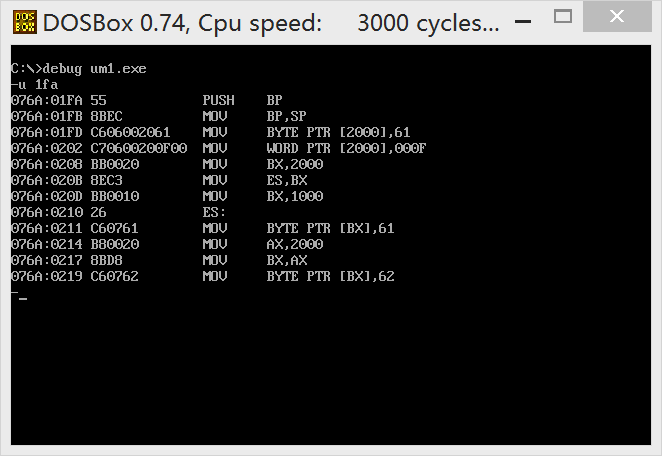
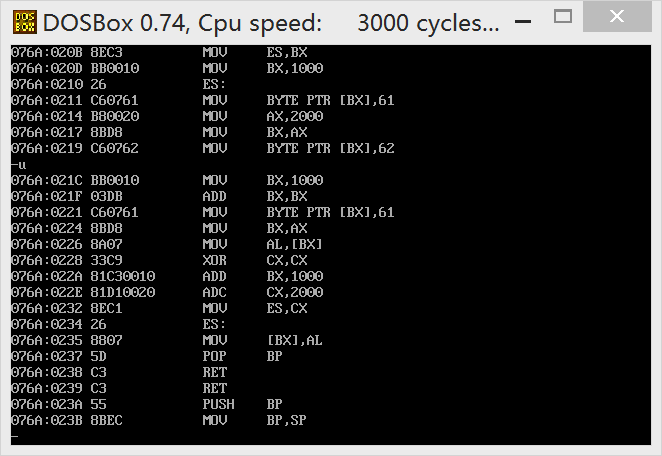
**综合研究三研究报告**

1. **研究过程展示**

1>.um1部分：在综合研究一的基础上。如图所示：

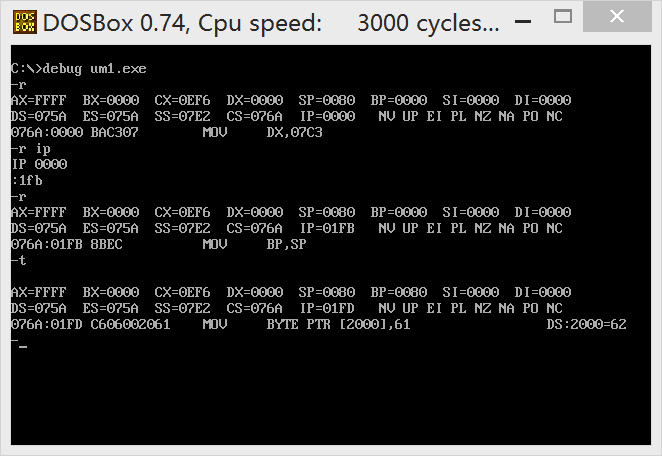
![}]($]E8_MTAJ~L8J@CVX3$K](data:image/png;base64,)

