

インターネット・ストリーミング SIMD 拡張命令

Shreekant (Ticky) Thakkar、インテル・コーポレーション、マイクロプロセッサ・プロダクト・グループ
Tom Huff、インテル・コーポレーション、マイクロプロセッサ・プロダクト・グループ

摘要

この記事では、インテル・Pentium® III プロセッサで新たに導入されたインターネット・ストリーミング SIMD 拡張命令の開発経緯と新規に追加された命令セットの概要について説明します。新しい拡張命令は SIMD-FP 命令、ニュー・メディア命令、およびストリーミング・メモリ命令の 3 つのカテゴリに分類されます。新しい拡張命令は、3D ジオメトリ・パイプラインの処理速度を前世代のプロセッサの約 2 倍に高速化し、リアルタイム MPEG-2 エンコードなど新しいアプリケーションの実行を可能にします。Pentium III プロセッサでは、この処理能力の向上が、従来のプロセッサに比べて約 10% 増のコンパクトなダイ・サイズにより実現されました。また、新たに定義された拡張命令は、短期的な性能向上の目標を達成しただけでなく、将来の実装のために必要なパフォーマンス・スケーラビリティも備えています。

はじめに

1995 年後半の時点で、量産型 PC の市場分野におけるビジュアル・コンピューティングの重要性が急速に高まりつつあることはすでに明らかでした。この市場の要求に応えるために、インテルは 1999 年の量産型 PC 市場での実現を目指して、新しいビジュアル・コンピューティングのための構想を立案しました。この目標を達成するには、アーケード・ゲーム機からワークステーションのアプリケーションまで、さまざまな用途に応じてスケラブルにパフォーマンスを発揮できるバランスの取れた 3D グラフィックス・プラットフォームを開発する必要がありました。3D ジオメトリの主要な処理は浮動小数点(FP)演算によって行われるため、3D の総合的なパフォーマンスを高めるには FP 演算の高速化が不可欠となります。

ユーザが目で見えて実感できるだけのパフォーマンスの向上を実現するには、IA-32 プロセッサのネイティブな FP パフォーマンスを従来の 1.5~2 倍に引き上げる必要がありました。3D グラフィックス・アプリケーションは、1

つ 1 つの 3D データ・タイプ(頂点)ごとに同じ演算処理を繰り返し実行する必要があるため、SIMD (Single Instruction Multiple Data) 並行処理モデルの適用によって大幅な高速化が期待できます。SIMD による並行処理は、汎用プロセッサにおいて 3D アプリケーションの FP 演算を高速化する最もコスト効率の良い方法であり、その高速化の手法は Intel® MMX® テクノロジー拡張命令による整数演算アプリケーション高速化の方法に類似しています(参考資料[1])。このような理由により、IA-32 を拡張する手段として SIMD-FP モデルが選ばれました。

命令セット・アーキテクチャ(ISA)の対象領域も拡大され、3D ジオメトリだけにとどまらず、MMX テクノロジーの使用に関する各ソフトウェア・メカからのフィードバックや、3D ソフトウェア・レンダリング、ビデオのエンコードとデコード、および音声認識のサポートも ISA に含まれるようになりました。また、演算処理とメモリ・アクセスの並行実行性を高めるために、キャッシュの使用を制御する命令も追加されました。IA-32 アーキテクチャに追加された新しい命令は総計で 70 に上り、しかも、それらの命令に対応する新たなステートも追加されました。新しいステートが追加されたのは、Intel® i386™ プロセッサで x87-FP が導入されて以来初めてのことです。この記事では、IA-32 アーキテクチャの定義過程の中で新たに導入された機能の特徴と各種の新しい命令について解説していきます。

アーキテクチャの定義

2 ワイド SIMD-FP と 4 ワイド SIMD-FP の選択

今回のアーキテクチャ拡張における最も重要な要素は、単精度浮動小数点演算の高速化です。この高速化を実現するにあたっては、32 ビット浮動小数点データの 2 ワイド並行処理と 4 ワイド並行処理のどちらを採用するかという選択が問題となりました。新しいアーキテクチャの主要な枠組みを決定づけることになるこの極めて重要な選択については、この後で詳しく説明します。

IA-32 での SIMD-FP の実現可能性について開発チームが最初に検討を行ったところ、新しいマイクロアーキテクチャでは、極端な複雑化やダイ・サイズに関わる大きなコスト増を伴うことなく、1クロック間の4ワイド単精度浮動小数点演算の実行が実現できるという結果が示されました。並行処理を実現する基本方針としては、既存の64ビット・ハードウェアの2重サイクル化が想定されました。この方式のパフォーマンス上の利点は、ジオメトリ・パイプラインの1.5~2倍(場合によってはそれ以上)の高速化が実現できることです。この高速化によって、従来よりも格段にリアルなビジュアル表現ができるようになります。

同様のパフォーマンス向上を実現する方法としては、スーパースケラ設計によって実行ユニットを追加することも考えられます。この方法にはプログラマにとって簡明で分かりやすいという利点がありますが、バスやレジスタ・ファイル・ポート、実行ユニットの数が多くなり、スケジューリングも複雑化するため、サイズやタイミングの面で大幅なコスト上昇を招きます。

