實驗二 ARM Assembly I

**0410001電資08陳宏碩**

# 實驗目的

熟悉基本ARMv7組合語言語法使用。

在這次實驗中需要同學了解

* 如何利用條件跳躍指令完成程式迴圈的操作
* 算數與邏輯操作指令使用
* 暫存器(Register)使用與基本函式參數傳遞
* 記憶體與陣列存取
* Random Number Generator 使用 (加分)
* FPU instructions使用 (加分)

# 實驗原理

請參考上課Assembly部分講義。

# 實驗步驟

## Hamming distance

計算兩個數長度為half-word(2bytes)的漢明距離，並將結果存放至result變數中。  
Please calculate the Hamming distance of 2 half-word (2 bytes) numbers, and store the result into the variable “result”.

|  |
| --- |
| .data  result: .byte 0  .text  .global main  .equ X, 0x55AA  .equ Y, 0xAA55  hamm:  //TODO  bx lr  main:  movs R0, #X //This code will cause assemble error. Why? And how to fix.  movs R1, #Y  ldr R2, =result  bl hamm  L: b L |

Note: 漢明距離主要是利用XOR計算兩數bit間差異個數，計算方式可參考下列連結。

Note: Hamming distance is basically using the XOR function to calculate the different number of “bits” of two numbers. Please check the following link for more information.

Reference: <https://en.wikipedia.org/wiki/Hamming_distance>

## Fibonacci serial

宣告一數值N ()，計算Fib(N)並將回傳值存放至R4暫存器  
Declare a number N() and calculate the Fibonacci serial Fib(N). Store the result into register R4.

|  |
| --- |
| .text  .global main  .equ N, 20  fib:  //TODO  bx lr  main:  movs R0, #N  bl fib  L: b L |

Note: 回傳值格式為signed integer，若Fib[N]結果 overflow的話回傳-2, 當N數值出過範圍時fib回傳-1，計算方式可參考下列連結

Note: The returned value should be in signed integer format. If the result of Fib(N) overflows, you should return -2. If the value of N is outside the accepted range, you should return -1. Check the following link for more details of the calculation.

Reference: <https://it.wikipedia.org/wiki/Successione_di_Fibonacci>

## Bubble sort

利用組合語言完成長度為8byte的8bit泡沫排序法。

Please implement the Bubble sort algorithm for the 8 bytes data array with each element in 8bits by assembly.

實作要求：完成do\_sort函式，其中陣列起始記憶體位置作為輸入參數R0，程式結束後需觀察arr1與arr2記憶體內容是否有排序完成。

Implementation Requirement: Fill-in the do\_sort function. The start address of the array is store in the R0 register. Observe the result of arr1 and arr2 in the memory viewer after calling the do\_sort functions. The two arrays should be sorted.

|  |
| --- |
| .data  arr1: .byte 0x19, 0x34, 0x14, 0x32, 0x52, 0x23, 0x61, 0x29  arr2: .byte 0x18, 0x17, 0x33, 0x16, 0xFA, 0x20, 0x55, 0xAC  .text  .global main  do\_sort:  //TODO  bx lr  main:  ldr r0, =arr1  bl do\_sort  ldr r0, =arr2  bl do\_sort  L: b L |

Note: 注意記憶體存取需使用byte alignment指令，例如：STRB, LDRB

Note: The memory access may require the instructions that support byte-alignment, such as STRB, LDRB.

## Monte-Carlo Method for Estimating Pi with FPU and RNG (加分題 10%)(Optional problems with additional 10% score)

透過STM32L476晶片上面的Random Number Generator硬體來產生亂數，並結合FPU使用進一步估算Pi的值

Using the Random Number Generator hardware on STM32L476 to generate numbers for estimating the value of Pi by using the FPU.

