Multiplication and Division Instructions

2. *Second group of Arithmetic and Logic instructions contains: Multiplication and Division Instructions (***MUL**, **IMUL**, **DIV**, **IDIV**).

These types of operands are supported:

Instruction	operand
MUL	REG
IMUL	memory
DIV	
IDIV	

REG 8-bit : AH, AL, BL, BH, CH, CL, DH, DL.

REG 16-bit :AX, BX, CX, DX, SI, DI, BP, SP.

memory: [BX], [BX+SI+7], variable, etc...

MUL and **IMUL** instructions affect these flags only: **CF**, **OF.** When result is over operand size these flags are set to **1**, when result fits in operand size these flags are set to **0**. For **DIV** and **IDIV** flags are undefined.

- **MUL -** Multiplies an unsigned multiplicand by an unsigned multiplier. MUL treats a leftmost 1-bit as a data bit, not a negative sign.
- **IMUL** – Multiplies a signed multiplicand by a signed multiplier. IMUL treats a leftmost bit as a sign (0 = positive , 1= negative).

Algorithm of MUL and IMUL:

MUL operand **IMUL operand**	
when operand is a **byte**: AX = AL * operand	when operand is a **word**: DX AX = AX * operand

Size of Multiplicand	Multiplicand	Multiplier(operand) Reg/memory	Product=Multiplicand * Multiplier(operand)	Example
8-bit(byte)	AL	8-bit (byte)	AX = AL * operand (byte)	MUL BL IMUL BL
16-bit(word)	AX	16-bit (word)	DX AX = AX * operand (word)	MUL BX IMUL BX

Examples of MUL instruction:

1.When multiply unsigned (byte * byte):

Ex1. Write the suitable instructions to multiply unsigned 30h by 20h.

```
MOV BL, 30h      ; BL= 30h
MOV  AL , 20h    ; AL = 20h
MUL BL            ; AX = AL * BL = 20h* 30h = 0600h
```

2.When multiply unsigned (word * word):

Ex2.Write the suitable instructions to multiply unsigned 1122h by 3344h.

```
MOV BX, 1122h     ; BX= 1122h
MOV  AX , 3344h   ; AX = 3344h
MUL BX               ; DX AX = AX * BX = 1122h* 3344h = 036E 5308h
```

3.When multiply unsigned (byte * word) or unsigned (word * byte):

Ex3.Write the suitable instructions to multiply unsigned   80h by 1F1Ch.

```
MOV BL, 80h          ; BL= 80h
MOV BH, 0             ; BH = 0
MOV  AX , 1F1Ch    ; AX = 1F1Ch
MUL BX                ; DX AX = AX * BX =1F1Ch* 0080h = 000F 8E00h
```

Examples of IMUL instruction:

1.When multiply signed (byte * byte):

Ex1.Write the suitable instructions to multiply signed 20h by F0h.

```
MOV BL, F0h       ; BL= F0h
MOV  AL , 20h    ; AL = 20h
IMUL BL            ; AX = AL * BL = 20h* F0h = FE00h
```

2.When multiply signed (word * word):

Ex2.Write the suitable instructions to multiply signed   8201h by E304h.

```
MOV CX, 8201h       ; CX= 8201h
MOV  AX , E304h    ; AX = E304h
IMUL CX               ; DX AX = AX * CX =  E304h* 8201h  = 0E43 EB04h
```

CBW instruction: Convert byte into word

The form of this instruction:

```
CBW
```

Note: this instruction without operand.

Algorithm :

```
If high bit  (b7) of AL = 1 then:
    AH = FFh  (255d)
 Else
    AH = 0
End
```

CWD instruction: Convert word into double word

The form of this instruction:

CWD

Note: this instruction without operand.

Algorithm :

```
If high bit (b15) of AX = 1 then:
    DX = FFFFh  (65535d)
 Else
    DX = 0
End
```

3.When multiply signed (byte* word) or (signed word * byte):
Ex3.Write the suitable instructions to multiply signed 80 h by 1F1Ch.

```
MOV AL, 80 h          ; AL= 80h
CBW                   ;  bit 7 of AL = 1 then AH = FF  : AX = FF80h
MOV  BX , 2F0Ch       ; BX = 2F0Ch
IMUL BX               ; DX AX = AX * BX = FF80h* 2F0Ch = 000F 8E00h
```

DIV and IDIV instructions:

- **DIV** – Divides an unsigned dividend by an unsigned divisor. DIV treats a leftmost 1-bit as a data, not a minus sign.
- **IDIV** – Divides a signed dividend by a signed divisor. IDIV treats a leftmost bit as a sign (0 = positive, 1= negative). To extend length of a signed dividend use CBW and CWD).

Algorithm of DIV and IDIV:

DIV operand **IDIV operand**	

when operand is a **byte**:
AL = AX / operand
AH = remainder
	when operand is a **word**:
	AX = (DX AX) / operand
	DX = remainder

DIV and IDIV						
Size of dividend	**Dividend**	**Divisor(operand)** **Rag/memory**	**Dividend/ divisor(operand)**	**Quotient**	**Remainder**	**Example**
16-bit(word)	AX	8-bit (byte)	AL = AX /operand(byte)	AL	AH	DIV BH IDIV BH
32-bit(double word)	DX AX	16-bit (word)	AX=(DX AX)/operand(word)	AX	DX	DIV CX IDIV CX

Examples of DIV instruction:

