**Arrays**

Table of Contents

[ASCII Values 3](#_Toc83228503)

[Duplicate Values 4](#_Toc83228504)

We can declare an array like this:

someArray DB 48H, 65H, 6CH, 6CH, 6FH, 00H

ASSEMBLY

Here, we have an array with six elements, each of which is a DB variable. If the values are larger than 255, we can use DW instead.

The elements occupy consecutive memory locations, which is why we can access a specific element like this:

MOV AL, someArray[3]

ASSEMBLY

Alternatively, we can use one of the index registers, BX, SI, DI or BP.

MOV SI, 3  
MOV AL, someArray[SI]

ASSEMBLY

## ASCII Values

In the Intel 8086, each element we are storing in an array is actually being interpreted as ASCII values. The first five elements of the variable we created above, someArray, translates to ‘Hello’.

We can also directly store a string, in which case the ASCII values of each of the characters will be stored.

someOtherArray DB 'Hello', 0

ASSEMBLY

someOtherArray is exactly the same as someArray. However, note that if the variable is storing DW values, then we cannot use strings like this. We must use individual ASCII values.

The difference between declaring an array with characters and just using a string variable is that a string variable requires an end-of-string marker, $, which an array does not. However, if we want to display the array, then we will need the end-of-string marker.

## Duplicate Values

If we need a large array which has the same value for every element, we can use the DUP operator.

someArray DB 5 DUP(0)

ASSEMBLY

This is the same as declaring:

someArray DB 0, 0, 0, 0, 0

ASSEMBLY

We can even use alternating values:

someArray DB 5 DUP(1, 2)

ASSEMBLY

Which is the same as declaring:

someArray DB 1, 2, 1, 2, 1, 2, 1, 2, 1, 2

ASSEMBLY

Example

The code below takes a single number as input from the user, , which can have the value of either 0, 1, 2 or 3, and then adds the first numbers in an array and display the results as an output.

Study the comments of the code carefully since they contain additional information.

ORG 0100H  
  
.DATA *; Data segment starts*a DB 3, 1, 2 *; 1-D array for number*b DB 00H  
message DB 'Enter the value of N:$' *; 1-D array for string*.CODE *; Code segment starts*MAIN PROC  
 MOV AX, @DATA *; store data-segment address in AX* MOV DS, AX *; store data-segment address in DS register* XOR AX, AX *; reset AX* MOV SI, OFFSET a *; SI holds offset address of first element of a* MOV DI, OFFSET b *; DI holds offset of first element of b* MOV DX, OFFSET message *; DX holds offset of first element of message  
 ; LEA DX, message ; equivalent of above line* MOV AH, 09H *; display string function* INT 21H *; display message, since DX holds message* MOV AH, 01H *; single-character input* INT 21H  
 MOV CL, AL *; store user input in CL* SUB CL, 48 *; convert input to decimal (since stored as ASCII)* XOR AL, AL *; reset AL register* Loop\_1: *; save first N numbers of array a* ADD AL, [SI] *; add element value to AL* INC SI *; go to next element* LOOP Loop\_1 *; CL value decrements, loop ends when 0*

MOV BL, AL *; save final result to BL* ADD BL, 48 *; convert decimal value to ASCII* MOV AH, 02H *; single-character output* MOV DL, BL *; print result* INT 21H  
MAIN ENDP  
END MAIN  
RET

ASSEMBLY