University of Central Punjab

(Incorporated by Ordinance No. XXIV of 2002 promulgated by Government of the Punjab)

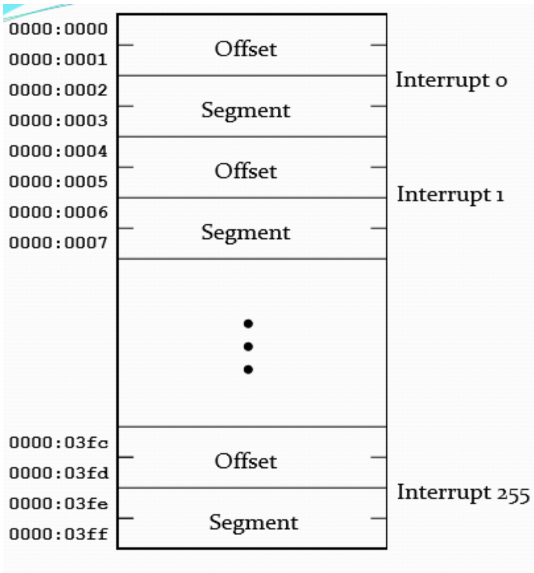
FACULTY OF INFORMATION TECHNOLOGY

Computer Organization and Assembly Language

Lab 14	
Topic	1. Interrupts 2. Interrupts hooking

Interrupt vector table-address mapping

- Offset: n*4 ; offset address of n^{th} interrupt
- Segment: n*4+2 ; base address of n^{th} interrupt



If N is the interrupt number then following operations are executed by the INT and IRET by the processor.

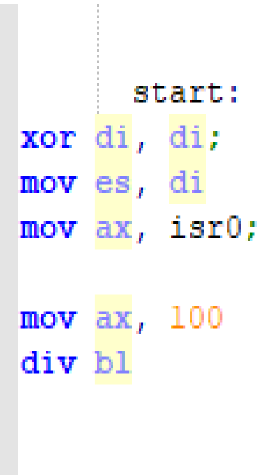
The operation of INT can be written as:

- sp ← sp-2
- [sp] ← flag
- sp ← sp-2
- if ← 0
- tf ← 0
- [sp] ← cs
- sp ← sp-2
- [sp] ← ip
- ip ← [0:N*4]
- cs ← [0:N*4+2]

The operation of IRET can be written as:

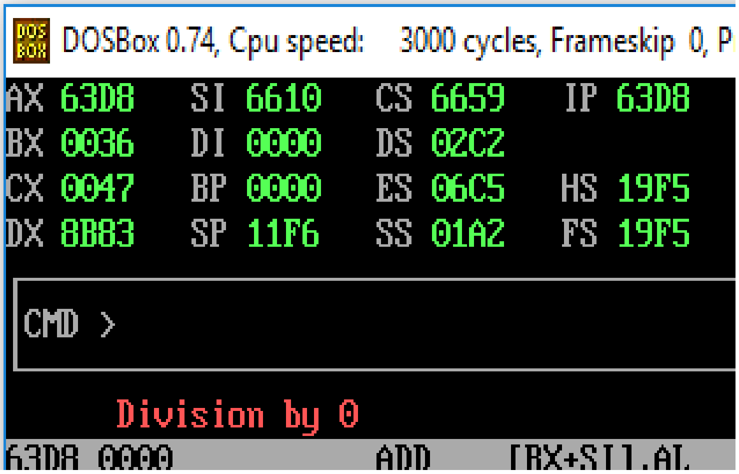
- ip ← [sp]
- sp ← sp+2
- cs ← [sp]
- sp ← sp+2
- flag ← [sp]
- sp ← sp+2

Interrupt zero: INT 0



```
     start:
xor di, di;
mov es, di
mov ax, isr0;

mov ax, 100
div bl
```



DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, P

```
AX 63D8    SI 6610    CS 6659    IP 63D8
BX 0036    DI 0000    DS 02C2
CX 0047    BP 0000    ES 06C5    HS 19F5
DX 8B83    SP 11F6    SS 01A2    FS 19F5

 CMD >

       Division by 0
63D8 0000             ADD     [BX+SI],AL
```

Example 3: ISR(Interrupt Service Routine) hooking-interrupt zero

```
[org 0x100]
jmp start
    message db 'Your message for divide overflow',0;



 isr0:

          pop ax                      ;pop the IP of div instruction
          push continue               ;push the IP of next instruction after "DIV"

   mov ax, 0xb800
   mov es, ax;
   mov si, message;
   mov ah,7
   nextchar:
   lodsb;
   cmp al, 0
   je skip
   stosw
   jmp nextchar
   skip:
iret

start:
xor di, di;
mov es, di
mov ax, isr0;
mov [es:0h*4],ax;
mov [es:0H*4+2], cs;
mov ax, 100

div bl                               ;when div interrupt is called it pushes the IP value of itself
                                     ;instead of the next instruction from where our code
                                     ; should continue after returning from interrupt.

continue:



mov ax,0x4c00
int 21h
```

Example 4: Another Interrupt hooking

```
[org 0x100]

jmp start


ISR0:

   MOV AX, 0XB800

   MOV ES, AX;

   MOV word [ES:0], 0X0741;

IRET


start:

XOR DI, DI;

MOV ES, DI

mov AX, ISR0;

MOV [ES:16h*4],AX;

MOV [ES:16h*4+2], CS;

mov ah,0;

int 0x16;


mov ax,0x4c00

int 21h
```

Note: After executing this interrupt, the contents of IVT against int 0x16 has been overwritten so the keyboard will not work properly.

Example 5: Interrupt hooking without using INT instruction

```
[org 0x100]
jmp start


ISR0:
   MOV AX, 0XB800
   MOV ES, AX;
   MOV word [ES:2], 0X0741;


IRET


start:
XOR DI, DI;
MOV ES, DI
mov AX, ISR0;
MOV [ES:17h*4],AX;
MOV [ES:17h*4+2], CS;
Pushf           ;push flag register
push cs         ;push code segment
push continue ;push IP (address of next instruction where to return)
jmp far [es:17h*4]      ;calling interrupt
continue:


mov ax,0x4c00
int 21h
```

Example 6: Interrupt unhooking.

```
[org 0x100]
jmp start

old_data: dd 0

ISR0:
   MOV AX, 0XB800
   MOV ES, AX;
   MOV word [ES:0], 0X0741;

          mov ax,0
          mov es,ax

          mov bx,[old_data]
          mov [ES:0x16*4],bx     ;saving the old values in a variable before overwritting.
          mov bx,[old_data+2]
          mov [ES:0x16*4+2],bx

          IRET

start:

XOR DI, DI;
MOV ES, DI
mov AX, ISR0;
mov bx,[ES:0x16*4]     ;saving the old values in a variable before overwriting.

mov [old_data],bx

mov bx,[ES:0x16*4+2]

mov [old_data+2],bx



MOV [ES:0x16*4],AX;              ;hooking the interrupt

MOV [ES:0x16*4+2], CS;


pushf

push cs

push continue

jmp far [es:0x16*4]

continue:


mov ax,0x4c00

int 21h
```

Note: Recover the old contents of IVT after executing your functionality via hooking.