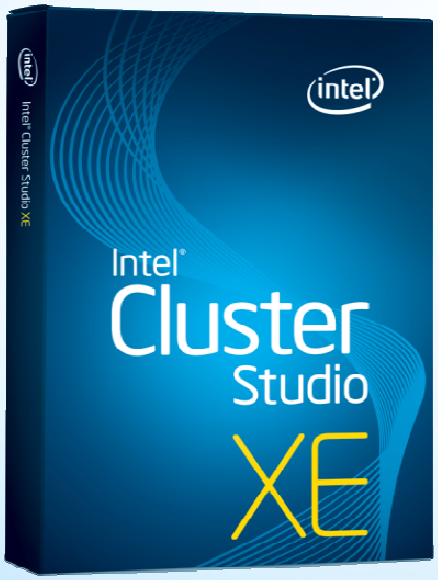
1. インテル® コンパイラー概要

Revision 1.1

エクセルソフト株式会社
黒澤 一平

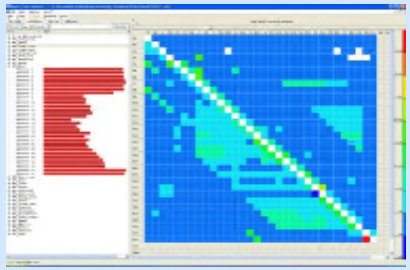
インテル® Cluster Studio XE 2012
クラスターシステム向けの総合開発ツール



Compiler
最適化、並列化のための C/C++、Fortran コンパイラー

Library
数値演算ライブラリー: インテル® MKL
最適化されたMPI: インテル® MPI
C++向け並列化ライブラリー: インテル® TBB
マルチメディアライブラリー: インテル® IPP

Analyzer
クラスター システム向け、シングルノード向けの性能解析ツール。パフォーマンス問題をグラフィカルに表示



ドキュメント凡例、用語

icc	インテル® C コンパイラー
icpc	インテル® C++ コンパイラー
ifort	インテル® Fortran コンパイラー
mpiicc	mpicc のインテル® C++ コンパイラー版
mpiicpc	mpic++, mpicxx のインテル® C++ コンパイラー版
mpiifort	mpif77,mpif90 のインテル® Fortran コンパイラー版
<Install dir>	各製品のインストールディレクトリー 例:<VTune Install dir> = /opt/intel/vtune_amplifier_xe/

インテル® Cluster Studio XE 2012 の構成詳細

フェーズ	製品名	機能	利点
ビルド	 インテル® MPI ライブラリー	ハイパフォーマンスな MPI ライブラリー	• ハイパフォーマンス、スケーラビリティー、インターコネクトの独立性、実行時のファブリック選択、アプリケーション・チューニングを実現
	 インテル® Composer XE	C/C++ 、Fortran コンパイラーとパフォーマンス・ライブラリー • インテル® スレッディング・ビルディング・ブロック • インテル® Cilk™ Plus • インテル® インテグレーテッド・パフォーマンス・プリミティブ • インテル® マス・カーネル・ライブラリー	• マルチコアと将来のメニーコアのパフォーマンスおよびスケーラビリティーを引き出すアプリケーションを開発するためのソリューション
検証	 インテル® Inspector XE	コードの品質を高める動的なメモリー/スレッド化解析とスタティック・セキュリティー解析	• 生産性とコードの品質を高め、コストを削減し、早期にメモリー/スレッド/セキュリティーの問題を発見 • 各クラスターノードで MPI に対応
検証 & チューニング	 インテル® トレース・アナライザー/コレクター	アプリケーションの正当性と動作を理解するための MPI パフォーマンス・プロファイラー	• MPI プログラムのパフォーマンスを解析し、並列アプリケーションの動作と通信パターンを視覚化して、hotspotを特定
チューニング	 インテル® VTune™ Amplifier XE	アプリケーションのパフォーマンスとスケーラビリティーを最適化するパフォーマンス・プロファイラー	• 従来の推測作業を排除し、短時間で容易にパフォーマンスとスケーラビリティーのボトルネックを特定 • 各クラスターノードで MPI に対応

