Quick Reference Guide to Optimization with Intel® C++ and Fortran Compilers v19.1

For IA-32 processors, Intel® 64 processors, Intel® Xeon Phi™ processors and compatible non-Intel processors.

Contents

For product and purchase information, visit the Intel® Software Development Tools site at: http://intel.ly/sw-dev-tools.

§ Intel® Xeon Phi™ processors are supported by compilers within Intel® Parallel Studio XE, but not within Intel® System Studio.

** Several of these options are available for both Intel® and non-Intel microprocessors but they may perform more optimizations for Intel microprocessors than they perform for non-Intel microprocessors.‡

Application Performance

A Step-by-Step Approach to Application Tuning with Intel Compilers

Before you begin performance tuning, check correctness of your application by building it without optimization using **/Od (-O0)**.

1. Measure performance using the general optimization options (Windows* **/O1, /O2 or /O3**; Linux* and macOS* **-O1, -O2, or -O3**) to see which one works best for your application. Most users should start at **/O2 (-O2)**, the default, before trying more advanced optimizations. Next, try **/O3 (-O3)** for loop-intensive applications.**

2. Fine-tune performance using processor-specific options such as **/Qx (-x)** or **/arch (-m)**. Examples are **/QxCORE-AVX512 (-xcore-avx512)** for the Intel® Xeon® Scalable processor family and **/arch:SSE3 (-msse3)** for compatible, non-Intel processors that support at least the Intel® SSE3 instruction set. Or **/QxHOST (-xhost)** will use the most advanced instruction set for the processor on which you compiled.**

3. Add interprocedural optimization (IPO), **/Qipo (-ipo)** and/or profile-guided optimization (PGO), **/Qprof-gen** and **/Qprof-use (-prof-gen** and **-prof-use)**, then measure performance again to determine whether your application benefits from one or both of them.

4. Use Intel® Advisor and Intel® VTune™ Amplifier†† to help you identify serial and parallel performance "hotspots" within your application that could benefit from further performance tuning. Use the compiler optimization report **/Qopt-report (-qopt-report)** to help identify individual optimization opportunities.

5. Optimize for parallel execution on multi-threaded, multi-core and multi-processor systems using: the auto-parallelization option **/Qparallel (-parallel)**; OpenMP* pragmas or directives along with the option **/Qopenmp (-qopenmp)**; or by using the Intel® Performance Libraries included with the product.** Use Intel® Inspector to reduce the time to market for threaded applications by diagnosing memory and threading errors and speeding up the development process.

6. Further optimize your application for SIMD through explicit vector programming using the SIMD features of OpenMP. The OpenMP directives you add are automatically recognized with -O2 and higher.

7. On the Intel® Xeon Phi™ x200 processor family, consider adding data prefetching based on the results of step 4**.**

For more details, please consult the main product documentation at https://software.intel.com/intel-software-technical-documentation .

** Several of these options are available for both Intel® and non-Intel microprocessors, but they may perform more optimizations for Intel microprocessors than they perform for non-Intel microprocessors.‡

†† Some features of this product cannot be used on non-Intel microprocessors.

