Palestine Technical University – Kadoorie

College of Engineering and Technology

Department of Computer Systems Engineering

Experiment number :7

Experiment title:

Arrays & AAM instruction

By:

Wael Melhem – 201911374

Supervisor:

Eng. Samer Sweileh

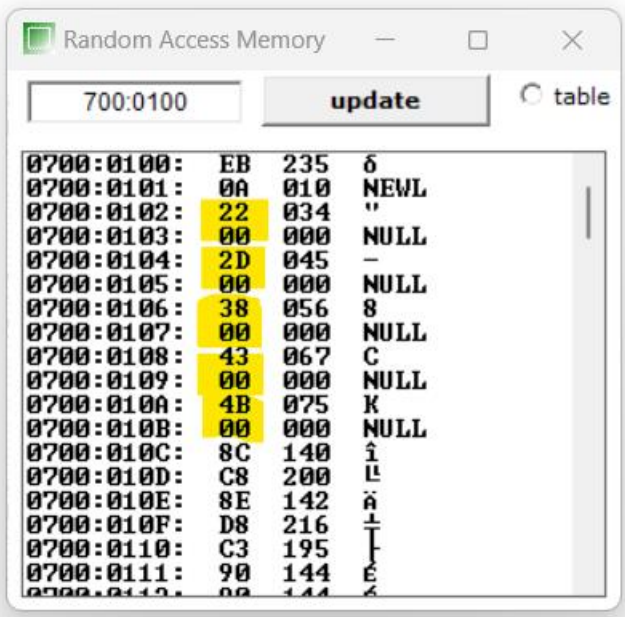
30 April, 2023,

Arrays:

- An array is a **<u>sequential collection of values</u>**, all of the **<u>same size and type</u>**.
- The data definition directives can be used for defining a one-dimensional array.
- To define a one-dimensional array of numbers: **NUMBERS DW 34, 45, 56, 67, 75.**

  Ex:

```
org 100h

.data
var DW 34,45,56,67,75
;allocate 6 word for this number
.code
main proc
mov ax,@data
mov ds,ax
;Wael Melhem


ret
```



- The above definition declares an array of five words each initialized with the numbers 34, 45, 56, 67, 75.
- This allocates 2x5 = 10 bytes of ***<u>consecutive</u>*** memory space.

Base Address of the array:

NUMBERS DW 34, 45, 56, 67, 75

- The <u>symbolic address of the first number will be NUMBERS</u> and that of the **second number will be NUMBERS + 2 and so on**.

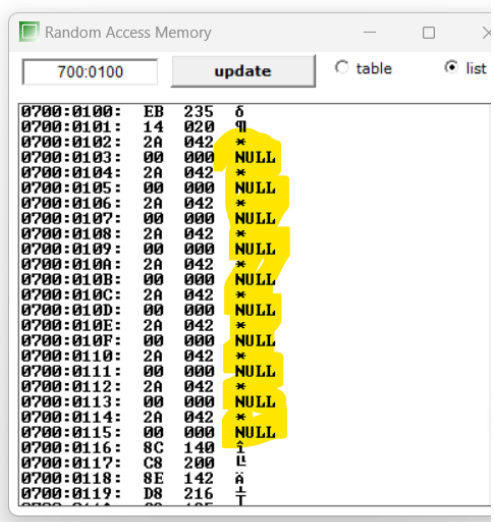
- Assume the offset address of the **array = 200h**, then:
    - **200h(NUMBERS**) contains the value **34.**
    - **202h**(**NUMBERS+2h**) contains the value **45.**
    - **204h**(**NUMBERS+4h**) contains the value **56**.
    - **206h**(**NUMBERS+6h**) contains the value **67**.
    - **208h**(**NUMBERS+8h**) contains the value **75**.

