Multiplication and Division instruction:

Multiplication and Division can be performed on signed or unsigned numbers. For unsigned numbers, **MUL** and **DIV** instructions are used, while for signed numbers **IMUL** and **IDIV** are used.

Mnemonic	Meaning	Format	Operation	Flags Affected
MUL	Multiply (unsigned)	MUL S	(AL) · (S8) → (AX) (AX) · (S16) → (DX),(AX)	OF, CF SF, ZF, AF, PF undefined
DIV	Division (unsigned)	DIV S	(1) Q((AX)/(S8)) → (AL) R((AX)/(S8)) → (AH) (2) Q((DX,AX)/(S16)) → (AX) R((DX,AX)/(S16)) → (DX) If Q is FF_{16} in case (1) or $FFFF_{16}$ in case (2), then type 0 interrupt occurs	OF, SF, ZF, AF, PF, CF undefined
IMUL	Integer multiply (signed)	IMUL S	(AL) · (S8) → (AX) (AX) · (S16) → (DX),(AX)	OF, CF SF, ZF, AF, PF undefined
IDIV	Integer divide (signed)	IDIV S	(1) Q((AX)/(S8)) → (AL) R((AX)/(S8)) → (AH) (2) Q((DX,AX)/(S16)) → (AX) R((DX,AX)/(S16)) → (DX) If Q is positive and exceeds $7FFF_{16}$ or if Q is negative and becomes less than 8001_{16}, then type 0 interupt occurs	OF, SF, ZF, AF, PF, CF undefined
AAM	Adjust AL for multiplication	AAM	Q((AL)/10) → (AH) R((AL)/10) → (AL)	SF, ZF, PF OF, AF,CF undefined
AAD	Adjust AX for division	AAD	(AH) · 10 + (AL) → (AL) 00 → (AH)	SF, ZF, PF OF, AF, CF undefined
CBW	Convert byte to word	CBW	(MSB of AL) → (All bits of AH)	None
CWD	Convert word to double word	CWD	(MSB of AX) → (All bits of DX)	None

(a)

Source
Reg8
Reg16
Mem8
Mem16

(b)

(a)Multiplication and division arithmetic instructions (b) Allowed operands.

How we multiply in hexadecimal?

Ex: multiply 2A3C * B7

__2A3C
____B7

_127A4
1D094

1E30E4

That's the answer (checked with a hex calculator). Digit by digit, let's go through it.

7*C = 7*12 = 84 Dec. = 54 Hex (write 4, carry 5)
7*3+5 = 26 Dec. = 1A Hex (write A, carry 1)
7*A+1 = 71 Dec. = 47 Hex (write 7, carry 4)
7*2+4 = 18 Dec = 12 Hex

This completes the 7*2A3C = 127A4

B*C = 11*12 = 132 Dec. = 84 Hex (write 4, carry 8)
B*3+8 = 11*3+8 = 41 Dec.= 29 Hex (write 9, carry 2)
B*A+2 = 11*10+2 = 112 Dec. = 70 Hex (write 0, carry 7)
B*2+7 = 11*2+7 = 29 Dec. = 1D Hex

This completes the B*2A3C = 1D094

Adding the : 127A4 + 1D0940
4+0 = 4
A+4 = E
7+9 = 16 = (write 0, carry 1)
2+0+1 = 3
1+D = E
1 = 1

Answer: 1E30E4

Note: The easy way to check the result of multiplication or division numbers in Hex.

- **Convert the numbers from Hex numbers to decimal numbers**
- **compute the operation of multiplication or division**
- **Convert the result from decimal numbers to Hex numbers.**

Ex: multiply 2A3C H * B7 H

2A3C H → 10812 Dec.

00B7 H → 183 Dec.

10812 * 183 = 1978596 Dec.

1978596 → 1E30E4 H

Ex: divide 0085 H / 35 H

0085 H → 133

35 H → 53

133 / 53 = 2.509433962264151

Q= 2 Dec. → 02 H

R= (2.509433962264151 – 2) * 53 = 27 Dec. → 1B H

❖ **Multiplication:-**

- **MUL Unsigned multiplication**
- **IMUL Signed multiplication**

MUL Instruction : MUL source.

This instruction multiplies an **unsigned** byte (8-bit) from source and **unsigned** byte in AL register or unsigned word (16-bit) from source and unsigned word in AX register.

The source can be a register or a memory location.

When the byte (8-bit) is multiplied by the contents of AL, the result is stored in AX.

The most significant byte is stored in AH and least significant byte is stored in AL.

(AL) × (8-bit operand) → (AX)

When a word (16-bit) is multiplied by the contents of AX, the most significant word of result is stored in DX and least significant word of result is stored in AX.

(AX) × (16-bit operand) → (DX, AX)

***Flags** : MUL instruction affected* **OF**, **CF**

SF,ZF,AF,PF undefined

IMUL Instruction : IMUL source

This instruction multiplies a **signed** byte from some source and a **signed** byte in AL, or a signed word from some source and a signed word in AX.

