U.D.C. 681.322.012-181.2-185.4:681.32.068

スーパーコンピュータHITAC S-810 FORTRANコンパイラ“VOS3 FORT77/HAP”

S-810 FORTRAN Compiler “VOS3 FORT77/HAP”

青山明夫*　*Akio Aoyama*
安村通晃**　*Michiaki Yasumura*

近年，科学・技術計算の規模は著しく増大している。この計算規模大形化にこたえるものとしてスーパーコンピュータS-810が開発された。S-810 FORTRANコンパイラは，S-810の性能を引き出すと同時に，使いやすいインタフェースを提供する。S-810の性能はベクトル演算の比率を高めることで引き出されるが，それはS-810 FORTRANコンパイラの高度な自動ベクトル化機能により実現される。また，自動ベクトル化機能により，通常のFORTRANプログラムでS-810を使える。

大規模数値計算で高速性と並んで必要な大容量記憶については，拡張記憶をFORTRAN言語レベルでサポートしている。また，ベクトル化率向上のためのFORTRANプログラムチューニング支援ツールも提供している。

1 緒　言

近年，大形計算機を用いた大規模数値計算の需要はその規模，性能の両面で増大する一方であり，現在の汎用計算機をもってしては，その需要に応じきれないところまできている。このため，科学技術計算専用の超高性能計算機(いわゆるスーパーコンピュータ)の出現が待望されていたが，日立製作所ではこのたびS-810アレイプロセッサを開発した[1]。

一般にアレイプロセッサは，ベクトル演算を高速に処理するもので，ベクトル命令と呼ばれる特別な命令セットをもっている。しかし，直接にベクトル命令を使ってプログラムを作ることは，手間がかかる上にプログラムの移行性を著しく損なう。そのため，ベクトルコンパイラと呼ばれる特別なコンパイラが必要になる。ベクトルコンパイラは,通常のFORTRANプログラムのDOループから，ベクトル化※1)可能な部分(**図1**)を検出し，それをベクトル命令列として出力する。これを自動ベクトル化[2]という。特に，S-810 FORTRANコンパイラ〔以下，FORT77/HAP(High speed Array Processor)と呼ぶ。〕は，従来のベクトルコンパイラよりもベクトル化可能性の検出能力が高く，かつS-810の高度なハードウェアアーキテクチャを十分に生かした，高い効率のベクトル命令列を生成する能力をもつため，FORT77/HAPによってS-810の性能を容易に引き出せる。

この論文では，まずFORT77/HAPの自動ベクトル化機能についてやや詳しく述べ，次に，FORT77/HAPの使いやすさ，すなわち従来プログラムからの移行の容易さについて述べる。S-810は，単に演算の高速性を提供するばかりではなく，磁気ディスク装置に比べて150あるいは300倍のデータ転送速度をもつ拡張記憶を備えている。大行列の計算に必要な拡張記憶のFORTRAN言語レベルでのサポートについて次に述べる。FORT77/HAPは高度な自動ベクトル化機能をもっているが，プログラマがプログラムをS-810用に少し手直しすると，更にベクトル化可能部分が広がり，高速性が期待できる場合がある。このような，ベクトルプロセッサ向けのFORTRANプログラムチューニングを支援するツールであるVECTIZERの機能を最後に説明する。

2 自動ベクトル化

S-810のようなスーパーコンピュータでは，ベクトル命令部分の実行速度は通常命令部分に比べて10倍以上ときには50倍以上もの性能比があるが，一つのプログラム全体での総合性能比は,一般にはこれほど高くならない。それはFORTRANプログラム内に，ベクトル化できない部分が残るためである。一つのプログラムの中でベクトル化可能部分の占める割合を

```
      DO  10   I=1,N                    DO  10   I=2,N
        A(I+1)=…                          A(I-1)=…
            =A(I)                             =A(I)
   10 CONTINUE                       10 CONTINUE
```

(a) ベクトル化可能な例　　(b) ベクトル化不可能な例

図1　ベクトル化可能なDOループと不可能なDOループ　(a)では実行順序に沿って後方にある参照A(I)が前方にあるA(I+1)で定義した値を参照するので，ベクトル化しても実行結果は変わらない。逆に(b)では，後方にある参照A(I)は前方にあるA(I−1)で定義した値ではなくDOループ実行前の値を参照しているので，ベクトル化すると実行結果が変わる。

※1) 通常のDOループの実行順序では，制御変数(**図1**では変数I)を固定してDOループ内の文を上から下へ実行し，次に制御変数を増加させて同じことを繰り返す。ベクトル化とは，この実行順序を次のように変更することである。すなわち，まず最初の文だけを制御変数の値すべてに対して実行してしまい，次に2番目以降の文に対して同じことを繰り返す。ベクトル化によって初めてベクトル命令の適用が可能になる。

