Linux Japan Vol.6への掲載原稿です.
3段組で約4ページ程度になると思いますが
体裁を整えるともう少し増えるかもしれません.
宜しくお願いします.

19char/46line
#10203040506070809101112131415161718192021222324252627282930

「Linux/Alphaによる数値計算ならびにRISCアーキテクチャのおはなし(第一回)」

清水尚彦 nshimizu@et.u-tokai.ac.jp

最近Intelとの関係がゴタゴタしたDigital
Equipment Corp.ですが, 特許の係争も一
通りかたがついたようですね. 私を含めて
成行きを心配する人が多かったのですが,
現時点で発表されている内容を見る限り
Alphaの将来は大丈夫なようです. という
訳で, でもないですがAlphaプロセッサで
動作するLinuxの話を連載で掲載させて頂
くことになりました. 連載の全体計画なん
て持っていないのでいろいろと寄り道をし
ながら主に数値計算に使う人をターゲット
として話を進めて行きたいと思います.
(でも, Linux/Alpha自体は日常のツールと
して十分役に立つだけのソフトがありま
ので, 数値計算なんて関係ないという人も
是非御覧ください. 何しろ現時点では
AlphaのワークステーションはSPECint95と
いう整数系のベンチマークでも世界最高速
をマークしているのでTeXを使いたいとか
いった用途でも十分性能がでる可能性があ
ります. それに64ビットのプロセッサなら
ではの高速な整数演算にも使い道がいろい
ろあります.)

どうまとめて行こうかなぁと思っていたと
ころにVol.5を入手したのでさっそく開けて
びっくり, 数値計算特集ではありませんか.
Alphaで数値計算といってもAlphaだけに限っ
た話ではないので一般的な話を交えて書い
ていこうとしていたので, 正直言って少し
ショックでした. 田口さんの連載との内容
の重複も少しはあるかもしれませんが, 御
容赦のほどをお願いします. 連載のスター
トということもあり自己紹介も兼ねて少し
雑談的な経験談とプロセッサの歴史の話か
らはじめさせていただきます. 私は大学
に赴任したのは3年程前の事ですがそれま
では大型コンピュータの設計なんて仕事を
していましたのでちょと硬い話が多くなる
かもしれません. だんだんといろいろな内
容を盛り込んで行くつもりですのでよろし
くお願いします.

1. マイクロプロセッサ/パーソナルコンピュー
タの黎明期のころ

1970年代のはじめにマイクロプロセッサが
出現してからコンピュータの性能向上はす
さまじい勢いで伸びてきました. 日本にい
るとプロセッサを設計する機会はあまりな
いのですが(あたりまえだなんて思わない
でください. アメリカでは大学でプロセッ
サを設計できる事は当り前なのですから.),
それでもこのころの「ラジオの製作」なん
て雑誌にも74シリーズのICを使ったコンピュー
タの製作記事が出たりしていました. その
ころ高校生だった私は「トランジスタ技術

」や「ラジオの製作」を読んでいたので自分で作ろうと思えば作れたはずなのですが，数百のICを接続するプリント基板を作るなんてアマチュア製作家にはとてもできないと感じたものです．そうこうするうちにマイクロプロセッサの値段も安くなってきてコツコツと部品を集める算段をしていましたが，アルバイトをしているわけでもなくお小遣いも小額だったのでなかなか進んでいませんでした．一番のネックは立ち上げの為の手段で，ROMライターなんて買えませんから自分で書き込み器を作れそうなヒューズ型のROMに狙いをつけて32バイトしかないROMでどうやって初期ロードプログラムを作るかなんてことで苦労していた覚えがあります．有名なApple IIやTandy, Commodoreなどの箱ものはとんでもなく高価だったし，TK-80などのボードすら手が出ない値段だったので今の高校生が自由にコンピュータを使っているのを見ると本当に良い時代になったんだなぁって思ってしまいます．:-)

