實驗一 實驗環境建立與Debugger操作

0510532 楊上萱

# Lab objectives 實驗目的

測試實驗器材

熟悉開發環境

# Steps 實驗步驟

2.1 專案建立與程式編譯

→執行lab1 2.1的main.s

→透過debugger，利用step into(F5)在registers視窗觀測數值變化

2.2 變數宣告與記憶體觀察

→將main.s修改成lab1 2.2的程式並編譯執行

→設定Objdump讓symbol table能被show在console上

→在console中查看X的memory address

→根據X的memory address在memory monitor中觀察X值變化

2.3 簡易算數與基本記憶體指令操作

→參考2.1、2.2的code

→在data section宣告X為4-bytes並給值5，Y為4-bytes並給值10，Z為4-bytes並給值0

→在main裡面，將X load 到r1，並且讓r1的值load 到r0；將Y load 到r3，並且讓r3的值load 到r2；將 Z load 到r5，並且讓r5的值load 到r4

→r0\*10+r2，並且load進r0

→分別將r0、r2、r4 store回r1(X)、r3(Y)、r5(Z)

→利用symbol table查看X, Y, Z的 memory address，並用memory monitor觀察它們的變化

# Results and analysis 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？

A: R2值為105。利用debugger觀察。

2.2 變數宣告與記憶體觀察

Q1: 變數X與str的初始值是由誰在何處初始化的？

在data section中宣告有初始值的全域變數X和str，程式啟動時會由startup\_stm32.s中的Reset\_Handler區塊進行初始化。

Q2: 若將X宣告改在text section對其程式執行結果會有何改變？

透過memory monitor觀察X，其值在執行後不會被改動。

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？

r2內容值為1819043144[in dec]，換成十六進位制其值為6C6C6548；

str在memory呈現內容為48656c6c，即是r2每個byte的相反。

會有這個結果是因為stm32用little-endian order來store value(LSB存在最低位的memory address)。

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。範例code宣告方式為：str: .asciz “Hello World” (包含\0)，

另一種宣告方式為：str: .ascii “Hello World\0”(因ascii沒有包含\0，

所以手動加上去)。

2.3 簡易算數與基本記憶體指令操作

利用symbol table找出X, Y, Z的memory address，分別為0x20000000, 0x20000004, 0x20000008。透過debugger和memory monitor觀察X, Y, Z的變化：X初始值為0x0005，Y為0x000A，Z為0x0000，經過運算後將r0、r2、r4的值store回r1、r3、r5，而r1、r3、r5已經在main一開始的地方被load X, Y, Z了，因此X, Y, Z也在store後一起被改變內容值，X, Y, Z的值分別為3C[in hex], 0A[in hex], CE[in hex]。

1. **Conclusions and ideas 心得討論與應用聯想**

心得討論：

第一次作業的內容看似簡單，卻讓初次寫ARM Assembly的我們花了許多時間，不過在回答問題的過程中，我們也查了許多資料、不斷的嘗試，因此對stm32和Assembly更加熟悉。

應用聯想：

之後的lab不用再把值刻到LCD上面了，透過debugger和memory monitor，可以更精確地知道值的變化，除錯也能更容易。

程式碼：(lab1\_1/main.c)

|  |
| --- |
| **.**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 |

程式碼：(lab1\_2/main.c)

|  |
| --- |
| **.**syntax unified  **.**cpu cortex**-**m4  **.**thumb  **.**data  X**:** **.**word 100  str**:** **.**asciz "Hello World!"  //str: .ascii "Hello World!\0"  **.**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 |

程式碼：(lab1\_3/main.c)

|  |
| --- |
| /\*  X = 5  Y = 10  X = X \* 10 + Y  Z = Y - X  \*/  **.**syntax unified  **.**cpu cortex**-**m4  **.**thumb  **.**data  X**:** **.**word 5 // X=5, 4 bytes  Y**:** **.**word 10 // Y=10  Z**:** **.**word 0 // Z=0  **.**text  **.**global main  main**:**  ldr r1**,** **=**X // r1 = X  ldr r0**,** **[**r1**]** // r0 = r1(X)  ldr r3**,** **=**Y // r3 = Y  ldr r2**,** **[**r3**]** // r2 = r3(Y)  ldr r5**,** **=**Z // r5 = Z  ldr r4**,** **[**r5**]** // r4 = r5(Z)  movs r6**,** #10  muls r0**,** r0**,** r6 // r0 \*= 10  adds r0**,** r0**,** r2 // r0 += Y  str r0**,** **[**r1**]**  subs r4**,** r2**,** r0 // Z = Y - X  str r4**,** **[**r5**]** // store r4 back to Z  str r2**,** **[**r3**]**  L**:** B L |