

**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 中断与反跟踪**

**实验时间： 2017-04-28，14：00-17：30 实验地点： 南一楼804室90号实验台**

**2017-05-07，14：30-18：00**

**指导教师： 李海波**

**专业班级：校交 201601班**

**学 号： U201612696 姓 名： 陈淏睿**

**同组学生： 景常进 报告日期： 2018年 5 月 8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：2018.5.8

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 实验目的与要求 2](#_Toc515198487)

[2 实验内容 2](#_Toc515198488)

[3 实验过程 3](#_Toc515198489)

[3.1 任务1 3](#_Toc515198490)

[3.1.1 实验步骤 3](#_Toc515198491)

[3.1.2 实验记录与分析 3](#_Toc515198492)

[3.2 任务2 4](#_Toc515198493)

[3.2.1 设计思想及存储单元分配 4](#_Toc515198494)

[3.2.2 流程图 4](#_Toc515198495)

[3.2.3 源程序 5](#_Toc515198496)

[3.2.4 实验步骤 6](#_Toc515198497)

[3.2.5 实验记录与分析 6](#_Toc515198498)

[3.3 任务3 8](#_Toc515198499)

[3.3.1 设计思想及存储单元分配 8](#_Toc515198500)

[3.3.2 流程图 8](#_Toc515198501)

[3.3.3 源程序 9](#_Toc515198502)

[3.3.4 实验步骤 10](#_Toc515198503)

[3.3.5 实验记录与分析 10](#_Toc515198504)

[3.4 任务4 10](#_Toc515198505)

[3.4.1 设计思想及存储单元分配 10](#_Toc515198506)

[3.4.2 流程图 10](#_Toc515198507)

[3.4.3 源程序 11](#_Toc515198508)

[3.4.4 实验步骤 17](#_Toc515198509)

[3.4.5 实验记录与分析 18](#_Toc515198510)

[3.5 任务1 18](#_Toc515198511)

[3.5.1 实验步骤 18](#_Toc515198512)

[3.5.2 实验记录与分析 19](#_Toc515198513)

[4 总结与体会 23](#_Toc515198514)

[参考文献 27](#_Toc515198515)

# 实验目的与要求

本次实验的主要目的与要求有下面5点，所有的任务都会围绕这5点进行，希望大家事后检查自己是否达到这些目的与要求。

(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 、10H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），观察中断矢量表中的信息。

（2） 编写程序，用 DOS系统功能调用方式获取，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，要求在程序返回DOS操作系统后，输入键盘上的小写字母时都变成了大写字母。

要求：

（1）在 DOS虚拟机或DOS窗口下执行程序，中断服务程序驻留内存。

（2）在DOS命令行下键入小写字母，屏幕显示为大写，键入大写时不变。执行TD，在代码区输入指令“mov AX,0”，看是否都变成了大写。

（3）选作：另外编写一个中断服务程序的卸载程序，将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：

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。

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。

可以采用计时、中断矢量表检查、堆栈检查、间接寻址等方式中的一种或多种方式反跟踪（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。

# 实验过程

## 任务1

### 实验步骤

1. 准备上机实验环境。

2. 在TD的代码窗口中ds段goto 0:0，观察中断矢量表。

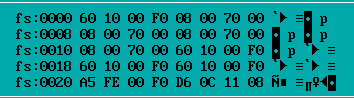
3. 在TD中输入指令，通过把[1H\*4]和[10H\*4]开始的一个字的内容分别送入ax,bx观察两个入口地址的值。

4. 在TD中输入指令，通过35号调用调用1H和10H,分别观察指令执行完毕后BX的值。

### 实验记录与分析

1. 实验环境条件：P3 1GHz，256M内存；WINDOWS XP下DOSBox0.72； TD.EXE 5.0。

2.在本实验前我知道了可以将制定虚拟机C盘的操作写进DOSBox 0.74 Options.bat，打开虚拟机后便会自动将目标文件夹设定为虚拟机的C盘。在显示DS段的窗口中goto 0:0，发现光标跳转至fs首址，说明fs的首址为物理地址0，如图3.1.1和图3.1.2所示。此处显示1H入口地址为0080H,10H入口地址为1300H。



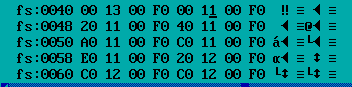


图3.1.1 & 图3.1.2直接观察中断矢量表

3.直接将1H和10H的入口地址分别送入EAX和EBX，观察到两寄存器的值如图3.1.3所示。发现1H的入口地址与上一步中结果不同，为0B1AH,10H的入口地址仍为1300H。



图3.1.3直接观察终端矢量表

4.用35号调用分别获取1H，10H的入口地址，结果如图3.1.4和3.1,5所示，01H入口地址为0B1AH,10H入口地址为1300H,与上一步相同。请教老师后得知td会修改中断矢量表的部分值，导致用TD直接观察的结果有所不同。



图3.1.4 35号调用获取1H入口地址



图3.1.5 35号调用获取10H入口地址

## 任务2

### 设计思想及存储单元分配

修改中断处理程序，先编写新的中断处理程序，然后将旧程序的IP,CS存入OLD\_INT，获取新程序的IP,CS放入老程序的相应位置。

1.存储单元分配

OLD\_INT：双字变量，前两个字节存储旧IP, 后两个字节存储旧CS。

2.寄存器分配

AX：负责中断程序参数。

DX：生成驻留节数。

### 流程图

图3.2.1是任务2内存驻留接管和新中断处理程序的流程图。

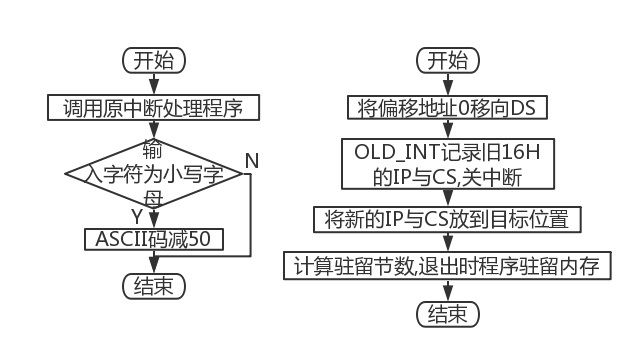


图3.2.1内存驻留接管和新中断处理程序的程序流程图

### 源程序

T1-1.asm

.386

CODE SEGMENT USE16

ASSUME CS:CODE, SS:STACK

OLD\_INT DW ?,?

