****

**汇编语言程序设计课程设计**

****

|  |  |
| --- | --- |
| **题目** | **DEBUG基本命令与算术运算指令** |
| **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5月10日** |

# 第二次实验：DEBUG基本命令与算术运算指令

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、在理解算术运算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）分析指定的源程序，执行到指定位置后（在源程序中有标注），寄存器、标志位的取值，解释指令的执行原理，并在实验报告中对必要的寄存器、标志位状态加以记载。

（3）分析程序片段的功能，必在实验报告中加以说明。

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

3.1 DEBUG基本命令的使用方法

（1）DEBUG的运行与退出

配置debug所需的文件（包括DOSBox主程序，masm.exe和link.exe等配置文件）并正确加载程序后运行DOSBOX虚拟机，界面如图1所示。

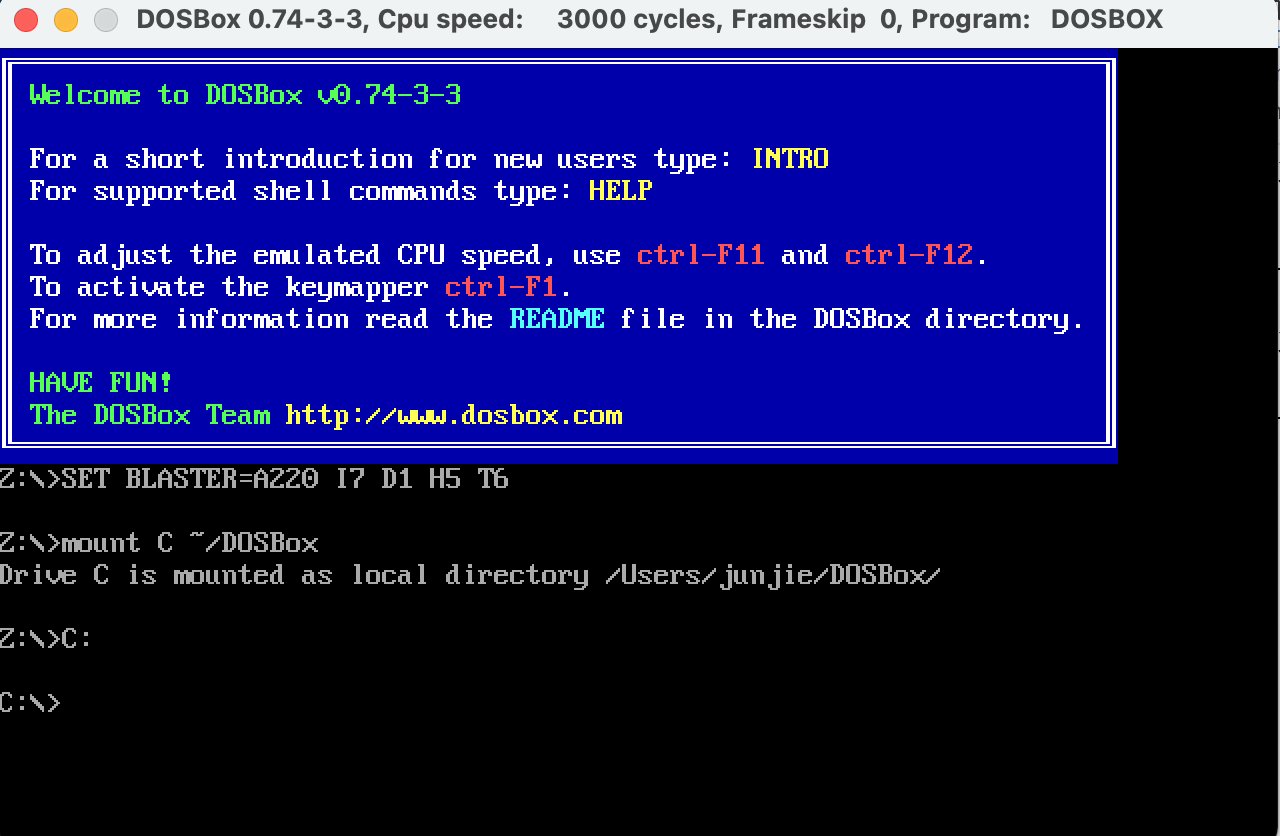


图1.1 DOSBox程序界面

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：

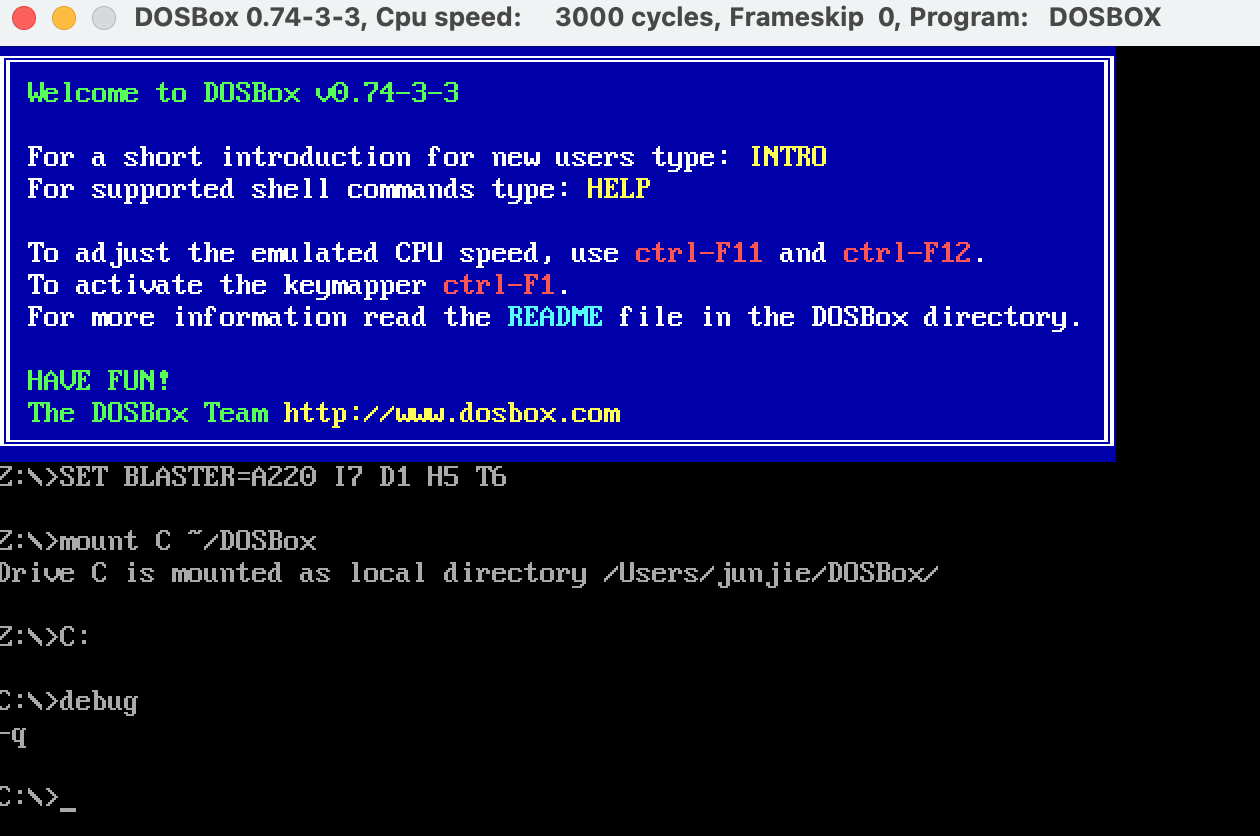


图1.2 DEBUG的运行与退出

（2）断点调试命令G

1.功能：从当前CS、IP指示的位置开始连续执行程序，执行到指定断点处（不包括断点地址指示的指令）暂停执行，在屏幕上显示CPU现场。

2.使用方法：

1. 连续执行，无断点：-g
2. 连续执行到断点位置：-g XXXX:XXXX

3.演示（此时以实验报告指定源码为例，以-A命令键入测试源代码）

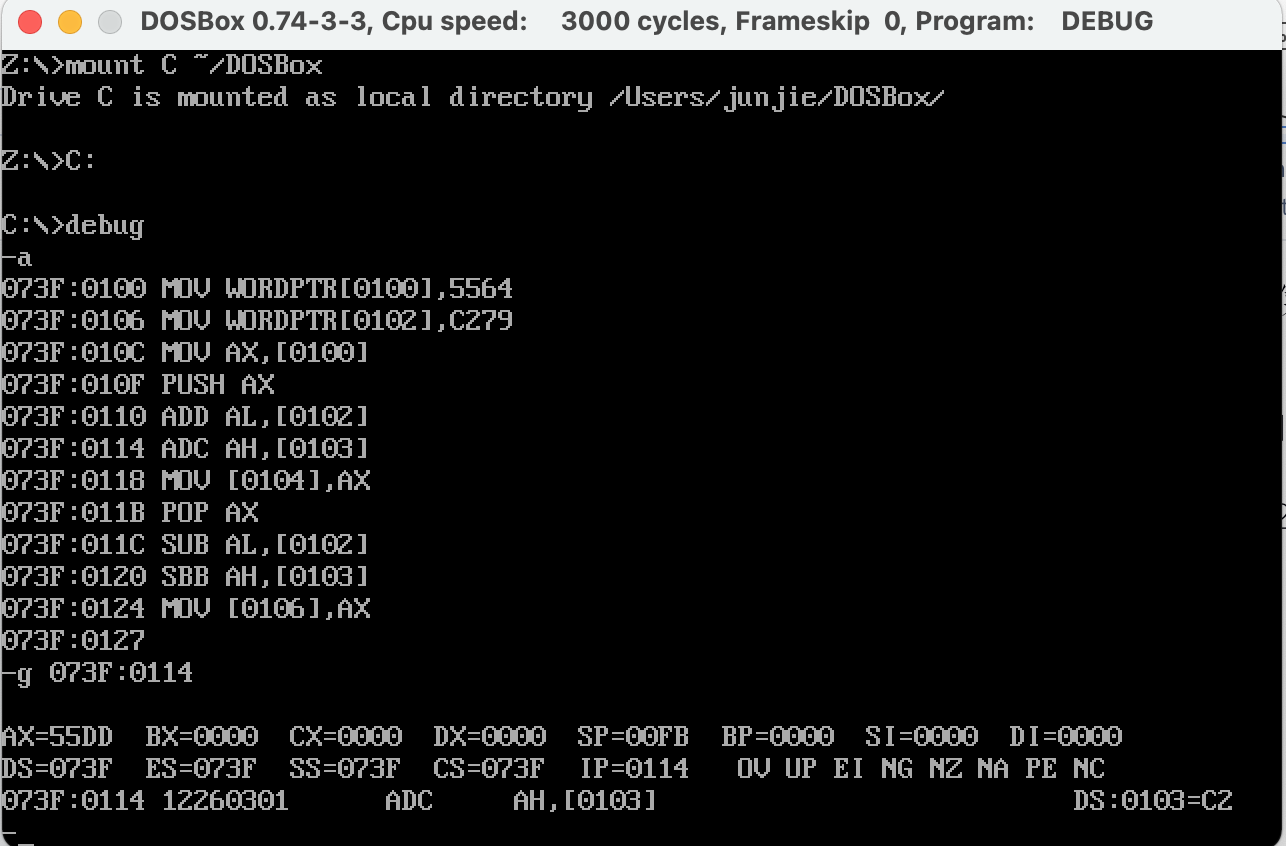


