インテル・アーキテクチャの性能最適化

池井　満

e-Marketing
Intel Japan K.K.
March 20, 2002

intel

ITANIUM. XEON.

目次

- 単一プロセッサ性能の概要
- アイテニアム™アーキテクチャ(ITA)
- 性能最適化ツール
- インター・プロシージャー 最適化(IPO)
- プロファイル ガイデッド 最適化(PGO)

intel

ITANIUM. XEON.

プロセッサ性能

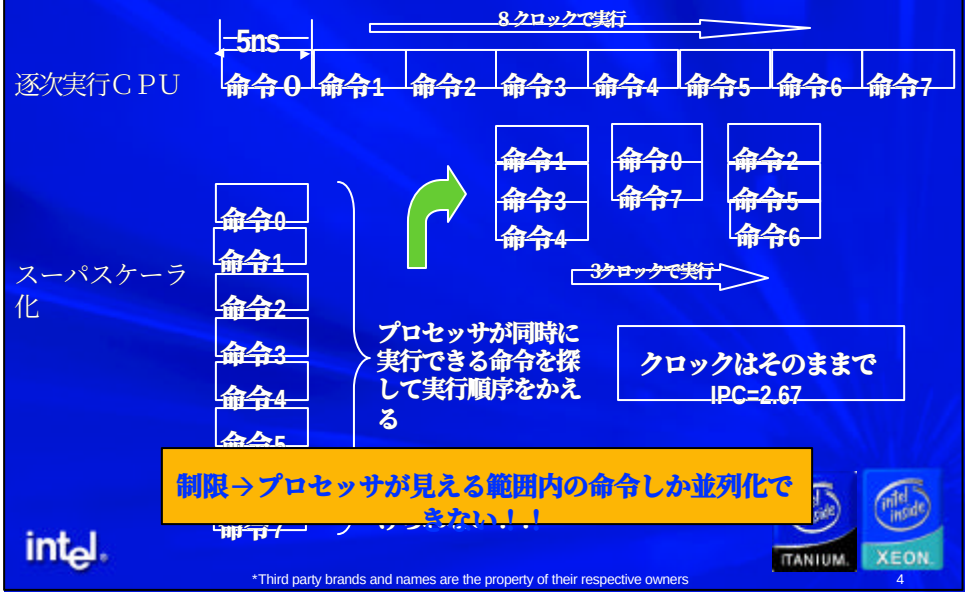
単一プロセッサ性能の向上

- 動作周波数の高速化　f
  - パイプライン（GHzオーダ）
- 命令レベル並列性（ILP）　w
  - OOOスーパスケーラ
  - EPIC（明示的並列命令計算）
- スレッドレベル並列性（TLP）　n
  - ハイパースレッド

性能　＝　f　×　w　×　n

プロセッサ性能

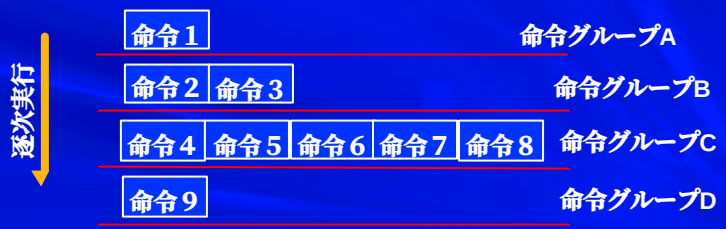
スーパスケーラによる並列化



プロセッサ性能

EPICによる並列化

- コンパイラがプログラム全体から並列に実行できる命令列を選択する
- 並列（同時）に実行できる命令は同じ命令グループとする



intel

ITANIUM. XEON.

ITAの特長

命令グループ

- 同時に実行可能な1つ以上の命令の固まり（命令数の制限無し）
- アイテニアム™の論理的な命令実行単位
- ブレーク・コード（;;）をアセンブリ・コード中に記述することによって、命令グループの境界を指示する、もしくは実行時に分岐によってターミネートする
- *例：*

```
add     r31 = r5, r6
mov     r4 = r31
add     r2 = r8, r9
mov     r7 = r2
mov     r15 = r27
mov     r16 = r28
mov     r17 = r29
add     r3 = r11, r20
mov     r10 = r3
```

intel

ITANIUM. XEON.

ITAの特長

命令グループ

- 同時に実行可能な1つ以上の命令の固まり 命令数の制限無し)
- アイテニアム™の論理的な命令実行単位
- ブレーク・コード (;;) をアセンブリ・コード中に記述することによって、命令グループの境界を指示する、もしくは実行時に分岐によってターミネートする
- *例：*

```
add     r31 = r5, r6 ;;
mov     r4 = r31
add     r2 = r8, r9 ;;
mov     r7 = r2
mov     r15 = r27
mov     r16 = r28
mov     r17 = r29
add     r3 = r11, r20 ;;
mov     r10 = r3
```

命令グループ A
命令グループ B
命令グループ C
命令グループ D

intel

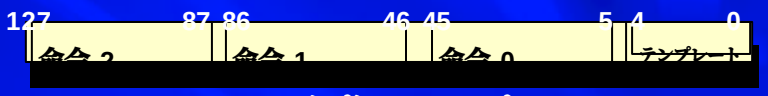
ITANIUM. XEON.