NEW16H: CMP AH,0 ;判断是否调用第一种键盘输入

JE QUIT

CMP AH,10H ;判断是否调用第二种键盘输入

JE QUIT

JMP DWORD PTR OLD\_INT ;继续原来的中断处理程序功能

QUIT: PUSHF

CALL DWORD PTR OLD\_INT

CMP AL,61H

JB BACK

CMP AL,7AH

JA BACK

SUB AL,20H

BACK: IRET

START: XOR AX,AX

MOV DS,AX ;0→DS

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

MOV CS:OLD\_INT,AX

MOV AX,DS:[16H\*4+2] ;保存旧的IP,CS

MOV CS:OLD\_INT+2,AX

CLI

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

MOV DS:[16H\*4+2],CS ;写入新的CS,IP时需要关中断

STI

MOV DX, OFFSET START+15 ;计算驻留节数

SHR DX,4

ADD DX,10H

MOV AL,0

MOV AH,31H

INT 21H

CODE ENDS

STACK SEGMENT STACK

DB 200 DUP(0)

STACK ENDS

END START

T1-2.asm

.386

CODE SEGMENT USE16

ASSUME CS:CODE,SS:STACK

;初始IP:11E0H;初始CS:0F000H

START: XOR AX,AX

MOV FS,AX ;0->DS

MOV BX,FS:[16H\*4]

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

CLI ;修改中断矢量表时关中断

MOV AX,DS:[BX-4]

MOV WORD PTR FS:[16H\*4],AX ;IP放回中断矢量表相应位置

MOV AX,DS:[BX-2]

MOV WORD PTR FS:[16H\*4+2],AX ;CS放回中断矢量表相应位置

STI

;MOV DX, OFFSET START+15 ;计算驻留节数

;SHR DX,4

;ADD DX,10H

MOV AH,31H ;退出时(DX)节主存单元驻留

INT 21H

CODE ENDS

STACK SEGMENT STACK USE16

DB 200 DUP(0)

STACK ENDS

END START

### 实验步骤

1. 参考书上的内存驻留程序，编写新的16H中断处理程序。

2. 汇编并连接程序，直至汇编不再报错。

3. 执行程序，若出现错误则单步调试直至产生正确结果。

### 实验记录与分析

1. 准备上机实验环境。

2. 参考书上的内存驻留程序，编写新的16H中断处理程序。编译运行中未出现问题，但运行并正常退出后发现控制台仍有光标闪烁但无法正常输入语句。取消驻留后在程序中直接尝试进行字符串输入，发现仍然无法输入。排除对旧入口地址的保存与新入口地址的引用方面的错误，得知新的中断处理程序的编写出了问题。请教同学后发现需要先判断键盘输入方式而不能上来就开始判断AL中的符号。



图3.2.2无法正常输入

3. 连接过程没有发生异常。

4. 汇编连接过程中无异常，如图3.2.3所示。

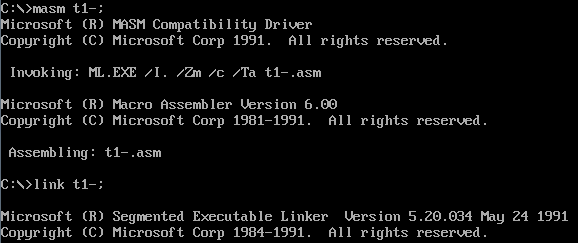


图3.2.3汇编连接正常

5. 在CAPS LOCK关的情况下输入英文字符，发现显示为大写字母，证明ASCII码修改成功，如图3.2.4所示。



图3.2.4字母显示为大写

6. 观察TD可知原中断处理程序的IP,CS分别为11E0H和0F000H，如图3.2.5和图3.2.6所示。由于退出t1-1后中断处理程序驻留内存，因此可将旧的IP和CS放回中断矢量表并驻留，即为卸载程序编制思路。



图3.2.5原IP

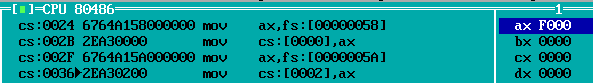


图3.2.6原CS

7. 按照此思路编制卸载程序，汇编过程中报错如图3.2.7所示。这使我再一次想起16进制最高位为字母时应在前面加0以便于机器识别。

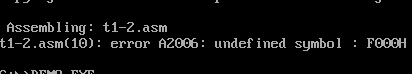


图3.2.7数据书写错误

8. 再次汇编连接，过程中一切正常。运行t1-1后再运行t1-2,发现小写功能恢复，但与此同时CAPS LOCK却失灵。

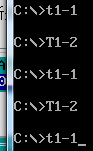


图3.2.8仅小写恢复

9.经过思考，我想到卸载程序只需要将初始IP与CS送回中断矢量表就算完成任务，没有必要驻留内存。尝试删掉驻留内存语句后汇编连接运行，发现CAPS LOCK功能恢复正常，如图3.2.9所示，但驻留内存后CAPS LOCKS失灵的原因还需进一步探究。