图1.3 调用-g XXXX:XXXX指令（以运行到073F:0114地址为例）

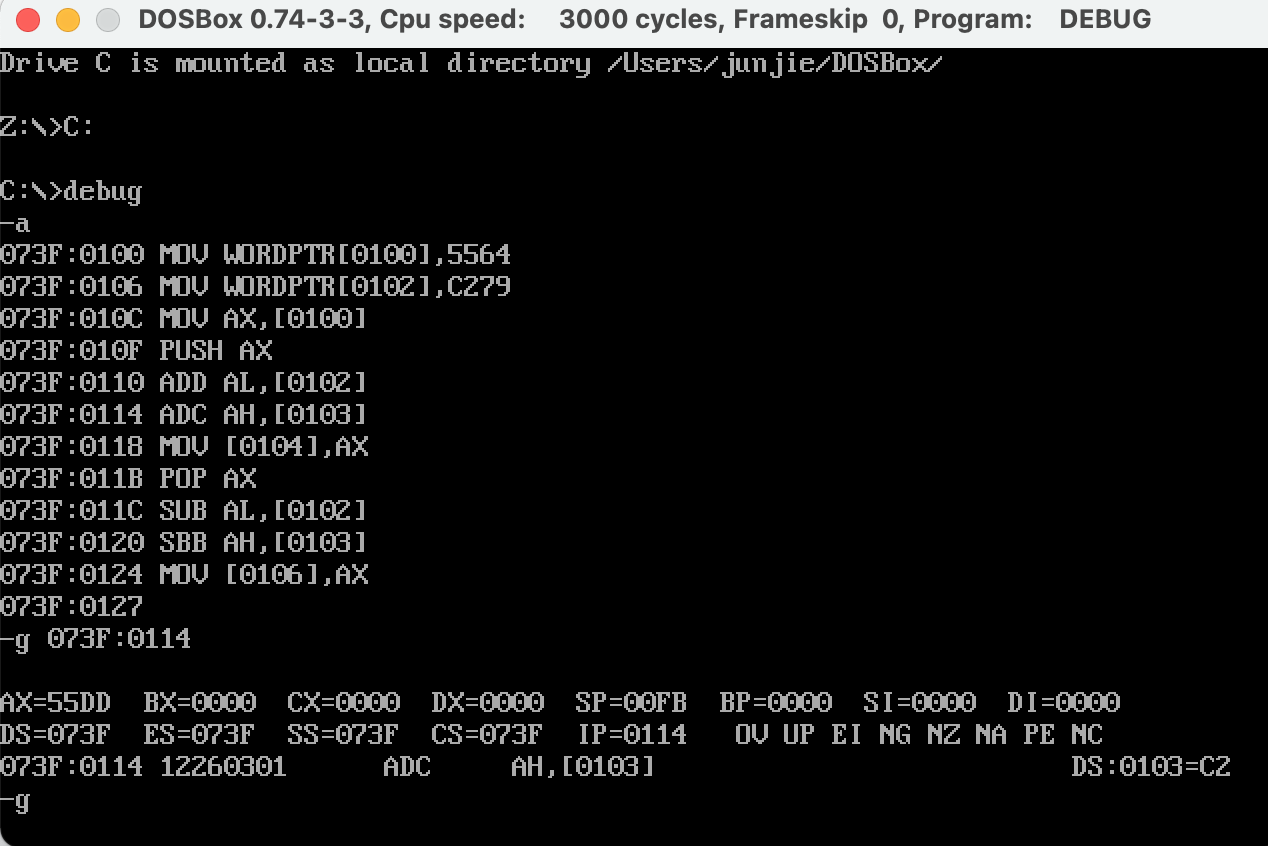


图1.4 调用-g指令（注意：此时程序会一直向下执行，没有终止）

4.注意：断点位置必须指定为某条指令的首字节地址

（3）内存单元查看命令D

1.功能：从当前CS、IP指示的位置开始连续执行程序，执行到指定断点处（不包括断点地址指示的指令）暂停执行，在屏幕上显示CPU现场。

2.使用方法：

1）从默认数据段内存地址开始显示（或接上次所显示的地址继续显示）：-D

2）从指定内存地址开始显示：-D XXXX:XXXX

3.演示（此时以实验报告指定源码为例，以-A命令键入测试源代码）

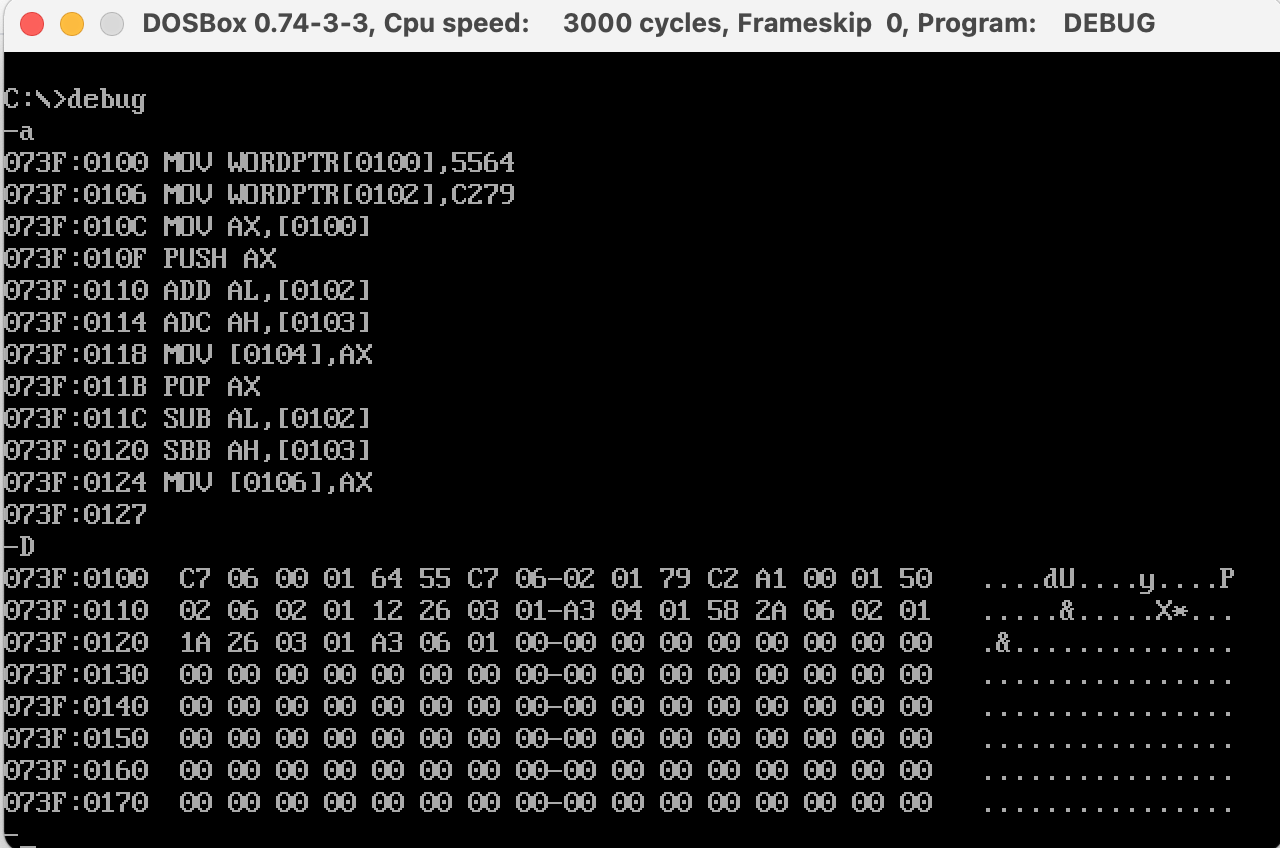


图1.5 调用-d指令

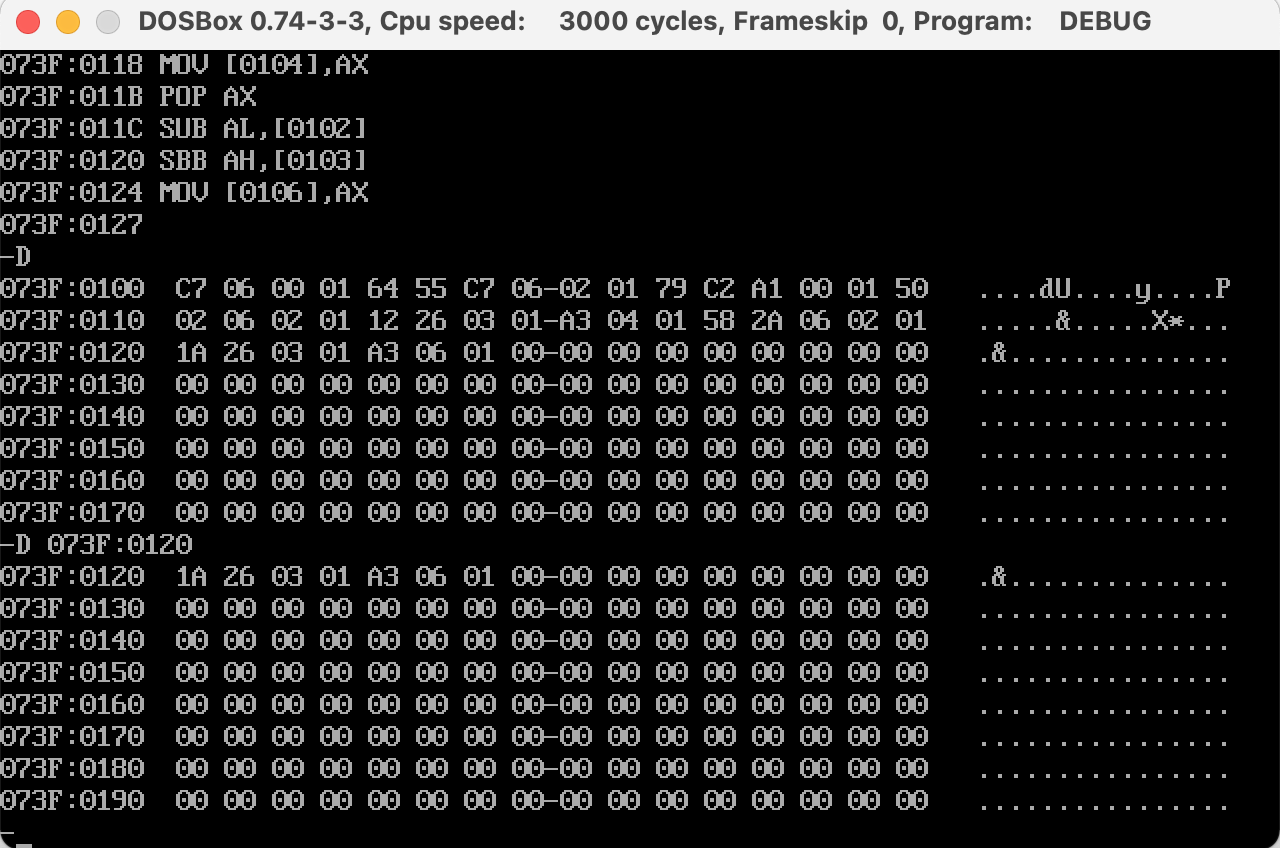


图1.6 调用-d XXXX:XXXX指令（以查看073F:0120内存为例）

