编号：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 实验 | 一 | 二 | 三 | 四 | 五 | 六 | 七 | 八 | 总评 | 教师签名 |
| 成绩 |  |  |  |  |  |  |  |  |  |  |

武汉大学国家网络安全学院

课程实验(设计)报告

题 目： 实验五：逆向工程与软件自我保护

专业(班)： 信安5班

学 号： 2022302181161

姓 名： 王亚鹏

课程名称： 软件安全实验

任课教师： 赵磊

2024年 11 月 27日

**目 录**

[实验5 逆向工程与软件自我保护 1](#_Toc183705280)

[5.1实验名称 1](#_Toc183705281)

[5.2实验目的 1](#_Toc183705282)

[5.3实验步骤及内容 2](#_Toc183705283)

[5.4 实验心得体会 9](#_Toc183705284)

## 实验5 逆向工程与软件自我保护

### 5.1实验名称

**逆向工程与软件自我保护**

### 5.2实验目的

 **掌握逆向工程的基本方法和工具使用**：熟悉逆向工程中常用的调试工具（如OllyDbg）和编辑工具（如010editor），学习如何分析软件内部逻辑和修改程序行为

 **理解软件自我保护技术**：深入了解常见的软件保护机制（如加壳、动态调试检测等）以及这些机制如何防止逆向分析和破解

 **学习破解软件的方法和原理**：通过对注册验证机制的分析与修改，理解破解软件的基本流程，如跳转逻辑修改、寄存器赋值修改等

 **加深对加壳与脱壳技术的认识**：通过实验，掌握加壳和脱壳的工作原理、操作流程，以及它们在软件保护与破解中的作用和对抗方式

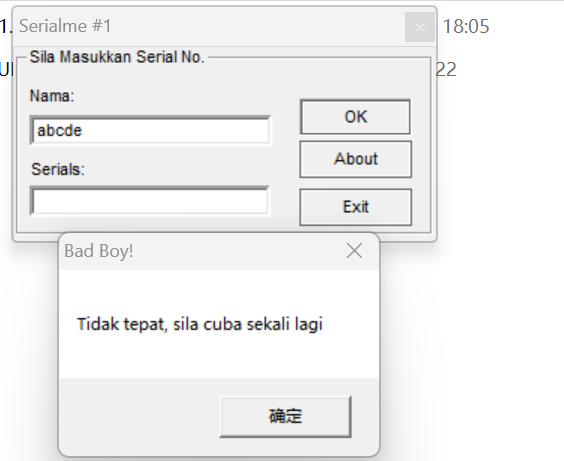
### 5.3实验步骤及内容

**第一阶段：软件保护破解**

**实验对象：crack.exe**

**爆破手段**

直接运行程序，注册，此时的结果如下图所示：

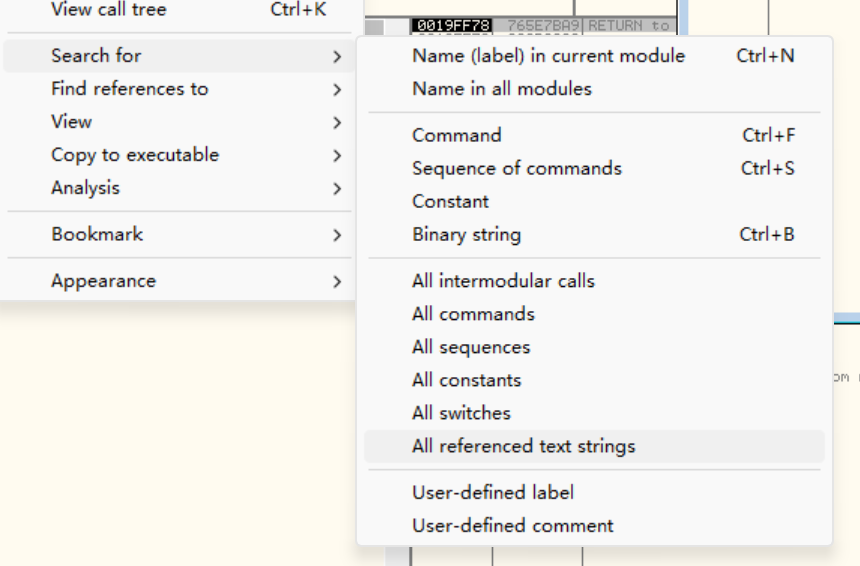


显示注册失败的标志

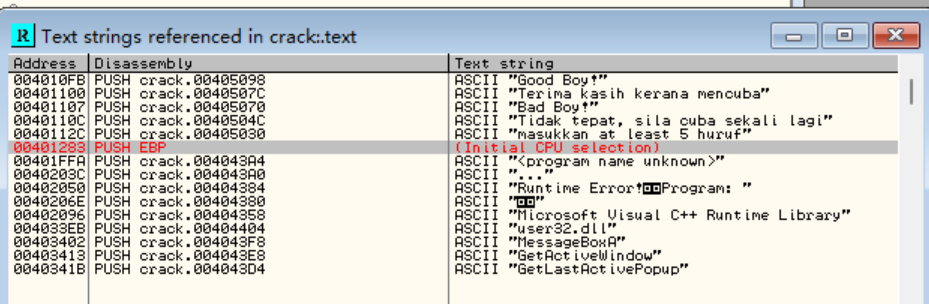
接下来我们使用OllyDbg和010editor对该程序进行修改，使得我们的注册能够成功

