**Lab Session 05**

**Objectives**

**Arrays & Loop**

1. Indexed operands
2. Pointers
3. JMP instruction
4. LOOP instruction



1. **Indexed Operator**
2. **Example in 16-bit mode**

**SYNTAX:**

**constant [ reg16 ] ; reg16 can either be SI, DI, BX or BP**

**[ constant + reg16 ]**

**EXAMPLE:**

.data

arrayB db 20, 40, 60, 80

arrayW dw 100, 150, 250, 300

.code

mov si, 1 ; SI = 0001

mov al, arrayB[si] ; AL = 40

mov al, [arrayB + 3] ; AL = 80

mov si, 2 ; SI = 2

mov cx, arrayW[si] ; CX = 150

mov cx, [arrayW + 4] ; CX = 250

1. **Example in 32-bit mode**

**SYNTAX:**

**constant [ reg32 ] ;reg32 can be any of the 32-bit general registers**

**[ constant + reg32 ]**

**EXAMPLE:**

.data

arrayB BYTE 20, 40, 60, 80

.code

mov esi, 0

mov al, arrayB[esi]

inc esi

mov al, arrayB[esi]

mov esi, 3

mov al, [arrayB + esi]

1. **Example using scale factors**

**SYNTAX:**

**constant [ reg32 \* TYPE constant]**

**EXAMPLE:**

.data

arrayW WORD 1000, 2000, 3000, 4000

.code

main PROC

mov ax, arrayW

mov esi, 1

mov ax, arrayW[esi \* TYPE arrayW]

mov esi, 2

mov ax, arrayW[esi \* TYPE arrayW]

mov esi, 3

mov ax, arrayW[esi \* TYPE arrayW]

call DumpRegs

1. **Pointers**

**SYNTAX:**

**constant1 TYPE OFFSET constant2**

**EXAMPLE:**

.data

arrayW WORD 1000, 2000, 3000, 4000

ptrW DWORD OFFSET arrayW

.code

main PROC

mov eax, [ptrW]

1. **JMP instruction**

**SYNTAX:**

**JMP destination**

**EXAMPLE:**

.code

top:

; any statements

jmp top

1. **LOOP instruction**

**SYNTAX:**

**LOOP destination**

**EXAMPLE # 01:**

.data

intArray WORD 100, 200, 300, 400, 500

.code

main PROC

mov esi, 0

mov ax, 0

mov ecx, LENGTHOF intArray

L1:

mov ax, intArray [esi]

add esi, TYPE intArray

loop L1

**EXAMPLE # 02:**

mov eax, 0

mov ebx, 0

mov ecx, 5

L1:

inc eax

mov edx, ecx

mov ecx, 10

L2:

inc ebx

loop L2

mov ecx, edx

loop L1

**ACTIVITIES:**

1. Use following array declarations:  
arrayB BYTE 60, 70, 80  
arrayW WORD 150, 250, 350  
arrayD DWORD 600, 1200, 1800

For each array, add its 1st and last element using scale factors and display the result in a separate register.

2. Initialize an array:

arr DWORD 1000, 2000, 4000, 6000

Initialize four different pointer variables with each of the elements of this array.

3. Use a loop with direct or indirect addressing to reverse the elements of an integer array in place. Do not copy elements to any other array. Use SIZEOF, TYPE and LENGTHOF operators to make program flexible.

4. Write a program that uses a loop to calculate the first ten numbers of Fibonacci sequence.

5. Write a program to sort the following array using Bubble Sort algorithm:

myArray BYTE 15, 10, 25, 20, 30