Microprocessor and Assembly Language
CSC-321

Sheeza Zaheer

Lecturer

COMSATS UNIVERSITY ISLAMABAD

LAHORE CAMPUS

Array and Array Addressing

OUTLINE

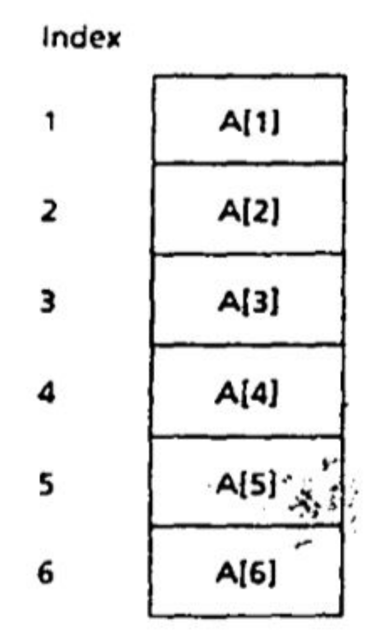
- Arrays
  - Introduction, Syntax and Examples
- Addressing Modes
  - Register Mode
  - Immediate Mode
  - Register Direct Mode
  - Register Indirect Mode

- **References**
  - **Chapter 10,** Ytha Yu and Charles Marut, “Assembly Language Programming and Organization of IBM PC

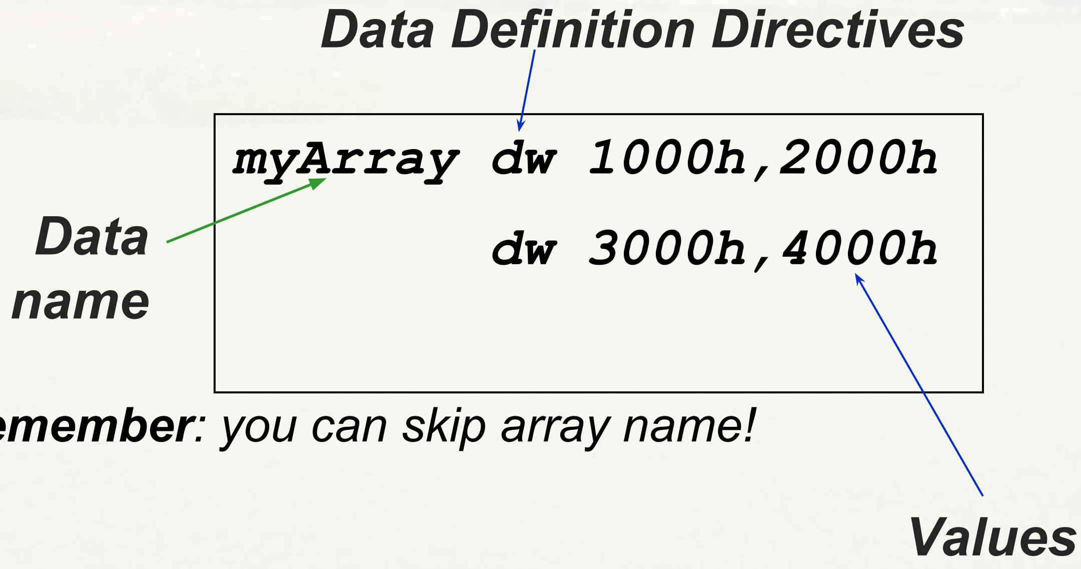
Arrays

- An ordered list of elements
- Syntax

  Array_name type value1, value2, value3
- Data Definition Directives
  - DB, DW, DD, DQ, DT



Index	
1	A[1]
2	A[2]
3	A[3]
4	A[4]
5	A[5]
6	A[6]



```
myArray dw 1000h,2000h
        dw 3000h,4000h
```

Remember: *you can skip array name!*

Contd..

