****

**汇编语言程序设计课程设计**

****

|  |  |
| --- | --- |
| **题目** | **DEBUG基本命令与数据传输指令** |
| **学院** | **计算机学院** |
| **专业** | **计算机科学与技术** |
| **学号** |  |
| **姓名** |  |
| **指导老师** | **余波老师** |
| **实验时间** | **2022年4月24日** |

# 第一次实验：DEBUG基本命令与数据传输指令

学号: 姓名:

## 一、实验目的：

1、熟练掌握DEBUG的基本调试命令，能够使用DEBUG编写、调试汇编语言程序片段。

2、在理解数据传输指令的基础上按照实验内容中指定的程序片段对程序进行调试和记录；

## 二、实验环境（硬件，软件环境）：

硬件环境： MACOS系统

软件环境：DOSBox v0.74-3-3

## 三、实验内容及要求（算法，程序，步骤，方法）：

1、实验内容

（1）在实验报告中说明所学DEBUG基本命令的使用方法。

（2）在实验报告中记载每条指令执行后，寄存器、标志位的变化情况，解释指令的执行原理。

2、实验要求

（1）能使用debug单步跟踪汇编执行。

（2）能使用debug查看寄存器的结果以及程序执行情况。

3、实验步骤

（1）DEBUG的运行与退出

由于win10系统已不再支持debug，故需要重新下载并安装8086虚拟机和debug程序，从官网上下载相关的文件（包括DOSBox主程序，masm.exe和link.exe等配置文件）并进行配置。正确加载程序后运行DOSBOX虚拟机，界面如图1所示。

