# Lab1

0610032 林仲偉 0610023 彭楚鈞

1. **What** 實驗要做什麼

熟悉基本 ARMv7 組合語言語法使用。

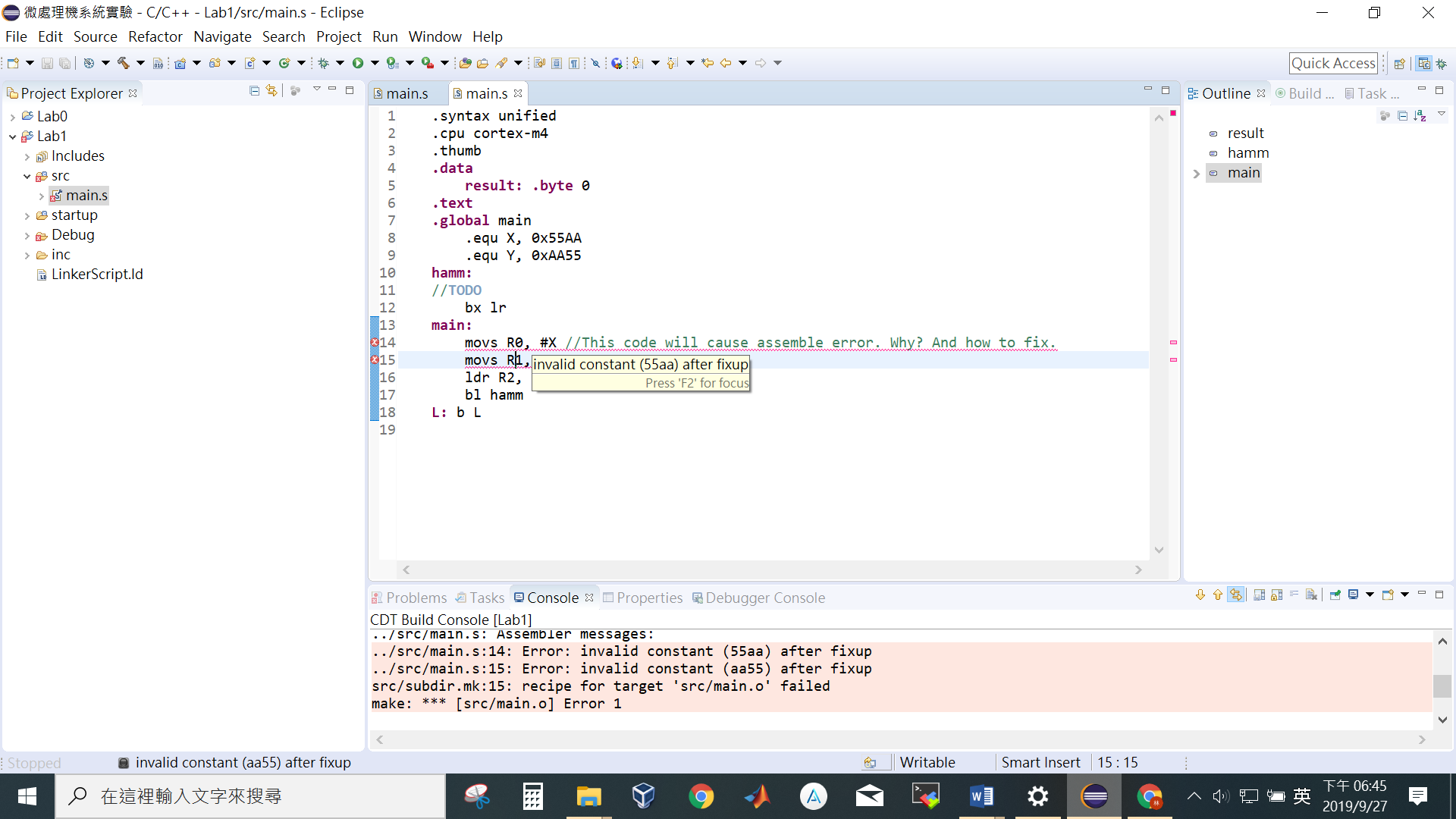
* 如何利用條件跳躍指令完成程式迴圈的操作
* 算數與邏輯操作指令使用
* 暫存器 Register
* 使用與基本函式參數傳遞
* 記憶體與陣列存取