1. When divide unsigned (word / byte):

Ex1.Write the suitable instructions to divide 2000h over 80h (unsinged).

```
MOV AX,2000h    ; AX= 2000h
MOV BL,80h       ; BL =80
DIV BL           ; AX = AX/BL ; AL (Quotient)= 40 ; AH(Remainder)=00
```

2. When divide unsigned (double word / word):

Ex2.Write the suitable instructions to divide 0020 C000h over 2000h (unsinged). Assume the word 2000h stored in memory locations starting at 7000h

```
MOV DX,0020h
MOV AX,C000h
MOV WORD PTR[7000 h],2000h  ; [7000]=00 ; [7001]=20h
DIV WORD PTR [7000h]          ;  DXAX = DXAX/ word ptr [7000h]
                               ;  DXAX = 0020 C000h / 2000h = 0000 0106
                             ; AX(Quotient)=0106 ; DX(Remainder)=0000
```

3.When divide unsigned (byte / byte) :

Ex3.Write the suitable instructions to divide 33 h over 12h (unsigned).

```
MOV AL,33h
MOV AH,00
MOV DL,12H
DIV DL          ;  AX= 0F02  ; AL(Quotient)=02 ; AH(Remainder)= 0F

OR
MOV AX,0033
MOV DL,12 H
DIV DL
```

4. When divide unsigned (word / word) :

Ex4.Write the suitable instructions to divide 2000 h over 1000h (unsigned)**.** Assume 2000 stored in memory location at F800h

```
MOV WORD PTR[F800h],2000h    ; [F800]=00,[F801]=20
MOV AX,[F800h]                ; AX = 2000
MOV DX,0                      ; DX=0
MOV BX,1000 H                 ; BX=1000
DIV BX                       ;  DXAX= DXAX/BX
                              ;  DXAX= 0000 2000/ 1000= 0000 0002
                          ;   AX(Quotient)= 0002; DX(Remainder)=0000
```

Examples of IDIV instruction:

1. When divide signed (word / byte):

Ex1.Write the suitable instructions to divide 2000h over 80h (singed).

```
MOV AX,2000h    ; AX= 2000h
MOV BL,80h      ; BL =80
IDIV BL         ; AX = AX/BL =00C0
                ; AL (Quotient)= C0 ; AH(Remainder)=00
```

2.When divide signed (double word / word):

Ex2.Write the suitable instructions to divide 0020 C000h over 8000h (singed).

```
MOV DX,0020h
MOV AX,C000h
MOV SI,8000h  ;
IDIV SI              ;  DXAX = DXAX/SI=   0020 C000/ 8000= 4000 FFBF
                     ;   AX(Quotient)= FFBF; DX(Remainder)=4000
```

3. When divide signed (byte / byte) :

Ex3. Write the suitable instructions to divide 9Eh over 12 h (signed)

```
MOV DL, 12H      ;  DL=12h
MOV AL,9EH       ;  AL= 9Eh
CBW              ;  AL = 9Eh ; B7  of AL =1 then AH=FF
IDIV DL          ; AX= AX/DL = FF9E/12= F8FEh
                 ; AL (Quotient)= FE ; AH(Remainder)=F8
```

4.When divide signed (word / word):

Ex4. Write the suitable instructions to divide 9900 over the content of memory location F900h (as signed word). Assume the content of memory location is 0300h

```
MOV AX,9900h                     ; AX= 9900h
CWD                               ; AX=9900 h; b15 oF AX=1 then DX=FFFF
MOV WORD PTR [F900h],0300h   ;  [F900]=00 , [F901]=03
IDIV WORD PTR[F900h   ]        ; DXAX= DXAX/WORD PTR[F900]=FF00 FFDE
                               ; AX(Quotient)=FFDE  ;  DX(Remainder)=FF00
```

INC, DEC, NOT, NEG instructions

3. *Third group of Arithmetic and Logic instructions contains*: INC, DEC, NOT, NEG

These types of operands are supported:

	operand
INC	REG
DEC	memory
NOT	
NEG	

REG 8-bit : AH, AL, BL, BH, CH, CL, DH, DL.

REG 16-bit :AX, BX, CX, DX, SI, DI, BP, SP.

memory: [BX], [BX+SI+7], variable, etc...

INC, **DEC** instructions affect these flags only: **ZF**, **SF**, **OF**, **PF**, **AF**.
NOT instruction does not affect any flags!
NEG instruction affects these flags only: **CF**, **ZF**, **SF**, **OF**, **PF**, **AF**.

INC instruction: Increment by 1

Algorithm
Operand =operand + 1

Ex.

```
MOV   DI ,7000h    ; DI = 7000h
INC DI               ; DI = 7001h
```

DEC instruction: Decrement by 1

Algorithm
Operand =operand - 1

Ex.

```
MOV   CL , 0Fh    ;  CL=0Fh
DEC    CL            ; CL = 0E1h
```

NOT instruction: reverse (Invert) each bit of operand (one's complement).

Algorithm

- if bit is 1 turn it to 0.
- if bit is 0 turn it to 1.

Ex.

```
MOV  byte ptr[9000], FF h          ; [9000]= FF
NOT byte ptr[9000]                 ; [9000]= 00
```

NEG instruction: make operand negative (two's complement). Actually it reverses each bit of operand and then adds 1 to it.

Algorithm

- Invert all bits of the operand
- Add 1 to inverted operand

Ex1.

```
MOV AL, 05       ; AL = 05
NEG  AL          ; AL= FBh
NEG  AL          ; AL = 05
```

4. *Fourth group of Arithmetic and Logic instructions contains*: ASCII and decimal adjustment (DAA , DAS , AAA , AAD, AAM , AAS).