Computer Organization and Architecture (EET2211)

LAB VII: Swap the upper nibble of a word with the lower nibble content of an accumulator

Siksha ‘O’ Anusandhan Deemed to be University, Bhubaneswar

Branch: Computer Science and Engineering		**Section:** ‘D’	
S. No.	**Name**	**Registration No.**	**Signature**
52	Saswat Mohanty	1941012407	Saswat Mohanty

Marks: ______/10

Remarks:

Teacher’s Signature

I. OBJECTIVE:

1) Write a program to swap the upper nibble of a word with the lower nibble content of an accumulator.

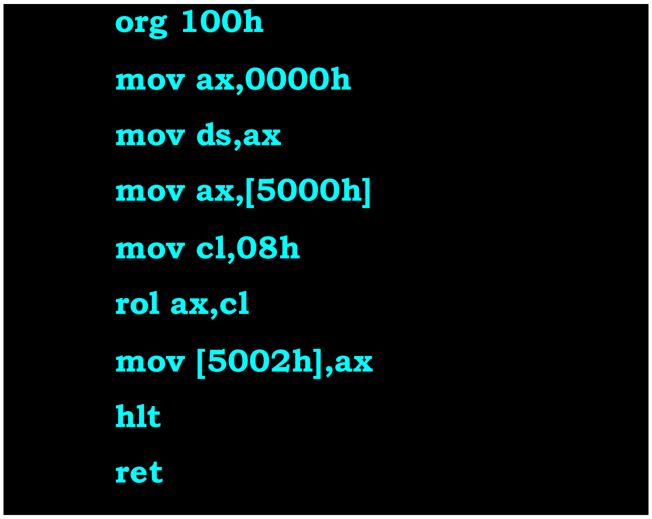
II. PRE-LAB

For Obj. 1:

a) **Swap the upper nibble of a word with the lower nibble content of an accumulator.**

[5000h] = 1234h
Output: 3412h

b) **Write the assembly code.**

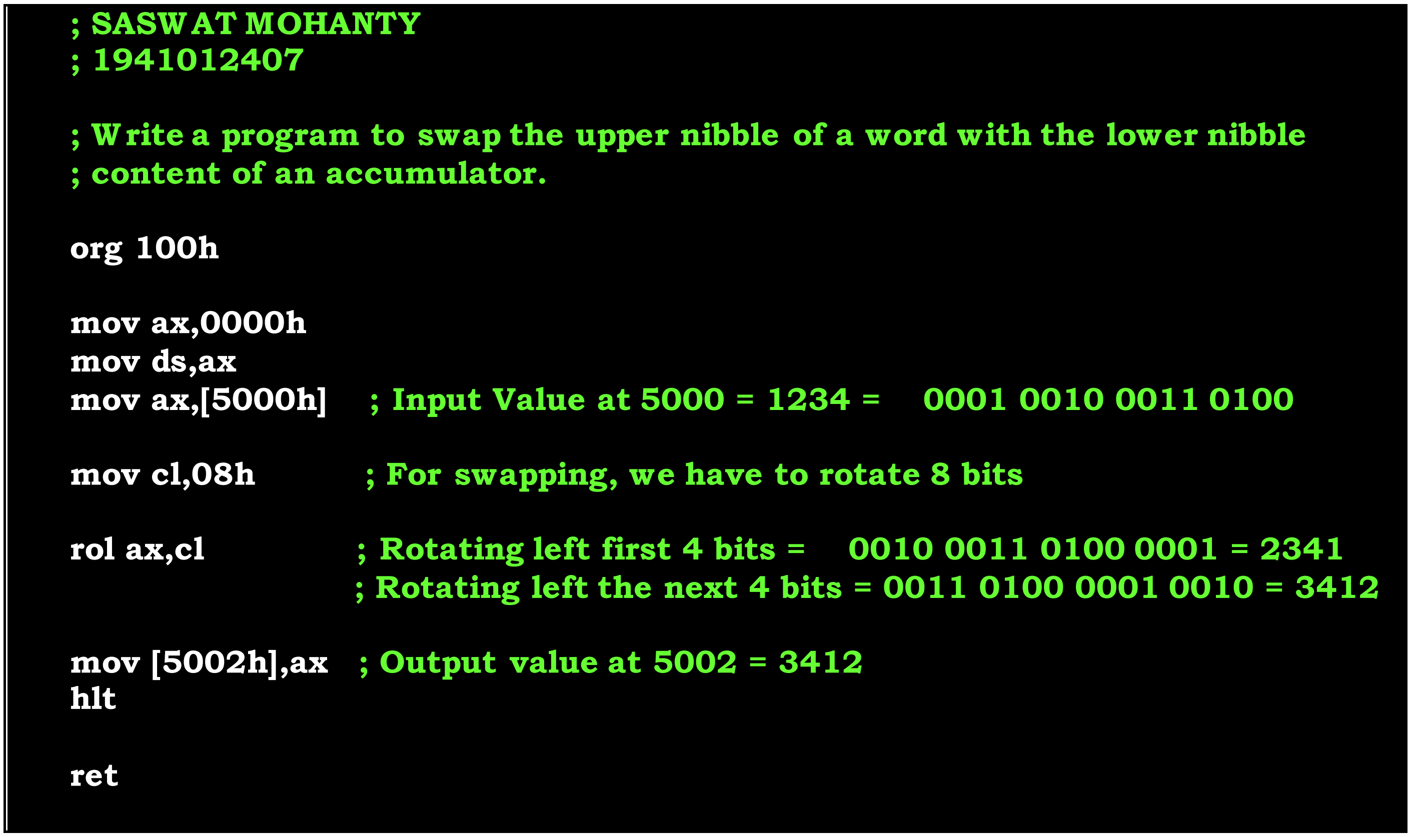


```
org 100h
mov ax,0000h
mov ds,ax
mov ax,[5000h]
mov cl,08h
rol ax,cl
mov [5002h],ax
hlt
ret
```

