**Macros and Procedures**

## 

### **📘 Overview**

Here we learned how to use **macros** and **procedures** in 8086 assembly language to make programs more **modular, readable, and reusable**. These constructs are essential for reducing redundancy and improving code organization in low-level programming.

## **🧩 1. Macros**

### **🔍 What I Learned:**

* Macros are like **code templates** that get expanded at compile time.
* They can take **parameters**, perform logic, and include local labels.
* Macros help avoid repeating code and are faster than procedures (no call/return overhead).

### **📌 Example 1: Factorial Using a Macro**

factorial MACRO n

LOCAL loop

MOV AX, 1

MOV CX, n

loop:

MUL CX

LOOP loop

ENDM

; Usage:

factorial 4 ; AX = 4 \* 3 \* 2 \* 1 = 24

### **📌 Example 2: Macro with Parameters – Max of Two**

max2 MACRO a, b

MOV AL, a

CMP AL, b

JG skip

MOV AL, b

skip:

ENDM

; Usage:

max2 5, 8 ; AL = max(5, 8) = 8

## **🔧 2. Procedures**

### **🔍 What I Learned:**

* Procedures are blocks of reusable code you can **CALL** and **RET** from.
* Useful for large or repeating tasks (e.g., printing, searching, string reversal).
* Parameters are passed via **registers**, memory, or the stack (we used registers).

### **📌 Example 1: Procedure to Print a Message**

PrintMsg PROC

LEA DX, msg

MOV AH, 09h

INT 21h

RET

PrintMsg ENDP

### **📌 Example 2: Max of Three Numbers (via Registers)**

; Inputs: AL, BL, CL → Output: max in AL

MaxOfThree PROC

CMP AL, BL

JG skip1

MOV AL, BL

skip1:

CMP AL, CL

JG done

MOV AL, CL

done:

RET

MaxOfThree ENDP

### **📌 Example 3: Search an Array for a Value**

SearchArray PROC

MOV CX, 10 ; 10 elements

LEA SI, nums

check:

MOV AL, [SI]

CMP AL, valToCheck

JE found

INC SI

LOOP check

; Not found

LEA DX, msgNotFound

JMP show

found:

LEA DX, msgFound

show:

MOV AH, 09h

INT 21h

RET

SearchArray ENDP

## **🔁 Macros vs Procedures**

| **Feature** | **Macros** | **Procedures** |
| --- | --- | --- |
| Executed When | Compile time (expanded inline) | Run-time (called with CALL) |
| Speed | Faster (no CALL/RET overhead) | Slightly slower (CALL/RET) |
| Reusability | High for short inline tasks | Better for large or looped tasks |
| Parameter Passing | Via macro parameters | Via registers or memory |

## **✅ Final Takeaways**

| **Skill Learned** | **Description** |
| --- | --- |
| Define and use macros | Created parameterized macros for math logic |
| Nested macro logic | Used macros inside macros (e.g. compare in findMax3) |
| Create clean procedures | Wrote modular, reusable routines |
| Register-based logic | Passed data via AL, BL, CL |
| Modular code structure | Separated logic cleanly via PROC/ENDP |