第6章　子程序结构

6.1 子程序的设计方法

- 6.1.1 过程定义伪操作
- 6.1.2 子程序调用和返回指令(CALL RET)
- 6.1.3 保存与恢复寄存器
- 6.1.4 子程序的参数传送
- 6.1.5 增强功能的过程定义伪操作

6.1.1 过程定义伪操作

- 格式：

```
procedure_name    PROC        attribute
                   …
                  RET
procedure_name    ENDP
```

- 说明：

(1)过程名为标识符，又是子程序入口的符号地址

(2)类型属性：

NEAR: 调用程序和过程在同一个代码段中(段内调用)

FAR: 调用程序和过程不在同一个代码段中(段间调用)

```
code  segment
main  proc  far
        ……
        call  subp
        ……
        ret
main  endp

subp  proc  near
        ……
        ret
subp  endp
code  ends
```

段内调用和返回

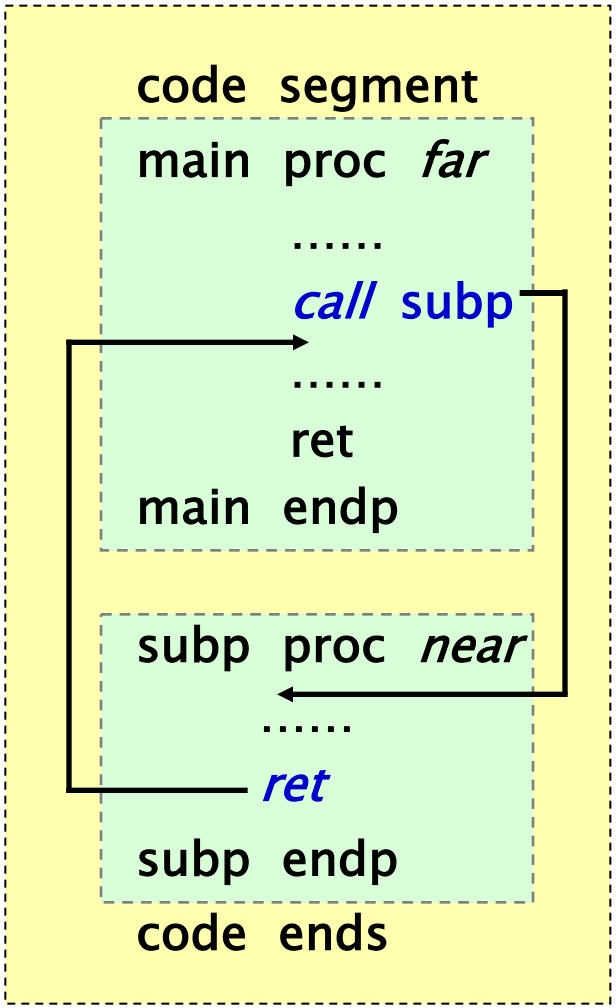
```
code1  segment
main   proc  far
         ……
         call  far ptr subp
         ……
         ret
main   endp
code1  ends

code2  segment
subp   proc  far
         ……
         ret
subp   endp
code2  ends
```

段间调用和返回

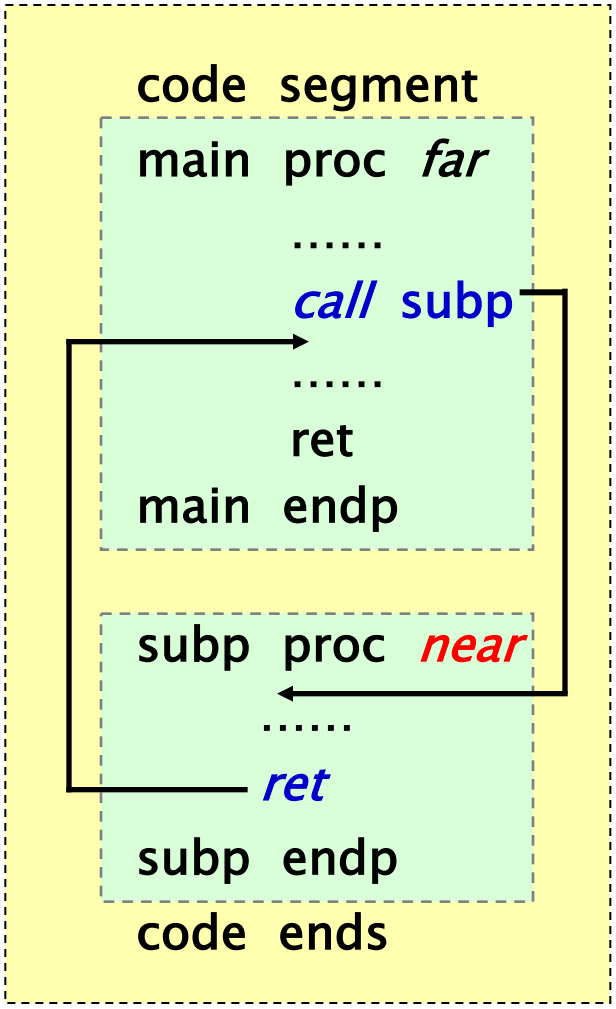
主过程**main**为**DOS**调用的一个子过程，应将**main**定义为**far**属性

6.1.2 子程序调用和返回指令(p98)

