

**课 程 实 验 报 告**

**课程名称： 汇编语言程序设计实验**

**实验名称： 实验六 输入输出与中断处理程序设计**

**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 1](#_Toc512887060)

[2 实验内容 1](#_Toc512887061)

[3 实验过程 2](#_Toc512887062)

[3.1 任务1 2](#_Toc512887063)

[3.1.1 设计思想及存储单元分配 2](#_Toc512887064)

[3.1.2 源程序 2](#_Toc512887065)

[3.1.3 实验步骤 4](#_Toc512887066)

[3.1.4 实验记录与分析 4](#_Toc512887067)

[3.2 任务2 7](#_Toc512887068)

[3.2.1 设计思想及存储单元分配 7](#_Toc512887069)

[3.2.2 源程序 7](#_Toc512887070)

[3.2.3 实验步骤 10](#_Toc512887071)

[3.2.4 实验记录与分析 10](#_Toc512887072)

[3.3 任务3 12](#_Toc512887073)

[3.3.1 设计思想及存储单元分配 12](#_Toc512887074)

[3.3.2 源程序 13](#_Toc512887075)

[3.3.3 实验步骤 14](#_Toc512887076)

[3.3.4 实验记录与分析 14](#_Toc512887077)

[4 体会 16](#_Toc512887078)

[5 注 16](#_Toc512887079)

[参考文献 17](#_Toc512887080)

