《漏洞利用及渗透测试基础》实验报告

姓名：管昀玫 学号：2013750 班级：计科一班

**实验名称：**

格式化字符串漏洞

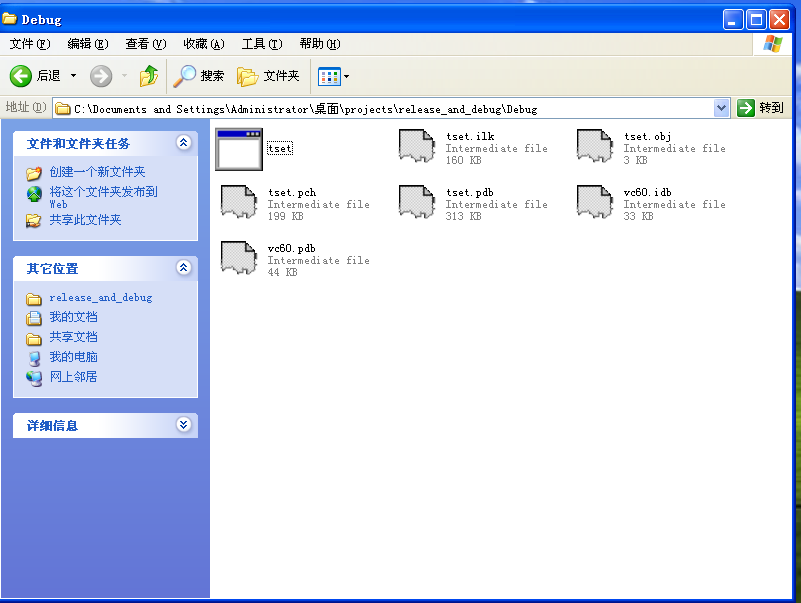
**实验要求：**

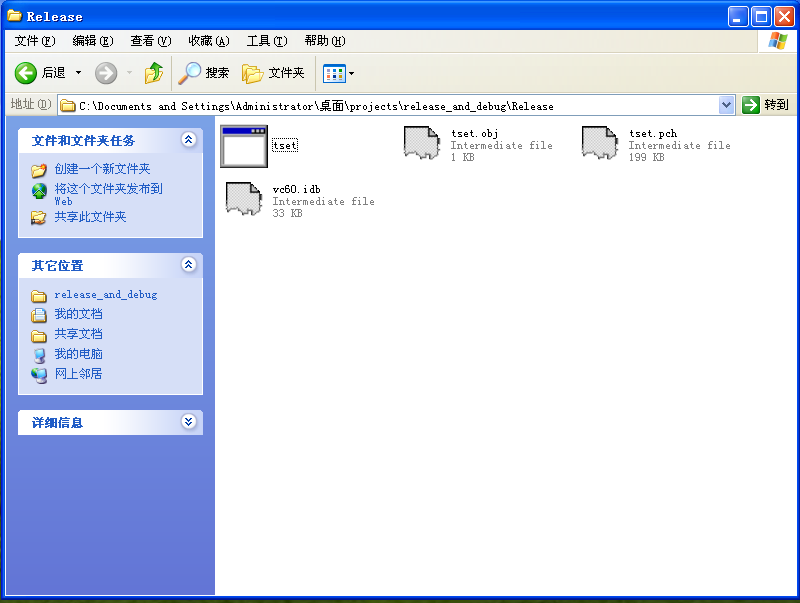
以第四章示例4-7代码，完成任意地址的数据获取，观察Relase模式和Debug模式的差异，并进行总结。

