Activity 1

1. Host

The host refers to the device that runs the OpenCL application, typically the Central Processing Unit (CPU) of a computer. It is responsible for dispatching computational tasks to other devices and managing memory.

1. Device

A device refers to the hardware component capable of performing computational tasks, such as a Graphics Processing Unit (GPU), Digital Signal Processor (DSP), or other accelerators. The host assigns computational tasks to the device for parallel processing.

1. Compute Unit

A compute unit is a parallel processing unit inside a device. Each compute unit consists of multiple processing elements that are capable of executing tasks in parallel.

1. Processing Elements

Processing elements are the basic execution units within a compute unit, responsible for executing instructions. Each processing element typically corresponds to a processing core.

1. Context

Context refers to the execution space within the OpenCL environment, containing devices, memory objects, program objects, and other resources needed to execute OpenCL programs. The context manages the interaction between the host and the devices.

1. Command Queue

A command queue is the mechanism used to send tasks to the device. The host uses the command queue to pass computational tasks to the device for execution. These tasks can include kernel execution, memory operations, etc.

1. Host program

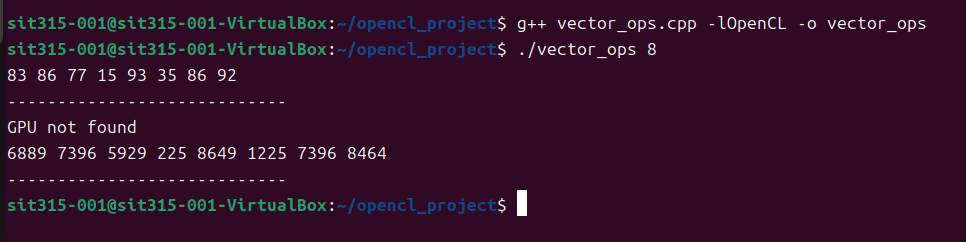
The host program is the application running on the host device (usually the CPU). It is responsible for setting up the OpenCL context, creating the command queue, compiling kernels, and managing data exchanges with the devices.

1. Program kernel

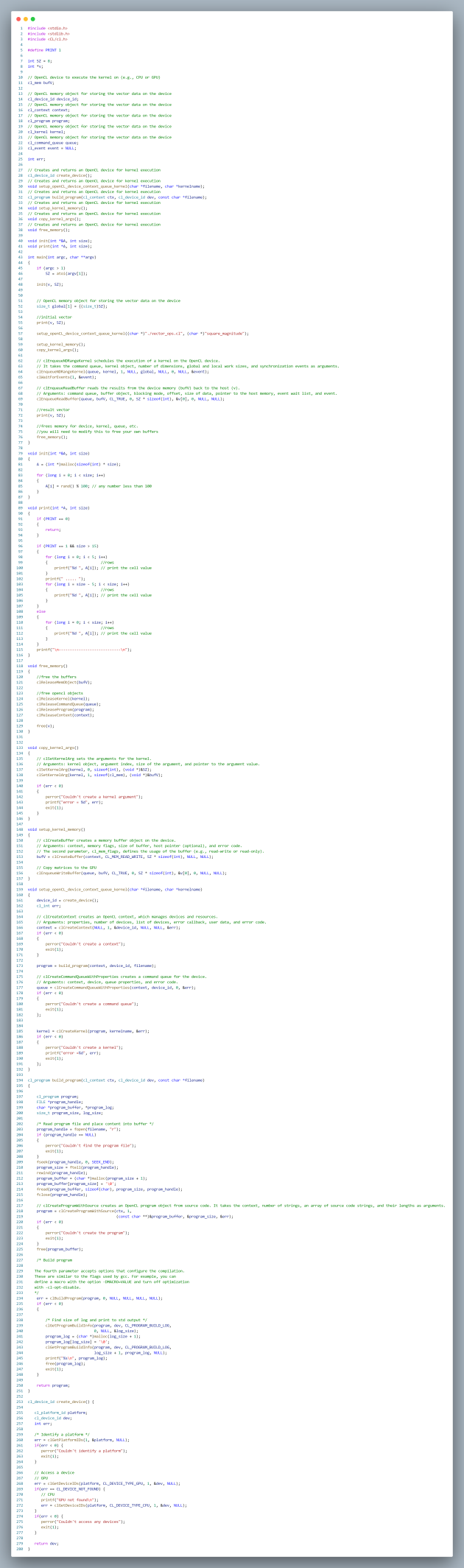
A program kernel refers to the parallelized code in an OpenCL program that executes on the device. It defines the function to perform a specific task and is typically loaded and executed on the device by the host program.

