

实验报告 Debug 用法实验

Hollow Man

一、实验环境

一台带有 MASM 软件的装有 Windows XP 系统的实验室计算机。

二、实验准备

用 Win+R 键打开“运行”，输入 cmd 并回车，打开“命令提示符”窗口程序。

在命令行中输入” cd /d D:\ “切换到 D 盘根目录。

输入“ MD JSL”创建 JSL 工作文件夹。

输入” cd JSL”切换到 JSL 工作目录

输入” copy C:\MASM* .”将程序文件拷贝进工作目录。

三、实验内容

a) 第一次

1. 任务 1

按提示用 a 命令输入指令，得到以下运行结果：

```
命令提示符 - debug
D:\JSL>debug
-a
1381:0100 mov ax,4E20
1381:0103 add ax,1416
1381:0106 mov bx,2000
1381:0109 add ax,bx
1381:010B mov bx,ax
1381:010D add ax,001A
1381:0110 mov bx,0026
1381:0113 add al,bl
1381:0115 add ah,bl
1381:0117 add bh,al
1381:0119 mov ah,0
1381:011B add al,bl
1381:011D add al,9C
1381:011F
-t
AX=4E20 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=1381 IP=0103  NU UP EI PL NZ NA PO NC
1381:0103 051614      ADD     AX,1416
-t
AX=6236 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=1381 IP=0106  NU UP EI PL NZ NA PE NC
1381:0106 BB0020      MOV     BX,2000
-t
AX=6236 BX=2000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=1381 IP=0109  NU UP EI PL NZ NA PE NC
1381:0109 01D8      ADD     AX,BX
-t
AX=8236 BX=2000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=1381 IP=010B  OU UP EI NG NZ NA PE NC
1381:010B 89C3      MOV     BX,AX
-t
AX=8236 BX=8236 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=1381 IP=010D  OU UP EI NG NZ NA PE NC
1381:010D 051A00      ADD     AX,001A
-t
AX=8250 BX=8236 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=1381 IP=0110  NU UP EI NG NZ AC PE NC
1381:0110 BB2600      MOV     BX,0026
-
```

```

命令提示符 - debug
AX=8236 BX=2000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=1381 IP=010B  OV UP EI NG NZ NA PE NC
1381:010B 89C3      MOV     BX,AX
-t
AX=8236 BX=8236 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=1381 IP=010D  OV UP EI NG NZ NA PE NC
1381:010D 051A00    ADD     AX,001A
-t
AX=8250 BX=8236 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=1381 IP=0110  NU UP EI NG NZ AC PE NC
1381:0110 BB2600    MOV     BX,0026
-t
AX=8250 BX=0026 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=1381 IP=0113  NU UP EI NG NZ AC PE NC
1381:0113 00D8      ADD     AL,BL
-t
AX=8276 BX=0026 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=1381 IP=0115  NU UP EI PL NZ NA PO NC
1381:0115 00DC      ADD     AH,BL
-t
AX=8876 BX=0026 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=1381 IP=0117  NU UP EI NG NZ NA PO NC
1381:0117 00C7      ADD     BH,AL
-t
AX=8876 BX=7626 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=1381 IP=0119  NU UP EI PL NZ NA PO NC
1381:0119 B400      MOV     AH,00
-t
AX=0076 BX=7626 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=1381 IP=011B  NU UP EI PL NZ NA PO NC
1381:011B 00D8      ADD     AL,BL
-t
AX=009C BX=7626 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=1381 IP=011D  OV UP EI NG NZ NA PE NC
1381:011D 049C      ADD     AL,9C
-t
AX=0038 BX=7626 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=1381 IP=011F  OV UP EI PL NZ AC PO CY
1381:011F 1300      ADC     AX,[BX+SI]
DS:7626=0000

