# Computer Organization & Assembly Language

- MUL & DIV Instructions

# Multiplication

- **MUL** instruction is used with unsigned operands
- Unsigned multiplication of 128 and 255 = 32, 640.
- 2 bytes multiplied, product = 1 word (16 bits)
  - AL is implied destination operand
  - Source operand can be a register or variable (8 bits)
  - 16-bit output is AX
- 2 words multiplied, product = 1 double word (32 bits)
  - AX is implied destination operands
  - Source operand can be a register or variable (16 bits)
  - 32-bit output is DX:AX

# MUL Instruction

- 8-bit multiplication:

```
MUL BL              ; product = AX
```

- 16-bit multiplication:

```
MUL DX              ; product = DX:AX
```

```
mov al,5
mov bl,10h
mul bl                  ; AX = 0050h
```

```
mov ax,500h
mov bx,100h
mul bx                  ; DX:AX = 00050000h
```

# MUL (Flags)

Carry and Overflow flags are set when the product extends into its high register:

```
mov ax,5000h
mov bx,10h
mul bx                  ; DX:AX = 00050000h
                        ; CF=1, OF=1
```

```
mov ax,500h
mov bx,10h
mul bx                  ; DX:AX = 00005000h
                        ; CF=0, OF=0
```

SF, ZF, PF, AF: undefined

# IMUL Instruction

- Use with signed operands
- Sign-extends the result into the high register
- CF=1 and OF=1 if the sign of the high register is different from the sign of the low register

```
mov al,-4
mov bl,4
imul bl      ; AX = FFF0h (-16), CF=0, OF = 0
```

```
mov al,48
mov bl,4
imul bl      ; AX = 00C0h (+192), CF=1, OF = 1
```

# IMUL Examples

```
mov ax,-128
mov bx,4
imul bx      ; DX:AX = FFFFh,FE00h  (-512)
             ; CF=0, OF = 0
```

# Example 9.6 (From Textbook)

Suppose A and B are two word variables:

**A = 5 * A – 12 * B** (Assume no overflow occurs)

**Solution:**

```
MOV AX, 5        ;AX = 5
IMUL A           ;AX = 5 * A
MOV A, AX        ;A = 5 * A
MOV AX, 12       ;AX = 12
IMUL B           ;AX = 12 * B
SUB A, AX        ;A = 5 * A – 12 * B
```

# Exercise for practice

- Suppose A, B, C are word variables and all products will fit in 16 bits.
- A = 5 x A – 7
- B = (A – B) x (B + 10)
- A = 6 – 9 x A

# Example 9.7

- Compute N! (factorial) for a positive integer N
- Assumption: product does not overflow 16 bits!
- Algorithm:

N! = 1 if N = 1

N x (N – 1) x (N – 2) x .. X 1 if N > 1

```
product = 1
term = N
FOR N times Do
        product = product x term
        term = term – 1
END FOR
```

# Example 9.7 Continued

```
; input CX = N
; output AX = N!

    MOV AX, 1
    TOP:
        MUL CX
        LOOP TOP
        RET
```

# DIV Instruction

- Dividend is *divided* by divisor
- *Result*: Quotient and Remainder
- Byte Form:
  - Dividend is AX
  - Divisor can be 8-bit register or variable
  - Quotient is AL, Remainder is AH
- Word Form:
  - Dividend is DX:AX
  - Divisor can be 16-bit register or variable
  - Quotient is AX, Remainder is DX
- Status flag values are undefined

# DIV Examples

```
mov ax,0083h        ; dividend
mov bl,2            ; divisor
div bl              ; AL = 41h, AH = 01h
```

```
mov dx,0            ; dividend, high
mov ax,8003h        ; dividend, low
mov cx,100h         ; divisor
div cx              ; AX = 0080h, DX = 0003h
```

# Divide Overflow

Happens when the quotient is too large to fit in the destination register. Causes a processor interrupt like divide error - overflow.

```
mov dx,0050h        ; dividend, high
mov ax,0000h        ; dividend, low
mov cx,10h          ; divisor
div cx              ; quotient= 50000h, cannot
                    ; fit in AX register
```

# IDIV Instruction

- Use for signed division
- Dividend must be sign-extended before executing the IDIV instruction:
  - CBW (Convert byte to word) extends AL into AH
  - CWD (Convert word to double word) extends AX into DX
- Status flag values are undefined

```
mov ax,-257          ; dividend
mov bl,2             ; divisor
idiv bl              ; AL = -128, AH = -1
```

```
mov ax,-5000              ; dividend
cwd                       ; extend into DX
mov bx,256                ; divisor
idiv bx                   ; AX = -19, DX = -136
```

# Decimal Output

- 240

Step 1: Divide 240 by 10. Quotient = 24, Remainder = 0

Step 2: Divide 24 by 10. Quotient = 2, Remainder = 4

Step 3: Divide 2 by 10. Quotient = 0, Remainder = 2

# Decimal Output Algorithm

1. IF AX < 0 THEN
2. Print a minus sign
3. Replace AX by its 2's complement
4. END IF
5. Get the digits in ax's decimal representation
6. Convert these digits to character and print them

Line 5:

Count = 0

REPEAT

    divide quotient by 10

    push remainder on the stack

    count = count + 1

UNTIL quotient = 0

# Decimal Output Algorithm Contd..

Line 6

FOR count times DO

    POP a digit from stack

    Convert it into a character

    Output the character

END FOR

# Decimal Input Algorithm

**Example**: Input 123

Total = 0
Read an ASCII digit
REPEAT
    convert character to binary value
    total = 10 x total + value
    read a character
UNTIL character is carriage return

**Practice 2nd version of decimal input algorithm given on pg 171**

# Chapter 9 Exercise

- Question # 1-6
- Question # 7, part a, b, c