National University of Computer and Emerging Sciences

FAST School of Computing **Fall-2020** **Islamabad Campus**

EE-229: Computer Organization and Assembly Language

Serial No:

Sessional Exam 2

Total Time: 1 Hour

Total Marks: 45

_____

Signature of Invigilator

Wednesday, 25th November, 2020

Course Instructors

Ameen Chilwan, Shams Farooq, Farwa Batool,
Rohail Gulbaz.

_____ _____ _____ _____

Student Name Roll No Section Signature

DO NOT OPEN THE QUESTION BOOK OR START UNTIL INSTRUCTED.

Instructions:

1. Attempt on question paper. Attempt all of them. Read the question carefully, understand the question, and then attempt it.
2. No additional sheet will be provided for rough work. Use the back of the last page for rough work.
3. If you need more space write on the back side of the paper and clearly mark question and part number etc.
4. After asked to commence the exam, please verify that you have SIX (8) different printed pages including this title page. There are a total of 7 questions.
5. Calculator sharing is strictly prohibited.
6. Use permanent ink pens only. Any part done using soft pencil will not be marked and cannot be claimed for rechecking.

	Q-1	Q-2	Q-3	Q-4	Q-5	Q-6	Q-7	Total
Marks Obtained								
Total Marks	**5**	**7**	**4**	**8**	**8**	**5**	**8**	**45**

Question 1 [5 Marks]

Following program has a function **add1** that takes two numbers from the stack and if sum of these two numbers is greater than 0, it returns their sum through stack, otherwise it returns 0 through stack. Code has some logical errors. Correct those errors so that you can get correct answer in dx register. You can add or modify existing lines but you cannot remove any line.

```
.code
jmp start

add1:

push bp
mov bp, sp
sub sp,2        ;; local variable

mov sp, [bp + 2]
mov [bp - 2], sp

mov ax, [bp + 4]
add [bp - 2 ], ax

mov ax, [bp -2]
mov [bp + 8], ax

cmp word [bp + 8], 0
ja end

mov word [bp + 8], 0
end:

pop bp
ret 2

start:

mov ax,5
push ax

mov bx, 2
push bx

call add1
pop dx ;; value of DX should be 7

mov ax, 4c00h
Int 21h
```

```
;multiple solutions can exist
.code
jmp start

add1:

push bp
mov bp,sp
sub sp,2

mov ax , [bp + 4]
mov [bp - 2], ax

mov ax , [bp + 6]
add [bp - 2 ] , ax

mov ax, [bp -2]
mov [bp + 8] ,ax

cmp word [bp + 8] , 0
ja end

mov word [bp + 8], 0

end:
add sp,2
pop bp

ret 4

start:

sub sp,2

mov ax,5
push ax

mov bx, 2
push bx

call add1
pop dx  ;value of DX should be 7

mov ax, 4c00h
Int 21h
```

Question 2 [7 Marks]

Consider following code and fill the **FILL STACK**. After executing complete code, update registers. Also show how **STACK** will look after execution. Stack starts at address **0100H** whereas data starts (**Array offset**) at **000EH.** ***NOTE:*** *Consider Line number as Instruction address*

```
1   .model small
2   .stack 0100h
3   .data
4       ary db 1,2,3
5   .code
6   JMP Main
7   Sary proc
8       PUSH Bp
9       mov bp,sp
10      mov si,[bp+6]
11      mov cx,[bp+4]
12      cmp cx,0
13      je L1
14      add al,[si]
15      dec cx
16      inc si
17      PUSH si
18      PUSH cx
19      call Sary
20      L1:
21          POP BP
22  RET 4
23  Sary ENDP
24  Main proc
25      mov ax,@data
26      mov ds,ax
27      PUSH dx
28      PUSH di
29      PUSH bx
30      mov ax,OFFSET ary
31      PUSH ax
32      mov ax,lengthof ary
33      PUSH ax
34      mov ax,0
35      mov si,0
36      mov cx,0
37      CALL Sary
38  Main EndP
```

FILL STACK	
00C6	
00C8	
00CA	~~00E2~~
00CC	~~Returning address 00021~~
00CE	~~0000~~
00E0	~~0011~~
00E2	~~00EA~~
00E4	~~Returning address 00021~~
00E6	~~0001~~
00E8	~~0010~~
00EA	~~00F2~~
00EC	~~Returning address 00021~~
00FE	~~0002~~
00F0	~~000F~~
00F2	~~0000~~
00F4	~~Returning address 00039~~
00F6	~~0003~~
00F8	~~000E~~
00FA	0000
00FC	0000
00FE	0000

