**組合語言**

**Final Project**

班級：資訊二甲

組別：第19組

組員：

10627116許逸翔

10627125王雲和

(一)設計說明：

參照 gcc 練習，熟悉組合語言以完成 Final project。

練習內容：

1. 熟悉fp與sp，完成local variable的宣告，還有stack中資料以及所儲存支位置，以改寫出有符合期中作業規定的midterm project 和符合期末作業規定的Final project，並且，為了方便閱讀與理解，在一開始都會有加上變數與fp的對照表，供programmer 使用。

2. 傳指標以及傳參數，以期實現main, id 與 name的參數傳遞。

3. 配置連續的記憶體給name和id以達成陣列形式，使其能以迴圈方式撰寫，方便存取及應用。

(二)設計重點說明：

**main.c:**

閱讀完範例main.c之後進行改寫，以符合我們所寫之id.s與name.s。main.s作為程式的主體，需要配和參數的傳遞並且在呼叫function時給予callee所需要的數值，並且接收回傳值。

**name.s**:

1. 配置連續記體，由於程式將以迴圈撰寫，需要使所有字串呈現相同大小 ，讓之後在做使用時可以移動相同數量的byte拿到下筆字串，故，以空白填補字串不足的地方。

2. 由於陣列沒有assign，因此需要使用strcpy來達成字串的複製。

3. 將add r4, r4, #1改寫成adcs r0, r1, r2，便可達成期中project的要求。 因為在這之前有printf，所以C bit為1，需要將r1設置為r4，r2設置為0，最後再將r4設為 r0。

**id.s:**

1. 配置連續記憶體，因為int的大小固定為4Bytes，可以用operand2 中的LSL#2來達成偏移量。

2. 在id輸入的部分有做防呆裝置，如果輸入數值小於零，利用conditional execution，做出忽略中間的i++，以及store id的值，讓程式能正常執行。

3. 在判斷輸入'p'的地方，有做防呆，若輸入不為’p’，則會用branch重入讀入直到’p’為止，在判斷’p’的地方要注意ldrb，不然會發生奇妙的錯誤。

**drawJuliaSet.s:**

1. 因為發現部分立即值無法直接指定，所以將不能直接指定的值存放在DATA區段，需要時再以load label與dereference的方式做存取。

2. 改寫自Ccode，為了加速部分程式進行，選擇不使用local variable，改用register 當經常使用的變數，但是因此需要特別注意該暫存器的使用是否有被修改到。

3. 除法的部分因為有使用除法函數，所以不需要自己建立除法演算法，來拖慢自己的速度。

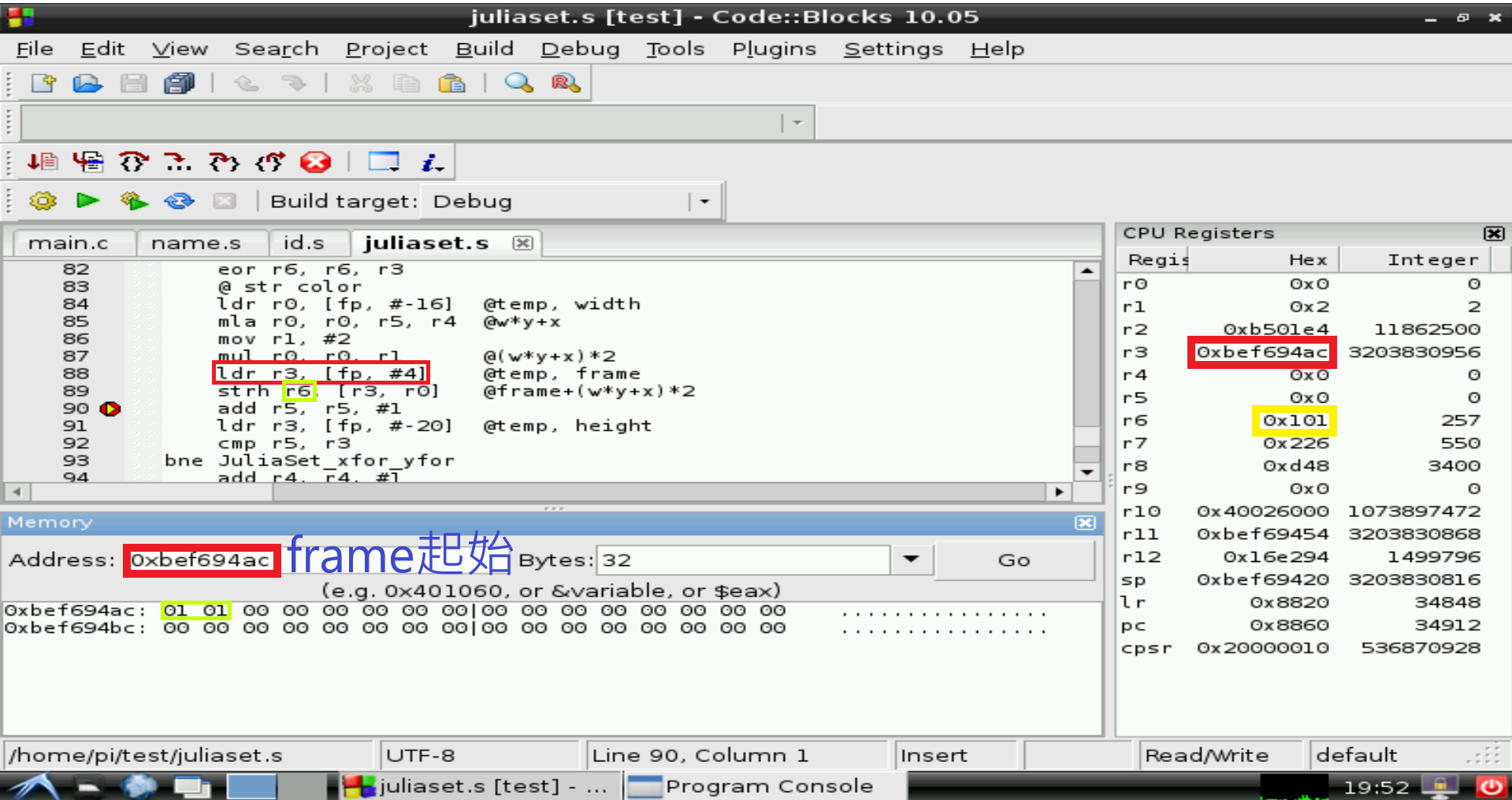
4. 將sub sp, fp, #4改寫成sbcs fp, r0, r1，以符合期末作業的要求，將r0設定為fp，利用conditional execution，判斷當時的C bit 的值為何，調整r1的值，使其與C bit等數值，在計算過後的最終值為4，完成後將計算後的fp去更新sp。

