Java Avancé

Single instruction, Multiple Data (SIMD)

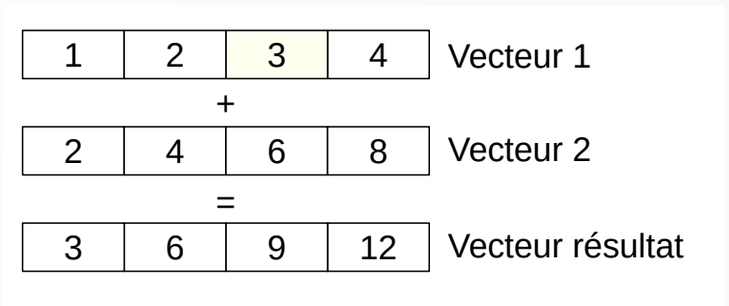
Guillaume Cau

Instructions SIMD

- Les instructions SIMD sont des instructions qui peuvent exécuter plusieurs opérations en parallèles sur des données différentes.

- Les opérations peuvent être :
  - Les opérations bit à bit (et, ou, non)
  - Les additions
  - Les soustractions
  - Les multiplications
  - Les divisions

Illustration :



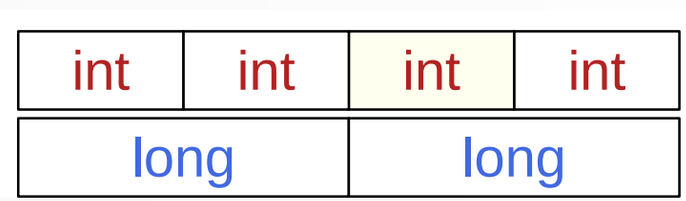
La « Vector Api »

- L’Api de vectorisation java fournit un ensemble de fonction permettant d’utiliser la vectorisation sur les processeurs possédant une architecture permettant de l’utiliser (AVX).

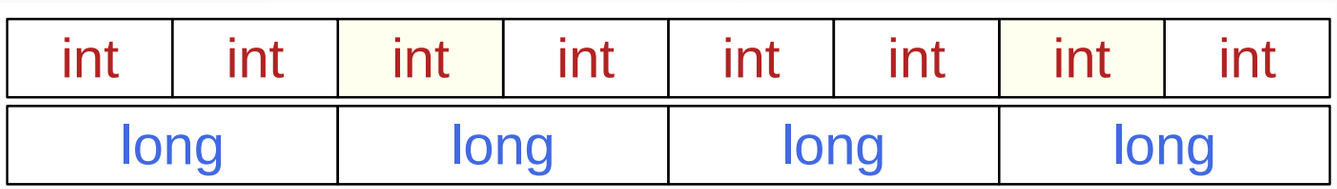
- Un vecteur ne peut être que d’un seul type à la fois (int, long, float etc.). Java fournit une classe pour chaque vecteur :
  - ShortVector
  - IntVector
  - FloatVector
  - Etc…

La taille des « Vector »

- La taille des vecteurs dépend de l’architecture de la machine :
  - AVX(1) : 128 bits



  - AVX2 : 256 bits



  - AVX3 : 512 bits

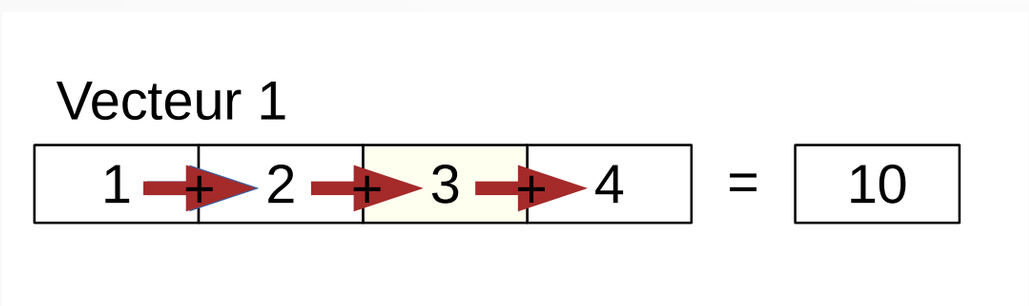
    Etc…

Création d'un objet Vector

- Pour commencer à utiliser les vecteurs, la première étape consiste à en obtenir un :
  - Créer un vecteur rempli de zéros : IntVector.zero()
  - Créer un vecteur à partir d'un tableau : IntVector.fromArray(SPECIES, array, index)
    - **SPECIES** étant un objet stockant les informations matérielles
    - **array** étant le tableau à transformer en vecteur
    - **index** étant l'index du tableau à partir du quel la transformation commence.
  - Pour obtenir **SPECIES**, il faut faire :
    VectorSpecies<Integer> SPECIES = IntVector.SPECIES_PREFERRED;
    L'objet **SPECIES** est important car il contient les informations matérielles comme la tailles des lanes.
  - La fonction **fromArray** transforme la partie du tableau entre **index** et **SPECIES.length()-1** en vecteur.

L’opération « reduceLanes »

- L’opération **reduceLanes** permet de retourner le résultat d’une opération entre chaque élément du même vecteur.
- int result = vector.reduceLanes(<VectorOperator>)
  - <VectorOperator> étant le type d’opération à effectuer (VectorOperators.ADD, VectorOperators.SUB etc.)
- Exemple : vector1.reduceLanes(VectorOperators.ADD)



L’opération « lanewise »

- L’opération **lanewise** permet de retourner le résultat d’une opération entre plusieurs vecteurs.
- IntVector resultVector = vector1.lanewise(<VectorOperator>, vector2);
- Exemple :

  IntVector resultVector = vector1.lanewise(VectorOperators.ADD, vector2);