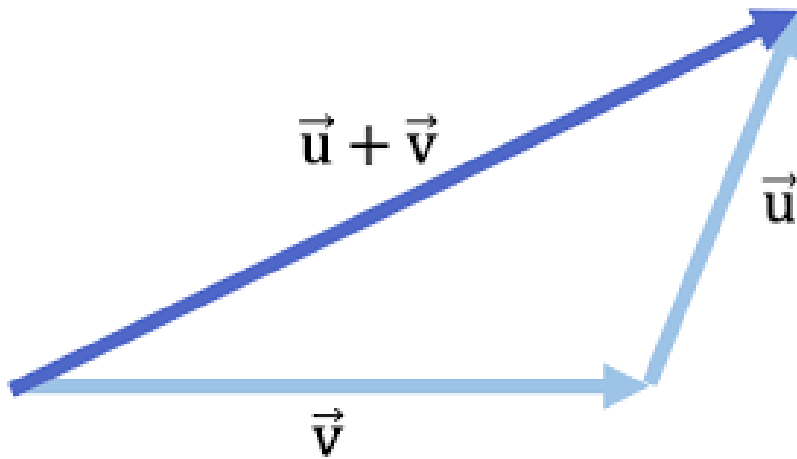


Práctica 06. Suma de vectores



U.A.Q. Fac. de Informática
Dra. Sandra Luz Canchola Magdaleno

Correo: sandra.canchola@uaq.mx

Dra. Reyna Moreno Beltrán

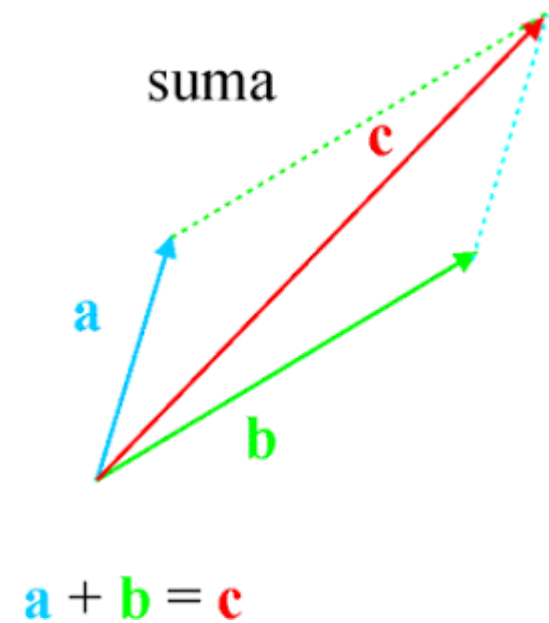
Correo: reyna.moreno@uaq.mx

Suma de vectores 2D y 3D

Sea \vec{u} y \vec{v} vectores en el mismo espacio , entonces:

$$\text{2D: } \vec{u} + \vec{v} = (u_1 + v_1, u_2 + v_2)$$

$$\text{3D: } \vec{u} + \vec{v} = (u_1 + v_1, u_2 + v_2, u_3 + v_3)$$



Suma de vectores n-Dimensionales

a	0	1	2	...	n-1
	a_0	a_1	a_2		a_{n-1}

b	0	1	2	...	n-1
	b_0	b_1	b_2		b_{n-1}

c=a+b	0	1	2	...	n-1
	$a_0 + b_0$	$a_1 + b_1$	$a_2 + b_2$		$a_{n-1} + b_{n-1}$

Proyecto CUDA



Create a new project

Choose a project template with code scaffolding to get started



CUDA 11.7 Runtime

A project that uses the CUDA 11.7 runtime

C++

CUDA

Windows

Linux

Cloud

Console

DataScience

Desktop

Machine Learning



CUDA 12.1 Runtime

A project that uses the CUDA 12.1 runtime

C++

CUDA

Windows

Linux

Cloud

Console

DataScience

Desktop

Machine Learning

Proyecto CUDA

Configure your new project

CUDA 12.1 Runtime

C++

CUDA

Windows

Linux

Cloud

Console

DataScience

Desktop

Machine Learning

Project name

Prog06_SumaVectores

Location

C:\TrabajoLaboratorio\CUDATopico2\Projects\Verano2024

Solution name ⓘ

Prog06_SumaVectores

☐ Place solution and project in the same directory

Project will be created in "C:\TrabajoLaboratorio\CUDATopico2\Projects\Verano2024\Prog06_SumaVectores\Prog06_SumaVectores\"

Back

Create

Operaciones de memoria (CPU)

- **malloc.**- Reserva un bloque de memoria de un tamaño definido de bytes, retornando un apuntador al inicio de dicho bloque. El contenido de dicho bloque no se inicializa por lo que es indeterminado. Ejemplo:

```
void* malloc (size_t size);  
buffer = (char*) malloc (sizeof(char)*100);
```

- **memset.**- asigna valores en secciones de memoria. Ejemplo:
Memset(variable, valor_a_asignar, tamaño_de_memoria)
Donde: tamaño_de_memoria se define como n * sizeof(tipo)

- **memcpy.**- Copia el contenido de un bloque de memoria referenciado por un apuntador a otro apuntador. Ejemplo:

```
void* memcpy( void* dest, const void* src, std::size_t count );  
memcpy (ptrDest, ptrOrigen, sizeof(int)*100);
```

- **free.**- Liberar la memoria reservada con el comando malloc. Ejemplo:
free(pointerName);
free(array2);

Operaciones de memoria (GPU)

- **cudaMalloc**.- asigna una sección de memoria en GPU de acuerdo con el espacio solicitado.

