組合語言與嵌入式系統 Final Project

110 學年度第 1 學期

老師：朱守禮 老師

組別 : 第 20 組

學生： 徐翊華 李俐瑩

修課班級 : 資訊二甲

一、背景

利用這次的 Final project 使我們比起 Midterm project 能更深入的了解「組合語言與嵌入式系統」。以往在上課都只能聽基本理論，但無法實作的我們著實難以理解其中指令的差別與實際運用上會遇到的困難。例如：第四大點提及的 ldr 與 mov，str 的必要性。然而藉此機會，我們更加的瞭解了組合語言之語法、指令、傳址方式、暫存器的使用等，同時也瞭解了在大一計概課所提到的 Call by value, Call by reference 與 scanf 的“&”用意為何與大二資料結構所提及的 Call by address。且這次的 Final project 更具挑戰性，因此無論是在 debug 的過程、與組員討論的過程，亦或是搜尋資料的同時都大大的增加了我們自主學習與合作的能力。且在最後在完成專題時，更加清楚的知道平常所撰寫的高階程式語言的實際運轉方式。

使用工具：qemu 樹莓派模擬器、notepad++、code::blocks

二、方法

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 與 ID.s。

- Name 函數:

```
01              .data
02  name_msg1:  .asciz "*****Print Name*****\n"
03              .align 2
04
05  name_msg2:  .asciz "Team 20\n"
06              .align 2
07              .asciz "YI-HUA HSU\n"
08              .align 2
09
10  name_msg3:  .asciz "LI-YING LI\n"
11              .align 2
12              .asciz "LI-YING LI\n"
13              .align 2
14
15  name_msg4:  .asciz "*****End Print*****\n"
16              .align 2
17
18              .text
19  Name:
20              stmfd sp!, {fp, lr}
21              ldr r0, =name_msg1
22              bl printf
23
24              ldr r0, =name_msg2
25              bl printf
26
27              mov r1, r13
28              adcs r13, r1, r2
29              mov r13, r1
30
31              ldr r0, =name_msg2+12
32              bl printf
33
34              ldr r0, =name_msg3
35              bl printf
36
37              ldr r0, =name_msg3+12
38              bl printf
39
40              ldr r0, =name_msg4
41              bl printf
42              mov r0, #0
43              ldmfd sp!, {fp, pc}
44
```

02 ~ 16 行 : 宣告組別名稱、組員名字，以及程式要輸出之資訊

20 行 : 將 fp、lr 備份到 stack 上

21 ~ 22 行 : 印出開始字串 "*****Print Name*****"

24 ~ 25 行 : 印出組別

27 ~ 29 行 : 題目要求之指令

31 ~ 32 行 : 印出第一位組員名字

34 ~ 35 行 : 印出第二位組員名字

37 ~ 38 行 ： 印出第三位組員名字(因為我們組內只有兩名成員，因此輸出重複的組員名字)

40 ~ 41 行 ： 印出結束字串 "*****End Print***** "

43 行 ： 將 fp、pc 放回記憶體

- ID 函數 ：

```
1       .data
2   n:      .word 0
3           .align 2
4           .word 0
5           .align 2
6           .word 0
7           .align 2
8           .word 0
9           .align 2
10
11  str1:   .asciz "%d"
12
13
14  str2:
15          .asciz "%s"
16          .align 2
17
18
19  str3:
20          .asciz "%d\n"
21
22
23  str4:
24          .asciz "p"
25          .align 2
26
27  str5:
28          .asciz ""
29          .align 2
30
31  str6:
32          .word 0
33          .align 2
34
35  id_info:
36      .asciz "*****Input ID*****\n"
37      .align 2
38
39  id_info1:
40      .asciz "**Please Enter Member 1 ID:**\n"
41      .align 2
```

02 ~ 33 行 ： 宣告多個 word 分別用來存放組員學號、學號之總和，並宣告字串用來設定程式所需之格式

35 ~ 41 行 ： 宣告題目指定格式之字串

```
43  id_info2:
44      .asciz "**Please Enter Member 2 ID:**\n"
45      .align 2
46
47  id_info3:
48      .asciz "**Please Enter Member 3 ID:**\n"
49      .align 2
50      .asciz "** Please Enter Command **\n"
51      .align 2
52      .asciz "*****Print Team Member ID and ID Summation*****\n"
53      .align 2
54
55  id_info4:
56      .asciz "\nID Summation = "
57      .align 2
58      .asciz "*****End Print*****\n\n"
59      .align 2
60
61  id_info5:
62      .asciz "Input error!!\n"
63      .align 2
64
65  id_info6:
66      .asciz "Correct command\n"
67      .align 2
68
69      .text
70      .global id
71
72  id:
73      stmfd sp!,{fp,lr,r0-r3}
74
75      ldr r0, =id_info
76      bl printf
77
78      @ first num
79      ldr r0, =id_info1
80      bl printf
81
82      ldr r0, =str1
83      ldr r1, =n
84      bl scanf
```

43 ~ 67 行 ： 宣告題目指定格式之字串

73 行 ： 將 fp、lr 備份至 stack 上，並備份 r0-r3，回傳至 main 之變數位址

79 ~ 80 行 ： 印出資訊讓使用者輸入第一位成員之學號

82 ~ 84 行 ： 讀入輸入之數字至 n ：scanf("%d", &n)

```
86      @ second num
87      ldr r0, =id_info2
88      bl printf
89
90      ldr r0, =str1
91      ldr r1, =n+4
92      bl scanf
93
94      @ third num
95      ldr r0, =id_info3
96      bl printf
97
98      ldr r0, =str1
99      ldr r1, =n+8
100     bl scanf
101
102     loop_a:
103         @ Enter command
104         ldr r0, =id_info3+32
105         bl printf
106
107         ldr r0, =str2
108         ldr r1, =str6
109         bl scanf
110         @ scanf 'p'
111
112         ldr r8, =str4 @r8 == 'p'
113         ldr r8, [r8, #0]
114         ldr r1, =str6
115         ldr r1, [r1]
116
117         cmp r1, r8
118         ldrne r0, =id_info5
119         ldreq r0, =id_info6
120         bl printf
121
122         ldr r1, =str6
123         ldr r1, [r1]
124         cmp r1, r8
125         beq done
126         blne loop_a
127 done:
128
129     addcc r0, r1, r2
```

87 ~ 88 行：印出資訊讓使用者輸入第二位成員之學號

90 ~ 92 行：讀入輸入之數字至 n+4 ：scanf("%d", &n+4)

