**實驗一 實驗環境建立與Debugger操作**

**0410001電資08陳宏碩**

**1. 實驗目的**

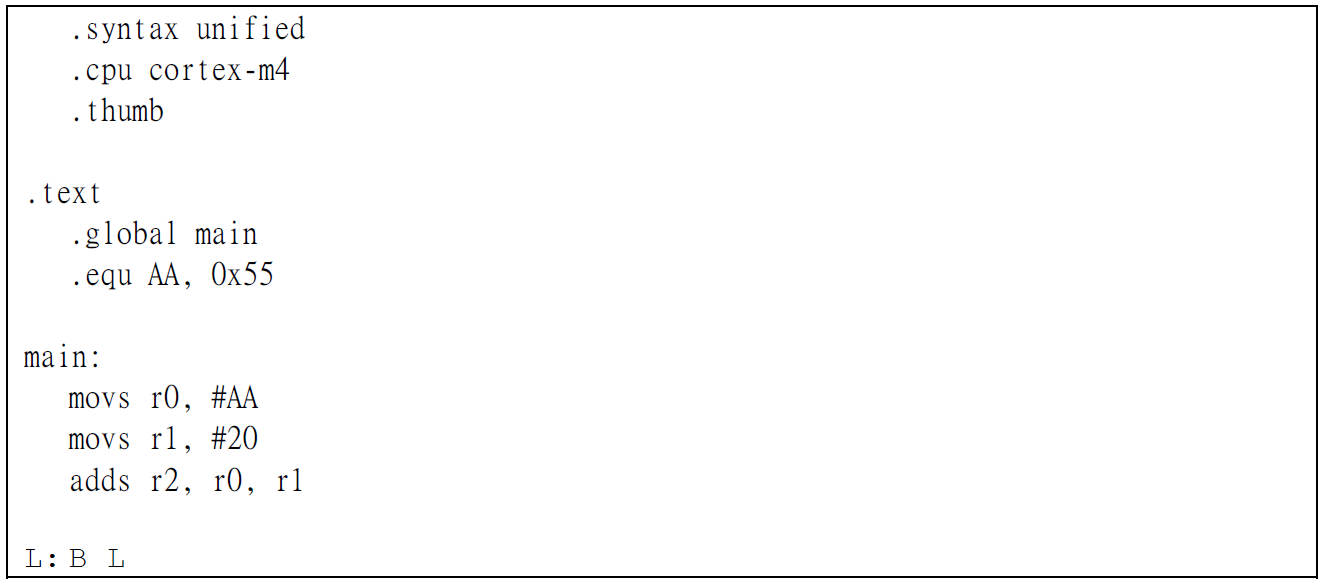
測試實驗器材

熟悉開發環境

**2.實驗步驟**

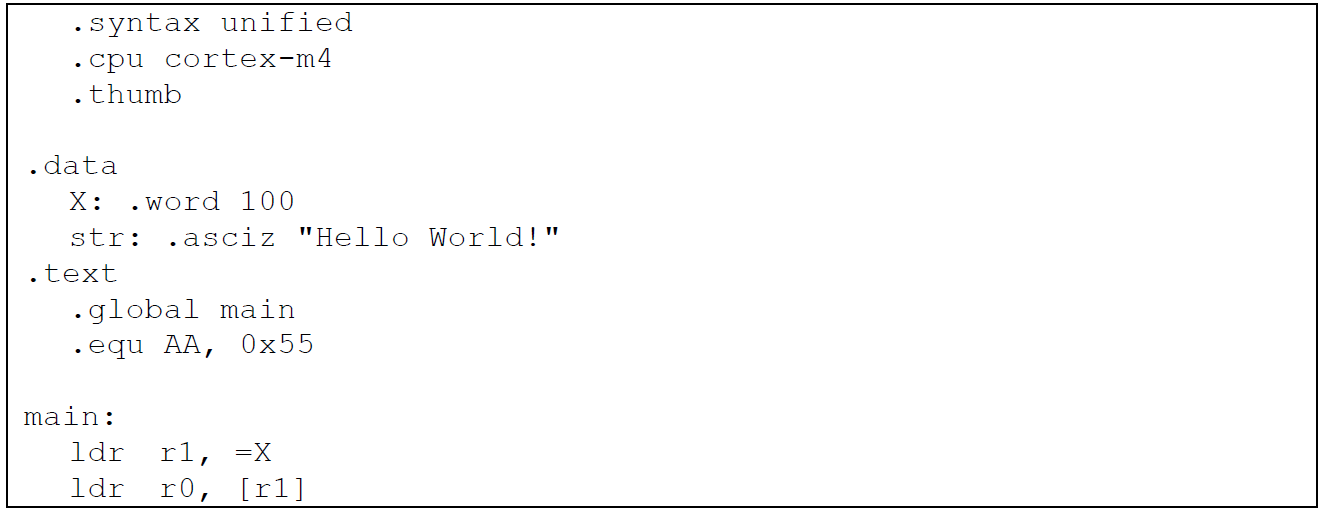
2.1. 專案建立與程式編譯

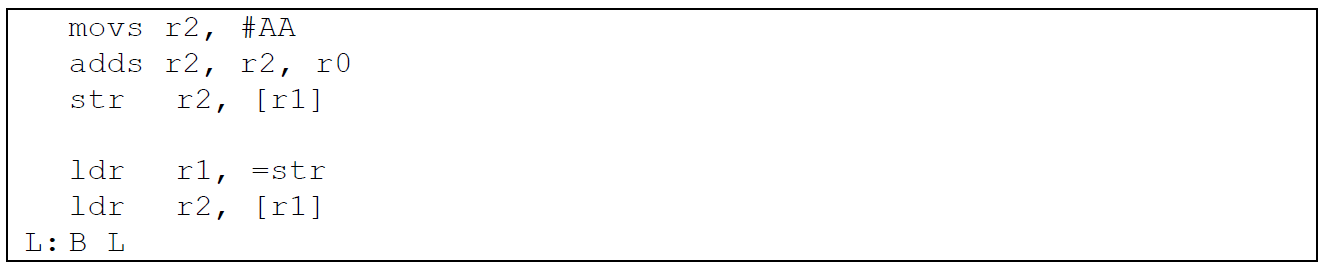
