IFT209 – Programmation système

Université de Sherbrooke

Laboratoire 5

Enseignant:	Michael Blondin
Date de remise:	dimanche 17 mars 2019 à 23:59
À réaliser:	en équipe de deux
Modalités:	remettre en ligne sur Turnin; une seule remise avec vos noms/CIP en commentaires en en-tête du code

Problème. Cette semaine, nous faisons une courte excursion en cryptographie. Dans ce domaine, les primitives de *chiffrement par bloc* permettent de mélanger les bits d'un message secret à l'aide d'une clé secrète afin d'empêcher que le message soit déchiffré par un adversaire. Un message chiffré peut seulement être déchiffré à l'aide de la clé secrète. Plutôt que d'opérer sur un message complet, le chiffrement par bloc opère sur chaque bloc de données du message, par ex. deux mots de 32 bits à la fois.

Le but de ce laboratoire est de déchiffrer ce message secret:

```
0x81FB4E63 0x58164679 0x4E934565 0x77DAFEB0 0xABD622CC 0x770C1BC5
```

Le message a été chiffré avec l'algorithme *Tiny Encryption (TEA)* mis au point par David Wheeler et Roger Needham à l'Université de Cambridge. Cet algorithme permet de chiffrer deux mots w_0, w_1 de 32 bits en mélangeant leurs bits à l'aide de quatre clés secrètes w_2, w_3, w_4, w_5 de 32 bits chacune.

Le message secret a été chiffré avec les clés suivantes:

$$w_2 = \texttt{0xABCDEF01}, \quad w_3 = \texttt{0x11111111}, \quad w_4 = \texttt{0x12345678}, \quad w_5 = \texttt{0x90000000}.$$

Vous devez:

(1) Implémenter la procédure de déchiffrement de TEA:

$\delta \leftarrow \texttt{9E3779B9}_{16}$ `// Constante`
$i \leftarrow 1$

tant que $i \leq 32$ **faire**
  $w_0, w_1 \leftarrow \mathsf{decode}_i(w_0, w_1, w_2, w_3, w_4, w_5)$
  $i \leftarrow i + 1$

retourner w_0, w_1

---

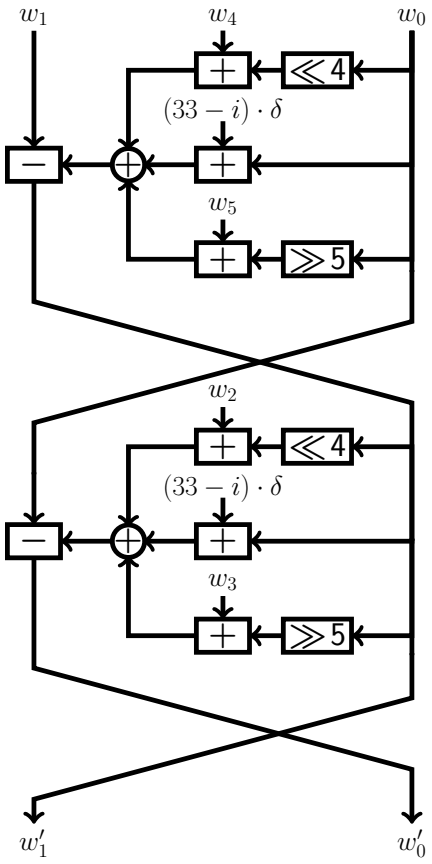
où decode_i est la fonction définie par le diagramme de la page suivante.

(2) Déchiffrer le message secret en déchiffrant les trois paires de mots avec les commandes suivantes:

```
echo "0x81FB4E63 0x58164679" | ./labo5
echo "0x4E934565 0x77DAFEB0" | ./labo5
echo "0xABD622CC 0x770C1BC5" | ./labo5
```

Le code partiel fourni s'occupe de la lecture et de l'affichage sous forme de caractères.