1. 注意
2. 从指定位置开始，输出128字节的内容
3. 输出的内容由起始位置，内存单元内容，每个内存单元对应的ASCLL码字符

（4）内存单元修改命令E

1.功能：修改指定地址内存单元的内容。

2.使用方法：-E XXXX：XXXX

3.演示（此时以实验报告指定源码为例，以-A命令键入测试源代码）

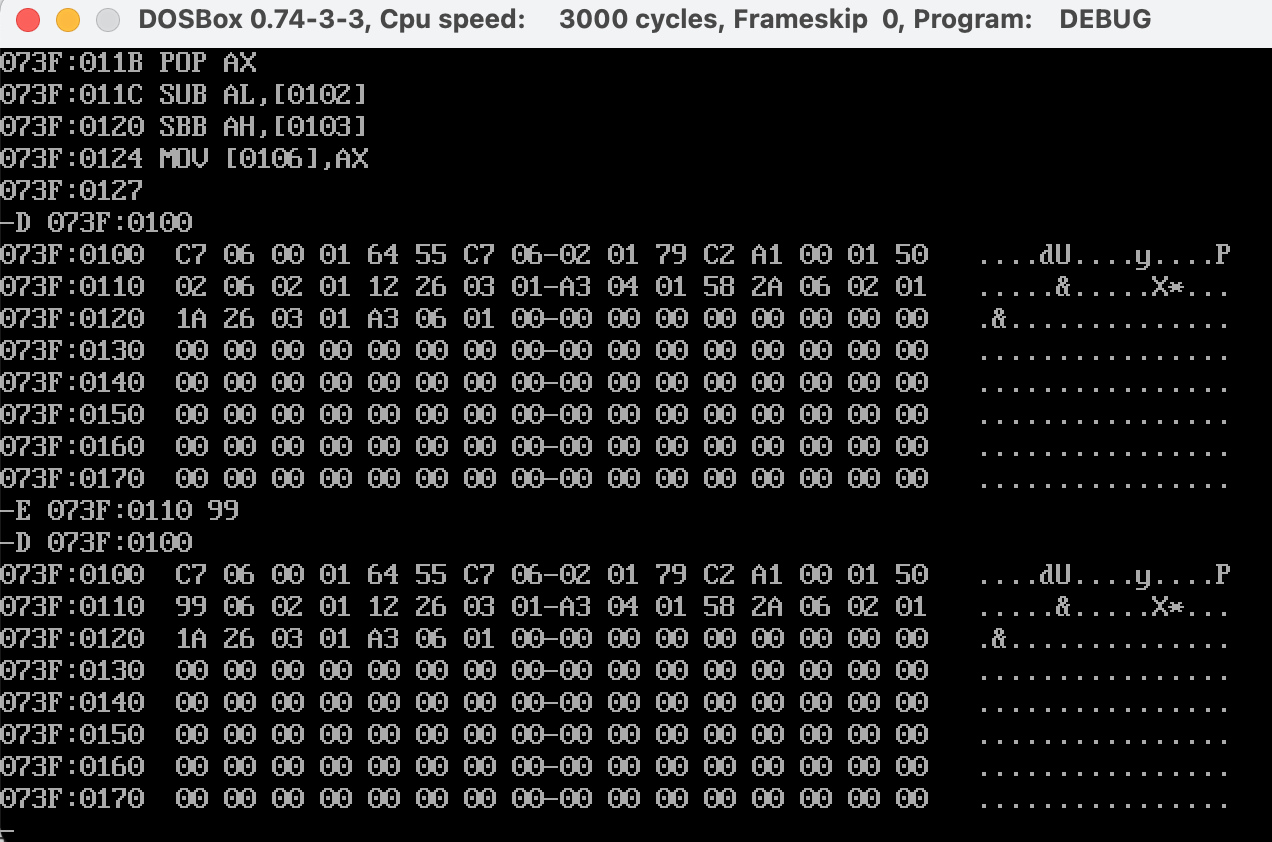


图1.7 调用-e XXXX:XXXX指令

1. 注意
2. 从指定位置开始，输出128字节的内容
3. 输出的内容由起始位置，内存单元内容，每个内存单元对应的ASCLL码字符

3.2 源程序的调试及分析

1. 首先运行DOSBox，键入-debug进入debug模式
2. 为了避免数据段、堆栈段内的操作覆盖代码段中已录入的指令，影响重复调试。在调试程序片段时，应使用r命令先修改DS、SS段寄存器，使之与CS指向不同位置。建议将DS提供的原始段基值加0100H，将SS提供的原始段基值加0200H。

