Linux/Alpha活用講座

清水　尚彦　<nshimizu@keyaki.cc.u-tokai.ac.jp>

第14回
Alpha vs. Athlonのベンチマーク

月日の経つのは速いもので、もう雑誌の世界はY2Kに突入です。世界中で騒がれている割には、日本は比較的のんびりしていると思うのは私だけでしょうか？

マイクロプロセッサフォーラムから

毎月のように新しい動きのあるプロセッサビジネスですが、毎年10月に開催されている「Microprosessor Forum」というプロセッサ関係者のお祭りでは、特に多くの新しい発表がされます。これは8月の「Hot Chips Symposium」と2月の「International Solid-State Circuits Conference（ISSCC）」などとともに有名な、マイクロプロセッサ関連のお祭りの1つですが、特にパソコン関係のプロセッサの出展が多いのが特徴です。今年のフォーラムでは64bitプロセッサの発表が多くありました。

詳しい情報は入手していませんが、インテルがMerced（開発コード名、正式名は「Itanium」）の詳細を発表したり、IBMがPower4を発表したり、CompaqがEV8（21464）を発表したりしています。Powerのシリーズはなかなかつぼを押さえた構成を取ると昔から思っていますが（PowerPCではないのに注意）、今回もメモリバンド幅に注意を払った良い設計がなされているようです。もっとも、5500ピンもあるようなチップをモジュール実装できる会社は世界に数社しかないので、誰でも使えるわけではありませんし、モジュールにするとおそらく500Wくらいの消費電力になるといわれているので、一般のユーザーに簡単に入手できるようなマシンとして出荷されることはなさそうです。

実は大きな驚きをもって迎えられたのはAMDの発表で、x86のアーキテクチャを拡張して64bitにしたという発表をしました。AMDの発表は次のURLから入手できます。

```
http://www.amd.com/products/cpg/mpf/speech/
slides99.ppt
```

インテルがx86のアーキテクチャに見切りをつけて新しい64bitプロセッサを発表しているのとは対照的に、AMDは互換性の大切さを強調しています。SPECmarkから見た限り、整数演算の性能ではx86が見劣りすることはなくなってきていますので、AMDの方向性は悪くないのかもしれません。また、AMDは問題になっている浮動小数点性能についても独自の拡張をすることで、RISCプロセッサ（ターゲットはAlphaです）に対抗できるとしています。さらにこの対決の興味深い点は、AthlonがAlpha 21264と同じバスインターフェイスを採用していることです。それによってバス性能については同じ土俵の上で戦えることになります。もっとも出荷されているマシンは同じチップセットを使っているわけではないので完全に同じではありません。ともあれ、数値計算のユーザーは性能さえ良ければプロセッサにはこだわらないという人が多いので、AMDの発表は気になるところでしょう。

Aベンチマーク対決

ちょうどタイミング良く編集部がAMDのAthlonプロセッサの載ったマシンを用意してくれたので、今月は「Alpha vs. Athlon」のベンチマーク対決としたいと思います。Athlonのベンチマークはいろいろな雑誌で出ているし、SPECの値も公開されているので、よくご存じの方も多いと思います。

```
http://www.specbench.org/
```

しかし、同じことを繰り返してもつまらないので、あまり他の雑誌で見掛けないような、数値計算のいくつかの側面を強調したベンチマークを対象にしました。そのうちのいくつかは私のオリジナルです。私のものも含めて、すべてWebページから入手できますので、読者の皆さんにも自分のマシンとの比較を簡単にしていただけます。

Athlon(600MHz)に対抗するのはご存じCompaqの「XP1000」(21264-500MHz)とVisual Technologyの「VT-5/600」(21164A-600MHz)になります。Athlonも700MHzが発表されているし、Compaqも667MHzのチップを出荷しています。また、Alphaの開発元であるAlpha Processor社(API)は750MHzを発表していたりしますが、今回は手元にあるマシンで評価しました。OSは、AthlonがStorm Linux 0.99で、XP1000とVT-5/600はDebian GNU/Linux 2.1です。数値計算が中心なのでカーネルのバージョンはあまり関係ないのですが、MPIのテストで、AlphaでLAMを動作させるためには2.2.xが必須になるため、カーネルはDebianの配布のものより新しくしています。

