次世代インテル® Itanium® アーキテクチャ：インテル® Itanium® 2 プロセッサ

Next Generation Intel® Itanium® architecture : Itanium® 2 processor

池　井　　満

要　約　Itanium® プロセッサは EPIC（Explicitly Parallel Instruction Computing）と呼ばれる，機械語命令を用いて並列性を明示的に記述できるアーキテクチャを採用した．豊富なレジスタと実行ユニットを持ち，その他，プレディケーション，投機実行，ソフトウェア・パイプライニングといった機能を備えており，パイプライン・フラッシュを削減し，メモリロードのレイテンシの影響を最小限にしている．Itanium® 2 プロセッサはさらにメモリ整数演算ユニットを二つ増やしたほか，3 次キャッシュをダイ上に実装しメモリのバンド幅を 3 倍にし，レイテンシを半分にしている．この結果，様々なアプリケーションの実行において，Itanium® プロセッサと比較してさらに 1.5 倍から 2 倍の性能向上を実現している．

Abstract　Itanium® processor introduces a EPIC (Explicitly Parallel Instruction Computing) architecture that allows to specify parallel machine instructions explicitly. It has a variety of registers and execution units, and new features such as predication, speculative execution, software pipelining that contribute to reduced pipeline flush and memory latency.

Itanium® 2 processor has 2 additional integer units and L 3 integrated cache with 3 x increase system bus bandwidth that contribute to reduced memory latency in half. As the result, Itanium® 2 processor achieves 50% thru 100% higher performance than Itanium® processor in various application implementations.

1. はじめに

インテルは 2000 年にインテル最初の本格的な 64 ビットプロセッサとなる Itanium®（アイテニアム®）プロセッサを発売した．このプロセッサは従来の RISC ベースの 64 ビット・アーキテクチャとは全く異なる EPIC アーキテクチャを採用し，このアーキテクチャの実現する高度な並列性は同プロセッサに新しい次元の性能をもたらした．このため，ユニシス社 ES 7000 を始めとする世界中のサーバやワークステーションにこれを採用して頂いている．

2002 年にインテルはこの Itanium® アーキテクチャ・ファミリの第 2 世代となる Itanium® 2 プロセッサの出荷を開始した．この Itanium® 2 プロセッサは，同じバイナリを用いた場合でもそのアプリケーションにより，Itanium® プロセッサ搭載システムの 2 倍の性能を実現できる．これは Itanium® アーキテクチャの優れた拡張性（スケーラビリティ）を示している．

そこで本稿ではこの Itanium® アーキテクチャの性能拡張性について解説する．まず採用した EPIC アーキテクチャについて説明し，これをサポートするために Itanium® アーキテクチャが採用した命令並列化手法について代表的なものを解説する．次に Itanium® 2 プロセッサで行った拡張内容について述べる．最後にインテルがプ

ロトタイプを用いて行った，Itanium® 2 プロセッサの初期評価の内容，結果について述べ，まとめたい．

2. インテル® Itanium® アーキテクチャ

Itanium® アーキテクチャの詳細を述べるのは本稿の目的ではない．ここでは Itanium® アーキテクチャの拡張性を説明のために必要なものに限って代表的なものを選んで解説する．詳細はインテル社マニュアル[1]，または解説書[2] を参照されたい．

2.1 明示的並列命令計算：Explicitly Parallel Instruction Computing（EPIC）

Itanium® アーキテクチャは EPIC と呼ばれる新しい命令実行方式を採用している．図 1 にこの概念図を示す．

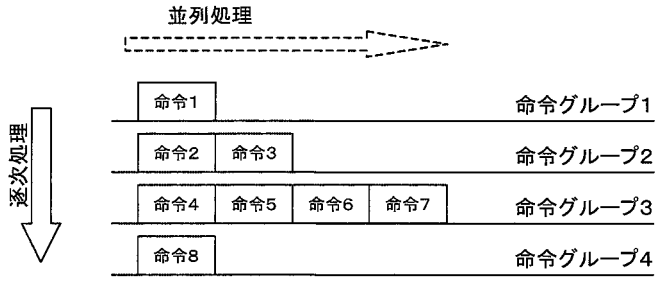

図 1　命令と命令グループ

計算機の機械語命令は一般に 1 次元の命令列として記述され，逐次的に実行されることを前提としている．これに対して EPIC では，図 1 に示すように命令を 2 次元的に表現しており，図 1 で横方向に表記した命令グループと呼ぶ概念を導入している．この命令グループとは 1 個以上の並列に実行可能な命令の集合である．例えば命令グループ 1 は命令 1 が 1 個だけの命令グループである．命令グループ 2 は命令 2 と命令 3 の同時に実行できる 2 個の命令を持つ命令グループであり，また命令グループ 3 は同様に同時に実行できる 4 個の命令（命令 4，命令 5，命令 6，命令 7）を持つ命令グループである．

