組合語言與嵌入式系統

期末 Project

110 學年度第 1 學期

老師：朱守禮老師

組別:第 36 組

班別:資訊二乙

學生：資訊三乙 10833230 李郁含

、資訊三乙 10844132 黃中憫

一、 背景

二、 方法

程式說明、設計重點說明之指定之說明項目

(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)drawJuliaSet.s 須包括 1 道一定要執行的指令：adds r14, r0, r15。

(4)3 種 addressing mode

(5)name.s 第七道要是 adcs r13, r1, r2

(6)將 main.s 改寫成 main.c

程式說明:

(1) drawJuliaSet:

起初，我們先將老師給的 project2 裡面的檔案 C 語言範本改寫，讓 main 傳進去的參數數量減少，如此一來才夠少量的暫存器做使用，下面這張圖便是改寫後的 drawJuliaSet 程式碼，之後再按照這個程式碼去做轉換，將它從 C 語言改寫成組語。

我們的drawJuliaSet.c

```c

void drawJuliaSet( int cY, int height, int16_t (*frame)[FRAME_WIDTH] ) {

    int16_t color;
    //RGB16 color;

    int maxIter = 255;

    int zx, zy;
    int tmp;
    int i;
    int x, y;

    for (x = 0; x < 640; x++) {
        for (y = 0; y < 480; y++) {
            zx = 1500 * (x - 320) / 320;
            zy = 1000 * (y - (480>>1)) / (480>>1);
            i = maxIter;

            while (zx * zx + zy * zy < 4000000 && i > 0) {
                int tmp = (zx * zx - zy * zy)/1000-700;
                zy = (2 * zx * zy)/1000 + cY;
                zx = tmp;
                i--;
            }


            color = ((i&0xff)<<8) | (i&0xff);
            color = (~color)&0xffff;

            frame[y][x] = color;
        }
    }
}
```

將一些不會隨著迴圈更改的值以數字直接代入，例如:雖然有傳 height 這個參數，但在為 width 的地方直接寫為 640、在為 height 的地方直接寫為 480、在為 cX 的地方直接寫為-700，雙迴圈中:

`zx = 1500 * (x - (width>>1)) / (width>>1);`

這行程式中的 width>>1 直接寫為 width>>1 = 640 / 2 = 320

而下一行的:

`zy = 1000 * (y - (height>>1)) / (height>>1);`

卻不改

為了能湊出做到指定之說明項目(1) 3 種 Operand2 格式

除了修改了以上這些，其餘部分基本上都和老師的範例程式差不多

接著便開始將他改寫成組語，首先我們是先將 drawJuliaSet 函數設成全域變數並將他放置於 text 段，接著將一些不是立即值的數放到 constant，然後將從 main 傳來的參數從暫存器 mov 進計畫好要固定放著的暫存器裏頭，r0(cY) mov 進 r11，r2(frame) mov 進 r4。

在 drawJuliaSet 裡面我們對於暫存器的規劃大概是這樣放:

r0 = temp

r1 = temp

r2 = temp

r3 = temp

r4 = frame

r5 = x

r6 = y

r7 = color, i

r8 = zx

r9 = zy

r10 = tmp

r11 = Cy

r13 = sp

r14 = lr

r15 = pc

其中 r0~r3，我們按照講義上所說，僅讓它做傳遞參數和暫存的功用，需要儲存的變數則是放在 r4~r11 裡面。

接著開始跑雙迴圈讓放進 r5 暫存器的 x 直接跟 640 比，讓放進 r6 暫存器的 y 直接跟 480 比，沒有符合條件的話就直接跑另外寫的 function outx 或 outy。

然後是運算式的部分，

`zx = 1500 * (x - 320) / 320`

運用了 sub、mul 搭配暫存器做簡單的運算，特別要說的是由於 1500 不是立即值的數因此是從 constant load 進暫存器

在寫這隻程式時，如同一開始所說的，將原本的簡化過後，直接將數字帶入，這樣可以減少使用的暫存器，不過有些數字並非立即值，不能直接放進去，所以我們有另外寫一個放非立即值的常數區域，在用到時才去裡面取出數值。

```
.constant:
    .word   1500            @.constant
    .word   4000000         @.constant+4
    .word   0xffff          @.constant+8
```

最後我們前面先將 320 mov 進 r1，(1500*(x - 320)) mov 進 r0，因為在呼叫 __aeabi_idiv 前要先設定，r0 為被除數，r1 為除數，然後存到 r8(zx)。

運算式: `zy = 1000 * (y - (height>>1)) / (height>>1)`

我們直接將 480 mov 進 r0，然後做邏輯右移，然後做運算，除法跟前面那個一樣，最後存到 r9(zy)。

接著做 `zx * zx + zy * zy` 的運算讓後面進入 while 進行迴圈的判斷

做`(zx * zx - zy * zy) / 1000 - 700` 的運算丟進 r10(tmp)

然後做 zy = (2 * zx * zy)/1000 + cY 運算、zx = tmp 運算及將 i 的值減一，更新 zy、zx 和 i，直到不符合條件便會跳到 whileout，將 i and 0xff 並存到 r0，再來將 r0 和 r0 邏輯左移 orr 的結果存到 r7，再將存在 constant 的 0xffff(不是立即值的數)存到 r0，最後做(~r7) & 0xffff，將 r0 bic r7 的結果存進 r7。

接著做 frame[y][x] = color = frame + 1280*y +2*x 的運算式，將結果 store 進 r7，後面更新 y+1 回到 fory 迴圈。

而 outy 的部分則是更新 x+1 並回到 forx 迴圈，outx 做最後的還原 r4-r11, lr 且將 program ccounter 設為 return address 去執行下一道指令。

(2) name:

先將標題名、組別和組員各個姓名(我們只有兩名組員)寫在 data 段，並包括本身 function 設成全域變數方便之後 main.s 可以 call。

為了能做到老師「Project 基本要求」的第五個要求:name.s 第七道要是 adcs r13, r1, r2，首先我們先將 r13 備份到 r1，再將前面各設成 0 的 r2、r3 相加等於 r2，做到 r13 = r1(r13) + r2(0)並設定 CPSR，最後一直 printf 組別和姓名。

```
name:   stmfd sp!, {lr}          @push return address onto stack
        ldr   r0,  =msgStart     @load pointer to format string
        mov   r1,  r13           @back up r13 to r1
        mov   r2,  #0            @r2 = 0
        mov   r3,  #0            @r3 = 0
        adds  r2,  r3            @r2+r3
        adcs  r13, r1, r2        @r13 = r1(r13) + r2(0) set CPSR
        bl    printf             @printf("*****Print Name*****\n")
```

(3) id:

先將標題名寫在 data 段，然後因為 main.c 是用 call by reference 的方式傳遞參數，所以我們先將 ID1、ID2、ID3 和 IDSum 變數的回傳位址存起來，等運算完後再將資料放入那個位址傳回 main.c。這部分是我們和期中時做的不一樣的地方。

```
id(&ID1,&ID2,&ID3,&IDSum);
```

```
id : stmfd  sp!,    {r4,r5,r6,lr}    @push return address onto stack

     ldr  r4,     =ID1address
     str  r0,     [r4]

     ldr  r4,     =ID2address
     str  r1,     [r4]

     ldr  r4,     =ID3address
     str  r2,     [r4]

     ldr  r4,     =IDSumaddress
     str  r3,     [r4]
```

```
ldr  r1,     =ID1address
ldr  r1,     [r1]
ldr  r0,     =ID1
ldr  r0,     [r0]
str  r0,     [r1]
```

之後將自己本身的 function、三組學號和它們的加總設成全域變數，方便之後 main.c 可以 call，scanf 三組 ID 後把它們加總起來設置到 IDSum，在加總的過程中，我們寫了老師「Project 基本要求」的三個要求: (1)3 種 addressing mode(2)3 種 operand2 格式 (3)3 道以上不同的非 Branch 指令的 Conditional Execution，三種不同的 Conditional Execution:addne、addeq、addpl 去偵測要不要做加的動作，operand2:將 IDSum 做了邏輯左移，又做減法算數右移回歸 IDSum 原本的值，接著 scanf 命令字元 command，經過比較若 command 是 p 的話就列出三組學號及其加總再回到 main.c；否則直接回到 main.c。

(4) main:

下圖是我們呼叫寫的 3 個.s 檔

```
void name();
void id(int* ID1,int* ID2,int* ID3,int* IDSum);
void drawJuliaSet(int cY,int ,int16_t (*frame)[FRAME_WIDTH]);
```

name.s

```
    //Dummy Function. Please refer to the specification of Project 1.
    name();
```

id.s

```
    //Dummy Function. Please refer to the specification of Project 1.
    id(&ID1,&ID2,&ID3,&IDSum);
```

drawJuliaSet.s

```
for( cY=400 ; cY>=min_cY; cY = cY + cY_step ) {

    // 計算目前cX,cY參數下的Julia set畫面
    drawJuliaSet( cY, FRAME_HEIGHT, frame );

    // 透過低階I/O操作呼叫Frame Buffer的驅動程式
    // (將畫面資料寫入Frame Buffer)
    write( fd, frame, sizeof(int16_t)*FRAME_HEIGHT*FRAME_WIDTH );

    // 移動檔案操作位置至最前端，以便下一次的畫面重新寫入
    lseek( fd, 0, SEEK_SET );
}
```

main 的部分我們是將老師給的 project2 裡面的檔案範例程式.c 檔稍作修改，將內容改成我們組的(學號、名字的部分)，並在呼叫 id 的函式加入傳遞 &ID1,&ID2,&ID3,&IDSum 參數，最後將 drawJuliaSet 函式改成 drawJuliaSet(cY, FRAME_HEIGHT, frame)只傳遞三個參數，大概是以上這些地方，其他地方都與老師的差不多，main function 流程大概是這樣，首先我們會先印出 Function1: Name，接著呼叫 name 先印出名字，然後再印出 Function2: ID 後呼叫 id，輸入完我們組員的學號之後，就會將學號和名字一起印出來，在 name 和 id 執行結束後，就會印出***** Please enter p to draw Julia Set

animation *****要求使用者輸入 p 來執行 drawJuliaSet，如果使用者輸入為 p 後，就會宣告一個 frame 的二維陣列，而這個二維陣列代表的是一張圖的每一格像素，陣列的 width 為 480 而 height 為 640，呼叫完 drawJuliaSet 後就會進入 write，將之前在 drawJuliaSet 設定好的圖畫出來，然後繼續跑迴圈，跑完迴圈後.先印出*.*.*.<:: Happy New Year ::>.*.*.*.後再次印出學號和名字

