112-1 組合語言與嵌入式系統 Final Project 報告

112 學年度第 1 學期

老師：朱守禮 老師
學生：
電資三 11020107 蘇伯勳
電資三 11020140 葉柏榆
電資二 11120115 林佩玟

一、 背景

請參考 i-Learning 上之 Project 範例程式。其中包含四個範例程式檔案：name.c, id.c, main.c, drawJuliaSet.c。四個檔案一起編譯後執行，可在視窗上，看到 Julia Set 繪製的動畫。

請參考 Midterm Project 的說明，使用 Midterm Project 所開發之 NAME 與 ID 兩個組合語言函數，列印組別、組員名字、與學號。同時以 ARM 組合語言重新設計 drawJuliaSet 函數。並修改 main.c。除了 main.c 以外，所有程式均需以 ARM Assembly 完成。分工：葉柏榆->code，其餘->報告。

二、 方法

共有四支程式：“name.s” “id.s” “ main.c” “drawJuliaSet.s”

程式需符合「Project 基本要求」：

1. 需使用 Midterm Project 的 2 個函數：NAME, ID，並分別存放於 name.s 與 id.s 檔案內。
2. 需以 ARM 組合語言重新撰寫計算 Julia Set 的函數：drawJuliaSet，並存放於 drawJuliaSet.s 檔案裡。
3. drawJuliaSet 函數之 ARM 組合語言程式，需滿足以下三個項目：
   [1] 需使用 Data Processing 指令中 13 種 Operand2 格式的當中 3 種以上。
   [2] 需包含 3 道以上的非 Branch 指令的 Conditional Execution（不包括 AL 或”不指定”條件）。
   [3] 在 drawJuliaSet.s 程式的第 10 道實體指令須為一道一定要執行的指令： **orr sp, lr, r1**

以下分別針對三支程式說明其功能、要求與設計方法：

1. name.s

(1)功能：印出組別與組員名單。

(2)要求：組別與組員必須分行印出。若組員不足三位，重複填入已有的組員姓名，直到用完三個 記憶體區塊為止。

(3)設計方法：由 main.c 呼叫傳入&name1, &name2, &name3 到 r0, r1, r2，在.data 段把組別、三位組員名字寫死，加入三個.word 為三位組員名字的記憶體位置並初始化為 0。在.text 段為了讓 name.s 能被 linker 看見而被 main.c 呼叫，name 函數使用.global 定義在此處。接下來的部分就與期中 Project 報告一致，不再特別說明。

2. id.s

(1)功能：輸入組員的學號，並印出組員學號與學號總和。

(2)要求：

(a) 所有程式須以 ARM Assembly 完成，並可順利執行。

(b) 組員學號需分行印出。(c) 若組員不足 3 位，請重複填入已有的組員學號即可，直到用完 3 個記憶體區塊為止。

(d) 報告除了程式說明外，需以 Code::Block 中，Debug 功能的顯示 Memory dump 的方 式，印出儲存組員學號與總和的記憶體區塊內容，並以螢幕截圖，貼在報告上並說明 其起始與結束記憶體位址。

(e) 在 id.s 程式的第 6 道實體指令須為一道一定要執行的指令：adds lr, pc, r0。

(3)設計方法：

由 main 依序傳入&id1, &id2, &id3, &sum，依序先進入暫存器 r0, r1,r2, r3，r0~r2 先用 mov 指令丟入 r5, r6, r7，接著為了達到特殊指令要求，第五道指令先將 lr 用 mov 備份到 r4，如此第六道指令 adds lr, pc, r0 便不會丟失 lr 原本的值，在第七道亦用 mov 將 lr 用 r4 還原，第八道指令將 r3(sum)丟入 r8。接下來就如同期中 Project 報告一樣，不再特別說明。

3. main.c

(1)功能：整合 drawJuliaSet.s、id.s 與 name.s，先讀入與印出相關資料，按下 p 鍵後繪製 Julia Set 動態畫面，最後整合組別、學號、姓名到最後一個畫面。

(2)要求：

(a) main 函數可直接使用 C 語言撰寫，並可順利執行。

(b) 在 main 中呼叫 NAME、ID、與 drawJuliaSet 之 ARM 組合語言函數，分別達成這三個函數的功能。

(c) 在 main 中使用 NAME 與 ID 所記錄的資料，輸出完整的組別、組員姓名、與組員學 號數值計算結果。

(d) main 函數需適當修改，以便能正確呼叫符合 Midterm Project

規格的 NAME 與 ID 函 數。並能正確呼叫 ARM 組合語言版本之 drawJuliaSet 函數。

（e）報告除了程式說明外，需以 Code::Block 中，Debug 功能的顯示 Memory dump 與 CPU 5 Register 的方式，印出 NAME、ID、drawJuliaSet 三個函數的所在位址與返回位址(Return Address)，並以螢幕截圖，貼在報告上並說明前述記憶體位址與其內容。