また、128ビットを超えるデータバスの採用も、コストとパフォーマンスのバランスを考慮した結果、やはり合理的な選択肢とはみなされませんでした。バスやレジスタはx87-FPでもすでに80ビットの幅を持っているため、128ビットのデータバスでは漸増的な拡張にしかありませんが、256ビットまで拡張すれば顕著なパフォーマンスの向上が期待できます。既述のように、128ビットの処理は実際には64ビットごとに分けて実行されますが、現実の処理の過程でしばしば出現するような、(例えば、加算-乗算-加算-乗算といったパターンで)異なる実行ユニットを使用する命令が交互に繰り返される場合には、単一の128ビット操作によるピーク速度に相当するスピードが維持できます。256ビット幅のSIMDユニットを実装する場合でも、同様の手法でスループットのピーク値を維持するためには、やはり実行ユニットのデータ幅を2倍に拡張する必要があります。また、SIMDのデータ幅を128ビット超に拡大すると、その幅の広い実行ユニットにデータを遅延なく転送するために、メモリの帯域幅を拡張することも必要になります。メモリの帯域幅は、必ずしもその他のアプリケーション領域の要求に応じてムーアの法則のペースで進歩し続けているとは言えず、その帯域幅を増やすにはコストがかかります。さらに、このアーキテクチャ拡張は3Dジオメトリの処理効率の向上を主目的としているため、現在のデスクトップ3Dアプリケーションでは(1ストリップあたり20頂点程度の)比

較的小型の三角形ストリップが使用される傾向が強いことを考えれば、4ワイドを超える並行処理機能を実装しても、それに見合うだけのパフォーマンス上の利得が期待できない可能性が高くなります。

以下では、2ワイドと4ワイドの選択に関わる2つの問題について説明していきます。

- ステート空間: オーバーラップか、あるいはレジスタの追加か
- Pentium® III プロセッサの実装

ステート空間

ステート空間に関しては、新しいステートをMMX/x87のFPレジスタにオーバーラップする方法と、ステートを新規に追加する方法の2つの選択肢がありました。前者の方法には、MMX®テクノロジーによる拡張の場合と同様に、オペレーティング・システム(OS)の変更が一切不要であるという大きな利点があります。しかし、この方法には欠点も数多く存在します。第1に、80ビット・レジスタは8つしか存在しないため、既存の空間では4ワイド128ビット・レジスタを4つまでしか実装できません。2ワイドのフォーマットを採用すればレジスタの数は増やせますが、それでは期待できるパフォーマンス向上の度合いが低下してしまいます。第2に、すでにレジスタの使用状況にほとんど余裕のないIA-32アーキテクチャにおいて、新たなステートをMMXレジスタと共有せざるを得なくなるという問題が発生します。新しいステートのセットを従来のレジスタにさらにオーバーラップさせることは、対処できないほどの複雑性を生むと考えられました。

新しいステートを追加する方法の利点は、実装の複雑さを抑えて、プログラミング・モデルの問題を軽減できることです。しかも、SIMD-FPをMMX命令またはx87命令と同時実行することも可能になります。これはOSメーカーや一般のソフトウェア・メーカーにとっては明らかなメリットです。逆にこの方法の欠点は、新しい機能を使用するにはOSによるサポートが不可欠であるという理由から、インテルがOSメーカーに依存せざるを得なくなることです。しかし、インテルは開発の比較的初期の段階で新しいステートのストア命令とリストア命令を導入することにより、この問題を解決しました。そのため、Pentium III プロセッサの発表時には、すでに最新の各種OSによるこの新しいステートのサポートが実現していました。

不明確な異常状態の発生を防ぐため、新しいステートはすべて x87-FP ステートとは分離されました。図 1 に、新しい 128 ビット・レジスタの構成を示します。この図には含まれていませんが、新たに制御/ステータス・レジスタの MXCSR も追加されています。MXCSR レジスタは、数値例外処理のマスク/アンマスク、丸めモードの設定、flush-to-zero モードの設定、およびステータス・フラグの確認に使用されます。

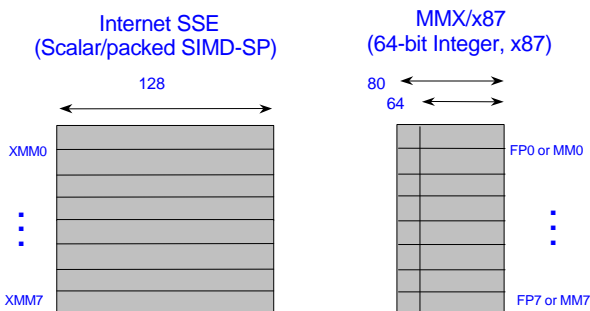


図 1: インターネット SSE 128 ビット・レジスタ

このほか、SIMD-FP 命令の数値例外を処理するための新しい割り込みベクトルも追加されました。

Pentium® III プロセッサの実装

Pentium III プロセッサは、1 つ 1 つの 4 ワイド演算マイクロ命令を 2 つの 64 ビット・マイクロ命令として実装しています。しかし、Pentium III プロセッサは(2 つの実行ポートを持つ)スーパースケーラとして実装されているので、(各命令によって 2 つの実行ユニットが交互に使用されると仮定すれば) 1 クロック・サイクルごとに完全な 4 ワイド SIMD 演算を実行することも可能です。その場合、理論上はアプリケーションのパフォーマンスが 4 倍に高まると考えられますが、マイクロアーキテクチャ内におけるマイクロ命令の圧迫などの理由で、実際のアプリケーションで実現されるパフォーマンスの向上は 2 倍程度にとどまります。将来の 128 ビットの実装では、さらに大幅な性能向上が見込まれています。

スカラ演算とパックド演算

アプリケーションではスカラ演算とパックド演算の両方が必要になることが多いため、新しい SIMD-FP モードではスカラ浮動小数点命令もサポートすることになりました。スカラ演算は x87-FP で実行して、新しいレジスタは SIMD-FP のためだけに使用することもできますが、x87-