94 ~ 96 行：印出資訊讓使用者輸入第三位成員之學號

98 ~ 100 行：讀入輸入之數字至 n+8 ：scanf("%d", &n+8)

102 行：迴圈開始

104~105 行：輸出資訊讓使用者輸入指令

107~109 行：讀入指令 ： scanf("%s", &str6)

112~115 行：讓 r8 = ‘p’，並傳 str6 之值至 r1

117~120 行：若輸入之指令等於 p 則輸出指令正確，反之，輸出指令錯誤

122~123 行：傳 str6 之值至 r1

124~126 行：若輸入之指令等於 p 則繼續執行下方程式，反之，再重新跑 loop_a

129 行：若 c 尚未設置則 r0 = r1 + r2

```
131     ldr r0, =id_info3
132     add r0, r0, #60
133     bl printf
134
135     @ print three nums
136     ldr r4, =n
137     ldr r5, =n+4
138     ldr r6, =n+8
139
140     ldr r0, =str3
141     ldr r1, [r4, #0]!
142     bl printf
143
144     ldr r0, =str3
145     ldr r1, [r5]
146     bl printf
147
148     ldr r0, =str3
149     ldr r1, [r6]
150     bl printf
151
152     ldr r0, =id_info4
153     bl printf
154
155     @ add three num
156     ldr r4, =n
157     ldr r4, [r4]
158     ldr r5, =n+4
159     ldr r5, [r5]
160     ldr r6, =n+8
161     ldr r6, [r6]
162
163     add r7, r4, r5
164     add r7, r7, r6, lsl#0
165
166     ldr r0, =n+12
167     str r7, [r0]
168
169     ldr r0, =str3
170     mov r1, r7
171     bl printf
172
173     ldr r0, =id_info4+20
174     bl printf
```

131~133 行：輸出"*****Print Team Member ID and ID Summation*****\n"

136~138 行：分別將存三個學號之址存於 r4、r5、r6

140~142 行：輸出第一位組員之學號

144~146 行：輸出第二位組員之學號

148~150 行：輸出第三位組員之學號

152~171 行：分別存三個學號之值於 r4、r5、r6，將三個學號相加之值存於 r7 並印出

173~174 行：輸出結束資訊

```
176         ldmfd sp!, {r0-r3}
177
178         ldr r4, =n
179         ldr r4, [r4]
180         str r4, [r0]
181         ldr r4, =n+4
182         ldr r4, [r4]
183         str r4, [r1]
184         ldr r4, =n+8
185         ldr r4, [r4]
186         str r4, [r2]
187         ldr r4, =n+12
188         ldr r4, [r4]
189         str r4, [r3]
190
191
192         ldmfd sp!,{fp, pc}
193
```

176：將暫存之變數位址放回 r0-r3

178～189 行 ： 將組員學號及學號總合放在 r0-r3(回傳至 main 做使用)

- drawJuliaSet 函數

```
1     .data
2
3   maxIter:    .word 255
4
5   zx:         .word 0
6
7   zy:         .word 0
8
9   tmp:        .word 0
10
11  i:          .word 0
12  x:          .word 0
13  y:          .word 0
14
15  color:      .short 0
16
17  fif_hun:    .word 1500
18  ten_hun:    .word 1000
19  four_mil:   .word 4000000
20
21
22      .text
23      .global juliaset
24
25  juliaset:
26      stmfd sp!,{fp,lr}
27      ldr r4, =x
28      mov r5, #0
29      str r5, [r4]
30      ldr r4, =y
31      mov r5, #0
32      str r5, [r4]
33
34  loop1:
35      ldr r4, =x
36      ldr r4, [r4]
37      cmp r4, r2
38      bge loop1_end @ if(>=)
39      blt loop2
```

3 ~ 19 行：變數宣告

27 ~ 29 行：設 x = 0

30 ~ 32 行：設 y = 0

35 ~ 39 行：進入第一個迴圈，if (x >= width) 就結束迴圈，x < width 就繼續執行

```
40
41  loop2:
42      ldr r5, =y
43      ldr r5, [r5]
44      cmp r5, r3
45      bge loop2_end @ if(>=)
46
47      @ ============================================
48      @ zx = 1500 * (x-(width>>1))/(width>>1)
49
50      mov r7, r2        @ r7 = width
51      mov r7, r7, asr #1  @ r7 = width >> 1
52
53      ldr r4, =x
54      ldr r4, [r4]
55      sub r6, r4, r7      @ r6 = x - ( width >> 1 )
56      ldr r8, =fif_hun    @ r8 = 1500
57      ldr r8, [r8]
58      mul r6, r8, r6      @ r6 = r8 * r6
```

42 ~ 45 行 ： 進入第一個迴圈，if (y >= height) 就結束迴圈

50 ~ 51 行 ：r7 = (width 右移 1)

53 ~ 58 行 ：r6 = 1500 * (x – width>>1)

```
59      mov r4, r0
60      mov r5, r1
61      mov r0, r6
62      mov r1, r7
63      mov r8, r2
64      mov r9, r3
65      bl __aeabi_idiv
66      ldr r6, =zx
67      str r0, [r6]
68      mov r0, r4
69      mov r1, r5
70      mov r2, r8
71      mov r3, r9
72
73      @============================================
74      @ zy = 1000 * (y-(height>>1))/(height>>1)
75
76      mov r7, r3        @ r7 = height
77      mov r7, r7, asr #1  @ r7 = height >> 1
78
79      ldr r5, =y
80      ldr r5, [r5]
81      sub r6, r5, r7      @ r6 = y - ( height >> 1 )
82      ldr r8, =ten_hun    @ r8 = 1000
83      ldr r8, [r8]
84      mul r6, r8, r6      @ r6 = r8 * r6
85      mov r4, r0
86      mov r5, r1
87      mov r8, r2
88      mov r9, r3
89      mov r0, r6
90      mov r1, r7
91      bl __aeabi_idiv
92      ldr r6, =zy
93      str r0, [r6]
94      mov r0, r4
95      mov r1, r5
96      mov r2, r8
97      mov r3, r9
```

59 ~ 71 行 ： 將 r0 – r3 移到別的暫存器，並讓 zx = 1500 * (x –width>>1) / (x – width>>1)

