IFT209 –Programmation système

Laboratoire 4

Enseignant: Rosa Garcia
Date de remise: indiquée sur **Turnin**
Modalités de remise: **Turnin**
À réaliser: personne seule ou en équipe de deux

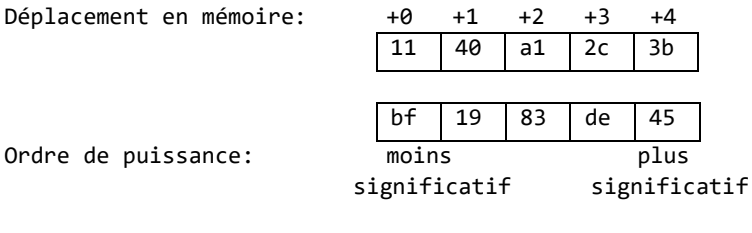
Le but de ce laboratoire est de pratiquer les notion liées au report et au débordement des nombres entiers.

Problème. Nous pouvons représenter un nombre entier de taille arbitraire (*n* bits) comme un vecteur d'octets représentant chacun 8 bits du nombre.

On réalise l'addition de deux tels nombres en additionnant tour à tour les octets les constituant à partir de l'octet le moins significatif, en prenant soin d'ajouter le report de l'addition précédente à chaque étape, puis en conservant chaque somme intermédiaire dans un vecteur représentant le résultat.

En inspectant le signe des deux nombres et du résultat (le bit 7 de leur octet le plus significatif), on peut déterminer s'il y a eu débordement.

Exemple. Soient les deux entier de 5 octets $\texttt{3b2ca14011}_{16}$ et $\texttt{45de8319bf}_{16}$, qui seront gardés en mémoire dans une suite d'octets en ordre *little-endian*:




```
à priori, report  = 0
11 + bf + report(0) = d0, report = 0
40 + 19 + report(0) = 59, report = 0
a1 + 83 + report(0) = 24, report = 1
2c + de + report(1) = 0b, report = 1
3b + 45 + report(1) = 81, report final = 0
```

Résultat:

d0	59	24	0b	81

```
Donc, 3b2ca14011 + 45de8319bf = 810b2459d0
signe(3b2ca14011) = signe(3b) = bit7(00111011) = 0
signe(45de8319bf) = signe(45) = bit7(01000101) = 0
signe(810b2459d0) = signe(81) = bit7(10000001) = 1
Le signe des deux nombres est identique, mais diffère de celui du résultat, on a donc un débordement.
```

Tâche. Vous devez écrire un sous-programme, en langage d'assemblage ARMv8, qui calcule la somme de deux nombres entiers d'une longueur de *n* octets, place le nombre résultant dans un vecteur de *n* octets, puis retourne le débordement et le report.

Votre sous-programme recevra dans `x0` et `x1` les adresses des vecteurs représentant les deux nombres à additionner, dans `x2` l'adresse du vecteur dans lequel vous écrirez le résultat et dans `x3` la longueur des nombres et du résultat.
Votre sous-programme devra également retourner dans `x0` le débordement (0 = faux, 1 = vrai). De plus, votre sous-programme doit allumer le bit C des codes condition avant de retourner s'il y a eu un report final dans votre addition. Le programme principal vérifiera l'état du bit C après l'appel à votre sous-programme.

Le code du programme principal, le code partiel du sous-programme (`entiers.as`) et le script de compilation (`Makefile`) sont fournis sur **Turnin**.

Tests. Par exemple, dans un terminal, pour le jeu d'entrées suivant:

```
5
3b2ca14011
45de8319bf
```

On obtiendrait:

```
3b 2c a1 40 11
+
45 de 83 19 bf
=
81 0b 24 59 d0
Report (C):0
Debordement (V):1
```

Rappels.

- Lors d'une addition sur 1 octet, il y a report si le résultat dépasse 255.
- Le bit C des codes condition peut être allumé si on effectue une addition sur 64 bits qui cause un report, ou une soustraction qui ne cause pas d'emprunt.
- Toute instruction de comparaison (`cmp`) modifie les codes conditions (donc le bit C).

Directives.

- Votre programme doit être obtenu en complétant le code fourni sur **Turnin**;
- Plusieurs fichiers de tests sont fournis sur **Turnin**;
- Votre programme doit être remis dans un seul fichier nommé `entiers.as`;
- Ne modifiez pas `main.as`, l'original sera utilisé pour la correction;
- Supposez que les paramètres reçus par votre sous-programme sont valides, entre autres, que le nombre d'octets est situé entre 1 et 8.

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

- 5 points si votre programme passe les cinq tests fournis sur **Turnin**;
- 3 points si votre programme affiche la bonne sortie sur d'autres tests choisis à la correction
- 2 points pour la lisibilité du code (indentation, commentaires, conventions).