**实验过程：**

1. 代码准备

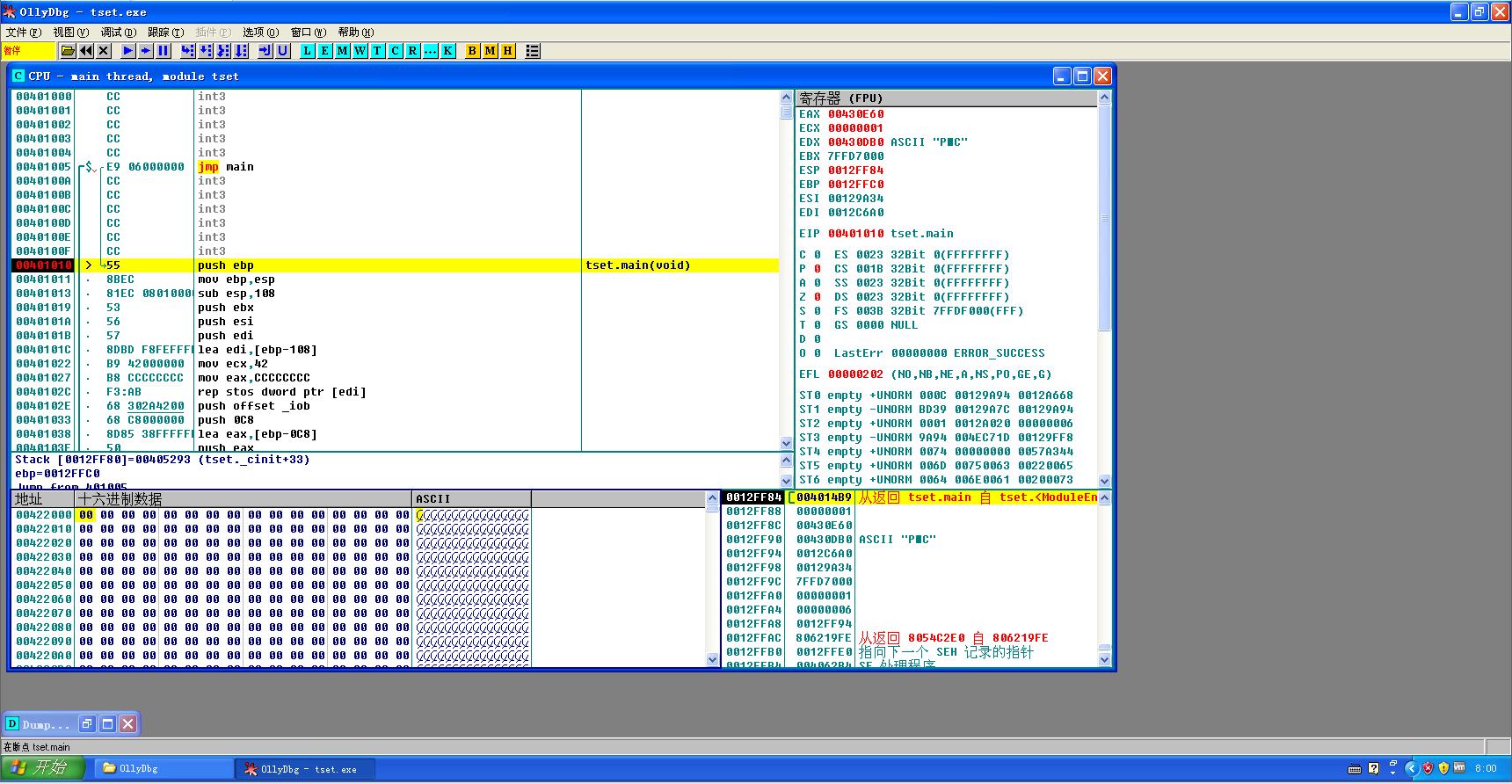
在VC6中输入示例4-7的代码，并且用debug和release两种模式进行编译，生成两种模式下的可执行文件





