Chapter-10

# Arrays and Addressing Modes

## 10.1 One-Dimensional Arrays

In Chapter 4, we used the DB and DW pseudo-ops to declare byte and word arrays; for example, a five-character string named MSG,

MSG DB 'abcde'

or a word array W of six integers, initialized to 10,20,30,40,50,60

W DW 10,20,30,40,50,60

The address of the array variable is called the **base address of the array**. If the offset address is assigned to W is 0200H, the array looks like this in memory:

| Offset address | Symbolic Address | Decimal Contents |
|---|---|---|
| 0200H | W | 10 |
| 0202H | W+2H | 20 |
| 0204H | W+4H | 30 |
| 0206H | W+6H | 40 |
| 0208H | W+8H | 50 |
| 020AH | W+AH | 60 |

## The DUP Operator

It is possible to define arrays whose elements share a common initial value by using the **DUP** (duplicate) operator. It has this form:

repeat_count DUP (value)

This operator causes value to be repeated the number of times specified by repeat_count. For example,

GAMMA DW 100 DUP (0)

sets up an array of 100 words, with each entry initialized to 0. Similarly,

DELTA DB 212 DUP (?)

creates an array of 212 uninitialized bytes. DUPs may be nested. For example,

LINE DB 5,4, 3 DUP (2, 3 DUP (0), 1)

which is equivalent to

LINE DB 5,4,2,0,0,0,1,2,0,0,0,1,2,0,0,0,1

## Location of Array Elements

The address of an array element may be specified by adding a constant to the base address. Suppose A is an array and S denotes the number of bytes in an element (S=1 for byte array, S=2 for word array). The position of the elements in array A can be determined as follows:

| Position | Location |
|---|---|
| 1 | A |
| 2 | A = 1 x S |
| 3 | A = 2 x S |
| . | . |
| . | . |
| . | . |
| N | A = (N-1) x S |

**Example 10.1:** Exchange the 10th and 25th elements in a word array W.

**Solution:** W[10] is located at address W + 9 x 2 = W + 18 and W[25] is at W + 24 x 2 = W + 48, so we can do the exchange as follows:

```
MOV  AX, W+18
XCHG  W+48, AX
MOV  W+18, AX
```

# 10.2 Addressing Modes

The way an operand is specified is known as its **addressing mode**. The addressing modes we have used so far are

(1) **Register Mode** - which means that an operand is a register.
(2) **Immediate Mode** - when an operand is a constant.
(3) **Direct Mode** - when an operand is a variable.

For example,

```
MOV  AX, 0          (Destination AX is register mode,
                     source 0 is immediate mode.)

ADD  ALPHA, AX      (Destination ALPHA is direct mode,
                     source AX is register mode.)
```

There are four additional addressing modes for the 8086:

(1) Register Indirect Mode
(2) Based Mode
(3) Indexed Mode
(4) Based Indexed Mode

These modes are used to address memory operands indirectly.

## 10.2.1 Register Indirect Mode

In this mode, the offset of the operand is contained in a register. We say that the register acts as a **pointer** to the memory location. The operand format is

[register]

The register is BX, SI, DI, or BP. For BX, SI, or DI, the operand's segment number is contained in DS. For BP, SS has the segment number.

For example, suppose that SI contains 0100H, and the word at 0100H contains 1234H. To execute

```
MOV  AX, [SI]
```

the CPU (1) examines SI and obtains the offset address 100H, (2) uses the address DS:0100H to obtain the value 1234H, and (3) moves 1234H to AX. This is not the same as

```
MOV   AX, SI
```

which simply moves the value of SI, namely 100H, into AX.

