Practica CUDA Nº 3

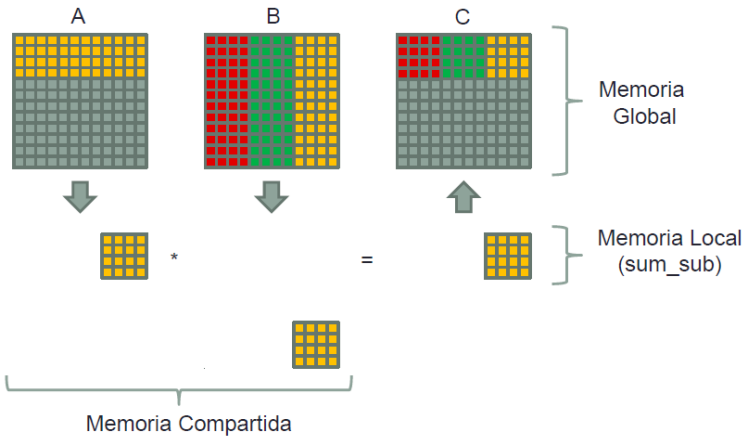
M. en C. Sandra Luz Morales Güitrón.

Realice los siguientes ejercicios, realice su reporte y no olvide las conclusiones.

1. Multiplicación de Matrices usando Memoria Compartida. (complete el código)

```
//Multiplicacion de Matrices en Memoria Compartida (SM)
//Ver SDK (matrixMul), Each block must be contain BLOCK_SIZE*BLOCK_SIZE threads
__global__ void Multiplica_Matrices_SM(float *C,float *A,float *B,
                                       int nfil,int ncol)
{
    //Indices de Bloques
    int bx = blockIdx.x;
    int by = blockIdx.y;
    //Indices de Hilos
    int tx = threadIdx.x;
    int ty = threadIdx.y;

    // Indice de la primer submatriz A procesada por el bloque
    int aBegin = ncol * BLOCK_SIZE * by;
    // Indice de la ultima submatriz A procesada por el bloque
    int aEnd   = aBegin + ncol - 1;
    // Tamaño de paso para iterar sobre las submatrices de A
    int aStep  = BLOCK_SIZE;
    // Indice de la primer submatriz B procesada por el bloque
    int bBegin = BLOCK_SIZE * bx;
    // Tamaño de paso para iterar sobre las submatrices de B
    int bStep  = BLOCK_SIZE * ncol;
    // Csub is used to store the element of the block sub-matrix
    // that is computed by the thread
    float sum_sub = 0.0f;
    // Ciclo sobre todas las submatrices de A y B
    for (int a = aBegin,b = bBegin;a <= aEnd;a += aStep, b += bStep) {
        //Memoria compartida para la submatriz A
        __shared__ float As[BLOCK_SIZE][BLOCK_SIZE];
        //Memoria compartida para la submatriz B
        __shared__ float Bs[BLOCK_SIZE][BLOCK_SIZE];

        // Almacenar las matrices desde la memoria global
        // a la memoria compartida; cada hilo almacena
        // un elemento de cada matriz
        As[ty][tx] = A[a + ncol * ty + tx];
        Bs[ty][tx] = B[b + ncol * ty + tx];
        // Sincronizamos los hilos para asegurar que se han cargado las matrices
        __syncthreads();

        // Multiplicamos las dos matrices
        #pragma unroll
        for (int k = 0; k < BLOCK_SIZE; k++)
            sum_sub += As[ty][k] * Bs[k][tx];
        // Sincronizamos para asegurar que el calculo anterior
        // se halla completadc, antes de almacenar las nuevas submatrices
        __syncthreads();
    }

    // Guardamos el resultado en la memoria global
    // Cada hilo guarda un elemento
    int c = ncol * BLOCK_SIZE * by + BLOCK_SIZE * bx;
    C[c + ncol * ty + tx] = sum_sub;
}
```

Multiplicación de Matrices usando Memoria Compartida



2.- Producto escalar

Realiza el producto escalar de dos vectores de forma paralela. El producto escalar de dos vectores se define la sumatoria del producto entre los elementos del vector.

Tome el ejemplo de http://lsi.ugr.es/jmantas/pdp/tutoriales/tutorial_mpi.php?tuto=04_producto_escalar y conviértalo a CUDA.

3.- MergeSort

Resolver el problema de la ordenación de forma paralela.

Tome el ejemplo de http://lsi.ugr.es/jmantas/pdp/tutoriales/tutorial_mpi.php?tuto=06_mergesort y conviértalo a CUDA.