Assignment 2

Multiprocessor Programming 521288S

Miika Sikala 2520562, msikala18@student.oulu.fi

Task: Element-wise addition of two matrices.

1. Implement a C program, that performs element-wise addition of two matrices. The implementation should define a "Add_Matrix" function that considers 2 100x100 matrices as input. The matrices have to be created using two dimensional pointers such as malloc, calloc etc. The final execution time of the implementation should be displayed as output.
2. Similarly, implement a simple OpenCL kernel "add_matrix" performing element-wise addition of two matrices. The add_matrix kernel should consider two matrices as the input similar to C function. The final execution time of the implementation should be displayed as output. Also, display the compute device information that is used for this implementation as the output (Check the "HelloWorld" report attached here for reference)

Expected result: A working code and 2-page report, all together in the form of a compressed folder

The report should contain about the task solved, brief description of your implement and the final screenshot of your outputs asked to be displayed under the assignment. In case any of them are very new to OpenCL, I am sharing a simple HelloWorld program report as an attachment here for reference. Please, upload your compressed folders using "studentname_studentnumber_assignment2.zip".

Introduction

As requested, I implemented 2 programs. The first one (add_matrix_no_ocl.c) sums two 100x100 matrices without OpenCL. The program is straightforward; first we allocate memory for the matrices, populate them with some data and then pass them to a function that sums the two. The result is stored in another matrix.

The second program (add_matrix.c) utilizes OpenCL to sum the matrices using parallelization on hardware (either GPU or CPU). The implementation is more complicated due to the boilerplate code required by OpenCL. Since we are summing two 2-dimensional matrices, we set the kernel dimensions to 2 and give global and local indices in two dimensions. To distribute the workload between compute units, we set a work group size of 10x10 (localSize). The target device can be changed using flag USE_CPU (0 = GPU, 1 = CPU).

We can optimize the implementation by accounting for the number of compute units and other settings in the target device.

Compiling using *gcc*:

```
gcc -Wall -I %OCL_ROOT%\include add_matrix.c %OCL_ROOT%/lib/x86_64/opencl.lib -o
add_matrix.exe && add_matrix.exe
```

Testing the implementations

The correctness of the implementations was tested by checking the results manually. The screenshots of the outputs of both programs are shown below.

Figure 1 shows the execution on CPU without OpenCL. Figures 2 and 3 show the execution of the OpenCL programs on GPU and CPU, respectively. The outputs also show the execution time and device information.

```
Sum two 100x100 matrices without using OpenCL.
Execution time: 121.000 us
```

Figure 1. Output of the first program, without OpenCL.

```
Sum two 100x100 matrices using OpenCL on GPU.
Compute device info:
        Device name:                    Ellesmere
        Device type:                    GPU
        Vendor ID:                      4098
        Maximum frequency:              1340 MHz
        Driver version:                 2639.5
        Device C version:               OpenCL C 2.0
        Compute units:                  36
        Max. work item dimensions:      3
        Max. work item sizes:           1024/1024/1024
        Max. work group size:           256
Execution time: 5.600 us
```

Figure 2. Output of the second program, with OpenCL on GPU.

```
Sum two 100x100 matrices using OpenCL on CPU.
Compute device info:
        Device name:                    AMD FX(tm)-8320 Eight-Core Processor
        Device type:                    CPU
        Vendor ID:                      4098
        Maximum frequency:              3793 MHz
        Driver version:                 2639.5 (sse2,avx,fma4)
        Device C version:               OpenCL C 1.2
        Compute units:                  8
        Max. work item dimensions:      3
        Max. work item sizes:           1024/1024/1024
        Max. work group size:           1024
Execution time: 116.600 us
```

Figure 3. Output of the second program, with OpenCL on CPU.

Execution times:

C implementation: 121.0 us

OpenCL (CPU): 116.6 us

OpenCL (GPU): 5.6 us

As the benchmark shows, the GPU implementation is extremely efficient compared to the plain C implementation. The CPU OpenCL implementation speeds up the calculation somewhat, but due to the number of compute units, the gain is very minimal (at least with local size 10).

References

[1] https://docs.microsoft.com/en-us/windows/win32/sysinfo/acquiring-high-resolution-time-stamps

[2] https://stackoverflow.com/questions/23550912/measuring-execution-time-of-opencl-kernels

[3] https://moodle.oulu.fi/pluginfile.php/1071684/mod_resource/content/1/OpenCL-1.2.pdf

Reporting

The “OpenCL implementation” includes some study of the OpenCL reference.

Task	**Hours**
C implementation	3h
OpenCL implementation	6h
Total	**9h**