インテル® ソフトウェア開発ツールで始める
インテル® コンパイラーによる高度な最適化

応用編

2019 年 4 月
iSUS 編集長
すがわら　きよふみ

本日の予定

10:30 – 17:00

• コードの現代化と最適化	10:30 – 11:00
• 並列アプリケーションのスケーラビリティーの問題を特定	11:00 – 12:00
昼食	
• インテル® コンパイラーによる高度な最適化	13:00 – 14:00
• インテル® VTune™ Amplifier 応用編	14:00 – 15:00
休憩	
• インテル® Advisor 応用編	15:15 – 16:15
• インテル® Inspector 応用編	16:15 – 16:45
• まとめと Q/A	16:45 – 17:00

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

内容

- ソースの移植に役立つ機能
- 利用可能なリソース
- 互換性制御
- コンパイラー・オプション
- コンパイラー・プラグマ
- インテル® マイクロアーキテクチャー Skylake Server† 向けプログラミング

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

移植に役立つ機能: 開発環境間でのオプション

- インテル® コンパイラーのコンパイラー・オプション・マッピング・ツールは、Windows* と Linux* の対応するオプションを知る簡単な機能を提供します
- Windows* ベースのアプリケーションの開発者が Linux* のアプリケーションを開発する場合、例えば、
  /GS- オプションに対応する Linux* オプションを知るには…

GS

バッファー・オーバーランを検出するコードを生成するかどうかを指定します。

構文

Linux* および macOS*:	なし
Windows*:	/GS[:keyword]
	/GS-

Intel Compiler 19.0 Update 3 Intel(R) 64 Visual Studio 2017

```
C:¥Users¥kiyo¥Desktop>icl /Qmap-opts /GS-
インテル(R) 64 対応インテル(R) C++ コンパイラー (インテル(R) 64 対応アプリケーション用) バージョン 19.0.3.203 ビルド 20190206
(C) 1985-2019 Intel Corporation.  無断での引用、転載を禁じます。

Intel(R) Compiler option mapping tool

mapping Windows options to Linux for C++

'-Qmap-opts' Windows option maps to
  --> '-map-opts' option on Linux
  --> '-map opts' option on Linux

'-GS-' Windows option maps to
  --> '-fno-stack-protector' option on Linux
  --> '-fno-stack-protector-all' option on Linux
  --> '-fno-stack-security-check' option on Linux
```

Linux* と Windows* 環境向け (C/F)

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

移植に役立つ機能: 開発環境間でのオプション (ツール)

```
map_opts [-nologo] [-t {linux | windows}] [-l {c | fortran}] -opts {オプション}
```

ターゲット　　言語

Intel Compiler 19.0 Update 3 Intel(R) 64 Visual Studio 2017

```
C:¥Users¥kiyo¥Desktop>map_opts -nologo -opts /GS-

'-GS-' Windows option maps to
  --> '-fno-stack-protector' option on Linux
  --> '-fno-stack-protector-all' option on Linux
  --> '-fno-stack-security-check' option on Linux

C:¥Users¥kiyo¥Desktop>map_opts -nologo -twindows -opts -g

'-g' Linux option maps to
  --> '-Zi' option on Windows
```

Linux* と Windows* 環境向け (C/F)

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

移植に役立つ機能: ソースの問題をレポート

■ **`-diag-enable=port-win`**
Windows* に移植するときにエラーになる可能性のある GNU* 拡張機能に対する診断メッセージを指定します。
この診断グループは、Linux* および macOS* システムでのみ利用できます

■ **`/Qdiag-enable:port-linux`**
Linux* システムに移植するときにエラーになる可能性のある言語機能に対する診断メッセージを指定します。
この診断グループは、Windows* システムでのみ利用できます

Intel Compiler 19.0 Update 3 Intel(R) 64 Visual Studio 2017

```
$ icl FMA_count.c /Qdiag-enable:port-linux
インテル(R) 64 対応インテル(R) C++ コンパイラー (インテル(R) 64 対応アプリ
ケーション用) バージョン 19.0.3.203 ビルド 20190206
(C) 1985-2019 Intel Corporation.  無断での引用、転載を禁じます。

FMA_count.c
FMA_count.c(15): 警告 #2245: "__declspec" は Microsoft* 拡張です。
  __declspec(align(64)) double one_vec[8] = {1, 1, 1, 1,1, 1, 1, 1};
  ^

FMA_count.c(16): 警告 #2245: "__declspec" は Microsoft* 拡張です。
  __declspec(align(64)) int shuf_vec[16] = {0, 1, 2, 3,4, 5, 6, 7,8, 9, 10,
 11,12, 13, 14, 15};
  ^

FMA_count.c(79): 警告 #2245: "__declspec" は Microsoft* 拡張です。
  __declspec(align(64)) double one_vec[8] = {1, 1, 1, 1,1, 1, 1, 1};
  ^

Microsoft (R) Incremental Linker Version 14.16.27027.1
Copyright (C) Microsoft Corporation.  All rights reserved.

-out:FMA_count.exe
FMA_count.obj
```

Linux* と Windows* 環境向け

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

移植に役立つ機能: 言語方言のサポート

■ **-fms-dialect[= 14.1 | 14 | 12] (2017, 2015, 2013)**
Linux* 環境で gcc とのリンク互換を保ちながら、Microsoft* Windows* と互換性のある限定的な言語方言のサポートを有効にします。これにより、Microsoft* の拡張や言語機能を使用して Windows* 上で記述されたコードの移植を可能にします

■ **/Qgcc-dialect[: ver] (ver は、440 以上の 3 桁)**
Windows* 環境で gcc と互換性のある限定的な方言のサポートを有効にします。これにより、gcc コンパイラー向けに記述されたコードの移植を可能にします

Linux* と Windows* 環境向け

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

移植に役立つ機能: 言語方言のサポート続き…

Intel Compiler 19.0 Update 3 Intel(R) 64 Visual Studio 2017

```
> icl gcc_lang.c /nologo
gcc_lang.c
gcc_lang.c(1): エラー: ";" を指定してください。
  typedef unsigned int uint32_t __attribute__ ((__mode__ (__SI__)));
                                ^

コンパイルは gcc_lang.c で異常終了しました (コード 2)。
```

`__attribute__` は gcc 拡張であるため、Visual* C++ や Windows* 版のインテル® コンパイラーでは、デフォルトではエラーとなります

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

移植に役立つ機能: MASM 形式のアセンブリー

- Linux* 上の C または C++ ファイル内で、ブロックおよび全関数の MASM 形式のアセンブリー・コードの使用を有効にします
  - `-use-msasm`
  - `-fasm-blocks`

例: `icc -fasm-blocks asm.c`

「インテル® 64 および IA-32 アーキテクチャー最適化リファレンス・マニュアル」に掲載される例の一部

```
#include <string.h>
#include <stdlib.h>
#include <immintrin.h>
#include <stdio.h>
#include <stdint.h>

static uint64_t rdtsc(void) {
unsigned int ax, dx;
   __asm__ __volatile__ ("rdtsc" : "=a"(ax), "=d"(dx));
   return ((((uint64_t)dx) << 32) | ax);
}

uint64_t fma_shuffle_tpt(uint64_t loop_cnt){
uint64_t loops = loop_cnt;
__declspec(align(64)) double one_vec[8] = {1, 1, 1, 1,1, 1, 1, 1};
__declspec(align(64)) int shuf_vec[16] = {0, 1, 2, 3,4, 5, 6, 7,8,
9, 10, 11,12, 13, 14, 15};

   __asm
   {
              vmovups zmm0, [one_vec]
              vmovups zmm1, [one_vec]
              vmovups zmm2, [one_vec]
              vmovups zmm3, [one_vec]
              vmovups zmm4, [one_vec]
              vmovups zmm5, [one_vec]
              ……
              dec rdx
              jg loop1
   }
}
```

Linux* 環境向け

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

利用可能なリソース: 言語規格 C/C++

Linux* と Windows* 環境向け

■ **-std=val, /Qstd=val**

特定の言語標準に準拠するようにコンパイラーに指示します

値		値	
c++17	2017 ISO C++ 標準機能のサポートを有効	gnu++98	1998 ISO C++ 規格と GNU* 拡張機能に準拠
c++14	2014 ISO C++ 標準機能のサポートを有効	gnu99	ISO C99 と GNU* 拡張機能に準拠
c++11, c++0x, gnu++0x	ISO/IEC 9899:2011 国際規格に準拠	c89	ISO/IEC 9899:1990 国際規格に準拠
c11	ISO/IEC 9899:2011 国際規格に準拠	gnu89	ISO C90 と GNU* 拡張機能に準拠
c99	ISO/IEC 9899:1999 国際規格に準拠	デフォルト	C は gnu89、C++ は gnu++98 (Linux*)

