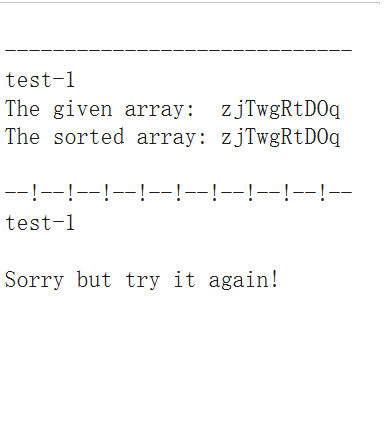
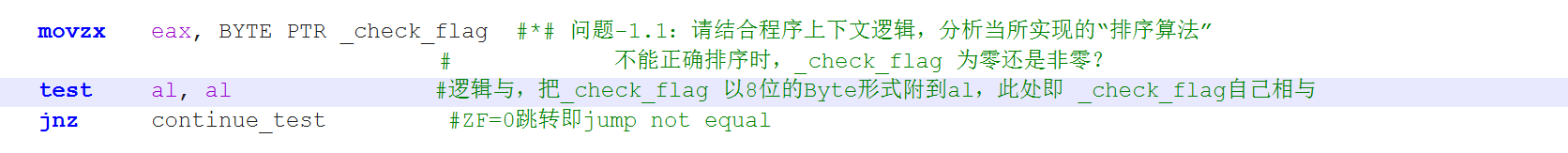
问题一：

