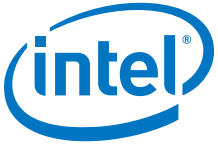


インテル® コンパイラー v11
最適化クイック・リファレンス・ガイド

IA-32 プロセッサー、インテル® 64 プロセッサー、
IA-64[1] プロセッサー

インテル® ソフトウェア開発製品

アプリケーション・パフォーマンス

インテル® コンパイラーでアプリケーションをチューニングする手順

パフォーマンス・チューニングを開始する前に、**/Od** (**-O0**) を使用して最適化を行わずにアプリケーションをビルドし、正常に動作することを確認してください。本バージョンのコンパイラーでは、すべての最適化レベルは、デフォルトで SSE2 命令セットのサポートを想定しています。SSE2 命令がサポートされないインテル® Pentium® III プロセッサーなど、以前の IA-32 プロセッサーで実行するには、**/arch:IA32** (Windows*) または **-mia32** (Linux*) を追加する必要があります。

1. 一般的な最適化オプション (Windows では **/O1**、**/O2**、**/O3**。Linux および Mac OS* では **-O1**、**-O2**、**-O3**) を使用してパフォーマンスを測定し、アプリケーションにとって最適なオプションを判断します。通常は、最初に **/O2** (**-O2**) (デフォルト) 試してから、より高度な最適化を行うと効果的です。次に、ループを多用するアプリケーションに対しては **/O3** (**-O3**) を試します。これは、IA-64[1] ベースのシステムでは特に効果的です。

2. ターゲットシステムが、インテル® Core™ i7 プロセッサーなどのインテル® Core™ プロセッサー・ファミリーを搭載する IA-32 アーキテクチャー・ベース・システムおよびインテル® 64 アーキテクチャー・ベース・システムの場合は、**/QxSSE4.2** (**-xsse4.2**) プロセッサー専用のオプションにより、最新の命令セットを利用したきめ細かな最適化が可能です。また、**/QxHOST** (**-xhost**) を指定し、コンパイルするホストマシンで利用可能な最上位の命令セットを使用して最適化することもできます。特定のプロセッサー向けに推奨するオプションの全リストは、「IA-32 プロセッサーとインテル® 64 プロセッサー向けに推奨するプロセッサー専用の最適化オプション」の表を参照してください。

3. インテル® VTune™ パフォーマンス・アナライザーを使用して、パフォーマンスの「hotspot」を識別すると最も効果的です。アプリケーション・コードでさらにチューニングの恩恵が得られる部分を特定することができます。また、インテル® コンパイラーの最適化レポートは、改善する余地がある箇所を確認するのに役立ちます。

4. プロシージャー間の最適化 (IPO) を行う **/Qipo** (**-ipo**) やプロファイルに基づく最適化 (PGO) を行う **/Qprof-gen** および **/Qprof-use** (**-prof-gen** と **-prof-use**) を追加して、パフォーマンスを再び測定し、これらの最適化がアプリケーションにとって効果的かどうかを確認します。

5. 並列パフォーマンス・オプション **/Qparallel** (**-parallel**)、**/Qopenmp** (**-openmp**)、または製品に含まれているインテル® パフォーマンス・ライブラリーを使用して、マルチスレッディング・システム、マルチコアシステム、またはマルチプロセッサー・システム向けにアプリケーションを最適化します。

6. インテル® スレッド・プロファイラーは、スレッド・アプリケーションの構造を理解し、そのパフォーマンスを最大限に引き出すのに役立ちます。インテル® スレッド・チェッカーは、スレッド化エラーの診断を行い、開発プロセスを効率化して、マルチスレッド・アプリケーションの開発期間を短縮します。両ツールとも、バイナリー・インストルメンテーションにより動作します。インテル® コンパイラーでソース・コード・インストルメンテーションを行うと、より詳細なソースコード情報を入手できます。

詳細は、コンパイラー・ドキュメントおよび『インテル® C++ & Fortran コンパイラーによるアプリケーションの最適化』(英語) ホワイトペーパーを参照してください。