1. **How** 實驗要怎麼做
2. **Hamming distance (30%)**

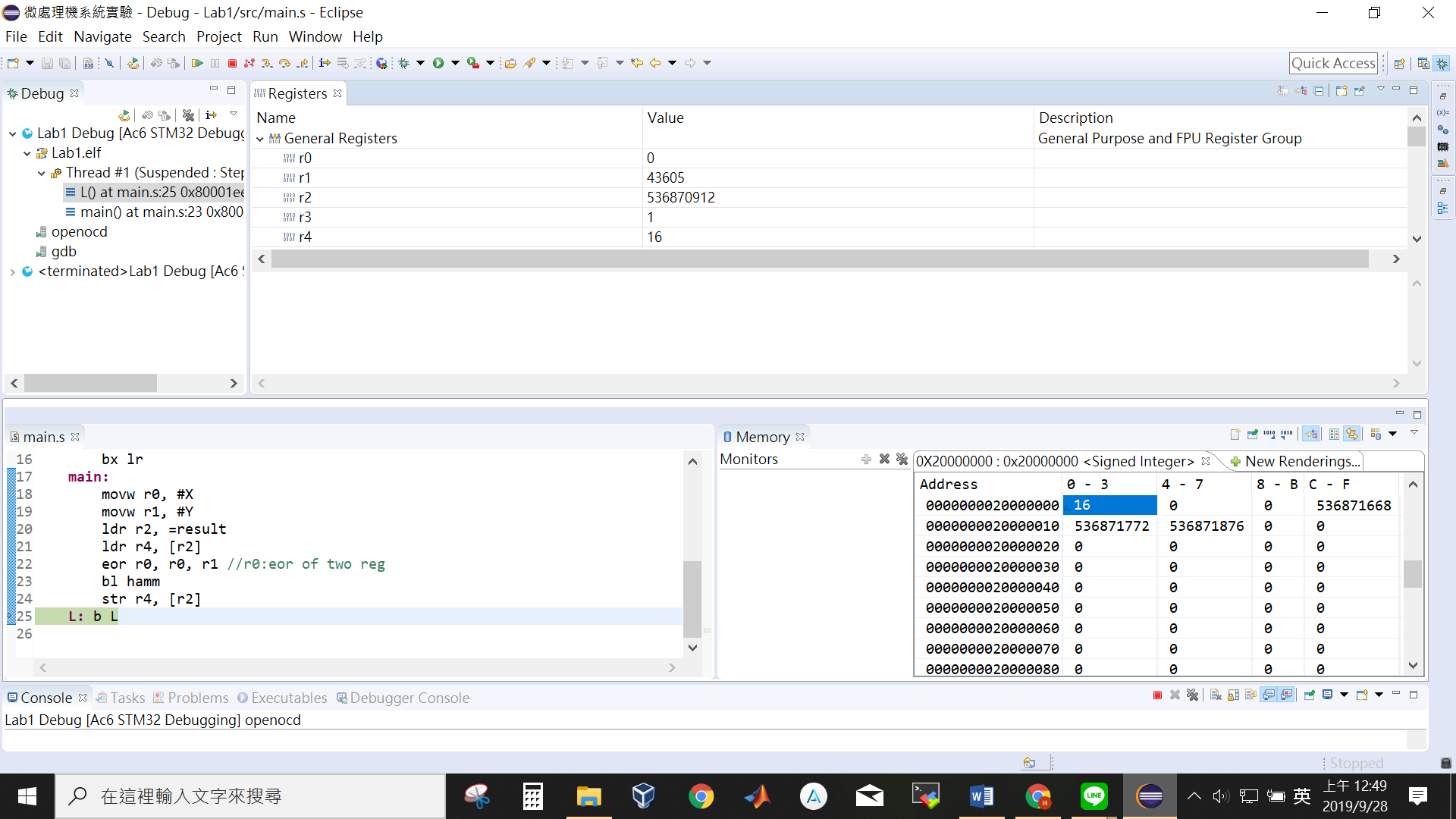
**計算兩個數長度為 half-word(2bytes) 的漢明距離，並將結果存放至 result 變**

**數中。修改 X, Y 並從 memory 看到結果**

0x55AA 和0xAA55這兩個值不是可用的常數，太大了。改用movt (upper 16 bits) 還有movw (lower 16 bits)即可。

因為ARM指令為32 bits,但是指令本身也需要bits來告訴CPU一些必需的資訊。所以資料數值的大小並不是2^32那麼大(實驗下來是8bits?)

hamming distance: 把兩個值做XOR之後的1的個數。計算一個二進制數字內1的個數方法:把該數字(r0)和1做and，把其結果累加(在r4)，並把該數字(r0)做shift right 1 bit，重複此過程直到該數字(r0)等於0。(代表沒有1了)

