Final Project

110學年度第1學期

老師:朱守禮老師

組別:第2組

班別:資訊二甲

學號:10927104 / 10927109

姓名:張郁琪 / 陳宥蓁

一、程式說明

drawJuliaSet.s:

```
.text
  .global drawJuliaSet

  drawJuliaSet :
    stmfd sp!, {r4-r11, lr} @ push r4-r11, lr onto stack

    mov r6, r0                @ cY
    mov r7, r1                @ frame
    mov r4, #0                @ r4 = 0 @ x
    moveq r5, #0              @ if(eq) r5 = 0 @ y  @@@@ Conditional Execution

    adds r14, r0, r15

loop1:
    cmp r4, #640              @@ x < width
    bge return_drawJuliaSet
    mov r5, #0                @ r5 = 0 @@ y = 0

loop2:
    cmp r5, #480              @@ y < height
    addge r4, r4, #1          @ if( r5 >= 480 )@ x++  @@@@ Conditional Execution
    bge loop1

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

    ldr r0, .constant         @ r0 = 1500
    mov r2, #640              @ r2 = 640  @ width = 640
    mov r1, r2, asr #1        @ r1 = width >> 1    @@@@  Operand2
    sub r2, r4, r1            @ r2 = x - (width>>1)
    mul r0, r0, r2            @ r0 = 1500 * (x - (width>>1))
    bl __aeabi_idiv           @ r0 = 1500 * (x - (width>>1)) / (width>>1)
    mov r8, r0                @ zx @ r8 = r0 = 1500 * (x - (width>>1)) / (width>>1)

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

    mov r0, #1000             @ r0 = 1000
    mov r2, #480              @ r2 = 480  @ height = 480
    mov r1, r2, asr #1        @ r1 = height >> 1
    sub r2, r5, r1            @ r2 = y - (height>>1)
    mul r0, r0, r2            @ r0 = 1000 * (y - (height>>1))
    bl __aeabi_idiv           @ r0 = 1000 * (y - (height>>1)) / (height>>1)
    mov r9, r0                @ zy  @ r9 = r0 = 1000 * (y - (height>>1)) / (height>>1)

 @---------------------- i = 255 --------------------------

    mov r10, #255         @ r10 = 255 @ r10 is i

 @---------zx * zx + zy * zy < 4000000 && i > 0-------------

    mul r0, r8, r8            @ r0 = zx * zx
    mul r1, r9, r9            @ r1 = zy * zy
    add r0, r0, r1            @ r0 = zx * zx + zy * zy
    ldr r1, .constant+4       @ r1 = 4000000
    cmp r0, r1
    bge temp                  @ ( r0 >= r1 ) @@ zx * zx + zy * zy > 4000000
    cmplt r10, #0             @ ( r0 < r1 ) @@@@ Conditional Execution
    ble temp                  @ ( r10 <= 0 )

loop3:
    mul r0, r8, r8            @ r0 = zx * zx
    mul r1, r9, r9            @ r1 = zy * zy
    sub r0, r0, r1            @ r0 = zx * zx - zy * zy                  @@@@   Operand2
    mov r1, #1000             @ r1 = 1000
    bl __aeabi_idiv           @ r0 = (zx * zx - zy * zy)/1000
    sub r11, r0, #700         @ r11 = (zx * zx - zy * zy)/1000 + cX     @@@@   Operand2

    mul r0, r8, r9            @ r0 = zx * zy
    mov r1, #2                @ r1 = 2
    mul r0, r0, r1            @ r0 = (2 * zx * zy)
    mov r1, #1000             @ r1 = 1000
    bl __aeabi_idiv           @ r0 = (2 * zx * zy)/1000
    add r0, r0, r6            @ r0 = (2 * zx * zy)/1000 + cY
    mov r9, r0                @ r9 = r0  @ zy = (2 * zx * zy)/1000 + cY

    mov r8, r11               @ zx = tmp

    sub r10, r10, #1          @ i--

    mul r0, r8, r8            @ r0 = zx * zx
    mul r1, r9, r9            @ r1 = zy * zy
    add r0, r0, r1            @ r0 = zx * zx + zy * zy
    ldr r1, .constant+4       @ r1 = 4000000
    cmp r0, r1                @ zx * zx + zy * zy < 4000000
    bge temp                  @ ( r0 >= r1 ) @@ zx * zx + zy * zy > 4000000
    cmplt r10, #0             @ ( r0 < r1 ) @@@@ Conditional Execution
    ble temp                  @ ( r10 <= 0 )
    b loop3

temp:
    and r0, r10, #0xff        @ r0 = ( i & 0xff )
    orr r1, r0, r0, lsl #8    @ r1 = ( i & 0xff ) | ( ( i & 0xff ) << 8 )  @@@@   Operand2
    ldr r0, .constant+8       @ r0 = 0xffff
    bic r3, r0, r1            @ r3 = 0xffff & (~color)

    mov r0, r7                @ frame
    mov r1, #1280             @ r1 = 1280 = 2 * 640
    mul r1, r1, r5            @ r1 = 1280 * y
    add r0, r0, r1            @ r0 = frame + 1280y
    add r0, r0, r4, lsl #1    @ r0 = frame + 1280y + 2x
    strh r3, [r0]             @ 2 byte @@ frame[y][x] = color
    add r5, r5, #1            @ y++
    b loop2

return_drawJuliaSet:
    mov r0, #0                @ move return code into r0
    ldmfd sp!, {r4-r11, lr}   @ pop r4-r11, lr from stack
    mov pc, lr

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

先備份需要用到的暫存裡到sp裡，r6 = r0 (第一個參數)、r7 = r1 (第二個參數)、r4 = 0 (初始化 x = 0)、if(eq) r5 (初始化 y = 0) = 0，r14 = r0 + r15 且設定cpsr (adds r14, r0, r15)，『 loop1 (外層for迴圈) 』將 r4 與 640比較大小 ，如果 r4 >= 640 ，就分支跳躍到return_drawJuliaSet這個label，如果 r4 < 640 ，就將 r5 = 0 (y = 0)，開始做『 loop2 (內層for迴圈) 』，將 r5 與 480比較大小 ，如果 r5 >= 480 ，先r4 = r4 + 1 後，分支跳躍到loop1這個label，如果 r5 < 480 ，就 ldr r0, .constant (從constant pool 的起始位置撈常數1500出來) ，r2 = 640 ， r1 = r2算術右移一位元，r2 = r4 - r1，r0 = r0 * r2 ，分支跳躍到__aeabi_idiv這個label做除法運算(除完結果會存在r0) ，r8 = r0 ，r0 = 1000， r2 = 480， r1 = r2算術右移一位元，r2 = r5 - r1，r0 = r0 * r2 ，分支跳躍到__aeabi_idiv這個label做除法運算(除完結果會存在r0)，r9 = r0，r10 = 255，r0 = r8 * r8，r1 = r9 * r9，r0 = r0 + r1， ldr r1, .constan+4 (從constant pool 的起始位置+4的位置抓常數4000000出來) ，將 r0 與r1比較大小，如果r0 >= r1 ，分支跳躍到temp這個label，如果r0 < r1，將r10 與 0 比較大小，如果r10 <= 0 就分支跳躍到temp這個label ，如果r10 > 0 ，r0 = r8 * r8，r1 = r9 * r9 ，r0 = r0 - r1 ，r1 = 1000分支跳躍到__aeabi_idiv這個label做除法運算(除完結果會存在r0)，r11 = r0 - 700，r0 = r8 * r9，r1 = 2，r0 = r0 * r1，r1 = 1000，分支跳躍到__aeabi_idiv這個label做除法運算(除完結果會存在r0)，r0 = r0 + r6，r9 = r0，r8 = r11，r0 = r8 * r8，r1 = r9 * r9，r0 = r0 + r1， ldr r1, .constan+4 (從constant pool 的起始位置+4的位置抓常數4000000出來) ，將 r0 與r1比較大小，如果r0 >= r1 ，分支跳躍到temp這個label，如果r0 < r1，將r10 與 0 比較大小，如果r10 <= 0 就分支跳躍到temp這個label ，否則就分支跳躍到loop3這個label。

- temp這個label是先將r10與0xff AND完的結果存到r0， r0 OR r0邏輯左移8位元的結果存到r1，ldr r1, .constan+8 (從constant pool 的起始位置+8的位置抓0xffff出來)，r0 AND (取r1的NOT)的結果存到r3，r0 = r7，r1 = 1280 (2 * 640)，r1 = r1 * r5，r0 = r0 + r1，r0 = r0 + r4邏輯左移1位元，將r3的值取2byte存到r0這個記憶體位置，r5 = r5 + 1，分支跳躍到loop2這個label。

- return_drawJuliaSet這個label先將r0 = 0(move return code into r0)，還原之前備份的暫存器，mov pc, lr，結束。

