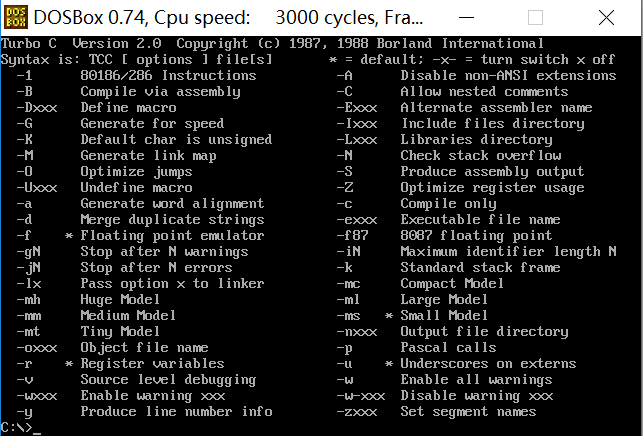
**c语言研究报告四**

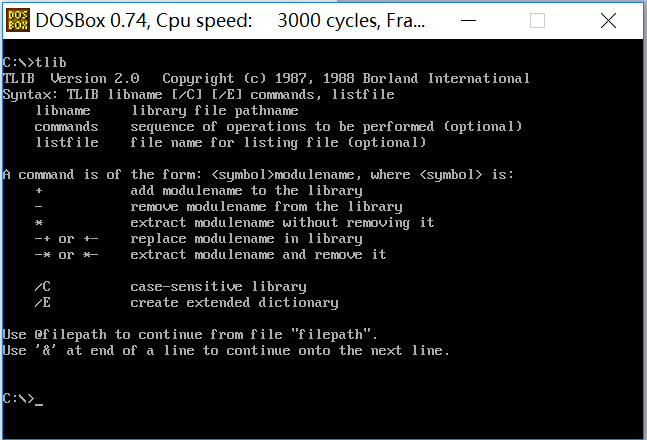
1. **研究过程展示**

**1>.程序1部分：**

Tcc 选项：

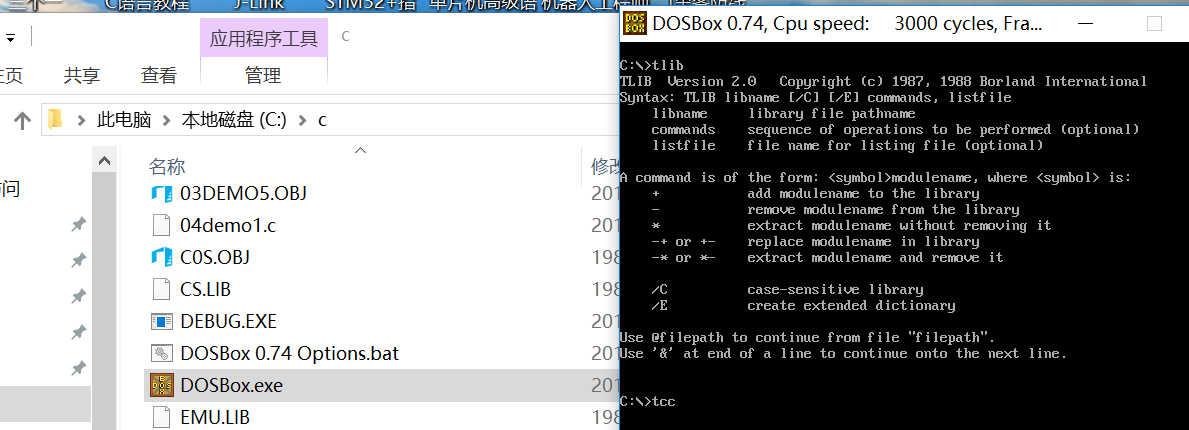


照猫画虎，看一下tlib：



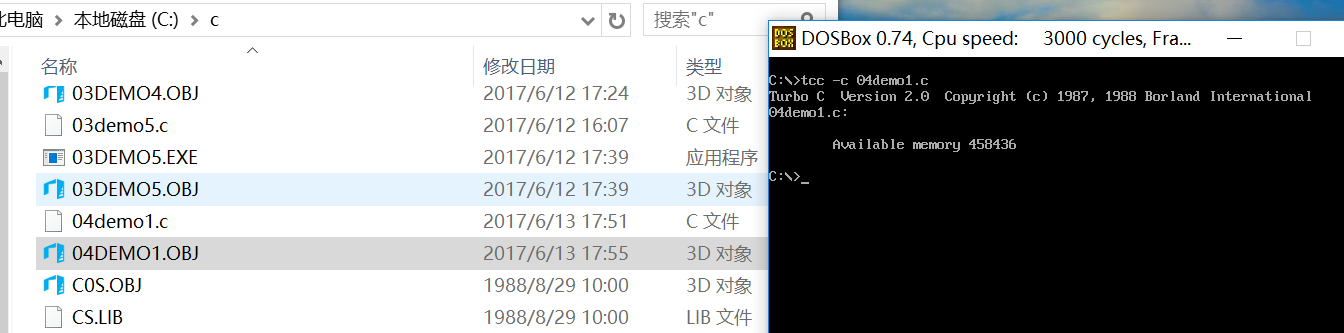