1. **查找显示注册结果相关代码**

首先在OllyDbg中打开程序，之后按照下图的方式点击运行，得到引用的字符串信息，这些信息就是注册结果的相关代码

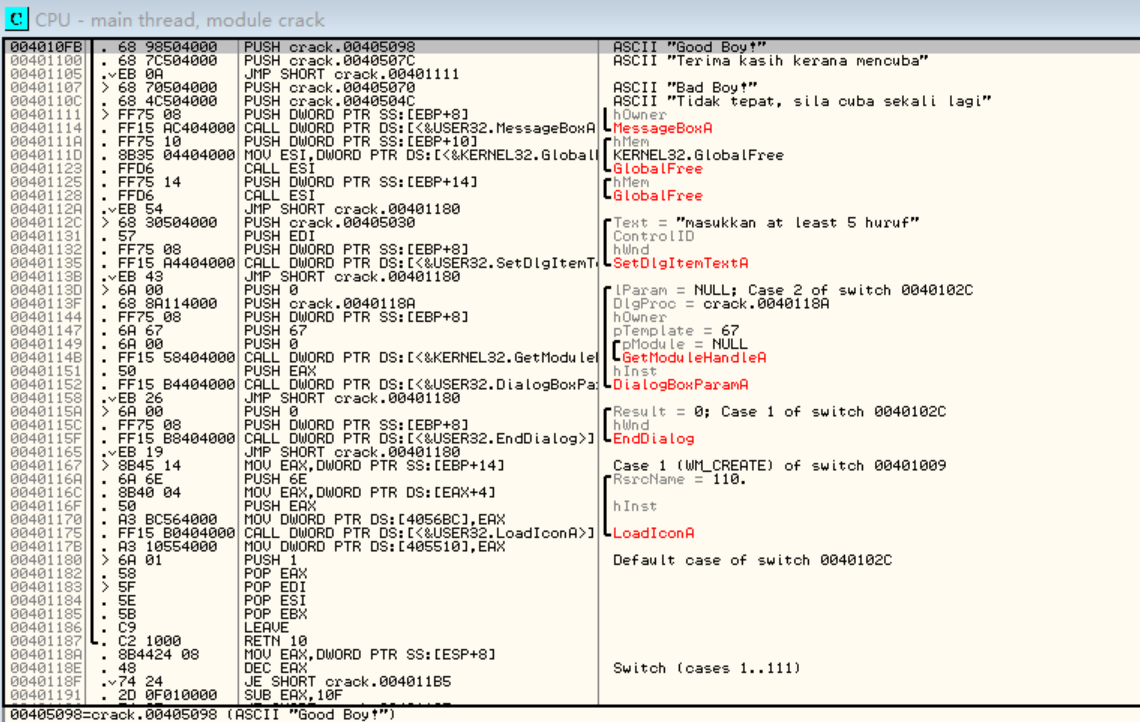


可以看到下图的注册相关字符串的信息



**（2）查找注册码验证相关代码**

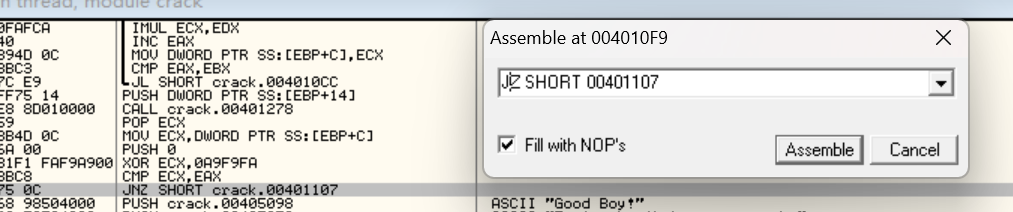
双击上图的字符串处即可跳转到相关代码区域



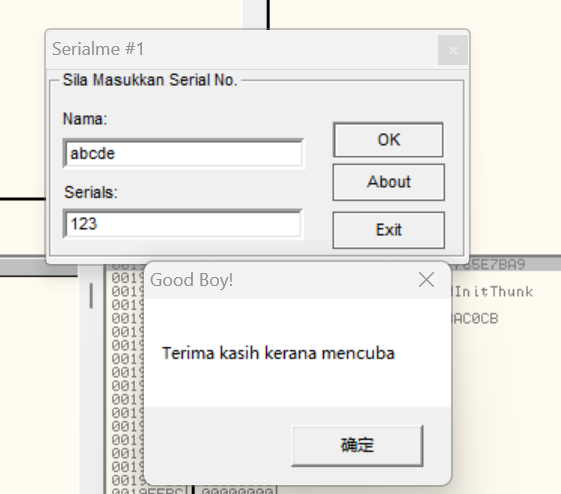
可以看到代码中的跳转信息，跳转信息是最重要的注册时的分支判断，它掌管了我们注册时的对话框中显示的信息

