

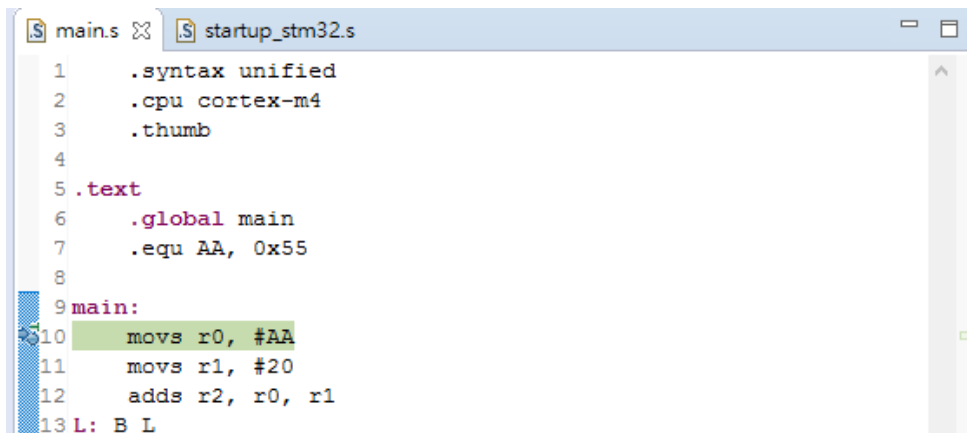
實驗名稱：實驗一 實驗環境建立與 Debugger 操作

實驗目的：測試實驗器材、熟悉開發環境

實驗步驟、結果與問題回答：

1. 專案建立與程式編譯

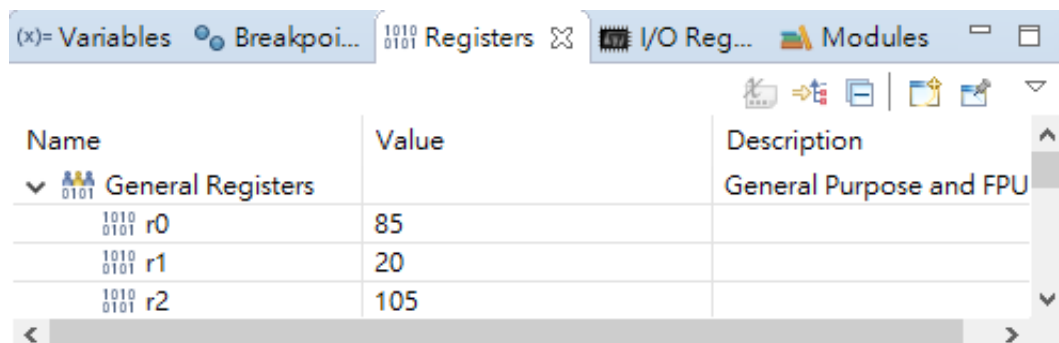
依照助教給的lab1_note教學，建立一個STM32 eclipse project，新增一個內容如下的main.s程式碼並透過debugger觀察程式執行結果。



```
1  .syntax unified
2  .cpu cortex-m4
3  .thumb
4
5  .text
6  .global main
7  .equ AA, 0x55
8
9  main:
10  movs r0, #AA
11  movs r1, #20
12  adds r2, r0, r1
13  L: B L
```

Q: 程式執行結束後 R2 值為多少？如何觀察？

A: 程式執行結束後 R2 為 105。將 Debug prospective 中右上角的視窗切換到 Registers，就可以查看程式進行中各個 register 的變化數值，如下圖。



Name	Value	Description
General Registers		General Purpose and FPU
r0	85	
r1	20	
r2	105	

2. 變數宣告與記憶體觀察

將main.s修改成以下程式碼並編譯執行觀察程式執行結果，並透過memory monitor觀察X內容值變化與回答問題。

```

main.s startup_stm32.s
1  .syntax unified
2  .cpu cortex-m4
3  .thumb
4
5  .data
6      X: .word 100
7      str: .asciz "Hello World!"
8
9  .text
10     .global main
11     .equ AA, 0x55
12
13 main:
14     ldr r1, =X
15     ldr r0, [r1]
16     movs r2, #AA
17     adds r2, r2, r0
18     str r2, [r1]
19
20     ldr r1, =str
21     ldr r2, [r1]
22 L: B L

