哈爾濱工業大學(深圳)

HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

汇编语言程序设计
第7&8讲：子程序结构

裴文杰

第7&8讲：子程序结构

子程序的引出与定义

子程序调用与和返回

子程序设计方法

子程序嵌套

递归子程序

程序的连接

wenjiecoder@outlook.com

第7&8讲：子程序结构

子程序的引出与定义

子程序调用与和返回

子程序设计方法

子程序嵌套

递归子程序

程序的连接

wenjiecoder@outlook.com

子程序的引出与定义

子程序又称为过程，是能完成特定功能有一定通用性的程序段，在需要时能被其它程序调用。

适用于设计子程序的两种情况：

1） 一个程序中反复出现的程序段，功能和结构相同，只是变量赋值不同；

2）不同程序用到一些常用的功能相同而独立的程序段，例如十进制与二进制的转换等。

调用子程序的程序称为主程序。

子程序设计的优势：

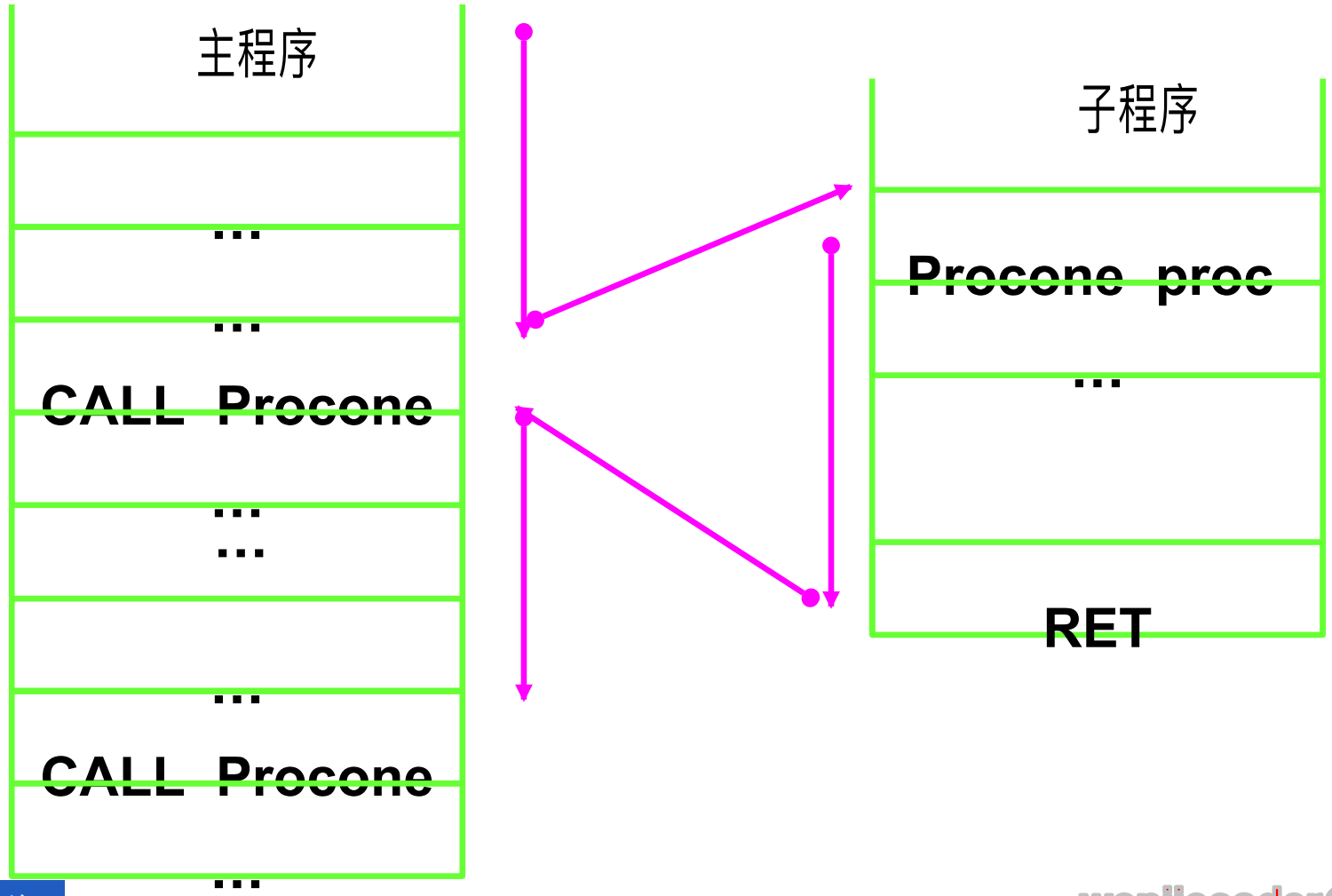
1）简化源程序结构、节省目标程序的存储空间，提高程序设计的效率；

2）子程序结构是模块化程序设计的基础：将程序按照功能划分为不同的模块，提高程序的可读性，可维护性和共享性。

wenjiecoder@outlook.com

子程序的引出与定义

子程序可以实现源程序的模块化，可简化源程序结构，可以提高编程效率



子程序的引出与定义

子程序的定义是由过程定义伪指令**PROC**和**ENDP**实现，格式如下：

```
过程名 PROC        [NEAR|FAR]    ；子程序开始定义
          …                      ；过程体
          RET                    ；返回主程序
过程名 ENDP                      ；子程序定义结束
```

过程名（子程序名）为符合语法的标识符

NEAR属性（段内近调用）的过程只能被相同代码段的其他程序调用。

FAR属性（段间远调用）的过程可以被相同或不同代码段的程序调用。

(1) 子程序和调用程序**在同一个代码段**中，则子程序定义为**NEAR**属性。

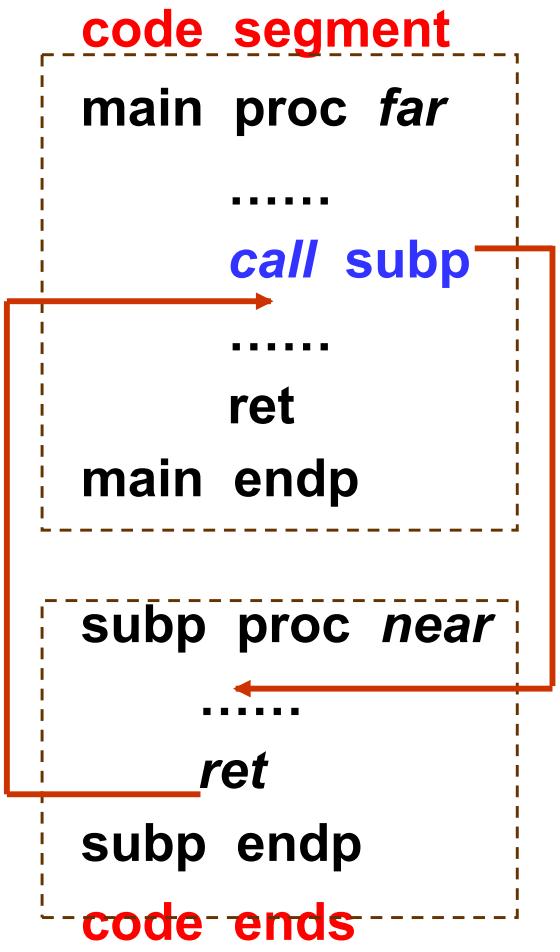
(2) 子程序和调用程序**不在同一个代码段**中，则子程序定义为**FAR**属性。

(3) **主程序**通常定义为**FAR**

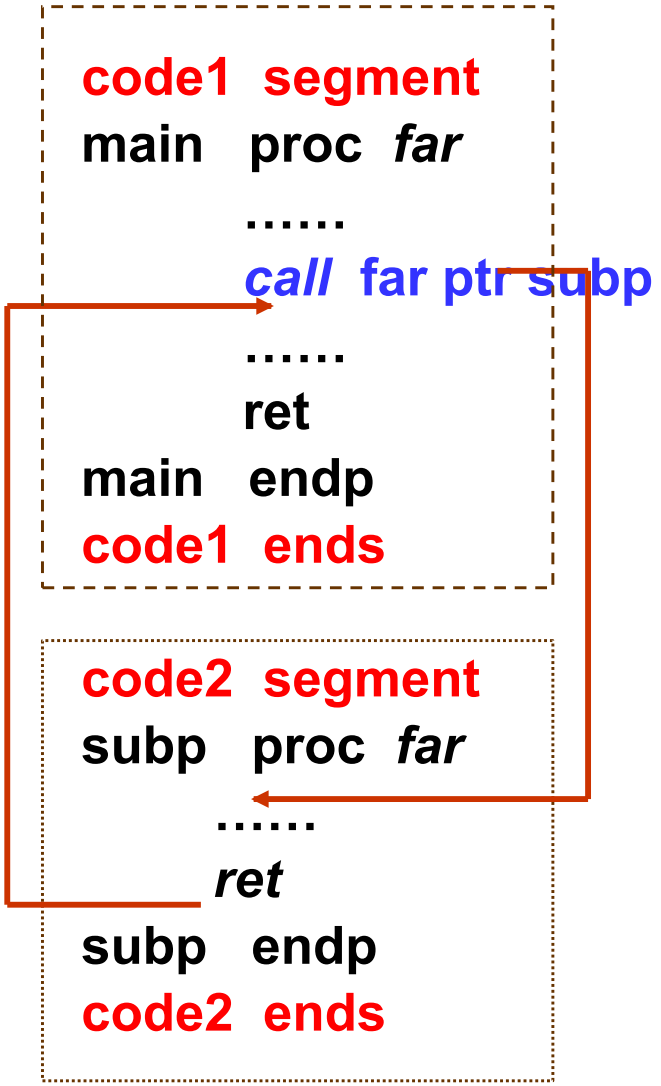
属性，这是因为主程序被看作DOS调用的一个子程序，以便执行完返回DOS。

控制转移指令

子程序调用和返回指令：



段内调用和返回



段间调用和返回

子程序的引出与定义

例：编写一个子程序，从键盘输入一位十进制数。

；子程序名：stdin
；功能：从键盘输入1位十进制数，若不是十进制数，则重新输入。
；入口参数：等待键盘输入
；出口参数：al中存放输入的数值

```
stdin   proc                ;缺省表示Near属性
again:    mov    ah, 1
          int    21h
          cmp    al, 30h
          jl     again
          cmp    al, 39h
          jg     again
          and    al, 0fh    ;ASCII码->0~9,  or (sub al, 30h)
          ret
stdin   endp
```

DOS系统调用INT 21H-单字符输入：

功能号1送AH

AL=输入字符

第7&8讲：子程序结构

子程序的引出与定义

子程序调用与和返回

子程序设计方法

子程序嵌套

递归子程序

程序的连接

wenjiecoder@outlook.com

子程序调用与返回

子程序调用与返回由**CALL**和**RET**指令实现。

- 子程序调用指令首先把子程序的返回地址（即**CALL**指令的下一条指令的地址）压入堆栈，然后转移到子程序的入口地址执行子程序。
  - 子程序和主程序在同一个代码段中称为段内调用；
  - 子程序和主程序不在同一个代码段中，称为段间调用。
- 子程序返回指令

