第8章　子程序设计

为了使程序结构更加清晰，把程序需要完成的任务分解为若干个子任务，把每个子任务设计成一个相对独立的程序，称为子程序，也称为过程。

8.1 子程序结构

- 8.1.1 子程序调用指令
- 8.1.2 过程定义与过程结构
- 8.1.3 保护和恢复现场寄存器

8.1.1 子程序调用指令

- 子程序定义：在模块化程序设计中，经常把程序中某些具有独立功能的部分编写成独立的程序模块，称为子程序。
- 主程序通过CALL指令调用子程序。
- 子程序执行完毕后通过RET指令回到主程序。

（1）CALL 调用指令

- 格式: CALL DST
- 操作：首先把下一条指令的地址（返回地址）压入堆栈保存，再把子程序的入口地址置入IP（CS）寄存器，以便实现转移。

  对于段内调用，只是向堆栈保存IP寄存器的值。

  对于段间调用，是先向堆栈保存CS寄存器的值，再向堆栈保存IP寄存器的值。

（2）RET 返回指令

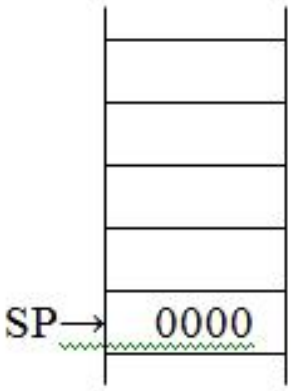
- 格式1: RET
- 格式2: RET EXP
- 操作：把堆栈里保存的返回地址送回IP（CS）寄存器，实现程序的返回。

  对于段内调用，弹出一个字到IP寄存器。

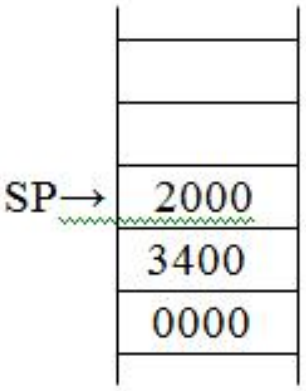
  对于段间调用，先弹出一个字到IP寄存器，再弹出一个字到CS寄存器。
- 例8.1

- 例8.1 代码段1中的主程序A调用代码段2中的子程序B，程序B调用代码段2中的子程序C，调用关系如图8-1所示，堆栈情况如图8-2所示。

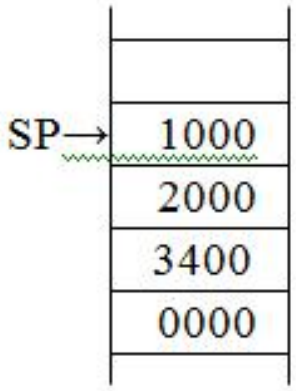
主程序A	程序B	程序C
…	…	…
…	…	…
CALL FAR PTR B	CALL NEAR PTR C	RET
(IP=2000H,CS=3400H)	(IP=1000H,CS=6200H)	
…	…	
…	…	
	RET	



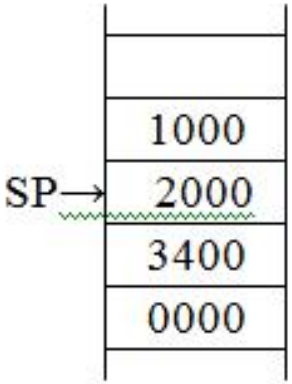
(1)A 调用 B 前



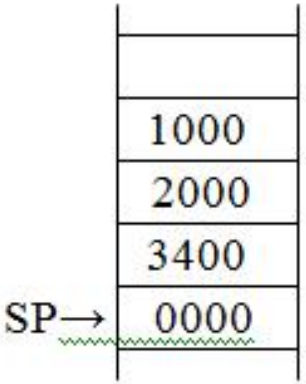
(2)A 调用 B 后



(3)B 调用 C 后



(4)C 返回 B 后



(5)B 返回 A

8.1.2 过程定义与过程结构

- 过程定义伪指令

```
Procedure_Name  PROC   Attribute
                ⋮
Procedure_Name  ENDP
```