Windows* 環境のデフォルトは、使用する Visual Studio* C++ のバージョンに準拠

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

利用可能なリソース: 言語規格 Fortran

Linux* と Windows* 環境向け

- **-f66, -f77rtl (Linux*) /f66, /f77rtl (Windows*)**
  特定の言語標準に準拠するようにコンパイラーに指示します

値		値	
f66	Fortran 66 セマンティクスを使用します	f77rtl	FORTRAN 77 ランタイム動作の使用をコンパイラーに指示します

デフォルトは Fortran 95 セマンティクスを使用します

- **-vms (Linux*) /vms (Windows*)**
  OpenVMS* Alpha システムおよび VAX* システム (VAX Fortran) 上の HP* Fortran のように振る舞います
- **-fpscomp (Linux*) /fpscomp (Windows*)**
  Fortran PowerStation からアプリケーションを移植する際に問題が生じた場合、fpscomp (または fpscomp all) を指定します

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

言語が混在したプログラミング

Linux* と Windows* 環境向け

Fortran と C/C++ が混在したプログラミングのまとめは次のとおりです:
Fortran と C は互いのルーチンを呼び出すことができるので、メインルーチンはどちらの言語でも記述することができます:

- Linux* および macOS* システムでは、メインルーチンが Fortran でない場合、アプリケーションをリンクする ifort コマンドラインで **-nofor-main** コンパイラー・オプションを指定します
- メインプログラムが Fortran で記述されている場合、Fortran コンパイラーは Fortran ランタイム・ライブラリー (RTL) を初期化するために必要なコードを自動的に作成します。RTL は、入力/出力と例外処理用の Fortran 環境を提供します
- メインプログラムが C/C++ で記述されている場合、メインプログラムで **for_rtl_init_**を呼び出して Fortran を初期化する必要があります。また、メインプログラムの最後で **for_rtl_finish_** を呼び出して Fortran RTL をシャットダウンする必要があります

詳細は、デベロッパー・ガイドおよびリファレンスの
「言語が混在したプログラミングのまとめ」を参照

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

利用可能なリソース: GNU* C と ANSI C

- **`-ansi`**

gcc オプションの **`-ansi`** と言語互換性を有効にし、**`-ansi`** オプションと同じレベルの ANSI 標準規格との適合性を提供します
ANSI 標準規格に厳密に準拠するには、**`-strict-ansi`** オプションを指定します

デフォルトはオフで、ANSI C より、GNU* C++ の方が強固にサポートされます

Linux* 環境向け

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

内容

- ソースの移植に役立つ機能
- 利用可能なリソース
- 互換性制御
- **コンパイラー・オプション**
  - **命令セットと命令スケジュール**
- コンパイラー・プラグマ
- インテル® マイクロアーキテクチャー Skylake Server† 向けプログラミング

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

利用可能なリソース: オプションとプラグマ

- icc, icpc, icl -help

Intel Compiler 19.0 Update 3 Intel(R) 64 Visual Studio 2017 - icl -help

```
Z:¥>icl -help
インテル(R) 64 対応インテル(R) C++ コンパイラー (インテル(R) 64 対応アプリケーション用) バージ
ョン 19.0.3.203 ビルド 20190206
(C) 1985-2019 Intel Corporation.  無断での引用、転載を禁じます。

                    インテル(R) C++ コンパイラー・ヘルプ
                    ====================================


 インテル(R) コンパイラーには、インテル製マイクロプロセッサーおよび
 互換マイクロプロセッサーで利用可能な命令セットの生成を制御する最適化
 オプションが含まれていますが、両者では結果が異なります。
```

- icc, icpc -help-pragma, icl /Qhelp-pragma

Intel Compiler 19.0 Update 3 Intel(R) 64 Visual Studio 2017

```
Z:¥>icl /Qhelp-pragma
インテル(R) 64 対応インテル(R) C++ コンパイラー (インテル(R) 64 対応アプリケーション用) バージ
ョン 19.0.3.203 ビルド 20190206
(C) 1985-2019 Intel Corporation.  無断での引用、転載を禁じます。

196041dummy3.c
PRAGMA                  ARGUMENTS
=====================================================
__printf_args           <none>
__scanf_args            <none>
instantiate             {id | function declaration}
do_not_instantiate      {id | function declaration}
can_instantiate         {id | function declaration}
inline_template        {id | function declaration}
pack                    {({push|pop {,name} {,n}|show})|([n])}
```

Linux* と Windows* 環境向け

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

生成する命令セットを含む、ターゲット機能をコンパイラーに指示する方法

最適化に関する注意事項

インテル® コンパイラーでは、インテル® マイクロプロセッサーに限定されない最適化に関して、他社製マイクロプロセッサー用に同等の最適化を行えないことがあります。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® ストリーミング SIMD 拡張命令 3、インテル® ストリーミング SIMD 拡張命令 3 補足命令などの最適化が該当します。インテルは、他社製マイクロプロセッサーに関して、いかなる最適化の利用、機能、または効果も保証いたしません。本製品のマイクロプロセッサー依存の最適化は、インテル® マイクロプロセッサーでの使用を前提としています。インテル® マイクロアーキテクチャーに限定されない最適化のなかにも、インテル® マイクロプロセッサー用のものがあります。この注意事項で言及した命令セットの詳細については、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。

注意事項の改訂 #20110804

インテル® コンパイラーは以下のコードを生成できます:

1. 特定の機能をサポートするプロセッサー専用のコード
2. 他社製マイクロプロセッサーで実行可能なコード
3. インテル® マイクロプロセッサー専用のコード

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

命令生成における最適化

プロセッサーのアーキテクチャーに最適な命令の**スケジュール**とプロセッサーがサポートする**命令セット**を生成

- -march (Linux*), /arch (Windows*)

生成する命令セットを含む、ターゲット機能をコンパイラーに指示します

- -mtune (Linux*), /tune (Windows*)

特定のプロセッサー向けの最適化を実行しますが、(-march, /arch とは違い) 拡張命令セットは使用しません

- -m (Linux*)

生成する命令セットを含む、ターゲット機能をコンパイラーに指示します

- -x (Linux*), /Qx (Windows*)

生成する命令セットと最適化を含む、ターゲット・プロセッサー機能をコンパイラーに指示します

- -ax (Linux*), /Qax (Windows*)

パフォーマンス上の利点がある場合、複数の機能固有の**自動ディスパッチ・コード**を生成するようにコンパイラーに指示します

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

/arch:引数 (Windows*)

このオプションで生成されたコードは、**対応する命令セットがサポートされている互換性**のあるインテル以外のプロセッサーで動作します

引数 (プロセッサーのコード名):
BROADWELL、CANNONLAKE、HASWELL、ICELAKE-CLIENT (または ICELAKE)、ICELAKE-SERVER、IVYBRIDGE、KNL、KNM、SANDYBRIDGE、SILVERMONT、GOLDMONT、GOLDMONT-PLUS、TREMONT、SKYLAKE、SKYLAKE-AVX512、CASCADELAKE、KABYLAKE、COFFEELAKE、AMBERLAKE、WHISKEYLAKE

引数 (命令セット):
MIC-AVX512、COMMON-AVX512、CORE-AVX512、CORE-AVX2、CORE-AVX-I、AVX、SSE4.2、SSE4.1、SSSE3、SSE3、SSE2

- /arch オプションと /Qx オプションは互いに排他的であり、同時に両方を指定することはできません。両方指定すると、コンパイラーは後に指定されたオプションを使用し、警告を発行します
- /Qax オプションと /arch オプションの両方を指定すると、コンパイラーはインテル固有の命令を生成しません

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

-march=引数 (Linux*)

このオプションは、**特定の機能をサポートするプロセッサー専用のコード**を生成するようにコンパイラーに指示します

引数 (コード名):
broadwell、cannonlake、haswell、icelake-client (または icelake)、icelake-server、ivybridge、knl、knm、sandybridge、silvermont、 goldmont、goldmont-plus、tremont、、skylake、skylake-avx512、CASCADELAKE、KABYLAKE、COFFEELAKE、AMBERLAKE、WHISKEYLAKE

引数 (命令セットとプロセッサー名):
common-avx512、core-avx512、core-avx2、core-avx-i、corei7-avx、corei7、atom、core2、pentium4m、pentium-m、pentium4、pentium3、pentium

- -ax オプションと -march オプションの両方を指定すると、コンパイラーはインテル固有の命令を生成しません
- -march=pentium4 を指定すると -mtune=pentium4 が設定されます
- 互換性のために、以前の引数値もサポートされますが、生成されるコードはデフォルトと同じです

