Practica 4: Programación CUDA
Arquitectura de Computadoras

Profesor: José Alberto Domingo Incera Dieguez
Alumno: Héctor Hugo Huipet Hernández

Información de mi computadora personal:

```
hugo-pc@Lenovo-Y50-70: ~/samples/NVIDIA_CUDA-7.0_Samples/1_Utilities/deviceQuery
cp deviceQuery ../../bin/x86_64/linux/release
hugo-pc@Lenovo-Y50-70:~/samples/NVIDIA_CUDA-7.0_Samples/1_Utilities/deviceQuery$ ls
deviceQuery  deviceQuery.cpp  deviceQuery.o  Makefile  NsightEclipse.xml  readme.txt
hugo-pc@Lenovo-Y50-70:~/samples/NVIDIA_CUDA-7.0_Samples/1_Utilities/deviceQuery$ ./deviceQuery
./deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "GeForce GTX 860M"
  CUDA Driver Version / Runtime Version          7.5 / 7.0
  CUDA Capability Major/Minor version number:    5.0
  Total amount of global memory:                 2048 MBytes (2147352576 bytes)
  ( 5) Multiprocessors, (128) CUDA Cores/MP:     640 CUDA Cores
  GPU Max Clock rate:                            1020 MHz (1.02 GHz)
  Memory Clock rate:                             2505 Mhz
  Memory Bus Width:                              128-bit
  L2 Cache Size:                                 2097152 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(65536), 2D=(65536, 65536), 3D=(4096, 4096, 4096)
  Maximum Layered 1D Texture Size, (num) layers  1D=(16384), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(16384, 16384), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 1 copy engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  Device supports Unified Addressing (UVA):      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 1 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 7.5, CUDA Runtime Version = 7.0, NumDevs = 1,
Device0 = GeForce GTX 860M
Result = PASS
hugo-pc@Lenovo-Y50-70:~/samples/NVIDIA_CUDA-7.0_Samples/1_Utilities/deviceQuery$ nvcc -V
```

Ejercicio 1

En el código anexo **intro.cu** se pueden observar las modificaciones al código de introducción, que permiten ejecutar la suma de los vectores ya sea en un solo bloque o de manera que pueda haber múltiples bloques. La suma de los arreglos produce que cada posición en el arreglo resultado c se almacene el número 256.

Ejercicio 2

El programa inicializa un vector de 1024 elementos y en cada posición almacena su valor de índice 0... 1023.
Posteriormente realiza la transferencia de los datos a la variable que ocupará el device para realizar el cálculo en el GPU, llama al kernel **square_array** y transfiere los valores del arreglo resultado devuelta al host. También registra cuánto tarda el programa en ejecutarse en milisegundos.

El código del kernel está diseñado para que cada hilo ejecute un ciclo sobre un conjunto de elementos del arreglo usando un patrón de memoria que puede ser modificado junto con la homogeneidad de los hilos. Cada hilo en un bloque recibe una dirección de inicio de los elementos que procesará, determinado por el índice del hilo y el valor del stride y del offset

Prueba	**STRIDE**	**OFFSET**	**GROUP_SIZE**	**T. Ejecución**
1	32	0	512	0.196960ms
2	16	0	512	0.193248ms
3	8	0	512	0.210176ms
4	32	1	512	0.187264ms
5	32	0	16	0.187616ms
6	32	0	8	0.190272ms
7	8	1	8	0.187680ms

Ejercicio 3

Producto Matricial

¿Cuántas veces se lee cada una de las entradas de A y B?
Cada entrada se lee N veces, donde N es el tamaño de la matriz cuadrada

Ejercicio 4

Área del conjunto de Mandelbrot

¿Detecta alguna diferencia en el desempeño obtenido en este ejercicio y el de producto matricial? **Si**

```
TIEMPO DE EJECUCIÓN: 0.076704 mSeg    matrix

TIEMPO DE EJECUCIÓN: 384.870117 mSeg   mandel
```

De ser positiva su respuesta, ¿A qué factores atribuye esta diferencia? **Quizás se debe a que para la matriz yo utilice un ejemplo de 36 elementos mucho más pequeño que el de mandel.**