

汇编语言程序设计课程作业（十）

姓名：袁昊男 学号：2018091618008

检测点 10.1

补全程序，实现从内存 1000:0000 处开始执行指令。

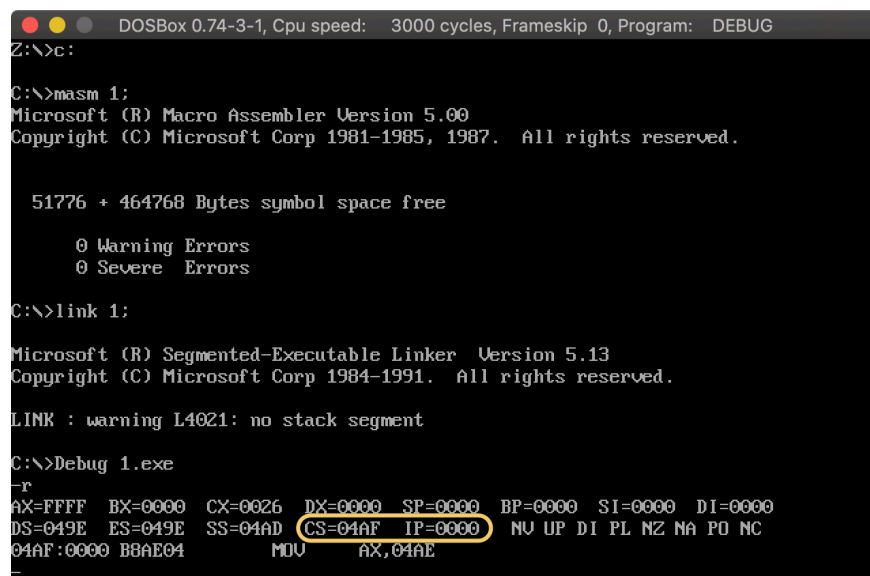
```
assume cs:code

stack segment
    db 16 dup(0)
stack ends

code segment
start: mov ax,stack
        mov ss,ax
        mov sp,16
        mov ax,1000h
        push ax
        mov ax,0
        push ax
        retf
code ends

end start
```

- (1) **分析：**retf 指令作用是从栈中弹出 2 个字单元，并先后修改 IP 与 CS 的值，故在压栈时 CS 的值首先入栈，IP 再入栈。其次 retf 实现了远转移。进而使指令从修改后的地址处开始执行。由于它所依赖的是栈中存储的内容，故在压栈、出栈过程中要注意顺序。
- (2) **跟踪、调试：**



```
DOSBox 0.74-3-1, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
Z:\>c:

C:\>masm 1:
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

51776 + 464768 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\>link 1:
Microsoft (R) Segmented-Executable Linker Version 5.13
Copyright (C) Microsoft Corp 1984-1991. All rights reserved.

LINK : warning L4021: no stack segment

C:\>Debug 1.exe
-r
AX=FFFF BX=0000 CX=0026 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=049E ES=049E SS=04AD CS=04AF IP=0000 04AF:0000 BBAE04
MOV AX,04AE
```

图 10.1.1 程序执行前 CS 与 IP 的值

```
DOSBox 0.74-3-1, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
AX=1000 BX=0000 CX=0026 DX=0000 SP=0010 BP=0000 SI=0000 DI=0000
DS=049E ES=049E SS=04AE CS=04AF IP=000B  NU UP DI PL NZ NA PO NC
04AF:000B 50          PUSH  AX
-t
AX=1000 BX=0000 CX=0026 DX=0000 SP=000E BP=0000 SI=0000 DI=0000
DS=049E ES=049E SS=04AE CS=04AF IP=000C  NU UP DI PL NZ NA PO NC
04AF:000C B80000     MOV   AX,0000
-t
AX=0000 BX=0000 CX=0026 DX=0000 SP=000E BP=0000 SI=0000 DI=0000
DS=049E ES=049E SS=04AE CS=04AF IP=000F  NU UP DI PL NZ NA PO NC
04AF:000F 50          PUSH  AX
-t
AX=0000 BX=0000 CX=0026 DX=0000 SP=000C BP=0000 SI=0000 DI=0000
DS=049E ES=049E SS=04AE CS=04AF IP=0010  NU UP DI PL NZ NA PO NC
04AF:0010 CB          RETF
-t
AX=0000 BX=0000 CX=0026 DX=0000 SP=0010 BP=0000 SI=0000 DI=0000
DS=049E ES=049E SS=04AE CS=1000 IP=0000  NU UP DI PL NZ NA PO NC
1000:0000 0000     ADD   EBX+SI,AL      DS:0000=CD
```

