

# 汇编语言程序设计实验报告

---

## 目录

|       |                  |    |
|-------|------------------|----|
| 1     | 实验目的与要求.....     | 1  |
| 2     | 实验内容.....        | 1  |
| 3     | 实验过程.....        | 2  |
| 3.1   | 任务 1.....        | 2  |
| 3.1.1 | 设计思想及存储单元分配..... | 2  |
| 3.1.2 | 源程序.....         | 2  |
| 3.1.3 | 实验步骤.....        | 3  |
| 3.1.4 | 实验记录与分析.....     | 3  |
| 3.2   | 任务 2.....        | 6  |
| 3.2.1 | 设计思想及存储单元分配..... | 6  |
| 3.2.2 | 流程图.....         | 6  |
| 3.2.3 | 源程序.....         | 7  |
| 3.2.4 | 实验步骤.....        | 8  |
| 3.2.5 | 实验记录与分析.....     | 8  |
| 3.3   | 任务 3.....        | 11 |
| 3.3.1 | 设计思想及存储单元分配..... | 11 |
| 3.3.2 | 流程图.....         | 12 |
| 3.3.3 | 源程序.....         | 12 |
| 3.3.4 | 实验步骤.....        | 14 |
| 3.3.5 | 实验记录与分析.....     | 14 |
| 3.4   | 任务 4.....        | 16 |
| 3.4.1 | 设计思想及存储单元分配..... | 16 |
| 3.4.2 | 流程图.....         | 17 |
| 3.4.3 | 源程序.....         | 18 |
| 3.4.4 | 实验步骤.....        | 23 |
| 3.4.5 | 实验记录与分析.....     | 23 |
| 3.5   | 任务 5.....        | 28 |
| 3.5.1 | 设计思想及存储单元分配..... | 28 |
| 3.5.2 | 实验步骤.....        | 28 |
| 3.5.3 | 实验记录与分析.....     | 28 |
| 4     | 总结与体会.....       | 33 |
|       | 参考文献.....        | 35 |

---

# 汇编语言程序设计实验报告

---

## 1 实验目的与要求

- (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。

## 2 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：(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。

**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）。

---

# 汇编语言程序设计实验报告

---

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。

## 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。

## 3 实验过程

### 3.1 任务 1

#### 3.1.1 设计思想及存储单元分配

设计思想：根据题目要求的三种方法，获取中断入口地址。具体操作在实验记录里有。

寄存器使用：在使用方法 3 时，利用 **bx** 存放段地址，**cx** 存放偏移地址。

#### 3.1.2 源程序

```
. 386
assume cs:code, ss:stack
stack segment use16 stack
    db 200 dup(0)
stack ends
code segment use16
start: xor ax, ax
       mov ds, ax

       mov ax, 3501h; 取 1h 中断入口地址
       int 21h

       mov ax, 3510h; 取 10h 中断入口地址
       int 21h

       mov ah, 4ch
       int 21h
code ends
end start
```

```
. 386
assume cs:code, ss:stack
stack segment use16 stack
    db 200 dup(0)
stack ends
code segment use16
start: xor ax, ax
       mov ds, ax
       xor bx, bx
       xor cx, cx
       mov bx, ds:[6h]; 1h 中断
```

# 汇编语言程序设计实验报告

```
mov cx, ds:[4h]
mov bx, ds:[42h];10h 中断
mov cx, ds:[40h]
mov ah, 4ch
int 21h
code ends
end start
```

## 3.1.3 实验步骤

- 1.直接运行 TD，首先计算中断类型码 1H,10H 所调用的中断矢量表位置的段地址和偏移地址，分别是 6H,4H;42H,40H。在数据段中输入对应地址，查看里面的数据。
- 2.准备上机实验环境，编译，连接程序 4code1\_1,4code1\_2。
- 3.根据不同的方法，利用 td 查看对应的数据。
- 4.比较三种方法得到的数据是否相同。
- 5.完成思考题。

## 3.1.4 实验记录与分析

1. 实验环境条件：P3 1GHz，256M 内存；WINDOWS 10 下 DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0；LINK.EXE 5.2；TD.EXE 5.0。

2.获取中断处理程序的入口地址

(1) 方法 1：根据中断类型码计算得到的其中断处理程序在中断矢量表中的位置，直接在数据段里观察。观察到 6H 存放数据为 0070，4H 存放数据为 0008，如图 3.1.1 所示；观察到 42H 存放数据为 F100，40H 存放数据为 0300，如图 3.1.2 所示。

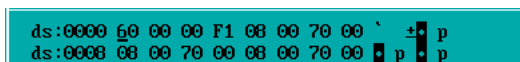


图 3.1.1 1H 中断在中断矢量表中位置

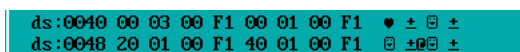


图 3.1.2 10H 中断在中断矢量表中位置

(2) 方法 2：直接调用 1H，10H 号中断，观察功能相应的出口参数。使用 35h 号系统功能调用，入口参数为 al：中断类型码；出口参数为 es：[bx]为入口地址（对应程序 4code1\_1）。观察下面两图，发现得到的 1H 中断入口地址为 07FA：0B1A，与（1）中得到的数据不相同。后来经过查阅资料，得出：td 执行后会修改 1 和 3 号等中断的矢量。发现得到的 10H 中断入口地址为 F100:0300，与（1）中所得数据相同。

# 汇编语言程序设计实验报告

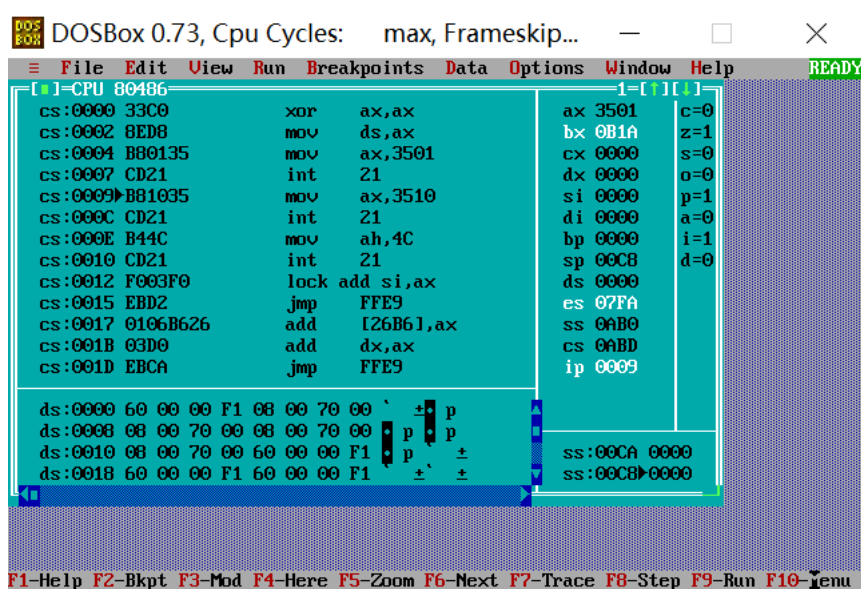


图 3.1.3 1H 中断经过系统功能调用得到的入口地址

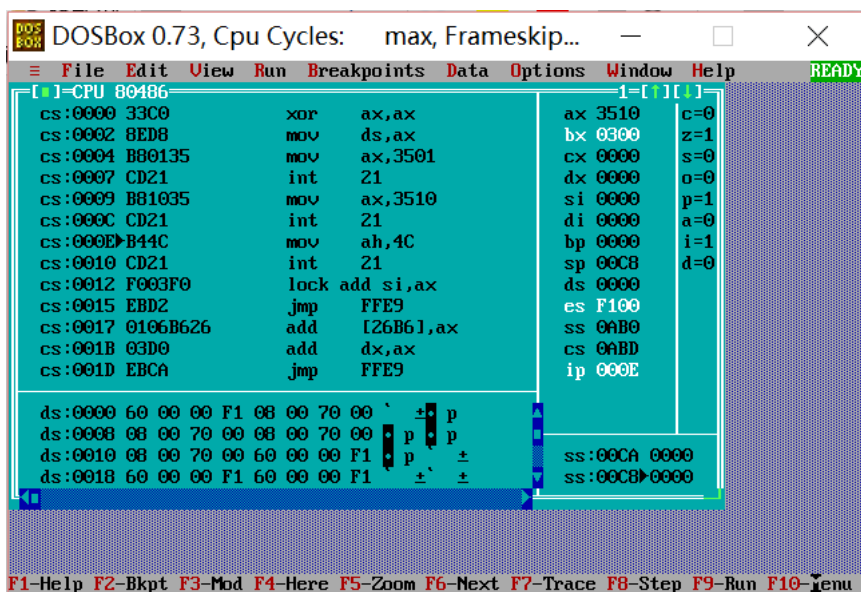


图 3.1.4 10H 中断经过系统功能调用得到的入口地址

(3) 方法 3: 直接读取相应内存单元, 观察读到的数据。观察下面两图, 发现得到的 1H 中断入口地址为 07FA: 0B1A, 与 (1) 中得到的数据不相同。原因在 (2) 中已经给出。发现得到的 10H 中断入口地址为 F100: 0300, 与 (1) 中所得数据相同。

