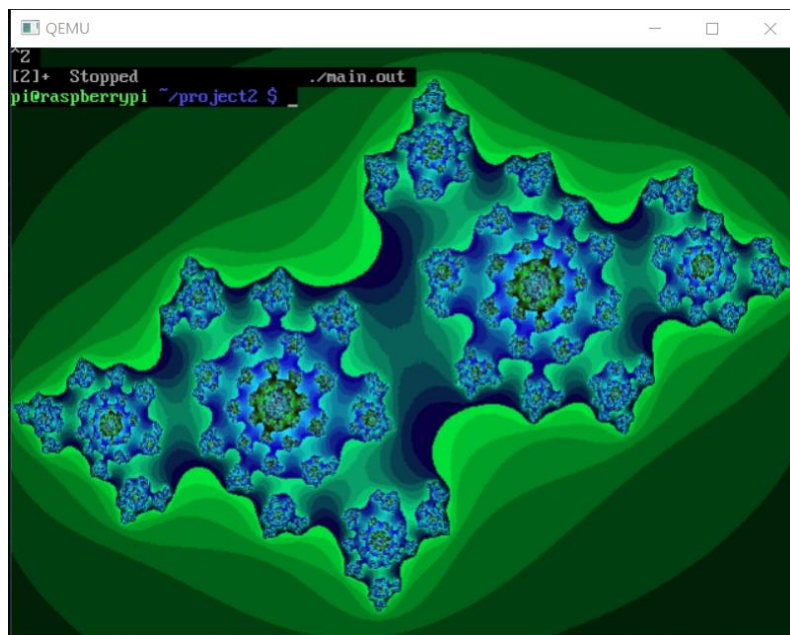


# Final Project

## Arm Assembly Language

106 學年度第 1 學期

老師：朱守禮 老師



資訊二甲

學生： 10527130 陳少洋

10527135 張智欽

10527140 初 元

## 一、 背景

Notepad++

RaspberryPi Simulator

## 二、 方法

程式說明：

沿用期中 Project 寫過的兩個函式 name 與 id，稍微修改一下，新增了一個 drawJuliaSet 函式，最後在整合到 main 裡。

在 name 函式中，印出已經事先配置好記憶體組別與組員名單。

在 id 函式中，先規劃 4 個記憶體位址作為之後輸入的組員與組別資料的暫存再以輸入的方式，輸入 3 個學號，記錄在前面準備好的其中 3 個記憶體位址中然後再將 3 個學號相加並紀錄於第 4 個記憶體位址，最後按下 p 鍵印出組員學號與學號總和。

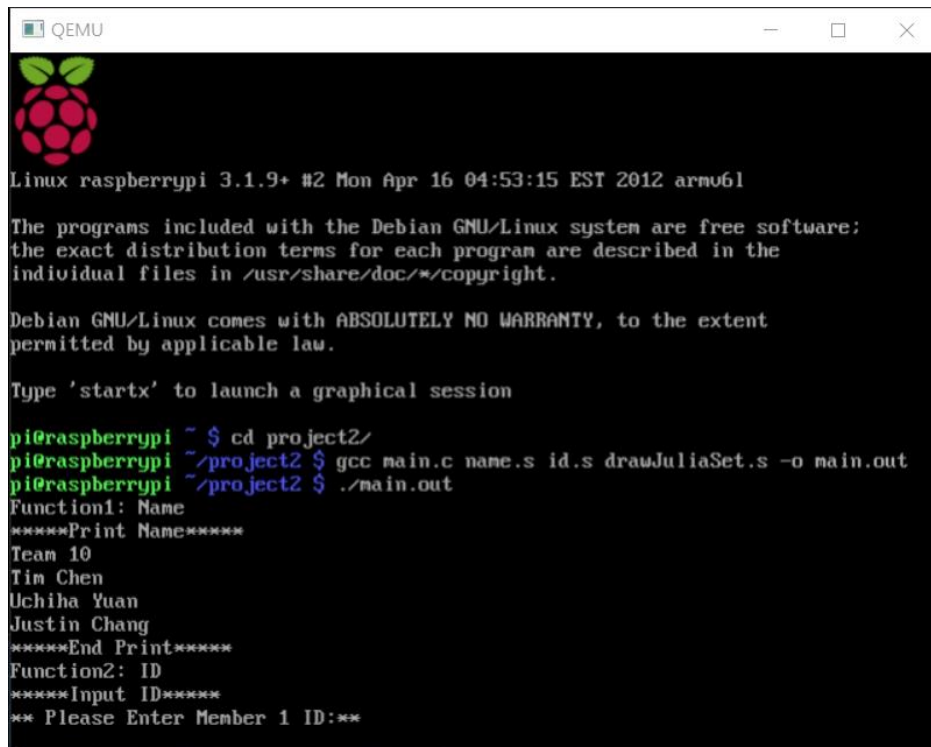
在 drawJuliaSet 函式中，依照老師給的 C code，改為 Assembly code。

在 main 函式中，由於是用 .c 檔，因此需要宣告前面寫好的三個函式，在最後整合輸出完整的資料。

設計重點：

了解如何配置記憶體與善用暫存器完成資料的輸入與輸出，並在 r0 ~ r3 暫存器都使用完時，使用 r4 ~ r11 暫存器，但要記得備份還原。嘗試使用數個 operand2 及非 branch 的 Conditional Execution 指令完成程式，輸入時用字串存避免讀入換行字元，以及在 main.c 檔裡如何宣告以及呼叫前面三個函式。

### 三、結果



```
Linux raspberrypi 3.1.9+ #2 Mon Apr 16 04:53:15 EST 2012 armv6l

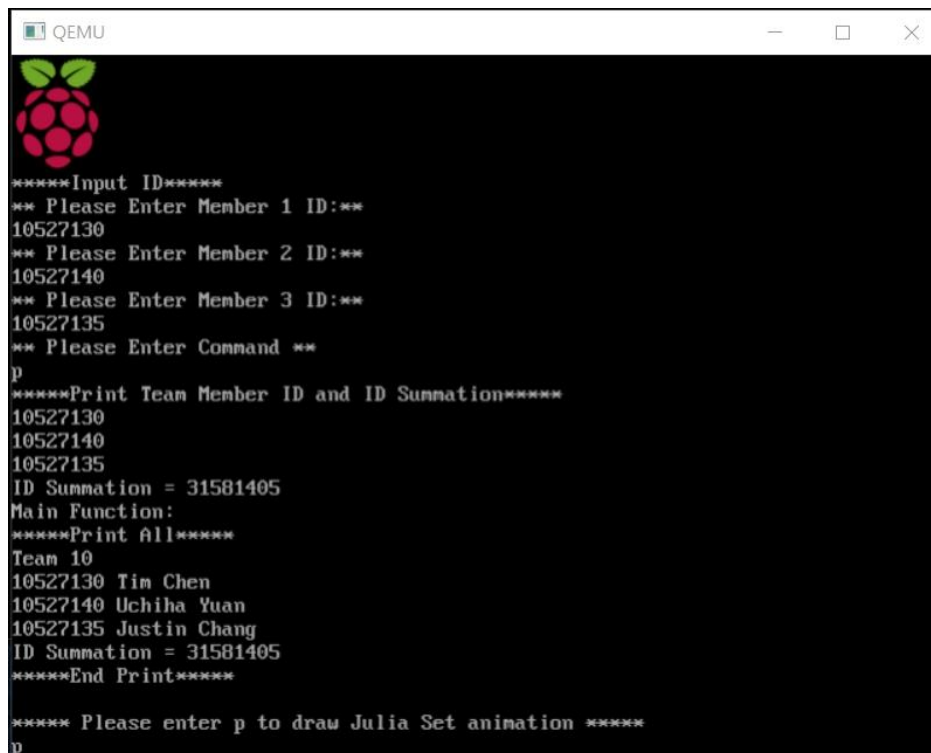
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Type 'startx' to launch a graphical session

pi@raspberrypi ~ $ cd project2/
pi@raspberrypi ~/project2 $ gcc main.c name.s id.s drawJuliaSet.s -o main.out
pi@raspberrypi ~/project2 $ ./main.out
Function1: Name
*****Print Name*****
Team 10
Tim Chen
Uchiha Yuan
Justin Chang
*****End Print*****
Function2: ID
*****Input ID*****
** Please Enter Member 1 ID:**
```

