Name: Seán Roche
Student Number: 17332021
Date of Submission: 22/12/17
Module: CS1021

Assignment #2 - Memory

Introduction

This report describes the development of three ARM Assembly Language programs on the topic of memory. The report is consequentially broken down into three sections:

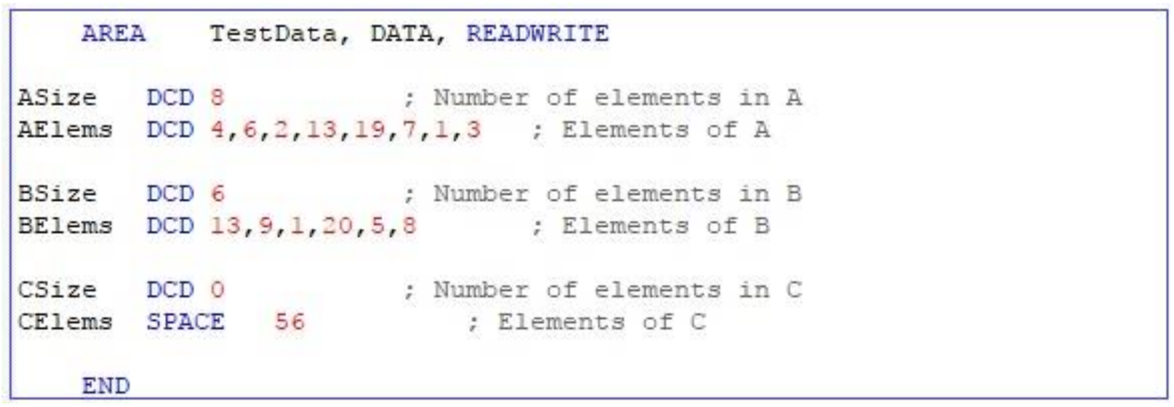
1. Sets – Symmetric Difference
2. Countdown Checker
3. Lottery

Each program along with the results are addressed in turn with an overall conclusion at the end.

Part 1 – Symmetric Difference

This section entails the writing of a program that takes two mathematical sets, A and B, calculates the symmetric difference and stores the result in a third set, C.

Sets A and B are already given, they contain 32-bit unsigned integers and are stored in memory.



```
	AREA	TestData, DATA, READWRITE

ASize	DCD 8			; Number of elements in A
AElems	DCD 4,6,2,13,19,7,1,3	; Elements of A

BSize	DCD 6			; Number of elements in B
BElems	DCD 13,9,1,20,5,8		; Elements of B

CSize	DCD 0			; Number of elements in C
CElems	SPACE	56		; Elements of C

	END
```

The conceptual approach I used to achieve this functionality may be summarised as follows:

- A while loop was set up that would procedurally compare values in set A to values in set B.
- When a match was found the loop would simply replace the numbers in both sets with a '!' and move to the next digit to be checked.

- When there is not a match the program branches to a separate section of code where it is ensured that the counter has not reached 0 to indicate that all have numbers have been checked. If this is not the case the value is stored in C and the size of C is increased by 1.
- When all of the numbers in set A have been checked the program branches to a second loop. In this loop all numbers in set B that have not been replaced by a '!' are added to set C, since they must be unique to set C.