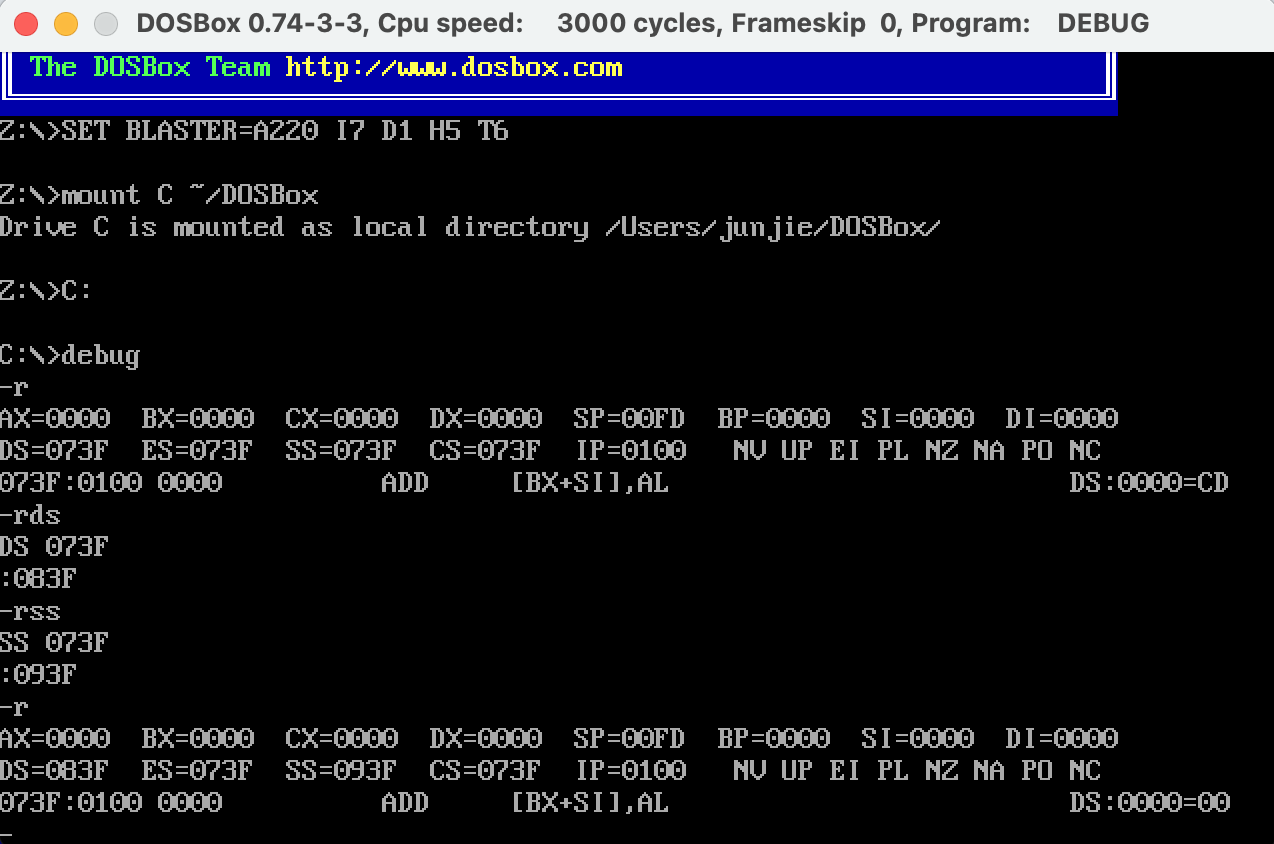


图2.1 修改段寄存器中的内容

1. 键入-a输入测试源代码

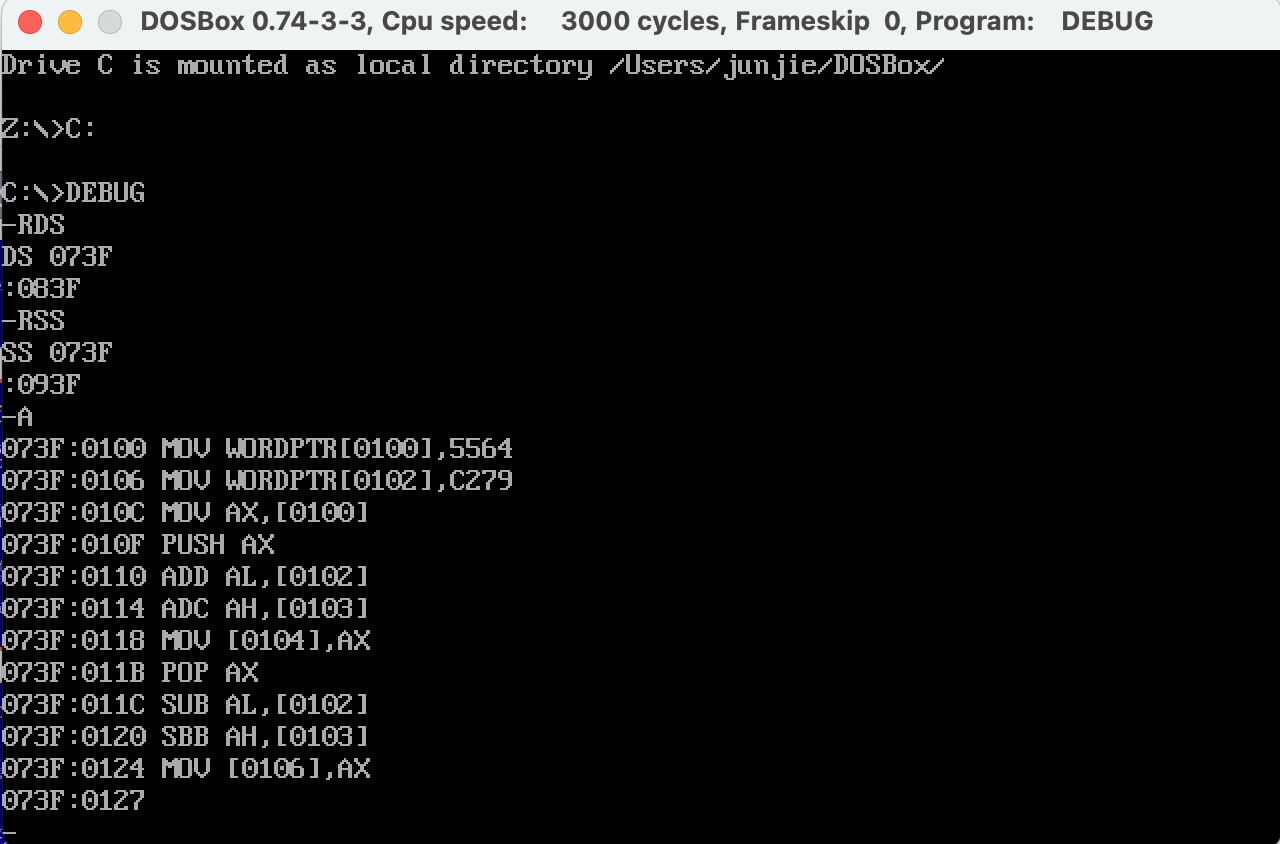


图2.2 键入测试源代码

1. 使用-g命令运行至ADD AL,[0102H]即073F:0114的断点停止

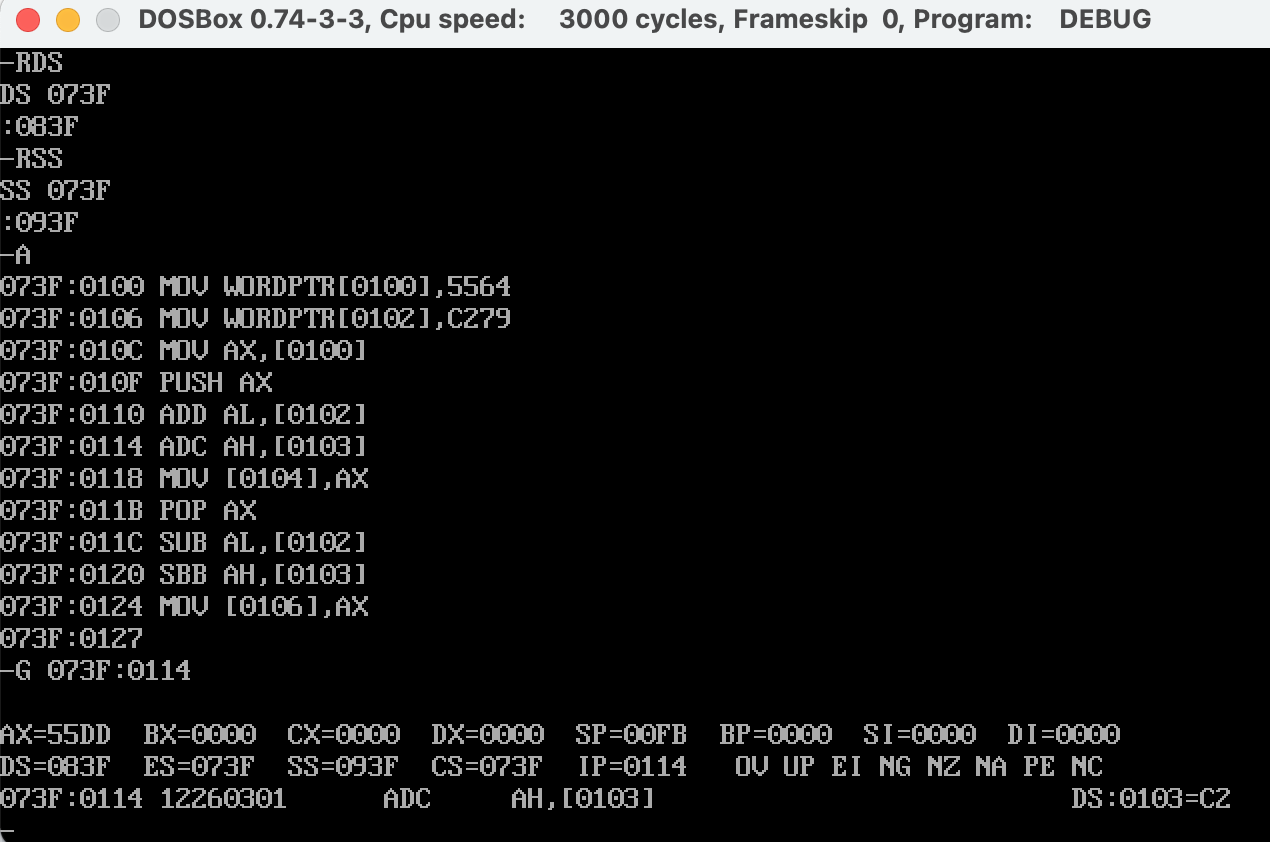


图2.3 第一个断点分析

下面对该步寄存器进行分析，在该条指令之前，先将5564H与C279H分别送入073F:0100与073F:0102中，再将[0100H]的内容（5564H）送入AX中，弹出AX栈，开始执行该步指令，即将[0102H]的内容（79H）与AL内容（64H）相加，相加结果送入AL中。

下面对该步标志位取值进行分析，由79H+64H，此时：