用tcc编译前后对比：

Tcc前：



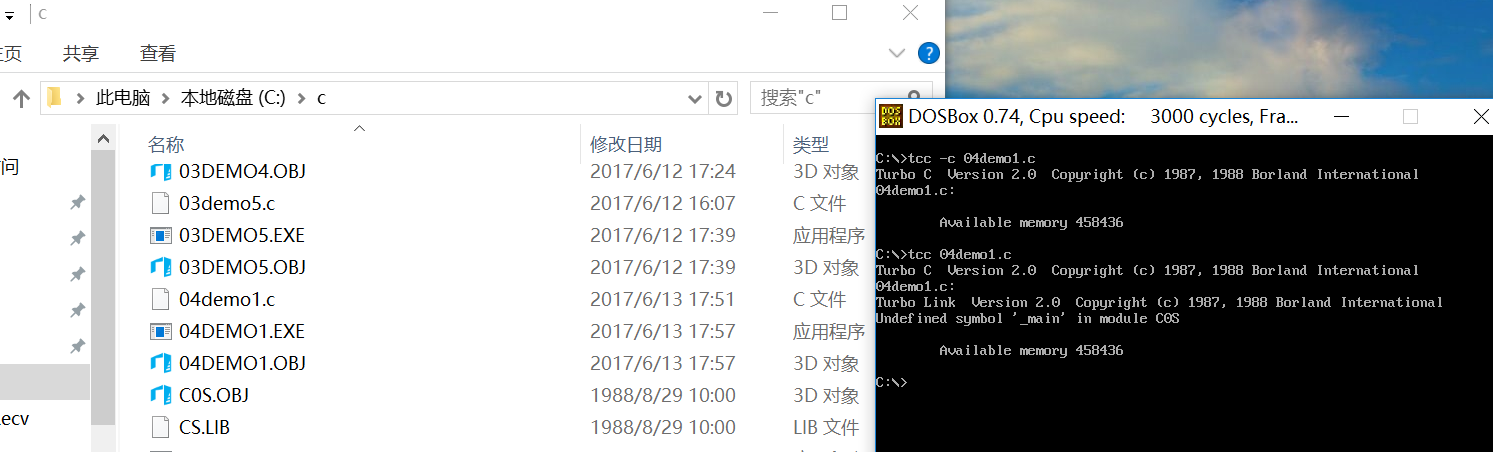
在tcc选型表中看到“-c”选项是“compile only”，试验一下：

Tcc -c 04demo1.c:



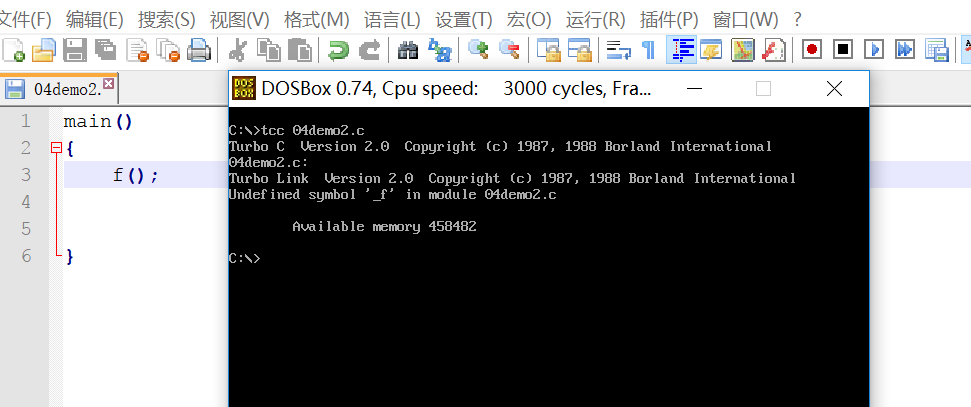
可以看到，只生成了04demo1.obj 文件，并没有像之前的 tcc 04demo1.c的时候，\*.obj与\*.exe都生成的情况发生。

