Exercise 02 - Vector addition

In this exercise we will consider the implementation of a kernel for carrying out vector addition. The operation is a fundamental data primitive often used in computations. It is a subclass of the widely used scalar multiplication plus vector addition (SAXPY) computed by the combined operation

$$z = ax + y$$

The generalization of this operation is to conceptually replace vectors with matrices of the same size (see "CUDA Programming Guide" for examples).

Concepts covered

The following concepts are covered in this exercise

- How to allocate and free memory on GPU.
- How to copy data from CPU to GPU.
- How to copy data from GPU to CPU.
- How to invoke GPU kernels.
- How to write a GPU kernel.

Files needed

The following files will be needed for the exercise

- `VecAdd.cu` (needs to be updated)
- `VecAdd_kernel.cu` (needs to be updated)

The code will not work until all work steps described below has been successfully completed.

Work steps

You will need to modify the C code supplied in the files.

The host code in `VecAdd.cu` needs to be updated through the steps

1. Allocate arrays for the vectors x, y and z in device memory.
2. Transfer arrays x, y and z from host to device.
3. Define the number of threads per block and blocks per grid to be used in the invocation of the kernel `VecAdd_kernel.cu`.
4. Transfer vector c from device to host memory.
5. Free allocated device memory for vectors x, y and z.

The device code in `VecAdd_kernel.cu` needs to be updated through the steps

6. Modify the kernel such that per thread it reads values from x and y from global device memory and stores the vector addition result for the element in question in z back in global device memory.

Compilation

Compile the final code using the included `Makefile` which includes the common parameters for compilation in `../../common.mk`. To compile successfully after CUDA has been successfully installed, you will need to make sure that the environment variables `CUDA_INSTALL_PATH` and `CUDA_SDK_DIR` are correct. In case of compilation errors you will need to debug the code until it compiles successfully.

Execution

Execute your compiled program. If you program executes successfully the output should look like this

```
Vector addition
PASSED
```