Homework 1, Due Date*: 12:00pm 09/01/2017, Cutoff Date*: 12:00pm 09/04/2017
***Late penalty will apply for past-due late submission, ****Submission will NOT be accepted after the cutoff deadline**
Submission: TWO .s files on Blackboard (please include _apm_ in the names if using APM or _uV_ if using uVision 5)

Modified from Textbook. Perhaps the simplest "serious" symmetric block encryption algorithm is the Tiny Encryption Algorithm (TEA). TEA operates on 64- bit blocks of plaintext using a 128- bit key. The ***plaintext*** is divided into two 32- bit blocks **(L_0, R_0)**, and the ***key*** is divided into four 32-bit blocks **(K_0, K_1, K_2, K_3)**. As shown in the diagram, encryption involves repeated application of a pair of rounds, defined as follows for rounds i and $i + 1$ (i starts with 1):

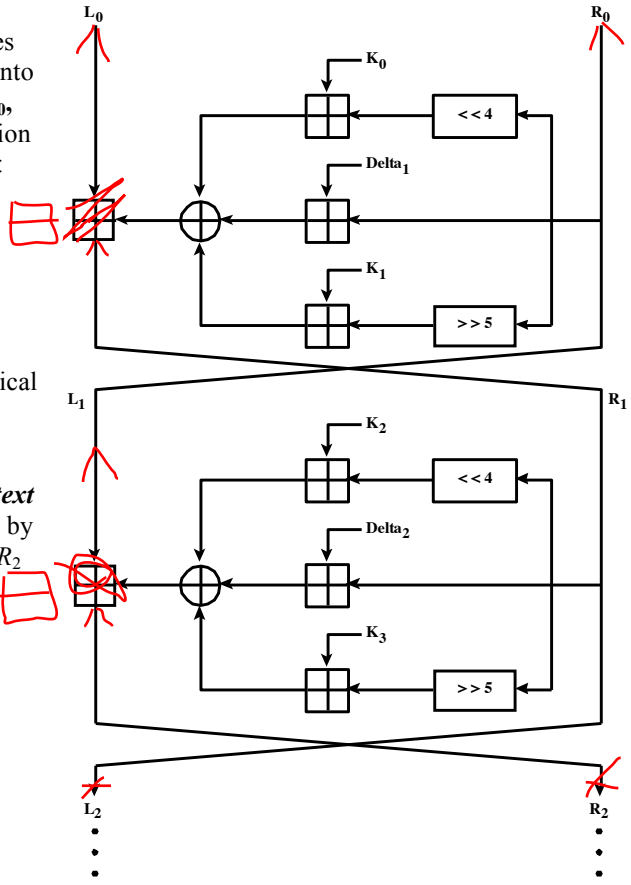
$L_i = R_{i-1}$ $\quad R_i = L_{i-1} \boxplus F(R_{i-1}, K_0, K_1, \delta_i)$

$L_{i+1} = R_i$ $\quad R_{i+1} = L_i \boxplus F(R_i, K_2, K_3, \delta_{i+1})$

where F is defined as

$F(X, K_m, K_n, \delta_y) = ((X << 4) \boxplus K_m) \oplus ((X >> 5) \boxplus K_n) \oplus (X \boxplus \delta_y)$

and where the logical left shift of x by y bits is denoted by x << y; the logical right shift of x by y bits is denoted by x >> y; δ_i (Delta$_i$) is a sequence of predetermined constants; and $\boxplus$ denotes addition-mod-2^{32}.



a. If only one ***pair*** of rounds, i.e., rounds 1 and 2, is used, then ***the ciphertext is the 64- bit block (L_2, R_2).*** You may express the ***encryption*** algorithm by representing L_2 as a function of L_0, R_0, K_0, K_1, and δ_1, and representing R_2 as a function of L_2, R_0, K_2, K_3, and δ_2.

b. The ***decryption*** algorithm is given as below. You may verify it by reverting the calculation in the ***block diagram***.

$R_0 = R_2 \boxminus [[(L_2 << 4) \boxplus K_2] \oplus [L_2 \boxplus \delta_2] \oplus [(L_2 >> 5) \boxplus K_3]]$

$L_0 = L_2 \boxminus [[(R_0 << 4) \boxplus K_0] \oplus [R_0 \boxplus \delta_1] \oplus [(R_0 >> 5) \boxplus K_1]]$

where $\boxminus$ denotes subtraction-mod-2^{32}.

Task 1. Program TEA_Encryption.s: Write an ARM program to implement TEA ***Encryption*** when there is only one pair of rounds.

- in the data area,
  - declare a ***word*** labeled with DeltaOne, initialize it as 0x11111111
  - declare a ***word*** labeled with DeltaTwo, initialize it as 0x22222222
  - declare four ***words*** labeled with KZero, KOne, KTwo, and KThree, initialize them using the key of your choice
  - declare two ***words*** labeled with LZero and RZero, initialize them according to the plaintext of your choice
  - declare two ***word*** labeled with LTwo and RTwo, initialize them as 0's
- in the main program,
  - Load the values of δ_1, δ_2, L_0, R_0, K_0, K_1, K_2, and K_3 to registers of your choice
  - Implement the **encryption** algorithm to calculate L_2 and R_2
  - Store the values of L_2 and R_2 to memory locations at LTwo and RTwo, respectively

Task 2. Program TEA_Decryption.s: Write another ARM program to implement TEA ***Decryption*** for one pair of rounds.

- in the data area,
  - declare a ***word*** labeled with DeltaOne, initialize it as 0x11111111
  - declare a ***word*** labeled with DeltaTwo, initialize it as 0x22222222
  - declare four ***words*** labeled with KZero, KOne, KTwo, and KThree, initialize them the same as Task 1
  - declare two ***words*** labeled with LTwo and RTwo, initialize them using the results of Task 1
  - declare two ***word*** labeled with LZero and RZero, initialize them as 0's
- in the main program,
  - Load the values of δ_1, δ_2, L_2, R_2, K_0, K_1, K_2, and K_3 to registers of your choice
  - Implement the **decryption** algorithm to calculate L_0 and R_0
  - Store the values of L_0 and R_0 to memory locations at LZero and RZero, respectively

Task 3. Debug and TEST your programs: The grading is based on TESTING results instead of code reading. **Up to 5% of the points may be given if your programs cannot be assembled**.