此处为了对比，删去生成的04demo1.obj文件，看一下 tcc 04demo1.c：



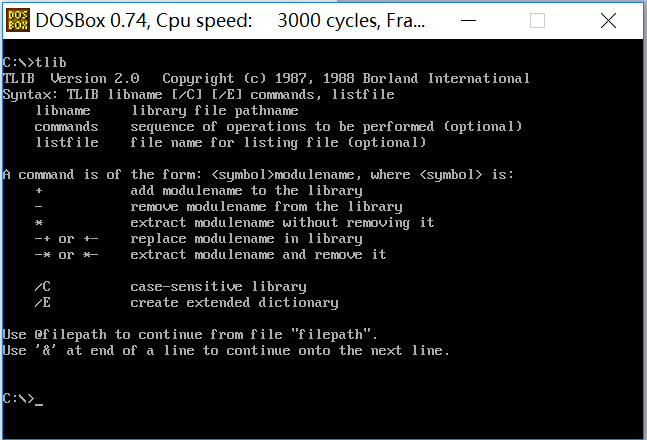
**2>.程序2部分：**

先看一下结果：

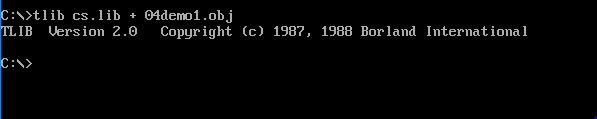
这里，出现“模块\_f未定义”提示，\_f其实就是f，只不过在这个过程中编译器对自己定义的函数名会有自己的操作。也就是f()未定义。

**3>.程序3部分：**

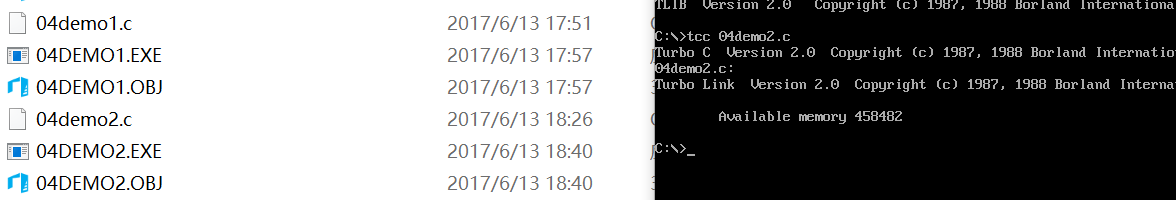
Tlib选项：



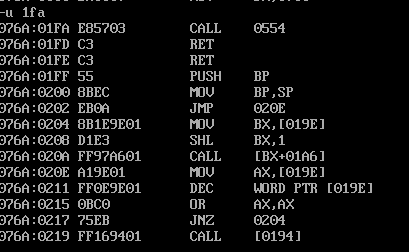
将04demo1.obj文件加入到当前目录的cs.lib 文件中：



再次tcc 04demo2.c文件，前提是先删去原来在第二步生成的04demo2.obj文件：



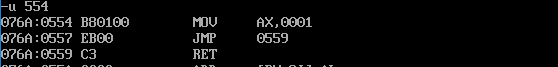
Debug 一下：



其中，main函数是：



f函数是：

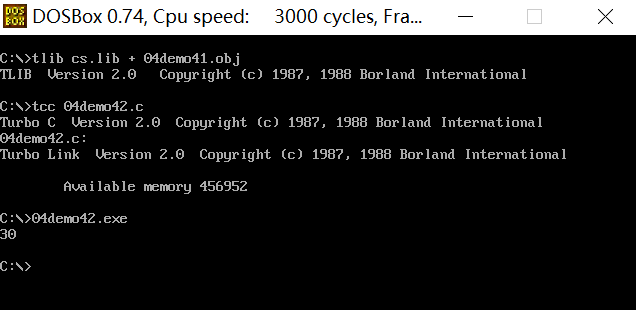


成功生成了04demo2.obj文件与04demo2.exe文件。

但是我们的f()的相关定义是放在了cs.lib文件中，并没有放在04demo2.c源程序文件中，编译的时候，只会对04demo2.c文件进行相关操作，连接的时候没有报错，说明编译器在链接的时候，如果在本程序文件中没有找到相关的函数定义，会去cs.lib文件中进行查找。说明函数f的代码是在链接的时候加入的。