-march=	代替オプション
pentium3	-xSSE
pentium4 pentium-m	-xSSE2
core2	-xSSSE3

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

-tune=引数 (Linux*)、/tune:引数 (Windows*)

このオプションは、特定のプロセッサー向けの最適化を実行しますが、(-march とは違い) **拡張命令セットは使用しません**

引数:

generic、broadwell、cannonlake、goldmont、goldmont-plus、haswell、icelake-client (または icelake)、icelake-server、ivybridge、knl、knm、sandybridge、silvermont、skylake、skylake-avx512、tremont

common-avx512、core-avx512、core-avx2、core-avx-i、corei7-avx、corei7、atom、core2、pentium-mmx、pentiumpro、pentium4m、pentium-m、pentium4、pentium3、pentium

作成される実行ファイルは下位互換ですが、生成されるコードは特定のプロセッサー向けに最適化されます:

- 例えば、-mtune=core2 (/tune:core2) で生成されたコードは、第 4 世代インテル® Core™ プロセッサー上でも正しく実行されますが、-mtune=haswell (/tune:haswell) で生成されたコードよりも実行速度が遅い可能性があります
- -mtune=haswell (/tune:haswell) または -mtune=core-avx2 (/tune:core-avx2) で生成されたコードは、インテル® Core™2 プロセッサーでも正しく実行されますが、-mtune=core2 または /tune:core2 で生成されたコードよりも実行速度が遅い可能性があります。
- -march=core-avx2 (/arch:core-avx2) で生成されたコードは、インテル® Core™2 プロセッサーなどの古いプロセッサーでは正しく実行されません

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

-m引数 (Linux*)

このオプションは、生成する命令セットを含む、ターゲット機能をコンパイラーに指示します。このオプションで生成されたコードは、対応する命令セットがサポートされている互換性のあるインテル以外のプロセッサーで動作します

引数:
avx、sse4.2、sse4.1、ssse3、sse3、sse2、sse、ia32

- このコンパイラー・オプションは、gcc で利用可能な -m オプションの設定の多くもサポートします。gcc -m の設定に関する詳細は、gcc ドキュメントを参照してください
- -x オプションと -m オプションは互いに排他的であり、同時に両方を指定することはできません。両方指定すると、コンパイラーは後に指定されたオプションを使用し、警告を発行します

gcc 互換オプション (-m)	代替オプション
fma	-march=core-avx2
bmi、avx2、lzcnt	-march=core-avx2
movbe	-march=atom –minstruction=movbe
crc32、aes、pclmul、popcnt	-march=corei7
vzeroupper	-march=corei7-avx
fsgsbase、rdrnd、f16c	-march=core-avx-i

gcc との互換性のために、コンパイラーは表のオプションを受け付けますが効果はありません

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

注意

- -arch、/arch、-mtune、/tune、-m で生成されるバイナリーではプロセッサー・チェックが行われません
- /arch:skylake-avx512 (-march=skylake-avx512) で命令を生成しても、必ずしも AVX-512 命令が生成されるとは限りません

生成したバイナリーは対象より下位のプロセッサーで実行すると例外が発生する可能性があります

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

インテル® コンパイラー専用オプション: -x, /Qx

このオプションは、生成する命令セットと最適化を含む、ターゲット・プロセッサー機能をコンパイラーに指示します。また、インテルの機能固有の最適化に加えて新しい最適化を有効にします

オプション	引数
-x引数 (Linux*) /Qx引数 (Windows*)	BROADWELL、CANNONLAKE、GOLDMONT、GOLDMONT-PLUS、HASWELL、ICELAKE-CLIENT (または ICELAKE)、ICELAKE-SERVER、IVYBRIDGE、KNL 、KNM、SANDYBRIDGE、SILVERMONT、SKYLAKE、SKYLAKE-AVX512、TREMONT、CASCADELAKE、KABYLAKE、COFFEELAKE、AMBERLAKE、WHISKEYLAKE COMMON-AVX512、MIC-AVX512、CORE-AVX512、CORE-AVX2、CORE-AVX-I、AVX、SSE4.2、SSE4.1、ATOM_SSE4.2、ATOM_SSSE3、SSSE3、SSE3、SSE2

-x オプションと /Qx オプションは、-m または /arch (または -ax および /Qax) では有効にされない追加の最適化を有効にします

注: SSE2 を除くすべての設定で CPU チェックが行われます。ただし、-O0 (Linux* および macOS*) または /Od (Windows*) を指定すると、CPU チェックは行われません

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

プロセッサー・チェックとは?

Intel Compiler 19.0 Update 3 Intel(R) 64 Visual Studio 2017

```
> icl code.c /arch:skylake-avx512 /nologo
code.c

> code
0.564642

> icl code.c /Qxskylake-avx512 /nologo
code.c

> code

Please verify that both the operating system
AVX512DQ, AVX512F, AVX512CD, AVX512BW, AVX512VL and CLWB instructions.
```

```
sub     rsp, 40                     ;1.7
xor     edx, edx                    ;1.7
mov     ecx, 3                      ;1.7
call    __intel_new_feature_proc_init
```

```
sub     rsp, 40                     ;1.7
mov     rdx, 020064199d9ffeH        ;1.7
mov     ecx, 3                      ;1.7
call    __intel_new_feature_proc_init
```


© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

コード中で CPU の命令セットをチェックする

```
#include <immintrin.h>
#include <stdio.h>

int main(int argc, char *argv[]) {
  const unsigned long knl_features =
      (_FEATURE_AVX512F | _FEATURE_AVX512ER |
       _FEATURE_AVX512PF | _FEATURE_AVX512CD );
  if ( _may_i_use_cpu_feature( knl_features ) )
    printf("This CPU supports AVX-512F+CD+ER+PF as intro
  else
    printf("This CPU does not support all Knights Landir
  return 1;
}
```

```
…
#define _FEATURE_AVX512DQ            (1ULL << 24)
#define _FEATURE_PTWRITE             (1ULL << 25)
#define _FEATURE_AVX512F             (1ULL << 27)
#define _FEATURE_ADX                 (1ULL << 28)
#define _FEATURE_RDSEED              (1ULL << 29)
#define _FEATURE_AVX512IFMA52        (1ULL << 30)
#define _FEATURE_AVX512ER            (1ULL << 32)
#define _FEATURE_AVX512PF            (1ULL << 33)
#define _FEATURE_AVX512CD            (1ULL << 34)
#define _FEATURE_SHA                 (1ULL << 35)
#define _FEATURE_MPX                 (1ULL << 36)
#define _FEATURE_AVX512BW            (1ULL << 37)
#define _FEATURE_AVX512VL            (1ULL << 38)
#define _FEATURE_AVX512VBMI          (1ULL << 39)
#define _FEATURE_AVX512_4FMAPS       (1ULL << 40)
#define _FEATURE_AVX512_4VNNIW       (1ULL << 41)
#define _FEATURE_AVX512_VPOPCNTDQ    (1ULL << 42)
#define _FEATURE_AVX512_BITALG       (1ULL << 43)
#define _FEATURE_AVX512_VBMI2        (1ULL << 44)
…
```

インテル® コンパイラーは、プロセッサーの機能をチェックする
_may_i_use_cpu_feature() 組込み関数をサポート

https://www.isus.jp/specials/target-intel-core-processors/

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

複数ファイルのソースをコンパイルする注意

プログラムソースが複数ファイルで構成される場合、異なるコード生成オプションや異なるコンパイラーを使用する場合は、注意が必要です

```
main(){
   func();
}
```

```
func(){
   …;
}
```

```
icl /QxSKYLAKE-AVX512 func.c /c
icl /QxAVX main.c func.obj

icl /QxSKYLAKE-AVX512 func.c /c
cl main.c func.obj
```

プロセッサー・チェックは main 関数でのみ行われます

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

インテル® コンパイラー専用オプション: -ax, /Qax

- このオプションは、パフォーマンス上の利点がある場合、複数の機能固有の自動ディスパッチ・コードを生成するようにコンパイラーに指示します
- また、汎用コードも生成します。通常、インテルの機能固有の自動ディスパッチ・コードは汎用コードよりも最適化されます。O3 などのその他のオプションは、汎用コードでどの程度の最適化を行うかを制御します
- -ax引数、/Qax引数の引数は –x、/Qx とコード名や命令セットを指定できます
- 2 つ以上の機能の値を組み合わせて使用できます。例えば、-axSSE4.1,SSSE3 (Linux* および macOS*) または /QaxSSE4.1,SSSE3 (Windows*) のように指定することができます