請依照助教給的lab1\_note教學，建立一個STM32 eclipse project，新增一個內容如下的main.s程式碼並透過debuger觀察程式執行結果。



2.2. 變數宣告與記憶體觀察

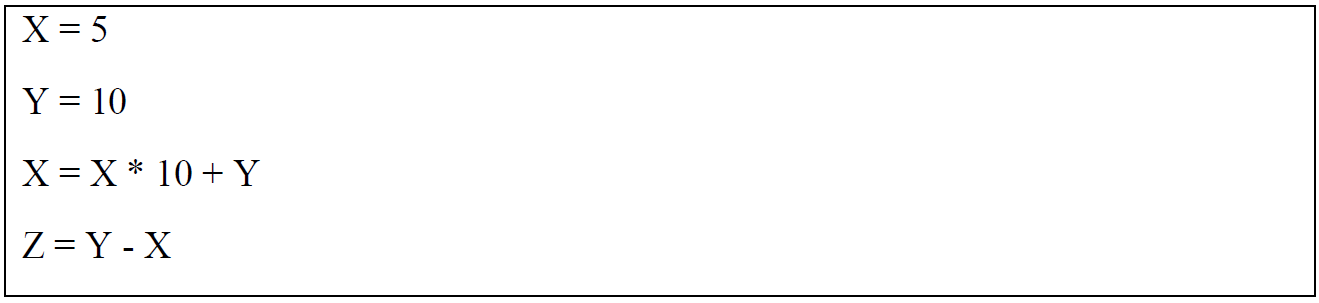
將main.s修改成以下程式碼並編譯執行觀察程式執行結果，並透過memory monitor觀察X內容值變化與回答問題。