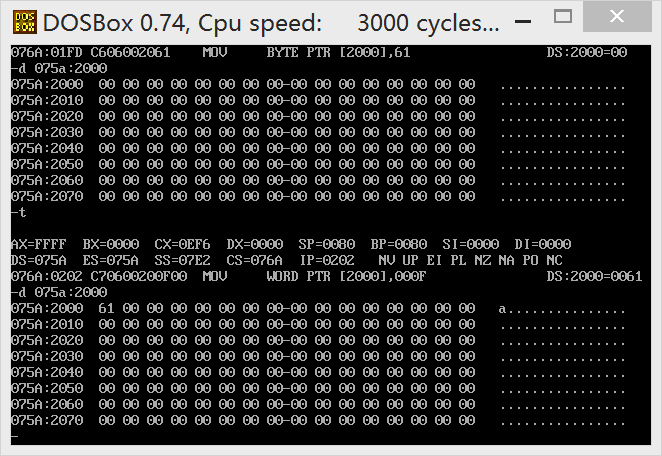




开始进行单步跟踪，并查看相关内存单元的内容：

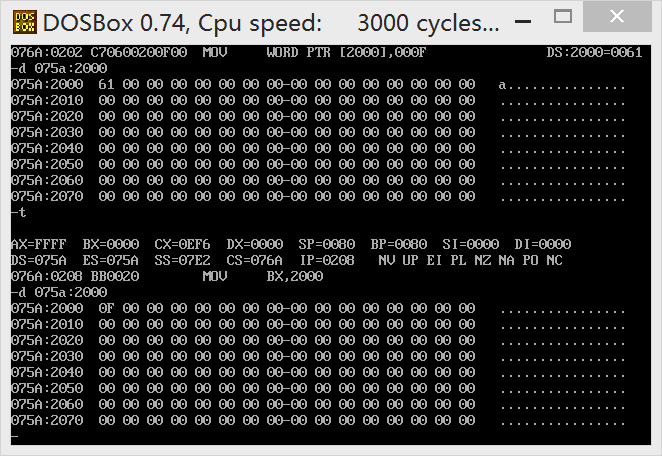
\*(char \*)0x2000=’a’; 执行前后比较：





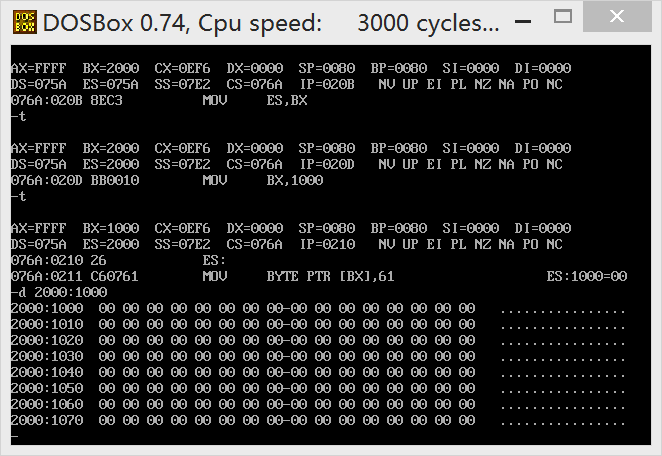
；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；

