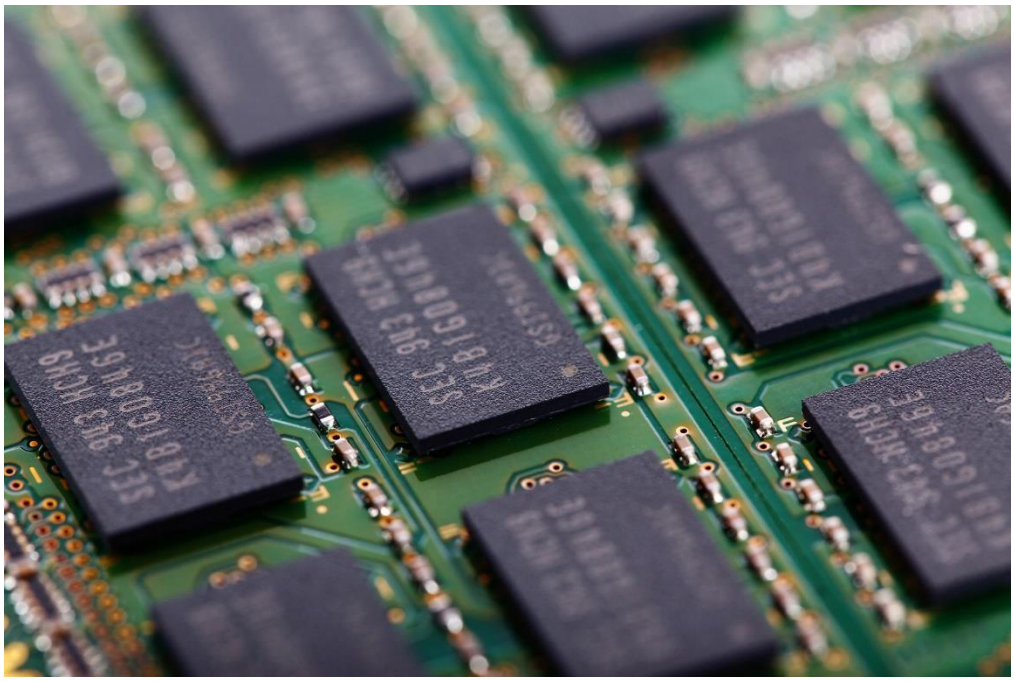
CS1021 - ASSIGNMENT 2



CS1021

'Memory' by Brandon Dooley

Throughout the following report I will document the various stages of the development of my ARM Assembly Language programs for Assignment 2. I will explore various tests that were made throughout the stages of development and difficulties that I encountered respectively.

CS1021 - Assignment 2

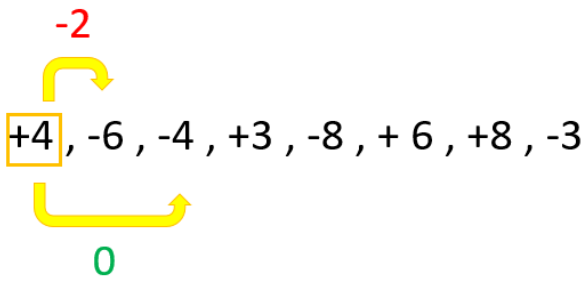
'MEMORY' BY BRANDON DOOLEY

PART ONE | Sets - Closure

"Design and write an ARM Assembly Language program that will determine whether the set is closed under the negation operation. In other words, your program should determine whether, for every integer in the set, the set also contains the negation of that integer."

This program begins by reading the size of the set from memory and also the start address of the elements contained within the set. The program also loads a second copy of the elements start address which is used as a second **comparison set.**

The program takes one element from the original set at a time. It first checks to see if said element is 0, if so it skips this element and continues to the next non-zero integer. It then runs through each element in the comparison set to see if any contain the negation of said element. This is done by adding the two elements together and seeing if the result is 0.



If the addition result is 0, both elements are then changed to zero within memory. The program then moves to the next element in the original set and compares it with the comparison set. This is repeated for all elements in the set. If the program successfully closes the last element by negation the set is therefore **closed under negation** and the value of R0 is set to 1. However if, for any element, the set doesn't contain the negation of such integer the set is **not closed under negation** and the value of R0 is set to 0 and the program terminates.