SPECのベンチマークでもそうですが、大事なのは公表された数値の大小ではなく、自分のプログラムがどれだけ速く動作するかなのです。その意味では1番良いベンチマークは自分の使うプログラムなのです。ところが、複数のシステムで自分のプログラムを走らせて確認するような機会を持てる人は少数です。そこで、公表された数値の中から自分のプログラムの特徴に近いものを選択して評価の基準を作る必要があります。この記事がそのような選択のための1つの参考になれば幸いです。

写真1 ベンチマークに使用したツクモオリジナルショップブランドのAthlonマシン

表1 ベンチマークに使用したAthlonマシンのスペック

発売元	九十九電機株式会社
型番	TS-A600/13J
CPU	600MHz AMD Athlonプロセッサ
システムバス	200MHz AMD Athlonシステムバス
1次キャッシュ	128KB(CPU内蔵)
2次キャッシュ	512KB(CPU内蔵)
メインRAM	128MB(SDRAM-DIMM)
ビデオRAM	32MB(SGRAM)
グラフィックアクセラレータ	Matrox社製 Millennium G400 (AGP 2xモード対応)

この製品に関するお問い合わせ
システムサポートセンター 03-3251-7395
http://www.tsukumo.co.jp

A数値計算ユーザーに気になるプロセッサの特徴

ベンチマークの数値をいきなり出しても、数字の大小だけではあまり有効な情報になりません。巷の記事では数値の大小だけでプロセッサの性能を論じることが多いのですが、もう少し突っ込んだ議論がなされるべきと私は思っています。そこで、ベンチマークの説明をする前に、数値計算のユーザーにとって気になるプロセッサアーキテクチャ上の特徴についてポイントを押さえておきましょう。アーキテクチャといっても命令セットの話ではなく、いわゆる内部アーキテクチャの話です。

命令セットアーキテクチャは、1つの制約条件として性能に影響する場合も多いのですが、90年代に設計された多くのアーキテクチャ、にとってはそれほど大きな制約はないのです。それ以前のものは制約が多いことがありますが、それでも、AMDのように独自の拡張によって解決する道があるので、とりあえず置いておきましょう。

といっても、独自拡張で困るのはソフトウェアの互換

性なので、完全に無視するわけにもいきませんね。x86を数値計算に使う場合に1番困るのはx8087以来使われている浮動小数点レジスタの構成です。実はIntelはここまでプロセッサが高性能になるとは思っていなかったのではないでしょうか。8087の浮動小数点レジスタはスタックの構造になっていて、一般的なレジスタマシンのような自由な使い方ができない上に、その数も8本しかありません。メモリとプロセッサ性能がバランスしている時代には問題なかったのですが、現在の高性能なシステムを支えるには、レジスタの数も構成も役不足です。AMDは64bit化にともなって「Technical Floating Point instructions（TFP）」という新しい命令セットを定義してこの問題の解決に当たっています。彼らはこれによってRISCプロセッサとSPECfp性能でも並ぶと予想しています。

●メモリバンド幅

プロセッサが計算を実行するためには、必要なデータをメモリから十分に供給できる必要があります。メモリバンド幅が十分でなければ、いくら高速なプロセッサをもってしても、どうしようもありません。計算量の多いものの中には、ブロック化などによってメモリバンド幅を低く押さえられる計算もありますが、一般のユーザーに使いやすいプロセッサは、十分なメモリバンド幅を持つものです。

例えば、LINPACKベンチマークの中に有名なDAXPYというサブルーチンがありますが、この1要素の計算には2つのメモリ読み出し（ロード）と1つの書き込み（ストア）が発生します。1要素は2つの浮動小数点演算を含んでいるので、演算あたり1.5個のデータ転送が発生していることになります。ちなみに倍精度の計算をすると1回のデータ転送あたり8バイトの転送が発生します。

Linpackの性能報告を見ると、性能上位の単一プロセッサのマシンでは数100MFLOPSという性能を出していることが分かります。これを例えば300MFLOPSとしても「300×1.5×8＝3.6GB/秒」というメモリバンド幅が必要になります。今、販売されているプロセッサで、このメモリバンド幅を供給できるものはそれほど多くはありません。もちろん、キャッシュからの転送性能はもっと速いものもありますから、Linpackのベンチマークのうち、ベクトル型スーパーコンピュータが苦手な100元と呼ばれる小さな問題では、RISCプロセッサのキャッシュに入るためRISCプロセッサが大変善戦しています。

