Roll No: 20P-0101

Name: Muhammad Sherjeel Akhtar

Subject: Computer Organization And Assembly Language

Lab Report: 4

Submitted To Respected Sir: Usman Abbasi

Addressing Modes:

**Direct Addressing*

Indirect Addressing

Register + Offset Addressing

Direct Addressing:

In this we simply give the address of our memory location.

Code:

```
first: dw 5
second: dw 15
subtraction: dw 0

1. mov ax, [first]
2. mov bx, [second]
3. sub ax, bx
4. mov [subtraction], ax
```

Indirect Addressing:

In indirect addressing, we use registers instead of variables.

Code:

```
num1: dw 1, 2, 3, 4, 5
addition: dw 0

1. mov bx, num1
2. mov cx, 5
3. mov ax, 0

4.loop:

5. add ax, [bx]
6. add bx, 2
7. sub cx, 1

8. jne loop:

9. mov [sum], ax
```

Register + Offset Addressing:

In register + offset addressing, different combinations of direct and indirect addressing are used.

Code:

```
num1: dw 1, 2, 3, 4, 5
sum: dw 0

1. mov bx, 0
2. mov cx, 5
3. mov ax, 0

4.loop:

5. add ax, [num1+bx]
6. add bx, 2
7. sub cx, 1

8.jnz loop:

9. mov [sum], ax
```

Branching:

Two types of branching which are as follow:

1- Conditional branching

2- Unconditional branching

Unconditional Branching:

Code:

```
[org 0x0100]

    jmp start        ; unconditional jump

    num1: dw   10, 20, 30, 40, 50, 10, 20, 30, 40, 50
    result: dw 0



    start:
    ; initialize stuff
    mov  ax, 0
```

Conditional branching:

The most common way to transfer control in assembly language is to use a conditional jump.

Code:

```
start:
; initialize stuff
mov  ax, 0
mov  bx, 0

outerloop:
    add  ax, [num1 + bx]
    add  bx, 2

    cmp bx, 20          ; Conditional jump
    jne  outerloop


mov  [result], ax

mov  ax, 0x4c00
int  0x21
```

Sorting Code in Assembly Language:

The code for sorting values in ascending order is:

Code:

```
jmp start                       ;unconditional

data: dw   6, 4, 5, 2


start:
    mov  cx, 4

    outerloop:
        mov  bx, 0

        innerloop:
            mov  ax, [data + bx]
            cmp  ax, [data + bx + 2]

            jbe  noswap


                ; the swap potion
                mov  dx, [data + bx + 2]
                mov  [data + bx + 2], ax
                mov  [data + bx], dx

            noswap:
            add  bx, 2
            cmp  bx, 6
            jne  innerloop

        ; check outer loop termination
        sub cx, 1
        jnz outerloop


    ; exit system call
    mov  ax, 0x4c00
    int  0x21
```

```
AX 0005   SI 0000   CS 19F5   IP 0135     Stack
BX 0006   DI 0000   DS 19F5
CX 0001   BP 0000   ES 19F5   HS 19F5
DX 0002   SP FFFE   SS 19F5   FS 19F5

CMD >

0131 81E90100        SUB    CX,0001
0135 75D7            JNZ    010E
0137 B8004C          MOV    AX,4C00
013A CD21            INT    21
013C C746DC0000      MOV    [BP-24],0000
0141 8E5EFC          MOV    DS,[BP-04]
0144 837D0E00        CMP    [DI+0E],0000
0148 7409            JZ     0153
014A 8B46F2          MOV    AX,[BP-0E]
```