# 汇编语言程序设计实验报告

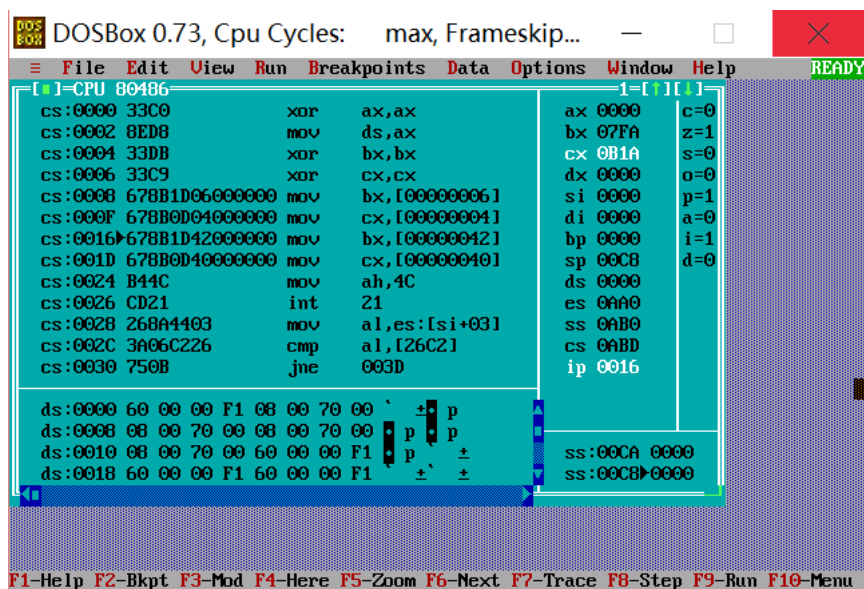


图 3.1.5 直接读取 1H 中断在中断矢量表中的内存单元的内容

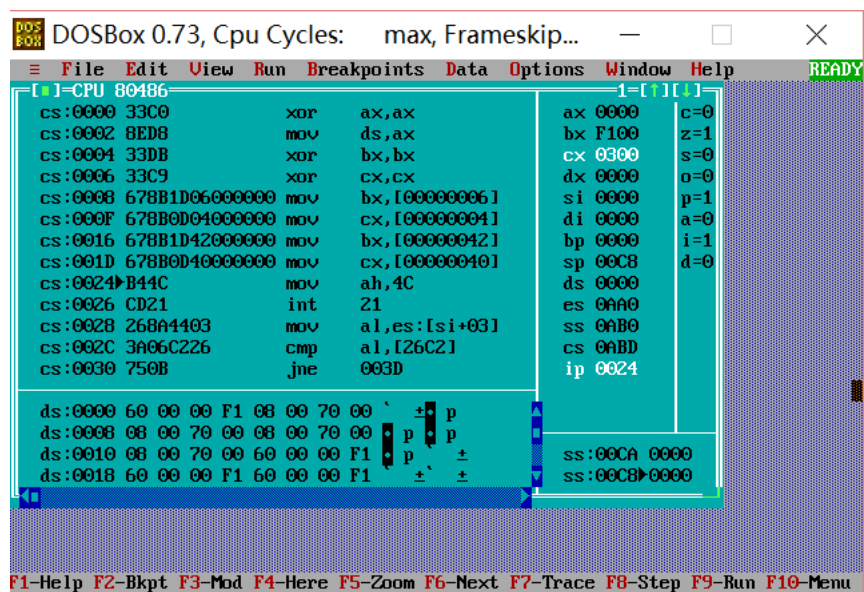


图 3.1.6 直接读取 10H 中断在中断矢量表中的内存单元的内容

## 3. 思考题

(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。

将 ds 值设为 0，直接转到 ds:0 数据区，0000:0000 到 0000:03FF 的空间存放中断矢量表。

(2) 如何计算某个中断入口在中断向量表内的偏移地址？

中断类型码  $n \times 4$  开始存放的子单元为入口地址的偏移地址。

(3) 程序中如何使用系统功能调用获取中断入口地址？

利用 35h 号调用，入口参数 al：中断类型码，出口参数：es: [bx]为入口地址。

(4) 程序中如何通过直接内存读取获取中断入口地址？

直接将内存中的值放入寄存器里，观察寄存器的值。

(5) 用 TD 把中断矢量表里的中断矢量的值随意改成其他值会有什么现象发生？

会使程序调用这些中断时，由于找不到正确的入口地址而使程序出错。

# 汇编语言程序设计实验报告

## 3.2 任务 2

### 3.2.1 设计思想及存储单元分配

设计思想：编写接管键盘中断的中断服务程序，将中断矢量表中中断号为 16h 开始的中断位置的 2 个字修改为新的中断程序的偏移地址和段首址，并保存旧的键盘驱动中断程序的偏移地址和段首址。在新的中断程序中，要先判断用户有没有调用要求修改的中断程序，如果没有，则直接进入原中断程序，如果调用了，就进入新修改的中断程序。改变现实大小写只要改变出口参数中 AL 的值即键入字符的 ASCII 码即可实现。

### 3.2.2 流程图

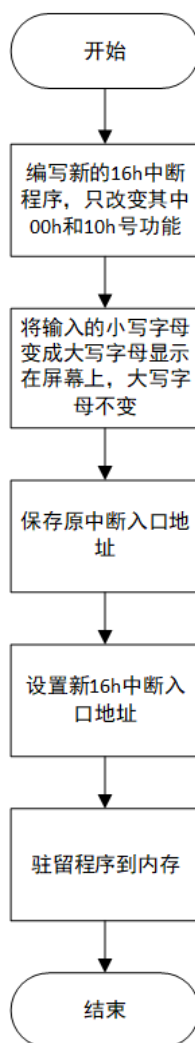


图 3.2.1 程序流程图

---

# 汇编语言程序设计实验报告

---

## 3.2.3 源程序

```
;一个接管键盘中断的中断服务程序并驻留内存
.386
assume cs:code,ds:data,ss:stack
data segment use16
data ends
code segment use16
old16h dw ?,?
new16h:cmp ah,00h
        je low_to_up
        cmp ah,10h
        je low_to_up
        jmp dword ptr old16h ;进入原中断
low_to_up:
        pushf
        call dword ptr old16h
        cmp al,97 ;输入的 ascii 码在小写字母区间
        jnb continuecmp
        jmp break
continuecmp:
        cmp al,122
        jna to_upper
        jmp break
to_upper:
        and al,11011111b
break: iret
start: xor ax,ax
        mov ds,ax
        mov ax,ds:[16h*4]
        mov old16h,ax ;保存偏移地址
        mov ax,ds:[16h*4+2]
        mov old16h+2,ax ;保存段地址
        cli
        mov word ptr ds:[16h*4],offset new16h ;新地址
        mov ds:[16h*4+2],cs
        sti
        mov dx,offset start+15 ;驻留
        shr dx,4
        add dx,10h
        mov al,0
        mov ah,31h
        int 21h
code ends
stack segment use16 stack
        db 200 dup(0)
stack ends
end start

;卸载
.386
assume cs:code,ss:stack
data segment use16
data ends
code segment use16
```



---

# 汇编语言程序设计实验报告

---

```
old16h dw ?,?
new16h: cmp ah, 00h
        je low_to_up
        cmp ah, 10h
        je low_to_up
        jmp dword ptr old16h ;进入原中断
low_to_up:
        pushf
        call dword ptr old16h
        cmp al, 97 ;输入的 ascii 码在小写字母区间
        jnb continuecmp
        jmp break
continuecmp:
        cmp al, 122
        jna to_upper
        jmp break
to_upper:
        and al, 11011111b
break:  iret
start:  xor ax, ax
        mov ds, ax
        mov ax, ds: [16h*4]
        mov old16h, ax ;保存偏移地址
        mov ax, ds: [16h*4+2]
        mov old16h+2, ax ;保存段地址
        cli
        mov word ptr ds: [16h*4], 11e0h ;卸载
        mov word ptr ds: [16h*4+2], 0f000h
        sti
        mov dx, offset start+15 ;驻留
        shr dx, 4
        add dx, 10h
        mov al, 0
        mov ah, 31h
        int 21h
code   ends
stack  segment use16 stack
        db 200 dup(0)
stack  ends
end start
```

## 3.2.4 实验步骤

1. 准备上机实验环境，编辑，汇编，连接文件。
2. 在 DOS 命令行下键入小写字母，观察屏幕是否显示为大写，键入大写时，观察是否不变。
3. 执行 TD，在代码区输入指令“mov AX,0”，看是否都变了大写。
4. 完成选做及思考题。

## 3.2.5 实验记录与分析

1. 实验环境条件：P3 1GHz，256M 内存；WINDOWS 10 下 DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0；LINK.EXE 5.2；TD.EXE 5.0。

## 汇编语言程序设计实验报告

2. 在 DOS 命令行下键入小写字母 hello，接着输入空格，然后键入大写字母 HELLO，观察屏幕上显示情况。

```
C:\>4code2_1
C:\>HELLO HELLO
```

图 3.2.2 执行程序后屏幕显示情况

3.在 TD 代码区输入指令“mov AX, 0”，发现小写没有变成大写。说明中断程序不是会影响到所有程序。

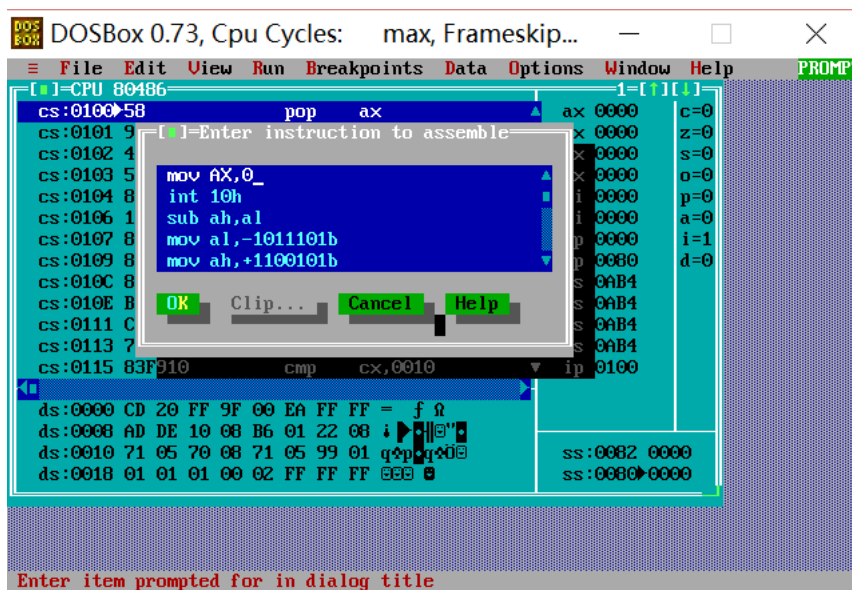


图 3.2.3 观察 TD 是否被影响

4.选做：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。运行程序，观察到中断服务程序确实卸载成功了。键盘恢复到了原来的状态。