* 日立製作所ソフトウェア工場　** 日立製作所中央研究所

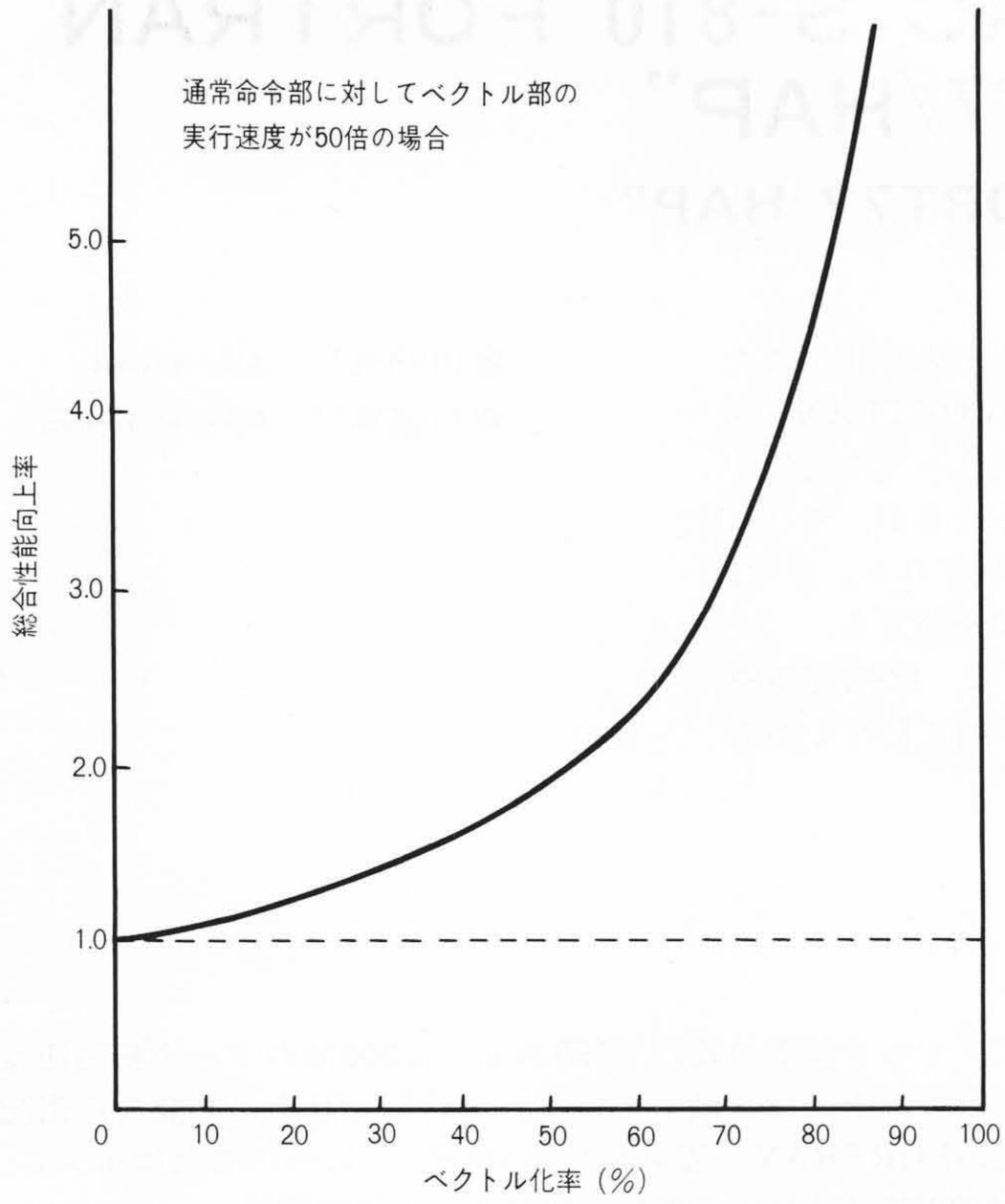

図2 ベクトル化率と総合性能向上率の関係 ベクトル化率50%では2倍弱であるが，ベクトル化率90%では8.5倍，95%では14.5倍になる。ベクトル化率の重要性が分かる。

ベクトル化率と呼ぶが，**図2**にベクトル化率と総合性能向上率との関係を示す。同図からも分かるとおり，ベクトル化率が50%程度では総合性能向上率はたかだか2倍しかならない。ところが，ベクトル化率が90%を越すと総合性能向上率は8倍以上となる。すなわち，ベクトル化率を上げることが，自動ベクトルコンパイラに課せられた極めて重要な役割といえる。従来のベクトルコンパイラでは多くのベンチマークプログラムに対して，ベクトル化率は平均で50%程度にしかならなかったが，FORT77/HAPでは，平均で80%[※2)]以上のベクトル化率を目指している。

表1にFORT77/HAPでの主要なベクトル化機能を示す。これらのベクトル化機能は，HITAC M-180～M-200H IAP[3)] (Integrated Array Processor)コンパイラ，及びHITAC M-280H IAPコンパイラのベクトル化機能を更に充実・拡張したものである。例えば，条件文に関しては，M-280H IAPコンパイラで実現した条件文の自動ベクトル化機能を更に充実し，対象範囲を拡大した。この場合の技術的な課題は，変数の定義参照関係の検査であったが，FORT77/HAPでは，これをより一般的に解決した(**図3**)。すなわち，従来は常に通るパスに変数の定義が先行しているか，あるいは定義が条件文下にある場合には，そのすべてのパスに定義があるときだけを許していたが，新たに**図3**の下方のように変数の定義がないパスがあってもよいように拡張した。

