



## Introducción

Este documento constituye el componente práctico con manos de la Aprenda CUDA en un tutorial de la tarde disponible aquí: <http://sc3.uis.edu.co/>

Se supone que usted tiene acceso a una computadora con una GPU NVIDIA CUDA y sistema operativo Linux. Para sistemas Windows, sólo tendrán que adaptarse configuraciones o el uso de clientes terminales para trabajar sobre una maquina remota que si emplee Linux. etc ( por favor, consulte la documentación de NVIDIA).

En el primer ejercicio buscamos ayudarle a empezar con su primer código **CUDA**. El segundo ejercicio se utiliza, como punto de partida una aplicación **CUDA** que “funciona mal” y que va a utilizar varias técnicas para optimizar el código y mejorar el rendimiento de la **GPU**.

Para obtener los archivos de la plantilla:

```
wget https://dl.dropboxusercontent.com/u/49956056/SC3\_2014-IntroCuda.zip
```

```
wget
```

(o descargar mediante el navegador web de esta dirección)

```
cd SC3_2014-IntroCuda
```

Para cada uno de los ejercicios existe una subcarpeta **src**, que contiene las plantillas de ejercicio.

### Glosario de Apoyo

- host (CPU).
- Device o dispositivo (GPU)
- Array (Vector)
- Kernel (nucleo o función CUDA)
- NUM\_BLOCKS
- THREADS\_PER\_BLOCK
- SMs (Streaming Multiprocessors)
- CUDA C (lenguaje nativo para programación paralela sobre GPUs NVIDIA)

# Primeros pasos con CUDA

## Introducción

El ejercicio introductorio está disponible en la carpeta **intro/src**. Este contiene una plantilla **CUDA**, archivo que Ud. va a editar.

El archivo de origen plantilla está claramente marcado con las secciones a editar, por ejemplo,

```
/* Parte 1A : asignar memoria del dispositivo */
```

Por favor vea a continuación las instrucciones. Si considera necesario, pida instrucciones o asesoría o puede consultar la Guía de programación **CUDA C** y los documentos de referencia manuales disponibles

<http://developer.nvidia.com/nvidia-gpu-computing-documentation>

## 1. Copiado Entre host y el dispositivo

El ejercicio introductorio es un código **CUDA** sencillo, mediante el cual buscamos invertir (multiplicar por -1), el valor de cada posición de un vector de enteros. Por medio de este introduciremos los conceptos importantes **CUDA**, como la administración de memoria entre los dispositivos y la invocación del **kernel**.

La versión final debe copiar el vector de enteros desde el host al dispositivo, **multiplicar cada elemento por -1** en el **dispositivo (GPU)**, y luego copiar el vector de vuelta al **host (CPU)**.

Comience desde la plantilla **introCuda.cu**.

- **Parte 1A:** Asignar memoria para la matriz en el dispositivo: utilice el puntero **d\_a** existente y la variable **SZ** (la cual ya tiene asignado el tamaño de la matriz en bytes) .
- **Parte 1B:** Copie el vector **h\_a** en el host hacia **d\_a** en el dispositivo.
- **Parte 1C:** Copia el vector **d\_a** del dispositivo de nuevo a **h\_out** en el host.
- **Parte 1D:** libere la memoria del vector **d\_a**. (usando `cudaFree()`)

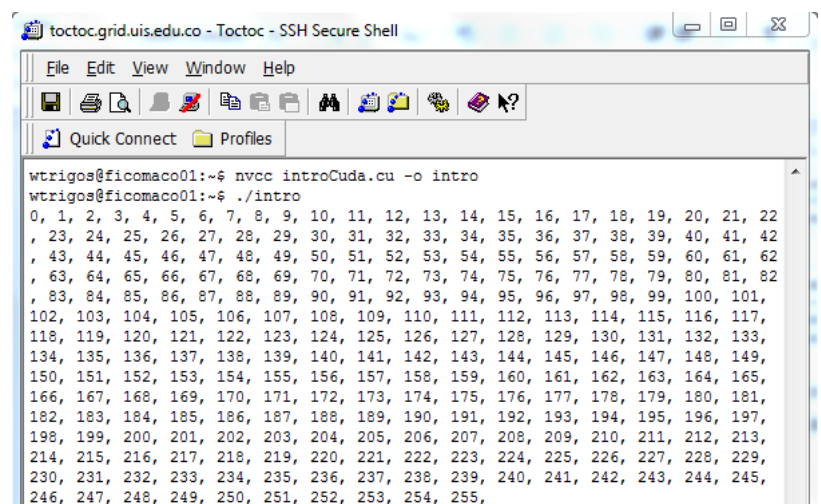
Para compilar, ejecute el comando:

```
nvcc -o intro introCuda.cu
```