*Third party brands and names are the property of their respective owners

7

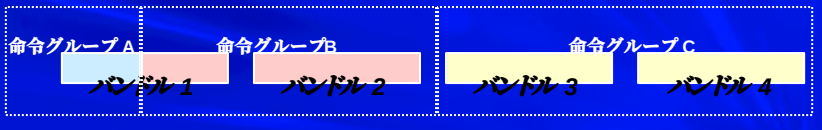
ITAの特長

命令バンドル



バンドルフォーマット

- バンドル :アイテニアム™の物理的処理単位
  - 3つの命令スロット(各41-ビット)とテンプレートフィールド(5ビット)
  - 命令グループは幾つかのバンドルにまたがることができる
  - *例:*



柔軟性のある発行と並列実行

intel

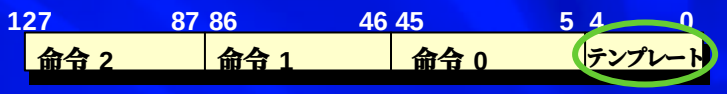
ITANIUM. XEON.

*Third party brands and names are the property of their respective owners

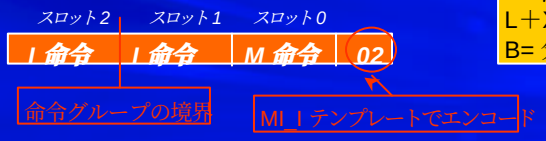
8

ITAの特長

命令テンプレート



- テンプレート
  - 各命令スロットの実行ユニットタイプを指定
    - MII, MLX, MMI, MFI, MMF, MIB, MBB, BBB, MMB, MFB
  - バンドル間やバンドル内の依存性を指定
    - MI_I, M_MI
    - MII_, MLX_, MMI_, 他
  - *例:*



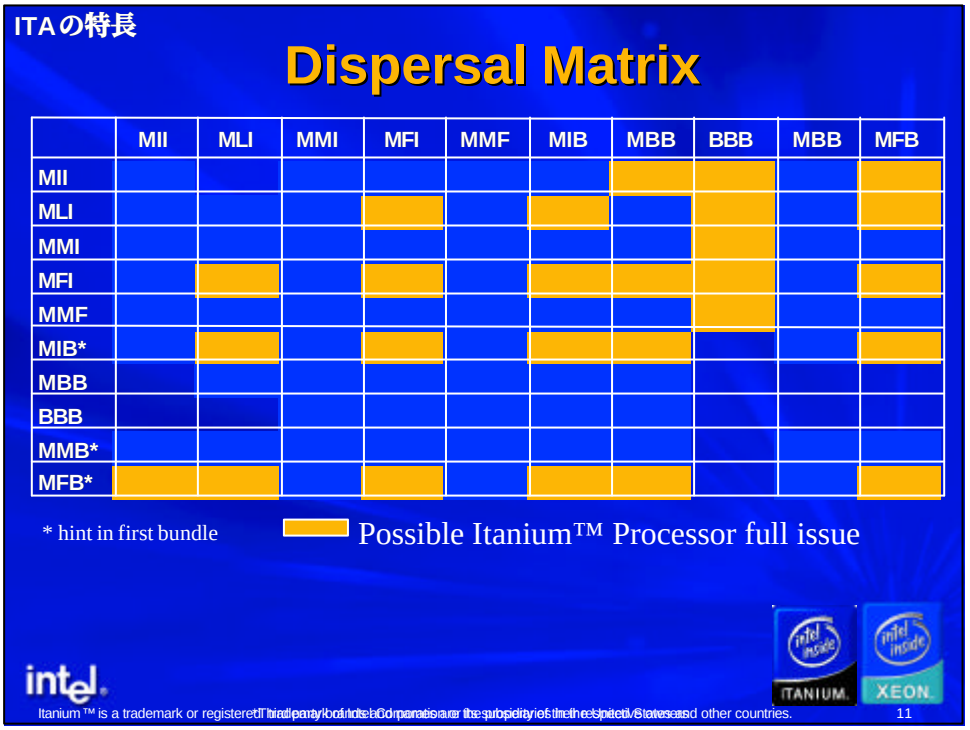
M= メモリ
I= 整数
A= メモリ/整数
F= 浮動小数点
L＋X= long即値
B= 分岐

