TDT4200 Problem Set 5
Introduction to CUDA

Mandatory (Pass/fail)

The assignment is due by Thursday 25/10-2012, at 20:00.

If you cannot meet this deadline, please contact the instructor.

Answers should be submitted on It's Learning. Your answer should consist of two files: `mandel.cu` for the CUDA assignment and [username]`.pdf` for the remaining tasks. The code should compile and run on the computer lab machines. Please make sure the code is readable with sensible comments.

1 Theory

1. Explain the architectural differences between modern CPUs and GPUs.
2. Mention a problem better suited to the GPU than the CPU. Explain why it is better suited to the GPU.
3. Mention a problem better suited to the CPU than the GPU. Explain why it is better suited to the CPU.
4. For some applications, the actual computation is faster on the GPU than CPU. However, the CPU version might still be faster. Explain why this can happen.

2 CUDA Programming

The topic for this assignment is the Mandelbrot fractal. You are given an unfinished CUDA program `mandel.cu` as well as a serial CPU version `mandel_c.c` for convenience. This program calculates the Mandelbrot set and optionally saves the result to a `.bmp` file. Call the program with `./mandel 1` to save the image file, and call with `./mandel 0` to calculate without saving. See figure 1 for a screenshot. The CUDA version creates `mandel1.bmp` and the C version creates `mandel2.bmp`.

For a given point, its coordinates are represented by a complex number c where the real and imaginary parts can be interpreted as the x and y-coordinates respectively. For each point, perform the following algorithm, where the input is c and the output is i, the number of iterations:

1. Let $z \leftarrow c$ and $i \leftarrow 0$. Define a maximum number of iterations MAX you want to perform.
2. If $|z| \geq 2$ or $i = \text{MAX}$ then terminate.
3. Let $z \leftarrow z^2 + c$.
4. Let $i \leftarrow i + 1$.
5. Go to 2.

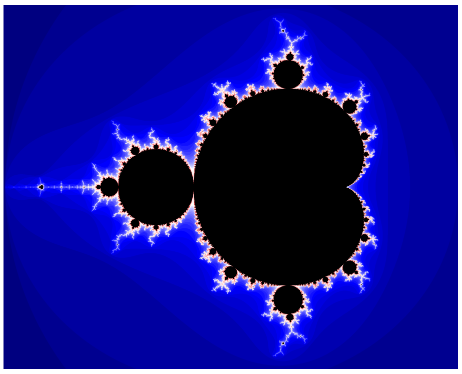

Figure 1: The Mandelbrot fractal

If z never diverges (that is, its absolute value never exceeds 2) the point c is said to be in the Mandelbrot set. Otherwise, the number of iterations i is the *escape time* for this point, and it will be used to determine the pixel colour. The absolute value of a complex number $a + bi$ is defined as $\sqrt{a^2 + b^2}$. In our program we use the equivalent and computationally cheaper condition $a^2 + b^2 \geq 4$ to check for divergence.

The c values are typically scaled so that the real part is contained in $[-2, 1]$ and the imaginary part is contained in $[-1, 1]$. The variables `xleft`, `xright` and `ycenter` can be changed if you want to draw another portion of the Mandelbrot fractal. The `yupper` and `ylower` variables are calculated from the others in a way that preserves the aspect ratio. The arrays `host_pixel` and `device_pixel` are both one-dimensional, and are arranged in a row-major order, which means that coordinate (x, y) is stored in index $x + y*$`XSIZE`. The provided macro `INDEX` will convert a pair of coordinates (x, y) to a one-dimensional index (see in `host_calculate()` for an example usage). Be aware that the first parameter is the x coordinate, and the second parameter is the y coordinate.

Your task is to finish `mandel.cu`. There are 5 subtasks to perform:

1. Create the kernel that performs the calculations.
2. Set up device memory.
3. Execute the kernel on the device.
4. Transfer the result from the device to the host.
5. Free the device memory.

Report the speedup that the CUDA program achieves compared to the serial version that runs on the host (CPU). Remember to count both the calculations and memory copying in the CUDA run time. `mandel.cu` has a serial version included, as well as code for timing both versions. Please use the original values of `xleft`, `xright` and `ycenter` for this part.

Floating point calculations on the GPU aren't necessarily IEEE 754 compliant, so you should expect a few errors even if your program is correct. If you can't spot the difference between the images from the CUDA version and the C version, your program is considered correct even if some there are tens or hundreds or errors.