```

由图示运行结果可以看到，每次执行完指令后，CS:IP 自动指向了下一个命令地址，并且按照指令的命令操作进行运算后，相关寄存器按照指令要求发生了数值的变化。

同理，用 e 命令直接输入机器码进内存，在运行前记得调整 CS:IP 指向命令开始语句的内存地址，也可得到同样的结果。

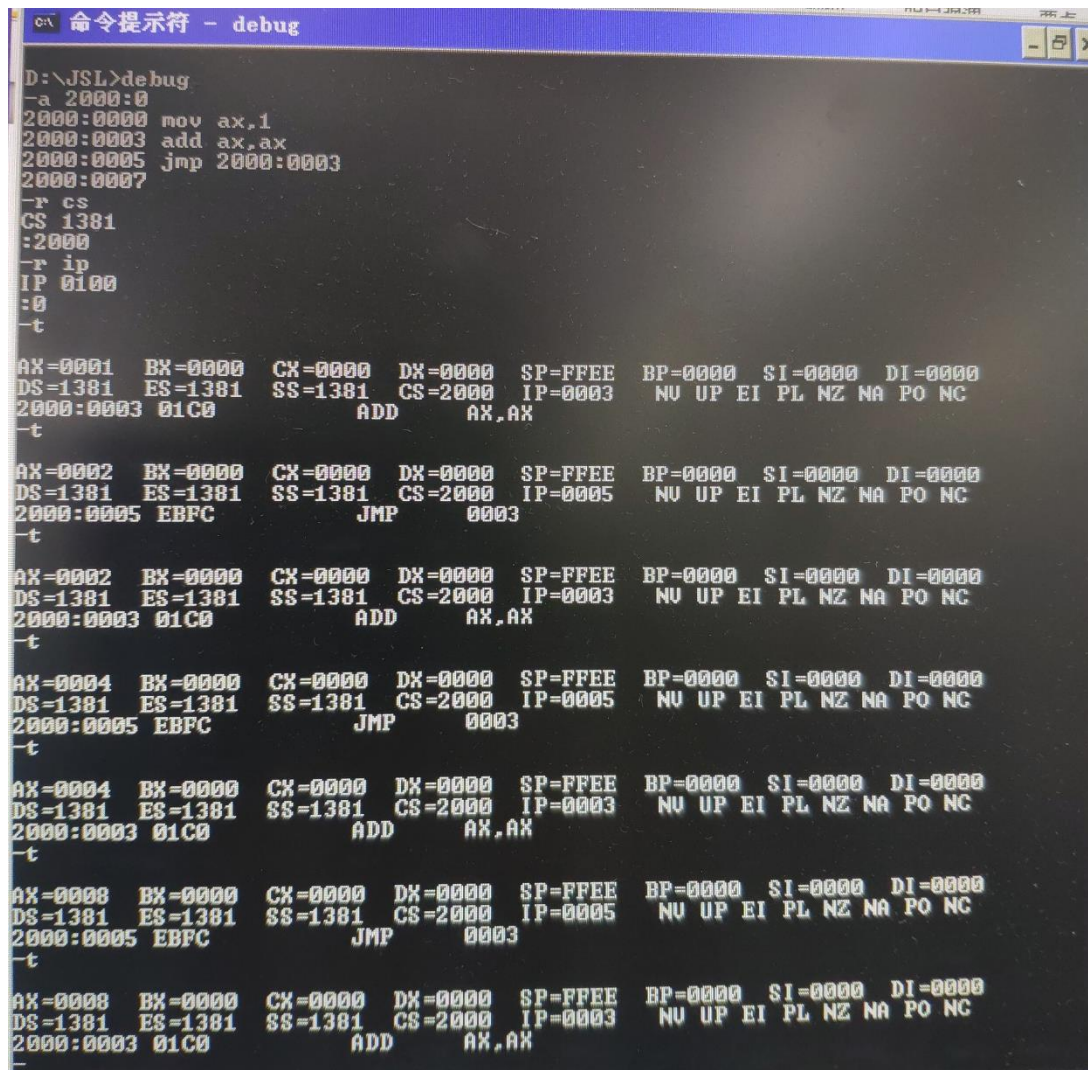
该程序指令含义详解：

- 将 4E20 写入 AX 中
- 将 1416 写入 AX 中
- 将 2000 写入 BX 中
- 将 AX+BX 的值写入 AX 中
- 将 AX 的值写入 BX 中
- 将 AX+BX 的值写入 AX 中
- 将 001A 写入 AX 中
- 将 0026 写入 BX 中
- 将 AL(AX 的后 2 个低位)+BL(BX 的后 2 个低位)的值相加写入 AL 中
- 将 BL 的值写入 AH(AX 的前 2 个高位)中
- 将 AL 的值写入 BH(BX 的前 2 个高位)中
- 将 0 写入 AH 中

