# Question CM #3 : Présentation CUDA

**1 : Quels sont les quatre types d’organisation de flux de données et de flux d’instructions ?**

* Single instruction : SI
* Multiple Instruction : MI
* Single Data : SD
* Multiple Data : MD

**2 : Quel est le type d’organisation d’une architecture mono-CPU ?**

Une architecture mono-CPU est de type SISD, car un CPU exécute une instruction et traite une donnée à la fois.

**3 : Quel est le type d’organisation d’une architecture multi-CPU ?**

Une architecture multi-CPU est de type MIMD, car les différents CPU peuvent exécuter plusieurs instructions et plusieurs données en même temps.

**4 : Quel est le type d’organisation d’une architecture GPU ?**

Une architecture GPU est de type SIMD, car un CPU exécute une même instruction sur plusieurs données en parallèle.

**5 : Parmi les CPU et les GPU lesquels sont les plus efficaces pour exécuter des algorithmes séquentiels et parallèles** ?

Les CPU sont les plus efficaces pour exécuter des algorithmes séquentiels.  
Les GPU sont les plus efficaces pour exécuter des algorithmes parallèles.

**6 : Quels sont les différentes strates de la structure physique d’un GPU ?**

* GPU
* GPC
* SM
* Warps
* CUDA Cores

**7 : Quel est le nom d’une fonction qui s’exécute sur un GPU ?**

Une fonction qui s’exécute sur un GPU s’appelle un Kernel

**8 : Quelles sont les différentes strates logicielles de l’exécution d’un kernel CUDA ?**

* Grid
* Block
* Thread

**9 : Sur quelle unité physique est exécutée la grille ?**

La grille est exécutée sur le GPU

**10 : Sur quelles unités physiques sont exécutés les blocks ?**

Les blocks sont exécutés sur les SM (streaming multi processor)

**11 : Sur quelles unités physiques sont exécutés les threads ?**

Les threads sont exécutés sur les CUDA Cores

**12 : Qu’est-ce qu’un Warp ?**

Un Warp est un regroupement de 32 CUDA Cores qui s’exécutent en parallèles.

**13 : Quel est le nom de l’extension d’un fichier CUDA ?**

L’extension des fichiers CUDA est .cu

**14 : Quel est le nom du compilateur CUDA ?**

Le compilateur CUDA s’appelle NVcc.

**15 : Par quoi commence la déclaration d’un Kernel ?**

La déclaration d’un Kernel commence par \_\_global\_\_

**16 : Quels sont les 2 paramètres à spécifier entre triples chevrons précédents les paramètres passés au kernel ?**

Les 2 paramètres à spécifier sont la taille de la grille et la taille des blocks.

**17 : Donnez un exemple type d’un appel de kernel.**

Kernel <<< nBlocs, threadsparbloc>>> (arguments) ;

**18 : A quoi correspond threadIdx.x dans un kernel qui traite des données 1D?**

threadIdx.x correspond à l’id du thread dans son block.

**19 : A quoi correspond blockDim.x dans un kernel qui traite des données 1D?**

blockDim.x correspond à la dimension du block dans lequel est exécuté le thread.

**20 : A quoi correspond blockIdx.x dans un kernel qui traite des données 1D?**

blockIdx.x correspond à l’id du block dans lequel est exécuté le thread.

**21 : Comment obtient-on l’index d’un thread dans un kernel dont la grille et les blocks sont de dimension 1 ?**

Index = blockIdx.x \* blockDim.x + threadIdx.x

**22 : Quelle est la fonction à utiliser pour allouer de la mémoire sur le GPU ?**

La fonction à utiliser est CudaMalloc()

**23 : Quelle est la fonction à utiliser pour libérer de la mémoire allouée sur le GPU ?**

La fonction à utiliser est CudaFree()

**24 : Quelle est la fonction à utiliser pour effectuer des transferts mémoire entre la mémoire CPU et la mémoire GPU et à quoi correspondent les 4 paramètres à utiliser ?**

La fonction à utiliser est CudaMemCpy(). Les 4 paramètres à utiliser sont :

* Le pointeur de destination
* Le pointeur de source
* Le nombre d’octets copiés
* Le type de transfert

**25 : Lors d’un transfert mémoire entre le CPU et un GPU, quel est le type de mémoire est utilisé du côté du CPU et du GPU ?**

Lors d’un transfert mémoire entre le CPU et un GPU le type de mémoire utilisé du côté du CPU est la RAM. Du côté du GPU, le transfert peut être fait sur la mémoire globale, la mémoire constante ou la mémoire texture.

**26 : Quels sont les différents types de mémoire sur lesquels les threads peuvent lire des données et/ou écrire des données ?**

* R/W per-thread registers
* R/W per-thread local memory
* R/W per-block shared memory
* R/W per-grid global memory
* Read only per grid constant memory
* Read only per-grid texture memory

**27 : Quels sont les types de mémoire accessible par absolument tous les threads ?**

La mémoire globale, la mémoire constante et la mémoire textures sont accessible par tous les threads d’un GPU.

**28 : Quel est le type de mémoire uniquement accessible par tous les threads d’un même block ?**

La mémoire partagée ou shared memory est uniquement accessible par les threads d’un même block.

**29 : Quels sont les types de mémoire qui sont réservés pour chaque threads ?**

Les registres et la mémoire locale sont les deux types de mémoire qui sont réservés pour chaque threads.

**30 : Est-ce que le GPU peut directement exécuter des instructions sur des données présentes sur la RAM de l’host et pourquoi ?**

Non, car il n’a pas directement accès aux données présente sur la RAM de l’host, il faut d’abord transférer les données sur la mémoire globale du GPU