UNIVERSIDAD PANAMERICANA

Universidad Panamericana
Campus Guadalajara
Escuela de ingenierías

Fundamentos de programación en paralelo

REPORTE DE PRÁCTICA

IDENTIFICACIÓN DE LA PRÁCTICA

Práctica	**9**	**Nombre de la práctica**	**Warp**
Fecha	**28/09/2021**	**Nombre del profesor**	**Alma Nayeli Rodríguez Vázquez**
Nombre del estudiante		**Mariana Ávalos Arce**	

OBJETIVO

El objetivo de esta práctica consiste en implementar un kernel para mostrar la configuración de los hilos considerando su agrupación en las unidades básicas conocidas como Warp.

PROCEDIMIENTO

Realiza la implementación siguiendo estas instrucciones.

Realiza un programa en C/C++ utilizando CUDA en el que implementes un kernel que imprima la siguiente información para cada hilo:

- El índice de hilo (threadIdx.x)
- El índice de bloque en x (blockIdx.x)
- El índice de bloque en y (blockIdx.y)
- El índice global (gId)
- El índice del warp al que pertenece (warpId)
- El índice global de bloque (gBlockId)

Para el lanzamiento del kernel deberás considerar la siguiente configuración:

- Bloques de 42 hilos en 1D
- Una malla de 2 por 2 bloques
- El kernel debe ser como el siguiente: `__global__ void warpDetalles();`

IMPLEMENTACIÓN

Agrega el código de tu implementación aquí.

```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"

#include <stdio.h>
#include <stdlib.h>

__host__ void checkCUDAError(const char* msg) {
	cudaError_t error;
	cudaDeviceSynchronize();
	error = cudaGetLastError();
	if (error != cudaSuccess) {
		printf("ERROR %d: %s (%s)\n", error, cudaGetErrorString(error), msg);
	}
}

__global__ void warpDetails() {
```

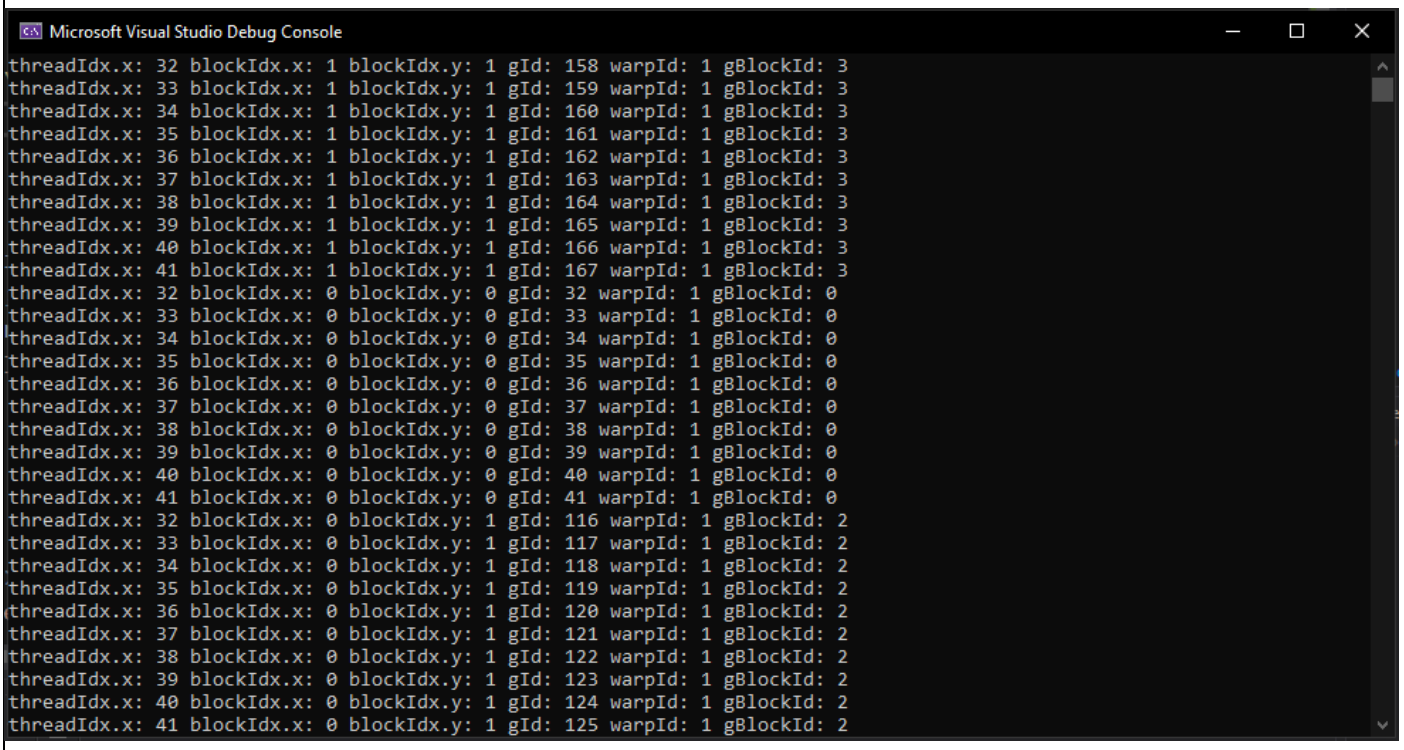
```
	int gId = blockIdx.y * (gridDim.x * blockDim.x) + (blockIdx.x * blockDim.x) + threadIdx.x;
	int warpId = threadIdx.x / 32; // index of warp per block, not unique
	int gBlockId = blockIdx.y * gridDim.x + blockIdx.x;
	printf("threadIdx.x: %d blockIdx.x: %d blockIdx.y: %d gId: %d warpId: %d gBlockId: %d\n",
threadIdx.x, blockIdx.x, blockIdx.y, gId, warpId, gBlockId);
}

int main() {
	dim3 block(42);
	dim3 grid(2, 2);
	warpDetails << < grid, block >> > ();
	checkCUDAError("Error at kernel");

	return 0;
}
```

RESULTADOS

Agrega la imagen de la consola con el despliegue de los resultados obtenidos.



CONCLUSIONES

Escribe tus observaciones y conclusiones.

Esta práctica permite visualizar la cantidad de threads lanzados contra la cantidad de threads que se requerían, y al tener una configuración que no es múltiplo de 32, se nota explícitamente los hilos desperdiciados. Además, esta práctica retoma esos índices de bloque, de malla e hilo para hacer cálculos de nuevos ids que nos permitan ubicar estos hiles y el desperdicio de recursos.