Examples	Bytes	Description	Pseudo-ops
array1 DB 10, 20,30,40	1	Define Byte	**DB**
array2 DW 1000, 2000	2	Define Word	**DW**
Var3 DD -214743648	4	Define Double Word	**DD**

Arrays

- Sequence of memory bytes or words
- **Example 1:**

B_ARRAY DB 10h, 20h, 30h

Symbol	**Address**	**Contents**
B_ARRAY	0200h	10h
B_ARRAY+1	0201h	20h
B_ARRAY+2	0202h	30h

***If B_ARRAY is assigned offset address 0200h by assembler**

Example 2

- W_ARRAY DW 1000, 40, 29887, 329

If W_ARRAY is assigned offset address 0300h by assembler

Symbol	Address	Contents
W_ARRAY	0300h	1000d
W_ARRAY+ 2	0302h	40d
W_ARRAY+ 4	0304h	29887d
W_ARRAY+ 6	0306h	329d

- ***High & Low Bytes of a Word***

  WORD1 DW 1234h

- *Low Byte = 34h, symbolic address is WORD1*
- *High Byte = 12h, symbolic address is WORD1+1*

The DUP Operator

- Used to define an array if initial value of elements are same.
- Examples:

GAMMA DW 100 DUP (0) ;sets an array of 100 elements initialized to 0

DELTA DB 20 DUP (?) ; creates an array with 20 uninitialized bytes

- **Nested DUP:**

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

- Address of an array element = base address + constant
- Suppose A is an array, S denotes no. of bytes in each element (S = 1 for a byte array, S = 2 for a word array).
- Position of elements in array A:

Position	**Location**
1	A
2	A = 1 * S
.	.
.	.
N	A = (N-1) * S

Example 10.1

- Exchange the 10th and 25th elements in a word array W

- Solution:

```
MOV AX, W + 18          ; 10th element = 9 * 2
XCHG W + 48, AX         ; 25th element = 24 * 2
MOV W + 18, AX
```

ADDRESSING MODES

Default Segment and Offset Registers

- CS: IP
- SS: SP, BP
- DS: BX, DI, SI

Addressing Mode

- The way an operand is specified is known as its addressing mode.
- **Modes**

1. Register Mode (Operand is register)
2. Immediate (Operand is constant)
3. Direct (Operand is variable)
4. Indirect
    1. Register Indirect
    2. Based
    3. Indexed
    4. Based Indexed (used with 2D array)

Register Mode

- **Operand = Register**
- Can be 8 or 16 bit register
- Efficient as no memory access required.
- **Example**

```
mov ax, bx
mov cl, al
```

Immediate Mode

- **Operand = Constant Expression**
- Constant Expression can be a number or a character.
- **Example**

mov ax, 5

mov dl,’a’

Important: Destination operand cannot be in immediate mode.

Direct Mode

- Direct operand refers to the contents of memory at a location identified by the label in the data segment.

- **Example**

```
.data
count db 20
wordList dw 1000h,2000h
.code
mov al, count
mov bx, wordList
```

Contd..

```
.data
  array db 10,20,30,40
.code
  mov al, array
  mov bl, array+1
  mov cl, array+2
  mov dl, array+3
```

Offset Operator

- Used to move the offset of a label into a register or variable.
- **Example**
  - **Assume offset of aWord is 0200H**

```
.data
aWord dw 1234
.code
mov bx, offset aWord
```

Register Indirect Mode

- Register contains the offset of data in memory.
- So, the register becomes a **pointer** to the memory location.
- Syntax:

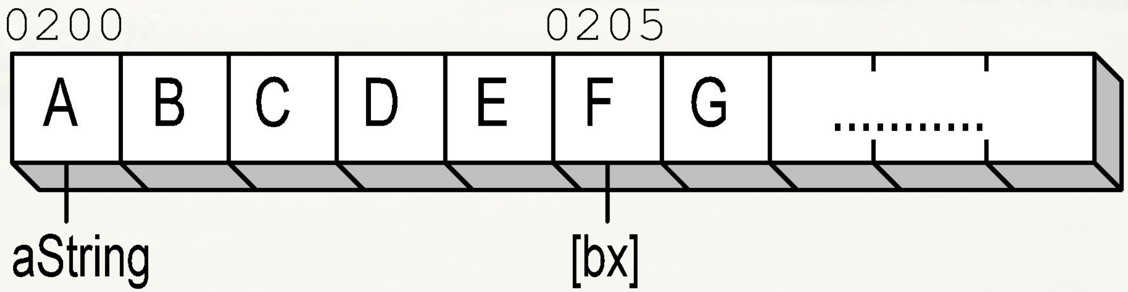
  [register]
- BX, SI, DI with default segment DS
- BP with default segment SS
- Remember the difference:

  MOV AX, SI

  MOV AX, [SI]

Example

```
.data
aString db "ABCDEFG“
.code
mov bx,offset aString
add bx,5
mov dl,[bx]
```



