## 第一二章宣讲会及研讨会问题及解答总结

## 第一章

**宣讲会已解决问题**

1. 为什么共有大小为64K的段，一个为代码段，数据和栈共用一个段？

答：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。

1. 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。

1. 写一个空程序，其对应的exe文件大小与available memory的和是64K。
2. 若用tc.exe链接成exe文件，还需要将graphics.lib（图形仿真库）复制到文件夹下，而tlink.exe链接的时候不需要。

**研讨会解决问题**

1. 怎么验证在巨模式下，程序可以超过64K？

用一个大于64k的程序 选择巨模式

C0？.obj c？.lib

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)

你需要用哪一种模式就把“？”改成首字

母 然后编译链接

验证之后会发现单个文件超过64k会出错 **(1.怎样编写一个超过64的文件？）**

但是系统会把它划分开，分开编译，在连接，这样可以成功。

**(2.但是其他的同学说如果分开编译链接，其他模式也能运行超过64K**

**留下一个课后解答的问题：巨模式是如何工作的？）**

**①怎样写一个程序超过64K？**

可以编写一个数组包含超过65536个数

**②巨模式工作方式**

C语言编译模式—巨模式(Huge)

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。

紧凑模式、大模式、巨模式数据区大小均允许超过64KB，即可以用数据far指针对不同数据段内的数据进行存取，它们同称为大数据存储模式。但有一点不同：紧凑模式和大模式按 C 的规定，其静态数据，即如数组、结构或其他类型的数据被定义为静态类型时，其数据量不能超过64KB，而只有巨模式才允许超过64KB。在大数据存储模式下，堆和栈分别在不同段内，多以动态数据和局部变量的形式存放，这样就不受64KB大小的限制，栈的增长不会影响堆的空间。

**注：此处学长要求大家尝试用tcc编辑器链接两个.c文件**

1. 如果代码小于64K的话，系统会分配64K吗？还是根据程序的大小来分配？

通过查看操作系统的书，我发现对于内存的分配可以分为固定分区和可变分区两种，对于，对于未知大小的程序，系统事先并没有给它固定的分区，所以会根据程序的大小来分配字节，也就是分配小于64K。

**（给出的答案是系统会根据程序的大小来分配内存）**

1. 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**

1. Available Memory和程序大小之间存在什么样的关系？

**可用空间和已用空间加起来应该是64K**

但是今天有位同学验证的.exe文件加上Available Memory的大小却**不大于**64k

## 第二章

**宣讲会已解决问题**

1. 打印函数地址的方式有哪些？

答：(1) printf(“%lx”, (long)f1);

1. printf(“%x %x”, \_CS, f1);

其中f1可以用\*f1和&f1代替。第一种方式打印出来的是内存地址，第二种方式打印出来的是段地址和偏移地址，当然两者是等价的。

1. 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。

3. 函数名f1是一个地址常量，而\*f1是一个地址变量，此处的值是一样的。

4. Debug加载程序后，ES和DS的值是相同的，而CS和它们俩的值不一样。

5. 当代码不超过一个段时，

printf(“%x”,\_CS); printf(“%x”, f1);

和 printf(“%x %x”, \_CS, f1)得到的结果是一样的。

6. Tcc -c a.c

tlink cs.obj+a.obj cs.lst

然后通过cs.lst可查看里面的内容，主要目的是为了查看我们所用的c0s.obj等4个文件的内容，以及对编译连接的影响（包括数据和代码的问题以及对文件大小的影响）

**研讨会解决问题**

1. debug加载程序的时候为什么不直接加载到printf输出的地址？debug加载以后对应的段地址有什么特殊的意义吗？

第一句话没懂,debug 不是一开始就跳main函数运行main函数吗？ 对于第二个问题，这里其实是由于DEBUG加载.exe文件，需要对其进行多种处理比如说 t p g r u 命令等，这些都需要加载很多函数，这些函数占用一定的内存空间，所以其段地址和直接打印的不一样。**（不是这样理解）**