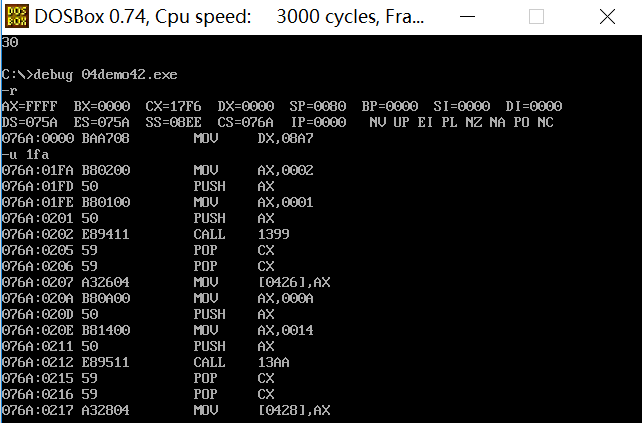
**4>.程序4部分：**

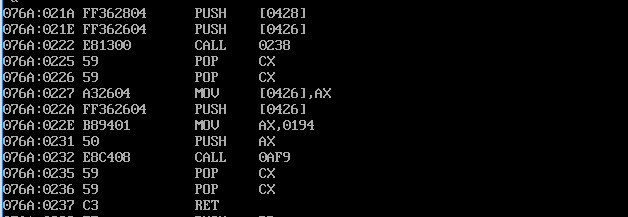
结果:



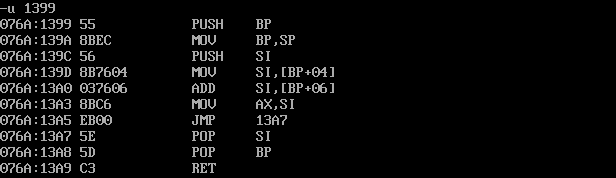
Debug之后：

Main函数内容：

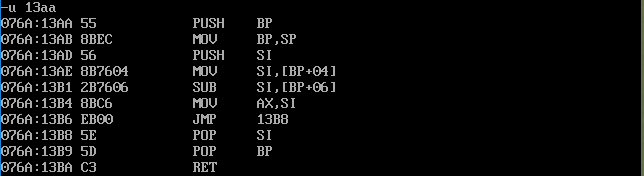




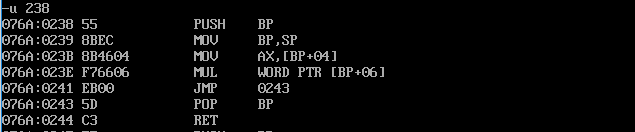
F1函数内容：



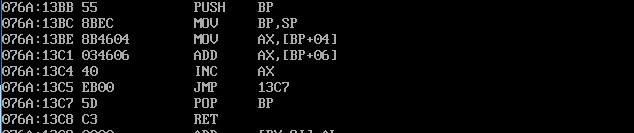
F2函数内容：



函数func的内容：



04demo42.c源程序中并没有调用f3，但是生成的04demo42.exe文件中有没有将f3中的代码连接进来呢？F2函数的地址是076a；13aah---076a:13bah，如果有f3很有可能在接下来的地址处存放，来看一下：



结果的确发现了f3函数。

为什么有f3呢？

先思考一下为什么不能有f3呢？

因为我们编写的04demo42.c源程序并没有调用f3呀，我们在现实生活中，对于不关心的事情就听之任之了，是不是我们把自己的主观臆断加在编译器的身上呢？编译器有这么智能吗？

我们知道，我们是将f1、f2、f3都编写在一个源程序里，并生成了一个\*.obj文件，那么看来这样机械式地将整个\*.obj文件内容都搬运链接就是tcc编译器做的事情了。