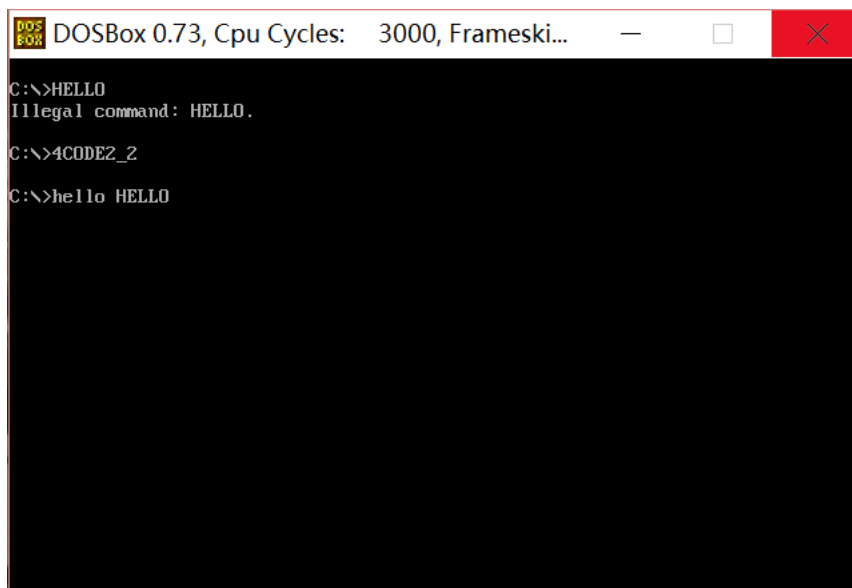


图 3.2.4 执行中断服务程序的卸载程序

5.思考题

## 汇编语言程序设计实验报告

(1) 有哪两种方式进入原中断服务程序？

有 `jmp` 和 `call` 指令。比如 `Jmp dword ptr 原中断程序`; `Call dword ptr 原中断程序`。

(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，请给出这种“其他”调试方法的描述并具体实施一下。

可以利用 TD 调试程序。

(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？

编写的中断驻留程序执行后能正常回到 DOS，DOS 能正常工作。在实验中，我尝试过重复驻留多次，但实验没有什么特别的现象，猜测如果重复驻留多次，总有时候会占满主存空间，从而出错。

(4) 同时打开另外一个虚拟 DOS 窗口，键盘大小写是否被替代？

同时打开另一个虚拟 DOS 窗口，键盘大小写没有被替代。

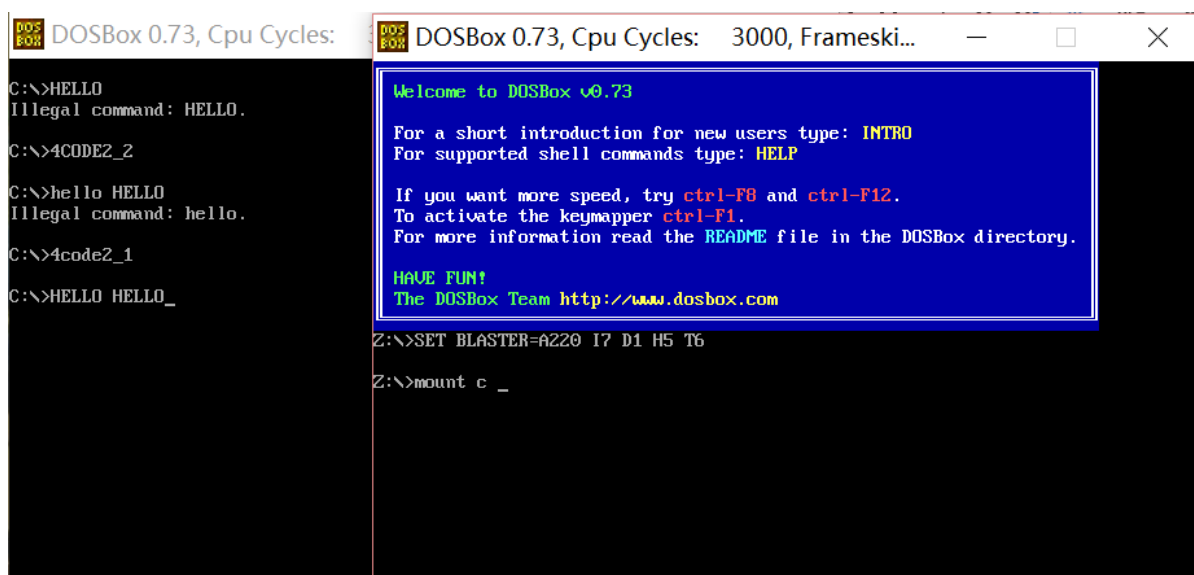


图 3.2.5 另一个 dos 窗口情况

(5) 如何确定自己编写的中断处理程序已被系统调用？

可以利用 `td` 调试，查找调用的中断处理程序在中断矢量表的位置，看其中的内容是否改变了。

(6) 选作的要求应该如何实现，如何找到保存的原中断入口地址？如何保证不会错误恢复？

选做的具体实现在代码段已有注释。如何找到原中断入口，任务 1 已经给出了提示。值得注意的是，如果将其保存在某个变量里面，这个变量的值很可能随着程序的进行而改变，而直接给中断矢量表赋值则不会出现类似情况。

# 汇编语言程序设计实验报告

---

## 3.3 任务 3

### 3.3.1 设计思想及存储单元分配

设计思想：利用 1 号系统功能调用读入字符，1 号调用将输入的字符转换成 ASCII 码放入 al 中。然后将 ASCII 码做一定处理，当做数字信息然后利用这个信息查看 CMOS 里的信息。由于 CMOS 里时间信息以压缩 BCD 码存放，将其进行分离，然后分别显示这两个 BCD 码转换成 16 进制后的信息。

寄存器使用：al 用于读字符，放 CMOS 信息，在乘法除法里被使用。

Bx 用于接收 ax 中信息，显示。

# 汇编语言程序设计实验报告

## 3.3.2 流程图

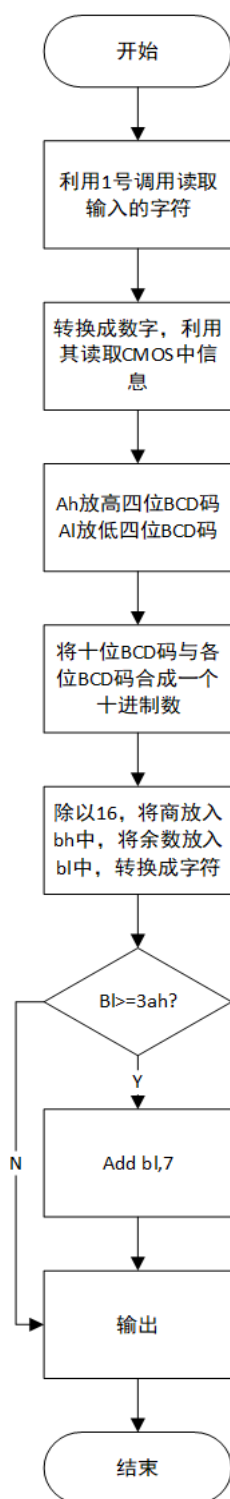


图 3.3.1 程序流程图

## 3.3.3 源程序

---

# 汇编语言程序设计实验报告

---

```
assume ss:stack,cs:code
stack segment use16 stack
    db 200 dup(0)
stack ends
code segment use16
start:
    mov ax,0
    mov ds,ax
    mov ah,1 ;读入一个字符，其 ASCII 码值放入 al
    int 21h
    sub al,30h ;输入的是字符，转换成数字
    out 70h,al
    in al,71h ;压缩 BCD 码
    mov ah,al
    and al,0fh ;al 放低四位 BCD 码
    shr ah,4 ;ah 放高四位 BCD 码
    mov bx,ax
    mov dl,0ah ;换行
    mov ah,2
    int 21h
    mov dl,0dh
    mov ah,2
    int 21h
    mov al,bh ;十位
    mov cl,10
    mul cl
    add al,bl
    mov ah,0
    mov cl,16 ;16 进制
    div cl
    mov bh,al
    mov bl,ah
    add bx,3030h ;转换成字符显示
    cmp bl,3ah ;超过了 9
    jnb over

output:
    mov dl,bh;16 进制输出
    mov ah,2
    int 21h
    mov dl,bl
    mov ah,2
    int 21h
    mov dl,'h'
    mov ah,2
    int 21h
    mov dl, 0ah
    mov ah, 2
    int 21h
    mov dl, 0dh
    mov ah, 2
    int 21h

    mov ah,4ch
    int 21h
over:
    add bl,7
```

---

# 汇编语言程序设计实验报告

```
jmp output
code ends
end start
```

## 3.3.4 实验步骤

1. 准备上机实验环境，编辑，汇编，连接文件。
2. 运行程序。在命令行状态下分别输入 0,2,4,6,8,9 分别显示硬件时间的秒，分，时，星期几，日期，月份，年份。也可以输入其他字符，看看输出。
3. 完成思考题。

## 3.3.5 实验记录与分析

1. 实验环境条件：P3 1GHz，256M 内存；WINDOWS 10 下 DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0；LINK.EXE 5.2；TD.EXE 5.0。
2. 运行程序，分别输入 0,2,4,6,7,8,9 看看显示的信息是否与系统时间一致。然后输入其他字符，观察输出。结果如下图，发现一致。