**Debug下加载的.exe文件时系统分配的一段虚拟内存，此处的段地址和在command下面加载的段地址不同。也就是说，command下面查看的.exe文件的段地址和在command下的debug下加载的.exe的段地址不同。**

**注：这里是需要大家研究为什么在command和debug下偏移地址一样?**

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**

1. &f，(\*)f，f三个有什么区别？

1> f只是一个标号,标号的段地址是其所在段的段地址，偏移地址是段地址到该标号的字节距离

2> 指针变量前面的\*表示该变量是指针型变量 指针变量是f，而不是\*f

\*f代表指针变量f指向的对象

3> & 取地址运算符 &f是变量f的地址

**注：还要继续研究\*f &f 的用法**

1. 如果代码超过一个段，那么在不同的段打印cs的值是否一样？如果在第一个段中定义了fun1，在中会打印处正确的段地址吗？

**Fun1只是一个函数名, 它在第一个段中，printf(“%x,%x\n”,\_CS,fun1)打印的是其所在段的段地址；如果fun1在第一个段，但是其程序超过了段的内容，其段地址还是f所在段的段地址。关于第二句话，你这里是打印的\_cs 而不是fun1 的段地址 所以不会打印正确的段地址，可以把第一个段的段地址先放到 一个变量里面，再打印这个变量。**

1. 怎么验证代码和数据、栈是在不同的段？

**代码 数据 栈都存在寄存器中 查看寄存器的段地址就可以知道是不是在同一个段**

1. Cmd和command有什么区别？

**我记得上课刚开始提的问题好像不是这个，而是在cmd下编译后运行程序看到结果后会马上跳出，解决方法是在你的main（）函数里面加一句**

**systerm（“pause”）** 或者  **getch（）** （系统问题）

**注：不推荐大家使用"system("pause");**

1. 不可移植。只适合DOS或Windows，但是不适合Linux等   
2. 耗费系统资源。调用系统命令system()，去做"暂停程序"的事情有点大材小用。   
3. 必须添加头文件：stdlib.h或者cstdlib   
总之这是一个坏方法，应该摒弃。

  推荐： C中，使用getchar();  C++中，使用cin.get();

