**Lab Manual – Keyboard Interrupt**

**Activity 0:** Assemble following piece of code and save its two copies as “printa.com” and “printb.com”. This code prints a rectangle on screen. Run and verify its functionality.

|  |
| --- |
| ; 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 1:** Write a program “PrintScreen.com” that saves old screen in a buffer (using MOVS instruction), clears the screen and then restores the old screen saved in buffer. Code to declare buffer is given below. Test your functions with following main program:

|  |
| --- |
| start:  **call saveScreen ;your function that will save the screen in buffer**  ;getch() code below, wait for user to enter any key  mov ah, 0 ; service 0 – get keystroke  int 0x16  ;after key entered, execute following code  call clrscr ; this will clear the screen if you enter any key  ;getch() code below, wait for user to enter any key  mov ah, 0 ; service 0 – get keystroke  int 0x16  ;after key entered, execute following code  **call restoreScreen ;your function to paste content of buffer back on screen**  ;getch() code below, wait for user to enter any key  mov ah, 0 ; service 0 – get keystroke  int 0x16  mov ax, 0x4c00 ; terminate program  int 0x21 |

**Expected Output:** Running PrintScreen.com on DOSBOX should wait for user to enter any key. Upon entering key it will show empty screen and wait for the key again. Upon entering another key it will show the previous screen (DOSBOX Introduction Screen) again and wait for another key to exit. Upon entering the key again, program will terminate and command prompt will appear again.

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

**Activity 2:** Write a program “tsr.com” that hooks the keyboard interrupt to your kbisr, makes it TSR (Terminate and Stay Resident) and leaves (the code is given in TSR example in textbook and we have done it in class). Your KBISR should save the screen in buffer and clear the screen **if user presses key ‘B’**. Also your KBISR should restore the screen **on release of key “B”**. Do not send key ‘B’ to original ISR (both the press and release codes), rest of the keys should be passed to original ISR (i.e. selective chaining).

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

**Activity 3:** Run following test case and verify the functionality of your TSR. Open DOSBOX and perform following operations on same DOSBOX window.

1. Run star.com. It should show the star moving and command prompt should come back.
2. Run clrscr.com. It should clear the screen and command prompt should return.
3. Run printa.com. It should print the rectangle on screen and command prompt should return.
4. Run your tsr.com. **Expected Result:** Command Prompt should return properly.
5. Type ‘B’. **Expected Result: Previous** screen should disappear for a while and reappear on releasing B. Command prompt should not display B.
6. Run “printb.com”. **Expected Result:** Command prompt should display “print” and then the screen should disappear and reappear.
7. Keep entering ‘b’ and see the behavior.
8. Type “printa.com” and run the file. **Expected Result:** It should print rectangle successfully.
9. Run AFD and quit AFD. Command prompt should come back.
10. Run “printa.com” again and while rectangle in printing press key ‘B’ 3 to 4 times. **Expected Result:** It should show empty screen for some time and rectangle should reappear (in the state when you pressed B).

**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.