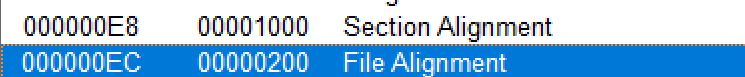
1. 修改对齐粒度

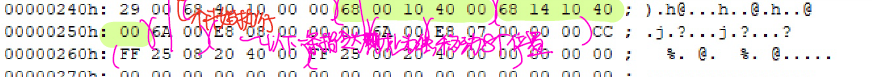


由于对齐填充了很多0，因此可以将内存和文件的对齐粒度改为4，使内存偏移和文件偏移一致，因此有很多入口地址需要更改

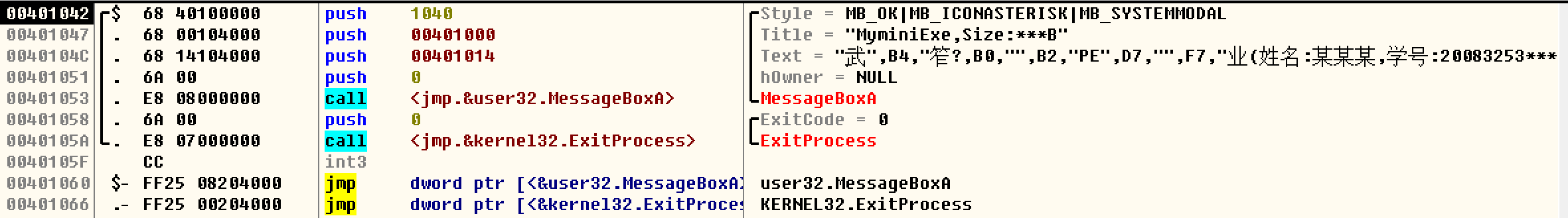
首先根据程序入口



