# CS1021: Assignment #2

My report on the second computing assignment is as follows:

The assignment was split into three parts: 2.1: Sets – Closure, 2.2: Sets – Symmetric Difference, 2.3: Anagrams

#### 2.1: Sets – Closure

This part involved testing a set of closure i.e. every numeral element requires its opposite to close, -4 to +4.

I first started this my finding half of the size of the set. This is how many times I had to test the set to see if it was closed.

```
15
    whileA
16
        CMP R11, #0 ; while (remainder=!0)
17
        BEO endWh
                        ; {
        SUB R11, R11, #2; {remainder = remainder-2
18
19
        ADD R13, R13, #1; {quatient ++
20
        B whileA
                       ; }
21 endWh
22
```

I was able to use the method above to half the set size as a condition to show that the set was closed. Every time an element was matched the element was turned to 0to prevent rematching and a counter R5 tracked the success rate. When the two were equal the set was closed.

```
; while
  whileCharsChecked
5
      LDR R10, =0
                                   ; sum
      CMP R3, R4
                                  ; (count<#ofElemsRemaining)
      BHS endWhileCharsChecked
                                  ; {
3
9
)
L
      CMP R8, #0
2
      BEQ endA
      ADD R3, R3, #1
ADD R1, R1, #4
3
                                  ; count = count + 1
                                  ; adress=address+4
5
      LDR R9, [R1]
                                  ; charB=Memory.word[adr]
5
      CMP R9, #0
      BEQ endB
      ADD R10, R8, R9
CMP R10, #0
3
                                   ; sum = charA + charB
      CMP R10, #0
3
                                  ; if(sum = 0)
)
      BEQ endWhileCharsMatch
                                  ; {
 endB
                                       ; }
2
          whileCharsChecked
                                  ; }
3
```

The while loop above takes two numbers of the elements compares one to another and if theyre not equal the next number is taken If the two numbers are a paired set the code will branch to the bottom where it gets reset. If there are no matches then the set isn't closed. If

the set contains a 0 in the middle it means it the element was already used and it will branch to increase the address and take the next number for comparison. Unfortunately due to some last minute alterations to the program I was able to create a problem I couldn't fix. Unfortunately when changing a number in the set to 0 the next number selected will be invalid.

### Outputs:

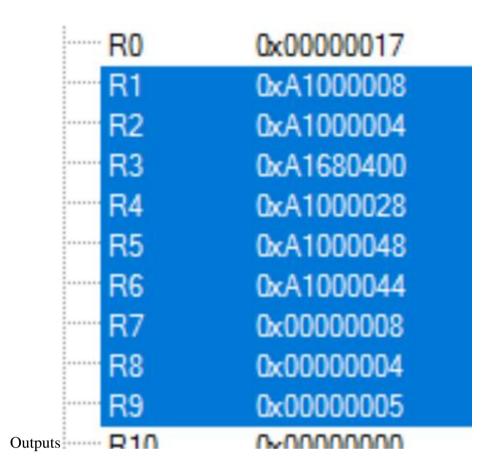
Unfortunately when I changed the input the output was still the same.

```
RO 0x0000001

ASize DCD 8 ; Number of elements in A AElems DCD +4,-6,-4,+3,-7,+6,+8,-3 ; Elements of A
```

# 2.2: Sets – Symmetric Difference

The symmetric difference project takes two sets compares them and put any numbers not in intersection into a new set -C. It also takes into account the amount of numbers in the size set. First I loaded all the various elements into corresponding registers. I used 2 while loops in the project to alternate between changing the element a and b.



## 2.3: Anagrams

This part involved the testing of 2 words against each each other and storing 1 in R0 if one word is an anagram of the other. To accomplish this I first changed each character in every word to lowercase. I took advantage of this lengthy process to count the number of letters in each word with a count. Due to my program taking even capital letters it can be put to more use.

