

**课 程 实 验 报 告**

**课程名称： 汇编语言程序设计实验**

**实验名称： 实验四 中断与反追踪**

**实验时间： 2019-4-22，18:30-21:50 实验地点： 南一楼803室95号实验台**

**指导教师： 许向阳**

**专业班级：计算机科学与技术201703班**

**学 号： U201714609 姓 名： 李慧骏**

**同组学生： 无 报告日期： 2019年 4 月 23日**

**原创性声明**

  本人郑重声明：本报告的内容由本人独立完成，有关观点、方法、数据和文献等的引用已经在文中指出。除文中已经注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品或成果，不存在剽窃、抄袭行为。

特此声明！

学生签名：

日期：2019.4.17

成绩评定

|  |  |  |
| --- | --- | --- |
| 实验完成质量得分（70分）（实验步骤清晰详细深入，实验记录真实完整等） | 报告撰写质量得分（30分）（报告规范、完整、通顺、详实等） | 总成绩（100分） |
|  |  |  |

指导教师签字：

                    日期：

目录

[**1 实验目的和要求 1**](#_Toc8068420)

[**2 实验内容 1**](#_Toc8068425)

[**3 实验过程 9**](#_Toc8068464)

[任务1 9](#_Toc8068465)

[3.1.1 设计思想与存储单元分配 9](#_Toc8068466)

[3.1.2 源程序 9](#_Toc8068467)

[3.1.3 实验记录 10](#_Toc8068468)

[3.1.4 思考题 12](#_Toc8068469)

[任务2 13](#_Toc8068470)

[3.1.1 设计思想与存储分配 13](#_Toc8068471)

[3.2.2 源代码 13](#_Toc8068472)

[3.1.3 程序测试 14](#_Toc8068473)

[3.2.4 思考题 15](#_Toc8068474)

[任务3 15](#_Toc8068475)

[3.3.1 源程序 15](#_Toc8068476)

[3.3.2 程序测试 19](#_Toc8068477)

[3.3.3 思考题 19](#_Toc8068478)

[任务4 20](#_Toc8068479)

[3.4.1 源代码 20](#_Toc8068480)

[3.4.2 程序测试 36](#_Toc8068481)

[任务5 38](#_Toc8068482)

[3.5.1 解密方法 38](#_Toc8068483)

[3.1.2 思考题 38](#_Toc8068484)

[4 四．总结与体会 39](#_Toc8068485)

# 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。

# 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，将键盘的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 |  |
|  |  |  |

# 实验过程

## 任务1

### 设计思想与存储单元分配

获取中断类型码10H对应的中断处理程序入口地址有如下三种方法：

1. 在TD中直接观察。
2. 35H调用，取AL=10H为入口参数。
3. 将主存中的DS:[40H]的IP与DS:[42H]的CS赋值给BX与CX寄存器。

### 源程序

程序名 sy4a.asm

.386

CODE SEGMENT USE16

ASSUME CS:CODE, DS:CODE

OLD\_INT DW ?,?

START: MOV AX,3513H ；此为查看13H若要查看1H改为MOV AX,3501H

INT 21H

MOV OLD\_INT,BX

MOV OLD\_INT+2,ES

MOV AH,4CH

INT 21H

CODE ENDS

END START

程序名 sy4b.asm

.386

CODE SEGMENT USE16

ASSUME CS:CODE, DS:CODE

OLD\_INT DW ?,?

START: XOR AX,AX

MOV DS,AX

MOV AX,DS:[13H\*4] ；此为查看13H，若要查看1H，将代码中所有13H改为1H即可

MOV OLD\_INT,AX

MOV AX,DS:[13H\*4+2]