1042对应文件偏移是242，可以找到代码执行开始的位置



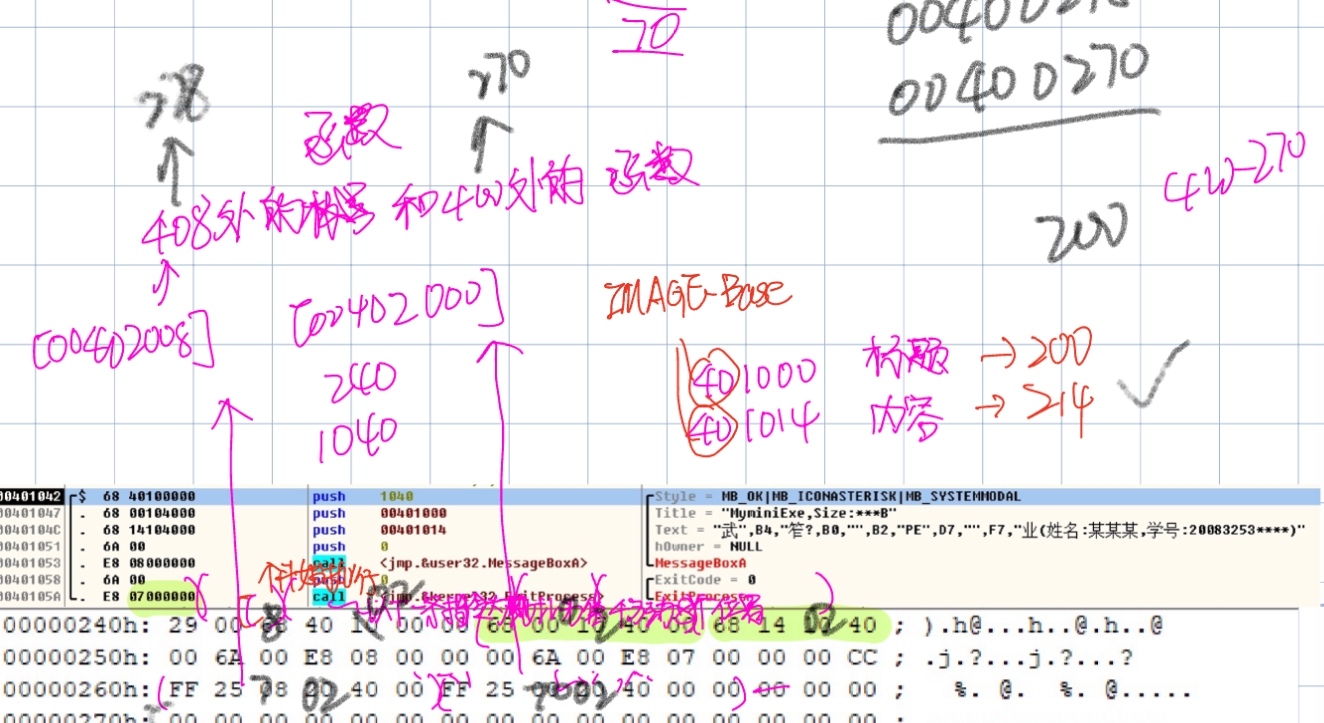
我们对照ollydbg中的解析来看



401000->标题; 401014->内容

402008->MessageBoxA; 402000->ExitProcess

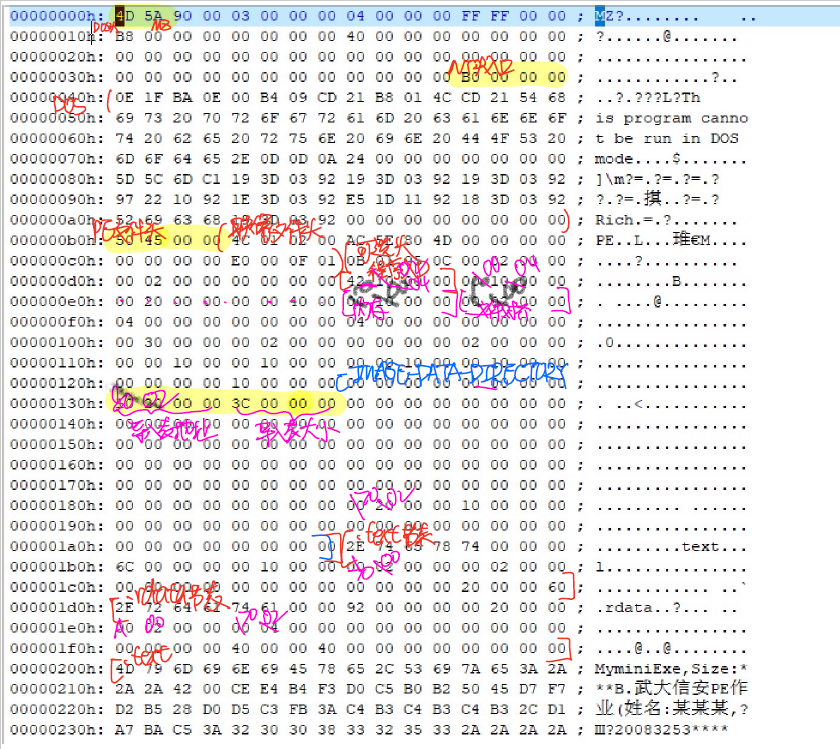