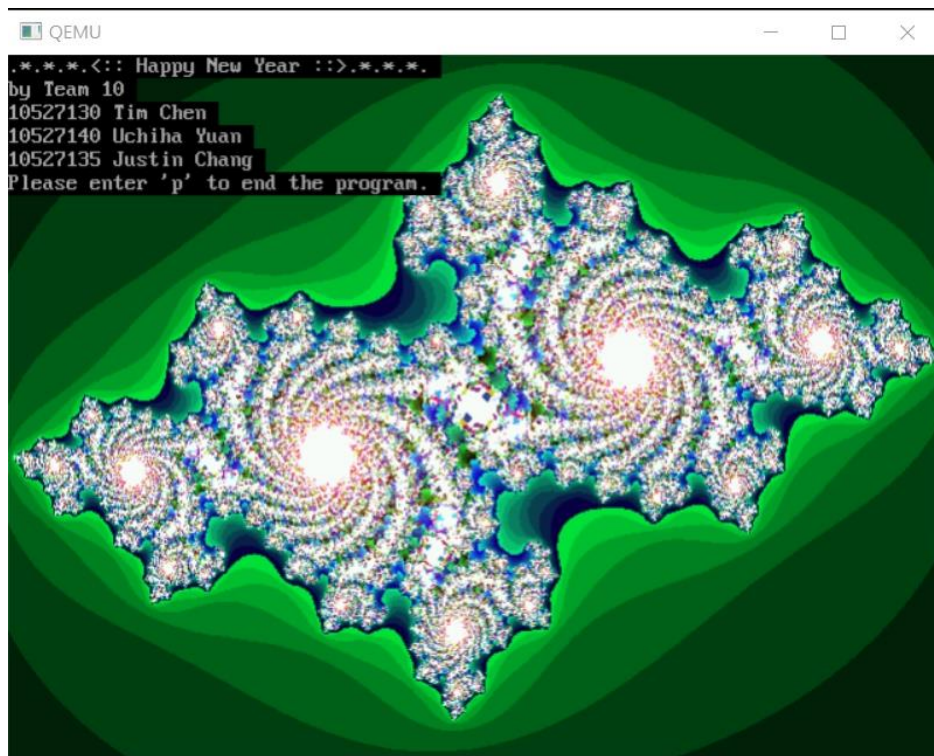
輸入：gcc main.c name.s id.s drawJuliaSet.s -o main.out 編譯組譯



```
*****Input ID*****
** Please Enter Member 1 ID:**
10527130
** Please Enter Member 2 ID:**
10527140
** Please Enter Member 3 ID:**
10527135
** Please Enter Command **
p
*****Print Team Member ID and ID Summation*****
10527130
10527140
10527135
ID Summation = 31581405
Main Function:
*****Print All*****
Team 10
10527130 Tim Chen
10527140 Uchiha Yuan
10527135 Justin Chang
ID Summation = 31581405
*****End Print*****

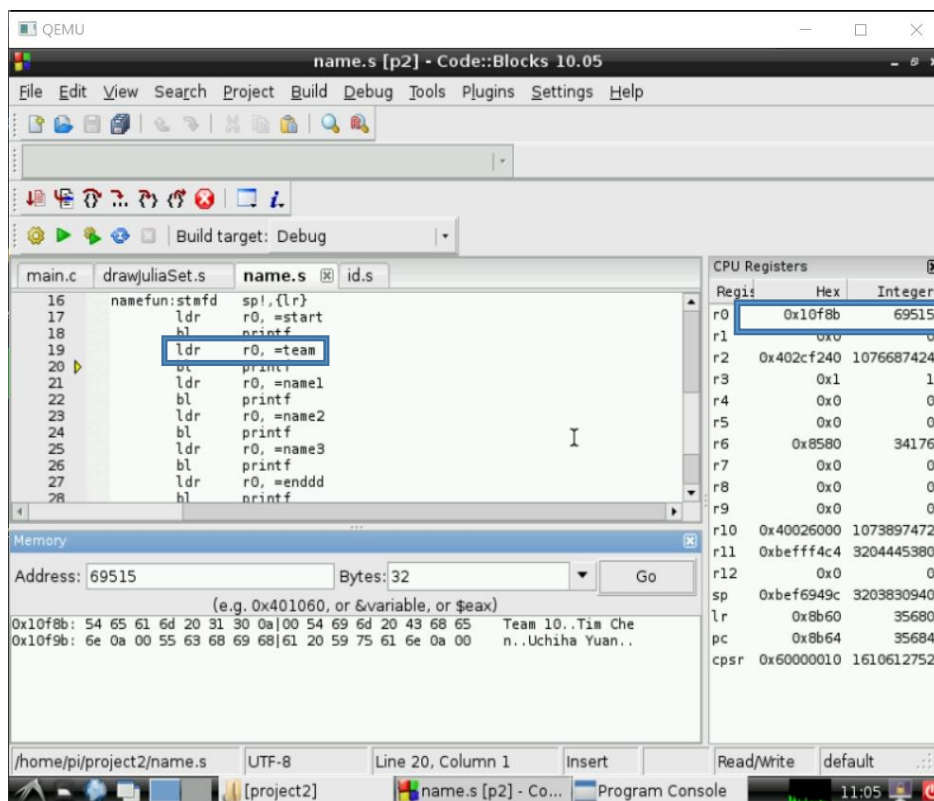
***** Please enter p to draw Julia Set animation *****
p
```

