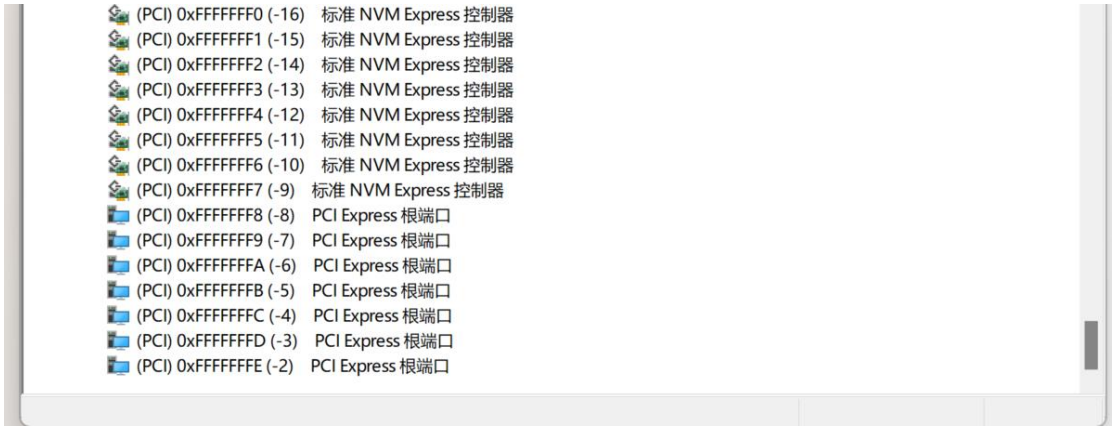
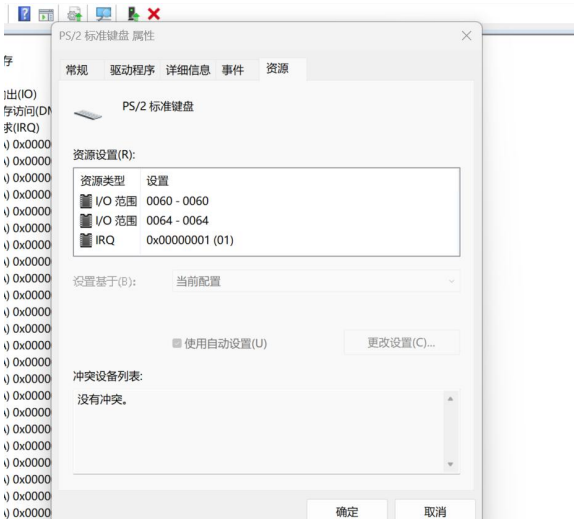
To Do 1: To identify the interrupts in use on your computer, open a terminal window and launch control panel as follows,

Figure3: Obtaining Control Panel on a laboratory PC.



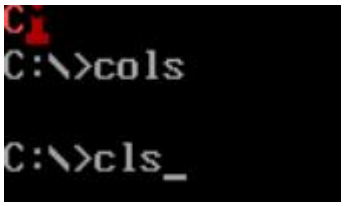


Part 2: Interrupts on DOS, we need to write directly to screen memory

Figure 7: Coding of attribute byte (e.g. bright red on a black background would be 4+8=12)

You will need to Clear Screen before you run the code as the text will scroll off the top the screen. Just type “cls <ret>”.

```
.MODEL small
.STACK
.DATA
.CODE
.STARTUP
mov ax,0b800h ; Base address of screen memory
mov es,ax ; Use extra segment register to access 0x800...
mov bx,0 ; Screen position x=0, y=0, (2*((80*y)+x))=0
mov al,'C' ; Print 'Hello' on the screen
mov es:[bx],al
inc bx
mov al,12 ; Bright red on a black background
mov es:[bx],al
inc bx ; Black on a red background
mov al,'M'
mov es:[bx],al
inc bx
mov al,64
mov es:[bx],al
quit: .EXIT
END
```



