資　料

スーパーコンピュータSX-1の性能向上支援ツールの利用法

後藤　米子*　三原　敏敬**

1. はじめに

本稿ではスーパーコンピュータSX-1システム（以下、SXと略す）に用意されているANALYZER/SX[15,16]、VECTORIZER/SX[15,17]およびOPTIMIZER[15,18]の利用法、主としてその操作法を説明する。これらは“性能向上支援ツール”（以下ツールと略す）とよばれるサービスプログラムで、利用者のプログラムの実行時の性能の解析をおこなったり、その結果に基づいたプログラムの書き換えを支援するためのものである。書き換えの指針については、すでに解説[5]がセンターニュースに掲載されているのでそれを参照していただきたい。

2節では、ツールの相互の関連とその利用形態、ソースプログラムの形式などに関する制約事項を述べる。すべてを列挙できないので、使用するにあたってはマニュアル15)～18)を参照していただきたい。なお、ツールから出力される情報を活用した性能向上のはかり方についても簡単にふれる。これに関連する記述はこのままの形では現在マニュアルにはなく、本稿では当センター主催の第2回SXプログラミング研究会での説明をもとに補足したものである。本稿でいうベクトル化率αは、文献などで通常用いられている“ベクトル化率”の近似値である。3節以降で各ツールのより詳細な機能、ジョブ制御言語（Job Control Language；JCLと略す）、起動コマンドなどの操作法について述べ、ツール個別の注意事項を説明する。ANALYZER/SXに関する記述が多いので、3節では簡易形を、4節では基本形を述べるが、他の2つは簡易形と基本形の双方を同じ節で述べる。SXのTSS[6,7]とジョブ制御言語[9]についてはすでに解説があるので、必要に応じて参照していただきたい。

2. ツールの概要とその活用

2.1 ツールの特徴

各ツールの主な特徴は次のとおりである。

ANALYZER/SXでは利用者のプログラムを解析して、次の情報を出力する。

- プログラムのベクトル化率
- 高い実行費用を占めるプログラム単位および実行文、DOループに関する情報

＊ 大阪大学大型計算機センター研究開発部　　＊＊ 関西日本電気ソフトウェア㈱

○プログラムのモジュール構造

○ベクトル化情報ファイルの内容

実行費用が高いとは、実行時にCPU時間を集中的に消費していること、またはその部分の実行回数がきわめて多いことをいう。利用者はこれらの情報を**2.3**節で述べるような順序で収集し、その結果書き換えが必要となれば、VECTORIZER/SXを使用する。

VECTORIZER/SXではANALYZER/SXが出力したベクトル化情報ファイルとそれを作成した時点でのソースプログラムとをもとに、プログラムの書き換えを画面エディタ[19]で対話的におこなう。さらにこの中からコマンドでFORTRAN77/SXのコンパイラを起動できるので、書き換えによるベクトル化の可否をその診断メッセージから直ちに知ることができる。ただし、その使用は画面エディタが使用できる端末に限定される。

OPTIMIZERでは、ソースプログラムを入力として手続きのインライン展開などの最適化をおこなった新たなソースプログラムを出力する。

各ツールの利用形態と処理形態を**表1**に、その相互の関係を**図1**に示す。補足事項を次に挙げる。

○ ANALYZER/SXでは複数個の作業用ファイルを使用する。簡易形ではこれらの容量を変更できないため、既定値で不足するときには基本形を使用しなければならない。

表1 ツールの利用形態

	ANALYZER/SX	VECTORIZER/SX	OPTIMIZER
簡易形	○（バッチ）	○（TSS）	×
基本形	○（バッチ, TSS）	○（TSS）	○（バッチ, TSS）

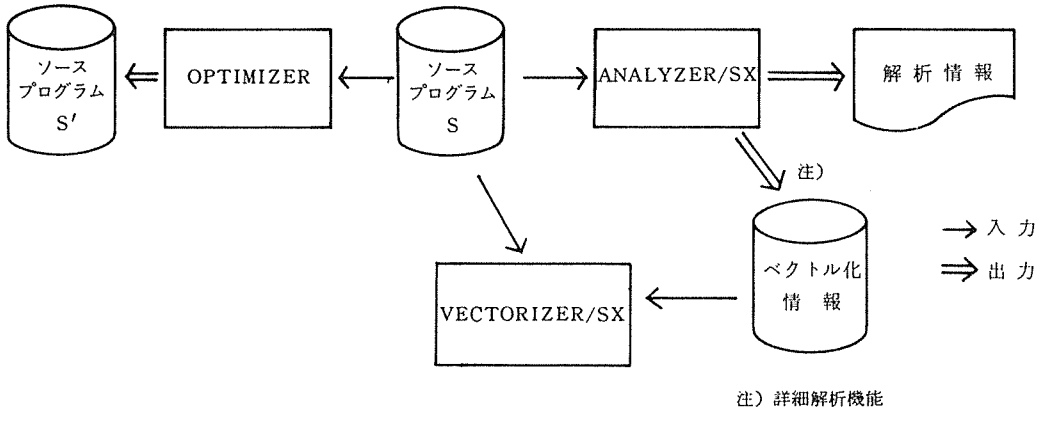

図1 ツールの関係とデータの流れ

- VECTORIZER/SXと連動させないで ANALYZER/SX だけを使用することは可能である。
- VECTORIZER/SXを使用する場合は、ベクトル化情報ファイルを作成したときのソースプログラムを使用以前に編集してはならない。
- ベクトル化情報ファイルは、簡易形では ACOS1000 側に、基本形では SX 側に持つ。ANALYZER/SXを使用する前に、利用者がファイルの作成(アロケートともいう)[6,9] をおこなわなければならず、この内容は詳細解析機能を用いたときに書き込まれる。