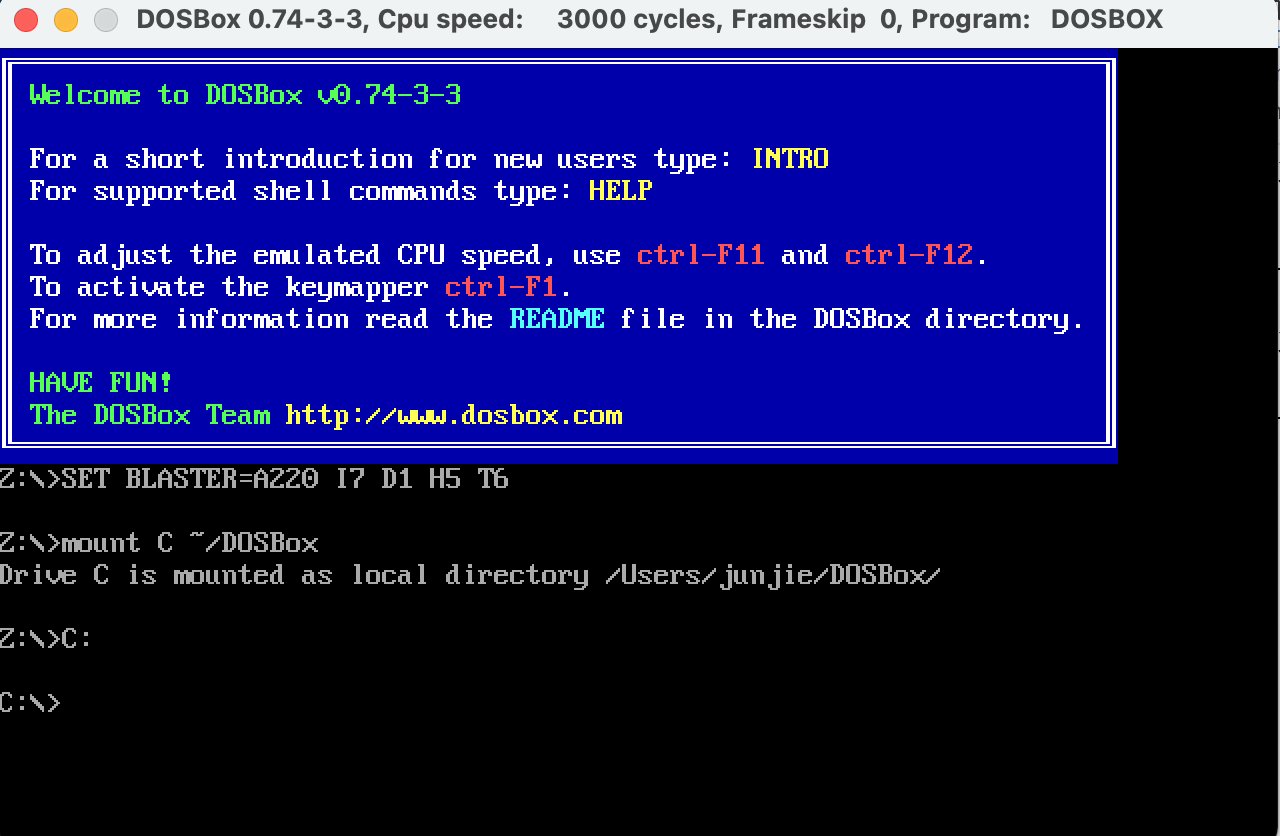


图1 DOSBox程序界面

在命令行直接输入“DEBUG”并回车，则运行DEBUG程序，其提示符为一个短横线。

如果要退出DEBUG，则使用q命令：-q。退出DEBUG后，窗口退回到DOS命令行提示符。界面如图2所示：

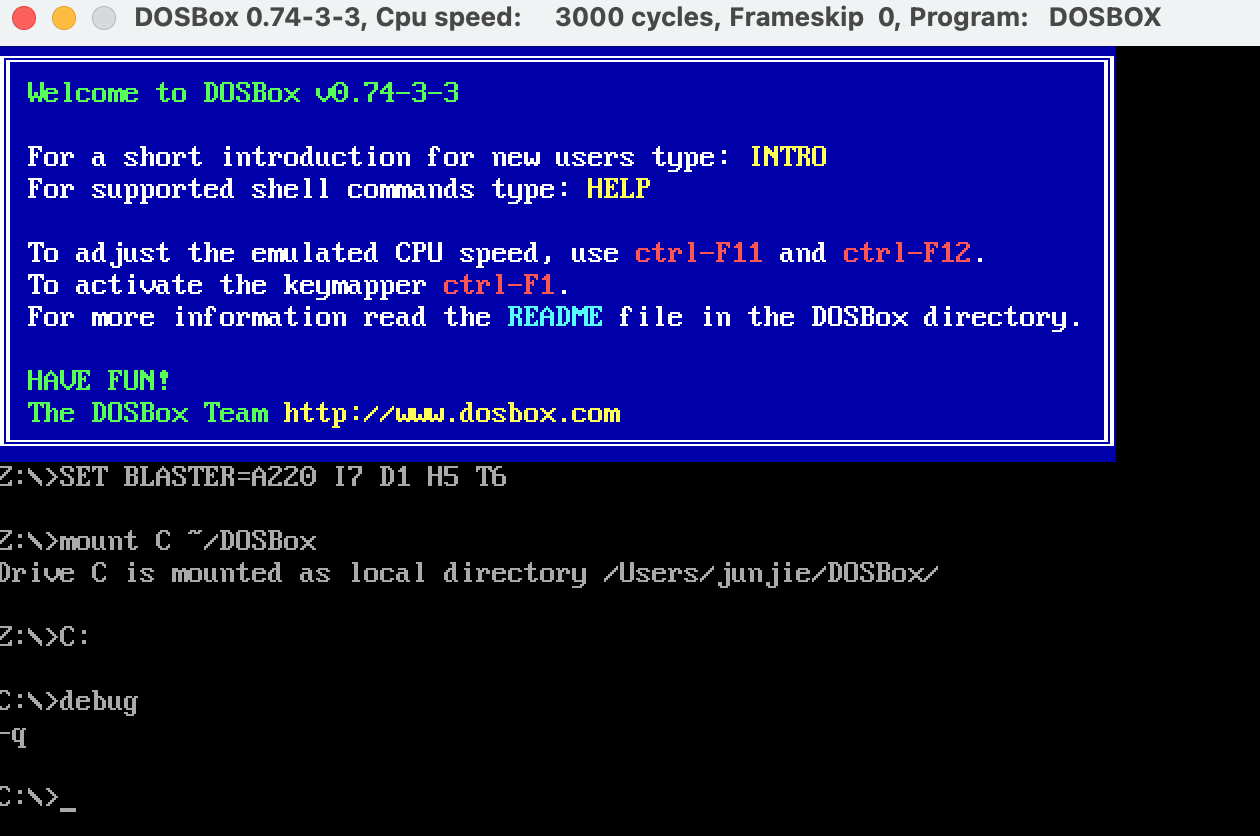


图2 DEBUG的运行与退出

（2）使用r命令查看寄存器内容

输入汇编指令后，可以借助指令对其进行运行查看。“-r”命令用于查看各寄存器的内容（在指令后输入寄存器的名字便可单个查看寄存器的内容）。使用该命令查看的寄存器内容如图3所示。查看寄存器的内容时需要注意，对于FR标志寄存器的内容，debug不会直接给出，而是会以字母代码的形式给出，对照表如下所示。

