**信息科学与工程学院**

**2019－2020学年第二学期**

实 验 报 告

课程名称： 微处理器原理与应用

实验名称： 实验1.2

专 业 班 级 通信工程 二班

学 生 学 号 201800121050

学 生 姓 名 孟麟芝

实 验 时 间 2020年2月15日

实验报告

【实验目的】

1. 掌握DEBUG的基本命令及其功能，掌握win10使用DEBUG功能

2. 了解使用机器指令编程，掌握汇编指令编程

【实验要求】

1. 掌握一些基本汇编指令的运用
2. 掌握DEBUG的R,D,E,U,T,A等指令的运用

【实验具体内容】

1. 预备知识：

（1）本次实验所需汇编指令

mov A,B 即将B的值赋给A

add ax,bx 即将ax，bx中的值相加的结果赋给ax

sub ax,bx 即将bx减去ax后的结果赋给ax

jmp 无条件跳转，用以改变CS，IP

（2）何为DEBUG

Debug是DOS和Windows都提供的实模式（8086方式）程序的调试工具。使用它可以查看CPU各种寄存器中的内存、内存情况和在机器码级别跟踪程序的运行

（3）DEBUG中的几个指令

-r 查看、改变CPU寄存器的内容

-d 查看内存中的内容

-e 改写内存中的内容

-u 将内存中的机器指令翻译成汇编指令

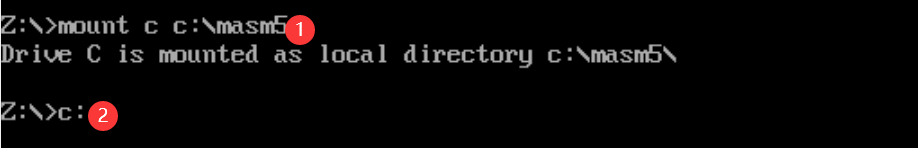
-t 执行一条机器指令

-a 以汇编形式在内存中写入一条机器指令

2.