※2) この値はあくまでも平均であって，プログラムによっては90%以上(場合によって100%近く)になることもある。このようなベクトル化率の向上が，後で述べるVECTIZERの支援による(プログラマの)FORTRANプログラム修正でできることも多い。

表1 FORT77/HAPのベクトル化機能 整数型，論理型，間接指標ベクトル，文の入れ換え，最大値，最小値，DOループ別オプションなどが，FORT77/HAPで新たに開発したベクトル化技術の主なものである。

項番	項目	機能
1	ベクトル化の対象	最内側DOループ
2	データ形	実数(単・倍)，複素数(単・倍)，整数，論理型
3	文の種類	代入文，条件文(ブロックIF文を含む。)
4	配列添字の形式	線形添字，非線形添字(間接指標ベクトルなど)
5	プログラム変換	ループ分割，ループ展開，文の入れ換え
6	マクロ演算	累和，累積，内積，一次巡回演算，最大値，最小値
7	組込み関数	FORTRAN77水準のすべて(ただし，文字型を除く。)
8	DOループ別オプション	VEC(強制ベクトル化)，NOVEC(強制ベクトル化不能)など
9	その他	インデックス変数のベクトル化など

更に，配列の添字形として，従来からある線形添字のほかに，配列要素など一般に非線形な添字形式も許している(**図4**)。一般に添字式がDOの制御変数の一次式になっていないときは非線形添字と呼ぶが，特に添字式が整数型の配列であるとき

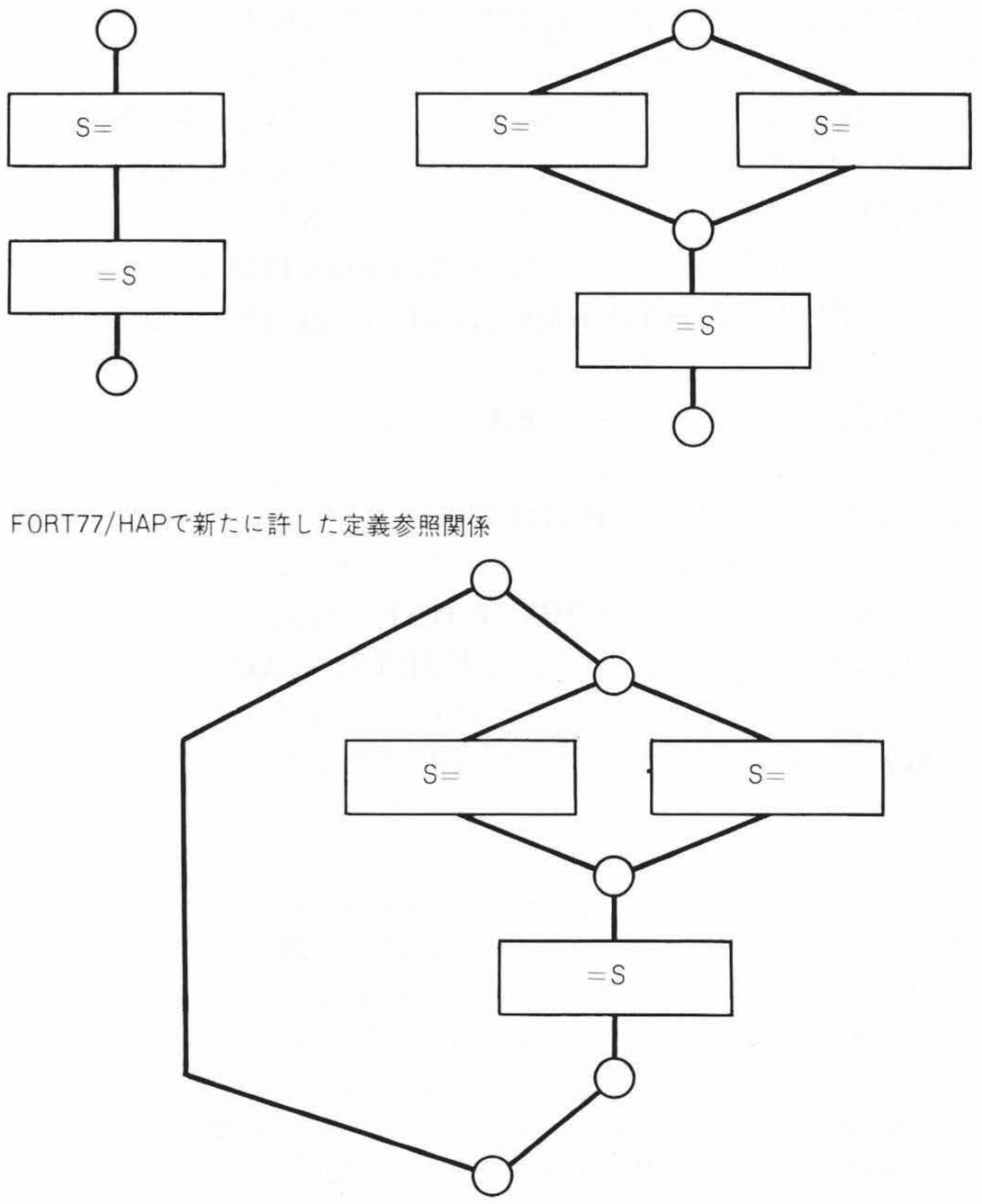

図3 条件文のネストと変数の定義参照関係 Sは変数を表わす。S=はSへの代入を，=Sは変数Sの参照を表わす。○—○などは制御の流れを表わす。FORT77/HAPでは，変数Sの定義が行なわれないパスがあってもよいように拡張した。