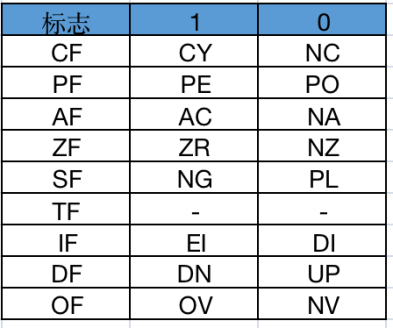


表1 FR标志寄存器内容对照表

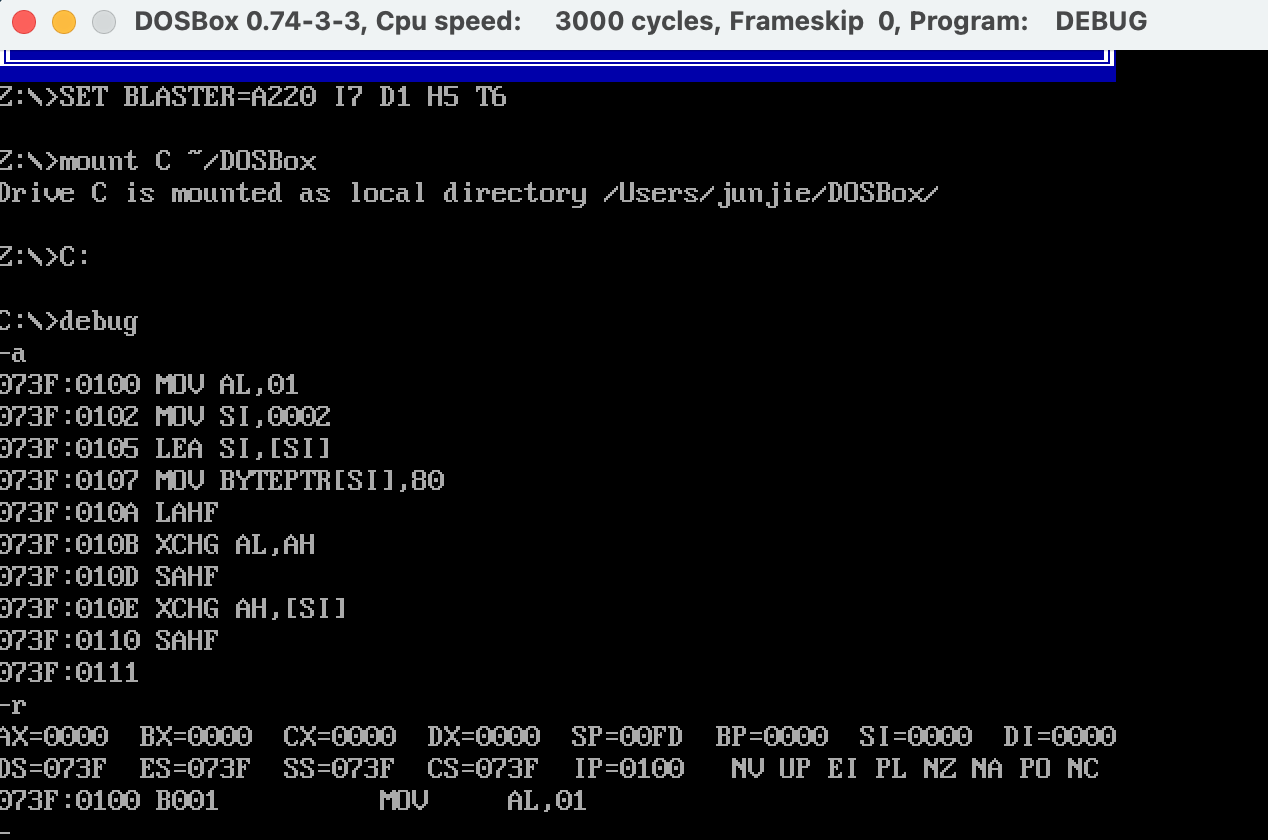


图3 调用-r指令

（3）DEBUG基本调试命令——汇编命令A

对虚拟机的目录进行配置，使目录指向用于存放编译文件的文件夹。随后，进入debug开始对源程序的编译。首先输入“debug -a”进行源代码的输入(“-a”指令用于在debug模式下输入汇编代码)，本实验的源代码固定，如图4所示。在输入时，若出现明显的语法错误，debug会在错误处显示一个“error”并提示用户重输。输入结束后，按回车退出编辑，如图5。

