

1. 下列程序执行完后，物理地址单元01070H的内容是什么？

```
MOV AX,00E0H  
MOV DS,AX  
MOV BX,0200H  
MOV CX,BX  
NEXT: MOV [BX],BL  
INC BL  
LOOP NEXT  
MOV AH,4CH  
INT 21H
```

解答这类问题，首先要明确物理地址和逻辑地址的关系，物理地址01070H对应到00E0H段的偏移地址为 $01070H - 00E00H = 270H$ ，执行上述程序段可知该单元内容为70H。

2. 已知当前的DS和ES指向同一段，且当前数据段从0000H到00FFH的内容分别是01, 02, 03, ..., OFEH, OFFH, 00H，问下列程序执行后，0000H~00FFH单元的内容有什么变化？

MOV DI, 0001H

MOV SI, 0000H

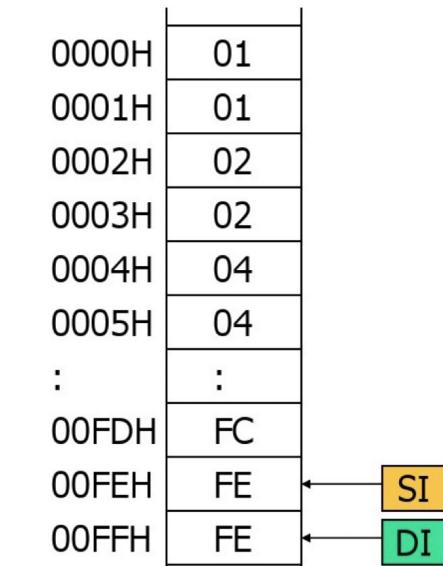
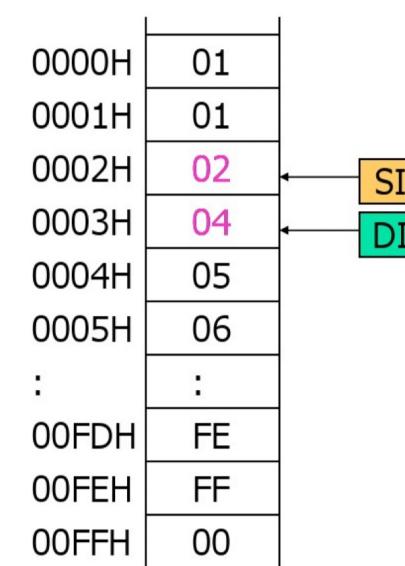
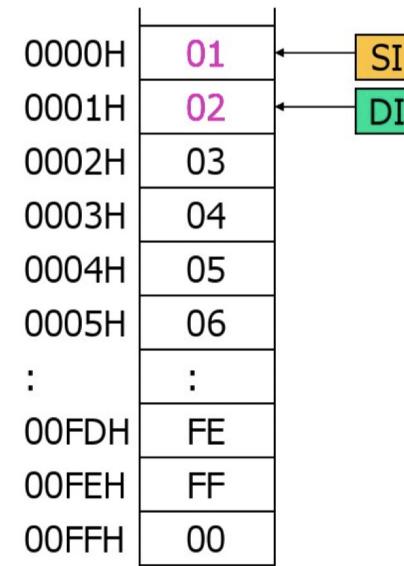
MOV CX, 0080H

CLD

REP MOVSW

MOV AH, 4CH

INT 21H



课后作业 — 第七章

3. 填充下面程序的空白, 使程序完成预定功能。程序的功能是将在NUMB开始的单元存放的一组数据中的正数和负数分别挑出来, 并分别存入NUMP和NUMM开始的单元, 其个数分别放入CNTP和CNTM单元。已知原数据的个数存放在CNTB单元。

```
DSEG SEGMENT
NUMB DB 10,-1,90,123,24,-17,30H,99,103,-1
CNTB DB 10
CNTP DB 0
CNTM DB 0
NUMP DB 10 DUP (0)
NUMM DB 10 DUP (0)
DSEG ENDS
CSEG SEGMENT
ASSUME CS:CSEG,DS:DSEG,ES:DSEG
START: MOV AX,DSEG
       MOV DS,AX
       MOV ES,AX
MOV CL,CNTB
MOV CH,( 0 )
XOR DX,DX
MOV SI,OFFSET NUMB
MOV DI,OFFSET NUMM
MOV BX,OFFSET NUMP
(CLD)
NEXT: LODSB
      ( AND AL,AL )
      JNS PLUS
      STOSB
      INC DL
UP: LOOP NEXT
     MOV CNTM,DL
     MOV CNTP,DH
     MOV AH,4CH
     INT 21H
PLUS: MOV [BX],AL
      ( INC BX )
      INC DH
      JMP UP
CSEG ENDS
END START
```

- |▶ 4. 编程：计算下述数据的平均值，并将大于平均值的数据存入AE开始的单元。已知数据为无符号数，数据个数存储在CNT中。数据段初始状态如下：

```
1:DSEG      SEGMENT  
2:DATA      DW      1524, 2748, 13, 56, 47, 634  
3:CNT       DW      6  
4:AE        DW      6 DUP (0)  
5:DSEG      ENDS
```

课后作业 – 第七章

4. 编程：计算下述数据的平均值，并将大于平均值的数据存入AE开始的单元。已知数据为无符号数，数据个数存储在CNT中。数据段初始状态如下：

```
2 DSEG SEGMENT
3 DATA DW 1524, 2748, 13, 56, 47, 634
4 CNT DW 6
5 AE DW 6 DUP(0)
6 DSEG ENDS
7
8 CSEG:SEGMENT
9     DS:DSEG CS:CSEG
10    START:
11        MOV AX,DSEG
12        MOV DS,AX
13        MOV SI,OFFSET DATA
14        MOV CX,CNT
15        PUSH CX
16        XOR AX,AX
17        XOR DX,DX
18    AGAIN:
19        ADD AX,[SI]
20        ADC DX,0
```

```
21        ADD SI,2
22        DEC CL
23        JNZ AGAIN
24        POP CX
25        DIV CX
26        XOR BP,BP
27    PER:
28        SUB SI,2
29        CMP AX,[SI]
30        JNB NEXT
31        MOV DX,[SI]
32        MOV DS:AE[BP],DX
33        ADD BP,2
34    NEXT:
35        DEC CX
36        JNZ PRE
37        MOV AH,4CH
38        INT 21H
39    CSEG ENDS
40    ENDS START
```

课后作业 – 第七章

4. 编程：计算下述数据的平均值，并将大于平均值的数据存入AE开始的单元。已知数据为无符号数，数据个数存储在CNT中。数据段初始状态如下：

```
2  SSEG  SEGMENT STACK
3  STACK  DB      50 DUP(0)
4  SSEG  ENDS
5
6  DSEG  SEGMENT
7  DATA  DW  1524, 2748, 13, 56, 47, 634
8  CNT   DW  6
9  AE    DW  6 DUP(0)
10 DSEG  ENDS
11
12 CSEG  SEGMENT
13  ASSUME DS:DSEG, CS:CSEG
14  ASSUME SS:SSEG
15 START:
16  MOV    AX, DSEG
17  MOV    DS, AX
18  MOV    AX, SSEG
19  MOV    SS, AX
20  MOV    SP, SIZE STACK
21  MOV    SI, OFFSET DATA
22  MOV    CX, CNT
23  PUSH   CX
24  XOR    AX, AX
```

```
25  XOR    DX, DX
26 AGAIN:
27  ADD   AX, [SI]
28  ADC   DX, 0
29  ADD   SI, 2
30  DEC   CL
31  JNZ   AGAIN
32  POP   CX
33  DIV   CX
34  XOR   BP, BP
35 PER:
36  SUB   SI, 2
37  CMP   AX, [SI]
38  JNB   NEXT
39  MOV   DX, [SI]
40  MOV   DS:AE[BP], DX
41  ADD   BP, 2
42 NEXT:
43  DEC   CX
44  JNZ   PER
45  MOV   AH, 4CH
46  INT   21H
47 CSEG  ENDS
48 END   START
```



東北大學
Northeastern University

汇编语言程序设计

主讲：

刘松冉

单位：

东北大学计算机学院

智慧系统国际联合实验室

联系方式：liusongran@cse.neu.edu.cn

个人主页：<http://faculty.neu.edu.cn/liusongran>

<https://liusongran.github.io/>



第八章 子程序设计

- 一. 子程序的引出
- 二. 子程序（过程）定义伪指令
- 三. 调用和返回指令
- 四. 子程序设计方法
- 五. 子程序嵌套
- 六. 递归子程序
- 七. 可重入子程序
- 八. 程序的连接



一. 子程序的引出

|> 在我们编写解决实际问题的程序时，往往会遇到多处使用相同功能的程序段，使用该程序段的唯一差别是对程序变量赋不同的值，例如计算：

$$S = \sqrt{2X} + \sqrt{3Y} + \sqrt{144}$$

计算上述函数需要多次使用开方运算，如果每次用到开方运算就编写一段开方程序，那么开方程序在程序中会多次出现，不仅书写麻烦，容易出错，编辑、汇编它时，也会花费较多时间。同时，由于冗长，占用内存也较多。如果把多次使用的功能程序编制为一个独立的程序段，每当用到这种功能时，就将控制转向它，完成功能后再返回到原来的程序，这就会大大减少编程工作量。

一. 子程序的引出

|> 使用子程序的好处：

- 1) 简化了程序设计过程，减少了工作量，节省了时间；
- 2) 源程序缩短，节省了机器汇编源程序的时间和存储目标代码的存储空间；
- 3) 增加了源程序的可读性，便于调试维护；
- 4) 有利于程序模块化、结构化和自顶向下的程序设计；
- 5) 子程序一旦编制成功，在开发研制各种软件时都可利用，大大缩短了软件的开发周期。

一. 子程序的引出

► 子程序设计需了解的问题：

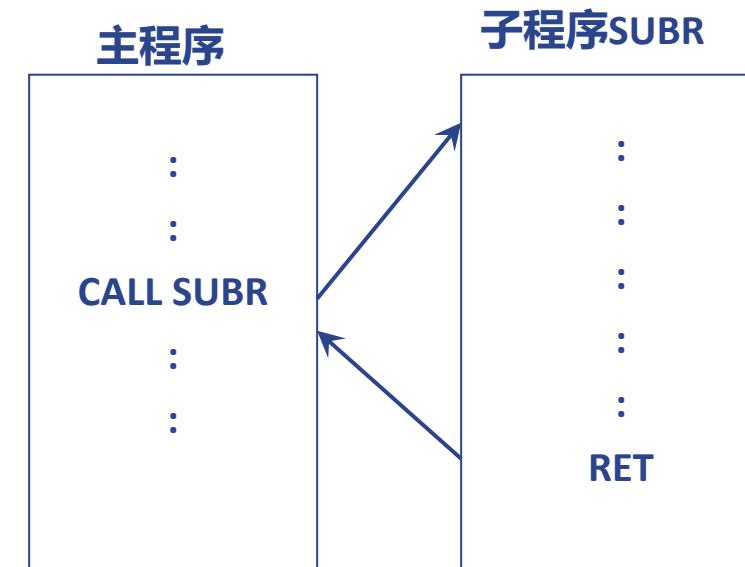
1) 子程序的定义与调用：

设计出可以完成相对独立功能的程序段，并对汇编程序提供必要的信息，使得对该程序段汇编之后，在需要时可以被其他主程序调用，这一过程称为子程序的定义。

已定义的子程序在什么时候被执行，执行多少次，都不是由子程序自身决定的。在主程序需要调用相应的子程序时，可以用调用子程序指令使得相应子程序被执行。

2) 主程序与子程序之间的调用与返回：

在主程序需要利用子程序的执行来完成某种工作时，就可以使用调用子程序指令来调用相应的子程序；在子程序完成它应该完成的工作之后，应当用返回指令返回调用它的主程序



二. 子程序(过程)定义伪指令

► 子程序定义伪指令所定义的子程序的一般格式：

```
PN  PROC [NEAR] / [FAR]      ; 说明过程开始  
    :  
    :  
PN  ENDP                      ; 说明过程结束
```

```
2   SQR00T1  PROC     NEAR  
3       XOR AX, AX  
4       AND DX, DX  
5       JZ  SQRT2  
6 SQRT1: MOV BX, AX  
7       SHL BX, 1  
8       INC BX  
9       SUB DX, BX  
10      JC   SQRT2  
11      INC AX  
12      JMP SQRT1  
13 SQRT2: MOV DX, AX  
14       RET  
15 SQR00T1 ENDP
```

三. 调用和返回指令

1. 调用分类

1) 段内调用与段间调用

段内调用：调用和返回过程不涉及CS的变化，只通过IP内容的变化实现控制的转反。

段间调用：调用和返回过程涉及CS的变化，需要通过CS和IP内容的变化实现控制的转反。

注意：当子函数/过程和主程序在同一代码段时，可以把子函数定义为NEAR或FAR。当不在同一代码段时，只能定义为FAR。

2) 直接调用与间接调用

直接调用：当调用指令使用子函数/过程名调用时，调用是通过把该函数/过程的指令入口地址偏移量直接送入IP中或CS和IP中来实现的。

间接调用：当调用指令使用子函数/过程名调用时，通过某寄存器或某存储单元指出被调用子程序的入口地址时。

注意：在实际使用时，直接调用因为方便清楚而使用较为广泛。

三. 调用和返回指令

|▶ 2. 调用指令

- 指令汇编格式: CALL PROCNAME/REGNAME/MEMLABEL

- 操作:

- 1) 段内调用: $SP \leftarrow SP - 2$
 $(SP+1, SP) \leftarrow IP$
 $IP \leftarrow \text{OFFSET PROCNAME/REGNAME/MEMLABEL}$
- 2) 段间调用: $SP \leftarrow SP - 2$
 $(SP+1, SP) \leftarrow CS$
 $CS \leftarrow \text{SEG PROCNAME/REGNAME/MEMLABEL}$
 $SP \leftarrow SP - 2$
 $(SP+1, SP) \leftarrow IP$
 $IP \leftarrow \text{OFFSET PROCNAME/REGNAME/MEMLABEL}$

- 受影响的状态标志位: 没有

三. 调用和返回指令

|▶ 3. 返回指令

- 指令汇编格式: RET [VAL]
- 操作:
 - 1) 段内返回: $IP \leftarrow (SP+1, SP)$
 $SP \leftarrow SP+2$
 $SP \leftarrow SP + VAL$ (如果选用VAL)
 - 2) 段间返回: $IP \leftarrow (SP+1, SP)$
 $SP \leftarrow SP+2$
 $CS \leftarrow (SP+1, SP)$
 $SP \leftarrow SP+2$
 $SP \leftarrow SP + VAL$ (如果选用VAL)
- 受影响的状态标志位: 没有

四. 子程序设计方法

- |▶ 1. 现场的保护和恢复
- 2. 子程序说明文件
- 3. 主程序与子程序之间的参数传递

四. 子程序设计方法

► 1. 现场的保护和恢复

子程序中需要使用的寄存器，有可能在主程序中正被用来保存某种中间结果，这些寄存器的值在从子程序返回主程序后还要继续使用，**这些寄存器的值或所需的标志位的值等信息称之为现场**。显然，子程序执行前需要保护现场，返回时要恢复现场。