Also solve Example 10.2, 10.3 and 10. 4 from textbook

Based and Indexed Modes

- 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 or 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)
- Register may be: SI, DI, BX, or BP.
- Based: If BX or BP used
- Indexed: IF SI or DI used
- Can be written as:

displacement [register]

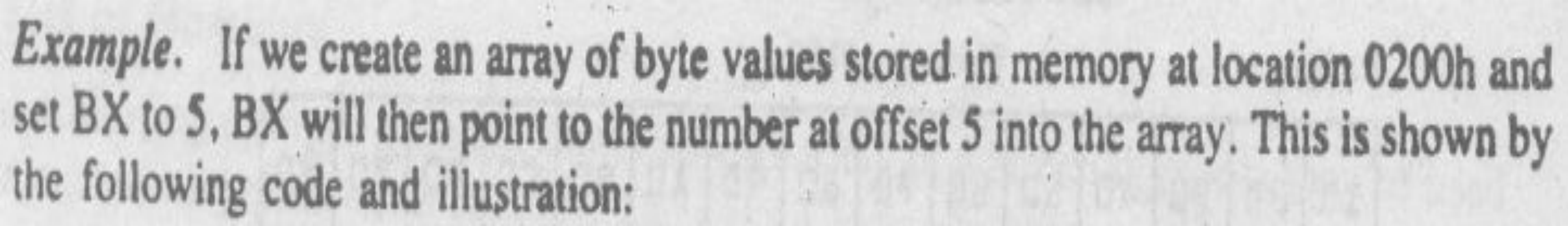
displacement + [register]

[register] + displacement

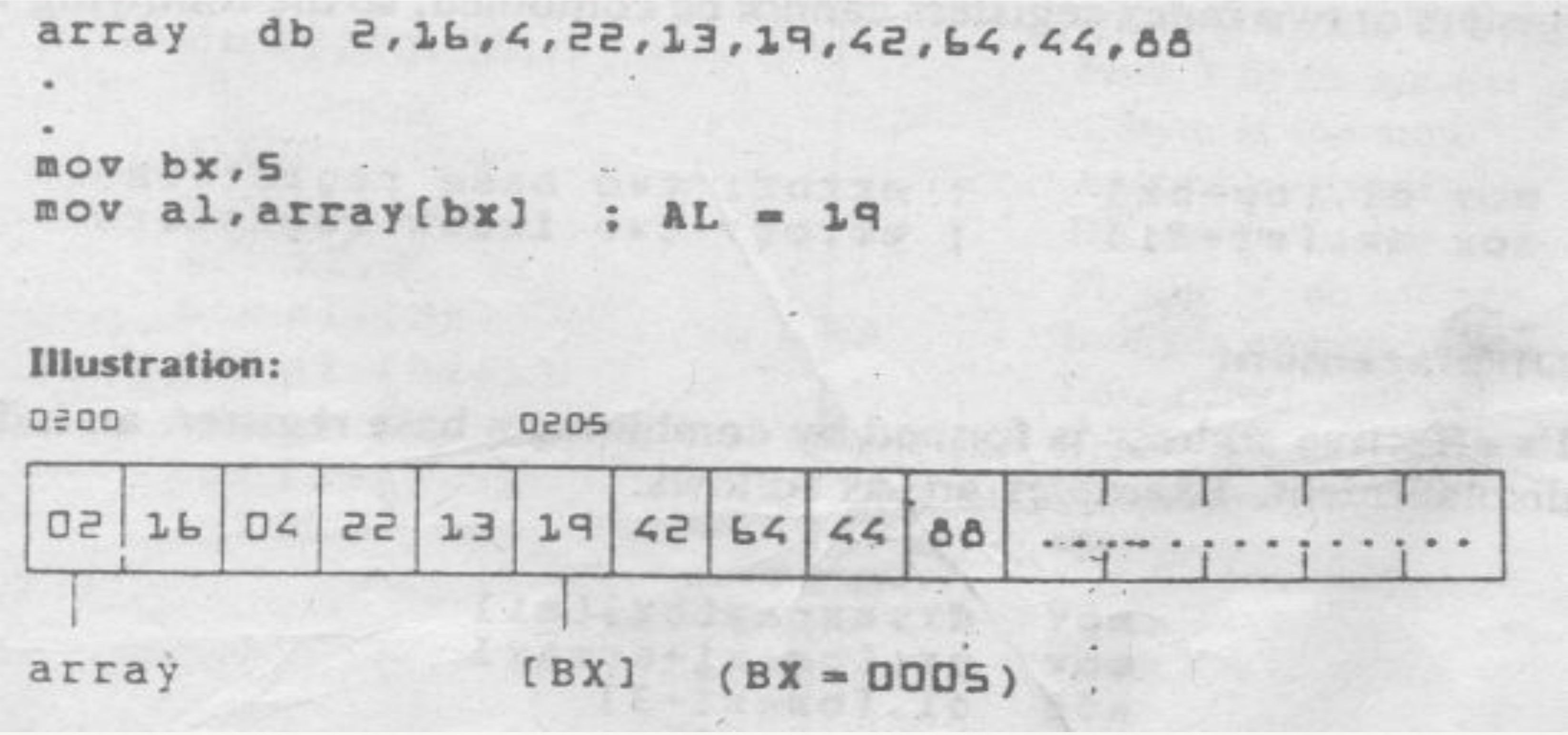
[displacement + register]