General Optimization Options and Reports **

Windows*	Linux* & macOS*	Comment
/Od	**-O0**	**No optimization**. Used during the early stages of application development and debugging.
/Os **/O1**	**-Os** **-O1**	**Optimize for size**. Omits optimizations that tend to increase object size. Creates the smallest optimized code in most cases. May be useful in large server/database applications where memory paging due to larger code size is an issue.
/O2	**-O2**	**Maximize speed**. Default setting. Enables many optimizations, including vectorization and intra-file interprocedural optimizations. Creates faster code than **/O1 (-O1)** in most cases.
/O3	**-O3**	Enables **/O2 (-O2)** optimizations plus more aggressive loop and memory-access optimizations, such as scalar replacement, loop unrolling, loop blocking to allow more efficient use of cache and additional data prefetching. The **/O3 (-O3)** option is particularly recommended for applications that have loops that do many floating-point calculations or process large data sets. These aggressive optimizations may occasionally slow down other types of applications compared to **/O2 (-O2)**.
/Qopt-report [:*n*]	**-qopt-report [=*n*]**	Generates an optimization report, by default written to a file with extension *.optrpt*. ***n*** specifies the level of detail, from ***0*** (no report) to ***5*** (maximum detail). Default is ***2***.
/Qopt-report-file:*name*	**-qopt-report-file=*name***	Writes an optimization report to ***stderr***, ***stdout*** or to the file ***name***.
/Qopt-report-phase:*name1*, *name2*, ...	**-qopt-report-phase=*name1*, *name2*,...**	Optimization reports are generated for optimization phases ***name1*, *name2*,** *etc.* Possible phases include: ***all*** – Optimization reports for all phases (default) ***loop*** – Loop nest and memory optimizations ***vec*** – auto-vectorization and explicit vector programming ***par*** – auto-parallelization ***openmp*** – threading using OpenMP ***cg*** – code generation ***ipo*** – Interprocedural Optimization, including inlining ***pgo*** – Profile Guided Optimization
/Qopt-report-routine:*substring*	**-qopt-report-routine=*substring***	Generates reports only for functions or subroutines whose names contain ***substring***. By default, reports are generated for all functions and subroutines.
/Qopt-report-filter:*"string"*	**-qopt-report-filter=*"string"***	Restricts reports to the file, function or subroutine and/or ranges of line numbers specified by ***"string"***, e.g. "myfile,myfun,1-10".
/Qopt-report-annotate[:*fmt*]	**-qopt-report-annotate[=*fmt*]**	Annotates source listing with optimization information (default off). Possible values of ***fmt*** are ***text*** (default) or ***html***.

** Several of these options are available for both Intel® and non-Intel microprocessors but they may perform more optimizations for Intel microprocessors than they perform for non-Intel microprocessors.‡

Parallel Performance **

Windows*	Linux* macOS*	Comment
/Qopenmp	**-qopenmp**	Multi-threaded code and/or SIMD code is generated when OpenMP* directives are present. For Fortran only, makes local arrays automatic and may require an increased stack size.
/Qopenmp-simd	**-qopenmp-simd**	SIMD code is generated when OpenMP SIMD directives are present. Default: **on at -O2 and higher**.
/Qopenmp-stubs	**-qopenmp-stubs**	Ignores OpenMP directives and links references to OpenMP run-time library functions to stub (dummy) functions assuming single-threaded operation.
/Qparallel	**-parallel**	The auto-parallelizer detects simply structured loops that may be safely executed in parallel, including the DO CONCURRENT construct, and automatically generates multi-threaded code for these loops.
/Qopt-matmul[-]	**-q[no-]opt-matmul**	This option enables [disables] identification of matrix multiplication loop nests and replaces them with a compiler-generated matmul library call for improved performance. This option is enabled by default if options **/O3 (-O3)** and **/Qparallel (-parallel)** are specified. It has no effect unless option **/O2 (-O2)** or higher is set.
/Qcoarray [:*kywd*]	**-coarray [=*kywd*]**	Enables the coarray feature of Fortran 2008 (Fortran only). ***kywd*** options are ***shared***, ***distributed***, ***coprocessor*** and ***single***. See the compiler reference guide for more detail.
/Qcoarray-num-images:*n*	**-coarray-num-images=*n***	***n*** specifies the number of images that run a coarray executable. **Off** by default (number of images determined at run-time). (Fortran only)
/Qcoarray-config-file:*filename*	**-coarray-config-file=*filename***	***filename*** specifies an MPI configuration file (may include a path). Default is **off**, MPI default settings are used. (Fortran only)
/Qmkl:*name*	**-mkl=*name***	Links to the Intel® Math Kernel Library (Intel® MKL). Off by default. Possible values of ***name*** are: ***parallel*** Links the threaded part of Intel MKL (default) ***sequential*** Links the non-threaded part of Intel MKL ***cluster*** Links cluster and sequential parts of Intel MKL (cluster-specific libraries are not available for macOS*)
/Qtbb	**-tbb**	Links to Intel® Threading Building Blocks (Intel® TBB). (C++ only)
/Qdaal[:*lib*]	**-daal[=*lib*]**	Links to the Intel® Data Analytics Acceleration Library (Intel® DAAL). ***parallel*** is the default value of ***lib***, links to the threaded library. ***sequential*** links to a non-threaded library version. (C++ only)