輸入：./main.out 執行檔案



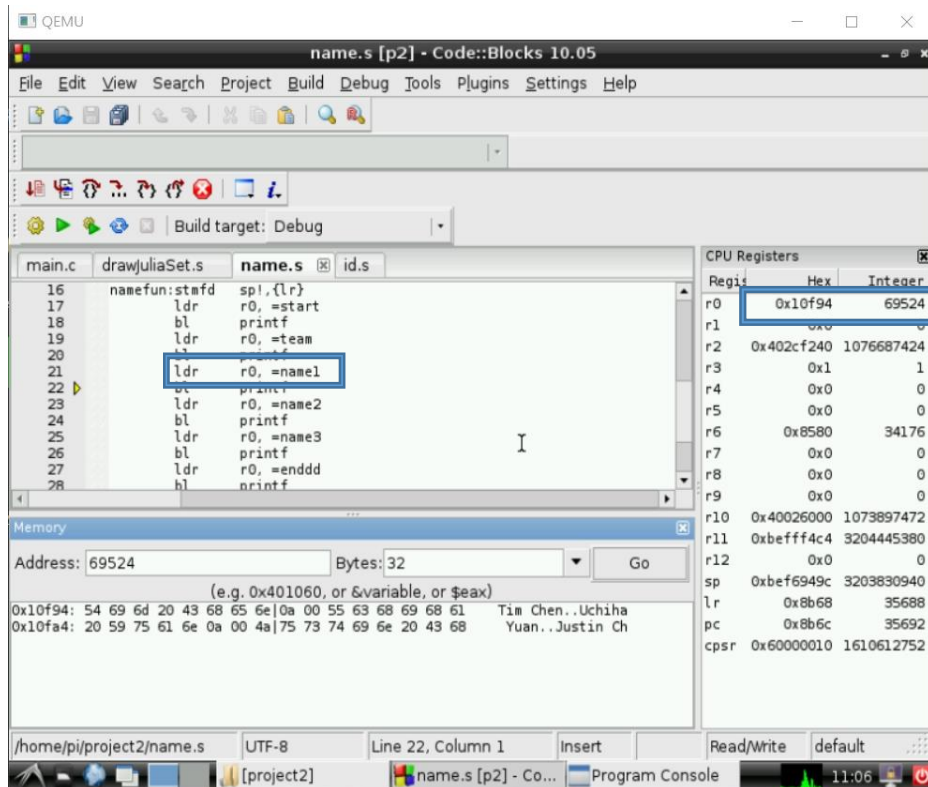
得到最後一幕結果。

## name.s



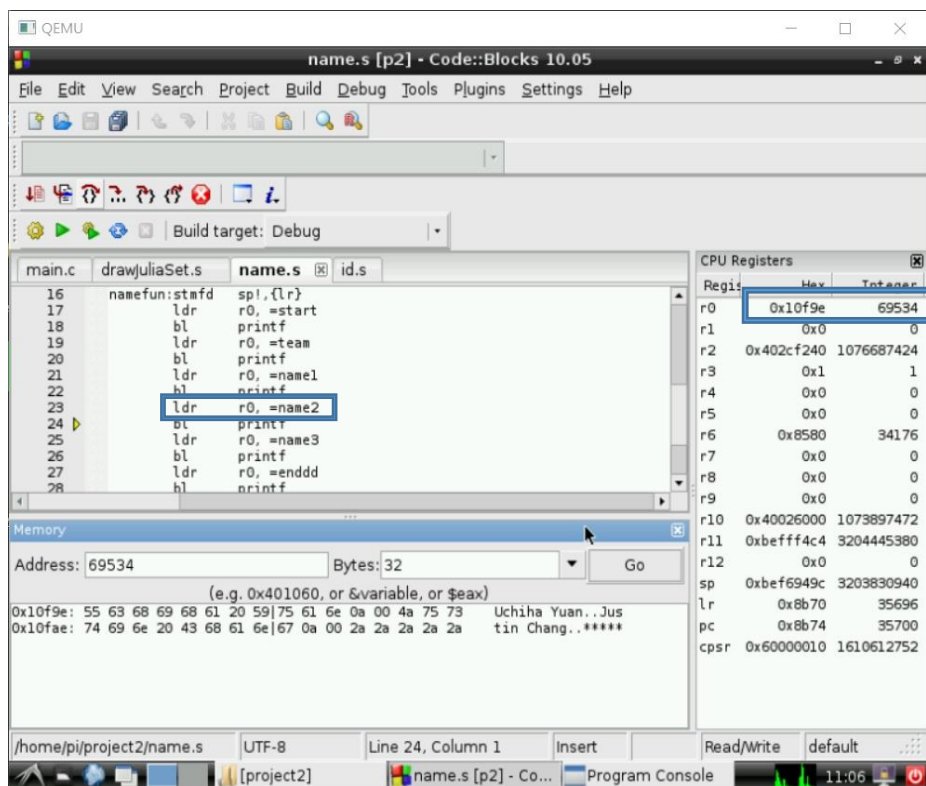
執行到第 19 行 team 的位址存入 r0

透過 CPU Reigsters 得知 Team 的位址在 0x10f8b



執行到第 20 行 name1 的位址存入 r0

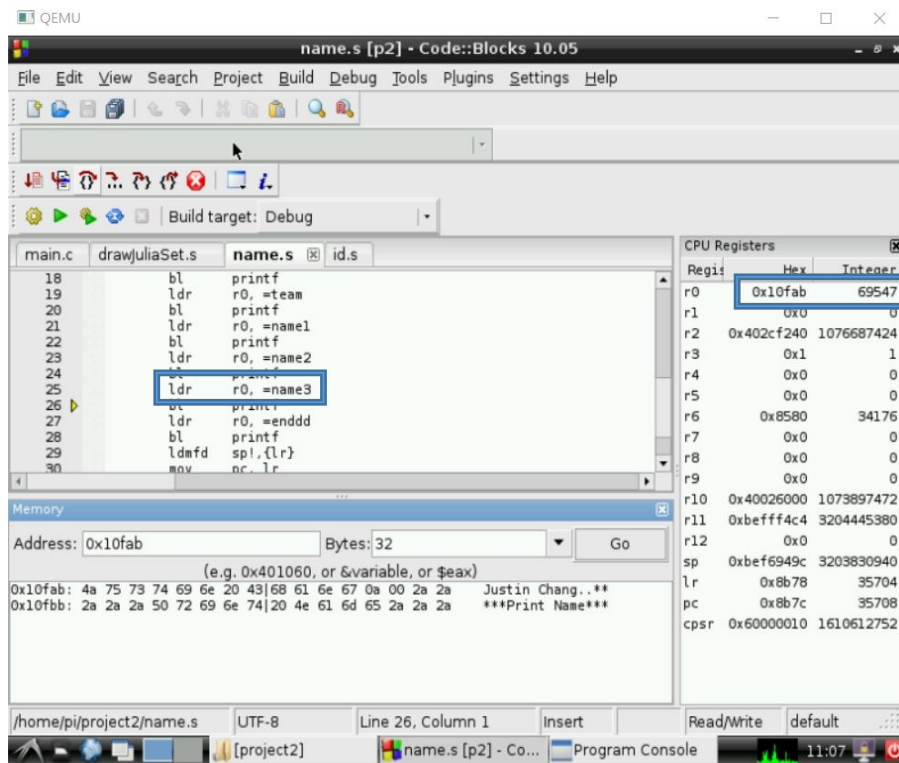
透過 CPU Reigsters 得知 Name1 的位址在 0x10f94



執行到第 23 行 name2 的位址存入 r0

透過 CPU Reigsters 得知 Name2 的位址在 0x10f9e

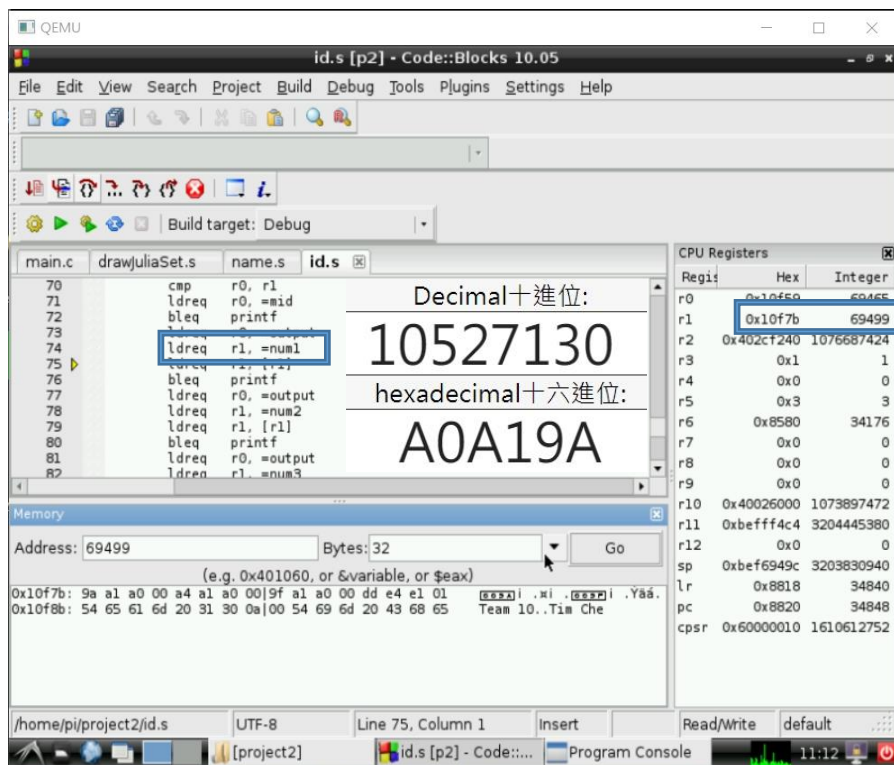




執行到第 25 行 name3 的位址存入 r0

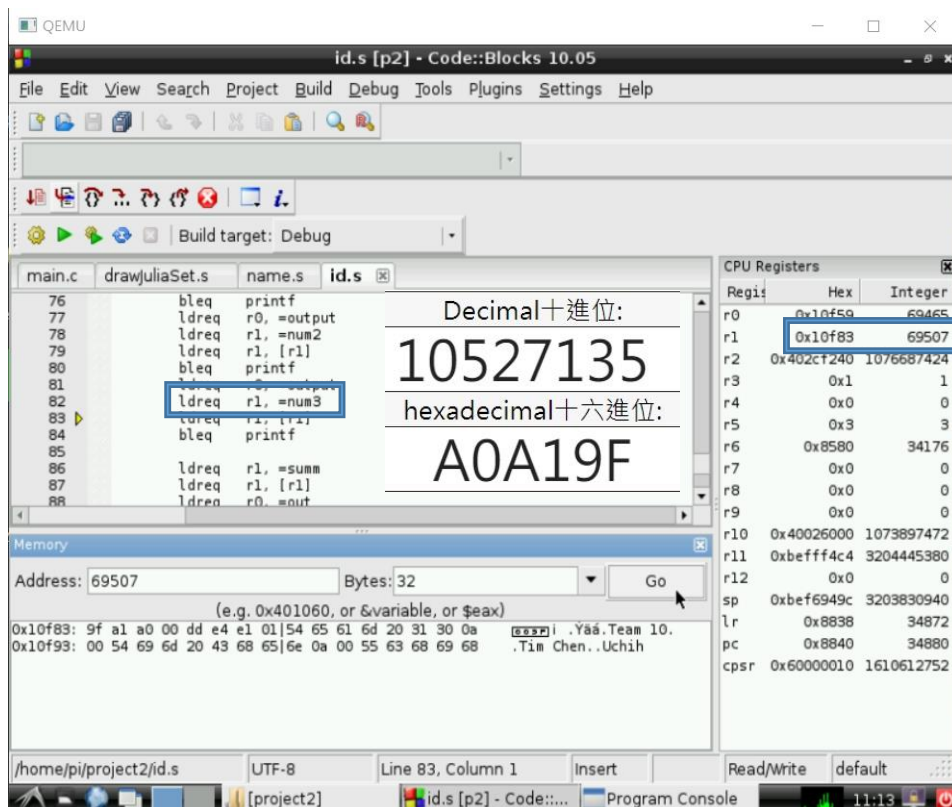
透過 CPU Reigsters 得知 Name3 的位址在 0x10fab