[1] IA-64 = インテル® Itanium® プロセッサー

一般的な最適化オプション

Windows	Linux Mac OS X	説明
/Od	-O0	**最適化は行われません。**このオプションは、アプリケーション開発の初期段階およびデバッグ時に使用します。アプリケーションが正常に動作することを確認した後は、より高度なオプションを使用してください。
/O1	-O1	**サイズの最適化を行います。**オブジェクトのサイズを増やす傾向がある最適化を省略します。多くの場合、最小限のサイズで最適化されたコードが作成されます。 コードサイズが大きいために、メモリーページングが問題になっている巨大なサーバー / データベース・アプリケーションにおいて、このオプションは効果的です。
/O2	-O2	**最速化します（デフォルト設定）。**ベクトル化を含む多くの最適化を有効にします。多くの場合、**/O1** (**-O1**) よりも速いコードを作成します。
/O3	-O3	**/O2** (**-O2**) の最適化に加えて、スカラー置換、ループアンロール、分岐を除去するコード反復、より効率的にキャッシュを使用するループ・ブロッキング、さらに IA-64 ベースのシステムにはデータ·プリフェッチ機能など、強力なループの最適化およびメモリーアクセスの最適化を行います。 **/O3** (**-O3**) オプションは、特に浮動小数点演算を多用するループや大きなデータセットを処理するループを含むアプリケーションに推奨します。これらの強力な最適化は、場合によっては **/O2** (**-O2**) の最適化よりもアプリケーションの実行が遅くなることがあります。
/Zi	-g	一般的な開発環境のデバッガーで使用できるデバッグ情報を生成します。このオプションは、**/O2** (**-O2**) （または別の **O** オプション）が指定されない限り、**/O2** (**-O2**) をオフにして **/Od** (**-O0**) をデフォルトにします。
/debug:full	-debug full	最適化されていないコードのシンボリック・デバッグに必要なローカル・シンボル・テーブル情報と、リンクに必要なグローバル・シンボル・テーブル情報を含むすべてのデバッグ情報を生成します。最大サイズのオブジェクト・モジュールを生成します。このオプションをデバッグ対象の C ライブラリー・ルーチンを呼び出すアプリケーションに対して指定する場合は、適切な C デバッグ・ライブラリーにリンクされるように、**/dbglibs** も指定する必要があります。 このオプションが最適化コードに使用されると、完全なシンボル情報が生成されます。これには、最適化のレベルにかかわらずローカル・シンボル・テーブル情報も含まれます。このオプションは、パフォーマンスがわずかに低下することがあります。

並列パフォーマンス

Windows	Linux Mac OS X	説明
/Qopenmp	-openmp	OpenMP* 宣言子がある場合、マルチスレッド・コードを生成する処理をパラレライザーに指示します。スタックのサイズを増やさなければならないことがあります。
/Qopenmp-report {0|1|2}	-openmp-report {0|1|2}	OpenMP パラレライザーの診断レベルを制御します。デフォルトのレベル 1 は、正常に並列化されたループ、領域、およびセクションをレポートします。
/Qparallel	-parallel	安全に並列実行可能な単純構造のループを検出し、そのループに対するマルチスレッド・コードを自動的に生成します。
/Qpar-report {0|1|2|3}	-par-report {0|1|2|3}	自動パラレライザーの診断レベルを次のように制御します。 **0** – 診断情報を表示しません。 **1** – 正常に並列化されたループを示します (デフォルト)。 **2** – 並列化されなかったループに関する情報が追加されます。 **3** –自動並列化の妨げになると判断されたか、または想定された依存関係についての情報 (並列化されない理由) が追加されます。
/Qpar-threshold[n]	-par-threshold[n]	ループ並列化による効果の可能性に基づいて、ループの自動並列化のしきい値を設定します (**n=0** から **n=100**。デフォルト : **n=100**)。 **0** – 計算量にかかわらず並列化を行います。 **100** – ループは並列実行が有効であることが確実な場合にのみ並列化されます。 **/Qparallel** (**-parallel**) とともに使用する必要があります。
/Qpar-schedule-*keyword*[:n]	-par-schedule-*keyword*[=n]	parallel for (DO) ループのスケジューリング・アルゴリズムを指定します。**n** はチャンクサイズです (連続するループ反復内)。keyword の値は次のとおりです。 **static** – 事前に定められたチャンクを各スレッドに順に割り当てます。 **dynamic** – 固定チャンクをランタイム時にスレッドへ動的に割り当てます。 **guided** – **n** の最小チャンクサイズで切り下げたサイズの変数チャンクにループを動的に分割します。 **runtime** – OMP_SCHEDULE 環境変数でランタイム時にスケジューリング・アルゴリズムとチャンクサイズが指定されます。
/Qopt-mem-bandwidth<n> (IA-64 のみ)	-opt-mem-bandwidth<n> (IA-64 のみ)	並列アプリケーションにより大きなメモリー帯域幅を必要とする特定の最適化を制限します。 **/Qopt-mem-bandwidth0** (**-opt-mem-bandwidth0**) - 制限なし (シリアルコンパイルの場合のデフォルト) **/Qopt-mem-bandwidth1** (**-opt-mem-bandwidth1**) – OpenMP 並列領域のループの最適化を制限します (**/Qparallel** (**-parallel**) または **/Qopenmp** (**-openmp**) のデフォルト)。 **/Qopt-mem-bandwidth2** (**-opt-mem-bandwidth2**) - すべてのループの最適化を制限します。MPI や他の並列アプリケーションで役立ちます。