このように明示的に並列性を記述する命令グループにより，プロセッサは動的な命令のスケジュールから開放される．また一つの命令グループに属する命令数には上限はなく，より高い命令の並列実行を行うことができる．一方，それぞれ異なる命令グループに属する（図 1 の逐次処理方向にずれた）任意の二つの命令の実行時刻は 1 クロック以上ずれることが保証される．今ここで，理想的な EPIC を搭載したプロセッサを考えると，これは毎クロック命令グループ単位で命令を処理できるはずである．そういう意味で命令グループとは Itanium® アーキテクチャの論理的な命令実行処理単位といえる．

2.2 バンドルとテンプレート

Itanium® アーキテクチャ命令列内の命令並列実行単位は命令グループであるが，これは可変長であり物理的な命令実行単位としては好ましくない．Itanium® アーキテクチャではこの命令グループの並列性を生かす物理的命令実行単位として，3 命令

をパック化したバンドルと呼ぶ入れ物を用いる．図 2 にこの構造を示す．

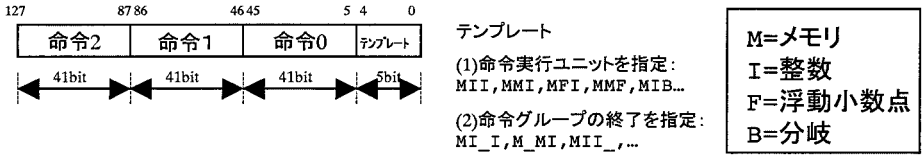

図 2　バンドルとテンプレート

一つのバンドルは 128 ビットで，41 ビットの命令 3 個と 5 ビットのテンプレートから構成されている．Itanium® アーキテクチャ命令は一般的な RISC プロセッサと同様に，ロード・ストア命令以外ではメモリ参照を行わない．基本的にはオペランドにレジスタを三つ用いた op r 0＝r 1，r 2 の命令体系を採用しており，メモリ（M），整数（I），浮動小数点（F）及び分岐（B）の四つの実行ユニットのいずれかで実行される．

テンプレートには次に示す二つの重要な働きがある．

1）　命令実行ユニットの指定：テンプレートの 4 ビットを用いて 12 種類の命令実行ユニットの組み合わせを定義している．例えば MFI のテンプレートは命令 0 がメモリユニット，命令 1 が浮動小数点ユニット，命令 2 が整数ユニットに送られる．

2）　命令グループ境界の指定：12 種類の組み合わせのうち 2 種類は“ __ ”を含んでおり，これはバンドル内での命令グループの終了を示す．またテンプレートの残りの 1 ビットはバンドル境界で命令グループが終了するかどうかを示している．

Itanium® アーキテクチャのプロセッサでは，1）と 2）により，このテンプレートを調べるだけで直ぐに複数の命令を複数の実行ユニットに配布することができる．この際に命令のデコードや依存性を調べる必要はない．このため，非常に高速に多数の命令を実行ユニットへ配布することが期待できる．このバンドルは Itanium® アーキテクチャ命令の物理的な命令実行最小単位である．Itanium® プロセッサでは二つのバンドルを同時に処理し，六つの命令を実行ユニットに配布することでクロック当たり最大 6 命令を同時に処理することができる．

2.3 レ ジ ス タ

多くの命令を並列に処理するためには，その処理結果を保持するレジスタも十分準備する必要がある．このために Itanium® アーキテクチャではレジスタの数を増やすだけではなく，様々な種類のレジスタを設けてプログラムの並列実行を支援している．図 3 に代表的な 3 種類のレジスタ，汎用レジスタ，浮動小数点レジスタ，プレディケートレジスタを示す．

Itanium® アーキテクチャは 128 個の 64＋1 ビット汎用レジスタを持っている．このレジスタはデータ格納用としては 64 ビットであるが，後述する投機実行の際に各レジスタの無効性を示す NaT（Not a Thing）ビットをそれぞれ別に 1 ビットずつ持っている．128 個のレジスタのうち，gr 0 から gr 31 の 32 個のレジスタはスタティ

