

**课 程 实 验 报 告**

**课程名称： 汇编语言程序设计实验**

**实验名称： 实验七 数据加密与反跟踪**

**实验时间： 2018-4-28，14：00-17：30 实验地点： 南一楼804室74号实验台**

**指导教师： 左琼 专业班级：计算机科学与技术201603班**

**学 号： U201614577 姓 名： 龙际全**

**同组学生： 无 报告日期： 2018年 4 月 29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：

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](#_Toc513488937)

[2 实验内容 2](#_Toc513488938)

[3 实验过程 3](#_Toc513488939)

[3.1 任务1 3](#_Toc513488940)

[3.1.1 设计思想及存储单元分配 3](#_Toc513488941)

[3.1.2 源程序 3](#_Toc513488942)

[3.1.3 实验步骤 5](#_Toc513488943)

[3.1.4 实验记录与分析 5](#_Toc513488944)

[3.2 任务2 8](#_Toc513488945)

[3.2.1 设计思想及存储单元分配 8](#_Toc513488946)

[3.2.2 源程序 9](#_Toc513488947)

[3.2.3 实验步骤 11](#_Toc513488948)

[3.2.4 实验记录与分析 12](#_Toc513488949)

[3.3 任务3 14](#_Toc513488950)

[3.3.1 设计思想及存储单元分配 14](#_Toc513488951)

[3.3.2 源程序 14](#_Toc513488952)

[3.3.3 实验步骤 17](#_Toc513488953)

[3.3.4 实验记录与分析 17](#_Toc513488954)

[3.4 任务4 19](#_Toc513488955)

[3.4.1 源程序 19](#_Toc513488956)

[3.4.2 实验步骤 40](#_Toc513488957)

[3.4.3 实验记录 40](#_Toc513488958)

[3.5 任务5 42](#_Toc513488959)

[3.5.1 设计思想及存储单元分配 42](#_Toc513488960)

[3.5.2 实验步骤 42](#_Toc513488961)

[4 体会 46](#_Toc513488962)

[5 注 47](#_Toc513488963)

[参考文献 48](#_Toc513488964)

# 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) 掌握基本的数据加密方法与程序反跟踪方法，同时了解破解程序的一些途径。

# 实验内容

任务1：用三种方式获取中断类型码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） 先输入待读取的CMOS内部单元的地址编号（可以只处理编号小于10的地址单元）。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正确。

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

### 设计思想及存储单元分配

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。

1.直接在td中观察即可。

2.系统功能调用AH=35H，入口参数为AL=1H,取中断信息。

3.直接把主存中的DS:[4H]→IP,DS:[6H]→CS赋值给寄存器BX和CX。

中断类型码10H同理：

1.直接在td中观察即可。

2.系统功能调用AH=35H，入口参数为AL=10H,取中断信息。

3.直接把主存中的DS:[40H]→IP,DS:[42H]→CS赋值给寄存器BX和CX。

### 源程序

1.程序名task1-1H-1

.386

code SEGMENT USE16

ASSUME CS:code

start: xor ax,ax

mov DS,ax

mov ah,35h;取中断信息

mov al,01h

int 21H

mov ah,4CH

int 21H

code ends

end start

2.程序名task1-1H-2

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

CODE SEGMENT USE16

ASSUME CS:CODE, SS:STACK

START: xor ax,ax

mov DS,ax

xor bx,bx

xor cx,cx

mov bx,DS:[4H]

mov cx,DS:[6H]

mov AH,4CH

int 21H

CODE ENDS

END START

3.程序名task1-10H-1

.386

code SEGMENT USE16

ASSUME CS:code

start: xor ax,ax

mov DS,ax

mov ah,35h;取中断信息

mov al,10h

int 21H

mov ah,4CH

int 21H

code ends

end start

4.程序名task1-10H-2

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

CODE SEGMENT USE16

ASSUME CS:CODE, SS:STACK

START: xor ax,ax

mov DS,ax

xor bx,bx

xor cx,cx

mov bx,DS:[40H]

mov cx,DS:[42H]

mov AH,4CH

int 21H

CODE ENDS

END START

### 实验步骤

1.直接运行td，置数据区为当前区执行Goto，在地址输入窗口输:40H{（0:[1\*4])->IP,(0:[1\*4+2]->CS}读取连续的4个字节，前两个为偏移地址，后两个为段值。然后置代码区为当前区。输入刚才得到的值，得到对应的程序。

2.将程序task1-1H-1.asm编译、链接，调试直到没有错误。运行td task1-1H-1，在td中单步调试，观察es与bx的改变情况；

3.将程序task1-1H-2编译、链接，调试直到没有错误。运行td task1-1H-2，在td中单步调试，观察寄存器的值。

中断码为10H时同理。

### 实验记录与分析

中断类型码为1H时：

1.实验环境条件：AMD A8-8600P 1.6GHz，8G内存；WINDOWS 10下DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0； LINK.EXE 5.2; TD.EXE 5.0。