保存现场与恢复现场的工作既可在调用程序中完成，也可在子程序中完成，程序设计时根据情况安排。如果子程序已经设计好了，而其中未保护主程序现场，那么调用程序在使用子程序之前应保护现场，从子程序返回后再恢复现场。

通常在主程序中保护现场，则一定在主程序中恢复；在子程序中保护现场，则一定在子程序中恢复。这样可以增强主程序和子程序之间的相对独立性，减少相互依赖，使程序结构清楚，减少错误。

四. 子程序设计方法

1. 现场的保护和恢复

保护现场和恢复现场的方法：

- 1) 利用压栈和出栈指令，将寄存器内容或状态标志位内容保存在堆栈中，恢复时再从堆栈中取出。

例子：

```
SQROOT1    PROC NEAR
            PUSH AX          ;保存现场
            PUSH BX
            PUSH CX
            ...
            POP  CX          ;子程序正常工作
            POP  BX          ;恢复现场
            POP  AX
            RET
SQROOT1    ENDP
```

- 2) 利用内存单元。用传送指令将寄存器的内容保存到指定的内存单元，恢复时再用传送指令取出。

四. 子程序设计方法

▶ 2. 子程序说明文件

- 1) **子程序名** (子程序入口地址) : 用过程 (子程序) 定义伪指令定义该过程 (子程序) 时的过程名, 这时过程 (子程序) 中第一条语句必须是子程序的入口指令; 否则应写子程序入口指令的标号或地址。
- 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任务。
- 3) **入口条件** : 说明子程序要求有几个入口参数, 这些参数表示的意义及存放位置。
- 4) **出口条件** : 说明子程序有几个输出参数 (运行结果), 这些参数表示的意义、存放的位置。
- 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。

子程序说明文件例子 :

- 1) 子程序名 : SQROOT1;
- 2) 程序功能 : 求双字节整数的平方根的整数部分;
- 3) 入口条件 : 被开方数放在DX中;
- 4) 出口条件 : 平方根在DX中;
- 5) 受影响的寄存器 : AX,BX,CX,DX 及标志寄存器F。

四. 子程序设计方法

|> 3. 主程序与子程序之间的参数传递

子程序运行时所需的加工对象应由调用程序提供，处理的结果应提供调用程序使用，二者之间要进行信息交换或者说相互之间要传递参数。

常用的传送方法有四种：

- 1) 约定寄存器法；
- 2) 约定存储单元法；
- 3) 堆栈法；
- 4) 约定参数地址指针法。

四. 子程序设计方法

|▶ 3. 主程序与子程序之间的参数传递

例：编写程序计算下述函数。其中X，Y为整数字数据，且存于PX和PY单元，计算结果存入RLT单元。

$$S = \sqrt{2X} + \sqrt{3Y} + \sqrt{144}$$

四. 子程序设计方法

► 3. 主程序与子程序之间的参数传递 – 约定寄存器法

例：编写程序计算下述函数。其中X，Y为整数字数据，且存于PX和PY单元，计算结果存入RLT单元。

$$S = \sqrt{2X} + \sqrt{3Y} + \sqrt{144}$$

约定寄存器法传送参数，即事先约定一些存放参数的通用寄存器，调用程序转向子程序时，先把要传送的参数放到约定好的寄存器中，子程序工作时，从约定的寄存器中取参数，然后把结果存入事先约定的寄存器中，调用程序再从约定的寄存器中取结果。这种方法是程序设计中最常用、最简单、最方便的方法，在传送参数不多的情况下都采用此方法。

四. 子程序设计方法

3. 主程序与子程序之间的参数传递 — 约定寄存器法

例：编写程序计算下述函数。其中X，Y为整数字数据，且存于PX和PY单元，计算结果存入RLT单元。

$$S = \sqrt{2X} + \sqrt{3Y} + \sqrt{144}$$

开平方子程序和调用程序，约定将被开平方数放在DX中，平方根也放在DX中，其结果求得平方根的整数部分。

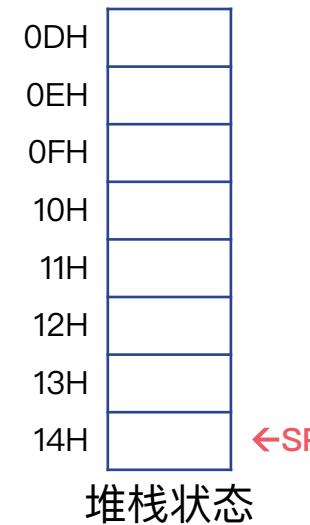
```
2 SQR00T1 PROC NEAR
3             XOR AX, AX      ; i←0
4             AND DX, DX      ; 测试被开方数
5             JZ  SQRT2        ; 被开方数为0
6 SQRT1:   MOV BX, AX        ; 形成奇数
7             SHL BX, 1
8             INC BX
9             SUB DX, BX        ; 被开方数减去奇数
10            JC   SQRT2       ; 不够减
11            INC AX          ; 够减, i增1
12            JMP SQRT1        ; 继续
13 SQRT2:   MOV DX, AX        ; DX←平方根
14            RET              ; 返回
15 SQR00T1 ENDP
```

子函数

四. 子程序设计方法

3. 主程序与子程序之间的参数传递 — 约定寄存器法

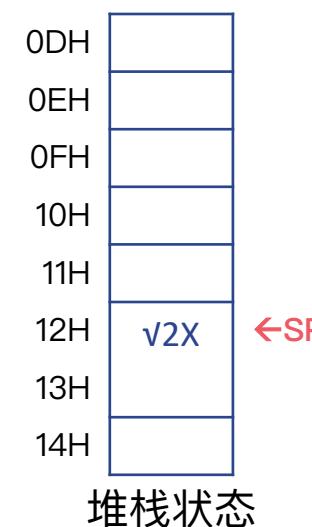
```
2  SSEG    SEGMENT      STACK
3  STKTOP  DB      20 DUP(0)
4  SSEG    ENDS
5
6  DSEG    SEGMENT
7  PX      DW      12345
8  PY      DW      2469
9  RLT     DW      0
10 DSEG   ENDS
11
12 CSEG   SEGMENT
13     ASSUME CS:CSEG, DS:DSEG
14     ASSUME SS:SSEG
15 MAIN1: MOV    AX, DSEG
16     MOV    DS, AX
17     MOV    AX, SSEG
18     MOV    SS, AX
19     MOV    SP, SIZE STKTOP
```



四. 子程序设计方法

3. 主程序与子程序之间的参数传递 — 约定寄存器法

```
20    MOV    DX, PX      ;取X  
21    ADD    DX, DX      ;计算2X  
22    CALL   SQR00T1    ;调用开平方子程序  
23    PUSH   DX          ;暂存结果 $\sqrt{2}X$   
24    →MOV   DX, PY      ;取Y  
25    MOV    AX, DX      ;计算 $3Y$   
26    ADD    DX, DX  
27    ADD    DX, AX  
28    CALL   SQR00T1    ;调用开平方子程序  
29    POP    AX          ;取出 $\sqrt{2}X$   
30    ADD    AX, DX      ;计算 $\sqrt{2}X + \sqrt{3}Y$   
31    PUSH   AX          ;暂存结果  
32    MOV    DX, 150  
33    CALL   SQR00T1  
34    POP    AX          ;取出中间结果  
35    ADD    AX, DX      ;计算最终结果  
36    MOV    RLT, AX      ;保存结果  
37    MOV    AH, 4CH  
38    INT    21H  
39 CSEG  ENDS  
40 END   MAIN1
```



| | | | |
|----|---------|------|--------|
| 2 | SQR00T1 | PROC | NEAR |
| 3 | | XOR | AX, AX |
| 4 | | AND | DX, DX |
| 5 | | JZ | SQRT2 |
| 6 | SQRT1: | MOV | BX, AX |
| 7 | | SHL | BX, 1 |
| 8 | | INC | BX |
| 9 | | SUB | DX, BX |
| 10 | | JC | SQRT2 |
| 11 | | INC | AX |
| 12 | | JMP | SQRT1 |
| 13 | SQRT2: | MOV | DX, AX |
| 14 | | RET | |
| 15 | SQR00T1 | ENDP | |

子函数

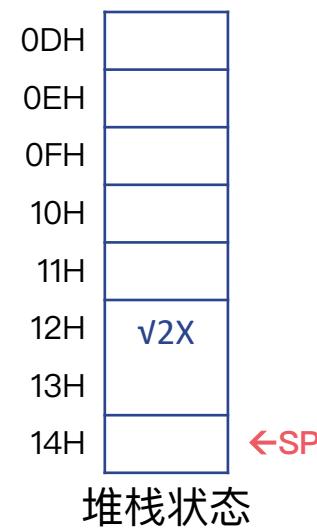
四. 子程序设计方法

3. 主程序与子程序之间的参数传递 — 约定寄存器法

```
20    MOV    DX, PX      ;取X  
21    ADD    DX, DX      ;计算2X  
22    CALL   SQR00T1    ;调用开平方子程序  
23    PUSH   DX          ;暂存结果 $\sqrt{2}X$   
24    MOV    DX, PY      ;取Y  
25    MOV    AX, DX      ;计算3Y  
26    ADD    DX, DX  
27    ADD    DX, AX  
28    CALL   SQR00T1    ;调用开平方子程序  
29    POP    AX          ;取出 $\sqrt{2}X$   
30    → ADD   AX, DX      ;计算 $\sqrt{2}X + \sqrt{3}Y$   
31    PUSH   AX          ;暂存结果  
32    MOV    DX, 150  
33    CALL   SQR00T1  
34    POP    AX          ;取出中间结果  
35    ADD   AX, DX      ;计算最终结果  
36    MOV    RLT, AX      ;保存结果  
37    MOV    AH, 4CH  
38    INT    21H  
39 CSEG  ENDS  
40 END   MAIN1
```

| | | | |
|----|---------|------|--------|
| 2 | SQR00T1 | PROC | NEAR |
| 3 | | XOR | AX, AX |
| 4 | | AND | DX, DX |
| 5 | | JZ | SQRT2 |
| 6 | SQRT1: | MOV | BX, AX |
| 7 | | SHL | BX, 1 |
| 8 | | INC | BX |
| 9 | | SUB | DX, BX |
| 10 | | JC | SQRT2 |
| 11 | | INC | AX |
| 12 | | JMP | SQRT1 |
| 13 | SQRT2: | MOV | DX, AX |
| 14 | | RET | |
| 15 | SQR00T1 | ENDP | |

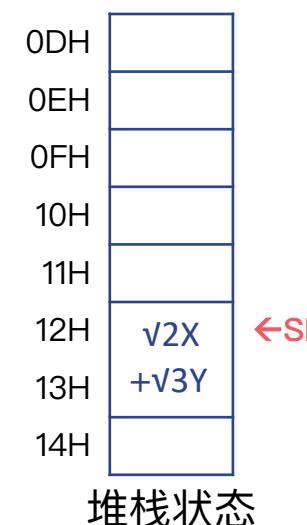
子函数



四. 子程序设计方法

3. 主程序与子程序之间的参数传递 — 约定寄存器法

```
20    MOV    DX, PX      ;取X  
21    ADD    DX, DX      ;计算2X  
22    CALL   SQR00T1    ;调用开平方子程序  
23    PUSH   DX          ;暂存结果 $\sqrt{2}X$   
24    MOV    DX, PY      ;取Y  
25    MOV    AX, DX      ;计算 $3Y$   
26    ADD    DX, DX      ;  
27    ADD    DX, AX      ;  
28    CALL   SQR00T1    ;调用开平方子程序  
29    POP    AX          ;取出 $\sqrt{2}X$   
30    ADD    AX, DX      ;计算 $\sqrt{2}X + \sqrt{3}Y$   
31    PUSH   AX          ;暂存结果  
32    →    MOV    DX, 150  
33    CALL   SQR00T1  
34    POP    AX          ;取出中间结果  
35    ADD    AX, DX      ;计算最终结果  
36    MOV    RLT, AX      ;保存结果  
37    MOV    AH, 4CH  
38    INT    21H  
39 CSEG  ENDS  
40 END   MAIN1
```



| | | | |
|----|---------|------|--------|
| 2 | SQR00T1 | PROC | NEAR |
| 3 | | XOR | AX, AX |
| 4 | | AND | DX, DX |
| 5 | | JZ | SQRT2 |
| 6 | SQRT1: | MOV | BX, AX |
| 7 | | SHL | BX, 1 |
| 8 | | INC | BX |
| 9 | | SUB | DX, BX |
| 10 | | JC | SQRT2 |
| 11 | | INC | AX |
| 12 | | JMP | SQRT1 |
| 13 | SQRT2: | MOV | DX, AX |
| 14 | | RET | |
| 15 | SQR00T1 | ENDP | |

子函数

四. 子程序设计方法

|▶ 3. 主程序与子程序之间的参数传递 – 约定存储单元

例：编写程序计算下述函数。其中X，Y为整数字数据，且存于PX和PY单元，计算结果存入RLT单元。**约定被开方数放在ARGX单元，计算结果放入ROOT单元。**

$$S = \sqrt{2X} + \sqrt{3Y} + \sqrt{144}$$

约定存储单元法传送参数，即事先约定一些存放参数的内存单元，调用程序转向子程序时，先把要传送的参数放到约定好的内存单元中，子程序工作时，从约定的内存单元中取参数，然后把结果存入事先约定的内存单元中，调用程序再从约定的内存单元中取结果。

四. 子程序设计方法

3. 主程序与子程序之间的参数传递 — 约定存储单元

例：编写程序计算下述函数。其中X，Y为整数字数据，且存于PX和PY单元，计算结果存入RLT单元。**约定被开方数放在ARGX单元，计算结果放入ROOT单元。**

```
2     SQR00T1 PROC NEAR
3             XOR AX, AX      ; i←0
4             AND DX, DX      ; 测试被开方数
5             JZ  SQRT2        ; 被开方数为0
6     SQRT1:  MOV BX, AX      ; 形成奇数
7             SHL BX, 1
8             INC BX
9             SUB DX, BX      ; 被开方数减去奇数
10            JC   SQRT2       ; 不够减
11            INC AX          ; 够减, i增1
12            JMP SQRT1        ; 继续
13            SQRT2:  MOV DX, AX  ; DX←平方根
14            RET
15     SQR00T1 ENDP
```

MOV DX, ARGX

MOV DX, ARGX

四. 子程序设计方法

3. 主程序与子程序之间的参数传递 — 约定存储单元法

```
1 ;*****EXAM8.1.2*****          39
2 SSEG SEGMENT STACK           40
3 STKTOP DB 20 DUP(0)         41
4 SSEG ENDS                   42
5
6 DSEG SEGMENT                43
7 PX DW 12345                 44
8 PY DW 2469                  45
9 RLT DW 0                     46
10 ARGX DW 0                  47
11 ROOT DW 0                  48
12 DSEG ENDS                  49
13
14 CSEG SEGMENT               50
15     ASSUME CS:CSEG, DS:DSEG 51
16     ASSUME SS:SSEG          52
17
18 > SQR00T2 PROC NEAR ...      53
33
34 MAIN1: MOV AX, DSEG          54
35     MOV DS, AX                55
36     MOV AX, SSEG              56
37     MOV SS, AX                57
38     MOV SP, SIZE STKTOP       58
1             - 60
2             CSEG ENDS          59
3             END MAIN1          61
```