怎样解决呢？也就是怎样将f1、f2与f3区分开，要谁不要谁，和谁玩不和谁玩？

可能有以下几种方法吧：

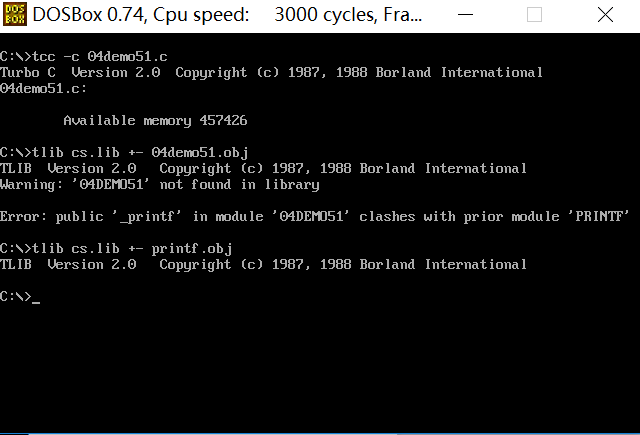
1. 在自己编写的04demo42.c程序中写明自己要和f1、f2玩，不和f3玩；
2. 在链接的时候，告诉编译器，自己要和f1、f2玩，不和f3玩，这里可以参考功能选项，不知道能否实现；
3. 暴力式的，一种是将f1、f2从原来生成\*.obj文件中剥离出来生成新的1.obj，将f3从\*.ob文件中剥离出来生成新的2.obj，再加入到cs.lib文件中；另一种是在编写源程序的时候将f1、f2模块编写在1.c文件中，将f3模块编写在2.c中。

搜到的网上资料：

当启动连接程序时，它会寻找“未定义的外部函数”，也就是说，它将在每一个库文件中查找源代码文件中未定义的函数。当它找到一个未定义的外部函数后，它会引入包含该函数定义的目标代码(obj)。不幸的是，如果这个函数是在一个包含其它函数定义的源文件中被编译的话，那么这些函数也会被包含进来，你的可执行代码中将包含一些不需要的代码。因此，将库函数放到各自的源文件中是很重要的——否则会浪费宝贵的程序空间。有些编译程序包含特殊的“精明的”连接程序，这些连接程序能查出不需要的函数并去掉它们，从而使这些函数不再进入你的程序。

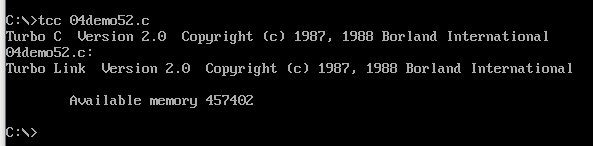
**5>.程序5部分：**

第一次的时候出现了错误，根据提示信息，应该是原来已定义的printf函数在printf模块中，所以我将原来生成的04demo51.obj重命名为printf.obj文件，再看一下：

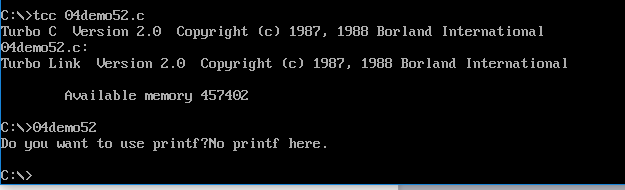


成功了，到底使用结果怎样呢？

Tcc 04demo52.c看一下：



运行：



1. **已思考研究并已解决问题汇总**
2. **已思考研究并未解决问题汇总**

**1>.自己提出的这几种动态编译的方式哪些可行，哪些不可行，理论上都可行吗？需要验证。**

**2>.自己对tlib的使用还很少，比如他可以直接用一个1.obj变成1.lib吗？这里的例子都是将1.obj这样的加入到现有的\*.lib中？还有正如之前的假设，我们可以实现剥离\*.lib中单独一个\*.obj模块中的其中一个或几个指定的函数吗？如果能，该如何做？需要验证**

1. **研究感想（心得体会）**

学到这里，再对比一下，之前自己也用过vs之类的专业级编程工具，但是总是觉得自己是“杀鸡用牛刀”，现在王爽老师针对tcc出教材，怎么说呢？如果将vs比作一条大鱼，一锅炖不下的那种的话，那么tcc应该就是一条小黄鱼吧。但是呢，王爽老师会教我们怎么把这条小黄鱼身上隐匿起来的肉都剔除出来，然后都填进我们的肚子里，吃的那叫个干干净净呀，啧啧，“嗝~”，直到让你响起饱嗝。一个词总结，物尽其用。