CHAPTER 10

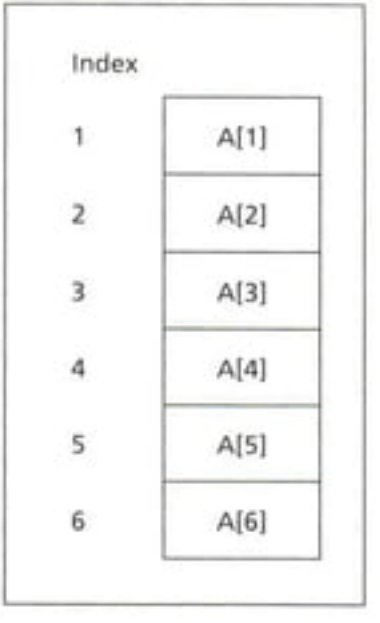
ARRAYS AND ADDRESSING MODES

Outline

- One-Dimensional Arrays
- Addressing Modes
- Two-Dimensional Arrays
- Based Indexed Addressing Mode

One-Dimensional Array A

- One-Dimensional Array is an ordered list of elements, all of same type
- DB & DW pseudo-ops to declare byte and word sized arrays



Arrays

```
W    DW        1000,40,29887,329
```

Address of the array variable is called ***base address of array***

Offset Address	Symbolic Address	Contents
0300h	W	1000d
0302h	W +2	40d
0304h	W +4	29887d
0306h	W +6	329d

The DUP Operator

- The DUP (duplicate) is used to define arrays whose elements share a common initial value.
- **repeat_count DUP (value)**
- GAMMA DW 100 DUP (0)
- DELTA DB 212 DUP (?)
- LINE DB 5, 4, 3 DUP (2, 3 DUP (0), 1)
- LINE DB 5,4,2,0,0,0,1,2,0,0,0,1,2,0,0,0,1

One-Dimensional Array A

W DW 10, 20, 30, 40, 50, 60

Offset address	Symbolic address	Decimal address
0200h	W	10
0202h	W + 2h	20
0204h	W + 4h	30
0206h	W + 6h	40
0208h	W + 8h	50
020Ah	W + Ah	60

Location of Array Elements

- The address of an array element can be specified by adding a constant to the base address
  - If A is an array
  - S is the size of every element in bytes

(S= 1 FOR BYTE ARRAY, S = 2 FOR WORD ARRAY)

Position	Location
1	A
2	A + 1 x S
3	A + 2 x S
.	
.	
N	A + (N-1) x S

Example: Exchange 10^{th} and 25^{th} element of an array

- W[10] is located at W + 9x2 = W + 18
- W[25] is located at W + 24x2 = W + 48

```
MOV AX, W + 18
XCHG W+48, AX
MOV W+18, AX
```

Addressing Modes

- The way an operand is specified
- **register mode**: an operand is a register.
- **immediate mode**: an operand is a constant.
- **direct mode**: an operand is a variable.

