

Nombre: Christopher García

Carné: 20541

Fecha de Entrega: 3 de noviembre, 2023.

Descripción: De forma individual, realice los siguientes ejercicios y prácticas para comenzar a familiarizarnos con CUDA y programación en GPUs. Para cada inciso, incluya evidencia de su procedimiento o salida (capturas de pantalla, etc., en un PDF). No olvide adjuntar su código. Dependiendo de si usa su computador o el Lab, puede que algunos comandos cambien levemente, por lo que deben estar atentos en dado caso para indagar y utilizar el comando adecuado.

Entregables: Deberá entregar un documento con las respuestas a las preguntas planteadas en cada ejercicio (incluyendo diagramas o screenshots si es necesario), junto con todos los archivos de código que programe debidamente comentados e identificados. La entrega de la hoja es individual.

Materiales: necesitará una máquina con GPU Nvidia. Puede utilizar Google Collab para esta actividad.

Contenido

Ejercicio 1 (50 puntos)

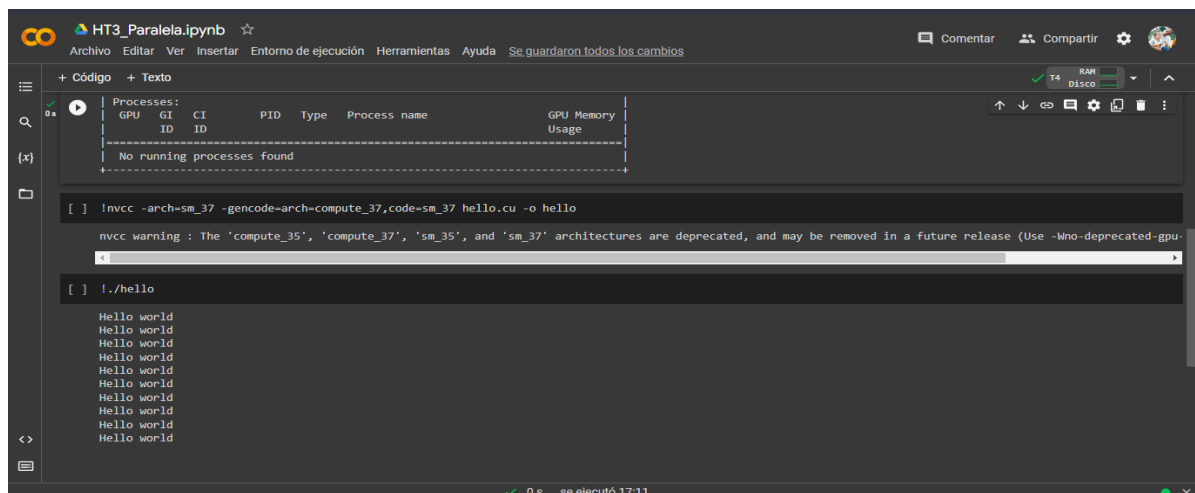
El programa `hello.cu` ilustra la forma básica del modelo de ejecución para CUDA. Realice las siguientes acciones para comprender el efecto de la configuración del kernel y su relación con el Compute Capability de una tarjeta.

1. Compile el programa (ignore la advertencia sobre código deprecado en caso le salga):

```
$ nvcc hello.cu -o hello
```

2. Ejecute el programa. Observe cuántas veces se imprime el mensaje y su conexión con la configuración de la llamada al kernel – `hello<<<g,b>>>()`:

```
$ ./hello
```



The screenshot shows a Google Colab notebook titled "HT3_Paralela.ipynb". The interface includes a menu bar with options like "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", "Herramientas", and "Ayuda". Below the menu, there are tabs for "Código" and "Texto". The "Código" tab is active, showing a terminal window with the following content:

```
[ ] In[ ]: nvcc -arch=sm_37 -gencode=arch=compute_37,code=sm_37 hello.cu -o hello
nvcc warning : The 'compute_35', 'compute_37', 'sm_35', and 'sm_37' architectures are deprecated, and may be removed in a future release (Use -Wno-deprecated-gpu-
[ ] In[ ]: ./hello
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
```

The terminal output shows the successful compilation of the program and the execution of the resulting binary, which prints "Hello world" multiple times. The status bar at the bottom indicates "0 s" and "se ejecutó 17:11".