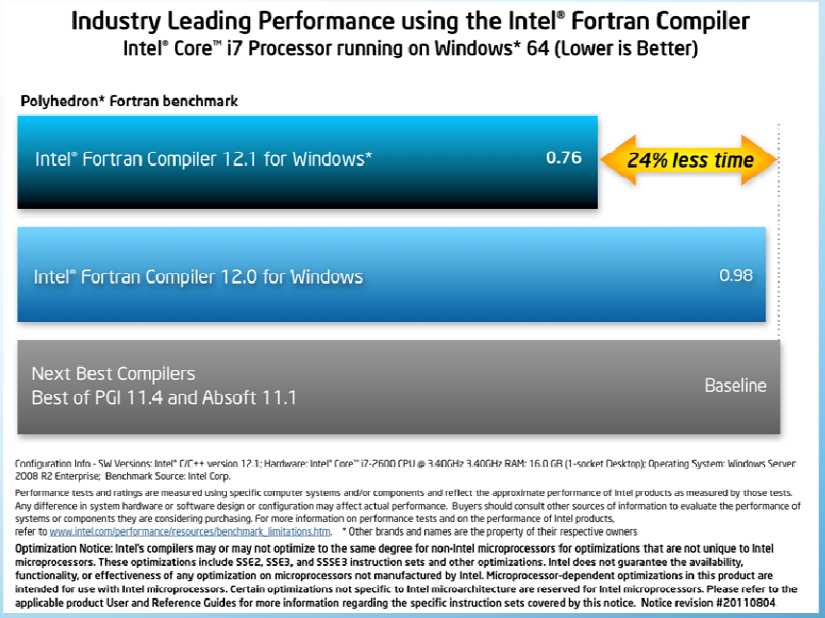
インテル® C++、Fortran コンパイラーの特長

- インテルおよび、インテル互換プロセッサーに対応した様々な最適化
  - ループの最適化
  - ベクトル化
  - プロシージャ間の最適化
  - プロファイルに基づく最適化 (PGO)
  - 自動並列化
  - ガイド付き自動並列化 (GAP)
  - プロファイル機能 (ループ、セキュリティ) など

- Sandy Bridge-EP に対応
- OpenMP 3.1 をサポート
- x86、x86_64 に対応
- GCC (Windows 版は Visual Studio) とのバイナリー互換
- C/C++言語と Fortran 言語の、混在言語開発が可能

インテル® Fortran コンパイラー 12.1 の仕様と新機能

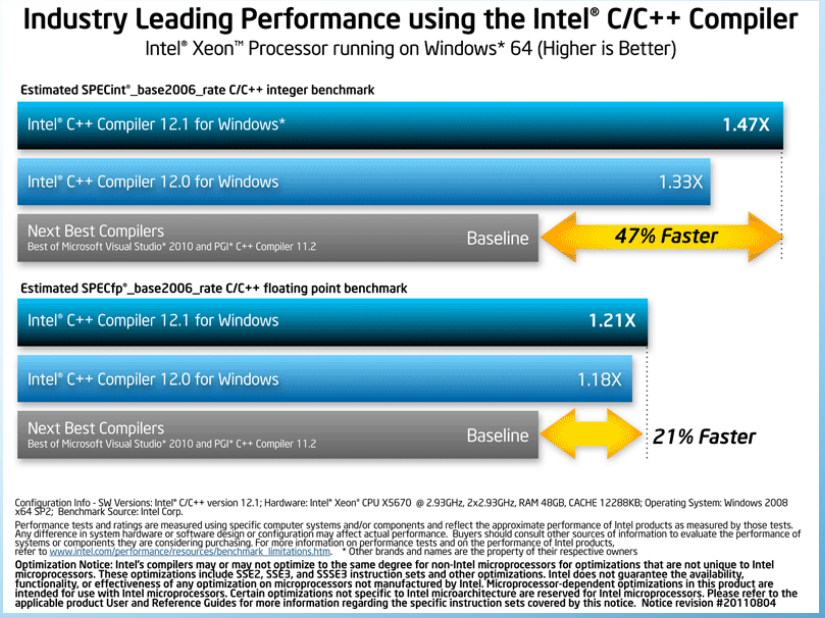
- パフォーマンスが大きく向上
- Co-array Fortran など、Fortran 2008 (ISO/IEC 1539-2:2004) の機能を実装
- Fortran IV/77/90/95/2003 をサポート
  (Fortran 2008は主要機能をサポート)
  - ISO 1539-1980、
    ISO/IEC 1539:1991、
    ISO/IEC 1539-1:1997、
    ISO/IEC1539-1:2004、
    ANSI X3.0-1966、
    ANSI X3.9-1978、
    ANSI X3.198-1992、
    ANSI X3J3/96-007 をフルサポート
  - ISO/IEC 1539-2:2004 の主要機能をサポート