相加后AL并没有向前进位，CF=0；相加后符号位发生变化，判断为溢出，OF=1；相加结果为DDH，具有偶数个“1”，PF=1；不发生辅助性进位，AF=0；运算结果不为0，ZF=0；最高11011101为1，SF=1；



1. 使用-g命令运行至ADC AL,[0103H]即073F:0118的断点停止

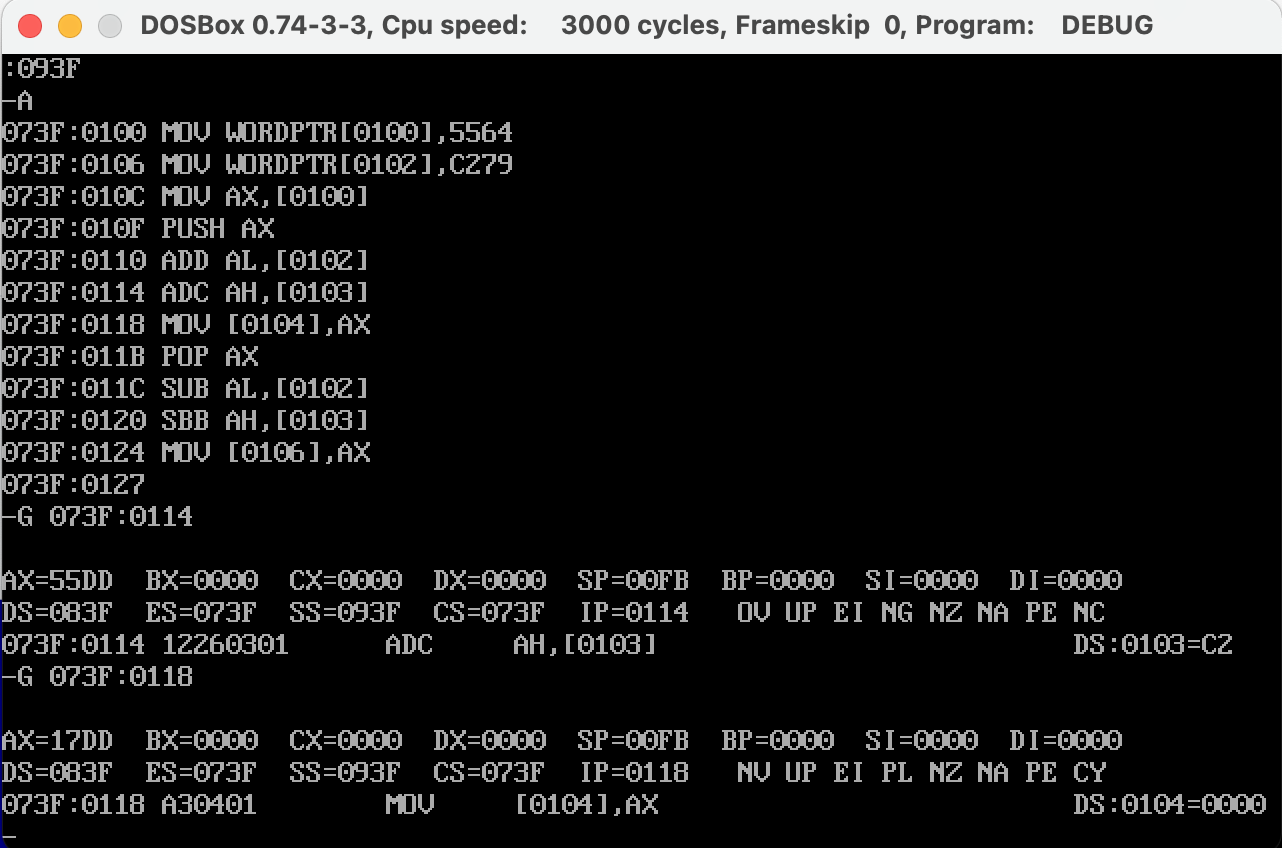


图2.4 第二个断点分析

下面对该步寄存器进行分析，开始执行该步指令，即将[0103H]的内容（C2H）与AH内容（55H）以及CF（此时为0）相加，相加结果送入AH中。

下面对该步标志位取值进行分析，由C2H+55H+0，此时：

相加后AH向前进位，CF=1；相加后符号为不发生改变，不发生溢出，OF=0；相加结果为17H，具有偶数个“1”，PF=1；不发生辅助性进位，AF=0；运算结果不为0，ZF=0；最高00010111为0，SF=0；