FP はスタック・レジスタ・モデルであり、一方、SIMD-FP はフラット・レジスタ・モデルであるため、その方法ではプログラミングのパラダイムが非常に煩雑なものになってしまいます。パラメータを渡す場合も、より多くの変換命令を使用するか、あるいは現在の実装のようにメモリを介して渡すことが必要になります。さらに、x87-FP 演算と SIMD-FP 演算では演算実行の形式が異なるため(SIMD-FP では 32 ビット、x87-FP では 80 ビット)、両者の演算結果に差が生じる可能性があります。

Pentium III プロセッサのスカラ演算の実装は、4 ワイド SIMD-FP のエミュレーションに使用されるため、そのままスカラ命令で使用するには問題がありました。パックド・データ用の命令をスカラ演算で使用すると、結局両方の 64 ビット・マイクロ命令が実行されるため、パフォーマンスに悪影響が出ます。特に除算や平方根演算など、レイテンシが長くパイプライン実行できない処理でパックド命令を使用すると、実行時間の面で非常にコストが高くなります。しかも、プログラムでは、使用されていないスロットでフォルトが発生していないことを確認する処理が必要になります。そこで、この問題に対処するために明示的なスカラ命令が定義されました。スカラ命令は、Pentium III プロセッサでは単一のマイクロ命令しか実行しません。スカラ演算が実行されると、ソース・レジスタの上位 3 つのコンポーネントはディスティネーション・レジスタに直接渡され、演算処理は下位のコンポーネント・ペアでのみ実行されます(次ページの図 4 参照)。したがって、Pentium III プロセッサは、このデータ・タイプの上位半分に対しては何の処理も実行する必要がありません。

SIMD-FP レジスタ上でマスクを使用して選択的に処理を実行することも考えられますが、設計の複雑化が避けられず、また、アプリケーションにおいてもそれほど強い必要性が認められないことから、その方法は採用しませんでした。

並行実行性の向上

SIMD による高いパフォーマンスを実現するには、メモリの帯域幅と実行性能のバランスを取ることが不可欠です。3D グラフィックスやビデオなどのマルチメディア・ワークロードの大半はストリーミング・アプリケーションであり、読み込まれたデータは、多くの場合、1 度使われただけで破棄されます。そして、ローカル・プロセッサのキャッシュ容量はストリーミング・アプリケーションのデータ・セット全体を格納できるほど大きくない

ため、メモリからデータをフェッチする間に実行ユニットでストールが発生することになります。アウト・オブ・オーダー実行やスペキュレーティブ・パイプラインを使用する方法でメモリ・アクセスのレイテンシを隠蔽するにはハードウェア・リソースを大幅に増強する必要があり、ダイ・サイズの拡大やコストの上昇が避けられません。この問題の解決策として考えられるもう 1 つの方法は、プログラミングによって 1 つのデータの処理と次のデータのフェッチをオーバーラップさせ、フェッチのレイテンシを実行時間で隠蔽することです。この方法は、複数の光源が存在する場面の 3D グラフィックス処理など、計算負荷の高い要素を含むアルゴリズムに対して最も効果があります。この方法を実現するため、新しい拡張命令では、プログラマがアーキテクチャ・ステートを操作せずに次のデータをプロセッサ側にプリフェッチできるように、キャッシュの使用を制御する命令が新たに導入されました。

ストリーミング・アプリケーションでは、ストリーミング・データとして扱うべきデータと再利用すべきデータを最も確に判断できるのはプログラマ自身です。たとえば、3D グラフィックスの場合、プログラマはコードや変換行列のデータはキャッシュ内に残し、逆に入力ディスプレイ・リストや出力コマンド・リストはストリーミング・データとして扱いたいと考えるでしょう。このような処理を実現するには、プログラマがデータのキャッシングを管理してキャッシュへの不要なデータの混入を最小限に抑えられるようにするプリミティブが必要です。そのため、新しい命令では、プリフェッチしたデータを格納するキャッシュ階層レベルをプログラマが指定できるよう、プリフェッチ操作のオプションが拡張されました。また、それを補完するものとして、キャッシュメモリへのライト・アロケートを伴わない(ストリーミング)ストアを実行する命令も追加されました。これらのストア命令では、不必要なライト・アロケートが発生しないため、キャッシュ内の必要なデータが別のデータで置換されてしまうことはありません。

プリフェッチ命令は、アーキテクチャ・ステートを一切更新しません。プリフェッチ命令の実装には各プロセッサに固有の部分が多少含まれているので、その機能を最大限に活用するには、それぞれの実装に応じたアプリケーションのチューニングが必要になる場合があります。ただし、設計の目標は、実装の違いによって大きなパフォーマンスの落ち込みが発生しないようにすることにあります。プリフェッチ命令はハードウェアに対して単に

指示を与えるだけで、例外やフォルトを発生させることはありません。

図 2 は、新しい拡張命令の各種機能の連携的な働きによる並行実行性の向上と実行時間の短縮を図解したものです。インターネット・ストリーミング SIMD 拡張命令が導入される前は、読み出しミスによるレイテンシ、実行に要する時間、そして格納ミスによるレイテンシの 3 つを順番に加算したものがそのまま全体の実行時間となっていました。拡張命令の追加により、プリフェッチを使用して読み出しミス・レイテンシと実行をオーバーラップさせることや、ストリーミング・ストア命令により格納ミス・レイテンシを短縮して実行とオーバーラップさせることが可能になりました。さらに、SIMD-FP 命令によって実行時間自体の短縮も実現されています。