MOV DX, PX ;取X
ADD DX, DX ;计算2X
MOV ARGX, DX ;存储至ARGX内存单元
CALL SQR00T2 ;调用开平方子程序
PUSH ROOT ;暂存结果 $\sqrt{2}X$
MOV DX, PY ;取Y
MOV AX, DX ;计算 $3Y$
ADD DX, DX
ADD DX, AX
MOV ARGX, DX ;存储至ARGX内存单元
CALL SQR00T2 ;调用开平方子程序
POP AX ;取出 $\sqrt{2}X$
ADD AX, ROOT ;计算 $\sqrt{2}X + \sqrt{3}Y$
PUSH AX ;暂存结果
MOV ARGX, 150
CALL SQR00T2
POP AX ;取出中间结果
ADD AX, ROOT ;计算最终结果
MOV RLT, AX ;保存结果
MOV AH, 4CH
INT 21H

四. 子程序设计方法

3. 主程序与子程序之间的参数传递 — 堆栈法

例：编写程序计算下述函数。其中X，Y为整数字数据，且存于PX和PY单元，计算结果存入RLT单元。**通过堆栈的方式传递参数和返回值。**

堆栈法传送参数是调用程序先将参数压入堆栈，子程序从堆栈中把参数取出，进行处理，然后把结果压入堆栈，调用程序再从堆栈中取出结果。

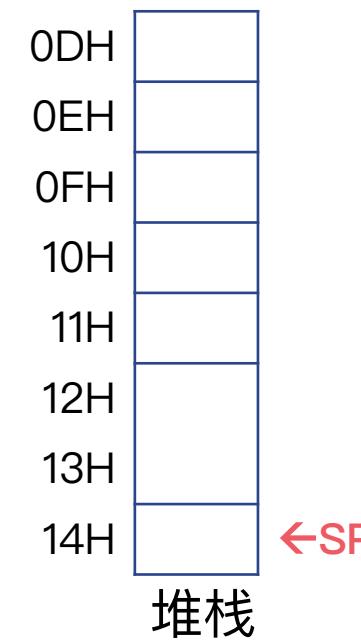
```
18 SQR00T3 PROC NEAR
19           INC  SP
20           INC  SP
21           POP  DX
22           XOR  AX, AX      ; i←0
23           AND  DX, DX      ; 测试被开方数
24           JZ   SQRT2      ; 被开方数为0
25 SQRT1:    MOV  BX, AX      ; 形成奇数
26           SHL  BX, 1
27           INC  BX
28           SUB  DX, BX      ; 被开方数减去奇数
29           JC   SQRT2      ; 不够减
30           INC  AX          ; 够减，i增1
31           JMP  SQRT1      ; 继续
32 SQRT2:    PUSH AX
33           DEC  SP
34           DEC  SP
35           RET
36 SQR00T3 ENDP
```

; 返回

```

36 MAIN1: MOV AX, DSEG
37     MOV DS, AX
38     MOV AX, SSEG
39     MOV SS, AX
40     MOV SP, SIZE STKTOP
41     → MOV DX, PX      ;取X
42     ADD DX, DX       ;计算2X
43     PUSH DX          ;NOTE:
44     CALL SQR00T3     ;调用开平方子程序
45     MOV DX, PY        ;取Y
46     MOV AX, DX        ;计算3Y
47     ADD DX, DX
48     ADD DX, AX
49     PUSH DX          ;NOTE:
50     CALL SQR00T3     ;调用开平方子程序
51     POP DX           ;NOTE:
52     POP AX           ;NOTE:
53     ADD AX, DX        ;计算 $\sqrt{2}X + \sqrt{3}Y$ 
54     PUSH AX          ;暂存结果
55     MOV DX, 150
56     PUSH DX          ;NOTE:
57     CALL SQR00T3
58     POP DX          ;取出中间结果
59     POP AX
60     ADD AX, DX        ;计算最终结果
61     MOV RLT, AX        ;保存结果
62     MOV AH, 4CH
63     INT 21H
64 CSEG ENDS
65 END MAIN1

```



```

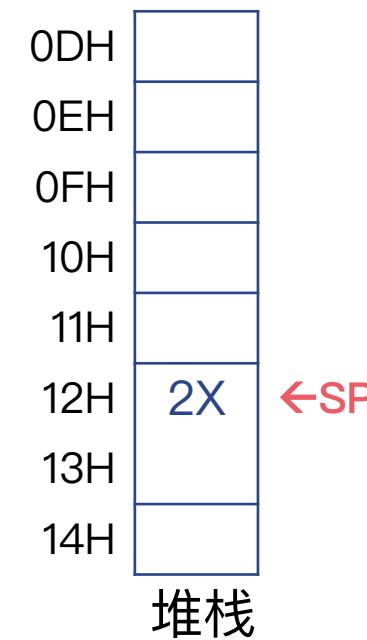
16 SQR00T3 PROC NEAR
17     INC SP
18     INC SP
19     POP DX
20     XOR AX, AX      ;i←0
21     AND DX, DX       ;测试被开方数
22     JZ SQRT2         ;被开方数为0
23 SQRT1: MOV BX, AX    ;形成奇数
24     SHL BX, 1
25     INC BX
26     SUB DX, BX       ;被开方数减去奇数
27     JC SQRT2         ;不够减
28     INC AX            ;够减, i增1
29     JMP SQRT1         ;继续
30 SQRT2: PUSH AX        ;DX←平方根
31     DEC SP
32     DEC SP
33     RET
34 SQR00T3 ENDP

```

```

36 MAIN1: MOV AX, DSEG
37     MOV DS, AX
38     MOV AX, SSEG
39     MOV SS, AX
40     MOV SP, SIZE STKTOP
41     MOV DX, PX      ;取X
42     ADD DX, DX      ;计算2X
43     PUSH DX          ;NOTE:
44     → CALL SQR00T3   ;调用开平方子程序
45     MOV DX, PY      ;取Y
46     MOV AX, DX      ;计算3Y
47     ADD DX, DX
48     ADD DX, AX
49     PUSH DX          ;NOTE:
50     CALL SQR00T3   ;调用开平方子程序
51     POP DX          ;NOTE:
52     POP AX          ;NOTE:
53     ADD AX, DX      ;计算 $\sqrt{2}X + \sqrt{3}Y$ 
54     PUSH AX          ;暂存结果
55     MOV DX, 150
56     PUSH DX          ;NOTE:
57     CALL SQR00T3
58     POP DX          ;取出中间结果
59     POP AX
60     ADD AX, DX      ;计算最终结果
61     MOV RLT, AX      ;保存结果
62     MOV AH, 4CH
63     INT 21H
64 CSEG ENDS
65 END MAIN1

```



```

16 SQR00T3 PROC NEAR
17     INC SP
18     INC SP
19     POP DX
20     XOR AX, AX      ;i←0
21     AND DX, DX      ;测试被开方数
22     JZ SQRT2         ;被开方数为0
23 SQRT1: MOV BX, AX  ;形成奇数
24     SHL BX, 1
25     INC BX
26     SUB DX, BX      ;被开方数减去奇数
27     JC SQRT2         ;不够减
28     INC AX           ;够减, i增1
29     JMP SQRT1        ;继续
30 SQRT2: PUSH AX      ;DX←平方根
31     DEC SP
32     DEC SP
33     RET
34 SQR00T3 ENDP

```

```

36 MAIN1: MOV AX, DSEG
37     MOV DS, AX
38     MOV AX, SSEG
39     MOV SS, AX
40     MOV SP, SIZE STKTOP
41     MOV DX, PX      ;取X
42     ADD DX, DX      ;计算2X
43     PUSH DX          ;NOTE:
44     CALL SQR00T3    ;调用开平方子程序
45     MOV DX, PY      ;取Y
46     MOV AX, DX      ;计算3Y
47     ADD DX, DX
48     ADD DX, AX
49     PUSH DX          ;NOTE:
50     CALL SQR00T3    ;调用开平方子程序
51     POP DX          ;NOTE:
52     POP AX          ;NOTE:
53     ADD AX, DX      ;计算 $\sqrt{2}X + \sqrt{3}Y$ 
54     PUSH AX          ;暂存结果
55     MOV DX, 150
56     PUSH DX          ;NOTE:
57     CALL SQR00T3
58     POP DX          ;取出中间结果
59     POP AX
60     ADD AX, DX      ;计算最终结果
61     MOV RLT, AX      ;保存结果
62     MOV AH, 4CH
63     INT 21H
64 CSEG ENDS
65 END MAIN1

```

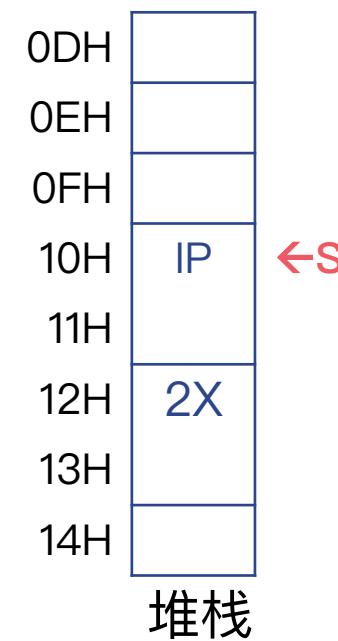
→

```

16 SQR00T3 PROC NEAR
17     INC SP
18     INC SP
19     POP DX
20     XOR AX, AX      ;i←0
21     AND DX, DX      ;测试被开方数
22     JZ SQRT2        ;被开方数为0
23 SQRT1: MOV BX, AX  ;形成奇数
24     SHL BX, 1
25     INC BX
26     SUB DX, BX      ;被开方数减去奇数
27     JC SQRT2        ;不够减
28     INC AX          ;够减, i增1
29     JMP SQRT1       ;继续
30 SQRT2: PUSH AX      ;DX←平方根
31     DEC SP
32     DEC SP
33     RET
34 SQR00T3 ENDP

```

;返回



```

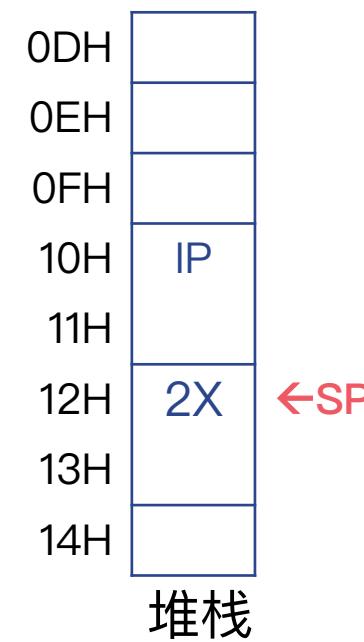
36 MAIN1: MOV AX, DSEG
37     MOV DS, AX
38     MOV AX, SSEG
39     MOV SS, AX
40     MOV SP, SIZE STKTOP
41     MOV DX, PX      ;取X
42     ADD DX, DX      ;计算2X
43     PUSH DX          ;NOTE:
44     CALL SQR00T3    ;调用开平方子程序
45     MOV DX, PY      ;取Y
46     MOV AX, DX      ;计算3Y
47     ADD DX, DX
48     ADD DX, AX
49     PUSH DX          ;NOTE:
50     CALL SQR00T3    ;调用开平方子程序
51     POP DX          ;NOTE:
52     POP AX          ;NOTE:
53     ADD AX, DX      ;计算 $\sqrt{2}X + \sqrt{3}Y$ 
54     PUSH AX          ;暂存结果
55     MOV DX, 150
56     PUSH DX          ;NOTE:
57     CALL SQR00T3
58     POP DX          ;取出中间结果
59     POP AX
60     ADD AX, DX      ;计算最终结果
61     MOV RLT, AX      ;保存结果
62     MOV AH, 4CH
63     INT 21H
64 CSEG ENDS
65 END MAIN1

```

```

16 SQR00T3 PROC NEAR
17     INC SP
18     INC SP
19     POP DX
20     XOR AX, AX      ;i←0
21     AND DX, DX      ;测试被开方数
22     JZ SQRT2        ;被开方数为0
23     SQRT1: MOV BX, AX ;形成奇数
24     SHL BX, 1
25     INC BX
26     SUB DX, BX      ;被开方数减去奇数
27     JC SQRT2        ;不够减
28     INC AX          ;够减, i增1
29     JMP SQRT1       ;继续
30     SQRT2: PUSH AX  ;DX←平方根
31     DEC SP
32     DEC SP
33     RET
34 SQR00T3 ENDP

```



```

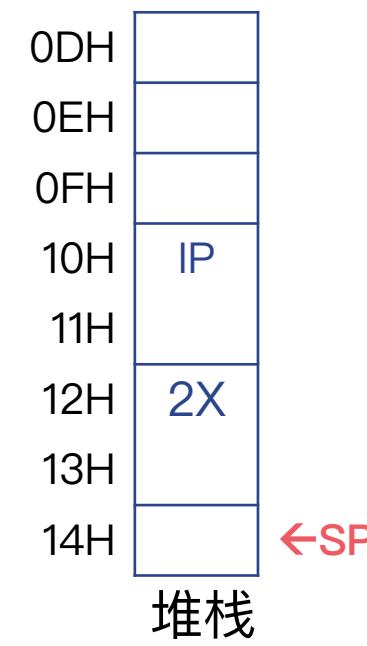
36 MAIN1: MOV AX, DSEG
37     MOV DS, AX
38     MOV AX, SSEG
39     MOV SS, AX
40     MOV SP, SIZE STKTOP
41     MOV DX, PX      ;取X
42     ADD DX, DX      ;计算2X
43     PUSH DX          ;NOTE:
44     CALL SQR00T3    ;调用开平方子程序
45     MOV DX, PY      ;取Y
46     MOV AX, DX      ;计算3Y
47     ADD DX, DX
48     ADD DX, AX
49     PUSH DX          ;NOTE:
50     CALL SQR00T3    ;调用开平方子程序
51     POP DX          ;NOTE:
52     POP AX          ;NOTE:
53     ADD AX, DX      ;计算 $\sqrt{2}X + \sqrt{3}Y$ 
54     PUSH AX          ;暂存结果
55     MOV DX, 150
56     PUSH DX          ;NOTE:
57     CALL SQR00T3
58     POP DX          ;取出中间结果
59     POP AX
60     ADD AX, DX      ;计算最终结果
61     MOV RLT, AX      ;保存结果
62     MOV AH, 4CH
63     INT 21H
64 CSEG ENDS
65 END MAIN1

```

```

16 SQR00T3 PROC NEAR
17     INC SP
18     INC SP
19     POP DX
20     XOR AX, AX      ;i←0
21     AND DX, DX      ;测试被开方数
22     JZ SQRT2        ;被开方数为0
23     SQRT1: MOV BX, AX ;形成奇数
24     SHL BX, 1
25     INC BX
26     SUB DX, BX      ;被开方数减去奇数
27     JC SQRT2        ;不够减
28     INC AX          ;够减,i增1
29     JMP SQRT1       ;继续
30     SQRT2: PUSH AX  ;DX←平方根
31     DEC SP
32     DEC SP
33     RET
34 SQR00T3 ENDP