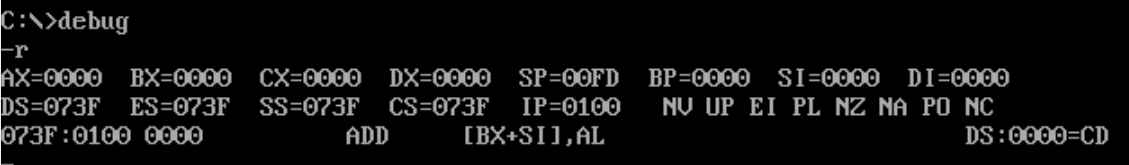
【调试DEBUG】

（1）在winxp环境下的CMD提供的DOS环境可直接进入DEBUG，在win10下需要由DOSBox实现



* 1. 将本地的c:\masm5挂载为虚拟环境中的C盘（可挂载多个盘，只要分别命名即可）
  2. 在虚拟环境中选择C盘符

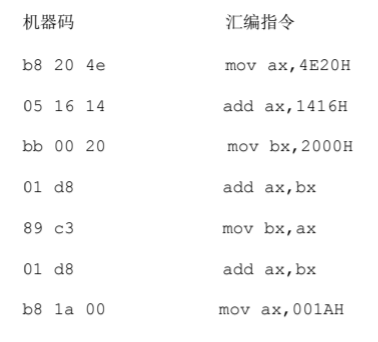
（2）打开debug程序

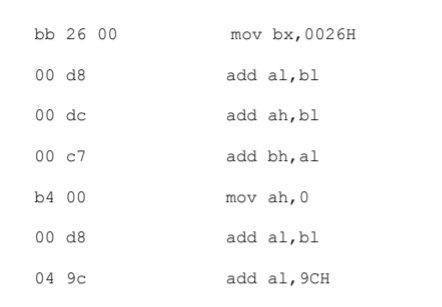


可见DEBUG已可正常使用

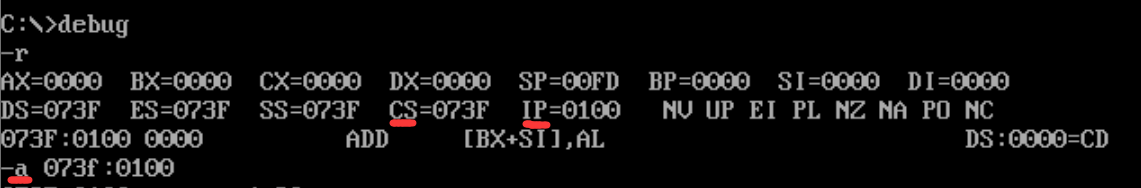
【第一个实验】

实验要求：使用DEBUG将下面的程序写入内存，逐条执行，观察每条指令执行后CPU中相关寄存器的内容变化

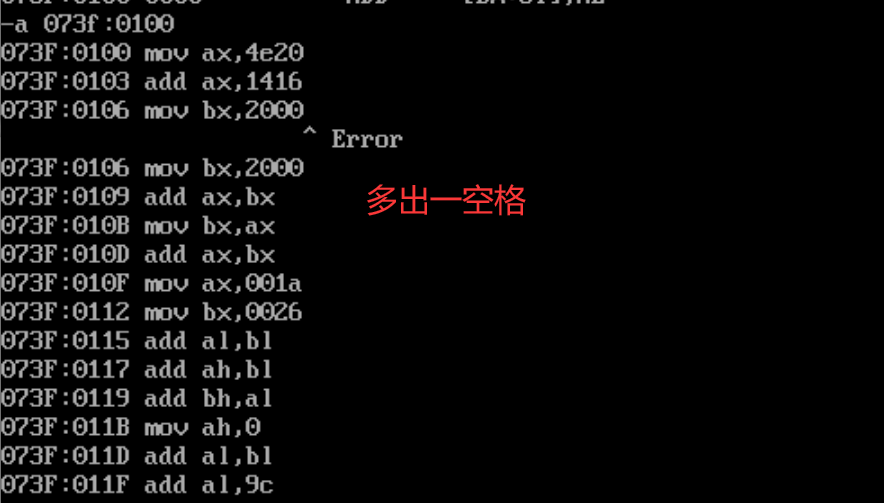




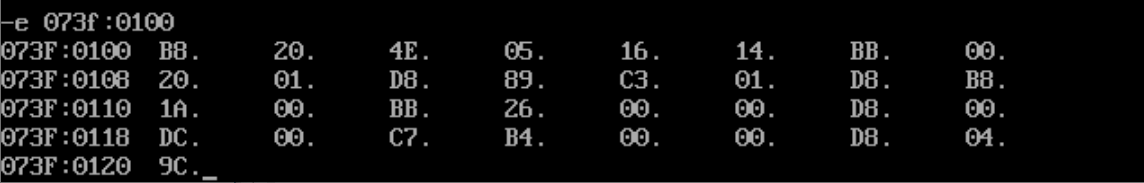
（1）使用-r指令查看当前CS:IP的位置，不妨在该位置开始写入指令



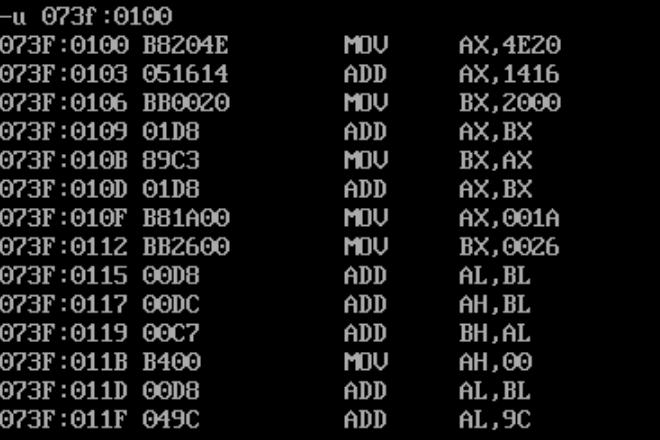
（2）使用-a指令写入内存



再尝试使用e命令写入（已为正确结果，直接按空格跳过更改）



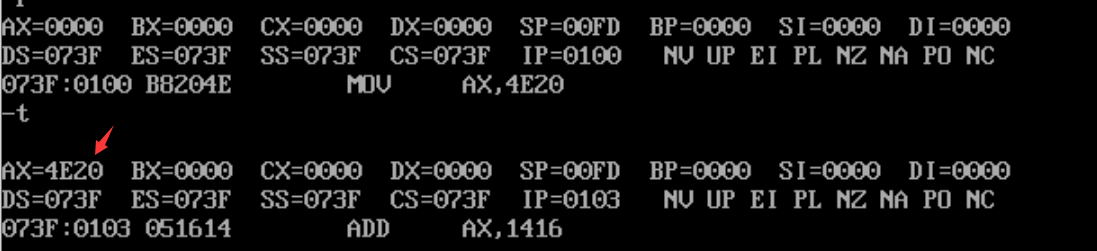
（3）使用-u命令查看输入的指令



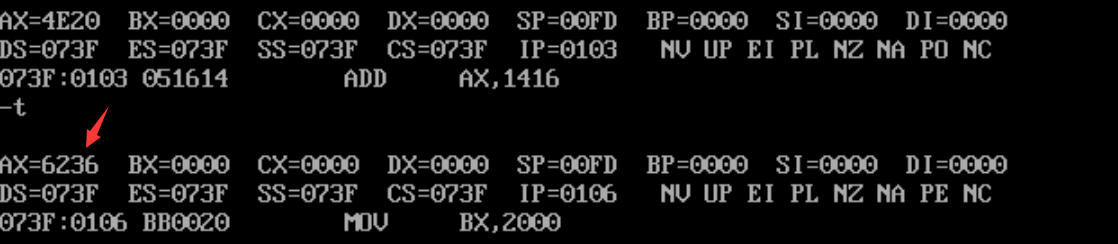
注意，如果不指定Max IP，则最多翻译20H个字节的指令

（4）使用-t指令逐条执行