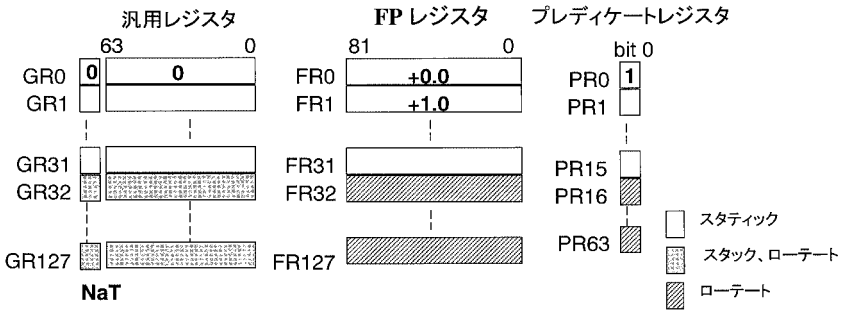

図 3　Itanium® アーキテクチャの代表的レジスタ

ックと呼ばれており一般的なプロセッサのレジスタと同様に一つのレジスタ名（例えば gr 0 は r 0）のみで参照される．残りの gr 32 より上位は次の二つの方法でリネームすることができる．

1）　スタック：レジスタを alloc 命令により最大 96 まで可変長の高速スタックとして使用できる．関数呼び出し等では引数としても利用できる．

2）　ローテート：一部の分岐命令でレジスタの参照番号をずらすことができる．これは後述するソフトウェア・パイプライニングで使用する．

スタックによるレジスタ・リネームを利用できるのは汎用レジスタのみである．

浮動小数点レジスタは 82 ビットで，Itanium® アーキテクチャはこれも 128 個持っている．浮動小数点データのコード化の方法は IEEE 拡張倍精度準拠の 80 ビットコードの指数部分に 2 ビットを追加したもので，これにより，収束演算の収束速度を改善している．また浮動小数点コードに NatVal（Not a Thing Value）と呼ぶコードを追加して，浮動小数点レジスタの投機実行処理時には汎用レジスタの NaT と同様な処理を行う．浮動小数点レジスタは fr 0 から fr 31 までがスタティックで，残りはローテートによるリネームが可能である．

プレディケート・レジスタは比較演算結果を格納するための 1 ビットのレジスタで，全部で 64 個存在する．Itanium® アーキテクチャではこのレジスタを用いて並列に多数の比較演算等を行い，その結果を同時に格納しておくことができる．このレジスタは pr 0 から pr 15 の 16 個はスタティックで，残りがローテートによるリネーミング可能である．このレジスタは次に述べるプレディケーションで利用される．Itanium® アーキテクチャではこの他にも分岐，性能，システム関連でたくさんのレジスタを持っている．

3.　命令の並列化

Itanium® アーキテクチャではコンパイラが並列性の高い EPIC コードを生成できるように様々なアーキテクチャ上の拡張を行っている．ここではこれらのうち代表的な三つについて説明する．

