

ARQUITECTURA DE COMPUTADORES

1.CÓDIGO

```
/*
 * ARQUITECTURA DE COMPUTADORES
 * 2º Grado en Ingenieria Informatica
 *
 * ENTREGA Básico 4:
 * >> Arrays multidimensionales
 *
 * Alumno: Antonio Alonso Briones
 * Fecha: 13/10/2024
 *
 */

// includes
#include <stdio.h>
#include <stdlib.h>
#include <cuda_runtime.h>

// declaración de funciones
__host__ void generar_matriz_random(int* matriz, int filas, int columnas)
{
    for (int i = 0; i < filas * columnas; i++)
    {
        matriz[i] = rand() % 9 + 1; // Valores entre 1 y 9
    }
}

__global__ void procesar_matriz(int* matriz_inicial, int* matriz_final, int
filas, int columnas)
{
    // Cálculo de los índices de fila y columna
    int idxColumna = threadIdx.x;
    int idxFila = threadIdx.y;

    if (idxFila < filas && idxColumna < columnas)
    {
        int idxGlobal = idxColumna + idxFila * columnas;

        // Si la columna es par (idxColumna % 2 == 0), se mantiene el valor
        if (idxColumna % 2 == 0)
        {
            matriz_final[idxGlobal] = matriz_inicial[idxGlobal];
        }
        else
        {
            matriz_final[idxGlobal] = 0;
        }
    }
}

int main(int argc, char** argv)
{
    // Declaración de variables
    int filas, columnas;
    printf("Introduce el número de filas: ");
```

```

scanf("%d", &filas);
printf("Introduce el número de columnas: ");
scanf("%d", &columnas);

// Obtener el número de dispositivos CUDA
int deviceCount;
cudaGetDeviceCount(&deviceCount);
if (deviceCount == 0)
{
    printf("No se han encontrado dispositivos CUDA compatibles.\n");
    return 1;
}

// Obtener propiedades del dispositivo (usamos el dispositivo 0)
cudaDeviceProp deviceProp;
cudaGetDeviceProperties(&deviceProp, 0);

// Mostrar propiedades del dispositivo
printf("\n> Propiedades del dispositivo seleccionado:\n");
printf("Nombre: %s\n", deviceProp.name);
printf("Capacidad de cómputo: %d.%d\n", deviceProp.major, deviceProp.minor);
printf("Número de multiprocesadores: %d\n", deviceProp.multiProcessorCount);
printf("Número máximo de hilos por bloque: %d\n", deviceProp.maxThreadsPerBlock);
printf("Máximo tamaño de bloque (x, y, z): (%d, %d, %d)\n",
        deviceProp.maxThreadsDim[0], deviceProp.maxThreadsDim[1], deviceProp.maxThreadsDim[2]);
printf("Máximo tamaño de la malla (x, y, z): (%d, %d, %d)\n",
        deviceProp.maxGridSize[0], deviceProp.maxGridSize[1], deviceProp.maxGridSize[2]);

// Definir el tamaño del bloque bidimensional
dim3 tamBloque(columnas, filas);

// Verificar que el tamaño del bloque no exceda los máximos permitidos
if (tamBloque.x > deviceProp.maxThreadsDim[0] || tamBloque.y > deviceProp.maxThreadsDim[1])
{
    printf("El tamaño del bloque excede el máximo permitido.\n");
    // Ajustar el tamaño del bloque si es necesario
    if (tamBloque.x > deviceProp.maxThreadsDim[0])
        tamBloque.x = deviceProp.maxThreadsDim[0];
    if (tamBloque.y > deviceProp.maxThreadsDim[1])
        tamBloque.y = deviceProp.maxThreadsDim[1];
}

// Calcular el número de bloques necesarios (en este caso, un solo bloque)
dim3 tamMalla(1, 1);

// Mostrar el número de hilos lanzados en cada eje
printf("\n> Configuración del kernel:\n");
printf("Dimensiones del bloque (hilos por bloque): (%d, %d, %d)\n", tamBloque.x, tamBloque.y, tamBloque.z);
printf("Dimensiones de la malla (bloques): (%d, %d, %d)\n", tamMalla.x, tamMalla.y, tamMalla.z);

// Declaración de punteros para host y device
int* hst_matriz_inicial, * hst_matriz_final;
int* dev_matriz_inicial, * dev_matriz_final;

