National University of Computer and Emerging Sciences



Lab Manual 13

Computer Organization and Assembly Language Lab

Department of Computer Science

FAST-NU, Lahore, Pakistan

**Problems – Software Interrupts**

**Practice Problems:**

**Activity 1:** Following program keeps taking a key from the user and filling the screen with this key. Fix the code such that it exits when user presses ESC (Escape).

|  |
| --- |
| ; Infinite Key Printing  [org 0x0100]  jmp start  ;------------------------------------------------  printKey: push ax  pop bx ; bx=ax  push es  push ax  push cx  push di    mov ax, 0xb800  mov es, ax ; point es to video base  xor di, di ; point di to top left column  mov al, bl  mov ah, 0x07 ; normal attribute  mov cx, 2000 ; number of screen locations  cld ; auto increment mode  rep stosw ; clear the whole screen  pop di  pop cx  pop ax  pop es  ret  ;------------------------------------------------  start: mov ah, 0 ; service 0 – get keystroke  int 0x16 ; call BIOS keyboard service  call printKey ; clear the screen  jmp start  mov ax, 0x4c00 ; terminate program  int 0x21 |

**Activity 2:** Following code takes keys from user and upon getting a key it prints a message on screen. Update this code and use BIOS service for message printing instead of using all following subroutines. (Do not clear the Screen.)

|  |
| --- |
| ;Example 8.3 - print string and keyboard wait using BIOS services  [org 0x0100]  jmp start  msg1: db 'hello world', 0  msg2: db 'hello world again', 0  msg3: db 'hello world again and again', 0  ;------------------------------------------------  strlen: push bp  mov bp,sp  push es  push cx  push di  les di, [bp+4] ; point es:di to string  mov cx, 0xffff ; load maximum number in cx  xor al, al ; load a zero in al  repne scasb ; find zero in the string  mov ax, 0xffff ; load maximum number in ax  sub ax, cx ; find change in cx  dec ax ; exclude null from length  pop di  pop cx  pop es  pop bp  ret 4  ;------------------------------------------------  clrscr: push es  push ax  push cx  push di  mov ax, 0xb800  mov es, ax ; point es to video base  xor di, di ; point di to top left column  mov ax, 0x0720 ; space char in normal attribute  mov cx, 2000 ; number of screen locations  cld ; auto increment mode  rep stosw ; clear the whole screen  pop di  pop cx  pop ax  pop es  ret  ;------------------------------------------------  printstr: push bp  mov bp, sp  push es  push ax  push cx  push si  push di  push ds ; push segment of string  mov ax, [bp+4]  push ax ; push offset of string  call strlen ; calculate string length  cmp ax, 0 ; is the string empty  jz exit ; no printing if string is empty  mov cx, ax ; save length in cx  mov ax, 0xb800  mov es, ax ; point es to video base  mov al, 80 ; load al with columns per row  mul byte [bp+8] ; multiply with y position  add ax, [bp+10] ; add x position  shl ax, 1 ; turn into byte offset  mov di,ax ; point di to required location  mov si, [bp+4] ; point si to string  mov ah, [bp+6] ; load attribute in ah  cld ; auto increment mode  nextchar: lodsb ; load next char in al  stosw ; print char/attribute pair  loop nextchar ; repeat for the whole string  exit: pop di  pop si  pop cx  pop ax  pop es  pop bp  ret 8  ;------------------------------------------------------  start: mov ah, 0 ; service 0 – get keystroke  int 0x16 ; call BIOS keyboard service  call clrscr ; clear the screen  mov ah, 0 ; service 0 – get keystroke  int 0x16 ; call BIOS keyboard service  mov ax, 0  push ax ; push x position  mov ax, 0  push ax ; push y position  mov ax, 1 ; blue on black  push ax ; push attribute  mov ax, msg1  push ax ; push offset of string  call printstr ; print the string  mov ah, 0 ; service 0 – get keystroke  int 0x16 ; call BIOS keyboard service  mov ax, 0  push ax ; push x position  mov ax, 0  push ax ; push y position  mov ax, 0x71 ; blue on white  push ax ; push attribute  mov ax, msg2  push ax ; push offset of string  call printstr ; print the string    mov ah, 0 ; service 0 – get keystroke  int 0x16 ; call BIOS keyboard service  mov ax, 0  push ax ; push x position  mov ax, 0  push ax ; push y position  mov ax, 0xF4 ; red on white blinking  push ax ; push attribute  mov ax, msg3  push ax ; push offset of string  call printstr ; print the string  mov ah, 0 ; service 0 – get keystroke  int 0x16 ; call BIOS keyboard service  mov ax, 0x4c00 ; terminate program  int 0x21 |

**in -Lab Questions:**

**Activity 3:** Run the code given below and see what it does? Check Service Details URL for description.

|  |
| --- |
| ; INT 21 DOS Service 9  ; Reference: http://www.delorie.com/djgpp/doc/rbinter/id/73/25.html  ; Following piece of code prints a $ terminated string on Console using DOS Service # 9  [org 0x0100]  jmp start  msg1: db 'Hello World ...$' ;'$' terminated string  start: mov dx, msg1 ; ds:dx points to '$' terminated string  mov ah, 9 ; service 9 – WRITE STRING TO STANDARD OUTPUT  int 0x21 ; dos services    mov ax, 0x4c00 ; terminate program  int 0x21 |

**Activity 4:** Run the code given below and see what it does? Check Service Details URL for description.

|  |
| --- |
| ; example 12.5 - buffer input using dos services  [org 0x0100]  jmp start  message: db 'User Entered following string:$'  ;following is input buffer in format required by service  buffer: db 80 ; Byte # 0: Max length of buffer  db 0 ; Byte # 1: number of characters on return  times 80 db 0 ; 80 Bytes for actual buffer space  ;buffer ends  start: mov dx, buffer ; input buffer (ds:dx pointing to input buffer)  mov ah, 0x0A ; DOS' service A – buffered input  int 0x21 ; dos services call  mov bh, 0  mov bl, [buffer+1] ; read actual size in bx i.e. no of characters user entered  mov byte [buffer+2+bx], '$' ; append $ at the end of user input  mov dx, message ; message to print  mov ah, 9 ; service 9 – write string  int 0x21 ; dos services  mov dx, buffer+2 ; user input buffer  mov ah, 9 ; service 9 – write string  int 0x21 ; dos services  mov ax, 0x4c00 ; terminate program  int 0x21 |

|  |
| --- |
| **Format of DOS input buffer:**  Offset Size Description  00h BYTE maximum characters buffer can hold  01h BYTE (call) number of chars from last input which may be recalled  (ret) number of characters actually read, excluding CR  02h N BYTEs actual characters read, including the final carriage return |

**Activity 5:**

Do the following using software interrupt.

1. Create a .txt file in the the current directory.
2. Open , read and close the existing 123.txt file.
3. Print the data of the . txt file.
4. Find you roll-no from the file and store it in Ax

Note: Create a txt file named 123.txt in the directory. The content of the file is as follow:

My name is XYZ.

I am currently doin BS- BDS from FAST - NUCES, Lahore.

My roll no is #12-3456.