- 将 BL 的值写入 AL 中
- 将 9C 和 AL 中的值相加，并写入 AL

2. 任务 2

按提示用 a 命令输入指令，得到以下运行结果：



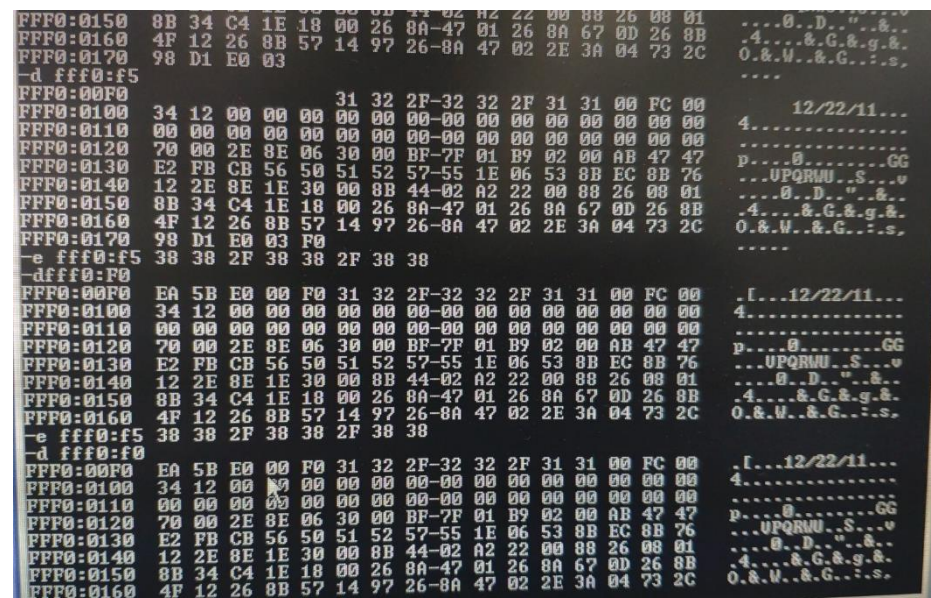
```
命令提示符 - debug
D:\JSL>debug
-a 2000:0
2000:0000 mov ax,1
2000:0003 add ax,ax
2000:0005 jmp 2000:0003
2000:0007
-r cs
CS 1381
:2000
-r ip
IP 0100
:0
-t
AX=0001 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=2000 IP=0003 NU UP EI PL NZ NA PO NC
2000:0003 01C0 ADD AX,AX
-t
AX=0002 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=2000 IP=0005 NU UP EI PL NZ NA PO NC
2000:0005 EBFC JMP 0003
-t
AX=0002 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=2000 IP=0003 NU UP EI PL NZ NA PO NC
2000:0003 01C0 ADD AX,AX
-t
AX=0004 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=2000 IP=0005 NU UP EI PL NZ NA PO NC
2000:0005 EBFC JMP 0003
-t
AX=0004 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=2000 IP=0003 NU UP EI PL NZ NA PO NC
2000:0003 01C0 ADD AX,AX
-t
AX=0008 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=2000 IP=0005 NU UP EI PL NZ NA PO NC
2000:0005 EBFC JMP 0003
-t
AX=0008 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1381 ES=1381 SS=1381 CS=2000 IP=0003 NU UP EI PL NZ NA PO NC
2000:0003 01C0 ADD AX,AX
```