IA-32 アーキテクチャーとインテル® 64 アーキテクチャー向けに推奨するプロセッサー専用の最適化オプション

Windows	Linux Mac OS X	説明
/Qx {SSE4.2| SSE4.1| SSE3_ATOM| SSSE3| SSE3|SSE2| HOST}	-x {sse4.2|sse4.1| sse3_atom| ssse3|sse3| sse2|host}	対象とするプロセッサーを指定します。指定したインテル® プロセッサー向けに専用のコードを生成します。指定したプロセッサーまたはそれ以降のプロセッサー上でのみ実行可能です。† **SSE4.2** – SSE4 高効率および高速な文字列処理を含む、インテル® プロセッサー向けの SSE4 命令、SSSE3 命令、SSE3 命令、SSE2 命令、SSE 命令を生成します。インテル® Core™ プロセッサー・ファミリー向けに最適化します。例 : インテル® Core™ i7 プロセッサー **SSE4.1** – インテル® プロセッサー向けの SSE4 ベクトル化コンパイラー命令およびメディア・アクセラレーター命令、SSSE3、SSE3、SSE2、および SSE 命令を生成します。インテル® 45nm Hi-k 世代インテル® Core™ マイクロアーキテクチャー・プロセッサー向けに最適化します。 **SSE3_ATOM** – インテル® プロセッサー向けの MOVBE 命令を生成します。インテル® Atom™ プロセッサーおよびインテル® Centrino® Atom™ プロセッサー・テクノロジー向けに最適化します。 **SSSE3** – インテル® プロセッサー向けの SSSE3 命令、SSE3 命令、SSE2 命令、SSE 命令を生成します。インテル® Core™ 2 プロセッサー・ファミリー向けに最適化します。 **SSE3** – インテル® プロセッサー向けの SSE3 命令、SSE2 命令、SSE 命令を最適化し、生成します。**/arch:SSE3** (**-msse3**) で有効にならない最適化を実行します。 **SSE2** – インテル® プロセッサー向けの SSE2 命令、SSE 命令を最適化し、生成します。**/arch:SSE2** (**-msse2**) で有効にならない最適化を実行します。 **HOST** – コンパイルホストでサポートされている命令を生成し、最適化します。これは、インテル® プロセッサー上では、そのホストで利用可能な最高の **/Qx** (**-x**) オプションに対応します。インテル以外のプロセッサー上では、**/arch** (**-m**) オプションに対応します。 **注** : Mac OS X では、**sse3** および **sse2** オプションはサポートされていません。
/arch: {SSE3| SSE2|IA32}	-m {sse3|sse2| ia32} (Linux のみ)	指定した命令セットを使用する最適化コードを生成します。指定した命令セットをサポートするプロセッサー上でのみ実行可能です。† **SSE3** – SSE3 命令、SSE2 命令、SSE 命令を生成します。コードは、SSE3 をサポートするインテル® プロセッサーやインテル製以外のプロセッサーでも実行可能です。 **SSE2** – SSE2 命令、SSE 命令を生成します。コードは、SSE2 をサポートするインテル® プロセッサーやインテル製以外のプロセッサーでも実行可能です。(デフォルト) **IA32** – インテル® Pentium® プロセッサーまたは以降のプロセッサー、あるいは互換性のあるインテル以外のプロセッサー上で実行する拡張命令セットなしのコードを生成します [IA-32 アーキテクチャーのみ]。
/Qax {SSE4.2| SSE4.1|SSSE3| SSE3|SSE2}	-ax {sse4.2| sse4.1|ssse3| sse3|sse2}	自動プロセッサー・ディスパッチを行います。対象となるインテル® プロセッサー向けに専用のコードを生成し、またデフォルトのコードも生成します。カンマで区切って複数の値を指定し、同一ファイルで追加のプロセッサー向けのチューニングを行うことができます。例 : **/QaxSSE4.1,SSE3**。デフォルトのコードは、**/Qx** (**-x**) または **/arch** (**-m**) オプションを追加して変更することができます。† 例えば、インテル® 45nm Hi-k 世代 インテル® Core™ マイクロアーキテクチャー・プロセッサー上で最高のパフォーマンスと、SSE3 のみをサポートする AMD プロセッサー上で優れたパフォーマンスを得るには、**/QaxSSE4.1 /arch:SSE3** (Linux では **-axsse4.1 -msse3**) を指定します。 これにより、自動プロセッサー・ディスパッチ機能を使用した 2 つのコードパスを含むバイナリーが生成されます。1 つのコードパスは、インテル® 45nm Hi-k 世代インテル® Core™ マイクロアーキテクチャー・プロセッサーの機能を最大限に活用します。もう 1 つのコードパスは、SSE3 をサポートする (ただし、SSE4 はサポートしない) インテルおよびインテル以外のプロセッサー上で実行できます。アプリケーションは、実行環境のインテル® プロセッサーをランタイムで自動判断し、専用またはデフォルトのいずれかのコードパスを選択します。 **注** : Mac OS X では、**sse3** および **sse2** オプションはサポートされていません。