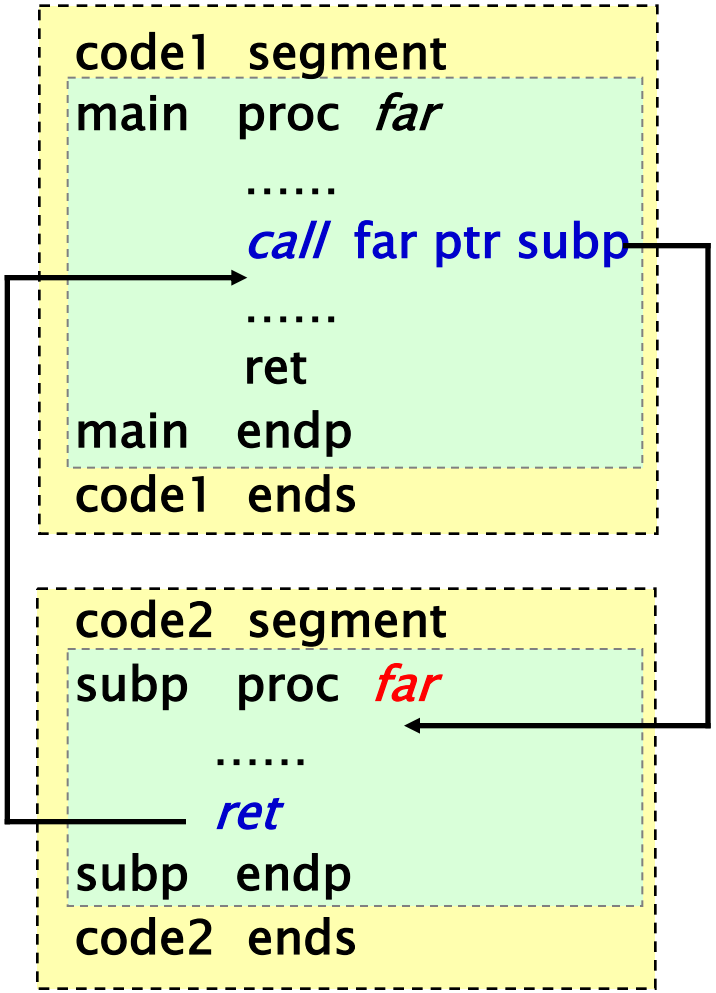
- 调用和返回：主程序调用（执行）子程序，在子程序执行完后又返回调用程序继续执行
- 调用(call):记录返回地址(进栈)
  更改指令地址
- 返回(ret): 地址出栈



分类



段内调用和返回



段间调用和返回

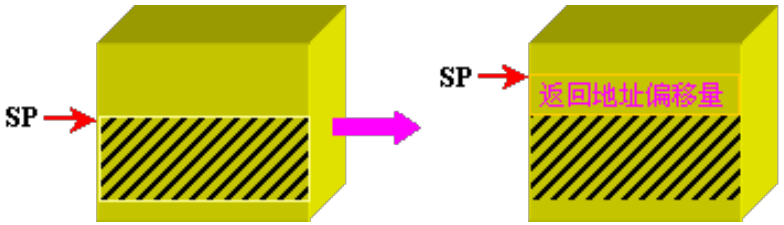
(1) 段内调用和返回

段内直接近调用

格式：**CALL**　过程名

操作：　**PUSH　　(IP)**　;返回地址进栈

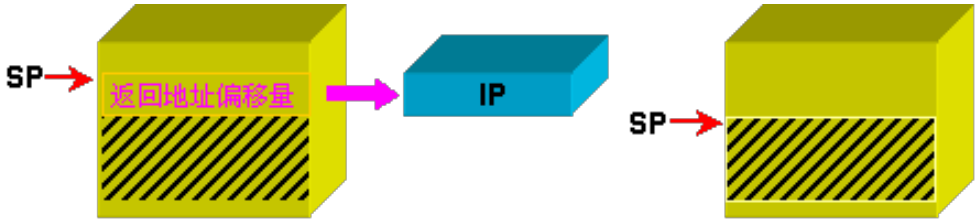
(IP)←(IP)+D16　;子程序入口地址



段内近返回

格式：**RET**

操作:　**(IP) ← POP()**



(2) 段间调用和返回

段间直接远调用

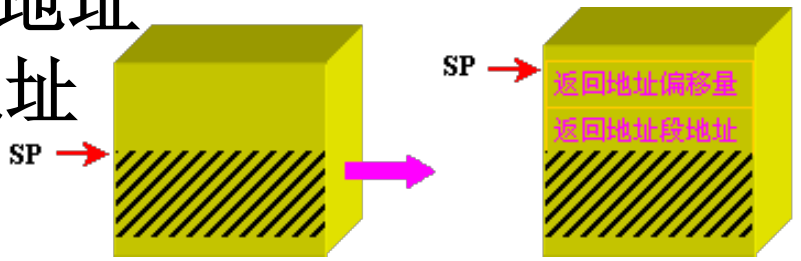
格式：　CALL　　FAR PTR 过程名

操作：　PUSH　　(CS)

PUSH　　(IP)

(IP) ← 过程名的偏移地址

(CS) ←过程名的段地址

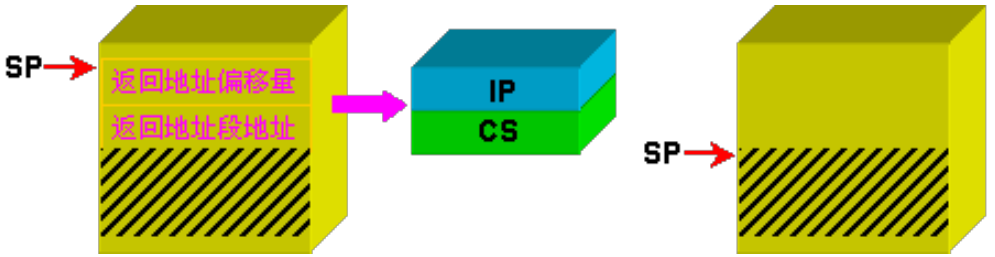


段间远返回

格式：RET

操作：　(IP) ← POP( )

(CS) ← POP( )



6.1.3 保存和恢复寄存器

- 调用程序和子程序所使用的寄存器常会发生冲突
- 在调用子程序前后都要使用的且子程序也要使用的寄存器内容应进行保存
- 一进入子程序，应该把子程序所需要使用的寄存器内容保存在堆栈中，在退出子程序前把寄存器内容恢复原状
- **CALL**使返回地址入栈，**RET**应使返回地址出栈，因此，子程序对堆栈的使用应特别小心

