M11207509_王佑強_LAB1-2

一. 目的&原理

程式碼1:

1. 做**有號數排序**加總結果放在R7, 如果有Overflow 放在R5
    a. 陣列長度存入n， 用來判斷使否以全部加總完成
    b. R2每次登加4個單位並將值放在R3加總給R7
    c. 如果加總過程有overflow, cpsr的V會變成大寫, 用MOVVS將溢位值存到R5
    d. 將加總數值存在R7
2. 做無號數排序加總結果放在R8, 如果有Overflow 放在R6
    a. 陣列長度存入n， 用來判斷使否以全部加總完成
    b. R2每次登加4個單位並將值放在R3加總給R7
    c. 如果加總過程有overflow, cpsr的C會變成大寫, 用MOVCS將溢位值存到R6
    d. 將加總數值存在R8

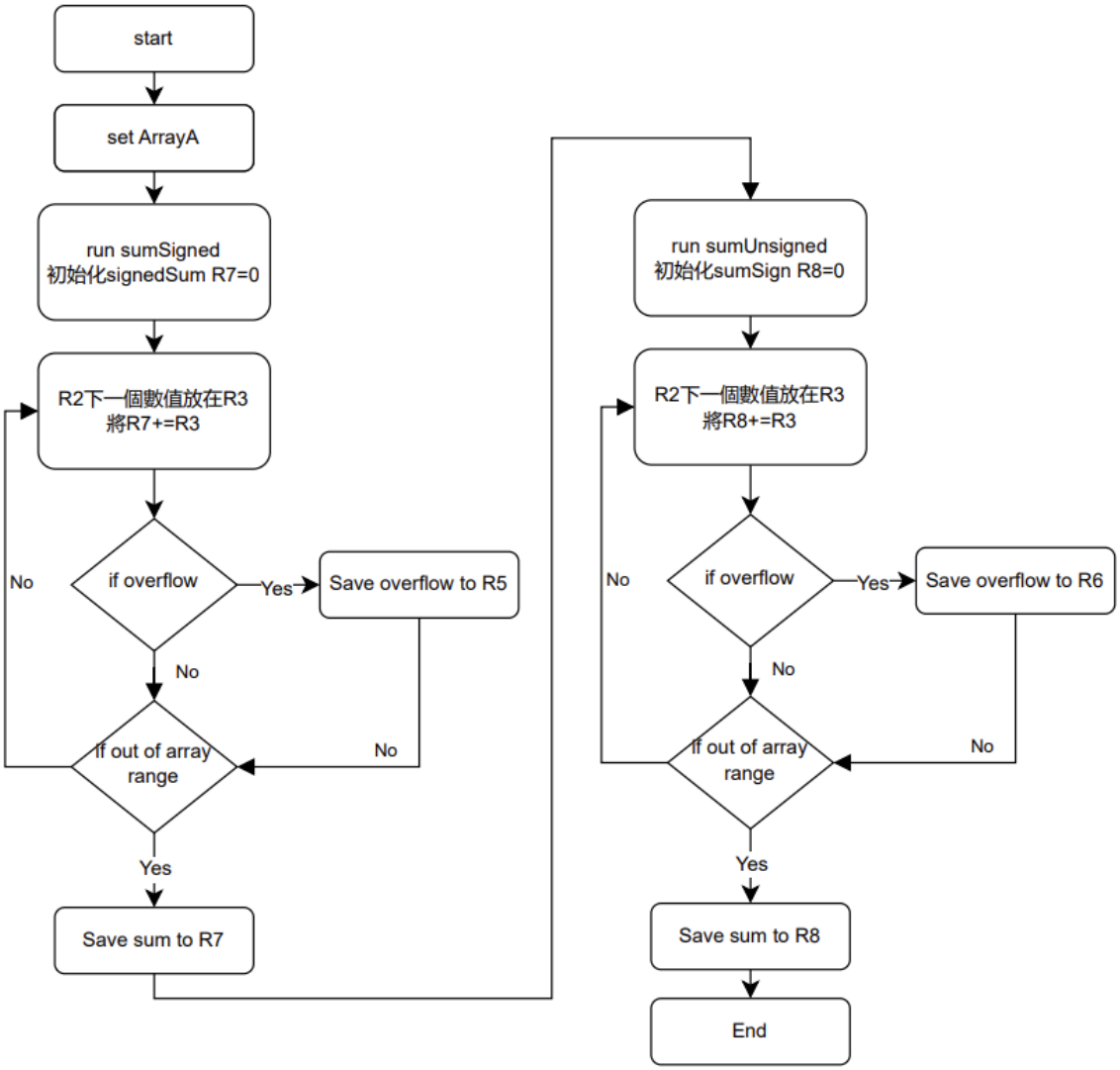
程式碼2:

1. 做有號數排序: bubble sort
    a. 先將MATRIXA資料copy到MATRIXB
    b. 再將r0 指向 MATRIXB進行排序
    c. 用兩層迴圈比較每個元素大小，R6存放當前元素[R3]，R7存放下一個元素[R3, #4]
    d. if R6<R7，不須交換，else的綁兩個值互換
2. 做無號數排序: bubble sort
    a. 先將MATRIXA資料copy到MATRIXC
    b. 再將r0 指向 MATRIXC進行排序
    c. 用兩層迴圈比較每個元素大小，R6存放當前元素[R3]，R7存放下一個元素[R3, #4]

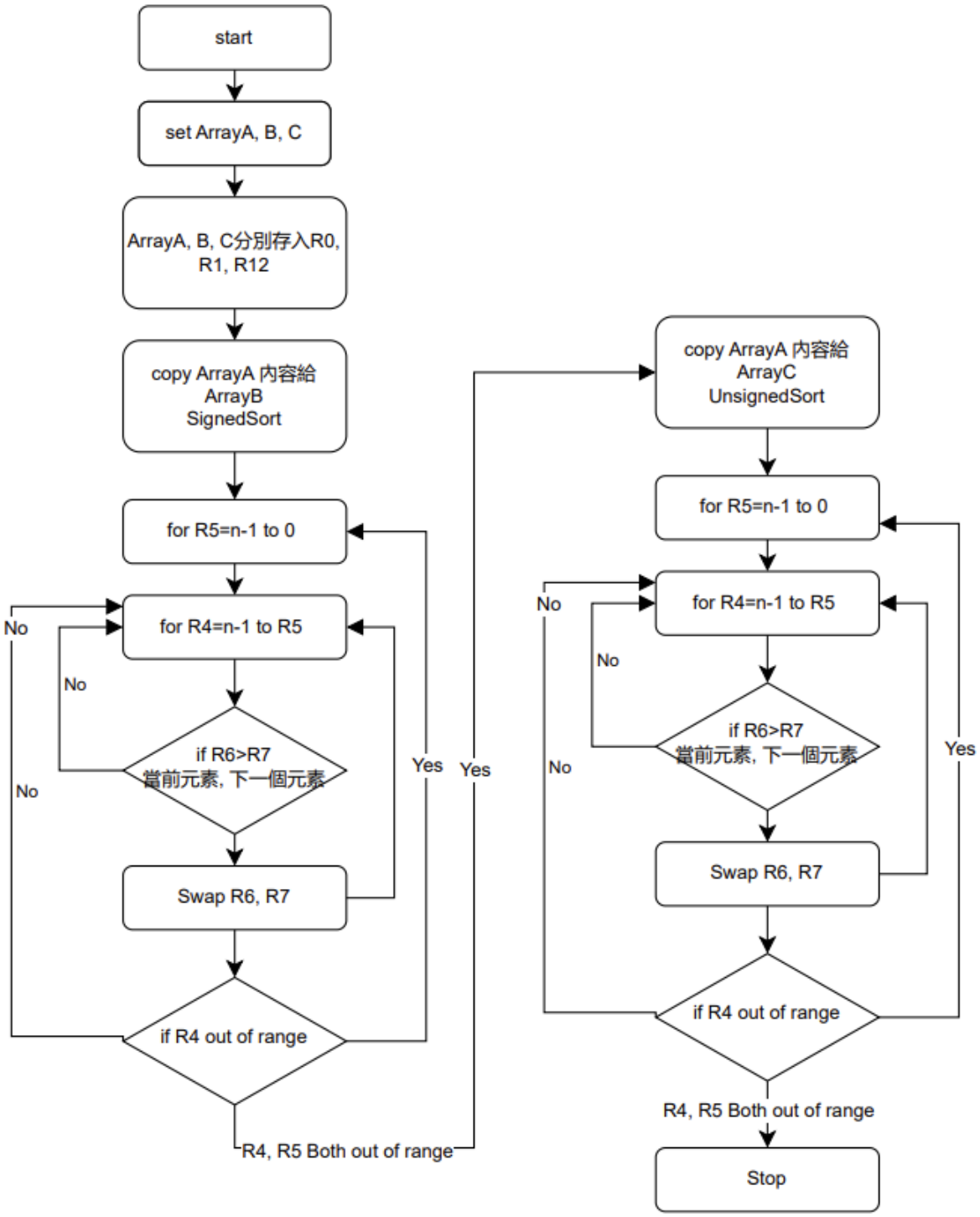
d. unsigned中使用CS決定交換與否

二. 程式流程圖

- 程式碼1



- 程式碼2



三. 程式碼(含註解)

