Exercises for lecture~06

Heterogeneous Computing for AI [PG2015]

Raghava Mukkamala (rrm.digi@cbs.dk, Copenhagen Business School, Denmark

https://raghavamukkamala.github.io/

Exercise 1: Converting an image to Grayscale using numpy arrays

Use the img_grayscale.py to answer this question.

To read on how images are represented as numpy arrays: https://www.geeksforgeeks.org/how-to-convert-images-to-numpy-array/

What is grayscale conversion: (Using Weighted or luminosity method) https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm

The file img_grayscale.py provides code whichloads the RGB 'lenna.png' image into numpy arrays. It then converts the image (with 3D arrays for RBG) with shape (512,512,3) into 2D arrays of shape (512,1536). The RBG values for each pixel are stored next to each other.

Your task is to go through these arrays and convert them to grayscale by using the following formula: grayscale_pixel = 0.2989*R + 0.5870*G + 0.1140*B

You do this by completing the grayscale_converter function.

The file has more information on how the task is split.

Exercise 2: Choosing Correct indexing + block/grid structure

We want to double the value of each element in a vector/array.

a) If we want to use each thread to calculate one output element of the vector doubling, what would be the expression for mapping the thread/block indices to data index?

```
A. i=threadIdx.x + threadIdx.y;

B. i=blockIdx.x + threadIdx.x;

C. i=blockIdx.x*blockDim.x + threadIdx.x;

D. i=blockIdx.x * threadIdx.x;
```

Once you've chosen your answer - Place it in the SourceModule - at the place called "TODO: write your mapping expression here"

We are not finished yet.

b) To make the code run, we have to also give it the block dimensions and grid dimensions. Place the correct dimensions for the grid and the block. Look for TODOs.

Please see google colab week6-ex2.ipyn for solution.

Exercise 3: Continuing from 2

We want to double the value of each element in a vector/array.

Assume that we want to use each thread to calculate two (adjacent) elements of a vector/array doubling. What would be the expression for mapping the thread/ block indices to i, the data index of the first element to be processed by a thread?

```
A. i = blockIdx.x*blockDim.x + threadIdx.x +2;
B. i = blockIdx.x*threadIdx.x*2;
C. i = (blockIdx.x*blockDim.x + threadIdx.x)*2;
D. i = blockIdx.x*blockDim.x*2 + threadIdx.x;
```

Note that you need only half the number of threads as you did in the previous question.

Exercise 4: Continuing from 3

We want to double the value of each element in a vector/array. And every thread doubles 2 values (but not consecutive values this time).

In the following we will use the following Grid and Block dimensions/ structure.

grid_dim = (2,1) block_dim = (32,1,1) We'll have a total of 64 threads.

We want to use each thread to calculate two elements of a vector/array doubling. Each thread block processes 2*blockDim.x consecutive elements that form two sections. All threads in each block will first process a section first, each processing one element. They will then all move to the next section, each processing one element. Assume that variable i should be the index for the first element to be processed by a thread. What would be the expression for mapping the thread/block indices to data index of the first element?

```
A. i=blockIdx.x*blockDim.x + threadIdx.x +2;
B. i=blockIdx.x*threadIdx.x*2;
C. i=(blockIdx.x*blockDim.x + threadIdx.x)*2;
D. i=blockIdx.x*blockDim.x*2 + threadIdx.x;
```

Hint: Lets divide the array of 128 elements into 4 parts as follows:

```
part A : element 0 to 32
part B : element 32 to 64
part C : element 64 to 96
part D : element 96 to 128
```

As an example: a thread that works on element 0 in part A will also work on element 64 in part C. A thread that works on element 35 in part B will also work on element 99 in part D.

Exercise 5: Selective doubling - Host and Device interaction

Lets start with the same setup we had for the previous exercises. Use the same tempelate notebook.

In this question, we are only going to double the values of an array every 4 elements. That is, element in position/index 0 will get its value doubled, then element in position/index 4 will get doubled. Similarly, the other indexes of values where value is doubled are 8,12,16,20...

We are going to solve this in two ways:

i. Please solve this question by loading the entire array of values into the device memory. Then double only values that are in index position described above. This is very similar to what we've been doing in the past 3 exercises.

ii. Please solve this question by ONLY sending in the values that we will be doubling into the device memory. I.e. Do not store the entire array in the device but only store the values which are going to be doubled. In this case, the array that is going to be sent/stored in the device memory will be 128/4 = 32 elements only.

```
First, extract these 32 elements from the array (indexed at 0,4,8,12...)
Transfer only these 32 elements into the device memory.
Then double these values and then write them back to host memory.
Put these values back into the original array in their correct places.
The final array should match the array we get in part i of this question.

(You will need to change most of the code base to answer this question.)
```