```



```

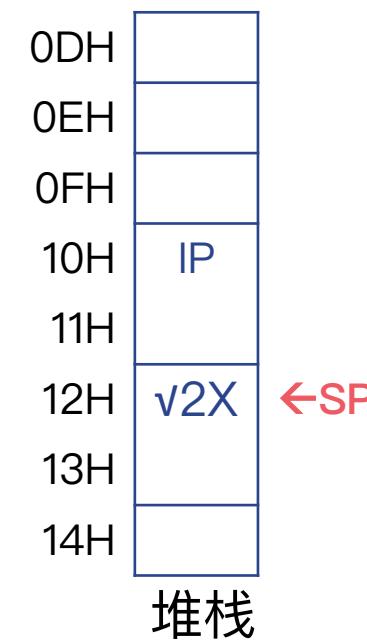
36 MAIN1: MOV AX, DSEG
37     MOV DS, AX
38     MOV AX, SSEG
39     MOV SS, AX
40     MOV SP, SIZE STKTOP
41     MOV DX, PX      ;取X
42     ADD DX, DX      ;计算2X
43     PUSH DX          ;NOTE:
44     CALL SQR00T3    ;调用开平方子程序
45     MOV DX, PY      ;取Y
46     MOV AX, DX      ;计算3Y
47     ADD DX, DX
48     ADD DX, AX
49     PUSH DX          ;NOTE:
50     CALL SQR00T3    ;调用开平方子程序
51     POP DX          ;NOTE:
52     POP AX          ;NOTE:
53     ADD AX, DX      ;计算 $\sqrt{2}X + \sqrt{3}Y$ 
54     PUSH AX          ;暂存结果
55     MOV DX, 150
56     PUSH DX          ;NOTE:
57     CALL SQR00T3
58     POP DX          ;取出中间结果
59     POP AX
60     ADD AX, DX      ;计算最终结果
61     MOV RLT, AX      ;保存结果
62     MOV AH, 4CH
63     INT 21H
64 CSEG ENDS
65 END MAIN1

```

```

16 SQR00T3 PROC NEAR
17     INC SP
18     INC SP
19     POP DX
20     XOR AX, AX      ;i←0
21     AND DX, DX      ;测试被开方数
22     JZ SQRT2        ;被开方数为0
23     SQRT1: MOV BX, AX ;形成奇数
24     SHL BX, 1
25     INC BX
26     SUB DX, BX      ;被开方数减去奇数
27     JC SQRT2        ;不够减
28     INC AX          ;够减, i增1
29     JMP SQRT1       ;继续
30     SQRT2: PUSH AX  ;DX←平方根
31     DEC SP
32     DEC SP
33     RET
34 SQR00T3 ENDP

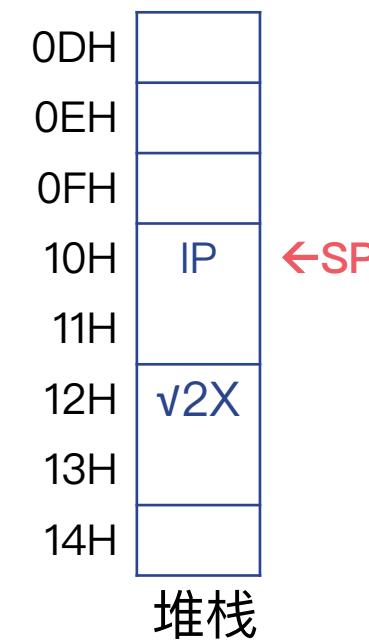
```



```

36 MAIN1: MOV AX, DSEG
37     MOV DS, AX
38     MOV AX, SSEG
39     MOV SS, AX
40     MOV SP, SIZE STKTOP
41     MOV DX, PX      ;取X
42     ADD DX, DX      ;计算2X
43     PUSH DX          ;NOTE:
44     CALL SQR00T3    ;调用开平方子程序
45     MOV DX, PY      ;取Y
46     MOV AX, DX      ;计算3Y
47     ADD DX, DX
48     ADD DX, AX
49     PUSH DX          ;NOTE:
50     CALL SQR00T3    ;调用开平方子程序
51     POP DX          ;NOTE:
52     POP AX          ;NOTE:
53     ADD AX, DX      ;计算 $\sqrt{2}X + \sqrt{3}Y$ 
54     PUSH AX          ;暂存结果
55     MOV DX, 150
56     PUSH DX          ;NOTE:
57     CALL SQR00T3
58     POP DX          ;取出中间结果
59     POP AX
60     ADD AX, DX      ;计算最终结果
61     MOV RLT, AX      ;保存结果
62     MOV AH, 4CH
63     INT 21H
64 CSEG ENDS
65 END MAIN1

```



```

16 SQR00T3 PROC NEAR
17     INC SP
18     INC SP
19     POP DX
20     XOR AX, AX      ;i←0
21     AND DX, DX      ;测试被开方数
22     JZ SQRT2        ;被开方数为0
23 SQRT1: MOV BX, AX  ;形成奇数
24     SHL BX, 1
25     INC BX
26     SUB DX, BX      ;被开方数减去奇数
27     JC SQRT2        ;不够减
28     INC AX          ;够减, i增1
29     JMP SQRT1       ;继续
30 SQRT2: PUSH AX     ;DX←平方根
31     DEC SP
32     DEC SP
33     RET
34 SQR00T3 ENDP

```

```

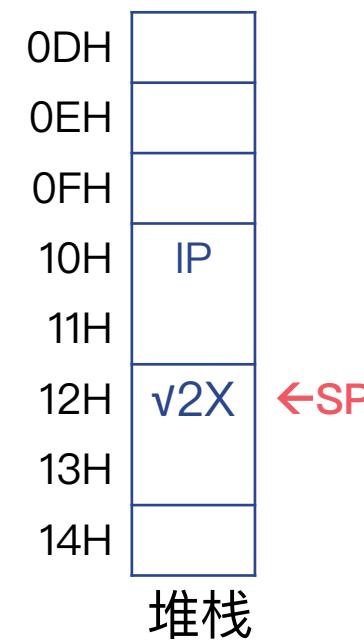
36 MAIN1: MOV AX, DSEG
37     MOV DS, AX
38     MOV AX, SSEG
39     MOV SS, AX
40     MOV SP, SIZE STKTOP
41     MOV DX, PX      ;取X
42     ADD DX, DX      ;计算2X
43     PUSH DX          ;NOTE:
44     CALL SQR00T3      ;调用开平方子程序
45     →MOV DX, PY      ;取Y
46     MOV AX, DX      ;计算3Y
47     ADD DX, DX
48     ADD DX, AX
49     PUSH DX          ;NOTE:
50     CALL SQR00T3      ;调用开平方子程序
51     POP DX          ;NOTE:
52     POP AX          ;NOTE:
53     ADD AX, DX      ;计算 $\sqrt{2}X + \sqrt{3}Y$ 
54     PUSH AX          ;暂存结果
55     MOV DX, 150
56     PUSH DX          ;NOTE:
57     CALL SQR00T3
58     POP DX          ;取出中间结果
59     POP AX
60     ADD AX, DX      ;计算最终结果
61     MOV RLT, AX      ;保存结果
62     MOV AH, 4CH
63     INT 21H
64 CSEG ENDS
65 END MAIN1

```

```

16 SQR00T3 PROC NEAR
17     INC SP
18     INC SP
19     POP DX
20     XOR AX, AX      ;i←0
21     AND DX, DX      ;测试被开方数
22     JZ SQRT2        ;被开方数为0
23     SQRT1: MOV BX, AX ;形成奇数
24     SHL BX, 1
25     INC BX
26     SUB DX, BX      ;被开方数减去奇数
27     JC SQRT2        ;不够减
28     INC AX          ;够减, i增1
29     JMP SQRT1        ;继续
30     SQRT2: PUSH AX    ;DX←平方根
31     DEC SP
32     DEC SP
33     RET
34 SQR00T3 ENDP

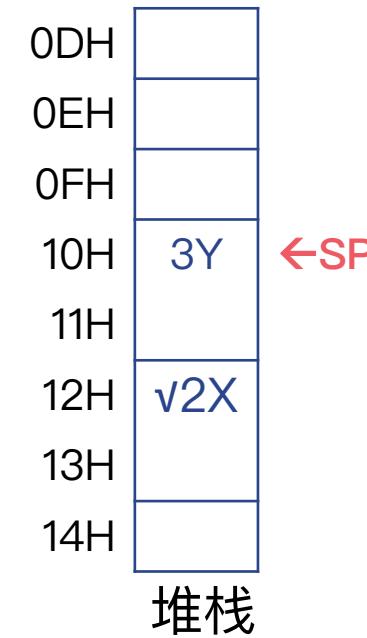
```



```

36 MAIN1: MOV AX, DSEG
37     MOV DS, AX
38     MOV AX, SSEG
39     MOV SS, AX
40     MOV SP, SIZE STKTOP
41     MOV DX, PX      ;取X
42     ADD DX, DX      ;计算2X
43     PUSH DX          ;NOTE:
44     CALL SQR00T3      ;调用开平方子程序
45     MOV DX, PY      ;取Y
46     MOV AX, DX      ;计算3Y
47     ADD DX, DX
48     ADD DX, AX
49     PUSH DX          ;NOTE:
50     → CALL SQR00T3      ;调用开平方子程序
51     POP DX          ;NOTE:
52     POP AX          ;NOTE:
53     ADD AX, DX      ;计算 $\sqrt{2}X + \sqrt{3}Y$ 
54     PUSH AX          ;暂存结果
55     MOV DX, 150
56     PUSH DX          ;NOTE:
57     CALL SQR00T3
58     POP DX          ;取出中间结果
59     POP AX
60     ADD AX, DX      ;计算最终结果
61     MOV RLT, AX      ;保存结果
62     MOV AH, 4CH
63     INT 21H
64 CSEG ENDS
65 END MAIN1

```



```

16 SQR00T3 PROC NEAR
17     INC SP
18     INC SP
19     POP DX
20     XOR AX, AX      ;i←0
21     AND DX, DX      ;测试被开方数
22     JZ SQRT2        ;被开方数为0
23     SQRT1: MOV BX, AX ;形成奇数
24     SHL BX, 1
25     INC BX
26     SUB DX, BX      ;被开方数减去奇数
27     JC SQRT2        ;不够减
28     INC AX          ;够减, i增1
29     JMP SQRT1       ;继续
30     SQRT2: PUSH AX  ;DX←平方根
31     DEC SP
32     DEC SP
33     RET
34 SQR00T3 ENDP

```

```

36 MAIN1: MOV AX, DSEG
37     MOV DS, AX
38     MOV AX, SSEG
39     MOV SS, AX
40     MOV SP, SIZE STKTOP
41     MOV DX, PX      ;取X
42     ADD DX, DX      ;计算2X
43     PUSH DX          ;NOTE:
44     CALL SQR00T3    ;调用开平方子程序
45     MOV DX, PY      ;取Y
46     MOV AX, DX      ;计算3Y
47     ADD DX, DX
48     ADD DX, AX
49     PUSH DX          ;NOTE:
50     CALL SQR00T3    ;调用开平方子程序
51     POP DX          ;NOTE:
52     POP AX          ;NOTE:
53     ADD AX, DX      ;计算 $\sqrt{2}X + \sqrt{3}Y$ 
54     PUSH AX          ;暂存结果
55     MOV DX, 150
56     PUSH DX          ;NOTE:
57     CALL SQR00T3
58     POP DX          ;取出中间结果
59     POP AX
60     ADD AX, DX      ;计算最终结果
61     MOV RLT, AX      ;保存结果
62     MOV AH, 4CH
63     INT 21H
64 CSEG ENDS
65 END MAIN1

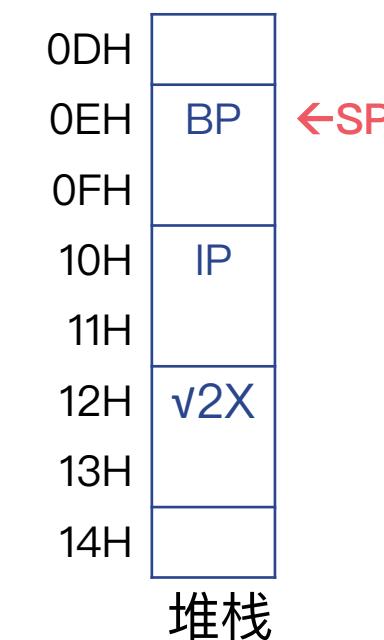
```

| | |
|------|------------|
| PUSH | BP |
| MOV | BP, SP |
| MOV | DX, [BP+4] |

```

16 SQR00T3 PROC NEAR
17     INC SP
18     INC SP
19     POP DX
20     XOR AX, AX      ;i←0
21     AND DX, DX
22     JZ SQRT2        ;测试被开方数
23     SQRT1: MOV BX, AX ;形成奇数
24     SHL BX, 1
25     INC BX
26     SUB DX, BX      ;被开方数减去奇数
27     JC SQRT2        ;不够减
28     INC AX          ;够减, i增1
29     JMP SQRT1       ;继续
30     SQRT2: PUSH AX  ;DX←平方根
31     DEC SP
32     DEC SP
33     RET
34 SQR00T3 ENDP

```



四. 子程序设计方法

► 3. 主程序与子程序之间的参数传递 — 约定参数地址指针法

例：设内存有三组无符号字整数，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。 。

当要**传送的参数较多**时，用约定地址指针法，即调用程序将参数存放的地址送入约定的地址指针，然后调用子程序，子程序从地址指针指出的地址取出所需参数。当返回参数较多时，可存入内存某区域，然后把其首地址指针放入约定的寄存器中，供调用程序使用。

四. 子程序设计方法

3. 主程序与子程序之间的参数传递 — 约定参数地址指针法

```
2  SSEG  SEGMENT STACK
3  STKTOP  DB      40 DUP(0)
4  SSEG  ENDS
5
6  DSEG  SEGMENT
7  LIST1  DW      35, 27, 165, 3, 1825, 603
8  CNT1  DW      6
9  LIST2  DW      438, 121, 496, 6321, 28, 17, 2, 105
10 CNT2  DW      8
11 LIST3  DW      18, 5, 19, 46, 3725
12 CNT3  DW      5
13 SUM   DW      0
14 DSEG  ENDS
15
16 CSEG  SEGMENT
17     ASSUME CS:CSEG, DS:DSEG
18     ASSUME SS:SSEG
19
20 > FMIN  PROC  NEAR           ;求数组中最小值...
34
35 MAIN:  MOV    AX, DSEG
36     MOV    DS, AX
37     MOV    AX, SSEG
38     MOV    SS, AX
39     MOV    SP, SIZE STKTOP
```