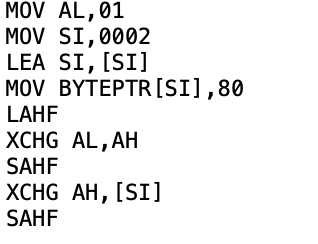


图4 源代码

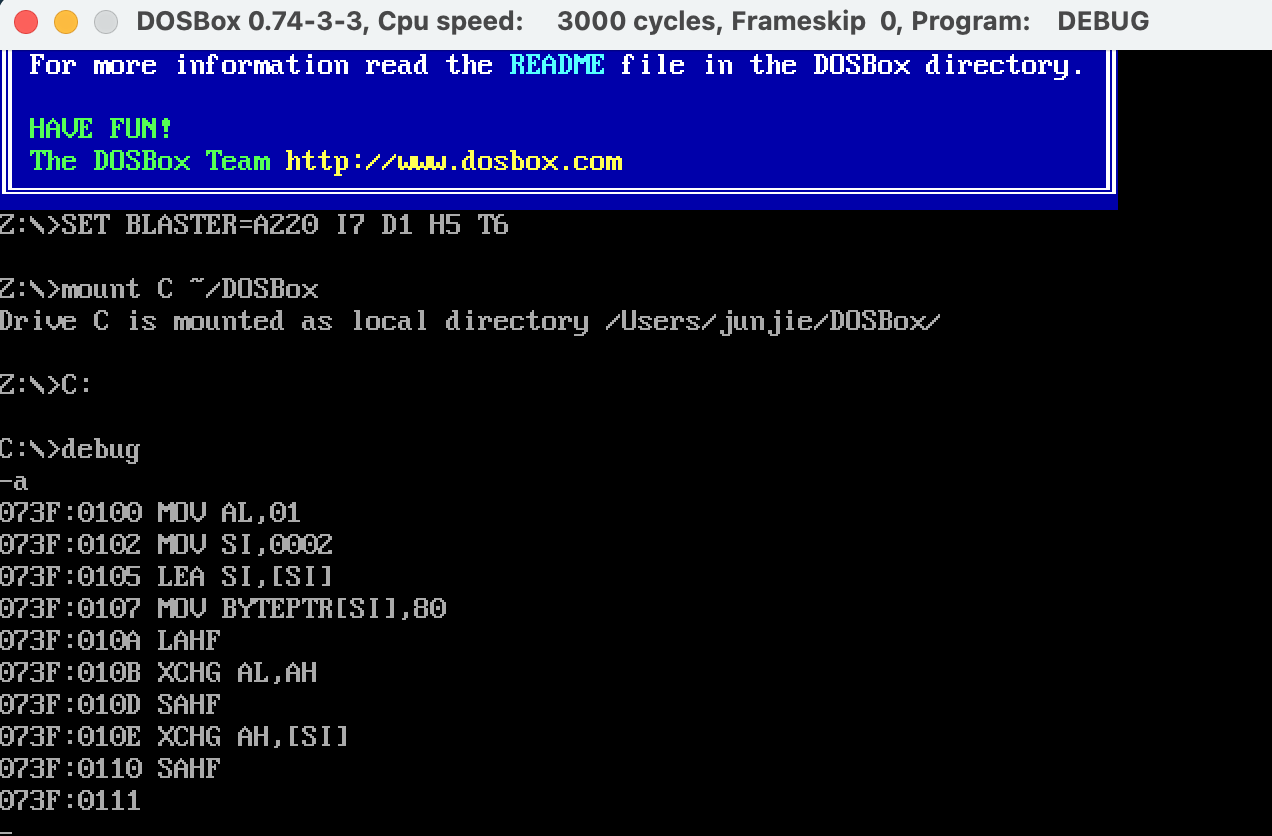


图5 汇编指令在debug模式下键入完毕

（4）DEBUG基本调试命令——反汇编命令U

将指定逻辑地址起始的机器指令序列反汇编为汇编指令序列，并在屏幕上回显。