** Several of these options are available for both Intel® and non-Intel microprocessors but they may perform more optimizations for Intel microprocessors than they perform for non-Intel microprocessors.‡

Recommended Processor-Specific Optimization Options **

Windows*	Linux* macOS*	Comment
/Qx*target*	**-x***target*	Generates specialized code for any Intel® processor that supports the instruction set specified by *target*. The executable will not run on non-Intel processors or on Intel processors that support only lower instruction sets. For possible values of *target* see Frequently Used Processor Names in Compiler **Note:** This option enables additional optimizations that are not enabled by the /arch or -m options. On 64 bit macOS, options **SSE3** and **SSE2** are not supported.
/arch:*target*	**-m***target*	Generates specialized code for any Intel processor or compatible, non-Intel processor that supports the instruction set specified by *target*. Running the executable on an Intel processor or compatible, non-Intel processor that does not support the specified instruction set may result in a run-time error. See the table Frequently Used Processor Names in Compiler Options for possible values of *target* . **Note:** Specifying a target value of **ia32** generates non-specialized, generic x86/x87 code. It is supported for IA-32 architecture targets only. It is not supported on macOS*.
/QxHOST	**-xhost**	Generates instruction sets up to the highest that is supported by the compilation host. On Intel processors, this corresponds to the most suitable **/Qx (-x)** option; on compatible, non-Intel processors, this corresponds to the most suitable of the **/arch (-m)** options.
/Qax*target*	**-ax***target*	May generate specialized code for any Intel processor that supports the instruction set specified by *target*, while also generating a default code path. See the table Frequently Used Processor Names in Compiler Options for possible values of *target* . Multiple values, separated by commas, may be used to tune for additional Intel processors in the same executable, e.g. **/QaxAVX,SSE4.2**. The default code path will run on any Intel or compatible, non-Intel processor that supports at least **SSE2**, but may be modified by using in addition a **/Qx** (**-x**) or **/arch** (**-m**) switch. For example, to generate a specialized code path optimized for the 4th generation Intel® Core™ processor family and a default code path optimized for Intel processors or compatible, non-Intel processors that support at least **SSE3**, use **/QaxCORE-AVX2 /arch:SSE3 (-axcore-avx2 -msse3** on Linux*). At runtime, the application automatically detects whether it is running on an Intel processor, and if so, selects the most appropriate code path. If an Intel processor is not detected, the default code path is selected.
/Qvecabi: ***cmdtarget***	**-vecabi=** ***cmdtarget***	Compiler creates vector variants of SIMD functions for targets specified by the **/Qx** or **/Qax (-x** or **-ax**) switches above.
/Qopt-zmm-usage:*high*	**-qopt-zmm-usage=*high***	Enables more aggressive generation of 512 bit SIMD instructions when used with **/QxCORE-AVX512** (Windows*) or **-xcore-avx512** (Linux* or macOS*).

Please see the online article Intel Compiler Options for Intel SSE and Intel AVX Generation and Processor-Specific Optimizations to view the latest recommendations for processor-specific optimization options.

The Intel® Compiler User and Reference Guides are available at Intel C++ Compiler 19.1 Developer Guide and Reference and Intel Fortran Compiler 19.1 Developer Guide and Reference.

