# 第一二章宣讲会及研讨会问题总结

1. 当前环境下生成的exe文件有两个最大为64K的段，这是由什么决定的？

一个段限制为64k是由CPU和寻址方式决定的，8086CPU有20根地址线，16根数据线，采用段地址\*10+偏移地址的方式来寻址，16位的偏移地址最大寻址范围为2^16=64K。而之所以分两个段，且代码占一个段，栈和数据占用一个段，是与你采用的模式有关系的。具体的模式对应的代码和数据、栈的情况可查阅网上资料。

在网上查询资料发现，TC2.0是一个集成的开发环境，它集成了以下文件：

INSTALL.EXE 安装程序文件

TC.EXE 集成编译

TCINST.EXE 集成开发环境的配置设置程序

TCHELP.TCH 帮助文件

THELP.COM 读取

TCHELP.TCH的驻留程序

README 关于Turbo C的信息文件

TCCONFIG.EXE 配置文件转换程序

MAKE.EXE 项目管理工具

TCC.EXE 命令行编译

TLINK.EXE Turbo C系列连接器

TLIB.EXE Turbo C系列库管理工具

C0?.OBJ 不同模式启动代码

C?.LIB 不同模式运行库

GRAPHICS.LIB 图形库

EMU.LIB 8087仿真库

FP87.LIB 8087库

\*.H Turbo C头文件

\*.BGI 不同显示器图形驱动程序

\*.C Turbo C例行程序(源文件)

其中: 上面的?分别为:

T Tiny(微型模式)

S Small(小模式)

C Compact(紧凑模式)

M Medium(中型模式)

L Large(大模式)

H Huge(巨大模式）

通过查看TC2.0文件夹下的文件，确实是这样。

那么我们试验1的生成的 a.exe文件那么就是小模式，C0S.OBJ（小模式启动代码）

CS.LIB（小模式的运行库），对于这个库我猜想与这Printf函数的调用有关。

于是我删掉Printf,发现连接过程中竟然出现了错误，提示不能打开CS.OBJ文件。

于是我查资料发现得编译器在编译的时候会把所有包含进的函数都进行编译，

这无疑会使源程序变得庞大而难于理解，并会影响执行效率。建立用户目标模块

库（lib文件）可以解决这个问题，你可以事先把有特定功能的通用的子函数集

编译完成后生成的obj文件打入lib文件，当链接器tlink.exe在进行链接操作时，

便只会把程序内部用到的通用函数所在的obj文件链接进来，这样生成的obj文件便会更精简，

生成的exe文件也更小。对于这个验证我没能继续进行。但我明白了其实，一个lib文件

是obj文件的集合。当然，其中还夹杂着其他一些辅助信息，目的是为了让编译器能够准确找到对应的obj文件。

在微模式下程序中的数据及代码均放在同一段内，即它们不超过 64KB。在微模式下代码段、堆栈段和数据段的段地址均相同，即CS=DS=SS=ES。

在小模式下，程序中的代码放在64KB的代码段内，数据放在64KB的数据段内。在小模式下，栈段、附加数据段和数据段均指向同一地址，它们合三为一，即DS=SS=ES，指针都是near，一般程序均采用小模式编译。

在中模式下，所有数据放在64KB的数据段内，因而数据段内使用near，代码量可以大于64KB（允许达到1MB），因而可以在不同的代码段内，代码段使用（far远程指针）。来自不同源文件的码模块放在不同的码段内。

在紧凑模式下，数据量超过64KB时，可放在多个数据段中，数据段内的指针是（far）。代码量不超过64KB，在一个段内，因而代码段内指针为近程的（near）。但在该模式下，静态数据仍不能超过64KB，堆用far指针来存取。代码、静态数据、堆栈、堆各有自己的段。堆只有远堆，没有近堆。

大模式下，代码及数据均采用far指针，且都可达到1MB。静态数据、堆栈、堆同紧凑模式，代码同中模式。静态数据仍跟紧凑模式一样，不能超过64KB。

巨模式下，代码段及数据段均用far指针，代码分布在不同的代码段内，数据也分布在不同的数据段内，它们来自不同的源程序，大堆栈只有一个。而且静态数据大小允许超过64KB。

所以当前环境下生成的exe文件有两个最大为64K的段，这是由编译模式决定的

2.对于函数所在的段地址的打印开始的想法就是打印函数所在的\_CS

如printf("%x/n",\_CS);

对于函数所在的偏移地址的打印是

printf("%x/n",main);

编译连接后发现 真的可以。 但是我就入debug调试发现我的函数的段地址和我打印出来的

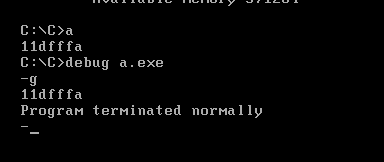
不一样，偏移地址一样。我没有想明白，我记得用char far好像可以

所以我输入：printf("%x",(char far)main);

编译连接的时候出现了错误，我很纳闷，突然发现%x不应该对应 char far 型

所以我改为：printf("%lx",(char far)main);

哈哈 终于成功了。



经过讨论好像我这个结论是错误的

# 补充：

打印函数地址的方式还有

(1) printf(“%lx”, (long)f1);

(2)printf(“%x %x”, \_CS, f1);

其中f1可以用\*f1和&f1代替。

1. debug中执行g 01fa（对课本上的程序讨论）为什么直接输出结果，然后退出了？

**直接执行了main函数里面的程序**

4. tc与tcc在编译链接.c文件的不同：生成.obj.exe文件大小不同，（tc需要链接GRAPHICS.LIB图形库文件）为什么要链接这个文件？在对于子函数处理方面，tc与tcc在处理上有何不同？

**graphics.lib是一个c语言图形库，TC2.0连接需要这个而tcc.exe不需要，我们可以理解为tc2.0在tcc.exe的基础上多了这么一个扩展，每一个库文件都相当于一个小模块，支持一种扩展。Turbo c包有两种编译器，集成开发环境下的叫做TC.EXE和命令行方式的叫做TCC.EXE. 集成开发环境包括:集成编辑器、命令行编译器、连接器、调试器，可以大致认为tc包含tcc**

5.程序中没有涉及到数学运算，为何需要maths.lib这个库文件？

**程序连接的顺序为 c0s.obj emu.lib maths.lib cs.lib**

**也就是说 emu.lib 和 math.lib 、math.lib和 cs.lib有关联，去掉这math.lib之后，emu.lib和后面的 cs.lib文件关联消失，所以，虽然没有涉及数学运算，但是如果没有math..lib的话，程序会出错.**

6.函数在内存中存放的顺序与什么有关？

答：与函数在程序中定义的顺序有关，而与声明顺序无关。

7.Exe文件中只有代码？还是连接了其他东西，如数据之类的？

**查看.c文件可知其大小小于.exe文件的大小 从.c文件要经过tcc编译。Tlink链接，不止包含代码. 选择某种模式会连接该模式下的.obj 和.lib文件**

8.怎么通过命令行指定采用的模式？

答：Tcc -mm a.c 其中mm是指medium mode，可以通过改变第一个m来指定不同的模式。