IFT209 – Programmation système

Université de Sherbrooke

Laboratoire 3

Enseignant:	Michael Blondin
Date de remise:	dimanche 17 février 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. Le but de ce laboratoire est d'écrire un programme qui lit un tableau, renverse les éléments de ce tableau, et affiche le tableau résultant. Un sous-programme pour l'affichage est déjà implémenté. Vous devez implémenter un sous-programme `read_tab` qui effectue la lecture d'un tableau, et un sous-programme `rev_tab` qui renverse les éléments d'un tableau:

Sous-programme `read_tab`

Premier paramètre:	adresse a
Effet:	— lit un entier n non signé de 64 bits (format `fmtLen`) où $1 \leq n \leq 1024$ — lit n entiers signés de 64 bits (format `fmtElem`) — stocke ces n entiers, consécutivement, à partir de l'adresse a
Valeur de retour:	n

Sous-programme `rev_tab`

Premier paramètre:	adresse d'un tableau t d'entiers signés de 64 bits
Deuxième paramètre:	nombre d'éléments du tableau t
Effet:	renverse l'ordre des éléments du tableau t en mémoire
Valeur de retour:	aucune

Vous devez également compléter `main` avec un appel à chacun de vos sous-programmes. Le tableau lu et renversé dans `main` doit être stocké dans la mémoire allouée à l'étiquette `tab:`.

Tests. Par exemple, dans un terminal, vous devriez obtenir:

```
5
6 0 -2 8 -5
[-5, 8, -2, 0, 6]

10
0 2 4 6 8 10 12 14 16 18
[18, 16, 14, 12, 10, 8, 6, 4, 2, 0]

15
1432234 543 345 65 8234 -234 -123 4 5 6 9123 -1239123 234234 -543543 1000000
[1000000, -543543, 234234, -1239123, 9123, 6, 5, 4, -123, -234, 8234, 65, 345, 543, 1432234]
```

où les deux premières lignes (en noir) sont les entrées, et la troisième ligne (en cyan) est la sortie.

Directives.

- Votre programme doit être obtenu en complétant le code partiel ci-bas;
- Votre programme doit être remis dans un seul fichier nommé `labo3.s`;
- Ne modifiez pas le point d'entrée ainsi que le format des entrées et sorties;
- Vous devez implémenter `read_tab` et `rev_tab` sous forme de *sous-programmes*, et faire les appels nécessaires dans `main`;
- Vous pouvez supposer que les valeurs en entrée sont valides; en particulier, la taille du tableau sera comprise entre 1 et 1024 (inclusivement).

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

- 1 point si votre programme lit un tableau;
- 1,5 points si votre sous-programme `read_tab` stocke correctement le tableau lu;
- 1,5 points si votre sous-programme `rev_tab` renverse les éléments d'un tableau;
- 2 points si votre programme passe les trois tests ci-dessus;
- 1 point si votre programme donne la bonne sortie sur d'autres entrées choisies à la correction;
- 1,5 points si votre code est bien indenté (codes d'opération, opérandes et commentaires alignés);
- 1,5 points pour la présence de commentaires significatifs facilitant la lecture et compréhension du code.

Code partiel.

```
.global main

.macro SAVE
  stp   x29, x30, [sp, -96]!
  mov   x29, sp
  stp   x27, x28, [sp, 16]
  stp   x25, x26, [sp, 32]
  stp   x23, x24, [sp, 48]
  stp   x21, x22, [sp, 64]
  stp   x19, x20, [sp, 80]
.endm

.macro RESTORE
  ldp   x27, x28, [sp, 16]
  ldp   x25, x26, [sp, 32]
  ldp   x23, x24, [sp, 48]
  ldp   x21, x22, [sp, 64]
  ldp   x19, x20, [sp, 80]
  ldp   x29, x30, [sp], 96
.endm

// Programme qui:
//  - lit un entier 1 ≤ n ≤ 1024 et des entiers signés a1, a2, ..., an
//  - affiche [an, ..., a2, a1]
main:                                       // int main()
                                            // {
               // Lire le tableau
               /*
                  code ici
                           */
```

```
                // Renverser le tableau
                /*
                   code ici
                            */

                // Afficher le tableau      //
                adr     x0, tab             //
                mov     x1, x19             //
                bl      print_tab           //   print_tab(tab, n)
                                            //
                mov     x0, 0               //
                bl      exit                //
                                            // }

read_tab:
                /*
                   code ici
                            */

rev_tab:
                /*
                   code ici
                            */

// Sous-programme: print_tab                //
// Entrées:                                 //
//   - adresse d'un tableau tab             //
//   - taille n de tab                      //
// Effet:  affiche les éléments de tab      //
// Sortie: aucune                           //
print_tab:                                  // void print_tab(tab[], n)
                SAVE                        // {
                mov     x19, x0             //
                mov     x20, x1             //   i = n
                adr     x0, fmtDebut        //
                bl      printf              //   afficher "["
print_tab100:                               //
                adr     x0, fmtElem         //   do {
                ldr     x1, [x19], 8        //     v = tab[n-i]
                bl      printf              //     afficher v
                sub     x20, x20, 1         //     i--
                cmp     x20, 0              //     if (i == 0)
                b.eq    print_tab200        //       break
                adr     x0, fmtSep          //     else
                bl      printf              //       afficher ", "
                b       print_tab100        //   }
print_tab200:                               //
                adr     x0, fmtFin          //   afficher "]\n"
                bl      printf              //
                RESTORE                     //
                ret                         // }
```

```
.section ".rodata"
fmtLen:     .asciz      "%lu"
fmtElem:    .asciz      "%ld"
fmtDebut:   .asciz      "["
fmtSep:     .asciz      ", "
fmtFin:     .asciz      "]\n"

.section ".bss"
            .align      8
tab:        .skip       1024*8              // Alloue 1 Kio de mémoire
```