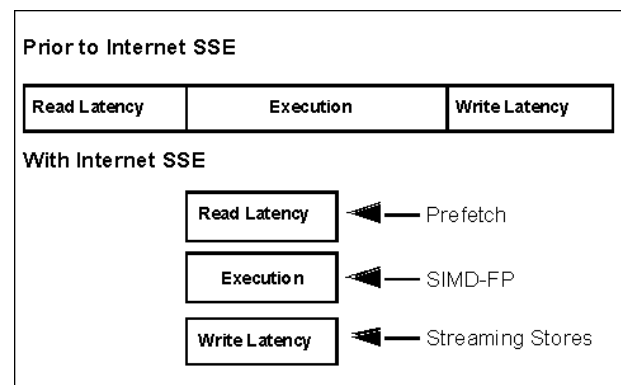


図 2: 並行実行によるパフォーマンスの向上

データ・アライメント

メモリの 16 バイト(128 ビット)境界にアライメントされていないデータへのアクセスを効率的に処理するためのハードウェア・サポートを実装するには、実装面積とタイミングの両面で高いコストが要求されます。開発過程では、マイクロコードによる支援機能を使用してアライメントの合っていないデータを検出・修正する方法と、一般保護フォルトを発生させる方法の 2 つが検討されました。ソフトウェア・メーカからのフィードバックでは、どのメーカも一致して前者の方法は採用すべきではないという意向を示しました。メーカ各社は、原因の追跡調査が困難なパフォーマンスの低下を暗黙的に発生させる可能性がある前者よりも、正しくアライメントされていないデータの存在をフォルトとして明示的に警告する後者の方が望ましいと考えたのです。その結果、メモリ・オペランドを使用するすべての演算命令において 16 バイト・アライメントが必須となりました。ただし、アライ

メントの整合を保証できない場合(ビデオなど)のために、アライメントされていないデータ用のロード命令とストア命令も用意されています。これらの命令は、明示的なコード・シーケンスによってアライメントを合わせる操作を実行した場合と同等、またはそれ以上の処理効率が実現できるように設計されています。

ゼロフラッシュ(Flush-To-Zero)モードとIEEE モード

浮動小数点演算では、厳密な単精度計算を必要とするアプリケーションに適した IEEE モードと、リアルタイム・アプリケーション向けの Flush-To-Zero(FTZ)モードの2つのモードがサポートされています。IEEE 標準に完全に準拠した IEEE モードにより、拡張命令は、厳密な精度とポータビリティが要求されるアプリケーション分野において将来まで幅広く利用できることが保証されています。一方、FTZ モードは、例外のマスクをサポートする高速なハードウェアとの組み合わせにより、演算実行のパフォーマンスを向上させます。FTZ モードは、例外がマスクされている場合、演算でアンダーフローが発生しても0の値を返します。リアルタイム 3D アプリケーションでは、計算精度が僅かに低下しても実用上は問題にならないため、処理の高速性を重視して、ほとんどの場合、FTZ モードが使用されます。

命令セット・アーキテクチャ

インターネット SSE 命令セットには、パックド・データ・オペランドのすべてのペアまたは最下位のペアを並行処理する命令が豊富に用意されています(表 1 参照)。パックド命令(二モニクの末尾が PS である命令)は、図 3 に示すように、オペランドのペアに対する演算を行います。一方、スカラー命令(二モニクの末尾が SS である命令)は、図 4 のように、常に 2 つのオペランドの最下位ペアに対する演算を実行します。

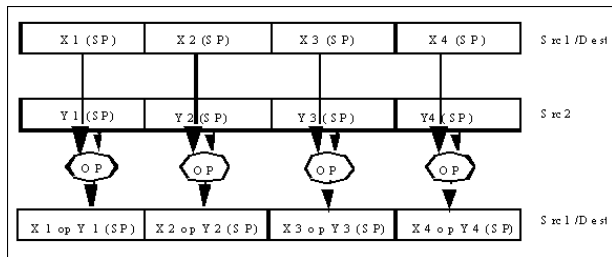


図 3: パックド演算

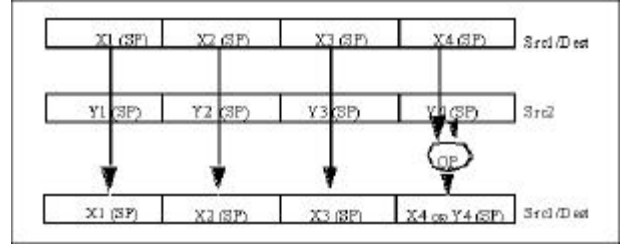


図 4: スカラー演算

		Packed Single	Scalar Single	Packed Integer
Arithmetic	ADD, SUB, MUL, DIV, MAX, MIN, RCP, RSQRT, SQRT	X	X	
Logical	AND, ANDN, OR, XOR	X		
Comparison	CMP, MAX, MIN	X	X	
	COMI, UCOMI		X	
Data Movement	MOV (load/store aligned), MOVUPS (load/store unaligned), MOVLPS, MOVLHPS, MOVHPS, MOVHLPS (load/store), MOVMSKPS MOVSS (load/store)	X X		
Shuffle	SHUFPS, UNPCKHPS, UNPCKLPS	X		
Conversions	CVTSS2SI, CVTTSS2SI, CVTSS2SS CVTPI2PS, CVTSP2PI, CVTTPS2PI		X	
State	FXSAVE, FXRSTOR, STMXCSR, LDMXCSR	X		
MMX Tech Enhancements	PINSRW, PEXTRW, PMULHU, PSHUFW, PMOVMASKB, PSAD, PAVG, PMIN, PMAX			X
Streaming/Prefetching	MASKMOVQ, MOVNTQ (aligned store) MOVTPS (aligned store) PREFETCH SFENCE			X

表 1: インターネット SSE 命令セット・アーキテクチャ

基本的な命令

基本的な命令として、ロード命令、ストア命令、加算、乗算、減算、除算、平方根などの算術命令、および、新たに追加された制御/ステータス・レジスタと保存/復帰ステートにアクセスするための命令が用意されています。