I started off with the assumption that the two words were anagrams and I was trying to prove the assumption wrong.

```
CMP R7, R8
   BEQ anagram
   ADD R9, R9, #1
                       ;count++
   ADD R2, R2, #1
                       ; adress=address+1
   LDRB R8, [R2]
                        ; charB=Memory.word[adr]
   CMP R8, #0x0
                        ; if (next char =0)
   BEQ notAnagram
   B while3
anagram
   ADD R8, R8, #0x20 ; char=upper ; if already used
   STRB R8, [R2]
   SUB R2, R2, R9
   LDR R9, =0
   LDRB R8, [R2]
```

It compared the first letter in both words to each other. If it didn't match the program then took the second letter from the second word. If however the two letter where the same then the letter in the second word was changed to uppercase in order to prevent the reusing of the letter. This was stored and then the adress was put back and another letter was taken in. If the next ketter in either word was null meant that it wasn't an anagram as it looped to the bottom and changed the register 0.

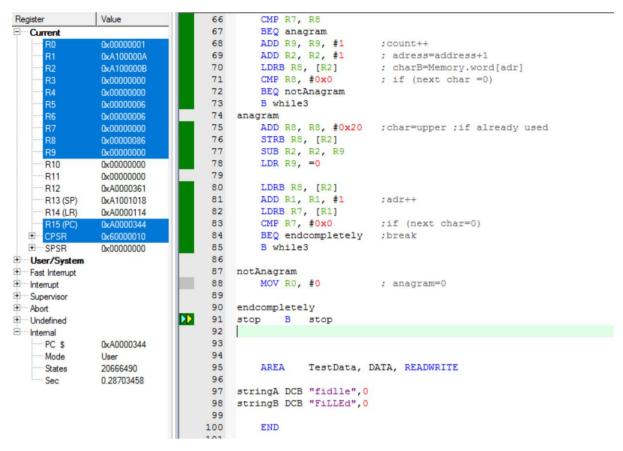
Tacos and coats was the sample anagram that I used as you can see below there was a 1 in register 0 to show that the two words are anagrams.

```
stringA DCB "tacos",0
stringB DCB "coats",0
RO 0x0000001
```

The default test string was bests and beets.

```
stringA DCB "bests",0
stringB DCB "beets",0
R0 0x00000000
```

I was very happy with this program as it works with various letters and number of letters. It doesn't fail when subject to vigorous testing procedure.



A mix of upper and lower case letters worked as illustrated above.

```
ADD R9, R9, #1
                             68
                             69
                                      ADD R2, R2, #1
        0xA100000B
                                                             ; adress=address+1
                                      LDRB R8, [R2]
CMP R8, #0x0
                             70
                                                             ; charB=Memory.word
        0xA100000C
0x000000000
                             71
                                                             ; if (next char =0)
        0x00000000
                             72
                                       BEQ notAnagram
        0x00000007
                             73
                                      B while3
                             74
                                  anagram
        0x00000007
                             75
                                      ADD R8, R8, #0x20
                                                             ; char=upper ; if alr
                             76
                                       STRB R8, [R2]
        0x00000086
                             77
                                       SUB R2, R2, R9
        0x00000000
                                      LDR R9, =0
                             78
        0x00000000
                             79
        0x00000000
                             80
                                      LDRB R8, [R2]
        0xA0000361
3 (SP)
                             81
                                      ADD R1, R1, #1
                                                              ;adr++
        0xA1001018
                                      LDRB R7, [R1]
                             82
4 (LR)
        0xA0000114
(PC)
        0xA0000344
                             83
                                       CMP R7, #0x0
                                                              ;if (next char=0)
                                      BEQ endcompletely
        0x60000010
                             84
                                                             :break
SR
        0x00000000
                             85
                                      B while3
                             86
System
                             87
                                  notAnagram
emupt
                                     MOV RO, #0 ; anagram=0
                             88
t
                             89
sor
                             90
                                  endcompletely
                             91
                                  stop
                                         В
ed
                             92
        0xA0000344
                             93
$
de
        User
                             94
                             95
                                      AREA
                                                TestData, DATA, READWRITE
        34583191
tes
                             96
        0.48032210
                             97
                                  stringA DCB "fi dlle",0
                                  stringB DCB "Fi LLEd", 0
                             98
```

Even spaces counted as shown above.