ちょっとだけ宣伝させてもらうと，私は「集積回路設計製作」という演習授業を大学で開講していますがここでは演習参加者の一人一人が独自のアーキテクチャのマイクロプロセッサを設計します．また，設計したプロセッサをFPGAという回路に載せてプログラムを動作させるところまでやっています．半田ゴテもラッピングも何も使わなくてもコンピュータ上で設計した結果の回路を使っていろいろなプログラムを動かせるので既にハードウェアの設計はソフトウェアと同じになっていますね．ちなみにこの授業の一部ではLinuxを使っています．LinuxやFreeBSDなどにはコンピュータを設計する為に必要なソフトウェアがたくさん移植されていますので当然の帰結ではあると思っていますが，ここまでたどり着くまでには長い道のりがありました．Linuxを使ったこの授業に興味がある人が多ければ機会を見て紹介しようと思います．

>>>>>>>>>>> ここから この部分をコラムにしてください．
大学に入ってから私はCommodore社のVIC-20というマシンを入手し，この機械とはずいぶん長くつき合っていました．このころのパーソナルコンピュータは前出の74シリーズと呼ばれる小規模なICをたくさん使っていたのですが，VIC-20には74シリーズは12個しか使っておらずシンプルな基板と「ソフトでできることはハードでやらない」といった割り切りがはっきりしている優れた設計がされていました．この機械のOSを解析することで私はリアルタイム処理と割り込みについてずいぶん勉強させてもらいました．タイマーを活用したリアルタイム処理の考え方は就職した後に設計した大型コンピュータの仮想機械のタイマー処理の考え方を理解するのにも役に立っています．昔の機械は小さかったから解析できたのであって今はそんなこと出来ないなんて思ってませんか，そこのあなた．時代は変わったけど今や優れたOSのソースコードが無償で入手出来るのです．ソースコードはコメントもついているし変数名一つとってもわかりやすいので昔の小さな機械のOSを解析

するよりもずっと簡単だと私は信じています．ですからゲーセンに行く時間があったら是非その半分の時間でも OS のソースコードを眺めるのに使って欲しいと思います．

さてそのころ世の中はマイクロプロセッサといえども OS を搭載することを前提に考えはじめています．例えば CP/M なんて OS がずいぶんとはやったものです．例によって私は貧乏学生だったのでとても OS だけに高いお金を出せないので CP/M は話だけで指をくわえて見ていました．16 ビットのプロセッサが出はじめたころには CP/M と互換性のあるシアトル DOS とかいったものの宣伝を「Bit」という雑誌に見つけたりしていろいろあるもんだなぁと感心していたらいつの間にかゲイツさんがこの OS を買っていたようです．:-) 私には手が出なかった CP/M の商用ライセンスは$5 らしいと知人から聞いたりすると市販価格とのギャップの大きさに憤慨したものですが，この商用ライセンスを取った国産ワープロが出てくると聞いてずいぶん期待をしたものです．(結局このワープロは NEC から文豪 mini5 として出荷され大部後になって私ははじめての OS 付きコンピュータとして mini5H というマシンを入手しました)そのうち世の中は PC-DOS と IBM PC/AT 互換機一色になって行くのは皆さんご存知の通りですね．
<<<<<<<<<<< ここまで この部分をコラムにしてください．

1980 年ころアメリカでは DEC の VAX11/780 というミニコンピュータが大ヒットになっていました．これは 1MIPS マシンと呼ばれていましたが，実際には IBM の 1MIPS マシンと同等の性能という意味の相対 MIPS を使っていたようです．この機械はカルフォルニア大学のバークレイ校で unix を移植してさらに仮想記憶等の拡張をした独自バージョンを配布しはじめた事によって大学の unix の標準機ともいえる位置付けになっていました．日本でも東大の計算機センターにはこのマシンが入っていました．私はこの下位機種で VAX11/730 というマシンを使っていました．今から見ると 0.2MIPS で 1MB のメモリのこのマシンはオモチャのようなものですが，これでもちゃんと unix を動かせるし Emacs や Franz Lisp などのプログラムできちんと仕事にも使えるすぐれものでした．

