**National Taiwan University of Science and Technology, NTUST**

**Design of Embedded Microprocessor Systems**

**EE5019701\_HW2**

教授:王乃堅

班級:電機所碩一

姓名:陳俊博

學號:M11207521

**目錄**

[**一、 目的&原理 3**](#_Toc164041283)

[**二、 程式流程圖 4**](#_Toc164041284)

[**三、 程式碼(含註解) 5**](#_Toc164041285)

[**四、 程式執行結果 7**](#_Toc164041286)

[**五、 心得 11**](#_Toc164041287)

1. **目的&原理**

使用泡沫排序法來處理一組數字：-10, 11, 20, 50, -20, -3。首先對這些有號數進行排序，使它們按從小到大的順序排列。這一過程不僅需確保數字間的正確順序，特別是要注意負數的處理，確保它們在正確的位置。排序完成後，將這些數字進行累加，同時檢查是否存在溢出的情況，這涉及到了進位的檢測，以確保計算的準確性。將最終的排序結果儲存於ArrayB中，而溢出情況和總和則分別存放在R5和R7暫存器中。接著針對相同的數字序列再次進行排序，這次將它們視作無號數來處理。在無號數的排序過程中，需注意不將它們當作負數來處理，因為無號數的值範圍與有號數不同，要求在排序時必須正確理解數值的範圍以避免錯誤。排序後的無符號數同樣進行累加，並且檢查是否溢出。結果將存放於ArrayC，而溢出情況和總和分別放在R6和R8暫存器中。

泡沫排序是一種簡單的排序演算法，通過重複遍歷需排序的數列，比較每對相鄰元素，並在順序錯誤的情況下交換它們，逐步將每個元素放到它應該在的位置。這個過程一直重複到沒有再需要交換的元素為止，即該數列已排序完成。

這個過程於數列的第一個元素，持續進行直到整個數列排序完成。在每次遍歷中，演算法從數列的起始位置開始，一直到最後一個尚未排序的元素。它比較每一對連續元素，即第i個和第i+1個元素。如果發現前者大於後者，則兩者會交換位置。

這樣的比較和交換過程確保了每輪遍歷都能將該輪中最大的元素推移到其應有的位置，為數列的未排序部分的最後。隨著每輪的完成，未排序的數列長度逐渐減少，因為每完成一輪遍歷後，至少有一個元素被放置在其最終位置上。在這個過程一直持續重複，直到不再需要進行任何交換，表示數列已排序完成。

**Time complexity:**

1. Best case: )

情況為input data恰巧由小→大呈現

(只需n-1次比較，且無swap發生)

1. Worst case: )

情況為input data恰巧由大→小呈現

(若n筆，則需swap n-1, n-2,..., 1次→n(n-1)/2次)

1. Average case: )。

**Space complexity:**

* )