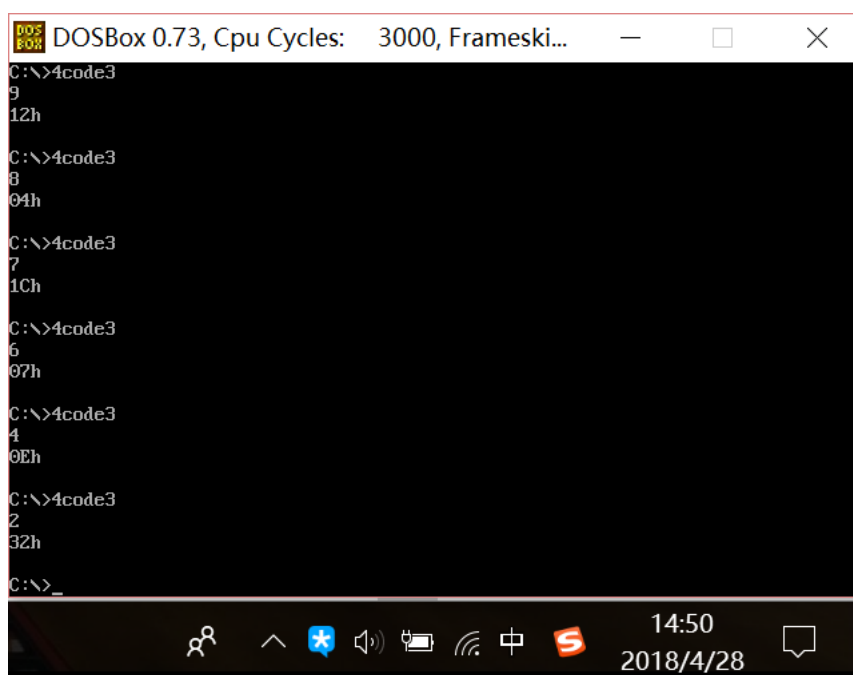


图 3.3.2 读取后的信息



图 3.3.3 读取后的信息

# 汇编语言程序设计实验报告

```
C:\>4code3
}
50h

C:\>4code3
Z
1Ah

C:\>4code3
Z
00h

C:\>
```

图 3.3.4 输入其他字符读取后的信息

## 3.思考题

(1) 如何直接在 TD 下使用 IN/OUT 指令获取 CMOS 数据？

向端口号 70h 写入要访问 CMOS 中的哪个字节，再将端口号 71h 中的字节单元数据送入寄存器 al，查看输出。

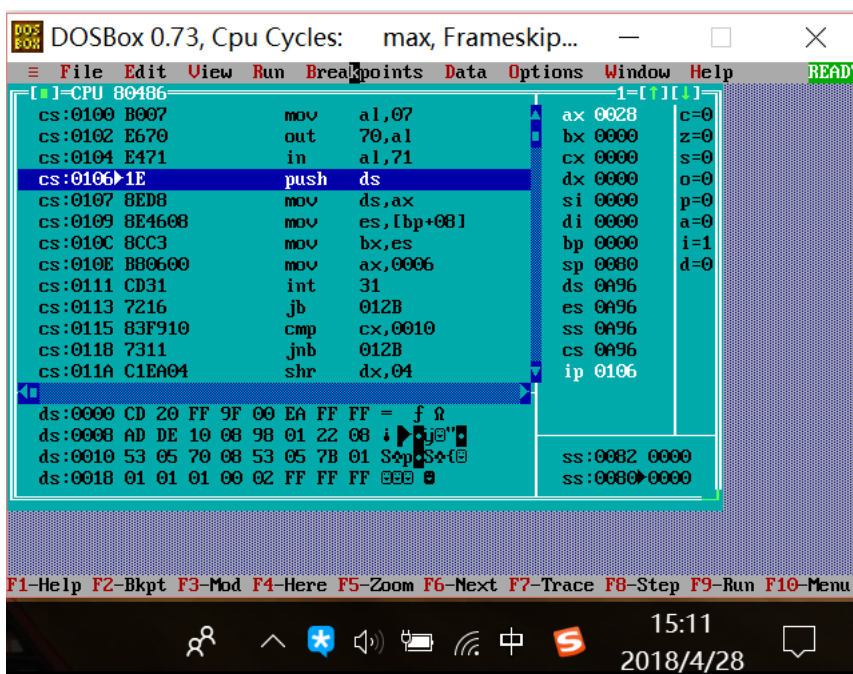


图 3.3.5 输入其他字符读取后的信息

(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？

一个 BCD 码占 4 位，而一个字节有 8 位。若把二个 BCD 码放在一个字节中，就叫压缩 BCD 码。比如如上图 3.3.5 中 al 中数据为 28，这就是压缩 BCD 码的格式，al 中二进制格式应该为 00101000，高四位为一个 BCD 码，相当于十进制中的 2，低四位为一个 BCD 码，相当于十进制中的 8。于是就显示成上图所示的情况。



# 汇编语言程序设计实验报告

---

## 3.4 任务 4

### 3.4.1 设计思想及存储单元分配

设计思想：将数据段中的密码进行加密，加密方式采用异或运算，密钥与老板姓名有关，同时使用随机数填充密码区，防止破解者猜到密码的长度。将商品的进货价与密码有关的密钥进行异或运算加密。然后在原来的基础上增加一个判断输入次数的功能。在实现反跟踪功能时，使用了间接转移反跟踪，中断矢量表反跟踪，堆栈检查反跟踪的方法。在一些关键语句附近使用反跟踪方法，增加一些无关语句扰乱视线，增加了破解者的破解难度。具体各个方法的介绍在实验记录中给出。在计算利润率模块中将进货价解密，然后计算。

寄存器使用与实验 3 任务 1 相同，在此不赘述。

# 汇编语言程序设计实验报告

## 3.4.2 流程图

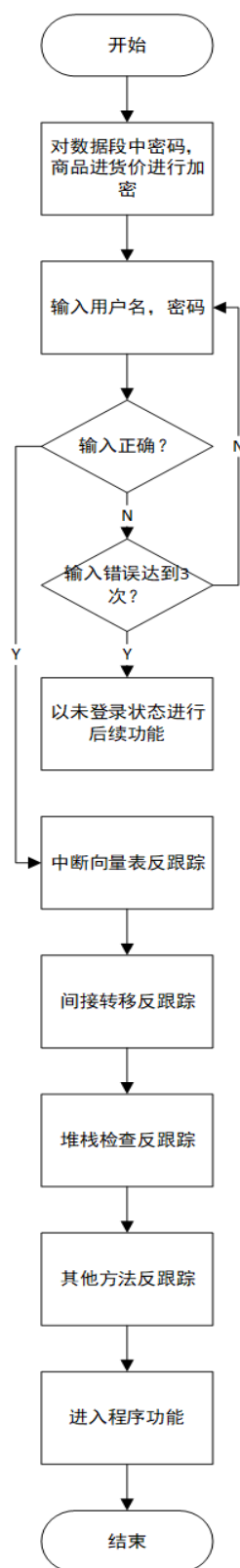


图 3.4.1 程序流程图

# 汇编语言程序设计实验报告

## 3.4.3 源程序

```
public  gbl,prol,find,gal,prorank,cost,prank,total,price,f2t10,gal,shop2,gbl,  shop1,  sold,
profit,  gname,query_1,query_2,modify_item
extrn fresh:far,rank:far,_outall:far
.386
assume  cs:code, ds:data, ss:stack
stack  segment      use16      para      stack      'stack'
      db 200 dup(0)
stack  ends
data    segment      use16      para      public  'data'
menu1 db 0dh,0ah,'-----menu-----',
      db 0dh,0ah,'  1 = inquire articles information'
      db 0dh,0ah,'  2 = modify articles information'
      db 0dh,0ah,'  3 = calculate the average rate of profit'
      db 0dh,0ah,'  4 = calculate the rank of profit'
      db 0dh,0ah,'  5 = output all articles information'
      db 0dh,0ah,'  6 = exit'
      db 0dh,0ah,'-----',0dh,0ah,'$'
menu2 db 0dh,0ah,'-----menu-----',
      db 0dh,0ah,'  1 = inquire articles information'
      db 0dh,0ah,'  2 = exit'
      db 0dh,0ah,'-----',0dh,0ah,'$'
menu3 db '1.modify shop1',0dh,0ah
      db '2.modify shop2',0dh,0ah,'$'
shop1 db 'shop1:',0dh,0ah,'$'
shop2 db 'shop2:',0dh,0ah,'$'
gname db '  name:$'
price db '  price:$'
total db '  total:$'
sold db '  sold:$'
cost db '  cost:$'
profit db '  profit:$'
prank db '  rank:$'
nameinput db 0ah,0dh,'Please input your name: ','$'
pwdinput db 0ah,0dh,'Please input your code: ','$'
iteminput db 0ah,0dh,'Please input the article name you want to inquire:',0dh,0ah,'$'
faillog db 0ah,0dh,'Failed to log in!','$'
changev db '-->$'

bname  db  'chaofan',0,0,0 ;用户名

;对密码进行加密
bpass  db  'm' xor 'f' ;异或运算加密
      db  'e' xor 'f'
      db  'i' xor 'f'
      db  0a3h,0d9h,0f5h ;随机数填充密码区到6个字符,防止被猜到
bpasslen db 3 xor 'm'

n      equ 3 ;商品个数
s1     db  'shop1',0
gal    db  'pen',7 dup(0) ;商品名称
      dw  35 xor 'e',56,70,25,? ;进价、售价、总数、已售
ga2    db  'book',6 dup(0)
      dw  12 xor 'e',30,25,5,?
ga3    db  'kindle',4 dup(0)
```

# 汇编语言程序设计实验报告

```

                dw 58 xor 'e', 170, 20, 11, ?
s2             db 'shop2', 0
gb1           db 'pen', 7 dup(0)
                dw 35 xor 'e', 50, 30, 24, ?
gb2           db 'book', 6 dup(0)
                dw 12 xor 'e', 28, 20, 15, ?
gb3           db 'kindle', 4 dup(0)
                dw 90 xor 'e', 280, 30, 12, ?
auth          db 0
sign          db ?
prol          dd 0
prorank       dw n dup(0)
buf           db 12 dup(?)
inname        db 10
                db 0
                db 10 dup(0)
inpwd         db 6
                db 0
                db 6 dup(0)
initem        db 10
                db 0
                db 10 dup(0)
innum         db 5
                db 0
                db 5 dup(0)

input_count   db 0 ;判断输入次数

E1 DW f3_6      ;地址表（用于间接转移反跟踪）
CMPPASS DW cmpb
CMPADMIN DW cmpa

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
;dos10
read macro a
    push ax
    push dx
    lea dx, a
    mov ah, 10
    int 21h
    pop dx
    pop ax
endm
;dos9
write macro a
    push ax
    push dx
    lea dx, a
    mov ah, 9
    int 21h
    pop dx
    pop ax
endm
```

# 汇编语言程序设计实验报告

```
;换行
crlf macro
    mov dl,0ah
    mov ah,2h
    int 21h
endm

code segment    use16    para    public    'code'
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

    jmp login
failtologin:
    mov auth,0
    write faillog
    inc input_count ;错误次数加 1
    cmp input_count,3
    je user
    crlf
login: write nameinput ;提示输入姓名
    crlf
    read inname
    crlf
    cmp byte ptr [inname+1],0 ;仅输入回车
    je user ;普通用户
    cmp byte ptr[inname+2],'q' ;若输入为 q
    je continuejudge ;继续判断
    cmp byte ptr[inname+1],7 ;长度不一致
    jne failtologin
password:
    write pwinput ;提示输入密码
    crlf
    read inpwd
    crlf
    mov cl,[bpasslen]
    xor cl,[inpwd+2]
```

# 汇编语言程序设计实验报告

```
cmp byte ptr[inpwd+1],cl ;长度不一致
jnz failtologin
mov cl,[inname+1]
mov ch,0

cli ;堆栈检查反跟踪
push CMPADMIN
;cmpa 的地址压栈

mov si,0

pop ax
mov bx,[esp-2] ;把栈顶上面的字（cmpa 的地址）取到
sti
jmp bx ;如果被跟踪，将不会转移到 cmpa

db 'hello!'
db 'A Great try'

cmpa: mov dl,[bname+si] ;判断用户名是否正确
mov bl,[inname+si+2]
cmp bl,dl
jnz failtologin
inc si
loop cmpa

mov cl,[inpwd+1]
mov ch,0
cli ;堆栈检查反跟踪
push CMPPASS ;cmpp 的地址压栈

mov si,0
pop ax
mov bx,[esp-2] ;把栈顶上面的字（cmpp 的地址）取到
sti
jmp bx ;如果被跟踪，将不会转移到 cmpp

db 'How to go?' ;定义的冗余信息，扰乱视线

cmpp: mov dl,[bpass+si] ;判断密码是否正确
mov bl,[inpwd+si+2]
xor bl,[inname+6] ;将输入的密码进行异或运算,不直接显示密文
cmp bl,dl
jnz failtologin
inc si
loop cmpp
mov auth,1 ;状态改变

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 'do you get?'

manager:
write menu1
```

