石家莊鐵道大學
SHIJIAZHUANG TIEDAO UNIVERSITY

在线开放课程

汇编语言程序设计

伪指令-2

主讲 ： 燕延

目录

一、过程定义伪操作

1、过程定义与调用

过程定义伪操作命令为PROC / ENDP，格式：

```
过程名  PROC  [NEAR / FAR]
        ……
        RET
        ……
过程名  ENDP
```

过程的程序块中应该有返回指令RET，但不一定是最后一条指令，也可以有不止一条RET指令。执行RET指令后，控制程序返回到原来调用指令的下一条指令。

调用一个过程的格式为：CALL 过程名

过程名实质上是过程入口的符号地址，它和标号一样，也有三种属性：段、偏移量和类型。

过程的类型属性可以是NEAR或FAR。

类型为NEAR的过程可以在段内被调用；

类型为FAR的过程还可以被其他段调用。

过程的定义和调用均可嵌套。例如：

```
NAME1  PROC   FAR
        ……
        CALL  NAME2
        ……
        RET
NAME2  PROC   NEAR
        …..
        RET
NAME2 ENDP
NAME1  ENDP
```

CALL和RET指令都有NEAR和FAR的属性，段内调用使用NEAR属性，但可以不显示地写出；段间调用使用FAR属性。

为了使用户的工作更加方便，80x86的汇编程序用PROC伪操作的类型属性来确定CALL和RET指令的属性。

这样，用户只需在定义过程时考虑它的属性，而CALL和RET的属性可以由汇编程序来确定。

【例1】　调用程序和过程在同一代码段中。

```
MAIN    PROC    FAR  ;  主程序
        ……
CALL    SUBR1
        ……
        RET
MAIN    ENDP
SUBR1   PROC    NEAR ;  过程(NEAR可省略)
        ……
        RET
SUBR1   ENDP
```

【例2】　调用程序和过程不在同一个代码段内。

```
SEGX   SEGMENT
       ……..
SUBT    PROC   FAR
       ……..
       RET
SUBT    ENDP
       ……..
       CALL  SUBT
       ……..
SEGX   ENDS
SEGY   SEGMENT
       ……..
       CALL  SUBT
       ……..
SEGY   ENDS
```

SUBT过程本段内和其它段都有调用，所以类型属性必须设为FAR才可满足要求。

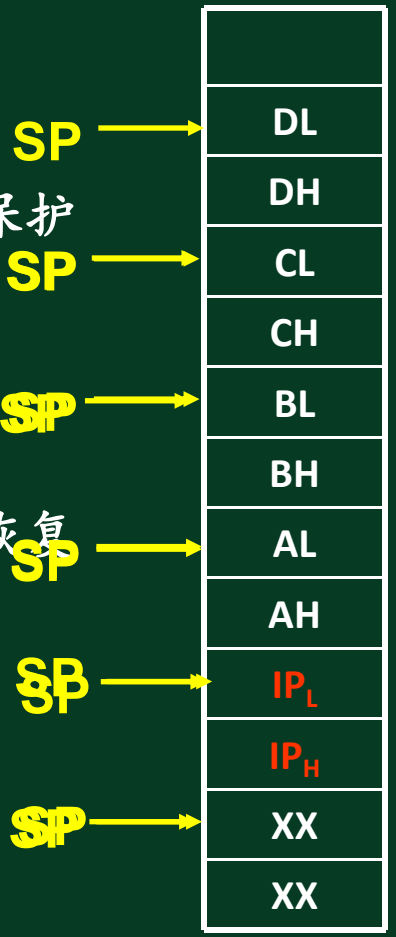
2. 现场保护与现场恢复

由于主程序和过程通常是分别编制的，所以它们所使用的寄存器往往会发生冲突。如果主程序在调用过程之前的某个寄存器内容在从过程返回后还有用，而过程又恰好使用了同一个寄存器，这就破坏了该寄存器的原有内容，因而会造成程序运行错误，这是不允许的。

为避免这种错误的发生，在进入过程后，就应该把过程所需要使用的寄存器内容保存在堆栈中，此过程称作**现场保护**；而在退出过程前把寄存器内容恢复原状，此过程称作**现场恢复**。现场保护与现场恢复分别使用压栈和弹出指令实现。

