Ejercicios

Invertir una lista en paralelo: Manejo de index y vectores unidimensionales

Manejo de indices y de vectores unidimensionales

- Devuelve el valor del thread en el eje x dentro del bloque en el que se este trabjando
- cudaMallo

Ejemplo invertir una lista

```
#include <stdio.h>
#define TPB 1024  //Thread por bloque
__global__ void invertirLista(int *in, int *out){   //El apuntador in es el arreglo que contiene el dato que vamoa a mover,
 //el apuntador out, esta vacio  y se ira llenando
    const int idxIn = threadIdx.x;         //Obtenemos el numero de indice del thread en el que se esta trabajando
    const int idxOut = blockDim.x - 1 - idxIn; // Calculamos el valor del nuevo indice al que se moverá el dato
    out[idxOut] =  in[idxIn];                    //Ponemos el dato en su nuevo index
}
int main(){
    unsigned int size =  TPB * sizeof(int);      //Esta variable contiene el espacio de memoria requerido por el sistema
    int* h_in = (int*) malloc(size);             // Reservamos espacio de memmoria dinamica dentro del CPU
    int i;
    for( i = 0; i < TPB; i++){                   //Llenamos el arreglo con lo datos.
        h_in[i] =  i;
    }
    int *d_in; cudaMalloc((void**)&d_in, size);        //Reservamos el espacio de memoria de la GPU
    int *d_out; cudaMalloc((void**)&d_out, size);    //Reservamos el espacio de memoria de la GPU
```

Ejemplo: Invertir una lista en paralelo

```
//Copiamos la memoria del CPU al GPU
    cudaMemcpy(d_in, h_in, size, cudaMemcpyHostToDevice);
    invertirLista<<<1, TPB>>>(d_in, d_out);              // Hacemos el lanzamiento de los kernel
    int* h_out = (int*) malloc(size);                    // Hacemos otra reservacion de  memoria
    cudaMemcpy(h_out, d_out, size, cudaMemcpyDeviceToHost); // Copiamos los datos del GPu al CPU
    cudaFree(d_in); cudaFree(d_out);                     //Liberamos el espacio de memoria  de la GPU

    printf(" IN / OUT \n");
    for(i = 0; i < TPB; i++){                            //Imprimimos el vector resultado.
        printf(" %d / %d \n", h_in[i], h_out[i]);
    }
    free(h_in); free(h_out);                             // Liberamos el espacio de la CPU
    return 0;
}
```