  负责把压入栈区的返回地址弹出送**IP**或**CS**：**IP**，实现返回主程序继续往下执行。子程序的返回也分为段内返回和段间返回。

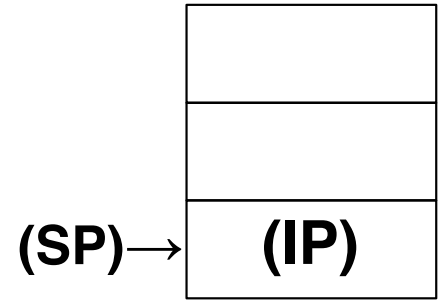
wenjiecoder@outlook.com

子程序调用与返回

子程序调用：隐含使用堆栈保存返回地址

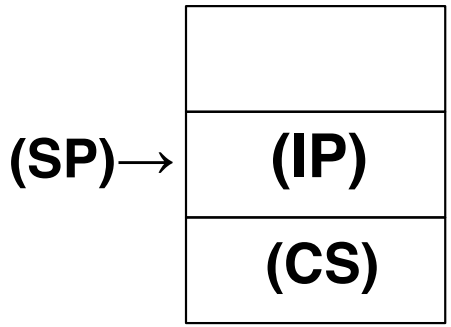
call near ptr subp

(1) 返回地址（IP）入栈
(2) 转子程序
(IP) ← subp的偏移地址



call far ptr subp

(1) 返回地址（CS、IP）入栈
(2) 转子程序
(CS) ← subp的段地址
(IP) ← subp的偏移地址



子程序调用与返回

返回指令

汇编格式： RET [VAL]

VAL为一个正偶数

VAL只是修改了返回后SP的值。

作用：调用子程序时先把所需参数入栈，调用结束后这些参数不再有用，可以修改堆栈指针使其指向参数入栈以前的值。

(1) 段内返回（近返回）

IP← (SP+1, SP)

SP ←SP+2

SP ←SP+VAL（如果选用了VAL）

(2) 段间返回（远返回）

IP ← (SP+1, SP)

SP ←SP+2

CS ← (SP+1, SP)

SP ←SP+2

SP ←SP+VAL（如果选用了VAL）

如果子程序不能正确使用堆栈而造成执行RET前SP并未指向进入子程序时的返回地址，则导致运行出错。

进栈：高地址->底地址，出栈：底地址->高地址

子程序调用与返回

例：带立即数返回

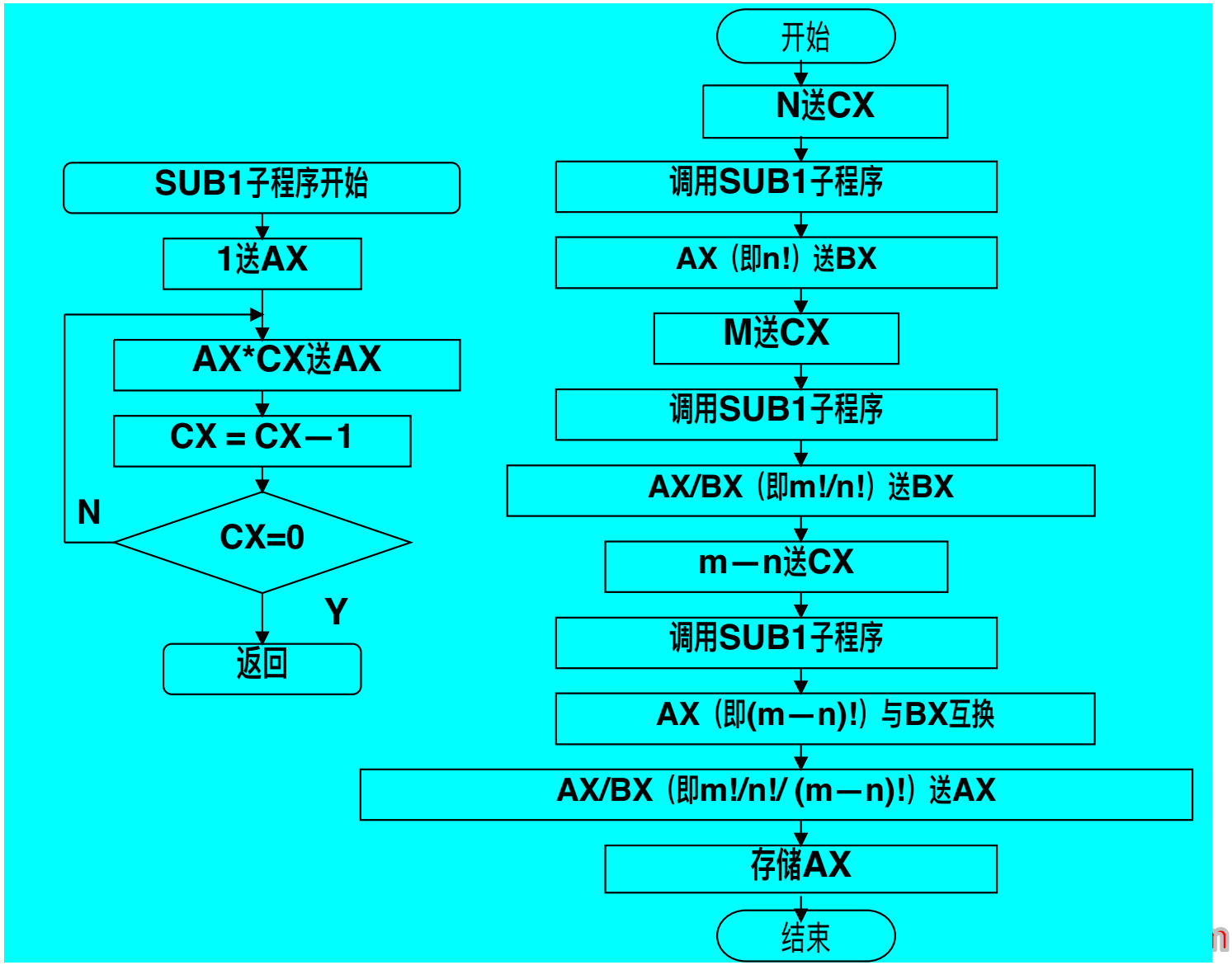
```
code  segment
main  proc  far
      ……
      push  参数1
      push  参数2
      push  参数3
      call  sub
      ……
      ret
main  endp

sub   proc  near
      ……
      ret  6
sub   endp
code  ends
```

堆栈段

把子程序所需的参数入栈，从子程序返回后，如果参数
不再有用，那么无需再pop出栈。因此通过ret 6来跳过它们。（这里假设每个参数占两个字节）

子程序调用与返回

例：以下程序用于计算=m!/(n!*(m-n)!)的值（m、n为自然数，且m>n）。（假设阶乘乘积大小不超过两个字节）



子程序调用与返回

实例：计算=m!/(n!*(m-n)!)，其中m=8，n=3.

```
DATA  SEGMENT
  M    EQU 8
  N    EQU 3
 RES   DW ?
DATA  ENDS

CODE  SEGMENT
      ASSUME CS:CODE,DS:DATA
START:
      MOV AX, DATA
      MOV DS, AX
      MOV CX, N
      CALL SUB1  ；调子程序计算n!
      MOV BX, AX
      MOV CX, M
      CALL SUB1  ；调子程序计算m!
      MOV DX, 0   ；CWD(?)
      DIV BX       ；m!/n!送 AX
      MOV BX, AX
      MOV CX,M
      SUB CX, N
      CALL SUB1   ；调用子程序，计算(m-n)!
      XCHG BX, AX
      MOV DX, 0
      DIV BX        ；m!/n!/ (m-n)! 送AX
      MOV RES, AX
      MOV AH, 4CH
      INT 21H

SUB1   PROC         ；计算阶乘的子程序
       MOV AX, 1
NEXT:MUL CX
       LOOP NEXT
       RET
SUB1  ENDP

CODE ENDS
       END START
```

已知是自然数，所以只考虑无符号数。

子程序调用与返回

实例：计算=m!/(n!*(m-n)!)，其中m=8，n=3.

```
;计算=m!/(n!*(m-n)!)，其中m=8，n=3.
DATA  SEGMENT
  M     EQU 8
  N     EQU 3
  RES   DW ?
DATA  ENDS

CODE  SEGMENT
main proc far
        ASSUME CS:CODE,DS:DATA
        MOV AX,DATA
        MOV DS, AX
        MOV CX, N
        CALL SUB1  ;调子程序计算n!
        MOV BX, AX
        MOV CX, M
        CALL SUB1  ;调子程序计算m!
        MOV DX, 0
        DIV BX     ;m!/n!送 AX
        MOV BX, AX
        MOV CX, M
        SUB CX, N
        CALL SUB1   ;调用子程序，计算(m-n)!
        XCHG BX, AX
        MOV DX, 0
        DIV BX      ;m!/n!/ (m-n)! 送AX
        MOV RES, AX
        MOV AH, 4CH
        INT 21H
main endp

SUB1  PROC          ;计算阶乘的子程序
          MOV AX,1
NEXT:  MUL CX
          LOOP NEXT
          RET
SUB1  ENDP

CODE ENDS
  END
```

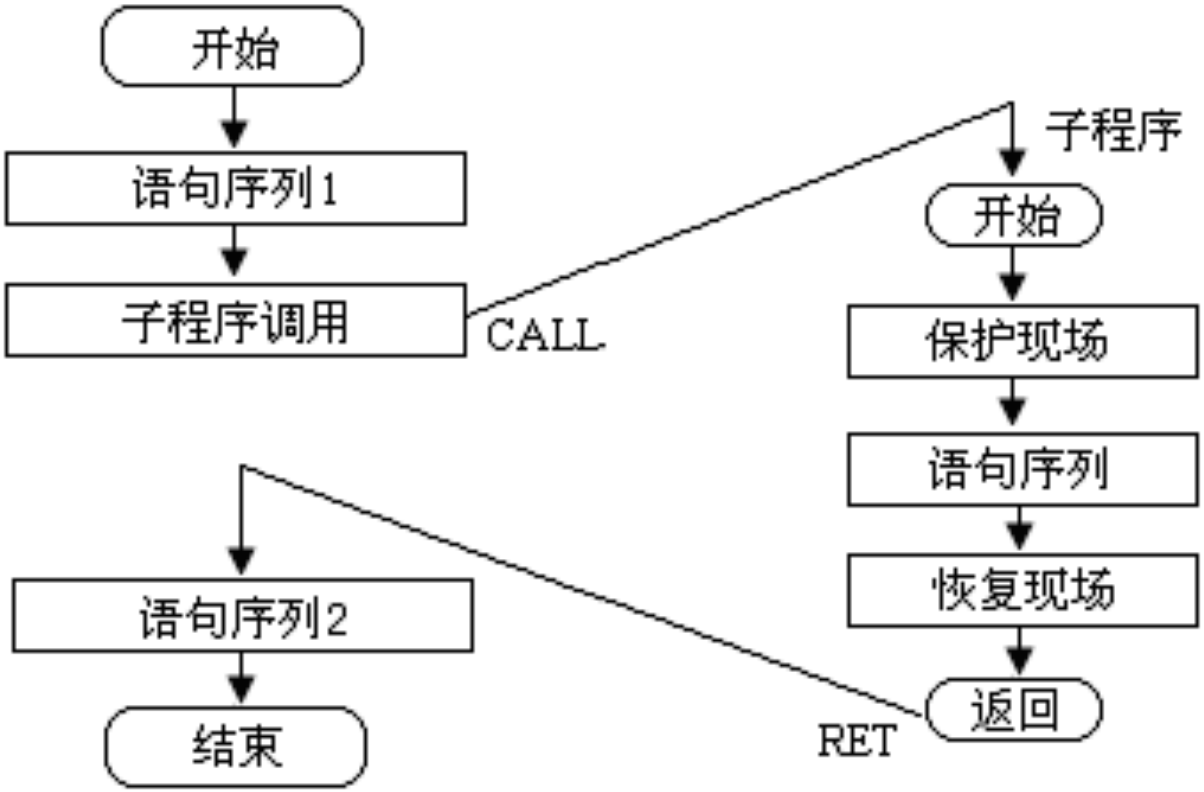
子程序调用与返回

寄存器内容的保护与恢复

通常主程序和子程序是分别编制的，所以它们可能会使用同一个寄存器。如果主程序中某个寄存器的内容在调用子程序后还要用，而子程序又恰好使用了同一个寄存器，当子程序修改了寄存器的内容后，返回到主程序时，该寄存器的内容也就不会是调用子程序前的内容，这样，常常会导致调用程序的出错。为此，必须进行现场保护。

wenjiecoder@outlook.com

子程序调用与返回



子程序调用与返回

保护现场和恢复现场的方法有两种：

（1）利用堆栈：

利用入栈指令PUSH保护现场，利用出栈指令POP恢复现场。

（2）利用内存单元：用传送指令MOV将寄存器的内容保存到指定的内存单元，需要时，再利用传送指令将指定的单元内容传送到指定的寄存器中加以恢复。

由于堆栈法比较方便，故使用的较多。

保护现场和恢复现场的工作可以在主程序中实现，也可以在子程序中实现，但要在同一部分实现。

wenjiecoder@outlook.com

子程序调用与返回

在主程序中保护和恢复现场的格式如下：

```
MAIN   PROC FAR
       :
       保护现场
       CALL   PN
       恢复现场
       :
MAIN ENDP

PN   PROC
     :；子程序
     :；主体
       RET
PN   ENDP
```

在子程序中保护和恢复现场的格式如下：

```
MAIN PROC FAR
     :
     :
     CALL   PN
     :
MAIN ENDP

PN   PROC
     保护现场
       :；子程序
       :；主体
     恢复现场
       RET
PN   ENDP
```

子程序设计方法

子程序的常见格式：

```
PROG        PROC                ;具有缺省属性的子程序
            PUSH   AX
            PUSH   BX
                   PUSH   CX    ; 保护现场
            PUSH   DX
•                   ┆
                   POP    DX
            POP    CX
            POP    BX           ; 恢复现场
            POP    AX
            RET                 ; 返回断点处
PROC        ENDP
```

注意：堆栈“先进后出”的操作特点，恢复寄存器的顺序不能搞错 。

子程序调用与返回

注意：

子程序的调用类型必须与定义类型一致。

若程序设计中使用了子程序，则需注意定义堆栈段。

思考：

既然Far属性既可以段内调用又可以段间调用，Near属性是否有必要存在?