2.方法一，直接观察fs段，可以观察到从0:4h开始低2个字节为0800，高2个字节为7000,如下图3.1.1所示：

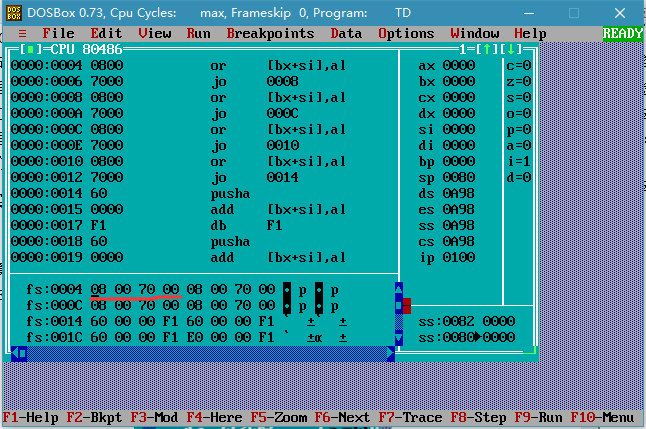


图3.1.1 td中直接观察中断矢量

3.方法二，调用35号系统功能，在TD内观察到es为07FC，bx为0B1A，如下图3.1.2所示；

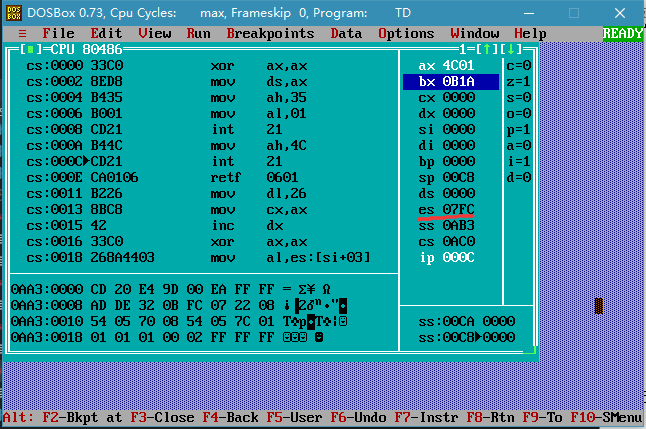


图3.1.2 35H系统功能调用查看中断矢量

4.方法三，将0:40h开始的低2个字节移至bx，高2个字节移至cx，得到cx为07FC，bx为0B1A，如下图3.1.3：

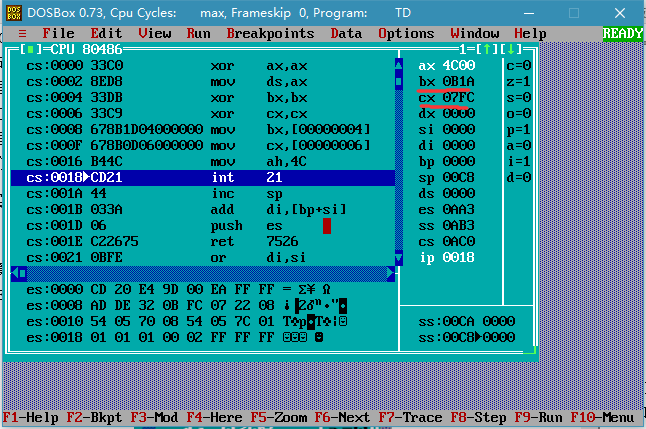


图3.1.3 将中断矢量赋值给BX以及CX

5.比较上述三种方法得到的结果，结果存在误差，查资料得，在1H中断工作在保护模式下，因此结果会有所不同。

同理，中断类型码为10H时：

1.实验环境条件：

2.方法一，直接观察fs段，可以观察到从0:40h开始低2个字节为0300，高2个字节为F100,如下图3.1.4所示：

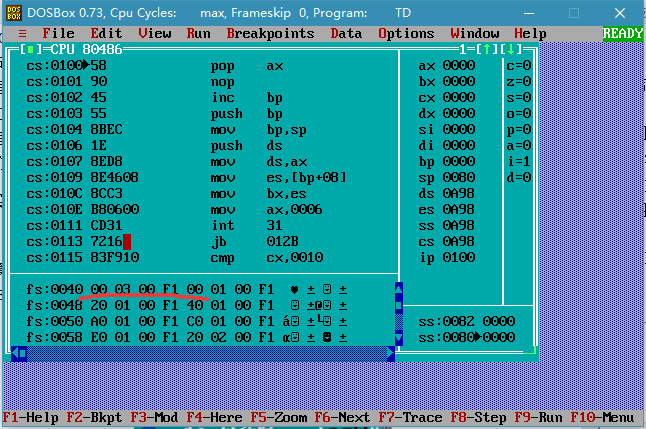


图3.1.4 td中10H中断矢量

3.方法二，调用35号系统功能，在TD内观察到es为F100，bx为0300，如下图3.1.5：

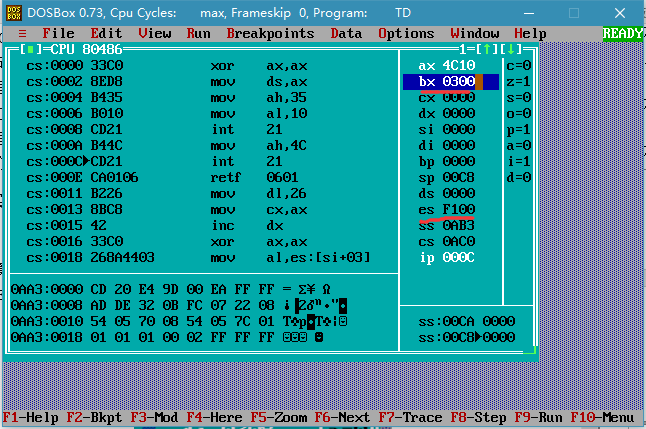


图3.1.5 利用35H号系统功能调用查看10H中断矢量

4.方法三，将0:40h开始的低2个字节移至bx，高2个字节移至cx，得到cx为F100，bx为0300，如下图3.1.6：

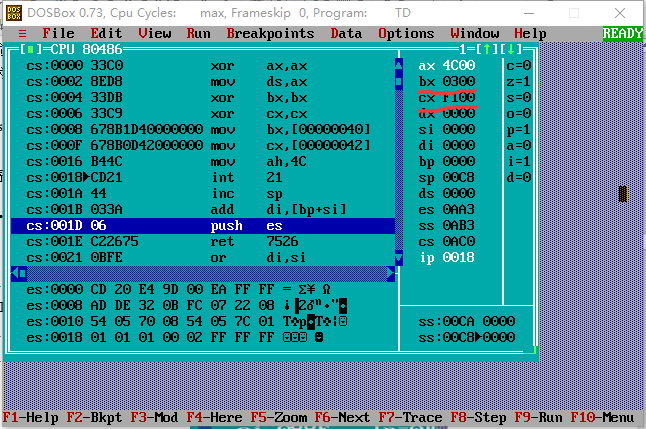


图3.1.6 将10H中断矢量赋值给bx和cx

5.比较上述三种方法得到的结果，三种方法得到的结果相同。

## 任务2

### 设计思想及存储单元分配

My\_16H设计思想：

修改键盘驱动中断程序(int 16h)，将中断矢量表0:4\*16h开始的4个字节修改为新的中断程序的偏移地址和段首址，并保存号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，先判断用户调用的是否为键盘驱动中断程序的读入键盘功能，若不是，则直接调用旧的键盘驱动中断程序;若是，则对输入进行处理。

实际上我们改变出口参数中AL的值即是键入字符的ASCII码即可实现。

卸载My\_16H的ori\_16设计思想：

若要卸载刚刚我们安装的中断处理程序，我们只需将系统中原有的中断矢量还原即可，而我们在新的my\_16H程序退出之后在系统主存中驻留了一些代码，包括新的中断处理程序以及保存原操作系统16H中断矢量信息的OLD\_INT，现在我们要还原中断矢量，首先要获取到OLD\_INT，一个为段值，一个为偏移量，而按照我们保存的方式OLD\_INT与我们新的16H中断处理程序在同一个段，偏移量为中断处理程序-4，新的16H中断处理程序的段值和偏移量又可以通过公式0:[16H\*4+2]和0:[16H\*4]获得，因此问题迎刃而解。

### 源程序

1，程序名task2-1-my\_16H用于重新接管16H号中断，使键盘输入的小写字母变为大写字母

.386

DATA SEGMENT USE16

DATA ENDS

CODE SEGMENT USE16

ASSUME CS:CODE,ss:STACK

OLD\_INT DW ?,?

;子程序名： my\_16H

;功能： 重新接管16H中断，

; 使键盘键入的小写字母变成大写字母，

; 大写字母不变.

;入口参数： ax，ah为中断选择类型，al为字符

;出口参数： 无

;知识点（来源出处）：教材P222-P223

;作者：boyjqlong@foxmail.com

my\_16H:cmp AH,00H

JE case\_change

cmp AH,10H

JE case\_change

JMP DWORD PTR OLD\_INT

case\_change:;大小写转换

PUSHF

CALL DWORD PTR OLD\_INT

;若想要完整的大小写转换，可以注释下一行

jmp case\_change\_s\_to\_l;直接跳转小写变大写

cmp al,'Z';大于Z为小写字母

jg case\_change\_s\_to\_l

case\_change\_l\_to\_s:;大写变小写

cmp al,'A'

jl case\_change\_QUIT

cmp al,’Z’

jg case\_change\_QUIT

add al,20h;大写变小写加20H

jmp case\_change\_QUIT

case\_change\_s\_to\_l:;小写变大写

cmp al,'a'

jl case\_change\_QUIT

cmp al,'z'

jg case\_change\_QUIT

sub al,20h;小写变大写减20H

jmp case\_change\_QUIT

case\_change\_QUIT:

IRET

START: XOR AX,AX

MOV DS,AX

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

MOV OLD\_INT,AX ;保存偏移部分

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

MOV OLD\_INT+2,AX ;保存段值

CLI

MOV WORD PTR DS:[16H\*4],OFFSET my\_16H

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 STACK

DB 200 DUP(0)

STACK ENDS

end START

（选做题）2，卸载安装的16H中断处理程序ori\_16，还原系统中断矢量表。

.386

DATA SEGMENT USE16

DATA ENDS

CODE SEGMENT USE16

ASSUME CS:CODE,ss:STACK

OLD\_INT DW ?,?;保存原有系统中断向量

START: XOR AX,AX

MOV DS,AX

mov si,word ptr ds:[16H\*4];偏移值

mov ax,word ptr ds:[16H\*4+2];段值

mov es,ax

mov ax,word ptr es:[si-4]

mov OLD\_INT,ax;si-4为对应原处理程序前驻留的内存

mov ax,word ptr es:[si-2]

mov OLD\_INT+2,ax;同理

CLI

;卸载中断程序

mov ax,OLD\_INT

mov word ptr ds:[16H\*4],ax;还原偏移值

mov ax,OLD\_INT+2

mov word ptr ds:[16H\*4+2],ax;还原段值

STI

MOV AH,4cH

INT 21H

CODE ENDS

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

end START

### 实验步骤

1. 使用编辑程序EDIT.EXE录入源程序task2-1-my\_16H.asm，编译，链接。

2. 观察提示信息，若出错，则用编辑程序修改错误，修改后再重新编译。

3. 执行该程序输入字符,观察输入的字母是否全部为大写字母。

4. 卸载同理，观察执行ori\_16.exe后，输入是否正常。

### 实验记录与分析

1. 实验环境条件：AMD A8-8600P 1.6GHz，8G内存；WINDOWS 10下DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0； LINK.EXE 5.2; TD.EXE 5.0。
2. My\_16H程序编译连接正常，如下图3.2.1所示：