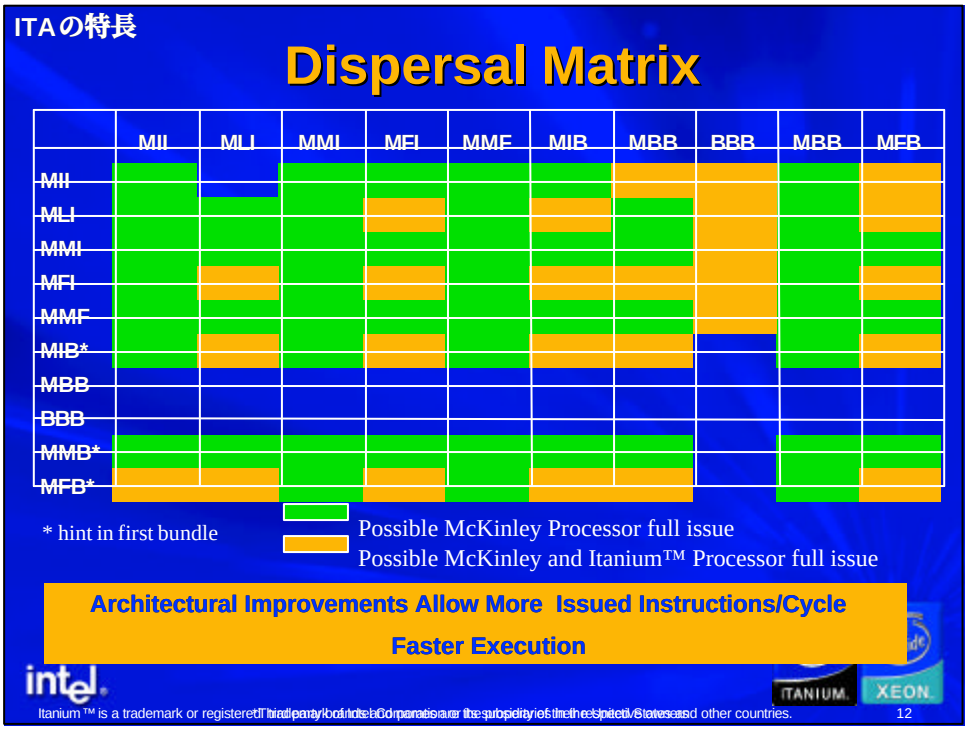
ITAの特長

命令発行 :分散

- ストップビットにより依存性チェックは必要ない
- テンプレートはルーティングを簡単にする
- 6つの命令節から最初に利用できる9つの発行ポートへの分散化
  - ストップビットの検出、リソースの超過要求またはコンフリクトまで続ける

簡単なハードウエアで高度な並列実行を実現

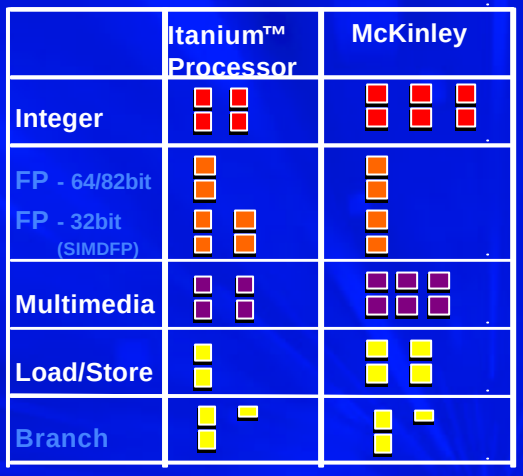

ITAの特長
Dispersal Matrix
MII MLI MMI MFI MMF MIB MBB BBB MBB MFB
MII
MLI
MMI
MFI
MMF
MIB*
MBB
BBB
MMB*
MFB*
* hint in first bundle
Possible Itanium™ Processor full issue
intel
ITANIUM
XEON
Itanium™ is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.
11


ITAの特長
Dispersal Matrix
MII MLI MMI MFI MMF MIB MBB BBB MBB MFB
MII
MLI
MMI
MFI
MMF
MIB*
MBB
BBB
MMB*
MFB*
* hint in first bundle
Possible McKinley Processor full issue
Possible McKinley and Itanium™ Processor full issue
Architectural Improvements Allow More Issued Instructions/Cycle
Faster Execution
intel
ITANIUM
XEON
Itanium™ is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.
12

ITAの特長

Functional Units

- Integer
  - 6 ALUs
  - 6 Multi-Media
- Memory Ports
  - 2 Load, 4FP Load
  - 2 Store
- Floating Point
  - 2 MACs
- Branch Ports
  - 3 Branches



Itanium™ Processor Binaries will run Faster on McKinley Processors

intel

ITANIUM. XEON.

