CS2208b Lab No. 5
Introduction to Computer Organization and Architecture

Monday March 14, 2016 (section 3 @ *SSC-1032* from 2:30 pm to 3:30 pm)

Tuesday March 15, 2016 (section 4 @ *SSC-1032* from 1:30 pm to 2:30 pm)

Tuesday March 15, 2016 (section 8 @ *HSB-14* from 3:30 pm to 4:30 pm)

Thursday March 17, 2016 (section 5 @ *SSC-1032* from 2:30 pm to 3:30 pm)

Friday March 18, 2016 (section 6 @ *SSC-1032* from 1:30 pm to 2:30 pm)

Friday March 18, 2016 (section 7 @ *SSC-1032* from 3:30 pm to 4:30 pm)

The objective of this lab is:

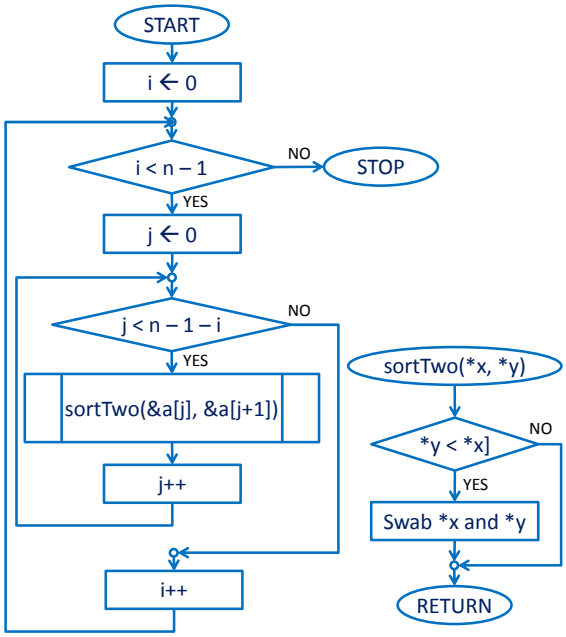
- To practice ARM assembly programming

If you would like to leave, and at least 30 minutes have passed, raise your hand and wait for the TA.
Show the TA what you did. If, and only if, you did a reasonable effort during the lab, he/she will give you the lab mark.

==========

Bubble sort algorithm

The bubble sort is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm, which is a comparison sort, is named for the way smaller elements ***bubble*** to the top of the list. This flowchart describes the algorithm. Below, is an implementation for this flowchart using C.




```
int main()
{
    int a[]={44, -56, 300, 65, -8, 32, 6, -87,
             54,  65,  87, 32, 65};
    int n, i, j;

    n = sizeof(a)/sizeof(a[0]);
    for (i = 0; i < n-1; i++)
    { for (j = 0; j < n-1-i; j++)
        sortTwo(&a[j], &a[j+1]);
    }
    return 0;
}
```

In ARM assembly, you can convert the algorithm as follows:

```
main        ADR r0,a  ;r0 is a pointer to array a
            ADR r1,endOfArray ;address at the end of the array
            SUB r1,r1,r0      ;number of the array bytes
            ASR r1,#2         ;r1: length of the array,i.e., n

            MOV r2,#0         ;r2: outer loop counter,i.e., i
            SUB r4,r1,#1      ;r4 is (n - 1)
startOuter  CMP r2,r4         ;compare i with (n - 1)
            BGE endOuter      ;if i >= (n - 1), then exit the outer loop
            MOV r3,#0         ;r3 is the inner loop counter, i.e., j
            SUB r5,r4,r2      ;r5 is (n - 1 - i)
startInner  CMP r3,r5         ;compare j with (n - 1 - j)
            BGE endInner      ;if j >= (n - 1 -j), then exit the inner loop
            ADD r6,r0,r3,LSL#2;r6 is a pointer to a[j]
            ADD r7,r6,#4      ;r7 is a pointer to a[j+1]
            BL  sortTwo       ;call sortTwo(*a[j],*a[j+1])
            ADD r3,r3,#1      ;increment inner counter j
            B   startInner    ;loop again (inner loop)
endInner    ADD r2,r2,#1      ;increment outer counter i
            B   startOuter    ;loop again (outer loop)
endOuter    B endOuter

a           DCD 44,-56,3,65,-8,32,6,-87,54,65,87,32,65
endOfArray  SPACE 1
```

PROBLEM SET

Before you start practicing this lab, you need to review and fully understand how to use *the Keil ARM Simulator*, as well as tutorial 8 *(Tutorial_08_ARM_Subroutine_Call_and_Return.pdf)*.

1. Fully understand the two codes above.

2. Implement the **`sortTwo`** function. The function must not alter any register value. If you want to use any, you have to store the original value in the stack before changing them and you have to retrieve the original value before returning from the function.

3. Change the **`main`** and the **`sortTwo`** functions in such a way that you utilize the stack when calling the **`sortTwo`** function, i.e., do ***not*** utilize the **`LR`**.

4. Change the main in such a way that if no change occurs inside the inner loop for one full cycle, then you exit the outer loop, as the list will be already sorted in such a situation.