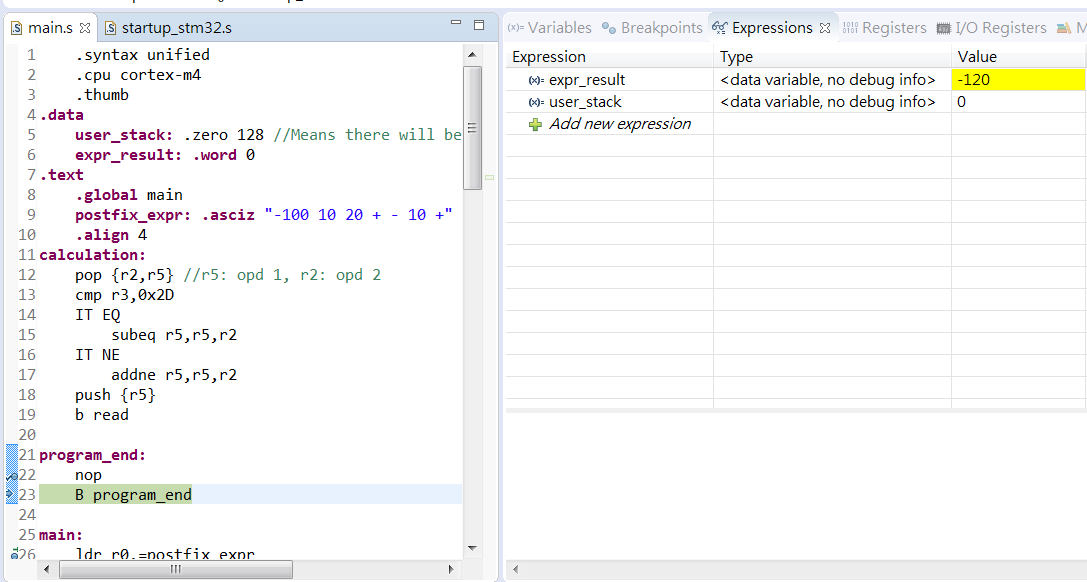
組別：6，組員：王柏堯、張彧豪

微處理機 Lab3 Report

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



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：

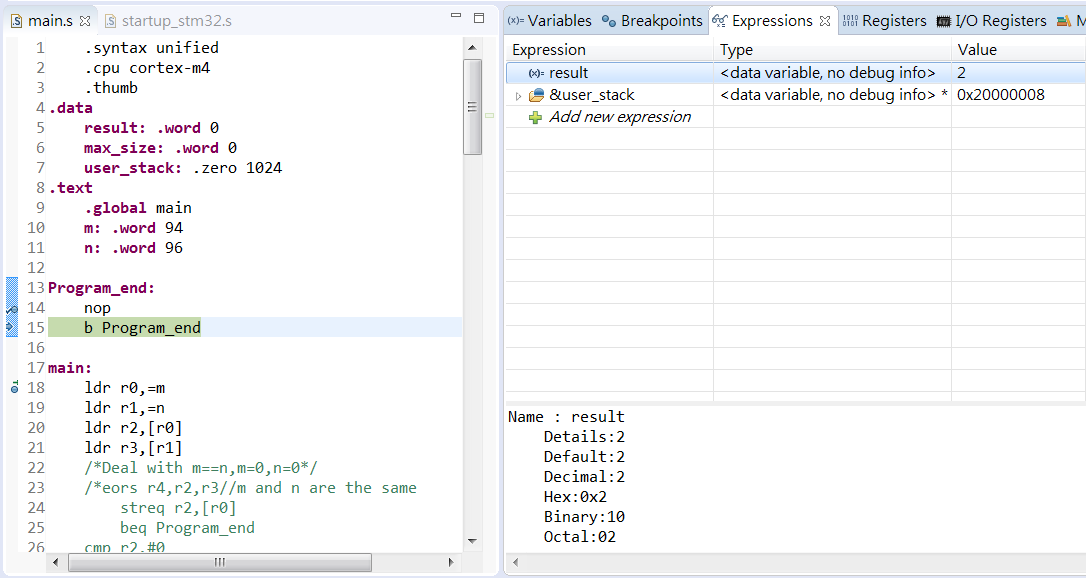
1. 若再讀到空白，表示此負號為一個運算子，跳入calculation計算並存入stack。
2. 若再讀到數字，先紀錄此數字為負數，跳入atoi把此負數給parse出來並存入stack。
3. 若再讀到'\0'，表示此負號為一個運算子且它位於字串最後，計數器減1，跳入calculation計算並存入stack。

* 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後跳出回主要程式區段。
* 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。

1. 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：

1. 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。
2. 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。
3. 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。
4. 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。