6.1.3 保存和恢复寄存器

```
subt   proc   near
    push      ax
    push      bx
    push      cx
    push      dx
    … …
    … …
    pop       dx
    pop       cx
    pop       bx
    pop       ax
    ret
subt   endp
```

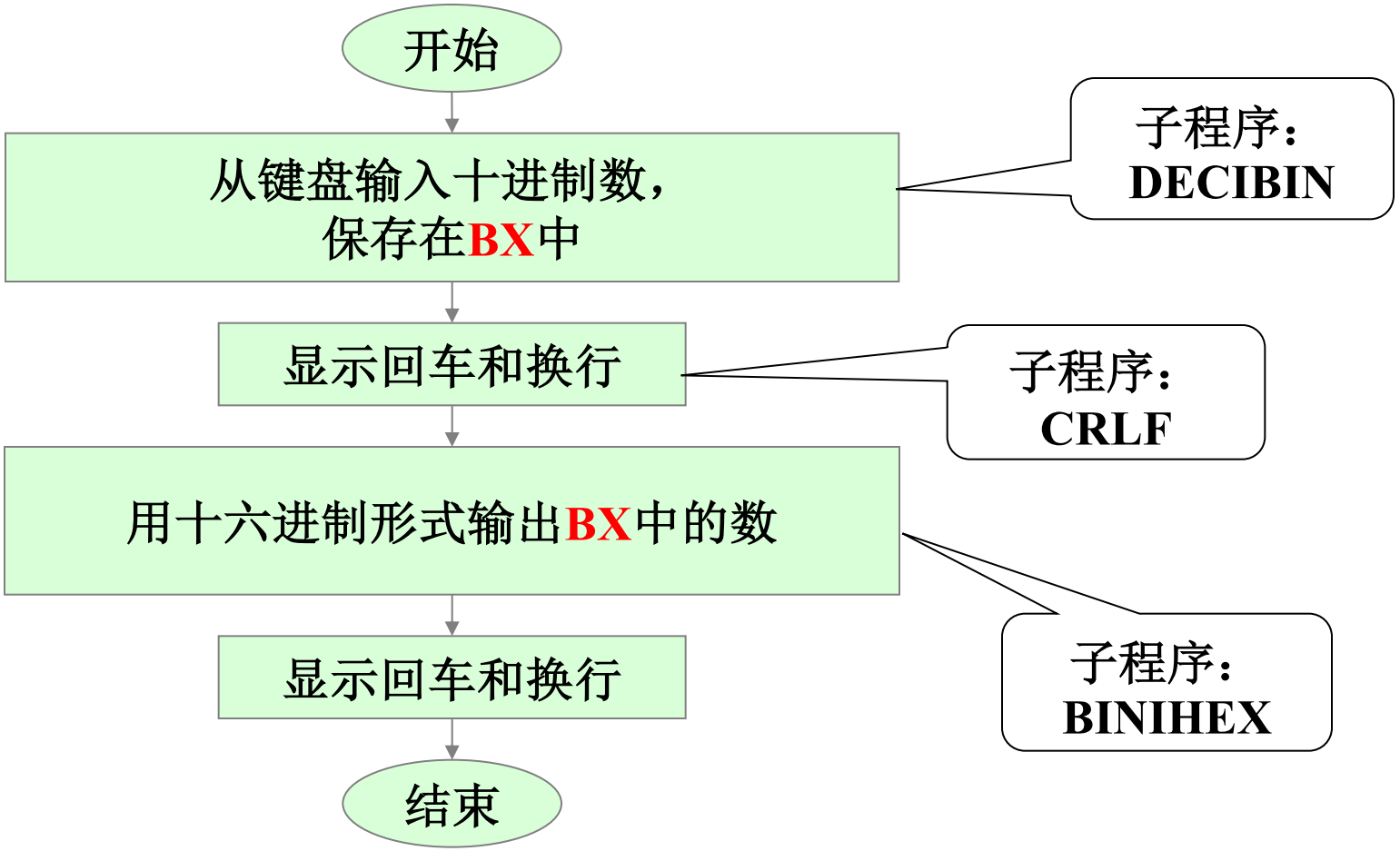
6.1.4 子程序的参数传送

- 1. 通过寄存器传送参数
- 2. 如调用程序和子程序在同一模块（文件）中，则子程序可直接访问模块中的变量
  通过存储器传送参数
- 3. 通过地址表传送参数地址
- 4. 通过堆栈传送参数或参数地址
- 5. 多个模块之间的参数传送

1. 通过寄存器传送参数

- 将调用程序和子程序都要使用的数据存放在寄存器中
- 最常用的一种方式，但参数很多时不能使用这种方法

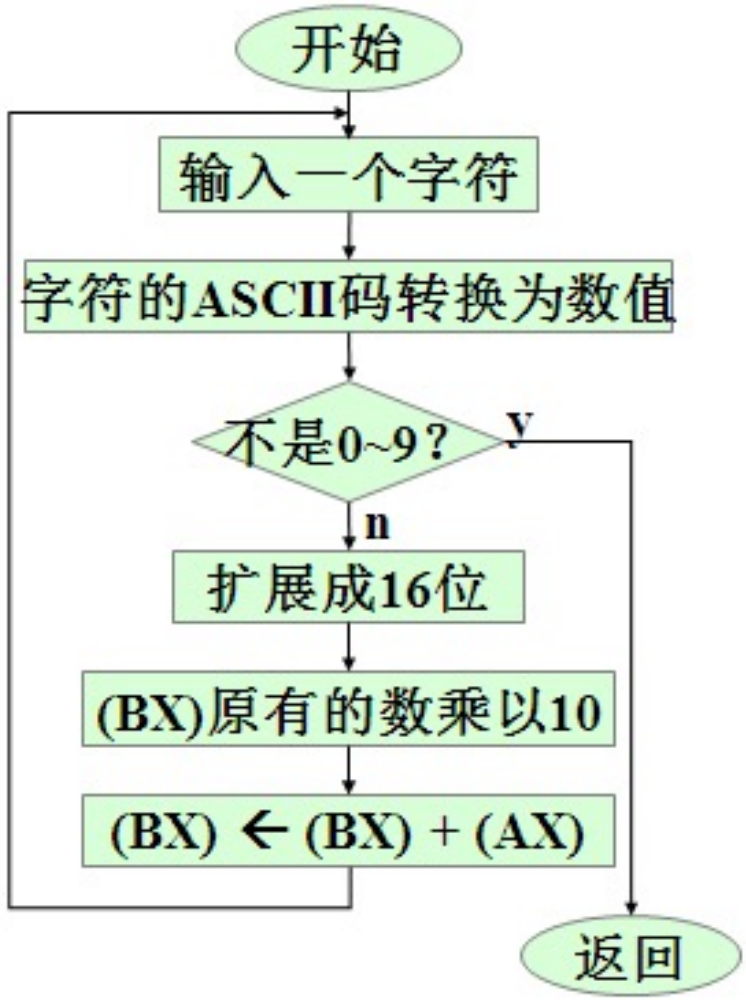
- 例6.3:十进制到十六进制转换程序。
  程序要求从键盘取得一个十进制数，然后把该数以十六进制形式在屏幕上显示出来。



