《汇编语言程序设计实验》
实验四任务

实验名称：实验四 中断与反跟踪

（本实验使用 4+4 两次课内上机学时完成，实验报告在第二次上机完成后的次日提交。其中，任务 1~3 建议在第一个课内 4 学时内完成，任务 4，5 建议在第二个课内 4 学时内完成）

一、实验目的和要求

(1) 掌握中断矢量表的概念；
(2) 熟悉 I/O 访问，BIOS 功能调用方法；
(3) 掌握实方式下中断处理程序的编制与调试方法；
(4) 熟悉跟踪与反跟踪的技术；
(5) 提升对计算机系统的理解与分析能力。

二、实验内容

任务 1：用三种方式获取中断类型码 1H 、13H 对应的中断处理程序的入口地址。

要求：首先要进入虚拟机状态，然后

（1） 直接运行调试工具（TD.EXE），在其数据区观察中断矢量表中的信息。

（2） 编写程序，用 DOS 系统功能调用（具体调用方法见教材示例及附录中的描述）方式获取，观察功能调用相应的出口参数与“(1)”看到的结果是否相同 （使用 TD 观看出口参数即可）。

（3）编写程序，直接读取相应内存单元，观察读到的数据与“(1)”看到的结果是否相同（使用 TD 观看程序的执行结果即可）。

任务 2：编写一个接管键盘中断的中断服务程序并驻留内存，其主要功能是：在程序驻留并返回到 DOS 操作系统后，输入键盘上的大写字母时都变成了小写字母。

要求：

（1）在 DOS 虚拟机下执行程序，中断服务程序驻留内存。

（2）在 DOS 命令行下键入小写字母时，屏幕显示不变，键入大写时，屏幕显示为小写。执行 TD，在代码区输入指令“mov AX,0”，看是否都变成了小写。执行实验三任务 1 的程序，输入大小写是否正常？

（3）选作：另外单独编写一个中断服务程序的卸载程序，将自己驻留的键盘中断服务程序恢复到原来的状态（只需要还原中断矢量表的信息，先前驻留的程序可以不退出内存）。

任务 3：读取 CMOS 内指定单元的信息，按照 16 进制形式显示在屏幕上。

要求：

（1） 在数据段定义一个待读取的 CMOS 内部单元的地址编号。再使用 IN/OUT 指令，读取 CMOS 内的指定单元的信息。

（2） 将读取的信息用 16 进制的形式显示在屏幕上。若是时间信息，可以人工判断一下是否与操作系统显示的时间一致。

（以上任务尽量在第一次 4 个课内学时阶段完成，上机前实验报告应完成该任务中除实验记录与分析、总结与体会之外的内容）

上述任务中，需要解决的问题提示：

任务编号	序号	需要尝试的操作
任务 1	1	打开 TD 之后，如何在数据区切换到中断向量表所在内存区域
	2	如何计算某个中断入口在中断向量表内的偏移地址？
	3	程序中如何使用系统功能调用获取中断入口地址？可以在 TD 中录入指令语句或编写完整程序来尝试。
	4	程序中如何通过直接内存读取获取中断入口地址？可以在 TD 中录入指令语句或编写完整程序来尝试。
	5	用 TD 把中断矢量表里的中断矢量的值随意改成其他值（或改成其他中断的中断矢量）会有什么现象发生？（比如修改 21H,1H,3H 的中断矢量）
	6	对整个中断矢量表中的入口地址进行观察，是否能看出什么特点？
	7	选择几个中断服务程序的入口地址，观察其对应的中断服务程序的代码（比如：看一下 0 号、2 号中断的处理程序干了些什么？）
任务 2	1	有哪两种方式进入原中断服务程序？(CALL 和 JMP)
	2	为避免未调试好的中断服务程序接管键盘中断时使键盘操作失灵，可以先用其他方法（比如：先不安装，仅当作子程序调用来调试；安装到其他非硬件的中断号上，利用软中断来调试等）调试该中断服务程序，调试好后再安装成接管键盘中断的状态。请给出“其他”调试方法的具体描述并实施一下。
	3	编写的中断驻留程序执行后能否正常返回到 DOS？DOS 是否还能正常工作？如果重复驻留多次，会有什么现象？如何避免重复驻留？
	4	同时打开另外一个虚拟 DOS 窗口，键盘大小写是否被替代？
	5	如何确定自己编写的中断处理程序已被系统调用？（除了大写字母已经被替换成小写的途径之外）
	6	选作的要求(3)应该如何实现，如何找到保存的原中断入口地址？ 如何保证不会错误恢复？（比如，你的程序还没有驻留，但运行了恢复程序）
任务 3	1	如何直接在 TD 下使用 IN/OUT 指令获取 CMOS 数据？
	2	CMOS 里的时间信息是按照压缩 BCD 码的形式存放的，举例说明压缩 BCD 码的格式是什么？

