

IFT209 – Programmation système

Université de Sherbrooke

Laboratoire 6

Enseignant: Michael Blondin
 Date de remise: dimanche 3 avril 2022 à 23h59
 À réaliser: en équipe de deux
 Modalités: remettre en ligne sur **Turnin**

Le but de ce laboratoire est de mettre en pratique l'arithmétique en virgule flottante et les sous-programmes.

Problème. Les plateformes comme Genote et Moodle calculent des statistiques lors de l'entrée de notes aux évaluations. En particulier, elles fournissent la *moyenne* et l'*écart-type* de notes $x_1, \dots, x_n \in \mathbb{R}_{\geq 0}$ définis par

$$\text{moy}(x) := \frac{1}{n} \cdot \sum_{i=1}^n x_i,$$

$$\text{ecart}(x) := \sqrt{\frac{1}{n} \cdot \sum_{i=1}^n (x_i - \text{moy}(x))^2}.$$

Vous devez implémenter un programme qui calcule l'écart-type de notes. Le code de lecture, d'affichage et de calcul de la moyenne est déjà fourni. En particulier, l'entrée attendue est la quantité de notes, suivie des notes en tant que nombres en virgule flottante double précision. Vous pouvez vous inspirer de l'implémentation du calcul de la moyenne.

Comme il s'agit de l'un de nos derniers programmes ARMv8, le code fourni est commenté par une traduction équivalente en C (qui serait entière compilable avec «gcc labo6.c -o labo6 -lm»). Cela n'a pas d'incidence sur le laboratoire, mais vous permettra de mieux cerner la correspondance avec du code complet de plus haut niveau (entiers, nombres en virgule flottantes, tableaux, boucles, données, fonctions, etc.)

Tests. Par exemple, dans un terminal, vous devriez obtenir ces résultats sur ces trois tests:

3	6	1
8.0	10.0	10.0
9.0	9.5	0.000000
10.0	3.75	
0.816497	8.25	
	9.5	
	7.75	
	2.105301	

où les premières lignes sont les nombres entrés au clavier, et où la **dernière ligne** correspond à la sortie affichée (avec saut de ligne). Afin d'entrer les données plus rapidement, vous pouvez utiliser ces commandes:

```
echo "3 8.0 9.0 10.0" | ./labo6
echo "6 10.0 9.5 3.75 8.25 9.5 7.75" | ./labo6
echo "1 10.0" | ./labo6
```

Directives.

- Vous devez compléter le code partiel ci-dessous;
- Votre programme doit être remis dans un seul fichier nommé `labo6.s`;
- Ne modifiez pas le point d'entrée ainsi que le format des entrées et sorties;
- Vous n'avez pas accès aux macros `SAVE` et `RESTORE` (elles ne sont de toute façon pas suffisantes);
- Supposez que les valeurs en entrée sont valides, et en particulier que $n \geq 1$.

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

- 1 point si votre sous-programme préserve bien les registres sur la pile;
- 1 point si votre sous-programme obtient la moyenne par appel de sous-programme;
- 1 point si votre sous-programme restaure bien les registres à partir de la pile;
- 4 points pour le calcul de l'écart-type;
- 1,5 points pour l'indentation du code (codes d'opération, opérandes et commentaires alignés);
- 1,5 points pour la lisibilité du code (commentaires, usage des registres, organisation du code, etc.)

Code partiel.

```

.global    main                                // #include <math.h> // sqrt
                                                // #include <stdio.h> // printf, scanf
                                                //
// Donnée statiques                          // double ecart_type(double tab[], unsigned long n);
.section ".bss"                               // double  moyenne(double tab[], unsigned long n);
        .align 8                               //
temp:    .skip 8                               // static unsigned long temp;
notes:   .skip 8000                            // static double notes[1000];
                                                //
.section ".rodata"                            //
fmtTaille: .asciz "%lu"                       // static const char* fmtTaille = "%lu";
fmtDonnee: .asciz "%lf"                       // static const char* fmtDonnee = "%lf";
fmtSortie: .asciz "%lf\n"                    // static const char* fmtSortie = "%lf\n";
                                                //
.section ".text"                              //
main:                                          // int main()
        // Lire la quantité de notes        // {
        adr    x0, fmtTaille                //
        adr    x1, temp                      //
        bl     scanf                        // scanf(fmtTaille, &temp);
        ldr    x19, temp                     // unsigned long n = temp;
                                                //
        // Lire les notes                    //
        mov    x20, 0                       // unsigned long i = 0;
        adr    x21, notes                    //
main_boucle:                                //
        cmp    x20, x19                      //
        b.hs   main_fin_boucle               // while (i < n)
                                                // {
        adr    x0, fmtDonnee                 //
        mov    x1, x21                       //
        bl     scanf                        // scanf(fmtDonnee, &notes[i]);
                                                //
        add    x20, x20, 1                   // i++;
        add    x21, x21, 8                   //
        b      main_boucle                   // }
main_fin_boucle:                            //
                                                //

```

```

// Calculer l'écart-type
adr    x0, notes
mov    x1, x19
bl     ecart_type
fmov   d8, d0
//      double ecart = ecart_type(notes, n);

// Afficher l'écart-type
adr    x0, fmtSortie
fmov   d0, d8
bl     printf
//      printf(fmtSortie, ecart);

mov    x0, 0
bl     exit
//      return 0;
// }

/*****
Entrée: adresse d'un tableau de nombres en virgule flottante double précision
       nombre d'éléments du tableau
Sortie: écart-type des éléments du tableau (en tant que population)
Usage: ...
*****/
ecart_type:
/*
CODE À COMPLÉTER
*/

/*****
Entrée: adresse d'un tableau de nombres en virgule flottante double précision
       nombre d'éléments du tableau
Sortie: moyenne des éléments du tableau
Usage: x0 - tab      d8 - acc
       x1 - n        d9 - val
       x19 - i       d10 - n (en double)
*****/
moyenne: // double moyenne(double tab[], unsigned long n)
// Préserver registres appelant
stp    x29, x30, [sp, -64]!
mov    x29, sp
stp    x19, xzr, [sp, 16] // /* on empile xzr pour compléter le double mot en trop */
stp    d8, d9, [sp, 32] // /* on empile d9 deux fois simplement car on ne peut pas
stp    d9, d10, [sp, 48] //      utiliser xzr pour remplir le dernier double mot */

// Calculer moyenne
mov    x19, 0 // unsigned long i = 0;
fmov   d8, xzr // double acc = 0.0; /* xzr car 0.0 pas supporté */
moyenne_boucle:
cmp    x19, x1 //
b.hs   moyenne_ret // while (i < n)
// {
ldr    d9, [x0, x19, lsl 3] // double val = tab[i];
fadd   d8, d8, d9 // acc += val;

add    x19, x19, 1 // i++;
b      moyenne_boucle // }
moyenne_ret:
ucvtf  d10, x1 //
fdiv   d0, d8, d10 // double moy = acc / n;

// Restaurer registres appelant
ldp    x19, xzr, [sp, 16]
ldp    d8, d9, [sp, 32]
ldp    d9, d10, [sp, 48]
ldp    x29, x30, [sp, 64]

ret    // return moy;
// }

```