COAL_A_p200165_R4

In the Lab four we have studied about the **direct, indirect, register offset** addressing, **branching conditions** and many other conditions.

1. **Direct Addressing:**

Direct Addressing is that we directly use variables(labels) for addressing and access the value using that labels.

2. **Indirect or Register Based Addressing:**

In Indirect addressing we move the address of labels into our register and use the registers to access different values from registers.

3. **Register + Offset Addressing:**

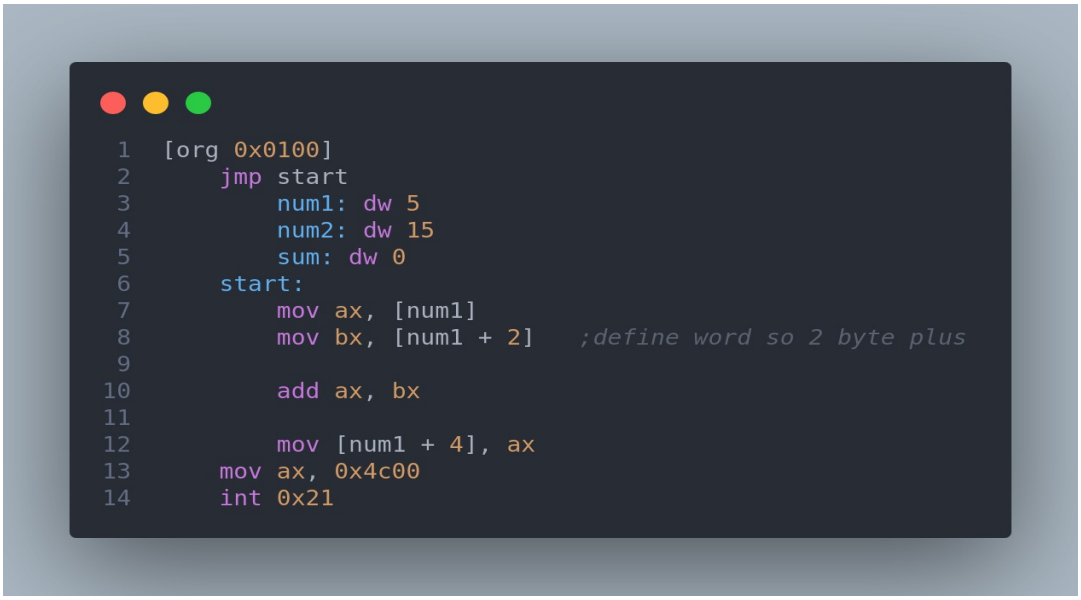
Different combinations of direct and Indirect addressing are used.

Example1

```
1   [org 0x0100]
2         jmp start
3               num1: dw 5
4               num2: dw 15
5               sum: dw 0
6         start
7               mov ax, [num1]
8               mov bx, [num2]
9
10              add ax, bx
11
12              mov [sum], ax
13        mov ax, 0x4c00
14        int 0x21
```

jmp instruction will jump the given address. Jmp is called unconditional jump. This is the example of direct addressing.

Example2



```
1  [org 0x0100]
2      jmp start
3          num1: dw 5
4          num2: dw 15
5          sum: dw 0
6      start:
7          mov ax, [num1]
8          mov bx, [num1 + 2]   ;define word so 2 byte plus
9
10         add ax, bx
11
12         mov [num1 + 4], ax
13     mov ax, 0x4c00
14     int 0x21
```

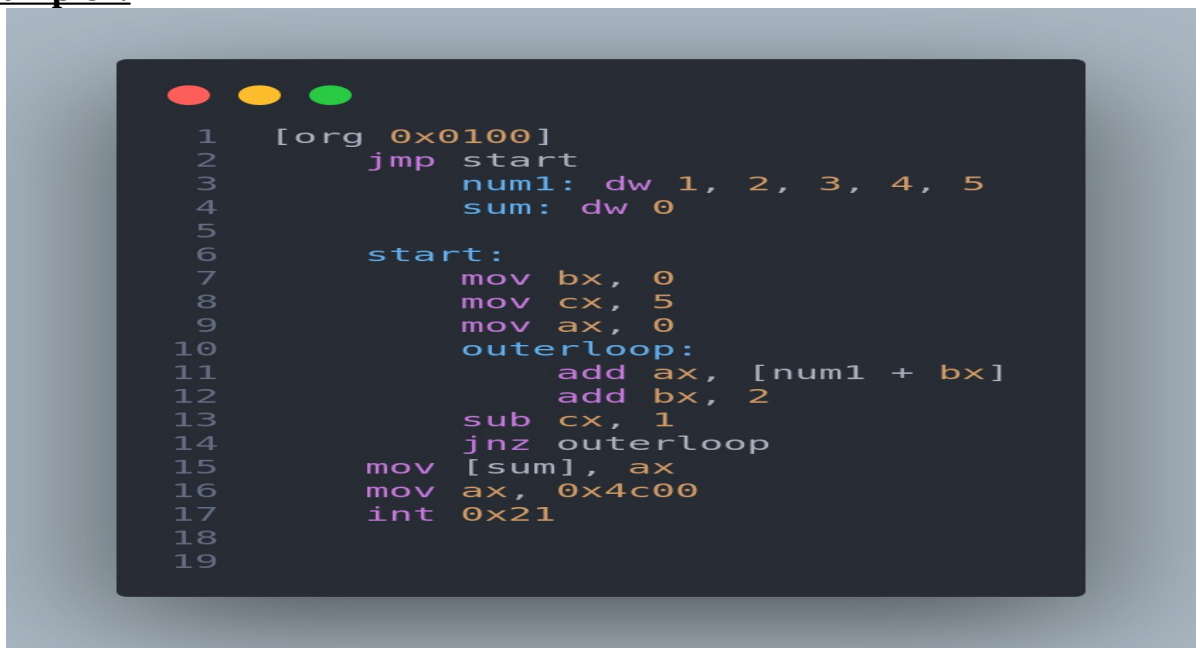
dw mean define word(2 bytes allocated in memory).

Example3:

```
1   [org 0x0100]
2       jmp start
3           num1: dw 1, 2, 3, 4, 5
4           sum: dw 0
5
6       start:
7           mov bx, num1
8           mov cx, 5
9           mov ax, 0
10          outerloop:
11              add ax, [bx]
12              add bx, 2
13          sub cx, 1
14          jnz outerloop
15      mov [sum], ax
16      mov ax, 0x4c00
17      int 0x21
18
19
```

jnz means that if zero flag is not set then jump to the given address. In this we have created the loop using the **jnz** instruction. This loop will run for 5 times because the cx(counter register) is 5. When cx react to 0 loop will exit. At each iteration all numbers will be added to ax register and at outside the loop the result moved from ax to sum label.

Example4:



```
1   [org 0x0100]
2       jmp start
3           num1: dw 1, 2, 3, 4, 5
4           sum: dw 0
5
6       start:
7           mov bx, 0
8           mov cx, 5
9           mov ax, 0
10          outerloop:
11              add ax, [num1 + bx]
12              add bx, 2
13          sub cx, 1
14          jnz outerloop
15      mov [sum], ax
16      mov ax, 0x4c00
17      int 0x21
18
19
```

This will work same as upper loop. The difference is that we are using direct addressing in this.(Label is used to get data from Ram)

Name : Jawad Ahmed
Roll No: 20P-0165
Section : 3A