**c语言综合研究十一报告**

1. **研究过程展示**

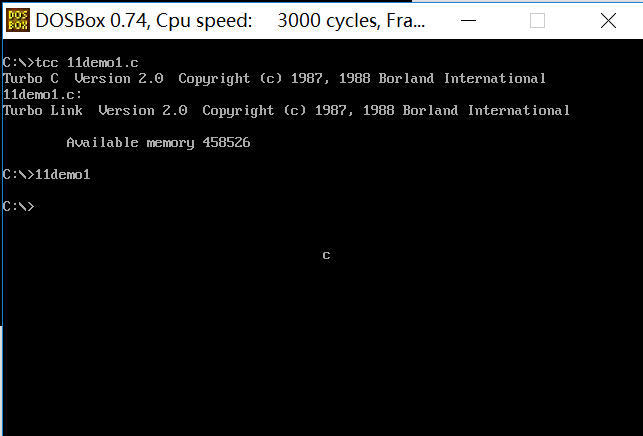
**1>.程序1部分：**

在执行程序main（）里只有一句语句：((void (far \*)())(long)a)();

分析语句：a是数组的首地址，将它强制转换成long型使它的数据包含段地址和偏移地址的数据，但此时它还不是一个地址而是一个long型变量，那么我们将它再强制类型转换成一个void（far \*）（）型的函数指针，它是一个远指针，指向一个void型的函数，所以这个函数的入口地址就是数组a的首地址。即程序是要执行以数组a里的元素构成的语句所组成的函数。那么我们要向数组a里填充的是一段内存空间的数据，这些数据连起来能被翻译成一段语句，这段语句的功能是在屏幕中间打印一个字符“c”。

虽然数组有存放数据的寻常印象，函数有操作数据的寻常印象，但是，这与之前汇编的时候。数据段和代码段可以联系一下，因为之前也编写过在数据段中的数据发挥代码的作用的程序。这里感觉是想要再次让大家回忆一下，代码和数据是没有分别的，同样的，不论是存放在函数里，还是存放在数组里，只要本质是一样的，那么结果也差不离。

运行一下：



程序部分：

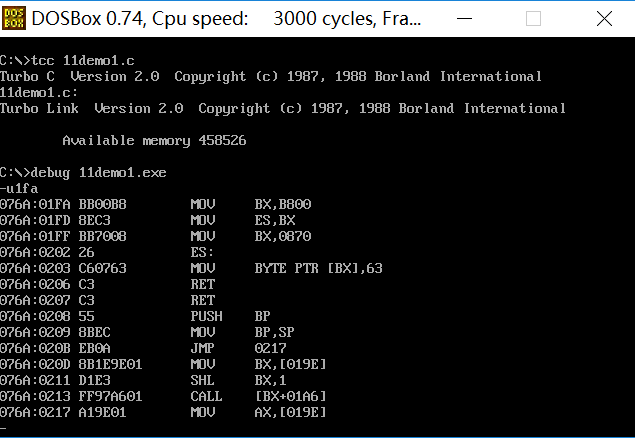
main()

{

\*(char far\*)(0xb8000000+160\*13+80) = 'c';

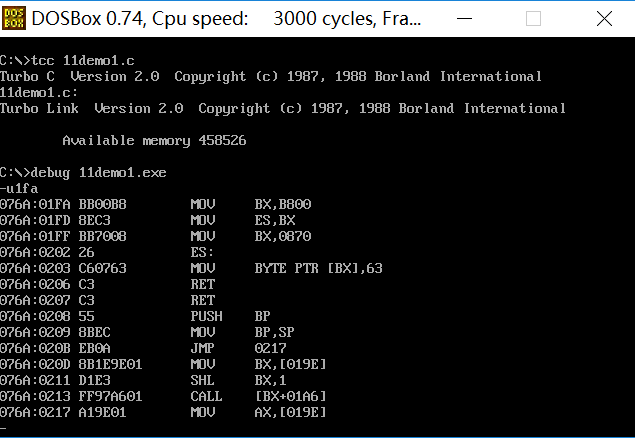
}

Debug 11demo1.exe：

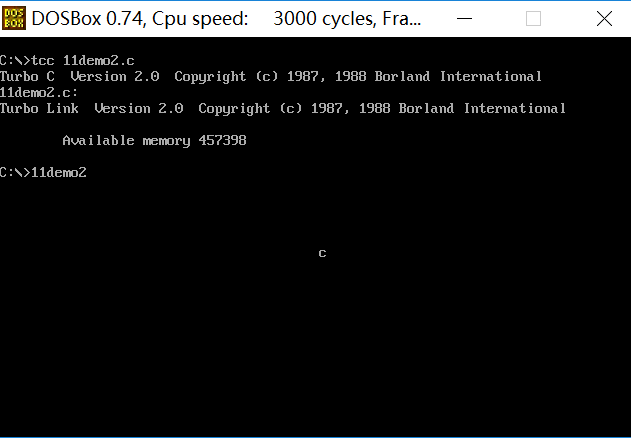


**2>.程序2部分：**

将以下16进制数复制到a数组中看一下：



运行一下：



可以看到，正常运行，但是没有正常返回。

程序部分：

Char a[200] = {0xbb,0x00,0xb8,0x8e,0xc3,0xbb,0x70,0x08,0x26,0xc6,0x07,0x63,0xc3};

