CSUS COLLEGE OF ENGINEERING AND COMPUTER SCIENCE
Department of Computer Science

CSc 163 – Parallel Programming with GPUs
Spring 2022 / Dr. Muyan

Assignment#4 (OpenCLVectorAdd)

Due Date: Monday, May 16th 11:59pm

(**This is also the final date for submission of A4 since the 5-day late policy does NOT apply to the last assignment. Hence, assignments submitted after 11:59pm Monday, May 16th will not be graded.)**

Steps for solving the assignment (refer to "Assignment Environment" lecture slides for details):

- Download the assignment from the Canvas and unzip it (now you have "A4" directory - you must have already done this to read this a4-instructions.pdf file).
- Build the source using CMake (in "build" directory that you create under "A4" directory).
- Copy the OpenCLVectorAdd dataset directly under "build" directory (under "A4/datasets" directory, we have only one dataset directory called OpenCLVectorAdd, copy it to "A4/build", e.g., copy contents of "A4/datasets/ OpenCLVectorAdd" to "A4/build/OpenCLVectorAdd").
- For the OpenCLVectorAdd problem (which is the only problem provided in A4):
    - Set the project properties as indicated in the below "Project Setup" instructions (e.g., add OpenCL.lib to the list of additional dependencies and set command line options).
    - Update the template code (template.cpp) as indicated in the below "Coding" instructions to solve the problem.
    - Build your solution under Visual Studio (right click on the project and hit "Build") and run it under Visual Studio (right click on the project and hit "Debug -> Start new instance").
    - Make sure your executable (located under "Debug" directory that resides under "build" directory) works from the command line as indicated in the below "Command-line Execution" instructions.

    - Create a new directory by giving it the name of the problem (i.e., OpenCLVectorAdd).
    - Copy the executable file (.exe file) and the updated template code (template.cpp) to this newly created directory.
- Zip the newly created directory in a file named as YourLastName-YourFirstName-a#.zip (e.g., Doe-Jane-a4.zip).
- Build (using CMake and Visual Studio) and test your assignment in a lab machine (**this step is especially important for the A4**, see the syllabus for tips) and **create a TEXT (i.e., not a pdf, doc etc.) file called "readme-a4.txt"** that indicates the lab and the lab machine you have used to test your assignment. You may also include additional information you want to share with the grader in this text file. **Make sure you have the executable files (.exe files) generated/tested on the lab machine in the zip file mentioned above.**
- Submit the **TWO files you have created above (zip file and txt file) SEPERATELY to Canvas** (i.e., do NOT place the txt file inside the zip file). You will receive the grader comments on your text file when grades are posted.

OpenCLVectorAdd dataset directory provides **five test cases**. You should test your program with all the provided test cases by updating the command line options provided in the below "Project Setup" instructions. For instance, to test test_case_1 of VectorAdd (located at "build\VectorAdd\Dataset\1") you should update the command line option provided below (which tests test_case_0 located at "build\VectorAdd\Dataset\0") by changing all subtexts that say "…\Dataset\0\..." with "…\Dataset\1\...".

If you want to quickly check whether all datasets work from the command prompt, you can use the attached text file (A4_batch.txt) that lists the command-line execution lines for the datatsets excluding the last part from the line that directs the output to result.txt. This allows us to display all results on the console instead of writing them to result.txt files. Copy/paste the contents of these text files to command prompt when you are under "build" directory to see whether your code returns correct results for all datasets.

Objective

The purpose of this problem is to introduce you to the OpenCL API by implementing vector addition. You will implement vector addition by writing the GPU kernel code as well as the associated host code.

Coding

Edit the template.cpp to perform the following:

- Initialize the workgroup dimensions
- Bind to platform
- Get ID for the device
- Create a context
- Create a command queue
- Create the compute program from the source buffer
- Build the program executable
- Create the compute kernel in the program we wish to run
- Create the input and output arrays in device memory for our calculation
- Write our data set into the input array in device memory
- Set the arguments to our compute kernel
- Execute the kernel over the entire range of the data set
- Wait for the command queue to get serviced before reading back results
- Read the results from the device
- Write the OpenCL kernel

Instructions about where to place each part of the code is demarcated by the `//@@` comment lines.

Project Setup

Since this is an OpenCL project, first you need to set following properties in this project:

- Add OpenCL.lib to the list of Additional Dependencies:

  OpenCLVectorAdd_Template project -> Properties -> Configuration Properties -> Linker -> Input -> Additional Dependencies and add the following to the beginning of the line:

$(CUDA_LIB_PATH)\OpenCL.lib;

(Make sure that your CUDA_LIB_PATH environment variable is assigned to the directory that has OpenCL.lib file. If you have CUDA 11.0 installed for instance this directory would be located at:

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.0\lib\x64)

Then you also need to set following properties to test your program on test_case_0:

- Right click on the OpenCLVectorAdd_Template project -> Properties -> Configuration Properties -> Debugging -> Commmand Arguments and enter the following:

```
-e .\OpenCLVectorAdd\Dataset\0\output.raw

-i

.\OpenCLVectorAdd\Dataset\0\input0.raw,.\OpenCLVectorAdd\Dataset
\0\input1.raw

-o .\OpenCLVectorAdd\Dataset\0\myoutput.raw -t vector >

.\OpenCLVectorAdd\Dataset\0\result.txt
```

(all in one line - do not forget to put spaces between the sub-lines - above you see five sub-lines)

Here, `input0.raw and input1.raw` are the input vectors. `output.raw` is the expected result (output vector that includes the addition of input vectors). `myoutput.raw` is your result.

You will see the output of your execution in `result.txt` (which indicates whether your result matches the expected result) which resides under `build\OpenCLVectorAdd\Dataset\0\`

Command-line Execution

The executable generated as a result of compiling the project (`build\Debug\ OpenCLVectorAdd_Template.exe)` can be run from the command-line using the following command (make sure you are in `build` directory):

```
.\Debug\OpenCLVectorAdd_Template

-e .\OpenCLVectorAdd\Dataset\0\output.raw

-i

.\OpenCLVectorAdd\Dataset\0\input0.raw,.\OpenCLVectorAdd\Dataset
\0\input1.raw

-o .\OpenCLVectorAdd\Dataset\0\myoutput.raw -t vector >

.\OpenCLVectorAdd\Dataset\0\result.txt
```

(all in one line - do not forget to put spaces between the sub-lines - above you see six sub-lines)