Multiplication and Division instructions for unsigned representation

Unsigned representation – contains natural numbers

Storage Type	Range (Low to High)	Powers of 2
Unsigned byte	0 to 255	0 to $(2^8 - 1)$
Unsigned word	0 to 65,535	0 to $(2^{16} - 1)$
Unsigned doubleword	0 to 4,294,967,295	0 to $(2^{32} - 1)$
Unsigned quadword	0 to 18,446,744,073,709,551,615	0 to $(2^{64} - 1)$

Multiplication Instruction
(for unsigned representation)

Multiplicand	Multiplier	Product
AL	reg/mem8	AX
AX	reg/mem16	DX:AX
EAX	reg/mem32	EDX:EAX

Syntax: **MUL op**

op is called explicit operand

The MUL is realized different accordind to the explicit operand:

op is *reg/mem8* => **MUL *reg/mem8*** => *AL * reg/mem8 = AX*

op is *reg/mem16* => **MUL *reg/mem16*** => *AX * reg/mem16 = DX:AX*

op is *reg/mem32* => **MUL *reg/mem32*** => *EAX * reg/mem32 = EDX:EAX*

Examples:
op is reg/mem8 => MUL reg/mem8 => AL * reg/mem8 = AX

*Eg 1: 3*4*

```
mov al, 3
mov bl, 4
mul bl      ;al*bl = ax
```

*Eg 2: c*4, c byte*

```
mov al, 4
mul byte [c]    ;al*[c]=ax
   or
mov al, [c]
mov bl, 4
mul bl;     al*bl=ax
```

*Eg. 3 a*b, a, b bytes*

```
mov al, [a]
mul byte[b]    ; al*[b] = ax
   or
mov al, [a]
mov bl, [b]
mul bl    ; al*bl = ax
```

Examples:
op is reg/mem16 => MUL reg/mem16 => AX * reg/mem16 = DX:AX

*Eg 1: 3*5*

```
mov ax, 3
mov bx, 5
mul bx    ;ax*bx = dx:ax
```

*Eg 2: c*4, c word*

```
mov ax, 4
mul word [c]    ;ax*[c]=dx:ax
```

or

```
mov ax, [c]
mov bx, 4
mul bx;     ax*bx=dx:ax
```

*Eg. 3: m*n, m, n word*

```
mov ax,  [m]
mul word[n] ; [m]*[n]=dx:ax
```

*Eg. 4: p*r, p-byte, r-word*

```
mov ax,0
mov al, [p]
mul word[r] ; [p]*[r]=dx:ax
```

Sau

```
Movzx AX, [p]
Mult word[r]  ; dx:ax = p*r
```

Examples:
op is reg/mem32 => MUL reg/mem32 => EAX * reg/mem32 = EDX:EAX

*Eg. 1: e*f, e, f doubleword*

```
mov eax, [e]
mul dword[f]  ; edx:eax=e*f
```

*Eg. 2: g*h, g-byte, h-doubleword*

```
mov eax, 0
mov al, [g]
mul dword[h]    ;edx:eax=rez g*h
```

Sau

```
Movzx eax, [g]
Mul dword[h]  ; edx:eax=rez g*h
```

*Eg. 3: 2*f, e, f doubleword*

```
mov eax, 2
mul dword[f]  ; edx:eax=2*f
```

Nu funct:

```
Mov eax, [f]
Mul 2  ; op explicit nu poate constanta!
```

*Eg. 4: i*j, i-word, j-doubleword*

```
mov eax, 0
mov ax, [i]
mul dword[j]    ;edx:eax=i*j
```

sau

```
Movzx eax, [i]
mul dword[j]    ;edx:eax=i*j
```