**Lab Manual – Keyboard Interrupt**

**Activity 1:** Write a program “tsr.com” that saves old screen in a buffer (using MOVS instruction), clears the screen and then restores the old screen saved in buffer. Add some delay after clear screen call to clearly see the functionality. Properly make three separate functions.

**Output:** Running tsr.com on DOSBOX should show empty screen for some time and the welcome screen should reappear.

|  |
| --- |
| abc: times 32 dw 0 ; space for 32 words  xyz: times 256 dw 0 ; space for 256 words |

**Activity 2:** Update your program “tsr.com” and hook keyboard interrupt such that if user **presses** key ‘A’ on keyboard it clears the screen and restores it **on release** after some delay. Do not send key ‘A’ to original ISR (both the press and release codes), rest of the keys should be passed to original ISR.

**[If TSR Functionality covered in class]** Your program should just hook the interrupt, make it TSR and leave. Do not write any loop in start.

**[Otherwise]** keep taking keys in a loop, exit on escape as we did in class. Print message to check the original BIOS functionality.

(If your program doesn’t work properly see help given in the end).

**Activity 3:** Assemble following piece of code and save its two copies as “printa.com” and “printb.com”.

|  |
| --- |
| ; hello world at desired screen location  [org 0x0100]  jmp start  ; subroutine to clear the screen  clrscr: push es  push ax  push di  mov ax, 0xb800  mov es, ax ; point es to video base  mov di, 0 ; point di to top left column  nextloc: mov word [es:di], 0x0720 ; clear next char on screen  add di, 2 ; move to next screen location  cmp di, 4000 ; has the whole screen cleared  jne nextloc ; if no clear next position    pop di  pop ax  pop es  ret  ;---------------------------------------------------------------------------  printRectangle: push bp  mov bp, sp  push es  push ax  push cx  push si  push di  mov ax, 0xb800  mov es, ax ; point es to video base  mov al, 80 ; load al with columns per row  mul byte [bp+12] ; multiply with row number  add ax, [bp+10] ; add col  shl ax, 1 ; turn into byte offset  mov di, ax ; point di to required location  mov ah, [bp+4] ; load attribute in ah  mov cx, [bp+6]  sub cx, [bp+10]  topLine: mov al, 0x2D ; ASCII of '-'  mov [es:di], ax ; show this char on screen  add di, 2 ; move to next screen location  call sleep;  loop topLine ; repeat the operation cx times  mov cx, [bp+8]  sub cx, [bp+12]  add di, 160  rightLine: mov al, 0x7c ; ASCII of '|'  mov [es:di], ax ; show this char on screen  add di, 160 ; move to next screen location  call sleep;  loop rightLine ; repeat the operation cx times    mov cx, [bp+6]  sub cx, [bp+10]  sub di, 2  bottomLine: mov al, 0x2D ; ASCII of '-'  mov [es:di], ax ; show this char on screen  sub di, 2 ; move to next screen location  call sleep;  loop bottomLine ; repeat the operation cx times  mov cx, [bp+8]  sub cx, [bp+12]  sub di, 160  leftLine: mov al, 0x7c ; ASCII of '|'  mov [es:di], ax ; show this char on screen  sub di, 160 ; move to next screen location  call sleep;  loop leftLine ; repeat the operation cx times  pop di  pop si  pop cx  pop ax  pop es  pop bp  ret 10  ;---------------------------------------------------------------------------  sleep: push cx  mov cx, 0xFFFF  delay: loop delay  pop cx  ret  ;---------------------------------------------------------------------------  start: call clrscr ; call the clrscr subroutine    mov ax, 2  push ax ; push top  mov ax, 10  push ax ; push left  mov ax, 20  push ax ; push bottom  mov ax, 60  push ax ; push right number    mov ax, 0x44 ; Red FG  push ax ; push attribute  call printRectangle ; call the printstr subroutine    ;---------------------------------------------------------------------------  mov ax, 0x4c00 ; terminate program  int 0x21 |

**Activity 4:** Run following test case and verify the functionality of your TSR.

1. Open DOSBOX and perform following operations on same DOSBOX window.
2. Open your tsr.com. **Expected Result:** Command Prompt should return properly.
3. Type ‘A’. **Expected Result:** Welcome screen should disappear for a while (duration of your delay) and reappear. Command prompt should not display A.
4. Open “printa.com”. **Expected Result:** Command prompt should display “print” and then the screen should disappear and reappear.
5. Keep entering ‘a’ and see the behavior.
6. Type “printb.com” and open the file. **Expected Result:** It should open and print rectangle successfully.
7. Open “printb.com” again and while rectangle in printing press key ‘A’ 3-4 times. **Expected Result:** It should show empty screen for some time and rectangle should reappear (in the state when you pressed A).

**Activity 5:** What did you learn?

**Help:** If you want ds:si or es:di to point to your screenBuffer make ds=cs or es=cs to point it to your segment.