${Lab name 實驗名稱}

0411306 范鑫

1. Lab objectives 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

1. Steps 實驗步驟

2.1 movs R0, #X //This code will cause assemble error. Why? And how to fix.

改成ldr R0, =X ,因為Ｘ值太大 無法直接movs給r0,所以改成存取此位址的值就可以了

使用xor得到兩數之間有哪些位元是不同的 然後用一個1 去做 shift 依次變成10, 100, 1000,10000....然後跟剛剛的值做and 如果and結果不為0則漢明距離+1

2.2 先判斷Ｎ是否為1,2或 大於100設定回傳值

先宣告fib數列的前兩個數字F1,F2 然後用一個暫存器 存兩數的合F3,然後 把F2給F1,F3給F2,之後繼續計算直到碰到需要的個數為止,或者觸發了overflow即可

2.3 重點在於使用“ldrb , strb” 來存取和儲存一個byte的值,

之後就只要照演算法去把排序法寫出來即可

1. Results and analysis 實驗結果與分析

2.1

漢名距離為16 (每個位元都不一樣）

2.2

Fib(20)=4181

2.3

array1: 14 19 23 29 32 34 52 61

array2: 16 17 18 20 33 55 AC FA

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

這次實驗讓我們學會了迴圈如何在組語言實現以及如何使用條件和非條件式跳躍指令,一開始對於迴圈的使用非常的陌生,但努力完成了第一個迴圈之後,思路就變得比較清晰了

5. Code

1.

————————————————————————————————————————————————————————————

.data

result: .byte 0

.text

.global main

.equ X, 0x55AA

.equ Y, 0xAA55

hamm:

ldr R3,[R2] // R3 =result

movs R4 ,R0 // R4 = x

movs R5 ,R1 // R5 = y

movs R6 ,#1 // R6 = 1

movs R0 ,#1 // int cont =1

eor R4,R5 // R4 = R4 ^ R5

loop:

movs R7 , R4 // R7 = R4

and R7 , R6 // R7 = R7 & R4

cmp R7 , #0 // R7 == 0 ?

bne add\_one // go to add\_one

loop\_2:

lsl R6, #1 // R6 << 1

cmp R0 , #16 // cont == 16?

beq exit // exit

add R0 , #1// R8= R8+1

b loop //迴圈繼續

exit:

str R3,[R2] // result = R3

bx lr

add\_one:

add R3,#1 //result = result +1

b loop\_2 // back to loop

main:

ldr R0, =X //This code will cause assemble error. Why? And how to fix.

ldr R1, =Y

ldr R2, =result

bl hamm

L: b L

————————————————————————————————————————————————————————————

2.

————————————————————————————————————————————————————————————

.text

.global main

.equ N, 20

fib:

cmp R0, #100 // (if N>100)

bhi exit3 //回傳-1

/\*movs R7,#0//if N < 0

sub R7,R7,#1

cmp R7 , #0

bcc exit3\*/

cmp R0 , #1 //if( N== 1 )

beq exit1 //回傳0

cmp R0 , #2 //if( N== 2 )

beq exit2 //回傳1

movs R6 ,#3 // int i=3;

movs R4 ,#0// R4 =0

movs R1,#0 // F1 =0

movs R2,#1 // F2 =1

loop:

add R4 , R1 ,R2 //F3=F1+F2

bvs exit4 // overflow?

movs R1 ,R2 //F1=F2

movs R2 , R4 //F2=F3

cmp R0 , R6 //if( i >= r0 )

beq exit //結束

add R6 , R6 , #1 // i+=1

b loop //迴圈繼續

exit:

bx lr

exit1:

movs R4,#0

bx lr

exit2:

movs R4,#1

bx lr

exit3:

movs R4 , #0

sub R4,R4,#1

bx lr

exit4:

movs R4 , #0

sub R4,R4,#2

bx lr

main:

movs R0, #N

bl fib

L: b L

————————————————————————————————————————————————————————————

3.

————————————————————————————————————————————————————————————

.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:

nextloop:

add r2 , #1 // r2= r2+1

ldrb r1 , [r0,r2]//r1=a[r2]

cmp r2 , #8 //r2==8?

beq exit // exit

movs r5 , r2// r5 =r2

compare:

ldrb r3 , [r0,r5]//r3 = a[r5]

cmp r1 , r3 //r1>r3?

bhi change //change r1 r3

continue:

add r5 , #1 // r5 = r5+1

cmp r5 ,#8 // r5 == 8?

beq nextloop // next loop

b compare // 迴圈繼續

change: //r1 和 r3 互換

movs r4,r1 //tmp = r1

movs r1,r3 // r1 = r3

movs r3,r4 // r3 = tmp

strb r1 ,[r0,r2]//存回記憶體

strb r3 ,[r0,r5]//

b continue

exit:

bx lr

main:

movs r2,#0//r2 = 0

sub r2 , #1// r2 = -1

ldr r0, =arr1

bl do\_sort

movs r2,#0

sub r2 , #1

ldr r0, =arr2

bl do\_sort

L: b L

————————————————————————————————————————————————————————————