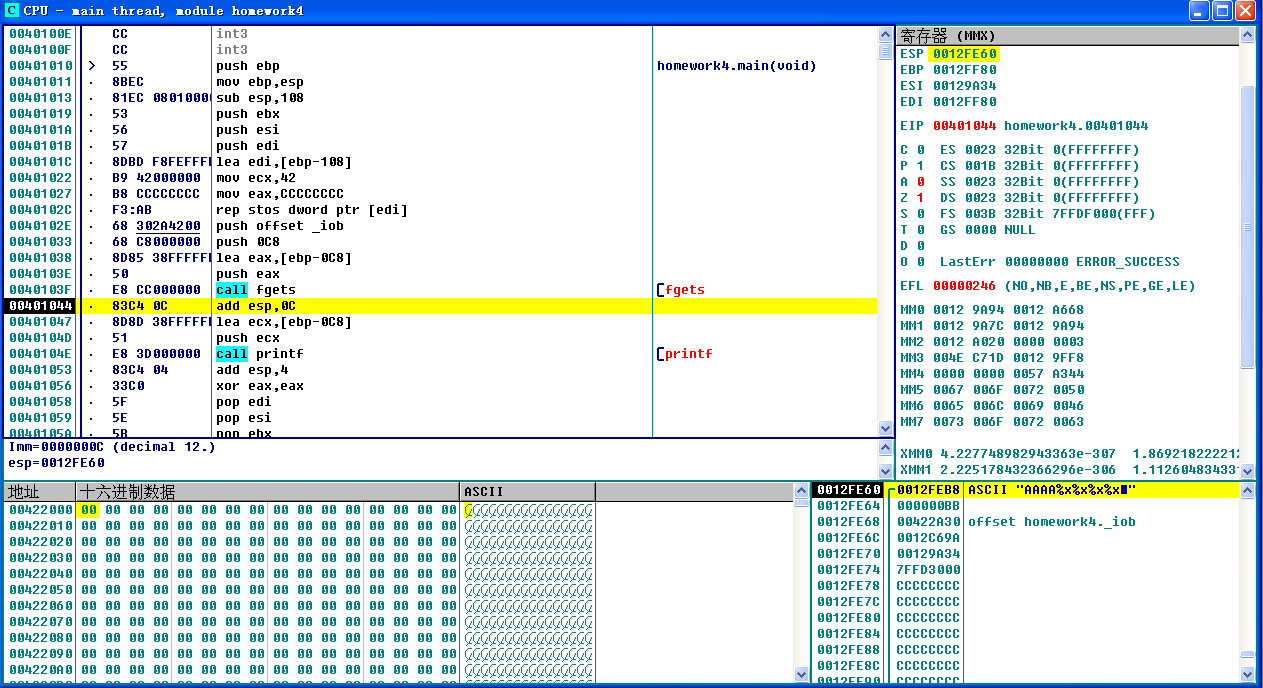
1. 使用OllyDBG观察debug模式下的状态

首先找到debug模式下的main函数：



在执行fgets函数时输入字符串：AAAA%x%x%x%x

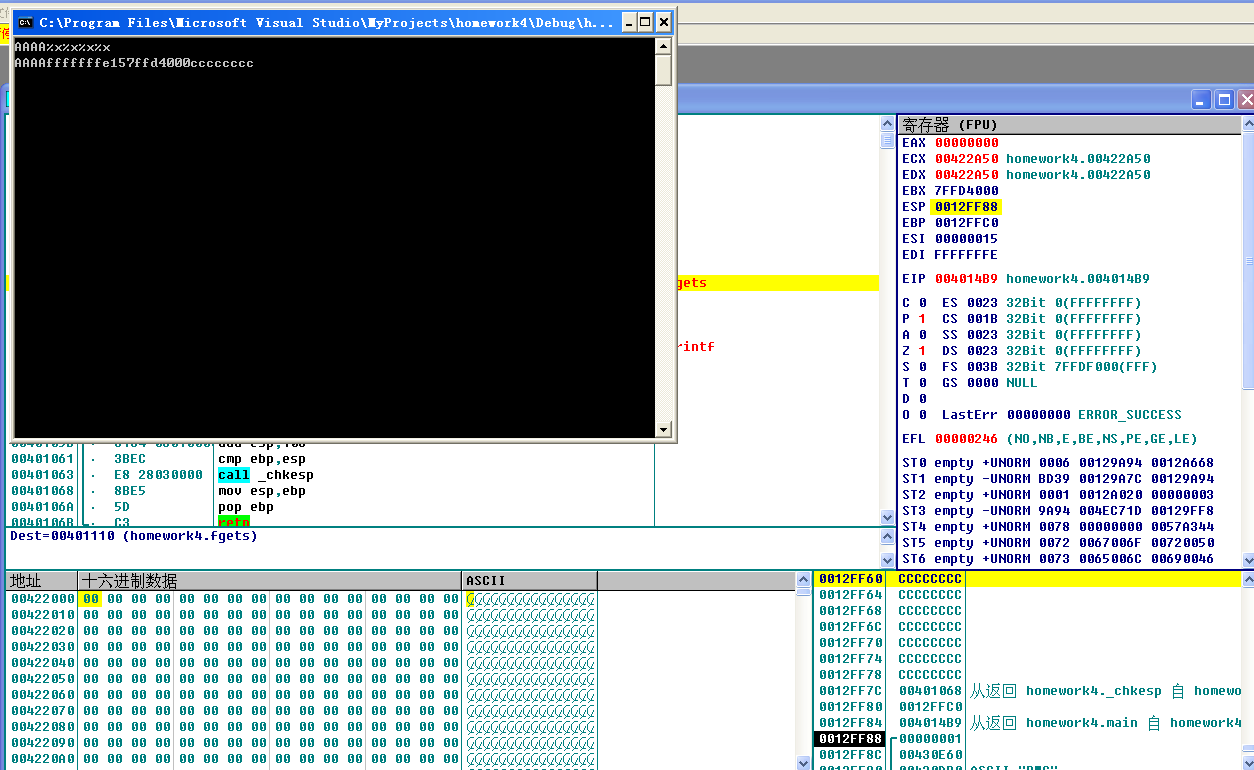