当 t 输入 16 次时，即程序循环了 8 次时，在 AX 寄存器中得到了 2 的 8 次方值 256。

该程序的原理是：

首先将 AX 赋值为 1，
然后将 AX 的数值变为 2 倍，
最后执行跳转，重复上一步。

3. 任务 3



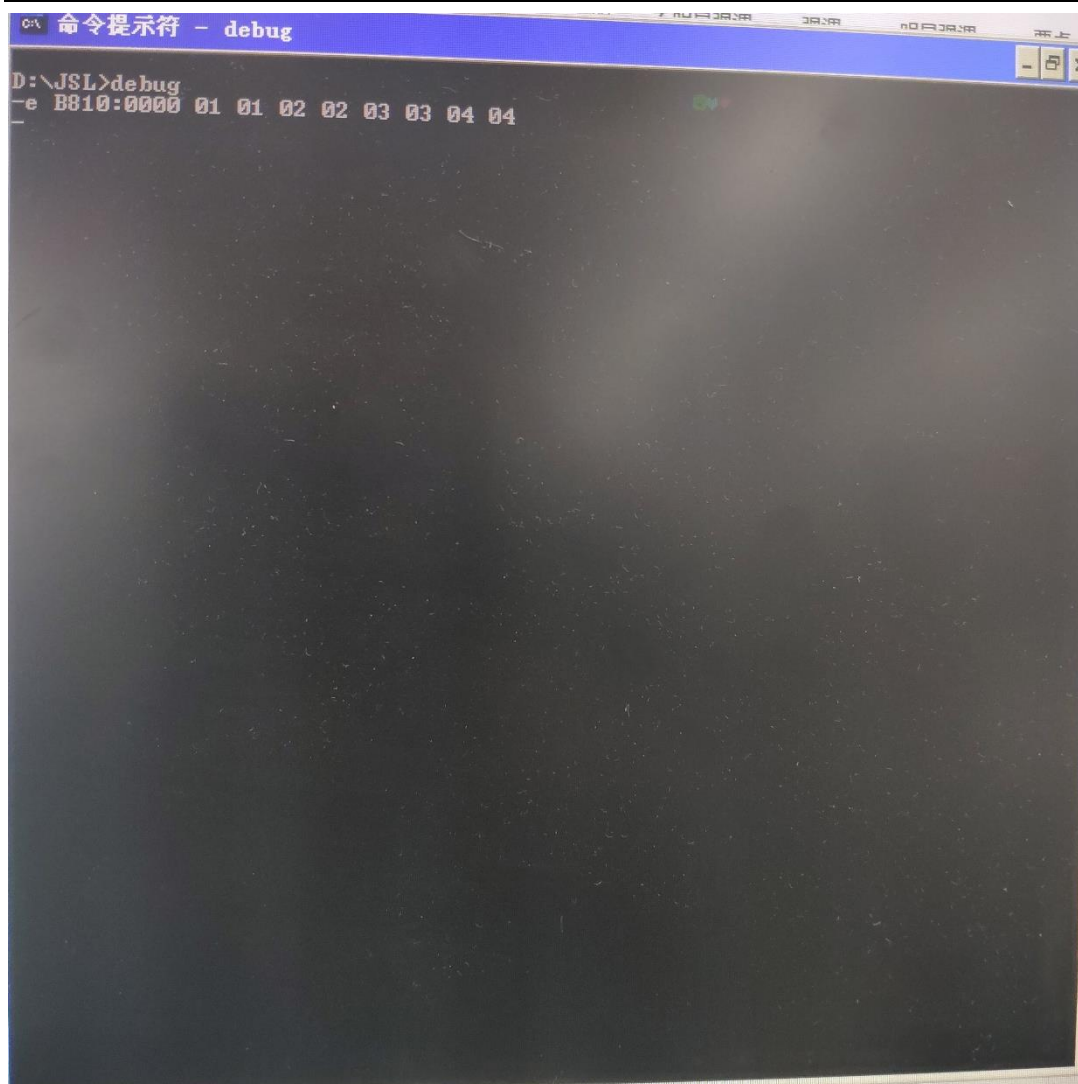
通过 d 指令我查找到了 PC 机主板 ROM 的生产日期存放在 FFF0:F5-FFF0:FC，是 2011 年 12 月 22 日生产的。

然后我试图通过 e 命令更改生产日期为 88/88/88，结果无法更改，其原因：ROM 是只读的，不能进行写入操作。

查阅教材，获知其原理：在 16 位系统中，C0000-FFFF 的 24KB 空间是各类 ROM 的地址空间，自然也就能够查询到主板 ROM 生产日期，并且不能进行写入操作了。

