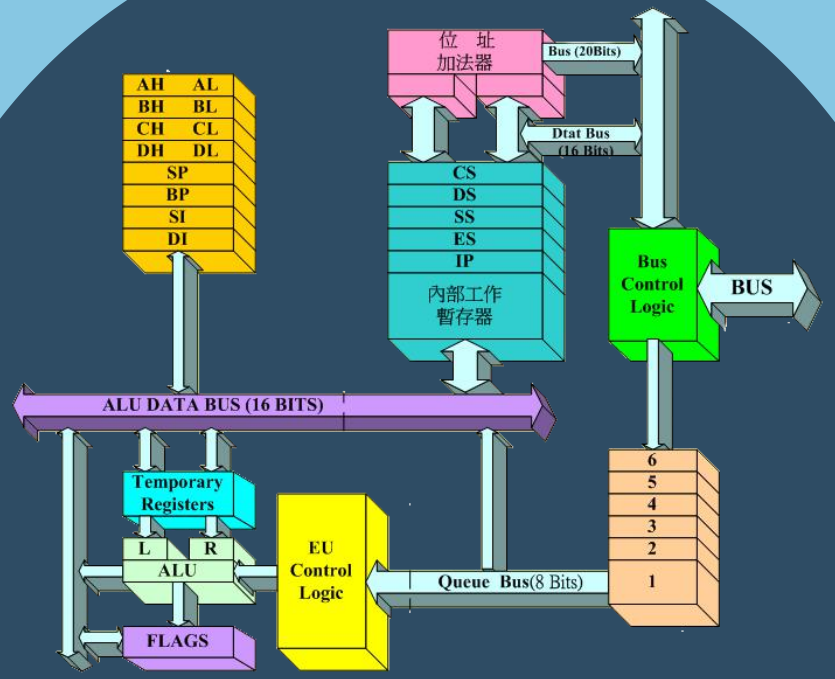
汇编语言的模块化程序设计

贺利坚　主讲

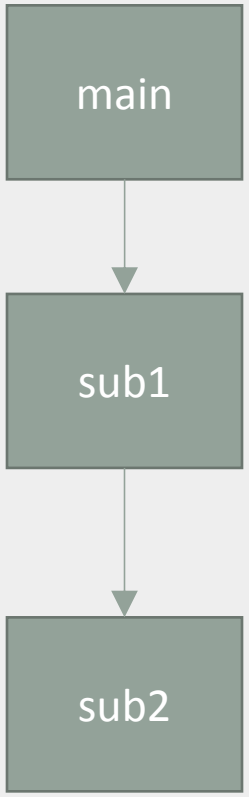


汇编语言程序设计

Assembly Language

模块化程序设计

```
1  assume cs:code
2  code segment
3  main: ...
4        call sub1
5        ...
6        mov ax, 4c00h
7        int 21h
8
9  sub1: ...
10       call sub2
11       ...
12       ret
13
14 sub2: ...
15       ...
16       ret
17 code ends
18 end main
```



- 调用子程序：call指令
- 返回：ret 指令
- 子程序：根据提供的参数处理一定的事务，处理后，将结果（返回值）提供给调用者。

参数和结果传递的问题

- 问题：根据提供的N，计算N的3次方。
- 考虑
  - （1）我们将参数N存储在什么地方？
  - （2）计算得到的数值，存储在什么地方？
- 方案
  - 用寄存器传递参数
  - 用内存单元进行参数传递
  - 用栈传递参数

```
1  assume cs:code
2  code segment
3  main: ...
4        call sub1
5        ...
6        mov ax, 4c00h
7        int 21h
8
9  sub1: ...
10       call sub2
11       ...
12       ret
13
14 sub2: ...
15       ...
16       ret
17 code ends
18 end main
```

```
#include <stdio.h>
int cube(int x);
int main()
{
   printf("%d\n",cube(2));
   return 0;
}
int cube(int x)
{
   int f;
   f=x*x;
   f=f*x;
   return f;
}
```

用寄存器来存储参数和结果是最常使用的方法

- 问题：根据提供的N，计算N的3次方。
- 考虑
  - （1）将参数N存储在什么地方？
  - （2）计算得到的数值，存储在什么地方？
- 用寄存器传递参数
  - 参数放到 bx 中，即(bx)=N
  - 子程序中用多个 mul 指令计算 N^3
  - 将结果放到 dx 和 ax中：(dx:ax)=N^3