ベクトル化情報ファイルの属性と容量の目安は次のとおりである。

- 順編成であること。基本形では、レコード形式 VB、ブロック長 1036 バイト、レコード長 1032 バイトである。
- ソースプログラムの行数が 1000 行で約 100 ブロック、5000 行で約 500 ブロックである。

2.2 使用上の制約

ツールを使用する場合に、ソースプログラムの形式、使用端末などに制約がある。とくに断りのないものは各ツールに共通である。

- ソースプログラムの制約
  - 一つのファイルにまとめられていること。
  - 多重文は許されない。
  - A_FAN_ の 6 文字ではじまる英字名を用いてはならない。
  - 固定形式であること。
  - VECTORIZER/SXを使用する場合、行番号つきかつ固定形式でなければならない。この形式は ACOS1000 の Fortran には存在しないので、簡易形で使用するときには固定形式に RESE# コマンドで行番号を付加すること。
  - VECTORIZER/SXを使用するときにはソースプログラムを直接 JCL と同じファイルに記述できない。
- 使用できる端末
  - VECTORIZER/SXは接続手順が ETOS31K または ETOS52G でなければならず、一般の無手順端末では使用できない。

2.3 解析情報の収集と活用

SXのスーパーコンピュータたる性能を十分に引き出すには、すでに繰返しいわれているように、

(a) プログラムのベクトル化率を可能なかぎり高める、

(b) そのベクトル命令の効率をよくする、

ことがきわめて重要である。(a)ではベクトル化率 α が、(b)では後で述べる加速率 r および平均

のベクトルの長さ l がわかれば、その評価が可能である。ANALYZER/SX を下記の順序で使用すれば、これらの判断の根拠となる情報が直接または間接に収集できる。なお、本稿でいう“ベクトルの長さ”はマニュアル[13,16]および解説[5]では“ループ長”と呼ばれているものである。

以下に性能向上のために標準的な手順を示す。一般利用者にとっては、ベクトル化率90%以上を達成することは難しいであろう。

(i) ベクトル化率を知る。

ANALYZER/SX の時間測定機能(**3.1.1** 節(a)参照)で、コンパイラオプション[4,13]に VECTOR(既定値)を指定して実行すると、プログラム単位ごとに次の情報が得られる。

- ベクトル演算率 α'(見出しV.OP.RATIOの下に表示されている)(**3.1.1** 節(a)参照)
- CPU時間 Tv

α' の値はベクトル化率 α の類似値である。Tv の大きいプログラム単位の α' の値によってだいたい下記のように判断し、小さいものについては無視してよい。

ケース1　$\alpha' \geqq 90\%$以上	ベクトル命令は十分使用されている。その効率を評価するために(iii)へ。
ケース2　$50\% \leqq \alpha' < 90\%$	ベクトル命令の使用がまだ十分でない。書き換えるべき該当箇所を知るために(ii)へ。
ケース3　$\alpha' < 50\%$	表面的な書き換えではベクトル化率の向上は難しい。ベクトル化向きのアルゴリズムの検討を要する。

SX の性能が発揮されていると言えるには、ベクトル化率が90%以上あることが望ましいが、その理由については文献[1]を参照のこと。

(ii) 実行費用の高い箇所を知る。

Tv の大きいプログラム単位(複数個あってよい)をその対象としてANALYZER/SXの詳細解析機能(**3.1.1** 節(b)参照)で実行すると、次の情報が得られる。α は α' より精度が高い。

- ベクトル化率 α
- DO ループごとのベクトル化の可否、実行費用、平均のベクトルの長さ
- 各文の実行費用

実行費用の高い DO ループをベクトル化するように解説[5]の**2** 節を参照して書き換える。

(iii) 加速率 r を調べる。

ANALYZER/SXの時間測定機能で、コンパイラオプションに NOVECTOR を指定して実行すると、プログラム単位ごとのスカラ命令だけを使用したときのCPU時間 Ts を得る。加速率 r は次の式から得られる。

$$r = \{Ts - Ts(1 - \alpha'/100)\} / \{Tv - Ts(1 - \alpha'/100)\}$$

このrから次のような判断をする。

ケース1　$r \geqq 10$倍　　効率は十分である。

ケース2　$r < 10$倍　　効率は不十分である。解説[5]の3節を参照して書き換え箇所を検討する。

1ベクトル命令当りの平均のベクトルの長さlは、(i)のリストから、次のように求める。ただし、VeおよびViはそれぞれ見出しのV.ELEMENT, V.INSTRの下に表示されている値である。