Para ejecutar:

```
./intro
```

Hasta ahora, el código simplemente hace copias del vector **h\_a** en el host hacia **d\_a** en el dispositivo, y luego copia **d\_a** hacia el vector **h\_out** en el host, por lo que el resultado de la impresión en pantalla debe ser el contenido inicial de **h\_a** – (los números del 0 al 255).



```
ttoctoc.grid.uis.edu.co - Toctoc - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
wtrigos@ficomaco01:~$ nvcc introCuda.cu -o intro
wtrigos@ficomaco01:~$ ./intro
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22
, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42
, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62
, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82
, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101,
102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117,
118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133,
134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149,
150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165,
166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197,
198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213,
214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229,
230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245,
246, 247, 248, 249, 250, 251, 252, 253, 254, 255,
```

Elaborado: William J. Trigos G.

Revisado: Carlos J. Barrios, SC3

## 2. Lanzamiento de Núcleos

Ahora vamos a editar la plantilla **introCuda.cu**. Para que realmente ejecute un **kernel** en el **dispositivo (GPU)**.

- **Part2A:** Configurar y poner en marcha el núcleo usando una cuadrícula de 1D y un bloque de un solo hilo (**NUM\_BLOCKS** y **THREADS\_PER\_BLOCK** ya están definidos para este caso) .
- **Part2B:** Implementar la función real del **kernel** para negar un elemento de matriz de la siguiente manera :

```
int idx = threadIdx.x ;
D_A [ idx ] = -1 * D_A [ idx ] ;
```

Compile y ejecute el código como lo hizo en la primera parte. Esta vez, la salida debe contener el resultado de la negación de cada elemento de la matriz de entrada. Dado que la matriz se inicializa a los números (0 al 255), deberá ver los números (0 al -255) en esta ocasión.

## 3. Mejorando el kernel (Multi-Thread → Multi-Block)

Este **kernel** funciona, pero tan sólo utiliza una secuencia de **threads**, tan solo emplea uno de los múltiples **SMs** disponible en la **GPU**. Se necesitan secuencias de roscado múltiple para aprovechar al máximo los recursos disponibles.

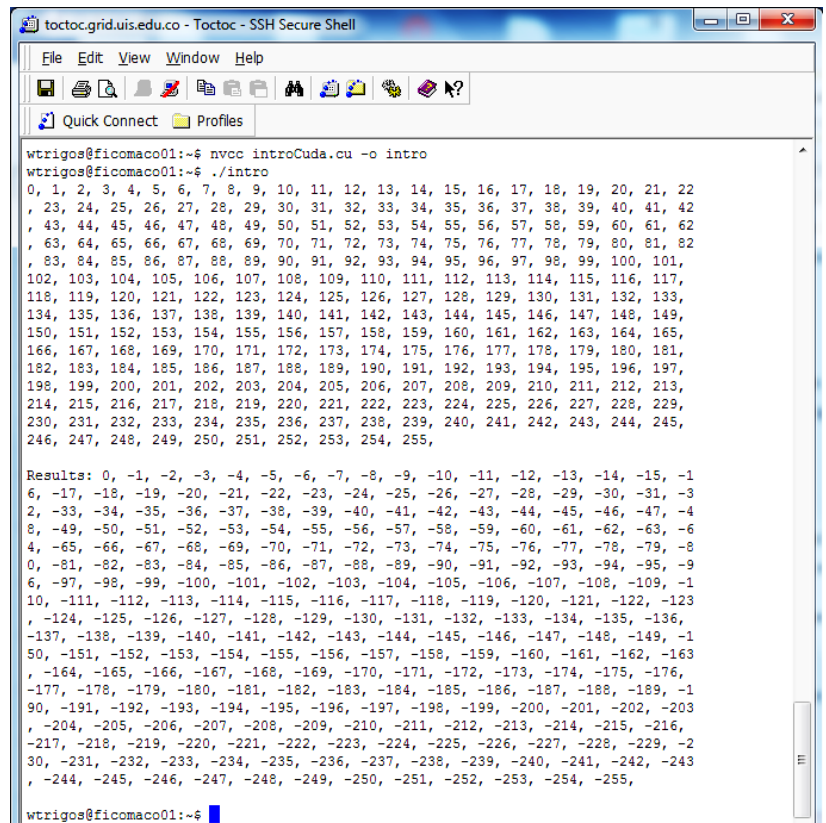
- **Parte 2C:** Implementar un nuevo **kernel**, esta vez permitiendo la ejecución de una secuencia de múltiples **threads**.