設計重點說明之指定之說明項目:

(1)需使用 Data Processing 指令中，13 種 Operand2 格式的當中 3 種以上。

a. sub r0, r5, #320 常數表示

```
sub     r0,    r5,    #320      @temp = x - 320 ***Operand2***
```

在做 zx = 1500 * (x - 320) / 320 這項運算式的其中 x - 320 這 part 的這道指令

b. mov r2, r0, LSR #1 邏輯右移

```
mov     r2,    r0,    LSR #1     @height>>1   ***Operand2***
```

在做 zy = 1000 * (y - (height>>1)) / (height>>1)這項運算式的其中 height>>1 這 part 的這道指令，前面已經先將 480(height)存到 r0，這道指令將 r0 邏輯右移 1 後存到 r2，再做後面的運算。

c. orr r7, r0, r0, LSL r1 邏輯左移

```
orr     r7,    r0,    r0, LSL r1 @ r7 = (i & 0xff) | ((i & 0xff) << 8 ) ***Operand2***
```

在做 color = (i & 0xff) | ((i & 0xff) << 8)這項運算式的這道指令，前面已經先將 i & 0xff 存到 r0，也將 8 存到 r1，這道指令將 r0 邏輯左移 r1(8)後和原本的 r0 orr 將結果存到 r7。

(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)。

a. movge r14, r2

```
adds  r14,  r0,  r15  @ !!!!!!
movge   r14,  r2  @ back r14 ***conditional Execution***
```

為了做到指定指令 adds r14, r0, r15，我們前面先將 r14(link register)備份到 r2，接著做完指定指令 adds r14, r0, r15 後做這項指令進行還原的部分。

b. moveq r0, r1

```
mov   r0,   #480          @r0 = 480(height)
mov   r2,   r0,  LSR #1   @height>>1  ***Operand2***
sub   r0,   r6,   r2      @temp = y - (height>>1)
mov   r1,   #1000         @r1 = 1000
cmp   r0,   r1            @help next instruction
moveq r0,   r1            @***conditional Execution***
```

為了做到這項指令我們在前面加了 cmp r0, r1，確保 r0 與 r1 不可能相同，就絕對不會 eq 而需要 mov r1 到 r0

c. subge r2, r2, r3

```
add   r2,   r0,   r1      @r2 = r0(zx * zx) + r1(zy * zy)
ldr   r3,   .constant+4   @r3 = 4000000
cmp   r2,   r3
bge   whileout            @zx * zx + zy * zy >= 4000000
@zx * zx + zy * zy < 4000000
subge  r2,   r2,   r3     @***conditional Execution***
```

因為前面已經有一個判斷 r2 是否 ge r3，如果是的話一定會跳到 whileout 因此後面再判斷一次是絕對不會需要做 r2-r3 這個指令的。

(3)須包括 1 道一定要執行的指令：adds r14, r0, r15。

我們的做法和期中 project 時的做法類似，r0 和 r15 因為不會變動所以我沒去做任何事，我唯一做的就是將 r14 這個 return address 備份到 r2 暫存器，然後就執行老師要求的這道指令。原本其實我還有加入備份還原 CPSR 的 MSR 和 MRS 指令，因為 adds 會更新 CPSR，可能會影響到後面的 Conditional Execution 執

行，不過後來我將這道指令移到判定 for 迴圈之前，這樣的話就不用擔心了，因為進迴圈的第一件事情就是比較，所以會重新設定 CPSR，所以最後的程式碼就如下圖，基本上只對 r14 做備份還原而已。

```
drawJuliaSet: stmfd   sp!,   {r4-r11,lr}     @push return address onto stack
              mov     r11,   r0              @r11 = r0(cY)
              mov     r4,    r2              @r4 = r2(frame)
              mov    r2,   r14   @ copy lr
              adds   r14,   r0,   r15   @ !!!!!!
              movge    r14,   r2   @ back r14 ***conditional Execution***

              mov     r5,    #0              @r5 = 0 int x=0
forx:         cmp     r5,    #640            @ x<640
              bge     outx
```

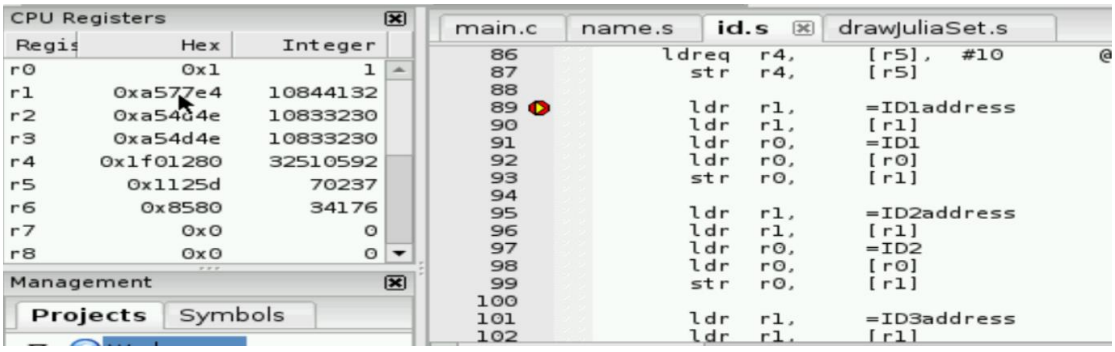
三、結果

程式驗證結果

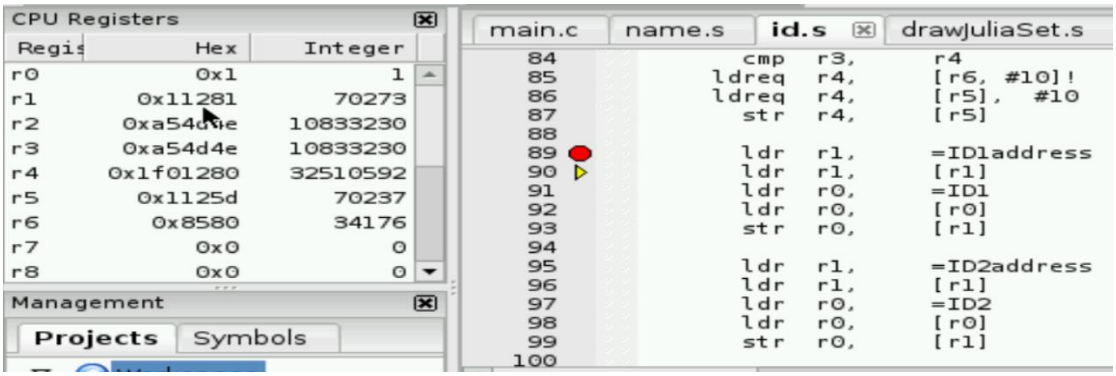
(1)id.s

以 id 的 ID1 為例，圖中 ID1address 是存回傳回 main.c 的位址，我們先將這個位址放到 r1，然後把正確的 ID1 放到 r0，最後把 ID1 放入 r1，這樣到時候它就會回傳到 main.c。其餘的 ID 皆是照這個做