四. 子程序设计方法

3. 主程序与子程序之间的参数传递 — 约定参数地址指针法

```
40      MOV      SI, OFFSET LIST1  
41      MOV      CX, CNT1  
42      CALL     FMIN  
43      MOV      BX, AX  
44      MOV      SI, OFFSET LIST2  
45      MOV      CX, CNT2  
46      CALL     FMIN  
47      ADD      BX, AX  
48      MOV      SI, OFFSET LIST3  
49      MOV      CX, CNT3  
50      CALL     FMIN  
51      ADD      BX, AX  
52      MOV      SUM, BX  
53      MOV      AH, 4CH  
54      INT      21H  
55      CSEG    ENDS  
56      END      MAIN
```

| | | | | |
|----|---------|------|----------|--------------|
| 20 | FMIN | PROC | NEAR | ;求数组中最小值 |
| 21 | | PUSH | SI | ;保存数组首址 |
| 22 | | MOV | AX, [SI] | ;取第一个数 |
| 23 | | DEC | CX | ;计数值减1 |
| 24 | | JZ | RETURN | ;计数值为0, 转 |
| 25 | FMIN2: | INC | SI | ;修改数组指针 |
| 26 | | INC | SI | |
| 27 | | CMP | AX, [SI] | ;与下一个数比较 |
| 28 | | JB | FMIN1 | ;AX中数小, 转 |
| 29 | | MOV | AX, [SI] | ;小数取入AX |
| 30 | FMIN1: | LOOP | FMIN2 | ;计数值减1, 不为0转 |
| 31 | RETURN: | POP | SI | ;恢复数组首值 |
| 32 | | RET | | ;返回 |
| 33 | FMIN | ENDP | | |

四. 子程序设计方法（示例）

► 例子：计算三组字数据中正数、负数和零的个数，并分别存入PCOUNT，MCOUNT和ZCOUNT单元。设三组数据首地址分别ARRY1，ARRY2和ARRY3，数据个数分别在CNT1，CNT2和CNT3单元存放。

子程序说明文件例子：

- 1) 子程序名（入口标号）：PMZN；
- 2) 子程序功能：求一组数据中正数、负数和零的个数；
- 3) 入口条件：数组首地址在SI中；数组数据个数在CX中；
- 4) 出口条件：正数个数在AX中；负数个数在BX中；零的个数在DX中；
- 5) 受影响的寄存器：AX，BX，DX和标志寄存器F。

四. 子程序设计方法（示例）

例子：计算三组字数据中正数、负数和零的个数，并分别累计存入PCOUNT，MCOUNT和ZCOUNT单元。设三组数据首地址分别ARRY1，ARRY2和ARRY3，数据个数分别在CNT1，CNT2和CNT3单元存放。

子程序说明文件例子：

- 1) 子程序名（入口标号）：PMZN；
- 2) 子程序功能：求一组数据中正数、负数和零的个数；
- 3) 入口条件：数组首地址在SI中；数组数据个数在CX中；
- 4) 出口条件：正数个数在AX中；负数个数在BX中；零的个数在DX中；
- 5) 受影响的寄存器：AX，BX，DX和标志寄存器F。

```
33  DSEG  SEGMENT
34  ARRY1  DW    15,-5,1,5,0,123,964,-327,0
35  CNT1   DW    9
36  ARRY2  DW    103,4,-8,-23,0,827,-936,0,0,18
37  CNT2   DW    10
38  ARRY3  DW    -29,-137,-23,0,4,0
39  CNT3   DW    6
40  PCOUNT  DW    0                                ;保存结果单元
41  MCOUNT  DW    0
42  ZCOUNT  DW    0
43  ADR    DW    OFFSET ARRY1, OFFSET CNT1
44  DW      DW    OFFSET ARRY2, OFFSET CNT2
45  DW      DW    OFFSET ARRY3, OFFSET CNT3
46  DSEG   ENDS
47
48  CSEG   SEGMENT
49  ASSUME CS:CSEG, DS:DSEG
50  ASSUME SS:SSEG
51  START: MOV  AX, DSEG
52  MOV  DS, AX
53  MOV  AX, SSEG
54  MOV  SS, AX
55  MOV  SP, LENGTH SKTOP
```

```

56      LEA     DI, ADR
57      MOV     CX, 03
58 AGAIN: MOV    SI, [DI]      ;取数组首址→SI
59      MOV    BX, [DI+2]    ;取数组数据个数首址
60      PUSH   CX          ;保存CX中的循环计数值
61      MOV    CX, [BX]      ;取数组数据个数→CX
62      CALL   FAR PTR PMZN
63      ADD    PCOUNT, AX   ;累加结果
64      ADD    MCOUNT, BX
65      ADD    ZCOUNT, DX
66      ADD    DI, 4        ;修改指针指向下一数组
67      POP    CX          ;恢复CX中循环计数
68      LOOP  AGAIN        ;CX计数值减1,非0转
69      MOV    AH, 4CH
70      INT    21H
71 CSEG  END
72      END    START

```

| | | | |
|----|--------|--------------------------|-------------|
| 2 | | PUBLIC | PMZN |
| 3 | CSEG | SEGMENT | |
| 4 | | ASSUME | CS:CSEG |
| 5 | PMZN | PROC | FAR |
| 6 | | PUSH | SI |
| 7 | | PUSH | CX |
| 8 | | XOR | AX,AX |
| 9 | | XOR | BX,BX |
| 10 | | XOR | DX,DX |
| 11 | PMZN0: | TEST WORD PTR[SI],0FFFFH | ;测试数据 |
| 12 | | JS MINUS | ;负转 |
| 13 | | JNZ PLUS | ;非0转 |
| 14 | | INC DX | ;为0,0计数器加1 |
| 15 | | JMP PMZN1 | |
| 16 | PLUS: | INC AX | ;正数计数器加1 |
| 17 | | JMP PMZN1 | |
| 18 | MINUS: | INC BX | ;负数计数器加1 |
| 19 | PMZN1: | ADD SI,2 | ;指向下一个数据 |
| 20 | | LOOP PMZN0 | ;循环计数减1,非0转 |
| 21 | | POP CX | ;恢复CX |
| 22 | | POP SI | ;恢复SI |
| 23 | | RET | |
| 24 | PMZN | ENDP | |
| 25 | CSEG | ENDS | |
| 26 | | END | |

五. 子程序嵌套

► 我们编制子程序时，如果用到已有的某种功能的子程序，也可以采用调用的方法，在一个子程序的内部再调用其它子程序，称为子程序嵌套。这样的子程序称为嵌套子程序。嵌套子程序的层数称为嵌套深度，一般设有硬件堆栈的计算机，嵌套深度只受堆栈区大小的限制。

假设我们已经有一个子程序，其说明文件和程序清单如下：

- (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F。

| | | | |
|---|--------|------|---------|
| 2 | HTOA | PROC | NEAR |
| 3 | | AND | AL, 0FH |
| 4 | | CMP | AL, 10 |
| 5 | | JC | HTOA1 |
| 6 | | ADD | AL, 07 |
| 7 | HTOA1: | ADD | AL, 30H |
| 8 | | RET | |
| 9 | HTOA | ENDP | |

| | | | | | |
|----|----------|---|----|----------|---|
| 31 | 00110001 | 1 | 41 | 01000001 | A |
| 32 | 00110010 | 2 | 42 | 01000010 | B |
| 33 | 00110011 | 3 | 43 | 01000011 | C |
| 34 | 00110100 | 4 | 44 | 01000100 | D |
| 35 | 00110101 | 5 | | | |

五. 子程序嵌套

► 我们编制子程序时，如果用到已有的某种功能的子程序，也可以采用调用的方法，在一个子程序的内部再调用其它子程序，称为子程序嵌套。这样的子程序称为嵌套子程序。嵌套子程序的层数称为嵌套深度，一般设有硬件堆栈的计算机，嵌套深度只受堆栈区大小的限制。

- (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：两位十六进制数在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F。

| | | | |
|----|-------|------|--------|
| 12 | BHTOA | PROC | |
| 13 | | PUSH | CX |
| 14 | | MOV | CH, AL |
| 15 | | MOV | CL, 04 |
| 16 | | SHR | AL, CL |
| 17 | | CALL | HTOA |
| 18 | | MOV | AH, AL |
| 19 | | MOV | AL, CH |
| 20 | | CALL | HTOA |
| 21 | | POP | CX |
| 22 | | RET | |
| 23 | BHTOA | ENDP | |

五. 子程序嵌套

|> 例子：假定在内存DATA单元存放着一个无符号的字节数据，将它在屏幕上显示出来。

```
29  CSEG    SEGMENT  
30      ASSUME CS:CSEG, DS:DSEG  
31  START: MOV     AX, DSEG  
32      MOV     DS, AX  
33      MOV     AL, DATA  
34      CALL    BHTOA  
35      PUSH    AX  
36      MOV     DL, AH  
37      MOV     AH, 2  
38      INT     21H  
39      POP     DX  
40      INT     21H  
41      MOV     AX, 4C00H  
42      INT     21H  
43  CSEG    ENDS  
44  END     START
```

六. 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。递归子程序一般用于解递归函数，递归函数定义具有下述两个组成部分：

1. 递归函数的递归过程必须是有限的。即每个递归函数有一个非递归终值，递归过程(子程序)一旦求得该值，就结束递归。
2. 任一函数项的值都有明确的定义，其值可由函数中的一项或几项值求得。

例子：阶乘函数，对于任一个大于等于0的正整数N，其函数值定义为：

$$FACT(N) = \begin{cases} 1 & (N = 0) \\ N * FACT(N-1) & (N > 0) \end{cases}$$

六. 递归子程序

算法：

- (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。

| | | | |
|----|--------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

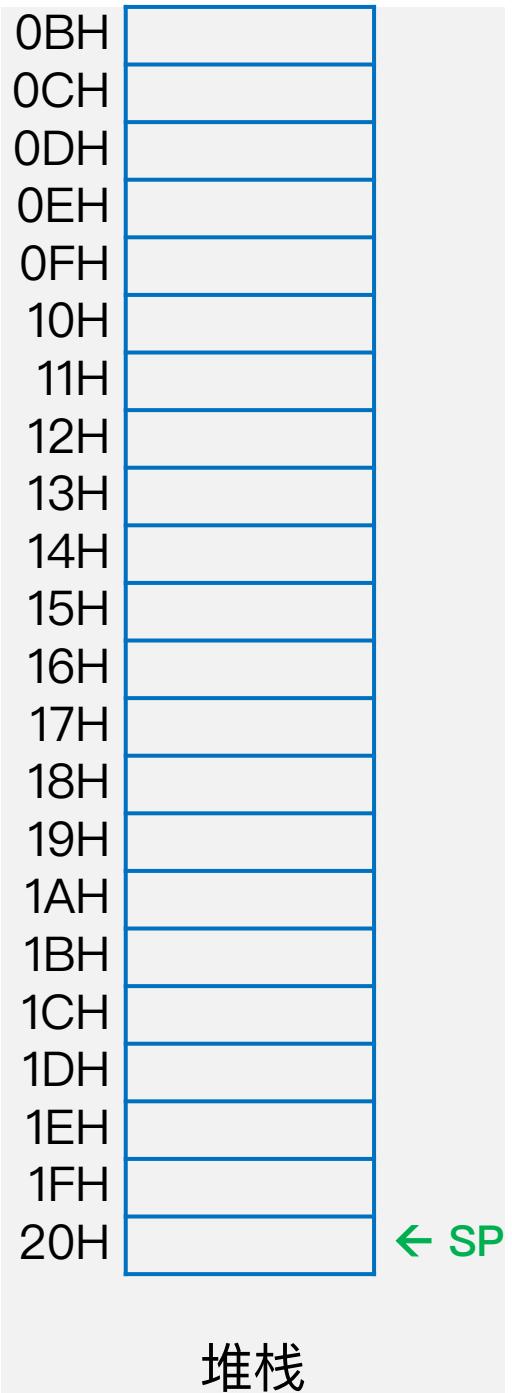
六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24  ←       MOV      BX, 4
25          CALL     FACT
26  xxyy   NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG    ENDS
30 END     MAIN

```

BX = ?
AX = ?
SP = 20H



| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

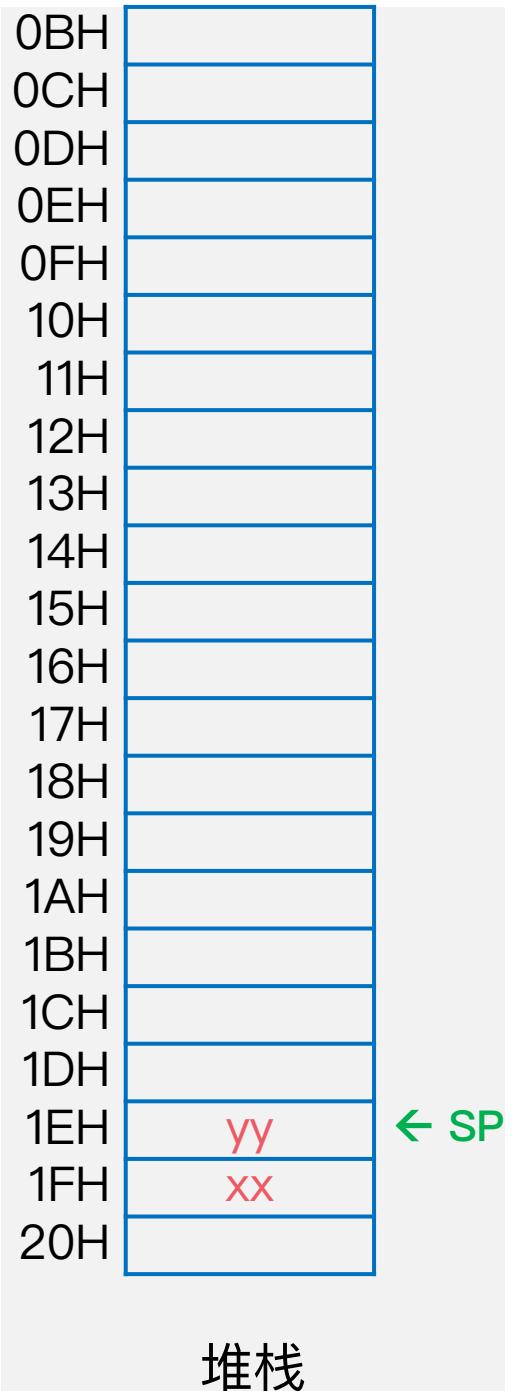
六. 递归子程序 — 例：求4!

```

21 MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25          CALL     FACT
26          xxyy NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG      ENDS
30 END      MAIN

```

BX = 4
AX = ?
SP = 1EH



| | | | |
|----|----------------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: <u>aabb</u> | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

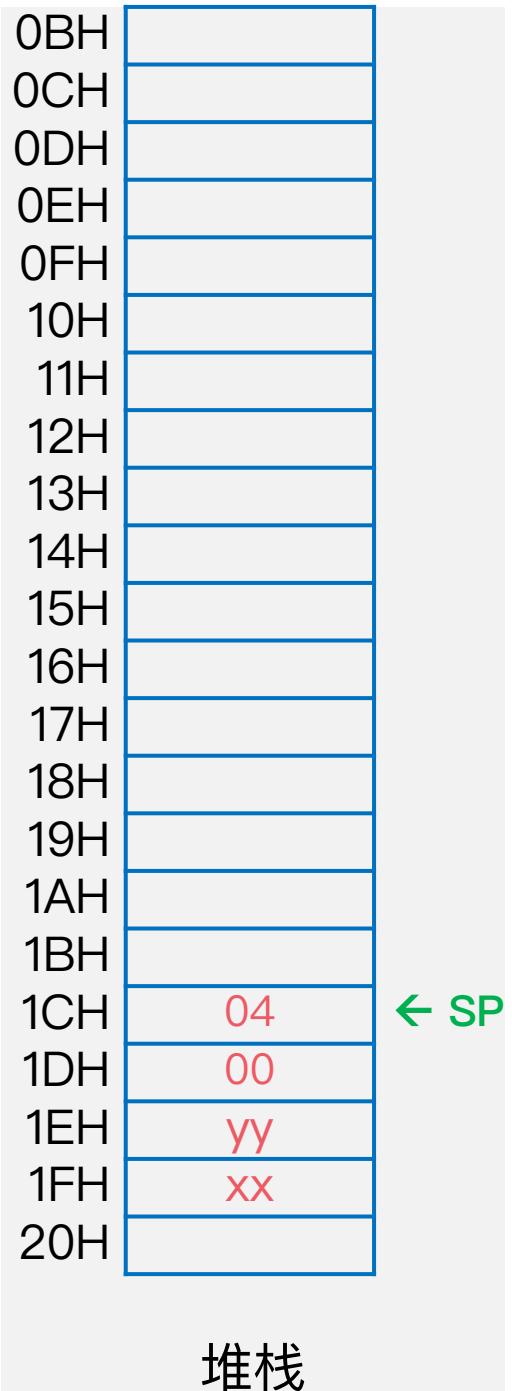
六. 递归子程序 — 例：求4!