$$l = Ve/Vi$$

この値は通常はベクトルレジスタの長さから128で押えられる。DOループに関しては(ii)の“DO文輪郭リスト”でわかる。

3. ANALYZER/SX（簡易形）

3.1 概　要

ANALYZER/SXの機能には、プログラムの実行時の性能の情報を収集するための(i)動的解析と、プログラムのモジュール間の構造を解析するための(ii)静的解析がある。(i)では被測定プログラムを通常のように実行する一方で同時に解析情報を収集するのに対し、(ii)では実行しない。いずれの場合も解析結果をプリンタに出力する。2節で述べたように動的解析のほうが重要であるから、本節でも最初に動的解析について述べ、静的解析については簡単に触れることにする。

3.1.1 動的解析機能

動的解析の機能には、さらに(a)時間測定と(b)詳細解析の二つの機能があり、これらは同時に使用できない。2.3節で述べたように、通常は最初に(a)で粗い情報を収集したのち、必要に応じて(b)でより詳細な情報を収集する。いずれもプログラムの実行はCPでも可能であるが、CPではAPに比べ演算速度が遅いこと[2]、(a)についてはベクトル演算率などが出力されないこと、の理由から本稿ではAPで使用する場合に限定して述べる。

(a) 時間測定機能

この機能の目的は実行時にCPU時間を大量に消費しているプログラム単位の検出とベクトル演算率とを得ることである。出力されるリストはサマリリストだけであり、各プログラム単位の次の情報が収集できる。

- 引用回数と1回の引用の平均CPU時間。
- 実行された総命令数（＝スカラ命令数＋ベクトル命令数）　$n1$
- 実行されたベクトル命令数　$n2$
- ベクトル命令で処理された要素数　$n3$
- ベクトル演算率$\alpha' = n3/(n1 - n2 + n3)$

このベクトル演算率は一般に言われているベクトル化率に類似のものである。なお、この機能ではソースプログラムのリストを出力できず、ベクトル化情報ファイルに情報の書き込みもできない。

(b) 詳細解析機能

この機能の名称はマニュアル[16]では単に動的解析とのみ表記されているが、本稿ではとりあえずこう呼んでおく。この機能の目的は、実行費用の高い箇所を見つけるために、下記の情報をDOループ単位または実行文の単位で収集することである。プログラム全体を解析の対象にできるが、効率よく活用するために、(a)で見当をつけた特定のプログラム単位をその対象とする部分解析を使用することを勧める。実行回数が少ない、実行時間の短いプログラム単位を詳細に解析する必要はないからである。なお、ベクトル化情報ファイルに情報の書き込みもおこなう。

○サマリリスト

プログラム単位ごとの(a)のサマリリストに相当した内容。ただし、ベクトル演算率ではなくベクトル化率が出力される。

○ DO文輪郭リスト

コンパイラオプションVECTOR＝(FULLMSG)を指定した場合と同じだけのベクトル化診断情報に、平均のベクトルの長さ、各DOループの実行費用をまとめたもの。

○編集プログラムリスト

ソースプログラムの段落付けしたリスト、各文の実行費用、ベクトル化状況など。

○原始プログラムリスト

ソースプログラムリストおよびIF文の分岐に関する情報、各文の実行費用。

○実行経路リスト

プログラムが異常終了した場合に強制的に出力する。

解析時に必要なのは主に上から3つであり、サマリリストおよびDO文輪郭リストは自動的に、編集プログラムリストは既定値で自動的に出力される。

3.1.2 静的解析機能

ベクトル化率が十分に得られない場合には、プログラムの一部、あるいは全体をベクトル化に適した解法やアルゴリズムに書き直す必要がある。このような場合、プログラムのモジュール構造、手続きや共通データの相互参照情報などを得ることができれば作業を効率よく行うことができる。静的解析機能は、このようなプログラムの静的な構造情報を解析し、プログラムの大幅な変更作業時に役立つ各種の情報を出力する(**3.2.2**節参照)。

3.2 $ SXANAL文

ANALYZER/SXの起動には$ SXANAL[15]文を用いる。下線は運用の既定値を意味する。動的解析ではDYNAMIC(既定値)、静的解析ではSTATICを指定する。

3.2.1 動的解析

時間測定機能を使用するときには PROGTIME を指定し、詳細解析機能を使用するときには起動させるためのパラメータは存在しない。このように両機能の区別はパラメータ PROGTIME の有無でおこなう。その他のパラメータは下記のとおりであり、とくに断りのないものは使用する機能に無関係である。

○ FMTLIST＝*p*, NOFMTLIST

FMTLIST では編集プログラムリストを出力し、NOFMTLIST では出力しない。*p* は各実行文の実行費用などの表示方法を選択し、既定値では COST である。PROGTIME 指定時には使用できない。