可以得到：



如上图右下角所示，栈区上存入了AAAA%x%x%x%x

观察此时栈区的状态：从0012FE68开始，向下依次存入的有很多0，还有7FFD80和很多的CCCC去填充空间

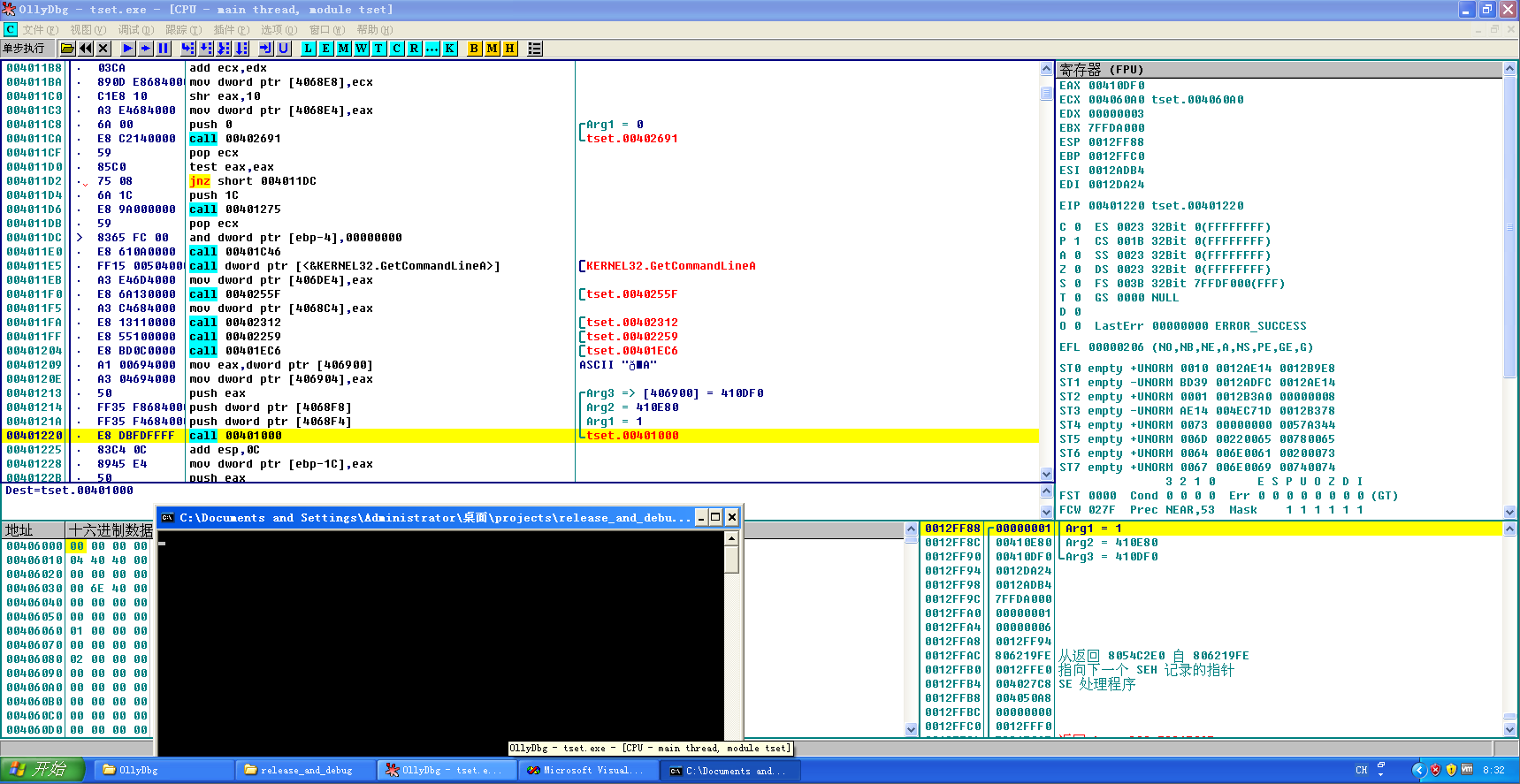
此时执行printf函数得到：



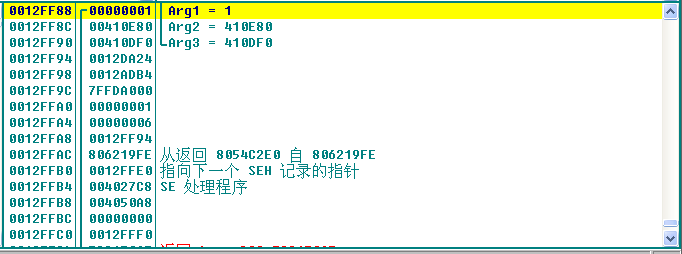
可以看见输出了栈中str以外的栈空间的数据。

1. 使用OllyDBG观察release模式下的状态

在release模式下，由于main函数在执行时，反汇编窗口右边的ASCII不再显示是call main，而显示的是test.某地址，所以通过按F8，一直执行到需要输入：