图3.2.9大写恢复

## 任务3

### 设计思想及存储单元分配

修改中断处理程序，先编写新的中断处理程序，然后将旧程序的IP,CS存入OLD\_INT，获取新程序的IP,CS放入老程序的相应位置。

1.存储单元分配

TIME：字节串，存放转化成ASCII码的日期。

ADS:字节串，存放需要调用的诸CMOS端口，从前向后分别为年-月-日-时-分-秒

2.寄存器分配

SI：负责TIME串的遍历。

BX：负责ADS串的遍历。

AL,AH：存放转换成ASCII码的字符并放入TIME相应位置。

### 流程图

图3.3.1是任务3显示系统时间的程序流程图。

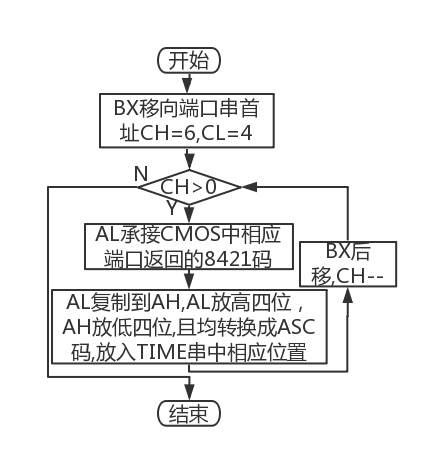


图3.2.1显示系统时间的程序流程图

### 源程序

STACK SEGMENT USE16 STACK

DB 20 DUP(0)

STACK ENDS

DATA SEGMENT

TIME DB 0,0,'-',0,0,'-',0,0,' ',0,0,':',0,0,':',0,0,'$' ;YY-MM-DD HH:MM:SS

ADS DB 09H,08H,07H,04H,02H,00H ;CMOS端口

DATA ENDS

CODE SEGMENT

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

START:

MOV AX,DATA

MOV DS,AX

LEA SI,TIME ;SI移向TIME首地址

LEA BX,ADS ;BX移向端口串首址

MOV CX,0604H ;CH为循环次数，CL为右移位数

LPA: MOV AL,DS:[BX]

OUT 70H,AL

IN AL,71H ;AL存放获得的一字节

MOV AH,AL

SHR AL,CL ;AL放高位

AND AH,0FH ;AH放低位

ADD AX,3030H

MOV WORD PTR DS:[SI],AX

INC BX

ADD SI,3

DEC CH

JNZ LPA

MOV DX,OFFSET TIME ;输出时间字符串

MOV AH,09H

INT 21H

MOV AH,4CH

MOV AL,0H

INT 21H

CODE ENDS

END START

### 实验步骤

1. 准备上机实验环境。

2. 根据3.3.1中变量设计定义变量，通过循环分别调用CMOS中年月日时分秒的端口，处理为十六进制ASCII码后放入相应串。

3. 重复汇编，连接过程，解决可能遇到的汇编问题。

4.运行程序，若出现问题则用TD单步调试直至输出正确。

### 实验记录与分析

1. 实验环境条件：Intel® Core™ i5-3230M CPU 2.60GHz，2.86G内存；WINDOWS 7下DOSBox0.74； notepad++ 7.55;MASM.EXE 6.0;LINK.EXE 5.2;TD.EXE 5.0。

2. 编写CMOS端口内容提取程序，目标为提取CMOS中记录的年/月/日/时/分/秒。首先将提取的8421码转为十进制。

3. 汇编链接时一切正常。

4. 运行程序，发现程序可以按照十进制输出正确结果。



图3.3.2按十进制输出结果正确

## 任务4

### 设计思想及存储单元分配

1. 设计思想：

对于用户名与密码字符采用函数加密方式，对其长度采用逻辑运算亲加密方式。采用间接转移，计时，检查堆栈，检查中断矢量表，添加无关代码等方式实现反跟踪，且将功能代码，反跟踪代码及无关代码尽量混在一起而非各自成段。

1. 存储单元分配：

ADR1~ADR5：均为DW型变量，存放代码段中标号的IP与CS。

OLDINT1,OLDINT3：均为DW型变量，分别存放1号与3号中断的入口地址用于在程序运行结束前还原。

### 流程图

图3.4.1是任务4反跟踪的程序流程图。

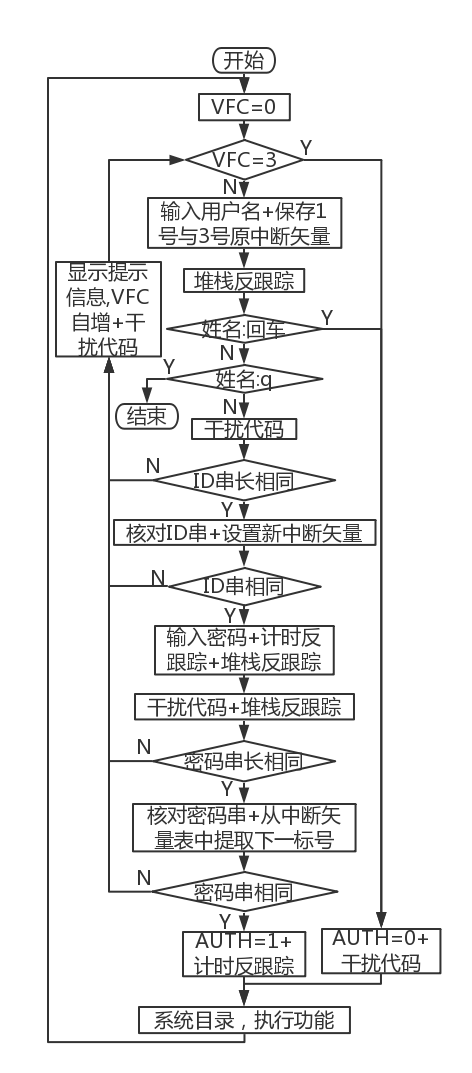


图3.4.1反跟踪流程图

### 源程序

┇ ；此行之前为实验三任务一相关宏定义

INTR MACRO T ;参数为寄存器名的一部分

IMUL SI,DS:[B&T&+16] ;进货数量

OR DX,DX

NEG DI

; 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

IMUL DI,DS:[B&T&+14]