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。



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

**3.實驗結果與分析**

2.1. 專案建立與程式編譯

Q: 程式執行結束後R2值為多少？如何觀察？

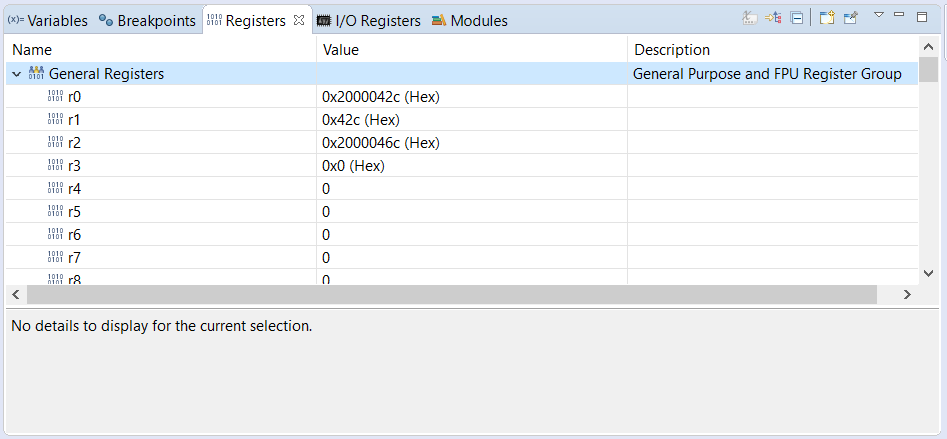
movs r0, #AA // r0 = 0x55, r0 = 85 (∵.equ AA, 0x55)

movs r1, #20 // r1 = 20

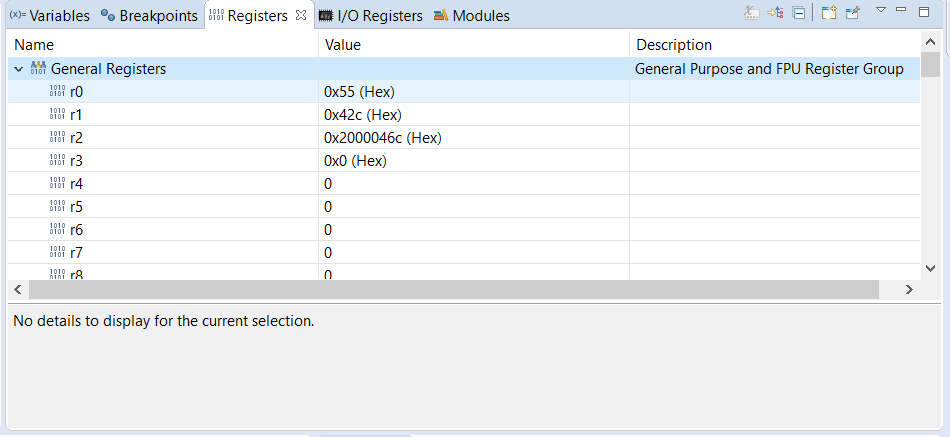
adds r2, r0, r1 // r2 = r1 + r0 = 85 + 20 = 105

故程式執行結束後R2值為105(十進位)或0x69(十六進位)

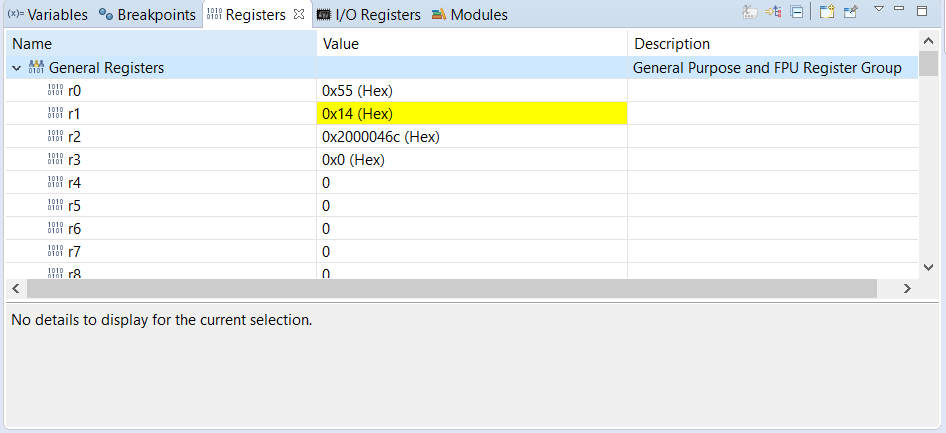
利用Registers Window 我們可以觀察registers的變化，如下圖一到圖四



