Aalto 2023

CP3b: fast solution with floats ★★

Please note that you can still submit, but as the course is already closed, your submissions will not be graded.

To get started with the development, **download the code templates**, unzip the file, edit cp.cc, and run `./grading test` or `./grading benchmark` to try it out — see the **instructions** for more details!

Upload your solution as a file here...

Please upload here the file cp.cc that contains your solution to task CP3b.

Choose File No file chosen

... or copy-paste your code here

Submit

Your submissions

Your submissions to CP3b will appear here; you can simply **reload** this page to see the latest updates.

What you will need to do in this task

Please read the **general instructions for this exercise** first. Here are the additional instructions specific to this task:

> Using all resources that you have in the CPU, solve the task as fast as possible. You are encouraged to exploit instruction-level parallelism, multithreading, and vector instructions whenever possible, and also to optimize the memory access pattern. In this task, you are permitted to use single-precision floating point numbers.

What I will try to do with your code

I will first run all kinds of tests to see that your code works correctly. You can try it out locally by running `./grading test`, but please note that your code has to compile and work correctly not only on your own computer but also on our machines.

If all is fine, I will run the benchmarks. You can try it out on your own computer by running `./grading benchmark`, but of course the precise running time on your own computer might be different from the performance on our grading hardware.

Benchmarks

Name	Operations	Parameters
benchmarks/1	1,004,000,000	nx = 1000, ny = 1000
the input contains 1000 × 1000 pixels, and the output should contain 1000 × 1000 pixels		
benchmarks/2a	16,016,000,000	nx = 1000, ny = 4000
the input contains 4000 × 1000 pixels, and the output should contain 4000 × 4000 pixels		
benchmarks/2b	16,016,000,000	nx = 1000, ny = 4000
the input contains 4000 × 1000 pixels, and the output should contain 4000 × 4000 pixels		
benchmarks/2c	15,991,989,003	nx = 999, ny = 3999
the input contains 3999 × 999 pixels, and the output should contain 3999 × 3999 pixels		
benchmarks/2d	16,040,029,005	nx = 1001, ny = 4001
the input contains 4001 × 1001 pixels, and the output should contain 4001 × 4001 pixels		
benchmarks/3	216,144,000,000	nx = 6000, ny = 6000
the input contains 6000 × 6000 pixels, and the output should contain 6000 × 6000 pixels		
benchmarks/4	729,324,000,000	nx = 9000, ny = 9000
the input contains 9000 × 9000 pixels, and the output should contain 9000 × 9000 pixels		

Here "operations" is our rough estimate of how many useful arithmetic operations you will at least need to perform in this benchmark, but of course this will depend on exactly what kind of an algorithm you are using.

Grading

In this task your submission will be graded using benchmarks/4: the input contains 9000 × 9000 pixels, and the output should contain 9000 × 9000 pixels.

The point thresholds are as follows. If you submit your solution no later than on Sunday, 14 May 2023, at 23:59:59 (Helsinki), your score will be:

Running time	Points
≤ 8.000 sec	1
≤ 4.500 sec	2
≤ 3.000 sec	3
≤ 2.200 sec	4
≤ 1.500 sec	5

If you submit your solution after the deadline, but before the course ends on Sunday, 04 June 2023, at 23:59:59 (Helsinki), your score will be:

Running time	Points
≤ 4.000 sec	1
≤ 2.500 sec	2
≤ 1.500 sec	3

Contest

Your submissions to this task will also automatically take part in the **contest**, and you can receive **up to 2 additional points** if your code is among the fastest solutions this year!

Running time	Extra points
≤ 1.20 × fastest	1
≤ 1.05 × fastest	2