線形添字の例

A(N*I+M)

非線形添字の例

B(L(I))

C(I*I)

図4　添字形式の例　A，B，Cは配列を表わし，Lは整数形の配列を表わす。Iは制御変数である。非線形添字のうち，B(L(I))のようなものを間接指標ベクトルと呼ぶ。

は間接指標ベクトルと呼ぶ。間接指標ベクトルのベクトル化は，スパース行列(ほとんどの要素が0である巨大行列)を取り扱う場合に効果が期待できる。

プログラム変換機能としては，ループ中のベクトル化可能部分と不能部分とを自動分割するループ分割機能や，端点添字や巡回添字に対してループ本体を展開するループ展開機能(**図5**)のほかに，データの定義・参照関係がベクトル化に適さない場合に，文の入れ換えを行なう機能(**図6**)などがある。

このうち，ループ分割機能はループの一部だけにベクトル化不能要因があり，他はベクトル化できる場合に全体をベクトル化不能とせず，ベクトル化できるところとできないところに「上下」に分割してベクトル化できる部分はベクトル化することによって，ベクトル化率を維持するものである。また，ループ展開は，DO文の繰返しのうち特定の回次(例えば最初の繰返し)を除いて他の繰返しに限定してベクトル化する技術である。**図5**(a)はループ展開によって，より良いベクトル命令列が生成できるようになる例である。同図(b)はループ

```
      DO  10    I=1,N                    S=0.0
      IF(I.EQ.1)  THEN                   X(1)=S+B(1)+…
          S=0.0                          DO  10  I=2,N
      ELSE                      ⇒        S=A(I)
          S=A(I)                         X(I)=S+B(I)+…
      END  IF                        10  CONTINUE
          X(I)=S+B(I)+…
   10 CONTINUE
```

(a) ループ展開(1)——端点条件の展開

```
      J=N                                J=N
      DO  20    I=1,N                    X(1)=Y(J)+…
      X(I)=Y(J)+…                        J=1
      J=I                       ⇒        DO  20  I=2,N
   20 CONTINUE                           X(I)=Y(J)+…
                                         J=J+1
                                      20 CONTINUE
```

(b) ループ展開(2)——巡回インデックスの展開

図5　ループ展開　ループ展開は，DO文の繰返しのうち特定の回次(例えば最初の繰返し)を除いて，他の繰返しに限定してベクトル化する技術である。

```
      DO  10    I=1,N                    DO  10    I=1,N
   10 D(I)=A(I)+E(I)          ⇒      10 A(I-1)=B(I)+C(I)
      A(I-1)=B(I)+C(I)                  D(I)=A(I)+E(I)
```

図6　文の入れ換え　入れ換え前のプログラムでは，実行の順序に沿って後方の参照A(I)がDOループ実行前の値を参照しているためにベクトル化できないが，この場合は文の順序を入れ換えても他の配列(B，C，D，E)の定義参照順序に影響を与えないので，文の入れ換えによりベクトル化できる。

展開によってベクトル化不能であったものが可能になる例である。なお，ループ分割がループを「上下に」(すなわち，文の並ぶ方向に)分割するのに対して，ループ展開はループを「左右に」(すなわち，繰返しの方向に)分割するものであると見ることができる。

文の入れ換え機能は，**図6**の配列Aとそれ以外の配列B，C，D，Eのように，データ参照解析でベクトル化不能となる要因となっている配列(**図6**の配列A)以外のものが，文の入れ換えによって定義参照の順序関係に影響を受けない場合に，隣接する文を前後に入れ換えてベクトル化可能とするものである。

演算の種類としては，加・減・乗・除など配列の要素ごとに行なう演算のほか，累和・累積・内積・一次巡回演算など，配列全体の間で行なうマクロ演算もコンパイラで自動ベクトル化する(**表2**)。これらのマクロ演算は，連立一次方程式や固有値計算などの基礎的な数値計算プログラムでよく出現するものである。

これらのマクロ演算が適用されるケースは，通常のベクトル化のためのデータ参照解析の結果ではベクトル化不能な部類に属するものであるが，**表2**に示すような機能をもつベクトル命令があるために，それらを用いることによって初めてベクトル化可能となるもので，ベクトル化率向上への寄与が大きい。

更に，FORTRAN77の言語仕様に含まれる組込関数のうち文字型関数以外のすべてをベクトル命令で計算するベクトル関数ライブラリを用意している。これらの関数が条件文下で使われているときは，条件が成立する要素だけにベクトルを圧縮して短くしたのち，ベクトル関数値を計算する工夫もし

表2　マクロ演算一覧　マクロ演算は表中に示すように，数値計算で頻繁に現われる複合的な一連の演算を一つのベクトル命令で行なうものである。このような命令を用いることによって，定義・参照の順序関係から見ると，ベクトル化できない形の文がベクトル化できるようになる。