```

程式執行到第18行後register和memory中的狀態：

1010 0101 r0	100	
1010 0101 r1	536870912	
1010 0101 r2	185	

Monitors		536870912 <Hex> 536870912: 0x20000000 <Unsigned Integer> New F			
536870912	Address	0 - 3	4 - 7	8 - B	C - F
536870916	20000000	185	1819043144	1867980911	560229490

程式執行完後register和memory中的狀態：

1010 0101 r0	100	
1010 0101 r1	536870916	
1010 0101 r2	1819043144	

Monitors		536870912 <Hex> 536870912: 0x20000000 <Unsigned Integer> New F			
536870912	Address	0 - 3	4 - 7	8 - B	C - F
536870916	20000000	185	1819043144	1867980911	560229490

Q1: 變數X與str的初始值是由誰在何處初始化的？

A1: X和str的初始值是assembler在程式開始之前就先寫進data section的。

Q2: 若將X宣告改在text section對其程式執行結果會有何改變？

A2: X的位置會從536870912變成134218284、str的位置會從536870916變成536870912(data section的開頭、原本X的位置)，而且因為text section只能read不能write，所以第18行的str r2, [r1]並不能成功改到X的值。

Q3: 程式執行完畢後r2內容與str字串在memory 前4個byte呈現內容有何差異？

A3: r2的前4個byte是6C 6C 65 48，而str的前四個byte是48 65 6C 6C。

Q4: 變數str "Hello World!" 有無其他種宣告方式？若有請說明其中一種。

A4: 除了.asciz以外，也可以用.ascii宣告。

3. 簡易算數與基本記憶體指令操作

這部分實驗需要在 **data section** 中宣告三個 X,Y,Z 長度為 4byte 的變數並利用 ARM 組合語言計算以下式子，找出這些變數的 **memory address** 並觀察程式執行結果。

$$X = 5$$

$$Y = 10$$

$$X = X * 10 + Y$$

$$Z = Y - X$$

```

1  .syntax unified
2  .cpu cortex-m4
3  .thumb
4
5  .data
6  X: .word 5
7  Y: .word 10
8  Z: .word
9
10 .text
11 .global main
12 .equ AA, 0xAA
13
14 main:
15     ldr r0, =Y
16     ldr r3, [r0]
17     ldr r0, =X
18     ldr r2, [r0]
19     movs r1, #AA
20     muls r2, r1
21     adds r2, r3
22     str r2, [r0]
23     subs r3, r2
24     ldr r1, =Z
25     str r3, [r1]
26
27 L: B L

```

memory address:

X: 536870912

Y: 536870916

Z: 536870920

程式開始時memory中的狀態：

Monitors				
536870912: 0x20000000 <Signed Integer>				
Address	0 - 3	4 - 7	8 - B	C - F
20000000	5	10	0	536871668
20000010	536871772	536871876	0	0

程式執行到第 22 行後 memory 中的狀態：

Monitors				
536870912: 0x20000000 <Signed Integer>				
Address	0 - 3	4 - 7	8 - B	C - F
20000000	60	10	0	536871668
20000010	536871772	536871876	0	0

程式執行完後memory中的狀態：

Monitors				
536870912: 0x20000000 <Signed Integer>				
Address	0 - 3	4 - 7	8 - B	C - F
20000000	60	10	-50	536871668
20000010	536871772	536871876	0	0

心得討論與應用聯想：

這次的實驗感覺都還只是一些基本的指令練習，還沒有用到什麼太複雜的東西。不過跟以前寫的程式比起來，在變數使用還有資料位置、大小的部分都要花更多心思安排，可能就像老師在第一堂課說的一樣，這是一個可以比較貼近硬體、用更接近機器的思維寫程式的機會，雖然還不知道之後的實驗會不會出現什麼可怕的東西，不過至少目前看起來似乎是挺好玩的。還有看到盒子裡還有這麼多鍵盤、七段顯示器之類的各種外接零組件，有心的話應該真的可以把這片板子做成各種東西吧。