Linux/Alpha活用講座

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

第11回
Compaqの高性能Fortranコンパイラ

知人が電子メール用にシャープの「アイゲッティ」というPDAを入手したと聞いて改めて調べてみると、機能と価格のバランスが良く、メール中毒症の私も納得して1台入手しました。メールを考えなければPDAとしてはHP 200LXで十分満足していますが、メールにはちょっと不便で別のものを探していたところだったので、ちょうど良いタイミングでした。でも、PDAとしては使い勝手はいまいちで結局、両方持ち歩くことになっています(慣れの問題かもしれませんが)。

アイゲッティで使うために手持ちのPHSの機種変更をしてメール転送専用に完全従量制のプロバイダを1つ契約しました。携帯メール端末の完成です。ちょっと惜しいのはメールをフォルダに分類したり送受信簿のバックアップができないことで、「これさえできれば文句無いのだけど」と思っています(標準搭載の他のソフトなんていらない)。こういったときに、アイゲッティから電子メールの内容を含む全データを送出し、それをIrDAで受けてLinuxにバックアップするということが簡単にできるといいのですけどね。誰か作りませんか？　もしくはアドオンソフトのような形でメールフォルダサポートのザウルスユーティリティ(MOREとかいうらしい)を作ったほうが便利なのかもしれません。

Alphaの話題

巷の噂では、来年IBMが銅配線のプロセスでAlphaの製造に参入するかもしれないとのことです。どうなることでしょうか……。あまり製造プロセスのことをご存知ない方も多いと思うので少し解説しますと、従来集積回路の配線はほとんどがアルミで行われてきました。これは、アルミは蒸着の特性がよく集積回路に合っていて、加工性にも優れ作りやすかったからです。でも、プリント基板やケーブルの中身はほとんどが銅になっています。これは銅の方が電気抵抗が少なく損失が減るためです。データを記憶する基本素子はラッチと呼ばれます。

プロセッサの周波数が上がってくると、ラッチからラッチまでの遅延時間を低く抑えないといけないので、設計は難しくなります。例えば、Compaqの設計陣が得意な単一クロックの二相設計で1GHzのプロセッサを作るときには、クロックの分配の揺らぎ(スキューといいます)を無視しても、データはクロック周期の半分である0.5nsで次のラッチに到着する必要があります。現実的にはスキューが0という集積回路は作れないので、実際に使える時間はもっと減ってしまいます。例えば、スキューが0.3nsであれば転送に使える時間は0.5−0.3＝0.2nsしかありません。

Alphaを始めとするCMOSプロセッサでは、転送の遅延のほとんどは浮遊容量の充電時間によって決まります。さらに、転送のルートを簡単な集中定数の回路に近似すると、この回路の時定数「$\tau = RC$」が遅延時間にほぼ比例します。そこで、この時定数を減らさないと速いプロセッサは作れないのです。Cはキャパシティですから、とにかく面積を小さくすると小さくなりますが、そ

れも限界があります。また、誘電率を小さくしてもやっぱり小さくなりますが、誘電率の変更は材料の選定が根本から変わってしまうのでプロセスの立ち上げが大変になります。

一方、Rは抵抗のことですが、これも材料によって抵抗率が決まっており、配線の太さを太くすることで下げられる性質のものです。太くすると配線の数を増やせないので、トランジスタの数が膨大になってきた最近のプロセッサでは材料を変えることが望まれていました。このところ、電気的に性質の良い銅を各社が採用し始めたのはこういった事情があるからです。来年、21264のクロックは1.4GHzになると予想されていますが、こういった高いクロックをサポートするために銅の技術が使われるというのは十分考えられます。

API(Alpha Processor Inc.)から、750MHzの21264プロセッサが発表されています。執筆時点ではこのプロセッサによって、僅差ですがSPEC Int_base95とSPEC Fp_base95も再び世界最高速に戻りました。SPECの中でbaseと呼ばれる数値は個々のプログラム毎のオプションの変更を許さないものです。ですから、一般的なプログラムの性質を見るときには、この「base」の数値によるほうがいいのです。もっともコンパイラが自動的に専用のコンパイル結果を出すようなこともできるのでbaseだからといって安心できる数値ではないのも事実です。

