


National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Computer Org. & Assembly Lang.	Course Code:	EE2003, EE1005
	Program:	BS (CS), BS (DS)	Semester:	Spring 23
	Exam Duration:	60 mins	Total Marks:	40
	Date:	27-2-2023	Weight	15%
	Exam Type:	Mid I	Pages:	7
Name: _____ Roll No.: _____ Section: _____				
Instructions: <ol style="list-style-type: none">1. Answer in the space provided. Rough sheets can be used, but they will not be collected or marked.2. If you think some information is missing then make an assumption and state it clearly.3. This is open book and open notes exam.4. Sharing of notes or calculators is strictly prohibited.				

Question 1 [CLO 1]

[5 marks]

i. _____ register holds the address of next instruction to be executed on CPU.

- A. Accumulator
- B. Base
- C. Program Counter
- D. Program Status Word

ii. Fill in the blank:

A component of the CPU ensures all devices in the computer, including memory, CPU, I/O are synchronized in time. That component is clock.

iii. Suppose you are designing a new CPU, and decide to include an instruction for exponentiation operation (x^y). This is an _____ decision.

- A. organization
- B. architecture

iv. For a computer with 25 bit address bus, the maximum possible memory that can be addressed is

- A. 1 MB
- B. 32 MB
- C. 32 KB
- D. 25 MB

v. In what category of instructions Control Unit is **not** involved

- A. Arithmetic
- B. Program Control
- C. Data movement
- D. Special
- E. None of the above

Question 2 [CLO 2]

A. Identify whether the following instructions are valid or invalid. In case of invalid, briefly state the problem.

[5 marks]

Instruction	Valid/Invalid and reason
mov byte [di], 2	valid
cmp [numA], [numB]	Can't have two memory operands
mov numB, 0x87	Label is not a valid destination
sbb cx, [cs:bp-8]	valid
xor bh, [ss:dx]	DX can't be used for indirect addressing

B. Find the first and the last physical address of a segment if the segment register contains 0xF014.

[2 marks]

SegmentReg $\times 16 = 0xF0140$

First physical addr = $F0140 + 0000 = F0140$

Last physical addr = $F0140 + FFFF = 0013F$ (carry discarded)

C. What physical address will be accessed in the following instruction if BX = 0xD029, SI = 0x4755, DS = 0x5000, SS = 0x9000.

mov [bx+si-60h], cx

[2 marks]

$BX + SI - 60 = 0x171E$ (carry discarded)

$DS \times 16 = 0x50000$

Physical addr = $5000 + 171E = 0x5171E$

D. Find out the value of three flags at the end of the following code?

[3 marks]

<pre> [org 0x100] jmp start num1: db 1Bh, 27h, 4Ch, 8Eh, 0h start: mov al,[num1] mov bl,[num1+1] add al, bl mov bl, [num1+2] add al, bl mov bl, [num1+3] add al, bl mov [num1+4], al mov ax, 0x4c00 int 0x21 </pre>	<p>OF = 1</p> <p>CF = 1</p> <p>SF = 0</p>
--	---

E. Considering the data given in above question (stored in memory label num1), complete the following table to show the data placement in memory. Remember, memory storage is shown in hex numbers.

[1 mark]

	0	1	2	3	4	5	6
DS:0113	1B	27	4C	8E	00		

Question 3 [CLO 3]

A. Write an assembly language code to reverse the data (bit sequence) stored in BX.
e.g. contents of BX before reversing: 734B (0111 0011 0100 1011)
after reversing: D2CE (1101 0010 1100 1110)

[7 marks]

```

    mov dx, 0
    mov cx, 16
L1:  shl bx, 1 ; OR shr bx, 1 then rcl dx, 1
    rcr dx, 1
    dec cx
    jnz L1
    mov bx, dx

```

Things to check

Loop with 16 iterations

shift and rotate

move to and back from DX (etc.)

- B. Write an assembly language program to perform pairwise scan operation on an array such that:
- If second element of the pair is even, then multiply 1st and 2nd element and store the result in first element place.
 - If second element of the pair is odd, then add 1st and 2nd element and store the result in first element place.
 - If the array contains odd number of elements, then leave the last element as it is.

Assume that the last element of the array is -1, an indicator to stop the array iteration. See the examples below for detail.

Example 1: even sized array (excluding the last element)

Given Array: 3, 5, 10, 9, 12, 16, -1

Updated Array: **8**, 5, **19**, 9, **192**, 16, -1

Example 2: odd sized array (excluding the last element)

Given Array: 3, 5, 10, 9, 12, 16, 23, -1

Updated Array: **8**, 5, **19**, 9, **192**, 16, 23, -1

Note: (1) You can assume that all numbers in the array fit in one byte. So the multiplication product will occupy one word. Code for 8 bit multiplication was discussed in class and textbook. You should utilize that.
(2) You can check the LSB to determine if a number is even or odd.

[10 marks]

You can start with the following template

```
[org 0x100]
```

```
jmp start
```

```
data: dw 3, 5, 10, 9, 12, 16, 23, -1
```

```
start:
```

```
[org 0x100]
```

```
    jmp start
```

```
data: dw 3, 5, 10, 9, 12, 16, 23, -1
```

```
start: mov bx,0
```

```
loop1: cmp word [data+bx], -1 ; check to stop loop
```

```
    jz end
```

```
    cmp word [data+bx+2], -1
```

```
    jz end
```

```
    mov ax, [data+bx+2] ; check next element either even or odd
```

```
    shr ax, 1
```

```
    jc odd
```

```
even:  mov cl, 8 ; multiply by using bit operations
```

```
    mov ax, [data+bx]
```

```
    mov dx, [data+bx+2]
```

```
    mov word [data+bx], 0 ; set result on first place to zero
```

```
chkbit: shr dx, 1
        jnc skip
        add [data+bx], ax
skip:    shl ax, 1
        dec cl
        jnz chkbit
        jmp next

odd:     mov ax, [data+bx] ; add elements in case of odd
        add [data+bx+2], ax

next:    add bx, 4 ; iterate to next elements
        jmp loop1

end:     mov ax, 0x4C00
        int 0x21
```

Things to check

loop for iterating over array

deciding even & odd

adding numbers

multiplying numbers

- End of Exam -