（f）報告中需提供 Julia Set 動畫的 5 張畫面，最後一張為包含全組資料與 Julia Set 之結束 畫面。請在 Console 下執行，取得完整的 Julia Set 畫面。

(3)設計方法：

先呼叫 name.s 與 id.s 兩個函數，再印出第一次的資料，並當讀入 p 鍵時準備好 frame buffer，計算並繪製 cX = -700, 400 ≦ cY ≦ 270 的 Julia Set 畫面，最後再印出 Happy New Year 與前面存取的資料。

4. drawJuliaSet.s

(1)功能：計算並決定 Frame 二維陣列裡每個元素的值，並以此來決定該元素投影至畫面(Frame Buffer)上的 Pixel 顏色。

(2)要求：參考範例程式，以 ARM 組合語言重新設計 drawJuliaSet 這個函數，儲存至 drawJuliaSet.s 檔案中，並滿足基本要求的三個項目。

(3)設計方法：這部分會分為幾個部分解釋，分別為 assembler directives、程式基本要求以及每個迴圈在做甚麼事。

assembler directives

.data：把不會動到的資料但因為是參數，故把它們丟在記憶體內，分別為 cX、width、height，並先初始化為 0。

.global:把 drawJuliaSet 函式使用.global 定義在此處，讓 drawJuliaSet.s 能被 linker 看見而被 main.c 呼叫。

.const:把常用到的常數 1500, 4000000, 0xffff, 1000 放在最末尾。

程式基本要求

【3 種不同的 Op2 格式】basic requirement 1

```
add r0, r1, #3 // r0 = r1 + 3
add r0, r1, r2 // r0 = r1 + r2
add r0, r1, r2, lsl #3 // r0 = r1 + (r2 lsl 3)
```

【3 種不同的非 branch 條件執行】basic requirement 2

```
cmp r0, r1
addlt r2, r0, r1 // if(lt) r2 = r0 + r1
cmp r0, r1
addne r0, r1, r2 // if(ne) r0 = r1 + r2
cmp r0, r1
```

sbcgt r0, r1, r2 // if(gt) r0 = r1 - r1 + carry -1

【特定指令 orr sp, lr, r1】basic requirement 3

```
mov r0, sp          @ backup
orr sp, lr, r1      @ The requirement 3
mov sp, r0          @ recall
```

（把 sp 存到 r0 放，再把 lr 和 r1 OR 後丟給 sp，最後把 r0 還給 sp）

迴圈

Xloop：

對應到 drawJuliaSet.c 中的 for (x = 0; x < width; x++){}部分。

Xloop:到 Yloop:之間是在初始化 x < width 與設定 y = 0，初始化完進入 Yloop。

Yloop：

對應到 drawJuliaSet.c 中的 for (y = 0; y < height; y++){}部分。

第一個區塊與上述幾乎相同，是在初始化 y < height；接著的三個小區塊分別對應三行 drawJuliaSet.c 之程式碼：

```
zx = 1500 * (x - (width>>1)) / (width>>1);
zy = 1000 * (y - (height>>1)) / (height>>1);
i = maxIter;
```

在這三個小區塊完成後，先對 while(zx * zx + zy * zy < 4000000 && i > 0){}的條件進行初始化，再進入 While:區塊。

While：

對應到 drawJuliaSet.c 的 while(zx * zx + zy * zy < 4000000 && i > 0){}部分。分為五個區塊，前四個對應到 drawJuliaSet.c 的四行程式碼：

```
int tmp = (zx * zx - zy * zy)/1000 + cX;
zy = (2 * zx * zy)/1000 + cY;
zx = tmp;
i--;
```

最後一個區塊是為了下一輪 while 迴圈所重新 reset/init 條件，與 Yloop:的最後一個區塊基本一致。不過在最後多了 b While 以確保在為滿足終止條件的情況下能進入下一次迴圈。

WhileDone：

分為四個區塊，前三個分別對應：

```
color = ((i&0xff)<<8) | (i&0xff);
color = (~color)&0xffff;
frame[y][x] = color;
```