APIのホームページに出ているSPEC発表の性能数値を表1、表2に示します(値は一般公開されているものです)。

APIからの新しい21264のシリーズはスロットBと呼ばれるPentium Ⅱのようなカード型になっています。この上に2次キャッシュを載せているのもPentium Ⅱなどと同じですね。これによってマザーボードとプロセッサの独立性を高めて、それぞれ独自に開発を進められるようになります。

APIのホームページにはAlphaに関するいろいろな情報も載っています。http://www.alpha-processor.com/を参照してください。さて、この中で注目すべきものが「UP1000」というマザーボードです。このボードはCompaqが出している他のマザーボードやAPIの出している上位ボード「UP2000」に性能的にはかなわないのですが、大きな特徴があります。

というのは、マザーボード上のチップセットがCompaq製ではなくAMDのものを使っているということです。AMDのAthlonの発表時にCompaq EV6バスを使うといっていたことを覚えている方もいると思いますが、AMD用のチップセットを逆に21264に流用したというのがこのUP1000になるようです。どうしてこれが大きな特徴かというと、システムの価格がドラスティックに低下する可能性があるからです。UP1000のに搭載されるNorth Bridgeと呼ばれるLSIは、AMDの設計した「AMD751」1個だけです。これがCompaqのチップセットになると、低価格を狙ったUP2000でも7個のLSIが必要になっています。AMD製のNorth Bridgeになったことで失うものはメモリ性能、マルチプロセッサ、64bit PCIバスです。マルチプロセッサは必要ないという人でも関係してくるのはメモリ性能の違いです。AMD751では、メモリバス性能こそ100MHzをサポートしていますが、メモリの幅は64bitとあまり広くとっていません。一方、Compaq製のUP2000のメモリの幅は256bitと非常に大きく取っています。メモリバスのスピードは83MHzですが、転送性能は3.3倍にもなりますのでメモリ性能に依存するようなアプリケーションにおいてはかなりの性能差が出てくることでしょう。それだと興味ないなんて言わないでください。つまり、UP1000ではプロセッサ以外はPCと同じわけですから、PCの値段にAlphaプロセッサの差額を足せば購入できるような価格レンジで出荷される可能性が高いということなのです。すでに「DS10」という低価格指向のサーバが出ていますが、オプションは多いほどよいのです。

表1　Top Results for selected manufacturers:SPECint_base95

Company/System	Result
API UP2000	31.8
HP 9000 Model N4000	30.8
Compaq AlphaServer DS20	23.6
Intel MS440GX (Pentium III Xeon 550 MHz)	23.6
Sun Ultra 60 Model 1450	16.2

表2　Top Results for selected manufacturers:SPECfp_base95

Company/System	Result
API UP2000	49.0
HP 9000 Model N4000	48.7
Compaq AlphaServer DS20	48.4
IBM RS/6000 43P-260	27.6
Sun Enterprise 3500	26.5
SGI Origin200 270 MHz R12000	23.7

CompaqのFortranコンパイラ

さて、先月号のAfter Hoursでお伝えした通り、CompaqからFortranのβリリースがありました。雑誌の出版時にまだβリリースが継続しているなら、http://www.digital.com/fortran/linux/からたどって、必要な事項を記入して送信すると、ダウンロードの手順がメールで送られて来ます。このFortranは、Fortran 95のフルスペックのものなので、Fortran 90以降の機能が必要な方は是非入手すべきでしょう。後で述べるように、まだβ版とはいっても性能はGNUのFortranよりもずっと優れていました。

当初はRPM形式だけのリリースでしたが、Debianのプロジェクトメンバーからの要請でdeb形式のものも置かれるようになっています。初期のdeb形式はシェアードライブラリの自動作成がうまくいかず、インストールが終った後に手動で作成する必要がありました。執筆時点では修正版が公開されています。修正版と前のバージョンはリリースバージョンの番号が同じで区別がつかないので、CPMLのシェアードライブラリがインストールされずにFORTRANのリンクができないという人は注意して再度CPMLだけ入れ直してみてください。

ただし、パッケージをまとめた人のシステムがunstableのリリースを使っている関係で、stableの人はそのままではインストールできません。ちょっと気持悪いですが、dpkgに--force-dependsオプションをつけて強制インストールします。