  - MOV AX, 0
  - ADD ALPHA, AX

ADDITIONAL ADDRESSING MODES

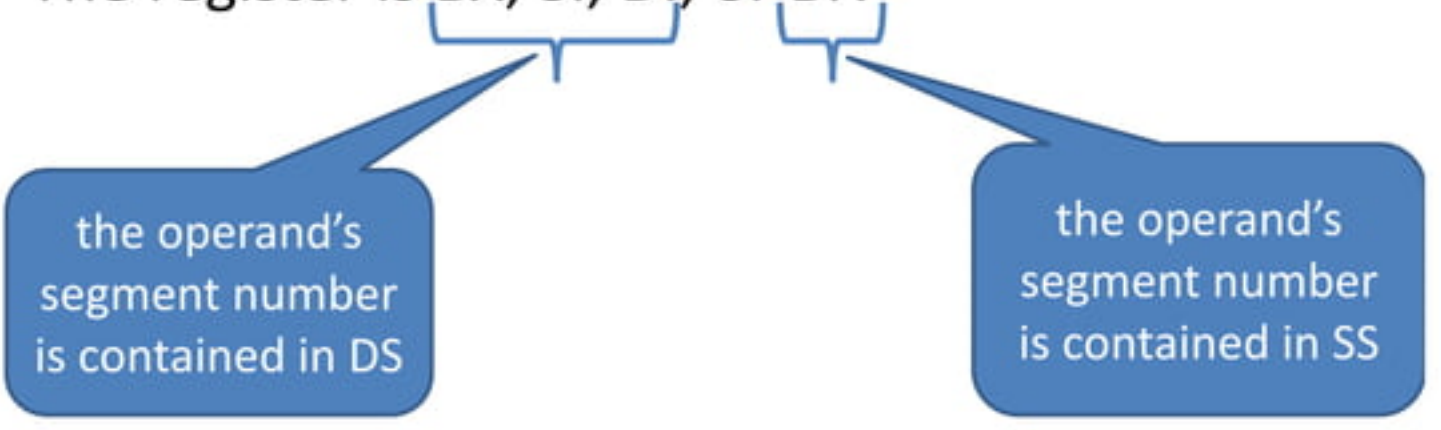
Four additional addressing modes to address memory operands indirectly

1. Register Indirect Mode
2. Based
3. Indexed
4. Based Indexed

Register Indirect Mode

Offset address of the operand is contained in a register. Register acts like a pointer to the memory location

- [register]
- The register is BX, SI, DI, or BP.



Suppose that SI contains 0100h, and the word at 0100h contains 1234h.

- MOV AX, [SI] ; AX = 1234h

  The CPU

  1. examines SI and obtains the offset address 100h,
  2. uses the address DS:0100h to obtain the value 1234h, and
  3. moves 1234h to AX.

- MOV AX, SI ; AX = 0100h

Suppose that

BX contains 1000h — Offset 1000h contains 1BACh

SI contains 2000h — Offset 2000h contains 20FEh

DI contains 3000h — Offset 3000h contains 031Dh

where the above offsets are in the data segment addressed by DS.

Tell which of the following instructions are legal. If legal, give the source offset address and the result or number moved.

		Source offset	Result
a.	MOV BX, [BX]	1000h	1BACh
b.	MOV CX, [SI]	2000h	20FEh
c.	MOV BX, [AX]	illegal source register	
d.	ADD [SI], [DI]	illegal memory-memory addition	
e.	INC [DI]	3000h	031Eh

Write some code to sum in AX the elements of the 10-element array W defined by

```
W    DW   10,20,30,40,50,60,70,80,90,100
```

The idea is to set a pointer to the base of the array, and let it move up the array, summing elements as it goes.

```
        XOR   AX, AX; AX holds sum
        LEA   SI, W          ; SI points to array W
        MOV   CX, 10         ; CX has number of elements
ADDNOS:
        ADD   AX, [SI]       ; sum = sum + element
        ADD   SI, 2          ; move pointer to
                             ; the next element
        LOOP  ADDNOS         ; loop until done
```

Home Assignment

- Write a procedure to reverse an array of N words

Based and Indexed Addressing Mode

- Operand's offset address is obtained by adding a number called ***displacement*** to the contents of a register

- ***Displacement*** can be of following forms;
  - Offset address of a variable — A
  - A constant (+ve or -ve) — -2
  - Offset address of a variable plus or minus a constant — A + 4
- If A is a variable;

Based and Indexed Addressing Mode

- [register + displacement]
- [displacement + register]
- [register] + displacement
- displacement + [register]
- displacement[register]
- The register is SI, DI, BX, or BP.

Based and Indexed Addressing Mode

Suppose W is a word array; BX contains 4

```
MOV     AX, W[BX]
```

Will move the element at address W + 4 to AX (the third element of array)

```
MOV     AX, [W + BX]
MOV     AX, [BX + W]
MOV     AX, W + [BX]
MOV     AX, [BX] + W
```

Rework the last example by using based mode.

```
        XOR   AX, AX        ; AX holds sum
        XOR   BX, BX        ; clear base register
        MOV   CX, 10        ; CX has number of elements
ADDNOS:
        ADD   AX, W[BX]     ; sum = sum + element
        ADD   BX, 2         ; index next element
        LOOP  ADDNOS        ; loop until done
```

Suppose that ALPHA is declared as

ALPHA DW 0123H, 0456h, 0789h, 0ABCDh

in the segment addressed by DS.