最後一個區塊是為了 Xloop 的 x++，再跳入 Yloop 的下一輪。

YDone：

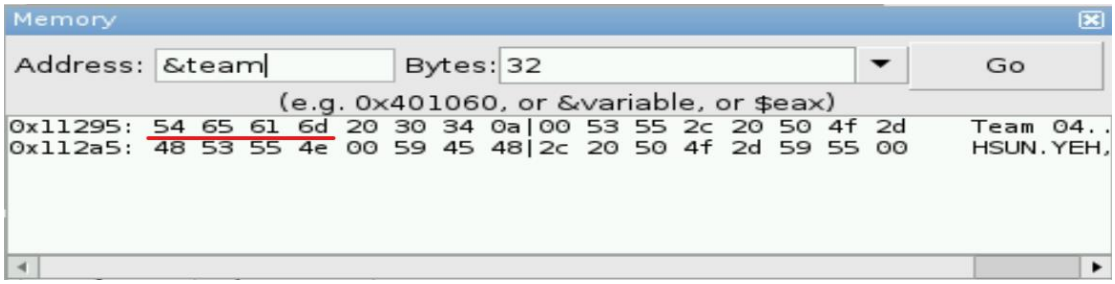
為了 Yloop 的 y++，再跳入 Xloop 的下一輪。

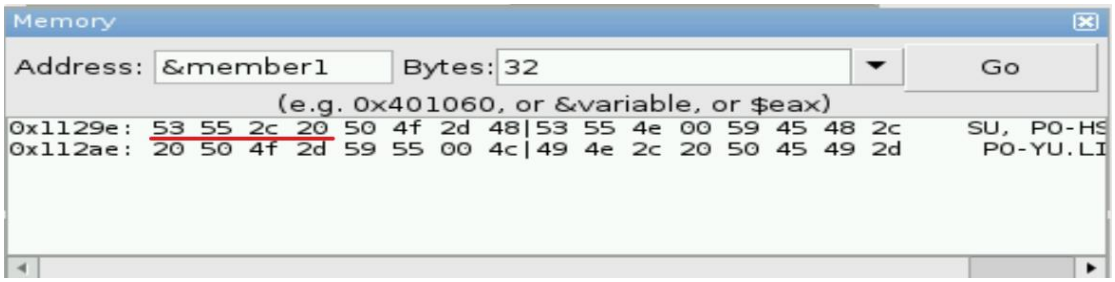
XDone：

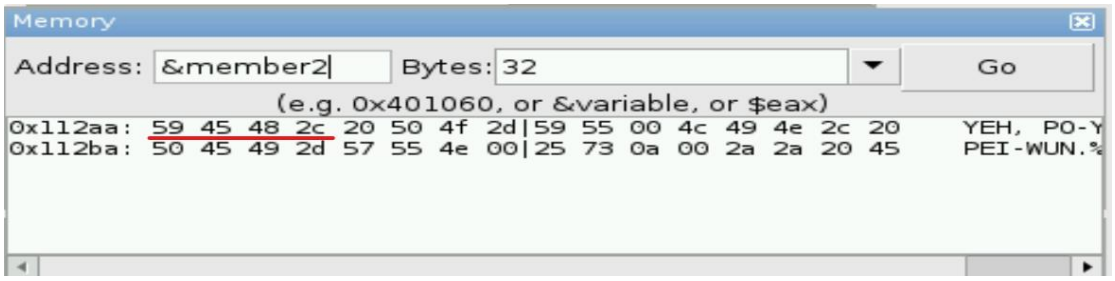
所有迴圈結束，接下來為了滿足要求一與二而有兩個小區塊，第三個是將 r4~r11 與 lr 拉回並更新 sp。