程序框架

```
decihex     segment
   assume   cs:decihex
;-------------------------------------
main        proc        far
            …
            …
main        endp
;-------------------------------------
decibin     proc        near
            …
decibin     endp
;-------------------------------------
binihex     proc        near
            …
binihex     endp
;-------------------------------------
crlf        proc        near
            …
crlf        endp
;-------------------------------------
decihex     ends
        end   main
```

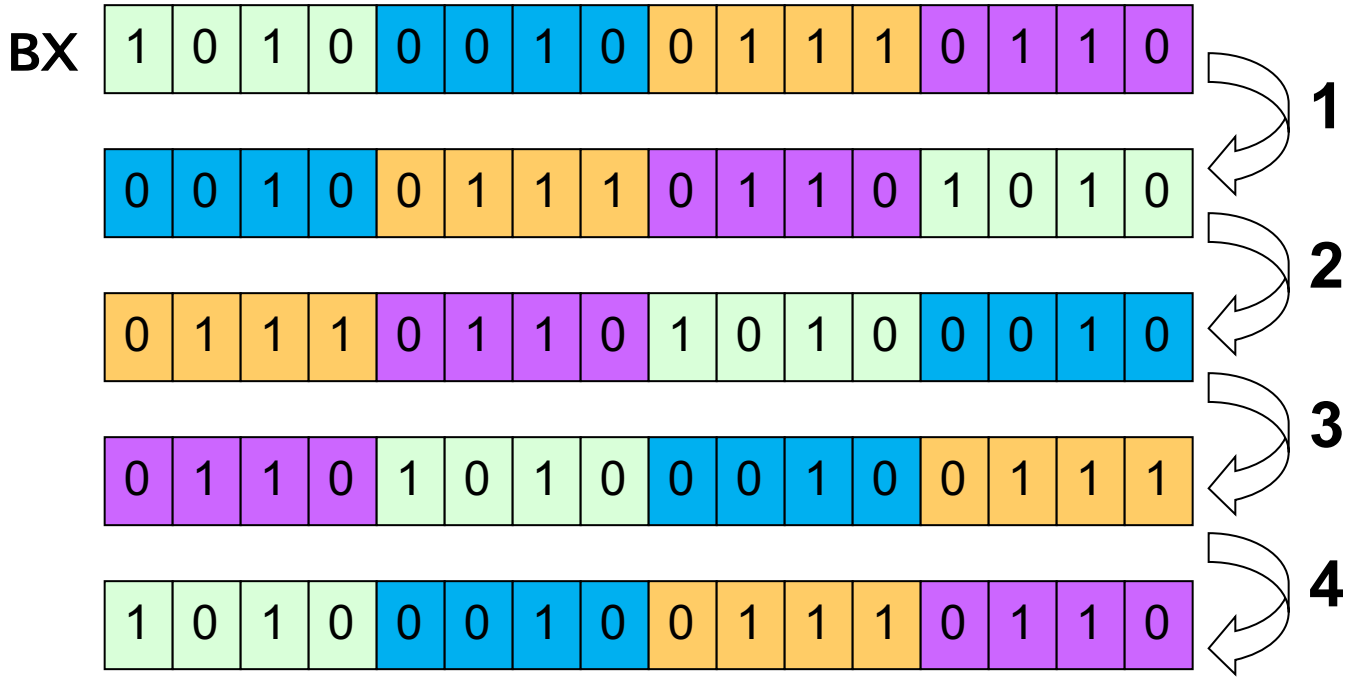
子程序
DECIBIN

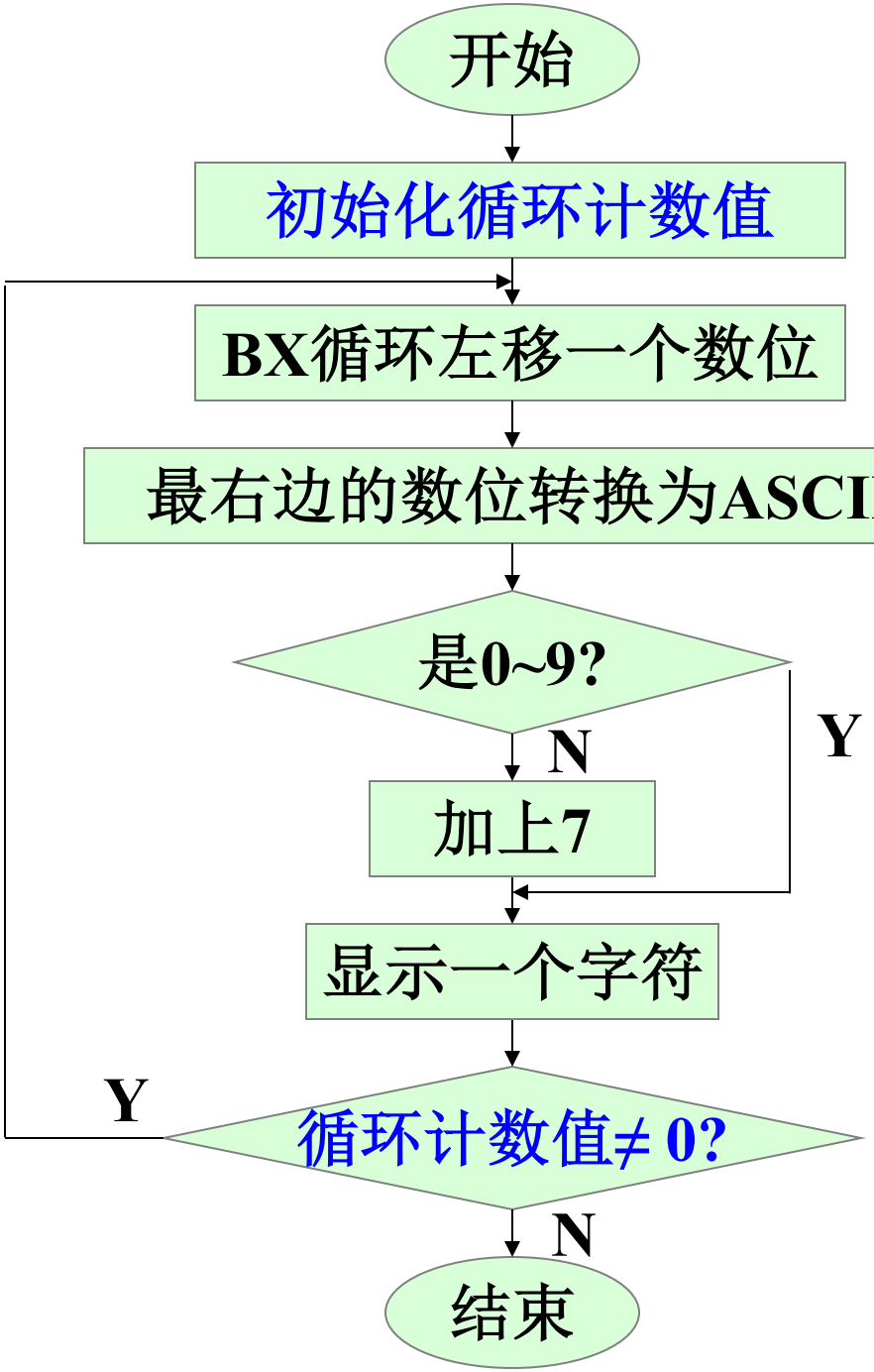



```
decibin  proc  near
                mov     bx,0
newchar:
                mov     ah,1
                int     21h
                sub     al,30h
                jl      exit
                cmp     al,9
                jg      exit
                cbw
                xchg    ax,bx
                mov     cx,10
                mul     cx
                xchg    ax,bx
                add     bx,ax
                jmp     newchar
exit:           ret
decibin  endp
```

子程序**BINIHEX**：把**BX**寄存器内的二进制数用十六进制的形式在屏幕上显示出来

如：0A276H






```
binihex  proc near
                mov     ch,4
rotate:         mov     cl,4
                rol     bx,cl
                mov     al,bl
                and     al,0fh
                add     al,30h
                cmp     al,3ah
                jl      printit
                add     al,7