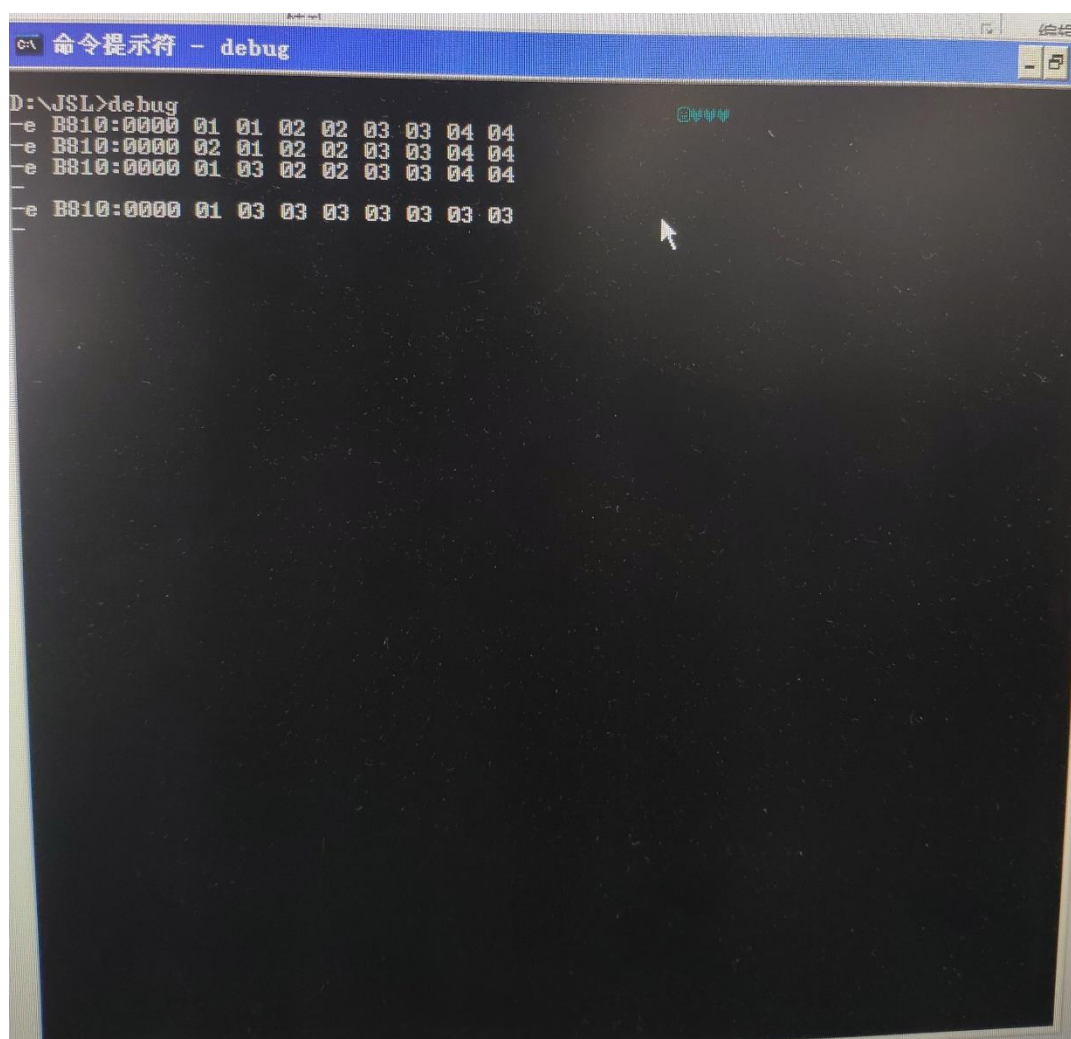
4. 任务 4

按照实验要求进行操作，得到如下结果：



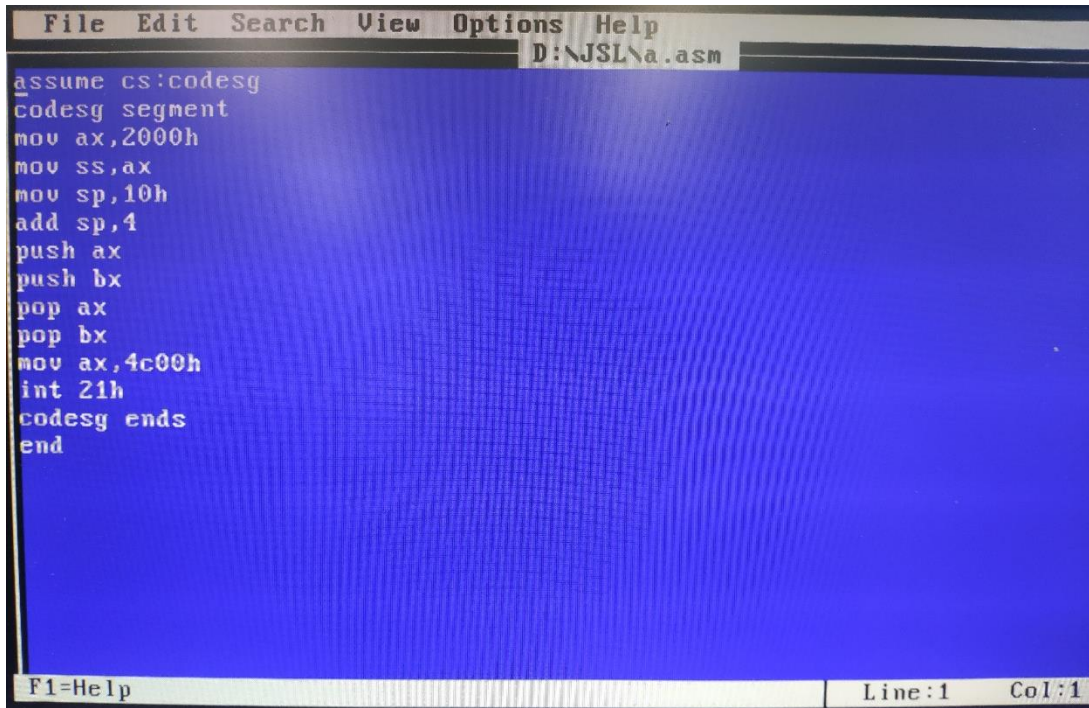
可以发现右上角出现了彩色字符。

继续改变数值，发现彩色字符的颜色和字符的内容都在变化：



```
命令提示符 - debug
D:\JSL>debug
-e B810:0000 01 01 02 02 03 03 04 04
-e B810:0000 02 01 02 02 03 03 04 04
-e B810:0000 01 03 02 02 03 03 04 04
-e B810:0000 01 03 03 03 03 03 03 03
```

改变地址值，发现字符的位置发生了移动：

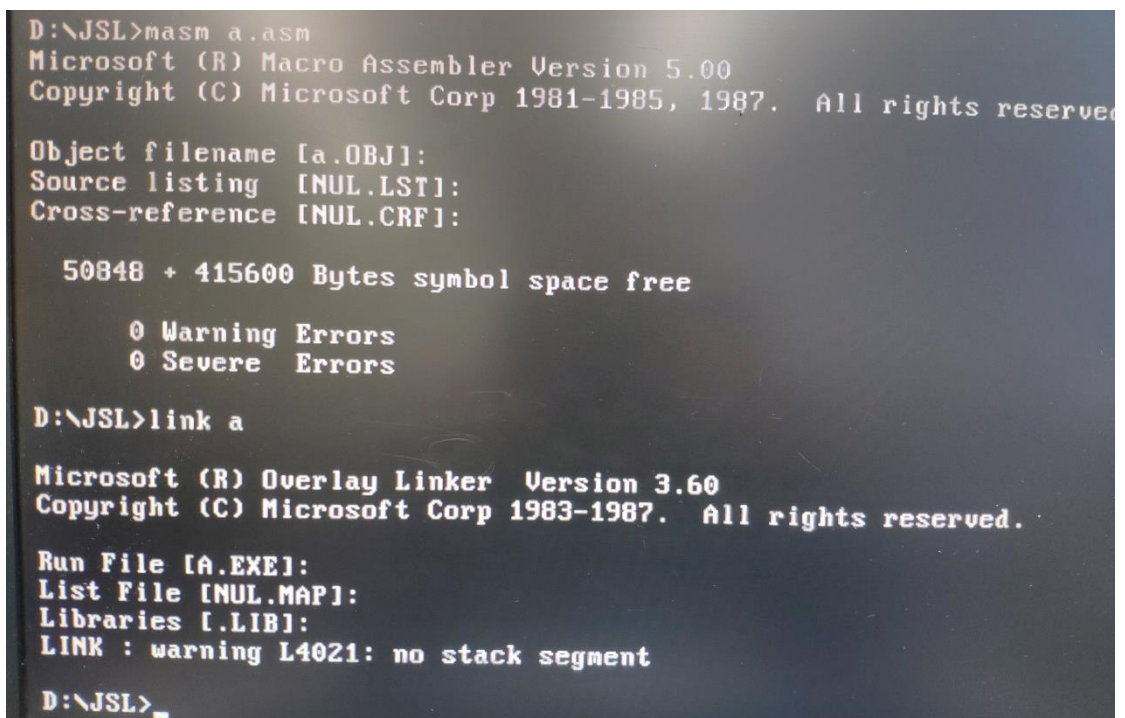


The screenshot shows a text editor window with a menu bar (File, Edit, Search, View, Options, Help) and a title bar (D:\JSL\ a.asm). The editor contains the following assembly code:

```
_assume cs:codesg
codesg segment
mov ax,2000h
mov ss,ax
mov sp,10h
add sp,4
push ax
push bx
pop ax
pop bx
mov ax,4c00h
int 21h
codesg ends
end
```

The status bar at the bottom indicates "F1=Help", "Line:1", and "Col:1".