首先在程序原始版本正常运作时是不能正确排序的：



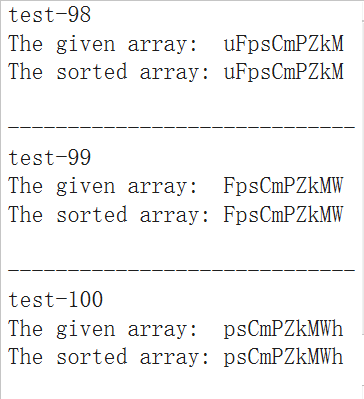
查找程序中的代码可知：



是此段代码的\_check\_flag影响是否正确排序的部分。具体原理由图上可知。

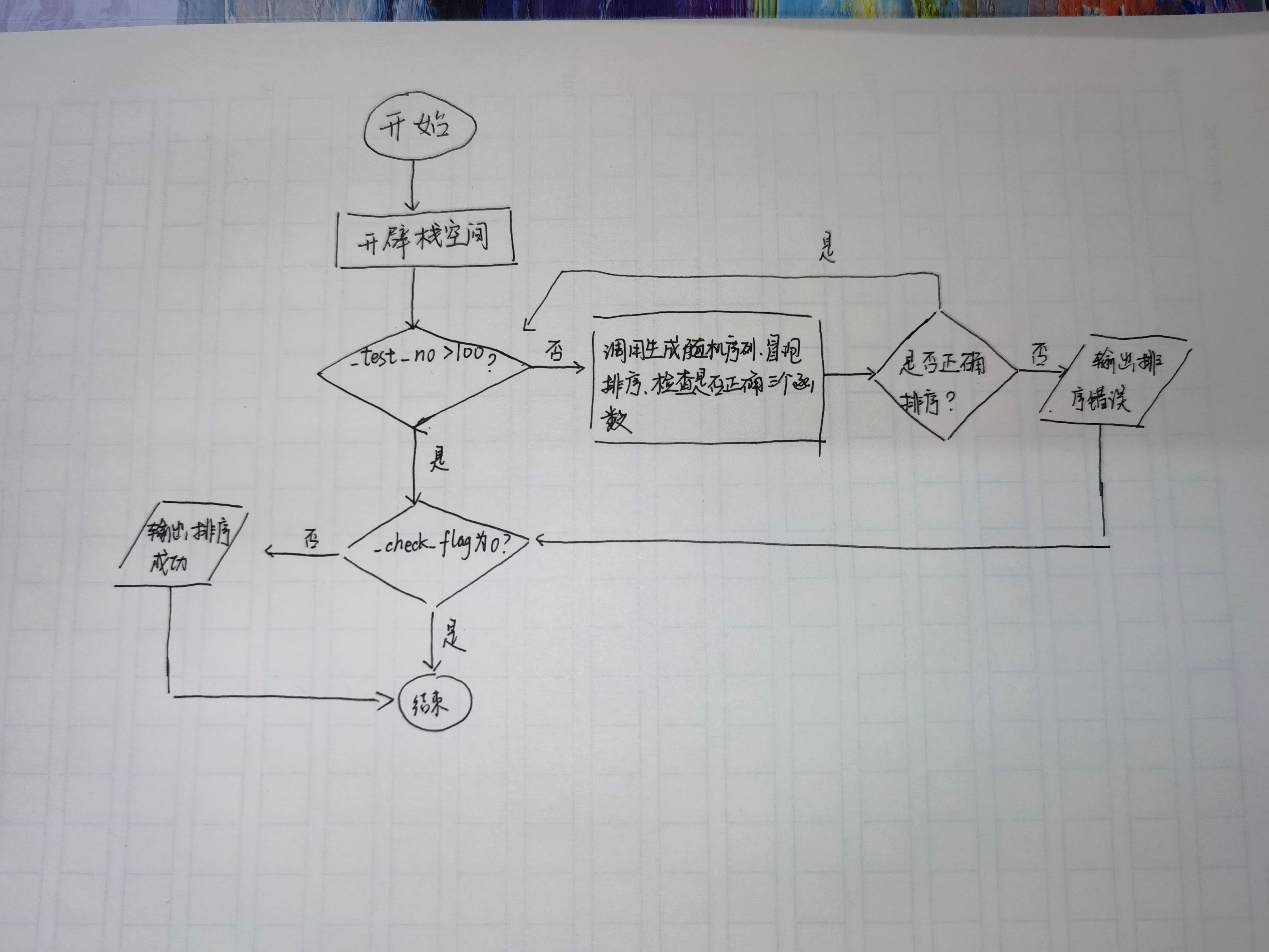
直接把jnz 改为无条件跳转jmp：





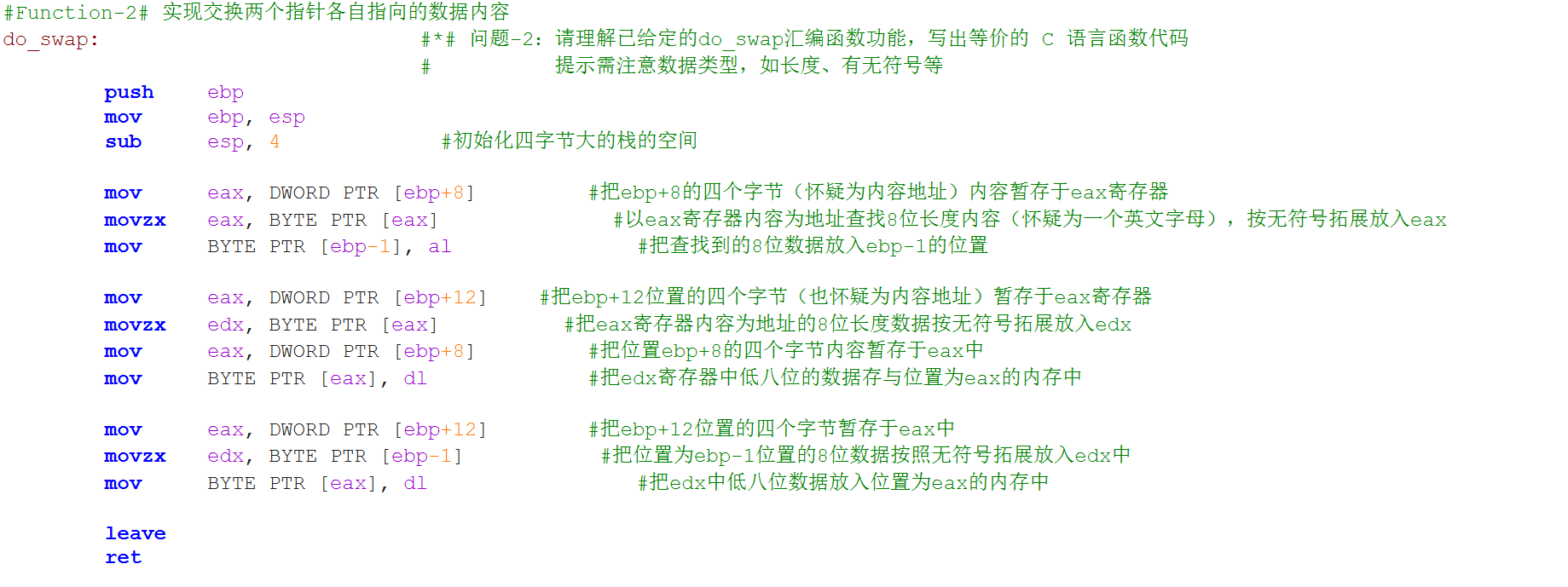
可以发现结果完全不同，可以推测出当排序结果不正确时jnz没有跳转，即ZF=1，即test后结果为0，那\_check\_flag必须为0，所以可知不能正确排序时，\_check\_flag为0。