```

21 MAIN:    MOV      AX, SSEG
22           MOV      DS, AX
23           MOV      SP, SIZE SKTOP
24           MOV      BX, 4
25     →     CALL     FACT
26     xxyy  NOP
27           MOV      AH, 4CH
28           INT     21H
29 CSEG      ENDS
30 END      MAIN

```

BX = 4
AX = ?
SP = 1CH



| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

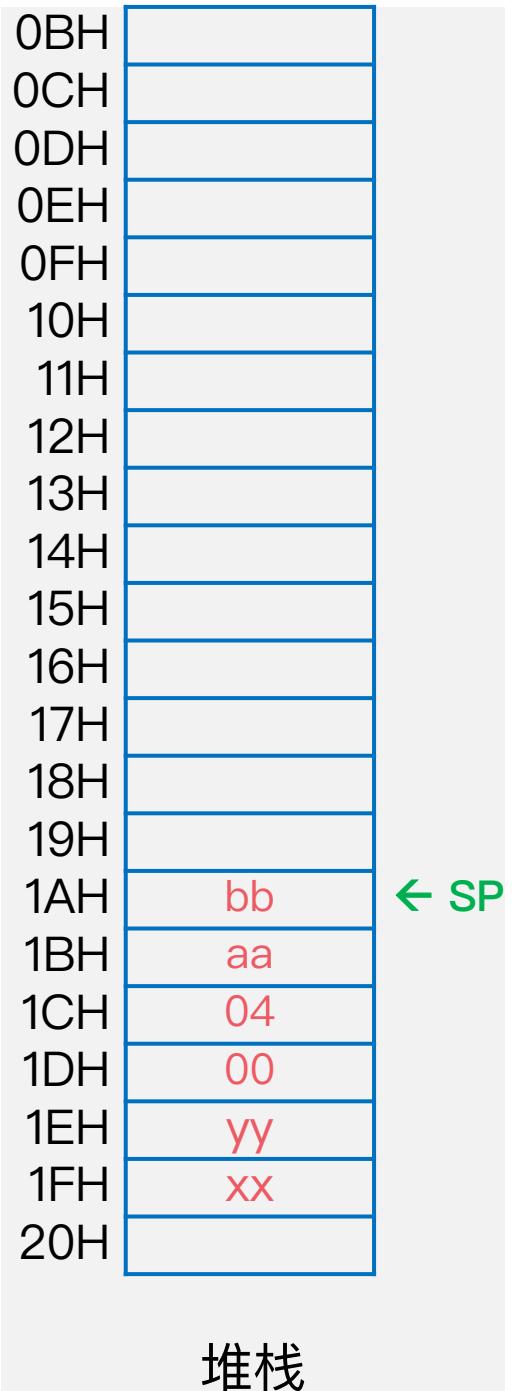
六. 递归子程序 — 例：求4!

```

21 MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25          CALL     FACT
26          xxyy NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG      ENDS
30 END      MAIN

```

BX = 3
AX = ?
SP = 1AH



| | | | |
|----|----------------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: <u>aabb</u> | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

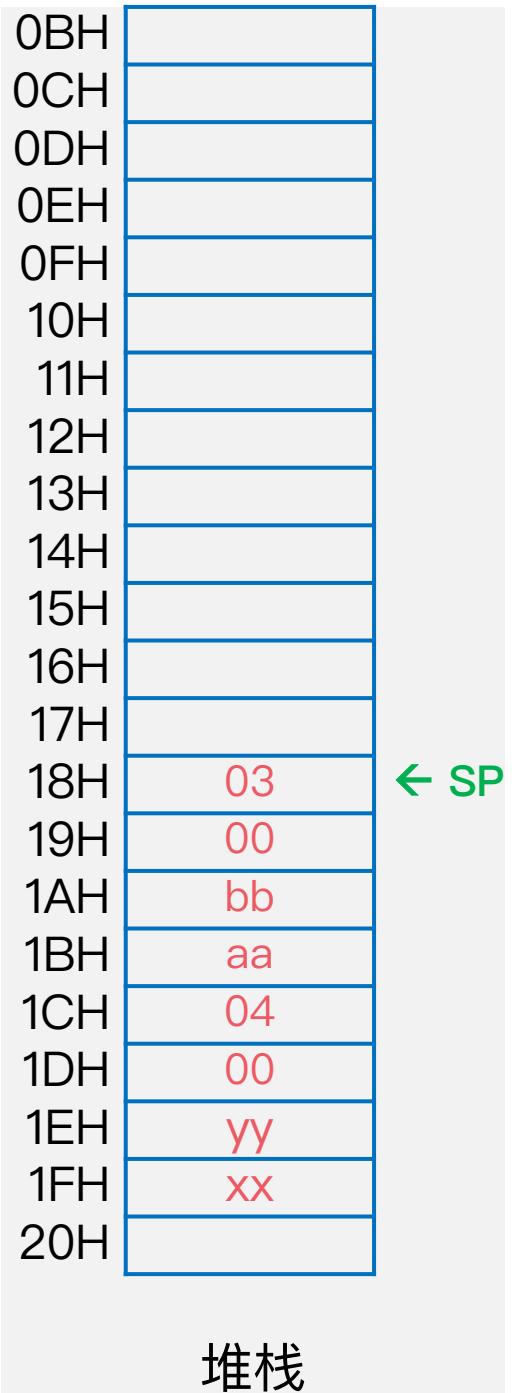
六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25  →       CALL     FACT
26  xxyy   NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG    ENDS
30 END     MAIN

```

BX = 3
AX = ?
SP = 18H



| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

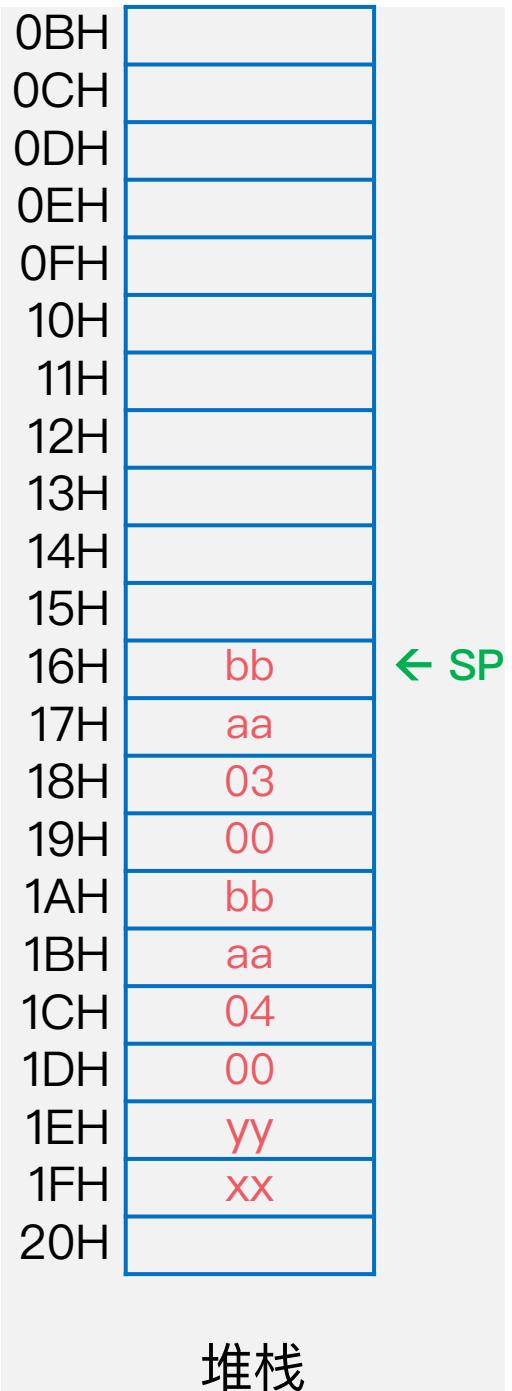
六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25  →       CALL     FACT
26  xxyy   NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG    ENDS
30 END     MAIN

```

BX = 2
AX = ?
SP = 16H



| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

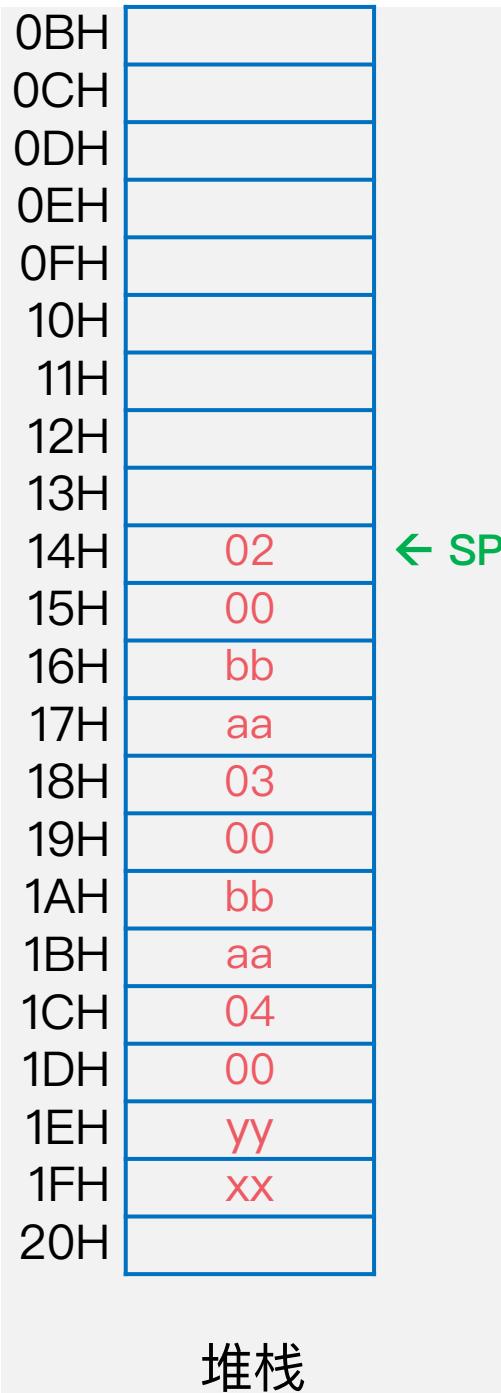
六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25  →       CALL     FACT
26  xxyy   NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG    ENDS
30 END     MAIN

```

BX = 2
AX = ?
SP = 14H



| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

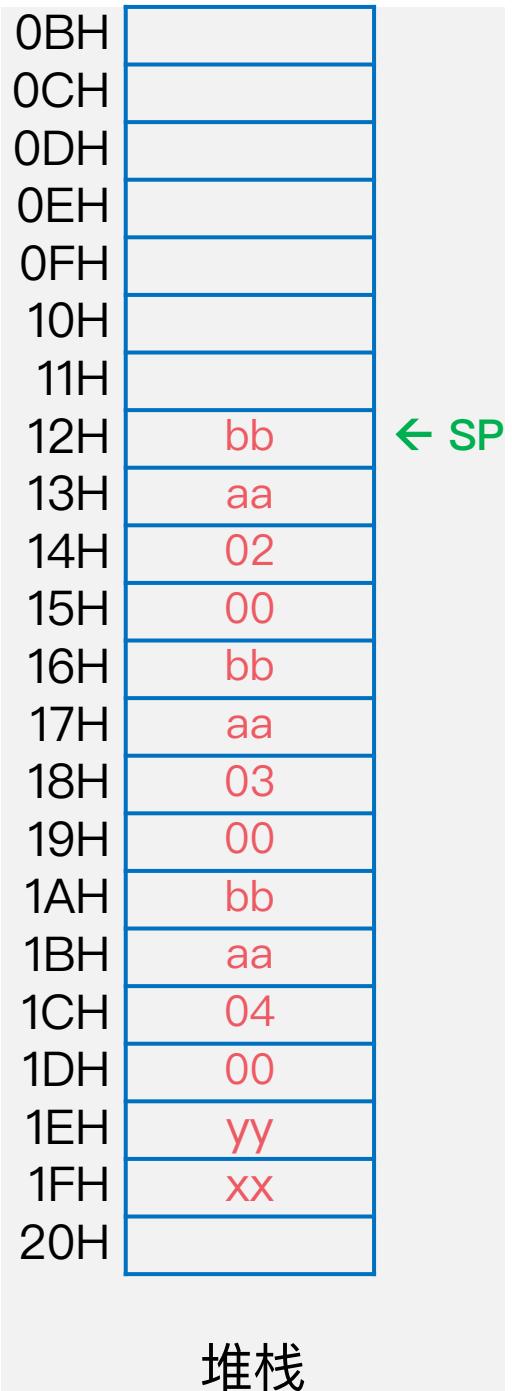
六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25  ←       CALL     FACT
26  xxyy   NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG    ENDS
30 END     MAIN

```

BX = 1
AX = ?
SP = 12H



| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25  ←       CALL     FACT
26  xxyy   NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG    ENDS
30 END     MAIN

```

BX = 1
AX = ?
SP = 10H

| | |
|-----|----|
| 0BH | |
| 0CH | |
| 0DH | |
| 0EH | |
| 0FH | |
| 10H | 01 |
| 11H | 00 |
| 12H | bb |
| 13H | aa |
| 14H | 02 |
| 15H | 00 |
| 16H | bb |
| 17H | aa |
| 18H | 03 |
| 19H | 00 |
| 1AH | bb |
| 1BH | aa |
| 1CH | 04 |
| 1DH | 00 |
| 1EH | yy |
| 1FH | xx |
| 20H | |