## id.s

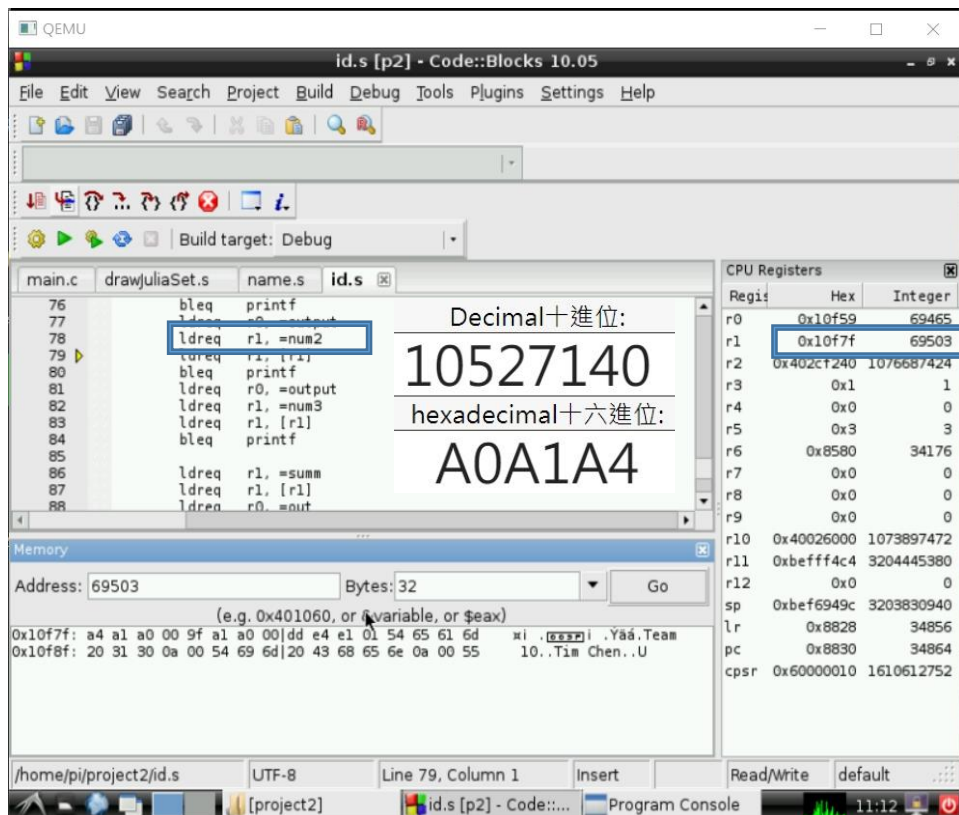


執行到第 74 行 num1 的位址存入 r1，透過 CPU Reigsters 得知

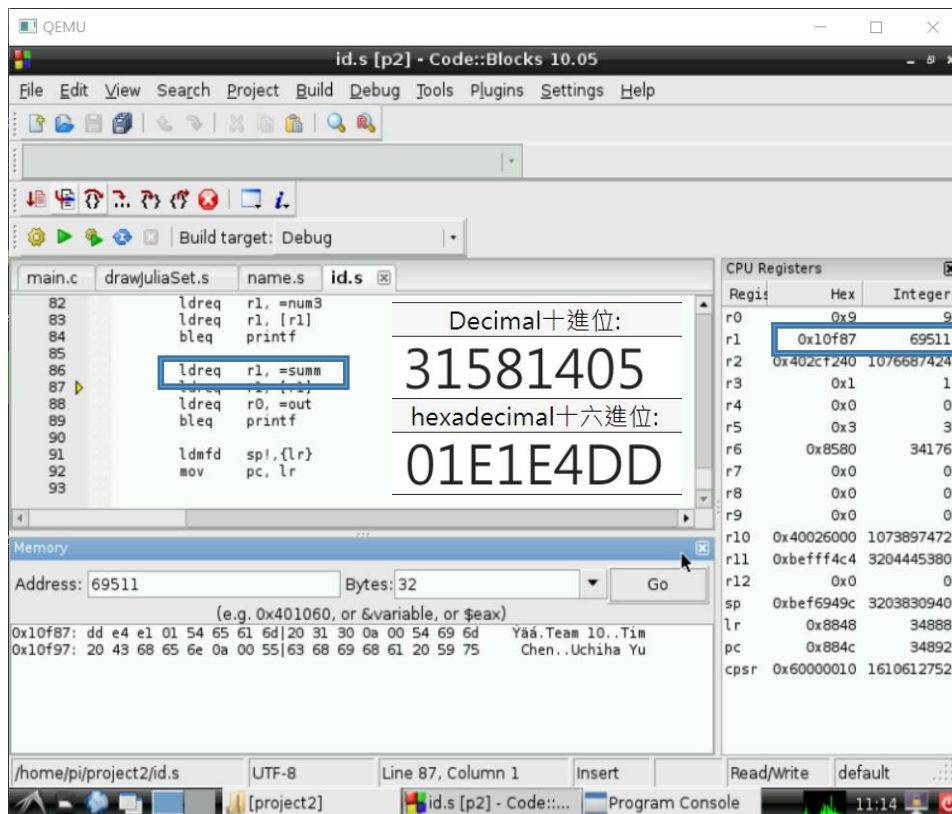
第一個學號 num1 的開始位址在 0x10f7b，而結束位址在 0x10f7b + 5。



執行到第 82 行 num2 的位址存入 r1，透過 CPU Reigsters 得知  
第二個學號 num2 的開始位址在 0x10f83，而結束位址在 0x10f83 + 5。



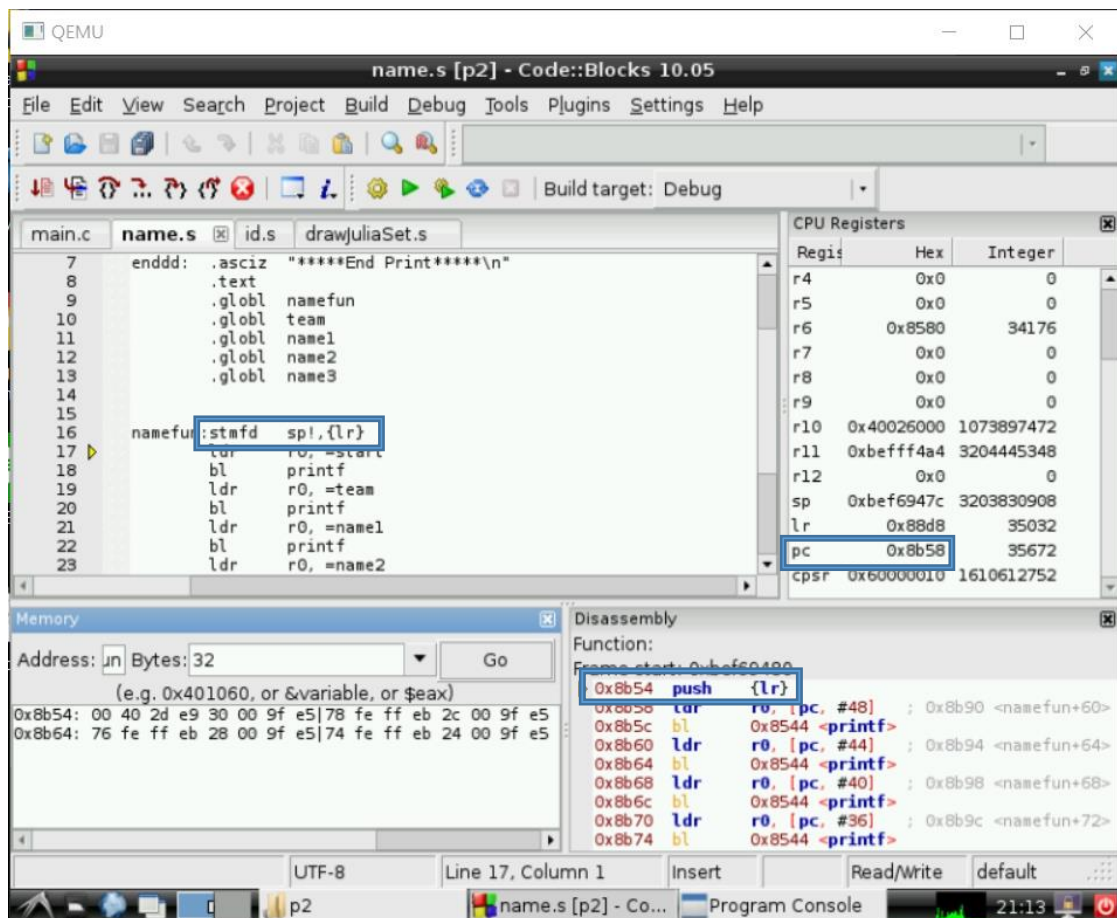
執行到第 78 行 num3 的位址存入 r1，透過 CPU Reigsters 得知  
第三個學號 num3 的開始位址在 0x10f7f，而結束位址在 0x10f7f + 5。



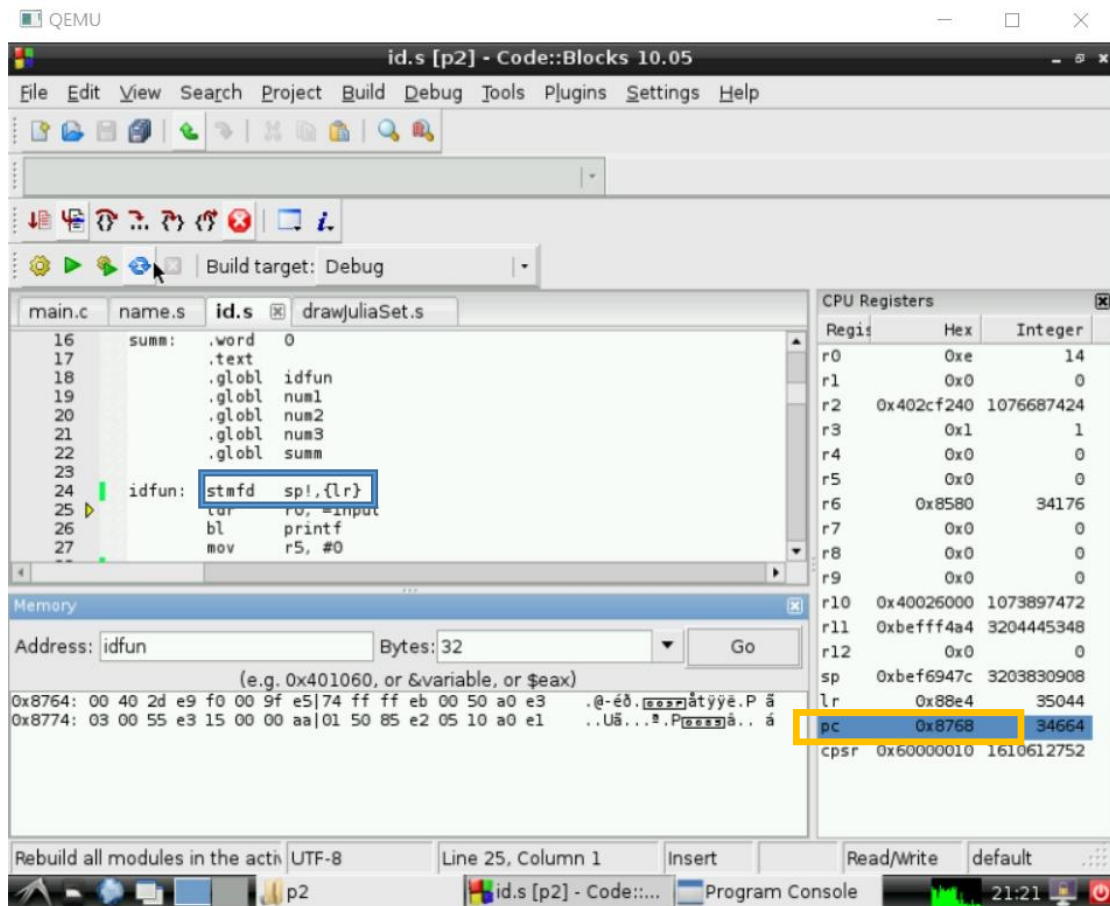
執行到第 86 行 sum 的位址存入 r1，透過 CPU Reigsters 得知 Sum 的開始位址在 0x10f87，而結束位址在 0x10f87 + 7。