图 10.1.2 程序执行后 CS 与 IP 的值

检测点 10.2

下面的程序执行后，ax 中的数值为多少？

| 内存地址 | 机器码 | 汇编指令 |
|--------|----------|-----------|
| 1000:0 | b8 00 00 | mov ax,0 |
| 1000:3 | e8 01 00 | call s |
| 1000:6 | 40 | inc ax |
| 1000:7 | 58 | s: pop ax |

- (1) 分析：call 指令在执行时，首先 push IP（此时 IP 值为 6h，即 CPU 将要执行的下一条指令地址），然后 jmp s，执行标号 s 处的代码，pop ax 弹栈写入 ax，此时(ax)=6h。因为没有 ret 指令，因此程序不会返回，所以 inc ax 没有执行。故 ax 中的数值为 6h。
- (2) 跟踪、调试：

```
DOSBox 0.74-3-1, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
04AF:0000 B80000     MOV   AX,0000
-t
AX=0000 BX=0000 CX=001D DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=049E ES=049E SS=04AD CS=04AF IP=0003  NU UP DI PL NZ NA PO NC
04AF:0003 E80100     CALL  0007
-t
AX=0000 BX=0000 CX=001D DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=049E ES=049E SS=04AD CS=04AF IP=0007  NU UP DI PL NZ NA PO NC
04AF:0007 58          POP   AX
-t
AX=0006 BX=0000 CX=001D DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=049E ES=049E SS=04AD CS=04AF IP=0008  NU UP DI PL NZ NA PO NC
04AF:0008 B8004C     MOV   AX,4C00
-t
AX=4C00 BX=0000 CX=001D DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=049E ES=049E SS=04AD CS=04AF IP=000B  NU UP DI PL NZ NA PO NC
04AF:000B CD21     INT   21
-t
Program terminated normally
```

图 10.2.1 程序执行后 ax 中的值为 6

检测点 10.3

下面的程序执行后，ax 中的数值为多少？

| 内存地址 | 机器码 | 汇编指令 |
|--------|----------------|----------------|
| 1000:0 | b8 00 00 | mov ax,0 |
| 1000:3 | 9A 09 00 00 10 | call far ptr s |
| 1000:8 | 40 | inc ax |
| 1000:9 | 58 | s: pop ax |
| | | add ax,ax |
| | | pop bx |
| | | add ax,bx |

分析：call far ptr 指令在执行时，首先 push CS（此时 CS 值为 1000h），然后 push IP（此时 IP 值为 8h，即 CPU 将要执行的下一条指令地址）。最后 jmp far ptr s，执行标号 s 处的代码，pop ax 将 IP 值弹栈写入 ax，此时(ax)=8h，add 指令后(ax)=10h，pop bx 将 CS 值弹栈写入 bx，此时(bx)=1000h，add 指令后(ax)=1010h。因为没有 ret 指令，因此程序不会返回，所以 inc ax 没有执行。故 ax 中的数值为 1010h。

检测点 10.4

下面的程序执行后，ax 中的数值为多少？

| 内存地址 | 机器码 | 汇编指令 |
|--------|----------|-------------|
| 1000:0 | b8 06 00 | mov ax,6 |
| 1000:3 | ff d0 | call ax |
| 1000:5 | 40 | inc ax |
| 1000:6 | | mov bp,sp |
| | | add ax,[bp] |

分析：call 指令在执行时，首先 push IP（此时 IP 值为 5h，即 CPU 将要执行的下一条指令地址），由于 ax 中的数值为 6h，因此转移到 1000:6 处执行。压栈后(sp)=(sp)-2，则当前 sp 中的数值为 fffeh，因此 add 指令执行结果为(ax)=(ax)+(ds:[ffeh])=6h+5h=0bh。

