

IFT209 – Programmation système

Université de Sherbrooke

Laboratoire 3

Enseignant: Michael Blondin
 Date de remise: dimanche 11 février 2024 à 23h59
 À réaliser: en équipe de deux
 Modalités: remettre en ligne sur [Turnin](#)

Le but de ce laboratoire est de mettre en pratique les parcours de tableau et les accès mémoire.

Problème. Une *carte* est une matrice de dimension $m \times n$ avec un unique élément égal à « * » et dont tous les autres éléments appartiennent à {>, <, v, ^}. Le symbole « * » représente une *cible*. Les quatre autres symboles représentent respectivement un déplacement vers la *droite*, la *gauche*, le *bas* et le *haut*.

Nous cherchons à déterminer le nombre de déplacements menant du point de départ au coin supérieur gauche, c-à-d. à l'indice (0, 0), vers la cible. Par exemple, 8 déplacements mènent à la cible sur cette carte:

```
>>>v
v<<<
>>*
```

Nous supposons qu'aucun élément ne mène en dehors de la carte et qu'il y a une et une seule occurrence du symbole « * ». Cependant, une carte peut contenir une *boucle infinie* qui ne mène jamais à la cible, par ex.:

```
>>>>v
>>^*v
^<<<<
```

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

- lit deux entiers non signés de 64 bits m et n spécifiant la taille d'une carte;
- lit les $m \cdot n$ caractères formant la carte;
- affiche le nombre de déplacements menant vers la cible ou l'impossibilité de l'atteindre, selon le cas.

Tests. Considérons les trois cartes ci-dessous. Le nombre de déplacements vers la cible des deux premières cartes est de 8 et 33 respectivement, alors qu'il est impossible d'atteindre la cible de la troisième carte:

```
3 3
>>>v
v<<<
>>*
```

```
4 10
>>>>>>>>vv
v<<<<v<<<<^
v^<^<>>>vv
>>>>^*<<>
```

```
3 5
>>>>v
>>^*v
^<<<<
```

Directives.

- Votre programme doit être obtenu en complétant le code partiel fourni sur **Turnin**;
- 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;
- Supposez que les valeurs en entrée sont valides; en particulier, $1 \leq m \leq 100$ et $1 \leq n \leq 100$;
- Vous pouvez tester votre programme avec une carte sauvegardée dans un fichier `foo` avec la commande « `./labo3 < foo` »; trois cartes sont fournies, par ex. exécutez « `./labo3 < carte1.txt` »;
- Utilisez le format « `fmtChar:` » afin de lire les caractères `{>, <, v, ^, *}`. Un caractère est stocké sur *un seul octet*. Lorsque chargé dans un registre, un caractère est représenté par l'un de ces nombres¹:

caractère	>	<	v	^	*
valeur numérique	62	60	118	94	42

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

- 1 point pour la lecture de la taille d'une carte;
- 3 points pour la lecture du contenu d'une carte;
- 6 points pour l'identification du nombre de déplacements menant à la cible, ou d'une boucle infinie (vous aurez au moins 3 points si vous passez les trois tests de la page précédente);
- 0 point pour la lisibilité du code. (*Dans les devoirs, il y aura des points pour l'indentation sur quatre colonnes; l'indication de l'usage des registres; et les commentaires. Vous êtes donc encouragé-e-s à suivre ces bonnes pratiques.*)

Code partiel.

```
.global main

main:
    // Lire la taille et le contenu de la carte
    /*
       Code ici
       */

    // Parcourir la carte
    /*
       Code ici
       */

    mov     x0, 0
    bl     exit

.section ".rodata"
fmtNum:   .asciz "%lu"
fmtChar:  .asciz "%c" // l'espace est importante, ne pas l'enlever
msgAtteint: .asciz "Cible atteinte en %lu déplacements.\n"
msgBoucle: .asciz "La cible ne sera jamais atteinte.\n"

.section ".bss"
temp:     .skip 8
carte:    .skip 10000
```

1. Il s'agit de leur code ASCII; nous expliquerons ce codage plus tard dans la session.