これだと依存性のチェックをしてくれないのでインストール順序に気をつけてインストールする必要があります。私は次の順番でインストールしました。

①libots	コンパイル時ライブラリ
②cpml	数値計算ライブラリ
③cfal	Fortranコンパイラ
④cfalrtl	ランタイムライブラリ

私自身、RPMからalienコマンドでdebianフォーマットへの変換をしてみましたが、RPMではアーカイブ中にはシェアードライブラリはなく、インストールのランタイムで作成するのですが、alienコマンドはDebianのpostinstを自動作成しないので、そのままではシェアードライブラリができません(CPMLのみの問題)。alienコマンドで変換したときの問題点は2つありました。1つはcfal-1.0-2.alpha.rpmの中のFortranの実行ファイルがスタティックリンクされているのですが、そうするとalienコマンドのダイナミックリンクのチェックが失敗します。Debianのメーリングリストで教えてもらいましたが、「alien -g」で展開だけしてdebian/rulesのファイルを変更することで対応できます。もう1つは先ほど書いたシェアードライブラリのインストール時生成のサポートです。これはpostinstを自分で書けば大丈夫だと思います。

実行例1がFortranのベンチマークの結果です。テストプログラムは構造/流体系などのシステムの性能の比較に使われることが多い、理研の姫野さんが作成された「ヒメノベンチ」を使っています。ベンチマークのマシンは今となっては旧式ですがEB164で、21164-300MHzのシステムです。

PAUSEコマンドの様子や、writeコマンドの出力結果が微妙に違うのは無視して、性能だけ比較してみてください。単一のプログラムの結果だけでは何とも言えませんが、同じプロセッサを使っても30%以上の性能向上がなされています。

このベンチマークを私のところから使えるマシンでも

実行例1

```
------------------- COMPAQ FORTRAN -----------------
$> fort -O5 hime98s.f -o hime98s.cpq
$> ./hime98s.cpq
  mimax=     129  mjmax=     65  mkmax=     65
  imax=     128  jmax=     64  kmax=     64
 cpu :    3.406250     sec.
  Loop executed for           13  times
  Gosa :  3.0001516E-03
  MFLOPS measured :   62.84919
  Score based on MMX Pentium 200MHz :   1.947604
FORTRAN PAUSE
PAUSE prompt>

------------------- G77 FORTRAN --------------------
$> g77 -O2 hime98s.f -o hime98s
$> ./hime98s
  mimax= 129 mjmax= 65 mkmax= 65
  imax= 128 jmax= 64 kmax= 64
 cpu :   4.67089891sec.
  Loop executed for  13 times
  Gosa :  0.0030001516
  MFLOPS measured :  45.8327293
  Score based on MMX Pentium 200MHz :  1.42028904
PAUSE  statement executed
To resume execution, type go.  Other input will terminate the job.
```

いくつか(Alpha以外のシステムを含む)実行してみましたので、参考までにテスト結果を示します(実行例2)。

このプログラムは三次元の領域のポアソン方程式を差分化し、ヤコビ法による反復計算によって各グリッドのポテンシャルを求めています。ヤコビ法というのは緩和法という反復演算による一次方程式の解法の1つです。この方法では各グリッドの近似ポテンシャルのうち、直前の近似計算ループの中で算出されたポテンシャルだけを新しいポテンシャルを計算するために用います。そのために、計算した結果をいったん「wrk2」という配列に格納し、後でポテンシャルの配列である「p」にコピーし直しています。

ヤコビ法を用いることで反復回数はSORなどに比較して多くなります。SORは次の反復で計算される方向を予測するので、比較的良好に予測できる場合には反復計算をしなくても良い近似が得られるからです。また、1回に隣接点のポテンシャルしか利用しないヤコビ法では、ある点から領域内の一番遠い点までその影響が及ぶのは、最低その間の距離分の反復をしてからということになるので、この面でも反復回数の多さが直観的に理解できると思います。

また、予測子を使わなくても計算した結果をなるべく早く使うという方法によっても、反復回数を低減できることも知られていて、この方法を「ガウスザイデル法」と呼びます。ガウスザイデル法では計算した結果をすぐに次の計算に用いるため、先ほどの計算の影響の及ぶ範囲が隣接点よりもずっと広くなることが分かると思います。SORも、通常ガウスザイデル型の反復を基本としています(あたりまえですね、反復回数を減らそうとしているのですから)。ところが、ガウスザイデル型の反復では、計算式の1つ1つに直前の計算への依存性が出ていて、計算を並列に実行したりすることが困難です。これに対してヤコビ法はそれぞれの計算式が完全に独立しているため、並列化が容易な方法として知られています。

