3.11 子程序调用返回指令

子程序：功能确定且独立的程序段。

优点：

- 可以将任何一段独立的程序归整为一个子程序，当需要该段程序时，只需调用子程序即可，调用后会自动返回到调用指令的下一条指令。因此采用子程序设计时，可以简化程序设计。
- 从调试程序的角度，由于原本在多处出现的程序段，缩减为子程序调用指令，使调试程序更加方便。

3.11 子程序调用返回指令

缺点：

- 采用子程序设计后。由于调用子程序和从子程序中返回需要执行指令，并且为保护某些寄存器的内容，需要进行压入堆栈和弹出堆栈的操作，因此会使程序执行速度受到一定的影响。

3.11 子程序调用返回指令

实现：

- 子程序调用是通过自动修改（IP）或（IP）&（CS）的内容实现的。
- 为了确保子程序调用后能够返回到调用指令之后，CALL指令会自动保存返回地址（IP或IP&CS），RET指令会自动返回到CALL指令的下一条指令。

3.11 子程序调用返回指令

1. 子程序调用指令CALL

子程序调用指令CALL有两种格式：

CALL　LABEL；调用入口地址为标号LABEL的子程序

CALL　OPR；调用子程序，其入口地址为操作数OPR的内容

3.11 子程序调用返回指令

2. 子程序返回指令RET

有三种格式：

RET　；用于段内子程序的返回，完成IP 出栈，即(IP)←(SP)

RETF　；用于段间子程序的返回，完成IP 出栈，CS 出栈

RET　n；完成RET(或RETF)指令功能后，(SP)←(SP)+n

3.11 子程序调用返回指令

3. 过程定义

在IBM PC汇编过程中，子程序通常以过程方式编写。

过程定义格式：

过程名　PROC　[类型]

　　RET

过程名　ENDP

- 过程名是用户给子程序起的名字，可以看作为标号，具有段地址、偏移地址和类型的属性。
- 子程序的类型可以取NEAR（近程过程，可供段内调用）和FAR（远程过程，可供段间调用），当类型缺省时，表示NEAR（近程过程）。

3.11 子程序调用返回指令

- 一个子程序名一但定义，就具有以下三个属性

段地址
段内偏移地址
> 是子程序第一条指令的地址

类型
> NEAR(可以缺省)
> FAR

3.11 子程序调用返回指令

子程序还可以是另一种形式：

〈标号〉：

RET

3.11 子程序调用返回指令

子程序调用与返回指令应用结构

1. 段内子程序调用与返回

```
；主程序
CODE   SEGMENT
       ASSUME  CS:CODE
START:
          ⋮
       CALL NEAR PTR SUB1
          ⋮
       MOV  AH, 4CH
       INT   21H
```

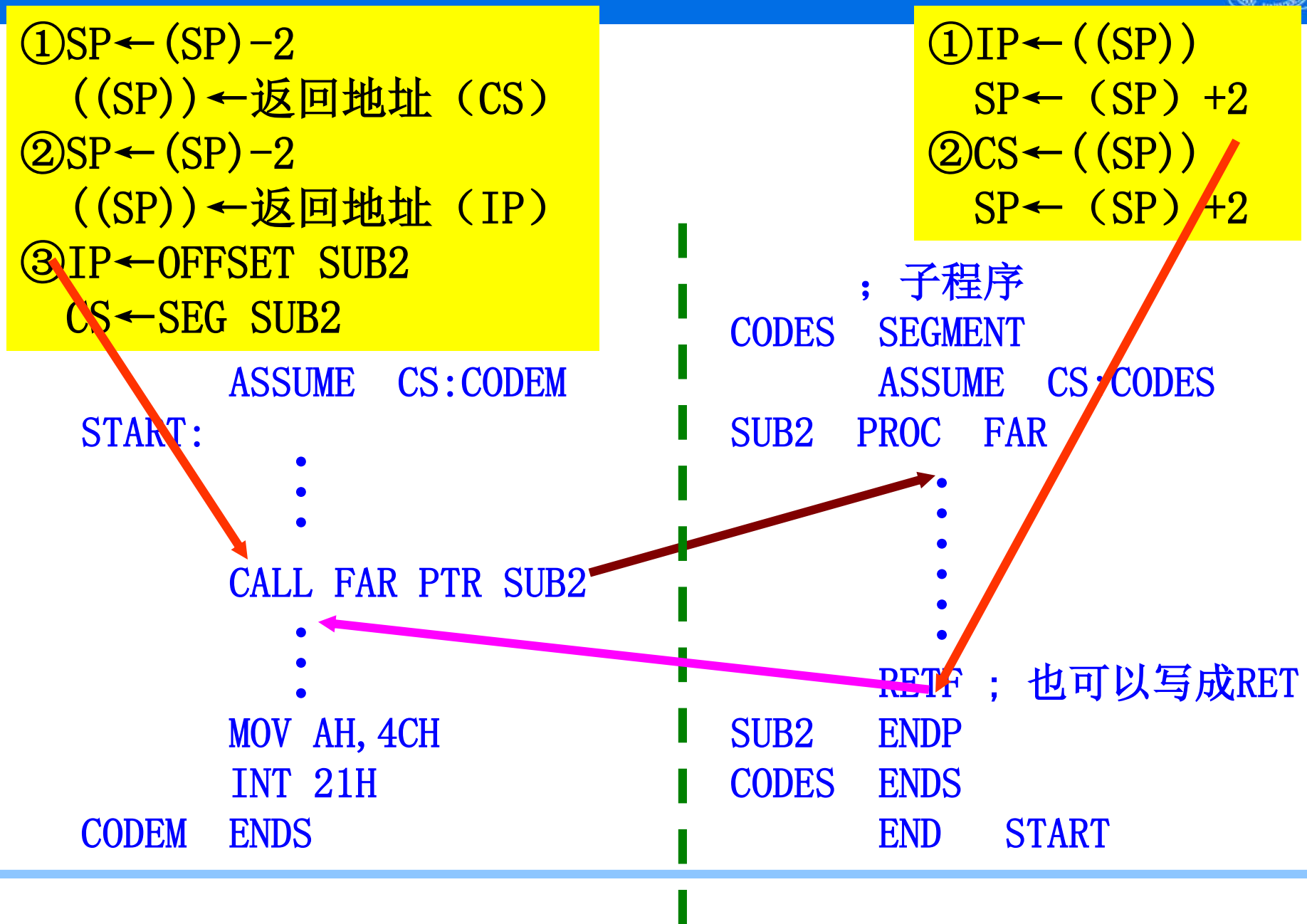
可以省略

```
;子程序SUB1
SUB1 PROC NEAR
          ⋮
       RET;
SUB1  ENDP
CODE  ENDS
       END  START
```