问题1.2：

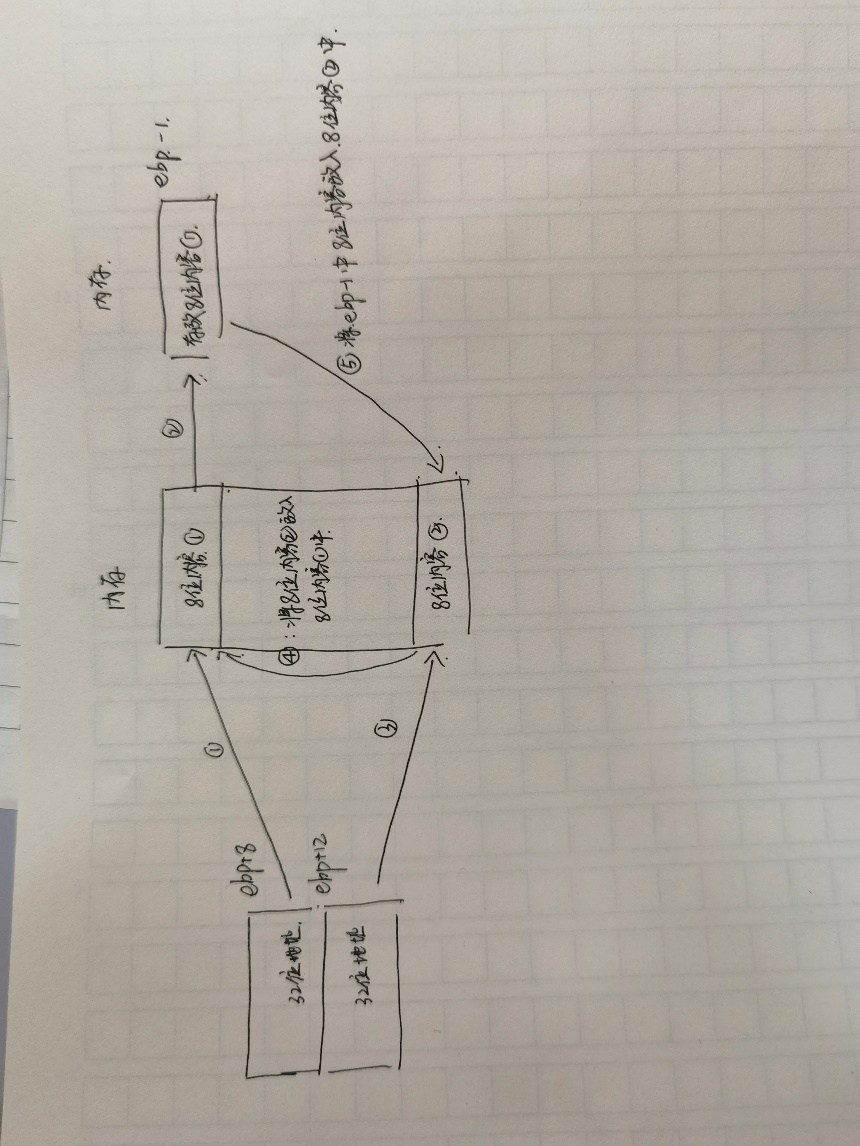


问题2：

首先根据汇编知识写出大致内容注释：



能想到是一个以间接寻址形式的内容调换，下面大致画出简图：



则大概可以写出C语言代码为：

void do\_swap(unsigned char \*a) //\*a作为ebp+8对应的32位数据

{

unsigned char c = \*a; //即ebp-1位置内暂存ebp+8地址对应的8位数据

\*a=\*(a+1); //即用ebp+12地址对应的数据取代ebp+8地址对应的数据

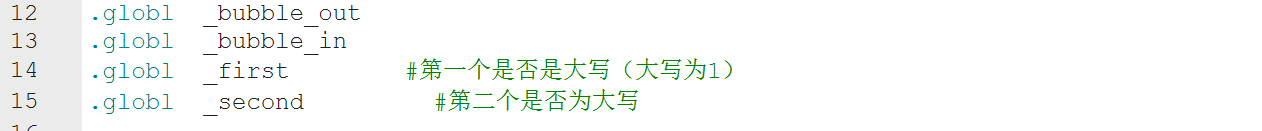
\*(a+1)=c; //即最后用ebp-1位置的数据再代换ebp+12位置对应的数据

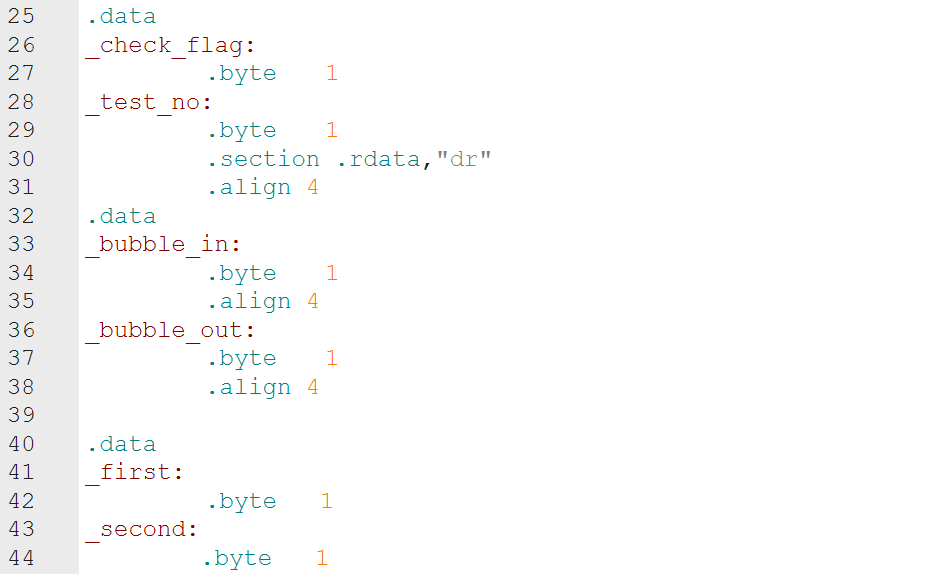
return;