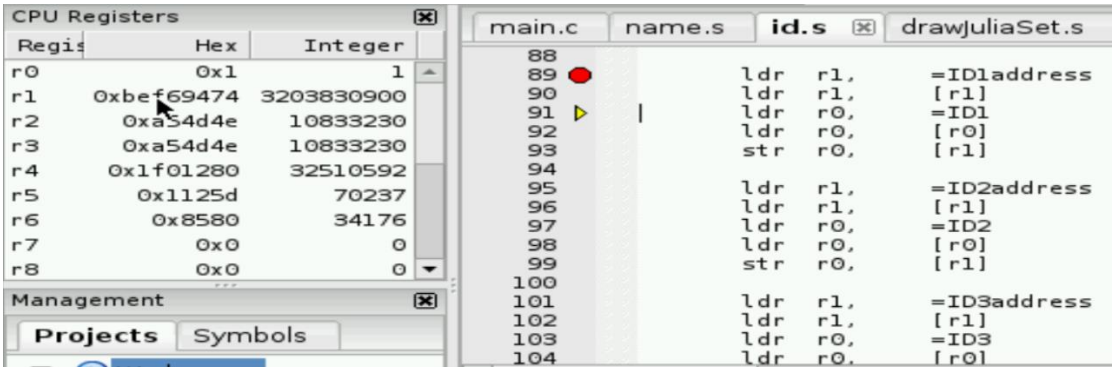
下圖是存回傳位址到 r1



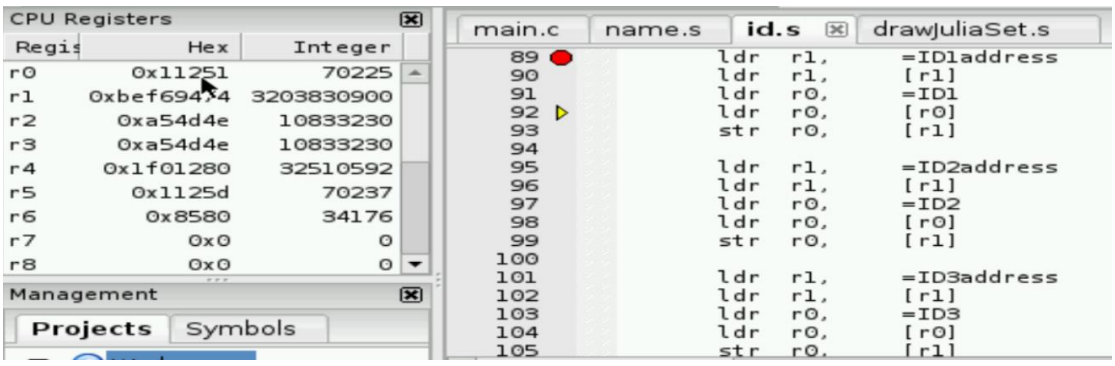
把值存入



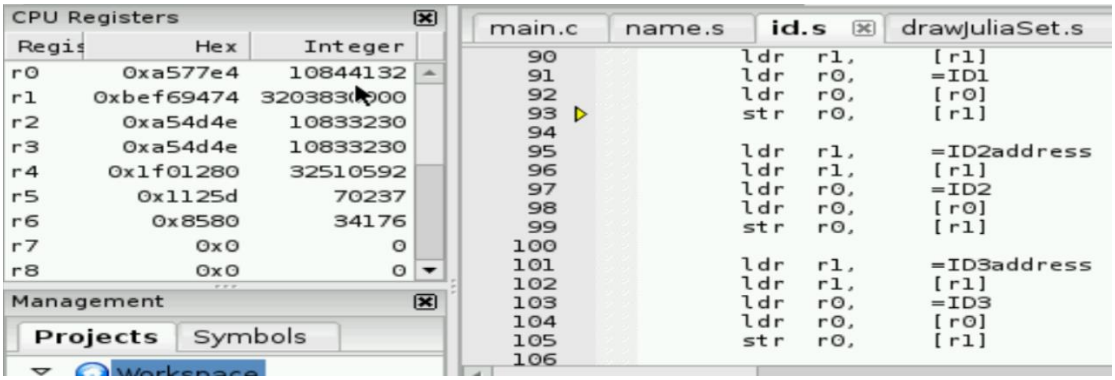
把正確的 ID 值的位址存到 r0



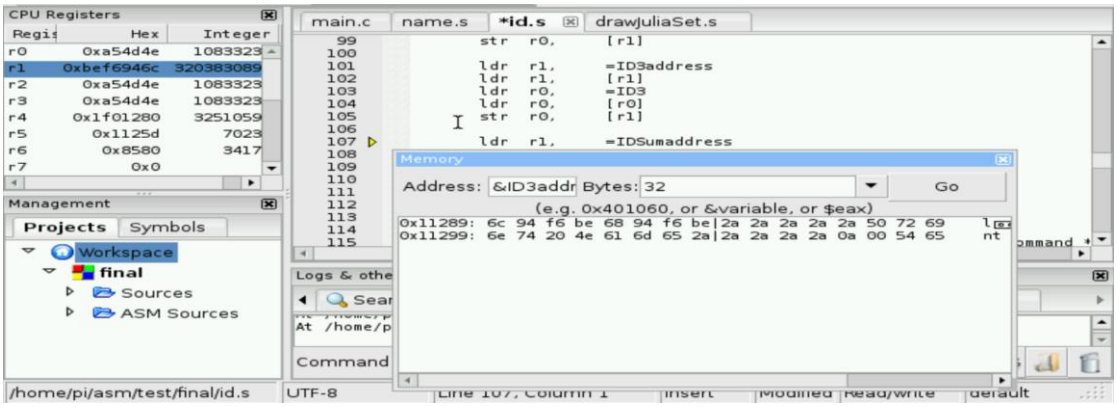
把正確的 ID 值存到 r0

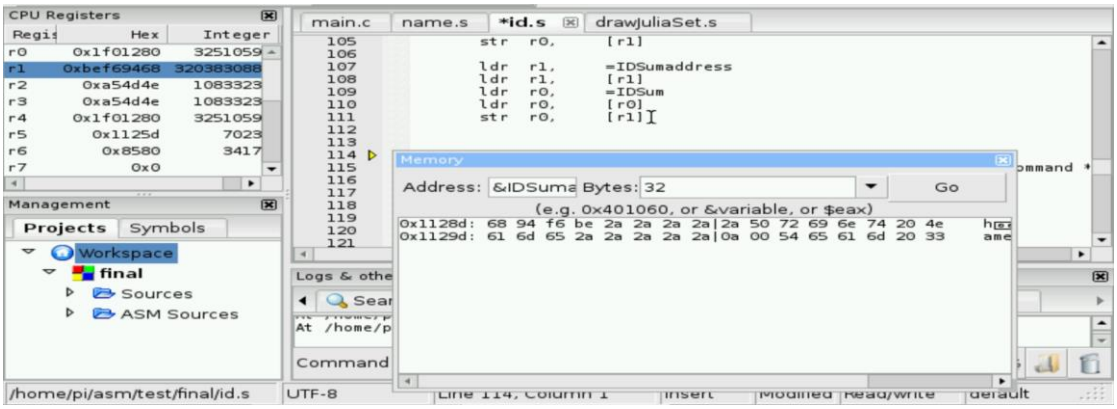


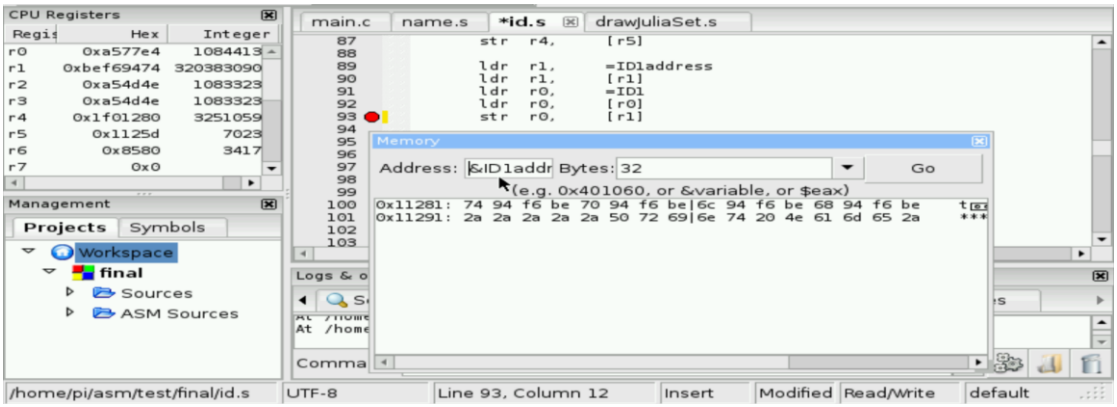
把正確的 ID 值的存到回傳位址

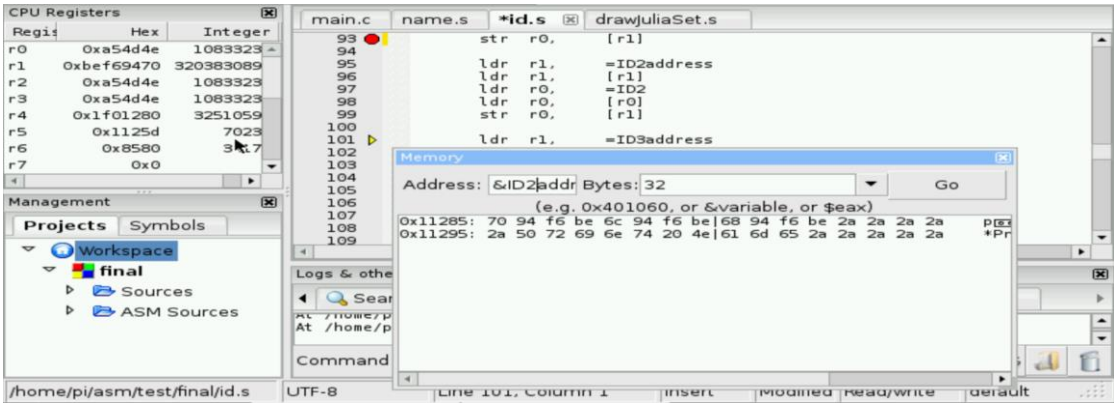


下圖為 ID 回傳 main.c 的位址

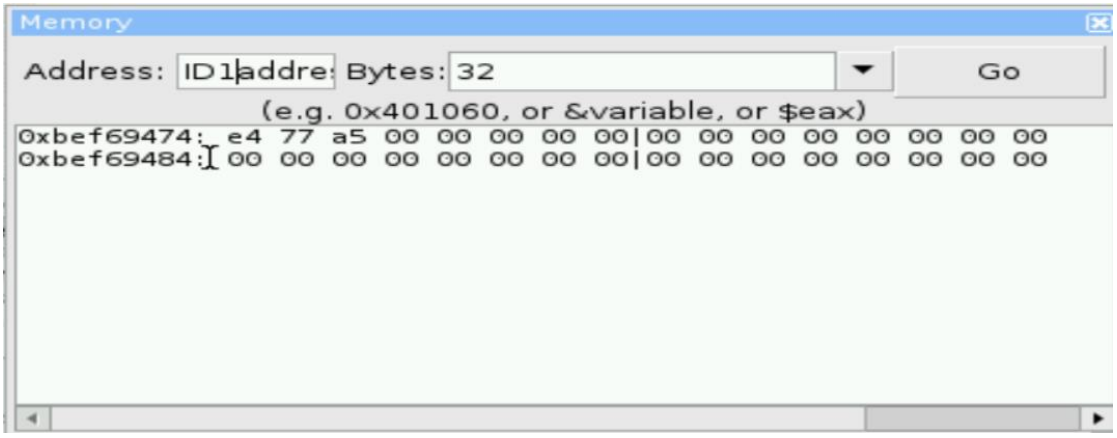


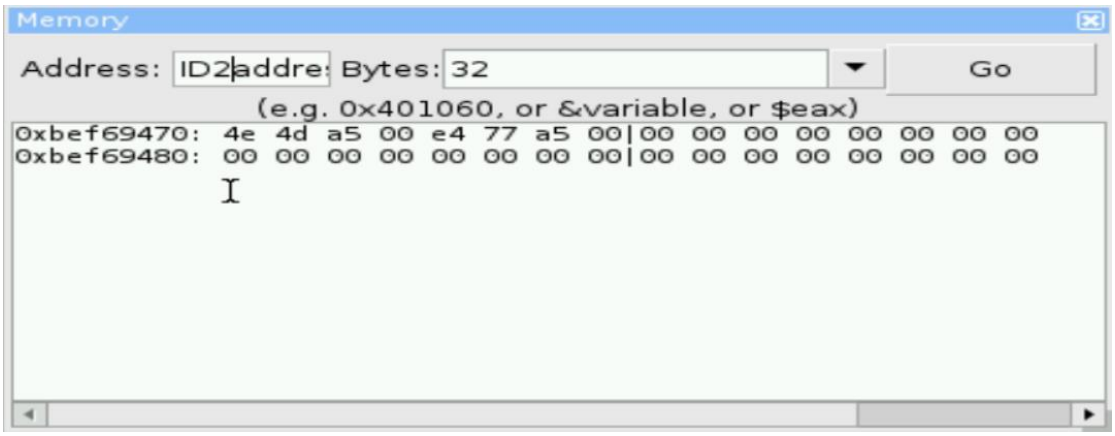