0 , -6 , 0 , +3 , -8 , + 6 , +8 , -3

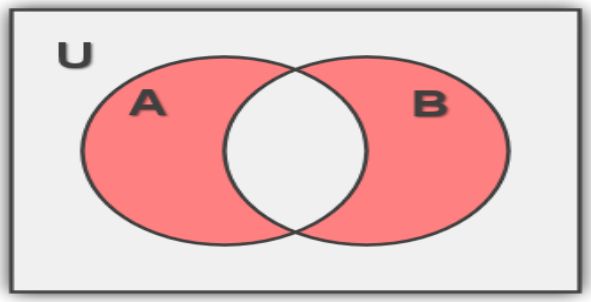
PART ONE | Testing

Input Sets	Expected Value (R0)	Actual Value (R0)
+4,-6,-4,+3,-8,+6,+8,-3	0x00000001	0x00000001 ✓
+4,-281,-4,+98,-8,+83,+1230	0x00000000	0x00000000 ✓
0	0x00000001	0x00000001 ✓
1	0x00000000	0x00000000 ✓

PART TWO | Symmetric Difference

"Assume two mathematical sets, A and B, containing 32-bit unsigned integers are stored in memory. Design and write an ARM Assembly Language program that will create a third set, C, that is the symmetric difference of A and B"

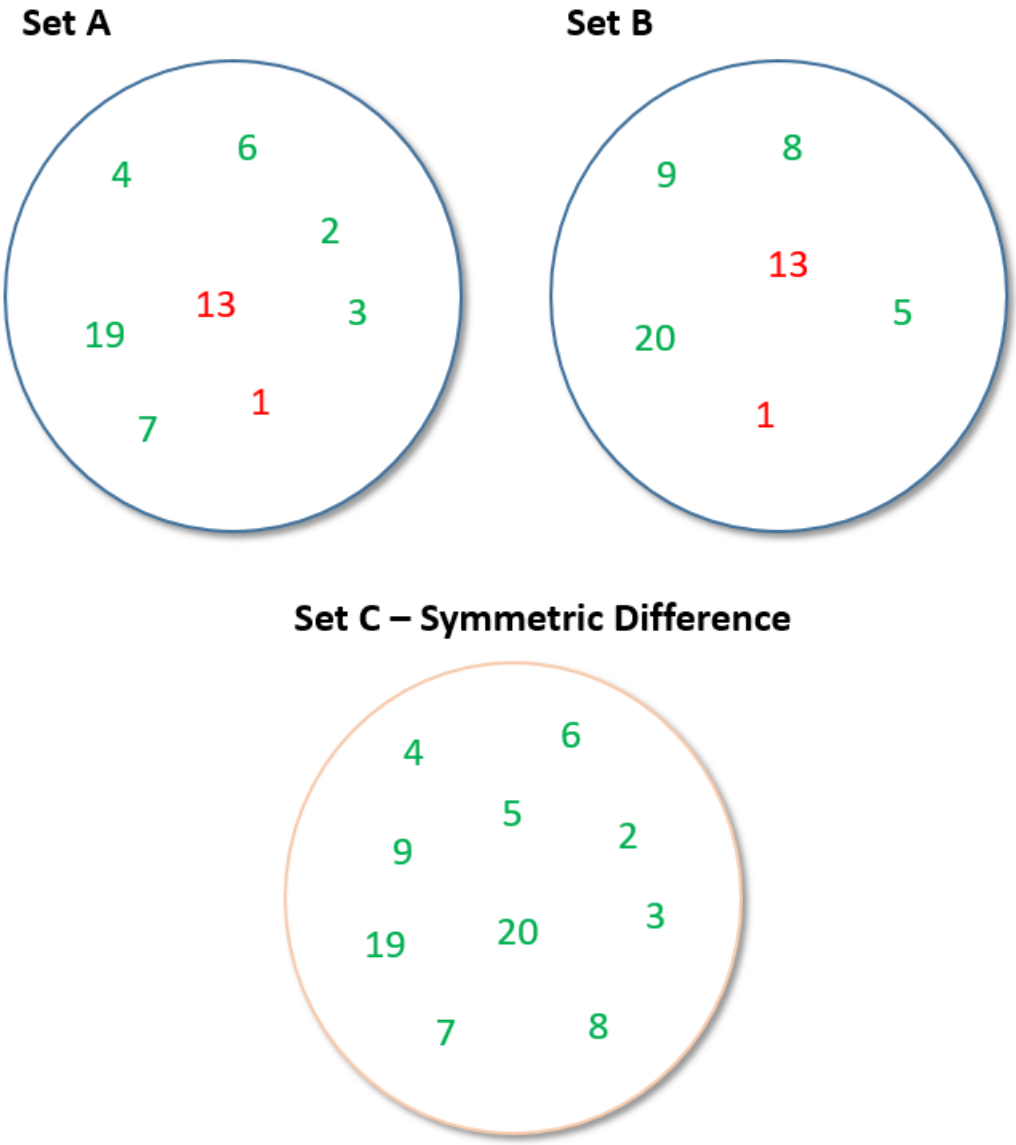
This program begins by taking each integer test value once at a time from each set and comparing it with each value in the corresponding set. If the test value is unique to only one set it is then moved to set C, the ***Symmetric Difference*** of both sets. This is done for every element in Set A and Set B.