- 如：

```
main  proc   far /near
            ⋮
main  endp
```

- 过程名是标识符，起到标号的作用，是子程序入口的符号地址。
- 过程的属性可以是FAR或NEAR类型。

对过程属性的确定原则是：

- NEAR为近，是段内调用。FAR类型为远，是跨段调用。
- 如调用程序和子程序在同一代码段，则使用NEAR属性；如调用程序和子程序不在同一代码段，则使用FAR属性。
- 主程序的过程定义属性应为FAR。

- 例8.2　调用程序和子程序在同一个代码段，调用程序和子程序并列。

```
main    proc    far
        …
        call    subr
        …
        ret
main    endp
subr    proc    near
        …
        ret
subr    endp
```

- 调用程序和子程序在同一个代码段，调用程序和子程序相互嵌套。

```
main    proc    far
        …
        call    subr
        …
        ret
subr    proc    near
        …
        ret
subr    endp
main    endp
```

- 例8.3 调用程序和子程序不在同一个代码段。

```
code1   segment
            …
main    proc    far
            …
            call    subr
            …
            ret
main    endp
code1   ends
        …
```

```
code2   segment
            …
            call    subr
            …
subr    proc    far
            …
            ret
subr    endp
code2   ends
```

8.1.3 保存和恢复现场寄存器

- 子程序调用 **CALL**：首先将返回地址压栈，然后把子程序的入口地址送入**IP/CS**寄存器。
- 子程序返回 **RET**：将堆栈里保存的返回地址送回**IP/CS**寄存器。
- 在子程序中对主程序的现场实施保护和恢复

  在进入子程序后，对将要使用的寄存器，先保存这些寄存器的值，在子程序退出前恢复这些寄存器的值。

- 保存和恢复寄存器

```
SUBRT   PROC   NEAR
        PUSH   AX
        PUSH   BX
        PUSH   CX
        PUSH   DX
          ⋮
        POP    DX
        POP    CX
        POP    BX
        POP    AX
        RET
SUBRT   ENDP
```