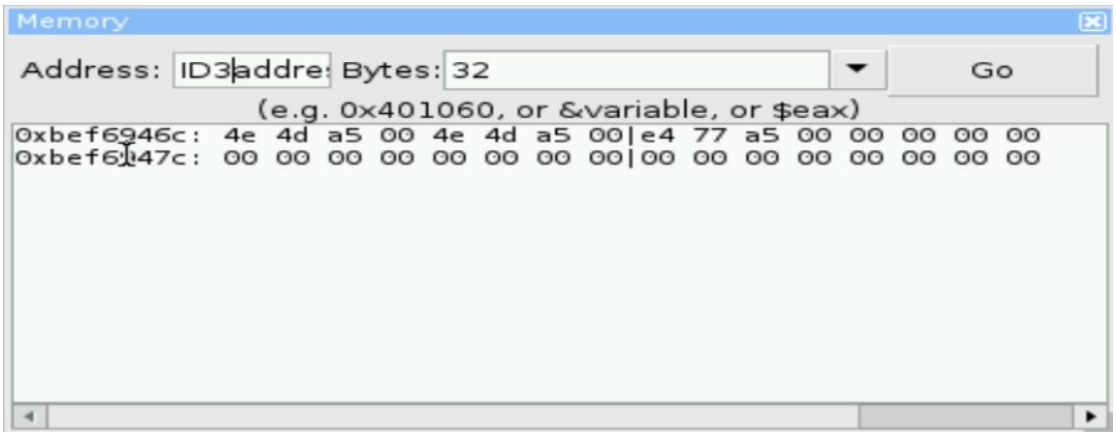


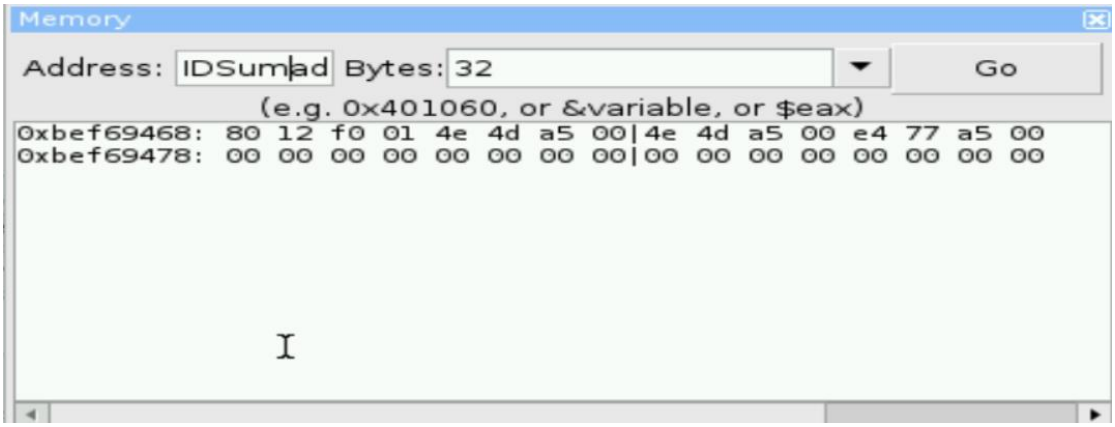


下圖為 ID1、ID2、ID3 和 IDSum 的值



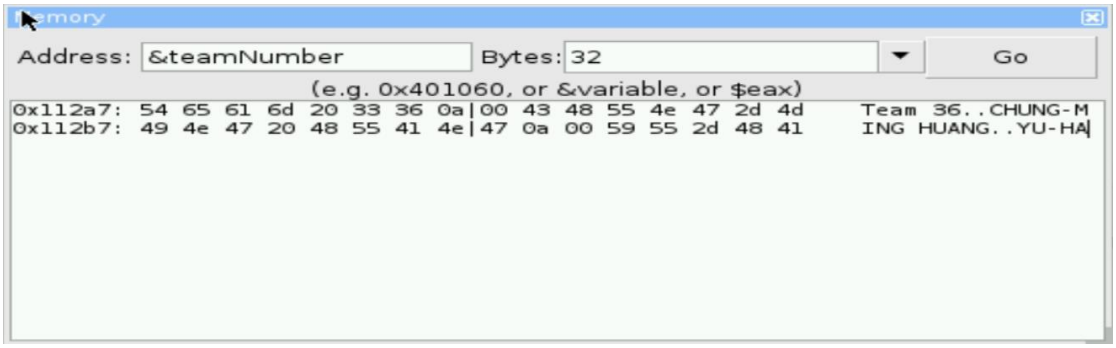


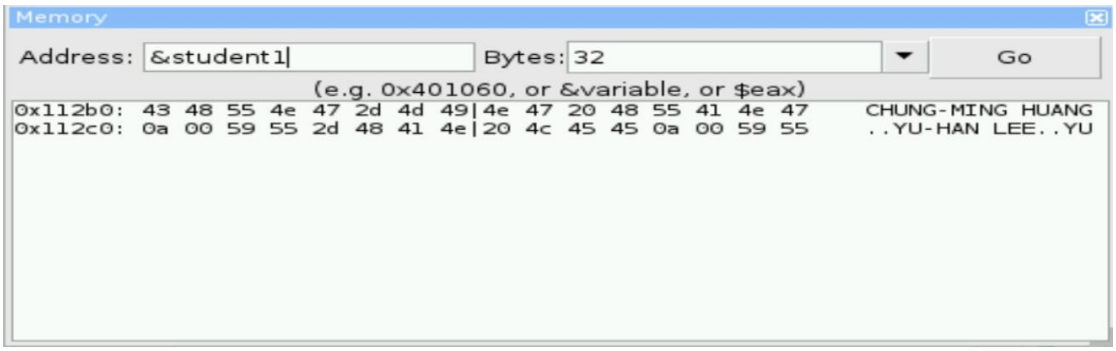


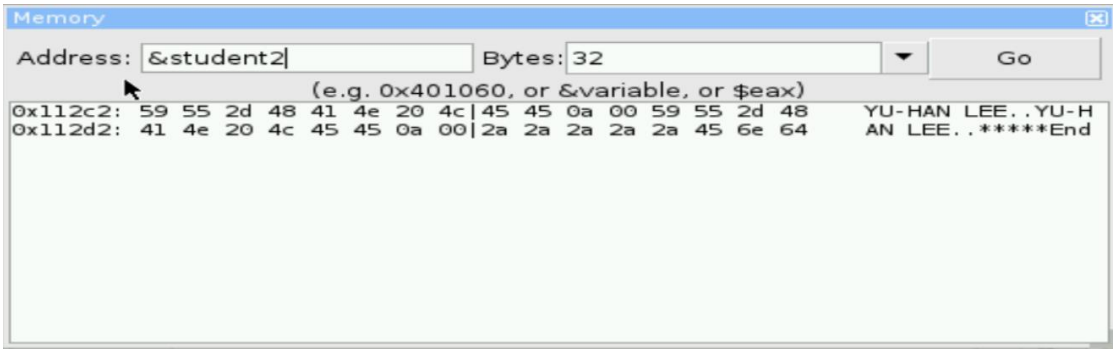


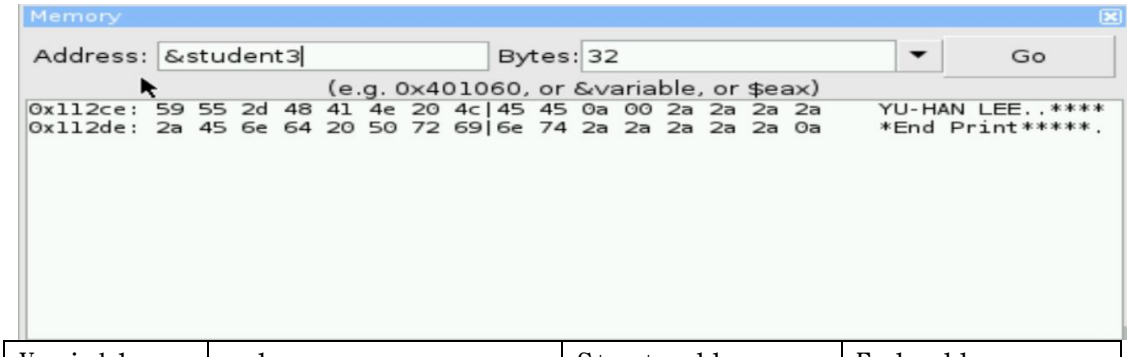
Variable	value	Start address	End address
id1	10844132	0xbef69474	0xbef69477
id2	10833230	0xbef69470	0xbef69473
id3	10833230	0xbef6946c	0xbef6946f
idSum	32510592	0xbef69468	0xbef6946b