MOV AX,DS:[B&T&+12] ;销售价

SAR DI,1

MOV DI,AX

LEA AX,GA2

MOV SI,DS:[B&T&+10] ;进货价

SHR SI,2

NEG DI

IMUL AX,DS:[B&T&+16]

MOV AX,DS:[B&T&+12] ;销售价

SAR DI,1

MOV DI,AX

IMUL DI,DS:[B&T&+10]

ADD SI,9

IMUL SI,DS:[B&T&+14] ;进货数量

OR DX,SI

NEG DI

AND DX,DI

NEG DI

IMUL DI,DS:[B&T&+14]

MOV AX,DS:[B&T&+12] ;销售价

SAR DI,1

MOV DI,AX

IMUL DI,DS:[B&T&+14]

; 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

; jne STEP2

; pop dx

MOV AX,DS:[B&T&+12] ;销售价

SAR DI,1

MOV DI,AX

MOV CH,10

IMUL DI,DS:[B&T&+16] ;已售数量

SUB DI,SI

SHL DX,3

MOV AX,DI

IMUL HKU

IDIV SI

ENDM

┇ ；此行之后为实验三任务一相关宏定义

┇ ；此行之前为实验三任务一相关数据段

ADR3 DW EXIT

ADR1 DW BRK10

ADR4 DW LPA\_2

ADR2 DW LPA\_1

ADR5 DW BAR1

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

┇ ；此行之后为实验三任务一数据段结尾

┇ ；此行之后为实验三任务一代码段开头

MOV DS,AX

;接管调试用中断，中断矢量表反跟踪(缩进4，小写)

;干扰程序(缩进16)

;堆栈检查反跟踪(缩进8,小写)

;计时反跟踪(缩进12,小写)

STEP1:

xor bx,bx

mov fs,bx

mov bl,IN\_NAME

cli

IO TIP\_1,9 ;显示提示：输入用户名

shr bl,2

IO IN\_NAME,10 ;输入用户名

inc bl ;bl=4

push ADR2 ;PASS2的地址压栈

mov ax,fs:[bx] ;保存原1号和3号中断矢量

mov OLDINT1,ax

mov ax,fs:[bx+2]

XOR BL,BL ;单独输入回车符时真实串长为0

LEA DI,[IN\_NAME+1] ;找到真实串长

pop ax

CMP BL,[DI]

JE BAR1

mov bx,[esp-2] ;把栈顶上面的字（PASS2的地址）取到，栈地址上小下大所以是减

JMP BRK8

BAR1:

MOV AUTH,0 ;认证方式:顾客

IO TIP\_5,9 ;显示提示：以游客方式登录

JMP STEP3

BRK8:

INC DI ;字符串本体首地址

CMP BYTE PTR [DI],'q' ;比较q

JE EXIT

IO TIP\_9,9 ;换行，回车

JMP STEP2 ;核验用户名正确性

BRK7:

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

IO TIP\_2,9 ;显示提示：输入密码

IO IN\_PWD,10 ;输入密码

IO TIP\_9,9 ;换行，回车

mov ah,2ch ;获取第二次秒与百分秒

int 21h

sti

cli ;堆栈检查反跟踪

push ADR4 ;PASS2的地址压栈

JMP BRK9 ;核验密码正确性

STEP2:

MOV DH,ES:[BP+SI]

AND DH,ES:[BP+DI]

sti

MOV CL,IN\_NAME+1 ;输入用户名串长

INC DX

LEA SI,[ECX]

mov OLDINT1+2,ax

MOV CH,CL ;输入用户名串长

mov ax,fs:[3\*4]

XOR BYTE PTR ES:[BX+SI],0FAH

mov OLDINT3,ax

OR CL,'x'

mov ax,fs:[3\*4+2]

SHR SI,2

mov OLDINT3+2,ax

CMP CL,BNAME

JNE BAR2

MOV DH,1

LEA DI,[IN\_NAME+2]

LEA SI,[BNAME+1]

jmp bx ;如果被跟踪，将不会转移到LPA\_1

LPA\_1:

XOR BH,BH

MOV BL,[DI]

SUB BL,42

cli ;设置新的中断矢量(被写在LPA\_1循环体里)

mov ax,OFFSET NEWINT

SHL BX,1

mov fs:[1\*4],ax

CMP BL,[SI]

mov fs:[1\*4+2],cs

JNE BAR2

INC SI

mov fs:[3\*4],ax

INC DI

DEC CH

mov fs:[3\*4+2],cs

sti

JNE LPA\_1

JMP BRK7

BAR2:

IO TIP\_7,9 ;显示提示:用户名错误

BRK25:

MOV CH,VFC

PUSH DX

INC DX

cli ;堆栈检查反跟踪

push ADR5 ;ADR5的地址压栈

CMP AX, 0

INC CH

MOV VFC,CH

MOV BX, 0

POP AX

ADD AL, '0'

pop ax

mov ax,[esp-2] ;把栈顶上面的字（PASS2的地址）取到

sti

MOV ES:[BX], AL

INC BX

CMP CH,3

JNE STEP1

MOV BYTE PTR ES:[BX+0], 0AH

MOV BYTE PTR ES:[BX+1], 0DH

MOV BYTE PTR ES:[BX+2], '$'

POP BX

jmp ax

LEA DX,[BX+2]

BRK9:

MOV DL,DH

MOV CH,0

pop bx

MOV CL,IN\_PWD+1 ;比较输入的串长与密码长度是否一样

MOV CH,CL

SHR DI,6

MOV DH,0

mov ax,[esp-2] ;把栈顶上面的字（PASS2的地址）取到

sti

XOR CL,'t'

CMP CL,BPASS

JNE BAR3

LEA DI,[IN\_PWD+2]

ADD SI,DX

LEA SI,[BPASS+1]

jmp ax

LPA\_2:

XOR BH,BH

MOV BL,[DI]

SUB BL,75

SHL BX,1

CMP BL,[SI]

JNE BAR3

mov dx,fs:[1\*4] ;检查中断矢量表是否被调试工具阻止修改或恢复

INC SI

INC DI

inc dx

DEC CH

JNE LPA\_2

;JMP BRK10

jmp dx

BAR3:

IO TIP\_8,9 ;显示提示:密码错误

JMP BRK25

BRK10:

MOV AUTH,1 ;认证方式:店主

IO TIP\_6,9 ;显示提示:以店主身份登录

STEP3:

cmp dx,[esp] ;计时是否相同

MOV BX,OFFSET GAN

jz STEP3 ;如果计时相同，通过本次计时反跟踪,转到STEP3

MOV CL,'2'

MOV ES:[BX],CX

MOV CL,'6'

MOV SI,1

CMP AUTH,1

JNE BRK6

mov bx,offset ADR3 ;如果计时不同，则把转移地址偏离ADR3(EXIT)

MOV ES:[SI],CX

MOV CL,'9'

MOV ES:[BX+2],CX

MOV CL,'6'

MOV SI,1

MOV ES:[BX+SI+2],CX

pop dx

IO TIP\_13,9 ;店主菜单

MOV AH,1 ;输选项

INT 21H

IO TIP\_9,9

CMP AL,'3'

JE STEP3\_3

┇ ；此行之后为实验三任务一相关代码段

┇ ；此行之前为实验三任务一相关代码段

EXIT:

cli ;还原中断矢量

mov ax,OLDINT1

mov fs:[1\*4],ax

mov ax,OLDINT1+2

mov fs:[1\*4+2],ax

mov ax,OLDINT3

mov fs:[3\*4],ax

mov ax,OLDINT3+2

mov fs:[3\*4+2],ax

sti

MOV AH,4CH ;exit

INT 21H

NEWINT: iret

TESTINT: jmp ADR1

┇ ；此行之后为实验三任务一相关代码段

┇ ；此行之前为实验三任务一功能四代码

PART5 PROC USES AX BX CX DX SI

MOV CL,M ;CL负责商店一的遍历

LEA BP,GA1

LPI: ;之前所有寄存器的值可作废

FND BRK16,GB1,J,K,18 ;循环比较BP与BX指向的串，相同时转到BRK16

BRK16:

XOR SI,SI

MOV CH,AL

LPI\_A: ;循环输出商品串

MOV DL,[BX+SI]

MOV AH,2

INT 21H

INC SI

DEC CH ;商品串未输出完毕

JNZ LPI\_A

cli ;堆栈检查反跟踪

XOR AH,AH

push BRK29 ;BRK29的地址压栈

IMUL AX,3

XOR DX,DX

MOV AX, ES:[EAX+EAX] ;用基变址寻址取操作数，L1为外循环，(SI)为循环变量，;相当于i

CMP BX, ES:[EAX+EAX] ;L2为内循环，(DI)为循环变量，相当于j

MOV CL,DH

XCHG AX, ES:[EBP+EAX] ;[BX+SI]<[BX+DI]，交换

MOV ES:[BP+SI], BX ;BP新值送回[BX+SI]

ADD AX,3 ;AH>=AL，不需交换，(AH)直接和后一个数比较，相当于j++

CMP AUTH,CL

JE BRK28

ADD CX,3 ;外层变量SI加3

pop dx

MOV AX,SI ;相当于j=i

mov cx,[esp-2] ;把栈顶上面的字（PASS2的地址）取到

sti

ADD DX,3

jmp cx ;如果被跟踪，将不会转移到PASS2

BRK29:

LEA DX,TIP\_12 ;制表

┇ ；此行之前为实验三任务一工功能五相关代码段

### 实验步骤

1. 准备上机实验环境。

2. 以实验三的t1.asm为基础按照样例程序所给的多种反跟踪方式插入相关代码。在添加反跟踪代码及干扰代码时应注意不可干扰原程序的数据传输，开关中断操作也不可交叉。

3.重复汇编，连接过程，解决可能遇到的汇编问题。

4.运行程序，检查是否运行正常。由于已添加反跟踪代码，因此用TD调试会变得较为困难。此时在调试时应考虑按反跟踪程序段一段一段用排除法检查，最终找到出问题的段落。

5.添加干扰程序段以进一步强化程序反跟踪能力，注意事项与2中相同。

6.重复步骤3,4直至程序运行正常并给出正确结果。

7.在功能三利润计算中添加解密函数，在功能五利润率显示中在以游客身份登陆时进行数据遮挡，同时添加干扰代码及反跟踪语句。

8.重复步骤3,4直至程序运行正常并给出正确结果。

### 实验记录与分析

1. 实验环境条件：Intel® Core™ i5-3230M CPU 2.60GHz，2.86G内存；WINDOWS 7下DOSBox0.74； notepad++ 7.55;MASM.EXE 6.0;LINK.EXE 5.2;TD.EXE 5.0。

2. 在用户名、密码比较及登录方式核对部分按照样例程序所给的多种反跟踪方式插入相关代码。在阅读示例程序时，我一开始对于PASS3: mov bx,es:[1\*4]，inc bx两句不理解，因为这里本来是应该跳转到TESTINT的语句的，但向中断矢量表从存入的却是NEWINT，而对IP加一也不清楚加了之后下一步该执行什么语句。请教老师与同学以后明白CS段中的数据占用的空间大小不一，像iret只占用1个字节，因此做inc bx后可达到TESTINT所在位置。

