Questions sur Loop Unrolling / Questions on Loop Unrolling

Question 1

Pourquoi utilise-t-on le Loop Unrolling en CUDA C ? / Why is Loop Unrolling used in CUDA C?

a) Réduire la latence des branchements conditionnels / Reduce latency of conditional branching.
b) Améliorer l'utilisation de la mémoire globale / Improve global memory usage.
c) Diminuer la taille du code binaire généré / Decrease the size of the binary code.
d) Éviter l'utilisation de la mémoire partagée / Avoid shared memory usage.

Réponse/Answer: a) Réduire la latence des branchements conditionnels / Reduce latency of conditional branching.

Question 2

Quel est le résultat de l'exécution du code suivant ? / What is the output of the following code?

```c
#include <stdio.h>

__global__ void loopUnrollingExample(int *arr) {
    int idx = threadIdx.x;
    arr[idx] = idx * idx;  // Première itération
    arr[idx + 1] = (idx + 1) * (idx + 1);  // Deuxième itération
}

int main() {
    int arr[8], *d_arr;
    cudaMalloc(&d_arr, 8 * sizeof(int));
    loopUnrollingExample<<<1, 4>>>(d_arr);
    cudaMemcpy(arr, d_arr, 8 * sizeof(int), cudaMemcpyDeviceToHost);
    cudaFree(d_arr);

    for (int i = 0; i < 8; i++) printf("%d ", arr[i]);
    return 0;
}
```

a) 0 1 4 9 16 25 36 49
b) 0 1 4 9 16 25 36 0
c) 0 1 4 9 0 0 0 0
d) 0 0 0 0 0 0 0 0

Réponse/Answer: b) 0 1 4 9 16 25 36 0.

Question 3

Quel facteur d'enroulement serait le plus efficace pour optimiser ce code ? / What unrolling factor would be most effective for optimizing this code?

```c
__global__ void reduceKernel(int *data) {
   int idx = threadIdx.x + blockIdx.x * blockDim.x;
   int sum = 0;

   for (int i = 0; i < 4; i++) {
      sum += data[idx + i];
   }
   data[idx] = sum;
}
```

a) 2
b) 4
c) 8
d) Aucun, la boucle est déjà optimale / None, the loop is already optimal.

Réponse/Answer: b) 4.

Question 4

Dans quel cas le Loop Unrolling est-il inefficace ? / When is Loop Unrolling ineffective?

a) Lorsque le nombre d'itérations est très élevé / When the number of iterations is very high.
b) Lorsque le nombre d'itérations est inconnu à la compilation / When the iteration count is unknown at compile-time.
c) Lorsque la mémoire partagée est saturée / When shared memory is saturated.
d) Lorsque des threads dans un warp divergent fortement / When threads in a warp diverge heavily.

Réponse/Answer: b) Lorsque le nombre d'itérations est inconnu à la compilation / When the iteration count is unknown at compile-time.

Question 5

Identifiez l'erreur dans le code suivant utilisant le Loop Unrolling. / Identify the error in the following code using Loop Unrolling.

```c
__global__ void loopUnrollingError(int *arr, int N) {
   int idx = threadIdx.x + blockIdx.x * blockDim.x;
```

```
    if (idx < N) {
        arr[idx] = idx;
        arr[idx + 1] = idx + 1;  // Potentielle erreur
    }
}
```

a) Le facteur d'enroulement est trop grand / The unrolling factor is too large.
b) idx + 1 peut dépasser la limite N / idx + 1 might exceed the limit N.
c) Le tableau arr est mal alloué / The array arr is incorrectly allocated.
d) Aucun problème / No issue.

Réponse/Answer: b) idx + 1 peut dépasser la limite N / idx + 1 might exceed the limit N.

Question 6

Comment améliorer ce code en utilisant le Loop Unrolling ? / How can this code be improved using Loop Unrolling?

```c
__global__ void sumArray(int *data, int N) {
    int idx = threadIdx.x + blockIdx.x * blockDim.x;
    int sum = 0;

    for (int i = idx; i < N; i += blockDim.x) {
        sum += data[i];
    }
    data[idx] = sum;
}
```

a) Diviser la boucle en plusieurs itérations non conditionnelles / Divide the loop into multiple unconditional iterations.
b) Utiliser `__syncthreads()` pour synchroniser les threads / Use `__syncthreads()` to synchronize threads.
c) Utiliser un facteur d'enroulement fixe pour traiter plusieurs éléments par thread / Use a fixed unrolling factor to process multiple elements per thread.
d) Ajouter une vérification de limites pour éviter les erreurs d'accès mémoire / Add boundary checks to avoid memory access errors.

Réponse/Answer: c) Utiliser un facteur d'enroulement fixe pour traiter plusieurs éléments par thread / Use a fixed unrolling factor to process multiple elements per thread.