‘Exam:**77**’,0AH,0DH,’$’

**;;**

;;在宏定义中使用的注释符，在;;后的注释在宏调用和宏展开时不会进行展开。

（文字说明）

（动画演示：主要演示宏展开的过程）

## 其他宏功能

**宏标号**

对于宏定义：

ABSOL MACRO OPER

CMP OPER,0

JGE NEXT

NEG OPER

NEXT:

ENDM

若在程序中多次调用该宏，进行宏展开后会出现标号的重复定义，这在程序中是不允许的。如果要在宏定义体内使用标号，需要用到LOCAL伪操作。

**格式**

LOCAL伪操作的格式为：

LOCAL 局部标号表

**说明**

* 局部标号表内的各标号之间要用逗号隔开。
* 汇编程序对LOCAL伪操作的局部标号表中的每个局部标号建立一个唯一的符号来代替展开中存在的每一个局部编号。
* LOCAL伪操作只能用于宏定义体内，并且必须是MACRO伪操作后的第一个语句，中间不允许有任何注释和分号标志。

对于上述宏定义，应修改为：

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 NEXT

+ NEG VAR

+??0000:

︙

+ CMP BX,0

+ JGE NEXT

+ NEG BX

+??0001:

︙

（文字说明）

（动画演示：主要展示汇编程序建立局部标号表）

**宏删除**

在宏汇编中，如果不需要某个宏时，可以选择使用伪操作PURGE将其删除。另外，由于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，同名的指令或伪操作就失效了，此时也要用到PURGE伪操作。

**格式**

PURGE 宏指令名

**说明**

如果要同时删除多个宏定义，各宏指令名之间需要用逗号进行隔开。

（文字说明）

**宏嵌套**

宏嵌套，即是在宏定义中包含宏，主要包含两种情况。在宏定义中进行宏调用和在宏定义中进行宏定义。

注意事项

* 宏定义体中的宏调用必须先被定义才能被调用。
* 在宏外部不能直接调用在宏内部定义的宏，需要先调用外部宏才能调用内部宏。

用例说明

* 在宏定义中进行宏调用

宏定义：

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

+ IMUL AX

+ MOV VAR3,AX

+ POP AX

+ POP DX

在LST列表清单中可以查看到上述的宏展开信息，但需要注意的是，DIF VAR1,VAR2语句是为用户方便而提供的，并不会占用储存单元，在最终的目标文件中这个语句是不存在的。

* 在宏定义中进行宏定义

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

这其中MACNAM是内层的宏定义名，同时又是外层定义的哑元，在调用DEFMAC时就会形成一个宏定义。

宏调用：

DEFMAC ADDITION,ADD

宏展开：

+ ADDITION MACRO X,Y,Z

PUSH AX

MOV AX,X

ADD AX,Y

MOV Z,AX

POP AX

ENDM

这样就形成了加法宏定义ADDITION。

（文字说明）

（动画演示：主要演示宏展开的过程）

## 宏库

定义

如果在程序中定义了多个宏，可以将这些宏放在独立的文件中进行保存。这种文件与高级语言中的库文件类似，称为宏库。

建立

把多个宏的宏定义放在一个文本文件中，为其起名并加上扩展名.MAC。

调用

在应用程序中使用宏指令之前，用INCLUDE伪指令将宏库调入。

（文字说明）

## 重复汇编

定义

汇编语言程序连续地重复完成相同的或者几乎完全相同的一组代码。

重复伪操作

格式

REPT 表达式

︙ （重复块）

ENDM

说明

* 表达式的值用于确定重复块的重复次数，表达式中如果包含外部或未定义的项则汇编程序会指示出错。
* 重复伪操作不一定要用在宏定义体内。

使用例

X=0

REPT 10

X=X+1

DB X

ENDM

汇编后将会产生

+ DB 1

+ DB 2

+ DB 3

︙

+ DB 10

不定重复伪操作

IRP伪操作

* 格式

IRP 哑元,<自变量表>

︙ （重复块）

ENDM

* 说明

1. 重复次数由自变量表中的自变量个数确定。
2. 自变量表必须使用尖括号括起，可以为常数、符号、字符串等。
3. 汇编程序会将重复块的代码进行重复，每进行一次重复，就会把重复块中的哑元用自变量表中的一项进行取代，下一次取代下一项。
4. IRP伪操作不一定要用在宏定义中。

* 使用例

IRP X,<1,2,3,4,5,6,7,8,9,10>

DB X

ENDM

汇编后可得：

+ DB 1

+ DB 2

+ DB 3

︙

+ DB 10

IPRC伪操作

* 格式

IRPC 哑元,字符串(或<字符串>)

︙ （重复块）

ENDM

* 说明

1. IRPC与IRP类似，但自变量必须为字符串。
2. 重复次数由字符串中的字符数确定。
3. 每次重复用字符串的下一个字符取代重复块中的哑元。
4. IRPC伪操作不一定要用在宏定义中。

* 使用例

IRPC K,ABCD

PUSH K&X

ENDM

汇编后可得：

+ PUSH AX

+ PUSH BX

+ PUSH CX

+ PUSH DX

（文字说明）

（动画展示主要展示汇编程序实现重复代码的过程）

## 条件汇编

作用

决定是否对某段程序进行汇编，可以控制程序具有不同功能以及最后生成的代码。

格式

IF条件伪操作

︙ }条件满足时汇编此块

(ELSE)

︙ }条件不满足时汇编此块

ENDIF

说明

* IF条件必须在汇编程序第一遍扫视后就成为确定的数值。
* ELSE语句以及之后的语句块可省略。
* 条件伪操作可以用在宏定义体内，也可以用在宏定义体外，也允许嵌套使用多次。

IF条件伪操作类型

|  |  |
| --- | --- |
| 类型 | 说明 |
| IF 表达式 | 汇编程序求出表达式的值，如此值不为0则满足条件。 |
| IFE 表达式 | 如果求出的值为0则满足条件 |
| IFDEF 符号 | 如果该符号已在程序中给出定义，或者使用EXTRN伪操作说明该符号是在外部进行定义的，则满足条件。 |
| IFNDEF 符号 | 如果该符号未定义或未使用EXTRN伪操作进行说明为外部符号则满足条件。 |
| IFB <自变量> | 如果自变量为空则满足条件。 |
| IFNB <自变量> | 如果自变量不为空则满足条件。 |
| IFIDN <字符串1>,<字符串2> | 如果字符串1和字符串2相同，则满足条件 |
| IFDIF <字符串1>,<字符串2> | 如果字符串1和字符串2不相同，则满足条件 |

（文字说明）