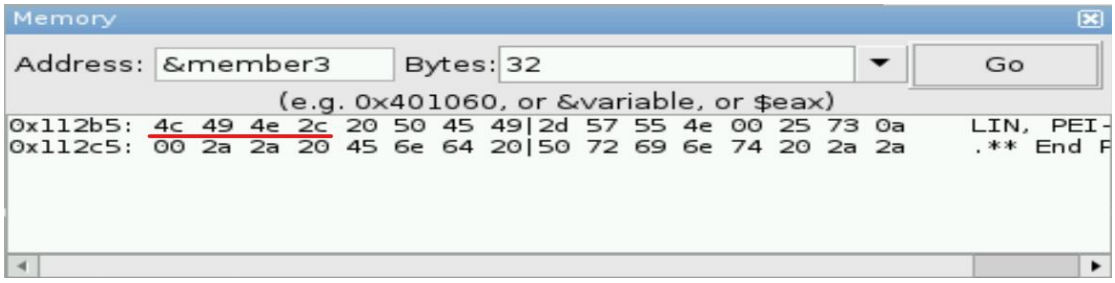
三、 結果

【name.s memory dump】印出儲存組別與組員資料的記憶體區塊內容，並以螢幕截圖，貼在報告上並說明其起始與結束記憶體位址：

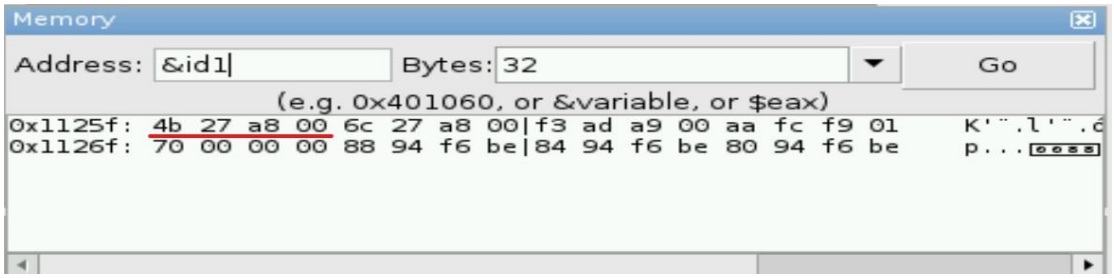


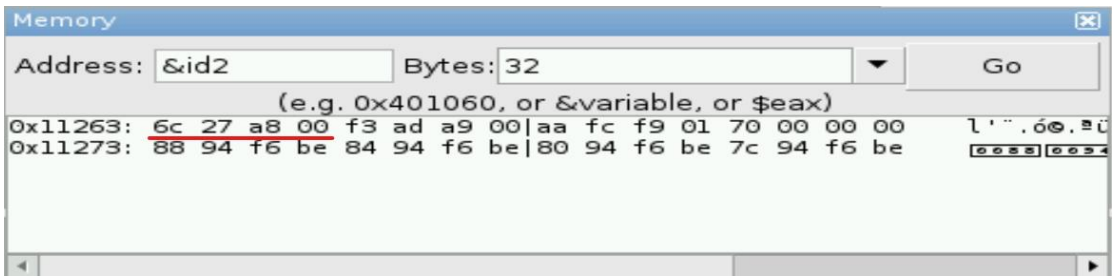


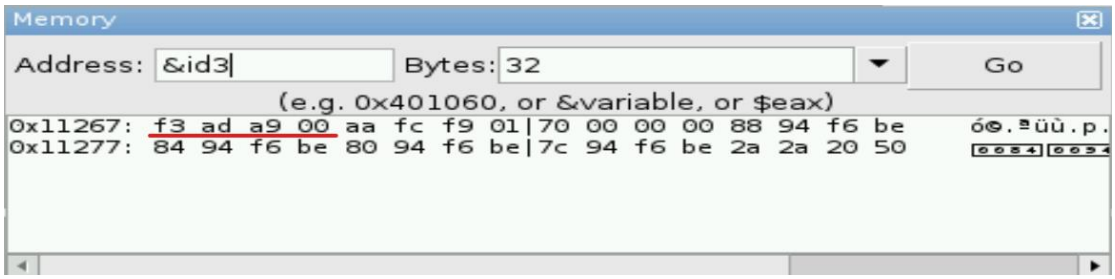


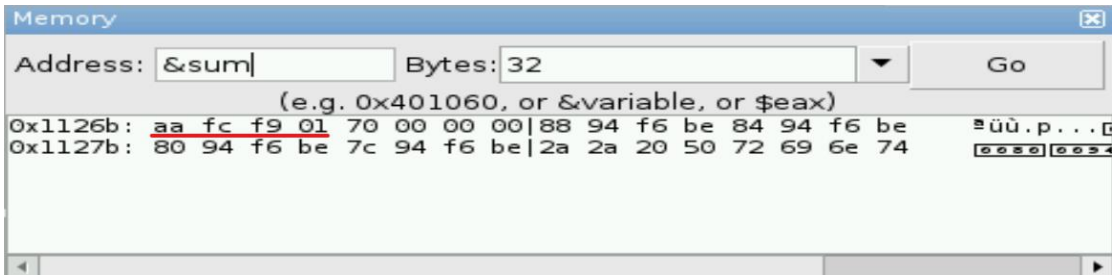