附录 参考资料

1. 键盘扫描码

键盘上的每一个键都有两个唯一的数值进行标识。为什么要用两个数值而不是一个数值呢？这是因为一个键可以被按下，也可以被释放。当一个键按下时，它们产生一个唯一的数值，当一个键被释放时，它也会产生一个唯一的数值，我们把这些数值都保存在一张表里面，到时候通过查表就可以知道是哪一个键被敲击，并且可以知道它是被按下还是被释放了。这些数值在系统中被称为键盘扫描码。

因此，键盘上按下某个字母时，系统是可以得到该字母的扫描码及 ASCII 码的。我们在通过中断处理程序修改键盘字母的定义时，如果只替换 ASCII 码部分，会导致扫描码及 ASCII 码不配套的问题，有些程序对此敏感，就可能会产生错误。本次实验的基本要求是替换 ASCII 码。

2. BIOS 键盘服务 INT 16H

本实验用到 **INT 16H** 的功能号是 00H 和 10H
功能号：00H 和 10H
功能：从键盘读入字符
入口参数：AH
　　＝00H—读键盘

＝10H—读扩展键盘

出口参数： AH＝键盘的扫描码
AL＝字符的 ASCII 码

其他一些功能号包括：
功能号：01H 和 11H
功能：读取键盘状态
入口参数：AH ＝01H—检查普通键盘
＝11H—检查扩展键盘

出口参数：ZF＝1—无字符输入，
否则，AH＝键盘的扫描码，AL＝ASCII 码。
功能号：02H 和 12H
功能：读取键盘标志
入口参数：AH ＝02H—普通键盘的移位标志
＝12H—扩展键盘的移位标志

出口参数：AL＝键盘标志(02H 和 12H 都有效)，
其各位之值为 1 时的含义如下：
位 7—INS 开状态
位 3—ALT 键按下
位 6—CAPS LOCK 开状态
位 2—CTRL 键按下
位 5—NUM LOCK 开状态
位 1—左 SHIFT 键按下
位 4—SCROLL LOCK 开状态 位 0—右 SHIFT 键按下
AH＝扩展键盘的标志(12H 有效)，
其各位之值为 1 时的含义如下：
位 7—SysReq 键按下
位 3—右 ALT 键按下
位 6—CAPS LOCK 键按下
位 2—右 CTRL 键按下
位 5—NUM LOCK 键按下
位 1—左 ALT 键按下
位 4—SCROLL 键按下
位 0—左 CTRL 键按下
功能号：03H
功能：设置重复率
入口参数：AH＝03H 对于 PC/AT 和 PS/2： AL＝05H
BH＝重复延迟
BL＝重复率
对于 PCjr：AL＝00H—装入缺省的速率和延迟
＝01H—增加初始延迟
＝02H—重复频率降低一半
＝03H—增加延迟和降低一半重复频率

＝04H—关闭键盘重复功能

出口参数： 无
功能号：04H
功能：设置键盘点击
入口参数：AH ＝04H
AL ＝00H—关闭键盘点击功能
＝01H—打开键盘点击功能
出口参数： 无
功能号：05H
功能：字符及其扫描码进栈
入口参数：AH＝05H
CH＝字符的描述码
CL＝字符的 ASCII 码
出口参数： CF＝1—操作成功，AL＝00H，否则，AL＝01H

3. CMOS 简介

CMOS 是主板上一块可读写的 RAM 芯片。用途：主要用来保存当前系统的硬件配置和操作人员对某些参数的设定，维持基本的系统时钟状态。CMOS 芯片是由一块纽扣电池供电。因此在关机状态内部信息也不会丢失。

CMOS 拥有两个端口号分别是 70h 和 71h

端口号	权限	长度	作用
70h	不可读可写	8bit	用它来设置 CMOS 中的数据地址，表明准备访问芯片中的那个字节
71h	可读可写	8bit	用它来读写由 70h 端口设定的芯片内部的某个字节单元