項番	項　目	形　式
1	累　和	S=S+A(i)
2	累　積	S=S*A(i)
3	内　積	S=S+A(i)*B(i)
4	一次巡回演算	A(i)=A(i-1)*B(i)+C(i)
5	最大値	S=MAX(S, A(i))
6	最小値	S=MIN(S, A(i))

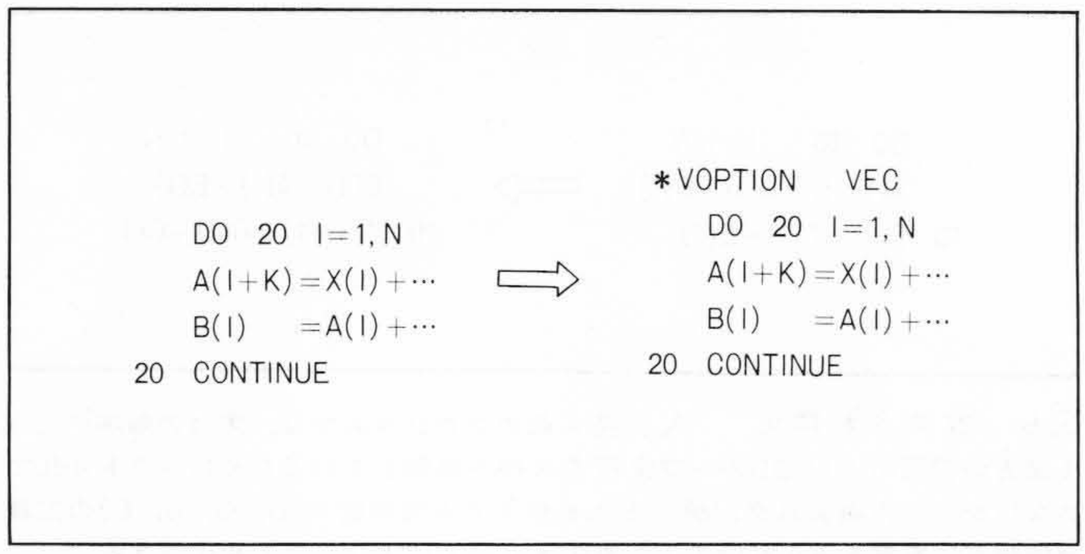

図7 配列の定義・参照関係不明の場合のオプション指定によるベクトル化 図中のKの値が分からないために，コンパイラにはベクトル化してよいかどうか分からない。しかし，A(I+K)による定義範囲とA(I)による参照範囲が重ならないと分かっているならば，＊VOPTIONパラメータでコンパイラに指示を与えることによってベクトル化できる。

ている。

ベクトル関数値を計算した後には，再び条件が成立しない要素を含めた元の状態に伸長する。この手法は，ベクトルの圧縮・伸長と呼ばれる。この手法により，条件文の下にある関数計算で条件の成立・非成立の情報を表わすマスク処理をベクトル関数内で行なう必要がなくなり，コンパイラの出力する命令例とベクトル関数とのインタフェースが単純になり，全体の効率が向上する。その他，配列の定義・参照の順序関係が不明でベクトル化が可能かどうかコンパイラに判断できないとき，DOループ別にベクトル化するかどうかをプログラマがコンパイラに指示するオプションも用意した(**図7**)。**図7**では，Kの値が分からないためにコンパイラにはベクトル化してよいかどうか判断できず，ベクトル化不能としてしまう。しかし，プログラマがKの値を知っていて，A(I+K)による定義範囲とA(I)による参照範囲が重ならないと分かっているならば，＊VOPTIONパラメータでVEC(Vectorigeの略)と指定することによって強制的なベクトル化を指示する。また，ループ回数がコンパイル時に不明(データとして読み込むときなど)であるが，実際にはループ回数が少なくベクトル化しても反って性能が低下する場合には，＊VOPTION NOVEC(No Vectorigeの略)を指定して，ベクトル化を抑止することができる。

なお，＊VOPTION VECによる強制ベクトル化は，コンパイラによるデータ参照解析結果が不明(ベクトル化に適合しているかどうか分からない。)の場合に有効であり，コンパイラにベクトル化不能とはっきり分かる場合には無効である。

表3 ベクトル命令列の最適化 項番1，2は，通常命令に対して一般のコンパイラが行なっていることをベクトル命令列に対しても適用したものである。項番3，4は，S-810のアーキテクチャを生かすために新たに開発した技術である。

項番	項目	備考
1	共通式削除	スカラテキスト最適化と共通技術
2	不変式追い出し	〃
3	ベクトルレジスタ割当て	ベクトルプロセッサ特有
4	通常命令との並列実行	S-810特有

