Faculty of Computing and Informatics (FCI)

Multimedia University

Cyberjaya

CSN6114 – Computer Architecture & Organization

Trimester 2410

Assignment 2 Report

Submitted to:

Dr. Goh Hui Ngo

Group Members:

NAME	STUDENT ID
IZZA NELLY BINTI MOHD NASIR	1211111583
AIN NUR YASMIN BINTI MUHD ZAINI	1211109582
MOHAMMAD HAZMAN BIN KHAIRIL ANWAR	1211110444
ABDUL ADZEEM ABDUL RASYID	1211109773
MUHAMMAD NABIL NAUFAL BIN MD ZAID	1221101160

PROBLEM STATEMENT

For Q6 to Q13 and Q15. The data is a signed integer of 32 bit. (include +ve and –ve numbers)

Q12. Sort the value in ascending order. Store the result in 2100 onwards.

ARM PROGRAM

```
1 ; Izza Nelly binti Mohd Nasir (1211111583 )
2 ; Ain Nur Yasmin binti Muhd Zaini (1211109582)
3 ; Mohammad Hazman bin Khairil Anwar (1211110444)
4 ; Abdul Adzeem Abdul Rasyid (1211109773)
5 ; Muhammad Nabil Naufal bin Md Zaid (1221101160)
6
7         mov     r0,#0x11000000
8         mov     r1,#0x00110000
9         mov     r2,#0x00001100
10        mov     r3,#0x00000011
11        add     r0, r0, r1
12        add     r0, r0, r2
13        add     r0, r0, r3
14        mov     r8,#0x2000
15        str     r0, [r8]
16
17        ;       move r0,#0x22223333
18        mov     r0,#0x22000000
19        mov     r1,#0x00220000
20        mov     r2,#0x00003300
21        mov     r3,#0x00000033
22        add     r0, r0, r1
23        add     r0, r0, r2
24        add     r0, r0, r3
25        str     r0, [r8,#4]
```

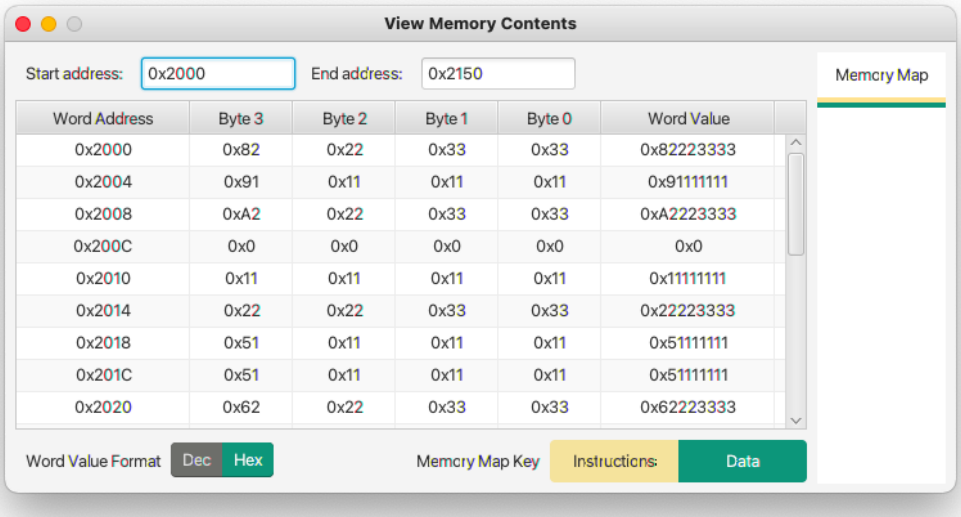
```
27        ;       move r0,#0x31111111
28        mov     r0,#0x31000000
29        mov     r1,#0x00110000
30        mov     r2,#0x00001100
31        mov     r3,#0x00000011
32        add     r0, r0, r1
33        add     r0, r0, r2
34        add     r0, r0, r3
35        str     r0, [r8,#8]
36
37        ;       move r0,#0x42223333
38        mov     r0,#0x42000000
39        mov     r1,#0x00220000
40        mov     r2,#0x00003300
41        mov     r3,#0x00000033
42        add     r0, r0, r1
43        add     r0, r0, r2
44        add     r0, r0, r3
45        str     r0, [r8,#12]
46
47        ;       move r0,#0x51111111
48        mov     r0,#0x51000000
49        mov     r1,#0x00110000
50        mov     r2,#0x00001100
51        mov     r3,#0x00000011
52        add     r0, r0, r1
53        add     r0, r0, r2
54        add     r0, r0, r3
55        str     r0, [r8,#8]
56        str     r0, [r8,#16]
```