CMOS 中的数据地址对照表。(供编程时查)

地址	数据	备注
00H	Time - Seconds	硬件时间的秒
01H	Alarm - Seconds	
02H	Time - Minutes	硬件时间的分
03H	Alarm - Minutes	
04H	Time - Hours	硬件时间的时
05H	Alarm - Hours	
06H	Date - Day of the week	
07H	Date - Day	
08H	Date - Month	
09H	Date - Year 29h	

0AH	Status Register A	
0BH	Status Register B	
0CH	Status Register C	
0DH	Status Register D	
0EH	Diagnostic Status	
0FH	Shutdown Status	
10H	A:	
11H	Reserved	
12H	0	
13H	Reserved	
14H	Equipment Installed	
15H	Base Memory (high byte)	
16H	Base memory (low byte)	
17H	Extended Memory (high byte)	
18H	Extended Memory (low byte)	
19H	0 (C:) Hard Disk Type	
1AH	1 (D:) Hard Disk Type	
1BH	Reserved	
1CH	Supervisor Password	
1DH	Supervisor Password	
1EH ~ 2DH	Reserved	
2EH	CMOS Checksum (high byte)	
2FH	CMOS Checksum (low byte)	
30H	Extended Memory (high byte)	
31H	Extended Memory (low byte)	
32H	Date - Century	
33H	Power On Status	
34H～3FH	Reserved	
40H～5FH	Extended CMOS	
60H	User Password	
61H	User Password	
62H～7FH	Extended CMOS	

任务 4：数据加密与反跟踪

在实验三任务 1 的**网店商品信息管理程序**的基础上，增加输入用户名和密码时，最大错误次数的限制，即，当输入错误次数达到三次时，直接按照未登录状态进入后续功能。老板的密码采用密文的方式存放在数据段中，各种商品的进货价也以密文方式存放在数据段中。加密方法自选（但不应选择复杂的加密算法）。

可以采用计时、中断矢量表检查、堆栈检查、间接寻址等反跟踪方法中的几种方法组合起来进行反跟踪（建议采用两种反跟踪方法，重点是深入理解和运用好所选择的反跟踪方法）。

为简化录入和处理的工作量，只需要定义三种商品的信息即可。

提示：为了使源程序的数据段中定义的密码、进货价等在汇编之后变成密文（也就是在最后交付出去的执行程序中看不到明文），可以使用数值运算符（参见教材 P48）对变量的初始值进行变换。例如，如果想使进货价 50 变成密文，加密算法是与老板密码中的字符“W”做异或运算，则可写成：

```
DB  50  XOR  'W'
```

任务 5：跟踪与数据解密

解密同组同学的加密程序，获取各个商品的进货价。

注意：两人一组，每人实现一套自己选择的加密与反跟踪方法，把执行程序交给对方解密（解密时间超过半小时的，说明反跟踪方法基本有效）。如何设计反跟踪程序以及如何跟踪破解，是本次实验报告中重点需要突出的内容。（分组可以按照学号顺序或座位次序依次构成两人一组。也可以自行调整。如果班上人数是奇数，则三人一组，甲解密乙的，乙解密丙的，丙解密甲的）

（以上任务 4 和 5 尽量在第二次 4 个课内学时阶段完成，上机前实验报告应完成该任务中除实验记录与分析、总结与体会之外的内容）

上述任务 4 和 5 中，需要解决的问题提示：

1. 若密码是用明文存放在数据段中的，如何更快地获取密码？
2. 若商品进货价是用明文存放在数据段中的，如何更快地获取进货价？（除了用调试工具在内存中去看，还可以将执行程序文件用二进制编辑工具打开，直接在文件里寻找所定义的商品信息）
3. 如何对密码实现快速的暴力破解？（描述一下实现的具体思路即可）
4. 如何综合利用静态反汇编和动态反汇编的信息破解程序？（在分析里总结一下即可）
5. 举例说明如何观察到程序中存在反跟踪的代码?举例说明如何应对反跟踪程序？（在分析里具体描述）
6. 思考一下，如何用 C 语言（不嵌入汇编语言）实现反跟踪？是否能发现汇编语言的特殊之处？（在体会里具体描述）
7. 当存在修改中断矢量表的代码时，一般会先关掉中断（也即执行 CLI 指令）。如果不想因为关中断指令的出现让跟踪者容易判断出后续存在反跟踪代码，应如何设计修改中断矢量表的代码，达到不用关中断的目的？
8. 是否可以通过修改 AUTH 的值来达到获取进货价的目的？是否可以通过观察该程序计算推荐度的过程来获取进货价？
9. 若将接管键盘中断的方法用到反跟踪中，是否有好处？（输入密码前接管键盘中断，密

