Problem set 6, SIMD Vectorization
TDT4200, Fall 2015

Deadline: 29.10.2015 at 23:59. Contact course staff if you cannot meet the deadline.

Evaluation: Pass/Fail

Delivery: Use It's Learning. Deliver exactly two files:

- *yourNTNUusername_ps6.pdf*, with answers to the code questions
- *yourNTNUusername_code_ps6.{zip |tar.gz |tar}* containing your modified versions of the files:
  - `Makefile`
  - `newImageIdea.c`
- Do *not* include any image files.
- Deliver the files the same way as on CMB/PS3 (in a zip file), preferably the same zip file.

The unmodified ppm.c and ppm.h files can be indluded.

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)
2. `Problem_set_6` in the TDT4200_h2015 group on `climb.idi.ntnu.no`. Use a web browser and make a user on the system. *Please* use the same username as your NTNU username. If not make a clear note of this in the delivery. After making a user join the group `TDT4200_h2015` before submitting solutions.

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

Part 1, Theory

Problem 1, General Theory

There are no theory questions.

Part 2, Code

There is only a single task in this assignment. This optimization task can take a *long* time to solve in a good way. Fulfilling the requirement is quite easy however. There are no hard requirements on performance, but it should probably be faster than the handout code.

Write short answers. Just point at the key idea/answer if possible. The staff does not have time to read long answers. *Sorry.* Quality over quantity.

Problem 1, SIMD Optimization

"Your so called optimized code is still too slow! We do not want to use more than a single core on our platform however. The other teams have asked to reserve them to their tasks. Just make a SIMD (?) version, like they did in this article I found: `http://www.hindawi.com/journals/vlsi/2012/413747/`."
– Your boss (the link is not helpful for this exercise)

Motivation: This task is real. It is not a toy, example or training task. Solving this in a *very* good way may have real commercial value. Note: a very good version is probably 10+ times faster than the PS3 winner.

You have to optimize the same code as in problem set 3 again using SIMD vectorization. We also hand out the 'CMB winner' code from problem set 3. Now the challenge is to improve already fast code. You can use this code as basis for your delivery. You may also use your original code from problem set 3 as basis, or make something new. If you do use the 'CMB winner' code a speedup is expected.

For this task use the files `Makefile`, `ppm.c` and `newImageIdea.c` from the `optimize_handout_ps6.zip` archive. You must also download the `flower.ppm` file and include it in the same directory as the code.

The `Makefile` handed out with this Problem set creates an executable named "`newImageIdea`". The program reads the `flower.ppm` image and creates 3 new images: `flower_tiny.ppm`, `flower_small.ppm` and `flower_medium.ppm`. Use the rule "`make run`" to run "`newImageIdea`" and create the images. Use the rule "`make check`" to create the correct images, and count the number of pixel errors your code produces.

The `newImageIdea.c` code implements a nice approach to solving the task in problem set 3. The code is *good*, but there are still room for improvements. Any parameter will enable file reading. You may modify the given `Makefile` if you think it helps your performance. Make sure that the existing "`make newImageIdea`" rule still works if you do.

a) Somehow optimize the code *with* vector (SIMD) instructions, *without* using MPI/OpenMP/OpenMP/MPI. Making use of vectorized code is *required*. A vector version might not be much faster without several changes however. This is your only program code delivery.

- If you use the 'CMB winner' code you should think about the following:
  - Data types (GCC vector type).
  - Data layout/usage.
  - Memory alignment (see 'Lecture 18, 15 Oct by Rune')
  - The conclusion of 'Lecture 18'.
- If you use your old code you should also think about the following:
  - Caches and access patterns. You may look in the book for hints.
  - Branches.
  - The optimization part of 'Lecture 18'.
- There are several ways to write SIMD code. You may select any of the following methods. Some are simpler than others however:
  - GCC vector extensions (recommended)
    `https://gcc.gnu.org/onlinedocs/gcc/Vector-Extensions.html`
  - Intel intrinsics (SSE/AVX)
  - NEON intrinsics
- However, There are (still) many different approaches to better performance, and several can be combined.
- You are again allowed to have some minor pixel errors in the final output in each image. Each pixel color is in the range of 0-255. We define pixel errors as pixels whose values differ from the images produced by the handout reference version of the code.
  - A few thousands pixels with ± 1 differences is fine.
  - A few hundred pixels with a larger difference is fine.
  - The included `checker.c` program will inform you of the number of bad pixels. Use the "`make check`" rule to run a check of how many pixel errors you have. The handout code have many errors already, but you may increase the number.

- The code must run on its-015-XX.idi.ntnu.no (XX being any of the lab machines in ITS015). You must report the time the program uses. If you use your own system report CPU specifications and the compiler. Just using the `time` command is acceptable.
- The code must run on the CMB website `climb.idi.ntnu.no`. It will be tested for energy efficiency, and must pass an automated test. Report the numbers `Time (s),Energy (j)` and `EDP (js)`.
- In your report answer these questions:
  i) State what version you used as basis for your hand-in.
  ii) Write a short list of what optimizations you have used. Details are not needed, unless you have done something very smart.
  iii) Write some comments on the difference between your problem set 3 code and the 'CMB winner' code.
  iv) What do you think is the current bottleneck in your vectorized code?

Finally, please make the code readable. Remove debug tests and unused code to make it shorter. Some comments can be good as well. If your best code is slow a better report is advised. Keep in mind that this is a very easy assignment compared to earlier assignments.

Additional details can be found in the recitation slides for this Problem set. The 'Lecture 18, 15 Oct by Rune' may contain some very useful information.