**（3）修改程序跳转**

我们根据函数的结构发现，只要将下图部分的跳转部分改为相反的判断条件，就可以绕过注册时的限制，最后注册成功



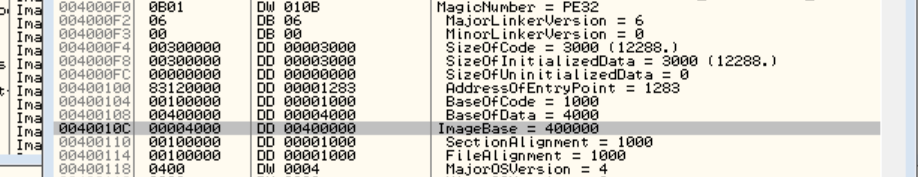
修改后的运行结果如下图所示



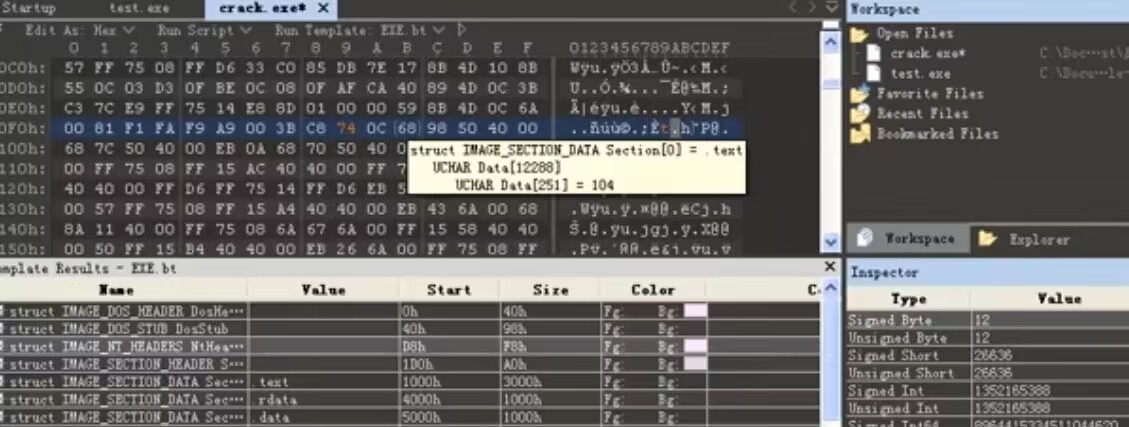
可以看到，此时的注册已经成功，我们的实验成功

还可以使用010editor对程序进行永久性的更改，这样就不需要动态运行程序时调整

首先获得程序在系统中的地址

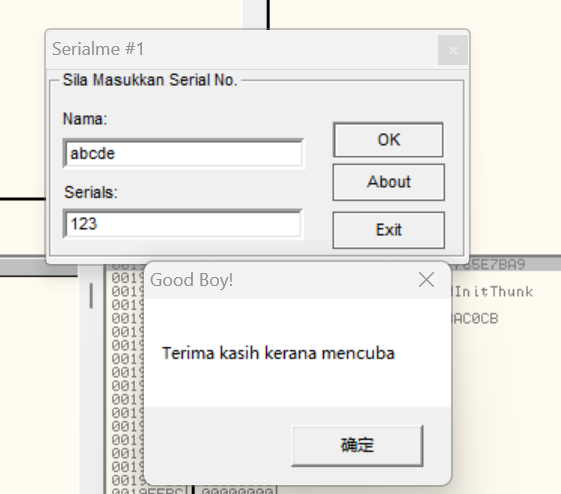


可以通过这个数值算出来这条指令在整个程序中的相对偏移，这样就可以在010editor中定位到此处



将如图所示的部分更改16进制数值（就是刚才的jz跳转指令）

这样就修改好了程序，不必每次都动态调整

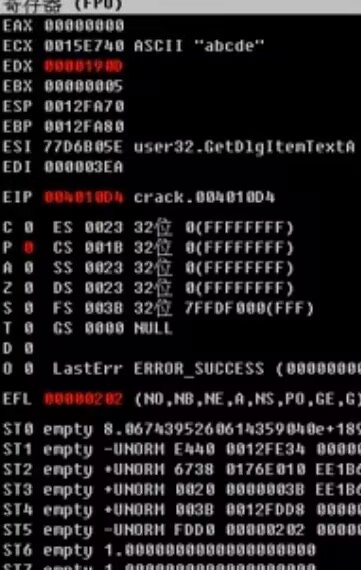


**编写注册机**

编写的大致流程如下图所示



在OllyDbg中的寄存器赋值部分跟踪查看的效果应该如下图所示：

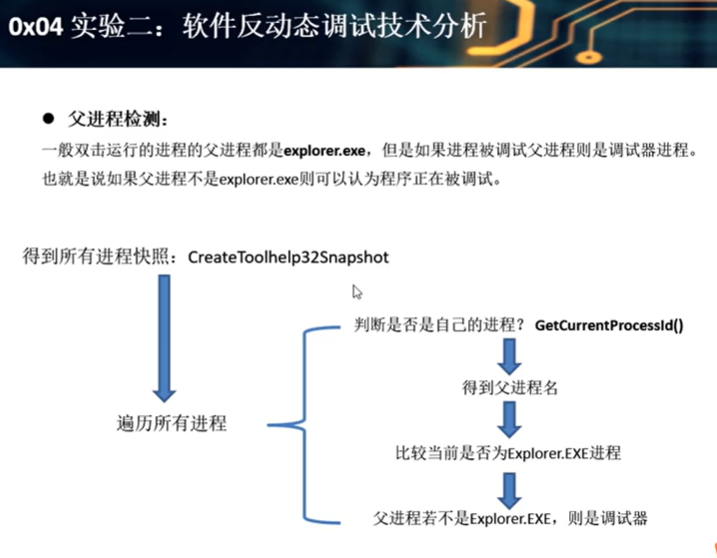


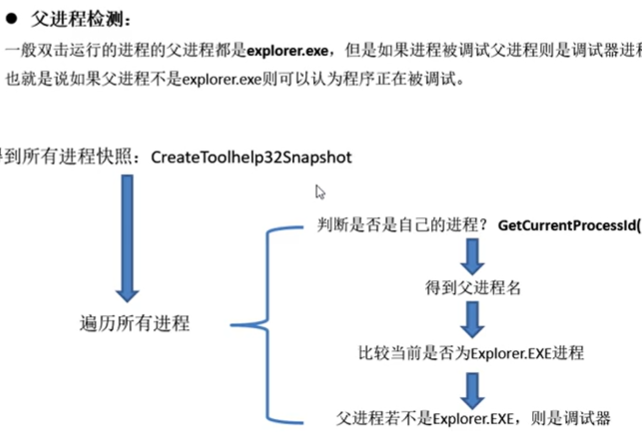