III. LAB:

Assembly Program:

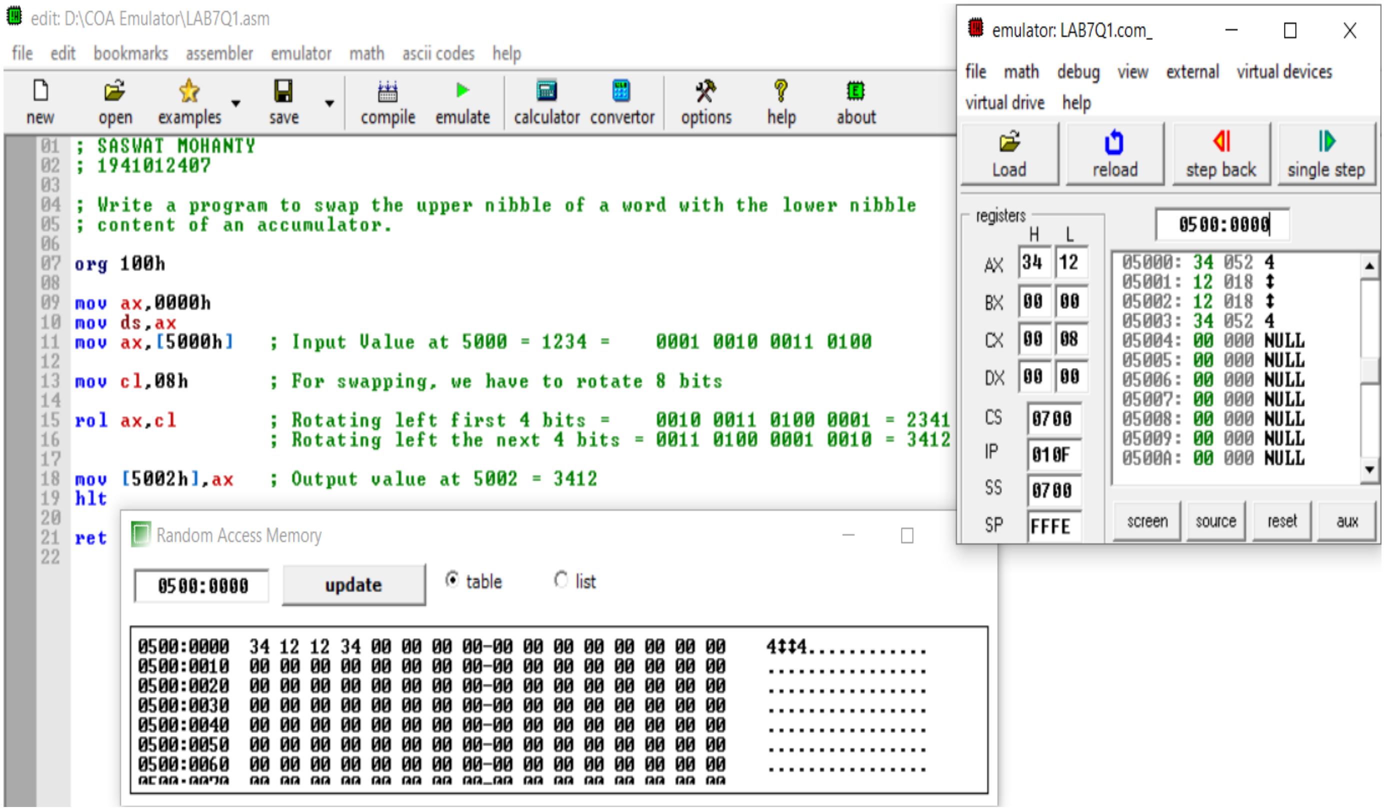
For Obj. 1:



```
; SASWAT MOHANTY
; 1941012407

; Write a program to swap the upper nibble of a word with the lower nibble
; content of an accumulator.

org 100h

mov ax,0000h
mov ds,ax
mov ax,[5000h]    ; Input Value at 5000 = 1234 =    0001 0010 0011 0100

mov cl,08h        ; For swapping, we have to rotate 8 bits

rol ax,cl         ; Rotating left first 4 bits =    0010 0011 0100 0001 = 2341
                  ; Rotating left the next 4 bits = 0011 0100 0001 0010 = 3412

mov [5002h],ax    ; Output value at 5002 = 3412
hlt

ret
```