キャッシュの制御

前述したように、マルチメディア・アプリケーションで高速なパフォーマンスを実現するには、あるデータ・ブロックの実行と次のデータ・ブロックのフェッチをある程度オーバーラップさせて並行的に行う必要があります。そのため、新しい拡張命令セットでは、アプリケーションでのオーバーラップをプログラマが制御するための PREFETCH 命令が追加されました。また、この命令を使用すればキャッシュ階層へのデータの配置を制御できるだけでなく、テンポラルな(何度も繰り返し使用される)キャッシュ・データの格納位置と非テンポラルな(読み出し後 1 度使われただけで破棄される)キャッシュデータの位置を区別することも可能になります。現在は 4 つのタイプのプリフェッチが実行可能ですが、将来的には命令がさらに追加定義される可能性があります。なお、PREFETCH 命令は SIMD に対応していないアプリケーションでも使用できます。

ストリーミング・ストア命令である MOVNTPS 命令(パックド単精度 FP)と MOVNTQ 命令(パックド整数)では、プログラマが 1 つ 1 つの命令ごとにライトコンバイン(WC)メモリ・タイプを指定できます。しかもこの指定は、メモリ・タイプ・レンジ・レジスタ(MTRR)またはページ属性テーブル(PAT)によってライトバック(WB)のメモリ・タイプが割り当てられているメモリに対しても有効です。この機能により、プログラマは命令ごとの単位でさまざまな WC メモリ・タイプ(ライトコンバイン、ライトコラプス、キャッシュ不可能、非ライトアロケート)の利点を最大限に活用できます。

フェンス命令(FENCE)

ソフトウェア制御による効率的な一貫性の維持を実現にするため、新しい拡張命令セットには、ストアフェンス命令(SFENCE)が含まれています。この命令を使用すると、フェンスの直前にストアされたすべてのデータは、次の一連のストア操作が完了するまで、フロントサイド・バス上で監視されます。SFENCE の使用が特に有効なのは、プロセッサからグラフィックス・アクセラレータへの書き込み時、あるいは、データの生成側と参照側が WC(ラ

イト・コンバイン)メモリ・タイプ・セマンティックでのストア操作を使用してデータをやり取りする際に、その間のデータの監視性を確保したい場合などです。

比較命令と条件分岐

基本的な単精度 FP 比較命令(CMP)は、既存の MMX 比較命令(PCMPSEQ、PCMPGT)に類似した操作を行い、比較の結果に基づいて値ごとにすべて 1 またはすべて 0 の冗長なマスクを生成します。そのため、比較の直後にこのマスクを使用して論理演算(AND、ANDN、OR、XOR)を行い、その結果によって条件の移動を実行することが可能です。また、MOVMSKPS/PMOVMSKB 命令を使用して、マスクの(各コンポーネントの最上位の)4 ビットを整数レジスタに移動することもできます。これらの命令は、3D ジオメトリのクリップ範囲のチェックやフロント/バックフェイス・カリングのチェックなど、データに依存する分岐処理を単純化する役割を果たします。その意味では、多くのソフトウェア・メーカから出されていた要望を満たす命令でもあります。

条件に関するもう 1 つの重要な使用モデルは、2 つの値(パックド・データまたはスカラ・データ)のどちらかを最大値または最小値として求める操作です。この操作は、すでに述べた条件的移動の場合とまったく同じ方法で実行することも可能ですが、MAX/MIN 命令または PMIN/PMAX 命令を使用すれば、1 つの命令を実行するだけですみます。これらの命令は、比較のための減算からのキャリーアウトを直接使用して、結果として返すべきソースを選択します。たとえば 3D ジオメトリやラスライズ操作では、MINPS/PMIN 命令の使用によってカラー・クランプの処理効率を高められます。また、多くの音声認識エンジンでは、その中核的なコンポーネントである隠れマルコフ・モデル(HMM)の評価のために、実行時間の約 80%が費やされていますが、PMIN 命令は、このカーネルの処理を 33%高速化し、アプリケーション全体のパフォーマンスも 19%向上させます。

CMP 命令では、比較演算の完全なセットを提供するために、8 ビットの即値を使用して基本的な比較プレディケートである EQ、LT、LE、UNORD、NEQ、NLT、および NLE がエンコードされます。また、これらのプレディケートを使用してソース・オペランドを入れ替えれば、このほかの 4 つのプレディケートも得られます。このように即値を使用してプレディケートをエンコードすることにより、比較演算に割り当てられるオペコードの数が大幅に削減されています。

データ操作

SIMD 演算による処理の高速化を実現するには、データを SIMD フォーマットに適合するように効率的に再編成する操作が不可欠です。たとえば、4 ワイド SIMD-FP フォーマットによる 3D ジオメトリの変換処理では、各頂点ごとに 4 つの成分(x、y、z、w)を一括して処理する方法と、各成分ごとに複数の頂点のデータをまとめて処理していく方法が考えられます。各頂点ごとに 3D データを構造体としてまとめる方法は、ディスプレイ・リストが各頂点のデータの配列になるため、AOS (Array-of-Structures)と呼ばれます。一方、データを成分ごとに処理する方法は、x、y、z、w の各成分データの配列を含む 1 つの構造体を使用するため、SOA (Structure-of-Arrays)と呼ばれます。ディスプレイ・リストの SIMD フォーマットとして理想的なのは、後者の SOA の方です。AOS 方式は次の 2 つの理由により、やや効率が低くなります。1) すべての SIMD 演算スロットが活用されるとは限らない(頂点の w 成分は不必要な場合がある)。2) 通常、水平換算操作($a * x + b * y + c * z$ などの内積計算)が必要になるが、この操作は複数の SIMD スロットを使用して結果のスカラ値を 1 つだけ生成する。水平換算とともにレイテンシの長い演算(平方根計算や除算)が使用される場合、効率低下の幅はさらに大きくなります。