1. **程式流程圖**

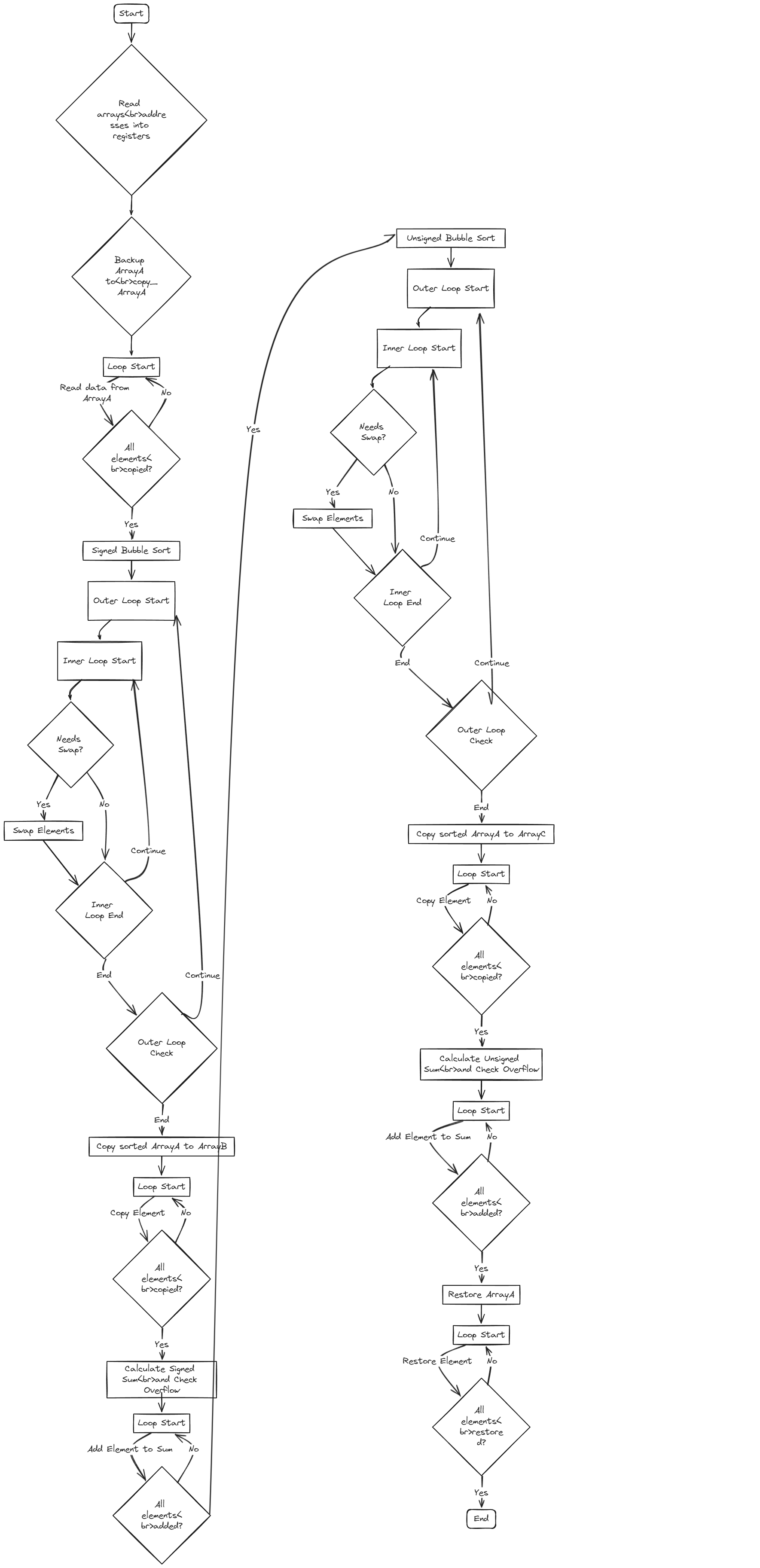


圖1 程式流程圖

1. **程式碼(含註解)**

AREA Matrix, CODE, READONLY ; AREA定義一個記憶體區域，包含指令或資料。

; Matrix為該記憶體區域的名稱。

; CODE為該區域用來存放程式碼。

; READONLY為程式在執行時這部分記憶體不能被修改。

ENTRY ; ENTRY表示程式開始執行的地方。

start ; 程式開始

LDR r0, =ArrayA ; 將ArrayA的地址載入至r0

LDR r10, =ArrayB ; 將ArrayA的地址載入至r10

LDR r11, =ArrayC ; 將ArrayA的地址載入至r11

LDR r12, =copy\_ArrayA ; 將copy\_ArrayA的地址載入至r12

MOV r1, #5 ; 將常數5移至r1 (用於迴圈count)

MOV r2, #0 ; 將常數0移至r2 (用於迴圈index)