** Several of these options are available for both Intel® and non-Intel microprocessors but they may perform more optimizations for Intel microprocessors than they perform for non-Intel microprocessors.‡

Optimizing for the Intel® Xeon Phi™ x200 product family

Windows*	Linux*	Comment
/Qopt-threads-per-core:*n*	**-qopt-threads-per-core=*n***	Hint to the compiler to optimize for ***n*** threads per physical core, where ***n=1***, ***2***, ***3*** or ***4***.
/Qopt-prefetch:*n*	**-qopt-prefetch=*n***	Enables increasing levels of software prefetching for ***n=0*** to ***5***.
/Qopt-prefetch-distance=*n1[,n2]*	**-qopt-prefetch-distance=*n1[,n2]***	Specifies how many vectorized loop iterations ahead to prefetch data. ***n1*** is for L2, ***n2*** (***≤n1***) is for L1 cache. Default is off (compiler chooses).
/Qimf-domain-exclusion:*n*	**-fimf-domain-exclusion=*n***	Specifies special case arguments for which math functions need not conform to IEEE standard. The bits of n correspond to the domains: ***0*** – extreme values (e.g. very large; very small; close to singularities); ***1*** – NaNs; ***2*** – infinities; ***3*** – denormals; ***4*** – zeros.
/align: array64byte	**-align array64byte**	Seek to align the start of arrays at a memory address that is divisible by 64, to enable aligned loads and help vectorization. (Fortran only)
/Qopt-assume-safe-padding	**-qopt-assume-safe-padding**	Asserts that the compiler may safely access up to 64 bytes beyond the end of array or dynamically allocated objects as accessed by the user program. User is responsible for padding. Off by default.

For more optimization detail, see https://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-code-named-knights-landing-application-readiness ; https://software.intel.com/articles/advanced-optimizations-for-intel-mic-architecture; and the Intel® Compiler User and Reference Guides at Intel C++ Compiler 19.1 Developer Guide and Reference and Intel Fortran Compiler 19.1 Developer Guide and Reference.

§ Intel Xeon Phi processors are supported by compilers within Intel® Parallel Studio XE, but not within Intel® System Studio

Interprocedural Optimization (IPO) and Profile-Guided Optimization (PGO) Options

Windows*	Linux* macOS*	Comment
/Qip	**-ip**	Single file interprocedural optimizations, including selective inlining, within the current source file.
/Qipo[*n*]	**-ipo[*n*]**	Permits inlining and other interprocedural optimizations among multiple source files. The optional argument ***n*** controls the number of object files created. Default for ***n*** is ***0*** (the compiler chooses). Caution: This option can in some cases significantly increase compile time and code size.
/Qipo-jobs[*n*]	**-ipo-jobs[*n*]**	Specifies ***n***, the number of commands (jobs) to be executed simultaneously during the link phase of Interprocedural Optimization (IPO). The default is ***1*** job.
/Ob2	**-finline-functions** or **-inline-level=2**	This option enables function inlining within the current source file at the compiler's discretion. This option is enabled by default at **/O2** and **/O3** (**-O2** and **-O3**). Caution: For large files, this option may sometimes significantly increase compile time and code size. It can be disabled by **/Ob0** (**-fno-inline-functions** on Linux* and macOS*)
/Qinline-factor:*n*	**-inline-factor=*n***	This option scales the total and maximum sizes of functions that can be inlined. The default value of ***n*** is 100, i.e., 100% or a scale factor of one.
/Qprof-gen [*:kywd*]	**-prof-gen [=*kywd*]**	Instruments a program for profile generation. ***kywd= threadsafe*** allows profile generation for threaded applications. ***kywd=srcpos*** and ***globdata*** collect additional data useful for function and data ordering.
/Qprof-use	**-prof-use**	Enables the use of profiling information during optimization.
/Qprof-dir *dir*	**-prof-dir *dir***	Specifies a directory for profiling output files, *.dyn and *.dpi. (Alternatively, the directory may be specified by the environment variable **PROF_DIR**).
/Qprof-gen-sampling	**-prof-gen-sampling**	Generates additional debug information for use in hardware event based profile generation using Intel® VTune™ Amplifier
/Qprof-use-sampling:*file*	**-prof-use-sampling=*file***	Enables the use of hardware event based profiling information from ***file*** during optimization