IBMは、Powerシリーズでメモリバンド幅を比較的多めに用意してきましたが、Power4では10GB/秒とスーパーコンピュータクラスのメモリバンド幅を用意することになっています。AlphaもEV7の世代（21364）ではDirect Rambusのインターフェイスを直接プロセッサに用意し、メモリバンド幅の大幅な向上を計ることになっています。このように数値計算の高速性を狙っているプロセッサでは、メモリバンド幅を大きく取ろうとする傾向にあります。

ところが、ここに普段あまり気が付かれない伏兵がいます。というのは、一般にRISCプロセッサのOSとして採用されるUNIX系のOSでは（NTもそうですが）、仮想記憶を用いています。また、不思議なことに、RISCプロセッサの多くは、仮想記憶の管理単位であるページの大きさが比較的小さなものが多いのです。さらに仮想記憶を実現するにはアドレス変換といって、メモリアクセスごとに変換テーブルを参照して物理アドレスを求める必要があるのですが、まともにテーブルを引くと性能は大きく落ちるので、普通、変換バッファもしくはTLBと呼ばれる高速化機構を用います。この変換バッファはプロセッサ内部に用意するため、シリコン面積を他の要素と取り合う関係からあまり大きなものを用意できないのが実状です。

そこで、プロセッサメーカーは何かしら設計の基準となるデータを元に、変換バッファのエントリの数を決めています。現在のところ、プロセッサの設計に最も影響のあるベンチマークがSPECマークと呼ばれるもので、この浮動小数点ベンチマークが数値計算関連の世界では最も参照されるものだといっても過言ではないでしょう。問題はSPECのベンチマークはいろいろなマシンで動作することを考えているため問題のサイズを大きくできないので、比較的小さな問題が多いことです。大体8MBのキャッシュがあると、SPECのベンチマークの多くはキャッシュヒットするようになって大きく性能が向上すると考えられるようです。そこで、プロセッサベンダはSPECの数値に影響するぎりぎりまで不要な資源を削減することになるため、変換バッファを十分なだけ持っているRISCプロセッサはほとんど無いのが現状です。

これに対して、数値計算のユーザーは大規模な計算を

することが多いのです。たとえば1万次元の密行列を倍精度で定義すると、800MBのデータ量が必要になります。メインフレームと呼ばれる大型コンピュータやベクトル型スーパーコンピュータでは変換バッファを多く用意したり、アドレス変換そのものをしないなどの方策で変換バッファの不足を解消しています。

これから分かるように、大規模数値計算のユーザーにとって重要なメモリバンド幅は、大規模な行列などをアクセスするときに発生する非常に間の空いた、いわゆる大きなストライドを持つ参照パターンを行ったときのメモリバンド幅であるといえます。ところが、こういった性能はプロセッサベンダにあまり有利でないからか、ほとんど参考にされることはありません。

●メモリレイテンシ

バンド幅とともに数値計算に大きく影響するものがメモリレイテンシです。これもあまりなじみのない言葉なので説明が必要だと思います。簡単に言うと、メモリの参照を要求してからデータが到着するまでの時間がメモリレイテンシです。大規模な計算では、キャッシュは役に立たないことが多いので、キャッシュではなく主記憶までのレイテンシが性能を決めることになります。レイテンシが問題であることは多くのプロセッサベンダも気がついていると思いますが、対応はまだまだゆっくりとしたものです。一般的なDRAMを使ったシステムでは、注意して設計しても100nsを切るレイテンシを実現することは難しく、大規模なシステムを作ると、うっかりすると500nsくらいには平気でなってしまいます。

先ほどのDAXPYを例に取ると、まったくレイテンシの対策がなされていないプロセッサで、レイテンシが0なら300MFLOPSの性能を持つとすれば、レイテンシが100nsになると、何と5.5MFLOPSに性能は低下してしまいます。いかにレイテンシが大事か良く分かると思います。レイテンシを実効的に短くするためにいくつかの技術が使われます。

