ETH zürich

M. Troyer, P. Koumoutsakos
ETH Zürich, HIT G 31.8
CH-8093 Zürich

High Performance Computing for Science and Engineering II

Spring semester 2015

Set 1 - SIMD

Issued: February 16, 2015
Hand in: February 23, 2015, 12:00pm

Question 1: SIMD compatibility

a) Find out which SSE/AVX version your machine and compiler support.[1] Which speedup do you expect from it?

Theoretically maximum achievable speedup is

	float	double
AVX	8x	4x
SSE	4x	2x

On a Mac you can check it by

```
sysctl -a | less
```

b) Find out which SSE/AVX version does the Euler cluster support.

AVX and SSE 4.2.
You can check the available flags on Euler by

```
less /proc/cpuinfo
```

Question 2: Parallel computation of squares of binomials

The provided skeleton code computes many squares of binomials.

```
for( int i = 0; i < N; ++i ) {
      z[i] = x[i]*x[i] + y[i]*y[i] + 2.*x[i]*y[i];
}
```

a) Vectorize the code using the manual SSE or AVX intrinsics and the report the speedup obtained compared to the serial version. Does it meet your expectations?

[1] `http://software.intel.com/sites/landingpage/IntrinsicsGuide/` for Intel processors

b) Now try to do the same using automatic vectorization and study the compiler's vectorization report to check that the loop is indeed vectorized. Comment on the performance relative to your manual version. **Hint:** With `gcc` the reduction is only vectorized with the `-funsafe-math-optimizations` option.[2]

As we do not have any branching or function calls in our loop, the compiler can vectorize the code automatically and performance is similar to our manual version

Timings on Euler:

```
sum_serial 1.98367s
sum_gcc 0.477534s
sum_sse 0.327582s
sum_avx 0.338854s
```

Summary

Summarize your answers, results and plots into a PDF document. Furthermore, elucidate the main structure of the code and report possible code details that are relevant in terms of accuracy or performance. Send the PDF document and source code to your assigned teaching assistant.

[2] `-funsafe-math-optimizations` enables more aggressive optimizations by allowing the compiler to violate some details of IEEE and ISO standards. This can cause problems with codes relying on, e.g., correct treatment of special values like infinities and NaN. Reference: `http://gcc.gnu.org/onlinedocs/gcc-4.8.2/gcc/Optimize-Options.html#index-funsafe-math-optimizations-906`