[register + displacement]

Example



Example. If we create an array of byte values stored in memory at location 0200h and set BX to 5, BX will then point to the number at offset 5 into the array. This is shown by the following code and illustration:

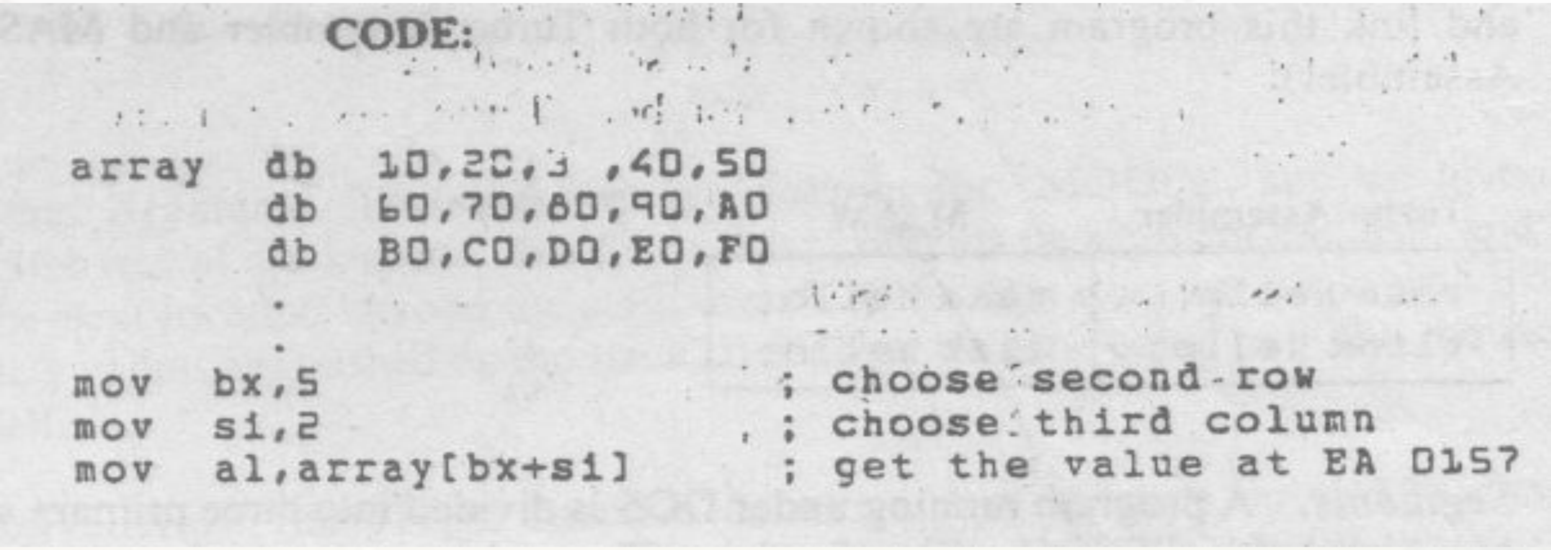


```
array   db 2,16,4,22,13,19,42,64,44,88
.
.
mov bx,5
mov al,array[bx]    ; AL = 19
```

Illustration:

