Lab1 ARM Assembly I

0616018 林哲宇

1. What 實驗要做什麼

這個實驗是要用 ARM Assembly 來寫出三支程式，分別為 Hamming distance, Fibonacci serial, Bubble sort。

其中 Hamming distance 的題目內容是會給兩個 2 bytes 數字，求出兩個數字的漢明距離。也就是說，要把兩個數的二進制做比較，算出不一樣的 bit 數，最後存到變數 result 中。

Fibonacci serial 則是給一個 1～100 的數字，最後求出費氏數列的那一項並存到 r4 暫存器中。

最後是 Bubble sort，給兩個陣列，要把它們從小到大或從大到小排列。

2. How 實驗要怎麼做

我是有好好做完 Lab0 才開始做 Lab1 的，所以基本的 debugger 用法大致了解，不過還是遇到了一些坑，以下：

(1) 不小心把 memory 的觀察視窗關掉了。之後查資料後才知道按 alt + shift + Q 可以叫出 options。

(2) 不知道怎麼設斷點。查資料後知道在要設斷點的那一行按

shift + ctrl + B，就可以設斷點了。

(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。

再來是這三題的解決過程，

Hamming distance：

這是助教提供的部分 code

```
      movs R0, #X //This code will cause assemble error. Why? And how
to fix.
```

試著跑了一次，發現真的會出錯。之後把數字調成 1 byte 似乎就沒問題，但是 2 bytes 以上就會噴錯，於是最後的改法是

```
    ldr r0, =#X //This code will cause assemble error. Why? And how to fix.
    ldr r1, =#Y
```

接著要把兩個數字做 xor，才可以算出兩個數的漢明距離，查資料後知道 arm 的 xor 是 eor 指令。再來就要開始算有哪幾個 bit 是 1 了，我們計算幾個 bits 的方式是，

```
12 rotate_bits:
13     // do "and gate" to determine whether to add 1
14     ldr r2, =XYxor
15     ldr r1, [r2]
16     cmp r1, 0
17     beq end_hamm
18     movs r2, #1
19     and r1, r1, r2
20     // add and store the value to result
21     ldr r2, =result
22     ldrb r3, [r2]
23     add r3, r3, r1
24     strb r3, [r2]
25     // divide XYxor by 2 then loop to rotate_bits
26     ldr r2, =XYxor
27     ldr r1, [r2]
28     lsr r1, 1
29     str r1, [r2]
30     b rotate_bits
```

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。

Fibonacci serial :

這題有三種情況，第一種是輸入的 N 小於一或大於一百，這個情況要讓最後的結果 r4 等於-1。

```
// jump to exceedN if N < 1 or N > 100
cmp r0, 1
blt exceedN
cmp r0, 100
bgt exceedN
```

利用 cmp 和 blt, bgt 來判斷 N 是否小於一或大於一百。

第二種是正常情況，一開始先把 r1, r2 兩個暫存器當作費氏數列的兩個數，如果 N <= 2，就直接回傳 1。

```
// fibonacci start
movs r1, 1
movs r2, 1
cmp r0, 2
ble N_1_2
```

如果 N > 2，則 r1 跟 r2 相加後給 r3，判斷有沒有 overflow 之後(如果有就是第三種情況)，把 r0 減一，如此循環值到 r0 等於 2 才離開迴圈。

```
loop:
    adds r3, r1, r2
    cmp r3, r2
    blt overflow
    movs r1, r2
    movs r2, r3
    subs r0, r0, 1
    cmp r0, 2
    bne loop
```

第三種情況就是前面所提到的 overflow，如果發生 overflow，得到

的 r3 應該會變成負數，也就是會小於費氏數列的上一項，所以透過 cmp 比較之後，就可以判斷是否 overflow 並回傳-2

Bubble sort:

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 的開始位址，以便之後 ldrb 和 strb 的時候方便使用。

```
// 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
```

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，交換位置的程式碼如下

```
10 change_place:
11     strb r3, [r6]
12     strb r4, [r5]
13     b loop2_cmp
```

利用 strb 蓋掉位址的那個 byte，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就能保證正確。

3. Feedback 實驗心得或建議

能夠用 assembly 寫程式還挺有成就感的，Lab1 就這麼有趣，讓我有點期待接下來的 Lab 會怎樣。以前就想學些底層的東西，然而因為實做不出什麼，所以也沒特別有動力去做，希望能在這堂課學到更多底層的知識。