第一次宣讲会问题记录

一、已经解决的问题：

宣讲阶段：

1. 头文件的作用和生成的四个阶段：
2. 预处理（include这个库寻找定义头文件的内容插入源程序中生成.i文件）
3. 编译(将.i文件翻译成生成.s后缀的汇编语言程序)
4. 汇编 （将.s文件翻译成机器语言，生成可重定位的目标程序，打包生成.o文件）
5. 连接（将各种代码和数据整合成一个单一可执行文件）
6. 为什么是实验中所写的源程序不需要stdio.h这个头文件呢？

因为C0S.obj的作用包括初始化，所以不一定需要include<stdio.h>，另外这个文件里面应该是只有一些函数的声明，并没有printf之类的函数的定义。

1. 有关于程序的的编译过程：

虽然最终都会编译成机器码，但存在有的程序不必编译成汇编语言，c语言必须转变成汇编语言，因为C语言是汇编编译。

1. main的偏移地址问题：

通过printf(“&x”,main)打印出来得到的。为什么能够打印出，main相当于一个标号。

1. 变量和标号的区别：

标号在编译过程中会被替换成具体值，而变量本身就是一个内存地址，在整个程序执行过程中一直存在。变量除了本身存的值外还有地址，而标号只有地址。

1. Buffer指针问题：

buffer指针指的是括号里地址的内容。 buffer(（char \*）\*(int far \*)0x200 ),buffer 的地址确实是0：200。Buffer[0]=0,赋值两个字节，尝试自己改变一下变成四个字节。

1. Sp、bp问题：

作用一：与SP联合使用，SP不能轻易更改，所以赋值给BP，用BP来实现我们的要求；第二个作用：栈帧，相当于一个链表，bp,sp代表一个栈帧的序列，为了方便调用其他函数，返回之后恢复调用前状态。

1. f函数不能正确返回的原因：

程序开始就是f函数，汇编代码中没进行调用就用了ret，所以返回不到正确位置，不知道程序跑到哪里去了，到哪里都是巧合

1. printf如何判断参数的传递：

通过第一个参数%d之类的来判断后面的参数个数的实现。而参数的个数由逗号分隔。

听会人提问阶段：

1. 实验一中程序没有写头文件，但是用了printf函数，怎么解释？

在编译连接过程所必须的五个文件里进行了具体实现。不在头文件里。

1. 每次编译main函数偏移都是01FA，原因？

猜测与环境、编辑器之类的有关。TC是在C0S.OBJ有一个call，call过去之后又一个call，call的地址是01FA，把入口实现到01FA。

1. 并没有打印出段地址，怎么确定1FA是当前的段地址的偏移？

直接打印cs寄存器。u命令查看都是正确的做法。有人提出一个有趣的做法：在main里写了一个死循环，所以一直在入口地址，所以打印出了正确的。

1. pop cs作用：恢复栈
2. 实验中某个程序加载之后发现，ss和ds是相等的会否产生冲突？

全局变量逐渐增多，就会发生冲突，栈溢出是有可能的。

1. 有人提出f函数中，语句ret不是小程序的结束，依据是虽然汇编语句很少，但是生成的可执行文件大小远大于代码大小。但其实exe文件不只是代码而已。文件大小与文件所占空间大小不同。原因是文件存储时以磁盘扇区最小单位划分， 不一定符合文件大小，会产生碎片。
2. 有人提出f函数中，dos加载f文件，语句ret返回到dos里面调用语句的下一句。否。不会反回dos，因为本身看整个过程就是在debug中，并不是dos加载的。
3. F函数虽然执行，但编译器和连接器会报错没有栈的原因？

没有C0S文件，所以没有初始化，自己申请栈时，c的编辑器不能识别汇编写的栈。

1. 如何在入参时和执行时判断参数的个数？

入参时printf判断传递参数的个数，% 来绝定打印参数个数，由逗号分隔参数，编译时用0来判断结尾。

1. C0S文件和写的文件如何结合？

其实只是僵硬的连在一起，是跳出去又跳回来。S后面是段结束语句。追问main标号的意义是什么。问题详见未解决问题四。

学长指出的问题：

1. printf真的不能打印出正确的cs吗？

在printf函数call之前就会入栈，所以肯定是打印正确的段地址。

1. bp问题：

bp,sp是因为要用栈来放局部变量，实现局部变量的作用域问题，栈帧就是入参位置的标记，子函数返回，栈就回复调用前的状态。

1. main函数偏移地址问题：

并不是所有的main函数一定偏移地址都是1FA，main函数前面再定义一个别的函数就会改变了。只是函数声明等放在main函数前不影响。

1. 针对栈溢出：

TC里面Option提供有一个编译模式，为你的程序选择合适的编译模式。动态申请内存。不可能完全避免。写程序需要留意。

1. 实验四与实验三 buffer语句不同：

指向null的含义，并不是不指向任何地方，只是指向的地方不安全，NULL的含义由某一个头文件决定。头文件来定义一些符号。

1. 未解决的问题：
2. 编译连接必须要添加的五个文件作用？有的明知用不到，为什么仍然要全部包含？五个文件中有一个文件对printf进行了实现，是哪个？请写程序证明。

二、考虑这几个语句含义：BUFFER[0]=0 &BUFFER BUFFER=0

三、标号S问题。即如何写一个c0s.obj使编译器和连接器在编译、连接的时候可以一上来就调用main函数，而不用管main函数所在的地址是什么，都能打印出正确main函数地址（main函数的调用只与main本身有关，与main函数的地址无关）。

四、为什么自写的c0s.obj，不能用malloc函数？malloc函数在哪里？如何证明？方向：可以用研究printf的方式来做，link连接各个obj、Lib来详细地看连接文件的过程。

五、 printf函数没有声明就被我们调用，但是编译连接都没有错，原因？

六、头文件和函数声明的作用。声明可以不做，那么为什么默认要做？两个猜测：为了推迟函数定义时间，让编译不报错；编译器做了类似声明的操作。

1. 扩展问题：
2. C语言里也存在标号，如go to语句。
3. 二维数组就是一个指向一维指针的数组。并不是二维指针。
4. 通过printf对数组名取地址取出的值与数组本身是几维也有关。
5. Printf(“%X”, )以十六进制输出。
6. 一段优雅的汇编代码：当函数返回是结构体时，从汇编中逻辑看不懂。（以后会遇到的。忽略。）
7. 内存五大区。代码段，数据段，栈段，等等，对于理解操作系统有好处
8. 所有的变量包括数组都有空间，但main是没有的
9. lib是库
10. ...是个va\_list宏定义

【附录】：

1. 真相都在代码里
2. 学习方法应该是实践探究来引导学习。
3. 学习的重点应该是学习基本能力，与系统和语言无关。去关注该关注的问题。不要被其他的问题所打扰。
4. 坚持用最初的TC好处在于比较细致，可以看到更多的问题。
5. 通过实践研究来引导学习比较合理。
6. 大学掌握的是一种解决问题的能力。

有诸多纰漏，望海涵

苗辰 包新泽

2017.5.24