Unsigned

MUL ρ
- if ρ byte : AX = ρ · AL
- if ρ word : DX:AX = ρ · AX
- if ρ doubleword: EDX:EAX = ρ · EAX

AX = 0 3 0 E

DX:AX = 0 7 F A : 3 2 E D

DX (high part) AX (low part)

DIV ρ
- ρ byte : AL = AX / ρ , AH = AX % ρ
- ρ word : AX = DX:AX / ρ , DX = DX:AX % ρ
- ρ doubleword : EAX = EDX:EAX / ρ , EDX = EDX:EAX % ρ

Signed:

iMUL ↶

a db -2 ⟶ FEh 11111110
b db -5 ⟶ FBh 11111011

Unsigned ⇒ 254·251 (a MUL b)
Signed ⇒ -2·-5

iDIV ↶

Examples:

((a+b)*c)/d , a,b,c,d bytes unsigned

```
bits 32
global start
extern exit
import exit (msvcrt.dll)
segment data use32 class=data
a db 1
b db 2
c db 3
d db 4
x db 0
```

```
segment code use 32 class = code
start:
mov AL, [a]
add AL, [b]
mul byte [c]  ; AX = AL * c        (a+b)*c
div byte [d]  ; AL = AX / d        AH = AX % d
mov [x], AL
push dword 0
call [exit]
```

Unsigned conversions

```
byte → word
mov AL, [a]
mov AH, 0          } ⇒ AX = [a]

word → doubleword   ; e dw 1
mov AX, [e]
mov DX, 0          } ⇒ DX:AX = [e]

dword → qword       ; f dd 10
mov EAX, [f]
mov EDX, 0         } ⇒ EDX:EAX = [f]
```

Signed conversion

byte → word: cbw ; byte AL → word AX

word → dword: cwd ; AX → DX:AX

cwde ; AX → EAX

dword → qword cdq ; EAX → EDX:EAX

(a - b * c) / d

a b c,d bytes, unsigned

```
bits 32
global start
extern exit
import exit msvcrt.dll
segment data use 32 class=data
    a db 60
    b db 5
    c db 10
    d db 2
    x db 0
segment use 32 class=code
    start:
```

```
mov  AL , [a]
mov  AH , 0     ; AX= a
mov  BX , AX    ; BX = a
mov  AL ,[b]
mul  byte [c]   ; AX = b*c
sub  BX , AX    ;  BX = (a-b*c)
mov  AX, BX  ;  AX= BX
div  byte [d]   ;  AL= AX / d , AH= AX % d
mov [x] , AL
push  dword 0
call [exit]
```

$(a * b) / d - c$, b-word , a,c,d - byte

signed interpretation

```
bits 32
global start
external exit
import exit msvcrt.dll
segment data use 32 class=data
    a db 1
    b dw -2
    c db 3
    d db 4
    x dw 0
segment use 32 class=code
start:
    mov AL, [a]
    cbw  ; AX = [a]
    imul word [b] ; DX:AX = a*b
    mov CX, AX
    mov AL, [d]
    cbw  ; AX = [d]
    mov BX, AX  ; BX = [d]
```

```
mov AX, CX   ;  DX:AX = (a*b)
idiv BX      ;  DX:AX : (a*b)/d
mov BX, AX   ;  BX = (a*b)/d
mov AL, [C]
cbw  ; AX=[C]
Sub BX, AX
mov [X], BX
push dword 0
call [exit]
```

a*b*c - d

a,b,c,d - byte , signed interpretation

```
segment data used class=data
 a db 1
 b db -2
 c db 3
 d db 4
segment used class = code
 start:
 mov AL, [a]
 imul byte [b] ; AX = a*b
```

```
mov BX , AX      BX = a*b
mov AL , [c]
cbw    ; AX = [c]
imul BX  ; DX:AX = a*b*c
push DX
push AX
pop EBX  ; EBX = a*b*c
mov AL, [d]
cbw   ; AX = d
cwde  ;  EAX = d
sub EBX, EAX  ; EBX = a*b*c - d
```