もっとも普及している技術はキャッシュメモリです。キャッシュはまさに、よく使うデータについてレイテンシを短くするための技術なのです。しかし、大規模計算では先ほど述べたようにキャッシュの参照はあまりヒットしなくなり、効果が急速に低下します。

大規模計算で特に有効なものがソフトウェアの制御によるプリフェッチと呼ぶ技術です。これはレイテンシを見越して、それだけ早くデータの先読みをキャッシュに対して行っておくものです。ただし、コンパイラがプログラムの動作を理解して必要なタイミングでプリフェッチを出すのは容易ではなく、なかなか普及しているとはいいがたいものです。

もう1つよく使われる技術は、アウトオブオーダー実行です。この仕組み自体はレイテンシ対策としてではなくもっと一般的な技術として開発されましたが、性能に1番影響するところはレイテンシの隠蔽が可能になるという点です。アウトオブオーダー実行では、レイテンシを待つ間、メモリ参照命令の先の命令を発行することができます。そこで、ただ待っているのではなく有効な命令を実行しているならば、その分、実質的にレイテンシは隠蔽されているとみなすことができます。

プリフェッチにしてもアウトオブオーダーにしても、レイテンシを隠蔽するにはレイテンシの期間に有効な演算を行う必要があります。ということは、その期間に演算を行うだけのデータが用意されていなくてはなりません。先ほどから使っているＤＡＸＰＹで説明すると300MFLOPSのDAXPY性能を有するプロセッサが100nsのレイテンシを隠蔽するには、「100nS×300MFLOPS＝30個」の演算を行うだけのデータをどこからか供給する必要があります。これは先ほどのデータ転送量との関係から見て「30×1.5×8＝360B」のデータをどこからか供給することに相当します。

プリフェッチであればこれをキャッシュに準備しますが、アウトオブオーダーではこのデータはリオーダーバッファに格納されることになります。つまり、この程度のレイテンシを隠蔽するためにリオーダーバッファのエントリは45個も必要ということになるのです。共有メモリの大規模な並列プロセッサを構成する場合には、レイテンシはもっとずっと大きくなるのが普通であり、今後のプロセッサではリオーダーバッファの大きさはもっとずっと大きなものが要求されることでしょう。ちなみに、ロード命令の頻度を20%くらいとするとリオーダーバッファ全体ではこの5倍くらい必要ということになるので、200個以上のリオーダーバッファがこの程度の問題でも必要なのです。現在の多くのプロセッサは、せいぜい100個くらいのリオーダーしかできないことを考えると、これは大変なことです。ロードとストアだけリ

オーダーするタイプのプロセッサもありますが、これは良いセンスだと思います。

●分岐予測精度

これはアウトオブオーダーに特に関係する問題ですが、200個も先の命令を実行するためには、その間に多くの条件分岐が入ります。分岐命令の頻度はビジネスアプリケーションでは20%くらいと言われていて、数値計算ではもっと少ないのですが、とりあえずこの数値を用いると、200個のうち40個は分岐命令ということになります。問題は分岐命令が分岐するかどうかはその直前までの演算によって決まることが多く、あらかじめ予測することは難しいことです。分岐予測の精度を90%とすると、40個の分岐命令を実行した後の命令が正しい分岐先である確率は1.5%にすぎません。すなわち、ほとんど無駄な計算をしている確率が98.5%もあるということです。これではせっかくのアウトオブオーダーの機構も意味をなしません。そこで、分岐予測の精度をできる限り向上することが、アウトオブオーダーで高い性能を得るための必須条件になります。

●命令レイテンシ

ある種の数値計算では、関数のテイラー展開やニュートン法による近似値の算出が重要になることがあります。こういった計算では、特に1つの命令の実行結果を次の命令がいつ使えるかという時間(命令レイテンシ)が重要になります。特に算術関数の近似計算などにこの命令レイテンシは大きな効果をもたらします。

Aベンチマークプログラムの説明とベンチマーク結果

さて、それではお待ちかねのベンチマークといきましょう。それぞれのベンチマークは簡単な説明と、その実行結果の性能もしくは実行時間を示し、AthlonとAlphaの比較をしていきます(コンパイラはgcc のほかに、本誌1999年12月号で紹介したCompaq Cコンパイラも使用します)。