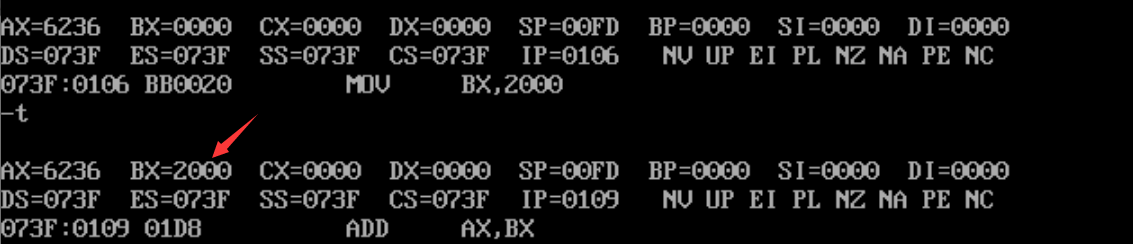
Mov ax,4e20



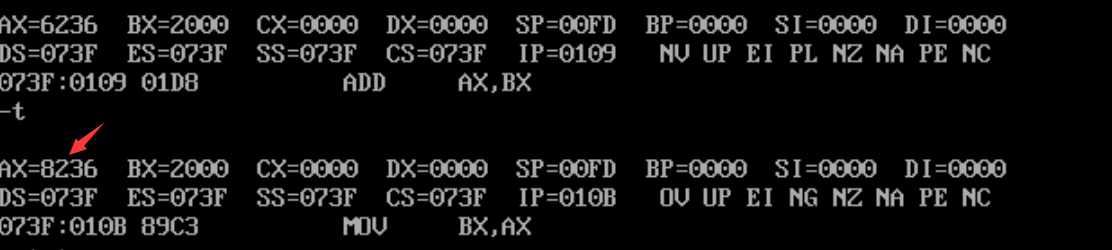
Add ax,1416



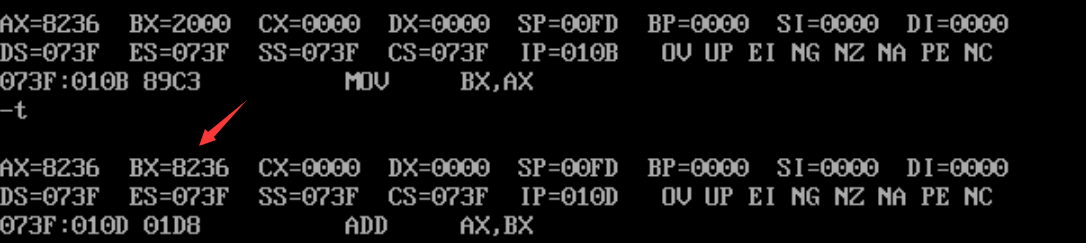
Mov bx,2000



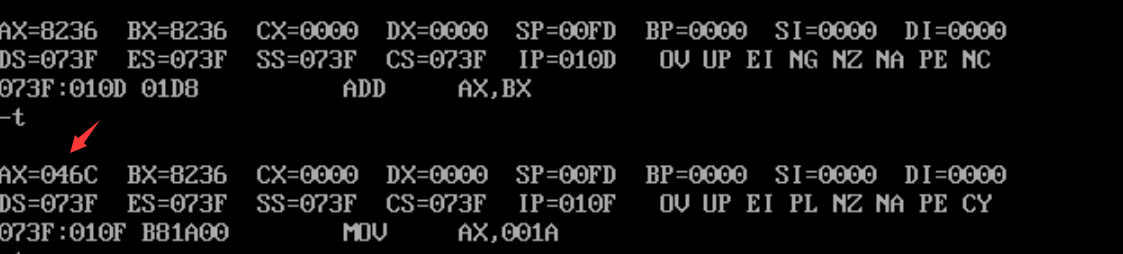
Add ax,bx



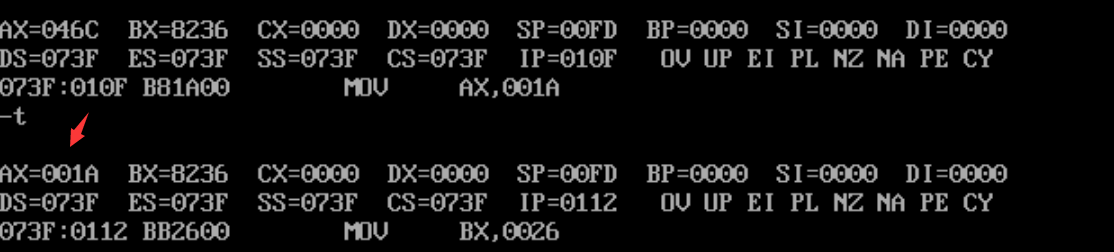
Mov bx,ax



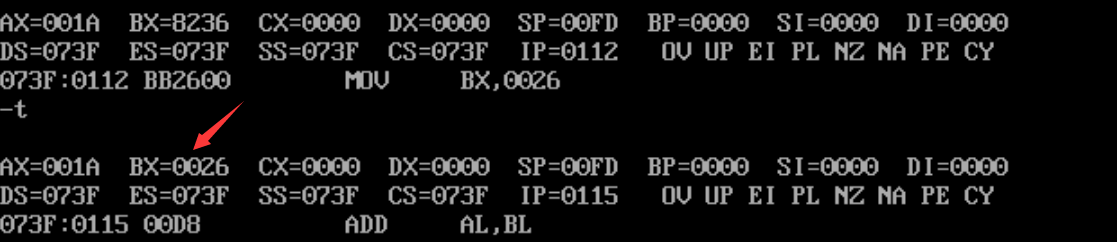
Add ax,bx



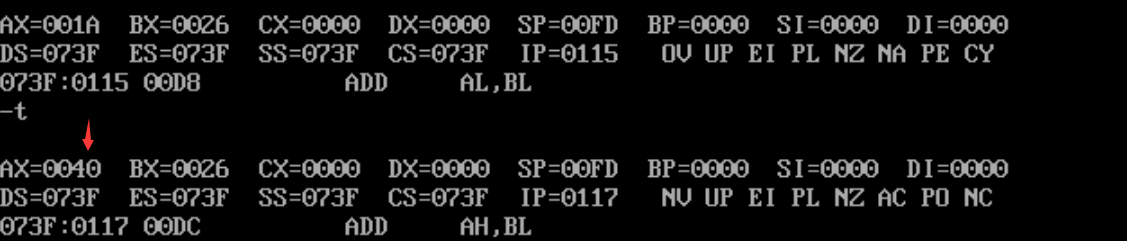
Mov ax,001a



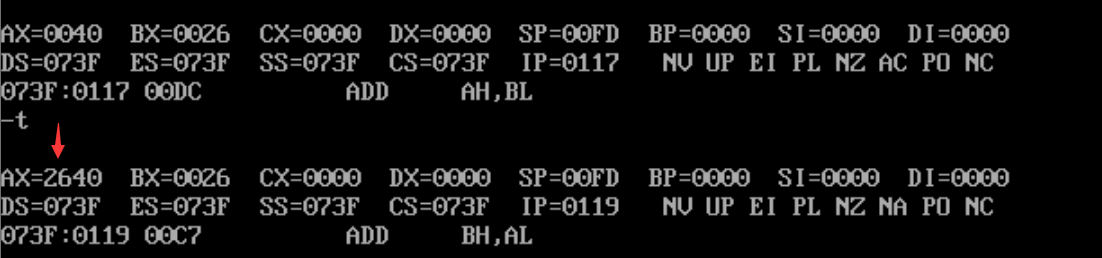
Mov bx,0026



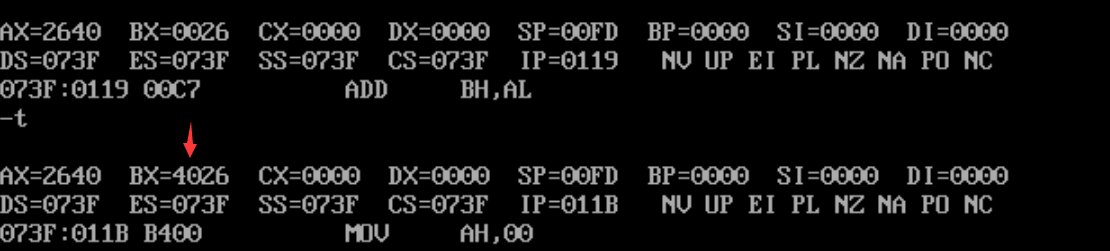
Add al,bl



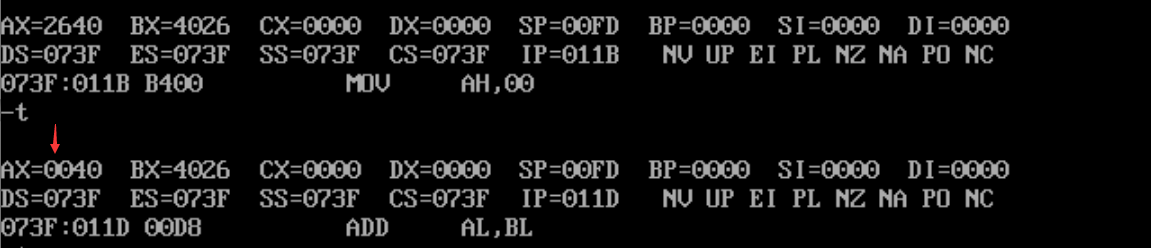
Add ah,bl



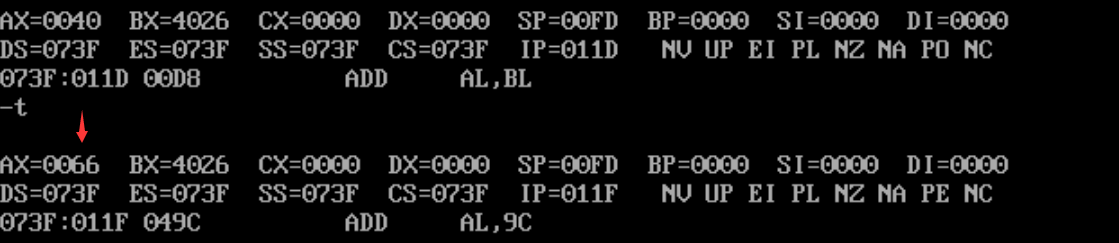
Add bh,al



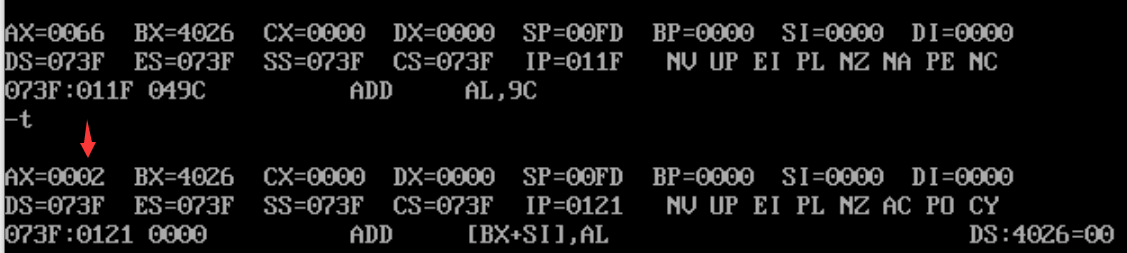
Mov ah,0



Add al,bl



Add al,9c



【第二个实验】

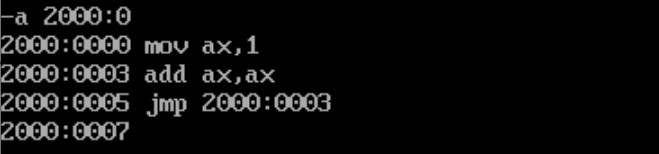
实验要求：将下面 3 条指令写入从 2000:0 开始的内存单元中，利用这 3 条指令计算 2的8次方。