MOV OLD\_INT+2,AX

MOV ES,WORD PTR OLD\_INT+2

MOV BX,WORD PTR OLD\_INT

MOV AH,4CH

INT 21H

CODE ENDS

END START

### 实验记录

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）。

1H：

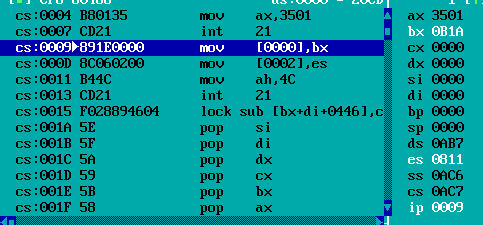


图3-1 1H出口参数

13H：

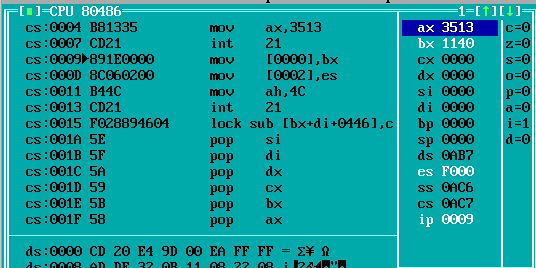


图3-2 13H出口参数

（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）。

1H：

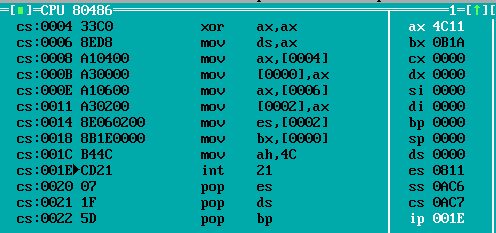


图3-3 1H执行结果

13H：

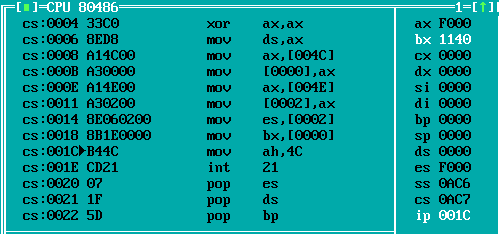


图3-4 13H执行结果

### 思考题

1. 如何计算某个中断入口在中断向量表内的偏移地址？

若取xH号中断，偏移地址为xH\*4，段地址为xH\*4+2。

2. 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。

可以运用语句 MOV AH,35H MOV AL,1H获得。

3. 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。

直接将段地址和偏移地址存入寄存器中进行跳转操作。

4. 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）

程序可能会崩溃或者卡住。

## 任务2

### 设计思想与存储分配

判断输入的字符是否是a~z或A~Z，若是则修改其ASCLL值，不是则判断下一个字符。利用AS寄存器来保存输入的字符串。

### 3.2.2 源代码

程序名 sy4c.asm

.386

DATA SEGMENT USE16

DATA ENDS

CODE SEGMENT USE16

ASSUME CS:CODE,SS:STACK

OLD\_INT DW ?,?

NEW16H: CMP AH,0H

JE LTM

CMP AH,10H

JE LTM

JMP DWORD PTR OLD\_INT

LTM:

PUSHF

CALL DWORD PTR OLD\_INT

CMP AL,40H; CMP >A

JS Q

CMP AL,5BH;CMP<=Z

JNS GO1

ADD AX,32;变大写

JMP Q

GO1: CMP AL,60H

JS Q

CMP AL,7BH

JNS Q

SUB AX,32

Q: IRET

START:

MOV AX,0H

MOV DS,AX

MOV AX,DS:[16H\*4]

MOV OLD\_INT,AX

MOV AX,DS:[16H\*4+2]

MOV OLD\_INT+2,AX

CLI

MOV WORD PTR DS:[16H\*4],OFFSET NEW16H

MOV DS:[16H\*4+2],CS

STI

MOV DX,OFFSET START+15

SHR DX,4

ADD DX,10H

MOV AL,0

MOV AH,31H

INT 21H

CODE ENDS

STACK SEGMENT USE16

DB 200 DUP(0)

STACK ENDS

END START

### 3.1.3 程序测试

输入时键盘为小写，在DOS运行sy4c.exe后，键盘输入的字符显示为大写如图3-5所示。

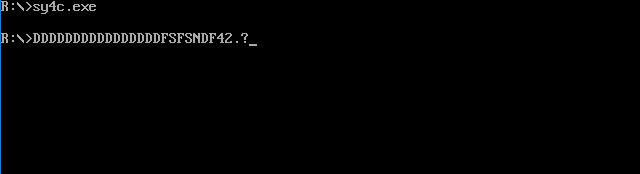


图3-5 程序运行测试

### 3.2.4 思考题

1. 有哪两种方式进入原中断服务程序？

CALL&JMP指令。

2. 同时打开另外一个虚拟DOS窗口，键盘大小写是否被替代？

不会被替代。

3. 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）

在TD中观察16H内的值的变化。

## 任务3

### 3.3.1 源程序

程序名 sy4d

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

CODE SEGMENT USE16

ASSUME CS:CODE, DS:CODE,SS:STACK

COUNT DB 18

HOUR DB ?,?,':'

MIN DB ?,?,':'

SEC DB ?,?