printit:        mov     dl,al
                mov     ah,2
                int     21h
                dec     ch
                jnz     rotate
                ret
binihex  endp
```

子程序

crlf

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

主程序
MAIN

```
main  proc far
          push   ds
          sub    ax,ax
          push   ax
          call   decibin
          call   crlf
          call   binihex
          call   crlf
          ret
main  endp
```

2、如调用程序和子程序在同一模块（文件）中，则子程序可直接访问模块中的变量

例**6.4:** 主程序**main** 和子程序**progadd**在同一源文件中，要求用子程序**progadd**累加数组中的所有元素，并把和（不考虑溢出的可能性）送到指定的存储单元去

程序框架

```
;*****************************************
;
data            segment
   array        dw    1,2,3,4,5,6,7,8,9,10
   count        dw    10
   sum          dw    ?
data            ends
;*****************************************
;
code            segment
;-----------------------------------------
main            proc  far
        assume        cs:code,ds:data
start:
                …
main            endp
;-----------------------------------------
progadd         proc  near
                …
progadd         endp
;-----------------------------------------
code            ends
;*****************************************
;
        end   start
```

主程序MAIN

```
main  proc   far
      assume  cs:code,ds:data
start:
          push  ds
          sub   ax,ax
          push  ax
          mov   ax,data
          mov   ds,ax
          call  progadd
          ret
main  endp
```

