

## 第6章 子程序结构

主讲：许万茹





北京交通大学

BEIJING JIAOTONG UNIVERSITY



# 本章教学内容



过程定义伪操作



子程序的调用与返回



保存与恢复寄存器



子程序的参数传送



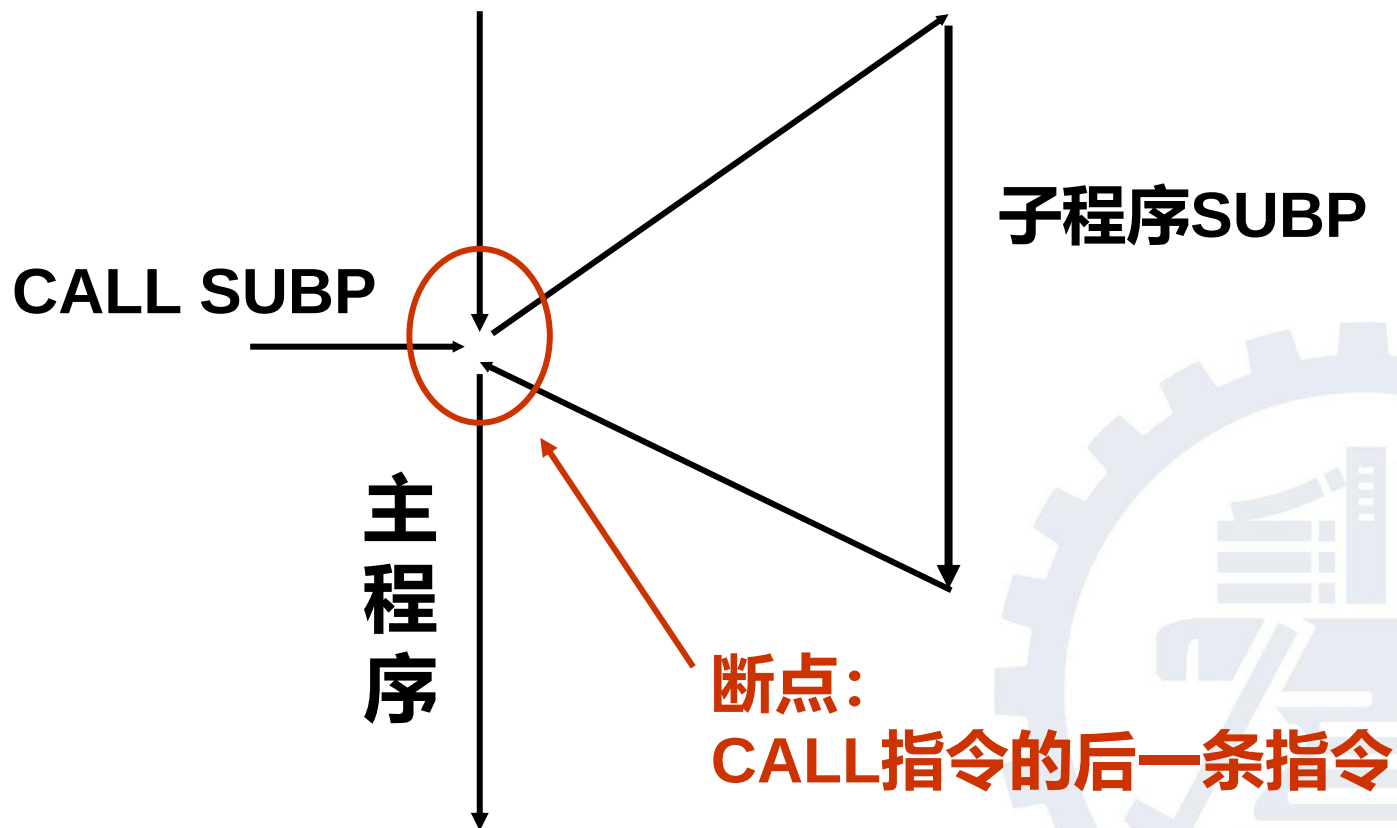
子程序的嵌套与递归

**重点：子程序的结构和输入输出参数的传递方法**



## 子程序的运行方式

特点：主动调用，而不是随机的。





## CALL 调用指令

**段内直接近调用: CALL DST (DST为子程序名)**

**执行操作:**  $(SP) \leftarrow (SP) - 2$  ;断点压入堆栈

$((SP)+1, (SP)) \leftarrow (IP)$  ; 主程序地址压栈

$(IP) \leftarrow (IP) + 16$  位位移量

**注意:** IP为Call指令的下一条指令的地址, 其与DST子程序的地址间的相对地址或位移量是固定的。

**段内间接近调用: CALL DST**

**(DST为寄存器如BX或存储器地址 WORD PTR [BX])**

**执行操作:**  $(SP) \leftarrow (SP) - 2$

$((SP)+1, (SP)) \leftarrow (IP)$

$(IP) \leftarrow (EA)$  (**DST为内存地址**)



## RET 返回指令

### 段内近返回: RET

**功能：**将堆栈中保存的2字节断点的偏移地址恢复IP中，CS不变

**执行操作：**  $(IP) \leftarrow ((SP)+1, (SP))$   
 $(SP) \leftarrow (SP) + 2$

### 段内带立即数近返回: RET EXP

**EXP**表示弹出断点之后，使SP内容再回退EXP个字节单元，作用是使断点之后的EXP个字节单元数据失效



# 1、过程定义伪操作

```
过程名  PROC  NEAR ( FAR )  
          :  
          :  
过程名  ENDP
```

- (1) **NEAR** 属性：调用程序和子程序在同一代码段中  
(段内调用)
- (2) **FAR** 属性：调用程序和子程序不在同一代码段中  
(段间调用)



## CODE SEGMENT

```
MAIN: MOV AX, DATA  
      MOV DS, AX
```

.....

```
      CALL SUBR1
```

.....

```
      MOV AH, 04H  
      INT 21H
```

```
      SUBR1 PROC NEAR
```

.....

```
      RET
```

```
      SUBR1 ENDP
```

```
CODE ENDS
```

```
END MAIN
```

## SEGX SEGMENT

.....

```
      CALL SUBT
```

.....

```
      PROC FAR
```

.....

```
      RET
```

```
      SUBT
```

```
      ENDP
```

```
SEGX
```

```
      ENDS
```

```
SEGY
```

```
SEGMENT
```