Fine-Tuning (All Processors) **

Windows*	Linux* & macOS*	Comment
/Qunroll[*n*]	**-unroll[*n*]**	Sets the maximum number of times to unroll loops. ***n=0*** disables loop unrolling. Default is **/Qunroll (-unroll)**, which uses default heuristics.
/Qopt-prefetch:*n*	**-qopt-prefetch=*n***	Enables increasing levels of software prefetching from ***n=0*** (default, off) to ***n=5*** (aggressive prefetching). Warning: excessive prefetching may result in resource conflicts that degrade performance.
/Qopt-prefetch-issue-excl-hint	**-qopt-prefetch-issue-excl-hint**	Enables generation of exclusive prefetches (in anticipation of a write) on processors that support the prefetchw instruction.
/Qopt-prefetch-distance:n1[,n2]	**-qopt-prefetch-distance=n1[,n2]**	Specifies how many vectorized loop iterations ahead to prefetch data. n1 is for L2, n2 (≤n1) is for L1 cache. Default is **off** (compiler chooses).
/Qopt-block-factor:*n*	**-qopt-block-factor=*n***	Specifies a preferred loop blocking factor ***n***, the number of loop iterations in a block, overriding default heuristics. Loop blocking, enabled at **/O3** (**-O3**), is designed to increase the reuse of data in cache.
/Qopt-streaming-stores:*mode*	**-qopt-streaming-stores *mode***	**always** Encourages generation of streaming stores that bypass cache, assuming application is memory bound with little data reuse **never** Disables generation of streaming stores **auto** Uses default compiler heuristics
/Qrestrict[-]	**-[no]restrict**	Enables [disables] pointer disambiguation with the **restrict** keyword. Off by default. (C/C++ only)
/Oa	**-fno-alias**	May assume no aliasing in the program. Off by default.
/Ow	**-fno-fnalias**	May assume no aliasing within functions. Off by default.
/Qalias-args[-]	**-fargument-[no]alias**	Implies function arguments may be aliased [are not aliased]. On by default. (C/C++ only). **-fargument-noalias** often helps the compiler to vectorize loops in C or C++ involving function array arguments.
/Qansi-alias[-]	**-[no-]ansi-alias**	Enables [disables] ANSI and ISO C Standard aliasability rules and those in the Fortran standard. Defaults: disabled on Windows*; enabled on Linux* and macOS*.
/Qopt-class-analysis[-]	**-q[no-]opt-class-analysis**	C++ class hierarchy information is used to analyze and resolve C++ virtual function calls at compile time. If a C++ application contains non-standard C++ constructs, such as pointer downcasting, it may result in different behavior. Off by default, but enabled with **/Qipo** (Windows*) or **-ipo** (Linux* and macOS*). (C++ only)
/Qvec-threshold:0	**-vec-threshold=0**	Asks the compiler to auto-vectorize loops even if it does not expect a performance benefit.
/Qvec[-]	**-[no-]vec**	Enables [disables] auto-vectorization. On by default at **/O2** (**-O2**)