backup\_ArrayA ; 備份ArrayA

LDR r3, [r0, r2, LSL #2] ; 將r0 + r2 \* 4位置的值載入至r3

STR r3, [r12, r2, LSL #2] ; 將r3中的值儲存至r12 + r2 \* 4位置

ADD r2, r2, #1 ; r2加1

CMP r2, #6 ; 將r2與6比較

BLT backup\_ArrayA ; 如果r2小於6，則跳回至backup\_ArrayA

; 重複到ArrayA中的值備份至copy\_ArrayA就結束

signed\_out\_loop ; 有號數bubble sort的外部迴圈

MOV r2, #0 ; 內部迴圈設定r2為0

signed\_in\_loop ; 有號數bubble sort的內部迴圈

LDR r3, [r0, r2, LSL #2] ; 將r0 + r2 \* 4位置的值載入至r3

ADD r9, r2, #1 ; 計算下一個index(r2 + 1)並儲存至r9

LDR r4, [r0, r9, LSL #2] ; 將r0 + r9 \* 4位置的值載入至r4

CMP r3, r4 ; 將r3與r4的值進行比較

BLT signed\_no\_swap ; 如果r3小於r4，則不需要交換; 否則進行交換

STR r4, [r0, r2, LSL #2] ; 將r4中的值儲存至r0 + r2 \* 4位置

STR r3, [r0, r9, LSL #2] ; 將r3中的值儲存至r0 + r9 \* 4位置

signed\_no\_swap ; 有號數不須交換的情況

ADD r2, r2, #1 ; index r2 = r2 + 1

CMP r2, r1 ; 將index r2 和 index r1比較

BLT signed\_in\_loop ; 如果r2小於r1，返回至內部迴圈繼續比較

SUBS r1, r1, #1 ; 如果r2沒有小於r1，index = r1 - 1

CMP r1, #0 ; 將index r1與0進行比較

BGT signed\_out\_loop ; 如果r1大於0，返回至外部迴圈; 否則代表完成排序

MOV r1, #0 ; 重設為r1為0，下一個function需用到

signed\_copy\_ArrayA\_to\_ArrayB ; 將排序好的有號數複製到ArrayB

LDR r3, [r0, r1, LSL #2] ; 將r0 + r1 \* 4位置的值載入至r3

STR r3, [r10, r1, LSL #2] ; 將r3中的值儲存至r10 + r1 \* 4位置

ADD r1, r1, #1 ; r1加1

CMP r1, #6 ; 將r1與6比較

BLT signed\_copy\_ArrayA\_to\_ArrayB ; 如果r1小於6，繼續複製值到ArrayB

fir\_initial ; 第一次初始化使用過的暫存器

MOV r1, #0

MOV r2, #0

MOV r3, #0

MOV r4, #0

MOV r5, #0

MOV r7, #0

MOV r9, #0

signed\_sum\_overflow ; 計算有號數的和與溢出次數

LDR r4, [r0, r2, LSL #2] ; 將r0 + r2 \* 4位置的值載入至r4

ADDS r7, r7, r4 ; 將有號數總和加到r7中，並更新溢出

ADC r5, r5, #0 ; 將前一加法的進位加到r5中

ADD r2, r2, #1 ; index r2 = r2 + 1

CMP r2, #6 ; 將r2與6進行比較

BLT signed\_sum\_overflow ; 如果r2小於6，則繼續加總

MOV r1, #5 ; 為無號數排序重設r1為5

unsigned\_out\_loop ; 無號數bubble sort的外部迴圈

MOV r2, #0 ; 為內部迴圈重設r2為0

unsigned\_in\_loop ; 無號數bubble sort的內部迴圈

LDR r3, [r0, r2, LSL #2] ; 將r0 + r2 \* 4位置的值載入至r3

ADD r9, r2, #1 ; 計算下一個index(r2 + 1)並儲存至r9

LDR r4, [r0, r9, LSL #2] ; 將r0 + r9 \* 4位置的值載入至r4

CMP r3, r4 ; r3與r4進行比較

BLO unsigned\_no\_swap ; 如果r3小於r4，則不須交換，否則進行交換

STR r4, [r0, r2, LSL #2] ; 將r4中的值儲存至r0 + r2 \* 4位置

STR r3, [r0, r9, LSL #2] ; 將r3中的值儲存至r0 + r9 \* 4位置

