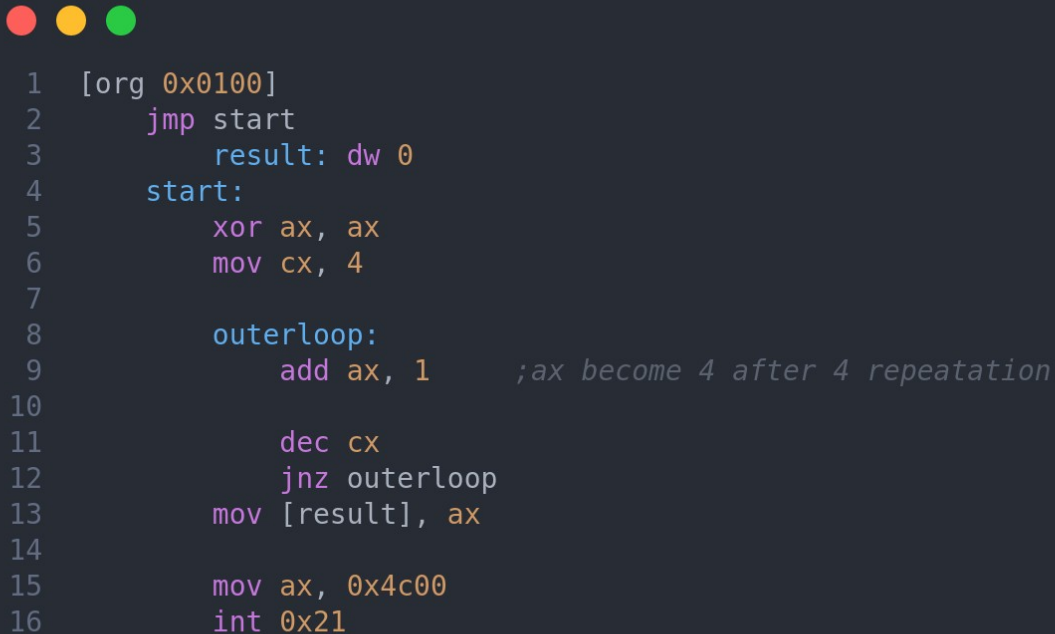


COAL_A_p200165_R5

➤ Simple Loop Examples



```
1  [org 0x0100]
2      jmp start
3      result: dw 0
4      start:
5          xor ax, ax
6          mov cx, 4
7
8          outerloop:
9              add ax, 1      ;ax become 4 after 4 repeation
10
11              dec cx
12              jnz outerloop
13          mov [result], ax
14
15          mov ax, 0x4c00
16          int 0x21
```

- Here is the simple loop in which we add 1 in ax register 4 times or whatever the value of cx(counter) register.



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

- In the Second Example, This loop works similar but this is picking up the data from ram and also writing that data to the RAM.

```
1 [org 0x0100]
2     jmp start
3     numbers: dw 5, 1, 1, 2
4     result: dw 0
5     start:
6         xor ax, ax
7         xor bx, bx
8         xor cx, cx
9
10        mov cx, 4
11
12        outerloop:
13            add ax, [numbers + bx]
14            add bx, 2
15
16            dec cx
17            jnz outerloop
18            mov [result], ax    ; sum of 4 numbers moved to result label
19
20            mov ax, 0x4c00
21            int 0x21
22
```

- In the third example we added the number (5, 1, 1, 2) all these numbers using the loop.

➤ Nested Loops

```
1 [org 0x0100]
2
3     jmp start
4
5     data: dw 6, 2, 4, 5
6     swap: db 0 ; use this as a flag
7
8     start:
9         mov cx, 4 ; make 10 passes, has to be outside the loop!
10
11        outerloop:
12            mov bx, 0
13            mov byte [swap], 0 ; why the "byte"?
14
15            innerloop:
16                mov ax, [data + bx]
17                cmp ax, [data + bx + 2] ; why did we move the value to AX?
18
19                jbe noswap ; if we don't have to swap, we just jump over the swap thing
20
21                ; the swap portion
22                mov dx, [data + bx + 2]
23                mov [data + bx + 2], ax ; again with the AX?
24                mov [data + bx], dx
25                mov byte [swap], 1
26
27            noswap:
28                add bx, 2
29                cmp bx, 6
30                jne innerloop
31
32            ; if we didn't swap even once, we should be done
33            cmp byte [swap], 1 ; don't need to load this in register?
34            je outerloop
35
36            ; check outer loop termination
37            dec cx, 1
38            jnz outerloop
39
40            ; exit system call
41            mov ax, 0x4c00
42            int 0x21
43
44
45
46
47
```