【id.s memory dump】印出儲存組別與組員資料的記憶體區塊內容，並以螢幕截圖，貼在報告上並說明其起始與結束記憶體位址：

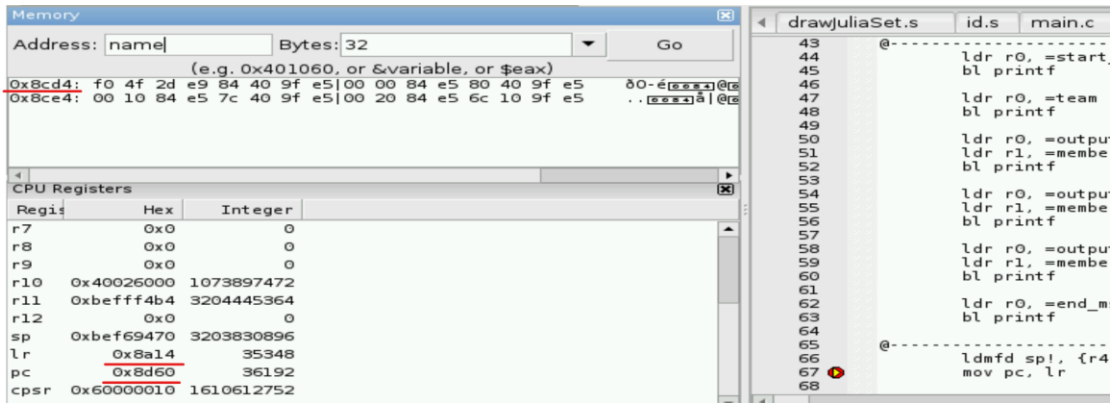




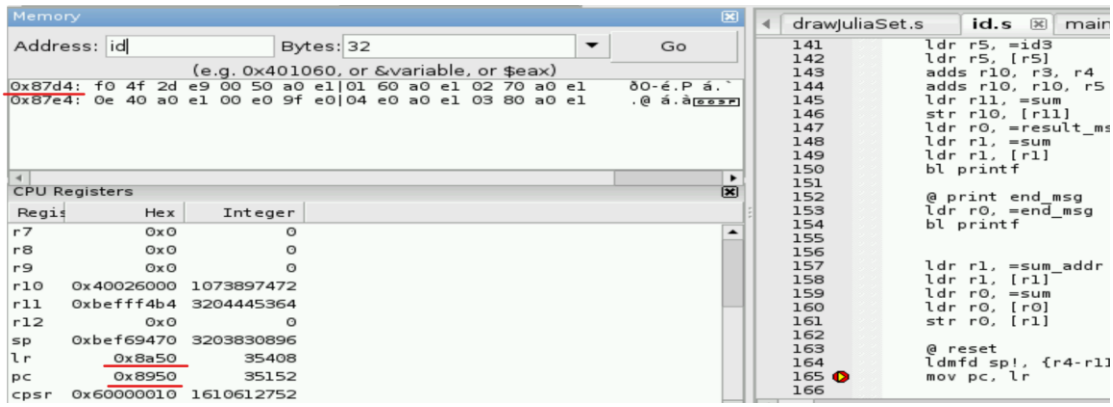




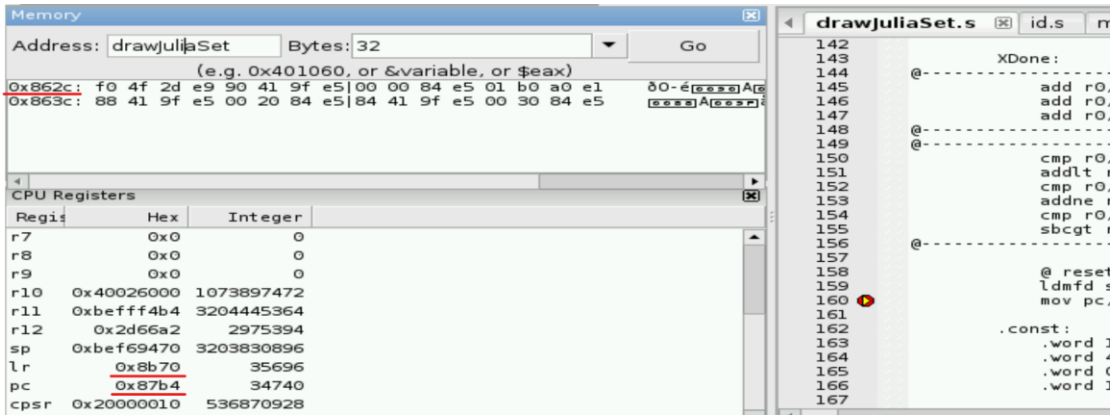
【main.c memory dump & CPU Registers】印出 NAME、ID、drawJuliaSet 三個函數的所在位址與返回位址(Return Address)，並以螢幕截圖，貼在報告上並說明前述記憶體位址與其內容：