码输入完毕后恢复原中断矢量。把密码字符串从明文转换成密文的转换算法写到接管后的键盘中断服务程序中，转换后的结果直接存在指定变量中，但不用破坏原键盘中断的返回值）

参考资料：

关于跟踪与反跟踪、加密与解密已经有了很系统的理论与方法，其完整、深入的知识点不是本课程的内容，本次实验仅仅是以加解密为背景，目的是初步体会汇编语言在此领域中的作用。下面介绍的内容对大家实现本次实验的目的有一定的帮助作用。

一、 反汇编、跟踪与密码破解方法简介

1、暴力破解密码

（1）对于密码以明文方式存放在待破解程序中的程序，可以利用文件编辑工具直接打开该待破解程序的文件并查看里面的内容，找到疑似密码的字符串，再通过实际测试，最终确定真实的密码。

（2）对于仅仅需要输入密码且没有出错次数限制的程序，可以采用暴力破解密码的方法来获取密码，也即编写一个程序，自动调用与执行待破解的程序，并按照枚举方式自动给待破解程序输入可能的密码，直至密码正确为止。

2、静态反汇编

利用反汇编工具直接处理待破解程序的执行文件，将执行文件翻译成比较直观易读的汇编语言的源程序（甚至 C 语言的源程序），再由人工阅读该源程序，可以找出加密算法（从而推出真实的密码），或者找到“密码检查通过之后”的程序入口，再用二进制文件编辑工具把执行文件中要求输入密码的代码直接改掉，改成直接跳转到密码比较正确之后的程序入口之处（当然需要事先知道转移指令的机器码）。

静态反汇编方法对于未加入反跟踪代码的待破解程序比较有效，在逆向工程中常用。若静态反汇编得到的源程序正确无误，则可以直接修改源程序，然后再汇编成执行程序，就可以得到我们希望的新程序了。

3、动态调试跟踪

单步执行待破解的程序，是可以很直观地获得程序的执行流程和完成的操作的。若碰到反跟踪程序段，则需要配合设置断点的方法，绕过反跟踪程序段设置的陷阱。

4、自动跟踪

为了避免人工跟踪时易被反跟踪以及时间较长的问题，可以通过自动跟踪工具进行跟踪。自动跟踪工具可以是类似 TD 的一个程序，只是不再有显示界面，而是自动把每次单步执行时获取的指令语句等信息按照反汇编的源程序的形式保存到文件中，直至待破解程序退出。然后，人工阅读所保存文件里的汇编语言指令语句序列，分析出程序的流程及算法，再推出密码。这种方法由于仍然使用了单步中断，所以，不适合于采用了堆栈检查或中断矢量表检查的反跟踪措施的程序的破解。

自动跟踪工具也可以采用类似虚拟机的方法实现。待破解程序被自动跟踪工具调入内存后被自动跟踪工具解释执行，在解释的过程中记录待破解程序的指令语句序列并保存到文件中，供后续分析之用。这种实现方法可以避免计时、堆栈检查、中断矢量表检查等绝大多数的反跟踪措施的影响。

在实际破解过程中，往往会将多种方法交叉使用，以利于更快地得到结果。

二、 反跟踪方法简介

1、采用间接转移抵制静态反汇编

无论是 JMP 指令还是 CALL 指令，都支持间接转移/调用，尤其是可以使用寄存器寻址方式实现间接转移/调用。由于寄存器的内容只有在程序执行之后才能确定具体的值，因此，这种方法可以阻止静态反汇编程序获取程序模块之间的调用关系。当然，这种方法也会让程序的可读性降低，因此，它也加大了人工动态调试跟踪时理解程序的难度。

与间接转移指令配套的措施是建立地址表。地址表可以保存任何标号或子程序的地址，当程序要转移到某个标号或子程序处执行时，可以先将地址表中保存的地址值送到某个寄存器中，再用 JMP 或 CALL 去转移(例如，在 16 位段中，段内转移用字寄存器 BX 保存地址值，用 JMP BX 或 CALL BX 进行转移)。为了扰乱视线，可以在 JMP 或 CALL 指令之后定义一些初始值随意的变量存储区，由于在程序正常控制下是不会执行到这段存储区的，所以，反汇编程序把这段存储区的数据反汇编成指令语句是没有任何意义的，这就达到了扰乱视线的目的。