インテル® C++ コンパイラー 12.1 の仕様と新機能

- パフォーマンスが大きく向上
- ラムダ関数など、C++11 (ISO/IEC 14882:2011) の機能を実装
- 浮動小数点数演算標準である IEEE 754-2008 をサポート
- インテル® の新しい並列化手法 (インテル Cilk Plus) に対応
- C/C++、C99をサポート※
  (C++11は主要機能をサポート)
  - ISO/IEC 9899:1990、
    ISO/IEC 9899:1999※、
    ISO/IEC 14882:1998) をフルサポート
  - ISO/IEC 14882:2011の主要機能をサポート

※128bit 表記の long double はサポートされません



インテルソフトウエアを使用するための環境変数を設定

スクリプトを実行し環境変数をする

- x86_64 インテル® コンパイラー、インテル®ライブラリー

  source <composer Install dir>/bin/compilervars.sh intel64

  または

  source <icc Install dir>/bin/iccvars.sh intel64

  source <ifort Install dir>/bin/ifortvars.sh intel64

- x86_64インテル® MPI

  source <Intel MPI Install dir>/bin64/mpivars.sh

- インテル® VTune Amplifier XE

  source <VTune Install dir>/amplxe-vars.sh

- インテル® Trace Analyzer/Collector

  source <TA/TC Install dir>/bin/itacvars.sh

・composer は icc と ifort の両方の環境設定が行なわれる
・Cシェルの場合には .sh の代わりに .csh を使用

XLSOFT

コンパイル コマンドの記述方法

- **シングルスレッド、マルチスレッド用**

```
C:       icc   [オプション] <ファイル>
C++:     icpc  [オプション] <ファイル>
Fortran: ifort [オプション] <ファイル>
```

- **MPI(シングルスレッド、マルチスレッド)用**

```
C:       mpiicc   [オプション] <ファイル>
C++:     mpiicpc  [オプション] <ファイル>
Fortran: mpiifort [オプション] <ファイル>
```

※ <ファイル> はソースファイルおよびオブジェクトファイル
※ [オプション] <ソースファイル> [オブジェクトファイル] の順序は任意に指定可能

コマンド例:

```
icc -O3 -xAVX -ipo main.c func1.c func2.c obj_c.o
icpc main.cpp func1.cpp func2.cpp obj_cpp.o
ifort -O3 -xSSE4.2 prog.f90 sub.f -o my_prog_name.out
```

※上記 obj_c.o は .c のオブジェクト、 obj_cpp は .cpp のオブジェクト

使用頻度の高いオプション例 (Linux 版)

自動並列化(マルチスレッド)	mpiicc -parallel 〈ファイル〉
OpenMP:	mpiicc -openmp 〈ファイル〉
ループの最適化	mpiicc -O3 〈ファイル〉
ベクトル化 (SandyBridge 対応)	mpiicc -xAVX 〈ファイル〉
プロシージャ間の最適化	mpiicc -ipo 〈ファイル〉
ベクトル化レポート	mpiicc -vec_report3 〈ファイル〉
インテル® Trace Collector をリンク	mpiicc -trace -g 〈ファイル〉 ※詳細はインテル® Trace Analyzer/ Collector の使用方法にて説明

推奨設定:

mpiicc -O3 -xAVX -ipo [-openmp] [-parallel] 〈ファイル〉

※ mpiicc の他に、iccやifort, mpiicpc, mpiifort を使用した場合も同様
(ただし -trace はmpiicc, mpiicpc, mpifort のみ対応)

主要なコンパイラーオプション一覧 (Linux 版)

オプション	概要
-O0	全ての最適化オプションを無効にする
-O1, -O2, -O3	ループの最適化 (3が最も強力)
-x, -ax, -m	ベクトル化、各プロセッサー向けの最適化
-ipo	プロシージャ間の最適化
-openmp	OpenMP を利用
-parallel	自動並列化
-prof_gen, -prof_use	プロファイルによる最適化 (Profile Guided Optimization〔PGO〕)
-guide	ガイド付き自動並列化
-vec_report*〈Number〉*	ベクトル化のレポート *(数字は1,2,3,4,5 選択可)*
-par_report*〈Number〉*	自動並列化のレポート *(数字は1,2,3 選択可)*
-fast	-ipo、-O3、-no-prec-div、-static、-xHOST を有効にする
-fp-model *〈model〉*	浮動小数点数演算の精度を制御 *(modelは、strict, precise, fast 等選択可)*

