This task is not mandatory and will not count in your grades for lab 3. You will get an extra 3 points added to your final grades. The deadline for this task is the same as lab 4. So you have about one month to finish this task.

5.1 Introduction

Bubble Sort is the simplest sorting algorithm based on the comparison. It repeatedly steps through the array, compares adjacent pairs and swaps them if they are in the wrong order. The pass through the array is repeated until the array is sorted. And the pseudocode of this algorithm is as follows.[1]

```
procedure BubbleSort (A: array of sortable elements. The index of A is from 0.)
   n = length(A)
   j = 0
     repeat
     swapped = false
     for i = 1 to n-1-j inclusive do
              /* if this pair is out of order */
              if A[i-1] > A[i] then
            /* swap them and remember something changed */
            swap( A[i-1], A[i] )
            swapped = true
             end if
     end for
              j = j +1
     until not swapped
end procedure
```

The following simple example illustrates how bubble sort works.[2]

Initialization:

The input of this algorithm in this example is an array out of order: (5 1 4 2 8). And we want to get an array in ascending order.

First Pass:

The pass process will start with the first number in the array. And it will ends when the **last element** of the array has been compared.

(5 1 4 2 8) –> (1 5 4 2 8), Here, algorithm compares the first two elements, and swaps since 5 > 1.

(1 5 4 2 8) –> (1 4 5 2 8), Swap since 5 > 4
(1 4 5 2 8) –> (1 4 2 5 8), Swap since 5 > 2
(1 4 2 5 8) –> (1 4 2 5 8), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

Second Pass:

The pass process will start with the first number in the array. And it will ends when the element, which is **second to last**, has been compared.

(1 4 2 5 8) –> (1 4 2 5 8)
(1 4 2 5 8) –> (1 2 4 5 8), Swap since 4 > 2
(1 2 4 5 8) –> (1 2 4 5 8)

Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one whole pass without any swap to know it is sorted.

Third Pass:

The pass process will start with the first number in the array. And it will ends when the element, which is **third to last**, has been compared.

(1 2 4 5 8) –> (1 2 4 5 8)
(1 2 4 5 8) –> (1 2 4 5 8)

In this pass, there is actually no swapping happened. So the algorithm will know that the array has been in ascending order. So we get the correct output (1 2 4 5 8).

5.2 Assembly Implementation (Task 7)

The two versions of implementation [2] of the Bubble Sort in C language has been provided as follow:

Version 1:

```
int main()
{
        SIZE = 8;
  int a[SIZE] = {95, 45, 15, 78, 84, 51, 24, 12};
        int i, j, temp;
        int swap;
        do{
        swap = 0;
```

```
        for (i = 0; i < n - 1 - j; i++)
        {
                if(a[i] > a[i + 1])
                {
                        temp = a[i];
                        a[i] = a[i + 1];
                        a[i + 1] = temp;
                        swap = 1;
                }
        }
        }while (swap == 1)
        for (i = 0; i < SIZE; i++)
        {
        printf("%d\n", number[i]);
        }
        return 0;
}
```

Version 2:

```
int main()
{
        SIZE = 8;
   int a[SIZE] = {95, 45, 15, 78, 84, 51, 24, 12};
        int i, j, temp;
        int swap;
        for( j = 0; j < n-1 ; j ++)
        {
        for (i = 0; i < n - 1 - j; i++)
        {
              if(a[i] > a[i + 1])
                {
                        temp = a[i];
                        a[i] = a[i + 1];
                        a[i + 1] = temp;
                }
        }
        }
        for (i = 0; i < SIZE; i++)
        {
```

```
        printf("%d\n", number[i]);
        }
        return 0;
}
```

According to the above implementation, write your own version of bubble sort in assembly language. Version 1 is more efficient than version 2, but you can choose the version you prefer. The input array of this program (the array out of order) can be initialized in the DATA section. For the convenience of the demo, the initial value of input array should be [4,9,3,1,6,3]. However, your program should be able to handle any given array. The output (the array in ascending order) should be displayed on LED (In the C code, it uses the printf to output the array to the screen). Since the LED can only display at most six characters, if you want to run your program using other test cases , it is recommended that the length of the input array is not greater than six. Please submit the source code (main.s) to Courseweb.

6. Reference

[1]. https://en.wikipedia.org/wiki/Bubble_sort
[2]. https://www.geeksforgeeks.org/bubble-sort/