The source can be register or memory location.

When a signed byte (8-bit) is multiplied by AL a signed result will be put in AX.

(AL) × (8-bit operand) → (AX)

When a signed word (16-bit) is multiplied by AX, the high-order word of the signed result is put in DX and the low-order word of the signed result is put in AX.

(AX) × (16-bit operand) → (DX, AX)

Note:

- If the upper byte of a 16-bit result (AH) or the upper word of 32-bit result (DX) contains only copies of the sign bit (all O's or all 1's), then the CF and the OF flags will both be 0's. The AF, PF, SF, and ZF flags are undefined after IMUL.

- To multiply a signed byte by a signed word it is necessary to move the byte into a word location and fill the upper byte of the word with copies of the sign bit. This can be done using **CBW** instruction.

***Flags** : IMUL instruction affected* **OF, CF**

SF,ZF,AF,PF undefined

- Note that the multiplication of two 8-bit number is 16-bit number
- Note that the multiplication of two 16-bit number is 32-bit number
- **IMUL** is similar to **MUL** but is used for signed numbers
- Note that the destination operand for instructions **MUL** and **IMUL** is either **AX** or both **DX** and **AX**

Ex:- what is the result of executing the following instruction?

MUL CL

What is the result of executing the following instruction?

IMUL CL

Assume that AL contains (-1) → FFH (the 2'complement of the number 1), CL contain (-2) → FEH (the 2'complement of the number 2).

Sol:

❖ Division

- **DIV : Unsigned division**
- **I D I V : signed division**

DIV Instruction : DIV source

This instruction is used to divide an **unsigned** word (16-bit) by a byte (8-bit) or to divide an **unsigned** double word (32-bit) by a word (16-bit).

When dividing a word by a byte, the word must be in AX register. After the division **AL** will contain an **8-bit quotient** and **AH** will contain an **8-bit remainder**.

(AX)/(8-bit operand) → (AH), (AL)

Note: *If an attempt is made to divide by 0 or the quotient is too large to fit in AL (greater than FFH), the 8086 will automatically execute a type 0 interrupt.*

When a double word is divided by a word, the most significant word of the double word must be in DX and the least-significant word must be in AX.

After the division **AX** will contain a **16-bit quotient** and **DX** will contain a **16-bit remainder**.

(DX, AX)/(16-bit operand) → (DX), (AX)

Note: *Again, if an attempt is made to divide by 0 or the quotient is too large to fit in AX register (greater than FFFFH), the 8086 will do a type 0 interrupt.*

For DIV instruction source may be a register or memory location.

To divide a byte by a byte, it is necessary to put the dividend byte in AL and fill AH with all 0's. Similarly, to divide a word by a word, it is necessary to put the dividend word in AX and fill DX with all 0's.

Flags : All flags are undefined after a DIV instruction.

IDIV Instruction : IDIV source

This instruction is used to divide a **signed** word (16-bit) by a **signed** byte (8-bit), or to divide a signed double word (32-bits) by a signed word(16-bit). The IDIV is similar to DIV instruction.

Note: if quotient is positive and exceeds 7FFF H or if quotient is negative and becomes less than 8001 H, then type 0 interrupt occurs.

Ex: Assume that each instruction starts from these values:

AL= 85H , BL= 35H , AH= 0H

1. **MUL BL** = AL . BL = 85H * 35H =

   133 Dec. * 53Dec.= 7049Dec.= 1B89H

2. **IMUL BL** = AL . BL = 2'S AL * BL = 2'S(85H) * 35 H=

   7BH *35H = 123 Dec. * 53 Dec. = 6519 Dec. =

   1977 H → 2'S comp →E689H → AX

   Note:
   D7 of AL = 1; it is mean a negative number. Convert to 2's complement and then multiply it.

3. **DIV BL** = AX/BL = 0085H/35H = 133 Dec./ 53 Dec.

   = 2.509433962264151

   Quotient=2 Dec. = 02H

   Remainder = 0.509433962264151 Dec. = 1B H

AH (remainder)	AL (quotient)
1B	02

4. **IDIV BL** = AX/BL = 0085H/35H = 133 Dec./ 53 Dec.

   Quotient=2 Dec. = 02H

   Remainder = 0.509433962264151 Dec. = 1B H

> *Note:*
> *D15 of AX = 0; it is mean a positive number .no need to Convert to 2's complement*

AH (remainder)	AL (quotient)
1B	02

Ex: Assume that each instruction starts from these values:

AL = F3H, BL = 91H, AH = 00H

1. **MUL BL** = AL * BL = F3H * 91H = 89A3H

   → AX = 89A3H

2. **IMUL BL** = AL * BL = 2'S AL * 2'S BL = 2'S (F3H) * 2'S (91H)

   = 0DH * 6FH = 05A3H

   → AX = 05A3H

> *Note:*
> *D7 of AL = 1 & D7 of BL =1; it is mean a negative numbers. Convert to 2's complement and then multiply it.*