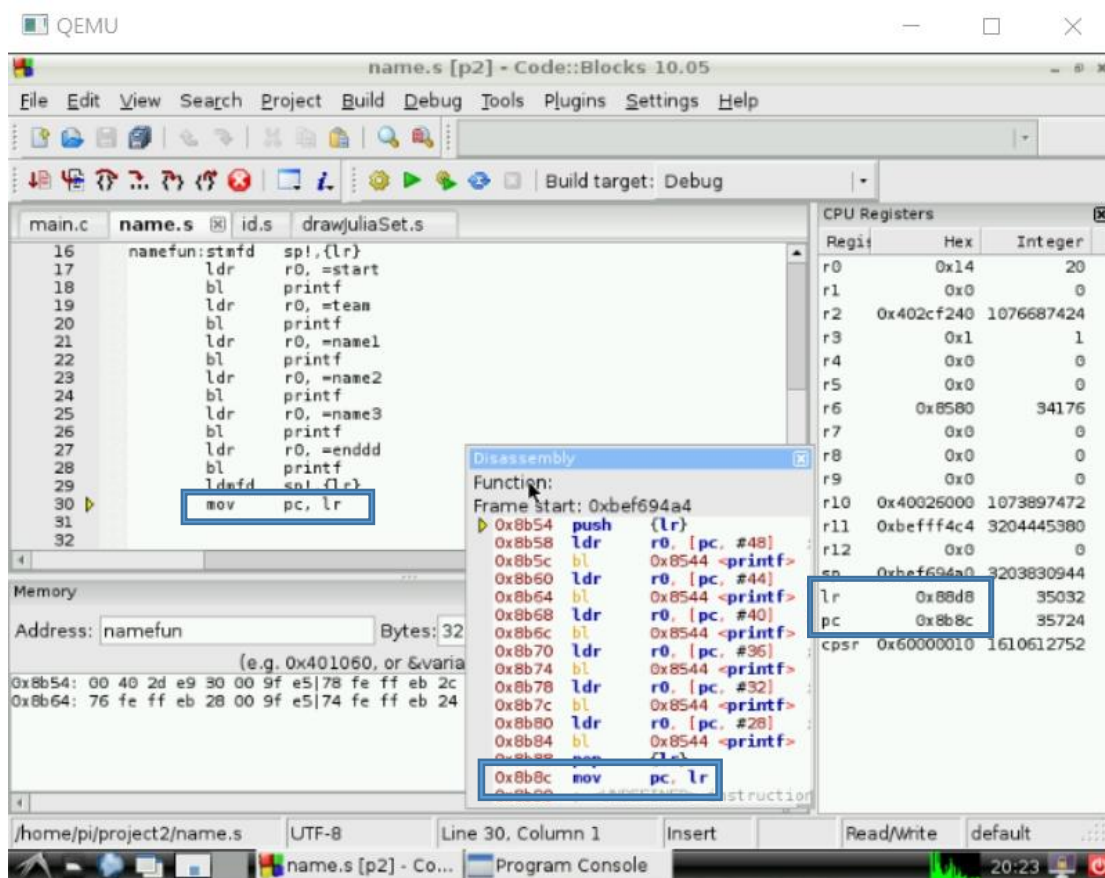
## main.s



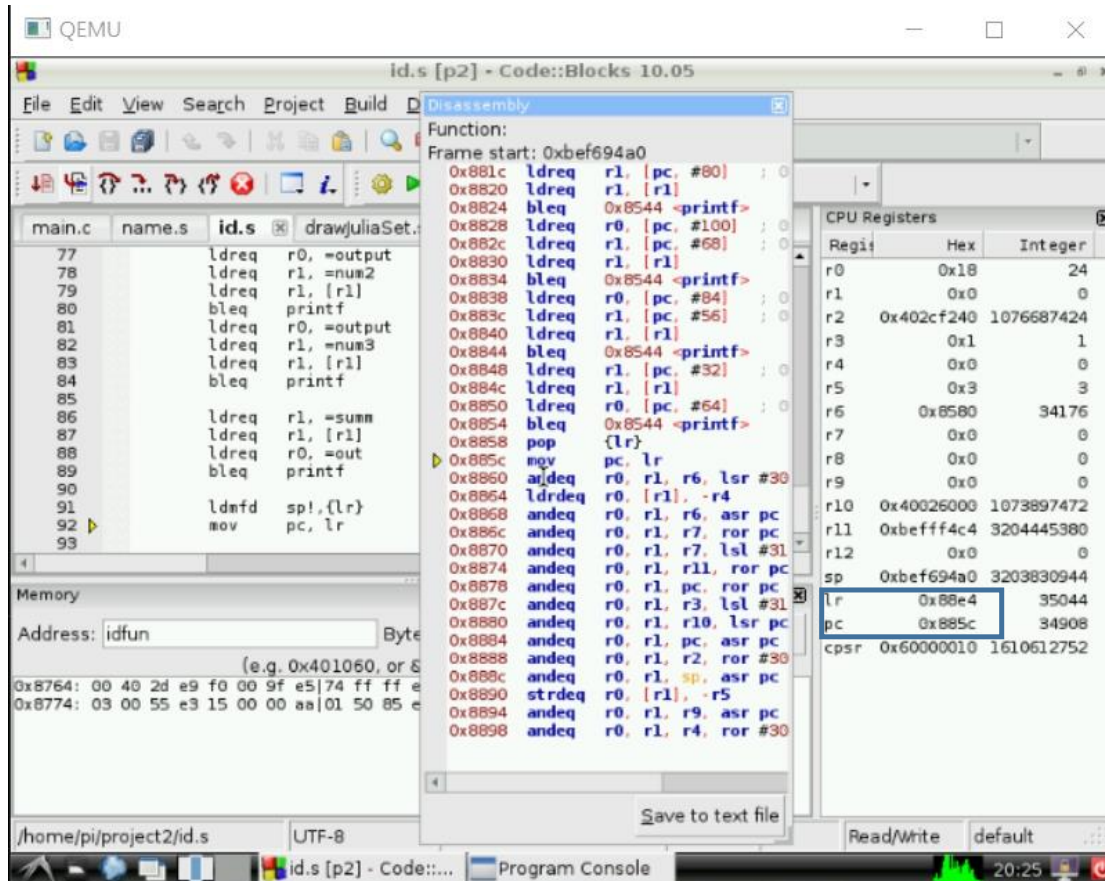
執行到第 16 行後，下一道指令 pc 指向 0x8b58，可得知 namefunction 的起始位址在 0x8b54，disassembly 視窗中第一道指令(stmfd sp!, {lr})的位址為 0x8b54，就是 namefunction 的起始位址。



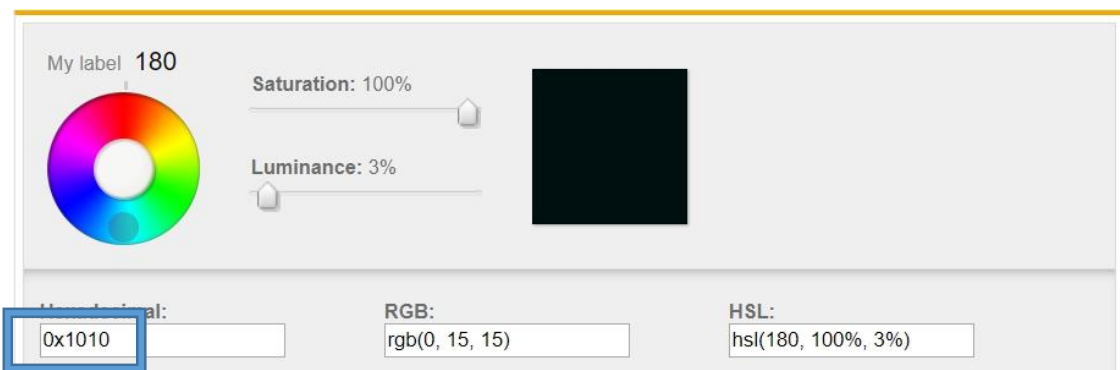
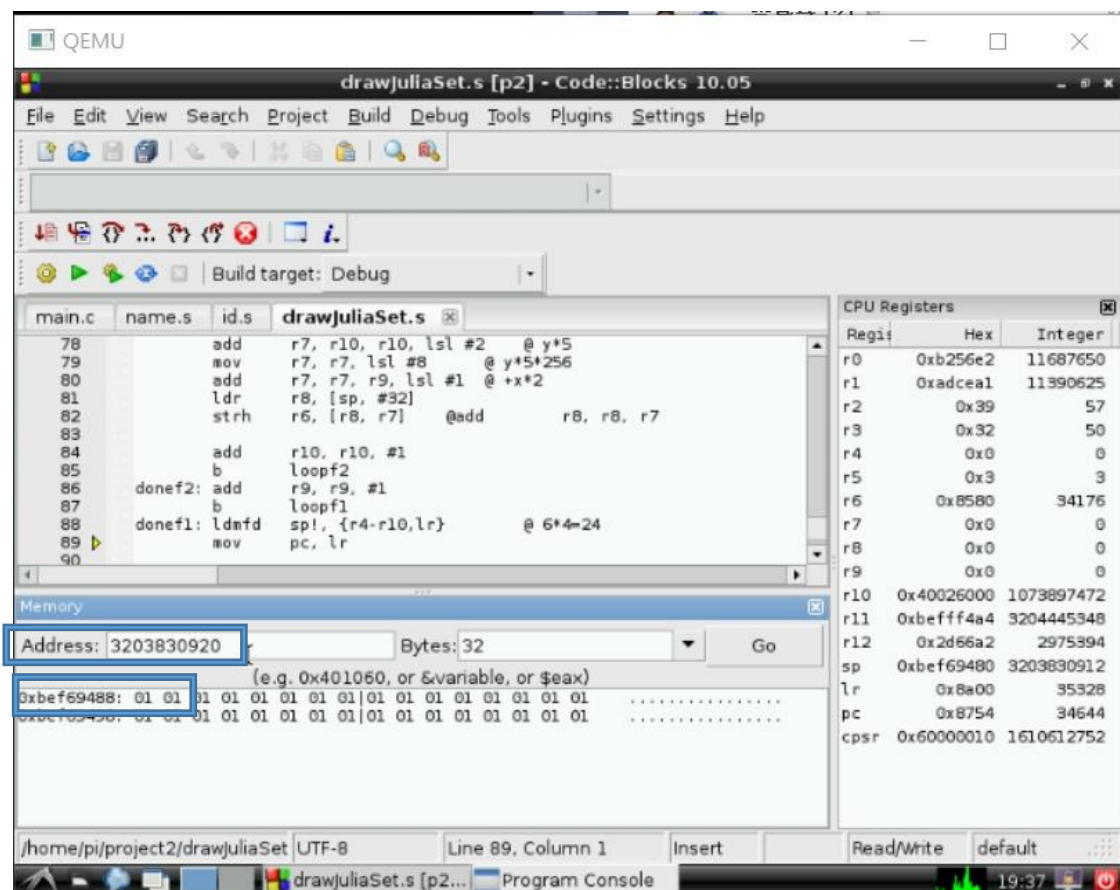
執行到第 24 行後，下一道指令 pc 指向 0x8768，可得知 idfunction 的起始位址在 0x8764。