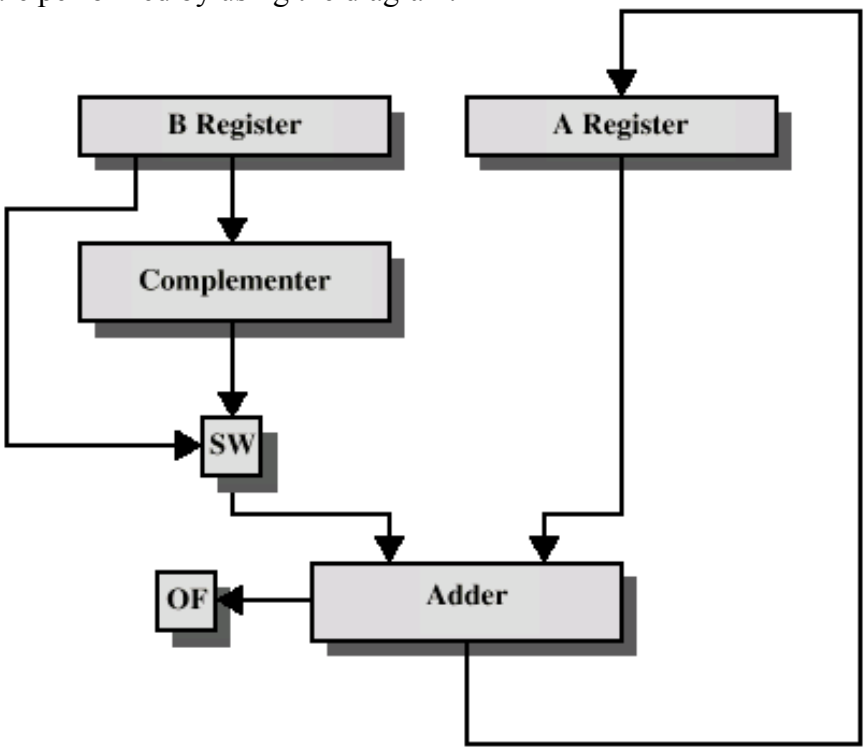
STACK AT END OF PROGRAM	
00C6	
00C8	
00CA	
00CC	
00CE	
00E0	
00E2	
00E4	
00E6	
00E8	
00EA	
00EC	
00FE	
00F0	
00F2	
00F4	
00F6	
00F8	
00FA	0000
00FC	0000
00FE	0000

REGISTERS

AX	0006
CX	0000
SI	0011
BP	0000

Question 3 [4 Marks]

The figure shows a block diagram for addition and subtraction hardware. Explain how the following operations are performed by using the diagram.



OF = overflow bit
SW = Switch (select addition or subtraction)

(A)+(B)	**A and B are directly moved to the Adder for addition.**
(A)-(B)	**B passes through the complementer and it becomes -B, then A and -B are added.**
(-A)+(-B)	Both are added
(A)-(-B)	Value in B passes through the complementer and then added to A.
(-A)-(B)	Value in B passes through the complementer and then added to A.
(-A)-(-B)	Value in B passes through the complementer and then added to A.

Question 4 [8 Marks]

Use Booth's algorithm to multiply -3 (take it as Q) by 10 (take it as M). Show all steps. Consider your computer's data width is 8-bits.

M (MULTIPLICAND)

0000 1010

A (ACCUMULATOR)	Q (MULTIPLIER)	Q_{-1}	
1111 0110	1111 1101	0	**Add**
1111 1011	0111 1110	1	**Shift**
0000 0101	0111 1110	1	**Add**
0000 0010	1011 1111	0	**Shift**
1111 1000	1011 1111	0	**Add**
1111 1100	0101 1111	1	**Shift**
1111 1100	0101 1111	1	**Add**
1111 1110	0010 1111	1	**Shift**
1111 1110	0010 1111	1	**Add**
1111 1111	0001 0111	1	**Shift**
1111 1111	0001 0111	1	**Add**
1111 1111	1000 1011	1	**Shift**
1111 1111	1000 1011	1	**Add**
1111 1111	1100 0101	1	**Shift**
1111 1111	1100 0101	1	**Add**
1111 1111	1110 0010	1	**Shift**
			Add
			Shift

Question 5 [8 Marks]

Find the value of AL (decimal) for the following set of instructions.

`mov ax, -80` `shr ax, 1`	**AL= D8**	`mov ax, -80` `shr al, 1`	**AL= 58**
`mov al, -80` `shr ax, 1`	**AL= 58**	`mov al, -80` `shr al, 1`	**AL= 58**
`mov ax, -80` `sar ax, 1`	**AL= D8**	`mov ax, -80` `sar al, 1`	**AL= D8**
`mov al, -80` `sar ax, 1`	**AL= 58**	`mov al, -80` `sar al, 1`	**AL= D8**

Question 6 [5 Marks]

SHL instruction performs unsigned multiplication when the multiplier is a power of 2 and any other number can be expressed in powers of 2. Write the instructions to find the product of AL into -9, where AL=14. A sample is provided below.

Sample Program: AX x 36

```
mov ax, 123
mov bx, ax
shl ax, 5
shl bx, 2
add ax, bx
```

Problem: AL=14, AL * (-9)

```
.model small
.stack
.data

.code
mov al,14
mov bl,al
mov cl,3
shl al,cl          ;; 14 * 2^3 = 112
add al,bl          ;; 112 + 14 = 126

add al,1           ;; 126 + 1 = 127
mov bl,1
sub bl,al          ;; 1 – 127 = -126
mov al,bl

mov ah,04ch
int 21h
end
```

Question 7 [8 Marks]

Write a code for extended SHIFT LEFT operation of 64 bit number ***num1***. Perform this shift operation for **num2** times. Note that your code should work for any values of num1 and num2, not just for the values given below.

```
.model small
.stack 0100h
.data
; data is defined here
num1 dq 01020304h   ;dq means define Quadword, it allocate 8 bytes
num2 word 0002h     ;dw means define Word, it allocate 2 bytes
.code
Start:

mov ax,@data
mov ds, ax


mov cx, [num2]

L1:

     mov ax,  word ptr  [num1]
     shl ax,1
     mov word ptr  [num1], ax


     mov ax,  word ptr  [num1 + 2]
     rcl ax,1
     mov word ptr  [num1 + 2], ax


     mov ax,  word ptr  [num1 + 4]
     rcl ax,1
     mov word ptr  [num1 + 4], ax


     mov ax,  word ptr  [num1 + 6]
     rcl ax,1
     mov word ptr  [num1 + 6], ax

Loop L1

mov ah,04ch
int 21h
```

ROUGH SPACE