13

ITAの特長

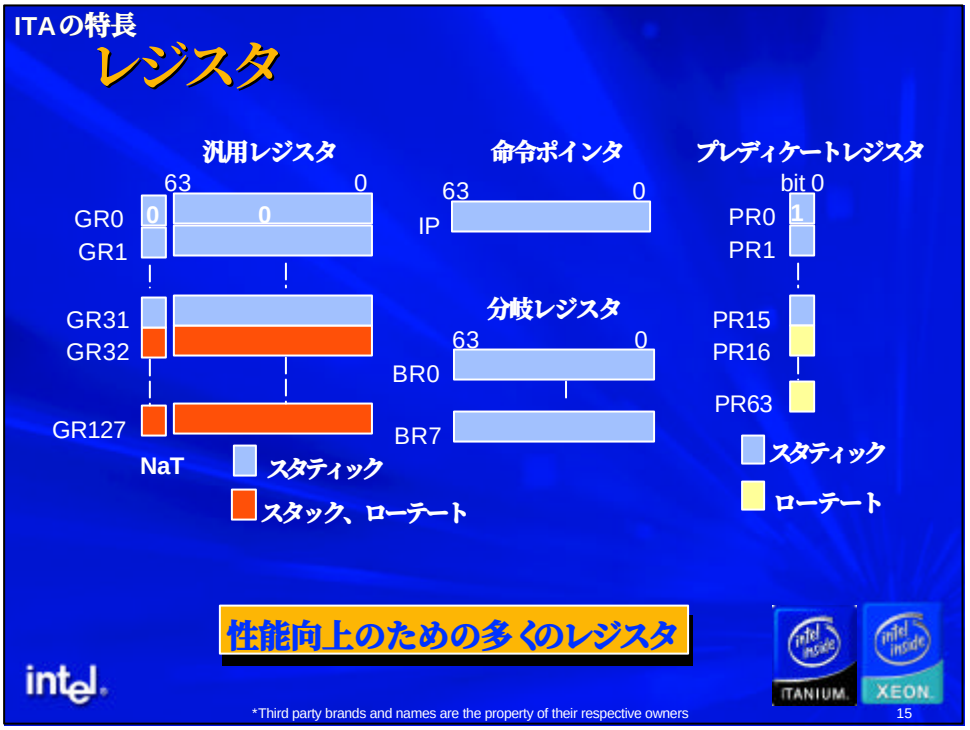
豊富なレジスタ群

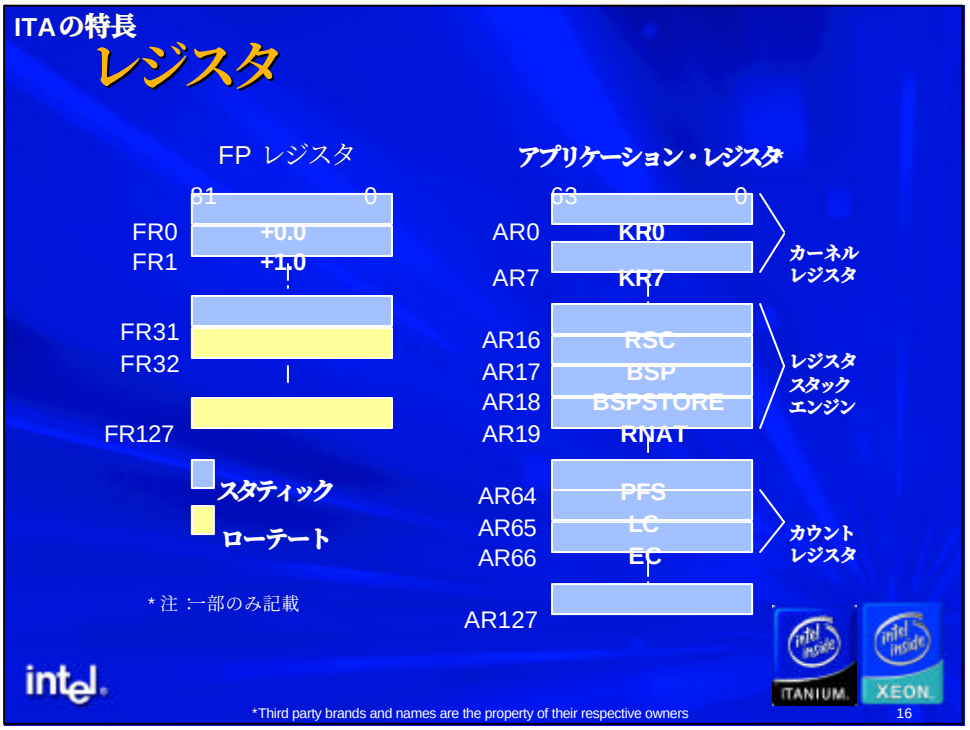
- 128個の汎用レジスタ（64ビット＋1）
- 8つの分岐レジスタ（64ビット）
- 128個の浮動小数点レジスタ（82ビット）
- 64個のプレディケーション・レジスタ（1ビット）
- 命令ポインタ（IP)(64ビット）
- アプリケーション・レジスタ
- 性能モニタ・レジスタ
- プロセッサ識別レジスタ（CPUID)