3. 汇编连接时一切正常，但运行过程中出现较多次死机情况。用排除法锁定有问题的程序段后发现错误原因均为新程序段干扰了原有代码的数据传输。对相关错误进行修改，重新运行。

4. 运行程序，发现程序可以按照十进制输出正确结果，如图3.4.2 所示。

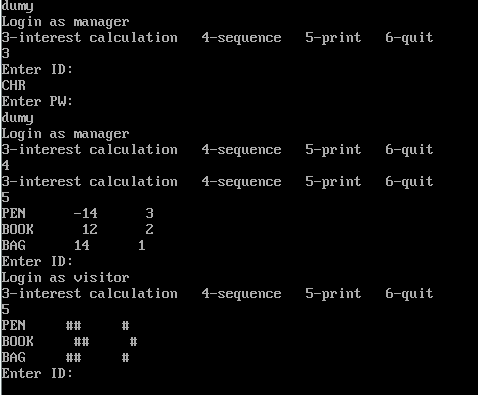


图3.4.2程序运行正常

## 任务1

### 实验步骤

1. 准备上机实验环境。

2. 在TD的代码窗口中打开队友的程序，首先观察中断矢量表，看是否能看出端倪。

3. 单步执行程序，结合反汇编语句中对于各地址的提示通过改变IP辅助调试。

4. 设法找到对密码的处理语句，反推出密码，正常执行程序并通过输入正确密码以店主身份进入系统。

### 实验记录与分析

1. 实验环境条件：Intel® Core™ i5-3230M CPU 2.60GHz，2.86G内存；WINDOWS 7下DOSBox0.74； notepad++ 7.55;MASM.EXE 6.0;LINK.EXE 5.2;TD.EXE 5.0。

2. 用TD打开程序后，首先发现将数据段装入ds，将中断矢量表装入es，如图3.5.1所示。

已知将中断矢量表装入es涉及的是中断矢量表反跟踪，因此将数据段装入ds后修改IP，直接跳到功能代码起始EA-0054.如图3.5.2所示。

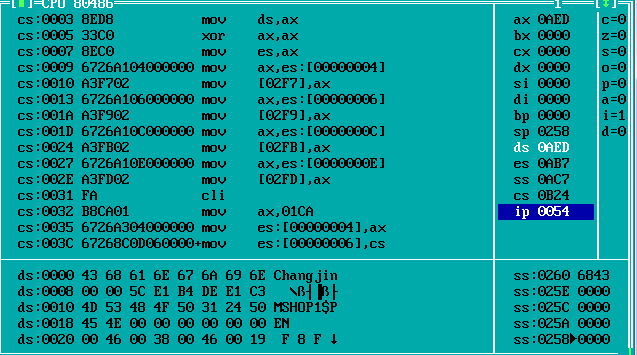


图3.5.1 修改IP为0054，跳过中断矢量表的修改

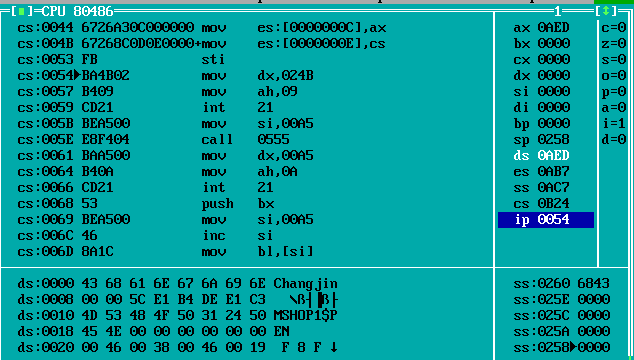


图3.5.2 功能代码开始

3.单步执行可知cs:0054-cs:005B为输入密码的提示。由图3.5.2同样看出之后将跳至cs:0555处执行。结合cs:0061-cs:0066的输入密码10号调用，配合ds段内容检查相关代码功能，判断出该子程序的功能是将ds:00a6开始的7个字节清零以保存将输入的密码，如图3.5.3所示。

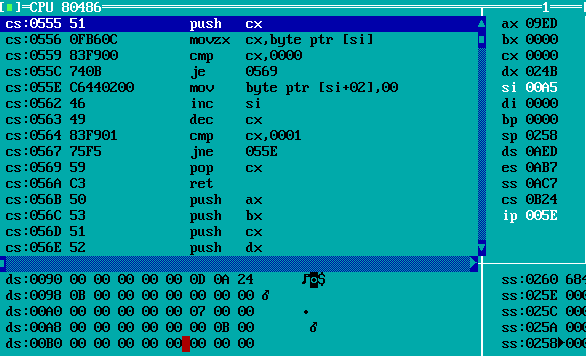


图3.5.3解读cs:0555处子程序功能

4.通过10号调用进行代码试探，试探密码为308532。密码存储串长为7，如图3.5.3所示，但由于需要存一位回车符，因此有效密码长度其实为6。cs:0068-cs:0075功能为删掉密码串尾的回车符，执行情况如图3.5.4所示，重点关注ds:00ad的变化；cs:0076-cs:007b功能为回车换行，执行情况如图3.5.4及图3.5.5所示。