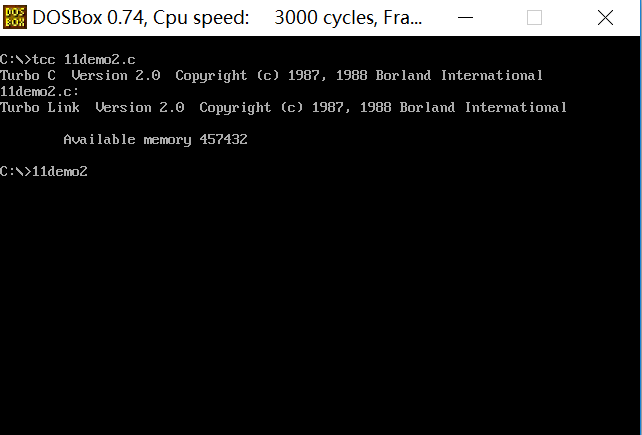
main()

{

( ( void (far \*)() )(long)a )();

}

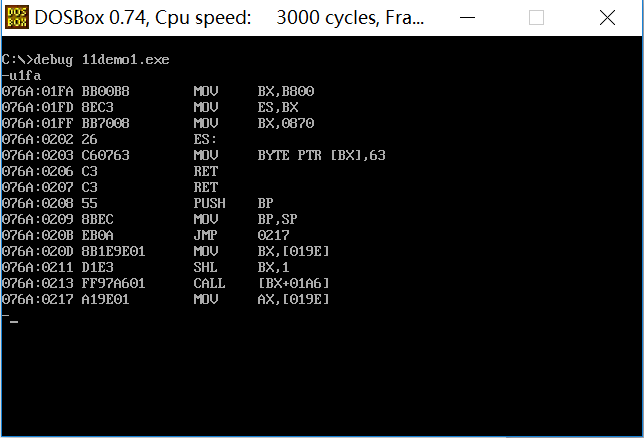
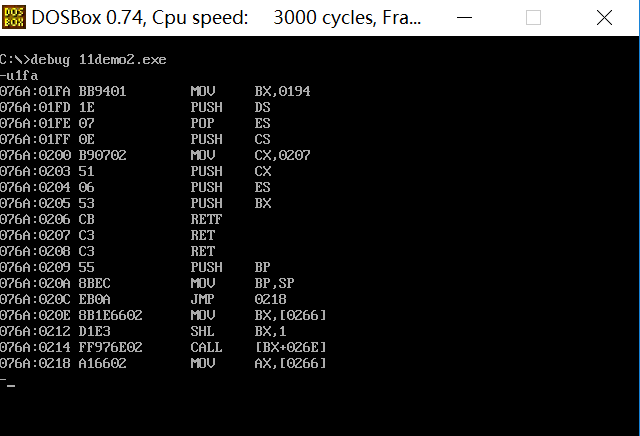
去掉“long”强制转换类型部分，再运行看一下：



结果很明显不行，程序死在那儿了。我们在将一个数组的首地址强制转换成函数指针时，一定要先将它强制转换成long型数据，这样它才能包含段地址和偏移地址，函数指针才能正确指向到函数。别经常用printf（）只打印习惯函数的偏移地址习惯了就轻易地忘记这一点，内存中真正的地址是段地址+偏移地址的组合。

分析部分：

这里既然结果不一样，就先比较一下两者的反汇编代码看一下：

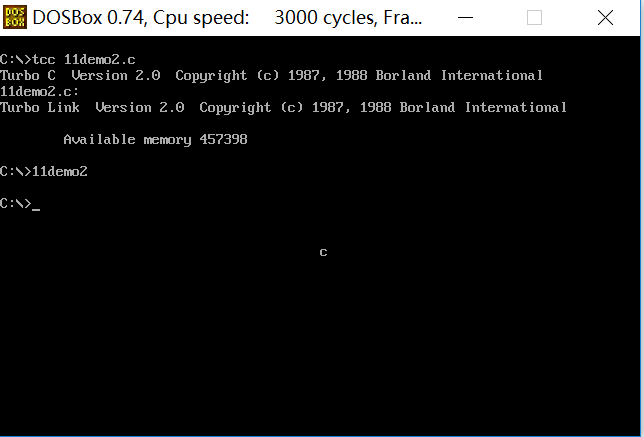


在这里，分析一下11demo2.exe 的反汇编代码。

应该是DS:0194H处存放着数组a的数据，前面将ds值赋给es。将真正的cs，以及执行完“数组里的代码”之后的ip也就是对应的ret的0207h赋给cx，先将cs，cx依次压入栈底，保存起来。然后将es，bx，对应的就是数组a的位置，依次压入栈中，经过retf，来一次“cs = es，ip = bx”的大换血，就实现将数组a中的数据当作代码来执行了。