这些位置都是地址，因此都可以修改

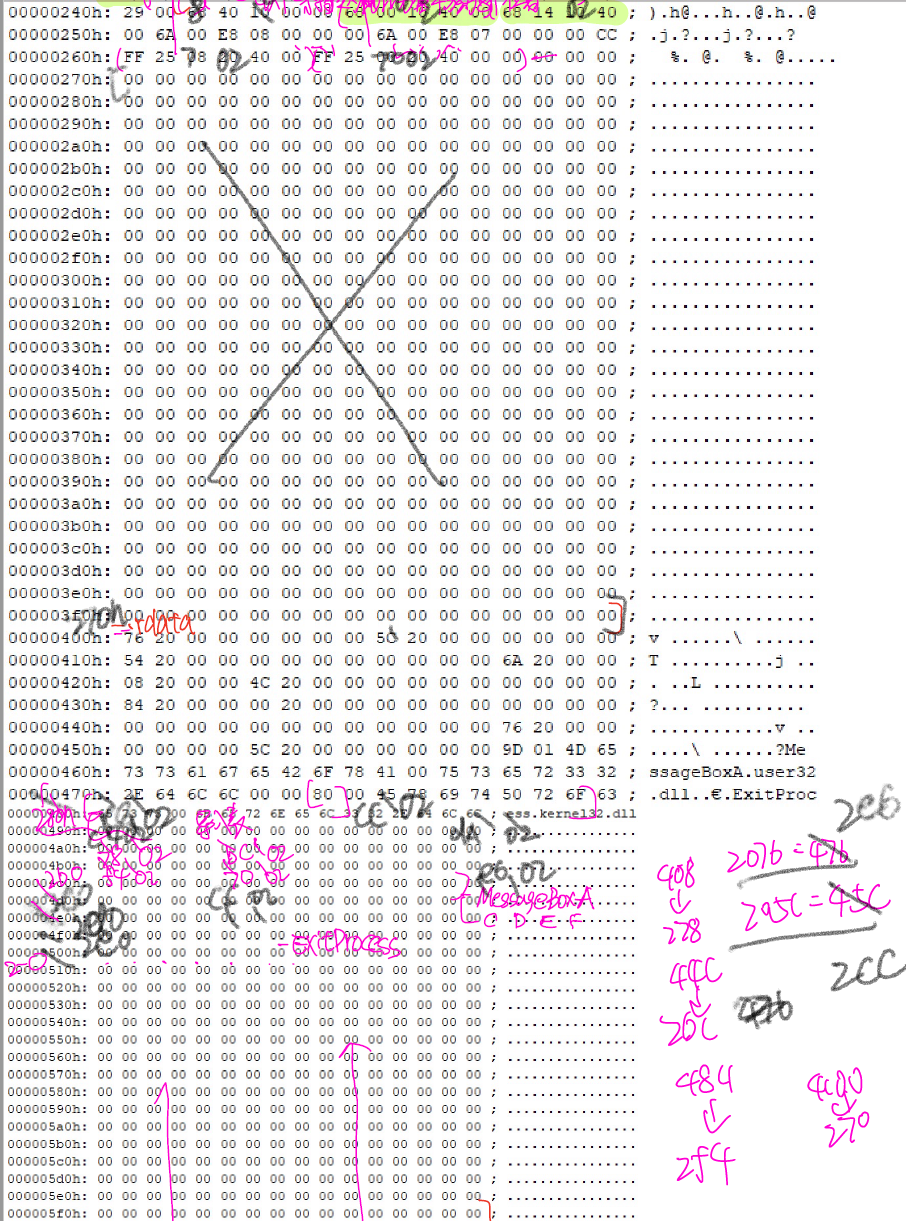


如1000->200，这样修改（此处跟文件偏移是100，内存偏移是2000有关，就不多讲了）

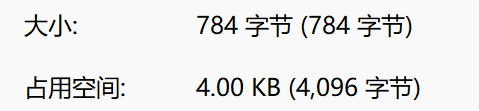
因此全篇涉及到地址的地方都要这样修改，我的步骤是先把该删的0都删掉，然后再看那个地址的变化，变化的的地址都要修改