† /arch (-m) オプション値 SSE3、SSE2、および IA32 は、対象となるインテル® プロセッサーと同じ機能を実装したインテル以外のプロセッサーでも動作するバイナリーを生成します。対応する /Qx (-x) オプション値は、/arch (-m) で有効にならない追加の最適化を実行しますが、インテル® プロセッサーでしか実行できません。

オンライン記事 「Intel® compiler options for SSE generation and processor-specific optimizations (SSE 生成とプロセッサー固有の最適化インテル® コンパイラー・オプション)」で、プロセッサー専用の推奨最適化オプションを参照してください。

本ガイドの内容

一般的な最適化オプション

パフォーマンス・チューニングを開始する前に、**/Od** (**-O0**) を使用して最適化を行わずにアプリケーションをビルドし、正常に動作することを確認してください。そして、**/O1**、**/O2**、または **/O3**（**-O1**、**-O2**、または **-O3**）を使用してパフォーマンス・チューニングを開始します。これらのオプションは、32 ビット版と 64 ビット版のすべてのインテル® プロセッサーを対象としたアプリケーションのチューニングで中心となる一般的な最適化オプションです。より高度な最適化を行う前に、異なるオプションを使用してパフォーマンスを比較することを推奨します。

並列パフォーマンス

マルチスレッディング、マルチコア、マルチプロセッサー搭載システムを対象とする場合、インテル® コンパイラーは、**/Qparallel** (**-parallel**) または **/Qopenmp** (**-openmp**) の 2 つの手法により、マルチスレッド・アプリケーションの開発をサポートします。

IA-32 アーキテクチャーとインテル® 64 アーキテクチャー向けに推奨するプロセッサー専用の最適化オプション

/Qx（Linux または Mac OS X では **-x**）オプションは、対応するインテル® プロセッサー、または最新のインテル® プロセッサーで最適なパフォーマンスを得るために推奨される最適化オプションです。例えば、45nm Hi-k 世代インテル® Core™ マイクロアーキテクチャー・プロセッサー向けには、**/QxSSE4.1** (**-xsse4.1**) を使用します。**/arch**（Linux または Mac OS X では **-m**）オプションは、対応する命令セットをサポートするすべてのプロセッサー（AMD* プロセッサーを含む）で最適なパフォーマンスを得られる推奨オプションです。（例 : SSE3 をサポートするプロセッサー向けの **/arch:sse3** (**-msse3**)）。**/Qax** (-ax) オプションは、対応するインテル® プロセッサー向けに最適化されたバイナリーを生成します。このバイナリーには、AMD プロセッサーを含む SSE2 をサポートするその他のシステム向けに最適化されたデフォルトコードも含まれています。このデフォルトコードは **/Qx** (**-x**) または **/arch** (**-m**) オプションで変更することができます。例えば、インテル® Pentium® III プロセッサーなど、以前の IA-32 プロセッサー向けのデフォルトコードのバイナリーを生成する場合は、**/arch:IA32** (Windows) または **-mia32** (Linux) を追加します。