【例3】现场保护与恢复

```
SUBT   PROC
       PUSH   AX      ; 现场保护
       PUSH   BX
       PUSH   CX
       PUSH   DX
       <过程体>:
       POP    DX      ; 现场恢复
       POP    CX
       POP    BX
       POP    AX
       RET
SUBT   ENDP
```



【例4】设从BUF开始存放若干无符号字节数据，找出其中的最小值并送到BL寄存器保存。

分析：本题用过程SEARCH来寻找最小数并送BL寄存器。

```
DATA    SEGMENT
        BUF     DB  25,13 ,23,100,123,78,134,90
        CNT     EQU    $-BUF  ；数据个数
DATA    ENDS
```

```
CODE  SEGMENT
      ASSUME      CS：CODE，DS：DATA
START：MOV  AX，DATA
       MOV  DS，AX
       LEA  SI，BUF ；首址送SI
       CALL  SEARCH ；找最小数送BL子程序
       MOV  AH，4CH；返回DOS
       INT  21H
```

```
SEARCH   PROC  NEAR
          MOV  BL，[SI]；假定第一个数为最小数
          MOV  CX，CNT-1    ；比较次数
 SEAR1:INC  SI；指向下一个数
          CMP  BL，[SI]；比较
          JBE  SEAR2  ；BL中的数小，转SEAR2
          MOV  BL，[SI]  ；BL中的数大，把它替换掉
 SEAR2:  LOOP   SEAR1；循环比较
          RET
SERCH ENDP
 CODE   ENDS
       END  START
```

二、 宏处理伪操作

如果在源程序中需要多次使用同一个程序段，可以将这个程序段定义为一个宏指令，每次需要时，即可简单地用宏指令名来代替(称为宏调用)，从而避免了重复书写，使源程序更加简洁、易读。

Microsoft宏汇编程序MASM提供了丰富的宏处理伪操作命令，下面讨论几种常用的宏处理伪操作。

1.MACRO/ENDM

宏定义格式如下：

宏指令名　MACRO

.....(宏定义体)

ENDM

进行宏定义后，就可以多次用宏指令名进行宏调用了。但是必须先定义，后调用。

汇编时，对每个宏指令名，MASM自动用相应宏定义体中的程序段代替，这个过程称为宏扩展。

使用宏的过程共有三步：首先进行宏定义；然后可以进行宏调用；最后汇编时由MASM进行宏展开。

宏定义允许嵌套，即宏定义体中可以包含另一个宏定义，而且宏定义体中也可以有宏调用，但是也必须先定义后调用。

宏定义伪操作允许带参数，此时所定义的宏指令具有较强的通用性。带参数的宏定义格式如下所示：

宏指令名 MACRO 参数 [，参数，…，参数]
……；宏指令体(宏体)
ENDM

以上宏定义中的参数称为形式参数或称哑元。当形式参 数不止一个时，各参数之间要用逗号分开。

以后宏调用时，应在宏指令名后面写上相应的实际参数或称实元。一般情况下，实际参数与形式参数的个数和顺序均为一一对应。但是，汇编程序允许两者的个数不等。

当实际参数多于形式参数时，多余的实际参数被忽略；当形式参数多于实际参数时，认为多余的形式参数为空。

[例5] 试编写一个宏定义，可将任意两个8位寄存器或存储单元中的压缩BCD数相加，并将结果存回第一个寄存器或存储单元。可编程如下：

```
DECADD   MACRO  OPRI，OPR2
         MOV  AL，OPRI
         ADD  AL，OPR2
         DAA
         MOV  OPRI，AL
              ENDM
```

假设有以下宏调用：

```
DECADD  DL，BUFFER
    ……
```

则汇编时进行宏扩展，得到以下指令的机器码

```
    DECADD  DL，BUFFER
+   MOV  AL，DL
+   ADD  AL，BUFFER
+   DAA
+   MOV  DL, AL
    …….
```

```
DECADD   MACRO OPRI，OPR2
         MOV  AL，OPRI
         ADD  AL，OPR2
         DAA
         MOV  OPRI，AL
         ENDM
```

2、LOCAL

伪操作LOCAL的作用是向宏汇编程序MASM指出宏定义体中的局部标号。利用这个伪操作将允许在宏定义体内使用标号。如果没有LOCAL伪操作，则当多次调用一个使用标号的宏定义时，通过宏扩展，宏定义体中的标号将在程序中多处出现，从而产生标号多重定义的错误。

LOCAL伪操作只能出现在宏定义体内，而且必须位于宏定义中其它所有语句（包括注释）之前。

格式如下：

LOCAL 局部标号[，.....]