Windows*	Linux* & macOS*	Comment
/Qstringop-strategy :*alg*	**-mstringop-strategy =*alg***	Sets algorithm ***alg*** to use for buffer manipulation functions such as *memcpy* and *memset.* ***libcall***: tells compiler to emit a library call; ***rep***: compiler inlines using ***rep movs*** or similar sequences; ***const_size_loop*** (default): expands to an inline loop if size is known at compile time and less than [**/Q** | **-m**]**stringop-inline-threshold**
/align: array*nn*byte	**-align array*nn*byte**	Seek to align the start of arrays at a memory address that is divisible by ***nn***, to facilitate aligned loads and help vectorization. (Fortran only)
/assume:[no] buffered_io	**-assume [no] buffered_io**	Accumulates data for successive sequential reads or writes into a buffer for more efficient I/O. Default is unbuffered. (Fortran only)
/assume:contiguous_ assumed_shape	**-assume contiguous_ assumed_shape**	Asserts that assumed shape dummy arguments will always be contiguous (have unit stride). (Fortran only)
/assume: contiguous_pointer	**-assume contiguous_pointer**	Asserts that pointer dummy arguments will always be contiguous (have unit stride). (Fortran only)
none	**-f[no-] exceptions**	For C++, **-fexceptions** is default and enables exception handling table generation. This may sometimes impede vectorization. **-fno-exceptions** causes exception specifications to be parsed, but ignored. Any use of try blocks and throw statements will produce an error if any function in the call chain has been compiled with -fno-exceptions. (Linux* only)
/Qfnsplit:*n*	**-fnsplit=*n***	Conditional code blocks with < ***n***% probability of being reached may be placed in a different code segment. (Linux* and Windows* only)
/Qimf-domain-exclusion:n	**-fimf-domain-exclusion=n**	Specifies special case arguments for which math functions need not conform to IEEE standard. The bits of n correspond to the domains: 0 – extreme values (e.g. very large; very small; close to singularities); 1 – NaNs; 2 – infinities; 3 – denormals; 4 – zeros.
/align: array64byte	**-align array64byte**	Seek to align the start of arrays at a memory address that is divisible by 64, to enable aligned loads and help vectorization. (Fortran only)
/Qopt-assume-safe-padding	**-qopt-assume-safe-padding**	Asserts that the compiler may safely access up to 64 bytes beyond the end of array or dynamically allocated objects as accessed by the user program. User is responsible for padding. **Off** by default.

** Several of these options are available for both Intel® and non-Intel microprocessors, but they may perform more optimizations for Intel microprocessors than they perform for non-Intel microprocessors.‡

Floating-Point Arithmetic Options

Windows*	Linux* & macOS*	Comment
/fp:*name*	**-fp-model *name***	Controls tradeoffs between performance, accuracy and reproducibility of floating-point results at a high level. Possible values of ***name***: ***fast[=1|=2]*** – Allows more aggressive optimizations at a slight cost in accuracy or reproducibility. (default ***fast=1***) ***consistent*** – Enables consistent, reproducible results between different optimization levels or between different processors of the same architecture. ***precise*** – Disallows compiler optimizations that might produce slight variations in floating point results, except for generation of fused multiply-add (FMA) instructions. ***except*** – Enforces floating point exception semantics. ***strict*** – Enables both the ***precise*** and ***except*** options and does not assume the default floating-point environment. Suppresses generation of fused multiply-add (FMA) instructions by the compiler.
/Qopt-dynamic-align[-]	**-q[no-]opt-dynamic-align**	Allows [disables] certain optimizations that depend on data alignment at run-time, and that could cause small variations in floating-point results when the same, serial application is run repeatedly on the same input data. On by default unless **/fp:precise** or **/fp:consistent** (**-fp-model precise** or **-fp-model consistent**) is set.
/Qftz[-]	**-[no-]ftz**	When the main program or dll main is compiled with this option, denormals (resulting from Intel® SSE or Intel® AVX instructions) at run time are flushed to zero for the whole program (dll). Default is **on** except at **/Od (-O0)**.
/Qimf-precision: *name*	**-fimf-precision: *name***	Sets the accuracy for math library functions. Default is OFF (compiler uses default heuristics). Possible values of ***name*** are ***high***, ***medium*** and ***low***. Reduced precision may lead to increased performance and vice versa, particularly for vectorized code.
/Qimf-arch-consistency: *true*	**-fimf-arch-consistency= *true***	Ensures that math library functions produce consistent results across different Intel or compatible, non-Intel processors of the same architecture. May decrease run-time performance. The default is "***false***" (off) unless **/fp:consistent** (**-fp-model consistent**) is set.
/Qprec-div[-]	**-[no-]prec-div**	Improves [reduces] precision of floating point divides. This may slightly degrade [improve] performance. Default is **OFF**.
/Qprec-sqrt[-]	**-[no-]prec-sqrt**	Improves [reduces] precision of square root computations. This may slightly degrade [improve] performance.
/Qprotect-parens[-]	**-f[no-]protect-parens**	Expressions are evaluated in the order specified by parentheses Default is off unless **/fp:precise** or **/fp:consistent** (**-fp-model precise** or **-fp-model consistent**) is set.
/Qfma[-]	**-[no]fma**	Suppresses generation of fused multiply-add (FMA) instructions by the compiler (may still be present in run-time libraries).