- IP←((SP))
- SP←(SP)+2

- SP←(SP)-2
- ((SP))←返回地址（IP）
- IP←(IP)+16位DISP

3.11 子程序调用返回指令



3.11 子程序调用返回指令

例. 编写子程序实现统计一个字（AX）中“1”的个数。

思路：利用移位指令或循环移位指令，每次对CF位进行检测

- 当CF=1时，则总个数加1；
- 当CF=0时，则总个数不变。

这种操作可以采用有条件转移指令来实现，但更方便的方式是采用ADC指令实现。

3.11 子程序调用返回指令

子程序如下：

```
COUNTER1   PROC NEAR
      PUSH AX
      MOV   CX, 16
      XOR    BL, BL
COU1:
      SHR    AX, 1
      ADC    BL, 0
      LOOP COU1
      POP   AX
      RET
COUNTER1 ENDP
```

3.11 子程序调用返回指令

例. 利用上例设计的子程序，统计字型变量VAR1中1的个数。

在数据段中定义变量VAR1和CounterVar1：

```
VAR1     DW     1234H
CounterVar1     DB     ?
```

程序片段：

```
MOV      AX, VAR1
CALL     COUNTER1
MOV      CounterVar1, BL
```

结果：单元CounterVar1的值为5，说明1234H中包含有5个“1”

3.11 子程序调用返回指令

例. 编写以十六进制数显示AL和AX内容的子程序（DISPAL、DISPAX）。

思路:

- 先编写显示AL寄存器内容的子程序DISPAL
- 由于AL中有两位十六进制数，每一位的值为0～9、0AH～0FH，在显示时需要将它们转换成相应的ASCII码
- 调用INT 21H的02H号功能进行显示

3.11 子程序调用返回指令

十六进制数变换成ASCII码（仅处理AL低4位字符）：

```
HEXtoASC PROC NEAR
        CMP AL, 10; 高四位清零情况下
        JB    LAB
        ADD AL, 7
LAB:
        ADD AL, 30H
        RET
HEXtoASC   ENDP
```

3.11 子程序调用返回指令

DISPAL子程序如下：

```
DISPAL   PROC   NEAR
   PUSH  AX
   PUSH  CX
   PUSH  DX
   PUSH  AX
   MOV   CL, 4; 处理高位十六进
                ; 制数
   SHR   AL, CL
   CALL  HEXtoASC; 十六进制数变
                 ; 换成ASCII码
   MOV   AH, 02;
   MOV   DL, AL;
                          显示一位字符
   INT   21H;
   POP   AX
   AND AL,0FH; 处理低位十六进制数
   CALL HEXtoASC; 十六进制数变换成
                ; ASCII码
   MOV AH,02
   MOV DL,AL
   INT 21H;          显示一位字符
   POP DX
   POP CX
   POP AX
   RET
DISPAL   ENDP
```

3.11 子程序调用返回指令

现在可以直接调用DISPAL实现显示AX的内容

子程序DISPAX内容如下：

```
DISPAX   PROC   NEAR
         XCHG AL,AH
         CALL DISPAL
         XCHG AH,AL
         CALL DISPAL
         RET
DISPAX  ENDP
```