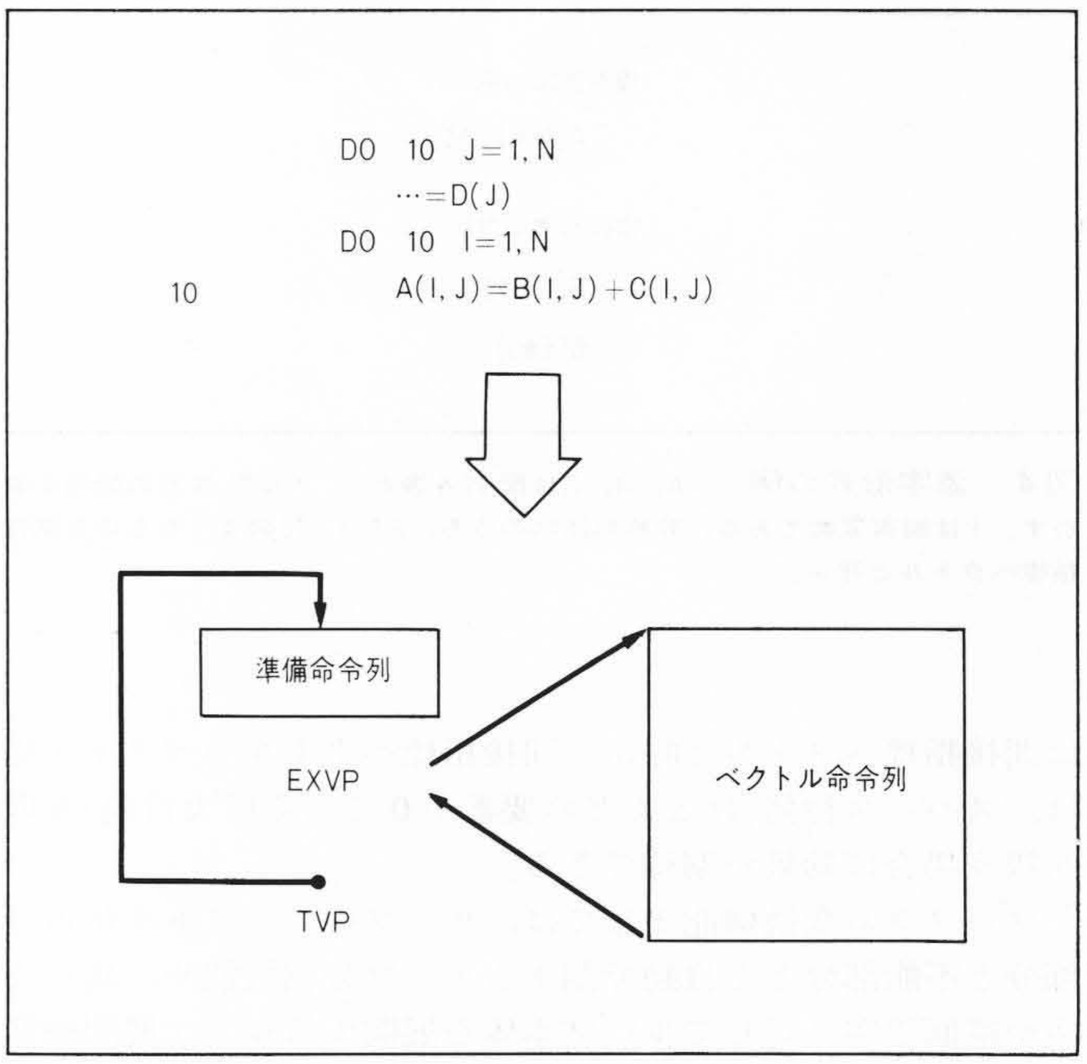

図8 ベクトル命令と通常命令の並列実行 二重ループになっている場合，外側DOループの繰返しについて見て，ある回次のベクトル命令の実行と，その次のベクトル命令に対する準備命令とが並列に実行できることがある。

すなわち，このときには＊VOPTION VECの指定があってもベクトル化しない。

以上述べてきた高度なベクトル化機能により，FORT77/HAPは高いベクトル化率を実現しているが，更にベクトル化した後のベクトル命令列に関してもS-810のもつ豊富なハードウェアリソースをより有効に生かすために，**表3**に示すようなベクトル命令列の最適化を行なっている。このうち，項番1，2は，汎用コンパイラの最適化と共通の技術である。項番3は，S-810が複数演算器をもち，かつレジスタと演算器との結合が極めて高度で複雑になっているために必要となるもので，汎用コンパイラのレジスタ割当てと大きく異なる新たな課題を解決する技術である(今回は紙面の都合上，その詳細は割愛する)。項番4も，S-810独特のアーキテクチャを生かすためのものであり，その例を**図8**に示す。このような，ベクトル命令列の種々の最適化により，FORT/77HAPは効率的なベクトル命令列を出力する。

3 移行性

近年の研究・開発に使われるプログラムは，大規模で複雑になってきている。また，既に開発した財産が多くあるうえに，世界的に流通しているプログラムをそのまま用いたいこともある。このような状況の下では，プログラムに大きな変更を施すことは困難である。したがって，従来の財産や国外のプログラムを生かせることが，スーパーコンピュータを使う際の手間を少なくする重要な条件の一つとなる。

移行性の具体的内容は以下のとおりである。

(1) 自動ベクトル化による標準言語仕様の採用

2章でも述べたとおり，FORT77/HAPは自動ベクトル化機能をもっている。このため，スーパーコンピュータのベクトル演算用の特別なFORTRAN言語仕様を習得する必要がなく，財産としてある既存のプログラムもFORT77/HAPで再コンパイルすることによってS-810に適用できる。

