

Introduction au calcul SIMD

1 Introduction

Les fichiers C nécessaires se trouvent à l'adresse www.ief.u-psud.fr/~lacas/Teaching.
Des documents sont téléchargeables à l'adresse www.ief.u-psud.fr/~lacas/Download.

Lire le document *règles de codage et routines de calcul* ([codage.pdf](#)) à l'adresse www.ief.u-psud.fr/~lacas/Teaching

Afin de simplifier le codage des différents opérateurs et de simplifier le *debug* des codes, Le codage se fait en deux étapes :

1. mise au point : codage de l'opérateur et validation sur un petit jeu de données, en comparant le résultat avec le code scalaire équivalent. C'est typiquement le corps de boucle (l'opérateur) sans les doubles boucles pour balayer les données.
2. implantation réelle : codage de la fonction destinée à traiter des données réelles, donc avec doubles boucles.

Une dernière étape consiste à comparer les performances des versions scalaires et SIMD. Vous serez évalué sur la vitesse de votre code.

Validation par des tests unitaire : c'est initialiser les données en entrée avec des valeurs telles que ces valeurs permettent de valider l'algorithme sans qu'il y ait risque de confusion (si les entrées ne sont pas indépendantes, une combinaison d'erreurs peut mener à un résultat numériquement juste). Il y a au moins trois façon de réaliser des tests unitaires en SIMD :

- algorithme *crayon-papier*,
- écrire le même algorithme, mais en scalaire
- utiliser un tableur (Microsoft Excel, OpenOffice Calc, ...)

2 Opérations simples

2.1 opérations entre registres SIMD

Soit l'opération de Multiplication-Accumulation (MAC) $y = ax + b$, avec a , x , b et y des registres SIMD flottants (`vfloat32`).

1. Codez cette opération dans la fonction `test_axb`,
2. Validez votre code en initialisant les registres à :
 $a = [1, 2, 3, 4]$, $x = [5, 6, 7, 8]$ et $b = [9, 10, 10, 12]$.
Utilisez les fonctions `init_vfloat32()` ou `_mm_set_ps()`, et `display_vfloat32()`

2.2 Addition de deux vecteurs (tableaux 1D) de registres SIMD

Soient les vecteurs \vec{X} et \vec{Y} de taille n et \vec{Z} leur somme :

$$\vec{Z} = \vec{X} + \vec{Y} = \sum_{i=0}^{n-1} X(i) + Y(i) \quad (1)$$

1. Codez cette addition vectorielle dans la fonction `add_vf32vector()`. La fonction de test (réalisant les allocations mémoire) est `test_add_vf32vector()`
Afin de séparer les instructions de calcul des instructions d'accès mémoire (principe des processeurs RISC), utilisez les registres SIMD x , y et z tels que $x = X(i)$, $y = Y(i)$ et $z = Z(i)$.
2. Comment initialiser les tableaux (via la fonction `init_vf32vector()`) pour valider votre code?

2.3 Produit scalaire de deux vecteurs (tableaux 1D) de registres SIMD : réduction totale

Soient les vecteurs \vec{X} et \vec{Y} de taille n et d leur produit :

$$d = \vec{X} \cdot \vec{Y} = \sum_{i=0}^{n-1} X(i) \times Y(i) \quad (2)$$

1. Codez ce produit scalaire *vectoriel* dans la fonction `dot_vf32vector`. La fonction de test et d'allocation mémoire est `test_dot_vf32vector`. Que faut-il faire en fin de traitement pour que le produit scalaire complet (la somme de **tous** les produits) soit dans chacun des quatre blocs du registre SIMD ?
2. Comment initialiser les tableaux (via la fonction `init_vf32vector()`) pour valider votre code ?

2.4 Somme de trois points de deux vecteurs (tableaux 1D) de registres SIMD : réduction partielle

Les opérateurs précédents réalisent des calculs entre registres SIMD. Une difficulté du calcul SIMD est la réalisation de calcul à l'intérieur d'un registre SIMD.

Soit le vecteur \vec{X} . On souhaite que le vecteur \vec{Y} contienne en tout point la somme de trois points du vecteur \vec{X} :

$$Y(i) = X(i-1) + X(i) + X(i+1) \quad (3)$$

1. On souhaite que \vec{Y} soit de taille n . Quelle doit être la taille de \vec{X} ?
2. Codez cette somme dans la fonction `sum_vf32vector`. La fonction de test et d'allocation est `test_sum_vf32vector`. Regardez la fonction `_mm_shuffle_ps()` et la macro `_MM_SHUFFLE` décrites dans la documentation Intel.
3. Validez la partie *addition de trois points* en codant, dans la fonction de test, l'addition de trois registres SIMD. Validez l'ensemble en comparant les résultats d'un tableur.

3 Filtre passe-bas

Les filtres moyenneurs (*average* en anglais) sont les plus simples des filtres passe-bas :

$$A_3 = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \quad (4)$$

1. Copiez et modifiez le code de la fonction `sum_vf32vector` pour obtenir la fonction `average3_vf32vector`.
2. Validez l'ensemble en comparant les résultats d'un tableur.