← SP

| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

堆栈

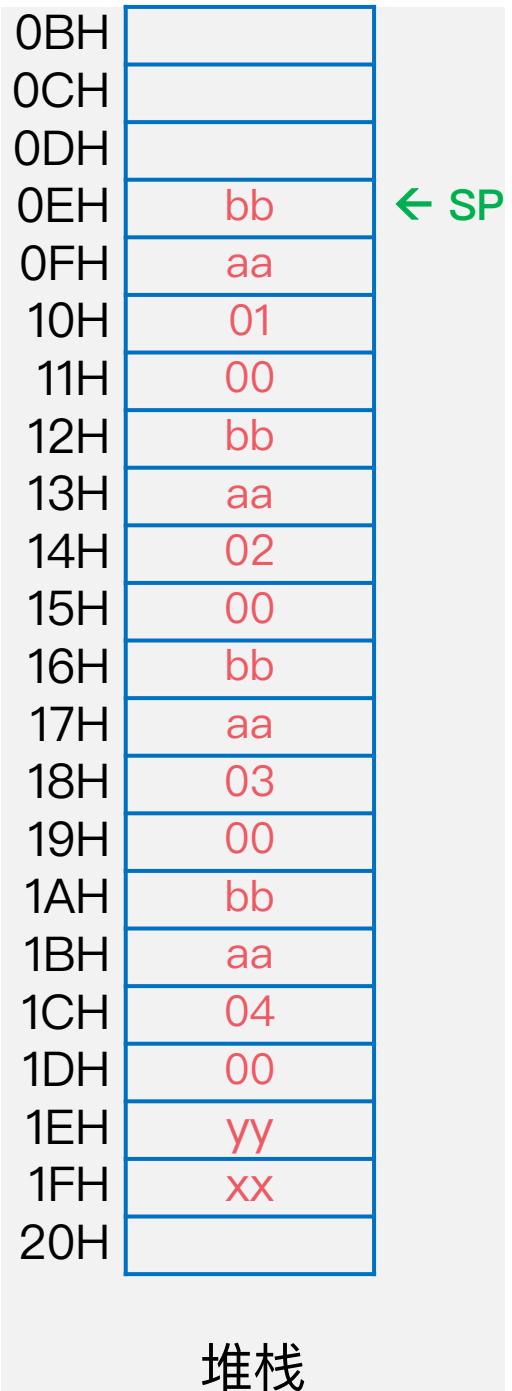
六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25  →       CALL     FACT
26  xxyy   NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG    ENDS
30 END     MAIN

```

BX = 0
AX = ?
SP = 0EH



| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

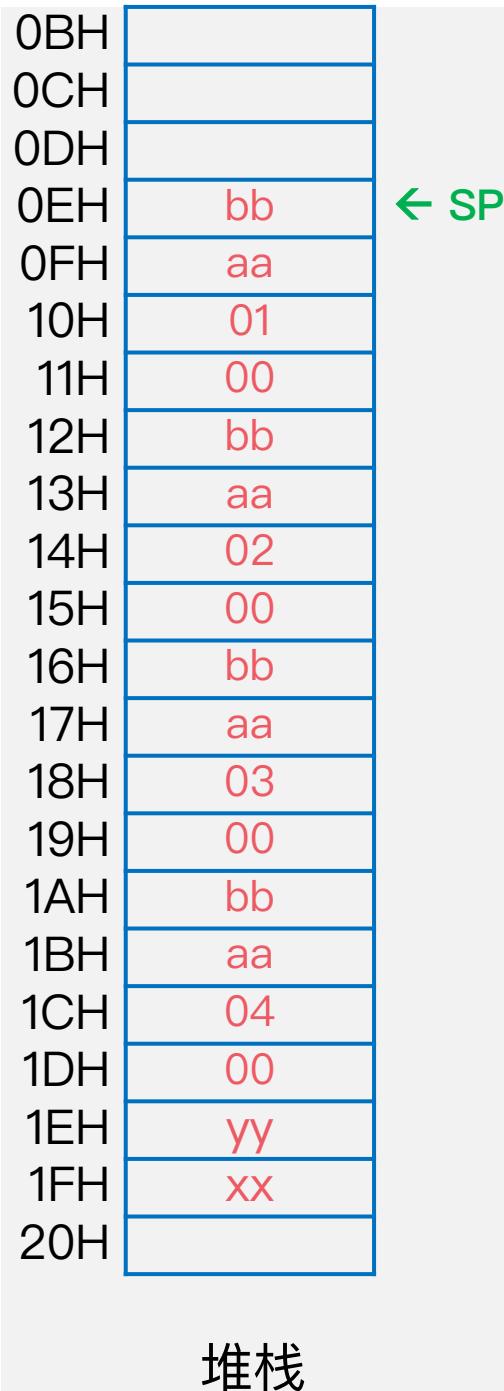
六. 递归子程序 — 例：求4!

```

21 MAIN:    MOV      AX, SSEG
22           MOV      DS, AX
23           MOV      SP, SIZE SKTOP
24           MOV      BX, 4
25     →     CALL     FACT
26     xxyy  NOP
27           MOV      AH, 4CH
28           INT     21H
29 CSEG      ENDS
30 END      MAIN

```

BX = 0
AX = ?
SP = 0EH



| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

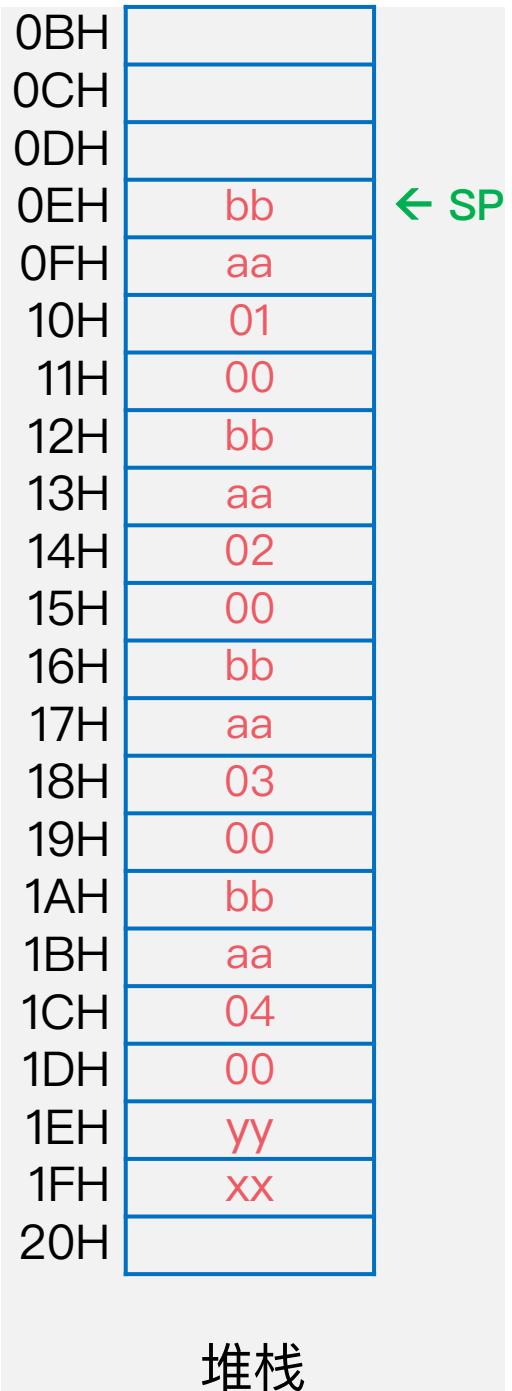
六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25  →       CALL     FACT
26  xxyy   NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG    ENDS
30 END     MAIN

```

BX = 0
AX = ?
SP = 0EH



| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

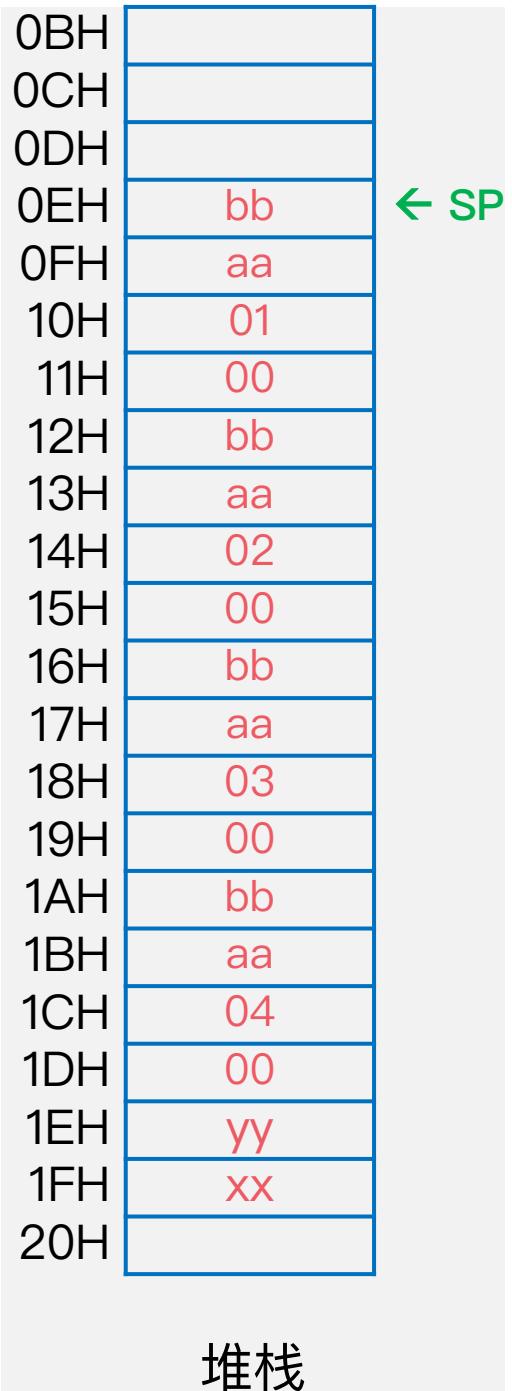
六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25  →       CALL     FACT
26  xxyy  NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG    ENDS
30 END     MAIN

```

BX = 0
AX = 1
SP = 0EH



| | | | |
|----|----------------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: <u>aabb</u> | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25  →       CALL     FACT
26  xxyy  NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG    ENDS
30 END     MAIN

```

BX = 0
AX = 1
SP = 10H

| | |
|-----|----|
| 0BH | |
| 0CH | |
| ODH | |
| 0EH | bb |
| 0FH | aa |
| 10H | 01 |
| 11H | 00 |
| 12H | bb |
| 13H | aa |
| 14H | 02 |
| 15H | 00 |
| 16H | bb |
| 17H | aa |
| 18H | 03 |
| 19H | 00 |
| 1AH | bb |
| 1BH | aa |
| 1CH | 04 |
| 1DH | 00 |
| 1EH | yy |
| 1FH | xx |
| 20H | |

← SP

| | | | |
|----|----------------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: <u>aabb</u> | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

堆栈

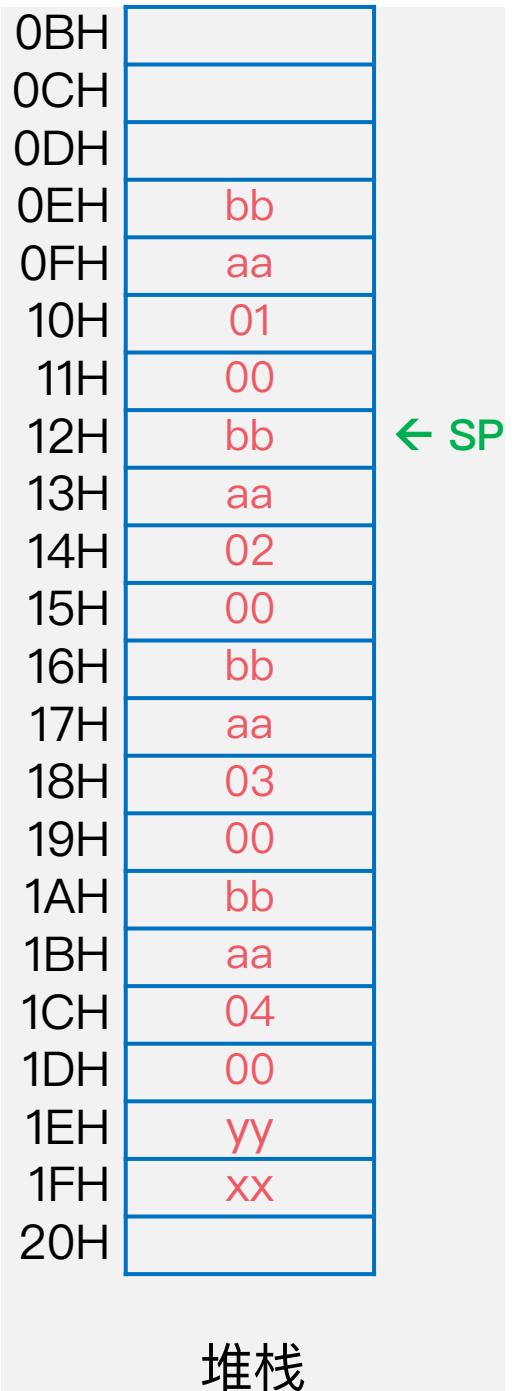
六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25  ←       CALL     FACT
26  xxyy   NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG    ENDS
30 END     MAIN

```

BX = 1
AX = 1
SP = 12H



| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

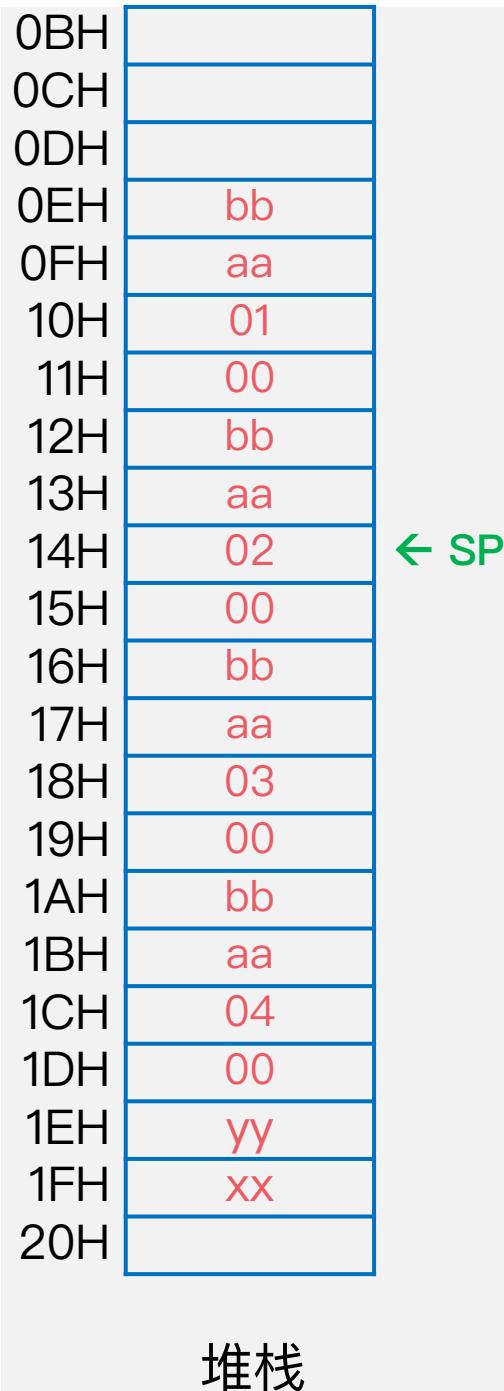
六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25  →       CALL     FACT
26  xxyy   NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG    ENDS
30 END     MAIN