地址表可以是静态的，也就是在定义地址表的时候就已经初始化好了对应的地址值。但地址表也可以是动态的，也就是只有在程序运行之后，通过程序赋值或修改之后才形成正确的地址值（对应地址值等数据需要程序动态赋值或修改后才能使用的做法，代码区也是可以动态生成的，也即通过执行一段程序后才能把后续要执行的程序代码恢复出来。这就可以有效地抵制静态反汇编工具了）。

2、通过计时的方法来抵制动态调试跟踪

人工单步调试时的指令执行间隔很容易达到秒级，而连续执行的程序中的指令执行间隔一般不会达到毫秒级（即使中间有系统中断打断），因此，通过计算指定的几条指令执行所花费的时间，就可以发现该程序是否被调试。

比较简单的计时方法是在执行指定指令前获取一下当前时间，然后执行这几条指令，之后再次获取当前时间，通过计算这两个时间的差值，即可判断出是否被跟踪。常用的计时程序的精度是 55ms，若考虑一下干扰因素，则时间的差值为 0 或 55ms 就属于未被跟踪的正常执行状态。

3、通过检查堆栈来抵制动态调试跟踪

动态调试过程中会产生单步中断，该中断响应过程会使用被调试程序的堆栈，也就是说被调试程序栈顶以上几个字存储区的内容会被中断响应过程修改。因此，只要预先在程序中执行压栈和出栈操作，即可设定栈顶以上几个字的内容，再获取栈顶以上几个字的内容，判断它们是否是先前压栈时的内容，即可判断是否被调试（当然，为了避免硬件中断的干扰，这段程序段执行前需要关闭可屏蔽中断，也就是要执行 CLI 指令，等这段程序执行完之后再开中断，也即执行 STI 指令）。

4、通过检查中断矢量表来抵制动态调试跟踪

动态调试工具都会接管系统原来的单步中断和断点中断的中断服务程序。如果能判断中断矢量表中 1 号和 3 号中断处理程序的入口地址被修改到了缺省地址以外的区域，即可判断出有调试工具在运行，则本程序退出。

如果无法判断是否有调试工具在运行，也可以在本程序中直接修改中断矢量表中 1 号和 3 号中断服务程序的入口地址，让它们都指向本程序中的一段代码（比如只是一条 IRET 指令）。有的调试工具会阻止你的程序修改中断矢量表，这时，你只需要在修改之后再去判断一下中断矢量表中对应位置是否改成了你的程序的入口地址，即可判断是否存在调试工具了（当然，你也可以把你的中断处理程序写成一段有用的代码，你是需要调用它的；若调试工具阻止你修改该中断矢量，当你去调用该中断时就不会达到你的目的，程序自然会出错）。

为了避免反跟踪程序段被跟踪者跳过，一般会通过增加大量的无关程序段来扰乱视线，也可以把有效工作程序段穿插到反跟踪程序中（比如，把密码串转换的程序段放到计时子程序中，这样，如果没有执行计时子程序的功能，也就少了密码串转换的功能）。

三、 加密方法简介

1、采用不公开的算法加密

当用户不用知道密码，但系统内的信息要做加密处理时，可以选用一种算法或函数表达式对数据进行编码，使得保存或传输的数据是编码后的密文数据。简单的编码方法包括对数据进行移位、求补、高低位交换等。

2、算法公开，密钥不公开

算法复杂度足够高，必须知道密码才能解密。例如，对称加密体系 DES。

3、算法和密钥根据需要隐藏与公开

为了提高算法复杂性不高的加解密方法的安全性，可以不公开加解密算法和密钥。但对于高复杂性的加解密方法，例如公钥加密体系 RSA，是可以公布算法及公钥，而只保留私钥的。