○ RUN＝AP

被測定プログラムの実行を AP 上で行う。既定値では CP で実行される。

○ CPARAMAP＝'*plist*'

被測定プログラムの実行を AP でおこなうときに、コンパイラオプションの並びを *plist* に一つの空白またはコンマで区切り指定する。AUTODBL, NOSOURCE オプションを指定してはならない。

○ PROG＝(*pglist*)

部分解析をおこなうプログラム単位名の並びを *pglist* に一つのコンマで区切り指定する。PROGTIME 指定時には使用できない。

○ SOURCE, NOSOURCE

SOURCE では **3.1.1** 節 (b) の原始プログラムリストを出力し、NOSOURCE では出力しない。詳細解析機能で IF 文の分岐情報が必要なときには SOURCE を指定しなければならない。PROGTIME 指定時には使用できない。

○ EXECLIST, NOEXECLIST

被測定プログラムの実行結果のうち、プリンタ出力のものを通常どおりに出力するとき EXECLIST を指定し、出力しないとき NOEXECLIST を指定する。

○ ＊LIBRARY＝AA, ＊LIBRARY＝MA, ＊LIBRARY＝AA/MA

ASL/SX または MATHLIB/SX を使用するときに指定する[4,21]。AA は AP 用の ASL/SX を、MA は AP 用の MATHLIB/SX を意味する。

次のオプションは ANALYZER/SX とは直接関係はないが、SX 固有の理由[9]で必要となる場合がある。

○ ENTRY＝*name*

主プログラムに対して PROGRAM 文で FTMAIN 以外の名前 *name* を与えているときに指

定する。

3.2.2 静的解析

以下のものは既定値ではすべて出力されないので、必要であれば指定しなければならない。

- ○ XREF　　相互参照リストを出力する。
- ○ STRUCT　　プログラム構成リストを出力する。
- ○ PROGXREF　　プログラム相互参照リストを出力する。
- ○ ARGLIST　　引数対応リストを出力する。
- ○ COMXREF　　共通ブロック相互参照リストを出力する。

3.3 起動法

ジョブ構成は基本的には通常のジョブの場合と差がない。以下、利用者番号を A60000 、支払いコードを A (校費)、利用者番号のパスワードを pass-word と表記することにする。

3.3.1 動的解析 (時間測定機能)

通常のジョブの実行の場合と比較すると、次のような差がある[15]。

- ○ $ FRT77 文、$ GO文は不要である。$ SXANAL 文 (**3.2** 節参照) で自動的に展開される。
- ○ ASL/SXなどのライブラリを使用する場合は、$ SXANAL文にパラメータを指定する。
- ○ JCLファイルの中にデータを記述する場合 (これをSYSINデータとよぶ) には $ DATA 文と $ ENDCOPY 文でこのデータ全体を囲まなければならない。

次のような被測定プログラムで時間測定機能を APで使用するときの JCL を**図2**に示す。図中の下線は既定値のため省略できることを示す。

- ○ CPU時間の上限は 600 秒である。(このことからジョブクラスはUとなる。)
- ○ ソースプログラムはパーマネントファイルの SRCFILE から入力する。主プログラムに PROGRAM TESTで主プログラム名TESTを与えている。
- ○ 入力データはSYSINファイルからREAD (5, …) とパーマネントファイルから READ (10, …) でおこなう。
- ○ 日本電気提供のライブラリ ASL/SXを使用する。

```
    $    JOB        A60000;A$pass-word,U
    $    SX         TYPE=A6,CPTIME=600
①  { $   SXANAL     DYNAMIC,PROGTIME,RUN=AP,
    $    ETC        ENTRY=TEST,*LIBRARY=AA
②   $    PRMFL      S*,R,S,A60000/SRCFILE
③   $    DATA       05,,COPY
④   1,2,3
⑤   $    ENDCOPY
⑥   $    PRMFL      10,R,S,A60000/DATAFILE
    $    ENDJOB
```

図2　ANALYZER/SXの時間測定機能（簡易形）[15]

【説明】

①ANALYZER/SXを起動する。

②ソースプログラムの格納されているパーマネントファイルの指定。$ SELECTA文またはSYSINファイルにソースプログラムを記述する場合には、その前後に次の JCL が必要である。したがって $ PRMFL 文を使用するのがよい。

```
$    DATA      S*,,COPY
$    SELECTA   ファイル記述子
$    ENDCOPY
```

③, ⑤SYSINデータがある場合に必要となる。④SYSINデータ。⑥標準のジョブの場合と同様。

3.3.2　動的解析（詳細解析機能）

上記と同一のプログラムで詳細解析を AP で使用し、部分解析をサブルーチン SUB1 と SUB3 に対しておこない、ベクトル化情報ファイルを作成するときには次のようにする。**図2**の①のPROGTIMEを取り除き、⑥の次に⑦を追加する。

```
①  { $   SXANAL    DYNAMIC,FMTLIST,RUN=AP,PROG=(TEST,SUB1,SUB3),
    $    ETC       ENTRY=TEST,*LIBRARY=AA
⑦   $    PRMFL     VF,W,S,A60000/VECTFILE
```

