**Lab 3 –**

**Problem example:**

; **Given the words A and B**, **compute the word C** as follows:

;- the bits **0-2 of C** are the same as the bits **10-12 of B**

**- the bits 2-0 of C are the same as the bits 12-10 of B**

;- the bits **3-6 of C** have the **value 1**

**- the bits 6-3 of C have the value 1**

;- the bits **7-10 of C** are the same as the bits **1-4 of A**

;- the bits 11-12 have the **value 0**

;- the bits 13-15 of C are the invert of the bits 9-11 of B

C - dw 0

**15 - 13 12 11 10 - 7 6 5 4 3**   **2 - 0**

**C** = **invB(11-9)**; **0 0**; **A(4-1);** **1 1 1 1** ; **B(12-10)**

**fortare biti in 1: OR 0 (nu ii doresc a fi pastrati), 1 (ii doresc)**

**operatia 2: OR cu MASCA: 0000 0000 0111 1000b**

**operatia 4: AND cu MASCA: 1110 0111 1111 1111b**

**operatia 5: XOR cu MASCA: 0000 1110 0000 0000b**

**var B si masca: => 1100 1101 0101 1100b**

**X**

**Inserarea in rez intermediar : OR**

**invB si C:**

; We will obtain the word C by successive "**isolation**" of bits sequences. The isolation of the bits 10-12 of B is done by obtaining the unchanged values of these bits, and initialising all other bits to 0. The isolation operation can be performed using the operator **AND** between the word B and the mask 000**111**0000000000. Once isolated, the sequence of bits is put on the right position by using a rotation operation.

; **The final word** is obtained by applying the operator **OR** between all intermediate results obtained by using isolations and rotations.

; Observation: bits are numbered from right to left

**bits 32** ;assembling for the 32 bits architecture

; the start label will be the entry point in the program

**global** **start**

**extern** **exit** ; we inform the assembler that the exit symbol is foreign, i.e. it exists even if we won't be defining it

**import** **exit msvcrt.dll**; exit is a function that ends the process, it is defined in msvcrt.dll

; msvcrt.dll contains exit, printf and all the other important C-runtime functions

**segment** **data use32 class=data** ; the data segment where the variables are declared

**15 12 | 11 8 | 7 4 | 3 0**

**a dw 0111011101010111b ; a = 0111 0111 0101 0111b**

**b dw 1001101110111110b ; b = 1001 1011 1011 1110b**

c **dw** 0

**segment** **code use32 class=code** ; code segment

**start:**

*;initialisation*

**mov** **bx, 0** ; we compute the result in bx

***; operatia 1: b2-0 = B(12-10)***

**mov** **ax, [b]** ; we isolate bits 10-12 of B

**and** **ax, 0001110000000000b ; AX = 0001 1000 0000 0000b**

**mov cl, 10 ; diferenta intre b12 si b2**

**ror ax, cl** ; we rotate 10 positions to the right

**;AX = 0000 0000 0000 0110b = 0006h**

**or bx, ax** ; we put the bits into the result

; BX=0000 0000 0000 0**110b = 0006h**

***; operatia 2: b6-3 = 1111***

**or bx, 0000000001111000b** ; we force the value of bits 3-6 of the result to the value 1

; BX=0000 0000 0**111 1110b = 007Eh**

***; operatia 3: b10-7 = A(4-1)***

**mov ax, [a]** ; we isolate bits 1-4 of A

**and ax, 0000000000011110b ; ;AX = 0000000000010110b**

**mov cl, 6 ; din diferenta b7-b1**

**rol ax, cl** ; we rotate 6 positions to the left

**;AX = 0000 0101 1000 0000b=0580h**

**or bx, ax** ; punem bitii in rezultat

; BX=0000 0**101 1111 1110b = 05FEh**

***; operatia 4: b12-11 = 00***

**;and bx, 1110011111111111b** ; facem biti 11-12 din rezultat sa aiba valoarea 0

; BX=000**0 0101 1111 1110b = 05FEh**

***; operatia 5: b15-13= invB(11-9)***

**mov ax, [b]**

**not ax** ; we invert the value of b

**and ax, 0000111000000000b** ; we isolate the bits 9-11 of B

; AX=0000 **010**0 0000 0000b=0400h

**mov cl, 4 ; b15-b11**

**rol ax, cl** ; we rotate 4 positions to the left

; AX= **010**0 0000 0000 0000b=4000h

**or bx, ax** ; punem bitii in rezultat

; BX=**0100 0101 1111 1110b = 45FEh**

**mov [c], bx** ; we move the result from the register to the result variable

**push dword 0** ;saves on stack the parameter of the function exit

**call [exit]** ;function exit is called in order to end the execution of the program

***;operatia 5: b15-13= invB(11-9) cu XOR***

**mov ax, [b]**

**xor AX, 0000111000000000b ; AX=1001010110111110**

**and AX, 0000111000000000b ; AX=00000100000000000b**

**mov CL, 4**

**shl AX, CL**

or BX, AX ; BX=**0100 0101 1111 1110b = 45 FEh**

**Homeworks:**

1 Given the words A and B, compute the doubleword C as follows:

* the bits 0-4 of C are the same as the bits 11-15 of A
* the bits 5-11 of C have the value 1
* the bits 12-15 of C are the same as the bits 8-11 of B
* the bits 16-31 of C are the same as the bits of A