Call指令实际上也是跳转到某一程序段执行，它和JMP指令有何不同?

段内跳转vs段间跳转，寻址的效率高。

Call会把它的下一条指令地址压入堆栈，然后跳转到子程序处；Ret会自动弹出返回地址，并返回执行。而JMP只是简单的跳转。
Call的本质相当于push+jmp；Ret的本质相当于pop+jmp。

子程序的引出与定义

子程序调用与和返回

子程序设计方法 (参数传递)

子程序嵌套

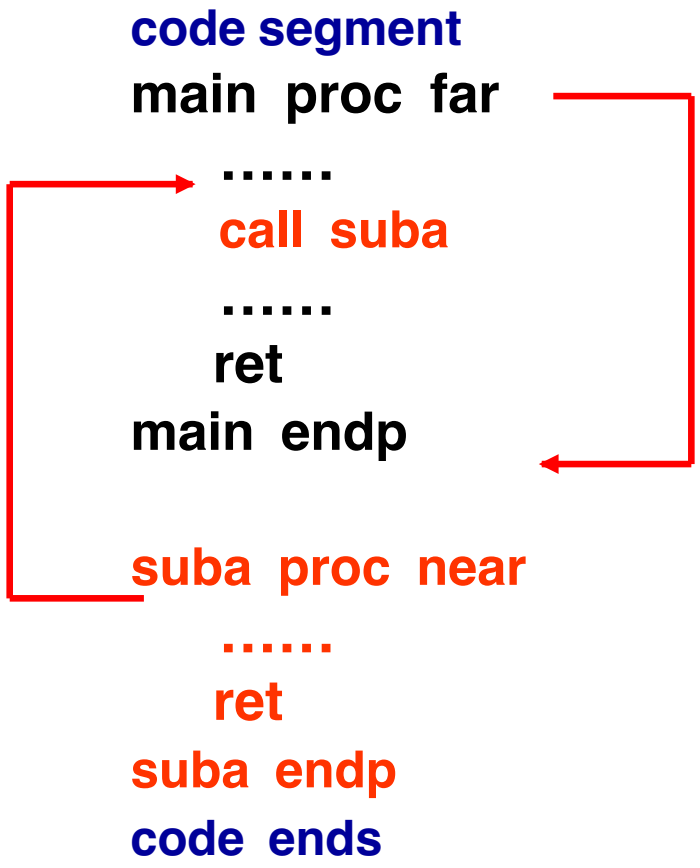
递归子程序

程序的连接

wenjiecoder@outlook.com

子程序设计方法

子程序可以放在代码段主程序开始执行之前的位置，也可放在代码段的末尾主程序执行终止后的位置。
当子程序和主程序在同一个代码段中，子程序的位置可以在主程序之前或之后定义。




```
code segment

main proc far
        .....
        call subr1
        .....
        ret

subr1 proc near
        .....
        ret
subr1 endp

main endp
code ends
```

子程序设计方法

例题：无参数传递的子程序：

```
;子程序功能：实现光标回车换行
dpcrlf proc   ;过程开始
        push ax       ;保护寄存器AX和DX
        push dx
        mov dl,0dh    ;显示回车
        mov ah,2
        int 21h
        mov dl,0ah    ;显示换行
        mov ah,2
        int 21h
        pop dx        ;恢复寄存器DX和AX
        pop ax
        ret      ;子程序返回
dpcrlf endp   ;过程结束
```

DOS系统调用INT 21H-单字符输出：

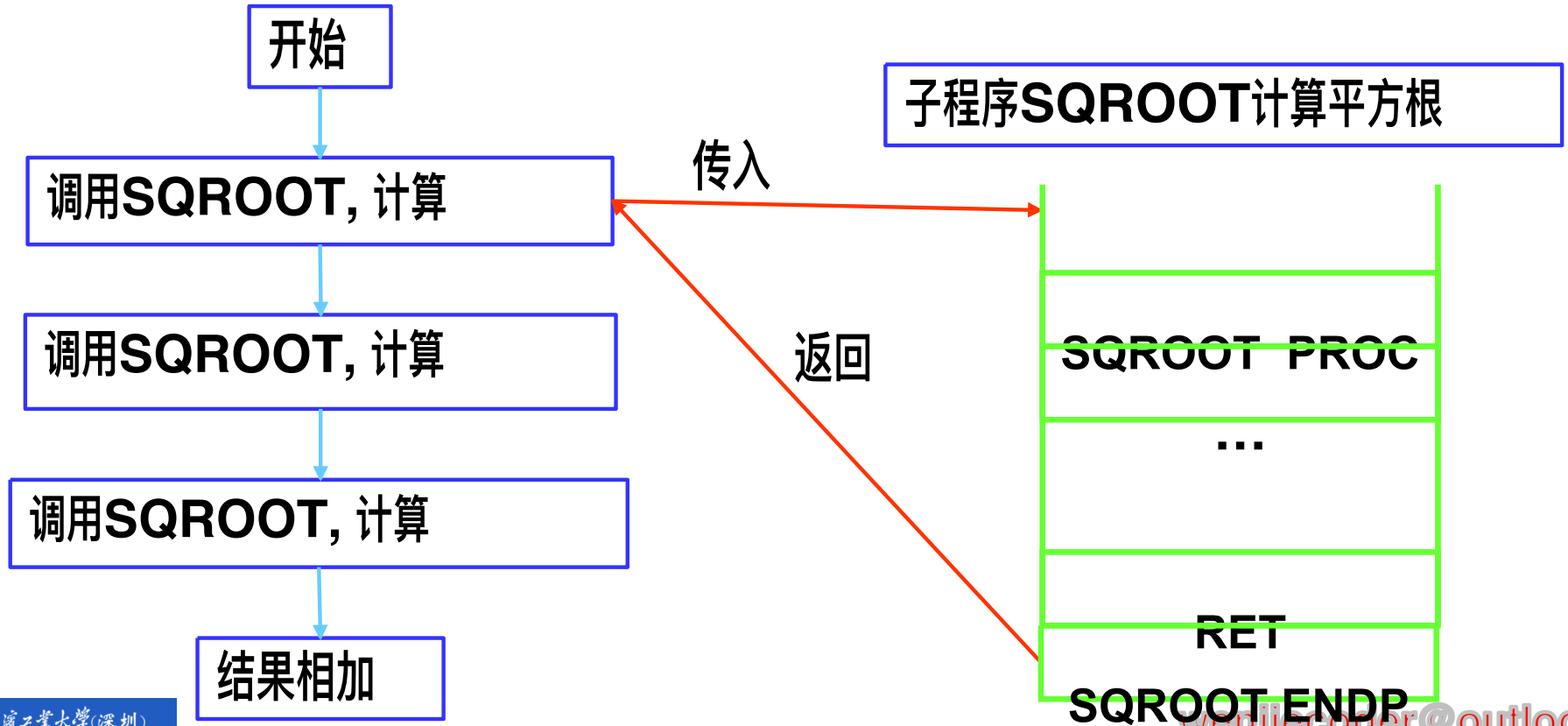
功能号2送AH

DL=输出字符

子程序参数传递

主程序在调用子程序时，经常需要传送一些参数给子程序，子程序运行完之后也经常需要回送一些结果信息给主程序。这种主程序和子程序之间的信息传递称为参数传递。

例： 设计程序计算的值



子程序参数传递

子程序说明文件：整个子程序的注释。
子程序说明文件应包含下述几项内容：

（1）子程序名（子程序入口地址）：用过程定义伪指令定义该过程时的过程名。

（2）子程序功能：用自然语言或数学语言等形式简单清楚地描述子程序完成的功能。

（3）入口条件：说明子程序有几个入口（输入）参数及其意义和存放位置。

主程序提供给子程序

（4）出口条件：说明子程序有几个出口（输出）参数，这些参数表示的意义及存放位置。

子程序返回给主程序

（5）受影响的寄存器：说明子程序运行后，哪些寄存器的内容被破坏了，以便使用者在调用该子程序之前注意保护现场。

例如：

（1）子程序名：SQROOT
（2）子程序功能：求一个整数字数据的平方根的整数部分
（3）入口条件：被开方数在CX中
（4）出口条件：平方根在CX中
（5）受影响的寄存器：CX及标志寄存器。

