

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, 用來判斷是否以全部加總完成
 - 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, 用來判斷是否以全部加總完成
 - 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

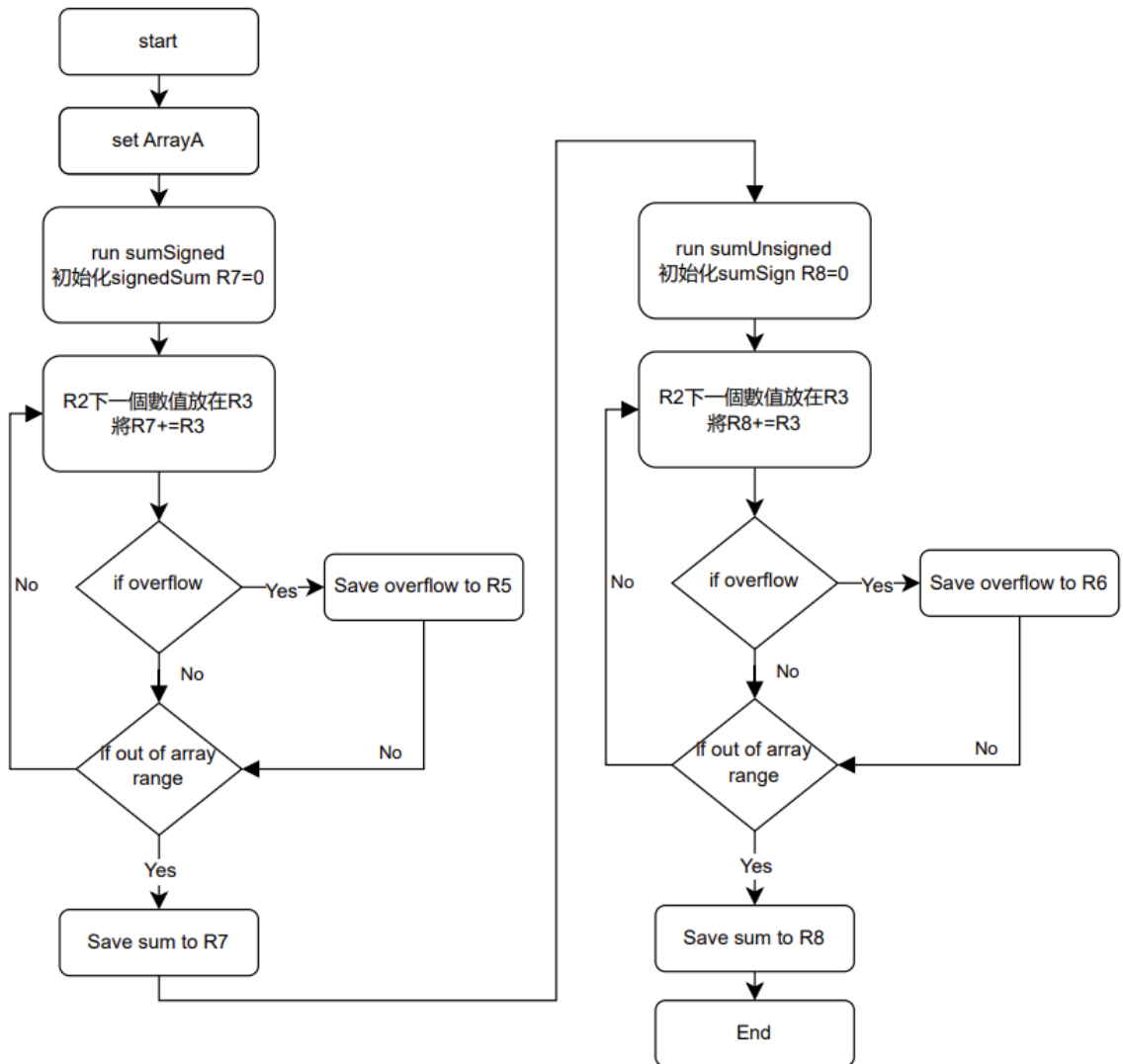
程式碼2:

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]