-axAVX2 (/QaxAVX2) を指定するとデフォルトの汎用命令セット SSE2 と AVX2 を使用する 2 つのバイナリーが生成されます

-axAVX2 –xAVX (/QaxAVX2 /QxAVX) と指定すると、デフォルトの汎用命令セットに AVX を使用するバイナリーが生成されます

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

汎用コードの生成

- -ax と -m オプション (Linux* および macOS*) または /Qax と /arch オプション (Windows*) の両方を指定すると、-m または /arch オプションで指定された設定と互換性のあるインテル以外のプロセッサーで実行できる汎用コードが生成されます
- -ax と -march オプション (Linux* および macOS*) または /Qax と /arch オプション (Windows*) をの両方を指定すると、コンパイラーはインテル固有の命令を生成しません

機能固有の関数バージョンを生成できることが判明した場合、コンパイラーはそれがパフォーマンスの向上につながるかどうかをチェックします。パフォーマンスが向上すると判明した場合、コンパイラーは機能固有バージョンと汎用バージョンの関数を生成します

プログラムの実行時に、使用されているインテル® プロセッサーに応じて、この 2 つのバージョンのどちらを実行するかが選択されます。この方法により、プログラムは従来のプロセッサーやインテル以外のプロセッサーとの互換性を保ちながら、最新のインテル® プロセッサー上でパフォーマンスを大幅に向上できます。インテル以外のプロセッサーは、常に汎用コードを実行します

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

-xHOST (Linux*)、/QxHOST (Windows*) オプション

- このオプションは、コンパイルを行うホスト・プロセッサーで利用可能な最上位の命令セット向けのコードを生成するようにコンパイラーに指示します
- このコンパイラー・オプションで生成される命令は、コンパイルを行うホスト・プロセッサーによって異なります

プロセッサーがインテル® AVX2 命令セットをサポートする例

インテル® プロセッサー上でコンパイルする場合:

[Q]xCORE-AVX2 に相当します。生成された実行ファイルは、インテル以外のプロセッサーと、インテル® AVX2 命令をサポートしていないインテル® プロセッサー上では動作しません

インテル以外のプロセッサー上でコンパイルする場合:

-march=core-avx2 オプション (Linux* および macOS*) または /arch:CORE-AVX2 オプション (Windows*) に相当します。生成された実行ファイルは、インテル® AVX2 命令以上をサポートするインテル® プロセッサーおよびインテル以外のプロセッサーで動作します。インテル® AVX2 命令をサポートしていないランタイム・プロセッサーでは、ランタイムエラーが発生することがあります

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

コマンドラインでコンパイルする際の注意点

Visual C++ とインテル® C++ のデフォルト最適化オプションは異なります:

- VC++: /Od
- インテル® C++: /O2

インテル® C++ では、/Zi、/ZI、/Z7、/debug オプションを指定するとデフォルトが /Od となるため、デバッグ情報を生成しつつ最適化を有効にするには /O2 オプションを指定する必要があります:

- icl src.c
- icl src.c /Zi /O2
- icl src.c /Zi /Qip (または /Qipo)

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

内容

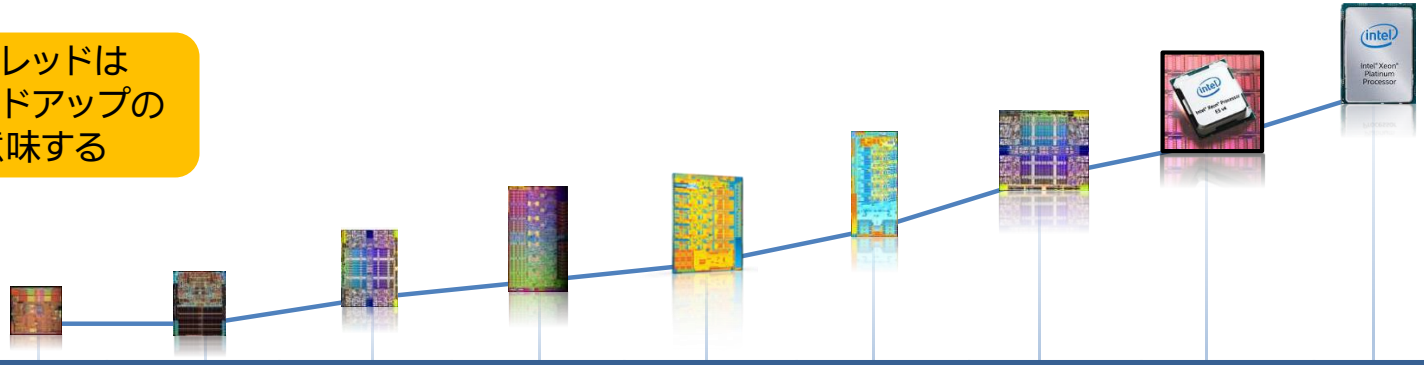
- ソースの移植に役立つ機能
- 利用可能なリソース
- 互換性制御
- コンパイラー・オプション
  - 命令セットと命令スケジュール
- **コンパイラー・プラグマ (アノテーションと明示的指示)**
- インテル® マイクロアーキテクチャー Skylake Server† 向けプログラミング

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

ハードウェアの変化はソフトウェア開発に影響する

コア数の増加とベクトルレジスター幅の拡大は、より多くのスレッドとさらなるパフォーマンスを意味する!
… しかし、それらのコアとレジスターを利用するようにソフトウェアを記述する必要がある

より多くのスレッドは
さらなるスピードアップの
可能性を意味する

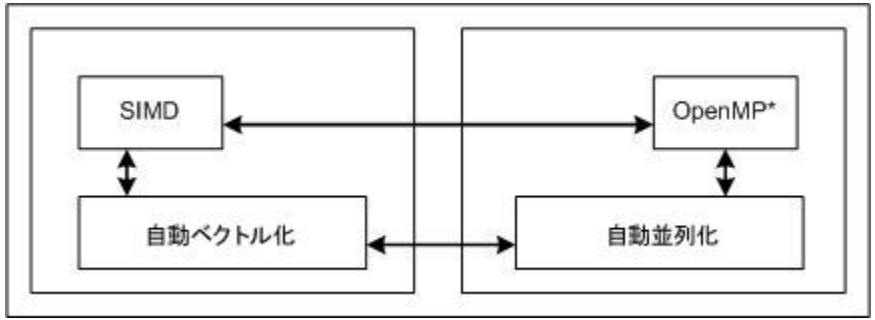


インテル® Xeon® プロセッサー	64 ビット	5100 番台	5500 番台	5600 番台	E5-2600	E5-2600 v2	E5-2600 v3	E5-2600 v4	Platinum 8180
コア	1	2	4	6	8	12	18	22	28
スレッド	2	2	8	12	16	24	36	44	56
SIMD 幅	128	128	128	128	256	256	256	256	512

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

ユーザー指示と SIMD ベクトル化

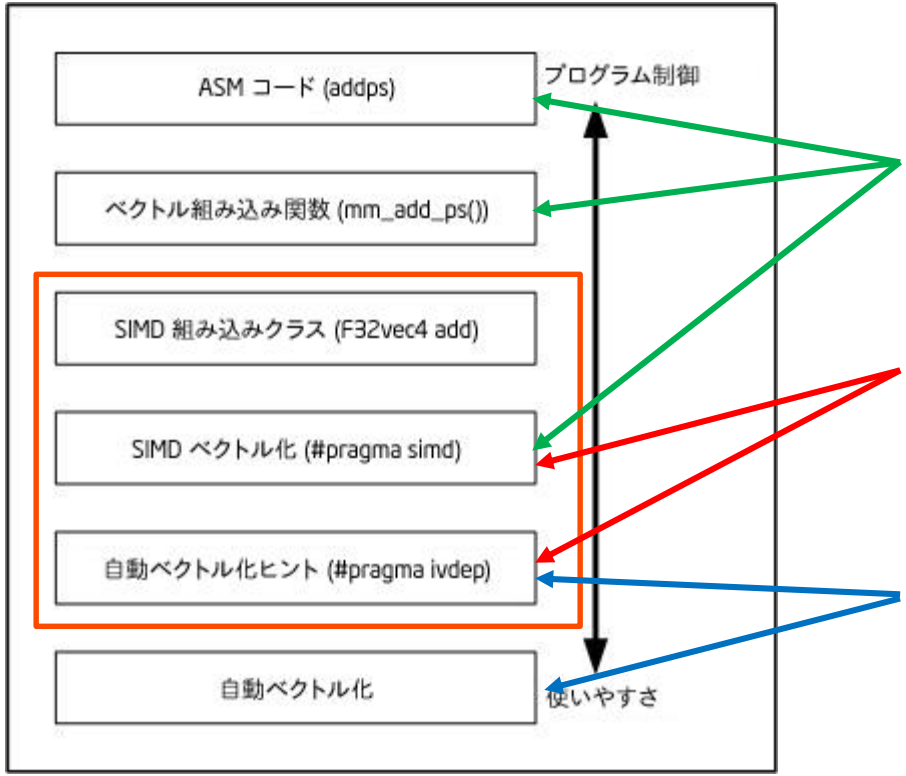
ユーザー指示または SIMD ベクトル化は、OpenMP* 並列化が自動並列化を補足するように、自動ベクトル化を補足します。下記の図でこの関係を示します。ユーザー指示によるベクトル化は SIMD (Single-Instruction, Multiple-Data) 機能として実装され、SIMD ベクトル化と呼ばれます