DEC はこのマシンでは 16 ビットの PDP11 という機械との互換モードを用意して従来マシンとの互換性に工面していました．現在多くの会社が 32 ビットと 64 ビットの両方に互換性のある unix を作ろうとしていますが，DEC はずっと前からこのような経験があるというのも不思議な巡り合わせです．ちなみに現在の Alpha の unix には 64 ビットモードしかありません．最初から 64 ビットのマシンとして設計されたためですが，各社の unix が 32 と 64 の間で苦労するだろうと思われるのでこの面でも一歩リードしている可能性はあります．ビジネスと技術は別のものなのでだからといって DEC の業績に貢献するわけではないのですが．

2．Alpha 出現前後の話

私が最初にAlphaチップの事を聞いたのは
ISSCC92という学会の発表題目でした．私
は学会に行ったわけではなく「日経エレク
トロニクス」っていう雑誌に出ていた速報
記事でこれはエライものが出たものだと思
いました．その当時クロック周波数100MHz
というのは大変困難な目標だったのですが，
あっさり200MHzという超高クロックを発表
したAlphaプロセッサには大変驚かされた
覚えがあります．（そのころ私がいた会社
ではRISCプロセッサを作るっていうプロジェ
クトもあったのですが，その目標クロック
はもっとずっと低いものでした．） 技術の
詳細はその後論文を入手して知ったのです
が，命令セットやメモリモデルにクロック
周波数を上げるための工夫がいろいろとな
されているだけではなく，当時はやってい
た Delayed Branchもスーパースケーラの
時代には無用の長物であるとしてあっさり
捨て去り小気味よい割りきり方をしている
と感じたものです．

で，私が一番驚いたのが回路技術の面でし
た．クロックを上げるにはクロック分配の
誤差であるクロックスキューを低減する必
要があるのですが，レジスタとレジスタで
正しくデータを送るために多くの会社では
重ならない2つのクロック信号を使ってい
ました．重ならないようにするにはいろい
ろと工夫が必要で安全を取ると当時の技術
(1μ m)では200MHzなんてとてもできない
と皆思っていた訳です．ところが，DECの
技術者はこの問題に真向から取り組みなん
と一つのクロックで回路全体を動作させる
ことに成功しました．これにはクロックの
1の信号で動作するレジスタと0の信号で動
作するレジスタを組み合わせてクロックは
一つでも実質的に2つのクロックと同等の
動作をさせるという逆転の発想がありまし
た．

驚いたのはクロック周波数だけでなく発熱
も凄くなんと30Wも出るということでびっ
くりしました．現在ではCPUに直接風を送り
冷却するのが普通になって 30Wくらいどうっ
てことないのですが，1992年当時にはセラ
ミックのパッケージの発熱が10Wを超える
なんて信じられないという感じがありまし
た．（ちなみに初期の66MHzのPentiumはた
しか13W位で装置メーカーからクレームの
嵐となったはずです．クロック周波数と発
熱は比例関係にありますから，PentiumよＵ
りAlphaはクロック当たりの仕事が少ない
ということになります．）もっともその直
前にDECは150Wでもヒートパイプを使えば
冷えるよっていった内容の技術発表をして
いたので，高速化の計画は着々と進んでい
たようです．

私は1991年のある学会でDECの技術者がBIPS
プロセッサの話をしているのを聞いたので
すが，(BIPSっていうのは1000MIPSの事で
す．あの当時これは到底到達できない雲の
上のような目標に見えたものです．が，今
ではAlphaは2500MIPSですから時代は変わっ
たものです.)このヒートパイプの技術は
BIPSの為に開発されていたのだと思います．
ところで，BIPSというプロジェクトはなん

とバイポーラトランジスタでの設計を目指していました．そのころ多くのプロセッサはすでにCMOSの将来性を歓迎してそろってCMOS化の道を走っていたのでこのプロジェクトは格別の感がありました．ちなみに最近では大型コンピュータやスーパーコンピュータすらCMOSで作られています．その後Alphaの出現とともにBIPSの話はどこかへ消えてしまったのですが，あのまま設計を続けても面白いプロジェクトだったと思います．