0x55AA 和0xAA55的hamming distance為16，存在0x20000000記憶體內。

1. 1
2. **Bubble sort (40%)**

**利用組合語言完成長度為 8byte 的 8bit 泡沫排序法。**以下為遞增排序。

r0: array data起始位置

r1: 排序[r0+r1]和[r0+r1+1]的data，直到r1=r2

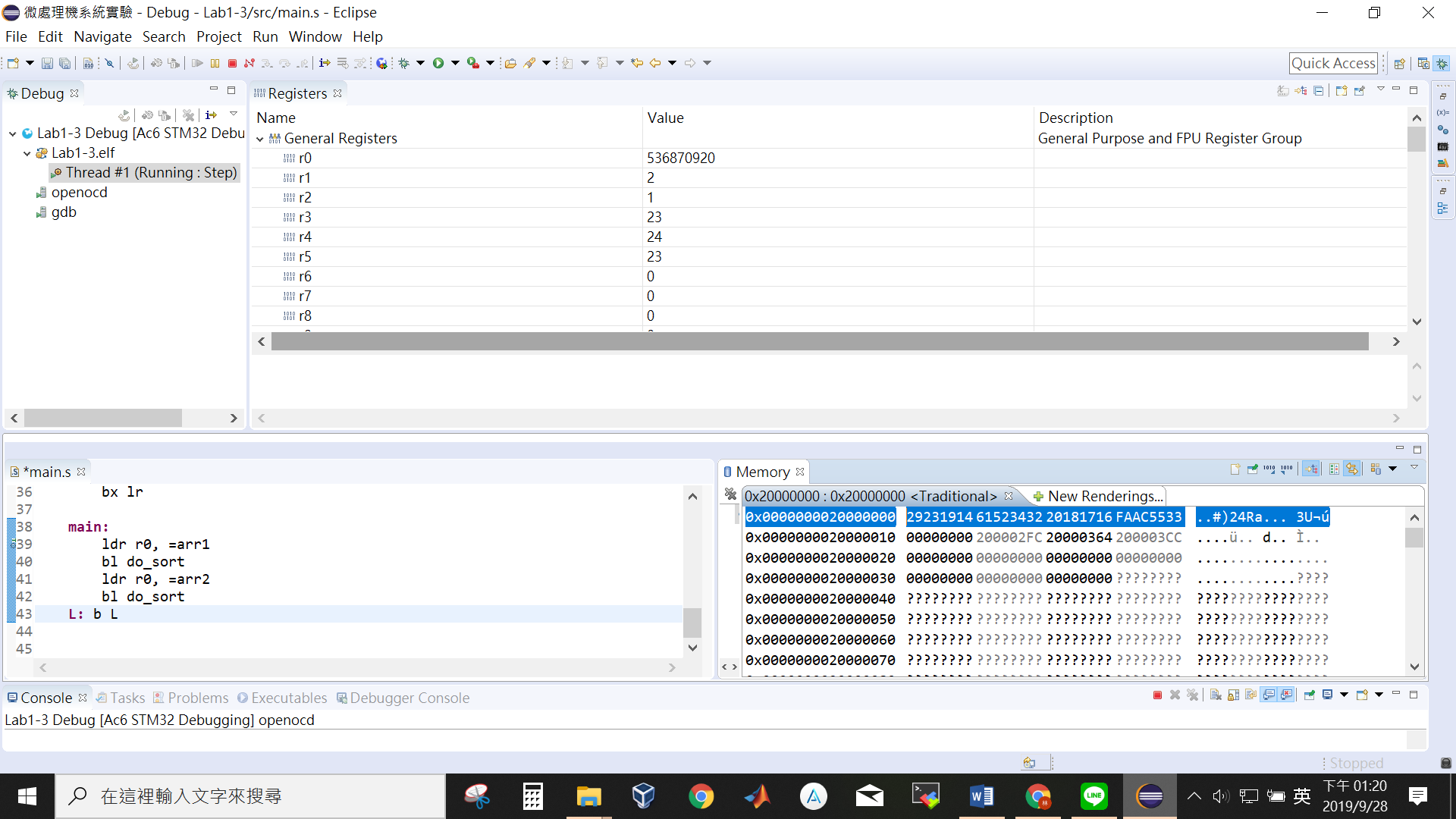
r2: 用來紀錄已排序好的data，使下次排序進行時不會動到它們，因為排序過程中，會保證每次出來的最右邊的數，是此次排序中最大的數。