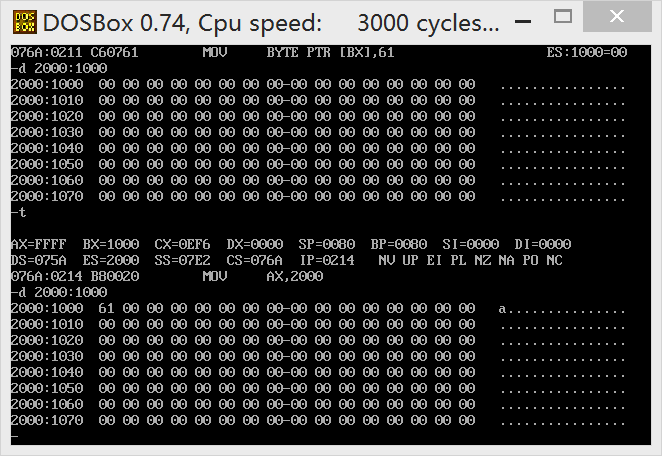
\*(int \*)0x2000=0xf; 执行前后比较：



；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；；

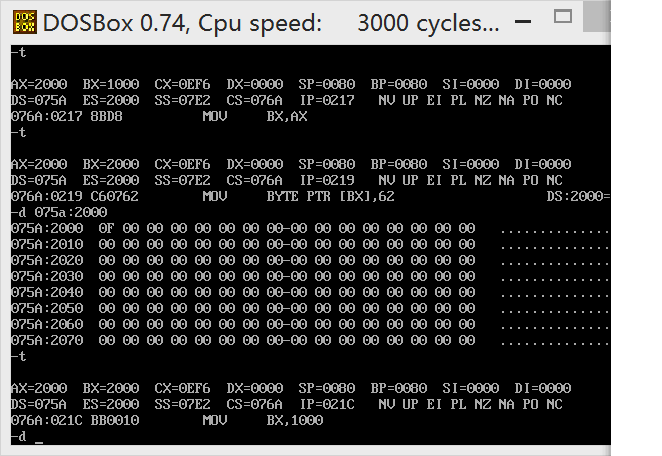
\*(char far \*)0x20001000=’a’; 执行前后比较：

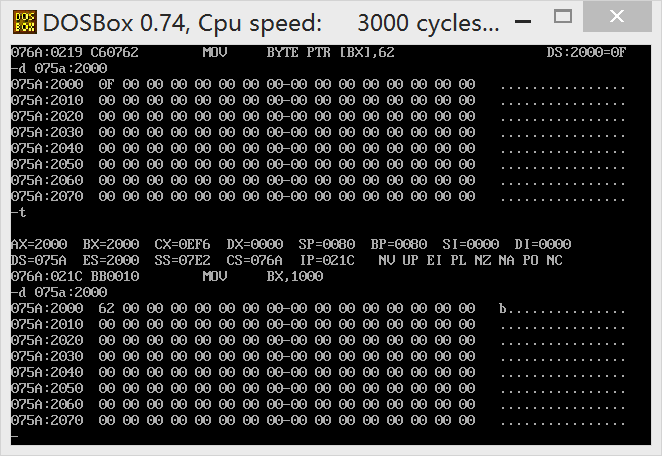




\_AX=0x2000;

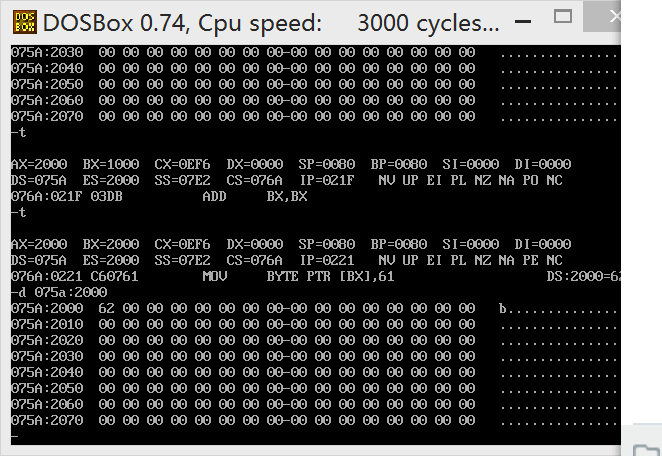
\*(char　\*) \_AX=’b’; 执行前后比较：

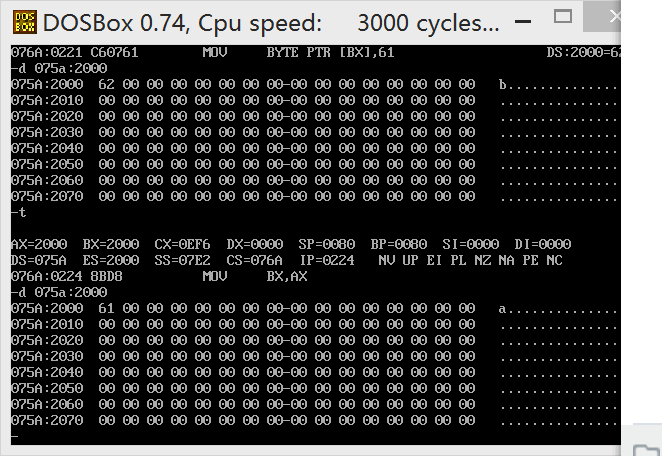




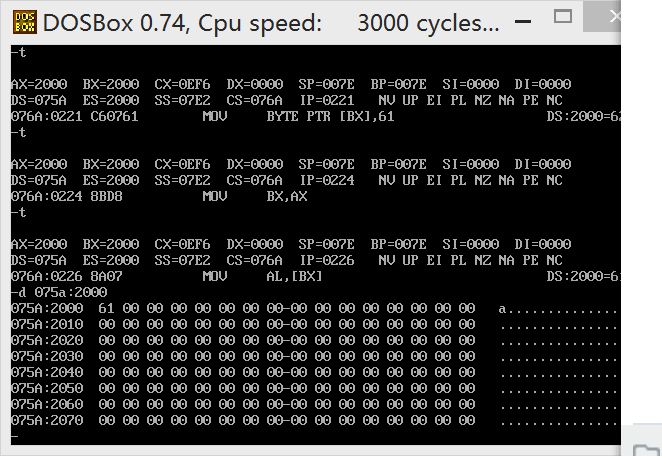
\_BX=0x1000;