3D ジオメトリの標準 API やラスタライズ・グラフィックス・コントローラが直接 SOA をサポートしていない場合など、データのスタティックな再編成は不可能と思われるケースも少なくありません。しかし、新しい拡張命令セットには、データを効率良く並べ替えて理想的な SOA フォーマットに変換したり、その逆に元の形式に戻したりするための以下のようなデータ操作命令が含まれています。

- UNPCKHPS/UNPCKLPS 命令 -- MMX のアンパック命令に類似した命令で、レジスタまたはメモリ・オペランドの上位部/下位部からの float 値をインターリーブします。
- SHUFPS/PSHUFW 命令 -- ブロードキャスト、回転、交換、反転など、ソース・オペランドのデータに対するさまざまな入れ替え操作をサポートします。
- MOVHPS/MOVLPS 命令 -- 隣接していない 4 つの AOS データ構造体から各頂点の成分データを効率良く集めて 1 つの 128 ビット・レジスタ(SOA)に格納する 64 ビットのロード/ストア命令で、SHUFPS と組

み合わせて使用します。MOVHPS/MOVLPS 命令は、SOA を AOS に逆変換するためにも使用できます。

- PINSRW/PEXTRW 命令 -- MMX レジスタ内のワード値に対する、メモリまたは 32 ビット整数レジスタからの分散/集合操作をサポートします。たとえば、テクスチャの成分データを集めるためや、SIMD ルックアップ・テーブルをサポートするために使用されます。また、PINSRW 命令は、隠れマルコフ・モデル(HMM)を使用した音声認識のパフォーマンスをカーネル・レベルでは 22%、アプリケーション・レベルでは 13%程度向上させます

SIMD フォーマットについては、さまざまな種類の 3D 変換/照明ビルディング・ブロックや、より完全なジオメトリ・パイプラインを使用した多様なテストが行われました。その結果、SIMD 演算を有効に活用するための上記の方法のうち、最高のパフォーマンスが得られるのはスタティック SOA を使用した場合であることがわかりました。照明効果(周辺光、拡散光、鏡面反射)をフルに実装したジオメトリのベンチマークでは、AOS フォーマットで直接演算を実行した場合、すでに述べた理由により、スタティック SOA の約半分のスループットしか実現できません。SHUFPS/MOVHPS/MOVLSP 命令の組み合わせを使用してデータを AOS から SOA、または SOA から AOS へダイナミックに再編成する方法では、スタティック SOA を使用する場合に比べて 20~25%のオーバーヘッドが余分に必要ですが、このパフォーマンスは、それ以外の方法で実現できるパフォーマンスよりも 6~10%程度優れています。しかも、このオーバーヘッドは定数値であるため、(光源の追加などにより)実行する SIMD 演算の回数が増大すれば、相対的に小さくなっていきます。ほとんどの API は頂点ベースの形式でディスプレイ・リストを扱うため、AOS のまま直接演算を実行することによりプログラミング・モデルが単純化できる場合もあります。AOS フォーマットの演算で必要となる水平操作のパフォーマンスを高めるために、拡張命令セットには MOVLHPS/MOVHLPS 命令が追加されています。この 2 つの命令は、加算、減算、論理演算など、広範な水平操作のエミュレーションをサポートします。ただし、一般には、やはり SOA フォーマットを使用した方がより高いパフォーマンスを実現できます。また、SOA フォーマットへのダイナミックな変換に使用する転置コードは、実際上は、コンパイラの pragma や組み込み関数を活用して隠蔽することができます。

変換

変換操作としては、整数と浮動小数点値の相互変換、スカラ・データとパックド・データ、レジスタまたはメモリ上のソースとディスティネーション、命令で暗黙的に使用される丸めモード、整数オペランドのサイズ(バイト、ワード、ダブルワード)など、さまざまなタイプの操作が可能です。しかし、実際の使用頻度の差や別の命令シーケンスによるエミュレータの可能性を考慮すれば、これらの要素の順列組み合わせによる多数の変換操作をすべて命令として実装することは現実的とは言えません。最終的な命令の定義は、以下のような方針に基づいて決定されました。

- 整数と浮動小数点値の変換については、パックド・データの SMID-FP と MMX[®]テクノロジー形式の変換 (CVTPI2PS、CVTTPS2PI、CVTTSS2SI)、および、スカラ・データの Scalar-FP と IA-32 整数の変換 (CVTSS2SI、CVTTSS2SI、CVTSS2SI)の両方をサポートする必要がある。
- ディスティネーションはレジスタとする。必要があれば、結果はストア命令を使用して明示的にメモリに移動できる。
- 浮動小数点値から整数への変換時に、制御レジスタの丸めモードの変更によって発生しがちなシリアライゼーション・パフォーマンスのペナルティを回避するため、CVTTPS2PI/CVTTSS2SI 命令では、切り捨ての丸めモードを暗黙的にエンコードする。
- インターネット・ストリーミング SIMD 拡張命令では、変換対象の整数としてダブルワード値のみをサポートする。ダブルワードと、ワードまたはバイトとの相互変換は、既存の MMX パック/アンパック命令を使用して効率良く実行できる。