wenjiecoder@outlook.com

子程序参数传递

主程序与子程序之间的参数传递：

入口参数（输入参数）：

主程序提供给子程序

出口参数（输出参数）：

子程序返回给主程序

参数的形式：

数据本身（传值）

数据的地址（传址）

wenjiecoder@outlook.com

子程序参数传递

模块内参数传递常用的方法有以下四种：

（1）寄存器法；

（2）用变量传递参数；

（3）堆栈法；

（4）参数地址指针法。

以计算下述函数为例，详细说明各种参数传递方法。

例：编制程序计算

wenjiecoder@outlook.com

子程序参数传递

1. 寄存器法

把入口和出口参数存于约定的寄存器中。

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(一般建议保护)。

子程序说明文件如下：

参数很多时不能使用这种方法

(1) 子程序名：SQROOT

(2) 子程序功能：求一个整数字数据的平方根的整数部分。

(3) 入口条件：被开方数在CX中。

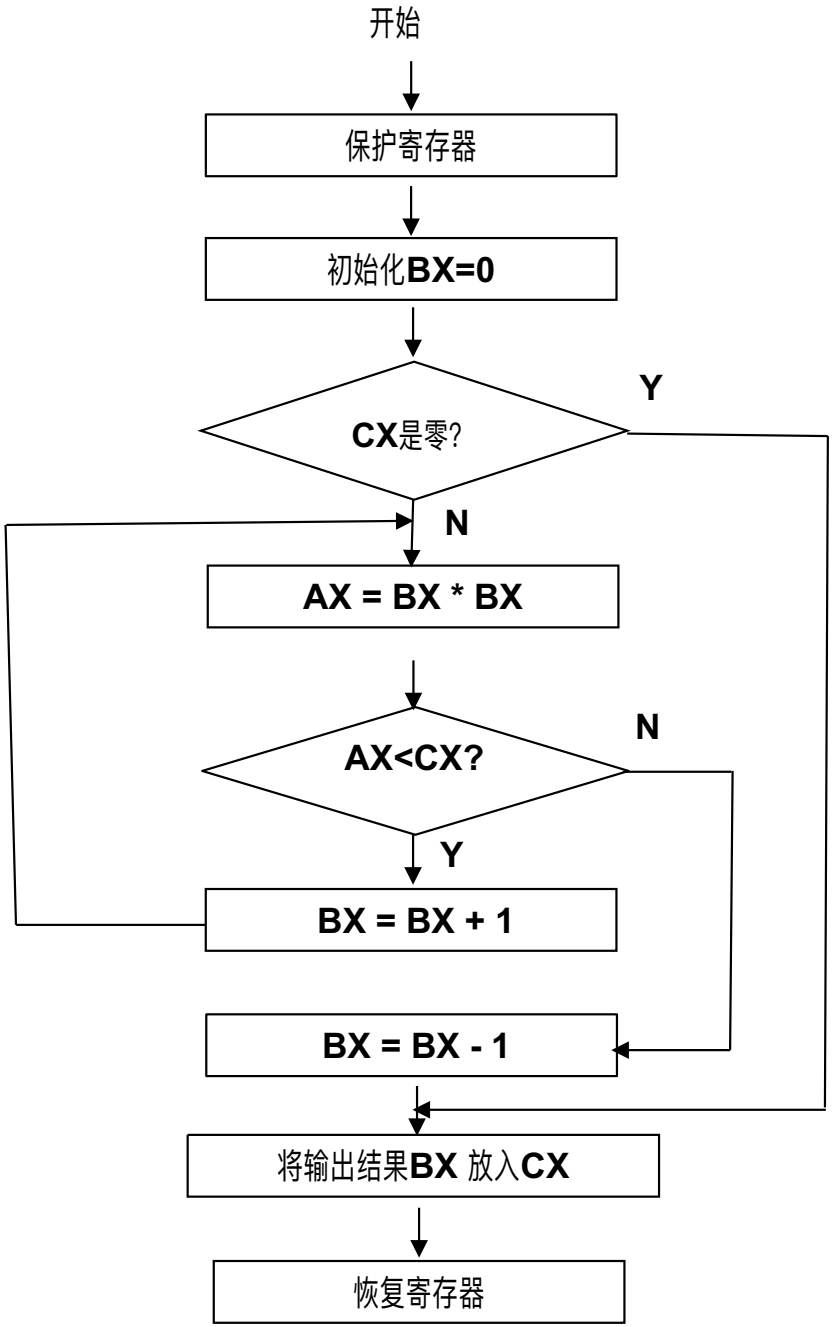
(4) 出口条件：平方根在CX中。

(5) 受影响的寄存器：CX及标志寄存器。

wenjiecoder@outlook.com

子程序参数传递

画出程序流程图



子程序参数传递

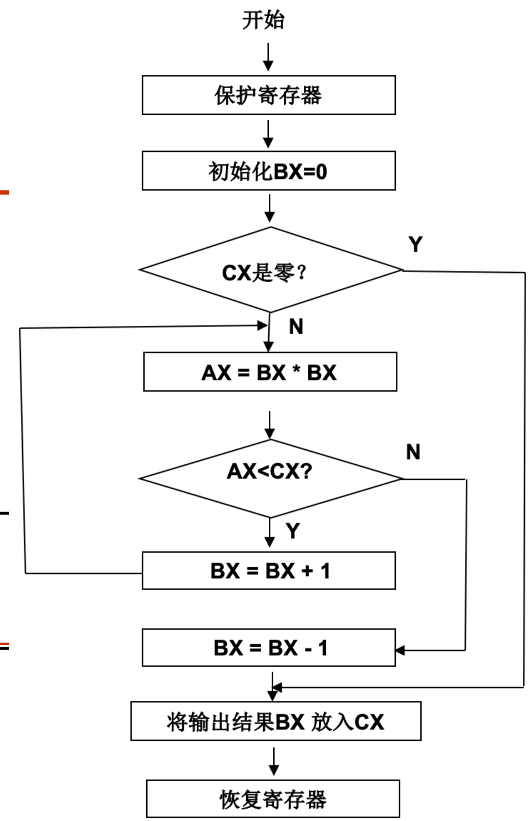
```
;子程序清单如下:
SQROOT      PROC
        PUSH    AX
        PUSH    BX
        PUSH    DX
        XOR     BX, BX
        AND     CX, CX
        JZ      SQRT3
SQRT1:
        MOV     AX, BX
        MUL     BX
        CMP     CX, AX
        JB      SQRT2
        INC     BX
        JMP     SQRT1
SQRT2:
        DEC     BX
SQRT3:
        MOV     CX, BX
        POP     DX
        POP     BX
        POP     AX
        RET
SQROOT      ENDP
```

保护寄存器

BX归零初始化，比mov更高效，指令占用内存空间更小

Check CX是否为零

恢复寄存器



子程序参数传递

```
; 主程序清单如下:
SSEG    SEGMENT  STACK
STK     DB       20 DUP(0)
SSEG    ENDS
DSEG    SEGMENT
PX      DW       1000
PY      DW       2000
RLT     DW       0
DSEG    ENDS
CSEG    SEGMENT
        ASSUME   CS:CSEG, DS:DSEG, SS:SSEG
START:  MOV      AX, DSEG
        MOV      DS,AX
        MOV      AX, SSEG
        MOV      SS, AX
```

定义堆栈段

定义数据段

定义代码段

```
        MOV      CX,PX
        SHL      CX,1
        CALL     SQROOT
        PUSH     CX
        MOV      CX,PY
        SHL      CX,1
        ADD      CX,PY
        CALL     SQROOT
        POP      AX
        ADD      AX, CX
        MOV      CX,150
        CALL     SQROOT
         ADD      AX, CX
        MOV      RLT, AX
        MOV      AH, 4CH
        INT      21H
CSEG    ENDS
             END      START
```

通过左移操作乘2更高效

通过左移操作乘2再加1
完成3倍乘法更高效

子程序参数传递

例:计算字节数组的校验和（校验和是指不记进位的累加）。

子程序名： checksuma

子程序功能：求一组字节数据的校验和；
入口参数：CX=元素个数，
DS:BX=数组的段地址：偏移地址
出口参数：AL=校验和；
受影响的寄存器：CX、BX、AL及标志寄存器。

主程序：

```
.code
        .startup                    ;定义程序初始入口点，并且产生设置DS, SS, SP
        的代码（含有DS←数组的段地址）
        mov bx, offset array        ;BX←数组的偏移地址
        mov cx, count               ;CX←数组的元素个数
        call checksuma              ;调用求和过程
        mov result, al              ;处理出口参数
        .exit 0                     ; 退出程序并返回操作系统
END
```


wenjiecoder@outlook.com

子程序参数传递

子程序：

```
checksuma    proc
        xor   al,al       ;累加器清0
suma:   add   al,[bx]     ;求和
        inc   bx          ;指向下一个字节
        loop  suma
        ret
checksuma    endp
```


wenjiecoder@outlook.com

子程序参数传递

2. 用变量传递参数

♦ 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。

子程序和调用程序在同一个源文件

子程序说明文件如下：

(1) 子程序名：SQROOT。

(2) 子程序功能：求一个整数字数据的平方根的整数部分。

(3) 入口条件：被开方数在ARGX单元。

(4) 出口条件：平方根在ROOT字单元。

(5) 受影响的寄存器：标志寄存器F。

子程序参数传递

```
; 主程序清单如下:
SSEG    SEGMENT  STACK
STK     DB       20 DUP(0)
SSEG    ENDS
DSEG    SEGMENT
PX      DW       1000
PY      DW       2000
RLT     DW       0
ARGX    DW       0
ROOT    DW       0
DSEG    ENDS
CSEG    SEGMENT
        ASSUME   CS:CSEG
        ASSUME   DS:DSEG
        ASSUME   SS:SSEG
START:
MOV     AX, DSEG
MOV     DS, AX
MOV     AX, SSEG
MOV     SS, AX
MOV     DX, PX
SHL     DX, 1
MOV     ARGX, DX
CALL    SQROOT
MOV     AX, ROOT
MOV     DX, PY
SHL     DX, 1
ADD     DX, PY
MOV     ARGX, DX
CALL    SQROOT
        ADD     AX, ROOT
        MOV     ARGX, 150
        CALL    SQROOT
        ADD     AX, ROOT
        MOV     RLT, AX
        MOV     AH, 4CH
        INT     21H
        ……
CSEG    ENDS
        END     START
```

子程序参数传递

```
;子程序清单如下:
SQROOT   PROC
         PUSH  AX
         PUSH  BX
         PUSH CX
         PUSH DX
         XOR   BX, BX
         MOV   CX,  ARGX
         AND   CX, CX
         JZ    SQRT3
SQRT1:
         MOV   AX, BX
         MUL   BX
         CMP   CX, AX
         JB    SQRT2
         INC   BX
         JMP   SQRT1
SQRT2:
         DEC   BX
SQRT3:
         MOV   ROOT, BX
         POP   DX
         POP   CX
         POP   BX
         POP   AX
         RET
SQROOT  ENDP
```

子程序参数传递

堆栈法

- 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。

优点：
参数不占用寄存器，和存储单元。参数存放在公共堆栈区，处理完后恢复。参数个数一般不限。

缺点：
由于参数和子程序混杂在一起，存取参数时候必须小心计算它在堆栈中的位置，要注意堆栈区的状态，以及数据的保存和恢复。

子程序说明文件如下：

(1) 子程序名：**SQROOT**。

(2) 子程序功能：**求一个整数字数据的平方根的整数部分**。

(3) 入口条件：**被开方数在堆栈中。**

(4) 出口条件：**平方根在堆栈中。**

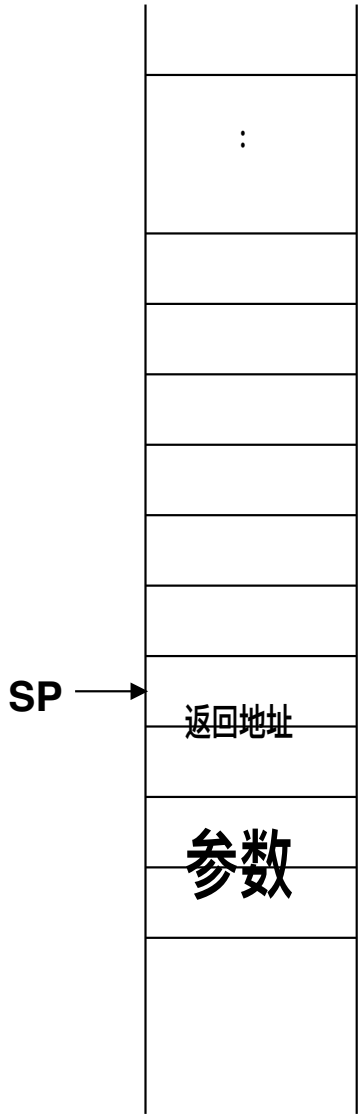
(5) 受影响的寄存器：标志寄存器。

子程序参数传递

```
PUSH    CX              ;堆栈传递入口参数
CALL    SQROOT
POP     CX              ;堆栈返回运算结果
```



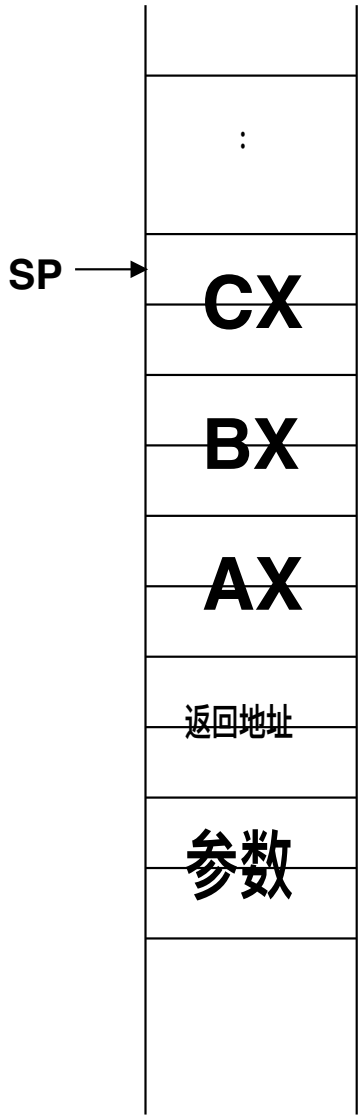
```
;子程序清单如下:
SQROOT  PROC
        PUSH    AX
        PUSH    BX
        PUSH    CX
        ADD     SP, 8
        POP     CX
        XOR     BX, BX
        AND     CX, CX
        JZ      SQRT3
SQRT1:
        MOV     AX, BX
        MUL     BX
        CMP     CX, AX
        JB      SQRT2
        INC     BX
        JMP     SQRT1
SQRT2:
        DEC     BX
SQRT3:
        PUSH    BX
        SUB     SP, 8
        POP     CX
        POP     BX
        POP     AX
        RET
SQROOT  ENDP
```