编程任务：计算data段中第一组数据的 3 次方，
结果保存在后面一组dword单元中。

```
assume cs:code
data segment
    dw 1,2,3,4,5,6,7,8
    dd 0,0,0,0,0,0,0,0
data ends
code segment
start:mov ax,data
    mov ds,ax
    mov si,0
    mov di,16
    ;循环处理
    mov ax,4c00h
    int 21h
    code ends
end start
```

```
    mov cx,8
s: mov bx,[si]
    call cube
    mov [di],ax
    mov [di].2,dx
    add si,2
    add di,4
    loop s
```

```
cube: mov ax,bx
      mul bx
      mul bx
      ret
```

```
; 汇编子程序
cube: mov ax,bx
      mul bx
      mul bx
      ret
```

如果需要传递的数据有3个、4个或更多，寄存器不够了，怎么办？

```
C:\>debug p10-5.exe
-g

Program terminated normally
-d 076a:0 2f
076A:0000  01 00 02 00 03 00 04 00-05 00 06 00 07 00 08 00
076A:0010  01 00 00 00 08 00 00 00-1B 00 00 00 40 00 00 00
076A:0020  7D 00 00 00 D8 00 00 00-57 01 00 00 00 02 00 00
```

用内存单元批量传递数据

- 方案
  - 将批量数据放到内存中，然后将它们所在内存空间的首地址放在寄存器中，传递给需要的子程序。
  - 对于具有批量数据的返回结果，也可用同样的方法。
- 编程：将data段中的字符串转化为大写。

```
assume cs:code
data segment
    db 'conversation'
data ends
code segment
  start: ……
code ends
end start
```

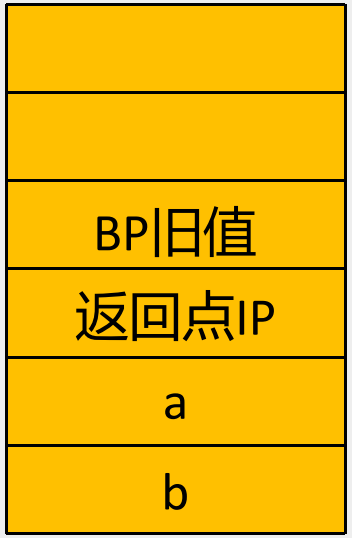
```
C:\>debug p10-6.exe
-r
AX=FFFF  BX=0000  CX=002A  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076B  IP=0000   NV UP EI PL NZ NA PO NC
076B:0000 B86A07        MOV     AX,076A
-g

Program terminated normally
-d 076a:0 f
076A:0000  43 4F 4E 56 45 52 53 41-54 49 4F 4E 00 00 00 00   CONVERSATION....
-
```

```
code segment
start:  mov ax,data
        mov ds,ax
        mov si,0
        mov cx,12
        call capital
        mov ax,4c00h
        int 21h
capital: and byte ptr [si],11011111b
        inc si
        loop capital
        ret
code ends
```

用栈传递参数

- 原理：由调用者将需要传递给子程序的参数压入栈中，子程序从栈中取得参数
- 任务：计算(a – b) ^ 3 ，a、b 为 word 型数据。
  - 进入子程序前，参数a、b入栈
  - 调用子程序，将使栈顶存放IP
  - 结果：(dx : ax) = (a – b) ^ 3
- 例：设 a = 3 、b = 1 ，计算：(a – b) ^ 3

```
mov  ax , 1
push  ax
mov  ax , 3
push  ax
call  difcube
```