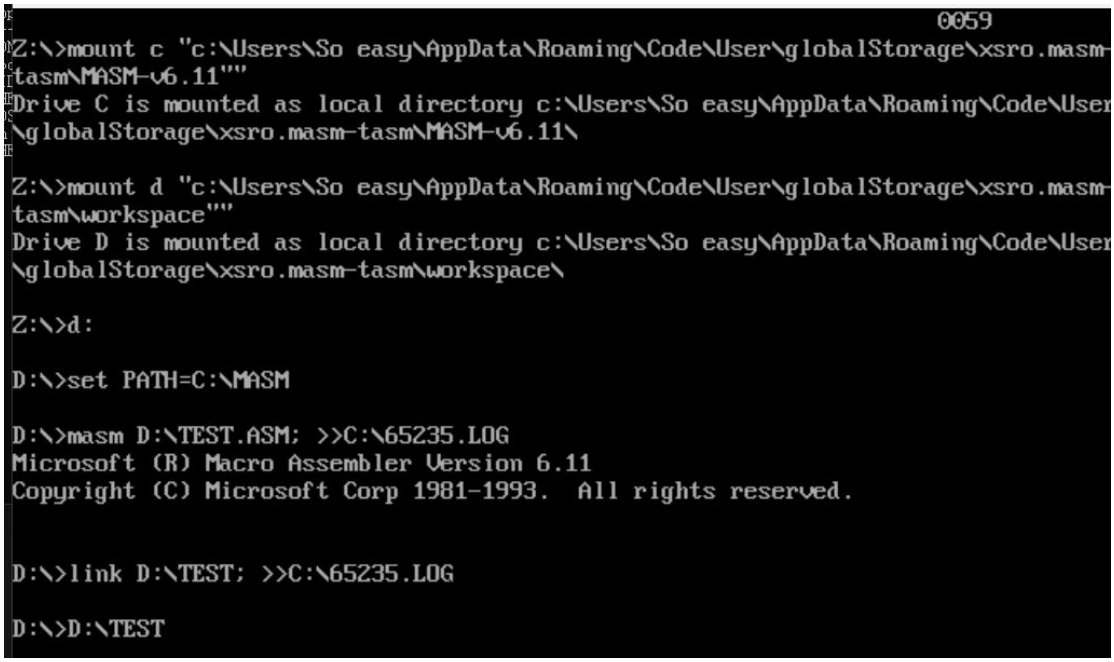
Part 3: Interrupts on DOS, investigation of the Timer and Keyboard interrupts

Compile and run the k1.asm program in the Dos Box emulator. This code will print “Hello” on the screen each time the ‘h’ key is pressed on the keyboard. The keyboard interrupt number is IRQ1 or INT9 found at address 0x24 in the vector table. Hardware interrupts are normally given an IRQ number on a DOS machine the IRQ number is the interrupt number less 8.

```
.MODEL small
.STACK
.DATA
.CODE
```

```
.STARTUP
jmp ov1
dseg        db '0','0','0','0'
ov1:
push es
mov    ax, 0000h
mov    es, ax
mov    bx, 020h
mov    ax, es:[bx]
mov    vtabip, ax
add    bx, 2
mov    ax, es:[bx]
mov    vtabcs, ax
pop    es

push es
mov    ax, 0000h
mov    es, ax
mov    bx, 020h
mov    ax, OFFSET isr
mov    es:[bx], ax
add    bx, 2
mov    ax, cs
mov    es:[bx], ax
pop    es

mov    ah, 031h
mov    al, 00h
mov    dx, 1024
int    021h
jmp    quit

isr:
pusha
mov    cx, 4
mov    bx, OFFSET dseg

back:
mov    al, cs:[bx]
inc    al
mov    cs:[bx], al
cmp    al, ':'
jnz    skip
```

```
mov    al, '0'
mov    cs:[bx], al
inc    bx
loop back

skip:
mov    ax, 0b800h
mov    es, ax
mov    bx, OFFSET dseg
mov    dx, 140
mov    cx, 4

nextc:
mov    al, cs:[bx]
xchg bx, dx
mov    es:[bx], al
xchg bx, dx
inc    bx
sub    dx, 2
loop nextc

ovr:
mov    dx, 020h
mov    al, 020h
out    dx, al
popa
db      0EAh

vtabip dw 0
vtabcs dw 0
quit: .EXIT
END
```