(2) 汎用FORTRANからの移行性

FORT77/HAPは，ベクトル演算用の特別な言語仕様がないだけでなく，汎用処理装置・汎用オペレーティングシステムの上で稼動している従来のFORTRANと同じ言語仕様であり，原始プログラムレベルでの移行性がある。このため，従来の汎用コンパイラを使う場合とほとんど同じ使い方でFORT77/HAPを使うことができる。

4 拡張記憶のサポート

拡張記憶は主記憶の下位に接続される半導体の大容量拡張記憶である。この拡張記憶をFORTRANプログラムから入出力文を用いてデータセットとしてアクセスできる。拡張記憶上のデータセットであることやその大きさは，OPEN文で指定する。拡張記憶上のデータセットは，ジョブステップ内の一時的データセットとしてだけ使え，ジョブステップ間の受け渡しはできない。

5 VECTIZER

5.1 VECTIZERの必要性

FORT77/HAPは高度な自動ベクトル化機能を備えているが，コンパイラにできる原始プログラムの変形には限界がある。また，値を入力データとして実行時に読み込むときのように，原始プログラム上には現われない情報があり，それを用いないとベクトル化判定ができない場合がある。そのようなときには，プログラマがFORTRANプログラムを多少書き変えることや，特定のDOループがベクトル化可能であることをコンパイラに連絡することが有効な方法である。

以上のプログラマとコンパイラの協力作業を円滑に進めるためのベクトル化アナライザがVECTIZERである。ベクトル命令を使って実行される部分の比率(ベクトル化率)を高めることがS-810の性能を発揮する上で大事なことは既に強調したとおりであり，そのために，VECTIZERによる原始プログラムのチューニングが重要になってくるわけである。

5.2 VECTIZERの特長

VECTIZERの出力情報を**表4**に示す。そこに示すとおりVECTIZERは原始プログラムのチューニング作業に必要な情報をすべて出力する。それらの情報を用いてチューニングを行なう手順を以下に示す。

(1) まず，VECTIZERの出力するベクトル化率を見て，それが十分であるかどうかを判定する。

(2) 十分でなければDOループごと(どのサブルーチンに属するDOループであるかも同時に表示される。)の実行比率を見て，比率の高いDOループがベクトル化されているかどうかを調べる。

(3) 実行比率の高いDOループがベクトル化されていなければ，そのDOループがベクトル化されない理由をベクトル化診断メッセージを参照して調べ，ベクトル化不能要因を解消する。

(4) 以上を十分なベクトル化率に達するまで繰り返す。

なお，VECTIZERが各DOループの実行比率を算出する過程では，実際に通常の命令でプログラムを実行する。したがって，長時間を要するプログラムでは，チューニング時にはデータを適当に変更する必要がある。

チューンアップの具体例を**図9**に示す。この例は，原始プログラムの変更によるベクトル化で，DOループ内のIF文による判定条件が繰返しごとに不変であることを利用し，DO文とIF文の包含関係を逆転させて，ベクトル化不能要因であるループ外への分岐をなくしている。

また，2章で説明した**図7**は，プログラマが＊VOPTIONパラメータを与えてコンパイラにベクトル化してよいことを連絡することによりベクトル化する例である。この場合，もし誤って＊VOPTIONパラメータを与えると不当な計算結果を出力することがある点に注意を要する。

```
      DO  10  I=1,N
        IF(A(J).EQ.0) GO TO  20
10      B(I,J)=B(I,J)+A(J)*C(I,J)
20    .....

                  ⇩

      IF(A(J).NE.0) THEN
        DO  10  I=1,N
10        B(I,J)=B(I,J)+A(J)*C(I,J)
      ENDIF
```

図9 チューンアップの具体例　FORTRANコーディングに小変更を加えて，ベクトル化する例を示す。

6 結　言

スーパーコンピュータS-810のFORTRANコンパイラであるFORT77/HAPの自動ベクトル化機能，移行性，拡張記憶サポート，VECTIZERについて述べた。いずれの技術も発展途上のものであり，更に高速で使いやすいコンパイラにするように今後とも努力する考えである。

表4 VECTIZERの出力情報　VECTIZERは原始プログラムのチューンアップに必要な情報をすべて出力する。

項番	出　力　情　報
1	ベクトル化率及び最内側の実行比率
2	プログラム単位(サブルーチン)ごとの実行比率
3	DOループごとの実行比率
4	ベクトル化不能要因(診断メッセージ)

参考文献

1) 小高，外：最大性能が630MFLOPSで1Gバイトの半導体拡張記憶が付くスーパコンピュータHITAC S-810，日経エレクトロニクス，**314**，159～184(昭58-4)

2) 梅谷，外：技術計算プログラムの自動ベクトル化技術，情報処理，**23**，1，29～40(昭57-1)

3) 小高，外：HITAC M-180内蔵アレイプロセッサ，日立評論，**60**，6，451～456(昭53-6)

論文抄録

VI. MOS FET

日立製作所　永田　穣・岡部健明

電気学会雑誌　58-1，19～21（昭58-1）

ディジダルLSIの主要素子であるMOS FET（Metal Oxide Semiconductor Field Effect Transistor）を高耐圧，大電流化した新しいパワー素子が目覚ましい勢いで発展している。すなわち，MOS FETのもつ高速動作，熱的安定性，高入力インピーダンス特性，エンハンスメント動作などパワー素子として優れた性質が注目され，各所で活発に研究開発が進められている。