刚进入时堆栈的情况

子程序参数传递

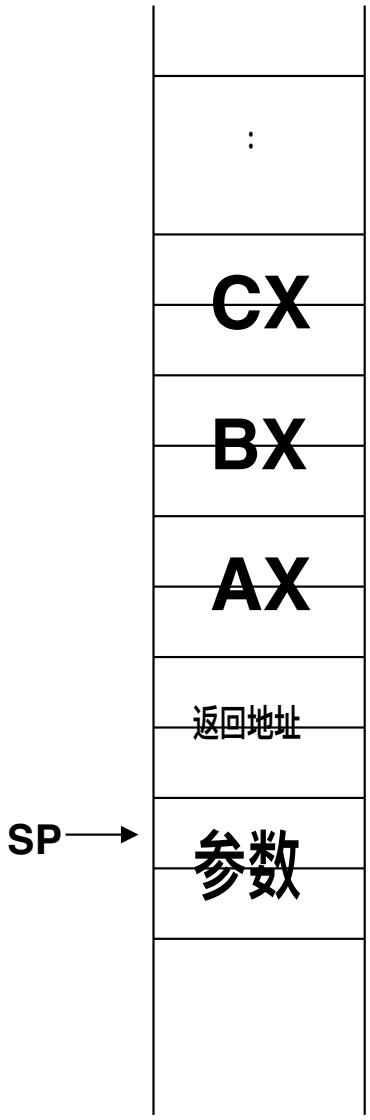
```
;子程序清单如下:
SQROOT   PROC
         PUSH    AX
         PUSH    BX
         PUSH    CX
         ADD     SP, 8
         POP     CX
         XOR     BX, BX
         AND     CX, CX
         JZ      SQRT3
SQRT1:
         MOV     AX, BX
         MUL     BX
         CMP     CX, AX
         JB      SQRT2
         INC     BX
         JMP     SQRT1
SQRT2:
         DEC     BX
SQRT3:
         PUSH    BX
         SUB     SP, 8
         POP     CX
         POP     BX
         POP     AX
         RET
SQROOT   ENDP
```



执行完保护现场后
堆栈的情况

子程序参数传递

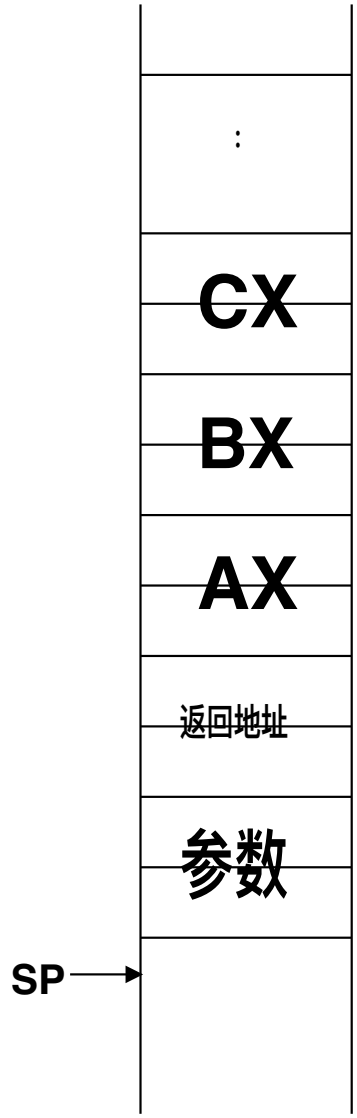
```
;子程序清单如下:
SQROOT   PROC
         PUSH    AX
         PUSH    BX
         PUSH    CX
         ADD     SP, 8
         POP     CX
         XOR     BX, BX
         AND     CX, CX
         JZ      SQRT3
SQRT1:
         MOV     AX, BX
         MUL     BX
         CMP     CX, AX
         JB      SQRT2
         INC     BX
         JMP     SQRT1
SQRT2:
         DEC     BX
SQRT3:
         PUSH    BX
         SUB     SP, 8
         POP     CX
         POP     BX
         POP     AX
         RET
SQROOT   ENDP
```



执行完SP+8后
堆栈的情况

子程序参数传递

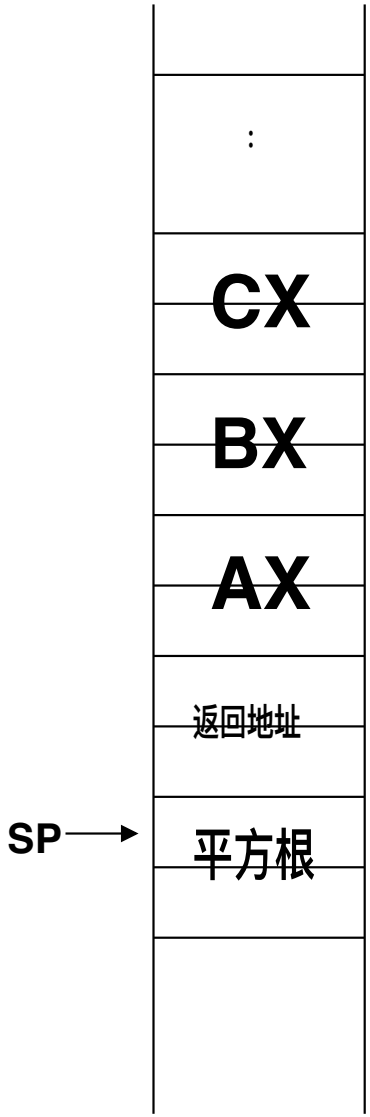
```
;子程序清单如下:
SQROOT   PROC
         PUSH    AX
         PUSH    BX
         PUSH    CX
         ADD     SP, 8
         POP     CX
         XOR     BX, BX
         AND     CX, CX
         JZ      SQRT3
SQRT1:
         MOV     AX, BX
         MUL     BX
         CMP     CX, AX
         JB      SQRT2
         INC     BX
         JMP     SQRT1
SQRT2:
         DEC     BX
SQRT3:
         PUSH    BX
         SUB     SP, 8
         POP     CX
         POP     BX
         POP     AX
         RET
SQROOT   ENDP
```



执行完POP CX后
堆栈的情况

子程序参数传递

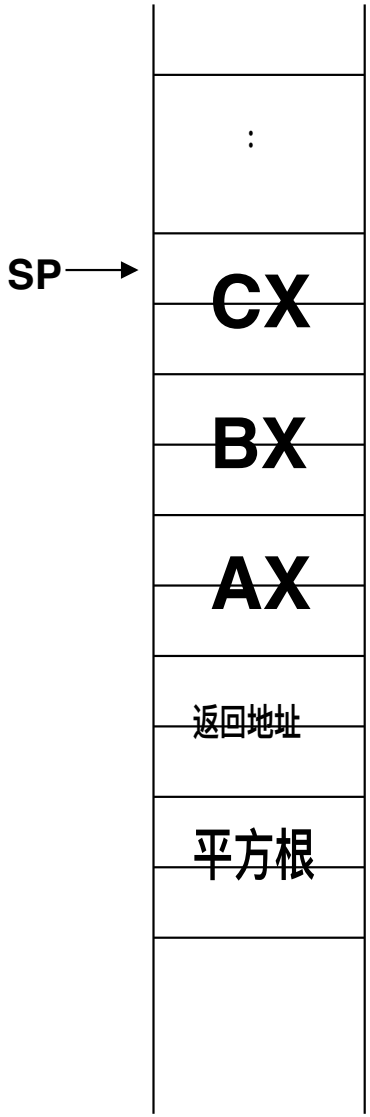
```
;子程序清单如下:
SQROOT   PROC
         PUSH    AX
         PUSH    BX
         PUSH    CX
         ADD     SP, 8
         POP     CX
         XOR     BX, BX
         AND     CX, CX
         JZ      SQRT3
SQRT1:
         MOV     AX, BX
         MUL     BX
         CMP     CX, AX
         JB      SQRT2
         INC     BX
         JMP     SQRT1
SQRT2:
         DEC     BX
SQRT3:
         PUSH    BX
         SUB     SP, 8
         POP     CX
         POP     BX
         POP     AX
         RET
SQROOT   ENDP
```



执行完PUSH BX后堆栈的情况

子程序参数传递

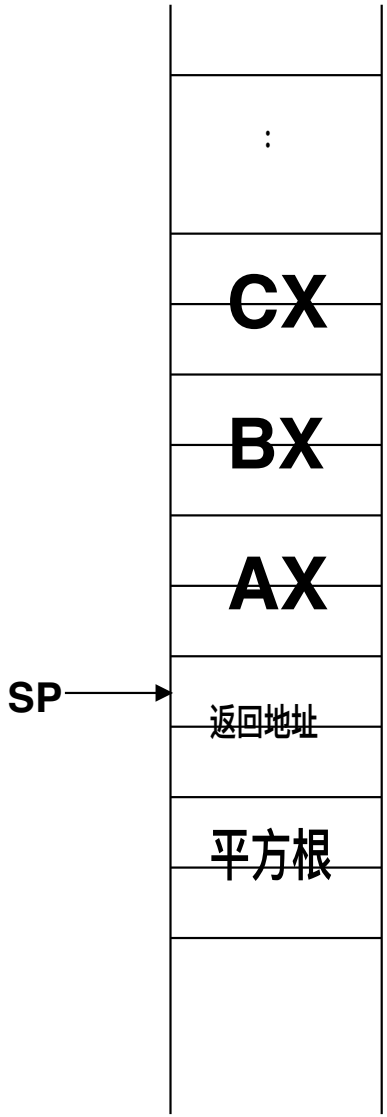
```
;子程序清单如下:
SQROOT   PROC
         PUSH    AX
         PUSH    BX
         PUSH    CX
         ADD     SP, 8
         POP     CX
         XOR     BX, BX
         AND     CX, CX
         JZ      SQRT3
SQRT1:
         MOV     AX, BX
         MUL     BX
         CMP     CX, AX
         JB      SQRT2
         INC     BX
         JMP     SQRT1
SQRT2:
         DEC     BX
SQRT3:
         PUSH    BX
         SUB     SP, 8
         POP     CX
         POP     BX
         POP     AX
         RET
SQROOT   ENDP
```



执行完SP-8后
堆栈的情况

子程序参数传递

```
;子程序清单如下:
SQROOT   PROC
         PUSH    AX
         PUSH    BX
         PUSH    CX
         ADD     SP, 8
         POP     CX
         XOR     BX, BX
         AND     CX, CX
         JZ      SQRT3
SQRT1:
         MOV     AX, BX
         MUL     BX
         CMP     CX, AX
         JB      SQRT2
         INC     BX
         JMP     SQRT1
SQRT2:
         DEC     BX
SQRT3:
         PUSH    BX
         SUB     SP, 8
         POP     CX
         POP     BX
         POP     AX
         RET
SQROOT   ENDP
```