}

问题3：

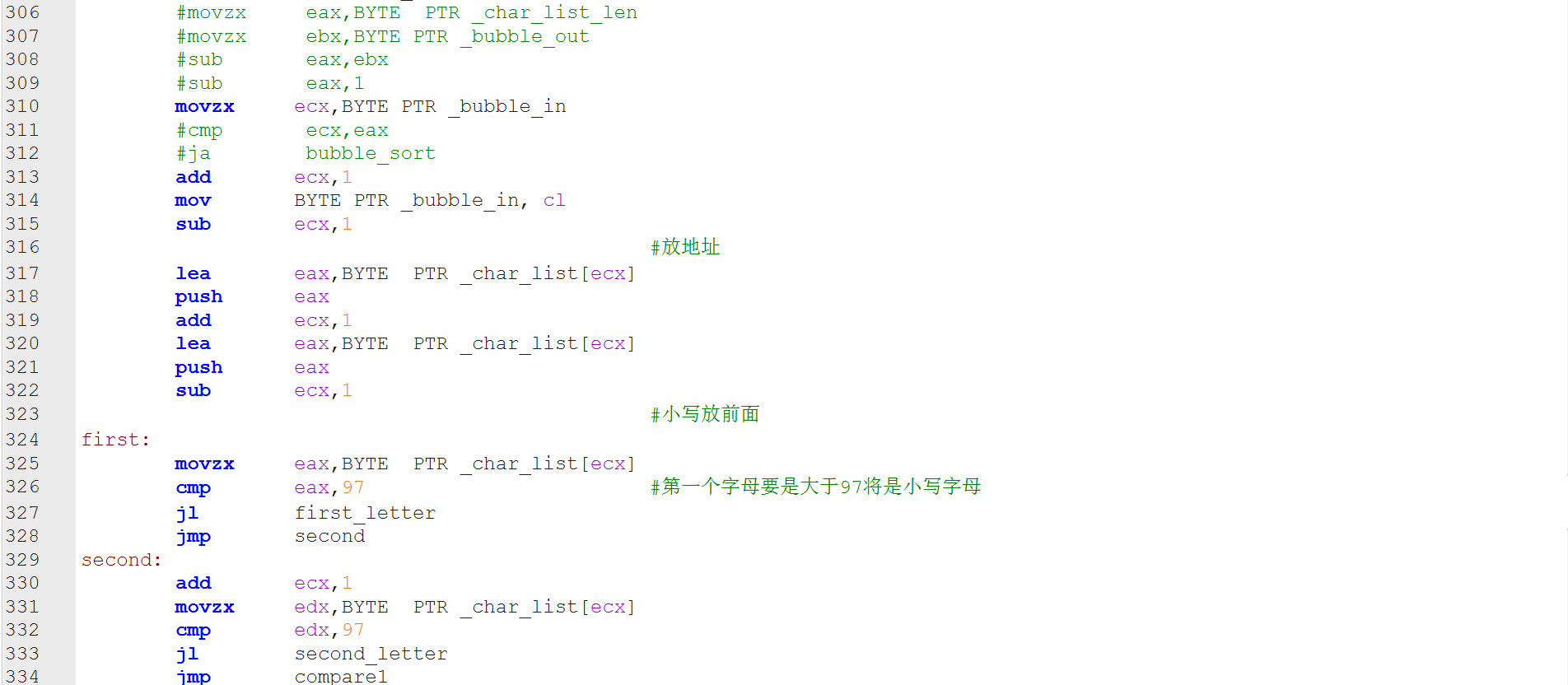
加入了四个全局变量

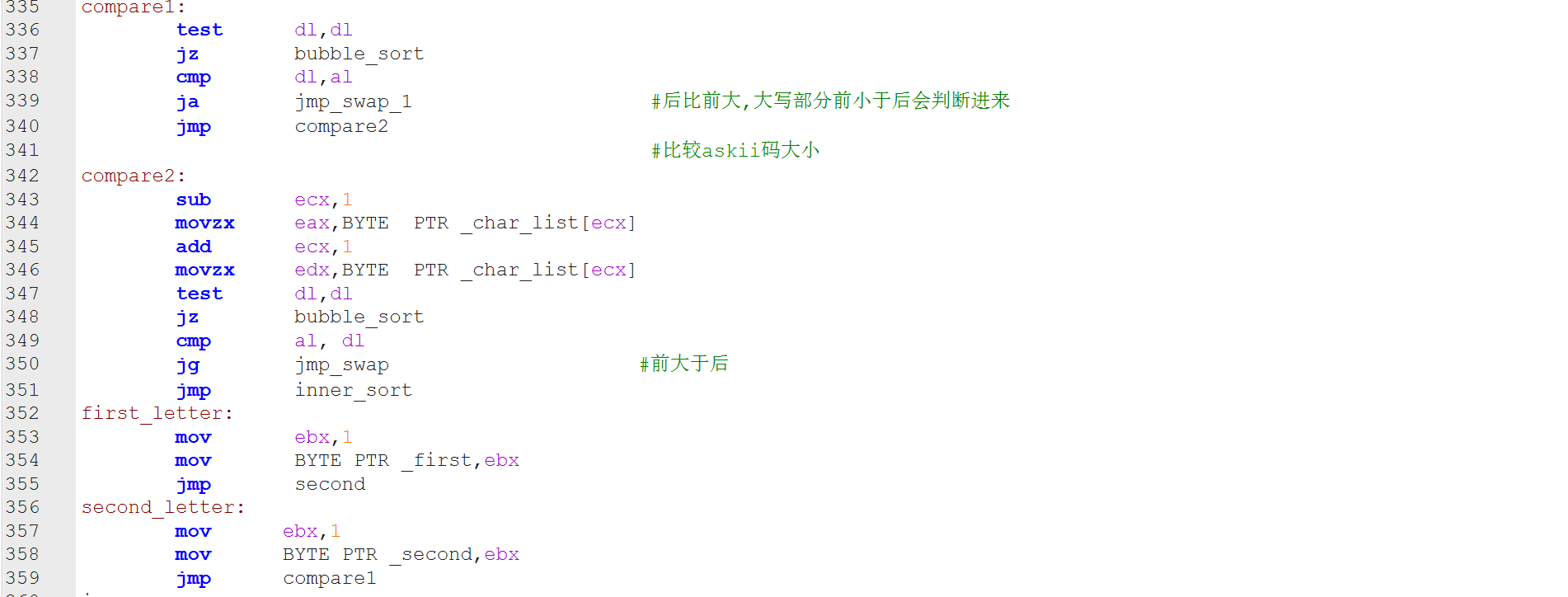


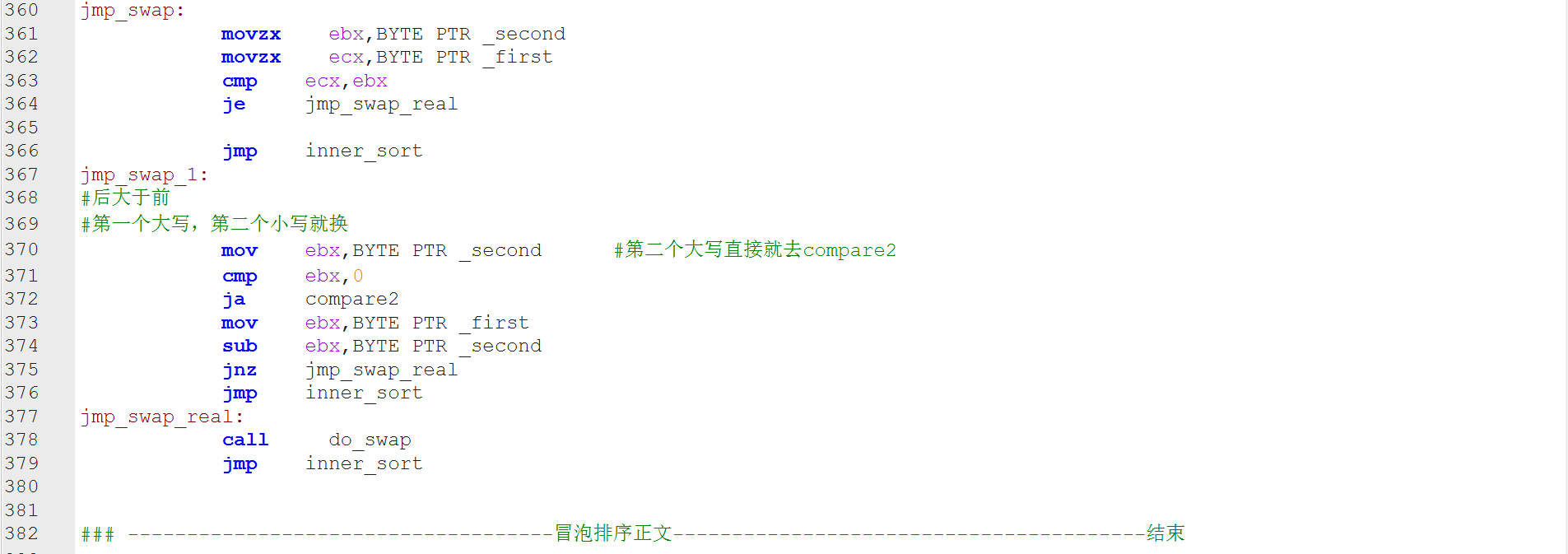


冒泡排序部分如图所示：

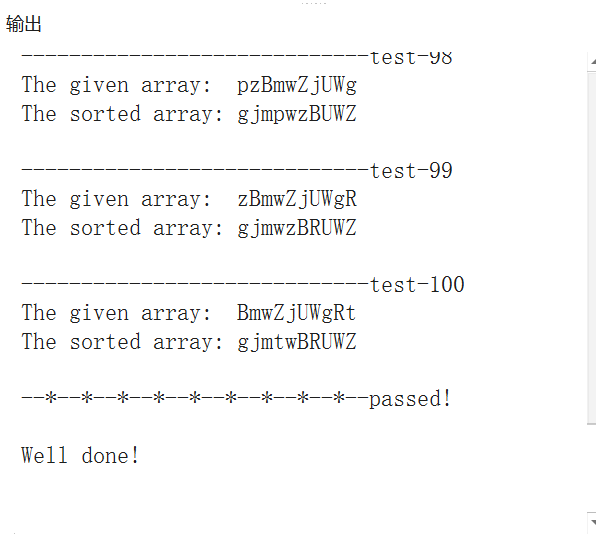








运行结果：



具体修改后的代码在文件中

3.24分割线：

首先第二题C语言代码进行修改：

根据movzx指令发现赋值到寄存器的内容为无符号拓展

则进行调换的内容应为unsigned char类型

void do\_swap(unsigned char \*a) //\*a作为ebp+8对应的32位数据

{

unsigned char c = \*a; //即ebp-1位置内暂存ebp+8地址对应的8位数据

\*a=\*(a+1); //即用ebp+12地址对应的数据取代ebp+8地址对应的数据

\*(a+1)=c; //即最后用ebp-1位置的数据再代换ebp+12位置对应的数据

return;