0200					0205					
02	16	04	22	13	19	42	64	44	88	. . . .
array					[BX]	(BX = 0005)				

Example



CODE:

```
array   db   10,20,30,40,50
        db   60,70,80,90,A0
        db   B0,C0,D0,E0,F0
        .
        .
mov   bx,5                        ; choose second row
mov   si,2                        ; choose third column
mov   al,array[bx+si]             ; get the value at EA 0157
```

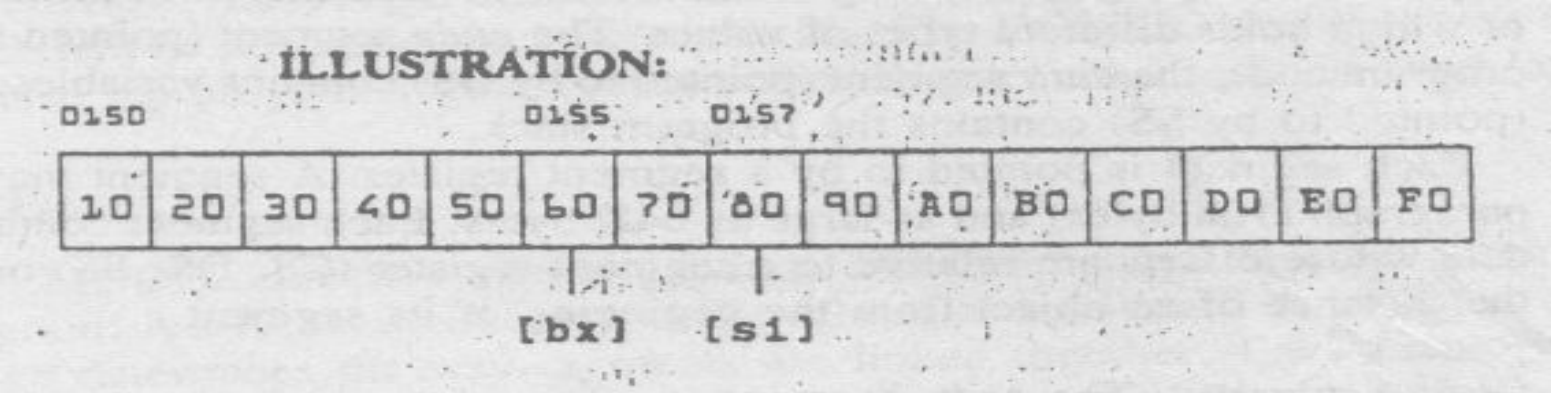


ILLUSTRATION:

0150					0155		0157							
10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
					[bx]		[si]							

Adding 8-bit Integers

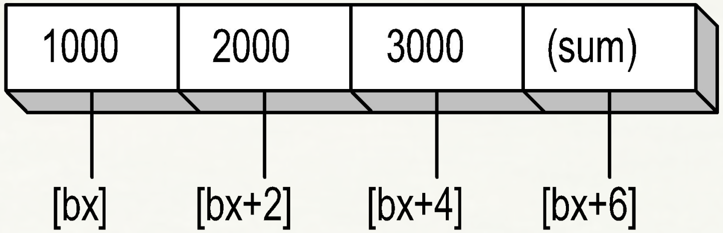
```
.data
aList db 10h,20h,30h
sum   db 0
.code
mov bx,offset aList
mov al,[bx]       ; AL = 10h
inc bx
add al,[bx]       ; AL = 30h
inc bx
add al,[bx]       ; AL = 60h
mov si,offset sum    ; get offset of sum
mov [si],al       ; store the sum
```

If you want to paste a code example such as this into a program, remember that the code segment must always begin with the following statements:

```
mov ax,@data
mov ds,ax
```

Adding 16-bit Integers

```
.data
wordList dw 1000h,2000h,3000h, 0
.code
mov bx,offset wordList
mov ax,[bx]         ; first number
add ax,[bx+2]       ; second number
add ax,[bx+4]       ; third number
mov [bx+6],ax       ; store the sum
```