然后使用 `masm a.asm` 命令和 `link a` 命令进行程序的编译和链接:



The screenshot shows a command prompt window with the following text:

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

Object filename [a.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:

50848 + 415600 Bytes symbol space free

0 Warning Errors
0 Severe Errors

D:\JSL>link a

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

Run File [A.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

D:\JSL>_
```

在以上操作步骤结束后, 使用 `dir` 命令, 可以看到文件夹下生成了 `a.asm`, `a.obj`, `a.exe` 文件:

```
D:\JSL>dir
驱动器 D 中的卷没有标签。
卷的序列号是 74E5-5714

D:\JSL 的目录

2019-10-20  08:52    <DIR>
2019-10-20  08:52    <DIR>
2019-10-20  08:50                156 a.asm
2019-10-20  08:52                532 A.EXE
2019-10-20  08:52                 71 A.OBJ
2003-03-27  12:00            20,634 DEBUG.EXE
2003-03-27  12:00            69,886 EDIT.COM
1999-09-08  15:00            64,982 LINK.EXE
1999-09-08  15:00           103,175 MASM.EXE
              7 个文件                259,436 字节
              2 个目录 209,351,139,328 可用字节

D:\JSL>
```

随后，使用 debug a.exe 进行程序的调试。

首先使用 u 命令查看当前程序的机器码：

```
D:\JSL>debug a.exe
-u
13DA:0000 B80020      MOV     AX,2000
13DA:0003 8ED0      MOV     SS,AX
13DA:0005 BC1000     MOV     SP,0010
13DA:0008 83C404     ADD     SP,+04
13DA:000B 50      PUSH    AX
13DA:000C 53      PUSH    BX
13DA:000D 58      POP     AX
13DA:000E 5B      POP     BX
13DA:000F B8004C     MOV     AX,4C00
13DA:0012 CD21      INT     21
13DA:0014 02B8FFFF   ADD     BH,[BX+SI+FFFF]
13DA:0018 50      PUSH    AX
13DA:0019 B80500     MOV     AX,0005
13DA:001C 50      PUSH    AX
13DA:001D 8D867AFE   LEA     AX,[BP+FE7A]
```

然后使用 t 命令进行程序的调试，当程序运行到 MOV AX, 4C00 时使用 p 命令继续进行调试：


```

AX=2000 BX=0000 CX=0014 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=13CA ES=13CA SS=13DA CS=13DA IP=0003  NU UP EI PL NZ NA PO NC
13DA:0003 8ED0          MOV     SS,AX
-t

AX=2000 BX=0000 CX=0014 DX=0000 SP=0010 BP=0000 SI=0000 DI=0000
DS=13CA ES=13CA SS=2000 CS=13DA IP=0008  NU UP EI PL NZ NA PO NC
13DA:0008 83C404        ADD     SP,+04
-t

AX=2000 BX=0000 CX=0014 DX=0000 SP=0014 BP=0000 SI=0000 DI=0000
DS=13CA ES=13CA SS=2000 CS=13DA IP=000B  NU UP EI PL NZ NA PE NC
13DA:000B 50           PUSH    AX
-t

AX=2000 BX=0000 CX=0014 DX=0000 SP=0012 BP=0000 SI=0000 DI=0000
DS=13CA ES=13CA SS=2000 CS=13DA IP=000C  NU UP EI PL NZ NA PE NC
13DA:000C 53           PUSH    BX
-t

AX=2000 BX=0000 CX=0014 DX=0000 SP=0010 BP=0000 SI=0000 DI=0000
DS=13CA ES=13CA SS=2000 CS=13DA IP=000D  NU UP EI PL NZ NA PE NC
13DA:000D 58           POP      AX
-