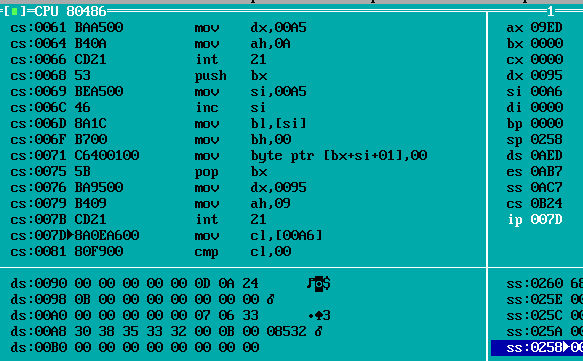


图3.5.4 35号调用获取1H入口地址

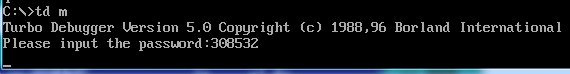


图3.5.5光标移向下一行

5. cs:007d-cs:0084判断输入密码串长，若为0则跳走；cs:0086-cs:0090则校验密码串长，分析可知其方程为x XOR 58H =5CH，如图3.5.6所示.在0-7内该数唯一确定为4，即为正确密码串长。

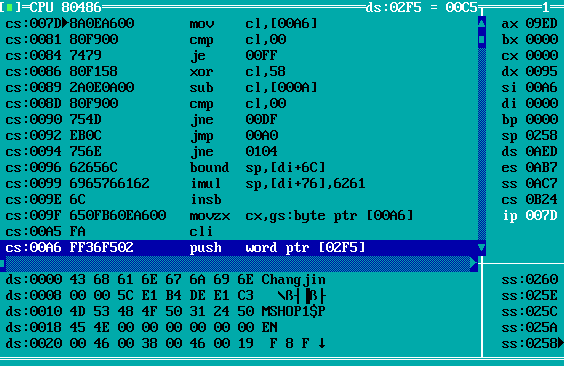


图3.5.6破解密码串长

6.重新用4位密码wasd进行试探。程序转到cs:009f执行，将cx置为4，如图3.5.7所示。cs:00a5-cs:00b5为堆栈反跟踪，破解时应记录入栈值，此处为00c5. cs:00aa,cs00ad分别将si和dl置为0和3，在后面起作用。

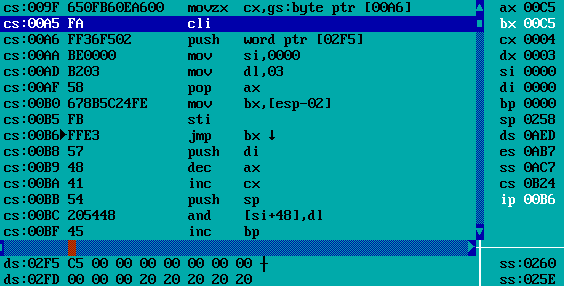


图3.5.6破解堆栈反跟踪

7. cs:00c5-cs:00d6为一循环，次数为4，如图3.5.7所示。观察ds段可知此处开始对密码进行操作。对四个字节分别有(x-29H)\*4=0E1H/0B4H/0DEH/0E1H，反推出系统内密码应为’test’。

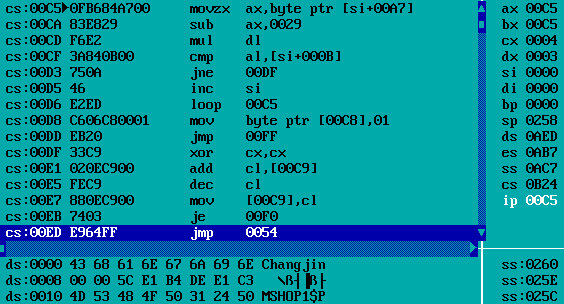


图3.5.7破解密码

8.用test测试输入，成功登入系统，如图3.5.8所示。

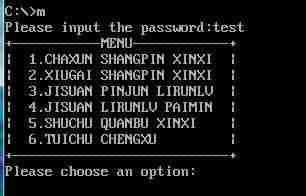


图3.5.8登录成功

9.图3.5.9，图3.5.10，图3.5.11分别为三种商品在两个商店中的信息。由于以店主身份登入，因此进货价已解密。

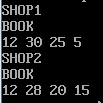
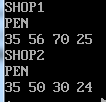
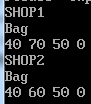
  

图3.5.9&图3.5.10&图3.5.11三种商品的全部信息

# 总结与体会

任务1的实验使我知道了中断矢量表的IP与CS在内存中的表示方式，同时熟悉了用int 35H查看入口地址的方法需要注意的是部分中断矢量表的值在TD中不会按原值显示，因此在使用时应从内存中提取而非靠观察。任务2使我对中断矢量表的接管方法有了了解，学会了内存驻留的方式。同时我也明白了对系统功能调用的修改也并不像我想象的那么简单，需要结合原功能调用提供的后台程序进行判断后方可使用。任务3让我对CMOS数据地址有了一定了解，对于8421码与ASCII码的转换也更加熟悉。

在任务4中我以防御方的角度接触到了逆向工程相关概念，了解了简单的数据加密方式以及多种反汇编的方法。我同样感受到反跟踪是一项“要想骗倒对手，先要骗倒自己”的操作，功能代码，反汇编代码和无关代码的混合给我的编程造成了一定困难，在这种情况下需要采取一定措施让自己看得清楚自己在写什么，除了充分注释以外我也采用了缩进差别的方法区分各种代码，一定程度上减小了自己的编程负担。有了反跟踪代码，调试也开始变得困难，在单步执行时稍不留神IP就会不知被自己的反跟踪拐去哪里，这对我们提出的要求是对反跟踪的方式及自己的功能做到心里有数，这样才能在单步调试时跳过陷阱，不至于真被自己骗倒。同时，也可采取排除法，一段一段的检查反汇编功能从而确定出错段落，逐渐缩小检查范围。