1. 使用-g命令运行至SUB AL,[0102H]即073F:0120的断点停止

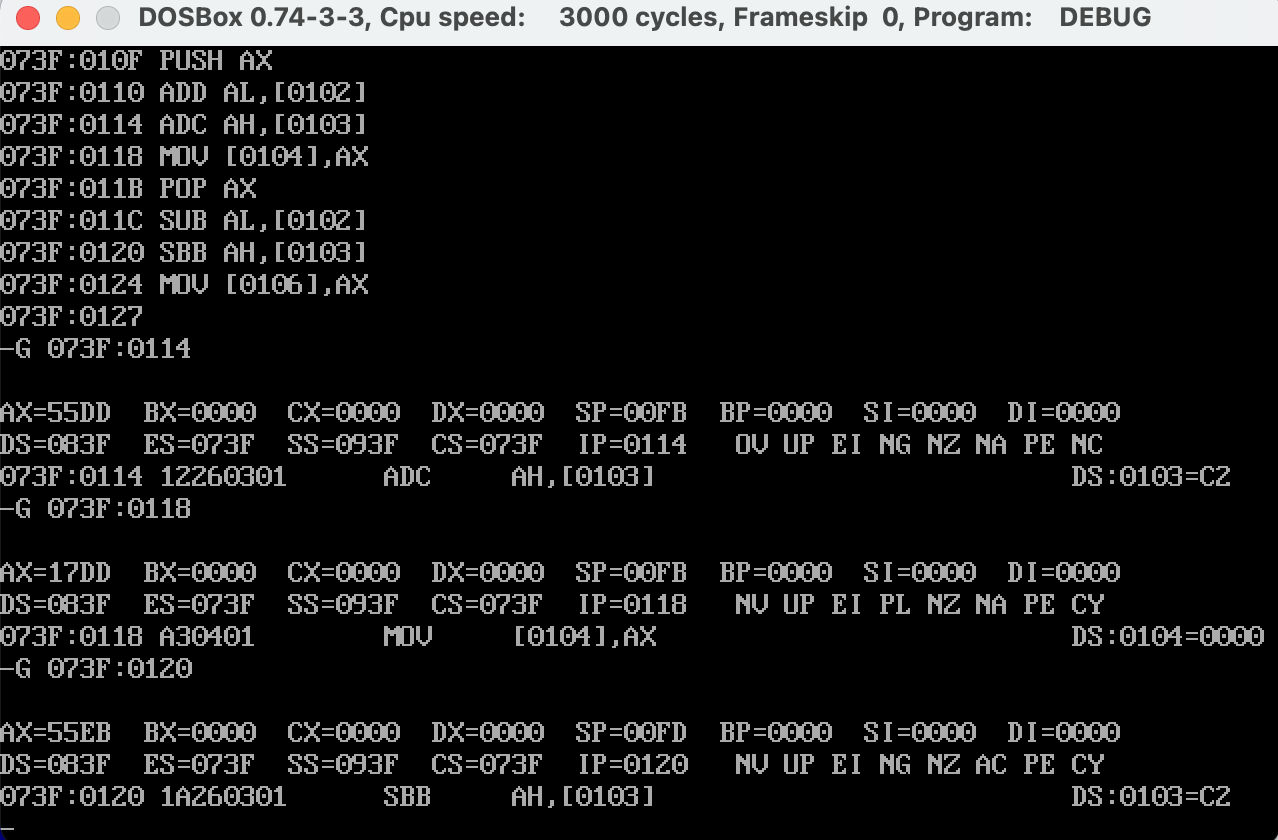
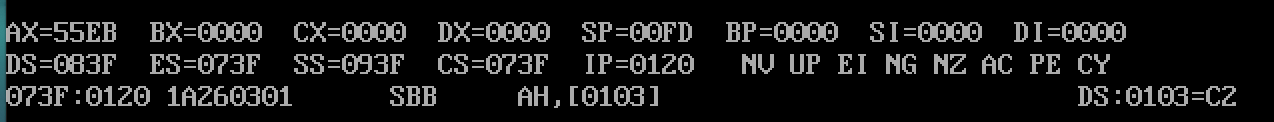


图2.5 第三个断点分析

下面对该步寄存器进行分析，弹出AX，AX恢复为5564H，开始执行该步指令，即将[0102H]的内容（79H）作为减数与AL（64H）内容作为被减数相减，相减结果送入AL中。

下面对该步标志位取值进行分析，此时：

相减后AL发生借位，CF=1；相减后不发生溢出，OF=0；相减结果为EBH，具有偶数个“1”，PF=1；发生辅助性进位，AF=1；运算结果不为0，ZF=0；最高11101011为0，SF=1；



1. 使用-g命令运行至SBB AH,[0103H]即073F:0124的断点停止

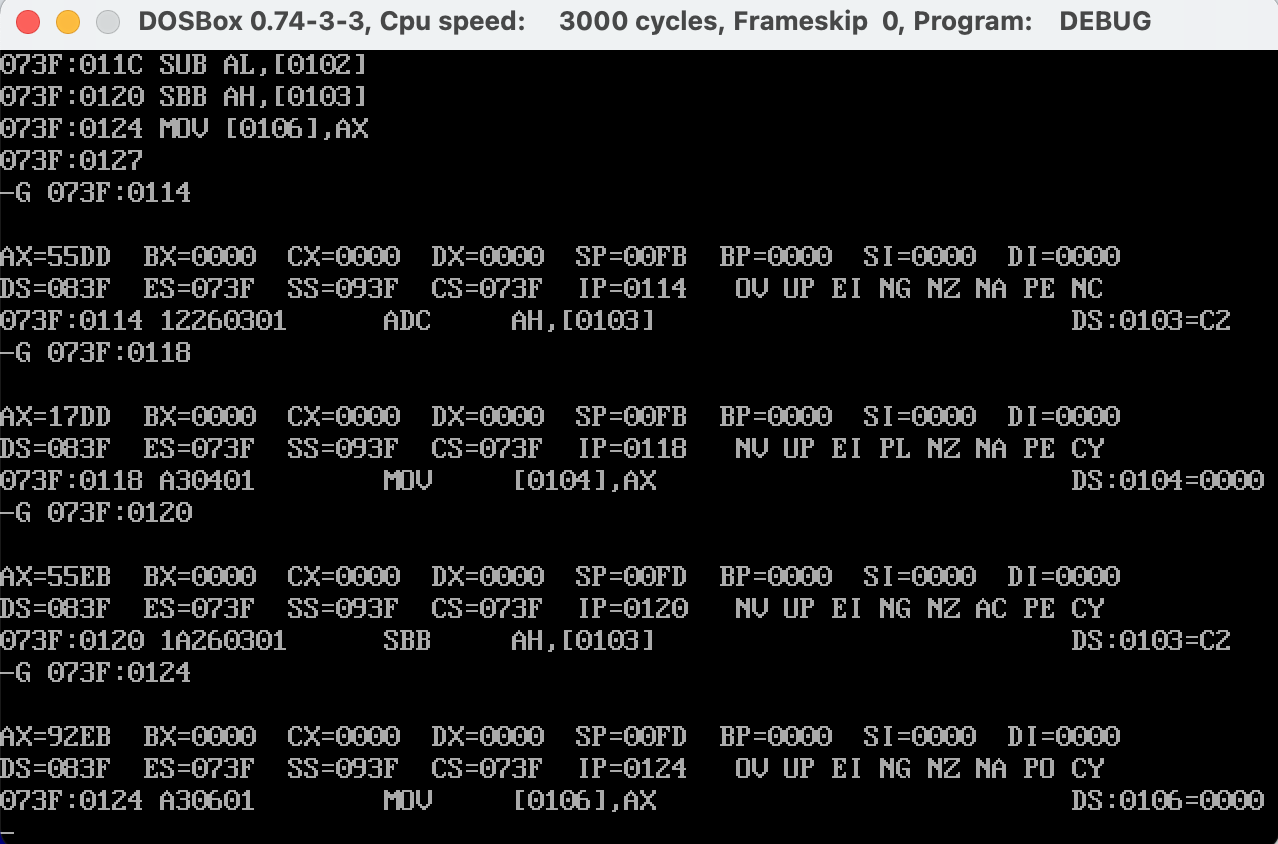


图2.6 第四个断点分析

下面对该步寄存器进行分析，开始执行该步指令，即将[0103H]的内容（C2H）与CF作为减数与AH（55H）内容作为被减数相减，相加结果送入AH中。

下面对该步标志位取值进行分析，此时：

相减后AL发生借位，CF=1；相减后发生溢出，OF=1；相减结果为92H，具有奇数个“1”，PF=0；不发生辅助性进位，AF=0；运算结果不为0，ZF=0；最高10010010为1，SF=1；