name.s 起始位址 0x8cd4 結束位址 0x8d60 返回位址 0x8a14

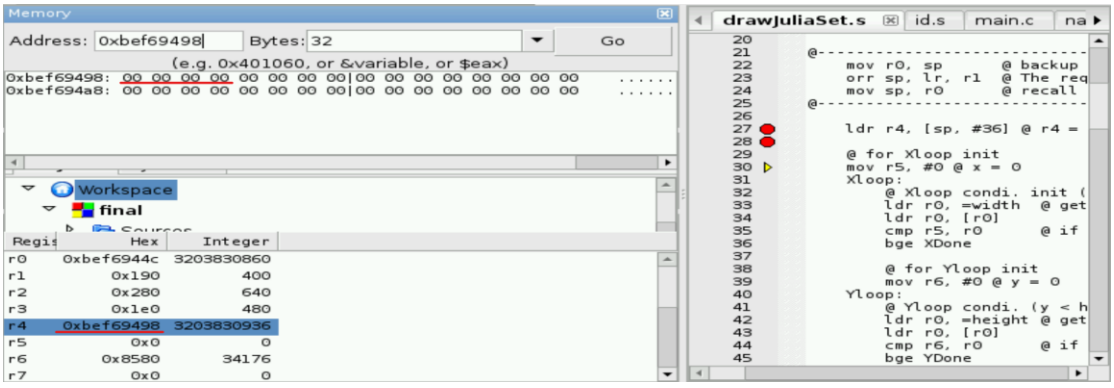


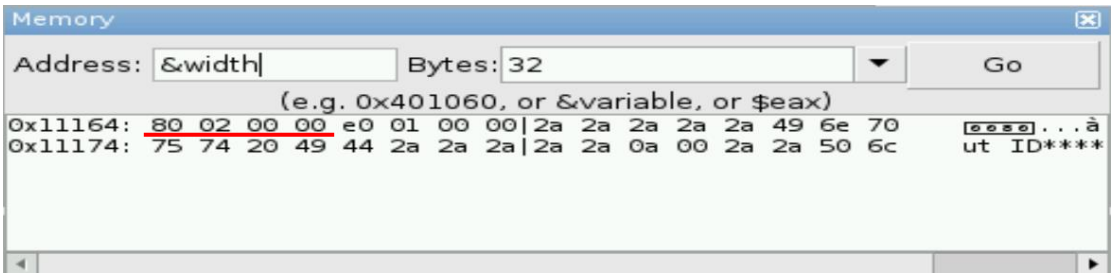
id.s 起始位址 0x87d4 結束位址 0x8950 返回位址 0x8a50