Ejemplo:

```
cudaMalloc((void**) &apuntador, tamaño_de_memoria)
```

Donde: tamaño_de_memoria se define como $n * \text{sizeof}(\text{tipo})$

- **cudaMemset**.- asigna valores en secciones de memoria.

Ejemplo:

```
Memset(apuntador, valor_a_asignar, tamaño_de_memoria)
```

Donde: tamaño_de_memoria se define como $n * \text{sizeof}(\text{tipo})$

- **cudaMemcpy**.- copia memoria hacia y desde el device.

Ejemplo:

```
cudaMemcpy(destino, origen, tamaño_de_memoria, indicador_flujo_de_inf)
```

Donde Indicador= cudaMemcpyHostToDevice, cudaMemcpyDeviceToHost,
cudaMemcpyDeviceToDevice

- **cudaFree**.- libera la memoria reservada por un apuntador.

Ejemplo:

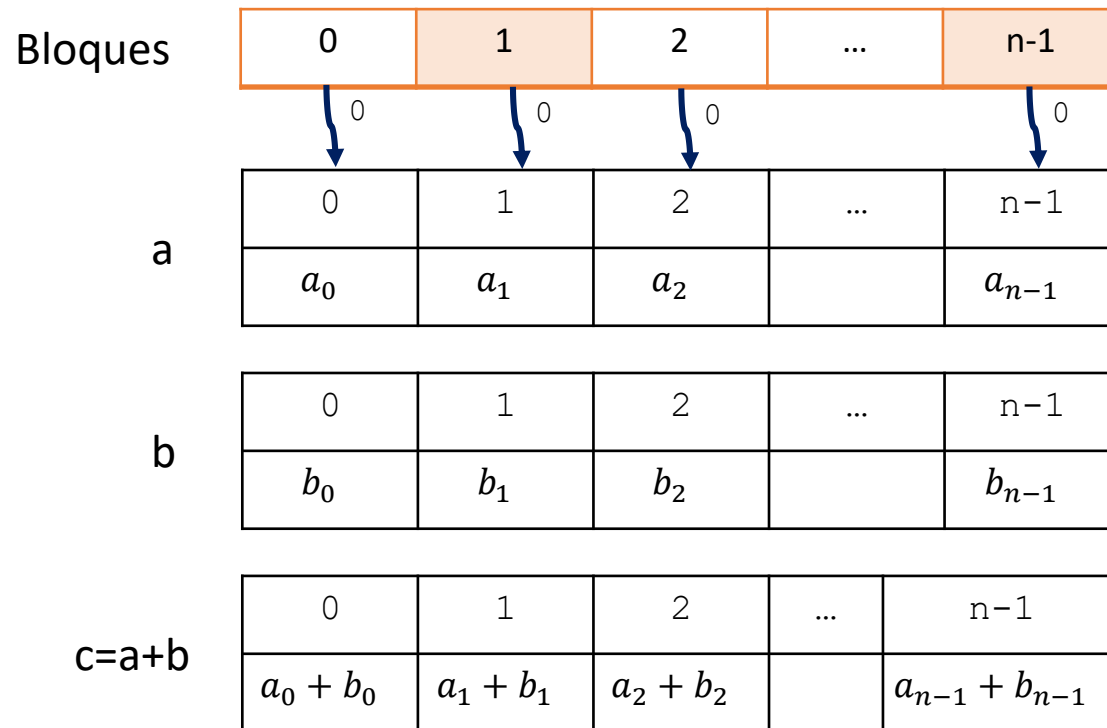
```
cudaFree (apuntador)
```

Memoria

CPU (Host)											
A01	length	50									
A05	maxHilos	1024									
A10	a	α	φ	η	λ	τ	κ	π	ε	...	ω
A15	b	χ	γ	φ	θ	ι	ϖ	υ	β	...	δ
A20	gpu_c	α	φ	η	λ	τ	κ	π	ε	...	ω
		+	+	+	+	+	+	+	+		+
		χ	γ	φ	θ	ι	ϖ	υ	β		δ
B01	cpu_c	α	φ	η	λ	τ	κ	π	ε	...	ω
		+	+	+	+	+	+	+	+		+
		χ	γ	φ	θ	ι	ϖ	υ	β		δ
B05											
B10	dev_a	D10									
B15	dev_b	D45									
B20	dev_c	D90									
C30											
E07											
E10											

GPU (Device)											
D01											
D05											
D10		α	φ	η	λ	τ	κ	π	ε	...	ω
D45		χ	γ	φ	θ	ι	ϖ	υ	β	...	δ
D90		α	φ	η	λ	τ	κ	π	ε	...	ω
		+	+	+	+	+	+	+	+		+
		χ	γ	φ	θ	ι	ϖ	υ	β		δ
F01											
F05											
F10											
F15											
F20											
G30											
H07											
I10											