76 ~ 97 行 ： 用一樣的方式讓 zy = 1000 * (y-(height>>1))/(height>>1)

```
 99      @=============================================
100
101      ldr r7, =maxIter
102      ldr r7, [r7]
103      ldr r6, =i
104      str r7, [r6]
105
106      @=============================================
107
108  loop3:
109      ldr r6, =zx          @ r6 = zx
110      ldr r6, [r6]
111      mul r8, r6, r6       @ r8 = zx * zx
112      ldr r7, =zy          @ r7 = zy
113      ldr r7, [r7]
114      mul r9, r7, r7       @ r9 = zy * zy
115      add r4, r8, r9       @ r4 = zx*zx + zy*zy
116      ldr r5, =four_mil
117      ldr r5, [r5]
118      cmp r4, r5           @ r4 < 4000000
119      bge loop3_end
120
121      ldr r6, =i
122      ldr r6, [r6]
123      cmp r6, #0           @ i > 0
124      ble loop3_end
```

101 ~ 104 行 ： 將 i 設為 maxIter

108 ~ 124 行 ： 若 zx*zx + zy*zy >= 4000000 或 i < = 0 就結束迴圈

```
127      @ tmp = ( zx*zx - zy*zy ) / 1000 + cX
128
129      sub r6, r8, r9       @ r6 = zx * zx - zy * zy
130      ldr r4, =ten_hun     @ r4 = 1000
131      ldr r4, [r4]
132      mov r5, r0
133      mov r7, r1
134      mov r8, r2
135      mov r9, r3
136      mov r0, r6
137      mov r1, r4
138      bl __aeabi_idiv    @ tmp = tmp / 1000
139      ldr r6, =tmp
140      str r0, [r6]
141      mov r0, r5
142      mov r1, r7
143      mov r2, r8
144      mov r3, r9
145      ldr r6, =tmp
146      ldr r6, [r6]
147      add r7, r6, r0       @ tmp = tmp + cX
148      ldr r6, =tmp
149      str r7, [r6]
```

129 行 : r6 暫存 zx*zx - zy*zy 之值

130 ~ 144 行 ： 將(zx*zx - zy*zy) / 1000 之值存入 tmp

145 ~ 149 行 : tmp = tmp + cx

```
151     @ zy = ( 2 * zx * zy ) / 1000 + cy
152
153     ldr r7, =zy          @ r7 = zy
154     ldr r7, [r7]
155     ldr r6, =zx          @ r6 = zx
156     ldr r6, [r6]
157     mul r7, r7, r6       @ zy = zy * zx
158     mov r7, r7, lsl #1    @ zy = zy * 2
159
160     mov r4, r0
161     mov r5, r1
162     mov r8, r2
163     mov r9, r3
164     mov r0, r7
165     ldr r1, =ten_hun
166     ldr r1, [r1]
167     bl __aeabi_idiv      @ zy = zy / 1000
168     ldr r7, =zy
169     str r0, [r7]
170     mov r0, r4
171     mov r1, r5
172     mov r2, r8
173     mov r3, r9
174     ldr r7, =zy
175     ldr r7, [r7]
176     add r8, r7, r1, lsl #0       @ zy = zy + cY
177     ldr r7, =zy
178     str r8, [r7]
179
180     ldr r8, =tmp
181     ldr r8, [r8]
182     ldr r6, =zx
183     str r8, [r6]     @ zx = tmp
184     ldr r4, =i
185     ldr r4, [r4]
186     sub r4, r4, #1       @ i--
187     ldr r5, = i
188     str r4, [r5]
189     b loop3
```

153 ~ 158 行 ： 將 2 * zx * zy 之值存入 zy

160 ~ 173 行 ：zy = zy /1000

174 ~ 178 行 ：zy = zy + cY

180 ~ 183 行 ：zx = tmp

186 ~ 189 行 ： 將 i 值減 1，重新執行第三個迴圈(loop3)

```
192  loop3_end:
193
194      @================================================
195      @ color = ~(((i&0xff)<<8) | (i&oxff)) & 0xffff
196      ldr r4, =i
197      ldr r4, [r4]
198      mov r4, r4, lsl #8  @ r6 = i << 8
199      uxth r6, r4
200
201      ldr r4, =i
202      ldr r4, [r4]
203      uxth r4, r4
204      orr r5, r6, r4      @ r5 = (r6|r4)
205      ldr r4, =color
206      strh r5, [r4]
207      ldr r5, =color
208      ldrh r5, [r5]
209      mvn r5, r5          @ r5 = !r5
210      ldr r6, =color
211      strh r5, [r6]
212
213      @================================================
214      @ frame[y][x] = color
215      ldr r7, =y
216      ldr r7, [r7]
217      ldr r5, =x
218      ldr r5, [r5]
219      mul r6, r7, r2
220      add r6, r6, r5              @ r6 = y*640 + x
221      mov r6, r6, lsl #1
222      add r6, r6, #8
223      ldr r4, =color
224      ldrh r4, [r4]
225      strh r4, [sp, r6]    @ frame[y][x]=color
226
227      @================================================
228
229      ldr r4, =y
230      ldr r5, [r4]
231      add r5, r5, #1
232      str r5, [r4]
233      b loop2
```

196 ~ 198 行：將向左移 8

199 行：將半字擴展到 32 位

201 ~ 204 行：將 ((i&0xff)<<8) | (i&oxff)) 之值暫存在 r5

205 ~ 211 行：將 ((i&0xff)<<8) | (i&oxff)) 的 1’complement 存至 color

215 ~ 222 行：計算二維陣列儲存的位址

223 ~ 225 行：將 color 值存入 frame[y][x]

229 ~ 233 行：將 y 值加 1，重新執行 loop2

```
235     loop2_end:
236         ldr r4, =x
237         ldr r5, [r4]
238         add r5, r5, #1
239         str r5, [r4]
240
241         mov r5, #0
242         ldr r4, =y
243         str r5, [r4]
244         b loop1
245
246     loop1_end:
247         mov r4, r14
248         adds r14, r0, r15
249         mov r14, r4
250         ldmfd sp!,{fp,pc}
```

235 ~ 239 行 : loop2 執行結束後，x 值加 1，y 值歸零並重複執行 loop1

247 ~ 250 行 : 執行題目要求之指令

- main 函數

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <stdint.h>
4  #include <string.h>
5  #include <sys/types.h>
6  #include <sys/stat.h>
7  #include <sys/mman.h>
8  #include <fcntl.h>
9  #define FRAME_WIDTH     640
10 #define FRAME_HEIGHT    480
11
12 #define FRAME_BUFFER_DEVICE "/dev/fb0"
13
14 extern void Name() ;
15 extern void id(int *sum, int *a, int *b, int *c) ;
16 extern void juliaset() ;
17
18
19 int main()
20 {
21     //RGB16
22     int16_t frame[FRAME_HEIGHT][FRAME_WIDTH];
23
24     int max_cX = -700;
25     int min_cY = 270;
26
27     int cY_step = -5;
28     int cX = -700;  // x = -700~-700
29     int cY ;   // y = 400~270
30
31     int fd = 0;
32
33     int name1, name2, name3, sum ;
34
35     printf( "Function1: Name\n" );
36
37     //Dummy Function. Please refer to the specification of Project 1.
38     Name();
39
40     printf( "Function2: ID\n" );
41
42     //Dummy Function. Please refer to the specification of Project 1.
43     id(&name1, &name2, &name3, &sum);
```

14 ~ 16 行 ： 宣告三個 assembly 函數

22 ~ 33 行 ： 宣告變數

35 ~ 43 行 ： 執行 ID、 Name 函數