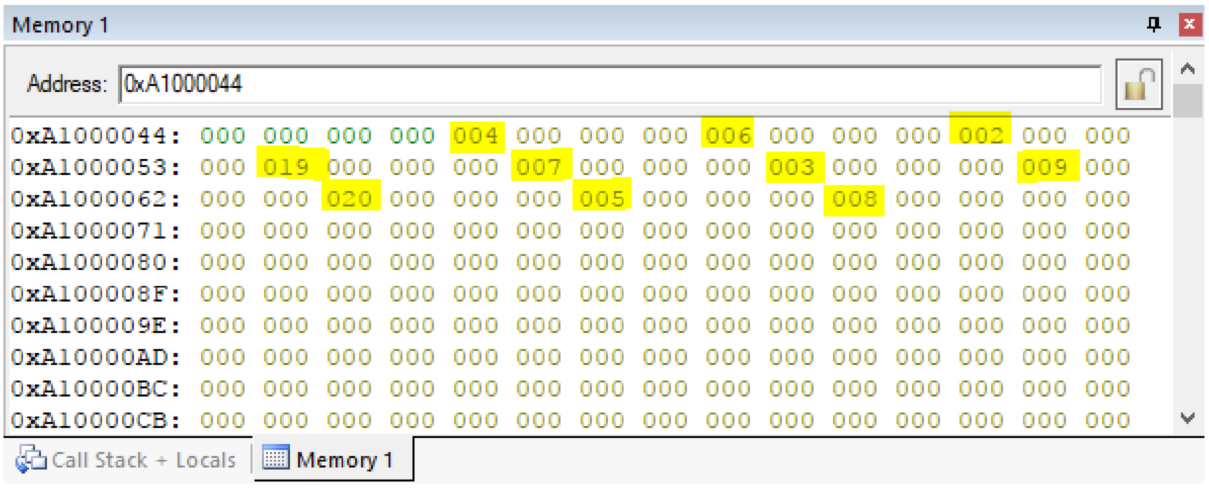
Psuedo Code

```
Let a = Address AElems
Let b = Address BElems
while(elements in A are unchecked)
      Load number from a to be compared
      Load number from b to be compared
      while(elements in b are unchecked)
            Compare a to b
            If a and b are the same
                  Jump to check next A value
            Else
                  Check next b value
      If all values in b are checked with no match
            Add the number to CElems
            Go to next A value and check the next a value
```

Result:

The program performed as expected with the correct set C being observed at the address stored in R4.

The symmetric difference of sets A and B is:
{2, 3, 4, 5, 6, 7, 8, 9, 19, 20}



```
0xA1000044
0xA1000044: 000 000 000 000 004 000 000 000 006 000 000 000 002 000 000
0xA1000053: 000 019 000 000 000 007 000 000 000 003 000 000 000 009 000
0xA1000062: 000 000 020 000 000 000 005 000 000 000 008 000 000 000 000
0xA1000071: 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
0xA1000080: 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
0xA100008F: 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
0xA100009E: 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
0xA10000AD: 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
0xA10000BC: 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
0xA10000CB: 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
```

Part 2 – Countdown Checker

For this part of the assignment the task was to build a program that would check to see if a word (string A), could be formed using only the nine letters given in string B, with each letter only being used once.

For example to form the word “street”, the word must include at least one “s”, two “t’s”, two “e’s” and one “r”.

The solution for this was similar to before but somewhat more complex:

- A while loop compares letters in the word to each of the nine letters in the string.
- When a match is found a ‘!’ is stored in place of the letter in both the word and the string of letters.
- When a 0 is detected the program branches to a second while loop. (i.e, the loop is null terminated). This indicates that all letters have been checked.
- In a third loop the program checks for ‘!’ until once again it reaches a 0. If it finds another letter (e.g. “g”), the program terminates.
- If all letters have been replaced by “!”, 1 is added to R0 to indicate that the word can be created.

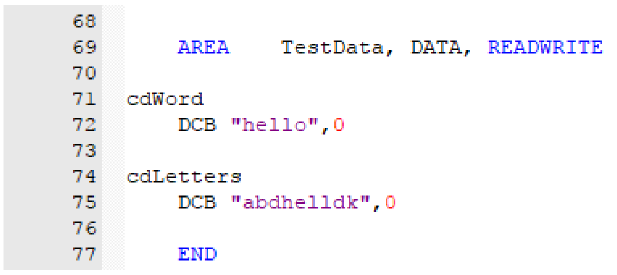
Psuedo Code

```
Let wordLetter    = Address cdWord
Let letter              = Address cdLetter
while(wordLetter != 0)
      Compare wordLetter with letter
      if(both are the same)
            Move onto the next wordLetter
            Remove the letter from the pool of letters i.e. overwrite
with    blank
      else
            while(wordLetter != letter)
                  Try the next letter
            if (hit all letter with no match)
                  Return incorrect (0)
```