BUF\_LEN=$-HOUR

CURSOR DW ?

OLD\_INT DW ?,?

NEW08H PROC FAR

PUSHF

CALL DWORD PTR CS:OLD\_INT

DEC CS:COUNT

JZ DISP

IRET

DISP: MOV CS:COUNT,18

STI

PUSHA

PUSH DS

PUSH ES

MOV AX,CS

MOV DS,AX

MOV ES,AX

CALL GET\_TIME

MOV BH,0

MOV AH,3

INT 10H

MOV CURSOR,DX

MOV BP,OFFSET HOUR

MOV BH,0

MOV DH,0

MOV DL,79-BUF\_LEN

MOV BL,07H

MOV CX,BUF\_LEN

MOV AL,0

MOV AH,13H

INT 10H

MOV BH,0

MOV DX,CURSOR

MOV AH,2

INT 10H

POP ES

POP DS

POPA

IRET

NEW08H ENDP

GET\_TIME PROC

MOV AL,4

OUT 70H,AL

JMP $+2

IN AL,71H

MOV AH,AL

AND AL,0FH

SHR AH,4

ADD AX,3030H

XCHG AH,AL

MOV WORD PTR HOUR,AX

MOV AL,2

OUT 70H,AL

JMP $+2

IN AL,71H

MOV AH,AL

AND AL,0FH

SHR AH,4

ADD AX,3030H

XCHG AH,AL

MOV WORD PTR MIN,AX

MOV AL,0

OUT 70H,AL

JMP $+2

IN AL,71H

MOV AH,AL

AND AL,0FH

SHR AH,4

ADD AX,3030H

XCHG AH,AL

MOV WORD PTR SEC,AX

RET

GET\_TIME ENDP

BEGIN: PUSH CS

POP DS

MOV AX,3508H

INT 21H

MOV OLD\_INT,BX

MOV OLD\_INT+2,ES

MOV DX,OFFSET NEW08H

MOV AX,2508H

INT 21H

NEXT: MOV AH,0

INT 16H

CMP AL,'q'

JNE NEXT

LDS DX, DWORD PTR OLD\_INT

MOV AX,2508H

INT 21H

MOV AH,4CH

INT 21H

CODE ENDS

END BEGIN

### 3.3.2 程序测试

选择读取CMOS内储存的时间信息，并将其显示在DOS界面右上角，输入小写字符q则退出该功能，与当前时间对比发现时间相同，程序正确。当前计算机时间如图3-6所示，DOS显示时间如图3-7所示。

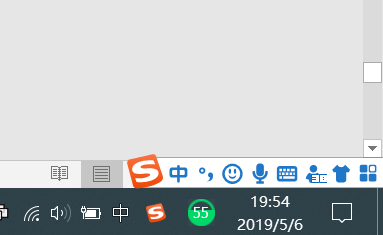


图3-6 计算机显示时间

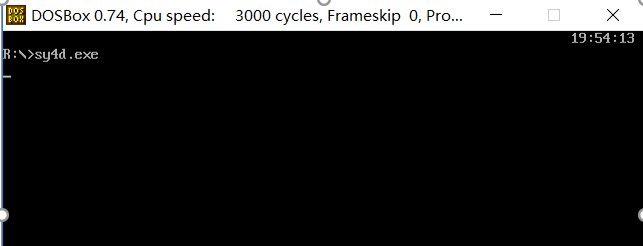


图3-7 DOS显示时间

### 3.3.3 思考题

1. 如何直接在TD下使用IN/OUT指令获取CMOS数据？

在TD的代码段中编写IN/OUT的一段语句来调用CMOS内的数据并单步调试。

## 任务4：

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。

### 3.4.1 源代码

.386

N EQU 10

NAMELEN EQU 9

PASALEN EQU 4

MUL128 EQU 128

CHECKINPUT MACRO A1,B1,C1,D1

LOCAL P1

PUSH AX

PUSH CX

MOV CL,D1

LEA SI,A1

LEA DI,B1+2

P1:

MOV AL,[DI]

CMP [SI],AL

JNE C1

INC SI

INC DI

DEC CL

JNE P1

POP CX

POP AX

ENDM

INCLUDE MACRO.LIB

STACK0 <200 DUP(0)>

DATA SEGMENT USE16

DBTEMP DB 0

DWTEMP DW 0

IN\_NAME DB 10

DB ?