8.2 子程序的参数传递

- 入口参数(调用参数)：主程序传递给子程序。
- 出口参数(返回参数)：子程序返回给主程序。
- 传递的参数：值传递和地址传递。

8.2.1 用寄存器传递参数

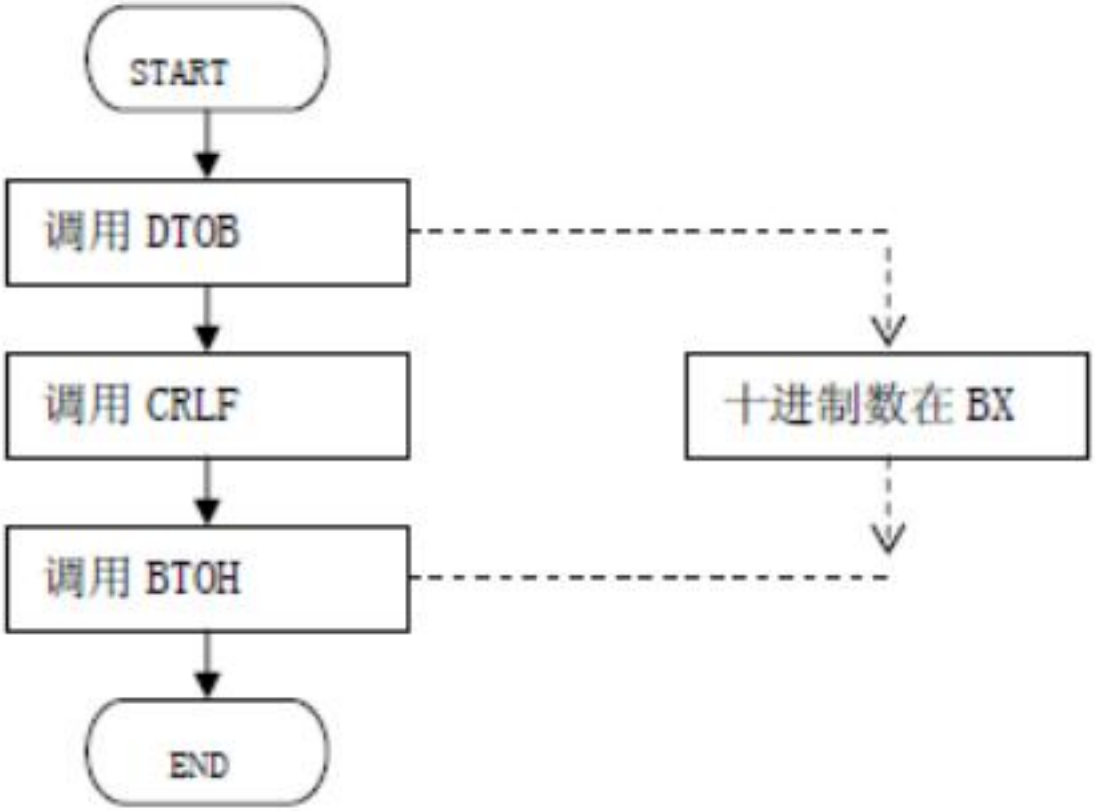
用寄存器传递参数就是约定某些寄存器存放将要传递的参数。该方法简单，执行的速度也很快。但由于寄存器数量有限，不能用于传递很多的参数。