r3 & r4:取出來要做比較的兩個data

r5: swap用

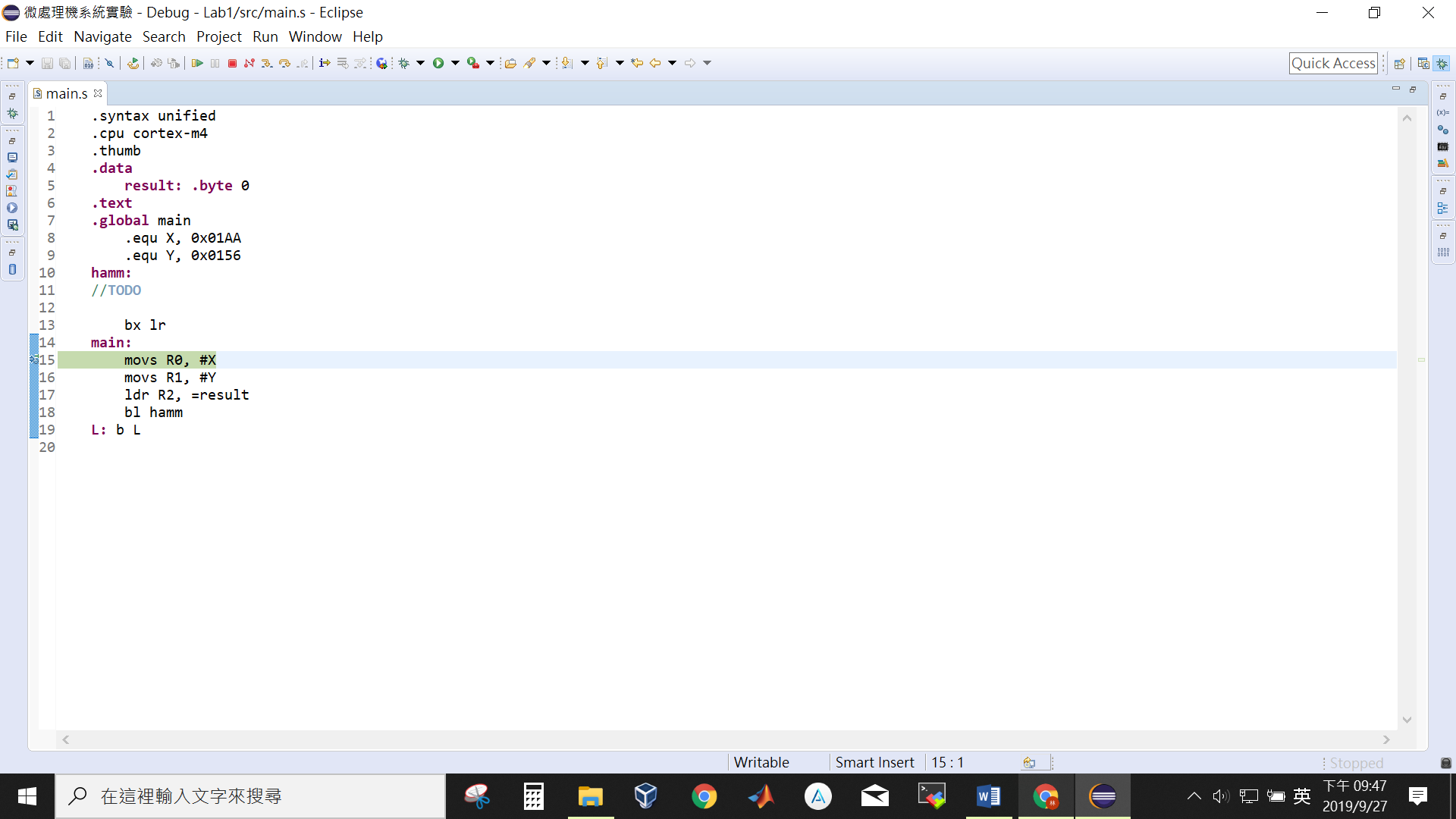
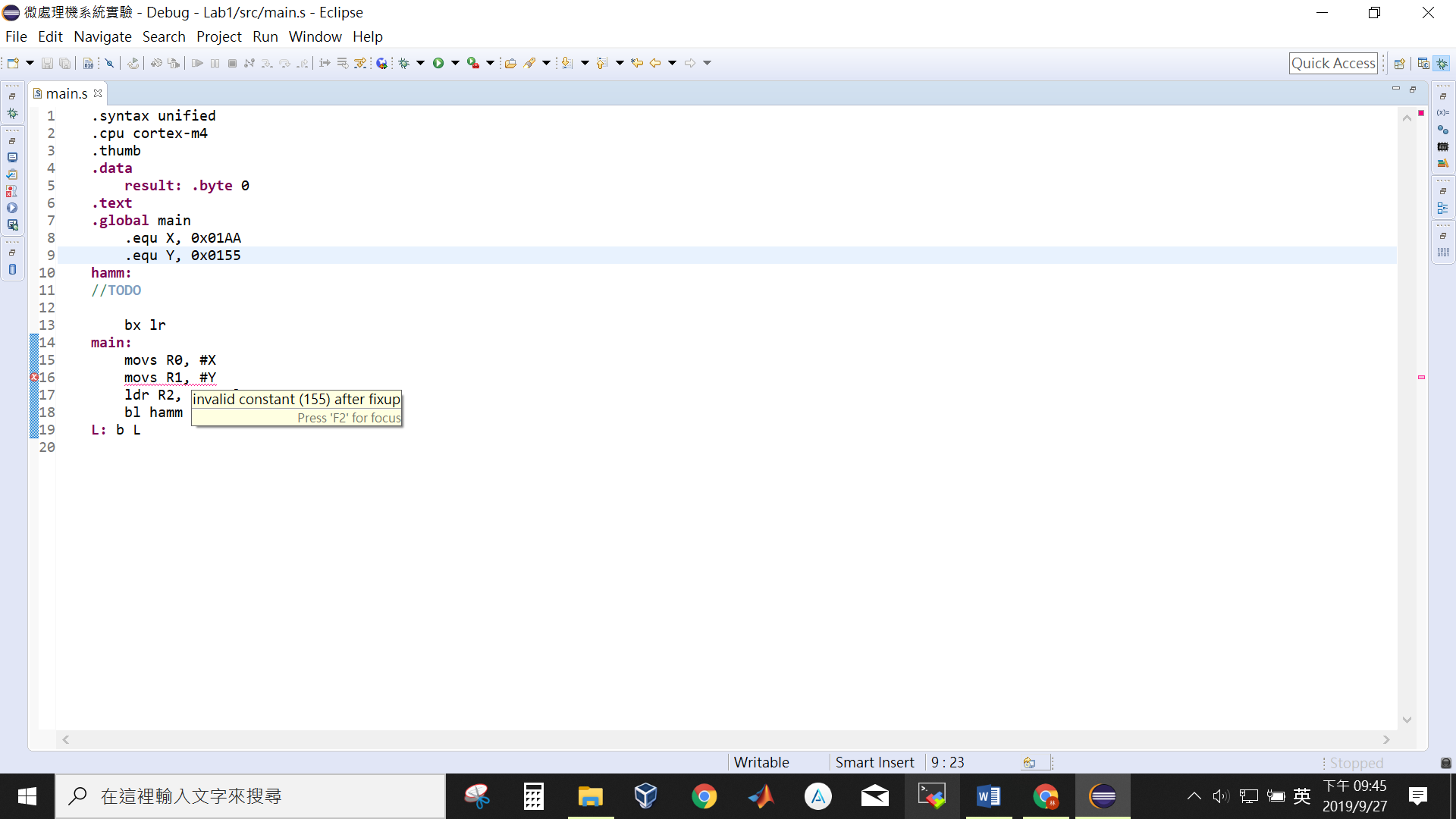