5. 因為i跟0xff做and之後只剩下8bit，左旋8bit之後相當於挪出一個(i&0xff)的位置，做or的結果會與相加結果相同，故最終以add r6, r6, r6 lsl #8 代替原有指令，再以mvn進行反向。又因為最後將執行strb的關系，所以(&0xffff)這個部分可以直接忽略，以期能加快執行速度。

(三)程式驗證結果、指定之說明項目、與相關說明之螢幕截圖，並加以解釋說明：

**-----------------------------** **JuliaSet.s: -----------------------------**

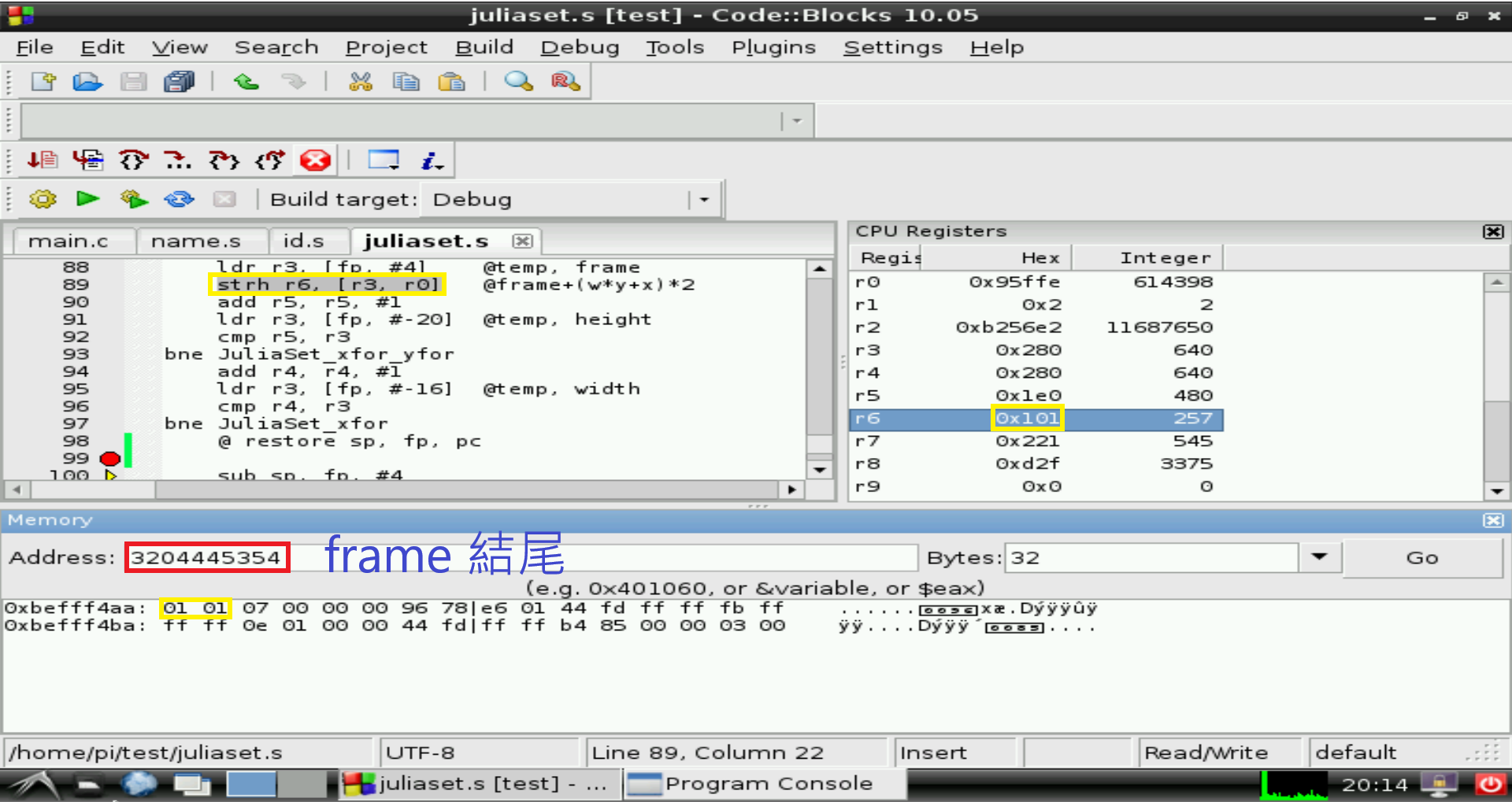
**frame\_起始：**

****

圖中 ldr r3, [ fp, #4 ] 是將frame起始位置load 進r3，而中斷點設置在第90行是希望去檢查是否有取錯位置。從黃色框框內的資料來看，可以察覺r6的資料確實有存進0xbef694ac，位置並無取錯。

又因此次中斷是第一次執行到這個位置，x, y為零，所以這個位置為frame + 0，故

0xbef694ac為 frame 之起始位置

**frame\_結尾：**

由於三層迴圈的關係，不便取得frame結尾的位置，故以手算的值當位置取值與r6。

r6的值可用原因為，從最後一次存入資料到break point之間 r6未做任何修改，故r6是frame結尾中之數值。

手算位置的方式：

從上圖可知一列有640筆資料，一筆資料2 bytes，故一列有1280 bytes。

而一共有480個列，故1280 \* 480 = 614400 bytes ( 307200 筆資料)

又array 由0起算，所以結尾是 307199，將其乘二( 2 bytes )，便可得 614398，加上frame 的 3203830956，得3204445354，該位置中的值與r6相符。