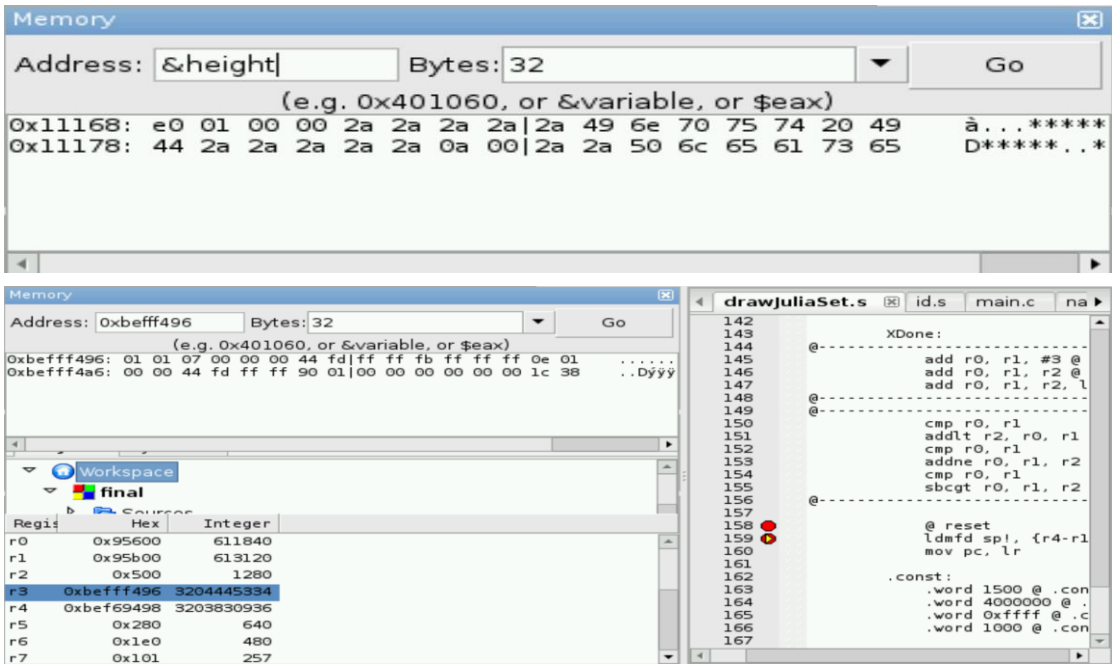


drawJuliaSet.s 起始位址 0x862c 結束位址 0x87b4 返回位址 0x8b70

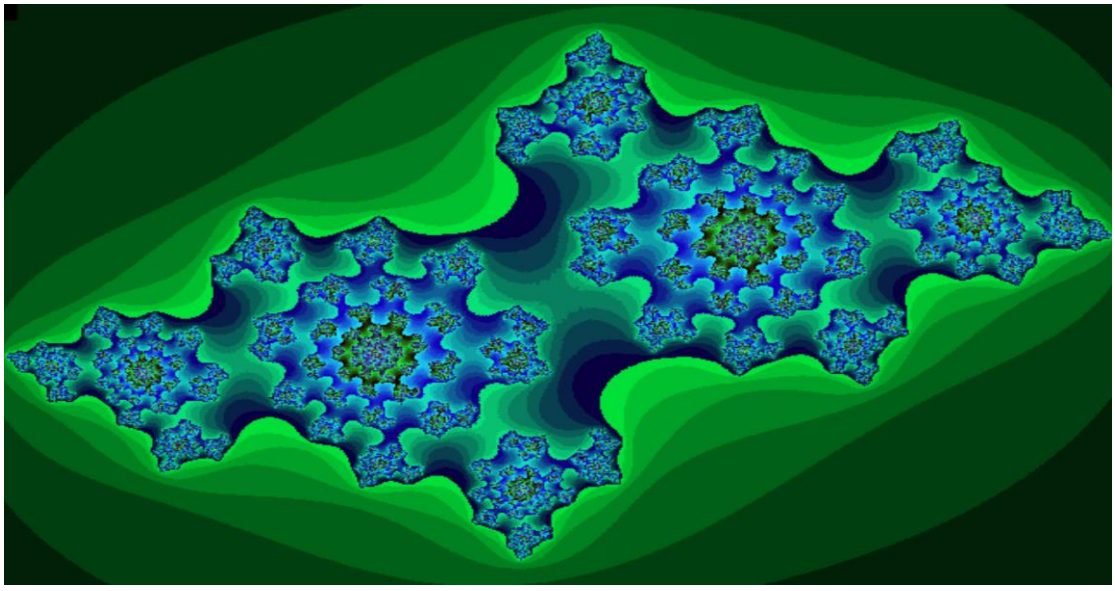
【drawJuliaSet.s】印出 frame 陣列的記憶體區塊部份內容，說明其意義。並以螢幕截圖，貼在報告上，說明 frame 陣列起始與結束記憶體位址：

