**實驗一 實驗環境建立與Debugger操作**

第一組 0410137劉家麟 0416324胡安鳳

1. **實驗目的**

測試實驗器材

熟悉開發環境

1. **實驗步驟**

**2.1 專案的建立與程式編譯**

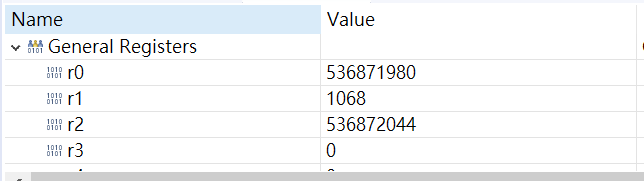
請依照助教給的lab1\_note教學，建立一個STM32 eclipse project，新增一個內容如下的main.s程式碼並透過debugger觀察程式執行結果。

|  |
| --- |
| .syntax unified  .cpu cortex-m4  .thumb  .text  .global main  .equ AA, 0x55  main:  movs r0, #AA  movs r1, #20  adds r2, r0, r1  L: B L |

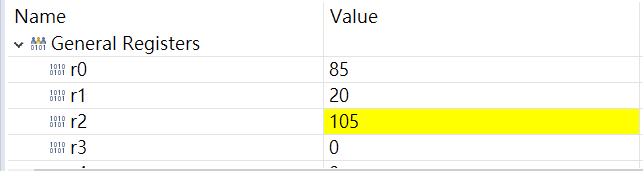
Q: 程式執行結束後R2值為多少？如何觀察？

A: 可以透過Debugger裡面的Register觀察，最後R2的值為105

# 程式尚未開始執行前



# 用Step in一步一步執行結束後，可觀察到R2為105



**2.2 變數宣告與記憶體觀察**

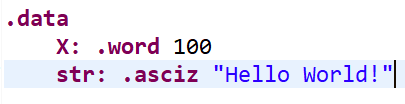
將main.s修改成以下程式碼並編譯執行觀察程式執行結果，並透過memory monitor觀察X內容值變化與回答問題。

|  |
| --- |
| .syntax unified  .cpu cortex-m4  .thumb  .data  X: .word 100  str: .asciz "Hello World!"  .text  .global main  .equ AA, 0x55  main:  ldr r1, =X  ldr r0, [r1]  movs r2, #AA  adds r2, r2, r0  str r2, [r1]  ldr r1, =str  ldr r2, [r1]  L: B L |

Q1: 變數X與str的初始值是由誰在何處初始化的？

A1: X = 100以及str = ”Hello World”的初始化是在data segment進行的

# .data segment



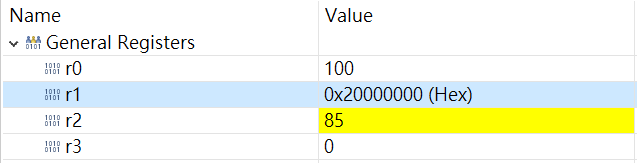
.data: 存放可寫(writeable)且初始化(initialized)的程式碼與資料

.text: 存放唯讀(read-only)的程式碼與資料

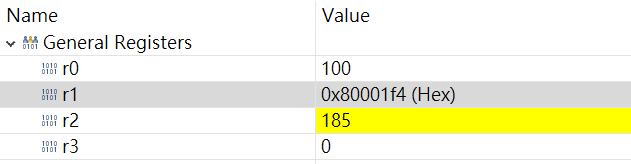
.bss: 存放未初始化的程式碼與資料

Q2: 若將X宣告改在 text section 對其程式執行結果會有何改變？

A2: 程式依舊可以執行，只是X的擺放位置由RAM變成ROM

# 把X宣告在data section，可觀察到位址在0x20000000(RAM)  
 

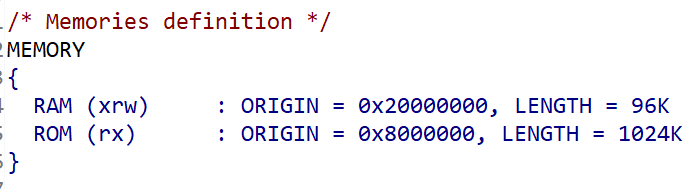
# 把X宣告在text section，可觀察到位址在0x80001f4(RAM)



那麼如何知道這兩個地址分別代表ram和rom呢？

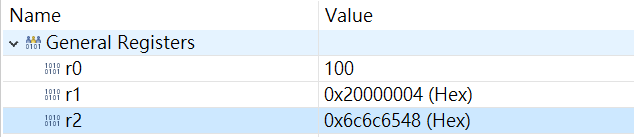
答：

# 從LinkerScript可以知道RAM跟ROM擺放的位址與大小

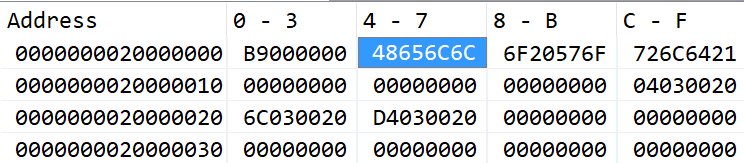


Q3: 程式執行完畢後r2內容與str字串在memory前4個byte呈現內容有何差異？

# 程式執行完畢後r2所存的內容為0x6C6C6548



# str字串在memory前4個byte為0x48656C6C



可觀察到r2內容與str在memory的前4個byte，順序相反

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

**2.3. 簡易算數與基本記憶體指令操作**

這部分實驗需要同學在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。

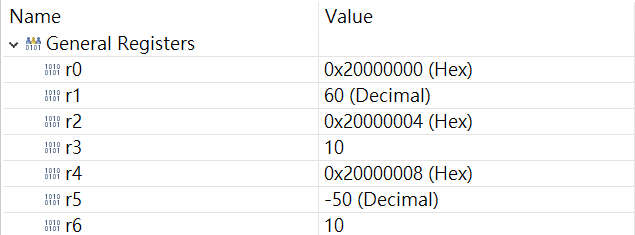
|  |
| --- |
| X = 5  Y = 10  X = X \* 10 + Y  Z = Y - X |

Note: 該程式需使用到算數指令MULS, ADDS, SUBS及記憶體讀寫操作指令LDR, STR

# code

|  |
| --- |
| .syntax unified  .cpu cortex-m4  .thumb  .data  X: .word 5  Y: .word 10  Z: .word 0  .text  .global main  .equ AA, 0x55  main:  # let r0, r2, r4 get the address of X, Y, Z  # make r1 = X's value  # make r3 = Y's value  ldr r0, =X  ldr r1, [r0]  ldr r2, =Y  ldr r3, [r2]  ldr r4, =Z  ldr r5, [r4]  movs r6, #10  muls r1, r1, r6  adds r1, r1, r3  str r1, [r0]  subs r5, r3, r1  str r5, [r4]  L: B L |

# 程式執行完畢後，registers所存的值



r0, r2, r4儲存的是data X, Y, Z的位址(address)

r1, r3, r5則是存data X, Y, Z的值(value)

r6則用來暫存要用來運算的10

最一開始X=5, Y=10 (initialized)

X = X\*10 + Y = 5\*10 + 10 = 60

Z = Y – X = 10 – 60 = -50

分別在r1, r5運算完後，再存回X(0x20000000), Z(0x20000008)的位址

由memory觀察0x20000000, 0x20000008這兩個位址可觀察到相同結果，代表有成功存取