Displaying a String

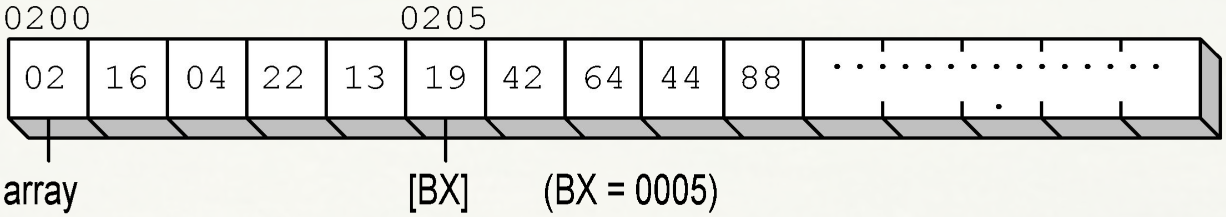
```
.data
string db "This is a string."
COUNT = ($-string)  ; calculate string length

.code
   mov   cx,COUNT   ; loop counter
   mov   si,offset string
L1:
   mov   ah,2     ; DOS function: display char
   mov   dl,[si]    ; get character from array
   int   21h        ; display it now
   inc   si       ; point to next character
   Loop  L1       ; decrement CX, repeat until 0
```

Two-Dimensional Array Example

Each row of this table contains five bytes. BX points to the beginning of the second row:

```
.data
ROWSIZE = 5
array   db  2h, 16h,  4h, 22h, 13h
        db 19h, 42h, 64h, 44h, 88h
.code
mov bx,ROWSIZE
mov al,array[bx]           ; AL = 19h
```



0200 0205

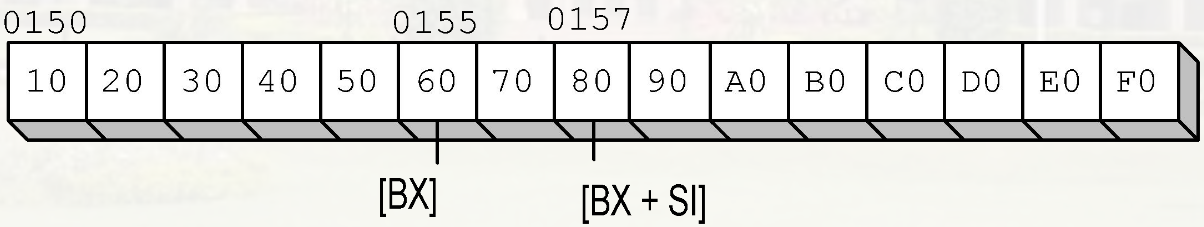
02 16 04 22 13 19 42 64 44 88 . . .

array [BX] (BX = 0005)

Based-Index Operands

Add the value of a base register to an index register, producing an effective address of 0157:

BX = 0155, SI = 0002



Example...

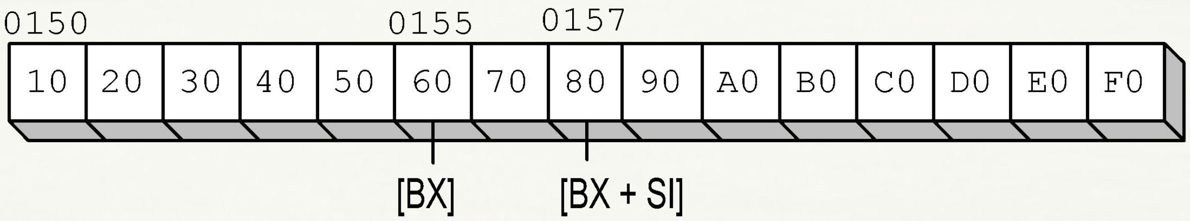
Base-Index Example

```
.data
ROWSIZE = 5
array  db  10h, 20h, 30h, 40h, 50h
       db  60h, 70h, 80h, 90h,0A0h
       db 0B0h,0C0h,0D0h,0E0h,0F0h


.code
mov  bx,offset array
; point to the array at 0150
add  bx,ROWSIZE      ; choose second row
mov  si,2          ; choose third column
mov  al,[bx + si]       ; get the value at 0157
```

Base-Index with Displacement

```
.data
ROWSIZE = 5
array db  10h, 20h, 30h, 40h, 50h
      db  60h, 70h, 80h, 90h,0A0h
      db 0B0h,0C0h,0D0h,0E0h,0F0h
.code
mov bx,ROWSIZE              ; row 1
mov si,2                    ; column 2
mov dl,array[bx + si]       ; DL = 80h
```


0150
0155
0157
10
20
30
40
50
60
70
80
90
A0
B0
C0
D0
E0
F0
[BX]
[BX + SI]