```

BX = 1
AX = 1
SP = 14H



| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25  →       CALL     FACT
26  xxyy   NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG    ENDS
30 END     MAIN

```

BX = 2
AX = 1*2
SP = 14H

| | |
|-----|----|
| 0BH | |
| 0CH | |
| ODH | |
| 0EH | bb |
| 0FH | aa |
| 10H | 01 |
| 11H | 00 |
| 12H | bb |
| 13H | aa |
| 14H | 02 |
| 15H | 00 |
| 16H | bb |
| 17H | aa |
| 18H | 03 |
| 19H | 00 |
| 1AH | bb |
| 1BH | aa |
| 1CH | 04 |
| 1DH | 00 |
| 1EH | yy |
| 1FH | xx |
| 20H | |

← SP

| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

堆栈

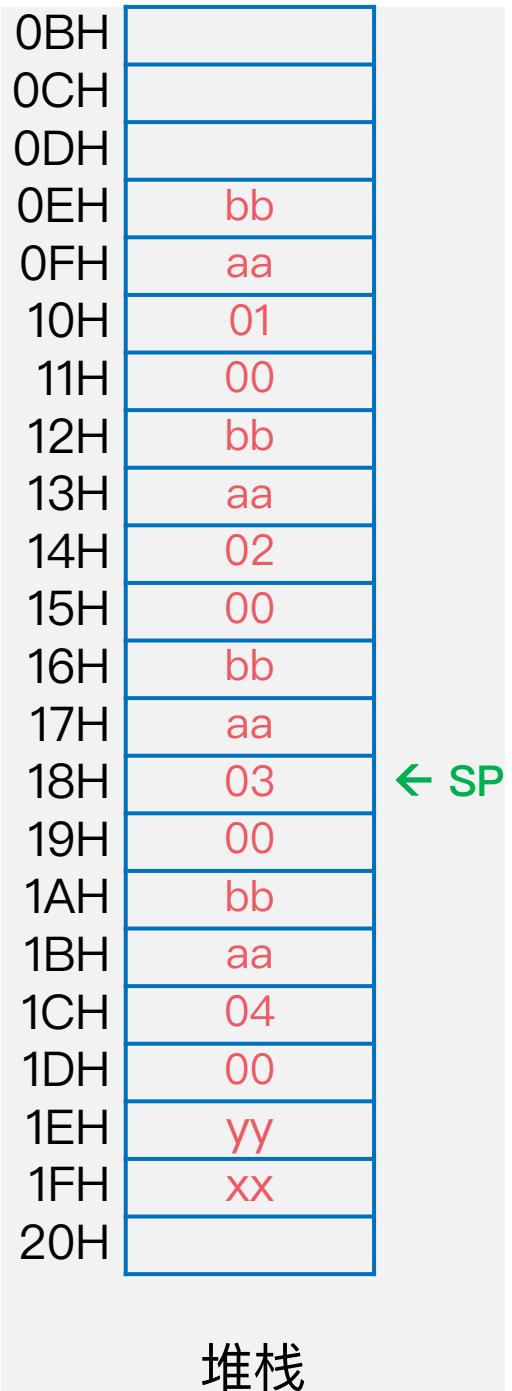
六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25  →       CALL     FACT
26  xxyy   NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG    ENDS
30 END     MAIN

```

BX = 2
AX = 1*2
SP = 14H



| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

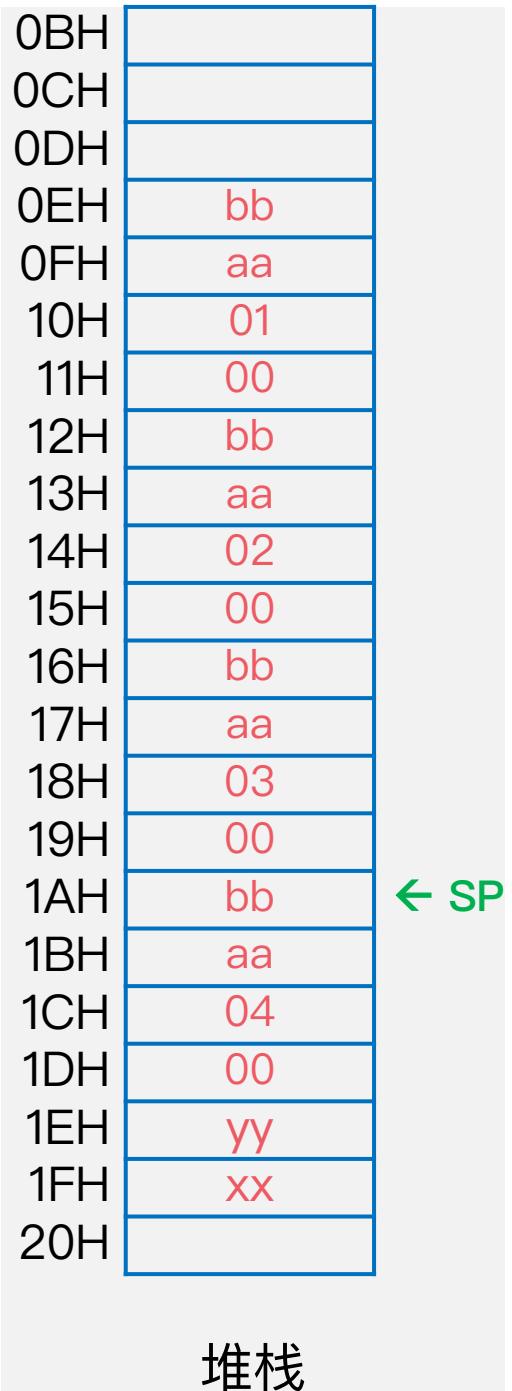
六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25  →       CALL     FACT
26  xxyy   NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG    ENDS
30 END     MAIN

```

BX = 3
AX = 1*2*3
SP = 1AH



| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

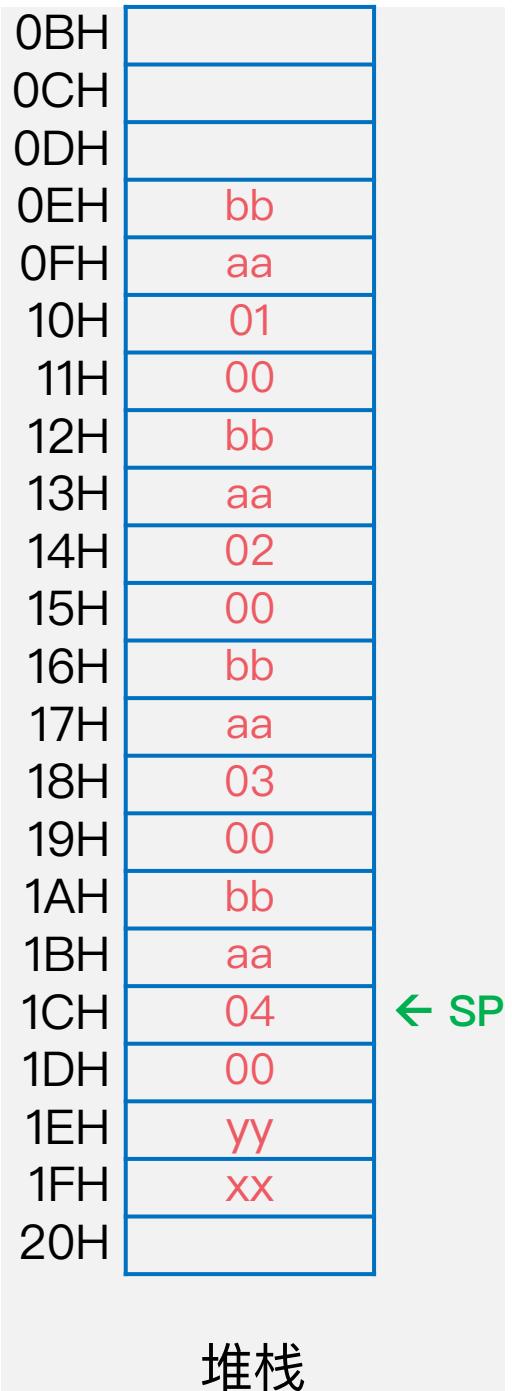
六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25  →       CALL     FACT
26  xxyy   NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG    ENDS
30 END     MAIN

```

BX = 3
AX = 1*2*3
SP = 1CH



| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

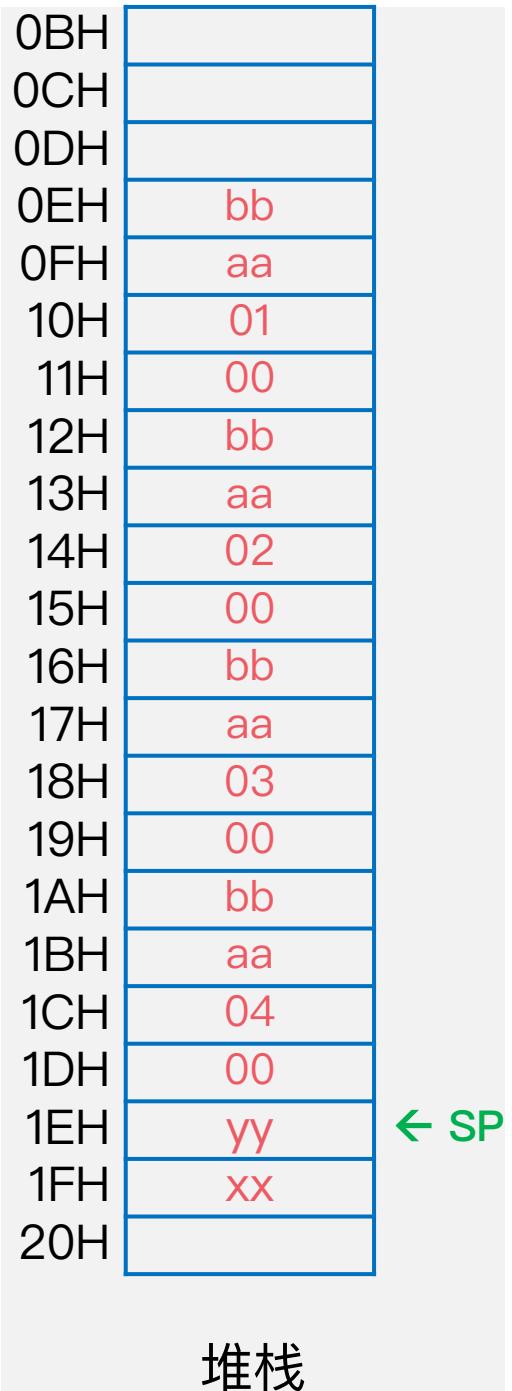
六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25  →       CALL     FACT
26  xxyy   NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG    ENDS
30 END     MAIN

```

BX = 4
AX = 1*2*3*4
SP = 1EH



| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

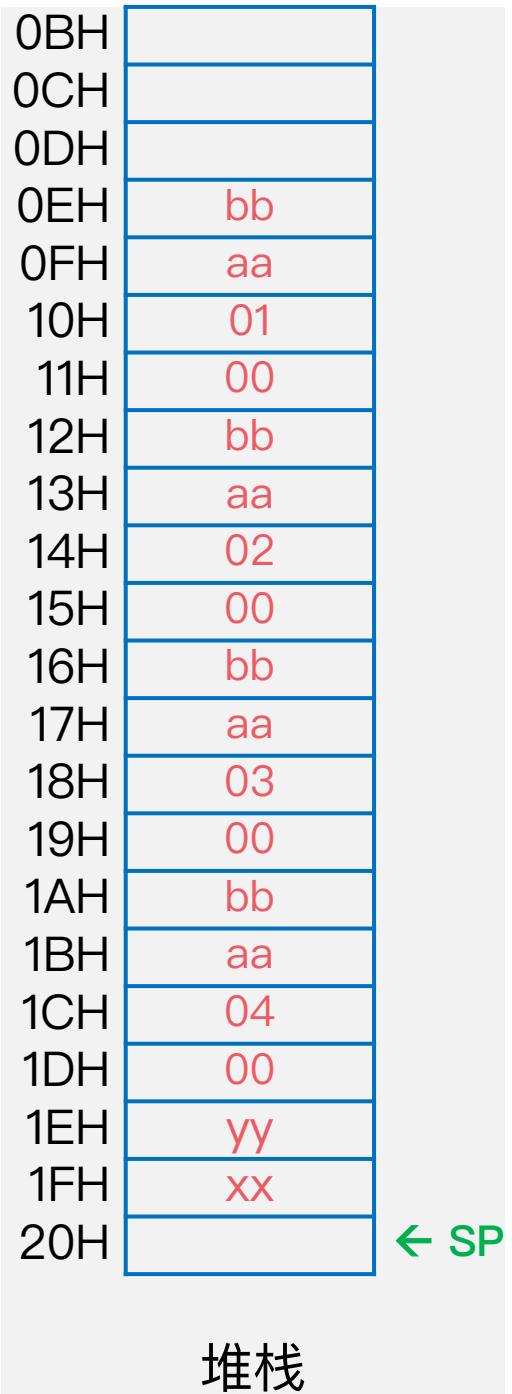
六. 递归子程序 — 例：求4!

```

21  MAIN:    MOV      AX, SSEG
22          MOV      DS, AX
23          MOV      SP, SIZE SKTOP
24          MOV      BX, 4
25          CALL     FACT
26  ← xxyy  NOP
27          MOV      AH, 4CH
28          INT     21H
29 CSEG      ENDS
30 END      MAIN

```

BX = 4
AX = 1*2*3*4
SP = 20H



| | | | |
|----|---------|------|--------|
| 2 | FACT | PROC | NEAR |
| 3 | | AND | BX, BX |
| 4 | | JZ | FACT1 |
| 5 | | PUSH | BX |
| 6 | | DEC | BX |
| 7 | | CALL | FACT |
| 8 | F: aabb | POP | BX |
| 9 | | MUL | BX |
| 10 | | RET | |
| 11 | FACT1: | MOV | AX, 1 |
| 12 | | RET | |
| 13 | FACT | ENDP | |

七. 可重入子程序

|> 一个子程序被某程序调用后，执行未结束，又被其它程序调用，这时，该程序都能正确地完成同时调用的各个任务，这样的子程序称为可重入子程序。

要使子程序具有可重入性，在子程序设计时必须满足下述条件：

- (1) 子程序中不能有可修改的指令或常数；
- (2) 当子程序的任何执行过程被中断时，寄存器的内容不能改变，以便恢复执行；
- (3) 子程序本身用堆栈作为自己的工作区，或由调用程序提供工作区。

八. 程序的连接

- • 编制一个解决实际问题的程序时，程序往往非常庞大，程序量可达数千条或数万条语句，甚至更多。这时，一个人很难完成，需要几个人乃至几十人共同编制。
- 这时可以采用模块化程序设计方法，把整个问题分解成若干个逻辑上比较完整的独立部分，然后由不同的人分别编写。每个部分称为程序的一个模块，各个模块可以独立编写、汇编和调试，最后连接成一个完整的可以执行的机器语言程序，以备装入运行。
- 这样的程序结构清晰，可读性和可维护性强。由于各模块独立编写、调试，所以编制简单，调试方便。
- 程序模块的连接信息是由程序段连接伪指令进行说明的，这些伪指令有：PUBLIC，EXTRN，GROUP和INCLUDE。