- 例8.4　从键盘输入一个十进制数（小于65536的正数），显示输出该数的十六进制形式。通过寄存器传送变量。

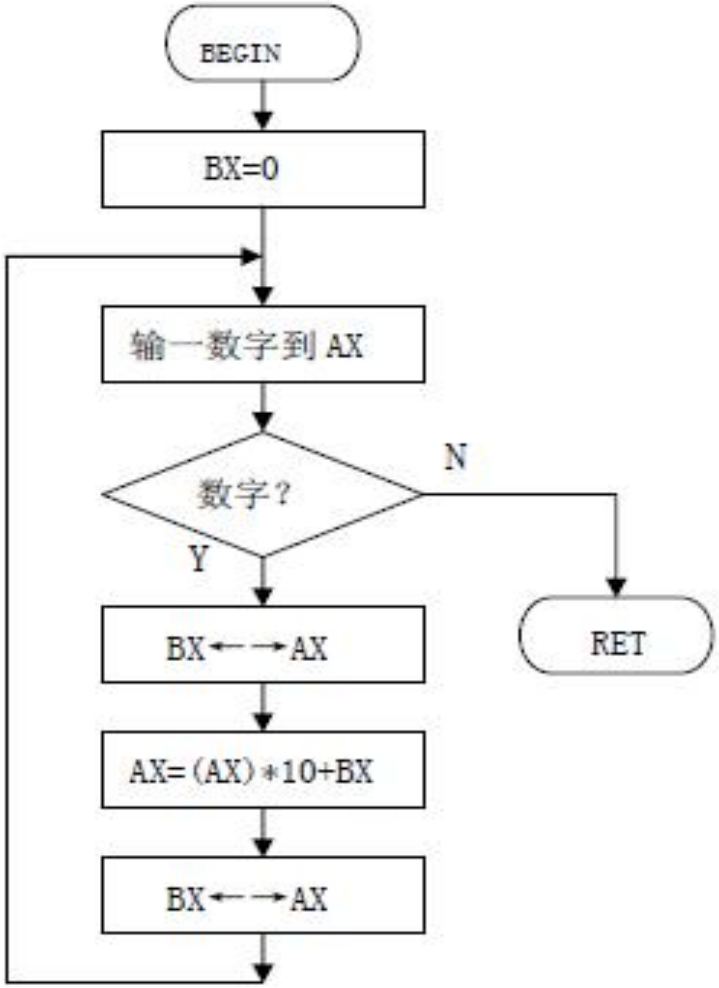
- 算法分析

  1. 输入的十进制数整合成二进制数；
  2. 整合的方法 A(n)=A(n-1)*10+B(n)；
  3. 二进制数转化为十六进制显示。

程序结构框图



DTOB子程序流程图



```
dtohex    segment
          assume  cs:dtohex
main      proc  far
          push  ds
          xor   ax, ax
          push  ax
          call  dtob  ；十进制数键盘输入整合为二进制
          call  crlf        ；输出回车换行
          call  btoh        ；二进制转为十六进制显示
          ret
main      endp
```

```
dtob    proc  near
        mov  bx, 0
input:  mov  ah, 1          ；键盘输入
        int  21h
        sub  al, 30h        ；把ascii码转变为数值
        jl  exit            ；如不是数则退出
        cmp  al, 9
        jg  exit            ；如不是数则退出
        cbw                 ；扩展为字
        xchg  ax,  bx       ；交换寄存器
        mov  cx, 10
        mul  cx             ；a(n)= a(n-1)×10
        xchg  ax, bx        ；交换寄存器
        add  bx,  ax        ；a(n)＝a(n)＋b(n)
        jmp  input
exit:   ret
dtob    endp
```

```
btoh    proc   near
        mov  ch, 4          ；准备输出4位十六进制数
shift:  mov  cl, 4          ；每次需移4位
        rol  bx, cl
        mov  al, bl
        and  al, 0fh        ；取最右4位
        add  al，30h        ；转为ascii
        cmp  al, 39h
        jle  dig            ；是0～9则转dig
        add  al, 7          ；是A～F
dig:    mov  dl，al         ；显示
        mov  ah，2
        int  21h
        dec  ch
        jnz  shift
        ret
btoh    endp
```

```
crlf      proc  near
          mov  dl, 0dh
          mov  ah,2
          int  21h
          mov  dl, 0ah
          mov  ah,2
          int  21h
          ret
crlf      endp
dtohex       ends
                end  main
```

8.2.2 用变量传递参数

- 参数较多时可以用约定的变量在过程间传递参数。

- 例8.5 键盘输入字符串到缓冲区后，对缓冲区内容降序排序并输出。

```
data    segment
        buff   db   16          ；缓冲区大小
        numb   db   ?           ；输入的字节数
        arry   db   16 dup(?)  ；缓冲区内容
data    ends
code    segment
        assume  cs:code,ds:data
main    proc  far
        push ds
        sub ax,ax
        push ax
        mov   ax,data
        mov   ds,ax
        call order
        ret
main    endp
```

```
order     proc  near
          lea dx,buff              ；输入缓冲区
          mov   ah,10
          int   21h
          mov  cl,numb             ；实际输入个数
          mov   ch,0
          mov   di,cx
lp1:      mov   cx,di
          mov   bx,0               ；下标
lp2:      mov   al,arry[bx]
          cmp   al,arry[bx+1]
          jge   cont
          xchg  al,arry[bx+1]
          mov  arry[bx],al
```

```
cont:     inc  bx
          loop lp2
          dec  di
          jnz   lp1
           call output
          ret
order     endp
```

```
;  ------------------
output      proc  near
mov   bl,numb        ；后面插入$以便显示
mov   bh,0
mov   byte ptr[arry+bx],'$'
mov   dx, offset arry
mov   ah,9
int   21h
 ret
output     endp
code       ends
end    main
```

8.2.3 用地址表传递参数的通用子程序

- 在主程序中建立一个地址表，把要传递的参数地址放在地址表中，然后把地址表的首地址放入寄存器，子程序通过寄存器间接寻址方式从地址表中取得所需参数，可以设计通用子程序处理其他类似字符串排序问题。