可以看到我们需要填充的部分都已经作为参数存在，可以在运行的时候自动传入

**第二阶段：软件反动态调试分析**

**实验对象：CrackMel.exe**

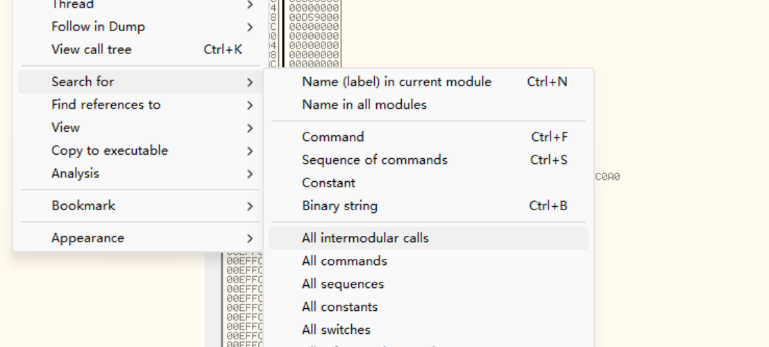
**（1）分析该实验对象通过父进程检测实现反OllyDbg调试的机理**



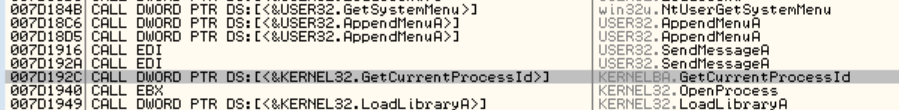


可以通过PPT了解到父进程检测的原理与大概流程

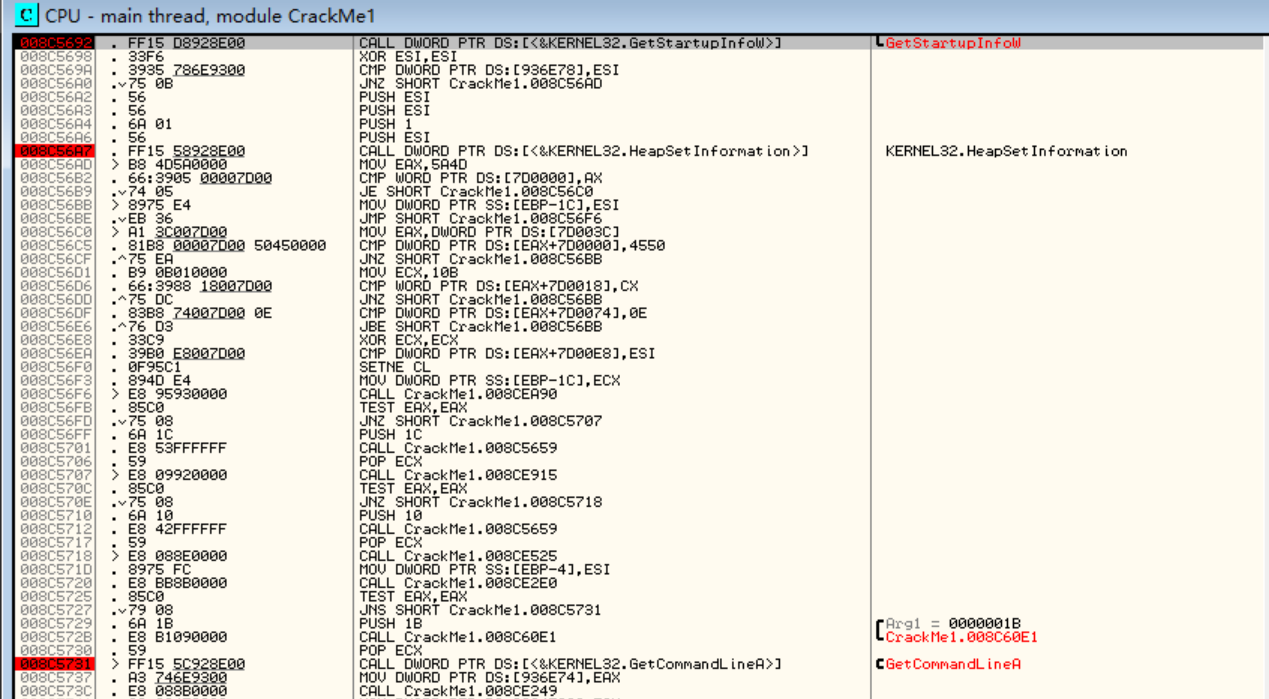
首先在OllyDbg中打开该实验对象



采用如图所示的选项进行所有进程调用的寻找



接着在这些进程之中找到如图所示的GetCurrentProcessId进程，对其下断点调试



结合如图所示的调试信息界面以及父进程检测的原理即可得知如何进行确定是否处于动态调试之中（即得到父进程名字进行比对）

**（2）分析该实验对象除了父进程检测之外的反动态调试技术**

可以结合视频讲解、并上网查找相关资料，这个程序中还存在对常见的动态调试软件名字的比对来确定当前是否处于一个可疑状态

**第三阶段：加壳与脱壳**

**实验对象：CrackmeUPX.exe**

1. **理解加壳与脱壳思路**