如果有多个局部标号，互相之间应该用逗号隔开。

汇编以后，MASM在每一次宏扩展中自动将一个新的标号代替原来的局部标号。新标号的形式为：？？0000至？？FFFF。所以在源程序中其他地方应该避免使用这种形式的符号。

【例6】源程序中多次需要将不同寄存器中的十六进制数转换为相应的ASCII码时，可以定义以下宏指令：

```
HEXTOASC   MACRO  REG
           LOCAL  NUME
           CMP  REG, 0AH
           JC  NUME
           ADD  REG, 07
NUME:      ADD  REG, 30H
           ENDM
```

有下面宏调用：

```
HEXTOASC   AL
……
HEXTOASC  BL
```

宏展开后：

```
+          CMP  AL, 0AH
+          JC   ??0000
+          ADD  AL, 07
+??0000: ADD AL, 30H
```

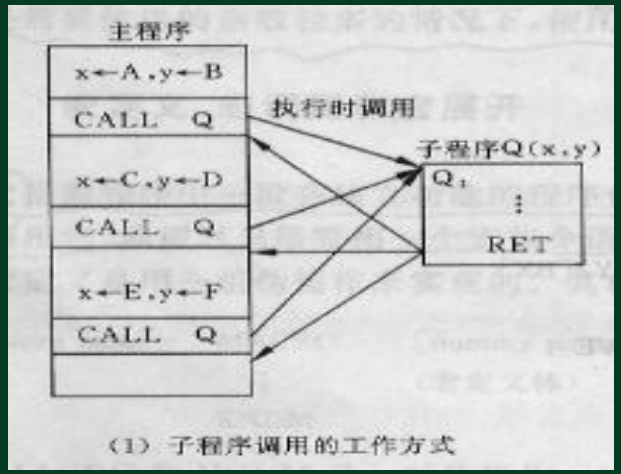
HEXTOASC AL

```
+          CMP  BL, 0AH
+          JC   ??0001
+          ADD  BL, 07
+??0001: ADD BL, 30H
```

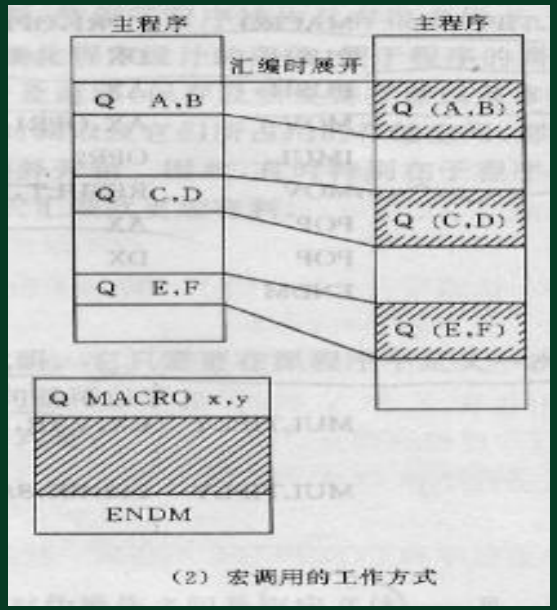
HEXTOASC BL

宏与子程序的区别

1、宏调用伪操作由宏汇编程序MASM在汇编过程中进行处理，在每个宏调用处，将相应的宏定义体插入；而调用指令CALL和返回指令RET则是CPU指令，执行CALL指令时，CPU控制程序转移到子程序的入口地址。



(1) 子程序调用的工作方式



(2) 宏调用的工作方式

2、宏指令简化了源程序，但不能简化目标程序，汇编后，在宏定义处不产生机器代码，但在每个宏调用处，通过宏展开，重复程序段的机器代码仍然出现多次，因此并不节省内存单元；对于子程序来说，在目标程序中，定义子程序的地方将产生相应的机器代码，但每次调用时只需用CALL指令，不再重复出现子程序的机器代码，一般来说可以节省内存容量。

3、从执行时间看，调用子程序和从子程序返回需要保护断点、恢复断点等，将额外占用CPU的时间；而宏指令则不需要，因此相对来说执行速度较快。此外，宏指令更加接近高级语言，而且传递参数更加方便。

本讲小结

1、过程的定义、过程调用及返回

2、宏定义、宏调用、宏展开