3.1　プレディケーション

プログラム中から並列性を見出す上での障害の一つは分岐である．分岐の多いプログラムでは基本ブロック[*1]が小さくなり，この結果高い並列性を得にくくなる．Ita-

nium® アーキテクチャではプレディケーションと呼ばれる新しい機能により分岐を減らしている．　図4にプレディケーションの例を示す．

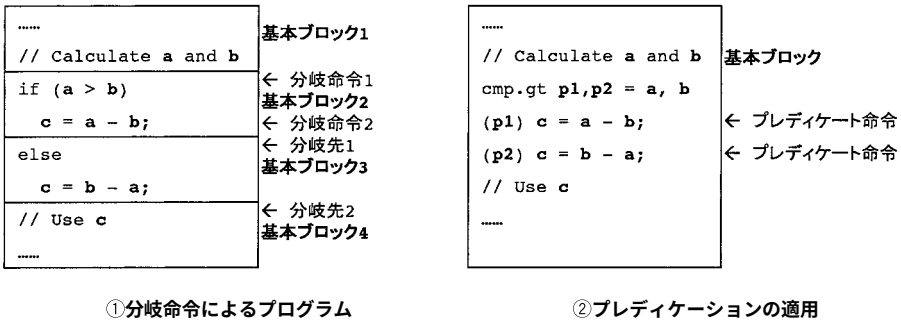

図 4　プレディケーション例

図4の①に変数 a と b の計算後その差 c を求めるプログラムの擬似 C コードを示す．このプログラムでは a と b を比較する if 文の直後に else ブロックへの分岐命令1があり，また c の演算の直後に else ブロックをバイパスする分岐命令2がある．分岐命令の分岐先はそれぞれ else ブロックの先頭と直後にある．この結果プログラムは図4に示したように四つの基本ブロックに分割される．一方②プレディケーションを用いた方法では基本ブロックは分割されることはない．比較命令は比較（a>b）した結果により，プレディケートレジスタ p1，p2 のペアを真（1，0）か偽（0，1）にセットする．プレディケート命令では，実行命令の前に（p1）のように，プレディケートレジスタを指定する．このようにプレディケートレジスタで修飾された命令は該当するプレディケートが真の場合にしか実行されない．したがって図4では二つのプレディケート命令は両方ともフェッチされて実行ユニットに送られるが，実際にはどちらか一つだけが実行されることになる．

このように，プレディケーションにより分岐命令を減らすことができ，基本ブロックを大きく保つことができる．分岐命令を減らすことにより分岐予測ミスに伴うパイプラインフラッシュのペナルティを減らすことが期待できるほか，より大きな基本ブロックからは，より高い並列性の抽出を期待することができる．

3.2　投機的実行命令

Itanium® アーキテクチャはデータをメモリからロードする命令に，アドバンスドロードとスペキュレーティブロードの2種類の投機的に実行できる命令を持っている．ここではこれらのうち，スペキュレーティブロードの使用例を図5に示す．

図5の（1）に示すようにロード命令 ld でデータをメモリ（r12でアドレスを間接指定）から r3 にロードしてすぐ使用する場合，ロード命令から使用命令までの間，メモリの値をロードするための待ちが発生する．このような場合は待ち時間に他の命令を実行できるように，ロード命令を予め早めに実行したい．ところが図5の例のようにロード命令の前に分岐命令がある場合，この命令がバリアとなって，ロード命令をバリアより前に移動することができない．分岐した場合には，このロード命令は実行されないはずだが，もしも移動して予め実行してそのロード命令で例外が発生する

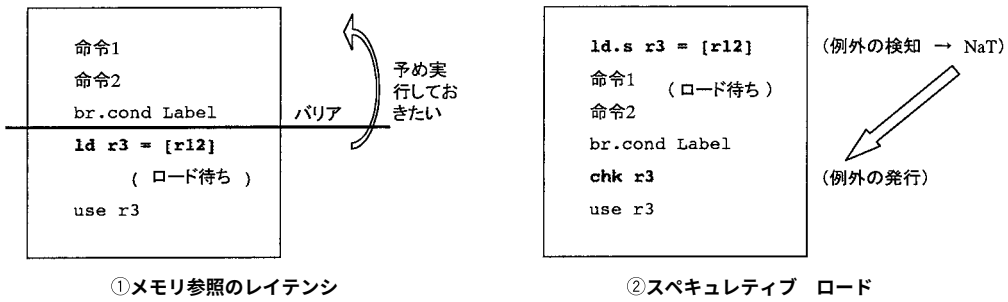

図 5 スペキュレーティブロードの例

と，実行しないはずの命令で例外を発生してしまうことになるからである．

そこで Itanium® アーキテクチャでは，このバリアを超えて移動できるスペキュレーティブ・ロード ld.s を準備している．図 5 の（2）に示すように，この ld.s は例外を検知した場合は例外を発生せずに NaT ビットに例外を記録する．本来のロード命令の位置に置いた chk.s 命令が実行され，NaT ビットを検知したときのみ例外を発生させる．したがって，分岐する場合には仮に NaT ビットがセットされていても例外発生はなく，また分岐せずに有効なロード命令が実行される場合は予めロード命令を実行しているために，ロード待ち時間中に他の命令を実行することができる．このように命令のスケジューリングの自由度を上げる投機的実行命令を用いることにより，命令のレイテンシを隠蔽したより並列性の高いプログラムを構成することができる．

