Multiplication and Division

Multiplication

Multiplication is performed on bytes, words.

- ✓ If **two 8-bit** numbers are multiplied, they generate a **16-bit** product;
- ✓ if **two 16-bit** numbers are multiplied, they generate a **32-bit** product;

Some flag bits (overflow and carry) change when the multiply instruction executes and produce predictable (متوقّع) outcomes. The other flags also change, but their results are unpredictable (متقلّب) and therefore are unused.

- ✓ In an **8-bit** multiplication, if the most significant 8 bits of the result are zero, **both CF and OF bits equal zero.** These flag bits show that the result is 8 bits wide (CF = 0) or 16 bits wide (CF= 1).
- ✓ In a **16-bit** multiplication, if the most significant 16-bits part of the product is 0, both CF and OF clear to zero.

8-Bit Multiplication.

With 8-bit multiplication, the **multiplicand** is always in the **AL** register, whether signed or unsigned. The multiplier can be any 8-bit register or any memory location.

The multiplication instruction contains **one operand** because it always multiplies the operand times the contents of register **AL**.

An example is :

```
MUL BL
```

Which multiplies the unsigned contents of AL by the unsigned contents of BL. After the multiplication, the unsigned product is placed in AX—a double-width product. Table below illustrates some 8-bit multiplication instructions.

Assembly Language	Operation
MUL CL	AL is multiplied by CL; the unsigned product is in AX
IMUL DH	AL is multiplied by DH; the signed product is in AX
IMULBYTE PTR[BX]	AL is multiplied by the byte contents of the data segment memory location addressed by BX; the signed product is in AX
MUL TEMP	AL is multiplied by the byte contents of data segment memory location TEMP; the unsigned product is in AX

Suppose that BL and CL each contain two 8-bit unsigned numbers, and these numbers must be multiplied to form a 16-bit product stored in DX. This procedure cannot be accomplished by a single instruction because we can only multiply a number times the AL register for an 8-bit multiplication. The **Example** below shows a short program that generates . This example loads register BL and CL with example data 5 and 10. The product, a 50, moves into DX from AX after the multiplication by using the MOV DX,AX instruction.

Example:

```
MOV BL,5 ;load data
MOV CL,10
MOV AL,CL ;position data
MUL BL ;multiply
MOV DX,AX ;position product
```

For signed multiplication, the product is in binary form, if positive, and in two's complement form, if negative. These are the same forms used to store all positive and negative signed numbers used by the microprocessor. If the program of previous Example multiplies two signed numbers, only the MUL instruction is changed to **IMUL.**

16-Bit Multiplication.

Word multiplication is very similar to byte multiplication. The difference is that AX contains the multiplicand instead of AL, and the 32-bit product appears in DX–AX instead of AX. The **DX** register always contains the most significant 16 bits of the product, and **AX** contains the least significant 16 bits. As with 8-bit multiplication, the choice of the multiplier is up to the programmer. The Table below shows several different 16-bit multiplication instructions.

Assembly Language	Operation
MUL CX	AX is multiplied by CX; the unsigned product is in DX–AX
IMUL DI	AX is multiplied by DI; the signed product is in DX–AX
MUL WORD PTR[SI]	AX is multiplied by the word contents of the data segment memory

Division

As with multiplication, division occurs on 8- or 16-bit numbers in the 8086–80286 microprocessors. These numbers are:

- ✓ Signed (**IDIV**) or
- ✓ Unsigned (**DIV**) integers.

The dividend is always a double-width dividend that is divided by the operand. This means that an 8-bit division divides a 16-bit number by an 8-bit number; a 16-bit division divides a 32-bit number by a 16-bit number; There is no immediate division instruction available to any microprocessor. None of the flag bits change predictably(بشكل متوقع) for a division.

A division can result in two different types of errors;

- ✓ One is an attempt to divide by zero and,
- ✓ The other is a divide overflow. A divide overflow occurs when a small number divides into a large number.

For example,

suppose that AX=3000 and that it is divided by 2. Because the quotient for an 8-bit division appears in AL, the result of 1500 causes a divide overflow because the 1500 does not fit into AL. In either case, the microprocessor generates an interrupt if a divide error occurs. In most systems, a divide error interrupt displays an error message on the video screen.

8-Bit Division.

An 8-bit division uses the AX register to store the dividend that is divided by the contents of any 8-bit register or memory location. The quotient moves into AL after the division with AH containing a whole number remainder. For a signed division, the quotient is positive or negative; the remainder always assumes the sign of the dividend and is always an integer.