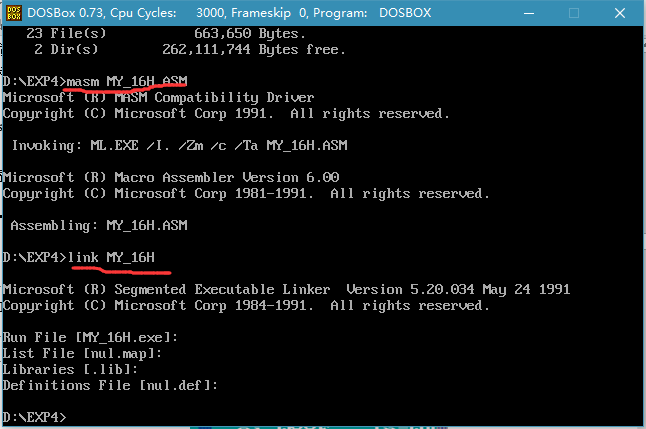


图3.2.1 编译链接正常

3. 运行my\_16H.exe，发现屏幕中输入的字母全变为了大写字母，如下图3.2.2所示：

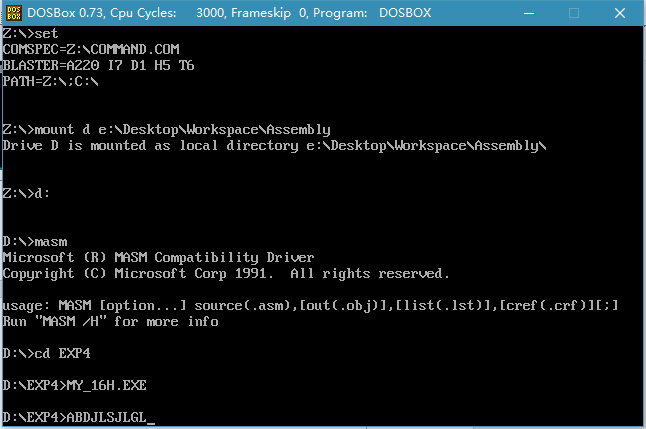


图3.2.2 输入字母全变为大写字母

4.通过编写代码发现，将旧的中断程序的偏移地址与首地址保存下来后，可有有两种方式调用旧的中断程序:

Jmp dword ptr old\_int;

Call dword ptr old\_int;

前一种方式，可直接结束新的(不返回至)中断程序;后一种方式，可返回至新的中断程序。

5.ori\_16程序编译连接正常，如下图3.2.3所示：

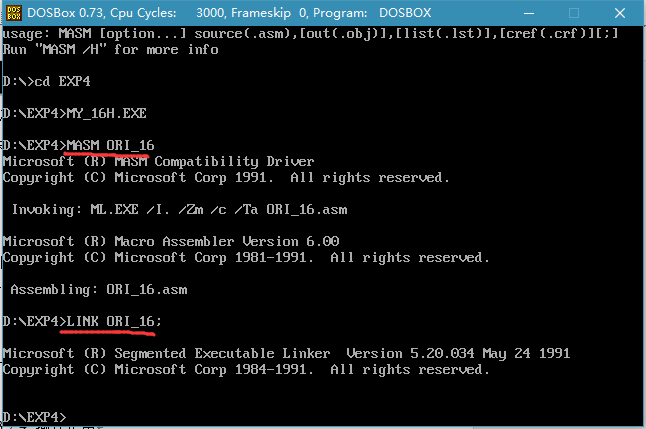


图3.2.3 ori\_16编译链接正常

6. 运行ori\_16.exe，发现输入已经正常，如下图3.2.4所示：

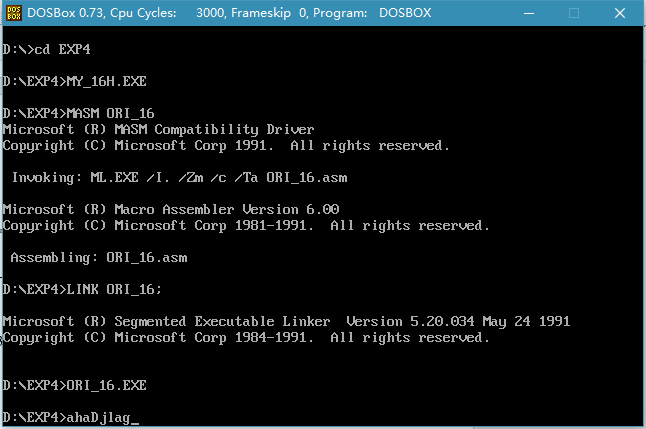


图3.2.4 输入正常

## 任务3

### 设计思想及存储单元分配

1号系统功能调用读入字符，将用户输入转化为数字，作为信息的偏移地址。然后调用in 指令，从指定端口取出数据即可。取出的数据根据BCD码的特性转换为十进制编码再调用radix数值转换子程序转换为16进制编码。

### 源程序

程序名task3

getchar macro

mov ah,1;dos1号功能调用

int 21h

endm

putchar macro char

mov dl,char

mov ah,2;dos2号功能调用

int 21h

endm

print macro string

lea dx,string

mov ah,9

int 21h

endm

.386

data segment use16

buf db 2

stringBuf db 16 dup(0);数字转换为字符串缓冲区

data ends

stack segment use16 stack

db 200 dup(0)

stack ends

code segment use16

assume ds:data,ss:stack,cs:code

start:

mov ax, data

mov ds, ax

;具体例子见于P226

;知识点：

; 1,cmos中70H端口为地址端口，可写不可读

; 2,71H端口为数据端口，该数据为70H所指地址的数据，可读可写

; 3,cmos中数据用bcd码存储，一个字节包含高四位和低四位的bcd码

; 4,bcd码与其他编码转换来自：

; https://blog.csdn.net/YuZhiHui\_No1/article/details/42806245

lopa:

getchar ;读入输入

sub al, 30h ;转化成十进制0~9

out 70h, al ;写入地址端口

in al, 71h ;读取信息

mov bh,al ;bh保存信息

shr al,4

and al,0fh ;al现为高位bcd码

mov cl,10

imul cl ;乘以高位权重10，结果在ax

mov ah,0 ;该操作可省略，由bcd码的特性决定

and bh,0fh ;bh为低位

add al,bh ;al现为完整的数据，即bcd码转换来的十进制数据

movsx eax,al ;将al扩展到eax

mov ebx,16 ;16进制

lea si,stringBuf

.if eax<0

neg eax

mov byte ptr [si],'-'

inc si

.endif

call radix

mov byte ptr [si],'H'

mov byte ptr [si+1],0ah

mov byte ptr [si+2],0dh

mov byte ptr [si+3],'$'

putchar 0ah ;输出换行

putchar 0dh ;输出回车

print stringBuf

jmp lopa ;循环

mov ah, 4ch

int 21h

;子程序名称：radix

;功能：将EAX中的无符号二进制数

;转换为P进制数

;参数：

;EAX--存放带转换的无符号二进制数

;EBX--存放要转换数制的基数

;SI--存放转换后的P进制ASCII码数字串的字节缓冲区首址

;返回：

;所求P进制ASCII码数字串按高位在前、地位在后的顺序存放在以SI为指针的字节缓冲区中

;SI--指向字节缓冲区中最后一个ASCII码的下一个字符处

;作者：来自华工80x86汇编语言程序设计课本136页

radix proc

push cx

push edx

xor cx,cx

radix\_lop1:

xor edx,edx

div ebx

push dx

inc cx

or eax,eax

jnz radix\_lop1

radix\_lop2:

pop ax

cmp al,10

jb radix\_l1

add al,7

radix\_l1:

add al,30H

mov [si],al

inc si

loop radix\_lop2

pop edx

pop CX

ret

radix endp

code ends

end start

### 实验步骤

1. 录入源程序task3，编译，链接。

2. 观察提示信息，若出错，则用编辑程序修改错误，修改后再重新编译。

3. 执行该程序，‘0’~‘9’，然后观察屏幕输出信息，验证是否正确。

### 实验记录与分析

1. 实验环境条件：AMD A8-8600P 1.6GHz，8G内存；WINDOWS 10下DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0； LINK.EXE 5.2; TD.EXE 5.0。
2. Task3程序编译链接正常，如下图3.3.1所示：