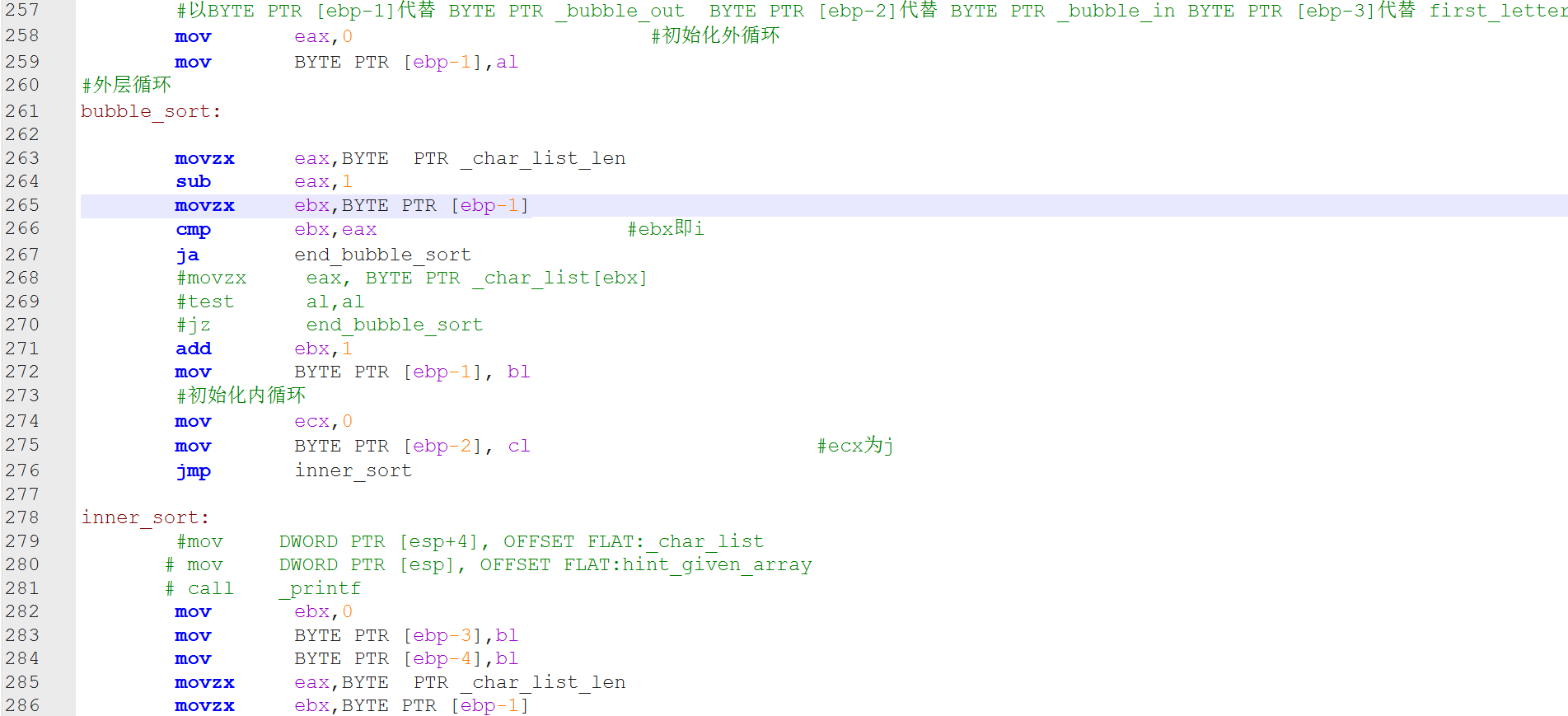
}

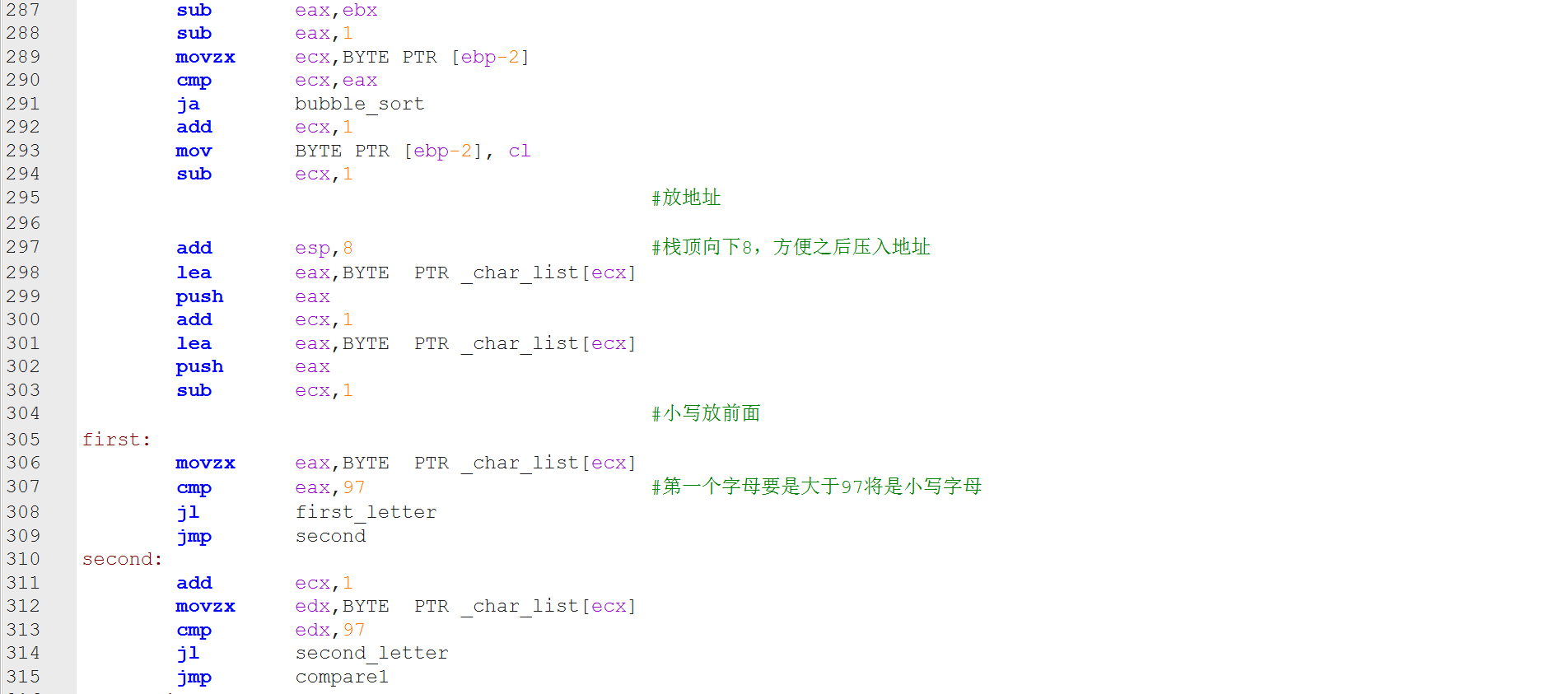
第三题修改：

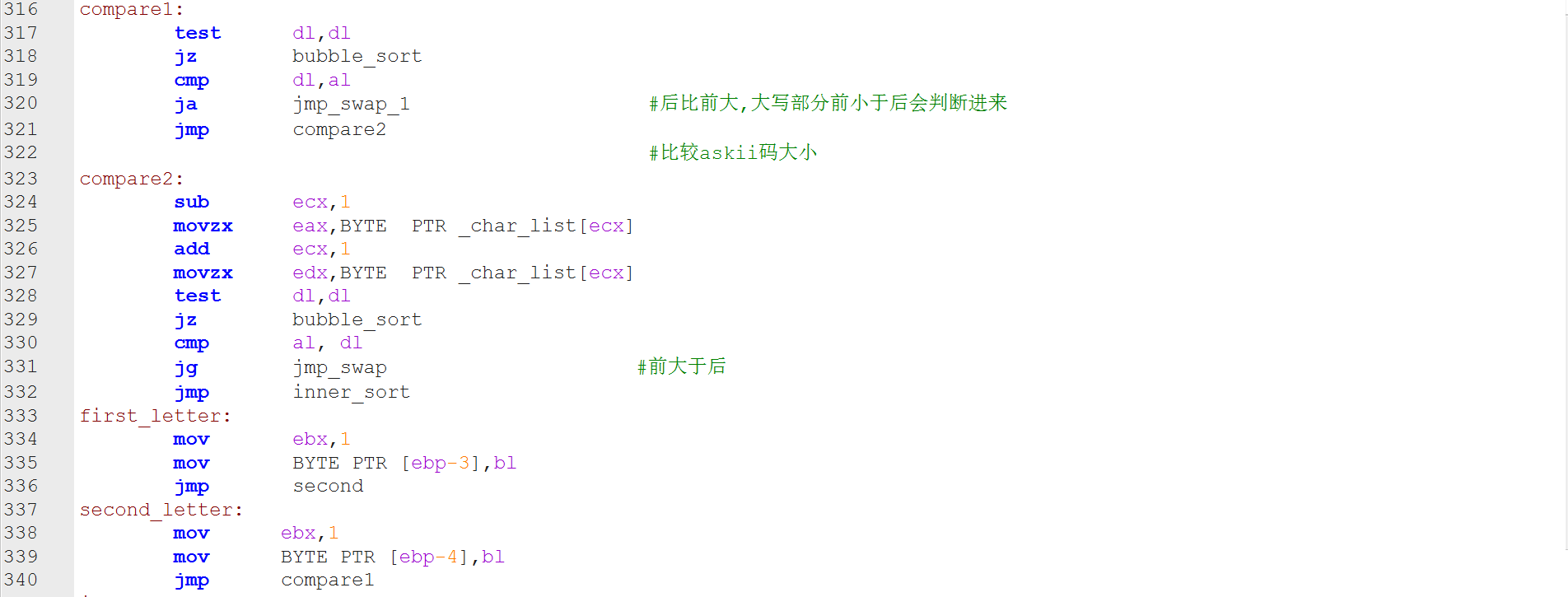