3.3 ソフトウェア・パイプライニング

特に科学技術計算等に多い，繰り返し回数の大きな演算ループプログラムを並列化する方法としてはソフトウェア・パイプライニング（SWP）がよく知られている．図 6 にこの例を示す．

```
C source:
for (i=0; i<max;i++)
  a[i]=b[i]+x*c[i];

Assembler:
loop:
{.mmi
(p16) ldfd f32=[r34],8
(p16) ldfd f37=[r33],8
      nop.I 0
}
{ .mfb
(p20) fma.d f42=f8,f36,f41
(p24) stfd [r32]=f46,8
 br.ctop.sptk loop ;;
}
```

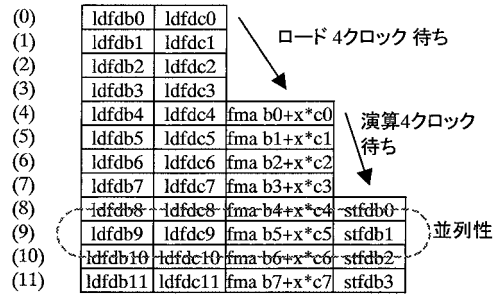

図 6 ソフトウェア・パイプライニング

図 6 の左上 C のプログラムに示すように配列 b[] の要素と配列 c[] の要素を x 倍したものを加算して配列 a[] に格納する場合を考える．この場合のアセンブラ出力は .mmi と .mfb で始まる二つのバンドルとなる．まずは，各命令先頭のプレディケ

ート指定と引数の浮動小数点レジスタ名を無視する．最初のバンドルにはb[i]（汎用レジスタr33でアドレス指定後8バイトポストインクリメント）とc[i]（汎用レジスタr34でアドレス指定後8バイトポストインクリメント）を読み出す二つの倍精度ロード命令ldfdと空きをうめるnop.i命令で構成される．残りのバンドルはb[i]＋x＊c[i]を演算する倍精度乗加算命令fma.dと結果をa[i]に格納するstfd命令と，繰り返しを行うbr.ctop命令から構成されている．

今，仮に上述のロードと演算にそれぞれプロセッサの4クロックを要すると仮定する．すると，1回のループを実行するのに，8クロックを要し，単純にmax個の全要素を処理するには8×maxのクロックを要することになる．ところが，このような場合は図6の右側に示したように，ループの実行順序構成し直して，命令をずらしながら実行することが可能である．つまり，最初のクロック（0）サイクルから（3）サイクルまではロード命令だけを実行する．そして，（4）サイクルからは前にロードした結果がレジスタに格納されるので，乗加算命令も実行できるようになる．そして，その4クロック後最初の乗加算命令の結果が得られる（8）サイクルからは毎サイクル結果を得ることができる．この場合の必要実行クロック数は，8＋maxとなる．maxが十分大きいと仮定すると後者の方が8分の1の時間で実行できることになる．このような手法をSWPと呼んでいる．

SWPの問題点は，ループ内の処理をずらして実行するためにプログラムが複雑になり余分な処理が必要になることにあった．しかしItanium® アーキテクチャではこの処理を専用のループ命令でそのまま行うことができる．実はプレディケート指定やレジスタのローテーションにより，図6で示したアセンブラのプログラムはSWP化されてその右のように実行される．詳細は文献[2]を参照して頂きたいが，プログラム中のbr.ctop命令がこのための分岐命令である．プログラムの同じサイクルで実行される命令が同じプレディケートレジスタp16，p20とp24で修飾されていることに注意されたい．このプログラムのSWP化されての動作は直感的には理解しにくいが，実際にはコンパイラがコードを生成するので，一般ユーザが心配する必要はない．このように，Itanium® アーキテクチャでは科学技術計算のループを自動的に並列化して実行することができる．

4. インテル® Itanium® 2プロセッサの拡張

従来のRISCアーキテクチャの制限を超え，大きな並列性を実現できるItanium® アーキテクチャを実現した最初の製品がItanium® プロセッサであるが，これをさらに拡張して性能の向上を実現したのがItanium® 2プロセッサである．

4.1 Itanium® 2プロセッサのアーキテクチャ

Itanium® 2プロセッサの特長を図7に示す．Itanium® 2プロセッサはItanium® プロセッサと同一の命令セットを持ち，その性能を向上させたものである．まず，システムバスのデータ幅を128ビットに広げ，周波数を400 MHzにすることにより，データ転送レートを6.4 GB/Sに拡張した．また3 MBの3次キャッシュをプロセッサと同じダイ上に持つことにより，キャッシュ性能を向上している．パイプラインの段数は8ステージで，実行ポートはItanium® プロセッサの9個から11個に増やしてい