.....

```
      CALL FAR PTR SUBT
```

.....

```
SEGY
```

```
      ENDS
```



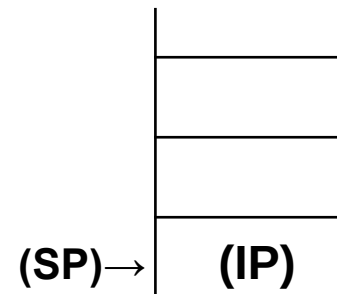


## 2、子程序的调用与返回

子程序调用：隐含使用堆栈保存返回地址

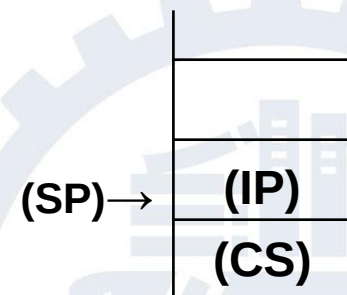
CALL **NEAR** PTR SUBP

- (1) 保存返回地址：仅offset地址
- (2) 转子程序

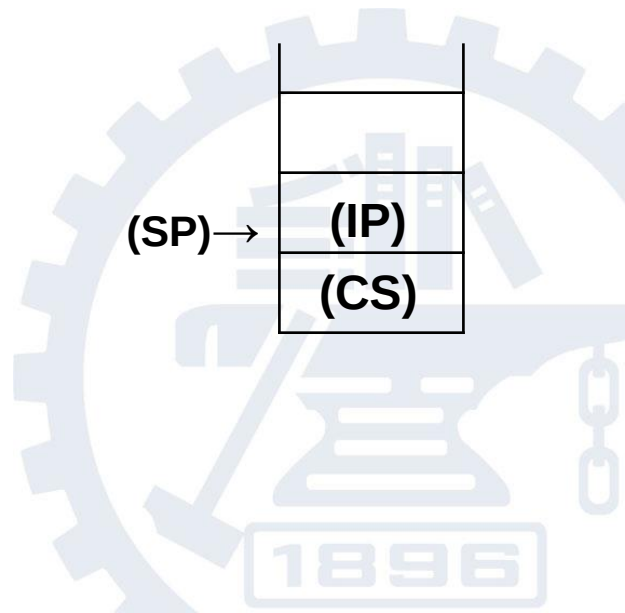


CALL **FAR** PTR SUBP

- (1) 保存返回地址：seg+offset
- (2) 转子程序



子程序返回：RET







### 3、保存与恢复寄存器

**SUBT PROC NEAR (FAR)**

**PUSH AX**

**PUSH BX**

**PUSH CX**

**PUSH DX**

**.....