實驗二 實驗結報

0316213 蒲郁文 & 0316323 薛世恩

實驗名稱

ARM Assembly I

實驗目的

- 熟悉基本 ARMv7 組合語言語法使用
- 算術邏輯操作
- 暫存器使用及函式參數傳遞
- 記憶體與陣列存取
- 條件跳躍指令完成程式迴圈

實驗步驟

Hamming distance

```
@ ARM Cortex-M4 Documents:
@
http://infocenter.arm.com/help/topic/com.arm.doc.dui0553a/DUI0553A_cortex_m4_dgug.pdf

    .syntax unified
    .cpu cortex-m4
    .thumb

.data
    result: .byte 0

.text
```

```
    .global main
    .equ X, 0x55AA
    .equ Y, 0xAA55

hamming_distance:
    eor  r0, r1
    cmp  r0, #0x0
    beq  done

count_ones:
    add  r3, #0x1
    sub  r1, r0, #0x1
    ands r0, r1
    bne  count_ones

done:
    bx   lr

main:
    @ http://www.davespace.co.uk/arm/introduction-to-
arm/immediates.html
    ldr  r0, =X
    ldr  r1, =Y
    ldr  r2, =result
    ldr  r3, [r2]
    @ https://en.wikipedia.org/wiki/Hamming_distance
    bl   hamming_distance
    str  r3, [r2]

forever:

    b    forever
```

Fibonacci serial

```
.data
    ans: .word 1
.text
    .global main
    .equ N,20

fibb:
    cmp R0,#100
    bgt outdone
    cmp R0,#1
    blt outdone
    mov R5,#2
    cmp R0,#3
    blt done

fib:
    mov R1,R2
    mov R2,R4
    add R4,R1,R2
    add R5,#1
    cmp R5,R0
    bne fib

done:
    bx lr
outdone:
    ldr R4,=0xFFFFFFFF
    bx lr

main:
    movs R0,#N
    movs R1,#1      @ R1=1
    movs R2,#1   @ R2=1
    movs R4,#1      @ R4=1

    bl fibb
```

```
    ldr R7,=ans
    str R4,[R7]

L: b L
```

Bubble sort

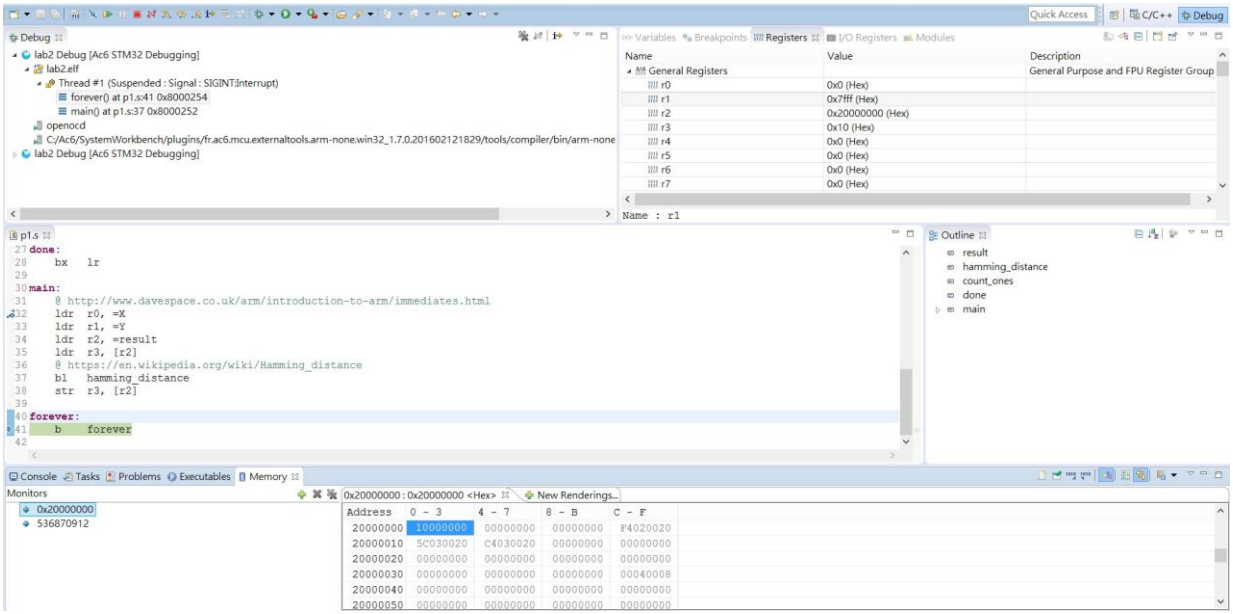
```
    .syntax unified
    .cpu cortex-m4
    .thumb

.data
    arr1: .byte 0x19, 0x34, 0x14, 0x32, 0x52, 0x23, 0x61, 0x29
    arr2: .byte 0x18, 0x17, 0x33, 0x16, 0xFA, 0x20, 0x55, 0xAC
    len:  .byte 0x08

.text
    .global main

bubble_sort:
    @ r2: iterator of outer loop, i
    mov  r2, #0x0
    push {lr}

start_outer:
    @ condition: i < len - 1
    sub  r4, r1, #0x1
    cmp  r2, r4
    bge  end_outer

    @ r3: iterator of inner loop, j
    mov  r3, #0x0

start_inner:
    @ condition: j < len - 1 - i
    sub  r5, r4, r2
    cmp  r3, r5
    bge  end_inner
```

```
    @ if arr[j] > arr[j+1] then swap
    ldrb r5, [r0, r3]
    add  r7, r3, #0x1
    ldrb r6, [r0, r7]
    cmp  r5, r6
    it   gt
    blgt swap

    add  r3, r3, #0x1
    b    start_inner

end_inner:
    add  r2, r2, #0x1
    b    start_outer

end_outer:
    pop  {pc}

swap:
    strb r5, [r0, r7]
    strb r6, [r0, r3]
    bx   lr

main:
    ldr  r0, =arr1
    ldr  r2, =len
    ldr  r1, [r2]
    @ http://www.iis.sinica.edu.tw/~cmwang/arm/Lecture08.ppt
    @ r0: array pointer
    @ r1: array length
    bl   bubble_sort
    ldr  r0, =arr2
    bl   bubble_sort

forever:

    b    forever
```

