實驗三 ARM Assembly II

**0410001電資08陳宏碩**

# 實驗目的

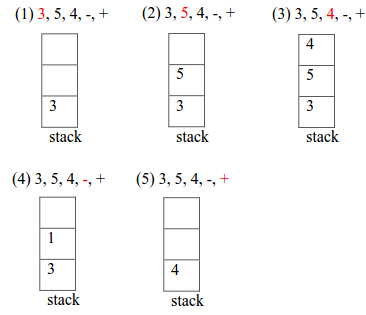
熟悉基本ARMv7組合語言語法使用。

# 實驗步驟

## Postfix arithmetic

操作stack來完成postfix的加減法運算

### Example: 3, 5, 4, -, +



### 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>

## 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>

# 實驗結果與分析

## Postfix arithmetic

如下是atoi的演算法，左圖為C++，右圖為ASM

|  |  |
| --- | --- |
|  |  |



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



-100



10和20



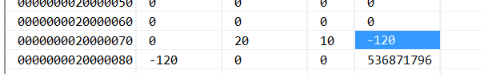
更新10+20=30



-100-30=-130



-130+10=-120



## 求最大公因數並計算最多用了多少stack size



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}

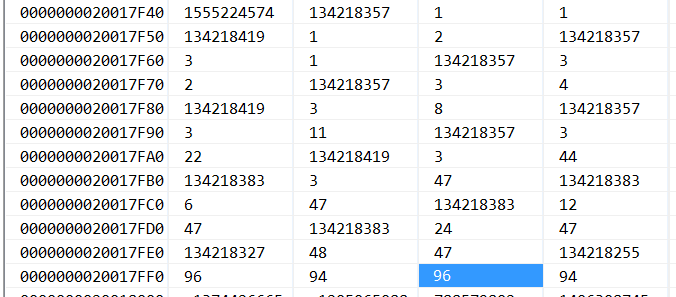
push {lr}

pop的順序需要倒過來，如下

pop {lr}

pop {r1}

pop {r0}



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。

# 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。