Fonction decode$_i$. L'algorithme de déchiffrement applique itérativement la fonction decode$_i$ illustrée par le diagramme suivant:



Chaque « fil » du diagramme transporte un mot de 32 bits. Les opérations $\boxed{+}$, $\boxed{-}$, $\boxed{\ll 4}$, $\boxed{\gg 5}$ et $\oplus$ correspondent respectivement à l'addition non signée; la soustraction non signée; le décalage logique de 4 bits vers la gauche; le décalage logique de 5 bits vers la droite; et le OU exclusif bit-à-bit. Chacune de ces opérations opère sur 32 bits. La soustraction reçoit d'abord le fil du dessus, puis celui celui de droite; donc la première soustraction est de la forme « $w_1 - \ldots$ » et la seconde soustraction de la forme « $w_0 - \ldots$ ». Les sorties w'_0 et w'_1 du diagramme correspondent respectivement aux nouvelles valeurs de w_0 et w_1 retournées par decode$_i$.

Tests. Afin de vous aider à déboguer votre programme, voici les valeurs de w_0 et w_1 que vous devriez obtenir après les trois premières itérations du déchiffrement des deux premiers mots du message secret:

	w_0	w_1
Valeurs initiales:	`0x81FB4E63`	`0x58164679`
Valeurs après decode$_1$:	`0x07FEF2E9`	`0x6B09E321`
Valeurs après decode$_2$:	`0xC581CF5E`	`0x385F2052`
Valeurs après decode$_3$:	`0x5AA6D345`	`0x8BDFCF24`

Directives.

— Votre programme doit être obtenu en complétant le code partiel ci-bas;
— Votre programme doit être remis dans un seul fichier nommé `labo5.s`;
— Ne modifiez pas le point d'entrée ainsi que le format des entrées et sorties;
— Vous devez seulement compléter le code du sous-programme `dechiffrer`;
— Vous pouvez supposer que les valeurs en entrée sont valides.

Pointage. Vous pouvez obtenir un maximum de 10 points. Vous obtenez:

— 5 points si votre programme déchiffre le message secret;
— 2 points si votre programme donne la bonne sortie sur d'autres messages secrets choisis à la correction;
— 1,5 points si votre code est bien indenté (codes d'opération, opérandes et commentaires alignés);
— 1,5 points pour la qualité et lisibilité du code (commentaires significatifs, organisation du code, etc.)

Si votre programme ne déchiffre pas le message secret correctement, des points seront tout de même accordés selon la gravité du problème.

Code partiel.

```
.global main

main:
        // Lire les deux mots à déchiffrer
        adr     x0, fmtEntree
        adr     x1, temp
        bl      scanf
        ldr     w19, temp

        adr     x0, fmtEntree
        adr     x1, temp
        bl      scanf
        ldr     w20, temp

        // Déchiffrer les deux mots
        mov     w0, w19
        mov     w1, w20
        ldr     w2, k0
        ldr     w3, k1
        ldr     w4, k2
        ldr     w5, k3
        bl      dechiffrer

        // Afficher le résultat
        mov     w19, w0
        mov     w20, w1

        adr     x0, fmtSortie
        mov     w1, w19
        mov     w2, w20
        bl      printf
```

```
        // Terminer le programme
        mov     x0, 0
        bl      exit

// Procédure de déchiffrement de l'algorithme TEA
// Entrées:
//   - mots w0 et w1 à déchiffrer (32 bits chacun)
//   - clés w2, w3, w4 et w5      (32 bits chacune)
// Sortie: mots w0 et w1 déchiffrés
dechiffrer:
        /* code ici */
        ret

.section ".rodata"
delta:      .word   0x9E3779B9
k0:         .word   0xABCDEF01
k1:         .word   0x11111111
k2:         .word   0x12345678
k3:         .word   0x90000000

fmtEntree:  .asciz  "%X"
fmtSortie:  .asciz  "%c %c\n"

.section ".data"
            .align  4
temp:       .skip   4
```