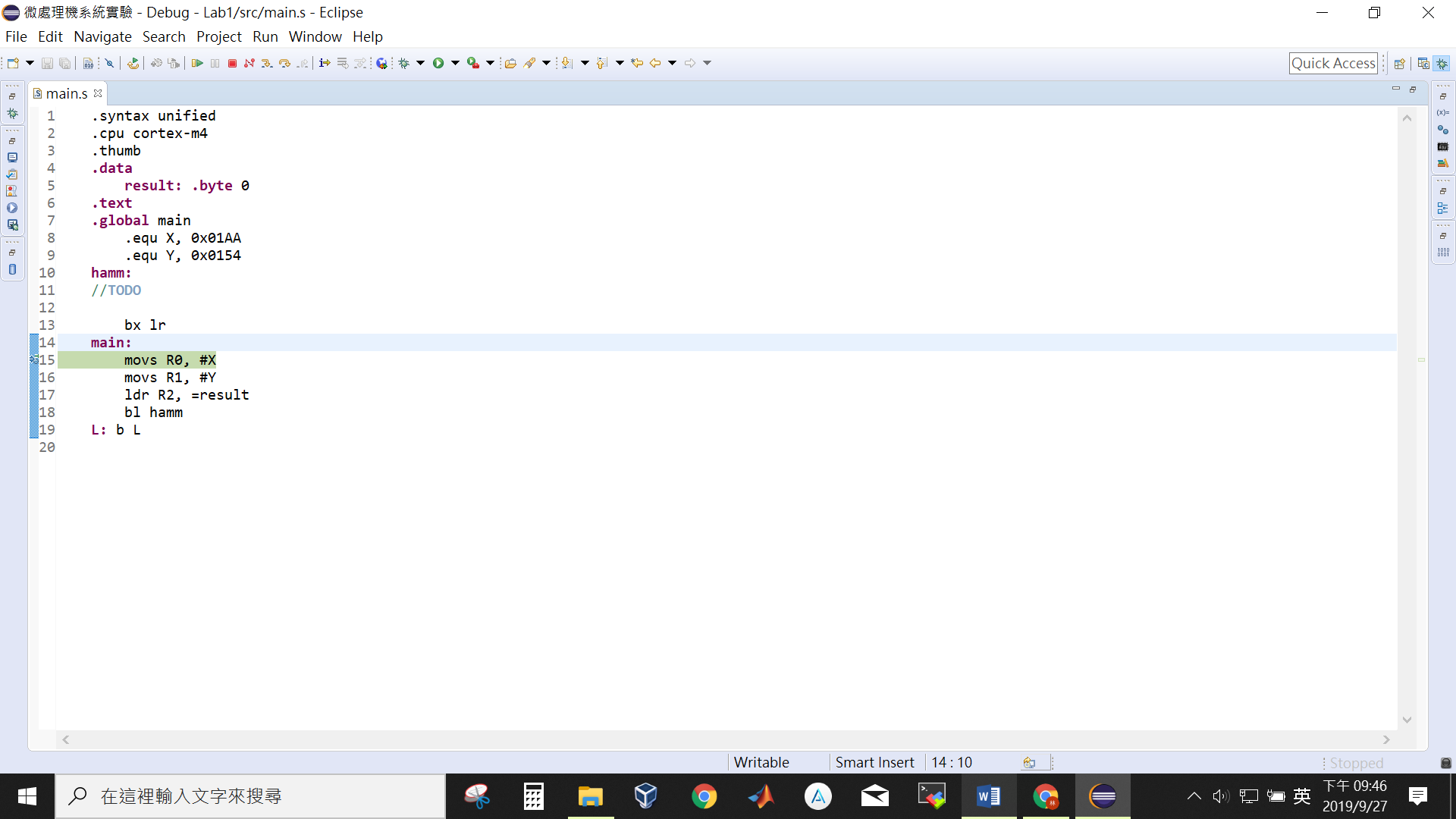
swap: 如果需要交換，再把換過的data用strb，存回去原本記憶體位置