---

## 汇编语言程序设计实验报告

---

```
        mov ah, 1
        int 21h
        crlf
        cmp al, '1'
        je f3_1
        cmp al, '2'
        je f3_2
        cmp al, '3'
        je f3_3
        cmp al, '4'
        je f3_4
        cmp al, '5'
        je f3_5
        cmp al, '6'
        je f3_6
        jmp manager
user:   write    menu2
        mov ah, 1
        int 21h
        crlf
        cmp al, '1'
        je f3_1
        cmp al, '2'
        je f3_6
        jmp user

NEWINT: iret
TESTINT: jmp manager
...
f3_6:
        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
在 fresh 模块中
...
mov cx, [si]      ;将进货价存入 ax
xor cl, 'e'
mov ebx, ecx      ;将进货价存入 ebx 中
...
```

---

# 汇 编 语 言 程 序 设 计 实 验 报 告

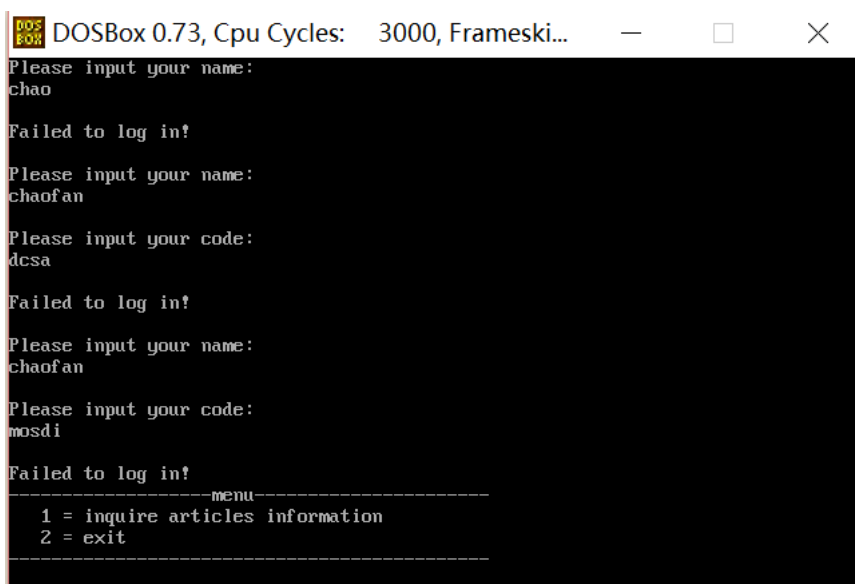
---

## 3.4.4 实验步骤

- 1.准备上机实验环境，编辑，汇编，连接文件。
- 2.运行程序，连续错误输入用户名和密码 3 次，则直接按照未登录状态进入后续功能。然后检查一些功能是否正常运行，特别是检查商品的进货价是否被加密了。
- 3.使用 TD 调试程序，检查自己的反跟踪是否有效。
- 4.完成思考题。

## 3.4.5 实验记录与分析

1. 实验环境条件：P3 1GHz，256M 内存；WINDOWS 10 下 DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0；LINK.EXE 5.2；TD.EXE 5.0。
2. 运行程序。
  - (1) 连续错误输入用户名和密码三次，观察到确实以未登录状态进入后续功能。



```
DOSBox 0.73, Cpu Cycles: 3000, Frameski...
Please input your name:
chao

Failed to log in!

Please input your name:
chaofan

Please input your code:
dcsa

Failed to log in!

Please input your name:
chaofan

Please input your code:
mosdi

Failed to log in!

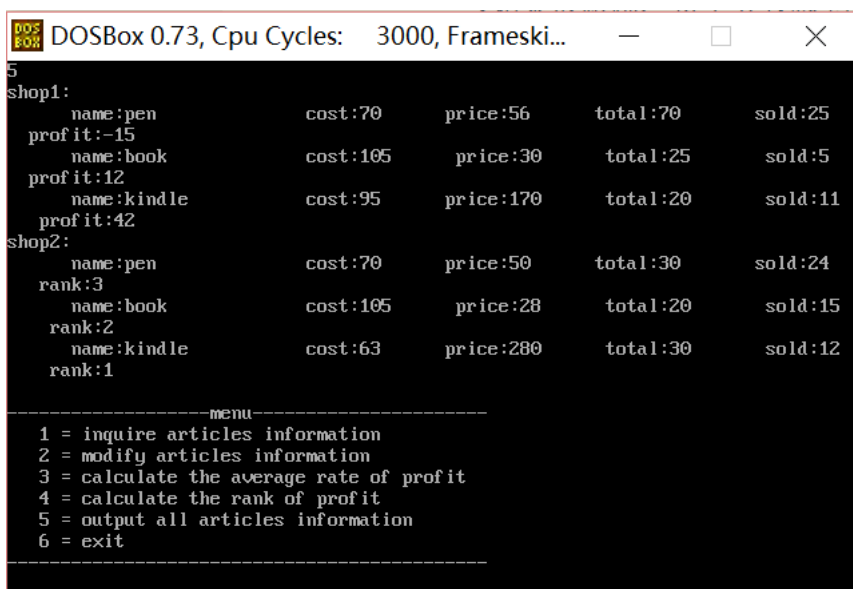
-----menu-----
1 = inquire articles information
2 = exit
-----
```

图 3. 4. 2 连续错误输入用户名和密码三次

- (2) 登录，看到商品的进货价确实是以密文方式存放在数据段，并且利润率计算正常。（说明在计算利润率程序中解密了）



# 汇编语言程序设计实验报告



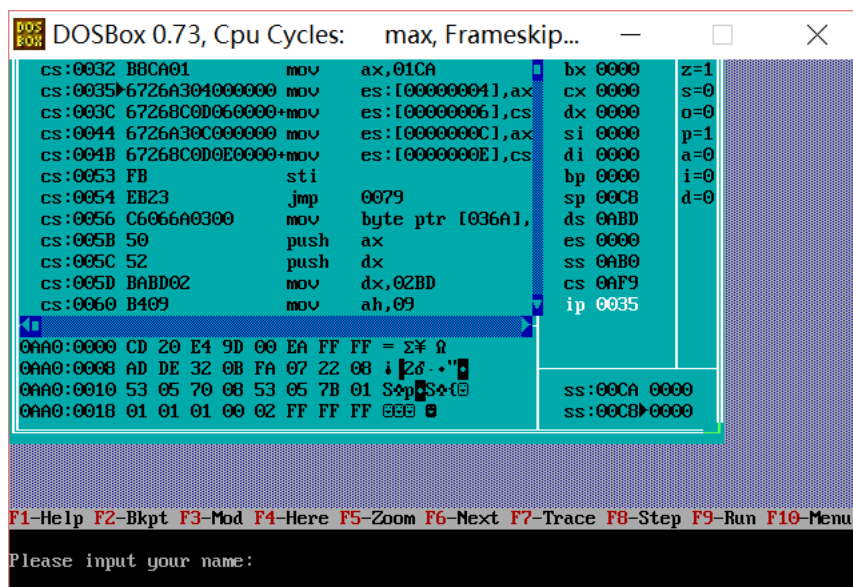
```
DOSBox 0.73, Cpu Cycles: 3000, Frameski...
5
shop1:
  name:pen      cost:70    price:56    total:70    sold:25
  profit:-15
  name:book     cost:105   price:30    total:25    sold:5
  profit:12
  name:kindle   cost:95    price:170   total:20    sold:11
  profit:42
shop2:
  name:pen      cost:70    price:50    total:30    sold:24
  rank:3
  name:book     cost:105   price:28    total:20    sold:15
  rank:2
  name:kindle   cost:63    price:280   total:30    sold:12
  rank:1

-----menu-----
1 = inquire articles information
2 = modify articles information
3 = calculate the average rate of profit
4 = calculate the rank of profit
5 = output all articles information
6 = exit
-----
```

图 3.4.3 商品的进货价以密文方式存放在数据段

3.调试程序，检查反跟踪是否有效。

(1) 中断矢量表反跟踪



```
DOSBox 0.73, Cpu Cycles: max, Frameskip...
cs:0032 B8CA01 mov ax,01CA bx 0000 z=1
cs:0035 6726A304000000 mov es:[00000004],ax cx 0000 s=0
cs:003C 67268C0D060000+mov es:[00000006],cs dx 0000 o=0
cs:0044 6726A30C000000 mov es:[0000000C],ax si 0000 p=1
cs:004B 67268C0D0E0000+mov es:[0000000E],cs di 0000 a=0
cs:0053 FB sti bp 0000 i=0
cs:0054 EB23 jmp 0079 sp 00C8 d=0
cs:0056 C6066A0300 mov byte ptr [036A], ds 0ABD
cs:005B 50 push ax es 0000
cs:005C 52 push dx ss 0AB0
cs:005D BABD02 mov dx,02BD cs 0AF9
cs:0060 B409 mov ah,09 ip 0035

0AA0:0000 CD 20 E4 9D 00 EA FF FF = Σ¥ ¨
0AA0:0008 AD DE 32 0B FA 07 22 08 i 2d - "
0AA0:0010 53 05 70 08 53 05 7B 01 Sep 34
0AA0:0018 01 01 01 00 02 FF FF FF 888 8

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
Please input your name:
```

图 3.4.4 中断矢量表反跟踪

中断矢量表反跟踪原理：“动态调试工具都会接管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一段代码。有的调试工具会阻止你的程序修改中断矢量表，这时，我们只需要在修改之后再判断一下中断矢量表中对应位置是否改成了我们的程序的入口地址，即可判断是否存在调试工具了。”

具体源代码在上面已经给出，在此不再截图。本次使用的是修改 1 号和 3 号中断矢量在中断中断矢量表中存放的入口地址。先接管中断，然后再检查中断矢量表是否被调试工具修改或恢复，如果