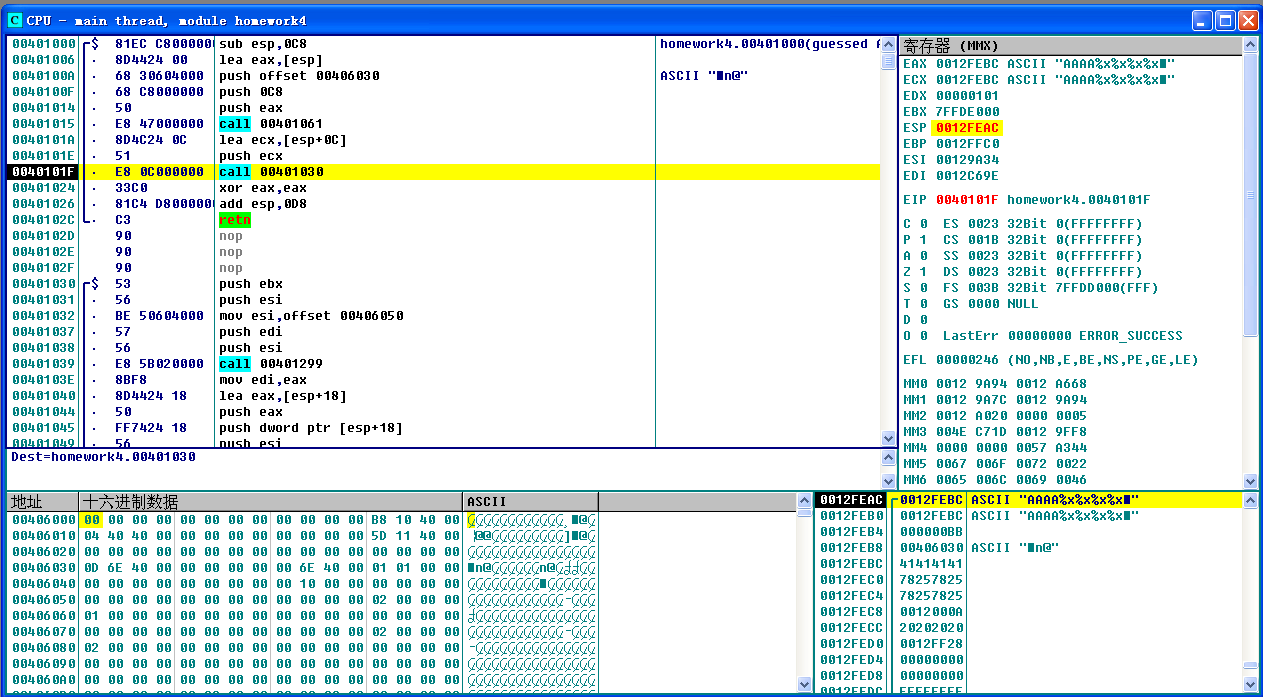


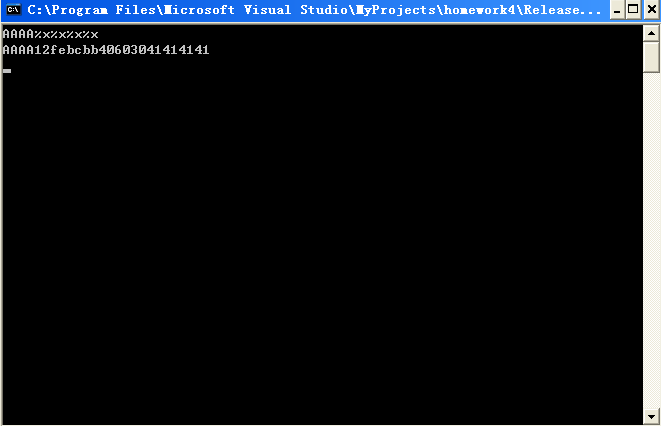
观察右下角栈的结构：



可以看到此时有Arg的参数入栈；

输入字符串之后，可以看到0012FEBC参数入栈了，栈上出现了两个AAAA%x%x%x%x，而下面的第三行就是41414141，就是字符串的起始地址。



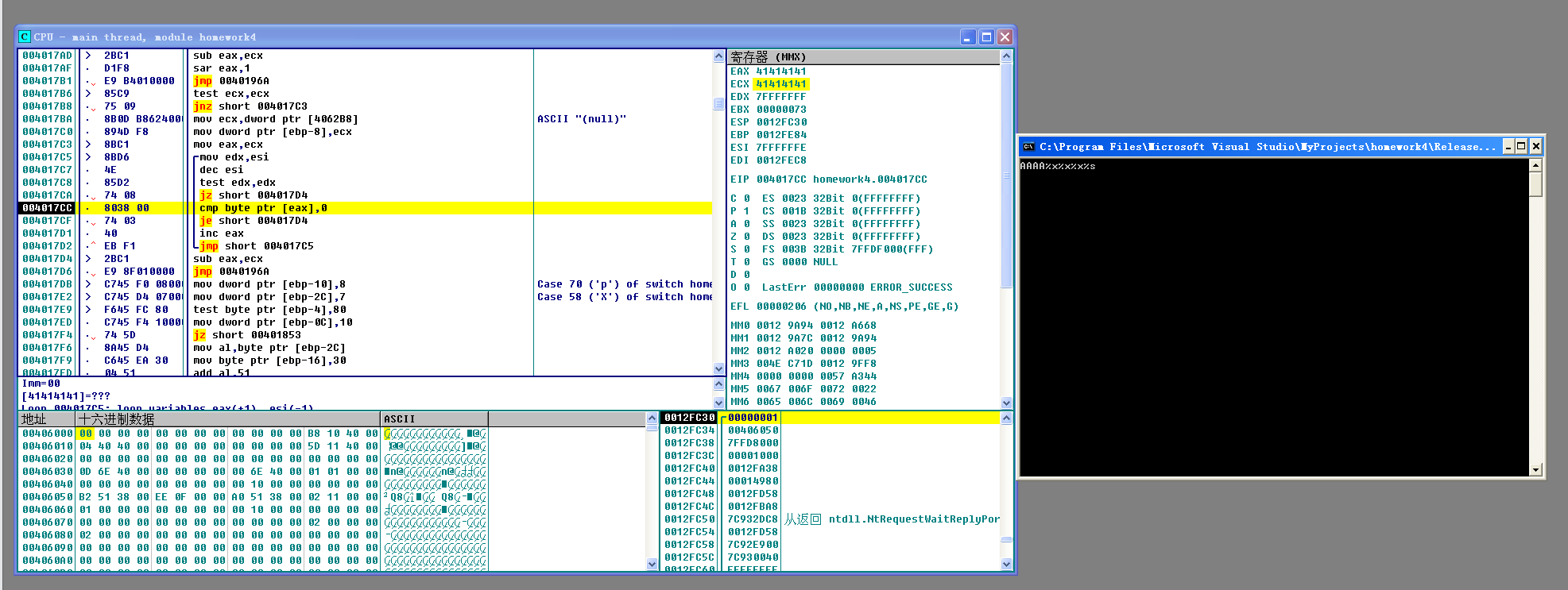
并且继续使用F8向下执行时，程序直接结束了。输出为：AAAA12FEBCBB40603041414141。

可以看到，第一个AAAA是输入的内容，而12FEBC、BB、406030、41414141都是0012FEBC之后的地址，而A是0x41,41414141则是AAAA的值。

由此推断第二个AAAA应当是printf函数的参数进行了入栈。并且栈没有像之前那样，给str真的去分配200字节的空间，而是只给了5个字节（在0012FEBC地址上的44 43 42 41 0A，其中0A是换行符），之后不足4字节的部分用0来填充。

尝试输入AAAA%x%x%x%s：

可以看到程序停滞



OD中有提示：



由于最后输入的是%s，程序取出栈顶指针，当做是指向字符串的指针。但栈顶是0x41414141，是一个非法地址，在访问的时候程序一场。但是，如果精心设计，可以获取任意地址的内容。