**加壳的核心思路**

加壳的目标是保护软件，使其难以被逆向工程或破解。具体思路包括：

* 隐藏核心代码： 将原始程序的代码加密、压缩或转换成难以阅读的形式。
* 插入保护机制： 添加动态校验、反调试、代码混淆等机制。
* 延迟还原： 在运行时解密或解压缩核心代码，原始代码直到运行时才加载到内存。
* 增加逆向难度： 利用复杂的算法或虚拟机保护，将核心逻辑转化为难以理解的伪代码。

**关键流程：**

1. 对可执行文件进行压缩或加密。
2. 在文件头插入壳代码，用于解压缩或解密。
3. 运行时由壳代码还原原始程序并跳转执行

**脱壳的核心思路**

脱壳的目标是去除壳代码，恢复原始程序的代码结构。具体思路包括：

* 静态分析： 对加壳后的程序进行文件分析，确定壳的类型和加壳方式。
* 动态调试： 运行程序并监控其运行过程，在解密或解压缩的瞬间截获原始代码。
* 识别还原点： 找到壳代码还原原始程序时的关键跳转点，获取解密后的完整程序。
* 重建程序： 将获取的内存中的原始代码提取并重新构建为可执行文件。

**关键流程：**

1. 使用工具（如IDA、OllyDbg）分析壳代码的结构和逻辑。
2. 动态调试程序，找到程序加载原始代码的时机。
3. 截取内存中的原始代码，导出并修复文件头
4. **尝试手动脱壳**