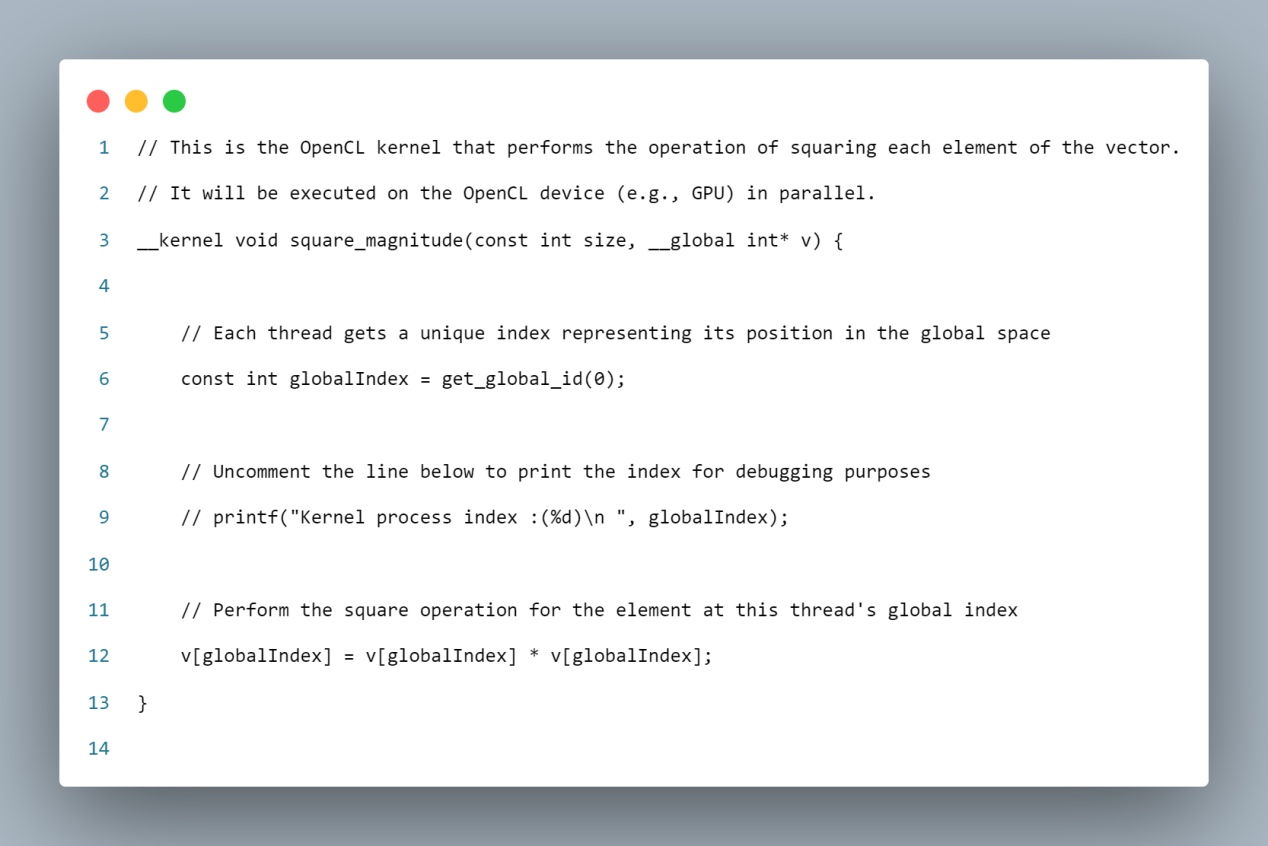
Activity 2

1.

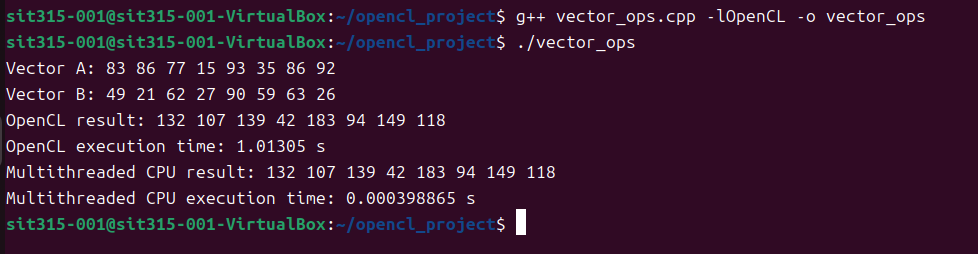


2.





3.



In this task, I compared the performance of OpenCL parallel vector addition with the multi-threaded CPU version. The multi-threaded CPU version runs directly on the CPU without the overhead of data transfer, performing well with a reasonable number of threads. In contrast, the OpenCL version excels at large-scale parallel computation, especially when GPU support is available, where it demonstrates superior performance. However, in a virtual machine environment without a GPU, OpenCL falls back to CPU execution, resulting in similar performance to the multi-threaded version while introducing additional memory copy overhead.