●MPIによる並列処理プログラム

これは並列SOR法によりスパースな係数行列の1次方程式の解を求めるプログラムです。C言語とMPIで記述してあり、並列コンピュータ上で動作させるべきものですが、今回はベンチマークのため、単一のプロセッサ上で複数のプロセスを動かして並列処理もどきの動作をさせています。係数行列はN×Nのメッシュに切ったラプラス方程式を5点差分で線形化した差分法の係数行列を対象にし、Nをパラメータとしていろいろなサイズを試しています。

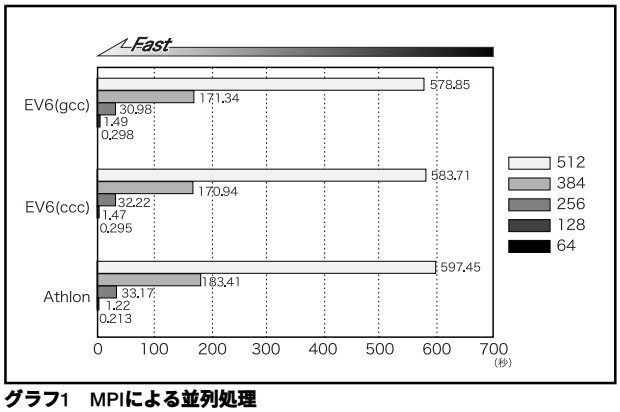
ベンチマークプログラムの「ppSOR」は、次のURLから入手できます。

```
http://shimizu-lab.et.u-tokai.ac.jp/pgm/ppSOR
```

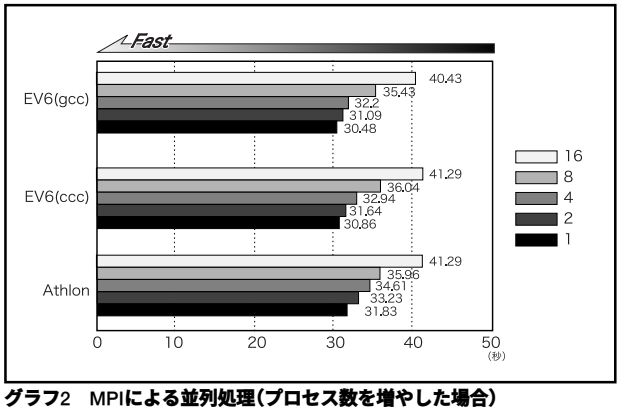
また、コンパイラのオプションは次のとおりです。

```
$ gcc -O6
$ ccc -O6 -tune ev6
```

まずは、メッシュの大きさを変えて試してみます。2つ



グラフ1　MPIによる並列処理



グラフ2　MPIによる並列処理(プロセス数を増やした場合)

のプロセス間でデータの転送を行うようにしています。

結果をグラフ1に示します。これを見るとほとんど変わらない時間で終わっています。若干の違いはあってもそれは誤差の範囲といってもいいほどの小さな違いでしかありません。実はこの結果はちょっと意外で、もっとEV6は頑張れるかと思っていたのですが、プログラムを見ると、演算するよりきっとMPIのライブラリを見る時間のほうが長いのだろうと納得しました。整数性能はAthlonのほうが速いことになっているので、MPIライブラリの実行時間はおそらくAthlonのほうが速いのでしょう。

次に、プロセスの数を増やしてみました。これによってプロセス間通信のオーバヘッドを見るつもりのベンチマークでした。ところが結果はグラフ2のとおりでした。なかなか思うようにデータが取れず、少々ショックです。

●清水DGEMM

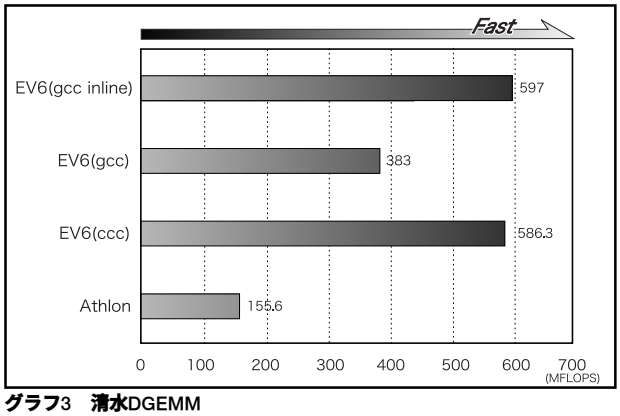
このプログラムは先月号の記事にも出てきたので、あまり説明はいらないと思います。コアの部分でメモリを要求するループを形成しています。しかし、キャッシュに入るようにブロック化しているため、キャッシュの性能は無関係です。DGEMMは、以下のURLからダウンロードできます。

```
http://shimizu-lab.et.u-tokai.ac.jp/pgm/gemm
```