注: SIMD ベクトル化機能は、インテル製マイクロプロセッサーおよび互換マイクロプロセッサーの両方で利用可能です。ベクトル化により呼び出されるライブラリー・ルーチンは、互換マイクロプロセッサーよりもインテル製マイクロプロセッサーにおいてより優れたパフォーマンスが得られる可能性があります。また、ベクトル化は、/arch (Windows*)、-m (Linux* および macOS*)、[Q]x などの特定のオプションによる影響を受けます。

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

ベクトル化コードを生成するアプローチ



ソースをベクトル化しても安全なことは、プログラマーが責任を持ちます

容易に導入可能

ベクトル化を適用するかどうかは、コンパイラーが判断

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

自動ベクトル化が適用されない例

```
void add_floats(float *a, float *b, float *c, float *d, float *e, int n)
{
int i;
     for (i=0; i<n; i++){
          a[i] = a[i] + b[i] + c[i] + d[i] + e[i];
     }
}
```

上記のコードはデフォルトのコンパイル **icl add_float.c /QxAVX2 /c** ではベクトル化されません: ベクトル化するには？

1. -ipo (/Qipo) オプションを使用する
2. restrict キーワードを使用する
3. #pragma ivdep を使用する
4. #pragma omp simd を使用する

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

-opt-report (/Qopt-report) で確認

```
コンパイラー・オプション: /QxAVX2 /Qopt-report5 /Qopt-report-phase:vec /c

最適化レポート開始: add_floats(float *, float *, float *, float *, float *, int)

    レポート: ベクトルの最適化 [vec]


ループの開始 C:\Users\kiyo\Desktop\2019\FTF_expert\code\vec\add_float.c(3,6)
   リマーク #15344: ループ はベクトル化されませんでした: ベクトル依存関係がベクトル化を妨げています。
   リマーク #15346: ベクトル依存関係: FLOW の依存関係が a[i] (4:11) と b[i] (4:11) の間に仮定されました。
   リマーク #15346: ベクトル依存関係: ANTI の依存関係が b[i] (4:11) と a[i] (4:11) の間に仮定されました。
ループの終了
```

コンパイラーは依存関係を否定できません

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

restrict キーワードで依存関係を否定

```
void add_floats(float *restrict a, float *restrict b, float *c, float *d,
float *e, int n) {
int i;
    #pragma vector aligned
     for (i=0; i<n; i++){
          a[i] = a[i] + b[i] + c[i] + d[i] + e[i];
     }
}
```

上記のコードは、
icl add_float.c /QxAVX2 /c /Qrestrict
で依存関係を否定できます: ベクトル化されるか？

```
ループの開始 C:¥add_restrict.c(3,6)
  リマーク #15388: ベクトル化のサポート: 参照 a[i] にアラインされたアクセスが含まれています。 [ C:¥add_restrict.c(4,11) ]
……
  リマーク #15389: ベクトル化のサポート: 参照 e[i] にアラインされていないアクセスが含まれています。 [ C:\add_restrict.c(4,46) ]
  リマーク #15381: ベクトル化のサポート: ループ本体内でアラインされていないアクセスが使用されました。
  リマーク #15305: ベクトル化のサポート: ベクトル長 8
  リマーク #15399: ベクトル化のサポート: アンロールファクターが 2 に設定されます。
  リマーク #15309: ベクトル化のサポート: 正規化されたベクトル化のオーバーヘッド 0.733
  リマーク #15300: ループがベクトル化されました。
  リマーク #15442: すべてのループは剰余ループとして実行されます。
  リマーク #15448: マスクなしアライン・ユニット・ストライド・ロード: 1
  リマーク #15449: マスクなしアライン・ユニット・ストライド・ストア: 1
  リマーク #15450: マスクなし非アライン・ユニット・ストライド・ロード: 4
  リマーク #15475: --- ベクトルのコストサマリー開始 ---
  リマーク #15476: スカラーのコスト: 17
  リマーク #15477: ベクトルのコスト: 1.870
  リマーク #15478: スピードアップの期待値: 6.650
  リマーク #15488: --- ベクトルのコストサマリー終了 ---
ループの終了
```

コンパイラーは、アライメントを解決できません

© 2019 iSUS

ivdep を使用

```
void add_floats(float *a, float *b, float *c, float *d, float *e,
int n) {
int i;
#pragma vector aligned
#pragma ivdep
     for (i=0; i<n; i++){
          a[i] = a[i] + b[i] + c[i] + d[i] + e[i];
     }
}
```

#pragma ivdep、#pragma vector aligned や
!DIR$ IVDEP、!DIR$ VECTOR ALIGNED でベクトル化の可能性をコンパイラーに通知

ivdep や vector はあくまでコンパラ―に対するヒントであるため、
すべてのケースでベクトル化を有効にできるとは限りません

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

omp simd を使用

```
void add_floats(float *a, float *b, float *c, float *d, float *e,
int n) {
int i;
#pragma omp simd aligned(a,b,c,d,e)
     for (i=0; i<n; i++){
          a[i] = a[i] + b[i] + c[i] + d[i] + e[i];
     }
}
```

コンパイラーに次のことを断定させる:

- a[]、b[]、c[]、d[]、e[] はエイリアスまたはオーバーラップしない
- a[]、b[]、c[]、d[]、e[] はアライメントされている
- 演算は可換である
- ターゲットのベクトル演算はスカラー演算よりも高速
- -qno-openmp-simd (/Qopenmp-simd-) で無効化できます

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

なぜ SIMD 拡張? OpenMP* 4.0 以前

コンパイラー・ベンダー固有の拡張機能をサポート

- プログラミング・モデル (例えば、インテル® Cilk™ Plus)
- コンパイラー・プラグマ (例えば、#pragma vector)
- 低レベルの構文 (例えば、__mm_add_pd() 組込み関数)

```
#pragma omp parallel for
#pragma vector always
#pragma ivdep
for (int i = 0; i < N; i++) {
    a[i] = b[i] + ...;
}
```

コンパイラーが "正しい" ことをするのを信頼する必要がある

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

ベクトル化に影響するプログラムの要素

ループ伝搬依存

```
DO I = 2, N
   A(I) = A(I-1) + B(I)
ENDDO
```

関数呼び出し

```
for (i = 1; i < nx; i++) {
  x = x0 + i * h;
  sumx = sumx + func(x, y, xp);
}
```

ポインター・エイリアシング

```
void scale(int *a, int *b)
{
   for (int i = 0; i < 1000; i++)
       b[i] = z * a[i];
}
```

不明なループカウント

```
struct _x { int d; int bound; };
void doit(int *a, struct _x *x)
{
   for(int i = 0; i < x->bound; i++)
    a[i] = 0;
}
```

間接メモリーアクセス

```
DO i=1, N
  A(B(i)) = A(B(i)) + C(i)*D(i)
ENDDO
```

外部ループ

```
DO I = 1, MAX
  DO J = I, MAX
    D(I,J) = D(I,J) + 1;
  ENDDO
ENDDO
```


© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

自動ベクトル化: シリアル･セマンティクスによる制限

```
for(i = 0; i < *p; i++) {
  A[i] = B[i] * C[i];
  sum = sum + A[i];
}
```

コンパイラーは以下をチェックする:

- *p はループ不変か?
- A[]、B[]、C[] はオーバーラップしているか?
- sum は、B[] および/または C[] とエイリアスされているか?
- 演算操作の順番は重要か?
- ターゲット上のベクトル演算はスカラー演算よりも高速であるか? (ヒューリスティックの評価)

自動ベクトル化は言語規則によって制限される: 意図することを表現できない

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

SIMD プラグマ/ディレクティブによる明示的なベクトル・プログラミング

```
#pragma omp simd reduction(+:sum)
for(i = 0; i < *p; i++) {
  A[i] = B[i] * C[i];
  sum = sum + A[i];
}
```

プログラマーの主張:

- *p はループ不変
- A[] は、B[] および C[] とオーバーラップしない
- sum は、B[] および C[] とエイリアスされていない
- sum はリダクションされる
- コンパイラーが効率良いベクトル化のため順番を入れ替えることを許容する
- ヒューリスティックの評価が利点をもたらさなくても、ベクトル化されたコードを生成する

明示的ベクトル・プログラミングにより何を意図するかを表現できる!

