263-2300-00: How To Write Fast Numerical Code
Assignment 5: 100 points
Due Date: Fr, April 24th, 17:00
http://www.inf.ethz.ch/personal/markusp/teaching/263-2300-ETH-spring15/course.html
Questions: fastcode@lists.inf.ethz.ch

Submission instructions (read carefully):

- (Submission)
  Homework is submitted through the Moodle system https://moodle-app2.let.ethz.ch/course/view.php?id=1317. Before submission, you must enroll in the Moodle course. Enrollment key is "263-2300".
- (Code)
  For all problems in this homework use the skeleton provided here. Submit **only** the file `src/comp.c` containing your solutions.
- (Late policy)
  You have 3 late days, but can use at most 2 on one homework, meaning submit latest 48 hours after the due time. Note that each homework will be available for submission on the Moodle system 2 days after the deadline. However, if the accumulated time of the previous homework submissions exceeds 3 days, the homework will not count.
- (Formats)
  If you use programs (such as MS-Word or Latex) to create your assignment, convert it to PDF and name it homework.pdf. When submitting more than one file, make sure you create a zip archive that contains all related files, and does not exceed 10 MB. Handwritten parts can be scanned and included or brought (in time) to Alen's or Daniele's office. Late homeworks have to be submitted electronically.
- (Plots)
  For plots/benchmarks, be concise, but provide necessary information (e.g., compiler and flags) and always briefly discuss the plot and draw conclusions. Follow (at least to a reasonable extent) the small guide to making plots (soon in lecture).
- (Neatness)
  5% of the points in a homework are given for neatness.

Exercises:

1. *Conjugate transpose (15 pts)*

   Consider the following 2×2 matrix M with complex entries:

   $$M = \begin{pmatrix} a_1 + ia_2 & b_1 + ib_2 \\ c_1 + ic_2 & d_1 + id_2 \end{pmatrix}.$$

   The conjugate transpose of M is the matrix

   $$M^* = \begin{pmatrix} a_1 - ia_2 & c_1 - ic_2 \\ b_1 - ib_2 & d_1 - id_2 \end{pmatrix}.$$

   Complete the function `simd_conjugate_transpose` provided in `src/comp.c` with the code to compute the conjugate transpose of M using SSE intrinsics (meaning you can use all instructions up to SSE4.2). Note that using the structure `complex_t` the complex numbers in M are stored in interleaved format, alternating real and imaginary parts: $[a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2]$.

2. *Pairs multiplication (30 pts)*

   Assume the following definition of multiplication between two pairs of real numbers:

   $$(a, b) \cdot (c, d) = (ac + 2bd, ad + bc), \ a, b, c, d \in \mathbb{R}\ .$$

   Complete the function `simd_pairs_multiplications` provided in `src/comp.c` used to multiply (with above definition) element-wise two arrays x, y each containing n (which is divisible by 4) pairs of real

numbers in single precision. The storage is again inteleaved, i.e., the arrays are the concatenation of the pairs. The array z is the result. Again you should use SSE (up to SSE4.2).

Hint: Structure your code after the following pattern:

`Loop { Load, Shuffle/Compute, Store }`

3. *Comparisons vectorization (50 pts)*

Assume a matrix $A(N \times N)$ initialized with floats in the interval $(0, 3]$. The function below rounds each entry in the matrix A to the next-largest integer (either 1, 2 or 3):

```
void sisd_ceil(float **A, int N){
  int i,j;
  for(i = 0; i < N; i++){
    for(j = 0; j < N; j++){
      if(A[i][j] <= 1){
        A[i][j] = 1;
      }
      else if(A[i][j] <= 2){
        A[i][j] = 2;
      }
      else{
        A[i][j] = 3;
      }
    }
  }
}
```

Write a vectorized version (`simd_ceil`) in `src/comp.c` using SSE4.2 intrinsics. In your code **do not** use any rounding functionalities, such as `_mm_ceil_ps`, `_mm_round_ps`, or others.