**Department of Computer Engineering**

BLG 351E  
Microcomputer Laboratory Experiment Report

Experiment No : 3

Experiment Date : 27.10.2017

Group Number : Friday - 13

Group Members :

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Surname** |
| 150160537 | OKAN | YILDIRIM |
| 150160546 | HASAN EMRE | ARI |
|  |  |  |
|  |  |  |

Laboratory Assistant : Yusuf Hüseyin Şahin

# Introduction

In this experiment we get used to use MSP430 Education Board, MSP430G2553 microcontroller and its assembly language in terms of coding more complex algorithms such as Bit-wise Encryption and Bubble Sort. Before the experiment, we have been studying on background information and MSP430\_introduction document for four hours to get familiar with the experiment and some commands we would use. We prepared some sketchy parts of codes for the experiment and came ready to experiment.

# Experiment

## Part 1 Encryption

According to the information given background information section and our preliminary work we wrote our assembly code given below:

---------------------------------------------------------------------------------------------------------------------------------

.data

data .byte 10010011b

key .byte 00010111b

temp .byte 00000000b

.text

---------------------------------------------------------------------------------------------------------------------------------

Firstly, we declared and initialized three variables on data section as you seen above. “data” is will be encrypted according to the key and temp is used for temporarily variable in order to swap bits.

SetupP1 bis.b #0FFh, &P1DIR

mov.b data,&P1OUT

mov.b data,R6 ;

mov.b data,R5

clrc

and.b #00001111b,R5 ;1st step to swap the most significant 4-bits

and.b #11110000b,R6 ; with the least significant 4-bits

rla.b R5

rla.b R5

rla.b R5

rla.b R5

clrc

rrc.b R6

rrc.b R6

rrc.b R6

rrc.b R6

bis.b R6,R5

mov.b R5,&P1OUT ; we want to know the result of first part

mov.b R5,data

mov.b data,temp

and.b #0101010101b,data ;2nd part: bits are grouped in pairs and swapped

and.b #1010101010b,temp

clrc

rrc.b temp

rla.b data

bis.b temp,data

xor.b key,data ; 3rd part: XOR operation is applied to the data with a key

mov.b data,&P1OUT ; display the result

Firstly, we initialized the output in order to display the result end of the code. Then we assigned the variables to registers. Actually, we tried to do it without register at the beginning, yet it made debugging more difficult unfortunately. Then we decided to use register. Moreover, According to the our preliminary work, we swapped the most significant 4-bits with the least significant 4-bits in way to use just four times ***rrc*** or ***rlc*** commands in advance. However, it did now work. Although we tried to use also ***rra*** and ***rla*** command instead of ***rrc*** or ***rlc***, it did now work. Then, we use another way to make the most significant 4-bits zero for a variable and make the least significant 4-bits zero for another temporarily variable by using ***and*** command. Then we used four times ***rla*** and ***rrc*** to swap bits. On last step, by using ***bis*** command to obtain a variable that the most significant 4-bits and the least significant 4-bits are swapped. ***bis*** command is worked because there is no carry due to the zeros. Moreover, we used ***clrc*** command at the somewhere because we try to avoid probable errors caused carry flag.

At the second part, our aim was to make group bits and swap pairs. Similar to the first part, we assign zero to bits by using ***and*** command. Then we use times ***rla*** and ***rrc*** just one time and then we use ***bis*** command again. Totally, all pairs are swapped.

Lastly, we applied XOR operation to the data with the key and display the result on LEDS. Totally, data is encrypted correctly.

## Part 2 Bubble Sort

In this section, we are wanted to sort given array by bubble sort. Moreover, it is required to declare the variable instead of using register. We studied before on 1\_MSP430\_Intrdoduction document and learn how to declare the variable on assembly code. We add counter variable in data section as given below:

.data

unsorted .byte 5,-9,12,4,-63,127,79,-128,21,65,-35,97

lastElement ; and retain current section.