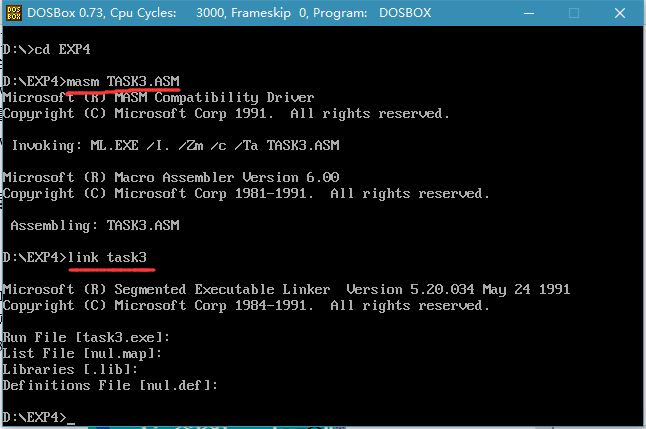
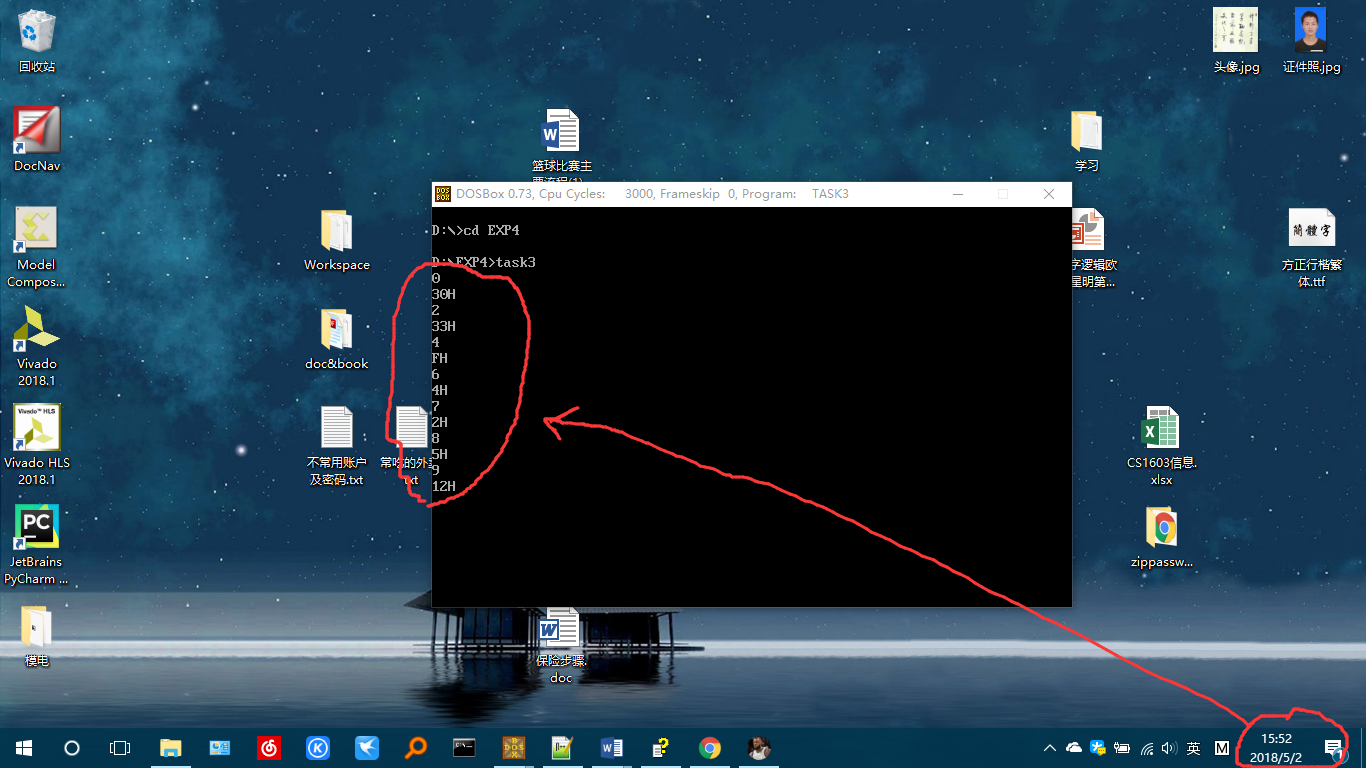


图3.3.1 task3编译链接正常

1. 根据bcd码的特性，取出的八位bcd码为两个十进制数，高四位为十进制数的十位，低四位为十进制数的个位，根据权重可还原该十进制数。
2. 依次输入0，2，4，6，7，8，9，显示硬件时间的秒，分，时以及年月日周，如下图3.3.2所示：



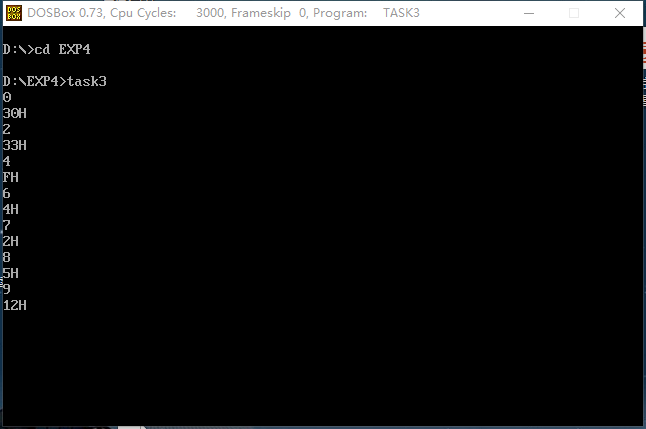




图3.3.2 显示秒，分，时，周，日，月，年

## 任务4

### 源程序

;使用寄存器dx,ax

;功能:将以oprd为首址的字符串输出

print macro oprd

push dx

push ax

lea dx,oprd

mov ah,9

int 21H

pop ax

pop dx

endm

;使用寄存器bx,dx,ax

;功能：输入一串字符串到oprd中

scan macro oprd

push bx

push dx

push ax

mov bx,offset oprd

add bx,2

call SetStringZero;输入前清零

lea dx,oprd

mov ah,10

int 21H

removeEnter oprd;清空回车

pop ax

pop dx

pop bx

endm

;使用寄存器ax

;从屏幕读取一个字符到al中

Getchar macro

mov ah,1

int 21H

endm

;使用寄存器dx，ax

;将字符char输出到屏幕

Putchar macro char

push dx

push ax

mov dl,char

mov ah,2

int 21H

pop ax

pop dx

endm

;使用寄存器dx,ax,si

;将数字num转化为字符串并输出

printNumString macro num

push dx

push ax

push si

mov dx,16

mov ax,word ptr num

lea si,numString

call f2t10

pop si

pop ax

pop dx

endm

;使用寄存器bx

;功能：输出以0结尾而不以$结尾的字符串

outString macro oprd1

push bx

mov bx,offset oprd1

call showStr

pop bx

endm

;oprd1为goodsOffset

;oprd2为ga1

;oprd3=偏移量，10=进货价，12=售货价，14=进货总数，16=已售数量,18=利润率

;oprd4=numString，存储数字字串的地址

;使用寄存器si

;功能：输出商品的某个字段，如售货价

printGoodsPartInfo macro oprd1,oprd2,oprd3

push si

push ax

mov si,oprd1

mov ax,oprd3

.if ax==10;进货价需要解密

mov ax,word ptr oprd2[si+oprd3]

xor ax,('N'+'O'-'P')

.else

mov ax,word ptr oprd2[si+oprd3]

.endif

printNumString ax

pop ax

pop si

endm

;功能：输出商品的全部信息

printGoodsAllInfo macro opd1,opd2

outString goodsBuyingPriceMsg

printGoodsPartInfo opd1,opd2,10;进货价

outString goodsSalesPriceMsg

printGoodsPartInfo opd1,opd2,12;销售价

outString goodsTotalStockMsg

printGoodsPartInfo opd1,opd2,14;进货总数

outString goodsSoldNumMsg

printGoodsPartInfo opd1,opd2,16;已售数量

endm

;功能：改变商品某个字段，如售货价

changeGoods macro msg,offsetNum,labelSelf,labelOther

push si

push bx

push cx

push dx

push ax

mov si,goodsOffset

mov bx,shopOffset

outString msg;用了寄存器bx

mov cx,offsetNum

add si,cx;售货价偏移量为12

mov ax,word ptr ga1[bx+si]