- 例8.6 采用通过地址表传递参数地址的方法，键盘输入缓冲区并对其内容排序和输出。

```
data   segment
       buff   db   16
       numb   db   ?
       arry   db   16 dup(?)
       table  dw   3 dup(?)        ；地址表
data     ends
code  segment
       assume cs:code,ds:data
main           proc  far
               push ds
               sub ax,ax
               push ax
```

```
        mov   ax,data
        mov   ds,ax
        mov   table,offset buff
        mov   table+2,offset numb
        mov   table+4,offset arry
        mov   bx, offset table   ；地址表首地址送bx,si
        mov   si,bx
        call order
        ret
main   endp
```

```
order    proc  near
         mov   dx,[bx]
         mov   ah,10
         int   21h
         mov   di,[bx+2]
         mov   cl,[di]            ；实际输入个数送cx
         mov   ch,0
         mov   di,cx
lp1：    mov   cx,di
         mov   bx,[si]            ；地址表首地址送bx
         add bx,2                 ；bx指向缓冲区
lp2:     mov   al,[bx]
         cmp   al,[bx+1]
         jge   cont
         xchg  al,[bx+1]
         mov  [bx],al
```

```
cont:     inc  bx
          loop lp2
          dec  di
          jnz   lp1
           call output
          ret
order     endp
```

```
;  ------------------
output      proc  near
mov   di,[si+2]          ；后面插入$以便显示
mov   bl,[di]
mov   bh,0
mov   di,[si+4]
mov   byte ptr[di+bx],'$'
mov   dx,di
mov   ah,9
int   21h
 ret
output     endp
code       ends
end    main
```

8.2.4 用堆栈传递参数的通用子程序

- 用堆栈传递参数地址，可以设计通用子程序。
- 例8.7 键盘输入缓冲区内容排序并输出，用堆栈传递参数地址。

```
data    segment
        dw  50 dup(?)          ；堆栈50个字
        tos     label  word    ；栈顶地址tos
        buff   db   16
        numb   db ?
        arry   db   16 dup(?)
data    ends
code   segment
       assume  cs:code，ds:data，ss:data
main   proc   far
        ；设置ss和sp
        mov    ax，data
        mov    ss，ax
        lea    sp，tos
        ；ds和0压入堆栈，以便返回dos
        push   ds
```

```
        xor   ax，ax
        push  ax
        mov   ax，data
        mov   ds，ax
        ；参数地址压入堆栈
        lea   bx, buff
        push  bx              ；buff的地址压入堆栈
        lea   bx, numb
        push  bx              ；numb的地址压入堆栈
        lea   bx, arry
        push  bx              ；arry的地址压入堆栈
        call  order
        ret
main    endp
```

```
order    proc   near
         mov   bp,sp
         mov   dx,[bp+6]          ；buff地址送dx
         mov   ah,10
         int   21h
         mov    di, [bp+4]        ；取numb的地址
         mov    cl,[di]
         mov    ch,0              ；numb送cx
         mov   di,cx
lp1:     mov   cx,di
         mov    bx，[bp+2]        ；arry的地址送bx
lp2:    mov   al,[bx]
         cmp   al,[bx+1]
         jge   cont
         xchg  al,[bx+1]
         mov  [bx],al
```

```
cont:    inc  bx
         loop lp2
         dec  di
         jnz   lp1
          call  output
          ret   6                ；修改sp指针并返回
order    endp
```

```
;  ------------------
output    proc  near
mov   di,[bp+4]          ；后面插入$以便显示
mov   bl,[di]
mov   bh,0
mov   di,[bp+2]
mov   byte ptr[di+bx],'$'
mov   dx, di
mov   ah,9
int   21h
 ret
output   endp
code      ends
 end    main
```