**

**.....**

**POP DX**

**POP CX**

**POP BX**

**POP AX**

**RET**

**SUBT ENDP**

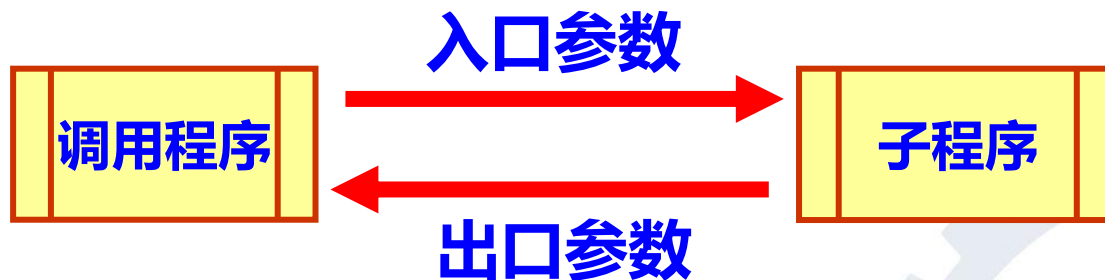




## 4、子程序的参数传递

- 参数传递：

调用程序和子程序之间的信息交互。



- 参数的形式：

- ① 数据本身（传值）
- ② 数据的地址（传址）



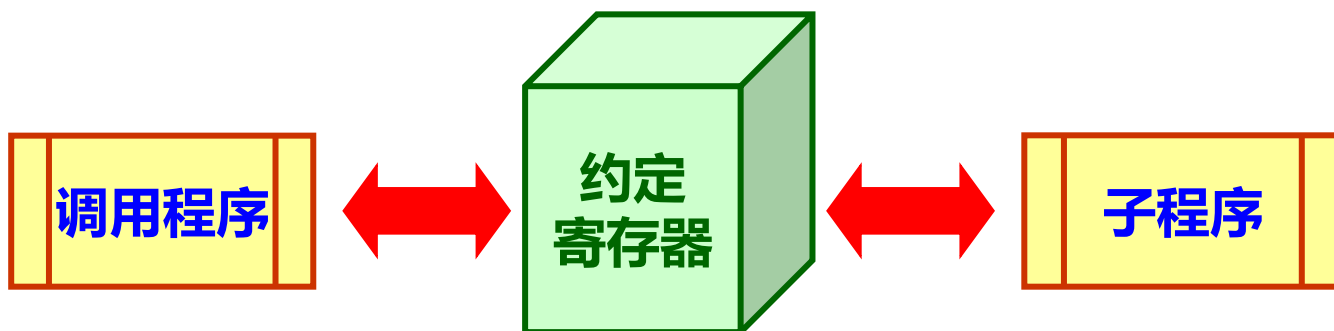
- **参数传递方法:**

- (1) 通过**寄存器**传送参数
- (2) 通过**存储单元**传送参数
- (3) 通过**堆栈**传送参数或参数地址
- (4) 多模块之间的参数传递





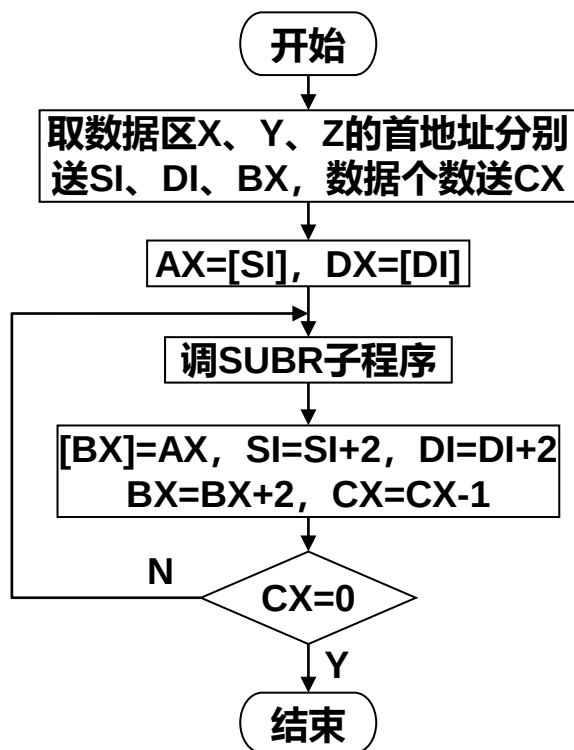
## [通过寄存器传送参数]



