

微處理機 Lab3 Report

- 一、 實驗名稱：ARM Assembly II
- 二、 實驗目的：熟悉基本 ARMv7 組合語言語法的使用。
- 三、 實驗步驟：
 1. Postfix arithmetic。
 2. 求最大公因數並計算最多用了多少 stack size。
- 四、 實驗結果與分析：

1. Postfix arithmetic：

```

1  .syntax unified
2  .cpu cortex-m4
3  .thumb
4  .data
5  user_stack: .zero 128 //Means there will be
6  expr_result: .word 0
7  .text
8  .global main
9  postfix_expr: .asciz "-100 10 20 + - 10 +"
10 .align 4
11 calculation:
12 pop {r2,r5} //r5: opd 1, r2: opd 2
13 cmp r3,0x20
14 IT EQ
15 subeq r5,r5,r2
16 IT NE
17 addne r5,r5,r2
18 push {r5}
19 b read
20
21 program_end:
22 nop
23 B program_end
24
25 main:
26 ldr r0,=postfix_expr
  
```

Expression	Type	Value
expr_result	<data variable, no debug info>	-120
user_stack	<data variable, no debug info>	0
Add new expression		

這題可以拆成兩個部分來說明，第一部份就是計算這部分，這部分其實不難，就只是做 pop 並根據相對應的運算子來算出結果後 push 回 stack。

我們覺得比較難的反而是前置的字串處理，因為是要用組合語言來做字串處理，做起來其實有點綁手綁腳。現在說明一下我們是如何做字串處理的，我們有用一個暫存器來當作計數器，作用是告訴我們下一個字在哪

裡，每讀完一個字後計數器就加 1，詳細過程如下：

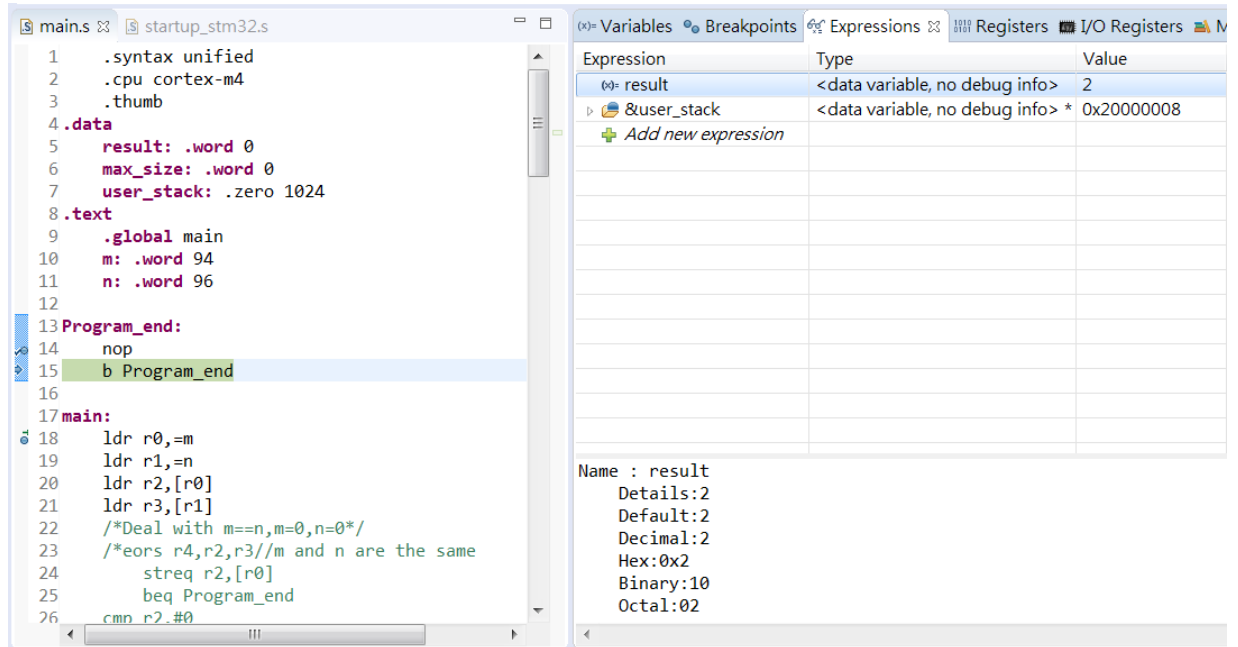
- 讀到'\0'，表示字串結束，stack pop 一次，把 pop 出來的值存入 `expr_result` 這個變數，程式結束。
- 讀到空白則忽略不看，繼續讀下去。
- 讀到負號，就必須再多讀一個字來幫我們判斷該走哪一步：
 - i. 若再讀到空白，表示此負號為一個運算子，跳入 `calculation` 計算並存入 `stack`。
 - ii. 若再讀到數字，先紀錄此數字為負數，跳入 `atoi` 把此負數給 `parse` 出來並存入 `stack`。
 - iii. 若再讀到'\0'，表示此負號為一個運算子且它位於字串最後，計數器減 1，跳入 `calculation` 計算並存入 `stack`。
- 讀到加號，表示為一個運算子，跳入 `calculation` 算出結果並存入 `stack`。
- 讀到屬於數字範圍的字，跳入 `atoi` 把該非負整數 `parse` 出來並存入 `stack`。

`atoi` 的實作：

- 若讀到屬於數字範圍的字，把先前存好的值放大 10 倍加上該次讀到數字的值，接著再繼續讀下去。
- 若讀到空白，代表已經把該運算元 `parse` 完畢，把其推進 `stack` 後跳回主要程式區段。

- 若讀到'\0'，表示先前給定的表達式就只有一個數字，將已 parse 完的運算元推進 stack，接著把計數器減 1 後跳出回主要程式區段。

2. 求最大公因數並計算最多用了多少 stack size：



這題的做法基本上就跟 wiki 上的 C-Recursion 的做法差不多，只是要想辦法把 C code 翻譯成組合語言，其實還蠻不好翻的，有些 if 的判斷式都要以很多的組語才能完成。

在實作過程我們覺得有幾個比較值得注意的地方：

- 第一個是該怎麼算出那個最大公因數，我們是拿 r0 來存最大公因數，r0 初始值為 1，當兩數確認皆為偶數後，將 r0 左移一位(將紀錄有 k 次雙偶數最後再乘以 2^k 換成每次遇到雙偶數就乘以 2)，而遞迴終止條件則是兩數相等或是兩數其一為 0，當符合終止條件後，再把 r0 乘以兩數中不為 0 的數值，這樣 r0 就是最大公因數。
- 第二個是判斷兩數的奇偶關係，我們是先把兩數分別跟 1 做 and 運算，並將得出的兩結果編碼成 0、1、2、3，藉由這四個數去判斷兩

數的奇偶關係並做相對應的動作。

- iii. 第三個就是判斷何時才是 `stack` 用最多的時候，因為是實作遞迴的版本，所以 `stack` 用最多的時候就在已經算出最大公因數且要開始回傳的當下，這時候拿 `stack base` 減去當下的 `stack pointer` 就可以得到 `stack` 使用的最大值了。

五、心得討論與應用聯想：

這次在實作上較無遇到較大的困難，在會使用 `SP` 的情況下，要實作遞迴確實容易，但對於 `GCD` 來說，我們不認為有必要用到 `SP`，因為所有值都是丟入 `STACK` 後就又要取用，那應該用 `REGISTER` 即可，另外我們在本次有將上次引以為戒，做了大量邊界測試，但仍然不確定助教所想要的 `GCD(0, X)` 應該為多少，暫時先以 `X` 為答案。