检测点 10.5

(1) 下面的程序执行后，ax 中的数值为多少？（注意：用 call 指令的原理来分析，不要在 Debug 中单步跟踪来验证你的结论。对于此程序，在 Debug 中单步跟踪的结果，不能代表 CPU 的实际执行结果。）

```
assume cs:code

stack segment
    dw 8 dup(0)
stack ends

code segment
start: mov ax,stack
       mov ss,ax
```

```

        mov sp,16
        mov ds,ax
        mov ax,0
        call word ptr ds:[0EH]
        inc ax
        inc ax
        inc ax
        mov ax,4c00h
        int 21h
code ends

end start

```

分析：栈段初始化并定义了 8 个 dword 型数据共 16 字节的栈空间。初始化，将栈段地址赋值给了 ss 与 ds，即数据段和栈段共享同一个内存空间段。执行 call word ptr 指令，将 IP 压栈（此时 IP 值为下一条指令 inc ax 的偏移地址），然后 IP 转移到 ds:[0EH]单元，而 ds:[0EH]单元即栈顶单元，因此执行 inc ax 指令。ax 值连续自增三次，最后 ax 中的数值为 3h。题目中特别提示 Debug 中单步跟踪结果与 CPU 实际执行结果不一致，发现在执行 call word ptr ds:[0EH]指令时，显示 ds:[0EH]值为 065DH，可能是程序中断导致。

(2) 下面的程序执行后，ax 和 bx 中的数值为多少？

```

assume cs:code

data segment
    dw 8 dup(0)
data ends

code segment
start: mov ax,data
        mov ss,ax
        mov sp,16
        mov word ptr ss:[0],offset s
        mov ss:[2],cs
        call dword ptr ss:[0]
        nop
s: mov ax,offset s
    sub ax,ss:[0cH]
    mov bx,cs
    sub bx,ss:[0eH]
    mov ax,4c00h
    int 21h
code ends

end start

```

1) **分析：**经过 Debug 跟踪、调试，分析如下：首先数据段初始化并定义了 8 个 dword 型数据共 16 字节的数据空间。初始化，将数据段地址赋值给了 ss，即栈段与数据段共享同一个内存空间段。然后将 s 段偏移地址放入 ss:[0]的字单元（ss:[0]=1ah），将 cs 值放入 ss:[2]的字单元（ss:[2]=(cs)=04afh），call dword ptr 指令现将 CS 入栈，再将 IP 入栈（CS 值为 04afh，IP 值为 0019h），后执行 jmp dword ptr ss:[0]，即跳转到 s 段

继续执行。首先 s 段偏移地址送入 ax，sub 指令执行后， $(ax) = (ax) - (ss:[0cH]) = 1h$ ，后将 cs 值送入 bx，sub 指令执行后， $(bx) = (bx) - (ss:[0eH]) = 04afh - 04afh = 0$ 。因此最终结果： $(ax) = 1h$ ， $(bx) = 0$ 。

2) 跟踪、调试:

```
DOSBox 0.74-3-1, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
DS=049E ES=049E SS=04AE CS=04AF IP=001A  NU UP DI PL NZ NA PO NC
04AF:001A BB1A00      MOV     AX,001A
-t
AX=001A BX=0000 CX=003E DX=0000 SP=000C BP=0000 SI=0000 DI=0000
DS=049E ES=049E SS=04AE CS=04AF IP=001D  NU UP DI PL NZ NA PO NC
04AF:001D 36          SS:
04AF:001E 2B060C00      SUB     AX,[000C]          SS:000C=0019
-t
AX=0001 BX=0000 CX=003E DX=0000 SP=000C BP=0000 SI=0000 DI=0000
DS=049E ES=049E SS=04AE CS=04AF IP=0022  NU UP DI PL NZ NA PO NC
04AF:0022 BCCB      MOV     BX,CS
-t
AX=0001 BX=04AF CX=003E DX=0000 SP=000C BP=0000 SI=0000 DI=0000
DS=049E ES=049E SS=04AE CS=04AF IP=0024  NU UP DI PL NZ NA PO NC
04AF:0024 36          SS:
04AF:0025 2B1E0E00      SUB     BX,[000E]          SS:000E=04AF
-t
AX=0001 BX=0000 CX=003E DX=0000 SP=000C BP=0000 SI=0000 DI=0000
DS=049E ES=049E SS=04AE CS=04AF IP=0029  NU UP DI PL ZR NA PE NC
04AF:0029 B8004C      MOV     AX,4C00
-
```

图 10.5.1 Debug 调试结果