大概要修改的地方我都在图中标了出来



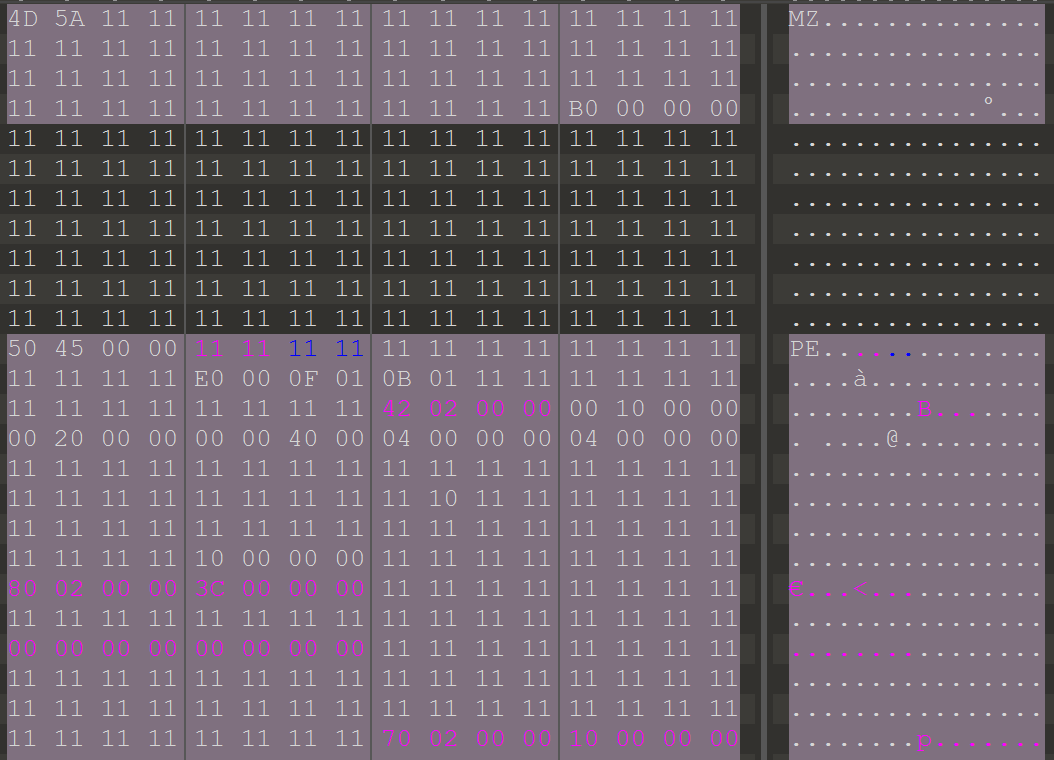


这样的话，改完之后可以压缩到784个字节



1. 去掉无效部分

这里就尝试了比较笨的方法，把没有用的地方先置1，看能不能运行，不能运行的话就说明这部分还有用

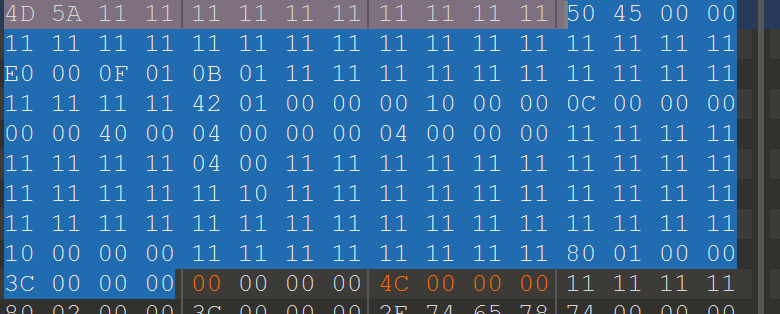


就比如DOS STUB没有用，我们就可以将其删掉，同时所有的地址都要上移4行，我们要修改所有的地址部分

三、重叠结构

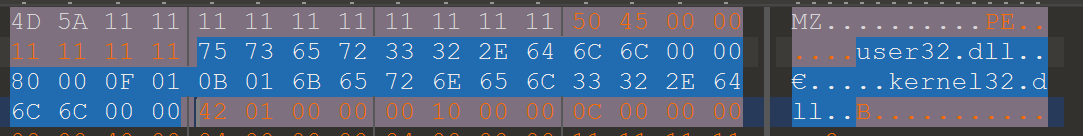
还有一些无效部分，我们可以通过重叠结构进行优化

接下来我们将DOS头和PE头进行重叠

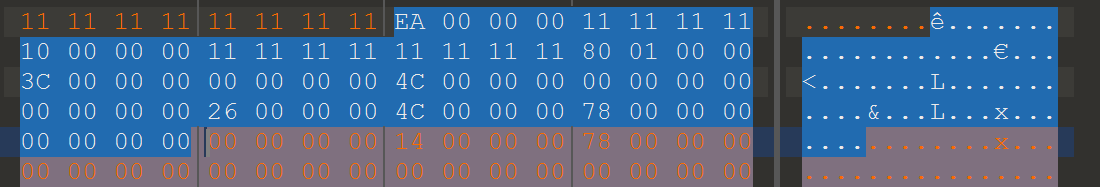


注意将DOS头原来03ch指向PE文件头的部分改为PE文件头的新地址，映像文件头中的程序入口RVA，可选头中导入表的位置都要做修改

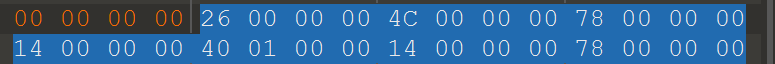
可以看到PE头从+08h（文件构建日期）到+14h（可选头大小）以及映像头的开始到程序执行入口之间的部分其实都没啥用，我们可以将dll文件塞到这里面：



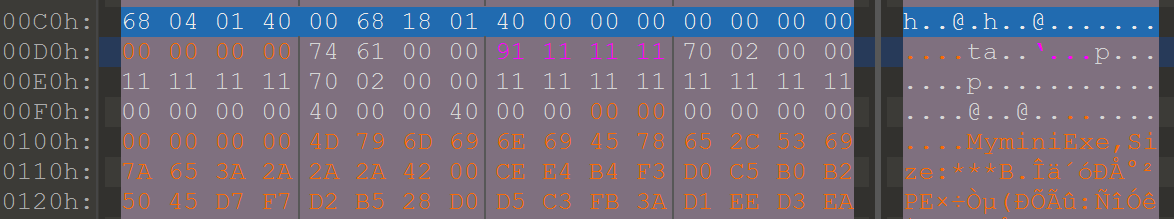