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

内容

- ソースの移植に役立つ機能
- 利用可能なリソース
- 互換性制御
- コンパイラー・オプション
  - 命令セットと命令スケジュール
- コンパイラー・プラグマ (アノテーションと明示的指示)
- **インテル® マイクロアーキテクチャー Skylake Server† 向けプログラミング**

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

インテル® AVX-512 向けの最適化: 推奨オプション

適切なオプションを指定してコンパイラーによる最適化を利用

Linux*	Windows*	説明
-xCORE-AVX512	/QxCORE-AVX512	インテル® Xeon® スケーラブル・プロセッサー向けに最適化 (インテル® AVX-512 を含む)
-xCOMMON-AVX512	/QxCOMMON-AVX512	上記のオプションで期待したスピードアップが得られない場合に使用するオプション
-fma	/Qfma	FMA (Fused Multiply-Add) 命令を有効にする (注: 丸めに影響!)
-O2	/O2	速度を最適化 (デフォルトで有効)
-g	/Zi	パフォーマンス・プロファイル・ツールで使用するデバッグ情報を生成

インテル® マス・カーネル・ライブラリー (インテル® MKL) などの最適化されたライブラリーを使用

線形代数
- BLAS
- LAPACK
- ScaLAPACK
- スパース BLAS
- スパースソルバー
- 反復法
- PARDISO
- クラスター・スパース・ソルバー

高速フーリエ変換
- 多次元
- FFTW インターフェイス
- クラスター FFT

ベクトル演算
- 三角関数
- 双曲線
- 指数
- 対数
- 累乗
- 累乗根
- ベクトル RNG

サマリー統計
- 尖度
- 変化係数
- 順序統計量
- 最小/最大
- 分散/共分散

ディープ・ニューラル・ネットワーク
- 畳み込み
- プーリング
- 正規化
- ReLU
- ソフトマックス

その他
- スプライン
- 補間
- 信頼領域
- 高速ポアソンソルバー

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

インテル® マイクロアーキテクチャー Skylake Server† 向けプログラミング

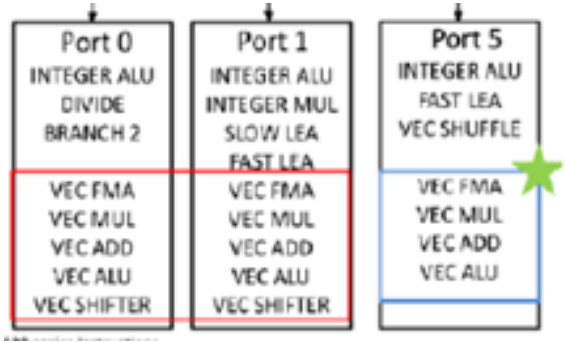
製品によって異なる内部ユニット

† 開発コード名

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

FMA ユニット

Skylake† マイクロアーキテクチャー・ベースのプロセッサーには、2 つのインテル® AVX-512 FMA ユニット (ポート 0 と 5) を備えるものがある一方、同じアーキテクチャー・ベースのプロセッサーでも単一のインテル® AVX-512 FMA ユニット (ポート 0) を持つものがあります



2 つの FMA ユニットで実行されることを想定して最適化されたコードは、単一 FMA ユニット上での実行には最適でない可能性があります

コンパイラーには FMA ユニット数を制御するオプションは用意されていません

† 開発コード名

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

インテル® Xeon® スケーラブル・プロセッサー

各ファミリーの特長

	インテル® Xeon® Bronze プロセッサー (3100シリーズ)	インテル® Xeon® Silver プロセッサー (4100シリーズ)	インテル® Xeon® Gold プロセッサー (5100シリーズ)	インテル® Xeon® Gold プロセッサー (6100シリーズ)	インテル® Xeon® Platinum プロセッサー (8100シリーズ)
汎用性の高いパフォーマンスとセキュリティー					
サポートされる最大コア数	8 コア	12 コア	14 コア	22 コア	28 コア
サポートされる最大周波数	1.70 GHz (8C/85W)	2.20 GHz (10C/85W)	3.60 GHz (4C/105W)	3.40 GHz (6C/115W)	3.60 GHz (4C/105W)
サポートされるCPUソケット数	最大 2	最大 2	最大 4	最大 4	最大 8
インテル® ウルトラ・パス・インターコネクト (インテル® UPI)	2	2	2	3	3
インテル® UPIの速度	9.6 GT/s	9.6 GT/s	10.4 GT/s	10.4 GT/s	10.4 GT/s
インテル® アドバンスト・ベクトル・エクステンション512 (インテル® AVX-512)	1 FMA	1 FMA	1 FMA	2 FMA	2 FMA
サポートされるメモリー速度(DDR4)	2133 MHz	2400 MHz	2400 MHz	2666 MHz	2666 MHz
サポートされる最大メモリー容量 (1ソケット当たり)	768 GB	768 GB	768 GB	768 GB / 1.5 TB	768 GB / 1.5 TB

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

FMA ユニット数を判定する

2 つの FMA ユニットを備えるプロセッサーは、FMA と shuffle 命令を同時に実行できますが、単一ユニットのプロセッサーは同時実行できません

- FMA 命令のみを実行する関数を実行
- FMA 命令と shuffle 命令を実行する関数実行
- 2 つの実行時間を基にプロセッサーが 1 つまたは 2 つの FMA ユニットを搭載するかを特定します

詳細は、『インテル® 64 および IA-32 アーキテクチャー最適化リファレンス・マニュアル』(248966-038) の 15.20 節「単一 FMA ユニットを備える場合」をご覧ください:
https://www.isus.jp/technical-document/

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

FMA ユニット数を確認

Windows* - インテル® Core™ i7-7800X X シリーズ・プロセッサー

```
$ icc -fasm-blocks FMA_count.c
$ ls
a.out  FMA_count.c
$ ./a.out
2 FMA server
$

processor       : 103
vendor_id       : GenuineIntel
cpu family      : 6
model           : 85
model name      : Intel(R) Xeon(R) Platinum 8170 CPU @ 2.10GHz
stepping        : 4
microcode       : 0x2000016
cpu MHz         : 1136.871
cache size      : 36608 KB
```

Intel Compiler 18.0 Update 1 Intel(R) 64 Visual Studio 2017

```
> icl fma_count.c
インテル(R) 64 対応インテル(R) C++ コンパイラー (インテル(R) 64 対応アプリケーシ
ョン用) バージョン 18.0.1.151 ビルド 20171018
(C) 1985-2017 Intel Corporation.  無断での引用、転載を禁じます。

fma_count.c
Microsoft (R) Incremental Linker Version 14.11.25547.0
Copyright (C) Microsoft Corporation.  All rights reserved.

-out:fma_count.exe
fma_count.obj

> fma_count
1 FMA server
```

Linux* - インテル® Xeon® Platinum 8170 プロセッサー

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

命令グループ (2 FMA)

- グループ A の命令: vadd*; vfmadd*; vfnmsub*; vfnmadd*; vfnmsub*; vmax*; vmin*; vmul*; vscalef*; vsub*; vcvt*; vgetexp*; vfixupimm*; vrange*; vgetmant*; vreduce*; vcmp*; vcomi*; vdpp*; vhadd*; vhsub*; vrndscale*; vround*
- グループ B の命令: vpmaddubsw; vpmaddwd; vpmuldq; vpmulhrsw; vpmulhuw; vpmulhw; vpmullw; vpmuludq

すべての命令のソースが FMA ユニットから供給される場合、FMA ユニットは高速バイパスをサポートします。グループ A はポート 0 とポート 5 で 4 サイクルのレイテンシー、グループ B はポート 0 とポート 5 で 5 サイクルのレイテンシーです

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

インテル® マイクロアーキテクチャー Skylake Server † 向けプログラミング

異なるキャッシュ・アーキテクチャー (インクルーシブと、非インクルーシブ)

† 開発コード名

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

キャッシュ最適化

配列 a, b, c はキャッシュのどの階層に収まるか?

