Problem set 2, CUDA Intro

TDT4200, Fall 2015

Deadline: 17.09.2015 at 23:59. Contact course staff if you cannot meet the deadline.

Evaluation: Pass/Fail

Delivery: Use itslearning. Deliver exactly two files:

- *yourNTNUusername_ps2.pdf*, with answers to the theory questions
- *yourNTNUusername_code_ps2.{zip |tar.gz |tar}* containing your modified versions of the files:
  - `Makefile`
  - `lenna.cu`
  - `lodepng.h`
  - `lodepng.cpp`
- Do *not* include `lenna512x512_inv.png`. Your handed in archive *must* have the same folder structure as the archives handed out (all files unpacked directly, no sub-directories whatsoever). Failing to do this can impact your passing or *failing*.

General notes: Code must compile and run on the following systems:

1. `its-015-XX.idi.ntnu.no` (`XX` being any of the lab machines in ITS015).

You should only make changes to the files indicated. Do not add additional files or third party code/libraries.

Part 1, Theory

Problem 1, Architectures & Programming Models

a) Briefly explain the differences between the following architectures:
Keywords being: Homogeneous cores, heterogeneous cores, clusters, NUMA, threads.

  i) Nvidia Maxwell
  ii) ARM big.LITTLE
  iii) Vilje @ NTNU
  iv) Typical modern-day CPU

b) Explain Nvidia's SIMT addition to Flynn's Taxonomy, and how it is realized, if at all, in each of the architectures from a).

c) For each architecture from a), report which classification from Flynn's Taxonomy is best suited. If applicable, also state if the architecture fits Nvidia's SIMT-classification. Give the reasoning(s) behind your answers.

Problem 2, CUDA GPGPUs

In the following subproblems, all questions asking for explanations/reasonings of/behind CUDA/NVIDIA terms/technology should be based on the Maxwell architecture, if the answer to the question differs between the different NVIDIA architectures.
Also mention this is the case when applicable.

a) Explain the terms *Threads*, *Grids*, *Blocks*, and how they relate to one another (if at all).

b) Consider an algorithm whose input has size $2n$, output has size $5n$, and execution time is $5hn \cdot 7h \cdot log_2(n)$ where $h = 1$ on the GPU and $h = 10$ CPU. The CPU-GPU bus has a bandwidth of r. How big must n be before it is faster to run the dataset with the algorithm detailed on the GPU instead of the CPU?

c) Which of `kernel1()` and `kernel2()` in Listing 1 will execute fastest, given that X and Y are gridDim and blockDim respectively, containing 3 integers with positive powers of 2 higher than 2^4? Explain why.

```
int main(){
    ...
    <<<X, Y>>>kernel1();
    <<<X, Y>>>kernel2();
    ...
}

void kernel1(){
    for(int i = 0; i < 8; ++i){
        if (threadIdx.x % 8 == i || threadIdx.y % 8 == i){
            some_other_task();
        }
    }
}

void kernel2(){
    for(int i = 0; i < 8; ++i){
        if (blockIdx.x % 8 == i || blockIdx.y % 8 == i){
            some_other_task();
        }
    }
}
```

Listing 1: CUDA code example.

d) Explain each of the following terms, and how each should be utilized for maximum effect in a CUDA program:
(Use prior problems in this problemset to illustrate their use where applicable).

i) Warps
ii) Occupancy
iii) Memory Coalescing
iv) Local Memory
v) Shared Memory

Figures can be used to answer this subproblem.

Part 2, Code

Neither of the programs you implement and hand in should print anything, nor save their output images to any other locations, or names, than as specified in the example code given!

In this problemset, we've given you two `zip`-files, named `lenna.zip` and `pinkfloy.zip`, respectively. We will first discuss the contents and program(s) included in `lenna.zip`.

The `Makefile` handed out in `lenna.zip` compiles the handed out `lenna.c` with the help of `lodepng.c`, creating the executable named "`cpu_version`". `cpu_version` executable has a serial approach to inverting the grey/white color-scale in the picture `lenna512x512_inv.png` back to its original shade/color.

Your job is to implement a program performing the exact same operations, except that the operations inverting the grey/white scale `unsigned char` values must happen on the GPGPU. In other words, you need to write a program utilizing an Nvidia kernel which performs this task.

`lodepng.cpp` is added in addition to its `.c` version, so that you can use the given `lenna.cu` file to implement your program.

Problem 1, CUDA Intro

a) In the CUDA file `lenna.cu` implement a kernel which performs the same job as the executable `cpu_version` does. The additionally required setup of memory and variables, freeing of the same, and transfers wrt. to the CUDA kernel are also required.

b) Implement a "`make cuda`" makefile rule which compiles (but does not execute) the CUDA executable "`gpu_version`".

c) Time the transfers of data to and from device, and report the percentage of total program run-time the transfers require. How would you suggest to improve this percentage?

Problem 2, Pinkfloyd Intro

NB: *This program is* not *a part of the mandatory hand-ins for this problem set.*
However, later PSs *will* use similar code, so it's worth having done, in preparation for later.

This program receives primitives as input for a canvas of certain quadratic size, and draws the different primitives on the canvas before saving the canvas-image as a `image.png`.
The program receives input from command line redirection at execution. The input is structured as follows:

```
<canvaswidth>,<canvasheight>
<number of primitives in input>
<primitive one>
<primitive two>
...
```

Primitives can be defined as the following:

```
circle centerx,centery radius hue,value
line startx,starty endx,endy thickness hue,value
```

- `hue` is a float given in degrees, from $[0, 360]$, inclusive.
- `value` is a float with value between $[0, 255]$, inclusive.
- The coordinates are given as float, with values from $[0, 1]$, inclusive.
  - The coordinate system is $(0, 0)$ in top left corner, and $(1, 1)$ in bottom right corner.

If two primitives overlap, their color `value`s are simply added together.

Additional details can be found in the recitation slides for this Problem set.