National University of Computer and Emerging Sciences

Laboratory Manual

for

Computer Organization and Assembly Language Programming

Course Instructor	Aleena Ahmad
Lab Instructor	Sana Ejaz
Semester	Fall 2024

Department of Computer Science

FAST-NU, Lahore, Pakistan

OBJECTIVES:

- How to perform bit operations.
- How to swap alternate bits.
- How to create a basic stack function.

Instructions:

1. Run and debug the programs, ensuring that they behave as expected.
2. Document your observations and note any issues encountered during implementation in a Word document.
3. Submit work in a single Word file with screenshots. No asm, lst , or com.

(Do not submit a zip folder)

Task 1: Write a program to swap the nibbles (4-bits = 1 nibble) in each byte of the AX register.

Sample:

AX before Swap	**1011 0010 0101 1101**	**0xB25D**
AX after Swap	**0010 1011 1101 0101**	**0x2BD5**

Task 2: Write a program to swap every pair of bits in the AX register i.e. swap bit # 0 with bit # 1, bit # 2 with bit # 3 and so on.

Sample:

AX before Swap	**10 11 00 10 01 01 11 01**
AX after Swap	**01 11 00 01 10 10 11 10**

Task 3: AX contains a non-zero number. Count the number of ones in it and store the result back in AX. Repeat the process on the result (AX) until AX contains one. Calculate in BX the number of iterations it took to make AX one. For example BX should contain 2 in the following case:

AX = 1100 0101 1010 0011 (input – 8 ones)
AX = 0000 0000 0000 1000 (after first iteration – 1 one)
AX = 0000 0000 0000 0001 (after second iteration – 1 one) STOP

Homework (Submit before 12:00a.m)
Bonus 2 marks if done in class after completing tasks 1-3

Task 4: Write a program that defines a basic stack function to calculate the sum of two numbers. The function should accept two integers as parameters, passed through the stack OR use hardcoded values in registers. The function should return the sum in AX. Use stack pointer registers, subroutines and push/pop to perform addition.

Practice (submission not required)

Exercise 1: For the code segment given below, write the contents of each register and memory after the execution.

a) Note: Label a is ds:103.

```
; Multiplication with multiplier in memory

[org 0x0100]

                    jmp start

a:        db 13
b:        db 5
res:      db 0

start:              mov cl,4
                    mov al, [a]

loop:               shr byte[b], 1
                    jnc skipAdd

                    add [res], al

skipAdd:            shl al,1
                    dec cl
                    jnz loop


mov ax, 0x4c00            ;terminate the program
int 0x21
```

(Verify your answer with debugger.)

b) Above code runs the loop 4 times even if b=0. Update above code such that it breaks the loop as soon as there is no 1 bit left in b. (Verify your answer with debugger.)

c)

```
 [org 0x0100]

                    jmp start

a:         dw 0x04E9

start:     mov ax, [a]
           mov bh, [a]
           mov dl, [a]
           shl ax, 1
           shl ax, 1
           shl ax, 1
           shl ax, 1
           shl word[a], 1
           shl word[a], 1
           shl word[a], 1
           shl word[a], 1
           rol bx, 1
           rol bx, 1
           rol bx, 1
           rol bx, 1
           rcl dl, 1
           rcl dl, 1
           rcl dl, 1
           rcl dl, 1
           rcl byte[a+1], 1
           rcl byte[a+1], 1
           rcl byte[a+1], 1
           rcl byte[a+1], 1

mov ax, 0x4c00             ;terminate the program
int 0x21
```

d)

[org 0x0100]	**BX**
mov bx, 0xFCA9	
ror bl,4	
ror bh,4	
mov ax, 0x4c00 ;terminate the program int 0x21	

e)

Code	AX	BX	DX
[org 0x0100]			
mov ax, 0xB25D			
mov bx,0x5555			
and bx, ax			
mov dx,0xAAAA			
and dx, ax			
shr dx, 1 shl bx, 1 add bx, dx mov ax, bx mov ax, 0x4c00 ;terminate the program int 0x21			