3．インテルペンティアムの衝撃

さて，1992年のISSCCの発表の年の8月に例年スタンフォード大学で開かれているホットチップシンポジウムの聴講に出かけました．Alphaの話はその時までに論文で知っていた以上の話はあまりなかったのですが，その年はIntelがP5というプロセッサ(ペンティアムのことです)をシンポジウムで発表するというのでみんなで大変期待していました．で，シンポジウムの予稿集を見るとIntelの発表の原稿は出ていないのです．あれ?と思っていると当日発表会場で配るといってました．なかなか慎重でしたが，発表はそれに輪をかけて慎重でした．それでも発表内容だけを見てもP5が相当高性能なプロセッサになる事は十分予想でき大変期待できる内容でした．特にすごいと思ったのはスーパースケーラの二つのパイプラインで同時にメモリ読み出しができる点でした．当時はマルチバンクのキャッシュのようなコストのかかる方法はあまり採用する会社はなくこの面ではIntelの一人勝ちってところでした．(今思えばあの当時Intelの株でも買っておけば今ごろは大金持ちですね.:-) 発表自体は全貌とはほど遠くほとんど情報を出さなかったのでみんながっかりしました．当日質問した人も「それはよい質問だが，まだ答えられない」なんて返答しかもらえませんでした．また，当日配った資料の表紙もこれを反映していてLSIの絵の一部にスポットライトが当たっていてそこに発表内容の「Super Scalar Integer Unit 」とか「Floating Point Unit」とか書いてある以外は暗くなっています．商業主義と学会のオープンな雰囲気とのギャップがあるのですが，よく考えればしばらく待てばきちんと発表できるのでこの半端な発表はIntelの学会へのサービスだったのかもしれません．このおかげでBYTEやらNEやらいろいろな会社の記者が世界中から集まって学会の宣伝になったのですから :-)

とはいっても，Intelがいくらお金持ちで0.8μmのBiCMOSなんて技術を投入してもクロック周波数は66MHzとAlphaの敵ではなかったのです．やはりこのあたりはアーキテクチャ設計の時代背景が色濃く出て来ます．後から来た方が有利な選択ができるのでAlphaやPOWERなどのプロセッサはいろいろな工夫をしています．というとIntelのIA64はすごく良さそうに聞こえますが，これはものが出ていないので論評を避けさせていただきます．コンパイラ担当の人達の話を聞く限りは数値計算系のプログラムではかなり命令並列度を上げられるような見

通しを出していましたが，並列度が上がっ
てもクロックが上がらなくては性能はでな
いのでIntelの御手並拝見といった所です．
個人的にはマルチポートの巨大なレジスタ
が必要なIA64はAlphaの21164などに比べる
とクロックアップが難しい感じがしますが
何しろ出て来るのが大部先の話なのでその
ころまでに技術的な問題点を解決するつも
りなのでしょう．

4．Alphaのアーキテクチャ
さて，おまちかねのAlphaの話です．この
アーキテクチャは次の点に注意して設計を
行ったとなっています．

・性能がすぐれていること　複数命令の実
行を妨げるような命令構成を避ける

・長寿であること　複数のOSのサポート，
64ビットのデータ長とリニアな64ビットア
ドレス空間の実現

・スケーラブルであること　マルチプロセッ
サのサポート

・汎用的であること　多くの言語をサポー
トすること．ソフトウェアのバイナリ変換

特に二番目の長寿であることはIBMのS/360
と比べてDECのアーキテクチャが短命であっ
たことからもソフトウェア資産を集積する
上でも重要な項目だと思います．64ビット
のアドレス空間であれば今後かなりの期間
に渡ってアーキテクチャの変更を検討する
必要は少なくなりますね．でも，VAXの32
ビットアーキテクチャが短命だった理由は
(まだ使われているので短命とは言いにく
いですが)性能が向上しなかったことによ
るものだと思います．そこで，一番の性能
向上を妨げないアーキテクチャというのも
DECの経験に照らしても非常に重要だった
ことは容易に想像できます．