# 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。

# 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

# 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：

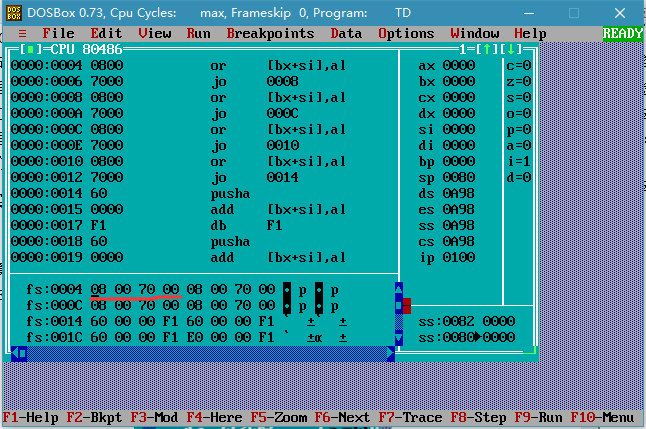


图3.1.1 td中直接观察中断矢量

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；

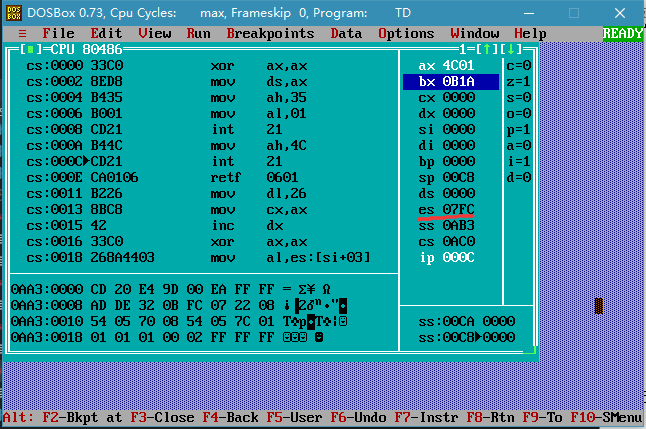


图3.1.2 35H系统功能调用查看中断矢量

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：

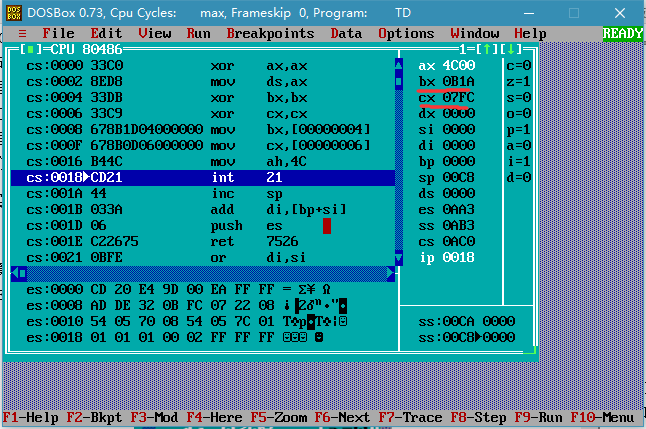


图3.1.3 将中断矢量赋值给BX以及CX

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：

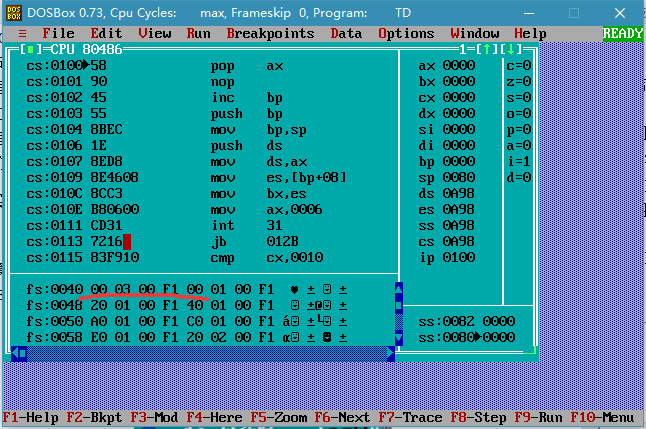


