LAB 05 Tasks

Task 01

Move an immediate value of fifteen thousand in a 32-bit register. Observe the carry and sign flag by incrementing the value of the 32-bit register by `1` using `ADD` instruction.

Task 02

Write down the values of the Carry, Sign, Zero, and Overflow flags after each instruction has executed:

```
mov ax, 0A7FF0h
add al, 15h          ; a CF =, SF =, ZF =, OF =
add ah, 1h           ; b CF =, SF =, ZF =, OF =
add ax, 2h           ; b CF =, SF =, ZF =, OF =
```

Task 03

Declare an array variable, `array1` with type `WORD` and initialize it with: `45, 32, 71, 44, 92`. Declare another array, `array2` with the same data type as before. This array should hold the sorted elements in ascending order from the first array. The elements are to be sorted manually. Output the sorted array using loop.

Task 04

- Define three arrays in the `.data` section as follows:
  - `arrayB: BYTE` array with elements `25, 45,` and `65`.
  - `arrayW: WORD` array with elements `155, 185,` and `225`.
  - `arrayD: DWORD` array with elements `645, 690,` and `735`.
- Declare three `DWORD` variables to store the sum of elements from each array: `SUM1, SUM2,` and `SUM3`.
- In the `.code` section, write a main procedure that follows these steps:
  - Load the addresses of the arrays into registers.
  - Calculate `SUM1` as the sum of the first elements of each array `(arrayB[0] + arrayW[0] + arrayD[0])`.
  - Calculate `SUM2` as the sum of the second elements of each array `(arrayB[1] + arrayW[1] + arrayD[1])`.
  - Calculate `SUM3` as the sum of the third elements of each array `(arrayB[2] + arrayW[2] + arrayD[2])`.
- Display the results using `WriteDec` and `Crlf` procedures from the `Irvine32` library.

Output:

```
825
920
1025
```