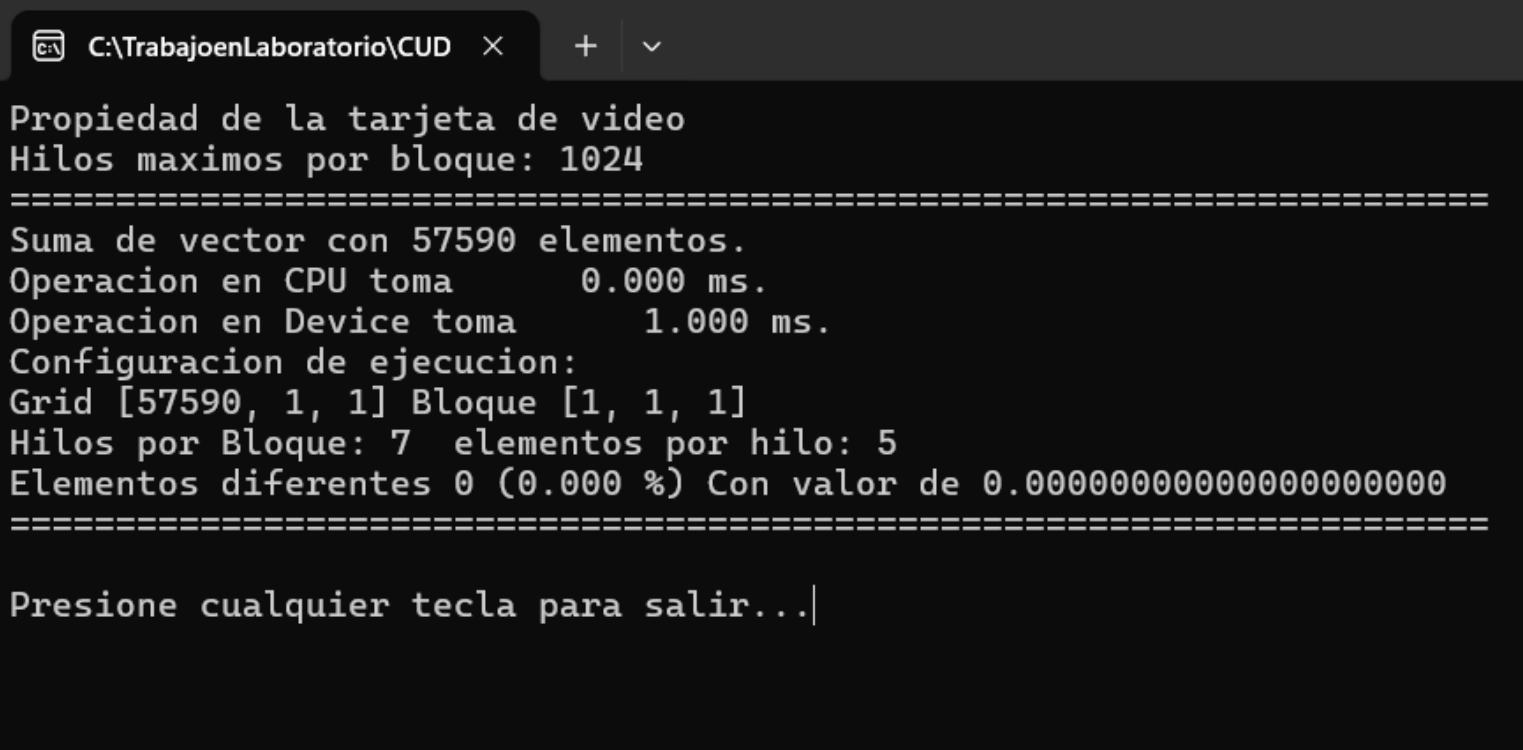
Caso 1. N Bloques con un hilo único



Caso 1. N Bloques con un hilo único

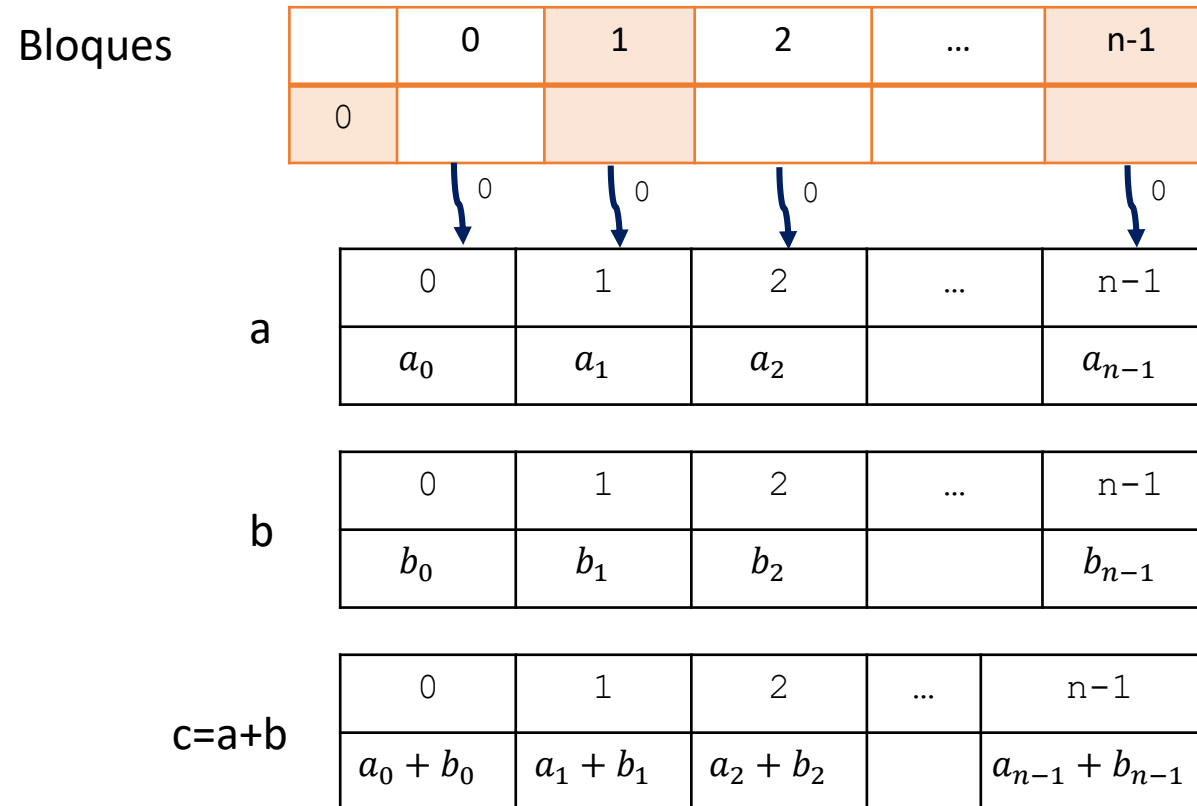
Ejemplo:

```
dim3 dimGrid(length);  
dim3 dimBlock(1);  
...  
int tid = blockIdx.x;  
c[tid] = a[tid] + b[tid];
```



```
C:\TrabajoLaboratorio\CUD x + v  
Propiedad de la tarjeta de video  
Hilos maximos por bloque: 1024  
=====  
Suma de vector con 57590 elementos.  
Operacion en CPU toma      0.000 ms.  
Operacion en Device toma   1.000 ms.  
Configuracion de ejecucion:  
Grid [57590, 1, 1] Bloque [1, 1, 1]  
Hilos por Bloque: 7  elementos por hilo: 5  
Elementos diferentes 0 (0.000 %) Con valor de 0.00000000000000000000  
=====  
Presione cualquier tecla para salir...|
```