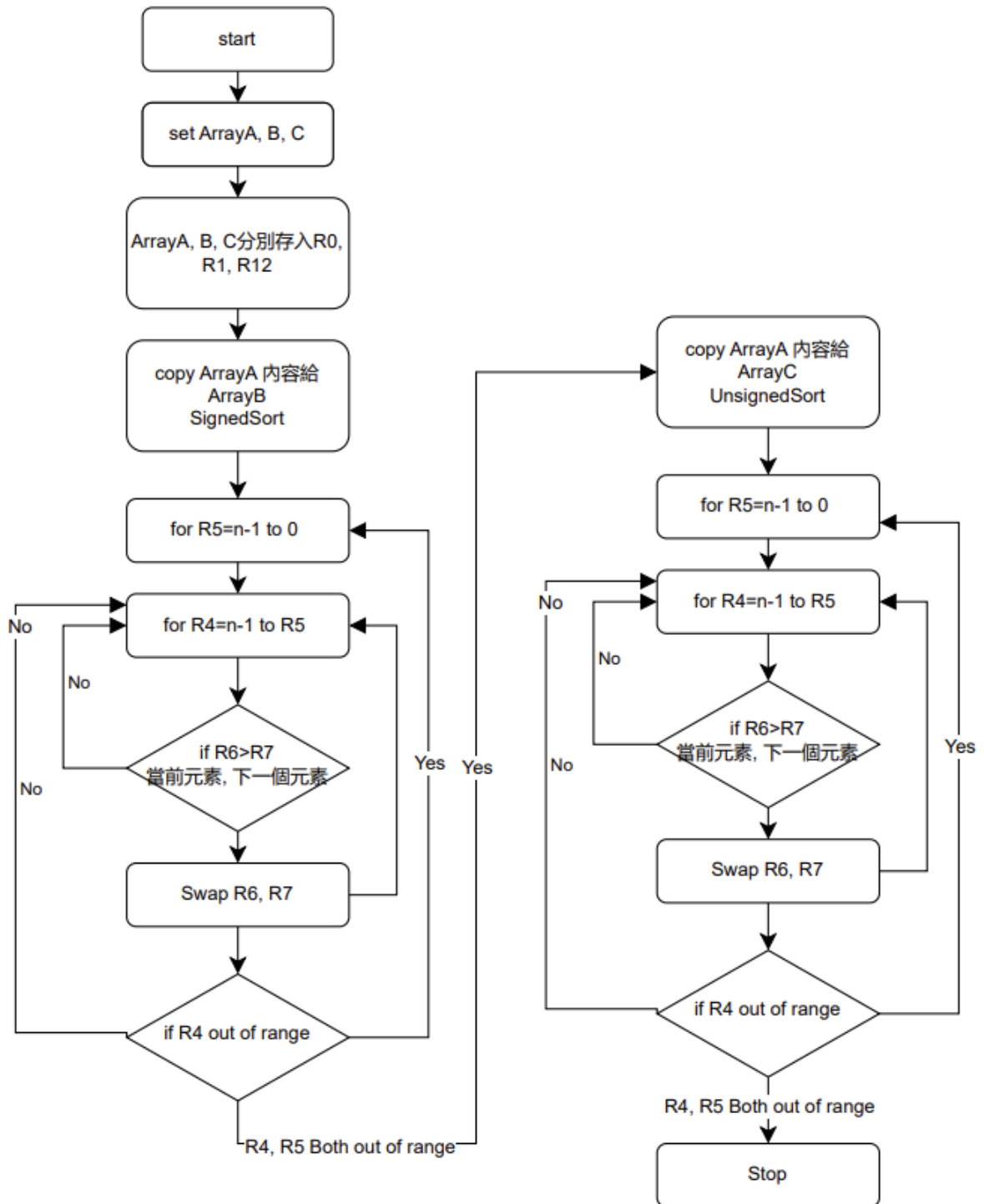
d. unsigned中使用CS決定交換與否

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

```

1  AREA Data, DATA, READWRITE
2  signedsum    DCD 0                ; 初始化 sum 為 0
3  sumunsign    DCD 0                ; 初始化無號數和 sumunsign 為 0
4  n            DCD 6                ; 陣列長度 (n = 6)
5  ptr          DCD ArrayA           ; 指向數組的指標 (int *ptr = ArrayA)
6  ArrayA       DCD -10, 11, 20, 50, -20, -3 ; 陣列: {-10, 11, 20, 50, -20, -3}
7
8      AREA Array, CODE, READONLY
9      ENTRY
10
11  start
12      BL sumSigned
13      BL sumUnsigned
14  stop
15
16  sumSigned
17      LDR R1, n                    ; 將 n 加載到 R1
18      LDR R2, =ptr                ; 將 ptr 的地址加載到 R2
19      LDR R2, [R2]
20      MOV R7, #0                  ; 初始化有號數總和 (R7) 為 0
21  loop
22      LDR R3, [R2], #4             ; 加載 *R2 (將 R2 增加 4, 即 int 的大小) 到 R3
23      ADD R7, R7, R3              ; 更新有號數和: R7 = R7 + R3
24      MOVVS R5, #1                ; 如果無號數和溢位, 則在 R5 存儲溢位資料
25      SUBS R1, R1, #1             ; 減少 n: R1 = R1 - 1; 如果 (R1 > 0)
26      BGT loop                   ; 如果條件滿足, 則跳轉到 loop
27      STR R7, signedsum           ; 將有號數和儲在 signedsum
28      MOV PC, LR                 ; 從函數返回
29
30  sumUnsigned
31      LDR R1, n                    ; 將 n 加載到 R1
32      LDR R2, =ptr                ; 將 ptr 的地址加載到 R2
33      LDR R2, [R2]
34      MOV R8, #0                  ; 初始化無號數和 (R8) 為 0
35  loop_unsigned
36      LDR R3, [R2], #4             ; 加載 *R2 (將 R2 增加 4, 即 int 的大小) 到 R3
37      ADDS R8, R8, R3             ; 更新無號數和: R8 = R8 + R3