.if cx==10;进货价需要解密

xor ax,('N'+'O'-'P')

.endif

printNumString ax;输出售货价

Putchar '>'

scan buf

.if buf[1]==0;输入回车

jmp labelOther;直接跳过

.endif

lea si,buf

add si,2;si为数字串偏移地址

xor cx,cx;cx清零

mov cl,buf[1];数字串长度，包括符号位

mov dx,16;使用十六位

call f10t2;将buf+2字符串转化为数字

.if si==-1

print errorInputMsg;转换十进制数溢出或者输入不合法

jmp labelSelf

.endif

mov si,goodsOffset

mov bx,shopOffset

add si,offsetNum;售货价偏移量为12

mov word ptr ga1[bx+si],ax;ax为转换后的数

pop ax

pop dx

pop cx

pop bx

pop si

endm

;输出商店信息

printShop macro oprd1,oprd2

outString shopNameMsg

outString oprd1

outString goodsNameMsg

mov si,goodsOffset

lea bx,oprd2

add bx,si

call showStr

outString goodsSalesPriceMsg

printGoodsPartInfo goodsOffset,oprd2,12

outString goodsTotalStockMsg

printGoodsPartInfo goodsOffset,oprd2,14

outString goodsSoldNumMsg

printGoodsPartInfo goodsOffset,oprd2,16

endm

;删除字符串中的换行符

removeEnter macro string

mov si,word ptr string[1]

and si,00FFH

mov string[si+2],0

endm

.386

;--------------------------------------------------

DATA SEGMENT USE16

in\_name DB 11

DB ?

DB 11 DUP(0)

in\_pwd DB 7

DB ?

DB 7 DUP(0)

goodsName DB 11

DB ?

DB 11 DUP(0)

shopName db 11

db ?

db 11 dup(0)

PR1 DW 0

PR2 DW 0

APR DW 0

AUTH DB 0

authTimes db 0;认证次数，超过三次自动以未登录方式进入程序

COST1 DW 0;成本

PROFIT1 DW 0;利润

COST2 DW 0;成本

PROFIT2 DW 0;利润

BNAME DB 'LONG JQ',3 DUP(0);老板姓名

;BPASS DB 'NOP';密码

BPASS DB 3 xor 'C' ;密码串长度为3，采用与常数43H即'C'异或的方式编码成密文

db ('N'-29H)\*3 ;真实密码为NOP。采用密文存储

db ('O'-29H)\*3

db ('P'-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

old\_int1 dw 0,0 ;保存1号中断原矢量表

old\_int3 dw 0,0 ;保存3号中断原矢量表

P1 dw pass1 ;地址表(用于间接转移反跟踪)

E1 dw OVER

P2 dw pass2

shopNum equ 2

goodsNum equ 3

numString db 12 dup(0);字类型最大值为10位

stringNum dw 0

buf db 12

db ?

db 12 dup(0)

sign db ?;正负数标志单元

N EQU 30

goodsOffset dw 0

shopOffset dw 0

SHOP1 DB 'SHOP1',0;商店1，六个字节

ga1 DB 'PEN',7 DUP(0);商品名称，十个字节

DW 35 xor ('N'+'O'-'P'),56,100,58,?;进货价，售货价，进货数量，已售数量，利润率，十个字节

ga2 DB 'BOOK',6 DUP(0)

DW 12 xor ('N'+'O'-'P'),30,100,72,?

ga3 db 'BAG',7 dup(0)

dw 14 xor ('N'+'O'-'P'),32,100,89,?

SHOP2 DB 'SHOP2',0;商店2

gb1 DB 'PEN',7 DUP(0);商品名称

DW 35 xor ('N'+'O'-'P'),50,30000,0,?;进货价，售货价，进货数量，已售数量，利润率

gb2 DB 'BOOK',6 DUP(0)

DW 12 xor ('N'+'O'-'P'),28,30000,0,?

gb3 db 'BAG',7 dup(0)

dw 14 xor ('N'+'O'-'P'),32,100,89,?

scan\_NAME\_MSG DB 0AH,0DH,'please scan name(scan q/Q to OVER):$'

scan\_PASS\_MSG DB 0AH,0DH,'please scan password:$'

LOGIN\_FAILED\_MSG DB 0AH,0DH,'Login failed! Please check the name or the password!$'

END\_MSG DB 0AH,0DH,'end of program, press any key to continue...$'

INPUT\_GOODSNAME\_MSG DB 0AH,0DH,'please scan the name of the goods:$'

INPUT\_GOODSNAME\_MSG\_AGAIN DB 0AH,0DH,'goods name scan error,please scan again:$'

THREE\_TIMES\_MSG db 0ah,0dh,'third time auth failed, directly access by login failed!$'

GRADE\_A\_MSG DB 0AH,0DH,'A$'

GRADE\_B\_MSG DB 0AH,0DH,'B$'

GRADE\_C\_MSG DB 0AH,0DH,'C$'

GRADE\_D\_MSG DB 0AH,0DH,'D$'

GRADE\_F\_MSG DB 0AH,0DH,'F$'

authMenuMsg db 0ah,0dh,0ah,0dh,'1=query goods information 2=change goods information'

db 0ah,0dh,'3=calculate goods apr 4=calculate apr ranking'

db 0ah,0dh,'5=output all goods information 6=program exit',0ah,0dh,'$'

authFailMenuMsg db 0ah,0dh,0ah,0dh,'1=query goods information 6=program exit',0ah,0dh,'$'

menuReminderMsg db 0ah,0dh,'please input your choice:$'

shopNameMsg db 0AH,0DH,'shop name:',0

goodsNameMsg db 0AH,0DH,'goods name:',0

goodsBuyingPriceMsg db 0ah,0dh,'buying price:',0

goodsSalesPriceMsg db 0AH,0DH,'sales price:',0

goodsTotalStockMsg db 0AH,0DH,'total stock:',0;

goodsSoldNumMsg db 0AH,0DH,'the number of sold:',0;

goodsAprMsg db 0ah,0dh,'average profit rate:',0

goodsAprRankingMsg db 0ah,0dh,'average profit rate ranking:',0

errorInputMsg db 0ah,0dh,'error input!$'

changeInfoMsg db 0ah,0dh,'input the info you want change below(press enter to quit)',0ah,0dh,'$'

teammateModuleMsg db 0ah,0dh,'module of teammate,sorry!',0ah,0dh,'$'

DATA ENDS

;---------------------------------------------------

STACK SEGMENT STACK

DB 200 DUP(0)

STACK ENDS

;---------------------------------------------------

CODE SEGMENT USE16

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

START:

MOV AX,DATA

MOV DS,AX ;设置数据段

xor ax,ax ;中断矢量表段值为0

mov es,ax ;接管调试用中断，中断矢量表反跟踪

mov ax,es:[1\*4] ;保存原1号和3号中断矢量

mov old\_int1,ax

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

mov old\_int1+2,ax

mov ax,es:[3\*4] ;保存原1号和3号中断矢量

mov old\_int3,ax

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

mov old\_int3+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

FUNCION1:

MOV SI,0;计数器

MOV CX,10;计数器

lea bx,in\_name

add bx,2

call SetStringZero

lea bx,in\_pwd

add bx,2

call SetStringZero

scan\_NAME:

print scan\_NAME\_MSG

scan in\_name

MOV AL,0

CMP AL,in\_name[3]

JE AUTH\_FAIL

scan\_PWD: ;输入密码

print scan\_PASS\_MSG

scan in\_pwd

MOV SI,0;计数器

MOV CX,10;计数器

FUNCTION2\_NAME:

MOV AL,in\_name[SI+2]

CMP AL,BNAME[SI]

JNE PRINT\_LOGIN\_FAILED

INC SI

DEC CX

JNZ FUNCTION2\_NAME

FUNCTION2\_PWD:

cli ;计时反跟踪开始

mov ah,2ch

int 21h

push dx ;dx保存获取的秒和百分秒

mov cl,in\_pwd+1 ;比较输入的串长和密码长度是否一样

xor cl,'C'

sub cl,BPASS ;BPASS首地址为长度与'C'异或

movsx bx,cl

add bx,offset P1

mov ah,2ch ;获取第二次秒和百分秒

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:

cmp cl,0 ;串长是否相等

jne PRINT\_LOGIN\_FAILED

cmp bx,offset P1 ;是否在异常环境运行

je OK2

jmp E1

db 'how are you' ;冗余信息，扰乱视线

OK2:

mov bx,[bx]

jmp bx

db 'how to go'

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 'aha, it is me again'

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,BPASS+1[si]

jnz PRINT\_LOGIN\_FAILED

inc si

loop pass2

jmp pass3

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 ;正常修改了的话会转到testint

db 'do you guess the password?'

