#include<stdio.h>

#include<cuda.h>

#include<conio.h>

#include<malloc.h>

int sizev, sizex, sizey ;

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

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

int numOfBlocks;

int blocksize=256;

float et;

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

{

extern \_\_shared\_\_ int pk[];

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

if (i<3) {\*(pk+i) = \*(ga+i);} ;

\_\_syncthreads();

if(i<size)

{

int k=0;

int sum=0;

for(k=0;k<3 ; k++)

{

sum = sum + pk[k]\*(\*(gb+((size\*k)+i))) ;

}

gc[i] = sum;

}

}

int main()

{

sizev = 3 ;

sizex = 3 ;

sizey = 1000000;

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

b=(int\*)malloc((sizex\*sizey)\*sizeof(int));

c=(int\*)malloc((1\*sizey)\*sizeof(int));

int i=0,j=0 ,sum=0;

for(i=0;i<sizev;i++)

{

\*(a+i)=i;

}

for(i=0;i<(sizex\*sizey);i++)

{

\*(b+i)=i;

}

numOfBlocks=sizey/blocksize;

if(sizey%blocksize>0) numOfBlocks++;

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

cudaMalloc((void\*\*)&gb,(sizex\*sizey)\*sizeof(int));

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

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

cudaMemcpy(gb,b,(sizex\*sizey)\*sizeof(int),cudaMemcpyHostToDevice);

cudaEvent\_t start,stop;

cudaEventCreate(&start);

cudaEventCreate(&stop);

cudaEventRecord(start,0);

VectorMatrix<<<numOfBlocks,blocksize,sizev\*sizeof(int)>>>(ga,gb,gc,sizey);

cudaEventRecord(stop,0);

cudaEventSynchronize(stop);

cudaEventElapsedTime(&et,start,stop);

cudaEventDestroy(start);

cudaEventDestroy(stop);

cudaDeviceSynchronize();

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

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

/\*

for (i=0 ; i<sizev ; i++)

{

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

}

printf("\n") ;

for (i=0 ; i<(sizex\*sizey) ; i++)

{

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

}

printf("\n") ;

for (i=0 ; i<sizey ; i++)

{

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

}

\*/

getch();

return 0 ;

}