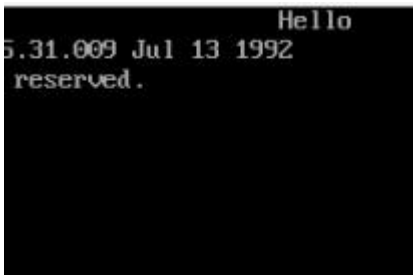


1/ Create a new program called kt1.asm that will increment the counter each time the <space> key is pressed on the keyboard (required). I would suggest starting with k1.asm and resaving it as kt1.asm and then adding the code you need from t1.asm. See the listing later in this handout for help but use the downloaded asm files to avoid rogue characters appearing.

```
.MODEL small
.STACK
.DATA
.CODE

.STARTUP
push    es
mov     ax, 0000h
mov     es, ax
mov     bx, 024h
mov     ax, es:[bx]
mov     vtabip, ax
add     bx, 2
mov     ax, es:[bx]
mov     vtabcs, ax
pop     es
push    es
mov     ax, 0000h
mov     es, ax
mov     bx, 024h
mov     ax, OFFSET isr
mov     es:[bx], ax
```

```
add         bx, 2
mov         ax, cs
mov         es:[bx], ax
pop         es
mov         ah, 031h
mov         al, 00h
mov         dx, 1024
int         021h

jmp         quit

isr:
pusha
mov         dx, 60h
in          al, dx
and         al, 127
cmp         al, 23h
jnz         ovr1
mov         ax, 0b800h
mov         es, ax
mov         bx, 140
mov         al, 'H'
mov         es:[bx], al
add         bx, 2
mov         al, 'e'
mov         es:[bx], al
add         bx, 2
mov         al, 'l'
mov         es:[bx], al
add         bx, 2
mov         al, 'l'
mov         es:[bx], al
add         bx, 2
mov         al, 'o'
mov         es:[bx], al
jmp         ovr2

ovr1:
mov         ax, 0b800h
mov         es, ax
mov         bx, 140
mov         cx, 5
again:
mov         al, ' '
```

```
mov         es:[bx], al
add         bx, 2
loop        again

ovr2:
mov         dx, 020h
mov         al, 020h
out         dx, al

popa
db          0EAh

vtabip    dw 0
vtabcs    dw 0

quit:     .EXIT
```