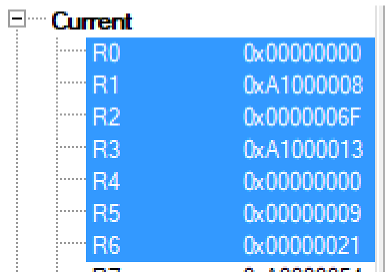
Result:

Fortunately the program functions perfectly. Storing 0 in R0 when the word cannot be formed and 1 when the word can be formed.

Negative Test:

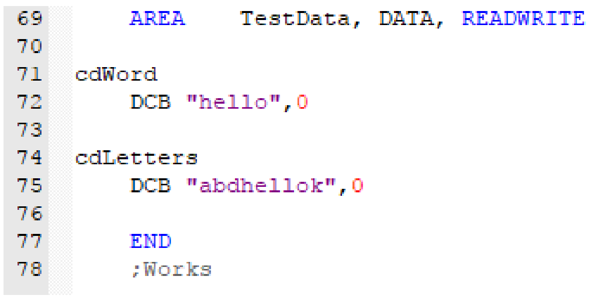


```

    AREA    TestData, DATA, READWRITE

cdWord
    DCB "hello",0

cdLetters
    DCB "abdhelldk",0

    END
```

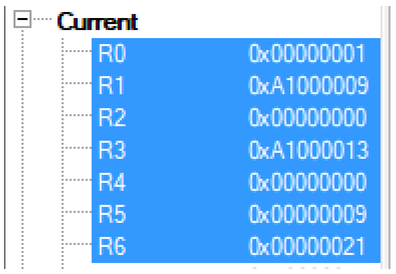


```
Current
  R0    0x00000000
  R1    0xA1000008
  R2    0x0000006F
  R3    0xA1000013
  R4    0x00000000
  R5    0x00000009
  R6    0x00000021
```

Positive Test:



```
    AREA    TestData, DATA, READWRITE

cdWord
    DCB "hello",0

cdLetters
    DCB "abdhellok",0

    END
    ;Works
```



```
Current
  R0    0x00000001
  R1    0xA1000009
  R2    0x00000000
  R3    0xA1000013
  R4    0x00000000
  R5    0x00000009
  R6    0x00000021
```

Part 3 – Lotto

This section describes the writing of a program that counts the number of prize winners for a lottery game.

Players choose six numbers between 1 and 32 with prizes for matching four, five or six numbers.

The lotto numbers, the number of tickets and the tickets are stored in memory.

The approach was as follows:

- The numbers on each ticket are compared to each number in the draw (using two for loops as before).
- Any match will increment a counter
- Once the six numbers on the ticket are checked, check the counter.
- If 4, 5 or 6 then increment the corresponding counter and store.

Psuedo Code

```
for (int I = 0 < ticketAmount.length; i++) {
counter = 0;
matchCount = 0;

      for (int counter = 0; counter < ticketLength; counter++)
      {

            currentNum = ticket[counter];

            for(int m+0; m < draw.length= m++)
            {

                  currentDraw = draw[m];

                  if (currentNum ++ currentDraw)
                     matchCount++;

      }
      Draw = draw[0]; // reset draw pointer to start
```

Result

Unfortunately, this solution did not perform as planned once constructed in ARM. The values for the counts were completely wrong.

Despite efforts to implement breakpoints and debug the program I did not find a solution.

I believe the program is somehow not resetting values after the first ticket is done.

Conclusion

Overall I think I learned a lot about memory in the writing of these three programs.

While I am quite happy with the success of the first and second program I am disappointed with the Lotto program. I believe if I spent more time I could have solved the problems. I may continue debugging it over the next few days.

I am also disappointed that while they for the most part fulfilled their requirements none of my programs went “the extra mile”.

I hope to improve on this in the future.