AUTH\_SUCCESS:

MOV BH,1

MOV BYTE PTR AUTH,BH

JMP FUNCTION3

AUTH\_FAIL:

MOV AL,'q'

CMP AL,in\_name[2]

JE OVER

MOV AL,'Q'

CMP AL,in\_name[2]

JE OVER

MOV AL,0

CMP AL,in\_name[2]

JNE scan\_PWD

MOV BH,0

MOV BYTE PTR AUTH,BH

JMP FUNCTION3

PRINT\_LOGIN\_FAILED:

print LOGIN\_FAILED\_MSG

add authTimes,1

mov al,authTimes

.if al==3

print THREE\_TIMES\_MSG

mov al,0

mov authTimes,al;认证次数清零

mov bh,0

mov byte ptr auth,bh

je FUNCTION3

.endif

JMP FUNCION1

function3:

jmp function3\_1\_menu

function3\_1\_menu:;显示菜单

mov al,AUTH

.if al==1

jmp authMenu

.else

jmp authFailMenu

.endif

authMenu:;已登陆

print authMenuMsg

print menuReminderMsg

Getchar

sub al,'0'

push ax

Getchar

pop ax

.if al==1

call function3\_2\_query

.elseif al==2

call function3\_3\_change

.elseif al==3

print teammateModuleMsg

.elseif al==4

print teammateModuleMsg

.elseif al==5

print teammateModuleMsg

.elseif al==6

call exit

.else

print errorInputMsg

.endif

jmp function3

authFailMenu:;未登录

print authFailMenuMsg

print menuReminderMsg

Getchar

sub al,'0'

push ax

Getchar

pop ax

.if al==1

call function3\_2\_query

.elseif al==6

call OVER

.else

print errorInputMsg

.endif

jmp function3

function3\_2\_query proc;查询商品

push dx

push ax

print INPUT\_GOODSNAME\_MSG

scan goodsName

call isGoods

mov si,goodsOffset

mov bx,0

.if si==1||bx==1

;print END\_MSG

jmp query\_ret

.elseif bx==shopNum\*(goodsNum\*20+6)

print errorInputMsg

jmp query\_ret

.elseif si==goodsNum\*20

print errorInputMsg

jmp query\_ret

.endif

printShop SHOP1,ga1

printShop SHOP2,gb1

query\_ret:

pop ax

pop dx

ret

function3\_2\_query endp

function3\_3\_change proc;修改商品信息

push bx

push ax

push dx

push si

push cx

outString shopNameMsg

scan shopName

outString goodsNameMsg

scan goodsName

call isInShop

call isGoods

mov si,goodsOffset

mov bx,shopOffset

.if si==1||bx==1

jmp change\_ret

.elseif si==goodsNum\*20||bx==shopNum\*(goodsNum\*20+6)

print errorInputMsg

jmp change\_ret

.endif

change\_buying\_price:

changeGoods goodsBuyingPriceMsg,10,change\_buying\_price,change\_sales\_price

change\_sales\_price:

changeGoods goodsSalesPriceMsg,12,change\_sales\_price,change\_total\_stock

change\_total\_stock:

changeGoods goodsTotalStockMsg,14,change\_total\_stock,change\_ret

change\_ret:

pop cx

pop si

pop dx

pop ax

pop bx

ret

function3\_3\_change endp

newint:;用于反跟踪

iret

testint:

jmp AUTH\_SUCCESS

OVER:

cli ;还原中断矢量

xor ax,ax

mov es,ax

mov ax,old\_int1

mov es:[1\*4],ax

mov ax,old\_int1+2

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

mov ax,old\_int3

mov es:[3\*4],ax

mov ax,old\_int3+2

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

sti

call exit

;子程序名称：exit

;功能：输出退出提示信息，将控制权交由操作系统

;参数：END\_MSG指向提示字符串

;返回：无

;作者：boyjqlong@foxmail.com

exit proc

print END\_MSG

Getchar

MOV AH,4CH

INT 21H

ret

exit endp

;子程序名称：showStr

;功能：输出以0结尾而不以$结尾的字符串

;参数：ds:bx指向待显示字符串首地址

;返回：无

;作者：boyjqlong@foxmail.com

showStr proc

push dx

push si

push ax

mov si,0

showChar:

mov dl,[bx+si]

cmp dl,0

je showStr\_return

;该功能可用宏Putchar代替

mov ah,2

int 21H

inc si

jmp showChar

showStr\_return:

pop ax

pop si

pop dx

ret

showStr endp

;子程序名称：SetStringZero

;功能：将以ds:bx为指针的字符串内容清零

;参数：ds:bx指向字符串首地址，

;返回：无

;注意事项：字符串以0结尾

;作者：boyjqlong@foxmail.com

SetStringZero proc

push dx

push ax

push si

mov si,0

SetStringZero\_core:

mov dl,byte ptr [bx+si]

cmp dl,0

je SetStringZero\_ret;等于0时跳出循环

mov byte ptr [bx+si],0

inc si

jmp SetStringZero\_core

SetStringZero\_ret:

pop si

pop ax

pop dx

ret

SetStringZero endp

;子程序名称：isInShop

;功能：判断某商品是否是商店里的商品

;参数：shopName为商店名

;返回：shopOffset

; shopName为空，shopOffset=1

; 商店名不匹配，shopOffset=shopNum\*(goodsNum\*20+6)

; 每个商品占20字节，每个商店占6个字节

; 成功匹配，shopOffset对应商店的偏移量

;作者：boyjqlong@foxmail.com

isInShop proc

push cx

push si

push bx

push ax

mov shopOffset,0

mov al,shopName[1]

.if al==0

mov shopOffset,1

jmp isInShop\_ret

.endif

mov shopOffset,0

isInShopcycle:

mov cx,6

mov si,0

mov bx,shopOffset

isInShopcmp:

mov al,SHOP1[bx+si]

cmp al,shopName[si+2]

jne next\_shop

inc si

dec cx

jnz isInShopcmp

jmp isInShop\_ret

next\_shop:

add shopOffset,goodsNum\*20+6

cmp shopOffset,shopNum\*(goodsNum\*20+6)

jne isInShopcycle

jmp isInShop\_ret

isInShop\_ret:

pop ax

pop bx

pop si

pop cx

ret

isInShop endp

;子程序名称：isGoods

;功能：判断某商品是否是商店里的商品

;参数：goodsName为商品名

;返回：goodsOffset

; goodsName为空，goodsOffset=1

; 商品名不匹配，goodsOffset=goodsNum\*20

; 每个商品占20字节，每个商店占6个字节

; 成功匹配，goodsOffset对应商品的偏移量

;作者：boyjqlong@foxmail.com

isGoods proc

push cx

push si

push bx

push ax

mov goodsOffset,0

mov al,goodsName[1]

