汇编笔记

大纲

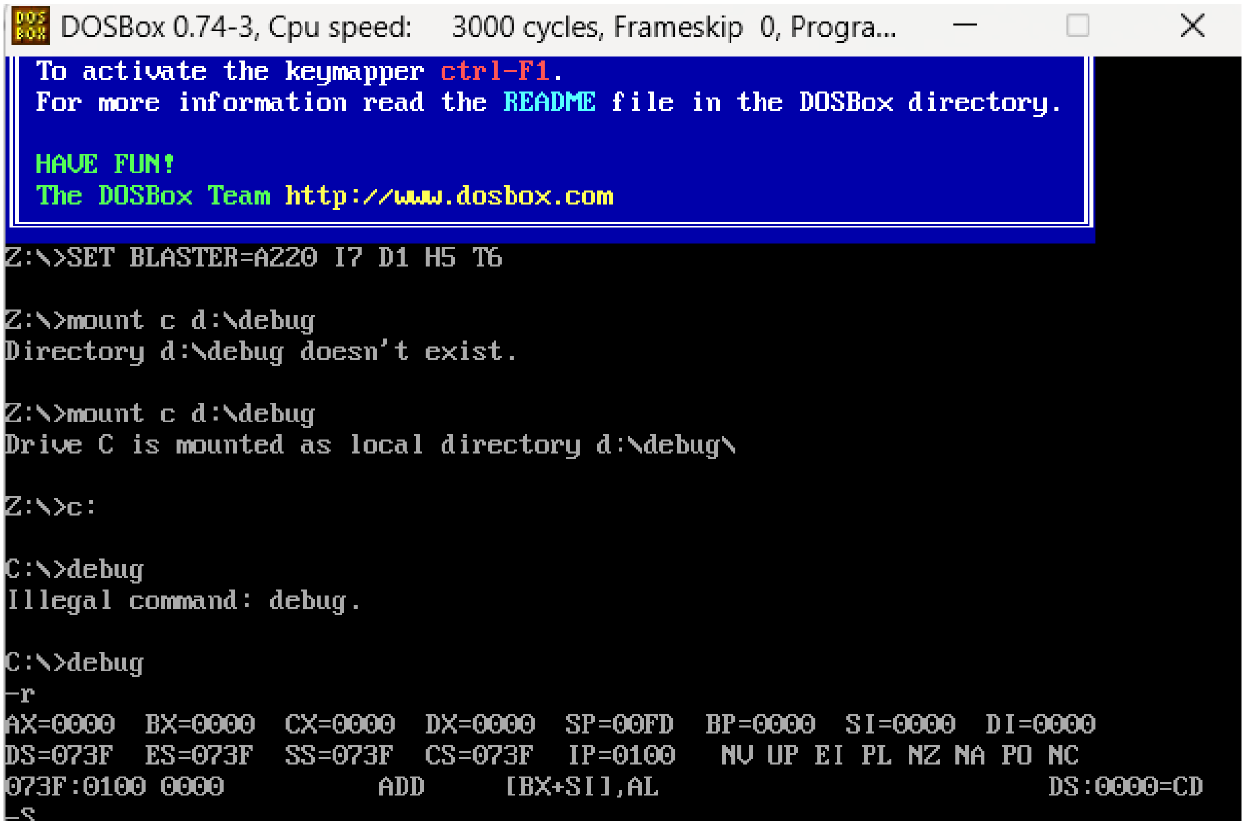
1. BCD码
    1. 压缩BCD码：使用四位二进制来表示以为BCD码。
    2. 非压缩BCD码：将8位二进制的高四位指令，仅仅使用低四位来表达一位BCD码，则被称为非压缩BCD码。

debug之中的主要命令

- `r`查看，改变CPU寄存器值中的内容
- `d`查看内存之中的内容
- `e`修改内存之中的内容

第一次实验

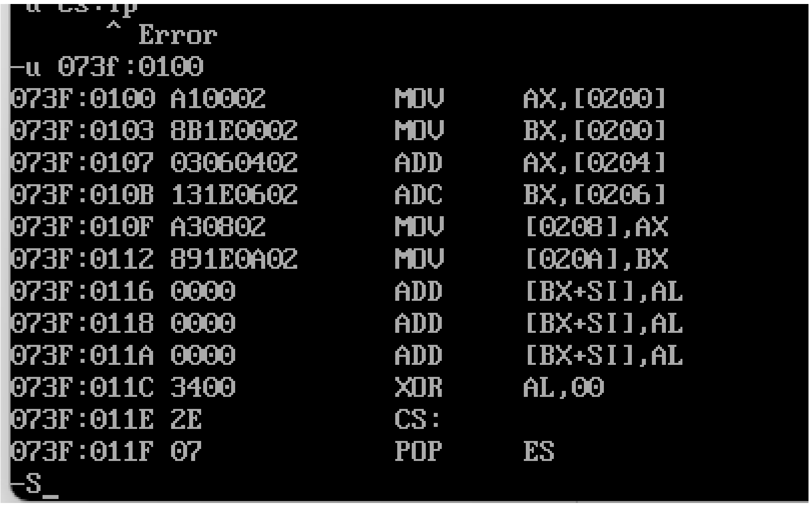
0. 找到dosbox所安装的地址，打开dosbox程序
1. 打开debug程序，使用了`r`命令查看初始寄存器之中的内容


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
 To activate the keymapper ctrl-F1.
 For more information read the README file in the DOSBox directory.

 HAVE FUN!
 The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount c d:\debug
Directory d:\debug doesn't exist.

Z:\>mount c d:\debug
Drive C is mounted as local directory d:\debug\

Z:\>c:

C:\>debug
Illegal command: debug.

C:\>debug
-r
AX=0000  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 0000          ADD     [BX+SI],AL                         DS:0000=CD
-S
```

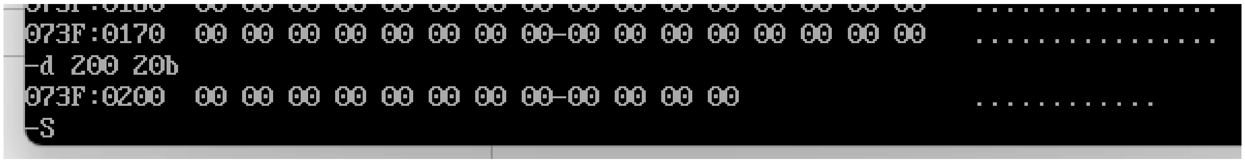
2. 使用a命令在`cs:ip`位置输入汇编代码

3. 使用u查看刚刚输入的汇编代码所对应的机器码



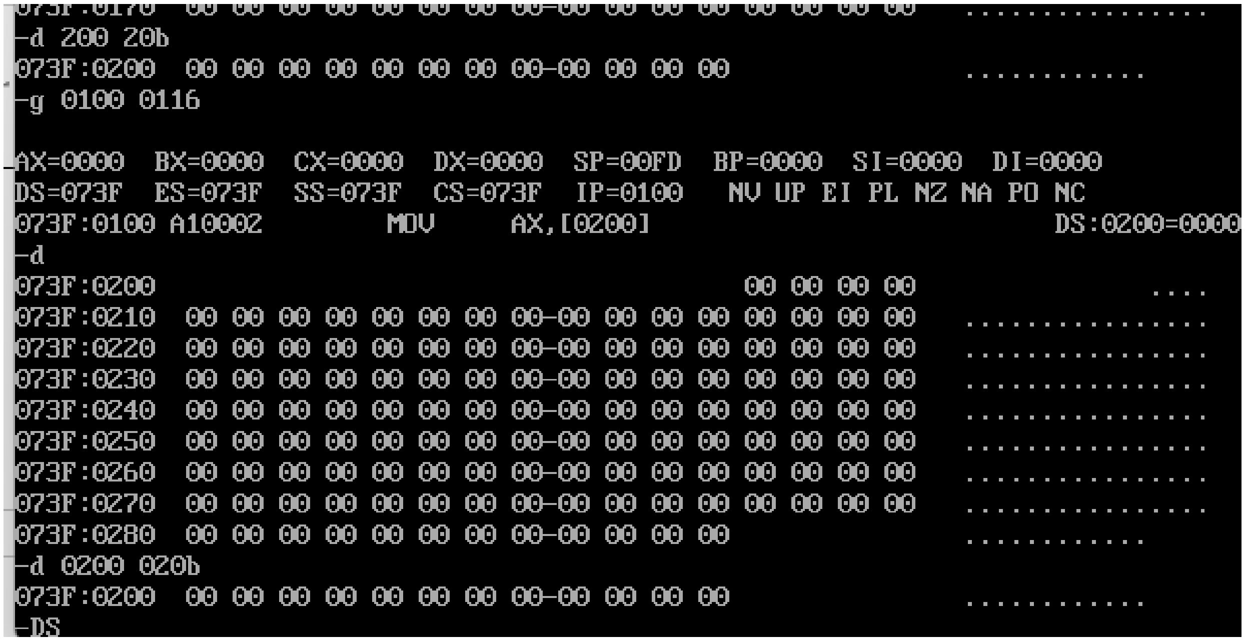
```
 u cs:ip
       ^ Error
-u 073f:0100
073F:0100 A10002        MOV     AX,[0200]
073F:0103 8B1E0002      MOV     BX,[0200]
073F:0107 03060402      ADD     AX,[0204]
073F:010B 131E0602      ADC     BX,[0206]
073F:010F A30802        MOV     [0208],AX
073F:0112 891E0A02      MOV     [020A],BX
073F:0116 0000          ADD     [BX+SI],AL
073F:0118 0000          ADD     [BX+SI],AL
073F:011A 0000          ADD     [BX+SI],AL
073F:011C 3400          XOR     AL,00
073F:011E 2E            CS:
073F:011F 07            POP     ES
-S_
```

4. 在程序未执行以前，使用d命令查看内存之中的内容



```
073F:0160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-d 200 20b
073F:0200  00 00 00 00 00 00 00 00-00 00 00 00               ............
-S
```

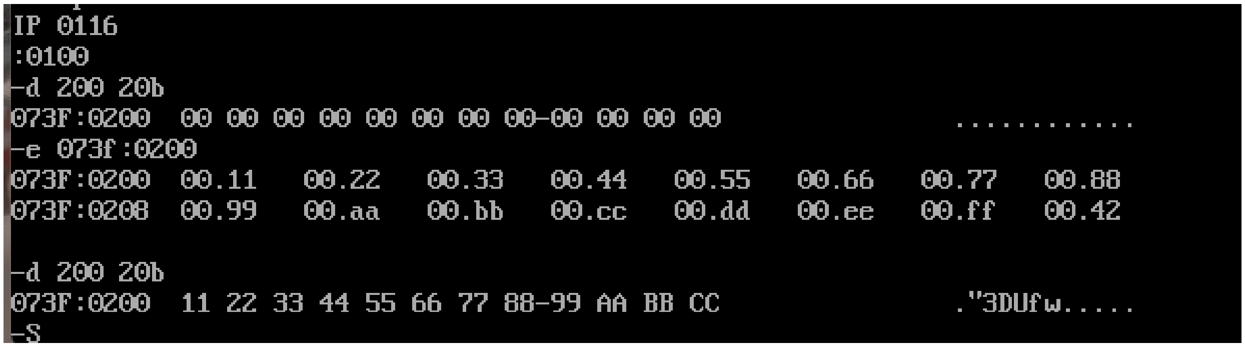
5. 执行命令，分别查看寄存器之中的内容和内存之中的内容



```
073F:0170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-d 200 20b
073F:0200  00 00 00 00 00 00 00 00-00 00 00 00               ............
-g 0100 0116