Esto es muy similar a la implementación del núcleo anterior, excepto por el índice para recorrer el vector, este se calcula de forma diferente:

```
int idx = threadIdx.x + ( blockIdx.x * blockDim.x ) ;
```

**NOTA:** No olvide cambiar también la invocación **kernel** para invocar **negate\_multiblock** este momento.

Con esta versión puede cambiar **NUM\_BLOCKS** y **THREADS\_PER\_BLOCK** para tener diferentes tiempos de ejecución, ya que se multiplican las unidades de procesamiento para efectuar trabajo sobre el vector.



```
ttoctoc.grid.uis.edu.co - Ttoctoc - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
wtrigos@ficomaco01:~$ nvcc introCuda.cu -o intro
wtrigos@ficomaco01:~$ ./intro
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22
, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42
, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62
, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82
, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101,
102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117,
118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133,
134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149,
150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165,
166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197,
198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213,
214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229,
230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245,
246, 247, 248, 249, 250, 251, 252, 253, 254, 255,
Results: 0, -1, -2, -3, -4, -5, -6, -7, -8, -9, -10, -11, -12, -13, -14, -15, -16,
-17, -18, -19, -20, -21, -22, -23, -24, -25, -26, -27, -28, -29, -30, -31, -32,
-33, -34, -35, -36, -37, -38, -39, -40, -41, -42, -43, -44, -45, -46, -47, -48,
-49, -50, -51, -52, -53, -54, -55, -56, -57, -58, -59, -60, -61, -62, -63, -64,
-65, -66, -67, -68, -69, -70, -71, -72, -73, -74, -75, -76, -77, -78, -79, -80,
-81, -82, -83, -84, -85, -86, -87, -88, -89, -90, -91, -92, -93, -94, -95, -96,
-97, -98, -99, -100, -101, -102, -103, -104, -105, -106, -107, -108, -109, -110,
-111, -112, -113, -114, -115, -116, -117, -118, -119, -120, -121, -122, -123,
-124, -125, -126, -127, -128, -129, -130, -131, -132, -133, -134, -135, -136,
-137, -138, -139, -140, -141, -142, -143, -144, -145, -146, -147, -148, -149, -150,
-151, -152, -153, -154, -155, -156, -157, -158, -159, -160, -161, -162, -163,
-164, -165, -166, -167, -168, -169, -170, -171, -172, -173, -174, -175, -176,
-177, -178, -179, -180, -181, -182, -183, -184, -185, -186, -187, -188, -189, -190,
-191, -192, -193, -194, -195, -196, -197, -198, -199, -200, -201, -202, -203,
-204, -205, -206, -207, -208, -209, -210, -211, -212, -213, -214, -215, -216,
-217, -218, -219, -220, -221, -222, -223, -224, -225, -226, -227, -228, -229, -230,
-231, -232, -233, -234, -235, -236, -237, -238, -239, -240, -241, -242, -243,
-244, -245, -246, -247, -248, -249, -250, -251, -252, -253, -254, -255,
wtrigos@ficomaco01:~$
```