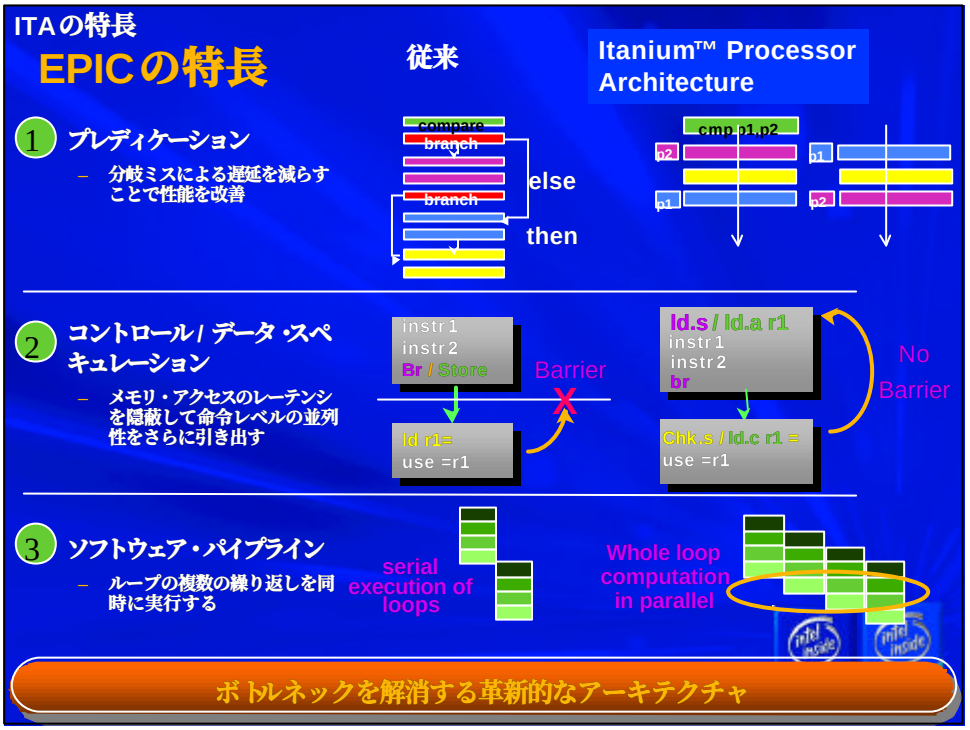
intel

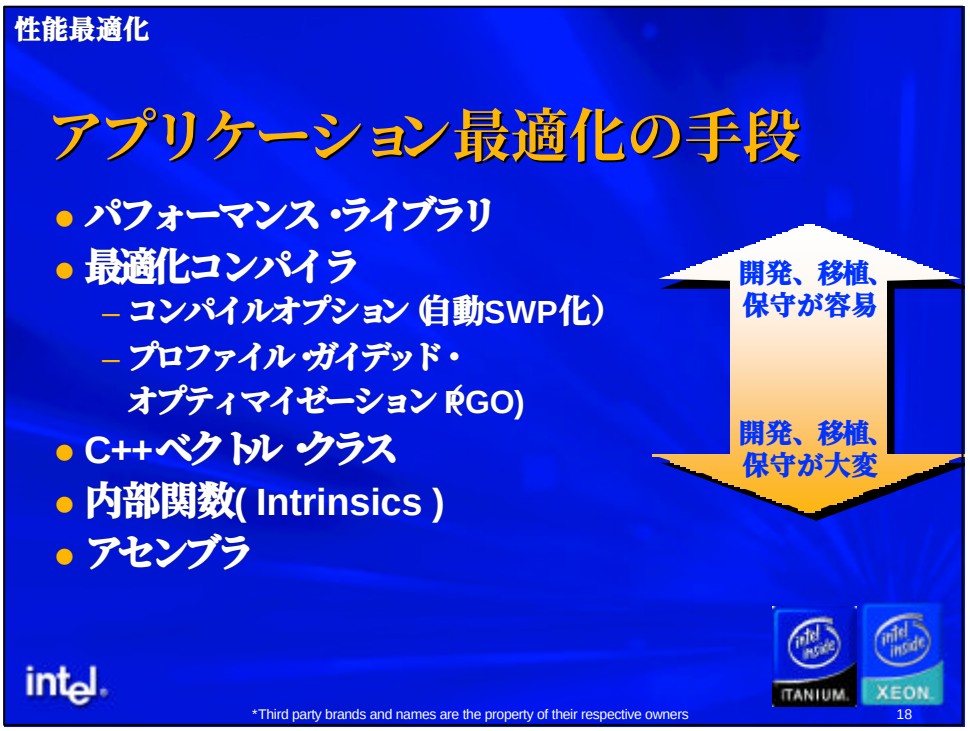
ITANIUM. XEON.