若要從frame中取得任意一格的位置，整理後可得公式為：

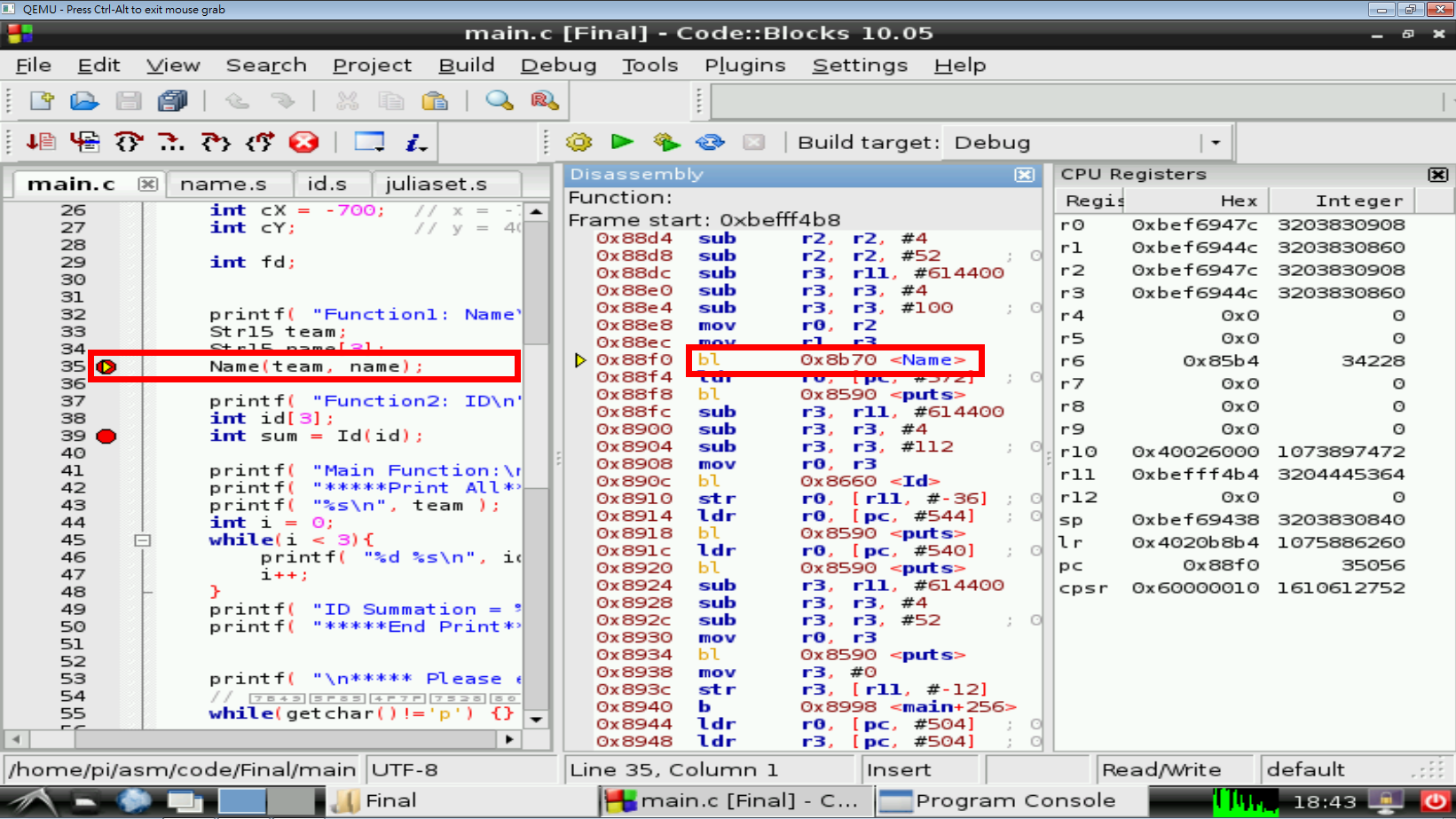
frame + ( width \* y + x ) \* 2

因為drawJuliaSet 的外兩層迴圈是 x < width, y < height