/Qimf-use-svml	**-fimf-use-svml**	The compiler uses the Short Vector Math Library (SVML) rather than the Intel® Math Library (LIBM) to implement scalar math functions.
/Qimf-force-dynamic-target	**-fimf-force-dynamic-target**	Code path through math library functions is selected at run-time based on processor type. Default **OFF**.
/Qfp-speculation safe	**-fp-speculation safe**	Compiler disables certain optimizations if there is a risk that these might cause a floating-point exception. Useful to set when floating-point exceptions are unmasked for debugging.

See also http://software.intel.com/articles/consistency-of-floating-point-results-using-the-intel-compiler

Processor Code Name With Instruction Set Extension Name Synonym

https://en.wikipedia.org/wiki/List_of_Intel_CPU_microarchitectures

Intel Microarchitecture Code Name	Microarchitecture Instruction Set
ICELAKE-SERVER	No synonym
ICELAKE-CLIENT	No synonym
CANNONLAKE	No synonym
SKYLAKE-AVX512	CORE-AVX512
KNM	MIC-AVX512
KNL	MIC-AVX512
SKYLAKE	CORE-AVX2
BROADWELL	CORE-AVX2
HASWELL	CORE-AVX2
SILVERMONT	SSE4.2
IVYBRIDGE	CORE-AVX-I
SANDYBRIDGE	AVX

Frequently Used Processor Names in Compiler Options

Intel Microarchitecture Code Name or Instruction Set	-x*code* /Qx*code*	-ax*code* /Qax*code*	-arch *code* /arch:*code*	-march=*code* *code* must be lower case	-mtune=*code* /tune:*code* *code* must be lower case
ICELAKE-SERVER	✓	✓	✓	✓	✓
ICELAKE-CLIENT	✓	✓	✓	✓	✓
CANNONLAKE	✓	✓	✓	✓	✓
SKYLAKE-AVX512	✓	✓	✓	✓	✓
KNM †	✓	✓	✓	✓	✓
KNL †	✓	✓	✓	✓	✓
SKYLAKE	✓	✓	✓	✓	✓
BROADWELL	✓	✓	✓	✓	✓
HASWELL	✓	✓	✓	✓	✓
SILVERMONT †	✓	✓	✓	✓	✓
IVYBRIDGE	✓	✓	✓	✓	✓
SANDYBRIDGE	✓	✓	✓	✓	✓

COMMON-AVX512 Δ	✓	✓			
CORE-AVX512	✓	✓			
MIC-AVX512	✓	✓			
CORE-AVX2	✓	✓	✓	✓	✓
SSE4.2	✓	✓			
CORE-AVX-I	✓	✓	✓	✓	✓
AVX	✓	✓	✓		
corei7-avx				✓	✓
ATOM_SSE4.2	✓	✓			
SSE4.1	✓	✓	✓		
ATOM_SSSE3	✓	✓			
corei7					✓
atom				✓	✓
SSSE3	✓	✓	✓		
SSE3	✓	✓	✓		
SSE2	✓	✓	✓		
SSE			✓		
IA32			✓		
core2			✓		✓
pentiummx					✓
pentiumpro					✓
pentium-m			✓		
pentium4			✓		
pentium3			✓		
pentium			✓		

† Windows* and Linux* only
Δ A subset of MIC-AVX512 and CORE-AVX512

Intel® C++ Compiler Based on the Modern LLVM* Framework

The Intel C++ Compiler Based on the Modern LLVM* Framework, often referred to as ICC NextGen, is invoked with the compiler option -qnextgen. This option is only available for Windows* or Linux* icc/icl/icpc. This option and functionality are not available for icc on macOS.

Not all compiler options referenced in this Quick Reference Guide are available when compiling with -qnextgen. A porting guide and additional usage information are available.

For product and purchase information, visit the Intel® Software Development Tools site at: http://intel.ly/sw-dev-tools.

‡ 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