8.2.5 用结构变量传递参数的通用子程序

- 结构就是把逻辑上互相关联的一组数据以某种形式组合在一起。
- 在程序中，若要多次使用相同的一组数据格式，那么我们就可以把这一组数据格式定义为一个结构数据。

- 结构类型的定义：结构名　**STRUC**

  ……

  结构名　**ENDS**
- **STRUC**伪指令只是定义了一种结构模式，还没有生成结构变量。
- 用结构预置语句生成结构变量并赋值。

  结构预置语句格式：

  变量　结构名 <各字段赋值>

- 对结构字段初值的修改，并非所有字段的初值都可以修改，只有简单结构字段和字符串字段初值才可以修改。简单结构字段是指由伪指令DB、DW或DD定义的单项变量。
- 多项变量的结构字段初值不能修改。例如下面就是多项的结构字段：

```
DW   10  DUP(?)
DB    12H, 34H
DB    ‘ABCD’,’1234’
```

- 例8.8 定义一个名为STUDENT的结构类型。

```
STUDENT  STRUC
    ID        DB    'AAAAAAAA'
    NAME   DB    3 DUP(0)
    JF1       DW    22H
    JF2       DW    ?
    JF3       DW    ?
    JF4       DW    ?
STUDENT   ENDS
```

- 例8.9　结构变量预置语句(定义结构变量)

```
STD1    STUDENT   <'A2031456',,,33H>
STD2    STUDENT   <  >
STDSS   STUDENT  100  DUP(<  >)
```

- 例8.10　结构变量的访问

```
MOV  SI,1
LEA   BX,STD1
MOV  AL, STD1.NAME[SI]
; 变量STD1的字段NAME的第2项送AL
MOV  AL, [BX]. NAME[SI]
; 变量STD1的字段NAME的第2项送AL
MOV  DL, STDSS+3*19.NAME[SI]
;变量STDSS第4条记录的字段NAME的第2项送AL
```

- 例8.11 键盘输入缓冲区内容排序并输出，用堆栈传递参数地址，使用结构类型.

```
data    segment
        dw  50 dup(?)              ；堆栈50个字
        tos   label  word          ；栈顶地址tos
        buff     db   16
        numb     db   ?
        arry     dw   16 dup(?)
data    ends
code    segment
        assume  cs:code,ds:data,ss:data
main    proc   far
        ；设置ss和sp
        mov    ax,data
        mov    ss,ax
        lea    sp,tos
        ；ds和0压入堆栈，以便返回dos
        push   ds
```

```
        xor    ax,ax
        push   ax
        mov    ax,data
        mov    ds,ax
        ；参数地址压入堆栈
        lea    bx, buff
        push   bx                ；buff的地址压入堆栈
        lea    bx, numb
        push   bx                ；numb的地址压入堆栈
        lea    bx, arry
        push   bx                ；arry的地址压入堆栈
        call   order
        ret
main    endp
```

```
order   proc   near
            par    struc
            pip    dw  ?
            p3     dw  ?
            p2     dw  ?
            p1     dw  ?
            par    ends
            mov    bp,sp
            mov    dx,[bp].p1        ；buff的地址送dx
            mov    ah, 10
            int    21h
            mov    di, [bp].p2       ；取numb的地址
            mov    cl,[di            ；numb送cx
            mov    ch,0
            mov   di,cx
    lp1:    mov   cx,di
            mov   bx,[bp].p3         ；arry地址送bx
            lp2:     mov   al,[bx]
            cmp   al,[bx+1]
            jge   cont
            xchg  al,[bx+1]
```

```
            mov  [bx],al
            cont:    inc  bx
            loop lp2
            dec  di
            jnz   lp1
            call output
            ret    6                ；修改sp指针并返回
order       endp
code        ends
            end    main
```

```
;  ------------------
output    proc  near
mov   di,[bp].p2
mov   bl,[di]
mov   bh,0
mov   di,[bp].p3
mov   byte ptr[di+bx],'$'
mov   dx, di
mov   ah,9
int   21h
 ret
output     endp
code       ends
           end    main
```