mov ax,1 （从2000:0开始的内存单元）

add ax,ax

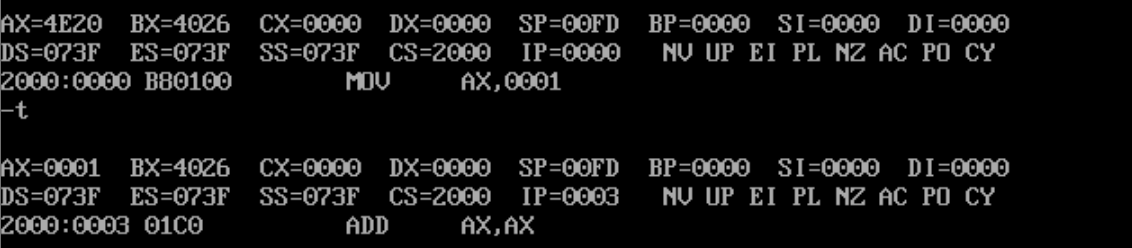
jmp 2000:0003 （观察跳到什么地方了？）

（1）将三条指令写入从2000：0开始的内存单元中

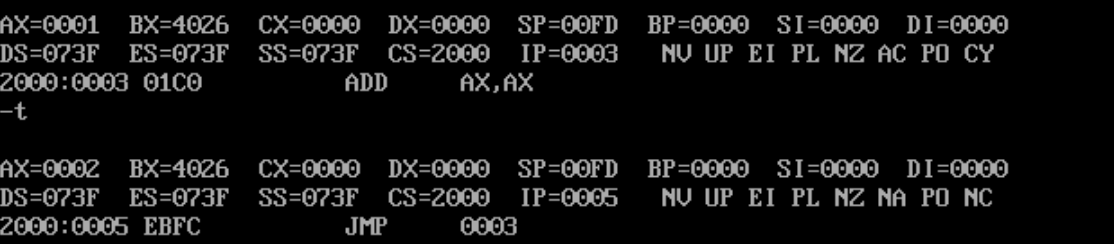


（2）逐条执行，直到ax中计算得出2的8次方

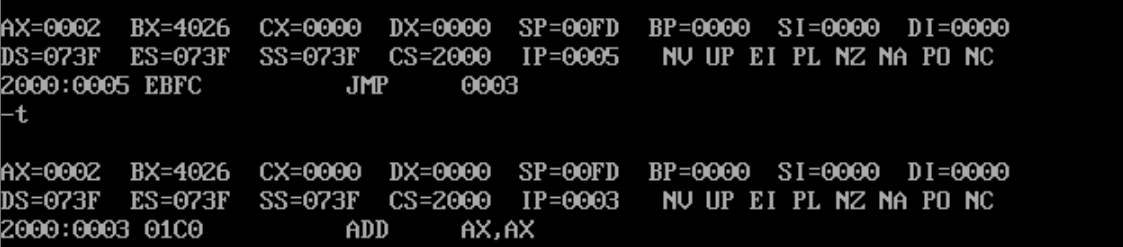
Mov ax,0001



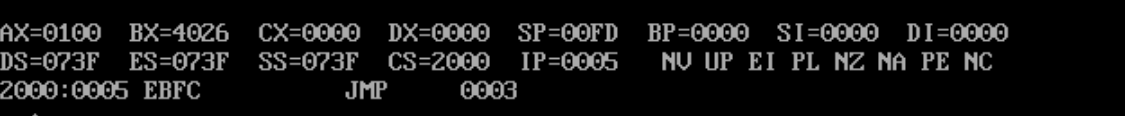
Add ax,ax



Jmp 0003

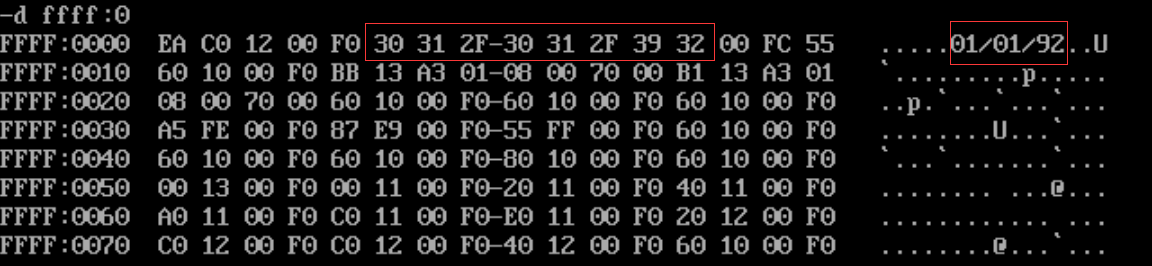


到此可见，程序发生循环，又跳回了ax+ax的步骤，实现循环乘2，经过循环，得到ax中为0100H,即得到所需结果



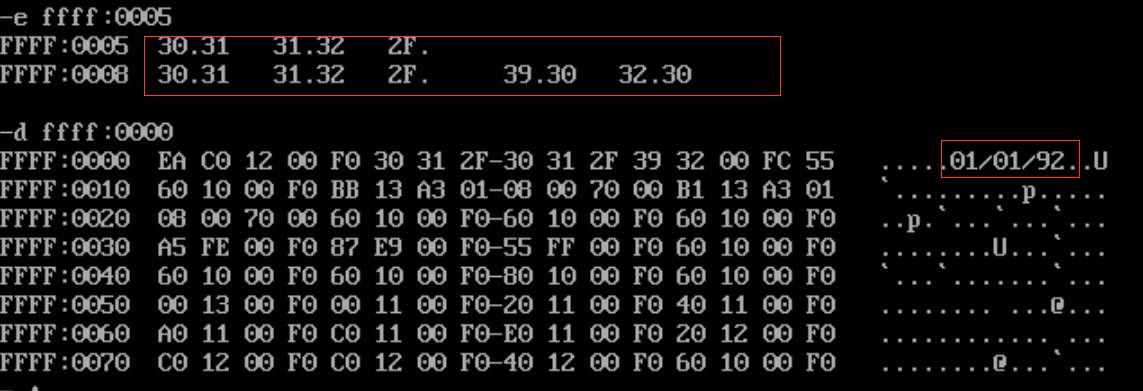
【第三个实验】

查看内存中的内容PC机主板上的ROM中写有一个生产日期，在内存FFF00H~FFFFFH的某几个单元中，请找到这个生产日期并试图改变它。（内存ffff:0005~ffff:000C(共8个字节单元中)处）



可见在该DOSBOX环境下的主板生产日期为92年1月1日，要注意，这是虚拟环境中虚拟的生产日期，并不是真实的日期

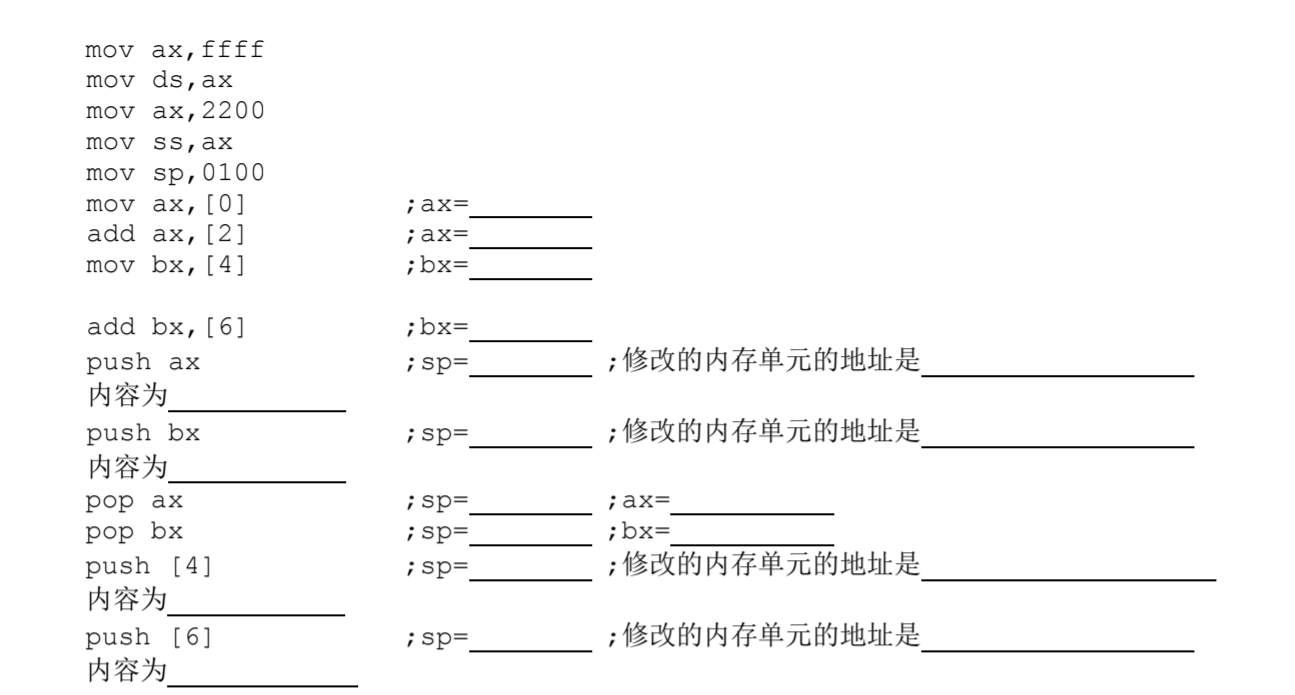
尝试修改为00年12月12日



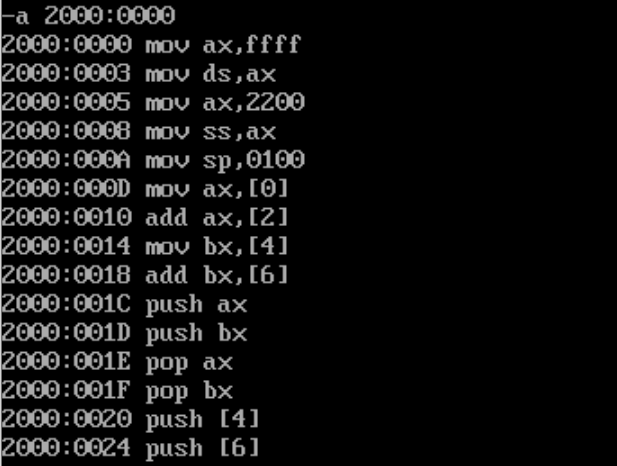
可见修改并没有起到作用，问题在于，向地址C0000~FFFFF的内存单元写入数据的操作是无效的，因为这等于改写只读存储器中的内容