最後在記憶體中的排序結果:



1. **Feedback** 實驗心得建議

* 0x0155是一個奇妙的數字，試mov最大位元值的時候這個數字剛剛好不能過



* 各種Branch指令 <https://www.itread01.com/content/1546939989.html>
* bx lr的作用等同於mov pc,lr (跳轉到lr中存放的地址處)

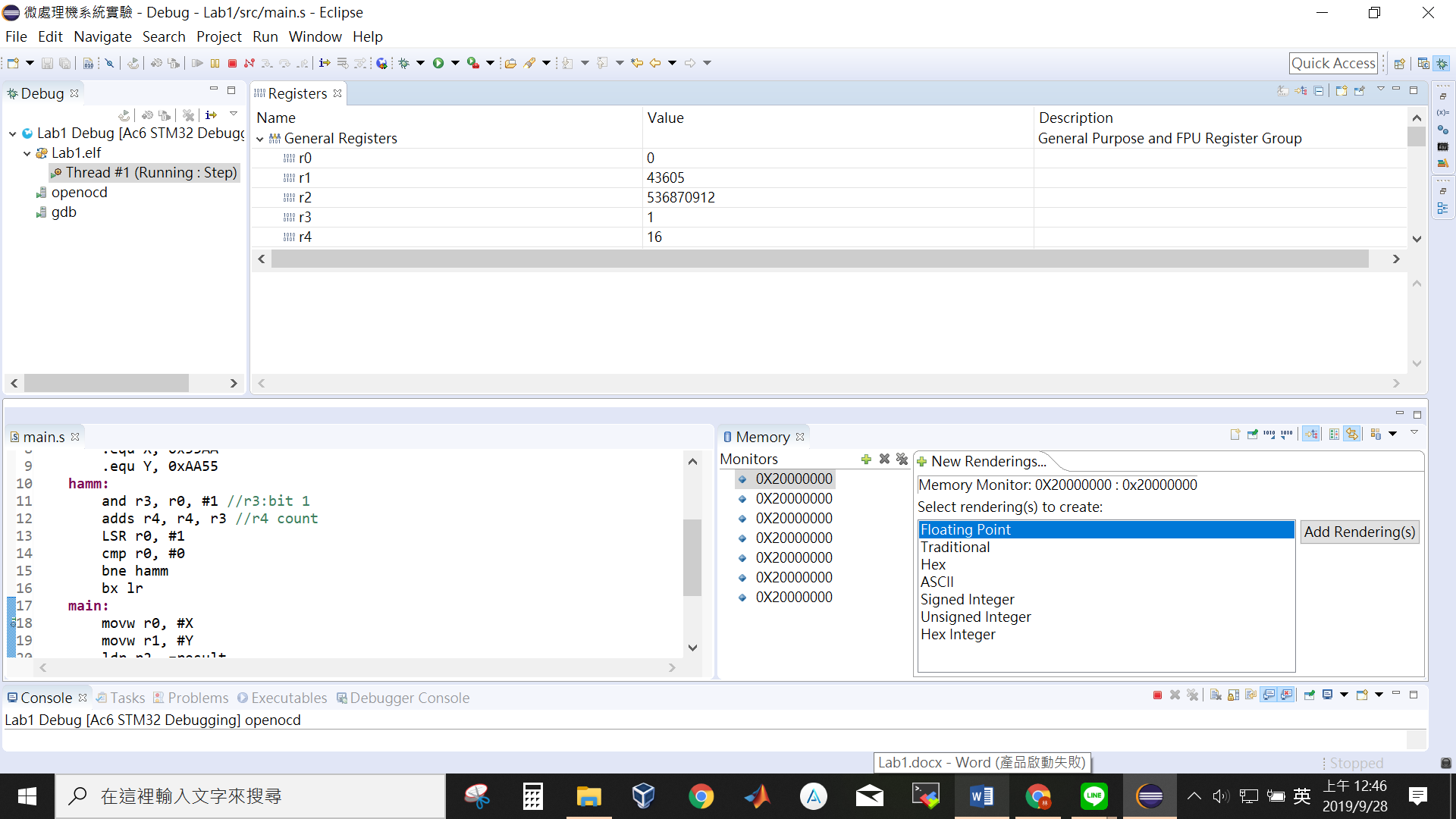
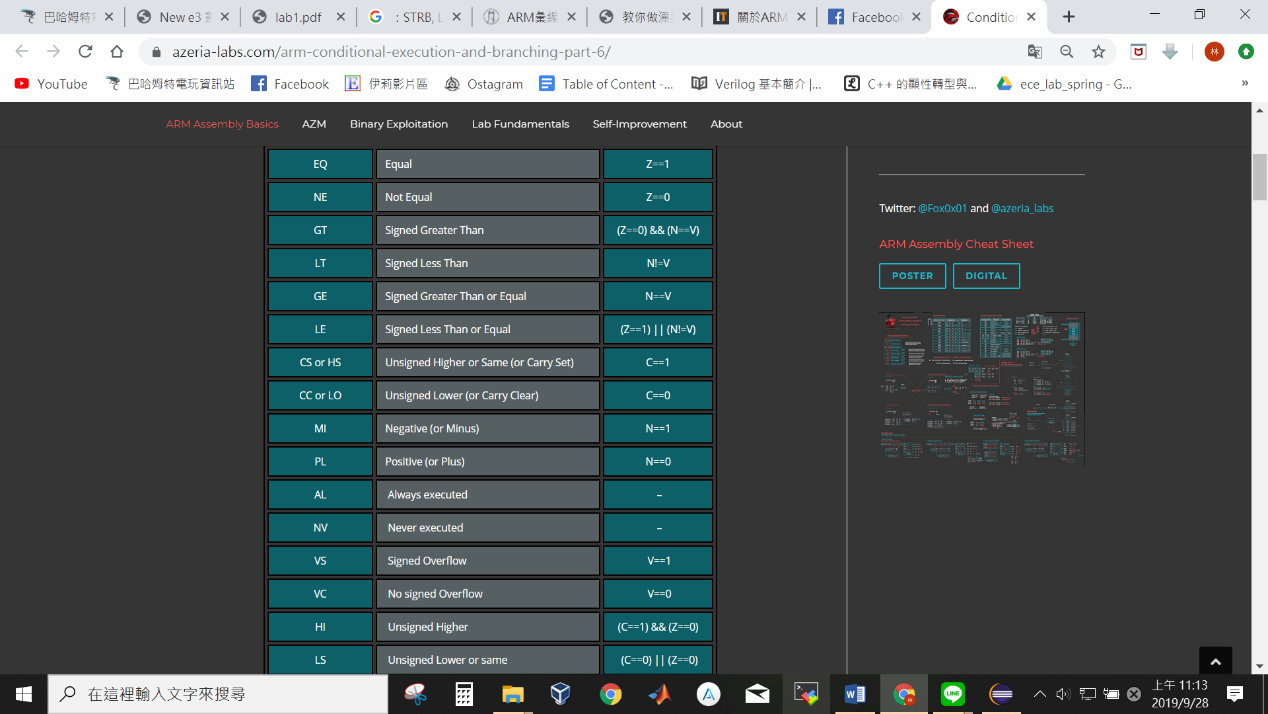
lr是連接寄存器(Link Register)，特殊用途有兩種：