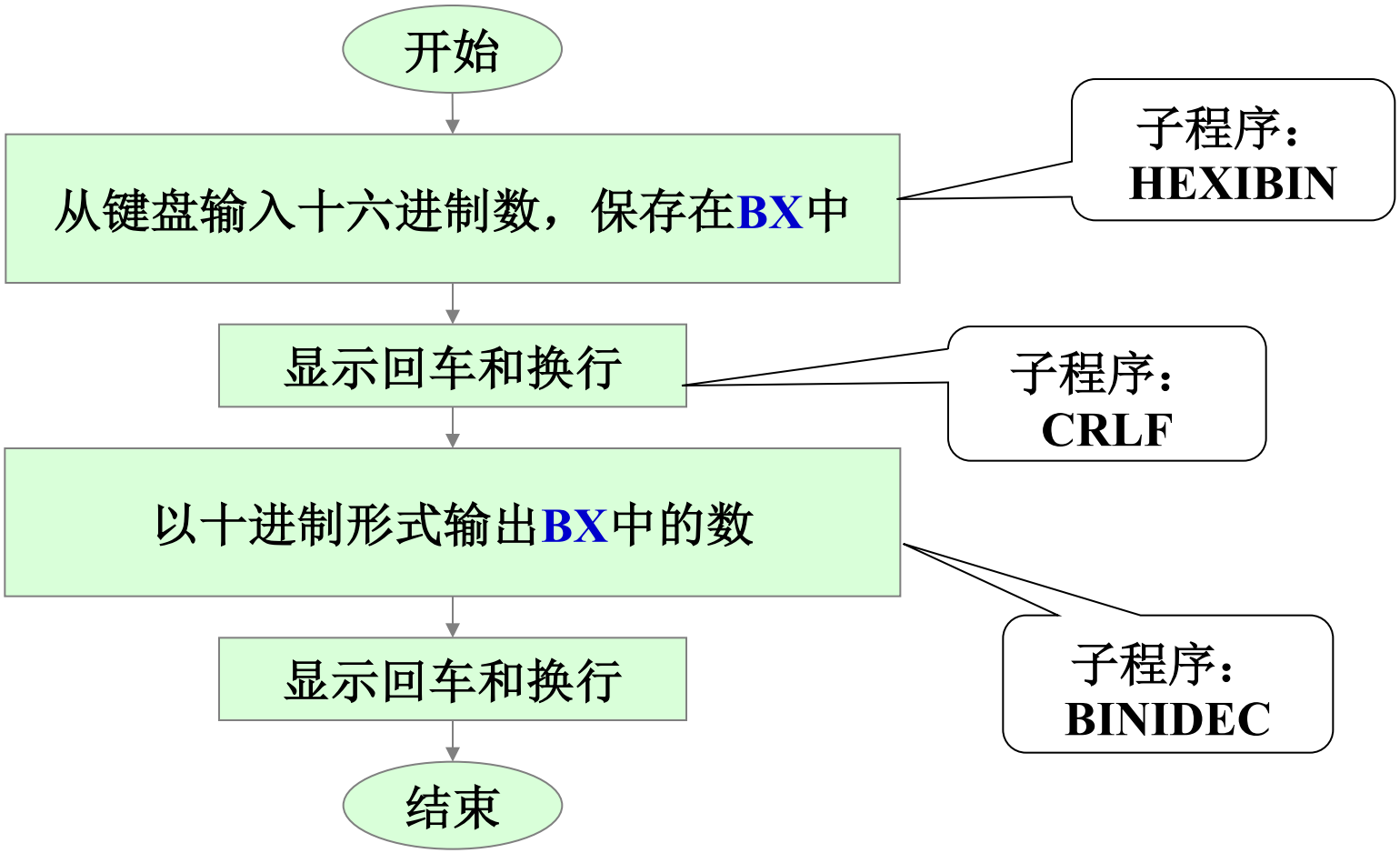
子程序progadd

```
progadd     proc   near
            push   ax              ;保存寄存器
            push   cx
            push   si
            lea    si,array
            mov    cx,count
            xor    ax,ax
next:       add    ax,[si]
            add    si,2
            loop   next
            mov    sum,ax
            pop    si              ;恢复寄存器
            pop    cx
            pop    ax
            ret
progadd     endp
```

6.3 子程序举例

例6.9：p225

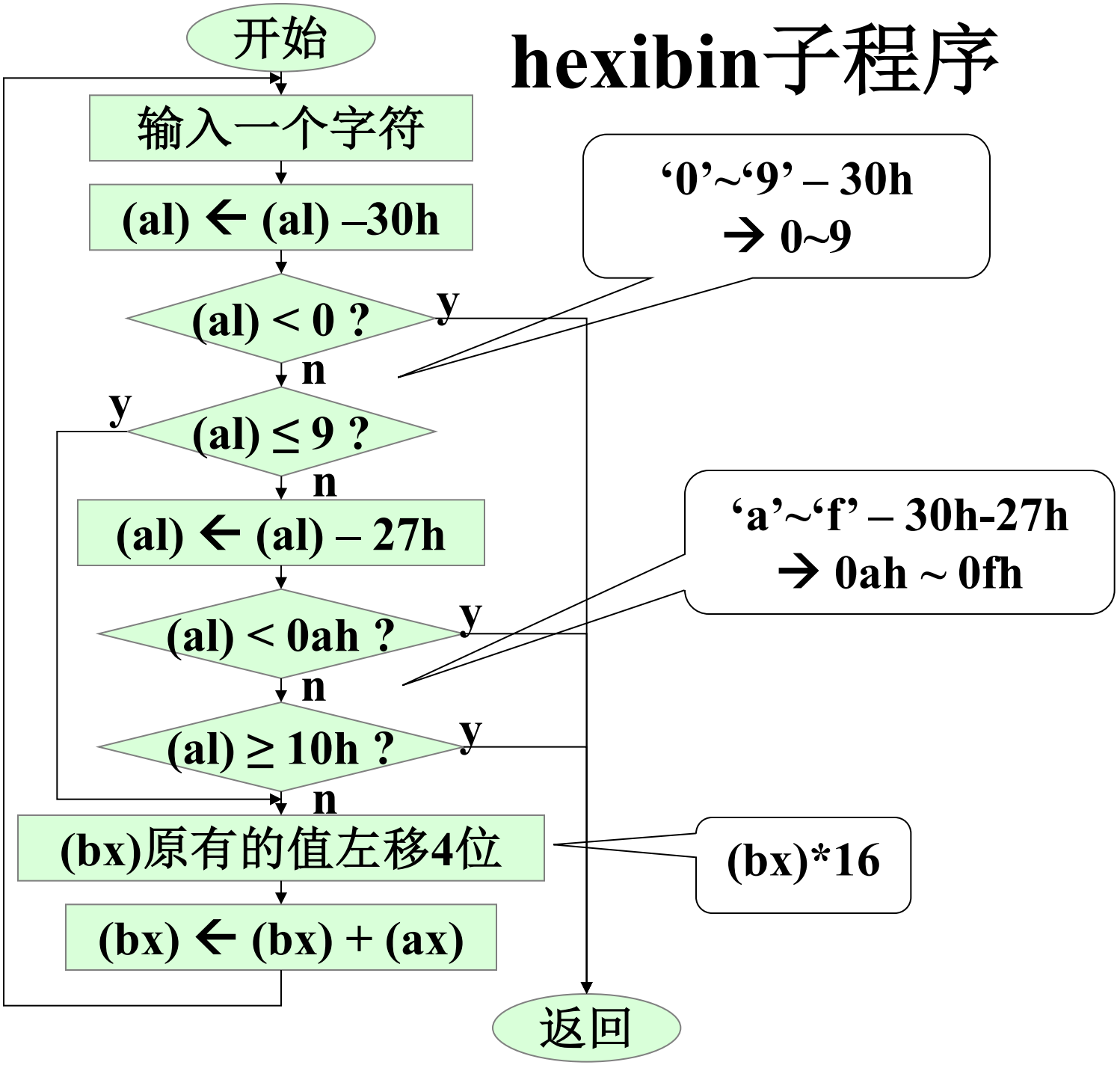
- 把从键盘输入的0 ~ FFFFH的十六进制正数转换为十进制数并在屏幕上显示出来