38      SUBS R1, R1, #1             ; 減少 n: R1 = R1 - 1; 如果 (R1 > 0)
39      BGT loop_unsigned          ; 如果條件滿足, 則跳轉到 loop_unsigned
40      MOVCS R6, #1                ; 如果無號數和溢位, 則在 R6 存儲溢位資料
41      STR R8, sumunsign
42      b loop                      ; BREAKPOINT!!!設在這裡
43
44  END
45

```

• 程式碼2

```

47 AREA Data, DATA, READWRITE
48 n DCD 6
49 ArrayA DCD -10, 11, 20, 50, -20, -3
50 ArrayB DCD 0, 0, 0, 0, 0, 0 ; 用來存儲排序後的數字的陣列
51 ArrayC DCD 0, 0, 0, 0, 0, 0 ; 用來存儲排序後的數字的陣列
52
53 AREA Array, CODE, READONLY
54 ENTRY
55 start
56 LDR R0, =ArrayA ; 載入 ArrayA 的地址
57 LDR R1, =ArrayB ; 載入 ArrayB 的地址
58 LDR R12, =ArrayC ; 載入 ArrayC 的地址
59 LDR R2, =n ; 載入陣列中元素的數量
60 LDR R2, [R2] ; 載入 n 的值
61 BL copy_ArrayB ; 調用函數從 ArrayA 複製陣列
62 BL copy_ArrayC ; 調用函數從 ArrayA 複製陣列到 ArrayC
63 LDR R0, =ArrayB ; 現在將 R0 指向 ArrayB 以進行排序
64 BL signedSort ; 用冒泡排序函數
65 LDR R0, =ArrayC ; 現在將 R0 指向 ArrayC 以進行排序
66 BL unsignedSort ; 調用冒泡排序函數
67 b bsort_skip ; BREAKPOINT!!!設在這裡
68 stop
69
70 ; 執行冒泡排序的函數
71 ; 輸入: R0 = 要排序的陣列地址, R2 = 陣列大小
72 signedSort
73 MOV R5, R2 ; 將大小複製到 R5
74 SUB R5, R5, #1 ; 為循環條件遞減大小
75 MOV R9, R0 ; 備份 R0 到 R9
76 signedSort_outer_loop
77 MOV R3, R9 ; 每次迭代重置 R3 到陣列的開始
78 MOV R4, R5 ; 重置 R4 為內循環的 R5
79 signedSort_inner_loop
80 LDR R6, [R3] ; 載入當前元素
81 LDR R7, [R3, #4] ; 載入下一個元素
82 CMP R6, R7 ; 比較元素
83 BLT signedSort_skip ; 如果順序正確, 跳過交換
84 STR R6, [R3, #4] ; 交換元素
85 STR R7, [R3]
86 signedSort_skip
87 ADD R3, R3, #4 ; 移動到下一個元素
88 SUBS R4, R4, #1 ; 遞減循環計數器
89 BNE signedSort_inner_loop
90 SUBS R5, R5, #1 ; 遞減外循環計數器
91 BNE signedSort_outer_loop
92 BX LR ; 從函數返回
93