```
.data
  msg1:  .asciz  "*****Print Name*****\n"
  team:  .asciz "Team 02\n"
  name1:  .asciz "Zhang, Yu-Qi\n"
  name2:  .asciz "Chen, Yu-Zhen\n"
  name3:  .asciz "Zhang, Yu-Qi\n"
  msg2:  .asciz  "*****End Print*****\n"

.text
  .globl  name

  name:
    stmfd  sp!,{lr}    @ push lr onto stack
    movvc  r0, r0,ASR #0 @ Arithmetic Shift Right
    ldr    r0, =msg1
    bl     printf      @ *****Print Name*****
    mov    r1, #0      @ r1 = 0
    addhss r2, r13, #0 @ r2 = r13 + 0 and set CPSR flags
    adcs   r13,r1,r2   @ r13 = r1+r2+c and set CPSR flags
    ldr    r0, =team
    bl     printf      @ Team 02
    ldr    r0, =name1
    bl     printf      @ Zhang, Yu-Qi
    ldr    r0, =name2
    bl     printf      @ Chen, Yu-Zhen
    ldr    r0, =name3
    bl     printf      @ Zhang, Yu-Qi
    ldr    r0, =msg2
    bl     printf      @ *****End Print*****
    mov    r0, #0      @ move return code into r0
    ldmfd  sp!, {lr}   @ pop lr from stack
    mov    pc, lr
```

name.s:
在data段先寫好組別、組員的英文名字、格式必要開頭及結尾，備份lr到sp，ldr r0, =msg1，分支跳躍到printf這個label印出*****Print Name*****，將r1 = 0，假如C set就讓r2 = r13 + 0 並set CPSR，r13 = r1 + r2 + c (adcs r13 ,r1,r2)並set CPSR，在依序將組別、組員的英文名字、*****End Print*****印出，return code into r0 (mov r0,#0)，還原lr，mov pc, lr，結束。

```
.data
 msg1:  .asciz "*****Input ID*****\n"
 msg2:  .asciz "** Please Enter Member 1 ID: **\n"
 msg3:  .asciz "** Please Enter Member 2 ID: **\n"
 msg4:  .asciz "** Please Enter Member 3 ID: **\n"
 msg5:  .asciz "** Please Enter Command **\n"
 msg6:  .asciz "*****Print Team Member ID and ID Summation*****\n"
 msg7:  .asciz "\n\nID Summation = %d\n"
 msg8: .asciz "*****End Print*****\n"
 Input: .asciz "%d"
 Output:  .asciz "%d\n"
 Command: .asciz "%s"
 p:  .asciz "p\0\0\0"

 id1:   .word 0
 id2:   .word 0
 id3:   .word 0
 sum:   .word 0
 Commandspace: .word 0

.text
 .globl  id

 NotP:
   ldr    r0, =msg8   @ *****End Print*****
   bl     printf
   ldmfd  sp!,{r7-r10, lr}   @ pop r7-r10, lr from stack
   mov    pc,  lr

 id:
   stmfd  sp!,{r7-r10, lr} @ push r7-r10, lr onto stack

   mov    r7, r0   @ r7 = r0
   mov    r8, r1   @ r8 = r1
```

```
   mov    r9 , r2  @ r9 = r2
   mov    r10, r3  @ r10 = r3

   mov    r3 , #0
   ldr    r0, =msg1
   bl     printf   @ *****Input ID*****

   ldr    r0, =msg2
   bl     printf   @ ** Please Enter Member 1 ID: **
   ldr    r0, =Input
   ldr    r1, =id1
   bl     scanf

   ldr    r0, =msg3
   bl     printf   @ ** Please Enter Member 2 ID: **
   ldr    r0, =Input
   ldr    r1, =id2
   bl     scanf

   ldr    r0, =msg4
   bl     printf   @ ** Please Enter Member 3 ID: **
   ldr    r0, =Input
   ldr    r1, =id3
   bl     scanf

   @@@@@@@@@@@@@@@@@@

   ldr    r1, =id1      @ load address of id1
   ldr    r1, [r1]      @ load value of id1 in r1
   ldr    r2, =id2      @ load address of id2
   ldr    r2, [r2]      @ load value of id2 in r2
   addpl  r1,r1,r2      @ if( N clear ) r1 = r1 + r2
   ldr    r2, =id3      @ load address of id3
   ldr    r2, [r2]      @ load value of id3 in r2
```

```
   addpl  r1,r1,r2      @ if( N clear ) r1 = r1 + r2
   ldr    r2, =sum      @ load address of sum
   str    r1, [r2]      @ store in r1 at the r2 address

   @@@@@@@@@@@@@@@@@@

   ldr    r1, =id1      @ load address of id1
   ldr    r1, [r1]      @ load value of id1 in r1
   str    r1, [r7]      @ store in r1 at the r7 address

   ldr    r1, =id2      @ load address of id2
   ldr    r1, [r1]      @ load value of id2 in r1
   str    r1, [r8]      @ store in r1 at the r8 address

   ldr    r1, =id3      @ load address of id3
   ldr    r1, [r1]      @ load value of id3 in r1
   str    r1, [r9]      @ store in r1 at the r9 address

   ldr    r1, =sum      @ load address of sum
   ldr    r1, [r1]      @ load value of sum in r1
   str    r1, [r10]     @ store in r1 at the r10 address
```