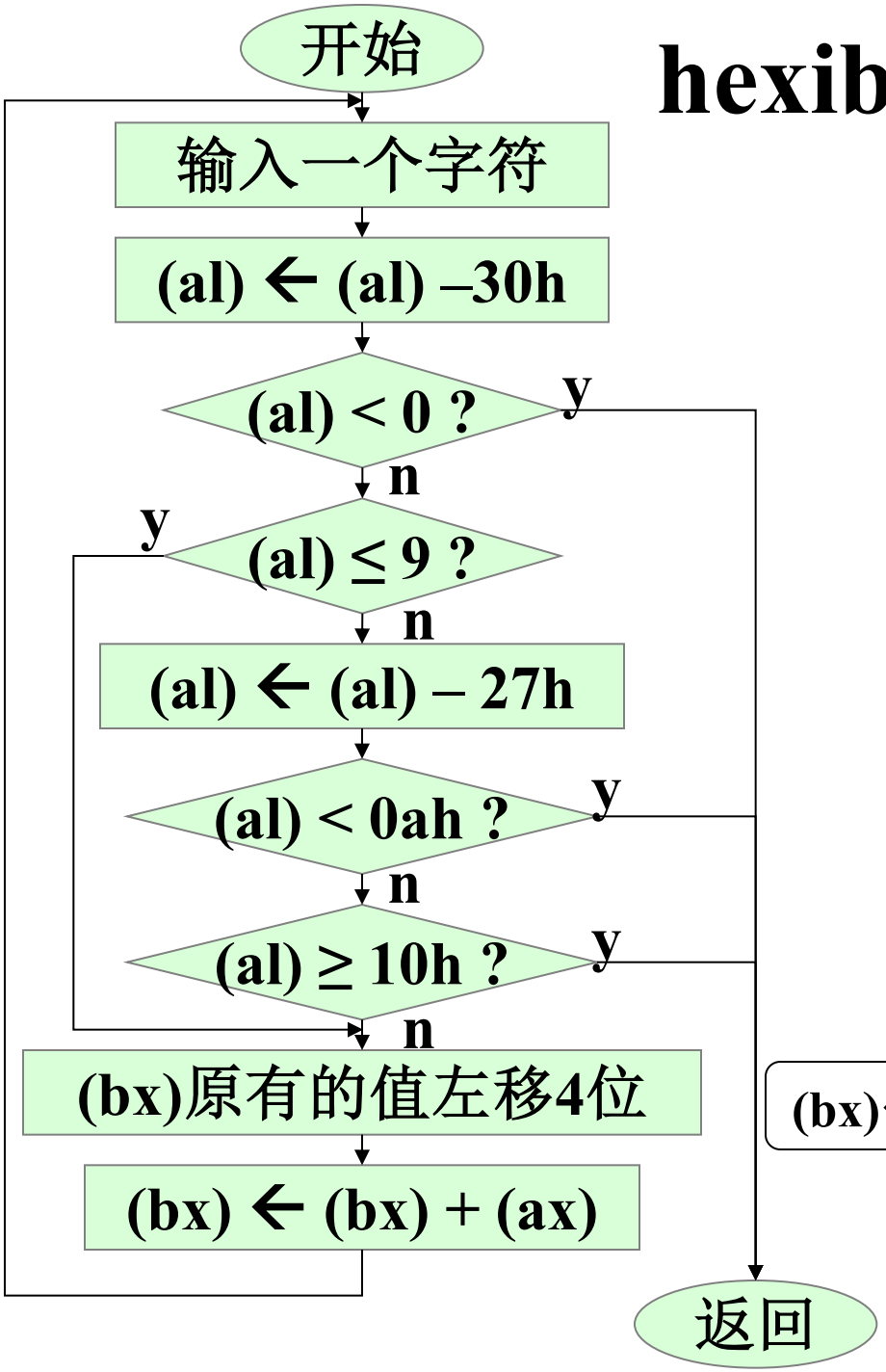
程序框架

```
display     equ     2h
key_in      equ     1h
doscall     equ     21h
;=====================================
hexidec     segment
;-------------------------------------
main            proc            far
        assume          cs:hexidec
start:
        …                       ;调用各子程序
main            endp
;-------------------------------------
hexibin         proc            near
        …                       ;输入十六进制
hexibin         endp
;-------------------------------------
binidec         proc            near
        …                       ;输出十进制
binidec         endp
;-------------------------------------
crlf            proc            near
        …                       ;显示回车换行
crlf            endp
;-------------------------------------
hexidec     ends
;=====================================
        end     start
```

hexibin子程序


开始
输入一个字符
(al) ← (al) –30h
(al) < 0 ?
y
n
(al) ≤ 9 ?
y
n
(al) ← (al) – 27h
(al) < 0ah ?
y
n
(al) ≥ 10h ?
y
n
(bx)原有的值左移4位
(bx) ← (bx) + (ax)
返回
‘0’~‘9’ – 30h
→ 0~9
‘a’~‘f’ – 30h-27h
→ 0ah ~ 0fh
(bx)*16