14


ITAの特長
レジスタ
汎用レジスタ
63
0
GR0
GR1
GR31
GR32
GR127
0
0
NaT
スタティック
スタック、ローテート
命令ポインタ
63
0
IP
分岐レジスタ
63
0
BR0
BR7
プレディケートレジスタ
bit 0
PR0
PR1
PR15
PR16
PR63
1
スタティック
ローテート
性能向上のための多くのレジスタ
intel
ITANIUM.
XEON.
*Third party brands and names are the property of their respective owners
15


ITAの特長
レジスタ
FP レジスタ
31
0
FR0
FR1
+0.0
+1.0
FR31
FR32
FR127
スタティック
ローテート
*注 一部のみ記載
アプリケーション・レジスタ
63
0
AR0
AR7
KR0
KR7
カーネル
レジスタ
AR16
AR17
AR18
AR19
RSC
BSP
BSPSTORE
RNAT
レジスタ
スタック
エンジン
AR64
AR65
AR66
PFS
LC
EC
カウント
レジスタ
AR127
intel
ITANIUM.
XEON.
*Third party brands and names are the property of their respective owners
16


ITAの特長
EPICの特長
従来
Itanium™ Processor Architecture
1 プレディケーション
– 分岐ミスによる遅延を減らすことで性能を改善
compare
branch
branch
else
then
cmp.p1,p2
p2
p1
p1
p2
2 コントロール/ データ・スペキュレーション
– メモリ・アクセスのレーテンシを隠蔽して命令レベルの並列性をさらに引き出す
instr 1
instr 2
Br / Store
Barrier
ld r1=
use =r1
ld.s / ld.a r1
instr 1
instr 2
br
No Barrier
Chk.s / ld.c r1 =
use =r1
3 ソフトウェア・パイプライン
– ループの複数の繰り返しを同時に実行する
serial execution of loops
Whole loop computation in parallel
ボトルネックを解消する革新的なアーキテクチャ


性能最適化
アプリケーション最適化の手段
● パフォーマンス・ライブラリ
● 最適化コンパイラ
– コンパイルオプション (自動SWP化)
– プロファイル・ガイデッド・オプティマイゼーション (PGO)
● C++ベクトル・クラス
● 内部関数(Intrinsics)
● アセンブラ
開発、移植、保守が容易
開発、移植、保守が大変
intel
ITANIUM
XEON
*Third party brands and names are the property of their respective owners
18

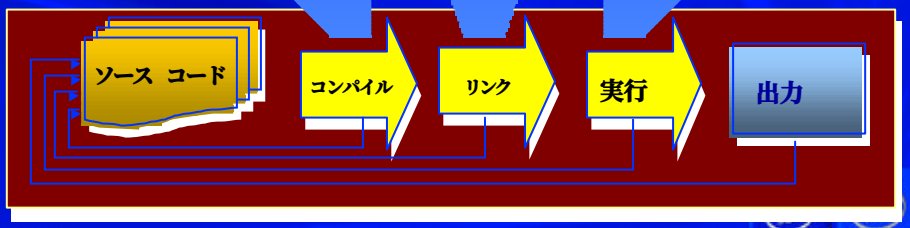
性能最適化

Intel® SW開発ツールでIAの最高性能を

- Intel® C++ コンパイラ
- Intel® Fortran コンパイラ
- KAI* C++ コンパイラ
- KAP/Pro ツールセット

Intel パーフォマンス ライブラリ

- VTune™ 性能アナライザ
- コンサルティング
- 並列アプリケーション・センタ(PAC)のOpenMP™ チューニング



intel

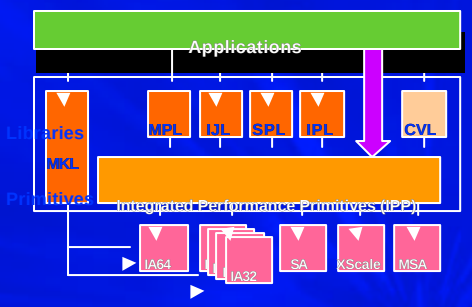
ITANIUM. XEON.