Caso 2. 1xN Bloques con un hilo único



Caso 2. 1xN Bloques con un hilo único

Ejemplo:

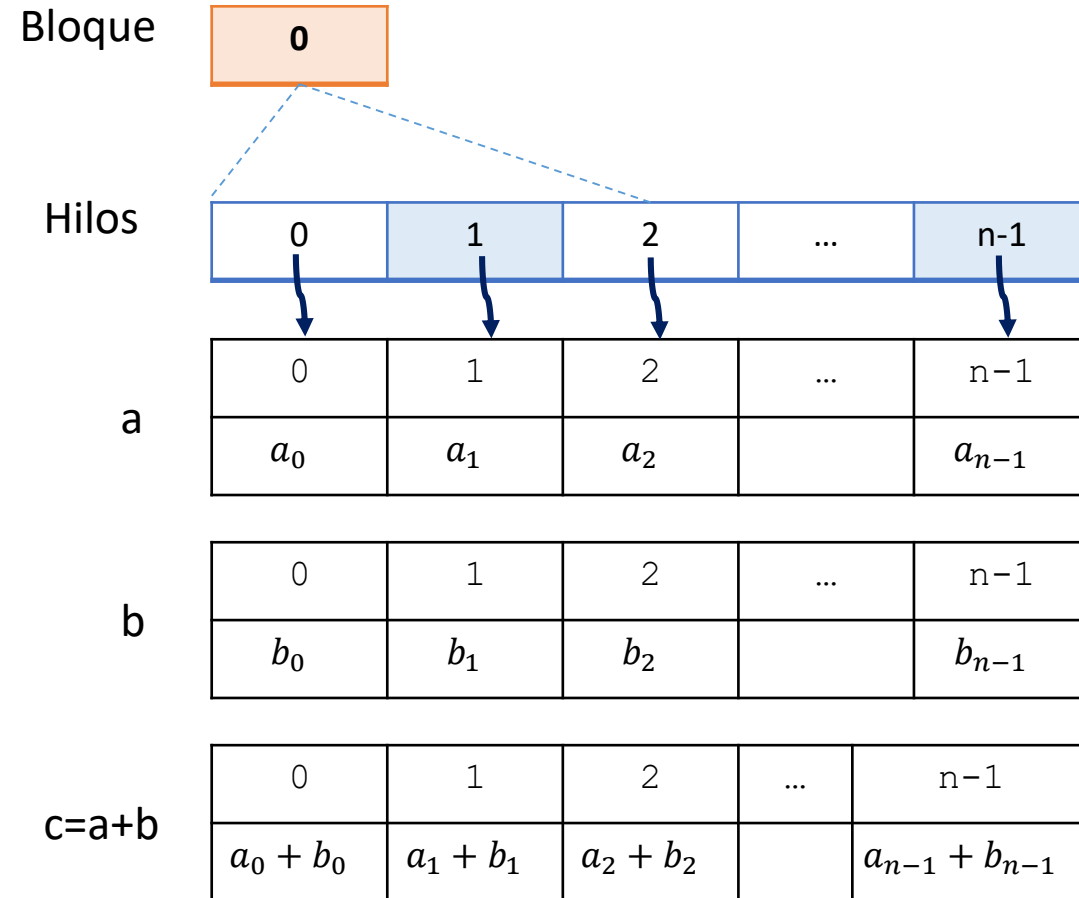
```
dim3 dimGrid(1, length);  
dim3 dimBlock(1);
```

...

```
int tid = blockIdx.y;  
c[tid] = a[tid] + b[tid];
```

```
C:\TrabajoLaboratorio\CUD × + v
Propiedad de la tarjeta de video
Hilos maximos por bloque: 1024
=====
Suma de vector con 57590 elementos.
Operacion en CPU toma      0.000 ms.
Operacion en Device toma   1.000 ms.
Configuracion de ejecucion:
Grid [1, 57590, 1] Bloque [1, 1, 1]
Elementos diferentes 0 (0.000 %) Con valor de 0.000000000000000000000000
=====
Presione cualquier tecla para salir...|
```

Caso 3. Un bloque con N hilos



Caso 3. Un bloque con N hilos

Ejemplo:

```
dim3 dimGrid (1);
dim3 dimBlock(length);
...
int tid = threadIdx.x;
c[tid] = a[tid] + b[tid];
```