# 汇编语言程序设计实验报告

正常修改了，则继续执行源程序，否则就跳转。

当使用 TD 进行调试程序时，当调试到反跟踪功能时，就会发现整个 dosbox 崩溃了，此时我们无法进行任何操作，dosbox 界面也变得奇怪，最后只能强制结束程序。

## (2) 间接转移反跟踪

```
E1 DW f3_6 ;地址表（用于间接转移反跟踪）
CMPPASS DW cmpmp
CMPADMIN DW cmpa
```

图 3.4.5 间接转移反跟踪

原理：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。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。与这个方法配套使用的就是建立地址表。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。

此方法通常与其他反跟踪方法结合起来使用，具体在下面的堆栈检查反跟踪中会有用到。当破解者想要单步调试程序时，当遇到了反跟踪程序段时，继续执行会跳到其它地方去，从而增加了破解者破解程序的难度。

## (3) 堆栈检查反跟踪

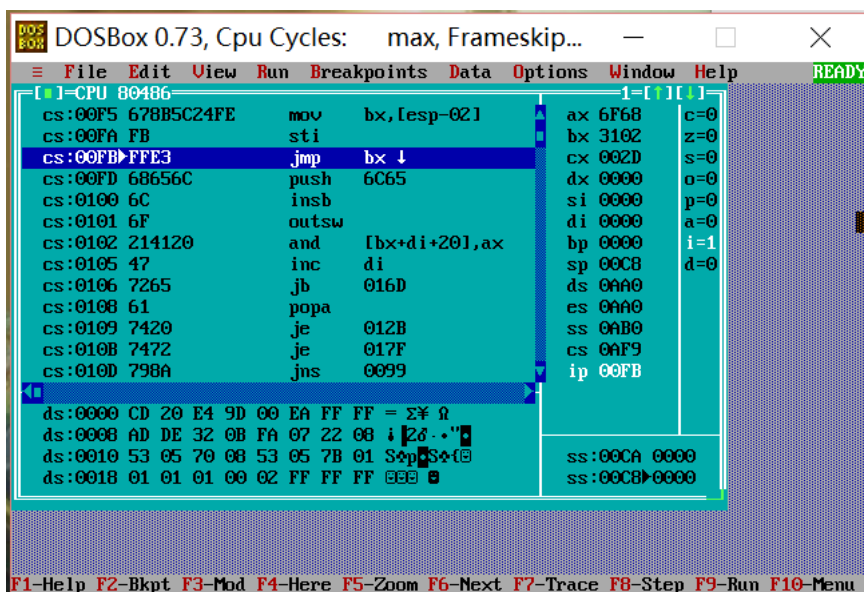


图 3.4.6 堆栈检查反跟踪

## 汇编语言程序设计实验报告

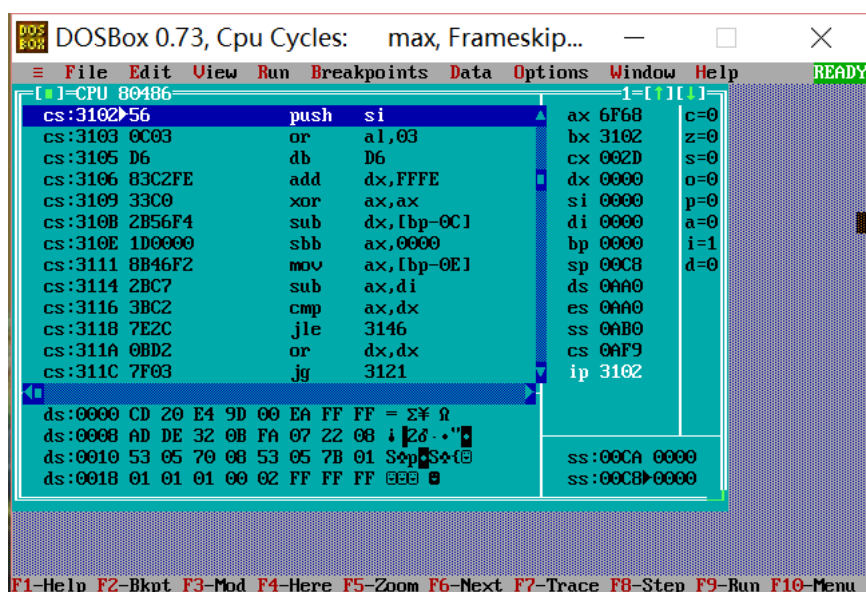


图 3.4.7 堆栈检查反跟踪

原理：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。这种方法经常结合间接转移反跟踪使用。

在判断用户名和判断密码的程序间加入反跟踪程序。在判断用户名的程序间加入反跟踪主要是为了扰乱破解者的视线。因为按照一般的思想，如果遇到了反跟踪，就会认为在这附近一定会有关键程序段，就会浪费大量时间在破解无效信息上，这也加大了破解难度。而在判断密码的程序间加入反跟踪程序则主要是保护有效信息，这是反跟踪程序的主要用法。我认为在无关程序段间加入反跟踪是一种很好的方法，因为这运用了逆向思维，猜到破解者的心理，从而运用这种惯性思维使破解者陷入泥潭。

同时也在反跟踪程序间混入了功能程序，比如判断用户名密码时，要先将 **si** 置 0，否则判断出错，如果不执行反跟踪程序，会让程序的功能无法正常使用，这也迫使破解者无法跳过反跟踪。

上面的图片表示单步调试到反跟踪程序后，指令 `jmp bx` 会让程序跳到一个奇怪的地方，从而有效保护了程序。

### （4）其他方法

除了上述方法，也可以在程序段间加入大量的扰乱信息，为了不让破解者轻易识别反跟踪程序从而跳过。如下图：（其他例子在程序中已经给出）

```
db 'hello!'
db 'A Great try'
```

图 3.4.8 扰乱信息

同时不在程序段中公开密钥，防止破解者通过反汇编直接获取。如下图，将密码长度的密钥不用明文直接显示在程序段，而是通过破解者输入的用户名中的信息进行加密。可以有效防止破解者直接获取密钥。

# 汇编语言程序设计实验报告

```
mov cl,[bpasslen]
xor cl,[inpwd+2]
cmp byte ptr[inpwd+1],cl ;长度不一致
```

图 3.4.9 间接加密

同理，我也在进行密码加密时使用了同样的方法，如下图：

```
mov dl,[bpass+si] ;判断密码是否正确
mov bl,[inpwd+si+2]
xor bl,[inname+6] ;将输入的密码进行异或运算,不直接显示密文
cmp bl,dl
```

图 3.4.10 间接加密

## 4.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？

直接通过 TD 跳到数据段中去寻找疑似密码的信息，由于是明文，可以比较迅速地获取密码。也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) 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？

跟思考题(1)相同，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？

对于密码以明文方式存放在待破解程序中的程序，可以直接利用 TD 调试工具进入数据段查看数据，也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。

对于仅仅需要输入密码且没有出错次数限制的程序，可以编写一个程序，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。

(4) 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？

在我的程序中，中断矢量表反跟踪程序直接去掉 cli, sti 指令即可。因为其中没有涉及到连续更改 ss 和 sp 的值。因为在对 ss 和 sp 操作的时候，如果有中断发生，中断保存现场的操作是将相关寄存器值保存到 ss:sp 指向的地址。如果 ss 或者 sp 没有完成赋值操作，这时候 ss:sp 则不知道指向什么地方。如果将系统或者其他应用的数据覆盖，则会导致系统/应用崩溃。因此在中断没有涉及到更改 ss: sp 的操作时，可以将不关中断，如果涉及到了，则将 ss, sp 换成其他的段。

# 汇编语言程序设计实验报告

## 3.5 任务 5

### 3.5.1 设计思想及存储单元分配

设计思想：根据一些常见的加密方式，思考出对应的策略，使用 td 调试加密程序，逐步破解出重要信息。具体步骤在实验记录中给出。

### 3.5.2 实验步骤

1. 准备上机实验环境，编辑，汇编，连接文件。
2. 使用 TD 调试程序，遇到反跟踪则寻找对应策略，直到破解出全部信息。
3. 完成思考题。

### 3.5.3 实验记录与分析

1. 实验环境条件：P3 1GHz，256M 内存；WINDOWS 10 下 DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0；LINK.EXE 5.2；TD.EXE 5.0。

2. 使用 TD 调试待破解程序。

(1) 直接进入数据段查看信息。

```
ds:FFF8 00 00 00 00 00 00 00 00
ds:0000 44 45 4E 47 59 4F 4E 47 DENGYONG
ds:0008 59 55 42 44 46 48 4A 4C YUBDFHJL
ds:0010 10 39 12 53 48 4F 50 31 9*SHOP1
```

图 3.5.1 数据段

```
ds:0018 00 50 45 4E 00 00 00 00 PEN
ds:0020 00 00 00 65 00 38 00 46 e 8 F
ds:0028 00 45 00 00 00 42 4F 4F E B00
ds:0030 4B 00 00 00 00 00 00 4A K J
```

图 3.5.2 数据段

```
ds:0030 4B 00 00 00 00 00 00 4A K J
ds:0038 00 1E 00 19 00 05 00 00 ▲ ↓ 全
ds:0040 00 58 42 4F 58 00 00 00 XBOX
ds:0048 00 00 00 22 00 C8 00 0A " || o
```

图 3.5.3 数据段

```
ds:0058 50 32 00 50 45 4E 00 00 P2 PEN
ds:0060 00 00 00 00 00 65 00 32 e 2
ds:0068 00 1E 00 18 00 00 00 42 ▲ ↑ B
ds:0070 4F 4F 4B 00 00 00 00 00 00K
```

图 3.5.4 数据段

```
ds:0078 00 4A 00 1C 00 14 00 0F J 全 *
ds:0080 00 00 00 58 42 4F 58 00 XBOX
ds:0088 00 00 00 00 00 22 00 C7 " ||
ds:0090 00 14 00 11 00 00 00 49 全 ◀ I
```

图 3.5.5 数据段

获取用户名 DENGYONGYU，发现后面出现 SHOP1 字样，推测中间可能存在着加密过的密码。后面几张图显示了商品信息，推测里面隐藏着加密后的进货价。



# 汇编语言程序设计实验报告

(2) 进入对方反跟踪程序段

1)