AX=0000  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 A10002        MOV     AX,[0200]                          DS:0200=0000
-d
073F:0200                                      00 00 00 00               ....
073F:0210  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0220  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0230  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0240  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0250  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0260  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0270  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0280  00 00 00 00 00 00 00 00-00 00 00 00               ............
-d 0200 020b
073F:0200  00 00 00 00 00 00 00 00-00 00 00 00               ............
-DS
```

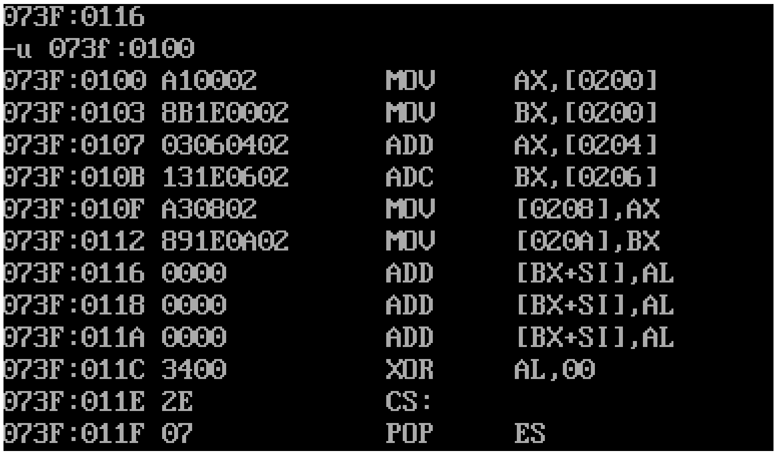
注：似乎因为0200~020b之中初始数值为0，导致执行了编译命令以后较原来的没有区别。稍后尝试一下在初始时便修改0200到020B区间的内容。

1. 使用`e`命令修改200到20b区间的内容



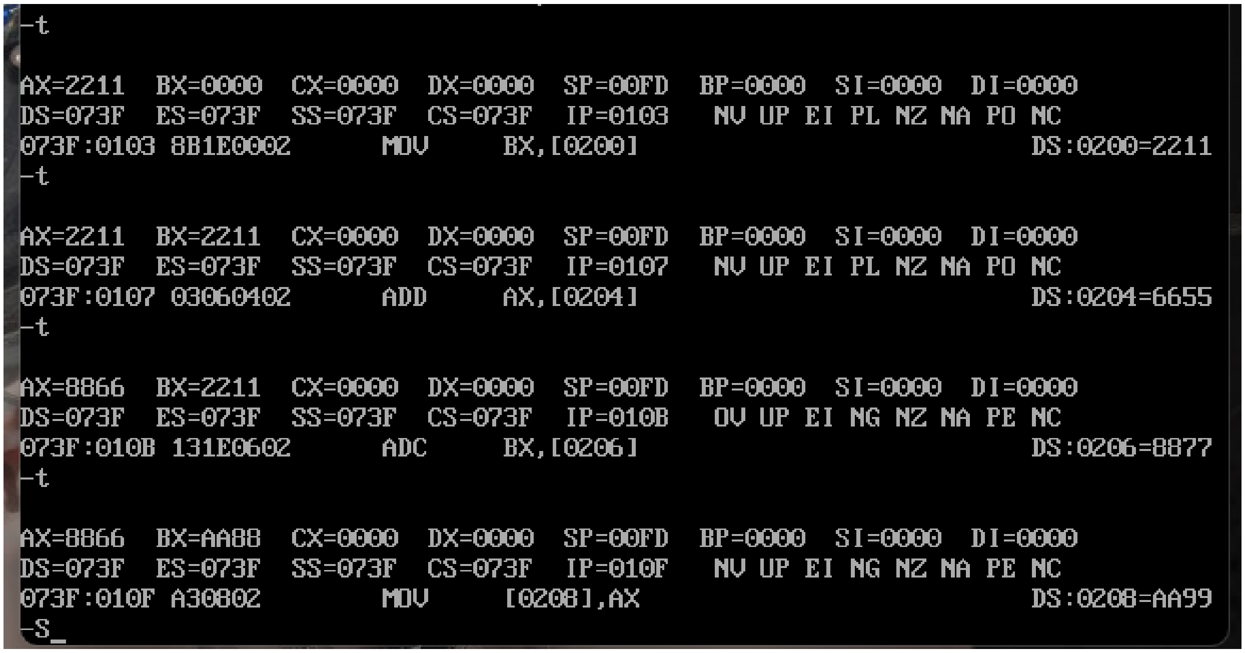
```
IP 0116
:0100
-d 200 20b
073F:0200  00 00 00 00 00 00 00 00-00 00 00 00               ............
-e 073f:0200
073F:0200  00.11   00.22   00.33   00.44   00.55   00.66   00.77   00.88
073F:0208  00.99   00.aa   00.bb   00.cc   00.dd   00.ee   00.ff   00.42

-d 200 20b
073F:0200  11 22 33 44 55 66 77 88-99 AA BB CC               ."3DUfw.....
-S
```

2. 输入汇编指令执行

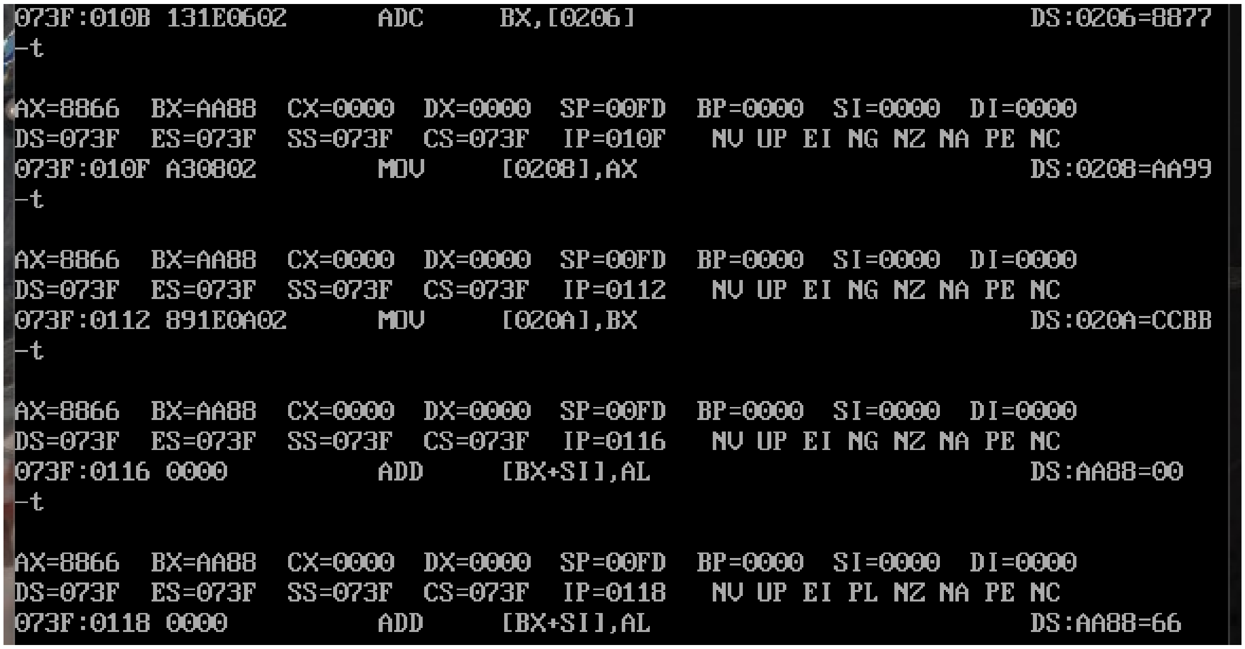


```
073F:0116
-u 073f:0100
073F:0100 A10002        MOV     AX,[0200]
073F:0103 8B1E0002      MOV     BX,[0200]
073F:0107 03060402      ADD     AX,[0204]
073F:010B 131E0602      ADC     BX,[0206]
073F:010F A30802        MOV     [0208],AX
073F:0112 891E0A02      MOV     [020A],BX
073F:0116 0000          ADD     [BX+SI],AL
073F:0118 0000          ADD     [BX+SI],AL
073F:011A 0000          ADD     [BX+SI],AL
073F:011C 3400          XOR     AL,00
073F:011E 2E            CS:
073F:011F 07            POP     ES
```

3. 执行指令

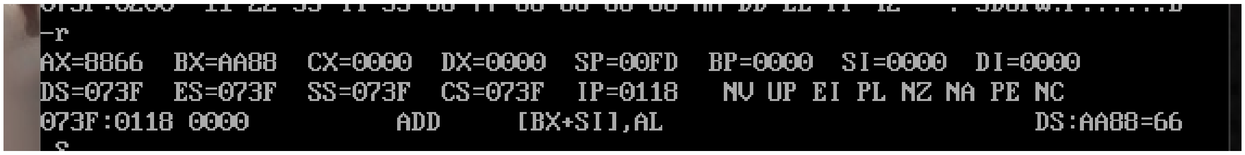


```
-t

AX=2211  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0103   NV UP EI PL NZ NA PO NC
073F:0103 8B1E0002      MOV     BX,[0200]                          DS:0200=2211
-t

AX=2211  BX=2211  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0107   NV UP EI PL NZ NA PO NC
073F:0107 03060402      ADD     AX,[0204]                          DS:0204=6655
-t

AX=8866  BX=2211  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010B   OV UP EI NG NZ NA PE NC
073F:010B 131E0602      ADC     BX,[0206]                          DS:0206=8877
-t

AX=8866  BX=AA88  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010F   NV UP EI NG NZ NA PE NC
073F:010F A30802        MOV     [0208],AX                          DS:0208=AA99
-S_
```



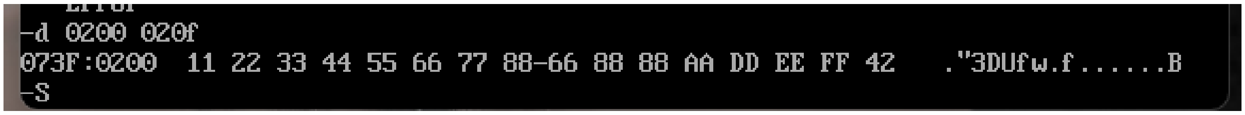
```
073F:010B 131E0602      ADC     BX,[0206]                          DS:0206=8877
-t

AX=8866  BX=AA88  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010F   NV UP EI NG NZ NA PE NC
073F:010F A30802        MOV     [0208],AX                          DS:0208=AA99
-t

AX=8866  BX=AA88  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0112   NV UP EI NG NZ NA PE NC
073F:0112 891E0A02      MOV     [020A],BX                          DS:020A=CCBB
-t

AX=8866  BX=AA88  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0116   NV UP EI NG NZ NA PE NC
073F:0116 0000          ADD     [BX+SI],AL                         DS:AA88=00
-t

AX=8866  BX=AA88  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0118   NV UP EI PL NZ NA PE NC
073F:0118 0000          ADD     [BX+SI],AL                         DS:AA88=66
```

3. 查看寄存器之中的内容



```
073F:0200  11 22 33 44 55 66 77 88-66 88 88 AA DD EE FF 42   ."3DUfw.f......B
-r
AX=8866  BX=AA88  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0118   NV UP EI PL NZ NA PE NC
073F:0118 0000          ADD     [BX+SI],AL                         DS:AA88=66
-S
```

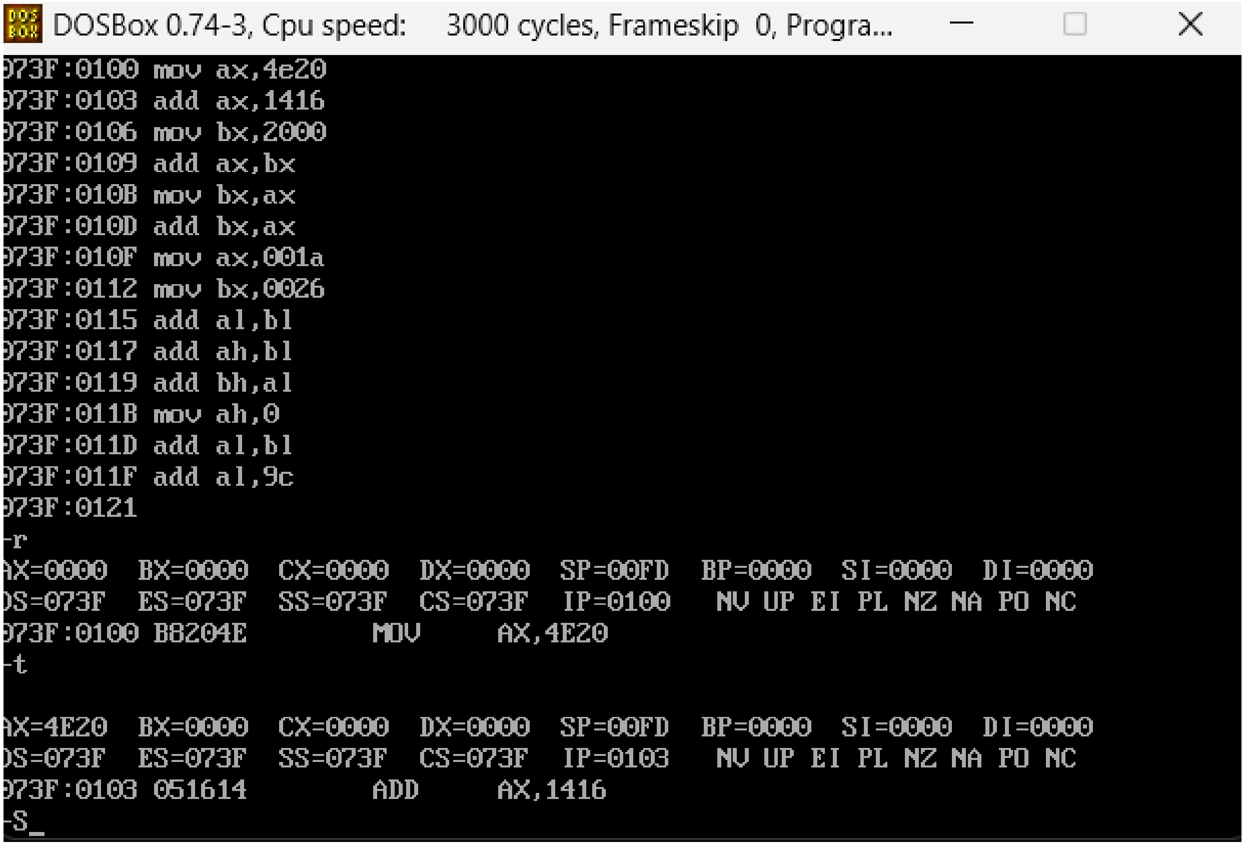
4. 查看0200到020b区间的内容