【第四个实验】

使用Debug，将下面的程序段写入内存，逐条执行，根据指令执行后的实际运行情况填空。（逐条执行，每条指令执行结果截图）

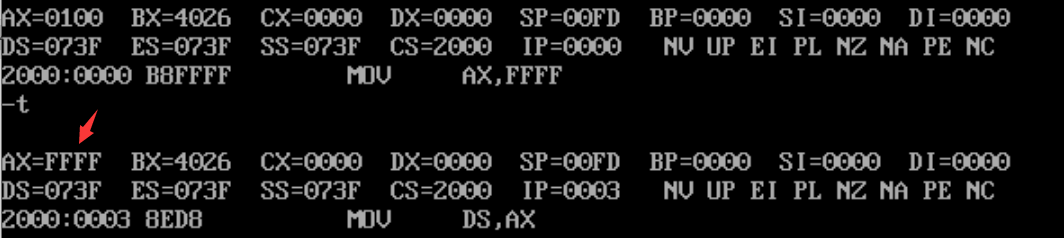


（1）将指令写入内存

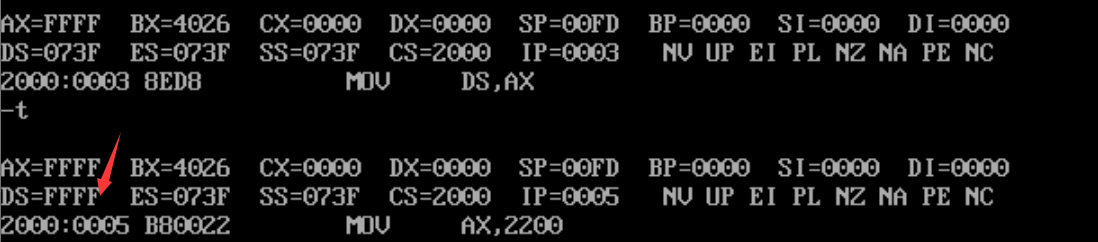


（2）逐条执行指令

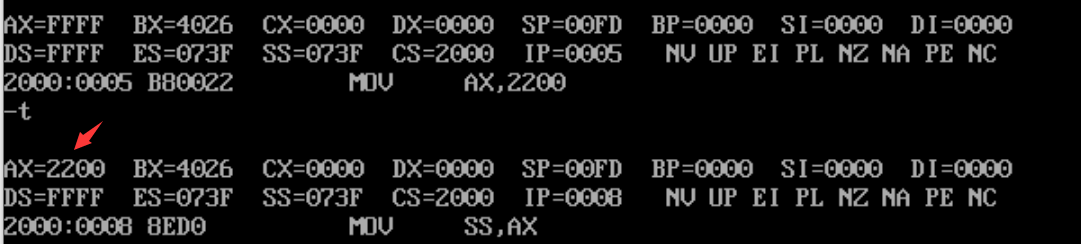
MOV AX,FFFF



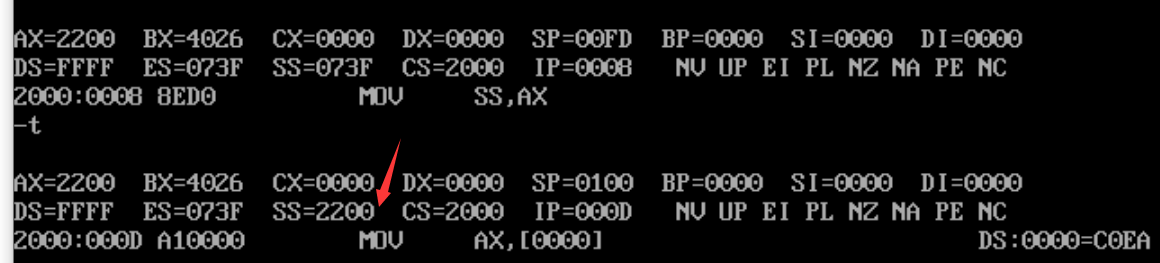
MOV DS,AX



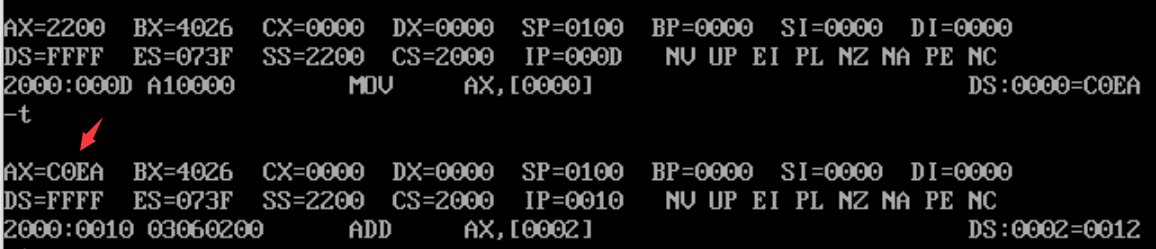
MOV AX,2200



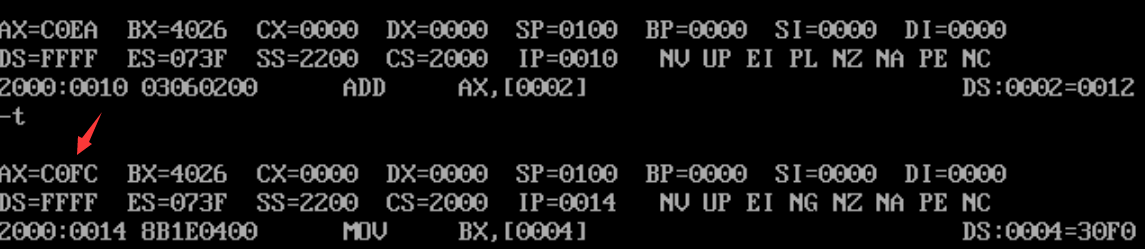
MOV SS,AX



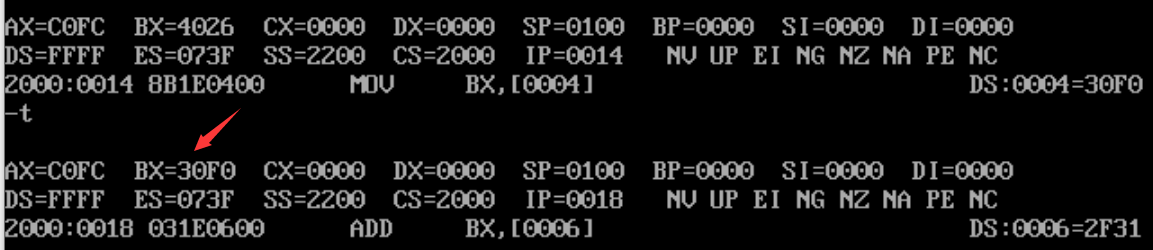
MOV AX,[0]



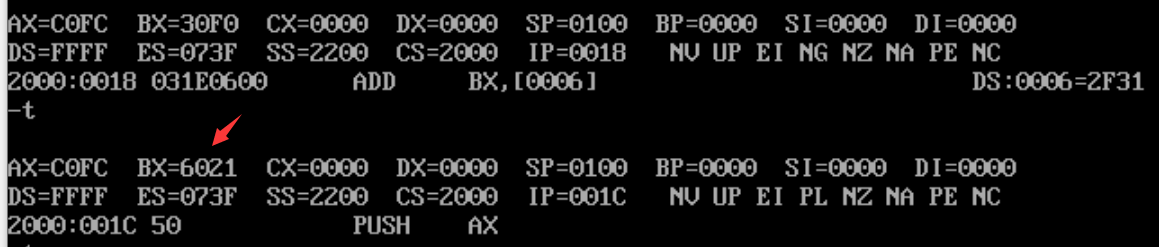
ADD AX,[2]



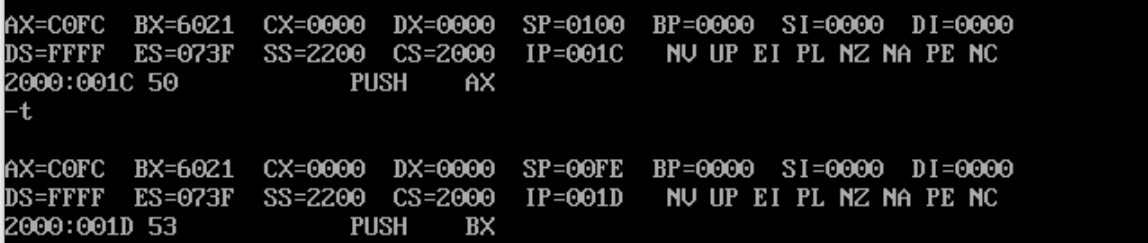
MOV BX,[4]