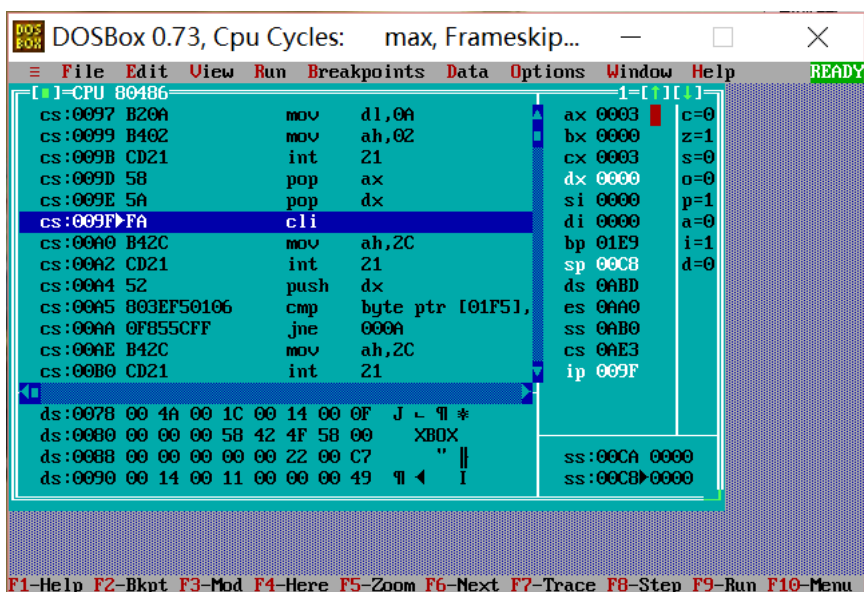


图 3.5.6 对方反跟踪程序

此时，先不继续执行，直接查看反汇编语句。发现一条重要语句。这是一条判断语句，而且是在输入了密码之后，初步推测后面的 06 是密码长度。

0A5 803EF50106 cmp byte ptr [01F5],06

图 3.5.7 对方反跟踪程序中的重要语句

看出这段程序是计时反跟踪程序，直接设置新 ip 跳过这段反跟踪程序。

2)

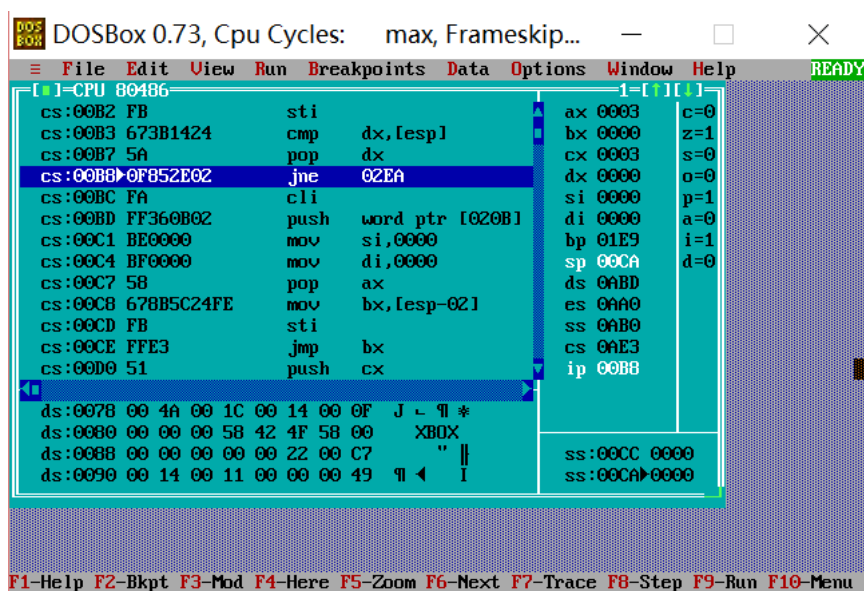


图 3.5.8 对方反跟踪程序中的重要语句

上图可以看出对方再次使用了反跟踪程序，看出后面的 jmp bx 语句，以及 push, pop 语句。推测这是一段堆栈检查反跟踪程序，运行其中的源程序代码，然后跳过即可。

3)

## 汇编语言程序设计实验报告

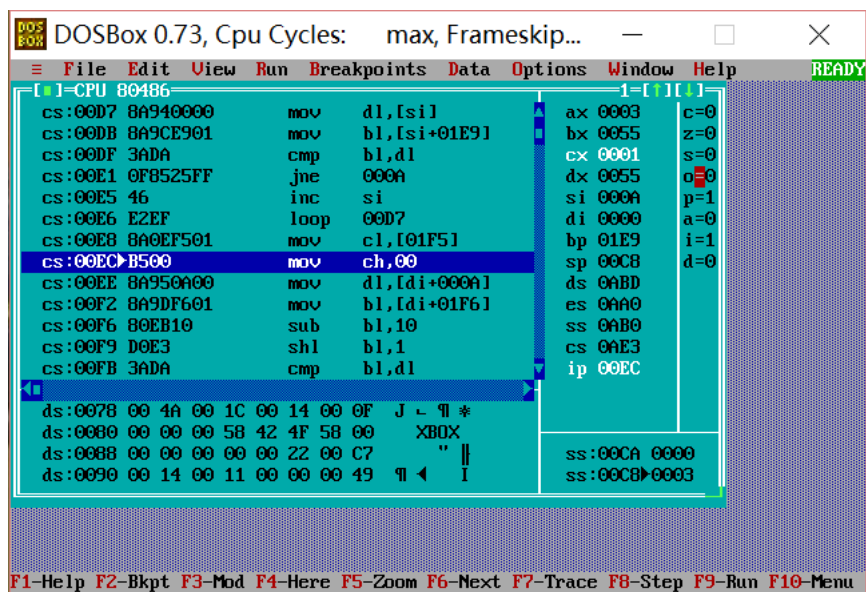


图 3.5.9 对方判断密码程序

由于在这段程序上面执行了 10 次循环,通过检查数据段,发现是与数据段中用户名进行比较。然后这里明显有个循环,且 cx 值为 1,我在刚才输入的密码长度就为 1,因此可以怀疑此段程序为比较密码的程序。继续观察,发现 dl 寄存器里存放的值为疑似密码段的数值,推测 dl 存放的为输入的密码。后面对 b1 做了一系列处理才和 dl 比较,推测这是对方的加密方式。推测加密方式为将原密码减去 10h,然后乘以 2 得到密文。

根据上面得到的疑似密码的信息,以及密码的长度为 6,通过逆运算可以得到对方的密码为 123456。

4) 使用获得的密码进行登录。

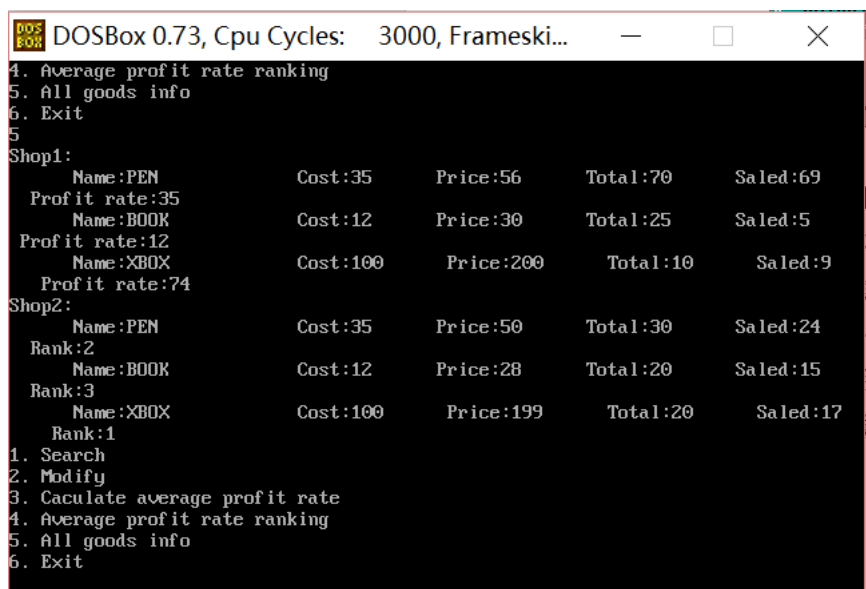


图 3.5.10 进入登录状态

在这里发现了奇怪的现象,就是这里的进货价看起来特别正常,通过计算发现如果按这个成本价计算,就会得到正确的利润。猜测对方可能在子程序中解密进货价时直接将数据段中的数据覆盖了,导致能看到正确的进货价。

# 汇编语言程序设计实验报告

5) 进入对方子程序

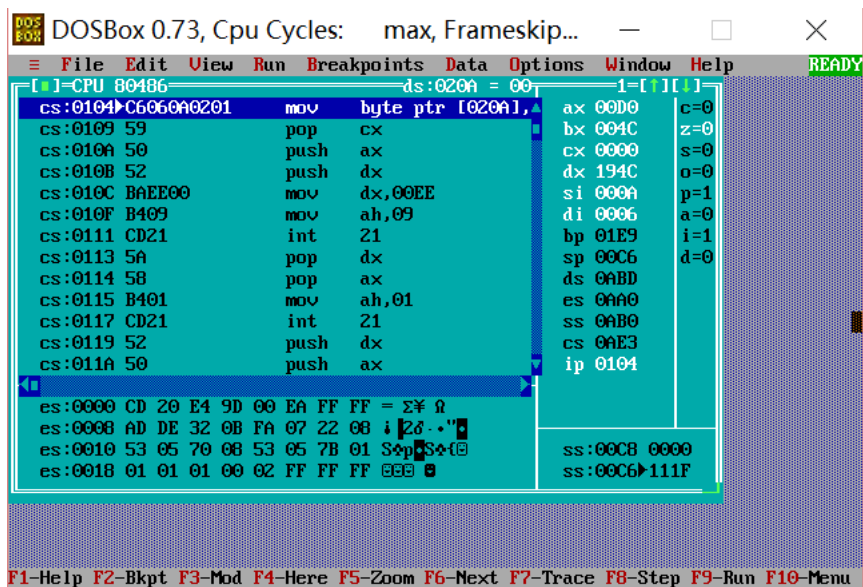


图 3.5.11 设置断点进入程序功能

选择 5 号功能，进入。

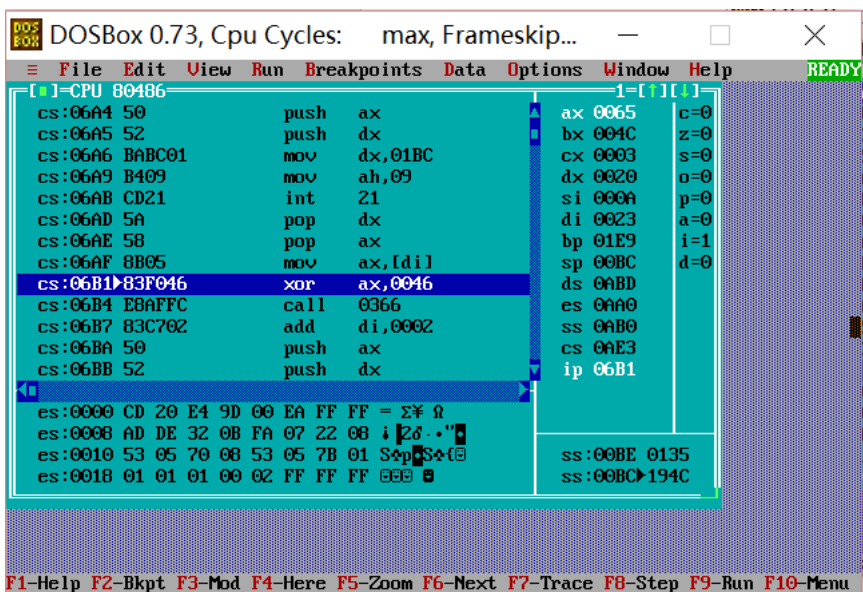


图 3.5.12 5 号功能具体反汇编语句

发现上图中 di 的值为 0023，发现是数据段中的进货价所在的位置。后面的 xor ax, 0046 疑似是加密语句，0046ascii 码对应着 F，猜测对方将进货价与 F 异或得到密文。

