1+2+3+2·5+3+2+12 = 25·5 P/

National University of Computer and Emerging Sciences, Lahore Campus

Course:	COAL	Course Code:	EE2003
Program:	BSCS BSDS BSR	Semester:	Fall 2023
Duration:	1 Hour	Total Marks:	30
Paper Date:	28-Sept-2023	Page(s):	3
Section:	All	Roll No.	22L-7503
Exam:	Midterm - I		

Instruction/Notes: **This is an open notes/book exam. Sharing notes and calculators is NOT ALLOWED.** All the answers should be written in provided space on this paper. Rough sheets can be used but will not be collected and checked. In case of any ambiguity, make reasonable assumptions. Questions during exams are not allowed.

Question 1 [CLO 1, 2] [15 Marks]: Answer following short questions:

i. **[1 Mark]** How many number of address lines (no. of bits) are required to access 2GB memory? 31 ✓ 1

ii. **[2 Marks]** SS = 0x012D, DS = 0x3F22 and BP = 0x00F2. Calculate the physical memory address of the destination operand for following statement: **Mov word [bp] , 7**
Show your working to get credit.

SS is associated to BP by default

```
 012D0
+   F2   ✓ 2
 013C2
```

At 013C2h the number will be stored

iii. **[3 Marks]** What will be the values of AX and BX registers in hex after the execution of the following piece of code?

```
[ORG 0x0100]
jmp start
num1: dd 0x7E945FA2
num2: dd 0xB2654104

start:
mov ax, [num1+2]
mov bx, [num2+1]
```

AX = 7E94 ✓

BX = 6541 ✓

3

iv. **[3 Marks]** Identify whether the following combinations for addressing are valid or not. Each part is independent of others.

	Valid/invalid
Mov ax , [bx - si]	invalid ✓
Mov ax, [bx+ di + 0x0300]	valid ✓
Mov al, [bx + si]	invalid ✗
Mov ah, [bh]	invalid ✓
Mov ax, [bh + bl]	invalid ✓
Mov ax, [0x0200]	valid ✓

→ Subtraction 2·5 ✓
→ Type mismatch
→ subscript
→ reg + reg

v. **[3 Marks]** Write assembly language code that calculates 2's complement of a number in the AX register. Your code should not exceed 2 instructions. No credit will be given if code exceeds 2 instructions.

```
not ax
add ax, 1        ✓ 3
```

Considering number already stored

0011

vi. **[3 Marks]** Identify whether the following jumps will be taken or not taken. Each part is independent of others. Show your working to get credit.

	Taken/Not Taken	Show your working below

Mov ax, -1
Mov bx, 0xFFFF
Cmp ax,bx
Je l1
	Taken ✓	ax → FFFF (-1)
		bx → FFFF
		FFFF
		- FFFF
		0000 → Equal

Mov ax,0x1924
Mov cx, 0x0123
Sub cx,ax
JO l1
	Taken
	(0^1→1)
		ax → 1924
		cx → 0123
		0123
		-1924
		E7FF
		↳ 1000 → 0001
		↳ 1110 XOR → 1
		↳ XOR → 0

Mov ax, FFFFh
Mov bx, FFFFh
Add ax, bx
L1: Mov ax, 0
Mov bx, 0
Jnc L1
	Not Taken
	(Mov does not affect flag)
		ax → FFFF
		bx → FFFF
		ax → FFFE
		ax → 0
		bx → 0
		C2 = 1

Question 2 [CLO 3] [15 Marks]: Parity of a number is odd if the total number of 1s in its binary is odd. Following examples show different numbers, their binary and parity.

Number	0xA7	0xA3	0x94	0xFF	0x00
Binary	1010 0111	1010 0011	1001 0100	1111 1111	0000 0000
Total No of 1s	5	4	3	8	0
Parity	Odd	Even	Odd	Even	Even

Write a program that removes odd parity numbers from an array and keeps even parity numbers in start. A sample array before and after execution of required program is shown below:

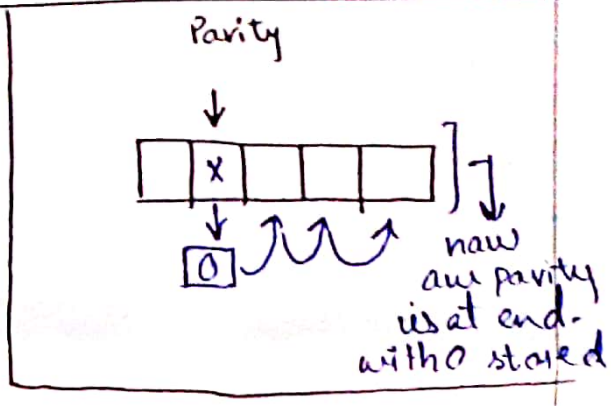
Array Before Execution:	0xA7, 0xA3, 0x94, 0xFF, 0x00
Array After Execution:	0xA3, 0xFF, 0x00, 0x00, 0x00 ;odd parity numbers have been removed

0011

Solution: (Your code should be generic for any size of array)

```
[ORG 0x0100]
jmp start
array:  db 0xA7, 0xA3, 0x94, 0xFF, 0x00
size: db 5
start:
;write your code below

    mov ax, [array+si]
    mov cx, [size]
    mov si, 0

l1:
    mov ax, [arr+si]
    and ax, ax
    jp compress

l2:
    inc si
    cmp si, cx
    jl l1
    jmp terminate

compress:
    mov [arr+si], 0
    mov di, si

l3:
    mov ax, [arr+di]
    mov bx, [arr+di+1]
    mov [arr+di+1], ax
    mov [arr+di], bx
    inc di
    cmp di, cx
    jl l3
    jmp l2

terminate:
MOV AX, 0x4C00      ; Terminate Program
INT 0x21
```

Structure of Compressed:



; Checking if the number is odd parity or not

5+5+2

12

; moving to next value and incrementing pointer

; making number 0, and hold current index

; swapping the number and sending them to end.