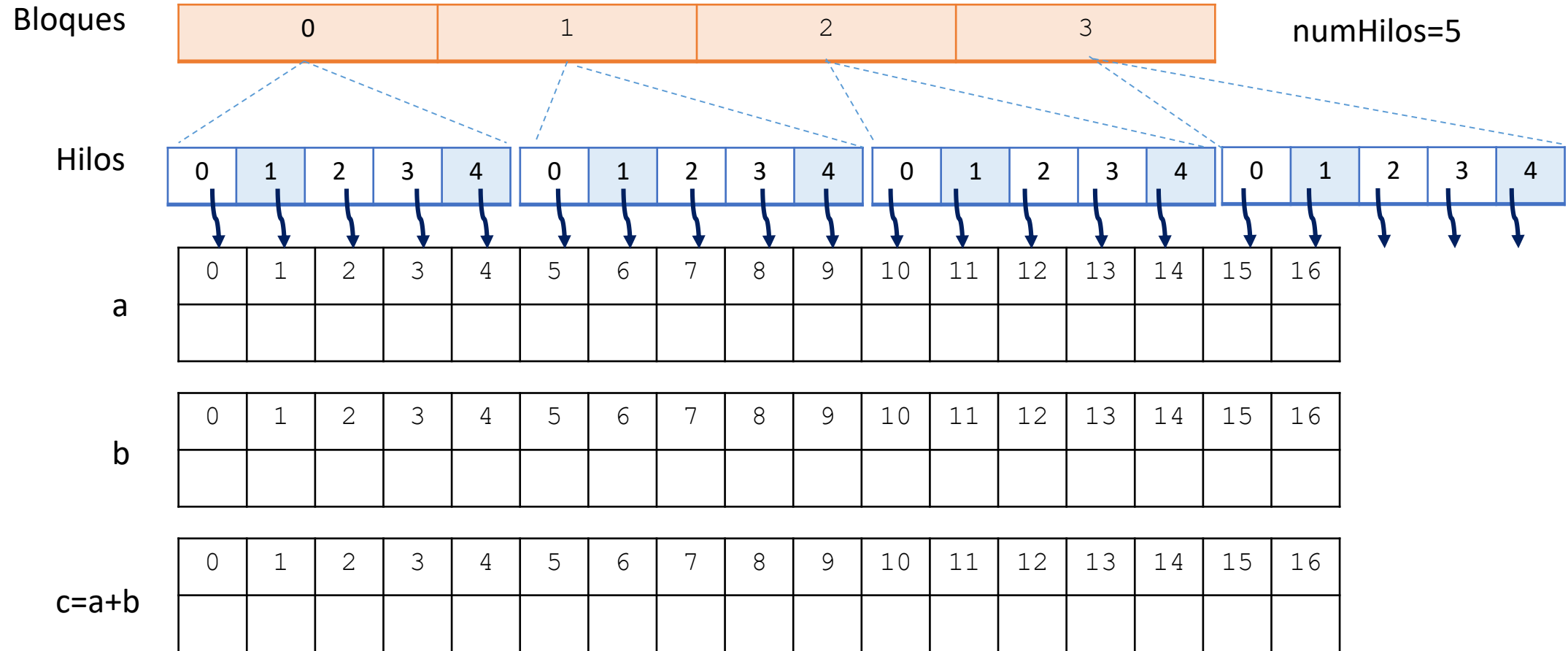
```
C:\TrabajoLaboratorio\CUD × + v
Propiedad de la tarjeta de video
Hilos maximos por bloque: 1024
=====
Suma de vector con 1024 elementos.
Operacion en CPU toma      0.000 ms.
Operacion en Device toma   1.000 ms.
Configuracion de ejecucion:
Grid [1, 1, 1] Bloque [1024, 1, 1]
Elementos diferentes 0 (0.000 %) Con valor de 0.00000000000000000000
=====
Presione cualquier tecla para salir...|
```

Caso 4. X bloques con numHilos c/u

Ejemplo:

Si length es 5000 y numHilos es 1024, entonces se generan $\lceil (5000/1024) \approx 5 \rceil$ 5 bloques, por lo tanto tenemos 5120 hilos. Si se requieren 5000 hilos, tenemos 120 hilos extras sin hacer trabajo (ociosos).