hexibin子程序



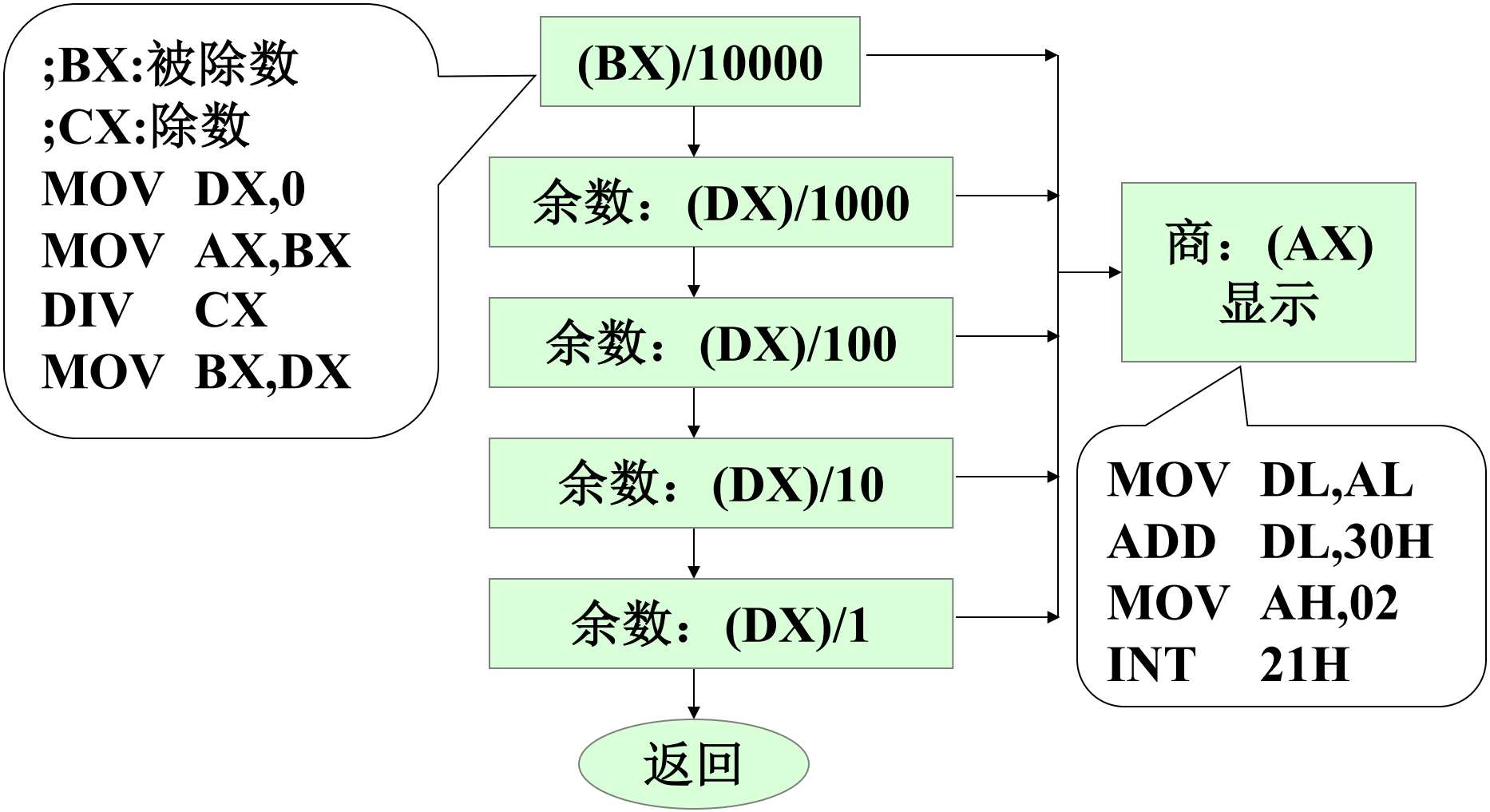

```
hexibin         proc    near
                mov     bx,0
newchar:        mov     ah,01h
                int     21h
                sub     al,30h
                jl      exit
                cmp     al,10
                jl      add_to
                sub     al,27h
                cmp     al,0ah
                jl      exit
                cmp     al,10h
                jge     exit
add_to:         mov     cl,4
                shl     bx,cl
                mov     ah,0
                add     bx,ax
                jmp     newchar
exit:           ret
hexibin         endp
```

0~9 ?

a~f ?

(bx)←(bx) ×16+(ax)

binidec子程序


;BX:被除数
;CX:除数
MOV DX,0
MOV AX,BX
DIV CX
MOV BX,DX
(BX)/10000
余数：(DX)/1000
余数：(DX)/100
余数：(DX)/10
余数：(DX)/1
返回
商：(AX)
显示
MOV DL,AL
ADD DL,30H
MOV AH,02
INT 21H

binidec子程序

```
binidec         proc    near
;将(bx)中的数转换为十进制
;  并显示
          mov     cx,10000d
          call    dec_div
          mov     cx,1000d
          call    dec_div
          mov     cx,100d
          call    dec_div
          mov     cx,10d
          call    dec_div
          mov     cx,1d
          call    dec_div
          ret
binidec         endp
```

```
dec_div         proc    near
;被除数:bx
;  除数: cx
;商: ax  显示
;  余数:dx  dx→bx,新的被除数
          mov     ax,bx
          mov     dx,0
          div     cx
          mov     bx,dx
          mov     dl,al
          add     dl,30h
          mov     ah,02h
          int     21h
          ret
dec_div         enp
```

数值输入输出小结

- 数值输入
  - 输入十进制**p199**
  - 输入十六进制**p225**
- 数值输出
  - 输出十进制**p225**
  - 输出十六进制**p199**