```
44      int id1 = name1;
45      int id2 = name2;
46      int id3 = name3;
47
48      //Dummy printout. Please refer to the specification of Project 1.
49      printf( "Main Function:\n" );
50      printf( "*****Print All*****\n" );
51      printf( "Team 20 :\n");
52      printf( "%d YI-HUA HSU\n", name1) ;
53      printf( "%d LI-YING LI\n", name2) ;
54      printf( "%d LI-YING LI\n", name3) ;
55      printf( "ID Summation = %d\n", sum ) ;
56
57      printf( "*****End Print*****\n" );
58
59
60      printf( "\n***** Please enter p to draw Julia Set animation *****\n" );
61      // 等待使用者輸入正確指令
62      while(getchar()!='p') {}
63      // 清除畫面
64      system( "clear" );
65      // 打開 Frame Buffer 硬體裝置的Device Node，準備之後的驅動程式呼叫
66      fd = open( FRAME_BUFFER_DEVICE, (O_RDWR | O_SYNC) );
67
68      if( fd<0 )
69      {   printf( "Frame Buffer Device Open Error!!\n" ); }
70      else
71      {
72
73          for( cY=400 ; cY>=min_cY; cY = cY + cY_step ) {
74
75              juliaset( cX, cY, FRAME_WIDTH, FRAME_HEIGHT, frame );
76
77              write( fd, frame, sizeof(int16_t)*FRAME_HEIGHT*FRAME_WIDTH );
78
79              lseek( fd, 0, SEEK_SET );
80          }
81
82          //Dummy printout. Please refer to the specification of Project 1.
```

45 ~ 46 行：儲存備份學號

49 ~ 60 行：印出組別、學號、組員姓名、學號加總

60 ~ 62 行：等待使用者輸入正確指令

64 行：清除畫面

66 行：打開 Frame Buffer 硬體裝置的 Device Node，準備之後的驅動程式呼叫

68 ~ 69 行：防呆，確認 Frame Buffer Device 成功開啟

73 ~ 79 行：在 x、y 設定範圍內執行 Julia Set 函數