表に示すオプションを使用して、サポートする命令を使い、特定のインテル® プロセッサー向けにパフォーマンスをチューニングすることができます。これまでの手順で各オプションがパフォーマンスに与える影響を検証し、最適なオプションを使用してください。また、インテル® コンパイラーの最適化レポートを使用することで、コンパイラーによって依存関係とみなされる問題や、SSE 命令によるループの並列実行を妨げているその他の問題を解決することができます。

プロシージャー間の最適化（IPO）オプションとプロファイルに基づく最適化（PGO）オプション

IPO は、関数のインライン化を制御することで、関数呼び出しで発生するオーバーヘッドを減少させ、最適化の機会を増やします。PGO は、ランタイム・フィードバックを提供することで、データとコードレイアウトに関する最適化手法を判断し、命令キャッシュ、ページングおよび分岐予測を向上させます。IPO を使用するとコードサイズが増加することがあります。これらのオプションを使用する際は、実行時のパフォーマンス、コンパイル時間、コードサイズを測定して評価を行ってください。インライン化すべき関数を判断できるため、IPO は PGO とともに使用すると効果的です。

浮動小数点演算オプション

インテル® コンパイラーには、すべてのインテル® アーキテクチャー上で浮動小数点演算結果の一貫性または精度を高めるオプションが用意されています（パフォーマンスに多少の影響があります）。浮動小数点オプションの詳細な情報については、『インテル® C++ コンパイラー・ドキュメント』と『インテル® Fortran コンパイラー・ドキュメント』の「コンパイラー・オプション」を参照してください。

きめ細かなチューニング（すべてのプロセッサー）

パフォーマンスの「hotspot」を識別した後、特定の関数の細部をチューニングするために、より詳細な情報をコンパイラーに渡します。最適化レポートおよびベクトル化レポートには、ポインター・エイリアシングやメモリーアクセスのオーバーラップなどが原因で、ループを十分に最適化できなかった箇所が表示されます。ソフトウェアのパイプライン化、ループアンロール、ベクトル化、プリフェッチを制御するその他のプラグマ、宣言子、組み込み関数を使用することで、アプリケーションのコードにおけるより細かなチューニングが可能です。詳細は、『インテル® C++ コンパイラー・ドキュメント』および『インテル® Fortran コンパイラー・ドキュメント』を参照してください。

プロシージャー間の最適化 (IPO) オプションとプロファイルに基づく最適化 (PGO) オプション

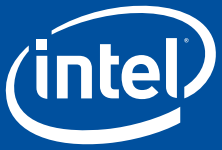
Windows	Linux Mac OS X	説明
/Qip	-ip	単一ファイルの最適化を行います。現在のソースファイルを対象にしたインライン化を含むプロシージャー間の最適化です。
/Qipo[value]	-ipo[value]	インライン化およびその他のプロシージャー間の最適化が複数のソースファイル間で行われます。オプションの **value** 引数には、コンパイル時に生成するオブジェクト・ファイルの最大数を指定します。デフォルトの **value** は 0 です (コンパイラーが選択)。 **警告** : 条件によってはコンパイル時間とコードサイズが大幅に増加する場合があります。
/Qipo-jobs[n]	-ipo-jobs[n]	プロシージャー間の最適化 (IPO) のリンクフェーズで、同時に実行するコマンド (ジョブ) の数を指定します。デフォルトは 1 ジョブです。
/Ob2	-finline-functions -finline-level=2	コンパイラーの判断に従って現在のソースファイルを対象にした関数のインライン化を有効にします。このオプションは、**/O2** および **/O3** (**-O2** および **-O3**) を指定すると有効になります。 **警告** : 大きなファイルでは、コンパイル時間とコードサイズが大幅に増加する場合があります。**/Ob0** (Linux および Mac OS X では **-fno-inline-functions**) を指定すると無効になります。
/Qinline-factor=n	-finline-factor=n	インライン化される関数の合計サイズと最大サイズを指定します。 n のデフォルト値は 100 (100%、スケール係数 1) です。
/Qprof-gen	-prof-gen	プロファイル生成用にプログラムをインストルメントします。
/Qprof-use	-prof-use	最適化中にプロファイリング情報を使用します。
/Qprof-dir *dir*	-prof-dir *dir*	プロファイル出力ファイル *.dyn および *.dpi を格納するディレクトリーを指定します。