使用：-u 073D:0107以查看该地址汇编指令，如图6所示。

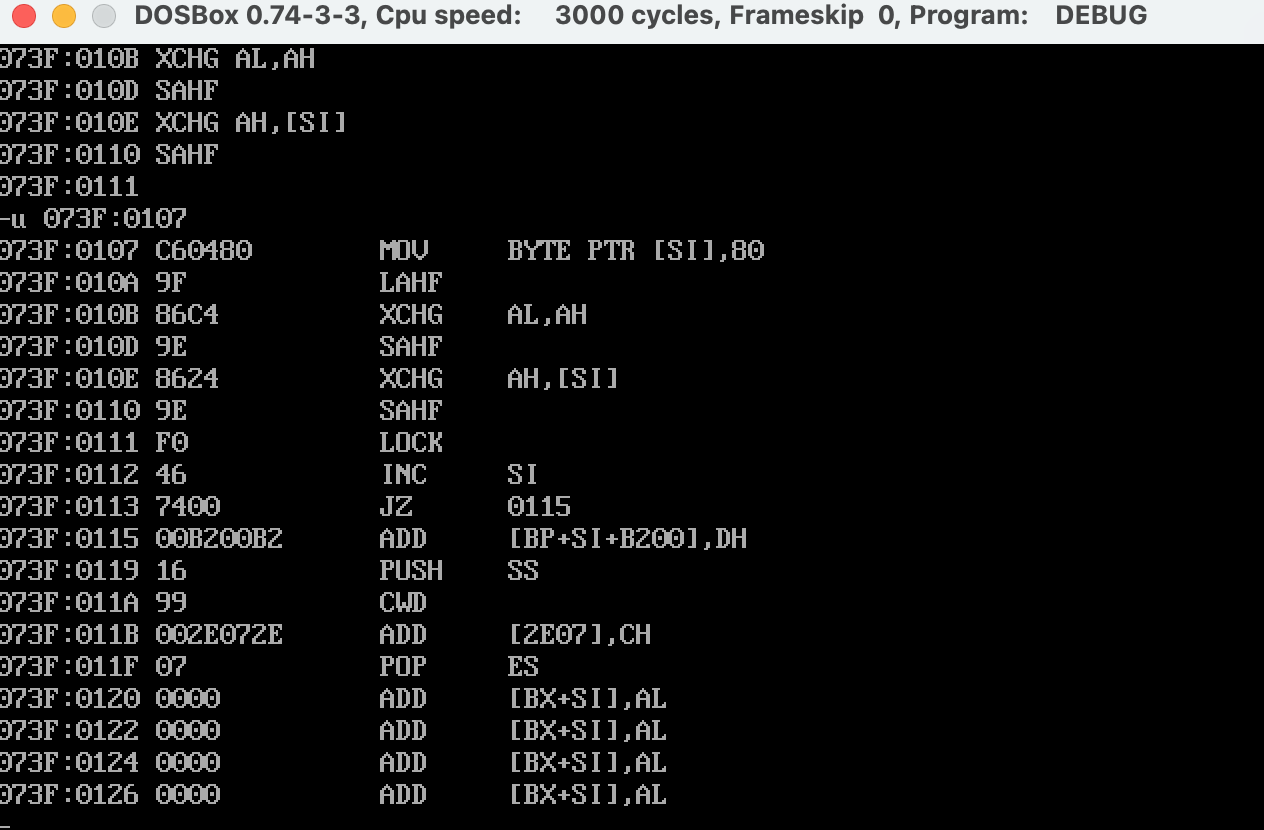


图6 汇编指令-u的调用

（5）DEBUG基本调试命令——单步调试命令T

对源程序进行单步跟踪检查，其指令为“-t”，每输入一次则单步执行下一句指令，并给出实时的寄存器内容。图7-图8反映了对于该段代码运用“-t”命令单步跟踪后的结果。