DUP (Duplicate) Operator:

- **DUP** is used to initialize an array with a number of items having the ***same value.***
- For examples:

1. NUMBERS DW 10 DUP ('*')

```
org 100h

.data
var   DW 10 DUP ('*')

;allocate 6 word for this number
.code
main proc
mov ax,@data
mov ds,ax
;Wael Melhem

ret
```



2. NUMBERS DW 6 DUP(?)

```
org 100h

.data
var   DW 6 DUP(?)

.code
main proc
mov ax,@data
mov ds,ax
;Wael Melhem


ret
```

Random Access Memory

700:0100 update

0700:0100:	EB	235	δ
0700:0101:	0C	012	♀
0700:0102:	00	000	NULL
0700:0103:	00	000	NULL
0700:0104:	00	000	NULL
0700:0105:	00	000	NULL
0700:0106:	00	000	NULL
0700:0107:	00	000	NULL
0700:0108:	00	000	NULL
0700:0109:	00	000	NULL
0700:010A:	00	000	NULL
0700:010B:	00	000	NULL
0700:010C:	00	000	NULL
0700:010D:	00	000	NULL
0700:010E:	8C	140	î
0700:010F:	C8	200	╚
0700:0110:	8E	142	Ä
0700:0111:	D8	216	╪
0700:0112:	C3	195	├
0700:0113:	90	144	É
0700:0114:	90	144	É
0700:0115:	90	144	É
0700:0116:	90	144	É
0700:0117:	90	144	É

3. Line DB 5,4,3 DUP(2, 3)= = Line DB 5, 4, 2, 3, 2, 3, 2, 3:

```
org 100h

.data
var   DB 5,4,3 DUP(2, 3 )

.code
main proc
mov ax,@data
mov ds,ax
;Wael Melhem


ret
```

Random Access Memory

700:0100 update table list

```
0700:0100   EB 08 05 04 02 03 02 03-02 03 8C C8 8E D8 C3 90
0700:0110   90 90 90 90 90 90 90 90-90 90 90 90 90 90 90 90
0700:0120   90 90 90 F4 00 00 00 00-00 00 00 00 00 00 00 00
0700:0130   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0700:0140   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0700:0150   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0700:0160   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0700:0170   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
```

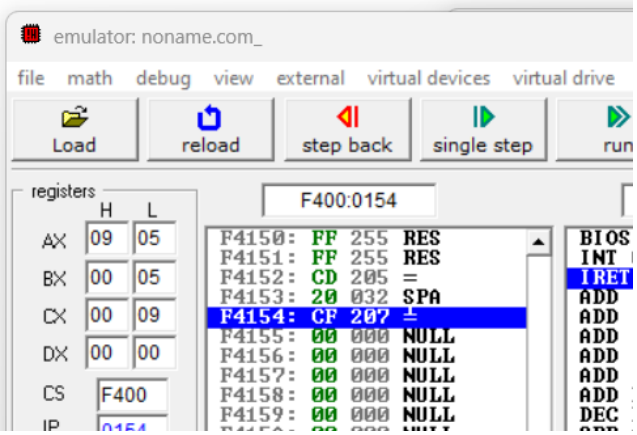
AAM instruction:

Mnemonic	Meaning	Format	Operation	O	D	I	T	S	Z	A	P	C
AAM	Adjust AL for multiplication	AAM	Convert the content of AX (a number < 100D) in two BCD digits: (AH)<-MSD; (AL)<-LSD	?				*	*	?	*	?

- It works as follows: AL is divided by 10, quotient is stored in AH & remainder in AL

Ex:

```
org 100h

;wael melhem
.code
main proc
mov al,90d
mov bl,5d
add al,bl ;al contain 5fh or 95d
          ;95 is less than 100
          ;ah is 95 95/10 =9
          ;al is the reminder =5
aam

ret
```



Ex. 1:

```
;print the Element of an array using jump or loop
org 100h

.data
arr db 2,3,7,5,8;allocate array with this value
len db $-arr     ; is the length of the array
.code
;wael melhem
main proc
    mov ax,@data
    mov ds,ax
    mov ax,0000h

    mov ah,02h ; use to print
    ;mov cx,len
    lea si,arr ;is the first location of the array
    print:
    mov dl,[si] ;dl contain the value of the current address
    add dl,'0' ;add 30h to make the current value in the ASCI
    int 21h   ;print it
    mov dl,' ';put a pace to print it
    int 21h
    inc si    ;make SI point in the next elemnt
    dec len   ;decrease the len by one because we print one item in this iter
    cmp len,0 ;compare if len is zero that main we print all elemnt or not
    jne print ;if len !0 jump to print if all elemnt printed breakl
    ;LOOP print

    ret
    main endp
end main

ret
```

emulator screen (80x25 chars)

```
2 3 7 5 8
```

Ex. 2:

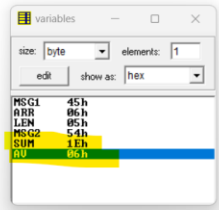
```
main proc
    mov ax,@data
    mov ds,ax
    mov ax,0000h

    mov dx,offset msg1 ;print the enter message
    call print_msg

    mov ah,01h  ; to enter
    mov cl,len  ; cl is 6 now

    lea si,arr  ; si is point to the first element in the array location
    Enter:
    int 21h     ; enter number
    sub al,30h  ; convert input from asci to hexa
    mov [si],al ; the current location of array contain the number enterd
    inc si      ;si point to the next location
    LOOP Enter  ; loop until cx =0

    call new_line    ; print new line by call the procedure
    mov dx,offset msg2 ;print the output message
    call print_msg


    mov ah,02h
    mov cl,len

    lea si,arr
    Print:
    mov dl,[si] ; dl contain the first elemnt of the array
    add dl,30h  ;convert it to asci
    int 21h     ;print it
    inc si
    LOOP Print  ; loop untl cx =0

    ret
    main endp


new_line proc
    mov ah,02h
    mov dl,0ah
    int 21h
    mov dl,0dh
    int 21h
    ret
new_line endp


print_msg proc
    mov ah,09h
    int 21h
    ret
print_msg endp

end main
```

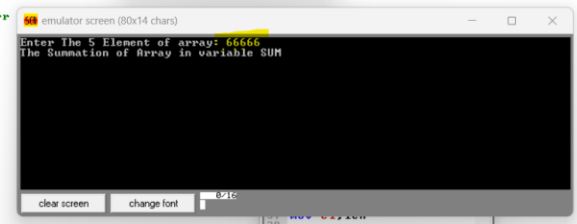
emulator screen (80x25 chars)

```
Enter The 6 Element of array: 568924
The entered Array is: 568924
```

Ex. 3:

```
;Enter 5 elements in the array and store the summation of element and Average them in variables SUM,AV respectively (using loop instructions just)
org 100h

.data
msg1 db 'Enter The 5 Element of array: $'
arr db 5 dup(?)
len db 5
msg2 db 'The Summation of Array in variable SUM$'
SUM db ?
AV db ?
.code                                   ; same the previous example
main proc
    mov ax,@data
    mov ds,ax
    mov ax,0000h

    mov dx,offset msg1
    call print_msg       ;print message

    mov ah,01h      ; to enter
    mov cl,len    ; cl is 5

    lea si,arr
    Enter:
    int 21h
    sub al,30h      ; enter 5 element and store then in arr
    mov [si],al
    inc si
    LOOP Enter

    call new_line          ; print new line
    mov dx,offset msg2    ; print the message
    call print_msg


    mov ah,02h
    mov cl,len

    lea si,arr
    mov al,00h
    summ:
    add al,[si]
    inc si
    LOOP summ      ; find the summation of the array

    mov SUM,al  ;al contain the sumation
    mov ah,00h;to initilaiz AX th same value of AL; because AX/bl
    mov bl,len
    div bl
    mov AV,al    ; the average in the av

    ret
    main endp

new_line proc
    mov ah,02h
    mov dl,0ah
```