之前版本没有利用bubble sort函数的栈空间，而是利用全局变量实现

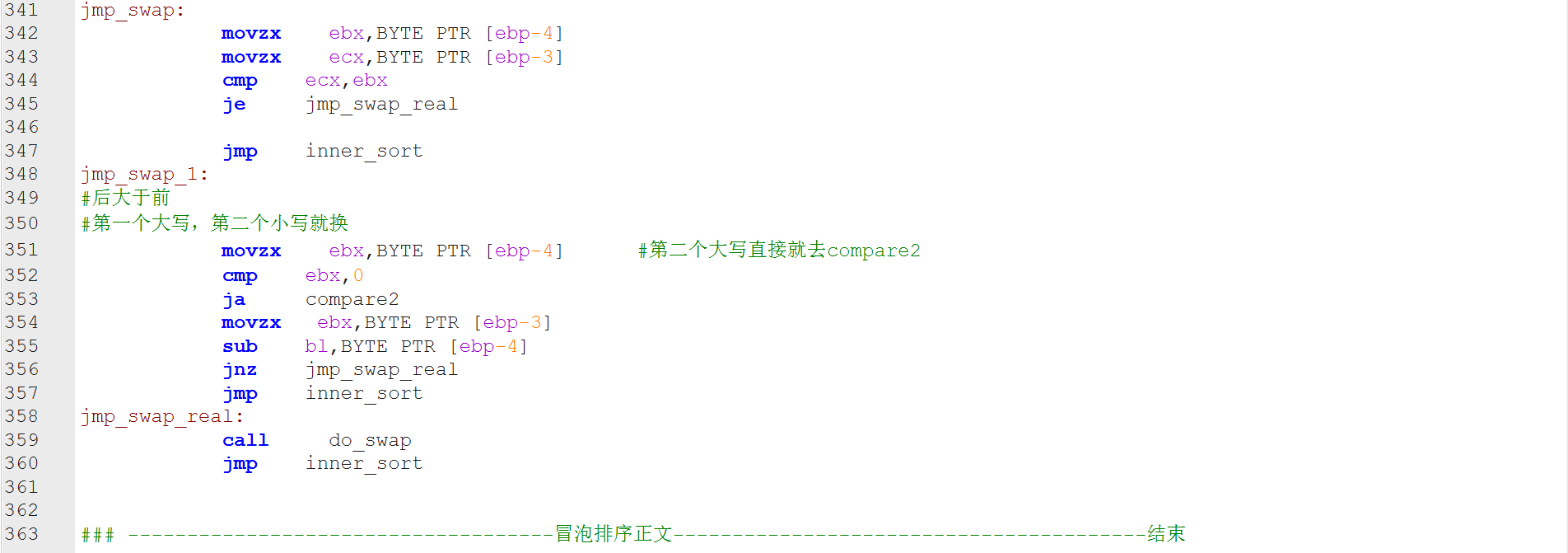
现根据以BYTE PTR [ebp-1]代替 BYTE PTR \_bubble\_out BYTE PTR [ebp-2]代替 BYTE PTR \_bubble\_in BYTE PTR [ebp-3]代替 first\_letter BYTE PTR [ebp-4]代替 second\_letter

的规则，以函数栈空间记录需要的数据：

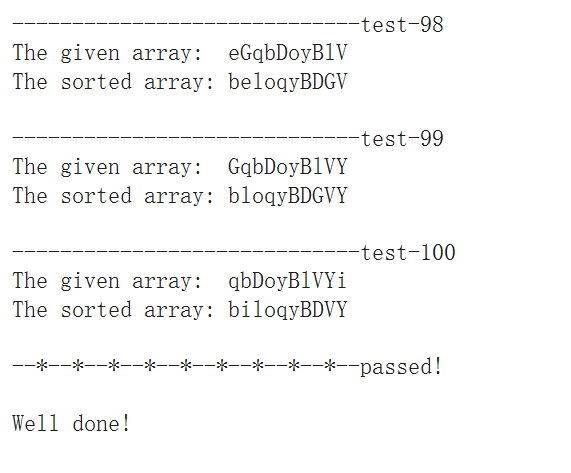








执行结果如图



修改版的程序也在文件中