#include<stdio.h>

#include<cuda.h>

//#include<conio.h>

#include<malloc.h>

int \*a,\*b,\*c;

int \*ga,\*gb,\*gc;

int numOfBlocks;

int blocksize=256;

float et;

int sizerowa , sizeclma;

\_\_global\_\_ void MatrixMatrixadd (int\* ga,int \*gb,int\* gc,int size)

{

int i=threadIdx.x + (blockIdx.x\*blockDim.x);

if(i<size)

{

gc[i] =ga[i]+gb[i];

}

}

int main()

{

printf(“Enter user matrix size\n”);

scanf(“%d %d”, &sizerowa , & sizeclma);

a=(int\*)malloc((sizerowa\* sizeclma)\*sizeof(int));

b=(int\*)malloc((sizerowa \* sizeclma)\*sizeof(int));

c=(int\*)malloc((sizerowa \* sizeclma)\*sizeof(int));

int i=0;

for(i=0;i<( sizerowa\* sizeclma);i++)

{

\*(a+i)=i;

}

for(i=0;i<( sizerowa\* sizeclma);i++)

{

\*(b+i)=i;

}

numOfBlocks=( sizerowa\* sizeclma) /blocksize;

if((sizerowa\* sizeclma)%blocksize>0) numOfBlocks++;

cudaMalloc((void\*\*)&ga,( sizerowa\* sizeclma) \*sizeof(int));

cudaMalloc((void\*\*)&gb,( sizerowa\* sizeclma)\*sizeof(int));

cudaMalloc((void\*\*)&gc,( sizerowa\* sizeclma)\*sizeof(int));

cudaMemcpy(ga,a,( sizerowa\* sizeclma)\*sizeof(int),cudaMemcpyHostToDevice);

cudaMemcpy(gb,b, sizerowa\* sizeclma \*sizeof(int),cudaMemcpyHostToDevice);

cudaEvent\_t start,stop;

cudaEventCreate(&start);

cudaEventCreate(&stop);

cudaEventRecord(start,0);

MatrixMatrixadd<<<numOfBlocks,blocksize>>>(ga,gb,gc,( sizerowa\* sizeclma));

cudaEventRecord(stop,0);

cudaEventSynchronize(stop);

cudaEventElapsedTime(&et,start,stop);

cudaEventDestroy(start);

cudaEventDestroy(stop);

cudaDeviceSynchronize();

cudaMemcpy(c,gc,( sizerowa\* sizeclma) \*sizeof(int),cudaMemcpyDeviceToHost);

printf(" parallel %f\n",et);

/\*

for (i=0 ; i<,( sizerowa\* sizeclma) ; i++)

{

printf("%d ", \*(a+i)) ;

}

printf("\n") ;

for (i=0 ; i<( sizerowa\* sizeclma); i++)

{

printf("%d ", \*(b+i)) ;

}

printf("\n") ;

for (i=0 ; i<,( sizerowa\* sizeclma) ; i++)

{

printf("%d ", \*(c+i)) ;

}

\*/

//getch();

return 0 ;

}