本次实验不要实现算法复杂度高的加密算法，应选用单纯的运算方法或简单的函数算法即可（应把精力放在如何反跟踪上）。

四、 程序举例

下面用一个学生成绩查询程序来简单地示意加密和反跟踪方法的应用情况（其中小写的指令语句属于反跟踪的程序段，颜色相同的部分属于同一种反跟踪方法）。学生在编写本实验的程序时可以只用其中一部分方法，而把主要精力放在灵活地应用这些方法上（比如把程序的功能与反跟踪程序更好地糅合在一起）。

```
.386
STACK SEGMENT USE16 STACK
      DB 200 DUP(0)
STACK ENDS
;
DATA SEGMENT USE16
D1    DB 0DH,0AH,'STUDENT NAME:$'
D2    DB 0DH,0AH,'CHN SCORE:$'
D3    DB 0DH,0AH,'MATH SCORE:$'
D4    DB 0DH,0AH,'ENG SCORE:$'
RADX DB 10
;以下对成绩表信息进行了简单的加密
BUF   DB  'z' XOR  'B','h'XOR 'a','a'XOR 't', 7 DUP(0)      ;学生名字为“zha”，采用与密码串
                                                            ;依次异或的方式加密
      DB     100 XOR 'B' ,85 XOR 'a', 80 XOR 't', ?         ;学生的成绩也依次与密码串异或。
      DB  'l' XOR  'B','i'XOR 'a','s'XOR 't','i' XOR 'B',6 DUP(0)
      DB     80 XOR 'B', 98 XOR 'a', 70 XOR 't',?
;以下对密码进行了简单的加密
PWD   DB  3 XOR 'C'          ;密码串的长度为 3，采用与常数 43H 异或的方式编码成密文
      DB  ('B' -29H)*3       ;真实密码为 Bat。采用函数(X-29H)*3 对保存的密码进行编码。
      DB  ('a' -29H)*3
      DB  ('t' -29H)*3
      DB  0A1H,5FH,0D3H      ;用随机数填充密码区到 6 个字符，防止破解者猜到密码长度
;
IN_PWD  DB 7                 ;使用者输入的密码区，最大长度 6 个字符
        DB ?
        DB 7 DUP(0)
STR1 DB 0DH,0AH,'PLEASE ENTER PASSWORD:$'
;
P1      DW  PASS1            ;地址表（用于间接转移反跟踪）
E1      DW  OVER
P2      DW  PASS2
OLDINT1 DW  0,0              ;1 号中断的原中断矢量（用于中断矢量表反跟踪）
OLDINT3 DW  0,0              ;3 号中断的原中断矢量
DATA ENDS
;
```

```
CODE SEGMENT USE16
     ASSUME CS:CODE,DS:DATA,SS:STACK
START: MOV AX,DATA
       MOV DS,AX
       xor  ax,ax                    ;接管调试用中断，中断矢量表反跟踪
       mov  es,ax
       mov  ax,es:[1*4]              ;保存原 1 号和 3 号中断矢量
       mov  OLDINT1,ax
       mov  ax,es:[1*4+2]
       mov  OLDINT1+2,ax
       mov  ax,es:[3*4]
       mov  OLDINT3,ax
       mov  ax,es:[3*4+2]
       mov  OLDINT3+2,ax
       cli                             ;设置新的中断矢量
       mov  ax,OFFSET NEWINT
       mov  es:[1*4],ax
       mov  es:[1*4+2],cs
       mov  es:[3*4],ax
       mov  es:[3*4+2],cs
       sti
       LEA DX,STR1
       MOV AH,9
       INT 21H
       LEA DX,IN_PWD                 ;输入密码字符串
       MOV AH,10
       INT 21H
         cli                         ;计时反跟踪开始
         mov  ah,2ch
         int  21h
         push dx                     ;保存获取的秒和百分秒
       MOV CL,IN_PWD+1               ;比较输入的串长与密码长度是否一样
       XOR CL,'C'
       SUB CL,PWD
       MOVSX  BX,CL
       ADD  BX,OFFSET P1
         mov  ah,2ch                   ;获取第二次秒与百分秒
         int  21h
         sti
         cmp  dx,[esp]                 ;计时是否相同
         pop  dx
         jz   OK1                      ;如果计时相同，通过本次计时反跟踪
         mov  bx,offset E1             ;如果计时不同，则把转移地址偏离 P1
OK1:     mov  bx,[bx]
         cmp  word ptr cs:[bx],0B60FH    ;是否是 PASS1 处的指令，其实是用于判断前面比较的
                                         ;串长是否相同
         jz   OK2
         jmp   E1
OK2:     jmp   bx
         db  'How to go'               ;定义的冗余信息，扰乱视线

PASS1: MOVZX CX,IN_PWD+1
          cli                          ;堆栈检查反跟踪