- 程式碼1

```
AREA Data, DATA, READWRITE
signedsum   DCD 0                          ; 初始化 sum 為 0
sumunsign   DCD 0                          ; 初始化無號數和 sumunsign 為 0
n           DCD 6                          ; 陣列長度 (n = 6)
ptr         DCD ArrayA                      ; 指向數組的指標 (int *ptr = ArrayA)
ArrayA     DCD -10, 11, 20, 50, -20, -3  ; 陣列: {-10, 11, 20, 50, -20, -3}

    AREA Array, CODE, READONLY
    ENTRY

start
    BL sumSigned
    BL sumUnsigned
stop

sumSigned
    LDR R1, n                              ; 將 n 加載到 R1
    LDR R2, =ptr                           ; 將 ptr 的地址加載到 R2
    LDR R2, [R2]
    MOV R7, #0                             ; 初始化有號數總和 (R7) 為 0
loop
    LDR R3, [R2], #4                       ; 加載 *R2(將 R2 增加 4,即 int 的大小)到 R3
    ADD R7, R7, R3                         ; 更新有號數和:R7 = R7 + R3
    MOVVS R5, #1                           ; 如果無號數和溢位,則在 R5 存儲溢位資料
    SUBS R1, R1, #1                        ; 減少 n:R1 = R1 - 1;如果(R1 > 0)
    BGT loop                               ; 如果條件滿足,則跳轉到 loop
    STR R7, signedsum                      ; 將有號數和儲在signedsum
    MOV PC, LR                             ; 從函數返回

sumUnsigned
    LDR R1, n                              ; 將 n 加載到 R1
    LDR R2, =ptr                           ; 將 ptr 的地址加載到 R2
    LDR R2, [R2]
    MOV R8, #0                             ; 初始化無號數和 (R8) 為 0
loop_unsigned
    LDR R3, [R2], #4                       ; 加載 *R2(將 R2 增加 4,即 int 的大小)到 R3
    ADDS R8, R8, R3                        ; 更新無號數和:R8 = R8 + R3
    SUBS R1, R1, #1                        ; 減少 n:R1 = R1 - 1;如果(R1 > 0)
    BGT loop_unsigned                      ; 如果條件滿足,則跳轉到 loop_unsigned
    MOVCS R6, #1                           ; 如果無號數和溢位,則在 R6 存儲溢位資料
    STR R8, sumunsign
    b loop                                 ; BREAKPOINT!!!設在這裡

    END
```

