UOW AUSTRALIA

SCIT

School of Computing & Information Technology

CSCI376 – Multicore and GPU Programming
SIM Session 2 2022

Assignment 1

Task 1 – Device Information (7 marks)

Write a program that uses OpenCL to do the following:

- Allow the user to enter whether they want to use a CPU or GPU device. Based on the user's selection, search the system for all CPU or GPU devices. (Note that some systems have multiple CPUs and GPUs).

  (1 mark)

- Based on the user's choice, display the following information for each CPU/GPU device that is available on the system:
  - Name of the platform that supports that device
  - Device type – CPU or GPU
    (CL_DEVICE_TYPE – either CL_DEVICE_TYPE _CPU or CL_DEVICE_TYPE_GPU)
  - Device name
    (CL_DEVICE_NAME)
  - Number of compute units
    (CL_DEVICE_MAX_COMPUTE_UNITS)
  - Maximum work group size
    (CL_DEVICE_MAX_WORK_GROUP_SIZE)
  - Maximum number of work item dimensions
    (CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS)
  - Maximum work item sizes
    (CL_DEVICE_MAX_WORK_ITEM_SIZES)
  - Global memory size
    (CL_DEVICE_GLOBAL_MEM_SIZE)
  - Local memory size
    (CL_DEVICE_LOCAL_MEM_SIZE)

  (2 marks)

- Based on the devices available, allow the user to select one device. Create a context using that device, and a command queue.

  (1 mark)

- Read the program source code from the provided "task1.cl" file and build the program. Display whether or not the program built successfully and display the program build log (display the build log even if the program built successfully).

  (1.5 marks)

- Find and display the number of kernels in the program. Create kernels from the program and display all the kernel names.

  (1.5 marks)

Task 2 – Data Management (8 marks)

Write a program that uses OpenCL to do the following:

- Create a C++ vector of unsigned chars to store alphabets. Initialise its content to: a-z and A-Z (i.e. 52 alphabets in total). Create another C++ vector to store 512 unsigned ints. Initialise its content to: 1-512.

  (1 mark)

- Create three OpenCL memory objects (i.e. cl::Buffer objects):
  - The first buffer is read-only and initialised with the contents of the alphabet vector.
  - The second buffer is write-only and created to store 52 unsigned chars.
  - The third buffer is read-and-write and created to store 512 unsigned ints.

  (2 marks)

- Enqueue two OpenCL commands:
  - To copy the contents from the first buffer into the second buffer.
  - To write the contents from the vector of 512 integers into the third buffer.

  (2 marks)

- Setup the OpenCL program to allow the user to select one device, create a context and command queue for that device. Then, build the provided "task2.cl" program and create a kernel for "task2".

  (1 mark)

- Set kernel arguments for the kernel that was previously created. For the first argument, pass a floating-point value of 21.43 to the kernel. For the second and third kernel arguments, set these to the second and third buffers that were previously created. Then, enqueue the kernel using the enqueueTask function.

  (1 mark)

- After returning from the enqueueTask function, read the contents from the two buffers, and display the results on screen.

  (1 mark)

Task 3 – Kernel Execution (10 marks)

Write an OpenCL program that uses a kernel (you will have to write the kernel yourself) to fill in the contents of an array of 1024 numbers in parallel. The program is to prompt the user to enter a number between 1 and 89 (inclusive). The program is to check whether the user entered a valid number, if not the program will quit. If a valid number was entered, enqueue a kernel (using the enqueueNDRangeKernel function) that accepts the number and an array, and fills in the contents of the array using the number (and the work-items' global IDs) as follows:

- If the user enters 1, the resulting contents of the array should be: 1, 2, 3, 4, 5,… until 1024
- If the user enters 2, the resulting contents of the array should be: 1, 3, 5, 7, 9,… until 2047
- If the user enters 3, the resulting contents of the array should be: 1, 4, 7, 10, 13,… until 3070
- …
- If the user enters 89, the resulting contents of the array should be: 1, 90, 179, 268, 357,… until 91048

After kernel execution, display the resulting contents on screen.

Write a description of what your program does. You may draw diagrams if it helps with your explanation. Include this with your submission.

For **ALL** tasks, include **screenshots** with your submission. The screenshots are to demonstrate that the programs work on your computer.

Please follow the tutor's instructions.

Instructions and Assessment

Submit your tasks as one zip file with three folders, named Task1, Task2 and Task3, and include all the required files in your submission.

The assignment must be your own work. If asked, you must be able to explain what you did and how you did it. Marks will be deducted if you cannot correctly explain your code. The marking allocations shown above are merely a guide. Marks will be awarded based on the overall quality of your work. Marks may be deducted for other reasons, e.g., if your code is too messy or inefficient, is not well commented, if you cannot correctly explain your code, etc. For code that does not compile, does not work or for programs that crash, the most you can get is half the assessment marks or less.