```
  Error
-d 0200 020f
073F:0200  11 22 33 44 55 66 77 88-66 88 88 AA DD EE FF 42   ."3DUfw.f......B
-S
```

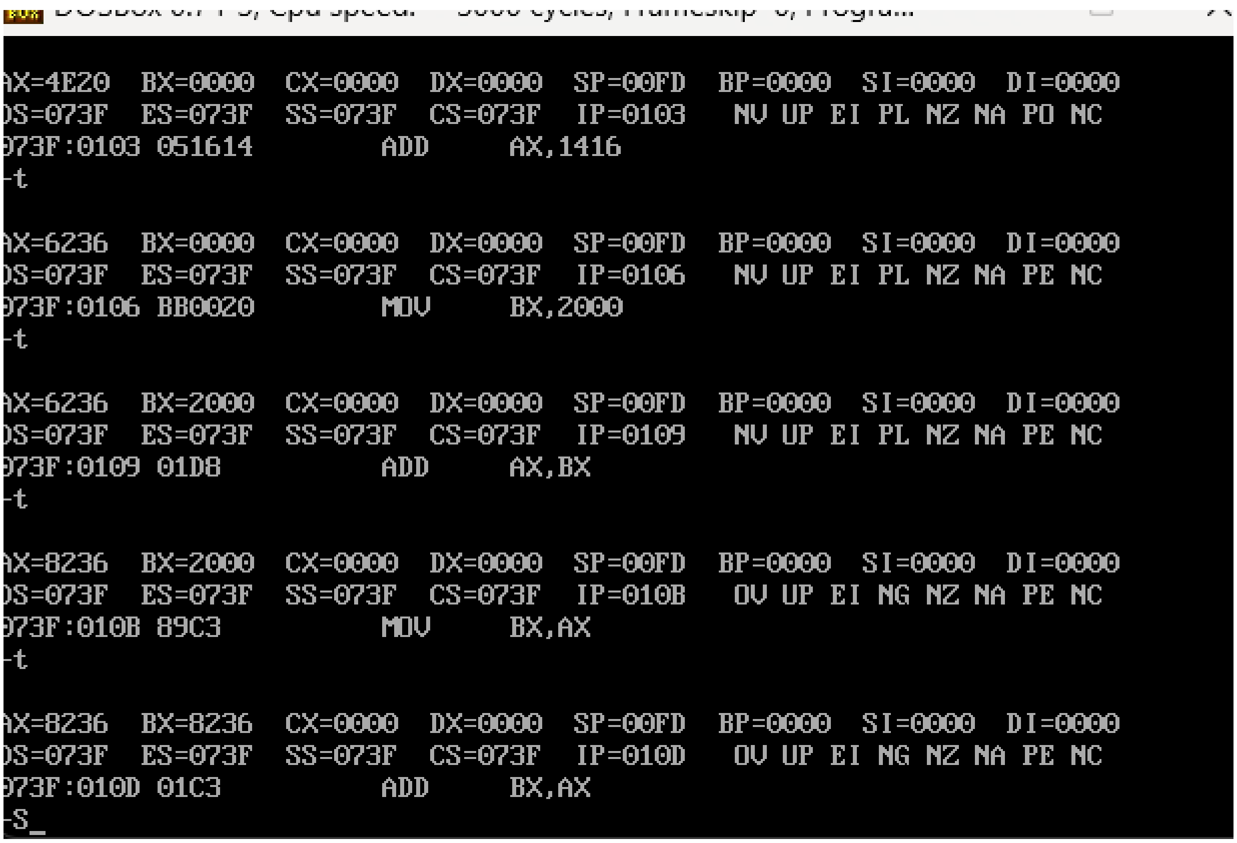
《汇编语言》王爽P45页习题

1. 输入指令

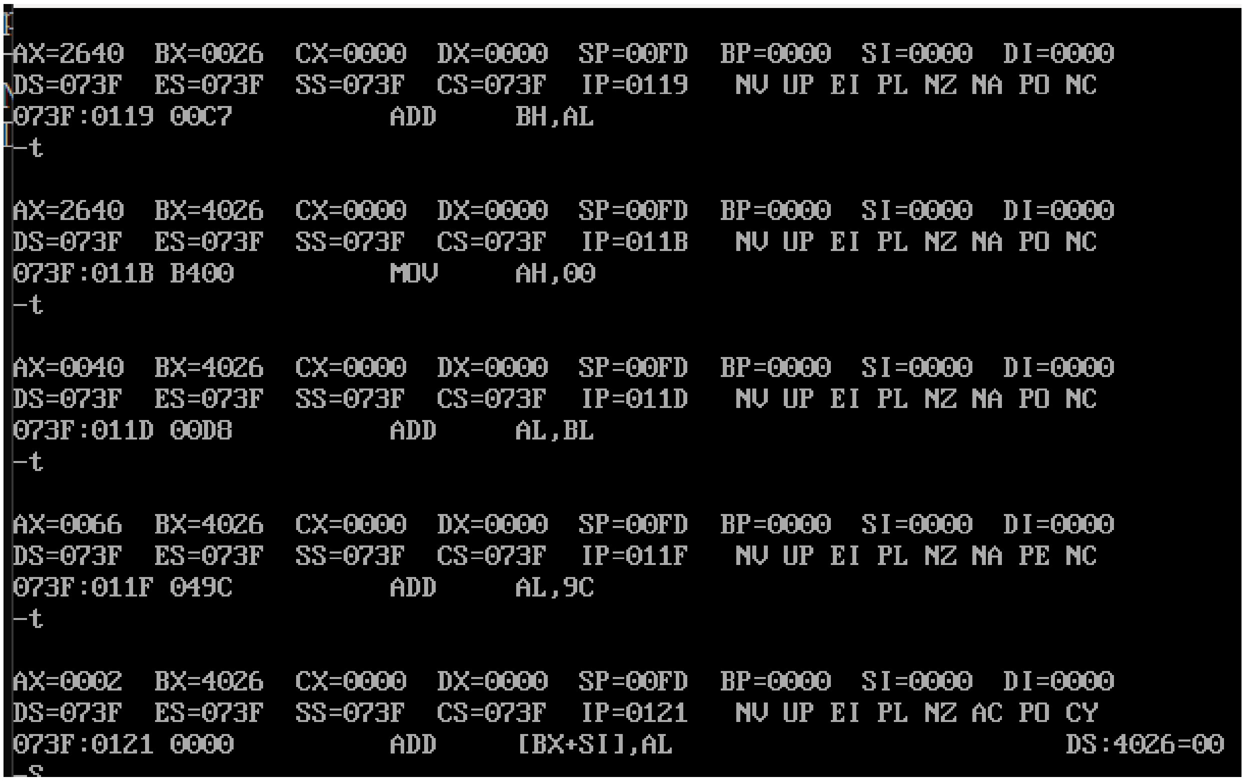

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
073F:0100 mov ax,4e20
073F:0103 add ax,1416
073F:0106 mov bx,2000
073F:0109 add ax,bx
073F:010B mov bx,ax
073F:010D add bx,ax
073F:010F mov ax,001a
073F:0112 mov bx,0026
073F:0115 add al,bl
073F:0117 add ah,bl
073F:0119 add bh,al
073F:011B mov ah,0
073F:011D add al,bl
073F:011F add al,9c
073F:0121
-r
AX=0000  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 B8204E        MOV     AX,4E20
-t

AX=4E20  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0103   NV UP EI PL NZ NA PO NC
073F:0103 051614        ADD     AX,1416
-S_
```

2. 执行命令

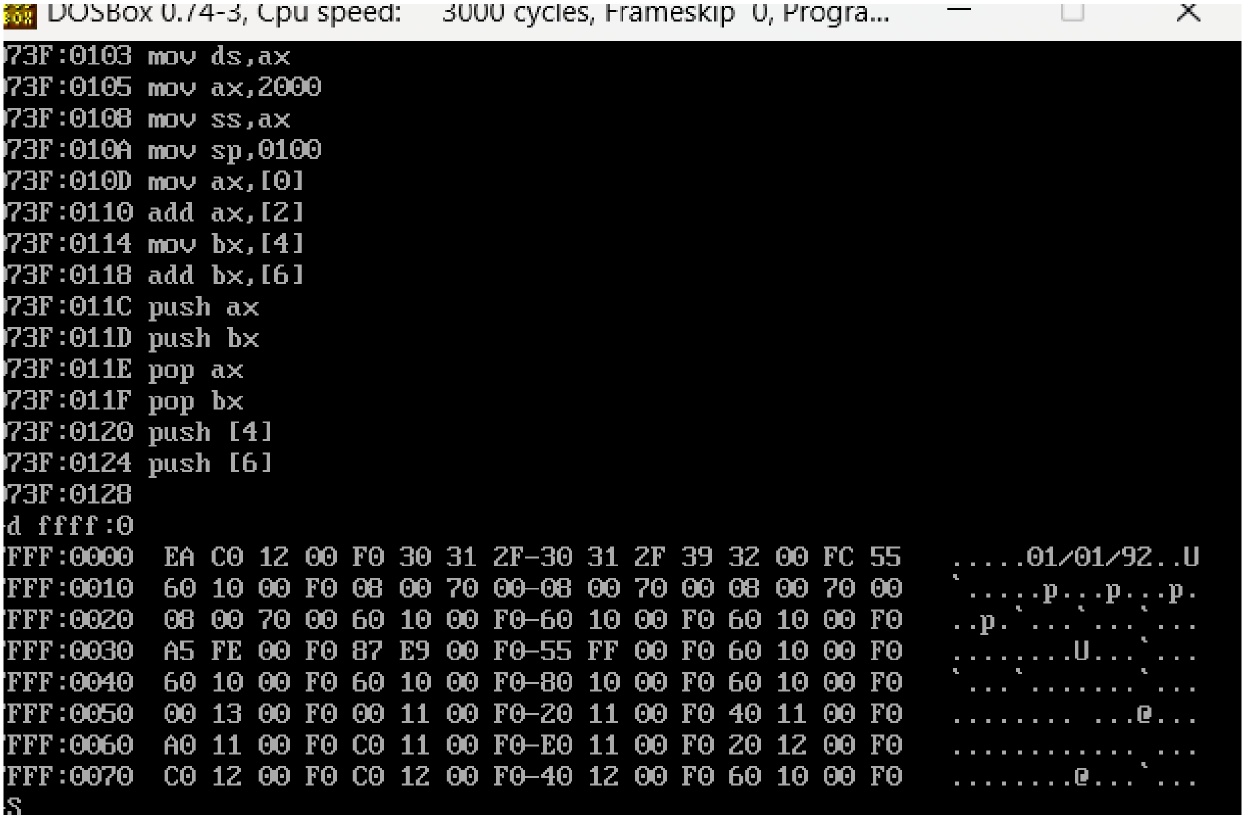


```
AX=4E20  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0103   NV UP EI PL NZ NA PO NC
073F:0103 051614        ADD     AX,1416
-t

AX=6236  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0106   NV UP EI PL NZ NA PE NC
073F:0106 BB0020        MOV     BX,2000
-t

AX=6236  BX=2000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0109   NV UP EI PL NZ NA PE NC
073F:0109 01D8          ADD     AX,BX
-t

AX=8236  BX=2000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010B   OV UP EI NG NZ NA PE NC
073F:010B 89C3          MOV     BX,AX
-t

AX=8236  BX=8236  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010D   OV UP EI NG NZ NA PE NC
073F:010D 01C3          ADD     BX,AX
-S_
```



```
AX=2640  BX=0026  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0119   NV UP EI PL NZ NA PO NC
073F:0119 00C7          ADD     BH,AL
-t

AX=2640  BX=4026  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=011B   NV UP EI PL NZ NA PO NC
073F:011B B400          MOV     AH,00
-t

AX=0040  BX=4026  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=011D   NV UP EI PL NZ NA PO NC
073F:011D 00D8          ADD     AL,BL
-t

AX=0066  BX=4026  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=011F   NV UP EI PL NZ NA PE NC
073F:011F 049C          ADD     AL,9C
-t

AX=0002  BX=4026  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0121   NV UP EI PL NZ AC PO CY
073F:0121 0000          ADD     [BX+SI],AL                         DS:4026=00
-S
```