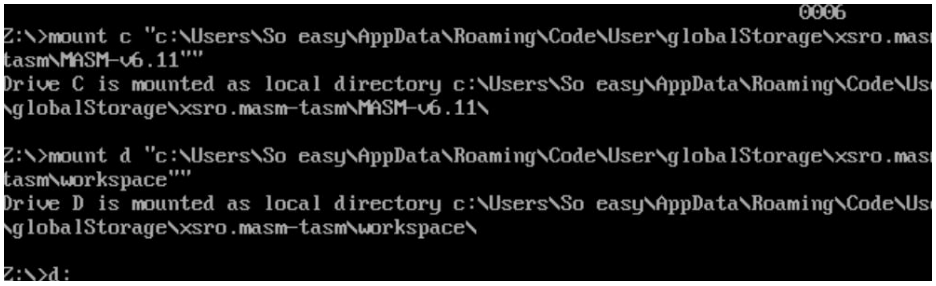


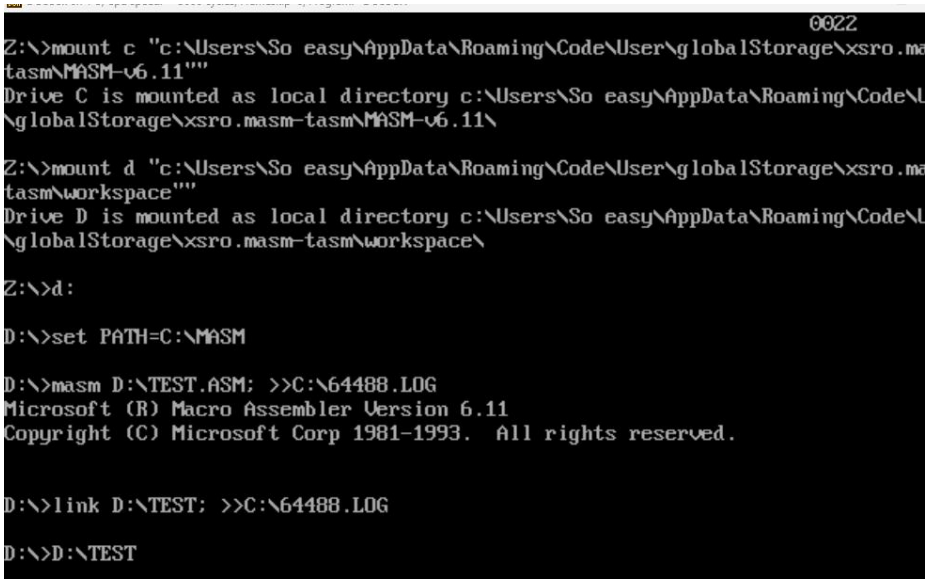
1/ Create a new program called kt1.asm that will increment the counter each time the <space> key is pressed on the keyboard (required). I would suggest starting with k1.asm and resaving it as kt1.asm and then adding the code you need from t1.asm. See the listing later in this handout for help but use the downloaded asm files to avoid rogue characters appearing.

```
.MODEL small
.STACK
.DATA
dseg db '0','0','0','0'                    ; Data segment storage for TSR
.CODE
.STARTUP
jmp ov1                                    ; Bypass data initialization in execution

ov1:
push es                                    ; Preserve extra segment register
mov ax, 0000h                              ; Zero out AX for DS reset
mov es, ax                                 ; Set extra segment to page 0 (IVT)
mov bx, 024h                               ; Address for INT 9 (keyboard IRQ1)
mov ax, es:[bx]                            ; Retrieve CS from IVT
mov vtabip, ax                             ; Store original CS
```

```
inc bx
inc bx
mov ax, es:[bx]                    ; Retrieve IP from IVT
mov vtabcs, ax                     ; Store original IP
pop es                             ; Restore ES register
push es                            ; Prepare ES for ISR address set
mov ax, 0000h                      ; Reset AX for DS
mov es, ax                         ; Set ES to page 0 (IVT)
mov bx, 024h                       ; Set up for new ISR
mov ax, OFFSET isr                 ; Address of new ISR
mov es:[bx], ax                    ; Set IP in IVT
inc bx
inc bx
mov ax, cs                         ; Segment of current program
mov es:[bx], ax                    ; Update CS in IVT
pop es                             ; Restore ES register
mov ah, 031h                       ; AH = 31h for TSR
mov al, 00h                        ; AL = 00h
mov dx, 1024                       ; Space to keep resident
int 021h                           ; TSR interrupt

jmp quit                           ; Never execute beyond this in normal flow

isr:
pusha                              ; Save all general registers
mov dx, 60h                        ; Port 60h, keyboard data
in al, dx                          ; Read scan code
cmp al, 1Ch                        ; Check if Enter key scan code
jz ovr1                            ; Jump if Enter pressed
cmp al, 39h                        ; Check if Space key scan code
jnz ovr2                           ; Jump if not Space

mov cx, 4                          ; Counter for loop
mov bx, OFFSET dseg
back:
mov al, cs:[bx]                    ; Get current counter digit
inc al                             ; Increment character code
mov cs:[bx], al                    ; Store back to memory
cmp al, ':'                        ; Check if rollover is needed
jnz skip                           ; Skip if no rollover
mov al, '0'                        ; Reset to '0'
mov cs:[bx], al                    ; Store reset digit
inc bx                             ; Move to next digit
loop back                          ; Loop back for next digit
```

```
ovr1:
mov cx, 4                          ; Counter for clearing digits
mov bx, OFFSET dseg
back1:
mov al, '0'                        ; Set all digits to '0'
mov cs:[bx], al                    ; Store '0' to each digit
inc bx                             ; Next digit
loop back1                         ; Loop for all digits

skip:
mov ax, 0b800h                     ; Video memory for text mode
mov es, ax                         ; Set ES to video memory
mov bx, OFFSET dsg                 ; Start of digits in memory
mov dx, 140                        ; Screen offset
mov cx, 4                          ; Digit count
nextc:
mov al, cs:[bx]                    ; Get digit from storage
xchg bx, dx                        ; Swap BX and DX for storing
mov es:[bx], al                    ; Write digit to video memory
xchg bx, dx                        ; Swap back
inc bx                             ; Next digit in memory
sub dx, 2                          ; Next screen position
loop nextc                         ; Repeat for all digits

jmp ovr2                           ; Jump to ISR end routine

ovr2:
mov dx, 020h                       ; End of interrupt command port
mov al, 020h                       ; Command for EO
out dx, al                         ; Send EOI to PIC
popa                               ; Restore all general registers

db 0EAh                            ; Opcode for JMP far
vtabip dw 0                        ; Placeholder for original IP
vtabcs dw 0                        ; Placeholder for original CS

quit:
.EXIT                              ; Exit and stay resident

END                                ; End of program marker
```





2/ Reset the counter to 0000 when the <ret> is pressed (optional).