執行到第 30 行後，下一道指令 pc 指向 0x8b8c，disassembly 視窗下面 mov 指令的位址也是 0x8b8c，可得知 namefunction 的返回位址就是 0x8b8c 本人。此時 lr 指向 0x88d8，return 回主函式後，程式從此位址開始執行。



執行到 92 行，下一道指令 pc 指向 0x885c，disassembly 視窗中 mov 指令的位址也是 0x885c，可得知 idfunction 的返回位址就是 0x885c 本人。  
此時 lr 指向 0x88e4，return 回主函式後，程式從此位址開始執行。



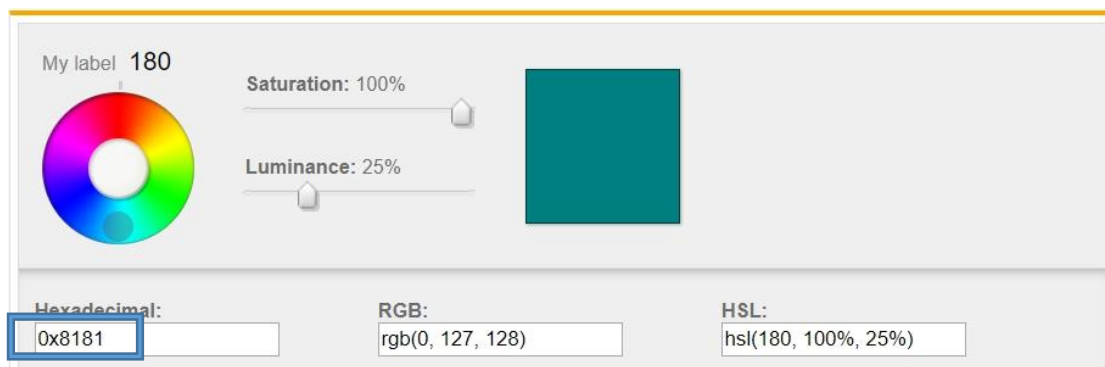
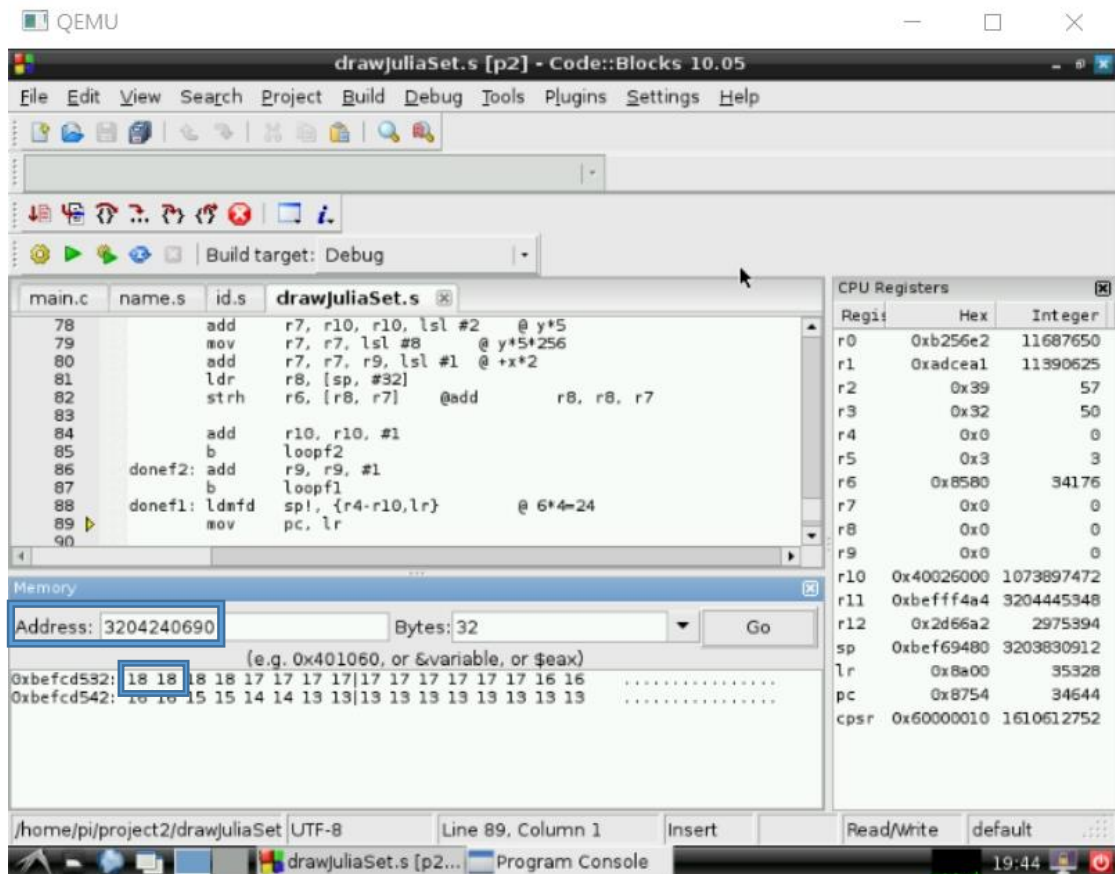
Frame[0][0] 的記憶體位址為 3203830920，照公式推算的話就是

$3203830920 + 0 * 2 * 320 + 2 * 0 = 3203830920$  得證。

而使用 memory dump 查到此顏色的十六進制碼為 0x1010

( 公式： $640 * 2y + 2x + \text{frame 起始位址} = \text{frame}[y][x]$  的位址 )



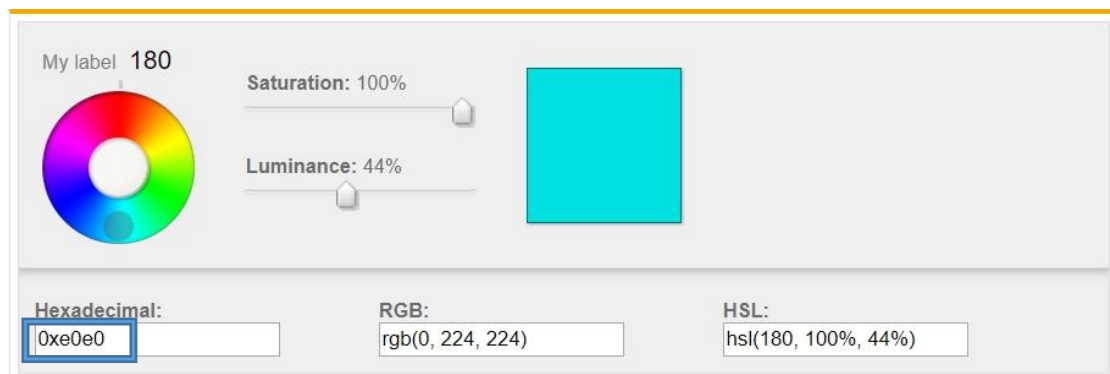
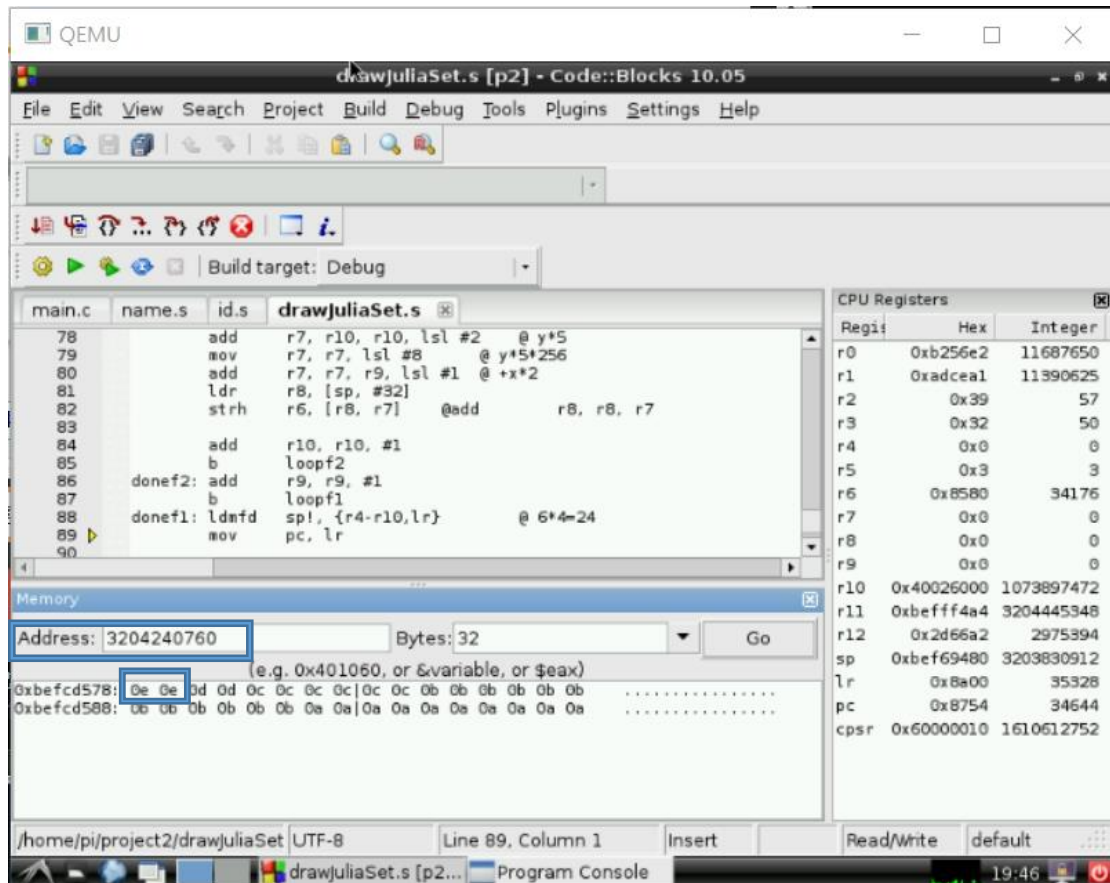