⑦はベクトル化情報ファイルの指定であり、ファイルコードは VF である。予め順編成のパーマネントファイルを作成しておかなければならない。⑦がない場合にはベクトル化情報は作成されない。

3.3.3　静的解析

静的解析で被測定プログラム全体の、構成リスト、共通ブロック参照リスト、引数対応リストを出力するには、**図3**のようにする。

```
$    JOB       A60000 ; A $ pass-word, S
$    SX        TYPE=A6
$    SXANAL    STATIC, STRUCT, COMXREF, ARGLIST
$    PRMFL     S*, R, S, A60000/SRCFILE
$    ENDJOB
```

図3　ANALYZER/SXの静的解析（簡易形）[15]

- ジョブクラスにはCPのものを使用する。
- $ SXANAL文のSTATIC以外のオプションを省略すると、何の出力も得られないことに注意すること。

3.4　注意事項

- $ JOB文のジョブクラスが意味するプロセッサと$ SXANAL文のRUNパラメータで指定するプロセッサは同一でなければならない。
- プログラムにもよるが実行時のCPU時間が詳細解析機能では3割程度増加する。時間測定機能では通常1割以下であり、無視できる程度である。
- 部分解析では次の制約がある。
  - 主プログラムを必ず指定すること。
  - プログラムを終了させるSTOP文を含むプログラム単位は、必ず指定すること。
  - プログラム単位の数は最大10個である。

4. ANALYZER/SX（基本形）

基本形においても、ANALYZER/SXの機能は3節で述べたことと同様である。TSS処理では、3節に述べた静的解析および詳細解析のCP版に相当するものが使用できるが、実行時の効率および出力量を考慮するとバッチ処理で使用するのがよい。起動法は簡易形とは異なる。

4.1　$ FANALSX文

ANALYZER/SX を起動するには $ FANALSX[16] 文を用いる。下記に示すもの以外は、$ SXANAL 文のオプションと同じである。オプションを一つの空白または一つのコンマで区切り、任意の順序で指定する。マニュアル[16]では下記以外にCPU時間を指定するためのCPTIMEパラメータがあるが、当センターの運用では$ SX文で指定しなければならない。

- ASSIGNi　　基本形の$ ASSIGN文に対応する。
- INLIB および MEMBER　　基本形の$ FORT77 SX文のINLIBおよびMEMBERに対応する。
- VECTFILE　　$ FANALSX固有のもの。ベクトル化情報ファイルの指定。
- LINLIBi　　基本形の$ LINKのINLIBiパラメータに対応する。

4.2 起動法

$ FANALSXで、バッチ処理に必要な他のJCLもすべてあわせて指定する。ただし、$ JOB文、$ SX文、$ ENDJOB文、SYSIN で指定するデータ関係を除外する。前述の**3.3.2**節と同一のことを基本形で行うと、**図4**に示すようになる。ただし、基本形であるからソースプログラムおよびデータのファイルはSX側のファイルに転送済み[6,8]でなければならない。

```
$          JOB      A60000;A$pass-word,U
$          SX       TYPE=SX,CPTIME=600
$FANALSX DYNAMIC, INLIB=(A60000.IL)
     MEMBER=(SRCFILE), FMTLIST, RUN=AP
     ENTRY=TEST
     LINLIB=(LIB.ASLAP)
     ASSIGN1=(FF10,A60000.DATAFILE)
     ASSIGN2=(SIN,COM1,FILESTAT=SYSIN)
     VECTFILE=(A60000.VECTFILE) ;
$INPUT COM1 ;
1,2,3
$ENDINPUT ;
$        ENDJOB
```

図4 ANALYZER/SXの詳細解析機能（基本形）

【説明】

○ $ FANALSX文以外の基本形のJCLについては解説[9]を参照のこと。

○基本形であるからTYPE＝SXは必須である。

○ソースプログラムは統合ライブラリ名 IL の中にメンバー名 SRCFILE、データは順編成でファイル名DATAFILEで登録している。

○データファイルの指定は本来は $ ASSIGN文で行うが、$ FANALSX文ではASSIGNiとしてiに1から順に番号を付加して識別する。

○ベクトル化情報ファイルの属性とその容量については**2.1**節を参照のこと。

5. VECTORIZER/SX

5.1 概　要

VECTORIZER/SXでは、ベクトル化情報ファイルとそれを作成した時点のソースファイルを入力として、性能向上のためのソースプログラムの編集作業をTSS 端末の画面上で効果的におこなう。ベクトル化情報の内容は ANALYZER/SX の動的解析の詳細解析機能で作成する（**3.1.1**節(b)参照）。**図5**は画面の種類とその遷移を端末のキーおよびサブコマンドで示したものである。各画面は番号の順に階層構造をなし、後述の起動コマンドをタイプインすると、図の①が表示される。これ以後は簡易形でも基本形でも同じである。以下、キーを押して番号の順に進めば書き

