HW3

程式目標：

a matrix \* b matrix = c matrix

程式想法：

第一部份-----> data section

將a矩陣存入=[1,2,3,4

5,6,7,8]

開32byte,每一個位置存入一個word

將b矩陣存入=[1,2,

3,4,

5,6,

7,8]

開32byte,每一個位置存入一個word

將c矩陣存入=[0,0,

0,0]

開16yte,每一個位置存入一個word

第二部份-------> text section

r0,r1,r2分別存入a,b,c的address（目地：為了方便獲取a,b,c陣列的位置）

r3,r4,r5類似pointer的概念,看這次是分別計算那一個row或col,並指向那些位置

r6為做乘法的次數（初始為0,最多到4）

接著跑進save\_return\_address 這個subroutine(目的：存入返回得地址-->用bl這個指令)

接著利用cal\_mul這個subroutine(目的：計算行列相乘和相加)

將值重新存入r1

範例：

/\* 看這次是分別計算那一個row或col,並指向那些位置 \*/

mov r6, #0

ldr r3, =a

ldr r4, =b

ldr r5, =c

bl save\_return\_address

/\* 存入返回得地址-->用bl這個指令 \*/

save\_return\_address:

stmfd r13!, {r14} -->將返回得地址存入r13(stack pointer)

bl cal\_mul -->做乘法以及加法計算

ldmfd r13!, {pc} -->將stack pointer中的地址pop出,並給r15(program counter）

/\* 計算行列相乘和相加 \*/

cal\_mul:

ldr r7, [r3], #4 ------> 因為 a matrix 是 row 每一次都前進一個位置（4個byte因此加4）, r7=[r3] , r3=r3+4

ldr r8, [r4], #8 ------> 因為 b matrix 是 col 每一次都前進兩個位置（8個byte因此加8）, r8=[r4] , r4=r4+8

mul r9, r7, r8 ------> 相乘後把結果存入r9

ldr r10,[r5] ------> 把[r5]存入到r10, 這時的r5是目前相加的結果

add r10, r9 ------> 把r10=r10(目前相加的結果)+r9(新相乘的到的結果）

str r10, [r5] ------> 把新的結果在存入r5

add r6, #1 ------> 把做的次數（loop 的次數）加1

cmp r6, #4 ------> 判斷r6是否等於4

blt cal\_mul ------> 小於4就重新跑-cal\_mul迴圈

moveq pc, r14 ------> 等於4就跑回save\_return\_address

編譯指令：

./arm-none-eabi-gcc -g hw3.s -o hw3.exe

心得：

一開始對組合語言的指令非常不熟悉,因為考了期中,沒想到就莫名其妙會了好多指令,(ldr,bl,moveq.....),因為有了這些基礎,寫程式時就快了很多！

而且中最令我覺得有趣的部份是

stmfd r13!, {r14}

ldmfd r13!, {pc}

moveq pc, r14

因為在c和c++不會有找不到返回位址的問題,但組合語言還要特地把r14存到r13stack裡,再把它一到r15告訴他下一到指令的位置,超酷！