National University of Computer and Emerging Sciences

Laboratory Manual

for

Computer Organization and Assembly Language

Course Instructors

Lab Instructor(s)

Section

Semester

Department of Computer Science

COAL Lab 12 Manual

Objectives:

- Revision
- Extended Addition & Subtraction using ADC, SBB
- Problems & Assignments

12.1 Extended Addition Subtraction

The ADC (add with carry) instruction adds both a source operand and the contents of the Carry flag to a destination operand.

Example1: Adding two 8-bit integers (FFh + FFh), producing a 16-bit sum in DL:AL, which is 01FEh.

```
mov DL,0
mov AL,0FFh
add AL,0FFh ; AL = FEh
adc DL,0 ; DL/AL = 01FEh
```

Example2: Adding two 32-bit integers (FFFFFFFFh + FFFFFFFFh), producing a 64-bit sum in EDX:EAX: 00000001FFFFFFFEh:

```
mov EDX,0
mov EAX,0FFFFFFFFh
add EAX,0FFFFFFFFh
adc EDX,0
```

The SBB (subtract with borrow) instruction subtracts both a source operand and the value of the Carry flag from a destination operand.

64-bit subtraction

```
mov EDX,7 ; upper half
mov EAX,1 ; lower half
sub EAX,2 ; subtract 2
sbb EDX,0 ; subtract upper half
```

Problem(s) / Assignment(s)

Discussion & Practice **Estimated completion time: 1 hr, 30 mins**

PROBLEM 12.1: *Extended Addition* **Estimated completion time:30mins**

Write a program that have following procedures,

`Extended_Add`, calculates the sum of two extended integers stored as arrays of bytes.

`Display_Sum`, displays the sum in its proper order, starting with the high-order byte, and working its way down to the low-order byte

Use the arrays,

```
op1 BYTE 34h,12h,98h,74h,06h,0A4h,0B2h,0A2h
op2 BYTE 02h,45h,23h,00h,00h,87h,10h,80h
```

Sum comes out to be,

```
0122C32B0674BB5736
```

PROBLEM 12.2: *Encryption using rotate operation* **Estimated completion time:30mins**

Write a program that performs simple encryption by rotating each plaintext byte a varying number of positions in different directions.

For example, in the following array that represents the encryption key, a negative value indicates a rotation to the left and a positive value indicates a rotation to the right. The integer in each position indicates the magnitude of the rotation:

```
key BYTE -2, 4, 1, 0, -3, 5, 2, -4, -4, 6
```

Your program should loop through a plaintext message and align the key to the first 10 bytes of the message. Rotate each plaintext byte by the amount indicated by its matching key array value. Then, align the key to the next 10 bytes of the message and repeat the process.

PROBLEM 12.3: *Finding relative prime numbers* **Estimated completion**

Harry wants to check the relative primality of 2 numbers. For this purpose, he checks the GCD (Greatest Common Divisor) of the numbers. If GCD comes out 1 then numbers are relative prime to each other. Harry requirements are as follows:

time:30mins

1. Procedure **DEC_IN** should load two registers (BX and DX) with two numbers. Numbers should be a 2 - digit decimal ranging from (01 - 99).
2. Procedure **GCD_AB** apply the logic for GCD of two numbers. GCD of two numbers is performed by dividing the greater number (in BX) by the smaller number (in DX) till the remainder is zero. If it is zero, the divisor is the GCD if not the remainder and the divisor of the previous division are the new set of two numbers. The process is repeated by dividing greater of the two numbers by the smaller number till the remainder is zero and GCD is found.
3. Also check if,
    a. The numbers are equal then GCD would be BX,
    b. BX<DX then exchange the contents of 2 register.
4. Procedure **DEC_OUT** should display the GCD on screen in decimal.

Sample 1:

```
Enter 1st Number: 20
Enter 2nd Number: 09
GCD is: 1
Numbers are relative prime
```

Sample 2:

```
Enter 1st Number: 09
Enter 2nd Number: 03
GCD is: 03
Numbers are not relative prime
```