る．アプリケーション・レジスタとプレディケーション・レジスタの数にはそれぞれ変更は無い．実行ユニットでは整数命令を実行できる整数ユニット数を 4 個から 6 個に強化している．また 1 クロックあたり 2 個のロードと 2 個のストアを同時に実行できる．そして 1 クロックあたり 2 バンドル 6 命令とサイクル当りの実行可能命令数はそのままで，実行周波数を 1 GHz に向上した．

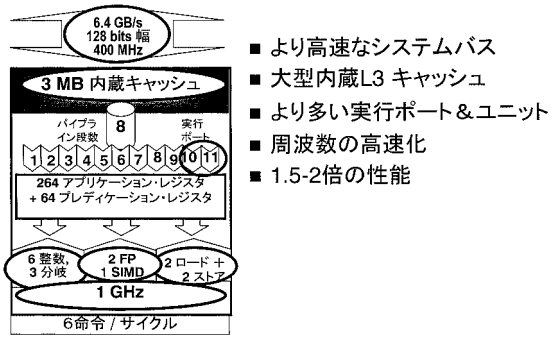

図 7　Itanium® 2 プロセッサの特長

4.2　キャッシュ性能の改良

図 8 に Itanium® 2 プロセッサのキャッシュ構成を示す．図 8 の破線内はプロセッサと同じダイ上に実装されている．破線内の矩形はそれぞれ左から 3 次（L 3），2 次（L 2）と 1 次（L 1）のキャッシュを表している．1 次キャッシュは上の命令キャッシュ（L 1 I）と下のデータ・キャッシュ（L 1 D）に分かれているが，他は命令・データ共用のユニファイド・キャッシュとなっている．L 3 と L 2 の矩形内の値は，それぞれ上から容量，ウェイ数，ライン長とレイテンシを示している．3 次キャッシュの容量は 3 MB で，12 ウェイ，ライン長は 128 バイトで，レイテンシが最低 12 から 15 クロックである．2 次キャッシュは容量 256 KB，8 ウェイ，ライン長は 128 バイトで，レイテンシが最低 5 から 7 クロックである．1 次キャッシュは命令データ共に 16 KB の 4 ウェイで，ライン長は 64 バイト，レイテンシは 1 クロックと非常に高速である．

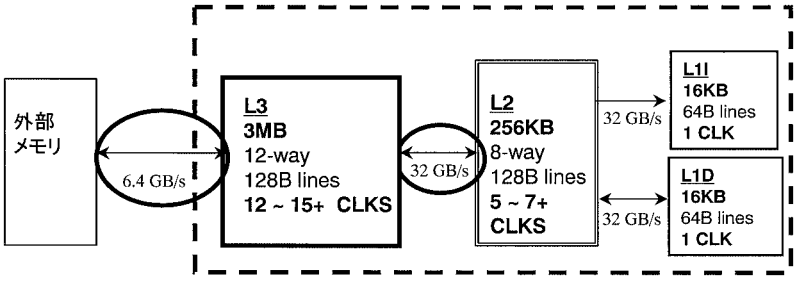

図 8　Itanium® 2 プロセッサのキャッシュ構成

Itanium® 2 プロセッサでは，全てのキャッシュレベルでレイテンシを Itanium® プロセッサの半分程度に抑えている．またキャッシュ間のデータ転送速度は図 8 の各キャッシュ間を結ぶ矢印の下に記載したように，3 次キャッシュ以上の全てのレベルに

おいて 32 GB/S を実現している．即ち，1 クロックに二つの 64 ビットデータをストアして，かつ二つの 64 ビットデータをロードするメモリ・アクセスの最高速実行を可能にしている．

4.3 実行ユニットの拡張

図 9 に Itanium® 2 プロセッサの実行ユニットの一覧を示す．図 9 の右側にはユニットの数を Itanium® プロセッサと比較して示した．整数の演算やマルチメディア命令を実行するユニットは 4 個から 6 個に増えた．このためこれらの整数命令をクロック当り 6 個まで実行することが可能となった．これら 6 個のユニットのうち，4 個はメモリ処理も実行することができる．このため Itanium® 2 プロセッサでは 2 個のロードと 2 個のストアを 1 クロックで処理することができる．Itanium® プロセッサは基本的に 2 個の倍精度（または拡張倍精度）の乗加算ユニットを持っており，2 個の乗加算を 1 クロックごとに実行することができる．ただし，これらの 2 個のユニットのうち，32 ビットの単精度浮動小数点演算を 2 個同時に乗加算する SIMD 演算を行えるユニットが，Itanium® 2 プロセッサでは 1 個に減っている．分岐処理ユニットの数は三つで増減はない．

