實驗三 ARM Assembly II

1. 實驗目的

熟悉基本 ARMv7 組合語言語法使用。

2. 實驗原理

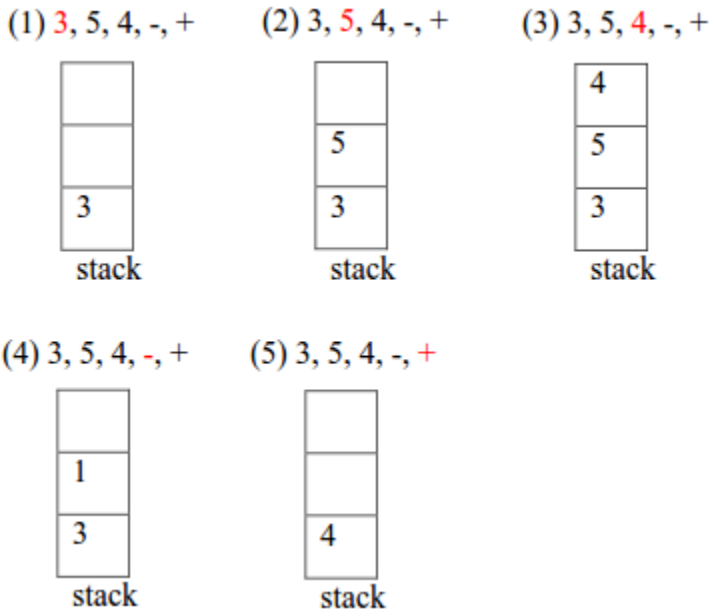
請參考上課 Assembly 部分講義。

3. 實驗步驟

3.1. Postfix arithmetic

操作 stack 來完成 postfix 的加減法運算

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



3.1.2. 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

strlen:
   //TODO: implement a "calculate the string length" function
   BX LR
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 ;Demo 時助教測資格式可能會有 error 請注意處理

Prototype of strlen:

Input : start address of the string (using register)

Output : string length (using register)

Prototype of atoi:

Input : start address of the string (using register)

Output : integer value (using register)

Requirement: 字串支援正負數,字串只需支援 10 進位,支援 error handling

Error handling: 字串格式錯誤的話,expr_result 存-1,程式跳至 program_end

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

Note:助教會在 demo 時修改 postfix_expr 數值

3.2. 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(reference 上有演算法),將結果存入變數 result 裡, 請使用 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)

Hint: stack 的操作

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

Note : 助教會在 demo 時修改 m, n 數值

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