2M バイト
256K バイト
32K バイト

```
for(j=1;j<n;j++){
      f=0;
      for(i=1;i<n;i++){
            f+=a[i]*b[i];
      }
      c[j]+=f;
}
```

1.375M バイト
1M バイト
32K バイト

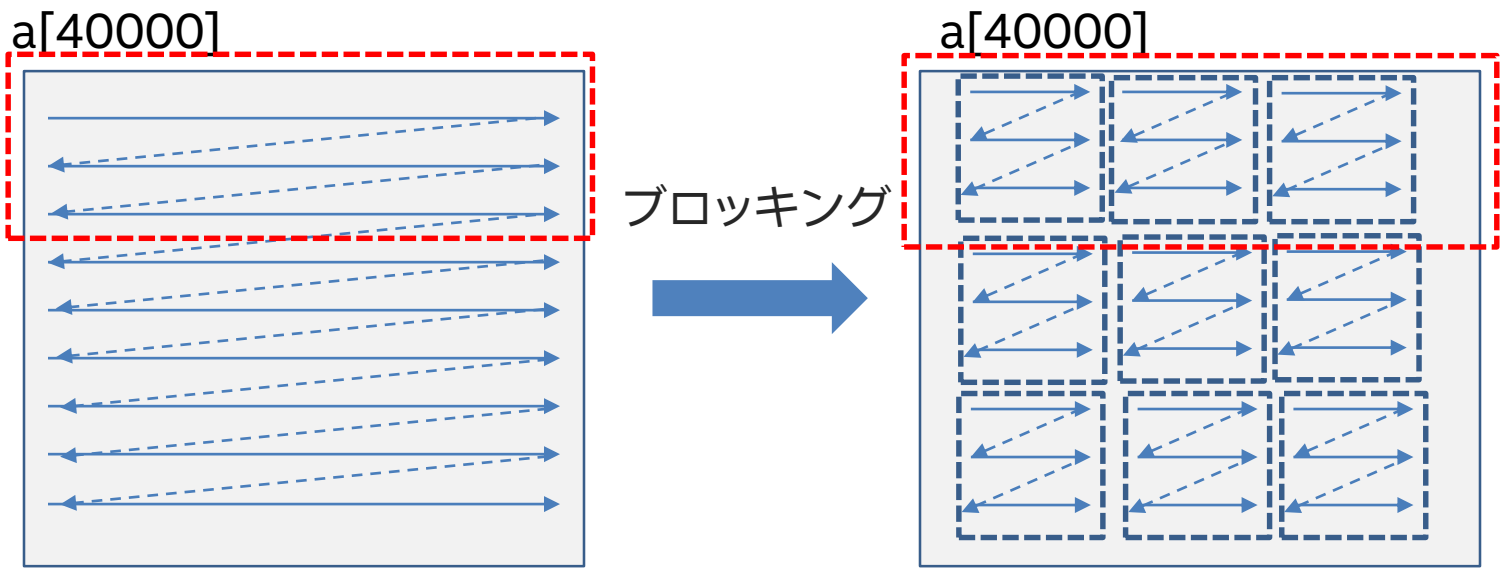
```
double a[100], b[100], c[100]
```

```
double a[40000], b[40000], c[40000]
```


© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

キャッシュ最適化 – 従来のプロセッサー

double a[40000], b[40000], c[40000] は、L2 キャッシュに収まらない



© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

ブロッキングの例

```
for(jj=1;jj<n/FACTOR+1;jj++){
  for(ii=1;ii<n/FACTOR+1;ii++){
    for(j=(jj-1)*FACTOR+1;j<min(jj*FACTOR,n);j++){
      f=0;
      for(i=(ii-1)*FACTOR+1;i<min(ii*FACTOR,n);i++){
        f=f+a[i]*b[i];
      }
      c[j]=c[j]+f;
    }
  }
}
```


© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

ブロッキングとパフォーマンス

プロセッサー	キャッシュ	ブロッキングなし	ブロッキングあり
インテル® Core™ i7-4790K プロセッサー(4.00GHz/4.40GHz)	8M/256K	0.47 秒	0.23 秒
インテル® Core™ i7-7800X X シリーズ・プロセッサー (3.50GHz/4.00GHz)	8.25M/1M	0.27 秒	0.29 秒
インテル® Core™ i7-5820K プロセッサー (3.30GHz/3.60GHz)	15M/256K	0.66 秒	0.28 秒
インテル® Xeon® プロセッサー E5-2699 v4 (2.20GHz/3.60GHz)	55M/256K	0.76 秒	0.30 秒

/QxCORE-AVX2 で生成した同じバイナリーを実行

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

インテル® マイクロアーキテクチャー Skylake Server† 向けプログラミング

命令セットと電力管理

† 開発コード名

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

Skylake Server† の電力管理

- Skylake Server† マイクロアーキテクチャーは、それぞれのコアの実行周波数を動的に選択します。選択される周波数は、命令タイプ、命令幅、および一定時間に実行されるベクトル命令数などの組み合わせに依存します。プロセッサーはまた、同様の特性を持つコアの数を考慮します

- Broadwell† マイクロアーキテクチャー・ベースのインテル® Xeon® プロセッサーも同じように振る舞いますが、256 ビット・ベクトル命令のみをサポートするため影響は軽度です。Skylake Server† マイクロアーキテクチャーがサポートするインテル® AVX-512 命令は、潜在的にインテル® AVX2 命令よりも多くの電流と電力を必要とします

- プロセッサーは、必要に応じて動的に最大周波数をより高いか低いレベルに調整します。そのため、プログラムが実行されている間の最大周波数は異なる可能性があります

† 開発コード名

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

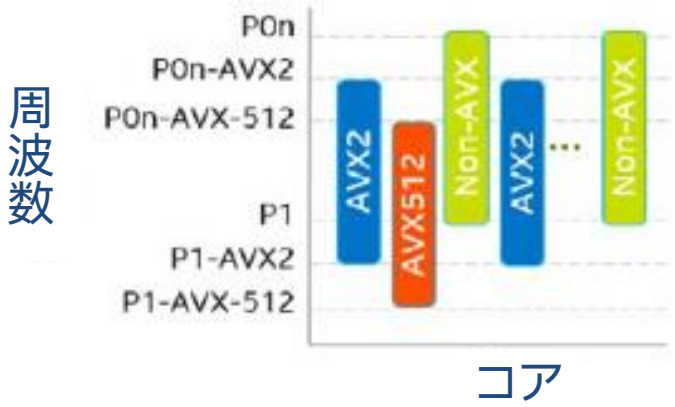
インテル® ターボ・ブースト・テクノロジーの最大コア周波数レベル

レベル	カテゴリー	周波数レベル	最大周波数 (POn)	命令タイプ
0	軽量なインテル® AVX2 命令	最も高い	最大	スカラー、インテル® AVX (128 ビット)、インテル® SSE、インテル® AVX2 (FP または INT MUL/FMA なし)
1	重いインテル® AVX 2 命令 + 軽量なインテル® AVX-512 命令	中間	最大 インテル® AVX2	インテル® AVX2 FP + INT MUL/FMA、インテル® AVX-512 (FP または MUL/FMA なし)
2	重いインテル® AVX-512 命令	最低	最大 インテル® AVX-512	インテル® AVX-512 FP + INT MUL/FMA

- 最大周波数 (POn) は、カテゴリー内のコア数に応じた周波数の配列で表されます
- 多くのコアがある時点でカテゴリーに含まれると、最大周波数は低くなります

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

ワークロードの混在と周波数



- それぞれのコアの周波数はワークロードの要求に基づいて独立して決定されます
- 新しいレベルへの移行には最大 500 μ秒かかります
- 高い周波数レベルに戻る前には、およそ 2 ms のタイマーが適用されます

次のパフォーマンス・モニタリング・イベントは、3 つの周波数レベルにおいて費やされるサイクル数を決定するために使用されます:

- CORE_POWER.LVL0_TURBO_LICENSE: 最大周波数が P0n でコアが動作したコアサイクル
- CORE_POWER.LVL1_TURBO_LICENSE: 最大周波数が P0n-AVX2 でコアが動作したコアサイクル
- CORE_POWER.LVL2_TURBO_LICENSE: 最大周波数が P0n-AVX-512 でコアが動作したコアサイクル

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

インテル製品　ユーザー別　サポート　intel　Japan (日本語)