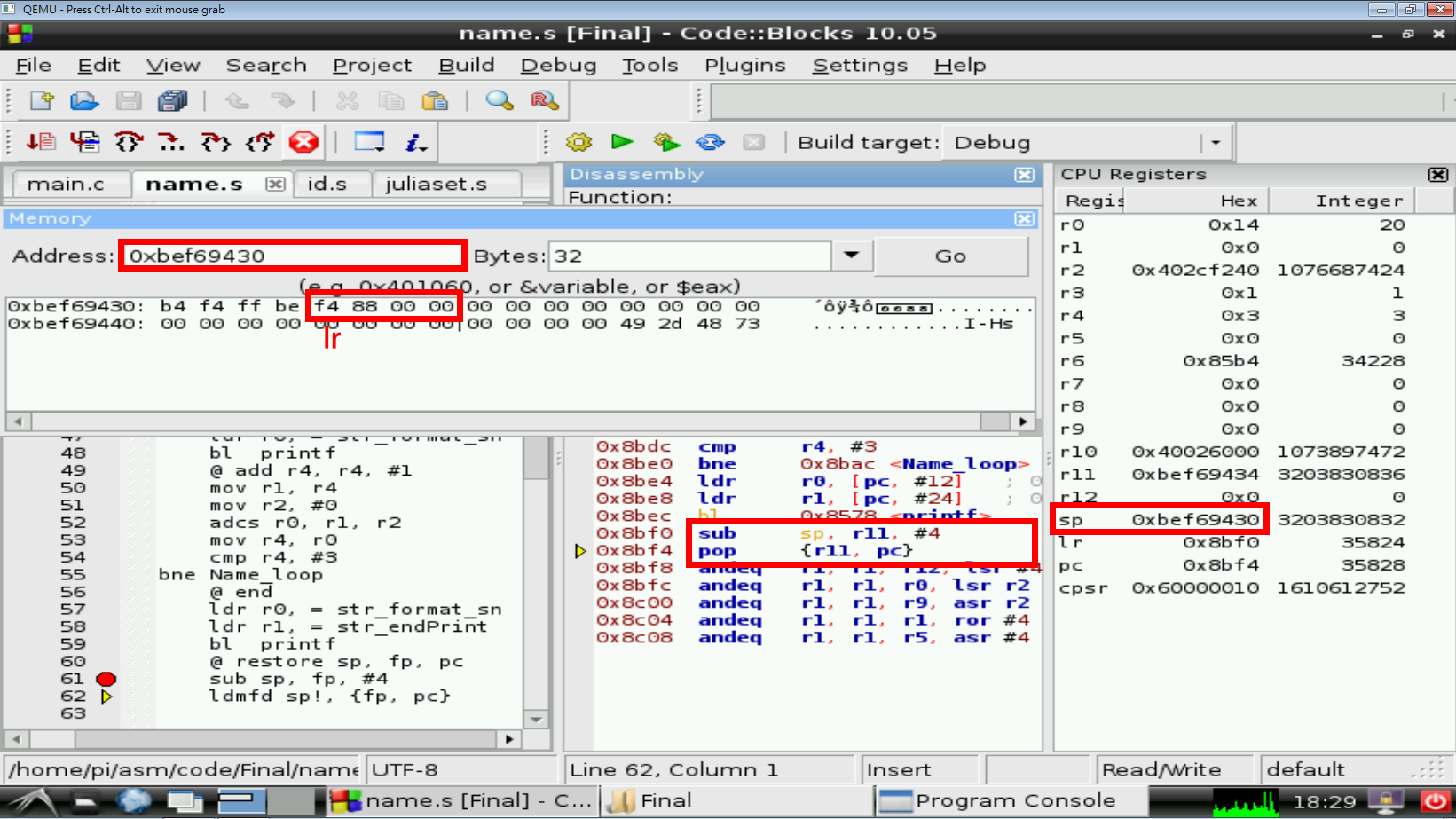
所以在執行到圖中位置時，代表 x, y以達 frame[height-1][width-1]，也就是frame的結尾，故將第99行作為中斷點( 此時結尾的值應該要被修改 )

