DAVY NOLAN
CS1021 ASSIGNMENT #2

Memory

Date: 22nd of December 2017

Davy Nolan
Assignment #2

Stage 1: Console Input

Aim:

The aim of this stage of the assignment is to "design and write an ARM Assembly Language program that will create a third set, C, that is the symmetric difference of A and B."

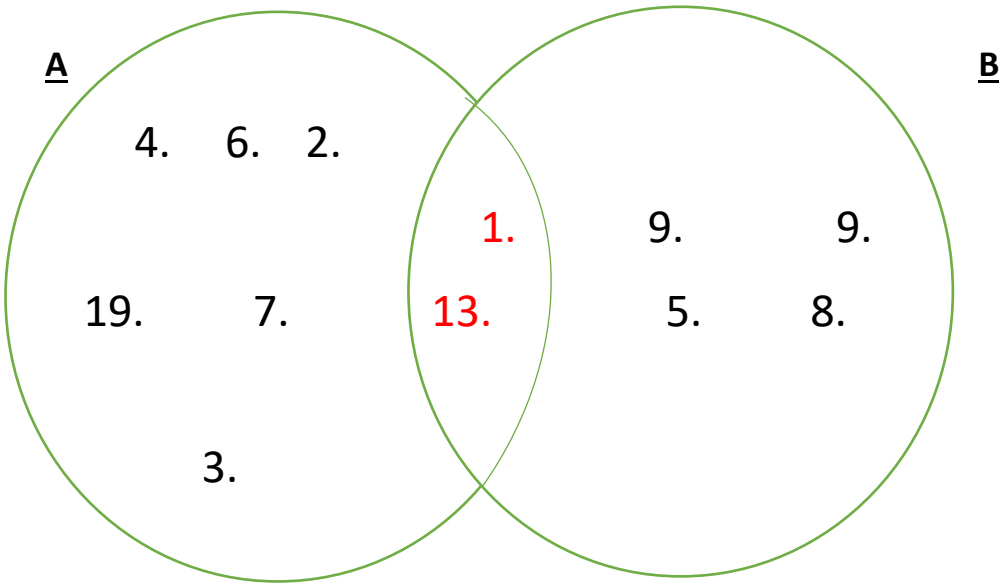
The following ARM Assembler directives illustrate how the sets are arranged:

```
1 ASize   DCD 8                    ; Number of elements in Set A
2 AElems  DCD 4,6,2,13,19,7,1,3    ; Elements in Set A
3
4 BSize   DCD 6                    ; Number of elements in Set B
5 BElems  DCD 13,9,1,20,5,8        ; Elements in Set B
```

Solution:

Set A = {4, 6, 2, 13, 19, 7, 1 ,3}

Set B = {13, 9, 1, 9, 5, 8}



Set C = Symmetric Difference = {4, 6, 2, 19, 7, 3, 9, 9, 5, 8}

It was in imperative that the sets were loaded into the program. Pseudocode was prepared in order to do so:

Pseudocode:

```
;Load amount of elements in A
;Load start of address of ASize
;Load elements of A

;Load amount of elements in B
;Load start of address of BSize
;Load elements of B

;Load amount of elements in C
;Load start of address of CSize
;Load elements of C
```

```
start
    LDR R1, =ASize
    LDR R1, [R1]
    LDR R2, =AElems

    LDR R3, =BSize
    LDR R3, [R3]
    LDR R4, =BElems

    LDR R5, =CSize
    LDR R5, [R5]
    LDR R6, =CElems
```

Next, pseudocode was prepared to explain how to check each element of sets A and B and to compare them together. The “ASize” and “BSize” strings were used as counts and each time the program moves on to the next element, it decrements the corresponding count.

```
;if(ASize != 0)
;{
;R7 = start address of AElems
;R8 = start address of BElems}
;if(1st element of A = 1st element of B)
;{
;store start address of AElems in R9
;store start address of BElems in R9
;move on to next element of A
;ASize--}


;if(amount of elements of B != 0)
;{
;move on to next element of B
;BSize--}
```

```
while
    CMP R1, #0
    BEQ endwhile1
    LDR R7, [R2]
    LDR R8, [R4]
    CMP R7, R8
    BNE endwhile2
    STR R9, [R2]
    STR R9, [R4]
    ADD R2, R2, #4
    SUB R1, R1, #1
    B while
endwhile2
    CMP R3, #0
    BEQ endwhile3
    ADD R4, R4, #4
    SUB R3, R3, #1
    B while
```

The program then continued to find the symmetric difference of sets A and B which is all of the elements that they don't have in common.

