



Vectorization for Intel® C++ & Fortran Compiler OpenMP* 4.0 Extensions (Linux*)

Lab

Disclaimer

The information contained in this document is provided for informational purposes only and represents the current view of Intel Corporation ("Intel") and its contributors ("Contributors") on, as of the date of publication. Intel and the Contributors make no commitment to update the information contained in this document, and Intel reserves the right to make changes at any time, without notice.

DISCLAIMER. THIS DOCUMENT, IS PROVIDED "AS IS." NEITHER INTEL, NOR THE CONTRIBUTORS MAKE ANY REPRESENTATIONS OF ANY KIND WITH RESPECT TO PRODUCTS REFERENCED HEREIN, WHETHER SUCH PRODUCTS ARE THOSE OF INTEL, THE CONTRIBUTORS, OR THIRD PARTIES. INTEL, AND ITS CONTRIBUTORS EXPRESSLY DISCLAIM ANY AND ALL WARRANTIES, IMPLIED OR EXPRESS, INCLUDING WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, NON-INFRINGEMENT, AND ANY WARRANTY ARISING OUT OF THE INFORMATION CONTAINED HEREIN, INCLUDING WITHOUT LIMITATION, ANY PRODUCTS, SPECIFICATIONS, OR OTHER MATERIALS REFERENCED HEREIN. INTEL, AND ITS CONTRIBUTORS DO NOT WARRANT THAT THIS DOCUMENT IS FREE FROM ERRORS, OR THAT ANY PRODUCTS OR OTHER TECHNOLOGY DEVELOPED IN CONFORMANCE WITH THIS DOCUMENT WILL PERFORM IN THE INTENDED MANNER, OR WILL BE FREE FROM INFRINGEMENT OF THIRD PARTY PROPRIETARY RIGHTS, AND INTEL, AND ITS CONTRIBUTORS DISCLAIM ALL LIABILITY THEREFOR. INTEL, AND ITS CONTRIBUTORS DO NOT WARRANT THAT ANY PRODUCT REFERENCED HEREIN OR ANY PRODUCT OR TECHNOLOGY DEVELOPED IN RELIANCE UPON THIS DOCUMENT, IN WHOLE OR IN PART, WILL BE SUFFICIENT, ACCURATE, RELIABLE, COMPLETE, FREE FROM DEFECTS OR SAFE FOR ITS INTENDED PURPOSE, AND HEREBY DISCLAIM ALL LIABILITIES THEREFOR. ANY PERSON MAKING, USING OR SELLING SUCH PRODUCT OR TECHNOLOGY DOES SO AT HIS OR HER OWN RISK.

Licenses may be required. Intel, its contributors and others may have patents or pending patent applications, trademarks, copyrights or other intellectual proprietary rights covering subject matter contained or described in this document. No license, express, implied, by estoppels or otherwise, to any intellectual property rights of Intel or any other party is granted herein. It is your responsibility to seek licenses for such intellectual property rights from Intel and others where appropriate. Limited License Grant. Intel hereby grants you a limited copyright license to copy this document for your use and internal distribution only. You may not distribute this document externally, in whole or in part, to any other person or entity. LIMITED LIABILITY. IN NO EVENT SHALL INTEL, OR ITS CONTRIBUTORS HAVE ANY LIABILITY TO YOU OR TO ANY OTHER THIRD PARTY, FOR ANY LOST PROFITS, LOST DATA, LOSS OF USE OR COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF YOUR USE OF THIS DOCUMENT OR RELIANCE UPON THE INFORMATION CONTAINED HEREIN, UNDER ANY CAUSE OF ACTION OR THEORY OF LIABILITY, AND IRRESPECTIVE OF WHETHER INTEL, OR ANY CONTRIBUTOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES. THESE LIMITATIONS SHALL APPLY NOTWITHSTANDING THE FAILURE OF THE ESSENTIAL PURPOSE OF ANY LIMITED REMEDY.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2[®], SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

See: <http://software.intel.com/en-us/articles/optimization-notice/>

Intel and Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2015, Intel Corporation. All Rights Reserved.