AX=2000 BX=0000 CX=0014 DX=0000 SP=0010 BP=0000 SI=0000 DI=0000
DS=13CA ES=13CA SS=2000 CS=13DA IP=000D  NU UP EI PL NZ NA PE NC
13DA:000D 58           POP      AX
-t

AX=0000 BX=0000 CX=0014 DX=0000 SP=0012 BP=0000 SI=0000 DI=0000
DS=13CA ES=13CA SS=2000 CS=13DA IP=000E  NU UP EI PL NZ NA PE NC
13DA:000E 5B           POP      BX
-t

AX=0000 BX=2000 CX=0014 DX=0000 SP=0014 BP=0000 SI=0000 DI=0000
DS=13CA ES=13CA SS=2000 CS=13DA IP=000F  NU UP EI PL NZ NA PE NC
13DA:000F B8004C        MOV     AX,4C00
-t

AX=4C00 BX=2000 CX=0014 DX=0000 SP=0014 BP=0000 SI=0000 DI=0000
DS=13CA ES=13CA SS=2000 CS=13DA IP=0012  NU UP EI PL NZ NA PE NC
13DA:0012 CD21          INT     21
-p

Program terminated normally
-

```

从图中我们可以清晰地看到 AX, BX, IP 寄存器的变化。

该程序分步执行含义详解：

- 将 AX 赋值为 2000H, 并且 IP+3
- 将 AX 赋值给 SS, 并且同时把 SP 赋值为 10H, 使得栈顶指针 SS:SP 指向 2000:10, 并且 IP+5
- 将 SP 的值加 4, 为下步中的入栈做准备, IP+3
- 将 AX 入栈, IP+1
- 将 BX 入栈, IP+1
- 栈顶弹出一个元素, 并将值赋给 AX, 此时 AX 为原先 BX 的值: 0000H, IP+1
- 栈顶弹出一个元素, 并将值赋给 BX, 此时 BX 为原先 AX 的值: 2000H, IP+1
- DOS 调用, 程序结束

程序运行结束后, 使用 d CS:0, 我们可以看到内存中仍然保存着程序指令的机器码:

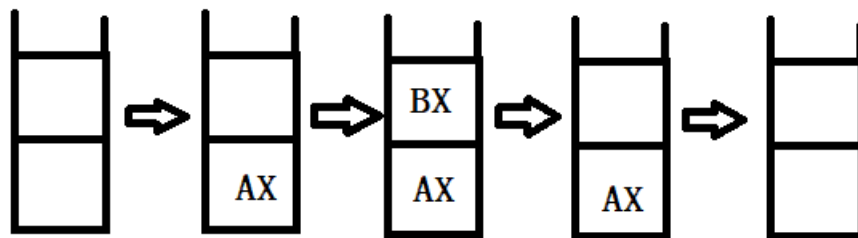
```

-d CS:0
13DA:0000 B8 00 20 8E D0 BC 10 00-83 C4 04 50 53 58 5B B8 .. .PSXI.
13DA:0010 00 4C CD 21 CC E8 CA 00-0B C0 74 08 FF 76 FE E8 .L.!.....t..v..
13DA:0020 D4 A1 EB 20 C7 44 02 01-00 8B 46 04 89 44 04 8B ... .D....F..D..
13DA:0030 C6 2D 2E 4B B9 0E 00 99-F7 F9 50 8B F8 E8 72 00 ... .K.....P...r.
13DA:0040 8B C7 EB 09 57 E8 AE A1-B8 FF FF 89 04 5E 5F C9 ....U.....^..
13DA:0050 C2 02 00 00 55 8B EC 57-56 8B 7E 04 39 3E 7C 01 ....U..UV..9>l.
13DA:0060 75 06 C7 06 7E 01 FF FF-39 3E 80 01 75 05 6A 00 u...~...9>..u.j.
13DA:0070 E8 0F 12 6B DF 0E 8B B7-2E 4B C1 E6 02 03 36 B4 ...k.....K....6.

```

该程序主要的功能是实现了 AX 和 BX 之间值的互换。

栈的变化：



四、实验总结

通过这两次实验课，我已经能够熟练使用 debug 的 r, d, e, u, a, t 命令进行程序的调试和内存地址值的修改。同时，我还了解到了不同内存地址对应的硬件设备和显卡的工作原理，并且能够用 masm 编写程序并进行连接操作，收获颇丰。