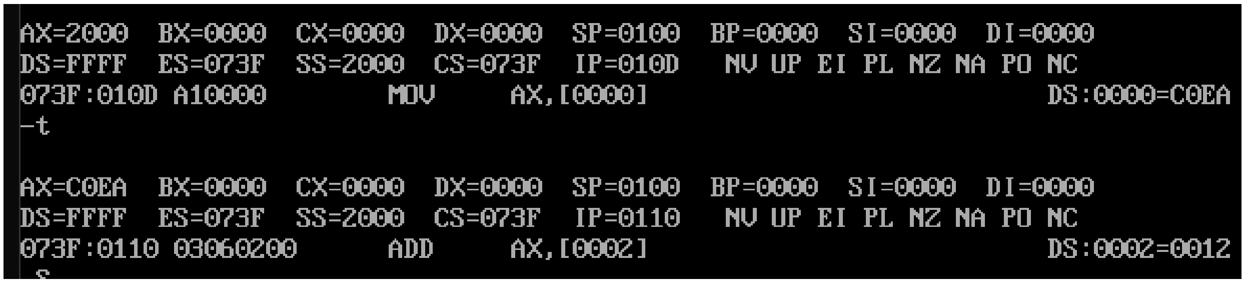
王爽P74实验


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
73F:0103 mov ds,ax
73F:0105 mov ax,2000
73F:0108 mov ss,ax
73F:010A mov sp,0100
73F:010D mov ax,[0]
73F:0110 add ax,[2]
73F:0114 mov bx,[4]
73F:0118 add bx,[6]
73F:011C push ax
73F:011D push bx
73F:011E pop ax
73F:011F pop bx
73F:0120 push [4]
73F:0124 push [6]
73F:0128
d ffff:0
FFF:0000  EA C0 12 00 F0 30 31 2F-30 31 2F 39 32 00 FC 55   .....01/01/92..U
FFF:0010  60 10 00 F0 08 00 70 00-08 00 70 00 08 00 70 00   `.....p...p...p.
FFF:0020  08 00 70 00 60 10 00 F0-60 10 00 F0 60 10 00 F0   ..p.`...`...`...
FFF:0030  A5 FE 00 F0 87 E9 00 F0-55 FF 00 F0 60 10 00 F0   ........U...`...
FFF:0040  60 10 00 F0 60 10 00 F0-80 10 00 F0 60 10 00 F0   `...`.......`...
FFF:0050  00 13 00 F0 00 11 00 F0-20 11 00 F0 40 11 00 F0   ........ ...@...
FFF:0060  A0 11 00 F0 C0 11 00 F0-E0 11 00 F0 20 12 00 F0   ............ ...
FFF:0070  C0 12 00 F0 C0 12 00 F0-40 12 00 F0 60 10 00 F0   ........@...`...
S
```

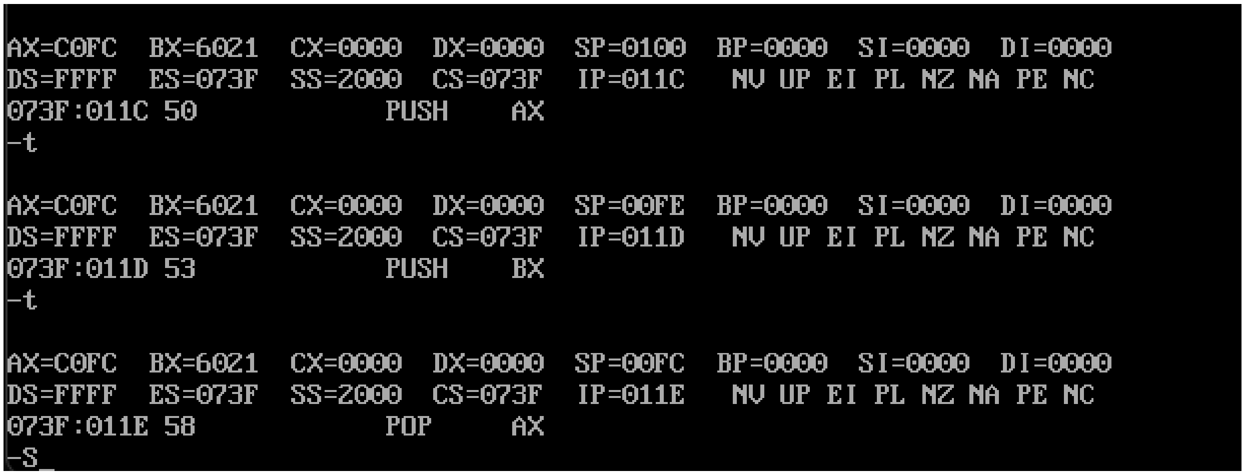
输入汇编代码，查看`ffff:0 f`区间的内容，发现是主板ROM上存储的生产日期。

之后逐步执行，查看结果：



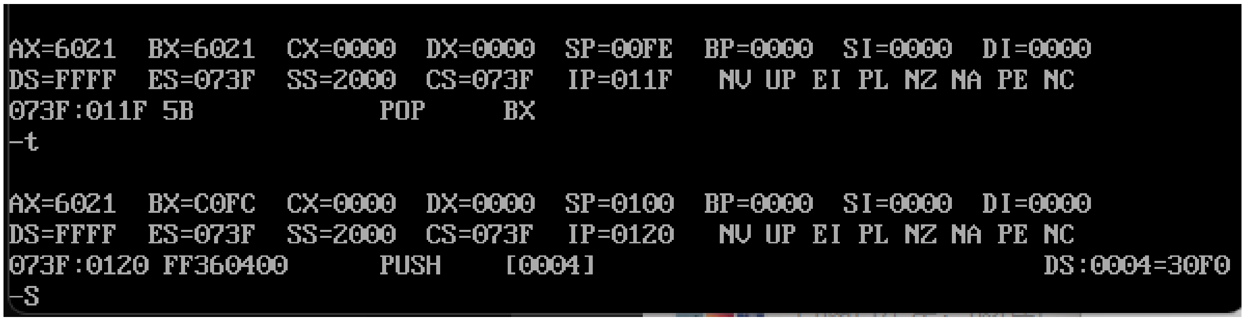
```
AX=2000  BX=0000  CX=0000  DX=0000  SP=0100  BP=0000  SI=0000  DI=0000
DS=FFFF  ES=073F  SS=2000  CS=073F  IP=010D   NV UP EI PL NZ NA PO NC
073F:010D A10000        MOV     AX,[0000]                          DS:0000=C0EA
-t

AX=C0EA  BX=0000  CX=0000  DX=0000  SP=0100  BP=0000  SI=0000  DI=0000
DS=FFFF  ES=073F  SS=2000  CS=073F  IP=0110   NV UP EI PL NZ NA PO NC
073F:0110 03060200      ADD     AX,[0002]                          DS:0002=0012
```

注意此步：由于ax的长度为十六位，两字节，所以执行`mov ax,[0]`时，要移入两字节的内容。但是，由于在内存之中一个字地位存储在内存低位，高位存储在内存高位。所以，`ax`之中，`(ah) = C0,(al) = ea`



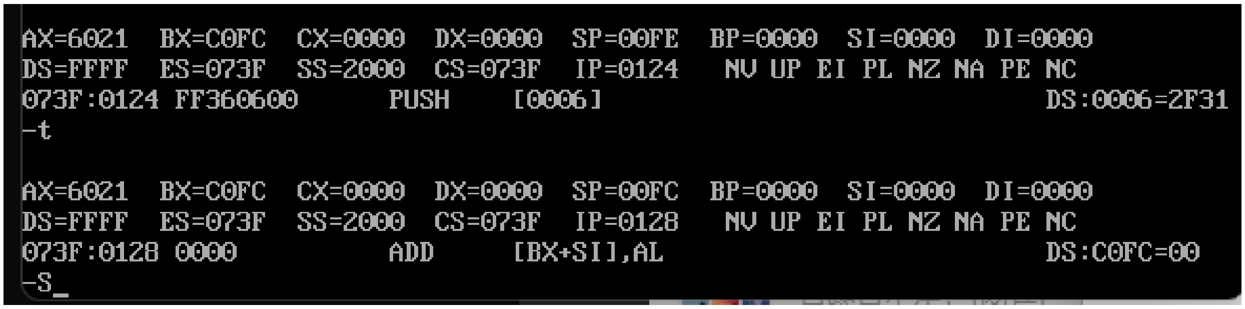
```
AX=C0FC  BX=6021  CX=0000  DX=0000  SP=0100  BP=0000  SI=0000  DI=0000
DS=FFFF  ES=073F  SS=2000  CS=073F  IP=011C   NV UP EI PL NZ NA PE NC
073F:011C 50            PUSH    AX
-t

AX=C0FC  BX=6021  CX=0000  DX=0000  SP=00FE  BP=0000  SI=0000  DI=0000
DS=FFFF  ES=073F  SS=2000  CS=073F  IP=011D   NV UP EI PL NZ NA PE NC
073F:011D 53            PUSH    BX
-t

AX=C0FC  BX=6021  CX=0000  DX=0000  SP=00FC  BP=0000  SI=0000  DI=0000
DS=FFFF  ES=073F  SS=2000  CS=073F  IP=011E   NV UP EI PL NZ NA PE NC
073F:011E 58            POP     AX
-S_
```

执行`push ax,push bx`之后，可以发现栈顶指针`sp`减小了2*2.



```
AX=6021  BX=6021  CX=0000  DX=0000  SP=00FE  BP=0000  SI=0000  DI=0000
DS=FFFF  ES=073F  SS=2000  CS=073F  IP=011F   NV UP EI PL NZ NA PE NC
073F:011F 5B            POP     BX
-t

AX=6021  BX=C0FC  CX=0000  DX=0000  SP=0100  BP=0000  SI=0000  DI=0000
DS=FFFF  ES=073F  SS=2000  CS=073F  IP=0120   NV UP EI PL NZ NA PE NC
073F:0120 FF360400      PUSH    [0004]                             DS:0004=30F0
-S
```

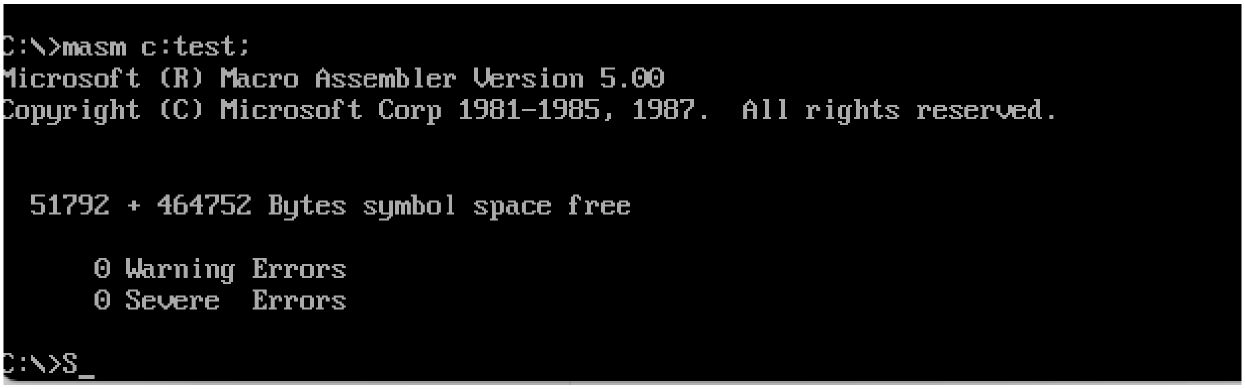
执行`pop ax,pop bx`之后，`ax bx`交换了数值。



```
AX=6021  BX=C0FC  CX=0000  DX=0000  SP=00FE  BP=0000  SI=0000  DI=0000
DS=FFFF  ES=073F  SS=2000  CS=073F  IP=0124   NV UP EI PL NZ NA PE NC
073F:0124 FF360600      PUSH    [0006]                             DS:0006=2F31
-t

