Multification & Division

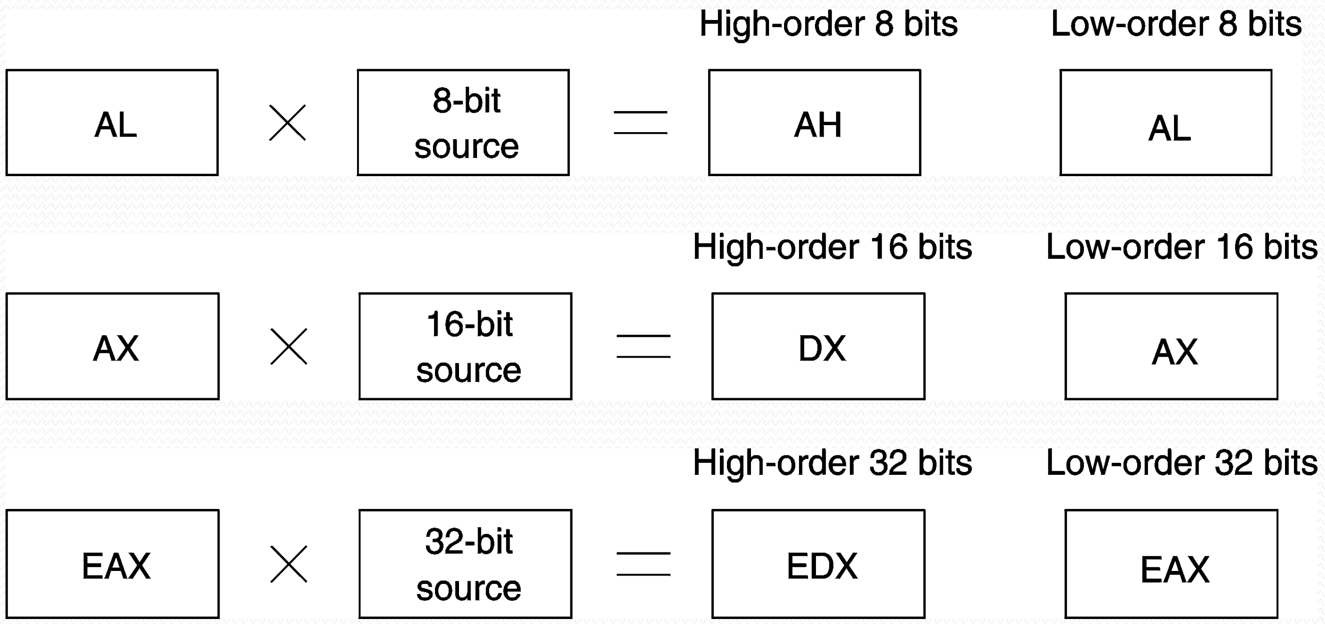
Reference: Assembly Language Programming and Organization of the IBM PC – Charles Marut – Chapter 9

MUL Instructions

- Unsigned multiplication

  `mul      source`

  - Depending on the **`source`** operand size, the location of the other source operand and destination are selected.



High-order 8 bits Low-order 8 bits

AL × 8-bit source = AH AL

High-order 16 bits Low-order 16 bits

AX × 16-bit source = DX AX

High-order 32 bits Low-order 32 bits

EAX × 32-bit source = EDX EAX

MUL Instructions(cont’d)

- Note that the product is stored in a register (or group of registers) twice the size of the operands.
- The operand can be a register or a memory operand

MUL Instructions(cont’d)

Multiplicand	**Multiplier**	**Product**
AL	*r/m8*	AX
AX	*r/m16*	DX:AX
EAX	*r/m32*	EDX:EAX

MUL Examples

```
Mov   al, 5h
Mov   bl, 10h
Mul   bl                ; AX = 0050h, CF = 0
```

(no overflow - the Carry flag is 0 because the upper half of AX is zero)

MUL Examples

```
.data
Val1   WORD        2000h
Val2   WORD        0100h
.code
Mov    ax, val1
Mul    val2        ;DX:AX = 00200000h, CF = 1
```

(Carry flag is 1 because DX is not equal to zero)

IMUL Instruction (Signed Multiply)

- Has the same syntax and uses the same operands as the MUL instruction except that it preserves the sign of the product.

IMUL Instruction

- IMUL sets the Carry and Overflow flags if the high-order product is not a sign extension of the low-order product.

```
Mov    al, 48
Mov    bl, 4
Imul bl                    ;AX = 00C0h,  OF = 1
```

AH is not a sign extension of AL, so the Overflow flag is set.

Example:

- Suppose AX contains FFFFh and BX contains FFFFh:

Instruction	Decimal Product	Hex Product	DX	AX	CF/OF
MUL BX	4294836225	FFFE0001	FFFE	0001	1
IMUL BX	1	00000001	0000	0001	0

DIV and IDIV

- DIV (Division) unsigned division.
- IDIV (Integer Division) signed division.
  - DIV reg IDIV reg
  - DIV mem IDIV mem
- The syntax is:
  - DIV divisor ; divisor is 8, 16, or 32-bit register or memory operand.

And

  - IDIV divisor ; divisor is 8, 16, or 32-bit register or memory operand.
- Always perform with accumulator (AX).
- Effected flag are only over and carry flag.

Byte Form

- AX is dividend
- AL keep the result/quotient
- AH keep the remainder

Divisor, BX	Dividend, AX	Quotient, AL
	Remainder, AH	

```
MOV AX, 0017h
MOV BX, 0001h
DIV BX                                  ; AX = 0017
```

Word Form

- DX:AX dividend.
- AX keep the result/quotient
- DX keep the remainder.

