**Department of Computer Engineering**

BLG 351E  
Microcomputer Laboratory Experiment Report

Experiment No : 3

Experiment Date : 23.10.2017

Group Number : Monday - 17

Group Members :

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Surname** |
| 150150302 | İrem Nur | Demirtaş |
| 150160706 | Merve Elif | Demirtaş |
| 150140719 | Cemal | Türkoğlu |
| 150140722 | İbrahim | Dolapçı |

Laboratory Assistant : Ahmet Arış

# Introduction

The aim of the experiment is using the MSP430 for implementing Bit-wise Encryption and Bubble Sort algorithm with assembly code. Encryption is a process of encoding a message of information in such a way that only authorized people can access it. Bubble sort is a sorting algorithm with running time is O(n²) that compare each adjacent element until there is no swap.

# Experiment

Code Composer Studio was used to contact with MSP430. A new assembly project was created to implement the code and debugged with step by step for understanding operations and observing registers value.

## BIT-WISE ENCRYTION

For encryption, these stages were followed in order with 8 bit data:

1. Swap most significant 4-bits of data with least significant 4-bits

( X7X6X5X4 X3X2X1X0  -> X3X2X1X0 X7X6X5X4 )

2. Group bits in pairs and swap again

( X3X2X1X0 X7X6X5X4 -> X2X3X0X1 X6X7X4X5 )

3. Apply XOR operation with a key to the data

( X2X3X0X1 X6X7X4X5 **⊕** k7k6k5k4k3k2k1k0 )

If reverse order is applied, original data can be decrypted.

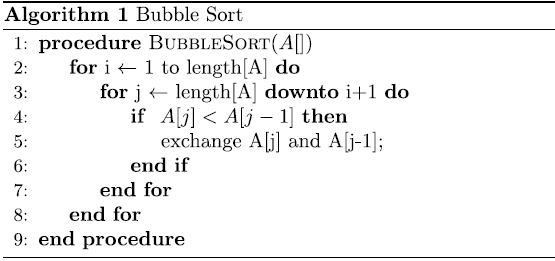
The code is shown below.

1. ;-------------------------------------------------------------------------------
2. .data
3. exampleData .byte 10010011b
4. exampleKey  .byte 00010111b
6. .text                           ; Assemble into program memory.
7. .retain                         ; Override ELF conditional linking
8. ; and retain current section.
9. .retainrefs                     ; And retain any sections that have
10. ; references to current section.
12. ;-------------------------------------------------------------------------------
13. RESET       mov.w   #\_\_STACK\_END,SP         ; Initialize stackpointer
14. StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer

17. ;-------------------------------------------------------------------------------
18. ; Main loop here
19. ;-------------------------------------------------------------------------------
21. Encrypt     mov.b exampleData, R4
22. mov.b exampleData, R5
23. and.b #0f0h, R4             ;And operation for taking most significant bits
24. clrc                        ;Clear carry flag for guarentee that not adding one after shifting
25. rrc.b   R4                  ;Shift right four time to move most significant part to least
26. rrc.b   R4                  ;significant part
27. rrc.b   R4
28. rrc.b   R4
29. and.b   #00fh, R5           ;And operation for taking least significant bits
30. rla.b   R5                  ;Shift left four time to move least significant part to most
31. rla.b   R5                  ;significant part
32. rla.b   R5
33. rla.b   R5
34. add.b   R5, R4              ;add these swapped two registers
35. mov.b   R4, R5              ;mov also another register
37. and.b   #055h, R4           ;and operation for taking odd numbers
38. and.b   #0AAh, R5           ;and operation for taking even numbers
40. rla.b   R4                  ;shift left the taken odd numbers
41. rra.b   R5                  ;shift right the taken even numbers
43. add.b   R4, R5              ;add them
44. xor.b   exampleKey, R5      ;xor operation with example key
45. clrc
46. ;-------------------------------------------------------------------------------

## BUBBLE SORT

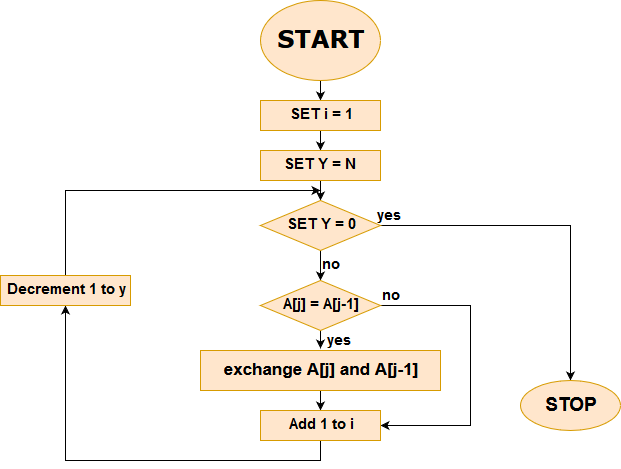
Bubble Sort was implemented with these pseudocode as shown below



For comparting adjacent pairs, started with last element until the first element.

The code is implemented as below.

1. .data
3. unsorted    .byte   5,-9,12,4,-63,127,79,-128,21,65,-35,97
4. lastElement
6. .text                           ; Assemble into program memory.
7. .retain                         ; Override ELF conditional linking
8. ; and retain current section.
9. .retainrefs                     ; And retain any sections that have
10. ; references to current section.
11. ;-------------------------------------------------------------------------------
12. RESET       mov.w   #\_\_STACK\_END,SP         ; Initialize stackpointer
13. StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer
14. ;-------------------------------------------------------------------------------
15. ; Main loop here
16. ;-------------------------------------------------------------------------------
18. mov     #unsorted, R5       ;mov the address of array to R5(i)
19. mov     #lastElement, R9    ;mov last element address to R9
20. dec.w   R9                  ;
21. For1        cmp     R5, R9              ;the end condition of first for loop
22. jeq     EndOfFor            ;if equal end the For1
23. mov     #lastElement, R6
24. dec.w   R6
25. For2        cmp     R6, R5              ;the end condition for second loop
26. jeq     Temp                ;if equal jump for increment i and for1 loop
27. mov.w   R6, R7              ;j
28. dec.w   R7                  ;(j-1)
29. cmp.b   0(R7), 0(R6)        ;compare adjacent pair
30. jl      Swap                ;if less than swap
31. Back        dec.w   R6
32. jmp     For2
34. Swap        mov.b   0(R6), R8           ;The swap operation based on memory
35. mov.b   0(R7), 0(R6)        ;R8 is used as temple
36. mov.b   R8, 0(R7)
37. jmp Back
39. Temp        inc.w   R5                  ;increment i
40. jmp     For1
42. EndOfFor



*Fig.1* Flowchart of Bubble Sort

# Conclusion

In this experiment, we struggled with Bubble Sort implementation cause of the nested loops and confused about registers, but successfully handled the problem. In conclusion, we learned the advanced assembly code implementation.