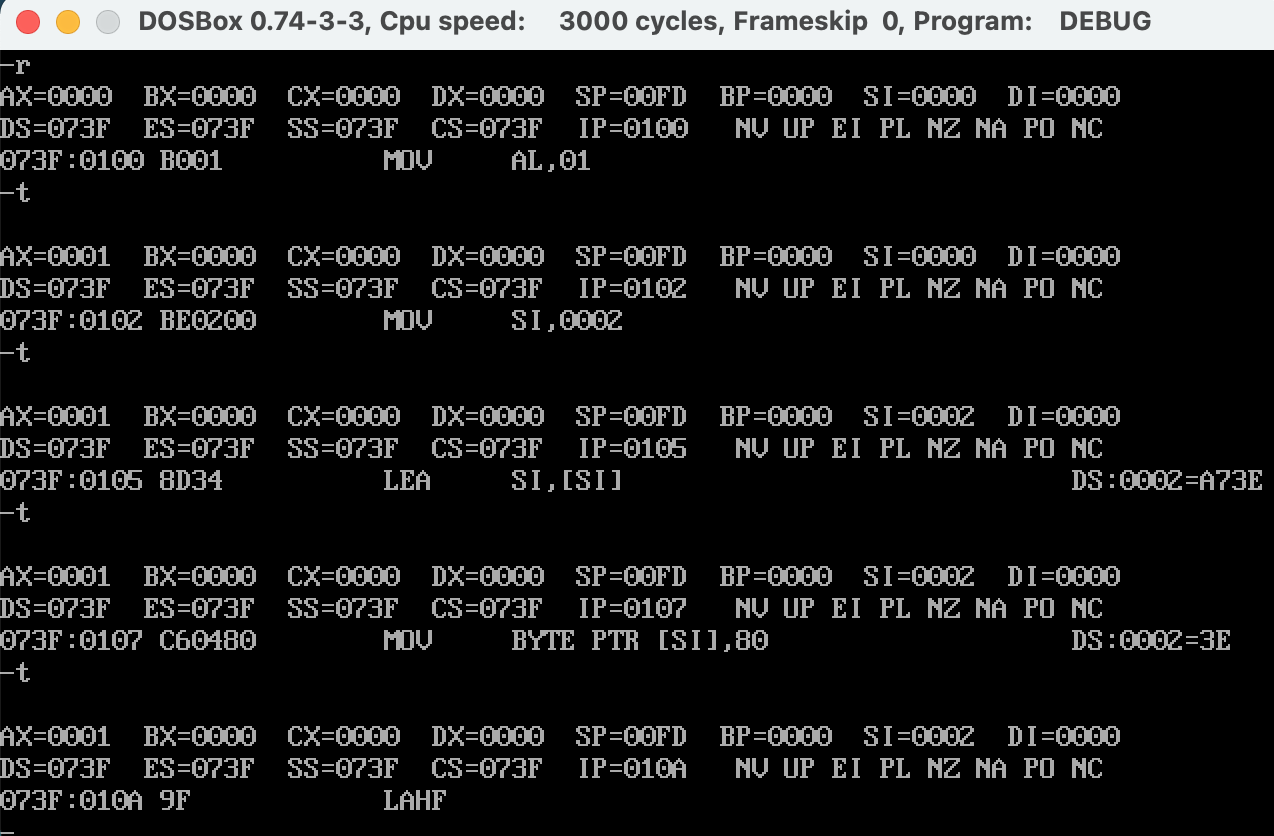


图7 执行结果1

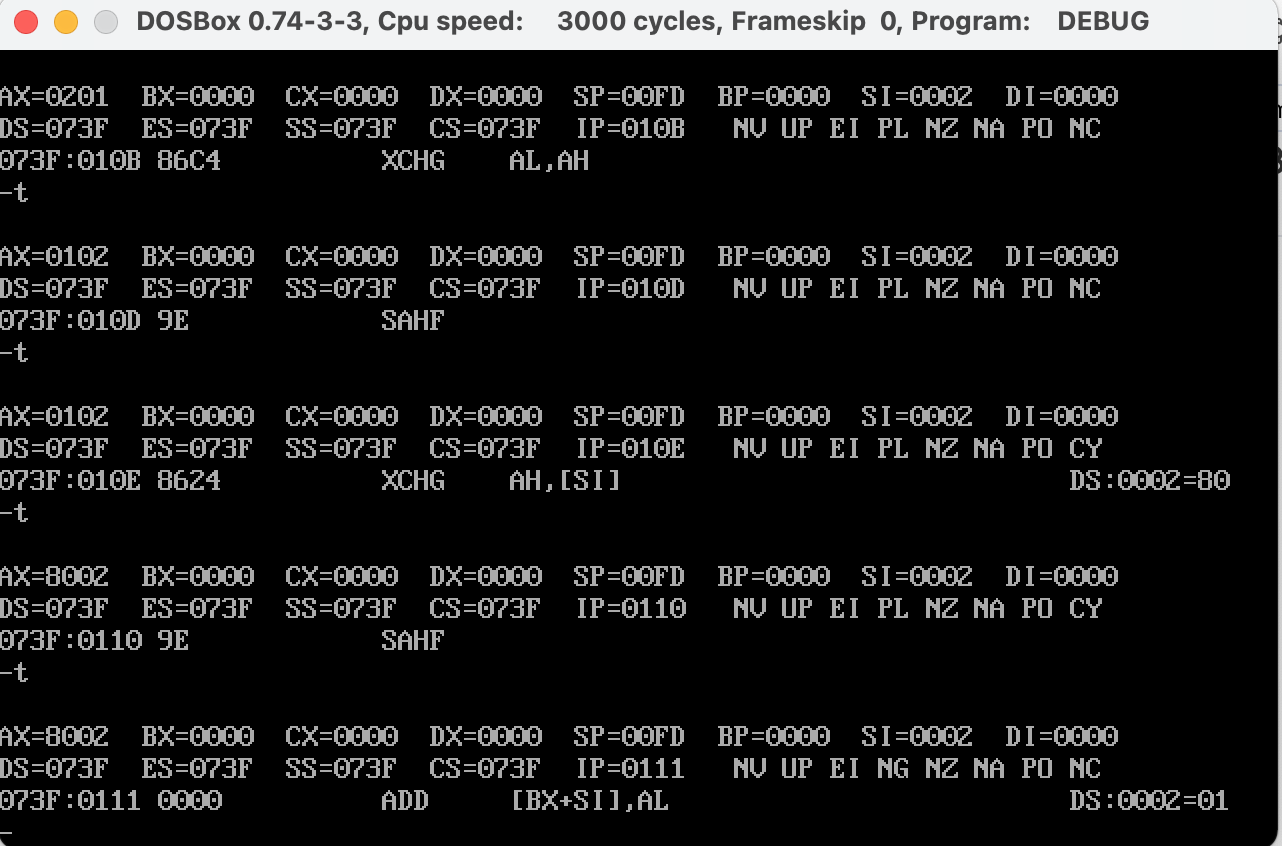


图8 执行结果2

4、实验源代码

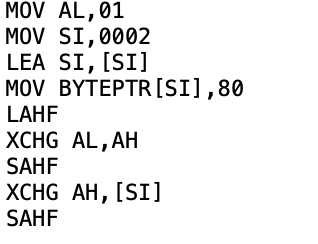


图9 源代码

## 4、实验结果分析：

## 整个程序执行结束后，各个寄存器的结果如图7所示。

AX=8002H, BX=0000H, CX=0000H, DX=0000H, SP=00FDH, BP=0000H, SI=0002H, DI=0000H, DS=073FH, ES=073FH, SS=073FH, CS=073FH, IP=0111H, FR:(OF=0, DF=0, IF=0, SF=1, ZF=0, AF=0, PF=0, CF=0).

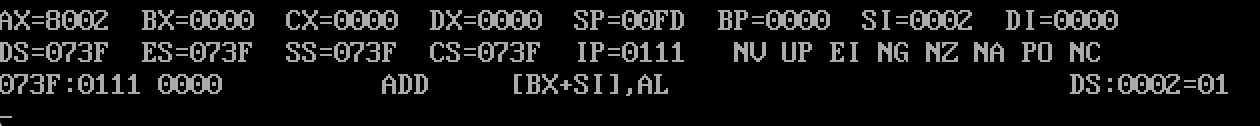


图10 程序执行结束时寄存器的结果

下面解释各步跟踪情况。

1.执行语句“MOV AL, 01H”时，01被送入AX低位，该指令不影响标志位，因此标志位保持不变，而AX被置为“0001”.

2.执行语句“MOV SI,0002H”时，0002被送入SI，标志位不变，SI被置为“0002”.

3.执行语句“LEA SI,[SI]”时，[SI]的有效地址被送入SI，该过程不影响标志位，而SI被置为0002.