DB 10 DUP(0)

IN\_PWD DB 10

DB ?

DB 10 DUP(0)

IN\_GOD DB 10

DB ?

DB 10 DUP(0)

AUTH DB 0

GA1 DB 'PEN',7 DUP(0),10

DW (35-20H)\*2,56,70,25,0,1 ;尾部增加一个推荐度排名

GA2 DB 'BOOK',6 DUP(0),9

DW (35-20H)\*2,30,25,5,0,1

GAN DB N-2 DUP('Temp-Value',(35-20H)\*2,15,0,30,0,5,0,1,0,0,0,1,0)

BNAME DB "LI HUIJUN",0 ;10

BPASS DB 4 XOR "C"

DB ('T'-20H)\*2 ;密码为TEST,用函数（X-20H)\*2加密

DB ("E"-20H)\*2

DB ("S"-20H)\*2

DB ("T"-20H)\*2

DB 57H,5AH,0BH ;随机数增加密码区长度，防止破解密码长度

SHOPNAME DB '\*\*TMALL\*\*','$' ;10

WEL\_SEN DB 'WELCOME TO TMALL','$'

NAME\_SEN DB 'PLEASE INPUT ACCOUNT NAME\*\*INPUT "ENTER" TO SKIP\*\*INPUT "q" EXIT','$'

PASS\_SEN DB 'PLEASE INPUT PASSWORD','$'

GOOD\_SEN DB 'PLEASE INPUT GOOD NAME','$'

PASS\_ERO DB 'PASSWORD ERROR','$'

NAME\_ERO DB 'ACCOUNT NAME ERROR','$'

EXIT\_SEN DB '1 EXIT','$'

SEARCH DB '2 search goods information','$'

REVISE DB '3 revise goods information','$'

COUNT DB '4 count recommendation degree','$'

SORT DB '5 recommendation degree rank','$'

TRAVERSE DB '6 print all goods information','$'

CHOOSE DB 'INPUT NUMBER TO CHOOSE OPERATION','$'

INF1 DB "ZHEKOU:","$"

INF2 DB "JIN HUO JIA:","$"

INF3 DB "XIAO SHOU JIA:","$"

INF4 DB "JIN HUO ZONG SHU:","$"

INF5 DB "XIAO SHOU LIANG:","$"

INF6 DB "TUI JIAN DU","$"

INF7 DB "ZHEKOU, JIN HUO JIA, XIAO SHOU JIA,JIN HUO ZONG SHU, XIAO SHOU LIANG,TUI JIAN DU,PAI MING","$"

INF8 DB "PEN","$"

INF9 DB "BOOK","$"

INF10 DB "TEMP-Value","$"

NN DB -23

NX DB 10

T DB ?

BUF DB 10

DB ?

DB 10 DUP(0)

P1 DW PASS1 ;地址表，间接转移反追踪

E1 DW OVER

P2 DW PASS2

OLDINT1 DW 0,0

OLDINT3 DW 0,0

DATA ENDS

CODE SEGMENT USE16

ASSUME CS:CODE,SS:STACK,DS: DATA

START:

STARTMOV

WRITE WEL\_SEN

GN1:

MOV CX,3 ;计数器

INNA:

WRITE NAME\_SEN

READ IN\_NAME

DEC CX

CMP CX,0 ;第四次输入名字

JE MENU

CMP IN\_NAME+1,0

JE MENU

CMP IN\_NAME+2,'q'

JE QTEXIT

CMP IN\_NAME+1,NAMELEN

JNE RENAME

CHECKINPUT BNAME,IN\_NAME,RENAME,NAMELEN

MOV CX,3 ;计数器

INPA:

DEC CX

CMP CX,0 ;第四次输入名字

JE MENU

WRITE PASS\_SEN

READ IN\_PWD

cli ;计时反追踪

mov ah,2ch

int 21h

push dx

CMP IN\_PWD+1,PASALEN

JNE REPASS

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

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,20H

MUL DL

CMP AL,BPASS+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 REPASS ;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.'