Caso 4. X bloques con numHilos c/u



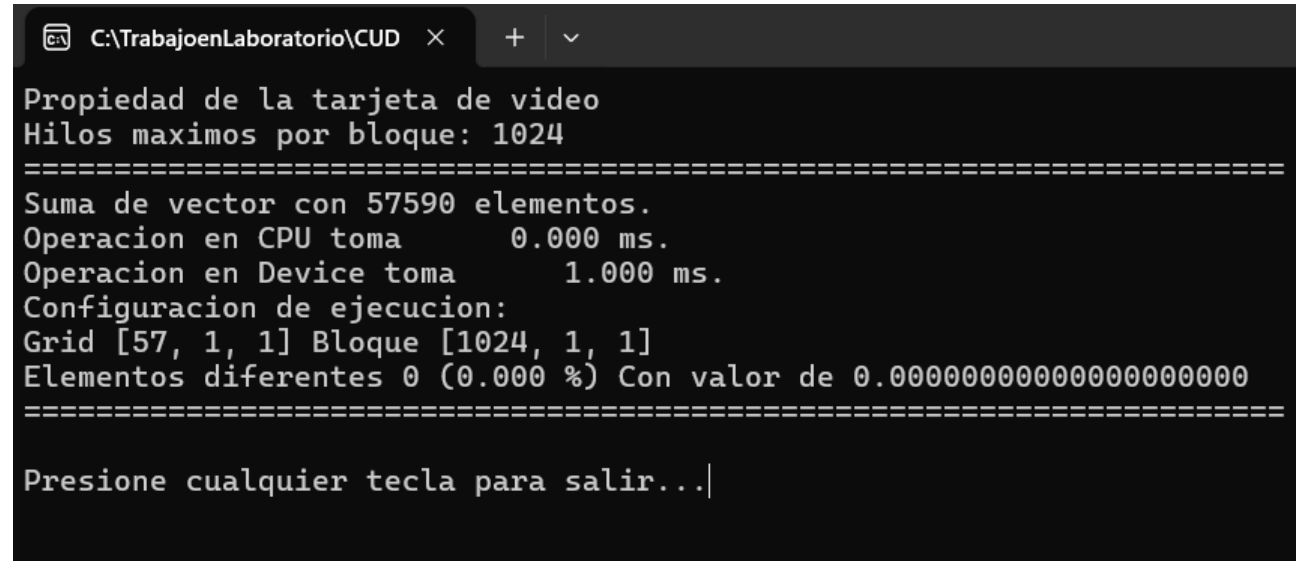
blockIdx.x	threadIdx.x	tid
0	0	0
0	1	1
0	2	2
0	3	3
0	4	4
1	0	5
1	1	6
1	2	7
1	3	8
1	4	9
2	0	10
2	1	11
2	2	12
2	3	13
2	4	14
3	0	15
3	1	16
3	2	17
3	3	18
3	4	19

$$\text{tid} = (\text{blockIdx.x} * \text{blockDim.x}) + \text{threadIdx.x}$$

Caso 4. X bloques con numHilos c/u

Ejemplo:

```
cudaDeviceProp devProp;
cudaGetDeviceProperties(&devProp, 0);
int maxHilos = devProp.maxThreadsPerBlock;
...
dim3 dimGrid(divEntera(length, maxHilos));
dim3 dimBlock(maxHilos);
...
int tid = (blockIdx.x*blockDim.x)+ threadIdx.x;
if (tid < length)
    c[tid] = a[tid] + b[tid];
```

A screenshot of a terminal window with a dark background and light-colored text. The window title bar shows the file path 'C:\TrabajoLaboratorio\CUD' and standard window controls. The output text is as follows:

```
Propiedad de la tarjeta de video
Hilos maximos por bloque: 1024
=====
Suma de vector con 57590 elementos.
Operacion en CPU toma      0.000 ms.
Operacion en Device toma   1.000 ms.
Configuracion de ejecucion:
Grid [57, 1, 1] Bloque [1024, 1, 1]
Elementos diferentes 0 (0.000 %) Con valor de 0.00000000000000000000
=====
Presione cualquier tecla para salir...|
```

Caso 5. X bloques con numHilos c/u tratando de generar los menos hilos ociosos

Ejemplo:

Si length es 5000 y maxHilos es 1024, entonces:

NumBloques= $5000/1024 = 4.8828 \approx 5$

NumHilos= $5000/5 = 1000$ (Hilos totales 5000)

Si length es 4833 y maxHilos es 1024, entonces:

NumBloques= $4833/1024 = 4.7197 \approx 5$

NumHilos= $4833/5 = 966.6 \approx 967$ (Hilos totales 4835)

Si length es 2512 y maxHilos es 1024, entonces:

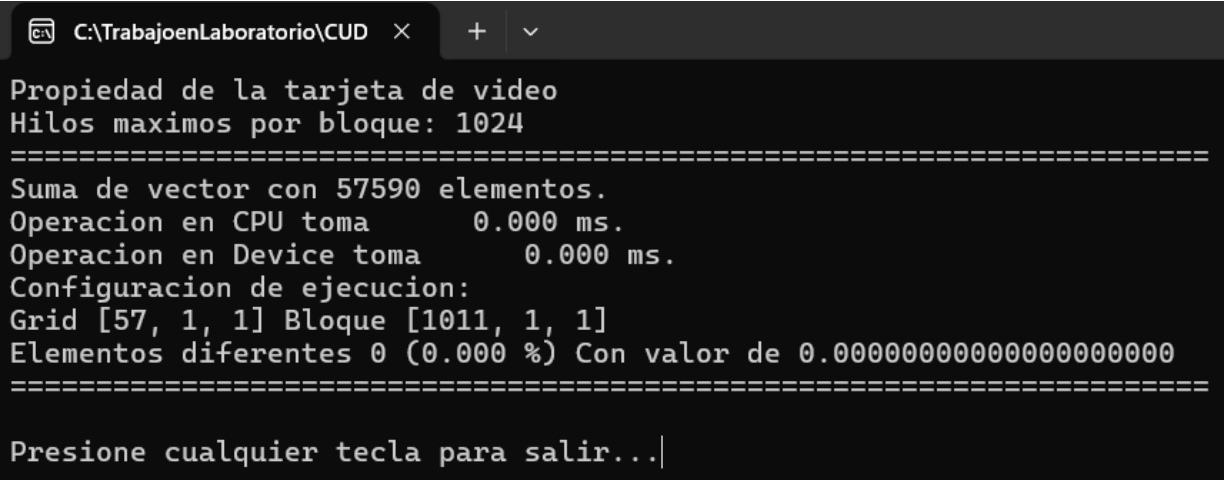
NumBloques= $2512/1024 = 2.453 \approx 3$

NumHilos= $2512/3 = 837.333 \approx 838$ (Hilos totales 2514)