コンパイラのオプションは以下の通りです。

```
$ gcc -O6 -DMAIN -DNOINLINECORE -DNOPREFETCH -
Dev6
$ ccc -O6 -tune ev6 -DMAIN -DNOINLINECORE -
DNOPREFETCH -Dev6
$ gcc -O6 -DMAIN -DAli_max=48 -DBlj_max=24 -
DA_row_block=341
```

結果はグラフ3です。gcc同士の結果を比較すると2倍以上EV6C(21264)の方が早くなっています。キャッシュに収まる演算性能はまだまだ差があります。



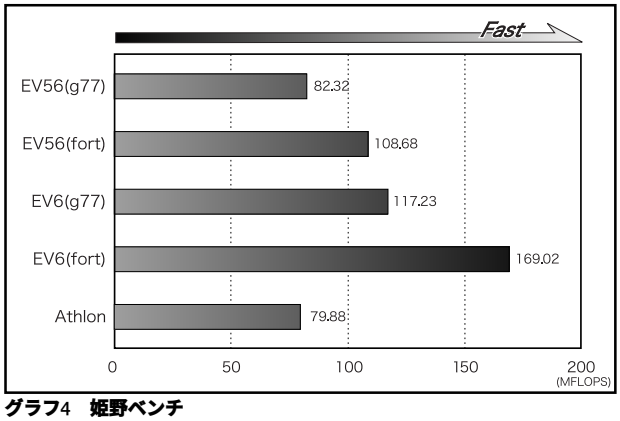
グラフ3　清水DGEMM

●姫野ベンチ

これも以前取り上げましたが、Athlonでの値がどうなるか気になる方もいるかもしれません。

```
$ fort -O6 -tune ev6
$ g77 -O6
```

グラフ4を見ると、AthlonはEV56C(21164A)とほぼ同等の性能をたたき出しています。なかなか優れものですね。でも、EV6C(21264)の値にはまだまだ及びません。このベンチマークは流体系のプログラムで良く使われるもので、演算とメモリ参照のバランスから見て、かなり実プログラムの性質に近いのではないかと思っています。



グラフ4　姫野ベンチ

●LINPACK

これはスーパーコンピュータのベンチマークとして有名なもので、連立方程式をLU分解法で解く時間から、性能をMFLOPS単位で出力します。値が大きなほど高性能です。従来のスーパーコンピュータは小さな行列ほど苦手としていたので、100元と呼ばれる100×100の行列を係数とする問題の性能でベンチマークは評価します。本来はここでも100元で評価したいところですが、あまりにも計測時間が短くなりすぎてまともに性能が出てこなかったので少し大きな問題にしています。

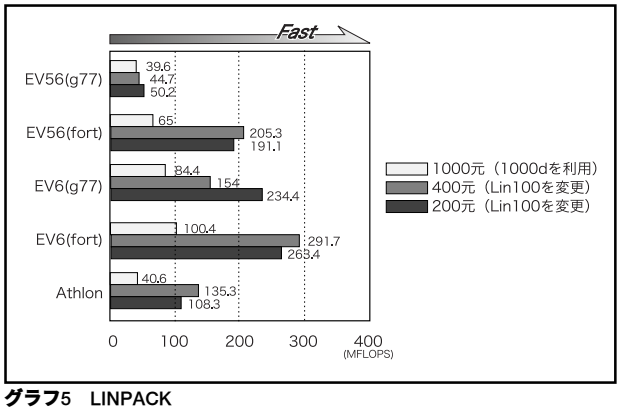
LINPACKベンチマークは以下のURLから入手できます。

```
http://phase.etl.go.jp/netlib/benchmark
```