AX=6021  BX=C0FC  CX=0000  DX=0000  SP=00FC  BP=0000  SI=0000  DI=0000
DS=FFFF  ES=073F  SS=2000  CS=073F  IP=0128   NV UP EI PL NZ NA PE NC
073F:0128 0000          ADD     [BX+SI],AL                         DS:C0FC=00
-S_
```

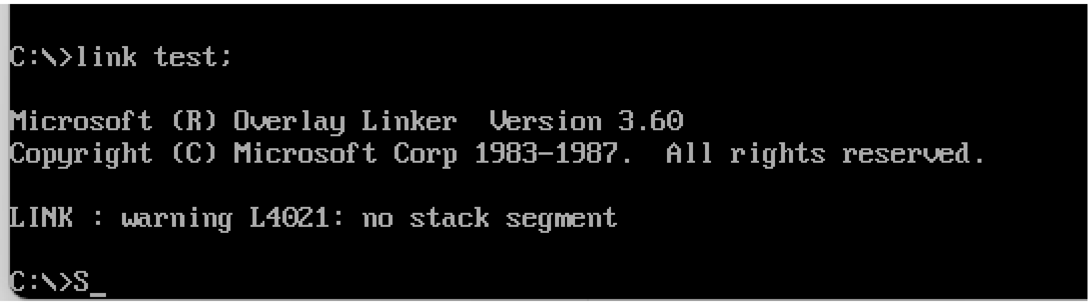
快捷编译，链接汇编程序

1. 创建`.asm`汇编程序源文件
2. 执行DOSbox，挂载c盘到masm以及link所在的文件夹
3. 编译：`masm c:test;`.**注意：1.最后的分号 2.由于将c盘挂载到了debug目录之下，将根地址更改为c：即可。**



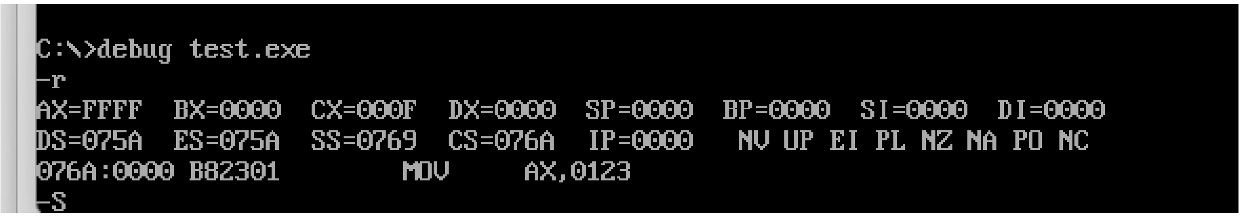
```
C:\>masm c:test;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51792 + 464752 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>S_
```

5. 连接：`link test;`



```
C:\>link test;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

LINK : warning L4021: no stack segment

C:\>S_
```

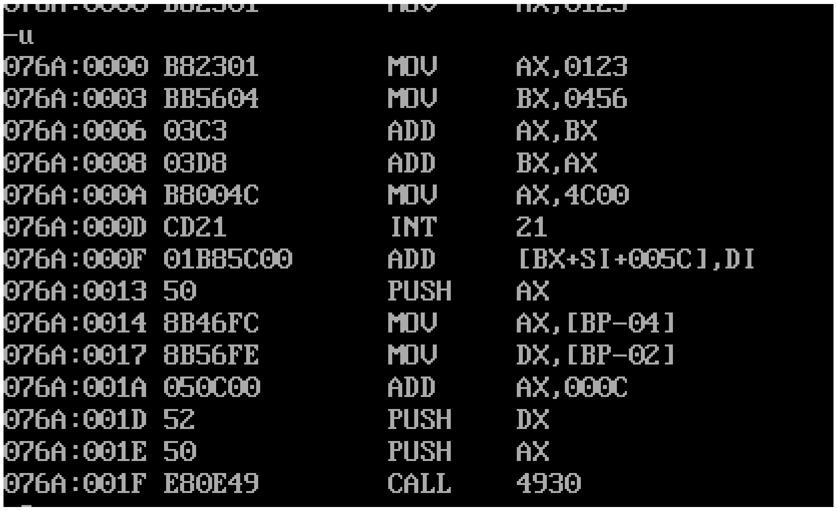
6. debug: `debug test.exe`



```
C:\>debug test.exe
-r
AX=FFFF  BX=0000  CX=000F  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0000   NV UP EI PL NZ NA PO NC
076A:0000 B82301        MOV     AX,0123
-S
```

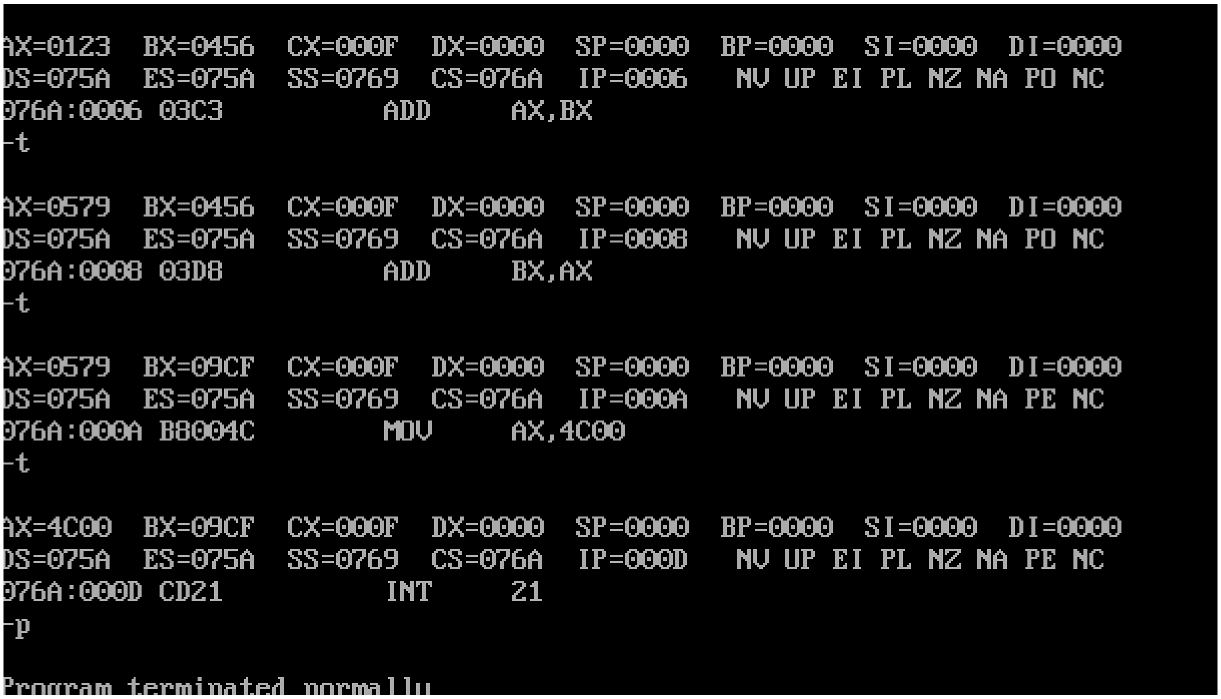
注意:

1. 此时`cs,ip`以及各个寄存器已经指向程序所在地址,同时注意下一条要执行的指令便是我们所写的程序.
2. 使用`u`命令查看代码:段地址便是`cs`所在地址



```
076A:0000 B82301        MOV     AX,0123
-u
076A:0000 B82301        MOV     AX,0123
076A:0003 BB5604        MOV     BX,0456
076A:0006 03C3          ADD     AX,BX
076A:0008 03D8          ADD     BX,AX
076A:000A B8004C        MOV     AX,4C00
076A:000D CD21          INT     21
076A:000F 01B85C00      ADD     [BX+SI+005C],DI
076A:0013 50            PUSH    AX
076A:0014 8B46FC        MOV     AX,[BP-04]
076A:0017 8B56FE        MOV     DX,[BP-02]
076A:001A 050C00        ADD     AX,000C
076A:001D 52            PUSH    DX
076A:001E 50            PUSH    AX
076A:001F E80E49        CALL    4930
-
```

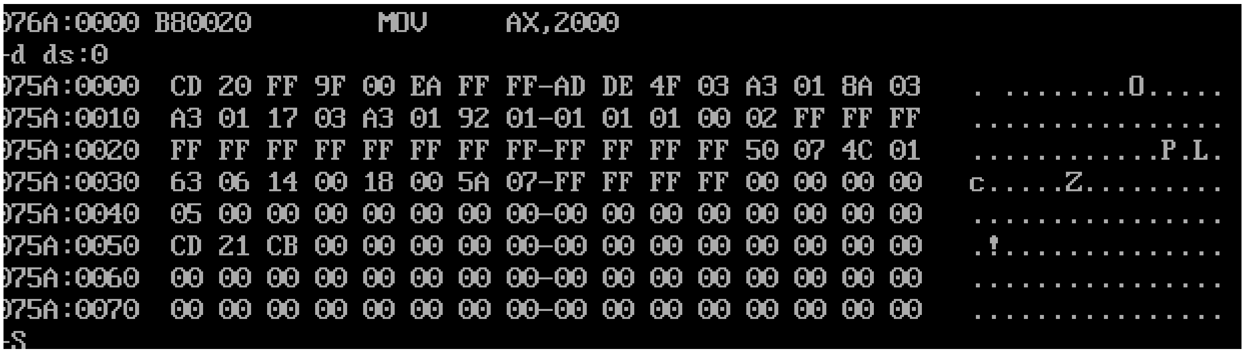
3. 使用`t`命令执行每一行代码,**注意最后一行的`int 21h`必须使用`p`命令执行**



```
AX=0123  BX=0456  CX=000F  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0006   NV UP EI PL NZ NA PO NC
076A:0006 03C3          ADD     AX,BX
-t

AX=0579  BX=0456  CX=000F  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0008   NV UP EI PL NZ NA PO NC
076A:0008 03D8          ADD     BX,AX
-t

AX=0579  BX=09CF  CX=000F  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=000A   NV UP EI PL NZ NA PE NC
076A:000A B8004C        MOV     AX,4C00
-t

AX=4C00  BX=09CF  CX=000F  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=000D   NV UP EI PL NZ NA PE NC
076A:000D CD21          INT     21
-p

Program terminated normally
```

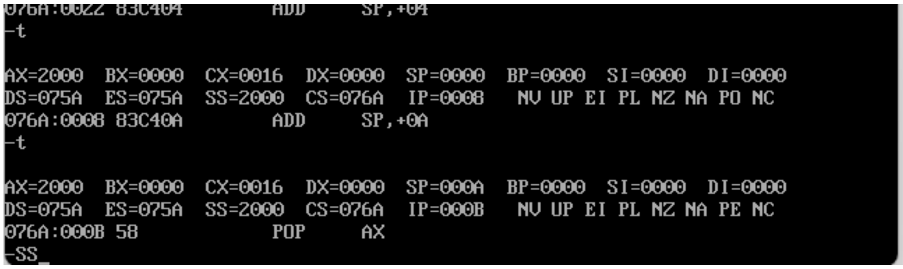
王爽P94页实验

1. 查看PSP之中的内容:以`CD 20`开头



```
076A:0000 B80020        MOV     AX,2000
-d ds:0
075A:0000  CD 20 FF 9F 00 EA FF FF-AD DE 4F 03 A3 01 8A 03   . ........O.....
075A:0010  A3 01 17 03 A3 01 92 01-01 01 01 00 02 FF FF FF   ................
075A:0020  FF FF FF FF FF FF FF FF-FF FF FF FF 50 07 4C 01   ............P.L.
075A:0030  63 06 14 00 18 00 5A 07-FF FF FF FF 00 00 00 00   c.....Z.........
075A:0040  05 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
075A:0050  CD 21 CB 00 00 00 00 00-00 00 00 00 00 00 00 00   .!..............
075A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
075A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-S
```

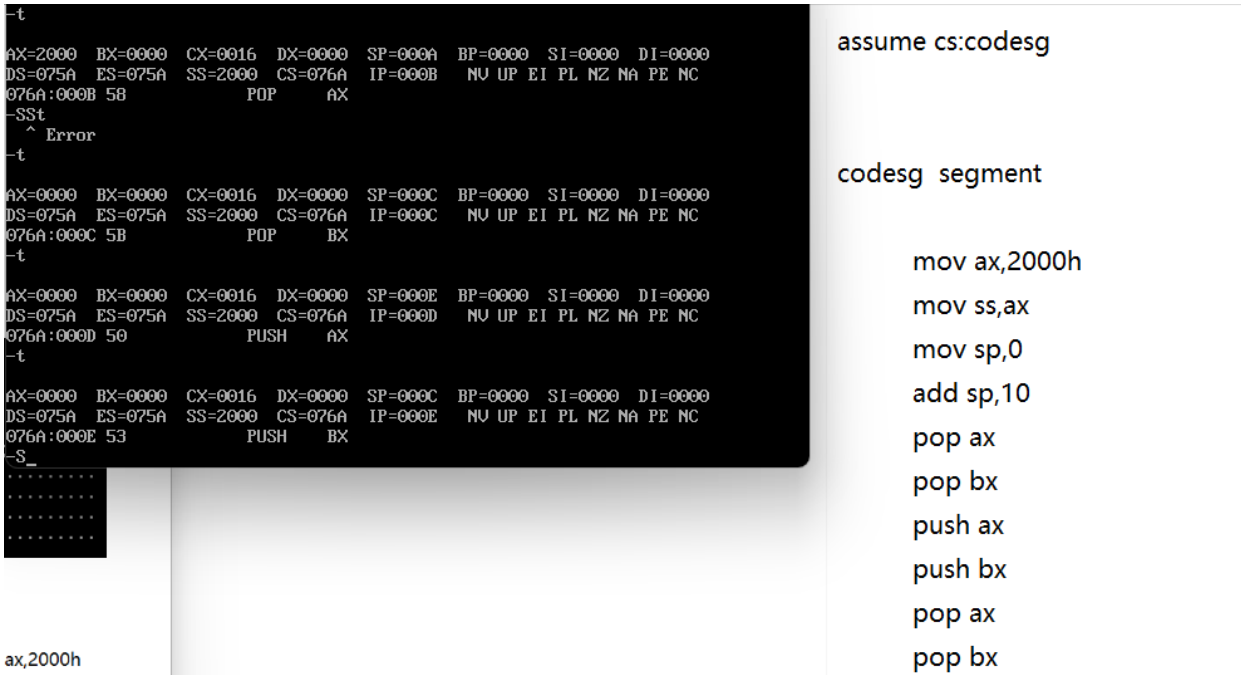
2. 执行程序



```
076A:0022 83C404        ADD     SP,+04
-t

AX=2000  BX=0000  CX=0016  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=2000  CS=076A  IP=0008   NV UP EI PL NZ NA PO NC
076A:0008 83C40A        ADD     SP,+0A
-t

AX=2000  BX=0000  CX=0016  DX=0000  SP=000A  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=2000  CS=076A  IP=000B   NV UP EI PL NZ NA PE NC
076A:000B 58            POP     AX
-SS_
```

```
mov ax,2000h
mov ss,ax
mov sp,0
add sp,10
pop ax
```

似乎在进行编译的时候,如果数字后边不加`h`会认为是十进制.此处`(SP) == 0Ah`.