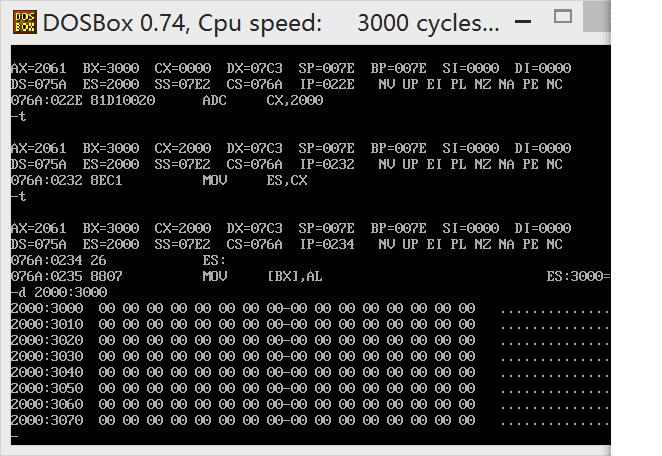
\*(char \*)(\_BX+\_BX)=’a’; 执行前后比较：

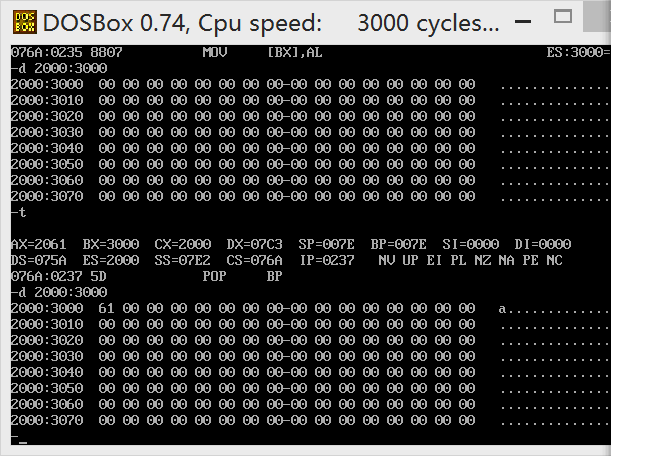




\*(char far \*)(0x20001000+\_BX)=\*(char \*)\_AX; 执行前后比较：



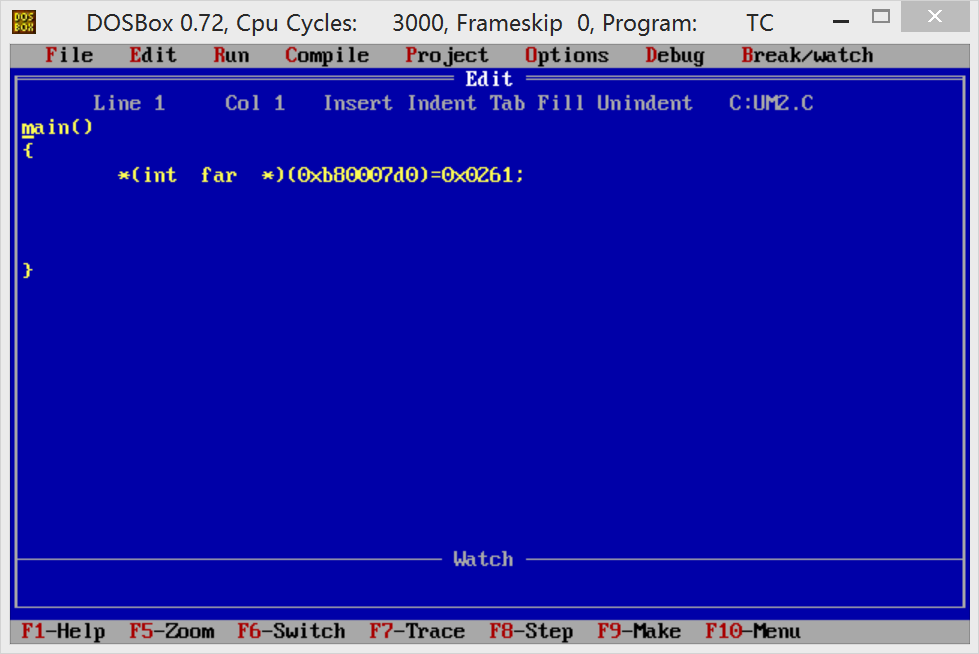