浮動小数点演算の最適化オプション

Windows	Linux Mac OS X	説明
/fp:name	-fp-model name	**/Op** (**-mp**) オプションや **/Qprec** (**-mp1**) オプションよりも、最適化を制限して浮動小数点結果の一貫性を制御するこのオプションを推奨します。**name** の 値は次のとおりです。 **fast=[1|2]** – 精度や一貫性を多少低くすることにより、さらに強力な最適化が可能になります。(デフォルトは、**fast=1**) **precise** – 浮動小数点コードでは精度に影響しない最適化のみ有効にします。 **double/extended/source** – 中間結果を倍精度、拡張精度、ソースの精度まで丸めます。上書きされない限り、**precise** も適用されます。 インテル® Fortran では、**double** オプションおよび **extended** オプションは利用できません。 **except** – 浮動小数点例外セマンティクスを使用します。 **strict** – **precise** オプションと **except** オプションを有効にし、fma 縮約を無効にする最も厳密な演算モードです。 **推奨**: 浮動小数点演算の一貫性や再現性が必要な多くの状況では、**/fp:precise /fp:source** (**-fp-model precise -fp-model source**) を推奨します。
/Qfp-speculation *mode*	-fp-speculation *mode*	次の mode で浮動小数点のスペキュレーションを有効にします。 **fast** – 浮動小数点演算のスペキュレーションを行います。(デフォルト) **off** – 浮動小数点演算のスペキュレーションを無効にします。 **safe** – 浮動小数点例外が発生する可能性がある場合はスペキュレーションを行いません。 **strict** – このモードは、off と同じです。
/Qftz[-]	-ftz[-]	メインプログラムまたはメインの DLL をこのオプションでコンパイルすると、プログラム (dll) 全体でランタイム時に生成されるデノーマル結果をゼロにフラッシュします。 IA-64 アーキテクチャー・ベース・システムでは、**/O3** (**-O3**) が指定された場合を除き、デフォルトでオフです。 IA-32 アーキテクチャー・ベース・システムおよびインテル® 64 アーキテクチャー・ベース・システムでは、**/Od** (**-O0**) が指定された場合を除き、デフォルトでオンです。ただし、SSE 命令からのデノーマル結果のみゼロにフラッシュします。
/Qfast-transcendentals[-]	-[no-]fast-transcendentals	sin や exp のような算術関数について、より高速で多少精度が低いバージョンを使用します。デフォルトは **/Qfast-transcendentals** (**-fast-transcendentals**) です。ただし、**/fp:precise** または **/fp:strict** (**-fp-model precise** あるいは **-fp-model strict**) が指定されると、**/Qfast-transcendentals-** (**-no-fast-transcendentals**) がデフォルトになります。
/Qfp-relaxed[-] (IA-64 のみ)	-[no-]fp-relaxed (IA-64 のみ)	divide や sqrt のような算術関数について、より高速で多少精度が低いコードシーケンスを有効 [無効] にします。デフォルトでは無効です。
/Qprec-div[-]	-[no-]prec-div	浮動小数点除算の精度を上げます。速度に多少影響します。
/Qprec-sqrt[-]	-[no-]prec-sqrt	平方根計算の精度を上げます。速度に多少影響します。

きめ細かなチューニング (すべてのプロセッサー)