Observations (with screen shots):

For Obj. 1:



```
; SASWAT MOHANTY
; 1941012407

; Write a program to swap the upper nibble of a word with the lower nibble
; content of an accumulator.

org 100h

mov ax,0000h
mov ds,ax
mov ax,[5000h]    ; Input Value at 5000 = 1234 =     0001 0010 0011 0100

mov cl,08h        ; For swapping, we have to rotate 8 bits

rol ax,cl         ; Rotating left first 4 bits =     0010 0011 0100 0001 = 2341
                  ; Rotating left the next 4 bits = 0011 0100 0001 0010 = 3412

mov [5002h],ax    ; Output value at 5002 = 3412
hlt

ret
```

```
     H   L
AX   34  12
BX   00  00
CX   00  08
DX   00  00
CS   0700
IP   010F
SS   0700
SP   FFFE
```

```
0500:0000
05000: 34 052 4
05001: 12 018 ↕
05002: 12 018 ↕
05003: 34 052 4
05004: 00 000 NULL
05005: 00 000 NULL
05006: 00 000 NULL
05007: 00 000 NULL
05008: 00 000 NULL
05009: 00 000 NULL
0500A: 00 000 NULL
```

0500:0000 ● table ○ list

```
0500:0000  34 12 12 34 00 00 00 00-00 00 00 00 00 00 00 00    4↕↕4............
0500:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
0500:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
0500:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
0500:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
0500:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
0500:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
0500:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
```

Conclusion:

It can be concluded that swap the upper nibble of a word with the lower nibble content of an accumulator when dry run and executed in system found to be same. Thus, the program to swap the nibbles was executed.

IV. POST LAB:

1. Explain briefly the advantages of memory segmentation in 8086.

Advantages of memory segmentation in 8086:-

- It allows to processes to easily share data.

- It allows extending the address ability of the processor, i.e. segmentation allows the use of 16 bit registers to give an addressing capability of 1 Megabytes. Without segmentation, it would require 20 bit registers.

2. Explain the IAS instruction format.

The IAS machine was a binary computer with a 40-bit word, storing two 20-bit instructions in each word. The memory was 1,024 words (5.1 kilobytes). Negative numbers were represented in two's complement format. It had two general-purpose registers available: the Accumulator (AC) and Multiplier/Quotient (MQ).

3. Briefly explain the following flags of 8086:

a) Carry Flag (CF) **b) Parity Flag (PF)** **c) Adjust Flag (AF)**

d) Zero Flag (ZF) **e) Sign Flag (SF)** **f) Overflow Flag (OF)**

a) **Carry Flag (CF): -** Holds the carry after addition or borrow after subtraction. Also indicates some error conditions as dictated by some programs and procedures.

b) **Parity Flag (PF): -** PF=0= odd parity; PF=1=even parity

c) **Adjust Flag (AF): -** Holds the carry (half carry) after addition or borrow after subtraction between bit positions 3 and 4 of the result (e.g. in BCD addition or subtraction)

d) **Zero Flag (ZF): -** Shows the result of the arithmetic or logic operation.

e) **Sign Flag (SF): -** Holds the sign of the result after an arithmetic/logic instruction execution.

f) **Overflow Flag (OF): -** Overflow occurs when signed numbers are added or subtracted. An overflow indicates the result has exceeded the capacity of the machine.