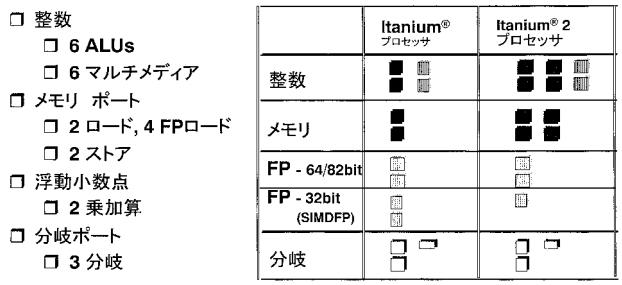

図 9　Itanium® 2 プロセッサの実行ポート

5. 性 能 評 価

Itanium® プロセッサにポーティングされたアプリケーションは今後の Itanium® アーキテクチャを採用したプロセッサ上での優れた性能向上＝スケーラビリティを実現すると予想されている．ここではバイナリを Itanium® 2 上で実行するだけで得られる 1.5 倍から 2 倍の性能向上について述べる．

5.1 対象アプリケーション

Itanium® 2 プロセッサを搭載したシステムは，エンタープライズ・レベルのアプリケーションからスーパコンピュータで行うような複雑な科学技術計算に至るまで広い領域の様々なアプリケーションで優れた性能を発揮することが期待される．表 1 にこれらのうち代表的な 6 分野と各々の分野での性能指標として評価したベンチマークプログラムを示す．

1) 大型データベース：大きなアドレス空間と拡張された並列性を用いて，より大きなデータに，より多くの同時アクセスを実現できる．最新のインメモリ・データベース（IMDB）では特に効果が大きい．

2) ビジネス：大型高速なオンダイ・キャッシュや高速入出力回路等システム全体

表 1　ベンチマークプログラム

アプリケーション分野	ベンチマークワークロード	性能指標
大型データ・ベース	IMDB を用いたトランザクション処理	オンライン・トランザクション処理
ビジネス	サプライ・チェーン	一般的な ERP 処理
コンピュータ・エイデッド・エンジニアリング（CAE）	CAE	WS 用のアプリケーション
ハイパーフォーマンス・コンピューティング（HPC）	SPECint/SPECfp2000	CPU 性能
	Linpack 10K	浮動小数点性能
	GAMESS	量子化学や科学技術計算
アプリケーション・サーバ	SPECJ2000	Java のビジネス・ベンチマーク
セキュリティ	Decypts/Sec	暗号化やセキュア・トランザクション処理

として得られる潤沢な処理性能により，拡大し続けるエンタープライズ市場に対応できる．ここでは一般的な ERP アプリケーションとしてサプライ・チェーンを取り上げる．

3）　コンピュータ・エイデッド・エンジニアリング（CAE）：機械や電子回路のデザインにおいて，より詳細なデータを広いアドレス空間に展開し，高性能な浮動小数点演算器により高速処理を行うことが可能である．

4）　ハイパーフォーマンス・コンピューティング（HPC）：64 ビットの仮想アドレス空間はクラスタ等のスケールアップにも利用可能であり，インフィニバンド等の高性能インターフェースを使用して数千プロセッサを接続できる．ここでは代表的なベンチマークを用いる．

5）　アプリケーション・サーバ：高性能な浮動小数点演算器と多くのレジスタにより，増えつづけるデータを高速に処理，解析し，図表化することができる．ここでは Java のベンチマークを用いる．

6）　セキュリティ：エンタープライズ・アプリケーションではセキュアなトランザクション処理がますます重要になりつつある．ここではセキュア・ソケット・レイア（SSL）の暗号化処理速度を評価する．

5.2　初期評価結果

六つの分野における性能の初期評価を Itanium® システムと Itanium® 2 ベースのプロトタイプ・システムを用いて行った．比較の対象となった Itanium® システムは 800 MHz の 4-ウェイのシステムで，2 GB のメモリを搭載している．また Itanium® 2 ベースのプロトタイプ・システムは 1 GHz の 4-ウェイシステムで同じく 2 GB のメモリを搭載して評価を行った．この性能評価の条件については別途文献[3] に述べられている．詳細はそちらを参照していただきたい．