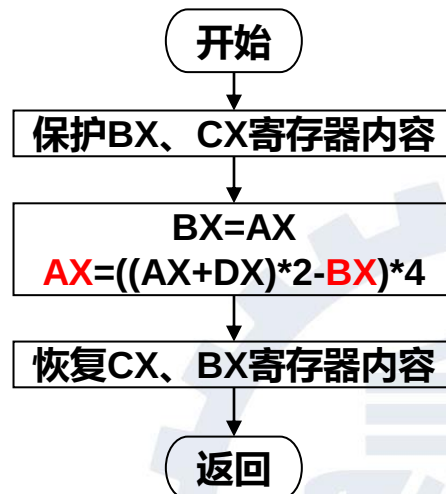
- 把参数存于约定的寄存器中，可以传值，也可以传址。
- 子程序对带有出口参数的寄存器不能保护和恢复（主程序视具体情况进行保护）
- 子程序对带有入口参数的寄存器可以保护，也可以不保护；但最好一致



例1：将给定的一组**字数据**X、Y代入 $Z=((X+Y) \times 2 - X) \times 4$  公式中，计算相应的Z值。假设Z的值不会超过16位。



(a)主程序框图



(b)子程序框图



北京交通大学

BEIJING JIAOTONG UNIVERSITY



DATA SEGMENT

X DW 5,3,8,9,2,5,3,4,7,1

Y DW 1,5,7,0,4,3,1,4,8,1

Z DW 10 DUP (?)

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

START:MOV AX,DATA

MOV DS,AX

LEA SI,X

LEA DI,Y

LEA BX,Z

MOV CX,Y-X ;字节数

SHR CX,1 ;字数/数组个数

REAPT:MOV **AX**,[SI]

MOV **DX**,[DI]

CALL SUBR

MOV [BX],AX

ADD SI,2

ADD DI,2

通过寄存器传递参数

ADD BX,2

LOOP REAPT

EXIT:MOV AH,4CH

INT 21H

SUBR PROC NEAR

PUSH BX

PUSH CX

MOV BX,AX

ADD AX,DX ; AX=X+Y

SAL AX,1 ; AX=(X+Y)×2

SUB AX,BX ; AX=(X+Y)×2-X

MOV CL,2

SAL **AX**,CL ; AX=((X+Y)×2-X)×4

POP CX

POP BX

RET

SUBR ENDP

CODE ENDS

END START

1896



## • 寄存器参数传递特点:

- 简单方便，只需约定寄存器即可。
- 寄存器的个数和容量非常有限，适用于传递较少的参数信息。

```
LEA SI,X  
LEA DI,Y  
LEA BX,Z  
MOV CX,Y-X ;字节数  
SHR CX,1 ;字数/数组个数  
REAPT:MOV AX,[SI]  
MOV DX,[DI]  
CALL SUBR  
MOV [BX],AX  
ADD SI,2  
ADD DI,2
```

通过寄  
存器传  
递参数

```
SAL AX,1 ; AX=(X+Y)×2  
SUB AX,BX ; AX=(X+Y)×2-X  
MOV CL,2  
SAL AX,CL ; AX=((X+Y)×2-X)×4  
POP CX  
POP BX  
RET  
SUBR ENDP  
CODE ENDS  
END START
```