此时我们要修改导入函数表，将其放到可选头之后，同时还要修改可选头中导入函数表的入口，由于上面我们给user32和kernel32挪位置了，所以我们要将导入函数表中对应的指针也进行相应修改



接下来我们再将代码节头往前合并，将其插入到引入函数表中

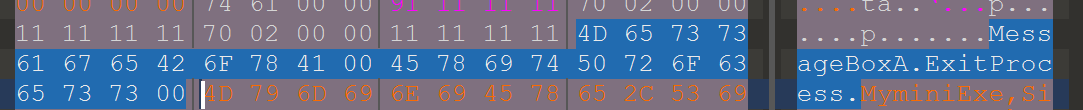


此处RVA和PointerToRawData为14h，都对应user32字符串指针，然后我们把代码段紧挨着节头表放入

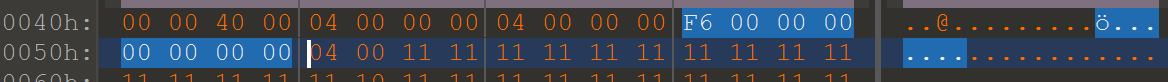


对应前面说的将标题和内容push进去，其中标题和内容所在的地址要修改成相应的地址，这里是0x400104和0x400118

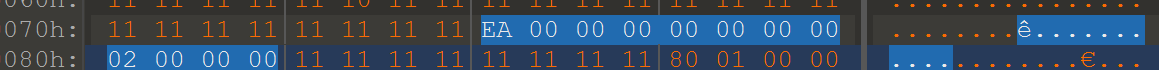
user32和kernel32已经挪动了，同理我们要找位置给ExitProcess和MesageBoxA挪进去,我发现标题前面有6字节的无效区域，我们可以直接将它们塞进去



同时要修改其引入的地址表项和引入名字表项，一个巧妙的注意是将引入地址表和引入名字表合并，我们在上面引入表中给他们找位置



其中4ch到53h指向ExitProcess的引入地址和引入名



而78h到7eh指向MessageBoxA的引入地址和引入名

最后将不必要的部分删去，之前置1且没有被占据的部分恢复正常值，其余的字符串放到代码之后，并且调整相应的地址