- In this example we nested loops in which sorted the data in ascending order.

```
1 [org 0x0100]
2
3 jmp start
4
5 data: dw 6, 2, 4, 5
6 swap: db 0 ; use this as a flag
7
8 start:
9     ; mov cx, 4 ; make 10 passes, has to be outside the loop!
10
11 outerloop:
12     mov bx, 0
13     mov byte [swap], 0 ; why the "byte"?
14
15     innerloop:
16         mov ax, [data + bx]
17         cmp ax, [data + bx + 2] ; why did we move the value to AX?
18
19         jae noswap ; if we don't have to swap, we just jump over the swap thing
20
21         ; the swap portion
22         mov dx, [data + bx + 2]
23         mov [data + bx + 2], ax ; again with the AX?
24         mov [data + bx], dx
25         mov byte [swap], 1
26
27     noswap:
28         add bx, 2
29         cmp bx, 6
30         jne innerloop
31
32     ; if we didn't swap even once, we should be done
33     cmp byte [swap], 1 ; don't need to load this in register?
34     je outerloop
35
36     ; check outer loop termination
37     ; sub cx, 1
38     ; jnz outerloop
39
40
41 ; exit system call
42 mov ax, 0x4c00
43 int 0x21
44
45
46
47
```

- In this example we sorted given integers in descending order using the nested loop.

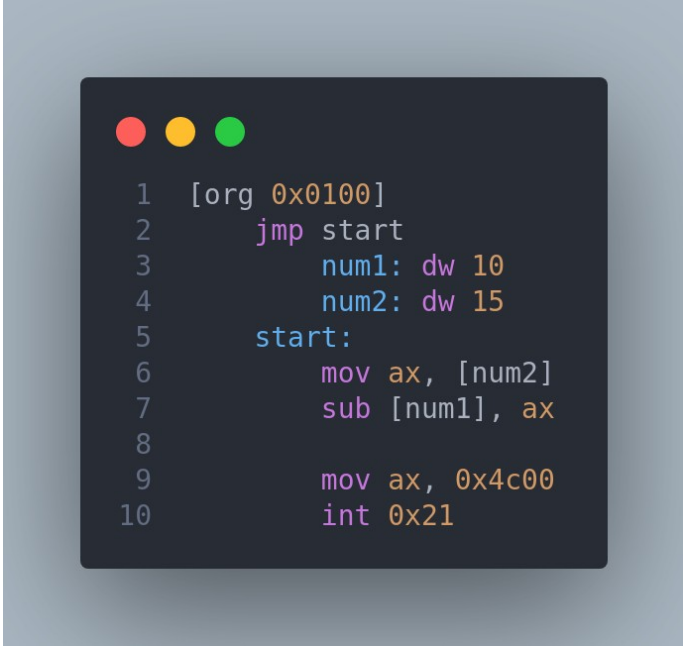
➤ Setting of Different Flags

➤ Sign Flag:

```
1 [org 0x0100]
2 jmp start
3     num1: dw 15
4     num2: dw 20
5 start:
6     mov ax, [num1]
7     cmp ax, [num2] ; sign flag will be setted
8
9     mov ax, 0x4c00
10    int 0x21
```

In this example, when (15-20) is done SF(Sign flag) will be setted.


➤ Carry Flag:



```
1  [org 0x0100]
2      jmp start
3      num1: dw 10
4      num2: dw 15
5      start:
6          mov ax, [num2]
7          sub [num1], ax
8
9          mov ax, 0x4c00
10         int 0x21
```

- In this example when subtracted 10 from 15 carry flag will be setted(Carry is taken for subtraction).

➤ Zero Flag:



```
1  [org 0x0100]
2      jmp start
3      num1: dw 0
4      start:
5          mov ax, [num1]
6          cmp ax, 0
7
8          mov ax, 0x4c00
9          int 0x21
```

In this example Zero Flag will be setted when ax value is compared with 0 and ZF will be setted.

Name : Jawad Ahmed

Roll No : 20P-0165

Section : 3A