unsigned\_no\_swap ; 有號數不須交換的情況

ADD r2, r2, #1 ; index r2 = r2 + 1

CMP r2, r1 ; index r2 與 index r1進行比較

BLT unsigned\_in\_loop ; 如果r2小於r1，返回內部迴圈

SUBS r1, r1, #1 ; 如果r2沒有小於r1，index r1 = r1 - 1

CMP r1, #0 ; r1與0進行比較

BGT unsigned\_out\_loop ; 如果r1大於0，返回外部迴圈; 否則代表完成排序

MOV r1, #0 ; 重設為r1為0，下一個function需用到

unsigned\_copy\_ArrayA\_to\_ArrayC ; 將排序好的無號數複製到ArrayC

LDR r3, [r0, r1, LSL #2] ; 將r0 + r1 \* 4位置的值載入至r3

STR r3, [r11, r1, LSL #2] ; 將r3中的值儲存至r11 + r1 \* 4位置

ADD r1, r1, #1 ; r1加1

CMP r1, #6 ; 將r1與6進行比較

BLT unsigned\_copy\_ArrayA\_to\_ArrayC ;如果r1小於6，繼續複製到ArrayC

sec\_initial ; 第二次初始化使用過的暫存器

MOV r1, #0

MOV r2, #0

MOV r3, #0

MOV r4, #0

MOV r6, #0

MOV r8, #0

MOV r9, #0

unsigned\_sum\_overflow ; 計算無號數的和與溢出次數

LDR r4, [r0, r2, LSL #2] ; 將r0 + r2 \* 4位置的值載入至r4

ADDS r8, r8, r4 ; 將無號數總和加到r8中，並更新溢出

ADC r6, r6, #0 ; 將前一加法的進位加到r6中

ADD r2, r2, #1 ; index r2 = r2 + 1

CMP r2, #6 ; 將r2與6進行比較

BLT unsigned\_sum\_overflow ; 如果r2小於6，則繼續加總

MOV r1, #0 ; 為r1設定為0

LDR r2, =copy\_ArrayA ; 將copy\_ArrayA的地址載入至r2

restore ; 還原ArrayA的數值

LDR r3, [r2, r1, LSL #2] ; 將r2 + r1 \* 4位置的值載入至r3

STR r3, [r12, r1, LSL #2] ; 將r3中的值儲存至r12 + r1 \* 4位置

ADD r1, r1, #1 ; r1加1

CMP r1, #6 ; 將r1與6進行比較

BLT restore ; 如果r1小於6，繼續還原ArrayA的值;否則還原完成

MOV r2, #0 ; 初始化值

MOV r3, #0 ; 初始化值

MOV r4, #0 ; 初始化值

stop ; 程式停止

MOV r0, #0x18

LDR r1, =0x20026

SWI 0x123456 ; 中斷

AREA Data, DATA, READWRITE ; 定義名為Data的資料區域，可讀寫

ArrayA dcd -10, 11, 20, 50, -20, -3 ; 初始化ArrayA

ArrayB dcd 0, 0, 0, 0, 0, 0 ; 初始化ArrayB

ArrayC dcd 0, 0, 0, 0, 0, 0 ; 初始化ArrayC

copy\_ArrayA dcd 0, 0, 0, 0, 0, 0 ; 初始化copy\_ArrayA

END ; 結束

1. **程式執行結果**

在這次的實驗中，有號數的總和被存放在r7暫存器，而溢位的結果則被放置在r5暫存器。從圖3的狀態觀察，r5暫存器的顯示值為3，這表示在加法過程中發生了三次溢位。r7暫存器的值為48，這代表經過有號數排序後的累加結果。具體到運行過程，當在Debug模式下按下f8（單步執行）時，可以看到r5暫存器的值從0變化到1，這發生在將-20與-10相加後得到-30的過程中，發生了第一次溢位，如圖4及圖5所示。接著在圖6和圖7中，也觀察到了類似的溢位情況，證明了總共發生了三次溢位。

至於有號數排序的結果則被放置在ArrayB中。在我的程式碼中，ArrayB的地址被載入到r10暫存器，從圖中可以看到r10暫存器的第一個位置是-20，隨後每隔四個位置分別是-10、-3、11、20和50，如圖8所示。

對於無符號數的處理，排序結果被放置在ArrayC中，且ArrayC的地址載入到r11暫存器。從圖中可以看到r11暫存器的第一個位置是11，隨後每隔四個位置依次是20、50、-20、-10和-3，如圖9所示。