Stage 2: Countdown Checker

Aim:

"Design and write an ARM Assembly Language program to determine if one string, A, can be formed from the nine letters contains in a second string, B."

Solution:

First of all, "cdWordSize" and "cdLetterSize" were added to the program to act as counts. These were set as values 5 and 9 due to the 5 letters being in the word "beets" and 9 in "daetebzsb".

```
45      AREA    TestData, DATA, READWRITE
46
47  cdWord
48      DCB "beets",0
49  cdWordSize  DCD 5
50  cdLetters
51      DCB "daetebzsb",0
52  cdLetterSize  DCD 9
53      END
```

Just like in stage 1, the program begins with all the registers being loaded. Counts were also loaded into register R9 and R10.

```
start
    LDR R1, =cdWord ; Load start address of word
    LDR R2, =cdLetters  ; Load start address of letters
    LDR R7, =cdLetterSize  ;Load amount of letters in cdletters
    LDR R8, =cdWordSize   ;Load amount of letters in word
    LDR R9, =0  ;count for common letters
    LDR R10, =1  ;count for cdLetterSize
```

Pseudocode was prepared to explain how to go about beginning this program.

```
;Load start address of word
;Load start address of letters
;if (1st letter of word = 1st letter of cdletters)
;{

;count++}
```

```
while
    LDR R3, [R1]
    LDR R4, [R2]
    CMP R3, R4
    BNE endwhile
equalLetters
    ADD R9, R9, #1
```

The program then had to go on and check for each letter in the 9-letter string and compare each letter to the first letter in the 5-letter word. If a letter matches, then the count increases by 1 until reaching 5. When the count reaches 5, we know that the word is legit and can be successfully created from the given 9 letters.

The program then must go on and check for the next letter in the word. A pseudocode was written up to do all of this:

```
;if (letter count != size of word)
;{

;move on to next letter in cdletters
;letterSizeCount++}
;if(letterSizeCount != letterSize)
;{

;if(letter of word = letter of cdletters)
; { repeat endwhile}


;try next letter of word
;go back to first letter of cdletters
;reset letterSizeCount
```

```
19      CMP R9, R8
20      BEQ correctLetters
21  endwhile
22      ADD R4, R4, #1
23      ADD R10, R10, #1
24      CMP R10, R7
25      BEQ nextLetter
26  compare
27      CMP R3, R4
28      BEQ equalLetters
29      B endwhile
30  nextLetter
31      ADD R3, R3, #1
32      LDR R4, [R2]
33      LDR R10, =1
34      B compare
```

The program also has to store a value of 1 in R0 if the word could be created from the given letters. I did this by stating in the program that if the letter count is equal to the word size, then branch to “correctLetters” where the program then loads the value 1 into R0.

```
19      CMP R9, R8          ;if (letter count != size of word)
20      BEQ correctLetters  ;{
```

```
35  correctLetters
36      LDR R0, =1          ;if the word can be created from letters
37                          ;in cdletters, then put 1 in R0
```

Methodology:

The program appears to have an error in it as when it is run for the given example, 1 is not the value in R0.

Stage 3: Lottery

Aim:

"Design and write an ARM Assembly Language program that will determine the number of tickets that match four numbers, five numbers and six numbers. (i.e. your program should produce three result values for the number of "match four" tickets, "match five" tickets and "match six" tickets.)"

Solution:

To make this program more functional and easier to work with, the "TICKET" string was split into three strings "TICKET1", "TICKET2", and "TICKET3". All these values were loaded into the registers at the beginning of the program. The "DRAW" string containing the winning numbers was also loaded at the beginning of the program. A number count of 6 was also loaded as each ticket consists of 6 numbers.

```
 5   start
 6       LDR R1, =TICKET1
 7       LDR R2, =TICKET2
 8       LDR R3, =TICKET3
 9       LDR R4, =DRAW
10       LDR R5, =6
```

```
34  COUNT   DCD 3            ; Number of
35  TICKET1 DCD 3, 8, 11, 21, 22, 31
36  TICKET2 DCD 7, 23, 25, 28, 29, 32
37  TICKET3 DCD 10, 11, 12, 22, 26, 30
38
39
40  DRAW    DCD 10, 11, 12, 22, 26, 30
```

I experienced a lot of trouble with the rest of stage 3 of this assignment so therefore the rest is unfinished.