```
83
84          printf( ".*.*.*.<:: Happy New Year ::>.*.*.*.\n" );
85          printf( "by Team 20\n" );
86          printf( "%d   Yi-Hua Hsu\n", id1 );
87          printf( "%d   Li-Ying Li\n", id2 );
88          printf( "%d   Li-Ying Li\n", id3 );
89
90          // 關閉 Device Node檔案，結束驅動程式的使用
91          close( fd );
92      }
93
94      // 等待使用者輸入正確指令
95      while(getchar()!='p') {}
96
97      return 0;
98  }
99
100
```

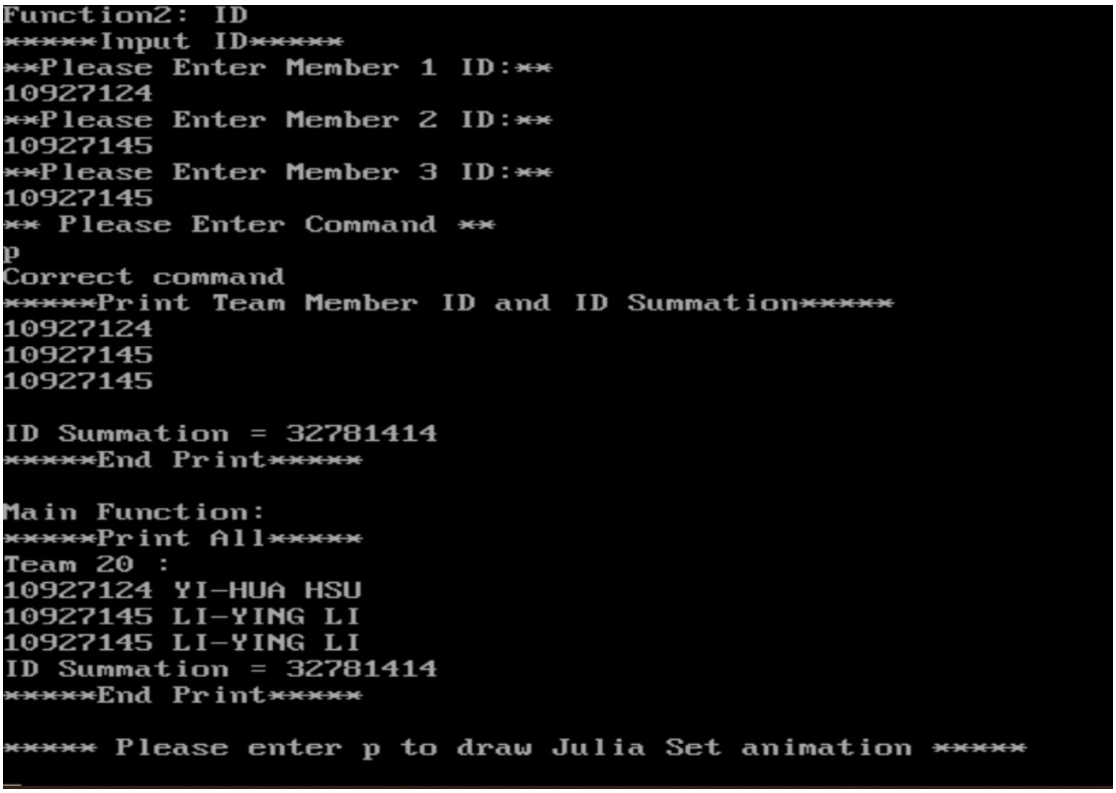
84 ~ 88 行：新年快樂!印出組員學號姓名

91 行：關閉檔案

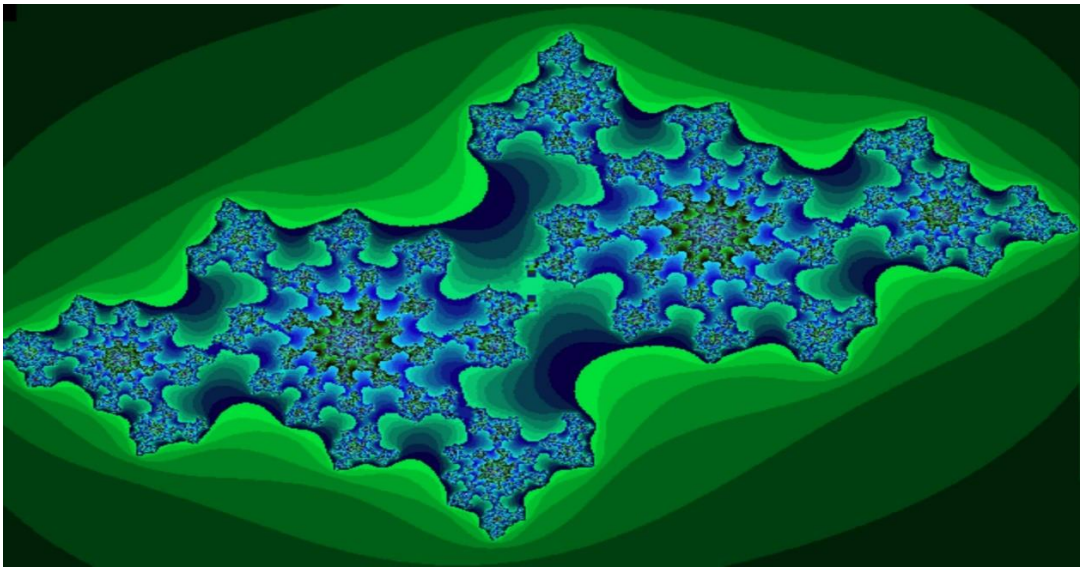
95 行：等待使用者輸入正確指令

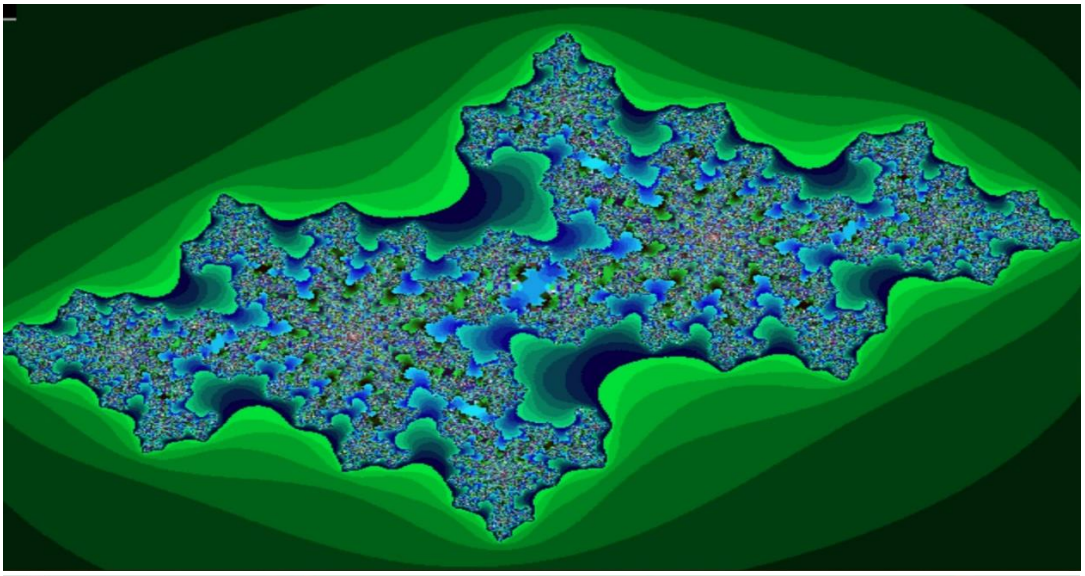
三、結果:

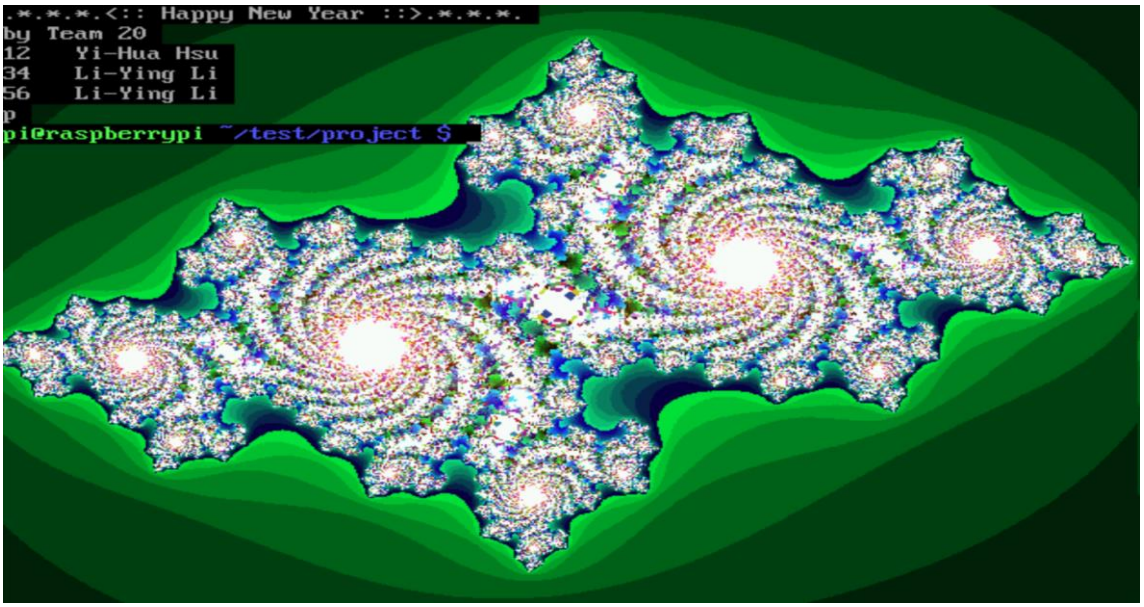
在 Main 中呼叫 Name 與 ID 函數，分別達成前兩支函數的分項功能。結合 Name 與 ID 函數所記錄的資料，輸出完整的組別、組員資訊、與組員學號數值及學號總和計算結果，結果如下:



- Julia Set 動畫的 5 張畫面:

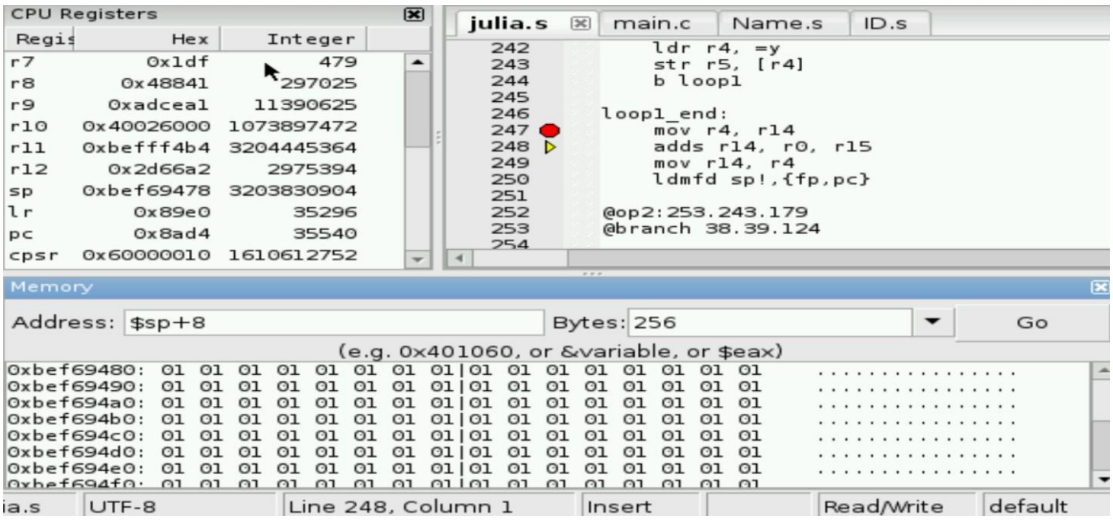




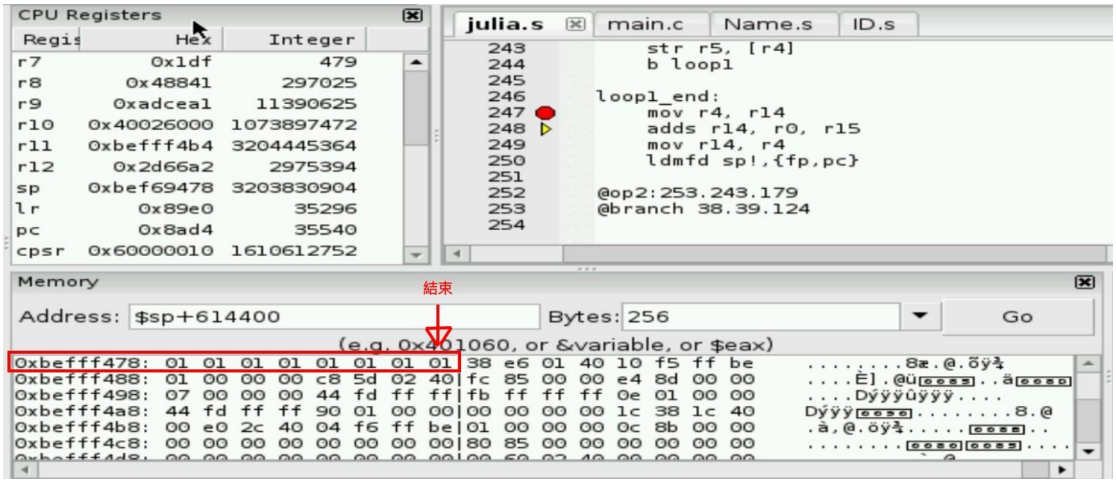

.*.*.*.<:: Happy New Year ::>.*.*.*.
by Team 20
12   Yi-Hua Hsu
34   Li-Ying Li
56   Li-Ying Li
p