```
   ldr    r0, =msg5
   bl     printf   @ ** Please Enter Command **

   ldr    r0, =Command
   ldr    r1, =Commandspace
   bl     scanf

   ldr    r0, =p   @ load address of p
   ldr    r0, [r0] @ load value of p in r0
   ldr    r1, =Commandspace  @ load address of Commandspace
```

```
   ldr    r1, [r1] @ load value of Commandspace in r1
   cmp    r0, r1  @ compare them
   blne   NotP    @ if( Commandspace != "p" ) branch to NotP

   @@@@@@@@@@@@@@@@@@

   ldreq  r0, =msg6
   bl     printf  @*****Print Team Member ID and ID Summation*****

   ldr    r0, =Output
   ldr    r1, =id1
   ldr    r1, [r1]
   bl     printf
```

id.s：

在data段先寫好格式要求後，備份r7,r8,r9,r10,lr到sp，r7 = r0, r8 = r1, r9 = r2, r10 = r3, r3, #0, ldr r0, =msg1，分支跳躍到printf這個label印出*****Input ID*****，ldr r0, =msg2，分支跳躍到printf這個label印出*****Please Enter Member 1 ID ******，分支跳躍到scanf這個label將學號輸入到規劃的記憶體位址(另外兩個學號也利用同樣的方式)後，將三個學號值相加成一個數值後，存入第4個規劃的記憶體位址(ldr r2,=sum ; str r1,[r2])，將三個學號及總和分別存到r7,r8,r9,r10(ldr r1,=id1 ;ldr r1, [r1] ; str r1,[r7]，另外兩個學號及總和也利用同樣的方式),ldr r0, =msg5，分支跳躍到printf這個label印出** Please Enter Command **分支跳躍到scanf這個label輸入command，將p的位址給r0 (ldr r0, =p)，在把r0記憶體位址中的值給r0(ldr r0, [r0])，將Commandspace的位址給r1 (ldr r1,=Commandspace)，在把r1記憶體位址中的值給r1(ldr r1, [r1])，比較r0,r1的值後更新CPSR(cmp r0, r1)，如果Z clear就分支跳躍到NotP這個label (輸入的不是p就不印學號跟總和，只印*****End Print******，還原lr，mov pc, lr，結束。) 如果Z set，ldreq r0, =msg6，分支跳躍到printf這個label印出*****Print Team Member ID and ID Summation*****後，再將3個學號與總和輸出，印*****End Print*

```
        ldr     r0, =Output
        ldr     r1, =id2
        ldr     r1, [r1, #0]
        bl      printf

        ldr     r0, =Output
        ldr     r1, =id3
        mov     r3, #0
        ldr     r1, [r1, r3]
        bl      printf

        ldr     r0, =msg7    @ ID Summation = %d
        ldr     r1, =sum
        mov     r3, #0
        ldr     r1, [r1, r3, ror #0]
        bl      printf

        ldr     r0, =msg8
        bl      printf       @ *****End Print*****

        mov     r0,#0        @ move return code into r0
        ldmfd   sp!,{r7-r10, lr}    @ pop r7-r10, lr from stack
        mov     pc, lr
```

****，return code into r0 (mov r0,#0)，還原之前備份的r7,r8,r9,r10,lr, mov pc, lr結束。

```c
# include <stdio.h>
# include <stdlib.h>
# include <sys/types.h>
# include <sys/stat.h>
# include <sys/mman.h>
# include <fcntl.h>

void name();
void id(int* id1,int* id2,int* id3,int* sum);
void drawJuliaSet(int cY, int16_t (*frame)[640]);

int main(){
  int id1, id2, id3, sum;

  printf("Function 1 : Name\n");
  name();

  printf("\nFunction 2 : ID\n");
  id(&id1, &id2, &id3, &sum);

  printf("Main Function :\n");
  printf("*****Print All*****\n");
  printf("Team 02\n");
  printf("%d  Zhang, Yu-Qi\n", id1);
  printf("%d  Chen,  Yu-Zhen\n", id2);
  printf("%d  Zhang, Yu-Qi\n", id3);
  printf("ID Summation = %d\n", sum);
  printf("*****End Print*****\n");
  printf("\n***** Please enter p to draw Julia Set animation*****\n");
  while( getchar() != 'p' ){} ;
  system( "clear" );

  int16_t frame[480][640];
  int cY, fd = open("/dev/fb0", (O_RDWR | O_SYNC));

  if ( fd < 0 ) printf("Frame Buffer Device Open Error!!\n");
  else {
    for( cY = 400 ; cY >= 270 ; cY = cY - 5 ){
      drawJuliaSet( cY, frame );
      write( fd, frame, sizeof(int16_t)*640*480 );
      lseek( fd, 0, SEEK_SET );
    } // for(cY)

    printf(".*.*.*.<:: Happy New Year ::>.*.*.*.\n");
    printf("by Team 02\n");
    printf("%d  Zhang, Yu-Qi\n", id1);
    printf("%d  Chen,  Yu-Zhen\n", id2);
    printf("%d  Zhang, Yu-Qi\n", id3);
    close(fd);
  } // else