4.执行语句“MOV BYTE PTR [SI],80”时，[SI]所指向的数被置为80，不影响标志位。

5.执行语句“LAHF”时，FR寄存器低8位被送入AH，不影响标志位，AH被置为02，即二进制的“00000010”.从这里可以看到，此时FR寄存器的低8位中，标志位全部置0.

6.执行语句“XCHG AL,AH”时，AH和AL的值被交换。该过程不影响标志位，AH变为01（即二进制的“00000001”），AL变为02.

7.执行语句“SAHF”时，AH的值被送入FR寄存器的低8位，标志位发生改变。此时FR被置为“00000001”，即CF从0置1.

8.执行语句“XCHG AH,[SI]”时，AH和SI寄存器所指向的值被交换。从前方程序可以判断，[SI]所指向的数为80，故此时AH被置为80(即二进制的“10000000”)，而[SI]被置为01.该过程不改变标志位。

9.执行语句“SAHF”时，AH的值被送入FR寄存器的低8位，标志位发生改变。此时FR被置为“10000000”，即CF从1置0，而SF从0置1.

## 5、实验体会：

实验过程应注意：

1.在debug的“-a”模式中，所有的操作数默认只能使用十六进制，如涉及十进制操作数需自己换算。

2.“-a”编辑模式下，编写代码容易出错(若出错则会提示ERROR)，因此该命令只适合编写短代码。

通过本次实验，了解了debug基本指令的用法，也学会了如何查看寄存器内存储的值，实验成功。