**REPORTE DE PRÁCTICA**

**IDENTIFICACIÓN DE LA PRÁCTICA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Práctica** | **1** | **Nombre de la práctica** | | **Propiedades del device** |
| **Fecha** | **11/08/21** | **Nombre del profesor** | | **Alma Nayeli Rodríguez Vázquez** |
| **Nombre del estudiante** | | | **Mariana Ávalos Arce** | |

**OBJETIVO**

|  |
| --- |
| El objetivo de esta práctica consiste en conocer las propiedades de la tarjeta gráfica mediante la utilización de la estructura cudaDeviceProp. |

**PROCEDIMIENTO**

|  |
| --- |
| Realiza la implementación siguiendo estas instrucciones. |
| Realiza un programa en C/C++ utilizando CUDA para conocer las propiedades de la tarjeta gráfica que estás utilizando. Para ello crea una variable del tipo cudaDeviceProp y muestra en pantalla la siguiente información de tu dispositivo:   1. Nombre 2. Número de Multiprocesadores (MP) 3. Máximo número de bloques por MP 4. Máximo número de hilos por MP 5. Máximo número de hilos por bloque 6. Tamaño máximo de la malla por cada dimensión 7. Máximo número de hilos por bloque en cada dimensión 8. Tamaño de un warp |

**IMPLEMENTACIÓN**

|  |
| --- |
| Agrega el código de tu implementación aquí. |
| #include "cuda\_runtime.h"  #include "device\_launch\_parameters.h"  #include <stdio.h>  int main()  {  int device = 0;  int\* count = &device;  cudaGetDeviceCount(count);  printf("Device count: %d\n", device);  cudaDeviceProp properties;  cudaDeviceProp\* pProperties = &properties;  cudaGetDeviceProperties(pProperties, device - 1);  printf("Name: %s\n", properties.name);  printf("multiProcessorCount: %d\n", properties.multiProcessorCount);  printf("maxBlocksPerMultiProcessor: %d\n", properties.maxBlocksPerMultiProcessor);  printf("maxThreadsPerMultiProcessor: %d\n", properties.maxThreadsPerMultiProcessor);  printf("maxThreadsPerBlock: %d\n", properties.maxThreadsPerBlock);  printf("maxGridSize x axis: %d\n", properties.maxGridSize[0]);  printf("maxGridSize y axis: %d\n", properties.maxGridSize[1]);  printf("maxGridSize z axis: %d\n", properties.maxGridSize[2]);  printf("maxThreadsDim x axis: %d\n", properties.maxThreadsDim[0]);  printf("maxThreadsDim y axis: %d\n", properties.maxThreadsDim[1]);  printf("maxThreadsDim z axis: %d\n", properties.maxThreadsDim[2]);  printf("warpSize: %d\n", properties.warpSize);  printf("major: %d\n", properties.major);  printf("minor: %d\n", properties.minor);  return 0;  } |

**RESULTADOS**

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

**CONCLUSIONES**

|  |
| --- |
| Escribe tus observaciones y conclusiones. |
| Se puede concluir bastante de esta práctica, ya que su objetivo fue conocer las propiedades de la tarjeta gráfica o GPU que cada máquina posee. Una de estas conclusiones a las que llego es que puedes manipular o ajustar las propiedades maxThreadsPerBlock y maxBlocksPerMultiProcessor, mientras que la multiplicación de estos números sea menor o igual a la propiedad maxThreadsPerMultiProcessor. Además, resultó que los bloques y el grid son tridimensionales, lo cual por fin me explicó muchas dudas de la sintaxis que había visto en códigos de ejemplo donde se llama un kernel y se especificaban unos parámetros enteros que ahora entiendo qué significan. Al conocer el Major Property y la Minor Property me hicieron mucho sentido ahora los problemas que tuve al instalar CUDA inicialmente, pues para configurar el entorno de trabajo, conocer estas propiedades es fundamental. |