### **Intro to Assembly**

### 

### **📌 1. MOV – Data Transfer**

**Purpose**: Move data between registers, memory, or constants.

MOV AL, 5 ; Move 5 into AL

MOV BL, AL ; Copy value from AL to BL

MOV AX, [1234h] ; Move data from memory address 1234h into AX (advanced usage)

**Note**: You cannot move memory-to-memory directly. Always use a register in between.

### **➕ 2. ADD – Addition**

**Purpose**: Add two values.

MOV AL, 4

MOV BL, 3

ADD AL, BL ; AL = 4 + 3 = 7

### **➖ 3. SUB – Subtraction**

**Purpose**: Subtract one value from another.

MOV AL, 7

MOV BL, 2

SUB AL, BL ; AL = 7 - 2 = 5

**What happens if the result is negative?** It becomes a **two's complement** value (e.g., 1 - 2 = 255 or FFh, because 8086 uses unsigned arithmetic by default).

### **⬆️ 4. INC – Increment**

**Purpose**: Increase the value of a register or memory by 1.

MOV AL, 5

INC AL ; AL = 6

### **⬇️ 5. DEC – Decrement**

**Purpose**: Decrease the value by 1.

MOV AL, 5

DEC AL ; AL = 4

### **🔄 6. NEG – Negate (Two’s Complement)**

**Purpose**: Converts a number to its negative form using 2's complement.

MOV AL, 5

NEG AL ; AL = -5 = FBh (in unsigned, shows as 251)

### **✖️ 7. MUL – Unsigned Multiplication**

#### **8-bit example:**

MOV AL, 4

MOV BL, 3

MUL BL ; AX = AL \* BL = 12

; Result is in AX (AL \* BL = 0x0C)

#### **16-bit example:**

MOV AX, 1234h

MOV BX, 0002h

MUL BX ; DX:AX = AX \* BX = 0x2468

; Result is split: AX = low 16 bits, DX = high 16 bits (if needed)

### **➗ 8. DIV – Unsigned Division**

#### **8-bit example:**

MOV AX, 10 ; Dividend (must be in AX)

MOV BL, 3 ; Divisor

DIV BL ; AL = quotient, AH = remainder (10 / 3 = 3 R1)

#### **16-bit example:**

MOV DX, 0 ; High part of 32-bit dividend

MOV AX, 2468h ; Low part (total dividend = 0x00002468)

MOV BX, 2 ; Divisor

DIV BX ; AX = quotient, DX = remainder (0x2468 / 2 = 0x1234)

### **🔢 9. Working with Numbers**

* **Binary**: MOV AL, 1010b
* **Decimal**: MOV AL, 10
* **Hexadecimal**: MOV AX, 0ABCDh

### **🔤 10. Characters and ASCII**

Characters are stored as their **ASCII values**.

MOV AL, 'A' ; AL = 65 = 41h

### **🧵 11. Strings and Output**

Strings must be terminated with $ and shown using DOS interrupts.

msg DB 'Hello, World!$'

LEA DX, msg

MOV AH, 09h

INT 21h ; DOS function to display string

### **🧱 12. Memory Segments and Initialization**

Every .COM program must initialize the DS register:

MOV AX, @DATA

MOV DS, AX

This is required to access variables from the .DATA segment.

### **💻 13. Full Example Recap**

.MODEL SMALL

.STACK 100H

.DATA

num1 DB 5

num2 DB 3

msg DB 'Demo Done$'

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

MOV AL, num1

MOV BL, num2

ADD AL, BL

SUB AL, 2

INC AL

DEC AL

NEG AL

MOV AL, 4

MOV BL, 2

MUL BL

MOV AX, 10

MOV BL, 3

DIV BL

LEA DX, msg

MOV AH, 09h

INT 21h

MOV AX, 4C00h

INT 21h

MAIN ENDP

END MAIN