主要なコンパイラーオプション一覧つづき (Linux 版)

オプション	概要
-prec-div	浮動小数点数演算の精度を向上させる
-static-intel, -shared-intel	インテル® ライブラリーのリンク方法を指定(初期値はstatic)
-mkl *[=lib]* *使用例:* *-mkl=cluster*	インテル® MKL をリンクする *(=lib は以下を選択可能)* *parallel* マルチスレッド化されたインテル® MKL のスレッドをリンクする。lib 未指定時のデフォルト *sequential* シングルスレッドのインテル® MKL のスレッドをリンクする *cluster* クラスターに対応した関数のインテル® MKL と、その他シングルスレッドのインテル® MKL をリンクする

スレッド・アフィニティーの調整

- OpenMP* スレッドを物理もしくは論理コアへの割り当てを調整
  - export KMP_AFFINITY=
    - physical　論理コアへ割り当てるまでにすべての物理コアを利用 (ハイパースレッディング)
    - compact　同じソケットの連続したコアへ割り当て (共有キャッシュを効率よく利用)
    - scatter　ソケットごとに交互に割り当て (メモリー帯域を最大限に活用)
  - メモリーやキャッシュのアクセス最適化に役立つ
  - ハイパースレッディングが有効の場合特に重要
  - 詳細はコンパイラーのドキュメントを参照

インテル® Composer XE 2011
最適化クイック・リファレンス・ガイド

- 最適化の手順と、基本的なオプションが一覧になったドキュメント
  http://jp.xlsoft.com/documents/intel/compiler/qr_guide_jp.pdf

1. ループの最適化
2. ベクトル化
3. プロシージャ間の最適化 (IPO)
   プロファイルに基づく最適化(PGO)
4. ガイド付き自動並列化(GAP)
5. インテル® VTune™ Amplifier XEの利用

高度な最適化: -O3
（High-Level Optimization: HLO）

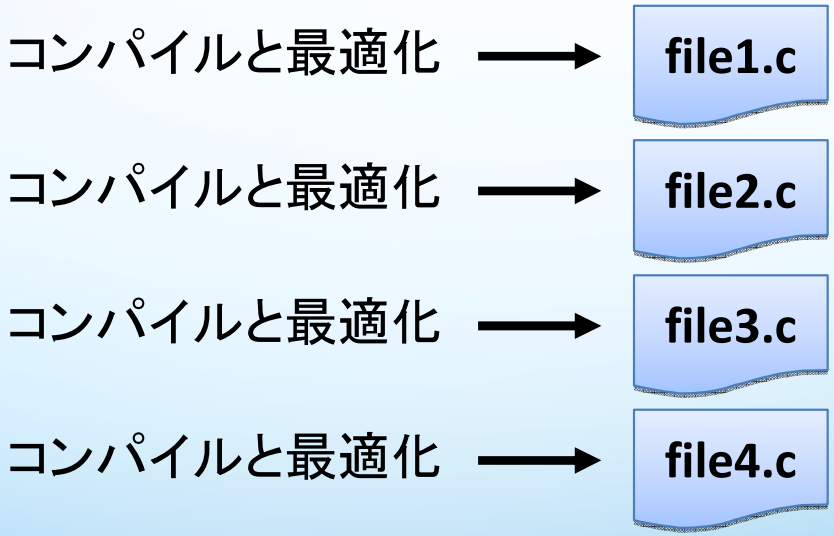
- ループの最適化
  - ループのアンロール、並べ替え、プリフェッチャの挿入
  - 自動ベクトル化オプション（-x, -ax）と併用することにより
    強力なデータ依存解析を行い、ベクトル化の可能性を高める

コンパイラーオプション	
-O0	最適化を無効にする
-O1	実行速度の最適化（コードサイズの増加なし）
-O2	実行速度の最適化（デフォルト）
-O3	高度な最適化

