IFT209 – Programmation système

Université de Sherbrooke

Devoir 2

Enseignant:	Michael Blondin
Date de remise:	mercredi 6 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. Soit $n \in \mathbb{N}_{\geq 2}$ un entier supérieur ou égal à 2. Nous disons que n est *premier* si ses seuls diviseurs sont 1 et n. Par exemple, les nombres premiers dans l'intervalle $[2, 30]$ sont:

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29.$$

Vous devez écrire un programme, en langage d'assemblage de l'architecture ARMv8, qui:

— lit deux entiers a et b représentés sur 64 bits et tels que $2 \leq a \leq b \leq 75\ 000$;
— affiche la quantité de nombres premiers dans l'intervalle $[a, b] = \{a, a+1, \ldots, b\}$.

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

```
2
30
10
```

```
10
20
4
```

```
5000
10000
560
```

où les deux premières lignes (en noir) sont les entrées, et la troisième ligne (en cyan) est la sortie. Vous devez donc afficher *un seul nombre*.

Directives.

— Votre programme doit être obtenu en complétant le code partiel de la page suivante;
— Votre programme doit être remis dans un seul fichier nommé `devoir2.s`;
— Ne modifiez pas le point d'entrée ainsi que le format des entrées et sorties;
— Votre programme doit terminer en 90 secondes ou moins sur toute entrée, où le *temps* est défini comme la somme du temps `user` et `sys` obtenu via la commande `echo a b | (time ./devoir2)` exécutée sur une machine du D4-1023;
— Vous pouvez supposer que les valeurs en entrée sont valides; en particulier, vous pouvez supposer que $2 \leq a \leq b \leq 75\ 000$.

Pointage. Vous obtenez:

— 1 point si votre programme lit deux nombres;
— 1 point si votre programme affiche un nombre;
— 2 points si votre programme passe les trois tests ci-dessus;
— 4 points si votre programme passe des tests sur d'autres entrées qui seront choisies à la correction;
— 2 points pour la lisibilité de votre code (indentation et commentaires).

Si vous passez les tests connus, mais que vous échouez certains des tests inconnus, notamment en excédant 90 secondes, vous pourrez obtenir jusqu'à 2.5/4 points pour les tests inconnus si votre solution est jugée quasi-complète.

Bonus.

- Vous obtenez 0.25 point bonus (absolu) à votre note finale de la session si votre programme retourne la bonne quantité *et* termine en au plus 30 secondes sur l'intervalle $[2, 200\ 000]$;
- Vous obtenez 0.5 point bonus (absolu) *additionnel* si votre programme retourne la bonne quantité *et* termine en au plus 15 secondes sur l'intervalle $[200\ 000\ 000\ 000, 200\ 000\ 010\ 000]$.

Notez que vous *ne pouvez pas* simplement pré-calculer vos résultats; nous vérifierons que vous avez bel et bien implémenté un algorithme qui fonctionne pour tout intervalle $[a, b]$.

Code partiel.

```
.global main

// Donne la quantité de nombres premiers dans un intervalle [a, b]
//
// Entrée: lit deux entiers de 64 bits a et b tels que 2 ≤ a ≤ b ≤ 75 000
// Sortie: quantité de nombres premiers dans l'intervalle [a, b]
// ...
main:
      // Lire a
      /*
         code ici
                  */

      // Lire b
      /*
         code ici
                  */

      /*
         plus de
         code ici
                  */

      bl    exit                      // quitter le programme


/* données ici au besoin */

.section ".rodata"
fmtLecture:   .asciz  "%lu"           // format des entrées a et b
fmtAffichage: .asciz  "%lu\n"         // format de la sortie
```