```

```

94 unsignedSort
95 MOV R5, R2 ; 將大小複製到 R5
96 SUB R5, R5, #1 ; 為循環條件遞減大小
97 MOV R9, R0 ; 備份 R0 到 R9
98 unsignedSort_outer_loop
99 MOV R3, R9 ; 每次迭代重置 R3 到陣列的開始
100 MOV R4, R5 ; 重置 R4 為內循環的 R5
101 unsignedSort_inner_loop
102 LDR R6, [R3] ; 載入當前元素
103 LDR R7, [R3, #4] ; 載入下一個元素
104 CMP R6, R7 ; 比較元素
105 STRCS R6, [R3, #4] ; 若條件滿足則交換元素
106 STRCS R7, [R3]
107 unsignedSort_skip
108 ADD R3, R3, #4 ; 移動到下一個元素
109 SUBS R4, R4, #1 ; 遞減循環計數器
110 BNE unsignedSort_inner_loop
111 SUBS R5, R5, #1 ; 遞減外循環計數器
112 BNE unsignedSort_outer_loop
113 BX LR ; 從函數返回
114
115 ; 從 ArrayA 到 ArrayB 複製陣列的函數
116 ; 輸入: R0 = 源陣列, R1 = 目標陣列, R2 = 陣列大小
117 copy_ArrayB
118 MOV R9, R0 ; 備份 R0 到 R9
119 MOV R10, R1 ; 備份 R1 到 R10
120 MOV R11, R2 ; 備份 R2 到 R11
121 copy_loop
122 LDR R3, [R9], #4 ; 從源載入單詞並更新地址
123 STR R3, [R10], #4 ; 將單詞存儲到目標並更新地址
124 SUBS R11, R11, #1 ; 遞減計數器
125 BNE copy_loop ; 如果計數器不為零則繼續循環
126 BX LR ; 從函數返回
127
128 ; 從 ArrayA 到 ArrayC 複製陣列的函數
129 copy_ArrayC
130 MOV R9, R0 ; 備份 R0 到 R9
131 MOV R10, R12 ; 備份 R12 到 R10
132 MOV R11, R2 ; 備份 R2 到 R11
133 copy_loopB
134 LDR R3, [R9], #4 ; 從源載入單詞並更新地址
135 STR R3, [R10], #4 ; 將單詞存儲到目標並更新地址
136 SUBS R11, R11, #1 ; 遞減計數器
137 BNE copy_loopB ; 如果計數器不為零則繼續循環
138 BX LR ; 從函數返回
139
140 END
141

```

四. 程式執行結果

- SUM: Unsigned有overflow, signed沒有

ARM7TDMI - Low Level Symbols	
Address	Symbol
0x00008074	ArrayA
0x00008018	loop
ARM7TDMI - Registers	
Register	Value
Current	{...}
r0	0x00000000
r1	0x00000000
r2	0x0000808C
r3	0xFFFFFFFF
r4	0x00000000
r5	0x00000000
r6	0x00000001
r7	0x00000030
r8	0x00000030
r9	0x00000000

- 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入的了解。