**Arrays and Addressing Modes**

### **📘 Overview**

Here explored how to **declare, access, and manipulate arrays** in Intel 8086 assembly using various **addressing modes**. This included both **byte arrays** and **word arrays**, as well as techniques to process them using **registers** and **looping structures**.

Key concepts covered:

* Declaring arrays using DB, DW, and DUP
* Accessing array elements with different addressing modes
* Performing operations like sum, copy, reverse, and condition checks
* Using SI, DI, BX, and combinations for addressing
* Applying loops and conditionals to process arrays

## **🔧 Core Concepts & Examples**

### **1️⃣ Array Declaration Using DB, DW, and DUP**

#### **🔍 What I Learned:**

* Arrays in assembly are declared using DB (byte) and DW (word).
* DUP(?) creates uninitialized arrays of a specific size.

#### **💡 Example:**

byteArray DB 10 DUP(?) ; BYTE array of size 10

wordArray DW 5 DUP(?) ; WORD array of size 5

### **2️⃣ Input Values Into Array**

#### **🔍 What I Learned:**

* Used INT 21h with AH = 01h to get user input
* Converted ASCII input (like '5') to numeric value by subtracting 30h
* Stored into array using indexed addressing

#### **💡 Example:**

MOV AH, 01h

INT 21h ; Input character → AL = '5' (ASCII 35h)

SUB AL, 30h ; Convert to number 5

MOV byteArray[SI], AL ; Store in array

### **3️⃣ Accessing Arrays with Addressing Modes**

#### **🔍 What I Learned:**

Access array elements using different methods:

| **Addressing Mode** | **Example** |
| --- | --- |
| **Direct** | MOV AL, byteArray[0] |
| **Indexed (SI)** | MOV AL, byteArray[SI] |
| **Based (BX)** | MOV AL, byteArray[BX] |
| **Indexed + offset** | MOV AL, byteArray[SI+2] |
| **Based + indexed** | MOV AL, byteArray[BX+SI] |

#### **💡 Example:**

MOV SI, 0

MOV AL, byteArray[SI] ; Load first value from array using SI

### **4️⃣ Summing Array Elements**

#### **🔍 What I Learned:**

Loop through the array and use ADD to accumulate a total.

#### **💡 Example:**

MOV CX, 10

MOV SI, 0

MOV AL, 0

sum\_loop:

ADD AL, byteArray[SI]

INC SI

LOOP sum\_loop

### **5️⃣ Reversing an Array**

#### **🔍 What I Learned:**

* Use two pointers: one forward (SI), one backward (DI)
* Copy values from source to destination in reverse order

#### **💡 Example:**

MOV CX, 10

MOV SI, 0

MOV DI, 9

rev\_loop:

MOV AL, byteArray[SI]

MOV reverseArray[DI], AL

INC SI

DEC DI

LOOP rev\_loop

### **6️⃣ Copying an Array**

#### **🔍 What I Learned:**

Copy each element from one array to another using a loop.

#### **💡 Example:**

MOV CX, 10

MOV SI, 0

cpy\_loop:

MOV AL, byteArray[SI]

MOV copyArray[SI], AL

INC SI

LOOP cpy\_loop

### **7️⃣ Finding Maximum and Minimum**

#### **🔍 What I Learned:**

* Loop through array
* Use CMP, JG, and JL to compare values
* Store max in one register, min in another

#### **💡 Example:**

MOV AL, byteArray[0]

MOV BL, AL ; Max

MOV BH, AL ; Min

cmp\_loop:

CMP AL, BL

JG set\_max

CMP AL, BH

JL set\_min

### **8️⃣ Detecting Even and Odd Numbers**

#### **🔍 What I Learned:**

* Use DIV to divide element by 2
* Check remainder (AH) to determine even or odd

#### **💡 Example:**

MOV AL, byteArray[SI]

MOV AH, 0

MOV BL, 2

DIV BL ; AL / 2

CMP AH, 0 ; If remainder = 0 → even

## **🧠 Final Takeaways**

| **Skill** | **Description** |
| --- | --- |
| Declaring arrays | Using DB, DW, and DUP(?) |
| Addressing data | Indexed, based, and combined addressing |
| Processing arrays | Sum, copy, reverse using loops |
| Conditional logic | Finding max, min, even/odd |
| Efficient looping | Used CX with LOOP, SI, DI for iteration |