最近FX!32という名前のバイナリ変換がNT
で使えるという話を聞いた事がある人もい
ると思います．Linux/Alphaでもem32とい
ったエミュレータが出ていますが，FX!32
はバイナリ変換ができるものです．実はNT
以前にDECはバイナリ変換の必要性に迫ら
れていました．上のリストのバイナリ変換
は先程出ていたVAXとDECが以前販売してい
たMIPSアーキテクチャのワークステーショ
ンのソフトウェアの移行措置として必要に
なったのです．そこで培った技術がおそら
くFX!32やFXPといったソフトに使われてい
るのだろうと思います．これらの技術はプ
ログラムの再コンパイルをせずにアプリケー
ションを新しいワークステーションに移行
できるので特にエンドユーザーには必要性
が高いものです．

さて，これらの目標を達成する為にとった
アーキテクチャ上の工夫には次のようなも
のがあります．

・条件コード，副作用のある命令，モード
ビット，バイト単位の書き込みを止める．

このうちのバイト単位の書き込みはデバイスドライバを書く都合上EV56の世代になって追加されましたが，それ以外の工夫はクロック向上のために役に立っていると思われます．条件コードは以前のプロセッサでは特別なフラグに格納される事が多かったのですが，この数によって命令のスケジューールに制限が出て十分な並列性を引き出せないことがありました．そこで，条件コードをフラグとして持つ替わりにレジスタに条件判定の結果を持たせればレジスタの数だけ条件を独立に持てるので制限は緩くなるといったものです．また，命令が副作用を持つと命令デコードに余分な負担を強いることになります．モードによってプロセッサの動作を変えることもハードウェアに余分な判定用の回路が必要となってあまり良い効果を生まないと考えたようで，最初からアドレスモードにしても64ビットしか用意されていません．そのわりに浮動小数点の演算はVAX互換とIEEE互換の二つの形式を使えるようにしているのですから，この点は不徹底だとは思います．(これはモードで切り替えるのではなく命令によって変わるので上の項目には反しないのですがそのサポートの為のハードウェアは少なくはないと思います.) 浮動小数点の形式の問題はなかなか奥が深く一筋縄ではいかない部分があるのでDECの選択は間違いとはいえないのですが，(実際CRAYだって困っていた問題なのですから.)せめてVAXの浮動小数点形式の命令を特権命令にしてエミュレーション専用としていれば後々便利だっただろうと思います．あ，これは特権「モード」を必要とするからダメなのか．:-)

・条件転送命令.

これは分岐命令を使わずにすむケースを増やし分岐オーバヘッドを低減するものです．同様なものとしてHPのPAアーキテクチャにはナリファイという命令の取消機能がありますが，命令数を絞って転送命令に限定したのでハードウェアの負担は減っていると思われます.

・大きな相対分岐の指定範囲(+/- 4Mbytes)

これが直接性能に関係するか疑問ですが，分岐の数を少し減らす効果はあります.

・効果が実装依存となる工夫の排除

遅延分岐やロードのスロットなどのたまたまある世代でのみ役に立つようなアイデアは採用しないということです．一時期遅延分岐は分岐ペナルティを低減する方法としてもてはやされましたが，同時発行する命令が多くなってくると遅延分岐による性能向上効果はほとんどなくなって替わりに例外的な命令発行方法を採ることによるデメリットが目立つようになりました。もちろん世代ごとに遅延スロットを変えるような対策をとればいいのですが，そうすると今度は世代ごとにバイナリの互換性が無くなりプロセッサのビジネスとして成り立たないことになります．同じことはMIPSが導入