图3.1.4 td中10H中断矢量

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：

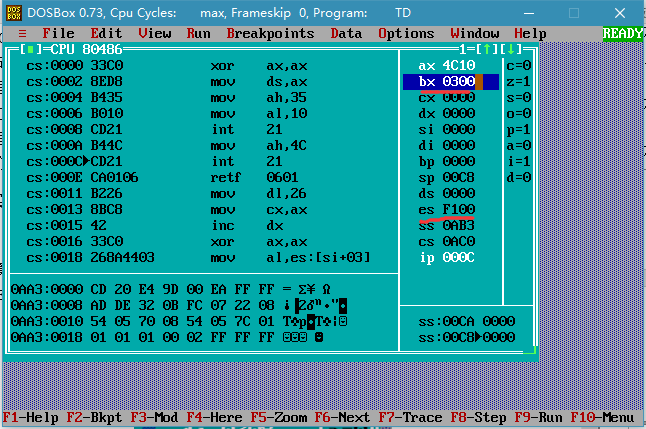


图3.1.5 利用35H号系统功能调用查看10H中断矢量

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：

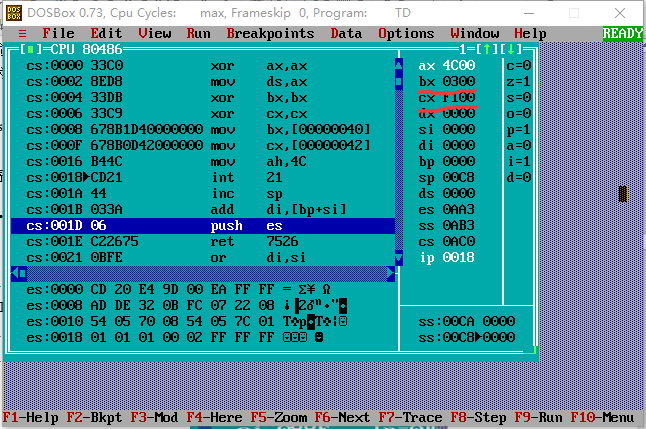


图3.1.6 将10H中断矢量赋值给bx和cx

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：

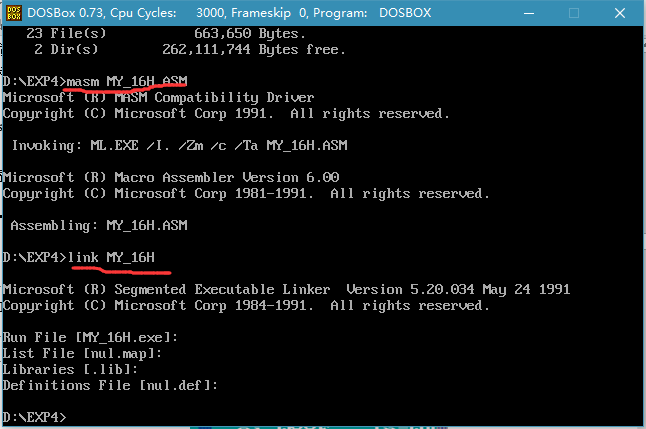


图3.2.1 编译链接正常

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：

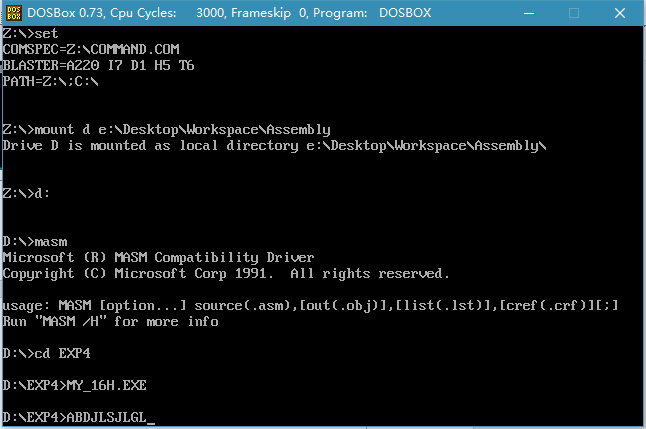


图3.2.2 输入字母全变为大写字母

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：