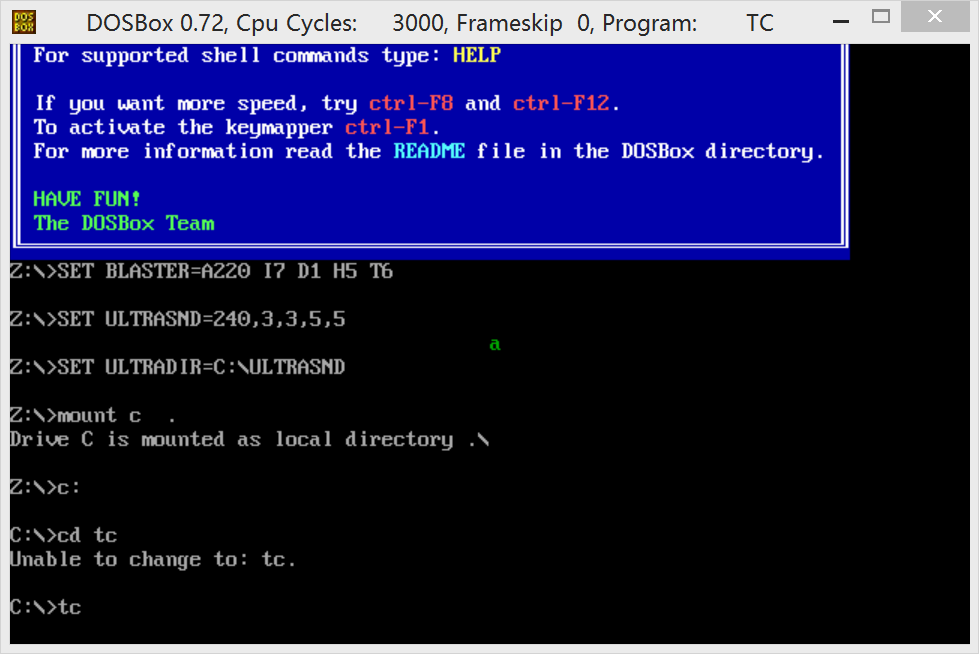
2>.um2部分：

由于char型是一个字节，而只用一条语句得用至少两个字节，所以这里可以用int型。如图：

程序：

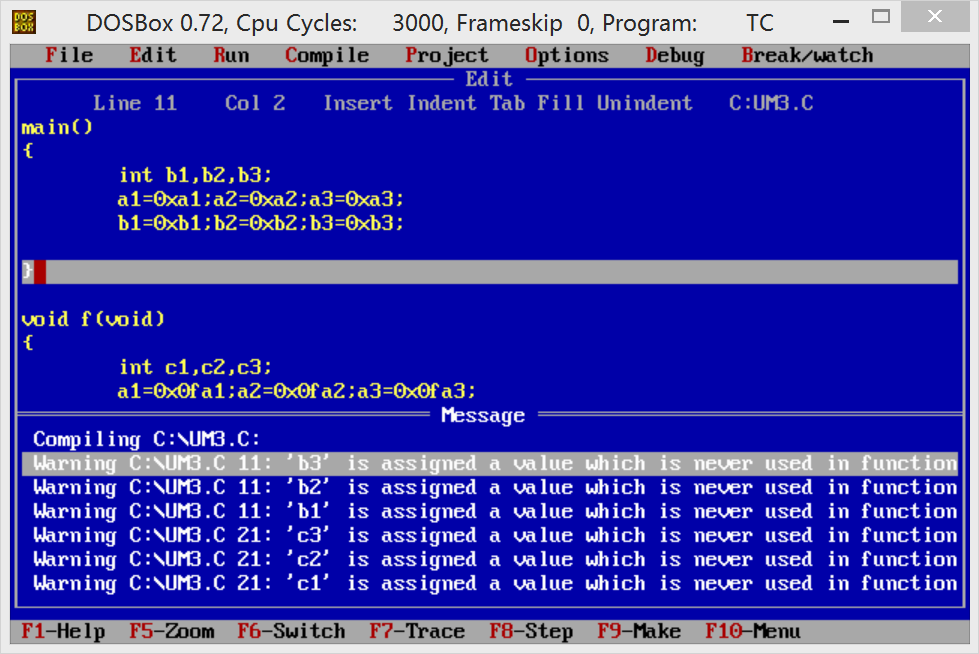


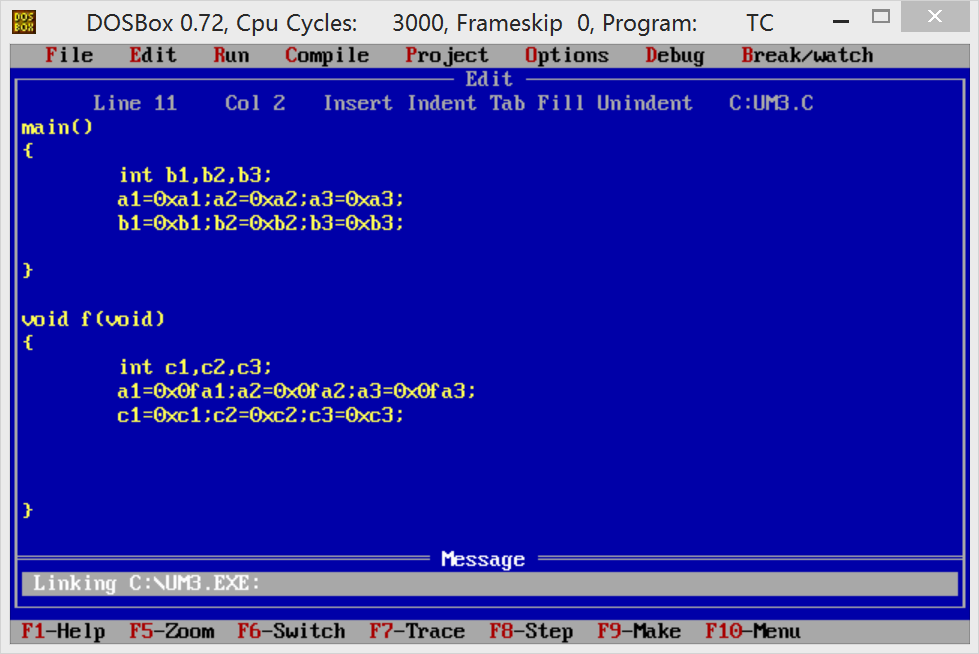