今回は直接は関係ありませんが、姫野ベンチではヤコビ法を用いているため、ベクトルや並列コンピュータを使っても容易に性能を上げることができるようになっています。

内部ループはJacobiのサブルーチンになっています。この内周のループを見ると分かりますが、ヤコビ法の基本演算は隣接点のポテンシャルを用いた積和演算となっています。そのためここでは算術関数は全く使っていま

実行例2

```
------------------------------------------------------------
1: NEC UP4800/770AD R10000 200MHz
コマンドオプション：   f77 -dn -Kopt=4

  mimax=         129 mjmax=          65 mkmax=          65
  imax=         128 jmax=          64 kmax=          64
 cpu :    8.090000    sec.
  Loop executed for           13 times
  Gosa :  3.0001516E-03
  MFLOPS measured :   26.46230
  Score based on MMX Pentium 200MHz :  0.8200279

------------------------------------------------------------
2: VT-Alpha 5/600 Alpha 21164A 600MHz(Debian GNU Linux/2.1)
コマンドオプション：   g77 -O2

  mimax= 129 mjmax= 65 mkmax= 65
  imax= 128 jmax= 64 kmax= 64
 cpu :   2.62890601sec.
  Loop executed for  13 times
  Gosa :  0.0030001516
  MFLOPS measured :  81.433136
  Score based on MMX Pentium 200MHz :  2.52349353

------------------------------------------------------------
3: VT-Alpha 5/600 Alpha 21164A 600MHz(Debian GNU Linux/2.1)
コマンドオプション：  fort -O5

  mimax=         129  mjmax=          65  mkmax=          65
  imax=         128  jmax=          64  kmax=          64
 cpu :    1.999024     sec.
  Loop executed for           13  times
  Gosa :  3.0001516E-03
  MFLOPS measured :   107.0923
  Score based on MMX Pentium 200MHz :   3.318633

------------------------------------------------------------
4: AlphaStation 200 4/100 Alpha 21064A 100MHz(Debian GNU Linux/
2.1)
コマンドオプション：   g77 -O2

  mimax= 129 mjmax= 65 mkmax= 65
  imax= 128 jmax= 64 kmax= 64
 cpu :   15.1826181sec.
  Loop executed for  13 times
  Gosa :  0.0030001516
  MFLOPS measured :  14.100337
  Score based on MMX Pentium 200MHz :  0.436948776

------------------------------------------------------------
5. NEC NX7000/250 (HP D470相当) PA-8000 160MHz
コマンドオプション：f77 +O4 -Ddtime=dtime_ -lU77

  mimax= 129 mjmax= 65 mkmax= 65
  imax= 128 jmax= 64 kmax= 64
 cpu :  3.23sec.
  Loop executed for  13 times
  Gosa : 3.00015E-03
  MFLOPS measured : 66.27866
  Score based on MMX Pentium 200MHz : 2.05388
```

せん。今回のFortranのリリースでは、算術ライブラリとコンパイラの両方がリリースされているので、性能が向上した場合にどちらが寄与しているのかが問題になります。ところが、姫野ベンチでは算術関数の呼び出しを行っていないのでCPMLの性能ではなく純粋にコンパイラの性能が高かったと結論づけられるでしょう。

大規模な構造や流体関係の計算には、このように線形の範囲で大量のデータを読みながら積和演算を行うものが多いのです。こういったベンチマークでもっとも影響があるのがメモリのスループットになるのですが、コンパイラのスケジューリングがこれほどまでに利くとは、実際、私自身驚いています。おそらくコンパイラは命令のスケジュールだけでなく配列のメモリ上の配置にも手を入れているのではないでしょうか？これがメモリスループットを大幅に向上させた21264になるとさらに一段と高い性能が得られる可能性があります。21264でのベンチマーク結果は今回は間に合いませんが、出来次第、報告します。

読者の皆さんも珍しいマシンを持っていたら性能評価をした結果を姫野さんに送ってあげてください。最後にベンチマークコードを示します(リスト1)。快く掲載の許可を頂けた姫野さんに感謝致します。