2. Given the words A and B, compute the doubleword C as follows:

* the bits 0-3 of C are the same as the bits 5-8 of B
* the bits 4-8 of C are the same as the bits 0-4 of A
* the bits 9-15 of C are the same as the bits 6-12 of A
* the bits 16-31 of C are the same as the bits of B

3. Given the words A and B, compute the doubleword C as follows:

* the bits 0-2 of C are the same as the bits 12-14 of A
* the bits 3-8 of C are the same as the bits 0-5 of B
* the bits 9-15 of C are the same as the bits 3-9 of A
* the bits 16-31 of C are the same as the bits of A

4. Given the bytes A and B, Compute the doubleword C as follows:

* the bits 8-15 of C have the value 0
* the bits 16-23 of C are the same as the bits of B
* the bits 24-31 of C are the same as the bits of A
* the bits 0-7 of C have the value 1

5 Given the bytes A and B, compute the doubleword C as follows:

* the bits 16-31 of C have the value 1
* the bits 0-3 of C are the same as the bits 3-6 of B
* the bits 4-7 of C have the value 0
* the bits 8-10 of C have the value 110
* the bits 11-15 of C are the same as the bits 0-4 of A

6 Given the word A and the word B, compute the doubleword C:

* the bits 8-15 of C have the value 0
* the bits 16-23 of C are the same as the bits of 2-9 of B
* the bits 24-31 of C are the same as the bits of 7-14 of A
* the bits 0-7 of C have the value 1

7 Given the words A and B, compute the doubleword C:

* the bits 0-4 of C have the value 1
* the bits 5-11 of C are the same as the bits 0-6 of A
* the bits 16-31 of C have the value 0000000001100101b
* the bits 12-15 of C are the same as the bits 8-11 of B

8. Compute the doubleword D as follows, b word, a word.

* the bits 8-15 are the same as the bits of C
* the bits 0-7 are the same as the bits 8-15 of B
* the bits 24-31 are the same as the bits 0-7 of A
* the bits 16-23 are the same as the bits 8-15 of A.

9. Given the word A and the byte B, compute the doubleword C as follows:

* the bits 0-3 of C are the same as the bits 6-9 of A
* the bits 4-5 of C have the value 1
* the bits 6-7 of C are the same as the bits 1-2 of B
* the bits 8-23 of C are the same as the bits of A
* the bits 24-31 of C are the same as the bits of B

10. Given the words A and B, compute the doubleword C as follows:

* the bits 0-6 of C have the value 0
* the bits 7-9 of C are the same as the bits 0-2 of A
* the bits 10-15 of C are the same as the bits 8-13 of B
* the bits 16-31 of C have the value 1

11. Given the byte A and the word B, compute the doubleword C as follows:

* the bits 0-7 of C have the value 1
* the bits 8-11 of C are the same as the bits 4-7 of A
* the bits 12-19 are the same as the bits 2-9 of B
* the bits 20-23 are the same as the bits 0-3 of A
* the bits 24-31 are the same as the high byte of B

12. Given the word A and the byte B, compute the doubleword C:

* the bits 0-3 of C have the value 1
* the bits 4-7 of C are the same as the bits 0-3 of A
* the bits 8-13 of C have the value 0
* the bits 14-23 of C are the same as the bits 4-13 of A
* the bits 24-29 of C are the same as the bits 2-7 of B
* the bits 30-31 have the value 1

13. Given the word A, compute the doubleword B as follows:

* the bits 28-31 of B have the value 1;
* the bits 24- 25 and 26-27 of B are the same as the bits 8-9 of A
* the bits 20-23 of B are the invert of the bits 0-3 of A ;
* the bits 16-19 of B have the value 0
* the bits 0-15 of B are the same as the bits 16-31 of B.

14. Given the words A and B, compute the doubleword C as follows:

* the bits 0-5 of C are the same as the bits 3-8 of A
* the bits 6-8 of C are the same as the bits 2-4 of B
* the bits 9-15 of C are the same as the bits 6-12 of A
* the bits 16-31 of C have the value 0