```
-t

AX=2000  BX=0000  CX=0016  DX=0000  SP=000A  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=2000  CS=076A  IP=000B   NV UP EI PL NZ NA PE NC
076A:000B 58            POP     AX
-SSt
  ^ Error
-t

AX=0000  BX=0000  CX=0016  DX=0000  SP=000C  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=2000  CS=076A  IP=000C   NV UP EI PL NZ NA PE NC
076A:000C 5B            POP     BX
-t

AX=0000  BX=0000  CX=0016  DX=0000  SP=000E  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=2000  CS=076A  IP=000D   NV UP EI PL NZ NA PE NC
076A:000D 50            PUSH    AX
-t

AX=0000  BX=0000  CX=0016  DX=0000  SP=000C  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=2000  CS=076A  IP=000E   NV UP EI PL NZ NA PE NC
076A:000E 53            PUSH    BX
-S_
```

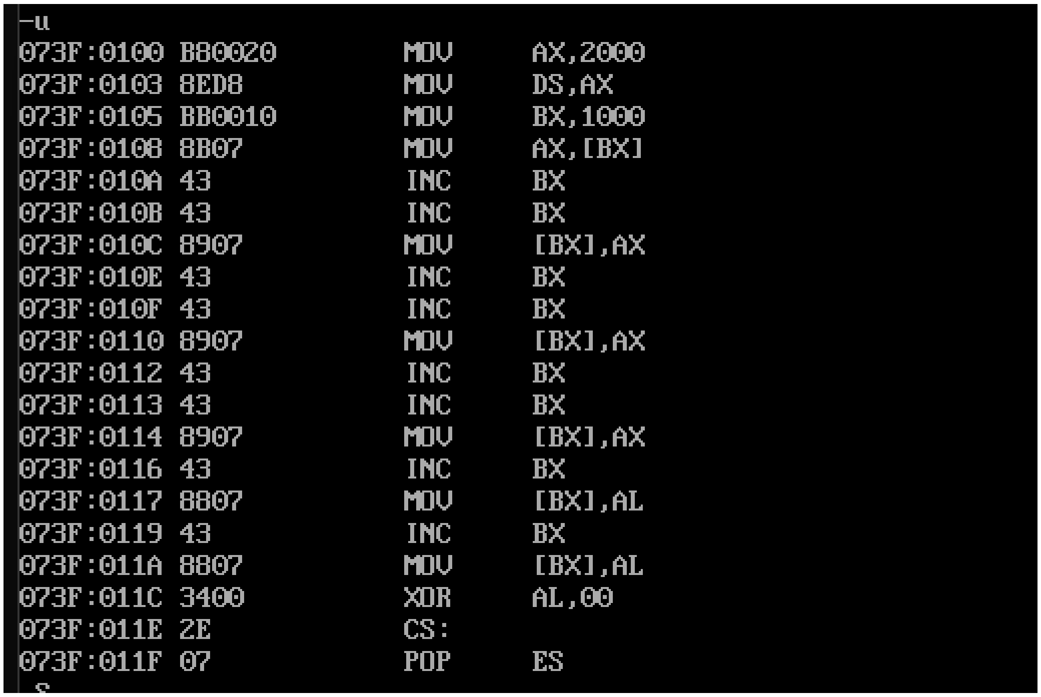
```
ax,2000h
```

```
assume cs:codesg


codesg  segment

        mov ax,2000h
        mov ss,ax
        mov sp,0
        add sp,10
        pop ax
        pop bx
        push ax
        push bx
        pop ax
        pop bx
```

继续执行,似乎出现了栈溢出.

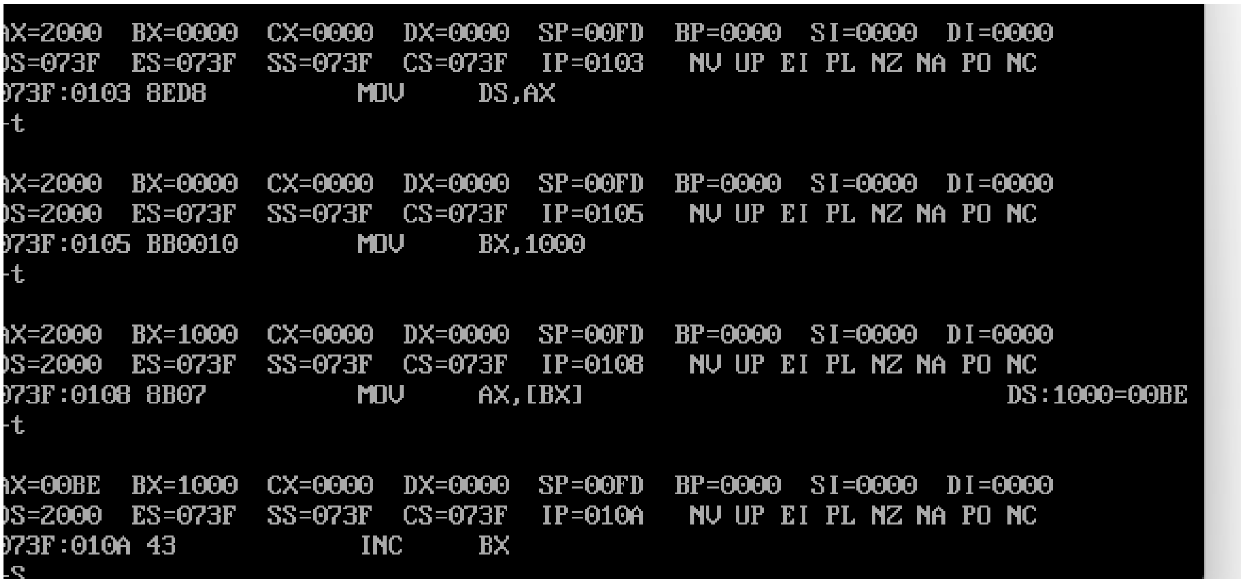
王爽P97页实验

1. 输入程序



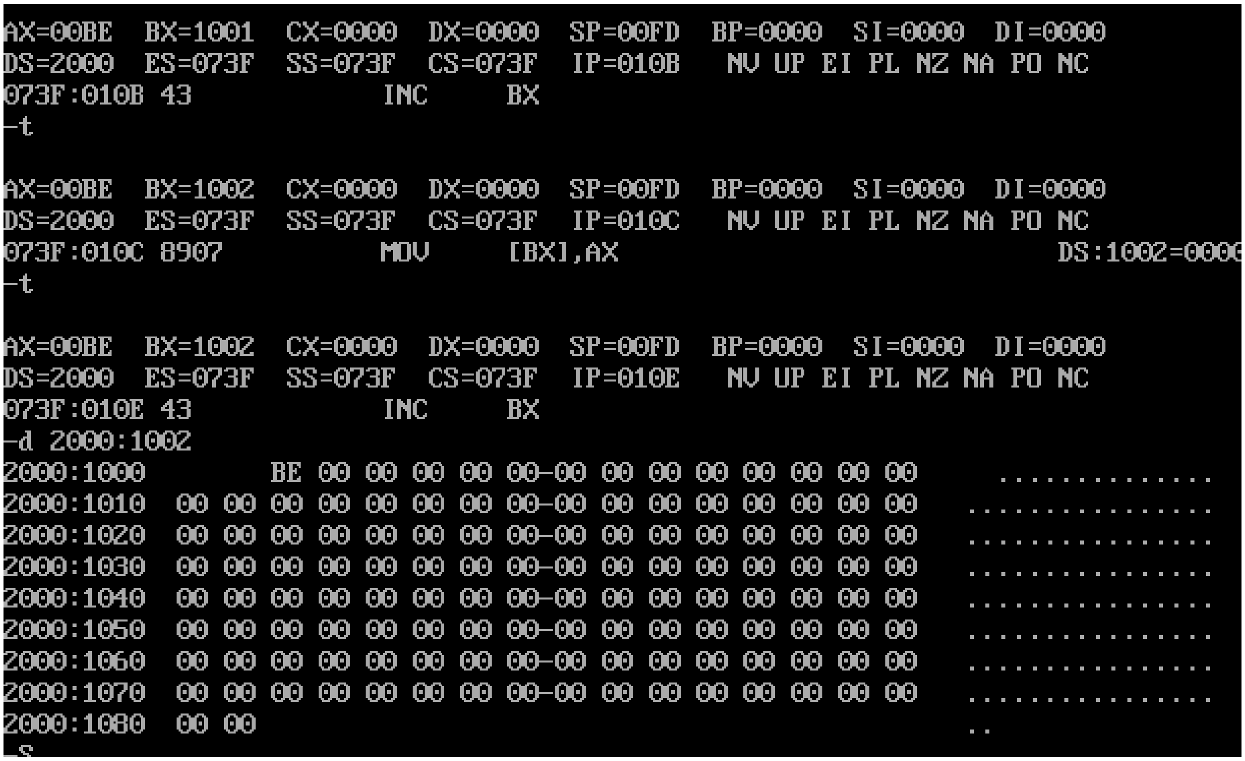
```
-u
073F:0100 B80020        MOV     AX,2000
073F:0103 8ED8          MOV     DS,AX
073F:0105 BB0010        MOV     BX,1000
073F:0108 8B07          MOV     AX,[BX]
073F:010A 43            INC     BX
073F:010B 43            INC     BX
073F:010C 8907          MOV     [BX],AX
073F:010E 43            INC     BX
073F:010F 43            INC     BX
073F:0110 8907          MOV     [BX],AX
073F:0112 43            INC     BX
073F:0113 43            INC     BX
073F:0114 8907          MOV     [BX],AX
073F:0116 43            INC     BX
073F:0117 8807          MOV     [BX],AL
073F:0119 43            INC     BX
073F:011A 8807          MOV     [BX],AL
073F:011C 3400          XOR     AL,00
073F:011E 2E            CS:
073F:011F 07            POP     ES
```

2. 逐步执行,查看结果



```
AX=2000  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0103   NV UP EI PL NZ NA PO NC
073F:0103 8ED8          MOV     DS,AX
-t

AX=2000  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=2000  ES=073F  SS=073F  CS=073F  IP=0105   NV UP EI PL NZ NA PO NC
073F:0105 BB0010        MOV     BX,1000
-t

AX=2000  BX=1000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=2000  ES=073F  SS=073F  CS=073F  IP=0108   NV UP EI PL NZ NA PO NC
073F:0108 8B07          MOV     AX,[BX]                            DS:1000=00BE
-t

AX=00BE  BX=1000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=2000  ES=073F  SS=073F  CS=073F  IP=010A   NV UP EI PL NZ NA PO NC
073F:010A 43            INC     BX
-S
```

可以发现:`[bx]`**指令的含义是:获取**`(ds*16+bx)`**处的内存**



```
AX=00BE  BX=1001  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=2000  ES=073F  SS=073F  CS=073F  IP=010B   NV UP EI PL NZ NA PO NC
073F:010B 43            INC     BX
-t

AX=00BE  BX=1002  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=2000  ES=073F  SS=073F  CS=073F  IP=010C   NV UP EI PL NZ NA PO NC
073F:010C 8907          MOV     [BX],AX                            DS:1002=000
-t

AX=00BE  BX=1002  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=2000  ES=073F  SS=073F  CS=073F  IP=010E   NV UP EI PL NZ NA PO NC
073F:010E 43            INC     BX
-d 2000:1002
2000:1000        BE 00 00 00 00 00-00 00 00 00 00 00 00 00     ..............
2000:1010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
2000:1020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
2000:1030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
2000:1040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
2000:1050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
2000:1060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
2000:1070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
2000:1080  00 00                                             ..
-S
```

loop指令

1. 使用标号来表示实际上要跳转的地址.在DOS之中,标号实际上表示要跳转执行程序行的地址.
2. 使用`g`命令可以执行到指定位置.比如,`g 1012`表示一直执行到1012地址
3. 使用`p`命令可以跳过重复的loop过程

loop实验：P121