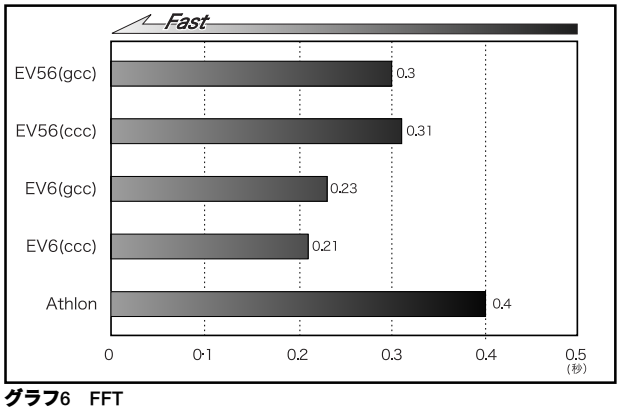
Fortranで書かれたプログラムの時間計測のため以下を追加しました。

```
FUNCTION SECOND()
REAL*4 DUMY(2)
SECOND=DTIME(DUMY)
RETURN
```

結果はグラフ5です。Compaq製のコンパイラではもっとずっと性能が出ていますが、一般的なg77では、1000元のような大きな問題はAthlonとEV56はほぼ同等の性能を出しています。小さな問題ではEV56の性能が低いというより、Athlonが思ったより健闘しています。Athlonでfortをコンパイルした結果はほぼ同じような傾向を示しています。一方、同じAlphaでもg77でコンパイルした結果は早い段階で性能が落ち始めるようです。この部分はもっと詳細に調査して原因を追求すると面白



グラフ5　LINPACK



グラフ6　FFT

いと思いますが、今回は時間がないので結果だけ報告します。Compaqのコンパイラがなければ21164Aの世代のAlphaはAthlonにずいぶん引き離されて負けています。Athlon侮りがたしというところです。21264の世代ではさすがにg77を使っても負けない実力を持っていますが、それでも条件によってはかなり接近した戦いになっています。

●FFT

信号処理だけでなく、πの計算などにも大活躍のFFTですが、これもメモリの参照パターンが独特のものがあるので、ベンチマークとして面白い特性を持っています。FFTのベンチマークは次のURLからダウンロードできるものを用いました。

```
ftp://ftp.nosc.mil/pub/aburto/fft
```

結果はグラフ6です。数値は秒になっているので小さいほど高性能です。この問題ではcccとgccの性能差はほとんどありませんでした。AthlonはFFTの問題は少し苦手のようですが、FFTの次数を変えてもっと評価してみないと1回の結果だけではまだ即断はできません。ですが、SPECのfp性能を見ると、この程度の性能差は妥当なのでしょう。

●小ベンチマーク3題

私のオリジナルの小さなベンチマークを3題実行しました。これはglibcの関数の設計をしたときの評価に用いたものが2つと、大規模なストライドの転送性能を計るものが1つになります。3つとも次のURLから入手できます。

```
http://shimizu-lab.et.u-tokai.ac.jp/pgm/
benchmark
```

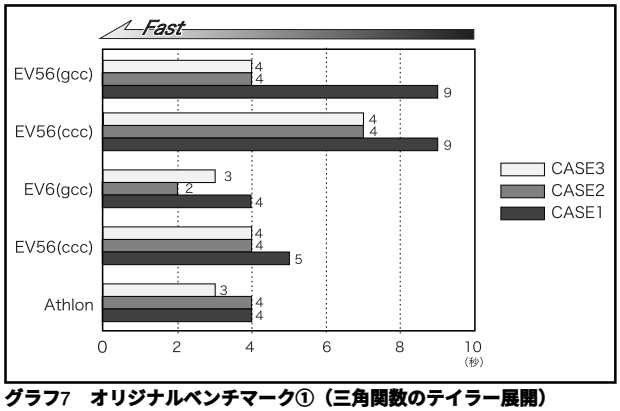
①まずは三角関数のテイラー展開の方法を変えた3つのルーチンによる時間測定です（グラフ7）。Athlonはどれもかなり良い性能を出しています。面白いのはこの問題ではCompaqのコンパイラはgccよりも性能が悪かったことです。テイラー展開は命令のレイテンシが比較的よく影響するのでEV56が低い性能になるのは当然ですが、cccとgccでこれだけ違うのは、cccは良かれと思って何かおかしな展開をしている可能性があります。