Windows	Linux Mac OS X	説明
/Qunroll[n]	-unroll[n]	ループをアンロールする最大回数を設定します。**/Qunroll0 (-unroll0)** はループアンロールを無効にします。デフォルトは、**/Qunroll (-unroll)** で、デフォルトのヒューリスティックを使用します。
/Qopt-prefetch[-]	-[no-]opt-prefetch	プリフェッチ挿入を有効または無効にします。
/Qopt-block-factor:n	-opt-block-factor=n	1 ブロックのループの反復数であるループ・ブロッキング係数 n を指定します。デフォルトのヒューリスティックを上書きします。ループ・ブロッキングは **/O3 (-O3)** で有効になり、キャッシュ中のデータ再利用率を高めるように設計されています。
/Qopt-streaming-stores:*mode*	-opt-streaming-stores *mode*	ストリーミング・ストアを生成するかどうかを指定します。***mode*** の値は次のとおりです。 **always** データ再利用がほとんどなく、アプリケーションがメモリーにバインドされていることを仮定して、キャッシュを省略するストリーミング・ストアの生成をコンパイラーに指示します。 **never** ストリーミング・ストアの生成を無効にします。 **auto** ストリーミング・ストア生成にデフォルトのヒューリスティックを使用します。
/Qrestrict[-]	-[no]restrict	**restrict** キーワードとともに指定すると、ポインターの一義化が有効 [無効] になります。デフォルトではオフです。(C++ のみ)
/Oa	-fno-alias	プログラムでエイリアシングしないことを前提に処理します。デフォルトではオフです。
/Ow	-fno-fnalias	関数内でエイリアシングしないことを前提に処理します。デフォルトではオフです。
/Qalias-args[-]	-fargument-[no]alias	関数の引数のエイリアス化を有効 [無効] にします。デフォルトではオンです。(C++ のみ)
/Qopt-class-analysis[-]	-[no-]opt-class-analysis	C++ クラス階層情報を使用して、コンパイル時に C++ 仮想関数の呼び出しを解析し解決します。C++ アプリケーションに標準的ではない C++ 構造 (ポインターのダウンキャスティングなど) が含まれている場合、動作が異なることがあります。デフォルトではオフです。ただし、**/Qipo** (Windows) または **-ipo** (Linux および Mac OS X) コンパイラー・オプションが指定されるとデフォルトでオンになり、改善された C++ 最適化を有効にします。(C++ のみ)
	-f[no-]exceptions	**-f-exceptions**: C++ のデフォルトです。例外処理テーブルの生成を有効にします。 **-fno-exceptions**: C、Fortran のデフォルトです。コードサイズが小さくなります。C++ では、例外指定は解析されますが、無視されます。構造化例外処理 (try ブロックや throw 文) を使用していると、一連の呼び出し中の関数が **-fno-exceptions** でコンパイルされている場合、エラーが発生します。
/Qopt-report[:n]	-opt-report [n]	最適化レポートを作成し、stderr に送ります。n は、0 (レポートなし) から 3 (最大大限の情報) の範囲で詳細レベルを指定します。デフォルトは 2 です。
/Qopt-report-phase*name*	-opt-report-phase*name*	最適化レポートは ***name*** に基づいて生成されます。同じコンパイルで複数回、このオプションを指定し、複数のフェーズから出力結果を得られます。***name*** によく使用される引数は次のとおりです。 **all** – 全フェーズのすべての最適化レポート (デフォルト) **ipo_inl** – プロシージャー間の最適化のインライン展開レポート **hlo** – 高レベル・オプティマイザー・レポート (ループおよびメモリーの最適化を含む) **hpo** – ハイパフォーマンス最適化レポート (ベクトライザーおよびパラレライザーを含む) **ecg_swp** – コード・ジェネレーターのソフトウェアのパイプライン化コンポーネントに関するレポートのみ (IA-64 上の Windows システムおよび Linux システムのみ) **pgo** – プロファイルに基づく最適化レポート
/Qopt-report-help	-opt-report-help	上記の **/Qopt-report-phase (-opt-report-phase)** で利用可能なすべての設定を表示します。コンパイルは実行されません。
/Qopt-report-routine:*rtn*	-opt-report-routine *rtn*	rtn で指定された文字列が名前に含まれる関数またはサブルーチンに関してのみレポートを生成します。デフォルトでは、すべての関数およびサブルーチンのレポートが生成されます。
/Qvec-report [n]	-vec-report [n]	ベクトライザーの診断レベルを次のように制御します。 n = **0**: 診断情報なし。 n = **1**: ベクトル化ループを示します (デフォルト)。 n = **2**: ベクトル化および非ベクトル化ループを示します。 n = **3**: ベクトル化ループを示し、他のループがベクトル化されなかった理由を示します。



製品情報および購入情報は、インテル® ソフトウェア開発製品 Web サイトを参照してください。

www.intel.com/cd/software/products/ijkk/jpn/compilers/

JPN/0901/10K/XL/SSG/KS 254349-006JA