Frame[320][85] 的記憶體位址為 3204240690，照公式推算的話就是

$3203830920 + 640 * 2 * 320 + 2 * 85 = 3204240690$  得證。

而使用 memory dump 查到此顏色的十六進制碼為 0x8181

( 公式： $640 * 2y + 2x + \text{frame 起始位址} = \text{frame}[y][x]$  的位址 )

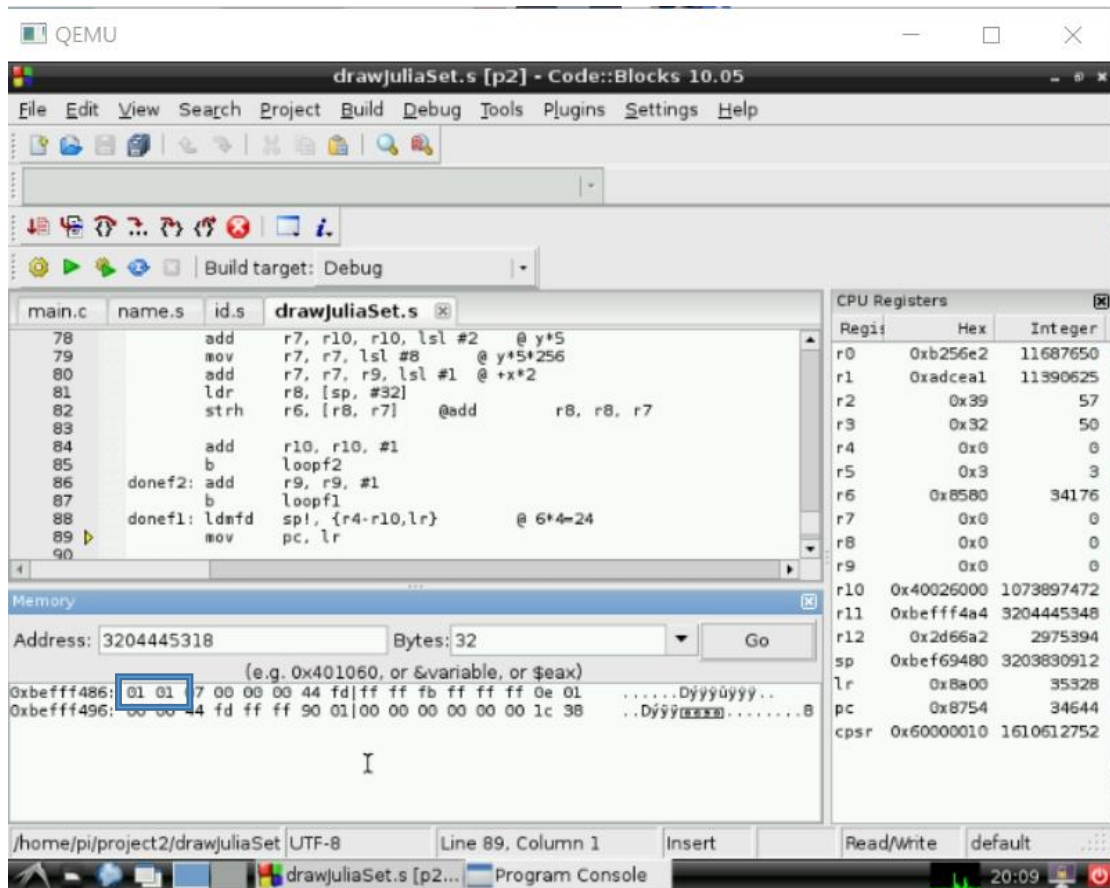


Frame[320][120] 的記憶體位址為 3204240760，照公式推算的話就是

$3203830920 + 640 * 2 * 320 + 2 * 120 = 3204240760$  得證。

而使用 memory dump 查到此顏色的十六進制碼為 0xe0e0

( 公式： $640 * 2y + 2x + \text{frame 起始位址} = \text{frame}[y][x]$  的位址 )