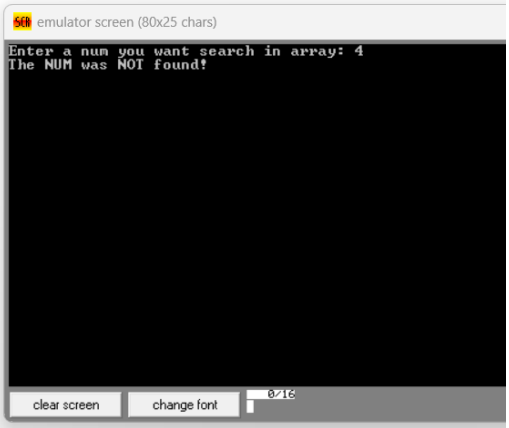
Exercise:

```
;Enter 5 elements in the array and store the summation of element and Average them in variables SUM,AV respective
org 100h

.data
msg1 db 'Enter The 5 Element of array: $'
arr db 5 dup(?)
len db 5
msg2 db 'The Summation of Array :$'
msg3 db 'The Average of Array :$'
SUM db ?
AV db ?
.code                                   ; same the previous example
main proc
    mov ax,@data
    mov ds,ax
    mov ax,0000h

    mov dx,offset msg1
    call print_msg       ;print message

    mov ah,01h      ; to enter
    mov cl,len    ; cl is 5

    lea si,arr
    Enter:
    int 21h
    sub al,30h      ; enter 5 element and store then in arr
    mov [si],al
    inc si
    LOOP Enter


    mov ah,02h
    mov cl,len

    lea si,arr
    mov al,00h
    summ:
    add al,[si]
    inc si
    LOOP summ      ; find the summation of the array

    mov SUM,al  ;al contain the sumation
    mov ah,00h;to initilaiz AX th same value of AL; because AX/bl
    mov bl,len
    div bl
    mov AV,al    ; the average in the av

    call new_line          ; print new line
    mov dx,offset msg2    ; print the message
    call print_msg

    mov al, [SUM]
    aam
    mov dl,ah
    mov cl,al
    add dl,30h
```

```
    ;wait
    mov ah,02h
    int 21h

    mov dl,cl
    add dl,30h
    int 21h

    call new_line          ; print new line
    mov dx,offset msg3    ; print the message
    call print_msg

    mov al, [AV]
    aam
    mov dl,ah
    mov cl,al
    add dl,30h

    mov ah,02h
    int 21h

    mov dl,cl
    add dl,30h
    int 21h


    ret
    main endp


new_line proc
    mov ah,02h
    mov dl,0ah
    int 21h
    mov dl,0dh
    int 21h
    ret
new_line endp


print_msg proc
    mov ah,09h
    int 21h
    ret
print_msg endp

end main
```

Ex. 4:

```
;Search the element in the array
org 100h
;wael melhem
.data
arr db 1,3,5,7,9
len db $-arr     ; we dont decrease one here
msg db 'Enter a num you want search in array: $'
msg1 db 'The NUM was found.$'            ; same pervious
msg2 db 'The NUM was NOT found!$'
.code
main proc
    mov ax,@data
    mov ds,ax
    mov ax,0000h

    mov dx,offset msg ;print message
    call print_msg

    mov ah,01h
    int 21h        ; enter the number and convert it from asci to number
    sub al,30h

    lea si,arr
    mov cl,len
    search:
    cmp [si],al  ; compare the current with the enterd value
    je FOUND       ; if equaled go to label found
    inc si
    LOOP search

    call new_line
    mov dx,offset msg2     ; if not print not found message
    call print_msg
    hlt

    FOUND:
    call new_line
    mov dx,offset msg1 ; print found message
    call print_msg

    ret
    main endp


new_line proc
    mov ah,02h
    mov dl,0ah
    int 21h
    mov dl,0dh
    int 21h
    ret
new_line endp


print_msg proc
    mov ah,09h
    int 21h
    ret
```



HW.:

```
;Search the element in the array
org 100h
;wael melhem
.data
msg1 db 'Enter the String max size is 60 chars : $'
str db 60 dup(?)
msg2 db 'The String is : $'
.code
main proc
    mov ax,@data
    mov ds,ax
    mov ax,0000h

    mov dx,offset msg1 ;print message
    call print_msg


    lea si,str
    mov ah,01h

    Enter:
    int 21h
    mov [si],al
    inc si
    cmp al,'$'
    loopne Enter

    call new_line
    mov dx,offset msg2
    call print_msg
    lea si,str
    mov ah,02h

    print:
    mov dl,[si]
    int 21h
    inc si
    cmp [si],'$'
    loopne print
    ret
    main endp


new_line proc
    mov ah,02h
    mov dl,0ah
    int 21h
    mov dl,0dh
    int 21h
    ret
new_line endp


print_msg proc
    mov ah,09h
    int 21h
    ret
print_msg endp
```