3. **DIV BL** = AX / BL = 00F3H / 91H =

AH (remainder)	AL (quotient)
62	01

4. **IDIV BL** = AX/BL = 00F3H / 2'S (91H) = 00F3H/ 6FH

   Quotient= 02H

   Remainder = 15 H

> *Note:*
> *D7 of BL =1; it is mean a negative numbers. Convert to 2's complement and then divide it.*

AH (remainder)	AL (quotient)
15	02

→, but $\frac{\text{Positive}}{\text{negative}}$ = negative , so

AH (remainder)	AL (quotient)
15	2'comp(02)

⇨

AH (remainder)	AL (quotient)
15	FE

Ex: Assume that each instruction starts from these values:

AX= F000H, BX= 9015H, DX= 0000H

1. MUL BX = F000H * 9015H =

DX	AX
8713	**B000**

2. IMUL BX =2'S(F000H) *2'S(9015H) = 1000 * 6FEB =

DX	AX
06FE	**B000**

3. DIV BL $\frac{AX}{BL} = \frac{F000H}{15H}$ = 0B6DH → more than FFH → Divide Error

4. IDIV BL $\frac{AX}{BL} = \frac{2'(F000H)}{15H} = \frac{1000H}{15H}$ =C3H → more than 7FH → Divide Error

Ex: Assume that each instruction starts from these values:

AX= 1250H, BL= 90H

1. IDIV BL $\frac{AX}{BL} = \frac{1250H}{90H} = \frac{positive}{negative} = \frac{positive}{2'negative} = \frac{1250}{2'(90H)} = \frac{1250H}{70H}$
= 29H quotient and 60H remainder

But 29H(positive) →2'S(29H)= D7H →

AH (Remainder)	AL (quotient)
60H	**D7H**

2. DIV BL $\frac{AX}{BL} = \frac{1250H}{90H}$ =20H →

AH (Remainder)	AL (quotient)
50H	**20H**

Note:

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. the *dividend*, must be converted to a 16-bit wide number in AX. This is accomplished differently for signed and unsigned numbers.

- For unsigned numbers, the most-significant 8-bits AH must be cleared to zero (zero-extended).
- For signed numbers, the least-significant 8-bits AL are sign (D_7)-extended into the most significant 8-bits AH.

A special instruction sign (D_7) - extends AL into AH, or converts an 8-bit signed number in AL into a 16-bit signed number in AX.

The **CBW** (convert byte to word) instruction performs this conversion.

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 in DX after a 16-bit division.

If AX is a 16-bit signed number, the **CWD** (convert word to double word) instruction sign-extends it into a signed 32-bit number.

No flag effected

- **CBW** : Convert byte into word.

if high bit of AL = 1 then:

AH = 255 (FF h)

else AH = 0

Ex:

```
MOV AX, 0          ; AH = 0 ,  AL = 0
MOV AL, -5         ; AX = 00FB h (251)
 CBW                ; AX = FFFB h (-5)
```

- **CWD** : Convert Word to Double word.

if high bit of AX = 1 then:

DX = 65535 (FFFF h)

else DX = 0

Ex:

```
MOV DX, 0          ; DX = 0
MOV AX, 0          ; AX = 0
MOV AX, -5         ; DX AX = 0000 h : FFFB h
CWD                ; DX AX = FFFF h : FFFB h
```

❖ As Fig. (a) shows, adjust instructions for BCD multiplication and division are also provided. They are adjust AX far multiply (AAM) and adjust AX for divide (AAD).The AAM instruction assumes that the instruction just before it multiplies two unpacked BCD numbers with their product produced in AL. The AAD instruction assumes that AH and AL contain unpacked BCD numbers.

- **AAM instruction:**

After the two unpacked BCD digits are multiplied, the AAM instruction is used to adjust the product to two unpacked BCD digits in AX.

Ex: Assume **AL** = 0000 0100 : Unpacked BCD 4 and

CL = 0000 0110 : Unpacked BCD 6

MUL CL → (AL *CL) → AX = 0000 0000 0001 1000 = 0018H

AAM → AX = 0000 0010 0000 0100 = 0204H Which is unpacked BCD for 24.

- **AAD Instruction :**

AAD converts two unpacked BCD digits in AH and AL to the equivalent binary number in AL. This adjustment must be made before dividing the two unpacked BCD digits in AX by an unpacked BCD byte.

After the division AL will contain the unpacked BCD quotient and AH will contain the unpacked BCD remainder. The PF, SF and ZF are updated. The AF, CF and OF are undefined after AAD.

Ex: Assume AX = 0403 unpacked BCD for 43 decimal, CL = 07H

Divide AX by unpacked BCD in CL

AAD → Adjust to binary before division

→ AX = 002BH = 2BH = 43 decimal.

DIV CL → Divide AX by unpacked BCD in CL.

AL → quotient = 06 unpacked BCD

AH → remainder = 01 unpacked BCD