# Tutorial Exercises

## Exercise 1

1. What will be the value of BX after the following instructions are carried out?

mov bx,029D6h

xor bx,8181h

1. What will be the value of EBX after the following instructions are carried out?

mov ebx,0AFAF649Bh

or ebx,3A219604h

1. In the following instruction sequence, show the resulting value of AL where indicated:

mov al, 01101111b

and al, 00101101b ; a.

mov al, 6Dh

and al, 4Ah ; b.

mov al, 00001111b

or al, 61h ; c.

mov al, 94h

xor al, 37h ; d.

## Exercise 2

Implement the following pseudocode in assembly language. Assume that val1, val2, val3 and X are 32-bit variables.

if ( val1 > val2 ) AND ( val2 > val3 )

X = val2

else

X = val3;

## Exercise 3

The objective of this example is to show how multiplication can be done entirely by shift and add operations. We consider multiplication of two unsigned 8-bit numbers.

In order to use the shift operation, we have to express the multiplier as a power of 2. For example, if the multiplier is 64, the result can be obtained by shifting the multiplicand left by six bit positions (because 26 = 64).

If the multiplier is not a power of 2, we have to express this number as a sum of powers of 2. For example, if the multiplier is 10, it can be expressed as 8+2, where each term is a power of 2. Then the required multiplication can be done by two shifts and one addition.

The question now is: How do we express the multiplier in this form? For example, if we look at the binary representation of the multiplicand (100 = 0000**1**0**1**0B), there is a 1 in bit positions with weights 8 (=2**3**) and 2 (=2**1**). Thus, for each 1 bit in the multiplier, the multiplicand should be shifted left by a number of positions equal to the bit position number. In the above example, the multiplicand should be shifted left by 3 and 1 bit positions and then added. This procedure is formalized in the following algorithm.

**Algorithm**: multiplication using shifts and adds

**Input:** two short numbers number1 and number2

**Output:** result = numbr1 \* number2

result := 0

for (i = 7 downto 0)

if ( bit(number2 , i) = 1) // bit(number2, i) is the ith bit of number2

result := result + number1\*2i

endif

endfor