



Lab1 Environment setup and Debugger Operation

實驗一 實驗環境建立與Debugger操作

1. Lab objectives 實驗目的

Test the kit.

Familiar with the development environment.

測試實驗器材
熟悉開發環境

2. Steps 實驗步驟

2.1. Project creation and Program compilation 專案建立與程式編譯(20%)

Create a STM32 eclipse project according to lab0. Add a "main.s" code as follows and observe program execution result through debugger.

請依照助教給的lab0教學，建立一個 STM32 eclipse project，新增一個程式碼如下的 main.s 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: BL
```

Q: What is the R2 value after the program is executed ? How to observe ?

程式執行結束後 R2 值為多少？如何觀察？

Name	Value	Description
 General Registers		General Purpose and FP
 r0	536870912	
 r1	536871984	
 r2	536872048	

(執行前)

 General Registers		General Purpose and FP
 r0	85	
 r1	20	
 r2	105	

(執行後)

執行後R2的值為105，可直接在Debug後的register的value部分直接觀察得知



2.2. Variable declaration and Memory observation 變數宣告與記憶體觀察 (40%)

Modify "main.s" into the following code and compile and execute and observe program execution result, and observe the X content value change through the memory monitor. Then answer the question.

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: BL
```

Q1: Where is the initial value of the variables X and str initialized by whom?

變數X與str的初始值是由誰在何處初始化的？

在 ".data" 的地方進行初始化的。

```
5 .data
6 X: .word 100
7 str: .asciz "Hello World!"
```

- Text Segment 又稱為 code segment，一般而言簡稱為 text，基本上就是存放程式指令碼的地方。我們寫的 C 程序在經過編譯器優化、編譯之後，會轉變成只有電腦才看得懂的機器指令碼，這些指令碼會形成最後的執行檔的一部分，並且在程式起跑時載入到記憶體中，存在 text 區。一般而言，text 區都會被系統設定為唯讀 (read only)，以避免被使用者不小心或者刻意修改，形成恐怖的不可預期的致命錯誤。

- Initialized Data Segment 有些變數是一執行就固定位置的，例如全域變數 (global variable)、靜態變數 (static variable)，這些變數如果在程式碼中有被使用者手動初始化 (這邊的手動初始化是指宣告時就順便賦值的初始化，事後自己賦值或者 memset 等等的並不 算)，就會被配置於此區塊。

由下圖的註解得知，初始化時，由flash搬到memory中，防止每次重開都不見。



```
45
46 /* Copy the data segment initializers from flash to SRAM */
47 ldr r0, =_sdata
48 ldr r1, =_edata
49 ldr r2, =_sidata
50 movs r3, #0
51 b LoopCopyDataInit
```

Q2: What happens the program execution result if I change the X declaration to the text section?

若將X宣告改在text section對其程式執行結果會有何改變？

X宣告在 “.data” 的時候 r1的位置為0x20000000(Hex)

General Registers		General Purpose and FP
r0	100	
r1	0x20000000 (Hex)	
r2	185	

X宣告在 “.text” 的時候 r1的位置為0x80001f0(Hex)

Name	Value	Description
General Registers		General Purpose and FP
r0	100	
r1	0x80001f0 (Hex)	
r2	185	

在兩個不同地方宣告的區別在於X擺放的位置從RAM變成ROM

(在LinkerScript.ld 可以找到memory的相關資訊)

```
61 /* Memories definition */
62 MEMORY
63 {
64   RAM (xrw)      : ORIGIN = 0x20000000, LENGTH = 96K
65   ROM (rx)       : ORIGIN = 0x80000000, LENGTH = 1024K
66 }
```

Q3: What is the difference between the r2 content and the str string in the first 4 bytes of memory after the program is executed?

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所存的值為0x6c6c6548

Name	Value	Description
General Registers		General Purpose and FP
r0	100	
r1	0x20000004 (Hex)	
r2	0x6c6c6548 (Hex)	



Str字串在memory前4個byte的值為0x48656c6c

Address	0 - 3	4 - 7	8 - B	C - F
0000000020000000	B9000000	48656C6C	6F20576F	726C6421
0000000020000010	00000000	00000000	00000000	0x20000008

兩者比較可得知，在memory的值與在register的值剛好順序相反(此為16進位，因此兩個數字一組是一個byte，分別為6c,6c,48,65)
因為板子上儲存方式為Little Endian 因此先存在最右邊的位置。



Q4: The variable str "Hello World!" Is there any other way to declare? If there is one, please explain one of them.

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

可以使用：

Str: .ascii , Hello World!'來宣告。兩者
的差別只在於字串後面有沒有

'\0'。

2.3. Simple arithmetic and basic memory instruction operations 簡易算數與基本記憶體指令操作(40%)

This part of the lab requires students to declare three X, Y, and Z variables of length 4 bytes in the data section and calculate the following formula using the ARM assembly language. Find the memory address of these variables and observe the program execution results.

這部分實驗需要同學在 data section 中宣告三個 X, Y, Z 長度為 4 byte 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: This program requires the use of arithmetic instructions MULS, ADDS, SUBS and memory read / write operation instructions LDR, STR

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

main:
ldr r1, =X
ldr r2, [r1]
ldr r3, =Y
ldr r4, [r3]
ldr r5, =Z
ldr r6, [r5]
movs r7, #10
muls r2, r2, r7
adds r2, r2, r4
subs r6, r4, r2









str r2, [r1]
str r6, [r5]
```

L: B L



首先在data section分別宣告X,Y,Z的初始值，因長度須為4byte因此使用word 宣告
之後分別把X,Y,Z的位址存到r1,r3,r5裡面，再用r2,r4,r6來存放X,Y,Z的值，r7則是用
來存放常數10的暫存器。

下圖為程式執行完畢後registers的狀態

Name	Value
 r0	536870912
 r1	0x20000000 (Hex)
 r2	60 (Decimal)
 r3	0x20000004 (Hex)
 r4	10
 r5	0x20000008 (Hex)
 r6	-50
 r7	10

運算如下：

$X = 5, Y = 10, Z = 0$

$X = X * 10 + Y \ (X = 60)$

$Z = Y - X \ (Z = -50)$

運算完後用str把算出來的結果再分別存到原本X,Z的位址。

下圖為memory的存取狀況

Address	0 - 3	4 - 7	8 - B	C - F
0000000020000000	60	10	-50	0
0000000020000010	0	536871676	536871780	536871884
0000000020000020	0	0	0	0
0000000020000030	0	0	0	0
0000000020000040	0	0	0	0

從圖中可以看到在0x20000000(X的位址)的值為60，在0x20000008(Z的位址)的值為-50。