Course Title: Microprocessors and Assembly Language Lab (CSE-4504)

Department of Computer Science and Engineering (CSE)
Islamic University of Technology (IUT), Gazipur

Lab # 05

Bit Manipulation Instructions and Stack Operations in Assembly Language Program.

Objective:

To understand some 8086 instructions related to bit manipulation and getting familiar with the use of 8086 stack operations using Assembly Language Program.

Theory:

- **Bit Manipulation Instructions**
  - **Logical Instructions**

    NOT: Invert each bit of a byte or word.

    NOT AX

    AND: And-ing each bit in a byte or word with the corresponding bit in another byte or word

    AND AX, F08AH

    OR: OR-ing each bit in a byte or word with the corresponding bit in another byte or word.

    OR AX, FFABH

    XOR: XOR-ing each bit in a byte or word with the corresponding bit in another byte or word.

    X**OR AX, AX** ; Resets the contents of AX register
  - **Shift Instructions**

    SHL: Shifting bits of word or byte left putting zero(s) in LSB(s).

    SHL AL, 1

    If you SHL (AL = 00000001) 1 times, you get a value of 2 in AL or 00000010. Do it again and your bit(s) shift up by 1 more. SHL is a quick way to multiply a value by 2,4,8, etc because every time you SHL you double the value.

    SHL AL, CL ; Takes the value in CL and shifts AL that many times.

    SHR: Shifting of word or byte right putting zero(s) in MSB(s). Shift right will half your original value, making quick division a snap.

    SHR AL, 1 **SHR AL, CL**

    ROL: It rotates bits of byte or word left, by count. MSB is transferred to LSB.

    ROL AL, 1 **ROL AL, CL**

    ROR: It rotates bits of byte or word left, by count. LSB is transferred to MSB.

    ROR AL, 1 **ROR AL, CL**

- **Stack Operation**

  Stack is a section of memory for storing **return addresses**. It is also used to save the contents of registers for the calling program while a procedure executes. Another use of stack is to hold data or addresses that will be acted upon by a procedure. PUSH and POP instructions operate on stack and can be used for swap operations between registers.

  PUSH and POP: 16-bit registers values can be **pushed** (the SS is first decremented by 2 and then the value is stored at the address in SP) or **popped** (the value is restored from the memory at SS and then SP is incremented by 2). For example:

  PUSH AX ; Push contents of AX register into stack segment SS:SP (top of stack)
  POP BX ; Pop the value from SS:SP (top of stack) into BX register

Assembly Language Program Example for PUSH and POP Instructions:

```
org 100h

.DATA

r1 DW 100
r2 DW 500
r3 DW 600

.CODE

mov dx, r2          ; r2 = dx
add dx, r3          ; r2+r3 = dx
push dx             ; save value of dx in stack
mov cx, r1          ;r1 = dx
push cx
add cx, dx          ;r1 + (r2+r3)

mov bx, cx          ;mov cx having (r1) + (r2+r3) into bx
pop cx              ;pop old cx (r1) back to cx
pop dx              ;pop (r2+r3) back to dx
mov ax, cx          ;mov old cx(r1) to ax

ret
```

Tasks to do:

1. Write an assembly language code that takes N as a decimal digit (0 ~ 9) input and shows its *factorial* as output. [Hint: Use loop or array or both **and** use bit manipulation instructions]

   Sample Input / Output:

   Input the value of N: 5 (*Take the Input from User*)
   The Factorial of N is: 120 (*Store the Output in a Register and Variable*)

2. Write an assembly language code that takes N as a decimal digit (0 ~ 9) input and shows *fibonacci* series up to N-th element of the series. Hint: Use loop or array or both **and** use bit manipulation instructions]

   Sample Input / Output:

   Input the value of N: 7
   Fibonacci Series: 0 1 1 2 3 5 8