For example,

if AX = 0010H (+16) and BL = 0FDH1 (-3) the

IDIV BL instruction executes, AX = 01FBH

Below Table lists some of the 8-bit division instructions.

Assembly Language	Operation
DIV CL	AX is divided by CL; the unsigned quotient is in AL and the unsigned remainder is in AH.
IDIV BL	AX is divided by BL; the signed quotient is in AL and the signed remainder is in AH.
DIV BYTE PTR[BP]	AX is divided by the byte contents of the stack segment memory location addressed by BP; the unsigned quotient is in AL and the unsigned remainder is in AH.

Example

```
MOV AL,NUMB ;get NUMB
MOV AH,0 ;zero-extend
DIV NUMB1 ;divide by NUMB1
MOV ANSQ,AL ;save quotient
MOV ANSR,AH ;save remainder
```

This Example illustrates a short program that divides the unsigned byte contents of memory location NUMB by the unsigned contents of memory location NUMB1. Here, the quotient is stored in location ANSQ and the remainder is stored in location ANSR. **Notice** how the contents of location NUMB are retrieved from memory and then zero-extended to form a 16-bit unsigned number for the dividend.

16-Bit Division.

Sixteen-bit division is similar to 8-bit division, except that instead of dividing into AX, the 16-bit number is divided into DX–AX, a 32-bit dividend. The quotient appears in AX and the remainder appears in DX after a 16-bit division. Table below lists some of the 16-bit division instructions.

Assembly Language	Operation
DIV CX	DX–AX is divided by CX; the unsigned quotient is AX and the unsigned remainder is in DX
IDIV SI	DX–AX is divided by SI; the signed quotient is in AX and the signed remainder is in DX
DIV NUMB	DX–AX is divided by the word contents of data segment memory NUMB; the unsigned quotient is in AX and the unsigned

Example

```
MOV AX,–100 ;load a –100
MOV CX,9 ;load +9
CWD ;sign-extend
IDIV CX
```

This Example shows the division of two 16-bit signed numbers. Here, in AX –100 is divided by +9 in CX. The CWD instruction converts the –100 in AX to –100 in DX–AX before the division. After the division, the results appear in DX–AX as a quotient of –11 in AX and a remainder of –1 in DX.

3- Basic Logic Instructions

The basic logic instructions include:

- ✓ AND
- ✓ OR
- ✓ Exclusive-OR, and
- ✓ NOT

Logic operations provide binary bit control in low-level software. The logic instructions allow bits to be set, cleared, or complemented. Low-level software appears in machine language or assembly language form and often controls the I/O devices in a system. **All logic instructions affect the flag bits**. Logic operations always **clear the carry and overflow flags**, while the other flags change to reflect the condition of the result.

Example AND instructions

Assembly Language	Operation
AND AL,BL	AL = AL and BL
AND CX,DX	CX = CX and DX

AND CL,33H	CL = CL and 33H
AND DI,4FFFH	DI = DI and 4FFFH
AND AX,[DI]	The word contents of the data segment memory location addressed by DI are ANDed with AX
AND [EAX],CL	CL is ANDed with the byte contents of the data segment memory location addressed by ECX

Example OR instructions.

Assembly Language	**Operation**
OR AH,BL	AL = AL or BL
OR SI,DX	SI = SI or DX
OR DH,0A3H	DH = DH or 0A3H
OR DX,[BX]	DX is ORed with the word contents of data segment memory location addressed by BX.

Example Exclusive-OR instructions.

Assembly Language	**Operation**
XOR CH,DL	CH = CH xor DL
XOR SI,BX	SI = SI xor BX
XOR AH,0EEH	AH = AH xor 0EEH
XOR DI,00DDH	DI = DI xor 00DDH
XOR DX,[SI]	DX is Exclusive-ORed with the word contents of the data segment memory location addressed by SI

NOT and NEG

Logical inversion, or the one's complement (**NOT**), and arithmetic sign inversion, or the two's complement (**NEG**). These are two of a few instructions that contain only one operand. Table below lists some variations of the NOT and NEG instructions.

Example NOT and NEG instructions.

Assembly Language	Operation
NOT CH	CH is one's complemented
NEG CH	CH is two's complemented
NEG AX	AX is two's complemented
NOT BYTE PTR[BX]	The byte contents of the data segment memory location addressed by BX are one's complemented

More Examples

Instructions	AL
MOV AL, 0101 0101B	0101 0101 B
AND AL,0001 1111B	0001 0101 B
OR AL,1100 0000B	1101 0101 B
XOR AL,0000 1111B	1101 1010 B