Fortranやライブラリが出てきて、その出来が思ったよりいいのでLinuxを数値計算のベースとすることが大変現実味を帯びてきています。今回用いた姫野ベンチでも、CompaqのFortranを用いてLinux/Alphaが大変優れた結果を出しています。安定性の向上と細かな調整を続けてLinux/AlphaがHPC分野のスタンダードとなる日も近いかも知れませんね。

リスト1　ヒノメベンチ

```
C**************************************************************
C
C This benchmark test program is measuring a cpu performance
C of floating point operation and memory access speed.
C
C Modification needed for testing turget computer!!
C Please adjust parameter : nn to take one minute to execute
C all calculation. Original parameter set is for PC with
C 200 MHz MMX PENTIUM, whose score using this benchmark test
C is about 32.3 MFLOPS.
C
C If you have any question, please ask me via email.
C written by Ryutaro HIMENO, October 3, 1998.
C Version 2.0
C ----------------------------------------------
C Ryutaro Himeno, Dr. of Eng.
C Head of Computer Information Center,
C The Institute of Pysical and Chemical Research (RIKEN)
C Email : himeno@postman.riken.go.jp
C ----------------------------------------------------------
------
C You can adjust the size of this benchmark code to fit your
target
C computer. In that case, please chose following sets of
C (mimax,mjmax,mkmax):
C small : 129,65,65
C midium: 257,129,129
C large : 513,257,257
C ext.large: 1025,513,513
C This program is to measure a computer performance in MFLOPS
C by using a kernel which appears in a linear solver of
pressure
C Poisson included in an incompressible Navier-Stokes solver.
C A point-Jacobi method is employed in this solver.
C ------------------
C Finite-difference method, curvilinear coodinate system
C Vectorizable and parallelizable on each grid point
C No. of grid points : imax x jmax x kmax including boundaries
C ------------------
C A,B,C:coefficient matrix, wrk1: source term of Poisson
equation
C wrk2 : working area, OMEGA : relaxation parameter
C BND:control variable for boundaries and objects ( = 0 or 1)
C P: pressure
C -----------------
C ------------------
C "use portlib" statement on the next line is for Visual
fortran
C to use UNIX libraries. Please remove it if your system is
UNIX.
C --------------------
C use portlib
C
      PARAMETER (mimax=129,mjmax=65,mkmax=65)
C     PARAMETER (mimax=257,mjmax=129,mkmax=129)
C     PARAMETER (mimax=513,mjmax=257,mkmax=257)
      PARAMETER (nn=13)
C
C Arrey
      common /pres/ p(mimax,mjmax,mkmax)
      common /mtrx/ a(mimax,mjmax,mkmax,4),
     1   b(mimax,mjmax,mkmax,3),c(mimax,mjmax,mkmax,3)
      common /bound/ bnd(mimax,mjmax,mkmax)
      common /work/ wrk1(mimax,mjmax,mkmax),wrk2(mimax,mjmax,mkmax)
C
C other constants
      common /others/ imax,jmax,kmax,omega
C
      dimension time0(2),time1(2)
C
      omega=0.8
      imax=mimax-1
      jmax=mjmax-1
      kmax=mkmax-1
C
C Initializing matrixes
      call initmt
      write(*,*) ' mimax=',mimax,' mjmax=',mjmax,'
mkmax=',mkmax
      write(*,*) ' imax=',imax,' jmax=',jmax,' kmax=',kmax
C
C Start measuring
```