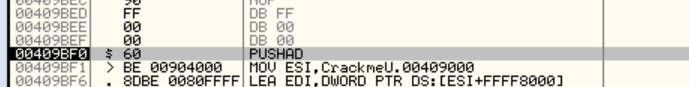
**实验对象：CrackmeUPX.exe**

首先使用PEID打开该实验对象

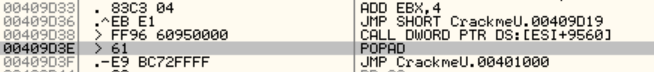


可以看到目前该实验对象是有壳保护的

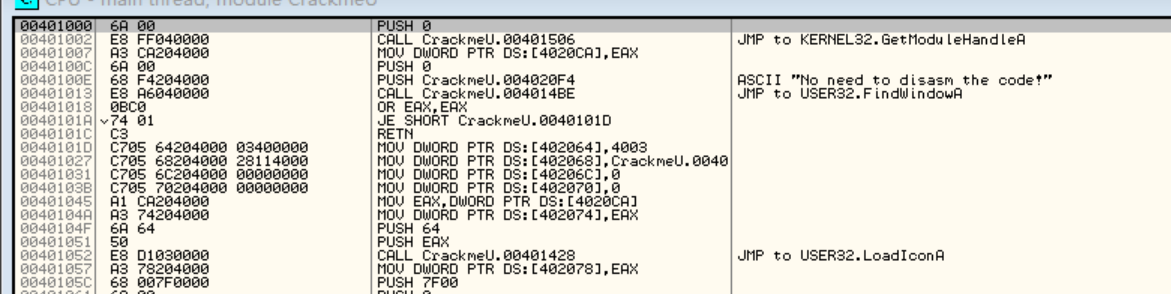
接下来在OllyDbg中打开



入口处的pushad指令是保存所有寄存器，那么必定有对应的popad指令



找到popad指令，发现下一条jmp指令极有可能是跳转到函数真正的入口地址，我们继续调试跟踪进入

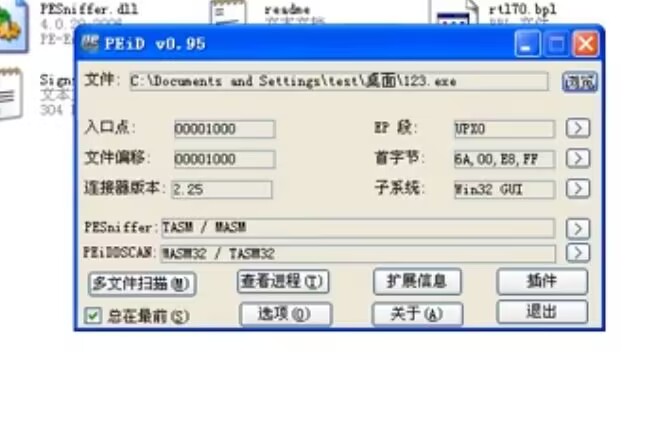


可以看到代码内容确实是真正的函数开始

接下来我们在OllyDbg的插件中去壳即可



将去壳后的程序再次使用PEID打开，可以看到壳已经脱去



到了这里，我们的手动脱壳就已经成功了

### 5.4 实验心得体会

在实际操作中巩固了理论知识，收获颇丰

1. **深入理解逆向工程的技术和思路**  
   实验让我初步掌握了逆向工程的基本流程，包括通过静态分析和动态调试工具（如OllyDbg）对程序进行分析，以及通过修改跳转逻辑和寄存器赋值来绕过程序的限制。实际操作过程中，感受到逆向工程是一项极具挑战性和逻辑性的工作，需要对程序运行原理有深入了解
2. **熟悉软件保护机制与破解方法**  
   在破解实验对象时，认识到现代软件保护技术的重要性。比如，通过父进程检测、调试器检测等方式，软件能够有效防止被调试和逆向分析。然而，这些保护机制也可以被分析和绕过。这种“保护与破解”的对抗让我更加深刻地理解了软件安全领域的复杂性
3. **加壳与脱壳的实践意义**  
   实验中的加壳与脱壳部分让我进一步理解了软件保护的技术实现原理。例如，加壳可以隐藏程序的核心逻辑，增加逆向分析的难度；而脱壳则需要动态调试并捕捉程序解密的瞬间。通过手动脱壳操作，学会了如何结合静态和动态分析技术，提取出程序的原始代码
4. **工具使用能力的提升**  
   在实验中，继续练习了OllyDbg和010editor的使用技巧，学会了如何通过断点调试、寄存器修改等方式调整程序运行逻辑，更加熟悉了这些简单却又实用的软件