  while( getchar() != 'p' ){} ;
  return 0;
} // main()
```

main.c:

先include好需要用到的函式庫，宣告int id1, id2, id3, sum，印出"Function 1 : Name"，呼叫name這個function，印出"Function 2 : ID" 呼叫id這個function並將id1, id2, id3, sum的位址傳進去，之後印出"Main Function :"、"*****Print All*****" 依序印出組別、學號(id1,id2,id3)、組員名字、學號的總和(sum)，印出"*****End Print*****"、"***** Please enter p to draw Julia Set animation*****"，跑回圈直到輸入的字元是P為止,system("clear") ，宣告frame陣列(int16_t frame[480][640]) 、cY、fd = open("/dev/fb0", (O_RDWR | O_SYNC))，假如 fd < 0 ，印出"Frame Buffer Device Open Error!!"，否則，跑for迴圈且設 cY = 400 ,只要cY >= 270就呼叫drawJuliaSet 將cY的值, frame的位址傳進去，執行完drawJuliaSet後執行write(fd, frame, sizeof(int16_t)*640*480)再執行lseek(fd, 0, SEEK_SET)後 cY = cY - 5，判斷條件(cY >= 270) 直到 cY < 270 跳出迴圈，跑回圈直到輸入的字元是P為止，return 0 ，結束。

二、設計重點說明