MOS FETの高耐圧化に関する研究は1970年代に入って本格的に始められ，ゲート電極とドレインとの重なりをなくしたオフセットゲート構造，ドレイン近傍に低不純物濃度のドリフト領域を設けた二重拡散構造などが発表されている。前者はソース，ドレイン，ゲートのすべての電極が表面にあり，電流が表面近傍だけを流れる横形構造パワーMOSとして，後者はドレイン電極を裏面にもつ縦形構造パワーMOSとして実用化されている。いずれの構造も高耐圧化は，ゲート電極端の電界集中を緩和すると同時に，ドレイン近傍の幅広い空乏層によって大部分の電圧を吸収するように設計されている。パワーMOSは高耐圧構造MOS FETセルをチップ上で多数個並列接続することによって，等価的に大きなチャネル長を実現したデバイスであるから，能率の良い高耐圧化構造と，単位面積当たりのセル密度を高めることが設計の重要なポイントとなる。例えば，同一レイアウト規則に従い設計されたnチャネル素子の単位面積当たりのコンダクタンス（オン抵抗の逆数）は，高耐圧領域ではドレイン耐圧のほぼ1.8乗に比例して減少しており，高耐圧化に伴ってオン抵抗が著しく増大する様子が分かる。また，横形構造は縦形構造に比べ，同一面積ではオン抵抗が高くなるが，破壊耐力，高周波特性などの点で優れているため，用途による使い分けが必要であろう。

初期のパワーMOS製品は，ドレイン耐圧60～90V，電流2A程度のものであったが，現在はドレイン耐圧800V，電流20A程度の素子まで幅広く市販されている。耐圧，電流以外の特性でも低オン抵抗素子から高速素子まで，各用途に応じたデバイスがそろえられている。

パワーMOSの用途は，高速動作と大きな破壊耐力という特長を生かし，スイッチング電源，DC-DCコンバータ，放電加工機，各種超音波発生装置，大電力送信機，オーディオアンプなど多岐にわたっている。高速スイッチング用途以外にも，周波数は低いが使いやすさ，耐破壊力などの理由から各種電動機制御，電子リレー，あるいは端末装置の駆動用などに使われるであろう。これらの用途では，スイッチング速度よりも低損失であることが望まれている。また，高周波帯への適用も注目される応用分野の一つであり，高周波特性に優れ，かつ負荷変動に強いパワーMOSの実現によって，動作電流の低減など高信頼度設計を可能とした。

目的樹木作成技法“PPDS”の開発

日立製作所　春名公一・薦田憲久・他2名

計測と制御　22-2，185～200（昭58-2）

複雑なシステムについて，それぞれの関与者がもっている要求は，企業のように比較的組織化が進んでいる場所でも，各関与者の使命観，経験，立場などの違いから局面的であり，全体として斉合がとれていないのが普通である。したがって，システムの要求分析では，各関与者がそれぞれの局面的な要求，問題意識をもち寄り，相互に啓蒙，協調，競合し合うなかで，全体の要求として選択，修正，整理されシステム課題の解決に役立つことが必要である。特に，提起された要求のなかには全体目的に部分的に反するものを含んでいる場合があり，その対策として新たな要求が発生する，というような学習が行なわれる。

本論文は，このようなシステム要求分析という活動を分析し，目的樹木を作成する方法とその支援システムPPDS（Planning Procedure to Develop System）を開発し，企業での種々のシステム課題に適用した結果を報告する。

PPDSによる目的樹木作成法では，ブレーンライティング法を改良した関与者利害関連表に基づいて発掘された問題点，要求が対話形グラフ処理システムを用いて目的樹木に構造化される。

関与者利害関連表を用いることによって，(1)10人前後のメンバーが2時間で200～300個の問題提起を行なうことができる，(2)各メンバーが対等に自分の立場から問題提起を行なえるため，参加意識が高くかつ具体的な問題をとらえることができる，(3)問題発掘が十分に広い範囲にわたってほぼ漏れなく行なわれたことを容易に確認できる，(4)発掘された問題のうち，どこを目的樹木にするか，すなわち問題領域の設定が行ないやすい，などの効果を得ることができる。

構造化アルゴリズムとしてフィードバック処理のためのヒューリスティックアルゴリズム“HSA”を開発し，目的樹木の作成支援に不可欠なサイクルを含むグラフ構造の処理を可能にした。この“HSA”を核に，目的樹木の修正，分割，統合，蓄積，検索などの機能をもつ対話形支援システムを開発することによって，従来人手では困難であった大規模な目的樹木（項目数200以上）の作成を可能にした。

関与者利害関連表，対話形計算機支援システムを用い，多関与者から成るシステムの問題点，要求を発掘し目的樹木に構造化する一連の手順を開発することによって，従来手法と比較して項目数100で2倍，150で4倍，200以上も可能という効率の向上が得られた。

以上の結果，企業の実際活動で，管理者を含む多部門の関与者の参加による問題発掘構造化が実用化された。

一方，PPDSは目的樹木作成以外にも，大規模シミュレーションモデルの作成，フォールトトリー作成，などの複雑な構造をもつシステムの構造分析，あるいは表現のために用いることができる。