8.3 多模块程序设计

多模块程序设计中，各个模块独立编写，分别汇编，便于程序的编写，调试和维护。

8.3.1 多模块之间的参数传递

- 局部符号和外部符号

  在本模块中定义，又只在本模块中引用的符号叫局部符号。

  在本模块中定义，而在另一模块中引用的符号叫外部符号。

- 对于外部符号，编程时需要明确说明。

- **EXTRN　符号名: 类型[，…]**

  在另一个模块中定义而要在本模块中使用的符号必须使用**EXTRN**伪操作说明。符号名为变量时，则**type**应为**BETY**，**WORD**，**DWORD**等；如符号名为标号或过程名，则**type**应为**NEAR**或**FAR**。
- **PUBLIC　符号名1，符号名2，……**

  在一个模块中定义的符号（包括变量、标号、过程名等）在提供给其他模块使用时，必须要用**PUBLIC**定义。

- 例8.12 主程序键盘输入缓冲区，子程序对缓冲区内容排序并输出，采用独立模块。

```
;  812main.asm
public  buff,numb,arry
extrn   order:far
data    segment
          buff   db   16
          numb   db   ?
          arry   db   16 dup(?)
data    ends
code    segment
assume  cs:code,ds:data
main proc far
          push ds
```

```
        sub  ax,ax
        push ax
        mov  ax,data
        mov  ds,ax
        lea  dx,buff
        mov  ah,10
        int  21h
        call order
        ret
main    endp
code    ends
        end main
```

```
; 812sub.asm
public  order
extrn   buff:byte,numb:byte,arry:byte
code    segment
assume  cs:code
order     proc  near
          mov   cl,numb
          mov   ch,0
          mov   di,cx
    lp1:  mov   cx,di
          mov   bx,0
    lp2:  mov   al,arry[bx]
          cmp   al,arry[bx+1]
          jge   cont
          xchg  al,arry[bx+1]
```

```
        mov   arry[bx],al
    cont:inc   bx
        loop  lp2
        dec   di
        jnz   lp1
        mov   bl,numb
        mov   bh,0
        mov   byte ptr[arry+bx],'$'
        mov   dx, offset arry
        mov   ah,9
        int 21h
        ret
order   endp
code    ends
        end
```