無符號數的總和與溢位次數結果顯示在r6暫存器（溢位次數）與r8暫存器（無號數總和），均為3次溢位和總和為48。

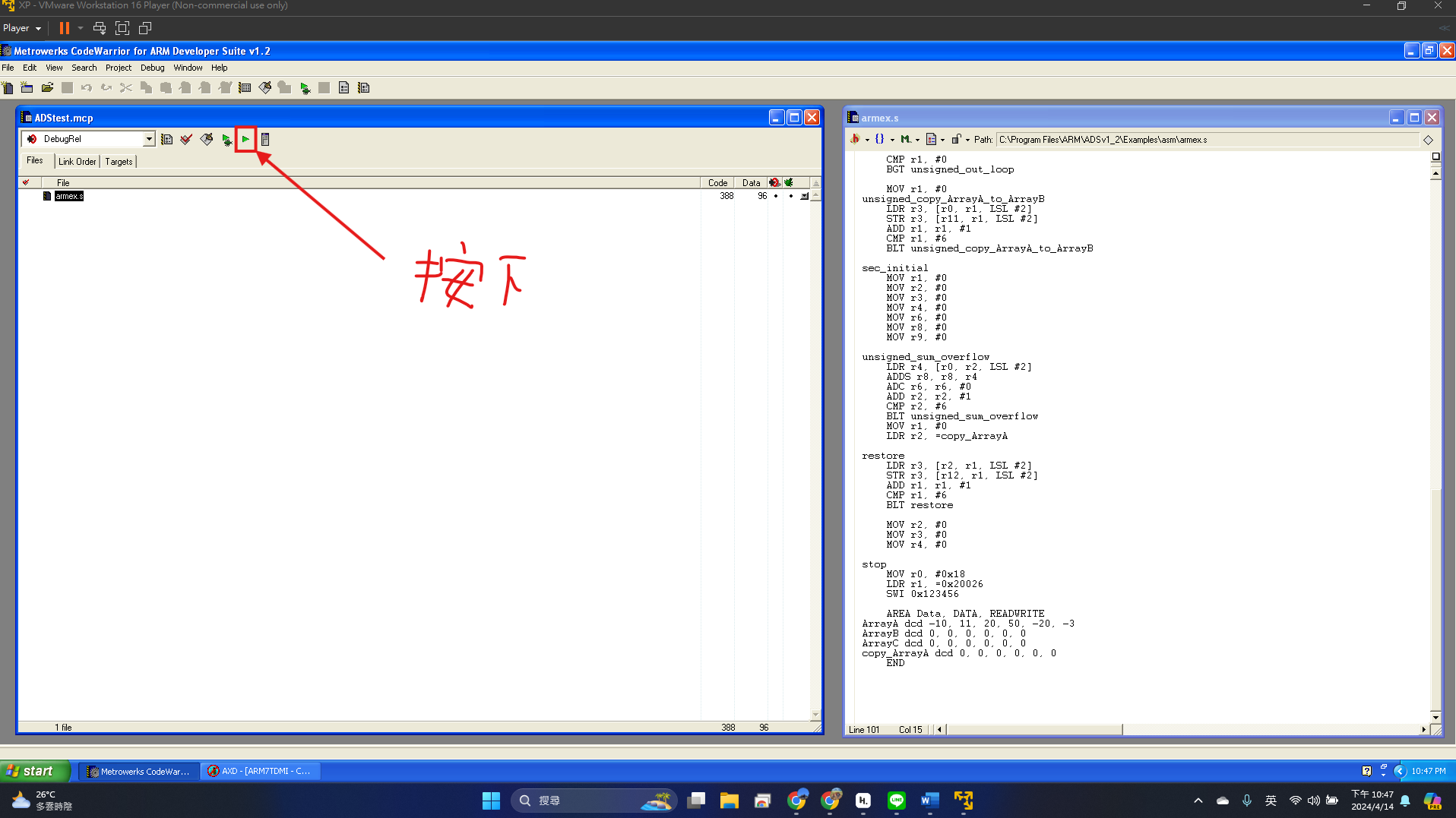


圖2 ARM Developer Suite v1.2介面

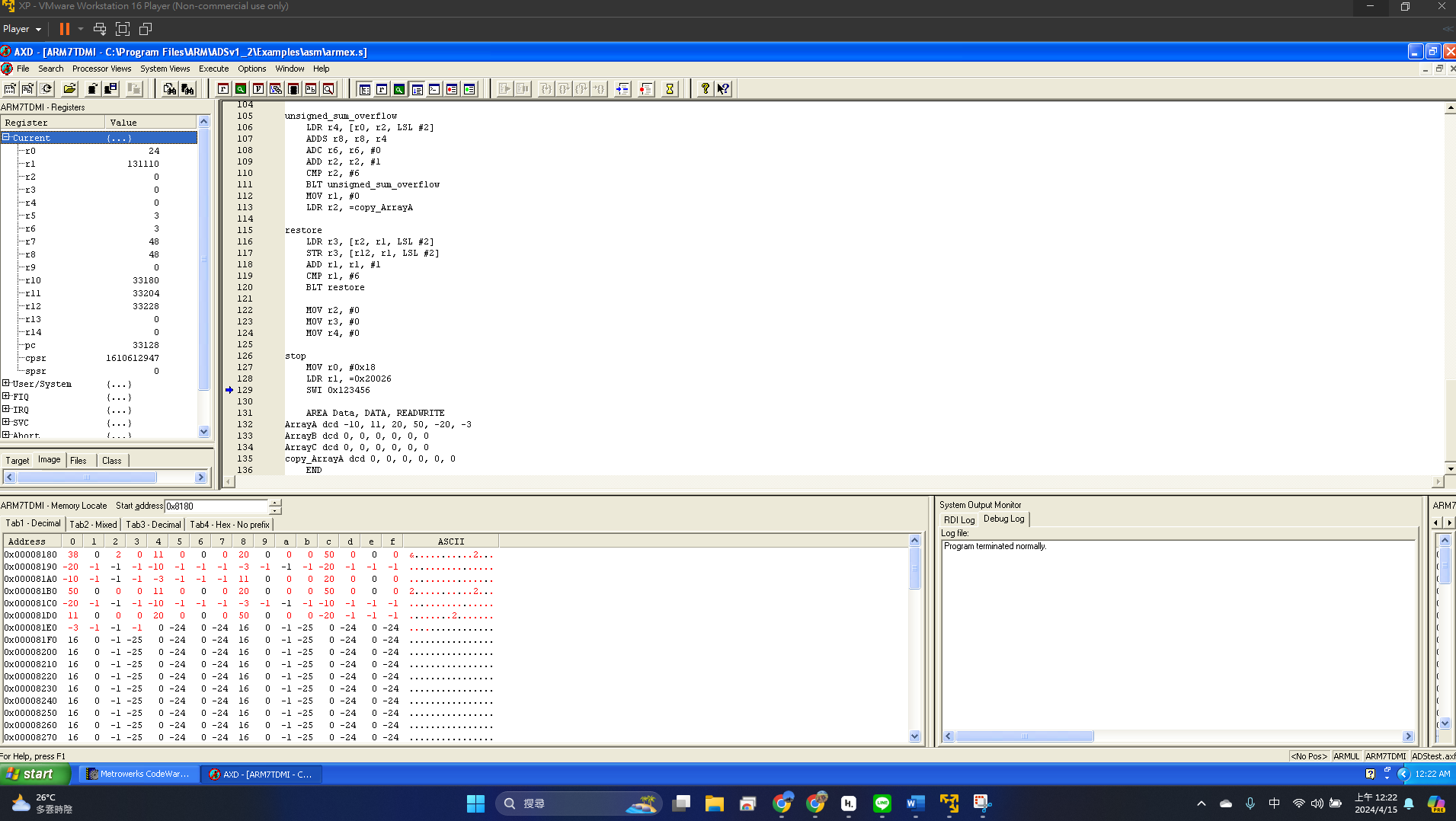


圖3 按下Run後執行結果之AXD介面

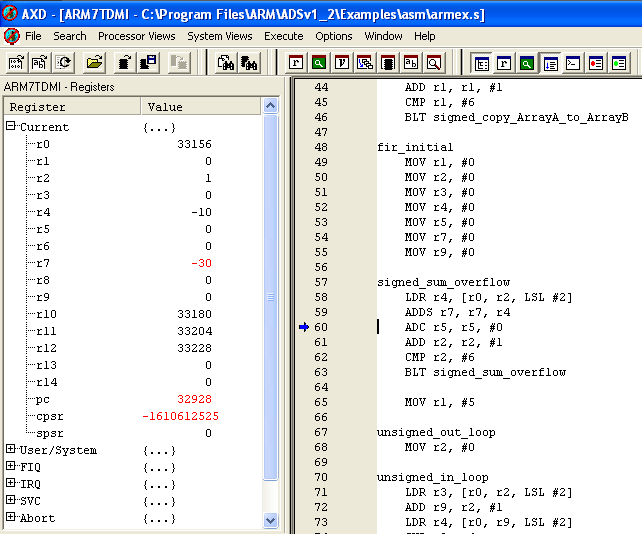