②次に平方根の計算の評価の時間測定です（グラフ8）。除算が支配的になるCASE1はEV6の圧勝ですが、全体にAthlonも悪くない性能を出しています。三角関数の計算と同じようにコンパイラの特性がここでも出ています。平方根はニュートン法による近似計算をするのですが、展開式はそれぞれ違ってもニュートン法のループごとに命令のレイテンシがかなり強く影響するので、Compaqのコンパイラがレイテンシ指向になっていない可能性が高いです。

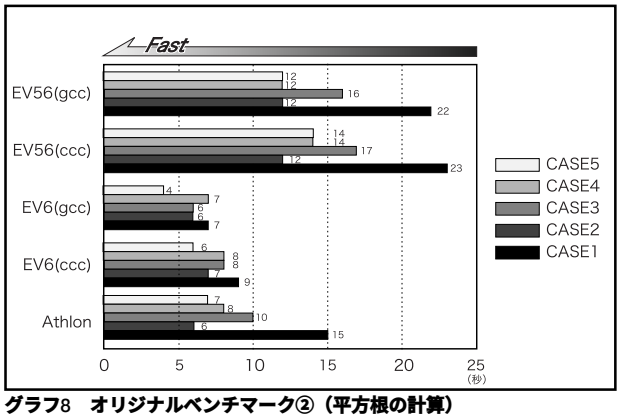
③最後に大規模ストライドのメモリ転送性能です（グラフ9）。大規模ストライドではキャッシュミスの対応だけでなく変換バッファミスの対応も高速性が要求されます。特に変換バッファのミスは、実際のメモリ中のページテーブルにそのページが存在しているかどうかミスが発生した時点では分からないので、アウトオブオーダーを停止するプロセッサがほとんどです。EV6もおそらくそうです。これを投機的にミスの扱いをすることができればかなりの高性能が得られるのですが、バッファミスを投機的に実行するプロセッサは少ないのが現状です。

うろ覚えですが、AMDのプロセッサはバッファミスの時にも次の命令の発行を止めなかったように聞いた覚えがあります。だとすると、AMDはこの問題に対してかなりよい条件を持っていることになります。EV56はアウトオブオーダーそのものを持っていないので、素直に実行しているだけですが、おかしな命令トラップを発生させなくて良い分だけ高速に動作しているのでしょう。

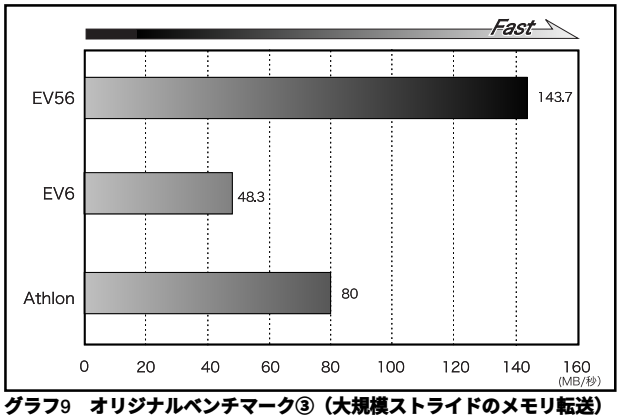
このベンチマークは最後になってしまいましたが、実は大切なものの1つだと私は思っています。大規模な数値計算では変換バッファをミスするのが当たり前になっているし、その状況で高性能を出す必要があるのですから、メモリバンド幅はこの大規模ストライドのバンド幅で考える必要があるのです。その点ではEV6の結果は物足りないといえましょう。



グラフ7　オリジナルベンチマーク①（三角関数のテイラー展開）



グラフ8　オリジナルベンチマーク②（平方根の計算）



グラフ9　オリジナルベンチマーク③（大規模ストライドのメモリ転送）

おわりに

AthlonとAlphaを数値計算の観点からいくつかのベンチマークをしてみました。Athlonはなかなか素性の良いプロセッサだと感じました。今後AMDの言葉通り数値計算を強化していくと、量産プロセッサであるだけにAlphaにとって手強い競合相手に成長していくでしょう。もっともAMDはかなりの損失を毎期計上しているので、どこまで投資余力が続くかも問題ではあります。