执行完恢复现场后堆栈的情况

子程序参数传递

主程序:

```
SSEG    SEGMENT
STK     DB 20 DUP(0)
SSEG    ENDS

DSEG    SEGMENT
PX      DW      50
DSEG    ENDS

CSEG SEGMENT
        ASSUME CS:CSEG,DS:DSEG,SS:SSEG
START:  MOV AX, DSEG
MOV     DS,AX
MOV     AX, SSEG
MOV     SS, AX
MOV     CX,PX
SHL     CX,1
PUSH    CX                  ;堆栈传递入口参数
CALL    SQROOT
POP     CX                  ;堆栈返回运算结果
MOV     AH, 4CH
INT 21H
```

子程序参数传递

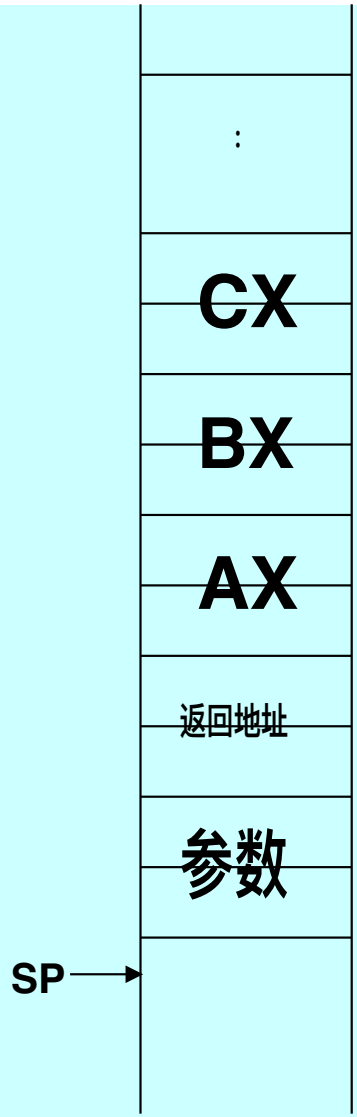
思考题:

如果子程序在得到参数后

还要使用堆栈将会如何?

怎样解决?

```
;子程序清单如下:
SQROOT   PROC
         PUSH    AX
         PUSH    BX
         PUSH    CX
         ADD     SP, 8
         POP     CX
         XOR     BX, BX
         AND     CX, CX
         JZ      SQRT3
SQRT1:
         MOV     AX, BX
         MUL     BX
         CMP     CX, AX
         JB      SQRT2
         INC     BX
         JMP     SQRT1
SQRT2:   DEC     BX
SQRT3:
         PUSH    BX
         SUB     SP, 8
         POP     CX
         POP     BX
         POP     AX
         RET
SQROOT   ENDP
```

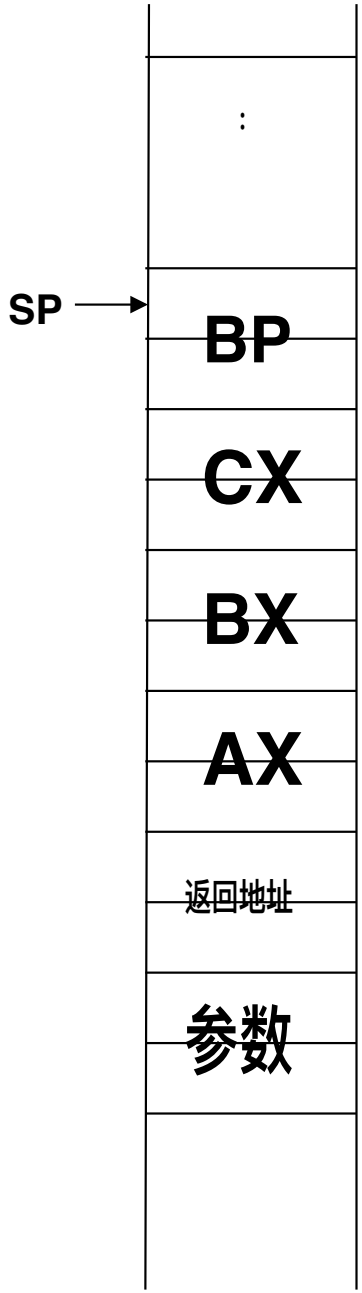


执行完POP CX后
堆栈的情况

子程序参数传递

用寻址指令代替POP指令，不改变SP的值!


```
;子程序清单如下:
SQROOT  PROC
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    BP
        MOV     BP, SP
        MOV     CX, [BP+10]
        XOR     BX, BX
        AND     CX, CX
        JZ      SQRT3
SQRT1:  MOV     AX, BX
        MUL     BX
        CMP     CX, AX
        JB      SQRT2
        INC     BX
        JMP     SQRT1
SQRT2:
        DEC     BX
SQRT3:
        MOV     [BP+10], BX
        POP     BP
        POP     CX
        POP     BX
        POP     AX
        RET
SQROOT  ENDP
```



4. 约定参数地址指针法

在主程序中建立一个地址表（连续内存空间），把需要传送给子程序的参数都存放在地址表中，然后把地址表的首地址通过寄存器传送到子程序中。

适用于大量参数的传递

wenjiecoder@outlook.com

4. 约定参数地址指针法

例：设内存有三组无符号字数据，三组数据的首地址分别为LIST1，LIST2和LIST3，数据个数分别存放在CNT1，CNT2和CNT3单元。编制程序计算三组数据中最小数之和并存入SUM开始的单元。

子程序说明文件如下：

(1) 子程序名：FMIN

(2) 子程序功能：求一组无符号字数据的最小值

(3) 入口条件：数据首地址在SI中，元素个数在CX中

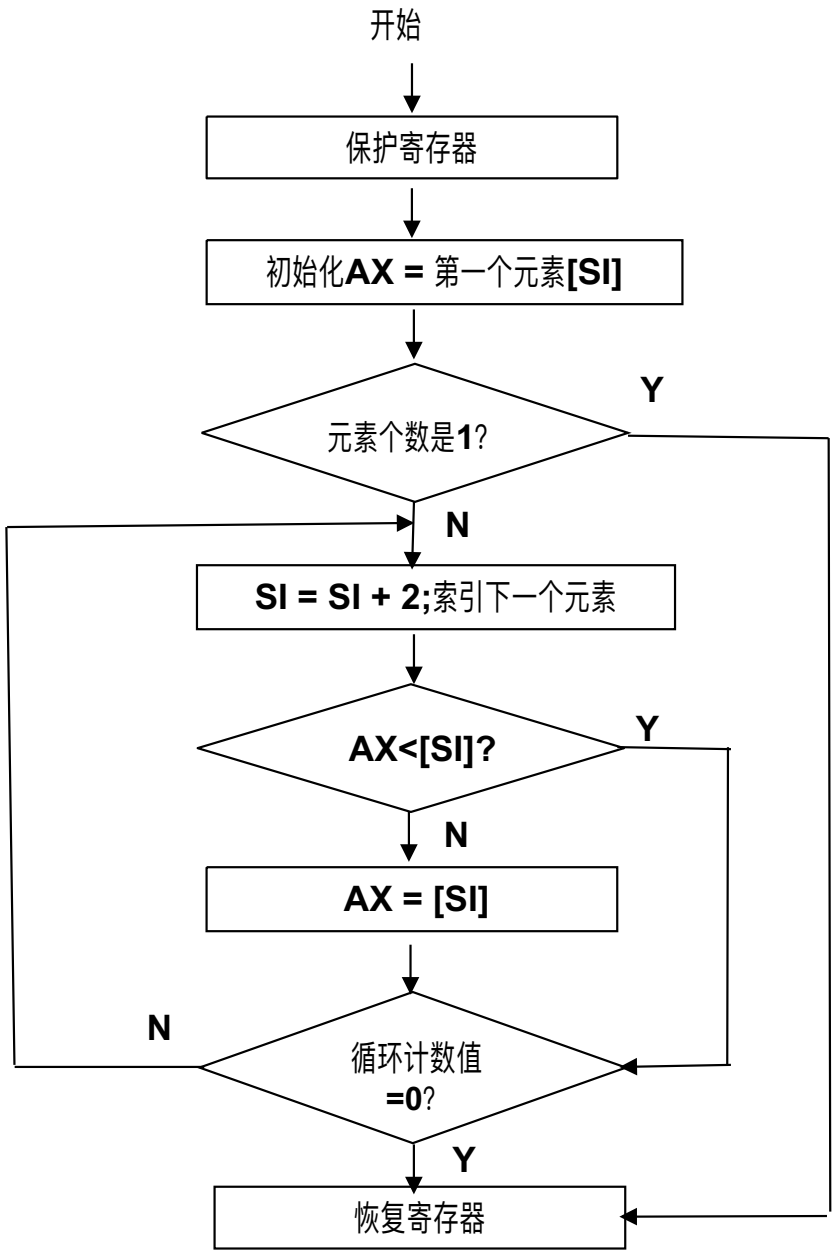
(4) 出口条件：最小值在AX中

(5) 受影响的寄存器：AX及标志寄存器。

子程序参数传递

子程序清单如下：

```
FMIN    PROC
        PUSH    SI
        PUSH    CX
        MOV     AX, [SI]
        DEC     CX
        JZ      RETN；数组长度为1，则结束
FMIN2:  ADD     SI, 2
        CMP     AX, [SI]
        JB      FMIN1
        MOV     AX, [SI]
FMIN1:  LOOP    FMIN2
RETN:   POP     CX
        POP     SI
        RET
FMIN    ENDP
```



子程序参数传递

```
; 主程序清单如下:
SSEG    SEGMENT STACK
STK     DB      20 DUP (0)
SSEG    ENDS
DSEG    SEGMENT
LIST1   DW      100, 110, 48, 130, 24, 150
CNT1    DW      6
LIST2   DW      200, 20, 220, 6, 240
CNT2    DW      5
LIST3   DW      300, 15, 340, 350
CNT3    DW      4
SUM     DW      0
DSEG    ENDS
CSEG    SEGMENT
        ASSUME  CS: CSEG, DS: DSEG
        ASSUME  SS: SSEG
START: MOV      AX, DSEG
        MOV     DS, AX
        MOV     AX, SSEG
        MOV     SS, AX
        LEA     SI, LIST1
        MOV     CX, CNT1
        CALL    FMIN
        MOV     BX, AX
        LEA     SI, LIST2
        MOV     CX, CNT2
        CALL    FMIN
        ADD     BX, AX
        LEA     SI, LIST3
        MOV     CX, CNT3
        CALL    FMIN
        ADD     BX, AX
        MOV     SUM, BX
        MOV     AH, 4CH
        INT     21H
CSEG    ENDS
        END     START
```

子程序的引出与定义

子程序调用与和返回

子程序设计方法

子程序嵌套

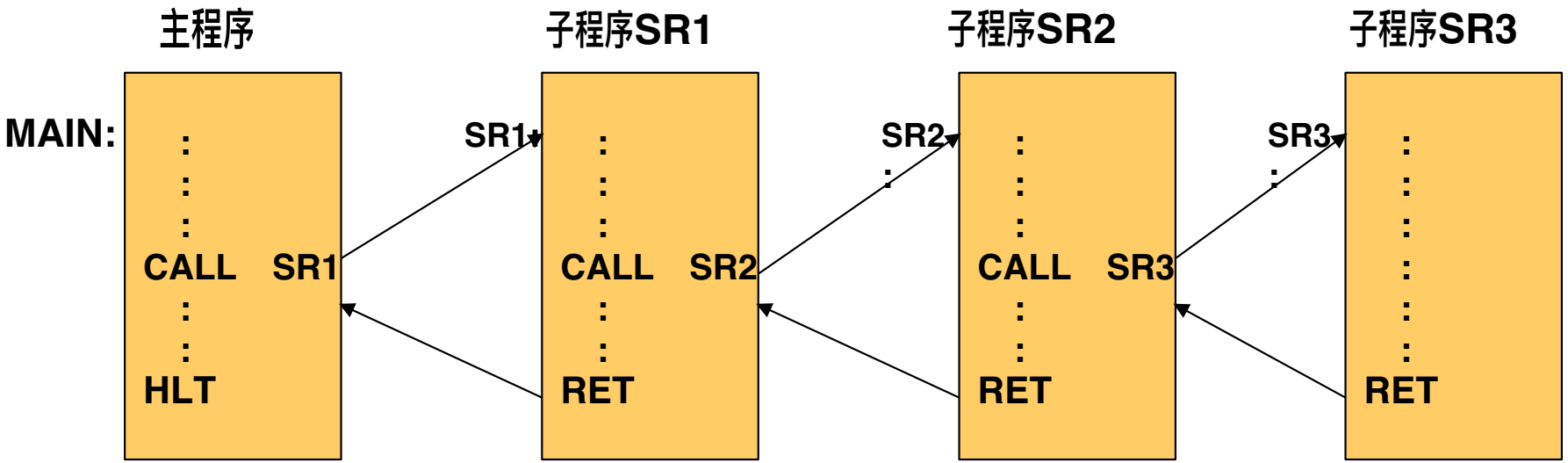
递归子程序

程序的连接

wenjiecoder@outlook.com

子程序嵌套

所谓子程序嵌套是指在一个子程序的内部再调用其它子程序,其结构可描述如下:



