Write a CUDA Program for : 1. Addition of two large vectors

#include <stdio.h>

#define N 8

#define numThread 2 // 2 threads in a block

#define numBlock 4 // 4 blocks

/\*

\* 1.

\* The 'kernel' function that will be executed on the GPU device hardware.

\*/

\_\_global\_\_ void add( int \*a, int \*b, int \*c ) {

// the initial index that this thread will work on

int tid = blockDim.x \* blockIdx.x + threadIdx.x;

while (tid < N) {

c[tid] = a[tid] + b[tid]; // The actual computation done by the thread

tid += blockDim.x; // Increment this thread's index by the number of threads per block:

// in this small case, each thread would then have a tid > N

}

}

/\*

\* The main program that directs the execution of vector add on the GPU

\*/

int main( void ) {

int \*a, \*b, \*c; // The arrays on the host CPU machine

int \*dev\_a, \*dev\_b, \*dev\_c; // The arrays for the GPU device

// 2.a allocate the memory on the CPU

a = (int\*)malloc( N \* sizeof(int) );

b = (int\*)malloc( N \* sizeof(int) );

c = (int\*)malloc( N \* sizeof(int) );

// 2.b. fill the arrays 'a' and 'b' on the CPU with dummy values

for (int i=0; i<N; i++) {

a[i] = i;

b[i] = i;

}

// 2.c. allocate the memory on the GPU

cudaMalloc( (void\*\*)&dev\_a, N \* sizeof(int) );

cudaMalloc( (void\*\*)&dev\_b, N \* sizeof(int) );

cudaMalloc( (void\*\*)&dev\_c, N \* sizeof(int) );

// 2.d. copy the arrays 'a' and 'b' to the GPU

cudaMemcpy( dev\_a, a, N \* sizeof(int),

cudaMemcpyHostToDevice );

cudaMemcpy( dev\_b, b, N \* sizeof(int),

cudaMemcpyHostToDevice );

// 3. Execute the vector addition 'kernel function' on th GPU device,

// declaring how many blocks and how many threads per block to use.

add<<<numBlock,numThread>>>( dev\_a, dev\_b, dev\_c );

// 4. copy the array 'c' back from the GPU to the CPU

cudaMemcpy( c, dev\_c, N \* sizeof(int),

cudaMemcpyDeviceToHost );

// verify that the GPU did the work we requested

bool success = true;

int total=0;

printf("Checking %d values in the array.\n", N);

for (int i=0; i<N; i++) {

if ( c[i] = (a[i] + b[i])) {

printf( "Addition: %d + %d = %d\n", a[i], b[i], c[i] );

success = false;

}

total += 1;

}

if (success) printf( "We did it, %d values correct!\n", total );

// free the memory we allocated on the CPU

free( a );

free( b );

free( c );

// free the memory we allocated on the GPU

cudaFree( dev\_a );

cudaFree( dev\_b );

cudaFree( dev\_c );

return 0;

}

/// OUTPUT ///////

(base) cg@cg-ThinkCentre-neo-50s-Gen-3:~/Downloads$ **nvcc add.cu -o add**

(base) cg@cg-ThinkCentre-neo-50s-Gen-3:~/Downloads$ **./add**

Checking 8 values in the array.

Addition: 1 + 1 = 2

Addition: 2 + 2 = 4

Addition: 3 + 3 = 6

Addition: 4 + 4 = 8

Addition: 5 + 5 = 10

Addition: 6 + 6 = 12

Addition: 7 + 7 = 14