          push  P2                     ;PASS2 的地址压栈
       MOV  SI,0
       MOV  DL,3
          pop  ax
          mov  bx,[esp-2]              ;把栈顶上面的字（PASS2 的地址）取到
          sti
          jmp  bx                      ;如果被跟踪，将不会转移到 PASS2
          db  'i do not know!'
```

```
PASS2:  MOVZX   AX,IN_PWD+2[SI]     ;比较密码是否相同。把输入的串变成密文，与保存的密文比较
        SUB   AX,29H
        MUL   DL
        CMP   AL,PWD+1[SI]
        JNZ   ERR2
        INC   SI
        LOOP PASS2
        JMP   PASS3
ERR2:   mov   ebx,OFFSET P1          ;当密码不等时，通过地址表计算出 OVER（退出）的位置
        mov   edx,1
        jmp   word ptr [ebx+edx*2]   ;指向 OVER
        db   'YES,get it'
;
PASS3: mov   bx,es:[1*4]             ;检查中断矢量表是否被调试工具阻止修改或恢复
       inc   bx
       jmp   bx                  ;正常修改了的化，这里将转移到 TESTINT，否则就不知道转到哪了
       db    'Now,you see.'
;
PASS4: LEA DX,D1                 ;进入实际功能区。这里仅实现显示 BUF 区第 2 个学生的姓名和成绩
       MOV AH,9
       INT 21H
       MOV SI,0
NEXT:  MOV DL,BUF+14[SI]
       CMP DL,0
       JE  SCORE
       MOV AX,SI
       DIV IN_PWD+1              ;求密码长度的模数（因为名字长度可能超过密码长度，
                                  ;取模之后可以保证循环使用密码串）
       MOVZX DI,AH
       XOR DL,IN_PWD+2[DI]       ;注意，使用用户输入的明文密码串进行解密，不仅不用去解密
                                  ;程序中的密码，而且可以抵抗跟踪者跳过密码判断过程直接
                                  ;转移到实际功能区后获取密文信息的做法
       MOV AH,2
       INT 21H
       INC SI
       JMP NEXT
SCORE: LEA DX,D2                 ;显示三科成绩
       MOV AH,9
       INT 21H
       MOV SI,0
       MOV AL,BUF+14+10[SI]
       XOR AL,IN_PWD+2[SI]
       MOV AH,0
       DIV RADX                   ;这里假设成绩最大为 99，所以只除一次 10
       ADD AX,3030H
       PUSH AX
       MOV DL,AL
       MOV AH,2
       INT 21H
       POP AX
       MOV DL,AH
       MOV AH,2
       INT 21H
       INC SI
;
       LEA DX,D3
       MOV AH,9
       INT 21H
       MOV AL,BUF+14+10[SI]
       XOR AL,IN_PWD+2[SI]
       MOV AH,0
```

```
        DIV RADX
        ADD AX,3030H
        PUSH AX
        MOV DL,AL
        MOV AH,2
        INT 21H
        POP AX
        MOV DL,AH
        MOV AH,2
        INT 21H
        INC SI
;
        LEA DX,D4
        MOV AH,9
        INT 21H
        MOV AL,BUF+14+10[SI]
        XOR AL,IN_PWD+2[SI]
        MOV AH,0
        DIV RADX
        ADD AX,3030H
        PUSH AX
        MOV DL,AL
        MOV AH,2
        INT 21H
        POP AX
        MOV DL,AH
        MOV AH,2
        INT 21H
        ;.........注意，本实验中，要把输入的密码转换成密文后与数据段中的密码比较，
         ;.........也就是用密文比较。进货价要用与老板密码相关的方法加密。
        ;..........计算推荐度时需要解密进货价，但不要把数据段中的密文数据给覆盖掉了。
        JMP OVER
;
NEWINT: iret
TESTINT: jmp PASS4
;
OVER:
        cli                             ;还原中断矢量
        mov  ax,OLDINT1
        mov  es:[1*4],ax
        mov  ax,OLDINT1+2
        mov  es:[1*4+2],ax
        mov  ax,OLDINT3
        mov  es:[3*4],ax
        mov  ax,OLDINT3+2
        mov  es:[3*4+2],ax
        sti
        MOV AH,4CH
        INT 21H
CODE    ENDS
        END START
```