MOV BYTE PTR AUTH,1

MENU:

WRITE EXIT\_SEN

WRITE SEARCH

CMP AUTH,0

JE NEXT1

WRITE REVISE

WRITE COUNT

WRITE SORT

WRITE TRAVERSE

NEXT1: WRITE CHOOSE

IN1 ;ASCII

CMP AL,'1'

JNE NEXT2

JMP QTEXIT

NEXT2: CMP AL,'2'

JNE NEXT3

CALL FAR PTR F2

JMP MENU

NEXT3: CMP AL,'3'

JNE NEXT4

CALL F3

JMP MENU

NEXT4: CMP AL,'4'

JNE NEXT5

CALL F4

JMP MENU

NEXT5: CMP AL,'5'

JNE NEXT6

CALL F5

JMP MENU

NEXT6: CMP AL,'6'

JNE NEXT1

CALL F6

JMP MENU

QTEXIT:

EXIT

RENAME:

WRITE NAME\_ERO

JMP INNA

REPASS:

WRITE PASS\_ERO

JMP INPA

F2 PROC FAR

GO3: WRITE GOOD\_SEN

READ IN\_GOD ;接收商品

MOV BL,IN\_GOD+1

MOV BH,0

MOV BYTE PTR IN\_GOD+2[BX],"$"

WRITE IN\_GOD+2 ;输出商品名

CRLF

LEA SI,GA1

MOV CX,4

ADD SI,10

MOV AL,[SI]

CBW

MOV DX,0

MOV BX,10

DIV BX

ADD AX,48

ADD DX,48

MOV T,DL

MOV BL,AL

OUT2 BL

MOV BL,T

OUT2 BL

ADD SI,3

GO4: CRLF

MOV AX,[SI]

MOV DX,0

MOV BX,10

DIV BX

ADD AX,48

ADD DX,48

MOV T,DL

MOV BL,AL

OUT2 BL

MOV BL,T

OUT2 BL

ADD SI,2

LOOP GO4

; ;测试

; MOV SP,8888H

RET

F2 ENDP

F3 PROC

GO8:

WRITE INF1

READ BUF

MOV BL,BUF+2

MOV AL,BL

SUB AL,48

MOV BL,BUF+3

SUB BL,48

MUL NX

ADD AL,BL

MOV AH,0

LEA SI,GA1

ADD SI,10

CMP AL,[SI]

JNZ GO8

MOV BL,BUF+5

MOV AL,BL

SUB AL,48

MOV BL,BUF+6

SUB BL,48

MUL NX

ADD AL,BL

MOV AH,0

MOV [SI],AL

GO5:

WRITE INF2

READ BUF

MOV BL,BUF+2

MOV AL,BL

SUB AL,48

MOV BL,BUF+3

SUB BL,48

MUL NX

ADD AL,BL

MOV AH,0

LEA SI,GA1

ADD SI,11

CMP AX,[SI]

JNZ GO5

MOV BL,BUF+5

MOV AL,BL

SUB AL,48

MOV BL,BUF+6

SUB BL,48

MUL NX

ADD AL,BL

MOV AH,0

MOV [SI],AX

GO6:

WRITE INF3

READ BUF

MOV BL,BUF+2

MOV AL,BL

SUB AL,48

MOV BL,BUF+3

SUB BL,48

MUL NX

ADD AL,BL

MOV AH,0

LEA SI,GA1

ADD SI,13

CMP AX,[SI]

JNZ GO6

MOV BL,BUF+5

MOV AL,BL

SUB AL,48

MOV BL,BUF+6

SUB BL,48

MUL NX

ADD AL,BL

MOV AH,0

MOV [SI],AX

GO7:

WRITE INF4

READ BUF

MOV BL,BUF+2

MOV AL,BL

SUB AL,48

MOV BL,BUF+3

SUB BL,48

MUL NX

ADD AL,BL

MOV AH,0

LEA SI,GA1

ADD SI,15

CMP AX,[SI]

JNZ GO7

MOV BL,BUF+5

MOV AL,BL

SUB AL,48

MOV BL,BUF+6

SUB BL,48

MUL NX

ADD AL,BL

MOV AH,0

MOV [SI],AX

RET

F3 ENDP

F4 PROC

MOV SI,N

XOR BX,BX

LOOP1:

MOVZX EAX,WORD PTR GA1[BX+11]