- Modifique el programa para correr 2 bloques de 1024 hilos. Modificarlo también para que imprima su nombre y carnet. Busque en el despliegue de consola el mensaje del último hilo de la serie (1023).

```
11 #include <stdio.h>
12 #include <cuda.h>
13
14 __global__ void hello()
15 {
16     int tid = blockIdx.x * blockDim.x + threadIdx.x;
17     printf("Hello World by Christopher García (20541)\n");
18
19     if (tid == 1023) {
20         printf("Thread 1023, Christopher García (20541)");
21     }
22 }
23
24 int main()
25 {
26     hello<<<2, 1023>>>>();
27     cudaDeviceReset();
28     return 0;
29 }
```

- Busque en el sitio de Nvidia el Compute Capability de la tarjeta que poseen las máquinas del Laboratorio (o de la computadora que está utilizando). Escriba acá el valor de CC y busque la tabla resumen con las características técnicas del CC:

Compute Capability: 7.5

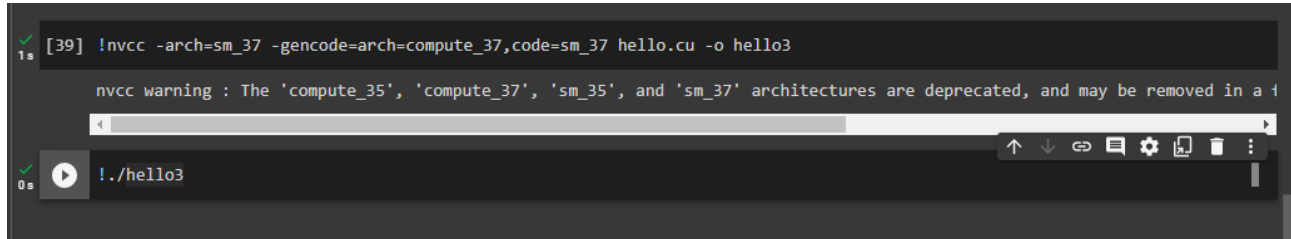
Sun Oct 29 00:08:23 2023

NVIDIA-SMI 525.105.17 Driver Version: 525.105.17 CUDA Version: 12.0									
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	ECC		
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute	M.		
							MIG	M.	
0	Tesla T4	Off	00000000:00:04.0	Off			0		
N/A	51C	P8	17W / 70W	0MiB / 15360MiB	0%	Default			N/A

Processes:							
GPU	GI	CI	PID	Type	Process name	GPU Memory	
ID	ID					Usage	
No running processes found							

<https://developer.nvidia.com/cuda-gpus>
En este link se obtuvo el CC en base a Tesla T4

5. Modifique el programa para correr 1 bloque de 2048 hilos.



The screenshot shows a terminal window with a dark background. The first command entered is `nvcc -arch=sm_37 -gencode=arch=compute_37,code=sm_37 hello.cu -o hello3`, which is preceded by a green checkmark and a '1s' timer. Below the command, a warning message is displayed: `nvcc warning : The 'compute_35', 'compute_37', 'sm_35', and 'sm_37' architectures are deprecated, and may be removed in a future release.` The second command entered is `./hello3`, preceded by a green checkmark and a '0s' timer. The terminal window has a scrollbar on the right and a status bar at the bottom.

- No se imprime nada, según la documentación de NVIDIA (<https://www.nvidia.com/es-es/data-center/tesla-t4/>) explica que la cantidad máxima de hilos por bloque es de 1024 entonces esta tarea es “imposible” y no se ejecuta.

6. Busque en la tabla de CC los siguientes datos de la GPU que está utilizando:

- i. Warp size
 - 32
- ii. Maximum number of threads per block
 - 1024
- iii. Maximum dimensionality of a grid of thread blocks
 - 3
- iv. Maximum size per grid dimension
 - 1024
- v. Maximum dimensionality of a thread block
 - 3
- vi. Maximum size per block dimension
 - 1024

- CUDA C Programming Guide. (2023). Nvidia.com.
<https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#features-and-technical-specifications>

Ejercicio 2 (50 puntos)

El programa `hello2.cu` ilustra la forma para calcular un identificador global al momento de usar hilos que pertenecen a bloques diferentes. Realice las siguientes acciones para comprender el efecto de la configuración del kernel y su relación con la forma de calcular el ID único de los hilos.