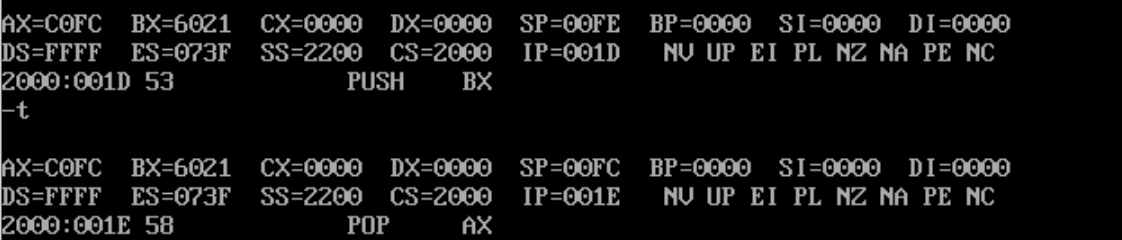
ADD BX,[6]



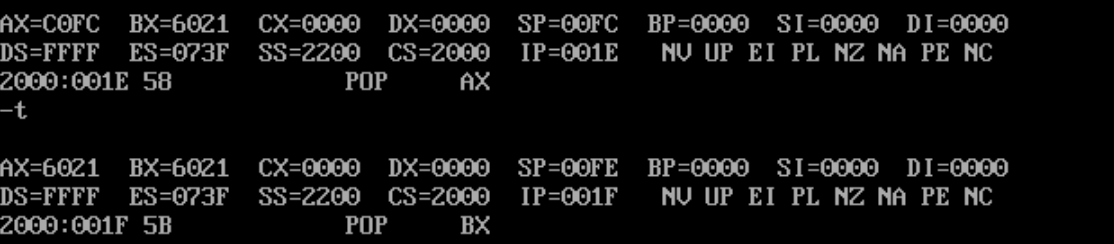
PUSH AX



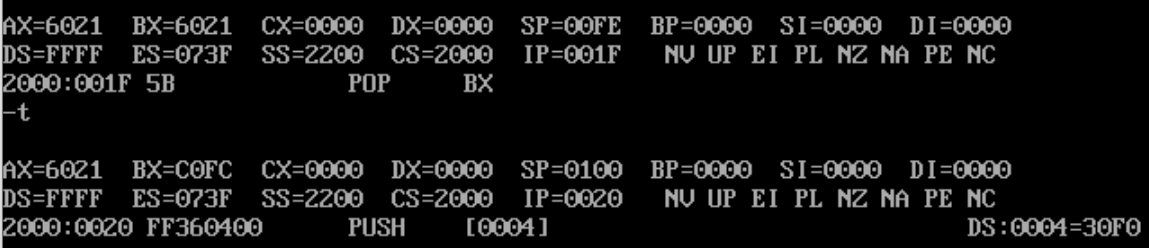
PUSH BX



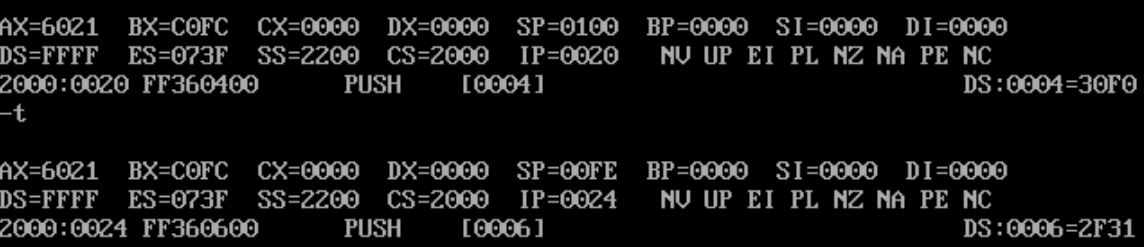
POP AX



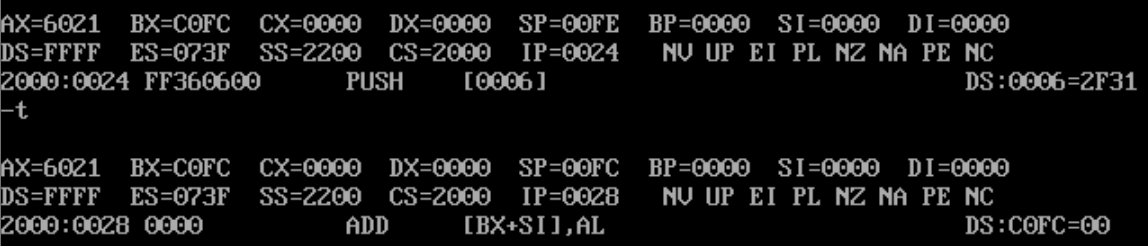
POP BX

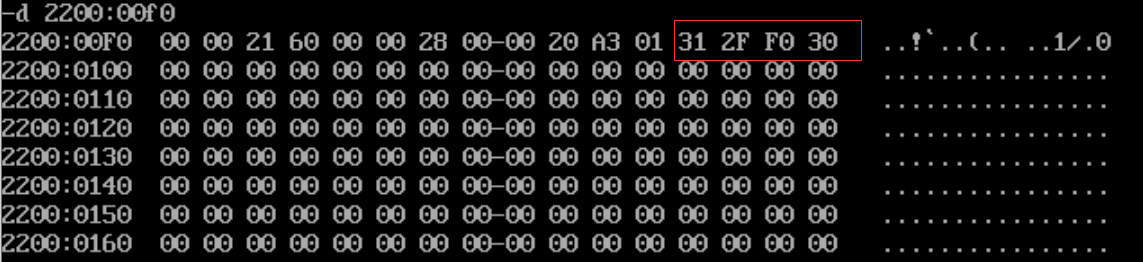


PUSH [4]



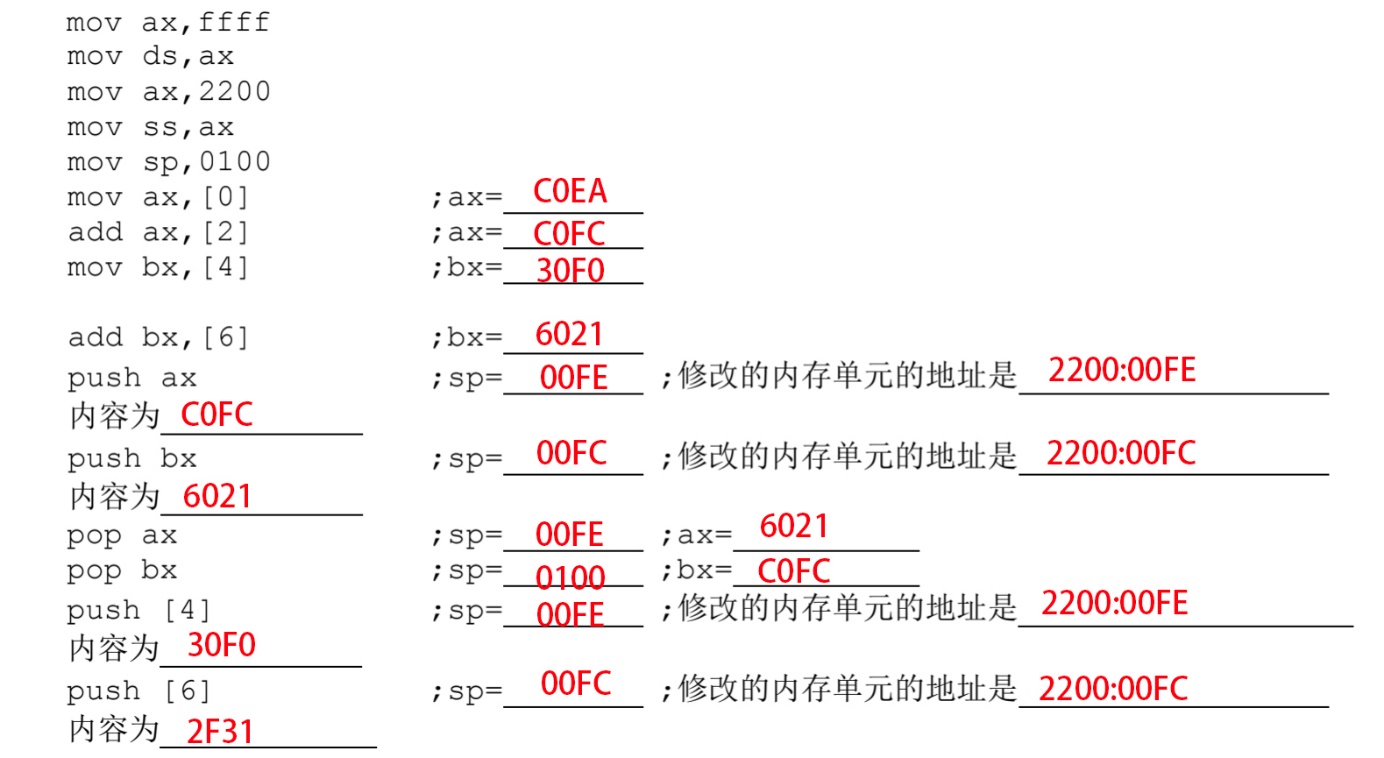
PUSH [6]