子程序嵌套

子程序嵌套注意事项：

嵌套深度（即嵌套的层次数）逻辑上没有限制，但由于子程序的调用需要在堆栈中保存返回地址以及寄存器等数据，因此实际上受限于开设的堆栈空间。

嵌套子程序的设计并没有什么特殊要求，除子程序的调用和返回应正确使用CALL和RET指令外，还要注意寄存器的保存与恢复，以避免各层子程序之间因寄存器使用冲突而出错。

wenjiecoder@outlook.com

子程序嵌套

子程序嵌套的例子：

有如下子程序说明文件：

（1）子程序名：HTOA；

（2）子程序功能：将一位十六进制数转换为ASCII码；

（3）入口条件：要转换的数据在AL中的低四位；

（4）出口条件：十六进制数的ASCII码在AL中；

（5）受影响的寄存器：AL和标志寄存器。

子程序清单如下：

```
HTOA    PROC
        AND     AL, 0FH
        CMP     AL, 10
        JC      HTOA1
        ADD     AL, 7
HTOA1:  ADD     AL, 30H
        RET
HTOA    ENDP
```

0~9：30H~39H，差值为30H

A~F：41H~46H，差值为37H

子程序嵌套

子程序嵌套的例子：

现在我们要编制将AL中的两位十六进制数转换为ASCII码的子程序BHTOA，就可以利用上述子程序实现。BHTOA的子程序说明文件如下：

（1）子程序名：BHTOA；

（2）子程序功能：将两位十六进制数转换为ASCII码；

（3）入口条件：要转换的数据在AL中；

（4）出口条件：高位十六进制数的ASCII码在AH中；

低位十六进制数的ASCII码在AL中；

（5）受影响的寄存器：AX和标志寄存器。

子程序清单如下：

```
BHTOA   PROC
        PUSH    CX
        MOV     CH, AL
        MOV     CL, 4
        SHR     AL, CL
        CALL    HTOA
        MOV     AH, AL
        MOV     AL, CH
        CALL    HTOA
        POP     CX
        RET
BHTOA   ENDP
```

子程序嵌套

思考题：如何编制将AX中的四位十六进制数转换为ASCII码的子程序QHTOA?

可以利用上述子程序实现。QHTOA的子程序说明文件如下：

(1) 子程序名：QHTOA；

(2) 子程序功能：将四位十六进制数转换为ASCII码；

(3) 入口条件：要转换的数据在AX中；

(4) 出口条件：

最高位数的ASCII码在BH中；次高位数的ASCII码在BL中；

次低位数的ASCII码在AH中；最低位数的ASCII码在AL中；

(5) 受影响的寄存器：BX，AX和标志寄存器。

子程序清单如下：

```
QHTOA   PROC
        PUSH    AX
        MOV     AL, AH
        CALL    BHTOA
        MOV     BX, AX
        POP     AX
        CALL    BHTOA
        RET
QHTOA   ENDP
```

子程序嵌套

```
QHTOA  PROC
   PUSH  AX
   MOV   AL, AH
   CALL BHTOA
   MOV   BX, AX
   POP    AX
   CALL BHTOA
   RET
QHTOA  ENDP
```

```
BHTOA  PROC
   PUSH  CX
   MOV   CH, AL
   MOV   CL, 4
   SHR    AL, CL
   CALL HTOA
   MOV AH, AL
   MOV AL, CH
   CALL HTOA
   POP   CX
   RET
BHTOA ENDP
```

```
HTOA    PROC
   AND    AL, 0FH
   CMP    AL, 10
   JC       HTOA1
   ADD    AL, 7
  HTOA1:
   ADD    AL, 30H
   RET
HTOA    ENDP
```

第7&8讲：子程序结构

子程序的引出与定义

子程序调用与和返回

子程序设计方法

子程序嵌套

递归子程序

程序的连接

wenjiecoder@outlook.com

递归子程序

子程序能直接或间接地调用自身称为递归调用，具有递归调用性质的子程序称为递归子程序。递归子程序用于解递归函数。

注意：

- 递归函数的递归过程必须是有限的。即每个递归函数都有一个非递归终值，递归过程一旦求得该值，就结束递归。
- 递归子程序必须采用寄存器或堆栈传递参数，递归深度受堆栈空间的限制。

wenjiecoder@outlook.com

递归子程序

例：编程计算**n!**

分析：

阶乘函数就是一个递归函数，对于任一个大于等于0的正整数N，其函数值定义为：

如：FACT为计算阶乘的函数，则4! 计算为：

FACT(4)= 4*FACT(3)

=4*3*FACT(2)

=4*3*2*FACT(1)

=4*3*2*1*FACT(0)

=4*3*2*1*1

=24

递归子程序

计算阶乘函数的算法归纳如下：

(1) 测试N=0吗？是，则令FACT（N）=1，返回；

(2) 保存N，并令N=N-1，调用自身求得FACT（N-1）；

(3) 顺序取出保存的N值（后保存的先取出）；

(4) 计算FACT（N）=N*FACT（N-1），并返回。

依据上述算法编制的计算阶乘函数的子程序说明文件及清单如下：

子程序说明文件：

(1) 子程序名：FACT；

(2) 子程序功能：计算N的阶乘（0≤N≤8）；

(3) 入口条件：N值在BX中；

(4) 出口条件：N！值在AX中；

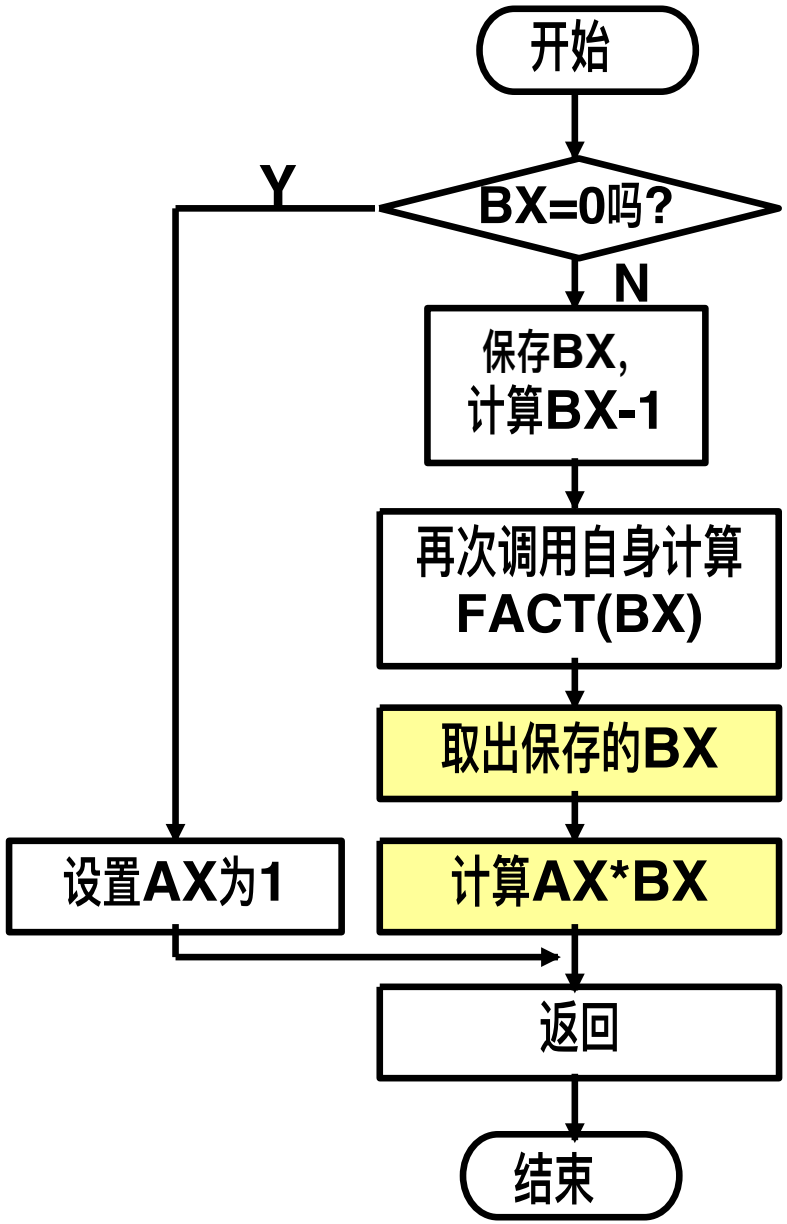
(5) 受影响的寄存器：AX，BX，DX和标志寄存器F。

wenjiecoder@outlook.com

递归子程序

(1)测试*N*=0吗? 是，则令FACT(*N*)=1，返回;
(2)保存*N*，并令*N*=*N*-1，调用自身求得FACT(*N*-1);
(3)顺序取出保存的N值（后保存的先取出）；
(4)计算FACT(*N*)=*N**FACT(*N*-1)，并返回。

```
FACT   PROC
       AND    BX,BX
       JZ     FACT1
       PUSH   BX
       DEC    BX
       CALL   FACT
       POP    BX
       MUL    BX
       RET
FACT1: MOV    AX,1
       RET
FACT   ENDP
```



递归子程序

```
FACT    PROC
        AND     BX,BX
        JZ      FACT1
        PUSH    BX
        DEC     BX  ;BX=1
        CALL    FACT
        POP     BX  ;BX=2
        MUL     BX
        RET
FACT1:  MOV     AX,1
        RET
FACT    ENDP
```

```
FACT    PROC
        AND     BX,BX
        JZ      FACT1
        PUSH    BX
        DEC     BX   ;BX=0
        CALL    FACT
        POP     BX   ;BX=1
        MUL     BX
        RET
FACT1:  MOV     AX,1
        RET
FACT    ENDP
```

```
FACT    PROC
        AND     BX,BX
        JZ      FACT1
        PUSH    BX
        DEC     BX
        CALL    FACT
        POP     BX
        MUL     BX
        RET
FACT1:  MOV     AX,1
        RET
FACT    ENDP
```

例：求2！，BX=2

递归子程序

例：求S=X！+Y！

。设X，Y存放在MNVAL开始的单元。其值在4~7之间。计算结果存入RLT单元。

程序清单如下：

```
DSEG    SEGMENT
        MNVAL   DW  4, 7
        RLT     DW  0
DSEG    ENDS
SSEG    SEGMENT STACK
        STK     DB      50 DUP (0)
SSEG    ENDS
CSEG    SEGMENT
        ASSUME CS: CSEG, DS: DSEG
        ASSUME SS: SSEG
START:  MOV     AX, DSEG
        MOV     DS, AX
        MOV     AX, SSEG
        MOV     SS, AX
        LEA     SI, MNVAL
        MOV     BX, [SI]
        CALL    FACT
        PUSH    AX
        MOV     BX, [SI+2]
        CALL    FACT
        POP     BX
        ADD     AX, BX
        MOV     RLT, AX
        MOV     AH, 4CH
        INT     21H
CSEG    ENDS
        END     START
```


wenjiecoder@outlook.com

子程序的引出与定义

子程序调用与和返回

子程序设计方法

子程序嵌套

递归子程序

程序的连接

wenjiecoder@outlook.com

子程序的连接

开发大型应用程序时，通常把复杂的程序分成很多功能独立的模块，分别编写成子程序，对各个子程序模块单独进行汇编产生相应的目标模块（**OBJ**文件），最后再用连接程序把它们连接起来，形成一个完整的可执行程序。采用这种模块化程序设计方法，程序不但结构清晰，也便于调试。