```
;1. 向内存0:200~0:23f依次传送数据0~63、
```

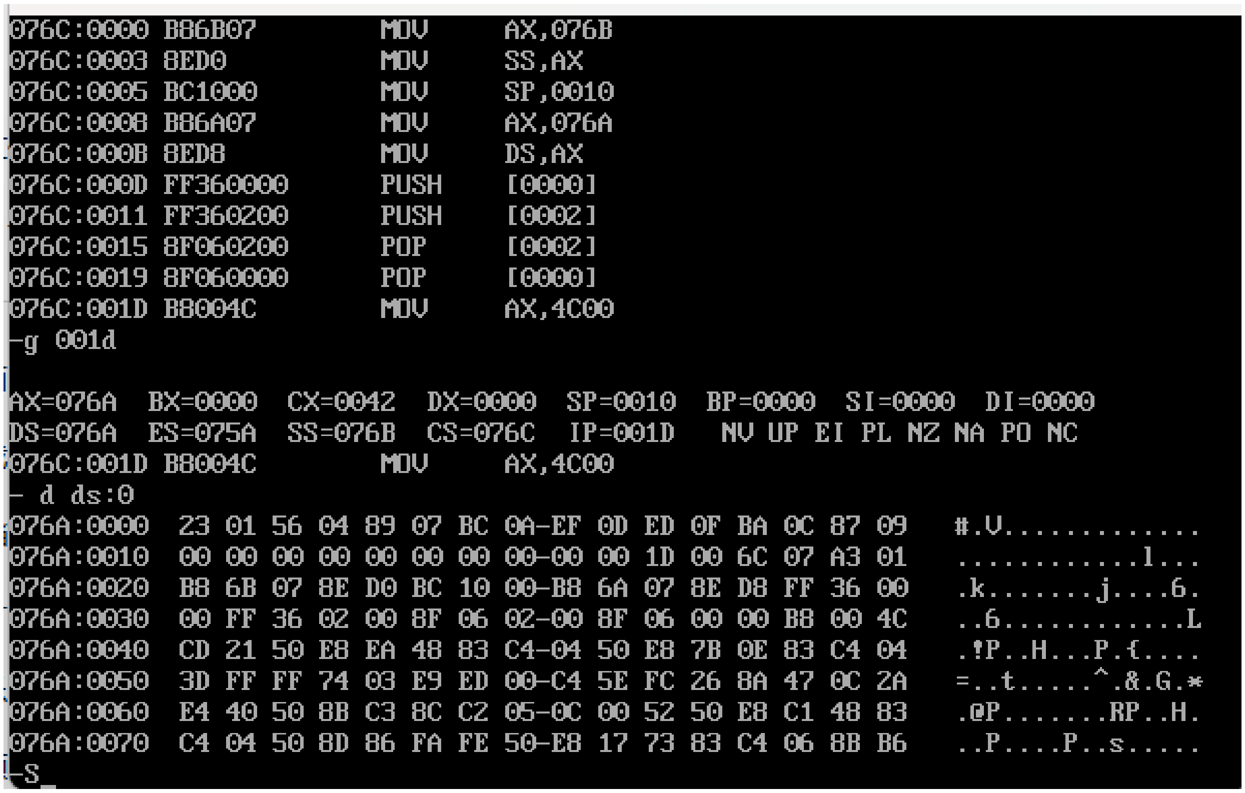
```
;2. 使用九条指令完成第一题
```

```
;3. 调试给出的程序，跟踪运行成果
```

具有多个段的程序

王爽P134页实验1

```
assume cs:code,ds:data,ss:stack

data segment
    dw 0123h,0456h,0789h,0abch,0defh,0fedh,0cbah,0987h
data ends

stack segment
    dw 0,0,0,0,0,0,0,0
stack ends

code segment

start:
    mov ax,stack
    mov ss,ax
    mov sp,16

    mov ax,data
    mov ds,ax

    push ds:[0]
    push ds:[2]
    pop ds:[2]
    pop ds:[0]

    mov ax,4c00h
    int 21h

code ends

end start
```

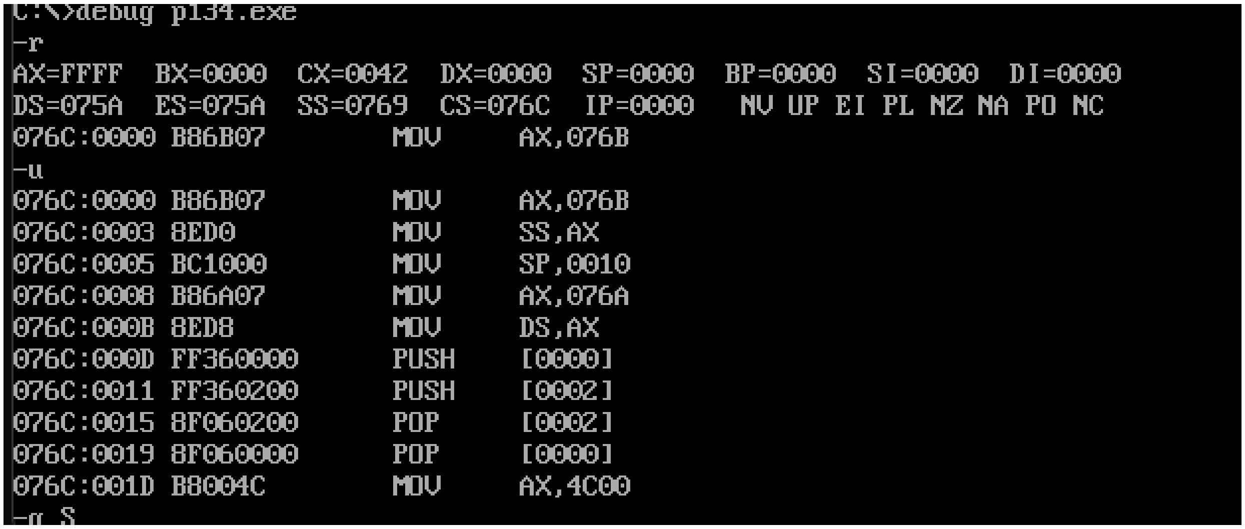


```
076C:0000 B86B07        MOV     AX,076B
076C:0003 8ED0          MOV     SS,AX
076C:0005 BC1000        MOV     SP,0010
076C:0008 B86A07        MOV     AX,076A
076C:000B 8ED8          MOV     DS,AX
076C:000D FF360000      PUSH    [0000]
076C:0011 FF360200      PUSH    [0002]
076C:0015 8F060200      POP     [0002]
076C:0019 8F060000      POP     [0000]
076C:001D B8004C        MOV     AX,4C00
-g 001d

AX=076A  BX=0000  CX=0042  DX=0000  SP=0010  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=076B  CS=076C  IP=001D   NV UP EI PL NZ NA PO NC
076C:001D B8004C        MOV     AX,4C00
- d ds:0
076A:0000  23 01 56 04 89 07 BC 0A-EF 0D ED 0F BA 0C 87 09   #.V.............
076A:0010  00 00 00 00 00 00 00 00-00 00 1D 00 6C 07 A3 01   ............l...
076A:0020  B8 6B 07 8E D0 BC 10 00-B8 6A 07 8E D8 FF 36 00   .k.......j....6.
076A:0030  00 FF 36 02 00 8F 06 02-00 8F 06 00 00 B8 00 4C   ..6............L
076A:0040  CD 21 50 E8 EA 48 83 C4-04 50 E8 7B 0E 83 C4 04   .!P..H...P.{....
076A:0050  3D FF FF 74 03 E9 ED 00-C4 5E FC 26 8A 47 0C 2A   =..t.....^.&.G.*
076A:0060  E4 40 50 8B C3 8C C2 05-0C 00 52 50 E8 C1 48 83   .@P.......RP..H.
076A:0070  C4 04 50 8D 86 FA FE 50-E8 17 73 83 C4 06 8B B6   ..P....P..s.....
-S_
```

1. `(ds = x)`,则`(ss) = x+1,(cs) = x+2`

实验二

```
assume cs:code,ds:data,ss:stack

data segment
    dw 0123h,0456h
data ends

stack segment
    dw 0,0
stack ends

code segment

start:
    mov ax,stack
    mov ss,ax
    mov sp,16       ;按理说sp = 4即可，不知道为什么要开辟这么大的栈空间

    mov ax,data
    mov ds,ax

    push ds:[0]
    push ds:[2]

    pop ds:[2]
    pop ds:[0]

    mov ax,4c00h
    int 21h

code ends
end start
```



```
C:\>debug p134.exe
-r
AX=FFFF  BX=0000  CX=0042  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076C  IP=0000   NV UP EI PL NZ NA PO NC
076C:0000 B86B07        MOV     AX,076B
-u
076C:0000 B86B07        MOV     AX,076B
076C:0003 8ED0          MOV     SS,AX
076C:0005 BC1000        MOV     SP,0010
076C:0008 B86A07        MOV     AX,076A
076C:000B 8ED8          MOV     DS,AX
076C:000D FF360000      PUSH    [0000]
076C:0011 FF360200      PUSH    [0002]
076C:0015 8F060200      POP     [0002]
076C:0019 8F060000      POP     [0000]
076C:001D B8004C        MOV     AX,4C00
-g s
```

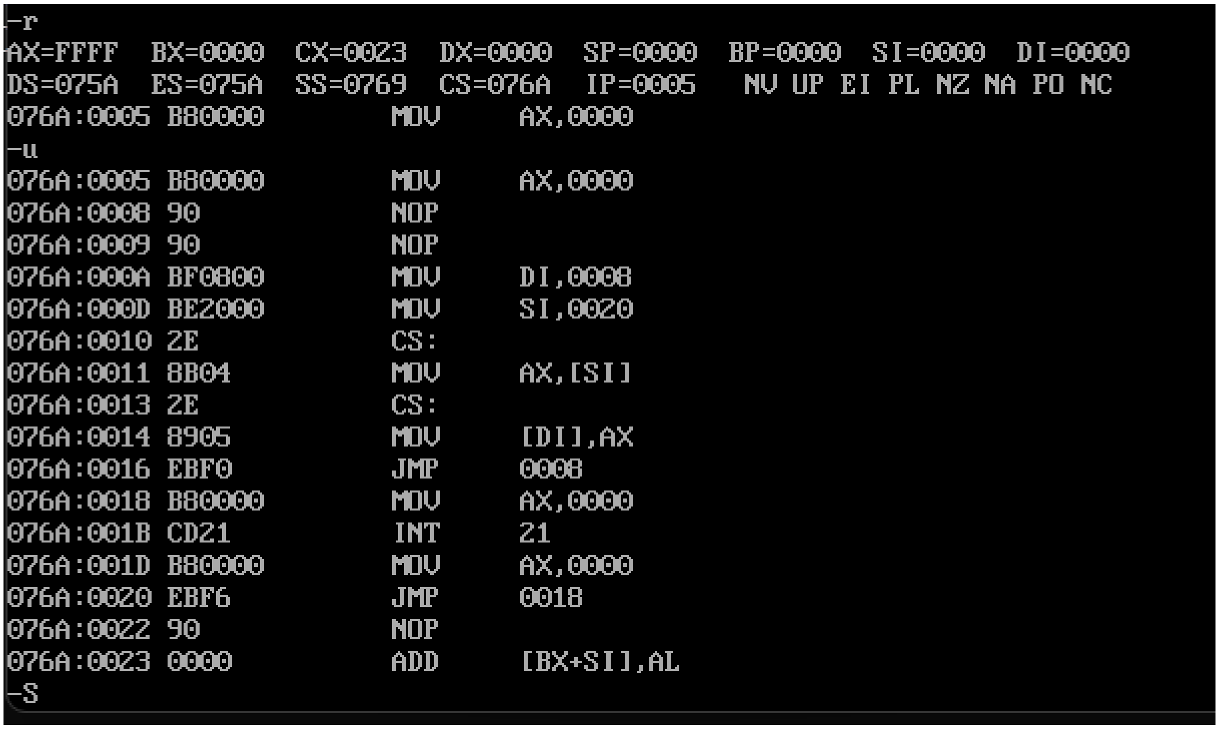
查看汇编指令,执行.

JMP指令

P187页实验

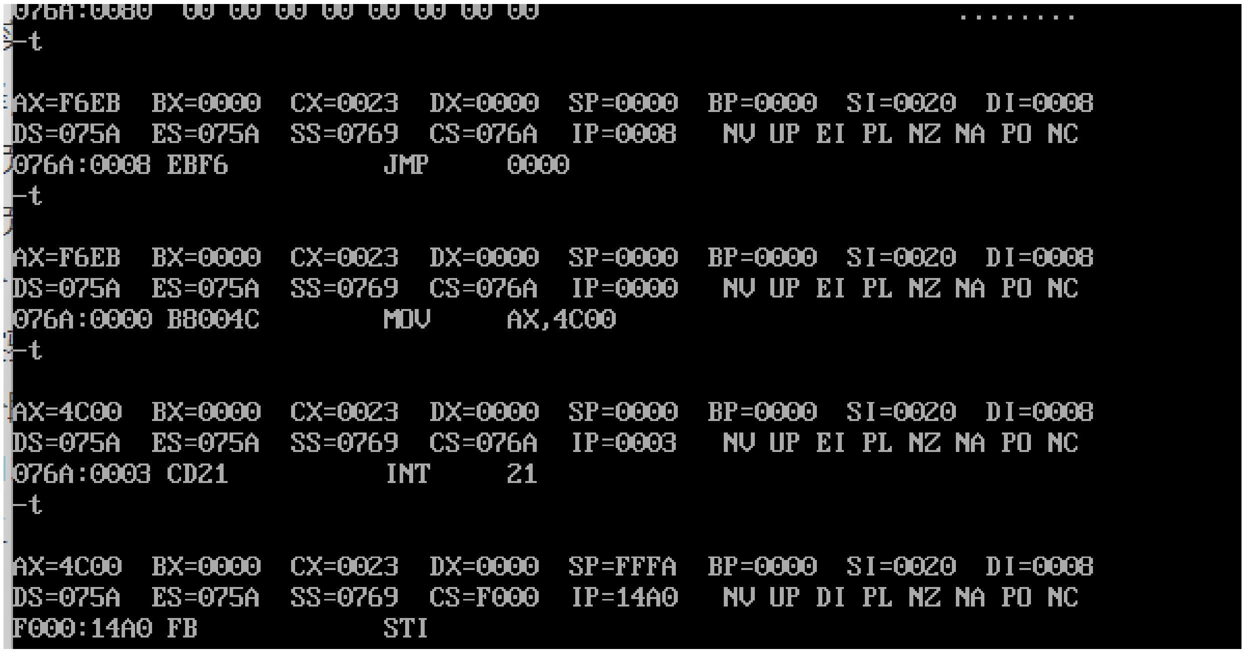
```
assume cs:codesg