**心得体会：**



能够获取任意内存数据的原因：调用这些函数时，如果给出了格式化符号串，但没有提供实际对应参数时，这些函数会将格式化字符串后面的多个栈中的内容弹出作为参数，并根据格式化符号将其输出。调用时如果传入”%x%x…%x”，则printf会打印出堆栈中的内容，不断增加%x的个数会逐渐显示堆栈中高地址的数据，从而导致堆栈中的数据泄漏。

Release和Debug的不同：Debug模式下，因为开辟了足够大的栈帧并初始化，char str[200]是从靠近EBP的地址分配空间，如果要读到str的地址，需要很多的格式化字符；但是，Relase模式下，可以看到，并没有严格按照制式的栈帧分配，而是考虑运行性能，在执行到printf(str)的时候，栈区自顶到底部分为存着“printf函数参数|fgets函数参数|str数组”的内容，在Main函数的retn语句前，才有一个add esp XX的处理。如图所示，printf参数往下的内容并不相同，因此在Debug和Release模式下进行输出，会得到不同的结果。

如果把%x换成%s，则能够输出对应地址所指的字符串。比如输入AAAA%x%x%x%s，就能获取0x41414141地址上数据的输入。因此，通过精心设计地址，可以读取任意内存地址的数据，造成极大的危害。