Suppose also that

BX contains 2 Offset 0002 contains 1084h

SI contains 4 Offset 0004 contains 2BACh

DI contains 1

Tell which of the following instructions are legal. If legal, give the source offset address and the result or number moved.

		Source offset	Number moved
a.	MOV AX, [ALPHA+BX]		
b.	MOV BX, [BX+2]		
c.	MOV CX, ALPHA[SI]		
d.	MOV AX, –2[SI]		
e.	MOV BX, [ALPHA+3+DI]		
f.	MOV AX, [BX] 2		
g.	ADD BX, [ALPHA+AX]		

PTR OPERATOR

- Operands of instruction must be of the same type

```
MOV AX, 1  ; legal word instruction
MOV BH, 5; legal byte instruction
MOV [BX], 1  ; illegal cant interpret whether destination is a
                ;byte operand pointed by bx or a word
```

- For destination to be byte

```
MOV BYTE PTR [BX], 1
```

- For destination to be word

```
MOV WORD PTR [BX], 1
```

Using PTR to Override a type

```
type PTR address_expression
```

If we delare

```
DOLLARS DB 1AH
CENTS DB 52H

MOV AX, DOLLARS ; ILLEGAL

MOV AX, WORD PTR DOLLARS; AL= DOLLARS, AH = CENTS
```

;will move 521AH to AX

Label pseudo op

- To get around problem of type conflict;

```
MONEY LABEL WORD
DOLLARS DB 1AH
CENTS DB 52H
```

- Declares MONEY as a word variable and components DOLLARS and CENTS as byte
- `MOV AX, MONEY ; LEGAL`
- `MOV AL, DOLLARS ; LEGAL`
- `MOV AH, CENTS ; LEGAL`

SEGMENT OVERRIDE

- BX, SI, DI specify offset relative to DS
- Possible to specify offset relative to other segments

```
Segment_register : [pointer_register]
```

```
MOV AX, ES:[SI]
```

CAN also be used with based and indexed addressing modes

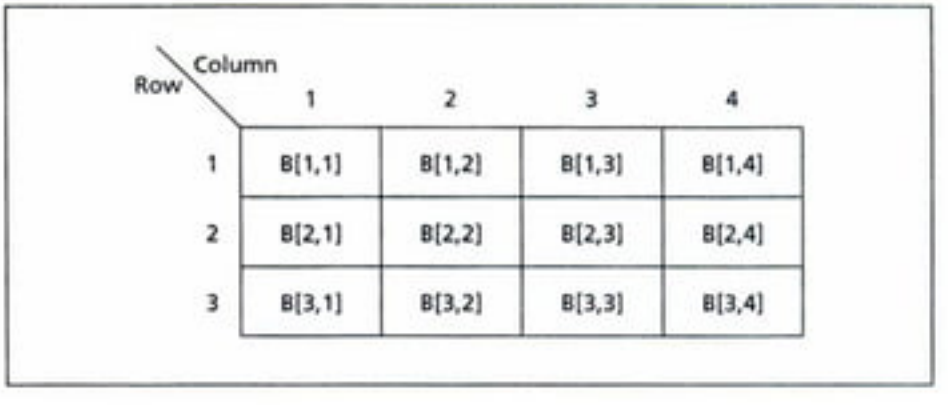
Accessing the STACK

- When BP specifies the offset address, SS supplies the segment number
- BP may be used to access elements of the stack
- Move top three elements of stack to AX, BX, CX without changing the stack

```
MOV BP, SP
MOV AX, [BP]
MOV BX, [BP + 2]
MOV CX, [BP + 4]
```

Two-Dimensional Array B

- An array of arrays
- A One Dimensional Array whose elements are also One Dimensional Array
- Arranged as rows and cloumns