實驗結果與問題回答

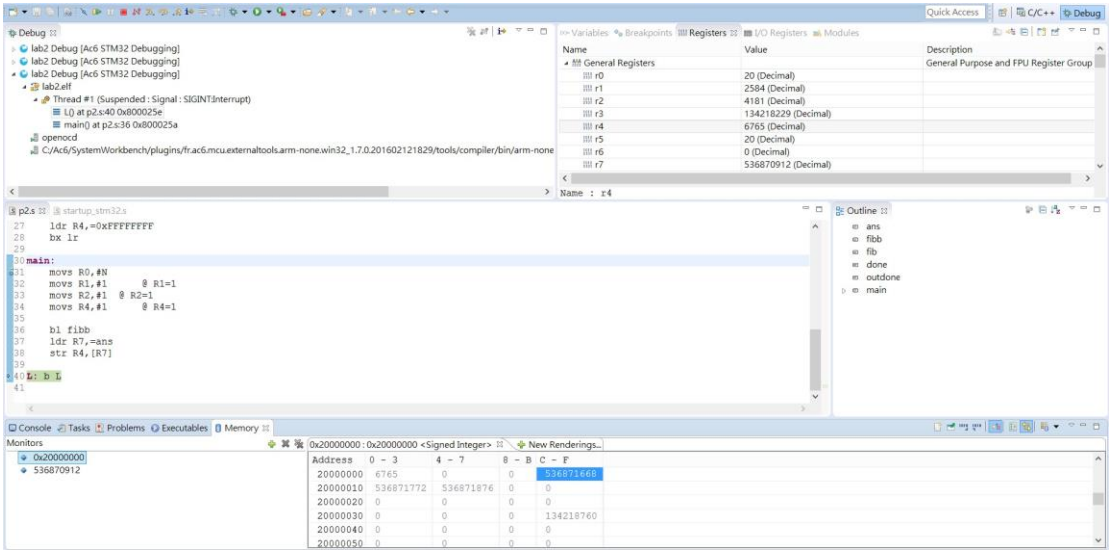
Hamming distance

先將 X 與 Y 做 XOR，再用 algorithm of Wegner，將數字中的 1 逐一消除，並計算有幾個 1。0xAA55 與 0x55AA 有兩個 byte 故不能 movs R0, #X，解決方法是改成 ldr R0, =X。



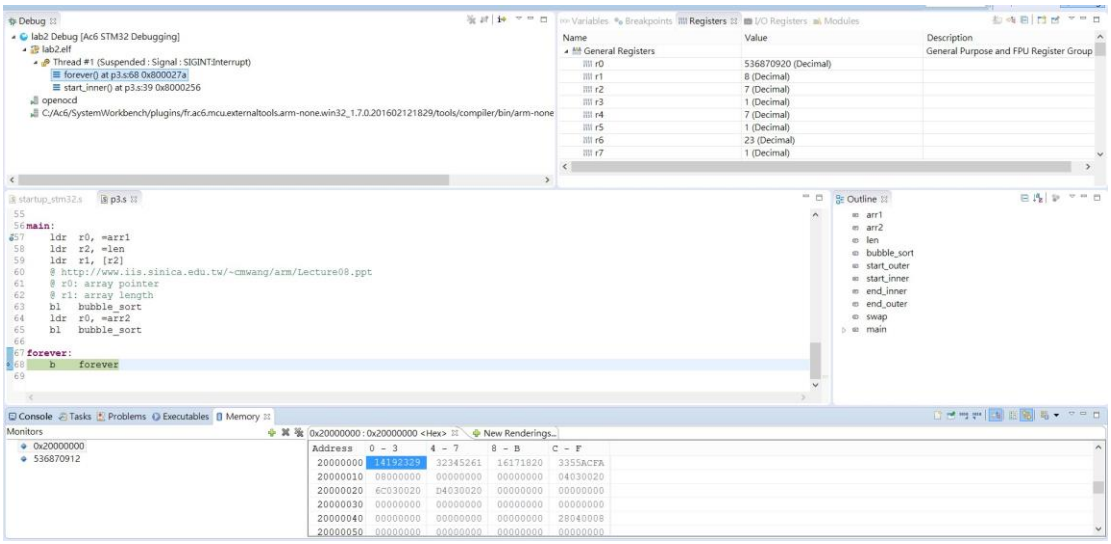
Fibonacci serial

先假設已經做好兩項(第一項=>1，第二項=>1)，並判斷他是否要數列第三個之後的數字，以及 N 是否在 1 到 100 之間。開始慢慢的平移然後加。在遇到邊界的時候停下來，就是答案。



Bubble sort

bubble_sort 函式會做兩層迴圈，內層迴圈還會再呼叫 swap 函式。因此會用 push 跟 pop 將 linking register 存進 stack 之中。



心得討論與應用聯想

- 我們很開心學習到如何使用判斷的條件。
- 這樣以後就不用慢慢算 Fibonacci 數列了，也不用有一大堆數字慢慢排序了，一整串 01 我也知道有幾個不一樣了呢！
- 我覺得我以後不用買計算機(工程用)了，自己寫比較有志氣。