(2)name.s









Variable	value	Start address	End address
teamNumber	“Team 36”	0x112a7	0x112af
student1	“CHUNG-MING HUANG”	0x112b0	0x112c1
student2	“YU-HAN LEE”	0x112c2	0x112cd
student3	“YU-HAN LEE”	0x112ce	0x112d9

(3)drawJuliaSet:

在 drawJuliaSet 中，我們先將各自變數的記憶體位置存好，其中我們有把 r0~r3 空出來，如此一來未來如果被動到才不會出問題，接著便開始將他轉變為.s，也順便完成了老師要求的指定之說明項目

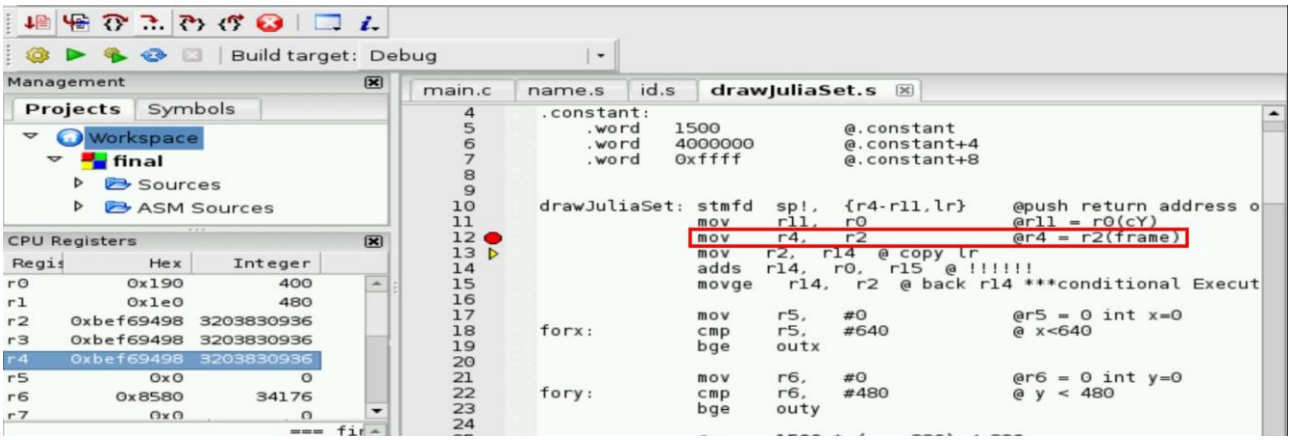
(1)需使用 Data Processing 指令中，13 種 Operand2 格式的當中 3 種以上。

(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)。

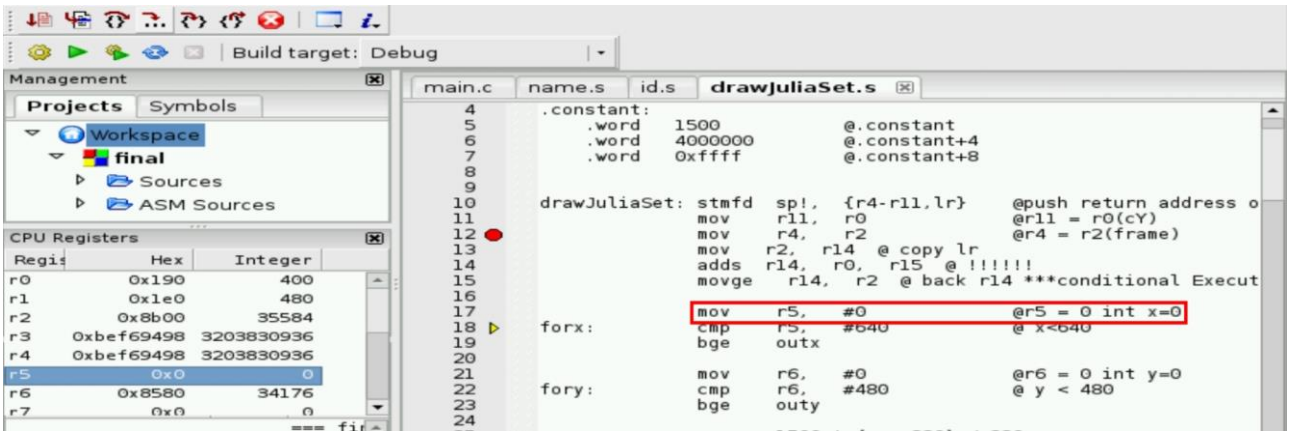
(3)須包括 1 道一定要執行的指令：adds r14, r0, r15。

此程式中的雙迴圈最外層的 forx 迴圈是 width，第二層的 fory 迴圈是 height，雙迴圈會跑經過每一幀的每一格將那一格填上顏色，接著再跑下一張圖，直到執行指令結束。

