

## Serie 5

**Exercise 1** Download the C code in the file `loop0.c`. Complete this code to achieve, on the given loop, the following operations: "prefetch", "unroll" of level 1, "prefetch+unroll of level 1", unroll of level 2 (4 instructions) and unroll of level 4 (8 instructions). Compile with the Intel compiler (`/opt/intel/Compiler/11.1/072/bin/ia32/icc`) and the options `icc -O0...`, then with the Gnu compiler `gcc -O0 ...`. Compare the CPU times.

**For the standard loop only**, notice the effect of the options `-O1`, `-O2`, `-O3`, `-unrolln` ( $n$ : depth), of the Intel compiler. Do the same for the options `-O1`, `-O2`, `-O3`, `-unroll_all_loops` of the Gnu compiler. What option (or combination of options) yields the fastest code for both compilers?

Finally, use the automatic vectorization of the loop: for the Intel compiler `icc -march=core2 -O3 ...` and for the gnu compiler `gcc -march=core2 -O3 -ftree-vectorizer-verbose=2`. Compare with the CPU times without vectorization.

**Exercise 2** Complete the file `loop1.c` by coding a matrix-vector multiplication (order  $n = 10000$ ) by columns, then by rows and compare the vectorization and CPU times obtained by compiling with: `gcc -march=core2 -O3 -ftree-vectorizer-verbose=2 ...`, `icc -march=core2`, `icc -O3 -march=core2`.

**Exercise 3** We want to code the matrix-vector multiplication of Exercise 2 as a loop of dot products using the function `ddot` of the BLAS library, and then the function `dgemv`. Complete the file `MultAx.c` as follows: 1) code the double loop, 2) replace the internal loop by a dot product for each component of  $Ax$ , 3) use the function `dgemv`. The prototypes of the functions `ddot` et `dgemv` are included in the file. Recall that, if one has `/usr/share/man` in the environment variable `MANPATH`, one can type `man ddot` to get the documentation of this function. The compilation and linking can be done with the help of the `Makefile`. Discuss the results of the three cases: 1) Compile with `gcc` and use the standard Blas, 2) Compile with `gcc` and use the optimized Goto Blas, 3) Compile with `icc` and use the Mkl library. In the latter case, initialize two environment variables as follows: `setenv LD_LIBRARY_PATH /opt/intel/Compiler/11.1/072/mkl/lib/32`; `setenv OMP_NUM_THREADS 1`.

**Exercise 4 (Amdahl's law)** Plot the speedup in function of the number of processors (1-200) for the Amdahl's law with  $f_p = 0.9999, 0.9990, 0.9900, 0.9000$ .