實驗三 ARM Assembly II

0410001 電資 08 陳宏碩

1. 實驗目的

熟悉基本 ARMv7 組合語言語法使用。

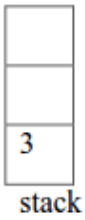
2. 實驗步驟

2.1. Postfix arithmetic

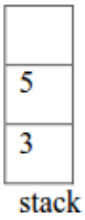
操作 stack 來完成 postfix 的加減法運算

2.1.1. Example: 3, 5, 4, -, +

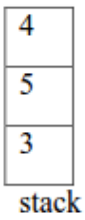
(1) 3, 5, 4, -, +



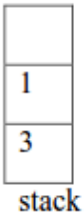
(2) 3, 5, 4, -, +



(3) 3, 5, 4, -, +



(4) 3, 5, 4, -, +



(5) 3, 5, 4, -, +

2.1.2. 實作要求

完成以下的程式碼，必須要利用 PUSH,POP 操作 stack 來完成 postfix expression 的運算，並將結果存進 expr_result 這個變數裡。

```
   .syntax unified
   .cpu cortex-m4
   .thumb

.data
   user_stack  .zero 128
   expr_result .word   0

.text
   .global main
   postfix_expr    .asciz     "-100 10 20 + - 10 +"

main:
   LDR  R0, =postfix_expr

//TODO: Setup stack pointer to end of user_stack and calculate the
expression using PUSH, POP operators, and store the result into
expr_result

program_end:
   B    program_end

atoi:
    //TODO: implement a "convert string to integer" function
    BX LR
```

postfix_expr 格式：postfix_expr 是一串 postfix 運算式的字串，每個數字/運算子之間會用 1 個空白來區隔；input 的數字是 10 進位整數，數字正負數皆支援，字串以 ascii value 0 作為結尾；可以假設此運算式必可求出解。

Prototype of atoi:

Input : start address of the string (using register)

Output : integer value (using register)

Hint:可以利用 MSR 來修改 MSP(Main Stack Pointer)的值

Reference:
http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0489f/CIHFIDAJ.html

http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0497a/CHDBIBGJ.html

2.2. 求最大公因數並計算最多用了多少 stack size

在程式碼中宣告 2 個變數 m 與 n ，並撰寫 Stein 版本的最大公因數，將結果存入變數 result 裡，請使用 recursion 的寫法，並使用 stack 傳遞 function 的 parameters，禁止單純用 register 來傳。

計算在 recursion 過程中，記錄最多用了多少 stack size，並將它存進 max_size 這個變數中。

```
.data
    result: .word  0
    max_size:  .word  0
.text
    m: .word  0x5E
    n: .word  0x60

GCD:
     //TODO: Implement your GCD function
   BX LR
```

Prototype of GCD:

Input : A,B (using stack)

Output : GCD value (using register), max stack size (using register)

```
   MOVS    R0, #1;
   MOVS R1, #2
   PUSH {R0, R1}
   LDR  R2,  [sp]     // R2 = 1
   LDR  R3, [sp, #4]  //R3 = 2
   POP     {R0, R1}
```

Reference:

GCD Algorithm（Euclid & Stein）：

http://www.cnblogs.com/drizzlecrj/archive/2007/09/14/892340.html

3. 實驗結果與分析

3.1. Postfix arithmetic

如下是 atoi 的演算法，左圖為 C++，右圖為 ASM

```
int atoi(char* p){
    int num = 0;
    int minus = 1;
    while(*(p)!=' '){
        num*=10;
        if (*p == '-'){
            minus = -1;
        }else {
            num += *p-'0';
        }
        p++;
    }
    num = num*minus;
    return num;
}
```

```
atoi://TODO: implement a "convert string
     //r3:num r4:minus r1:p r2:*p r5 10
    movs r3, #0
    movs r4, #1
while:
    ldrb r2, [r1]
    cmp r2, #32
    beq exit
    movs r5, #10
    muls r3, r3, r5
    cmp r2, #45
    beq L1
    bne L2
L1:
    movs r4, -1
    b L
L2:
    adds r3, r3, r2
    subs r3,r3, #48
L:  adds r1,r1, #1
    b while
exit:
    muls r3, r3, r4
    adds r1, #1
    BX LR
```

0000000020000070	0	20	10	-120
0000000020000080	-120	0	0	5...