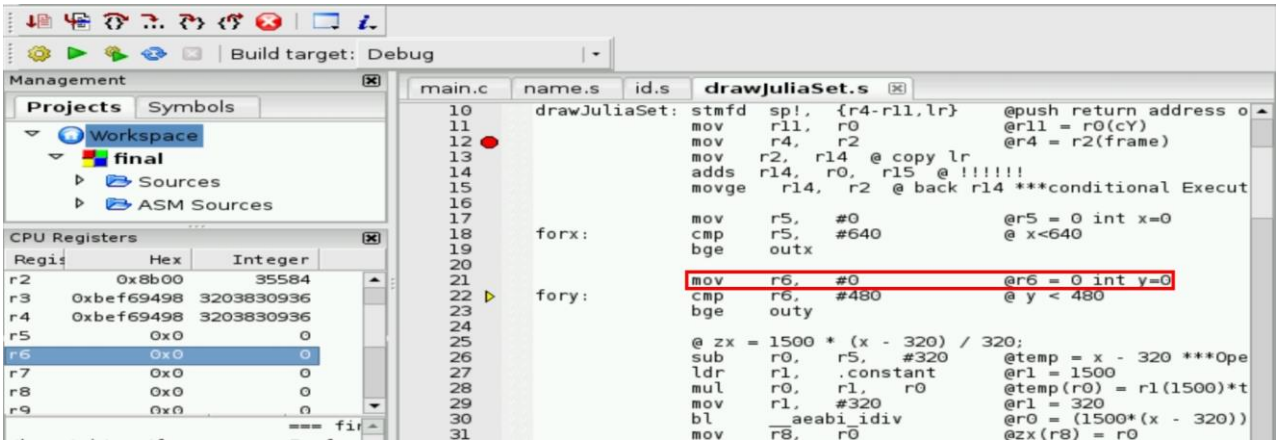
這是將 frame 從原本 main 傳遞進來的在 r2 mov 到 r4



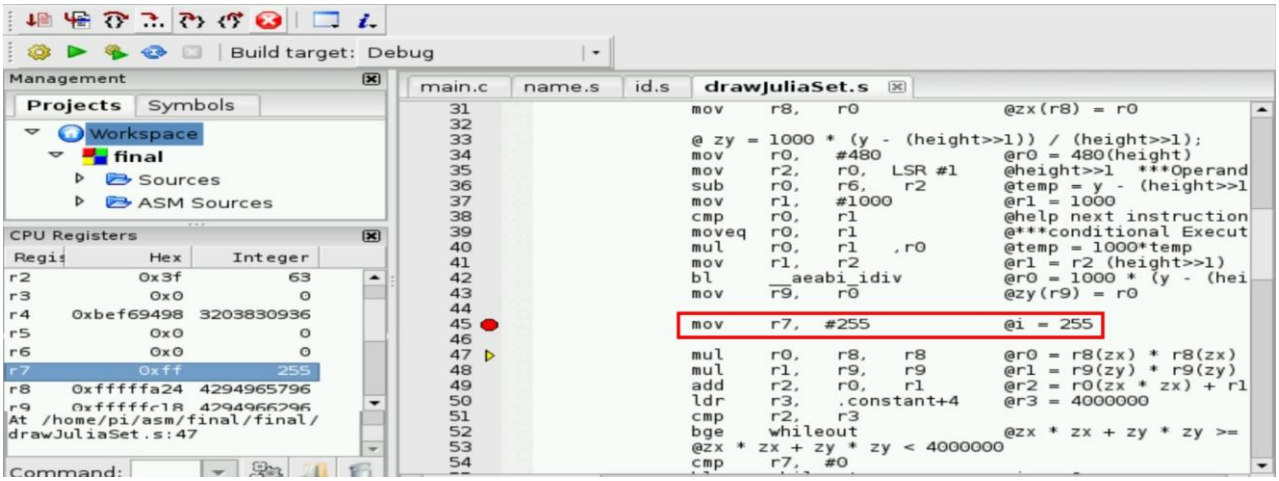
這是將 x 存到 r5，且初始值為 0



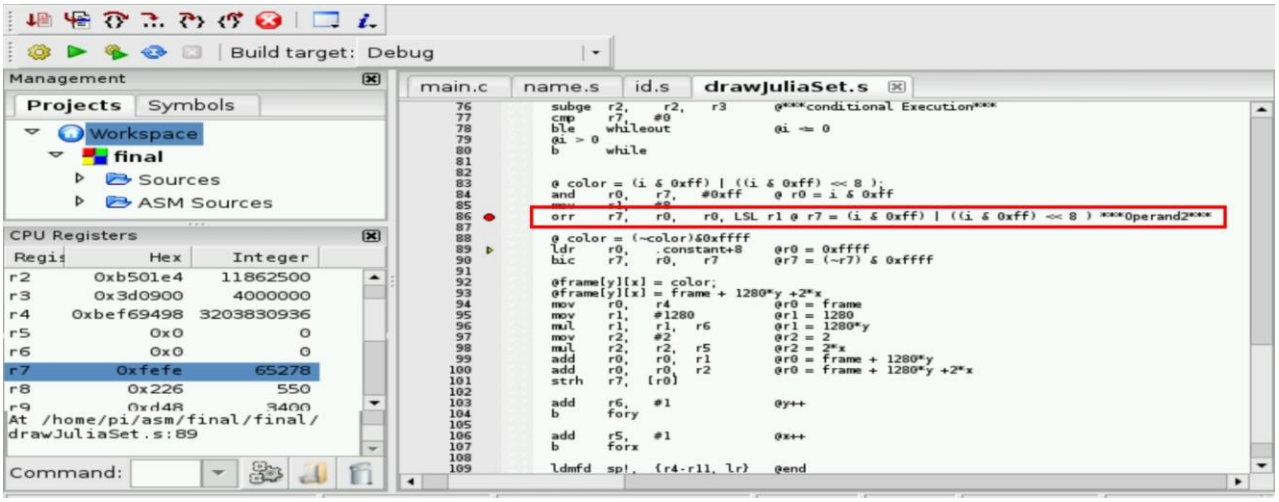
這是將 y 存到 r6，且初始值為 0



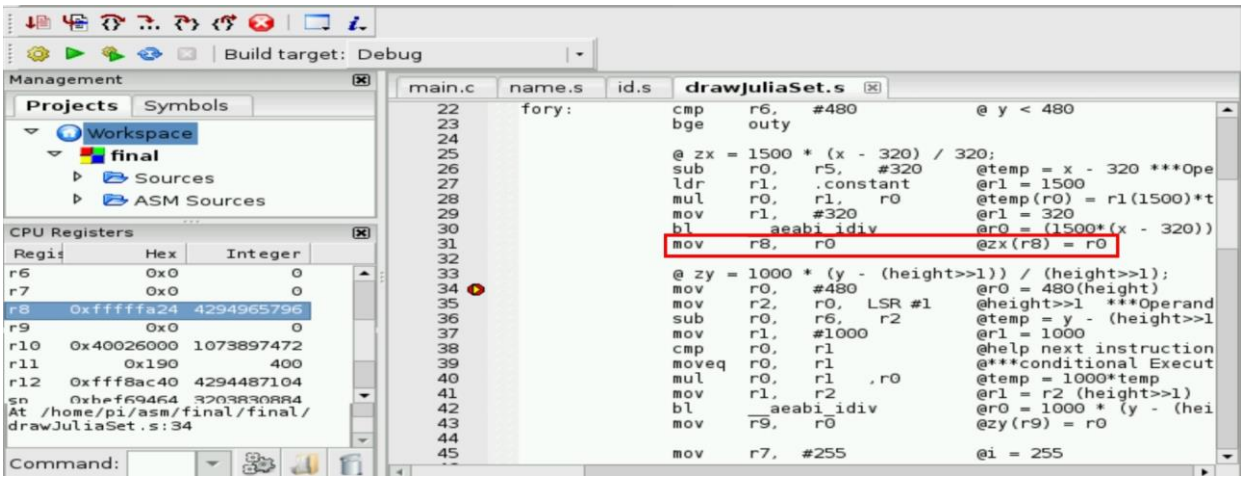
這是將 i 存到 r7，且初始值為 255



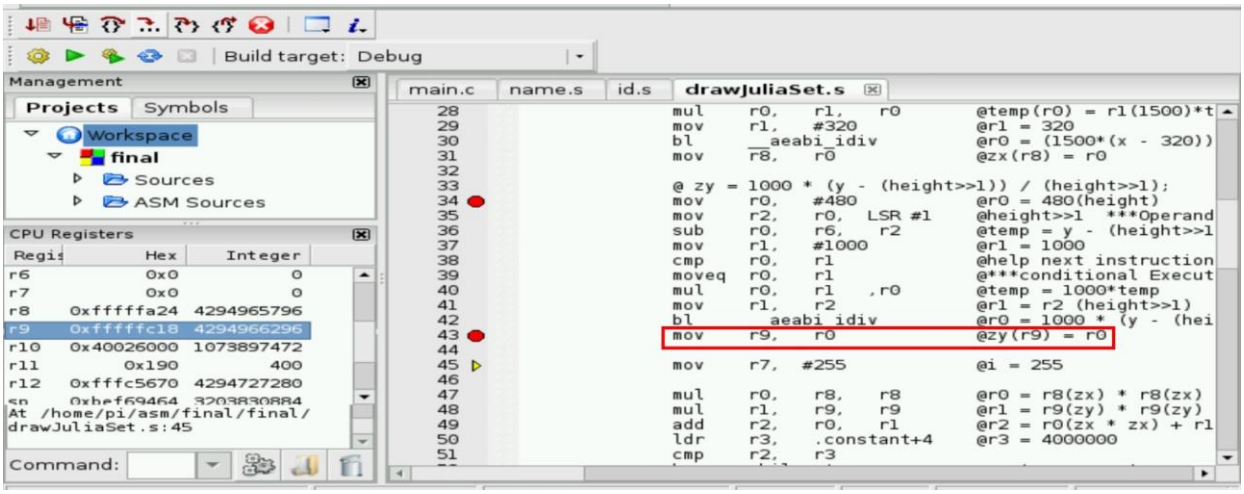
這是將 color 存到 r7



