**Question 1** (10 points)

An affine transformation on a vector is an operation with the following form:

where *A* is a matrix and *b*, *c*, *d* are vectors of elements.

This kind of transformation can also be applied to matrix and vectors of bits by substituting multiplications with a bitwise AND and additions with a bitwise XOR, i.e.:

**Note:** the sum of the products is performed using XORs as well.

Write a routine in ARM assembly called bitwiseAffineTransformation that receives in input:

* the address of the matrix *A*
* the 8-bit value of *b*
* the 8-bit value of *c*

The routine computes the following affine transformation:

Keep in mind the following information:

* Each row of the matrix is stored in a single byte
* The bits in each row of the matrix are ordered left-to-right from most to least significant bit
* The vectors are ordered top-to-bottom from most to least significant bit

**Example**

The bit matrix is stored as a vector of 8 bytes in a variable called transformationMatrix:

transformationMatrix DCB 0xF8, 0x7C, 0x3E, 0x1F, 0x8F, 0xC7, 0xE3, 0xF1

With *b* = 0xAA and *c* = 0x63, the routine computes the following transformation:

*a7* AND *b* = 11111000 AND 10101010 = 10101000

1 XOR 0 XOR 1 XOR 0 XOR 1 XOR 0 XOR 0 XOR 0 = 1

*d7* = 0 XOR 1 = 1

*a6* AND *b* = 01111100 AND 10101010 = 00101000

0 XOR 0 XOR 1 XOR 0 XOR 1 XOR 0 XOR 0 XOR 0 = 0

*d6* = 1 XOR 0 = 1

*a5* AND *b* = 00111110 AND 10101010 = 00101010

0 XOR 0 XOR 1 XOR 0 XOR 1 XOR 0 XOR 1 XOR 0 = 1

*d5* = 1 XOR 1 = 0

*a4* AND *b* = 00011111 AND 10101010 = 00001010

0 XOR 0 XOR 0 XOR 0 XOR 1 XOR 0 XOR 1 XOR 0 = 0

*d4* = 0 XOR 0 = 0

*a3* AND *b* = 10001111 AND 10101010 = 10001010

1 XOR 0 XOR 0 XOR 0 XOR 1 XOR 0 XOR 1 XOR 0 = 1

*d3* = 0 XOR 1 = 1

*a2* AND *b* = 11000111 AND 10101010 = 10000010

1 XOR 0 XOR 0 XOR 0 XOR 0 XOR 0 XOR 1 XOR 0 = 0

*d2* = 0 XOR 0 = 0

*a1* AND *b* = 11100011 AND 10101010 = 10100010

1 XOR 0 XOR 1 XOR 0 XOR 0 XOR 0 XOR 1 XOR 0 = 1

*d1* = 1 XOR 1 = 0

*a0* AND *b* = 11110001 AND 10101010 = 10100000

1 XOR 0 XOR 1 XOR 0 XOR 0 XOR 0 XOR 0 XOR 0 = 0

*d1* = 1 XOR 0 = 1

The result of the transformation is *d* = 110010012

Important notes:

1. Write your code inside the “ARM” directory
2. The assembly subroutine must comply with the ARM Architecture Procedure Call Standard (AAPCS) standard (in terms of parameter passing, returned value, callee-saved registers).

**Question 2** (8 points)

Extend the previous exercise as follows.

Initialize TIMER1 to count to 0xFFFF. When the counter reaches 0xFFFF, the timer is reset and no interrupt is generated.

When INT0 is pressed:

1. Read the value of the timer counter of TIMER1.
2. Perform the XOR of the most and least significant byte of the value.

Remember that the timer counter is 32 bits long: you have to take into account only on the least significant 16 bits! If the value you read from the timer counter is 0x000042AA, the most significant byte is 0x42 and the least significant is 0xAA.

1. Show the value you obtained on the LEDs.

When KEY1 is pressed:

1. Call bitwiseAffineTransformation passing:
   1. The constant array [0x8F, 0xC7, 0xE3, 0xF1, 0xF8, 0x7C, 0x3E, 0x1F] as matrix *A*
   2. the value that is being showed on the LEDs as *B*
   3. the value 0x63 as *C*
2. Show the transformed value on the LEDs, using TIMER0 to blink the LEDs with a period of 0.5 seconds (meaning that they have to stay on for 0.25 seconds and off for 0.25 seconds)