1. Descargue, compile y ejecute `hello2.cu`. Observe la relación de la configuración de la llamada al kernel con la geometría de los hilos y el resultado. Escriba la respuesta a los dos enunciados:
 - i. **Máximo ID de los hilos: 239**
 - ii. **Ejecución de los hilos en orden: La ejecución de hilos no tiene un orden específico, se imprimen en bloques de diez elementos de números ordenados entre 0 y 239. Lo que logró percibir de la generación de `myID` es que se obtiene el hilo de su bloque y dimensión correspondiente**
2. Observe que la fórmula genérica para cálculo del ID global está en los comentarios. Modifique el programa para que imprima también su nombre y carné. Luego, realice la siguiente modificación al programa (al inicio del main) y use la fórmula genérica para derivar el nuevo cálculo de ID:

`dim3 g (4,2);`

`dim3 b (32,16);`

`hello <<<g, b>>>();`

```

Hello world from 4883 by Christopher García 20541
Hello world from 4884 by Christopher García 20541
Hello world from 4885 by Christopher García 20541
Hello world from 4886 by Christopher García 20541
Hello world from 4887 by Christopher García 20541
Hello world from 4888 by Christopher García 20541
Hello world from 4889 by Christopher García 20541
Hello world from 4890 by Christopher García 20541
Hello world from 4891 by Christopher García 20541
Hello world from 4892 by Christopher García 20541
Hello world from 4893 by Christopher García 20541
Hello world from 4894 by Christopher García 20541
Hello world from 4895 by Christopher García 20541
Hello world from 3136 by Christopher García 20541
Hello world from 3137 by Christopher García 20541
Hello world from 3138 by Christopher García 20541
Hello world from 3139 by Christopher García 20541
Hello world from 3140 by Christopher García 20541
Hello world from 3141 by Christopher García 20541
Hello world from 3142 by Christopher García 20541
Hello world from 3143 by Christopher García 20541
Hello world from 3144 by Christopher García 20541
Hello world from 3145 by Christopher García 20541
Hello world from 3146 by Christopher García 20541
Hello world from 3147 by Christopher García 20541
Hello world from 3148 by Christopher García 20541
Hello world from 3149 by Christopher García 20541
Hello world from 3150 by Christopher García 20541
Hello world from 3151 by Christopher García 20541

```

```

12 #include <cuda.h>
13
14 __global__ void hello ()
15 {
16     int myID = (blockIdx.z * gridDim.x * gridDim.y +
17               blockIdx.y * gridDim.x +
18               blockIdx.x * blockDim.x * blockDim.y * blockDim.z +
19               threadIdx.z * blockDim.x * blockDim.y +
20               threadIdx.y * blockDim.x +
21               threadIdx.x);
22
23 // Simplification of above
24 //grid: 3D --- z,y,x: all dims and blockids
25 //block: 1D -- x
26 //int myID = ( blockIdx.z * gridDim.x * gridDim.y +
27 //            blockIdx.y * gridDim.x +
28 //            blockIdx.x * blockDim.x +
29 //            threadIdx.x);
30
31 printf ("Hello world from %i by Christopher García 20541\n", myID);
32 }
33
34 int main ()
35 {

```

3. Revise nuevamente la información del Compute Capability respecto a las dimensiones máximas de hilos-bloque en x, y, & z para una grilla. Cree una configuración para lanzar exitosamente el kernel para procesar 100,000 datos. (Sugerencia: busque una configuración que lance como mínimo 100,000 hilos. Modifique el kernel para que imprima el mensaje únicamente si es el ID global máximo)

```

[13] Invc -arch=sm_37 -gencode=arch=compute_37,code=sm_37 hello2_1.cu -o hello2C
nvcc warning : The 'compute_35', 'compute_37', 'sm_35', and 'sm_37' architecture
[14] ./hello2C
Max Threads Per Block (x): 1024
Max Threads Per Block (y): 1024
Max Threads Per Block (z): 64
Max Blocks Per Grid (x): 2147483647
Max Blocks Per Grid (y): 65535
Max Blocks Per Grid (z): 65535
Hello world from 100351 by Christopher García 20541

```

```

50 printf("Max Blocks Per Grid (y): %d\n", prop.maxGridSize[1]); // 65535
51 printf("Max Blocks Per Grid (z): %d\n", prop.maxGridSize[2]); // 65535
52
53 int totalData = 100000;
54 int maxThreadsX = 1024;
55 int maxThreadsY = 1024;
56 int maxThreadsZ = 64;
57 int maxBlocksX = 2147483647;
58 int maxBlocksY = 65535;
59 int maxBlocksZ = 65535;
60
61 dim3 threadsPerBlock(
62     min(totalData, maxThreadsX),
63     min(1, maxThreadsY),
64     min(1, maxThreadsZ)
65 );
66
67 dim3 blocksPerGrid(
68     min((totalData + threadsPerBlock.x - 1) / threadsPerBlock.x, maxBlocksX),
69     min(1, maxBlocksY),
70     min(1, maxBlocksZ)
71 );
72
73 hello <<< blocksPerGrid, threadsPerBlock >>> ();
74 cudaDeviceReset();
75
76 return 0;
77 }

```