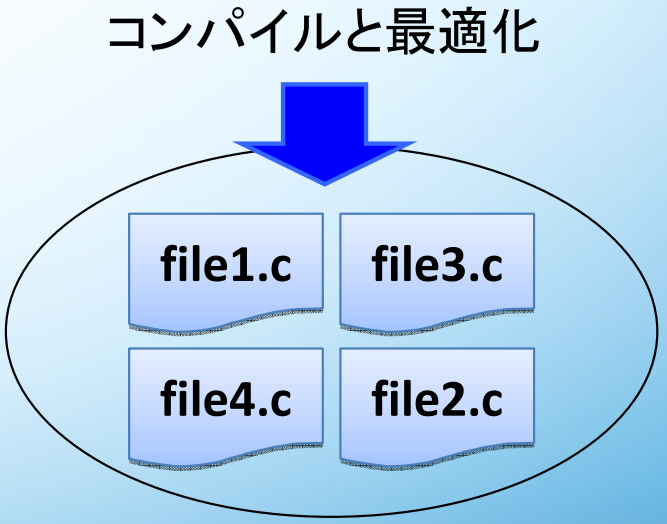
プロシージャ間の最適化: -ipo（Inter Procedural Optimization: IPO）

- 複数ソースファイル間での参照関係を考慮し、プログラム全体の構成を最適化
  - 関数のインライン展開、定数伝播、不要な処理の削除など
  - エイリアス解析などにより、自動ベクトル化を促進

IPOを使用しない場合



IPOを使用した場合



自動並列化: -parallel（Parallelizer）

- ループ文を解析し、マルチスレッド化されたコードを生成
  - 安全性: 各反復に依存性があると並列化しない
  - 効率性: 処理量が小さい場合には並列化しない
- レポートオプションにて、並列化の成否を確認可能
  - -par_report[n]　　n は 0,1,2,3 のいずれか

```
int_sin.cpp(74): (col. 4) remark: ループは並列化されませんでした: 並列依存関係が存在しています。.
int_sin.cpp(92): (col. 6) remark: ループは並列化されませんでした: 計算量が不足しています。.
```

- 効率性については、しきい値による調整が可能
  - -par-threshold[n]　n は 0～100 の間

```
int_sin.c(92): (col. 6) remark: ループが自動並列化されました。.
```

自動ベクトル化: -m, -x, -ax（Vectorizer）

- スカラー演算を SIMD（Single Instruction Multiple Data）演算に自動変換して、処理効率の良いコードを生成

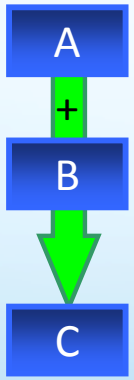
例）

```
for ( i=0; i<MAX; i++ )
    c[i] = a[i] + b[i];
```