pi@raspberrypi ~/test/project $

- drawJuliaSet 函式指定說明項目:

  1. frame 陣列的記憶體區塊部份內容:



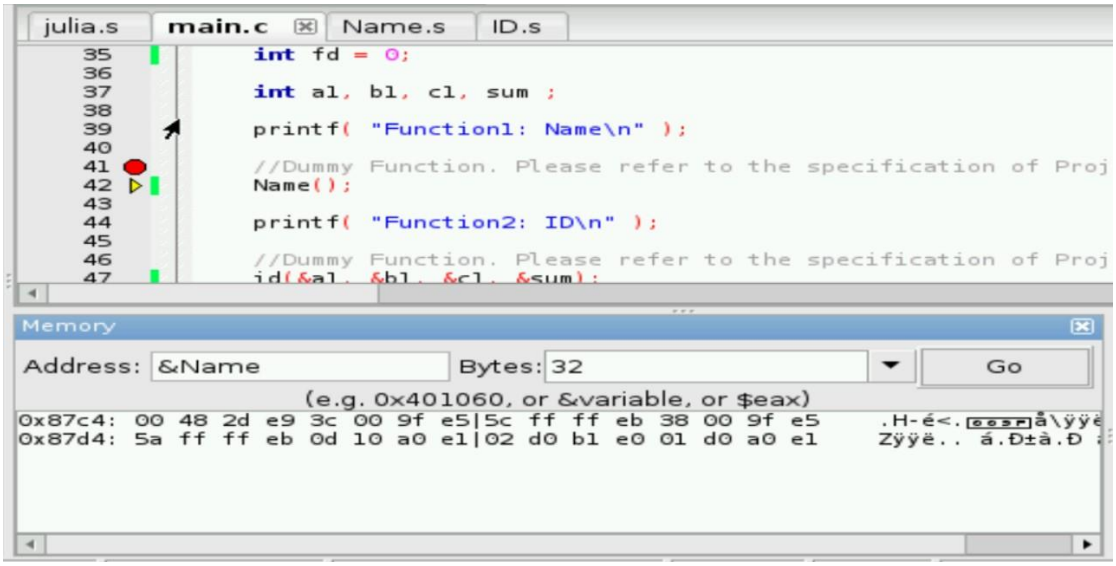
起始記憶體位址:0xbef69480 ($sp+8)



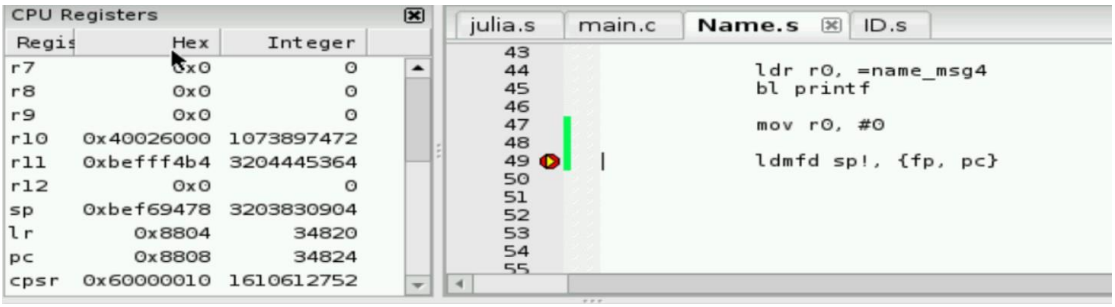
結束記憶體位址: :0xbef6948f($sp+614400)

- Main 函式指定說明項目 ：

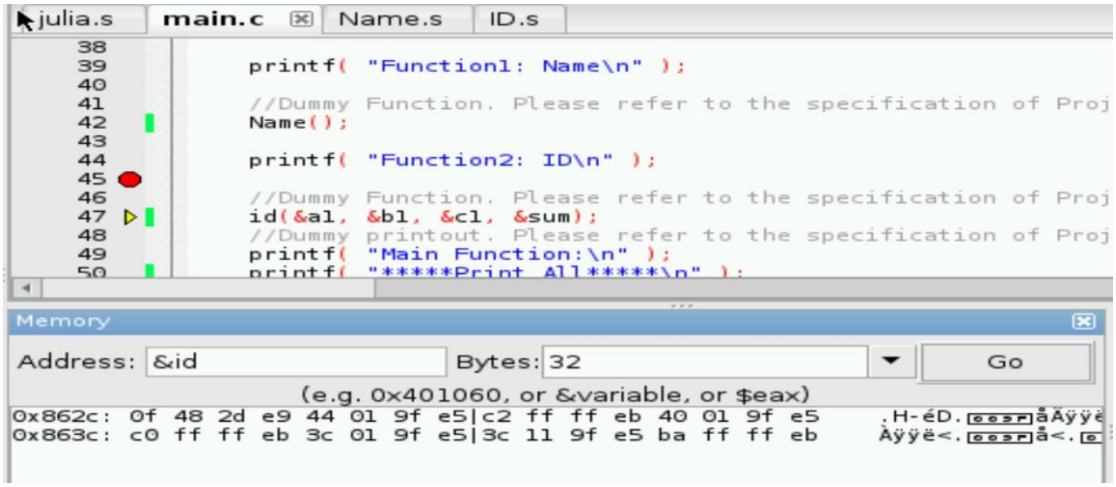
1.Name 函式所在位址: 0x87c4



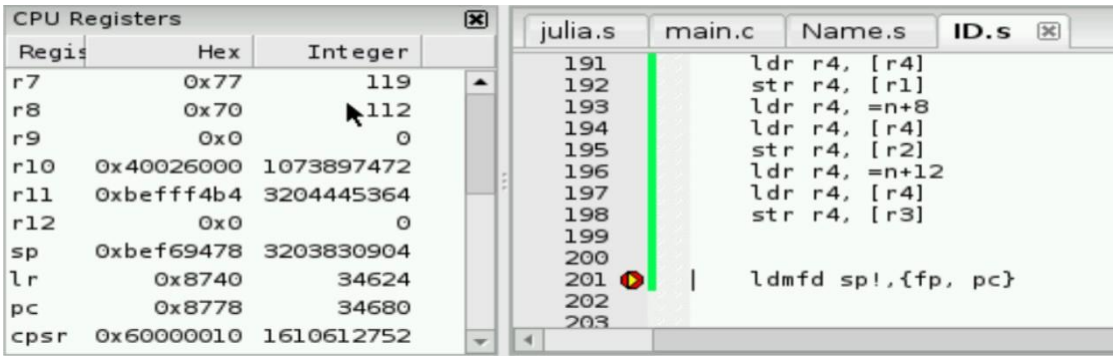
2. Name 函數返回地址 0x8804



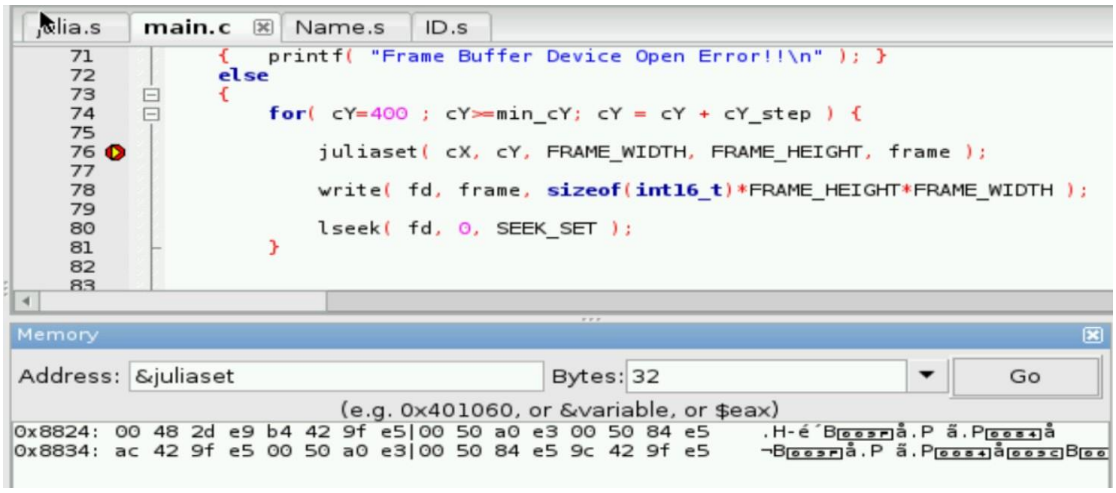
3. ID 函數所在位址: 0x862c



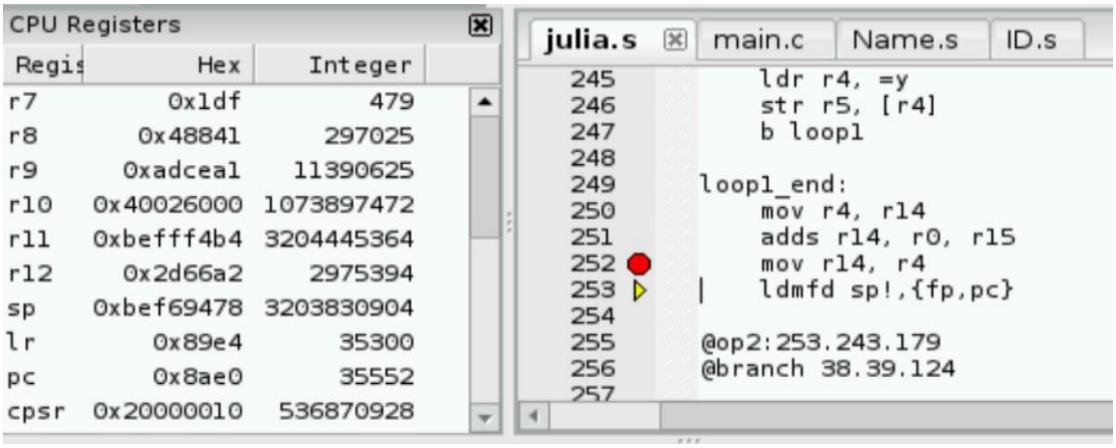
4.ID 函數返回地址: 0x8740



5. drawJuliaSet 函數所在位址: 0x8824



6. drawJuliaSet 函數返回地址: 0x89e4



四、討論

在撰寫程式中遇到之困難：

1. ldr 與 mov 指令的不熟悉，課本中的註解在兩條指令中的註解接寫到 Rm = Rn，因此在實際使用上時遇到了些許錯亂。
2. .data 段中的 ”.align” 的必要性。
3. 迴圈中的 index 值都沒有被正確加一。
4. 程式最終效率問題。
5. 反組譯以後的效率問題。
6. 在 Arm 語言裡 Call by address。
7. 在 drawJuliaSet 執行以前，main.c 裡的 int name1, name2, name3 分別存放著由 id( )回傳的三個學號。但執行完以後卻無法在”Happy New Year” 中正常使用。

五、結論

針對撰寫中所遇到的困難找到相對應的解決辦法：

1. 查閱課本與實際使用後更加了解與熟悉兩個指令之間的不同。
