~~MUL [a]~~

MUL byte [a]

~~MUL 3~~

~~MUL byte 3~~

~~MUL byte [3] ; illegal memory access~~

MOV BL, 3

MUL BL

MUL BL ; AL \* BL -> AX

MOV AX, FFFFh

MOV BL, 1

DIV BL ; AL=AX/BL = FFFFh / 1 = FFFFh

AL=FFFFh – division by zero (overflow)

IDIV BL ; AL=AX/BL = (-1) / 1 = -1

;AL = FFh

;AX -> DX:AX

MOV DX, 0

MOV BH, 0 ;BL -> BX

DIV BX ; DX:AX / BX -> AX = FFFFh

AX -> EAX vorzeichenlos

MOV DX, 0

Push DX

Push AX

Pop EAX

MOV BX, AX

MOV EAX, 0

MOV AX, BX

1. EDX:EAX + ECX:EBX

Add EAX, EBX

Adc EDX, ECX

1. DX:AX – BX

Vorzeichenlos:

Sub AX, BX

Sbb DX, 0 ;conversie fara semn

Vorzeichenbehaftet:

Mov CX, AX ; DX:CX – BX

Mov ax, bx ; DX:CX – AX

Mov BX, dx ;BX:CX – AX

Cwd ; AX -> DX:AX

; BX:CX – DX:AX

Sub CX, AX

Sbb BX, DX

1. (a\*b)/c a – byte, b – doubleword, c- word

**Vorzeichenlos:**

Mov AL, [a]

Mov AH, 0

Mov DX, 0; DX:AX = [a]

Push DX

Push AX

Pop EAX

Mul dword[b]; EDX:EAX = a\*b

Mov EBX, 0;

Mov BX, [c]; EBX = c

Div EBX; EAX = EDX:EAX / EBX = (a\*b)/c

**Vorzeichenbehaftet:**

Mov AL,[a]

Cbw ; AX = a

Cwde ;EAX =a

Mov EBX, [b]; EBX = b

Imul EBX; EDX:EAX = a\*b

Mov EBX, EAX;

Mov AX, [c]

Cwde; EAX = [c]

; EDX:EBX / EAX

Mov ECX, EAX ; ECX = [c]

Mov EAX, EBX ;

Idiv ECX; EDX:EAX / ECX > EAX = (a\*b)/c

-------------------------------------------------------------------------------------------------------------------------------------

Gegeben seien **die Wörter A und B**. Berechne das **Wort C**, wobei:

•Die Bits 0-4 aus C sind gleich mit den Bits 5-9 aus A

•Die Bits 5-8 aus C sind gleich mit den Bits **2-5** aus B

•Die Bits 9-10 aus C haben den Wert 1

•Die Bits 11-12 aus C haben den Wert 0

•Die Bits 13-15 aus C sind komplementär zu den Bits 1-3 aus A

**b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0**

Mov bx, 0 ;formam C

Mov ax, [a]

AND AX, 0000...01111100000b

MOV CL, 5

SHR ax, CL ; shiftam la dreapta pe ax cu 5 pozitii 0 0 0 0 0 **0...0 b9 b8 b7 b6 b5**

**OR BX, AX**

MOV AX, [b]

AND AX, 0...0111100b ;b2-5

MOV CL, 3

SHL AX, CL

OR BX, AX

OR BX, 0..011000000000b ; Die Bits 9-10 aus C haben den Wert 1

AND BX, 11..10011111111111b ; Die Bits 11-12 aus C haben den Wert 0

Mov AX, [a]

NOT AX ; inversat toti bitii

AND AX, 0..01110b

MOV CL, 12

SHL AX, CL

OR BX, AX ; Die Bits 13-15 aus C sind komplementär zu den Bits 1-3 aus A

00000 1010 OR

1011 00000 OR

11 000000000