- name 函數設計重點: 規劃 4 個記憶體空間存放組別及姓名，解決第 7 道指令會改到 sp 暫存器的麻煩(因為 adcs r13,r1,r2 是將 r13 = r1 + r2 + c 會導致記憶體區段錯誤，所以我們先將 r1 = 0，addhss r2 , r13, #0，這樣一來不會改到 r13 且可以將 c bit set 為 0，因為我們用 CPU registers 中的 CPSR 得知 了 c bit 在這之前是 1)，解決之後就是依序 load 先前規劃的記憶體位址，bl printf(call printf) 印出，完成 name 程式要求。

- id 函數設計重點: 規劃 4 個記憶體空間存放學號及學號總和， 將main呼叫id函數的各個參數位址給r7~r10(mov r7, r0, mov r8, r1, mov r9 , r2, mov r10, r3)，以輸入的方式(call scanf)，輸入三個組員的學號於先前規劃的記憶體位址，輸入完後將 3 組輸入學號加總(add)且把值放於先前規劃的記憶體位址後，個別 load 先前規劃的記憶體位址，個別將該位址的值 store 到r7~r10(這樣main才可使用)，以輸入的方式(call scanf)輸入 command，利用 cmp 比較輸入的 command 是否為 p，是的話就分行印出學號、總和與結束 Print，不是的話就印結束 Print，完成 id 程式要求。

- drawJuliaSet 函數設計重點: 主要是計算並決定 Frame 二維陣列裡每個元素的值，以此來決定該元素投影至畫面上的顏色，我們使用多個暫存器，因為會傳cY、frame這兩個參數進去，一開始這兩個參數分別會在r0、r1，但由於之後會需要用到r0與r1，所以利用mov r6, r0(r6 = r0)，mov r7, r1(r7 = r1)解決，另外r4 是 x，r5 是 y 。