圖4 無發生溢位

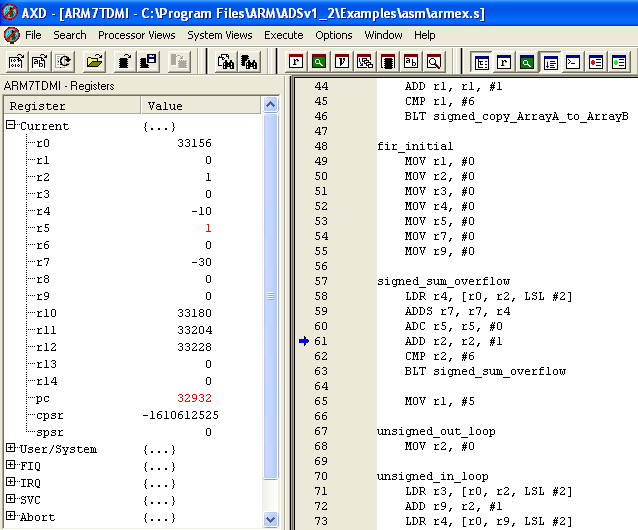


圖5 發生第一次溢位

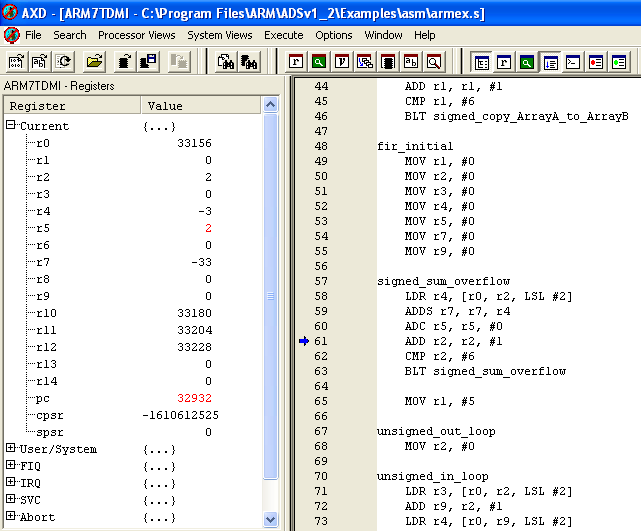


圖6 發生第二次溢位

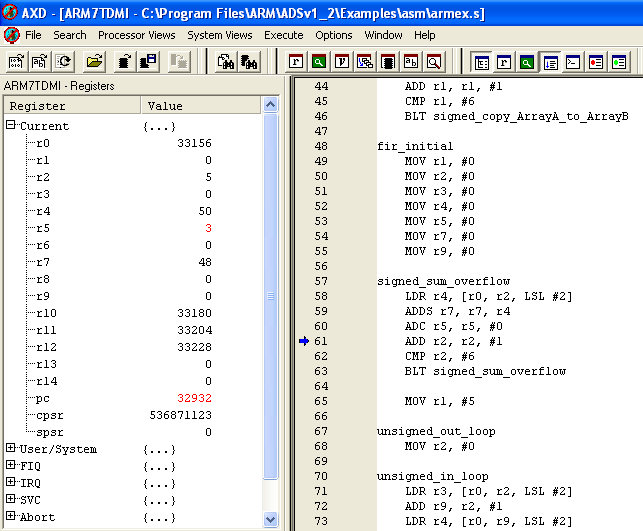


圖7 發生第三次溢位

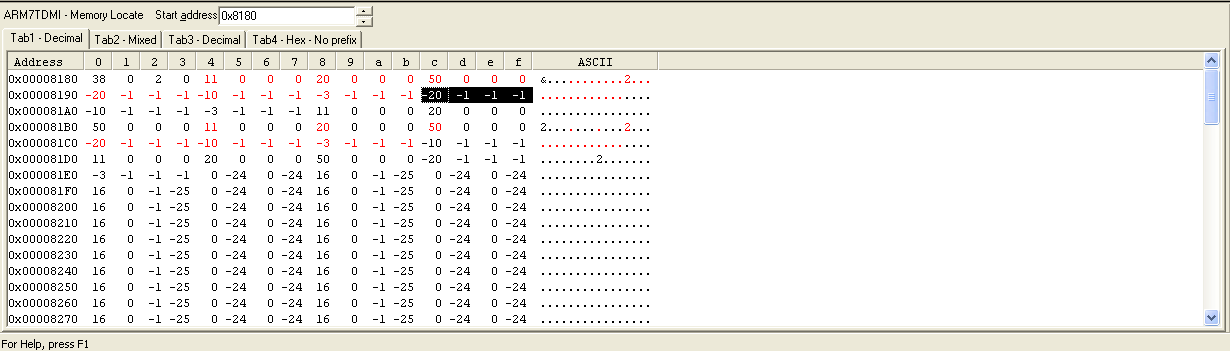