**3.4.1 Enabling FPU (Floating Point Unit) and Floating Point Manipulation**

請參考M4 programming manual.pdf 來開啟FPU計算功能，並進行下列運算

Please check the M4 programming manual to enable the functionality of FPU and do the following calculation.

|  |
| --- |
| .syntax unified  .cpu cortex-m4  .thumb  .data  x: .float 0.123  y: .float 0.456  z: .word 20  .text  .global main  enable\_fpu:  //Your code start from here  bx lr  main:  bl enable\_fpu  ldr r0,=x  vldr.f32 s0,[r0]  ldr r0,=y  vldr s1,[r0]  vadd.f32 s2,s0,s1  // Your code start from here  **//Calculate the following values using FPU instructions**  **//and show the register result in your report**  // s2=x-y  // s2=x\*y  // s2=x/y  // load z into r0,  // copy z from r0 to s2,  // convert z from U32 to float representation F32 in s2  // calculate s3=z+x+y  L: b L |

**3.4.2 Random Number Generator**

開啟RNG功能，產生一組(x,y)點在單位平面裡.

Enable the functionality of RNG and generate a sample point in the unit area.

|  |
| --- |
| .syntax unified  .cpu cortex-m4  .thumb  .text  .global main  .equ RCC\_BASE,0x40021000  .equ RCC\_CR,0x0  .equ RCC\_CFGR,0x08  .equ RCC\_PLLCFGR,0x0c  .equ RCC\_CCIPR,0x88  .equ RCC\_AHB2ENR,0x4C  .equ RNG\_CLK\_EN,18  // Register address for RNG (Random Number Generator)  .equ RNG\_BASE,0x50060800 //RNG BASE Address  .equ RNG\_CR\_OFFSET,0x00 //RNG Control Register  .equ RNGEN,2 // RNG\_CR bit 2  .equ RNG\_SR\_OFFSET,0x04 //RNG Status Register  .equ DRDY,0 // RNG\_SR bit 0  .equ RNG\_DR\_OFFSET,0x08 //RNG Data Register (Generated random number!)  //Data Settings for 3.4.4  .equ SAMPLE,1000000  set\_flag:  ldr r2,[r0,r1]  orr r2,r2,r3  str r2,[r0,r1]  bx lr  enable\_fpu:  //Your code in 3.4.1    bx lr  enable\_rng:  //Your code start from here  //Set the RNGEN bit to 1    bx lr  get\_rand:  //Your code start from here  //read RNG\_SR  //check DRDY bit, wait until to 1  //read RNG\_DR for random number and store into a register for later usage  bx lr  main:  //RCC Settings  ldr r0,=RCC\_BASE  ldr r1,=RCC\_CR  ldr r3,=#(1<<8) //HSION  bl set\_flag  ldr r1,=RCC\_CFGR  ldr r3,=#(3<<24) //HSI16 selected  bl set\_flag  ldr r1,=RCC\_PLLCFGR  ldr r3,=#(1<<24|1<<20|1<<16|10<<8|2<<0)  bl set\_flag  ldr r1,=RCC\_CCIPR  ldr r3,=#(2<<26)  bl set\_flag  ldr r1,=RCC\_AHB2ENR  ldr r3,=#(1<<RNG\_CLK\_EN)  bl set\_flag  ldr r1,=RCC\_CR  ldr r3,=#(1<<24) //PLLON  bl set\_flag  chk\_PLLON:  ldr r2,[r0,r1]  ands r2,r2,#(1<<25)  beq chk\_PLLON  //Your code start from here  //Enable FPU,RNG  //Generate 2 random U32 number x,y  //Map x,y in unit range [0,1] using FPU  //Calculate the z=sqrt(x^2+y^2) using FPU  //Show the result of z in your report  L: b L |

**3.4.3 Estimation of Pi**

使用Monte Carlo Method來估算Pi的值

Using Monte Carlo Method to estimate the value of Pi.