✧ 作業要求指令: adds r14, r0, r15(因為我們一開始有先備份原本的lr，且後來有還原，所以直接改到r14不會有什麼問題。)

✧ 條件執行：

1. mov r1, r2, asr #1 (暫存器移位)
2. sub r11, r0, #700 (立即值)
3. sub r2, r5, r1 (暫存器)

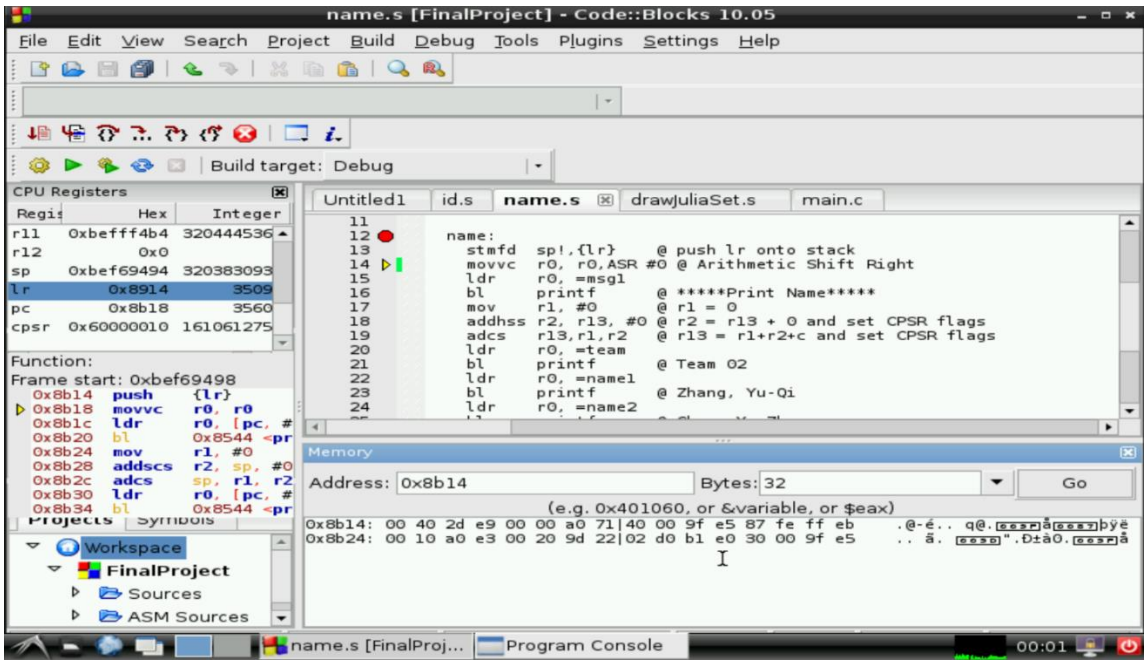
Operand2：

1. moveq r5, #0 (eq條件)
2. addge r4, r4, #1 (ge條件)
3. cmplt r10, #0 (lt條件)

三、結果截圖

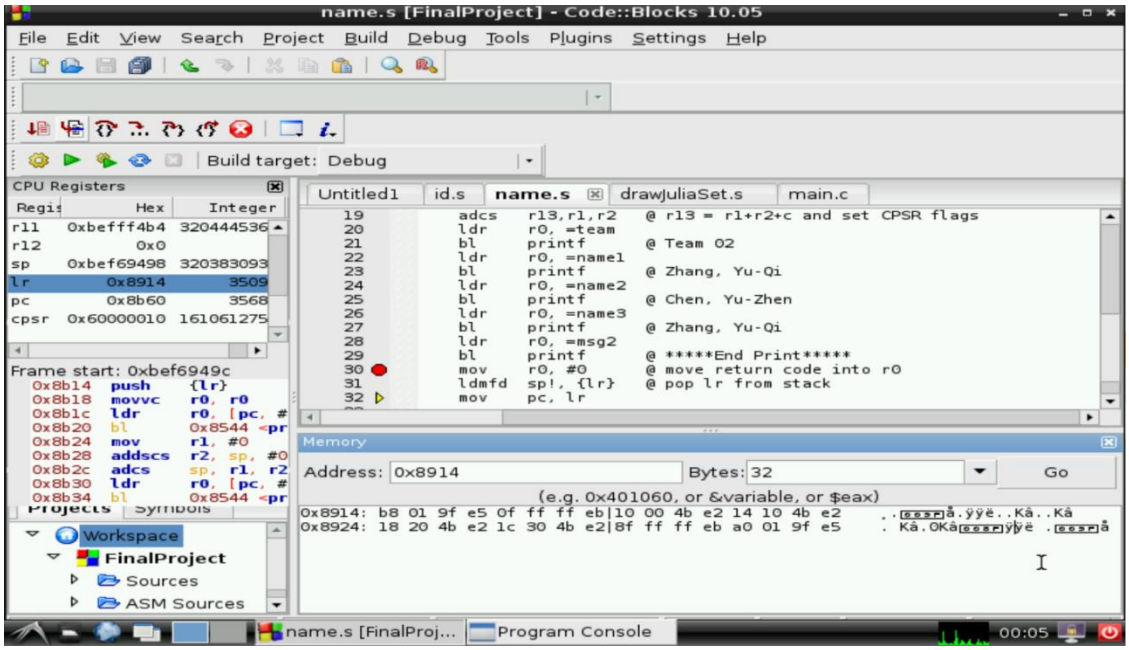
name函數的所在記憶體位址:0x8b14

所在記憶體位址內容:是name第一道指令



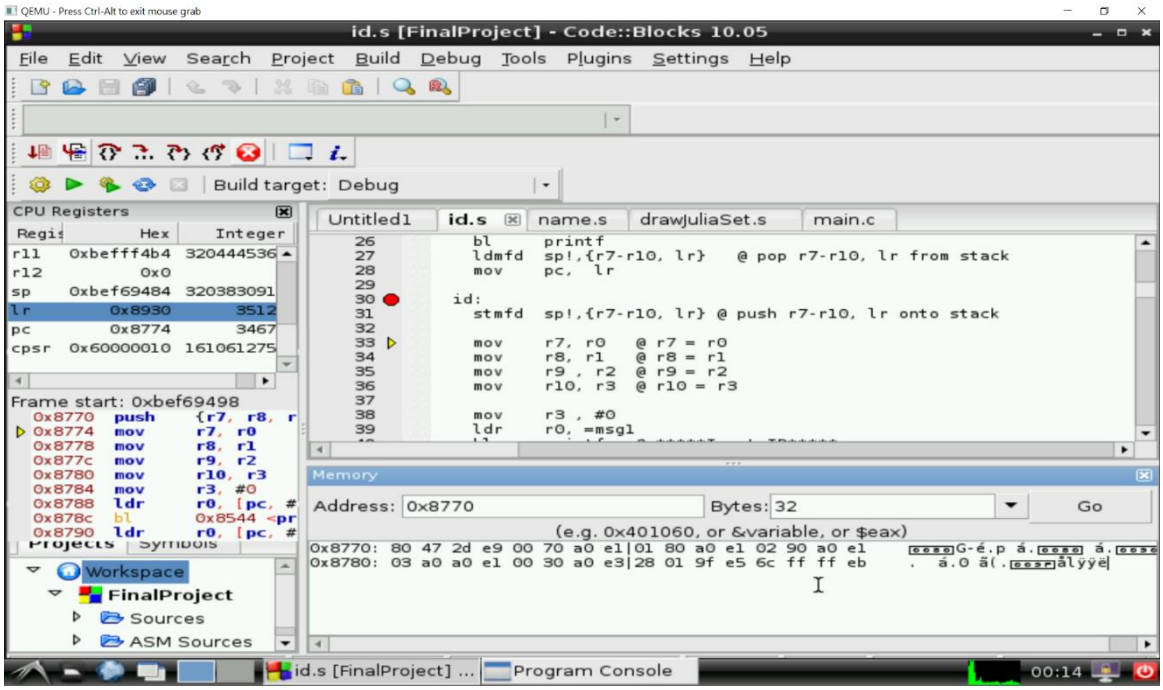
name函數的返回記憶體位址:0x8914

返回記憶體位址內容:是在main.c 呼叫name的下一道指令



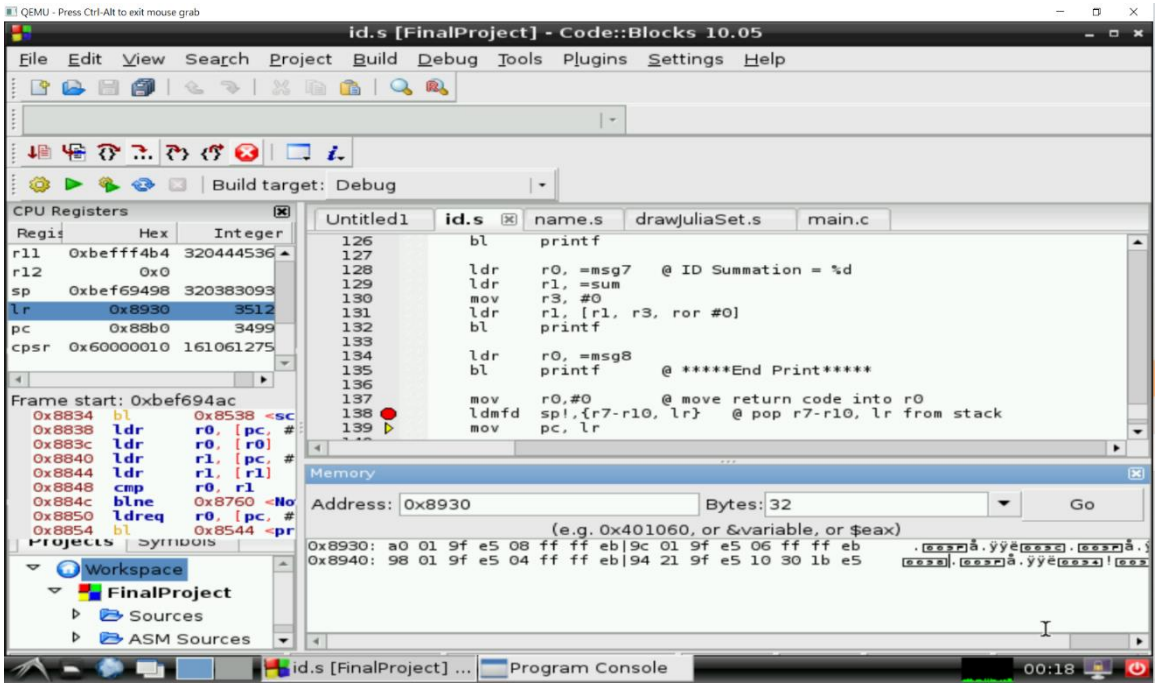
id函數的所在記憶體位址:0x8770

所在記憶體位址內容:是id第一道指令



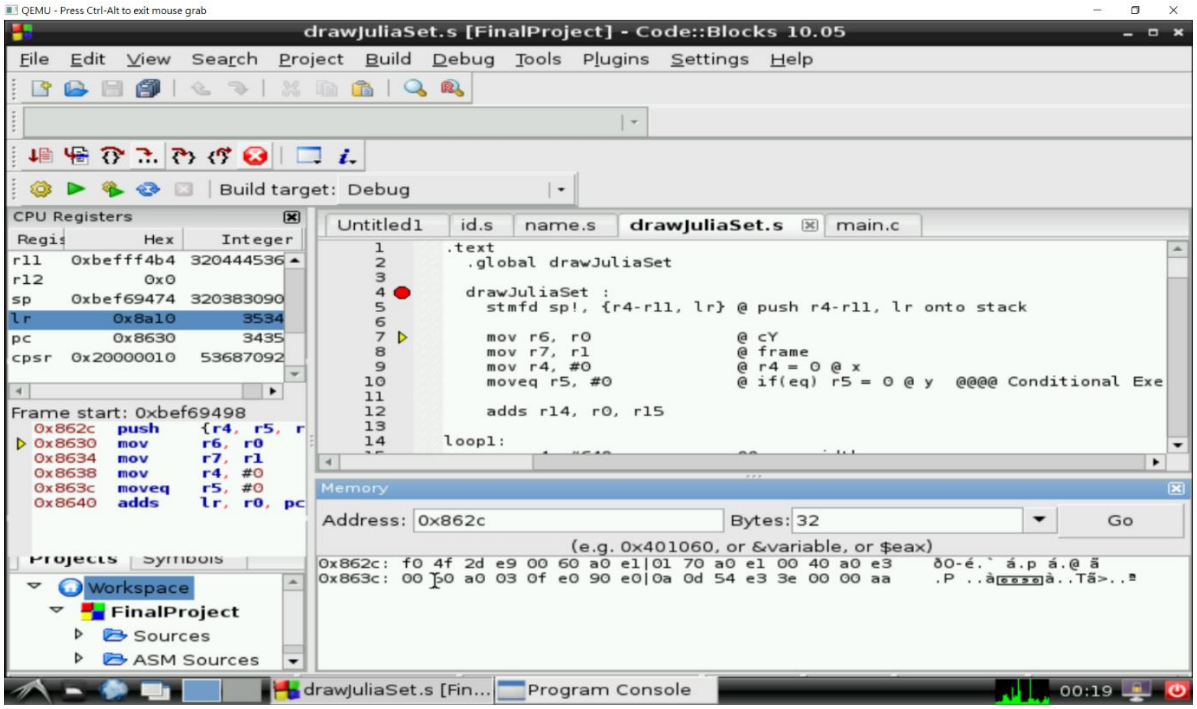
id函數的返回記憶體位址:0x8930

返回記憶體位址內容:是在main.c 呼叫id的下一道指令



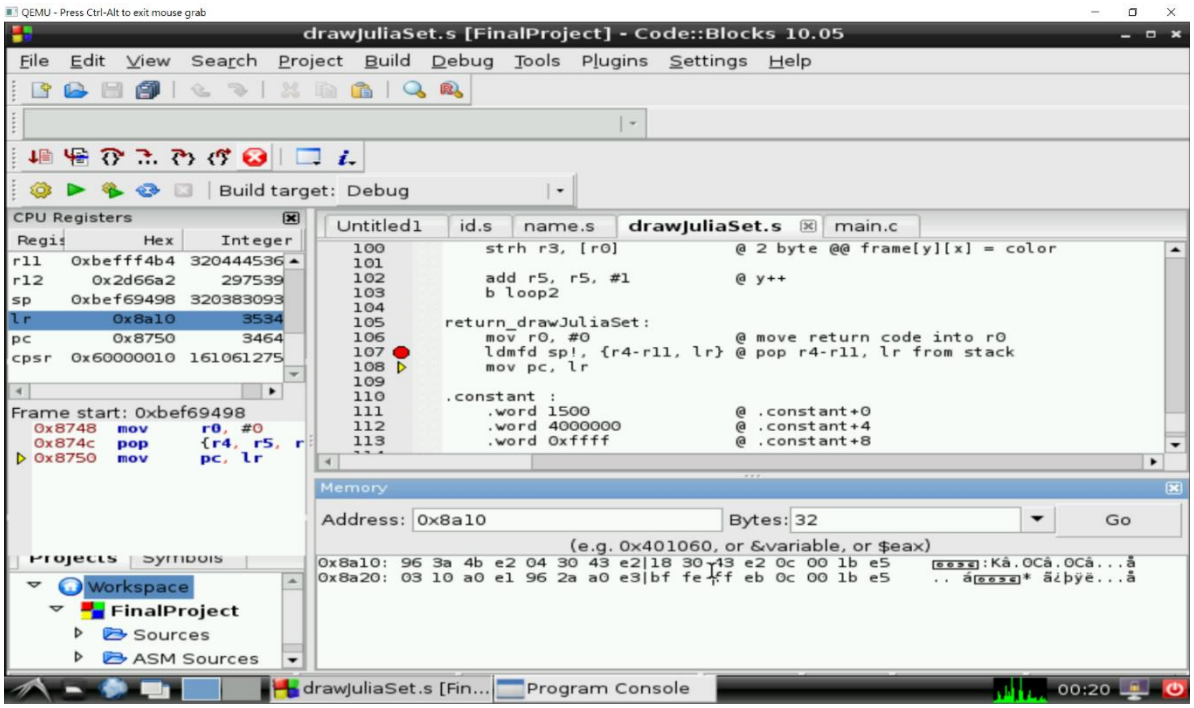
drawJuliaSet函數的所在記憶體位址:0x862c

所在記憶體位址內容:是drawJuliaSet第一道指令