圖8 有號數ArrayB排序之結果

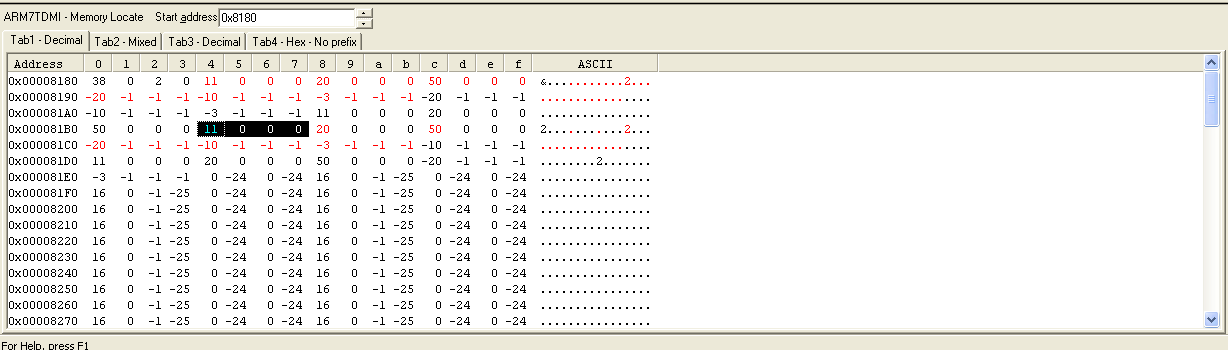


圖9 無號數ArrayC排序之結果

1. **心得**

在這次的實驗中，我使用了泡沫排序法來處理有號數和無號數，並進行了不同情況下的數值累加和溢出檢查。透過這個實驗，我對嵌入式系統中資料處理和記憶體管理有了初步的認識。

在這次實驗中，針對一些數值進行了累加和溢出的檢測。使用ARM組合語言來進行這些操作，學習如何用ARM組合語言進行數學運算和溢出檢查，對我來說是非常有用的技能，這不僅提高對ARM組合語言的熟練度，也加深了對於底層CPU工作原理的理解。這種技能對於任何需要精確控制硬體運作的開發工作都是非常寶貴的。

這次的實驗增加了我的組合語言程式技巧，也加深了我對處理器運作和記憶體管理的理解。這些經驗對我來說很讚，讓我對未來開發更複雜的系統或進行最佳化有了更多的信心。但突然要我們寫排序真的讓我差點哭出來…