任务5中我以进攻方的角度第一次进行了逆向工程。我最强烈的感受还是要对基本操作心里有数，比如当前代码装的是什么数据段。一开始调试的时候我就因为连数据段都没装进DS导致要输入代码时根本不显示控制台窗口，害得自己折腾了半天。逆向工程是一项极度需要耐心的活，尤其是在面对数不胜数的跳转的时候。这次要感谢队友的是他并没有在跳转上设置过多的障碍，让我好歹能成功破解程序，但工作后不论对程序的反跟踪要求还是逆向工程要求都要上升不少，反跟踪环节也会更加复杂，比如加密方式肯定不会是现在这样写一个简单函数或者做一次逻辑操作。所以趁着在学校多熟悉代码，打牢攻防基本功还是十分重要的，这样在实践时就不会还被一些基本的逻辑困住，能用更多的精力解读更为复杂的加密过程。

思考题（实验步骤中未涉及的）：

一、

5.中断处理程序的CS/IP被修改，按照错误的入口地址会找到错误的程序段，执行时不仅得不到应有的功能还可能导致系统死机。

二、

2.可以先不将新的中断处理程序驻留内存，而是在写入新的入口地址后尝试进行字符串小写字母输入，看是否在屏幕上显示大写字母。直到输出正确时方可将新的中断处理程序驻留内存。

具体操作：删除内存驻留语句并定义一长为12的串用于输入。载入新中断处理程序后用小写进行串输入，结束后直接卸载新中断处理程序并尝试控制台输入，结果如下图所示，说明中断处理程序编制正确，可进行内存驻留。



图4.1退出程序后输入回复正常

3.可以返回DOS，且DOS正常工作。若反复执行多次内存驻留程序，则后一次保存的“旧”IP，CS实为上一次运行中断处理程序后的IP和CS，实际上形成嵌套，需执行相同次数的卸载程序才可将初始IP与CS放回中断矢量表，如图4.2所示。但也有不管执行了多少次内存驻留，用一次卸载恢复正常的方法：记录第一次内存驻留时保存的IP,CS，将其直接放进中断矢量表。执行结果如图4.3所示。具体到本实验而言,CS初始IP为11E0H,初始CS为0F000H。

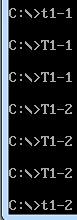
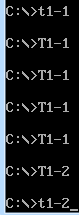
 

图4.2内存驻留程序嵌套 图4.3一次退出

4.不会被替代，如图4.4所示。

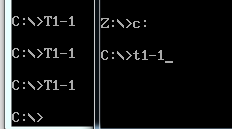


图4.4一个虚拟机中的修改对另一虚拟机无影响

5.执行一次内存驻留后用TD单步进行第二次，观察到此次保存IP与CS的值已被改变，IP的值与写程序时第一条指令前定义了双字变量OLD\_INT的情况吻合，如图4.5所示。

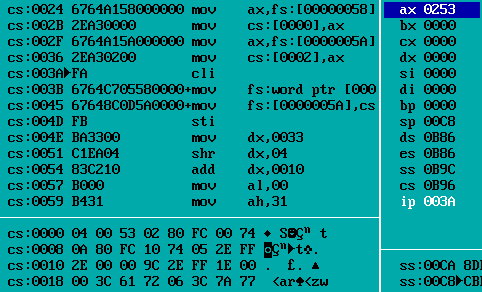


图4.5 IP与CS的值均被改变

6. 鉴于初始IP,CS被定义在16H新CS的前4个字节，因此卸载程序中取CS地址后减4即为旧IP，减2即为旧CS，将其装入中断矢量表中16H对应位置即完成；可用在TD中观察到的初始IPC与CS进行判断，若以为初始值则不执行卸载直接返回。

三、

1.将目标端口交给AL,用端口70H表明准备访问该字节；用71H将访问字节的输出参数交给AL。

2.以00H（秒）中内容距离。若当前系统描秒数为59，则一个字节的压缩BCD码为01011001，即高半字节保存十位数字的8421码，底半字节保存个位数字的8421码。

四+五、

1.观察ds段中数据，排除有明显语义的提示信息及结构有规律的数据信息后，从剩下的数据内可更容易找出用明码保存的密码段，如图4.6所示，此时密码段为‘PASSWD’。

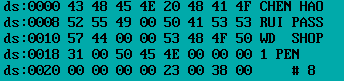


图4.6明文保存密码

2.第一种方法是直接用TD在内存中观察，如图4.7所示；第二种方法是直接将程序用二进制编辑工具打开，在文本中寻找进货价定义。

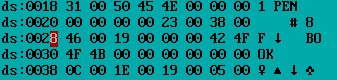


图4.7明文保存进货价

3. 设计思路：首先设定密码长度可能的上下限，在当前串长限制下遍历所有的数字与字符组合并调用待破解文件尝试。若破解成功或失败时返回对应值以决定继续下一次尝试还是显示正确密码。

4. 对于功能程序段落，可以直接用动态反汇编，单步执行观察程序运行流程；但对于反跟踪语句，干扰代码较多的部分或通过地址间接指令转移，则需要用静态反跟踪，对于堆栈反跟踪一类单步调试时会修改目标值的部分，记录原值并输入IP手动跳转；对于通过地址间接转移，则需要观察数据段中保存的地址以确定转移到新语句后是否能继续正常执行，直至破译密码加密代码或直接破译隐藏的目标信息。

5. 通过示例熟悉反跟踪的一般操作步骤，在反跟踪时可做到较为容易的辨别，具体例子见3.5.2；对于堆栈反跟踪，修改中断矢量表一类单步调试时会修改目标值的部分，记录原值并在call或jump处输入正确的IP并手动跳转；对于通过地址间接转移，则需要观察数据段中保存的地址以确定转移到新语句后是否能继续正常执行，直至破译密码加密代码或直接破译隐藏的目标信息;对于计时中断，需要手动修改IP跳过两次计时比较的部分。

# 参考文献

[1]王元珍，曹忠升，韩宗芬《80X86汇编语言程序设计》“第6章 输入/输出和WIN32编程”