1. 使用-g命令运行结束即073F:0127的断点停止

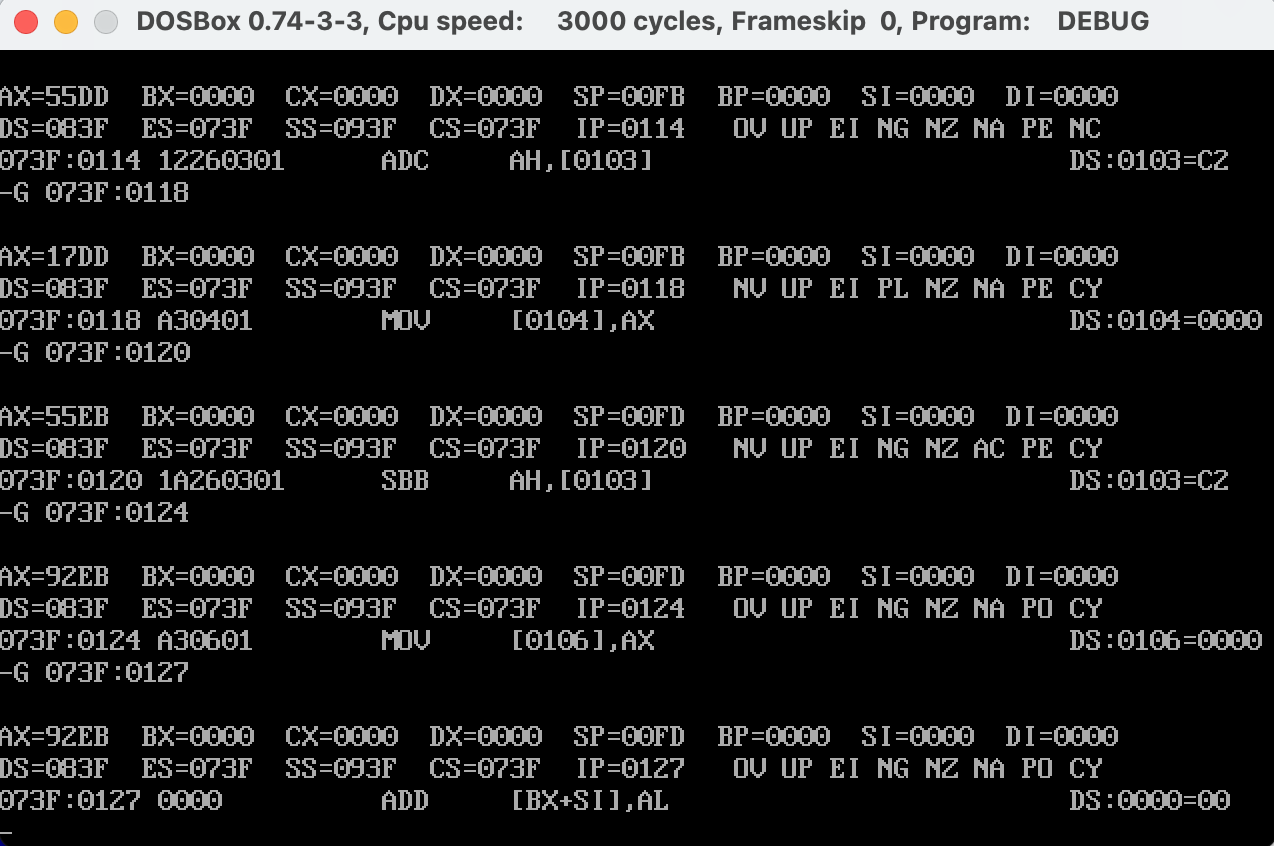
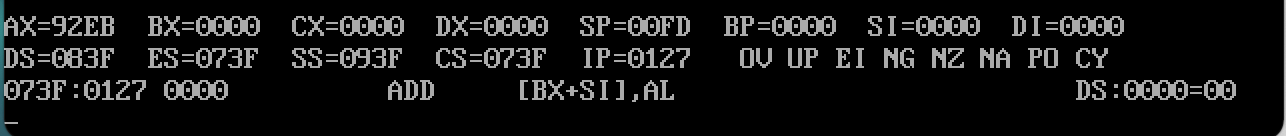


图2.7 最终结果分析

下面对该步寄存器进行分析，开始执行该步指令，即将AX的内容送入[0106H]中

不影响标志位，此时AX=92EBH，[0104H]=17DDH，[0106H]=92EBH



4、实验源代码

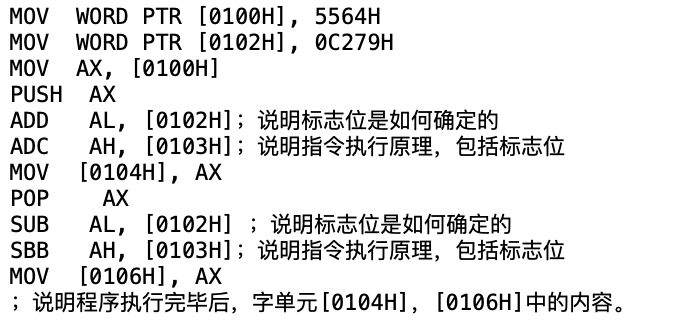
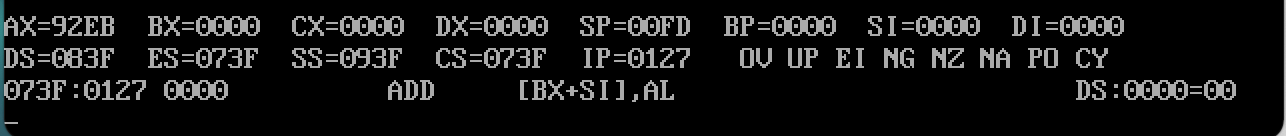


图9 源代码

## 实验结果分析：

根据以上对程序每步调试的分析，可以得到该程序的功能是将AX分别与[0102H]字内存单元中的内容分别相加与相减，分别存放在[0104H]，[0106H]字内存单元中。

实验结果符合预期功能设计，实验成功！



## 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。

3.在调试上述程序片段时，应使用r命令先修改DS、SS段寄存器，使之与CS指向不同位置。建议将DS提供的原始段基值加0100H，将SS提供的原始段基值加0200H。避免数据段、堆栈段内的操作覆盖代码段中已录入的指令，影响重复调试。

4.调试时要有耐心，认真分析每一步结果。

5.PUSH指令出栈会恢复。

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。