*Third party brands and names are the property of their respective owners

性能最適化

パフォーマンス・ライブラリ

- IAの最新技術のより素早い、より良い採用を促進するために、ソフトウェア開発者にとっての敷居を下げる
  - –ハンド・アセンブラではなくAPI
  - –インテルの最新及び未来のプラット・フォーム上で最高の性能を実現するアプリケーションを開発
  - –新しい機能を用いた製品を早く市場に提供
  - –ワークステーション、デスクトップ、ノート、ハンドヘルドといった異機種上でそれぞれ高性能を実現



developer.intel.com/software/products

intel

ITANIUM. XEON.

*Third party brands and names are the property of their respective owners

性能最適化

ベクトル演算関数例

- 配列の要素に演算を行う
- ippsAdd_[16s|32f|64f]
  - ショート、浮動小数点、倍精度の配列の加算
- ippsMul_16s_Sfs
  - 2つのショートを乗算して結果をスケールする
- ippsConvert_16s32f
  - ショートを浮動小数点数に変換する

intel

IPO

プロシージャ間最適化とは

最適化をファイル間に拡張

IPO無しでは (または -ip)

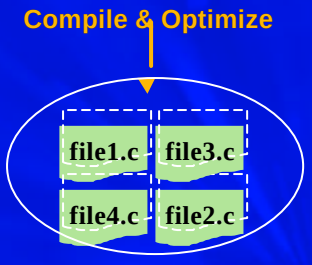
Compile & Optimize → file1.c

Compile & Optimize → file2.c

Compile & Optimize → file3.c

Compile & Optimize → file4.c

IPO有り



intel

IPO

プロシージャ間最適化の効果

- Lynx の成功例
  - インテルのSpice-like 回路シミュレータ
  - アルゴリズム的には高度に最適化済
  - インテル　コンパイラ (icc) で O2 & IPO付
    - 1.2x - 5.2x スピードアップ [gcc -O (2x typical)]
    - 1x - 2.4x スピードアップ [icc -O2 (1.2x typical)]

intel

IPO

利点

- 最適化の可能性を広げる
- 関数順序の最適化
- 関数呼び出しの削減
- コード サイズを大幅に削減
- IA-32 及び Itanium™ アーキテクチャ

intel

IPO

特に効果のあるプログラム

- 多くの小さな汎用関数を持つ
- 頻繁にコンストラクタ／デコンストラクタを利用
- 小さな（1行の）メンバ関数を持つ

IPO

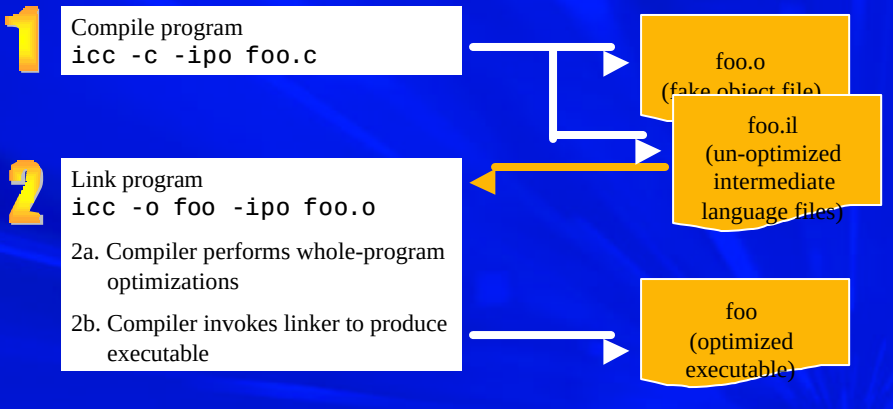
使用方法

1. Compile program
   `icc -c -ipo foo.c`

2. Link program
   `icc -o foo -ipo foo.o`

   2a. Compiler performs whole-program optimizations

   2b. Compiler invokes linker to produce executable



PGO

プロファイル ・ガイデッド最適化とは？

- プログラム実行中に集めたプロファイル情報を基にその後のビルドの最適化を向上させる

intel.

PGO

プロファイル ・ガイデッド最適化の効果

- CSIM 成功例
  - Intel 回路シミュレータ
  - “最適化が困難”
  - インテル　コンパイラ で PGO の結果
    - 1.3x スピードアップ[gcc -O2 or icc -O2]
  - 回路削減により $1.5M の節約

intel.

