School of Computing: assessment brief

Module title	Parallel Computation
Module code	XJCO3221
Assignment title	Coursework 3
Assignment type and description	OpenCL Programming Assignment
Rationale	Implementation of a general purpose GPU program in OpenCL. Correctly implement a GPU kernel to perform a common numerical task.
Guidance	See overpage (equivalent to COMP3221)
Weighting	15%
Submission deadline	10am (China), Tuesday 9th May.
Submission method	Gradescope
Feedback provision	Marks and comments returned *via* Gradescope
Learning outcomes assessed	Apply parallel design paradigms to serial algorithms. Evaluate and select appropriate parallel solutions for real world problems.
Module lead	Peter Jimack

1. **Assignment guidance**

   For this coursework you are required implement OpenCL code for matrix-vector multiplication in which the calculation is performed on a GPU. The matrix M is assumed to be square (but not necessarily symmetric), with N rows and N columns, and the vector $\mathbf{x}$ is of size N. The solution $\mathbf{y}$, also an N-vector, is then given by $\mathbf{y} = M\mathbf{x}$. For the purposes of this coursework, N must be a power of 2, and can take any value up to and including $N = 256$.

2. **Assessment tasks**

   To ensure the matrix occupies one contiguous region in memory, it is allocated as a one-dimensional array `M` of size `N*N`. This means that the matrix element at row `row` and column `col` is indexed as `M[row*N+col]`. If the vectors `x` and `y` are stored as standard vectors of size `N`, matrix-vector multiplication can be written in serial C-code as

   ```
   int row, col;
   for( row=0; row<N; row++ )
        for( col=0; col<N; col++ )
             y[row] += M[row*N+col] * x[col];
   ```

   Note this assumes that the vector `y` has first been initialised to all zeroes.

   Starting code has been provided that reads the value of `N` from the command line, and then allocates host (CPU) memory for the matrix, `hostMatrix`, the vector `x` (‘`hostVector`’), and the solution vector `y` (‘`hostSolution`’). It also initialises the matrix and the vector with random values. After compiling the code as per the instructions in lectures (using `nvcc -lOpenCL`), execute on the batch queue using the script provided (by typing `sbatch cwk3-submit.sh`). Note that the content of `cwk3-submit.sh` is initially as follows:

   ```
   #!/bin/bash

   #SBATCH --partition=gpu --gres=gpu:t4:1

   ./cwk3 8
   ```

   This will initialise and display an 8×8 matrix and 8-vector. It will also display the solution vector, which is currently all zeroes. You need to allocate device memory, implement and launch an OpenCL–kernel, and transfer data to and from the GPU, to calculate the output vector correctly.

   You should use a 1-dimensional index space size when launching your `NDRangeKernel`; you will gain no extra marks by using a 2–dimensional kernel (although a 2–dimensional kernel is in principle more efficient, it requires techniques we have not yet covered in lectures).

   The starting code consists of the following files:

`cwk3.c`	:	The starting point for your solution.
`ckw3.cl`	:	The kernel code. Currently this file only contains a comment.
`helper_cwk.h`	:	Includes the same helper routines as used in the lecture examples, plus some specific routines for this coursework.
`cwk3-submit.sh`	:	A simple submission script to use with `sbatch` on `cloud-hpc1.leeds.ac.uk`.

3. **General guidance and study support**

   If you have any queries about this coursework, in the first instance please ask during your timetabled lab session.

   Please note that you only need to use the material in Lectures 14, 15 and 16 for this coursework.

4. **Assessment criteria and marking process**

   Your code will be downloaded from Gradescope and tested for functionality on a system that is identical to `cloud-hpc1.leeds.ac.uk`. Staff will then inspect your code the allocate the marks as per the mark scheme (see below), and then upload the mark and feedback to Gradescope.

5. **Submission requirements**

   Submission is *via* Gradescope.

   You should only modify the files `cwk3.c` and `cwk3.cl`, consequently only these should be uploaded.

   It is essential that you **do not alter the file `helper_cwk.h` or remove calls to the routines it contains**, as it will be replaced with a modified version for the assessment.

   All submissions will be compiled and executed on a Linux machine that is equivalent to `cloud-hpc1.leeds.ac.uk`.

   Note that there is no **Check submission** portal for for this coursework as previously, as Gradescope's autograder system does not support OpenCL (even on a CPU). Therefore you should submit your solution directly to the submission portal for assessment.

6. **Academic misconduct and plagiarism**

   Academic integrity means engaging in good academic practice. This involves essential academic skills, such as keeping track of where you find ideas and information and referencing these accurately in your work.

   By submitting this assignment you are confirming that the work is a true expression of your own work and ideas and that you have given credit to others where their work has contributed to yours.

   Code similarity tools will be used to check for collusion, and online source code sites will be checked.

7. **Assessment/marking criteria**

   There are 15 marks in total:

6 marks	:	Calculation is performed correctly.
4 marks	:	Kernel design, execution, and management.
5 marks	:	Device memory usage and management.