**Example 10.2:** Suppose that

| | |
|---|---|
| BX contains 1000H | Offset 1000H contains 1BACH |
| SI contains 2000H | Offset 2000H contains 20FEH |
| DI contains 3000H | Offset 3000H contains 031DH |

Tell which of the following instructions are legal and which are illegal.

a. MOV BX, [BX]
b. MOV CX, [SI]
c. MOV BX, [AX]
d. ADD [SI], [DI]
e. INC [DI]

**Solution:**

| | Source offset | Result |
|---|---|---|
| a. | 1000H | 1BACH |
| b. | 2000H | 20FEH |
| c. | Illegal source register | (must be BX, SI, or DI) |
| d. | Illegal memory-memory addition | |
| e. | 3000H | 031EH |

**Example 10.3:** Write some code to sum in AX the elements of the 10-element array W defined by

```
W    DW   10, 20, 30, 40, 50, 60, 70, 80, 90, 100
```

**Solution:**

```
        XOR    AX, AX
        LEA    SI, W
        MOV  CX, 10
ADDNOS:
        ADD   AX, [SI]
        ADD   SI, 2
        LOOP ADDNOS
```

# 10.2.2 Based and Indexed Addressing Mode

In these modes, the operand's offset address is obtained by adding a number called a **displacement** to the contents of a register. Displacement may be any one of the following:

the offset address of a variable
a constant (positive or negative)
the offset address of a variable plus or minus a constant

If A is a variable, examples of displacements are:

| | |
|---|---|
| A | (offset address of a variable) |
| -2 | (constant) |
| A + 4 | (offset address of a variable plus a constant) |

The syntax of an operand is any of the following equivalent expressions:

[register + displacement]
[displacement + register]
[register] + displacement
displacement + [register]
displacement [register]

The register must be BX, BP, SI, or DI. If BX, SI, or DI is used, DS contains the segment number of the operand' address. If BP is used, SS has the segment number.

The addressing mode is called **based** if BX (base register) or the BP (base pointer) is used.

The addressing mode is called **indexed** if SI (source index) or the DI (destination index) is used.

For example, suppose W is a word array, and BX contains 4. In the instruction

```
MOV  AX, W[BX]
```

The displacement is the offset address of variable W. The instruction moves the element at address W + 4 to AX. This is the third element in the array. The instruction could also have been written in any of these forms:

```
MOV  AX, [W + BX]
MOV  AX, [BX + W]
MOV  AX, W + [BX]
MOV  AX, [BX] + W
```

As another example, suppose SI contains the address of a word array W.

```
MOV  AX, [SI+2]
```

The instruction could also have been written in any of theses forms:

```
MOV AX, [2+SI]
MOV AX, 2+[SI]
MOV AX, [SI]+2
MOV  AX, [SI+2]
MOV  AX, 2[SI]
```

**Example 10.5:** Rework example 10.3 by using based mode.

**Solution:**

```
XOR  AX, AX
XOR  BX, BX
MOV  CX, 10
ADDNOS:
ADD  AX, W[BX]
ADD  BX, 2
LOOP  ADDNOS
```

**Example 10.6:** Suppose the ALPHA is declared as

```
ALPHA   DW   0123H, 0456H, 0789H, 0ABCDH
```

in the segment addressed by DS. Suppose also that

BX contains 2
SI contains 4
DI contains 1

Offset 002 contains 1084H
Offset 004 contains 2BACH

Tell which of the following instructions are legal:

a. MOV AX, [ALPHA+BX]
b. MOV BX, [BX+2]
c. MOV CX, ALPHA[SI]
d. MOV AX, -2[SI]
e. MOV BX, [ALPHA+3+DI]
f. MOV AX, [BX] 2
g. MOV BX, [ALPHA+AX]

**Solution:**

| Source offset | Number moved |
|---|---|
| a. ALPHA+2 | 0456H |
| b. 2+2 = 4 | 2BACH |
| c. ALPHA+4 | 0789H |
| d. -2+4 = 2 | 1084H |
| e. ALPHA+3+1 = ALPHA+4 | 0789H |
| f. Illegal form of source operand | |
| g. Illegal source register | |

**Example 10.7:** Replace each lowercase letter in the following string by its upper case equivalent. Use index addressing mode.

```
MSG   DB      'this is a message'
```

**Solution:**

```
MOV   CX, 17
XOR   SI, SI
TOP:
CMP  MSG[SI], ‘ ‘
JE   NEXT
AND   MSG[SI],  0DFH
NEXT:
INC  SI
LOOP  TOP
```