## [通过约定存储单元传送参数]



### • 特点:

- 约定存储单元交互信息，即主程序和子程序直接采用同一个变量名共享同一个变量，实现参数的传递
- 可以在调用程序与子程序间传递大量数据



## 例2：累加数组中的元素

```
DATA SEGMENT
ARY DW 1,2,3,4,5,6,7,8,9,10
COUNT DW 10
SUM DW ?
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
MAIN: MOV AX,DATA
      MOV DS,AX

      CALL PROADD

      MOV AX,4C00H
      INT 21H

CODE ENDS
END MAIN
```

```
PROADD PROC NEAR
    PUSH AX
    PUSH CX
    PUSH SI
    LEA SI,ARY
    MOV CX,COUNT
    XOR AX,AX
NEXT: ADD AX,[SI]
      ADD SI,2
      LOOP NEXT
    MOV SUM,AX
    POP SI
    POP CX
    POP AX
    RET
PROADD ENDP
```

入口  
参数

出口  
参数



### 例3：如果数据段定义如下：

DATA SEGMENT

ARY DW 1,2,3,4,5,6,7,8,9,10

COUNT DW 10

SUM DW ?

ARY1 DW 10,20,30,40,50,60,70,80,90,100

COUNT1 DW 10

SUM1 DW ?

DATA ENDS





北京交通大学

BEIJING JIAOTONG UNIVERSITY



```
CODE SEGMENT
```

```
    ASSUME CS:CODE,DS:DATA
```

```
MAIN:MOV AX,DATA
```

```
    MOV DS,AX
```

```
    LEA SI,ARY
```

```
    MOV CX,COUNT
```

```
    CALL PROADD
```

```
    MOV SUM,AX
```

```
    LEA SI,ARY1
```

```
    MOV CX,COUNT1
```

```
    CALL PROADD
```

```
    MOV SUM1,AX
```

```
    MOV AX,4C00H
```

```
    INT 21H
```

```
CODE ENDS
```

```
    END MAIN
```

多次调用同  
一个子程序

```
PROADD PROC NEAR
```

```
    XOR AX,AX
```

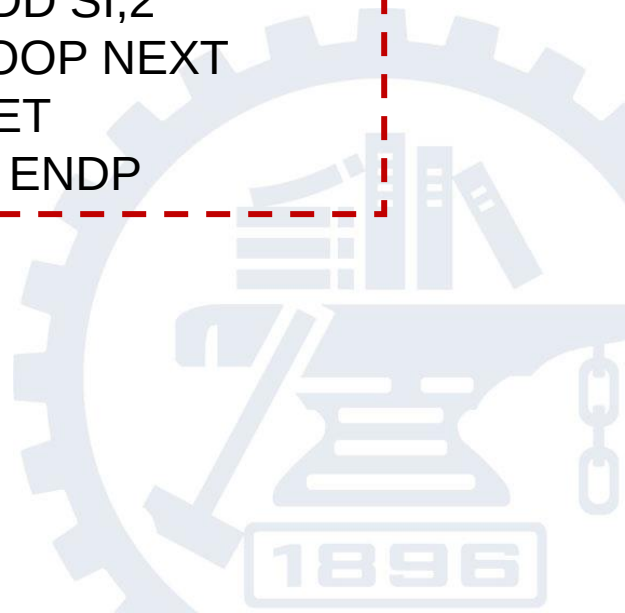
```
    NEXT: ADD AX,[SI]
```

```
    ADD SI,2
```

```
    LOOP NEXT
```

```
    RET
```

```
PROADD ENDP
```





## [通过堆栈传送参数]

- 主程序将子程序的入口参数压入堆栈，子程序从堆栈中取出参数；子程序将出口参数压入堆栈，主程序弹出堆栈取得它们
- 适用于参数较多，子程序有多层嵌套、递归调用的情况
- 步骤：主程序把参数或参数地址压入堆栈；子程序使用堆栈中的参数或通过栈中参数地址取到参数（用BP访问堆栈段）；子程序返回时使用RET n指令调整SP指针，以便删除堆栈中已用过的参数，保持堆栈平衡，保证程序的正确返回。



## 伪指令补充1:

### 1, LABEL伪指令

**含义:** 为本伪指令之后的标号/变量定义一个不同类型的别名。

**用法:** 变量/标号 LABEL 类型(byte, word, dword, near, far)

**例子:**

**VAR LABEL WORD**

**X DB 'AB'**

**; 变量VAR和其后的X指向内存中的同一单元, 但两  
; 者类型分别为WORD类型、BYTE类型;**

**MOV AX, VAR**

**; 等价于 MOV AX, 4241H**

**MOV AL, X**

**; 等价于 MOV AL, 41H**





## 例4：累加数组中的元素

DATA SEGMENT

ARY DW 10,20,30,40,50,60,70,80,90,100

COUNT DW 10

SUM DW ?