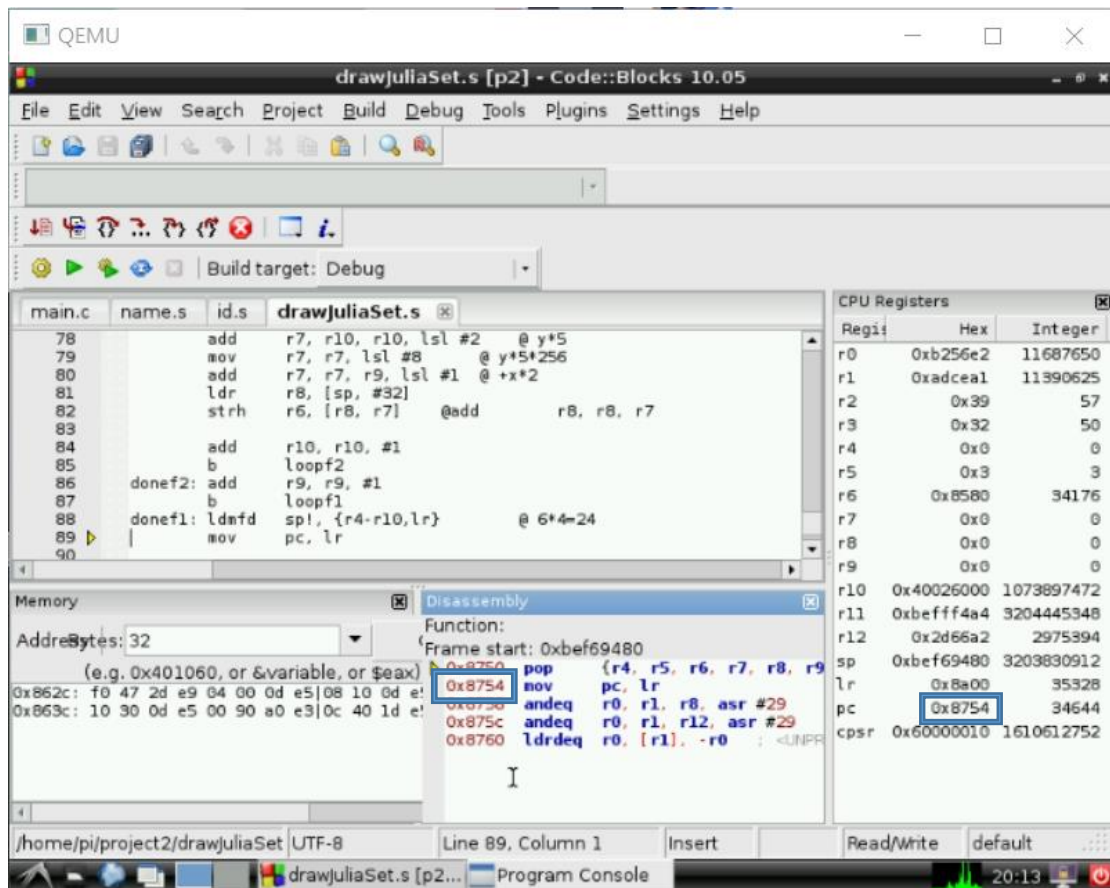


Frame[479][639] 是陣列的最後一項記憶體位址在 3204445318，

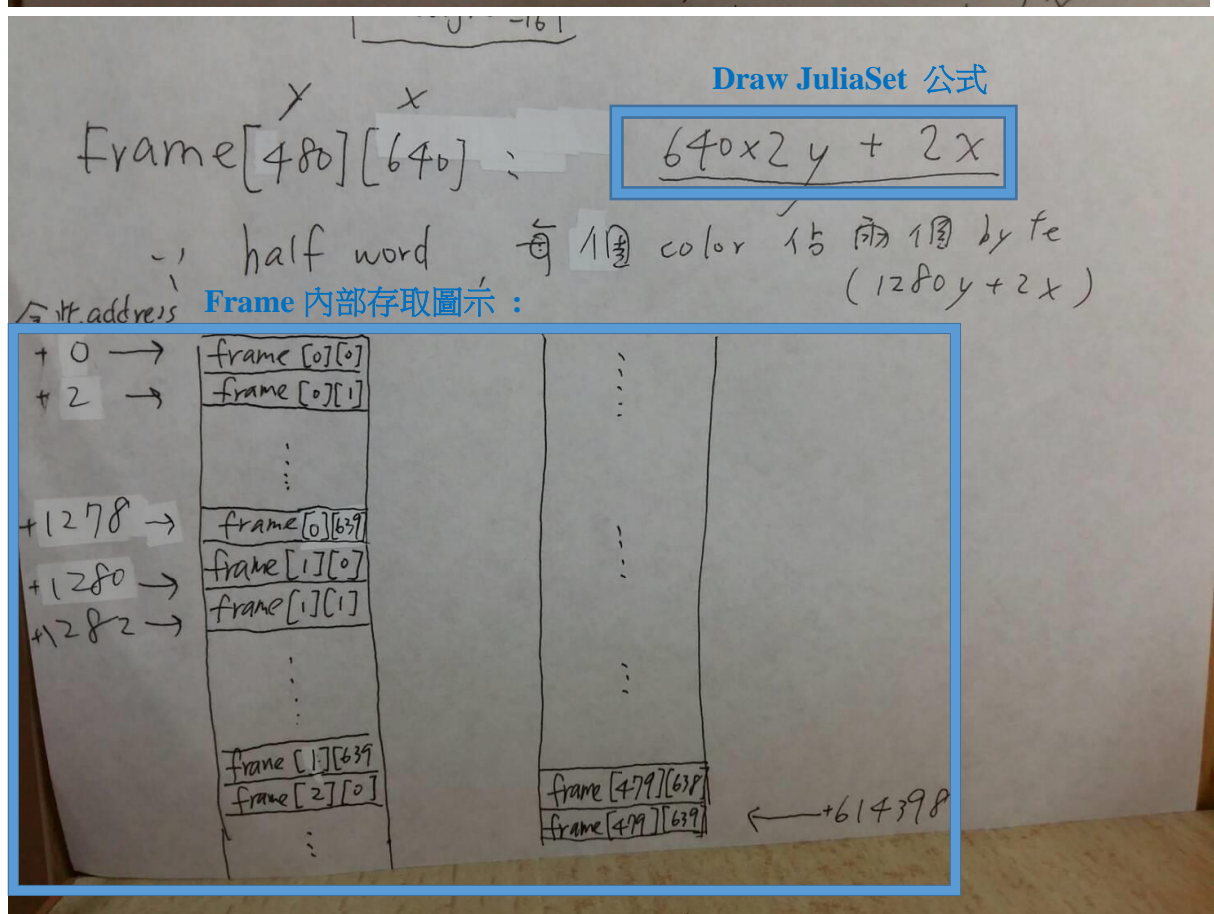
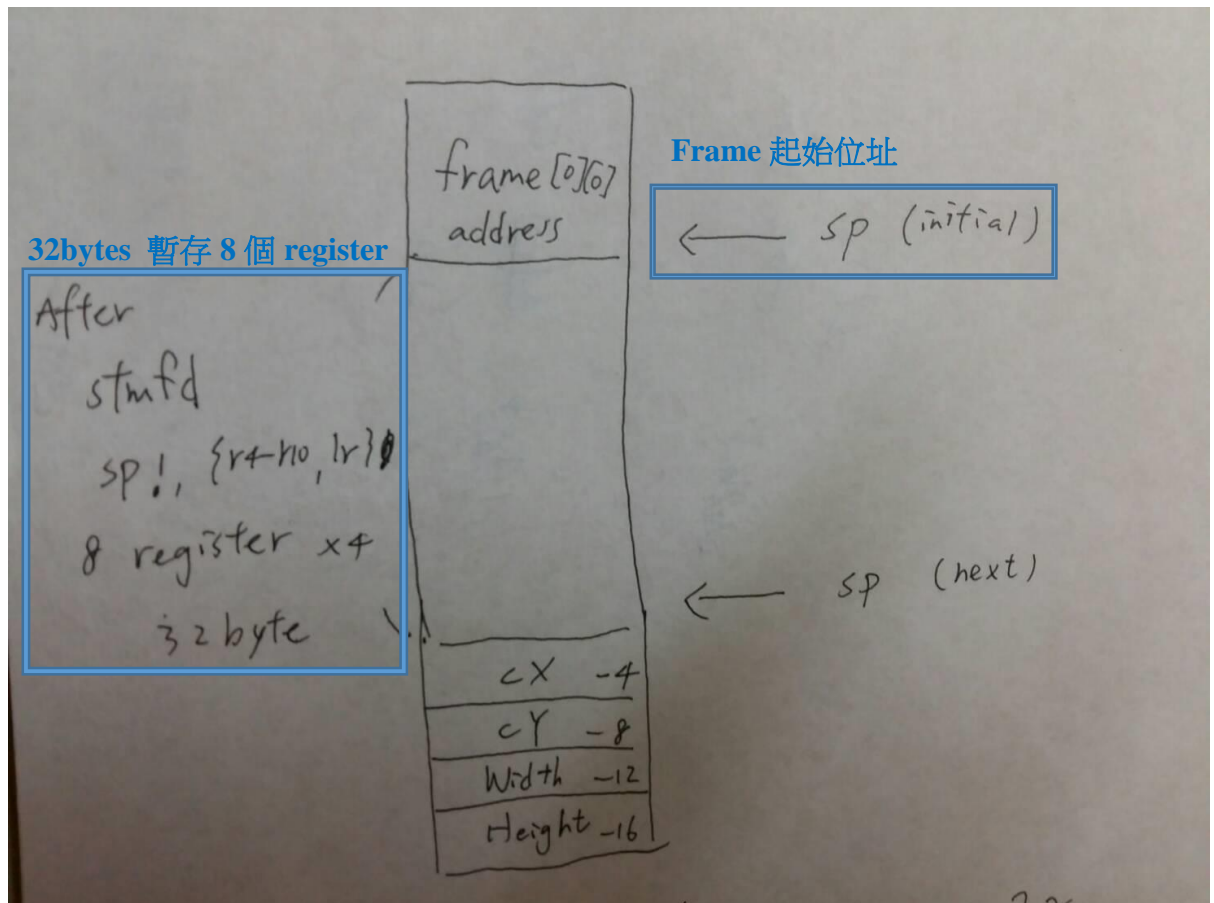
套入公式  $640 * 2 * 479 + 2 * 639 + 3203830920 = 3204445318$  得證

而使用 memory dump 查到最後的顏色十六進制碼為 0x1010 跟 Frame[0][0]同色

( 公式： $640 * 2y + 2x + \text{frame 起始位址} = \text{frame}[y][x]$  的位址 )



執行到 89 行，下一道指令 pc 指向 0x8754，disassembly 視窗中 mov 指令的地址也是 0x8754，可得知 drawJuliaSet function 的返回位址就是 0x8754 本人。此時 lr 指向 0x8a00，return 回主函式後，程式從此位址開始執行。





#### 四、討論

與上次 Midterm Project 一樣，依照講義給的指令，這次寫起來就比較熟練，另外，在寫 drawJuliaSet 時，因為樹梅派是使用 Arm V6，而 Arm V6 沒有除法指令，除法指令是在 Arm V7 時才有，然後我們在看 compiler 時，發現除法是使用外部函式庫裡的 function 叫做\_\_aeabi\_idiv，所以我們就拿來使用，我們在操作 r0 / r1 時，我們將結果存在 r0，而迴圈條件設定在 r2、r3，可是在 branch and link 時，它會將 r0 ~ r4 的值洗掉，導致我們會遇到無窮迴圈，因此我們將迴圈條件設定在 r9、r10，還有在寫完時遇到了 bug，發現竟然是不小心將 r10 寫成 r0，差點就衝動把整個刪掉重寫，這次錯誤也提醒我們要多注意細節。

#### 結論：

我們的結論是經過 Midterm Project 的練習，這次在寫的過程中，一些基本的指令就比較少用錯，而這次的 drawJuliaSet 使用到比較多暫存器，但可直接使用的暫存器只有 r0 ~ r3，因次我們使用 r4 ~ r11 來存放，不過使用時要記得備份還原，另外，drawJuliaSet 是我們之前沒接觸過的東西，雖然有老師提供的 C code，不過並不是很了解，所以有去研究了一下，過程中，我們發現了 i 是代表 256 種顏色，根據寬度跟長度就可以計算那個位置的顏色，也學到了二維陣列的儲存方式，也上網學到了如何在 main.c 檔裡使用寫好的三個.s 檔。

#### 心得：

終於完成這次的 final project，這次在操作暫存器、指令等功能，都比較得心應手，在上課時看到老師操作 drawJuliaSet 時，用短短幾行的 C code，竟然就能做出這麼酷炫的圖，覺得非常新奇，但在寫的時候，由於第一次接觸，比較陌生一些，好險有老師提供的 C code，在跟同學討論，上網了解一下實際運作，寫起來就熟練許多，也藉由這次的練習，更熟悉了 Arm 暫存器的使用方

式，像老師上課提到的 r4 ~ r11 要使用必須先備份還原。最後，感謝老師這學期的教導，讓我們學到了豐富的組語以及嵌入式系統知識，希望下學期的計算機組織也能順利通關。

## 六、未來展望

經過上次的 midterm project，又經歷了這次的 final project，感覺已經漸漸習慣了組語這門科目，想到剛開學的懵懂無知，對這門科目充滿未知的恐懼，到現在已經寫完了兩支程式。很開心在這門科目學到了很多，像是如何將 C code 轉換成 Assembly code 以及了解電腦內部實際處理指令的運作方面。剛開始學習時，聽學長姐說過，組語是跟資訊安全方面有關，一開始還不是很懂，現在終於稍微了解到兩者之間的相關性，希望未來的學習路程上，能夠繼續學習組語相關的深入知識，也希望未來組語能成為我們的一項必備技能。

### 各組員分工方式與負責項目：

程式碼、code block 操作：	陳少洋	10527130
報告結果分析、協助 debug：	初元	10527140
報告結果與討論、協助 debug：	張智欽	10527135