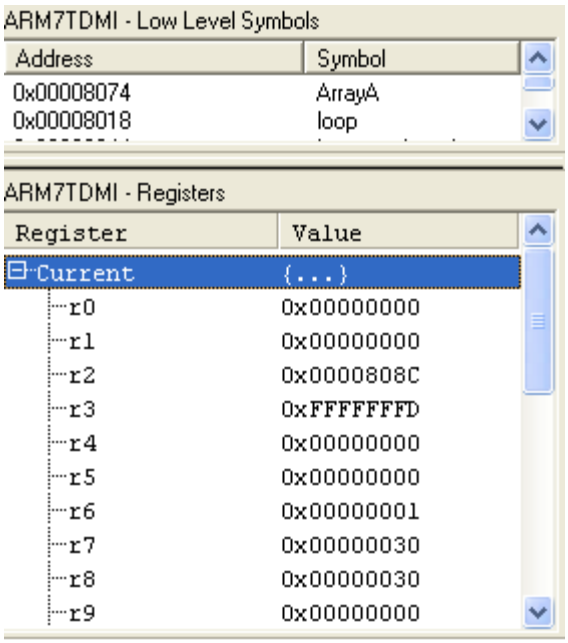
- 程式碼2

```
AREA Data, DATA, READWRITE
n      DCD 6
ArrayA DCD -10, 11, 20, 50, -20, -3
ArrayB DCD 0, 0, 0, 0, 0, 0       ; 用來存儲排序後的數字的陣列
ArrayC DCD 0, 0, 0, 0, 0, 0       ; 用來存儲排序後的數字的陣列

    AREA Array, CODE, READONLY
    ENTRY
start
    LDR R0, =ArrayA  ; 載入 ArrayA 的地址
    LDR R1, =ArrayB  ; 載入 ArrayB 的地址
    LDR R12, =ArrayC ; 載入 ArrayC 的地址
    LDR R2, =n       ; 載入陣列中元素的數量
    LDR R2, [R2]     ; 載入 n 的值
    BL  copy_ArrayB  ; 調用函數從 ArrayA 複製陣列
    BL  copy_ArrayC  ; 調用函數從 ArrayA 複製陣列到 ArrayC
    LDR R0, =ArrayB  ; 現在將 R0 指向 ArrayB 以進行排序
    BL  signedSort   ; 用冒泡排序函數
    LDR R0, =ArrayC  ; 現在將 R0 指向 ArrayC 以進行排序
    BL  unsignedSort ; 調用冒泡排序函數
    b bsort_skip     ; BREAKPOINT!!!設在這裡
stop

; 執行冒泡排序的函數
; 輸入: R0 = 要排序的陣列地址, R2 = 陣列大小
signedSort
    MOV R5, R2       ; 將大小複製到 R5
    SUB R5, R5, #1   ; 為循環條件遞減大小
    MOV R9, R0       ; 備份 R0 到 R9
signedSort_outer_loop
    MOV R3, R9       ; 每次迭代重置 R3 到陣列的開始
    MOV R4, R5       ; 重置 R4 為內循環的 R5
signedSort_inner_loop
    LDR R6, [R3]     ; 載入當前元素
    LDR R7, [R3, #4] ; 載入下一個元素
    CMP R6, R7       ; 比較元素
    BLT signedSort_skip    ; 如果順序正確，跳過交換
    STR R6, [R3, #4] ; 交換元素
    STR R7, [R3]
signedSort_skip
    ADD R3, R3, #4   ; 移動到下一個元素
    SUBS R4, R4, #1  ; 遞減循環計數器
    BNE signedSort_inner_loop
    SUBS R5, R5, #1  ; 遞減外循環計數器
    BNE signedSort_outer_loop
    BX LR            ; 從函數返回
```

```
unsignedSort
    MOV R5, R2       ; 將大小複製到 R5
    SUB R5, R5, #1   ; 為循環條件遞減大小
    MOV R9, R0       ; 備份 R0 到 R9
unsignedSort_outer_loop
    MOV R3, R9       ; 每次迭代重置 R3 到陣列的開始
    MOV R4, R5       ; 重置 R4 為內循環的 R5
unsignedSort_inner_loop
    LDR R6, [R3]     ; 載入當前元素
    LDR R7, [R3, #4] ; 載入下一個元素
    CMP R6, R7       ; 比較元素
    STRCS R6, [R3, #4]; 若條件滿足則交換元素
    STRCS R7, [R3]
unsignedSort_skip
    ADD R3, R3, #4   ; 移動到下一個元素
    SUBS R4, R4, #1  ; 遞減循環計數器
    BNE unsignedSort_inner_loop
    SUBS R5, R5, #1  ; 遞減外循環計數器
    BNE unsignedSort_outer_loop
    BX LR            ; 從函數返回

; 從 ArrayA 到 ArrayB 複製陣列的函數
; 輸入: R0 = 源陣列, R1 = 目標陣列, R2 = 陣列大小
copy_ArrayB
    MOV R9, R0       ; 備份 R0 到 R9
    MOV R10, R1      ; 備份 R1 到 R10
    MOV R11, R2      ; 備份 R2 到 R11
copy_loop
    LDR R3, [R9], #4  ; 從源載入單詞並更新地址
    STR R3, [R10], #4 ; 將單詞存儲到目標並更新地址
    SUBS R11, R11, #1 ; 遞減計數器
    BNE copy_loop     ; 如果計數器不為零則繼續循環
    BX LR             ; 從函數返回

; 從 ArrayA 到 ArrayC 複製陣列的函數
copy_ArrayC
    MOV R9, R0       ; 備份 R0 到 R9
    MOV R10, R12     ; 備份 R12 到 R10
    MOV R11, R2      ; 備份 R2 到 R11
copy_loopB
    LDR R3, [R9], #4  ; 從源載入單詞並更新地址
    STR R3, [R10], #4 ; 將單詞存儲到目標並更新地址
    SUBS R11, R11, #1 ; 遞減計數器
    BNE copy_loopB    ; 如果計數器不為零則繼續循環
    BX LR             ; 從函數返回

    END
```

四. 程式執行結果

- SUM: Unsigned有overflow, signed沒有



- Sort: 分別為ArrayB, ArrayC輸出結果

0x00008120	-20	-10	-3	11
0x00008130	20	50	11	20
0x00008140	50	-20	-10	-3

五. 其他(不一定要)

六. 心得

因為以前沒有修過組合語言與計算機組織相關課程，對這類語言開發很陌生，上網查了很多資料，這次幫助最大的就是運用CPSR 的NZCV這幾個flag來判斷是否有overflow, 以及搭配不同指令去夏條件unsigned, signed對這些flag的判斷進行操作，雖然花很多時間但對這部分有更深的了解。