Note:

1. Report中請至少附上三次使用一百萬個點估算完的Pi值的register結果截圖

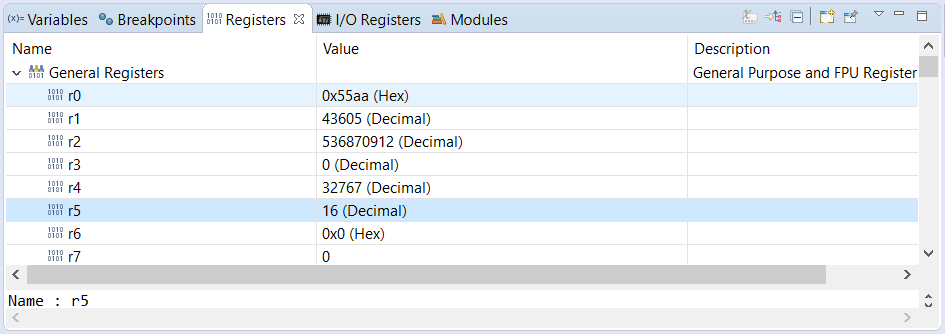
Please attach the screenshots of the register for at least 3 estimation results using 1 million sample points.

1. 請使用3.4.2的程式模板進行修改，以避免修改到RCC設定影響RNG功能

Please use the code template provided in 3.4.2 for this problem. RNG may raise error if the settings of RCC are incorrect.

Reference : http://www.eveandersson.com/pi/monte-carlo-circle

# 實驗結果與分析

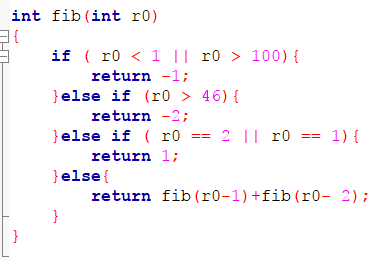
1. 

將之先存在r5並存入result並得0xAA55與0x55AA的距離為16

2.

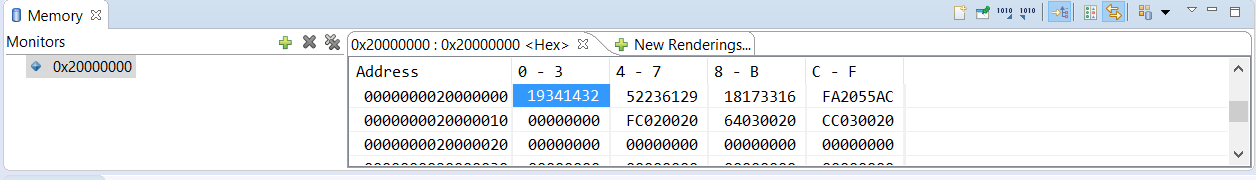


Fib(10) = 55

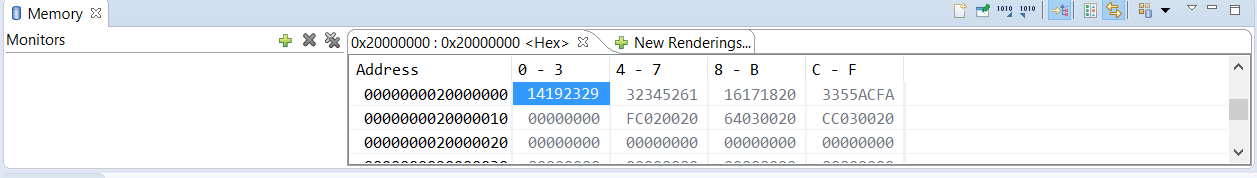


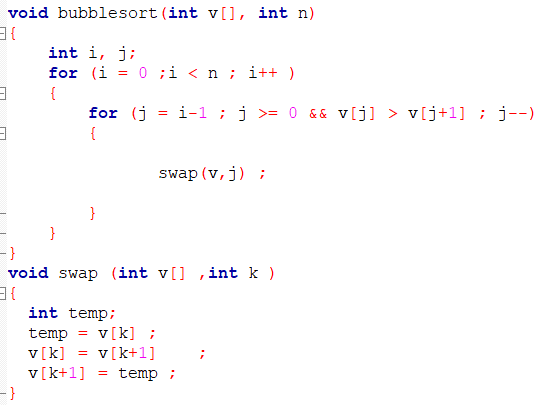
3.

Before sort



After sort

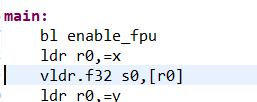




4.