逆数、および逆数の平方根

除算と平方根の演算は、ジオメトリ処理の基本的な要素の1つです。たとえば、座標変換処理では、 x 、 y 、 z の各座標の値を透視座標の W で除算する操作が多用されます。また、鏡面反射の処理にはべき関数が使用されますが、この関数は通常、除算を使用して近似的にエミュレートされます。さらに、ジオメトリ処理で同様に多用される操作の1つである正規化でも、 $1/\sqrt{\quad}$ の計算が必要になります。新しい拡張命令セットでは、これらの演算を最適化するために、近似演算命令の RCP と RSQRT が導入されました。この2つの命令はハードウェアのルック

アップ・テーブルを使用して実装されており、仮数部が12ビットであるため、本質的に、IEEE 標準に完全準拠した仮数部24ビットの DIV および SQRT よりは精度が低くなります。しかし、これらの近似命令には、フル精度の命令に比べて大幅に高速であるというメリットが備わっています。より高い精度が必要な場合は、近似命令とシングル Newton-Raphson (N-R)イテレーションを併用すれば、IEEE 命令とほぼ同等(仮数部22ビット)の精度が得られます。この逆数演算のための N-R イテレーションでは乗算2回と減算1回が必要になるため、総計のレイテンシはやや大きく、IEEE 命令よりもスループットが低くなります。最大の精度が必要ない基本的なジオメトリ・パイプラインの処理では、近似命令を使用することにより、15%程度のパフォーマンス向上が実現できます。

符号なし乗算と、バイト・マスク書き込み

D3D チームなどの開発者たちと議論を重ねるなかで、3D ラスタライズのパフォーマンス効率が高いのは、符号なしの MMX 乗算命令が存在しないためであることが判明しました。ラスタライズは、元来、符号のないピクセル・データを扱う処理ですが、既存の PMULHW 命令では符号付きデータに対する演算しか実行できません。符号なしデータ用の PMULHW 命令を追加することにより、符号に関するデータ変換のオーバーヘッドが解消され、アプリケーション・レベルで8~10%程度のパフォーマンス向上が実現しました。

バイト・マスク書き込み命令の MASKMOVQ は、特定のラスタライズ処理や画像処理アプリケーションの効率化を目的として追加された命令です。この命令は、次のような優れた機能をサポートします。

- PCMPGEQ/PCMPGT 命令で生成されたバイト・マスク、またはメモリからロードされたバイト・マスクを使用して、もう一方のソース・オペランドのバイト値を選択的にメモリへ直接書き込みます。このマスクは、データとともに並行的に効率良くメモリ・サブシステム(ライトコンバイン・バッファ、バス・キュー・エントリ、バス・バイト・イネーブル)を通じて転送されます。条件的移動や条件分岐を使用した従来の実装では、実行パイプラインで処理されるマイクロオペレーションがかなり多くなることや、分岐予測ミスが発生する可能性があることから、バイト・マスクによるパフォーマンスの向上は小幅にとどまっていた。

- MASKMOVQ 命令は、その他の非一時的なストリーミング・ストア・キャッシュバリティ命令と同様、WC メモリ・セマンティックを実装しています。そのため、不要なオーナーシップ読み出しのために帯域幅が浪費されることがなく、また、キャッシュは完全にバイパスされるので、一時的なキャッシュ・データが乱れる恐れもありません。

バックド・データの平均値

動き補正は、MPEG-2 デコード・パイプラインの主要な機能の 1 つです。動き補正は、キー・フレームの間のデータを補間して、出力画像ストリームの各フレームを再構成する処理を行います。このデータ補間処理では、主として、異なるマクロブロック(16x16 ピクセルの大きさを持つ画像領域の単位)に属するピクセル・データの平均値を求める方法が使用されます。MPEG-2 の規格上、平均演算の結果は最も近い整数値に丸めることが必要です。計算値が整数のちょうど中間の値である場合は、より大きい整数値に切り上げられます。つまり、この規格が要求する平均演算は 9 ビット精度の演算に相当します。MMX 命令の精度は 8 ビットまたは 16 ビットですが、データの並行処理性を高めるには 8 ビットの命令を使用する方が望ましいと言えます。PAVG 命令は、9 ビット精度の平均演算を実行することにより、8 ビット命令の有用性を高める役割を果たします。複数の演算命令を使用して結果を累算するアプリケーションでは、ワード値用の PAVGW 命令を使用して、精度をさらに高めることができます。

現在、Pentium® II プロセッサ・ベースのシステム(266MHz)で動作する DVD プレーヤの動き補正では、メモリ・レイテンシと実行時間のバランスが取れています。PAVG 命令は、カーネルでの処理を 25% 高速化しました。PAVG 命令を使用して動き補正コードをインストールしたプレーヤでは、アプリケーション・レベルで 4~6% の高速化が実現されています(高速化の度合いはビデオ・クリップの種類によって異なります)。より解像度の高い HDTV デジタル・テレビのフォーマットではさらに効果が大きく、アプリケーション・レベルで 10% 程度のパフォーマンスの向上を実現できます

バックド・データの絶対差の和

ビデオ・エンコードのパイプラインは多数の段階に分かれた非常に複雑な過程ですが、その実行時間のかなりの部分は、動き評価の処理によって占められています(現状

では 40~70%)。動き評価の処理では、現在のフレームのサブブロックを直前のフレームや直後のフレームの同じ領域と比較することにより、最も良くマッチするデータが特定されます。この処理を行うことにより、マッチしたデータの位置を示すベクタと、それ以外の位置の差分データだけでサブブロックの情報を表すことが可能になり、元のサブブロックのデータ全体を使用する場合に比べて出力ストリームを大幅に圧縮できます。

動き評価では、通常、比較計量法として平方差の和(SSD)または絶対差の和(SAD)のどちらかの方法が使用されます。どちらの方法にも、評価対象の種類によって異なるさまざまな長所・短所がありますが、マッチの質を評価するための計量法として総合的にみれば、両者に大きな差はありません。

