IFT209 – Programmation système

Université de Sherbrooke

Laboratoire 5

Enseignant:	Michael Blondin
Date de remise:	dimanche 17 mars 2024 à 23h59
À réaliser:	en équipe de deux
Modalités:	remettre en ligne sur Turnin

Cette semaine nous faisons une brève excursion en cryptographie afin de mettre en pratique la manipulation de bits, ainsi que la mise au point de sous-programmes.

Problème. Dans le domaine de la cryptographie, 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·e 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:

```
0x50160C73 0x7334CD38 0x7253D929 0x36DEB8D0 0x974045AF 0xA486BB01
```

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 ces clés:

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

Vous devez:

(1) Implémenter la procédure de déchiffrement de TEA:

$\delta \leftarrow 9E3779B9_{16}$ `// Constante magique globale`
$i \leftarrow 1$
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

---

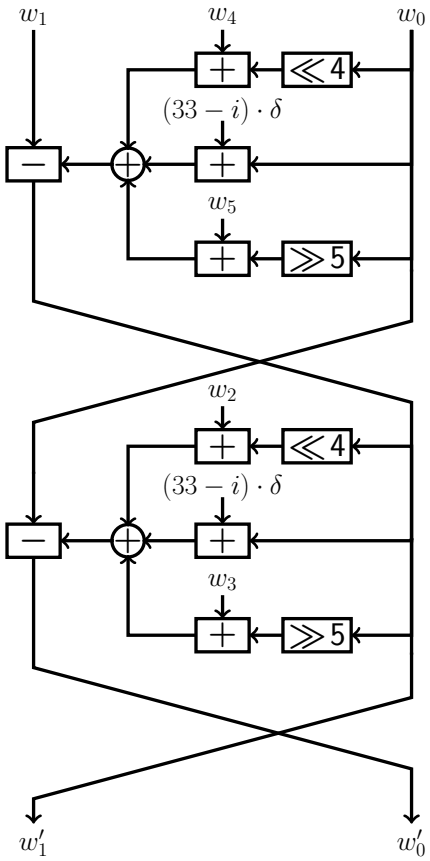
où decode_i est la fonction définie par le diagramme de la page suivante.

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

```
echo "0x50160C73 0x7334CD38" | ./labo5
echo "0x7253D929 0x36DEB8D0" | ./labo5
echo "0x974045AF 0xA486BB01" | ./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 decode$_i$ illustrée par ce diagramme:



Chaque « fil » du diagramme transporte un mot de 32 bits. Les opérations $\boxed{+}$, $\boxed{-}$, $\boxed{\ll k}$, $\boxed{\gg k}$ et $\oplus$ correspondent respectivement à l'addition non signée; la soustraction non signée; le décalage logique de k bits vers la gauche; le décalage logique de k 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	`0x50160C73`	`0x7334CD38`
Valeurs après decode$_1$	`0x8233116A`	`0xDD23DEE0`
Valeurs après decode$_2$	`0x58AA99A1`	`0xD185A39E`
Valeurs après decode$_3$	`0xD0CA3D3B`	`0xE4532A94`

Directives.

— Pour simplifier le laboratoire, nous « trichons » en retournant *deux* valeurs via `w0` et `w1`;
— Votre programme doit être obtenu en complétant le code partiel ci-dessous;
— 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`;
— Supposez que les valeurs en entrée sont valides.

Pointage. Vous pouvez obtenir jusqu'à 10 points répartis ainsi:

— 6 points pour le déchiffrage du message secret;
— 4 points pour le déchiffrage d'autres messages secrets choisis à la correction;
— 0 point pour l'indentation: codes d'opération, opérandes et commentaires alignés (recommandé);
— 0 point pour la lisibilité: commentaires, usage des registres, organisation du code, etc. (recommandé).

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

Code partiel.

```
.include "macros.s"
.global  main

main:                           // main()
    // Lire mots w0 et w1       // {
    //             à déchiffrer //
    adr     x0, fmtEntree       //
    adr     x1, temp            //
    bl      scanf               //   scanf("%X", &temp)
    ldr     w19, temp           //   w0 = temp
                                //
    adr     x0, fmtEntree       //
    adr     x1, temp            //
    bl      scanf               //   scanf("%X", &temp)
    ldr     w20, temp           //   w1 = temp
                                //
    // Déchiffrer w0 et w1      //
    mov     w0, w19             //
    mov     w1, w20             //
    ldr     w2, k0              //
    ldr     w3, k1              //
    ldr     w4, k2              //
    ldr     w5, k3              //
    bl      dechiffrer          //   w0, w1 = dechiffrer(w0, w1, w2, w3, w4, w5)
                                //
    // Afficher message secret  //
    mov     w19, w0             //
    mov     w20, w1             //
                                //
    adr     x0, fmtSortie       //
    mov     w1, w19             //
    mov     w2, w20             //
```

```
    bl      printf              //   printf("%c %c\n", w0, w1)
                                //
    // Quitter programme        //
    mov     x0, 0               //
    bl      exit                //   return 0
                                // }
/********************************************************************************
  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
                */

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

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

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