實驗結果

當 stack 是空的時候 sp 會指向 0x20000080 當我們 push 一個數值的時候會向上加 4(ex:0x2000007C 第一個 stack 的位址)，因為一個 word 是四個 bytes，最後我們答案 pop 出來並存入 expr_result(0x20000080)，如圖所示。

紀錄變化過程

0000000020000050	0	0	0	0
0000000020000060	0	0	0	0
0000000020000070	0	0	0	-100

-100

0000000020000060	0	0	0	0
0000000020000070	0	20	10	-100

10 和 20

0000000020000060	0	0	0	0
0000000020000070	0	20	30	-100

更新 10+20=30

0000000020000060	0	0	0	0
0000000020000070	0	20	30	-130

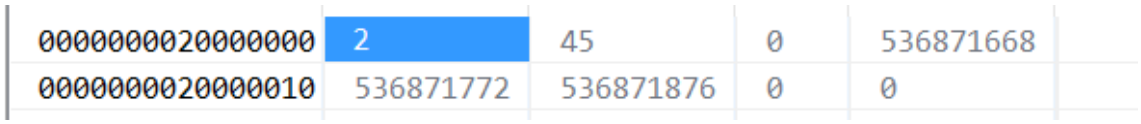
-100-30=-130

0000000020000060	0	0	0	0
0000000020000070	0	20	10	-130

-130+10=-120

0000000020000060	0	0	0	0
0000000020000070	0	20	10	-120
0000000020000080	-120	0	0	536871796

3.2. 求最大公因數並計算最多用了多少 stack size



0000000020000000	2	45	0	536871668
0000000020000010	536871772	536871876	0	0

實驗結果

96 和 94 的最大公因數為 2’

Gcd(96,94) = 2*gcd(48,47) = 2*gcd(24,47)=2*gcd(12,47)=2*gcd(6,47)=2*gcd(3,47)=2*gcd(44,3)=2*gcd(22,3)=2*gcd(11,3)=2*gcd(8,3)=2*gcd(5,3)=2*gcd(2,3)=2*gcd(1,3)=2*gcd(2,1)=2*gcd(1,1)=2*gcd(0,1)=2

呼叫了 15 次的 gcd function，一次 push 3 個 word，故會用到 3*15=45words 的 size

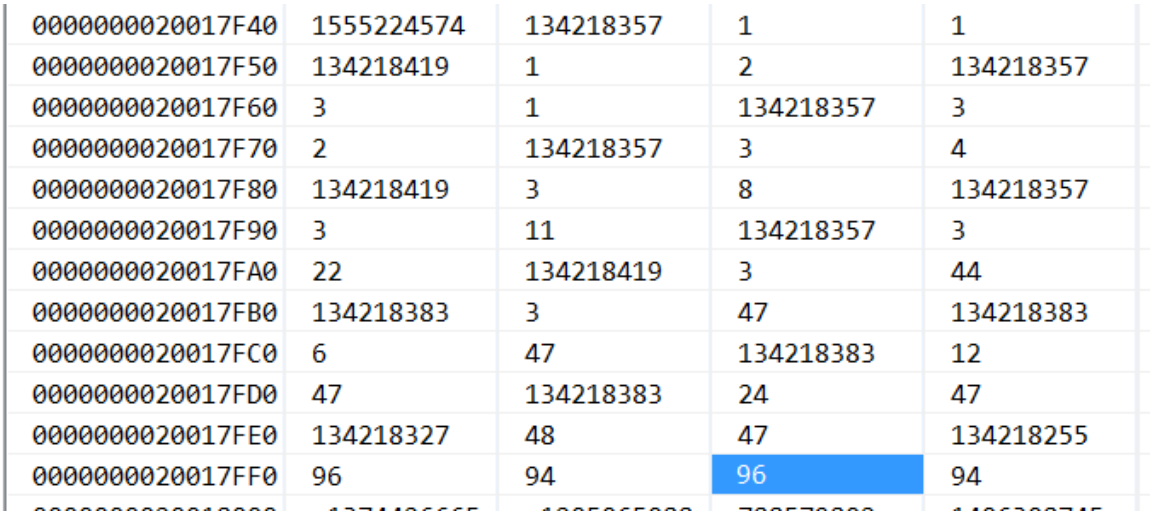
如上圖所示

而在每一次呼叫函數的時候都需要將值記錄下來，這樣才不會因為在函數裡呼叫函數的時候，更改到值而導致計算錯誤或無法回到原函數的地方。

```
push {r0}
push {r1}
push {lr}
```

pop 的順序需要倒過來，如下

```
pop {lr}
pop {r1}
pop {r0}
```



0000000020017F40	1555224574	134218357	1	1
0000000020017F50	134218419	1	2	134218357
0000000020017F60	3	1	134218357	3
0000000020017F70	2	134218357	3	4
0000000020017F80	134218419	3	8	134218357
0000000020017F90	3	11	134218357	3
0000000020017FA0	22	134218419	3	44
0000000020017FB0	134218383	3	47	134218383
0000000020017FC0	6	47	134218383	12
0000000020017FD0	47	134218383	24	47
0000000020017FE0	134218327	48	47	134218255
0000000020017FF0	96	94	96	94

我並沒有更改 sp 原本所指定的地方，總共有 45 個 words 曾經被存入 stack，和上面敘述相符，驗證了我的假設。

4. 心得討論與應用聯想

這次的實驗運用到了 stack pointer，stack 是在程式裡面相當重要的一個東西，許多的程式都需要運用到這個資料結構，而這次運用在簡單的 gcd 和 Postfix arithmetic 讓我們練習，我覺得相當的好，在寫機器語言的時候須要考慮到很多記憶體的配置，但也能讓我們更加了解程式的運作。