How Two-Dimensional Array are stored

Suppose array B has

10, 20, 30, 40 in the first row,

50, 60, 70, 80 in the 2nd row,

&

90, 100, 110, 120 in the 3rd row

Row-Major Order

```
B    DW   10, 20, 30, 40
     DW   50, 60, 70, 80
     DW   90, 100, 110, 120
```

Used when elements in a row are to be processed together sequentially

Column-Major Order

```
B    DW   10, 50, 90
     DW   20, 60, 100
     DW   30, 70, 110
     DW   40, 80, 120
```

Used when elements in a column are to be processed together sequentially

Locating an element in an array

- Consider *M xN* array stored in row major order
- Size of element is *S*
- *To find location of A[I,j]*
- Find where the row *i* begins
- Location of jth element in that row
  - Row 1 begins at A
  - Row 2 begins at A + *N x S*
  - Row 3 begins at A + 2 x *N x S*
  - Row i begins at A + (*i-1*) x *N x S*

jth element in a row is located at *(j-1) x S*

Final Result

Location of A[i,j]

$A + \{ (i-1) \times N + (j - 1)\} \times S$

- For column major ordered array;

Location of A[i,j]

$A + \{ (i-1) + (j - 1) \times M\} \times S$

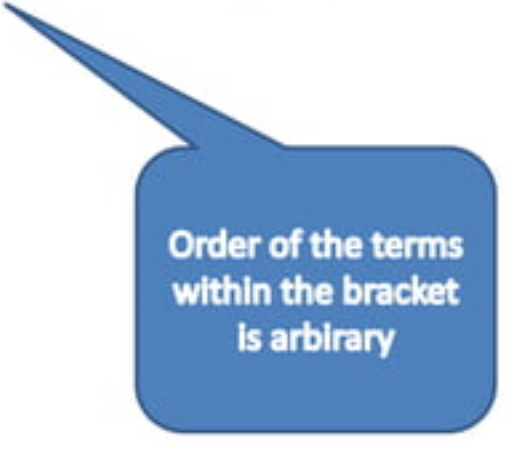
Based Indexed Addressing Mode

- Offset address of the operand is the sum of;
  - Contents of a base register (BX or BP)
  - Contents of an index register (DI or SI)
  - Optionally a variable's offset address
  - Optionally a constant (+ve or -ve)

If BX is used DS is the segment ragister, If BP is used, SS is the segment register

Based Indexed Addressing Mode

- variable [base_register][index_register]
- [base_register + index_register + variable + constant]
- variable [base_register + index_register + constant]
- constant [base_register + index_register + variable]



Based Indexed Addressing Mode

- W is a word variable
- BX contains 2
- Si contains 4

```
MOV AX, W[BX][SI]
```

Will move contents of W + 6 to AX

```
MOV AX, [W + BX + SI]
MOV AX, W[BX + SI]
```

- M is a 5x7 word array stored in row major order, write some code to;
  - Clear row 3
  - Clear column 4

1. Clear row 3

For an MxN array; Row i begins at A + (*I-1*) x N x *S*

Thus in a 5x7 array, row 3 begins at; A + (*3-1*) x7x2 = *A + 28;*

```
   MOV BX, 28
   XOR SI, SI
   MOV CX, 7
CLEAR:
   MOV A[BX][SI], 0
   ADD SI,2
   LOOP CLEAR
```

- M is a 5x7 word array stored in row major order, write some code to;
  - Clear row 3
  - Clear column 4

2. Clear column 4

For an MxN array; column j begins at A + (j-*1*) *S*

Column 4 begins at A + (*4-1*) *x 2 = A + 6;*

Since A is a 7 column array stored in row major order, to get to the next element in column 4 we need to add 7x 2 = 14 ;

```
    MOV SI, 6
    XOR BX, BX
    MOV CX, 5
CLEAR:
    MOV A[BX][SI], 0
    ADD BX,14
    LOOP CLEAR
```

An application: Average Test Scores

```
sum[j] = 0
i = 1
FOR  5  times  DO
   sum[j] = sum[j] + score[i, j]
   i = i + 1
END_FOR
```