The comparison is done in a similar way to part one, however, this time a **subtraction comparison** is used to see if Value A - Value B = 0. If that is the case, the element is not unique and is a part of the *Union Set*.

However, if the **subtraction comparison** is not 0 for all elements in the comparison set, the element is therefore unique and part of the *Symmetric Difference Set*, i.e Set C.

Every time an element is moved to the *Symmetric Difference Set* the size of the set is incremented, starting from 0. Allowing the user to determine how many of the original elements are contained within Set C.



PART TWO| Testing

Input Sets	Expected Set C	Actual Set C
A = **4,6,2,13,19,7,1,3** B = **13,9,1,20,5,8**	4, 6, 2, 19, 7, 3 , 9, 20, 5, 8 C Size = 10	4, 6, 2, 19, 7, 3 , 9, 20, 5, 8 ✓ C Size = 10 ✓
A = **0** B = **1**	0, 1 C Size = 2	0, 1 ✓ C Size = 2 ✓
A = **123, 28, 102, 3** B = **1920, 39**	123, 28, 102, 3, 1920, 39 C Size = 6	123, 28, 102, 3, 1920, 39 ✓ C Size = 6 ✓
A = **Null** B = **Null**	Null C Size = 0	Null ✓ C Size = 0 ✓

PART THREE | Anagram

"Two strings are anagrams of each other if one string can be formed by rearranging the characters of the other. Design and write an ARM Assembly Language Program to determine if one string is an anagram of another string."

This program begins by going through each letter in both strings individually and changing them to lower case, if they are not lower case already.

The program then takes each letter from String A one at a time and compares it to every letter in String B until it finds one that it is the same. If it doesn't find one that is the same the strings **are not anagrams** of each other and therefore R0 is set to 0 and the program is terminated.

However, if it does find the same letter in String B both letters are changed to the SPACE ASCII Key (0x20) which is used to flag letters that have already been marked. The program then takes the next letter from String A and runs through String B performing comparisons whilst **skipping values that contain the SPACE Flag**.

If the program successfully reaches the end of String A it checks to ensure that all the characters of String B contain the SPACE Flag before terminating and indicating that both Strings **are anagrams** of each other and therefore storing the value 1 in R0.

PART THREE | Testing

Input Strings	Expected Value (R0)	Actual Value (R0)
String A = **Bests** String B = **bests**	0x00000001	0x00000001 ✓
String A = **TaCoS** String B = **COATS**	0x00000001	0x00000001 ✓
String A = **FiveGuys** String B = **FishBox**	0x00000000	0x00000000 ✓
String A = **Best tacos** String B = **BEST coats**	0x00000001	0x00000001✓

ANALYSIS| Overall Program

I feel that the program on an overall level serves its purpose both efficiently and clearly. Within reason it allows the user to perform various functions, using memory manipulation sufficiently and effectively to do so.

I felt the assignment itself was challenging yet rewarding. I feel I have gained immense knowledge of the assembly language itself and also learned how to use memory to perform tasks that would otherwise not have been possible . I have also realized the necessity of thoroughly and rigorously testing programs to ensure all possible inputs and scenarios are handled.

Brandon Dooley (#16327446)

CS1021 Assignment 2 – Memory

Submitted 21/12/2016