Micro Investigación

1. **¿Qué es OpenCL?**
OpenCL(Open Computing Language) es una herramienta que permite crear aplicaciones con paralelismo a nivel de datos o tareas. Estas aplicaciones pueden ejecutarse ya sea en el CPU como en el GPU. Esta herramienta mejora la velocidad y la respuesta en un gran rango de aplicaciones ya sea en el mercado de videojuegos, entretenimiento, aplicaciones científicas, entre otros.[1]

2. **¿Qué es un kernel en OpenCL y cómo se define?**
El kernel en OpenCL es básicamente una función la cual es ejecutada por un dispositivo OpenCL, estos dispositivos de OpenCL pueden ser unidades centrales de procesamiento(CPU) o unidades de procesamiento gráfico(GPU).[2]
La forma en la que se define un kernel en OpenCL es de la siguiente manera:
__kernel void identificador ((__global | __local | __constant) tipoDato Identificador)
La palabra __kernel especifica la función que va a ser ejecutada en el dispositivo. Los tokens _global,_local especifican si se opera para la memoria global, local, entre otras.

3. **¿Cómo se diferencia entre asignación de trabajo a CPU y asignación a GPU?**
OpenCL permite la asignación de trabajo tanto a CPU como a GPU, para especificar la asignación de trabajo, OpenCL provee el parámetro CL_DEVICE_TYPE de la función clGetDeviceIDs. Este parámetro puede ser especificado como CL_DEVICE_TYPE_CPU para la asignación de trabajo a CPU y CL_DEVICE_TYPE_GPU para la asignación de trabajo a GPU.

Hola mundo en OpenCL
1. Analice el código hello.c. A partir del análisis del código, extraiga cuáles son los pasos generales para la generación de aplicaciones utilizando OpenCL. Además, determine, ¿Qué debería cambiar del código, para que el kernel sea ejecutado en un GPU en lugar de un CPU?

Los pasos generales para la generación de aplicaciones utilizando OpenCL son las siguientes:
1. Lectura del archivo kernel.
2. Obtener la lista de plataformas OpenCL
3. Seleccionar el dispositivo
4. Crear un contexto OpenCL.
5. Crear una cola de comandos.
6. Crear los objetos de memoria
7. Crear el objeto del programa.
8. Compilar el programa
9. Crear el objeto kernel
10. Setear los argumentos del kernel
11. Ejecutar el kernel
12. Realizar la lectura de los objetos de memoria
13. Liberar la memoria

Para especificar si el kernel debe ejecutarse en CPU o en GPU, esto se realiza especificando el
parámetro CL_DEVICE_TYPE de la función mostrada en la siguiente figura:

```
ret = clGetDeviceIDs( platform_id, CL_DEVICE_TYPE_CPU, 1,
        &device_id, &ret_num_devices);
```

En la figura anterior se muestra para la ejecución del kernel en CPU, para la ejecución del kernel en GPU se debe cambiar el parámetro CL_DEVICE_TYPE_CPU por el parámetro CL_DEVICE_TYPE_GPU.

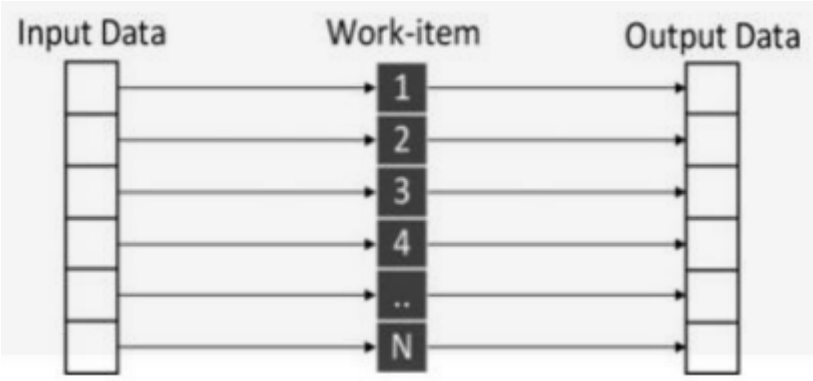
2. Analice el código fuente del kernel hello.cl. A partir del análisis del código, determine ¿Qué operación se realiza con los vectores de entrada? ¿Cómo se identifica cada elemento a ser procesado en paralelo y de qué forma se realiza el procesamiento paralelo?
La operación que realiza los vectores de entrada es la suma vectorial como puede visualizarse en el segmento de código mostrado en la siguiente figura:

```
// Do the operation
C[i] = A[i] + B[i];
```