效果：



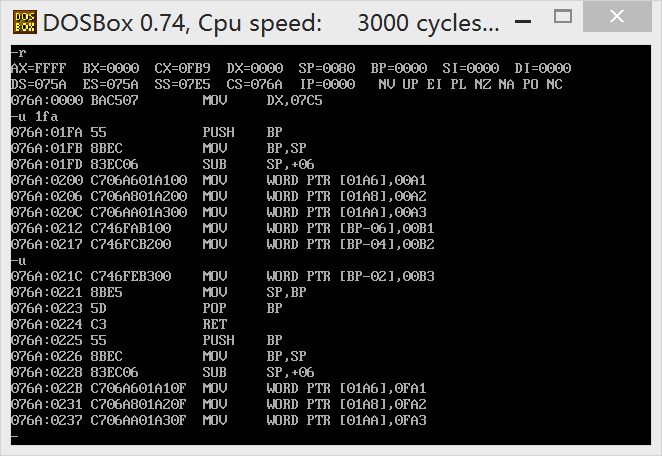
3>.um3部分：

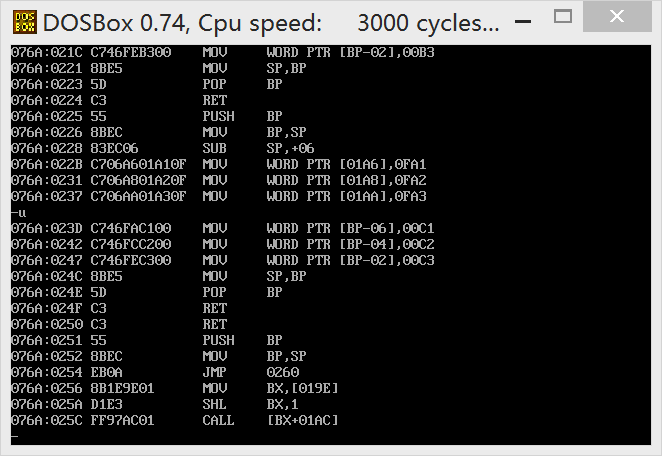
编译时有许多警告，最终可以连接成um3.exe。





Debug：



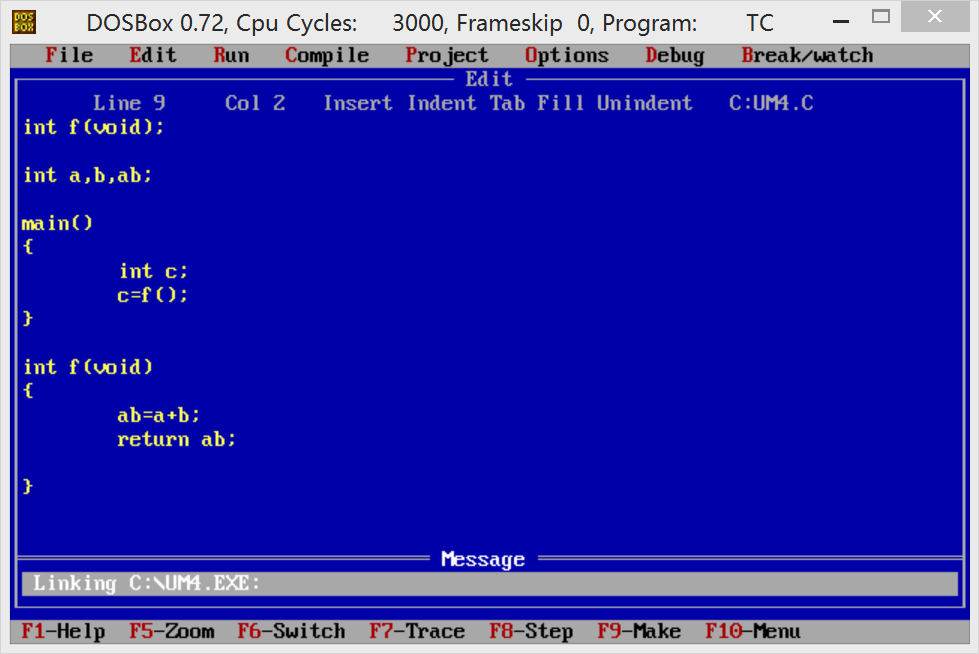


反汇编可看出全局变量a1的地址为01a6，a2的为01a8，a3的为01aa，局部变量b1、b2、b3的地址是由bp-6、bp-4、bp-2表示，而在反汇编程序开始处有将sp的赋值给bp的语句**“mov bp，sp ”**，sp自减了6，因为sp存放的是栈的栈顶偏移地址，可知实际上程序是将b1、b2、b3依次放入了栈中。但是在程序结尾处有**“mov bp，sp”**，把栈顶指针又还原了，这时b1、b2、b3的内存空间已不在栈中，随时可以被其他程序的数据覆盖。