**-------------------------------** **main.c: -------------------------------**

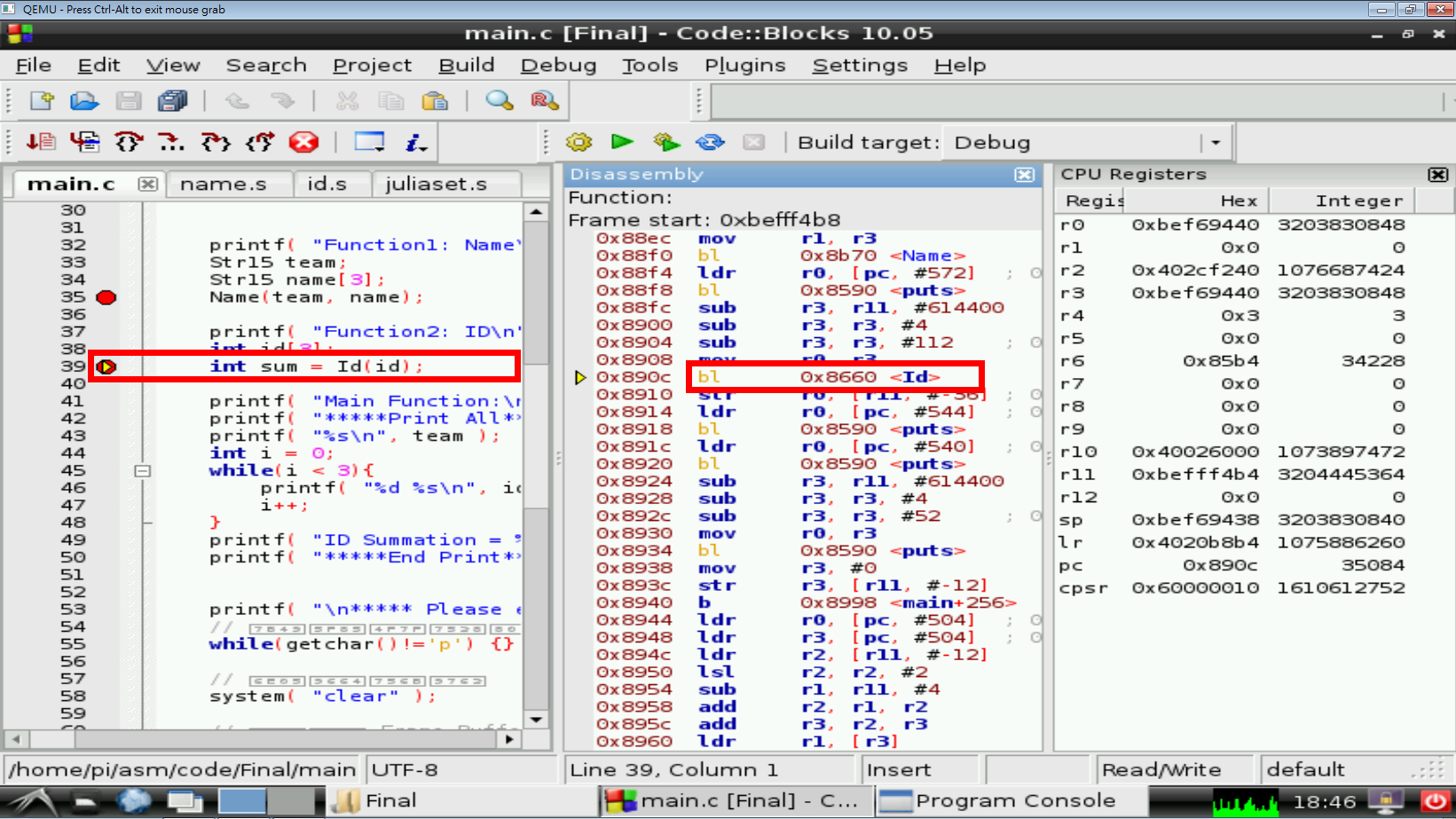
**name.s\_call**

****

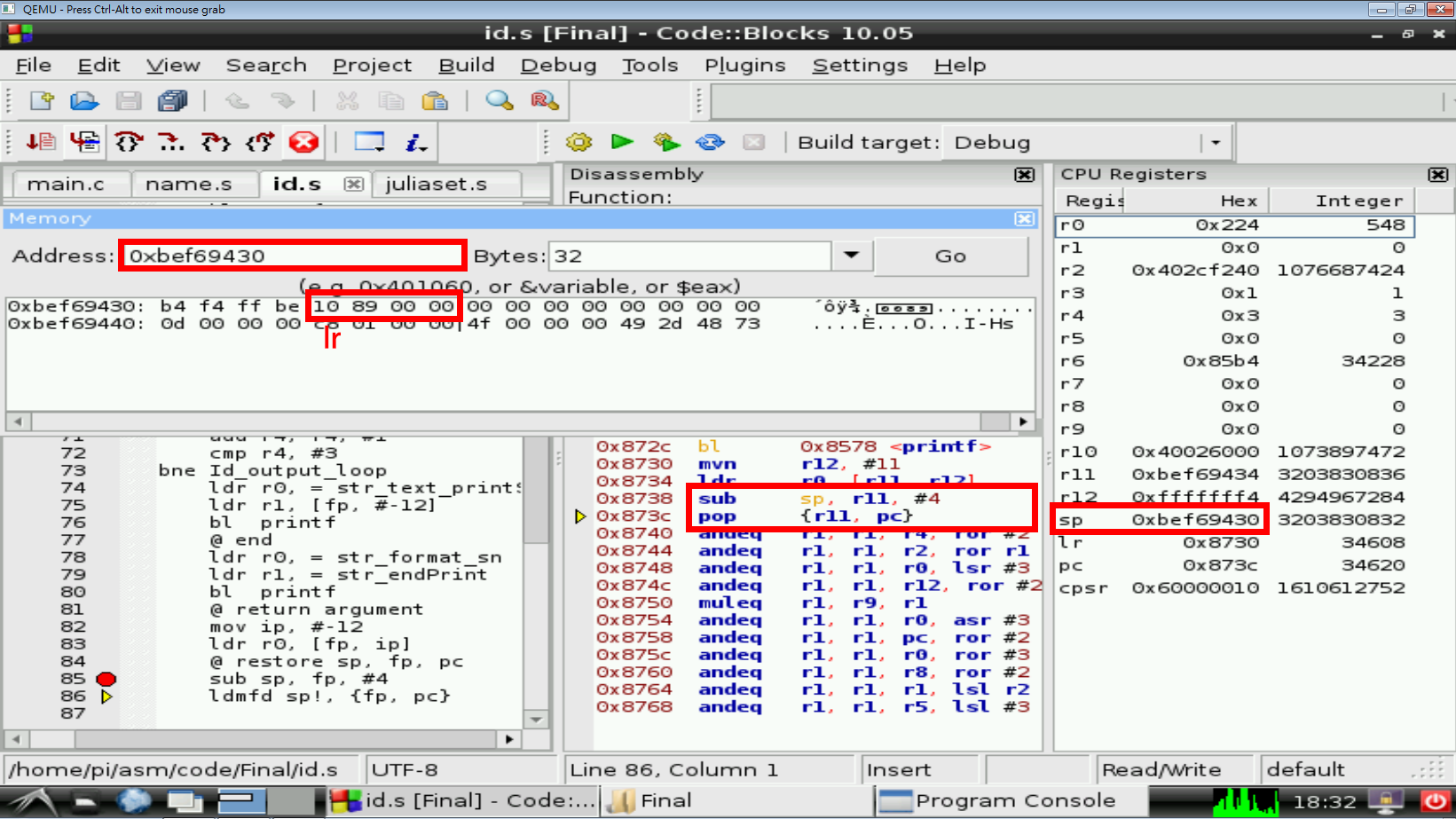
**name.s\_return**

****

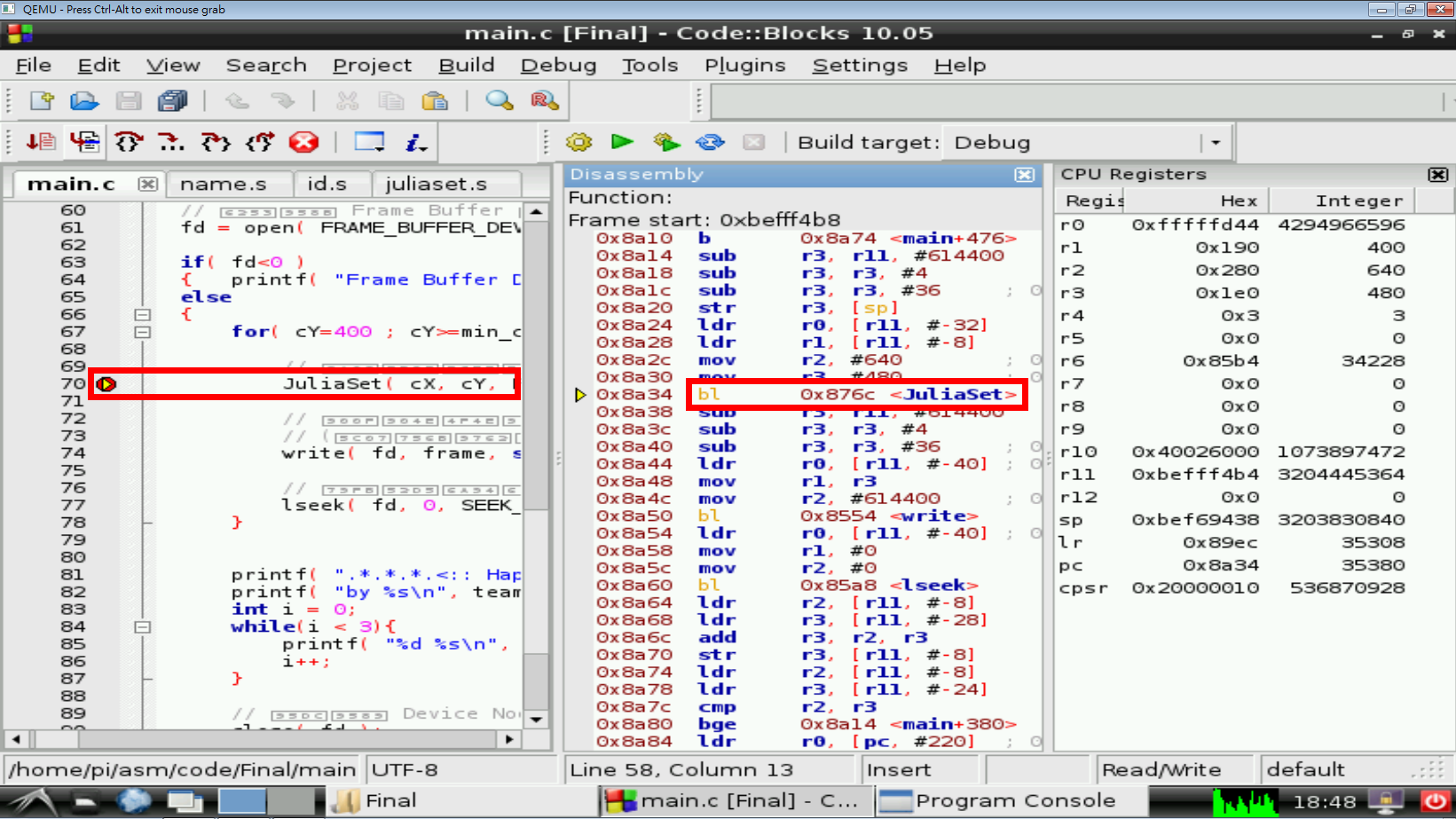
**id.s\_call**

****

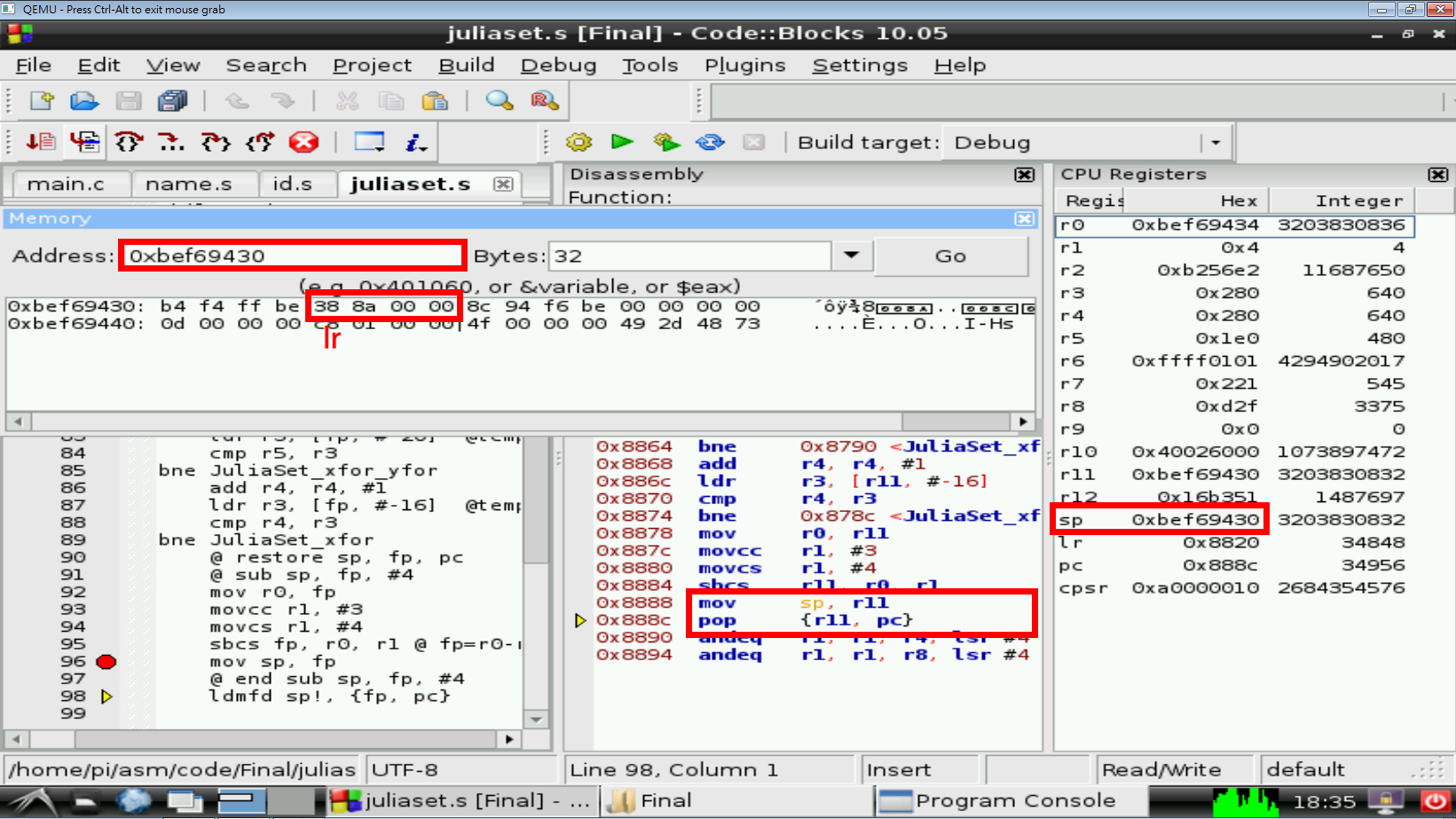
**id.s\_return**

****

**juliaset.s\_call**

****

**juliaset.s\_return**

****

(四)心得感想：