SHL EAX,7

CDQ

DIV WORD PTR GA1[BX+13]

MOV SI,AX ;CX 当个temp

MOVZX EAX,WORD PTR GA1[BX+17]

SHL EAX,7

CDQ

DIV WORD PTR GA1[BX+15]

ADD AX,SI

MOV WORD PTR GA1[BX+19],AX

ADD BX,23

DEC SI

JNE LOOP1

RET

F4 ENDP

F5 PROC

PUSH AX

XOR SI,SI

LUP1:

;某个商品排名=比其推荐度大的个数+1

IMUL DI,SI,23

MOV CX,N

MOV AX,WORD PTR GA1[DI+19]

XOR BX,BX

LUP2:

CMP WORD PTR GA1[BX+19],AX

JNG SKIP1

INC WORD PTR GA1[DI+21]

SKIP1:

ADD BX,23

DEC CX

JE LUP2

INC SI

CMP SI,N

JL LUP1

POP AX

RET

F5 ENDP

F6 PROC

; MOV BL,IN\_GOD+1

; MOV BH,0

; MOV BYTE PTR IN\_GOD+2[BX],"$"

; WRITE IN\_GOD+2 ;输出商品名

; CRLF

WRITE INF8

WRITE INF7

LEA SI,GA1

MOV CX,6

ADD SI,10

MOV AL,[SI]

CBW

MOV DX,0

MOV BX,10

DIV BX

ADD AX,48

ADD DX,48

MOV T,DL

MOV BL,AL

OUT2 BL

MOV BL,T

OUT2 BL

ADD SI,1

GO9:

CRLF

MOV AX,[SI]

MOV DX,0

MOV BX,10

DIV BX

ADD AX,48

ADD DX,48

MOV T,DL

MOV BL,AL

OUT2 BL

MOV BL,T

OUT2 BL

ADD SI,2

LOOP GO9

WRITE INF9

WRITE INF7

LEA SI,GA2

MOV CX,6

ADD SI,10

MOV AL,[SI]

CBW

MOV DX,0

MOV BX,10

DIV BX

ADD AX,48

ADD DX,48

MOV T,DL

MOV BL,AL

OUT2 BL

MOV BL,T

OUT2 BL

ADD SI,1

GO10:

CRLF

MOV AX,[SI]

MOV DX,0

MOV BX,10

DIV BX

ADD AX,48

ADD DX,48

MOV T,DL

MOV BL,AL

OUT2 BL

MOV BL,T

OUT2 BL

ADD SI,2

LOOP GO10

WRITE INF10

WRITE INF7

LEA SI,GAN

MOV CX,6

ADD SI,10

MOV AL,[SI]

CBW

MOV DX,0

MOV BX,10

DIV BX

ADD AX,48

ADD DX,48

MOV T,DL

MOV BL,AL

OUT2 BL

MOV BL,T

OUT2 BL

ADD SI,1

GO11:

CRLF

MOV AX,[SI]

MOV DX,0

MOV BX,10

DIV BX

ADD AX,48

ADD DX,48

MOV T,DL

MOV BL,AL

OUT2 BL

MOV BL,T

OUT2 BL

ADD SI,2

LOOP GO11

RET

F6 ENDP

CODE ENDS

END START

### 3.4.2 程序测试

1. 最大输入测试

在输入用户名时分三次输入了非BOSS名称“fff1”、“fff2”、“fff3”后，程序直接跳转到了访客模式，如图3-8所示。

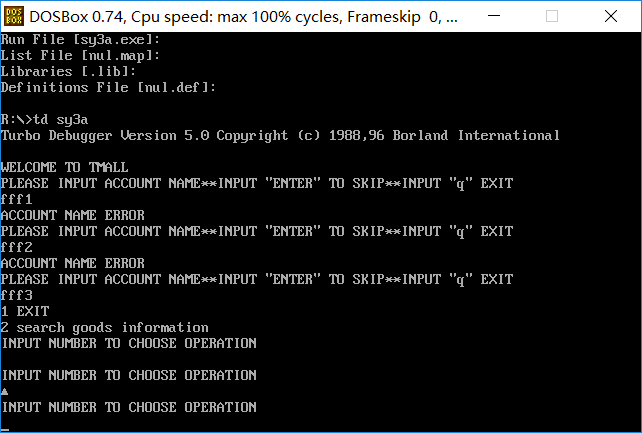


图3-8 最大输入测试

2. 加密方法

进货价加密为利用公式（x-20H）\*2，密码加密方法与进货价公式相同，同时增加了密码区的长度防止密码长度被破解，而记录实际密码长度的部分用4 XOR C进行加密，代码截图如图3-9所示。