codesg segment

    mov ax,4c00h
    int 21h

start:
    mov ax,0
s:  nop
    nop

    mov di,offset s
    mov si,offset s2
    mov ax,cs:[si]
    mov cs:[di],ax
s0: jmp short s

s1: mov ax,0
    int 21h
    mov ax,0

s2: jmp short s1
    nop

codesg ends
end start
```

在debug之中查看编译后的代码



```
-r
AX=FFFF  BX=0000  CX=0023  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0005   NV UP EI PL NZ NA PO NC
076A:0005 B80000        MOV     AX,0000
-u
076A:0005 B80000        MOV     AX,0000
076A:0008 90            NOP
076A:0009 90            NOP
076A:000A BF0800        MOV     DI,0008
076A:000D BE2000        MOV     SI,0020
076A:0010 2E            CS:
076A:0011 8B04          MOV     AX,[SI]
076A:0013 2E            CS:
076A:0014 8905          MOV     [DI],AX
076A:0016 EBF0          JMP     0008
076A:0018 B80000        MOV     AX,0000
076A:001B CD21          INT     21
076A:001D B80000        MOV     AX,0000
076A:0020 EBF6          JMP     0018
076A:0022 90            NOP
076A:0023 0000          ADD     [BX+SI],AL
-S
```

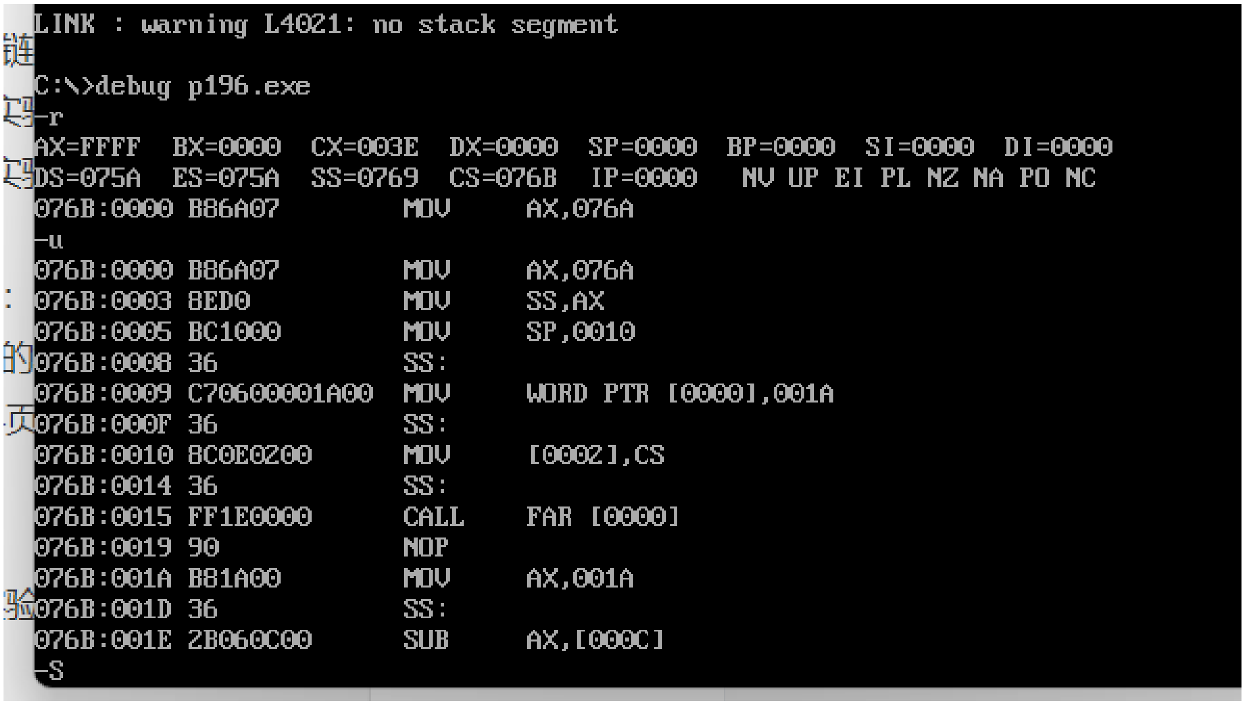
执行了`mov cs:[di],ax`之后，`(cs:0008 = EBF6)`，向前跳转8位。在标号S2处，向前跳转8位为`s1`的地址，而在地址0008处，向前跳转8位的地址为0000



```
076A:0080  00 00 00 00 00 00 00 00                    ........
-t

AX=F6EB  BX=0000  CX=0023  DX=0000  SP=0000  BP=0000  SI=0020  DI=0008
DS=075A  ES=075A  SS=0769  CS=076A  IP=0008   NV UP EI PL NZ NA PO NC
076A:0008 EBF6          JMP     0000
-t

AX=F6EB  BX=0000  CX=0023  DX=0000  SP=0000  BP=0000  SI=0020  DI=0008
DS=075A  ES=075A  SS=0769  CS=076A  IP=0000   NV UP EI PL NZ NA PO NC
076A:0000 B8004C        MOV     AX,4C00
-t

AX=4C00  BX=0000  CX=0023  DX=0000  SP=0000  BP=0000  SI=0020  DI=0008
DS=075A  ES=075A  SS=0769  CS=076A  IP=0003   NV UP EI PL NZ NA PO NC
076A:0003 CD21          INT     21
-t

AX=4C00  BX=0000  CX=0023  DX=0000  SP=FFFA  BP=0000  SI=0020  DI=0008
DS=075A  ES=075A  SS=0769  CS=F000  IP=14A0   NV UP DI PL NZ NA PO NC
F000:14A0 FB            STI
```

P196

```
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
```



```
LINK : warning L4021: no stack segment

C:\>debug p196.exe
-r
AX=FFFF  BX=0000  CX=003E  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
-u
076B:0000 B86A07        MOV     AX,076A
076B:0003 8ED0          MOV     SS,AX
076B:0005 BC1000        MOV     SP,0010
076B:0008 36            SS:
076B:0009 C70600001A00  MOV     WORD PTR [0000],001A
076B:000F 36            SS:
076B:0010 8C0E0200      MOV     [0002],CS
076B:0014 36            SS:
076B:0015 FF1E0000      CALL    FAR [0000]
076B:0019 90            NOP
076B:001A B81A00        MOV     AX,001A
076B:001D 36            SS:
076B:001E 2B060C00      SUB     AX,[000C]
-S
```

考试

可能涉及到的内容

- 保存中断向量，执行终端过程之中的基本操作
- 显示回车换行
- 显示十进制数字

1. 终端程序的编写：
   1. CPU执行中断程序主要包含以下几个步骤：
      1. `pushf`：将全部标志寄存器入栈
      2. `push cs,push ip`：将程序指针入栈
      3. 设置`tf = 0,if = 0`：其中，if寄存器用于判断是否执行外中断例程，tf负责判断是否执行单步中断
      4. `cs = word ptr 4n+2`
      5. `if = word ptr 4n`
   2. 编写中断程序主要需要一下几个步骤：
      1. 将中断程序复制到`0：200`地址处
      2. 将中断向量表更改为程序起始位置

3. 一个例子：0号中断的中断程序

```
stack segment stack
    db 128 dup(0)
stack ends
data segment
    db 128 dup(0)
data ends
code segment
    assume ss:stack,cs:code,ds:data

start:
        mov ax,cs
        mov ds,ax
        mov si,offset func          ; ds:si指向复制的源地址
        mov ax,0
        mov es,ax                   ; !注意不能直接将立即数放入到段寄存器之中
        mov di,0200h                ; es:di指向目标地址
        cld                         ; 设置复制向执行
        mov cx,offset func_end-offset func_start
        rep movsb

        mov ax,0
        mov ds,ax
        mov bx,0
        mov [bx],word ptr 0200h
        mov [bx+2],word ptr 0h      ; 将中断程序的地址放入到中断向量表

        mov ax,4c00h
        int 21h
func:
        jmp func_start              ; 程序开始不是可以执行的代码，所以需要跳转
        string db "overflow!"
func_start:
        push dx
        push ax
        push ds
        mov dx,seg string           ; 获取string的段地址
        mov dx,offset string
        mov ah,09h
        int 21h
        pop ds
        pop ax
        pop dx
        mov ax,4c00h
        int 21h
func_end:
        nop                         ; 需要返回DOS控制
code ends
end start
```

2. 显示十进制阿拉伯数字

3. 数据段中有字符串变量S，长度为100。统计其中小写字母的个数，并以16进制的方式输出个数值。

```
data segment
            s db dup("?")
data ends
stack segment stack
            512 db dup(0)
stack ends
code segment
            assume cs:code,ds:data,ss:stack
start:
            mov ax,data
            mov ds,ax
            mov ax,stack
            mov ss,ax
            mov bx,offset s
            mov cx,99
            mov si,0
            mov dx,0
s:
            mov ax,[bx+si]
            inc si
            inc si
            cmp ax,'a'
            jb continue
            cmp ax,'z'
            ja continue
            inc dx
continue:
            loop s

                                ; 接下来进行输出个数
            mov ax,dx
            xor dx
PrintDIgit:
            and ah,0f0h         ; 保存最高位
            push cx
            mov cl,4
            shl ah,cl
            pop cx
            cmp ah,9
            jna Print
            add ah,07h
Print:
            mov ah,al
            add al,30h
            mov ah,02h          ; 调用输出字符中断程序，忘了是哪个编号了
            int 21h
            pop ax
            mov ax,cx
            jcxz func_end
            jmp PrintDIgit
func_end:
            mov ax,4c00h
            int 21h

```

4. 统计16位数ax之中1的个数

```
; 以下为核心部分代码
            mov bx,1
            mov ax,4c12h
            mov dx,0
            mov cx,15
s:
            test ax,bx
            jnz count
            shl bx,1
count:
            inc dx
```

5. 如果`ax`的最低位为0，请将`bx`的次低位置1，否则，将最高位取反

```
            test ax,1
            jz s1
            xor ax, 10000000b
s1:
            xor bx,00000010b
```

6. 将数据段之中长度为100的学生成绩数组score求平均值，并且以十进制的方式输出

```
stack segment stack
        128 db dup(0)
stack ends
data segment
        score dw 100 dup(?)
data ends
code segment
        mov ax,data
        mov ds,ax
        mov ax,stack
        mov ss,ax
        lea bx,score
        mov si,0
        mov dx,0
        mov cx,99
s:
        mov ax,[bx+si]
        div 64h             ; 进行除法以后，商放在al，余数放在ah
        mov ah,0
        add dx,al
        inc si
        inc si
        loop s

Print:
        ; 将数字按十进制进行输出
        mov bl,10
        div bl
        ; 需要保存商
        mov cl,al
        mov ch,0
        ; 如果余数存放在ah之中
        mov al,ah
        add al,30h
        mov dl,al
        mov ah,02h
        int 21h
        mov cx,ax
        jcxz print_end
        jmp Print
print_end:
        mov ax,4c00h
        int 21h

code ends
end start


```