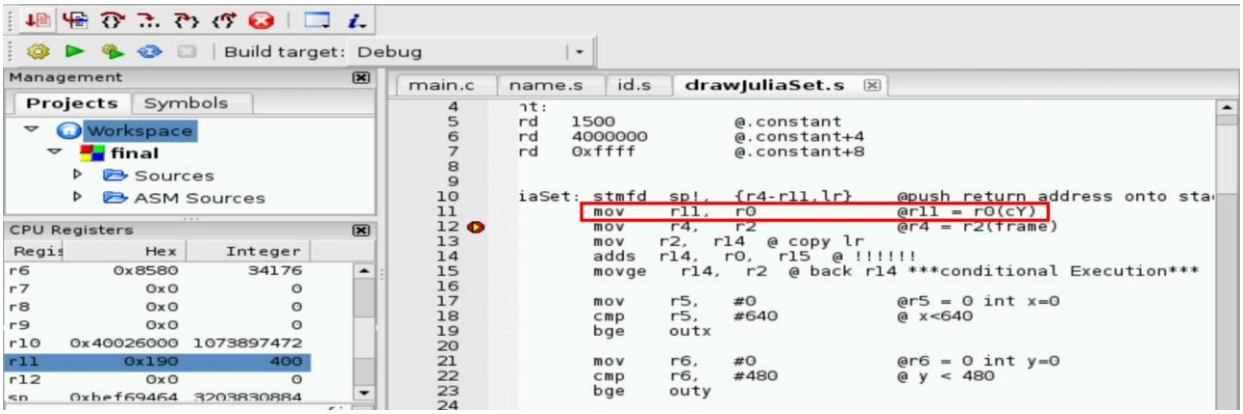
這是將 zx 存到 r8



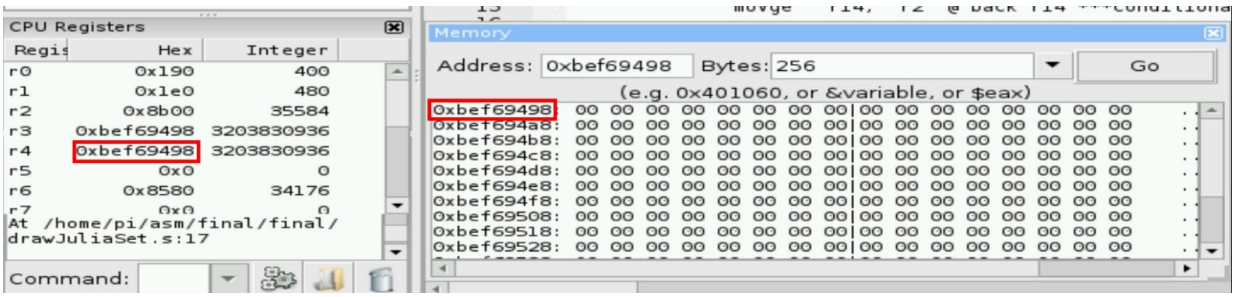
這是將 zy 存到 r9



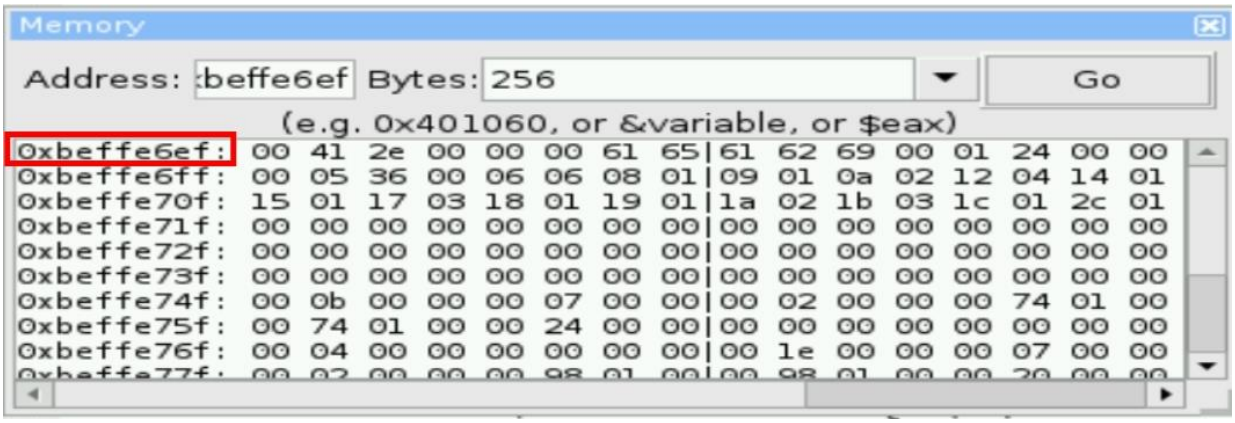
這是將 cY 存到 r11



Frame 起始位置



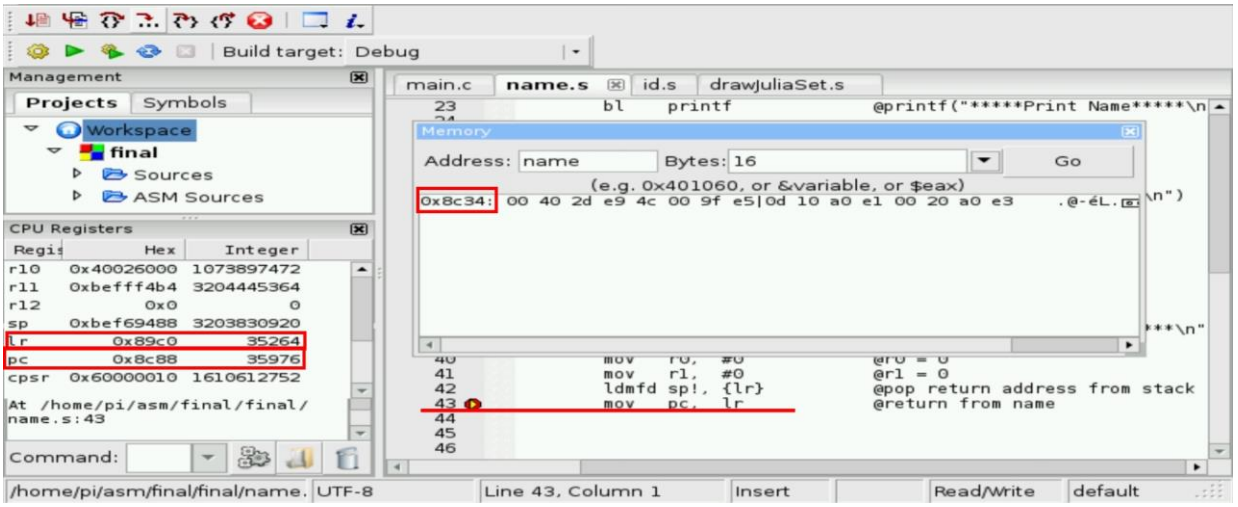
Frame 結束位置



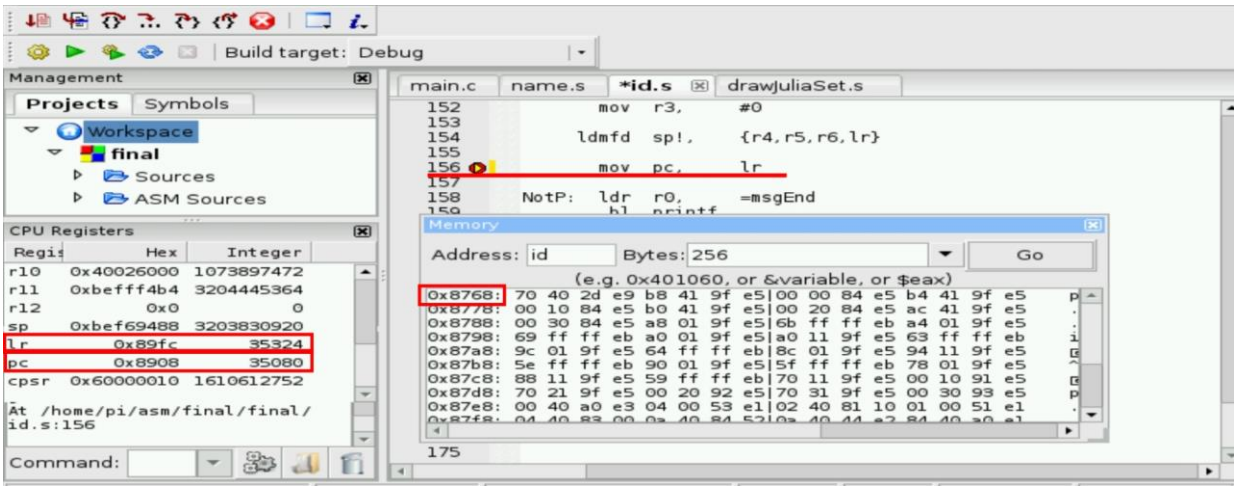
Variable	value	Start address	End address
frame		0xbef69498	0xbeffe6ef

(4) main:

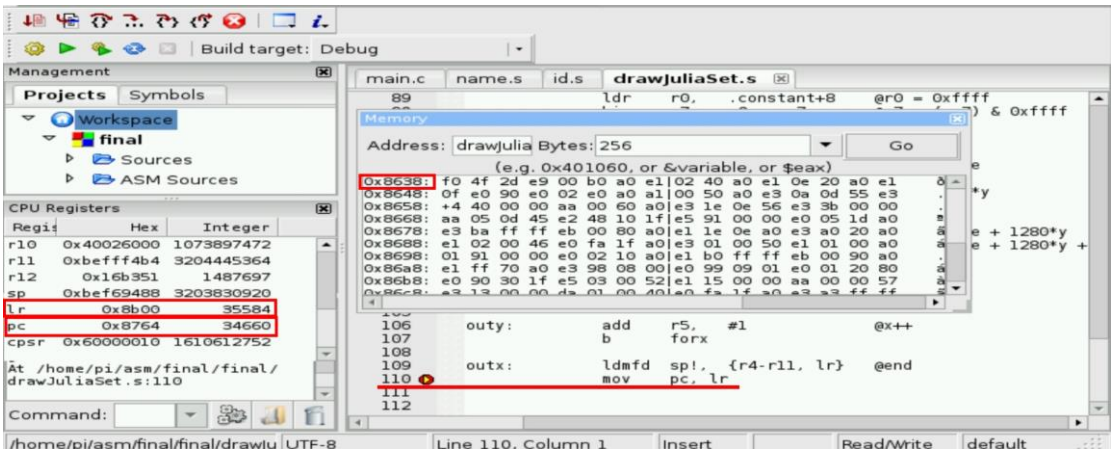
name 的起始位置、結束位置、回傳位置



id 的起始位置、結束位置、回傳位置



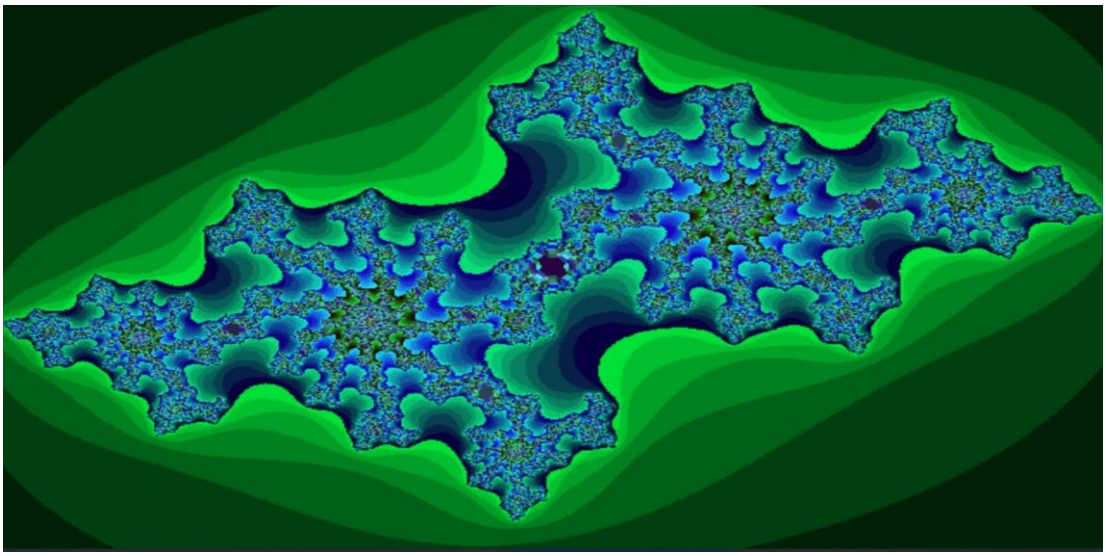
drawJuliaSet 的起始位置、結束位置、回傳位置

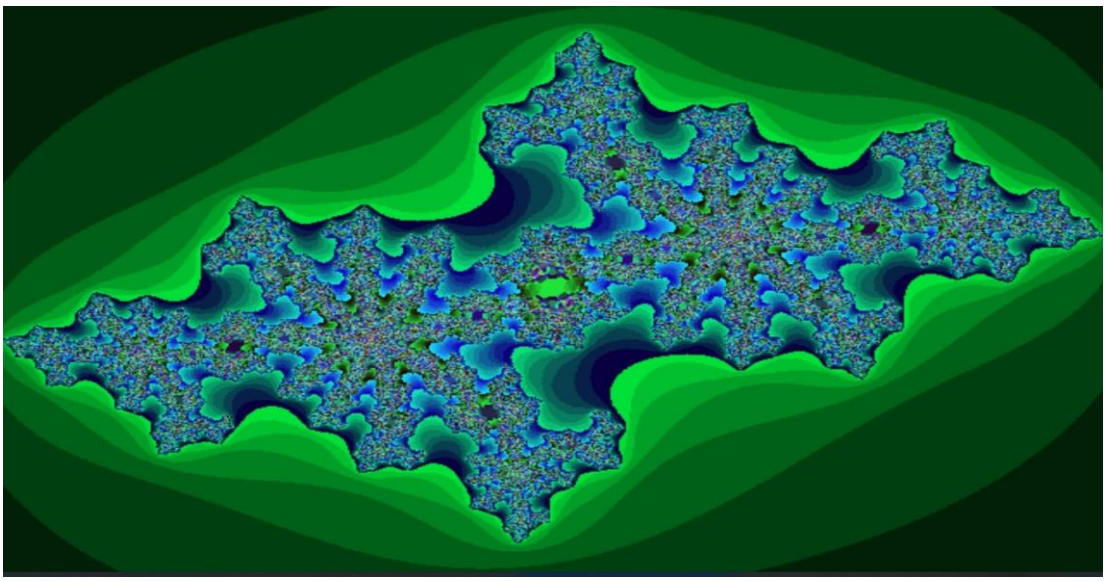


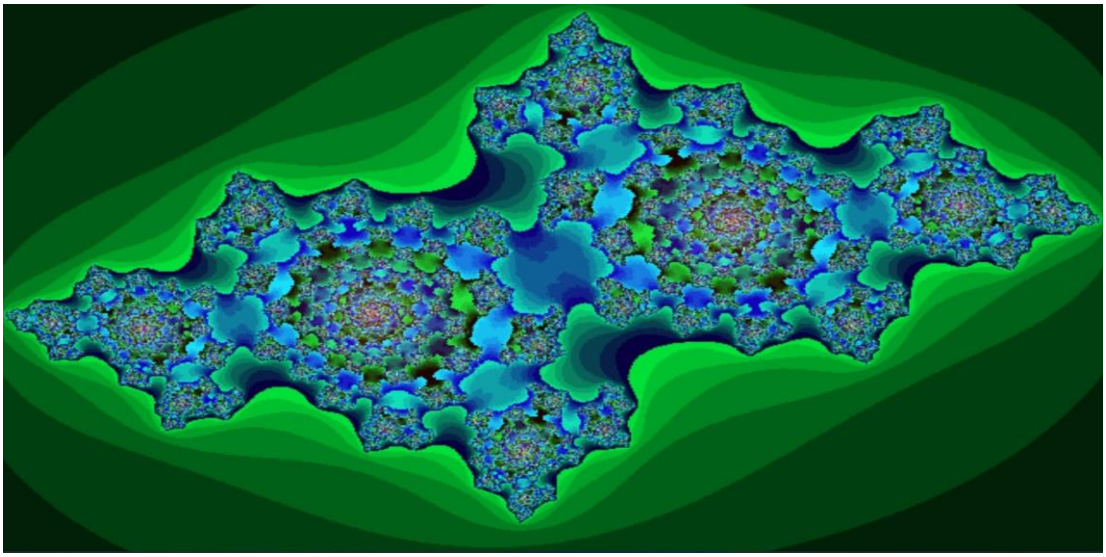
Function	value	Start address	End address	Return address
name	name function 的內容	0x8c34	0x8c88	0x89c0
id	id function 的內容	0x8768	0x8908	0x89fc
drawJuliaSet	drawJuliaSet 的內容	0x8638	0x8764	0x8b00

(5) 最後完整結果:

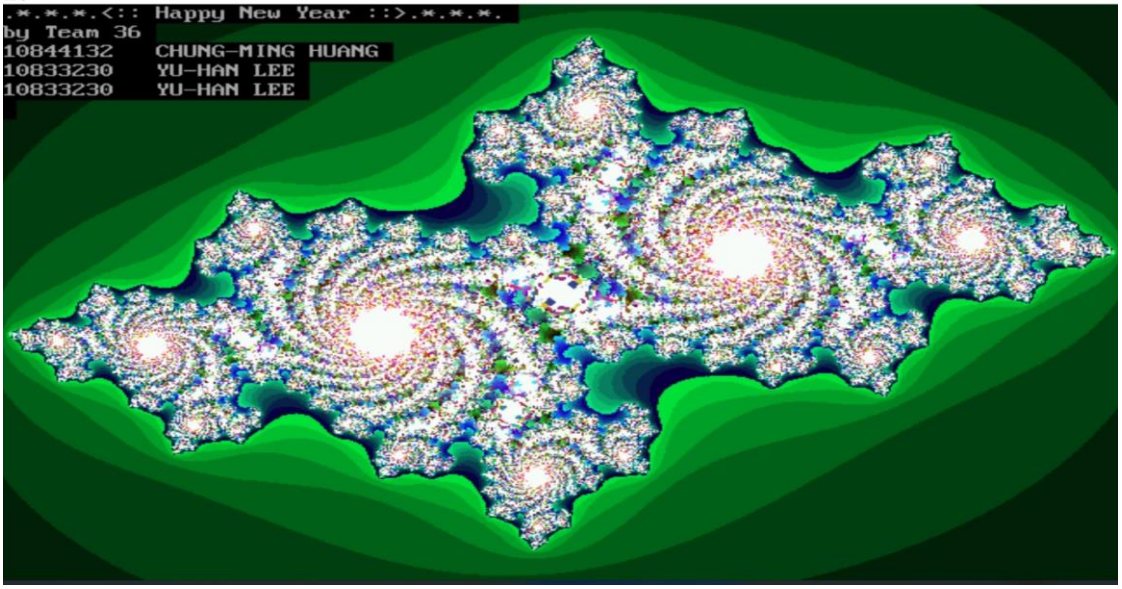
```
******Input ID******
**Please Enter Member1 ID:**
10844132
**Please Enter Member2 ID:**
10833230
**Please Enter Member3 ID:**
10833230
**Please Enter Command **
p
******Print Team Member ID and ID Summation******
10844132
10833230
10833230


ID Summation = 32510592
******End Print******


Main Function:
******Print All******
Team 36
10844132    CHUNG-MING HUANG
10833230    YU-HAN LEE
10833230    YU-HAN LEE
ID Summation = 32510592
******End Print******

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







下圖為最後結果


.*.*.*.<:: Happy New Year ::>.*.*.*.
by Team 36
10844132 CHUNG-MING HUANG
10833230 YU-HAN LEE
10833230 YU-HAN LEE

四、討論

各組員分工方式與負責項目:

10833230 李郁含

drawJuliaSet 撰寫、製作組合語言與嵌入式系統成果報告、Bug 測試、寫註解、處理 Project 基本要求

10844132 黃中憫

drawJuliaSet 撰寫、製作組合語言與嵌入式系統成果報告、Bug 測試、寫註解、處理 Project 基本要求

五、結論 六、未來展望

10833230 李郁含

一開始想說先將老師給的 project2 檔試試看先看成果長甚麼樣子，但讓 project2 檔解壓縮就遇到問題了，本來用 7z 上網找了很多指令都沒用，後來先將老師的解壓縮再壓縮成 zip 檔最後也是上網找到在終端機解壓縮的指令就解決了。

由於老師給的 drawJuliaSet 的變數太多而我們寫到後面發現暫存器不夠用，因此只好減少變數的傳遞，寫完我們經過修改的老師的 drawJuliaSet 後也 DEBUG 完後我們跑了一次，發現整個圖跟老師的完全不一樣，我們非常慌張，繼續 DEBUG 一個一個找，想找出問題的所在，後來發現似乎是暫存器的問題我們改善了我們對於暫存器的規劃後圖案就變回來了。

在這一次期末的 FINAL PROJECT 實在跟 MIDTERM 差太多了，花了我們不少時間問題也比 MIDTERM 多太多，不過學到了很多關於組合語言裡暫存器的使用，經過這次的期末 PROJECT，認為實在給予我們很大的受用，我們更加了解了電腦內部的

運作和記憶體使用的部分，我們相信經過這堂課，有了這些經驗將會對我們未來的程式之路更加順利。

10844132 黃中憫

這次的 project 中，我們碰到的問題數量比期中 project 時還要多很多，像一開始 id 的值在 main 裡面無法讀到，因為我們沒有弄清楚 C 語言傳遞參數的方法，然後在寫 drawJuliaSet 時，一開始因為沒有簡化導致參數太多，暫存器不夠用，這讓我們兩個寫得很痛苦，因為用 sp 時不時就跳出問題，加上裡面又有很多迴圈…還有除法的用法也卡我們很久，因為上 google 找的資料都不大符合我們要的，有些甚至是錯誤的，這讓我們的進度常常被延遲。不過還好這些問題在最後一一被我們解決，有些是從講義上找到解決方法，有些是從 google 上面找到作法，有些則是我們自己想出的解法。雖然寫這個 project 時很痛苦，

時常因為找不到問題而在那邊不斷的 loop，一遍又一遍的檢查，但又一直找不到 bug 在哪裡…但是中間還是學到了很多。我期望自己在未來能提升找 bug 的能力，這樣以後在寫這種大型的專案或題目時可以更好的找出自己的錯誤所在並去修正它。