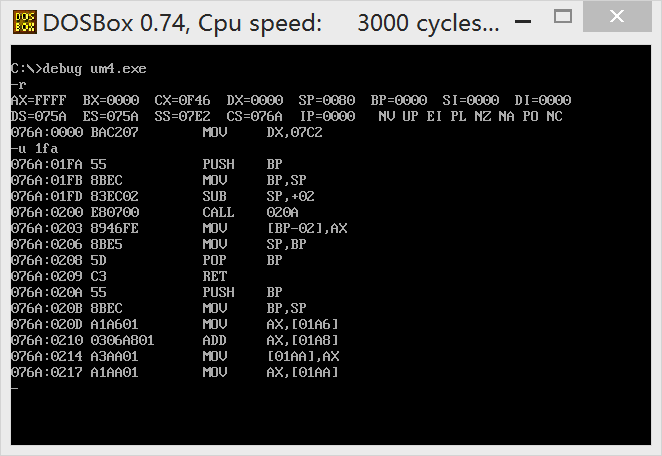
可以看出，全局变量是存放在内存空间的指定内存地址处，而局部变量是存放在栈中。

因为要用bp记录sp原来的位置，以便函数结束时将sp还原，所以要对bp进行保护，所以会有push bp;mov bp,sp。

4>.um4部分：



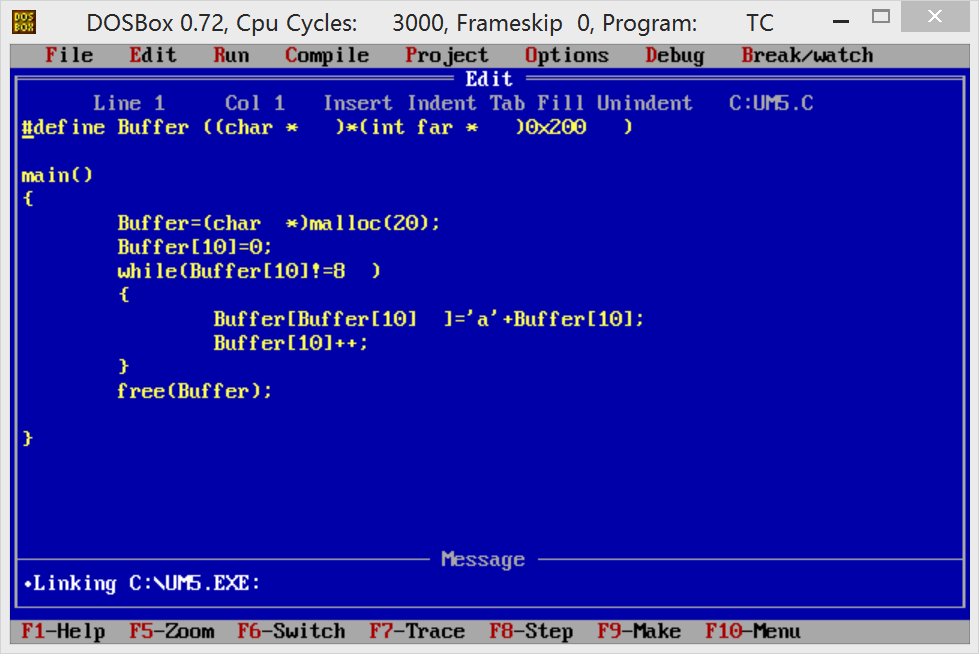
用debug 的u命令反汇编一下，要查看返回值存放在哪里，主要看下图中的一条关键语句就可以了。



即“mov [BP-2],AX”,可知这个程序中将返回值存放在寄存器AX中。

但是这又是否是特例呢？还是这是通用的做法？既然涉及到值传递，这是否又与参数的传递有关呢，或许对了解综合研究五的printf()有关。

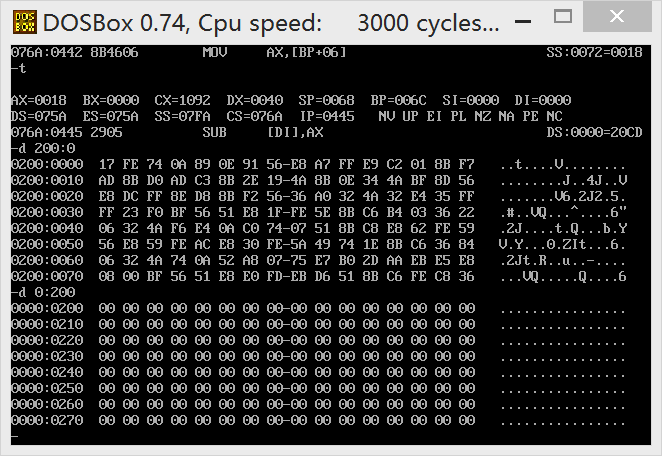
5>.um5部分：



这里两处内存地址都没有改变，需要再验证。

#define是宏定义，在程序中用Buffer代替((char \*)\*(int far \*)0x02000000)

malloc（20）是开辟20个字节的内存空间。

****

1. **已思考研究并已解决问题汇总**

**1>.** **C语言将全局变量存放在哪里？将局部变量存放在哪里？每个函数开头的“push bp；mov bp，sp。有何含义？”**

全局变量是存放在内存空间的指定内存地址处，而局部变量是存放在栈中。

因为要用bp记录sp原来的位置，以便函数结束时将sp还原，所以要对bp进行保护，所以会有push bp;mov bp,sp。

**2>.** **C语言将函数的返回值放在哪里？**

由“mov [BP-2],AX”,可知这个程序中将返回值存放在寄存器AX中。

1. **已思考研究并未解决问题汇总**

**1>.**“mov [BP-2],AX”,可知这个程序中将返回值存放在寄存器AX中。

**但是这又是否是特例呢？还是这是通用的做法？既然涉及到值传递，这是否又与参数的传递有关呢，或许对了解综合研究五的printf()有关**。

**2>.函数间一个参数是如何传递的，多个参数呢？有没有总结出的好的规律？**

1. **研究感想（心得体会）**

有时候自己以为对一件东西了解了，可是真正再次接触的时候，又会让人觉得“这绝对是个新玩意儿，我不认识它”。当我们学了汇编，再学C语言，反汇编过来的汇编语言，结构还是原来的结构，但是套路就是“C”的套路了，之前编写汇编代码，自己有自己编写方式，但是现在不同了，要接触C，现在的学习方法是猜”C”这个人编写汇编时候的风格方式是如何的，然后再亲自看“C”这个人编写的汇编代码实际到底是咋样的，而且，这个“C”绝对是我的老师，比我要对的多得多，我还得抱着敬佩加好好学习的态度。真的是不服不行啊。