C/C++

In addition to yesterday:

1.6 SIMD-Enabled Functions (OpenMP* 4.0)

With OpenMP* 4.0, SIMD-enabled functions have been added to the standard. They differ slightly in terms of terminology and syntax to the Intel version used above. However, they require a compiler option `-openmp` or `-openmp-simd` to be specified in addition. The latter is used here because it does not introduce an OpenMP* runtime like the former one and just enables the SIMD-enabled functions.

1. Take the original version from activity 1.1 and change the code in `multiply.c` to use OpenMP* SIMD-enabled functions. Also change `driver.c` to create a parallel section with one thread for doing the matrix vector multiplication.

```
$ icc -openmp-simd -O2 $SIMD multiply.c driver.c -o matvector
$ ./matvector
```

Record execution time _____.

Hint: Transform inner loop; don't forget to take the non-unit stride into account (not necessary though)

Solution: `solutions/openmp4_simd`

2. Now re-apply the steps from activity 1.1 to 1.3 to the current version. What is the best combination?

Record execution time _____.

Note:

For the example we use throughout this activity there should be almost no difference compared to the auto-vectorized version. OpenMP* SIMD-enabled functions become more powerful for more complex scenarios because they limit the C/C++ side-effects the compiler has to deal with.

Hint: Apply the same steps as in previous activities. Not all might be needed, though. `#pragma ivedp` won't work here because it cannot be combined with SIMD-enabled functions.

Solution: `solutions/openmp4_simd_best`

Activity 3 – Pragma SIMD

In addition to yesterday:

Pragma SIMD with OpenMP* 4.0

The same steps from the activity above can be repeated for using the OpenMP* 4.0 equivalent. The semantics are exactly the same. The only difference is the syntax compared to the Intel version shown before. For the OpenMP* 4.0 variant, please use the following solutions instead:

3. Solution: `solutions/openmp4_simd`
4. Solution: `solutions/openmp4_reduction`
6. Solution: `solutions/openmp4_linear`
7. Solution: `solutions/openmp4_safelength`

Note:

The examples need to be compiled with compiler option `-openmp` or `-openmp-simd` set. Otherwise the pragma will be ignored. The latter should be used here because it does not introduce an OpenMP* runtime like the former one and just enables the SIMD-enabled functions.

Fortran

In addition to yesterday:

1.6 SIMD-Enabled Functions (OpenMP* 4.0)

With OpenMP* 4.0, SIMD-enabled functions have been added to the standard. They differ slightly in terms of terminology and syntax to the Intel version used above. However, they require a compiler option `-openmp` or `-openmp-simd` to be specified in addition. The latter is used here because it does not introduce an OpenMP* runtime like the former one and just enables the SIMD-enabled functions.

1. Take the original version from activity 1.1 and change the code in `multiply.f90` to use OpenMP* SIMD-enabled functions. Also change `driver.f90` to create a parallel section with one thread for doing the matrix vector multiplication.

```
$ ifort -openmp-simd -fpp -O2 $SIMD driver.f90 multiply.f90 -o matvector
$ ./matvector
```

Record execution time _____.

Hint: Transform inner loop; don't forget to take the non-unit stride into account (not necessary though)

Solution: `solutions/openmp4_simd`

2. Now re-apply the steps from activity 1.1 to 1.3 to the current version. What is the best combination?

Record execution time _____.

Note:

For the example we use throughout this activity there should be almost no difference compared to the auto-vectorized version. OpenMP* SIMD-enabled functions might become more powerful for complex kernels.

Hint: Apply the same steps as in previous activities. Not all might be needed, though.

Solution: `solutions/openmp4_simd_best`

Activity 3 – Directive SIMD

In addition to yesterday:

Directive SIMD with OpenMP* 4.0

The same steps from the activity above can be repeated for using the OpenMP* 4.0 equivalent. The semantics are exactly the same. The only difference is the syntax compared to the Intel version shown before. For the OpenMP* 4.0 variant, please use the following solutions instead:

3. Solution: `solutions/openmp4_simd`
4. Solution: `solutions/openmp4_reduction`
6. Solution: `solutions/openmp4_linear`
7. Solution: `solutions/openmp4_safelength`

Note:

The examples need to be compiled with compiler option `-openmp` or `-openmp-simd` set. Otherwise the directive will be ignored. The latter should be used here because it does not introduce an OpenMP* runtime like the former one and just enables the SIMD-enabled functions.