リスト1　つづき

```
c
      cpu0=dtime(time0)
C
C Jacobi iteration
      call jacobi(nn,gosa)
C
      cpu1= dtime(time1)
      cpu = cpu1
      write(*,*) 'cpu : ',cpu,'sec.'
      nflop=(kmax-2)*(jmax-2)*(imax-2)*34
      if(cpu1.ne.0.0) xmflops2=nflop/cpu*1.0e-6*float(nn)
      write(*,*) ' Loop executed for ',nn,' times'
      write(*,*) ' Gosa :',gosa
      write(*,*) ' MFLOPS measured :',xmflops2
      score=xmflops2/32.27
      write(*,*) ' Score based on MMX Pentium 200MHz :',score
      pause
      END

      subroutine initmt

      PARAMETER (mimax=129,mjmax=65,mkmax=65)
C     PARAMETER (mimax=257,mjmax=129,mkmax=129)
C     PARAMETER (mimax=513,mjmax=257,mkmax=257)
C
C Arrey
      common /pres/ p(mimax,mjmax,mkmax)
      common /mtrx/ a(mimax,mjmax,mkmax,4),
     1   b(mimax,mjmax,mkmax,3),c(mimax,mjmax,mkmax,3)
      common /bound/ bnd(mimax,mjmax,mkmax)
      common /work/ wrk1(mimax,mjmax,mkmax),wrk2(mimax,mjmax,mkmax)
C
C other constants
      common /others/ imax,jmax,kmax,omega
C
      do k=1,mkmax
        do j=1,mjmax
          do i=1,mimax
              a(i,j,k,1)=0.0
              a(i,j,k,2)=0.0
              a(i,j,k,3)=0.0
              a(i,j,k,4)=0.0
              b(i,j,k,1)=0.0
              b(i,j,k,2)=0.0
              b(i,j,k,3)=0.0
              c(i,j,k,1)=0.0
              c(i,j,k,2)=0.0
              c(i,j,k,3)=0.0
              p(i,j,k) =0.0
              wrk1(i,j,k)=0.0
              bnd(i,j,k)=0.0
          enddo
      enddo
      enddo
      do k=1,kmax
        do j=1,jmax
          do i=1,imax
              a(i,j,k,1)=1.0
              a(i,j,k,2)=1.0
              a(i,j,k,3)=1.0
              a(i,j,k,4)=1.0/6.0
              b(i,j,k,1)=0.0
              b(i,j,k,2)=0.0
              b(i,j,k,3)=0.0
              c(i,j,k,1)=1.0
↘             c(i,j,k,2)=1.0
              c(i,j,k,3)=1.0
              p(i,j,k) =float((k-1)*(k-1))/float((kmax-1)*(kmax-1))
              wrk1(i,j,k)=0.0
              bnd(i,j,k)=1.0
            enddo
          enddo
        enddo
        return
        end

        subroutine jacobi(nn,gosa)
        PARAMETER (mimax=129,mjmax=65,mkmax=65)
C       PARAMETER (mimax=257,mjmax=129,mkmax=129)
C        PARAMETER (mimax=513,mjmax=257,mkmax=257)
C
C Arrey
        common /pres/ p(mimax,mjmax,mkmax)
        common /mtrx/ a(mimax,mjmax,mkmax,4),
       1   b(mimax,mjmax,mkmax,3),c(mimax,mjmax,mkmax,3)
        common /bound/ bnd(mimax,mjmax,mkmax)
        common /work/ wrk1(mimax,mjmax,mkmax),wrk2(mimax,mjmax,mkmax)
C
C other constants
        common /others/ imax,jmax,kmax,omega
C
        do 900 loop=1,nn
        gosa=0.0
        DO 100 K=2,kmax-1
        DO 100 J=2,jmax-1
        DO 100 I=2,imax-1
        S0=a(I,J,K,1)*p(I+1,J,K)+a(I,J,K,2)*p(I,J+1,K)
       1     +a(I,J,K,3)*p(I,J,K+1)
       3     +b(I,J,K,1)*(p(I+1,J+1,K)-p(I+1,J-1,K)
       *     -p(I-1,J+1,K)+p(I-1,J-1,K))
       4     +b(I,J,K,2)*(p(I,J+1,K+1)-p(I,J-1,K+1)
       *     -p(I,J+1,K-1)+p(I,J-1,K-1))
       5     +b(I,J,K,3)*(p(I+1,J,K+1)-p(I-1,J,K+1)
       *     -p(I+1,J,K-1)+p(I-1,J,K-1))
       6     +c(I,J,K,1)*p(I-1,J,K)+c(I,J,K,2)*p(I,J-1,K)
       *     +c(I,J,K,3)*p(I,J,K-1)+wrk1(I,J,K)
        SS=(S0*a(I,J,K,4)-p(I,J,K))*bnd(I,J,K)
        GOSA=GOSA+SS*SS
        wrk2(I,J,K)=p(I,J,K)+OMEGA *SS
100     CONTINUE
C
        DO 200 K=2,kmax-1
          DO 200 J=2,jmax-1
            DO 200 I=2,imax-1
              p(I,J,K)=wrk2(I,J,K)
200     CONTINUE
C
900     continue
C
C End of iteration
        return
        end
```