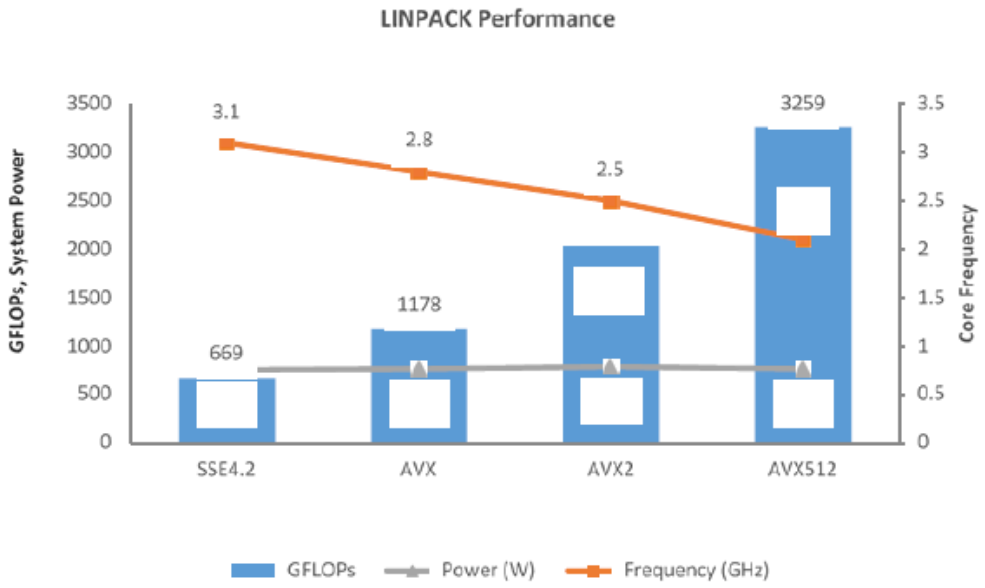
サポートホーム › 製品仕様 › プロセッサー　製品仕様の検索

製品名	ステータス	発売日	コアの数	ターボ・ブースト利用時の最大周波数	プロセッサー・ベース動作周波数	キャッシュ	比較 すべて | なし
Intel® Core™ i9-7980XE Extreme Edition Processor	Launched	Q3'17	18	4.20 GHz	2.60 GHz	24.75 MB	
Intel® Core™ i9-7960X X-series Processor	Launched	Q3'17	16	4.20 GHz	2.80 GHz	22 MB	
Intel® Core™ i9-7940X X-series Processor	Launched	Q3'17	14	4.30 GHz	3.10 GHz	19.25 MB	
Intel® Core™ i9-7920X X-series Processor	Launched	Q3'17	12	4.30 GHz	2.90 GHz	16.5 MB L3	
インテル® Core™ i9-7900X X シリーズ・プロセッサー	Launched	Q2'17	10	4.30 GHz	3.30 GHz	13.75 MB L3	
インテル® Core™ i7-7820X X シリーズ・プロセッサー	Launched	Q2'17	8	4.30 GHz	3.60 GHz	11 MB L3	
インテル® Core™ i7-7800X X シリーズ・プロセッサー	Launched	Q2'17	6	4.00 GHz	3.50 GHz	8.25 MB L3	
インテル® Core™ i7-7740X X シリーズ・プロセッサー	Launched	Q2'17	4	4.50 GHz	4.30 GHz	8 MB	

https://ark.intel.com/ja/products/series/125191/Intel-Xeon-Scalable-Processors

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

ワークロードとパフォーマンス



- インテル® AVX-512 命令を実行するワークロードが、命令全体に占める割合が大きいと、インテル® AVX2 命令に比べパフォーマンス・ゲインは高くなりますが、動作周波数は低くなる可能性があります

- 例えば、高い比率で重いインテル® AVX-512 命令を最大周波数で要求するディープラーニングのワークロードは、インテル® AVX2 をターゲットとする同じワークロードに対し、1.3 倍から 1.5 倍のパフォーマンスを向上できます (両者とも Skylake Server† マイクロアーキテクチャー上で実行)

† 開発コード名

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

ワークロードと推奨コンパイラー・オプション

最適なコンパイラー・オプションを特定するため、次に示す命令セットオプションでアプリケーションをビルドして、最も高いパフォーマンスを示す命令セットを選択します:

- -xCORE-AVX2 -mtune=skylake-avx512 (Linux* および macOS*)

  /QxCORE-AVX2 /tune=skylake-avx512 (Windows*)

- -xCORE-AVX512 -qopt-zmm-usage=low (Linux* および macOS*)

  /QxCORE-AVX512 /Qopt-zmm-usage:low (Windows*)

- -xCORE-AVX512 -qopt-zmm-usage=high (Linux* および macOS*)

  /QxCORE-AVX512 /Qopt-zmm-usage:high (Windows*)

GCC コンパイラーでも、同様の low/high zmm を指定するオプションがサポートされています

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

メモリー依存のワークロードの例

```
Intel Compiler 18.0 Update 1 Intel(R) 64 Visual Studio 2017
> matavx2

Using clock() for wall clock time
(matavx2)
Problem size: c[1024][1024] += a[1024][1024] * b[1024][1024]

Matmul kernel wall clock time = 0.510000 sec.

Result (524800.000000)

> matavx512low

Using clock() for wall clock time
(matavx512low)
Problem size: c[1024][1024] += a[1024][1024] * b[1024][1024]

Matmul kernel wall clock time = 0.494000 sec.

Result (524800.000000)

> matavx512high

Using clock() for wall clock time
(matavx512high)
Problem size: c[1024][1024] += a[1024][1024] * b[1024][1024]

Matmul kernel wall clock time = 0.479000 sec.

Result (524800.000000)
```

/QxCORE-AVX2 /tune=skylake-avx512 (Windows*)

/QxCORE-AVX512 /Qopt-zmm-usage:low (Windows*)

/QxCORE-AVX512 /Qopt-zmm-usage:high (Windows*)

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

計算集約型ワークロードのテスト例

Intel Compiler 18.0 Update 1 Intel(R) 64 Visual Studio 2017

シリアル

```
> primeavx2
(primeavx2) 1から50000000までの範囲の素数の数は 3001135個。計算時間12.67秒
> primeavx512low
(primeavx512low) 1から50000000までの範囲の素数の数は 3001135個。計算時間13.90秒
> primeavx512high
(primeavx512high) 1から50000000までの範囲の素数の数は 3001135個。計算時間7.65秒
```

Intel Compiler 18.0 Update 1 Intel(R) 64 Visual Studio 2017

OpenMP* 並列化

```
> primeavx2
(primeavx2) 1から50000000までの範囲の素数の数は 3001135個。計算時間2.44秒
> primeavx512low
(primeavx512low) 1から50000000までの範囲の素数の数は 3001135個。計算時間2.39秒
> primeavx512high
(primeavx512high) 1から50000000までの範囲の素数の数は 3001135個。計算時間1.75秒
```

選択Intel Compiler 18.0 Update 1 Intel(R) 64 Visual Studio 2017

```
> piavx2
Data type double
Pi = 3.141593 Time = 0.095000

> piavx512low
Data type double
Pi = 3.141593 Time = 0.093000

> piavx512high
Data type double
Pi = 3.141593 Time = 0.058000
```


© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

まとめ: アーキテクチャー最適化の推奨事項

- **推奨事項**: アプリケーションの共有およびプライベート・データのサイズを、小さな非インクルーシブ L3 キャッシュと大きな L2 キャッシュに合わせて再度バランスを取ります
- **推奨事項**: L2 キャッシュがアプリケーションの帯域幅要件を満たせば、Skylake Server† マイクロアーキテクチャーでは L2 へのブロック化を検討します
- **推奨事項**: データを共有しない場合、アプリケーションはコアあたりのキャッシュ容量を L3 キャッシュだけでなく L2 と L3 キャッシュサイズの合計であると見なすべきです
- **推奨事項**: データは 64 バイト境界でアライメントします

† 開発コード名

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

まとめ:インテル® AVX-512 向けの推奨事項

- **推奨事項**: 実行ターゲットが固定である場合、/Qx (-x) オプションで固有命令セットを生成します
- **推奨事項**: /Qopt-zmm-usage オプションで、ZMM レジスターの使用を制御し、パフォーマンスを計測します。必要に応じて OpenMP* の simdlen、safelen 節を使用します
- **推奨事項**: 実行ターゲットが固定でない場合、/Qax (-ax) オプションを使用してサポートされる最も低い命令セットに合わせてプロセッサー・ディスパッチを行います
- **推奨事項**: FMA ユニット数に対応したプログラミングを行う場合、サポートされる FMA ユニット数を手動で判定します

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

組込み関数を使用するガイドライン

- 静的および動的に割り当てられているバッファーを 64 バイトにアライメントします
- 定数向けの補助バッファーサイズを倍にします
- "__mm256_組込み関数名" のプリフィクスを "__mm512_" に変更します
- 変数のデータ型名を __m256 から __m512 へ変更します
- ループの反復回数を半分にします (倍のストライド長に)

アセンブリーを使用するガイドライン

- 静的および動的に割り当てられているバッファーを 64 バイトにアライメントします
- 必要であれば補足のバッファーサイズを 2 倍にします
- 命令ニーモニックの前に “v” プリフィクスを追加します
- レジスター名を ymm から zmm へ変更します
- ループの反復回数を半分にします (またはストライド長を倍にします)

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

インテル® コンパイラーがサポートする高度なコンパイラー・オプションやプラグマやディレクティブを活用してください…

© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

iSUS