一，用來保存子程序返回地址；先通過BL或BLX指令調用子程序時(bl …)，硬體會自動將子程序執行完後，要返回的地址保存在R14中。在子程序執行完要返回時，把LR的值複製到程序計數器PC (bx lr) 即可實現子程序返回

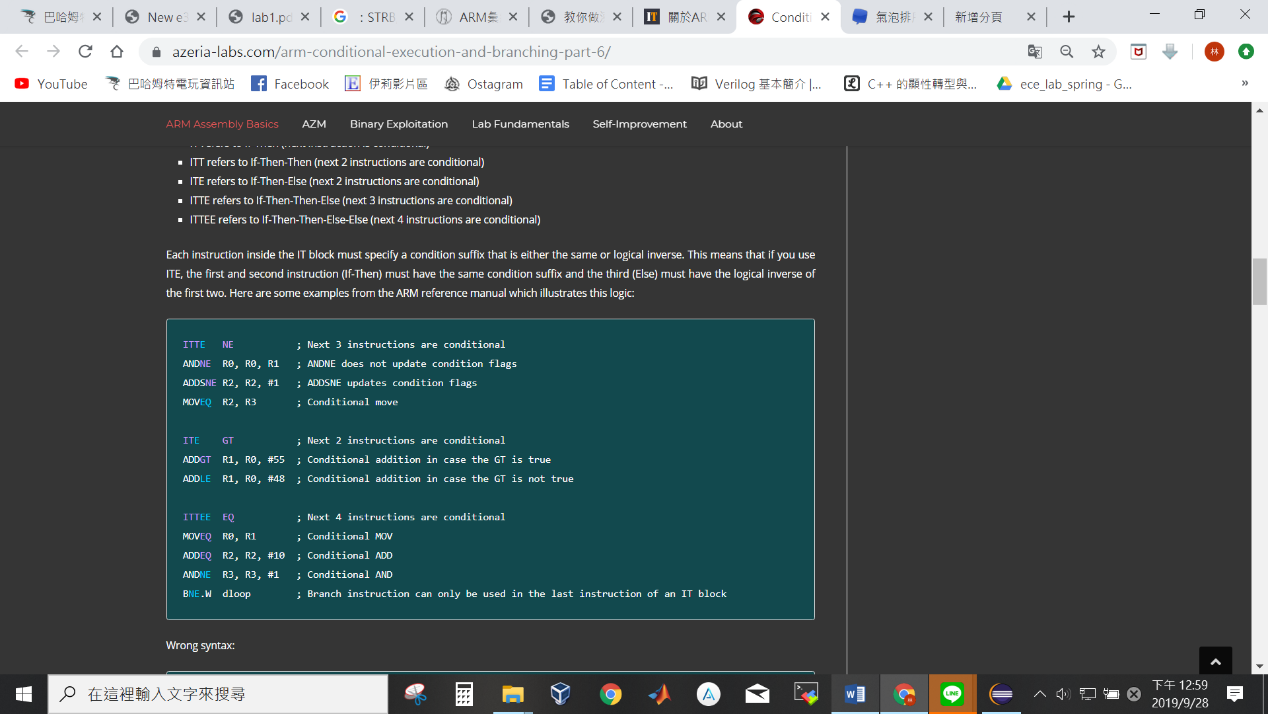
二、當異常發生時，LR值等於異常發生時PC的值減4（或者減2），因此在各種異常模式下，可以根據LR的值返回到異常發生前的相應位置繼續執行。

* cmp: 把兩個reg做compare(相減)，把4種flag存在APSR。加上CONDITIONAL EXECUTION之後的指令，會自己去拿它需要的資訊，例如bEQ, bNE會自己看現在的zf值，決定是否跳轉。註:像add**s**也會update flag，”s” 代表Status 。

APSR (Application Processor Status Register): 特別的reg，用來做各種條件判斷用。

* CF=1，有借位
* VF=1，溢位
* NF=1，結果是負數
* ZF=1，結果全是零
* 連續的CONDITIONAL EXECUTION：

要先告知有幾行。(ITE,ITTE,ITTEE)



* reg是4 bytes，要用ltrb & strb來取得特定byte內容，用法同ltr & str

ltrb r0, [<addr>,<shift>]

strb r0, [<addr>,<shift>]

* 直接使用數字，要加#
* load data address，要加=
* 把reg內容當位址來取值，要用[<reg>]