Divisor, AX	Dividend, DX:AX	Quotient, AX
	Remainder, DX	

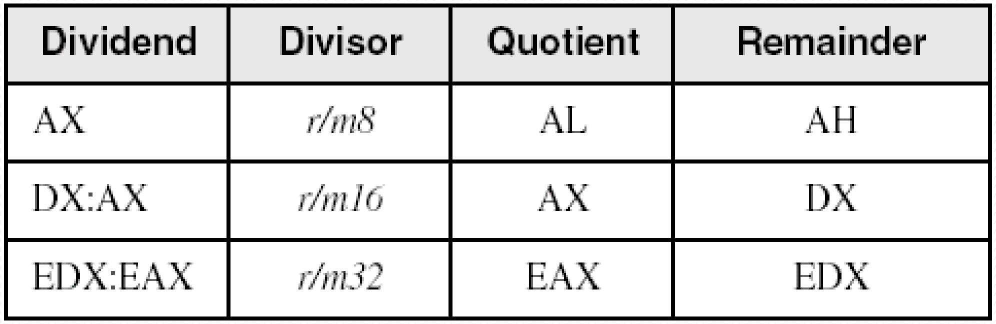
```
MOV AX,4022h
MOV DX,0000h
MOV CX,1000h
DIV CX                          ; AX = 0004
                               ;DX = 0022
```

DIV Instruction

- The DIV (unsigned divide) instruction performs 8-bit, 16-bit, and 32-bit division on unsigned integers
- A single operand is supplied (register or memory operand), which is assumed to be the divisor
- Instruction formats:

```
DIV r/m8
DIV r/m16
DIV r/m32
```

Default Operands:



Dividend	Divisor	Quotient	Remainder
AX	*r/m8*	AL	AH
DX:AX	*r/m16*	AX	DX
EDX:EAX	*r/m32*	EAX	EDX

DIV Examples

Example: 8-bit Unsigned Division

```
Mov   ax,0083h          ;dividend
Mov   bl, 2h            ;divisor
Div   bl                ; AL = 41h, AH = 01h
```

Quotient is 41h, remainder is 1

IDIV Instruction(Signed Division)

- Performs signed integer division, using the same operands as the DIV instruction

- The dividend must be sign-extended into the high order register before IDIV executes.

IDIV Instruction (signed division)

- IDIV divides AX, DX:AX, or EDX:EAX (dividend) by an 8, 16, or 32-bit ***signed*** register or memory operand (divisor)

- Syntax:

  IDIV *divisor* ; divisor is 8, 16, or 32-bit register or memory operand.

- **Operands:**

Divisor (explicit)	Dividend(implicit)	Quotient	Reminder
8-bit reg/mem operand	AX	AL	AH
16-bit reg/mem operand	DX:AX	AX	DX
32-bit reg/mem operand	EDX:EAX	EAX	EDX

CBW, CWD, CDQ Instructions

- The CBW, CWD, and CDQ instructions provide important sign-extension operations:
  - CBW (convert byte to word) extends AL into AH
  - CWD (convert word to doubleword) extends AX into DX
  - CDQ (convert doubleword to quadword) extends EAX into EDX

Examples:

```
  .data
 Byteval              SBYTE        -48
.code
  mov  al, byteval   ;dividend
  cbw                              ;extend AL into AH
  mov                bl, 5         ;divisor
  idiv               bl            ;AL = -9, AH = -3
```

Divide Overflow

- If the quotient is too large to fit into the destination operand, a divide overflow results. This causes a CPU interrupt, and the current program halts.

```
Mov    ax, 1000h
Mov    bl, 10h
Div    bl                          ;AL cannot hold 100h
```

DIV Overflow

We can use 16-bit divisor to reduce the possibility of divide overflow.

```
Mov   ax, 1000h
Mov   dx, 0                    ;clear DX
Mov   bx, 10h
Div   bx                       ;AX = 0100h
```

Dividing by Zero

- We don’t know enough (yet!) to prevent an overflow, but we can prevent a division by zero by comparing the divisor with zero before proceeding

- If divisor is zero, jump to an error return and skip the code with the divide.

IDIV Examples

Example: 16-bit division of –48 by 5

```
mov  ax,-48
cwd           ; extend AX into DX
mov  bx,5
idiv bx  ; AX = -9,  DX = -3
```

Example: 32-bit division of –48 by 5

```
mov  eax,-48
cdq       ; extend EAX into EDX
mov  ebx,5
idiv ebx  ; EAX = -9,  EDX = -3
```

dividend

(Word division)

- In word division , the divided is in DX:AX even if the actual divided will fit in AX . In this case DX should be prepared as follows :

- For DIV , DX should be cleared.

- For IDIV , DX should be made the sign extension of AX . The instruction CWD(convert word to doubleword) will do the extension.

Example

Example: Divide – 1100 by 7 .

Solution :

```
MOV AX, -1100      ; AX gets dividend
CWD                 ; extend sign to DX
MOV BX, 7           ; BX has divisor
IDIV BX             ; AX gets quotient , DX

has remainder
```

Byte division

- In byte division , the divided is in AX . If the actual divided is a byte, then AH should be prepared as follows:

- For DIV,AH should be cleared.
- For IDIV , AH should the sign extension of AL. The instruction CBW (convert byte to word) will do the extension.

Example

- Divide the signed value of the byte variable 1050 by -7.

Solution :

```
        MOV AL , XBYTE    ; AL has divided
        CBW                ; Extend sign to AH
         MOV BL , -7        ; BL has divisor
         IDIV BL            ; AL has quotient ,AH
has remainder
```

There is no effect of CBW and CWD on the flags.

THE END