DATA ENDS

STACK SEGMENT

DW 100 DUP (?) ; 首个字节地址为SS首地址;

TOS LABEL WORD ; TOS为栈底, 其具体值相当于\$

STACK ENDS ; 因为使用到堆栈段来传递参数, 所以要先定义STACK





**CODE1** SEGMENT

MAIN PROC FAR

ASSUME CS:CODE1,DS:DATA,SS:STACK

START:MOV AX,STACK

MOV SS,AX

MOV SP,OFFSET TOS ; 初始时栈顶与栈底地址相同, 为SS:SP

MOV AX,DATA

MOV DS,AX

MOV BX,OFFSET ARY ; 通过堆栈传递参数时要先将参数地址压入栈中

PUSH BX

MOV BX,OFFSET COUNT

PUSH BX

MOV BX,OFFSET SUM

PUSH BX

CALL **FAR** PTR PROADD

MOV AX,4C00H

INT 21H

MAIN ENDP

**CODE1** ENDS





北京交通大学

BEIJING JIAOTONG UNIVERSITY



```
CODE2 SEGMENT
ASSUME CS:CODE2
```

```
PROADD PROC FAR
```

```
PUSH BP
MOV BP, SP
```

```
PUSH AX
PUSH CX
PUSH SI
PUSH DI
```

```
MOV SI,[BP+0AH]
MOV DI,[BP+8]
MOV CX,[DI]
MOV DI,[BP+6]
```

```
CODE2 ENDS
END START
```

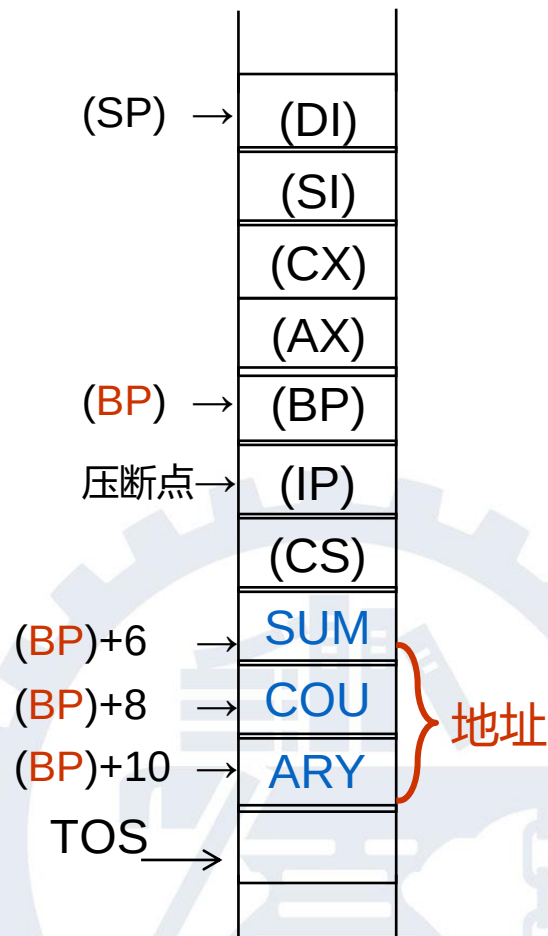
```
XOR AX, AX
NEXT:
ADD AX, [SI]
ADD SI, 2
LOOP NEXT
MOV [DI],AX
```

```
POP DI
POP SI
POP CX
POP AX
```

```
POP BP
```

```
RET 6; (SP)<-(SP)+6,
即废除压入栈中的ARY/
COUNT/SUM,清空栈
```

```
PROADD ENDP
```



1896



## [多模块之间的参数传递]

- 在某一个模块中定义，而在另外一个模块中引用的符号称为**外部符号**。PUBLIC和EXTRN与外部符号有关

- **PUBLIC 伪指令**

格式：PUBLIC 符号[,...]