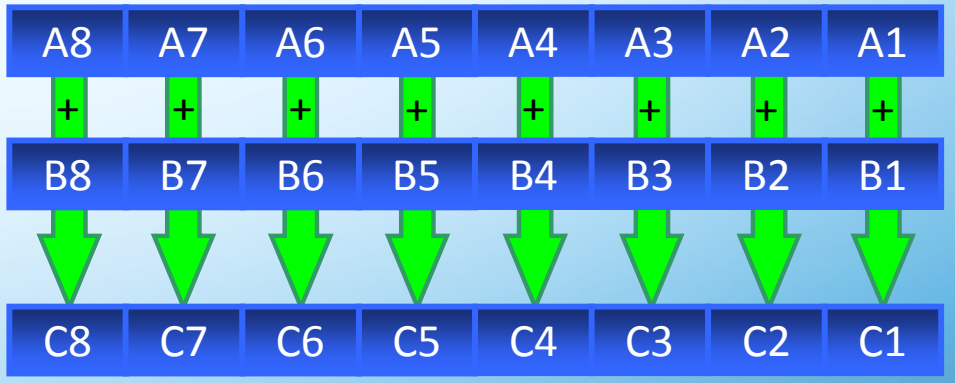
-no-vec（ベクトル化をしない）

-xAVX（Intel AVX 命令を使用）

スカラー演算：1命令で1結果 × MAX 回



SIMD演算：1命令で8結果 ×（MAX/8）回



※ AVX 、float の場合

自動ベクトル化オプション -m, -x, -ax の違い

オプション	概要
-x<*code*> <*code*>=Host,AVX,SSE4.2,SSE4.1,SSE3_ATOM,SSSE3,SSE3,SSE2	インテルプロセッサー専用の最適化を行い、AVX、SSE4.2などの命令セットを生成する 例: -xAVX は AVX、SSE4.2、SSE4.1、SSSE3、SSE3、SSE2、SSEを生成
-ax<*code*> <*code*>=AVX,SSE4.2,SSE4.1,SSSE3,SSE3,SSE2	インテルプロセッサー専用の最適化と、AFXやSSE4.2 等と、汎用命令セットを生成する 例: -axAVX は AVX、SSE4.2、SSE4.1、SSSE3、SSE3、SSE2、SSEと、汎用コードを生成
-m<*code*> <*code*>=ia32, sse2, sse3, ssse3, sse4.1	インテル互換プロセッサーに対応する、ia32、sse2命令セットを生成する 例: -mSSE4.1 は SSE4.1、SSSE3、SSE3、SSE2、SSE を生成

-ax オプション補足

- 複数のプロセッサーに対応させたい場合、対象プロセッサーを複数設定することができる

例: -axAVX,SSE4.2,SSE3 は以下のように動作する

- AVX をサポートするプロセッサー: AVX までを実行
- SSE4.2 をサポートするプロセッサー: SSE4.2 までを実行
- SSE4.1 をサポートするプロセッサー: SSE3 までを実行
- SSE2 をサポートするプロセッサー: 汎用コードを実行

ガイド付き自動並列化: -guide
（Guided Auto Parallelization: GAP）

- コンパイラーが最適化に関するアドバイスを提供
  - 自動ベクトル化　　-guide-par
  - 自動並列化　　-guide-vec
  - データ構造の改良　　-guide-data-trans
- メッセージ例（自動ベクトル化について）

gap_vec.c(8): remark #30536: (LOOP) 必要に応じて、-fargument-noalias オプションを追加してコンパイラーによる型ベースの一義化解析を向上できます (オプションはコンパイル全体に適用されます)。これにより、行 8 のループがベクトル化され最適化が向上します。[確認] コンパイル全体でこのオプションのセマンティクスに沿っていることを確認してください。[別の方法] ルーチン "matrix_mul_matrix" のすべてのポインター型の引数に "restrict" キーワードを追加することで同様の効果が得られます。これにより、行 8 のループがベクトル化され最適化が向上します。[確認] "restrict" ポインター修飾子のセマンティクスに沿っていることを確認してください。ルーチンにおいて、そのポインターによってアクセスされるすべてのデータは、ほかのポインターからはアクセスできません。

浮動小数点数演算の精度: -fp-model

- 浮動小数点数の演算は、コンパイラーの最適化により異なった結果を生じる場合がある
- precise や strict を指定した場合は、一部のベクトル化が抑制される（パフォーマンスと精度はトレードオフ）

ソースコードの記述

```
float t0, t1, t2
t0 = 4.0f + 0.1f + t1 + t2
```

オプション	効果
-fp-model fast（デフォルト）	演算順序の変更を含む様々な最適化を行う `t0 = 4.1f + t1 + t2`
-fp-model precise	演算順序を変更しない、その他の最適化は行う `t0 = (4.1f + t1) + t2`
-fp-model strict	演算順序を変更せず、厳密に計算する `t0 = (((4.0f + 0.1f) + t1) + t2)`

XLSOFT

最適化に関する注意事項

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

Optimization Notice

最適化に関する注意事項

インテル コンパイラーは、互換マイクロプロセッサー向けには、インテル製マイクロプロセッサー向けと同等レベルの最適化が行われない可能性があります。これには、インテル ストリーミング SIMD 拡張命令 2 (インテル SSE2)、インテル ストリーミング SIMD 拡張命令 3 (インテル® SSE3)、ストリーミング SIMD 拡張命令 3 補足命令 (SSSE3) 命令セットに関連する最適化およびその他の最適化が含まれます。インテルでは、インテル製ではないマイクロプロセッサーに対して、最適化の提供、機能、効果を保証していません。本製品のマイクロプロセッサー固有の最適化は、インテル製マイクロプロセッサーでの使用を目的としています。インテル® マイクロアーキテクチャーに非固有の特定の最適化は、インテル製マイクロプロセッサー向けに予約されています。この注意事項の適用対象である特定の命令セットの詳細は、該当する製品のユーザー・リファレンス・ガイドを参照してください。

改訂 #20110804

XLSOFT