The main program is as follows:

Main mov #unsorted,R5 ;move first element’s address of array to R5

mov #lastElement,R9 ;move last element address + 1 to R9

dec R9 ;R9 equal to last element address

mov R5,R7 ;move first element address to R7

inc R7 ;increment R7 for hold the second element address

Start mov #lastElement,R6 ;the address of last element +1 to R6

dec R6 ;decrement R6 for hold the last element address

mov R6,R8 ;move R6 to R6

dec R8 ;decrement R8 for hold the last element address -1

innerloop cmp.b @R6,0(R8) ;compare last element and previous element of last

JL swap ;if less than last element jump to swap label

swapped dec R6 ;returns after swap process ;decrement R6

dec R8 ;decrement R8

cmp R6,R5 ;compare R6 and R5 to understand that at the begin

jeq outerloop ;if equal to R5 jump to outerloop label

jmp innerloop ;if not equal jump to innerloop label

outerloop inc R5 ;increment R5

inc R7 ;increment R7

cmp R5,R9 ;compare R5 and R9 to understand that at the end

jeq Finish ;if we are at the end, jump to Finish label

jmp Start ;if we aren’t at the end, jump to Start label

swap mov.b @R6,R10 ;move R6 to R10 which are temporary variable

mov.b @R8,0(R6) ;move R8 to R6

mov.b R10,0(R8) ;move R10 to R8

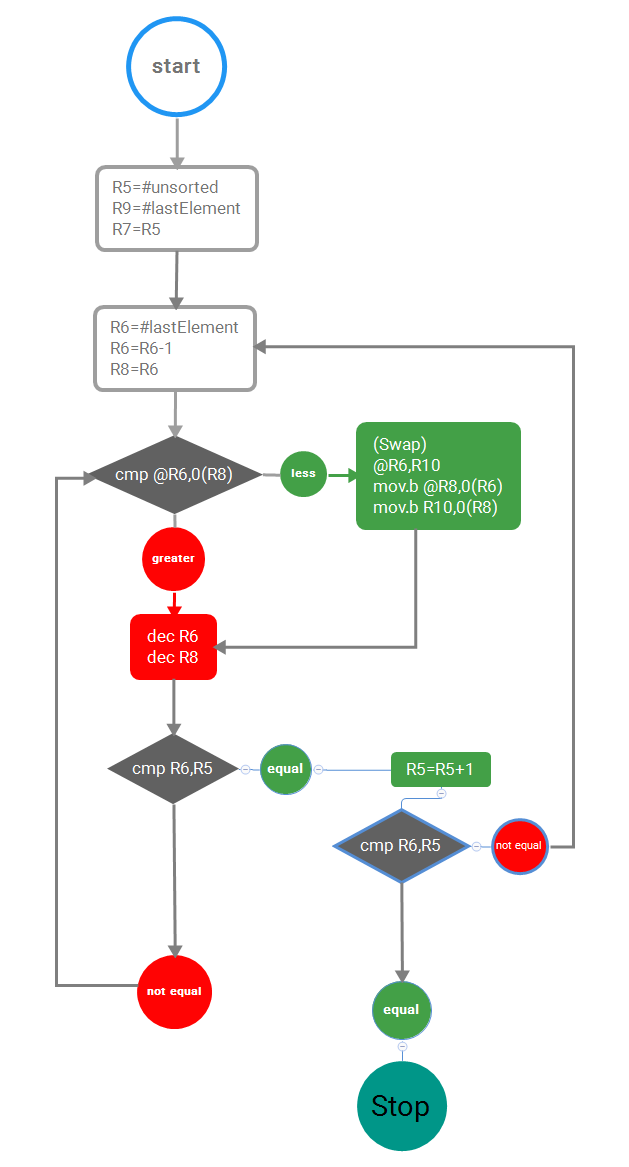
jmp swapped ;jump to swapped label

; end of the swap process inside of R6 and inside of R8 swapped each other.

Finish noop ;when the program comes Finish label, no operation

At the beginning of the program, we have assigned the first element of the array to R5, the second element to R7, the last element to R9 and R6, and the previous element from last element to R8 for traverse the array.

To do the sorting, there are two loops in the algorithm, the inner loop in the outer loop. In the inner loop, the last element is compared with the previous element itself. If the last element is small, it is replaced. This process continues until the first element is reached. At this point, the smallest element is moved to the beginning. Then register R5, which holds the address of the first element, is incremented by one to hold the address of the second element. The position of the first element in this list is preserved. Afterwards, it enters the inner loop again and necessary operations are done. When the program reaches the outer loop, an element arrives at its correct order. This process continues until the address in register R5 is equal to the address of the last element. When it is equal, the whole array is sorted.



# Conclusion

In the encryption section, we tried to shift the given word to us 4 times to the left or right with the commands rrc and rlc. But we did not succeed. Because the rrc and rlc commands operate with carry, the delay between the bits occurs because the element in the first bit remains in the carry.

However, in the Bubble sort section, two elements were swapped together during the swap operation. We learned that the move command is caused by doing the operation as word. We added a “.b” to the move command to make the operation as a bit.

In the bubble sort section, it is quite useful to see what we have stored in the memory while debugging.