.if al==0

mov goodsOffset,1

jmp isGoods\_ret

.endif

isGoodscycle:

mov cx,10

mov si,0

mov bx,goodsOffset

isGoodscmp:

mov al,ga1[bx+si]

cmp al,goodsName[si+2]

jne next\_goods

inc si

dec cx

jnz isGoodscmp

jmp isGoods\_ret

next\_goods:

add goodsOffset,20

cmp goodsOffset,goodsNum\*20

jne isGoodscycle

jmp isGoods\_ret

isGoods\_ret:

pop ax

pop bx

pop si

pop cx

ret

isGoods endp

;子程序名称：f10t2

;功能：将以si位置真的字节存储区中的有符号十进制数字串转换成二进制数送入ax/eax之中

;参数：

;si--指向待转换的有符号十进制数字串存储区首址

;dx--转换为16位或者32位二进制数标志

;dx=16对应转换为16位送入ax，dx=32对应转换为32送入eax

;cx--存放待转换十进制数字串的长度

;返回：

;si,si=-1代表溢出或者数字串有非法字符

;eax/ax存放转换后的二进制数

;作者：来自华工80x86汇编语言程序设计课本146页

f10t2 proc

push ebx

mov eax,0

mov sign,0

mov bl,[si]

cmp bl,'+'

je f10t2\_f10

cmp bl,'-'

jne f10t2\_next2

mov sign,1

f10t2\_f10:

dec cx

jz f10t2\_err

f10t2\_next1:

inc si

mov bl,[si]

f10t2\_next2:

cmp bl,'0'

jb f10t2\_err

cmp bl,'9'

ja f10t2\_err

sub bl,30H

movzx ebx,bl

imul eax,10

jo f10t2\_err

add eax,ebx

jo f10t2\_err

js f10t2\_err

jc f10t2\_err

dec cx

jnz f10t2\_next1

cmp dx,16

jne f10t2\_pp0

cmp eax,7fffh

ja f10t2\_err

f10t2\_pp0:

cmp sign,1

jne f10t2\_qq

neg eax

f10t2\_qq:

pop ebx

ret

f10t2\_err:

mov si,-1

jmp f10t2\_qq

f10t2 endp

;子程序名称：f2t10

;功能：将ax/eax中的有符号二进制数以十进制形式在显示器上输出

;参数：

;ax/eax--存放待转换的有符号二进制数

;dx--存放32位有符号二进制数的标志，

;dx=16为16位对应ax，dx=32为32位对应eax

;返回：

;转换后的带符号十进制数在显示器上输出

;调用子程序：radix

;作者：来自华工80x86汇编语言程序设计课本141页

f2t10 proc far

push ebx

push si

lea si,numString

cmp dx,32

jne f2t10\_b

movsx eax,ax

f2t10\_b:

or eax,eax

jns f2t10\_plus

neg eax

mov byte ptr [si],'-'

inc si

f2t10\_plus:

mov ebx,10

call radix

mov byte ptr [si],'$'

lea dx,numString

mov ah,9

int 21H

pop si

pop ebx

ret

f2t10 endp

;子程序名称：radix

;功能：将EAX中的无符号二进制数

;转换为P进制数

;参数：

;EAX--存放带转换的无符号二进制数

;EBX--存放要转换数制的基数

;SI--存放转换后的P进制ASCII码数字串的字节缓冲区首址

;返回：

;所求P进制ASCII码数字串按高位在前、地位在后的顺序存放在以SI为指针的字节缓冲区中

;SI--指向字节缓冲区中最后一个ASCII码的下一个字符处

;作者：来自华工80x86汇编语言程序设计课本136页

radix proc

push cx

push edx

xor cx,cx

radix\_lop1:

xor edx,edx

div ebx

push dx

inc cx

or eax,eax

jnz radix\_lop1

radix\_lop2:

pop ax

cmp al,10

jb radix\_l1

add al,7

radix\_l1:

add al,30H

mov [si],al

inc si

loop radix\_lop2

pop edx

pop CX

ret

radix endp

CODE ENDS

END START

### 实验步骤

1. 编写程序task4.asm;
2. 编译链接task4.asm，若有错误则修改代码task4.asm直至没有错误为止
3. 运行task4.exe。

### 实验记录

1. 实验环境条件：AMD A8-8600P 1.6GHz，8G内存；WINDOWS 10下DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0； LINK.EXE 5.2; TD.EXE 5.0。
2. 编译链接正常，如下图3.4.1：

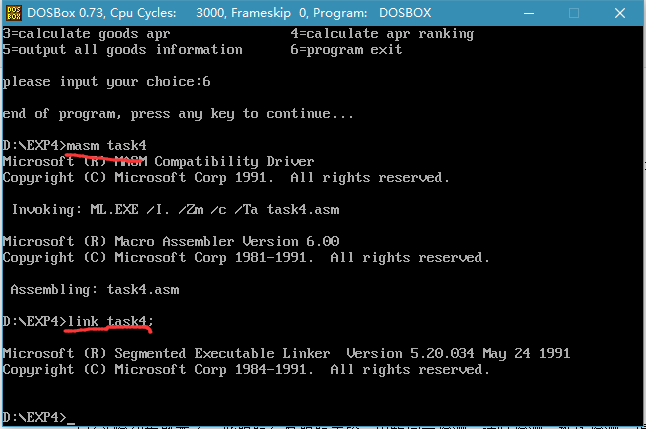


图3.4.1 编译链接正常

1. 输入错误密码三次，直接以游客方式进入，防止被以枚举方式暴力破解，如下图3.4.2：

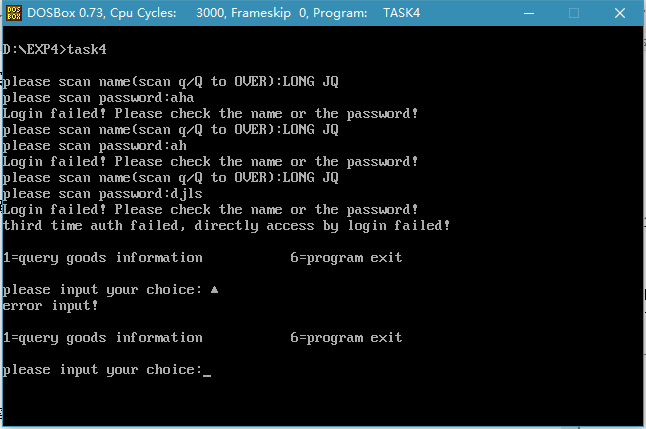


图3.4.2 认证次数超过三次，直接以游客方式进入

1. 正常功能已在上次实验中验证，在此不再赘述。
2. 当试图在td里单步执行程序时，由于td接管1号和3号中断，因此会产生冲突导致td崩溃退出程序，截图如下3.4.3：

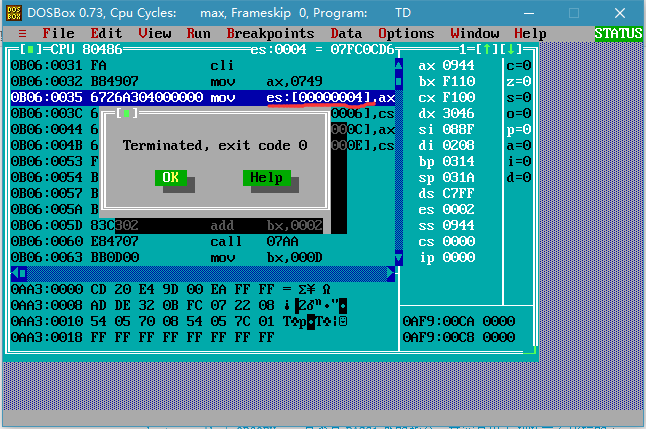


图3.4.3调试时程序异常退出

1. 修改ip使程序跳过接管中断，在td里单步调试，输入密码后，当我们试图通过单步执行查看密码所在数据段时，由于我们程序设有计时反跟踪，认为程序运行在异常环境，因此直接退出如图3.4.4（我的用户名是LONG JQ，密码是NOP）：

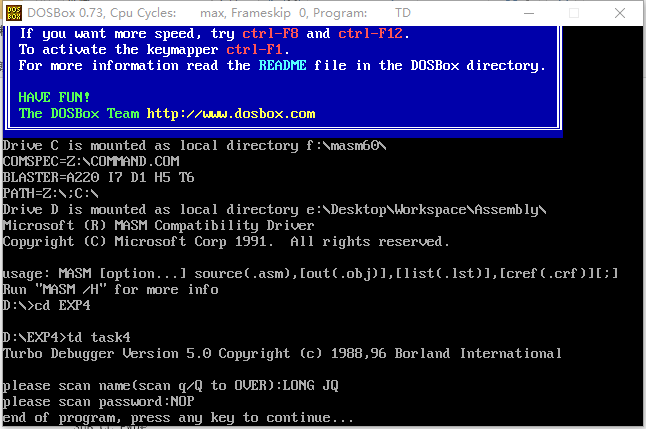


图3.4.4 调试时程序异常退出

7、当破解人员利用反汇编工具运行程序时，栈顶数据的值被修改，可通过此检测程序是否是在反汇编环境下执行；若程序在异常环境下运行，则立即终止程序。因没有有效的反汇编工具，所以这个功能体现不出来。