換えに必要な情報が表示されるので、これを参考にしながら最後の④の画面で SX の画面エディタのコマンドを用いて編集作業をする。なお、表示される情報は ANALYZER/SX でのリスト類と同様である。

編集の単位はプログラム単位であり、変更したいプログラム単位を選択すると、パーマネントファイルからその内容が作業用ファイル（エディタのバッファ）にコピーされる。編集はこのバッファ上でおこなわれ、ベクトル化診断をおこなう際のコンパイラの入力にもこのバッファの内容が使用される。Wコマンドがタイプインされるとパーマネントファイルへ書き戻されるが、この時点でソースプログラムは変更されてしまうため、これ以後はそのプログラム単位の（VECTORIZER/SX を使用した）編集は不可能となるので、注意が必要である。

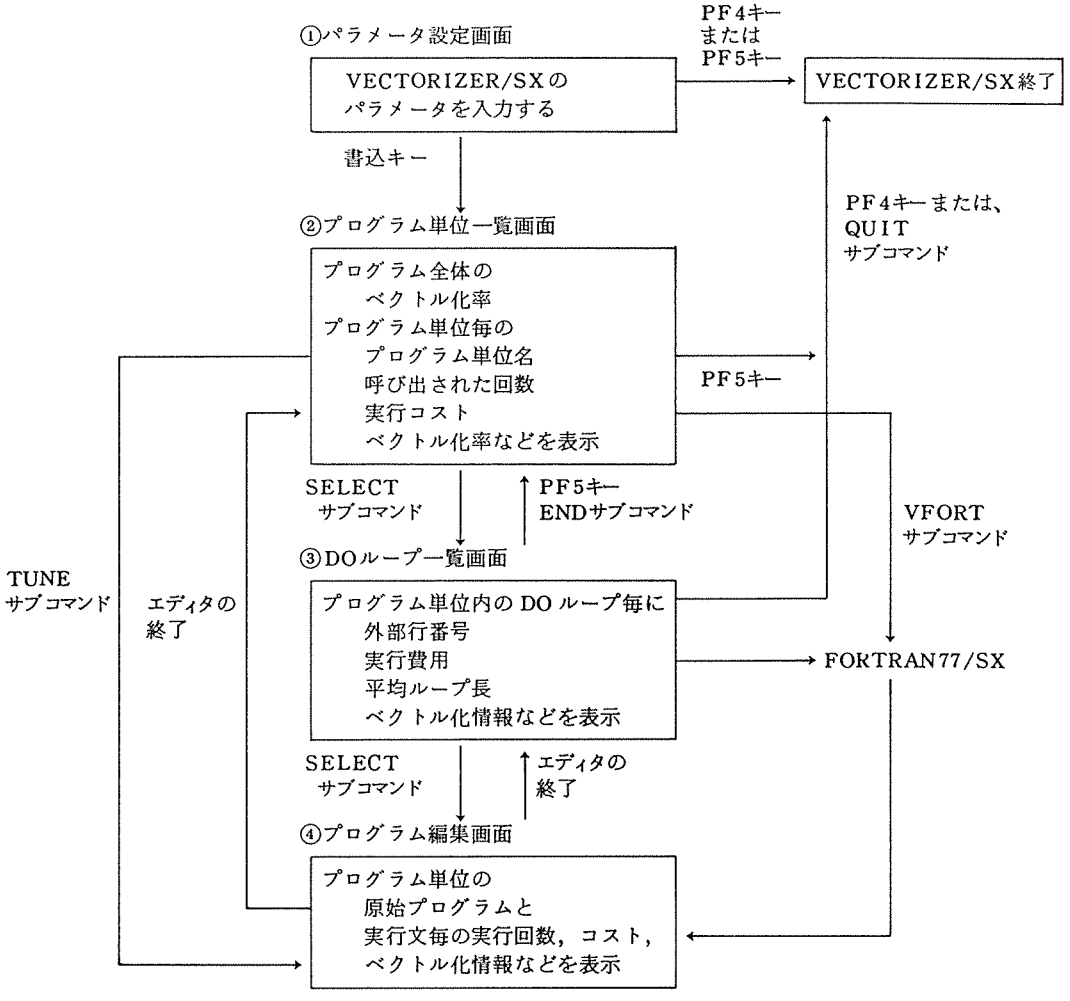

図 5　画面遷移と画面操作

5.2 操作法

VECTORIZER/SXを起動するためのコマンドは次のとおりである。

〔簡易形〕

SYSTEM選択レベルまたはビルドモードで、次のコマンドをタイプインする[15]。

SXVECT　ベクトル化情報ファイル名　ソースファイル名

〔基本形〕

SXTSSコマンドでSXのTSSに入ったあと、次のコマンドをタイプインする[17]。

VECTORIZE　ベクトル化情報ファイル名

基本形では、ベクトル化情報ファイルの先頭にソースプログラムファイルの名前を記録しているのでその指定が不要である。

表2と**表3**にプログラム一覧画面とDOループ一覧画面で用いるサブコマンドとキーを示す。コマンドの下線は省略形を意味する。

5.3 注意事項

- プログラム編集画面でSXのTSSコマンドを実行することはできない。
- VECTORIZER/SXのVFORTコマンドでFORTRAN77/SXコンパイラを呼び出すとき、そのオプションは利用者が指定できないため運用の既定値となる。