Para la ejecución de elementos en paralelo, OpenCL provee la función clEnqueueNDRangeKernel. Cada elemento a ser procesado en paralelo se identifica como parámetros en el kernel. Según la documentación oficial[1], los parámetros a especificarse en esta función se muestran en la siguiente figura:

```
cl_int clEnqueueNDRangeKernel (cl_command_queue command_queue,
                               cl_kernel kernel,
                               cl_uint work_dim,
                               const size_t *global_work_offset,
                               const size_t *global_work_size,
                               const size_t *local_work_size,
                               cl_uint num_events_in_wait_list,
                               const cl_event *event_wait_list,
                               cl_event *event)
```

Los parámetros global_work_size especifica los work groups que van a ser procesados, en el código hello este parámetro es de 1024. Para el parámetro local_work_size, este especifica la cantidad de work-items para los cuales va a estar compuesto un work-group, el cual es 64 para el código hello. Estos work-items realizan la ejecución en paralelo como puede visualizarse en la siguiente figura:

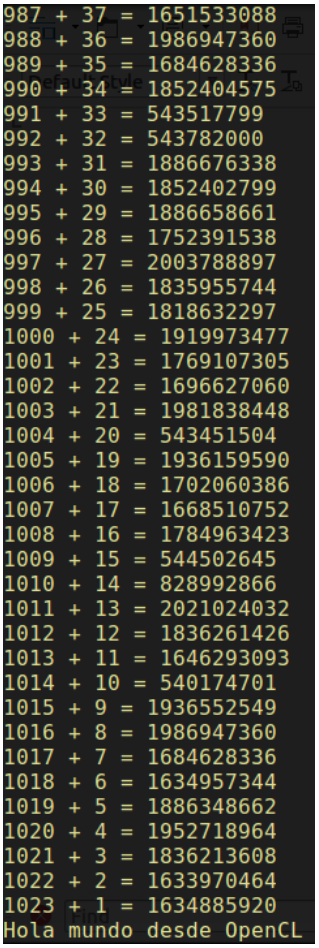


Fuente: OpenCL Programming Book[3]

3. Realice la ejecución de la aplicación hello. ¿Qué hace finalmente la aplicación?

```
988 + 36 = 1024
989 + 35 = 1024
990 + 34 = 1024
991 + 33 = 1024
992 + 32 = 1024
993 + 31 = 1024
994 + 30 = 1024
995 + 29 = 1024
996 + 28 = 1024
997 + 27 = 1024
998 + 26 = 1024
999 + 25 = 1024
1000 + 24 = 1024
1001 + 23 = 1024
1002 + 22 = 1024
1003 + 21 = 1024
1004 + 20 = 1024
1005 + 19 = 1024
1006 + 18 = 1024
1007 + 17 = 1024
1008 + 16 = 1024
1009 + 15 = 1024
1010 + 14 = 1024
1011 + 13 = 1024
1012 + 12 = 1024
1013 + 11 = 1024
1014 + 10 = 1024
1015 + 9 = 1024
1016 + 8 = 1024
1017 + 7 = 1024
1018 + 6 = 1024
1019 + 5 = 1024
1020 + 4 = 1024
1021 + 3 = 1024
1022 + 2 = 1024
1023 + 1 = 1024
Hola mundo desde OpenCL
```

4.Repita los pasos anteriores, pero cambie el código para que se ejecute sobre GPU (sin importar si se cuenta con los drivers o no).



Referencias

[1] Khronos (s. f.). OpenCL - The open standard for parallel programming of heterogeneous systems Recuperado el 07 de Marzo del 2017, de https://www.khronos.org/opencl/

[2] Woolley, C. (s. f.). Introduction to OpenCL Recuperado el 07 de Marzo del 2017, de http://www.cc.gatech.edu/~vetter/keeneland/tutorial-2011-04-14/06-intro_to_opencl.pdf