**PMAS Arid Agriculture University Rawalpindi**

**University Institute of Information Technology**

**LAB MANUAL – XIV, XV**

**Class/Program: BS (CS)** **Course: COAL (CS530)**

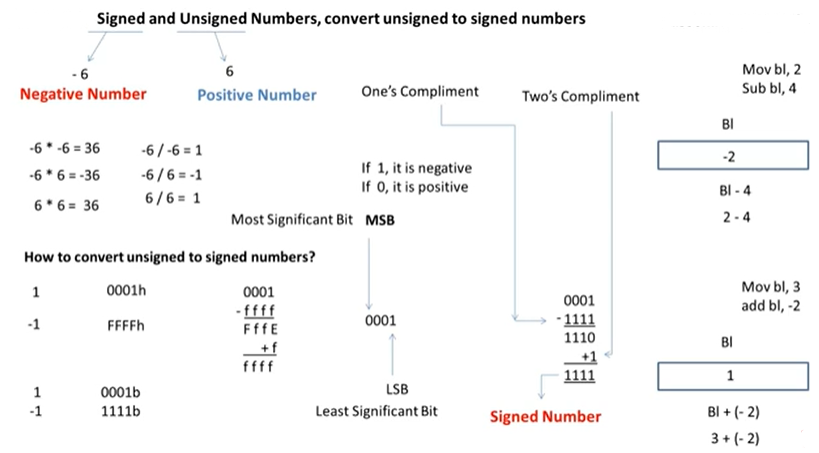


**Objectives:**

**1. signed and unsigned numbers**

**2. Introduction to Multiplication in assembly**

**3. Introduction to Division in assembly**



## The MUL/IMUL Instruction

There are two instructions for multiplying binary data. The MUL (Multiply) instruction handles unsigned data and the IMUL (Integer Multiply) handles signed data. Both instructions affect the Carry and Overflow flag.

### Syntax

The syntax for the MUL/IMUL instructions is as follows −

MUL/IMUL multiplier

Multiplicand in both cases will be in an accumulator, depending upon the size of the multiplicand and the multiplier and the generated product is also stored in two registers depending upon the size of the operands. Following section explains MUL instructions with three different cases −

|  |  |
| --- | --- |
| **Sr.No.** | **Scenarios** |
| 1 | **When two bytes are multiplied −**  The multiplicand is in the AL register, and the multiplier is a byte in the memory or in another register. The product is in AX. High-order 8 bits of the product is stored in AH and the low-order 8 bits are stored in AL.  Arithmetic1 |
| 2 | **When two one-word values are multiplied −**  The multiplicand should be in the AX register, and the multiplier is a word in memory or another register. For example, for an instruction like MUL DX, you must store the multiplier in DX and the multiplicand in AX.  The resultant product is a doubleword, which will need two registers. The high-order (leftmost) portion gets stored in DX and the lower-order (rightmost) portion gets stored in AX.  Arithmetic2 |
| 3 | **When two doubleword values are multiplied −**  When two doubleword values are multiplied, the multiplicand should be in EAX and the multiplier is a doubleword value stored in memory or in another register. The product generated is stored in the EDX:EAX registers, i.e., the high order 32 bits gets stored in the EDX register and the low order 32-bits are stored in the EAX register.  Arithmetic3 |

### Example

MOV AL, 10

MOV DL, 25

MUL DL

...

MOV DL, 0FFH ; DL= -1

MOV AL, 0BEH ; AL = -66

IMUL DL

## The DIV/IDIV Instructions

The division operation generates two elements - a **quotient** and a **remainder**. In case of multiplication, overflow does not occur because double-length registers are used to keep the product. However, in case of division, overflow may occur. The processor generates an interrupt if overflow occurs.

The DIV (Divide) instruction is used for unsigned data and the IDIV (Integer Divide) is used for signed data.

### Syntax

The format for the DIV/IDIV instruction −

DIV/IDIV divisor

The dividend is in an accumulator. Both the instructions can work with 8-bit, 16-bit or 32-bit operands. The operation affects all six status flags. Following section explains three cases of division with different operand size −

|  |  |
| --- | --- |
| **Sr.No.** | **Scenarios** |
| 1 | **When the divisor is 1 byte −**  The dividend is assumed to be in the AX register (16 bits). After division, the quotient goes to the AL register and the remainder goes to the AH register.  Arithmetic4 |
| 2 | **When the divisor is 1 word −**  The dividend is assumed to be 32 bits long and in the DX:AX registers. The high-order 16 bits are in DX and the low-order 16 bits are in AX. After division, the 16-bit quotient goes to the AX register and the 16-bit remainder goes to the DX register.  Arithmetic5 |
| 3 | **When the divisor is doubleword −**  The dividend is assumed to be 64 bits long and in the EDX:EAX registers. The high-order 32 bits are in EDX and the low-order 32 bits are in EAX. After division, the 32-bit quotient goes to the EAX register and the 32-bit remainder goes to the EDX register.  Arithmetic6 |