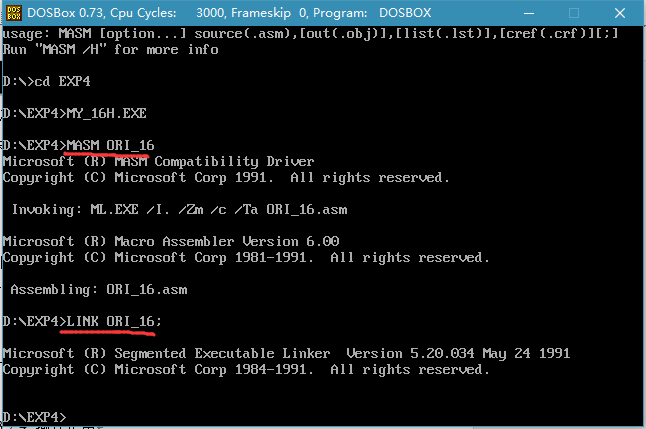


图3.2.3 ori\_16编译链接正常

6. 运行ori\_16.exe，发现输入已经正常，如下图3.2.4所示：

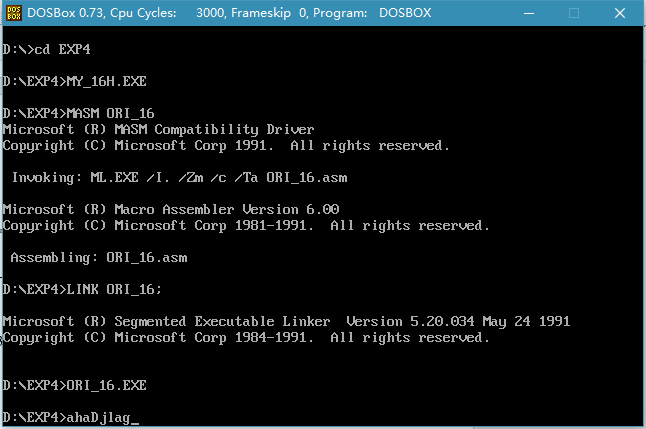


图3.2.4 输入正常

## 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：

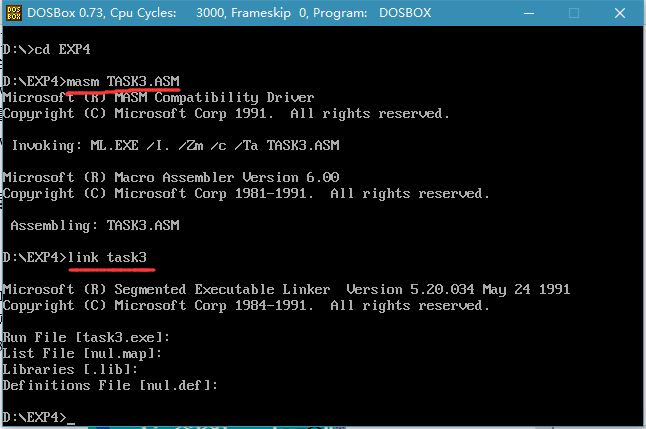
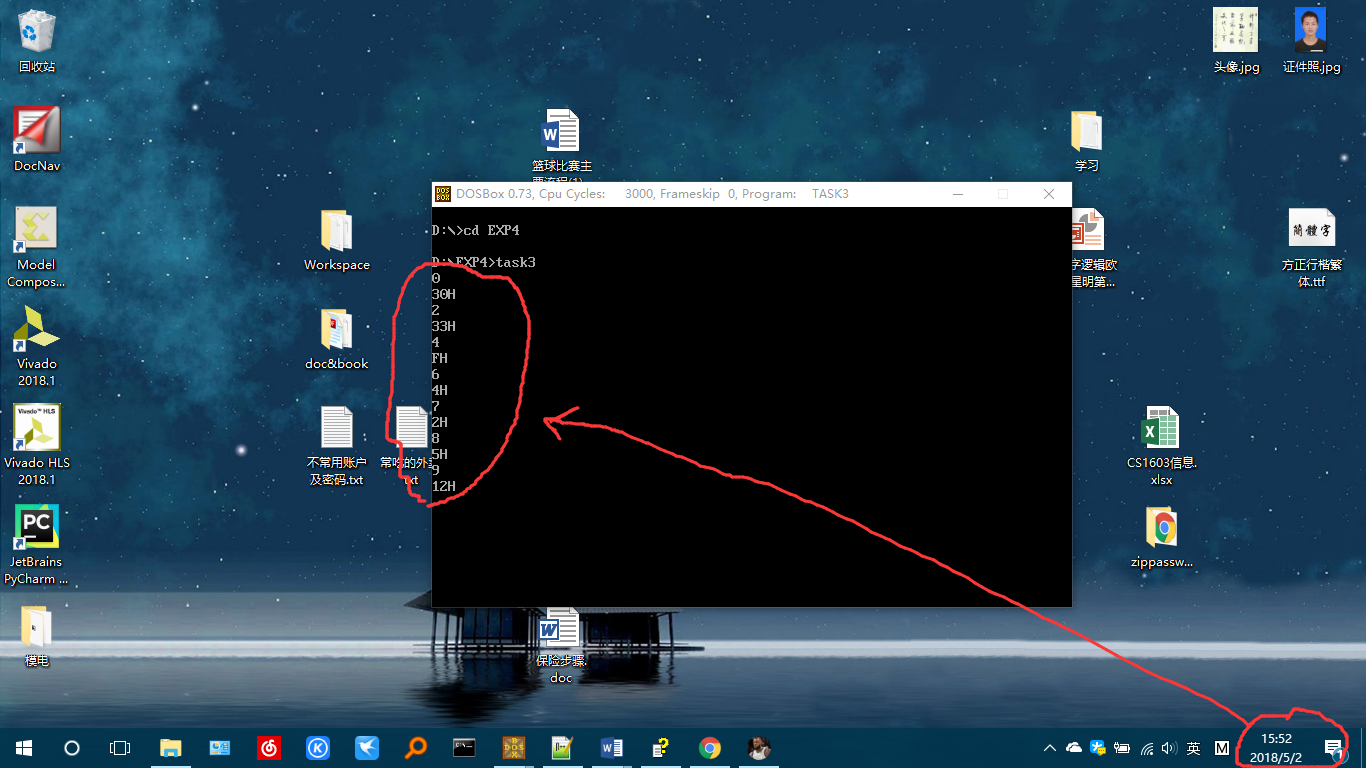


图3.3.1 task3编译链接正常

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：



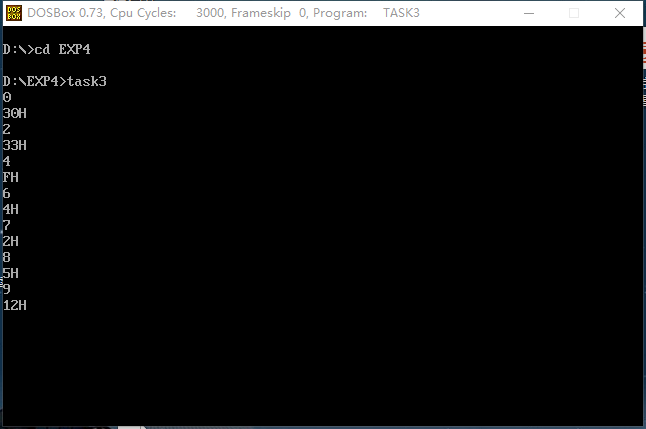




图3.3.2 显示秒，分，时，周，日，月，年

# 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。

# 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