表2　プログラム単位一覧画面

サブコマンドとキー	意　味
SELECT	プログラム単位を選択してDOループ一覧画面へ
WRITE	プログラム単位をソースファイルに書き戻す
TUNE	プログラム単位を選択してプログラム編集画面へ
VFORT	FORTRAN77/SXコンパイラを呼び出した後プログラム編集画面へ
QUIT	VECTORIZER/SXの終了
END	VECTORIZER/SXの終了
書込キー	順方向に1画面分，画面が送られる
Ⅲキー 注)	逆方向に1画面分，画面が送られる
PF4キー	VECTORIZER/SXの終了
PF5キー	VECTORIZER/SXの終了
PF6キー	画面が壊れたときに現在の画面を再表示する

注）6950Nでは、LOCATEコマンドを使用する。

表3　DOループ一覧画面

サブコマンドとキー	意　　味
SELECT	DOループを選択してプログラム編集画面へ
WRITE	プログラム単位をソースファイルに書き戻す
VFORT	FORTRAN77／SXコンパイラを呼び出した後プログラム編集画面へ
QUIT	VECTORIZER/SXの終了
END	プログラム単位一覧画面へ
書込キー	順方向に1画面分，画面が送られる
Ⅲキー 注)	逆方向に1画面分，画面が送られる
PF4キー	VECTORIZER/SXの終了
PF5キー	プログラム単位一覧画面へ
PF6キー	画面が壊れたときに現在の画面を再表示する

注）6950Nでは、LOCATEコマンドを使用する。

6. OPTIMIZER

6.1 概　要

OPTIMIZERは、Fortran77言語で記述されたソースプログラムにコンパイラでおこなうことのできないさまざまな最適化を図り、新たなソースプログラムを生成し性能向上を計る汎用のサービスプログラムである。

OPTIMIZERは、次のような最適化を行う。なお、実際の展開形は必ずしもこの通りではない。

以下、プログラムの例において、左側はもとの形を、右側はその展開形を示す。

①外部手続きのインライン展開

DOループ中で外部手続きであるサブルーチン副プログラムや関数副プログラムを引用している所に、これらの外部手続きを組込む。

```
   DO  10   I=1,N
10 X(I)=Y(I)*FNORM(Z(I))
      ⋮

   FUNCTION  FNORM(X)
   PARAMETER  (A=0.398794228)
   FNORM=A*EXP(-0.5*X**2)
   RETURN
   END
```

```
   DO  10   I=1,N
10 X(I)=Y(I)*A*EXP(-0.5*Z(I)**2)
```

②DOループ入れ替えによる最適化

DOループが密な多重ループの場合、内側と外側のDOループを入れ替えメモリ参照の効率化をはかる。

```
      DO 10  I=1,M                    DO 10  J=1,N
      DO 20  J=1,N                    DO 20  I=1,M
      A(I,J)=B(I,J)+C(I,J)            A(I,J)=B(I,J)+C(I,J)
   20 CONTINUE                     20 CONTINUE
   10 CONTINUE                     10 CONTINUE
```

③DOループ展開による最適化

DOループ本体を2倍に展開して、DOループの繰返し回数を半分にする。

```
      DO 10  J=1,1000                 DO 10  I=1,1000,2
      DY(J)=DY(J)+DA(J)*D             DY(J)=DY(J)+DA(J)*D
   10 CONTINUE                        DY(J+1)=DY(J+1)+DA(J+1)*D
                                   10 CONTINUE
```

④演算評価順序の変更

DOループ本体中の式の中から、不変項を捜して可換律により変換し、不変項からなる式(不変式)を生成してDOループの外へ移動する。

```
      DO 10  I=1,1000                 DP9999=C+2.0
      A(I)=C+2.0+B(I)                 DP9998=F*3.0
      D(I)=F*E(I)*3.0                 DO 10  I=1,1000
   10 CONTINUE                        A(I)=B(I)+DP9999
                                      D(I)=E(I)*DP9998
                                   10 CONTINUE
```

OPTIMIZERはSX専用のサービスプログラムではないために、最適化を行うとかえって逆効果になる場合もある。たとえば③の項目は、汎用コンピュータでは通常CPU時間の短縮につながるが、SXでは、ベクトルの長さが短くなるなどの理由でかえって遅くなってしまうことがある。反対に①の項目は、例のように今までベクトル化できなかったDOループがベクトル化可能になり、大きな効果を生むこともある。外部サブルーチン、あるいは外部関数がDOループ中にあるためにベクトル化ができないような場合は、OPTIMIZERで展開可能かどうか検討してみるとよい。

