0616018 林哲宇, 0616032 張哲銓

1. Hamming distance
   a. 修改 X, Y 並從 memory 看到結果

   A: 把 X, Y 值修改並且觀察 memory，答案是正確的。

```
.data
    result: .byte 0
    XYxor: .word
.text
.global main
    .equ X, 0x55AA
    .equ Y, 0xAA55
rotate_bits:
    // do "and gate" to determine whether to add 1
    ldr r2, =XYxor
    ldr r1, [r2]
    cmp r1, 0
    beq end_hamm
    movs r2, #1
    and r1, r1, r2
    // add and store the value to result
    ldr r2, =result
    ldrb r3, [r2]
    add r3, r3, r1
    strb r3, [r2]
    // divide XYxor by 2 then loop to rotate_bits
    ldr r2, =XYxor
    ldr r1, [r2]
    lsr r1, 1
    str r1, [r2]
    b rotate_bits
hamm:
    // XYxor = X ^ Y and it should be divided to 0 at last to calculate hamming distance
    eor r1, r1, r0
    ldr r2, =XYxor
    str r1, [r2]
    bl rotate_bits
    end_hamm:
        bx lr
main:
    ldr r0, =#X //This code will cause assemble error. Why? And how to fix.
    ldr r1, =#Y
    ldr r2, =result
    bl hamm
L: b L
```

主要想法就是把兩個數做 xor 之後，要看裡面有幾個 bits 是 1。我們計算幾個 bits 的方式是，先把 xor 的結果和 1 做 and，如果是 1，則 result 就加 1，然後把 xor 的結果右移一位。重複上述動作直到 xor 的結果為 0。

中途遇到的困難有

(1) 不小心把 memory 關掉了。感謝 stackoverflow 大大說要按 alt + shift + Q。

(2) 不知道怎麼設斷點。再次感謝 stackoverflow 大大說按 shift + ctrl + B。

(3) 程式有 loop，一直按 F5 很累，東試西試之後才發現按 F8 可以 continue。

2. Fibonacci serial

a. 修改 N 並從 register 看到結果

A: 根據題目要求，把答案放到 register r4 上，如果 overflow 則為 -2 ，N<1 或 N>100 則為 -1。利用 debugger 右上角的 Registers 選項觀察

b. 怎麼去偵測 overflow ?

A: 利用 cmp 和 blt, bgt 來判斷是否小於一或大於一百。

```
4 .text
5 .global main
6     .equ N, 100
7 fib:
8     // jump to exceedN if N < 1 or N > 100
9     cmp r0, 1
10    blt exceedN
11    cmp r0, 100
12    bgt exceedN
13    // fibonacci start
14    movs r1, 1
15    movs r2, 1
16    cmp r0, 2
17    ble N_1_2
18    loop:
19        adds r3, r1, r2
20        cmp r3, r2
21        blt overflow
22        movs r1, r2
23        movs r2, r3
24        subs r0, r0, 1
25        cmp r0, 2
26        bne loop
27    movs r4, r2
28    b endfib
29 N_1_2:
30    movs r4, 1
31    b endfib
32 exceedN:
33    movs r4, -1
34    b endfib
35 overflow:
36    movs r4, -2
37    b endfib
38 endfib:
39    bx lr
40 main:
41    movs R0, #N
42    bl fib
43 L: b L
```

把 r1, r2 當作費氏數列的兩個數，如果 N < 2，就直接回傳 1。如果 N > 2，則 r1 跟 r2 相加後給 r3，判斷有沒有 overflow 之後，把 r0 減一，如此循環值到 r2 等於 2 才離開迴圈。

3. Bubble sort

a. 看 arr1, arr2 的排序結果

A: 最後結果為由小排到大

b. 修改程式 (題目在 demo 時公布)

```
.data
    arr1: .byte 0x19, 0x34, 0x14, 0x32, 0x52, 0x23, 0x61, 0x29
    arr2: .byte 0x18, 0x17, 0x33, 0x16, 0xFA, 0x20, 0x55, 0xAC
.text
.global main
change_place:
    strb r3, [r6]
    strb r4, [r5]
    b loop2_cmp
do_sort:
    // r1, r2 are loop counters
    movs r1, 0
    loop1:
        movs r2, 0
        loop2:
            // r3 is the first value, and r4 is the second. r5, r6 are addresses of arr
            add r5, r0, r2
            ldrb r3, [r5]
            add r2, r2, 1
            add r6, r0, r2
            ldrb r4, [r6]
            // if r4 < r3 then change the place.
            cmp r3, r4
            bgt change_place
            // if r2 < 7 then continue loop2
            loop2_cmp:
                cmp r2, 7
                bne loop2
        // if r1 < 7 then continue loop1
        add r1, r1, 1
        cmp r1, 7
        beq end_loop1
        b loop1
    end_loop1:
        bx lr
main:
    ldr r0, =arr1
    bl do_sort
    ldr r0, =arr2
    bl do_sort
L: b L

```

把 r1, r2 分別當作雙重迴圈中的兩個計數器，r5, r6 則用來存放 arr 的位址，以便之後 ldrb 和 strb 的時候方便使用。接著就是把陣列中第一個參數和第二個參數比，如果第一個比較大，就交換位置，再來把第二個參數和第三個參數比，以此類推，直到第七個參數和第八個參數比為止。以上的動作要重複八次才能保證正確。