反汇编的代码好不一样啊，可是实现功能都一样。返回方式也不一样，11demo1是ret,11demo2是retf。

把数组中最后的0xc3修改为0xcb试一下：



竟然正常返回了，为什么会这样？

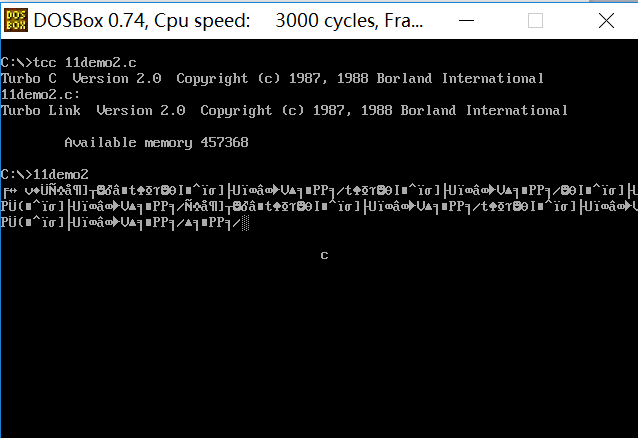
C3 是 ret，pop ip；

Cb 是 retf，pop ip，pop cs；

其实就是一个段内返回，一个段间返回。对应的，说明之前的是段内调用还是段间调用。这里很明显正确的是段间调用，因为数组a是一个全局变量，存储在静态存储区。

把程序改一下，将数组a定义在main函数内，最后那个16进制数，再改回0xc3.

运行一下：



可以看到程序打印出了字符“c”。但是，此处不仅有乱码，程序还死在那儿了。

很不对。

程序部分：

main()

{

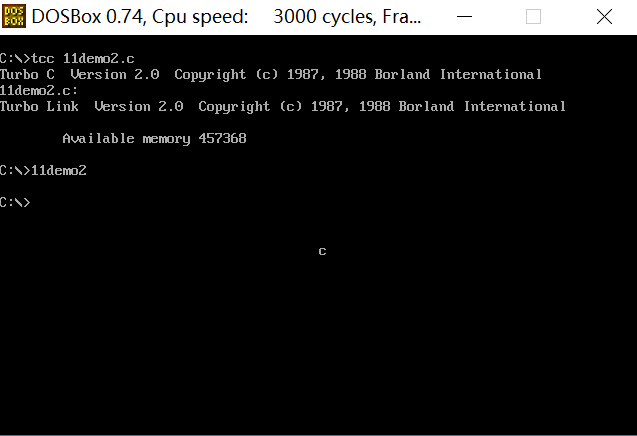
char a[200] = {0xbb,0x00,0xb8,0x8e,0xc3,0xbb,0x70,0x08,0x26,0xc6,0x07,0x63,0xc3};

( ( void (far \*)() )(long)a )();

}

只好再将最后的“0xc3”改为“0xcb”试一下。

运行一下：



程序竟然又正常了。

程序部分：

main()

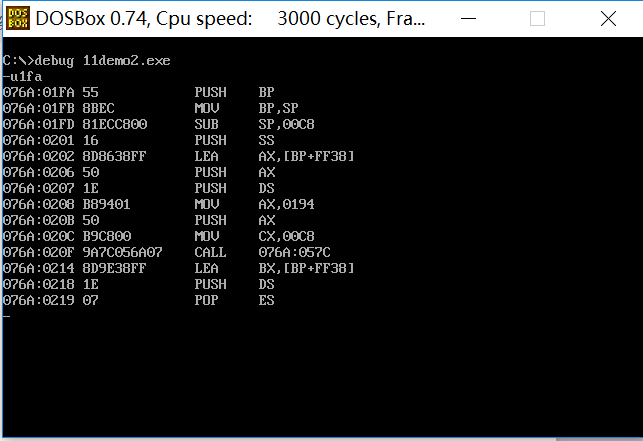
{

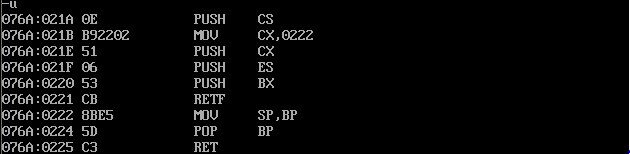
char a[200] = {0xbb,0x00,0xb8,0x8e,0xc3,0xbb,0x70,0x08,0x26,0xc6,0x07,0x63,0xcb};

( ( void (far \*)() )(long)a )();

}

Debug看看：





这里，数组a存放在栈段中，调用的时候也是段间调用，返回对应段间返回。

1. **已思考研究并已解决问题汇总**

不同的位置，同样的返回问题？

看返回问题，可以联系与之相应的调用。但是具体情况也要具体分析。写程序要转换程序和数据时一定要注意这个程序的位置，它的数据能否直接移植到其他程序里面使用。

1. **已思考研究并未解决问题汇总**
2. 用其他的方法实现又会遇到怎样的问题？
3. **研究感想（心得体会）**

很多情况需要具体情况具体分析，但是有些情况的“经验主义”式地“举一反三”可以帮助我们更快地贴近“具体情况具体分析”。算是个方法吧。