上記のような最適化をOPTIMIZERが行うためには、ある条件を満たしてなければならないが、これについては、マニュアルを参照していただきたい。

6.2 操作法

OPTIMIZERはCPで実行され、バッチ処理では$FOPTIMIZE文、TSSではFOPTIMIZEコマンドで起動する。以下に、外部手続きをインライン展開する場合の例を上げる。統合ライブラリA60000.ILの中のメンバーSRCFILEで引用している外部手続き名 FUNC をインライン展開

し、新しくできたソースプログラムをSRCFILEEに出力する。

［バッチ］

```
$          JOB       A60000;A$pass-word,S
$          SX        TYPE=SX
$FOPTIMIZE OUTLIB=(A60000.IL)
     COMFILE=COM1
     INLINE=(SRCFILE(FUNC));
$INPUT COM1 ;
//OPT NAME=SRCFILE,NEWNAME=SRCFILEE
$ENDINPUT ;
$          ENDJOB
```

［ＴＳＳ］

```
FOPTIMIZE IL(SRCFILE) OUTSOURCE=&(SRCFILEE) INLINE=(SRCFILE(FUNC))
```

主なパラメータは、以下のとおりである。既定値では機能は抑制されているので、必要であれば指定しなければならない。

- INLINE= サブファイル名（プログラム単位名）

  外部手続きのインライン展開の最適化を指定するとともに、被引用側のプログラム単位名と、そのソースプログラムが存在するサブファイル名を指定する。引用側のソースプログラムは、INLIBあるいはOUTLIBパラメータと//OPTコマンドあるいはMEMBERパラメータで指定する。

- DOCHG　　ＤＯループの入れ替えの最適化をする。
- EVALCHG　ＤＯループ中の演算評価順序の最適化をする。
- UNROLL　　ＤＯループの展開の最適化をする。

7. おわりに

いままでACOS1000あるいはHFPで実行していたプログラムを、そのままSXで実行しただけでも数倍、十数倍の性能を引き出すことができるが、プログラムをほんの数行変更するだけでさらに数倍の性能になることもある。このためにも本稿で述べてきた性能向上支援ツールを上手に使いこなすことで、どの部分を変更すればよいかを見つけたり、その部分の変更作業を効率よく進めていくことができる。むろんこれらを用いなくても、SXでジョブを実行することは可能であり、一方、プログラムの書き換えもコンパイラが出力するベクトル化診断メッセージを見ながらACOS1000のエディタでできることを申し添えておく。

本稿がSXを使用するうえで、利用者の方々のお役に立てば幸いである。

参　考　文　献

1) 渡辺，近藤，端山，大中，藤井：スーパーコンピュータ SX－1の概要(1)，大阪大学大型計算機センターニュース，Vol.15, No.4 (1986).

2) 藤井：スーパーコンピュータ SX－1の概要(2)，大阪大学大型計算機センターニュース，Vol.15, No.4 (1986).

3) 大中，後藤：SX FORTRAN77 概要(1)，大阪大学大型計算機センターニュース，Vol.16, No.1 (1986).

4) 後藤，大中：SX FORTRAN77 概要(2)，大阪大学大型計算機センターニュース，Vol.16, No.1 (1986).

5) 片山，河原，大中：FORTRAN77/SXにおける高速化技法，大阪大学大型計算機センターニュース，Vol.16, No.1 (1986).

6) 馬野：スーパーコンピュータ SX－1のタイム・シェアリング・システム ATSS-AFの使い方（その１），大阪大学大型計算機センターニュース，Vol.16, No.1 (1986).

7) 馬野：スーパーコンピュータ SX－1のタイム・シェアリング・システム ATSS-AFの使い方（その２），大阪大学大型計算機センターニュース，Vol.16, No.2 (1986).

8) 多喜：ACOS-1000とSX－1とのファイル転送について，大阪大学大型計算機センターニュース，Vol.16, No.1 (1986).

9) 後藤：スーパーコンピュータ SX－1 のジョブ制御言語入門，大阪大学大型計算機センターニュース，Vol.16, No.2 (1986).

10) 片山：第1回 SXプログラミング研究会報告，大阪大学大型計算機センターニュース，Vol.16, No.2 (1986).

11) 萬，花村，宮平：SXシステムにおける有効なプログラミング例，大阪大学大型計算機センターニュース，Vol.16, No.2 (1986).

12) GGB11－1　FORTRAN77言語説明書，日本電気(1985).

13) GGB12－2　FORTRAN77, 77/SX プログラミング手引書，日本電気(1986).

14) GDA12－2　SXCP解説書，日本電気(1985).

15) GJF11－1　MSF－6利用説明書，日本電気(1986).

16) GGB14－2　ANALYZER/SX説明書，日本電気(1986).

17) GGB13－2　VECTORIZER/SX説明書，日本電気(1986).

18) GGB15－2　OPTIMIZER説明書，日本電気(1986).

19) GED12－2　ATSS-AF 画面エディタ説明書，日本電気(1985).

20) GCA13－2　エラーメッセージハンドブック，日本電気(1986).

21) 大阪大学大型計算機センター速報，No.134 (1986).

22) 大阪大学大型計算機センター速報，No.135 (1986).