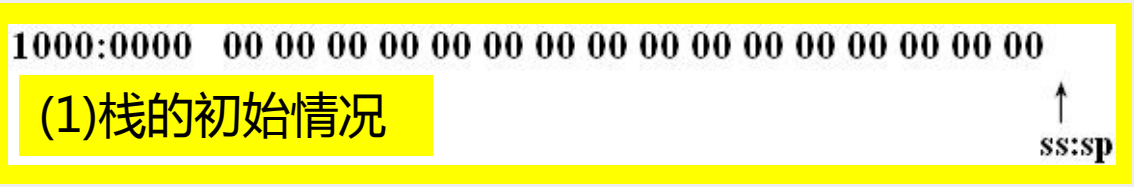
指令 ret n 的含义

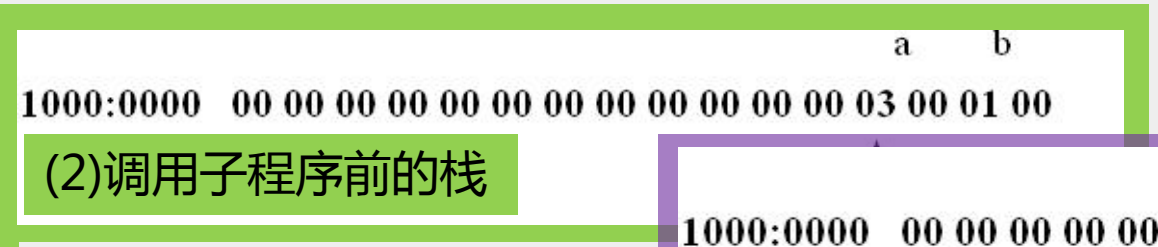
```
pop  ip
add  sp , n
```

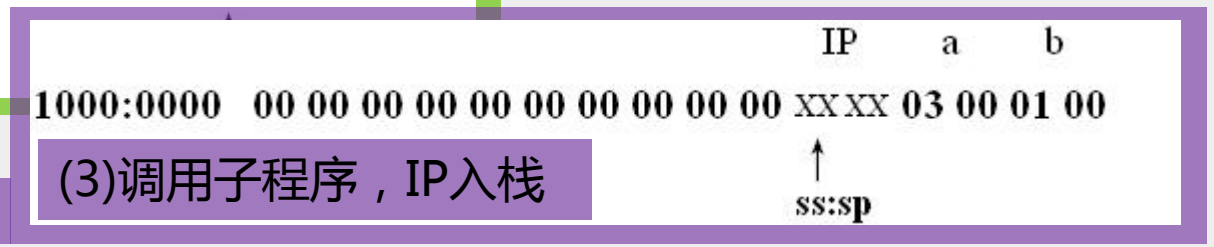
```
difcube : push  bp
      mov  bp , sp
      mov  ax , [ bp + 4 ] ; 将栈中a的值送入ax 中
      sub   ax , [ bp + 6 ] ; 减栈中b的值
      mov  bp , ax
      mul  bp
      mul  bp
      pop  bp
      ret  4
```

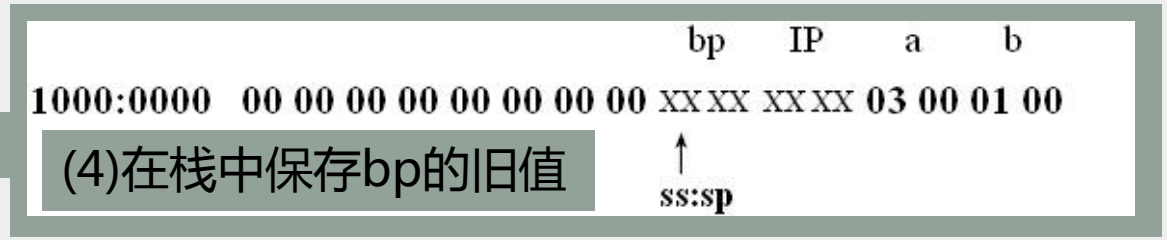
程序的执行过程中栈的变化

```
code segment
start:  mov  ax , 1
        push  ax
        mov  ax , 3
        push  ax
        call  difcube
        mov ax,4c00h
        int 21h
difcube : push  bp
        mov  bp , sp
        mov  ax , [ bp + 4 ]
        sub   ax , [ bp + 6 ]
        mov  bp , ax
        mul  bp
        mul  bp
        pop  bp
        ret  4
code ends
```



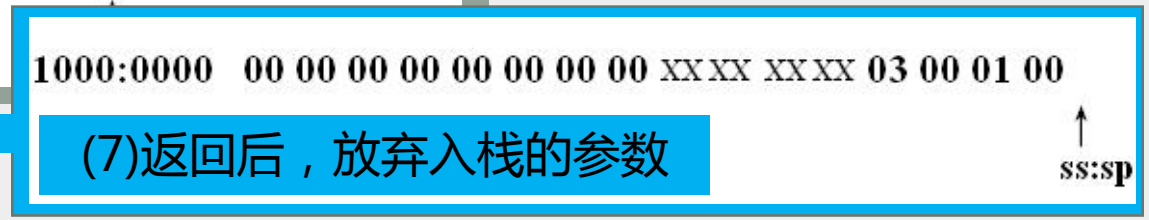






重要技术：子程序要用bp，为避免丢失有用数据，先入栈，返回前出栈。

(5) 从栈中获得参数并计算
计算结果(返回值)在dx和ax中



小结：参数和结果传递的问题

- 问题：根据提供的N，计算N的3次方。
- 考虑
  - （1）我们将参数N存储在什么地方？
  - （2）计算得到的数值，存储在什么地方？
- 方案
  - 用寄存器传递参数
  - 用内存单元进行参数传递
  - 用栈传递参数