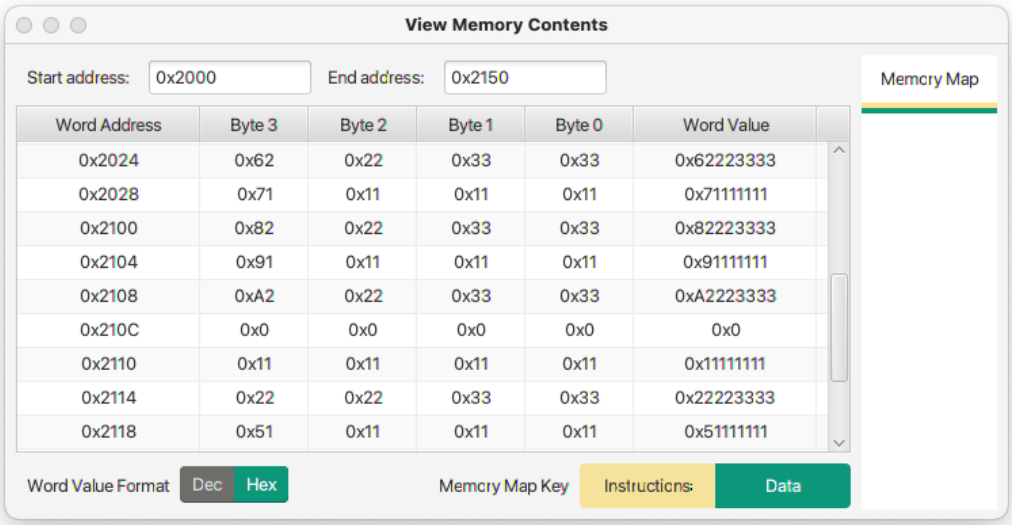
```
58          ;       move r0,#0x62223333
59          mov     r0,#0x62000000
60          mov     r1,#0x00220000
61          mov     r2,#0x00003300
62          mov     r3,#0x00000033
63          add     r0, r0, r1
64          add     r0, r0, r2
65          add     r0, r0, r3
66          str     r0, [r8,#12]
67          str     r0, [r8,#20]
68
69          ;       move r0,#0x71111111
70          mov     r0,#0x71000000
71          mov     r1,#0x00110000
72          mov     r2,#0x00001100
73          mov     r3,#0x00000011
74          add     r0, r0, r1
75          add     r0, r0, r2
76          add     r0, r0, r3
77          str     r0, [r8,#24]
78
79          ;       move r0,#0x82223333
80          mov     r0,#0x82000000
81          mov     r1,#0x00220000
82          mov     r2,#0x00003300
83          mov     r3,#0x00000033
84          add     r0, r0, r1
85          add     r0, r0, r2
86          add     r0, r0, r3
87          str     r0, [r8,#28]
```