Q3.4.1.1: 如果enable\_fpu留空，程式會停在哪裡？為什麼？

If the enable\_fpu function is empty in the above code, where will the program stop at and why?

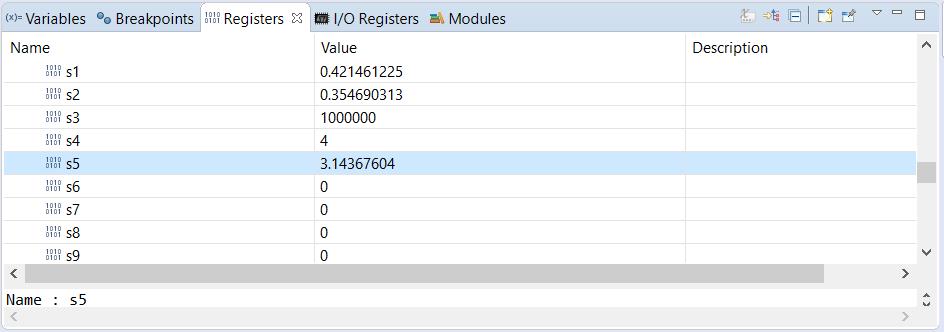
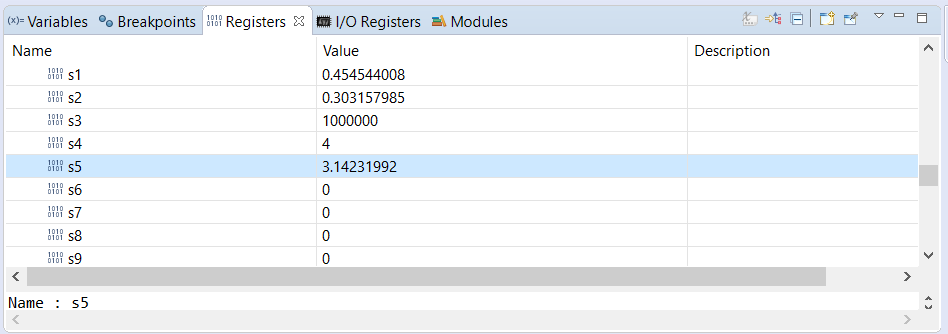


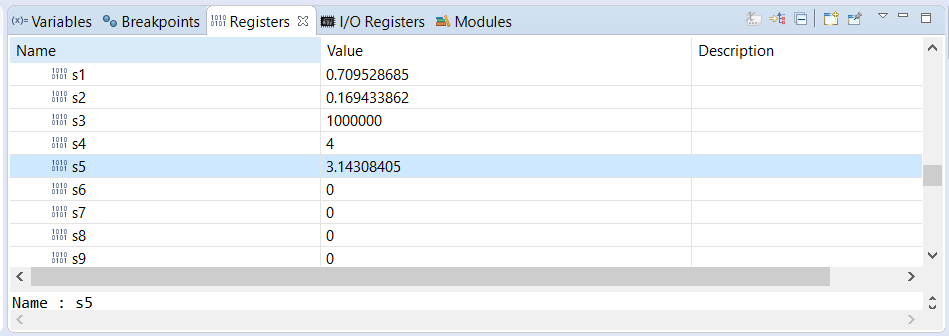
會停在的第一個有用到浮點數的指令，因為FPU還沒啟動

Q3.4.1.2: 為什麼需要將U32轉成F32格式再相加？如果想直接load 一個值代表20到s2中不需轉換就能運算，應該將z修改成多少才能得到相同答案？

Why do we need to convert the U32 to F32 format before the addition? If we want to directly load a value represents 20 for calculation without further format conversion, what value should we modify to z in order to get the same answer?

因為浮點數的格式與整數的格式不相同，需要轉換，不然所得的值會是錯的，需將之修改為z: .float 20.0





# 心得討論與應用聯想

此次實驗主要是讓我們練習利用 assembly language來實作條件判斷、迴圈 的操作，以及簡單的函式呼叫 。基本上所有的程式都是用這些條件判斷與迴圈所建構，因此能熟練運用此次實驗的技巧就能實作出大部分我們所需的程式。