1. **Explique el funcionamiento del compilador de Cuda (nvcc,ptx,ocg,cl y gcc)**

El compilador de cuda (NVCC) utiliza un preprocesador (CUDAFE) que divide la aplicación en la CPU y la GPU, creando un archivo preprocesado C para la entrada de GPU: foo.s (lenguaje ensablador) para GPU y foo.cpp para CPU.

La entrada de la GPU es procesada por nuestro compliador Open64, llamado nvopencc, que emite un lenguaje ensamblador llamado PTX.

La PTX se pasa entonces a OCG, que asigna los registros y la lista de instrucciones de acuerdo con el chip en particular que se está utilizando y optimizando. Tras esto se obtiene código objeto para la GPU, que más tarde da lugar al ejecutable

1. **Suponiendo que tenemos un Kernel con 128 bloques y que cada uno tiene medio Warp de hilos, ¿Cuál es la cantidad total de hilos? ¿Qué posición ocupara el hilo 12 del bloque 16?**

128 bloques \* 16 hilos = 2048 hilos

El hilo 12 del bloque 16 ocupa la posición = (16 \* 16 + 12) = 268

1. **Describe el funcionamiento del cauce gráfico (procesador de vértices, rasterizado, procesador de fragmentos, etc.)**

Inicialmente, a la GPU le llega la información de la CPU en forma de vértices. El primer tratamiento que reciben estos vértices se realiza en el *vertex shader*. Aquí se realizan transformaciones como la rotación o el movimiento de las figuras. Tras esto, se define la parte de estos vértices que se va a ver (*clipping*), y los vértices se transforman en píxeles mediante el proceso de rasterización. Estas etapas no poseen una carga relevante para la GPU.

Donde sí se encuentra el principal cuello de botella del chip gráfico es en el siguiente paso: el *pixel shader*. Aquí se realizan las transformaciones referentes a los píxeles, tales como la aplicación de texturas. Cuando se ha realizado todo esto, y antes de almacenar los píxeles en la [caché](https://es.wikipedia.org/wiki/Cach%C3%A9_(inform%C3%A1tica)), se aplican algunos efectos como el *antialiasing*, *blending* y el efecto niebla.

1. **En los modelos que tienen un procesador G80, ¿Cuántas matrices de 256 hilos pueden correrse en un SM si cada una consume 4kb de memoria compartida y 1024 registros? Justifique la respuesta a partir de las limitaciones de la G80**

La G80 tiene varios puntos críticos que podrían ocasionar problemas en este caso en concreto

Lo primero, son los registros. La limitación en este campo de la G80 se encuentra en los 8192, por lo que, si SOLO hablásemos de ellos, podría correr 8192/1024 = 8 matrices.

Lo segundo, es la memoria compartida. Con un límite de 16k, en el ejemplo en el que nos movemos, podría correr hasta 4 matrices.

Pero en este caso, lo que nos va a limitar son los hilos. Una G80 puede correr como máximo 768 hilos en un SM, por lo que 768/256 = 3 matrices son las que podrían ejecutarse.

1. **¿En qué consiste la técnica de renderizado de una imagen? ¿Y el culling y clipping en el cauce gráfico?**

Habitualmente se utiliza el termino renderizado para definir el proceso por el cual se pretende imitar un entorno tridimensional, formado por estructuras poligonales, luces, texturas y materiales, simulando ambientes y estructuras físicas verosímiles. También se suele utilizar para procesos 2D que requieren cálculos complejos como la edición de vídeo, la animación o el desarrollo de efectos visuales.

Culling es el proceso usado para determinar que superficies y partes de superficies son visibles o no desde algún punto de observación.

Clipping es un método de habilitar o deshabilitar selectivamente operaciones de renderización dentro de una región de interés definida. Un algoritmo de renderización solo dibuja píxeles en la intersección entre la región de recorte y el modelo de la escena. Las líneas y superficies ubicadas fuera del área de visualización son eliminadas.

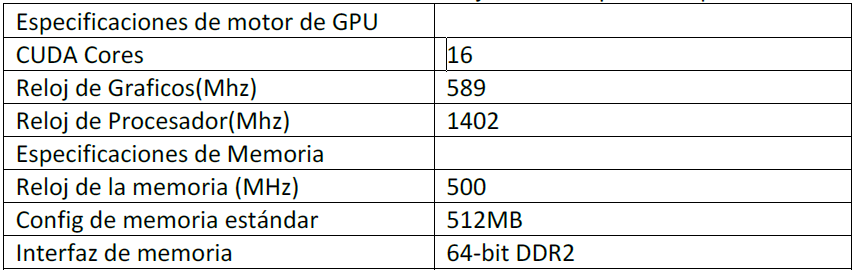
1. **Indique los niveles de paralelismo propuestos en la Taxonomía de Flynn. Explique y ponga un ejemplo de cada uno**

**Single instruction single data -** Computador [secuencial](https://es.wikipedia.org/wiki/Sistema_secuencial) que no explota el paralelismo en las instrucciones ni en flujos de datos. Es la Arquitectura Von-Neumann. Un único procesador ejecuta un sólo flujo de instrucciones para operar datos en una única memoria

**Single instruction multiple data -** Todas las unidades ejecutan la misma instrucción sincronizadamente, pero con datos distintos. Es un computador que explota varios flujos de datos dentro de un único flujo de instrucciones para realizar operaciones que pueden ser paralelizadas de manera natural. Puede ser de Arquitectura vectorial o Arquitectura matricial.

**Multiple instruction single data -** Poco común debido al hecho de que la efectividad de los múltiples flujos de instrucciones suele precisar de múltiples flujos de datos. Sin embargo, este tipo se usa en situaciones de paralelismo redundante, como por ejemplo en navegación aérea, donde se necesitan varios sistemas de respaldo en caso de que uno falle

**Multiple instruction multiple data -** varios procesadores autónomos que ejecutan simultáneamente instrucciones diferentes sobre datos diferentes. Los sistemas distribuidos suelen clasificarse como arquitecturas MIMD

1. **¿Cuál es el ancho de banda máximo para la siguiente tarjeta:** 

(2 \* 0.5 \* 64) / 8 = 8 GB/s

1. **Supongamos que un núcleo tiene bloques de 256 hilos, 16 instrucciones independientes de acceso a la memoria global por hilo y que cada hilo usa 11 registros y los accesos a la memoria global consumen 200 ciclos ¿Cuántos bloques de un SM se puede ejecutar en estas condiciones?**

Dado que las instrucciones y los ciclos no nos interesan para el cálculo, tenemos que comprobar el número de hilos y de registros totales para ver cual sobrepasa el límite antes (768 y 8192 respectivamente)

Dado que los bloques tienen 256 hilos y cada hilo usa 11 registros, hay un total de 2816 registros por bloque. El máximo de hilos que pueden ejecutarse en un SM es de 768, lo que hace que SOLO POR NUMERO DE HILOS, puedan ejecutarse 3, pero como al ejecutar 3 excederíamos el número de registros disponibles, en este caso el número máximo de bloques ejecutables es de dos.

1. **Suponiendo que tenemos un bloque de hilos de 128 x 256 y que estamos trabajando sobre una G80**
   1. **Realiza la configuración del Kernel en función del ancho de banda del bloque de hilo para un desarrollo en la memoria global**
   2. **Realiza la configuración del Kernel en función del ancho de bloque de hilos. Multiples bloques con memoria compartida y teniendo en cuenta la tesela más óptima.**