2. 在 midterm project 中即有此疑問，因此在這次撰寫 drawJuliaSet 時刻意不添加.align，然而並未造成任何影響。在網路上也有查詢相關資料，其內容為將資料的存放地址對齊，是個偽指令。
3. 我們只有利用 ldr 與 add 去將此暫存器的值加一，但並未儲存回其應該儲存的地方，在使用 str 指令儲存後問題及解決。
4. 在最原先最一開始打完 Final project 時，程式執行時間非常久，因此開始思考可以改進的地方藉此提高程式效能。其主要解決方法有二：
   i. 減少程式碼，於一開始即將 R0~R3 存放至 sp，即可省略在每次呼叫 printf 或其他需要使用 R0~R3 暫存器時必須將其先備份至其他暫存器的步驟。

ii. 少用除法，多利用加法與其 2 次方的關係。經過實測，少了兩條乘法指令程式執行時間少了三秒。

5. 將原先的 drawJuliaSet 反組譯成 assembly code，發現其程式碼卻比我們的 Final project 來得多，可是其執行時間依然少了許多，其原因來在釐清。

6. 因為 main 是以“C”撰寫，因此無法 Call by reference，而是必須 call by address，在一開始即在 main 裡遇到困難，原因在於不熟悉 Call by address 的寫法，宣告 function 時使用的是“*”，而在呼叫 function 時使用的是“&”。但也是藉由呼叫時使用“&”讓我們得以知曉回傳應也是個位置。而一開始我們只有把 R0~R3 指向我們要回傳的目標的位置，但實際上並不然，應是將目標存進一開始 R0~R3 的位置。

7. 其發生原因不詳，猜測為動到記憶體。解決辦法為多設變數 id1, id2, id3 去存回傳的 name1, name2, name3，在”Happy New Year”中使用此三新增變數來成功正確 printf。

六、心得感想

1. 徐翊華：

經過 Midterm project 後撰寫這次的 Final project 可謂開場順利，沒有了一開始的茫然、徬徨無助的感覺，而是一看到.c 檔腦海中即可以開始構思 assembly 的架構與寫法，了解後即發現其實其與高階語言也有雷同之處，只是多了部使用暫存器的指令。以迴圈而言，我只比高階語言要多一步 ”b” 讓他知道要去執行哪裡，index 要加一時要多一行指令讓其等於 index 的位置等。因此熟悉以後其實會發現組合語言沒有想像中的那麼困難。