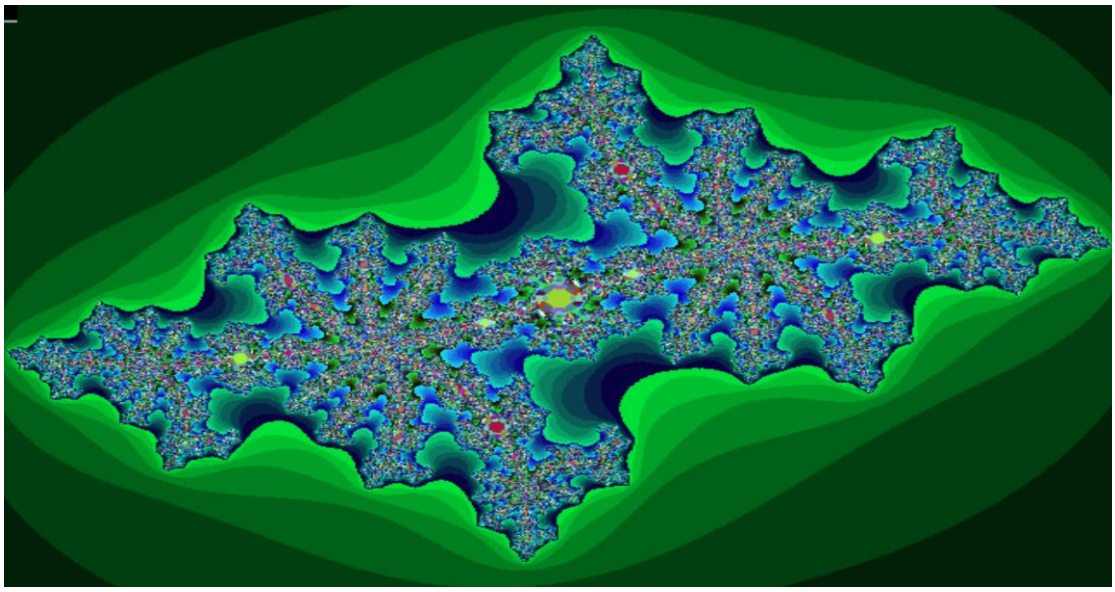


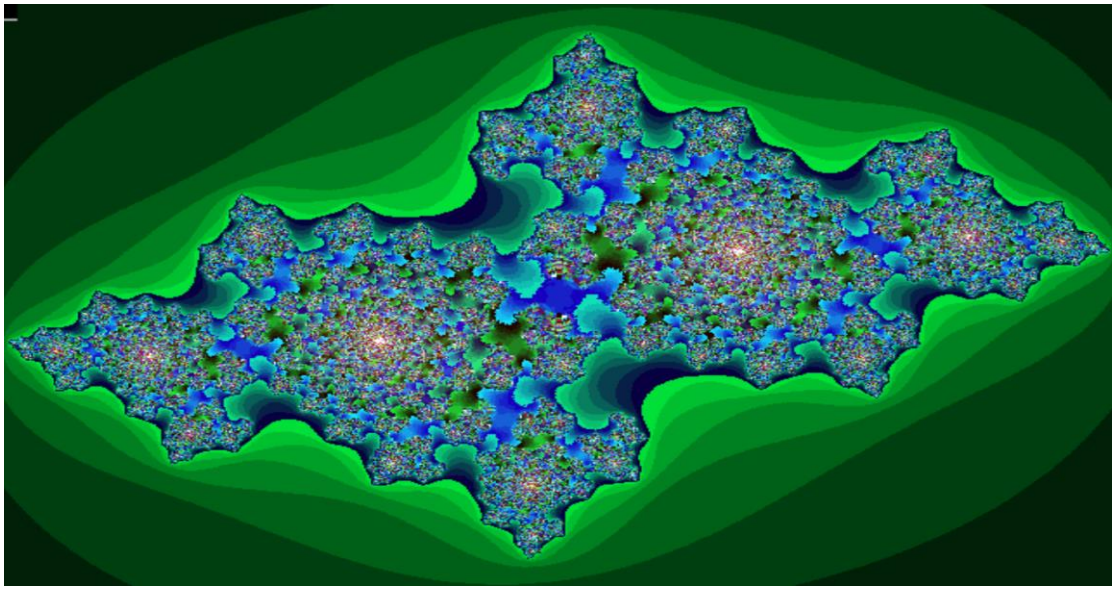


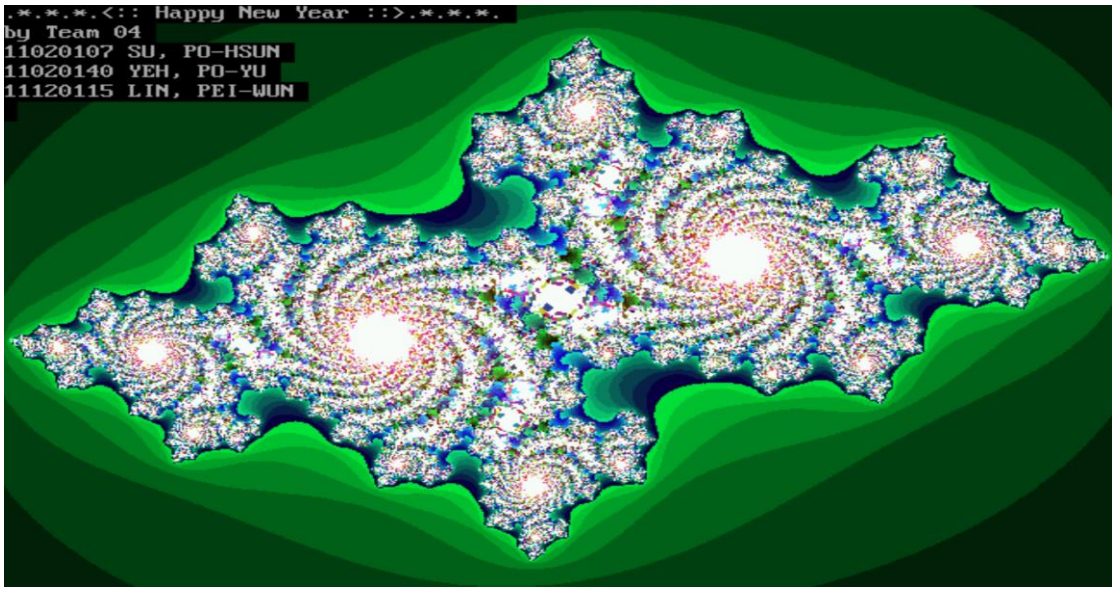


【Julia Set 動畫畫面*5】









四、 討論

我們規劃 r0~r3 = temp，因為不確定__aeabi_idiv 會不會改動到它們。並且 r4 = frame, r5 = x, r6 = y, r7 = i || color, r8 = zx, r9 = zy, r10 = tmp, r11 = cY。由於有三個迴圈且是巢狀，在確認各個 conditional execution 時必須較小心。

五、 結論

很慶幸我們只要用組合語言寫 drawJuliaSet.s 就好了。如果不只計算，連繪製與硬體部分都要用組合語言撰寫（亦即 main），那估計得卡上一個月也不一定能完成。另外就是 memory dump 的功能還是較為不熟悉，常常不知道怎麼找 address，在寫高階語言時不用擔心的問題這時就出現了。透過這次機會能夠慢慢熟悉這些功能是很令人滿足的。

六、 未來展望

期望在未來如果遇到了必須使用 ARM assembly 撰寫的情況時，能夠將這堂課的所學應用到那時，例如在自行開發 GameBoy 的遊戲時就能使用 ARM Thumb Assembly，將所學派上用場。