

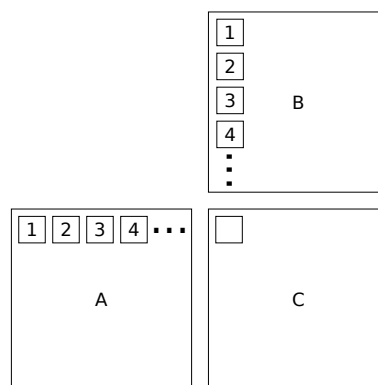
SIMD TP2

Multiplication de matrices

La liste des intrinsics SSE/AVX est disponible à l'adresse suivante : <https://software.intel.com/sites/landingpage/IntrinsicsGuide/>

1 Version scalaire

Multiplier 2 matrices A et B consiste pour chaque élément $[i, j]$ de la matrice résultat C à calculer le produit scalaire de la ligne i de la matrice A par la colonne j de la matrice B :



Voici l'algorithme scalaire correspondant à cette approche (matmul-ex1.cpp) :

```

1  #include <iostream>
2  #include <chrono>
3  #include <algorithm>
4
5  int main()
6  {
7      std::size_t dim = 1024;
8
9      float * A = new float[ dim * dim ];
10     float * B = new float[ dim * dim ];
11     float * C = new float[ dim * dim ];
12
13     std::generate_n( A, dim * dim, []() { return std::rand()%5; } );
14     std::generate_n( B, dim * dim, []() { return std::rand()%5; } );
15
16     auto start = std::chrono::system_clock::now();
17
18     // Pour chaque élément (i,j) de la matrice C on calcul le produit scalaire
19     // de la ligne i par la colonne j.
20     for( std::size_t i = 0 ; i < dim ; ++i )
21     {
22         for( std::size_t j = 0 ; j < dim ; ++j )
23         {
24             for( std::size_t k = 0 ; k < dim ; ++k )
25             {
26                 C[ i * dim + j ] += A[ i * dim + k ] * B[ k * dim + j ];
27             }
28         }
29     }
30
31     auto stop = std::chrono::system_clock::now();

```

```

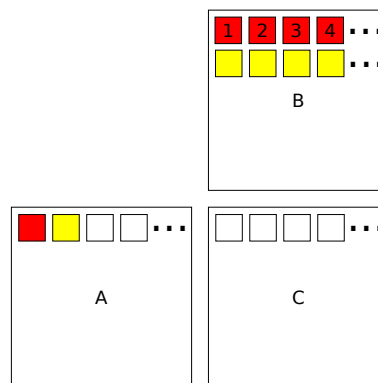
32
33     std::cout <<
34         std::chrono::duration_cast<std::chrono::milliseconds>( stop - start ).count()
35         << "ms" << std::endl;
36
37     delete [] A;
38     delete [] B;
39     delete [] C;
40
41     return 0;
42 }

```

2 Version scalaire 2

La version ci-dessus n'est pas directement vectorisable car on accède aux éléments de la matrice B par colonne dont les données ne sont pas contiguës en mémoire donc impossibles à charger efficacement de manière vectorielle.

Pour pouvoir vectoriser efficacement, il faut partir d'un algorithme scalaire parcourant les matrices dans l'ordre ci-dessous :



Écrire le code correspondant dans le fichier `matmul-ex2.cpp` et vérifier que le résultat est correct.

3 Version vectorielle

Pour simplifier les versions vectorisées, on suppose que la dimension de la matrice est un multiple de 8.

1. Écrire la version vectorisée dans le fichier `matmul-ex3.cpp` en utilisant les intrinsics SSE.
2. Comparer les performances entre les différentes versions pour une matrice de taille 2048x2048 :
 - Quel est le gain apporté par la réorganisation des boucles ?
 - Quel est le gain apporté par la vectorisation SSE ?
 - Les gains sont-ils ceux attendus ?