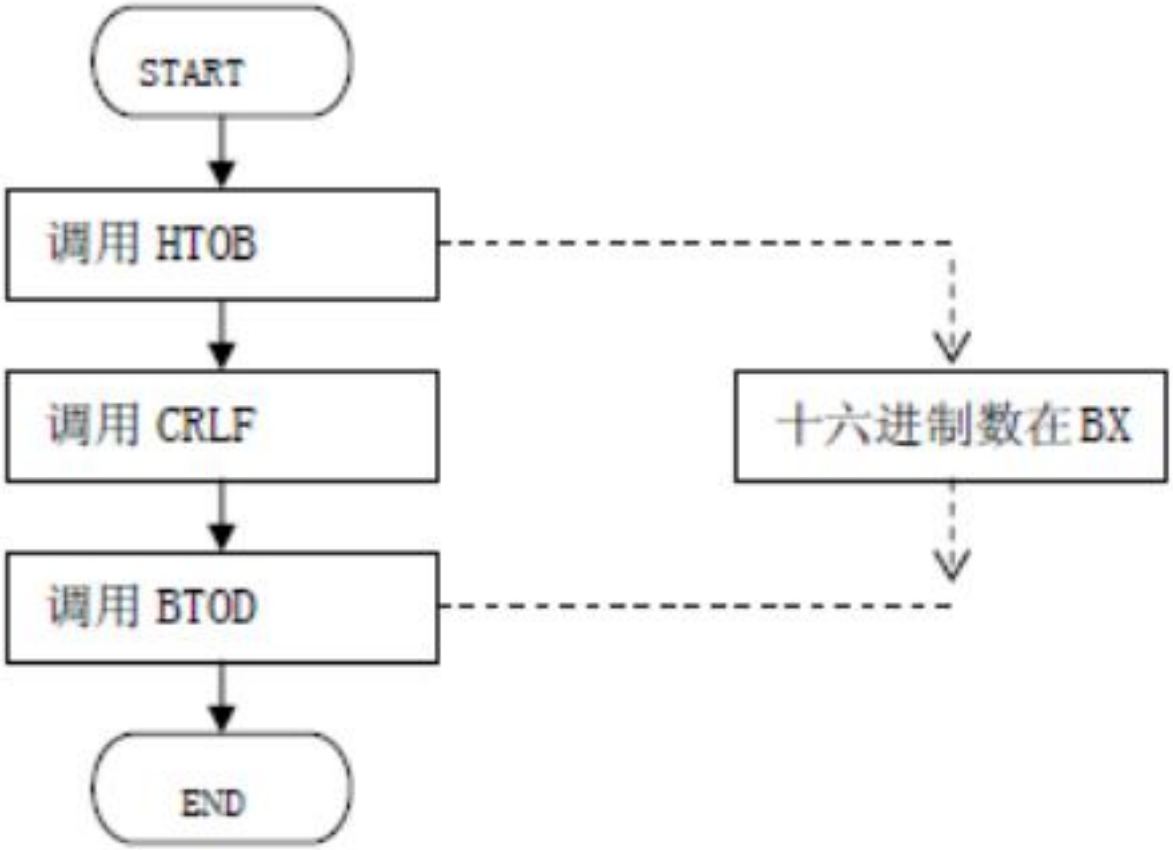
- 各模块先分别汇编，然后再连接:
  Link 812main+812sub

- **Link** 的次序影响结果，主模块在前面。

8.3.2 显示十进制数的通用模块

- 例8.13　从键盘输入一个十六进制数（不超过四位），显示输出该数的十进制形式。

- 算法分析：
  1. 输入的十六进制数转化为二进制数；
  2. 把二进制数用十进制显示；
  3. 辗转相除法。


START
调用 HTOB
调用 CRLF
调用 BTOD
END
十六进制数在BX

主程序文件HTODPRO.ASM

```
extrn   htob:far,crlf:far,btod:far
code    segment
    assume  cs:code
main    proc  far
        push  ds
        xor   ax，  ax
        push  ax
        call  htob
        call  crlf
        call  btod
        ret
main    endp
code    ends
        end   main
```

子程序文件htobpro.asm

```
public  htob
code1   segment
        assume  cs:code1
htob    proc  far
start:   mov   bx，0          ；初始化
         mov   ch，4
         mov   cl，4
inchr：  mov   ah，1          ；键盘输入
         int   21h
         cmp   al，30h
         jl    exit            ；非法输入
         cmp   al，39h
         jle   dig             ；输入是数字0～9
```

```
        cmp   al，41h
        jl    exit          ；非法输入
        cmp   al，46h
        jg    exit          ；非法输入
        sub   al，37h       ；输入是大写A～F
        jmp   ls4
dig：   sub   al，30h
ls4：   shl   bx，cl
        add   bl，al
        dec   ch
        jnz   inchr
exit:   ret
htob    endp
code1   ends
        end
```

子程序文件btodpro.asm

```
public  btod
code2   segment
          assume  cs:code2
btod    proc  far
          mov   cx，10000
          call  ddiv
          mov   cx, 1000
          call  ddiv
          mov   cx, 100
          call  ddiv
          mov   cx, 10
          call  ddiv
          mov   cx, 1
          call  ddiv
          ret
btod      endp
```

```
ddiv    proc  near
    mov   ax，bx
    mov   dx, 0
    div   cx
    mov   bx, dx
    mov   dl, al
    add   dl, 30h
    mov   ah, 2
    int   21h
    ret
ddiv      endp
code2   ends
          end
```

子程序文件crlfpro.asm

```
public  crlf
code3   segment
        assume  cs:code3
crlf    proc  far
        mov   dl，0ah
        mov   ah, 2
        int   21h
        mov   dl, 0dh
        mov   ah, 2
        int   21h
        ret
crlf     endp
code3   ends
         end
```