SSD は、因数分解のテクニックにより、符号なしの乗算-累算(バイトからワード)演算を使用して実装することが可能です。ただし、その場合は、累算値が取りうる値を問題なく扱うために 24 ビットの精度が必要になり、汎用のデータ・タイプでは効率的に対処できません。そこで、この方法の代わりに PSADBW 命令を使用すれば、同時に 8 バイトの処理が可能なバイト・レベルの並行実行性を保ちながら、累算値を 16 ビット幅のワード値の範囲内で扱えるようになります。動き評価処理の内部のループでは、1 つの PSADBW 命令がほぼ 7 つの MMX 命令に匹敵する効果を持ちます。このループで MMX 命令の効率が高いのは、主として、MMX テクノロジーでは符号なしのバイト演算がサポートされておらず、バイト演算をゼロ拡張とワード演算によってエミュレートする必要があるためです。PSADBW 命令は、このループの高速化により、動き評価のパフォーマンスを約 2 倍に高めることが確かめられています。

まとめ

インターネット・ストリーミング SIMD 拡張命令は、量産型 PC のプラットフォームにおいて、魅力あふれる新次元のビジュアル・コンピューティングを実現します。Pentium® III プロセッサでは、新しい単精度 SIMD-FP 命令セット・アーキテクチャにより、浮動小数点演算のパフォーマンスを約 2 倍に高めるといった目標が達成されました。この高速化によってリアルタイム 3D アプリケーションのパフォーマンスが大きく向上するため、Pentium III プロセッサを搭載したシステムでは、複雑な空間のレンダリングをリアルタイムで実行することも可能になります。この新しい命令セットは、インテルにとって、PC プ

ラットフォームにおけるビジュアル・パフォーマンスのさらなる向上を目指す上で、大きな前進を意味する重要な1歩となりました。

ビデオのパフォーマンスを高める SIMD 整数命令が追加されたことにより、画質と圧縮率のトレードオフを考慮しながら、MPEG-1 または MPEG-2 フォーマットのビデオをリアルタイムでエンコードすることも可能になります。新しい拡張命令は DVD 再生にも効果を発揮し、Pentium III プロセッサのタイムフレームの中で、その他のプロセスをマルチタスクで処理する余裕を十分に残しながら、毎秒 30 フレームのデコードを実現します。Pentium III プロセッサの動作周波数の高速化がさらに進めば、1 メガピクセル HDTV フォーマットのデコードも可能になるでしょう。当初は動き補正をサポートするハードウェアが必要ですが、プロセッサの一層の高速化とともに、やがて完全にソフトウェアのみによるデコードが実現できるようになります。また、新しい命令の効果により、ホーム・ビデオの編集でも従来の写真の編集に近い操作性が実現されます。

新しいプリフェッチ命令とパックド・データ整数命令は、音声認識アプリケーションでのエラー率の低減と辞書サイズの縮小を実現します。また、新たに導入されたキャッシュアビリティ命令は、マルチメディア・アプリケーション全般において、メモリ・アクセスと実行の並行性を高めるために有効です。

命令定義チームは、命令に関するさまざまな要望の検討と、アプリケーションで実現可能なパフォーマンス向上の分析に精力的に取り組み、非常に短い期間で新しい命令セット・アーキテクチャを完成させました。新たに追加された命令は、3 つの実装チームによる厳正なチェックにより、命令セットに含めるに値すると認められたものばかりです。その上で、最終的な命令定義は、パフォーマンス上の利得、ダイ・サイズやタイミングへの影響、コードのポータビリティなど、多数の制約を十分に考慮して決定されました。Pentium III プロセッサにおけるインターネット・ストリーミング SIMD 拡張命令の実装コストは、ダイ・サイズの観点からみれば約 10% にすぎません。これは、Pentium[®] プロセッサおよび Pentium[®] II プロセッサに MMX[®] テクノロジーを実装するコストとほぼ同等です。

謝辞

Srinivas Chennupati、Patrice Roussel、Vladimir Pentkovski、Mohammad Abdallah の各氏をはじめ、開発のために尽力した定義チームの全メンバーに感謝します。IA-64 の定義とセマンティックの導入は、多くの命令の一貫性を維持する上で大きな力となりました。また、定義策定中にさまざまな助言をいただいた、Glenn Hinton、Bob Colwell、Fred Pollack の各氏にも感謝の意を表します。

参考資料

- [1] Alex Peleg ほか著 『MMX[™] Technology Extension to the Intel[®] Architecture』、DTTC Proceedings 1996 (インテルの社内ドキュメント)
- [2] Millind Mittal ほか著 『MMX[™] Technology Architecture Overview』、インテル・テクノロジー・ジャーナル 1997 年第 3 四半期号、
<http://developer.intel.com/technology/itj/q31997.htm> (英語)

著者紹介

Shreekant (Ticky) Thakkar、マイクロプロセッサ・プロダクト・グループの主任プロセッサ・アーキテクト。Pentium[®] Pro のマルチプロセッサ・アーキテクチャの開発に従事したのち、Pentium[®] III プロセッサでは、インターネット・ストリーミング SIMD 拡張命令の開発リーダーを務める。

電子メール・アドレス: ticky.thakkar@intel.com

Tom Huff、オレゴンのマイクロプロセッサ・プロダクト・グループに所属するアーキテクト。コア・チームのアーキテクトの 1 人として、IA-32 アーキテクチャのインターネット・ストリーミング SIMD 拡張命令の定義策定に貢献。現在は、Willamette プロセッサ・プロジェクトでマルチメディアのパフォーマンス解析に従事。ミシガン大学で電気工学の理学修士号および博士号を取得。

電子メール・アドレス: tom.huff@intel.com