Assembly : Arrays, Addressing Modes and String Instructions

Author	Md. Zarif Ul Alam
Date Created	@March 22, 2021 6:50 PM
Tag	CSE315 Microprocessors Microcontrollers and Embedded Systems

1D Array

Declaration

```
ARA DW 10,20,30,40

; ARA      -> 10
; ARA + 2H -> 20
---
```

```
; syntax : repeat_count DUP(value)

ARA DW 4 DUP(0) ; 0 0 0 0
ARA DW 4 DUP(?) ; ? ? ? ?

; nested DUP

LINE DB 5,4,3 DUP(2,3 DUP(0),1)

; 5,4,3 DUP(2,3 DUP(0),1)
; 5,4,3 DUP(2,0,0,0,1)
; 5,4,3,2,0,0,0,1,2,0,0,0,1,2,0,0,0,1
```

2D Array

Declaration

```
ARA DW 10,20
    DW 40,50
```

Location

Locating an element in a 2d array

- Suppose an M by N array A is stored in a row major order
- Size of the elements is S (S=1 for byte array and 2 for word array)
- To find the location of j-th element in the i-th row
  - Find where row i begins
  - Find the location of the j-th element in that row
- A[i,j] has address

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

Addressing mode

register indirect mode

- `offset` stored in register
- format
  - $[register]$
- register : BX, SI, DI, BP
- if BX, SI, DI contains offset , segment number in DS
- if BP contains offset, segment number in SS

Write a 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

```
        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
```

based mode

- format
  - $[register + displacement]$

- $[displacement + register]$
- $[register] + displacement$
- $displacement + [register]$
- $displacement\ [register]$

- register : BX, BP
- displacement

- Displacement can be
  - the offset address of a variable (e.g., A)
  - a constant (positive or negative) (e.g., -2)
  - the offset address of a variable plus or minus a constant (A + 2)

- If BX is used as register, DS contains the segment number
- If BP is used as register, SS contains the segment number

- Suppose W is a word array and BX contains 4
- Displacement is the offset address of variable W.
- So, to move the 3rd element of the array to AX, we can write

```
MOV AX, [BX + W]      ;[register + displacement]
MOV AX, [W + BX]      ;[displacement + register]
MOV AX, [BX] + W      ;[register] + displacement
MOV AX, W + [BX]      ;displacement + [register]
MOV AX, W [BX]        ;displacement [register]
```

indexed mode

- format
  - $[register + displacement]$
  - $[displacement + register]$
  - $[register] + displacement$
  - $displacement + [register]$
  - $displacement\ [register]$
- register : SI, DI
- displacement

- Displacement can be
  - the offset address of a variable (e.g., A)
  - a constant (positive or negative) (e.g., -2)
  - the offset address of a variable plus or minus a constant (A + 2)

- If SI, DI is used as register, DS contains the segment number

based indexed mode

- format
  - $variable[baseregister][indexregister]$
  - $[baseregister + indexregister + variable + constant]$
  - $variable[baseregister + indexregister + constant]$
  - $constant[baseregister + indexregister + variable]$
- base register : BX, BP
- index register : SI, DI

Suppose A is 5 by 7 word array stored in row major order. Write a code to clear the 3rd row using based indexed mode

```
        MOV     BX, 28      ; BX indexes row 3
        XOR     SI, SI      ; SI will index columns

        MOV     CX, 7       ; number of elements in row
CLEAR:
        MOV     A[BX] [SI], 0           ; clear third row, element j
        ADD     SI, 2       ; go to next column

        LOOP    CLEAR       ; loop until done
```

XLAT

AL = BX[AL]

XLAT/XLATB--Table Look-up Translation

Opcode	Instruction	Description
D7	XLAT *m8*	Set AL to memory byte DS:[(E)BX + unsigned AL]
D7	XLATB	Set AL to memory byte DS:[(E)BX + unsigned AL]

- XLAT can be used to convert a byte value into another value that comes from a table
  - The byte to be converted must be in AL
  - BX must have the offset address of the conversion table

```
TABLE   DB 030h,031h,032h,033h,034h,035,036h,037h,038h,039h
        DB 041h,042h,043h,044h,045h,046h

        MOV  AL,0Ch             ;number to convert
        LEA  BX,TABLE           ;BX has table offset
        XLAT                    ;AL has 'C'
```