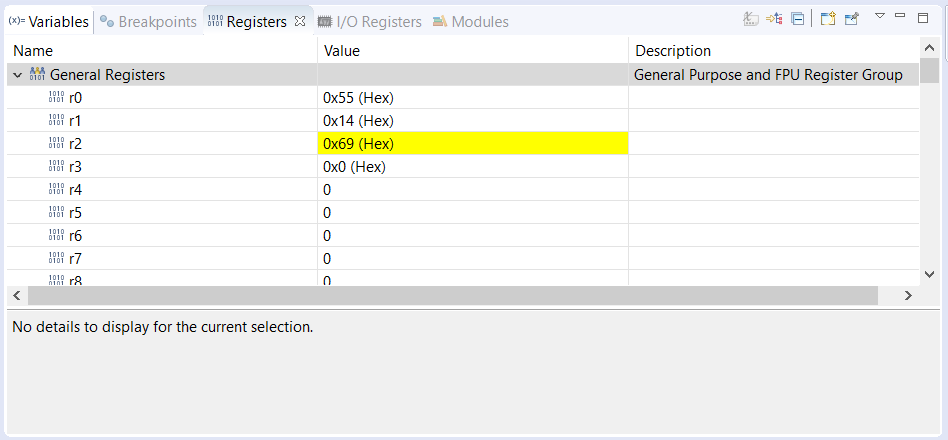
圖一:初始狀態



圖二: movs r0, #AA



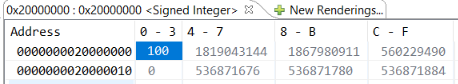
圖三: movs r1, #20



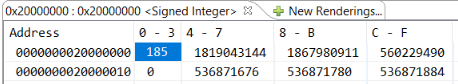
圖四: adds r2, r0, r1，最後的執行結果

2.2. 變數宣告與記憶體觀察

X 的初始值為100(0x64)



執行結束後X變為185(0xB9)



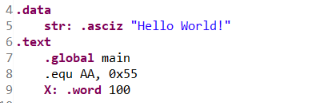
Q1: 變數X與str的初始值是由誰在何處初始化的？

變數X與str的初始值是由Assembler在程式執行前存入memory裡面的data segment的

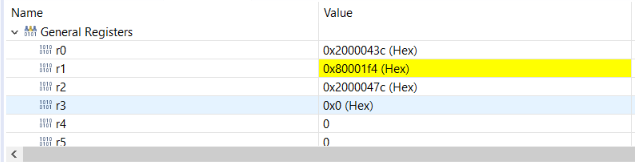
Q2: 若將X宣告改在text section對其程式執行結果會有何改變？

若將X宣告在text section如下圖所示，會造成X的位址從原本的0x20000000變成0x80001f4，而str的位址則向前提前變為0x20000000，原本X的位址，而因為text section只能read不能write，所以str r2, [r1]並不能成功修改X的值。

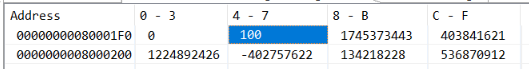
將X宣告改在text section :



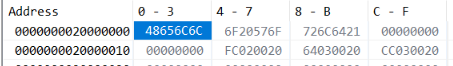
將X的位址存到r1，由register window 可以得知X的位址變為0x80001f4:



由memory monitor 觀察X，最後還是為100:



由memory monitor 觀察str



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？

.data

X: .word 100

str: .asciz "Hello World!"

ldr r1, =X // r1 = the address of X

ldr r0, [r1] // load the value of X into r0

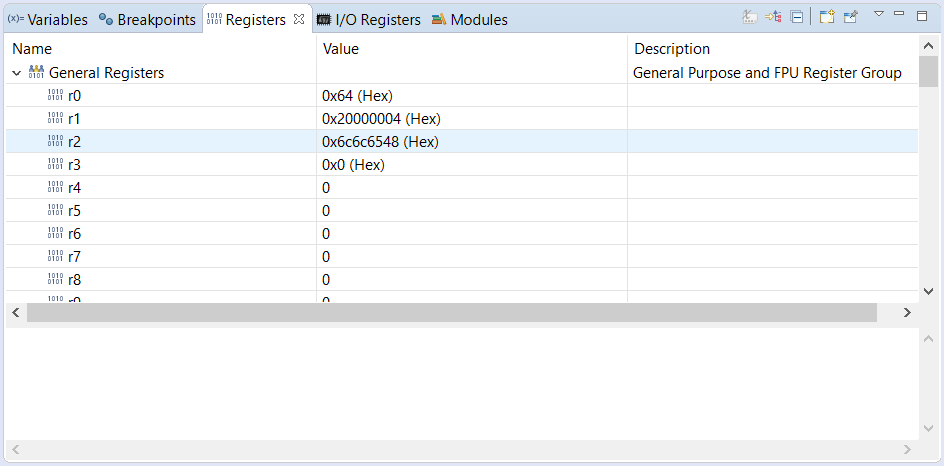
movs r2, #AA // set r2 = 0x55(∵.equ AA, 0x55)