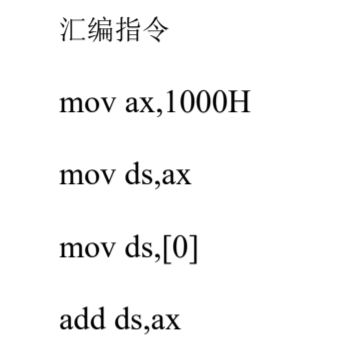
结束最后一步操作后，可见栈内存放着最后放入的两个数据，DS:0006和DS:0004的数据：2F31和30F0

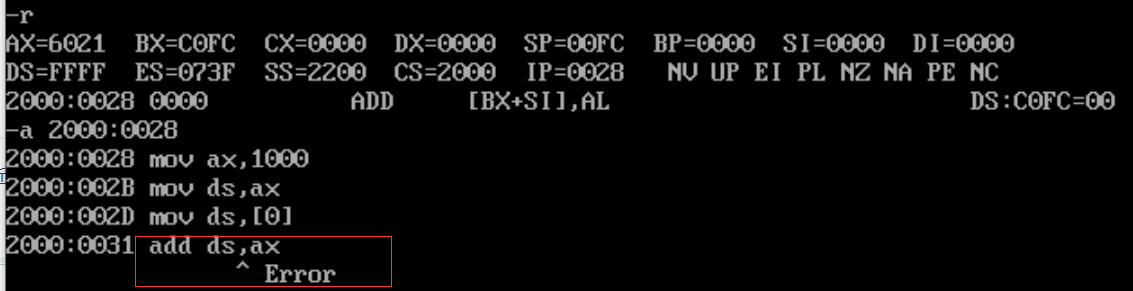
（3）根据实际运行情况填空



【第五个实验】

使用Debug，将下面的程序段写入内存，逐条执行，观察每条指令执行后，CPU中相关寄存器中内容的变化。（逐条执行，每条指令执行结果截图）如果有问题请说明原因



（1）将指令写入内存后逐条执行

要注意，段寄存器不可以参与算术运算

（2）将add ds，ax指令替换为如下三条指令后逐条执行

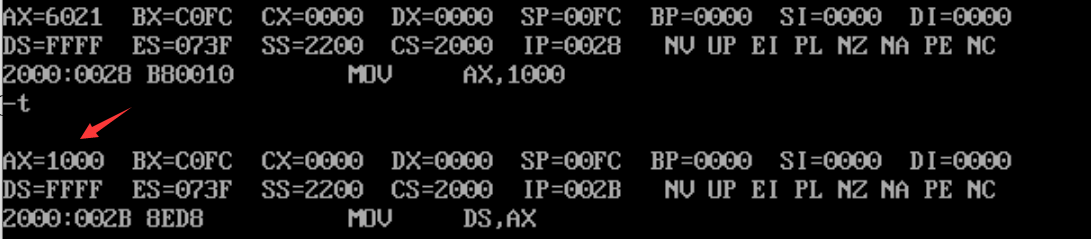
mov bx,ds

add ax,bx

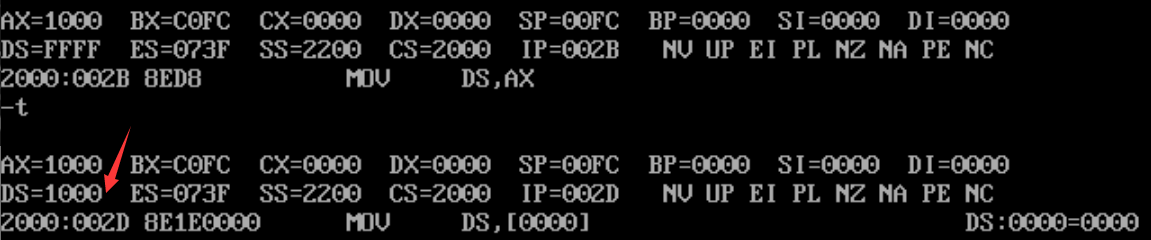
mov ds,ax

（3）逐条执行

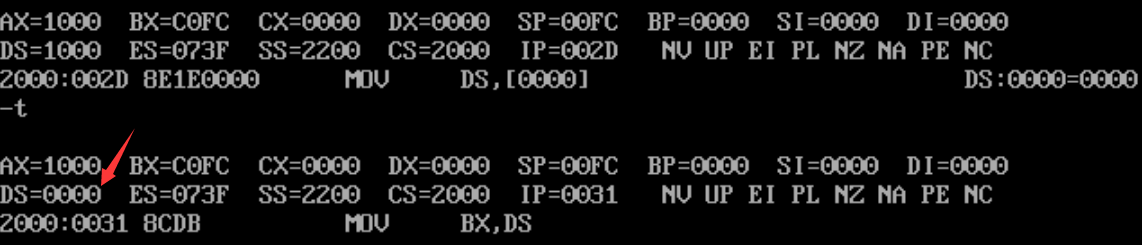
MOV AX,1000



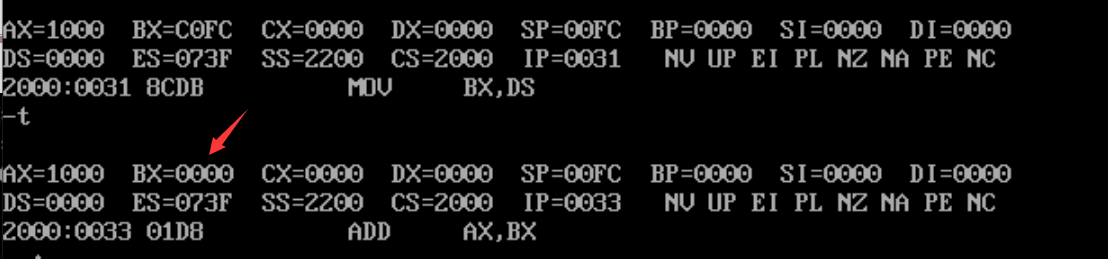
MOV DS,AX



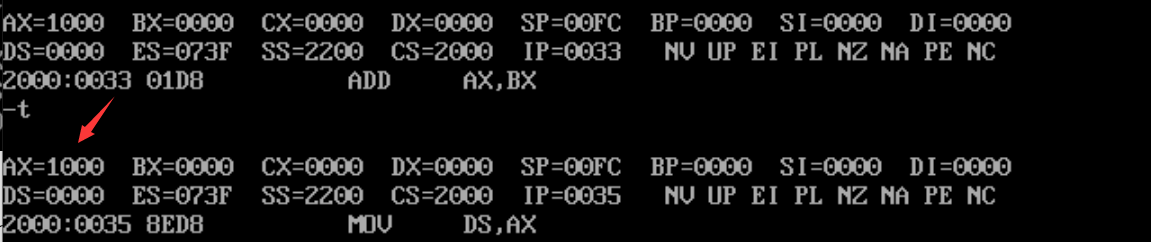
MOV DS,[0]