而這次的作業相比於上次而言更要我們去了解每個程式碼的用意，不像上次是有課本可以直接照著參考，因此在過程中還是遇到了些許困難，例如記憶體區段錯誤與指令的使用錯誤等，但最後依然一一解決。

很慶幸的我有一個很好的組員，不論是分工亦或是合作都十分順利，互相補齊對方不足的地方，讓此次的作業可以順利完成。因為我的 mac 無法順利下載 qemu 樹莓派模擬器，因此都必須額外跟她約時間來處理相關問題，真的非常非常謝謝她。

2. 李俐瑩：

原本以為經過了 midterm project 就能摸透組合語言，沒想到在開始做 final project 處處卡關，才發現原來期中的作業幾乎可以照著課本上的範例寫出來，但這次的 project 是要真正理解每個指令，並且知道應該要坐傳值還是傳址。

剛開始我們直接照著想法打完整個程式，執行時才發現到處都出現記憶體區段錯誤，改完了這個後又發現所有的值都只有放在暫存器，而沒有放回記憶體，導致所有的值依然是 0，將所有的程式碼再重新打一遍後，才終於做好這次的作業。

經過了這次的 final project 我們更加的理解每個指令的用途以及用法，也學習到了 julia set，還有如何在組合語言中 call by address，最重要的是和組員一起完成這份報告的成就感。

七、組員分工：

徐翊華：程式撰寫、報告撰寫

李俐瑩：程式撰寫、報告撰寫

八、未來展望：

身為「資工人」，我們理應去了解我們平常所撰寫的高階語言在電腦上的實際運作方式，去深入瞭解他，因為這都將永遠跟著我們，不可只是一隻一知半解或是敷衍了事。而這堂課及很好地利用了兩次的 project 讓我們熟悉組合語言，讓我們可以在未來更具競爭能力，畢竟組合語言是少數外系沒有只有資工有學的課程。

而其亦可運用在「逆向工程」，例如：「惡意程式分析」、「漏洞挖掘」、「軟體破解」等領域。而在缺乏原始程式碼的情況下，逆向工程技術也成為了唯一一個去理解惡意程式或著軟體產品的手段。(網路資訊彙整、Midterm project)