```

```

int tamMatriz = filas * columnas * sizeof(int);

// Reserva de memoria en el host
hst_matriz_inicial = (int*)malloc(tamMatriz);
hst_matriz_final = (int*)malloc(tamMatriz);

// Inicialización de la matriz inicial
printf("\n> Generando la matriz inicial...\n");
srand(0); // Semilla fija para reproducibilidad
generar_matriz_random(hst_matriz_inicial, filas, columnas);

// Reserva de memoria en el device
cudaMalloc((void**)&dev_matriz_inicial, tamMatriz);
cudaMalloc((void**)&dev_matriz_final, tamMatriz);

// Copiar la matriz inicial al device
cudaMemcpy(dev_matriz_inicial, hst_matriz_inicial, tamMatriz, cudaMemcpyHostToDevice);

// Lanzamiento del kernel
printf("> Procesando la matriz en el device...\n");
procesar_matriz << <tamMalla, tamBloque >> > (dev_matriz_inicial,
dev_matriz_final, filas, columnas);

// Sincronizar para esperar a que el kernel termine
cudaDeviceSynchronize();

// Copiar datos desde el device al host
cudaMemcpy(hst_matriz_final, dev_matriz_final, tamMatriz, cudaMemcpyDeviceToHost);

// Impresión de resultados
printf("\n> MATRIZ INICIAL:\n");
for (int i = 0; i < filas; i++)
{
    for (int j = 0; j < columnas; j++)
    {
        printf("%2d ", hst_matriz_inicial[j + i * columnas]);
    }
    printf("\n");
}

printf("\n> MATRIZ FINAL:\n");
for (int i = 0; i < filas; i++)
{
    for (int j = 0; j < columnas; j++)
    {
        printf("%2d ", hst_matriz_final[j + i * columnas]);
    }
    printf("\n");
}

// Liberar memoria
free(hst_matriz_inicial);
free(hst_matriz_final);
cudaFree(dev_matriz_inicial);
cudaFree(dev_matriz_final);

// Salida
printf("*****\n");
printf("<pulsa [INTRO] para finalizar>");
getchar(); getchar();
return 0;

```

```
}
```

2. RESULTADO DE LA COMPILACIÓN

```
kernelcu - X
Básico 4 - (Ámbito global) - main(int argc, char ** argv)
Salida
Mostrar salida de Depurar
'Básico 4.exe' (Win32): 'C:\Users\aloan\source\repos\Básico 4\Debug\Básico 4.exe' cargado. Símbolos cargados.
'Básico 4.exe' (Win32): 'C:\Windows\System32\ntdll.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\kernel32.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\kernelbase.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\apphelp.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\ucrntime140d.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\ucrtbased.dll' cargado.
El subproceso 1512 terminó con código 0 (0x0).
'Básico 4.exe' (Win32): 'C:\Windows\System32\nvccuda.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\advapi32.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\mmdev.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\sechost.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\bcrypt.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\gdi32.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\win32u.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\gdi32full.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\msvcrt_win.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\ucrtbase.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\user32.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\lsm32.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\DXCore.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\DriverStore\FileRepository\nvhml.inf_and64_c4739b7ed91feb94\nvccuda64.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\shlwapi.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\version.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\msasn1.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\cryptnet.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\cryptbase.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\wldp.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\combase.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\oleaut32.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\drvstore.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\devobj.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\cfgmgr32.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\cfgmgr32.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\cfgmgr32.dll' descargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\cfgmgr32.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\cfgmgr32.dll' descargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\nvapi64.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\setupapi.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\shell32.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\ole32.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\DriverStore\FileRepository\nvhml.inf_and64_c4739b7ed91feb94\nvdxgml64.dll' cargado.
'Básico 4.exe' (Win32): 'C:\Windows\System32\kernel.appcore.dll' cargado.
El subproceso 29956 terminó con código 0 (0x0).
El subproceso 12384 terminó con código 0 (0x0).
El subproceso 13612 terminó con código 0 (0x0).
El subproceso 16424 terminó con código 0 (0x0).
El subproceso 13156 terminó con código 0 (0x0).
El programa '[17812] Básico 4.exe' terminó con código 0 (0x0).
```

3. SALIDA POR PANTALLA

```
Consola de depuración de Mi - X + -
Máximo tamaño de bloque (x, y, z): (1024, 1024, 64)
Máximo tamaño de la malla (x, y, z): (2147483647, 65535, 65535)

> Configuración del kernel:
Dimensiones del bloque (hilos por bloque): (5, 5, 1)
Dimensiones de la malla (bloques): (1, 1, 1)

> Generando la matriz inicial...
> Procesando la matriz en el device...

> MATRIZ INICIAL:
3 7 8 0 9
8 5 3 2 4
4 3 9 3 9
1 9 6 1 2
7 5 2 5 8

> MATRIZ FINAL:
3 0 8 0 9
8 0 3 0 4
4 0 9 0 9
1 0 6 0 2
7 0 2 0 8

*****
<pulsa [INTRO] para finalizar>

C:\Users\aloan\source\repos\Básico 4\Debug\Básico 4.exe (proceso 17812) se cerró con el código 0 (0x0).
Para cerrar automáticamente la consola cuando se detiene la depuración, habilite Herramientas ->Opciones ->Depuración ->Cerrar la consola automáticamente al detenerse la depuración.
Presione cualquier tecla para cerrar esta ventana. . .
```