Introduction à la programmation SIMD

Vous pouvez trouver les prototypes de toutes les fonction AVX ainsi que leur coût en cycles sur le site d'Intel : `http://software.intel.com/sites/landingpage/IntrinsicsGuide`.
Il suffit de filtrer les types d'instructions AVX, AVX2 et FMA à gauche de la page. Pour compiler, utilisez la commande suivante :
g++ -O2 -mavx2 -mfma fichier.cpp -o fichier

Part 1

Copier un tableau

Le but de cet exercice est d'apprendre les bases du calcul SIMD en l'appliquant à la copie d'un tableau dans un autre.

1. Allouer deux tableaux A et B de flottants de taille N, puis initialiser A tel que $A[i] = i$.
2. Ecrire une fonction non-vectorisé qui copie le contenue de A dans B.
3. Ecrire une deuxième fonction vectorisée qui effectue la même opération.
4. Comparér le temp d'exécution total de chaque version pour 1000 appels consécutifs pour $N = 1024$.

Part 2

Produit scalaire

Le but de cet exercice est de calculer le produit scalaire de deux vecteurs x et y:

$$x^T y = \sum_{i=1}^{N} x_i y_i$$

avec vectorisation.

1. Allouer deux tableaux x et y de floatants taille N (divisible par 8), puis les initialiser.
2. Ecrire une fonction non-vectorisée qui calcule le produit scalaire de x et y.
3. Ecrire une deuxième fonction vectorisée qui effectue la même opération.
4. Ecrire une troisième fonction vectorisée qui utilise l'instruction fused-multiply-add (FMA) pour effectuer la même opération.
5. Ecrire une quatrième fonction qui fait un déroulement de la boucle par un facteur de 2 et 4 (c'est à dire, qui éffectue 2 ou 4 itérations de la version précédente dans une seule itération.
6. Comparér le temp d'exécution total de chaque version pour 1000 appels consécutifs.
7. Essayer de vectoriser le code automatiquement en rajoutant l'option de compilation "-ftree-vectorise" et tester les performances.