Chapter-9

Multiplication and Division Instructions

Overview

In Chapter 7, left and right shifts can be used for multiplying and dividing by powers of 2. In this chapter, we introduce instructions for multiplying and dividing any numbers.

The process of multiplication and division is different for signed and unsigned numbers, so there are different instructions for signed and unsigned multiplication and division.

9.1 MUL and IMUL (Signed Versus Unsigned Multiplication)

Because signed and unsigned multiplication lead to different results, there are two multiplication instructions:

MUL (multiply) for unsigned multiplication.

IMUL (integer multiply) for signed multiplication.

These instructions multiply bytes or words. If two bytes are multiplied, the product is a word (16 bits). If two words are multiplied, the product is a doubleword (32 bits). The syntax of these instructions is

MUL source

and

IMUL source

Byte Form

For byte multiplication, one number is contained in the source and the other is assumed to be in AL. The 16-bit product will be in AX. The source may be a byte register or memory byte, but not a constant.

Word Form

For word multiplication, one number is contained in the source and the other is assumed to be in AX. The most significant 16 bits of the doubleword product will be in DX, and the least significant 16 bits will be in AX (we sometimes write this as DX:AX). The source may be a 16-bit register or memory word, but not a constant.

For multiplication of positive numbers (0 in the msb), MUL and IMUL give the same result.

Effect of MUL/IMUL on the Status Flags

SF, ZF, AF, PF :	Undefined.
CF/OF :	
After MUL, CF/OF	= 0 if the upper half of the result is zero.
	= 1 otherwise
After IMUL, CF/OF	= 0 if the upper half of the result is the sign extension of the lower half (the bits of the upper half are same as the sign bit of the lower half)
	= 1 otherwise

For both MUL and IMUL, CF/OF = 1 means that the product is too big to fit in the lower half of the destination (AL for byte multiplication, AX for word multiplication).

Examples

Example 9.1: Suppose AX contains 1 and BX contains FFFFH)

Instruction	Decimal product	Hex product	DX	AX	CF/OF
MUL BX	65535	0000FFFF	0000	FFFF	0
IMUL BX	-1	FFFFFFFF	FFFF	FFFF	0

Example 9.2: 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

9.2 <u>Simple Applications of MUL and IMUL</u>

Example 9.6 : Translate the high-level language assignment statement A = 5 x A – 12 x B into assembly code.

Solution :

```
MOV AX, 5
IMUL A
MOV A, AX
MOV AX, 12
IMUL  B
SUB A, AX
```

Example 9.7 : Write a procedure FACTORIAL that will compute N! for a positive integer N.

<u>Solution</u>:

```
FACTORIAL  PROC
MOV AX, 1
TOP:
MUL CX
LOOP TOP
RET
FACTORIAL ENDP
```

9.1 DIV and IDIV (Signed Versus Unsigned Division)

There are separate instructions for signed and unsigned division:

DIV (divide) for unsigned division.

IDIV (integer divide) for signed division.

The syntax is

DIV divisor

and

IDIV divisor

These instructions divide 8 (or 16) bits into 16 (or 32) bits. The quotient and remainder have the same size as the divisor.

9.1 DIV and IDIV (Signed Versus Unsigned Division)

Byte Form

In this form, the divisor is an 8-bit register or memory byte. The 16-bit dividend is assumed to be in AX. After division, the 8-bit quotient is in AL and the 8-bit remainder is in AH. The divisor may not be a constant.

Word Form

Here the divisor is a 16-bit register or memory word. The 32-bit dividend is assumed to be in DX:AX. After division, the 16-bit quotient is in AX and the remainder is in DX. The divisor may not be a constant.

For signed division, the reminder has the same sign as the dividend. If dividend and divisor are positive, DIV and IDIV give the same result.

The effect of DIV/IDIV on the flags is that all status flags are undefined.

Example 9.8: Suppose DX contains 0000H, AX contains 0005H, and BX contains 0002H.

Instruction	**Decimal quotient**	**Decimal remainder**	**AX**	**DX**
DIV BX	2	1	0002	0001
IDIV BX	2	1	0002	0001

Example 9.9: Suppose DX contains 0000H, AX contains 0005H, and BX contains FFFEH.

Instruction	**Decimal quotient**	**Decimal remainder**	**AX**	**DX**
DIV BX	0	5	0000	0005
IDIV BX	-2	1	FFFE	0001

<u>Example</u> 9.10: Suppose DX contains FFFFH, AX contains FFFBH, and BX contains 0002H.

Instruction	**Decimal quotient**	**Decimal remainder**	**AX**	**DX**
IDIV BX	-2	-1	FFFE	FFFF
DIV BX	DIVIDE OVERFLOW			

<u>Example</u> 9.11: Suppose AX contains 00FBH, BL contains FFH.

Instruction	**Decimal quotient**	**Decimal remainder**	**AX**	**DX**
DIV BL	0	251	FB	00
IDIV BL	DIVIDE OVERFLOW			