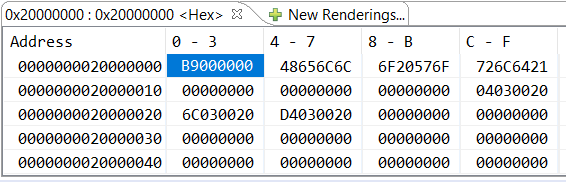
adds r2, r2, r0 //r2 = r2 + r0 = 0x55 + 0x64(100) = 0xB9

str r2, [r1] // store r2 into X

ldr r1, =str // r1 = the address of str

ldr r2, [r1] // load the value of str into r2

依照程式碼，r2內容應與str字串在memory前4個byte相同，但如下圖所示，我們發現兩者的順序稍微不同在register裡0x6c6c6548而在memory裡為48656c6c，兩者順序為相反，但其實兩者是相同的，只是在memory裡的顯示為一次顯示一個byte然後由小的位元顯示到大的位元，故在memory前面的為低位元，而後面的是高位元，我推測可能是因為memory式利用stack的方式存取所以會造成這樣的現象。



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

有，除了使用.asciz外，還可以使用.ascii宣告如下，兩者的最大差別為利用.asciz宣告每一個string都會跟隨一個zero byte，執行結果會相同

str: .ascii "Hello World!"

2.3. 簡易算數與基本記憶體指令操作

計算的組合語言如下:

.syntax unified

.cpu cortex-m4

.thumb

**.data**

**X:** **.word** 5 //declare X as 5

**Y:** **.word** 10 // declare Y as 10

**Z:** **.word** 0 // declare Z as 0

**.text**

**.global** main

**main:**

ldr r2, =X // load the address of X into r2

ldr r3, =Y // load the address of Y into r3

ldr r0, [r2] // load the value of X into r0

ldr r1, [r3] // load the value of X into r0

movs r4, #10 // set r4 to 10

muls r0, r0, r4 // r0 = r0 \* r4 = 5 \* 10 = 50

adds r0, r0, r1 // r0 = r0 + r1 = 50 + 10 = 60

subs r5, r1, r0 // r5 = r1 – r0 = 10 – 60 = -50

str r0, [r2] // store the value of r0 into X

ldr r6, =Z // load the address of Z into r6

str r5, [r6] // store the value of r5 into Z

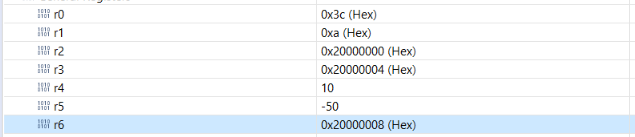
**L:** B L

X 的 memory address 為 0x20000000(r2)

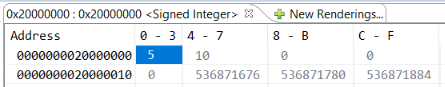
Y 的 memory address 為 0x20000004(r3)

Z 的 memory address 為 0x20000008(r6)

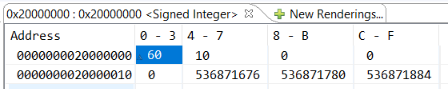
下圖為執行結果的 register window



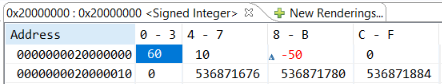
Memory 初始的樣子:

.

將計算過後的X值存入 X = 10\*X + Y = 10\*5 + 10 = 60



將計算過後的Z值存入 Z = Y – X = 10 – 60 = -50



**4.心得討論與應用聯想**

這次的實驗為第一次的實驗，主要在讓我們認識這一塊板子，讓我覺得相當特別的是這塊板子還有debug的功能，這樣會讓之後的實作變相對來說輕鬆，但在實驗過程中，因為還是有許多背後的原理，所以很多的過程和來龍去脈還沒有相當的清楚，可能之後上完課會有更多的收穫。而在看user manual 也發現感覺還有蠻多可以開發的東西，相信之後我們把各個外部介面的零件裝上去後，我們可以做出很有趣的東西，第一次的實驗相當的好，非常容易了解。