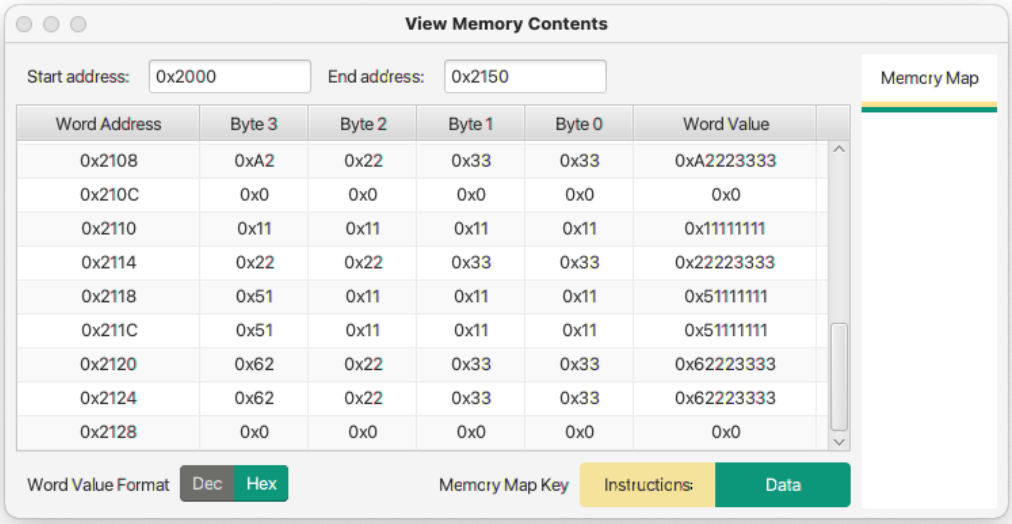
```
89          ;       move r0,#0x91111111
90          mov     r0,#0x91000000
91          mov     r1,#0x00110000
92          mov     r2,#0x00001100
93          mov     r3,#0x00000011
94          add     r0, r0, r1
95          add     r0, r0, r2
96          add     r0, r0, r3
97          str     r0, [r8,#32]
98
99          ;       move r0,#0xA2223333
100         mov     r0,#0xA2000000
101         mov     r1,#0x00220000
102         mov     r2,#0x00003300
103         mov     r3,#0x00000033
104         add     r0, r0, r1
105         add     r0, r0, r2
106         add     r0, r0, r3
107         str     r0, [r8,#36]
```

```
ENTRY

start_sort
        mov     r9, #10          ; Number of elements
        ldr     r8, =0x2000          ; Base address of data array

bubble_sort
        mov     r4, #0           ; loop_outer index i
        mov     r5, r9           ; r5 will keep track of n-i-1 for loop_inner

loop_outer
        cmp     r4, r9           ; COMPARE loop_outer index with number of elements
        beq     transfer_data        ; IF all elements have been processed, call transfer_data

        mov     r6, #0           ; loop_inner index j
        mov     r7, r8           ; Pointer to  first element

loop_inner
        ldr     r0,[r7]          ; LOAD current element
        ldr     r1, [r7, #4]       ; LOAD next element
        cmp     r0, r1           ; COMPARE current element with next element
        ble     no_swap            ; IF in order, CALL no_swap

        ;       Swap elements
        str     r1, [r7]
        str     r0, [r7, #4]

no_swap
        add     r7, r7, #4         ; Move to next element
        add     r6,r6, #1        ; INCREMENT loop_inner index
        cmp     r6,r5            ; COMPARE loop_inner index with n-i-1
        blt     loop_inner           ; IF j < n-i-1, CONTINUE loop_inner

        add     r4, r4, #1         ; INCREMENT loop_outer index
        sub     r5, r5, #1         ; DECREMENT comparison count for next pass
        b     loop_outer           ; REPEAT loop_outer

transfer_data
        ldr     r10, =0x2100         ; Base address for storing sorted data
        mov     r6,#0            ; RESET loop_inner index for transfer loop
        mov     r7,r8            ; Source base address (original sorted array)

copy_loop
        cmp     r6,r9            ; COMPARE loop_inner index with number of elements
        beq     done             ; IF all elements are copied, FINISH
        ldr     r0, [r7], #4       ; LOAD from source and post-increment address
        str     r0,[r10], #4       ; STORE to destination and post-increment address
        add     r6, r6, #1         ; INCREMENT loop_inner index
        b     copy_loop          ; REPEAT copy_loop

done
```

MEMORY CONTENTS






View Memory Contents
Start address: 0x2000
End address: 0x2150
Memory Map
Word Address | Byte 3 | Byte 2 | Byte 1 | Byte 0 | Word Value
0x2108 | 0xA2 | 0x22 | 0x33 | 0x33 | 0xA2223333
0x210C | 0x0 | 0x0 | 0x0 | 0x0 | 0x0
0x2110 | 0x11 | 0x11 | 0x11 | 0x11 | 0x11111111
0x2114 | 0x22 | 0x22 | 0x33 | 0x33 | 0x22223333
0x2118 | 0x51 | 0x11 | 0x11 | 0x11 | 0x51111111
0x211C | 0x51 | 0x11 | 0x11 | 0x11 | 0x51111111
0x2120 | 0x62 | 0x22 | 0x33 | 0x33 | 0x62223333
0x2124 | 0x62 | 0x22 | 0x33 | 0x33 | 0x62223333
0x2128 | 0x0 | 0x0 | 0x0 | 0x0 | 0x0
Word Value Format
Dec
Hex
Memory Map Key
Instructions
Data