PGO

利点

- より正確な分岐予測
- より良いレジスタ・アロケーション
- IPOのインラインの改善
- 基本ブロックの移動
  - `status = UtilityFunc (arg1, arg2, arg3)`
    `if (status != 0) // Not expected to fail`
    `HandleErr (status);`
  - 命令キャッシュ使用の改善
- IA-32 及び Itanium™ アーキテクチャ

PGO

特に有効なプログラム

- 常に実行されるパスが決まっている
- If 文や switch 文の多い
- If 文や switch 文の階層が深い

PGO

使用方法

1 Compile+link to add instrumentation
`icc –o foo -prof_gen foo.c`

→ foo (instrumented executable)

2 Execute instrumented program
`./foo`

→ 12345678.dyn (dynamic profile)

3 Compile+link using feedback
`icc –o foo -prof_use foo.c`

→ pgopti.dpi (merged .dyn files)

→ foo (optimized executable)

intel

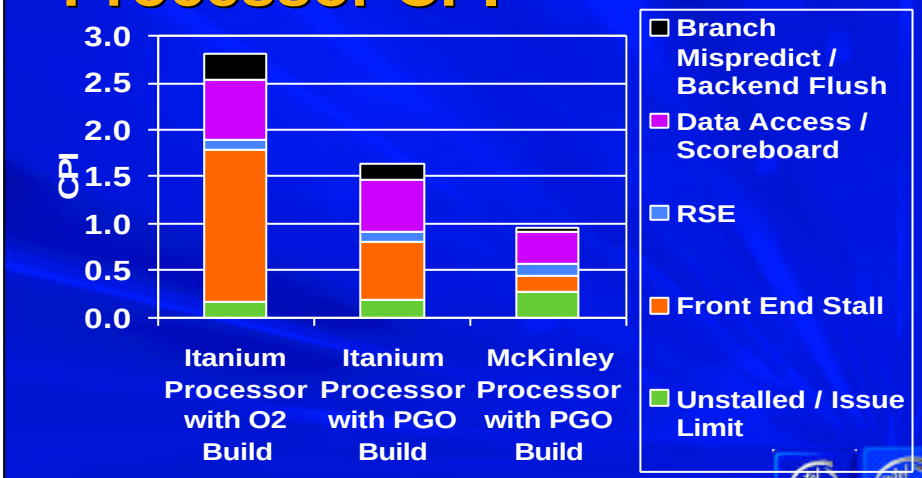
ITANIUM. XEON.

*Third party brands and names are the property of their respective owners

31

PGO

McKinley and Itanium™ Processor CPI

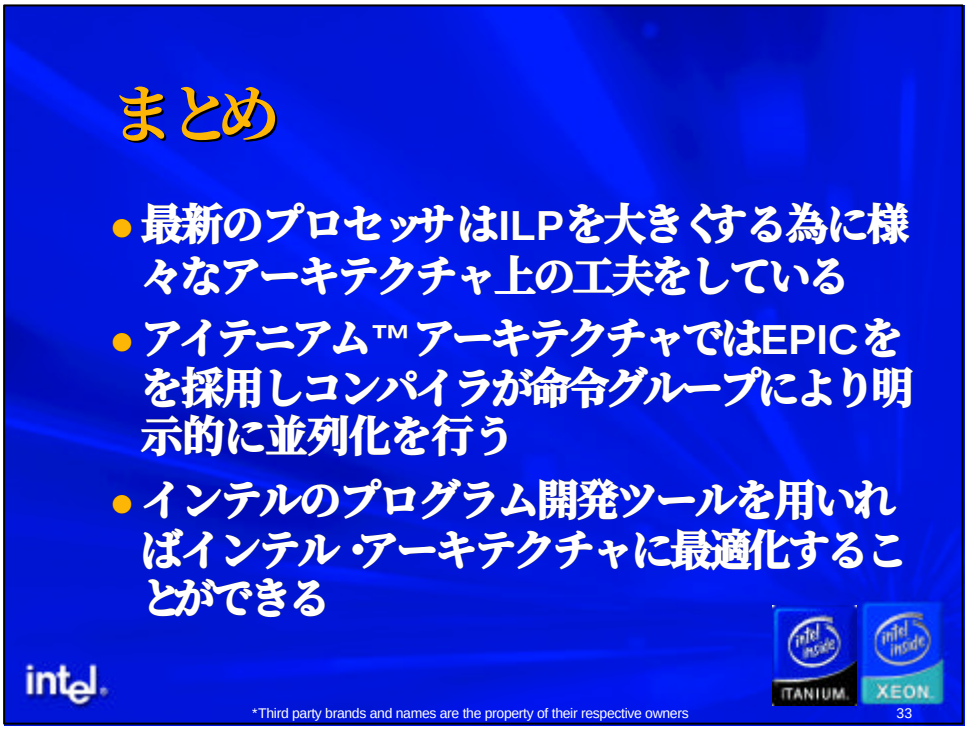


intel

ITANIUM. XEON.

*Third party brands and names are the property of their respective owners

32


まとめ
最新のプロセッサはILPを大きくする為に様々なアーキテクチャ上の工夫をしている
アイテニアム™アーキテクチャではEPICをを採用しコンパイラが命令グループにより明示的に並列化を行う
インテルのプログラム開発ツールを用いればインテル・アーキテクチャに最適化することができる
intel
ITANIUM
XEON
*Third party brands and names are the property of their respective owners
33