Array Size Calculation with the $ Operator

SIZE OF AN ARRAY

When using an array, it is often useful to know its size. This can be done by explicitly stating the size of the array in the declaration, or by letting the assembler calculate it for you.

To explicitly state the size of an array, you simply list the number of elements in the array in the declaration. For example, the following declaration declares an array of 10 bytes:

```
array BYTE 10
```

To let the assembler calculate the size of an array, you can use the $ operator, which represents the current location counter (LC).

The **LC is a special register** that keeps track of the current address in memory where the assembler is writing code.

To calculate the size of an array using the LC, you simply subtract the offset of the array from the current LC. For example, the following declaration calculates the size of the array variable:

```
array BYTE 10, 20, 30, 40
array_size = ($ - array)
```

The array_size variable will now contain the value 4, which is the number of elements in the array variable.

It is important to note that the array_size variable must be declared immediately after the array variable. This is because the LC will be incremented by the size of the array variable when the assembler encounters it.

CALCULATING THE SIZE OF A STRING

To calculate the size of a string, you can use the same method as above. Simply subtract the offset of the string from the current LC. For example, the following declaration calculates the size of the string variable:

```
string BYTE "This is a long string, containing" BYTE "any number of characters"
string_size = ($ - string)
```

OR

```
string BYTE "This is a long string, containing"
       BYTE "any number of characters"
string_size = ($ - string)
```

The string_size variable will now contain the size of the string variable, including the null terminator.

SIZE OF ARRAYS OF WORDS AND DOUBLEWORDS

When calculating the size of an array of words or doublewords, you need to take into account the size of each element in the array. Words are 2 bytes wide, and doublewords are 4 bytes wide.

To calculate the size of an array of words, you simply divide the total array size (in bytes) by 2. For example, the following declaration calculates the size of the list variable:

```
list WORD 1000h, 2000h, 3000h, 4000h
list_size = ($ - list) / 2
```

The list_size variable will now contain the value 4, which is the number of elements in the list variable.

To calculate the size of an array of doublewords, you simply divide the total array size (in bytes) by 4. For example, the following declaration calculates the size of the list variable:

```
list DWORD 10000000h, 20000000h, 30000000h, 40000000h
list_size = ($ - list) / 4
```

The list_size variable will now contain the value 4, which is the number of elements in the list variable.

The image you sent shows the following text:

```
array BYTE 16
```

This is a declaration of an array named array with 16 bytes of storage. The BYTE keyword specifies that the array elements are bytes.

The word array is not a keyword in assembly language. It is simply a variable name. However, the array variable has a special meaning in assembly language.

When the assembler encounters an array variable, it allocates the specified amount of storage for the array and keeps track of the offset of the array in memory.

This means that the array variable can be used to access the array elements. For example, the following instruction will load the value of the first element of the array variable into the AL register:

```
array BYTE 16
mov al, array[0]
```

The array[0] expression refers to the first element of the array variable. The offset of the first element is calculated by the assembler.

In summary, the array variable is not a keyword in assembly language, but it has a special meaning. It is a variable that is used to store an array of data.

What does array byte 16 mean? does it mean the first element of the array is 16 or it means am writing down a variiable called array that will hold bytes, which are typically strings/string arrays?

• The statement array BYTE 16 in assembly language means that you are declaring a variable called array that can store 16 bytes. This could be used to store a string, a byte array, or any other type of data that is 16 bytes or less in size.

• The first element of the array variable will not necessarily be the value 16. The value of the first element will depend on what data is stored in the array. For example, if you store the string "Hello, world!" in the array variable, the first element will be the ASCII code for the letter 'H'.

• If you store a byte array in the array variable, the first element will be the first byte in the array. To access the first element of the array variable, you would use the following expression: **a-rray[0].**

• This expression will return the first byte in the array. To access the second element of the array variable, you would use the following expression: **array[1].** This expression will return the second byte in the array. And so on.

Array BYTE 10
Array BYTE 10, 20, 30, 40

difference???

The difference between the two array declarations is that the first one declares an array with a single element with the value 10, while the second one declares an array with four elements, with the values 10, 20, 30, and 40.

• **Array BYTE 10** allocates space for an array that can hold up to 10 bytes, but it doesn't set specific initial values. You need to assign values to the elements of the array separately.

• **Array BYTE 10, 20, 30, 40** declares an array and initializes it with specific values, where 10 is the first element, 20 is the second, 30 is the third, and 40 is the fourth.

---

1. Array BYTE 10:

This declaration allocates space for an array that can hold up to 10 bytes. It does not set the

initial value of the first element to 10. Instead, it allocates memory for a potential array of 10 bytes but doesn't assign any specific values to those bytes. So, there is no initial value of 10 for the first element. You would need to assign values to the individual elements of the array separately if you intend to use them.

2. Array BYTE 10, 20, 30, 40:

This declaration not only allocates space for an array but also initializes it with specific values. In this case, the array has four elements, and their initial values are 10, 20, 30, and 40. The 10 here is indeed the initial value of the first element.

To summarize:

- **Array BYTE 10** allocates space for an array of 10 bytes but does not assign an initial value of 10 to the first element. **It's merely reserving memory.**

- **Array BYTE 10, 20, 30, 40** declares an array with four elements and initializes them with the specified values, where 10 is the initial value of the first element.

The key distinction is whether you're allocating memory only or initializing values during the declaration. The 10 in the first declaration is related to the size (number of bytes) allocated, not an initial value for an element.