**雲和:**

這次在寫之前，有先將drawJuliaSet.c轉成.s檔，然後花了很多時間與精力閱讀並理解 drawJuliaSet在做哪些事情，並且了解fp跟sp的運作。所以在自己開始寫時，遇到的不多。

只是在寫完時，看著自己完成的內容不免感到有點驚訝，在寫C code時可能只是簡單的一行程式，用assembly表達卻要用一個大段落去實現，而且因為怕太慢，所以常用的變數被存在暫存器，導致寫程式時需要很小心，否則就有改到不該改的值的風險。

雖然drawJuliaSet花了我很多時間撰寫，但是寫完之後的debug卻相當迅速，主要遇到的問題在於有些數字無法直接給定，需要用label存數字在load進暫存器，以及忘記dereference，但是因為期中作業有完成，所以在遇到bug時不會花太多時間處理。

**逸翔:**

這次主要把整期中的程式重構，盡量把static variable改成local variable，理解了fp作為local起點的意義，把data連續放入sp，用迴圈去存取。也嘗試了各種方式的作法，傳值的部分用過傳參數、傳指標，發現各種不同的好處與壞處，當然最重要的還有回傳資料給C，接口必須要相同，不然會產生錯誤。

另外一個部份讓我印象深刻的就是簡化程式碼，且巧妙的運用結果的特性，利用一行add將兩道指令合併。也更加熟悉Debug模式，觀察Register的數值，大幅度的方便找出問題的所在，像是指標的指標，常常會少ldr一次。又或者是確認div只會會不會修改r0~r3，已完成參數的運用。

組語可以說是既簡單又複雜，簡單的程式碼，複雜的內容完全靠programmer自行組裝創造，可以說是想像力就是超能力。這次其中project讓我進步不少，希望我能理解更進階的指令或是special instruction，融入到程式內，讓組語的速度更快，更好閱讀。

(五)未來展望和修改方向:

1. 為了撰寫方便，導致註解數量不足，未來希望能增加註解，增加程式可讀性。

2. 因為是hand code assembly，正常情況下，程式執行速度要比c code快，但是在執行結果上看來，這隻程式沒辦法達到c code的執行速度，希望未來可以加速程式的執行。

3. 讓使用者自行輸入參數，-1.0< cX < 1.0 以及 0.0< cY < 1.0，使程式可以產生

不只一種圖形。

(六)組員分工與負責項目：

**許逸翔10627116：**

研究傳址方式，改寫期中內容與隊友的juliaset.s，使其速度加快一些，dump function的address與return address，共同撰寫說明文件。

**王雲和10627125：**

閱讀gcc產生之內容，理解drawjuliaset.s運作方式，寫出一版

drawjuliaset.s，共同撰寫說明文件。