Department of
Computer Science and Engineering

Title: Implement Array and String in Assembly Language Programming

Microprocessors and Microcontrollers Lab

CSE 304

Green University of Bangladesh

1 Objective(s)

- To understand the use of Array in Assembly Language Program.
- To understand the use of String in Assembly Language Program.

2 Problem analysis

2.1 Array

Arrays can be seen as chains of variables. A text string is an example of a byte array; each character is presented as an ASCII code value (0..255). Here are some array definition examples:

```
a DB 48h, 65h, 6Ch, 6Ch, 6Fh, 00h
```

In the definition above, we define an array whose each element is 1 bytes long and assign a as its identifier.

```
b DB 'Hello', 0
```

b is an exact copy of the a array, when compiler sees a string inside quotes it automatically converts it to set of bytes. This chart shows a part of the memory where these arrays are declared:

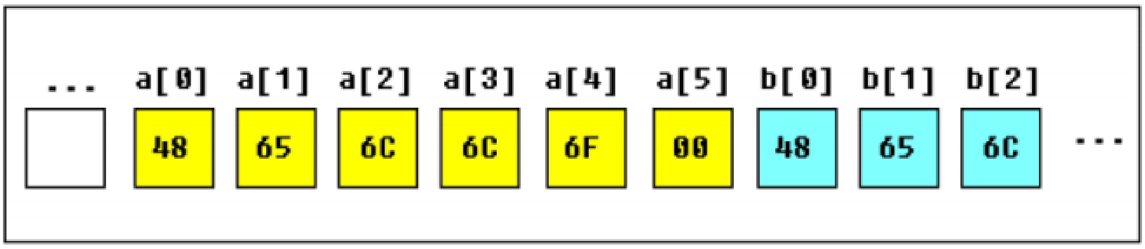

Figure 1: Array Structure

You can access the value of any element in array using square brackets, for example:

```
MOV AL, a[3]
```

You can also use any of the memory index registers BX, SI, DI, BP, for example:

```
MOV SI, 3
MOV AL, a[SI] ;Please note that only these registers can be used inside square brackets (as memory pointers):
BX, SI, DI, BP!
```

If you need to declare a large array you can use DUP operator. The syntax for DUP:
number DUP (value(s))
number - number of duplicate to make (any constant value).
value - expression that DUP will duplicate.

For example:
c DB 5 DUP(9) c DB 9, 9, 9, 9, 9 ; is an alternative way of declaring:
one more example:
d DB 5 DUP(1, 2)
d DB 1, 2, 1, 2, 1, 2, 1, 2, 1, 2 ; is an alternative way of declaring:
Of course, you can use DW instead of DB if it's required to keep values larger then 255, or smaller then -128.

2.2 String

We can store a string in .data segment. Here we have provided an example :

```
.DATA
S1 DW ’Hello World$’
```

2.2.1 Getting the Address of a String Variable

There is **LEA** (Load Effective Address) instruction and alternative **OFFSET** operator. Both OFFSET and LEA can be used to get the offset address of the variable.

- **OFFset:** The offset is typically used to get the offset of a label within a segment. It doesn’t perform complex calculations and is primarily used for accessing data within the current code or data segment.

  ```
  .data
  myString db "Hello, World!", 0
  .code
  mov ax, offset myString ; Load the offset of myString into ax and ax now contains the offset of myString
  ```

- **LEA** lea is a versatile instruction used for calculating memory addresses. It can perform arithmetic operations and is commonly used for more complex address calculations.

  ```
  .data
  array db 10, 20, 30, 40, 50
  index dd 2 ; Index of the element we want
  .code
  mov si, [index] ; Load the index into si
  lea ax, [array + si] ; Calculate the effective address of array + si ; ax now contains the address of array[2],
  which is 30
  ```

2.2.2 Printing a string

To print a string, we have to write the following instructions:

```
.DATA
S1 DW ’Hello World$’
.code
LEA DX,S1
MOV AH,09h
int 21h
```

3 Example of Array and String Code in Assembly

```
org 100h

.DATA ; Data segment starts
A db 3, 1, 2, 2, 1 ;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
mov ds, ax

xor ax,ax
mov si, OFFSET A
mov di, OFFSET B
```

```

mov dx, OFFSET message ; Load Effective Address of the message in DX register
; lea dx, message ; (similar meaning that Load Effective Address)
mov ah, 09h ;display string function
int 21h ;display message

mov ah, 01h
int 21h
mov cl, al
sub cl, 48 ; to convert the ascii value of 3 to decimal 3

xor al, al

Loop_1:
add al, [Si]
inc Si
loop Loop_1

mov bl, al
add bl, 48 ; to convert the ascii value of the output to decimal

mov ah, 02h
mov dl, 0Dh ; Clear Buffer
int 21h
mov dl, 0Ah ; for newline
int 21h

mov dl, bl
int 21h

mov ah, 4ch
int 21h

MAIN ENDP
END MAIN
RET
```

4 Sample Input/Output (Compilation, Debugging & Testing)

To derive summation of 3 + 1 + 2 + 2 + 1 using array A. Here, value of N is given by user where N=5 and output 9 will be shown in the output window:

```
Enter the value of N: 5
9
```

5 Discussion & Conclusion

Based on the focused objective(s) to understand about array and string in assembly language programming, the additional lab exercise made me more confident towards the fulfilment of the objectives(s).

6 Lab Task (Please implement yourself and show the output to the instructor)

1. Write an assembly language code to print out the elements in an array in reverse order.
2. Write an assembly language code to:
   a. Take any number of inputs in an array.
   b. Print out the elements in an array.

[NB: In all program you should use string for input and output message]

7 Lab Exercise (Submit as a report)

- Write an assembly language code to take natural number series as input and as output, show:
  a. The summation of odd numbers.
  b. The summation of even numbers.

[NB: In this program you should use string for input and output message]

8 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.