**关于cmd和command的区别是 cmd是32位命令行 command是16位命令行 这两个不是一个东西，前面并不是后面的缩写。[2333]**

7. 上述的第6个问题如何得到cs.lst等可以查看代码的文件方法是怎样的？

**.lst 只是一个后缀名，windows里面显示后缀名是为了方便识别，打来的方法可以用对应的工具打开，比如说.txt可以用记事本打开 .docx 可以用word文档打开.**

宣讲会记录人： 高建花 刘宁

研讨会主持人：周运腾

**学长补充：**

1. 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？

（提示：分别对于在tcc和tc下编译链接生成的exe文件，反汇编查看代码。）

**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**

**graphics.lib是运行之前里环境用到的，为什么之后的tc里也要连接这个？**

**在后面编辑的时，比如在tc界面写.c程序，这里会显示黄色的代码，这里也要用到图形库文件，所以也要连接GRAPHICS.LIB**

1. 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.**

1. 在C语言中，函数实际上是一个指针吗？指针函数怎么定义？为什么可以直接将函数名当作标识符常量来打印？

**1> 函数名是一个地址，函数指针是是指向这个地址的一个变量，不是一个概念。**

**2> 如果在程序中定义了一个函数，在编译时，编译系统为函数代码分配一段存储空间，这段存储空间的起始地址（又称入口地址）称为这个函数的指针。**

**可以定义一个指向函数的指针变量，用来存放某一函数的起始地址，这就意味着此指针变量指向该函数。eg：int（\*p）（int，int）；**

**3> 函数名就相当于函数代码的首地址，是一段数据，所以可以当做常量来打印。**

**研究体会：**非常感谢学长在本次研讨会中给大家提供的一系列问题的解答，像学长所说的，我们研究问题不是把它研究的越来越复杂，而是要逐渐把它研究清晰，并理解，不能简单地说‘我认为’‘你认为’，要通过自己验证得到准确的答案，不要主观猜测。

**补充：**

C 语言中提供了6种编译模式，这6种模式是：

微模式（Tiny），小模式（Small），中模式（Medium），紧凑模式（Compact），大模（Large）和巨模式（Huge）。它们之间的关系如下图所示。用户可以按照自己的程序大小及需要进行选择。  
　　　　　　│ 小程序　　 │ 大程序  
　　━━━━┿━━━━━━┿━━━━━━━━  
　　 小数据 │ 微，小　　 │ 中  
　　 大数据 │ 紧凑　　　 │ 大，巨  
 所谓小程序就是指程序只有一个程序段，大小不超过64KB，缺省的码（函数）指针是near（近程指针）。所谓大程序就是指程序只有多个程序段，每个程序段不超过64KB，但总程序量可超过64KB，缺省的码指针是far（远程指针）。小数据就是指数据只有一个数据段，缺省的数据指针是near。大数据就是指数据有多个数据段，缺省的数据指针是far。

由上可知，我们所说的只有一个代码段的程序是小程序，它的代码不超过64kb，在编译时会以默认的编译模式：小模式来编译。即前面研究里所说的tcc a.c生成的exe文件有一个代码段，一个栈和数据段就可以理解了。我们用TC2.0的时候不可缺少的相关文件里有关编译模式的是c0s.obj和cs.lib，所以TC2.0默认的编译模式是小模式。所以默认的编译只能编译代码量不超过64kb的文件。

关于不同模式的区别，查询资料如下：

C语言编译模式—微模式(Tiny)

 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。在微模式下，数据指针都是 near，一般小程序可采用此编译模式进行编译。还可用 DOS 中的 EXE2BIN 转换程序将.EXE 程序转换成.COM 程序。代码段、数据段和堆栈段均在同一段内，对它们进行寻址时，均以同一地址偏移的参考点，具有这种特点的段又称为属于同一组段（DGROUP），栈是向上生长的，即每压栈一次，栈指针SP减2，即向地址减少的方向移动，它开始的初始值指向栈底，即0xffff（64KB）。堆是向下生长的，即向增加地址的方向改变。堆和栈地址相向生长，当两者未相遇时，便出现了自由空间。一般程序均是这种状态，当占用栈地址较多时，两者可能重合并覆盖部分堆空间。

C语言编译模式—小模式(Small)  
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。在小模式下，内存分配如下图所示。从图中可以看出数据段、堆栈段和附加段为同一段组，即它们的偏移地址均以同一段地址为参考点。

C语言编译模式—中模式(Medium)  
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）。这种编译模式适用于大代码量、小数据量的大程序。中模式下的内存分配如下图所示。

C语言编译模式—紧凑模式(Compact)

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时，可在一个段内，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。紧凑模式下的内存结构如下图所示。

C语言编译模式—大模式(Large)

大模式下，代码及数据均采用far指针，且都可达到1MB。静态数据仍跟紧凑模式一样，不能超过64KB。大模式下的内存结构如下图所示。

C语言编译模式—巨模式(Huge)

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。巨模式下的内存结构如下图所示。

紧凑模式、大模式、巨模式数据区大小均允许超过64KB，即可以用数据far指针对不同数据段内的数据进行存取，它们同称为大数据存储模式。但有一点不同：紧凑模式和大模式按 C 的规定，其静态数据，即如数组、结构或其他类型的数据被定义为静态类型时，其数据量不能超过64KB，而只有巨模式才允许超过64KB。在大数据存储模式下，堆和栈分别在不同段内，多以动态数据和局部变量的形式存放，这样就不受64KB大小的限制，栈的增长不会影响堆的空间。

**不同模式的区别在于可编译的代码量和数据量不一样。那么在编译的时候对于超过64kb的文件该如何选择编译模式呢？**

无论采用哪一种编译模式，C 源程序编译生成的代码和数据量都不能超过64KB，对于超过的源程序，可以视代码或数据多少将其分解成两个或多个程序分别编译。大代码量程序要选用大代码编译模式（中模式、大模式和巨模式），大数据量程序应选用大数据编译模式（紧凑模式、大模式和巨模式），这样编译生成的.obj 文件将会带给连接程序信息，将代码和数据安排在不同段内。这样生成的.exe 文件在加载时将告诉 DOS 该程序应如何装入代码段和数据段，如何初始化寄存器。这样，就可确定在不同编译模式下开辟数据区的大小，即大于64KB，或不超过64KB。