## 任务5

### 设计思想及存储单元分配

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

### 实验步骤

1. 运行TD，破解敌人的程序enemy.exe；

2. 单步执行时发现td异常退出，退出ip为0032，观察发现为修改了中断矢量表与td产生冲突，再次调试到0031时，直接修改ip为0054，调试正常；

3. 继续调试，直至程序提示我们输入用户名和密码，该过程无异常；

4. 继续调试，发现程序中出现地址，怀疑为用户名或者密码，首先跳到ds:0002，发现内存中对应我们刚刚输入的用户名ahah，如图3.5.1：

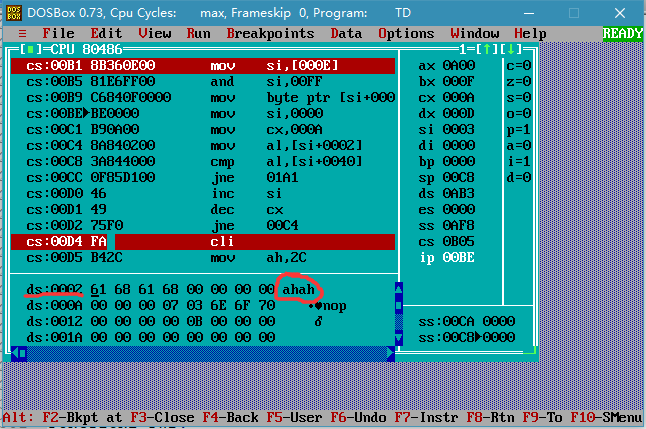


图3.5.1 地址0002数据段内容

1. 继续调试，有理由相信ds:0040为老板用户名，得到老板用户名zhang，检查上下的代码发现为直接比较，因此zhang应该就为老板的用户名，重新启动调试，发现输入用户名zhang时程序没有像中间jne 01a1跳转，因此用户名破解成功，而且有理由相信cs:01a1可能为认证失败的程序入口：

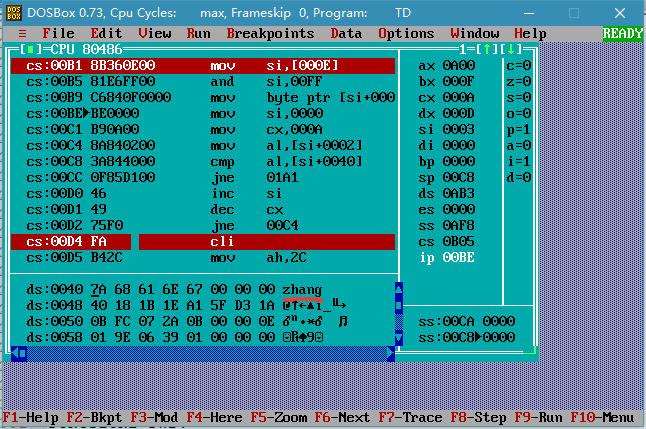


图3.5.2 地址0040数据段内容

1. 继续调试，发现程序异常退出，分析代码，发现前后有ah=2c的系统调用中断，查询资料知用于获取系统时间，并保存到dx中，疑似反跟踪代码，因此修改ip直接跳过，遇到类似的也同样处理，修改ip后程序正常运行，直到运行到类似用户名比较前，高度警惕，如下图3.5.3：

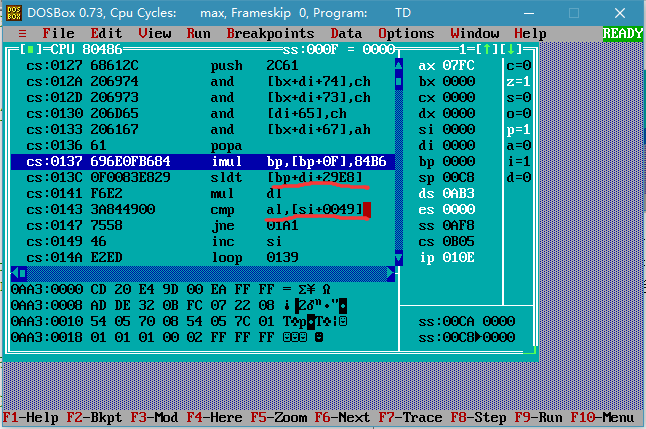


图3.5.3 出现两个地址

1. 令人惊喜的是发现神奇的01a1，之前比对用户名时已经怀疑该地址可能为认真失败的程序入口，跳到改地址发现并不是字符型数据，猜想到为密文，如图3.5.4：

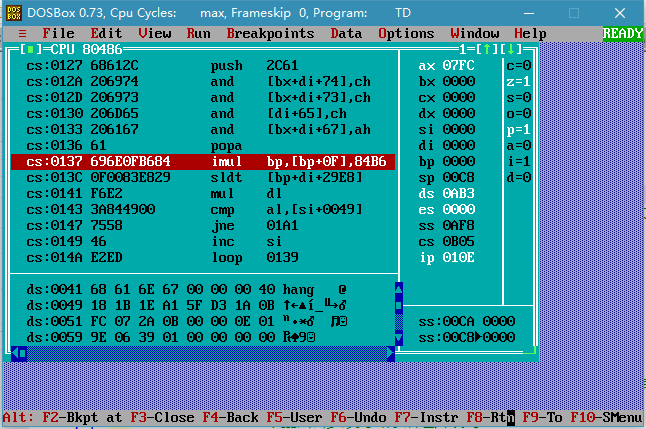


图3.5.4 地址0049内容，非字符型数据

1. 综合比较前后，知道密码长度和加密算法，解密后的密码为123，what！费尽千辛万苦得到的密码竟然这么简单。
2. 用户名输入zhang，密码输入123，wonderful！成功进入系统，进货价迎刃而解。如下图3.5.5：

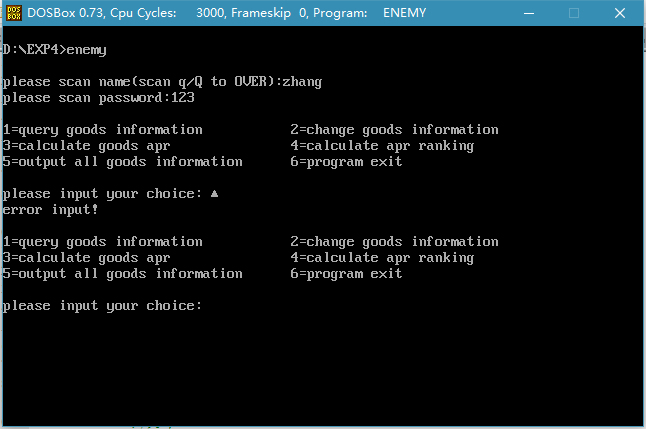


图3.5.5 用户名和密码正确

# 体会

这次实验让我获益良多，具体为：

1. 了解了系统底层，比如中断矢量表，键盘扫描码以及CMOS。
2. 认识到我们的电脑是非常脆弱的，中断矢量表是非常脆弱的，想一想哪一天你电脑中了病毒，病毒直击我们电脑的底层，后果简直不堪设想。
3. 在这次实验之前，我虽然不愿意承认但也不得不承认，我心底有一个疑问，我们为什么要学汇编语言，这次实验之前，我这样跟自己说，为了更好地理解高级语言的工作机理，为了更深入地理解C语言的指针，为了拓宽自己的知识面，为了锻炼自己吃苦耐劳的能力，为了成为一个更加称职的科班生，我有这样的疑问我并不觉得可耻，这只能代表我没有深入地了解这门学科。这次实验之后，我羞愧于自己此前的无知，要我现在来回答这个问题我会这样回答，当然前面几个回答同样成立，为了更加了解我正在使用的计算机，为了更加地有效率，为了深入计算机本质。
4. 破解对方的密码真的是有意思啊。

# 注

有个别实验代码较多较长，不适于直接在word中直接操作。因此本人将本次实验所有代码及二进制可执行文件上传至GitHub，具体网址为<https://github.com/DragonDriver/AssemblyExp/tree/master/Exp4>。

# 参考文献

[1] 王元珍 曹忠升 韩宗芬.《80x86汇编语言程序设计》.华中科技大学出版社