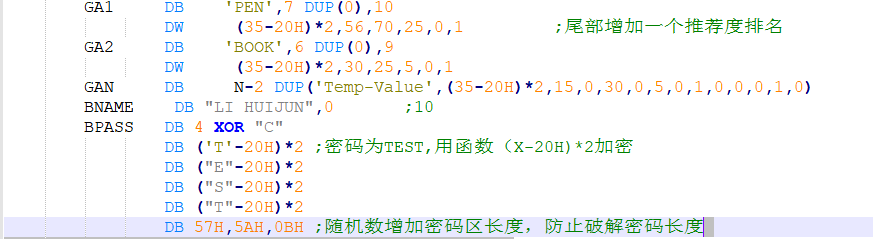


图3-9 加密代码展示

3. 反追踪测试

本次实验中我总共使用了两种反跟踪方法，第一种为中断矢量表检查，第二种为计时检查。

中断矢量表检查：通过在程序中添加修改中断矢量表中1号和3号调用的功能，在td等逐步调试工具在运行这段修改中断矢量表的程序中就会无法再运行，如图3-10所示。

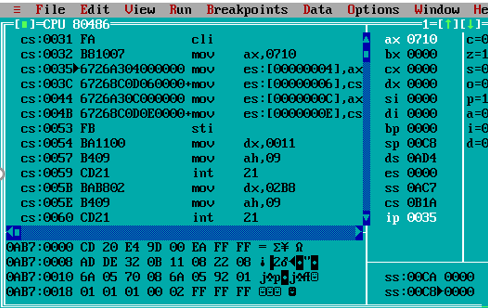


图3-10 中断矢量表检查

计时检查：在输入密码结束到比对账号密码开始之前加入一段不影响程序的计时代码，若用td等调试文件，由于运行时间太久，判断非正常执行程序，而退出程序，成功防止了用td破解，如图3-11所示。

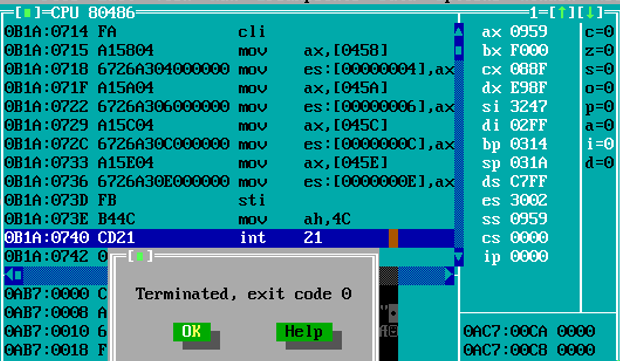


图3-11 计时检查

## 任务5

### 3.5.1 解密方法

账号与密码部分破解：单步调试程序，在账号与密码对比前直接跳过该阶段，进入BOSS部分。

功能模块判断：在TD中观察代码段，找到含有大量的9号调用语句的部分，猜测此部分就是功能模块。

### 3.5.2 思考题

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？

直接在TD中进入数据段寻找密码。

1. 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）

直接在TD中进入数据段寻找各商品的进货价。

1. 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）

在TD中直接跳过执行密码判断部分的程序。

# 四．总结与体会

任务1-3：在本次实验中，我明白了中断矢量表在内存中的情况以及如何修改中断矢量表、IN/OUT的具体使用方法以及内存与外部设备的关系。

任务4：在本次实验中我尝试了几种常用的反跟踪的方法，包括中断矢量表检查与计时检查。

任务5：在本次实验中我明白了如何通过静态反汇编和动态反汇编结合，使用TD破解程序，通过分析可能为反跟踪代码的部分，来跳过这段程序防止反跟踪。

在本次破解程序与反跟踪的任务中，我明白了汇编语言这种低级语言的意义，由于更接近与机器语言，汇编比C语言这类高级语言可以更方便的修改数据域和内存中的数据，如中断矢量表，可以更好的实现对数据的保护。