CS1022 Exercise Set #1
Hilary Term Weeks 2 and 3
Addressing Modes and Arrays

1 Addressing Modes

(a) For each of the instructions below, predict the precise memory operation that will be performed. Your prediction should include the memory address that is accessed and the new value contained in the base address register, R1, if it has been modified.

Verify your solutions using µVision.

```
1        LDR     R0, [R1, #8]
2        LDRB    R0, [R1, #1]!
3        LDR     R0, [R1], #4
4        STR     R0, [R1, R2, LSL #2]
```

Assume the following initial values in R1 and R2:

`R1=0xA1000080, R2=0x00000042`

2 Arrays

(a) Translate each of the pseudo-code statements below into ARM Assembly Language. Assume that variables f, i and j correspond to registers R4, R5 and R6 respectively. A is a one-dimensional array of 16-bit unsigned values with a starting address contained in R10. B is a 16×16 (two-dimensional) array of 32-bit unsigned values with a starting address contained in R11. (Assume that B is stored in row-major order.)

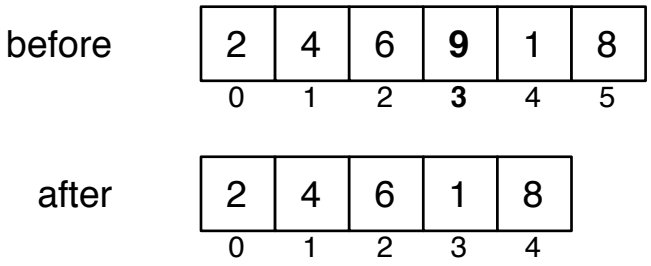
(i) f = A[i]

(ii) A[j] = A[j+2]

(iii) f = B[i][j] * B[j][i]

(b) Write an ARM Assembly Language program that will remove an array element from a specified index in an array of word-size values. The figure below illustrates an array in which an element is removed from index 3. **When removing the element from the array, your program should move the subsequent elements towards the start of the array to fill the "gap" created in memory.** Begin by storing the start address of a test array in R0 and the index of a test element to remove in R1. Assume the number of elements in the array is stored in R2.

Note: a similar problem has been set as the GRADED EXERCISE below.



(c) The pseudo-code below describes an algorithm to multiply two $N \times N$ matrices, A and B, and store the result in a third $N \times N$ matrix, R. Translate the pseudo-code into an ARM Assembly Language program.

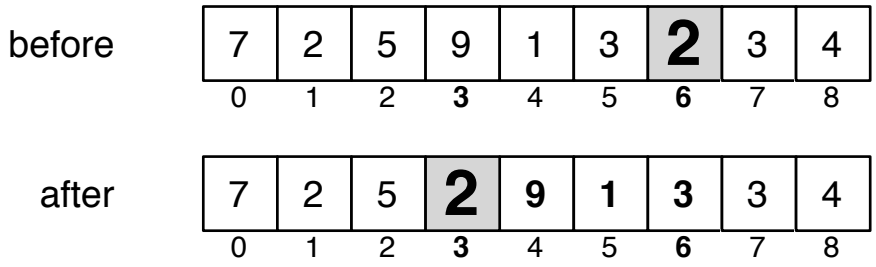
(You might find it instructive to practice 3×3 matrix multiplication on paper first, to better understand the algorithm.)

```
for (i = 0; i < N; i++) {
  for (j = 0; j < N; j++) {
    r = 0;
    for (k = 0; k < N; k++) {
      r = r + ( A[i,k] * B[k,j] );
    }
    R[i,j] = r;
  }
}
```

3 GRADED EXERCISE: Relocate Array Element

Write an ARM Assembly Language program that will relocate an array element from an old index to a new index in an array of word-size values.

The figure below illustrates an array in which an element is relocated from old index 6 to new index 3. **Note how the elements between the old and new indices have been moved to fill the "gap" that was left in the array.**



4 BONUS PROBLEM: Magic Squares

A "magic square" is an $N \times N$ array in which

- every element is unique and
- the sum of the elements in each row, each column and each of the two diagonals is

$$\frac{N(N^2+1)}{2}$$

The following is an example of a magic square:

8	1	6
3	5	7
4	9	2

Design and write an ARM assembly language program that will determine whether an $N \times N$ array stored in memory is a magic square. **You may assume that the elements are unique** and need only check the sum of the rows, columns and diagonals.

Assume that R1 contains the address of the start of the 2-D array in memory, R2 contains the array dimension, N, the array is stored in row-major order and each element is one word (32 bits).