図 10 にそれぞれのベンチマーク負荷を実行したときの Itanium® 2 ベースのシステムで得られる性能向上倍数（スピード・アップ）値を示す．最も性能向上の少なかったセキュリティの SSL デコード・エンコードの場合で 1.5 倍の性能向上が得られた．またデータベース（IMDB）やアプリケーション・サーバ（SPECJ 2000）では 1.9 倍と 2 倍近い性能向上が得られている．このように，Itanium® 2 プロセッサに特に最適

化することなく，ただ同じバイナリを実行するだけで，ここに示したすべてのアプリケーションで 1.5 から 2 倍近い性能向上が得られることがわかった．

このように，Itanium® 2 プロセッサにおいては，他の既存のアーキテクチャに比べて絶対値の高い性能が得られるだけに留まらない．インテルの半導体技術で得られた動作周波数の上昇分に加えて，さらに大きな性能向上を得ることができる．これは，高度な並列処理を行うことが可能な優れたアーキテクチャにより，世代間での大きな性能向上が実現できるからである．

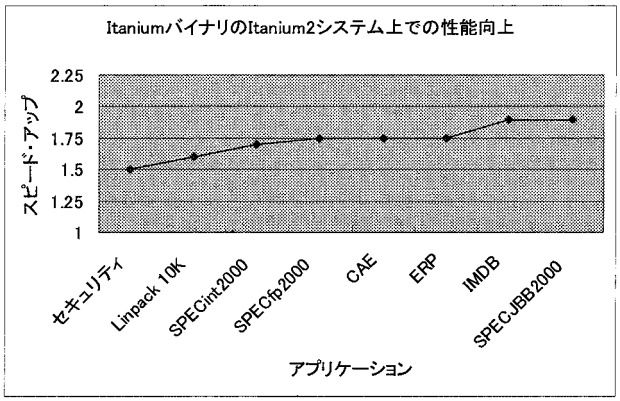

図 10　Itanium® 2 システムでの性能向上

6. おわりに

インテルの 64 ビットプロセッサ Itanium® 2 プロセッサは Itanium® プロセッサ・ファミリの第 2 世代として登場し，そのプロセッサ性能絶対値の高さだけではなく，Itanium® アーキテクチャ性能のスケーラビリティの高さを示している．本稿では，これを実現している Itanium® アーキテクチャの特徴とその Itanium® 2 プロセッサでの拡張内容について解説をした．また最後に Itanium® 2 プロセッサの性能向上が得られているアプリケーション分野とその程度についてプロトタイプを用いた評価結果を示した．

ここで述べた 1.5 倍から 2 倍の性能向上はプロトタイプ上で手持ちのバイナリをそのまま実行しただけの結果であって，十分な性能評価とはいえない．真の性能向上値を得るには，さらに様々なプラットフォームでのアプリケーションの再コンパイルも含めた最適化を行う必要がある．

この Itanium® アーキテクチャの各プロセッサ世代の性能のスケーラビリティの大きさはアーキテクチャの将来性を期待させる．インテルは Itanium® プロセッサ・ファイミリの今後について，Itanium® 2 プロセッサとその 2 世代先のプロセッサ（コード名：Madison，Montecito）まで共通プラットフォームでサポートすることを表明している．もしもプロセッサの差し替えだけで次世代のプロセッサでも Itanium® 2 プロセッサと同様の性能のスケールアップが望めるとすると，そのようなプラットフォームを採用したサーバは非常に魅力的システムといえる．

＊1　先頭が実行されれば分岐せずに必ず実行される命令列のブロック

参考文献　［1］　Intel® IA-64 アーキテクチャ・ソフトウェア・デベロッパーズ・マニュアル第 1-4 巻, 2000 年 7 月
［2］　池井満, IA-64 プロセッサ基本講座, オーム社, 2000 年 8 月
［3］　Intel®, The next generation Intel® Itanium Processor, June 2002

執筆者紹介　池　井　　満（Mitsuru Ikei）
インテル株式会社プラットフォーム＆ソリューションズマーケッティング本部勤務 シニアエンジニア，IEEE, ACM および情報処理学会会員
1993 年インテル株式会社に入社．分散メモリ並列計算機 PARAGON のプログラム最適化を行って以来，様々な IA システムへのプログラム移植，最適化のサポートを行っている．