Práctico 2 del curso GPGPU:
Primeros pasos con CUDA

Ejercicio 1

Parte a

El texto que se encuentra en el archivo secreto.txt ha sido encriptado sustituyendo cada caracter mediante una función definida como

$$E(x) = (Ax + B)\ mod\ M \tag{1}$$

donde A y B son las claves del cifrado, *mod* es el operador de módulo, y A y M son co-primos. Para este ejercicio el valor de A =15, B =27 y M =256 (la cantidad de caracteres en la tabla ASCII extendida). La función de desencriptado se define como:

$$D(x) = A^{-1}(x - B)\ mod\ M \tag{2}$$

Donde A^{-1} es el inverso multiplicativo de A módulo M. En este caso $A^{-1} = -17$.

Como cada caracter puede ser encriptado y desencriptado de forma independiente podemos utilizar la GPU para desencriptar el texto en paralelo. Para esto debemos lanzar un kernel que realice el desencriptado, asignando un thread por caracter. Usando como base `ej1.cu`:

1. Implementar el kernel `decrypt_kernel` utilizando un solo bloque de threads (en la dimensión x). A, B y M están definidas en el código como macros.
2. Reservar memoria en la GPU para el texto a desencriptar.
3. Copiar el texto a la memoria de la GPU.
4. Configurar la grilla de threads con 1 bloque de n threads (tamaño de bloque a elección) e invocar el kernel
5. Transferir el texto a la memoria de la CPU para desplegarlo en pantalla.

Parte b

Modificar el código del kernel y la definición de la grilla de threads para que utilice varios bloques, procesando textos de largo arbitrario.

Parte c

Modificar el código del kernel y la definición de la grilla de threads para que utilice una cantidad fija de bloques (por ejemplo 128), para procesar textos de largo arbitrario.

Ejercicio 2

Implemente una función que tome como entrada el texto desencriptado y genere un vector de tamaño 256 con la cantidad de ocurrencias de cada caracter en el texto. El conteo de la cantidad de ocurrencias debe realizarse en la GPU, utilizando un arreglo en memoria global, y el resultado debe quedar en la memoria de la CPU. Recuerde que si dos hilos acceden concurrentemente a la misma posición de un vector y al menos uno de ellos escribe, debe utilizarse algún mecanismo para evitar una condición de carrera.

Notas:

- **El operador mod no es el mismo que (%) para números negativos, por lo que se provee una función módulo en el código.**
- **Para resolver los ejercicios deberá utilizar las variables especiales:**
  - **threadIdx.x, blockIdx.x, blockDim.x y gridDim.x**
- **Para compilar se utiliza el comando** `nvcc`**:**
  - **Ejemplo:** `nvcc -lineinfo ej1.cu -o ej1_sol`
  - **Esto debe realizarse en un nodo de cómputo (no en el nodo login)**
  - **Recuerden setear las variables de entorno**
    - `export PATH=$PATH:/usr/local/cuda/bin`
    - `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib64`