Caso 5. X bloques con numHilos c/u tratando de generar los menos hilos ociosos

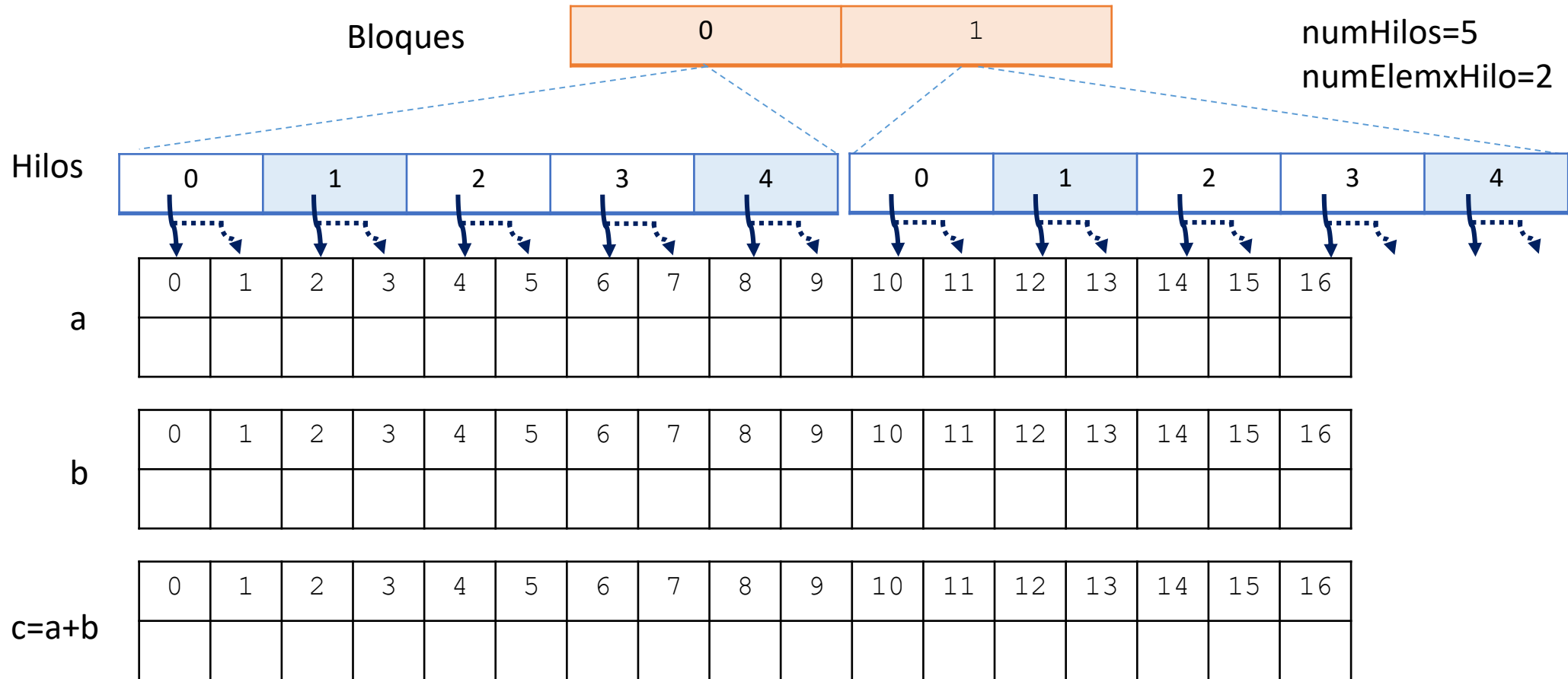
Ejemplo:

```
cudaDeviceProp devProp;
cudaGetDeviceProperties(&devProp, 0);
int maxHilos = devProp.maxThreadsPerBlock;
...
int numBloques = divEntera(length, maxHilos);
int numHilos = divEntera(length, numBloques);
dim3 dimGrid(numBloques);
dim3 dimBlock(numHilos);
...
int tid = (blockIdx.x*blockDim.x)+ threadIdx.x;
if (tid < length)
    c[tid] = a[tid] + b[tid];
```



```
C:\TrabajoenLaboratorio\CUD x + v
Propiedad de la tarjeta de video
Hilos maximos por bloque: 1024
=====
Suma de vector con 57590 elementos.
Operacion en CPU toma      0.000 ms.
Operacion en Device toma   0.000 ms.
Configuracion de ejecucion:
Grid [57, 1, 1] Bloque [1011, 1, 1]
Elementos diferentes 0 (0.000 %) Con valor de 0.00000000000000000000
=====
Presione cualquier tecla para salir...|
```

Caso 6. X bloques con numHilos que atienden cada uno a numElemxHilo c/u



blockIdx.x	threadIdx.x	tid	Elementos atendidos
0	0	0	0, 1
0	1	1	2, 3
0	2	2	4, 5
0	3	3	6, 7
0	4	4	8, 9
1	0	5	10, 11
1	1	6	12, 13
1	2	7	14, 15
1	3	8	16, 17
1	4	9	18, 19