したロードスロットにも言えますが，Alpha
ではこれらの実装依存のトリッキーな仕組
みは採用しないこととしています．

同時代のアーキテクチャであるPOWERと比
較すると設計者の思想の違いが感じられて
面白いものがあります．POWERでは複数の
独立したユニット間の並列性をいかに引き
出すかに重点が置かれていて，分岐を担当
するユニットに条件コード，リンクレジス
タ，カウントレジスタなどを配置してます．
これらがすべてソフトウェアのコンテキス
トになるのでタスクスイッチのときには少
々リソースが多いことになりますが，なる
べく多くの命令を同時に実行し，かつ分岐
が必要なケースではなるべく早く分岐の為
の情報を分岐ユニットに集めるようにして
命令の読み出しを先行してできるような工
夫がなされています．条件コードが命令の
スケジュールの邪魔をしないように条件コー
ドの数を増やして命令中で指定できるよう
にしたり，A*B+Cの形の内積演算をサポート
したり詳しく見るとこれだけでも面白い技
術がたくさん詰まっています．特に内積演
算はPOWERと他社のワークステーションの
関数演算性能に大きな差をつける原因とな
って，その後HP,MIPSなども追従していま
す．Alphaでは内積演算はサポートしてい
ません．そのため一部の関数ではHPやMIPS
のプロセッサに負けることがあります．こ
れは設計の考え方が違うことによりますが
クロック周波数が大きく勝っていれば全体
として性能が上がることになるでしょう．
同様の議論は演算のクロック数にも言えま
す．HPはAlphaの演算クロック数はHPに比
べて多いので性能が出ないといっていたの
ですが，クロック数だけでなくクロック周
波数も合わせて考えないと正確な答えは求
まりません．ほとんどのRISCプロセッサは
加算，減算，乗算などの演算は毎クロック
発行できるのですが，計算結果を次に使え
るまでのクロック数を演算レイテンシと呼
んでこれが短い程良いプロセッサと考えら
れています．ところが，500MHzでレイテン
シが4サイクルのAlphaと200MHzで3サイク
ルのレイテンシを持つPA8000とどちらが早
いと思いますか?

Alphaアーキテクチャのプロセッサはすで
にいくつも作られて販売されています．
個々のAlphaチップの工夫に付いても色々
書きたいことがありますが，それはまた別
のお話ということで．:-)

5．数値計算はじめの一歩

さて，数値計算と書かれた記事を読む人々
は大きく二つのグループに分けられるので
はないでしょうか．はじめのグループは数
値計算は本来の仕事のためにしかたなくや
らなければならないけど，できたら避けて
通りたい人達，つぎのグループは数値計算
を楽しんで自分で深く研究したい人達です．
どちらの人も大きなプログラムをいきなり
作って正しく性能も良いものを作るのはそ
んなに簡単ではありません．

そのなかで，簡単に数値計算をはじめるた

めに素晴らしいソフトがあります．これは Vol．5でも紹介されています Octave というソフトです．私は日常的に簡単な計算にはこのソフトを活用しています．多くの人々にもっと活用して頂きたいと思っていますので，簡単な使い方を連載の中で取り上げていきます．もちろんLinux/Alpha でもこの素晴らしいソフトは動作します．

どのくらい簡単かというと

```
a = [1 2 3; 4 5 6; 7 8 0]
b = [1 0 0]
x=a\b
```

と打つと

```
x =

  -1.77778
   1.55556
  -0.11111
```

なんて答えが得られます．これは

```
a*x = b
```

という連立一次方程式の解になっています．また，FFT を使う場合も簡単です．

```
I=[1;0;1;0]
fft(I)
```

と打つと

```
ans =

  2
  0
  2
  0
```

と答えが得られます．

このように対話的に使うだけでなくプログラム言語としての側面も持っていまして，数値計算の色々なアルゴリズムを確認するときにこのプログラム言語である特徴がたいへん役に立ちます．
Help に出ているつぎの例を見てみましょう．

```
   fib = ones (1, 10);
   for i = 3:10
     fib (i) = fib (i-1) + fib (i-2);
   endfor
```

これはフィボナッチ数の計算プログラムです．まず，ones という関数で変数 fib を初期化したのち，アルゴリズムに従って先頭から数値を求めて行きます．
Octave で使える制御コマンドは for の他に while，if などがあり，もちろん関数もあります．この例で示すように BASIC 程の記述が可能な上，前に書いたように行列を直接変数として扱えますので本に出ているアルゴリズムをそのままプログラムにすることができて，簡単なプロトタイプの用途には十分な機能があります．Octave はインタ