说明：在一个模块中定义的符号（变量、标号、过程名）在提供给其他模块使用时，必须要用PUBLIC定义该符号为外部符号。



## [多模块之间的参数传递]

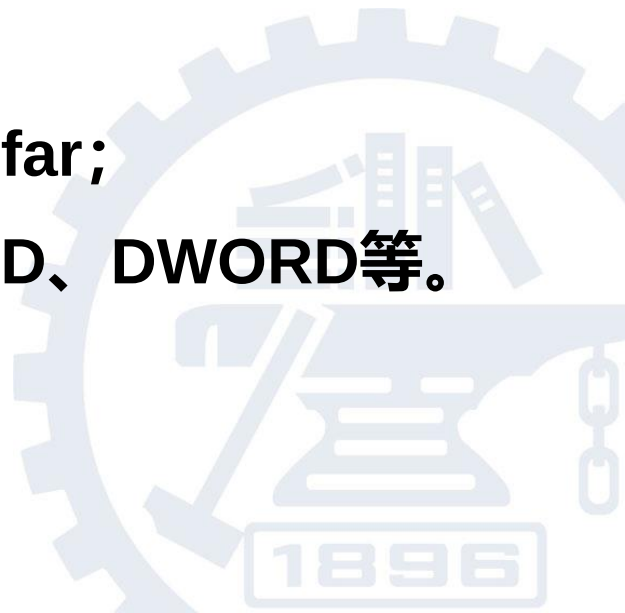
- **EXTRN 伪指令**

格式: **EXTRN 符号:类型[,...]**

说明: 在另一个模块中定义而要在本模块中使用的符号必须要用EXTRN说明。

若符号为标号或过程名, 则类型near或far;

若符号为变量, 则类型为BYTE、WORD、DWORD等。





## 例5：主程序 (LIE6D6.asm)

```
PUBLIC D1,D2,N1                ; 全局变量定义
EXTRN LIE6D6A: FAR             ; 外部过程说明
EXTRN LIE6D6B: FAR             ; 外部过程说明
DAT SEGMENT PARA 'DAT'        ; 数据段
    D1 DB 98H,35H,54H,78H
    N1=$-D1
    D2 DB 12H,34H,56H
DAT ENDS
STAC SEGMENT PARA 'STACK' ; 堆栈段
    STAL DW 100 DUP(?)
STAC ENDS
```



```
CODE SEGMENT PARA 'CODE' ; 代码段
    ASSUME CS: CODE, DS: DAT, SS: STAC
STA  PROC FAR

    ...
    LEA SI,D1                ; 全局变量的引用
    LEA DI,D2
    MOV CX,N1

    ...
    CALL LIE6D6A             ; 外部过程的引用

    ...
    CALL LIE6D6B             ; 外部过程的引用

    ...
    RET
STA  ENDP
CODE ENDS
    END STA
```



## 子程序1 LIE6D6A.asm

```
PUBLIC LIE6D6A          ; 全局过程  
CODE SEGMENT PARA 'CODE'  
    ASSUME CS: CODE  
LIE6D6A PROC FAR  
    PUSH AX  
  
    ...  
    POP AX  
    RET  
LIE6D6A ENDP  
CODE ENDS  
END
```







## 子程序2 LIE6D6B.asm

PUBLIC LIE6D6B ; 全局过程

EXTRN D1: BYTE, D2: BYTE

CODE SEGMENT PARA 'CODE'

ASSUME CS: CODE

LIE6D6B PROC FAR

...

LEA SI,D1

; 全局变量

LEA DI,D2

...

RET

LIE6D6B ENDP

CODE ENDS

END





## 如何产生可执行文件

### 1、三个模块分别汇编产生各自的目标文件

C:\masm>masm LIE6D6.asm

产生 LIE6D6.obj



C:\masm>masm LIE6D6A.asm

产生 LIE6D6A.obj



C:\masm>masm LIE6D6B.asm

产生 LIE6D6B.obj



### 2、连接

C:\masm>link LIE6D6 LIE6D6A LIE6D6B

产生 LIE6D6.exe 文件





**1、试编写一个汇编程序，能对键盘输入的小写字母用大写字母显示出来。**

**（要求采用子程序格式，即采用子程序完成将小写字母转化成大写字母）**



**作业**

**2、有2个数组：**

**ary1 db 12,-35,0,126,-90,-5,68,120,1,-19**

**ary2 db 24,25,0,-38,-89,99,68,100,2,-20**

**比较两个数组的对应位，将大的数放在ary1数组中，小的数放在ary2中（要求采用子程序格式）**