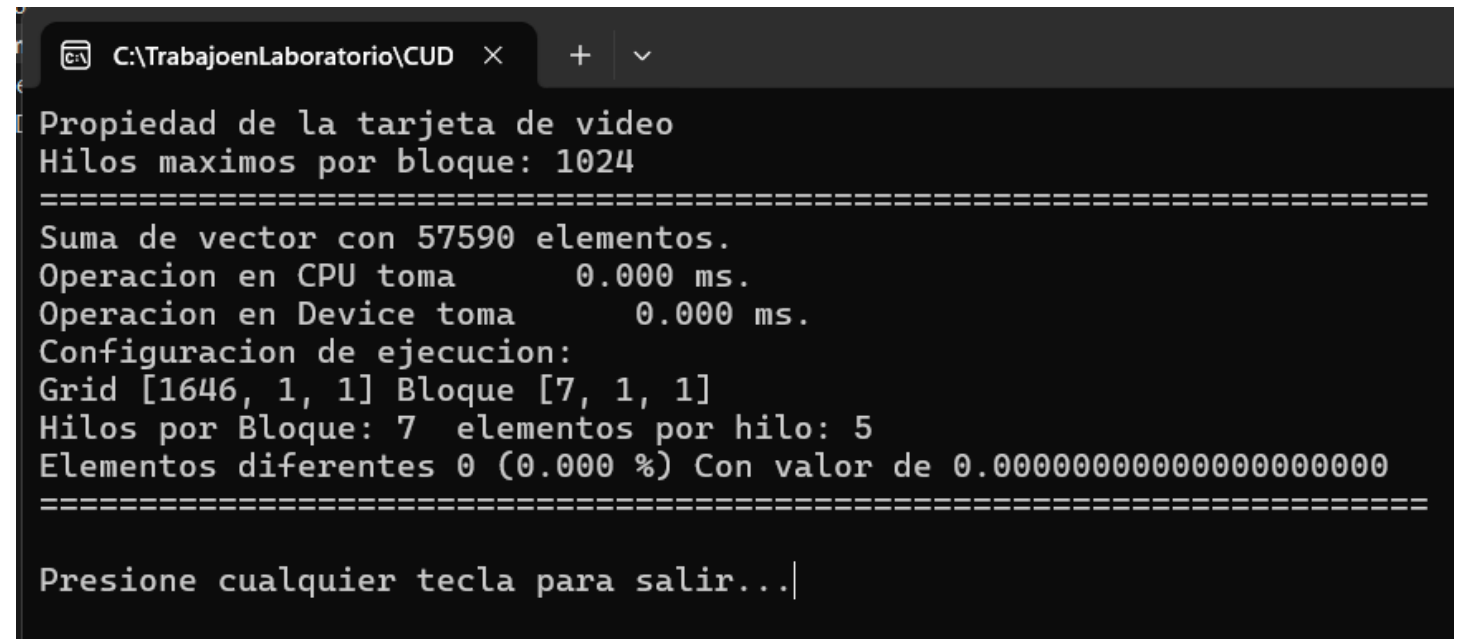
$$\text{tid} = (\text{blockIdx.x} * \text{blockDim.x}) + \text{threadIdx.x}$$

$$\text{PrimerElemento} = \text{tid} * \text{numElemxHilo}$$

Caso 6. X bloques con numHilos que atienden cada uno a numElemxHilo c/u

Ejemplo:

```
#define elemxHilo 5
#define hilosxBloque 7
...
int numBloques = divEntera(length,
    hilosxBloque * elemxHilo);
dim3 dimGrid(numBloques);
dim3 dimBlock(hilosxBloque);
...
int tid = ((blockIdx.x * blockDim.x) + threadIdx.x) * elemxHilo;
for (int i = 0; i < elemxHilo; i++) {
    if ((tid + i) < length)
        c[tid + i] = a[tid + i] + b[tid + i];
}
```

A screenshot of a terminal window with a dark background. The title bar shows the file path 'C:\TrabajoenLaboratorio\CUD'. The terminal output displays the following text: 'Propiedad de la tarjeta de video', 'Hilos maximos por bloque: 1024', a separator line of equals signs, 'Suma de vector con 57590 elementos.', 'Operacion en CPU toma 0.000 ms.', 'Operacion en Device toma 0.000 ms.', 'Configuracion de ejecucion:', 'Grid [1646, 1, 1] Bloque [7, 1, 1]', 'Hilos por Bloque: 7 elementos por hilo: 5', 'Elementos diferentes 0 (0.000 %) Con valor de 0.00000000000000000000', another separator line of equals signs, and 'Presione cualquier tecla para salir...|'.

```
C:\TrabajoenLaboratorio\CUD x + v
Propiedad de la tarjeta de video
Hilos maximos por bloque: 1024
=====
Suma de vector con 57590 elementos.
Operacion en CPU toma      0.000 ms.
Operacion en Device toma   0.000 ms.
Configuracion de ejecucion:
Grid [1646, 1, 1] Bloque [7, 1, 1]
Hilos por Bloque: 7 elementos por hilo: 5
Elementos diferentes 0 (0.000 %) Con valor de 0.00000000000000000000
=====
Presione cualquier tecla para salir...|
```

Bibliografía

- Documentación **CUDA C++ Programming Guide** NVIDIA. 2024
<https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html>
- Sitio **CUDA Toolkit Documentation** NVIDIA, 2024.
<https://docs.nvidia.com/cuda/index.html>
- Storti, Duane; Yurtoglu, Mete. **CUDA for Engineers: An Introduction to High-Performance Parallel Computing**. Addison Wesley. 2015.
- Cheng, John; Grossman, Max; McKercher. **Professional CUDA C Programming**. Edit. Wrox. 2014.
- Sanders, Jason; Kandrot, Edward. **CUDA by Example: An Introduction to General-Purpose GPU Programming**. Addison Wesley. 2011.
- Kirk, David; Hwu, Wen-mei. **Programming Massively Parallel Processors: A Hands-on Approach**. Elsevier. 2010.

Gracias por su atención



**U.A.Q. Fac. de Informática
Campus Juriquilla**

Dra. Sandra Luz Canchola Magdaleno
sandra.canchola@uaq.mx
Cel. 442-1369270

Dra. Reyna Moreno Beltrán
reyna.moreno@uaq.mx