プリタですが，行列計算の部分はパッケージを使っているので行列計算主体の場合にはCで書いた場合とそれほど遜色ない性能を発揮できると思いますので，もっともっと使われてもいいと思います．というわけで，連載ではOctaveの使い方も少しずつ紹介していこうと思っています．

6．おわりに
Linux/Alpha固有の話は今回はありませんでした．ですが，この話を読んでAlphaに関心を持ってくれる読者が一人でも出てくれれば望外のよろこびです．Linux/Alphaに関しての情報はドキュメントの更新を開発者自ら行っている関係でどうしても古い情報になりがちです．そこで連載ではなるべく最新の性能に関する話題について触れていきます．

今，日本のLinux/Alphaのメイリングリストでは数値計算の高速化を含めて議論が活発に行われています．例えば，1000x1000の行列の行列行列積で500MFLOPS以上の性能を発揮するdgemmサブルーチンの開発や各種関数の高性能化など数値計算に興味がある人ならぜひ参加していただきたいと思います．これらの開発の結果はStataboware というLinux/Alphaのパッケージにぞくぞくと採用されていきますので，誰でも簡単に高性能数値計算ができるようになっていくと思います．このパッケージの開発とメンテを行っている後藤さんはたいへんなパワーで開発を進められており非常に感謝しております．今後の連載の記事の中でもこのパッケージを題材として色々な話題をとりあげていきます．Redhat Linuxを使う人も考え方は参考にできます．でも，特にFORTRANを使う必要がある方にはRedhatはあまりお勧めしていません．この件に関してはいろいろな議論が出ていますので，公開されているメイリングリストをお読みください．

　なお，メイリングリストとStataboware の情報はつぎのURLを参照ください．

http://www.kuamp.kyoto-u.ac.jp/linux-alpha-jp

-------- ちょっとだけStataboware-1.0のはなし --------
記事中に記載されたURLからStataboware のFTP サイトにたどり着けます．現在の安定バージョンは1.0になっています．以下特徴とパッケージに添付されるソフトの概要をREADMEより参考までに転記します．

********** ここから **********
2．パッケージの特徴

i)　安定である ： 私は EB164(300MHz, メモリ 128MB)で作業を行っていますが、このパッケージを使用していて落ちたことはありません(除、カーネルのテスト)。また、MIATA(500MHz, SD-RAM 128MB) での動作確認も行いましたが、極めて安定して動作しました。

ii) 数値計算での使用を考慮に入れてある ： f2c に加え、 g77 を入れてあります。私が試した限りでは g77 のパフォーマンスは f2c + gcc よりも上です。現時点では g77 に大きなバグはないようです。

iii) 必要最低限の日本語環境が添付されている ： mule-2.3(19.28 based)やASCII-pTeX、xdvi、tgif、xfig、canna 等の基本的な日本語環境を提供。

kon や XFree86-3.3.1 を併用して数値計算をさせながら、メールや TeX
による文書作成ができるようになっています。

3. パッケージに含まれている主なソフト

基本環境
bash, tcsh, zsh, binutils, sh-utils, mtools, gzip, sysvinit,
getty-ps, proc-progs, kon, 他

開発環境
gcc-2.7.2.2, binutils-2.8.1, f2c, g77-0.5.20, make,
libg++-2.7.2(パッチ提供:越野氏), flex, yacc, dejagnu,
ncurses, bison 他

ネットワーク関連
nfs, ping, ifconfig 等の基本的なネットワーク関連, yp client,
yp-server

X11R6 関連
XF86_S3, XF86_Mach64, XF86_TGA, XF86_SVGA、その他
X11R6(XFree86-3.3.1)
一式, kinput2, kterm, そのほか諸々の X プログラム。

日本語関連
Mule 各種(Canna, Canna + X-toolkit, Wnn, Wnn + X-toolkit),
ASCII-pTeX(SJIS, EUC), META Font, Xdvi, mh-6.8.3JP,
Canna, Wnn(クライアントのみ、サーバは ECOFF 版を添付。おまけです),
ghostscript-5.01j, jperl-4.036, perl-5.003, jman, jgroff-0.99

********** ここまで **********