**Question 4 :**

**4 kernels CUDA**

A : Divise la matrice en 1D n blocs contenant p threads où un thread i calcule composante vecteur résultat à l’indice i

256 threads par bloc

?? Calculer la norme euclidienne intermédiaire dans chaque bloc (communication) pour accélérer le calcul ??

B : Contient 1 bloc de 1 thread pour calculer la norme euclidienne totale du vecteur résultat

C : Contient n blocs de p threads où un thread i calcule la composante normée du vecteur résultat à l’indice i

D : Contient 1 bloc de 1 thread pour calculer l’écart quadratique entre le vecteur résultat actuel et précédent

**Question 5 :**

**Communications CPU-GPU entre kernels :**

Pas besoin de communication auprès du CPU entre l'exécution des kernels A, B, C et D.

Il n’y a pas de système de contrôle de la part du CPU entre ces kernels : il les lance séquentiellement sans contrôle.

**Communications CPU-GPU entre chaque itération :**

Le CPU doit récupérer la valeur de l’écart entre le vecteur résultat précédent et actuel pour gérer la condition d’arrêt de la méthode de la puissance.

*Mémoire global :*

Accessible par tous les blocs, tous les threads

*Mémoire partagés :*

Accessible par tous les threads d’un même bloc

*Mémoire local :*

Exclusif pour un thread

**Question 6:**

**Kernel A :** Chaque thread calcul fait matmul de 1 ligne

\_\_global\_\_ void matmulKernel(float\* d\_A,float\* d\_B,float\* d\_C,int n) {

unsigned int j = blockDim.x\*blockIdx.x+threadIdx.x;

if (i < n && j < n){

float temp=0;

for(int k=0; k<n; k++)

temp = temp + d\_A[j \* n + k] \* d\_B[k \* n];

d\_C[j] = temp;

}

}

**Kernek B :**

**Kernel C :**

**Kernel D :**

DIM:

gridDim.x (nb bloc dim x)

blockDim.x (nb thread par bloc dim x)

blockIdx.x (indice bloc sur la grille)

threadIdx.x (indice thread dim du bloc)

int main(){

#define N 1024

#define TAILLE\_BLOC\_X 16