COAL_A_p200165_R5

➢ Simple Loop Examples

```
[org 0x0100]
    jmp start
        result: dw 0
    start:
        xor ax, ax
        mov cx, 4

        outerloop:
            add ax, 1     ;ax become 4 after 4 repeatation

            dec cx
            jnz outerloop
        mov [result], ax

        mov ax, 0x4c00
        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.

```
[org 0x0100]
    jmp start
        num1: dw 5, 2
        result: dw 0
    start:
        mov cx, 3
        mov bx, [num1 + 2]
        outerloop:
            add [num1], bx

            dec cx
            jnz outerloop
        mov [result], bx
        mov ax, 0x4c00
        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.

```
[org 0x0100]
    jmp start
        numbers: dw 5, 1, 1, 2
        result: dw 0
    start:
        xor ax, ax
        xor bx, bx
        xor cx, cx

        mov cx, 4

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

            dec cx
            jnz outerloop
        mov [result], ax       ; sum of 4 numbers moved to result label

        mov ax, 0x4c00
        int 0x21

```

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

➢ Nested Loops

```
[org 0x0100]

jmp start

data: dw   6, 2, 4, 5
swap: db   0    ; use this as a flag

start:
    ; mov  cx, 4                          ; make 10 passes, has to be outside the loop!

    outerloop:
        mov  bx, 0
        mov  byte [swap], 0             ; why the "byte"?

        innerloop:
            mov  ax, [data + bx]
            cmp  ax, [data + bx + 2]    ; why did we move the value to AX?

            jbe  noswap                 ; if we don't have to swap, we just jump over the swap thing

                ; the swap potion
                mov  dx, [data + bx + 2]
                mov  [data + bx + 2], ax    ; again with the AX?
                mov  [data + bx], dx
                mov  byte [swap], 1

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

        ; if we didn't swap even once, we should be done
        cmp  byte [swap], 1     ; don't need to load this in register?
        je   outerloop

        ; check outer loop termination
        ; sub cx, 1
        ; jnz outerloop


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

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

```
[org 0x0100]

jmp start

data: dw   6, 2, 4, 5
swap: db   0    ; use this as a flag

start:
    ; mov  cx, 4                          ; make 10 passes, has to be outside the loop!

    outerloop:
        mov  bx, 0
        mov  byte [swap], 0             ; why the "byte"?

        innerloop:
            mov  ax, [data + bx]
            cmp  ax, [data + bx + 2]    ; why did we move the value to AX?

            jae  noswap                 ; if we don't have to swap, we just jump over the swap thing

                ; the swap potion
                mov  dx, [data + bx + 2]
                mov  [data + bx + 2], ax    ; again with the AX?
                mov  [data + bx], dx
                mov  byte [swap], 1

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

        ; if we didn't swap even once, we should be done
        cmp  byte [swap], 1     ; don't need to load this in register?
        je   outerloop

        ; check outer loop termination
        ; sub cx, 1
        ; jnz outerloop


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

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

- <u>Setting of Different Flags</u>
- <u>Sign Flag:</u>

```
[org 0x0100]
    jmp start
        num1: dw 15
        num2: dw 20
    start:
        mov ax, [num1]
        cmp ax, [num2]    ;sign flag will be setted

    mov ax, 0x4c00
    int 0x21
```

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

- ## Carry Flag:

```
[org 0x0100]
    jmp start
        num1: dw 10
        num2: dw 15
    start:
        mov ax, [num2]
        sub [num1], ax

        mov ax, 0x4c00
        int 0x21
```

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

- ## Zero Flag:

```
[org 0x0100]
    jmp start
        num1: dw 0
    start:
        mov ax, [num1]
        cmp ax, 0

        mov ax, 0x4c00
        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