drawJuliaSet函數的返回記憶體位址:0x8a10

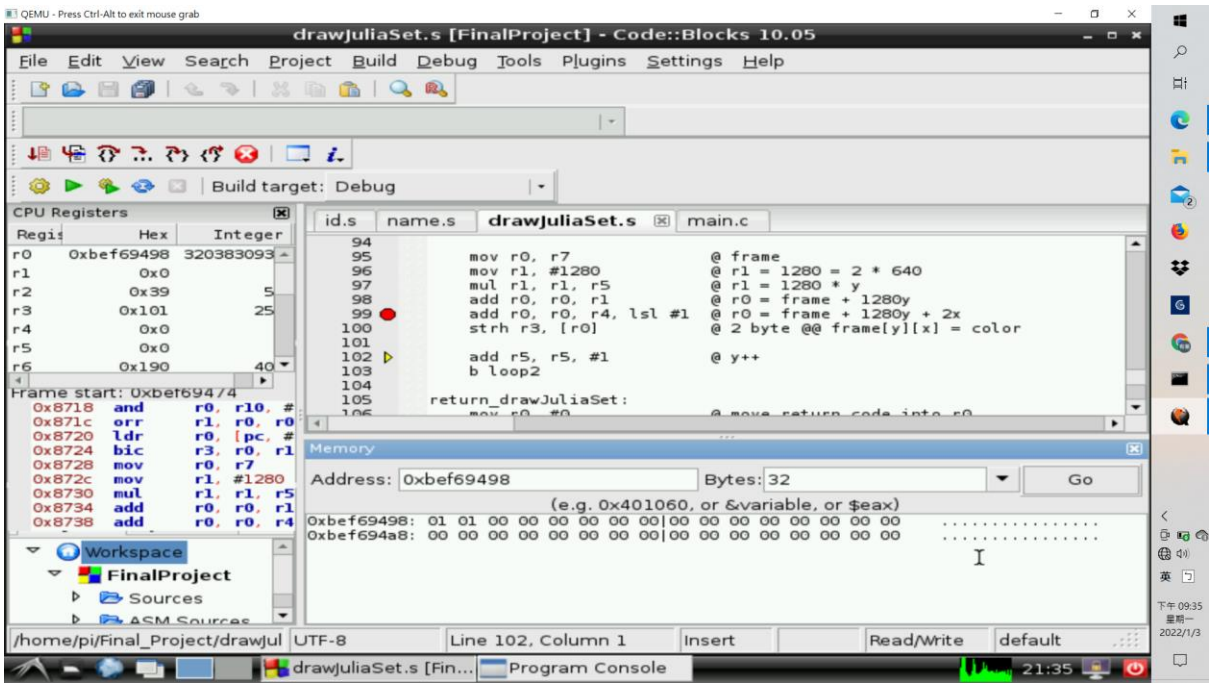
返回記憶體位址內容:是在main.c 呼叫drawJuliaSet的下一道指令



frame陣列開始記憶體位址:0xbef69498

(因為一開始我們將frame的初始位址給r7，所以在95行我們再將r0 = r7，且由於剛開始 x = 0, y = 0，所以 r0還是frame的初始位址。)

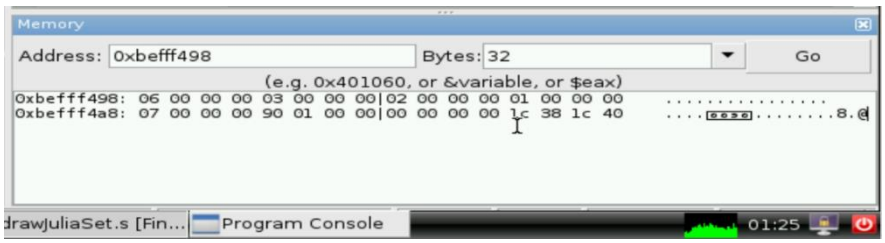
frame陣列開始的記憶體區塊內容: 計算好的color



frame陣列結束記憶體位址:0xbefff498

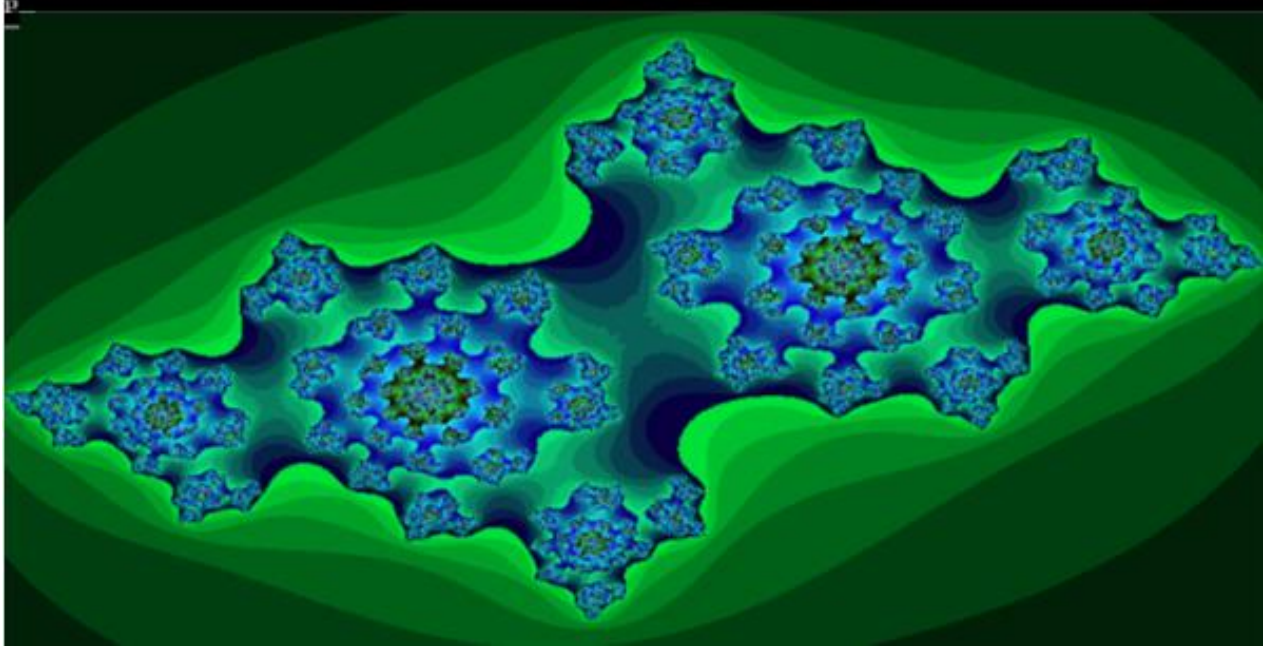
結束位置是利用起始位置的十進位(3203830936) + (1280*480=614400) = 3204445336

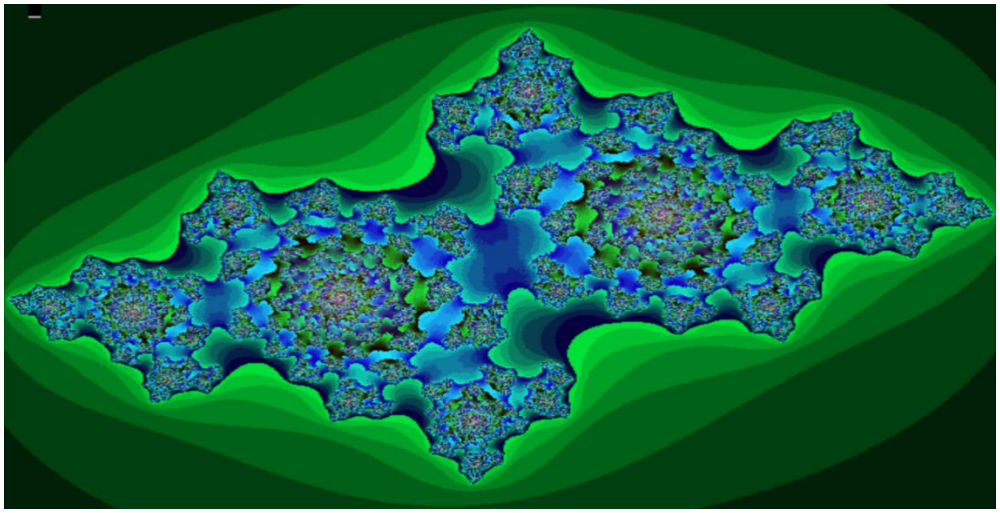
frame陣列結束的記憶體區塊內容:計算好的color

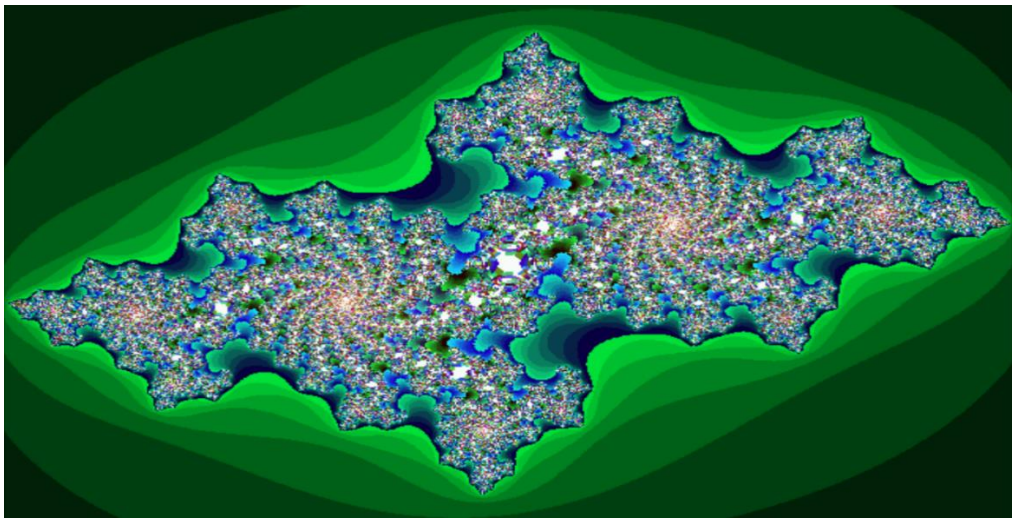


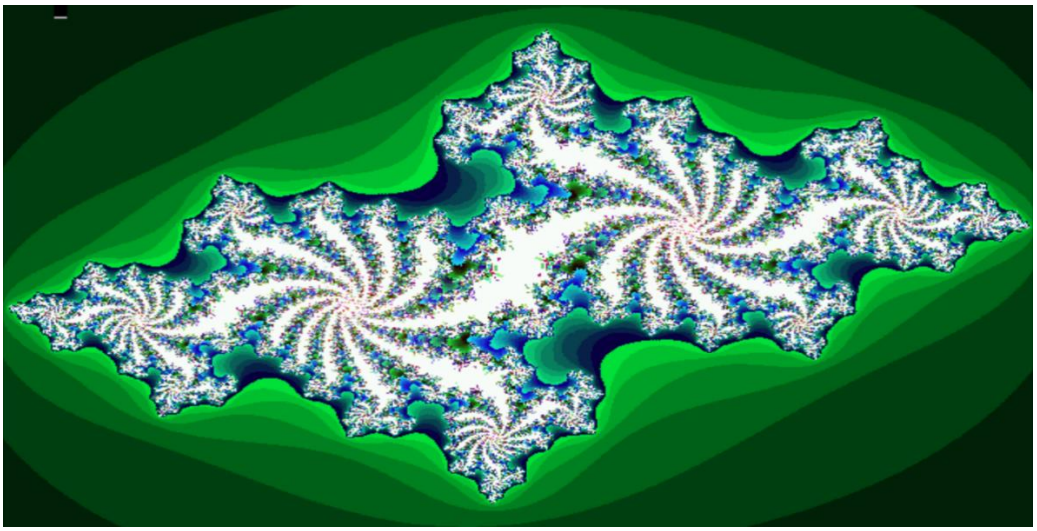
繪製 Julia Set 畫面：

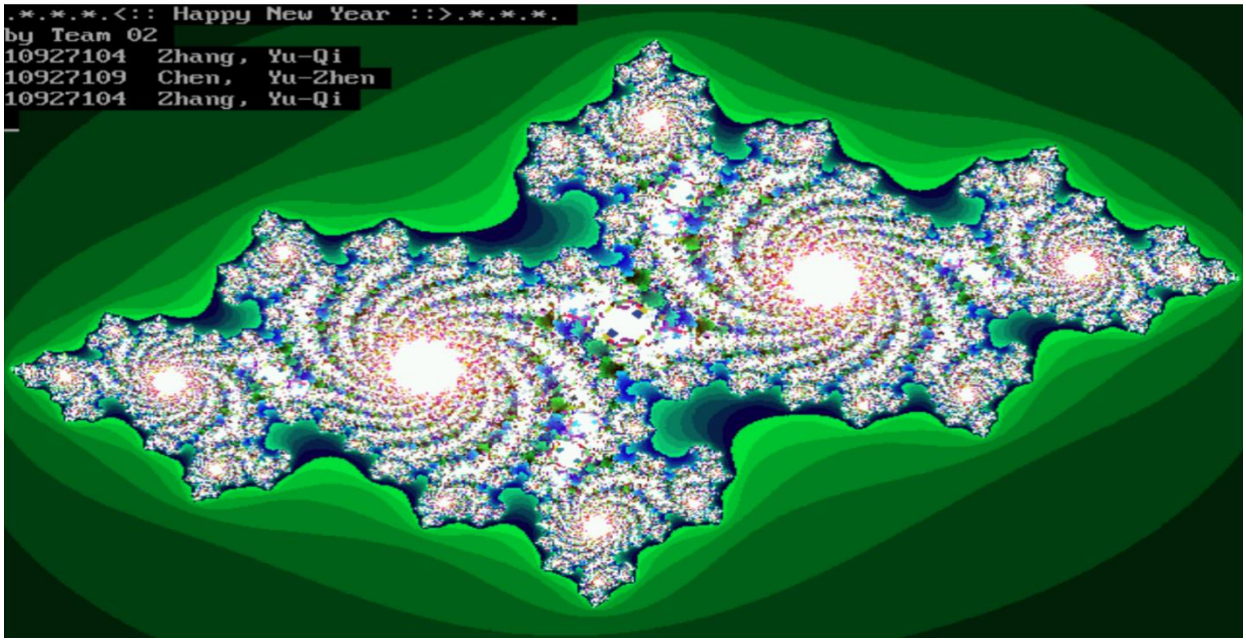
```
pi@raspberrypi ~/asm/Final $ ./a.out
Function 1 : Name
*****Print Name*****
Team 02
Zhang, Yu-Qi
Chen, Yu-Zhen
Zhang, Yu-Qi
*****End Print*****

Function 2 : ID
*****Input ID*****
** Please Enter Member 1 ID: **
10927104
** Please Enter Member 2 ID: **
10927109
** Please Enter Member 3 ID: **
10927104
** Please Enter Command **
p
*****Print Team Member ID and ID Summation*****
10927104
10927109
10927104


ID Summation = 32781317
*****End Print*****
Main Function :
*****Print All*****
Team 02
10927104  Zhang, Yu-Qi
10927109  Chen,  Yu-Zhen
10927104  Zhang, Yu-Qi
ID Summation = 32781317
*****End Print*****

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











四、心得

我們聽說這次Final Project需要兩個禮拜才能完成的時候真的有點嚇到，我們很怕沒辦法在期限內完成，因為公布了這次作業的時候我們還有一些其他的作業還沒完成，所以這次我們就提早開始打，沒有像Midterm Project一樣拖到期限快到了才打。

剛開始打final的時候確實有點不太知道要如何下手，但想通了之後，就變得不難了，除了某些地方要思考一下要如何把C code轉換成Assembly，其他的部分就很順利地解決了。

五、分工方式與負責項目

一同參與Final Project程式撰寫與報告製作。