采用模块化程序设计，各模块之间会存在着相互调用，即一个模块会引用在另一个模块中定义的标识符（包括变量、标号、过程名等）。

标识符有两种：

1）在本模块中定义，供本模块使用的标识符称为局部标识符；**2**）在一个模块中定义，而又在另一个模块中引用的标识符称为外部标识符。

需要遵循的原则：

① 声明共用的变量、过程等
② 实现正确的段组合
③ 处理好参数传递问题

模块连接方法

（1）在汇编时把各个模块连接在一起

使用INCLUDE伪指令，把几个已独立编制好的.ASM文件在汇编时连接在一起，形成一个完整的.OBJ文件。

INCLUDE　包含伪指令

格式：INCLUDE　FN

;FN为源程序模块的文件名，每一个INCLUDE后面只能有一个文件名。

注意：

- 利用INCLUDE伪指令包含其它文件，其实质仍然是一个源程序，只是分成几个文件书写；
- 被包含的文件不能独立汇编，是依附主程序而存在的；
- 所以合并的源程序之间的各种标识符，应该统一规定，不能发生冲突；
- 另外由于是源程序的结合，每次汇编都要包括对被包含文件文本的汇编，增加了汇编的时间。

子程序的连接

例：从键盘输入一个数n（(0≤n≤8)，求
程序模块，供主程序调用。

```
;子程序名：stdin.asm
;功能：完成从键盘输入一位十进制数
;入口参数：等待键盘输入
;出口参数：al中存放输入的数值
;public stdin

stdin   proc
again:  mov     ah, 1
        int     21h
        cmp     al, 30h
          jlagain
        cmp     al, 39h
        jg      again
        and     al, 0fh
        ret
stdin   endp
```

```
; DISPBXD.asm
DISPBXD PROC
        mov     ch, 4           ;循环次数
rotate: mov     cl, 4           ;移位次数
        rol     bx, cl          ;bx循环左移4位
        mov     al, bl          ;移位后的低8位送al
        and     al, 0fh ;取al的低四位
        add     al, 30h         ;0-9转ascii码，加30h
        cmp     al, 3ah         ;比较是否是a-f
        jl      printit         ;如果小于3ah，直接输出
        add     al, 7h          ;反之说明是a-f，则加7
printit: mov    dl, al
        mov     ah, 2
        int     21h
        dec     ch              ;循环次数减1
        jnz     rotate          ;若zf不为0则循环
        ret
DISPBXD ENDP

FACT    PROC
        AND     BX,BX
        JZ      FACT1
        PUSH    BX
        DEC     BX
        CALL    FACT
        POP     BX
        MUL     BX
        RET
FACT1:  MOV     AX,1
        RET
FACT    ENDP
```

子程序的连接

例：从键盘输入一个数**n**（**(0≤n≤8)**，求**n!**，并用十六进制输出结果。前面几个例子的子程序编写成子程序模块，供主程序调用。

masm main.asm

使用的子程序要安排在代码段中，所以在此包含进去。

```
;主程序main.asm
;extrn     stdin:far,fact:far,dispbxd:far    不需要
Stack     segment
          dw        128       dup(0)
Stack     ends
Code      segment
          Assume    cs:code,ss:stack
Start:    mov       ax, stack
          mov       ss,  ax
          call      stdin
          mov       ah,0
          mov       bx,ax
          call      fact
          mov       bx,ax
          call      dispbxd
          mov       ah,4ch
          int       21h
          include stdin.asm
          include dispbxd.asm
code      ends
end       start
```

子程序的连接

(2) 在LINK连接时把各个模块连接在一起

把多个.OBJ文件连接成一个完整的.EXE文件。

要求:

- 各源程序要设置必要的段地址，至少要设置代码段（其它段根据需要安排）；
- 模块中要使用其它模块的标号时，用EXTRN语句说明；
- 本模块中存在可被其它模快引用的标号时，用PUBLIC语句声明。EXTRN和PUBLIC语句放在所有段的前面。

子程序的连接

声明共用的变量、标号、过程
各个模块间共用的变量、过程等要说明

PUBLIC 标识符 [,标识符...]　　;定义标识符的模块使用

EXTRN 标识符:类型 [,标识符:类型...]　　;调用标识符的模块使用

标识符是变量名、标号、过程名等。
类型是byte / word / dword（变量）或near / far（过程）。
在一个源程序中，public/extrn语句可以有多条。
各模块间的public/extrn伪指令要互相配对，并且指明的类型互相一致。

wenjiecoder@outlook.com

例：设模块A要引用模块B中的两个变量和标号：ARG和L_NAME，则在模块B中要用PUBLIC说明，而在模块A中要用EXTRN说明。

模块A:

```
        EXTRN   ARG: WORD
        EXTRN   L_NAME: FAR
A_SEGD  SEGMENT  PUBLIC
D_BYTE  DB       10  DUP(0)
A_SEGD  ENDS
A_SEGC  SEGMENT
        ASSUME
CS:A_SEGC,DS:A_SEGD
START:  MOV      AX,A_SEGD
        MOV      DS,AX
        MOV      BX,ARG
        …
        JMP      FAR PTR L_NAME
A_SEGC  ENDS
        END      START
```

模块B:

```
        PUBLIC  ARG, L_NAME
A_SEGD  SEGMENT  PUBLIC
ARG     DW       1234H,5678H
A_SEGD  ENDS
B_SEGC  SEGMENT
        ASSUME CS:B_SEGC,DS:A_SEGD
BEGIN:  MOV      AX,B_SEGD
        MOV      DS,AX
        …
L_NAME:          CMP      AH,10
        …
B_SEGC  ENDS
        END
```

子程序的连接

例：从键盘输入一个数**n（(0≤n≤8)**，求**n!**，并输出结果。前面几个例子的子程序编写成子程序模块，供主程序调用。

```
;主程序main.asm
extrn     stdin:far,fact:far,dispbxd:far

Stack     segment   stack
          dw        128       dup(0)
Stack     ends
Code      segment
          Assume    cs:code, ss:stack
Start:    call      stdin
          mov       ah,0
          mov       bx,ax
          call      fact
          mov       bx,ax
          call      dispbxd
          mov       ah,4ch
          int       21h
code      ends
end       start
```

1.各个模块分别汇编，生成目标文件stdin.obj、dispbxd.obj和main.obj。

2.然后一起连接：

link main.obj stdin.obj dispbxd.obj

生成可执行程序：main.exe

当子程序模块很多时，可以把它们统一管理起来，存入一个或多个子程序库中。库文件可以把它看成是子程序的集合。(MASM系统提供了库管理程序LIB.EXE，可以建立、组织和维护子程序库，使用的时候用include伪指令将该库包含进来。)

格式如下：

LIB　库文件名+子程序目标文件名

如：Lib mylib.lib stdin.obj+dispbxd.obj

子程序的连接

例：从键盘输入一个数n（(0≤n≤8)，写成子程序模块，供主程序调用。

```
;子程序名：stdin.asm
;功能：完成从键盘输入一位十进制数
;入口参数：等待键盘输入
;出口参数：al中存放输入的数值
public stdin

code segment
        assume cs:code

stdin   proc far
again:  mov   ah, 1
        int   21h
        cmp   al, 30h
        jl    again
        cmp   al, 39h
        jg    again
        and   al, 0fh
        ret
stdin   endp

code ends
```

```
; DISPBXD.asm
public      dispbxd, fact
code segment
   assume cs:code
DISPBXD PROC        FAR
            mov     ch, 4           ;循环次数
rotate:     mov     cl, 4           ;移位次数
            rol     bx, cl          ;bx循环左移4位
            mov     al, bl          ;移位后的低8位送al
            and     al, 0fh ;取al的低四位
            add     al, 30h         ;0-9转ascii码，加30h
            cmp     al, 3ah         ;比较是否是a-f
            jl      printit         ;如果小于3ah，说明是0-
9，直接输出
            add     al, 7h          ;反之说明是a-f，则加7
printit:mov dl, al
            mov     ah, 2
            int     21h
            dec     ch              ;循环次数减1
            jnz     rotate          ;若zf不为0则循环
            ret
DISPBXD     ENDP

FACT        PROC    FAR
            AND     BX,BX
            JZ      FACT1
            PUSH    BX
            DEC     BX
            CALL    FACT
            POP     BX
            MUL     BX
            RET
FACT1:      MOV     AX,1
            RET
FACT        ENDP
code ends
```

补充例题

子程序补充例题：

利用子程序方法将键盘输入的一组带符号十进制字数据存储到缓存中。

子程序从键盘输入一个有符号十进制数；子程序还包含将ASCII码转换为二进制数的过程。

输入时，负数用“－”引导，正数直接输入或用“＋”引导。

子程序用寄存器传递出口参数，主程序调用该子程序输入10个数据。

wenjiecoder@outlook.com

分析：

① 首先判断输入为正或负数，并用一个寄存器记录。

② 接着输入0~9数字（ASCII码），并减30H转换为二进制数。

③ 然后将前面输入的数值乘10，并与刚输入的数字相加得到新的数值。

④ 重复②、③步，直到输入一个非数字字符结束。

⑤ 负数进行求补，转换成补码；否则直接保存数值。

本例采用16位寄存器表达数据，所以只能输入+327677~-32768间的数值。

但该算法适合更大范围的数据。

wenjiecoder@outlook.com

补充例题

```
.data
    count= 10
    array   dw count dup(0)  ;预留数据存储空间
.code
.startup
    mov cx,count
    mov bx,offset array
again:  call read           ;调用子程序输入一个数据
    mov [bx],ax             ;将出口参数存放缓冲区
    inc bx
    inc bx
    call dpcrlf ;调用子程序，光标回车换行以便输
                入下一个数据
    loop again
.exit 0
```

```
;输入有符号10进制数的通用子程序
;出口参数：AX＝补码表示的二进制数值
;                CX=0表示正数,CX=-1表示负数
;说明：负数用“－”引导，正数用“＋”引导或直接输入
;数据范围是＋32767～－32768
read        proc
            push bx
;           push cx
            push dx
            xor bx, bx  ;BX保存结果
            xor cx, cx
            ;CX为正负标志，0为正，－1为负
            mov ah, 1  ;输入一个字符
            int 21h
            cmp al, '+'
            ;是“＋”，继续输入字符
            jz read1
```

补充例题


```
        cmp al, '-'
        ;是“-”，设置-1标志
        jnz read2
        ;非“+”和“-”，转read2
        mov cx, -1
read1: mov ah, 1
        ;继续输入字符
        int 21h
read2: cmp al,'0'
        ;不是0~9之间的字符，则输入数据结束
        jb read3
        cmp al, '9'
        ja read3
        sub al, 30h
        ;是0~9之间的字符，则转换为二进制数
        ;利用移位指令，实现数值乘10
        shl bx, 1
        mov dx, bx
        shl bx,1
        shl bx,1
        add bx,dx
        mov ah,0
        add bx,ax
        ;已输入数值乘10后，与新输入数值相加
        jmp read1          ;继续输入字符
read3: cmp cx,0
        jz read4
        neg bx             ;是负数，进行求补
read4: mov ax,bx           ;设置出口参数
        pop dx
        pop cx
        pop bx
        ret   ;子程序返回
read        endp
            ;使光标回车换行的子程序
dpcrlf      proc
            ...            ;省略
dpcrlf      endp
            end
```

第7讲作业：

Page 241：6.7

第8讲作业：

Page 244：6.15

wenjiecoder@outlook.com