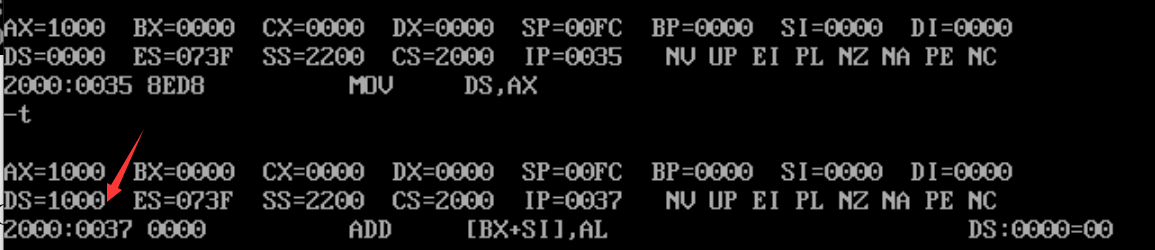
MOV BX,DS



ADD AX,BX



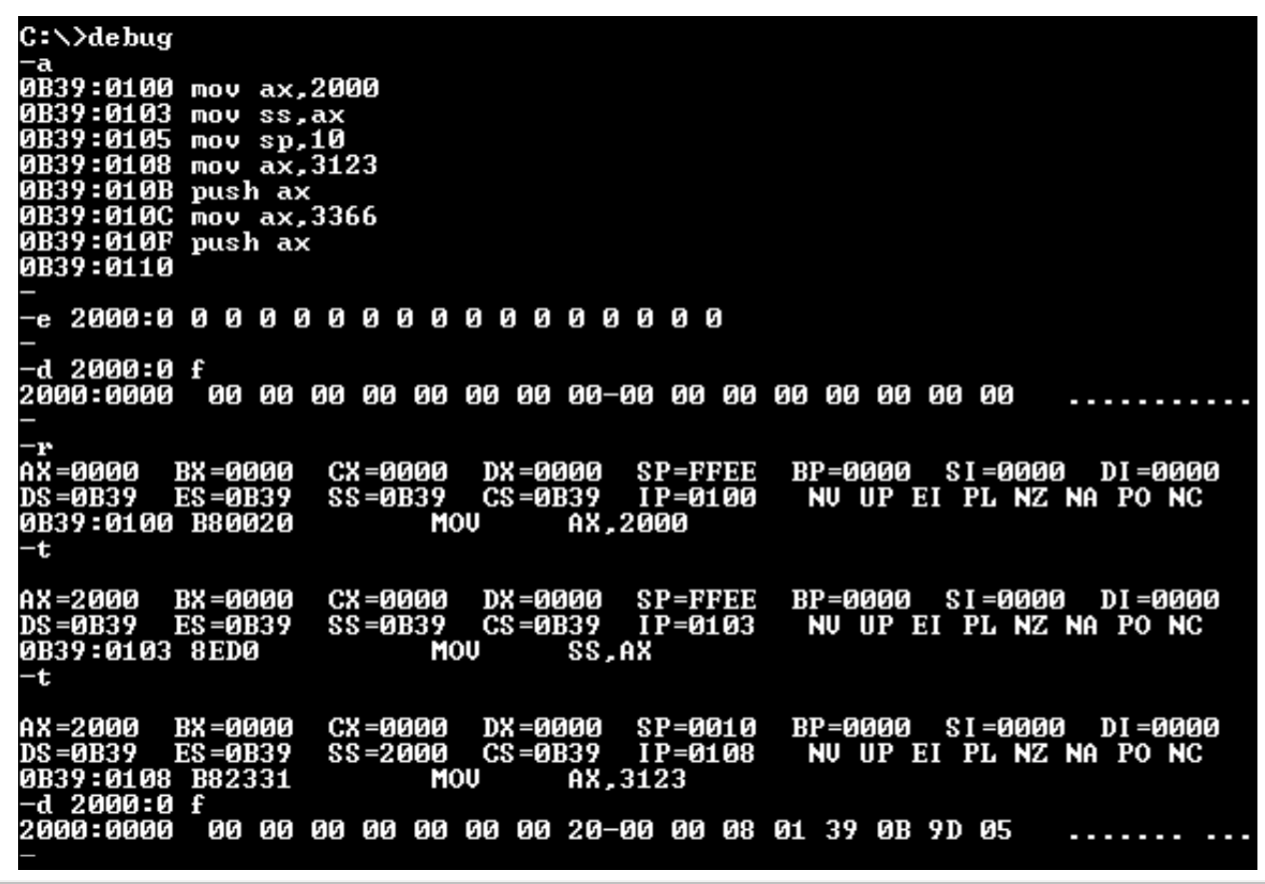
MOV DS,AX



实现了原功能

【第六个实验】

仔细观察下图中的实验过程，然后分析：为什么2000:0~2000:f中的内容会发生改变？



在执行mov ss,ax时，它的下一条指令mov sp,10也紧跟着被执行。另外，-t指令将会引发单步中断，CPU执行1号中断程序，将标志寄存器入栈，TF、IF设置为0，CS、IP等段寄存器入栈，再对CS、IP做特定修改。可见图中右边6个字节从右至左分别为标志寄存器的值，CS，IP

【实验心得】

1. 将本地盘挂载为虚拟盘时可挂载多个，类似日常使用电脑的硬盘分区
2. 写入指令时要先注意CS:IP的位置，再进行写入
3. 从地址0~9FFFF的内存单元中读取数据，实际上就是在读取主随机存储器中的数据，向地址A0000~BFFFF的内存单元中写数据，就是向显存中写数据，这些数据会被显示卡输出到显示器上。向地址C0000~FFFFF的内存单元写入数据的操作是无效的，因为这等于改写只读存储器中的内容
4. 在执行对段寄存器SS的修改时，它的下一条指令也紧跟着被执行
5. -t指令将会引发单步中断，CPU执行1号中断程序，将标志寄存器入栈，TF、IF设置为0，CS、IP等段寄存器入栈，再对CS、IP做特定修改
6. -u指令亦可像-d指令一样指定IP的范围，如果不指定Max IP，则最多翻译20H个字节的指令
7. -e指令逐个修改内存单元中的数据时，若不想修改，直接按空格即可
8. 段寄存器不能用于算术运算的指令（add ax,ds与add ds,ax均不可）
9. 栈指针SP始终指向栈顶数据，而当栈为空时，指向栈底的高一位内存单元
10. POP指令和PUSH指令本质上都是两次操作，push为先SP-2再送入，pop为先送出再SP+2
11. 当add指令中发生溢出时，将会略去高位，且，例如add al，bl溢出时，不会因为AL溢出而将溢出的1放入AX的第三位
12. 汇编指令中不分大小写，写指令时不需要带“H”

【课上问题】

1. 南北桥芯片：一块主板上，以CPU插槽为北，北桥芯片是最靠近CPU插槽的芯片，负责与CPU联系并控制内存。南桥芯片是离CPU插槽较远的芯片，负责I/O总线之间的通讯（如通过USB，SATA等与外存储设备通信）。
2. 主频：即CPU的时钟频率，CPU每个时钟周期执行一条指令。时钟频率的SI单位是Hz，表示1s内能执行的指令数，很大程度反映着CPU的快慢
3. 外频：主板上具有一个统一的时钟信号发生器，控制着CPU与主板上其他单元（如内存）的通信，这一时钟频率为外频，随着CPU主频的提升，外频可能低于主频，传输数据的速度就会遇到“瓶颈”
4. 倍频：在CPU中，倍频=主频/外频。
5. Fsb：前端总线，不能与外频混淆。它的速度指的是CPU和北桥芯片之间数据传输的速度
6. 32位系统对内存的支持：32位系统能寻址的内存单元最多为2^32个，即最多能支持4GB的内存
7. Debug程序与我们所用的DOS环境下的编译器，对指令的支持不尽相同，如对mov ax,[2]，编译器将其认为是2H赋给ax，debug将其认为是（DS：0002）赋给ax。对16位数，debug中不可加H，在IDE中则必须加H，不然编译结果会出错