0261

National University of Computer and Emerging Sciences

Lahore Campus

# COAL (EE2003)

MAY

## Sessional-II Exam

**Computer organization and assembly language**

Date: November 5th 2024

**Total Time (Hrs):** 1

**Total Marks:** 40

**Total Questions:** 2

40

**Course Instructor(s)**

AA,AA,SF,SI,SM

MUZAMMIL SHAHID

23L-1004 BCS-3E

Roll No Section

Student Signature

**Instructions :** Attempt all questions. It is an open book exam only Assembly Language Programming Lecture Notes are allowed. Calculators are allowed. You can use rough sheets.

***CLO #:2 Describe the working of important x86 assembly primitives, including arithmetic, branching, bit manipulation, addressing modes and interrupt handling.***

**Q1: [marks 4x2 + 2x2 + 2 + 2 +2 + 2 = 20]**

offset of ISR: 0x01DC ✓

segment of ISR: 0x01DE ✓

| | |
|---|---|
| **(a)** The segment and offset of interrupt service routine of int 77h are placed at: offset: 0x01DC segment: 0x0000<br>**(b)** what is the total size of the IVT table? 1024 B (1KB)<br>**(c)** Which registers are changed by the iret instruction? ip, cs, sp, flag<br>**(d)** Which interrupt is hooked by the instructions given on right side? $(64)_{16}$ | **Code for part (d)**<br>mov ax, 0<br>mov es, ax<br>mov [es:400], mySub<br>mov [es:402], cs |

**(e)** Replace the following independent invalid instructions with a single instruction that has the same effect.

| | | |
|---|---|---|
| i) | mov ax, [ss:sp]<br>add sp, 2 | **Solution:** pop ax ✓ |
| ii) | sub sp,2<br>mov [ss:sp], ax | **Solution:** push ax ✓ |

**(f)** Write a code to swap two registers ax and bx without using temporary space (local variable) on stack or memory. You are only allowed to use **stack operations.**

**Solution:**

```
push ax
push bx
pop  ax
pop  bx
```

✓

| | |
|---|---|
| **(g)** What would be the value of SP after execution of the following code?<br>Initial value of SP = 0xFFFE<br>**Solution:**<br>SP = 0xFFF8 ✓ | Jmp start<br>Routine:<br>Ret<br>start:<br>Call routine<br>Push ax 0xFFFC<br>Sub sp,4 0xFFF8 |

**(h)** Complete the following code to place asterisk '*' character on the left diagonal of the screen.

1. Mov ax,0xb800
2. Mov es, ax
3. Mov ax,0x0742

20

02

4. Mov cx, 25
5. L1: Mov word [es: si] , ax ; assuming that si = 0 initially
6. add si, 162
7. Loop l1

(i)

| Suppose the following declarations have been made<br>str1: db 'FGHIJ'<br>str2: db 'ABCDE00000 '<br>Write instructions to move str1 to the end of str2, producing the string 'ABCDEFGHIJ' using string instructions | **Solution:**<br>mov si, str1<br>push ds<br>pop es<br>mov di, str2<br>add di, 5<br>mov cx, 5<br>cld<br>rep movsb |
|---|---|

***CLO #:2 Describe the working of important x86 assembly primitives, including arithmetic, branching, bit manipulation, addressing modes and interrupt handling.***

Q2: Write a subroutine to swap the odd rows with the even ones in the video memory i.e. swap row 0 with row 1. Row 2 with row 3 and so on using string instructions. **[20 marks]**

**Solution:**

```
swapOddEven:
  push bp
  mov bp, sp

  push ax  ; Can be replaced with pusha
  push bx
  push es
  push di
  push ds
  push si
  push cx
  push dx
        ; FUNC START
  mov cx, 1000 ; 1000 iterations required since each iteration
               will deal with two cells (2000/2 = 1000)
  mov ax, 0xb800
  mov ds, ax
  xor si, si
```

PTO →

```
L1:
  cld
  LODSW      ; Saving even row to AX
  push ds
  pop es
  mov di,si  - moving [ds:si] to [es:di] to make [es:di] point
  mov bx,ax    towards the writing location
  LODSW      -> saving initial value into anothes registes to
                make way for second value
  std        -> moving second value
  mov dx,ax  ; moving second value to anothes register
  mov ax,bx  ; moving first value to ax
  STOSW      ; writing first value to [es:di], di will now decsement since
                                                df was set
  mov ax,dx  ; moving second value to ax
  STOSW      ; writing second value to [es:di]
Loop L1      ; looping 1000 times

pop dx  ; FUNC END
pop cx  ; can be replaced with popa
pop si
pop ds
pop di
pop es
pop bx
pop ax
pop bp

ret
```

20