%%cu

using namespace std;

#include<stdio.h>

#include<iostream>

#define w 32

#define h 32

#define N w\*h

\_\_global\_\_ void mult(int \*d\_a , int \*d\_b ,int \*d\_c,int \*d\_thread, int count) ;

int main() {

int count=2;

int \*a = new int[count\*count];

int \*b = new int[count\*count];

int \*c = new int[count\*count];

int \*thread = new int[count\*count];

int \*d\_a,\*d\_b,\*d\_c,\*d\_thread;

for(int i=0;i<count\*count;i++) {

a[i]=rand() % 4;

b[i]=rand() % 4;

cout<<a[i]<<" "<<b[i]<<"\n";

}

if(cudaMalloc(&d\_a,sizeof(int)\*count\*count)!=cudaSuccess) {

cout<<"Not okay at a";

}

if(cudaMalloc(&d\_b,sizeof(int)\*count\*count)!=cudaSuccess) {

cout<<"Not okay at b";

cudaFree(d\_a);

}

if(cudaMalloc(&d\_c,sizeof(int)\*count\*count)!=cudaSuccess) {

cout<<"Not okay at c";

cudaFree(d\_a);

cudaFree(d\_b);

}

if(cudaMalloc(&d\_thread,sizeof(int)\*count\*count)!=cudaSuccess) {

cout<<"Not okay at thread";

cudaFree(d\_a);

cudaFree(d\_b);

}

if(cudaMemcpy(d\_a,a,sizeof(int)\*count\*count,cudaMemcpyHostToDevice)!=cudaSuccess){

cout<<"Not copied at a";

}

if(cudaMemcpy(d\_b,b,sizeof(int)\*count\*count,cudaMemcpyHostToDevice)!=cudaSuccess){

cout<<"Not copied at b";

}

dim3 blockofthreads(count,count);

dim3 gridofblocks(1,1);

mult<<<gridofblocks,blockofthreads>>>(d\_a,d\_b,d\_c,d\_thread,count);

if(cudaMemcpy(c,d\_c,sizeof(int)\*count\*count,cudaMemcpyDeviceToHost)!=cudaSuccess) {

cout<<"Problem with result";

}

if(cudaMemcpy(thread,d\_thread,sizeof(int)\*count\*count,cudaMemcpyDeviceToHost)!=cudaSuccess) {

cout<<"Problem with result";

}

else{

for(int i=0;i<count\*count;i++) {

cout<<c[i]<<" ";

}

cout<<"Printing total threads and unique ids";

for(int i=0;i<count\*count;i++) {

cout<<thread[i]<<" "<<"\n";

}

}

return 0;

}

\_\_global\_\_ void mult(int \*d\_a , int \*d\_b , int \*d\_c,int\*d\_thread, int count) {

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

int col=blockIdx.y\*blockDim.y + threadIdx.y;

d\_thread[row\*count+col]=row\*count+col;

int sum=0;

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

sum=sum+d\_a[row\*count+i]\*d\_b[i\*count+col];

}

d\_c[row\*count+col]=sum;

}

//sample output -'3 2\n1 3\n1 3\n2 0\n9 9 8 3 Printing total threads and unique ids0 \n1 \n2 \n3 \n'