同时进入功能 3，发现其对进货价也做了这个运算，因此确定对方的加密方式。



# 汇编语言程序设计实验报告

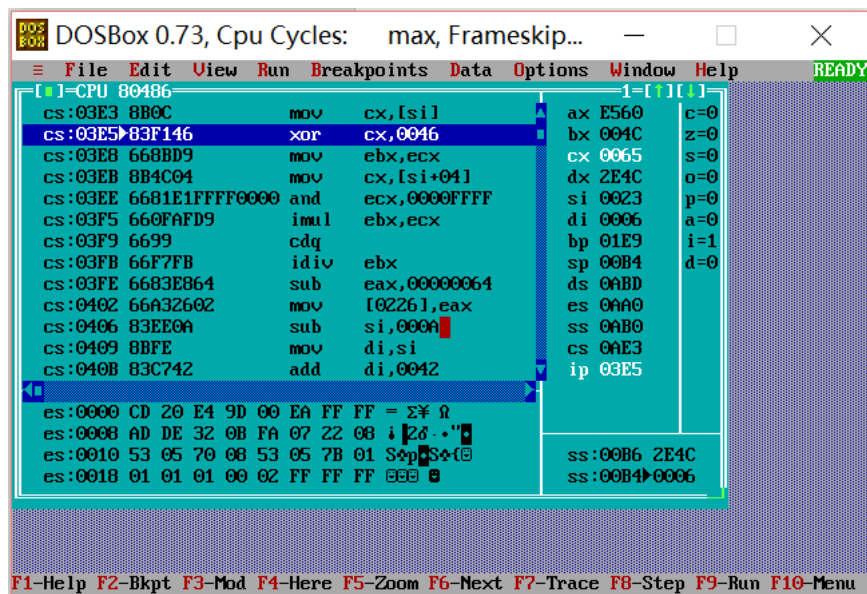


图 3.5.13 3 号功能具体反汇编语句

根据之前得到的加密过的进货价，将其与 F 异或，发现得到的进货价是正确的进货价。至此得到对方的全部信息。

### 3. 思考题

思考题在任务 4 中已经全部完成。

---

# 汇编语言程序设计实验报告

---

## 4 总结与体会

在本次实验任务 1 中，首先学会了用不同的方式获取中断类型码对应的中断处理程序的入口地址，可以用 `td` 直接观察中断矢量表中的信息，也可以用 `dos` 系统功能 `35h` 号调用，获取入口地址，还可以直接读取相应的内存单元。

在这次任务中，虽然实现过程十分简单。但在这次任务中值得注意的是用 `td` 直接观察中断矢量表中的信息不总是可靠的，比如在这次实验中观察 `1h` 的入口地址，直接观察的话是会出错的，这点可以具体在上文中的实验步骤中看到。这次任务主要就是深入理解中断矢量表的概念，这是中断中比较重要的部分。

在本次任务 2 中，学会了自己编写中断程序。这次要求自己编写的中断程序比较简单，就是将输入的小写字母变成大写字母并显示到屏幕上。在这次实验中学会了基本的中断怎么写，学会了如何将其中断程序接管系统原有的中断程序。也可以只改变原中断的一部分功能，这时就可以用 `call` 指令或者 `jmp` 指令直接跳到原中断程序继续执行。同时也知道了中断程序的影响范围是有限的，比如在 `td` 里就不能将输入的小写字母编程大写字母；还有另外打开一个 `dos` 窗口也不能将输入的小写字母变成大写字母。最后还学会了如何“卸载”自己的中断程序，其实就是将中断矢量表的入口地址改回原来的值，不过在这个过程中，利用变量的话可能会出错，当程序执行时，变量里面的只可能会发生变化，不是你想要的那个值，从而导致恢复错误。

在本次任务 3 中，主要了解了 `in` 和 `out` 指令的应用，也知道了 CMOS 中的时间信息是利用压缩 BCD 码存储的。本次任务主要是让我们更熟悉 I/O 访问，也教会了我怎么处理压缩 BCD 码。这些过程都在上文中的实验步骤里呈现了。

在本次任务 4 中，我学会了如何初步使自己的程序变得更安全。在这次任务中，首先增加了错误次数判断功能，当输入错误达到 3 次后，直接以未登录状态进入后续功能。同时，为了不让破解者直接查看数据段中的内容达到获取有用信息的目的，将信息进行加密后放进数据段中。同时在程序段中加入反跟踪程序，防止破解者进行单步调试破解程序。通过这次任务，我了解到了中断矢量表反跟踪，间接转移反跟踪，堆栈检查反跟踪这些方法的简单实现。其中中断矢量表容易被看出来，而使用堆栈检查则灵活得多。同时我也了解到反跟踪程序不一定要在有效信息附近出现，它当然也可以运用到一般的无效信息上。因为这可以大大增加破解者的破解难度，扰乱破解者的视线。将一般的思维逆转，采取不一样的方式来应对破解。这些加在无效信息上的反跟踪程序一定会大量浪费破解者的时间。同时如果对这些反跟踪程序很熟悉，就可以在有效程序段和无效程序段中掺杂反跟踪程序。使得破解者无法不理睬这些反跟踪程序，这防止了破解者直接跳过这些反跟踪手段，正常进入到程序的功能中去。让破解者不得不正视我们的反跟踪手段。除了这些方法，还可以结合使用大量无效代码来干扰视线，并且在代码段中尽量不要直接出现密钥以及有效信息。应当对这些信息进行一定的处理，具体操作在实验记录里已经有了。

在本次任务 5 中，我首先根据一些反跟踪的基本方法，来初步构想一个破解思路。首先利用静态反汇编，不单步调试程序，直接查看数据段。这种方法让我直接获取到了对方的用户名信息，加密后的密码信息，以及加密后的进货价信息。得到了这些数据后，我就开始想对方是如何加密的，于是我就利用 `TD` 调试对方的程序，进行动态反汇编破解程序。结合静态反汇编和动态反汇编的信息，可以有效破解程序。让我察觉到对方使用了反跟踪程序的原因是，通过反汇编，我在 `TD` 中看

# 汇编语言程序设计实验报告

---

到了 cli 语句, 这个指令的功能是关中断, 一般涉及到修改中断矢量表才会出现类似的语句。在正常的程序中很小几率会出现, 于是我就提高了警惕。果然这个指令后面就跟着遗传反跟踪的代码。对于常见的中断矢量表反跟踪, 直接设置断点, 或者设置新 CS:IP 即可跳过反跟踪程序。对于间接转移反跟踪, 则可以一步一步地试, 看看程序如果检查到被跟踪会跳到哪个地方去, 同时这段代码肯定会有跳到正常代码处的操作, 我们只需小心判断一下即可。如果使用了寄存器, 可以查看寄存器里面的值, 看看是不是异常。对于计时反跟踪, 直接设置新 CS:IP 跳过或者设置断点即可。而上面所有跳过方法都需要注意对方是不是把有效代码混合进里面了, 如果是的话, 就要挑出这些代码执行, 以免后面的程序出错。在本次程序中, 要对一些循环敏感一些, 因为这很有可能是判断用户名或者密码是否正确的程序, 像我就是根据一些循环确定了代码的功能, 从而找出了对方的加密方式。另一个问题就是如何用 C 语言实现反跟踪, 关于这个问题我在网上寻找了大量资料, 发现有大半的方法都是在 C 语言中内嵌汇编语言。另外一些程序都是调用了一些已有的函数, 进行反跟踪, 这些资料我目前看不明白, 这和汇编相差甚远。C 作为一门高级语言, 有很多地方提供了便利, 但在另外一些方面就不那么直接了。汇编作为底层语言, 可以直接影响系统。掌握汇编有利于我们深入理解系统知识。

在本次试验, 最大的收获就是更深层次地理解了中断矢量表的概念, 掌握了如何编写, 调试自己写的中断处理程序。中断矢量表的确是一个很重要的概念, 这是中断服务程序实现的基本, 同时我也还知道了一些比较冷门的知识, 比如 td 执行后直接观察到的中断矢量表里面的数据可能不是真实的数据。还有一个更重要的部分是学会了基本中断程序的编写, 调试方法。可以自己编写一些中断程序, 在程序遇到异常时可以根据自己的需要决定程序的走向。这可以让我们的代码更加安全。最后的任务主要是让我们了解 in, out 指令的用法, 以及 CMOS 里信息的存储方式。

在任务 4 和任务 5 中, 最大的收获就是学会了跟踪以及反跟踪的方法, 初步接触到了信息安全的领域, 在这个大数据时代, 信息安全变得尤为重要, 如果不多加小心, 就很容易被不法分子恶意盗取信息以此牟利。尽管这次实验中所运用到的跟踪与反跟踪技术十分稚嫩, 但对我们初步了解这些技术是很有帮助的。尽管我们还不能成为这些领域的专家, 但我们现在就已经有了安全的意识, 在以后的学习中, 也会将这些意识带到之后的学习中去。

# 汇 编 语 言 程 序 设 计 实 验 报 告

---

## 参考文献

- [1] 王元珍、韩宗芬、曹忠升.《80X86 汇编语言程序设计》. 华中科技大学出版社:2005 年 04 月
- [2] 《2018 汇编语言程序设计实验四题目》